Multi-dimensional High-order Discretizations and Fast Solution Algorithms for Simplified Spherical Harmonic Approximations

by

Matt Kabelitz

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Nuclear Engineering and Radiological Sciences)
in the University of Michigan
2023

Doctoral Committee:

Assistant Professor Brendan Kochunas, Chair
Dr. Michael Hackemack
Dr. Gabriel Kooreman
Professor Shravan Veerapaneni
Professor Won Sik Yang

Matt Kabelitz
mkbz@umich.edu
ORCID iD: 0009-0003-3492-2118

Dedication

I dedicate this dissertation work to my loving wife, Kavita, for her constant support.

Acknowledgments

I also extend special thanks to the members of the NURAM research group, both past and current, who have all provided extraordinary and helpful insights on the development of this research.

This research was performed under appointment to the Rickover Fellowship Program in Nuclear Engineering sponsored by Naval Reactors Division of the U.S. Department of Energy.

Table of Contents

List of Figures

List of Tables

List of Appendices

List of Acronyms

GSP_N Generalized SP_N.

SP_N Simplified P_N.

AFEN Analytic Functional Expansion Nodal.

ANM Analytic Nodal Method.

CFEM Continuous Finite Element.

DG-FEM Discontinuous-Galerkin Finite Element Method.

FEM Finite Element Method.

FMM Fast Multipole Method.

GTIN Generalized Transverse Integration Nodal.

HODLR Hierarchical Off-Diagonal Low-Rank.

HPS Hierarchical Poincaré-Steklov.

HSS Hierarchical Semi-Separable.

IP-FEM Interior Penalty Finite Element Method.

LGL Legendre Gauss-Lobatto.

MMS Method of Manufactured Solutions.

MOC Method of Characteristics.

MOX Mixed-Oxide.

NEM Nodal Expansion Method.

PDE Partial Differential Equation.

SANM Semi-Analytic Nodal Method.

SVD Singular Value Decomposition.

TI Transverse Integration.

TPE Tensor Product Extension.

UNM Unified Nodal Method.

VNM Variational Nodal Method.

WRM Weighted Residuals Method.

Abstract

Efficient and flexible algorithms for solving low-order transport problems are highly desirable in a great number of applications. Such solvers are useful not only as rapid design and evaluation tools in their own right, but as acceleration techniques to be used in larger, more precise simulations. The focus of this thesis is a novel discretization and solver for the elliptic simplifications to the neutron transport equation; encompassing diffusion, Simplified P_N (SP_N) methods, and Generalized Simplified P_N which is the evolution of new theory pertaining to the SP_N boundary conditions. Demonstrations on a 1D test problem show that our new 4th-order Legendre-Gauss-Lobatto discretization has the same h-convergance (5th order) as traditional NEM, achieving the maximum convergence order imposed by the Bramble-Hilbert lemma. Unlike more traditional finite element methods, our discretization also preserves continuity at interfaces, and minimizes the number of basis functions used to represent these conditions. Our numerical results also show that this order of convergence extends to multidimensional problems without the need for transverse integration, and is capable of being applied to certain kinds of unstructured mesh.

The system of equations this discretization produces are also amenable to simplification and solution via hierarchical fast solvers such as the Hierarchical Poincaré-Steklov Method. This method contains explicit representations of certain conserved quantities, making it attractive for an extension of the 1D nodal method into multiple dimensions. This hierarchical method is implemented in a proof-of-concept multigroup code with the capability to handle deformed structured meshes, with no spatial homogenization. The solver utilizes no transverse leakage approximation in its multidimensional calculations, and is designed to operate on a sub-pin mesh. An eigenvalue solver based on power iteration is also implemented. Clear avenues exist for the acceleration of this solver with structured linear algebra packages, as well as the expansion from the deformed grid to a truly unstructured mesh. This solver is verified using a Method of Manufactured Solutions order-of-accuracy study which proves that maximal convergence order is still obtainable. Additionally, eigenvalue results are pre-

sented for 2D problems derived from the common benchmark C5G7, and are compared with reference obtained from the Method of Characteristics code MPACT. This analysis reveals typical errors characteristic of all methods based upon a spherical harmonics representation of the angular flux, further indicating proper functioning of the solver procedure. Overall, we establish a new discretization and solver framework for low-order transport equations. Future work may focus on developing high performance parallel implementations, more numerically robust implementations, and more generic unstructured mesh implementations.

Chapter 1

Introduction

1.1. Motivation

There are enormous incentives for improvements of all kinds in Partial Differential Equation (PDE) solvers for reactor applications. In terms of accuracy, a more accurate solver increases operational knowledge of the reactor core; allowing a skilled designer to create systems that can push the limits of reactor power and lifespan. In terms of speed, a faster solver not only allows designers to work faster, but also allows regulators and evaluators to more quickly analyze new designs. In terms of capability, solvers that can simulate more regimes allow the exploration of new reactor technologies that may one day become the industry standard. The entire space of potential improvements to reactor design hinge on simulations that can augment the current state of the art.

The work in this dissertation is designed around reaching a code that has a large domain of applicability at a high level of performance, while accepting some trade-offs in absolute accuracy. This does not mean that we seek a poor or weakly-converged solution; but rather that proven and conventional low-order multigroup discretizations are the principal targets as opposed to techniques such as Method of Characteristics (MOC) [4] or Monte Carlo [8]. The ones targeted specifically are those discretizations arising from the simplified spherical harmonic approximations of transport, such as diffusion or Simplified P_N (SP_N).

These discretizations have been extensively used, but they have known and well-studied shortcomings. For example, they are known to struggle in regimes where the angular flux is very sharply peaked in angle. While in theory higher-order expansions of N can fix this, they cannot overcome the approximations inherent to the assumptions inherent to this "Sim-

plified" approach. Conventional wisdom and experience generally suggests that expansions past SP_5 or SP_7 are not worthwhile in terms of accuracy gain vs. the added computational cost [47]. However, recent advances in the theory surrounding these discretizations have emerged as a Generalized SP_N (GSP_N) [13], which will be discussed further in Section 1.4.1. It is because of this combination of both a reliable history and new developments in the theory that these simplifications are selected for study in this dissertation.

1.2. The Weak Formulation

1.2.1. PDE Solver Theory

The field of solver theory is one of the foundational branches of numerical analysis, which primarily concerns itself with the numeric solution of PDEs. One of the principal concepts in this theory is known as the *Weak Formulation* of a PDE. Consider a partial differential equation for the function ϕ, defined below over a domain $\mathcal{D}$:

$$-\nabla^2\phi(\vec{x}) = f(\vec{x}) \qquad \text{for} \qquad \vec{x} \in \mathcal{D}, \tag{1.1a}$$

$$\phi(\vec{x}) = \phi_0(\vec{x}) \qquad \text{for} \qquad \vec{x} \in \partial\mathcal{D}. \tag{1.1b}$$

Here the function f represents an inhomogeneous source term, while the function ϕ_0 represents a Dirichlet boundary condition. This specific equation (the Poisson Equation with Dirichlet boundary conditions) is chosen as an example because it is among the simplest non-trivial elliptic PDEs.

The representation of Equation (1.1) is known as the *Strong Form*, which is typically used if one intends to solve a PDE by hand. This formulation does not readily make it clear how this equation may be solved in a discrete manner, however; particularly when the functions f and ϕ_0 may be allowed to take on any arbitrary definition. Rather than work with these functional equalities to solve PDEs numerically, we work instead with the *Weak Form*:

$$-\int_{\mathcal{D}} \psi(\vec{x})\nabla^2\phi(\vec{x})\, d\vec{x} = \int_{\mathcal{D}} \psi(\vec{x}) f(\vec{x})\, d\vec{x} \qquad \text{for all} \qquad \psi(\vec{x}) \in L^2(\mathcal{D}), \tag{1.2a}$$

$$\phi(\vec{x}) = \phi_0(\vec{x}) \qquad \text{for} \qquad \vec{x} \in \partial\mathcal{D}. \tag{1.2b}$$

The weak form states that the above relation holds true for all square-integrable functions

ψ. Strictly speaking, this representation is obtained by simply operating on Equation (1.1) both sides of Equation (1.2a) by $\int_{\mathcal{D}} \psi(\vec{x})(\cdot)\, d\vec{x}$, which is known as the *dual* of the function ψ, sometimes written ψ^*. This *dual* ψ^* is a bounded linear operator on the region $\mathcal{D}$, which is uniquely defined for any square-integrable function ψ.

While this equation appears more complicated it is actually a significant simplification; instead of the functional statement of equality in the strong form, we have a much more tractable scalar comparison. Additionally, by changing and using different functions ψ, this can provide us with as many unique algebraic equations as we might choose. Note however, that the topic of boundary conditions has not entirely improved. How we treat the boundary conditions we have (and the manner in which they transform) is a complex topic which will come up many times throughout this research.

1.2.2. Discretization

A fundamental part of solver theory is the process of discretization. Because no computer can work in truly infinite-precision arithmetic, the continuous problem must somehow be discretized into a finite number of unknowns. This discretization is not always as simple as, for example, sampling a solution at a finite number of points. Rather, it is more common in modern methods to instead represent the solution as a linear combination of a finite number of continuous functions. While the result may not seem discretized in that it has a precisely-defined value everywhere, this is still a discretization because we have reduced a concept which has infinite dimensionality (the true solution) to one with finite dimensionality (a linear combination).

In general the weak form of a PDE is much more amenable to this type of discretization than the strong form, for reasons which will become clear in the next section. Suffice to say that by choosing a finite set of representative and trial functions, we can obtain the set of algebraic equations we seek. The most straightforward way to translate this concept of the weak formulation into algebraic equations is known as the Weighted Residuals Method (WRM), also called the Method of Weighted Residuals [17, 18].

The Weighted Residuals Method

The underlying idea behind the WRM is quite general, and is in fact the basis of a wide range of popular methods; including virtually every form of Finite Element Method (FEM),

Nodal Method [19], Collocation method [29], and more. The fundamental derivation starts by writing a representation of the solution in terms of known 'basis functions' as:

$$\phi(\vec{x}) \cong \sum_{j=1}^{N} \phi_j b_j(\vec{x}). \tag{1.3}$$

This equation may then be substituted into the strong form representation (Equation (1.1)) and rearranged like so:

$$f(\vec{x}) - \sum_{j=1}^{N} \phi_j \mathcal{T} b_j(\vec{x}) \approx 0, \tag{1.4}$$

which is a statement that, given perfect coefficients ϕ_j of a finite number of basis functions the residual should approach zero.

We have assumed linearity of the differential operator $\mathcal{T}$; while this is not a requirement for WRM in general, it will be assumed throughout this work, since the SP_N and GSP_N approximations are linear. From here we convert to the weak form as described in Section 1.2. We do this by conceptualizing the dual ψ^* as an inner product, which is simpler to write and understand than a complicated integral formulation. ϕ^* is then defined as:

$$\psi^* = (\psi, \cdot)_{\omega_0}^{\mathcal{D}} = \int_{\mathcal{D}} \psi(\vec{x})(\cdot)\, \omega_0(\vec{x}) d\vec{x}, \tag{1.5}$$

where the integration is carried out on some domain $\mathcal{D}$ with a *principal weighting function*. This is a property of the inner product, and will be discussed later in this section.

The weak form is a statement that, when the above operator is applied to both sides of Equation (1.4), the equation will hold for all functions $\psi(\vec{x})$ that lie within some functional space. For the purposes of this work, the space will be taken as the set of square-integrable functions over the domain, $L^2(\mathcal{D})$. To form the WRM discretization, we must choose a set of N weighting functions. Applying the dual of each of these weighting functions to the residual and asserting equality to zero then generates a set of N algebraic equations. Assuming the underlying PDE is relatively well-behaved, this will create a well-posed problem. Rep-

resenting these weighting functions as $\omega_i(\vec{x}), 1 \leq i \leq N$, a set of equations are obtained:

$$\left(\omega_i(\vec{x}), f(\vec{x}) - \mathcal{T}\sum_{j=1}^{N}\phi_j b_j(\vec{x})\right)_{\omega_0}^{\mathcal{D}} = 0, \qquad 1 \leq i \leq N. \tag{1.6}$$

Note that the set of functions $\omega_i(\vec{x})$ is distinct from the principal weighting function $\omega_0(\vec{x})$. While they are denoted similarly, they possess a different purpose. The principal weighting function is there primarily to simplify the use of certain orthogonal functions. For example, the Chebyshev polynomials may be used as a basis function set $b_j(\vec{x})$; but doing so efficiently requires setting the principal weighting function ω_0 such that the Chebyshev polynomials are orthonormal. In this sense, the principal weighting function is a property of the inner product, and is distinct from the functions $\omega_i(\vec{x})$ which are a property of the method as a whole.

Many methods exist to choose the weights ω_i and these choices lead to a huge number of different methods. The N algebraic equations obtained have dependent variables ϕ_j, and the properties of the algebraic system and its solution depend heavily upon the choice of functions (both $\omega_i(\vec{x})$ and $b_i(\vec{x})$); particularly the spaces they span. It is this mechanism which drives the differences between Nodal Expansion Method (NEM) and FEM solutions of a given elliptic problem. To illustrate this phenomenon, consider a simple case where one is given N node points x_i for a 1D problem, and the weighting functions are defined simply as a delta-function $\delta(x - x_i)$. The algebraic equations simplify down quite neatly into:

$$\sum_{j=1}^{N}\phi_j \mathcal{T} b_j(x_i) = f(x_i), \qquad 1 \leq i \leq N, \tag{1.7}$$

which is simply an expression of the Collocation method. In this way the function sets allow a great deal of customizability [17].

Boundary Conditions

There is one very important observation to be made concerning Equation (1.6); it does not contain any information concerning boundary conditions. Before getting into the specifics of how to include these, it is important to discuss how to handle large domains in a WRM.

Consider a full problem domain $\mathcal{D}$ composed of heterogeneous materials, and suppose we

want to discretize such a domain into smaller parts– each homogenous. This technique has significant advantages in the stability of the underlying method, since sharp discontinuities exist at cell boundaries where they can be treated specially; rather than relying on basis refinement to accurately capture sharp transition regions. To accommodate this, the domain $\mathcal{D}$ would be split into N_c distinct, non-overlapping subdomains $\mathcal{D}^c$ such that

$$\mathcal{D} = \bigcup_{c=1}^{N_c} \overline{\mathcal{D}^c} \qquad \text{and} \qquad \emptyset = \mathcal{D}^i \cap \mathcal{D}^j \ \forall i \neq j, \tag{1.8}$$

where $\overline{\mathcal{D}}$ represents the closure of $\mathcal{D}$. Basis and weighting functions are then modified to be cell-based by multiplying them with an additional 'characteristic function' defined as

$$\chi^c(\vec{x}) = \begin{cases} 1 & \vec{x} \in \mathcal{D}^c \\ 0 & \text{otherwise} \end{cases}, \tag{1.9}$$

and changing their argument to map to the appropriately smaller domain.

This is a more mathematically rigorous way to look at the situation, but clearly it complicates the underlying concepts. We use this point of view to exhibit the nature of interface conditions. This domain decomposition is a way to, without loss of generality, treat the separate homogenous regions as independent problems coupled by these interface conditions. These are much like a boundary condition, but slightly weaker in form; an example are the first-derivative jump condition at material interfaces for the flux in diffusion theory:

$$D_L \frac{d\phi^L}{dx}\Big|_{x=x_0} = D_R \frac{d\phi^R}{dx}\Big|_{x=x_0}, \tag{1.10}$$

where D, the diffusion coefficient, is a material-dependent property. In principle, neither ϕ^R nor ϕ^L is known, but knowing one provides a complete boundary condition for the other. Thus, it is the primary mechanism by which the domain is decomposed into smaller, coupled problems.

Whether an interface or a boundary condition is used, the information still must be injected into the discretization somewhere. Effectively, there are three options to choose from:

1. Insert a homogenous interface/boundary condition by only including basis functions which satisfy this condition.

2. Insert the interface/boundary condition through the weighting functions.

3. Insert the interface/boundary condition as a substitution into each algebraic equation.

Most transport discretizations involve a set of mixed, non-homogenous boundary conditions. Technique 1 relies on a homogenous condition, which removes it from consideration. Technique 2, inserting via the weighting functions, however, is possible. If properly done, it can even provide guarantees that the numeric solution will satisfy the boundary conditions exactly (even if the solution itself is not exact). This topic is discussed further in Section 2.1.1.

The third method refers to a transformation of the Laplacian term of a PDE. More specifically, an integration-by-parts or Gaussian identity is applied to this term which results in a set of equations containing only first-order derivatives and boundary values of the cells. This is discussed further in Section 2.2.

1.2.3. Functional Spaces in PDE Solvers

The vector spaces spanned by the sets of functions $[\omega_i]_{i=1}^{i=N}$ and $[b_i]_{i=1}^{i=N}$ have special significance in terms of the properties of the approximate solution. This is clearest when discussing span $\left\{[b_i]_{i=1}^{i=N}\right\}$; the approximate solution must lie in this vector space. The absolute minimum error in solving a given problem, therefore, is related to a hypothetical projection of the true solution into this space. For this reason, an approximate solution space that is as close as possible to the true solution space is highly desired.

The space spanned by the weighting functions span $\left\{[\omega_i]_{i=1}^{i=N}\right\}$ is more complicated but no less important. While the basis span is related to error bounds, the weighting span is more closely tied to the types of error modes allowed to be present in the approximate solution. This is perhaps clearest to see when discussing the Collocation Method, in which the weighting functions are defined as a set of delta functions with support at predetermined points. It is clear to see from Equation (1.7) that this essentially selects points from the domain at which the residual of the approximate solution will be exactly zero; that is, the PDE is (locally) satisfied exactly at these special points. This does not mean that the value of the approximate and true solutions are equal; but it does mean that the value of the approximate solution and its derivatives at that point exactly solve the PDE. Other choices are possible, which result in more complex behaviors; for example, in the case of an elliptic PDE, a weighting function which is constant will create an approximate solution that obeys a conservation law.

Later, the review of Nodal Methods and Finite Element Methods of Chapter 2 will use these observations to understand the fundamental differences between both categories of methods. The observations discussed in this section have strong impacts on how the solutions of each method should be expected to behave.

1.3. History

SP_N is a simplification of the P_N transport equations first conceived by Gelbard [21]. The P_N equations themselves arise when one projects the angular flux into a finite number of angular moments, or spherical harmonics, assigning coefficients to some number of these functions. However, this system grows very quickly with the number of moments N and is challenging to solve; the number of PDEs grows as $(N+1)^2$, and they are coupled together in non-trivial ways. Gelbard created SP_N out of this system by noting that in 1D slab geometry, the streaming term of the P_N equations was extraordinarily simple. Rather than a mess of couplings at the streaming term, the result was a trivial number of problems of the same form as the simpler diffusion equation. By simply replacing the x coordinate and $\frac{d}{dx}$ operators with their 3D analogues $\vec{r}$ and ∇, a 3D form is obtained which Gelbard called the SP_N equations.

Needless to say, this approach is not very rigorous, something Gelbard knew perfectly well. However, this was not done carelessly; but rather from a place of deep understanding about the differential operators involved. The relative ease of implementation into an existing diffusion code meant that it was occasionally applied by various researchers, despite having a relatively weak theoretical backing. Every time SP_N was used, the method seemed to provide results with enhanced accuracy, and these successes eventually led to a much more rigorous analysis of SP_N theory. This was the subject of two successful analyses which confirmed SP_N's place in the hierarchy of transport simplifications. These studies were carried out by Larsen and Pomraining asymptotically [35] and continued variationally by Pomraining [54].

Pomraining's work determined conclusively that SP_N was related asymptotically to the slab geometry P_N equations, while Larsen et al. presented an asymptotic derivation demonstrating that SP_N represented an asymptotic correction to diffusion theory. While both studies were informed by the structure of the equations Gelbard arrived at, they nonetheless serve as important independent confirmations of SP_N as its own unique representation of "low-order" transport. Additional variational analyses were developed to derive the SP_N

equations, which are in some ways more convincing than the asymptotic version, if less intuitive. Of particular note in these later studies is Brantley and Larsen's work which produced a more rigorous derivation of the boundary conditions [10].

In addition to these derivations, there were others performed prior to 1990 which are of interest; particularly those that identified situations where P_N reduces to SP_N. This prospect is simultaneously deeply unexpected and profoundly interesting. P_N has in principle $(N+1)^2$ unknown functions, while SP_N has only $(N+1)/2$. For these systems to return approximate solutions which are in every aspect identical is quite shocking. This equivalence is discussed in Gelbard's own original paper, as well as a recent review article by McClarren [47]. Of additional note is a terse abstract in ANS transactions by Selengut [57] who claimed a P_3 equivalent SP_3 material interface condition. Such a condition could theoretically provide P_N-SP_N equivalence in piecewise-homogenous regions, which is by far the most common implementation of existing transport simulation codes. However, it is rather doubtful that this is achievable in general [11]. Nonetheless, it is somewhat of an open topic in the field.

...In any case, the boundary conditions as derived by any of these theories remain somewhat of a sticking point. They include derivations of boundary conditions, but these derivations almost uniformly assume a local 1D behavior in the flux. While this forms a very natural extension of SP_N as it was originally derived (from the 1D P_3 equations) it would seem to pose a potential issue in multiple dimensions; and it is this issue that GSP_N is formulated to solve.

Work on the various simplifications to transport continues to evolve as new approximations based on P_N emerge or change. The most recent development in this saga is the introduction of a so-called GSP_N by Chao [13], who reformulated SP_N in order to relax the pre-existing requirement that a solution be 'locally 1D' near material interfaces. Dropping this requirement involves substantial mathematical work, which can be seen in full in the entire set of Chao's papers [11, 12, 13, 14].

The key concept enabling this is the realization that in SP_N, the principal simplification from full transport is that the n-th angular flux moment $\phi_n(\vec{x})$ is rotationally symmetric about the vector of neutron current $\nabla\phi_n(\vec{x})$. This idea plays a leading role in all the math that follows; the relaxation of locally 1D behavior is altered in favor of this key concept. This does not affect the 'bulk' governing equations but radically changes how boundary and interface conditions are derived. Additionally, Chao takes care to call out that the interface conditions which manifest in GSP_3 are exactly the form evoked in Selengut's paper in his

work studying $SP_3 - P_3$ equivalence. This claim would be the strongest evidence yet that the P_N system of equations may be reduced to a SP_N-like system in the case of piecewise homogenous regions, and would represent a significant advancement in low-order transport calculations.

1.4. SP_N Transport Theory

The targets of this work are low-order transport approximations which eliminate the explicit dependence of the neutron flux on angle, instead introducing functions which are angular moments of the true flux; such as scalar flux and neutron current. Specifically, those that originate from the P_N approximation, where the angular flux is considered to be a linear combination of spherical harmonic functions in angle, up to order N. These moments are themselves spatially dependent, since the P_N approximation is one of angular dependence:

$$\psi(\vec{x}, \hat{\Omega}) = \sum_{n=0}^{N} \sum_{m=-N}^{N} \Phi_{nm}(\vec{x}) Y_{nm}(\hat{\Omega}), \tag{1.11}$$

where often, one takes Y_{nm} as the real spherical harmonic for simplicity. P_N itself gives rise to a family of low-order transport approximations; though by far the most significant is SP_N.

1.4.1. Derivation of the SP_3 Equations

A full reproduction of the various works deriving SP_N from transport [10, 35, 54] is out of scope of this dissertation. It may be achieved through either an asymptotic or variational approach and the references listed are quite rigorous in deriving both the SP_N method (principally SP_3) and its boundary conditions. However, a brief derivation similar to the original Gelbard paper [21] will be included here, starting from the time-independent multigroup neutron transport equation:

$$\underbrace{(\hat{\Omega} \cdot \nabla)\psi^g(\vec{x}, \hat{\Omega})}_{\text{Streaming}} + \underbrace{\Sigma_t^g \psi^g(\vec{x}, \hat{\Omega})}_{\text{Total Interaction}} = \underbrace{\frac{1}{4\pi} \sum_{g'=1}^{G} \int_{4\pi} \Sigma_{s, g \leftarrow g'}(\hat{\Omega} \cdot \hat{\Omega}') \psi^g(\vec{x}, \hat{\Omega}') \, d\hat{\Omega}'}_{\text{Scattering Source}} + \underbrace{\frac{1}{4\pi} Q^g(\vec{x})}_{\substack{\text{Inhomogeneous or} \\ \text{Fission Source}}} . \tag{1.12}$$

For the purpose of this derivation, we will consider only a single group and allow the source terms to include the group-to-group scattering source. Thus,

$$\underbrace{(\hat{\Omega}\cdot\nabla)\psi(\vec{x},\hat{\Omega})}_{\text{Streaming}}+\underbrace{\Sigma_t\psi(\vec{x},\hat{\Omega})}_{\text{Total Interaction}}=\underbrace{\frac{1}{4\pi}\int_{4\pi}\Sigma_s(\hat{\Omega}\cdot\hat{\Omega}')\psi(\vec{x},\hat{\Omega}')\,d\hat{\Omega}'}_{\text{Scattering Source}}+\underbrace{\frac{1}{4\pi}S(\vec{x})}_{SourceTerms}. \tag{1.13}$$

From here, one may take the equation into a 1D, azimuthally isotropic form; effectively, the P_N slab geometry. Here, the P_N equations exhibit very simple coupling, a fact exploited in deriving SP_N:

$$\hat{\Omega}_z\frac{d}{dx}\psi(x,\hat{\Omega}_x)+\Sigma_t\psi(x,\hat{\Omega}_x)=\frac{1}{2}\int_{-1}^{1}\Sigma_s(\hat{\Omega}_x\hat{\Omega}'_x)\psi(x,\hat{\Omega}'_x)\,d\Omega'_x+\frac{1}{4\pi}S(x). \tag{1.14}$$

One may expand ψ as in Equation (1.11) and operate on Equation (1.14) by $\int_{4\pi}Y_{n0}(\hat{\Omega})(\cdot)\,d\hat{\Omega}$. Assuming the real spherical harmonics are used, this expands the flux as desired into the azimuthally symmetric spherical harmonic functions. The factor N is arbitrary, but as we seek the SP_3 equations, we choose $N=3$ to obtain 4 coupled differential equations. Writing the flux moments Φ_{n0} as ϕ_n:

$$\frac{d}{dx}\phi_1(x)+\Sigma_t\phi_0(x)=\Sigma_{s0}\phi_0(x)+\frac{1}{2}S(x), \tag{1.15a}$$

$$\frac{d}{dx}\left(\frac{1}{3}\phi_0(x)+\frac{2}{3}\phi_2(x)\right)+\Sigma_t\phi_1(x)=\Sigma_{s1}\phi_1(x), \tag{1.15b}$$

$$\frac{d}{dx}\left(\frac{2}{5}\phi_1(x)+\frac{3}{5}\phi_3(x)\right)+\Sigma_t\phi_2(x)=\Sigma_{s2}\phi_2(x), \tag{1.15c}$$

$$\frac{d}{dx}\left(\frac{3}{7}\phi_2(x)\right)+\Sigma_t\phi_3(x)=\Sigma_{s3}\phi_3(x), \tag{1.15d}$$

where the usual P_N closure ($\Phi_{nm}(\vec{x})=0$ for $n>N$) is used. If the conventional removal cross-sections and diffusion coefficient are defined as

$$\Sigma_{rn}=\Sigma_t-\Sigma_{sn},\qquad D_0=\frac{1}{3}(\Sigma_{r1})^{-1},\qquad\text{and}\qquad D_2=\frac{9}{35}(\Sigma_{r3})^{-1}, \tag{1.16}$$

the form of Equation (1.15) becomes a bit simpler. Equations (1.15b) and (1.15c) may be used to solve for the odd moments in terms of even ones, yielding:

$$\frac{d}{dx}\phi_1(x) + \Sigma_{r0}\phi_0(x) = \frac{1}{2}S(x), \tag{1.17a}$$

$$-\frac{1}{\Sigma_{r1}}\frac{d}{dx}\left(\frac{1}{3}\phi_0(x) + \frac{2}{3}\phi_2(x)\right) = \phi_1(x), \tag{1.17b}$$

$$\frac{d}{dx}\left(\frac{2}{5}\phi_1(x) + \frac{3}{5}\phi_3(x)\right) + \Sigma_{r2}\phi_2(x) = 0 \tag{1.17c}$$

$$-\frac{1}{\Sigma_{r3}}\frac{d}{dx}\left(\frac{3}{7}\phi_2(x)\right) = \phi_3(x). \tag{1.17d}$$

These may be collapsed into second-order form by plugging Equations (1.17b) and (1.17d) into Equations (1.17a) and (1.17c). After doing so, the second-order derivatives terms are made into Laplacians in order to obtain a set of multidimensional equations:

$$-D_0\nabla^2\Big(\phi_0(\vec{x}) + 2\phi_2(\vec{x})\Big) + \Sigma_{r0}\phi_0(\vec{x}) = Q(\vec{x}), \tag{1.18a}$$

$$-D_2\nabla^2\phi_2(\vec{x}) + \Sigma_{r2}\phi_2(\vec{x}) = \frac{2}{5}\Big[\Sigma_{r0}\phi_0(\vec{x}) - Q(\vec{x})\Big]. \tag{1.18b}$$

Note this source Q is not the same as the original $Q(\vec{x})$ in Equation (1.12), but is either the inhomogeneous (for fixed-source problems) or fission (for eigenvalue problems) source plus group-to-group scattering, as these pertain to the scalar flux.

In this formulation the scalar flux is equivalent to the function ϕ_0. However, it is convenient to instead use the common convention of using a representation $\hat{\phi}_0 \equiv \phi_0(\vec{x}) + 2\phi_2(\vec{x})$ as the unknown, because this better retains the diffusion form of the two equations:

$$-D_0^g\nabla^2\hat{\phi}_0^g(\vec{x}) + \Sigma_{a0}^g\left[\hat{\phi}_0^g(\vec{x}) - 2\phi_2^g(\vec{x})\right] = S^g(\vec{x}), \tag{1.19a}$$

$$-D_2^g\nabla^2\phi_2^g(\vec{x}) + \left[\Sigma_{a2}^g + \frac{4}{5}\Sigma_{a0}^g\right]\phi_2^g(\vec{x}) = \frac{2}{5}\left[\Sigma_{a0}^g\hat{\phi}_0^g(\vec{x}) - S^g(\vec{x})\right]. \tag{1.19b}$$

Aside from the questions of mathematical rigor introduced by Equation (1.18) (let alone casting the equation into a 1D azimuthally-symmetric form), this derivation is somewhat opaque and does not tell us much about how we may expect this approximation to behave.

The detailed derivations of Larsen, Brantley, and Pomraining [10, 35, 54] provided significant insights about the performance of SP_N; namely, where it performed well, where it broke down, and the reasons why. These are most intuitively understood from the asymp-

totic analysis; as they arise naturally from the nature of the derivation. In these analyses, a small dimensionless parameter is introduced to the P_N equations such that

$$\sigma_t \to \frac{\sigma_t}{\epsilon}, \qquad \sigma_s \to \frac{\sigma_s}{\epsilon}, \qquad \sigma_a \to \epsilon^2 \sigma_a, \qquad Q \to \epsilon Q. \tag{1.20}$$

Therefore, in principle, the approximation will be most accurate when:

- Total and Scattering cross-sections are of similar order, and much larger than absorption ($\sigma_t \gtrsim \sigma_s \gg \sigma_a$) (i.e. scattering-dominated systems)
- The source terms (encompassing external, scattering, and fission) are small ($1 \gg Q$)

In practice, the theory maintains a surprising amount of flexibility with regard to where it tends to be valid. The above assumptions need not be overwhelmingly true, and it is rare (though possible) that diffusion theory outperforms SP_N. Because diffusion itself may also be derived via this analysis, by dropping terms of order ϵ^4 relative to SP_N's terms of ϵ^6, this tends to occur when the assumptions made above are categorically incorrect; which is uncommon in most reactor designs and materials.

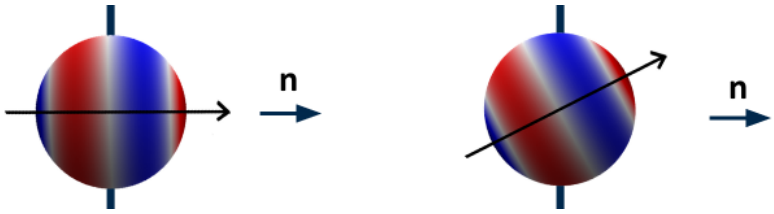
1.4.2. Boundary and Interface Conditions

The interface conditions obtained by Brantley, Larsen, and Pomraining throughout their various publications may be expressed as an equality on the boundary of two regions, i and j, between certain quantities. For $\vec{x}$ on this boundary, and a unit vector $\hat{n}_i$ or $\hat{n}_j$ representing the outward-pointing normal with respect to either region:

$$\hat{\phi}_0^i(\vec{x}) = \hat{\phi}_0^j(\vec{x}), \qquad D_0^i(\hat{n}_i \cdot \nabla)\hat{\phi}_0^i + D_0^j(\hat{n}_j \cdot \nabla)\hat{\phi}_0^j = 0, \tag{1.21a}$$

$$\phi_2^i(\vec{x}) = \phi_2^j(\vec{x}), \qquad D_2^i(\hat{n}_i \cdot \nabla)\phi_2^i + D_2^j(\hat{n}_j \cdot \nabla)\phi_2^j = 0, \tag{1.21b}$$

which are notably similar to the familiar diffusion interface conditions. This extension of this simple form to the SP_3 equations is a consequence of the 'locally-1D' assumption, depicted in Figure 1.1. The assumption presumes that the gradient of the scalar flux (and thus, the principal axis of the SP_N expansion) is parallel to the normal direction at that interface (top). In principle, however, there is no such restriction, and these may be misaligned (bottom); which may be significant in multidimensional problems. An alternate formulation of these



(a) Locally-1D Assumption (b) Relaxed Assumption

Figure 1.1: Spherical Harmonic Moments and the Locally 1D Assumption

conditions is a set of conserved quantities, denoted with an underline:

$$\underline{\phi}_0(\vec{x}) = \hat{\phi}_0(\vec{x}), \tag{1.22a}$$

$$\underline{J}_0(\vec{x}, \hat{n}) = D_0(\hat{n} \cdot \nabla)\hat{\phi}_0(\vec{x}), \tag{1.22b}$$

$$\underline{\phi}_2(\vec{x}) = \phi_2(\vec{x}), \tag{1.22c}$$

$$\underline{J}_2(\vec{x}, \hat{n}) = D_2(\hat{n} \cdot \nabla)\phi_2(\vec{x}). \tag{1.22d}$$

These may be separated into "flux-like" and "current-like" conserved quantities which have even- and odd-ordered derivatives, respectively.

It is also possible to construct a representation based upon partial currents, which are sometimes more useful; consider two adjacent cells with scalar fluxes $\phi^L(x)$ and $\phi^R(x)$, and their values at a hypothetical central boundary $x = x_0$. Then, recall the definition of the partial current operators under diffusion theory:

$$\mathcal{J}^+(D, \phi(x)) = \frac{1}{4}\phi(x) - \frac{D}{2}\frac{d\phi}{dx} \qquad \text{and} \qquad \mathcal{J}^-(D, \phi(x)) = \frac{1}{4}\phi(x) + \frac{D}{2}\frac{d\phi}{dx}. \tag{1.23}$$

These definitions apply due to the continuity relations in Equation (1.21) on the simple principle that the two moments are fully decoupled from one another under this formulation of the interface conditions. The conservation laws may then be expressed as either Equation (1.24a) or Equation (1.24b) below:

$$\phi^L(x_0) = \phi^R(x_0), \qquad D^L\frac{d\phi^L}{dx}\Big|_{x=x_0} = D^R\frac{d\phi^R}{dx}\Big|_{x=x_0}, \tag{1.24a}$$

$$\mathcal{J}^+(D^L, \phi^L)\Big|_{x=x_0} = \mathcal{J}^+(D^R, \phi^R)\Big|_{x=x_0}, \qquad \mathcal{J}^-(D^L, \phi^L)\Big|_{x=x_0} = \mathcal{J}^-(D^R, \phi^R)\Big|_{x=x_0}. \tag{1.24b}$$

The equivalence of these representations can be seen by simply noting that

$$2\left(\mathcal{J}^{+}(D,\phi)\Big|_{x=x_0}+\mathcal{J}^{-}(D,\phi)\Big|_{x=x_0}\right)=\phi \quad \text{and} \quad \mathcal{J}^{+}(D,\phi)\Big|_{x=x_0}-\mathcal{J}^{-}(D,\phi)\Big|_{x=x_0}=D\frac{d\phi}{dx}. \tag{1.25}$$

The boundary conditions are a bit of a separate matter; though they still look quite familiar

$$\frac{1}{2}\hat{\phi}_0(\vec{x})+D_0^i(\hat{n}_i\cdot\nabla)\hat{\phi}_0(\vec{x})=\frac{3}{8}\phi_2(\vec{x})+\int_0^{2\pi}\int_{-1}^{0}2P_1(|\mu|)\psi^b(\vec{x},\mu,\varphi)\,d\mu\,d\varphi, \tag{1.26a}$$

$$\frac{21}{40}\phi_2(\vec{x})+D_2^i(\hat{n}_i\cdot\nabla)\phi_2(\vec{x})=\frac{3}{40}\hat{\phi}_0(\vec{x})+\frac{3}{5}\int_0^{2\pi}\int_{-1}^{0}2P_3(|\mu|)\psi^b(\vec{x},\mu,\varphi)\,d\mu\,d\varphi. \tag{1.26b}$$

The generally familiar form is, again, due entirely to the assumption of local 1D behavior.

1.4.3. The Generalized SP_N Extension

In Chao's formulation, GSP_N [11, 12, 13, 14], the locally 1D assumption of Figure 1.1 is claimed to be significantly relaxed or eliminated. This formulation derives certain quantities which must be conserved at interfaces, which are then used to define interface and boundary conditions. These come in two types: flux-like quantities (denoted with a $\underline{\phi}$) and current-like quantities (denoted with a $\underline{J}$); much like in Equation (1.22). However, Chao's conserved quantities are significantly more complicated. For isotropic scattering and a defined unit normal $\hat{n}$, we have:

$$\underline{\phi}_0(\vec{x})=\phi_0(\vec{x}), \tag{1.27a}$$

$$\underline{J}_0(\vec{x},\hat{n})=-\frac{1}{3\Sigma_t}(\hat{n}\cdot\nabla)(\phi_0(\vec{x})+2\phi_2(\vec{x})), \tag{1.27b}$$

$$\underline{\phi}_2(\vec{x},\hat{n})=\phi_2(\vec{x})-\frac{3}{2}\left(\nabla^2-(\hat{n}\cdot\nabla)^2\right)\left(\frac{2}{15\Sigma_t^2}\phi_0(\vec{x})+\frac{11}{21\Sigma_t^2}\phi_2(\vec{x})\right), \tag{1.27c}$$

$$\begin{aligned}\underline{J}_2(\vec{x},\hat{n})=&-\frac{9}{35}\frac{(\hat{n}\cdot\nabla)}{\Sigma_t}\left[\phi_2(\vec{x})-\frac{5}{2}\frac{\left(\nabla^2-(\hat{n}\cdot\nabla)^2\right)}{\Sigma_t^2}\left(\frac{2}{15}\phi_0(\vec{x})+\frac{11}{21}\phi_2(\vec{x})\right)\right]\\&-\frac{2}{15}\frac{(\hat{n}\cdot\nabla)}{\Sigma_t}\left(\phi_0(\vec{x})+2\phi_2(\vec{x})\right).\end{aligned} \tag{1.27d}$$

These quantities are essentially corrections on the boundary to account for the fact that the flux gradient is no longer assumed to be parallel to the surface normal as it is in the Brantley-Larsen formulation. Because of this, handling the angular dependence on the boundary is more complicated and conservation of these quantities (flux moments and net current moments) must be adjusted to match this new relaxation.

One of the significant complications of GSP_N is that these quantities all exist only on the boundary, except for the zeroth-order flux moment ϕ_0. The corrected second-order flux moment is defined only on the boundary, and not within the internal space of a cell. By contrast, the Brantley-Larsen conditions have the property where the flux-like conserved quantities are all continuous and defined everywhere; though current values are still purely a boundary quantity.

