Meshless Methods for the Neutron Transport Equation

by

Brody R. Bassett

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
2018

Doctoral Committee:

Assistant Professor Brian C. Kiedrowski, Chair
Professor Thomas J. Downar
Associate Professor Krzysztof J. Fidkowski
Professor Edward W. Larsen

Brody R. Bassett

brbass@umich.edu

ORCID ID: 0000-0003-3310-9451

Contents

List of Algorithms

List of Figures

List of Tables

List of Appendices

List of Acronyms

CASL	Consortium for Advanced Simulation of Light Water Reactors
CFEM	Continuous finite element method
CSG	Constructive solid geometry
DEM	Diffuse element method
DFEM	Discontinuous finite element method
EFG	Element-free Galerkin
FEM	Finite element method
GA	Gaussian
GMRES	Generalized minimal residual method
IAEA	International Atomic Energy Agency
ICSBEP	International Criticality Safety Benchmark Evaluation Project
IFBA	Integral fuel burnable absorber
ILUT	Incomplete LU decomposition with threshold
IMQ	Inverse multiquadric
LDFE	Linear discontinuous finite element
MLPG	Meshless local Petrov-Galerkin
MLS	Moving least squares
MMS	Method of manufactured solutions
MOOSE	Multiphysics Object Oriented Simulation Environment
MQ	Multiquadric
PDE	Partial differential equation
RBF	Radial basis function
SAAF	Self adjoint angular flux
SPH	Smoothed particle hydrodynamics
SUPG	Streamline upwind Petrov-Galerkin
VERA	Virtual Environment for Reactor Applications

List of Symbols

Continuous-energy transport

ψ	Angular flux
ψ^b	Homogeneous angular flux on boundary
ψ^{inc}	Nonhomogeneous angular flux on boundary
ψ^{init}	Initial angular flux
ℓ	Spherical harmonic degree index
m	Spherical harmonic order index
P_ℓ	Legendre polynomials
Y_ℓ^m	Spherical harmonics
ϕ_ℓ^m	Spherical harmonic moments of the angular flux
v	Velocity
N	Neutron density
$\boldsymbol{x}$	Position
$\boldsymbol{\Omega}$	Direction
E	Energy
t	Time
Σ_t	Total cross section
Σ_s	Scattering cross section
χ	Fission spectrum
Σ_f	Fission cross section
ν	Average number of neutrons created in fission event
q	Nonhomogeneous source
ρ	Albedo boundary constant
V	The problem domain
∂V	The problem boundary
$\boldsymbol{n}$	Boundary normal direction
k	k-eigenvalue

Multigroup transport

g	Group index
G	Total number of groups
E_g	Lower energy bound for group g
ψ_g	Multigroup angular flux
$\phi_{\ell,g}^m$	Multigroup flux moments

(transport variables with g index, e.g. $\Sigma_{t;g}$)

Discrete ordinates transport

n	Direction index
N	Number of directions
L	Number of spherical harmonic degrees
$\boldsymbol{\Omega}_n$	Discrete direction
c_n	Directional integration weights
$\psi_{n,g}$	Discrete angular flux
$q_{n,g}$	Discrete nonhomogeneous source

Operators for the transport equation

$\mathcal{L}$	Streaming and collision operator
$\mathcal{S}$	Scattering operator
$\mathcal{F}$	Fission operator
$\mathcal{M}$	Moment-to-discrete operator
$\mathcal{D}$	Discrete-to-moment operator
$\mathcal{R}$	Reflection operator
$\mathcal{I}$	Identity operator
$\mathcal{P}$	Preconditioner to the $\mathcal{L}$ operator
$\mathcal{B}$	Basis function summation operator

Spatial discretization

i	Basis function index
j	Weight function index
J	Number of spatial points
b_i	Basis function
w_j	Weight function
$\tilde{w}_{j,n}$	SUPG-modified weight function
τ	SUPG proportionality constant
$\alpha_{i,n,g}$	Expansion coefficients for angular flux
$\alpha_{i,n,g}^b$	Expansion coefficients on the boundary
$\beta_{i,\ell,g}^m$	Expansion coefficients for moments of angular flux

Integral values

$\bar{i}$	Integral values involving b_i and w_j

(transport variables with overline, e.g. $\overline{\Sigma}_{t;i,j,g}$)

Meshless functions

r	Dimensionless distance
Γ	Dimensionless meshless function
g_i	Single RBF function in a set
ϵ_i	Shape parameter
d	Distance measure
r_{sup}	Support radius of function
h_i	Single MLS function in a set

Meshless integration

ξ, η, ζ	Localized coordinates for integrals
J	Jacobian determinant

Coupled heat conduction

T	Temperature
ρ	Mass density
c_p	Specific heat at constant pressure
$\dot{q}$	Volumetric heat generation rate
k	Thermal conductivity
h	Convection (heat transfer) coefficient
T_∞	Convective fluid temperature
ζ_i	Expansion coefficients for temperature
σ	Microscopic neutron cross section
Q	Total volumetric heat generation rate
κ	Energy release from fission

(spatial discretization variables for basis and weight)

Constructive solid geometry

$\boldsymbol{g}_0$	Initial location of particle
$\boldsymbol{g}_1$	Direction of particle
s	Distance of particle from initial location

Abstract

Mesh-based methods for the numerical solution of partial differential equations (PDEs) require the division of the problem domain into non-overlapping, contiguous subdomains that conform to the problem geometry. The mesh constrains the placement and connectivity of the solution nodes over which the PDE is solved. In meshless methods, the solution nodes are independent of the problem geometry and do not require a mesh to determine connectivity. This allows the solution of PDEs on geometries that would be difficult to represent using even unstructured meshes.

The ability to represent difficult geometries and place solution nodes independent of a mesh motivates the use of meshless methods for the neutron transport equation, which often includes spatially-dependent PDE coefficients and strong localized gradients. The meshless local Petrov-Galerkin (MLPG) method is applied to the steady-state and k-eigenvalue neutron transport equations, which are discretized in energy using the multigroup approximation and in angle using the discrete ordinates approximation. The MLPG method uses weighted residuals of the transport equation to solve for basis function expansion coefficients of the neutron angular flux. Connectivity of the solution nodes is determined by the shared support domain of overlapping meshless functions, such as radial basis functions (RBFs) and moving least squares (MLS) functions.

To prevent oscillations in the neutron flux, the MLPG transport equation is stabilized by the streamline upwind Petrov-Galerkin (SUPG) method, which adds numerical diffusion to the streaming term. Global neutron conservation is enforced by using MLS basis and weight functions and appropriate SUPG parameters. The cross sections in the transport equation are approximated in accordance with global particle balance and without constraint on their spatial dependence or the location of the basis and weight functions. The equations for the strong-form meshless collocation approach are derived for comparison to the MLPG equations. Two integration schemes for the basis and weight functions in the MLPG method are presented, including a background mesh integration and a fully meshless integration approach.

The method of manufactured solutions (MMS) is used to verify the resulting MLPG method in one, two and three dimensions. Results for realistic problems, including two-dimensional pincells, a reflected ellipsoid and a three-dimensional problem with voids, are verified by comparison to Monte Carlo simulations. Finally, meshless heat transfer equations are derived using a similar MLPG approach and verified using the MMS.

These heat equation are coupled to the MLPG neutron transport equations, and results for a pincell are compared to values from a commercial pressurized water reactor.

Chapter 1

Introduction

Commonly-used methods for solving the Boltzmann transport equation, such as the discontinuous finite element method (DFEM) and the finite difference method, require separating the domain into discrete spatial cells or elements, each of which often has constant material properties and localized solution nodes. The meshing process can be computationally expensive and labor-intensive, particularly for unstructured meshes. Meshless methods, such as the meshless local Petrov-Galerkin (MLPG) method considered here, do not require a mesh, which facilitates more flexibility in the placement of solution nodes in the problem and specification of the problem-dependent coefficients, such as the cross sections for the neutron transport equation.

The application of the MLPG discretization to the neutron transport equation involves approximations to the cross sections, calculation of meshless functions, representation of the geometry, stabilization of the equations to prevent oscillations, and optimizations to improve computational cost. Section 1.1 presents a general history of meshless methods. Section 1.2 discusses past applications of meshless methods to the transport equation and challenges to consider when solving the neutron transport equation. Section 1.3 discusses the novel work in this dissertation, including the differences from past work on meshless transport and on mesh-based methods for neutron transport. Section 1.4 presents an overview of the remainder of this dissertation.

1.1 History of meshless methods

Smoothed particle hydrodynamics (SPH), the first modern meshless method, was developed in 1977 for use in astrophysics problems [1, 2]. In the SPH method, fluid motion is represented using moving particles smoothed by kernel functions, which contrasts with mesh-based methods in which the fluid moves through stationary mesh elements (Eulerian), the mesh moves with the fluid (Lagrangian), or the two are combined (Arbitrary Lagrangian-Eulerian [3]). The SPH method has been applied to astrophysics, fluid dynamics, explosion mechanics, solid mechanics, heat conduction and magnetohydrodynamics [4]. Boundary conditions are difficult to enforce using the SPH method, and as such many of the applications include unbounded

problems. The reproducing kernel particle method developed in 1995 includes corrections to SPH [5] and has been applied to similar problems. Other meshless methods for fluid flow and solid dynamics include the finite point method developed in 1996 [6] and the point interpolation method developed in 2001 [7].

The collocation approach, in which the PDE is enforced at collocation points to find the coefficients of a basis function expansion, was developed in 1990 [8] and has been applied to a diverse set of PDEs [9], including fluid dynamics, heat conduction, convection-diffusion, transport, and diffusion problems [10]. With the original global basis functions, the PDE solution matrix is dense and difficult to solve. The accuracy of the method increases as the basis functions become essentially flat, but this makes the linear system severely ill-conditioned [11]. For advection problems, the streaming operator cannot be easily stabilized. Despite these drawbacks, the collocation method is simple to apply and accurate for many problems.

The diffuse element method (DEM) developed in 1992 [12] applies the Galerkin method of weak-form residuals using moving least squares (MLS) functions. The element-free Galerkin (EFG) approach developed in 1994 extends the DEM method with more accurate derivatives, a background integration mesh independent of the solution nodes, and accurate enforcement of essential boundary conditions [13]. The EFG method has been successfully applied to a variety of problems, including fracture, fluid flow, and heat transfer.

The meshless local Petrov-Galerkin (MLPG) method, first described in 1998 [14], adds additional properties to the weak-form solution that are useful for the solution of PDEs. In the original approach, local compact Gaussian and spline functions are used to create the MLS basis. In the MLPG method, the basis and weight functions have limited spatial support, and connectivity is determined by the overlapping support regions of the basis and weight functions. Unlike the DEM and EFG methods, the MLPG approach uses the local weak form, which permits integration without a background mesh and creates sparse solution matrices. While the MLPG solution can still exhibit oscillations, as in the collocation approach, the additional freedom to specify the weight functions independent of the basis functions allows for the use of common stabilization techniques used in other weighted residual methods, such as the streamline upwind Petrov-Galerkin (SUPG) method [15, 16].

There exist a variety of other meshless methods not covered here. For an overview of meshless methods and their mathematical properties, see Ref. [17].

1.2 Application of meshless methods to transport

Transport codes have several features that require special attention when applying meshless methods:

- The advective streaming term may necessitate stabilization.
- Boundary conditions should be accurately represented.

- The material properties can be variable in space.
- Due to the dependence on space, energy and direction, the number of unknowns is much higher than in many other applications.
- Particle balance should be preserved if possible.

The methods such as SPH that are commonly applied to fluid flow and solid dynamics usually represent the interactions of a medium with itself, whereas in the neutron transport equation the neutrons do not interact with each other but instead with the background material. In addition, these methods can struggle with boundary representation, which makes their application to transport problematic. The radiative transfer equations have been coupled to SPH hydrodynamics simulations in astrophysics where the boundary conditions are neglected, but the transport is either between individual particles or with simplified transport physics [18].

Section 1.2.1 presents a history of applications of meshless methods to the transport equation. The methods that have been successful in solving the transport equation are those designed for interpolation and the solution of general PDEs, such as the collocation and MLPG approaches. Section 1.2.2 discusses difficulties in solving the neutron transport equation, including the differences between neutron and radiative transfer and challenges for other discretization methods for the neutron transport equations.

1.2.1 History of meshless methods for transport

Early work applying meshless methods to transport utilized common methods such as the fast Fourier transform [19] and Chebyshev spectral methods [20] for time-dependent radiative transfer. The first few studies using more flexible meshless methods appeared thereafter, in which least squares collocation methods were used for radiative heat transfer. The even-parity transport equation was shown to be stable and accurate for radiative transfer problems, while the primitive variables approach was shown to be unstable in regions with low absorption [21, 22]. Various meshless methods were successfully applied to solve the coupled radiative transport and conductive heat transfer equations [23, 24], including the use of MLPG with moving least squares (MLS) basis functions and weight functions skewed along the upwind direction for stabilization [25], which is similar to the spatial discretization used here.

The neutron transport equation has not been extensively studied using meshless methods. In 2017, Kashi et. al used meshless basis functions to solve the weak transport equation inside each homogeneous element in a two-dimensional Cartesian finite element mesh, producing a method very similar to the discontinuous finite element method (DFEM) but with basis functions commonly used in meshless methods [26]. Despite

Table 1.1: Representative review of transport work on meshless methods

First author	Reference	Year	First-order	Second-order	Radiative transfer	Heat conduction	Neutron transport	Localized basis	Heterogeneous	Energy-dependent*	Strong form	Weak form	Stabilized	SUPG	Particle balance	Dimension	Max points	Max directions
Sadat	[21]	2006	✓	✓	✓			✓			✓					2	400	24
Liu	[25]	2007	✓		✓	✓		✓				✓	✓			2	441	24
Liu	[22]	2007	✓		✓			✓			✓					2	961	24
Tan	[29]	2009	✓	✓	✓			✓			✓					2	1700	40
Kindelan	[23]	2010	✓		✓				✓		✓					2	961	64
Wang	[30]	2010		✓	✓			✓			✓					3	92500	48
Sadat	[24]	2012		✓	✓	✓		✓			✓					3	3375	48
Zhao	[31]	2013		✓	✓			✓	✓		✓					2	1899	1296
Wang	[32]	2016	✓		✓	✓										2	1144	800
Kashi	[26]	2017	✓				✓		†	✓		✓				2	6†	144
Current study		2018	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓‡	3	74088	8192

*Refers to energy dependence of the transport equation, not the energy equation used in coupled radiative and conductive heat transfer.
†Uses a mesh with up to 6 nodes in each element. Inside of each element, the material is homogeneous.
‡Particle balance preserved for weak form.

the differences in the source term and energy dependence, many of the meshless techniques used in other transport problems are applicable to neutron transport.

The neutron diffusion equation is stable and simple to solve using meshless methods. The element-free Galerkin method with moving least squares basis and weight functions has been applied to one group diffusion with spatially-varying cross sections in one and two dimensions [27]. The multigroup diffusion problem has been solved using the collocation method with global basis functions [28].

Table 1.1 includes a summary of the methods that have been used to solve the transport equation. Most past studies have been on the radiative transfer equation, sometimes with coupled heat conduction. Most cases use the strong form of the transport equation, which prevents application of stabilization to the advection operator. Heterogeneous material properties have only been briefly considered in two cases and only for strong-form approaches. None of the historical methods reviewed preserve particle balance. The only case of stabilization reviewed includes skewed or shifted weight functions, which prevents particle balance and creates directionally-dependent weak-form integrals. Neutron transport presents additional difficulties discussed in Sec. 1.2.2 that require consideration.

1.2.2 Challenges specific to neutron transport

There are a few important distinctions between the radiative transfer and neutron transport equation. Some common approximations made in radiative transfer meshless methods, such as energy independence, homogeneous materials and directionally-dependent weak-form integrals, would be unsuitable for neutron transfer. While the radiative transfer equation does include frequency (or energy) dependence that can be converted into a multigroup form [33], the gray approximation is often used, which neglects energy dependence. Neutron transport is heavily dependent on energy due to the different behavior of neutrons in fast, resonance and thermal energies.

A primary use for neutron transport is in nuclear reactors, which have heterogeneous, discontinuous cross sections that make accurate integration expensive. For some of the problems considered here, directionally-dependent integrals would take several CPU years to complete without additional optimizations. For this dissertation, the energy dependence is discretized using the multigroup approximation, the materials are assumed to be spatially-dependent, and the weak-form integration is performed independent of discrete direction dependence. This allows for reasonably efficient solution of the transport equation for realistic problems with energy dependence and a large number of angular directions.

In standard deterministic neutron transport methods, material properties are often assumed to be constant within each cell. Monte Carlo methods with delta tracking [34] or functional expansion tallies [35] more accurately account for spatially varying cross sections, but at a high computational cost. One treatment for spatially varying cross sections in a deterministic multiphysics code is an averaging of the cross section in each cell using a numerical quadrature [36], which does not explicitly guarantee global particle conservation.

For meshless methods, spatially-dependent PDE coefficients (such as the cross sections in the neutron transport equation) present unresolved issues for strong-form solutions. Evaluation of the coefficients at the collocation points [30, 31, 37] makes an implicit assumption that the coefficients are continuous and have small gradients, as the coefficients between these evaluation points are not considered. Localization of the basis functions [38], such as through domain decomposition [39], makes the opposite assumption that coefficients are piecewise constant and requires that the basis functions be tailored to the discontinuities. For a discretization with arbitrary basis functions and coefficients with discontinuities, neither method is as generally applicable as an appropriate approximation using the weak form of the transport equation. In the MLPG discretization, spatially variable PDE coefficients can be accounted for in the integration step, which allows for proof of convergence to the correct weak-form limit (see Sec. 2.7).

Another area where many deterministic radiation transport codes struggle is curved surface representation. Method of characteristics codes can exactly represent curved surfaces but become computationally

expensive for three-dimensional simulations. During the sweep process of a DFEM transport code, an element with a curved surface may have a reentrant condition on the boundary, which leads to an indeterminate sweep order. A few methods to accurately account for higher-order surfaces include choosing a single outward normal for each surface [40], temporarily splitting the high-order cells in a direction-dependent fashion [41], solving for a second-order source on a first-order mesh [36] and assembling and directly solving the full FEM matrix [42]. In the MLPG method with the cross sections in Sec. 2.8, curved surfaces inside the domain are accounted for in the integration step, not the meshing or transport steps.

The MLPG method used here is very similar mathematically to solving the radiation transport equation using a continuous finite element method (CFEM). The self adjoint angular flux (SAAF) form of the transport equation has been discretized using CFEM and implemented with SUPG stabilization [43]. The inverse of the total cross section is used as the SUPG stabilization parameter (see Sec. 2.4) except in voids, which require special treatment in the SAAF equations. Near void regions, a characteristic distance for each element is used, which is similar to the treatment of the SUPG parameter in Sec. 3.2.

1.3 Comparison to past approaches

The approach in this dissertation to the solution of the neutron transport equation differs significantly from past meshless approaches. None of the reviewed meshless approaches for the transport equation ensure particle balance. In the current approach, global particle balance is preserved by using an MLPG discretization with MLS basis and weight functions and SUPG stabilization (Sec. 2.7). For radiation transport, this ensures that the conservation properties of particle balance based on which the transport equation is derived are carried through to the discretized equations [44].

Multigroup continuous and discontinuous cross sections are accurately represented without subdivision of the problem. The past approaches for consideration of heterogeneous cross sections in strong-form methods consist of evaluating the cross sections at the solution nodes and do not accurately solve the problems considered in this dissertation. Heterogeneous cross sections have not previously been considered for weak-form meshless solution of the transport equation. The approach to heterogeneous cross sections considered in this dissertation, including a spatial approximation to the cross sections, preserves particle balance for the weak-form equations (Sec. 2.8). A new cross section approach based on the approximate cross sections from the weak form is derived that more accurately accounts for discontinuous cross sections in the strong form (Sec. 2.9).

Optimizations are presented in Chapter 3 for efficient solution of the meshless discretization of the transport equation. Other authors have described possibilities for parallelization of meshless methods for the transport equation, but parallelization has not been explicitly described and implemented as it is here (Sec.

3.5). The numerical integration of the transport equation is optimized (Chapter 4) to allow for integration of more difficult geometries. The stabilization of the weak-form equations is accomplished without adding directionally-dependent integration (Sec. 2.8), which has not been done previously. The integration can be performed using either a fully meshless approach or a background mesh. Neither integration scheme is completely novel for meshless methods, although some of the meshless integration techniques extend past approaches to more general cases (Sec. 4.1). This is the first comparison of the integration methods for transport applications (Sec. 4.4). Past approaches to meshless transport do not describe optimizations or preconditioners for the inversion of the streaming and collision operator. The current approach includes stabilization that allows for iterative methods with preconditioning (Sec. 3.3). In addition, modern iterative methods are applied to the solution of the full meshless transport system (Sec. 3.4), which extends methods used in mesh-based neutron transport codes to the meshless case, with operators acting directly on expansion coefficients. Combined, the optimizations allow consideration of problems with between two and five orders of magnitude more unknowns than previous applications of meshless methods to the transport equation.

This dissertation combines many of the techniques used previously in meshless transport methods, and adds additional techniques (Table 1.1). Stabilization of the weak form, dependence on energy, consideration of heterogeneous materials and localized basis functions have each been done separately but only once have two of these been done at the same time (for the localized basis and heterogeneous materials). Here, the transport equation is discretized using a localized basis over a heterogeneous, energy-dependent material using both the strong form and the stabilized weak form. This is the first application of the weak-form transport equation to problems in three dimensions (Secs. 5.2, 6.3, and 7.4) and the first application of SUPG to the meshless transport equation.

Within neutron transport, including mesh-based methods, the use of particle balance to derive an approximate cross section technique (Sec. 2.8.2) is novel and could be applied to FEM discretizations. The use of a constructive solid geometry (CSG) as commonly seen in Monte Carlo neutron transport codes is extended here for use in integration of the material properties in the system. The numerical integration of the cross sections with dependence on the basis and optionally the weight functions contrasts with the homogenization techniques used in many FEM neutron transport codes.

Even with optimizations, the MLPG simulations for the transport equation carry a high computational cost compared to many mesh-based methods. The two largest factors affecting the performance of the method are the initialization of the basis and weight function integrals, and the solution of the streaming term of the transport equation. Unlike many mesh-based methods, such as the DFEM, the MLPG discretization does not solve the transport equation separately on small spatial subdomains. Instead, for each direction and energy, the transport equation is solved over the entire spatial domain using a single linear problem. The

storage of the matrices representing these linear problems and preconditioners for these matrices contributes to the high memory cost of the method (see Sec. 3.6).

1.4 Overview

This dissertation is structured as follows. Chapter 2 contains a derivation of the MLPG transport equation with SUPG stabilization from the strong-form equations, including approximations to the neutron cross sections that maintain particle balance over the problem domain. Chapter 3 discusses the implementation of these equations, including the linear solvers and preconditioners used for the solution of the transport equation, choices of parameters for the basis and weight functions, and parallelization of the MLPG code. Results for the performance of the code are included in this chapter, including a discussion of parallel performance and the preconditioners used for the streaming term of the transport equation. In Chapter 4, two integration techniques for the MLPG basis and weight functions are presented, including integration using a background mesh and meshless integration without a background mesh. Convergence results for the integration indicate good agreement for the two integration methods.

The code is verified in Chapter 5 using the method of manufactured solutions. For the problems considered, MLPG equations are stable and accurate for a wide range of parameters, including for discontinuous cross sections. Strong-form results presented for comparison show a strong dependence of the solution on the choice of parameters and inaccurate results for problems with discontinuous cross sections. Chapters 6 and 7 present results for steady-state and eigenvalue problems, respectively, with benchmarks from Monte Carlo simulations. The spatial and angular convergence of the results and the effect of SUPG stabilization are considered. In Chapter 8, the heat conduction equation is discretized using MLPG and coupled to the neutron transport equation. In Chapter 9, the results are summarized and recommendations are made for future research.

Appendix A defines the Legendre polynomials and spherical harmonics used in the transport equation. Appendix B presents equations for the CSG used to represent the material properties in the simulations. The implementation is similar to a CSG in a Monte Carlo code. Appendix C includes an alternate derivation of the strong form of the transport equations with results for simple problems.

Chapter 2

Meshless discretization of the neutron transport equation

The neutron angular flux $\psi(\boldsymbol{x}, \boldsymbol{\Omega}, E, t)$ is defined in terms of the neutron density $N(\boldsymbol{x}, \boldsymbol{\Omega}, E, t)$ and the neutron velocity $v(E)$ as

$$\psi(\boldsymbol{x}, \boldsymbol{\Omega}, E, t) \equiv v(E) N(\boldsymbol{x}, \boldsymbol{\Omega}, E, t),$$

and depends on position $\boldsymbol{x}$, direction $\boldsymbol{\Omega}$, energy E and time t. The neutron transport equation simulates streaming, scattering, absorption and fission of neutrons to calculate the neutron angular flux. The full form of the neutron transport without delayed neutrons is

$$\begin{aligned}\frac{1}{v(E)}\frac{\partial}{\partial t}\psi(\boldsymbol{x}, \boldsymbol{\Omega}, E, t) + \boldsymbol{\Omega}\cdot\boldsymbol{\nabla}\psi(\boldsymbol{x}, \boldsymbol{\Omega}, E, t) + \Sigma_t(\boldsymbol{x}, E)\psi(\boldsymbol{x}, \boldsymbol{\Omega}, E, t) \\ = \int_0^\infty \int_{4\pi} \Sigma_s(\boldsymbol{x}, \boldsymbol{\Omega}'\cdot\boldsymbol{\Omega}, E' \to E)\psi(\boldsymbol{x}, \boldsymbol{\Omega}', E', t)\, d\Omega' dE' \\ + \frac{\chi(\boldsymbol{x}, E)}{4\pi}\int_0^\infty \int_{4\pi} \nu(\boldsymbol{x}, E')\Sigma_f(\boldsymbol{x}, E')\psi(\boldsymbol{x}, \boldsymbol{\Omega}', E', t)\, d\Omega' dE' + q(\boldsymbol{x}, \boldsymbol{\Omega}, E, t), \\ \boldsymbol{x} \in V, \quad \boldsymbol{\Omega} \in 4\pi, \quad 0 < E < \infty, \quad 0 < t, \end{aligned} \tag{2.1a}$$

with the accompanying boundary and initial conditions,

$$\psi(\boldsymbol{x}, \boldsymbol{\Omega}, E, t) = \psi^{inc}(\boldsymbol{x}, \boldsymbol{\Omega}, E, t) + \rho\psi(\boldsymbol{x}, \boldsymbol{\Omega}_r, E, t), \qquad \boldsymbol{x} \in \partial V, \quad \boldsymbol{\Omega}\cdot\boldsymbol{n} < 0, \quad 0 < E < \infty, \quad 0 < t, \tag{2.1b}$$

$$\psi(\boldsymbol{x}, \boldsymbol{\Omega}, E, t = 0) = \psi^{init}(\boldsymbol{x}, \boldsymbol{\Omega}, E), \qquad \boldsymbol{x} \in \partial V, \quad \boldsymbol{\Omega} \in 4\pi, \quad 0 < E < \infty, \tag{2.1c}$$

and the variables

$\psi(\boldsymbol{x}, \boldsymbol{\Omega}, E, t)$	Angular flux at time t in direction $\boldsymbol{\Omega}$ with energy E,
$\psi^{inc}(\boldsymbol{x}, \boldsymbol{\Omega}, E, t)$	Incoming angular flux at the problem boundary,
$\Sigma_t(\boldsymbol{x}, E)$	Total cross section at energy E,
$\Sigma_s(\boldsymbol{x}, \boldsymbol{\Omega}'\cdot\boldsymbol{\Omega}, E' \to E)$	Cross section for scattering from direction $\boldsymbol{\Omega}'$ and energy E' to direction $\boldsymbol{\Omega}$ and energy E,

$\chi(\boldsymbol{x}, E)$ Fission spectrum at energy E,

$\nu(\boldsymbol{x}, E)$ Average number of neutrons created in a fission event with incoming neutron energy E,

$\Sigma_f(\boldsymbol{x}, E)$ Fission cross section at energy E, and

$q(\boldsymbol{x}, \boldsymbol{\Omega}, E, t)$ Internal source at time t in direction $\boldsymbol{\Omega}$ with energy E.

