GPU-accelerated Fourier-continuation solvers and physically exact computational boundary conditions for wave scattering problems

Thesis by

Tim Elling

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2013

(Defended July 6, 2012)

Acknowledgements

I owe a great debt of gratitude to Professor Oscar P. Bruno, who served not only as my advisor, but also as an excellent teacher, mentor, and friend. Over the past years he has shared with me his deep wisdom, great enthusiasm, and endless patience. Most importantly, he has always challenged my assumptions, not only of my work but also of myself. I would like to thank all of those whose help and support has made this thesis possible: Nathan Albin helped me with many discussions concerning the FC(Gram) method, and in particular its parallel implementation; David Hoch provided assistance to understand the details of the boundary condition work; Professor Tim Warburton graciously provided a specially modified version of his MIDG code for the hybrid solver, as well as his sage advice regarding the stability of time-domain methods in general. Edwin Jimenez helped me improve my mathematical writing, in particular in the preparation of this text. Conversations with Professor Catalin Turc both furthered my understanding of the equivalent source algorithm and encouraged my personal intellectual growth. I appreciate the continuous efforts of the various administrative staff at Caltech for their kindness and patience, in particular Sheila Shull, Sydney Garstang, and Carmen Sirois in the CMS department, and Tess Legaspi and Gloria Brewster in the registrar's office. The specialized GPU implementations in this work would not have been possible without the help of our system administrators Chad Schmutzer and Will Yardley, who supported my many esoteric hardware and software requests.

I would also like to thank Randy Paffenroth for reminding me early on that math can and should be fun. I am grateful to Mason Porter, who, when I was a Caltech undergraduate, taught me more about applied mathematics in one day than I ever had thought possible. I would like to thank my family—my sister Jennifer Tucker, brother Rob Elling, and my parents Bob and Lois Elling—for helping me become the person I am today. I would especially like to thank my grandparents: Bessie Elling for teaching me to be strong, Bob Elling, Sr., for teaching me to be curious, Esther Robinson for teaching me to be decent,

and Herman Robinson for first introducing me to computers on his Apple][+, which I still own today.

I would like to express my most profound gratitude to Christina Chow, my loving wife, without whose tireless support I may never have dared partake in this endeavor. Her boundless energy and encouragement have given me the drive needed to face the many late nights and long weekends that fueled this research.

Finally, I wish to thank my son, Tycho Elling, whose smiles make the world go round.

Abstract

Many important engineering problems, ranging from antenna design to seismic imaging, require the numerical solution of problems of time-domain propagation and scattering of acoustic, electromagnetic, elastic waves, etc. These problems present several key difficulties, including numerical dispersion, the need for computational boundary conditions, and the extensive computational cost that arises from the extremely large number of unknowns that are often required for adequate spatial resolution of the underlying three-dimensional space. In this thesis a new class of numerical methods is developed. Based on the recently introduced Fourier continuation (FC) methodology (which eliminates the Gibbs phenomenon and thus facilitates accurate Fourier expansion of nonperiodic functions), these new methods enable fast spectral solution of wave propagation problems in the time domain. In particular, unlike finite difference or finite element approaches, these methods are very nearly dispersionless—a highly desirable property indeed, which guarantees that fixed numbers of points per wavelength suffice to solve problems of arbitrarily large extent. This thesis further puts forth the mathematical and algorithmic elements necessary to produce highly scalable implementations of these algorithms in challenging parallel computing environments—such as those arising in GPU architectures—while preserving their useful properties regarding convergence and dispersion.

Additionally, this thesis develops a fast method for evaluation of computational boundary conditions which is based on Kirchhoff's integral formula in conjunction with the FC methodology and an accelerated equivalent source integration method introduced recently for solution of integral equation problems. The combination of these ideas gives rise to a physically exact radiating boundary condition that is nonlocal but fast. The only known alternatives that provide all three of these features are only applicable to a highly restrictive class of domains such as spheres or cylinders, whereas the Kirchhoff-based approach considered here only requires a bounded domain with nonvanishing thickness. As is the

case with the FC scattering solvers mentioned above, the boundary-conditions algorithm is modified into a formulation that admits efficient implementation in GPU and other parallel infrastructures.

Finally, this thesis illustrates the character of the newly developed algorithms, in both GPU and parallel CPU infrastructures, with a variety of numerical examples. In particular, it is shown that the GPU implementations result in thirty- to fiftyfold speedups over the corresponding single CPU implementations. An extension of the boundary-condition algorithm, further, is demonstrated, which enables for propagation of time-domain solutions over arbitrarily large spans of empty space at essentially null computational cost. Finally, a hybridization of the FC and boundary condition algorithm is presented, which is also part of this thesis work, and which provides an interface of the newly developed algorithms with legacy finite-element representations of geometries and engineering structures. Thus, combining spectral and classical PDE solvers and propagation methods with novel GPU and parallel CPU implementations, this thesis demonstrates a computational capability that enables solution, in novel computational architectures, of some of the most challenging problems in the broad field of computational wave propagation and scattering.

Contents

Part I

Preliminaries

Chapter 1

Introduction

Many physical processes of considerable scientific and engineering importance, ranging from acoustics to electromagnetics, elasticity, and seismic phenomena, concern wave propagation and scattering. Acoustic scattering (deflection of pressure waves by particles or bodies in a medium) is the prototype for many of these physical processes. The precise mathematical statement of wave interaction amounts to a system of conservation laws which is often formulated as a hyperbolic system of partial differential equations relating spatial and temporal derivatives of a perturbation in the underlying medium (in the case of acoustic waves) or in free space (in the case of electromagnetic waves).

This thesis is primarily concerned with exterior acoustic scattering problems. Exterior problems of acoustic scattering involve the propagation of acoustic waves in an unbounded medium until they impinge upon one or more obstacles. Since the waves can be described as solutions of linear partial differential equations, the total solution u is given by the relation $u = u_i + u_s$ in terms of its two constituent components, the unimpeded incident wave u_i, and the wave u_s that is scattered from the obstacle(s). The physical nature of a scatterer is typically quantified as either "sound-soft" (resp. "sound-hard"), implying that $u = 0$, and thus $u_s = -u_i$ (resp. $\frac{\partial u}{\partial n} = 0$, and thus $\frac{\partial u_s}{\partial n} = -\frac{\partial u_i}{\partial n}$) on the scattering surface. More generally, so-called impedance boundary conditions, which amount to a linear combination of these two, are sometimes relevant. A complete set of equations for a prototypical exterior scattering problem, arising from a field u_i incident upon a volume Θ

with impedance boundary Γ, may be expressed in the form

$$\frac{1}{c^2}\frac{\partial^2}{\partial t^2}u_s - \Delta u_s = f(x,t) \qquad \text{in } \mathbb{R}^3 \setminus \Theta \times (0,\infty) \qquad (1.1)$$

$$u_s(x,0) = u_0(x) \qquad x \in \mathbb{R}^3 \setminus \Theta \qquad (1.2)$$

$$\frac{\partial}{\partial t}u_s(x,0) = \dot{u}_0(x) \qquad x \in \mathbb{R}^3 \setminus \Theta \qquad (1.3)$$

$$au_s + bn \cdot \nabla u_s = -au_i - bn \cdot \nabla u_i \qquad (x,t) \in \Gamma \times (0,\infty) \qquad (1.4)$$

$$\lim_{r\to\infty} r\left(\frac{\partial}{\partial r}u_s + \frac{1}{c}\frac{\partial}{\partial t}u_s\right) = 0 \qquad r = \|x\|_2, \qquad (1.5)$$

where $f(x,t)$ is a compactly supported source and the necessary initial conditions are given by $u_0(x)$ and $\dot{u}_0(x)$.

In order for the scattered field u_s to be uniquely determined it is necessary to require a certain condition of *radiation at infinity* (equation (1.5)) which was first introduced by Sommerfeld [70]. The direct scattering problem, then, is to determine this radiating solution u_s given knowledge of the partial differential equation, the incidence u_i, and appropriate conditions (based on the physical properties under consideration) at the boundary of the scatterer(s). The numerical solution of exterior scattering problems poses unique challenges not present in bounded domains. Unbounded regions are usually truncated to render them suitable for numerical simulation, but doing so necessitates the specification of boundary conditions at the *artificial* boundary of the computational domain, which relate in subtle ways to the Sommerfeld radiation condition.

This work presents a combined strategy for the solution of time-domain acoustic scattering problems that allows for highly efficient execution in parallel computing environments, with a particular emphasis on graphics processing units (GPUs). An interior solver for hyperbolic PDEs is described using the accelerated Fourier continuation-Gram polynomial (FC(Gram)) methodology [18, 53] in Chapter 3 and 5, in particular extending the work of [1] to support a very fine-grained level of task parallelism, as described in Chapter 4. This allows for significant gains in performance by taking advantage of GPUs, with GPU-to-CPU improvement ratios of a factor of no less than 28. In addition, a variant of the physically exact approach for evaluation of computational boundary conditions described in [42] is presented in Chapters 6 and 7, taking advantage of the more-compact FC(Gram) continuation strategy (cf. Section 3.3 in [42]). These boundary conditions are adapted for

GPU execution, reducing the already-fast CPU boundary condition computing times by a factor of 50 on average in the GPU implementation, as shown in Section 9.3. Perhaps most importantly, this work provides the first efficient application of such an exact, radiating boundary condition to a nonconvex computational domain, and it in fact demonstrates a framework for the solution of multiple-scattering problems in Section 9.5 whose computing cost per time-step *remains independent of the distance between the scattering surfaces*!

The remainder of this chapter presents a literature review on topics concerning the two main original contributions in this thesis, namely 1) Time-domain solvers for PDE problems on unbounded domains by means of numerical algorithms on bounded computational regions, including a corresponding discussion of various numerical methods for evaluation of computational boundary conditions, (Section 1.1); as well as 2) Implementation in modern computing hardware leading to efficient, scalable methods for solution of computational science problems (Section 1.2).

1.1 Time domain methods and applications

1.1.1 Numerical PDE solvers

One of the oldest and best-known approaches for the numerical solution of time-domain wave propagation is the Yee scheme [78], also known as the finite-difference time-domain (FDTD) method. The FDTD scheme is appealing for its simplicity, and still sees broad commercial and scientific applicability, even though it is only accurate to second-order in both space and time. As is well known and evidenced, for example, in [20] (and shown in Section 9.1 of this thesis), a second-order FDTD method typically requires at least 96 points *per wavelength*, over a computational domain only 16 wavelengths across, in order to achieve a numerical accuracy of even 1%. Furthermore, as the acoustic size of the domain grows, the per-wavelength discretization must be increased in order to preserve accuracy, resulting in a requisite discretization that grows *super-linearly* with respect to the volume of the domain. Clearly, efficient solution of large scattering problems requires development of higher-order numerical methods. A thorough, modern accounting of the FDTD approach can be found in [74].

A description of a variety of high-order finite-difference methods for the solution of hyperbolic systems can be found in [50, 58]. In such methods derivatives are approximated as

linear combinations of neighboring sample points on a regular lattice, with coefficients (or "stencil") chosen in such a way that the truncation error, as evaluated through consideration of a relevant Taylor series expansion, tends to zero with a desired power of the mesh size. High-order FD schemes require increasingly wide stencils, posing difficulty near the boundary of the domain. So-called Padé or compact schemes, such as presented in [48, 61], partially alleviate this difficulty by posing an implicit equation for the derivatives, requiring solution of a banded linear system at each iteration. In either case, explicit or implicit FD, some accuracy is typically sacrificed near the computational boundary.

In order to model conservation laws directly, finite volume methods (FVM) [49], which are applicable to structured and unstructured meshes alike, produce the numerical solution via evaluation of numerical fluxes. By avoiding a formulation in terms of numerically-computed derivatives, FVMs are especially robust in the face of discontinuous solutions or "shocks". Unfortunately, high-order convergence in FVMs is difficult to achieve, and often involves a nonlinear flux-limiter such as in [51].

The finite element method (FEM), within the broader class of Galerkin methods, replace the continuous PDE with a discrete weak-formulation. Originally developed for elliptic [63] and hyperbolic problems in diffusion, elasticity and structural analysis [77] and formalized in [72], this approach proceeds by dividing the computational domain into a number of geometric elements, over which the solution is represented as some linear combination of a set of piecewise-continuous basis functions. Perhaps the chief advantage of the FEM is its applicability to unstructured meshes, allowing for a broad class of computational domains to be discretized with relative ease. Evolving a time-dependent hyperbolic problem forward in time in some cases requires the solution of a sparse linear system at each time-step. Alternatively, the discontinuous Galerkin finite element methods [39] (DG-FEM) impose no conditions on the continuity between elements, allowing for an explicit update rule after the computation of a numerical flux term similar to that which appears in FVMs, at the expense of a greater number of numerical degrees of freedom.

Spectral methods provide unparalleled convergence and minimal (or zero) numerical dispersion, and are perhaps the most appealing for problems of linear acoustic waves, but also the most restrictive in their applicability. Pseudospectral time-domain methods take advantage of either discrete Fourier or Chebyshev expansions to describe the solution in space, a discussion of which may be found in [74]. Fourier collocation methods produce

spectrally-convergent and dispersionless solutions, but may only be applied to a highly restrictive class of domains—most commonly rectangular parallelepipeds exhibiting periodic boundary conditions (possibly under even- or odd-extension). Chebyshev collocation methods [10], on the other hand, either require stringent CFL conditions, due to refinement near the boundaries, or must be used in the context of implicit solvers, imposing the cost of solving a linear system every time-step.

1.1.2 Computational boundary conditions

For all of the numerical approaches discussed above, accurate treatment of an unbounded physical domain with a finite, bounded computational domain requires the specification of appropriate boundary conditions at the new (artificial) boundary. One famous approach, proposed first in [52] and later expanded in [26, 27], is applied in the form of a pseudo-differential operator based on increasingly accurate local approximations. This was later adapted in [57] for the case of electromagnetic waves. Similar local asymptotic approximations generalizing this framework are presented by [8]. Another local boundary operator is proposed in [41] that is exact for plane waves traveling in a number of discrete directions. Yet another related approach described in [45] constructs a pseudodifferential operator that is perfectly absorbing for solutions traveling at a specified group velocity. One of the most broadly used methods today involves the construction of a "perfectly matched layer" (PML), first proposed for scattering problems in electromagnetism by [9]. This artificial layer acts as an absorbing media while producing minimal internal reflections, if the PML is taken to be sufficiently large. A more complete theoretical understanding of the PML was established in [21], and further improvements to the approach are developed in [22, 64]. The "complete radiation boundary condition", or CRBC, is a recently introduced local method that reduces the impact of long-time errors on the solution is described in [36].

In contrast to the methods outlined above, which attempt to control the local behavior of the solution at the computational boundary by approximating a nonreflecting or absorbing interface, a family of properly convergent methods exist based on the exact, global behavior of the radiating solution. A solution constructed through the combination of Dirichlet and Neumann operators is proposed in [67], but requires costly additional interior evaluations. An exact condition based on Kirchhoff's integral formula and applicable to very general geometries was originally proposed by [75]. First implemented by [31] for linear acoustics,

and later adapted by [37] to electromagnetics, the historical difficulty in this otherwise powerful approach lies in the costly evaluation of the integral representations of the solution, dominating the computational time of the interior solvers. A fast alternative based on Fourier and Laplace transforms was developed by [3, 4], but loses geometric generality, as it is only applicable to simple (e.g., spherical, cylindrical) computational domains, thus leading to very large unnecessary computational costs for problems for which the domain aspect ratio deviates significantly from one. Another efficient integro-differential approach was independently developed by [69] and [35], which is only applicable to spherical domains. In [65] a fast method based on Huygens' principle and, more specifically, the presence of lacunae, solves an auxiliary system with the introduction of an inhomogeneity in a layer conforming to the computational boundary.

1.1.3 Numerical methods developed in this thesis

As mentioned in Section 1.1 and discussed in detail in Section 3.2.3, the most commonly used numerical PDE solvers for wave propagation problems suffer from significant loss of efficiency as the acoustical size of the computational domain is increased—owing to the linear growth of the multiplicative constants in the algorithms error estimates with respect to the diameter of the domain; see equation (3.18) and Section 3.2.3. This phenomenon, which is caused by the underlying "dispersion error" (which is also known as "pollution error" in the finite-element context), is effectively resolved in the context of spectral methods such as the Fourier collocation and Chebyshev algorithms. The Fourier-continuation-(FC)-based PDE solvers [1, 18, 53] considered in this work, which are described in detail in Chapter 2, are spectrally accurate, and hence nearly dispersionless in the interior of the domain, but they are much more general than classical spectral methods: unlike the Fourier collocation method, the FC algorithms can be applied to general nonperiodic problems in general domains, and unlike the Chebyshev spectral methods, they do not entail highly restrictive Courant-Friedrichs-Lewy (CFL) time-step constraints. In this thesis, an FC-based numerical solver for the acoustic wave equation is introduced (Chapter 3), and it is hybridized with other numerical solvers (Chapter 8).

Similarly, methods for evaluation of computational boundary conditions are unable to provide a convergent representation on arbitrary boundaries while maintaining computational efficiency—local methods methods are efficient but inexact, while known "efficient"

nonlocal methods potentially require the costly extension of the computational domain to a sphere. In these regards, this thesis extends the work of [42] (a fast method that can evaluate convergent, nonlocal boundary conditions) to cases that include nonconvex and even disjoint domains—a feature demonstrated for the first time in the present work (Section 9.5).

1.2 Parallel computing

Moore's law has been a continual boon for all scientific fields that rely on numerical simulations. Originally presented in the 1965 paper [56], this "law" was an expression of the simple factual observation that the number of transistors available on each new generation of integrated circuits appeared to be growing exponentially. Though the figure is not exact, the period over which these devices double in complexity (measured simply by transistor count) is often quoted to be 18 months [46]. What was once a rather casual observation, however, has become a self-fulfilling prophecy. For years now, Moore's law has been a continually moving target that the semiconductor industry places on its own research and development efforts. Even though nearly 50 years have passed since Moore's original publication, this trend continues today, and with the current technological horizon, is expected to continue for at least several years, possibly into the next decade [47].

Of far more interest to the mathematical community as *consumers* of computing technology, however, is the implied performance. Fortunately this trend has also applied to measures of computational progress that, from the computational viewpoint, have far more practical interest—including available random access memory (RAM) and floating point operations per second (FLOPS). As an example, Intel's original Pentium P5 architecture [44], introduced in March of 1993, could perform up to 75 MFLOPS, and it accommodated memory access at up to 554 MB/s. In contrast, November of 2008 saw the release of the first of Intel's i7 line, the predominant architecture today, the first instance of which was the 965 Extreme Edition—which is capable of approximately 51.2 GFLOPS while reading memory at 27 GB/s. That is, over a span of 15 years, the floating point speed increased by a factor of roughly 680 (doubling more than nine times over), while the memory bandwidth increased by a factor of 47 (doubling more than five times).

Unfortunately, reaping the benefits of this trend is not as simple as purchasing the latest

and greatest on the market. While the potential for raw computational speed has increased, the requirements for harnessing the full extent of the available computing power have grown in complexity. There are two major areas where such developments have influenced this work. Firstly, there has been an increasing gap between the ability of a state of the art processor to access memory (MB/s) and the rate at which it can perform computation (FLOPS). If a computation requires a large amount of memory access relative to the number of floating point operations performed, then the practical increase in computing power over the last decade is significantly reduced. Notice that, from the release of the first Pentium to the i7, the comparative speedup between floating point and memory speeds differs by more than an order of magnitude. Furthermore, the i7 965 EE cannot read memory fast enough (at 4 bytes per single precision value) to meet the demands of its floating point capabilities, assuming every operation requires at least one newly fetched operand from memory (such as summation over an array).

Secondly, there has been an increasing trend, over roughly the past decade or so, towards multi- and many-core architectures. The i7 processor previously mentioned is only capable of reaching its peak performance of 51.2 GFLOPS when a considerable degree of parallelism has been accounted for, taking advantage of all four cores in addition to using vectorized floating point operations (SSE) on each. If these considerations are neglected, the same device would only be capable of roughly 3.2 GFLOPS. This second difficulty is of even greater concern after noting that not every computation *can* be efficiently parallelized. Amdahl's law [5] states no amount of parallelism can ever accelerate a computation beyond the time required for the longest purely sequential subcomputation. A formal (and slightly more general) statement is that, given a set of atomic computational tasks and the partial ordering implied by their dependencies, the minimal time to complete the computation is the largest sum of task times over any subset for which the partial ordering implies a unique full ordering. Algorithms that do scale well in this domain, in part by minimizing this property, will be of ever-greater importance as this trend continues.

1.2.1 Shared and distributed memory environments

Most large-scale parallel computers available today are "distributed memory" clusters, comprised of a large number of individual computers (or "nodes") combined with some kind of interconnect (InfiniBand and Ethernet are currently the most common) allowing for com-

munication between them. The most widespread (but not only) model for programming in such an environment is the message-passing model, the most famous implementations thereof adhering to the message passing interface (MPI) standard [68], in which a separate copy of the program is run on each node, with interdependencies in the computation managed by the passing of messages, either point to point (matched send and receive pairs) or collective group communications (such as broadcast and parallel reductions). In order to efficiently use such a device, there are two concerns that are typically addressed: the amount of computational work for which each node is responsible should be reasonably balanced, and the cost of communication incurred by distributing said work should be limited.

This is in contrast to the less-scalable but simpler "shared memory" model, where a single program is split into several parallel "threads", each having direct access to the same physical memory. Two of the most common APIs for this style of development are POSIX-threads, for which the manual creation and management of each computational thread is necessary, and OpenMP [24], which abstracts this work into a much cleaner interface, but is only well-supported by more recent compilers and tools. Shared memory models have the advantage of greatly reducing the cost of communication, but place a greater burden on the programmer to ensure program correctness (thread safety). Furthermore, while most modern desktop computers naturally support this style of parallelism, the cost of shared memory hardware tends to grow much faster, as a function of the number of cores, than the cost of a comparable distributed memory system, and even still necessarily sacrifices some memory performance owing to purely physical limitations in their construction.

This work does not present an implementation targeted to MPI. Instead, support for both CUDA (described in the following section) and OpenMP are demonstrated. While this thesis fully addresses the issues that arise from use of the two latter communication environments (which are rather similar from a software development standpoint), in the course of the developing the CUDA implementation, the primary concerns for MPI support, namely division of labor and efficiency of communication, *are* addressed. This thesis work thus introduces concepts that enable applicability of the highly efficient novel FC and boundary condition methods, see Section 1.1.3, in all three types of modern computational infrastructures, although it demonstrates the applicability of these concepts only in the cases of CUDA and OpenMP.

1.2.2 General purpose graphics processing units

Modern GPUs are an extreme case of the many-core engineering trend. Originally designed as specialized hardware to perform a fixed number of simple functions on a per-pixel or per-texel basis, current-generation GPUs are fully programmable and often feature a number of cores totaling in the hundreds. Each of these cores, however, is far more limited in functionality than the corresponding cores of a traditional desktop CPU.

NVIDIA Corporation's Compute Unified Device Architecture (CUDA) provides a programming model [59] specifically for these modern many-core programmable GPUs. CUDA provides both an abstraction of the underlying hardware and its capabilities, as well as a programming environment (CUDA for C) for developing applications. In the CUDA model, a single GPU is comprised of a uniform global memory space plus a number of symmetric multiprocessors (SMPs). Each SMP possesses a number of parallel computing cores called streaming processors (SPs), which collectively share a small pool of local memory and a *single* instruction unit—a detail of great importance, since even though an SMP on a typical GPU may have 16 SPs, they must all execute identical instructions in lock-step. This results in what NVIDIA has dubbed the "single-instruction, multiple-thread" or "SIMT" execution model, whose nomenclature follows from "single-instruction, multiple-data" (SIMD) and similar designations. For simplicity, a collection of threads running in parallel, over identical instructions, on an SMP is referred to as a "threadblock".

Following the SIMT architecture, the second major deviation from standard CPU architectures is the manner in which memory is accessed. Each SMP possesses a single high-bandwidth, but high-latency, connection to global GPU memory. Furthermore, the memory caching mechanisms present on typical desktop CPUs to hide this physically necessary latency are conspicuously absent, requiring a developer to explicitly manage these delays, most importantly through a form of thread multiplexing discussed in more detail in Section 4.1. Finally, since an SMP may only access global GPU memory by reading or writing a 16-, 32-, or 64-byte contiguous block at a time, the underlying SPs must have their memory access patterns planned out in such a way that they collectively "coalesce" into these block operations, or else performance may be (quite literally) decimated.

While numerical computing remains a niche application for GPUs today, many of their innovations are influencing mainstream chip designers such as Intel with their Larrabee

architecture [66] (now superseded by the Intel many integrated core (MIC) architecture). They also provide, effectively, a snapshot into the anticipated future, with respect to relative floating point versus memory performance. In targeting these devices for development, then, one not only uses the most powerful floating-point devices currently available, but also potentially gains a leg-up on the architectural trends of the future.

1.2.3 Limited precision arithmetic

It is becoming increasingly common for GPUs to be designed with support for double precision arithmetic, partly owing to a growing interest in such capabilities from the academic community. More commonly, however, double precision arithmetic is only available with reasonable performance (half that of single precision arithmetic) on a subset of the newest generation of devices—and on older devices it is available either at a significant performance penalty or not at all. Therefore, in this thesis, care has been taken to ensure that the algorithms used retain both accuracy and stability, insofar as is possible, when implemented with only single precision floating point arithmetic. This design lends itself to one other advantage—supposing that the provided accuracy is sufficient, and the performance is memory bound (as is increasingly likely given newer architectures), then a single precision implementation may offer up to twice the speed of an equivalent double precision code.

Chapter 2

Fourier continuation

Fourier collocation methods take advantage of the rapid convergence of Fourier series for smooth, periodic functions in order to enable the numerical solution of suitable problems in a manner that is both very accurate and efficient. Once a discrete function is expanded in a Fourier basis, common numerical operations such as differentiation become simple to implement (as a diagonal operator) and yield spectral convergence. When solving time-dependent partial differential equations, spurious high-frequency oscillations associated with nonlinearity or instability in the numerical solution may also be damped directly in a manner analogous to the introduction of artificial viscosity, improving the stability of such methods without significantly impairing the numerical accuracy and without greatly increasing numerical dispersion. Furthermore, the fast Fourier transform [23] (FFT) provides an efficient means of mapping a function to and from the discrete Fourier basis in only $O(N \log N)$ time. The limitation of the Fourier collocation methods, however, is that the PDE solution must be a periodic function over a rectangular domain: if the Fourier collocation method is applied to a nonperiodic function directly, the resulting representation converges very poorly, with slow convergence in the domain interior, and with an $\mathcal{O}(1)$ error near the boundary. In other words, if Fourier collocation is used for nonperiodic functions, then the Gibbs phenomenon takes place. The Fourier continuation method provides a means to address this difficulty, and thus enables spectral treatment of PDE problems whose solutions are not periodic functions.

The Fourier continuation (FC) method [1, 11, 15, 18, 19, 53] overcomes the aforementioned limitations by seeking an interpolating trigonometric function which is periodic on a domain *larger* than the original one—allowing for a smooth transition between the function values at the original boundaries of the nonperiodic domain; see Figure 2.1. To intro-

duce the Fourier continuation method, consider a given (generically nonperiodic) function $f : [0, 1] \to \mathbb{C}$, defined in the interval $[0, 1]$, whose discrete samples

$$f_n = f(x_n), \quad x_n = \frac{n}{N-1}, \quad n = 0 \ldots N-1 \tag{2.1}$$

are known. In order to represent the function f by a Fourier series while avoiding the Gibbs phenomenon, a Fourier expansion of f is sought on an interval $[0, b]$ with $b > 1$. That is, the Fourier continuation method seeks to produce a function f^c of the form

$$f^c(x) = \sum_{k=-W}^{W} c_k e^{2\pi i \frac{kx}{b}} \tag{2.2}$$

which agrees with f as closely as possible, in the in the interval $[0, 1]$ where f is defined. The function f^c smoothly transitions over the interval $[1, b]$ to create a b-periodic extension, as depicted in Figure 2.1 for the particular case of the function $f(x) = x$. Unfortunately, it is no longer possible to directly compute the coefficients c_k, of f^c with a simple application of an FFT. The remainder of this chapter discusses two methods for the construction of such expansions, including an FFT-based algorithm with a cost essentially identical to that of a single FFT.