Despite all these changes, it is easy to note that if the locally-1D condition with respect to the material interface is enforced on Equation (1.27), they trivially reduce to the Brantley-Larsen forms. Simply take the $(\nabla^2 - (\hat{n}\cdot\nabla)^2)$ terms to zero and the result appears (with some constant differences due to different conventions in Chao's work).

The boundary conditions are significantly simpler to write, though their derivation is a complicated process involving some intermediate quantities used to present equations Equation (1.27) (which are not reproduced here). The vacuum boundary conditions for Chao's formulation with isotropic scattering are given as:

$$\underline{J}_0(\vec{x}) = \frac{1}{2}\underline{\phi}_0(\vec{x}) + \frac{5}{8}\underline{\phi}_2(\vec{x}), \tag{1.28a}$$

$$\underline{J}_2(\vec{x}) = -\frac{7}{24}\underline{\phi}_0(\vec{x}) + \frac{35}{24}\underline{\phi}_2(\vec{x}), \tag{1.28b}$$

with a general formulation for any albedo present in [13].

If we are to adapt these to the more conventional form where $\hat{\phi}_0(\vec{x}) = \phi_0(\vec{x}) - 2\phi_2(\vec{x})$, the

conserved quantities of Equation (1.27) transform to:

$$\underline{\phi}_0(\vec{x}) = \hat{\phi}_0(\vec{x}) - 2\phi_2(\vec{x}), \tag{1.29a}$$

$$\underline{J}_0(\vec{x}) = -\frac{(\hat{n}\cdot\nabla)}{3\Sigma_t}\hat{\phi}_0(\vec{x}), \tag{1.29b}$$

$$\underline{\phi}_2(\vec{x}) = \phi_2(\vec{x}) - \frac{3}{2}\left(\nabla^2 - (\hat{n}\cdot\nabla)^2\right)\left(\frac{2}{15\Sigma_t^2}\phi_0(\vec{x}) + \frac{9}{35\Sigma_t^2}\phi_2(\vec{x})\right), \tag{1.29c}$$

$$\underline{J}_2(\vec{x}) = -\frac{(\hat{n}\cdot\nabla)}{\Sigma_t}\left[\frac{2}{15}\hat{\phi}_0(\vec{x}) + \frac{9}{35}\phi_2(\vec{x})\right] + \frac{9}{14}\frac{(\hat{n}\cdot\nabla)(\nabla^2 - (\hat{n}\cdot\nabla)^2)}{\Sigma_t^3}\left[\frac{2}{15}\hat{\phi}_0(\vec{x}) + \frac{9}{35}\phi_2(\vec{x})\right]. \tag{1.29d}$$

1.5. Outline

In this dissertation, discretizations and solvers will be discussed that arise from foundational PDE solver theory such as the WRM. These discussions are presented as follows:

In Chapter 2, the origins and properties of various forms of both the NEM and the FEM are discussed. The most advantageous of these are synthesized into an original discretization applicable to the spherical harmonic simplifications of the linear Boltzmann transport equation. Chapter 2 also contains a 1D convergence study where the errors of these types of methods are examined and evaluated on even footing. This comprehensive study of varying discretizations represents original work not found cohesively in the literature on low-order transport discretization methods.

In Chapters 3 and 4, a solver recently published in the applied mathematics community [27, 45, 46] known as Hierarchical Poincaré-Steklov (HPS) is derived and adapted to governing equations of interest; Diffusion, SP_3, and GSP_3. While the algorithms behind HPS are not novel work, the adaptations made in order to apply this to the equations of interest are significant extensions that enable its application to a wider variety of problems. In addition to handling a more general set of boundary and interface conditions, the extensions paint a clear picture of how to enable the use of this solver on unstructured grids.

The theory is then extended to a 2D Method of Manufactured Solutions (MMS) verification study of a naïve implementation of this solver in Chapter 5. Verification is conducted for both a heterogeneous and non-separable problem, on deformed grids developed to simulate an unstructured mesh as a proof-of-concept. Chapter 6 extends this study into multigroup

reactor problems to examine how the governing equations of diffusion and SP_3 behave using this solver implementation on a sub-pin and pin-homogenized meshes. This consists of sub-pin mini-lattice problems derived from the C5G7 benchmark, as well as a reproduction of a pin-homogenized assembly problem studied by Chao et al. [14].

Lastly, Chapter 7 contains some implementation details on the solver as implemented. Specifically, this includes some topics on how efficient computations may be conducted as well as topics pertaining to the extension to truly unstructured meshes. Additionally, future areas involving the corner point balance, an open question resulting this work, are addressed. A performance analysis is reproduced with some details specific to low-order transport. The closing Chapter 8 ends discussing the conclusions of this work and a note on future research directions.

Chapter 2

Nodal and Finite Element Methods

Constructing a discretization is often an intensive process that starts with examining existing methods and identifying their strengths and shortcomings. The nodal methods invented for diffusion have been the standard for nearly 50 years. More recently, there has been increased use of FEMs for diffusion and low-order transport. However, their exact mathematical relationship to existing methods and relative performance has not been thoroughly explored.

This chapter will perform a review of such solvers as they apply to low-order transport. Sections Sections 2.1 and 2.2 will cover common existing implementations of the NEM and FEM, commenting on their differences, strengths, and weaknesses as the chapter progresses. Particular attention is paid to how each extends to multiple dimensions or the treatment of conserved quantities. The advantages of each are synthesized into a novel form of discretization in Section 2.3, here called the Legendre-Gauss-Lobatto method. Section 2.4 contains a comprehensive study of the methods described in this chapter as applied to a simple, 1D 2-group diffusion test problem; in a unified analysis which has not been found in existing literature.

2.1. Nodal Expansion Method

The NEM, also called the Method of Moments in other applications [48], is one of the classic methods of obtaining solutions to general PDEs. In fact, the method has quite a long history, with the core concepts stretching back almost as far as the origin of the weak formulation itself. The NEM as used for neutronics applications solidified in 1977, with Finnemann

providing a way to efficiently solve certain 3D problems via the framework of Transverse Integration (TI) [19]. This development propelled this new TI-NEM to be a workhorse for reactor calculations in subsequent decades.

2.1.1. Canonical 1D-NEM

The conventional form of the NEM uses the Legendre Polynomials as a basis function to represent the flux. Their properties of completeness mean that they are guaranteed certain forms of convergence and generate systems of equations that are solvable and not badly-behaved. Additionally, properties of orthogonality greatly simplify some aspects of the derivation. The 1D formulation is fairly simple, and arises from a desire to produce a numerical solution which:

1. Obeys neutron conservation (continuity) with respect to node-averaged fluxes.

2. Obeys current continuity at cell interfaces.

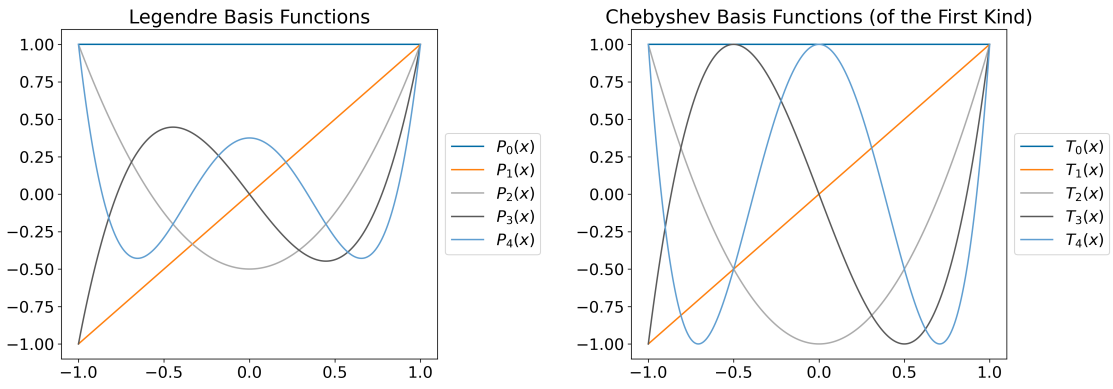
Because each internal cell boundary has two associated interface conditions, and the external boundaries each have one associated boundary condition, this requires three or more degrees of freedom per cell in 1D.

The end result is a framework for a discretization in 1D where three equations per cell are fully specified, and $N-3$ remain free, where N is the degrees of freedom per cell. In all varieties of NEM currently in use, a WRM approach is used to close this system, which requires $N-3$ additional functions to be specified. Broadly speaking, in a canonical NEM implementation, two choices are common. The first is known as a Galerkin or semi-Galerkin choice, and the second known as Moments weighting. In a semi-Galerkin approach, the basis functions are used as the weighting functions to close the system. By contrast, the moments approach simply uses the monomials $\{x^i, i \geq 0\}$.

Notably, both Legendre and Moments weighting include the constant function. This weighting is what must be enforced to satisfy node-averaged neutron conservation; so in practice, only 2 weighting functions need be excluded. Typically, the high-order polynomials are left out; since the low-order polynomials more effectively reduce interpolation error.

There is some evidence presented in the original Finnemann paper [19] that a Moments weighting results in a more accurate solution by virtue of computational stability. However, it also makes the algebra, and therefore, the implementation, significantly more complex; and

so the semi-Galerikin approach remains the most common. In exact-precision arithmetic, the two approaches will produce identical results, since both weightings span the same vector space.



(a) Legendre Basis Functions **(b) Chebyshev Basis Functions (of the First Kind)**

Figure 2.1: Common basis functions in Nodal Methods

2.1.2. NEM and Spectral Tau Methods

NEMs in general are a specific subset of a larger category of methods known as 'Spectral-' or Lanczos-τ methods [34], a general class of PDE discretizations based in part on the WRM procedure [30]. In Section 1.2.2, different methods by which BC information may be included in WRM procedures was discussed. Commonly, WRM supposes homogenous boundary conditions and a set of basis functions which obey these conditions; however, as we noted, many problems exist for which this is difficult or impossible. Spectral-τ methods were first derived to account for this deficiency, and the procedure is essentially the same as discussed in Section 2.1.1. A WRM procedure is used to generate a deficient set of algebraic equations using the lowest polynomial order bases as weighting functions. The deficient system is then closed by asserting that the trial solution obeys boundary (or interface) conditions.

There are a few differences, however; most notably, the standard basis functions in spectral-τ methods are Chebyshev polynomials of the first kind. Practical methods often make extensive use of recurrence relations to simplify the Laplacian or other differential terms. Additionally, p-refinement is most common when more accurate solutions are desired. That is, enhanced accuracy is obtained by increasing the polynomial order of the basis functions,

rather than subdividing the mesh. The use of Chebyshev polynomials provides a more robust guarantee of convergence (which is spectral in p) for general problems, including time-dependent ones. However, for the class of problems arising from Diffusion/SP_N, Legendre polynomials are sufficient to provide this guarantee, and are significantly easier to implement given their simpler orthogonality. Note however, that while p-refinement is rare in Nodal Methods, they still inherit this spectral convergence in p.

Formulation of BC as Operators

While in parts of this section, the WRM and NEM methods have been framed in opposition to one another, they are more closely related than it may seem. The conservation laws are trivial to write in the strong form; we simply enforce equality of the partial current operators defined in Equation (1.23) (or, equivalently, continuity of flux and net current). But they may also be written as functions to be used in a WRM procedure, further unifying the methods.

Consider a 1D, 1-group problem of either Diffusion or SP_N using a Brantley-Larsen BC formulation. Then, at a single boundary (suppose at some position $x = x'$), there exist two continuity relations associated with the positive and negative partial currents:

$$\omega^{\pm}(x) = \sum_{\text{all cells } c} (\mathcal{T}_c^*)^{-1}(\delta_{c,a} - \delta_{c,b}) \left(\sum_{i=1}^{\infty} \frac{1}{\|b_i\|^2} \mathcal{J}^{\pm}(D_c, b_i)(x') b_i(x) \right), \tag{2.1}$$

Where $\mathcal{T}_c^*$ is the adjoint of the differential operator for cell c, and $\mathcal{J}^{\pm}$ is the familiar partial current operator acting on the function b_i. The functional norm is defined using the same inner product relation as the WRM procedure; that is, $\|b_i\|^2 = (b_i, b_i)_{\omega_0}$. In most cases, this is the ordinary L^2 functional norm. Note also that this definition presumes the existence of an infinite set of basis functions spanning the space associated with this inner product; and further that the chosen basis is a finite subset of this. This is trivially satisfied by any set of polynomials which are 'complete', including the Legendre polynomials.

To understand why we claim this is the conservation weighting function, we consider the resulting algebraic equation when the corresponding residual is equated to zero. First, define a residual function in cell c as:

$$r_c(x) = Q_c(x) - \mathcal{T}_c \phi^c(x) = Q_c(x) - \mathcal{T}_c \sum_{i=1}^{N} \phi_i^c b_i(x), \tag{2.2}$$

and the true, analytic solution as $\bar{\phi}(x)$. Then the WRM procedure yields:

$$\left(\mathcal{T}_a^{-*}\sum_{i=1}^{\infty}\mathcal{J}^{\pm}(D_a,b_i)(x')\frac{b_i(x)}{\|b_i\|^2},r_a(x)\right)^{\mathcal{D}^a}-\left(\mathcal{T}_b^{-*}\sum_{i=1}^{\infty}\mathcal{J}^{\pm}(D_b,b_i)(x')\frac{b_i(x)}{\|b_i\|^2},r_b(x)\right)^{\mathcal{D}^b}=0. \quad (2.3)$$

From here, we use the definition of the adjoint to transpose it onto the other term in the inner product. We also use the identity:

$$\mathcal{T}_c^{-1}r_c(x)=\mathcal{T}_c^{-1}Q_c(x)-\phi^c(x)=\bar{\phi}_c(x)-\phi^c(x)=\sum_{i=1}^{\infty}\overline{\phi^c}_i b_i(x)-\sum_{i=1}^{N}\phi_i^c b_i(x).$$

We then have:

$$\left(\sum_{i=1}^{\infty}\mathcal{J}^{\pm}(D_b,b_i)(x')\frac{b_i(x)}{\|b_i\|^2},\sum_{i=1}^{\infty}\overline{\phi^c}_i b_i(x)-\sum_{i=1}^{N}\phi_i^a b_i(x)\right)_{\mathcal{D}^a}-$$
$$\left(\sum_{i=1}^{\infty}\mathcal{J}^{\pm}(D_b,b_i)(x')\frac{b_i(x)}{\|b_i\|^2},\sum_{i=1}^{\infty}\overline{\phi^c}_i b_i(x)-\sum_{i=1}^{N}\phi_i^b b_i(x)\right)_{\mathcal{D}^b}=0 \quad (2.4)$$

$$\sum_{i=1}^{\infty}\overline{\phi^a}_i\mathcal{J}^{\pm}(D_b,b_i)(x')-\sum_{i=1}^{N}\phi_i^a\mathcal{J}^{\pm}(D_b,b_i)(x')-$$
$$\left[\sum_{i=1}^{\infty}\overline{\phi^b}_i\mathcal{J}^{\pm}(D_b,b_i)(x')-\sum_{i=1}^{N}\phi_i^b\mathcal{J}^{\pm}(D_b,b_i)(x')\right]=0 \quad (2.5)$$

$$\mathcal{J}^{\pm}\left(D_a,\overline{\phi^a}\right)(x')-\mathcal{J}^{\pm}(D_a,\phi^b)(x')-\mathcal{J}^{\pm}\left(D_b,\overline{\phi^b}\right)(x')+\mathcal{J}^{\pm}(D_b,\phi^b)(x')=0 \quad (2.6)$$

$$\mathcal{J}^{\pm}(D_a,\phi^a)(x')=\mathcal{J}^{\pm}(D_b,\phi^b)(x') \quad (2.7)$$

Where the movement of the flux coefficients inside the $\mathcal{J}^{\pm}$ operator is made possible by the fact that $\mathcal{J}^{\pm}$ is linear in its second argument.

This procedure tells us two things; firstly, that using the somewhat abstract and convoluted weighting function is fundamentally equal to requiring conservation of the partial currents. In effect, using the more convenient conservation approach is no different from using a full WRM with this weighting function. Secondly, it tells us that these weighting functions $\omega^{\pm}$ span the space of solutions which *do not* follow the appropriate interface condition between subdomains a and b. This is the mechanism by which the WRM and all its

derived methods (Spectral-τ, Nodal Methods, and more) are able to produce algebraic equations which maintain fundamental conservation laws.

2.1.3. Analytic and Semi-Analytic Methods

A semi- or fully-analytic method is one which uses basis functions which are more closely tied to the true solution space of the underlying problem. These spaces are clearly defined under certain conditions that happen to be satisfied in some solver implementations; that is, a lack of up-scattering and a polynomial source term. In these cases, the solution space may be defined as the set of polynomials up to a certain order (determined by the polynomial order of the source) plus enough functions to span the null space of the differential operator. For 1D diffusion, these are the hyperbolic trigonometric functions `sinh` and `cosh`, which take as an argument the positional variable divided by diffusion length.

Purely Analytic Nodal Methods

The Analytic Nodal Method (ANM) [58] is perhaps the most popular out of the purely analytic methods. In 3D, the only approximation performed in this method is the approximation of quadratic transverse leakage (discussed in Section 2.1.4). However, most formulations are limited to 2 groups as the algebraic complexity of any higher-group formulations rapidly becomes extremely complex [37]. As conventionally implemented, the ANM typically computes only the node-averaged fluxes and face-averaged currents. Obtaining the flux in any more detail requires a flux reconstruction method.

This motivated the development of the Analytic Functional Expansion Nodal (AFEN) method, which solves the multidimensional diffusion equation while simultaneously generating the appropriate basis functions [50]. This process of finding the corresponding basis functions is similar to what is mentioned above; finding the null space of the operator. In this case, an eigenvalue/eigenvector decomposition is performed, and the resulting information is used to build the basis.

Additionally, a method described as the Unified Nodal Method (UNM) has also been developed based on the ANM with the goal of removing problematic instabilities [38]. Specifically, the ANM has a tendency to be unstable in cells which are very near-critical. The principal idea is to decouple the 2-group problem via similarity transformation, and solve the component parts. The basis functions for this approach are defined on $x \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ and may be

written

$$A_0(x) = 1, \tag{2.8a}$$

$$A_1(x) = 2x, \tag{2.8b}$$

$$A_2(x) = 6x^2 - \frac{1}{2}, \tag{2.8c}$$

$$A_3(x) = \frac{1}{K^g_{3,c}}\left[\frac{1}{2}\sinh(2\kappa^g_c x) - x\sinh(\kappa^g_c)\right], \tag{2.8d}$$

$$A_4(x) = \frac{1}{K^g_{4,c}}\left[\frac{\kappa^g_c}{2}\cosh(2\kappa^g_c x) + 3\left(x^2 - \frac{1}{2}\right)\sinh(\kappa^g_c) - 3\left(x^2 - \frac{1}{12}\right)\kappa^c_g\cosh(\kappa^c_g)\right], \tag{2.8e}$$

where if λ_p is the eigenvalue corresponding to a given group, then:

$$\kappa^g_c = \frac{h}{2}\sqrt{\lambda_p}, \qquad K^g_{3,c} = \kappa^g_c\cosh(\kappa^g_c) - \sinh(\kappa^g_c), \qquad K^g_{4,c} = (\kappa^g_c)^2\sinh(\kappa^g_c) - 3K^g_{3,c}. \tag{2.9}$$

The Semi-Analytic Nodal Method

A different evolution of the ANM led to the formulation of what is known as the Semi-Analytic Nodal Method (SANM). This evolution was motivated by having a clean and simple way to extend the accurate ANM solutions to an arbitrary number of groups [52]. In order to decouple the complicated scattering relations that arise from such a method, the scattering sources are projected into a polynomial form (Equation (2.10a) shows a 1D example). The flux, however, is still solved over an augmented polynomial space; typically some variation of $\mathcal{A}_g$ of Equation (2.10a), where $\mathcal{P}_N$ is the space of polynomials up to degree N. When the scattering source $S_{g\leftarrow g'}$ is computed, this is projected into polynomial space via the projection operator $P_{g'}$.

$$\mathcal{A}_g = \mathcal{P}_N \cup \{\sinh(x/L_g), \cosh(x/L_g)\}, \tag{2.10a}$$

$$S_{g\leftarrow g'}(x) = \sum_{g'\neq g} P_{g'}\Sigma_{g\leftarrow g'}\Phi_{g'}(x), \qquad \text{where} \qquad P_g : \mathcal{A}_g \mapsto \mathcal{P}_N. \tag{2.10b}$$

This projection injects some error, but the problem is much more tractable in its derivation and implementation. It is also significantly easier to adjust an existing NEM code to use this method than any of the fully analytic ones.

2.1.4. Transverse Integration

While 1D problems are all well and good for an abstract, theoretical analysis, real-world problems demand 2D or 3D treatment. TI is primarily a way to decompose a multi-D problem into a set of coupled 1D problems that are easier to solve and may be easily iterated upon. The underlying mechanism is true to the method's name; by identifying 'corridors' of nodes, or subdomains in the problem, the PDE may be integrated along the transverse dimensions such that one ends up with a set of simple 1D problems. Consider a 3D diffusion equation, in a single group for simplicity:

$$\begin{aligned} -\nabla \cdot (D\nabla\phi(\vec{x})) + \Sigma_r\phi(\vec{x}) &= Q(\vec{x}), \\ \frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} - D\frac{\partial^2}{\partial z^2}\phi(\vec{x}) + \Sigma_r\phi(\vec{x}) &= Q(\vec{x}). \end{aligned} \tag{2.11}$$

The transverse integration operator for a corridor parallel to the z-direction is:

$$\frac{1}{\Delta x \Delta y}\int_{-\Delta x/2}^{\Delta x/2}\int_{-\Delta y/2}^{\Delta y/2}(\cdot)\,dy\,dx. \tag{2.12}$$

Letting an overbar denote the flux and source terms which have been integrated by this operator, we see that applying this operator to the 3D diffusion equation yields:

$$\frac{1}{\Delta x}\int_{-\Delta x/2}^{\Delta x/2}\frac{\partial J_x}{\partial x}\,dx + \frac{1}{\Delta y}\int_{-\Delta y/2}^{\Delta y/2}\frac{\partial J_y}{\partial y}\,dy - D\frac{\partial^2}{\partial z^2}\bar{\phi}(z) + \Sigma_z\bar{\phi}(z) = \bar{Q}(z). \tag{2.13}$$

One key contribution of Finnemann was to provide the technique for estimating the leakage source from neighboring nodes in the transverse directions. This is done via a parabolic approximation with respect to the adjacent nodes [19]. This essentially resolves the issue of the added integral terms in Equation (2.13), fully transforming it into a 1D equation. The iteration proceeds by recomputing these transverse terms as necessary in between the iterations, which cycle between the x, y, and z directions. The flux representation for this method may be written like so:

$$\phi(\vec{x}) = \phi_0 P_0(\vec{x}) + \sum_{i=1}^{N}\phi_i^x P_i(x) + \sum_{i=1}^{N}\phi_i^y P_i(y) + \sum_{i=1}^{N}\phi_i^z P_i(z). \tag{2.14}$$

Equation (2.14) is specifically selected such that, for a given TI operator, only the constant term and the longitudinal variable remain after application.

Limits of the Transverse Integration Procedure

The transverse integration procedure has limits, however. Most evidently, one needs to be able to define these corridors and the transverse directions. In structured grids, either Cartesian or hexagonal, this is not an issue; but if a more complicated problem necessitates an unstructured grid, the method is wholly inapplicable, without some way to map the result to a satisfactory grid. Furthermore, there are to the authors' knowledge, as-yet unsatisfied questions pertaining to the transverse leakage source. Specifically, the necessity of the quadratic approximation is a curious one. More accurate approximations seem to do nothing for the method, and indeed have been found to make it worse in some cases [36]. This is highly counter-intuitive; and a suitable explanation has yet to be found.

2.1.5. Variational Nodal Methods

The limits of TI are a long-standing problem in the field of nuclear engineering, and this work is far from the first attempt at eliminating them. Another category of method which eliminates TI in multidimensional expansions is known as the Variational Nodal Method (VNM). The basic approach, as summarized by Zhang and Li in [61], is to, in terms of the spatial and angular variables:

1. **Write a functional that has the neutron transport equation as a stationary condition.** That is, if this functional F is stationary at a certain value of its arguments (for simplicity, assume two functions ρ and η), the solution of the neutron transport equation may be computed from these functions (ρ and η). In this context, stationary means that a perturbation of the arguments first-order in a small factor δ leads to a perturbation in the overall functional which is second-order or higher in δ.

$$F[\rho + \delta\rho, \eta + \delta\eta] = F[\rho, \eta] + O(\delta^2) \tag{2.15}$$

2. **Introduce a series of truncated spatial and angular expansions into the functional.** Effectively, this is a limitation of the arguments ρ and η in Equation (2.15) to a finite

number of degrees of freedom. Whether these are in space, angle, or some mixture is up to the designer of a given method.

3. **Find the stationary conditions of this "reduced" or "truncated" functional.**
   This process may be conducted by deriving the first order perturbation terms, and asserting that they cancel. This allows for the computation of the degrees of freedom in the arguments of F; and therefore, determining the flux per item 1.

The general nature in which one may write and solve for these spatial and angular expansions means that the VNM is extremely flexible in its design. However, this flexibility also makes it extremely difficult to categorize; as so much of the characteristics of the VNM depend on the exact method one uses to arrive at the final equations. The most typical application involves the second-order transport equation, derived using the even- and odd-parity angular fluxes:

$$\psi_e(\vec{x},\hat{\Omega}) = \frac{1}{2}\left[\psi(\vec{x},\hat{\Omega}) + \psi(\vec{x},-\hat{\Omega})\right] \quad \text{and} \quad \psi_o(\vec{x},\hat{\Omega}) = \frac{1}{2}\left[\psi(\vec{x},\hat{\Omega}) - \psi(\vec{x},-\hat{\Omega})\right]. \tag{2.16}$$

Assuming isotropic scattering, this gives rise to the second-order neutron transport equation:

$$\hat{\Omega}\cdot\nabla\psi_o(\vec{x},\hat{\Omega}) + \Sigma_t(\vec{x})\psi_e(\vec{x},\hat{\Omega}) = \Sigma_s(\vec{x})\int_{4\pi}\psi_e(\vec{x},\hat{\Omega})d\hat{\Omega} + Q(\vec{x}) \tag{2.17a}$$

$$\hat{\Omega}\cdot\nabla\psi_e(\vec{x},\hat{\Omega}) + \Sigma_t(\vec{x})\psi_o(\vec{x},\hat{\Omega}) = 0, \tag{2.17b}$$

which is used to derive a functional for the variational method. Such a choice will guarantee continuity of the neutron flux and current, as mentioned by Zhang and proven by Palmiotti, Carrico, and Lewis [20]. But this is not necessarily the only way one may arrive at such a conservative form.

While a full review of the numerical details comprising the many VNM implementations is out of scope of this document, the review provided by Zhang [61] is quite comprehensive. The implementation of VARIANT [51] is notable for being both the first, as well as perhaps the most widely used. It has also been implemented in ERANOS [16]. More context of the VNM, particularly in the backdrop of finite element and response matrix methods, is available in a review conducted by Lewis and Dilber [40].

2.2. Finite Element Methods

The FEM, like NEM has a very long history, and is much more popular in other fields. In general, the FEM has a simpler formulation and a wider range of applicability, since most formulations of FEM require only the first derivative of the functions used. While it was initially conceived in the early 40s [15], a full and rigorous analysis of the underlying theory did not emerge until the late 60s ([5, 49, 62], among others), at which point development of diffusion and transport simulation codes was already well underway. Furthermore, early versions of the FEM applied to multidimensional problems were quite memory-intensive, since they produced a fully-coupled system of equations not amenable to TI (see Section 2.2.1). Though FEM and NEM draw on many similar ideas, these paths very quickly diverged.

Broadly speaking, the common FEMs used today can be divided into the Discontinuous-Galerkin Finite Element Method (DG-FEM) and the Continuous Finite Element (CFEM) varieties. These come with their own unique advantages and disadvantages which will be discussed in the following sections. One of the more notable parts of the FEM common to most of its varieties is how 2nd order PDEs are reduced to a form involving only first derivatives of the input functions. This is sometimes referred to as a bilinear form, which arises from applying integration-by-parts to a second-derivative term in the governing equations. Consider an equation corresponding to a weighting function $\omega_i(\vec{x})$ 'moment' of the Poisson equation:

$$\begin{aligned} \int_{\mathcal{D}^c} \omega(\vec{x})\nabla^2\phi_j(\vec{x})d\vec{x} &= 0 \\ \int_{\mathcal{D}^c} \left[\nabla\cdot(\omega(\vec{x})\nabla\phi(\vec{x})) - (\nabla\omega)\cdot(\nabla\phi)\right]\, d\vec{x} &= 0 \\ \int_{\partial\mathcal{D}^c} \omega(\vec{x})(\hat{n}\cdot\nabla)\phi(\vec{x})\, d\vec{x} - \int_{\mathcal{D}^c} (\nabla\omega)\cdot(\nabla\phi)\, d\vec{x} &= 0. \end{aligned} \tag{2.18}$$

This weak formulation of the equation is central to most DG-FEM implementations, though it is in principle no different mathematically to the original form. The need to only take a single derivative is highly desirable, since a numerical derivative inherently reduces stability of many calculations. Additionally, it allows for the use of purely linear basis functions if desired. Note how the surface term is defined on a cell boundary. This term cannot in principle be localized to a given cell, as it belongs to both simultaneously. Resolving this issue will be discussed further in Section 2.2.2

2.2.1. Tensor Product Extension

The incompatibility with TI and the increased memory use relative to NEM stems from the method by which FEM is extended to multiple dimensions. The standard procedure for doing so is called Tensor Product Extension (TPE). If there are D dimensions, then the 1D basis and weighting are extended into their N-D variants (denoted here with an underbar) as:

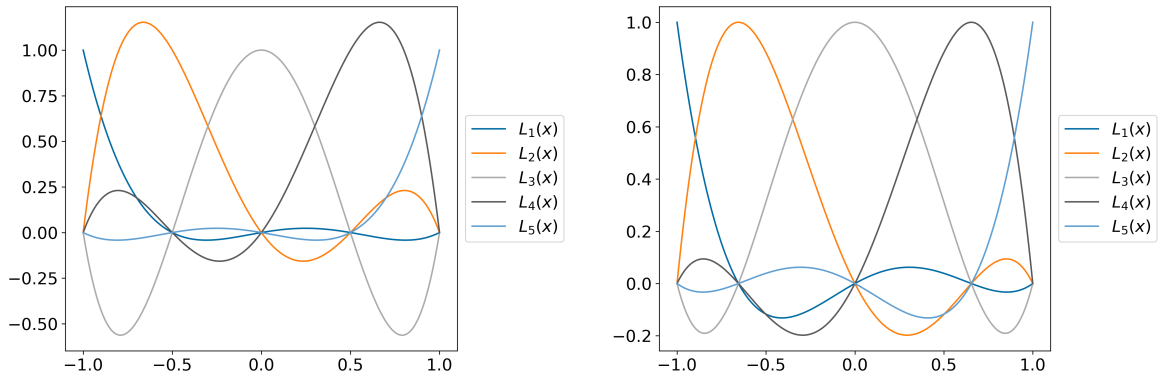
$$\underline{b}_{i_1 \ldots i_D}(\vec{x}) = \prod_{d=1}^{D} b_{i_d}(\hat{e}_d \cdot \vec{x}), \qquad \underline{\omega}_{i_1 \ldots i_D}(\vec{x}) = \prod_{d=1}^{D} \omega_{i_d}(\hat{e}_d \cdot \vec{x}); \tag{2.19}$$

where $\hat{e}_d$ is the d-th dimension's unit vector. Naturally, this raises the number of the unknowns to the power D, which is one reason TPE was not heavily used for multidimensional reactor calculations in [19]. Such an implementation was far too costly for the computers of the time, and because all basis functions depend on all 3 spatial unknowns, transverse integration does not decouple the problem as it does for a Legendre basis.

2.2.2. Discontinuous Galerkin FEM

In the DG-FEM, the regions of the problem have a large degree of independence. Like NEM, basis functions are defined within a single cell, and interact with their neighbors only through the boundary terms in Equation (2.18). Unlike NEM, there are no *explicit* relations enforcing continuity between these representations. The consequence of this is that the solution does not obey any strict form of continuity, hence the 'Discontinuous' identifier. Nonetheless, a correct implementation of the relevant Diffusion/SP_N equations tends to result in very minor discontinuities provided a reasonable input [2, 3].

This method is also fully Galerkin, and also an example of a 'pure' WRM. A common choice of basis is the N-th order Lagrange polynomials, though in principle a wide variety of valid choices exist. The Lagrange polynomials simply tend to make the underlying algebra more simple. These possess the useful properties that each basis function may be identified with a single 'node' within the subdomain, at which this basis function takes on the value 1 while all others are zero. Common choices for these points include equidistant nodes, Chebyshev nodes, or the nodes of various different numeric quadrature rules. Generating the algebraic equations is done by using these points in a numeric quadrature rule to evaluate the integrals resulting from the WRM procedure.



(a) Lagrange-type basis with Newton-Cotes Nodes **(b) Lagrange-type basis with Gauss-Lobatto Nodes**

Figure 2.2: Common basis sets in Finite Element Methods

An important aspect of the DG-FEM is how boundary quantities are defined. Consider Equation (2.18), reproduced below:

$$\int_{\partial\mathcal{D}} \omega(\vec{x})(\hat{n}\cdot\nabla)\phi(\vec{x})\,d\vec{x} - \int_{\mathcal{D}} (\nabla\omega)\cdot(\nabla\phi)\,d\vec{x} = 0. \tag{2.20}$$

The term $(\hat{n}\cdot\nabla)\phi$ evaluated in the integral on the left is commonly referred to in applied mathematics as the numerical flux, though for consistency it will be referred to as a numerical current here. These quantities are defined on the boundary, and thus must be determined by a relation involving cells on both sides. This choice obviously impacts the sparsity pattern of the underlying discretization. As we will see in Section 2.4 it is also a significant factor in the speed of convergence.

2.2.3. Numeric Currents

These currents may include what are known as *Penalty Terms*, in which case they form an Interior Penalty Finite Element Method (IP-FEM). Penalty terms enforce the continuity of flux or current in a specific way or with prescribed weights. These must be determined *a priori* and in a sufficiently general way such that they work for a wide variety of problems.

An important note is that the interfacial currents are not guaranteed to be conserved (in the strong sense) for a general choice of numeric current. Only methods which can be said to be *completely conservative*, to use the terminology of Arnold et al. [2] will produce a solution

that satisfies this conservation. Otherwise, it may only be said to be conserved in a weak sense, in the limit of many basis functions.

To rigorously define these numeric currents, a second-order problem must be broken into a system of first-order differential equations. Consider the 1D single-group diffusion equation in a homogenous medium

$$-D\nabla^2\phi(\vec{x}) + \Sigma_r\phi(\vec{x}) = Q(\vec{x}). \tag{2.21}$$

And use Fick's law to break it up into first-order equations:

$$\nabla\cdot\vec{J}(\vec{x}) + \Sigma_r\phi(\vec{x}) = Q(\vec{x}), \tag{2.22a}$$

$$\vec{J}(\vec{x}) + D\nabla\phi(\vec{x}) = 0. \tag{2.22b}$$

Now, suppose we wish to solve this on 2 adjacent cells, sharing a boundary at x_0 with normal vector $\hat{x}$. We will convert this to a weak formulation by defining two test functions; one scalar $b(\vec{x})$ and one vector $\vec{\beta}(\vec{x})$. These are defined such that both b and each component of $\vec{\beta}(\vec{x})$ belongs to the set of square-integrable functions $L^2(\mathcal{D})$, where $\mathcal{D}^c$ is the domain of the cell these bases belong to. Operating on Equation (2.22a) with $\int_{\mathcal{D}^c} b(\vec{x})(\cdot)\,dV$ and Equation (2.22b) with $\int_{\mathcal{D}^c}\vec{\beta}(\vec{x})\cdot(\cdot)\,dV$:

$$\int_{\mathcal{D}^c} b(\vec{x})\nabla\cdot\vec{J}(\vec{x})\,dV + \Sigma_r\int_{\mathcal{D}^c} b(\vec{x})\phi(\vec{x})\,dV = \int_{\mathcal{D}^c} b(\vec{x})Q(\vec{x})\,dV, \tag{2.23a}$$

$$\int_{\mathcal{D}^c}\vec{\beta}(\vec{x})\cdot\vec{J}(\vec{x})\,dV + D\int_{\mathcal{D}^c}\vec{\beta}(\vec{x})\cdot\nabla\phi(\vec{x})\,dV = 0. \tag{2.23b}$$

The divergence law is then applied to each differential term, such that:

$$\int_{\mathcal{D}^c} b(\vec{x})\nabla\cdot\vec{J}(\vec{x})\,dV = \int_{\partial\mathcal{D}^c} b(\vec{x})\vec{J}(\vec{x})\cdot d\vec{S} - \int_{\mathcal{D}^c}\nabla b(\vec{x})\cdot\vec{J}(\vec{x})\,dV, \tag{2.24a}$$

$$\int_{\mathcal{D}^c}\vec{\beta}(\vec{x})\cdot\nabla\phi(\vec{x})\,dV = \int_{\partial\mathcal{D}^c}\phi(\vec{x})\vec{\beta}(\vec{x})\cdot d\vec{S} - \int_{\mathcal{D}^c}\nabla\cdot\vec{\beta}(\vec{x})\phi(\vec{x})\,dV. \tag{2.24b}$$

The flux and currents $\phi(\vec{x})$ and $\vec{J}(\vec{x})$ on the boundary $\vec{x}\in\partial\mathcal{D}^c$ are then replaced with their

numeric variants; denoted with a hat:

$$-\int_{\mathcal{D}^c} \nabla b(\vec{x}) \cdot \vec{J}(\vec{x})\, dV + \int_{\partial\mathcal{D}^c} b(\vec{x}) \hat{\vec{J}}(\vec{x}) \cdot d\vec{S} + \Sigma_r \int_{\mathcal{D}^c} b(\vec{x}) \phi(\vec{x})\, dV = \int_{\mathcal{D}^c} b(\vec{x}) Q(\vec{x})\, dV, \tag{2.25a}$$

$$-D\int_{\mathcal{D}^c} \nabla \cdot \vec{\beta}(\vec{x}) \phi(\vec{x})\, dV + D\int_{\partial\mathcal{D}^c} \hat{\phi}(\vec{x}) \vec{\beta}(\vec{x}) \cdot d\vec{S} + \int_{\mathcal{D}^c} \vec{\beta}(\vec{x}) \cdot \vec{J}(\vec{x})\, dV = 0. \tag{2.25b}$$

A choice of these numeric currents is required in order to fully define a DG-FEM. A full survey including studies of many different schemes for choosing these currents has been performed by Arnold et al. [3]. We note that if an expression for $\hat{\vec{J}}$ may be derived purely in terms of ϕ, it is trivial to plug back in Fick's law to Equation (2.25a) to obtain a second-order PDE for the flux alone.

Of these schemes, a handful are notable for being fully consistent and stable. To write them compactly, we define some notation. First, let ϕ_h and $\vec{J}_h$ be the approximate fluxes and currents as represented in the chosen basis set. Then, on a boundary between elements, let the two sides be represented with a $^+$ and $^-$ superscript; each with their own outward-pointing unit normals $\hat{n}$. The average and jump operators may then be defined:

$$\{f\} = \frac{1}{2}\left(f^+ + f^-\right), \qquad [[f]] = f^+\hat{n}^+ + f^-\hat{n}^-, \qquad [[\vec{g}]] = \vec{g}^+ \cdot \hat{n}^+ + \vec{g}^- \cdot \hat{n}^-. \tag{2.26}$$

These include the classic IP-FEM method, a method presented by Brezzi et al. [2], and a method presented by Bassi et al. [7]. Here, the parameter β and the functions α_j and α_r

	$\hat{\phi}$	$\hat{\vec{J}}$
IP	$\{\phi_h\}$	$\{-D\nabla_h\phi_h\} - \alpha_j([[\phi_h]])$
Brezzi et al.	$\{\phi_h\}$	$\{\vec{J}_h\} - \alpha_r([[\phi_h]])$
Bassi et al.	$\{\phi_h\}$	$\{-D\nabla_h\phi_h\} - \alpha_r([[\phi_h]])$

Table 2.1: Selected numeric currents of unified DG-FEM analysis [3]

are effectively user-defined constants that must be known *a priori*. Arnold et al. go into some detail on how precisely to choose these values to achieve stability and consistency, but ultimately they depend in some way on user-specified values and the grid spacing.