The notation $\boldsymbol{x} \in V$ refers to positions inside the problem domain, while $\boldsymbol{x} \in \partial V$ refers to positions on the problem boundary. The parameter ρ allows for the albedo boundary condition with $\rho = 1$ for a reflective boundary condition and $\rho = 0$ for a vacuum boundary condition. The reflection angle is

$$\boldsymbol{\Omega}_r \equiv \boldsymbol{\Omega} - 2(\boldsymbol{\Omega} \cdot \boldsymbol{n})\boldsymbol{n}. \tag{2.2}$$

Defining the spherical harmonics moments of the equation in terms of the real spherical harmonics functions $Y_\ell^m(\boldsymbol{\Omega})$ as

$$\phi_\ell^m(\boldsymbol{x}, E, t) \equiv \int_{4\pi} Y_\ell^m(\boldsymbol{\Omega})\,\psi(\boldsymbol{x}, \boldsymbol{\Omega}, E, t)\,d\Omega, \tag{2.3}$$

and using a similar expression for the representation of the scattering source in terms of the Legendre polynomials $P_\ell(\mu_0)$ (with the scattering cosine $\mu_0 = \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}$),

$$\Sigma_{s;\ell}(\boldsymbol{x}, E' \to E) \equiv 2\pi \int_{-1}^{1} P_\ell(\mu_0)\,\Sigma_s(\boldsymbol{x}, \mu_0, E' \to E)\,d\mu_0, \tag{2.4}$$

the angular flux and scattering cross section can be expressed as an expansion involving spherical harmonics moments,

$$\psi(\boldsymbol{x}, \boldsymbol{\Omega}, E, t) = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y_\ell^m(\boldsymbol{\Omega})\,\phi_\ell^m(\boldsymbol{x}, E, t), \tag{2.5a}$$

$$\Sigma_s(\boldsymbol{x}, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}, E' \to E) = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} \Sigma_{s;\ell}(\boldsymbol{x}, E' \to E)\,Y_\ell^m(\boldsymbol{\Omega})\,Y_\ell^m(\boldsymbol{\Omega}'). \tag{2.5b}$$

See Appendix A for definitions of the Legendre polynomials and spherical harmonics as well as their application to the scattering term in the transport equation. Applying these expansions to the scattering and fission terms, the transport equation becomes

$$\begin{aligned}\frac{1}{v(E)}\frac{\partial}{\partial t}\psi(\boldsymbol{x},\boldsymbol{\Omega},E,t)+\boldsymbol{\Omega}\cdot\boldsymbol{\nabla}\psi(\boldsymbol{x},\boldsymbol{\Omega},E,t)+\Sigma_t(\boldsymbol{x},E)\,\psi(\boldsymbol{x},\boldsymbol{\Omega},E,t)\\ =\sum_{\ell=0}^{\infty}\frac{2\ell+1}{4\pi}\sum_{m=-\ell}^{\ell}Y_\ell^m(\boldsymbol{\Omega})\int_0^\infty\Sigma_{s;\ell}(\boldsymbol{x},E'\to E)\,\phi_\ell^m(\boldsymbol{x},E',t)\,dE'\\ +\frac{\chi(\boldsymbol{x},E)}{4\pi}\int_0^\infty\nu(\boldsymbol{x},E')\,\Sigma_f(\boldsymbol{x},E')\,\phi_0^0(\boldsymbol{x},E',t)\,dE'+q(\boldsymbol{x},\boldsymbol{\Omega},E,t),\\ \boldsymbol{x}\in V,\quad\boldsymbol{\Omega}\in 4\pi,\quad 0<E<\infty,\quad 0<t.\end{aligned}\tag{2.6}$$

The simplification of the scattering cross section is physically accurate, as the scattering angle for a neutron incident on a nucleus is a probability function for the scattering angle between the incoming and outgoing directions.

To numerically solve Eq. (2.6), the spatial, angular, time and energy domains are discretized individually. For the energy and angular discretizations, the multigroup method (Sec. 2.1.1) and the discrete ordinates method (Sec. 2.1.2), respectively, are common approximations [44] that are used here. The time discretization is either neglected in favor of the steady-state approximation (Sec. 2.1.3) or converted to the k-eigenvalue approximation (2.1.4). The spatial discretization is performed using the MLPG method (Secs. 2.2 to 2.9).

2.1 Angular and energy discretization of the transport equation

2.1.1 Multigroup transport equation

The multigroup approximation begins with an approximation of the neutron spectrum, $\Psi(\boldsymbol{x},E)$, over which each of the energy-dependent material properties is weighted over the energy range $E\in[E_g,E_{g-1})$ for chosen energy groups with index g to get multigroup cross sections, neutron velocities and fission sources,

$$\Sigma_{t;g}(\boldsymbol{x})\equiv\frac{\int_{E_g}^{E_{g-1}}\Sigma_t(\boldsymbol{x},E)\,\Psi(\boldsymbol{x},E)\,dE}{\int_{E_g}^{E_{g-1}}\Psi(\boldsymbol{x},E)\,dE},\tag{2.7a}$$

$$\Sigma_{s;\ell,g'\to g}(\boldsymbol{x})\equiv\frac{\int_{E_g}^{E_{g-1}}\int_{E_{g'}}^{E_{g'-1}}\Sigma_{s;\ell}(\boldsymbol{x},E'\to E)\,\Psi(\boldsymbol{x},E')\,dE'dE}{\int_{E_{g'}}^{E_{g'-1}}\Psi(\boldsymbol{x},E')\,dE'},\tag{2.7b}$$

$$\frac{1}{v_g(\boldsymbol{x})}\equiv\frac{\int_{E_g}^{E_{g-1}}\frac{1}{v(E)}\Psi(\boldsymbol{x},E)\,dE}{\int_{E_g}^{E_{g-1}}\Psi(\boldsymbol{x},E)\,dE},\tag{2.7c}$$

$$\nu\Sigma_{f;g}(\boldsymbol{x}) \equiv \frac{\int_{E_g}^{E_{g-1}} \nu(\boldsymbol{x},E)\,\Sigma_f(\boldsymbol{x},E)\,\Psi(\boldsymbol{x},E)\,dE}{\int_{E_g}^{E_{g-1}} \Psi(\boldsymbol{x},E)\,dE}. \tag{2.7d}$$

The multigroup fission spectrum, angular flux and source are integrated without neutron spectrum weighting,

$$\chi_g(\boldsymbol{x}) \equiv \int_{E_g}^{E_{g-1}} \chi(\boldsymbol{x},E)\,dE, \tag{2.7e}$$

$$\psi_g(\boldsymbol{x},\boldsymbol{\Omega},t) \equiv \int_{E_g}^{E_{g-1}} \psi(\boldsymbol{x},\boldsymbol{\Omega},E,t)\,dE, \tag{2.7f}$$

$$q_g(\boldsymbol{x},\boldsymbol{\Omega},t) \equiv \int_{E_g}^{E_{g-1}} q(\boldsymbol{x},\boldsymbol{\Omega},E,t)\,dE. \tag{2.7g}$$

Applying this energy integration to the continuous-energy angular flux expansion in Eq. (2.5a) results in the multigroup angular flux expansion

$$\psi_g(\boldsymbol{x},\boldsymbol{\Omega},t) = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y_\ell^m(\boldsymbol{\Omega})\,\phi_{\ell,g}^m(\boldsymbol{x},t)\,d\Omega, \tag{2.8a}$$

where the spherical harmonics moments are defined as

$$\phi_{\ell,g}^m(\boldsymbol{x},t) \equiv \int_{4\pi} Y_\ell^m(\boldsymbol{\Omega})\,\psi_g(\boldsymbol{x},\boldsymbol{\Omega},t)\,d\Omega, \tag{2.8b}$$

With these variables defined, the transport equation is converted to multigroup form by integrating the transport equation [Eq. (2.6)] over $E \in [E_g, E_{g-1}]$,

$$\begin{aligned}
&\frac{\partial}{\partial t}\int_{E_g}^{E_{g-1}} \frac{1}{v(E)}\psi(\boldsymbol{x},\boldsymbol{\Omega},E,t)\,dE + \boldsymbol{\Omega}\cdot\boldsymbol{\nabla}\int_{E_g}^{E_{g-1}} \psi(\boldsymbol{x},\boldsymbol{\Omega},E,t)\,dE + \int_{E_g}^{E_{g-1}} \Sigma_t(\boldsymbol{x},E)\,\psi(\boldsymbol{x},\boldsymbol{\Omega},E,t)\,dE \\
&\qquad = \sum_{\ell=0}^{\infty}\frac{2\ell+1}{4\pi}\sum_{m=-\ell}^{\ell} Y_\ell^m(\boldsymbol{\Omega})\int_{E_g}^{E_{g-1}}\int_0^{\infty} \Sigma_{s;\ell}(\boldsymbol{x},E'\to E)\,\phi_\ell^m(\boldsymbol{x},E',t)\,dE'dE \\
&\qquad + \int_{E_g}^{E_{g-1}} \frac{\chi(\boldsymbol{x},E)}{4\pi}\int_0^{\infty} \nu(\boldsymbol{x},E')\,\Sigma_f(\boldsymbol{x},E')\,\phi_0^0(\boldsymbol{x},E',t)\,dE'dE + \int_{E_g}^{E_{g-1}} q(\boldsymbol{x},\boldsymbol{\Omega},E,t)\,dE, \\
&\qquad\qquad \boldsymbol{x}\in V, \quad \boldsymbol{\Omega}\in 4\pi, \quad g=1,\ldots,G, \quad 0<t.
\end{aligned} \tag{2.9}$$

If weighted integrals of the physical data over $\psi(\boldsymbol{x},E)$ are approximately equal to weighted integrals over $\psi(\boldsymbol{x},\boldsymbol{\Omega},E,t)$ (e.g. for Σ_t,

$$\frac{\int_{E_g}^{E_{g-1}} \Sigma_t(\boldsymbol{x},E)\,\Psi(\boldsymbol{x},E)\,dE}{\int_{E_g}^{E_{g-1}} \Psi(\boldsymbol{x},E)\,dE} \approx \frac{\int_{E_g}^{E_{g-1}} \Sigma_t(\boldsymbol{x},E)\,\psi(\boldsymbol{x},\boldsymbol{\Omega},E,t)\,dE}{\int_{E_g}^{E_{g-1}} \psi(\boldsymbol{x},\boldsymbol{\Omega},E,t)\,dE} \tag{2.10}$$

should hold), the transport equation can be simplified to

$$\frac{1}{v_g(\boldsymbol{x})}\frac{\partial}{\partial t}\psi_g(\boldsymbol{x},\boldsymbol{\Omega},t)+\boldsymbol{\Omega}\cdot\boldsymbol{\nabla}\psi_g(\boldsymbol{x},\boldsymbol{\Omega},t)+\Sigma_{t;g}(\boldsymbol{x})\,\psi_g(\boldsymbol{x},\boldsymbol{\Omega},t)$$
$$=\sum_{\ell=0}^{\infty}\frac{2\ell+1}{4\pi}\sum_{m=-\ell}^{\ell}Y_\ell^m(\boldsymbol{\Omega})\sum_{g'=1}^{G}\Sigma_{s;\ell,g'\to g}(\boldsymbol{x})\,\phi_{\ell,g'}^m(\boldsymbol{x},t)$$
$$+\frac{\chi_g(\boldsymbol{x})}{4\pi}\sum_{g'=1}^{G}\nu\Sigma_{f;g'}(\boldsymbol{x})\,\phi_{0,g'}^0(\boldsymbol{x},t)+q_g(\boldsymbol{x},\boldsymbol{\Omega},t),$$
$$\boldsymbol{x}\in V,\quad \boldsymbol{\Omega}\in 4\pi,\quad g=1,\ldots,G,\quad 0<t. \tag{2.11a}$$

Integrating the initial and boundary conditions [Eqs. (2.1b) and (2.1c), respectively] over the energy group ranges produces the multigroup initial and boundary equations,

$$\psi_g(\boldsymbol{x},\boldsymbol{\Omega},t)=\psi_g^{inc}(\boldsymbol{x},\boldsymbol{\Omega},t)+\rho\psi_g(\boldsymbol{x},\boldsymbol{\Omega}_r,t),\qquad \boldsymbol{x}\in\partial V,\quad \boldsymbol{\Omega}\cdot\boldsymbol{n}<0,\quad g=1,\ldots,G,\quad 0<t, \tag{2.11b}$$
$$\psi_g(\boldsymbol{x},\boldsymbol{\Omega},t=0)=\psi_g^{init}(\boldsymbol{x},\boldsymbol{\Omega}),\qquad \boldsymbol{x}\in\partial V,\quad \boldsymbol{\Omega}\in 4\pi,\quad g=1,\ldots,G. \tag{2.11c}$$

2.1.2 Discrete ordinates transport equation

The discrete-ordinates (or S_N) approximation discretizes the angular variable of the transport equation by transporting the neutrons in discrete directions, which are used as ordinates for the integration of the angular flux to calculate its spherical harmonics moments. The multigroup transport equation [Eq. (2.11a)] is evaluated at the discrete directions $\boldsymbol{\Omega}_n$, which results in the multigroup S_N transport equation,

$$\frac{1}{v_g(\boldsymbol{x})}\frac{\partial}{\partial t}\psi_{n,g}(\boldsymbol{x},t)+\boldsymbol{\Omega}_n\cdot\boldsymbol{\nabla}\psi_{n,g}(\boldsymbol{x},t)+\Sigma_{t;g}(\boldsymbol{x})\,\psi_{n,g}(\boldsymbol{x},t)$$
$$=\sum_{\ell=0}^{L}\frac{2\ell+1}{4\pi}\sum_{m=-\ell}^{\ell}Y_\ell^m(\boldsymbol{\Omega}_n)\sum_{g'=1}^{G}\Sigma_{s;\ell,g'\to g}(\boldsymbol{x})\,\phi_{\ell,g'}^m(\boldsymbol{x},t)$$
$$+\frac{\chi_g(\boldsymbol{x})}{4\pi}\sum_{g'=1}^{G}\nu\Sigma_{f;g'}(\boldsymbol{x})\,\phi_{0,g'}^0(\boldsymbol{x},t)+q_{n,g}(\boldsymbol{x},t),$$
$$\boldsymbol{x}\in V,\quad n=1,\ldots N,\quad g=1,\ldots,G,\quad 0<t, \tag{2.12a}$$

with the boundary and initial conditions,

$$\psi_{n,g}(\boldsymbol{x},t)=\psi_{n,g}^{inc}(\boldsymbol{x},t)+\rho\psi_{n_r,g}(\boldsymbol{x},t),\qquad \boldsymbol{x}\in V,\quad \boldsymbol{\Omega}_n\cdot\boldsymbol{n}<0,\quad g=1,\ldots,G,\quad 0<t, \tag{2.12b}$$
$$\psi_{n,g}(\boldsymbol{x},t=0)=\psi_{n,g}^{init}(\boldsymbol{x}),\qquad \boldsymbol{x}\in V,\quad n=1,\ldots N,\quad g=1,\ldots,G. \tag{2.12c}$$

The discrete direction of reflection for the boundary equation is

$$\boldsymbol{\Omega}_{n_r} = \boldsymbol{\Omega}_n - 2\left(\boldsymbol{\Omega}\cdot\boldsymbol{n}\right)\boldsymbol{n}, \tag{2.13}$$

which is assumed to also belong to the set of discrete directions. The discrete angular flux and source are defined as

$$\psi_{n,g}\left(\boldsymbol{x},t\right) \equiv \psi_g\left(\boldsymbol{x},\boldsymbol{\Omega}_n,t\right), \tag{2.14a}$$

$$q_{n,g}\left(\boldsymbol{x},t\right) \equiv q_g\left(\boldsymbol{x},\boldsymbol{\Omega}_n,t\right). \tag{2.14b}$$

The angular flux expansion [Eq. (2.8a)] is integrated over energy and the angular integration in the definition of the spherical harmonics moments [Eq. (2.8b)] is converted to a discrete sum using the angular quadrature points $\boldsymbol{\Omega}_n$ and weights c_n,

$$\phi^m_{\ell,g}\left(\boldsymbol{x}\right) = \sum_{n=1}^{N} c_n Y^m_\ell\left(\boldsymbol{\Omega}_n\right)\psi_{n,g}\left(\boldsymbol{x}\right), \tag{2.15a}$$

$$\psi_{n,g}\left(\boldsymbol{x}\right) = \sum_{\ell=0}^{L}\frac{2\ell+1}{4\pi}\sum_{m=-\ell}^{\ell} Y^m_\ell\left(\boldsymbol{\Omega}_n\right)\phi^m_{\ell,g}\left(\boldsymbol{x}\right). \tag{2.15b}$$

These expansions allow conversion between the discrete form of the angular flux, which is used for the streaming operator, and the moment form, which is used for the scattering and fission terms. The spherical harmonics expansion is truncated to a finite degree L, with the requirement that the quadrature set should accurately integrate the spherical harmonics up to the specified degree.

2.1.3 Steady-state transport equation

The steady-state assumption simplifies the transport equation by removing the time dependence, which also removes the time derivative,

$$\begin{aligned}\boldsymbol{\Omega}_n \cdot \boldsymbol{\nabla}\psi_{n,g}(\boldsymbol{x}) + \Sigma_{t;g}(\boldsymbol{x})\,\psi_{n,g}(\boldsymbol{x}) \\ = \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y_\ell^m(\boldsymbol{\Omega}_n) \sum_{g'=1}^{G} \Sigma_{s;\ell,g'\to g}(\boldsymbol{x})\,\phi_{\ell,g'}^m(\boldsymbol{x}) \\ + \frac{\chi_g(\boldsymbol{x})}{4\pi} \sum_{g'=1}^{G} \nu\Sigma_{f;g'}(\boldsymbol{x})\,\phi_{0,g'}^0(\boldsymbol{x}) + q_{n,g}(\boldsymbol{x}), \\ \boldsymbol{x}\in V, \quad n=1,\dots N, \quad g=1,\dots,G. \end{aligned} \tag{2.16a}$$

The boundary condition similarly neglects time dependence,

$$\psi_{n,g}(\boldsymbol{x}) = \psi_{n,g}^{inc}(\boldsymbol{x}) + \rho\psi_{n_r,g}(\boldsymbol{x}), \qquad \boldsymbol{x}\in V, \quad \boldsymbol{\Omega}_n\cdot\boldsymbol{n}<0, \quad g=1,\dots,G, \tag{2.16b}$$

and the initial condition is no longer needed. This form of the transport equation is applicable to subcritical problems with a given inhomogeneous source of inside the problem or on its boundary.

2.1.4 k-eigenvalue transport equation

The k-eigenvalue problem simplifies the time-dependent transport problem with multiplication to an eigenvalue problem for multiplication factor of the problem. The source term of the transport equation $q_{n,g}$ [Eq. (2.16a)] is removed and the average number of neutrons in a fission event ν is divided by the multiplication factor (or k-eigenvalue), $\nu \to \nu/k$,

$$\begin{aligned}\boldsymbol{\Omega}_n \cdot \boldsymbol{\nabla}\psi_{n,g}(\boldsymbol{x}) + \Sigma_{t;g}(\boldsymbol{x})\,\psi_{n,g}(\boldsymbol{x}) \\ = \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y_\ell^m(\boldsymbol{\Omega}_n) \sum_{g'=1}^{G} \Sigma_{s;\ell,g'\to g}(\boldsymbol{x})\,\phi_{\ell,g'}^m(\boldsymbol{x}) \\ + \frac{\chi_g(\boldsymbol{x})}{4\pi k} \sum_{g'=1}^{G} \nu\Sigma_{f;g'}(\boldsymbol{x})\,\phi_{0,g'}^0(\boldsymbol{x}), \\ \boldsymbol{x}\in V, \quad n=1,\dots N, \quad g=1,\dots,G. \end{aligned} \tag{2.17a}$$

The values $k<1$, $k=1$ and $k>1$ correspond to a subcritical, critical and supercritical system, respectively. The k-eigenvalue boundary equation is homogeneous and therefore excludes the incoming flux on the

boundary,

$$\psi_{n,g}(\boldsymbol{x}) = \rho\psi_{n_r,g}(\boldsymbol{x}), \qquad \boldsymbol{x} \in V, \quad \boldsymbol{\Omega}_n \cdot \boldsymbol{n} < 0, \quad g = 1, \ldots, G. \tag{2.17b}$$

This form of the equation is applicable to systems in which fission is a primary source of neutrons.

2.2 Operator form of the transport equation

To simplify the following sections, the transport equation is written in operator notation, in which the operators are implicit matrices that modify a vector of values. An example of a simple operator is the identity operator, $\mathcal{I}$, which when applied to a vector x returns that same vector, i.e. $\mathcal{I}x = x$. The neutron transport equation can be written in operator notation [45] as

$$\mathcal{L}^s\psi = \mathcal{M}\left(\mathcal{S}^s + \mathcal{F}^s\right)\phi + q^s, \tag{2.18a}$$

where $\mathcal{L}$ is the streaming and collision operator, $\mathcal{M}$ is the moment-to-discrete operator, $\mathcal{S}$ and $\mathcal{F}$ are the scattering and fission operators, ψ and ϕ are the angular flux and its moments, respectively, and q is the inhomogeneous source term. The s superscript denotes that some of the operators are specific to the strong-form equations. The k-eigenvalue equation can be similarly written [46] as

$$\mathcal{L}^s\psi = \mathcal{M}\left(\mathcal{S}^s + \frac{1}{k}\mathcal{F}^s\right)\phi. \tag{2.18b}$$

Manipulated forms of this equation also require the discrete-to-moment operator $\mathcal{D}$. For the multigroup, discrete-ordinates transport equation [Eq. (2.16a)], these operators are defined as

$$\left(\mathcal{L}^s\psi\right)_{n,g} = \boldsymbol{\Omega}_n \cdot \boldsymbol{\nabla}\psi_{n,g} + \Sigma_{t;g}\psi_{n,g}, \tag{2.19a}$$

$$\left(\mathcal{M}\phi\right)_{n,g} = \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y_\ell^m\left(\boldsymbol{\Omega}_n\right)\phi_{\ell,g}^m, \tag{2.19b}$$

$$\left(\mathcal{D}\psi\right)_{\ell,g}^m = \sum_{n=1}^{N} Y_\ell^m\left(\boldsymbol{\Omega}_n\right) c_n\psi_{n,g}, \tag{2.19c}$$

$$\left(\mathcal{S}^s\phi\right)_{\ell,g}^m = \sum_{g'=1}^{G} \Sigma_{s;\ell,g'\to g}\phi_{\ell,g'}^m, \tag{2.19d}$$

$$\left(\mathcal{F}^s\phi\right)_{\ell,g}^m = \delta_{\ell,0}\chi_g \sum_{g'=1}^{G} \nu_{g'}\Sigma_{f;g'}\phi_{0,g'}^0, \tag{2.19e}$$

with the internal source vector

$$\left(q^s\right)_{n,g} = q_{n,g}, \tag{2.19f}$$

and the Kronecker delta function

$$\delta_{a,b} = \begin{cases} 1, & a = b, \\ 0, & \text{otherwise.} \end{cases} \tag{2.20}$$

The boundary equation is included in the operator equations in Sec. 2.3. For further details on various operator forms of the transport equation and descriptions of iteration methods to solve the operator equations, see Secs. 2.6 and 3.4.

2.3 Weak form of the transport equation

The weak form of the transport equation is derived by multiplying the strong-form equations [Eqs. (2.19)] by a series of weight functions w_j and integrating over the volume of support for these weight functions, V_j,

$$(\mathcal{L}^w\psi)^*_{j,n,g} = \int_{V_j} w_j \mathbf{\Omega}_n \cdot \boldsymbol{\nabla}\psi_{n,g} dV + \int_{V_j} w_j \Sigma_{t;g}\psi_{n,g} dV, \tag{2.21a}$$

$$(\mathcal{S}^w\phi)^m_{j,\ell,g} = \sum_{g'=1}^{G} \int_{V_j} w_j \Sigma_{s;\ell,g'\to g}\phi^m_{\ell,g'} dV, \tag{2.21b}$$

$$(\mathcal{F}^w\phi)^m_{j,\ell,g} = \delta_{\ell,0} \int_{V_j} w_j \chi_g \sum_{g'=1}^{G} \nu_{g'}\Sigma_{f;g'}\phi^0_{0,g'} dV. \tag{2.21c}$$

The streaming term is integrated by parts to produce surface integrals for the outgoing flux and the incoming flux,

$$(\mathcal{L}^w\psi)_{j,n,g} = \int_{\mathbf{\Omega}_n\cdot\boldsymbol{n}>0} (\mathbf{\Omega}_n \cdot \boldsymbol{n})\, \psi_{n,g} w_j dS - \int_{V_j} \psi_{n,g}\mathbf{\Omega}_n \cdot \boldsymbol{\nabla} w_j dV + \int_{V_j} w_j \Sigma_{t;g}\psi_{n,g} dV, \tag{2.21d}$$

$$(q)_{j,n,g} = \int_{\mathbf{\Omega}_n\cdot\boldsymbol{n}<0} |\mathbf{\Omega}_n \cdot \boldsymbol{n}|\, \psi^{inc}_{n,g} w_j dS + \int_{V_j} w_j q_{n,g} dV. \tag{2.21e}$$

Reflection is separated from the known incoming angular flux to get a reflective boundary operator $\mathcal{R}$ that is a applied to the unknown angular flux on the boundary,

$$\left(\mathcal{R}^w\psi^b\right)_{j,n,g} = \rho \int_{\mathbf{\Omega}_n\cdot\boldsymbol{n}<0} |\mathbf{\Omega}_n \cdot \boldsymbol{n}|\, \psi_{n_r,g} w_j dS. \tag{2.21f}$$

The superscript w denotes the operators specific to the weak form of the transport equation. The operator forms of the weak steady-state and k-eigenvalue transport equations that include the reflection term are

$$\mathcal{L}^w\psi = \boldsymbol{\mathcal{M}}^w \left(\boldsymbol{\mathcal{S}}^w + \boldsymbol{\mathcal{F}}^w\right)\boldsymbol{\phi} + q, \tag{2.22a}$$

$$\mathcal{L}^w\psi = \boldsymbol{\mathcal{M}}^w \left(\boldsymbol{\mathcal{S}}^w + \frac{1}{k}\boldsymbol{\mathcal{F}}^w\right)\boldsymbol{\phi}, \tag{2.22b}$$

where the augmented solution is

$$\boldsymbol{\phi} = \begin{bmatrix} \phi \\ \\ \psi^b \end{bmatrix} \tag{2.23}$$

and the augmented operators are

$$\boldsymbol{\mathcal{M}}^w = \begin{bmatrix} \mathcal{M} & \mathcal{R}^w \end{bmatrix}, \tag{2.24a}$$

$$\boldsymbol{\mathcal{S}}^w = \begin{bmatrix} \mathcal{S}^w & 0 \\ \\ 0 & \mathcal{I}^b \end{bmatrix}, \tag{2.24b}$$

$$\boldsymbol{\mathcal{F}}^w = \begin{bmatrix} \mathcal{F}^w & 0 \\ \\ 0 & 0 \end{bmatrix}. \tag{2.24c}$$

ψ^b is the values of the angular flux on the boundary and $\mathcal{I}^b$ is the identity operator for this quantity, $\mathcal{I}^b\psi^b = \psi^b$.

Equations (2.21) represent an independent set of linear equations that can be spatially discretized by inserting a basis function expansion of the angular flux and its moments. For the MLPG method, such a discretization produces oscillations caused by advection in the streaming operator [Eq. (2.21a)]. Stabilization added to the streaming operator helps prevent these non-physical oscillations.

2.4 SUPG stabilization of the transport equation

While advection in a continuous Petrov-Galerkin method creates oscillations, diffusion dampens them. This property motivates the SUPG method, which adds numerical diffusion in the streamline direction by modifying the advection term, using this modification to change to the weight functions appropriately and applying the altered weight functions to the entire transport equation [15].

To derive an appropriate diffusion term to add to the advection operator, a diffusion operator is applied to the angular flux,

$$-\boldsymbol{\nabla}\cdot(\boldsymbol{D}\cdot\boldsymbol{\nabla})\,\psi_{n,g} = 0 \tag{2.25}$$

with the diffusion tensor $\boldsymbol{D} = \boldsymbol{\Omega}_n\boldsymbol{\Omega}_n$. This is converted to weak form by multiplying by the weight function

w_j and integrating by parts,

$$-\int_{S_j}\left[\boldsymbol{n}\cdot\left(\boldsymbol{D}\cdot\boldsymbol{\nabla}\right)\psi_{n,g}\right]w_j dS + \int_{V_j}\left(\boldsymbol{\Omega}_n\cdot\boldsymbol{\nabla}\psi_{n,g}\right)\left(\boldsymbol{\Omega}_n\cdot\boldsymbol{\nabla}w_j\right)dV = 0. \tag{2.26}$$

The process of adding the volume integral in this diffusion equation [Eq. (2.26)] to the streaming operator in the weak transport equation [Eq. (2.21a)],

$$\int_{V_j}\left(\boldsymbol{\Omega}_n\cdot\boldsymbol{\nabla}\psi_{n,g}\right)\left(w_j+\boldsymbol{\Omega}_n\cdot\boldsymbol{\nabla}w_j\right)dV, \tag{2.27}$$

can be replicated by augmenting the weight functions in Eqs. (2.21a) to (2.21c) by the streaming operator,

$$\tilde{w}_{j,n} = w_j + \tau\boldsymbol{\Omega}_n\cdot\boldsymbol{\nabla}w_j. \tag{2.28}$$

Here τ is a proportionality constant with units of length that is discussed in more detail in Sec. 3.2.2. The augmented weight functions in Eq. (2.28) are used everywhere, not only for the streaming term. Because of this, the equations retain the properties of a Petrov-Galerkin method, specifically that the exact solution to the transport equation fulfills the discretized equations.