2.1 Fourier continuation based on singular value decomposition

The most direct (if not the most efficient or best conditioned) approach for the determination of the Fourier series coefficients c_k results [11, 15, 19] as the N known data points together with the definition of f^c are interpreted as a system of N equations for M unknowns:

$$f_n = \sum_{k=-W}^{W} c_k e^{2\pi i \frac{kx_n}{b}}. \tag{2.3}$$

To address the potential ill conditioning in this problem it is advantageous to let $M < N$, in which case this system of equations is solved in the least-squares sense. (A fast and well conditioned alternative to the present SVD-based approach is put forth in Section 2.2, but

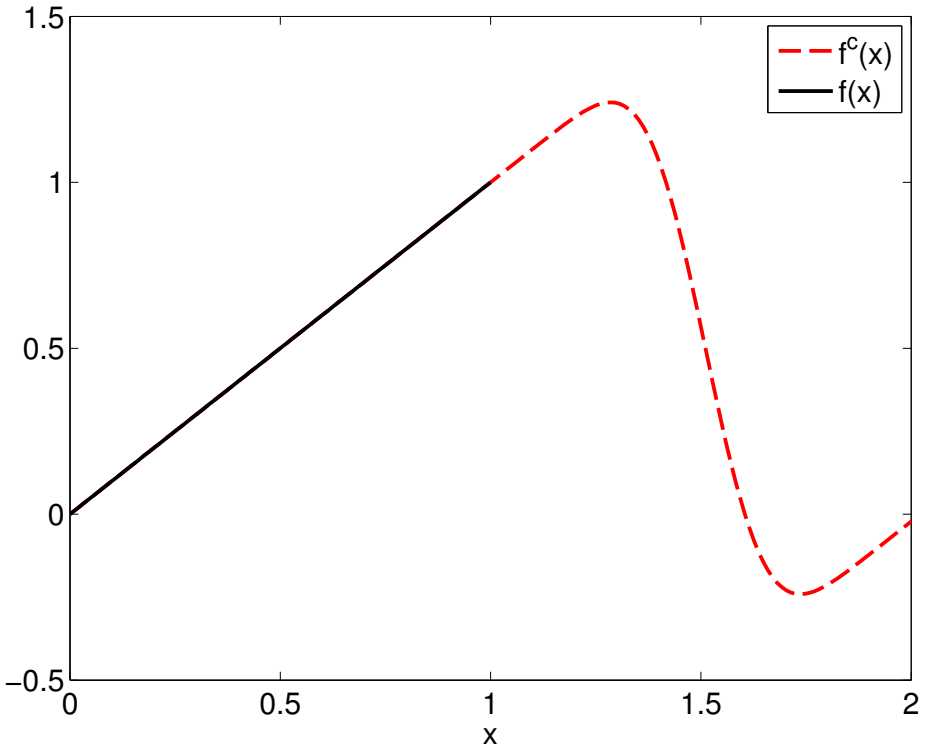


Figure 2.1: Fourier continuation $f^c(x)$ of the function $f(x) = x$

the present discussion may prove useful nevertheless.) Defining the matrix A by[1]:

$$A_{mn} = \begin{cases} e^{2\pi i \frac{m x_n}{b}} & \text{when } m \leq W \\ e^{2\pi i \frac{(W-m) x_n}{b}} & \text{otherwise,} \end{cases} \tag{2.4}$$

the desired coefficients c_k are then given by the components of the solution vector y of the problem

$$\min_y \|Ay - b\|_2, \tag{2.5}$$

where $b_n = f_n$. The singular value decomposition [33]

$$A = USV^H \tag{2.6}$$

(where U, V are unitary matrices and where S is a nonnegative diagonal matrix) is a natural choice for the solution of least-squares problems of this type. Defining the pseudo-inverse S^+ of the diagonal matrix S as the diagonal $\max(M,N) \times \max(M,N)$ matrix whose nonzero

[1]The convention used here follows [1], with indices starting at zero rather than one.

elements equal the reciprocals of the corresponding nonzero elements of S, in terms of this decomposition the least solution y is given by

$$y = VS^{+}U^{H}b. \tag{2.7}$$

The least-squares problem just considered becomes ill conditioned as N and/or M are increased—in light of which, it is necessary to use a truncated pseudoinverse of S, S_{ϵ}^{+}, that is, a diagonal matrix with coefficients defined by

$$(S_{\epsilon}^{+})_{mm} = \begin{cases} (S_{mm})^{-1} & \text{when } \frac{S_{mm}}{S_{0,0}} \geq \epsilon \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

The parameter ϵ is typically taken to be on the order of machine precision. As demonstrated by [15], this method converges to order $r-1$ provided that $f(x) \in C^{r}$.

As mentioned above, this approach is computationally expensive and ill conditioned. Indeed, evaluation of the singular value decomposition requires $O(MN^2 + N^3)$ operations, and solution of the system (per set of unknowns, once the SVD is available) requires $O(M^2 + MN)$ operations. Although in the surface representation context [15] these issues do not pose significant difficulties, the application of this approach to the solution of time-domain PDEs would give rise to a prohibitively expensive solver. Furthermore, the ill-conditioning inherent in the solution of the least-squares problem would limit severely the size of the problems that could be considered. Both of these difficulties are eliminated in the Fourier continuation algorithm presented in the following section.

2.2 Accelerated Fourier continuation based on Gram polynomials

In contrast to FC(SVD) algorithm presented in the previous section, the accelerated FC(Gram) approach, introduced in [18] and further modified for explicit time-marching algorithms in [1], first seeks to determine a number $C > 0$ of artificial function values which smoothly transition from f_{N-1} to f_0 over the interval $[1, b]$, where the size of the continuation interval is set explicitly by $b = (N + C)/(N - 1)$. Once this extension $f_{\text{ext}} = \{f_N, \ldots, f_{N+C-1}\}$ is determined, the frequency domain coefficients c_j are computed by an FFT of size $N + C$.

The main idea underlying the evaluation of the C extension values involves consideration of the function $f(x)$, defined in the original interval $[0,1]$, along with its translation by distance b, $f(x-b)$, as implied by the fact that a b-periodic extension is desired. Selecting a positive integer d that prescribes a certain number of "matched point-values" (the length of the interval where matching takes place is given by $\delta = (d-1)h$) consider the set of d right-most discretization points in the interval $[0,1]$, namely $\mathcal{D}_{\text{right}} = \{x_{N-d}, \ldots, x_{N-1}\}$ as well as the set of d left-most discretization points $\mathcal{D}_{\text{left}} = \{b + x_0, \ldots, b + x_{d-1}\}$ in the interval $[b, b+1]$; see, e.g., Figure 2.2. Roughly speaking, the additional C needed function values in the interval $[1, b]$ are obtained as the point values of an auxiliary trigonometric polynomial, with periodicity interval $[1-\delta, 2b-(1-\delta)]$ and with appropriately selected bandwidth, produced by means of an FC(SVD) fit to the function values on $\mathcal{D}_{\text{right}} \cup \mathcal{D}_{\text{left}}$, as described in what follows.

Polynomial interpolation of the function values on $\mathcal{D}_{\text{right}} \cup \mathcal{D}_{\text{left}}$ is produced by means of two Gram (orthonormal) polynomial bases $\mathcal{B}_{\text{right}}$ and $\mathcal{B}_{\text{left}}$ of the respective spaces of polynomials of degree $< d$ on the intervals $[1-\delta, 1]$ and $[b, b+\delta]$ and associated orthogonal projections. (The sets $\mathcal{B}_{\text{right}}$ and $\mathcal{B}_{\text{left}}$ are orthonormal bases of the space of polynomials of degree $< d$ with respect to the natural discrete scalar product defined by the discretization points $\mathcal{D}_{\text{right}}$ and $\mathcal{D}_{\text{left}}$, respectively.) The algorithm then proceeds by precomputing, for each polynomial $p \in \mathcal{B}_{\text{right}}$, a smooth function defined for $x \geq 1-\delta$, which approximates p closely in the matching interval $[1-\delta, 1]$, and which blends smoothly to zero for $x \geq b$. Such rightward extension is obtained by means of appropriately oversampled least-squares approximations by Fourier series of periodicity interval $[1-\delta, b-(1-\delta)]$, as described in [1]. Similarly, the scheme obtains, for each polynomial $p \in \mathcal{B}_{\text{right}}$ a smooth blending function that agrees with p in the matching interval $[b, b+\delta]$ and vanishes for $x \leq 1$.

Once such smooth blending functions have been precomputed (via a high-precision implementation of FC(SVD) on a refined mesh), an FC extension can be produced, for any function, from the function values at the set of points $\mathcal{D}_{\text{right}} \cup \mathcal{D}_{\text{left}}$—since the interpolating polynomials of degree $d-1$ on $\mathcal{D}_{\text{right}} \cup \mathcal{D}_{\text{left}}$ can be expressed as linear combinations of the polynomials in the bases $\mathcal{B}_{\text{right}}$ and $\mathcal{B}_{\text{left}}$, with coefficients that can be obtained rapidly by means of scalar products. This periodic-extension procedure is demonstrated in Figure 2.2. As indicated above, once the additional C extension values have been obtained, an application of the discrete Fourier transform in the interval $[0, b]$ to the vector of function values

f_j augmented by the C additional "continuation" values f_{ext} just constructed yields the desired trigonometric continuation polynomial.

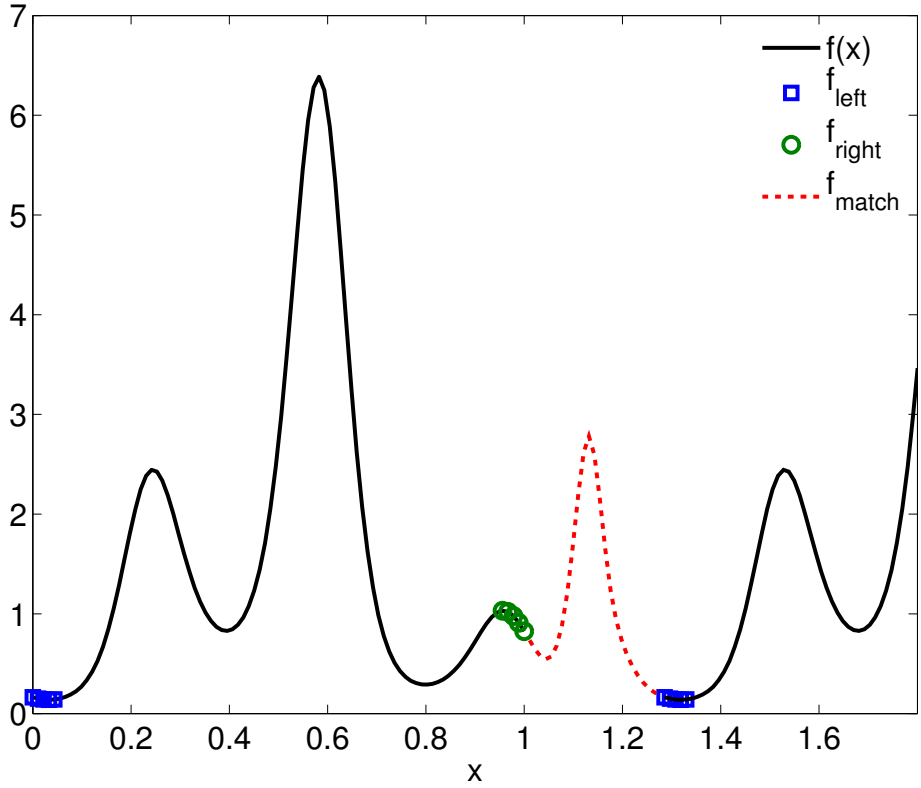


Figure 2.2: The function $f(x) = e^{\sin(5.4\pi x - 2.7\pi) - \cos(2\pi x)}$, sampled at $N = 92$ evenly spaced points over the interval $x \in [0, 1]$, and its continuation via FC(Gram), using $d = 5$ matching points and $C = 25$ extension points. The periodic continuation is superimposed to the right.

The operator that extends N given function values into $N + C$ samples of a smooth periodic function is a sparse matrix operator with simple structure, composed of the $N \times N$ identity matrix and two small $C \times d$ submatrices. The action of the extension operator can be expressed in the block-matrix form

$$\tilde{f} = \begin{bmatrix} I_N \\ A \end{bmatrix} f = \begin{bmatrix} f \\ Af \end{bmatrix}, \tag{2.9}$$

where f is the N-vector of original function values, $\tilde{f}$ is the $(N + C)$-vector of extended function values, I_N is the $N \times N$ identity matrix and A is a sparse $C \times N$ matrix containing the blending-to-zero basis information and whose only nonzero entries occur in its first d

and last d columns. Using the notation

$$f_l = (f_0, \ldots, f_{d-1})^T, \quad f_r = (f_{N-d}, \ldots, f_{N-1})^T, \tag{2.10}$$

the vector Af has the form

$$Af = A_l Q_l^T f_l + A_r Q_r^T f_r. \tag{2.11}$$

Here, each column of Q_l and Q_r contain the d point values of one of the elements of the polynomial basis, and the columns of A_l and A_r contain the C point values that blend the corresponding polynomials to zero, toward the left and right, respectively. As described above, when the left and right blending functions are added, the C needed extension function values result. Because each of these are small, fixed matrices that can easily be precomputed, the application of the extension operator (2.11) can be performed very rapidly. By applying the FFT to the extended vector of function values $\tilde{f}$ the discrete coefficients c_k are obtained and thus the desired representation (2.2) results.

The matrices A_r and A_l differ only by row-ordering, and similarly, Q_r and Q_l only differ by column-ordering. As a consequence of this fact, only half as much data needs be stored/loaded as compared to the original presentation in the FC-AD work [18]. That is, defining R_m as an $m \times m$ matrix with ones on the antidiagonal, serving to reverse the order of elements in an m-vector under multiplication, the composition of matrices $A_l Q_l$ may be alternatively computed using

$$A_l Q_l = R_C A_r Q_r R_d. \tag{2.12}$$

Remark 2.2.1. In some cases it is advantageous to prescribe different values of the parameter d for each of the two Gram bases, $\mathcal{B}_{\text{left}}$ and $\mathcal{B}_{\text{right}}$. In this case the respective parameters are denoted d_{left} and d_{right}, and the order of the pair of Gram bases for the given interval is defined by $(d_{\text{left}}, d_{\text{right}})$.

2.3 Numerical differentiation

Once a convergent Fourier series representation of a function is obtained, the corresponding numerical derivative of the discrete series may be computed directly.

$$f^c(x) = \sum_{m\in K} c_m e^{2\pi i \frac{mx}{b}} \tag{2.13}$$

$$f_x^c(x) = \sum_{m\in K} 2\pi i \frac{m}{b} c_m e^{2\pi i \frac{mx}{b}} \tag{2.14}$$

This can equivalently be seen as a diagonal operator applied to the Fourier coefficients, taking each c_m to $2\pi i \frac{m}{b} c_m$. Once the coefficients have been scaled by this complex factor, they may be inverted in $O(N \log N)$ time via an inverse FFT, and finally restricted to the original N discretization points. The discrete FC differential operator, then, is implemented as the following sequence of operations:

1. Expand the leftmost d and rightmost d values of f in the Gram polynomial bases $\mathcal{B}_{\text{right}}$ and $\mathcal{B}_{\text{left}}$, for $\mathcal{O}(d^2)$ operations.

2. Sum the precomputed Gram polynomial extensions in the interval $[1, b]$, $\mathcal{O}(Cd)$ operations.

3. Compute the FFT of $\tilde{f}$, $\mathcal{O}((N + C)\log(N + C))$ operations.

4. Differentiate the Fourier coefficients, $\mathcal{O}(N + C)$ operations.

5. Invert the FFT, $\mathcal{O}((N + C)\log(N + C))$ operations.

This algorithm has an overall the cost of order $\mathcal{O}(d^2 + Cd + (N + C)\log(N + C))$, which is dominated by the favorable scaling of the requisite FFTs.

2.3.1 Data filtering

It is often the case that higher-frequency terms correspond to numerical errors or noise rather than an important component of the solution. A particular convenience of the FC-based methods is that, as an intermediate step, the numerical data is expanded explicitly in Fourier series—this allows normally expensive frequency domain filters to be applied directly and efficiently. These filters map a function f_c to a smoothed approximation f_c^σ

by applying a scaling factor to the Fourier coefficients that dampen the contribution of higher-frequency terms. Given the finite series expansion

$$f_c(x) = \sum c_k e^{ikx}, \tag{2.15}$$

the smoothed approximation is defined by

$$f_c^\sigma(x) = \sum \sigma(\frac{2k}{N}) c_k e^{ikx} \tag{2.16}$$

for a suitable function $\sigma(s)$, defined over the normalized range of frequencies $s \in [-1, 1]$.

It was observed in [1] that the spectral filter

$$\sigma(s) = e^{-\alpha|s|^p}, \tag{2.17}$$

discussed also in [38] and displayed in Figure 2.3, is particularly well suited to explicit time-domain solvers built using the FC(Gram) method. In [1] the parameter α is chosen to roughly equal $-\ln \epsilon_{\text{machine}}$ (where $\epsilon_{\text{machine}}$ is the "machine epsilon" [71]) in order to make the coefficients effectively vanish for $|s| = 1$, and p is taken to grow linearly with N, which results in a spectrally accurate filter. In the context of this work, however, it is found that a different choice of filter parameters is appropriate. The precise reasons for this, as well as the particular selection of values that prove to be useful, are presented in Chapter 4.

Algorithmically, the application of these filters follows that of the numerical differentiation operators from the previous section. Furthermore, it is especially convenient to compose the filtering and differentiation operators in frequency space. Not only does this eliminate the cost of additional FC expansions and forward and inverse FFTs, but it is also results in better numerical conditioning. Similarly, if both the results of the filtering operator and the differentiation operator are needed at the same time, work may be saved in this FFT-based approach by sharing the computed Fourier expansion between the two operators.

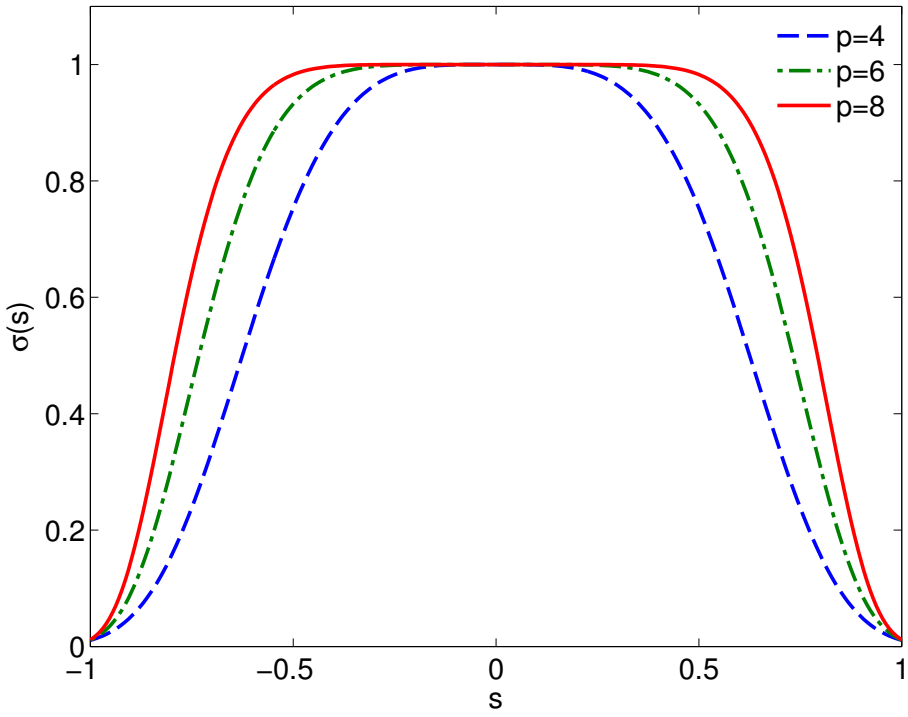


Figure 2.3: The exponential filter $\sigma(s)$ for several values of the order p, all using parameter $\alpha = -\ln 10^{-2}$

Part II

Hyperbolic solver for bounded domains

Chapter 3

One-dimensional solvers

This chapter focuses on the spatially one-dimensional version of the types of problems under consideration in this thesis. Under this greatly simplified setting an FC-based method-of-lines is introduced, its stability and accuracy are studied, and its efficiency is compared with those of other available high-order solvers. Additional comparisons of the FC one-dimensional solver are presented in sections 4.2.2 and 9.1. The full three-dimensional solver is introduced in Chapter 5 and demonstrated in Chapter 9.

3.1 First-order hyperbolic systems

Hyperbolic equations characterize "wave-like" phenomena. Formally, a PDE is hyperbolic if the Cauchy initial value problem is locally solvable in the neighborhood of an arbitrary, noncharacteristic initial surface [29]. A linear hyperbolic PDE may, by the introduction of a sufficient number of auxiliary unknowns, be re-expressed as a first-order hyperbolic *system*. In one dimension, these systems for a vector unknown $\mathbf{u}(x,t)$ take the form

$$\mathbf{u}_t + A(x,t)\mathbf{u}_x = f(x,t) \tag{3.1}$$

for some everywhere-diagonalizable matrix $A(x)$ and inhomogeneity $f(x,t)$.

Equipped with an FC-type operator for computing spatial derivatives, it becomes immediately possible to attempt the numerical solution of the system via the method of lines—by first discretizing the problem in space in such a way that it becomes semidiscrete, and then integrating the resulting ODE forward in time. For the one-dimensional problems under consideration it is desirable to select an equidistant discretization in x, since such discretiza-

tions minimize the restrictions imposed by the CFL condition. Unless otherwise stated, for all of the examples in this chapter the domain is the unit interval $[0, 1]$: a problem on any other bounded interval is reduced to the interval $[0, 1]$ by an affine change of variables. The sample points are then

$$x_n = nh, \quad h = \frac{1}{N-1}, \quad n = 0 \dots N-1; \tag{3.2}$$

the corresponding function values at time $t = t_m$ will be denoted by $\mathbf{u}_n^m \approx \mathbf{u}(x_n, t_m)$. (In the event that the domain under consideration is periodic the mesh size $h = \frac{1}{N-2}$ is used instead, in order to avoid the prescription of a redundant sample point.)

Of the possible methods for time integration, the explicit Runge-Kutta and Adams-Bashforth methods [71], both of orders three and four, are of particular interest in the context of this thesis. The region of absolute stability for all four of these methods includes a symmetric interval, around the origin, along the imaginary axis. This is especially helpful for the class of problems discussed in this thesis. (In the case of the one-dimensional wave equation on a bounded domain, with typical (Dirichlet or Neumann) boundary conditions, for example, the eigenvalues of $A(x)$ are strictly imaginary.)

In order to accommodate nontrivial computational boundary conditions that may take the form of an ODE in time at the boundary, such as those discussed in Chapter 6, it is desirable to avoid the additional complexity introduced by the intermediate steps of the Runge-Kutta methods. Therefore the method of choice for time integration will be the Adams-Bashforth method, of order four (AB-4). It is necessary to initialize the first several steps of the method—but this is typically inconsequential, as these scattering problems normally have initial conditions of zero throughout the domain.

3.1.1 Advection equation

The simplest hyperbolic system is the linear advection equation for the scalar unknown $u = u(x, t)$,

$$u_t + cu_x = 0, \tag{3.3}$$

which admits the trivial exact solution $u(x, t) = u_0(x + ct)$. Although simple, this problem provides a useful testbed for the development and study of numerical methods for general

hyperbolic systems in arbitrary spatial dimensions. Boundary conditions for hyperbolic problems are required wherever the characteristic curves enter the domain. Without loss of generality, in what follows it is assumed that $c > 0$.

3.1.2 Wave equation

The most common second-order form of the one-dimensional wave equation,

$$\begin{aligned} u_{tt} + c^2 u_{xx} &= 0 \\ u(x,0) &= f(x) \\ u_t(x,0) &= g(x), \end{aligned}$$

is not expressed in the standard form of a first-order hyperbolic system. It is readily verified, however, that this equation is equivalent to the system

$$\begin{pmatrix} u \\ v \end{pmatrix}_t + \begin{pmatrix} 0 & -c \\ -c & 0 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}_x = 0 \tag{3.4}$$

with the corresponding initial condition on v

$$v(x,0) = \int_0^1 g(s)ds + \text{const.} \tag{3.5}$$

With an appropriate normalization, the unknowns u and v in this system correspond to pressure and velocity, respectively.

3.2 Accuracy, stability, and dispersion

In this section it is shown that the numerical approach mentioned above in this chapter (an FC-based method-of-lines with time-evolution based on AB-4), gives rise to high-order convergent methods with excellent (minimal) numerical dispersion properties: as shown in the following sections the discrete derivative approximation enjoys spatial accuracy of order $\mathcal{O}(\Delta x^d)$ [18] and the semidiscrete operator has a $\|\cdot\|_2$ norm that grows only linearly with N. Stability, on the other hand, is typically very difficult to discuss analytically with the nonnormal (in the linear algebraic sense) FC-type operators. In the one-dimensional case

presently under consideration stability will be demonstrated over a wide range of parameters, and the corresponding CFL condition and Courant number of the algorithm will be found. Finally, the near-dispersionless character of the method will be demonstrated by examining the performance of the algorithm as the size of spatial domain and the corresponding final solution time are dramatically increased while using a fixed number of sample points per wavelength.

3.2.1 Advection equation

Consider first the advection problem (3.3), and take zero initial conditions, in conjunction with the boundary condition

$$u(0,t) = e^{-a(t-0.5))^2}$$

where the parameter $a = -4\ln 10^{-16}$ is chosen such that the function vanishes, to numerical precision, at $t = 0$. This problem admits the exact solution

$$u(x,t) = e^{-a(t-x-0.5)^2}.$$

As part of this work it was found experimentally, in agreement with the conclusions presented in [1, 53], that $d_{\text{left}} \leq 5$ must be used in order to ensure numerical stability where the physical boundary condition is applied. On the other hand, at the free right-side boundary, it is possible to select $d_{\text{right}} = 12$. This does not improve the overall order of the scheme in this case, since the left-side boundary still contributes a lower-order error, but it does offer some advantage, as will be shown below.

In order to study the convergence of the method, numerical solutions were evolved up to time $T = 2$, at which point the exact solution vanishes to machine precision within the computational domain. The time step Δt was taken sufficiently small so as to allow the error to be dominated by the Δx contribution. Both the peak error (worst error at any time-step) as well as the final error (at $T = 2$) are displayed in Figure 3.2. The peak error is a better estimate of the overall numerical accuracy of the method, but the final error is indicative of how well certain qualitative properties of the PDE are preserved, in particular how reliably the numerical solution exits the domain. In this regards the choice

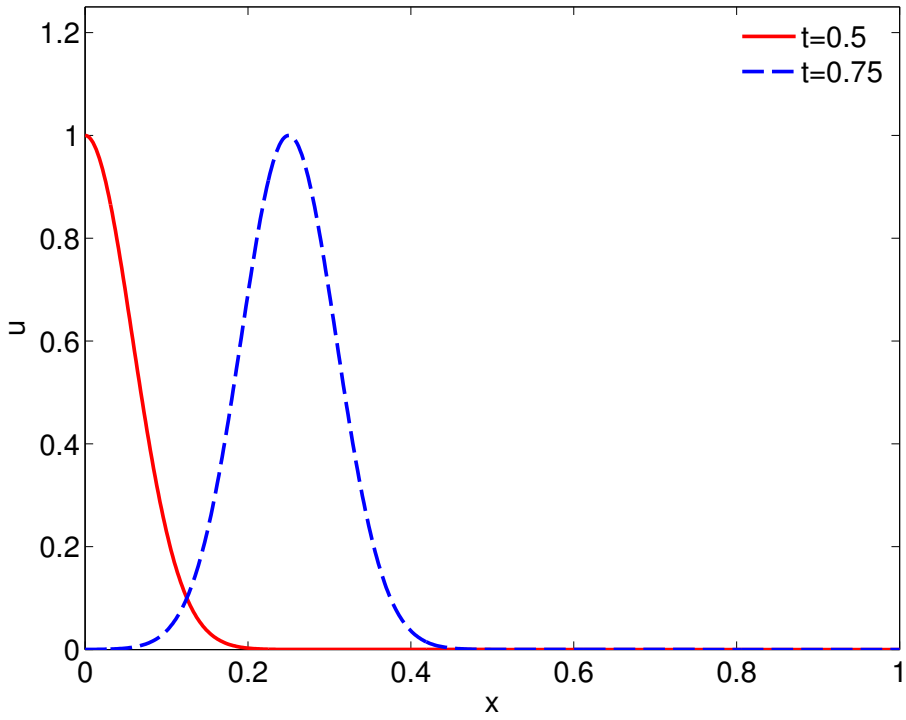


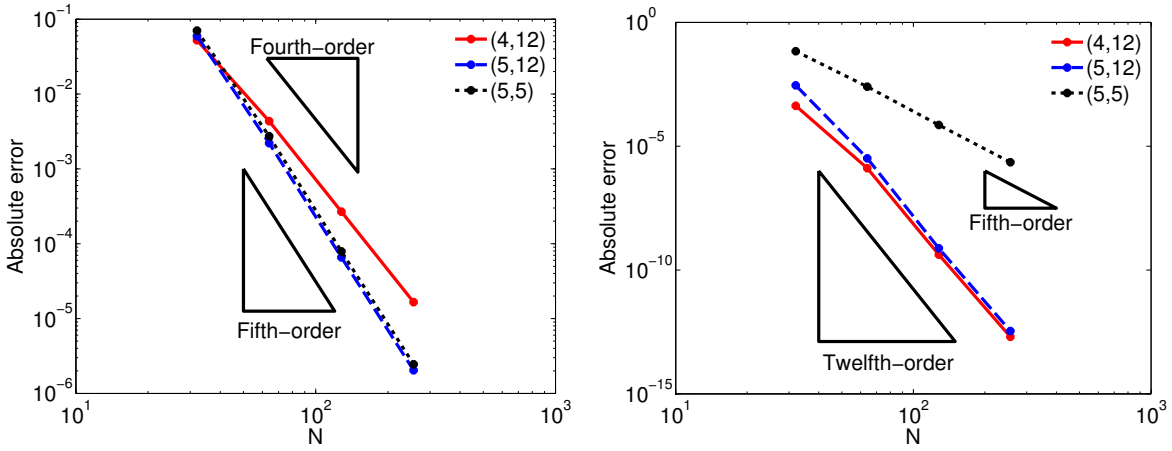
Figure 3.1: Advecting Gaussian

of a higher-order matching polynomial at the right-side boundary clearly distinguishes itself in Figure 3.2(b), even though the formal order of accuracy is not improved, as is seen in Figure 3.2(a) (though the error does decrease slightly as well).