Of particular note in Table 2.1 is the IP-FEM method which satisfies the constraint of an expression for $\hat{\vec{J}}$ defined solely in terms of the flux ϕ_h. It is therefore this method which is analyzed in Section 2.4. However, care must be taken with mixed boundary conditions;

a case difficult to find in the literature. The challenge is arriving at an expression for the penalty terms which does not cause spurious penalties at the boundaries of the problem, where they can lead to detrimental effects on convergence. For IP-FEM, the function α_j is known to have the form:

$$\alpha_j(y) = \eta_e y, \tag{2.27}$$

per Arnold et al. [3]; where the quantity η_e is a positive real number which may depend on the given edge e. Requiring these spurious penalty terms to vanish at the edges means that the evaluation of $\hat{\vec{J}}$ at the boundary must be zero. Writing this out for the boundary of a problem where the + side is within the domain, and the − half is absent:

$$\hat{\vec{J}}\Big|_{\partial\mathcal{D}^c} = -D\{\nabla_h \phi_h\} - \eta_e[[\phi_h]] = -\frac{D}{2}\nabla_h \phi_h^+ - \eta_e \phi_h^+ \hat{n} \tag{2.28}$$

$$= -\frac{D}{2}(\hat{n}\cdot\nabla_h)\phi_h^+ - \eta_e \phi_h^+. \tag{2.29}$$

We may force this to equal zero by asserting the boundary condition on ϕ_h^+. In the case of the zero-reentrant flux condition, this requires setting $\eta_e = \frac{1}{4}$.

2.2.4. Continuous FEM

The CFEM is an implementation of the FEM which is fundamentally quite different from the DG-FEM seen in most applications today. Rather than basis functions being associated with the interior of each node, basis functions are instead associated with the intersection points of the mesh, which is often unstructured. Basis functions are no longer uniquely identified within a subdomain, but rather are defined on the problem domain as a whole (though in practice, they are defined only on a small, local cluster of nodes). A requirement of the implementation is that on the boundary of any given basis function's support, it drops to zero.

Several complexities in this approach have led to DG-FEM becoming much more popular for general differential equations. One issue is that increasing the basis function order is often very difficult, as it involves adding extra basis functions on edges and interiors, which interact with each other in potentially complex ways. In particular, forms of parallelism based on domain decomposition will see their communication cost (and implementation

complexity) increase significantly.

For these reasons many implementations use simple piecewise linear bases, which need only be defined at the aforementioned intersection of subdomains. However, this is a big problem in applications with strict conservation laws, since this does not provide enough degrees of freedom to enforce all relevant conservation laws within each cell. While *h*-refinement addresses this to some extent, it still leaves an approximate solution which does not satisfy conservation laws in any strong sense.

2.3. The Legendre-Gauss-Lobatto Method

The Legendre Gauss-Lobatto (LGL) method is a new type of discretization arising from WRM in much the same way as NEM and FEM do. This method has been developed with a focus on maintaining the specific properties of accuracy we care about in NEM, including a strict enforcement of conservation laws, while inheriting the flexibility of FEM methods. For reasons which will be discussed in the next chapter, this method in particular is very amenable to a variety of hierarchical fast solver techniques.

The 'base' of this method is essentially a DG-FEM procedure with collapsed basis functions at cell boundaries, such that the end result reduces to a CFEM in a 1D geometry. Consider, in 1D, the Discontinuous-Galerkin equation for cell c:

$$D_c \int_{\mathcal{D}^c} \frac{db_i}{dx} \sum_{j=1}^{N} \phi_j \frac{db_j}{dx}\, dx - D_c \left[b_i(x) \sum_{j=1}^{N} \phi_j \frac{db_j}{dx} \right]_{x=\partial\mathcal{D}^{c-}}^{\partial\mathcal{D}^{c+}} + \Sigma_t \int_{\mathcal{D}^c} b_i(x) \sum_{j=1}^{N} \phi_j b_j(x)\, dx = \int_{\mathcal{D}^c} b_i(x) Q(x)\, dx, \tag{2.30}$$

where $\mathcal{D}^c$ is the subdomain of interest, and also happens to be the support of the basis function b_i; since these bases are defined on a cell-by-cell basis. The basis functions above form a Lagrange basis of the form given in Equation (2.31), with nodes x_k defined as the Gauss-Lobatto numeric quadrature points:

$$L_j(x) = \prod_{k \neq j} \frac{x - x_k}{x_j - x_k}. \tag{2.31}$$

An important property of this Lagrange basis is that it forms a *partition of unity*. This means that the constant function is always in the span of a Lagrange basis, which is essential in maintaining neutron conservation. Additionally, this basis has been constructed to allow for quadrature-based self-lumping [44]. This will later enable fast numerical quadrature evaluations in Chapter 4

From here, we condense neighboring basis functions between cells. This is done to force the approximate solution to be continuous; effectively making this into a CFEM scheme. In 1D, the basis then changes to resemble that in Figure 2.3, which notably has a first-derivative discontinuity at the boundary. This might seem unwanted, but it is in fact, expected and desirable; without it, the basis could not hope to accurately capture the 'kink' in the flux which appears at material interfaces.

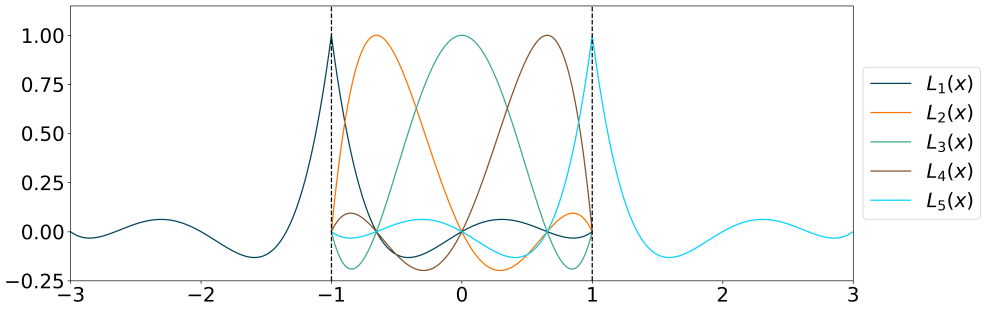


Figure 2.3: LGL Basis Functions

This has the somewhat startling result that we may drop the leakage term, and still enforce current continuity. To prove this, consider Equation (2.30) for $i \neq 1, i \neq N$. In this case, we will always have $b_i(\partial\mathcal{D}^+) = b_i(\partial\mathcal{D}^-) = 0$, so the streaming term may be dropped with no consequence. However, consider now $i = N$ for cell L. This basis is identical to the basis corresponding to $i = 1$ of the neighboring cell to the right, which we will call R. In the WRM approach, the bounds of the integral fundamentally span the whole problem; it is only because we have decomposed it into subdomains that we are able to simplify this to distinct regions. Because the basis now spans multiple regions, the domain of the integral must

likewise expand; and so we end up with:

$$\int_{\mathcal{D}^L \cup \mathcal{D}^R} \underline{D}(x) \frac{d\underline{b}}{dx} \frac{d\underline{\phi}}{dx}\, dx - D_L \left[b_N(x) \sum_{j=1}^{N} \phi_j \frac{db_j}{dx} \right]_{x=\partial\mathcal{D}^{L-}}^{\partial\mathcal{D}^{L+}} - D_R \left[b_1(x) \sum_{j=1}^{N} \phi_j \frac{db_j}{dx} \right]_{x=\partial\mathcal{D}^{R-}}^{\partial\mathcal{D}^{R+}} + \int_{\mathcal{D}^L \cup \mathcal{D}^R} \underline{\Sigma}_t(x) \underline{b}(x) \underline{\phi}(x)\, dx = \int_{\mathcal{D}^L \cup \mathcal{D}^R} \underline{b}(x) \underline{Q}(x)\, dx. \tag{2.32}$$

Note the underbar notation, which denotes a piecewise quantity for compactness; returning the relevant material property depending on whether x is located in subdomain L or R. The basis function $\underline{b}$ is piecewise as well, returning $b_N(x)$ for $x \in \mathcal{D}^L$ and $b_1(x)$ for $x \in \mathcal{D}^R$. Here we use the fact that b_1 and b_N are zero at the boundary the two cells do not share, allowing that term to be eliminated. At the other boundary, they are both necessarily 1, since the Lagrange basis is a partition of unity. Then:

$$\int_{\mathcal{D}^L \cup \mathcal{D}^R} \underline{D}(x) \frac{db}{dx} \frac{d\underline{\phi}}{dx}\, dx - \left[D_L \frac{d\phi_L}{dx}\Big|_{x=\partial\mathcal{D}^{L+}} - D_R \frac{d\phi_R}{dx}\Big|_{x=\partial\mathcal{D}^{R-}} \right] + \int_{\mathcal{D}^L \cup \mathcal{D}^R} \underline{\Sigma}_t(x) \underline{b}(x) \underline{\phi}(x)\, dx = \int_{\mathcal{D}^L \cup \mathcal{D}^R} \underline{b}(x) Q(x)\, dx. \tag{2.33}$$

The bracketed term is precisely the application of the continuity of net current which must be satisfied to obey conservation laws. We simply drop this term, and the resulting approximate solution is guaranteed to be both continuous and consistent with the first-derivative jump condition of the typical material interface condition for Diffusion or Brantley-Larsen SP_N. As has been stated elsewhere in this work, this is equivalent to enforcing conservation of partial currents.

2.4. Convergence Analysis

To analyze the convergence of each of these methods, reference implementations were created for a 1D, 2-group heterogeneous diffusion problem. The purpose of this was to put each discretization method on equal footing, in order to meaningfully compare errors produced in each approach. Since this type of problem has a known analytic solution, error comparisons can be made directly without any additional complications regarding the existence or behavior of the true solution.

A depiction of this case is shown in Section 2.4 with material information included. Analytic solutions to this problem with vacuum boundary conditions are presented in Appendix A.

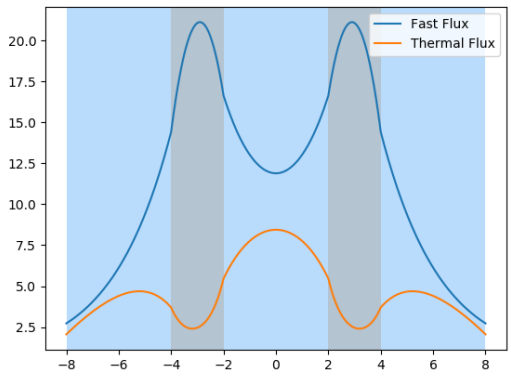


Figure 2.4: Analytic solution to a simple 1D/2G Diffusion problem

Fuel Region:

$$\begin{aligned} \Sigma_t^f &= 0.5 & \Sigma_{s0}^f &= 0.41 & \Sigma_{s1}^f &= 0.25\Sigma_{s0}^f \\ \Sigma_t^t &= 1.0 & \Sigma_{s0}^t &= 0.2 & \Sigma_{s1}^t &= 0.25\Sigma_{s0}^t \\ \Sigma_{s2\leftarrow 1} &= 0.04 & Q^f &= 10 & Q^t &= 0 \end{aligned}$$

Moderator Region:

$$\begin{aligned} \Sigma_t^f &= 0.25 & \Sigma_{s0}^f &= 0.11 & \Sigma_{s1}^f &= 0.25\Sigma_{s0}^f \\ \Sigma_t^t &= 0.5 & \Sigma_{s0}^t &= 0.5 & \Sigma_{s1}^t &= 0.25\Sigma_{s0}^t \\ \Sigma_{s2\leftarrow 1} &= 0.11 & Q^f &= 0 & Q^t &= 0 \end{aligned}$$

Since the NEM/DG-FEM both technically permit using different basis functions, both the Lagrange and Legendre Polynomials were tested. The different discretizations were formed explicitly as matrices and solved exactly, to minimize the effect of solver error on the discretization results. Convergence was analyzed by examining the groupwise sum of the functional L^2 norm over the entire problem domain, which entails computing an integral of the error over the entire space. This is done using an extremely high-accuracy adaptive integration procedure; not a low-resolution pointwise estimation. This approach was chosen to minimize the additional discretization error.

Results are shown in Figure 2.5, which consists of generally expected results. Most obviously, we have the semi-analytic methods UNM and SANM behave exactly as expected, with UNM hovering near the square root of machine precision for any mesh size and SANM dropping quickly due to the decreasing error in group-to-group scattering. Note that the rise upwards at small grid sizes is largely due to poor conditioning in the implementation, which could be tweaked to work better in the small-cell limit. However, regardless of the implementation details, the small-cell limit of SANM does have stability issues, which can only be partially addressed without reformulating the method.

The remaining solver types achieve the expected maximum convergence, with the excep-

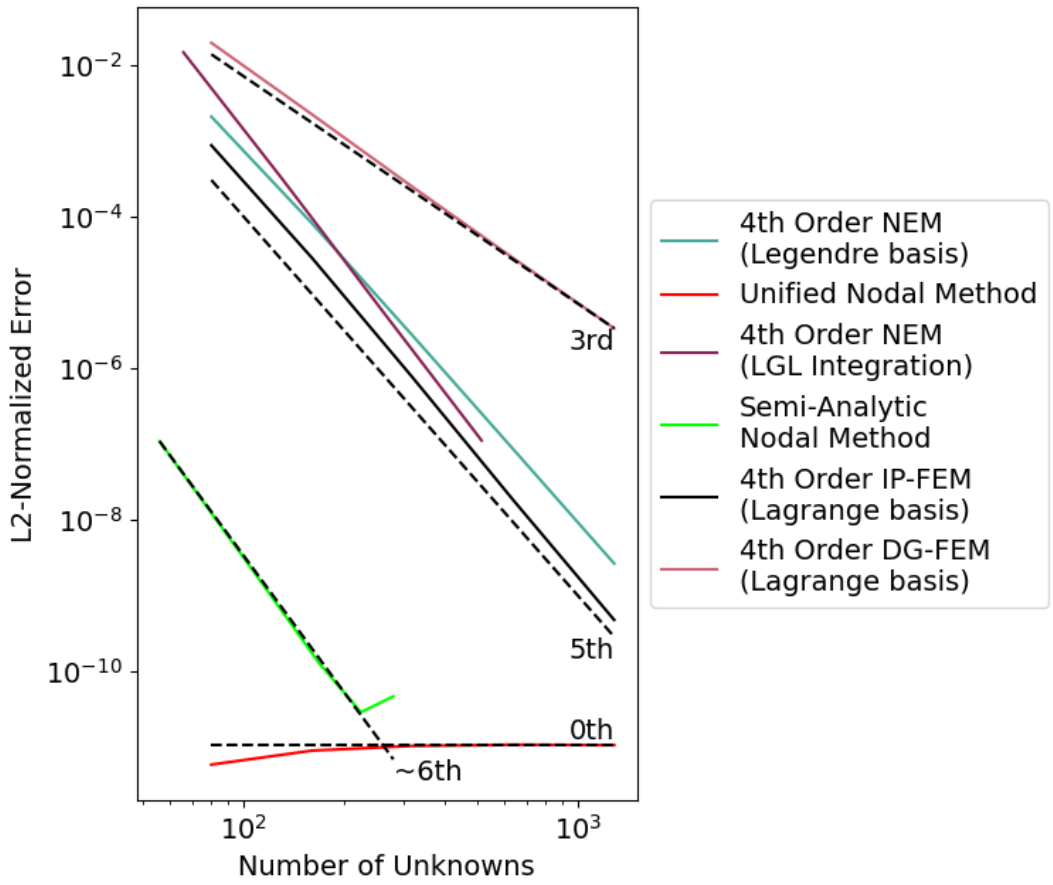


Figure 2.5: h-convergence of 1D 2-group Test Problem

tion of the one labeled DG-FEM. There is some super-convergence observed for the LGL case. This is sometimes observed for CFEMs, so it is not surprising to see it appear here.

The subpar convergence of "DG-FEM" is caused by the choice of numeric current mentioned in Section 2.2.2. This method shown uses a form of current construction vaguely reminiscent of upwinding. Effectively, the numeric currents are derived from the expression for conservation of the partial current into the node. For example, the numeric current

$j^c_{c+1/2}$ representing the numeric current for cell c at boundary $c + 1/2$ would be given by:

$$\frac{1}{4}\phi^c(x_{c+1/2}) + \frac{D^c}{2}\frac{d\phi^c}{dx}\bigg|_{x_{c+1/2}} = \frac{1}{4}\phi^{c+1}(x_{c+1/2}) + \frac{1}{2}j^c_{c+1/2},$$
$$j^c_{c+1/2} = \frac{1}{2}\left(\phi^c(x_{c+1/2}) - \phi^{c+1}(x_{c+1/2})\right) + D^c\frac{d\phi^c}{dx}\bigg|_{x_{c+1/2}}. \tag{2.34}$$

This approach is, to again use the terminology of Arnold et. al. *fully conservative*, but does not obey the full set of properties required for an optimal FEM method.

2.4.1. Theoretical Convergence Limits

For any WRM-based PDE solver, the fundamental limit of convergence for a polynomial basis is a result obtained by the Bramble-Hilbert Lemma, which states that:

$$\inf_{v\in\mathcal{P}^{m-1}}\|u(x) - v(x)\|_{L^2} \leq C(m)h^m\|u^{(m)}\|_{L^2}; \tag{2.35}$$

i.e. a polynomial v of degree $m-1$ fundamentally must have a minimum interpolation error proportional to the grid spacing h^m. So the best possible convergence we can hope for is 5th order, which is achieved by the NEM, IP-FEM, and LGL methods. Note that the Semi-Analytic methods are not subject to this maximum because their basis functions are non-polynomial in nature.

Chapter 3

Fast Solvers and Hierarchical Methods

"Fast Solvers" undoubtedly sounds like a particularly vague and optimistic way to label any form of algorithmic solver; after all, what is or is not considered 'fast' can change exceedingly rapidly both with theory and physical computing hardware. However, it does take on a concrete meaning in the applied mathematics community, and essentially refers to a class of algorithms which solve a given problem with optimal- or near-optimal scaling with respect to the number of unknowns. Naturally then, 'optimal' refers to $O(N)$ algorithms, where N is the number of unknowns. While definitions of this 'near-optimal' phrasing vary, a common definition is 'less than 1 polynomial order away from optimal', in limit of large-N. In this sense, $O(N \log N)$ would be considered near-optimal and thus 'Fast'; though it is worth impressing that this is not a universal definition.

This relaxed definition categorizes many commonly-used algorithms as 'Fast'; the Fast Fourier Transform, for example, being among the most prominent though it does not achieve linear performance. Another relevant example is the Fast Multipole Method (FMM). Originally developed as a way to accelerate force calculations on collections of charges (Figure 3.1), the FMM sacrifices a small amount of accuracy in order to rapidly accelerate certain calculations; to the point where they can take $O(N \log N)$ time in some cases. This loss in accuracy may be customizable down to the error already incurred by working in finite-precision arithmetic. Such methods are sometimes called Quasi-Direct.

In this chapter, some of the theory behind developing these types of algorithms will be covered in Section 3.1. Additionally, the concept of Poincaré-Steklov operators will be introduced in Section 3.1.1 to facilitate this theory as it pertains to the subject of interest; elliptic PDEs. This section will also discuss notational conventions which will be used through-

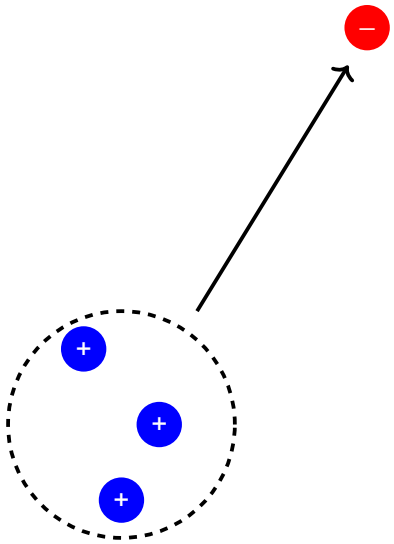


Figure 3.1: Charge Lumping in the FMM

out this dissertation. These topics will then be used to reproduce a derivation of the HPS method. While the derivation put forth in Section 3.2 is not original work, the solver has been adapted to be fit for purpose in low-order transport problems.

3.1. Hierarchical Solver Theory

The blueprint for creating fast algorithms often proceeds somewhat like this:

1. Recursively partition the unknowns into a finite set of groups until the problem is trivial (or at least tractable); creating a 'tree' structure.
2. Perform a trivial calculation upon every leaf in this tree.
3. Iterate on the couplings that connect nodes with their parents until the solution is satisfactory.

Depending on the problem, either step 2 or step 3 may dominate the work required; an explanation of the consequences of this may be seen in Section 7.5. This has the potential to perform very well; provided that there is a way to decompose the problem that results in step 3 remaining tractable.

The question then arises; for elliptic PDEs, which couplings are the strongest? The answer to this is the *spatial* coupling. Qualitatively, this may be observed by considering the discretization of the Laplacian operator ∇^2. The 5-point Laplacian stencil for uniform grids is

$$\nabla_h^2 u(x,y) = \frac{1}{h^2}[u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1}] + O(h^2); \qquad u_{i,j} = u(x_i, y_j), \tag{3.1}$$

where h is small, and therefore nearby points $u_{i,j}$ and $u_{i\pm1,j\pm1}$ are tightly coupled. In the limit of small grid spacing, the magnitude of the coupling in this differential operator will overwhelm anything else; even the coupling between angular moments in the SP_N or GSP_N equations.

Another way to demonstrate this is via Green's functions. The Green's function of a differential operator defines the response of the solution at x to a delta source at x'. Consider the 3D Laplacian problem:

$$G(\vec{x}, \vec{x}') = -\frac{1}{4\pi}\|\vec{x} - \vec{x}'\|^{-1}, \tag{3.2}$$

which demonstrates that the coupling in this example falls off as the inverse of distance.

Whereas in the Fast Fourier Transform algorithm, the weakest couplings are 'cut' by even-odd parity, we must find a way to 'cut' the far-away spatial couplings of an elliptic PDE. This is achieved by splitting the domains spatially into many independent cells. If done properly, such a decomposition will reduce the average distance between two points in any subdomain; effectively severing the weakest coupling while leaving the strongest intact.

3.1.1. Poincaré–Steklov Operators

A PDE may be solved independently on small pieces such as these, but they must somehow be stitched back together into a cohesive whole. This difficulty is a consequence of the well-known challenge in solving these types of problems, where information takes a long time to diffuse through the tight couplings between points. While it is most obvious in finite difference-style discretizations, it remains a problem in others. To handle this, we introduce a new tool known as the Poincaré–Steklov operator.

This operator maps the Dirichlet boundary condition of an elliptic PDE to the Neumann boundary condition which yields the same solution. As either condition is known to uniquely

determine a solution, this mapping and its inverse are well-defined. Used in conjunction with an interface condition, this operator may be used to effectively solve for flux values at the boundaries between cells, even across material discontinuities. These operators have been used in domain decomposition methods for both homogeneous problems [1] (in which the boundaries have no physical meaning) as well as heterogeneous problems [55] (in which the coefficients of the PDE change abruptly across these boundaries). The flexibility in which this approach enables domain decomposition motivates the creation of a hierarchical data structure such that this boundary may be separated from the two cells connected to it; a formal description of which follows in the next section.

The remainder of this chapter will concern existing fast algorithms for dealing with these topics. While the algorithms themselves are not novel, their application and the adaptations required in applying these to SP_N and GSP_N is original work.

3.1.2. Geometric Decomposition

Consider for simplicity a square problem domain subdivided in the x and y directions such that the number of subdivisions N is a power of two. This yields N^2 overlapping subdomains, or nodes, which will form the leaves of a binary tree. Within each of these leaves, a representation using the LGL basis functions is adopted, which utilizes a set of unknowns which may be thought of as being located at the Gauss-Lobatto points in the cell.

This representation spans the space of polynomials belonging to the 2D-TPE of polynomials up to order p within each cell. However, because LGL basis functions are being used, this means that the unknowns located on the cell boundaries are shared between the two cells on that boundary. To draw a distinction between these two types of unknowns, they are broken into two sets; points on the boundary of these fine cells belong to a boundary set $\mathcal{I}_b$; while those on the interior belong to an interior set $\mathcal{I}_i$.

To effectively solve for each unknown in the problem, we consider the problem domain as a whole. We may create a root node, with its own boundary and interior sets. This is done by extracting a set of unknowns from the problem domain, such that the remaining unknowns form two disjoint sets, as seen in Figure 3.2. This extracted set should not contain any points interior to the leaves.

In Figure 3.2, the interior and boundary sets are colored blue and red, respectively; a convention that will be used for the remainder of this document. Points in gray exist, but

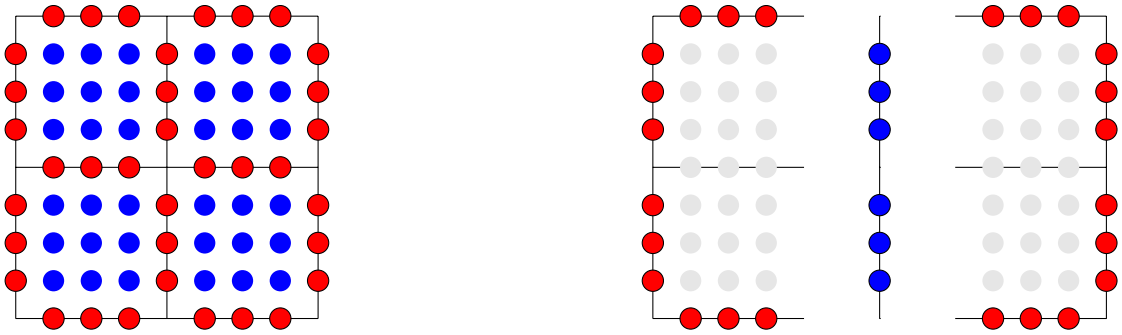


Figure 3.2: Depiction of a paritioning of the root node.

are not included in either of these named sets. On the left, these are colored with respect to the interior and boundary of the leaves. On the right, the definition of the interior/boundary of the root is shown; the extracted points become the interior, while the boundary of the root (which is the boundary of the problem itself) becomes the exterior.

This partitioning is carried out recursively, with the scheme in full resembling a binary tree as in Figure 3.3. An example root node γ, branch node β and leaf node α are depicted in

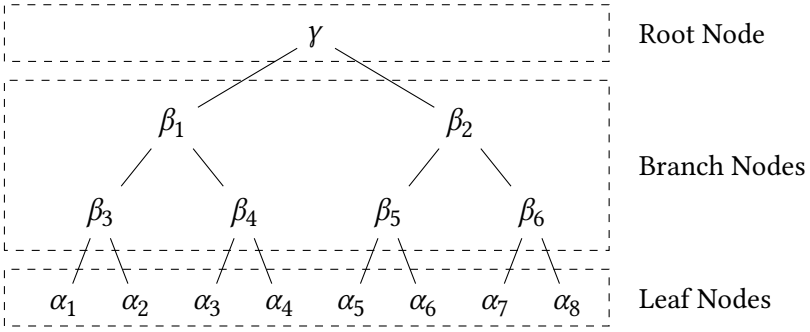


Figure 3.3: Domain Tree Structure

Figure 3.4. Each node in the tree of Figure 3.3 has its own distinct interior and boundary sets. When necessary, one may distinguish between the index sets for each of these nodes with a superscript, e.g. $\mathcal{I}_b^\alpha$ and $\mathcal{I}_b^\beta$. Note that Figure 3.4 also shows a new set of points colored in black on the leaf node α. These points belong to a third index space $\mathcal{I}_c$ which is isolated from the others for reasons which will be discussed further in Section 4.7.

The reason for this binary partition is motivated by the work that is involved in coupling the solutions on these regions together. While a quad-tree may seem more intuitive for a 2D problem, it will become clear in Section 3.2.2 and Section 7.5.1 that the work involved in

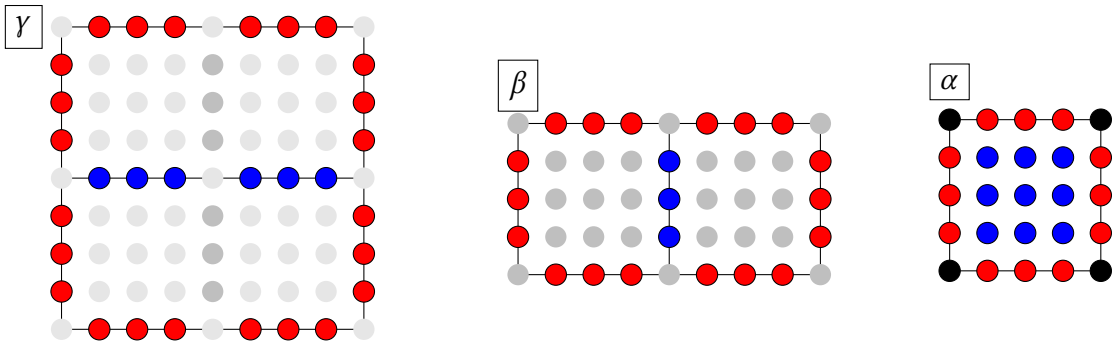


Figure 3.4: Depiction of Index Spaces for Leaf and Branch Nodes.

coupling these regions is proportional to the size of the interface between them. In fact, this proportionality is super-linear, and so using a quad-tree (doubling the size of the interface) is most likely not advisable.

3.1.3. Operator Partitions and Index Spaces

A significant motivation for defining the index spaces $\mathcal{I}_b$, $\mathcal{I}_i$, and $\mathcal{I}_c$ is how they allow convenient shorthand to refer to the values of a vector on a subset of points. Or, in addition, the restriction of an operator to a subset of its full row/column space. For example, suppose that a problem domain consists of many leaf nodes, and we wish to write the coefficients of the flux ϕ which correspond to the interior of a 2D cell τ; expressed as $\phi(\mathcal{I}_i^\tau)$.

Alternatively, suppose we have an equation $\underline{\underline{A}}\underline{\phi} = \underline{Q}$, where $\underline{\underline{A}} : \mathcal{I}_{\text{all}} \mapsto \mathcal{I}_{\text{all}}$, where $\mathcal{I}_{\text{all}} = \mathcal{I}_b \cup \mathcal{I}_i \cup \mathcal{I}_c$. These index spaces already make the definition of such an operator much easier to write, but they further allow compact representations of partitions of A such that:

$$\begin{bmatrix} A(\mathcal{I}_b;\mathcal{I}_b) & A(\mathcal{I}_b;\mathcal{I}_i) & A(\mathcal{I}_b;\mathcal{I}_c) \\ A(\mathcal{I}_i;\mathcal{I}_b) & A(\mathcal{I}_i;\mathcal{I}_i) & A(\mathcal{I}_i;\mathcal{I}_c) \\ A(\mathcal{I}_c;\mathcal{I}_b) & A(\mathcal{I}_c;\mathcal{I}_i) & A(\mathcal{I}_c;\mathcal{I}_c) \end{bmatrix} \begin{bmatrix} \phi(\mathcal{I}_b) \\ \phi(\mathcal{I}_i) \\ \phi(\mathcal{I}_c) \end{bmatrix} = \begin{bmatrix} Q(\mathcal{I}_b) \\ Q(\mathcal{I}_i) \\ Q(\mathcal{I}_c) \end{bmatrix}. \tag{3.3}$$

This notation will be used frequently throughout the rest of this dissertation, and will often be shortened further like so: $A(\mathcal{I}_i;\mathcal{I}_b) = A_{ib}$, or $\phi(\mathcal{I}_c^\tau) = \phi_c^\tau$. The operator $\mathcal{N}$ will also make frequent appearances as the operator which defines the current-like conserved quantities of the elliptic PDE discretized by A. As these quantities are defined along the boundary of the problem, it satisfies the mapping $\mathcal{N} : \mathcal{I}_{\text{all}} \mapsto \mathcal{I}_b$. For SP_3, the definitions of these quantities

are contained in Equation (1.22) (Diffusion will use the same but for the zero-th moment only). GSP_3 will instead use the current definitions of Equation (1.29).

3.2. Hierarchical Poincaré–Steklov Solver

The Hierarchical Poincaré-Steklov Solver is a type of solver for elliptic PDEs discussed in length in recent papers [27, 45] and a textbook [46] written by Professor Per-Gunnar Martinsson from The University of Texas (Austin). A central theme, in Martinsson's own words is that "only small amounts of information survive across long distances [which] can be used to build solvers for elliptic PDEs that are fast, robust, and highly accurate".

Exploiting this information sparsity is critical to achieving the fast performance of the method.

3.2.1. Summary of Solution Procedure

The solution procedure contains two steps; the build stage and the solve stage. The method presupposes the existence of the treelike structure of subdomains discussed in Section 3.1.2. However, this tree may be generated simultaneously with the build stage if a method of decomposition is prescribed.

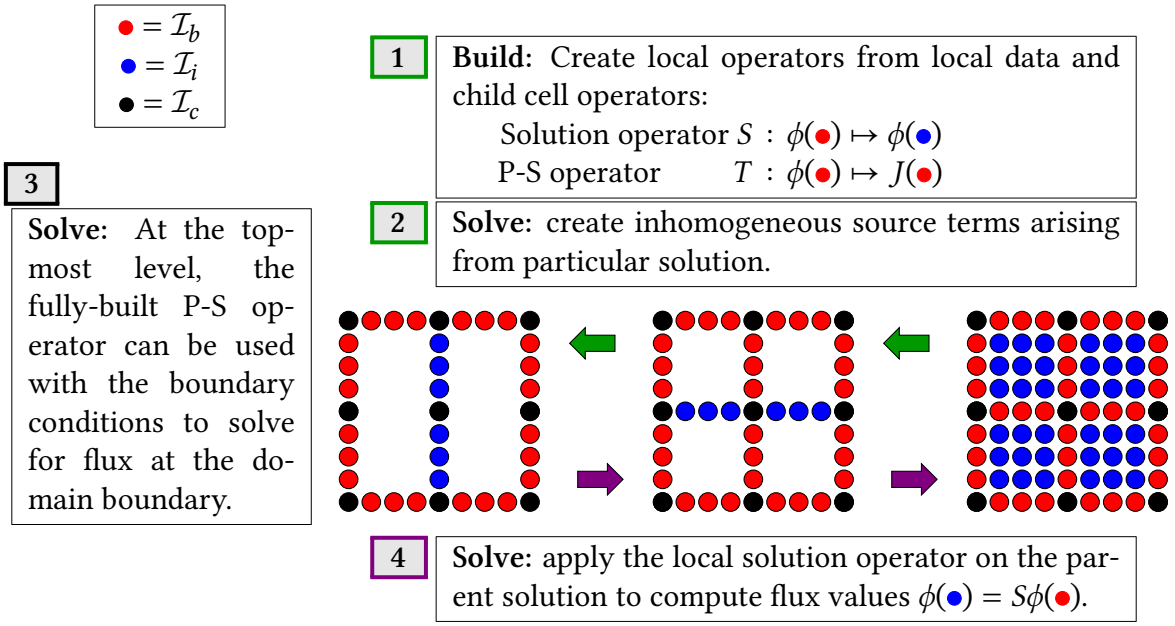


Figure 3.5: HPS Solver Algorithm for 2D Grid

The build stage decomposes the problem matrix into an *Interpolative Decomposition*, which may be used in the solve stage to solve the given matrix for any number of different source terms. Essentially, the solve stage is an efficient application of the global differential operator's inverse. This is highly advantageous for things like power iteration or group-sweeping that require iteration, with a changing source term. While there are undoubtedly concerns of the memory required to store this decomposition, there are effective ways to do so owing to the way this decomposition is structured. Namely, it is composed of a hierarchical structure of highly compressible operators; allowing it to be represented with relatively low memory overhead in such a way that the most important couplings are preserved. These types of storage methods are discussed further in Section 7.3. As for the algorithm, the details of the build and solve stages are discussed in Sections 3.2.2 and 3.2.3, respectively. How this is embedded into a larger solver for fixed-source or eigenvalue calculations is covered in Section 4.8.

3.2.2. Build Stage

The algorithm begins, as discussed in Section 3.1, by considering how to solve the PDE restricted to an isolated leaf node. This is Step 1 depicted in Figure 3.5. If the local solution to the PDE ϕ^τ is separated into a homogeneous ϕ^H and particular ϕ^P solution, a pair of equations is obtained by restricting Equation (3.3) to the interior set:

$$A^\tau_{ii}\phi^H_i + A^\tau_{ib}\phi^H_b + A^\tau_{ic}\phi^H_c = 0 \qquad \text{and} \qquad A^\tau_{ii}\phi^P_i = Q^\tau_i, \tag{3.4}$$

since ϕ^P_b and ϕ^P_c are zero by the definition of a particular solution. The solution operator S for a given cell τ (written S^τ) is given by solving the homogeneous equation for ϕ_i in terms of ϕ_b;

$$\phi^H_i = -(A^\tau_{ii})^{-1}A^\tau_{ib}\phi^H_b - (A^\tau_{ii})^{-1}A^\tau_{ic}\phi^H_c. \tag{3.5}$$

$$\text{Define:} \qquad S^\tau = -(A^\tau_{ii})^{-1}A^\tau_{ib} \qquad \text{and} \qquad R^\tau = -(A^\tau_{ii})^{-1}A^\tau_{ic}\ ; \tag{3.6}$$

$$\phi^H_i = S^\tau\phi^H_b + R^\tau\phi^H_c. \tag{3.7}$$

The other operator needed is the Poincaré-Steklov operator T. The previous expression for S is inserted into the definition of the current $J = \mathcal{N}\phi$ as follows, noting that the contribution

due to the induced current of the particular solution must also be included:

$$\mathcal{N}^\tau \begin{bmatrix} \phi_b^H \\ \phi_i^H + \phi_i^P \\ \phi_c^H \end{bmatrix} = \mathcal{N}^\tau \begin{bmatrix} I \\ S^\tau \\ 0 \end{bmatrix} \phi_b^H + \mathcal{N}^\tau \begin{bmatrix} 0 \\ R^\tau \\ I \end{bmatrix} \phi_c^H + \mathcal{N}^\tau \begin{bmatrix} 0 \\ (A_{ii}^\tau)^{-1} Q_i^\tau \\ 0 \end{bmatrix}, \tag{3.8}$$

and proceed to define:

$$T^\tau = \mathcal{N}^\tau(\mathcal{I}_b; \mathcal{I}_b \cup \mathcal{I}_i) \begin{bmatrix} I \\ S^\tau \end{bmatrix} \quad G^\tau = \mathcal{N}^\tau(\mathcal{I}_b; \mathcal{I}_i \cup \mathcal{I}_c) \begin{bmatrix} R^\tau \\ I \end{bmatrix} \quad H^\tau = \mathcal{N}^\tau(\mathcal{I}_b; \mathcal{I}_i)(A_{ii}^\tau)^{-1} Q_i^\tau. \tag{3.9}$$

Therefore,

$$J = T^\tau \phi_b^H + G^\tau \phi_c^H + H^\tau Q_i^\tau. \tag{3.10}$$

In this scheme, the corner points are treated as external to the HPS solver itself, to be resolved by a corner point balance; discussed further in Section 4.7. The operator G^τ is necessary to incorporate the effects these corner points have on interfacial currents, and R^τ incorporates the effects on the interior. H^τ incorporates the effects due to the PDE's particular solution. These operators are computed and stored for every leaf node in the problem, and map between the following index spaces:

$$S^\tau : \mathcal{I}_b^\tau \mapsto \mathcal{I}_i^\tau, \quad T^\tau : \mathcal{I}_b^\tau \mapsto \mathcal{I}_b^\tau, \quad R^\tau : \mathcal{I}_c^\tau \mapsto \mathcal{I}_i^\tau, \quad G^\tau : \mathcal{I}_c^\tau \mapsto \mathcal{I}_b^\tau, \quad H^\tau : \mathcal{I}_i^\tau \mapsto \mathcal{I}_b^\tau. \tag{3.11}$$

For reference, the size of these index spaces (assuming a quadrilateral grid and basis functions of order p) for the leaves are:

$$\text{size}\{\mathcal{I}_b^\tau\} = 4q(p-1), \qquad \text{size}\{\mathcal{I}_i^\tau\} = q(p-1)^2, \qquad \text{size}\{\mathcal{I}_c^\tau\} = 4q, \tag{3.12}$$

where q is the number of angular moments; in diffusion, $q = 1$, while for SP_N or GSP_N $q = N$. For the 4th order basis on quadrilaterals, and assuming the SP_3 equations, this gives sizes:

$$S^\tau : 18 \times 24, \qquad T^\tau : 24 \times 24, \qquad R^\tau : 18 \times 8, \qquad G^\tau : 24 \times 8, \qquad H^\tau : 24 \times 18, \tag{3.13}$$

for the leaves. The sizes for branches vary; but the branches do not have an associated G^τ or R^τ operators, since they do not need to carry forward values on the corners.