The full SUPG transport equations are derived by replacing w_j with $\tilde{w}_{j,n}$ in Eq. (2.21) and integrating the streaming operator by parts for only the w_j term from the augmented weight function [Eq. (2.28)], which leaves the surface integrals unchanged. The equations become

$$\left(\mathcal{L}^u\psi\right)_{j,n,g} = \int_{\boldsymbol{\Omega}_n\cdot\boldsymbol{n}>0}\left(\boldsymbol{\Omega}_n\cdot\boldsymbol{n}\right)\psi_{n,g}w_j dS + \int_{V_j}\left[-\psi_{n,g}+\tau\boldsymbol{\Omega}_n\cdot\boldsymbol{\nabla}\psi_{n,g}\right]\left(\boldsymbol{\Omega}_n\cdot\boldsymbol{\nabla}w_j\right)dV + \int_{V_j}\tilde{w}_{j,n}\Sigma_{t;g}\psi_{n,g}dV, \tag{2.29a}$$

$$\left(\mathcal{M}\mathcal{S}^u\phi\right)_{j,n,g} = \sum_{\ell=0}^{L}\frac{2\ell+1}{4\pi}\sum_{m=-\ell}^{\ell}Y_\ell^m\left(\boldsymbol{\Omega}_n\right)\sum_{g'=1}^{G}\int_{V_j}\tilde{w}_{j,n}\Sigma_{s;\ell,g'\rightarrow g}\phi_{\ell,g'}^m dV, \tag{2.29b}$$

$$\left(\mathcal{M}\mathcal{F}^u\phi\right)_{j,n,g} = \frac{1}{4\pi}\int_{V_j}\tilde{w}_{j,n}\chi_g\sum_{g'=1}^{G}\nu_{g'}\Sigma_{f;g'}\phi_{0,g'}^0 dV, \tag{2.29c}$$

$$\left(q^u\right)_{j,n,g} = \int_{\boldsymbol{\Omega}_n\cdot\boldsymbol{n}<0}\left|\boldsymbol{\Omega}_n\cdot\boldsymbol{n}\right|\psi_{n,g}^{ext}w_j dS + \int_{V_j}\tilde{w}_{j,n}q_{n,g}dV, \tag{2.29d}$$

where the u superscript denotes operators specific to the SUPG equations. The resulting addition to the streaming term in Eq. (2.29a) applies numerical diffusion preferentially to neutrons whose direction of flight $\boldsymbol{\Omega}_n$ is codirectional with the gradient of the angular flux $\boldsymbol{\nabla}\psi_{n,g}$. Due to the directional dependence of the

weight functions, the scattering and fission operators are combined with the moment-to-discrete operator. The decoupling of these operators is discussed in Sec. 3.2.2. The reflection operator does not change from the unstabilized form of the equations [Eq. (2.21f)].

2.5 Petrov-Galerkin form of the transport equation

To complete the discretization of the SUPG-stabilized transport equations [Eqs. (2.29)], free variables are added to match the number of constraints. The Petrov-Galerkin method uses functional expansions for the angular flux and its moments as the unknown variables for the weak form of the transport equation,

$$\psi_{n,g}(\boldsymbol{x}) = \sum_{i=1}^{J} \alpha_{i,n,g} b_i(\boldsymbol{x}), \tag{2.30a}$$

$$\phi_{\ell,g}^{m}(\boldsymbol{x}) = \sum_{i=1}^{J} \beta_{i,\ell,g}^{m} b_i(\boldsymbol{x}), \tag{2.30b}$$

with the basis functions $b_i(\boldsymbol{x})$, the angular flux expansion coefficients $\alpha_{i,n,g}$ and the moment expansion coefficients $\beta_{i,\ell,g}^{m}$. Using the definition of the spherical harmonics moments in Eq. (2.15a), $\beta_{i,\ell,g}^{m}$ can be defined in terms of $\alpha_{i,n,g}$,

$$\beta_{i,\ell,g}^{m} \equiv \sum_{n=1}^{N} w_n Y_{\ell}^{m}(\boldsymbol{\Omega}_n) \alpha_{i,n,g}. \tag{2.31}$$

This allows direct conversion between the two types of coefficients. Substituting the basis expansions into Eqs. (2.29), the operators of the transport equation become

$$\begin{aligned}(\mathcal{L}^p\alpha)_{j,n,g} = \sum_{i=1}^{J} \Bigg[&\int_{\boldsymbol{\Omega}_n\cdot\boldsymbol{n}>0} (\boldsymbol{\Omega}_n\cdot\boldsymbol{n}) b_i w_j dS + \int_{V_{i,j}} (-b_i + \tau\boldsymbol{\Omega}_n\cdot\boldsymbol{\nabla} b_i)(\boldsymbol{\Omega}_n\cdot\boldsymbol{\nabla} w_j) dV \\ &+ \int_{V_{i,j}} \tilde{w}_{j,n}\Sigma_{t;g} b_i dV \Bigg] \alpha_{i,n,g},\end{aligned} \tag{2.32a}$$

$$(\mathcal{M}\mathcal{S}^p\beta)_{j,n,g} = \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y_{\ell}^{m}(\boldsymbol{\Omega}_n) \sum_{i=1}^{J}\sum_{g'=1}^{G} \left(\int_{V_{i,j}} \tilde{w}_{j,n}\Sigma_{s;\ell,g'\to g} b_i dV \right) \beta_{i,\ell,g'}^{m}, \tag{2.32b}$$

$$(\mathcal{M}\mathcal{F}^p\beta)_{j,n,g} = \frac{1}{4\pi} \sum_{i=1}^{J} \left(\int_{V_{i,j}} \tilde{w}_{j,n}\chi_g \sum_{g'=1}^{G} \nu_{g'}\Sigma_{f;g'} b_i dV \right) \beta_{i,\ell,g'}^{m}, \tag{2.32c}$$

$$\left(\mathcal{R}^p\alpha^b\right)_{j,n,g} = \rho \sum_{i=1}^{J} \left(\int_{\boldsymbol{\Omega}_n\cdot\boldsymbol{n}<0} |\boldsymbol{\Omega}_n\cdot\boldsymbol{n}| b_i w_j dS \right) \alpha_{i,n_r,g}^{b}, \tag{2.32d}$$

$$(q^p)_{j,n,g} = \int_{\boldsymbol{\Omega}_n\cdot\boldsymbol{n}<0} |\boldsymbol{\Omega}_n\cdot\boldsymbol{n}| \psi_{n,g}^{ext} w_j dS + \int_{V_j} \tilde{w}_{j,n} q_{n,g} dV, \tag{2.32e}$$

where $V_{i,j}$ is the intersection of the support region for the basis function b_i and weight function w_j (see Sec. 4.1 for details). The p superscript denotes the operators specific to the Petrov-Galerkin form of the

equations.

The operators for the Petrov-Galerkin equations are applied directly to the expansion coefficients instead of to physical flux values, which changes Eqs. (2.22) to

$$\mathcal{L}^p \alpha = \boldsymbol{\mathcal{M}}^p \left(\boldsymbol{\mathcal{S}}^p + \boldsymbol{\mathcal{F}}^p \right) \boldsymbol{\beta} + q^p, \tag{2.33a}$$

$$\mathcal{L}^p \alpha = \boldsymbol{\mathcal{M}}^p \left(\boldsymbol{\mathcal{S}}^p + \frac{1}{k} \boldsymbol{\mathcal{F}}^p \right) \boldsymbol{\beta}, \tag{2.33b}$$

with the same augments an in Eqs. (2.24) and the augmented coefficients

$$\boldsymbol{\beta} = \begin{bmatrix} \beta \\ \\ \alpha^b \end{bmatrix}. \tag{2.34}$$

Equations (2.32) and (2.33) represent a fully-determined linear system of equations, meaning a standard linear or eigenvalue solver could solve these directly. However, iterative schemes such as those in Sec. 2.6 can significantly reduce the cost of solving the transport equation.

2.6 Iterative schemes for the neutron transport equation

To reduce the number of unknowns, the coefficients for the moments of the angular flux, β, are chosen as the solution variable for the transport equation. This assumes that the directional dependence of the exact angular flux can be accurately represented in terms of a spherical harmonics expansion, which is the case for a solution with limited anisotropy. The reflective boundary adds additional solution variables for the angular flux expansion coefficients at the boundary, α^b.

A few additional operators are needed to remove the angular flux coefficients α from the the operator form of the transport equation. Using the operator $\mathcal{A}\alpha = \alpha^b$ that removes non-boundary terms from α, augmented forms of the discrete-to-moment operator are defined as

$$\boldsymbol{\mathcal{D}} = \begin{bmatrix} \mathcal{D} \\ \\ \mathcal{A} \end{bmatrix}, \tag{2.35a}$$

$$\boldsymbol{\mathcal{D}}_q = \begin{bmatrix} \mathcal{D} \\ \\ 0 \end{bmatrix}. \tag{2.35b}$$

To isolate the α values, the inverse of the streaming operator, $(\mathcal{L}^p)^{-1}$, is applied to Eqs. (2.33),

$$\alpha = (\mathcal{L}^p)^{-1} \boldsymbol{\mathcal{M}}^p (\boldsymbol{\mathcal{S}}^p + \boldsymbol{\mathcal{F}}^p) \boldsymbol{\beta} + (\mathcal{L}^p)^{-1} q^p, \tag{2.36}$$

$$\alpha = (\mathcal{L}^p)^{-1} \boldsymbol{\mathcal{M}}^p \left(\boldsymbol{\mathcal{S}}^p + \frac{1}{k} \boldsymbol{\mathcal{F}}^p \right) \boldsymbol{\beta}. \tag{2.37}$$

The $\mathcal{L}^p$ operators for each group and direction are independent and the inversion can be done independently (see Sec. 3.3). Applying the discrete-to-moment operators to both sides of these equations and rearranging the terms results in a form of each of the equations that is suitable for Krylov iterative methods,

$$\left[\boldsymbol{\mathcal{I}} - \boldsymbol{\mathcal{D}} (\mathcal{L}^p)^{-1} \boldsymbol{\mathcal{M}}^p (\boldsymbol{\mathcal{S}}^p + \boldsymbol{\mathcal{F}}^p) \right] \boldsymbol{\beta} = \boldsymbol{\mathcal{D}}_q (\mathcal{L}^p)^{-1} q^p, \tag{2.38a}$$

$$\boldsymbol{\mathcal{D}} (\mathcal{L}^p)^{-1} \boldsymbol{\mathcal{M}}^p \boldsymbol{\mathcal{F}}^p \boldsymbol{\beta} = k \left(\boldsymbol{\mathcal{I}} - \boldsymbol{\mathcal{D}} (\mathcal{L}^p)^{-1} \boldsymbol{\mathcal{M}}^p \boldsymbol{\mathcal{S}}^p \right) \boldsymbol{\beta}, \tag{2.38b}$$

with the identity operator $\boldsymbol{\mathcal{I}}\boldsymbol{\beta} = \boldsymbol{\beta}$. The α^b boundary terms in $\boldsymbol{\beta}$ are only used to calculate the reflective boundary angular flux. Calculating the value of $(\mathcal{L}^p)^{-1} q^p$ (which is physically the first-flight source) includes iteration in problems with reflection, as q^p is the uncollided flux which may reflect several times through the problem before interacting with the materials. The boundary terms are zeroed out by the operator $\boldsymbol{\mathcal{D}}_q$ as the reflection contribution from the source term q^p has already been included by this iteration process. Solution methods using these operator forms of the transport equation are discussed further in Sec. 3.4.

2.7 Particle balance

For the steady-state neutron transport equation [Eq. (2.18a)], the number of neutrons entering any chosen control volume in each energy group and direction through streaming, fission, scattering and any external sources is exactly balanced by the number of neutrons leaving through streaming or collision. The neutron conservation equation is a weaker statement that neutrons are conserved in a specific control volume. For finite volume and finite element discretizations, the control volumes are the elements. For the MLPG method, the control volume is the entire problem domain. The global statement of conservation can be derived by integrating the steady state equations [Eqs. (2.18a)] over the problem domain V,

$$\left(\mathcal{L}^c \psi\right)_{j,n,g} = \int_{\boldsymbol{\Omega}_n \cdot \boldsymbol{n} > 0} (\boldsymbol{\Omega}_n \cdot \boldsymbol{n}) \, \psi_{n,g} dS + \int_V \Sigma_{t;g} \psi_{n,g} dV, \tag{2.39a}$$

$$\left(\mathcal{S}^c \phi\right)^m_{j,\ell,g} = \sum_{g'=1}^{G} \int_V \Sigma_{s;\ell,g' \to g} \phi^m_{\ell,g'} dV, \tag{2.39b}$$

$$\left(\mathcal{F}^c \phi\right)^m_{j,\ell,g} = \delta_{\ell,0} \int_V \chi_g \sum_{g'=1}^{G} \nu_{g'} \Sigma_{f;g'} \phi^0_{0,g'} dV. \tag{2.39c}$$

$$\left(q^c\right)_{j,n,g} = \int_{\boldsymbol{\Omega}_n \cdot \boldsymbol{n} < 0} |\boldsymbol{\Omega}_n \cdot \boldsymbol{n}| \, \psi^{inc}_{n,g} dS + \int_V q_{n,g} dV. \tag{2.39d}$$

The c superscript denotes the conservation operators. As in the weak form derivation, the moment-to-discrete and discrete-to-moment operators do not change.

A sufficient condition to show that the SUPG-stabilized weak-form equations derived in Sec. 2.4 satisfy the conservation equations [Eqs. (2.39)] is that

$$\sum_{j=1}^{J} w_j = 1 \tag{2.40}$$

for the weight functions chosen to create the augmented weight functions in Eq. (2.28). If this holds, it also follows that

$$\sum_{j=1}^{J} \tilde{w}_{j,n} = \sum_{j=1}^{J} w_j = 1 \tag{2.41}$$

for every n as the sum annihilates the derivative. With this property, summation of the SUPG-stabilized transport equation [Eqs. (2.29)] over all weight function indices j directly results in the conservation equations [Eqs. (2.39)]. This indicates that the weighted SUPG equations preserve global particle balance. This condition does not hold for arbitrary weight functions or many directionally-dependent stabilization techniques, such as skewing or offsetting weight functions, for which the sum in Eq. (2.41) may be directionally dependent.

The Petrov-Galerkin equations from Sec. 2.5 can likewise be shown to reduce to the conservation equation. Assuming that the basis function expansion in Eq. (2.30a) can accurately model the angular flux solution and that the weight functions sum to unity [Eq. (2.40)], the sum of the MLPG equations [Eqs. (2.32)] over all weight functions j is

$$\left(\mathcal{L}^p \alpha\right)_{n,g} = \sum_{i=1}^{J} \left[\int_{\boldsymbol{\Omega}_n \cdot \boldsymbol{n} > 0} \left(\boldsymbol{\Omega}_n \cdot \boldsymbol{n}\right) b_i dS + \int_{V_i} \Sigma_{t;g} b_i dV \right] \alpha_{i,n,g}, \tag{2.42a}$$

$$\left(\mathcal{M}\mathcal{S}^p \beta\right)_{n,g} = \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y_\ell^m\left(\boldsymbol{\Omega}_n\right) \sum_{i=1}^{J} \sum_{g'=1}^{G} \left(\int_{V_i} \Sigma_{s;\ell,g' \to g} b_i dV \right) \beta^m_{i,\ell,g'}, \tag{2.42b}$$

$$\left(\mathcal{M}\mathcal{F}^p \beta\right)_{n,g} = \frac{1}{4\pi} \left(\sum_{i=1}^{J} \int_{V_i} \chi_g \sum_{g'=1}^{G} \nu_{g'} \Sigma_{f;g'} b_i dV \right) \beta^m_{i,\ell,g'}, \tag{2.42c}$$

$$\left(\mathcal{R}^p \alpha^b\right)_{n,g} = \rho \sum_{i=1}^{J} \left(\int_{\boldsymbol{\Omega}_n \cdot \boldsymbol{n} < 0} |\boldsymbol{\Omega}_n \cdot \boldsymbol{n}| \, b_i dS \right) \alpha^b_{i,n_r,g}. \tag{2.42d}$$

$$\left(q^p\right)_{n,g} = \int_{\boldsymbol{\Omega}_n \cdot \boldsymbol{n} < 0} |\boldsymbol{\Omega}_n \cdot \boldsymbol{n}| \, \psi^{ext}_{n,g} dS + \int_V q_{n,g} dV, \tag{2.42e}$$

where V_i is the support region of the basis function b_i. This is the conservation equation for the angular flux expansion.

Methods that preserve particle balance are much more likely to to converge to the correct solution. In Sec. 2.8, the balance equations are also used in deriving an approximation to the material properties of the transport equation. The restriction placed on the weight functions in Eq. (2.40) guides the selection of appropriate weight functions in Sec. 2.10.

2.8 Integral and cross section discretization

To simplify the transport equations further, the integrals of the basis and weight functions and the cross sections can be explicitly defined. The required integrals from Eqs. (2.32) that are independent of the material properties include

$$\bar{i}^{S}_{bw;i,j,u} \equiv \int_{S_{i,j,u}} b_i w_j dS, \tag{2.43a}$$

$$\bar{i}^{V}_{bw;i,j} \equiv \int_{V_{i,j}} b_i w_j dV, \tag{2.43b}$$

$$\bar{\boldsymbol{i}}^{V}_{b\nabla w;i,j} \equiv \int_{V_{i,j}} b_i \boldsymbol{\nabla} w_j dV, \tag{2.43c}$$

$$\overline{\boldsymbol{I}}_{\nabla b\nabla w;i,j} \equiv \int_{V_{i,j}} \boldsymbol{\nabla} b_i \boldsymbol{\nabla} w_j dV \tag{2.43d}$$

where $S_{i,j,u}$ is the intersection of the surface with index u and the support regions of the weight and basis functions w_j and b_i. The external source can be simplified by assuming a spherical harmonic expansion for the internal source moments $Q^m_{\ell,g}$,

$$q_{n,g} = \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y^m_\ell \left(\boldsymbol{\Omega}_n\right) Q^m_{\ell,g}, \tag{2.44}$$

and separating the two terms of the SUPG weight function into two integrals using Eq. (2.28),

$$\int_{V_j} \tilde{w}_{j,n} Q^m_{\ell,g} dV = \overline{Q}^{m,0}_{j,\ell,g} + \tau \boldsymbol{\Omega}_n \cdot \overline{\boldsymbol{Q}}^{m,1}_{j,\ell,g}, \tag{2.45}$$

where the source integrals are defined as

$$\overline{Q}^{m,0}_{j,\ell,g} \equiv \int_{V_j} w_j Q^m_{\ell,g} dV, \tag{2.46a}$$

$$\overline{\boldsymbol{Q}}^{m,1}_{j,\ell,g} \equiv \int_{V_j} \boldsymbol{\nabla} w_j Q^m_{\ell,g} dV. \tag{2.46b}$$

The integral over the boundary source can be defined similarly to the other boundary surface integrals as

$$\overline{\psi}^{ext}_{j,n,g,u} \equiv \int_{S_{j,s}} \psi^{ext}_{n,g} w_j dS,$$

where $S_{j,u}$ is the intersection surface of u and the support region for weight function w_j.

2.8.1 Full cross section method

In a realistic simulation, the cross sections inside some of the integrals are spatially dependent because the weight functions are not constrained to regions of homogeneous material. Due to the SUPG weighting, the integrals involving the cross sections depend on the basis function i, the weight function j, the group g and the direction n. The directional dependence is removed in the same way as for the nonhomogeneous source in Eqs. (2.46). The cross sections are split into two separate sets of integrals, one for the weight function and one for its gradient,

$$\int_{V_{i,j}} \tilde{w}_{j,n} \Sigma_{t;g} b_i dV = \overline{\Sigma}^0_{t;i,j,g} + \tau \boldsymbol{\Omega}_n \cdot \overline{\boldsymbol{\Sigma}}^1_{t;i,j,g}, \tag{2.47}$$

with the explicit cross section integrals

$$\overline{\Sigma}^0_{t;i,j,g} \equiv \int_{V_{i,j}} w_j \Sigma_{t;g} b_i dV, \tag{2.48a}$$

$$\overline{\boldsymbol{\Sigma}}^1_{t;i,j,g} \equiv \int_{V_{i,j}} \boldsymbol{\nabla} w_j \Sigma_{t;g} b_i dV. \tag{2.48b}$$

The fission cross section $\Sigma_{f;g}$, fission spectrum χ_g and the average number of neutrons produced in a fission ν_g are combined into group-to-group cross sections,

$$\int_{V_{i,j}} \tilde{w}_{j,n} \chi_g \sum_{g'=1}^{G} \nu_{g'} \Sigma_{f;g'} b_i dV = \sum_{g'=1}^{G} \left(\overline{\Sigma}^0_{f;i,j,g'\to g} + \tau \boldsymbol{\Omega}_n \cdot \overline{\boldsymbol{\Sigma}}^1_{f;i,j,g'\to g} \right), \tag{2.49}$$

with the integrals

$$\overline{\Sigma}^0_{f;i,j,g'\to g} \equiv \int_{V_{i,j}} w_j \chi_g \nu_{g'} \Sigma_{f;g'} b_i dV, \tag{2.50a}$$

$$\overline{\boldsymbol{\Sigma}}^1_{f;i,j,g'\to g} \equiv \int_{V_{i,j}} \boldsymbol{\nabla} w_j \chi_g \nu_{g'} \Sigma_{f;g'} b_i dV. \tag{2.50b}$$

The scattering cross section is similarly discretized,

$$\overline{\Sigma}^0_{s;i,j,\ell,g'\to g} \equiv \int_{V_{i,j}} w_j \Sigma_{s;\ell,g'\to g} b_i dV, \tag{2.51a}$$

$$\overline{\boldsymbol{\Sigma}}^1_{s;i,j,\ell,g'\to g} \equiv \int_{V_{i,j}} \boldsymbol{\nabla} w_j \Sigma_{s;\ell,g'\to g} b_i dV. \tag{2.51b}$$

This is referred to in this dissertation as the full cross section discretization, for which no approximation of the cross section integrals is made.

The transport equation operators with these known integrals become

$$\left(\mathcal{L}^p\alpha\right)_{j,n,g} = \sum_{i=1}^{J}\Bigg[\sum_{u:\boldsymbol{\Omega}_n\cdot\boldsymbol{n}_k>0}\left(\boldsymbol{\Omega}_n\cdot\boldsymbol{n}_u\right)\bar{\bar{i}}^S_{bw;i,j,u} - \boldsymbol{\Omega}_n\cdot\bar{\bar{\boldsymbol{i}}}^V_{b\nabla w;i,j} + \tau\boldsymbol{\Omega}_n\cdot\left(\boldsymbol{\Omega}_n\cdot\overline{\boldsymbol{I}}_{\nabla b\nabla w;i,j}\right) \tag{2.52a}$$

$$+\overline{\Sigma}^0_{t;i,j,g} + \tau\boldsymbol{\Omega}_n\cdot\overline{\boldsymbol{\Sigma}}^1_{t;i,j,g}\Bigg]\alpha_{i,n,g}, \tag{2.52b}$$

$$\left(\mathcal{M}\mathcal{S}^p\beta\right)_{j,n,g} = \sum_{\ell=0}^{L}\frac{2\ell+1}{4\pi}\sum_{m=-\ell}^{\ell}Y_\ell^m\left(\boldsymbol{\Omega}_n\right)\sum_{i=1}^{J}\sum_{g'=1}^{G}\left(\overline{\Sigma}^0_{s;i,j,\ell,g'\to g} + \tau\boldsymbol{\Omega}_n\cdot\overline{\boldsymbol{\Sigma}}^1_{s;i,j,\ell,g'\to g}\right)\beta^m_{i,\ell,g'}, \tag{2.52c}$$

$$\left(\mathcal{M}\mathcal{F}^p\beta\right)_{j,n,g} = \frac{1}{4\pi}\sum_{i=1}^{J}\sum_{g'=1}^{G}\left(\overline{\Sigma}^0_{f;i,j,g'\to g} + \tau\boldsymbol{\Omega}_n\cdot\overline{\boldsymbol{\Sigma}}^1_{f;i,j,g'\to g}\right)\beta^m_{i,\ell,g'}, \tag{2.52d}$$

$$\left(\mathcal{R}^p\alpha^b\right)_{j,n,g} = \rho\sum_{i=1}^{J}\left(\sum_{u:\boldsymbol{\Omega}_n\cdot\boldsymbol{n}_k<0}\left|\boldsymbol{\Omega}_n\cdot\boldsymbol{n}_u\right|\bar{\bar{i}}^S_{bw;i,j,u}\right)\alpha^b_{i,n_r,g}. \tag{2.52e}$$

$$\left(q^p\right)_{j,n,g} = \sum_{u:\boldsymbol{\Omega}_n\cdot\boldsymbol{n}_k<0}\left|\boldsymbol{\Omega}_n\cdot\boldsymbol{n}_u\right|\overline{\psi}^{ext}_{j,n,g,u} \tag{2.52f}$$

$$+\sum_{\ell=0}^{L}\frac{2\ell+1}{4\pi}\sum_{m=-\ell}^{\ell}Y_\ell^m\left(\boldsymbol{\Omega}_n\right)\left(\overline{Q}^{m,0}_{j,\ell,g} + \tau\boldsymbol{\Omega}_n\cdot\overline{\boldsymbol{Q}}^{m,1}_{j,\ell,g}\right). \tag{2.52g}$$

As no further approximations have been made to the discretized equations, the results for particle balance from Sec. 2.7 directly follow. The boundary surfaces are assumed to have a constant normal direction, $\boldsymbol{n}_u$, which allows the surface integrals $\bar{\bar{i}}^S_{bw;i,j,u}$ to be performed independent of direction. The surface integrals become the sum of the contributions from all the boundary surfaces downwind ($u:\boldsymbol{\Omega}_n\cdot\boldsymbol{n}_u<0$) or upwind ($u:\boldsymbol{\Omega}_n\cdot\boldsymbol{n}_u>0$) of the problem in the direction $\boldsymbol{\Omega}_n$.