To determine the CFL condition numerically, the solver may be run for many different discretizations in both space and time so that a consistent CFL relationship of the

$$\mu = \frac{\Delta t}{\Delta x} \leq C \tag{3.6}$$

is obtained, under which solution remains bounded for all time. In order to achieve greater confidence in such a numerically determined Courant number C, *randomized* initial data was used. This strategy takes advantage of the fact that time-stepping the solution is equivalent to application of the power iteration scheme (for evaluation the largest eigenvalue of a matrix, see [71]) to the linear time evolution operator. (It is not strictly necessary that randomized data be used, but it allows for a greater degree of confidence to be achieved with a small amount of computational effort. Since the initial conditions are random, their inner product with any unstable eigenmodes is almost surely nontrivial, thus ensuring



(a) Peak advection error

(b) Final advection error

Figure 3.2: Spatial convergence of the 1d advection solver for various values of the order $(d_{\text{left}}, d_{\text{right}})$

comprehensive testing of solution space with one, or a few, code runs.) If the solution, then, remains comfortably bounded for a very large number (typically millions to hundreds of millions) of time-steps, it can be inferred that the method is stable insofar as any practical application is concerned. In Figure 3.3 the error as a function of time is shown for several choices of μ. These few examples suggest $C \approx \frac{1}{7}$, which has been tested extensively, for many values of N, all run for millions of time-steps with random initial data.

In order to study convergence with respect to time, on the other hand, a spatially periodic configuration is considered—since for such a configuration the accuracy the of FC solver is particularly high, and, thus, the need for fine spatial meshes (which, in view of the CFL constraint, would prevent consideration of the larger time-steps in a time-convergence analysis) is minimized. For this convergence test the advection equation in the interval $[0, 1]$ was considered, with the initial condition

$$u(x, 0) = e^{-a(x-0.5)^2}$$

and with the periodic boundary conditions

$$u^{(n)}(0, t) = u^{(n)}(1, t), \quad n = 0, 1. \tag{3.7}$$

Remark 3.2.1. The periodic FC algorithms considered in this thesis enforce periodic bound-

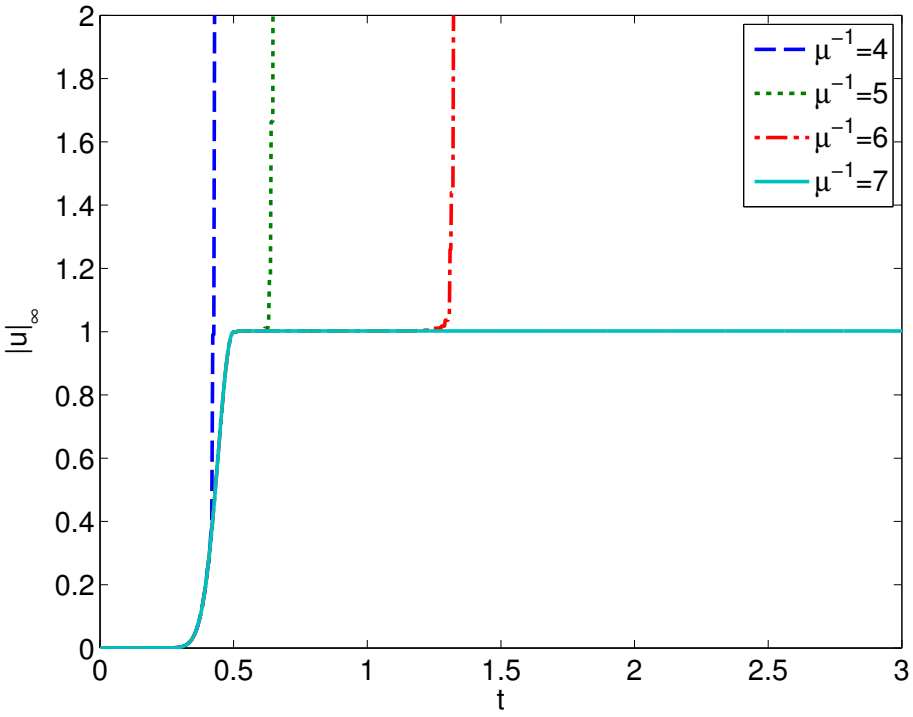


Figure 3.3: Maximum norm over time of the FC numerical solution of the advection equation the for several values of the parameter $\mu = \frac{\Delta t}{\Delta x}$. For $\mu^{-1} = 7$ the solution remains bounded indefinitely.

ary conditions such as (3.7) *implicitly*, by extending the discrete mesh points $x_n = nh$ to include a number d_f of additional points

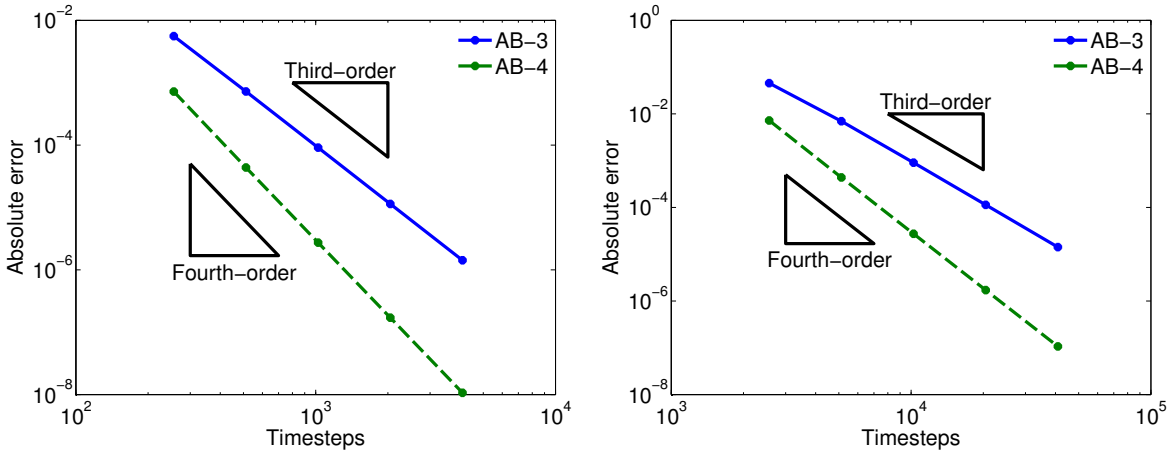
$$\{x_{-d_f}, x_{-d_f+1}, \ldots, x_0, \ldots, x_{N-1}, \ldots, x_{N-1+d_f}\}$$

which, reaching past each one of the two boundary points, are also used as solution sampling points. The additional "fringe" points (namely $x_{-d_f}, \ldots, x_{-1}$ and $x_N, \ldots, x_{N-1+d_f}$) do not, however, correspond to additional unknowns in the numerical system—instead, the function values at the fringe points are prescribed to equal the function values at the corresponding image points within the domain $[0, 1)$; for example, $u_{-d_f} = u_{N-d_f}$. This is equivalently described in block-matrix form: taking F_M to be some FC-type operator (either differentiation or filtering) over M points (as previously defined in Chapter 2), the corresponding

periodic operator $\tilde{F}_M$ is defined by

$$\tilde{F}_M = \begin{pmatrix} 0 & I_M & 0 \end{pmatrix} F_{M+2d_f} \begin{pmatrix} 0 & I_{d_f} \\ I_M & \\ I_{d_f} & 0 \end{pmatrix}. \tag{3.8}$$

In order to demonstrate the convergence over a large range of discretizations in time, the spatial discretization is also refined, maintaining a fixed ratio $\mu = 1/8$. Figure 3.4 shows the resulting convergence using AB-3 and AB-4, where in both cases the error is clearly dominated by the order of the time integration scheme.



(a) One period (b) Ten periods

Figure 3.4: Temporal convergence of 1d advection solver

Remark 3.2.2. The application of the FC method to a periodic domain (which is clearly unnecessary since the full Fourier collocation method could be utilized instead) is only used, here and later in this thesis, to demonstrate the behavior of the FC method under very controlled settings. It is worth pointing out, however, that the FC method in this context does not take advantage of the $[0, 1]$-periodicity—since, indeed, it uses a periodicity interval larger than 1.

3.2.2 Acoustics

This section presents results akin to those put forth in the previous section, but, this time, for the full hyperbolic system (3.4) associated with the linear wave equation with zero

Dirichlet boundary conditions

$$u(0,t) = u(1,t) = 0$$

and with initial conditions given by

$$\begin{aligned} u(x,0) &= e^{-a(x-0.5)^2} \\ v(x,0) &= -e^{-a(x-0.5)^2}. \end{aligned}$$

Note that these initial conditions specify a right-moving wave packet much like the one used in the advection example of the previous section. In fact, by taking the periodic extension

$$G(x) = \sum_{n=-\infty}^{\infty} \chi_{[0,1]}\left(x+2n\right) e^{-a(x+2n-0.5)^2},$$

where $\chi_{[0,1]}$ denotes the characteristic function of the interval $[0,1]$, the exact solution for all time can be expressed in closed form

$$\begin{aligned} u(x,t) &= G(x-t) - G(x+t-1) \\ v(x,t) &= G(x+t-1) - G(x-t). \end{aligned}$$

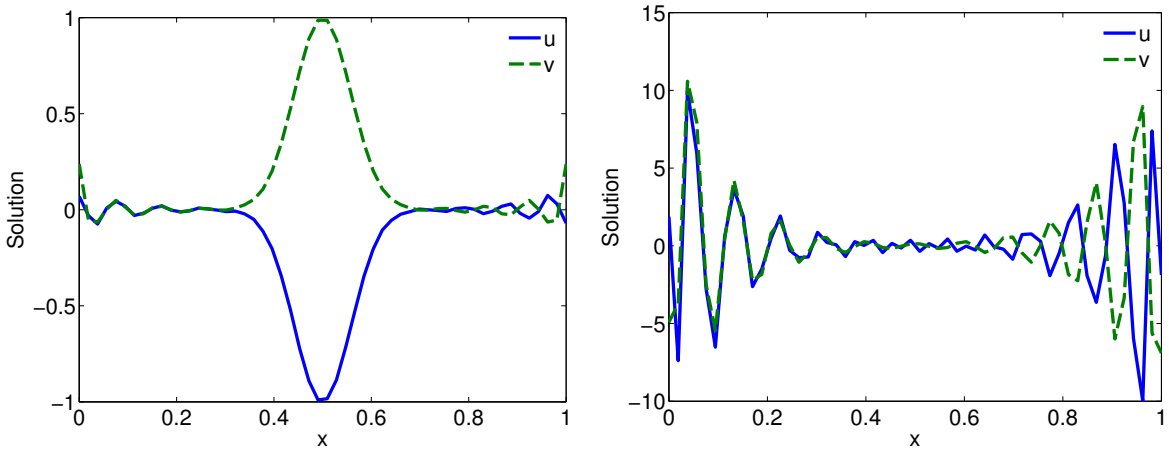
Boundary conditions may be applied on any linear combination of u and v that contains a component in the direction of the incoming characteristic derivative. The boundary condition here is applied to u only, while the corresponding boundary values of v remain free variables in the system. In order to ensure stability two steps were taken, namely 1) The degree of the matching Gram polynomials were restricted at the physical boundaries, but now at both ends of the domain, as well as for both unknowns; and 2) The exponential filter (2.16) introduced in the previous chapter was used. This filter is applied twice at each time-step: once as part of the numerical differentiation, and again, independently, to the solution unknowns. The overall filtered explicit r-step time integration scheme with weights w_j can be expressed in the form

$$u^{n+1} = I_\sigma u^n + \Delta t \sum_{j=0}^{r-1} w_j D_\sigma u^{n-j}, \tag{3.9}$$

where D_σ denotes the filtered FC derivative operator, and I_σ is the FC filtering operator (in

the unfiltered Adams-Bashforth methods, this operator would simply be the identity). An important observation is that, since the matrices I_σ and D_σ do not necessarily commute, the traditional stability criterion (root condition) for the Adams-Bashforth methods does not apply, at least not exactly. With this caveat, however, the root condition remains a good *indicator* of stability, whose predictions can be once again verified via comprehensive numerical studies.

Figure 3.5 demonstrates that, in absence of the exponential filter mentioned above, spurious oscillations in the numerical solution occur. As shown in Figure 3.6, in contrast,



(a) Solution at time $t = 1$.

(b) Solution at time $t = 1.5$.

Figure 3.5: Instability of the unfiltered FC wave-equation solver at two different points in time. The problem was discretized using $N = 54$ points in space. At $t = 1$ the instability is visible, as it first occurs, and by $t = 1.5$ it strongly dominates the true solution.

the filtered algorithm is stable provided a CFL condition is satisfied. The filter parameters

$$p = 8 \quad \text{and} \quad \alpha = -8\mu \ln 10^{-2} \tag{3.10}$$

are used here and elsewhere in this thesis—whenever the exponential filter is applied. This is notably different from the typical usage of this filter as described in [38], since the coefficients do not numerically vanish for the highest frequency modes. On the other hand, this selection is entirely sufficient to ensure stability for the solvers considered in this thesis, and furthermore, it is a closer approximation of the identity operator. The resulting method has Courant number roughly $C \approx \frac{1}{7}$, as is demonstrated in Figure 3.6.

Detailed space and time convergence studies for the wave-equation system, similar to

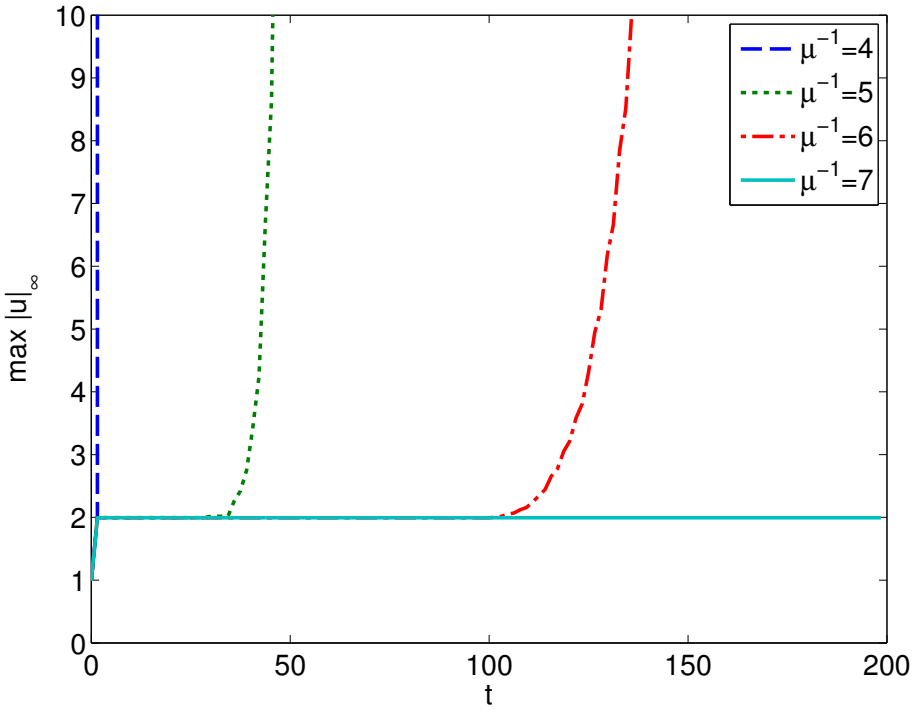
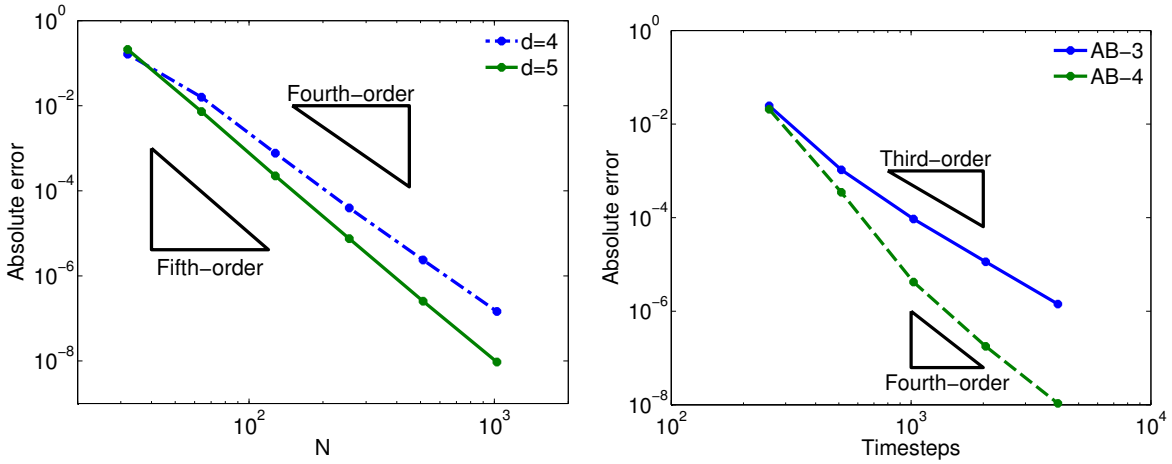


Figure 3.6: Maximum norm of the computed solution of the wave equation (in first-order system form) for several values of the parameter $\mu = \frac{\Delta t}{\Delta x}$, using a frequency domain filter with $p = 8$ and $a = -\ln 10^{-2}$

those presented in Section 3.2.1 for the advection equation, are presented here in Figure 3.7. The present error curves replicate almost *exactly* those presented earlier for the advection equation with matching polynomials of degrees $d_{\text{left}} = d_{\text{right}} = 5$.

Remark 3.2.3. There is no need to independently test the stability of this method for Neumann boundary conditions: such boundary conditions may be put forth equivalently as a Dirichlet condition on the auxiliary unknown v. Since the two unknowns of the system are interchangeable in all other respects, the stability for the Dirichlet problem immediately implies stability for the Neumann problem.



(a) Spatial convergence.

(b) Temporal convergence.

Figure 3.7: Convergence of the numerical solution of the wave equation in first-order system form; cf. the corresponding graphs presented in Figures 3.2 and 3.4 for the advection equation.

3.2.3 Dispersion

One of the greatest strengths of the FC methodology is the nearly dispersionless character that results as the method is applied to problems that include some sort of hyperbolic character and/or wave propagation. In order to quantify this characteristic of the FC method, it is useful to consider, as in [1], numerical solutions of equation (3.3) in large domains, in such a way that the propagation errors over long distances may be evaluated. Using the domain $[0, L]$ with $L = 500$ and, for simplicity, periodic boundary conditions (see Remark 3.2.1), the evolution of the initial conditions

$$u(x, 0) = e^{-(x-10)^2}$$

is considered, for which the exact solution is given by

$$u(x, t) = \sum_{n=-\infty}^{\infty} \chi_{[0,L]}\,(x + nL - t)\, e^{-(x+nL-t-10)^2}.$$

To place the performance of the FC method for this experiment into an appropriate context, its accuracy is compared to that resulting from an eighth-order centered difference scheme, the "spectral-like" Pade method of order four presented [48], as well as the traditional Fourier collocation approach. For each of these spatial operators, the resulting

semidiscrete problem is integrated in time using the AB-4 ODE solver for a sufficiently small time-step that the error for the spatial discretization dominates the solution error. (The value $\Delta t = \Delta x/200$ was used in all cases.) The resulting system was evolved up to the final time $T = 480$, just before the solution reaches the right boundary—thus highlighting the character of the FC method as it propagates waves within the interior of the computational domain. (The boundary behavior of the FC method was demonstrated in previous sections.) Figure 3.8 shows that, in this case, the convergence of the FC algorithm closely matches that of Fourier collocation. It should be noted that this example does not capture the error due to the polynomial approximation in the matching regions—this aspect, for which the FC method also displays superior properties, will be considered in detail in Chapter 4.

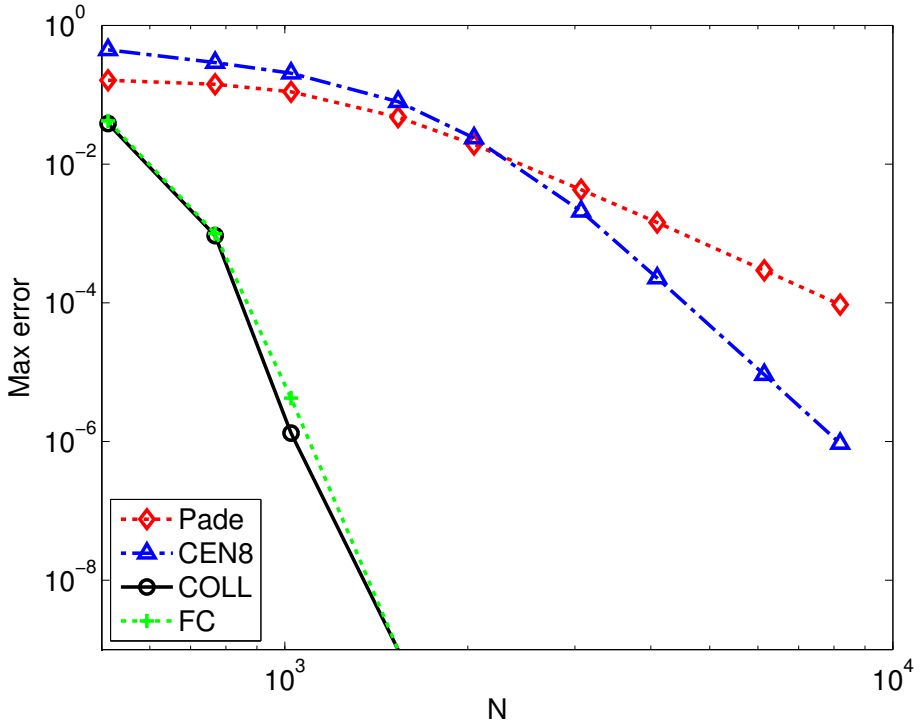


Figure 3.8: Convergence of the solution of the advection problem (3.3) resulting from use of the eighth-order centered difference, fourth-order Pade-like implicit system, Fourier collocation, and FC differentiation operators.

To examine the performance, demonstrated in Figure 3.8, of the various high-order methods under consideration, Fourier analysis was applied to the semidiscrete problem

$$u_t + Du = 0. \tag{3.11}$$

Putting aside at first the particular choice of spatial differentiation approximation D (either that arising from the FC method, or from the other high-order methods considered in Figure 3.8 or, indeed, from any spatial discretization method), a discrete initial condition $f(x)$ may be represented as a linear combination of the Fourier basis functions

$$\psi_k(x) = e^{2\pi ikx/L} \tag{3.12}$$

of the form

$$f(x) = \sum_{k=-N/2}^{N/2} f_k \psi_k(x) \tag{3.13}$$

where f_k denote the Fourier coefficients. It is natural and convenient to examine the behavior of each basis function independently, since this choice of basis diagonalizes any differentiation operator with a translationally invariant kernel [73], such as the finite difference or Fourier collocation schemes. Assuming initial conditions $\psi_k(x)$, the exact solution of the continuous problem is given $\psi_k(x-t) = \psi_k(x)\psi_k(-t)$.

In the corresponding semidiscrete problem the continuous spatial derivative $\frac{\partial}{\partial x}$ is replaced by the discrete approximation D. Introducing the scaled coordinate $y_j = Nx_j/L = x_j/\Delta x$ and scaled wave-number $\omega_k = 2\pi k/N$ (so that $\omega_k \in [-\pi, \pi]$ independently of the discretization used), there holds

$$D\phi_k(y_j) = i\omega'_k \phi_k(y_j) \tag{3.14}$$

where

$$\phi_k(y) = e^{i\omega_k y}, \tag{3.15}$$

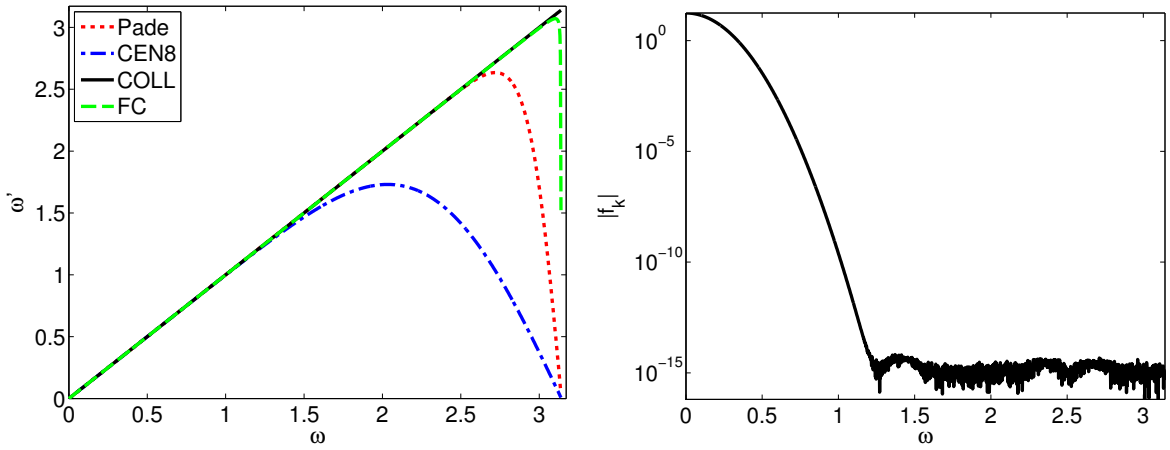
and where ω'_k is the modified *numerical* wave-number of the operator. The function $\omega' = \omega'(\omega)$ for a given discrete operator is called the "dispersion relation" of the operator. The time-dependent factor for the corresponding semidiscrete solutions is given by

$$\phi'_k(-t) = e^{-i\omega'_k t/\Delta x}. \tag{3.16}$$

Since the FC operator is not translationally invariant, the basis (3.12) does not exactly diagonalize the corresponding FC differentiation operator. In such a case equation (3.14)

does not hold exactly, and the dispersion relation may instead be redefined, following [1], by the expression

$$\omega_k' = \frac{(D\phi_k, \phi_k)}{i(\phi_k, \phi_k)}. \tag{3.17}$$



(a) Dispersion relations

(b) Magnitude of DFT coefficients of $f(x)$

Figure 3.9: Dispersion relation for various schemes, and decay of the Fourier coefficients of the Gaussian-bump initial condition $f(x) = e^{-(x-10)^2}$

Historically, the dispersion error has been related to the departure of ω' from ω, that is, by the deviation of the graph $\omega' = \omega'(\omega)$ from the ideal line $\omega' = \omega$. In Figure 3.9(a) the dispersion relations for the spatial differentiation operators under consideration are compared, and, for reference, in Figure 3.9(b) the Fourier coefficients for $N = 5000$ points are shown. It can be seen that the coefficients of $f(x)$ vanish to machine precision for all $\omega \geq 1.25$, restricting the solution to a range of frequencies seemingly well approximated by all of the spatial operators under consideration. Consideration of Figure 3.8, however, shows this not to be the case.

To fully explain this disagreement, closer inspection of the dispersion relation is required. To do this it can be noted that, for each basis function ψ_k, the error in the approximate solution is given by

$$\begin{aligned} |\psi_k(x)\psi_k(-t/\Delta x) - \psi_k(x)\psi_k'(-t/\Delta x)| &= |\psi_k(-t/\Delta x) - \psi_k'(-t/\Delta x)| \\ &= |e^{-it\omega_k} - e^{-i\omega_k' t/\Delta x}| \\ &= |1 - e^{i(\omega_k' - \omega_k)t/\Delta x}|. \end{aligned} \tag{3.18}$$

In other words, the approximate solution contains a phase error equal to

$$\frac{t}{\Delta x}|\omega_k' - \omega_k|, \tag{3.19}$$

large values of which imply that the overall solution is necessarily highly inaccurate. For the example under consideration it is given $t/\Delta x = 480/0.1 = 4800$, and hence large errors in the numerical solution may arise even when $|\omega_k' - \omega_k|$ is small. In Figure 3.10, the error estimate $4800|\omega_k' - \omega_k|$ is displayed for the various methods under consideration, yielding, in each case, the expected error in the final solution on a per-frequency basis. The results displayed in this figure justify the superior performance of the FC method first observed in Figure 3.8, in spite of the fact that, as noted earlier from consideration of Figure 3.9(a), the dispersion relations for all four differentiation methods under consideration are indistinguishable for values of $\omega \leq 1.25$, for which the exact solution has Fourier coefficients above the machine precision level.

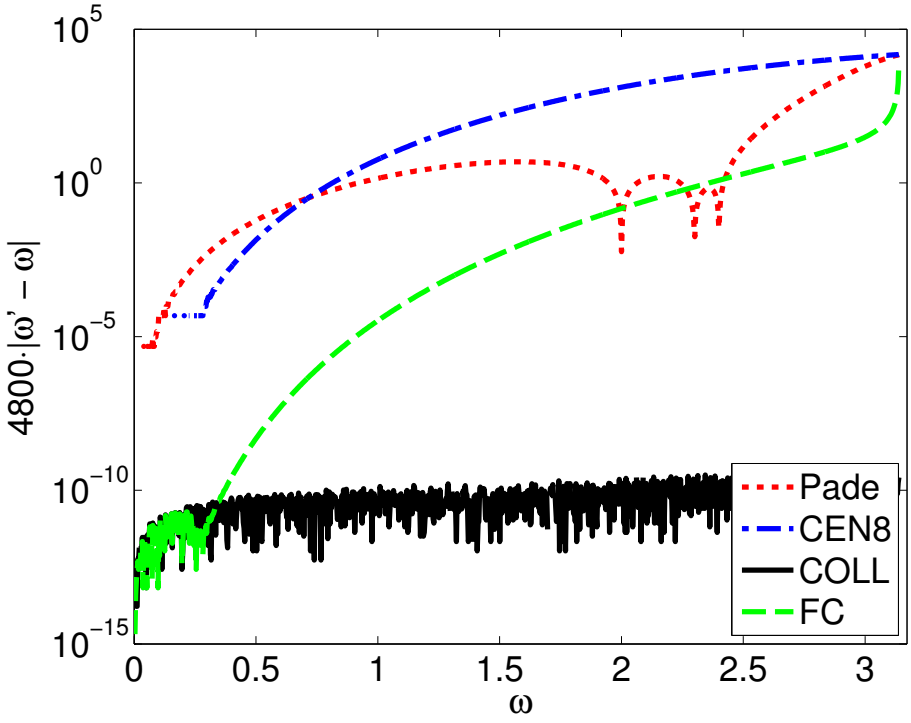


Figure 3.10: Error estimate $4800|\omega_k' - \omega_k|$ for the model advection problem as a function of ω

Chapter 4

Segmentation and parallel computation

This chapter describes modifications of the basic FC algorithms presented in Chapter 2 that enable implementation of the corresponding FC-based PDE solvers *for arbitrary spatial dimensions* (see Chapters 3 and 5) in cutting edge high-performance parallel computing infrastructures. As shown in this and subsequent chapters, the resulting algorithm is well adapted for execution in specialized modern many-core processors such as GPUs and multi-core CPU clusters, with high-quality parallel scalability and minimal impact on the excellent numerical properties of the underlying numerical methods.