For a more general formulation, one may define a quadrature rule on a given 2D element, and count up the number of quadrature points on the boundary edges N_f, interior N_i, and corners N_v.

$$\text{size}\{\mathcal{I}_b^\tau\} = qN_f, \qquad \text{size}\{\mathcal{I}_i^\tau\} = qN_i, \qquad \text{size}\{\mathcal{I}_c^\tau\} = qN_v. \tag{3.14}$$

One may avoid storing these operators *on the leaves* if one is willing to recompute them every time a solution is needed. This may be advantageous for extremely large problems, as it will reduce total memory requirements drastically. However, this makes the solution procedure somewhat slower (though it does not affect the scaling of the method).

The next step of the build stage is to compute these local operators for the branch nodes, which represent the merging of two children. Consider Figure 3.6, describing how the index spaces of two merging boxes, denoted α and β, overlap. While $\mathcal{I}_i^\alpha$ and $\mathcal{I}_i^\beta$ are disjoint, the boundary and corner sets are not. The intersection of these two sets becomes the new interior, while the union of the two boundaries excluding this overlap forms the new exterior.

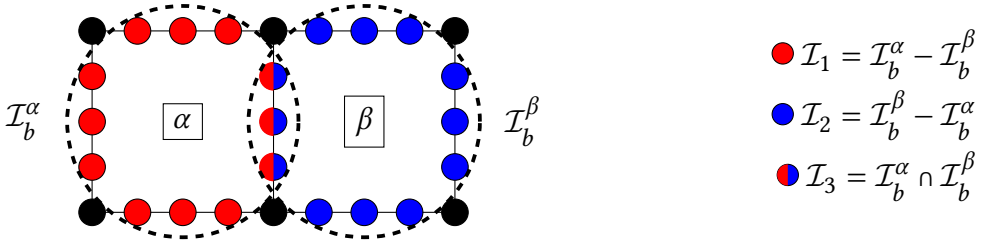


Figure 3.6: Index Spaces in a Merge Operation

All operators for the cell τ with children α and β may be derived by requiring conservation laws to hold on the interior index space and solving for ϕ_3. From the definition of the subdomains' Poincaré-Steklov operators, we have the two matrix equations

$$\begin{bmatrix} T_{11}^\alpha & T_{13}^\alpha \\ T_{31}^\alpha & T_{33}^\alpha \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} J_1 \\ J_3 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} T_{22}^\beta & T_{23}^\beta \\ T_{32}^\beta & T_{33}^\beta \end{bmatrix} \begin{bmatrix} \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} J_2 \\ J_3 \end{bmatrix}. \tag{3.15}$$

However, the sign convention for J_3 differs between these two cells, since N is computed using an outward-oriented normal vector. Asserting conservation laws hold on $\mathcal{I}_3$ therefore

means enforcing that the sum of each J_3 is equal to zero:

$$T^{\alpha}_{31}\phi_1 + T^{\alpha}_{33}\phi_3 + T^{\beta}_{32}\phi_2 + T^{\beta}_{33}\phi_3 = 0$$

$$\phi_3 = (T^{\alpha}_{33} + T^{\beta}_{33})^{-1}\begin{bmatrix} -T^{\alpha}_{31} & -T^{\beta}_{32} \end{bmatrix}\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}, \tag{3.16}$$

$$S^{\tau} \equiv (T^{\alpha}_{33} + T^{\beta}_{33})^{-1}\begin{bmatrix} -T^{\alpha}_{31} & -T^{\beta}_{32} \end{bmatrix}, \qquad \phi_3 = S^{\tau}\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}. \tag{3.17}$$

With S^{τ} defined, one can show by substituting in ϕ_3 to the expressions for J_1 and J_2 in Equation (3.15) that:

$$\begin{aligned}\begin{bmatrix} J_1 \\ J_2 \end{bmatrix} &= \begin{bmatrix} T^{\alpha}_{11}\phi_1 + T^{\alpha}_{13}\phi_3 \\ T^{\beta}_{22}\phi_2 + T^{\beta}_{23}\phi_3 \end{bmatrix} = \begin{bmatrix} T^{\alpha}_{11} & 0 \\ 0 & T^{\beta}_{22} \end{bmatrix}\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} + \begin{bmatrix} T^{\alpha}_{13} \\ T^{\beta}_{23} \end{bmatrix}\phi_3 \\ &= \left(\begin{bmatrix} T^{\alpha}_{11} & 0 \\ 0 & T^{\beta}_{22} \end{bmatrix} + \begin{bmatrix} T^{\alpha}_{13} \\ T^{\beta}_{23} \end{bmatrix} S^{\tau}\right)\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix},\end{aligned} \tag{3.18}$$

$$T^{\tau} \equiv \begin{bmatrix} T^{\alpha}_{11} & 0 \\ 0 & T^{\beta}_{22} \end{bmatrix} + \begin{bmatrix} T^{\alpha}_{13} \\ T^{\beta}_{23} \end{bmatrix} S^{\tau}, \qquad \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} = T^{\tau}\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}. \tag{3.19}$$

We now pause to comment on the physical meaning of the terms in Equation (3.19). The first term is a composite operator handling the response on the boundary of each child individually. Because the boundary set is composed of 2 disjoint parts, this produces a diagonal structure. The second term expresses the effect from flux transiting across one cell to affect the current on the other side.

Equation (3.17) is somewhat alarming, however. It contains a matrix inverse, and while the properties of T are such that there are fast methods available to compute it, it will be a notable factor in the expected runtime. This will be discussed further in Section 7.3.

When this process is completed for every node, the decomposition is fully built. This algorithm in full is succinctly defined in Algorithm 1. The end result of this decomposition is a single T operator for the root, and an S^{τ} and H^{τ} defined for each node. In addition, all leaves will have an R^{τ} and G^{τ}. Everything else may be discarded as we iterate up the levels of the tree.

Of some note is the traversal pattern, which is depth-first. This pattern of traversal means

Algorithm 1 HPS Build Stage

Let $\mathcal{D}$ be a Binary Tree of subdomains, with each node τ containing $\phi(\mathcal{I}_i^\tau)$

for $\tau \in$ DFS($\mathcal{D}$) **do** ▷ Depth-First Traversal

  if τ is a Leaf **then**

    $X^\tau \leftarrow (A_{ii}^\tau)^{-1}$

    $S^\tau \leftarrow -X^\tau A_{ib}^\tau$ ▷ $\phi(\mathcal{I}_b^\tau) \mapsto \phi(\mathcal{I}_i^\tau)$

    $T^\tau \leftarrow \mathcal{N}_{bb}^\tau + \mathcal{N}_{bi}^\tau S^\tau$ ▷ $\phi(\mathcal{I}_b^\tau) \mapsto J(\mathcal{I}_b^\tau)$

    $R^\tau \leftarrow -X^\tau A_{ic}^\tau$ ▷ $\phi(\mathcal{I}_c^\tau) \mapsto \phi(\mathcal{I}_i^\tau)$

    $H^\tau \leftarrow \mathcal{N}_{bi}^\tau X^\tau$ ▷ $Q(\mathcal{I}_i^\tau) \mapsto J(\mathcal{I}_b^\tau)$

    $G^\tau \leftarrow \mathcal{N}_{bc}^\tau$ ▷ $\phi(\mathcal{I}_c^\tau) \mapsto J(\mathcal{I}_b^\tau)$

  else

    Let α and β be the children of τ

    Let $\mathcal{I}_1 = \mathcal{I}_b^\alpha - \mathcal{I}_b^\beta$ ▷ See Figure 3.6

      $\mathcal{I}_2 = \mathcal{I}_b^\beta - \mathcal{I}_b^\alpha$

      $\mathcal{I}_3 = \mathcal{I}_b^\alpha \cap \mathcal{I}_b^\beta = \mathcal{I}_i^\tau$

    $X^\tau \leftarrow (T_{33}^\alpha + T_{33}^\beta)^{-1}$

    $S^\tau \leftarrow X^\tau \begin{bmatrix} -T_{31}^\alpha & -T_{32}^\beta \end{bmatrix}$ ▷ $\phi(\mathcal{I}_b^\tau) \mapsto \phi(\mathcal{I}_i^\tau)$

    $H^\tau \leftarrow \begin{bmatrix} T_{13}^\alpha \\ T_{23}^\beta \end{bmatrix} X^\tau$ ▷ $J(\mathcal{I}_i^\tau) \mapsto J(\mathcal{I}_b^\tau)$

    $T^\tau \leftarrow \begin{bmatrix} T_{11}^\alpha & 0 \\ 0 & T_{22}^\alpha \end{bmatrix} + \begin{bmatrix} T_{13}^\alpha \\ T_{23}^\beta \end{bmatrix} S^\tau$ ▷ $\phi(\mathcal{I}_b^\tau) \mapsto J(\mathcal{I}_b^\tau)$

    Delete T^α and T^β

  end if

end for

that no node is 'processed' without first processing its children. It is a way to ensure that the required T operators are present during the build step of the branches and root. Also of importance are the index spaces defined on the branches/root, which are a more general version of those defined on Figure 3.6. From here we move on to the solve stage.

3.2.3. Solve Stage

The solve stage of the algorithm represents steps 2–4 of Figure 3.5. As mentioned previously, the corner points are assumed to be known. Step 2 involves an "upward" sweep from the leaf nodes as particular solutions are constructed from the applied source Q and the corner points. Let ϕ and g be 2 vectors of unknowns spanning the whole space. Here, g represents the currents induced by the inhomogeneous source term. The purpose of the upward sweep is to propagate these currents to the boundary of the problem. To do so, we may compute

$$\phi(\mathcal{I}_i^\tau) = \begin{cases} X^\tau Q_i^\tau + R^\tau \phi(\mathcal{I}_c^\tau) \\ X^\tau g(\mathcal{I}_i^\tau) \end{cases} \qquad g(\mathcal{I}_b^\tau) \quad = \begin{cases} -H^\tau Q_i^\tau(\mathcal{I}_i^\tau) - G^\tau \phi(\mathcal{I}_c^\tau) & \tau \text{ is a leaf} \\ -H^\tau \phi(\mathcal{I}_i^\tau) & \tau \text{ is a branch} \end{cases}, \tag{3.20}$$

where the order of the sweep proceeds from the children to the parents. The extra corner terms R^τ and G^τ are no longer needed after processing the leaves, because they must only be accounted for once in the sum.

When this upward pass is complete, the induced current at the domain boundary has been fully constructed. In step 3, this induced current is used in conjunction with the root T operator to compute the value of the flux moments at the boundary via the specified boundary condition.

The final step, the downward sweep, then proceeds to compute the solution from the boundary, mapping the solution to node interiors. In this iteration,

$$\phi(\mathcal{I}_i^\tau) = \phi(\mathcal{I}_i^\tau) + S^\tau \phi(\mathcal{I}_b^\tau), \tag{3.21}$$

with the reverse of the previous condition; the parent of any given node must be processed before the children. Once this iteration is complete, ϕ contains the solution to the discretized PDE. The procedure is outlined in Algorithm 2

The first stage of this algorithm is the depth-first traversal which, again, ensures that children are processed before parents. In this stage, a temporary variable g is used to accumulate

Algorithm 2 HPS Solve Stage

Let $\mathcal{D}$ be a Binary Tree of subdomains, with each node τ containing the indices $\mathcal{I}_i^\tau$

Let g and ϕ be vectors defined over all of $\mathcal{D}$ and the domain boundary.

for $\tau \in \text{DFS}(\mathcal{D})$ **do** ▷ Depth-First Traversal

  if τ is a Leaf **then**

    $\phi(\mathcal{I}_i^\tau) = X^\tau \left[Q(\vec{x}_k)\right]_{k \in \mathcal{I}_i^\tau}$

    $g(\mathcal{I}_i^\tau) = -H^\tau \left[Q(x_k)\right]_{k \in \mathcal{I}_i^\tau} - G^\tau \phi(\mathcal{I}_c^\tau)$

  else

    $\phi(\mathcal{I}_i^\tau) = X^\tau g(\mathcal{I}_i^\tau)$

    $g(\mathcal{I}_i^\tau) = -H^\tau \phi(\mathcal{I}_i^\tau)$

  end if

end for

$\phi(\mathcal{I}_b^{\text{root}}) = \left[\phi(\vec{x}_k)\right]_{k \in \mathcal{I}_b^{\text{root}}}$

for $\tau \in \text{BFS}(\mathcal{D})$ **do** ▷ Breadth-First Traversal

  $\phi(\mathcal{I}_i^\tau) = \phi(\mathcal{I}_i^\tau) + S^\tau \phi(\mathcal{I}_b^\tau)$

end for

the currents induced by the inhomogeneous source. At the root node, the flux is then found using the boundary condition. For simplicity, this algorithm shows the procedure for the Dirichlet condition; which simply requires setting the flux. Neumann conditions may be used instead by solving a system involving g, T^{root}, and the specified values. The procedure is completed with a downward-sweep, shown here with a breadth-first traversal. This type of traversal will process the parent before moving onto the children, ensuring that $\phi(\mathcal{I}_b^\tau)$ will always exist at the time a node is processed.

3.3. Summary

In this section, the HPS algorithm has been written more or less as contemporarily derived. There are a handful of adjustments; but otherwise this derivation is not especially remarkable, and generally similar to existing versions [46]. Perhaps the most significant adjustment has been the more codified relationship between the corner points and the bulk problem; a topic which will return in Section 4.7.

This algorithm is quite general, and applies to virtually any elliptic PDE. Non-elliptic PDEs could in theory still be solved by this method; but the guarantee of compressibility is lost, and there would be little motivation in pursuing it. However, deriving the required operators is not simple. A prescription is needed for the operators $A(\mathcal{I}_i, \mathcal{I}_{\text{all}})$ and $\mathcal{N}$, used in the definitions for the leaves above. These are closely tied to the physics of the underlying problem, and deriving these will be the topic of the upcoming chapter.

Chapter 4

Derivation of HPS Kernels for Low-Order Transport

4.1. Indexing

While an indexing scheme has been hinted at in some previous sections, a more concrete one will be laid down here. In a cell, the order of indices proceeds from the boundary to the interior to the corners. Figure 4.1 shows this for 4th-order basis functions.

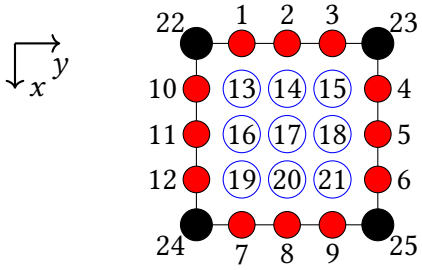


Figure 4.1: Depiction of Cell Indexing.

This scheme is altered for lower- or higher-ordered bases by expanding or shrinking the interior (blue) square and shifting the corner indices appropriately. Accordingly, one cannot use less than quadratic bases; to do so would make the interior set empty. Note that the coordinates match the indexing scheme of 2D arrays; the first coordinate x corresponds to the first index of the array, and so on.

Along the boundaries, the ordering is *not* rotational because the ordering must be shared

with adjacent nodes. A rotational ordering would cause complications in referencing boundary values because the ordering flips when considering two adjacent nodes. In a structured grid, the ordering may be taken as always increasing in x and y, the global coordinates. This eliminates ambiguity because the structured nature of the mesh may always be mapped to a Cartesian grid.

However, this is merely a convenient simplification. An unstructured grid may still be ordered; but this order must be determined and assigned when the unstructured grid is bisected to form nodes.

4.2. Energy Group Scattering

The derivations to follow in Sections 4.4 to 4.6 are for a single energy group. The reason for this is that the work to be done depends strongly on the coupling width between cells. The number of groups is a multiplier on this quantity, because a given spatial point will have at least as many unknowns as there are groups.

Since repeated solves of this decomposition are fast, we opt to build separate decompositions for each energy group and resolve scattering behavior via group sweeping. The structure of the discretization makes any other form of acceleration or blocking in energy group infeasible. As the energy group is a direct multiplier on the size of most operators, it substantially increases the work involved while interfering with their compressibility. The scattering source is updated continually through the sweep, which proceeds among the groups involved in the upscattering process until convergence. Convergence in this instance is measured by the absolute maximum norm ℓ^1.

In the scheme used for the remainder of this paper, the solution within each energy group (as well as its corner point values; see Section 4.7) is fully converged before continuing to the next group. Partial convergence for the group-wise flux has been implemented, but not observed to provide a benefit in the test cases studied for Chapters 5 and 6.

4.3. Coordinate Transformation

The differential equations forming the problem of interest; whether that is diffusion, SP_N, or GSP_N; are all given in global coordinates $\vec{x} = (x_1, x_2)$. However, when solving on individual

cells, the quadrilaterals that form the mesh are taken to the unit cell $\vec{u} = (u_1, u_2) \in [-1, 1]^2$ This has a significant impact on the computation of any derivatives or integrals we intend to compute, which must take this transformation into account.

Recall that a linear mapping may be described in matrix form as:

$$\begin{bmatrix} \partial x_1 \\ \partial x_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial x_1}{\partial u_1} & \frac{\partial x_1}{\partial u_2} \\ \frac{\partial x_2}{\partial u_1} & \frac{\partial x_2}{\partial u_2} \end{bmatrix} \begin{bmatrix} \partial u_1 \\ \partial u_2 \end{bmatrix}, \qquad \begin{bmatrix} \partial u_1 \\ \partial u_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \frac{\partial u_1}{\partial x_2} \\ \frac{\partial u_2}{\partial x_1} & \frac{\partial u_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \partial x_1 \\ \partial x_2 \end{bmatrix}.$$

These are two conventions for the Jacobian matrix J. We will use the form:

$$J = \begin{bmatrix} \frac{\partial x_1}{\partial u_1} & \frac{\partial x_1}{\partial u_2} \\ \frac{\partial x_2}{\partial u_1} & \frac{\partial x_2}{\partial u_2} \end{bmatrix} = \left(\begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \frac{\partial u_1}{\partial x_2} \\ \frac{\partial u_2}{\partial x_1} & \frac{\partial u_2}{\partial x_2} \end{bmatrix} \right)^{-1}, \qquad \text{or} \qquad J_{ij} = \frac{\partial x_i}{\partial u_j}, \qquad J^{-1}_{ij} = \frac{\partial u_i}{\partial x_j}. \tag{4.1}$$

These relations may be written much more compactly using the implied-sum notation, where repeated indices imply a sum over that index. This notation will be used throughout the remainder of this section, where:

$$\frac{\partial}{\partial u_j} = J_{ij} \frac{\partial}{\partial x_i} \qquad \text{and} \qquad \frac{\partial}{\partial x_j} = J^{-1}_{ij} \frac{\partial}{\partial u_i}. \tag{4.2}$$

The gradient operator in local coordinates is trivial to derive, and expressed in Equation (4.3). Because it is a single derivative, the spatial variation of the transformation does not enter into the expression. The gradient of a function f is simply expressed in local coordinates as:

$$(\nabla_{\vec{x}} f)_i = \frac{\partial}{\partial x_i} f = \frac{\partial u_j}{\partial x_i} \frac{\partial}{\partial u_j} f = J^{-1}_{ji} (\nabla_{\vec{u}} f)_j = J^{-T} \nabla_{\vec{u}} f \tag{4.3}$$

For a normal derivative we may simply prepend the above expression with an n_i (expressed in global coordinates) to contract this into a scalar.

Higher-order derivatives are more involved, however. When discussing GSP_3 in Section 4.6 we will see that we need a total of 4 higher-order differential operators. These are ∇^2, $(\hat{n} \cdot \nabla)^2$, $(\hat{n} \cdot \nabla)^3$, and $(\hat{n} \cdot \nabla)\nabla^2$. In order to simplify the resulting calculations, we will define tensors Ξ:

$$\Xi_{ijk} = \frac{\partial}{\partial x_j} \frac{\partial}{\partial x_k} u_i \qquad \Xi_{\mu ijk} = \frac{\partial}{\partial x_i} \frac{\partial}{\partial x_j} \frac{\partial}{\partial x_k} u_\mu \tag{4.4}$$

and let $\mathbb{D}_d^u$ be a rank-d tensor denoting the d-th mixed derivative in the coordinate system specified by u, such that $\mathbb{D}_1^u = \nabla_{\vec{u}}$. It is by definition symmetric under index permutations.

Now, consider the Laplacian ∇^2:

$$\begin{aligned}\nabla^2_{\vec{x}} f = (\mathbb{D}_2^x f)_{ii} &= \frac{\partial}{\partial x_i}\left(\frac{\partial}{\partial x_i} f\right) = \frac{\partial}{\partial x_i}\left(\frac{\partial u_j}{\partial x_i}(\nabla_{\vec{u}} f)_j\right) = \left(\frac{\partial}{\partial x_i}\frac{\partial u_j}{\partial x_i}\right)(\nabla_{\vec{u}} f)_j + \frac{\partial u_j}{\partial x_i}\left(\frac{\partial}{\partial x_i}(\nabla_{\vec{u}} f)_j\right) \\ &= \Xi_{jii}(\nabla_{\vec{u}} f)_j + \frac{\partial u_j}{\partial x_i}\frac{\partial u_k}{\partial x_i}(\mathbb{D}_2^u f)_{kj},\end{aligned} \tag{4.5}$$

and its normal derivative $(\hat{n}\cdot\nabla)\nabla^2$:

$$\begin{aligned}(\hat{n}\cdot\nabla)\nabla^2 &= n_\alpha \frac{\partial}{\partial x_\alpha}\left[\Xi_{jii}(\nabla_{\vec{u}} f)_j + \frac{\partial u_j}{\partial x_i}\frac{\partial u_k}{\partial x_i}(\mathbb{D}_2^u f)_{kj}\right] \\ &= n_\alpha\left[\frac{\partial}{\partial x_\alpha}\left(\Xi_{jii}(\nabla_{\vec{u}} f)_j\right) + \frac{\partial}{\partial x_\alpha}\left(\frac{\partial u_j}{\partial x_i}\frac{\partial u_k}{\partial x_i}(D_2^u f)_{kj}\right)\right] \\ &= n_\alpha\Big[\Xi_{j\alpha ii}(\nabla_{\vec{u}} f)_j + \Xi_{jii}\frac{\partial u_\mu}{\partial x_\alpha}(\mathbb{D}_2^u f)_{\mu j} + \Xi_{j\alpha i}\frac{\partial u_k}{\partial x_i}(\mathbb{D}_2^u f)_{kj} \\ &\qquad + \frac{\partial u_j}{\partial x_i}\Xi_{k\alpha i}(\mathbb{D}_2^u f)_{kj} + \frac{\partial u_j}{\partial x_i}\frac{\partial u_k}{\partial x_i}\frac{\partial u_\mu}{\partial x_\alpha}(\mathbb{D}_3^u f)_{\mu kj}\Big].\end{aligned} \tag{4.6}$$

From here we move on to the second directional derivative $(\hat{n}\cdot\nabla)^2$. Note that we are assuming non-curvilinear mesh boundaries, so the normal vectors are fixed. The result may be written as:

$$(\hat{n}\cdot\nabla_{\vec{x}})^2 = n_\alpha\frac{\partial}{\partial x_\alpha}\left(n_\gamma\frac{\partial}{\partial x_\gamma} f\right) = n_\alpha n_\gamma\frac{\partial}{\partial x_\alpha}\left(\frac{\partial u_\delta}{\partial x_\gamma}\frac{\partial}{\partial u_\delta} f\right) = n_\alpha n_\gamma\left(\Xi_{\delta\alpha\gamma}(\nabla_{\vec{u}} f)_\delta + \frac{\partial u_\delta}{\partial x_\gamma}\frac{\partial u_\beta}{\partial x_\alpha}(\mathbb{D}_2^u f)_{\beta\delta}\right). \tag{4.7}$$

To go from this to the third directional derivative we simply prepend another directional derivative operator:

$$\begin{aligned}(\hat{n}\cdot\nabla_x)^3 &= n_\mu\frac{\partial}{\partial x_\mu}\left(n_\alpha n_\gamma\left(\Xi_{\delta\alpha\gamma}(\nabla_{\vec{u}} f)_\delta + \frac{\partial u_\delta}{\partial x_\gamma}\frac{\partial u_\beta}{\partial x_\alpha}(\mathbb{D}_2^u f)_{\beta\delta}\right)\right) \\ &= n_\mu n_\alpha n_\gamma\frac{\partial}{\partial x_\mu}\left(\Xi_{\delta\alpha\gamma}(\nabla_{\vec{u}} f)_\delta + \frac{\partial u_\delta}{\partial x_\gamma}\frac{\partial u_\beta}{\partial x_\alpha}(\mathbb{D}_2^u f)_{\beta\delta}\right) \\ &= n_\mu n_\alpha n_\gamma\left[\Xi_{\delta\mu\alpha\gamma}(\nabla_{\vec{u}} f)_\delta + \Xi_{\delta\alpha\gamma}\frac{\partial u_\nu}{\partial x_\mu}(\mathbb{D}_2^u f)_{\nu\delta} + \frac{\partial}{\partial x_\mu}\left(\frac{\partial u_\delta}{\partial x_\gamma}\frac{\partial u_\beta}{\partial x_\alpha}(\mathbb{D}_2^u f)_{\beta\delta}\right)\right].\end{aligned} \tag{4.8}$$

The final term within the brackets above may be expanded as:

$$\frac{\partial}{\partial x_\mu}\left(\frac{\partial u_\delta}{\partial x_\gamma}\frac{\partial u_\beta}{\partial x_\alpha}(\mathbb{D}_2^u f)_{\beta\delta}\right) = \Xi_{\delta\mu\gamma}\frac{\partial u_\beta}{\partial x_\alpha}(\mathbb{D}_2^u f)_{\beta\delta} + \frac{\partial u_\delta}{\partial x_\gamma}\Xi_{\beta\mu\alpha}(\mathbb{D}_2^u f)_{\beta\delta} + \frac{\partial u_\delta}{\partial x_\gamma}\frac{\partial u_\beta}{\partial x_\alpha}\frac{\partial u_\nu}{\partial x_\mu}(\mathbb{D}_3^u f)_{\nu\beta\delta}.$$

And so:

$$\begin{aligned}(\hat{n}\cdot\nabla_x)^3 = n_\mu n_\alpha n_\gamma\Bigg[&\Xi_{\delta\mu\alpha\gamma}(\nabla_{\vec{u}} f)_\delta + \Xi_{\delta\alpha\gamma}\frac{\partial u_\nu}{\partial x_\mu}(\mathbb{D}_2^u f)_{\nu\delta} + \Xi_{\delta\mu\gamma}\frac{\partial u_\beta}{\partial x_\alpha}(\mathbb{D}_2^u f)_{\beta\delta} \\ &+ \frac{\partial u_\delta}{\partial x_\gamma}\Xi_{\beta\mu\alpha}(\mathbb{D}_2^u f)_{\beta\delta} + \frac{\partial u_\delta}{\partial x_\gamma}\frac{\partial u_\beta}{\partial x_\alpha}\frac{\partial u_\nu}{\partial x_\mu}(\mathbb{D}_3^u f)_{\nu\beta\delta}\Bigg].\end{aligned} \tag{4.9}$$

For a description of how the tensors Ξ may be computed, see Appendix B.

4.4. Diffusion

With the preliminaries now over, let us know derive the HPS kernel for the diffusion equations. This necessitates a scheme for determining A_{ii}, A_{ib}, and A_{ic} from Equation (3.3); as well as the operator $\mathcal{N}$. These operators will allow the construction of all the required matrices in Algorithms 1 and 2.

We first derive a scheme for the sub-matrices of A. These are generated from the WRM equations associated with the internal basis functions $b_\mu\ \forall \mu \in \mathcal{I}_i$. In this scheme, μ is effectively an integer indexing a given point within $\mathcal{I}_i$, as in Figure 4.1. Using this notation, the differential operator for diffusion may be plugged into Equation (1.6) to obtain:

$$\int_{\text{cell }\tau} b_\mu(\vec{x})\left[\sum_{\nu\in\mathcal{I}_{\text{all}}}\phi_\nu\left(-D^\tau\nabla_{\vec{x}}^2 + \Sigma_r^\tau\right)b_\nu(\vec{x})\right]d\vec{x} = \int_{\text{cell }\tau} b_\mu(\vec{x})Q(\vec{x})d\vec{x}, \tag{4.10}$$

where ν, like μ, is indexing basis functions in the manner of Figure 4.1. The entries of A are then given by extracting terms of this sum; the row μ, column ν entry of A is the coefficient of the basis expansion function b_ν of equation corresponding to weighting function $\omega_\mu = b_\mu$, since a Galerkin scheme is used. Plugging in the differential operator to Equation (1.6) and

selecting b_μ from the sum, each entry of the matrix may be computed as:

$$A^\tau(\mathcal{I}_i, \mathcal{I})_{\mu\nu} = -D^\tau \int_{\text{cell } \tau} b_\mu(\vec{x}) \nabla^2_{\vec{x}} b_\nu(\vec{x}) \, d\vec{x} + \Sigma_r^\tau \int_{\text{cell } \tau} b_\mu(\vec{x}) b_\nu(\vec{x}) \, d\vec{x}. \tag{4.11}$$

We now seek to simplify the streaming term by eliminating second-order derivatives, for reasons which will become clear shortly. Expanding via an integration-by-parts yields

$$\begin{aligned}\int_{\text{cell } \tau} b_\mu(\vec{x}) \nabla^2_{\vec{x}} b_\nu(\vec{x}) \, d\vec{x} &= \int_{\text{cell } \tau} \left[\nabla_{\vec{x}} \cdot (b_\mu(\vec{x}) \nabla_{\vec{x}} b_\nu(\vec{x})) - (\nabla_{\vec{x}} b_\mu(\vec{x})) \cdot (\nabla_{\vec{x}} b_\nu(\vec{x})) \right] d\vec{x} \\ &= \oint_{\partial(\text{cell } \tau)} b_\mu(\vec{x}) \nabla_{\vec{x}} b_\nu(\vec{x}) \, d\vec{x} - \int_{\text{cell } \tau} \nabla_{\vec{x}} b_\mu(\vec{x}) \cdot \nabla_{\vec{x}} b_\nu(\vec{x}) d\vec{x}. \end{aligned} \tag{4.12}$$

With the streaming term decomposed, we may now make a simplification; since we are only concerned with $\mu \in \mathcal{I}_i$, and the basis functions associated with this space are zero on the boundary, we can eliminate the surface terms entirely. The expression then simplifies to:

$$A^\tau(\mathcal{I}_i, \mathcal{I})_{\mu\nu} = D^\tau \int_{\text{cell } \tau} \nabla_{\vec{x}} b_\mu(\vec{x}) \cdot \nabla_{\vec{x}} b_\nu(\vec{x}) d\vec{x} + \Sigma_r^\tau \int_{\text{cell } \tau} b_\mu(\vec{x}) b_\nu(\vec{x}) \, d\vec{x}. \tag{4.13}$$

From here we convert the derivatives in global coordinates and function evaluations to the local ones. This is particularly important because a normal derivative in global coordinates does not translate to a normal derivative in local coordinates. In general, off-diagonal terms in the inverse Jacobian will create a derivative in local coordinates with components in both the normal and tangential directions. It is necessary to insert an operation by this matrix at each gradient computation as in Equation (4.3)

$$\begin{aligned} A^\tau(\mathcal{I}_i, \mathcal{I})_{\mu\nu} = D^\tau &\int_{\text{cell } \tau} (J^{-T}(\vec{u}) \nabla_{\vec{u}} b_\mu(\vec{u})) \cdot (J^{-T}(\vec{u}) \nabla_{\vec{u}} b_\nu(\vec{u})) |J^{-1}(\vec{u})| d\vec{u} \\ &+ \Sigma_r^\tau \int_{\text{cell } \tau} b_\mu(\vec{u}) b_\nu(\vec{u}) |J^{-1}(\vec{u})| d\vec{u}, \end{aligned} \tag{4.14}$$

where cell τ is conventionally the reference cell domain ($[-1,1]^2$ for 2D, $[-1,1]^3$ for 3D). Note also that $|J^{-1}(\vec{u})|$ denotes the determinant of the inverse Jacobian matrix.

There are multiple options for evaluating this integral. The simplest and fastest is to simply use a Gauss-Lobatto quadrature as this basis has been designed for. This quadrature is only exact for polynomials up to order $2n-3$, where n is the number of quadrature points,

meaning that this will never yield exact results given the presence of a polynomial of order $2n$ as well as the Jacobian.

If more precise results are required, one may also choose to use a more detailed numeric integration procedure. While this is a large cost owing to the number of Jacobian evaluations it will require, it does not fundamentally affect the scaling of the method. The effects that these two choices have on the error will be seen in Chapter 5.

Implementing the quadrature introduces a sum in γ over the full index space. Let $\vec{u}_\gamma$ and ω_γ be the quadrature points and weights, respectively; and further, let J_γ^{-T} be the inverse Jacobian matrix transposed and evaluated at quadrature point γ. Then, since the basis functions are orthonormal with respect to this quadrature, the diffusion kernel for A may be expressed as:

$$A^\tau(\mathcal{I}_i,\mathcal{I})_{\mu\nu} = \sum_{\gamma\in\mathcal{I}_{\text{all}}} \frac{\omega_\gamma}{|J_\gamma|}\left[D^\tau\left(J_\gamma^{-T}\cdot\left[\nabla_{\vec{u}}b_\mu\right]_{\vec{u}=\vec{u}_\gamma}\right)\cdot\left(J_\gamma^{-T}\cdot\left[\nabla_{\vec{u}}b_\nu\right]_{\vec{u}=\vec{u}_\gamma}\right)+\Sigma_r^\tau\delta_{\mu\gamma}\delta_{\nu\gamma}\right]. \tag{4.15}$$

This diffusion kernel may be simplified via the definition of a "local current" vector $\vec{\zeta}_{\gamma\mu} = (J_\gamma^{-T}\left[\nabla_{\vec{u}}b_\mu\right]_{\vec{u}=\vec{u}_\gamma})$. This term does not have any physical meaning, but is a convenient quantity, allowing the kernel to be written compactly as:

$$A^\tau(\mathcal{I}_i,\mathcal{I})_{\mu\nu} = \sum_{\gamma\in\mathcal{I}_{\text{all}}} \frac{\omega_\gamma}{|J_\gamma|}\left[D^\tau(\vec{\zeta}_{\gamma\mu}\cdot\vec{\zeta}_{\gamma\nu})+\Sigma_r^\tau\delta_{\mu\gamma}\delta_{\nu\gamma}\right]. \tag{4.16}$$

To complete the kernel definition, we define also the current operator $\mathcal{N}$ as:

$$\mathcal{N}^\tau(\mathcal{I}_b,\mathcal{I})_{\mu\nu} = -D^\tau\left(\hat{n}_\mu\cdot J_\mu^{-T}\cdot\left[\nabla_{\vec{u}}b_\nu\right]_{\vec{u}=\vec{u}_\mu}\right) = -D^\tau\left(\hat{n}_\mu\cdot\vec{\zeta}_{\mu\nu}\right), \tag{4.17}$$

by extracting from Equation (1.22) the definition of $\underline{J}_0$.

4.5. SP_3

This process may be repeated for SP_3 with only minor changes. The most significant change occurs because SP_3 is a PDE over 2 fields (ϕ_0 and ϕ_2) rather than just one; which affects the indexing and flattening of $\mu,\nu,\gamma\in\mathcal{I}_{\text{all}}$. For our purpose, we will arrange these so that the locality of the sets $\mathcal{I}_b$, $\mathcal{I}_i$, and $\mathcal{I}_c$ are conserved; that is, the field index will be the innermost

component of the loop, such that the coefficients ϕ_0 and ϕ_2 at a point are always adjacent.

We will reuse the previous definition of the local current, $\vec{\zeta}$. Furthermore, we adopt the convention for flux moments that define the scalar flux as $\phi = \phi_0 - 2\phi_2$ (preserving the diffusion-like structure of the underlying PDE). Then, the SP_3 kernel for $A^\tau_{\mu\nu}$ may be expressed as the 2×2 matrix acting on the 0th and 2nd order moments:

$$A^\tau(\mathcal{I}_i, \mathcal{I})_{\mu\nu} = \sum_{\gamma \in \mathcal{I}_{\text{all}}} \frac{\omega_\gamma}{|J_\gamma|} \begin{bmatrix} D^\tau_0(\vec{\zeta}_{\gamma\mu} \cdot \vec{\zeta}_{\gamma\nu}) + \Sigma^\tau_{r0}\delta_{\mu\gamma}\delta_{\nu\gamma} & -2\Sigma^\tau_{r0}\delta_{\mu\gamma}\delta_{\nu\gamma} \\ -\frac{2}{5}\Sigma^\tau_{r0}\delta_{\mu\gamma}\delta_{\nu\gamma} & D^\tau_2(\vec{\zeta}_{\gamma\mu} \cdot \vec{\zeta}_{\gamma\nu}) + \left(\Sigma^\tau_{r2} + \frac{4}{5}\Sigma^\tau_{r0}\right)\delta_{\mu\gamma}\delta_{\nu\gamma} \end{bmatrix}. \tag{4.18}$$

As before, the current operator is defined to complete the kernel derivation:

$$\mathcal{N}^\tau(\mathcal{I}_b, \mathcal{I})_{\mu\nu} = \begin{bmatrix} D^\tau_0\left(\hat{n}_\mu \cdot \vec{\zeta}_{\mu\nu}\right) & 0 \\ 0 & D^\tau_2\left(\hat{n}_\mu \cdot \vec{\zeta}_{\mu\nu}\right) \end{bmatrix}. \tag{4.19}$$

4.6. GSP_3

The interface conditions defined in Chao's series of papers [11, 12, 13, 14] are defined through the definition of 2 flux-like and 2 current-like conserved quantities. These conserved quantities are reproduced below for convenience, using the same convention as the end of Section 1.4.3 where scalar flux $\phi = \phi_0 - 2\phi_2$.

$$\underline{\phi}_0(\vec{x}) = \phi_0(\vec{x}) - 2\phi_2(\vec{x}), \tag{4.20a}$$

$$\underline{J}_0(\vec{x}, \hat{n}) = -\frac{\nabla_n}{3\Sigma_t}\phi_0(\vec{x}), \tag{4.20b}$$

$$\underline{\phi}_2(\vec{x}, \hat{n}) = \phi_2(\vec{x}) - \frac{3}{2}\left(\nabla^2 - \nabla_n^2\right)\left(\frac{2}{15\Sigma_t^2}\phi_0(\vec{x}) + \frac{9}{35\Sigma_t^2}\phi_2(\vec{x})\right), \tag{4.20c}$$

$$\underline{J}_2(\vec{x}, \hat{n}) = -\frac{\nabla_n}{\Sigma_t}\left[\frac{2}{15}\phi_0(\vec{x}) + \frac{9}{35}\phi_2(\vec{x})\right] + \frac{9}{14}\frac{\nabla_n(\nabla^2 - \nabla_n^2)}{\Sigma_t^3}\left[\frac{2}{15}\phi_0(\vec{x}) + \frac{9}{35}\phi_2(\vec{x})\right], \tag{4.20d}$$

where $\nabla_n = (\hat{n} \cdot \nabla)$. The underbar on the LHS of these equations denotes that the value is conserved at boundaries.