2.8.2 Basis cross section approximation

The cross sections from the full cross section discretization have a high number of dependencies. The scattering cross section, for instance, depends on the basis function, weight function and its gradient, scattering moment, incoming group and outgoing group. To simplify these cross sections and reduce the number of cross sections to store, a second cross section discretization, referred to in this dissertation as the basis

cross section discretization, begins with the postulate that a cross section representation of the form

$$\int_{V_{i,j}} \tilde{w}_{j,n} \Sigma_{t;g} b_i dV \approx \overline{\Sigma}_{t;i,g} \int_{V_{i,j}} \tilde{w}_{j,n} b_i dV \tag{2.53}$$

can conserve global particle balance. To derive a cross section such that this global balance is preserved, the equation is summed over the weight functions j [with the assumption that Eq. (2.40) holds] to get

$$\int_{V_i} \Sigma_{t;g} b_i dV = \overline{\Sigma}_{t;i,g} \int_{V_i} b_i dV. \tag{2.54}$$

Note that the region of integration is no longer over dependent on the weight function region of support V_j but instead on the basis function region of support V_i. The left side of this equation is the collision term in the particle balance equation. The approximate cross section is solved for in this equation to get

$$\overline{\Sigma}_{t;i,g} \equiv \frac{\int_{V_i} \Sigma_{t;g} b_i dV}{\int_{V_i} b_i dV}. \tag{2.55}$$

With this approximate cross section, the transport equations reduce to the balance equation as before. The balance conditions are applied to the scattering and fission cross sections as well, which produces the same normalized weighting by the basis function,

$$\overline{\Sigma}_{f;i,g'\to g} \equiv \frac{\int_{V_i} \chi_g \nu_{g'} \Sigma_{f;g'} b_i dV}{\int_{V_i} b_i dV}, \tag{2.56}$$

$$\overline{\Sigma}_{s;i,\ell,g'\to g} \equiv \frac{\int_{V_i} \Sigma_{s;\ell,g'\to g} b_i dV}{\int_{V_i} b_i dV}. \tag{2.57}$$

This approximation significantly reduces the number of cross section integrals to compute and store, as the cross section integrals are no longer dependent on the weight function or its gradient. With these approximations, the transport equation becomes

$$\begin{aligned}\left(\mathcal{L}^b \alpha\right)_{j,n,g} = \sum_{i=1}^{J} \Bigg[&\sum_{u:\boldsymbol{\Omega}_n\cdot\boldsymbol{n}_u>0} \left(\boldsymbol{\Omega}_n \cdot \boldsymbol{n}_u\right) \bar{i}^{S}_{bw;i,j,u} - \boldsymbol{\Omega}_n \cdot \bar{\boldsymbol{i}}^{V}_{b\nabla w;i,j} + \tau \boldsymbol{\Omega}_n \cdot \left(\boldsymbol{\Omega}_n \cdot \overline{\boldsymbol{I}}_{\nabla b \nabla w;i,j}\right) \\ &+ \overline{\Sigma}_{t;i,g} \left(\bar{i}^{V}_{bw;i,j} + \tau \boldsymbol{\Omega}_n \cdot \bar{\boldsymbol{i}}^{V}_{b\nabla w;i,j}\right) \Bigg] \alpha_{i,n,g},\end{aligned} \tag{2.58a}$$

$$\left(\mathcal{M}\mathcal{S}^b \beta\right)_{j,n,g} = \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y_\ell^m\left(\boldsymbol{\Omega}_n\right) \sum_{i=1}^{J} \sum_{g'=1}^{G} \overline{\Sigma}_{s;i,\ell,g'\to g} \left(\bar{i}^{V}_{bw;i,j} + \tau \boldsymbol{\Omega}_n \cdot \bar{\boldsymbol{i}}^{V}_{b\nabla w;i,j}\right) \beta^m_{i,\ell,g'}, \tag{2.58b}$$

$$\left(\mathcal{M}\mathcal{F}^b \beta\right)_{j,n,g} = \frac{1}{4\pi} \sum_{i=1}^{J} \sum_{g'=1}^{G} \overline{\Sigma}_{f;i,g'\to g} \left(\bar{i}^{V}_{bw;i,j} + \tau \boldsymbol{\Omega}_n \cdot \bar{\boldsymbol{i}}^{V}_{b\nabla w;i,j}\right) \beta^m_{i,\ell,g'}. \tag{2.58c}$$

The basis-weighted operators have the superscript b. The source and reflection operators are unchanged from Eq. (2.52).

In addition to preserving particle balance as discussed in Sec. 2.7, the SUPG stabilization also allows the cross sections to be integrated without directional dependence. If the weight functions were instead skewed or offset in the upwind direction, these integrals cross sections would additionally be dependent on the direction $\boldsymbol{\Omega}_n$, which would increase the number of integration variables by one to several orders of magnitude.

2.9 Discretization of the strong form of the transport equation

The weak form of the transport equation is converted back into the strong form by replacing each weight function in the weak transport equation [Eq. (2.21)] with a Dirac delta function,

$$w_j(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{x}_j), \tag{2.59}$$

which has the property

$$\int_V \delta(\boldsymbol{x} - \boldsymbol{x}_j) f(\boldsymbol{x}) \, dV = f(\boldsymbol{x}_j). \tag{2.60}$$

This results in the equations

$$\left(\mathcal{L}^t \psi\right)_{n,g} = \boldsymbol{\Omega}_n \cdot \boldsymbol{\nabla} \psi_{n,g}(\boldsymbol{x}_j) + \Sigma_{t;g}(\boldsymbol{x}_j) \psi_{n,g}(\boldsymbol{x}_j), \tag{2.61a}$$

$$\left(\mathcal{S}^t \phi\right)^m_{\ell,g} = \sum_{g'} \Sigma_{s;\ell,g'\to g}(\boldsymbol{x}_j) \phi^m_{\ell,g'}(\boldsymbol{x}_j), \tag{2.61b}$$

$$\left(\mathcal{F}^t \phi\right)^m_{\ell,g} = \delta_{\ell,0} \chi_g(\boldsymbol{x}_j) \sum_{g'=1}^{G} \nu_{g'}(\boldsymbol{x}_j) \Sigma_{f;g'}(\boldsymbol{x}_j) \phi^0_{0,g'}(\boldsymbol{x}_j), \tag{2.61c}$$

$$\left(q^t\right)_{n,g} = q_{n,g}(\boldsymbol{x}_j). \tag{2.61d}$$

The discrete-to-moment and moment-to-discrete operators remain the same as for the weak case. The boundary condition [Eq. (2.16b)] is weighted by the same Dirac delta function,

$$\psi_{n,g}(\boldsymbol{x}_j) = \psi^{inc}_{n,g}(\boldsymbol{x}_j) + \rho \psi_{n_r,g}(\boldsymbol{x}_j). \tag{2.61e}$$

To complete the discretization, basis function expansions [Eqs. (2.30)] are inserted into the operators,

$$\left(\mathcal{L}^t \psi\right)_{n,g} = \sum_{i=1}^{J} \left[\boldsymbol{\Omega}_n \cdot \boldsymbol{\nabla} b_i(\boldsymbol{x}_j) + \Sigma_{t;g}(\boldsymbol{x}_j) b_i(\boldsymbol{x}_j)\right] \alpha_{i,n,g}, \tag{2.62a}$$

$$\left(\mathcal{S}^t\phi\right)^m_{\ell,g} = \sum_{g'} \Sigma_{s;\ell,g'\to g}\left(\boldsymbol{x}_j\right) \sum_{i=1}^{J} b_i\left(\boldsymbol{x}_j\right) \beta^m_{i,\ell,g'}, \tag{2.62b}$$

$$\left(\mathcal{F}^t\phi\right)^m_{\ell,g} = \delta_{\ell,0}\chi_g\left(\boldsymbol{x}_j\right) \sum_{g'=1}^{G} \nu_{g'}\left(\boldsymbol{x}_j\right) \Sigma_{f;g'}\left(\boldsymbol{x}_j\right) \sum_{i=1}^{J} b_i\left(\boldsymbol{x}_j\right) \beta^m_{i,\ell,g'}, \tag{2.62c}$$

and the boundary condition,

$$\sum_{i=1}^{J} b_i\left(\boldsymbol{x}_j\right) \alpha_{i,n,g} = \psi^{inc}_{n,g}\left(\boldsymbol{x}_j\right) + \rho \sum_{i=1}^{J} b_i\left(\boldsymbol{x}_j\right) \alpha_{i,n_r,g}, \tag{2.62d}$$

which completes the discretization of the transport equations. The superscript t denotes the operators specific to the strong form. This is the standard strong form of the transport equations. Since the Dirac delta functions chosen as weight functions do not satisfy Eq. (2.40), there is no guarantee of conservation.

Unlike for the weak form of the transport equation, the cross sections in Eqs. (2.62) are not volume-weighted. They are equivalent to the full cross section discretization from Sec. 2.8 with a Dirac delta weight function. If, however, the strong-form equations are derived directly from the fully-discretized transport equations with basis function cross section weighting [Eqs. (2.58)], they retain volume weighting of the cross sections,

$$\left(\mathcal{L}^{tb}\alpha\right)_{j,n,g} \sum_{i} \left[\boldsymbol{\Omega}_n \cdot \boldsymbol{\nabla} b_i\left(\boldsymbol{x}_j\right) + \overline{\Sigma}_{t;i,g} b_i\left(\boldsymbol{x}_j\right)\right] \alpha_{i,n,g}, \tag{2.63a}$$

$$\left(\mathcal{M}\mathcal{S}^{tb}\beta\right)_{j,n,g} = \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y^m_\ell\left(\boldsymbol{\Omega}_n\right) \sum_{i=1}^{J} \sum_{g'=1}^{G} \overline{\Sigma}_{s;i,\ell,g'\to g} b_i\left(\boldsymbol{x}_j\right) \beta^m_{i,\ell,g'}, \tag{2.63b}$$

$$\left(\mathcal{M}\mathcal{F}^{tb}\beta\right)_{j,n,g} = \frac{1}{4\pi} \sum_{i=1}^{J} \sum_{g'=1}^{G} \overline{\Sigma}_{f;i,g'\to g} b_i\left(\boldsymbol{x}_j\right) \beta^m_{i,\ell,g'}. \tag{2.63c}$$

The superscript th denotes operators specific to the basis-weighted collocation operators. The boundary conditions from Eq. (2.62d) still apply. Equations (2.63) also carry no guarantee of particle conservation but do include integral weighting of the cross sections. In Sec. 7.3.3, this is shown to improve the result of the strong-form equation for a problem with heterogeneous materials.

For the strong-form equations, the boundary condition is not incorporated directly into the transport equation. Instead, the points $\boldsymbol{x}_j$ are separated into those on the problem boundary and those inside the problem domain. The boundary condition is applied to every boundary point on upwind boundaries, $\boldsymbol{\Omega}_n \cdot \boldsymbol{n} < 0$, while the transport equation is applied to every point inside the domain and every boundary point not on upwind boundaries, $\boldsymbol{\Omega}_n \cdot \boldsymbol{n} > 0$. The iterative methods for the weak form [Eqs. (2.38)] apply for these equations with the strong-form operators as well, with the distinction that the $\mathcal{L}$ operator applies the

boundary condition for the upwind boundary points [Eq. (2.62d)].

2.10 Functions for meshless methods

The basis functions and weight functions that provide the functional expansions and test functions in the transport equations [Eqs. (2.32) and (2.62)] have not been defined. The two requirements discussed in Sec. 2.7 for the weak form to satisfy the conservation equation are that the weight functions sum to one [Eq. (2.40)] and that the basis functions can accurately represent the exact solution. A third property that makes computation much more efficient is if the radius of support for the basis and weight functions is compact. A compact function has a value of zero outside of its support region. This limits the number of basis functions that overlap with each weight function and the number of functions with a nonzero value at any given point, which makes the integrals in the MLPG equations tractable and leads to sparse matrices for the streaming operator (see Sec. 2.6).

2.10.1 Radial basis functions

A common class of functions used in meshless methods is the radial basis function (RBF). The value of an RBF depends only on the distance traversed by the input vector, regardless of the number of dimensions. An example of a RBF with the dimensionless distance r is the Gaussian function

$$\Gamma_{\text{gauss}}(r) = e^{-r^2}. \tag{2.64}$$

To make this RBF compact, the value of the Gaussian function can be decreased by a constant to have a zero value at and after a finite value R of the dimensionless radial value r,

$$\Gamma_{\text{comp gauss}}(r) = \begin{cases} \frac{e^{-r^2} - e^{R^2}}{1 - e^{-R^2}}, & r < R, \\ 0, & \text{otherwise.} \end{cases} \tag{2.65}$$

A good choice for R may be around $R = 5.0$, which limits the discontinuity in the derivative at $r = R$ to around 10^{-10}. When integrating the basis and weight functions, this discontinuity can decrease the accuracy of the integration. Other popular RBFs include the multiquadric and inverse multiquadric functions,

$$\Gamma_{\text{mult}}(r) = \sqrt{1 + r^2}, \tag{2.66}$$

$$\Gamma_{\text{inv mult}} = \frac{1}{\sqrt{1 + r^2}}. \tag{2.67}$$

These functions are shown in Fig. 2.1b.

There also exist compact RBF functions with no derivative at the edge of the support radius, such as the Wendland functions [47],

$$\Gamma_{x,y}(r) = \begin{cases} \gamma_{x,y}(r) & r \leq 1, \\ 0, & \text{otherwise}, \end{cases} \tag{2.68}$$

where a few of the functions are defined as

$$\gamma_{1,1}(r) = (1-r)^3 (1+3r), \tag{2.69a}$$

$$\gamma_{1,2}(r) = (1-r)^5 \left(1+5r+8r^2\right), \tag{2.69b}$$

$$\gamma_{1,3}(r) = (1-r)^7 \left(1+7r+19r^2+21r^3\right), \tag{2.69c}$$

$$\gamma_{3,1}(r) = (1-r)^4 (4r+1), \tag{2.69d}$$

$$\gamma_{3,2}(r) = (1-r)^6 \left(3+18r+35r^2\right), \tag{2.69e}$$

$$\gamma_{3,3}(r) = (1-r)^8 \left(1+8r+25r^2+32r^3\right). \tag{2.69f}$$

These functions are computationally inexpensive to evaluate, but unlike the Gaussian, multiquadric, and inverse multiquadric functions, they have limited smoothness (i.e. a limited number of continuous derivatives). For the weak transport equation, which only contains first derivatives, this is not a practical limitation.

RBFs can be used directly as basis or weight functions in the Petrov-Galerkin method from Sec. 2.5,

$$g_i(\boldsymbol{x}) = \Gamma\left(\epsilon_i d(\boldsymbol{x}, \boldsymbol{x}_i)\right), \tag{2.70}$$

where d is some distance measure such as the Euclidean distance,

$$d_{L_2}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{2.71}$$

This L_2 distance creates functions that are radially symmetric about the center of the function at $\boldsymbol{x}_i$. The L_1 distance would create a square region of support. For the following chapters, the L_2 distance is used in all cases. These functions have the shape parameter ϵ_i, which controls the support radius of the function. If the support radius of an RBF is r_{sup} [e.g. $r_{sup} = 1$ for the example in Eq. (2.68)] and the desired support radius of the basis or weight function is r_i, the shape parameter is defined as

$$\epsilon_i = \frac{r_{sup}}{r_i}. \tag{2.72}$$

The gradient of the RBF is taken using the chain rule,

$$\boldsymbol{\nabla} g_i(\boldsymbol{x}) = \epsilon_i \left[\boldsymbol{\nabla} d(\boldsymbol{x}, \boldsymbol{x}_i)\right] \circ \left[\boldsymbol{\nabla} \Gamma\left(\epsilon_i d(\boldsymbol{x}, \boldsymbol{x}_i)\right)\right], \tag{2.73}$$

where the $\circ$ denotes the Hadamard (or entrywise) product. Sets of radial basis functions do not generally form a partition of unity, i.e.

$$\text{there exists } \boldsymbol{x} \text{ such that } \sum_{i=1}^{I} g_i(\boldsymbol{x}) \neq 1, \tag{2.74}$$

which means that they do not meet the desired requirements for particle conservation.

2.10.2 Moving least squares functions

The moving least squares (MLS) method transforms a meshless basis into a new basis that can represent a certain order of polynomials exactly. When used for interpolation, the method minimizes the error of the function with respect to given values in a least-squares sense. The MLS functions are defined as [48]

$$h_i(\boldsymbol{x}) = \boldsymbol{p}^{\top}(\boldsymbol{x}) \boldsymbol{A}^{-1}(\boldsymbol{x}) \boldsymbol{B}_i(\boldsymbol{x}), \tag{2.75a}$$

where

$$\boldsymbol{p}^{\top}(\boldsymbol{x}) = \begin{bmatrix} 1 & x & y & z & xy & xz & yz & x^2 & y^2 & z^2 & \dots \end{bmatrix}, \tag{2.75b}$$

$$\boldsymbol{A}(\boldsymbol{x}) = \sum_{i=1}^{n} g_i(\boldsymbol{x}) \boldsymbol{p}(\boldsymbol{x}_i) \boldsymbol{p}^{\top}(\boldsymbol{x}_i), \tag{2.75c}$$

$$\boldsymbol{B}_i(\boldsymbol{x}) = g_i(\boldsymbol{x}) \boldsymbol{p}(\boldsymbol{x}_i). \tag{2.75d}$$

The $g_i(\boldsymbol{x})$ functions are the RBF weighting functions from the other meshless basis. The action of $\boldsymbol{A}^{-1}(\boldsymbol{x})$ is to normalize the value of the basis functions at the point $\boldsymbol{x}$ so the new basis can represent functions in the set of polynomials $\boldsymbol{p}(\boldsymbol{x})$ exactly. When multiplied by $\boldsymbol{B}_i(\boldsymbol{x})$, the function represents the value for the new, normalized basis $h_i(\boldsymbol{x})$. Derivatives of the MLS functions can be taken using the product rule,

$$\frac{\partial h_i(\boldsymbol{x})}{\partial x_j} = \frac{\partial \boldsymbol{p}^{\top}(\boldsymbol{x})}{\partial x_j} \boldsymbol{A}^{-1}(\boldsymbol{x}) \boldsymbol{B}_i(\boldsymbol{x}) + \boldsymbol{p}^{\top}(\boldsymbol{x}) \frac{\partial \boldsymbol{A}^{-1}(\boldsymbol{x})}{\partial x_j} \boldsymbol{B}_i(\boldsymbol{x}) + \boldsymbol{p}^{\top}(\boldsymbol{x}) \boldsymbol{A}^{-1}(\boldsymbol{x}) \frac{\partial \boldsymbol{B}_i(\boldsymbol{x})}{\partial x_j}, \tag{2.76a}$$

Algorithm 2.1 Calculation of the values and derivatives of MLS functions

function MLS VALUES($\boldsymbol{x}$)
  find all functions with a nonzero value at $\boldsymbol{x}$
  for each local function i **do**
    calculate the value of the RBF function $g_i(\boldsymbol{x})$ and its derivative $\partial_{x_j} g_i(\boldsymbol{x})$ for each dimension j
    calculate the value of the polynomial function $p(\boldsymbol{x}_i)$
  end for
  calculate the value of the polynomial function $p(\boldsymbol{x})$ and its derivative $\partial_{x_j} p(\boldsymbol{x})$ for each dimension j
  calculate the matrices $\boldsymbol{A}(\boldsymbol{x})$ and $\partial_{x_j} \boldsymbol{A}(\boldsymbol{x})$ for each dimension j
  invert or **decompose** the matrix $\boldsymbol{A}(\boldsymbol{x})$
  calculate the matrix $\partial_{x_j} \boldsymbol{A}^{-1}(\boldsymbol{x})$
  for each local function i **do**
    calculate the vectors $\boldsymbol{B}_i(\boldsymbol{x})$ and $\partial_{x_j} \boldsymbol{B}_i(\boldsymbol{x})$ for each dimension j
    calculate the value of the MLS function $h_i(\boldsymbol{x})$ and its derivatives $\partial_{x_j} h_i(\boldsymbol{x})$ for each dimension j
  end for
  return values and derivatives of MLS functions
end function

with the values

$$\frac{\partial \boldsymbol{B}_i(\boldsymbol{x})}{\partial x_j} = \frac{\partial g_i(\boldsymbol{x})}{\partial x_j} \boldsymbol{p}(\boldsymbol{x}_i), \tag{2.76b}$$

$$\frac{\partial \boldsymbol{A}^{-1}(\boldsymbol{x})}{\partial x_j} = -\boldsymbol{A}^{-1}(\boldsymbol{x}) \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_j} \boldsymbol{A}^{-1}(\boldsymbol{x}), \tag{2.76c}$$

$$\frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_j} = \sum_{i=1}^{n} \frac{\partial g_i(\boldsymbol{x})}{\partial x_j} \boldsymbol{p}(\boldsymbol{x}_i) \boldsymbol{p}^{\top}(\boldsymbol{x}_i). \tag{2.76d}$$

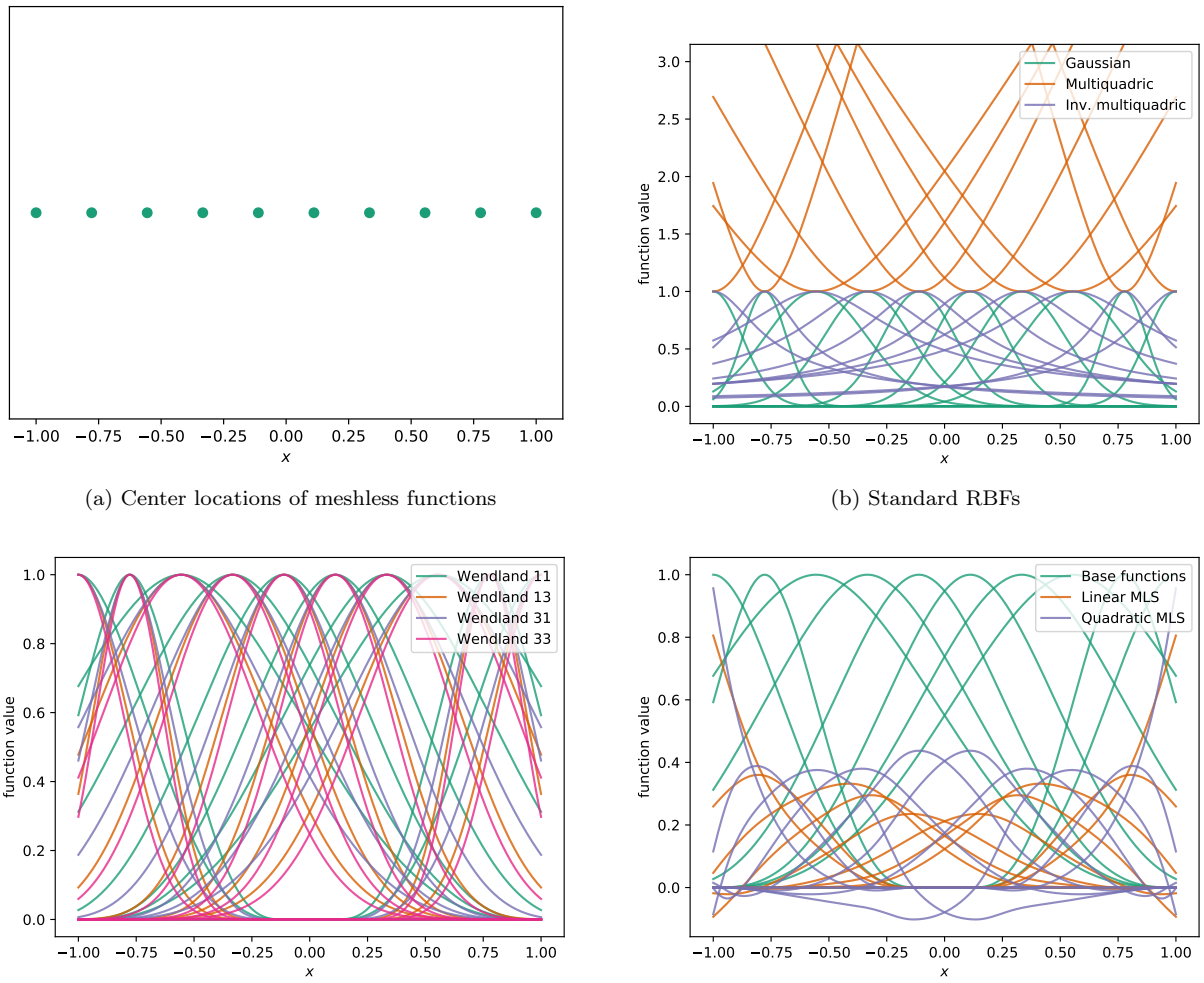
In order for the inverse of the $\boldsymbol{A}$ matrix to be well-defined, there should be at least as many parent RBF functions at every point in the domain as there are polynomials in the set $\boldsymbol{p}$. Ideally, the values of every MLS function desired at a given point are calculated simultaneously, as in Alg. 2.1.

The MLS functions meet all three of the conditions mentioned at the start of this section. Unlike the RBF basis functions, MLS functions do form a partition of unity,

$$\sum_i h_i(\boldsymbol{x}) = 1 \text{ for every } \boldsymbol{x}, \tag{2.77}$$

which allows for proof of convergence in a Petrov-Galerkin method. In addition, the ability to represent polynomials locally gives some indication that the functions can accurately represent a continuous function, because in the limit of many functions the solution in the support region of any one function appears constant or linear. Finally, by using compact RBF functions to create the MLS basis, the MLS functions themselves are compact.

Figure 2.1 shows various meshless functions for a simple set of equally-spaced centers in 1D. The standard RBF functions are global. The multiquadric functions in particular do not converge to a finite limit as



(a) Center locations of meshless functions

(b) Standard RBFs

(c) Wendland RBFs

(d) Moving least squares functions with the original Wendland11 functions

Figure 2.1: Meshless functions in 1D for 10 equally-spaced center positions and 8 neighbors in radius calculation

$r \to \infty$. Unlike the global RBFs, the Wendland RBFs have a finite support radius. Note, however, that like the standard RBFs, the sum of these functions is not equal to one at every point. This prevents proof of neutron conservation (Sec. 2.7). Finally, note the MLS functions in Fig. 2.1d that are created using the initial basis of the Wendland 11 RBFs in Fig. 2.1c. The sum of all the MLS functions at every point in the domain is equal to one. The linear MLS functions are mostly positive except near the boundaries, where they can go negative. The quadratic MLS functions, on the other hand, have negative values for much larger sections of the domain. Because of this, the results in Chapters 5, 6 and 7 use the linear MLS functions. Empirical tests showed no improvement using the quadratic functions instead of the linear functions in transport calculations.

Chapter 3

Implementation of meshless neutron transport

The process of numerically solving the MLPG form of the neutron transport equations from Chapter 2 involves:

1. Creating the energy and angular discretizations (Sec. 3.1);
2. Creating the spatial discretization (Sec. 3.2), including
   (a) Defining the physical parameters of the system, such as the material properties, boundary sources and problem dimensions;
   (b) Creating the weight functions; and
   (c) Performing integration of the basis and weight functions, the boundary and internal sources and the cross sections (Chapter 4); and
3. Defining the operators representing the individual terms of the transport equation and solving the resulting linear system of equations using an appropriate solution technique (Sec. 3.4).

This section discusses how these steps are optimized to keep the total simulation cost reasonably low, including through parallelization (Sec. 3.5). Performance figures for the MLPG method are included in Sec. 3.6.

3.1 Energy and angular discretization

The independent variables for the energy discretization in the multigroup equations include the number of energy groups and the upper and lower energy bounds of each group. These parameters define which energies each group represents in the integration of the the multigroup cross sections, the source and the neutron flux (Sec. 2.1.1). If cross sections are generated through a Monte Carlo simulation, these energies are used as the energy bounds for the energy tally. For all of the problems in Chapters 5, 6 and 7, the cross sections are generated or chosen ahead of the meshless simulation and the benchmark solutions (where applicable) are generated in multigroup space. This is done to isolate the spatial discretization error, which

may be overshadowed by the error in converting the continuous-energy cross sections to multigroup cross sections.

The independent variables for the angular discretization are the number of scattering moments for the scattering cross section and the number of directions for the discrete form of the solution. The number of scattering moments defines the angular size of the scattering cross section and the number of spherical harmonic moments. The directions of flight $\mathbf{\Omega}_n$ are a set of integration ordinates on the sphere. In one-dimensional Cartesian geometry, the angular dependence is axially symmetric about the x axis. Gauss-Legendre ordinates and weights are used for the one-dimensional angular ordinates and weights. In two-dimensional Cartesian geometry, the angular dependence is symmetric about the $x-y$ plane. For the two- and three-dimensional problems in this dissertation, the ordinates and weights for integration over the unit sphere are from the LDFE quadrature [49]. The discrete-ordinates assumption made for the angular discretization in Sec. 2.1.2 does not preclude convergence to a continuous-in-angle solution. As the number of directions increases, the error due to the angular discretization should decrease.

3.2 Spatial discretization

In the discretized transport equations [Eq. (2.52)], there are no constraints on the spatial dependence of the cross sections, internal source and boundary sources. These values can be provided directly as functions or, as in a Monte Carlo code, by a constructive solid geometry (CSG). The CSG uses analytic surfaces to define regions of (usually constant) material. For more information the implementation of the CSG, see Appendix B. The manufactured solutions in Chapter 5 use functional forms for the cross sections, while the results for more realistic problems in Chapters 6 and 7 use a CSG. The full initialization process for the spatial discretization is shown in Alg. 3.2.