4.1 Thread multiplexing

In some cases it is desirable to configure a parallel computation in such a way that a computational task is divided over a large number of threads—possibly in a number of threads that outnumbers the number of available computing cores. In the latter case some or all processors must execute more than one thread, thus giving rise to "thread multiplexing". Use of thread multiplexing with numbers of threads that far outnumber the numbers of processors can lead to highly efficient load balancing during runtime provided special software or hardware support is available to lessen the overhead associated with switching between threads. In fact, in the GPU literature, the nomenclature "thread-multiplexing" is almost exclusively reserved for such "many-thread-per-core" situations—which, in fact, are particularly well suited for execution in GPU infrastructures.

Thread multiplexing is explicitly built into a number of modern concurrency-oriented

languages, including the CUDA programming environment for GPUs as well as the CPU languages Erlang [6], Scala [60], and Go [34]. Not only does multiplexing readily enable dynamic load-balancing, but, for CUDA applications, it is in fact necessary, as discussed below, in order to produce memory-efficient code.

Each SMP on a CUDA-capable device may multiplex over a set of executable threads in order to hide the extremely long latency involved in global GPU memory access. Once a threadblock has made such a memory request, it halts progress until the operation is completed, or "blocks the threadblock", and the SMP switches to a new *ready* threadblock at a negligible cost. As long as a sufficient number of distinct tasks is available to the SMP, this multiplexing strategy hides the long memory-access times (which in many cases could otherwise dominate the computational cost) by "staggering" many parallel read/writes in time. Effective hiding of memory-access times requires that the problem be subdivided into a number of tasks far outnumbering the number of cores. In addition, the memory footprint of each task must be small enough that many threads can be resident simultaneously within the local memory of an SMP, in order to allow for efficient context switching from blocking to ready threadblocks.

4.2 Line segmentation

The largest atomic unit of computation associated with the FC methodology presented in Section 2.2 is the FFT (direct and inverse) along each line in the domain. While it is possible to implement these transforms in parallel, the communication cost incurred in doing so typically dominates the computational time. Thus parallel FFTs present a significant problem: as larger and larger domains are considered, either the requisite communication (if each FFT is parallelized) or the atomic subcomputations themselves (if separate FFTs are evaluated in separate cores) grow in size without bound. Fortunately the size of the FFTs required by the FC method grows sublinearly, as $N^{1/d}$, with the overall number N of unknowns, for a given problem in d-dimensional space. But, if left unchecked, this growth still hinders the GPU performance—as the per-thread memory footprint increases to a level for which a very small number of threads can remain resident per core, and efficient thread multiplexing becomes impossible.

With only slight modification, however, the FC method may be executed in a way that

only requires $\mathcal{O}(1)$ storage and computational work per atomic task. In fact, as a by-product of this modification, the overall computational cost required by the FC algorithm is reduced to $\mathcal{O}(N)$—eliminating the logarithmic component arising from the FFT. The modified approach is, in fact, straightforward: the FC method need not be applied to an entire line of points at once. Each line may instead be split into smaller segments consisting of a number n_s of discretization points each. The more compact FC operator, now of size n_s, is applied independently to each segment.

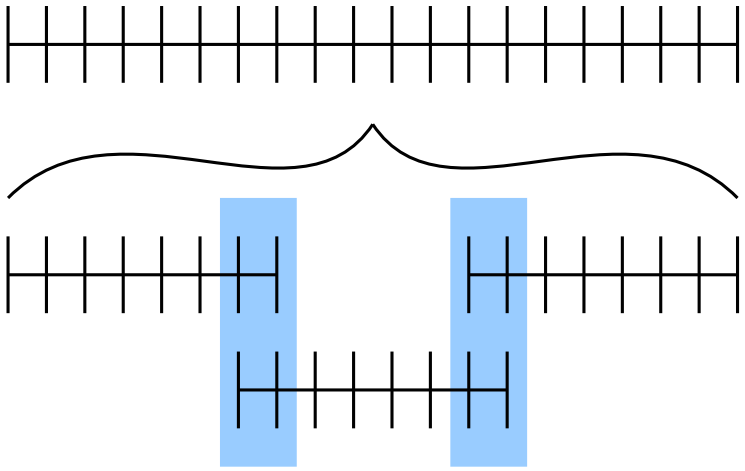


Figure 4.1: A line containing $N = 20$ points, split into three segments with $n_s = 8$ points each. Here there are $d_{\text{in}} = 2$ points per overlap region (colored in blue), resulting in 7, 6, and 7 interior points for the three segments, from left to right.

Instead of using interdomain boundary conditions, the approach presented here ensures consistency of the PDE solution across pairs of adjacent segments by selecting segment-placements for which any two neighboring segments on a given line overlap and share no less than a number d_{in} of "fringe" points. Thus, Gram polynomials are constructed using different numbers of matching points on the left and right matching subintervals, see Remark 2.2.1, with orders $(d_{\text{left}}, d_{\text{right}}) = (d_0, d_{\text{in}})$ for the left-most segment, $(d_{\text{left}}, d_{\text{right}}) = (d_{\text{in}}, d_{\text{in}})$ for internal segments, and $(d_{\text{left}}, d_{\text{right}}) = (d_{\text{in}}, d_0)$ for rightmost segments. Throughout this work the values $d_0 = 5$ and $d_{\text{in}} = 12$ were used. Clearly, under this setup two values of the solution are obtained, for any time-step, at points within overlap regions (one from

the computation on the segment extending to the left, and another from the segment extending to the right). This ambiguity is resolved by selecting the result from the segment for which the evaluation point is most-internal. If there are an odd number of points in the overlap, the rightmost value is assumed for the (equallydistant) central point.

Remark 4.2.1. The segmentation method described above in this section is a direct generalization of the one-dimensional FC solver for periodic problems presented in Section 3.1.1.

A useful feature in this context is that, for interior overlaps where the matching polynomials of the FC(Gram) method are *not* applied at physical boundaries, Gram polynomials of very high degree can be used while retaining stability. In line with the observations made in [1], the value $d_{\text{in}} = 12$ provides very high order of accuracy on the interior while still allowing for a stable overall numerical scheme.

For simplicity, convenience and efficiency in GPU implementation, the segmentation structure is constructed in such a way that all the segments have the same size. When a line does not evenly divide into segments of length n_s, as is frequently the case, this constraint may be accommodated by simply increasing some of the intermediate segment overlaps. Furthermore, the GPU implementation takes advantage of this structure by replacing the FFT-based sequence of operations in the FC(Gram) algorithm with a single, dense matrix-vector product (see Remark 4.2.2 below). Matrices corresponding to each type of FC(Gram) operator (differentiation, filtering) are precomputed with respect to the prescribed segment size n_s. The requirement that segments have a fixed size, which introduces only a small amount of redundant computation, eliminates special cases in the evaluation of the differential operators, and it further improves parallelism in the GPU implementation by guaranteeing that *all* of the resulting matrix-vector products have identical dimensions.

Note that the order of accuracy of the spatial operator is unaffected—using the same number of matching points $d = 5$ at all physical boundaries, the only additional approximation introduced by segmentation of the domain is accurate to *twelfth order*.

Remark 4.2.2. If n_s is sufficiently small, it is preferable to explicitly construct a matrix form of the FC operator, which may then be applied in $\mathcal{O}(n_s^2)$ operations per segment, and thus $\mathcal{O}(Nn_s)$ operations overall per time-step. This has been found to yield superior performance, as compared to the FFT-based approach, for parameter values $n_s \leq 40$, though in practice the break-even value of n_s depends on the hardware used, as well as the

availability of high-performance, small-sized FFT implementations. Special care must be taken in constructing the matrix form, however, or else some numerical precision is lost. In particular, subtractive cancellation effects may be avoided by precomputing the matrix using higher-precision arithmetic—an inexpensive initialization step requiring only $\mathcal{O}(n_s^2)$ operations.

4.2.1 Stability

When computed via a segmented application of FC(Gram), the filtering operator S_σ is no longer defined continuously over the entire domain. It therefore becomes necessary to consider the action of the operator S_σ on the d_{in} points shared by two neighboring segments. Even though this filter serves to dampen higher-frequency oscillations, the Fourier series expansions are necessarily different between the two segments, and therefore the filtered unknown u_σ inside of the overlap may disagree when computed from the right or from the left. Since half of the point values of the solution in the overlap are computed from respective Fourier series in the two adjoining segments, the filtering procedure typically introduces a discontinuity within each segment of a size comparable with the numerical error of the solution. In light of the comparatively mild filter used in the FC solvers considered in this thesis, see Chapter 3, this discontinuity is very small, and the methods resulting from the segmented FC method retain numerical stability. For the filter parameter value $\alpha = -\ln 10^{-16}$ used in [1], on the other hand, a larger mismatch caused by the filter between solutions in neighboring segments occurs and, as has been observed with the solvers presented in this work, the method can become unstable. Thus, the mild filter parameters $\alpha = -8\mu \ln 10^{-2}$ are used for all of the numerical results presented in this thesis.

4.2.2 Dispersion

The introduction of segmentation could conceivably have a negative impact on the excellent numerical dispersion properties of the FC method. Indeed, a traveling wave solution crosses Gram polynomial matching regions $\mathcal{O}(N^{1/3})$ times in three-dimensional space (albeit with a small constant of proportionality) as opposed to the $\mathcal{O}(1)$ crossings inherent in the original unsegmented algorithm. To address this concern, studies of dispersion error similar to those presented in Section 3.2.3 are presented here but using the segmented operator instead, including several examples of segmented and unsegmented FC spatial operators

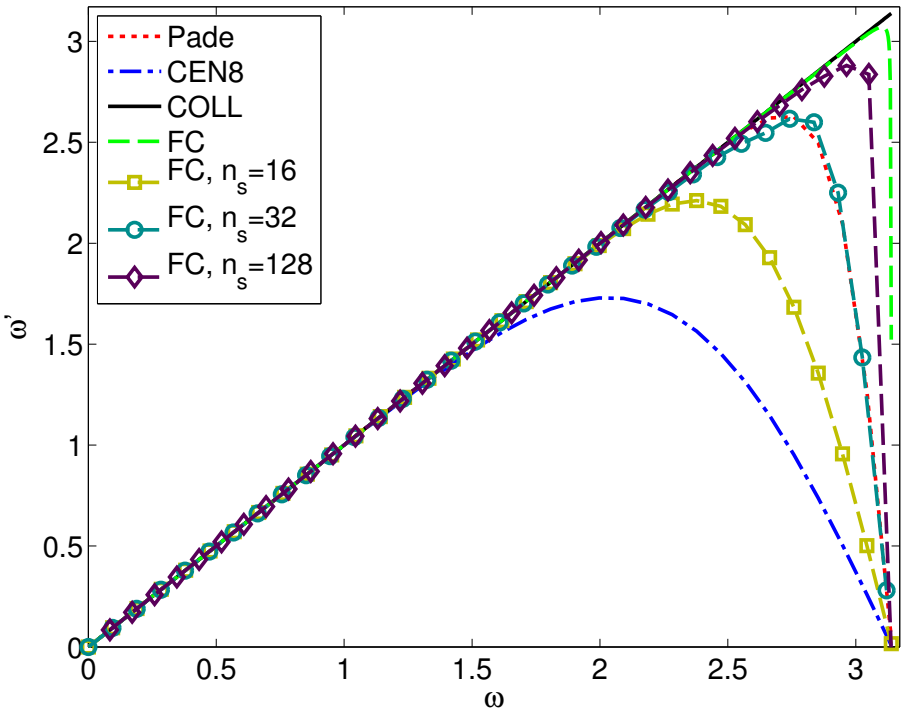


Figure 4.2: Expanded plot of dispersion relations, now including the behavior of segmented FC, for n_s equal to 16, 32, and 128. A cursory visual inspection suggests that the relation for $n_s = 32$ is comparable to that of the Pade scheme. In fact, the convergence of the corresponding FC solution is significantly faster than that of the corresponding Pade solution.

corresponding to values of n_s equal to 16, 32, and 128. Figure 4.2 shows an expanded set of numerical dispersion relations (comparable to Figure 3.9) which include results for the segmented FC variants. A cursory inspection might suggest that the $n_s = 32$ scheme possesses dispersion characteristics similar to those associated with the spectral-like Pade scheme [48], and that the $n_s = 16$ segmented scheme is significantly more dispersive. As it happens, however, the graphical deviations from the exact line $\omega' = \omega$ are insufficient to fully judge the dispersions produced by the algorithm over long distances (cf. Section 3.2.3 for a comparable discussion concerning the unsegmented algorithm). A discussion of the true dispersion character of the segmented FC algorithm is presented in what follows.

An error estimate analogous to the one shown in Figure 3.10, that characterizes the dispersion character of the segmented FC method more precisely than Figure 4.2, is given by equation (3.19) and displayed in Figure 4.3. This estimate demonstrates that all three of the segmented operators considered above ($n_s = 16$, 32 and 128) retain most of the

dispersionless character of the unsegmented FC method. While some precision is certainly lost relative to the unsegmented FC scheme, all three of the segmented approaches considered here achieve errors of $1 \cdot 10^{-4}$ to $1 \cdot 10^{-5}$ at discretizations of approximately 10 points per wavelength ($(\omega \approx 0.5)$ for which the spectral-like Pade method and the eighth-order centered difference algorithm produce an accuracy of no more than one digit.

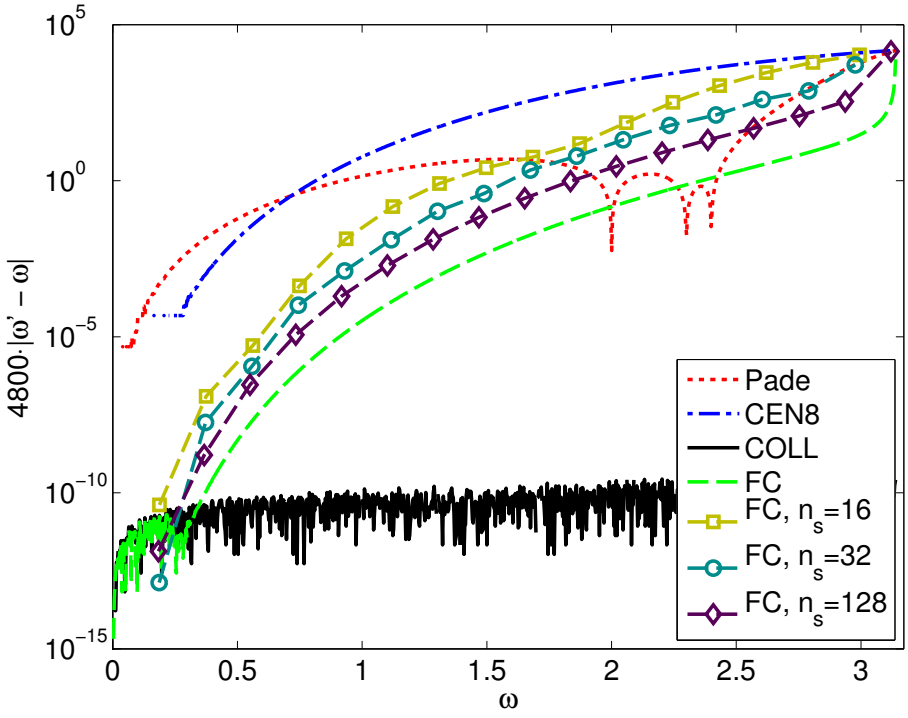


Figure 4.3: Error estimate $4800|\omega'_k - \omega_k|$ for the model advection problem as a function of ω, including behavior of the segmented FC methods for values of n_s equal to 16, 32, and 128

4.2.3 Relative performance

As mentioned in the previous section, the segmented FC method is somewhat less accurate than its unsegmented counterpart, and the question thus arises as to what is the cost that would be incurred, say, in a single processor, to produce by means of segmented FC method the accuracy resulting from the unsegmented approach.

To address this issue it may be noted, for example, that numerical experiments consistently show that the segmented approach with $n_s = 32$ requires 1.4 times as many points to achieve the same numerical error as the unsegmented approach (cf. Figure 4.3). In three dimensions, the denser sampling required for the segmented algorithm to reach an

error comparable to that resulting from the unsegmented method would lead to an overall increase in computing time by a factor of $1.4^4 \approx 3.84$. However, this cost factor is mitigated by the fact that the segmented operator can be evaluated more efficiently. Indeed, the unsegmented FC operator requires roughly 0.5 CPU seconds, in a modern CPU, to evaluate a numerical derivative with respect to a single variable at one million points in two- and three-dimensional space (1000×1000 and $100 \times 100 \times 100$ points in two- and three-dimensional space, respectively) [18]. The segmented approach, in turn, can be applied to the same problem in shorter computing time: approximately 0.16 seconds per million unknowns on the same hardware. Taking both factors into account, the segmented approach thus requires an increased computing time by a factor of $1.4^4 \times \frac{0.16}{0.5} \approx 1.23$ over the unsegmented algorithm, to produce the same accuracy. At such a low overhead, the segmented approach enables a sufficiently fine-grained level of parallelism to allow for efficient execution on GPU infrastructures.

Chapter 5

Multipatch scattering solver

The FC numerical solvers presented in Chapters 3 and 4 can be generalized to any number of spatial dimensions. However, in contrast with the one-dimensional case, computational boundaries in higher dimensions can give rise to significant complexity and must be treated adequately. This chapter 1) outlines the standard form of hyperbolic problems in multiple spatial dimensions, 2) presents an extension of the FC operators to the multidimensional context which allows for the use of curvilinear coordinates, and 3) presents an overset grid strategy which can be used to decompose the domain into a number of overlapping, possibly curved patches, along with a procedure for enforcement continuity and smoothness of solutions across the artificial boundaries.

5.1 Hyperbolic problems in multiple spatial dimensions

The present chapter introduces extensions of the one-dimensional PDE solvers put forth in Chapter 3 to general hyperbolic systems of the form

$$u_t + \sum_{j=1}^{d} A_j(x) u_{x_j} = f(x,t) \tag{5.1}$$

in d-dimensional spatial domains Ω ($d > 1$). Once again, the Cauchy problem is locally uniquely solvable given initial conditions on a noncharacteristic surface. A necessary and sufficient condition for hyperbolicity for equation (5.1) is the diagonalizability of any (non-trivial) linear combination of the matrices $A_j(x)$, for all x [28].

For example, the traditional second-order scalar form of the linear acoustic wave equa-

tion

$$u_{tt} + c^2 \nabla^2 u = 0 \tag{5.2}$$

can easily be expressed as a hyperbolic system. Indeed, following the corresponding calculation for the one-dimensional equation (3.4), here a d-component *vector* quantity $\mathbf{v}$ is introduced along with the system

$$\begin{aligned} u_t &= -c \nabla \cdot \mathbf{v} \\ \mathbf{v}_t &= -c \nabla u, \end{aligned} \tag{5.3}$$

which is clearly equivalent to equation (5.2). Taking $d = 3$ as an example, and defining the 4-component vector unknown

$$\mathbf{u} = \begin{pmatrix} u \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{pmatrix}, \tag{5.4}$$

the system may be written in the form

$$\mathbf{u}_t + A_1 \frac{\partial \mathbf{u}}{\partial x_1} + A_2 \frac{\partial \mathbf{u}}{\partial x_2} + A_3 \frac{\partial \mathbf{u}}{\partial x_3} = 0 \tag{5.5}$$

where the matrices A_j are given by

$$A_1 = \begin{pmatrix} 0 & -c & 0 & 0 \\ -c & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad A_2 = \begin{pmatrix} 0 & 0 & -c & 0 \\ 0 & 0 & 0 & 0 \\ -c & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad A_1 = \begin{pmatrix} 0 & 0 & 0 & -c \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -c & 0 & 0 & 0 \end{pmatrix} \tag{5.6}$$

and therefore the system is hyperbolic (as anticipated), and, in fact, symmetric-hyperbolic, that is, the matrices A_j are real symmetric for all j.

5.2 Generalized numerical operators

To extend the fine-grained parallel-scalable FC-type operators described in Chapter 4 to the present d-dimensional context, a numerical operator D_1 is introduced, which applies

the corresponding FC derivative operator along each line parallel to the first spatial dimension. Analogous operators D_j are defined, for $2 \leq j \leq d$. These operators compute spatial derivatives with respect to the local parameterization in each patch, however, so if curvilinear coordinates are used, the chain rule must be applied in order to construct numerical approximations of the true spatial derivatives. As an example with $d = 3$, a numerical approximation of the gradient of a scalar function u within a given patch takes the form

$$\nabla u \approx J^{-1} \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix} u \tag{5.7}$$

and, similarly, the divergence of a vector unknown $\mathbf{v}$ may be computed by

$$\nabla \cdot v \approx \sum_{j=1}^{d} e_j^T J^{-1} \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix} \mathbf{v}_j \tag{5.8}$$

where e_j is the j-th standard basis vector, and J^{-1} is the inverse Jacobian matrix corresponding to the local coordinate system.

For some particular curvilinear coordinates (including those arising from use of spherical coordinates), it is possible to calculate $\nabla \cdot v$ in fewer than d^2 applications of an FC-type operator over lines in the domain, since some of the components of the vectors $e_j^T J^{-1}$ may be zero. In the interest of generality, however, the implementation presented here ignores such special cases, instead supporting only the two extremes—an arbitrary change of variables, for which no assumptions beyond smoothness are made, and *no* change of variables, where $J^{-1} = I$ and, hence, only d applications of such an FC-type operator are required.

Finally, it is necessary to describe the filtering algorithm in the present setting. To do this, let $S_1, \ldots, S_d$ be the one dimensional filtering operators in the directions $x_1, \ldots, x_d$, respectively. Note that, since each parameter patch is discretized as a cube within its own coordinates, the subdomain is separable and the smoothing operators all commute with one another. Having established that the order of filtering operations is irrelevant, smoothing over each patch may be achieved by a simple composition of the individual smoothing

operators,

$$S = \prod_{j=1}^{d} S_j \approx I \tag{5.9}$$

Certain special cases will be introduced in Section 5.5 which do not preserve this separability property, resulting in filtering operations which no longer commute. Experimentally, however, this choice of a patch-wise spatial filter remains sufficient to stabilize the FC solver without accuracy deterioration.

5.3 Domain decomposition

For most computational domains of interest, a single, rectangular Cartesian patch is insufficient to capture both the domain and a possibly complex boundary. By taking the overset grid approach (see [12] and references therein), a given domain can be decomposed into a number of overlapping patches, a local coordinate transform may be chosen so that each patch is logically a simple domain (such as a cube) within its own parameter space, but conforms to an arbitrary (smooth) boundary or intervening space as needed.

As long as the scattering boundary itself is smooth, it is always possible to decompose a tubular neighborhood thereof in this fashion. These conforming patches are typically then embedded into a larger Cartesian patch, as demonstrated for a simple circular boundary in Figure 5.1. It should be noted that boundaries with edges, corners, or relatively sharp curvatures pose some difficulty, since sufficiently fine discretizations of the neighborhood of these features may have unfavorable consequences with respect to the global CFL condition. A resolution for this issue, based on use of temporal subcycling, is discussed in Section 8.1.3.

5.4 Patch interpolation

The setting outlined in Section 5.3 obviates the need to use domain-cutting operations that are often prohibitively complex. A significant difficulty that arises in this context, however, concerns enforcement of a sufficient degree of continuity and smoothness in the patch-overlap regions. For the one-dimensional segmented operator introduced in Chapter 4, it is convenient that the mesh points for adjacent segments line up *exactly* (which they do, as they are defined as overlapping, contiguous subsets of the same base mesh). In the present patch-interpolation context, in contrast, the conditions are not quite so favorable—but a

similar approach can be used. For some integer parameter p, the subset of mesh points in a given patch that lie within a distance $(p-1)\Delta x$ (measured in the local coordinate system) of a patch boundary that does *not* coincide with a true physical boundary (or computational boundary of the overall domain) are called "fringe" points. These fringe points do not correspond to true degrees of freedom in the solver, but are implicitly defined by a value interpolated from the patch for which that point is most internal. The overlaps between the patches is chosen to be large enough that each fringe point in a given patch is "sufficiently internal" to some other patch, that is there exists another patch, containing that point, but for which said point does *not* lie in its fringe region.

At each time-step, after the solution in each patch has been evolved independently of the evolution in other patches, the unknown at each fringe point is re-assigned by interpolating from an appropriate neighboring patch. These operations are performed by means of polynomial interpolation of a certain order r, where the order is chosen to match the order of the boundary matching Gram polynomials, and, thus, the overall spatial order of the solver. A multidimensional version of Neville's algorithm is employed for this task. In addition, a careful choice of interpolation domain is made in order to allow as many fringe points as possible to share the same set of $(r+1)^d$ sample mesh points. This is most easily accomplished by dividing the patches a priori into interpolation domains of r intervals to a side, and then simply grouping target points by interpolation domain. This approach has been found to be especially useful in the GPU implementation of this method. Even though the number of floating point operations is naturally greater (by roughly a factor of r) than it would be if precomputed weights were used for interpolation, as is done in [12], the memory overhead is reduced quite significantly, increasingly so as the interpolating polynomial degree r is increased. Across the various experiments within this work, memory overhead has been reduced typically by a factor of 50 using this optimization, as compared to a naive implementation.

5.5 Complex interfaces

The overlapping patch methodology presented thus far can be further extended to support hole cutting—so that not every patch needs to be discretized as the entirety of a cube within its respective parameter space.

Due to the nature of the FC operators which act only along lines in parameter space, the validity of the remaining mesh (once arbitrary holes have been cut) can be qualified by the following: each point that remains on the computational mesh must, along each cardinal direction in the local parameter space, be a member of a sequence of n_s contiguous mesh points corresponding to a computational segment. Any convex hole which is sufficiently interior (by n_s points), for example, satisfies this criterion trivially. The hole cut need not even be smooth, and in most cases where this condition would be violated, the hole need only be enlarged slightly in order to accommodate the condition.

The concept of fringe region must also be extended to include a sufficient layer of points around any cut holes. The same concept used previously to define fringe points—that is, a fringe point in a patch is any discretization point in that patch within a distance equivalent to r discretization points of the patch boundary—similarly now holds for cut holes.

To illustrate the hole-cutting/overlapping-patch methodology consider the following two-dimensional example. Let the PDE domain Ω be given by

$$\Omega = \{x \mid \|x\|_\infty \leq 1 \quad \text{and} \quad \|x\|_2 \geq 0.26\} \tag{5.10}$$

which corresponds to a square of side 2 with a circular hole of radius 0.26, both centered at the origin. This may be decomposed into overlapping patches $\Omega_1 \ldots \Omega_5$,

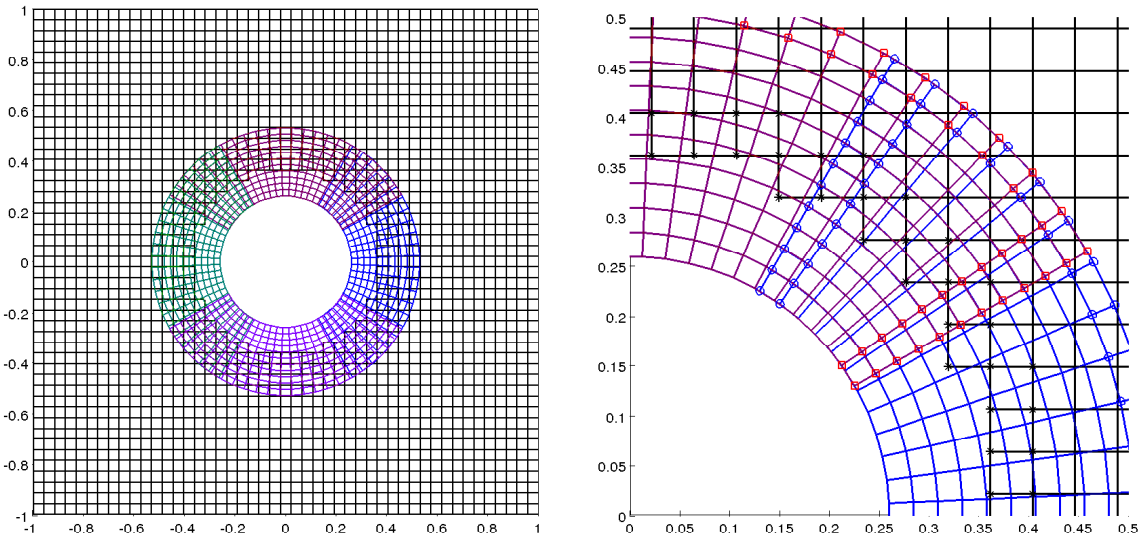
$$\Omega = \bigcup_{j=1}^{5} \Omega_j, \tag{5.11}$$

as described in what follows.