Naturally, the flux-like quantities have even-ordered derivatives, while the current-like quantities have odd ones. It is also worth noting that both second-order quantities have

a directionality in their definition; rendering conservation on the boundary a functional equality. We will return to this issue in Section 4.7

Under our discretization, preserving the flux-like quantities demands that on boundaries, we store the 'underbar' quantities $\underline{\phi_0}$ and $\underline{\phi_2}$. The remaining interior and corner quantities are stored as the coefficients of a Lagrange basis over the set of points, as in normal SP_3 of Section 4.5. This means that an operator is necessary to convert the versions of flux that are stored on boundaries $\underline{\phi}(\mathcal{I}_b)$ and the actual basis coefficients $\phi(\mathcal{I}_b)$.

4.6.1. The Transfer Operator

The transfer operator $\mathcal{T}$ is a way to handle this conversion so it can be encoded into the linear operators that form the HPS kernel. It is a linear map such that

$$\begin{bmatrix} \underline{\phi}_b \\ \phi_i \\ \phi_c \end{bmatrix} = \mathcal{T} \begin{bmatrix} \phi_b \\ \phi_i \\ \phi_c \end{bmatrix} = \begin{bmatrix} \mathcal{T}(\mathcal{I}_b;\mathcal{I}_b) & \mathcal{T}(\mathcal{I}_b;\mathcal{I}_i) & \mathcal{T}(\mathcal{I}_b;\mathcal{I}_c) \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \phi_b \\ \phi_i \\ \phi_c \end{bmatrix}, \tag{4.21}$$

where in practice, a Lagrange basis guarantees this transformation is always full rank and $\mathcal{T}(\mathcal{I}_b;\mathcal{I}_i) = 0$.

The operator itself may be defined element-wise via the flux-like conserved quantities of Equation (4.20). Equation (4.22a) shows this equation in the same format as $\mathcal{N}$ of the HPS kernels:

$$\mathcal{T}_{\mu\nu} = \begin{bmatrix} \delta_{\mu\nu} & -2\delta_{\mu\nu} \\ -\frac{1}{5\Sigma_t^2}\left(\nabla^2 b_\nu - \nabla_n^2 b_\nu\right)_{\vec{u}=\vec{u}_\mu} & \delta_{\mu\nu} - \frac{27}{70\Sigma_t^2}\left(\nabla^2 b_\nu - \nabla_n^2 b_\nu\right)_{\vec{u}=\vec{u}_\mu} \end{bmatrix} \quad \text{for} \quad \mu \in \mathcal{I}_b,\ \nu \in \mathcal{I}_{\text{all}}, \tag{4.22a}$$

$$\mathcal{T}_{\mu\nu} = \begin{bmatrix} \delta_{\mu\nu} & 0 \\ 0 & \delta_{\mu\nu} \end{bmatrix} \quad \text{for} \quad \mu \in \mathcal{I}_i \cup \mathcal{I}_c,\ \nu \in \mathcal{I}_{\text{all}}. \tag{4.22b}$$

To inject this change of basis, the governing equations applied via A as well as the current

expressions must be modified by a similarity transformation:

$$\mathcal{T}A\mathcal{T}^{-1}\begin{bmatrix}\underline{\phi}_b\\ \phi_i\\ \phi_c\end{bmatrix} = \mathcal{T}Q \qquad \text{and} \qquad J = \mathcal{N}\mathcal{T}^{-1}\begin{bmatrix}\underline{\phi}_b\\ \phi_i\\ \phi_c\end{bmatrix}. \tag{4.23}$$

This has some fairly profound impacts on the HPS algorithm as stated in Algorithm 1. These impacts arise from 2 sources: Equation (3.4) and Equation (3.8). We reproduce them below with the added transfer operator:

$$A^\tau_{ii}\phi^H_i + A^\tau_{ib}\mathcal{T}^{-1}_{bb}\underline{\phi}^H_b + \left(A_{ib}\mathcal{T}^{-1}_{bc} + A^\tau_{ic}\right)\phi^H_c = 0$$

$$\phi^H_i = -(A^\tau_{ii})^{-1}A^\tau_{ib}\mathcal{T}^{-1}_{bb}\underline{\phi}^H_b - (A^\tau_{ii})^{-1}\left(A_{ib}\mathcal{T}^{-1}_{bc} + A^\tau_{ic}\right)\phi^H_c. \tag{4.24}$$

Redefine: $$S^\tau = -(A^\tau_{ii})^{-1}A^\tau_{ib}\mathcal{T}^{-1}_{bb} \qquad \text{and} \qquad R^\tau = -(A^\tau_{ii})^{-1}\left(A_{ib}\mathcal{T}^{-1}_{bc} + A^\tau_{ic}\right); \tag{4.25}$$

$$\phi^H_i = S^\tau\underline{\phi}^H_b + R^\tau\phi^H_c. \tag{4.26}$$

This redefines the operators S and R, but otherwise the algorithm is unchanged. Note that restrictions of the inverse transfer operator apply *after* the inverse is computed. Otherwise, the rectangular $\mathcal{T}^{-1}_{bc}$ would not be well-defined. Turning to the currents:

$$\mathcal{N}^\tau\mathcal{T}^{-1}\underline{\phi} = \mathcal{N}^\tau\begin{bmatrix}\mathcal{T}^{-1}_{bb}\underline{\phi}^H_b + \mathcal{T}^{-1}_{bc}\phi^H_c\\ \phi^H_i + \phi^P_i\\ \phi^H_c\end{bmatrix} = \mathcal{N}^\tau\begin{bmatrix}\mathcal{T}^{-1}_{bb}\\ S^\tau\\ 0\end{bmatrix}\phi^H_b + \mathcal{N}^\tau\begin{bmatrix}\mathcal{T}^{-1}_{bc}\\ R^\tau\\ I\end{bmatrix}\phi^H_c + \mathcal{N}^\tau\begin{bmatrix}0\\ (A^\tau_{ii})^{-1}Q^\tau_i\\ 0\end{bmatrix}, \tag{4.27}$$

and again, proceed to define T^τ, G^τ, and H^τ. However, the definitions are slightly modified:

$$T^\tau = \mathcal{N}^\tau(\mathcal{I}_b;\mathcal{I}_b\cup\mathcal{I}_i)\begin{bmatrix}\mathcal{T}^{-1}_{bb}\\ S^\tau\end{bmatrix}, \qquad G^\tau = \mathcal{N}^\tau\begin{bmatrix}\mathcal{T}^{-1}_{bc}\\ R^\tau\\ I\end{bmatrix}, \qquad H^\tau = \mathcal{N}^\tau(\mathcal{I}_b;\mathcal{I}_i)(A^\tau_{ii})^{-1}Q^\tau_i, \tag{4.28}$$

which recovers the same relation as in Section 3.2.2:

$$J = T^\tau\phi^H_b + G^\tau\phi^H_c + H^\tau Q^\tau_i. \tag{4.29}$$

This redefines T and G, but the actual build and solve stages (Algorithms 1 and 2) are otherwise unchanged. Furthermore, these changes only apply to the leaf nodes; so the actual impact on the algorithm as a whole is minimal.

With these differential operators defined element-wise, the transfer operator $\mathcal{T}$ has a full definition. This carries through all the operators in Algorithms 1 and 2, completing the adjustments to the kernel.

4.7. Corner Point Solver

Everything discussed in both this chapter and Chapter 3 has been limited to deriving a solution on the interior and boundary points of cells, never the corner points. Instead, a corner point solution has been assumed to be known, and corrective terms (G^τ and R^τ) derived from that known value.

There is no clean way to assign these corner points to a cell or face, since there are multiple correct ways to do so. Consider a corner on the northeast part of a quadrilateral; this point could be correctly identified with either the northern or eastern face. Doing so would lead to a conservation law applied in one direction but not the other, in an arbitrary choice.

The coupling of these corners also poses a problem. Consider Equation (4.18) for $v \in \mathcal{I}_c$. Clearly all the delta terms vanish, as the sets $\mathcal{I}_i$ and $\mathcal{I}_c$ are disjoint. However, in the case of a diagonal Jacobian matrix (indicating a orthogonal Cartesian mesh), the dot products of local currents $\vec{\zeta}$ will also vanish. This is also true for the case of $\mathcal{N}_{\mu\nu}$, which means that the corner points do not couple at all to the bulk problem. However, they must be computed if a detailed flux reconstruction is desired. The fact they do not couple simply means that the other flux moments need not be recomputed when these values are obtained.

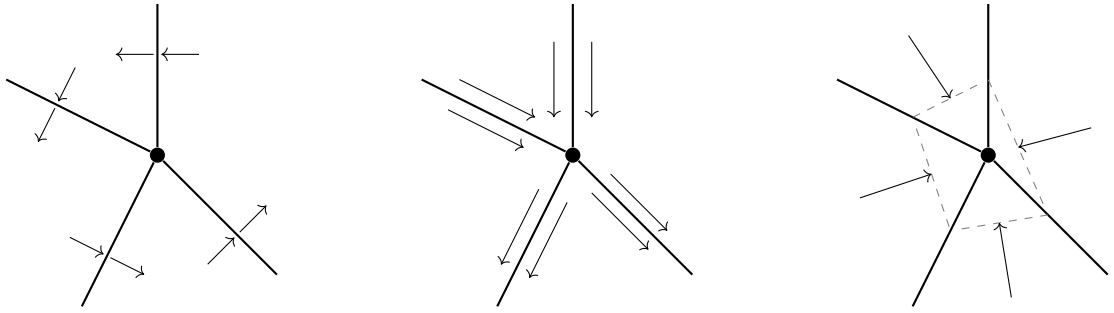
It is worth mentioning that this problem is by no means unique. While the circumstances are somewhat different, early multi-D nodal methods also required corner reconstruction techniques to provide anything more than a node-averaged flux. Slightly more recently, these types of methods have also been used to compute terms arising from TI in hexagonal geometries; when 6 elements intersect at a single point.

In the case of GSP_3 or in the case of a non-regular grid, the coupling terms between the corners and edges do not vanish. Here, the corner points must be resolved in an iteration with the other flux moments; as changes in each affect the values obtained for the other. For this iteration, each leaf node defines a corner point flux values per field, per corner of the

element. Then, an external corner point solver can use the interior and boundary fluxes to derive source terms for a solve operation on these values, and updated corner points may be computed. These are used with the corrective operators G^τ and R^τ in the HPS solver, and the process iterates until convergence. More details on this process will be covered in Section 4.8.1.

The operator generated in this process does not change from iteration to iteration. It may be generated once per problem and factored, if desired, to enable rapid corner point solves. While a scheme for doing this for optimal scaling is not implemented, a factorization using SuperLU [41] has been found to be sufficient for the problems examined in this dissertation.

The equations for these corner point relations come from trying to best satisfy the existing continuity relations on the corner points. Consider a corner consisting of an intersection between 4 elements, depicted in Figure 4.2c. In this case, we define 4 unknowns per field; one per element. In addition to continuity of scalar flux at the corner point, we have also several equations arising from the vector equation corresponding to continuity of the net current. These include continuity of normal currents across boundaries, continuity of parallel currents along boundaries, and a "balance expression" depicted in Figure 4.2c; all located at the corner itself. Which of these we choose to generate the equations depends on the problem at hand; see Sections 4.7.1 and 4.7.2.



(a) Normal Current Equality **(b) Parallel Current Equality** **(c) Current Balance**

Figure 4.2: Expressions of Current Continuity

This approach has significant drawbacks; among others, it is a significant computational cost which carries the risk of destroying the method's optimal performance. However, it is important to note that finding a way around this is an open topic of research. While this method has been chosen for its mathematical and physical rigor, there is in principle nothing wrong with using an approximate method. The crux of this is that whatever approximate

method is used, it must remain stable. However, maintaining this stability is a very thorny problem; and thus far, no approximate method tested has performed adequately for physical meshes.

4.7.1. SP_3

The conventional interface conditions for SP_3 make these the simplest. At an interface, we know that the scalar flux and second moment are typically treated as continuous. This means that for both fields, if the elements touching the corner are indexed by cardinal directions, we may state:

$$\phi^{\mathrm{NW}}(\vec{x}_c) = \phi^{\mathrm{NE}}(\vec{x}_c) = \phi^{\mathrm{SE}}(\vec{x}_c) = \phi^{\mathrm{SW}}(\vec{x}_c). \tag{4.30}$$

Naturally, this is only 3 equations. To close the system we pick the current balance equation depicted in Figure 4.2c, which effectively enforces:

$$\sum_{\text{corner } c} -\hat{r}^c \cdot \vec{J}_0^c(\vec{x}_c) = 0, \tag{4.31}$$

where the unit vector $-\hat{r}$ is the vector directed into the corner point, bisecting the angle formed by the cell edges.

4.7.2. GSP_3

GSP_3 maintains the requirement that scalar flux is continuous, but we can no longer assume that ϕ_2 will be continuous in general. We have conservation of the underbarred quantity $\underline{\phi}_2$, but as mentioned in Section 4.6 this is a functional equality over angle and not straightforward to implement. What we may do instead is assert that the fluxes are equal when considering the normal direction across cells, as shown in Figure 4.3 for a regular grid.

However, these are not independent in all circumstances. For a regular grid, this only amounts to three linearly independent conditions. We again must close the system, and so we choose the current balance equation on the second current for this purpose. So, for the first moment, we have:

$$\underline{\phi}_0^{\mathrm{NW}}(\vec{x}_c) = \underline{\phi}_0^{\mathrm{NE}}(\vec{x}_c) = \underline{\phi}_0^{\mathrm{SE}}(\vec{x}_c) = \underline{\phi}_0^{\mathrm{SW}}(\vec{x}_c), \tag{4.32}$$

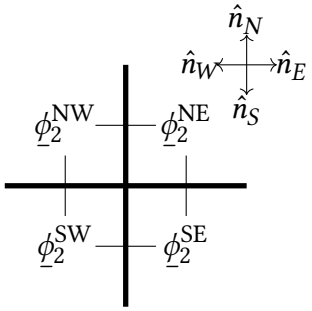


Figure 4.3: Conservation of $\underline{\phi}_2$ for GSP_3

$$\sum_{\text{corner } c} -\hat{r}^c \cdot \vec{J}_0^c(\vec{x}_c) = 0. \tag{4.33}$$

And for the second moment:

$$\underline{\phi}_2^{\mathrm{NE}}(\vec{x}_c, \hat{n}_W) = \underline{\phi}_2^{\mathrm{NW}}(\vec{x}_c, \hat{n}_E), \tag{4.34a}$$
$$\underline{\phi}_2^{\mathrm{SE}}(\vec{x}_c, \hat{n}_W) = \underline{\phi}_2^{\mathrm{SW}}(\vec{x}_c, \hat{n}_E), \tag{4.34b}$$
$$\underline{\phi}_2^{\mathrm{NW}}(\vec{x}_c, \hat{n}_S) = \underline{\phi}_2^{\mathrm{SW}}(\vec{x}_c, \hat{n}_N), \tag{4.34c}$$
$$\underline{\phi}_2^{\mathrm{NE}}(\vec{x}_c, \hat{n}_S) = \underline{\phi}_2^{\mathrm{SE}}(\vec{x}_c, \hat{n}_N), \tag{4.34d}$$

$$\sum_{\text{corner } c} -\hat{r}^c \cdot \vec{J}_2^c(\vec{x}_c, -\hat{r}^c) = 0. \tag{4.35}$$

These second moment equations are an over-specified system, and must be reduced to 4 conditions. This is done by reducing Equation (4.34) to 3 equations; but the manner in which this is done is somewhat arbitrary. While one may simply be omitted, this has the potential to introduce an asymmetric error term relative to the same problem solved with the simpler SP_3 or diffusion conditions. It is suggested instead to reduce these conservation equations to 3 by averaging two equations (one x-directed, one y-directed) together.

The choice of RHS for Equation (4.35) is a pragmatic one. As mentioned previously, this corner point solver must be embedded in an iteration with HPS; which is formally described in Section 4.8. This causes no small number of difficulties in regard to stability of the overall solver. While $\underline{J}_2$ is not a physical quantity (and thus there is no reason to necessarily expect it to have zero inflow at the corners), the actual value of this is somewhat of an open question. Likewise, the question of what effects forcing it to zero might have is also somewhat of an

open question. It is thought that this will make the solution of the equation more "SP_3-like"; a theory supported by the conclusions of Chapter 6.

4.8. Total Algorithm

4.8.1. Fixed-Source Solver

HPS may be embedded into a group-sweeping and corner point iteration to form a fixed-source solver, shown in Algorithm 3. The fixed-source solver is useful for problems containing a known inhomogeneous source, or for analyses like Method of Manufactured Solutions. After initialization, the decomposition is computed for each energy group. This decomposition will never need to be recomputed for the same problem, unless the diffusion, scattering, or removal cross-sections change. The group sweeping iteration then begins, proceeding until convergence of every Φ_g^{hps} and Φ_g^{crn}; which are the fluxes resolved by the HPS and corner point solvers, respectively. Any metric may be used to determine convergence; here the maximum absolute difference is used with a fixed tolerance.

Within the group sweeping iteration, the scattering source is updated continuously; so as one sweeps down the groups, the source is updated each time. The source is only necessary on the interior points; meaning that the corner points need not be used to compute it. The algorithm shows the iteration proceeding for all groups for simplicity, but once each group has been solved for at least once, only groups involved in up-scattering must be revisited. After the scattering source is computed, the iteration between the HPS and corner point solver begins. The same convergence metric as before is used to determine convergence of Φ.

When the convergence criteria are satisfied, the group-wise flux is stored in Φ^{hps} and Φ^{crn}.

4.8.2. Eigenvalue Solver

The fixed-source solver as defined may then itself be embedded into a larger eigenvalue iteration. The scheme presented here is a straightforward power iteration scheme, with no acceleration technique. Simple accelerations like a Wielandt shift are possible, but omitted for simplicity. As before, the decompositions are first computed on all energy groups, and will never need to be recomputed unless the PDE coefficients change.

Algorithm 3 Fixed-Source Solver Procedure

Given a vector of inhomogeneous sources Q_g
$\Phi_g^{hps}, \Phi_g^{crn} \leftarrow \vec{1} \qquad \forall 1 \leq g \leq G$
for g=1,G **do**
    Compute the HPS Decomposition for group g
end for
while Φ_g^{hps} and Φ_g^{crn} are not converged $\forall g$ **do**
    for g=1,G **do**
        Compute $Q_g^{\text{scat}} = \sum_{g' \neq g} \Sigma_{s,g \leftarrow g'} \Phi_g^{hps}$ ▷ Scattering Source
        while Φ_g^{hps} and Φ_g^{crn} are not converged **do**
            Update Φ_g^{hps} using $Q_g + Q_g^{\text{scat}}$ and Φ_g^{crn} via HPS procedure
            Update Φ_g^{crn} using Φ_g^{hps} via corner point solver
        end while
    end for
end while

The power iterations then begin with a computation of the initial fission source (assuming a flat flux). It only needs to be computed on the interior of the problem; hence the corner points are not used in its computation. With the fission source accounted for, the group sweeping iteration begins, computing and updating the scattering source as in Section 4.8.1. This part of the iteration is fundamentally equivalent to Algorithm 3, including the corner point iterations.

When this fixed-source solver completes, the estimate for the eigenvalue and critical flux are updated by computing the norm of the total flux. Critical flux is then normalized. For this to properly work, the HPS and corner solutions must be recombined or concatenated; shown here using the $\oplus$ symbol. The type of norm used for this purpose is unimportant, so long as it is a true norm and is used both to compute k and normalize the flux. The implementation presented in this dissertation computes total power from Φ^{hps} and Φ^{crn}, and uses this as a normalization factor.

Algorithm 4 Eigenvalue Solver Procedure

$\Phi_g^{hps}, \Phi_g^{crn} \leftarrow \vec{1} \qquad \forall 1 \leq g \leq G$
$k \leftarrow 1$
for g=1,G **do**
    Compute the HPS Decomposition for group g
end for
while Φ_g^{hps}, Φ_g^{crn} and k are not converged $\forall g$ **do**
    for g=1,G **do**
        Compute $Q_g^{\text{fiss}} = \chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'}\Phi_{g'}^{hps}$ ▷ Fission Source
    end for
    while Φ_g^{hps} and Φ_g^{crn} are not converged $\forall g$ **do**
        for g=1,G **do**
            Compute $Q_g^{\text{scat}} = \sum_{g' \neq g} \Sigma_{s,g \leftarrow g'}\Phi_g^{hps}$ ▷ Scattering Source
            while Φ_g^{hps} and Φ_g^{crn} are not converged **do**
                Update Φ_g^{hps} using $Q_g^{\text{scat}} + Q_g^{\text{fiss}}$ and Φ_g^{crn} via HPS procedure
                Update Φ_g^{crn} using Φ_g^{hps} via corner point solver
            end while
        end for
    end while
    $k \leftarrow \|\Phi_g^{hps} \oplus \Phi_g^{crn}\|$
    $\Phi_g^{hps}, \Phi_g^{crn} \leftarrow \frac{\Phi_g^{hps}, \Phi_g^{crn}}{\|\Phi_g^{hps} \oplus \Phi_g^{crn}\|}$
end while

Chapter 5

The Method of Manufactured Solutions

5.1. MMS for Single-Field Differential Equations

The Method of Manufactured Solutions is a well-studied technique for PDE verification. It is useful in instances where true analytic solutions to a PDE are difficult to come by. This makes it a useful tool in multidimensional convergence analyzes where non-trivial solutions are extremely difficult to construct. The basic idea is quite straightforward, and will be illustrated here with the diffusion equation.

One starts with the desired final solution Φ_{MMS}. The necessary source term Q_{MMS} is then computed as

$$D_0 \nabla^2 \Phi_{\text{MMS}}(\vec{x}) + \Sigma_{a0} \Phi_{\text{MMS}}(\vec{x}) = Q_{\text{MMS}}(\vec{x}). \tag{5.1}$$

The source term is then given to the method under analysis to solve. Order of convergence is obtained by comparing the error with the known true solution as the grid is refined.

The output of the solver is a vector of basis coefficients corresponding to the evaluation of the computed solution at the Gauss-Lobatto points within each cell. The L^2 error on the domain may be computed by evaluating the sum of integrals over the cells $\mathcal{D}^c$:

$$\|\phi(\vec{x}) - \Phi_{\text{MMS}}(\vec{x})\|_{L^2(\mathcal{D})} = \sqrt{\sum_c \int_{\mathcal{D}^c} \left(\phi(\vec{x}) - \Phi_{\text{MMS}}(\vec{x})\right)^2 \, d\vec{x}} \tag{5.2}$$

$$\approx \sqrt{\sum_c \sum_k \omega_k \left(\phi(\vec{u}_k) - \Phi_{\text{MMS}}(\vec{u}_k)\right)^2 \left|\frac{\partial \vec{x}}{\partial \vec{u}}\right|}. \tag{5.3}$$

5.2. MMS for Multi-Field Differential Equations

5.2.1. Construction of Source Terms

The moment that multiple fields are added to a differential equation, the complexity of an MMS convergence study increases dramatically. This is because while the reduction of fields into the final solution is known, the partition of the manufactured solution into each field is not known. That is, Φ_{MMS} is known, and it is known to equal $\phi_0(\vec{x}) - 2\phi_2(\vec{x})$. However, this gives zero information about either ϕ_0 or ϕ_2 alone, and so is insufficient to compute source terms.

To sidestep this issue, one must work with the governing equation of the PDE. Recall Equation (1.19), taken here for 1 region and 1 group for simplicity.

$$-D_0\nabla^2\phi_0(\vec{x}) + \Sigma_{a0}\left[\phi_0(\vec{x}) - 2\phi_2(\vec{x})\right] = Q(\vec{x}), \tag{5.4a}$$

$$-D_2\nabla^2\phi_2(\vec{x}) + \left[\Sigma_{a2} + \frac{4}{5}\Sigma_{a0}\right]\phi_2(\vec{x}) = \frac{2}{5}\left[\Sigma_{a0}\phi_0(\vec{x}) - Q(\vec{x})\right]. \tag{5.4b}$$

Manipulating the Equation (5.4a) assuming a known Φ_{MMS}:

$$D_0\nabla^2\left[\Phi_{\text{MMS}}(\vec{x}) + 2\phi_2(\vec{x})\right] + 2\Sigma_{a0}\phi_2(\vec{x}) = \Sigma_{a0}\phi_0(\vec{x}) - Q_{\text{MMS}}(\vec{x}), \tag{5.5}$$

which may be substituted into the RHS of the other equation:

$$\begin{aligned}
-D_2\nabla^2\phi_2(\vec{x}) + \left[\Sigma_{a2} + \frac{4}{5}\Sigma_{a0}\right]\phi_2(\vec{x}) &= \frac{2}{5}\left(D_0\nabla^2\left[\Phi_{\text{MMS}}(\vec{x}) + 2\phi_2(\vec{x})\right] + 2\Sigma_{a0}\phi_2(\vec{x})\right) \\
-D_2\nabla^2\phi_2(\vec{x}) + \Sigma_{a2}\phi_2(\vec{x}) &= \frac{4D_0}{5}\nabla^2\Phi_{\text{MMS}}(\vec{x}) + \frac{4}{5}\nabla^2\Phi_2(\vec{x}) \\
-\left(\frac{4}{5}D_0 + D_2\right)\nabla^2\phi_2(\vec{x}) + \Sigma_{a2}\phi_2(\vec{x}) &= \frac{4D_0}{5}\nabla^2\Phi_{\text{MMS}}(\vec{x}) \\
\nabla^2\phi_2(\vec{x}) - \frac{\Sigma_{a2}}{\frac{4}{5}D_0 + D_2}\phi_2(\vec{x}) &= \frac{4D_0}{4D_0 + 5D_2}\nabla^2\Phi_{\text{MMS}}(\vec{x}).
\end{aligned} \tag{5.6}$$

This is technically an improvement, as we have a well-posed PDE for Φ_2 alone. However, it is impossible to analytically solve in general. The upcoming section will focus on a 1D-extruded flux, such that this PDE reduces to a 1D Ordinary Differential Equation. In Section 5.2.3, we will return to the question of a truly 2D non-separable solution.

5.2.2. Modelling Heterogeneous Problems

While a 1D extruded problem is rather simple, it can be made more interesting by modelling a heterogeneous problem, with material discontinuity. One potential choice of manufactured solution in this case is

$$f_{\text{quad}}(z) = \begin{cases} C^- \left(\frac{z^2}{2a} + x\right) & -a < z < 0 \\ C^+ \left(-\frac{z^2}{2a} + x\right) & 0 < z < a \end{cases} \qquad \text{and} \qquad \begin{matrix} \Phi_{\text{mms}}^{x-\text{quad}}(\vec{x}) = f_{\text{quad}}(x) \\ \Phi_{\text{mms}}^{y-\text{quad}}(\vec{x}) = f_{\text{quad}}(y) \end{matrix}. \tag{5.7}$$

The resulting polynomial source will imply a certain structure of solution. For diffusion, this implication is trivial; the diffusion solution will, of course, be equal to this piecewise quadratic. C^+ and C^- may then be found using the interface condition.

For SP_3, this source instead implies a form of:

$$\phi_0^{\pm}(z) = C_0^{0\pm} + C_1^{0\pm} z + C_2^{0\pm} z^2 + H_0^{0\pm} \cosh(z/L) + H_1^{0\pm} \sinh(z/L), \tag{5.8}$$

$$\phi_2^{\pm}(z) = C_0^{2\pm} + C_1^{2\pm} z + C_2^{2\pm} z^2 + H_0^{2\pm} \cosh(z/L) + H_1^{2\pm} \sinh(z/L). \tag{5.9}$$

where L is the diffusion length $L^2 = D/\Sigma_r$ for the material of interest. One may create a homogeneous system of equations with respect to the coefficients C and H which holds for solutions obeying boundary conditions, interface conditions of ϕ_0 and ϕ_2, and equality of terms such that $\phi_0(z) - 2\phi_2(z) = f_{\text{quad}}(z)$. Any choice of coefficients C and H falling into the null space of this matrix will give rise to a 1D-extruded solution to the PDE.

5.2.3. Non-Separable Problems

Solving for a truly 2D solution is a much more difficult problem. While some analytic solutions exist, they do so typically by way of simplicity or a symmetry reduction to a 1D form. If a true 2D solution is desired, the most practical way to go about it is to define a suitable problem and apply a Fourier method to solve it to arbitrary precision. A problem is defined with even Φ_{MMS} over $[-b, b]^2$ and reflective boundary conditions. The relevant eigenfunctions of ∇^2 on this problem are:

$$b_{ij}(x, y) = \cos(\omega_i x)\cos(\omega_j y) \qquad \text{with} \qquad \omega_k = \frac{2\pi k}{b}. \tag{5.10}$$

If the Laplacian of the manufactured solution also obeys the boundary condition, solving the PDE is trivial. First defining some constants:

$$L^{-2} = \frac{\Sigma_{a2}}{\frac{4}{5}D_0 + D_2} \qquad \text{and} \qquad A_0 = \frac{4D_0}{4D_0 + 5D_2}, \tag{5.11}$$

the solution may be expanded as:

$$\Phi_{\text{MMS}}(x, y) = \sum_{i,j=1}^{N} C_{ij} b_{ij}(x, y). \tag{5.12}$$

This allows the coefficients to be computed:

$$\begin{aligned} -(\omega_i^2 + \omega_j^2)C_{ij} - \frac{C_{ij}}{L^2} &= A_0 \frac{\iint b_{ij}(\vec{x})\nabla^2\Phi_{\text{MMS}}(\vec{x})\, d\vec{x}}{\iint b_{ij}^2(\vec{x})\, d\vec{x}} \\ C_{ij} &= -\frac{A_0}{\omega_i^2 + \omega_j^2 + L^{-2}} \frac{\iint b_{ij}(\vec{x})\nabla^2\Phi_{\text{MMS}}(\vec{x})\, d\vec{x}}{\iint b_{ij}^2(\vec{x})\, d\vec{x}}. \end{aligned} \tag{5.13}$$

While it requires many integral evaluations, this allows the true solution to be computed to an arbitrary level of precision. With enough terms, this approach allows us to study multidimensional spatial convergence accurately. The manufactured source may be obtained by plugging in the now-known forms of Φ_{MMS} and ϕ_2 into Equation (5.4a):

$$Q_{\text{MMS}}(\vec{x}) = -D_0\nabla^2 \left[\Phi_{\text{MMS}}(\vec{x}) - \sum_{i,j} C_{ij} b_{ij}(\vec{x}) \right] + \Sigma_{r0}\Phi_{\text{MMS}}(\vec{x}). \tag{5.14}$$

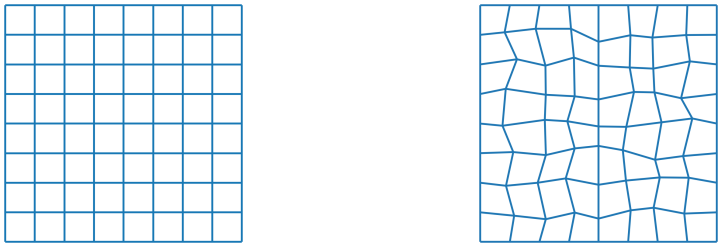
5.3. Numerical Results

5.3.1. Heterogeneous MMS

As mentioned in Section 5.2.2, the solution space for heterogeneous solutions is limited to 1D extrusions. However, while the solution itself may be limited to 1D extrusions, we are free to perturb the mesh in order to make this case decidedly non-trivial. We consider 7-group SP_3 and GSP_3 based on cross-sections obtained from the C5G7 benchmark [39]. The geometry under study has fuel in the positive x half-plane, and moderator in the negative x

half-plane.

The vertices of the mesh describing this problem are perturbed by a parameter proportional to the grid size to simulate how convergence might behave on an unstructured grid. This pseudo-random factor δ satisfies $-\sigma h \leq \delta \leq \sigma h$, where h is the length of a side of the unperturbed element. To guarantee convexity, we must have $0 \leq \sigma < \frac{1}{4}$. An example of this perturbation on an 8×8 grid is depicted in Figure 5.1. Note that for this case, the perturbation normal to the material interface has been zeroed out. This is done to maintain a deterministic boundary and fuel volume. Interfaces in both the x and y directions are used to demonstrate symmetry.



(a) σ=0.0 **(b)** σ=0.25

Figure 5.1: Structured Grid Perturbation

For the manufactured solution itself, we choose Equation (5.7) in both the x and y directions. The kernel evaluation is carried out using Gauss-Lobatto integration as described in Chapter 4. For diffusion, this results in a matrix A defined as:

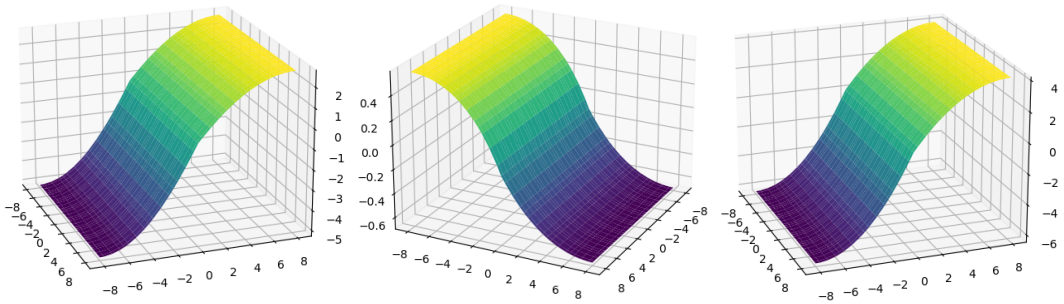
$$A^\tau(\mathcal{I}_i, \mathcal{I})_{\mu\nu} = \sum_{\gamma \in \mathcal{I}_{\text{all}}} \frac{\omega_\gamma}{|J_\gamma|} \left[D^\tau(\vec{\zeta}_{\gamma\mu} \cdot \vec{\zeta}_{\gamma\nu}) + \Sigma_r^\tau \delta_{\mu\gamma}\delta_{\nu\gamma} \right], \qquad \vec{\zeta}_{\gamma\mu} = (J_\gamma^{-T} \left[\nabla_{\vec{u}} b_\mu \right]_{\vec{u}=\vec{u}_\gamma}). \tag{5.15}$$

For SP_3/GSP_3, both use the following prescription for A:

$$A^\tau(\mathcal{I}_i, \mathcal{I})_{\mu\nu} = \sum_{\gamma \in \mathcal{I}_{\text{all}}} \frac{\omega_\gamma}{|J_\gamma|} \begin{bmatrix} D_0^\tau(\vec{\zeta}_{\gamma\mu} \cdot \vec{\zeta}_{\gamma\nu}) + \Sigma_{r0}^\tau \delta_{\mu\gamma}\delta_{\nu\gamma} & -2\Sigma_{r0}^\tau \delta_{\mu\gamma}\delta_{\nu\gamma} \\ -\frac{2}{5}\Sigma_{r0}^\tau \delta_{\mu\gamma}\delta_{\nu\gamma} & D_2^\tau(\vec{\zeta}_{\gamma\mu} \cdot \vec{\zeta}_{\gamma\nu}) + \left(\Sigma_{r2}^\tau + \frac{4}{5}\Sigma_{r0}^\tau\right) \delta_{\mu\gamma}\delta_{\nu\gamma} \end{bmatrix}. \tag{5.16}$$

These two sets of equations are distinguished by their respective formulations of the operator $\mathcal{N}$; for more details, see Section 4.6.

Results are shown in Figure 5.3, which may at first glance seem rather confusing. Here,



(a) ϕ_0 (b) ϕ_2 (c) ϕ

Figure 5.2: MMS Quadratic Function

diffusion is computing the correct solution exactly at every grid spacing; while SP_3 and GSP_3 have a much larger error, converging at 4th order. Diffusion is able to capture the exact solution because the quadratic manufactured solution falls perfectly within the span of the basis. The same cannot be said for SP_3 and GSP_3, which, as shown in Equation (5.8), have hyperbolic trigonometric terms. These terms vanish when the two moments are combined into a scalar flux; but they still incur a discretization error, which manifests as convergence despite the scalar flux falling in the span of the bases separately.

Aside from falling one order short from what is expected from the Bramble-Hilbert lemma (Equation (2.35)), the convergence is otherwise very well-behaved. For the x and y directed fluxes, the errors are entirely indistinguishable, as their plots completely overlap. This is expected, but shows that the solver is preserving rotational symmetry. Additionally, the convergences depend very little on the parameter of mesh perturbation σ.

To remedy the shortfall in convergence, the kernel may be adjusted to use a more intensive numerical integration scheme to compute the integrals of Equation (4.14) and the SP_3 analogue. For our purposes, we pass this task off to QUADPACK [53]. These integration schemes are able to recover detail that the approximate one does not; primarily because they will evaluate the Jacobian (and basis functions) many more times and at many different points than the standard Gauss-Lobatto scheme. Naturally, this detailed integration scheme is much more costly. However, it does not impact the scaling of the method; since it is scales linearly with the number of leaf nodes. In either case, when the more advanced scheme is used, the maximum convergence rate of fifth order is obtained, shown in Figure 5.4.