Algorithm 3.2 Initialization of the spatial discretization

initialize spatial discretization options
read in positions of the basis and weight function centers
create a k-d tree for distance calculations
calculate the radii of the meshless functions
initialize RBF functions
for each basis or weight function i **do**
    find all other basis and weight functions that intersect with function i
    check that no other centers are too close to the center of function i
end for
create MLS functions using RBF functions
create the basis and weight functions using the MLS functions
find the basis and weight functions that intersect with the boundaries of the problem
read in cross sections and sources
perform basis and weight function integration

3.2.1 Basis and weight functions

The centers of the basis and weight functions can be placed independently of one another and of the problem geometry, with a few caveats. For accurate representation of the solution, a higher concentration of centers should be placed in areas where high gradients are expected. If the weight function centers are placed too close together, the equations are no longer be independent and the system is underdetermined, while if the basis centers are placed too close together, the unknown variables are no longer be independent and the system is overdetermined. To prevent these issues, the code checks each basis and weight function center for other centers that are too close. This is quantified for each center by comparing the radius of the meshless function r_i and the distance to the nearest other center d_i^{nearest}. This is quantified in the code by a ratio of these two quantities, which should be larger than a predefined tolerance ϵ,

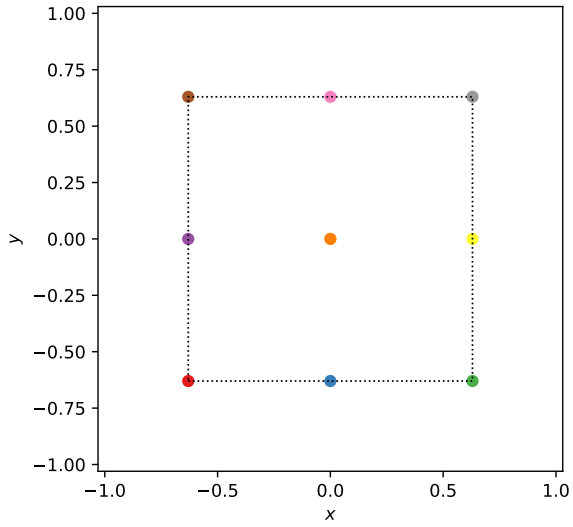
$$\epsilon < \frac{d_i^{\text{nearest}}}{r_i},$$

which for these calculations is defined to be $\epsilon = 10^{-3}$. For randomized points, this tolerance may need to be increased to properly protect against an ill-conditioned set of points.

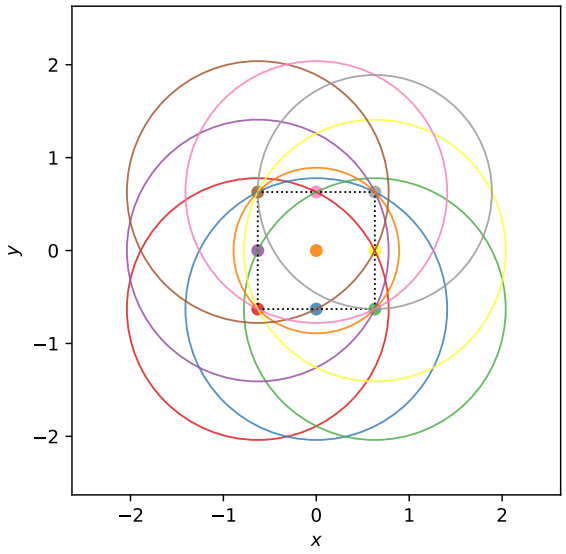
The algorithm to calculate the radii of the basis and weight functions is listed in Alg. 3.3. The purpose of the algorithm is to ensure that at the center of each basis or weight function, at least N nearest points have a nonzero value. If the point spacing is not too irregular (e.g. sets of isolated points that don't "see" other any other sets), this also ensures basis and weight function coverage throughout the problem domain. The connectivity of the chosen points and radii is calculated using a k-d tree [50] from nanoflann [51]. To find which basis functions overlap with each weight function, the weight function radius is added to the maximum basis function radius in the problem and a radius search is performed to find candidates for intersection, which are then individually checked for actual intersection.

Algorithm 3.3 Calculation of the support radii for the basis and weight functions

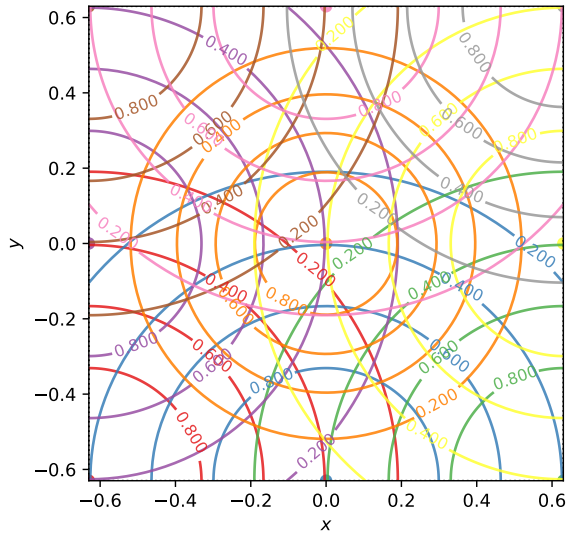
```
initialize all radii to zero
for each point i do
    find the nearest N neighboring points
    for each neighboring point j do
        calculate the distance from point j to point i
        if current radius of point j < distance to point i then
            set current radius of point j = distance to point i
        end if
    end for
end for
for each point i do
    multiply the radius of point i by a chosen scalar
end for
```



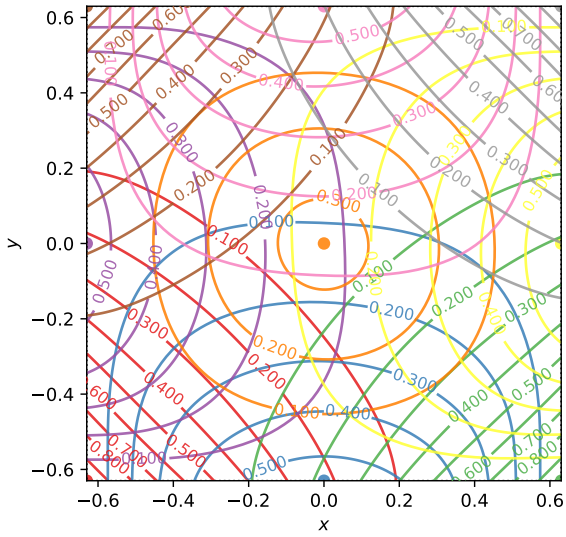
(a) Centers are placed at chosen locations in the domain and on its boundary.



(b) The radii are chosen such that a specified number of functions extend to or past each point.



(c) A standard RBF basis using Wendland 11 functions is created using the functions and radii. The functions are symmetric and do not sum to a constant.



(d) Values of the linear MLS functions are calculated by normalizing the RBF basis. The functions are asymmetric and sum to one at every point in the domain.

Figure 3.1: Process of creating a set of MLS functions

A one-dimensional plot of MLS functions with a Wendland function basis in 1D is shown in Fig. 2.1. The process of creating a set of MLS functions is illustrated for a 2D example in Fig. 3.1. The centers of the basis and weight functions are chosen to be a set of 3^2 Cartesian points. The radii are calculated such that for each function, eight total centers, including the center of the original function, are included in or on the edge of the support radius. The standard RBF function basis is created with these support radii. As in

Fig. 2.1c, these functions do not sum to one at every point in the domain. Note that each RBF function is symmetric about its center. Using the original RBF basis, the values of the MLS functions can be calculated anywhere in the domain. The values of these functions in Fig. 3.1d do sum to one at every point in the domain but are no longer symmetric about the centers.

Once a set of basis and weight functions is created, the integration can be performed as described in Chapter 4. For the strong-form equations, the integration is only performed for the basis function weighting [Eqs. (2.63)], as the point weighting uses cross sections evaluated at the collocation points. The center points of the basis functions are also used as the evaluation (or collocation) positions for the equations. Because of this, a few additional constraints are placed on the placement of the basis function centers for the strong form. First, there must be points on the problem boundaries to satisfy the boundary conditions, which unlike the weak form are not included in the transport equation. Second, no boundary point should be on more than one boundary surface, such as at a corner, as this would create a contradiction in the value of the surface normal and definition of which points are upwind for a chosen direction. The final constraint is the same as for the weak form, namely that to avoid ill-conditioning, the basis function centers should be placed as evenly as is practical for the problem geometry.

3.2.2 SUPG stabilization

The parameter that controls the addition of SUPG stabilization into the problem, τ from Eq. (2.28), is chosen to be weight-function dependent (i.e. $\tau \to \tau_j$) according to the radius of the weight function r_j,

$$\tau_j = cr_j, \tag{3.1}$$

where c is user-defined constant. With the shape parameter from Eq. (2.72), the RBF value and derivative from Eqs. (2.70) and (2.73) and the τ_j value from Eq. (3.1) with a dimensionless support radius of $r_{sup} = 1$, the SUPG-augmented weight function [Eq. (2.28)] becomes

$$\tilde{w}_{j,n} = \Gamma\left(\epsilon_j d\left(\boldsymbol{x}, \boldsymbol{x}_i\right)\right) + c\boldsymbol{\Omega}_n \cdot \left(\left[\boldsymbol{\nabla} d\left(\boldsymbol{x}, \boldsymbol{x}_i\right)\right] \circ \left[\boldsymbol{\nabla}\Gamma\left(\epsilon_j d\left(\boldsymbol{x}, \boldsymbol{x}_i\right)\right)\right]\right). \tag{3.2}$$

Equation (3.2) shows that for the of τ_j in Eq. (3.1), the function-to-derivative ratio in the stabilized weight function becomes independent of the weight function radius r_j. A constant τ would add comparatively more numerical diffusion for weight functions with smaller radii. This does mean that the condition for

conservation [Eq. (2.41)] is not preserved, as

$$\sum_{j=1}^{J} \tau_j \mathbf{\Omega}_n \cdot \boldsymbol{\nabla} w_j \neq 0 \tag{3.3}$$

in general, even where Eq. (2.40) holds. However, in empirical tests, the weight function-dependent τ_j shows better performance and convergence for problems with varying radii r_j than a constant τ.

In both the full [Eqs. (2.52)] and basis [Eqs. (2.58)] fully-discretized equations, the moment-to-discrete operator attached to the scattering, fission and implicitly the source can be separated by separating the terms including τ from the terms included in the transport equation without SUPG. The summation of the SUPG terms is then performed using a modified moment-to-discrete operator. This adds an additional index for the gradient terms, which is d in the following equations. For example, for the "full" cross section equations, the scattering term becomes

$$\left(\mathcal{S}^d \beta\right)_{j,n,g,d} = \begin{cases} \sum_{i=1}^{J} \sum_{g'=1}^{G} \overline{\Sigma}^0_{s;i,j,\ell,g' \to g} \beta^m_{i,\ell,g'}, & d = 0, \\ \sum_{i=1}^{J} \sum_{g'=1}^{G} \left(\overline{\mathbf{\Sigma}}^1_{s;i,j,\ell,g' \to g}\right)_d \beta^m_{i,\ell,g'}, & d = 1,2,3. \end{cases} \tag{3.4}$$

and the moment-to-discrete operator becomes

$$\left(\mathcal{M}^d \beta\right)_{j,n,g} = \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y^m_\ell \left(\mathbf{\Omega}_n\right) \sum_{d=0}^{3} f_d \tilde{\beta}^m_{j,\ell,g,d}, \tag{3.5}$$

where $\tilde{\beta}^m_{j,\ell,g,d}$ is the result of the $\mathcal{S}^d$ scattering operator and

$$f_d = \begin{cases} 1, & d = 0, \\ \tau \left(\mathbf{\Omega}_n\right)_d, & d = 1,2,3. \end{cases} \tag{3.6}$$

With these definitions, $\mathcal{M}^d \mathcal{S}^d \beta = \left(\mathcal{M}\mathcal{S}^p\right) \beta$. The results for the fission cross term directly follow. This same method can be applied to the volume part of the source term,

$$\left(q^d_V\right)^m_{j,\ell,g,d} = \begin{cases} \overline{Q}^{m,0}_{j,\ell,g}, & d = 0, \\ \left(\overline{\boldsymbol{Q}}^{m,1}_{j,\ell,g}\right)_d, & d = 1,2,3. \end{cases} \tag{3.7}$$

with the same moment-to-discrete operator. These optimizations cut down on unnecessary computational cost in the scattering, fission and internal source operators.

3.3 Streaming and collision operator

The $\mathcal{L}$ operator in both the weak and strong forms [Eqs. (2.32a) and (2.62a), respectively] represents the application of a set of independent, sparse matrices for each direction n and energy group g, with weighted equations corresponding to the rows. The number of nonzero elements in each row j is the number of basis functions that intersect with the applicable weight function w_j. The $\mathcal{L}^{-1}$ operator, or the inverse of the $\mathcal{L}$ operator, is equivalent to the solution of the system

$$\mathcal{L}\alpha = q \tag{3.8}$$

for the coefficients α given the vector q. The sparse $\mathcal{L}$ matrix reduces storage cost and permits the use of sparse linear algebra packages to solve Eq. (3.8).

3.3.1 Linear solver packages

Several solvers from Trilinos [52] are implemented to solve the $\mathcal{L}^{-1}$ linear problem in Eq. (3.8). The KLU solver from Amesos uses the LU decomposition to directly solve the $\mathcal{L}^{-1}$ problem. The direct solver is much more expensive in memory and computational cost than iterative solvers and is not generally needed for the weak-form equations with SUPG stabilization. The strong-form equations and the weak-form equations without SUPG stabilization have far worse conditioning without this diffusive stabilization and generally do not work with iterative solvers.

The Pseudo-Block generalized minimal residual (GMRES) solver from the Belos package is used for iterative solution of the linear problem. GMRES is a Krylov subspace method that relies on repeated applications of the $\mathcal{L}$ operator [Eq. (3.8)] to solve the linear problem. Because only the matrix and not its inverse is used, the cost of the decomposition in a direct solve is allayed and only the application of the $\mathcal{L}$ matrix is needed, not the explicit matrix. This solver can be used without a preconditioner for problems that are well-conditioned. For most problems, a preconditioner improves the performance significantly.

3.3.2 Preconditioners

To improve the conditioning of the GMRES solution, a preconditioner can be added to the problem. A preconditioner is an approximation to the inverse of the original linear system, $\mathcal{L}^{-1}$. Applying this preconditioner $\mathcal{P}^{-1}$ to both sides of the original linear equation [Eq. (3.8)] results in a left-preconditioned system,

$$\mathcal{P}^{-1}\mathcal{L}\alpha = \mathcal{P}^{-1}q. \tag{3.9}$$

For a good choice of preconditioner, the combined matrix $\mathcal{P}^{-1}\mathcal{L}$ (which need not be explicitly formed) has a lower condition number than the matrix $\mathcal{L}$. The simplest preconditioner is the matrix $\mathcal{L}^{-1}$, which solves the system exactly in one iteration but provides no speedup compared to using a direct solver. A right-preconditioned system has a similar form,

$$\mathcal{L}\mathcal{P}^{-1}\mathcal{P}\alpha = q. \tag{3.10}$$

These preconditioners are applied here using the ILUT (incomplete LU decomposition with threshold) preconditioner from the Ifpack package of Trilinos. Instead of storing the entire LU decomposition of the preconditioner $\mathcal{P}$, which is much less sparse than the original matrix, the ILUT decomposition instead stores an approximate form of the LU decomposition. The ILUT preconditioner allows specification of the level of fill and the drop tolerance of this approximate decomposition. The level of fill controls to the number of elements kept in the decomposition with respect to the original matrix. For a level of fill of 1.0, for instance, the $\mathcal{P}^{-1}$ matrix has approximately the same number of elements as the original $\mathcal{P}$ matrix. The dropping technique of the ILUT preconditioner ensures that the remaining elements of the approximate $\mathcal{P}^{-1}$ matrix are larger than the drop tolerance. For most of the calculations in Chapters 5, 6 and 7, the level of fill is 1.0 and the drop tolerance is 10^{-12}.

To precondition the streaming operator, a suitable preconditioner $\mathcal{P}$ must be defined. The most obvious option for a preconditioner is to use the streaming operator matrix itself, $\mathcal{L} = \mathcal{P}$, and then let the ILUT preconditioner form an approximate inverse. The $\mathcal{L}$ operation for the MLPG transport equations actually represents the application of linearly independent matrices for each group and direction, which can be denoted as $\mathcal{L}_{n,g}$. The ILUT preconditioner stores a separate ILUT approximation for these preconditioners $\mathcal{P}_{n,g}^{-1}$ for each direction n and group g. This is the preconditioner used for most of the results in the following chapters, but this method does have major drawbacks in memory cost. These preconditioners can be formed anew for each $\mathcal{L}^{-1}$ application, but this requires an excessive amount of computation that overshadows the benefits of the preconditioner.

The second preconditioner implemented in the code relies on the assumption that a solution for the angular flux values at the centers of the weight functions, $\psi_{n,g}(\boldsymbol{x}_j)$, would be better-conditioned than a solution for the coefficients of the basis function expansion $\alpha_{i,n,g}$. The angular flux at the weight function centers $\boldsymbol{x}_j$ is calculated from the basis function expansion as

$$\mathcal{B}\alpha = \psi, \tag{3.11}$$

where

$$\mathcal{B}_{i,j,n,g} = b_i\left(\boldsymbol{x}_j\right). \tag{3.12}$$

Solving Eq. (3.11) for α and inserting this expansion into the $\mathcal{L}$ linear system from Eq. (3.8),

$$\mathcal{L}\mathcal{B}^{-1}\psi = q, \tag{3.13}$$

results in an equation that solves for the angular flux at the weight function centers. To convert back to a solution for the angular flux expansion coefficients, the definition from Eq. (3.11) is again applied to get

$$\mathcal{L}\mathcal{B}^{-1}\mathcal{B}\alpha = q, \tag{3.14}$$

which is a right-preconditioned system for the $\mathcal{L}$ operator with the preconditioner $\mathcal{B}$. Unlike the first preconditioner, the matrix $\mathcal{B}$ is independent of both energy group and direction. The storage cost is therefore negligible, which reduces the memory requirements of solving the MLPG transport equations compared to the first preconditioner by one to two orders of magnitude. However, because this preconditioner is not as good of an approximation to the original $\mathcal{L}$ matrices, the number of GMRES iterations required to converge the solution can increase significantly. If the problem is solvable using GMRES without preconditioning, this second preconditioner often takes as many iterations as the unpreconditioned system, which indicates as expected that the preconditioner is not a good approximation to the matrix. However, the preconditioner also allows solution of problems that are too ill-conditioned for unpreconditioned GMRES, which for these problems validates the assumption that the solution for ψ instead of α would improve the conditioning of the solve.

3.4 Iteration on the scattering and fission sources

The streaming problem in Sec. 3.3 represents the solution for the angular flux for a given fixed neutron source. To solve the full transport equation, an additional solution step is required in iterating over the scattering and fission sources. The steady-state and k-eigenvalue cases are considered separately. For simplicity in notation, the Petrov-Galerkin superscripts from Eqs. (2.38) are dropped.

3.4.1 Steady-state outer iterations

For a problem with no scattering or fission, the steady-state transport equation [Eq. (2.38a)] becomes

$$\beta = \mathcal{D}\mathcal{L}^{-1}q \tag{3.15}$$

and a single application of the $\mathcal{L}^{-1}$ operator produces the solution. Equation (3.15) represents the first-flight source, or the source neutrons that have not interacted with the material. If there are reflective boundaries in the problem, a simple source iteration scheme is used to converge on the first-flight source, which includes neutrons that have been reflected but have not interacted with the material. After the neutrons have interacted with the material, they may scatter or produce fission neutrons. The neutrons from these subsequent generations are added to the first-flight source to calculate the full solution to the neutron transport equation. The iterations over the scattering and fission sources, referred to as the outer iterations, can be done directly through source (or Richardson) iteration,

$$\beta^{\ell+1} = \mathcal{D}\mathcal{L}^{-1}\mathcal{M}\left(\mathcal{S}+\mathcal{F}\right)\beta^{\ell} + \mathcal{D}\mathcal{L}^{-1}q, \tag{3.16}$$

where the index ℓ refers to the iteration. This iteration process is continued until the solution β converges to a chosen tolerance. The source iteration equation updates the scattering and fission sources for each subsequent generation of neutrons. If the initial guess for the solution is $\beta^1 = \mathcal{D}\mathcal{L}^{-1}q$, then β^{ℓ} represents the ℓ^{th} flight neutrons. For problems with high scattering ratios where the neutrons may scatter hundreds of times before leaving the problem through absorption or leakage, source iteration is not ideal as it requires an iteration to simulate each generation of neutrons.

The Krylov solvers discussed in Sec. 3.3 provide a simple way to speed up the calculations. Eq. (3.16) can be rewritten into the form used throughout Chapter 2,

$$\left[\mathcal{I} - \mathcal{D}\mathcal{L}^{-1}\mathcal{M}\left(\mathcal{S}+\mathcal{F}\right)\right]\beta = \mathcal{D}\mathcal{L}^{-1}q, \tag{3.17}$$

and solved by applying a GMRES solver to invert the combined operator on the lefthand side of the equation. The combined operator does not need to be formed explicitly, as GMRES only requires the action of the operator. For the solution of this system, the code uses AztecOO, which is another Trilinos package, without preconditioning. Preconditioners such as diffusion synthetic acceleration [53] have been successfully applied to speed up convergence of the scattering and fission sources for various other transport discretizations, but are not investigated here for the MLPG equations.

3.4.2 Eigenvalue outer iterations

The standard operator form of the k-eigenvalue equation [Eq. (2.18b)] for the coefficients β is written

$$\mathcal{L}\beta = \mathcal{M}\left(\mathcal{S} + \frac{1}{k}\mathcal{F}\right)\mathcal{D}\beta, \tag{3.18}$$

where again the operators convert from the coefficients to the physical neutron flux where appropriate. One method of solving this equation is fixed point iteration, which updates the coefficients using a similar process to source iteration,

$$\beta^{\ell+1} = \mathcal{D}\mathcal{L}^{-1}\mathcal{M}\left(\mathcal{S} + \frac{1}{k^{\ell}}\mathcal{F}\right)\beta^{\ell}. \tag{3.19a}$$

After each iteration, the eigenvalue is updated as

$$k^{\ell+1} = \frac{\left\|\mathcal{F}\beta^{\ell+1}\right\|}{\left\|\frac{1}{k}\mathcal{F}\phi^{\ell} - \mathcal{S}\left(\beta^{\ell+1} - \beta^{\ell}\right)\right\|}, \tag{3.19b}$$

where the norm $\|(\cdot)\|$ represents a volume summation or integral [54]. Power iteration, which is more costly per iteration than fixed point iteration but also more stable, is written as

$$\beta^{\ell+1} = \left(\mathcal{I} - \mathcal{D}\mathcal{L}^{-1}\mathcal{M}\mathcal{S}\right)^{-1}\mathcal{D}\mathcal{L}^{-1}\mathcal{M}\frac{1}{k}\mathcal{F}\beta^{\ell}, \tag{3.20a}$$

while the update of the eigenvalue takes the form

$$k^{\ell+1} = \frac{\left\|\mathcal{F}\beta^{\ell+1}\right\|}{\left\|\frac{1}{k^{\ell}}\mathcal{F}\beta^{\ell}\right\|}. \tag{3.20b}$$

The $\left(\mathcal{I} - \mathcal{D}\mathcal{L}^{-1}\mathcal{M}\mathcal{S}\right)^{-1}$ term in the power iteration represents the solution of a steady-state transport problem

$$\left(\mathcal{I} - \mathcal{D}\mathcal{L}^{-1}\mathcal{M}\mathcal{S}\right)\beta = s, \tag{3.21}$$

where s is the given source. Thus, each power iteration requires a full steady-state solution when using power iteration.

Similar to the steady-state equation, the k-eigenvalue problem can be solved more efficiently by using Krylov iterative methods. For a Krylov solver that requires isolation of the eigenvalue (such as Block Krylov-Schur), the Krylov iteration can be written as

$$\left(\mathcal{I} - \mathcal{D}\mathcal{L}^{-1}\mathcal{M}\mathcal{S}\right)^{-1}\mathcal{D}\mathcal{L}^{-1}\mathcal{M}\mathcal{F}\beta = k\beta, \tag{3.22}$$

where again a full steady-state problem must be solved for each eigenvalue iteration. Generalized eigensolvers reduce the cost further by supporting an operator on the eigenvalue side of the equation,

$$\mathcal{D}\mathcal{L}^{-1}\mathcal{M}\mathcal{F}\beta = k\left(\mathcal{I} - \mathcal{D}\mathcal{L}^{-1}\mathcal{M}\mathcal{S}\right)\beta. \tag{3.23}$$

This form of the eigenvalue equation is solved here with unpreconditioned Generalized Davidson using Anasazi, another Trilinos package. This significantly reduces computational cost compared to the other eigensolvers [55].

3.5 Parallelization

The code for the MLPG equations uses OpenMP [56] to implement thread-level parallelization. For the meshless integration technique (Sec. 4.1) the independence of the integrals for each weight function allows for parallel integration. For the background integration technique (Sec. 4.2), the integration is parallelized for the background cells and surfaces. As the basis and weight functions are not exclusive to a single background cell, each thread receives its own copy of the integrals. After the integration is complete for all threads, these integrals are summed together, which produces the full integral of each basis and weight function over the entire problem domain. Due to the adaptive integration technique introduced in Sec. 4.2, the time required to integrate the basis and weight functions over each cell is not constant. As such, the scheduling of the parallel integration is dynamic.

The operators such as $\mathcal{D}$, $\mathcal{M}$, $\mathcal{S}$ and $\mathcal{F}$ are parallelized for the spatial index. For instance, the weighted fission sources for the weight functions w_j are calculated in parallel. Because of the shared-memory model, the overlap of the basis function coefficients between the various weighted fission sources is not an issue.

If directionally-dependent preconditioners (Sec. 3.3.2) are used, their ILUT decomposition is computed in parallel. If the preconditioners are independent of direction, each thread computes and uses a separate copy of the preconditioner. The streaming and collision operator is independent for each group and direction and is parallelized in direction. The GMRES operator that performs the outer iterations does not occupy a large percentage of the computational time and is not parallelized.

For extension to a higher number of processors than one node, the Trilinos linear algebra packages used for the outer and inner transport iterations support fully parallel computations using MPI. The weighting of the solution coefficients to calculate the scattering and fission sources and the $\mathcal{L}^{-1}$ operator are the two places where the code requires communication between different spatial points, and these could be parallelized using MPI for the communication and Trilinos for the parallel linear solution of the equations.

3.6 Performance figures from results

The meshless method trades flexibility in geometric specification for computational cost. The main contributors to higher computational cost are the integration step and the initialization and application of the matrices representing the $\mathcal{L}^{-1}$ operation for each direction and energy group. Unlike mesh-based

methods, which usually have constant material parameters in each cell and use simple polynomial basis functions, the presented method must query the CSG for material properties and uses functions that require many quadrature points to integrate accurately. The number of basis functions at each integration point ranges from 10–30 in 2D to 20–90 in 3D. Each of these basis functions is expensive to calculate in comparison to a polynomial, particularly when using an MLS basis [see Eqs. (2.75) and (2.76)]. The integrals must simultaneously converge over the cross sections and the basis and weight functions, making problems such as the IFBA pincell problem (Sec. 7.1) with discontinuous cross sections and radii that vary over more than an order of magnitude difficult to integrate accurately (Sec. 4.5).

The performance of the code is measured for several of the problems as specified in Chapters 6 and 7. While the base cost of the MLPG method is high, the time and memory required scales approximately linearly with the number of points for any given problem. Table 3.1 shows the results of an empirical fit of the timing and memory data to the number of points. The solve scales near-linearly for the problems considered. The exception to the linear scaling is the initialization time of the native ILUT preconditioners for the $\mathcal{L}^{-1}$ matrices in three dimensions, which scales approximately as $N^{1.5}$ for the number of points N.

Values for the timing and memory requirements for sample problems that have around 5000 points are shown in Table 3.2. Most of the timing values are highly problem-dependent. The VERA pincell cases have similar integration times to the 3D problems due to the geometric complexity and small basis and weight function radii near the fuel boundary. The pincell problems have no leakage and low absorption outside of the IFBA region in the thermal group, which means the boundary source takes many iterations to converge. Adjusting for the number of points, each iteration for the VERA and ICSBEP problems takes approximately the same time.

Storage of the native ILUT preconditioners for each direction and group represents most of the high memory cost. The computational cost of initializing the ILUT preconditioners makes recomputation at each application of the $\mathcal{L}^{-1}$ operator impractical. For details on the memory usage and computational cost of the two preconditioners, see Sec. 3.6.2.