The patch Ω_1 is given by $\Omega_1 = \{x \mid \|x\|_\infty \leq 1 \quad \text{and} \quad \|x\|_2 \geq 0.4\}$ with a Cartesian mesh. The patch Ω_1 resembles Ω, but its hole is somewhat larger—to ensure that a neighborhood of the interior boundary $\|x\|_2 = 0.26$ is excluded. The patches Ω_2 through Ω_5, in turn, are polar coordinate domains which cover a neighborhood of the interior physical boundary; they are given by

$$\Omega_j = \left\{ (r\cos\theta, r\sin\theta) \mid r \in [0.26, 0.53] \text{ and } \theta \in \left[\frac{-\pi}{3}, \frac{\pi}{3}\right] + \frac{(j-1)\pi}{2} \right\}, \quad j = 1 \ldots 4. \tag{5.12}$$

Figure 5.1(a) displays the resulting overlapping-patch decomposition together with a



(a) Five-patch parameterization (b) Detailed view, fringe points marked by small circles

Figure 5.1: Sample discretization of a 2d box domain containing a circular internal boundary

discretization of the corresponding patches for which the largest grid spacing on the polar patches matches the uniform spacing on the Cartesian patch, and such that the radial spacing and the smallest angular spacing on the polar patches coincide. Finally, in Figure 5.1(b), a set of fringe points (arising from the fringe parameter $r = 1$, which, for improved visibility, is taken much smaller than used in practice) are shown in closer detail for three of the patches.

Part III

Numerical boundary conditions for unbounded domains

Chapter 6

Kirchhoff's integral formula

Kirchhoff's integral formula [7] provides an analytic interpretation of Huygens's principle for wave motion in three dimensions. To introduce this formula, following the conventions of [7], the retarded value $[u]$ of a function $u(x,t)$ of position and time is used, where for any point y in space, and letting $r = \|x-y\|_2$, $[u]$ is defined by the expression

$$[u] = u(y, t - \frac{r}{c}). \tag{6.1}$$

In other words, for fixed (x,t) values, $[u](y)$ gives the value taken by the field u at the position y at a time r/c before the "present" time t.

Equipped with this notation, Kirchhoff's integral formula states that, for the domain exterior to any closed surface $\mathcal{S}$ that encloses all sources and inhomogeneities, a radiating solution to the wave equation may be expressed in the form

$$u(x,t) = \frac{1}{4\pi}\int_{\mathcal{S}} \left\{ [u]\,\frac{\partial}{\partial n}\left(\frac{1}{r}\right) - \frac{1}{cr}\frac{\partial r}{\partial n}\left[\frac{\partial u}{\partial t}\right] - \frac{1}{r}\left[\frac{\partial u}{\partial n}\right] \right\} ds(y) \tag{6.2}$$

where $\frac{\partial}{\partial n}$ is the derivative in the direction of the outward-facing normal. For x outside of $\mathcal{S}$ this integral explicitly yields the solution u at the point (x,t) strictly in terms of its causal dependencies; for x inside $\mathcal{S}$, in turn, the Kirchhoff integral vanishes identically. Clearly, this integral expression may be used to evaluate the solution u at the boundary of a finite computational domain and, as shown in Chapter 7, a corresponding fast high-order convergent algorithm for truncation of the computational domain can thus be devised (but see Figure 6.2 and associated text).

In order to fully take advantage of this integral representation of the solution at the

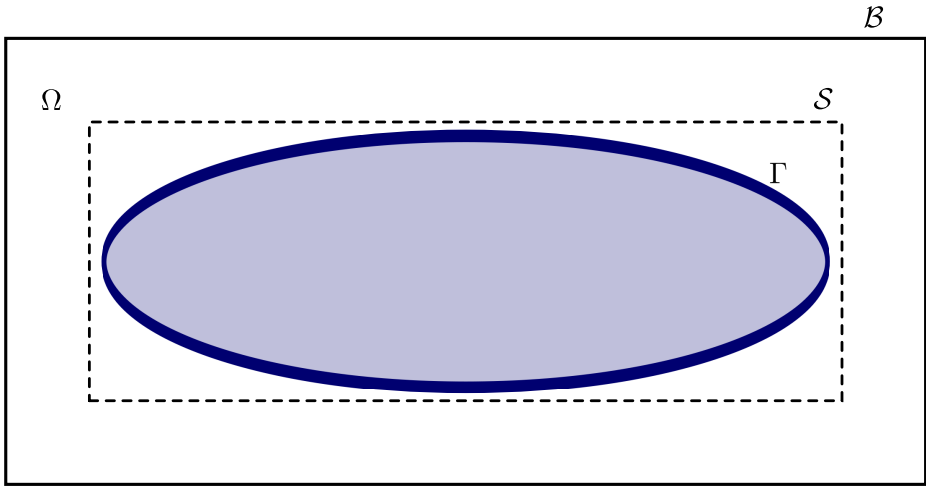


Figure 6.1: Illustration of a computational domain Ω enclosing an ellipsoidal scattering boundary Γ, along with the artificial boundary $\mathcal{B}$, on which the radiating solution may be computed via an integral over the intermediate Kirchhoff surface $\mathcal{S}$

boundary, two important issues must be considered: 1) It must be determined how such boundary conditions may be used in such a way that the resulting algorithm is stable; and 2) An accurate and efficient algorithm must be used for the numerical evaluation of the Kirchhoff integrals: a naive approach would lead to an expensive methodology, for which the evaluation of the boundary conditions would dominate the overall computational costs in terms of both computing time and memory.

This chapter investigates problem 1) in a one-dimensional context, and it introduces a boundary condition at the computational boundary for which stability can be expected in the overall three-dimensional time-domain solver. A discussion in Chapter 7 then addresses problem 2) by evaluation of the necessary integrals via an excursion into the frequency domain—without the accuracy losses that typically arise from the Gibbs phenomenon.

6.1 One-dimensional interpretation

Following [42], a boundary condition similar to equation (6.2), which is useful in the determination of computational boundary conditions leading to stability, is established in what follows for the case of the one-dimensional wave equation in the semi-infinite domain $[0, \infty)$. For definiteness, zero initial conditions are prescribed in conjunction with the

"physical boundary condition"

$$u(0,t) = f(t), \quad t \geq 0; \tag{6.3}$$

for consistency it is required that $f(t) = 0$ for $t \leq 0$.

The combination of equation (3.4) with the initial and boundary conditions introduced above admits only the right-moving solution

$$u(x,t) = \begin{cases} f(t-x) & \text{if } \ t-x \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6.4}$$

The solution of this problem via numerical methods requires truncation of the PDE domain to some bounded interval $[0, x_{\mathcal{B}}]$, and, thus, an additional *numerical* boundary condition at $x = x_{\mathcal{B}}$—which, generically, may be expressed in the form

$$\mathcal{L}u(x_{\mathcal{B}}, t) = g(t) \tag{6.5}$$

for some operator $\mathcal{L}$ and some function $g(t)$.

The evaluation of the expression (6.5) requires knowledge of u (and possibly its derivatives) at the point $x = x_{\mathcal{B}}$. Such values can be obtained by means of the Kirchhoff formula, the one-dimensional version of which is particularly simple: using a "Kirchhoff point" $x_{\mathcal{S}} \in (0, x_{\mathcal{B}})$ (that, substituting for the Kirchhoff surface $\mathcal{S}$, separates $x_{\mathcal{B}}$ from all sources) the Kirchhoff formula takes the form of the delayed potential

$$u(x_{\mathcal{B}}, t) = u(x_{\mathcal{S}}, t - \frac{x_{\mathcal{B}} - x_{\mathcal{S}}}{c}). \tag{6.6}$$

This expression could in principle be used directly as a computational boundary condition (thus taking $\mathcal{L} = I$, taking $x_{\mathcal{S}}$ to coincide with a mesh point, and selecting the time-step Δt in such a way that it evenly divides $\Delta x/c$). With such a computational boundary condition, the boundary value at $x_{\mathcal{B}}$ coincides with the value of the numerical solution at $x_{\mathcal{S}}$ and at a certain number of time-steps prior to the present time t.

To explore the properties of these computational boundary conditions, the one-dimensional FC solver introduced in Chapter 3 is used in the interval $[0, x_{\mathcal{B}}]$, with $x_{\mathcal{B}} = 1$; for definite-

ness a number $N = 128$ of discretization points is used in the computational domain $[0, 1]$. A "Kirchhoff point" is then placed at $x_{\mathcal{S}} = x_{\mathcal{B}} - n_{\text{diff}}\Delta x$, for some integer parameter n_{diff}, and a time-step of $\Delta t = \Delta x/16$ is used. This ensures that the boundary data depends explicitly on data at a discretization point from a discrete, integer number of time-steps prior to the current time t.

Unfortunately, the use of the boundary condition in this form does *not* give rise to a stable numerical scheme: Figure 6.2 demonstrates the high-frequency oscillations that develop in the corresponding numerical solution. This is in agreement with the behavior observed in [32, 42]. In [32] it is suggested that a dissipative interior scheme can be used to alleviate this difficulty. In order to preserve the high-order convergence and near-dispersionless character of the FC solvers, however, a different approach, following [42] and based on consideration of the Sommerfeld radiation condition, is taken.

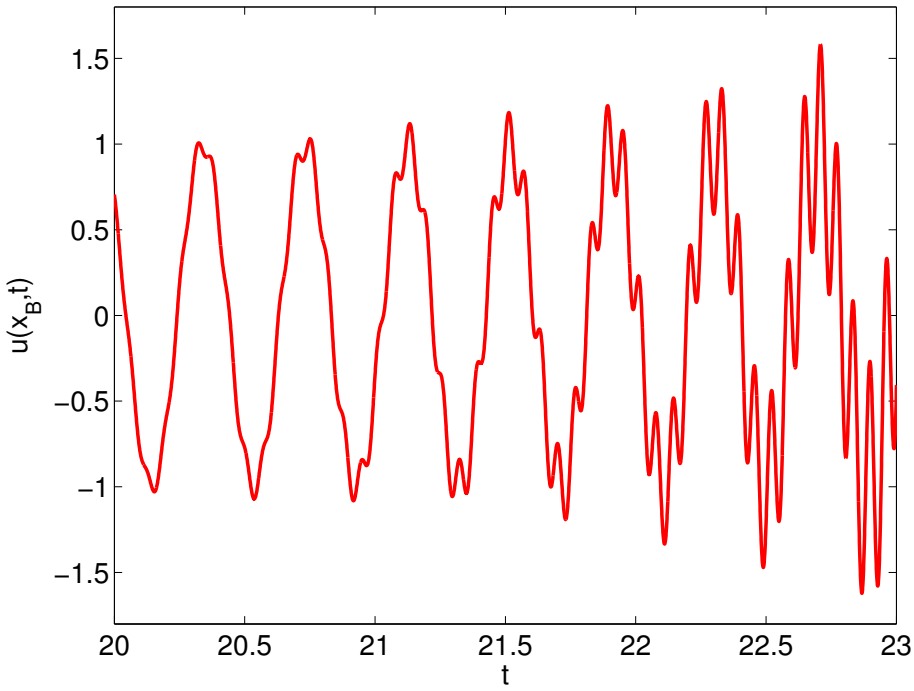


Figure 6.2: The Kirchhoff-like boundary condition applied in Dirichlet form results in a long-time instability at the boundary $x = x_{\mathcal{B}}$.

In the present one-dimensional context, Sommerfeld's radiation condition reads

$$\left(\frac{1}{c}\frac{\partial}{\partial t} + \frac{\partial}{\partial x}\right) u|_{x=x_{\mathcal{B}}} = 0. \tag{6.7}$$

Rather than apply this directly at the boundary (which would have limited use when generalized to problems in multiple spatial dimensions), equation (6.7) can be used as the basis for a boundary operator $\mathcal{L}$. Taking some linear combination of a differential operator of this form with a Dirichlet condition results in

$$\mathcal{L}_\alpha = \alpha + \frac{1}{c}\frac{\partial}{\partial t} + \frac{\partial}{\partial x}, \tag{6.8}$$

with respect to the parameter α, which must be nonzero if a Neumann boundary condition is to applied at $x = 0$—otherwise the solution is not unique, as any constant on the interior satisfies both the wave equation and the resulting boundary condition.

In order to apply the boundary operator $\mathcal{L}_\alpha u = g$ to the one-dimensional FC solver, the boundary equation (6.5) may be expanded and rearranged to yield

$$u_t(x_\mathcal{B}, t) = c\left\{g(t) - \alpha u(x_\mathcal{B}, t) - \frac{\partial u}{\partial x}(x_\mathcal{B}, t)\right\} \tag{6.9}$$

where

$$g(t) = \mathcal{L}_\alpha u(x_\mathcal{S}, t - \frac{x_\mathcal{B} - x_\mathcal{S}}{c}). \tag{6.10}$$

Rather than project the boundary discretization points to some Dirichlet boundary value, equation (6.9) yields an equation for the time derivative at the boundary. This boundary equation is then integrated using the same time integration scheme as the rest of the solver.

The unstable solution displayed in Figure 6.2 is stabilized with this Sommerfeld-type boundary operator if, for $n_{\text{diff}} > 2$, a system of overdetermined matching Gram polynomials is used—the same $d = 5$ matching points are used near each boundary, but only the first four basis polynomials are extended to create the continuation. This results in a reduction of the spatial accuracy by precisely one order. In addition, the range of values of α that gives rise to stability, which are summarized in Table 6.1, is slightly narrower than the corresponding range observed in [42], but the approach presented here still possesses a significantly higher order of accuracy (fourth as opposed to second order, and full spectral accuracy in the domain interior).

More importantly, this complication does not arise in the context of three-dimensional solvers, for which no reduction of order or over-determination of the matching polynomials is required, as is demonstrated by the examples in Chapter 9. The generalization of

n_{diff}	Stable range for α
1	$[-0.05, 1.75]$
4	$[-0.05, 1.5]$
16	$[-0.05, 1]$

Table 6.1: Range of values of α leading to stability in the FC solver for the 1d wave equation in first-order system form

equation (6.8) suggested in [42] to an arbitrary number of spatial dimensions is

$$\mathcal{L}_\alpha = \alpha(x) + \frac{1}{c}\frac{\partial}{\partial t} + \mathbf{d}(x) \cdot \nabla, \tag{6.11}$$

where the vector $\mathbf{d}(x)$ is taken as the outward-facing unit normal on the surface $\mathcal{B}$ at the point x, and where the scalar field $\alpha(x)$ given by

$$\alpha(x) = \frac{\mathbf{d}(x) \cdot x}{\|x\|_2^2}. \tag{6.12}$$

(But see also Section 9.3: a different choice of $\mathbf{d}(x)$ is found to give rise to a more favorable CFL condition in the context of the FC solver in cases in which the computational boundary contains corner points.)

Chapter 7

FC-ES: Equivalent source algorithm for numerical boundary conditions

For the fully three-dimensional wave equation, Kirchhoff's integral formula (6.2) depends on values of the solution u over the surface $\mathcal{S}$ for a continuous interval of time. In order to use this formula to evaluate computational boundary conditions, the Kirchhoff integral expression must be computed accurately and efficiently for each point on the boundary surface $\mathcal{B}$. A direct evaluation of the necessary values of the Kirchhoff integral would be exceedingly expensive—as it would require integration over $\mathcal{O}(N^{2/3})$ source points for each one of $\mathcal{O}(N^{2/3})$ observation points, resulting in an overall computational complexity of $\mathcal{O}(N^{4/3})$ for the boundary condition algorithm—which would exceed the computational cost of the interior solver, and thus would dominate the total computing time.

In order to avoid the excessive costs inherent in a direct evaluation of Kirchhoff integrals, this chapter introduces a transformation of the time-domain integral (6.2) to the frequency-domain, using the methods of Chapter 2, without the accuracy losses that are typically associated with the Gibbs phenomenon. In addition this chapter provides a fast algorithm, based on the equivalent source method [16], for evaluation of the frequency-domain integrals arising from the Kirchhoff formula, similar to the approach first described in [42]. Special consideration is given to parallel implementation and algorithmic performance under single precision arithmetic, so as to allow for fast and accurate execution on GPUs.

7.1 Expression of the Kirchhoff integral in the frequency domain via Fourier continuation

The evaluation of the three-dimensional Kirchhoff integral formula, at some time t_0, requires knowledge of the solution $u(x,t)$ for $x \in \mathcal{S}$ over some finite interval in time. In detail, defining $R_{\min}$ as the shortest distance between points in $\mathcal{S}$ and points in $\mathcal{B}$,

$$R_{\min} = \inf_{\substack{x\in\mathcal{S}\\ y\in\mathcal{B}}} \|x-y\|_2, \tag{7.1}$$

and, similarly, $R_{\max}$ as the longest distance between points in $\mathcal{S}$ and points in $\mathcal{B}$,

$$R_{\max} = \sup_{\substack{x\in\mathcal{S}\\ y\in\mathcal{B}}} \|x-y\|_2, \tag{7.2}$$

the boundary condition on $\mathcal{B}$ at time t_0 depends on past values of the solution on the surface $\mathcal{S}$ over the temporal interval

$$\left[t_0 - \frac{R_{\max}}{c}, t_0 - \frac{R_{\min}}{c}\right]. \tag{7.3}$$

Therefore, to make use of equation (6.2), a history of values on the surface $\mathcal{S}$ extending $\frac{R_{\max}}{c}$ into the past is required. Moreover, since each boundary point on $\mathcal{B}$ requires integration over the intersection of $\mathcal{S}$ with a different past-light cone, a direct implementation of equation (6.2) would require interpolation of the (discrete) time histories to arbitrary times in the time interval (7.3) (as is done in [32]). Finally, with $\mathcal{O}(N^{2/3})$ discretization points on the surfaces $\mathcal{S}$ and $\mathcal{B}$, naive integration would require a costly $\mathcal{O}(N^{4/3})$ operations, and quickly dominate the computational effort of any interior solver—which is in fact linear $\mathcal{O}(N)$ for the FC method presented in this work.

These difficulties are alleviated by re-expressing equation (6.2) in terms of frequency-domain integrals, as indicated in what follows. Using the previously introduced Fourier continuation method, a time-periodic FC function $\tilde{u}$ is constructed which agrees with the original function u in the time interval $[t_0 - \frac{R_{\max}}{c}, t_0 - \frac{R_{\max}}{c} + T_0]$ for a certain duration T_0, and which extends it smoothly for an additional time interval T_c, resulting in a periodic function with period $T = T_0 + T_c$.

Provided $T_0 > \frac{R_{\max}-R_{\min}}{c}$ is sufficiently large, $\tilde{u}$ agrees with u in the interval (7.3), and

$\tilde{u}$ can therefore be used instead of u in the Kirchhoff integral formula for $t = t_0$. More precisely, if $\tilde{u} = u$ for the interval $[t_0 - \frac{R_{\max}}{c}, t_0 - \frac{R_{\max}}{c} + T_0]$, then the Kirchhoff integral of $\tilde{u}$ agrees with that of u on the boundary $\mathcal{B}$ for times $t \in [t_0, t_0 + T_0 - \frac{R_{\max} - R_{\min}}{c}]$. This interval is nonempty as long as the lower bound (stated above) on T_0 is satisfied. For practical usage in explicit time-marching methods, the evaluation of the numerical boundary condition must depend only on the values of the interior solution already computed, corresponding to those at present or past time-steps—consequently, the choice $T_0 = \frac{R_{\max}}{c}$ results in the largest possible interval in time for which the periodically continued $\tilde{u}$ may be used for the evaluation of the Kirchhoff-based computational boundary conditions. This selection of T_0 is also the *optimal* choice with respect to efficiency. This boundary expansion has cost proportional to T_0, and thus has a relative computational efficiency (again with respect to T_0) determined by the relative rate with which it must be recomputed, quantified by

$$\frac{\min\left\{\frac{R_{\min}}{c}, T_0 - \frac{R_{\max} - R_{\min}}{c}\right\}}{T_0}, \tag{7.4}$$

which is clearly maximized for $T_0 = \frac{R_{\max}}{c}$. This process may be repeated over shifted intervals in time, offset by increments of $\frac{R_{\min}}{c}$ resulting in times $t_n = n\frac{R_{\min}}{c}$, at each instance matching $\tilde{u}$ to u on the surface $\mathcal{S}$ over the time-interval $[t_n - \frac{R_{\max}}{c}, t_n - \frac{R_{\max}}{c} + T_0]$, and remaining valid for the boundary integral at times $t \in [t_n, t_n + \frac{R_{\min}}{c}]$.

Equipped with the time-periodic approximation $\tilde{u}$, which has been shown to be interchangeable with u within the Kirchhoff representation for $t \in [t_n, t_n + \frac{R_{\min}}{c}]$, a Fourier series expansion of the integral formula (6.2) is now derived—by producing corresponding expansions for each one of the two components u_s and u_d defined by

$$\begin{aligned} u &= u_s + u_d \\ u_s &= -\frac{1}{4\pi}\int_{\mathcal{S}} \frac{1}{r}\left[\frac{\partial \tilde{u}}{\partial n}\right] ds \\ u_d &= \frac{1}{4\pi}\int_{\mathcal{S}} [\tilde{u}]\,\frac{\partial}{\partial n}\left(\frac{1}{r}\right) - \frac{1}{cr}\frac{\partial r}{\partial n}\left[\frac{\partial \tilde{u}}{\partial t}\right] ds. \end{aligned}$$

Expanding $u_s(x,t)$ in a Fourier series

$$u_s(x,t) = \sum_{k=-\infty}^{\infty} u_{s,k}(x)\; e^{2\pi i k t/T} \tag{7.5}$$

over the period $[0, T]$ followed by interchange of time and space integration yields

$$\begin{aligned} u_{s,k} &= \frac{1}{4\pi T}\int_0^T e^{-2\pi i k\frac{t}{T}}\int_{\mathcal{S}}\frac{1}{r}\left[\frac{\partial\tilde{u}}{\partial n}\right]dsdt \\ &= \frac{1}{4\pi T}\int_{\mathcal{S}}\frac{1}{r}\int_0^T e^{-2\pi i k\frac{t}{T}}\left[\frac{\partial\tilde{u}}{\partial n}\right]dtds \\ &= \int_{\mathcal{S}}\frac{e^{ikr/c}}{4\pi r}\cdot\frac{1}{T}\int_0^T e^{-2\pi i k\frac{t}{T}}\frac{\partial\tilde{u}}{\partial n}dtds \\ &= \int_{\mathcal{S}} G_{2\pi k/c}\frac{\partial\tilde{u}_k}{\partial n}ds, \end{aligned} \tag{7.6}$$

where $\tilde{u}_k$ are the Fourier coefficients of the Fourier continuation function $\tilde{u}$, and where $G_{2\pi k/c}$ is the free-space Green's function with wave-number $\nu = 2\pi k/c$:

$$G_\nu = \frac{e^{i\nu r}}{4\pi r}. \tag{7.7}$$

The integral u_d may be expanded in a similar fashion, yielding the double layer integral

$$u_{d,k} = \int_{\mathcal{S}}\frac{\partial G_{2\pi k/c}}{\partial n}\tilde{u}_{k,d}ds \tag{7.8}$$

whose kernel equals the normal derivative of the Green's function (7.7). (The x, y, and z derivatives needed for the evaluation of $\frac{\partial\tilde{u}}{\partial n}$ are computed via the FC method every time-step in the solver framework presented here.) The integrals (7.6) and (7.8) are computed, then, in the frequency domain, followed by an application of the inverse FFT to recover the desired time-series values on the boundary $\mathcal{B}$. It is worthwhile to note that, since the function $\tilde{u}$ is both smooth and periodic, the corresponding Fourier series representation converges very quickly, and the function $\tilde{u}$ can be approximated accurately using only a comparatively small number of Fourier modes.

The "detour" over the Fourier domain facilitated by the Fourier continuation method enables use of existing fast algorithms for evaluation of integrals of the form (7.6) and (7.8). As first demonstrated by [42], the equivalent source methodology [16] is particularly well-suited to this setting, allowing for the boundary expansions to be computed in sublinear time, with respect to the volumetric discretization of unknowns on the interior, as detailed in the following section.

It is worthwhile to note that, as a byproduct of the frequency domain formulation

introduced in this section, the need for interpolation in time mentioned in the paragraph following equation (7.3) is eliminated.

7.2 Acceleration of frequency-domain integrals

Given a suitable choice of integration nodes x_j and weights w_j (in the implementation considered in this thesis the nodes x_j correspond to FC mesh-points that lie on the Kirchhoff surface and the corresponding w_j are high-order Newton-Cotes weights, see Section 9.3), the integrals (7.6) and (7.8) are numerically evaluated by way of a discrete summation of the form

$$\psi(\mathbf{x}) = \sum_j w_j \left\{ \frac{\partial \tilde{u}_k(\mathbf{x}_j)}{\partial n(\mathbf{x}_j)} G_{2\pi k/c}(\mathbf{x}_j - \mathbf{x}) + \tilde{u}_k(\mathbf{x}_j) \frac{\partial G_{2\pi k/c}(\mathbf{x}_j - \mathbf{x})}{\partial n(\mathbf{x}_j)} \right\}. \tag{7.9}$$

Fast evaluation of these nonsingular integrals (sums) is achieved by means of a generalization of the acceleration strategy introduced in [16], based on the use of certain distributions of monopole and dipole "equivalent sources" on Cartesian grids and sparse 3D FFTs, as discussed in sections 7.2.1 through 7.2.2.

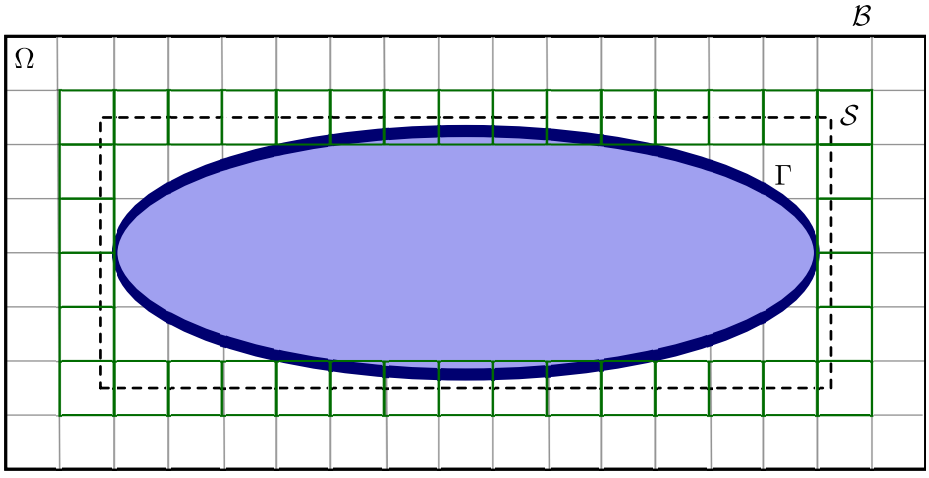


Figure 7.1: Decomposition of the rectangular parallelepiped C whose boundary, in this example, coincides with the computational domain boundary $\mathcal{B}$. Each cube c_i which intersects the Kirchhoff surface $\mathcal{S}$ (cubes shown in green) must be discretized by means of equivalent sources.

The method [16] relies on a partition of a rectangular parallelepiped C circumscribing the boundary $\mathcal{B}$ into $L_1 \cdot L_2 \cdot L_3$ identical cubic cells c_i of side H (as shown in Figure 7.1)

that do not admit inner acoustic resonances; the parameters are selected in such a way that the real number $-(2\pi k/c)^2$ is not a Dirichlet eigenvalue of the Laplace operator in the domain c_i.

The main elements in this acceleration algorithm are sets of "equivalent sources", which can be used to represent accurately, in large regions of space, the fields produced by the "true" surface sources (on Γ) contained in each cell c_i. As prescribed in [16], the equivalent sources that represent the fields generated by true sources contained in c_i are located on 2D Cartesian meshes Π_i^ℓ contained on circular neighborhoods (with radii slightly larger than half the diameter of c_i) of pairs of opposing faces of the cells c_i. Thus, the contributions to the discrete integral from discretization points contained in c_i are approximated, with high-order accuracy, by a number M^{eq} of equivalent sources placed on Π_i^ℓ—for all points in space nonadjacent to c_i, and for $l = 1, 2, 3$. The precise concept of adjacency used herein (namely, two cells c_i are adjacent if and only if they share a face, an edge, or a vertex) guarantees that the approximation used for a cell c_i is valid, with exponentially small errors, in the complement of the union of all cells c_j adjacent to c_i. Clearly, the union of c_i and all of the 26 cells c_j adjacent to it constitute a cubic region of side $3H$; in what follows, the boundary of the triple-size cubic region is denoted by S_i. Further, taking for each l the definition $\Pi^\ell = \bigcup_i \Pi_i^\ell$, it can be noted that Π^ℓ is a set of points on a Cartesian grid contained in the union of L_ℓ equispaced planes parallel to the plane $x_\ell = 0$.