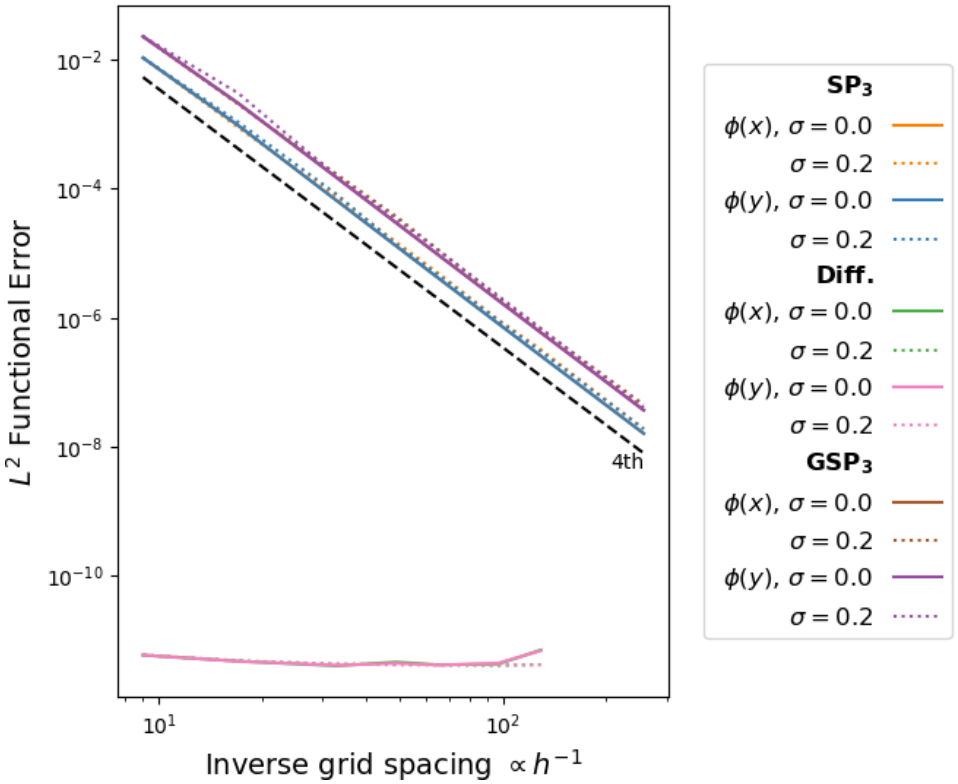


Figure 5.3: Quadratic MMS Results with Gauss-Lobatto Integration

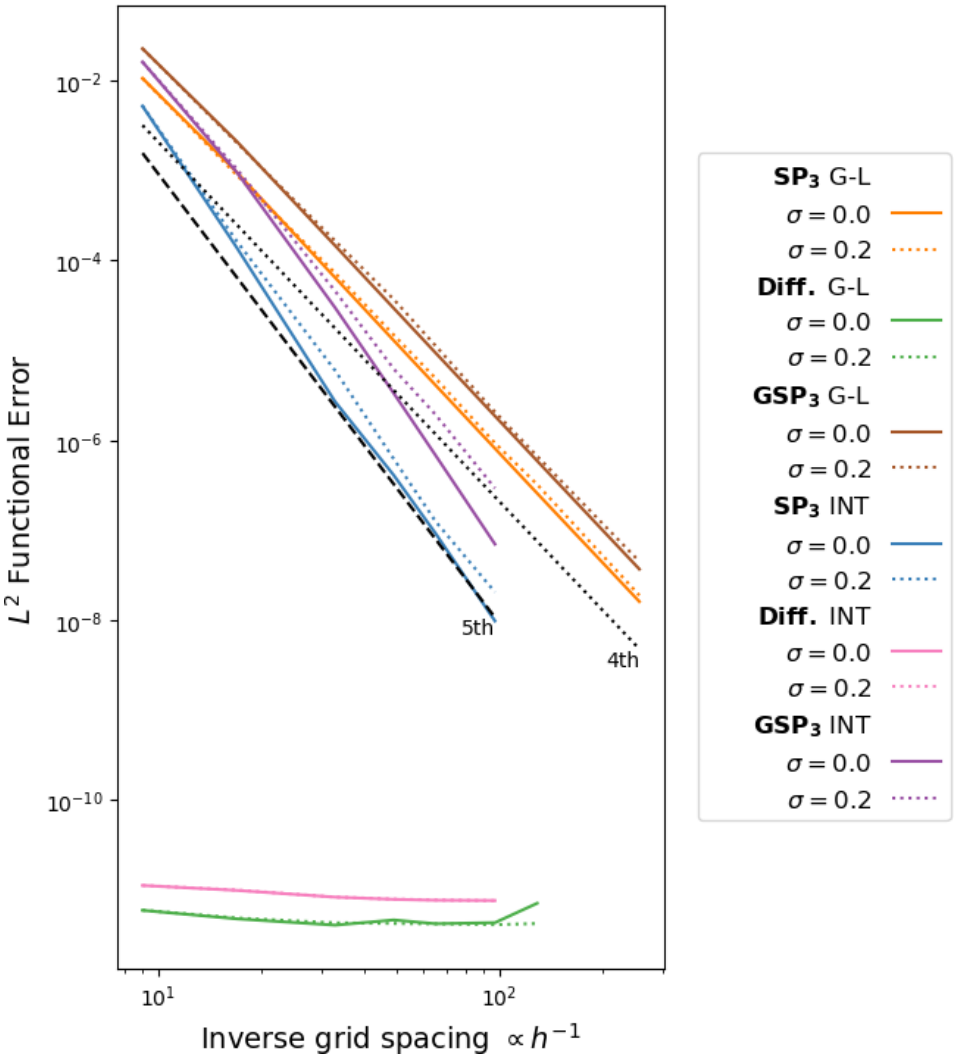


Figure 5.4: Quadratic MMS Results with QUADPACK Integration

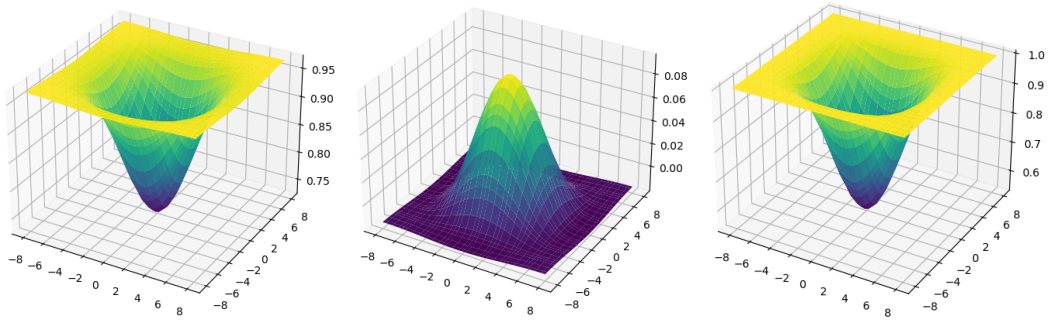
5.3.2. Non-separable MMS

In the non-separable case we consider a homogeneous medium containing a fuel material. Within this material, we force a "Quadratic Cosine" solution:

$$\Phi^{\text{cosq}}_{\text{MMS}}(\vec{x}) = \cos\left(\left[\left(\frac{x}{a}\right)^2 - 1\right]\left[\left(\frac{y}{a}\right)^2 - 1\right]\right), \tag{5.17}$$

defined over the same domain as the quadratic one, $\vec{x} \in [-b, b]^2$. This rather convoluted function was chosen because both it and its second derivative satisfy a reflective boundary condition. 2D Fourier moments C_{ij} (see Equation (5.13)) were taken for all $i + j < 128$, where the largest magnitude moment was less than 1.5×10^{-9}.

The same mesh perturbations shown in Figure 5.1 are used, as is the Gauss-Lobatto integration scheme.



(a) ϕ_0 **(b)** ϕ_2 **(c)** ϕ

Figure 5.5: MMS Quadratic Cosine Function

As before, Figure 5.7 achieves only 4th order convergence. Using QUADPACK to perform a numeric integration in the kernel, as in the previous case, recovers the full expected order of convergence. The results are plotted on Figures 5.6 and 5.7. In general, they mirror all the important properties of the extruded case; with the caveat that diffusion is now converging, as the solution is no longer as simple.

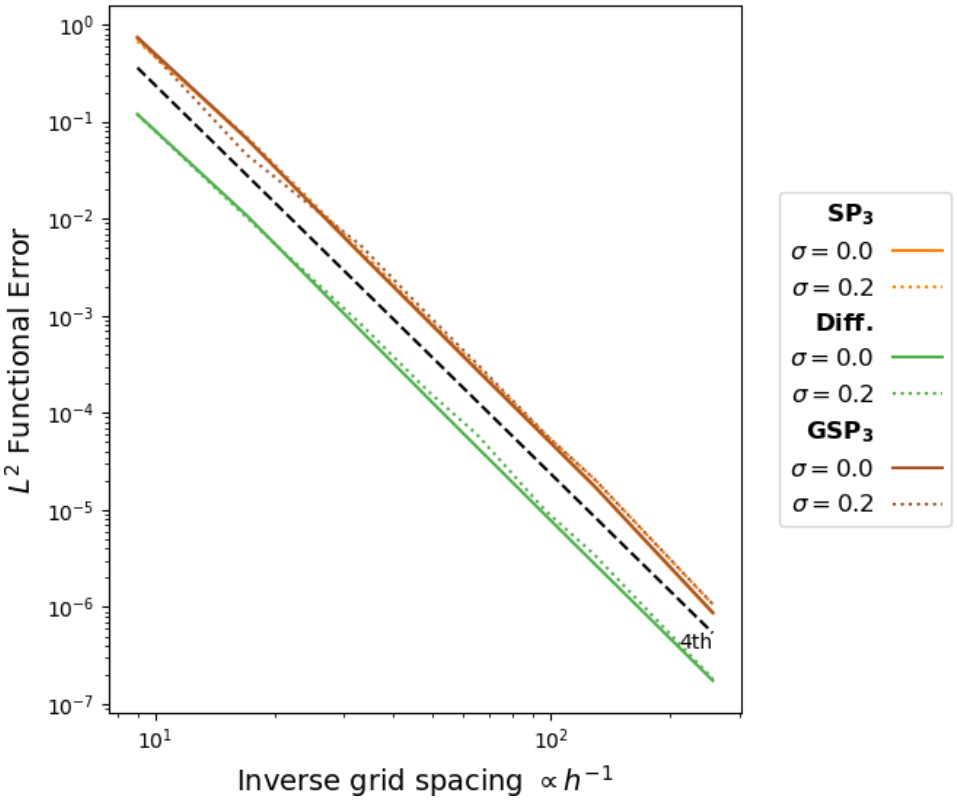


Figure 5.6: Quadratic Cosine MMS Results with Gauss-Lobatto Integration

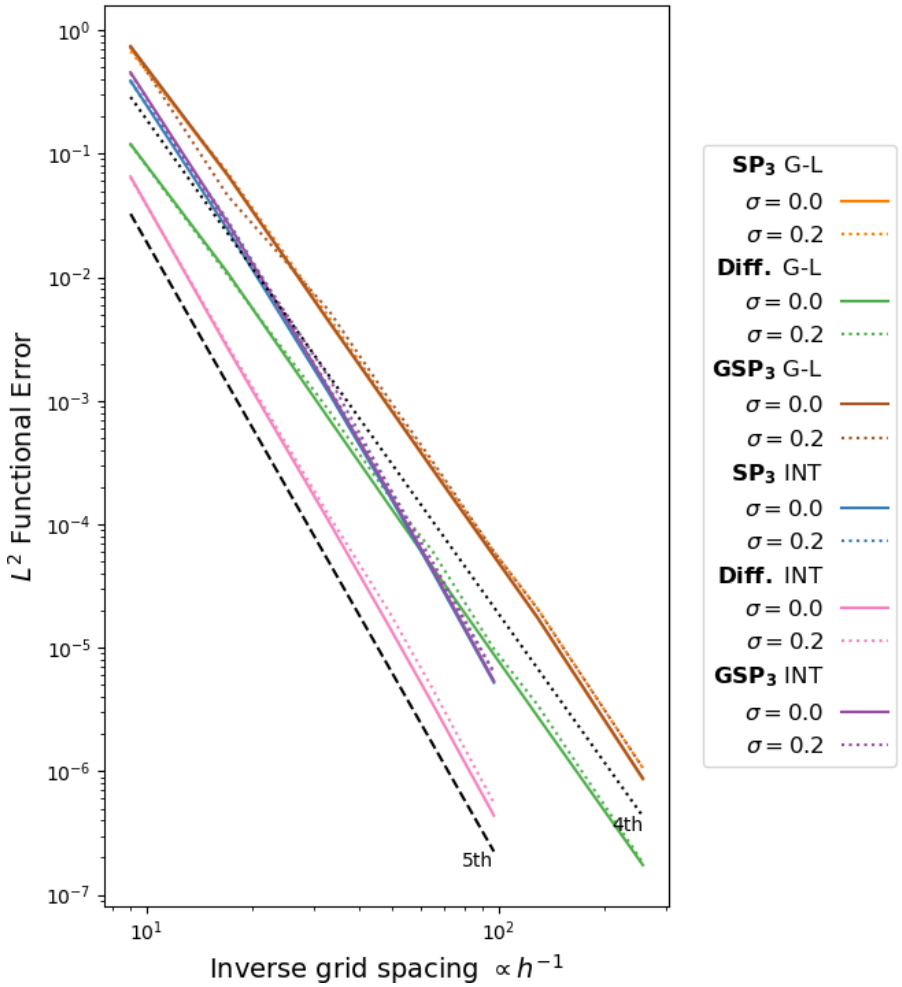


Figure 5.7: Quadratic Cosine MMS Results with QUADPACK Integration

5.3.3. Conclusions

These results are highly encouraging and demonstrate the proper functioning of the solver implementation. While the 1D extruded, non-perturbed GSP_3 cases reduce to SP_3, the other cases do not do so in general; as the tangential derivative terms of Equation (1.27) are non-trivial in every other case. Additionally, these results show that the HPS solver is capable of recovering the full order of convergence expected from a polynomial expansion method via the Bramble-Hilbert lemma Equation (2.35). While the simpler and faster numerical integration is expected to be sufficient for most cases, it would be concerning if the limit could not be recovered.

This shows we have successfully completed the first part of our overall goal. The study outlined here demonstrates quite conclusively that we have a working high-order, multidimensional discretization of the SP_3 and GSP_3. Furthermore, this is solved via our targeted fast method, HPS; though the existing implementation is suboptimal. This method enforces continuity relations in a strong manner, and has been analyzed with a novel MMS technique for multi-field equations not previously observed in the literature. On top of all of these achievements, the degradation in performance for working on deformed grids is incredibly slight; which indicates that this approach could readily extend to an unstructured mesh.

Chapter 6

Numerical Results

6.1. Testing Methodology

6.1.1. Benchmark Specification

All tests in this section are derived from the C5G7–2D benchmark [39], a widely available test frequently used to examine many types of neutronics solvers. The discretizations analyzed range from the simplified spherical harmonic approximations discussed here, to S_N, Method of Characteristics, and even Monte Carlo codes. C5G7 specifies a 7-group neutron structure with energy boundaries shown in Table 6.1

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
Upper [eV]	2.0E+07	1.0E+06	5.0E+05	1.0E+03	1.0E+02	10	0.0635
Lower [eV]	1.0E+06	5.0E+05	1.0E+03	1.0E+02	10	0.0636	

Table 6.1: C5G7 Energy Bounds

The geometry of the benchmark is quite simple. Pin cells exist on a lattice of 1.26 cm. All cells are either reflector cells (empty) or contain one of a fuel, control, or instrumentation rod. Regardless of type, all are assumed to have the same radius of 0.54 cm and are fully homogeneous. Fuel rods vary and include Mixed-Oxide (MOX) fuel of varying plutonium fraction; all at 3.3% enrichment.

6.1.2. Problem Mesh

Sub-pin meshes are utilized in this analysis. These are constructed using an area-preserving quadrilateral mesh. This mesh remains structured, though it will obviously be deformed to accommodate the cylindrical pins. Several refinement levels are considered, constructed via repeated bisection of mesh cells.

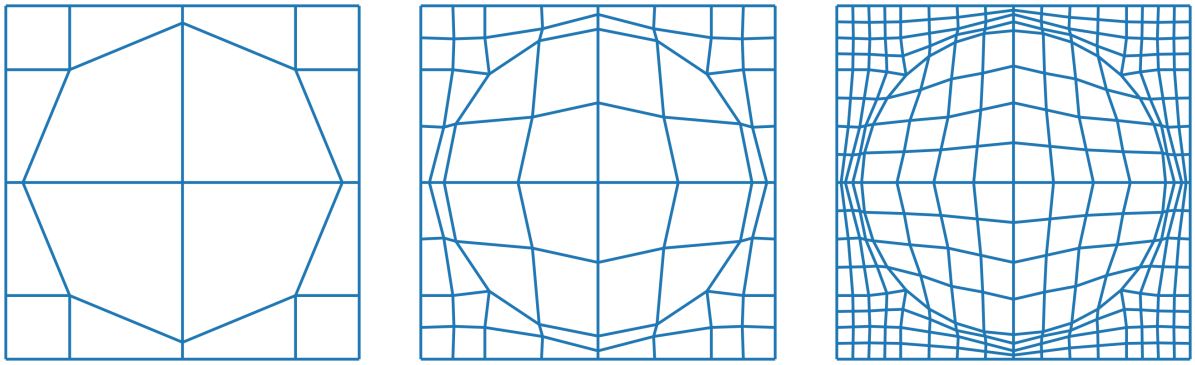


Figure 6.1: Pincell Mesh at Various Refinement Levels.

One may expect this highly-refined mesh at the boundary of the fuel element to pose problems for numerical methods. However, they aid greatly in resolving the sharp flux gradient at the boundary, and attempts to prune these elements have detrimental effects on the spatial convergence.

However, this mesh is used only for the diffusion and SP_3 results. While many attempts were made to incorporate GSP_3 results, stability issues have prevented the acquisition of consistent eigenvalues and critical fluxes. These difficulties arise from the three-way combination of high-ordered derivatives in the interface condition, high flux gradients, and corner point iteration. Regretfully, a suitable stabilization scheme has yet to be found. However, GSP_3 does perform adequately on pin-homogenized lattices; a topic discussed in Section 6.4.

6.1.3. Quantities of Interest

Results are compared via the computed eigenvalue of the system and relative pin powers, when available. In the case of the HPS solver, these quantities are found via an unsophisticated power iteration scheme with no acceleration; discussed in Section 4.8.2. The performance of this can be improved via Wielandt shift or other algorithms; but these are beyond the scope of this work.

Spatial convergence is analyzed as a function of the grid refinement; depicted in Figure 6.1. The reference criticalities and pin powers is a highly-converged MOC solution obtained using the deterministic code MPACT [31].

6.2. Fuel Pin Tests

Before moving into multi-pin cases, a fuel pin test is performed and compared to the MOC reference. These utilize the UO_2 fuel material at 3.3% enrichment and moderator cross-sections, with reflective boundary conditions. A qualitative plot depicting the computed shape of the critical power is given in Figure 6.2

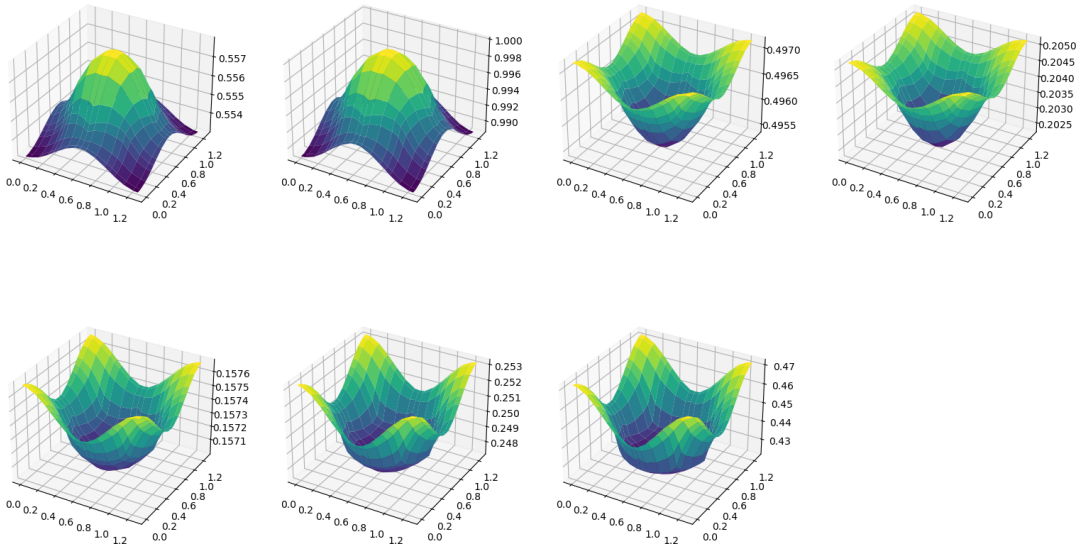


Figure 6.2: Critical Flux for UO_2 Fuel Pin

Subdivisions per Side of Pin Cell	4	8	16	32	Reference
Diffusion	1.32583	1.32577	1.32576	1.32575	1.32586
Δk_{eff} [pcm]	-3	-9	-10	-11	
SP_3	1.32370	1.32362	1.32358	1.32357	1.32586
Δk_{eff} [pcm]	-216	-224	-228	-229	

Table 6.2: Criticality Results for Single UO_2 Fuel Pin

In this instance, diffusion produces startlingly accurate results in the eigenvalue. While for this case in isolation, it seems like a very good result, there is reason to be cautious. The

eigenvalue alone is not a particularly robust indicator of the solution quality, and it is possible to arrive at a good figure for criticality with a poor flux shape. We will see in Section 6.3 that there is more to look at here than just the eigenvalue.

The SP_3 eigenvalue results, by comparison are notably worse. This is not altogether uncharacteristic behavior for SP_3 as applied to a sub-pin mesh [42]. While SP_3 is "more correct" than diffusion in this regime, as more variables in angle are used; this does not resolve the issue of the underlying approximation obeying the same limitations. These methods in general are known to be inaccurate within a few mean free paths of any material discontinuities; and excessive spatial refinements can interact with this inaccuracy to degrade the quality of solutions further.

However, even here the increase in error is relatively slight. A 230 pcm error off of an MOC method is still relatively accurate when discussing methods based on spherical harmonic expansions.

6.3. Mini Lattice Tests

Three small lattices are studied for comparison with the MOC reference.

These are a 3×3 fuel lattice, 3×3 fuel with 8.7% plutonium fraction MOX center, and a 3×3 with control rod center. The 3×3 fuel lattice is naturally identical to the nominal fuel radius of the previous section, since they are both infinitely repeated. Identical results are expected and obtained.

The control rod case is expected to break down somewhat for any of the spherical harmonic transport simplifications, owing to the strong directionality of the flux near the absorber.

6.3.1. Fuel Lattice

The 3×3 fuel lattice is an unremarkable simulation compared to the 1×1 fuel pin. The flux shape (Figure 6.3) and convergence results (Table 6.3) are identical to the 1D counterpart, which is a strict requirement of a valid discretization scheme. Given that this is a quasi-direct method, the eigenvalue being exactly equal is expected.

One property of note is that as in the single-pin case, the eigenvalues converge quite quickly with spatial refinement in this scheme. The difference from coarse to fine is under

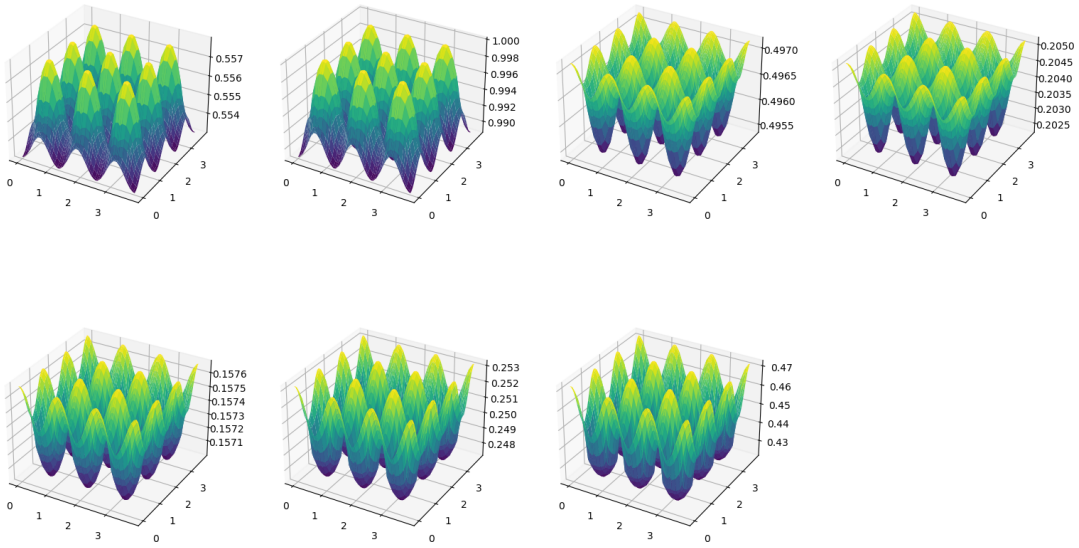


Figure 6.3: Critical Flux for UO_2 Lattice

Subdivisions per Side of Pin Cell	4	8	16	32	Reference
Diffusion	1.32583	1.32577	1.32576	1.32575	1.32586
Δk_{eff} [pcm]	-3	-9	-10	-11	
SP_3	1.32370	1.32362	1.32358	1.32357	1.32586
Δk_{eff} [pcm]	-216	-224	-228	-229	

Table 6.3: Criticality Results for UO_2 Fuel Lattice

20 pcm for both the Diffusion and SP_3 equations.

6.3.2. MOX Lattice

We examine next the critical flux shape for the UO_2-MOX lattice, depicted in Figure 6.4. The form of the solution includes all the normal features expected with this geometry. In particular, a sharp peak in fast neutron flux localized around the MOX fuel pin, and corresponding depression in the thermal flux. The physical cause of this is the MOX pin's comparatively high fission cross-section at thermal energies. The difference in eigenvalue results between diffusion and SP_3 is less stark here; but still significant. Again, the SP_3 eigenvalue appears to indicate worse results.

However, as mentioned previously, this eigenvalue alone is not enough to assess solution quality. If the relative pin powers are compared with the reference solution, a percentage

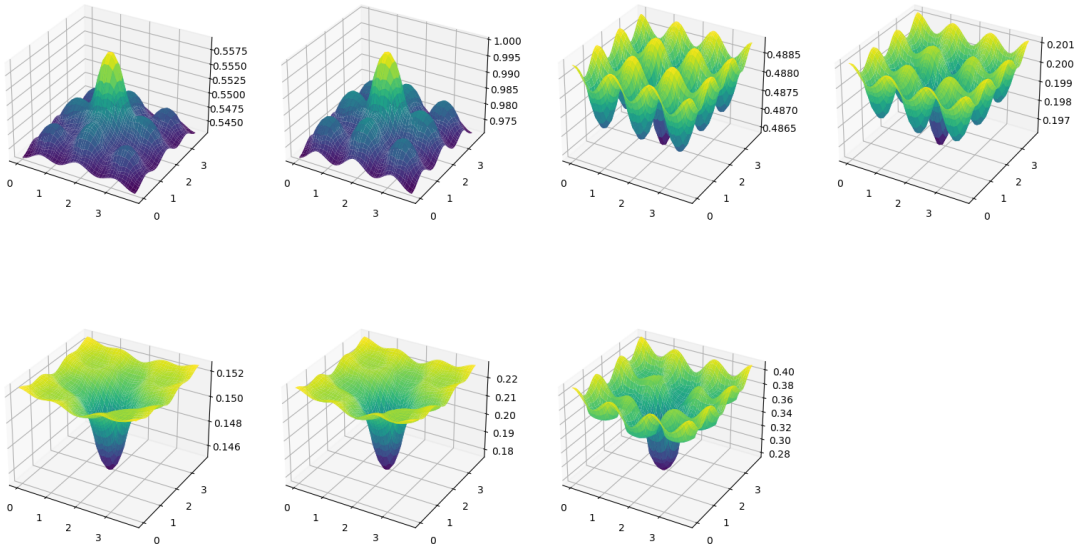
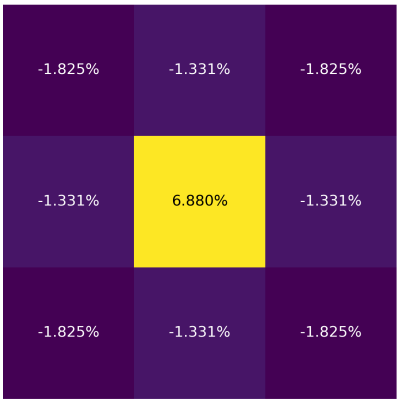


Figure 6.4: Critical Flux for MOX Pin Surrounded by UO_2

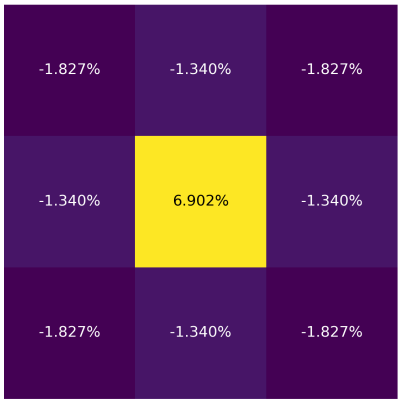
Subdivisions per side of Pin Cell	4	8	16	32	Reference
Diffusion	1.29616	1.29611	1.29610	1.29609	1.29949
Δk_{eff} [pcm]	-333	-338	-339	-340	
SP_3	1.29526	1.29519	1.29517	1.29516	1.29949
Δk_{eff} [pcm]	-423	-430	-432	-433	

Table 6.4: Criticality Results for UO_2-MOX Lattice

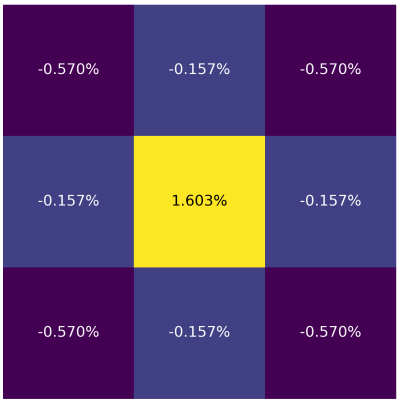
error may be obtained. Figures 6.5a and 6.5b show the errors in pin power relative to MPACT for 2 and 32 subdivisions per side of a pincell, respectively. As in the eigenvalues, convergence is slight, and very slightly worsens relative to the reference. Errors in the central pin are quite large, nearing 7%. The SP_3 solutions for these same numbers of subdivisions, depicted in Figures 6.5c and 6.5d, are greatly improved, resolving the power of the central pin to less than 2% with a similar improvement in the surrounding pins. This fact is a very strong indicator that the diffusion eigenvalue looks as good as it does due to a beneficial cancellation of errors; and not because it is actually obtaining a better solution.



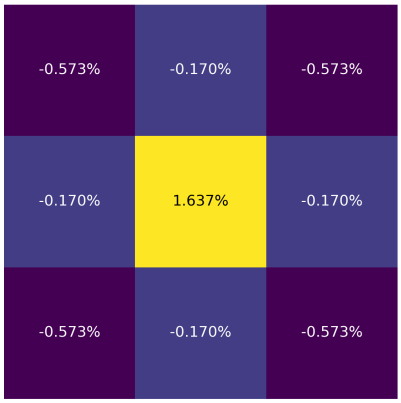
(a) Diffusion 2×2 Subdivisions



(b) Diffusion 32×32 Subdivisions



(c) SP_3 2×2 Subdivisions



(d) SP_3 32×32 Subdivisions

Figure 6.5: MOX Mini-Lattice Pin Powers

6.3.3. Control Rod Lattice

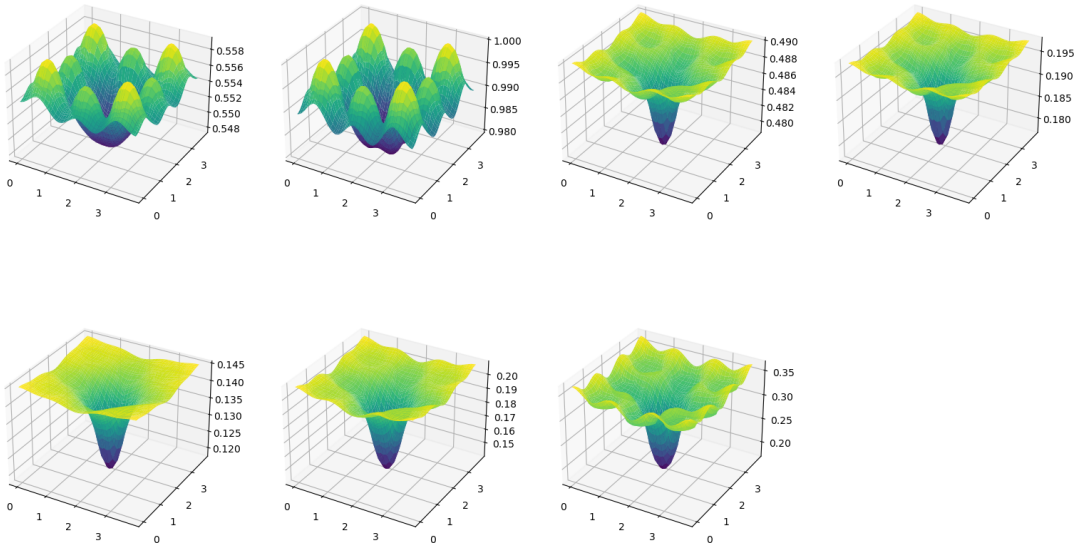


Figure 6.6: Critical Flux for Control Rod Surrounded by Fuel

Moving on to the critical flux shape of the UO_2-Control lattice (Figure 6.6), we again see the expected qualitative features. A deep dip in the flux for the lower energy groups is readily apparent at the central lattice point; along with a peaking of fluxes at the corner lattice sites.

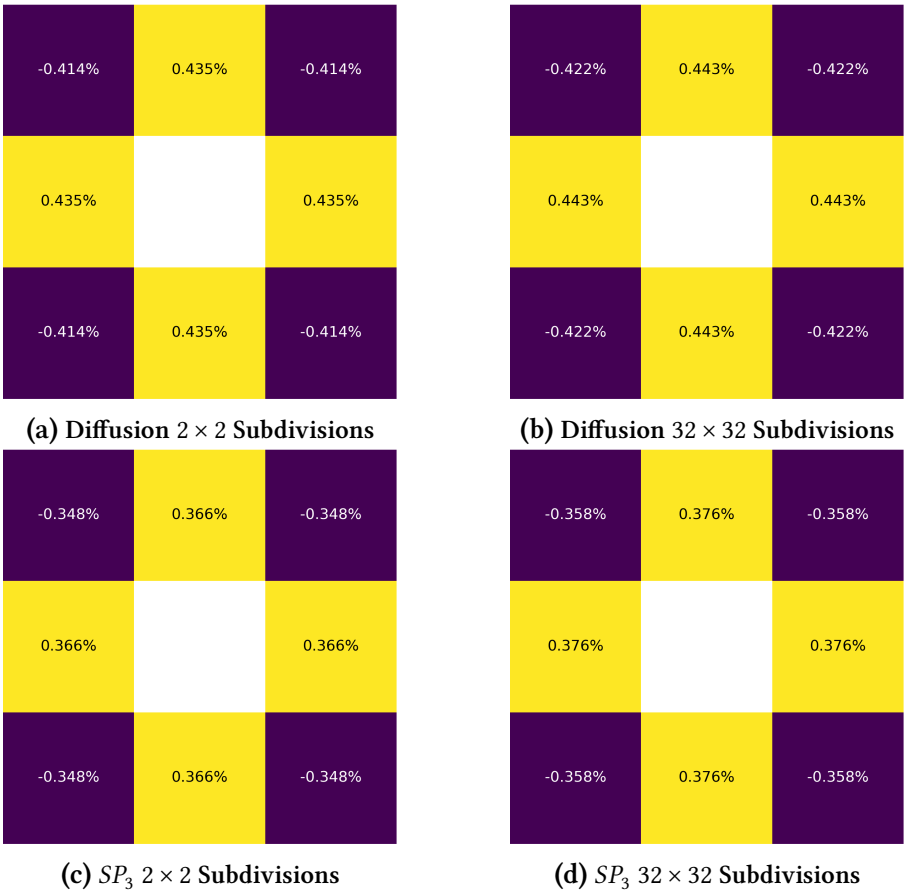
The eigenvalue results for this case, shown in Table 6.5, are quite poor; which is somewhat expected due to the tendency of the simplified spherical harmonics methods to break down near strong absorbers. The higher-order SP_3 method is generally better able to cope with these scenarios, which is likely why the SP_3 eigenvalue is at last more accurate than diffusion.

Subdivisions per Side of Pin Cell	4	8	16	32	Reference
Diffusion	0.84855	0.84863	0.84868	0.84870	0.89505
Δk_{eff} [pcm]	-4650	-4642	-4637	-4635	
SP_3	0.87541	0.87550	0.87554	0.87556	0.89505
Δk_{eff} [pcm]	-1964	-1955	-1951	-1949	

Table 6.5: Criticality Results for UO_2-Control Rod Lattice

Examining the relative pin powers shows that the available errors are quite small. This strongly suggests that the majority of the error in the flux shape for both methods occurs in

the central absorber; which is unsurprising, as this is where the directional peaking of the angular flux will be the largest.



(a) Diffusion 2×2 Subdivisions **(b) Diffusion 32×32 Subdivisions**

(c) SP_3 2×2 Subdivisions **(d) SP_3 32×32 Subdivisions**

Figure 6.7: Control Rod Mini-Lattice Pin Powers

6.4. Pin-Homogenized Lattice Tests

The GSP_3 method has at the time of this writing one primary implementation known as the Generalized Transverse Integration Nodal (GTIN) method. This method has been benchmarked by Chao et al. in a paper covering several different geometries [14]. One of these geometries (Problem 2) is a pin-homogenized lattice with Gadolinium integral fuel burnable absorber pins, in a 17×17 lattice depicted in Figure 6.8. Cross-section data for this 2-group problem is available in the original study [14].

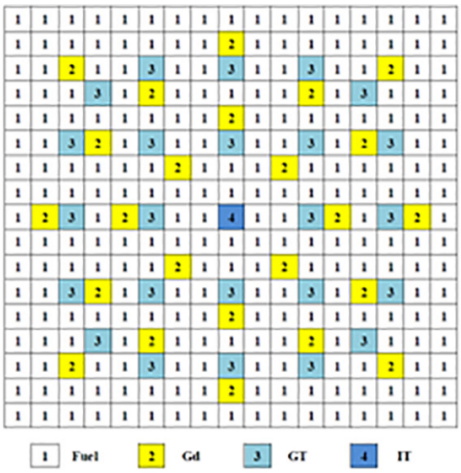


Figure 6.8: Full Homogenized Lattice Test Case

This case specifically is chosen due to the large differences observed by Chao et al. between the diffusion, SP_3, and GSP_3 method implemented in GTIN, termed $GSP_3^{(0)}$ by Chao. The superscript in this notation denotes a treatment of anisotropic scattering; and is identical to the implementation of GSP_3 as described in this dissertation. This pin-homogenized assembly uses a regular mesh, with 1, 2, 4, or 8 subdivisions. It should be noted that the homogenized material does not change as the mesh is refined; meaning the results are not representative of the underlying geometry. They are presented in order to analyze the spatial convergence of this HPS method. Criticality values and the MOC reference [14] are shown in Table 6.6.

The results corresponding to 1 region per pincell agree very well with the analysis of GTIN. There are minor improvements in accuracy which may be related to eliminating the transverse leakage approximation. However, as this grid is refined, the GSP_3 solution appears

Subdivisions per Side of Pin Cell	1	2	4	8	Reference
Diffusion	1.17835	1.17767	1.17756	1.17755	1.18874
Δk_{eff} [pcm]	-1039	-1107	-1118	-1119	
SP_3	1.18775	1.18663	1.18639	1.18637	1.18874
Δk_{eff} [pcm]	-99	-211	-235	-237	
GSP_3	1.18899	1.18717	1.18658	1.18642	1.18874
Δk_{eff} [pcm]	25	-157	-216	-232	

Table 6.6: Criticality Results for Homogenized Gad. Lattice

to converge to the same value as SP_3. The reason for this is currently unknown, but could be related to the current method of handling the corner point balance; as the effects of this would increase as spatial resolution increases.

Pin power comparison results for this case are presented below (reference data obtained from Chao et al.) Tabulated in Table 6.7 are the maximum pin power errors between the MOC reference and each method, given in %. These are given for the same spatial refinements as in Table 6.6; 1, 2, 4, and 8 subdivisions per side of a pincell. Here the story for maximum pin

	1	2	4	8
Diffusion	3.322	3.339	3.349	3.350
SP_3	0.917	0.912	0.930	0.932
GSP_3	0.682	0.790	0.888	0.920

Table 6.7: Pin Power Maximum Percentage Error

power error is much the same as for the criticalities. GSP_3 initially provides a much better error than SP_3, which quickly decays as the mesh is refined; both of which give significantly improved errors relative to diffusion. To reiterate, while the precise cause is undetermined, it is believed that this is due to effects from the corner point solver becoming more prominent on high-resolution meshes.

6.5. Conclusions

These mesh refinement studies of the high-order discretization and solver demonstrate a clear and rapid convergence of both the eigenvalue and critical flux to a specific solution. While the results of this convergence are not fully consistent with the MOC reference, the

manner in which the discrepancy behaves is consistent with the behavior of the simplified spherical harmonics approximations. That is, the error is maximized when the angular flux becomes very sharply peaked. So while the magnitudes of these errors are sometimes at the limits of what could be considered an acceptable solution, this still demonstrates a success in the solver's ability to converge to the solution of a complex 2D PDE.

While the GSP_3 implementation could not converge on this sharply varying flux using the pin-resolved mesh, it was analyzed on a pin-homogenized assembly case with reflective boundary conditions. This analysis showed the same rapid convergence as the previous case; but rather than an anomalously accurate diffusion solution, there was a clear progression in accuracy in the eigenvalue from diffusion to SP_3 to GSP_3. As the mesh is refined, however, the GSP_3 eigenvalue solution approaches the same value as SP_3. Pin powers, as studied using the maximum percentage difference from the reference, followed an identical trend. This is believed to highlight the need for a more rigorous corner point treatment.

Chapter 7

High Performance HPS

While the original aim of this dissertation was to achieve a highly performant implementation, this goal turned out to be too lofty. In lieu of this, we provide a description of the qualitative aspects of several performance critical steps in the algorithm. These topics form the basis of future extensions to this work; should the opportunity arise.

Supplementing this discussion, an approximate performance analysis is reproduced within this section to estimate expected performance gains. This includes a complexity study as a modified reproduction of the work of Hao and Martinsson [27], as well as an original study of the compressibility of the operators relating specifically to the diffusion and SP_3/GSP_3 systems of equations.