Table 3.1: Empirical scaling of memory and timing with number of points N with native ILUT preconditioner

Scaling parameter		1D	2D	3D
Memory		$\sim N$		
Timing	Integration			
	$\mathcal{L}^{-1}$ initialization			$\sim N^{1.5}$
	Solve		$\sim N^{1.15}$	$\sim N^{1.2}$

Table 3.2: Example performance figures using four processors with standard ILUT preconditioner

Parameter		Slab 1D	VERA 1B	VERA 1E	Kobayashi	ICSBEP
Problem type		Fixed	Eigenvalue	Eigenvalue	Fixed	Eigenvalue
Spatial dimensions		1	2	2	3	3
Num. points		5120	5556	5439	4913	4998
Num. directions		256	256	256	512	512
Num. groups		2	2	2	1	1
Num. integration cells		5000	16384	16384	8000	4160
Num. quadrature points per cell		16	256–1024	256–12544	512	1728
Memory (GB)		1.48	2.79	3.24	4.99	5.01
Number of iterations		16	52	46	19	20
Timing (sec)	Total	37	330	751	427	767
	Integration	1	59	480	232	568
	$\mathcal{L}^{-1}$ initialization	11	29	59	123	116
	Solve	24	241	211	71	83

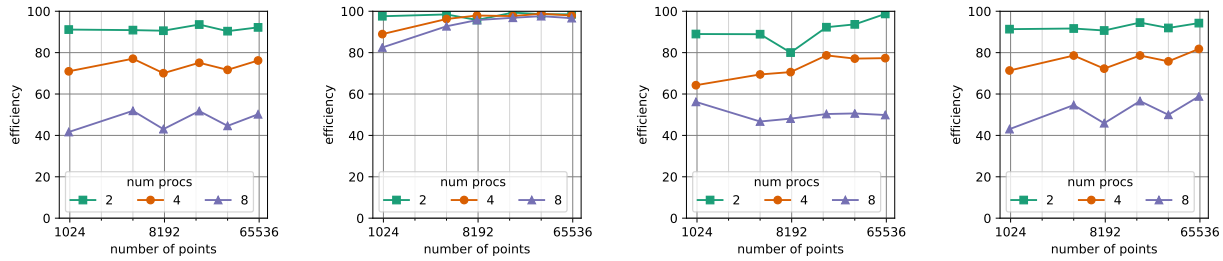
3.6.1 Parallel performance

Due to the shared-memory model of the MLPG code, the number of processors for which the code can be tested is limited. The parallelism does not involve heavy communication between processes for the parallel portions of the code, so good efficiency is expected for these portions of the code. However, there are some sections of the code, including the initialization of the spatial geometry and the outer GMRES solver, are run in serial and could have an impact on parallel performance. The code is not heavily optimized for parallel performance, and so while these results do show good speedup for some cases, near-optimal speedup is not expected.

To test the parallel performance, the Kobayashi problem from Sec. 6.3 with scattering is run in parallel for 1, 2, 4 and 8 processors for between 1000 and 64000 points with the same problem parameters as in Table 6.1, except with 128 directions to decrease the total runtime of the parallel tests. The efficiency of the parallelism is calculated as

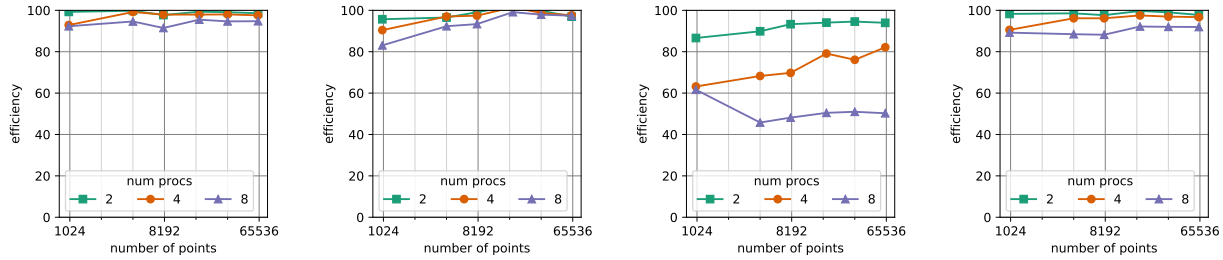
$$E_p = \frac{T_1}{pT_p}, \tag{3.24}$$

where T_p is the execution time of the problem on p processors. The efficiency is calculated for each of the spatial initialization step (which is dominated by the integration of the basis and weight functions), the initialization of the preconditioners and solution of the operator equations, and the total time.



(a) Spatial initialization (b) Preconditioner initialization (c) Solution (d) Total

Figure 3.2: Parallel efficiency for the Kobayashi problem with scattering, linear MLS functions



(a) Spatial initialization (b) Preconditioner initialization (c) Solution (d) Total

Figure 3.3: Parallel efficiency for the Kobayashi problem with scattering, RBF functions

Figure 3.2 shows the parallel efficiency for the simulation using linear MLS functions created from a Wendland 11 basis. The preconditioner initialization has near-optimal efficiency as no communication between the processes is needed. The solution of the problem falls off in efficiency as the number of processors increases, from around 80 percent for 4 processors to 50 percent for 8 processors. This is possibly due to the serial sections of code in the solution step. The integration step of the spatial initialization does not involve heavy communication between processors or long serial sections, so the drop in efficiency signals that the integration step is likely slowed by either shared data access between processors due to the interdependent MLS functions or by the disparity in calculation times for MLS functions with few or many Wendland basis functions.

The same study with the same problem and parameters is done for the Wendland 11 functions without linear MLS normalization. In Fig. 3.3, the preconditioner initialization and solution show similar results, but the spatial initialization efficiency becomes near-optimal, at over 90 percent efficiency for all choices of number of processors and points. The only difference between these two studies is the normalization of the linear MLS functions, which significantly slows parallel performance. It is likely that with optimization the linear MLS functions could be calculated in parallel with similar efficiency. Because the integration step is more efficient, the total parallel efficiency is near-optimal for this problem. For a problem with a scattering

source that is more difficult to converge, the suboptimal solution efficiency would reduce the total efficiency.

3.6.2 Preconditioner performance

The native ILUT preconditioner works well for most problems as long as the radii of the basis and weight functions are not too large. The fill level of the $\mathcal{L}^{-1}$ matrices can be increased to compensate for larger radii, but at the cost of computational performance. From empirical tests, if the problem does not converge for a fill level of 1.0, then it is likely that the point setup itself is too irregular (see Sec. 3.2.1), the SUPG parameter for numerical diffusion is too low (Sec. 3.2.2) or the chosen radii of the basis and weight functions is too large. Generally, the native preconditioner reduces the required number of inner iterations from several hundred to around 4 to 10.

The basis function value preconditioner does not work as reliably well as the native preconditioner and requires many more iterations to converge as it is not a good approximation to the original $\mathcal{L}$ matrix, as discussed in Sec. 3.3.2. For instance, for an IFBA pincell (Sec. 7.1) with 3522 points and 256 directions on 4 processors, the native preconditioner requires 57 seconds to initialize the preconditioner matrices and 349 seconds to solve the full operator equations, while the basis function preconditioner requires 7 seconds to initialize the preconditioner matrices and 3164 seconds to solve the equations (Table 3.3). This appears to be almost entirely due to a larger number of GMRES iterations to solve the $\mathcal{L}^{-1}$ system. However, the basis function preconditioner requires much less memory, at 0.55 GB instead of 3.02 GB. For problems with more directions and energy groups, the difference in memory usage is correspondingly larger, as the basis function preconditioner scales only with the number of points, whereas the native preconditioner scales linearly with the product of the number of points, directions and groups. For this problem, the GMRES solver without a preconditioner does not converge.

There also exist problems for which the basis function preconditioner does not work well. For instance, for the partially-scattering Kobayashi problem (Sec. 6.3) with 64000 points and 512 directions, the $\mathcal{L}^{-1}$ operation with the basis function preconditioner does not converge within 8000 GMRES iterations for some directions. However, for this problem the GMRES method without preconditioning does converge (see Table

Table 3.3: Preconditioner performance for VERA 1E problem with 4 processors, 3522 points and 256 directions

Preconditioner	Num. precs.	Memory	Prec. init.	Solve	Outer	Inner
Basis function	1	0.549 GB	6.6 sec	3164.3 sec	45	475–668
Native	512	3.017 GB	56.7 sec	348.9 sec	45	7–8
None	0	does not converge				

Table 3.4: Preconditioner performance for Kobayashi problem with 8 processors, 64000 points and 512 directions

Preconditioner	Num. precs.	Memory, (basis / full)	Prec. init.	Solve	Outer	Inner
Basis function	1	does not converge				
Native	512	72 / 75 GB	1037.3 sec	251.9 sec	20	6–10
None	0	4 / 7 GB	0.0 sec	2826.7 sec	20	93–154

3.4). On 8 processors, the ILUT method requires 1037 seconds to initialize the preconditioner matrices and 251 seconds to solve the problem. The GMRES method without a preconditioner requires no time to initialize the preconditioner matrices and 2826 seconds to solve the problem. For this problem, then, the total cost of the initialization and solution steps is around twice as high for the solver without preconditioning. This problem requires around 20 applications of the $\mathcal{L}^{-1}$ operator. For a problem that requires many more applications of this operator, such as a problem with high scattering ratios, the fixed cost of initialization of the ILUT matrices and low cost of the solution step for the preconditioned system would be much more efficient.

For the Kobayashi problem, the memory required to perform the simulation with preconditioning is around 75 GB, most of which is the storage of the ILUT preconditioners. This memory cost scales approximately linearly with the product of the number of angles, groups and points. The memory cost of the simulation without preconditioning is around 7 GB, much of which is the storage of the integrated cross sections. For the basis function cross section weighting, the memory usage decreases to around 4 GB. The memory cost scales linearly with the product of the number of groups and points. An empirical equation for the memory in GB for the two solution techniques applied to the Kobayashi problem is

$$M_{\text{no prec}} \approx \begin{cases} 6.3 \times 10^{-5} J, & \text{basis weighting,} \\ 1.1 \times 10^{-4} J, & \text{full weighting,} \end{cases} \tag{3.25}$$

$$M_{\text{prec}} \approx M_{\text{no prec}} + 2.1 \times 10^{-6} JN, \tag{3.26}$$

where J is the number of spatial points and N is the number of directions. The scaling of the preconditioned method with the number of directions makes simultaneous angular and spatial convergence difficult to achieve, but the solution without preconditioning does not converge for many problems, including the VERA problem mentioned earlier in this section.

Section 9.3 discusses the need for more efficient solution methods for the $\mathcal{L}^{-1}$ operation. Better preconditioners would significantly reduce the computational cost and memory requirements for the MLPG method

as presented here.

Chapter 4

Integration methods for meshless transport

Once the connectivity of the basis and weight functions is determined as described in Sec. 3.2, the integration [e.g. in Eqs. (2.32), (2.48), (2.50) and (2.55)] is performed using one of two methods, a meshless integration technique (Sec. 4.1) or a background mesh (Sec. 4.2). The domain of integration for the MLPG discretization is defined by the region of support of the basis and weight functions. For the functions described in Sec. 2.10, the support region of each basis or weight function is a circle in 2D or a sphere in 3D. The region of support for the intersection of a basis and weight function is often a 2D or 3D lens. These geometries are complicated somewhat by the addition of boundaries, which limit the region of support to the physical problem domain.

An integral over the region R can be performed numerically as

$$\int_R f(\boldsymbol{x})\, dV = \sum_i c_i f(\boldsymbol{x}_i), \tag{4.1}$$

where c_i are the integration weights and $\boldsymbol{x}_i$ are the integration ordinates. One method of calculating the integration ordinates and weights is by using a tensor product quadrature. Starting with two integration quadratures for the integrals over x and y (with the ordinates and weights x_i, y_j, c_i^x, c_j^y), the tensor-product quadrature over a 2D Cartesian region with the limits $x \in R_x$ and $y \in R_y$ can be written as

$$\int_{R_y}\int_{R_x} f(\boldsymbol{x})\, dxdy = \sum_{i,j} c_{i,j} f(\boldsymbol{x}_{i,j}), \tag{4.2a}$$

where

$$\boldsymbol{x}_{i,j} = (x_i, y_j), \tag{4.2b}$$

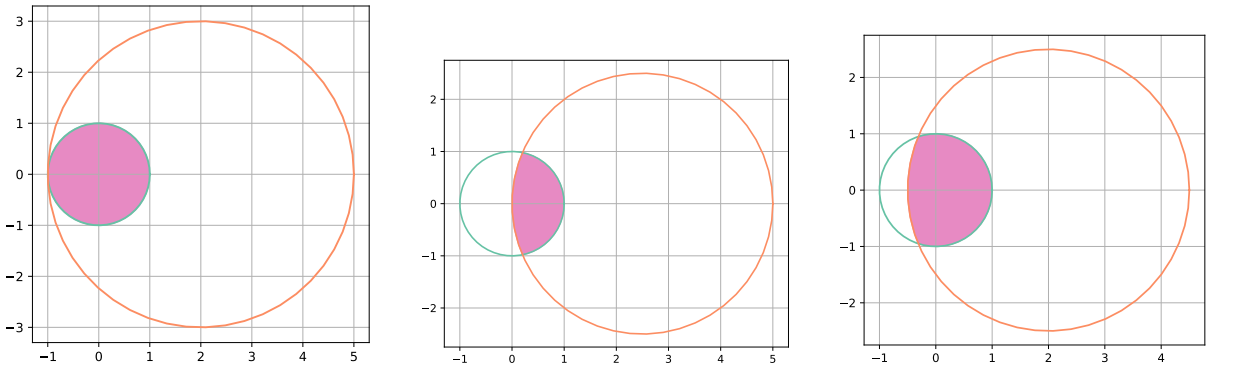
$$c_{i,j} = c_i^x c_j^y. \tag{4.2c}$$

Integrals for the geometries described in Secs. 4.1 and 4.2 can be done in the same way.

4.1 Meshless integration

The first integration method performs the integrals numerically over the region of support of the basis and weight functions. Similar methods are used in Refs. [57, 58, 59]. Other meshless integration methods that involve conversion to boundary integrals [14, 60] would not be directly applicable here, as the use of Green's theorem or other volume-to-boundary conversion techniques requires that the partial derivatives of the integrands be continuous, which is not the case for the discontinuous material properties in many neutron transport applications.

This section focuses specifically on two-dimensional integrals of basis and weight functions with a cylindrical region of support. The integrals of the two-dimensional intersection between a basis and weight function can be simplified to three cases: a standard lens, a non-standard lens and a full cylinder (see Fig. 4.1). For integrals near the boundaries, each of these regions could be intercepted by up to two boundary surfaces at the corners of the problem, which could be in an arbitrary orientation with respect to the basis and weight functions. If the weights for the quadrature points past the boundary were simply set to zero, the quadrature would no longer be as accurate as the function would effectively have a discontinuity in the integrand. Specific integration methods for intersection of lenses with different combinations of boundaries could be derived, but the integrals would quickly become unnecessarily complicated. Instead, for integration near boundary surfaces, the integration region is overlaid by a Cartesian tensor quadrature over the integration region, as shown in Fig. 4.2b. The following sections derive the Cartesian tensor quadrature, cylindrical quadrature and the two cases of lens quadratures.



(a) Cylindrical integration region (b) Standard lens integration region (c) Non-standard lens integration region

Figure 4.1: Meshless integration regions in transformed coordinates

4.1.1 Cartesian regions

Integration quadratures such as the Gauss-Legendre quadrature used here are often given over the range $[-1, 1]$. To calculate the integration ordinates, the geometry is mapped onto simple Cartesian regions. In one dimension, to convert from $x \in [x_1, x_2]$ to $\xi \in [-1, 1]$, the mapping

$$x(\xi) = \frac{1-\xi}{2} x_1 + \frac{1+\xi}{2} x_2 \tag{4.3a}$$

is applied. The differential length for the mapping is then

$$dx = \frac{x_2 - x_1}{2} d\xi. \tag{4.3b}$$

After the change of variables, a one-dimensional integral over the Cartesian region $x \in [x_1, x_2]$ has the form

$$\begin{aligned} \int_{x_1}^{x_2} f(x)\, dx &= \frac{x_2 - x_1}{2} \int_{-1}^{1} f(x(\xi))\, d\xi \\ &= \frac{x_2 - x_1}{2} \sum_i c_i f(x(\xi_i)) \\ &= \sum_i C_i f(X_i), \end{aligned} \tag{4.4a}$$

where ξ_i are the original integration ordinates in $\xi \in [-1, 1]$ and c_i are the original integration weights. The modified integration ordinates and weights are then

$$X_i = x(\xi_i), \tag{4.4b}$$

$$C_i = \frac{x_2 - x_1}{2} c_i. \tag{4.4c}$$

For a Cartesian region in 2D, the conversion from $x \in [x_1, x_2]$ and $y \in [y_1, y_2]$ to $\xi \in [-1, 1]$ and $\eta \in [-1, 1]$ follows the same process, using the mappings

$$x(\xi) = \frac{1-\xi}{2} x_1 + \frac{1+\xi}{2} x_2, \tag{4.5a}$$

$$y(\eta) = \frac{1-\eta}{2} y_1 + \frac{1-\eta}{2} y_2. \tag{4.5b}$$

The differentials are independent due to the orthogonality of the coordinate system,

$$dx = \frac{x_2 - x_1}{2} d\xi, \tag{4.5c}$$

$$dy = \frac{y_2 - y_1}{2} d\eta, \tag{4.5d}$$

and a two-dimensional integral over this region becomes

$$\begin{aligned}\int_{x_1}^{x_2} \int_{y_1}^{y_2} f(x,y)\, dxdy &= \frac{x_2 - x_1}{2} \frac{y_2 - y_1}{2} \int_{-1}^{1} \int_{-1}^{1} f(x(\xi), y(\eta))\, d\xi d\eta \\ &= \frac{x_2 - x_1}{2} \frac{y_2 - y_1}{2} \sum_{i,j} c_i^x c_j^x f(x(\xi_i), y(\eta_j)). \end{aligned} \tag{4.6}$$

The modified integration ordinates and weights can be calculated similarly to those in Eq. (4.4).

4.1.2 Cylinder

In cylindrical coordinates, the integrals are first mapped from cylindrical to Cartesian coordinates, and then finally to a local coordinate system. The mapping from radial coordinates with $r \in [r_1, r_2]$ and $\theta \in [\theta_1, \theta_2]$ to Cartesian coordinates is

$$x(r,\theta) = x_0 + r\cos\theta, \tag{4.7a}$$

$$y(r,\theta) = y_0 + r\sin\theta, \tag{4.7b}$$

where $\boldsymbol{x}_0 = (x_0,\ y_0)$ is the location of the center of the cylinder. The integral is then converted to local coordinates by the transformations

$$\begin{aligned} r &= \frac{1-\xi}{2} r_1 + \frac{1+\xi}{2} r_2, \\ \theta &= \frac{1-\eta}{2} \theta_1 + \frac{1+\eta}{2} \theta_2, \end{aligned}$$

for $\xi \in [-1, 1]$ and $\eta \in [-1, 1]$. As the coordinate system is orthogonal as for the Cartesian case, the differentials are independent,

$$\begin{aligned} dr &= \frac{r_2 - r_1}{2} d\xi, \\ d\theta &= \frac{\theta_2 - \theta_1}{2} d\eta. \end{aligned}$$

The integrals after being converted to local coordinates are

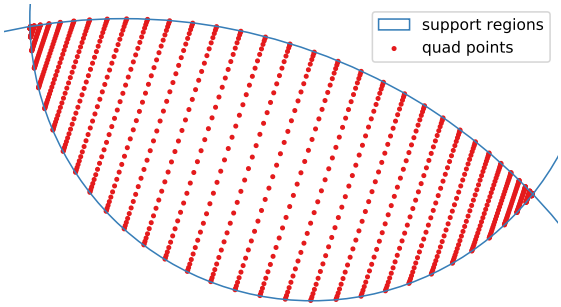
$$\int_{\theta_1}^{\theta_2} \int_{r_1}^{r_2} f(x(r,\theta), y(r,\theta))\, r dr d\theta = \frac{r_2 - r_1}{2} \frac{\theta_2 - \theta_1}{2} \int_{-1}^{1} \int_{-1}^{1} f(x(r(\xi), \theta(\eta)), y(r(\xi), \theta(\eta)))\, r(\xi)\, d\xi d\eta$$

$$= \frac{r_2 - r_1}{2} \frac{\theta_2 - \theta_1}{2} \sum_{i,j} f\left(x\left(r\left(\xi_i\right), \theta\left(\eta_j\right)\right), y\left(r\left(\xi_i\right), \theta\left(\eta_j\right)\right)\right) r\left(\xi_i\right). \quad (4.9)$$

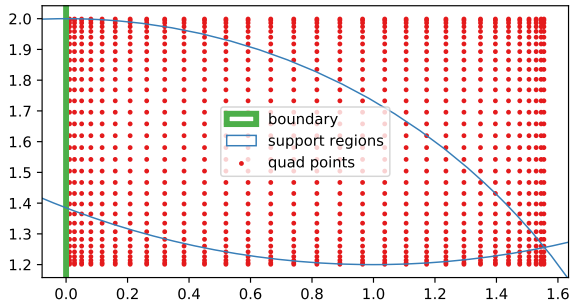
This quadrature region and integration points are shown in Figs. 4.1a and 4.2d, respectively.

4.1.3 Standard lens

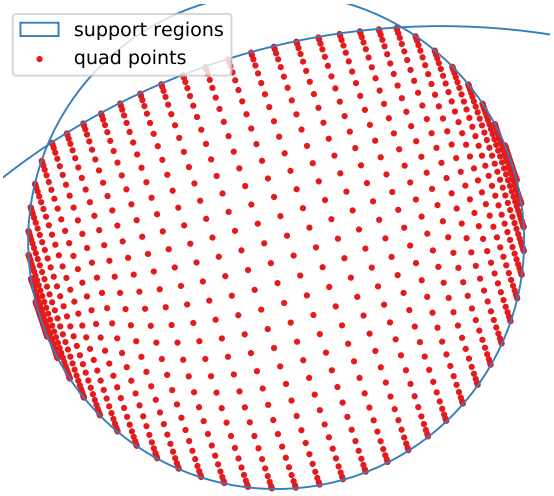
For the simple lens pictured in Figs. 4.1b and 4.2a the two intercepts of the circles are not past the center of either circle. The coordinate system is rotated so both radii are located at the same point on the y axis and then translated so that one of the circles is located at the origin. In this coordinate system, all integrals appear similar to the one pictured in Fig. 4.1b.



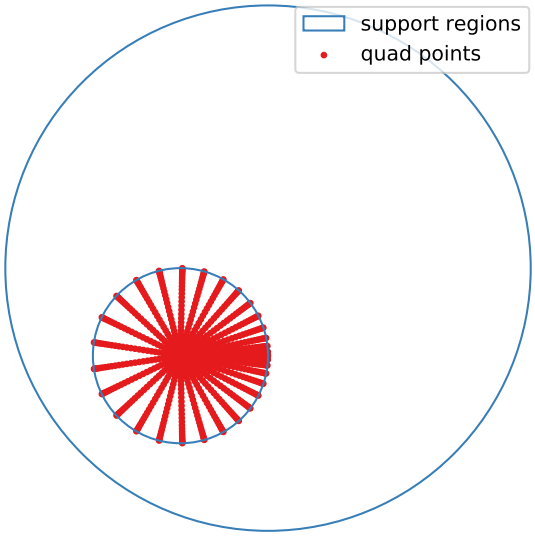
(a) Standard lens integration quadrature



(b) Quadrature near a boundary



(c) Non-standard lens integration region



(d) Cylindrical integration region

Figure 4.2: Quadratures for meshless integration

This transformation from the Cartesian coordinates x, y to simplified coordinates $\bar{x}, \bar{y}$ has the form

$$\begin{bmatrix} \bar{x} \\ \\ \bar{y} \end{bmatrix} = \begin{bmatrix} x_1 \\ \\ y_1 \end{bmatrix} + \frac{1}{d} \begin{bmatrix} \Delta x & -\Delta y \\ \\ \Delta y & \Delta x \end{bmatrix} \begin{bmatrix} x \\ \\ y \end{bmatrix}, \tag{4.10a}$$

with the distances

$$\Delta x = x_2 - x_1, \tag{4.10b}$$

$$\Delta y = y_2 - y_1, \tag{4.10c}$$

$$d = \sqrt{\Delta x^2 + \Delta y^2}. \tag{4.10d}$$

In this coordinate system, the equations for each circular support region can be written

$$\bar{x}^2 + \bar{y}^2 = r_1^2, \tag{4.11a}$$

$$\left(\bar{x} - d\right)^2 + \bar{y}^2 = r_2^2, \tag{4.11b}$$

where the radii of the circles (which remain unchanged in this coordinate system) are r_1 and r_2. The two intercepts of these circles are located at the points $(\bar{x}_{int},\ \bar{y}_{int})$ and $(\bar{x}_{int},\ -\bar{y}_{int})$, with

$$\bar{x}_{int} = \frac{d^2 + r_1^2 - r_2^2}{2d}, \tag{4.12a}$$

$$\bar{y}_{int} = \frac{\sqrt{2d^2\left(r_1^2 + r_2^2\right) - \left(r_1^2 - r_2^2\right)^2 - d^4}}{2d}. \tag{4.12b}$$

The integration limits in the $\bar{x}$ dimension are variable with respect to $\bar{y}$,

$$\bar{x}_{min}\left(\bar{y}\right) = d - \sqrt{r_2^2 - \bar{y}^2}, \tag{4.13a}$$

$$\bar{x}_{max}\left(\bar{y}\right) = \sqrt{r_1^2 - \bar{y}^2}, \tag{4.13b}$$

while the integration limits in the $\bar{y}$ dimension go from $-\bar{y}_{int}$ to $\bar{y}_{int}$.

The conversion of the integral from Cartesian to simplified coordinates has a Jacobian determinant of one,

$$J_{(x,y)\to(\bar{x},\bar{y})} = \begin{vmatrix} \dfrac{dx}{d\bar{x}} & \dfrac{dx}{d\bar{x}} \\ \\ \dfrac{dy}{d\bar{x}} & \dfrac{dy}{d\bar{y}} \end{vmatrix} = 1, \tag{4.14}$$

which means the integral can be transformed to the simplified coordinates as

$$\int_V f(x,y)\,dV = \int_{-\bar{y}_{int}}^{\bar{y}_{int}} \int_{\bar{x}_{min}(\bar{y})}^{\bar{x}_{max}(\bar{y})} f\left(x\left(\bar{x},\bar{y}\right), y\left(\bar{x},\bar{y}\right)\right) d\bar{x}d\bar{y}. \tag{4.15}$$

This integral is transformed a second time from the simplified coordinate system into a standard coordinate system $\xi \in [-1,1]$ and $\eta \in [-1,1]$ using

$$\bar{x} = \frac{\bar{x}_{max}\left(\bar{y}\left(\eta\right)\right) - \bar{x}_{min}\left(\bar{y}\left(\eta\right)\right)}{2}\xi + \frac{\bar{x}_{max}\left(\bar{y}\left(\eta\right)\right) + \bar{x}_{min}\left(\bar{y}\left(\eta\right)\right)}{2}, \tag{4.16a}$$

$$\bar{y} = \bar{y}_{int}\eta. \tag{4.16b}$$

Unlike the first transformation, the Jacobian determinant for this transformation is not equal to a constant,

$$J_{(\bar{x},\bar{y})\to(\xi,\eta)} = \begin{vmatrix} \dfrac{d\bar{x}}{d\xi} & \dfrac{d\bar{x}}{d\eta} \\ \dfrac{d\bar{y}}{d\xi} & \dfrac{d\bar{y}}{d\eta} \end{vmatrix} = \left(\frac{\bar{x}_{max}\left(\bar{y}\right) - \bar{x}_{min}\left(\bar{y}\right)}{2}\right)\bar{y}_{int}. \tag{4.17}$$

This Jacobian is used to map the integral to the local coordinates,

$$\int_V f(x,y)\,dV = \int_{-1}^{1}\int_{-1}^{1} f\left(x\left(\bar{x},\bar{y}\right), y\left(\bar{x},\bar{y}\right)\right)\left(\frac{\bar{x}_{max}\left(\bar{y}\right) - \bar{x}_{min}\left(\bar{y}\right)}{2}\right)\bar{y}_{int}d\xi d\eta. \tag{4.18}$$

For simplicity, the dependencies of $\bar{x}$ and $\bar{y}$ on ξ and η have been suppressed. The conversion of this continuous integral to a numerical quadrature follows the same procedure as in past sections.

4.1.4 Non-standard lens

The second type of lens uses the same coordinate system and intercepts as in Sec. 4.1.3. The difference is that the $\bar{x}$ intercept is past the center of one of the circles, as shown in Fig. 4.1c. Assuming that the first circle with r_1 has the smaller radius ($r_1 < r_2$) and that the intercept is past the center of the first circle, the integration is of the same form as for the standard lens but with $\bar{x}$ limits of

$$\bar{x}_{min}\left(\bar{y}\right) = \begin{cases} d - \sqrt{r_2^2 - \bar{y}^2}, & |\bar{y}| \le \bar{y}_{int}, \\ -\sqrt{r_1^2 - \bar{y}^2}, & \text{otherwise}, \end{cases}$$

$$\bar{x}_{max}\left(\bar{y}\right) = \sqrt{r_1^2 - \bar{y}^2}.$$

If $r_2 > r_1$, then the definitions reverse,

$$\bar{x}_{min}(\bar{y}) = d - \sqrt{r_2^2 - \bar{y}^2}$$

$$\bar{x}_{max}(\bar{y}) = \begin{cases} \sqrt{r_1^2 - \bar{y}^2}, & |\bar{y}| \leq \bar{y}_{int}, \\ d + \sqrt{r_2^2 - \bar{y}^2}, & \text{otherwise}, \end{cases}.$$

The integral can be calculated using Eq. (4.18) with these $\bar{x}$ limits and the same $\bar{y} \in (-\bar{y}_{int},\ \bar{y}_{int})$ limits as before.