7.2.1 Equivalent sources and FFTs

At each point in Π_i^ℓ, one acoustic monopole $\xi_{i,j}^{(m)\ell} G_{2\pi k/c}(x - x_{i,j}^\ell)$ and one acoustic dipole $\xi_{i,j}^{(d)\ell} \partial G_{2\pi k/c}(x - x_{i,j}^\ell)/\partial x_\ell$ for $j = 1, \ldots, M^{eq}/2$ are placed for a total of M^{eq} equivalent sources on Π_i^ℓ. The fields $\psi^{c_i,\text{true}}$ radiated by the c_i-true sources (that is, all surface sources on $\mathcal{S}$ contained within c_i) are approximated by fields $\psi^{c_i,\text{eq}}$ radiated by the c_i-equivalent sources, that is, by the expression

$$\psi^{c_i,\text{eq}}(x) = \sum_{j=1}^{\frac{1}{2}M^{eq}} \left(\xi_{i,j}^{(m)\ell} G_{2\pi k/c}(x - x_{i,j}^\ell) + \xi_{i,j}^{(d)\ell} \partial G_{2\pi k/c}(x - x_{i,j}^\ell)/\partial x_\ell \right). \tag{7.10}$$

For each ℓ, and for an adequately chosen number M^{eq} of equivalent sources supported on Π_i^ℓ, the unknown monopole and dipole intensities $\xi_{i,j}^{(m)\ell}$ and $\xi_{i,j}^{(d)\ell}$ in equation (7.10) are

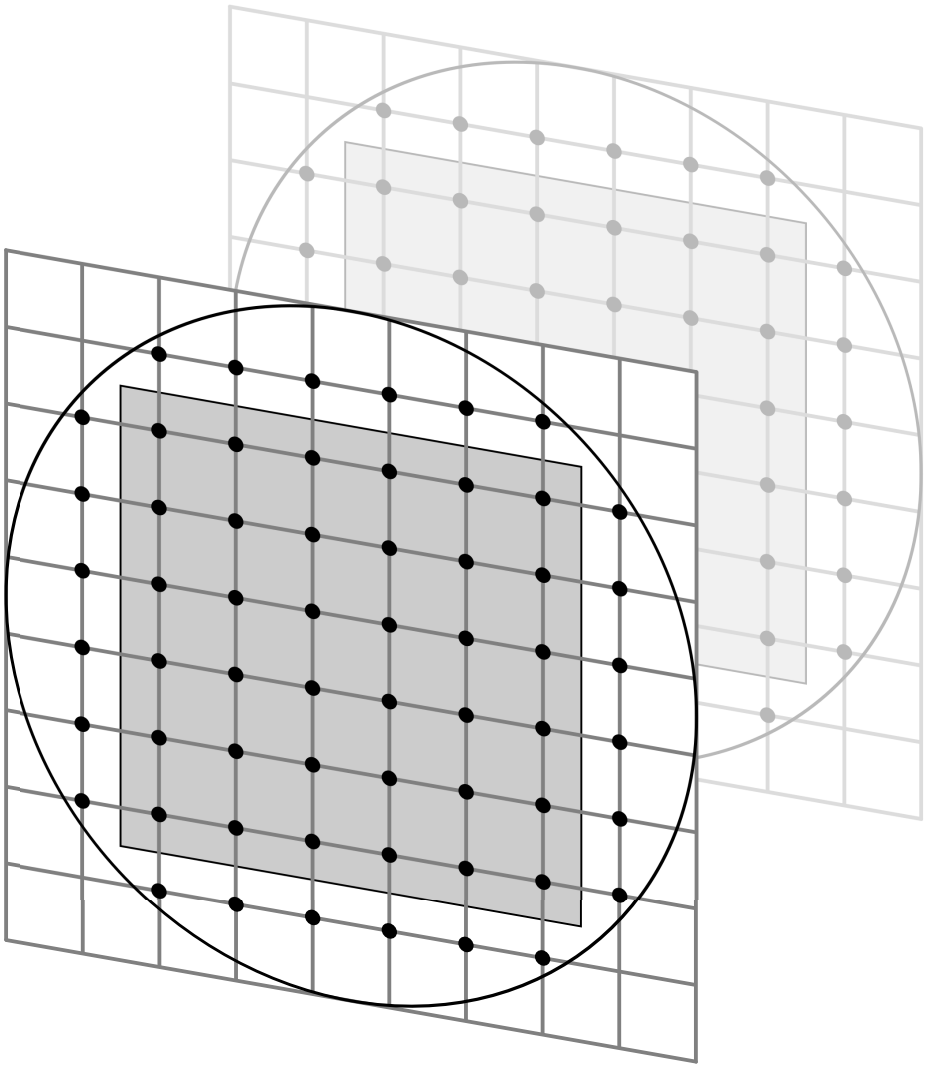


Figure 7.2: A distribution of equivalent source points Π_i^ℓ due to a circular extension of the parallel faces (in gray) of the cube c_i

chosen so as to minimize, in the mean-square norm, the array of differences $(\psi^{c_i,\mathrm{eq}}(x) - \psi^{c_i,\mathrm{true}}(x))$ for x varying over a number n^{coll} of adequately selected collocation points on S_i. Hence, for each ℓ, the intensities in (7.10) are obtained in practice as the least-squares solution of an overdetermined linear system $A\xi = b$, where A is an $n^{\mathrm{coll}} \times M^{\mathrm{eq}}$ matrix. As in [16], the work presented here exploits certain symmetries in order to reduce by a factor of eight the computational cost associated with the solution of these least-square problems.

Because for a given ℓ, the circular regions Π_i^ℓ are not pairwise disjoint, it is necessary to combine equivalent source intensities for all sources supported at a given point x' that corresponds to two different cells, say, c_r and c_s for which $x' = x_{r,p}^\ell = x_{s,q}^\ell$ for some integers

p and q. Thus, the total field is given by

$$\psi^{(*)\ell} = \sum_{x' \in \Pi^\ell} \left(\xi_{x'}^{(m)\ell} G_{2\pi k/c}(x - x') + \xi_{x'}^{(d)\ell} \partial G_{2\pi k/c}(x - x')/\partial x_\ell \right) \tag{7.11}$$

where $\xi_{x'}^{(m)\ell}$ and $\xi_{x'}^{(d)\ell}$ denote the sum of all intensities of equivalent sources located at a point $x' \in \Pi^\ell$:

$$\xi_{x'}^{(m)\ell} = \sum_{x_{i,j}^\ell = x'} \xi_{i,j}^{(m)\ell}, \quad \xi_{x'}^{(d)\ell} = \sum_{x_{i,j}^\ell = x'} \xi_{i,j}^{(d)\ell}. \tag{7.12}$$

Clearly, the quantity $\psi^{(*)\ell}$ in equation (7.11) is in a form suitable for evaluation, by means of 3D FFTs, at all points on the grid Π^ℓ not coinciding with the equivalent source points.

Remark 7.2.1. The original presentation of this acceleration method describes a certain "correction step for local fields", in which the contribution from equivalent sources that lie close to evaluation points are subtracted and correctly reintroduced. This costly procedure accounts for as much as 61% of the computing time (of the equivalent source algorithm) in the examples given in [17]. In the context of the present work, however, there is always a nonvanishing separation between the surfaces $\mathcal{S}$, over which sources are integrated, and $\mathcal{B}$, where the resulting fields are evaluated. Thus this "correction" step may be obviated by ensuring that the positioning of these two surfaces also implies that any cells c_i, c_j enclosing points on $\mathcal{S}, \mathcal{B}$, respectively, are strictly nonadjacent. A sufficient condition is that $R_{\min} > 2\sqrt{3}H$, and a necessary condition is that $R_{\min} > H$. Intermediate distances (between these two inequalities) may also be suitable, but the nonadjacency requirement must be checked directly. The exclusion of this step not only simplifies the approach, but also doubles the effective speed of the resulting boundary condition method.

7.2.2 Interior Dirichlet solutions

To obtain approximations of the Kirchhoff integrals $\psi^{(\text{true})}(\mathbf{x})$ (that is, the fields generated at $\mathbf{x}$ by the true discrete surface sources contained on Γ) at surface points $x \in \mathcal{B} \cap c_i$, the algorithm employs solutions to the Helmholtz equation within c_i, with Dirichlet boundary conditions given by $\psi^{(\text{eq})\ell}$, $l = 1, 2, 3$. These Dirichlet problems can be solved uniquely (in view of the assumption that $2\pi k/c$ is not a resonant frequency), and thus the good approximation properties of the Kirchhoff integral on the boundary of each cell c_i translate

into good approximations for the same integral on the surface $\mathcal{B}$. Following [16], the algorithm presented here produces the needed solutions of the Dirichlet problems by means of approximations of the following either the form

$$P(\mathbf{x}) = \sum_{j=1}^{n^w} \gamma_j e^{2\pi i \frac{k}{c} \mathbf{u}_j \cdot \mathbf{x}} \tag{7.13}$$

or, in the event that the quantity $2\pi k/c$ is near (or equal to) zero [1]

$$P(\mathbf{x}) = \sum_{j=1}^{n^w} \gamma_j G_{2\pi k/c}(\mathbf{v}_j - \mathbf{x}), \tag{7.14}$$

each one of which is valid within cells c_i nonadjacent to Γ. Here, the $\mathbf{u}_j$ are unit vectors that adequately sample the surface of the unit sphere, and $\mathbf{v}_j$ are similarly sampled from the enclosing cube of side $3H$, S_i. The coefficients γ_j are obtained in such a way that the relation $P(\mathbf{x}) = \psi^{(\text{true})}$ is satisfied, in the least-squares sense, for all $\mathbf{x}$ in an adequately chosen collocation mesh on the surface of the cube c_i.

The representations in equations (7.13) and (7.14) are particularly convenient given the need to evaluate the boundary operator $\mathcal{L}_\alpha$, applied to the Kirchhoff integral. Time differentiation in the frequency domain simplifies to scalar multiplication, and the representative basis functions may be differentiated exactly to produce the required normal derivative on $\mathcal{B}$, in a fashion similar to that used for evaluation of gradients in [13, 14]. In other words, equation (7.13) and (7.14) give

$$\mathcal{L}_\alpha P(\mathbf{x}) = \sum_{j=1}^{n^w} \gamma_j \left(\alpha + i\frac{\omega}{c} + 2\pi i \frac{k}{c} \mathbf{d}(\mathbf{x}) \cdot \mathbf{u}_j \right) e^{i \frac{k}{c} \mathbf{u}_j \cdot \mathbf{x}} \tag{7.15}$$

and

$$\mathcal{L}_\alpha P(\mathbf{x}) = \sum_{j=1}^{n^w} \gamma_j \left[\left(\alpha + i\frac{\omega}{c} \right) G_{2\pi k/c}(\mathbf{v}_j - \mathbf{x}) + \mathbf{d}(\mathbf{x}) \cdot \nabla_{\mathbf{v}_j} G_{2\pi k/c}(\mathbf{v}_j - \mathbf{x}), \right], \tag{7.16}$$

respectively.

[1] This second form, generalized slightly from the usual expansion for $2\pi k/c = 0$, is necessary in view of the increasing near-linear-dependence of the plane wave basis functions as $|2\pi k/c|$ vanishes.

7.2.3 Single precision least squares

The linear systems for ξ and γ, used to compute expansions for the outgoing and incoming solutions may be solved efficiently by means of precomputed matrix factorizations, owing to the fact that the relative collocation and equivalent source positions (or direction vectors, in the case of the plane wave expansion) remain fixed relative to the position of each cubic cell c_i. In [16] a QR factorization was employed for this purpose, requiring one unitary matrix-vector product and one triangular back-substitution per cubic cell per ℓ. When restricted to single precision arithmetic, however, the conditioning of the matrices prevents this approach from attaining the desired level of accuracy. Therefore, in this work, a truncated SVD is precomputed in double precision, then projected down to single precision for use during GPU execution. The SVD approach requires two unitary and one diagonal matrix-vector products, which is asymptotically as much as 1.5 times as much computational effort as the QR-based approach, assuming all singular values are taken into account. Due to the fast convergence of the combined single- and double-layer representation, however, the truncated SVD results in a reduced-rank representation with very similar, and in some cases superior, operation counts as compared to the previous approach. Combined with the higher efficiency of matrix-vector products (relative to back-substitution), the SVD approach is almost always as fast, if not faster, than the original QR approach.

This behavior is quantified in the case of the most costly equivalent source configuration used in the examples presented in Chapter 9. For the wave-number $\nu = 0$ the cube is taken to have a side of $H = 0.08$, with circular faces of radius $1.6\frac{H}{2}$. Equivalent sources are distributed with a density of 8 points per H, and collocation points are distributed at 12 points per $3H$, resulting in a system of 864 equations for 496 unknowns. A point source of unit intensity is placed at the most challenging location (as observed by [16]), situated on an edge of the cube, halfway between the two circular faces of equivalent sources at position $(H/2, H/2, 0)$, here with the faces perpendicular to the z-axis. Table 7.1 summarizes the differences in performance between the QR and truncated SVD approaches to solving this linear system, assuming the corresponding matrix factorizations have been precomputed.

In each test, the relative singular value tolerance ϵ is chosen to be as large as possible without impairing the accuracy of the resulting system. The computing time remains the same in the double-precision case (where most of the singular values are retained), and

Precision	Method	Time (seconds)	Relative error	$\|\xi\|_\infty$
Single	QR	7.3×10^{-4}	4.5×10^{-2}	1.4×10^{5}
	SVD, $\epsilon = 10^{-6}$	3.7×10^{-4}	6.5×10^{-5}	3.0×10^{0}
Double	QR	3.2×10^{-3}	2.7×10^{-6}	1.4×10^{5}
	SVD, $\epsilon = 10^{-10}$	3.2×10^{-3}	5.3×10^{-7}	1.0×10^{2}

Table 7.1: Comparison of the performance of the QR and truncated SVD approaches for evaluation of an equivalent source representation in a single cube, averaged over 1000 runs. The error is evaluated by means of a finely discretized enclosing cube of side $3.5H$. In the single precision case, 218 singular values (out of 496) are used, while in the double precision case, 422 singular values are used.

improves by nearly a factor of two in the single-precision case (when retaining only half of the singular values). In both cases the resulting accuracies of the representations are improved via the truncated SVD. Furthermore, the solution norm ($\|\cdot\|_\infty$) is significantly lower with this approach, mitigating the risk of subtractive cancellation occurring during the global convolution operation.

7.3 Implementation of the equivalent-source algorithm in CUDA-capable devices

The discussion presented above reduces the problem of evaluation of the computational boundary conditions introduced in Chapter 6 to two main algorithms, namely, 1) Transformation of the time-domain problem into the frequency domain (Section 7.1), and 2) Fast evaluation of the resulting frequency-domain integrals by means an equivalent-source algorithm (Section 7.2). The first of these two algorithms does not require extensive computing times, and is therefore implemented at low cost on the host CPU. The second of these algorithms requires more intensive computations, and is therefore implemented on the GPU.

The equivalent source-algorithm consists of the following sequence of operations:

1. For each c_i intersecting Γ, evaluate the true field $\psi^{c_i,\text{true}}$ at the collocation points over S_i.

2. For each l, c_i intersecting Γ, use the previously computed fields to solve the linear least-squares problem for the unknown equivalent sources $\xi_{i,j}^{(m)\ell}$ and $\xi_{i,j}^{(d)\ell}$.

3. For each ℓ, sum the contributions at overlapping points Π_i^ℓ and convolve against the monopole and dipole kernels in order to evaluate $\psi^{(*)\ell}$.

4. For each c_i intersecting $\mathcal{B}$, using the convolved fields for $\ell = 1, 2, 3$, solve the linear least-squares problem for the plane wave (or monopole) expansion.

5. For each boundary point on $\mathcal{B}$, evaluate the field arising from the plane wave (or monopole) expansion over the enclosing cell c_i computed in step 4.

Steps 1 and 5 represent the least amount of computational effort, and while they require (concise) custom written CUDA code, they are evaluated very efficiently with minimal effort. Steps 2–4, on the other hand, represent the bulk of the computing time for the algorithm, but rely entirely on standard computational primitives. Step 3 involves computing a sequence of large 3D FFTs, a task for which the CUDA SDK's built-in CUFFT library is well-suited.[2] Steps 2 and 4, finally, may be optimized by solving the linear equations for many c_i (or all, memory permitting) simultaneously. The computation is thus performed as a small number of large, dense matrix-matrix products, for which a standard, well-optimized library function from the CUDA SDK can be leveraged. By not requiring that individual cubes be computed in sequence, the CUBLAS `cgemm` implementation is afforded the greatest possible flexibility in the distribution of the corresponding computation over the available hardware.

The performance of the resulting GPU implementation, in comparison to the CPU-based approach, is demonstrated in Table 9.2 in Section 9.3, in the context of a full application of the boundary condition method, along with the interior time domain solver described in Chapter 5.

[2] As opposed to the implementation of the segmented FC method presented in Chapter 4, where its use would have required several additional read-write cycles per time-step.

Chapter 8

Hybrid FC/DG solver

This chapter presents a hybrid approach which combines the interior FC solver described in Chapter 5 and the MIDG discontinuous Galerkin finite element code [39] (DG-FEM) with the FC-ES computational boundary conditions presented in Chapter 7. Hybridization with DG-FEM provides greater flexibility in the treatment of the scattering surface Γ by FC solvers, as it enables consideration of surfaces represented by well-tested and mature mesh-generation codes such as GMSH [30] and it provides an interface with legacy geometries, while still taking advantage of the accuracy and efficiency of the FC-based interior solver.

(Unlike traditional finite element methods, discontinuous Galerkin methods do not impose any a priori conditions on the continuity of the discrete basis functions between neighboring elements. This results in a larger number of degrees of freedom, and hence a larger number of unknowns. However, due to the greater locality of the computation, it facilitates the construction of highly parallel, explicit solvers. Interactions between elements are quantified by a numerical flux, in a manner similar to that used by finite volume methods.)

8.1 DG-FEM interface

The DG-FEM solver used in these examples is based on the MIDG (mini discontinuous Galerkin) package [40]. Only small changes have been made in order to facilitate interfacing the code to the FC solver presented herein, while preserving a strict separation between the internals of each code.

In the hybrid algorithm the domain is decomposed using an extension of the approach presented in Chapter 5. A region immediately around the scattering surface(s), up to a rectangular parallelepiped interface (or disjoint union of such), is discretized with a finite

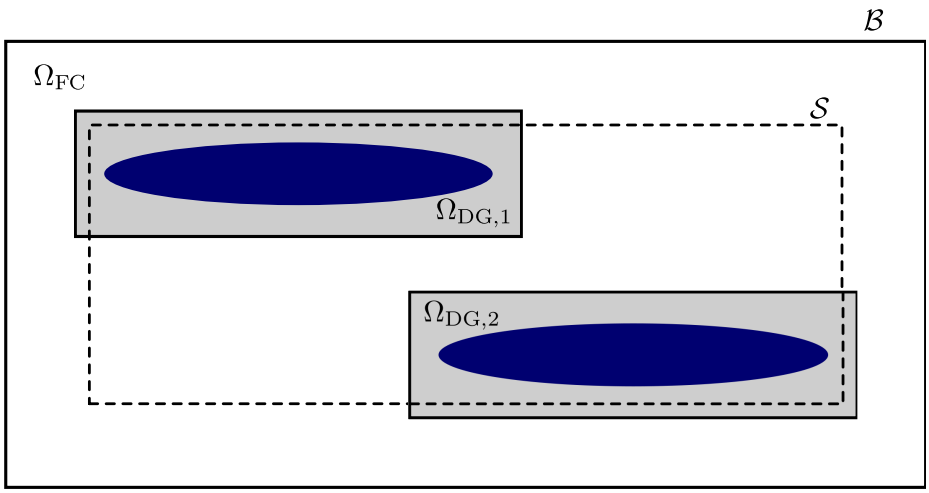


Figure 8.1: A hybrid model with two DG-FEM subdomains, $\Omega_{\mathrm{DG},1}$ and $\Omega_{\mathrm{DG},2}$, marked in gray, enclosing two scattering surfaces, marked in dark blue. In addition, the FC domain is shown along with the Kirchhoff surface $\mathcal{S}$ and enclosing computational boundary $\mathcal{B}$ of the overall computation domain Ω.

element mesh for the DG-FEM solver. The FC mesh, in turn, is taken as a portion of a Cartesian mesh in a region surrounding the DG domain(s): the FC mesh equals that part of the Cartesian mesh contained in the complement of the DG parallelepiped(s). (The assumption that the interface between the FC and DG regions is a rectangular parallelepiped is only introduced for simplicity: neither the FC or DG components of the hybrid require it.) Figure 8.1 displays a generic arrangement: DG meshes are used in the domains $\Omega_{\mathrm{DG},1}$ and $\Omega_{\mathrm{DG},2}$, and a Cartesian mesh is used in $\Omega_{\mathrm{FC}} = \Omega \setminus (\Omega_{\mathrm{DG},2} \cup \Omega_{\mathrm{DG},2})$.

8.1.1 Data specification for DG

The nature of the DG-FEM formulation requires the evaluation of a numerical flux over the faces of each element. For the faces that comprise the exterior boundary of the DG domain, an *incoming* flux must somehow be computed from the data on the FC side. The precise form of this flux is defined by the quantity

$$\mathbf{R}(u, \mathbf{v}) = u - \mathbf{n} \cdot \mathbf{v}$$

evaluated on either side of the interface. This term is included in a weak formulation of the wave equation, integrated against the DG test functions ϕ.

The unknowns $u, \mathbf{v}$ are interpolated from the FC grid to quadrature points on the

interface boundary $\partial\Omega^{DG}$ prescribed by the DG algorithm. The capability to provide such interpolations is available as part of the multi-patch FC method. Clearly this approach for FC-to-DG communication maintains a strict "black-box" separation between the inner workings of the two algorithms.

8.1.2 Data specification for FC

The DG algorithm, in turn, provides data to the FC algorithm in the form of the patch-interpolation strategy outlined in Section 5.3. With reference to the notations introduced in that section, each grid line in the FC domain that intersects the FC-DG interface is extended so that it penetrates into the finite element mesh by a number r of FC fringe points. The solution values at these fringe points are interpolated from the DG-FEM domain, using the DG representation, at every time-step of the FC method (see Section 8.1.3). The only requirement imposed on the DG-FEM implementation in these regards is the ability to evaluate the solution at points interior to its own computational domain.

8.1.3 Temporal subcycling

One additional difficulty arises in the form of the CFL condition for the DG-FEM solver. Owing to the necessary refinement of DG elements near the scattering surface as well as the high order of the DG polynomials basis used in the present context (fourth order in the examples considered in this thesis), the time-step required for DG stability is very small. The time-steps restrictions associated with the FC algorithm are much more lax, and it is therefore desirable to select the FC and DG time-steps independently.

Clearly, use of two different time-steps requires interpolation in time. This is most simply accomplished by using polynomial interpolation over a number of previous time-steps, in such a way that the interpolation accuracy order matches the order of accuracy of the time integration scheme. Furthermore, by enforcing that the larger (FC) time-step be an integer multiple of the smaller (DG) time-step, time-interpolation needs only occur in one direction, namely, when communicating data from the FC domain to the DG domain.

Chapter 9

Numerical results

This chapter presents a variety of numerical examples resulting from the application of the FC, FC-ES, and FC-DG hybridized algorithms to a sequence of acoustic problems featuring increasingly complex geometries. Both the high-order convergence and the computational efficiency of the resulting methodologies are demonstrated in both CPU and GPU architectures. All CPU tests presented in this chapter were executed on an Intel Nehalem E5520 with a clock speed of 2.27 GHz; the corresponding GPU results were obtained from runs on an NVIDIA Tesla C-1060 GPU at 1.30 GHz.

9.1 FC PDE solver: comparison with the FDTD scheme

In order to place the FC numerical solvers in the context of the existing literature, this section presents a comparison of the FC and FDTD solvers [74]. The FDTD methods are robust, well tested, and remain widely used. Furthermore, the availability of mature and highly optimized implementations (such as CPU implementation Meep [62] of a Yee scheme for Maxwell's equations, and the corresponding GPU proof-of-concept implementation [55]) provides an excellent benchmark for the computational efficiency of the algorithms presented herein.

The analysis presented in this section suggests that use of the FC method should be generally quite advantageous. Indeed, some of the estimates in this section indicate that tenfold improvements in computing times and three-hundred-fold improvement in memory requirements result from use of FC methods over the corresponding requirements of the FDTD algorithm for problems as small as 16 wavelengths in diameter. Very significant additional improvements result for problems of larger acoustical size: the results of this

section, for example, indicate that use of a FDTD algorithm for the acoustic hyperbolic system (5.3) in a three-dimensional geometry 512 wavelengths in diameter (as required for simulation of scattering by a F-22 Raptor aircraft [76] at the *lowest* frequency of X-band, used by the radar onboard the F-22) requires 2 *exabytes* of memory, more than one thousand times the memory available on the largest distributed computing clusters currently operating, such as the Sequoia BlueGene/Q and K Supercomputer [54]; a computation based on the FC solver, in turn, would require 69 TB—that is, 30,000 times less memory than that required by FDTD, and 20 times less than that available in aforementioned computers. Those estimates indicate, further, an improvement of the FC method over the FDTD method by a factor of 4,000 in computing time.

(As shown in Section 3.2.3, the FC methods have also demonstrated significant performance improvements over previous high-order methods. In reference [1], further, a wide range of hybrids of high-volume and discontinuous Galerkin algorithms [25] of various orders of accuracy are considered; there it is shown that the FC methodology gives rise to improvements in computing times by factors of the order of 200 over the best of the high-order algorithms considered in the latter reference. Additional improvements in computing times, ranging from factors of 200 to 3.3 million, have been reported in various publications [1, 2, 18, 53].)

In order to demonstrate the relative efficiency of the FC solver in a highly challenging regime, in this section comparisons of sampling efficiency are made in one-dimensional problems, followed by extrapolation to three dimensions on the basis of actual timings of the respective solvers on three-dimensional volumetric meshes. Both, the Yee and FC schemes are specialized here to the case of a polarized plane wave traveling in vacuum, in which case the Maxwell's equations simplify to

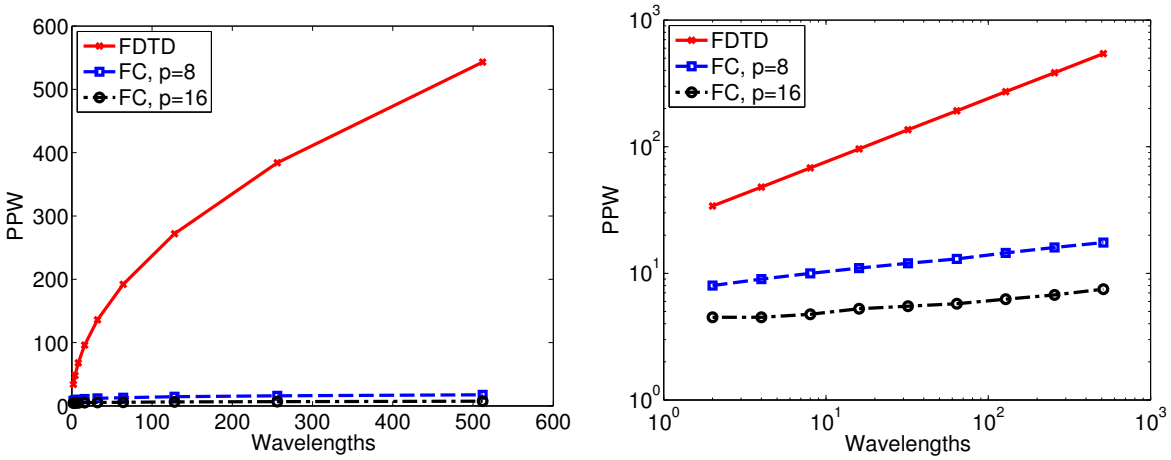
$$\begin{aligned} \frac{\partial E^y}{\partial t} &= -\frac{1}{\epsilon_0 \mu_0} \frac{\partial H^z}{\partial x} \\ \frac{\partial H^z}{\partial t} &= -\frac{\partial E^y}{\partial x}, \end{aligned} \tag{9.1}$$

a system which, taking normalized units so that $\epsilon_0 \mu_0 = 1$, is equivalent to the hyperbolic form of the wave equation (3.4), with $u = E^y$, $v = H^z$, and $c = 1$. For the Yee scheme the fields u and v are sampled on uniform, staggered meshes over the periodic interval $x \in [0, 1]$, $x_j = \frac{j-1}{N-1}$. The discrete fields $u_1 \dots u_N$ and $v_{1+1/2} \dots v_{N+1/2}$ are then evolved in

time according to the second-order, centered-time, centered-space rule

$$\begin{aligned} u_j^{n+1} &= u_j^{n-1} + \frac{\Delta t}{\Delta x}\left(v_{j+1/2}^n - v_{j-1/2}^n\right) \\ v_j^{n+1} &= v_j^{n-1} + \frac{\Delta t}{\Delta x}\left(u_{j+1}^n - u_j^n\right). \end{aligned} \tag{9.2}$$

This system exactly captures the behavior of the fully three-dimensional Yee scheme when applied to an axis-aligned, polarized plane wave, and it therefore provides a lower bound on the discretization required to achieve a given accuracy in the three-dimensional cases.



(a) Regular (b) Log-scale

Figure 9.1: Required sampling, in points per wavelength (PPW), for the FC and FDTD methods to produce, with an error of less than 1%, a traveling wave solution for a given number of wavelengths. Curves for two segmented FC solvers are shown—one using the filter parameter $p = 8$, consistent with the that used in the three-dimensional solvers presented in this chapter, and another with a milder, $p = 16$ filter, suitable for cases with larger segment sizes.

The one-dimensional Yee and FC solvers are compared by determining, numerically, the sampling in points per wavelength (PPW) required to achieve an error of less than 1% when evolving the exact solution

$$u(x,t) = v(x,t) = \sin\left(4\pi\left(x - t\right)\right) \tag{9.3}$$

a number of W wavelengths, with W ranging from $W = 2$ to $W = 512$. Figure 9.1 shows the corresponding discretization requirements, as a function of the acoustical size, at both

normal and logarithmic scales. The number of points per segment n_s in the FC solver is allowed to vary slightly in order to more readily accommodate the domain, but is always bounded by $n_s \leq 40$. For domains with 16 wavelengths to a side, corresponding to a similar scale as in the fully three-dimensional experiments described in the remainder of this chapter, FC has a superior sampling efficiency by a factor of $96/11 \approx 8.73$. In order to achieve even 1% error in three dimensions, then, the FDTD method for the wave equation requires an amount of memory of no less than

$$(4 \text{ unknowns}) \times (3 \text{ grids}^1) \times (16 \text{ waves})^3 \times \left(96 \frac{\text{points}}{\text{wave}}\right)^3 \times \left(8 \frac{\text{bytes}}{\text{point}}\right) = 348 \text{ GB}, \quad (9.4)$$

and is therefore an intractable problem for a single modern workstation. For the FC method, by comparison, only

$$(4 \text{ unknowns}) \times (6 \text{ grids}^2) \times (16 \text{ waves})^3 \times \left(11 \frac{\text{points}}{\text{wave}}\right)^3 \times \left(8 \frac{\text{bytes}}{\text{point}}\right) \approx 1 \text{ GB} \quad (9.5)$$

is required, cutting the memory requirements by a factor of 348 and easily fitting within the available resources on a typical desktop PC. An acoustic problem in a cube measuring 512λ to a side would require an astronomical $2 \cdot 10^9$ GB, or roughly 2 EB. The FC solver would only require 71 TB, an improvement by a factor of over $3 \cdot 10^4$, bringing that same problem within the reach of modern supercomputers.