7.1. Operator Computation and Storage

7.1.1. Differential Operators

Among the most time-consuming parts of this research code implementation are the routines which compute the differential operators to be used in the definitions of $\mathcal{N}$. This is especially pronounced in GSP_N, which needs a full set of linear operators discussed at length in Section 4.6 and Appendix B. The most difficult to construct are any of those operators which involve $\hat{n}$. These must be precomputed for a set of normal vectors corresponding to normal, tangent, or corner-directed unit vectors.

For expediency, these operators may be generated as a pair of tensors; one of rank-k and one of rank-$(k+2)$, where k is the order of the directional derivative being discretized. For

example, $(\hat{n} \cdot \nabla)$ can be stored as a list of rank-1 tensors (the unit vectors of interest) and a rank-3 tensor which is contracted with the desired unit vector when necessary. This is intuitive for the first-order derivatives, but less so for higher-ordered versions.

For example, the operator $(\hat{n} \cdot \nabla)^3 f$ necessitates a rank-3 and rank-5 tensor, defined as:

$$\left[(\hat{n} \cdot \nabla)^3\right]_{\mu\nu} = \eta_{ijk}\rho_{ijk\mu\nu} \qquad \eta_{ijk} = \hat{n}_i\hat{n}_j\hat{n}_k \qquad \rho_{ijk\mu\nu} = \left[\left(\mathbb{D}_3^x\right)_{ijk}\right]_{\mu\nu}, \tag{7.1}$$

where the square brackets $[\cdot]$ denote a discretization of the differential operators to the basis equations used. This increases the memory footprint, but only in accordance with the number of leaf nodes being simultaneously processed. Once a leaf node has been processed and the HPS operators (such as S and T) are created, these constructions may be safely deleted.

7.1.2. Leaf Storage

Storage of the leaf-level operators of the interpolative decomposition can be quite expensive, as alluded to in Section 3.2.2. These operators include S, R, G, and H defined in Section 3.2.2; whose sizes on the leaves (for SP_3 and GSP_3) are reproduced here:

$$S^\tau : 18 \times 24, \qquad R^\tau : 18 \times 8, \qquad G^\tau : 24 \times 8, \qquad H^\tau : 24 \times 18. \tag{7.2}$$

For diffusion, one may halve the sizes in both dimensions. While individually these are very small operators, a simple implementation will store the 4 operators for each leaf as implied by Algorithm 1. This amounts to just short of 10kB per group per mesh cell, which can increase to an exorbitant memory cost for large or highly detailed problems. If these memory requirements become too great, the leaf operators may be deleted during factorization and rebuilt in the solve stage. This adds to the duration of the solve stage, but does not increase the scaling; since it is only for the leaves that these must be regenerated.

7.2. Mesh Partitioning

The partitioning of the mesh has the potential to dramatically affect the runtime of the solver. For this method, the two most important metrics for determining what constitutes a "good" mesh partition are

1. The fullness of the tree representing hierarchical domain decomposition.

2. The "numerical area" of the partitions in this hierarchy.

Maximizing the fullness of the tree may be thought of as a heuristic comprised of minimizing the tree height, and maximizing the tree balance. Here, the "numerical area" may be thought of as the number of unknowns (not the physical area) that exist on the partition separating two subdomains of the problem.

Furthermore, if any sort of parallelism is implemented in the node calculations, such a structure may also limit the available concurrency. Simple schemes may decompose the problem at a given height of the mesh partitioning tree into a set of "embarrassingly parallel" problems; meaning that lower tree heights better enable parallelism. More advanced schemes with task queues and dependencies will find that the dependency graph is isomorphic to the mesh partitioning tree, and so arrive at the same end result. However, the heights of the tree may not be reduced arbitrarily without consequences, as doing so runs into the other limitation: the numerical area of the interfaces.

This numerical area may have a strong impact on performance, for reasons which are discussed in Section 7.5.1. Essentially, at every merge operation conducted in Algorithm 1, a matrix must be inverted with a size corresponding to this numerical area. The more unknowns there are, the bigger this matrix becomes, and the performance may suffer. The numerical areas of partitions in a problem depend strongly on the partitioning method used. Currently, a standalone tool included in the `METIS` libraries known as `mpmetis` is used. This tool very effectively partitions a mesh into distinct subdomains regardless of the underlying physical shape; and the process is easily generalized to an unstructured mesh format.

`mpmetis` is configured to utilize recursive bisection as the partitioning method. This strikes a good balance between the numerical area connecting partitions and overall tree balance. The reason a bisection is chosen rather than splitting the region into 4, 8, or more regions is discussed briefly at the end of Section 3.1.2; but it effectively comes down to the fact that this increases the numerical area of the partitions greatly. In theory, it also exposes more parallelism; but more work is needed to determine if allowing this higher degree of concurrency is worthwhile. Regardless of the underlying partitioning scheme the impacts of the potentially large interfaces can be mitigated through structured linear algebra. This aspect of the algorithm is discussed next in Section 7.3.

7.3. Compressible Linear Algebra

The efficient use of a fast direct method demands that the underlying matrices, representing compressible operators within the problem, are efficiently stored and utilized. If stored or used naïvely, this leads to a breakdown of the 'fast' nature of the method, requiring huge amounts of memory and yielding slow performance. This is, in fact, a large reason why direct methods are unused in most nuclear applications; simple approaches to discretize the problem yield intractable numerical problems even for relatively simple configurations.

These compressible operators discussed above are not sparse in structure, but they are sparse in information. Revealing this information sparsity may be done by storing the operator as a rank-structured matrix. One way to do this is by computing the Singular Value Decomposition (SVD) and truncating all singular values smaller than a specified tolerance; but this is an extremely expensive and wasteful approach. A more effective way is to use a partitioned matrix representation that divides up the underlying operator in such a way that separates the low-rank blocks from the dense ones. It can then store the low-rank blocks in such a way that the operator does not need to be re-decomposed every time we wish to work with it.

As a class of matrix representations, these enable the methods discussed in Section 3.2 by allowing the fast multiplication and inversion techniques which lend HPS its excellent scaling. In this way, the solver can distinguish which information is significant, and therefore must be computed; and which information is below some tolerance, and can be safely discarded. This is a clear advantage when compared to conventional direct solvers which will compute all couplings even if they only affect the original solution by an amount smaller than machine precision.

7.3.1. HODLR Matrices

Hierarchical Off-Diagonal Low-Rank (HODLR) matrices are the most prescribed rank-structured format in that they are fairly simple and strictly defined. In a HODLR matrix, the matrix A is represented as a 2×2 block matrix. The off-diagonal blocks are represented by the outer product of a number $k \ll n/2$ singular vectors, where n is the matrix size. The on-diagonal blocks are represented as HODLR matrices themselves; or dense matrices if they are sufficiently small. Typically, k is kept constant for all matrices in the recursive structure.

These matrices have been extensively studied [6]; the important characteristics will be

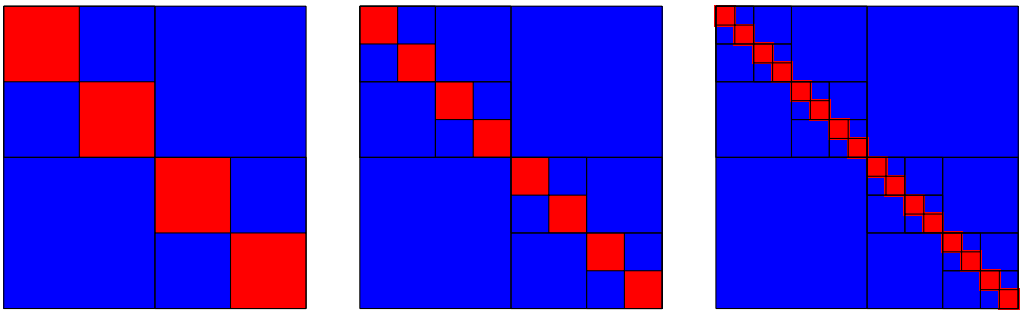


Figure 7.1: Depiction of HODLR matrix representation.

reproduced here, but the derivations are not included. Such a format will require

$$\text{Memory} = \underbrace{O\left(2kn\log_2(n/n_p)\right)}_{\text{Off-Diagonal}} + \underbrace{O\left(n\,n_p\right)}_{\text{On-Diagonal}} \tag{7.3}$$

with n_p is the size of the dense blocks, and n is the size of the entire matrix. k is the maximum rank of the off-diagonal blocks, again assumed to be constant. The complexity of matrix-vector multiplication comes out to

$$O_{MV}(nk\log_2(n)) \tag{7.4}$$

While the factorization cost is approximately

$$O_{LU}\left(k^3 n\log_2 n + k^2 n\log_2^2 n\right) \tag{7.5}$$

A format like this suffers from complexities on the off-diagonal block when the vectors which form the submatrix must change. This is most apparent when considering the sum of two HODLR matrices. This grows the number of outer product vectors to as much as $2k$, and a scheme must be determined that will reduce this back down to k. Even for large blocks, however, the SVD may be a viable way to perform this projection, given the fixed low-rank nature of the submatrices.

7.3.2. HSS Matrices

Hierarchical Semi-Separable (HSS) Matrices are HODLR matrices that satisfy a special condition known as the "Nestedness Condition." This relates U and V of the SVD for the compo-

nent blocks in such a way that these bases for the parents may be derived from the children or vice versa. To illustrate such a matrix format, consider a matrix A such that at a given 'level' $\ell \geq 1$, one has a HODLR matrix:

$$A_i^{(\ell)} = \begin{bmatrix} A_{2i}^{(\ell+1)} & U_{2i}^{(\ell+1)} S_{2i}^{(\ell+1)} V_{2i}^{*(\ell+1)} \\ U_{2i+1}^{(\ell+1)} S_{2i+1}^{(\ell+1)} V_{2i+1}^{*(\ell+1)} & A_{2i+1}^{(\ell+1)} \end{bmatrix} \tag{7.6}$$

Note that this level ℓ grows as the submatrices shrink. Qualitatively, the nestedness condition asserts that the row spaces for each column and the column space for each row are equal. What this means in practice is that the basis matrix U_i^{ℓ} may be computed from the U matrices in the submatrix $A_{2i}^{(\ell+1)}$ such that:

$$U_i^{(\ell)} = \begin{bmatrix} U_{2i}^{(\ell+1)} & 0 \\ 0 & U_{2i+1}^{(\ell+1)} \end{bmatrix} X_i^{(\ell)}, \tag{7.7}$$

meaning that the space described by $U_i^{(\ell+1)}$ may be decomposed into two orthogonal spaces described by $U_{2i}^{(\ell+1)}$ and $U_{2i+1}^{(\ell+1)}$. The same applies to the row spaces V and a new matrix Y. For a more detailed discussion on nestedness, see [6].

Storing the information required to compute this requires storing two $2k \times k$ matrices X and Y at each level, but eliminates the requirements for storing the singular vectors except at the root level. The overall storage requirements therefore improve to $O(kn)$. Matrix-vector multiplication also improves to $O(kn)$, which will be discussed in detail shortly. Interestingly, usage of these data structures bears incredibly strong relationships to the FMM [60]. This connection is quite deep, and in fact one may think of the use of these operators (HSS and the $\mathcal{H}$ matrices) as implying a form of algebraic FMM. This algebraic nature sidesteps many of the complexities of a traditional geometric FMM such as interaction lists; which are instead inferred from the structure of the matrix itself.

To demonstrate this connection and the manner in which it speeds up computation, consider the application of such a matrix A to a vector x. The vector is first successively projected into the space where the diagonal product with S may be conducted. This necessitates a product with the matrices $V^{*(p)}$, where p is the largest level ℓ and creates a set of new vec-

tors x_i^p $\forall i$:

$$x_i^p = V_i^{(p),*} x\left(\mathcal{I}_i^p\right), \tag{7.8}$$

where $\mathcal{I}_i^p$ is an index space corresponding to the entries of x being multiplied by $V_i^{*(p)}$ We follow up by computing the vectors of subsequent levels $1 \leq \ell < p$ by:

$$x_i^\ell = Y_i^{(\ell),*} \begin{bmatrix} x_{2i}^{\ell+1} \\ x_{2i+1}^{\ell+1} \end{bmatrix}, \qquad \forall i \text{ at level } \ell \text{ for } \ell = p-1, \ \ldots \ , 1. \tag{7.9}$$

The resulting vectors may be multiplied with the diagonal singular values.

$$\begin{bmatrix} y_{2i}^\ell \\ y_{2i+1}^\ell \end{bmatrix} = \begin{bmatrix} 0 & S_{2i,2i+1}^{(\ell)} \\ S_{2i+1,2i}^{(\ell)} & 0 \end{bmatrix} \begin{bmatrix} x_{2i}^\ell \\ x_{2i+1}^\ell \end{bmatrix} \qquad \forall i \text{ at level } \ell \text{ for } \ell = 1, \ \ldots \ , p-1. \tag{7.10}$$

This solution y is then propagated back up to the highest levels where it may be changed back into the appropriate space via the U operators:

$$\begin{bmatrix} y_{2i}^{\ell+1} \\ y_{2i}^{\ell+1} \end{bmatrix} = \begin{bmatrix} y_{2i}^{\ell+1} \\ y_{2i}^{\ell+1} \end{bmatrix} + X_i^{(\ell)} y_i^\ell \qquad \forall i \text{ at level } \ell \text{ for } \ell = 1, \ \ldots \ , p-1. \tag{7.11}$$

Finally, the final operations by U happen as well as the contributions of the dense blocks:

$$y\left(\mathcal{I}_i^p\right) = U_i^{(p)} y_i^p + A\left(\mathcal{I}_i^{(p)}, \mathcal{I}_i^{(p)}\right) x\left(\mathcal{I}_i^{(p)}\right). \tag{7.12}$$

This completes the product, stored in y. A notable feature of this algorithm for the product is that each submatrix which forms a part of the HSS matrix A is utilized in a matrix product only once. This means that the complexity of a matrix-vector operation is the same as the storage complexity, referenced earlier as $O(kn)$ [6]. This utilization pattern is helpful in that sophisticated algorithms have a predictable access pattern for the matrices making up the HSS representation. This means the code is theoretically able to reliably prefetch or evict these operators, which individually tend to be small, from memory; resembling a form of blocked matrix multiplication.

7.3.3. Hierarchical Matrices

The structured matrix formats of Sections 7.3.1 and 7.3.2 are special cases of a generalized concept known as a Hierarchical or $\mathcal{H}$ matrix.

These are thoroughly described by Hackbush et al. in a wide array of published works [23, 24, 25, 26]. For the purposes of this work, it is sufficient to understand that HODLR matrices generalize to the $\mathcal{H}$ class of matrices, while HSS matrices generalize to $\mathcal{H}^2$ matrices. The superscript ^{2} implies the "doubly-hierarchical" structure of both the matrix structure and the basis vectors stored in this decomposition. This concept is related to the "Nestedness Condition" of Section 7.3.2.

The data structures involved in the $\mathcal{H}^2$ matrices, and the patterns contained in the algorithms for operations such as the inverse or matrix-vector multiplication, resemble strongly a form of Algebraic FMM. This is particularly apparent in the description of the matrix-vector multiplication of Section 7.3.2. "Algebraic" here denotes a kind of algorithm where information is not encoded based on neighbor lists or geometric relations, but rather in the strengths of couplings between elements of the matrix; a complete abstraction from localized quantities. In this sense, we are "lumping" basis coefficients together, and computing only the action of these coefficients when it will yield a result above what will affect the discretized math. A more complete analysis of the relationship between the data structures and algebraic FMM can be found in [60].

While some libraries for the handling of hierarchical matrices exist, such as H2lib [9], HLIBpro [33], and STRUMPACK [56], one that can support this work must be able to support not only the matrix-vector product and inverse operations, but also a partition operation. This is necessary to handle the merge operations in the definition of the Poincaré-Steklov operators. While these capabilities are theoretically part of most libraries implementing these styles of matrices, the conventions are radically different. In particular, the process by which these are built in HPS, with smaller matrices embedded into larger ones, is theoretically a very efficient way to construct these compressible operators. However, it is not how hierarchical matrices are conventionally defined. For example, HSS matrices are conventionally initialized top-down using a user-supplied cluster tree representing the structure of the problem. This is not an insurmountable problem, but solving it will likely require custom build routines. This works best when the underlying library exposes a lot of the underlying data structures, as H2lib [9] does.

The libraries implementing these hierarchial matrices are known to provide substantial benefits in both runtime and memory savings. For a multifrontal solver, HSS matrix libraries have been demonstrated to provide up to a 7× total speedup, while using as little as 24% of the memory of the ordinary multifrontal solve [22].

7.4. The Corner Point Balance

Solving the corner point balance, even if properly decomposed, represents a significant cost which is exterior to the "fast" algorithm presented in Section 3.2. For performance reasons, it is highly desired to establish an approximate but numerically stable way to derive a working approximation of these values. The method of resolving the corner points currently exists as a precise solver designed for maximum consistency and stability; but this is not strictly necessary. The iterative nature of Algorithms 3 and 4 means that only an approximate method is required; but no matter what approximation is chosen, it must lead to a stable method upon iteration between HPS and this corner point solver.

Finding such an approximate method is an open problem in resolving this work. The 'exact' method as derived in Section 4.7 works for diffusion and SP_3, but struggles for certain problems in GSP_3. However, this method itself has a few concerns hanging over it as well; particularly in the conservation of the second angular moments which have a tenuous physical meaning. And so, this itself is also subject to future research.

One such example of a pattern in which one may work with the corner point balance approximately is related to past implementations of corner point reconstruction. In some previous approaches, such as that used in TRIPEN [32], a system would be constructed for these corner points and solved using the ILU(0)-BiCGStab algorithm. Alternatively, some iterative spatial sweep over the corners is also possible, and may provide improved performance if the sweep can be implemented in parallel and converges quickly.

A working approximate method enables the fast solver to perform as advertised, while greatly helping the future possibility of an extension into 3D space. The problem of corner points is significantly exacerbated in 3D as the number of them grows to include points on the edges of cuboids as well as cuboid vertices. Solving such a system would be challenging without a reliable approximate method.

7.5. Complexity Analysis

7.5.1. Operation Count

Naturally the performance of any method should not be taken on faith alone, so here a rudimentary complexity analysis will be reproduced here based on the work of Martinsson [46]. The analysis within this section is a reproduction of the aforementioned paper where minor changes are made to convert it to a 2D analysis rather than the 3D case as printed.

Assume that we have a 2D mesh which is recursively partitioned into halves until the leaf nodes are reached. Consider a mesh with N total unknowns, on a cube of $n \times n$ points. To form the hierarchy, the overall mesh is recursively bisected in the x and y directions in turn until the leaf nodes (which have a total of m unknowns each) are defined. The number of levels in this hierarchy will be represented by L which must be divisible by 2. Each level l then contains 2^l regions, if the root (or largest) node is considered to be $l = 0$, and the problem has 2^L leaf nodes. It becomes important to know the number of unknowns which exist on the interface in a given level; these couple the nodes together. This value, denoted m_l, is estimated as $m_l \sim 2^{-\lfloor l/2 \rfloor} n$.

Both the build and solve stages will be analyzed in this section. First, the build time, which has 2 separate components; the leaves and the merge operations. The leaves are relatively simple to estimate. In a worst-case scenario, there is one material per subdomain, which makes the dominant cost the factorization of A_{ii}^{τ} for each cell. This cost will be $O(2^L m^2) \sim O(Nm)$, which is linear in the number of unknowns. Certain accelerations are possible in the case of regular mesh cells or duplicate regions. However, these will be left aside for purposes of generality.

The next part to consider are the merge operations. The dominant cost here is by a wide margin, the inverse operation on the T operators. This is because for non-leaf nodes, the number of boundary and internal unknowns are of the same order, and this factorization will certainly be a similar or higher complexity to the multiplication, depending on the underlying data structure used. The size of this matrix is, of course, the size of the interface between the 2 nodes being merged. Suppose that this factorization will take some number of operations $O(m_l^{\alpha})$ where α is some positive number. For a naïve formulation, it will be equal to 3; but algorithms involving rank-structured matrices reduce this to some real number in the range $[1, 3]$. The precise value will be determined by the compressibility of the

underlying matrix, and so varies from case to case; in general for elliptic problems it follows a rough logarithmic trend with size [46]. Matrices arising from discretizations that maintain compressibility as the problem size increases will therefore see a better α. A brief study on the compressibility of the operators arising from Low-order transport in Chapter 4 will be conducted in Section 7.5.2.

The number of merges required at each level is half the number of nodes per level. Add to this the leaf node time, and the order of operation count may be represented as:

$$\sum_{l=0}^{L} 2^l (m_l)^\alpha + Nm^2 \sim \sum_{l=0}^{L} 2^l (2^{-l/2} n)^\alpha + Nm \tag{7.13}$$

$$\sim \sum_{l=0}^{L} 2^{l(1-\alpha/2)} n^\alpha + Nm \tag{7.14}$$

$$\sim N^{\alpha/2} \sum_{l=0}^{L} 2^{l(1-\alpha/2)} + Nm, \tag{7.15}$$

a nearly identical result to that obtained in Martinsson's work [46], but here is adapted to two dimensions. As he noted, this permits two regimes; in the case of an α less than 2, we wind up with a total cost $O(Nm)$ dominated by the term corresponding to the leaf calculations. Alternatively, if α is greater than 2 (but still less than 3, the worst case) we instead may expect a complexity of $O(N^{\alpha/2})$ which could in principle span $O(N)$ to $O(N^{3/2})$.

Martinsson's work analyzed the case of a three-dimensional problem, where the only significant change is that the break-point α becomes $\frac{3}{2}$. In that case, the scaling in the leaf-dominated regime changes to $O(Nm^2)$. Meanwhile, the upper end of the estimate changes to $O(N^{2\alpha/3})$, thus spanning $O(N)$ to $O(N^2)$.

However, this analysis is somewhat simplistic and says nothing of the likelihood of where α may fall. This is not a failing of the analysis itself but rather a statement of how much depends on the compressibility of the differential operators, and the exact complexities of the structured linear algebra algorithms. Despite this, more detailed analyses have been performed which better take into account the impact of the low-rank inversion techniques mentioned throughout this section. These reveal the estimates above to be rather pessimistic, and the number of interaction ranks for elliptic problems tends to scale much more gently than the worst-case analyses suggest; something examined in the next section. In a numerical

experiment and in-depth analysis performed by Hao and Martinsson [27], a still pessimistic, but tighter upper bound on the build stage was $O(N^{4/3})$ for general 3D problems.

7.5.2. Compressibility Study

The compressibility of the Poincaré-Steklov operator T is a critical part that guarantees the stated performance of the method. This operator is generated as part of the build stage in Section 3.2.2 and is one of the more computationally-intensive parts of the code, as discussed in Section 7.5.1. This compressibility is governed by the numerical rank of the off-diagonal blocks. Recall the definition of T in terms of the merger of two children:

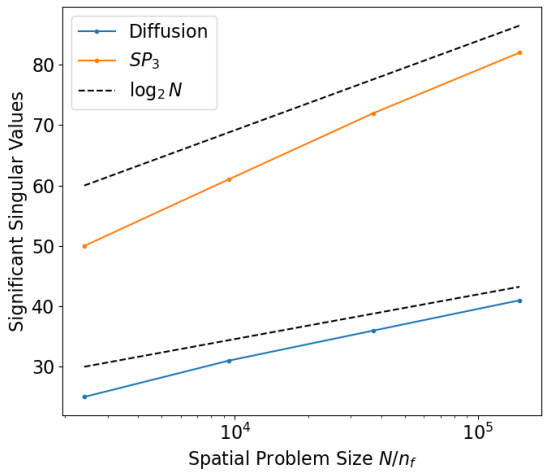
$$T^\tau \equiv \begin{bmatrix} T^\alpha_{11} & 0 \\ 0 & T^\beta_{22} \end{bmatrix} + \begin{bmatrix} T^\alpha_{13} \\ T^\beta_{23} \end{bmatrix} S^\tau, \tag{7.16}$$

where the subscripts indicate the index spaces of a merge operation, as in Figure 3.6. The off-diagonal blocks for this T^τ may be computed from the second term;

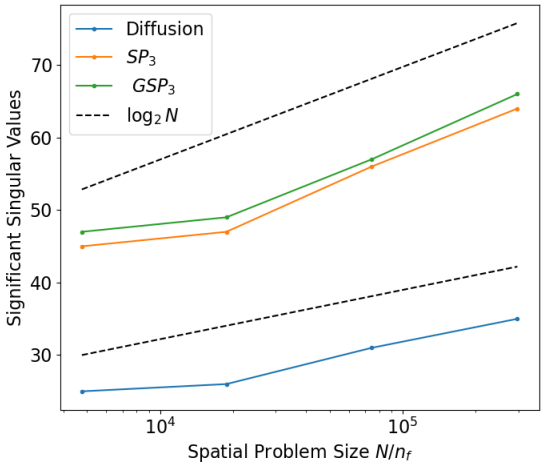
$$\begin{bmatrix} T^\alpha_{13} \\ T^\beta_{23} \end{bmatrix} \begin{bmatrix} S^\tau_{31} & S^\tau_{32} \end{bmatrix} = \begin{bmatrix} T^\alpha_{13}S^\tau_{31} & \boxed{T^\alpha_{13}S^\tau_{32}} \\ \boxed{T^\alpha_{23}S^\tau_{31}} & T^\alpha_{23}S^\tau_{32} \end{bmatrix}. \tag{7.17}$$

The numerical rank of these at a given tolerance ϵ (for the purpose of this research, we use $\epsilon = 1.0 \times 10^{-12}$) may be obtained by computing the SVD of these blocks and counting the number of singular values above this tolerance.

This analysis is performed on the real problems of Sections 6.3 and 6.4. Data is obtained by finding the maximum of the numerical rank between the two off-diagonal blocks of the root T operator. While not perfect, this is a satisfactory estimate of overall compressibility. The number of significant singular values $\sigma > \epsilon$ is plotted in Figure 7.2 against the spatial problem size N/n_f. Here, N is the total number of unknowns for a given system while n_f is the number of fields in the PDE; for GSP_3 and SP_3, this is 2; while for diffusion this is 1. As the problem size grows, this is expected to increase at a rate of $\sim \log_2 N$. Figure 7.2a shows the compressibility of the 3×3 mini-lattices of Section 6.3. These values for the three different cases are not meaningfully different; so only one (the MOX lattice) is plotted here. The two dashed lines correspond to the two expected trends of $\log_2 N$. For this problem, the compressibility is quite close to the intended scaling; with only a short ramp-up period. The



(a) Compressibility for 3×3 Mini-Lattice



(b) Compressibility for Pin-Homogenized Assembly

Figure 7.2: Compressibility in Low-Order Transport

magnitudes are also rather notable; only $\sim$ 80 significant singular values at the top level, with an overall problem size of over 100,000 is quite remarkable.

The compressibility for the pin-homogenized assembly of Section 6.4 is plotted in Figure 7.2b. While the initial trend of the ramp-up is inverted (a flatter increase rather than a sharper one), the story is much the same. GSP_3 has a slightly increased number of singular values for the same problem as compared to SP_3, but this increase is quite minimal. All governing equations still tend to the same overall trend of $\log_2 N$.

This trend in compressibility is, on its own, insufficient to make a determination about the scaling bottlenecks in Section 7.5.1. However, matrix ranks following a trend of $\log_2 N$ is a tighter bound than used in the study conducted by Hao and Martinsson [27]; which was noted to be pessimistic. That pessimistic bound corresponded to an $\alpha = 2$. Thus, we may reasonably expect the introduction of a structured linear algebra package into the existing implementation to result in a method which scales as $O(N \log N)$ or better in 2 dimensions, to use the same formulas as in Section 7.5.1.

Chapter 8

Conclusions and Future Work

8.1. Summary and Conclusions

This dissertation follows the development of a novel solver and discretization from foundational theory and analysis such as the WRM, all the way through to final evaluation with C5G7-derived small lattice problems. Our motivation for carrying out this implementation and study has been to create a fast neutronics solver with a very wide domain of applicability based upon the proven simplified spherical harmonic discretizations. Such a solver could enable design-space evaluations of reactor designs which may be very different from the technologies currently seen in use today.

In Chapter 1, an overview of the developments in spherical harmonic discretizations was presented; including both the original history of SP_N theory and recent extensions such as GSP_N. These extensions are thought to allow a more rigorous boundary treatment by removing the locally-1D assumption at material interfaces. This theory formed the backdrop for the development and analysis of a novel 1D discretization with collapsed basis functions, described as the LGL discretization. This discretization lends itself particularly well to certain recently-published algorithms from the applied math community. Chapter 3 presents one such algorithm, known as HPS, in a form amenable to the discretizations studied. HPS is desired specifically because of its notion of conserved interface quantities and the explicit form of interface conditions.

Analysis of HPS as applied to Diffusion, SP_3, and GSP_3 was divided into a MMS study in Chapter 5 and more complex benchmark-derived problems in Chapter 6. MMS analysis verified the solver implementation and demonstrated that the maximum convergence order (5th)

imposed by the Bramble-Hilbert lemma may be achieved by this discretization. However, the quadrature-based self-lumping built into the discretization makes a simpler numerical integration more tractable, with a penalty of only one order of convergence. The more comprehensive analysis of Chapter 6 consisted of a test on a deformed pin-resolved mesh modelling 3 mini-lattice problems for diffusion and SP_3; followed by a large pin-homogenized assembly problem analyzed with diffusion, SP_3, and GSP_3. The results, consisting of criticality and pin-power comparisons, revealed errors typical of the simplified spherical harmonic equations. That is, errors became largest for problems with large amounts of directionality in the angular flux (for example, near strong absorbers). GSP_3 performed well at low refinement levels for the lattice problem, reproducing similar results as Chao et al. [14]. Refinements, however, caused the GSP_3 solution to approach SP_3 values; which is thought to be due to the treatment of the corner balance.

Chapter 7 first covers various topics of performance related to this solver in Sections 7.1 and 7.2, before moving onto the topic of structured linear algebra in Section 7.3. Such a computational package is a high priority for future work, as discussed in that section. It enables the kind of special treatment for compressible operators and efficient storage implementations that lend the HPS method its optimized scaling. The corner point solver is also discussed, with some direction on the future efforts in that space. This too is a high priority for future work; as stable methods for this balance are necessary if the stated scaling behavior is to be reached. With both of these concerns addressed, a proper scaling study in time and memory cost could be performed on the solver implementation; which would much more conclusively prove the results of the brief complexity analysis and compressibility study included at the end of Chapter 7.

The design space for nuclear reactors has been growing for some time. This growth is not limited to exclusively Gen-IV reactors such as molten salt or gas-cooled cores, but includes also subtler changes which make approximations once commonly-used become troublesome. Smaller cores and greater heterogeneity in neutron flux, design features that occur in small modular reactors, both have the potential to strain common assumptions used in the analysis of large light water reactors. The work in this thesis demonstrates that solver techniques exist that leverage proven modelling techniques while avoiding some of these assumptions; predominantly the transverse leakage approximation. The benefits inherent to these hierarchical fast solvers; their scaling in operations and memory, the parallelism inherent in these domain decompositions; are enormous in today's supercomputing climate

and it is our hope that these structured methods will continue to be analyzed for use in a wide variety of scientific computing applications.

8.2. Matrix Decomposition Updates

Of critical importance for reactor design calculations is the ability to incorporate feedback from other governing physics of the problem in question. For example, thermal feedback has a profound impact on the neutronics of a problem given how it alters the cross-sections and therefore the coefficients of the underlying governing equation; be it diffusion, SP_3 or GSP_3.

While this in principle would force a re-computation of the global operator, requiring a new build step as discussed in Section 3.2.2, there is research being performed into coefficient updates for these matrix decompositions [43]. An optimized coefficient update is expected to complete in roughly the same amount of time as a solve step, which would be a dramatic improvement over a full refactor. While the estimates in this reference are likely to be optimistic (primarily due to the fact that coefficient updates for transport affect the boundary conditions between cells), they still provide a pathway to performing updates without resorting to a full rebuild of the problem. Support for these coefficient updates is highly desired as a topic of future work to make this solver approach more amenable to multi-physics coupling.

8.3. Truly Unstructured Grids

Results presented in Chapter 6 utilized a deformed but ultimately structured grid. However, this limitation is not imposed by transverse leakage. Rather, it is imposed by the manner in which this work has structured the corner point solver and the iterations between it and the HPS method; particularly, that described in Algorithm 3 and section 4.7 Relaxing this assumption is simple in principle; but does require some extra data pertaining to traversals of the unstructured grid after the partitioning process described in Section 7.2. To borrow an expression, the complexity comes not from anything strictly mathematical but from the bookkeeping involved in storing the mesh and appropriate neighbor information.

Additionally, setting up the corner point equations in unconventional mesh cell configu-

rations (such as the intersection of 3 quadrilaterals on the domain boundary) can be more difficult than the prescribed intersection of 2 or 4 which occur in the structured grid. The prescribed intersections are very predictable, and it is trivial to determine whether a given corner needs boundary conditions or interface conditions based on whether it is the intersection of 4 or fewer elements, respectively. In less predictable configurations, more bookkeeping is required; but it is not a limit of the physics or problem representation.

8.3.1. Non-Quadrilateral Elements

Automated meshing tools frequently choose to use triangular meshes as opposed to quadrilateral ones. Part of this is a conventional detail of how FEMs are normally applied, but triangular meshes are substantially easier to generate and generally suffer from fewer problems than their quadrilateral counterparts. This is primarily because there are far fewer "degenerate" triangular mesh cells (collinear or coincident points) while quadrilateral meshes may become concave, self-intersecting, or themselves collinear.

Regardless of the reason, since there are so many tools dedicated to these triangular meshes it would be a good idea to have a solver that can support these. Support for these types of meshes is not trivial but it amounts to a relatively short list of code changes. The kernels of Chapter 4 must be altered to be defined upon a unit right triangle. In order to do this, one needs:

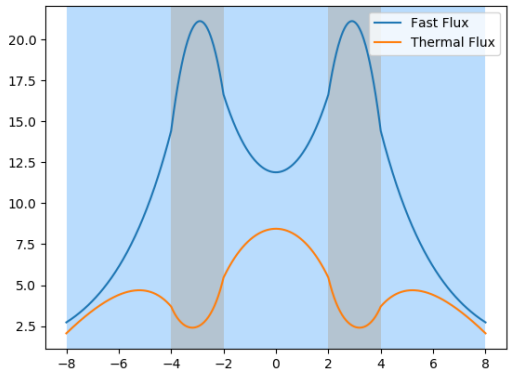
1. A Gauss-Lobatto quadrature rule on the unit triangle
2. High-order Lagrange basis functions defined on these nodes
3. A re-derivation of the discrete differential operators (Section 4.3) and associated tensors (Appendix B)

The most difficult of these is item 2. Quadrature rules have been studied on the unit triangle and suitable versions found [59]. However, defining a basis on this quadrature is because the information encoded in the differential operators quite complex and is fundamentally different from the tensor product counterpart on the quadrilaterals [28]. These new bases (and a set of shape functions corresponding to the three vertices of the triangle), if found, enable a re-derivation of the differential operators of the problem.

Appendix A

Analytic Diffusion Solutions

Recall the problem geometry and material data specified in Chapter 2: Noting the symmetry



Fuel Region:

$$\Sigma^F_{t,1} = 0.5 \quad \Sigma^F_{s0,1} = 0.41 \quad \Sigma^F_{s1,1} = 0.25\Sigma^f_{s0}$$
$$\Sigma^F_{t,2} = 1.0 \quad \Sigma^F_{s0,2} = 0.2 \quad \Sigma^F_{s1,2} = 0.25\Sigma^t_{s0}$$
$$\Sigma^F_{s0,2\leftarrow 1} = 0.04 \quad Q^F_0 = 10 \quad Q^F_1 = 0$$

Moderator Region:

$$\Sigma^M_{t,1} = 0.25 \quad \Sigma^M_{s0,1} = 0.11 \quad \Sigma^M_{s1,1} = 0.25\Sigma^f_{s0}$$
$$\Sigma^M_{t,2} = 0.5 \quad \Sigma^M_{s0,2} = 0.5 \quad \Sigma^M_{s1,2} = 0.25\Sigma^t_{s0}$$
$$\Sigma^M_{s0,2\leftarrow 1} = 0.11 \quad Q^M_1 = 0 \quad Q^M_2 = 0$$

of the problem, analytic solutions will be derived for $x \in [0, 8]$, separated into 3 regions denoted with a superscript; the inner moderator 1, fuel region 2, or outer moderator 3. The region $0 \leq x < a$ is the inner moderator, and is denoted $M1$; for this case, $a = 2$ cm. The region $a \leq x < b$ is fuel, denoted F, with $b = 4$ cm for this case. The region $b \leq x < c$ is the outer moderator, denoted $M2$. $c = 8$ cm for this case. Coordinates may be assumed to be positive.

First, define some diffusion coefficients and removal cross-sections:

$$\Sigma^F_{r0,g} = \Sigma^F_{t,g} - \Sigma^F_{s0,g}, \tag{A.1a}$$
$$\Sigma^M_{r0,g} = \Sigma^M_{t,g} - \Sigma^M_{s0,g}, \tag{A.1b}$$
$$\Sigma^F_{r1,g} = \Sigma^F_{t,g} - \Sigma^F_{s1,g}, \tag{A.1c}$$
$$\Sigma^M_{r1,g} = \Sigma^M_{t,g} - \Sigma^M_{s1,g}. \tag{A.1d}$$

Note that $M1$ and $M2$ are assumed to be the same material M.

$$D_1^F = (3\Sigma_{r1,1}^F)^{-1} \qquad D_2^F = (3\Sigma_{r1,2}^F)^{-1} \qquad D_1^M = (3\Sigma_{r1,1}^M)^{-1} \qquad D_2^M = (3\Sigma_{r1,2}^M)^{-1}$$
$$L_1^F = \sqrt{\frac{D_1^F}{\Sigma_{r0,1}}} \qquad L_2^F = \sqrt{\frac{D_2^F}{\Sigma_{r0,2}}} \qquad L_1^M = \sqrt{\frac{D_1^M}{\Sigma_{r0,1}}} \qquad L_2^M = \sqrt{\frac{D_2^M}{\Sigma_{r0,2}}} \tag{A.2}$$

As the problems lack up-scattering, the fast flux is solved first via method of undetermined coefficients. The assumed forms of the flux are as follows, with the superscript indicating the region index.

$$\phi_1^{M1}(x) = C_0^1 \cosh(x/L_1^M) \tag{A.3a}$$
$$\phi_1^F(x) = C_1^1 \sinh(x/L_1^F) + C_2^1 \cosh(x/L_1^F) + A_0^1 \tag{A.3b}$$
$$\phi_1^{M2}(x) = C_3^1 \sinh(x/L_1^M) + C_4^1 \cosh(x/L_1^M) \tag{A.3c}$$

We have 5 interface and boundary conditions to satisfy (symmetry at $x = 0$ is automatically satisfied by the form for ϕ_1^F). These are given as continuity of flux, continuity of net current, and zero re-entrant current at the outer boundary.

$$\phi_1^{M1}(a) = \phi_1^F(a), \tag{A.4a}$$
$$\phi_1^F(b) = \phi_1^{M2}(b), \tag{A.4b}$$
$$D_1^M \frac{d\phi_1^{M1}}{dx}\Big|_{x=a} = D_1^F \frac{d\phi_1^F}{dx}\Big|_{x=a}, \tag{A.4c}$$
$$D_1^M \frac{d\phi_1^F}{dx}\Big|_{x=b} = D_1^F \frac{d\phi_1^{M2}}{dx}\Big|_{x=b}, \tag{A.4d}$$
$$0 = \frac{1}{4}\phi_1^{M2}(c) + \frac{D^M}{2}\frac{d\phi_1^{M2}}{dx}\Big|_{x=c}. \tag{A.4e}$$

We may also define A_0^1 from the particular solution:

$$A_0^1 = \frac{Q_1^F}{\Sigma_{r0,1}} \tag{A.5}$$

Evaluating the first 4 conditions:

$$C_0^1 \cosh\left(\frac{a}{L_1^M}\right) - C_1^1 \sinh\left(\frac{a}{L_1^F}\right) - C_2^1 \cosh\left(\frac{a}{L_1^F}\right) = A_0^1, \tag{A.6a}$$

$$C_0^1 \frac{D^M}{L_1^M} \sinh\left(\frac{a}{L_1^M}\right) - C_1^1 \frac{D^F}{L_1^F} \cosh\left(\frac{a}{L_1^F}\right) - C_2^1 \frac{D^F}{L_1^F} \sinh\left(\frac{a}{L_1^F}\right) = 0, \tag{A.6b}$$

$$C_1^1 \sinh\left(\frac{b}{L_1^F}\right) + C_2^1 \cosh\left(\frac{b}{L_1^F}\right) - C_3^1 \sinh\left(\frac{b}{L_1^M}\right) - C_4^1 \cosh\left(\frac{b}{L_1^M}\right) = -A_0^1, \tag{A.6c}$$

$$C_1^1 \frac{D_1^F}{L_1^F} \cosh\left(\frac{b}{L_1^F}\right) + C_2^1 \frac{D_1^F}{L_1^F} \sinh\left(\frac{b}{L_1^F}\right) - C_3^1 \frac{D_1^M}{L_1^M} \cosh\left(\frac{b}{L_1^M}\right) - C_4^1 \frac{D_1^M}{L_1^M} \sinh\left(\frac{b}{L_1^M}\right) = 0. \tag{A.6d}$$

The final condition requires some extra work to put into the proper form.

$$\begin{aligned} \frac{1}{4}\left[C_3^1 \sinh\left(\frac{b}{L_1^M}\right) + C_4^1 \cosh\left(\frac{b}{L_1^M}\right)\right] + \frac{D_1^M}{2L_1^M}\left[C_3^1 \cosh\left(\frac{b}{L_1^M}\right) + C_4^1 \sinh\left(\frac{b}{L_1^M}\right)\right] = 0 \\ C_3^1\left[\sinh\left(\frac{b}{L_1^M}\right) + \frac{2D_1^M}{L_1^M} \cosh\left(\frac{b}{L_1^M}\right)\right] + C_4^1\left[\cosh\left(\frac{b}{L_1^M}\right) + \frac{2D_1^M}{L_1^M} \sinh\left(\frac{b}{L_1^M}\right)\right] = 0 \end{aligned} \tag{A.6e}$$

Combined with the 4 equations above, this forms a linear system of 5 equations and 5 unknowns. This may be solved analytically, but the process does not provide any profound insights; in practice, simply constructing a matrix with the appropriate elements and solving the resulting system will yield a perfectly good solution.