4.1.5 Methodology

For the integrals only involving weight functions and not basis functions, as in Eq. (2.46), the region of integration is cylindrical except near boundaries, where the Cartesian quadrature is used instead. For integrals involving both basis and weight functions, one of the lens quadratures or the cylindrical quadrature are used away from boundaries and the Cartesian quadrature is used for lens regions that intersect boundaries. One benefit of the meshless integration approach is that unlike the background mesh integration in Sec. 4.2, the meshless integration is independent for each weight function. This means that the integration of weight functions can be done in parallel with no special considerations for shared data.

Due to the simpler parallelization and independent nature of the meshless integration, the implementation is in some ways simpler than the background mesh integration. The issue with the meshless integration is that much of the computational cost of the integration is evaluation of the weight and basis functions, which is particularly expensive for the MLS functions (Sec. 2.10.2). The basis functions each intersect with many weight functions, some of which have support regions that overlap. This means that the quadrature sets for the basis and weight functions overlap many times over. In addition, the weight function values must be calculated for each individual basis function integral separately as the quadrature points differ. Some of this cost could be defrayed by doing the integrals of the basis functions over the entire weight function region, but this would reduce the accuracy of the quadrature.

4.2 Background mesh integration

In contrast to the meshless integration method, the background integration mesh method uses non-overlapping quadratures to increase integration efficiency. The integration regions no longer conform to the exact integration domains as in the meshless method, but as the cost of the method is lower per area integrated, more quadrature points can be used to compensate. Because the basis and weight functions chosen in Sec. 2.10 go to zero at the edge of their support region, there are no discontinuities in the functions that

Algorithm 4.4 Procedure for integration using a background mesh

```
initialize all integrals to zero
for each background cell or face (region i) do
    create a quadrature that integrates region i
    find the basis and weight functions that intersect with region i
    for each quadrature point j do
        evaluate each basis and weight function and their derivatives at point j
        find the values of the cross sections and sources at point j
        for each required integral do
            add the contribution of this quadrature point to the global integral
        end for
    end for
end for
```

need special treatment when using a background integration mesh. The cross sections, however, do add discontinuities that require consideration.

To perform the integration using a background mesh, the problem is overlaid with a Cartesian mesh to create cells for the volume integration and faces along the boundaries for the surface integration. This background mesh is agnostic to the problem geometry and is only used for the integration step, not for the solution of the transport equations. Algorithm 4.4 describes the background integration procedure. In summary, the integration is performed by choosing a quadrature for each integration cell, evaluating all needed data at each quadrature point and then adding the contribution of each quadrature point to the global integral. The quadrature inside of each Cartesian cell is an outer product Gaussian quadrature, e.g. for a function $f(x, y, z)$,

$$\int_{\text{cell}} f(x, y, z)\, dV = \sum_{i,j,k} c_i^x c_j^y c_k^z f(x_i, y_j, z_k), \tag{4.19}$$

where the one-dimensional quadrature integration ordinates are x_i, y_i and z_k with corresponding weights of c_i^x, c_j^y and c_k^z. This integral is done using the same transformations as the Cartesian quadrature in Eq. (4.4) applied to the 1D, 2D or 3D volume integral.

The integration quadrature for each cell can be adaptively chosen for problems in which radii of the intersecting functions differ significantly. For the adaptive quadrature, a minimum number of integration ordinates across the basis or weight function N_{rad} is specified. The number of integration ordinates for the cell along each dimension is chosen using the basis or weight function with the smallest radius r inside the cell as

$$N_d = \frac{\ell_d}{r} N_{rad}, \tag{4.20}$$

where ℓ_d is the length of the cell in that dimension, for instance the side length of a Cartesian cell, which defines an outer product quadrature with $\prod_d N_d$ total ordinates.

The integration is done independently for each cell and face, which allows for parallelization of the

integration step (see Sec. 3.5). Each processor needs access to the basis and weight function data for each integration cell assigned to it. Once the integration has been performed for the cells assigned to each processor, the integrals are reduced to sum the contributions of each integration cell to the overall integrals of the basis and weight functions.

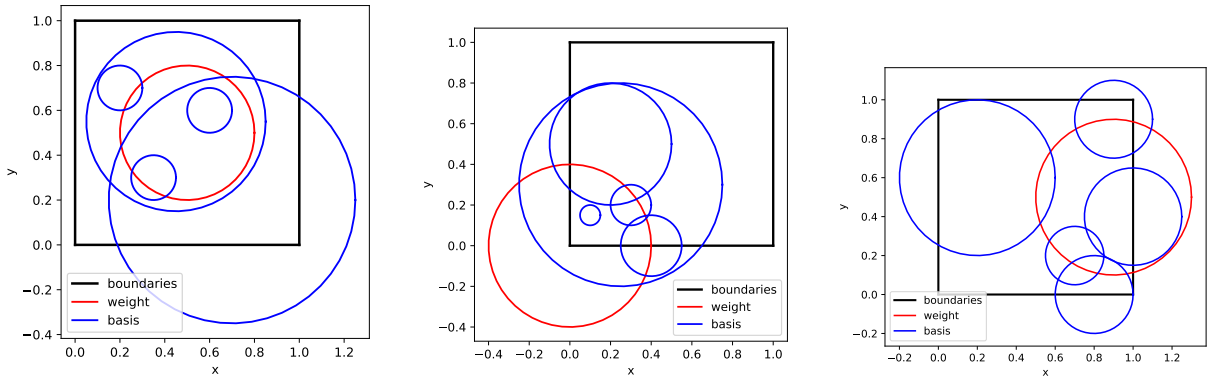
4.3 Integration verification against benchmark problem

To verify the integration methods, three configurations of basis and weight function intersections (including those described in Sec. 4.1) are considered and the integrals are performed individually to high precision using Mathematica [61]. The positions of the basis and weight functions for each case are shown in Fig. 4.3. The basis and weight functions use the Wendland 11 RBF. The integrals $\bar{i}^V_{bw;i,j}$ and $\overline{\boldsymbol{I}}_{\nabla b \nabla w;i,j}$ from Eq. (2.43) are compared to the benchmark values by calculating the total L_2 error,

$$\epsilon_{L_2} = \frac{1}{N_{\text{weight}}} \sum_j \sqrt{\frac{\sum_i \left(\bar{i}_{bw;i,j} - \bar{i}^{\text{benchmark}}_{bw;i,j} \right)^2}{\sum_i \left(\bar{i}^{\text{benchmark}}_{bw;i,j} \right)^2}}. \tag{4.21}$$

For the derivative integrals, the values summed inside the square root term include the derivatives.

For the meshless integration method, between 4^2 and 2048^2 quadrature points are used to integrate each basis and weight function pair. For the background integration method, quadratures with the same range of values are used in each of the 10^2 background cells. This results in a higher number of points inside each integration region for the background mesh, which is balanced by the increased accuracy of a specialized quadrature for the meshless integration method. Due to the small number of weight and basis functions, the two quadratures aren't directly compared in terms of computational cost in this section.



(a) First configuration (b) Second configuration (c) Third configuration

Figure 4.3: Basis and weight function configurations for verification of numerical integration against benchmark

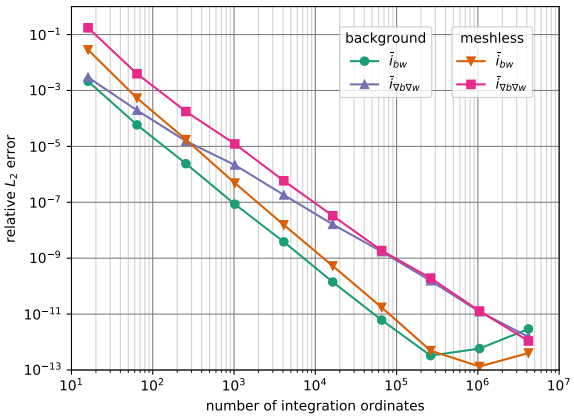


Figure 4.4: Convergence for the benchmark integration problem for meshless and background mesh integration

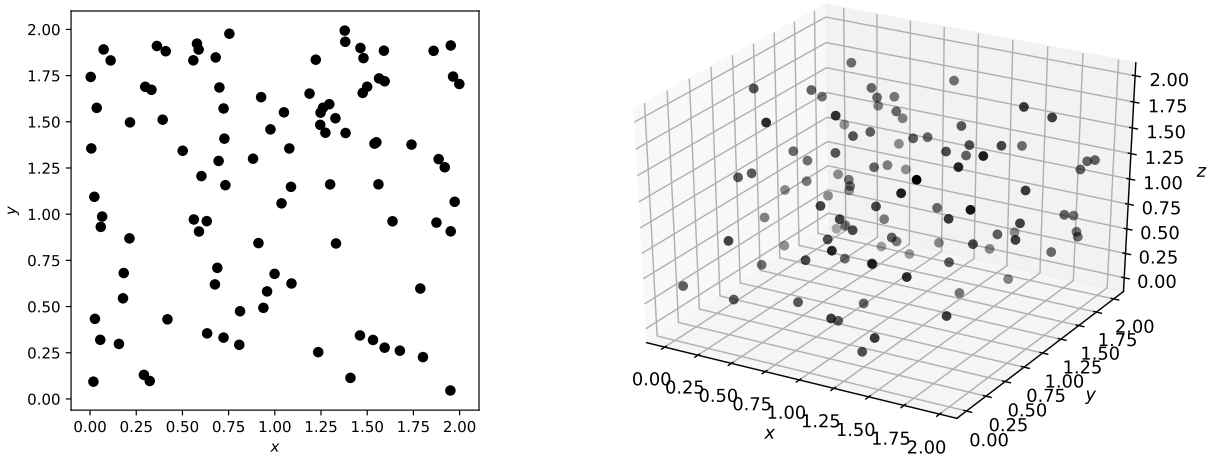
The convergence results for the two integration methods are shown in Fig. 4.4. Both integration methods show approximately fourth-order convergence as the number of integration points in one of the dimensions increases. Once the number of integration ordinates reaches about 1024^2 for either method, the relative L_2 error for the $\bar{i}_{bw}$ integrals is below 10^{-12}. For problems with more integration ordinates, the error increases again, likely because of numerical roundoff. The derivative integrals $\bar{i}_{\nabla b \nabla w}$ converge at around the same rate and by 2048^2 ordinates, the results from both methods have a relative error below 10^{-12}.

4.4 Comparison of integration methods

For comparison of the basis and weight function integration methods, a set of 100 compact Gaussian functions [Eq. (2.65)] with a dimensionless cutoff distance of $R = 5$ are created in 1D, 2D and 3D. The centers of these functions are placed at random in the problem domain, $\boldsymbol{x} \in [-2, 2]$. Figure 4.5 shows the point configurations for the 2D and 3D problems. The randomized points result in some clusters of points and some points without close neighbors. This means that the calculated radii vary over more than an order of magnitude, similar to a realistic problem with small features that need to be resolved.

For the results with a background mesh, 100 (1D and 2D) or 125 (3D) background cells are placed with even spacing. The number of integration ordinates in a cell for the adaptive method is initially set to 8^d (for the dimension d) before potentially being modified according to Eq. (4.20). The number of ordinates for the background mesh without adaptive integration refers to the number in each integration cell, which multiplied by the number of cells is the total number of integration ordinates. For the meshless integration, the number of integration ordinates is fixed for each integral.

A benchmark solution is calculated using the adaptive integration method with a high number of integration ordinates (4096 in 1D, 1024^2 in 2D and 128^3 in 3D) and the result for the other methods is compared



(a) 2D

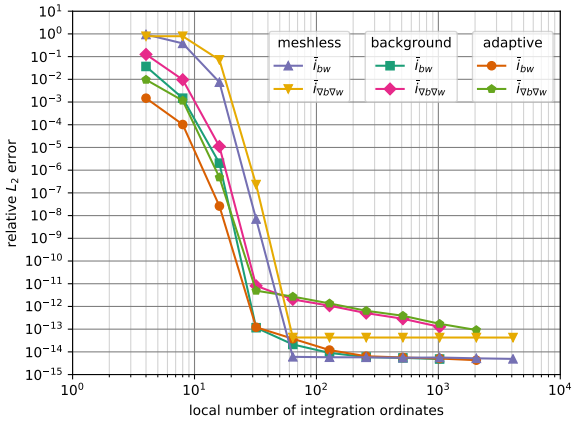
(b) 3D

Figure 4.5: Randomized positions of centers for comparison of integration methods

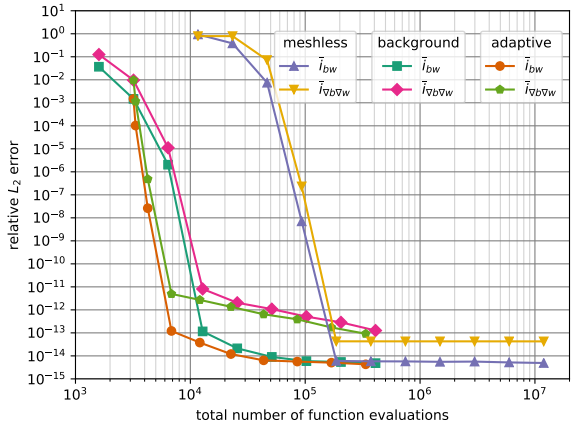
to this solution. The same L_2 error as in Eq. (4.21) is used to find the errors in the integrals of the basis and weight functions and their derivatives with respect to the benchmark solution. To compare the relative performance of the methods, the error is presented as a function of the local number of integration ordinates and separately as a function of the total number of basis and weight function evaluations, which represents a large part of the integration cost.

The results for the L_2 error in 1D, 2D and 3D are shown in Fig. 4.6. All three methods show at least eighth-order convergence with the integration ordinate spacing. The convergence rate is highest for the adaptive case. Both the adaptive and the meshless integration methods scale the quadrature to the size of the radii, which improves the convergence rate. In 1D, the relative error converges to 10^{-13} for both the integrals and the derivatives. In 2D and 3D, the relative error converges to 10^{-11} and 10^{-10}, respectively. The adaptive integration method achieves the best results for a given number of function evaluations. The meshless integration requires around an order of magnitude more function evaluations to achieve the same error as the background mesh methods. Due to the simple Wendland 11 functions used and the additional complexity of the background integration, the computational time is similar in this problem for a given error for the meshless and background integration methods.

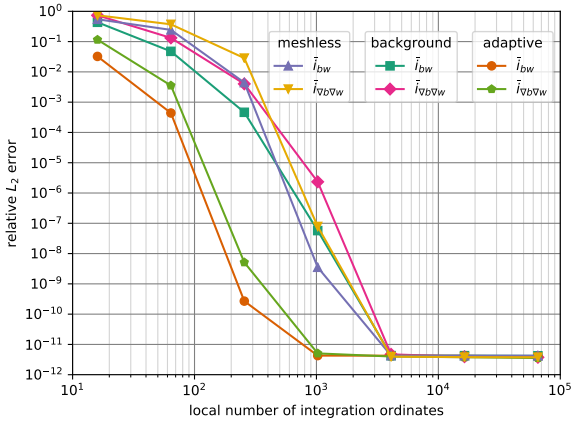
While the performance of the meshless integration in this problem is not significantly worse than the background integration, that is not the case for a problem with MLS functions. The compact Gaussian functions are inexpensive to evaluate, whereas the MLS functions are not. As all the basis and weight functions are evaluated at the same quadrature points, all the values at a single quadrature point as described in Sec. 2.10.2 can be calculated simultaneously. For the meshless method, the weight and basis function



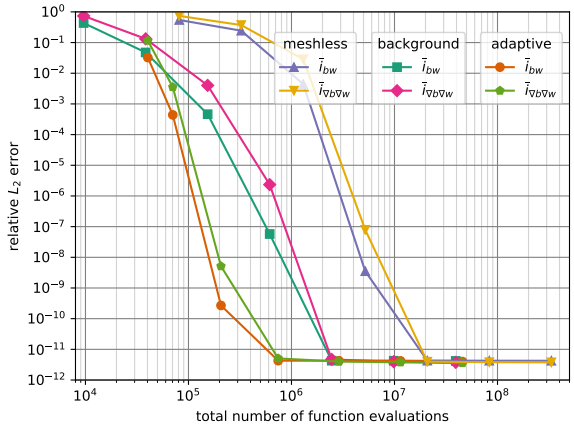
(a) Convergence with local number of integration ordinates, 1D



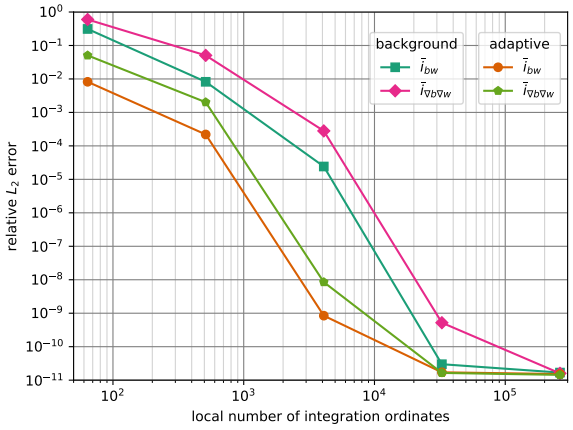
(b) Convergence with total number of function evaluations, 1D



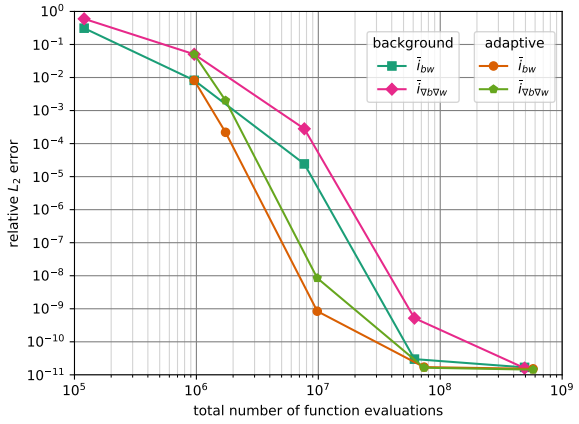
(c) Convergence with local number of integration ordinates, 2D



(d) Convergence with total number of function evaluations, 2D



(e) Convergence with local number of integration ordinates, 3D



(f) Convergence with total number of function evaluations, 3D

Figure 4.6: Comparison of integration methods, convergence to benchmark solution

do not share a common quadrature, which multiplies the cost of integration for MLS functions by 20 to 70 times, depending on the dimension of the problem and the radii of the basis and weight functions. As such, applying the meshless integration method to MLS functions for a large number of integration ordinates is not practical and the meshless integration is not used for the results in the following chapters. In addition, the benefits of a quadrature that exactly conforms to the integration domain are not as important in problems with large discontinuities, which are best integrated by a quadrature with a large number of evaluation points as opposed to a quadrature that very accurately integrates a certain class of smooth equations.

4.5 VERA pincell integration problem

This final integration test investigates the use of MLS functions functions in a heterogeneous medium. For reasons explained at the end of Sec. 4.4, only the background integration mesh methods are applied here. The geometry for the heterogeneous test is a small section of the VERA 1E geometry from Sec. 7.1. As shown in Fig. 4.7, the section under consideration ($0.282 \leq \boldsymbol{x} \leq 0.354$) is near the fuel boundary and includes all five materials from the original VERA problem. The cross sections vary over several orders of magnitude, which makes the material integration particularly difficult. The points are preferentially placed near the IFBA region where the discontinuities in the cross sections are largest and the gradient in the transport solution is largest. For reference, this is a subset of the 34085-point set used for the VERA 1E problem (Sec. 7.1) with additional points added around the boundaries. Additional points are added along the boundaries to ensure coverage of the functions throughout the problem. This is essential for the calculation of the MLS functions, which requires that several functions have nonzero values at every point in the domain (see Sec.

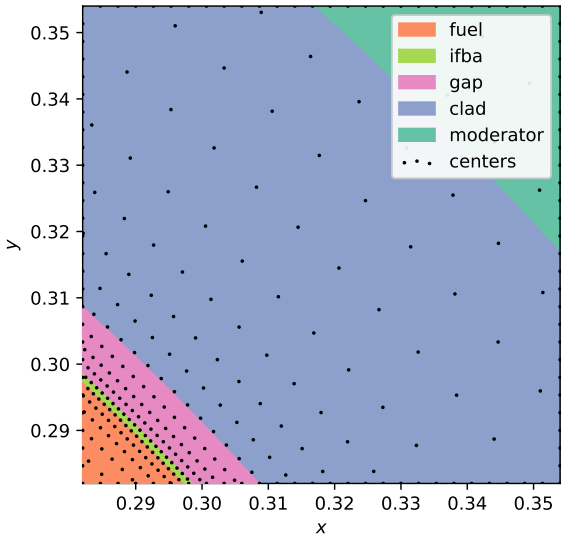


Figure 4.7: VERA pincell integration geometry

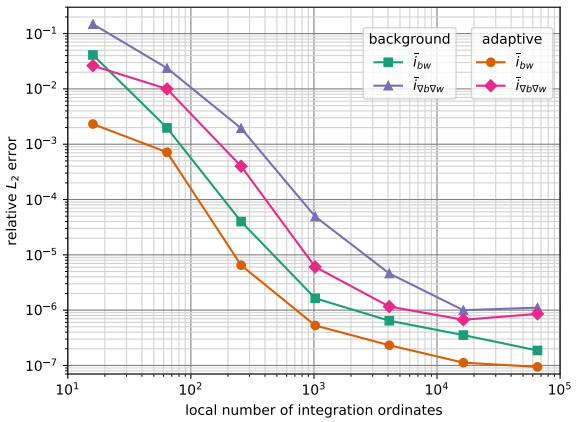
2.10.2).

The radii of the basis and weight functions are calculated using Alg. 3.3 with eight neighbors. The radii range between 0.0015 near the IFBA region to 0.015 near the clad-moderator boundary. These disparate radii make the integration difficult, as a Cartesian background mesh with a constant number of integration points includes more integration points for basis and weight functions with larger radii. When using a background mesh without an adaptive number of integration ordinates, the same number of ordinates per unit area are used for both the functions with large radii and the functions with small radii. When the enough integration points have been added to resolve the integrals of the smallest functions, the integrals of the largest functions may have more points than is required for accurate integration.

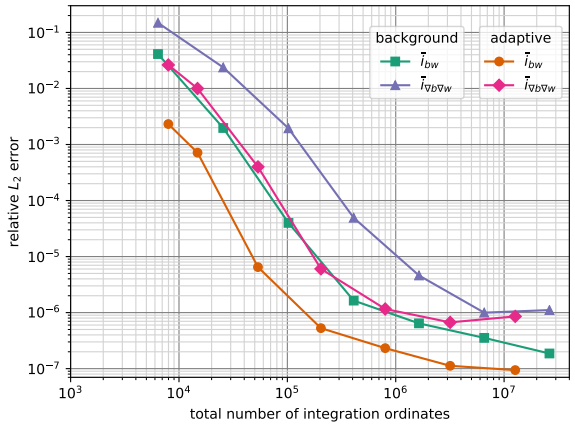
The subdomain of the VERA 1E problem considered here has a length 17.5 times smaller than the length of the full problem, and an area of 306.25 times smaller. The goal of this integration study is not to simply get a good integration result, but to get a good integration result in a timely manner. The number of background cells chosen is 20^2, which would be equivalent to a background mesh with 350^2 cells for the full pincell problem. This results in $4N$ total ordinates across the radii of the largest functions and $0.4N$ ordinates across the radii of the smallest functions, where N^2 is the number of integration ordinates in the Cartesian product quadrature. The number of ordinates is varied from 4^2 to 256^2 for the background integration without an adaptive quadrature. For the adaptive quadrature, the number of ordinates required across each weight function radius [Eq. (4.20)] is varied between 4 and 256, with a baseline number of integration ordinates in a cell of 4^2. The benchmark calculation is performed using 512^2 as the baseline number of integration ordinates and 512 as the minimum number of integration ordinates across each radius.

Figure 4.8 shows the error of the integrals with an increasing number of integration ordinates. For the number of integration points considered, the basis and weight function integrals converge to 10^{-7} relative error and the basis and weight derivative integrals converge to 10^{-6} relative error. To achieve a similar error, the adaptive integration technique requires between 2 and 10 times fewer global integration points than the method without adaptive integration.

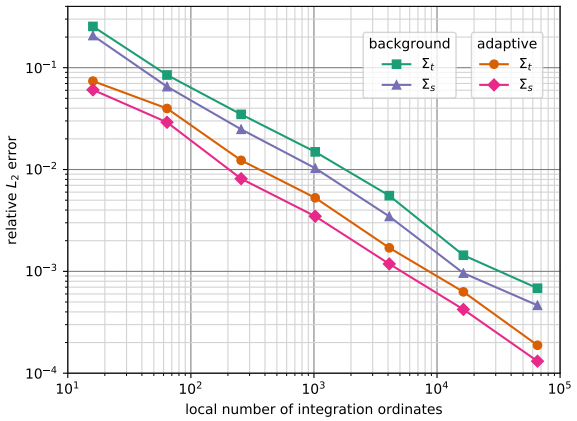
The integrals of the basis and weight functions over the total and scattering cross sections [Eqs. (2.48) and (2.51)] converge less quickly than the MLS function integrals due to the discontinuities in the integration domain. The chosen discretization has a higher density of basis and weight functions near the largest of the discontinuities. The adaptive integration puts more integration points near these functions due to their small radii, which also increases the accuracy of integration for the discontinuities. Compared to the background mesh integration without adaptivity, the adaptive method again requires far fewer global integration ordinates (an order of magnitude for the cases with the most ordinates) to achieve similar errors for the cross section integrals and has a higher rate of convergence to the benchmark result. For the highest number of integration



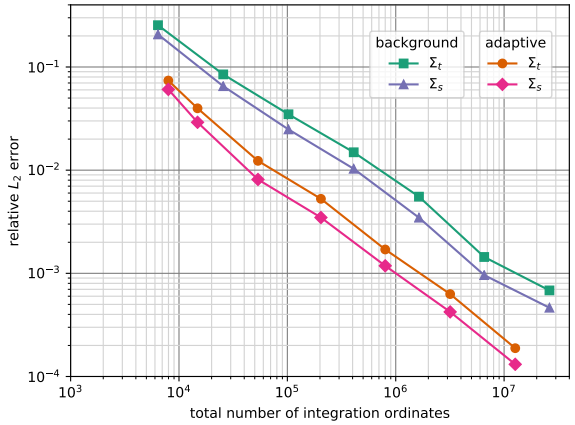
(a) Basis and weight function integrals by local number of integration ordinates



(b) Basis and weight function integral L_2 error by global number of integration ordinates

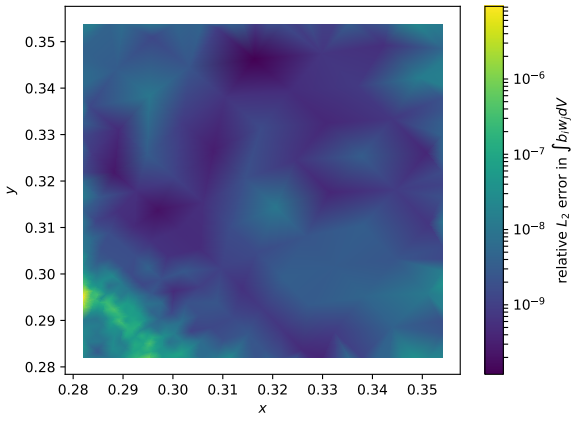


(c) Cross section integral L_2 error by local number of integration ordinates

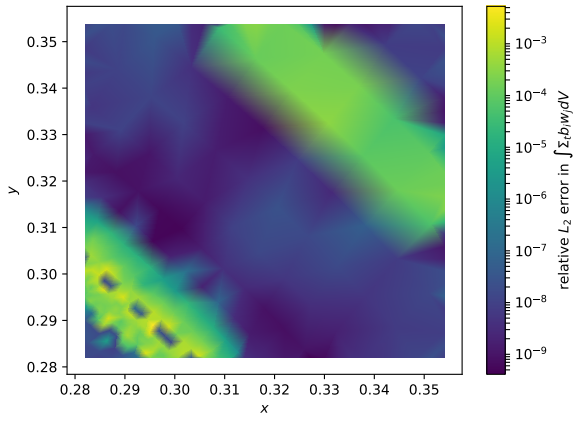


(d) Cross section integral L_2 error by global number of integration ordinates

Figure 4.8: Convergence of VERA pincell integration to benchmark solution



(a) Relative error in the integral $\int_V b_i w_j dV$



(b) Relative error in the integral $\int_V \Sigma_t b_i w_j dV$

Figure 4.9: VERA pincell integration error for adaptive problem with 256^2 local integration ordinates

ordinates considered, the relative L_2 error of the adaptive integration is around 10^{-4} for both the total and cross section integrals.