In order to compare the computational speed of the respective solvers, times for both Meep and FC are measured, in a single core, when evaluating a single time-step over a mesh of 256^3 points. The FDTD is significantly faster per unknown, requiring only 0.67 seconds to evolve 6 equations, compared to 27 seconds for the FC method to evolve 4 equations. Accounting also for the differences in the three-dimensional CFL conditions as well as the required numbers of points per wavelength, the relative speedup achieved with FC is a factor of

$$\left(\frac{\sqrt{3}}{16}\right) \times \left(\frac{4}{6}\right) \times \left(\frac{96}{11}\right)^4 \times \left(\frac{0.67}{27}\right) \approx 10.4, \quad (9.6)$$

for a three-dimensional domain 16λ across, and it grows to a factor of over $4 \cdot 10^3$ for

[2]The Yee solver requires storage for three complete meshes, corresponding to the solution at the next, current, and previous time-steps.

[2]The FC solver requires storage for the solution at the current and next time-steps, as well as values of the time derivative over the last four time-steps.

the larger 512λ case. The result of the differences in memory usage and computational performance is that, using the FC method instead of FDTD, a single workstation may solve problems of a size previously only practically approached with the aid of supercomputers, and supercomputers may be used to solve problems that were entirely out of reach for the FDTD algorithm.

9.2 Simple examples in three-dimensional space

This section presents a number of simple examples, including three-dimensional accuracy tests in problems for which exact solutions exist, as well as an illustration of the overlapping patch methodology described in Chapter 5; demonstrations of the overall methodologies introduced in this thesis for significantly more complex geometries and including use of convergent computational boundary conditions are presented in Sections 9.3 through 9.5.

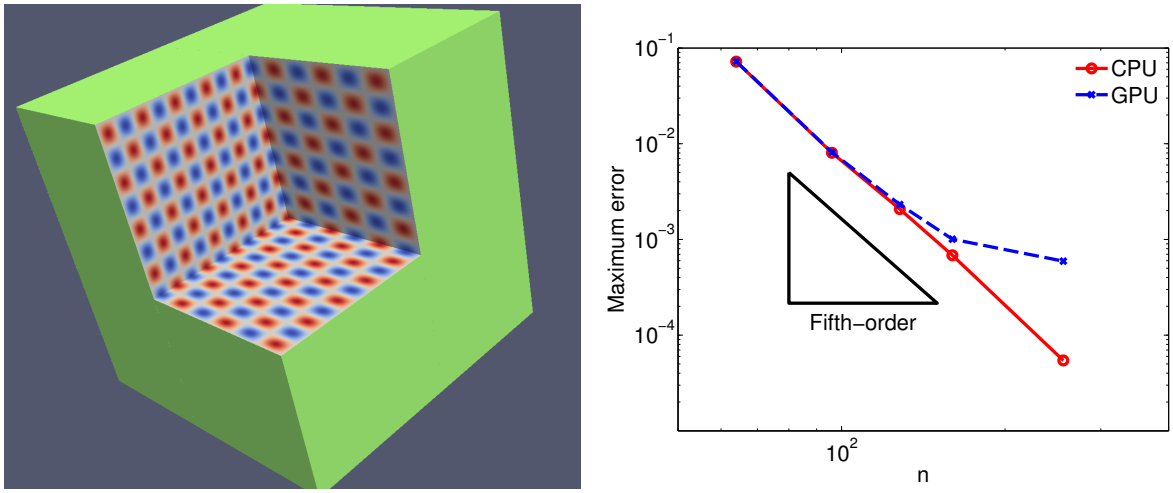
9.2.1 Normal modes in a cube

While solution of PDE problems in a simple cubic geometry does not require use of most of the methodologies introduced in Chapter 5, this geometric configuration does offer a set of exact solutions which are both simple and well known—normal modes, also known as standing waves—which provide an excellent first set of tests for the fully three-dimensional PDE solvers under consideration. Thus equation (5.3) is solved over the domain $\Omega = [0,1]^3$, coupled with zero Dirichlet boundary conditions on u, and no boundary conditions on v.

This configuration admits the familiar solutions

$$\begin{aligned} u_{\ell,m,n}(x,t) &= \sin(\ell\pi x_1)\sin(m\pi x_2)\sin(n\pi x_3)\cos(\sqrt{\ell^2+m^2+n^2}\,t) \\ v_{\ell,m,n}(x,t) &= -\begin{pmatrix} \ell\pi\cos(\ell\pi x_1)\sin(m\pi x_2)\sin(n\pi x_3) \\ m\pi\sin(\ell\pi x_1)\cos(m\pi x_2)\sin(n\pi x_3) \\ n\pi\sin(\ell\pi x_1)\sin(m\pi x_2)\cos(n\pi x_3) \end{pmatrix} \frac{\sin(\sqrt{\ell^2+m^2+n^2}\,t)}{\sqrt{\ell^2+m^2+n^2}} \end{aligned}$$

for integers $\ell, m, n > 0$. Figure 9.2(a) shows a sample solution at time $t = 0$ for parameters $(\ell, m, n) = (10, 14, 18)$. Figure 9.2(b) demonstrates the convergence of the solution at a final time $T = 1$ for a sequence of spatial discretizations. At the finest discretization considered, $N = 256^3$ spatial mesh points, each time-step requires 47.4 seconds when executed on a



(a) Cutaway of interior solution

(b) Error in solution at time $T = 1$

Figure 9.2: The FC method applied to the normal mode solution of index $(\ell, m, n) = (10, 14, 18)$, solved to time $T = 1$. Both CPU and GPU implementations give rise to fifth-order convergence at coarser discretizations and higher error levels, but the single precision arithmetic inherent in the GPU implementation pollutes the GPU solution at lower error levels.

single CPU, and reduces to only 1.66 seconds on the GPU, an improvement by a factor of 28.6.

9.2.2 Sphere in a cube

In order to evaluate the performance of the FC solver in a simple setting which includes overlapping patches and inter-patch interpolation, a problem concerning a sound-soft sphere of radius $r = 0.25$ contained within the cube $[-1, 1]^3$ is considered; clearly this problem amounts to a three-dimensional version of the one introduced in Section 5.5. The decomposition is entirely analogous to that presented in the earlier section: it includes a large Cartesian patch cut by a spherical hole concentric with and of a slightly larger radius than the sound-soft sphere (cf. Figure 5.1(b) for the two-dimensional rendition), together with a sequence of *six* spherical-coordinate patches each one of which amounts to a square in angular parameter space (the corresponding two-dimensional situation, including four polar-coordinate patches, is depicted in Figure 5.1(a)). Finally, the mesh size in each patch is chosen according to similar criteria as those used in Section 5.5, with the added constraint that the mesh sizes in each one of the two angular directions in the square parameter-space discretizations coincide. This selection does decrease slightly the minimal grid spacing, and

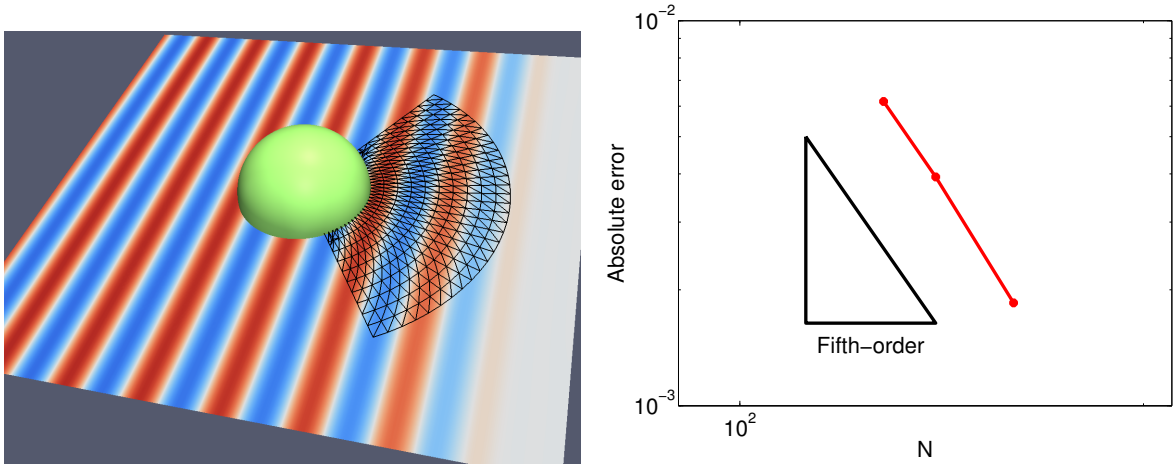
hence the corresponding time-step required by the CFL condition.[3]

Vanishing initial conditions are used in this example, along with boundary conditions (on both the internal sphere and external cube boundaries) corresponding to a right-moving plane wave that smoothly transitions from an amplitude of zero (initially) to one (asymptotically, as $t \to \infty$), as detailed in what follows. Letting

$$f_k(s) = \begin{cases} e^{-1/(ks)^2} \sin ks & \text{if } s > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9.7}$$

the Dirichlet boundary condition is set to

$$u(x,t)|_{x \in \partial\Omega} = -f_k(x_1 - t - 1). \tag{9.8}$$



(a) Artificial plane wave solution

(b) Convergence

Figure 9.3: Right-moving artificial plane wave solution evaluated by the FC method from correspondingly artificial Dirichlet boundary conditions on Γ and $\mathcal{B}$. A wire-frame grid (coarsened, for visibility) is overlaid on the first spherical-coordinate patch to illustrate the overlapping patch decomposition.

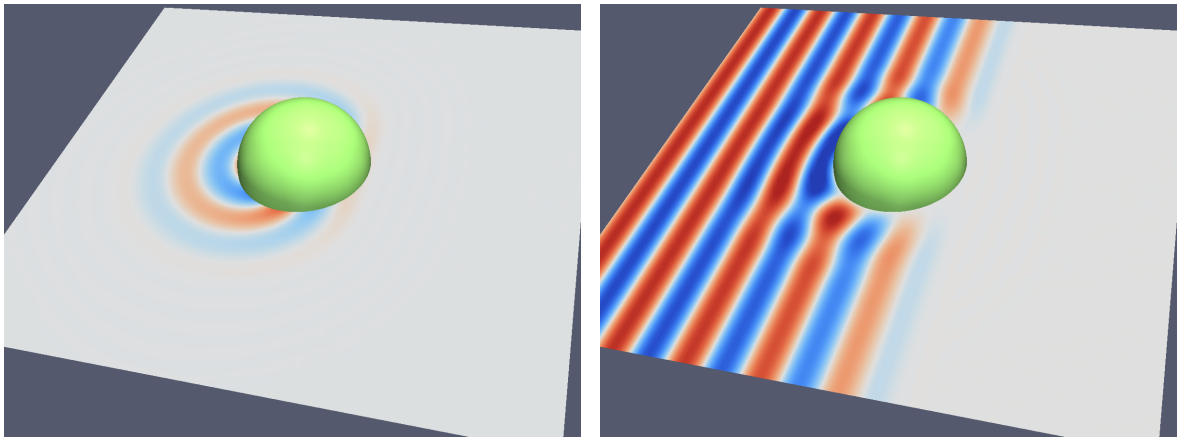
This simple right-moving wave is identically zero throughout the domain at $t = 0$, and, up to the numerical accuracy of the solver, will move through the domain undisturbed by the domain boundaries. In Figure 9.3(a), the $x_3 = 0$ slice of the solution at $T = 2$ for $k = 24$ is shown. Again, the convergence of the algorithm as the discretization is increased

[3]Since the conforming spherical-coordinate patches comprise the majority of computational unknowns, the overall impact of the more constrained CFL condition on the *global* time-step is minimal.

is shown in Figure 9.3(b).

A final example in this section also concerns the domain just considered and related boundary conditions. (This example further demonstrates how use of the particular choice made for the function f_k can be applied to solve PDE problems involving arbitrary physically motivated driving sources.) Prescribing Dirichlet boundary conditions as above in this section on the spherical boundary and zero on the outer (cube) boundary, the resulting solution equals the scattered field resulting from an incident field of the form $-f_k(x,t)$ on a sound-soft sphere up to the point that the scattered wave reaches the cube boundary. Figure 9.4(a) demonstrates this radiating solution at time $T = 1.$, for $k = 24$; the associated accuracy should be described closely by the right graph in Figure 9.4(b).

This solution continues to represent the true wave scattered by the sphere up to time $T = 1.5$, at which point the scattered field impinges upon the boundary of the cube, and the Dirichlet zero boundary condition applied there no longer coincides with a physically correct *radiating* boundary solution. In order to evolve this system further in time, then, either the domain may be enlarged (at considerable computational expense), or a numerical radiating boundary condition, such as, e.g., that presented in Chapter 7 and demonstrated in Section 9.3, must be applied.



(a) Scattered field u_s (b) Total field $u = u_s + u_i$

Figure 9.4: The scattered field produced by the FC solver for physically correct Dirichlet boundary conditions corresponding to an incident plane wave on Γ and zero on $\mathcal{B}$, at time $T = 1$. Soon after this point in time, the scattered wave impinges on the computational boundary, and radiating boundary conditions must be used to evolve the system further.

9.3 Computational boundary conditions for the sphere-in-cube problem

Using the FC-ES boundary conditions described in Chapter 7 with Sommerfeld-like boundary operator (6.11), in this section the solution for the sphere-in-cube problem presented in Section 9.2.2 is continued beyond the time for which the scattered wave reaches the artificial boundary of the computational domain.

The parameterization of the domain remains the same as in the previous section, and, a cubic Kirchhoff surface $\mathcal{S}$ enclosing the sphere is now used as needed by the FC-ES boundary conditions described in Section 7.2. This cubic surface is selected in such a way that the faces are evenly sampled by the discretization points of the enclosing patch, and just large enough that it does not intersect the cut hole, as depicted in Figure 9.5.

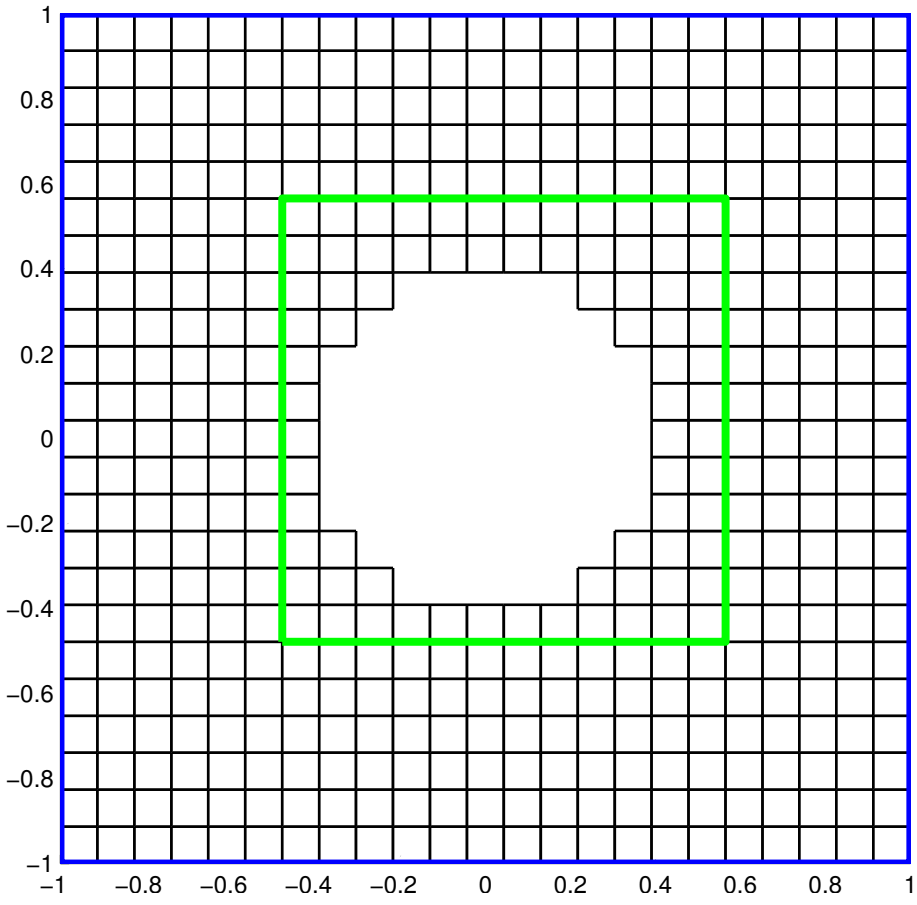


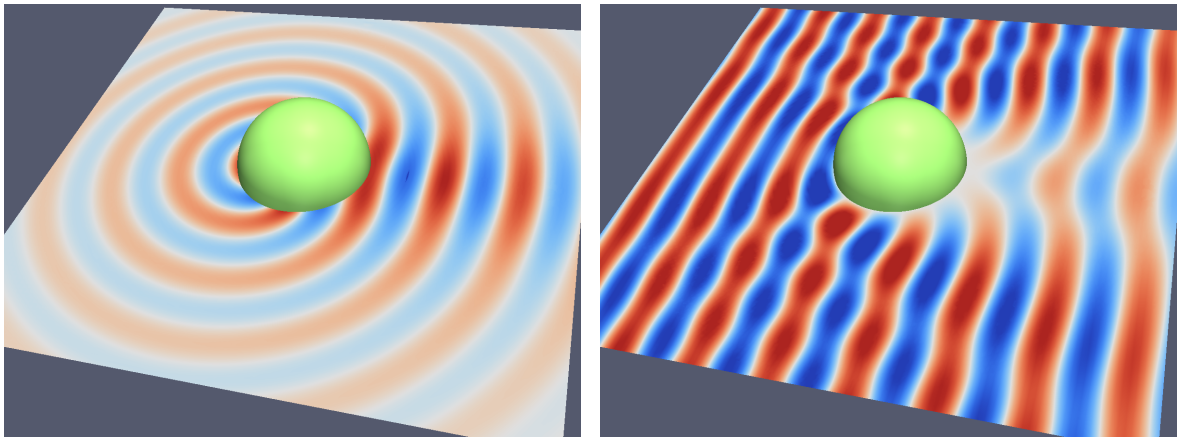
Figure 9.5: Two-dimensional cross section of the Kirchhoff surface $\mathcal{S}$ as embedded in the enclosing Cartesian mesh, using a fairly coarse discretization for ease of visualization. (Figure 5.1 displays a section of the complete patch decomposition.)

Since the faces of this cube are discretized by regular Cartesian grids, a closed composite Newton-Cotes rule of accuracy order 7 based on this grid, with corresponding integration weights w_j, is used. (Note that the error resulting from the corresponding Kirchhoff integration for observation points on the computational boundary is of an accuracy order higher than that inherent in the boundary portion of the FC method.) If necessary, the Kirchhoff surface is enlarged slightly, until the number n of discretization points in each direction is equal to $n(m-1)+1$, with $m = 7$ for this choice of integrating polynomial degree, so that the composite integration rule correctly divides into a number of equally sized intervals.

Figure 9.6 presents the numerical solution at $T = 2.5$ (cf. Figure 9.4 in Section 9.2.2). The application of the FC-ES boundary conditions introduces insignificant additional error, and it allows for the solution to be evolved to an arbitrary point in time. The needed FC(Gram) expansion in time of the scattered field on the Kirchhoff surface $\mathcal{S}$ (see Section 7.1) for this example was taken over a time-period $T_0 = \frac{R_{\max}}{c} = 2.79$ and updated every $\frac{R_{\min}}{c} = 0.389$ time units. For this test, the first $M = 42$ positive and negative terms in the FC(Gram) expansion[4] were used; they were integrated rapidly on the Kirchhoff surface for all points on the computational boundary using the equivalent-source representation described in Section 7.2. Note that use of time frequencies up to $M = 42$ implies sampling of discrete wave-numbers up to $k_{\max} = 92.33$—clearly above the $k = 26$ inherent in the present problem. For efficiency, cube sizes ranging over $H \in [0.05, 0.1]$ were used for the various wave-numbers involved, with a sufficient sampling of equivalent sources, collocation points, and plane wave (or monopole) sources such that the resulting FC-ES method has an error less than $1 \cdot 10^{-4}$, smaller than the anticipated error in the time-domain FC solver.

A high-level breakdown of the computational time required for this problem is provided in Table 9.1, showing the comparatively small computing times required by the FC-ES radiating boundary condition algorithm. Table 9.2 summarizes the time required by CPU and GPU implementations of these boundary integral evaluations, at each step of this method (as outlined in Section 7.3), when applied to the most computationally intensive frequency term used in this experiment. This additionally demonstrates the suitability of this method for such many-core architectures, as the GPU implementation of the FC-ES boundary condition improves over the corresponding CPU implementation by a factor of

[4]The negative frequency terms are not explicitly integrated. Since the function u is real-valued, the negative frequency integrals are simply the complex conjugates of the positive frequency integrals.



(a) Scattered field u_s (b) Total field $u = u_s + u_i$

Figure 9.6: Radiating solution from a sound-soft sphere at $T = 2.5$, produced by the time-domain FC method and FC-ES boundary conditions. The resulting shadow is clearly visible in the total field.

Section	Seconds per FC-ES update	Percentage overall
FC interior solver	66780	97.75
FC(Gram) on $\mathcal{S}$	52	0.08
FC-ES integration	1417	2.07
IFFT on $\mathcal{B}$	67	0.10
Total	68316	—

Table 9.1: Single-core CPU times required for the sphere-in-box geometry using $N = 8224768$ discretization points. One complete FC-ES update cycle spans 3180 time-steps of the interior FC solver.

51.28.

Use of the Sommerfeld-like boundary operator $\mathcal{L}_\alpha$ with parameters $\alpha(x)$, $\mathbf{d}(x)$ as recommended by [42], and covered briefly in Section 6.1, results in a worsened CFL condition for the three-dimensional FC method described here, reducing the Courant number from $C = 1/16$ to $C = 1/24$. This arises due to the discontinuity in the outward-facing normal vector at edges and corners of the cubic boundary $\mathcal{B}$. In fact, any outward-facing unit vector may be used, and thus here it is convenient to prescribe

$$\mathbf{d}(x) = \frac{x}{\|x\|_2}. \tag{9.9}$$

Section	Time (CPU)	Time (GPU)	Speedup
1) Evaluate collocation fields $\psi^{c_i,\text{true}}$	6.82	0.02	341
2) Compute equivalent sources $\xi_{i,j}^{(m),l}$, $\xi_{i,j}^{(d)l}$	0.59	0.05	11.8
3) Global convolution for $\psi^{(*)l}$	61.90	1.30	47.6
4) Compute plane wave sources $\gamma_{i,j}$	0.28	0.03	9.33
5) Evaluate boundary operator $\hat{\mathcal{L}_\alpha} P(x)$	3.23	0.02	161.5
Total time for frequency-domain integral	72.82	1.42	51.28

Table 9.2: Time in seconds required to compute each portion of the frequency-domain integral, as outlined in Section 7.3. In this example there are $N_{\mathcal{S}} = 57624$ integration points and $N_{\mathcal{B}} = 153600$ boundary points, solved for the lowest nonzero frequency mode. The unusually high gains in steps 1) and 5) are due to significantly the higher throughput of trigonometric functions on the GPU: the Intel CPU considered has a peak per-core throughput of 1/117 trigonometric evaluations per cycle [43], whereas the NVIDIA GPU supports a variable per-core rate of as much as 1 trigonometric evaluation per cycle [59] in the ideal (small angle) case.

With this slight modification to the parameters $\alpha(x)$ and $\mathbf{d}(x)$ of $\mathcal{L}_\alpha$, the resulting numerical scheme has Courant number 1/17, nearly as good as that required when prescribing only Dirichlet conditions.

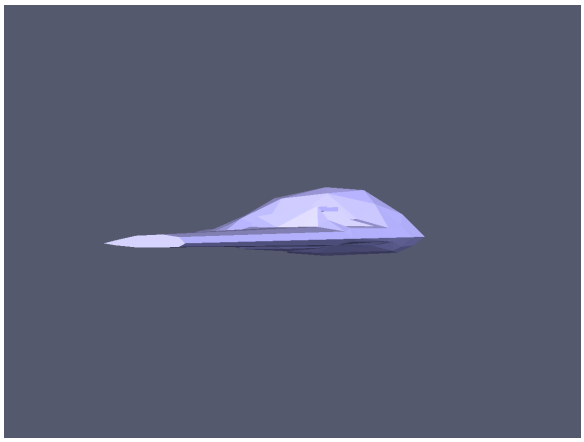
9.4 Stealth aircraft

In order to demonstrate the efficacy and simplicity of the hybrid FC-DG approach, the scattered field from a sound-hard stealth aircraft is computed, as arising from an incident ramped plane wave prescribed by the boundary condition on $\mathbf{v}$

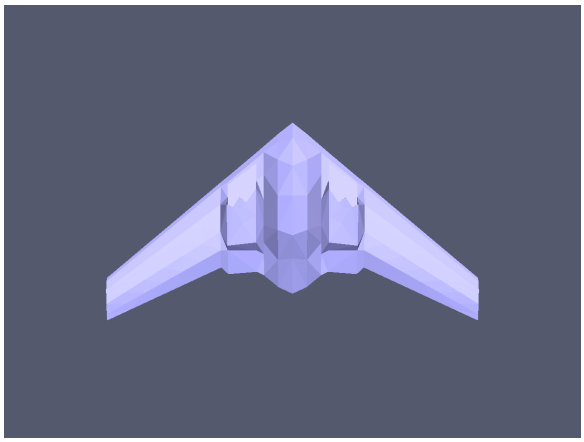
$$\mathbf{v}(x,t)|_{\partial\Omega} = \frac{1}{2}\begin{pmatrix}1\\0\\0\end{pmatrix}(1+\tanh(30t-3))\cos(k(t-\mathbf{x}_1)) \tag{9.10}$$

for wave-number $k = 26\pi$. Several closeups of the aircraft geometry are shown in Figure 9.7, detailing the piecewise-linear, nonconvex surface Γ. A narrow region of dimensions $0.76 \times 0.2 \times 0.5$ tightly surrounding this surface is discretized with a finite element mesh for the DG-FEM solver, which is in turn embedded in a larger cubic FC mesh of side 1.2. The FC-ES boundary conditions are parametrized as in the example in Section 9.3, now integrating

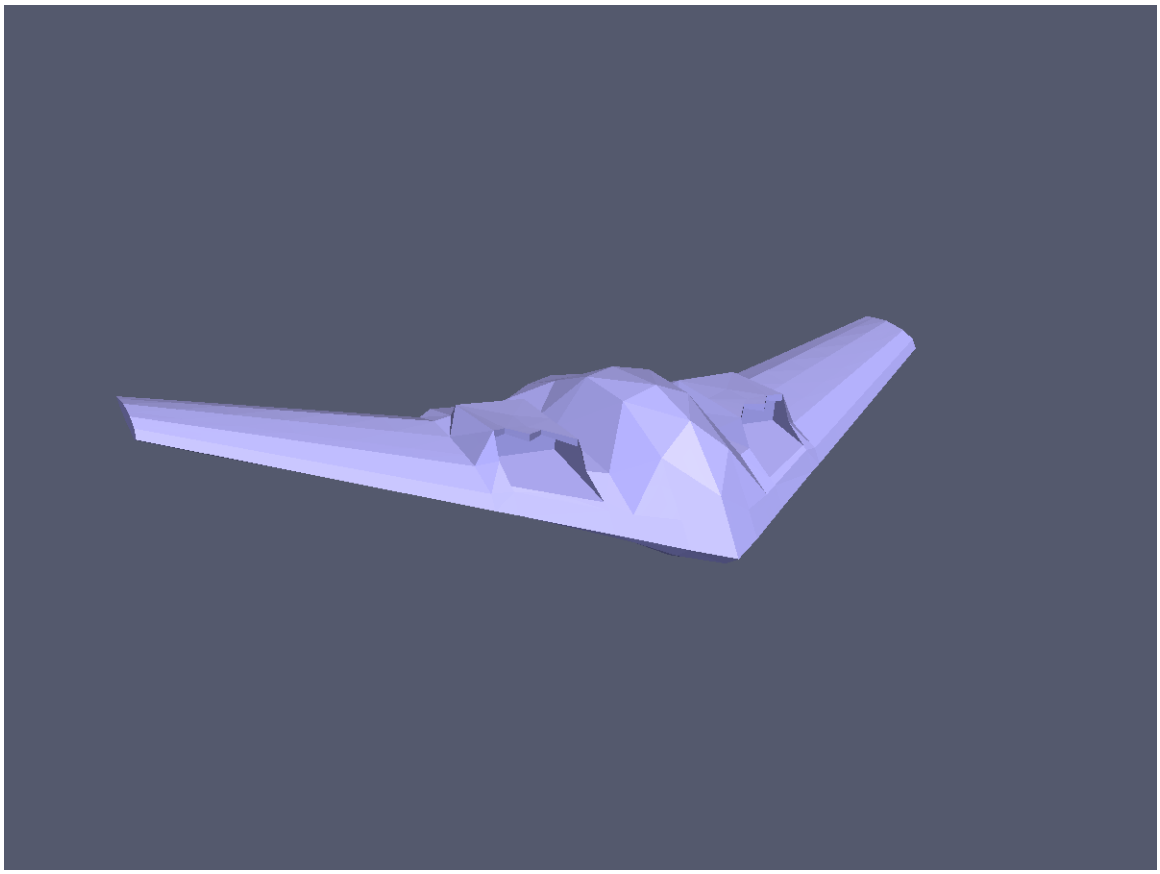
only $M = 32$ discrete frequency terms (in Section 9.3, $M = 42$ terms were included over a correspondingly larger domain). For an example of the required computing times, see Table 9.3 in the following section.