As for the thermal solutions, the path is much the same, but a new form of the particular solution is now used. The equations, again computed via the method of undetermined coefficients, are as follows:

$$\phi_2^{M1}(x) = C_0^2 \cosh(x/L_2^M) + A_0^2 \cosh(x/L_1^M) \tag{A.7a}$$

$$\phi_2^F(x) = C_1^2 \sinh(x/L_2^F) + C_2^2 \cosh(x/L_2^F) + A_1^2 \sinh(x/L_1^F) + A_2^2 \cosh(x/L_1^F) \tag{A.7b}$$

$$\phi_2^{M2}(x) = C_3^2 \sinh(x/L_2^M) + C_4^2 \cosh(x/L_2^M) + A_3^2 \sinh(x/L_1^M) + A_4^2 \cosh(x/L_1^M) \tag{A.7c}$$

The relations closing the system are very similar to the previous (fast) case:

$$\phi_2^{M1}(a) = \phi_2^F(a), \tag{A.8a}$$

$$\phi_2^F(b) = \phi_2^{M2}(b), \tag{A.8b}$$

$$D_2^M \frac{d\phi_2^{M1}}{dx}\Big|_{x=a} = D_2^F \frac{d\phi_2^F}{dx}\Big|_{x=a}, \tag{A.8c}$$

$$D_2^M \frac{d\phi_2^F}{dx}\Big|_{x=b} = D_2^F \frac{d\phi_2^{M2}}{dx}\Big|_{x=b}, \tag{A.8d}$$

$$0 = \frac{1}{4}\phi_2^{M2}(c) + \frac{D^M}{2}\frac{d\phi_2^{M2}}{dx}\Big|_{x=c}. \tag{A.8e}$$

Before continuing, expressions for the coefficients A must be derived. This may be obtained by plugging in the particular solutions to the differential equation:

$$A_0^2\left[\frac{D_2^M}{(L_1^M)^2} + \Sigma_{r0,2}^M\right] = \Sigma_{s0,2\leftarrow 1}^M C_0^1,$$

$$A_0^2 = \frac{\Sigma_{s0,2\leftarrow 1}^M}{D_2^M}\left[(L_1^M)^{-2} + (L_2^M)^{-2}\right]^{-1} C_0^1. \tag{A.9}$$

The same approach is used to define the other 4 A coefficients:

$$A_1^2 = \frac{\Sigma_{s0,2\leftarrow 1}^F}{D_2^F}\left[(L_1^F)^{-2} + (L_2^F)^{-2}\right]^{-1} C_1^1, \tag{A.10a}$$

$$A_2^2 = \frac{\Sigma_{s0,2\leftarrow 1}^F}{D_2^F}\left[(L_1^M)^{-2} + (L_2^M)^{-2}\right]^{-1} C_2^1, \tag{A.10c}$$

$$A_3^2 = \frac{\Sigma_{s0,2\leftarrow 1}^M}{D_2^M}\left[(L_1^M)^{-2} + (L_2^M)^{-2}\right]^{-1} C_3^1, \tag{A.10b}$$

$$A_4^2 = \frac{\Sigma_{s0,2\leftarrow 1}^M}{D_2^M}\left[(L_1^M)^{-2} + (L_2^M)^{-2}\right]^{-1} C_4^1. \tag{A.10d}$$

Using these relations, and the interface/boundary conditions, a matrix equation may be generated for the thermal group. This may then be solved and combined with the particular solution, in a similar manner to the fast flux.

Appendix B

Mesh Derivative Tensors for the Quadrilateral Case

This appendix describes the practice by which the rank-3 and rank-4 Ξ tensors of Section 4.3 may be computed.

Inverting the bilinear interpolation is likely impossible analytically, and certainly infeasible. Instead, we make use of implicit differentiation. Take the expression for $x(u,v)$ and $y(u,v)$:

$$x(u,v) = \sum_{j=1}^{4} x_j^c \sigma_j(u,v) \qquad\qquad y(u,v) = \sum_{j=1}^{4} y_j^c \sigma_j(u,v)$$

Where x_j^c, y_j^c are the global coordinates of the corner indexed by j and $\sigma(u,v)$ are the bilinear shape functions. For a cell over $[-1,1]^2$, these take the form:

$$\sigma_j(u,v) = \frac{1}{4}(u \pm 1)(v \pm 1), \tag{B.1}$$

where the signs depend on which corner j the index refers to. By operating on these with second-order derivatives, the desired terms of Ξ may be solved for.

$$\frac{\partial^2}{\partial x^2}x(u,v) = \sum_{j=1}^{4} x_j^c \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\frac{\partial \sigma_j}{\partial u} + \frac{\partial v}{\partial x}\frac{\partial \sigma_j}{\partial v}\right)$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\frac{\partial^2 u}{\partial x^2}\frac{\partial \sigma_j}{\partial u} + \frac{\partial u}{\partial x}\frac{\partial}{\partial x}\frac{\partial \sigma_j}{\partial u} + \frac{\partial^2 v}{\partial x^2}\frac{\partial \sigma_j}{\partial v} + \frac{\partial v}{\partial x}\frac{\partial}{\partial x}\frac{\partial \sigma_j}{\partial v}\right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{000}\frac{\partial \sigma_j}{\partial u} + \frac{\partial u}{\partial x}\left(\cancel{\frac{\partial u}{\partial x}\frac{\partial^2 \sigma_j}{\partial u^2}} + \frac{\partial v}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v}\right) + \Xi_{100}\frac{\partial \sigma_j}{\partial v} + \frac{\partial v}{\partial x}\left(\frac{\partial u}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v} + \cancel{\frac{\partial v}{\partial x}\frac{\partial^2 \sigma_j}{\partial v^2}}\right)\right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{000}\frac{\partial \sigma_j}{\partial u} + \Xi_{100}\frac{\partial \sigma_j}{\partial v} + 2\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v}\right]$$

$$\sum_{j=1}^{4} x_j^c \frac{\partial \sigma_j}{\partial u}\Xi_{000} + x_j^c \frac{\partial \sigma_j}{\partial v}\Xi_{100} = -2\sum_{j=1}^{4} x_j^c \frac{\partial u}{\partial x}\frac{\partial v}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v}$$

A similar process on the y coordinates yields:

$$0 = \sum_{j=1}^{4} y_j^c \left[\Xi_{000}\frac{\partial \sigma_j}{\partial u} + \Xi_{100}\frac{\partial \sigma_j}{\partial v} + 2\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v}\right]$$

These two equations combined yield a system of 2 equations with unknowns Ξ_{000} and Ξ_{100}. Thus, these two equations determine these two values in the tensor.

Similarly, applying $\frac{\partial^2}{\partial y^2}$ yields the 2 equations:

$$\frac{\partial^2}{\partial y^2}x(u,v) = \sum_{j=1}^{4} x_j^c \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\frac{\partial \sigma_j}{\partial u} + \frac{\partial v}{\partial y}\frac{\partial \sigma_j}{\partial v}\right)$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\frac{\partial^2 u}{\partial y^2}\frac{\partial \sigma_j}{\partial u} + \frac{\partial u}{\partial y}\frac{\partial}{\partial y}\frac{\partial \sigma_j}{\partial u} + \frac{\partial^2 v}{\partial y^2}\frac{\partial \sigma_j}{\partial v} + \frac{\partial v}{\partial y}\frac{\partial}{\partial y}\frac{\partial \sigma_j}{\partial v}\right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{011}\frac{\partial \sigma_j}{\partial u} + \frac{\partial u}{\partial y}\left(\cancel{\frac{\partial u}{\partial y}\frac{\partial^2 \sigma_j}{\partial u^2}} + \frac{\partial v}{\partial y}\frac{\partial^2 \sigma_j}{\partial u \partial v}\right) + \Xi_{111}\frac{\partial \sigma_j}{\partial v} + \frac{\partial v}{\partial y}\left(\frac{\partial u}{\partial y}\frac{\partial^2 \sigma_j}{\partial u \partial v} + \cancel{\frac{\partial v}{\partial y}\frac{\partial^2 \sigma_j}{\partial v^2}}\right)\right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{011}\frac{\partial \sigma_j}{\partial u} + \Xi_{111}\frac{\partial \sigma_j}{\partial v} + 2\frac{\partial u}{\partial y}\frac{\partial v}{\partial y}\frac{\partial^2 \sigma_j}{\partial u \partial v}\right]$$

and

$$0 = \sum_{j=1}^{4} y_j^c \left[\Xi_{011} \frac{\partial \sigma_j}{\partial u} + \Xi_{111} \frac{\partial \sigma_j}{\partial v} + 2 \frac{\partial u}{\partial y} \frac{\partial v}{\partial y} \frac{\partial^2 \sigma_j}{\partial u \partial v} \right]$$

We note that this is the same result as flipping all but the first index of Ξ and swapping $\partial x \Rightarrow \partial y$.

The mixed derivative is a little different; starting again from the x coordinate:

$$\begin{aligned}
\frac{\partial^2}{\partial x \partial y} x(u,v) &= \sum_{j=1}^{4} x_j^c \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial y} \frac{\partial \sigma_j}{\partial u} + \frac{\partial v}{\partial y} \frac{\partial \sigma_j}{\partial v} \right) \\
0 &= \sum_{j=1}^{4} x_j^c \left[\frac{\partial^2 u}{\partial x \partial y} \frac{\partial \sigma_j}{\partial u} + \frac{\partial u}{\partial y} \left(\frac{\partial}{\partial x} \frac{\partial \sigma_j}{\partial u} \right) + \frac{\partial^2 v}{\partial x \partial y} \frac{\partial \sigma_j}{\partial v} + \frac{\partial v}{\partial y} \left(\frac{\partial}{\partial x} \frac{\partial \sigma_j}{\partial v} \right) \right] \\
0 &= \sum_{j=1}^{4} x_j^c \Bigg[\Xi_{001} \frac{\partial \sigma_j}{\partial u} + \frac{\partial u}{\partial y} \left(\cancel{\frac{\partial u}{\partial x} \frac{\partial^2 \sigma_j}{\partial u^2}} + \frac{\partial v}{\partial x} \frac{\partial^2 \sigma_j}{\partial v \partial u} \right) \\
&\qquad + \Xi_{101} \frac{\partial \sigma_j}{\partial v} + \frac{\partial v}{\partial y} \left(\frac{\partial u}{\partial x} \frac{\partial^2 \sigma_j}{\partial u \partial v} + \cancel{\frac{\partial v}{\partial x} \frac{\partial^2 \sigma_j}{\partial v^2}} \right) \Bigg] \\
0 &= \sum_{j=1}^{4} x_j^c \left[\Xi_{001} \frac{\partial \sigma_j}{\partial u} + \Xi_{101} \frac{\partial \sigma_j}{\partial v} + \frac{\partial^2 \sigma_j}{\partial u \partial v} \left(\frac{\partial u}{\partial y} \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial u}{\partial x} \right) \right]
\end{aligned}$$

Analogously:

$$0 = \sum_{j=1}^{4} y_j^c \left[\Xi_{001} \frac{\partial \sigma_j}{\partial u} + \Xi_{101} \frac{\partial \sigma_j}{\partial v} + \frac{\partial^2 \sigma_j}{\partial u \partial v} \left(\frac{\partial u}{\partial y} \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial u}{\partial x} \right) \right]$$

Since Ξ_{ijk} is symmetric under permutations of j, k this is sufficient to determine the entire tensor. We must now repeat the process for $\Xi_{\mu ijk}$. While this tensor is symmetric under permutations of i, j, k we must still do a significant amount of algebra.

A reminder from the previous page:

$$\frac{\partial}{\partial x}\frac{\partial \sigma_j}{\partial u} = \frac{\partial v}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v} \qquad \frac{\partial}{\partial x}\frac{\partial \sigma_j}{\partial v} = \frac{\partial u}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v}$$
$$\frac{\partial}{\partial y}\frac{\partial \sigma_j}{\partial u} = \frac{\partial v}{\partial y}\frac{\partial^2 \sigma_j}{\partial u \partial v} \qquad \frac{\partial}{\partial y}\frac{\partial \sigma_j}{\partial v} = \frac{\partial u}{\partial y}\frac{\partial^2 \sigma_j}{\partial u \partial v}$$

$$\frac{\partial^3}{\partial x^3}x(u,v) = \sum_{j=1}^{4} x_j^c \frac{\partial^2}{\partial x^2}\left(\frac{\partial u}{\partial x}\frac{\partial \sigma_j}{\partial u} + \frac{\partial v}{\partial x}\frac{\partial \sigma_j}{\partial v}\right)$$

$$0 = \sum_{j=1}^{4} x_j^c \frac{\partial}{\partial x}\left[\frac{\partial^2 u}{\partial x^2}\frac{\partial \sigma_j}{\partial u} + \frac{\partial^2 v}{\partial x^2}\frac{\partial \sigma_j}{\partial v} + 2\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v}\right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\frac{\partial^3 u}{\partial x^3}\frac{\partial \sigma_j}{\partial u} + \frac{\partial^2 u}{\partial x^2}\left(\frac{\partial}{\partial x}\frac{\partial \sigma_j}{\partial u}\right) + \frac{\partial^3 v}{\partial x^3}\frac{\partial \sigma_j}{\partial v} + \frac{\partial^2 v}{\partial x^2}\left(\frac{\partial}{\partial x}\frac{\partial \sigma_j}{\partial v}\right)\right.$$
$$\left. + 2\left(\frac{\partial^2 u}{\partial x^2}\frac{\partial v}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v} + \frac{\partial u}{\partial x}\frac{\partial^2 v}{\partial x^2}\frac{\partial^2 \sigma_j}{\partial u \partial v} + \frac{\partial u}{\partial x}\frac{\partial v}{\partial x}\cancel{\left[\frac{\partial}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v}\right]}\right)\right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{0000}\frac{\partial \sigma_j}{\partial u} + \Xi_{000}\frac{\partial v}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v} + \Xi_{1000}\frac{\partial \sigma_j}{\partial v} + \Xi_{100}\frac{\partial u}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v}\right.$$
$$\left. + 2\left(\Xi_{000}\frac{\partial v}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v} + \Xi_{100}\frac{\partial u}{\partial x}\frac{\partial^2 \sigma_j}{\partial u \partial v}\right)\right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{0000}\frac{\partial \sigma_j}{\partial u} + \Xi_{1000}\frac{\partial \sigma_j}{\partial v} + 3\left(\Xi_{000}\frac{\partial v}{\partial x} + \Xi_{100}\frac{\partial u}{\partial x}\right)\frac{\partial^2 \sigma_j}{\partial u \partial v}\right]$$

With the auxiliary equation:

$$0 = \sum_{j=1}^{4} y_j^c \left[\Xi_{0000}\frac{\partial \sigma_j}{\partial u} + \Xi_{1000}\frac{\partial \sigma_j}{\partial v} + 3\left(\Xi_{000}\frac{\partial v}{\partial x} + \Xi_{100}\frac{\partial u}{\partial x}\right)\frac{\partial^2 \sigma_j}{\partial u \partial v}\right]$$

Under an exchange of $\partial x \iff \partial y$:

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{0111} \frac{\partial \sigma_j}{\partial u} + \Xi_{1111} \frac{\partial \sigma_j}{\partial v} + 3 \left(\Xi_{011} \frac{\partial v}{\partial y} + \Xi_{111} \frac{\partial u}{\partial y} \right) \frac{\partial^2 \sigma_j}{\partial u \partial v} \right]$$

$$0 = \sum_{j=1}^{4} y_j^c \left[\Xi_{0111} \frac{\partial \sigma_j}{\partial u} + \Xi_{1111} \frac{\partial \sigma_j}{\partial v} + 3 \left(\Xi_{011} \frac{\partial v}{\partial y} + \Xi_{111} \frac{\partial u}{\partial y} \right) \frac{\partial^2 \sigma_j}{\partial u \partial v} \right]$$

Now we need a mixed derivative:

$$\frac{\partial^3}{\partial x^2 \partial y} x(u,v) = \sum_{j=1}^{4} x_j^c \frac{\partial^2}{\partial x^2} \left(\frac{\partial u}{\partial y} \frac{\partial \sigma_j}{\partial u} + \frac{\partial v}{\partial y} \frac{\partial \sigma_j}{\partial v} \right)$$

$$0 = \sum_{j=1}^{4} x_j^c \frac{\partial}{\partial x} \left[\frac{\partial^2 u}{\partial x \partial y} \frac{\partial \sigma_j}{\partial u} + \frac{\partial^2 v}{\partial x \partial y} \frac{\partial \sigma_j}{\partial v} + \frac{\partial^2 \sigma_j}{\partial u \partial v} \left(\frac{\partial u}{\partial y} \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial u}{\partial x} \right) \right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\frac{\partial^3 u}{\partial x^2 \partial y} \frac{\partial \sigma_j}{\partial u} + \frac{\partial^2 u}{\partial x \partial y} \left(\frac{\partial}{\partial x} \frac{\partial \sigma_j}{\partial u} \right) + \frac{\partial^3 v}{\partial x^2 \partial y} \frac{\partial \sigma_j}{\partial v} + \frac{\partial^2 v}{\partial x \partial y} \left(\frac{\partial}{\partial x} \frac{\partial \sigma_j}{\partial v} \right) \right.$$
$$\left. + \cancel{\left(\frac{\partial}{\partial x} \frac{\partial^2 \sigma_j}{\partial u \partial v} \right)} \left(\frac{\partial u}{\partial y} \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial u}{\partial x} \right) + \frac{\partial^2 \sigma_j}{\partial u \partial v} \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial y} \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial u}{\partial x} \right) \right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{0001} \frac{\partial \sigma_j}{\partial u} + \Xi_{001} \frac{\partial v}{\partial x} \frac{\partial^2 \sigma_j}{\partial u \partial v} + \Xi_{1001} \frac{\partial \sigma_j}{\partial v} + \Xi_{101} \frac{\partial u}{\partial x} \frac{\partial^2 \sigma_j}{\partial u \partial v} \right.$$
$$\left. + \frac{\partial^2 \sigma_j}{\partial u \partial v} \left(\frac{\partial^2 u}{\partial x \partial y} \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial x \partial y} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial^2 u}{\partial x^2} \right) \right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{0001} \frac{\partial \sigma_j}{\partial u} + \Xi_{001} \frac{\partial v}{\partial x} \frac{\partial^2 \sigma_j}{\partial u \partial v} + \Xi_{1001} \frac{\partial \sigma_j}{\partial v} + \Xi_{101} \frac{\partial u}{\partial x} \frac{\partial^2 \sigma_j}{\partial u \partial v} \right.$$
$$\left. + \frac{\partial^2 \sigma_j}{\partial u \partial v} \left(\Xi_{001} \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \Xi_{100} + \Xi_{101} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \Xi_{000} \right) \right]$$

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{0001} \frac{\partial \sigma_j}{\partial u} + \Xi_{1001} \frac{\partial \sigma_j}{\partial v} \right.$$
$$\left. + \frac{\partial^2 \sigma_j}{\partial u \partial v} \left(2\Xi_{001} \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \Xi_{100} + 2\Xi_{101} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \Xi_{000} \right) \right]$$

And companion equation:

$$0 = \sum_{j=1}^{4} y_j^c \left[\Xi_{0001} \frac{\partial \sigma_j}{\partial u} + \Xi_{1001} \frac{\partial \sigma_j}{\partial v} + \frac{\partial^2 \sigma_j}{\partial u \partial v} \left(2\Xi_{001} \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \Xi_{100} + 2\Xi_{101} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \Xi_{000} \right) \right]$$

Again, under the exchange of $\partial x \iff \partial y$:

$$0 = \sum_{j=1}^{4} x_j^c \left[\Xi_{0011} \frac{\partial \sigma_j}{\partial u} + \Xi_{1011} \frac{\partial \sigma_j}{\partial v} + \frac{\partial^2 \sigma_j}{\partial u \partial v} \left(2\Xi_{001} \frac{\partial v}{\partial y} + \frac{\partial u}{\partial x} \Xi_{111} + 2\Xi_{101} \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \Xi_{011} \right) \right]$$

$$0 = \sum_{j=1}^{4} y_j^c \left[\Xi_{0011} \frac{\partial \sigma_j}{\partial u} + \Xi_{1011} \frac{\partial \sigma_j}{\partial v} + \frac{\partial^2 \sigma_j}{\partial u \partial v} \left(2\Xi_{001} \frac{\partial v}{\partial y} + \frac{\partial u}{\partial x} \Xi_{111} + 2\Xi_{101} \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \Xi_{011} \right) \right]$$

Bibliography

[1] V. Agoshkov and V. Lebedev. “Poincaré-Steklov operators and domain decomposition methods in variational problems”. In: *Computational Processes and Systems* 2 (1985), pp. 173–227.

[2] D. N. Arnold, F. Brezzi, B. Cockburn, and D. Marini. “Discontinuous Galerkin Methods for Elliptic Problems”. In: 2000, pp. 89–101. DOI: 10.1007/978-3-642-59721-3_5. URL: http://link.springer.com/10.1007/978-3-642-59721-3_5.

[3] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. “Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems”. In: *SIAM Journal on Numerical Analysis* 39.5 (Jan. 2002), pp. 1749–1779. ISSN: 0036-1429. DOI: 10.1137/S0036142901384162. URL: http://epubs.siam.org/doi/10.1137/S0036142901384162.

[4] J. R. Askew. *A characteristics formulation of the neutron transport equation in complicated geometries*. Tech. rep. GENERAL STUDIES OF NUCLEAR REACTORS. United Kingdom, 1972, p. 14. URL: http://inis.iaea.org/search/search.aspx?orig_q=RN:38027599.

[5] I. Babuska. “Error-Bounds for Finite Element Method in Numer”. In: *Math* 16 (1971), pp. 322–333.

[6] J. Ballani and D. Kressner. “Matrices with Hierarchical Low-Rank Structures”. In: *Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications : Cetraro, Italy 2015*. Ed. by M. Benzi and V. Simoncini. Cham: Springer International Publishing, 2016, pp. 161–209. ISBN: 978-3-319-49887-4. DOI: 10.1007/978-3-319-49887-4_3. URL: https://doi.org/10.1007/978-3-319-49887-4_3.

[7] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, and M. Savini. “A high order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows”. In: *Turbomachinery - Fluid Dynamics and Thermodynamics, European Conference, 2.* ; 1997, pp. 99–108. ISBN: 90-5204-032-X. URL: https://www.tib.eu/de/suchen/id/tema%3ATEMAM97071562579.

[8] G. I. Bell. "On the Stochastic Theory of Neutron Transport". In: *Nuclear Science and Engineering* 21.3 (1965), pp. 390–401. DOI: 10.13182/NSE65-1. eprint: https://doi.org/10.13182/NSE65-1. URL: https://doi.org/10.13182/NSE65-1.

[9] S. Börm. *H2lib software library*. 2017.

[10] P. S. Brantley and E. W. Larsen. "The Simplified P 3 Approximation". In: *Nuclear Science and Engineering* 134.1 (Jan. 2000), pp. 1–21. ISSN: 0029-5639. DOI: 10.13182/NSE134-01. URL: https://www.tandfonline.com/doi/full/10.13182/NSE134-01.

[11] Y.-A. Chao. "A new and rigorous SP_N theory – Part II: Generalization to GSP_N". In: *Annals of Nuclear Energy* 110 (Dec. 2017), pp. 1176–1196. ISSN: 03064549. DOI: 10.1016/j.anucene.2017.08.020. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454917302414.

[12] Y.-A. Chao. "A new and rigorous SP_N theory – Part III: A succinct summary of the GSP_N theory, the P_3 equivalent $GSP_3^{(3)}$ and implementation issues". In: *Annals of Nuclear Energy* 119 (Sept. 2018), pp. 310–321. ISSN: 03064549. DOI: 10.1016/j.anucene.2018.04.029. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454920304667%20https://linkinghub.elsevier.com/retrieve/pii/S0306454918302147.

[13] Y.-A. Chao. "A new and rigorous SP N theory for piecewise homogeneous regions". In: *Annals of Nuclear Energy* 96 (Oct. 2016), pp. 112–125. ISSN: 03064549. DOI: 10.1016/j.anucene.2016.06.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S030645491630398X.

[14] Y.-A. Chao, L. Peng, and C. Tang. "A new and rigorous SP_N theory – Part IV: Numerical qualification of $GSP_3^(0)$ and the generalized transverse integration nodal method". In: *Annals of Nuclear Energy* 149 (Dec. 2020), p. 107768. ISSN: 03064549. DOI: 10.1016/j.anucene.2020.107768. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306454920304667.

[15] R. Courant. "Variational methods for the solution of problems of equilibrium and vibrations". In: *Bulletin of the American Mathematical Society* 49 (1943), pp. 1–23. URL: https://api.semanticscholar.org/CorpusID:16547277.

[16] J. Doriath, J. Ruggieri, G. Buzzi, J. Rieunier, P. Smith, P. Fougeras, and A. Maghnouj. "Reactor analysis using a variational nodal method implemented in the ERANOS system". In: *Proc. Topl. Mtg. Advances in Reactor Physics*. 1994, pp. 11–15.

[17] B. Finlayson and L. Scriven. "The method of weighted residuals–a review". In: *Applied Mechanics Reviews* (1966), pp. 735–748.

[18] B. A. Finlayson. *The Method of Weighted Residuals and Variational Principles*. Philadelphia, PA: Society for Industrial and Applied Mathematics, Dec. 2013. ISBN: 978-1-61197-323-5. DOI: 10.1137/1.9781611973242. URL: http://epubs.siam.org/doi/book/10.1137/1.9781611973242.

[19] H. Finnemann, F. Bennewitz, and M. R. Wagner. "Interface Current Techniques for Multidimensional Reactor Calculations". In: *Atomkernenergie* 30.2 (1977), pp. 123–128. URL: https://inis.iaea.org/search/search.aspx?orig_q=RN:9350107.

[20] C. B. C. G. Palmiotti and E. E. Lewis. "Variational Nodal Transport Methods with Anisotropic Scattering". In: *Nuclear Science and Engineering* 115.3 (1993), pp. 233–243. DOI: 10.13182/NSE92-110. eprint: https://doi.org/10.13182/NSE92-110. URL: https://doi.org/10.13182/NSE92-110.

[21] E. Gelbard. "SIMPLIFIED SPHERICAL HARMONICS EQUATIONS AND THEIR USE IN SHIELDING PROBLEMS". In: (1961).

[22] P. Ghysels, X. S. Li, F.-H. Rouet, S. Williams, and A. Napov. *An efficient multi-core implementation of a novel HSS-structured multifrontal solver using randomized sampling*. 2015. arXiv: 1502.07405 [cs.MS].

[23] W. Hackbusch. "A sparse matrix arithmetic based on H-matrices. I. Introduction to $\mathscr{H}$-matrices". In: *Computing* 62.2 (1999), pp. 89–108. ISSN: 0010-485X. DOI: 10.1007/s006070050015.

[24] W. Hackbusch, L. Grasedyck, and S. Börm. "An introduction to hierarchical matrices". In: *Mathematica bohemica* 127.2 (2002), pp. 229–241. ISSN: 0862-7959. URL: http://www.mis.mpg.de/de/publications/preprints/2001/prepr2001-105.html.

[25] W. Hackbusch and B. N. Khoromskij. "A sparse $\mathscr{H}$-matrix arithmetic. Part II. Application to multi-dimensional problems". In: *Computing* 64.1 (2000), pp. 21–47. ISSN: 0010-485X. DOI: 10.1007/s006070050002. URL: http://www.mis.mpg.de/de/publications/preprints/1999/prepr1999-22.html.

[26] W. Hackbusch, B. N. Khoromskij, and S. A. Sauter. "On $\mathscr{H}^2$-matrices". In: *Lectures on applied mathematics : proceedings of the symposium organized by the Sonderforschungsbereich 438 on the occasion of Karl-Heinz Hoffmann's 60th birthday, Munich, June 30 - July 1, 1999*. Ed. by H.-J. Bungartz, R. H. W. Hoppe, and C. Zenger. Berlin [u. a.]: Springer, 2000, pp. 9–29. ISBN: 3-540-66734-2. URL: http://www.mis.mpg.de/de/publications/preprints/1999/prepr1999-50.html.

[27] S. Hao and P.-G. Martinsson. "A direct solver for elliptic PDEs in three dimensions based on hierarchical merging of Poincaré–Steklov operators". In: *Journal of Computational and Applied Mathematics* 308 (Dec. 2016), pp. 419–434. ISSN: 03770427. DOI: 10.

1016/j.cam.2016.05.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377042716302308.

[28] B. T. Helenbrook. "On the Existence of Explicit hp-Finite Element Methods Using Gauss–Lobatto Integration on the Triangle". In: *SIAM Journal on Numerical Analysis* 47.2 (2009), pp. 1304–1318. DOI: 10.1137/070685439.

[29] M. Hussaini, D. Kopriva, and A. Patera. "Spectral collocation methods". In: *Applied Numerical Mathematics* 5.3 (1989), pp. 177–208. ISSN: 0168-9274. DOI: https://doi.org/10.1016/0168-9274(89)90033-0. URL: https://www.sciencedirect.com/science/article/pii/0168927489900330.

[30] D. Johnson. "Chebyshev Polynomials in the Spectral Tau Method and Applications to Eigenvalue Problems". In: *NASA Contractor Report* (1996).

[31] B. Kochunas, B. Collins, D. Jabaay, T. J. Downar, and W. R. Martin. "Overview of development and design of MPACT: Michigan parallel characteristics transport code". In: (July 2013). URL: https://www.osti.gov/biblio/22212692.

[32] B. Kochunas, Y. Xu, and T. Downar. "Development of the triangular nodal method tripen for prismatic HTR analysis". In: *International Conference on the Physics of Reactors 2010, PHYSOR 2010* 2 (Jan. 2010), pp. 1378–1388.

[33] R. Kriemann. "HLIBpro". In: *ULR: http://www. hlibpro. com* (2012).

[34] C. Lanczos. "Trigonometric Interpolation of Empirical and Analytical Functions". In: *Journal of Mathematics and Physics* 17.1-4 (1938), pp. 123–199. DOI: https://doi.org/10.1002/sapm1938171123. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sapm1938171123. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm1938171123.

[35] E. W. Larsen and G. C. Pomraning. "The P N Theory as an Asymptotic Limit of Transport Theory in Planar Geometry—I: Analysis". In: *Nuclear Science and Engineering* 114.1 (May 1993), pp. 86–86. ISSN: 0029-5639. DOI: 10.13182/NSE93-A24018. URL: https://www.tandfonline.com/doi/full/10.13182/NSE93-A24018.

[36] R. D. Lawrence. "The DIF3D Nodal Neutronics Option for Two- and Three-Dimensional Diffusion Theory Calculations in Hexagonal Geometry". In: *Technology* March 1983 (1983).

[37] R. D. Lawrence. "Progress in nodal methods for the solution of the neutron diffusion and transport equations". In: *Progress in Nuclear Energy* 17.3 (Jan. 1986), pp. 271–301. ISSN: 01491970. DOI: 10.1016/0149-1970(86)90034-X. URL: https://www.sciencedirect.com/science/article/pii/014919708690034X.

[38] H. C. Lee and C. H. Kim. "Unified Formulation of Nodal Expansion Method and Analytic Nodal Method Solutions to Two-Group Diffusion Equations". In: *Nuclear Science and Engineering* 138.2 (2001), pp. 192–204. ISSN: 0029-5639. DOI: 10.13182/nse01-a2209.

[39] E. E. Lewis, M. A. Smith, N. Tsoulfanidis, G. Palmiotti, T. A. Taiwo, and R. N. Blomquist. "Benchmark specification for Deterministic 2-D/3-D MOX fuel assembly transport calculations without spatial homogenisation (C5G7 MOX)". In: 2001. URL: https://api.semanticscholar.org/CorpusID:125614806.

[40] E. Lewis and I. Dilber. "Finite element, nodal and response matrix methods: A variational synthesis for neutron transport". In: *Progress in Nuclear Energy* 18.1 (1986). Finite Element and Allied Methods, pp. 63–74. ISSN: 0149-1970. DOI: https://doi.org/10.1016/0149-1970(86)90013-2. URL: https://www.sciencedirect.com/science/article/pii/0149197086900132.

[41] X. S. Li. "An Overview of SuperLU: Algorithms, Implementation, and User Interface". In: 31.3 (Sept. 2005), pp. 302–325.

[42] D. Litskevich and B. Merk. "SP3 Solution versus Diffusion Solution in Pin-by-Pin Calculations and Conclusions Concerning Advanced Methods". In: *Journal of Computational and Theoretical Transport* 43.1-7 (2014), pp. 214–239. DOI: 10.1080/00411450.2014.913184.

[43] X. Liu, J. Xia, and M. De Hoop. "Fast factorization update for general elliptic equations under multiple coefficient updates". In: *SIAM Journal on Scientific Computing* 42.2 (2020), A1174–A1199.

[44] P. G. Maginot, J. C. Ragusa, and J. E. Morel. "Lumping Techniques for DFEM SN Transport in Slab Geometry". In: *Nuclear Science and Engineering* 179 (2015), pp. 148–163. URL: https://api.semanticscholar.org/CorpusID:123730397.

[45] P. G. Martinsson. "The Hierarchical Poincare-Steklov (HPS) solver for elliptic PDEs". In: (June 2015). arXiv: 1506.01308. URL: http://arxiv.org/abs/1506.01308.

[46] P.-G. Martinsson. *Fast Direct Solvers for Elliptic PDEs*. Philadelphia: SIAM, 2020.

[47] R. G. McClarren. "Theoretical Aspects of the Simplified P n Equations". In: *Transport Theory and Statistical Physics* 39.2-4 (Mar. 2010), pp. 73–109. ISSN: 0041-1450. DOI: 10.1080/00411450.2010.535088. URL: http://www.tandfonline.com/doi/abs/10.1080/00411450.2010.535088.

[48] E. Newman and K. Kingsley. “An introduction to the method of moments”. In: *Computer Physics Communications* 68.1 (1991), pp. 1–18. ISSN: 0010-4655. DOI: https://doi.org/10.1016/0010-4655(91)90191-M. URL: https://www.sciencedirect.com/science/article/pii/001046559190191M.

[49] J. Nitsche. “Lineare spline-funktionen und die methoden von Ritz für elliptische randwertprobleme”. In: *Archive for Rational Mechanics and Analysis* 36.5 (1970), pp. 348–355.

[50] J. M. Noh and N. Z. Cho. “A New Approach of Analytic Basis Function Expansion to Neutron Diffusion Nodal Calculation”. In: *Nuclear Science and Engineering* 116.3 (1994), pp. 165–180. DOI: 10.13182/NSE94-A19811. eprint: https://doi.org/10.13182/NSE94-A19811. URL: https://doi.org/10.13182/NSE94-A19811.

[51] G. Palmiotti, E. E. Lewis, and C. Carrico. *VARIANT: VARIational Anisotropic Nodal Transport for multidimensional cartesian and hexadgonal geometry calculation*. Argonne National Laboratory Argonne, IL, 1995.

[52] S. P. Palmtag. *A two-dimensional, semi-analytic expansion method for nodal calculations*. Tech. rep. Oak Ridge, TN (United States): Oak Ridge Institute for Science and Education (ORISE), Aug. 1995. DOI: 10.2172/505709. URL: http://www.osti.gov/servlets/purl/505709/.

[53] R. Piessens, E. de Doncker-Kapenga, C. W. Überhuber, and D. K. Kahaner. *Quadpack: a subroutine package for automatic integration*. Vol. 1. Springer Science & Business Media, 2012.

[54] G. Pomraning. “Asymptotic and variational derivations of the simplified PN equations”. In: *Annals of Nuclear Energy* 20.9 (Sept. 1993), pp. 623–637. ISSN: 03064549. DOI: 10.1016/0306-4549(93)90030-S. URL: https://linkinghub.elsevier.com/retrieve/pii/030645499390030S.

[55] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Ed. by R. Glowinski, Y. Kuznetsov, G. Meurant, J. Périaux, and O. Widlund. SIAM, 1991, pp. 58–81.

[56] F.-H. Rouet, X. S. Li, P. Ghysels, and A. Napov. “A Distributed-Memory Package for Dense Hierarchically Semi-Separable Matrix Computations Using Randomization”. In: *ACM Trans. Math. Softw.* 42.4 (June 2016). ISSN: 0098-3500. DOI: 10.1145/2930660. URL: https://doi.org/10.1145/2930660.

[57] D. Selengut. “A New Form of the P3 Approximation”. In: *Trans. Amer. Nucl. Soc. 13: 437-888(Nov 1970)*. (Jan. 1970), pp. 625–626. URL: https://www.osti.gov/biblio/4090881.

[58] K. Smith. “An analytic nodal method for solving the two-group, multidimensional, static and transient neutron diffusion equations /”. In: (Aug. 2005).

[59] Y. XU. “ON GAUSS—LOBATTO INTEGRATION ON THE TRIANGLE”. In: *SIAM Journal on Numerical Analysis* 49.1/2 (2011), pp. 541–548. ISSN: 00361429. URL: http://www.jstor.org/stable/23074410 (visited on 08/16/2023).

[60] R. Yokota, G. Turkiyyah, and D. Keyes. *Communication Complexity of the Fast Multipole Method and its Algebraic Variants*. 2014. arXiv: 1406.1974 [cs.DC].

[61] T. Zhang and Z. Li. “Variational nodal methods for neutron transport: 40 years in review”. In: *Nuclear Engineering and Technology* 54.9 (2022), pp. 3181–3204. ISSN: 1738-5733. DOI: https://doi.org/10.1016/j.net.2022.04.012. URL: https://www.sciencedirect.com/science/article/pii/S1738573322002157.

[62] M. Zlámal. “On the finite element method”. In: *Numerische Mathematik* 12 (1968), pp. 394–409.