(a) Side view



(b) Top view



(c) Perspective view

Figure 9.7: Stealth aircraft geometry. Despite the low resolution used to describe the geometry, a very finely sampled finite element mesh must be used in the volume immediately surrounding the surface—which accounting for approximately 4% of the volume of the total computational domain.

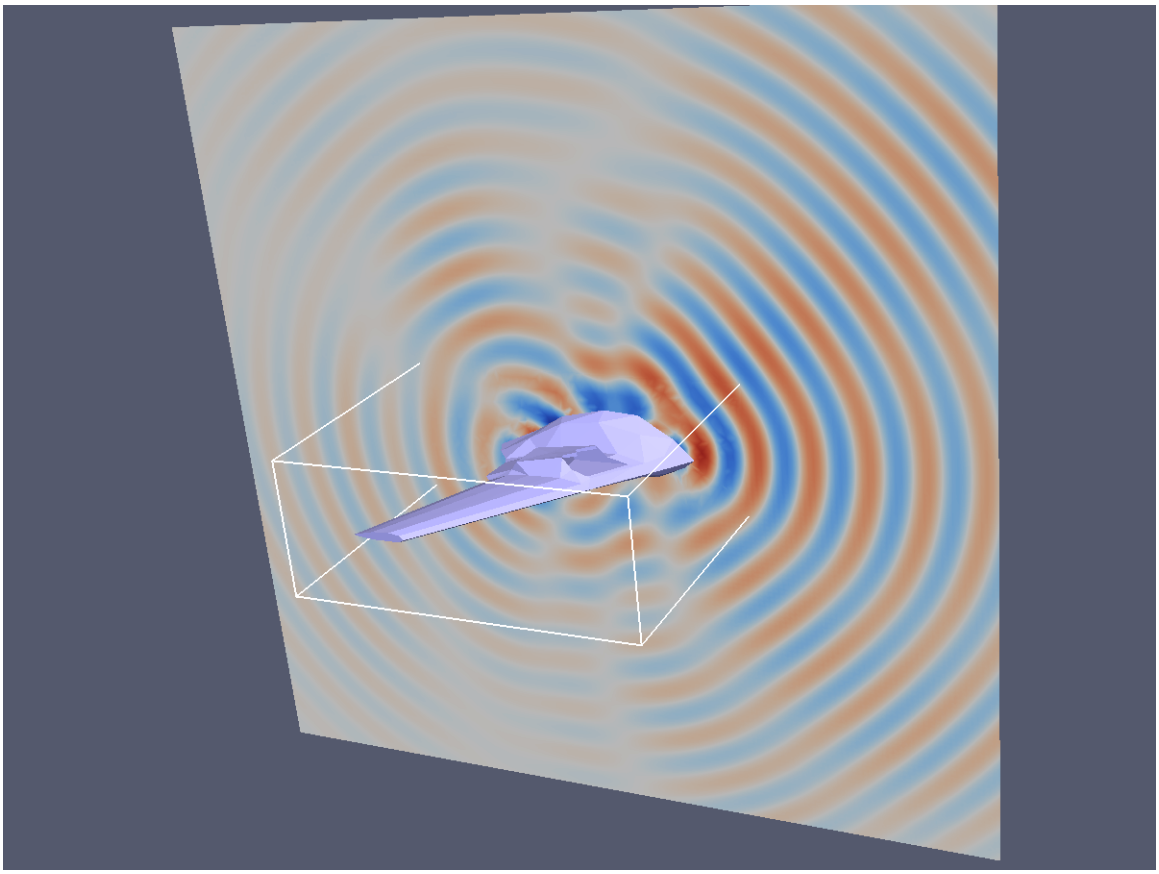


Figure 9.8: Scattered field resulting from right-moving plane wave, as computed by the FC/DG hybrid solver with FC-ES computational boundary conditions. The wire-frame outline shows the extent of the DG-FEM subdomain.

9.5 Multiple aircraft

The experiment of Section 9.4 is now expanded to include several aircraft separated by a large distance—more specifically two aircraft in close formation, and another two in similar formation a large distance away. Only the immediate volume around each disjoint group of aircraft needs to be evolved in the time domain, demonstrating the flexibility of this approach in handling nonconvex (in this example, not even simply connected) domains. Figure 9.9 shows the relative orientation of the aircraft, the FC subdomains, and the *undiscretized* intervening space.

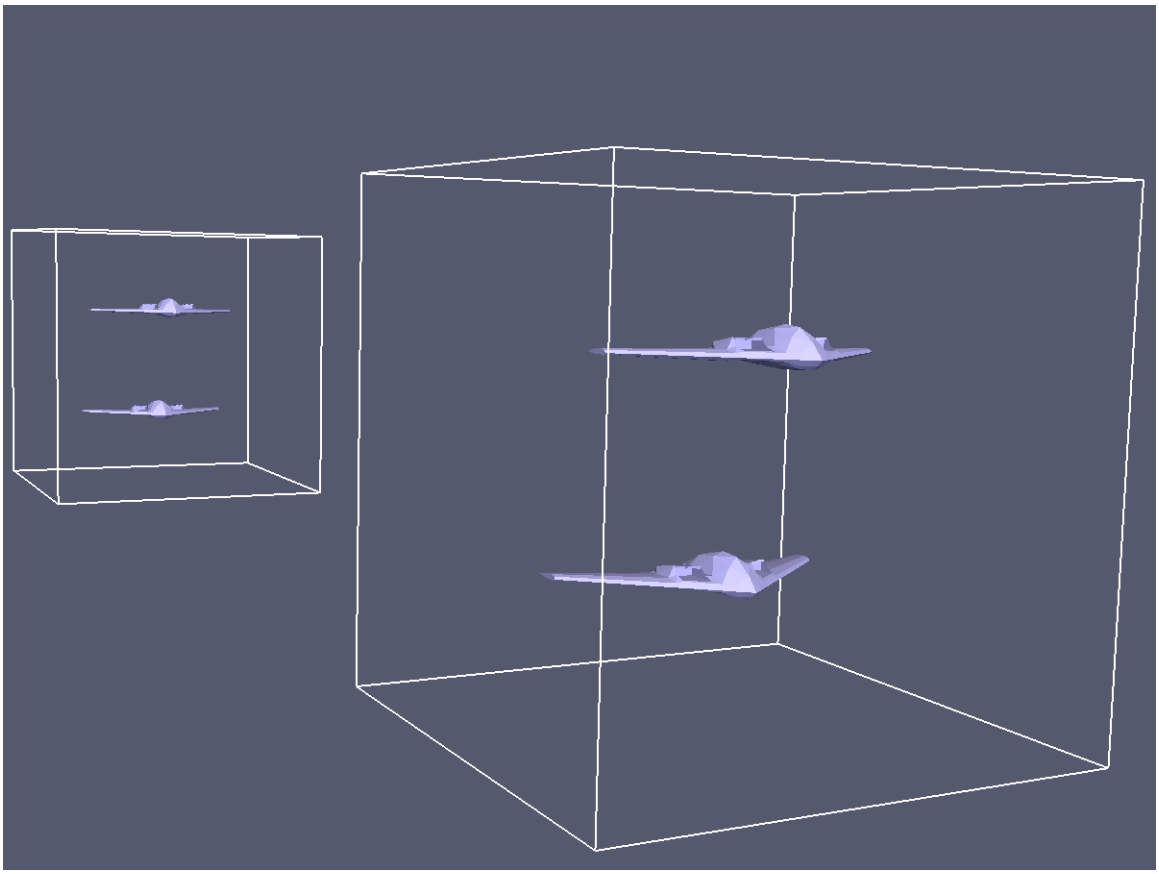
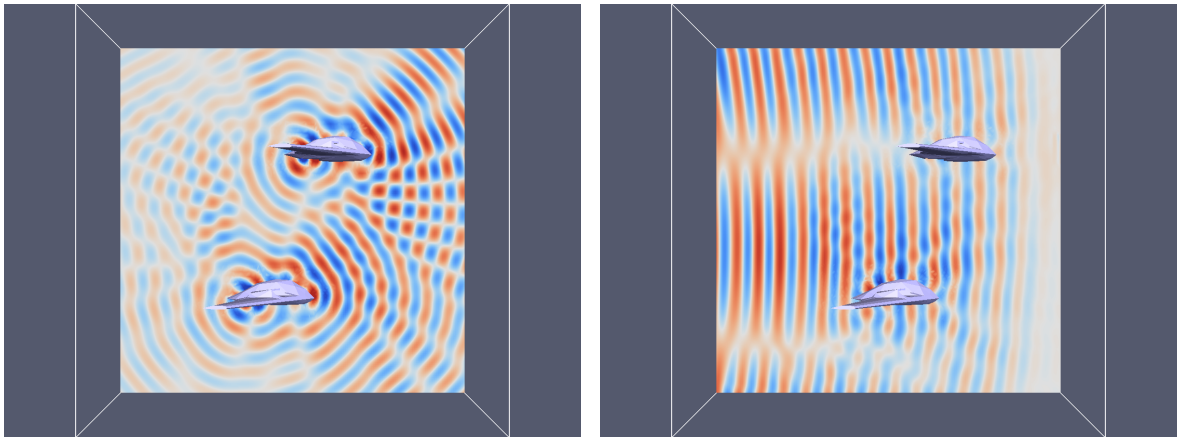


Figure 9.9: Four aircraft grouped in two disjoint domains, enclosing two aircraft each. The white outlines show the extent of the FC discretization. The volume between the two FC domains is treated, at negligible computational cost, by the FC-ES boundary condition algorithm.

In fact, it is possible to compute the boundary conditions in such a way that the computational cost (per time-step) remains completely independent of the separation between the two groups of scatterers illustrated in Figure 9.9. For each pairwise combination of local

domains Ω_j, Ω_k, the integral over $\mathcal{S}_j$ to be used as boundary conditions on $\mathcal{B}_k$ may each be computed separately—the integral is summed over the union of the Kirchhoff surfaces, but the Fourier series expansion on each may be evaluated independently, alleviating the need to expand over a large interval in time. This does result in a cost that is quadratic in the number of disjoint domains[5], but the sparse convolution grids used to accelerate the integration, as described in Chapter 7, only need to be large enough to enclose the larger of $\mathcal{S}_j$ and $\mathcal{B}_k$, rather than both simultaneously. This is accomplished by taking the spatial separation into account directly in the convolution kernel, obviating the need to discretize the entirety of the convex hull of the two (distant) boundaries.

Such a configuration is shown in Figures 9.9 and 9.10, with the corresponding radiating solution at time $T = 6$ arising from an incident field only on the leftmost two aircraft—this scenario is chosen so that the multiple scattering interactions with the nonilluminated, secondary aircraft can easily be appreciated. It is readily seen that the upper-right aircraft is in the shadow of the illuminated aircraft to the left. In addition, this shadow does not appear on the leading-edge of the scattered field, and only establishes itself a few wavelengths behind the moving front—a "precursor" behavior that is difficult to capture with purely frequency-domain methods.



(a) Left-side aircraft, illuminated

(b) Right-side aircraft, secondary scatterers

Figure 9.10: Closeup view of the scattered field at time $T = 6$ over each one of the two disjoint domains depicted in Figure 9.9. Each cubic domain measures approximately 15.6 wavelengths to a side, and the two domains are separated by a gap of 49.4 wavelengths.

Table 9.3 gives a breakdown of the required computing time for each component in the

[5] An alternative which may be used to alleviate this concern is outlined in Section A.2.

FC-DG hybrid solver. The FC, DG, and FC-ES algorithms are executed entirely on the GPU, while the relatively inexpensive FC(Gram) and FFTs on $\mathcal{S}$, $\mathcal{B}$ are executed on the CPU in order to save the more-limited GPU memory.

Section	Seconds	Percentage overall
FC solver	19674	53.0
DG-FEM solver	10776	29.0
Communication FC to DG	2726	7.3
Communication DG to FC	221	0.6
FC(Gram) on $\mathcal{S}$	566	1.5
FC-ES integration	2096	5.6
IFFT on $\mathcal{B}$	1046	2.8
Total	37106	—

Table 9.3: Computing times required by each portion of the FC-DG hybrid solver to solve the four-aircraft problem up to $T = 6$ using the Tesla C-1060 GPU. Note that the relative cost of the boundary conditions is increased relative to that shown in Table 9.1, due to the quadratic cost in the disjoint-domain methodology and the wider (asymptotically a factor of two) time intervals required on $\mathcal{S}_j$ to maintain the consistency of Kirchhoff's integral formula.

Appendix A

Future work

A.1 Electromagnetics: Maxwell's equations

Many wave-scattering problems of engineering importance involve electromagnetic rather than acoustic waves, and hence require the solution of Maxwell's equations. Since the methods developed in this work are intentionally built around the general framework of first-order hyperbolic systems, the FC solver is naturally extensible to problems of electromagnetic scattering. Furthermore, the FC-ES radiating boundary conditions can be applied to both nondissipative and dissipative (after extending the equivalent and plane wave source representations) materials. Of particular research interest is the determination of the way in which the FC methodology can be best applied to multiple material interfaces and variable coefficient media.

A.2 Superscalar algorithm

It has been assumed until now in this work that the ratio $R_{\min}/R_{\max}$ is bounded away from zero. This assumption requires that the size of the gap between Kirchhoff and boundary surfaces must remain bounded from below, relative to the size of the computational domain. It is possible to conceive of cases where this constraint is violated—extremely large concave scatterers, or even a spherical shell of increasing radius but fixed thickness. Any such case reveals an additional, subtle cost in the FC-ES method, growing linearly with $R_{\max}/R_{\min}$, the factor of redundancy with which the overlapping time-series data on $\mathcal{S}$ is integrated. Consider the case of a spherical shell of fixed thickness D but of increasing radius R, discretized volumetrically with $N = \mathcal{O}(R^2)$ unknowns. In this scenario, in fact, the ratio

$R_{\max}/R_{\min}$ grows linearly in R without bound. (Note that even a very mild growth in the thickness of the domain alleviates this difficulty.) As a consequence, the requisite FC-ES boundary condition evaluation time is proportional to $\mathcal{O}(R^{11/3})$, clearly super-linear even with respect to the volume of the convex hull of the domain. This super-linear growth is, however, very mild, and furthermore the performance comparison in Table 9.1 suggests a *very* favorable constant of proportionality. For sake of completeness, however, a remedy to this asymptotic cost is discussed in what follows.

A recursive procedure is described here that replaces the linear cost in $R_{\max}/R_{\min}$ with a logarithmic one. The time-series data on $\mathcal{S}$ can be decomposed, by the introduction of a partition of unity, into a recent, fast-updating component u_{fast} and an earlier, slow-updating component u_{slow}. The slow-updating component is extended by zero into the future by an application of FC(Gram) over a length of time equal to the support of the recent, fast-updating component. In doing so, the Kirchhoff integral with respect to u_{slow} remains a valid representation of a *portion* of the solution for an amount of time unconstrained by $R_{\min}$.

In order for the subdivision into $u_{\text{fast}} + u_{\text{slow}}$ to be of practical use, however, a method for evaluating u_{fast} more quickly is described, requiring approximately half the computing time required for the integral of u_{slow}. This is in fact be achieved by ensuring that the periodic continuation of the fast solution u_{fast} remains identically zero outside the support of its respective partition of unity for a period of time no less than $6\sqrt{2}\frac{H_{\max}}{c}$, where $H_{\max}$ is the largest cube size used in the FC-ES method. In doing so, the function u_{fast} can then be periodically continued over an interval in time of *half* the duration used for u_{slow}, and correspondingly fewer frequency domain terms need be kept to accurately capture a given numerical solution. The corresponding integrals are computed over a period in time smaller than the *temporal* dependence of the Kirchhoff integral formula. This is made possible by the fact that the time interval for which u_{fast} remains zero is sufficiently long to result in a well-defined *spatial* support in the transformed frequency domain integral—the contribution from each cubic cell c_i's equivalent source expansion on an arbitrary cube surface corresponds exclusively to either a field arising from the true, half-period signal or one arising from the "false" periodically repeated data. This relation from temporal- to spatial-locality allows for the correct field values to be exclusively integrated (convolved) after careful adjustment of the corresponding support of the Green's function kernels has been made. Furthermore,

this reduction of the support in the kernel also results in a reduction in required FFT grid sizes, and thus corresponding computing times, by a factor of approximately 3.4 at the first subdivision, and asymptotically approaching 8 (relative to the original grid size).

This approach can be applied recursively until approaching the minimal period of $6\sqrt{2}\frac{H_{\max}}{c}$. At each level, the signal is decomposed into one that remains valid for full duration (for that level of recursion), and one that is valid for half as long, but may be evaluated at half the cost—hence the logarithmic cost in both time and memory. Unfortunately, this approach as described has a relatively large constant of proportionality (relative to the FC-ES method as described earlier in this work), only offering significant gains for very large, irregularly shaped domains. The additional cost arises from the logarithmic number of subdivided intervals, the one-time doubling of the original time-series period in order to accommodate a sufficiently long zero-region, and perhaps most severely the increased number of Fourier series terms required to capture the influence of the multiplicative partition of unity, a phenomenon observed in a related context in [42].

Bibliography

[1] Albin, N., and Bruno, O. A spectral FC solver for the compressible Navier-Stokes equations in general domains I: Explicit time-stepping. *Journal of Computational Physics* (2011).

[2] Albin, N., Bruno, O., Cheung, T., and Cleveland, R. Fourier continuation methods for high-fidelity simulation of nonlinear acoustic beams.

[3] Alpert, B., Greengard, L., and Hagstrom, T. Rapid evaluation of nonreflecting boundary kernels for time-domain wave propagation. *SIAM Journal on Numerical Analysis* (2000), 1138–1164.

[4] Alpert, B., Greengard, L., and Hagstrom, T. Nonreflecting boundary conditions for the time-dependent wave equation. *Journal of Computational Physics 180*, 1 (2002), 270–296.

[5] Amdahl, G. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18–20, 1967, Spring joint computer conference* (1967), ACM, pp. 483–485.

[6] Armstrong, J. *Making reliable distributed systems in the presence of sodware errors*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2003.

[7] Baker, B., and Copson, E. *The mathematical theory of Huygens' principle*. Oxford: Clarendon Press, 1950.

[8] Bayliss, A., and Turkel, E. Radiation boundary conditions for wave-like equations. *Communications on Pure and Applied Mathematics 33*, 6 (1980), 707–725.

[9] Berenger, J. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of computational physics 114*, 2 (1994), 185–200.

[10] Boyd, J. *Chebyshev and Fourier spectral methods.* Dover Publishers, 2001.

[11] Boyd, J. A comparison of numerical algorithms for Fourier extension of the first, second, and third kinds. *Journal of Computational Physics 178*, 1 (2002), 118–160.

[12] Brown, D., Henshaw, W., and Quinlan, D. Overture: An object-oriented framework for solving partial differential equations on overlapping grids. *Object Oriented Methods for Interoperable Scientific and Engineering Computing, SIAM* (1999), 245–255.

[13] Bruno, O., Elling, T., Paffenroth, R., and Turc, C. Electromagnetic integral equations requiring small numbers of Krylov-subspace iterations. *Journal of Computational Physics 228*, 17 (2009), 6169–6183.

[14] Bruno, O., Elling, T., and Turc, C. Regularized integral equations and fast high-order solvers for sound-hard acoustic scattering problems. *International Journal for Numerical Methods in Engineering* (2012), n/a–n/a.

[15] Bruno, O., Han, Y., and Pohlman, M. Accurate, high-order representation of complex three-dimensional surfaces via Fourier continuation analysis. *Journal of Computational Physics 227*, 2 (2007), 1094–1125.

[16] Bruno, O., and Kunyansky, L. A fast, high-order algorithm for the solution of surface scattering problems: basic implementation, tests, and applications. *Journal of Computational Physics 169*, 1 (2001), 80–110.

[17] Bruno, O., and Kunyansky, L. Surface scattering in three dimensions: an accelerated high–order solver. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences 457*, 2016 (2001), 2921–2934.

[18] Bruno, O., and Lyon, M. High-order unconditionally stable FC-AD solvers for general smooth domains I. Basic elements. *Journal of Computational Physics 229*, 6 (2010), 2009–2033.

[19] Bruno, O. P. Fast, high-order, high-frequency integral methods for computational acoustics and electromagnetics. In *Topics in computational wave propagation*, vol. 31 of *Lect. Notes Comput. Sci. Eng.* Springer, Berlin, 2003, pp. 43–82.

[20] Chen, Z., Taflove, A., and Backman, V. Photonic nanojet enhancement of backscattering of light by nanoparticles: a potential novel visible-light ultramicroscopy technique. *Optics express 12*, 7 (2004), 1214–1220.

[21] Chew, W., and Weedon, W. A 3d perfectly matched medium from modified Maxwell's equations with stretched coordinates. *Microwave and optical technology letters 7*, 13 (1994), 599–604.

[22] Collino, F., Monk, P., et al. The perfectly matched layer in curvilinear coordinates.

[23] Cooley, J., and Tukey, J. An algorithm for the machine calculation of complex Fourier series. *Math. Comput 19*, 90 (1965), 297–301.

[24] Dagum, L., and Menon, R. OpenMP: an industry standard API for shared-memory programming. *Computational Science & Engineering, IEEE 5*, 1 (1998), 46–55.

[25] Dumbser, M. Arbitrary high order P_N/P_M schemes on unstructured meshes for the compressible Navier–Stokes equations. *Computers & Fluids 39*, 1 (2010), 60–76.

[26] Engquist, B., and Majda, A. Absorbing boundary conditions for numerical simulation of waves. *Proceedings of the National Academy of Sciences 74*, 5 (1977), 1765.

[27] Engquist, B., and Majda, A. Radiation boundary conditions for acoustic and elastic wave calculations. *Communications on Pure and Applied Mathematics 32*, 3 (1979), 313–357.

[28] Evans, L. *Partial Differential Equations*, vol. 19 of *Graduate Studies in Mathematics*. American Mathematical Society, 2000.

[29] Garabedian, P. *Partial Differential Equations*. American Mathematical Society, 1998.

[30] Geuzaine, C., and Remacle, J. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering 79*, 11 (2009), 1309–1331.

[31] Givoli, D. Numerical methods for problems in infinite domains.

[32] Givoli, D., and Cohen, D. Nonreflecting boundary conditions based on Kirchhoff-type formulae. *Journal of Computational Physics 117*, 1 (1995), 102–113.

[33] Golub, G., and Van Loan, C. *Matrix Computations*, third ed. The Johns Hopkins University Press, 1996.

[34] Griesemer, R., Pike, R., Thompson, K., et al. The Go programming language. `http://golang.org`, 2012 (accessed June, 2012).

[35] Grote, M., and Keller, J. Nonreflecting boundary conditions for Maxwell's equations. *Journal of Computational Physics 139*, 2 (1998), 327–342.

[36] Hagstrom, T., and Warburton, T. Complete radiation boundary conditions: minimizing the long time error growth of local methods. *SIAM Journal on Numerical Analysis 47*, 5 (2009), 3678–3704.

[37] He, S., and Weston, V. Wave-splitting and absorbing boundary condition for Maxwell's equations on a curved surface. *Mathematics and computers in simulation 50*, 5 (1999), 435–455.

[38] Hesthaven, J., Gottlieb, S., and Gottlieb, D. *Spectral methods for time-dependent problems*, vol. 21. Cambridge Univ Pr, 2007.

[39] Hesthaven, J., and Warburton, T. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*, vol. 54. Springer-Verlag New York Inc, 2008.

[40] Hesthaven, J., and Warburton, T. Nodal discontinuous Galerkin methods: algorithms, analysis, and applications. `http://nudg.org`, 2012 (accessed May, 2012).

[41] Higdon, R. Numerical absorbing boundary conditions for the wave equation. *Math. Comput 49*, 179 (1987), 65–90.

[42] Hoch, D. *Nonreflecting Boundary Conditions Obtained From Equivalent Sources For Time-Dependent Scattering Problems*. PhD thesis, California Institute of Technology, 2008.

[43] Intel Corporation. IA-64 and IA-32 architectures optimization reference manual. `http://www.intel.com/products/processor/manuals`, 2012, (accessed June, 2012).

[44] Intel Corporation. Intel microprocessor quick reference. `http://www.intel.com/pressroom/kits/quickreffam.htm#pentium`, 2012 (accessed June, 2012).

[45] Jiang, H., and Wong, Y. Absorbing boundary conditions for second-order hyperbolic equations. *Journal of Computational Physics 88*, 1 (1990), 205–231.

[46] Kanellos, M. Moore's Law to roll on for another decade. *CNET News.com* (2003).

[47] Kanellos, M. New life for Moore's Law. *CNET News.com* (2005).

[48] Lele, S. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics 103*, 1 (1992), 16–42.

[49] LeVeque, R. *Finite volume methods for hyperbolic problems*, vol. 31. Cambridge Univ Pr, 2002.

[50] LeVeque, R. *Finite difference methods for ordinary and partial differential equations*. Society for Industrial and Applied Mathematics Philadelphia, 2007.

[51] LeVeque, R., Berger, M., et al. Clawpack software version 4. `http://www.clawpack.org`, 2012 (accessed June, 2012).

[52] Lindman, E. "Free-space" boundary conditions for the time dependent wave equation. *Journal of computational physics 18*, 1 (1975), 66–78.

[53] Lyon, M., and Bruno, O. High-order unconditionally stable FC-AD solvers for general smooth domains II. Elliptic, parabolic and hyperbolic PDEs; theoretical considerations. *Journal of Computational Physics 229*, 9 (2010), 3358–3381.

[54] Meuer, H., Strohmaier, E., Dongarra, J., and Simon, H. Top 500 Supercomputers. `http://top500.org`, 2012 (accessed June, 2012).

[55] Micikevicius, P. 3d finite difference computation on GPUs using CUDA. In *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units* (2009), ACM, pp. 79–84.

[56] Moore, G., et al. Cramming more components onto integrated circuits. *Proceedings of the IEEE 86*, 1 (1998), 82–85.

[57] Mur, G. Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations. *Electromagnetic Compatibility, IEEE Transactions on*, 4 (1981), 377–382.

[58] Nordström, J., and Carpenter, M. High-order finite difference methods, multi-dimensional linear problems, and curvilinear coordinates. *Journal of Computational Physics 173*, 1 (2001), 149–174.

[59] NVIDIA Corporation. NVIDIA CUDA C programming guide. `http://www.nvidia.com/object/cuda_home_new.html`, 2012 (accessed June, 2012).

[60] Odersky, M., et al. An Overview of the Scala Programming Language. Tech. Rep. IC/2004/64, EPFL, Lausanne, Switzerland, 2004.

[61] Orszag, S., and Israeli, M. Numerical simulation of viscous incompressible flows. *Annual Review of Fluid Mechanics 6*, 1 (1974), 281–318.

[62] Oskooi, A. F., Roundy, D., Ibanescu, M., Bermel, P., Joannopoulos, J. D., and Johnson, S. G. MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method. *Computer Physics Communications 181* (January 2010), 687–702.

[63] Pelosi, G. The finite-element method, Part I: RL Courant [Historical Corner]. *Antennas and Propagation Magazine, IEEE 49*, 2 (2007), 180–182.

[64] Petropoulos, P. Reflectionless sponge layers as absorbing boundary conditions for the numerical solution of Maxwell's equations in rectangular, cylindrical, and spherical coordinates. *SIAM Journal on Applied Mathematics* (2000), 1037–1058.

[65] Ryaben'kii, V., Tsynkov, S., and Turchaninov, V. Global discrete artificial boundary conditions for time-dependent wave propagation. *Journal of Computational Physics 174*, 2 (2001), 712–758.

[66] Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., et al. Larrabee: a many-core x86 architecture for visual computing. *ACM Transactions on Graphics (TOG) 27*, 3 (2008), 18.

[67] Smith, W. A nonreflecting plane boundary for wave propagation problems. *Journal of Computational Physics 15*, 4 (1974), 492–503.

[68] Snir, M., Otto, S., Walker, D., Dongarra, J., and Huss-Lederman, S. *MPI: The complete reference*. MIT press, 1995.

[69] Sofronov, I. Artificial boundary conditions of absolute transparency for two-and three-dimensional external time-dependent scattering problems. *European Journal of Applied Mathematics 9*, 06 (1998), 561–588.

[70] Sommerfeld, A., Stern, M., Kuerti, G., Ramberg, E., Laporte, O., Moldauer, P., Bopp, F., Meixner, J., Kestin, J., and Straus, E. *Lectures on theoretical physics*, vol. 3. Academic Press New York, 1964.

[71] Stoer, J., and Bulirsch, R. *Introduction to Numerical Analysis*, vol. 12. Springer Verlag, 2002.

[72] Strang, G., and Fix, G. An analysis of the finite element method. *Prentice-Hall 124* (1973), 125.

[73] Strikwerda, J. *Finite difference schemes and partial differential equations*. Society for Industrial Mathematics, 2004.

[74] Taflove, A., and Hagness, S. Computational electrodynamics: The finite-difference time-domain method, 2005.

[75] Ting, L., and Miksis, M. Exact boundary conditions for scattering problems. *The Journal of the Acoustical Society of America 80* (1986), 1825–1827.

[76] USAF. Facsheets: F-22 Raptor. `http://www.af.mil/information/factsheets/factsheet.asp?id=199`, 2012 (accessed June, 2012).

[77] Weaver, W., and Gere, J. *Matrix analysis of framed structures*, vol. 3. Van Nostrand Reinhold New York, 1990.

[78] Yee, K. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *Antennas and Propagation, IEEE Transactions on 14*, 3 (1966), 302–307.