Figure 4.9 shows the relative L_2 error for the adaptive integration with 256 ordinates as a function of position for the basis and weight function integrals and the cross section integrals. The basis and weight functions with the highest relative integration error are those with the smallest radii. The cross section integrals error is effectively identical to the basis and weight function integral error in regions with constant cross section values. The integration is least accurate near the fuel boundary, where there are three materials within 10 μm. The error also increases near the material discontinuity at the clad-moderator boundary.

One large difference between the Gaussian functions integrated in Sec. 4.4 and the linear MLS functions integrated here is that the former have a simple polynomial series expansion, whereas the latter do not. The Gauss-Legendre tensor product quadrature is designed to integrate polynomials with high accuracy and as such works well for functions such as the Gaussian and the Wendland functions that can be represented as polynomials. The integrals of linear MLS functions require more integration points than either of these to be accurate.

The MLS functions have good properties for solving the transport equation, including partition of unity and the ability to turn a relatively sparse set of RBFs into a basis that can accurately represent polynomials, but they are costly to evaluate and difficult to integrate accurately. The background mesh with an adaptive quadrature provides predictable results for the integration of MLS functions of disparate sizes at a reasonable level of computational cost.

Chapter 5

Verification by the method of manufactured solutions

In this section, the method of manufactured solutions (MMS) is used to verify that the MLPG equations are accurately solved using the discretizations in Chapter 2. This includes derivation of the method of manufactured solutions for the transport equations (Sec. 5.1); presentation of the two test problems (Sec. 5.2); and optimization of the solution for the radii of the basis and weight functions and the SUPG parameters (Sec. 5.3).

5.1 The method of manufactured solutions

The process of verification by the MMS begins with a manufactured solution to the equation, which for the MLPG equations derived in Chapter 2 is the moments of the angular flux, $\phi_{\ell,g}^{m}(\boldsymbol{x})$. The manufactured solution, which is denoted here by $\Phi_{\ell,g}^{m}(\boldsymbol{x})$, is inserted into the original equation to analytically calculate the moments of the internal source, $Q_{\ell,g}^{m}(\boldsymbol{x})$, and the boundary source, $\psi_{n,g}^{inc}(\boldsymbol{x})$. A simulation is performed using these sources to calculate a numerical solution to the problem. If the code is working correctly, the numerical solution matches the manufactured solution.

To calculate the expected internal source for the transport equation, the internal source is expanded using spherical harmonics [as in Eq. (2.44)]. The spherical harmonics moments are orthogonal, meaning that

$$\int_{4\pi} Y_{\ell}^{m}(\boldsymbol{\Omega}) Y_{\ell'}^{m'}(\boldsymbol{\Omega}) \, d\Omega = \frac{4\pi}{2\ell+1} \delta_{\ell,\ell'} \delta_{m,m'}, \tag{5.1}$$

where $\delta_{a,b}$ is the Kronecker delta function from Eq. (2.20). For example, multiplying the internal source term in the transport equation by $Y_{\ell'}^{m'}$ and integrating over the unit sphere results in the equation

$$\int_{4\pi} Y_{\ell'}^{m'} \left[\sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y_{\ell}^{m} Q_{\ell,g}^{m} \right] d\Omega = \begin{cases} Q_{\ell,g}, & \ell' = \ell' \text{ and } m' = m', \\ 0, & \text{otherwise}, \end{cases} \tag{5.2}$$

which isolates the moments of the internal source. To calculate the manufactured internal source, the manufactured flux is inserted into the multigroup transport equation [Eq. (2.11a)] with the steady-state ap-

proximation. This equation with the known manufactured solution $\Phi^m_{\ell,g}$ is multiplied by $Y^{m'}_{\ell'}$ and integrated over all directions. Using the orthogonality property, the resulting transport equation can be solved for $Q_{\ell,g}$,

$$Q^m_{\ell,g} = \int_{4\pi} Y^m_\ell \Bigg[\boldsymbol{\Omega} \cdot \boldsymbol{\nabla} \Psi_g + \Sigma_{t;g} \Psi_g \\ - \sum_{\ell'=0}^{\infty} \frac{2\ell'+1}{4\pi} \sum_{m'=-\ell}^{\ell} Y^{m'}_{\ell'} \sum_{g'=1}^{G} \Sigma_{s;\ell',g'\to g} \Phi^{m'}_{\ell',g'} - \frac{\chi_g}{4\pi} \sum_{g'=1}^{G} \nu_{g'} \Sigma_{f;g'} \Phi^0_{0,g'} d\Omega \Bigg]. \tag{5.3}$$

Replacing the manufactured angular flux Ψ_g by a spherical harmonics expansion in terms of $\Phi^m_{\ell,g}$ and using orthogonality to perform the integration for the scattering and fission terms, the equation simplifies to

$$Q_{\ell,g} = \sum_{\ell'=0}^{\infty} \sum_{m'=-\ell}^{\ell} \left(\frac{2\ell'+1}{4\pi} \int_{4\pi} \boldsymbol{\Omega} Y^m_\ell Y^{m'}_{\ell'} d\Omega \right) \cdot \boldsymbol{\nabla} \Phi^{m'}_{\ell',g} + \Sigma_{t;g} \Phi^m_{\ell,g} \\ - \sum_{g'=1}^{G} \Sigma_{s;\ell,g'\to g} \Phi^m_{\ell,g'} - \delta_{\ell,0} \chi_g \sum_{g'=1}^{G} \nu_{g'} \Sigma_{f;g'\to g} \Phi^m_{\ell,g'}. \tag{5.4}$$

Due to the $\boldsymbol{\Omega}$ term inside the remaining angular integral, if the manufactured solution has spherical harmonics moments of maximum degree L, the calculated source $Q^m_{\ell,g}$ has nonzero values of one spherical harmonic degree higher, or $L+1$. This integral, which is of the form

$$\frac{2\ell'+1}{4\pi} \int_{4\pi} Y^{m''}_1 Y^{m'}_{\ell'} Y^m_\ell d\Omega \tag{5.5}$$

for $m'' = -1, 0, 1$, can be performed analytically using Clebsch-Gordon coefficients. For simplicity, this integral is performed numerically for the problems in Sec. 5.2 using a Gauss-Legendre quadrature with 256 ordinates in 1D or an LDFE quadrature with 16384 or 32768 ordinates in 2D or 3D, respectively.

The boundary source for the manufactured equations is set to the value of the desired source, or

$$\psi^{inc}_{n,g} = \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} Y^m_\ell \left(\boldsymbol{\Omega}_n \right) \Phi^m_{\ell,g}, \tag{5.6}$$

and reflection is disabled. If the boundary integration is done correctly, the value at the boundary is equal the desired solution.

The process of testing the manufactured solution includes:

1. Defining the problem geometry, including cross sections;
2. Initializing the internal source as defined by Eq. (5.4) and the boundary source defined by Eq. (5.6);
3. Performing the integration (for the weak form) or evaluation (for the strong form) of the internal source

and the boundary source using these cross sections and sources;

4. Solving the steady-state transport equation; and

5. Calculating the error of the numerical solution with respect to the initial manufactured solution.

The MMS allows benchmarking of problems that would be difficult to solve using established codes. For instance, in Sec. 5.2, the transport equation is solved with continuously-variable cross sections, which as discussed in Sec. 1.2.2 is not a common capability for a neutron transport code. Another benefit of the MMS is that it removes errors (in deterministic calculations) or uncertainties (in Monte Carlo calculations) from the benchmark solution.

5.2 Introduction to continuous and discontinuous manufactured problems

The two manufactured problems considered here are a problem with piecewise constant cross sections and a problem with continuous, sinusoidal cross sections. The manufactured solutions and cross sections are defined in three dimensions. For two dimensions, the solutions and cross sections are evaluated at $z = 0$. For both problems, the basis and weight function centers are placed on the nodes of an equally-spaced Cartesian grid with an equal number of points in each dimension, $N_x = N_y = N_z$. The problems use 256 and 512 angular quadrature ordinates in 2D and 3D, respectively. The number of points, the number of neighbors in the radius calculation, the Wendland function [Eqs. (2.69)] and the value of the SUPG parameter τ are all varied to find optimal values that converge appropriately and retain good conditioning for the $\mathcal{L}^{-1}$ operation. For each permutation of parameters, the flux $\phi^m_{\ell,g}$ is calculated and compared to the initial manufactured flux, $\Phi^m_{\ell,g}$. The relative L_1 integral error over the entire problem domain V,

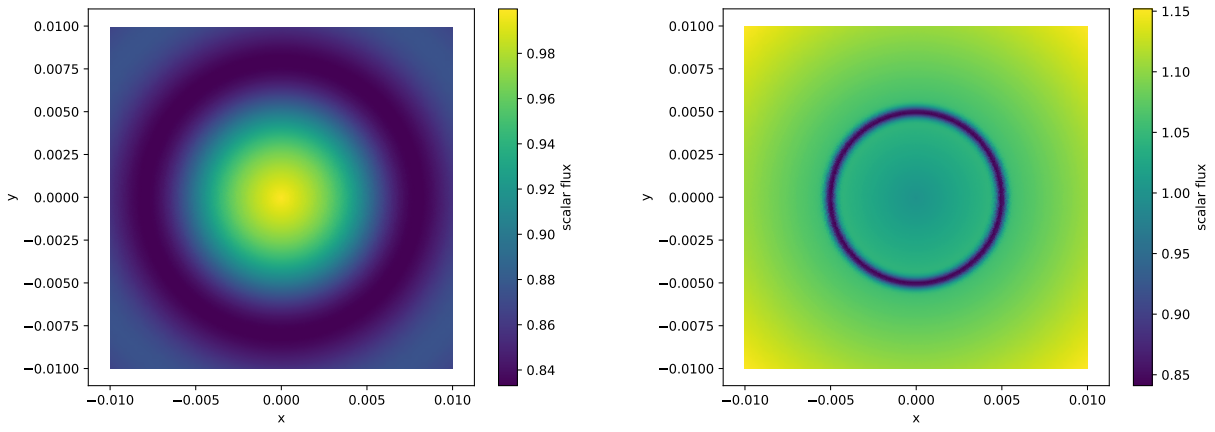
$$\left(L_1 \text{ integral error}\right)_g = \frac{\int_V \left|\phi^0_{0,g} - \Phi^0_{0,g}\right| dV}{\int_V \Phi^0_{0,g} dV}, \tag{5.7}$$

is then calculated numerically using a Gauss-Legendre outer product quadrature to measure convergence and compare between parameters.

The first problem, which has two energy groups and a domain of $-0.01 \leq x, y, z \leq 0.01$, uses the cross sections in Table 7.6 from the VERA 1E problem as described in Sec. 7.1. The cross sections are piecewise constant and depend only on the distance from the origin, $r = \|\boldsymbol{x}\|$, making the cross sections cylindrically symmetric in two dimensions and spherically symmetric in three dimensions. The three cross sections used include the fuel cross sections from $0.0 \leq r \leq 0.0045$, the IFBA (integral fuel burnable absorber) cross sections from $0.0045 < x \leq 0.0055$ and the gap cross sections for $0.0055 < r$. These cross sections represent a fissionable isotope, a strong thermal absorber and a near-vacuum. The solution to this problem is also

Table 5.1: Manufactured solution coefficients for radially symmetric problem [see Eq. (5.8)]

Group	$f_{0,g}^0$	$f_{1,g}^{-1}$	$f_{1,g}^0$	$f_{1,g}^1$	a_g	b_g	c_g	d_g
1	1.0	0.0	0.1	-0.05	-10.0	-0.1	-10^5	-0.0075
2	2.0	0.0	-0.2	0.1	10.0	-0.2	10^7	-0.005



(a) Fast energy group

(b) Thermal energy group

Figure 5.1: Solution to 2D manufactured radial problem

chosen to have cylindrical or spherical symmetry,

$$\Phi_{\ell,g}^m(\boldsymbol{x}) = f_{\ell,g}^m \exp\left(a_g \|\boldsymbol{x}\|\right)\left[1 + b_g \exp\left(c_g\left[\|\boldsymbol{x}\| + d_g\right]^2\right)\right], \tag{5.8}$$

with the constants from Table 5.1. Figure 5.1 shows the solution in two dimensions for the fast and thermal groups. The thermal group solution has a sharp valley in the IFBA region, which tests the effect of sharp gradients on the number of neighbors and choice of Wendland function.

The second problem tests the ability of the code to reproduce a monoenergetic sinusoidal function given continuous cross sections. The domain is $\boldsymbol{x} \in [-2, 2]$ and the solution is of the form

$$\Phi_\ell^m(\boldsymbol{x}) = f_\ell^m\left(1 + \frac{1}{5}\cos(\pi x)\cos\left(\frac{\pi}{5}y\right)\cos\left(\frac{2\pi}{5}z\right)\right), \tag{5.9}$$

with the f_ℓ^m constants from Table 5.2a. This results in a single period for the sinusoidal function in the x dimension and less than half a period in the y and z dimensions (Fig. 5.2). The cross sections are

$$\Sigma(\boldsymbol{x}) = \Sigma_0\left(1 + \frac{1}{2}\cos\left(\frac{\pi}{5}x\right)\cos\left(\frac{6\pi}{5}y\right)\cos\left(\frac{2\pi}{5}z\right)\right), \tag{5.10}$$

where Σ_0 is represents the base value of each cross section from Table 5.2b and Σ represents the spatially-

Table 5.2: Manufactured solution and cross section data for sinusoidal problem

(a) Solution coefficients [see Eq. (5.9)]

f_0^0	f_1^{-1}	f_1^0	f_1^1
1.0	0.01	0.1	-0.05

(b) Cross sections [see Eq. (5.10)]

Σ_t	$\Sigma_{s;0}$	$\Sigma_{s;1}$	$\nu\Sigma_f$
1.0	0.5	0.07	0.0

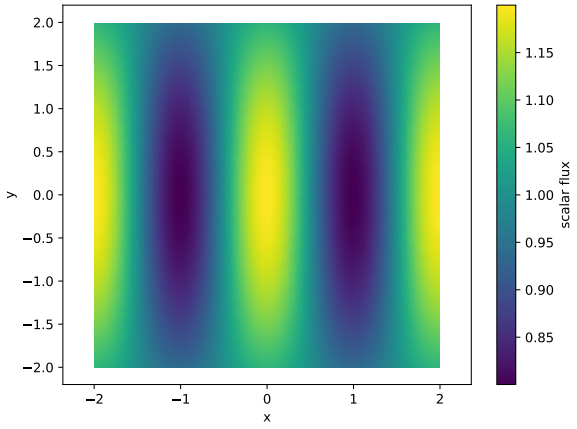


Figure 5.2: Solution to 2D manufactured sinusoidal problem

dependent value of this cross section (e.g. to calculate $\Sigma_{s;1}(\boldsymbol{x})$, Σ_0 is set to the corresponding value of 0.07 from the table).

Both problems include linearly anisotropic scattering. As described in Sec. 5, the manufactured internal source requires that the solution be performed with an additional scattering moment above the scattering order, which raises the total number of spherical harmonic moments in the solution from three to six in 2D and from four to nine in 3D. These additional moments are expected to have a numerical solution near zero for the converged transport solution. The following results use the scalar flux for the error calculation, as shown in Eq. (5.7). For all cases, the first spherical harmonics moments of the equation have similar errors to the scalar flux and the additional moments have values near zero.

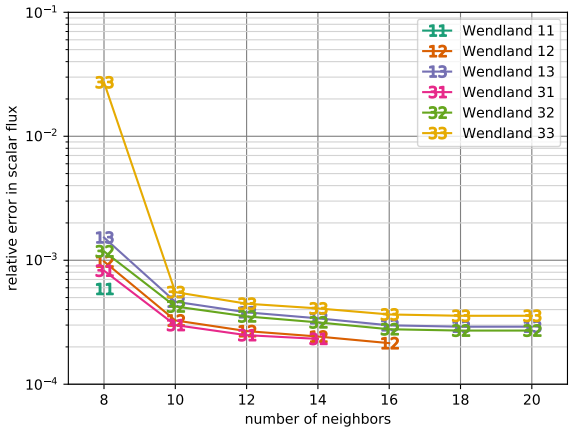
5.3 Results for manufactured problems

These results include optimization for the radii of the basis and weight functions, optimization for the SUPG parameters, convergence results and summary of parameters for use in more realistic problems. The radial and sinusoidal results are presented together for comparison of the effect of continuous and discontinuous cross sections on the weak and strong discretization methods.

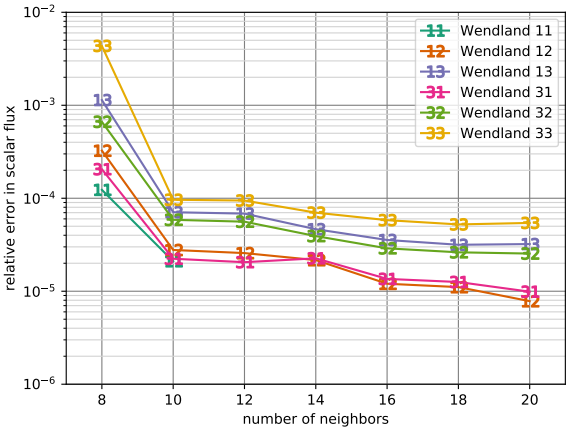
5.3.1 Optimization for the number of neighbors

To find an accurate value for the number of neighbors in the radius calculation (Alg. 3.3) for the weak form, the value is varied between 8 to 20 neighbors in 2D and 12 to 26 neighbors in 3D. For the MLPG solution, the SUPG stabilization parameter is fixed at $\tau = r_j$, or $c = 1$ in Eq. (3.1), and the number of points is fixed at 4096 for the 2D problems and for the 3D radial problem and 32768 for the 3D sinusoidal problem. The full cross section weighting technique is used for all problems.

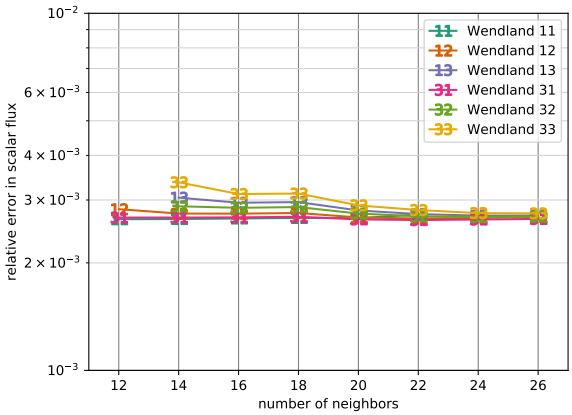
For the 2D radial problem (Fig. 5.3a) solution by the MLPG equations, the Wendland 11 function shows the best agreement with the manufactured solution at 8 neighbors, with just about 10^{-4} relative error in the solution. For 10 to 14 neighbors, the Wendland 31 function is the optimal choice at around 2×10^{-5} relative error. For the 2D sinusoidal problem (Fig. 5.3b), the Wendland 11 function shows the best agreement for 8



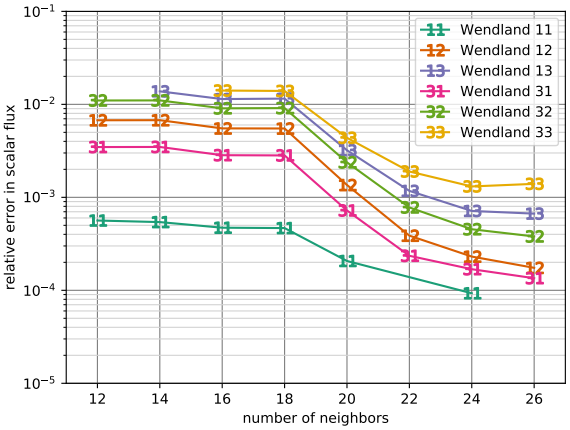
(a) Radial problem, 2D, 4096 points



(b) Sinusoidal problem, 2D, 4096 points



(c) Radial problem, 3D, 4096 points

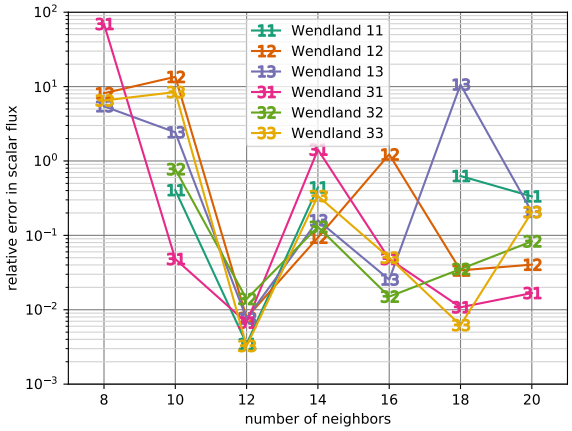


(d) Sinusoidal problem, 3D, 32768 points

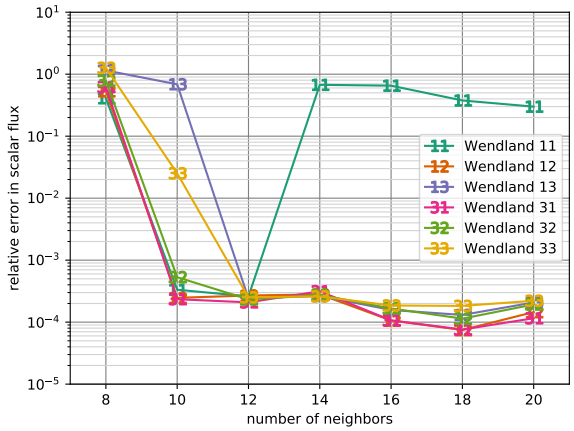
Figure 5.3: Dependence of the error of the manufactured problems on number of neighbors in the radius calculation, weak form

to 10 neighbors and Wendland 31 function is again better for more neighbors. For the radial and sinusoidal problems, the Wendland 11 function does not converge for more than 10 or 12 neighbors, respectively. For the 3D radial problem (Fig. 5.3c), the Wendland11 and Wendland 31 functions perform similarly well, but the results are relatively agnostic to the basis function and have around 3×10^{-3} error for most Wendland function and radius choices. Finally, for the 3D sinusoidal problem the results between basis and weight functions differ more significantly, likely due to the higher number of points. The Wendland 11 function performs best by almost an order of magnitude, at around 5×10^{-4} relative error for 12 to 18 neighbors and then decreasing to less than 10^{-4} for 24 neighbors.

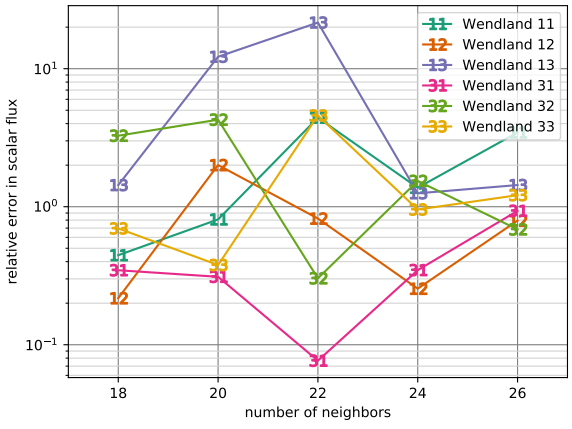
Compared to the weak discretization, the strong discretization depends much more on an optimal choice for the number of neighbors to get an accurate solution for problems with large gradients and material discontinuities. The point weighting is used for the strong-form neighbor calculations. In 2D, the sinusoidal



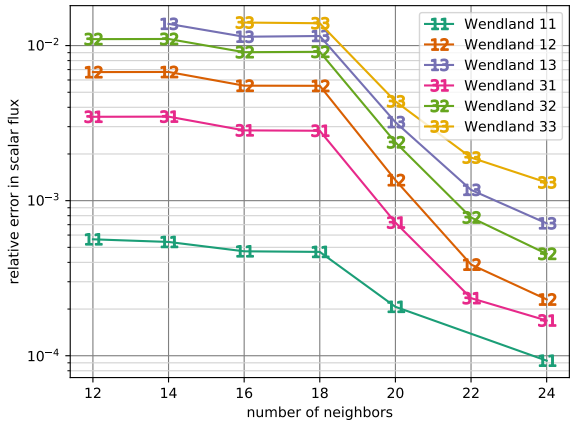
(a) Radial problem, 2D, 4096 points



(b) Sinusoidal problem, 2D, 4096 points



(c) Radial problem, 3D, 4096 points



(d) Sinusoidal problem, 3D, 32768 points

Figure 5.4: Dependence of the error of the manufactured problems on number of neighbors in the radius calculation, strong form

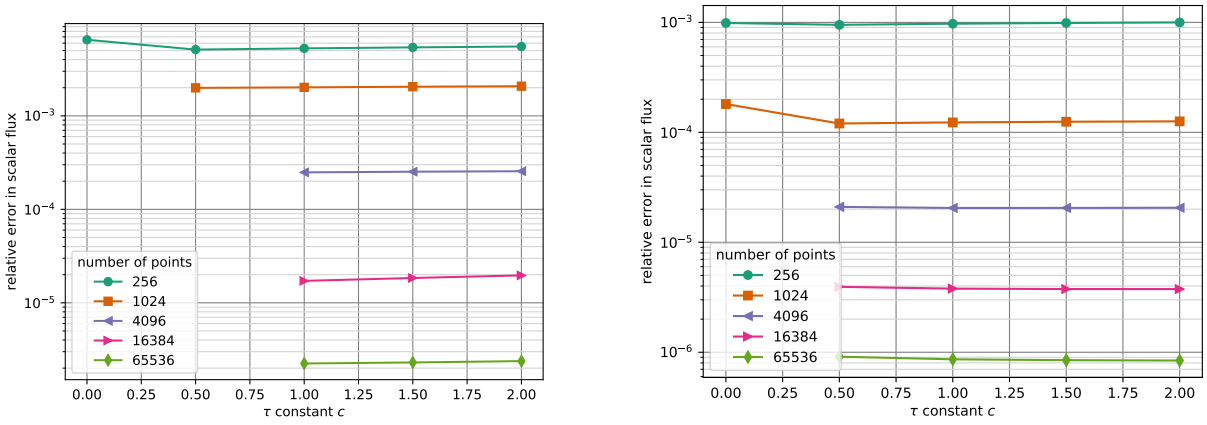
problem (Fig. 5.4b) has a large range of acceptable neighbor parameters from 10 to 20 for which the relative error of the solution is below 10^{-3}. In comparison, for the radial problem (Fig. 5.4a) the error of the solution depends strongly on the number of neighbors. Only for 12 neighbors do all the basis functions produce results below 10^{-2} relative error for more than one basis function choice. By changing the number of neighbors from this value by only two in the radius calculation, the error increases for some basis functions to around 10 times larger than the solution itself.

This chaotic behavior of the solution with respect to the shape parameter is shown more clearly in the 3D results. The sinusoidal problem behaves normally (Fig. 5.4d) and achieves error of below 10^{-3} for any number of neighbors with the Wendland 11 function. The other functions converge to similar values by 20 to 24 neighbors. For the radial problem (Fig. 5.4c), the scattering source did not converge any number of neighbors less than 18. For the solutions that did converge, only one is below 10^{-1} in relative error, which may be due to cancellation of error. This chaotic nature of the solution is likely not instability in the linear solve of the $\mathcal{L}^{-1}$ operator, which is performed by a direct solve for the strong problems (see Sec. 3.3). Instead, this is an incorrect result given by the $\mathcal{L}^{-1}$ operator for a majority of parameters. This may be due to the lack of integration to account for the materials accurately or due to instabilities that propagate through the problem caused by localized gradients. In either case, the results indicate poor performance for the strong form discretization as presented in Sec. 2.9 for problems with discontinuities and localized gradients. The strong method may still be applicable to problems with continuous cross sections and smooth solutions, such as the sinusoidal problem presented here.

5.3.2 Optimization for the stabilization parameter

The relative error with respect to the SUPG parameter [Eq. (3.1)] for the weak form is shown for the radial and sinusoidal problems in Figs. 5.5a and 5.5b, respectively. These results use the Wendland 31 function with 12 neighbors and full cross section weighting. In both cases, the result of the calculation is largely unchanged by the value of τ from about $c = 1.0$ to $c = 2.0$. However, the iterative $\mathcal{L}^{-1}$ solve did not converge for many of the $c = 0.0$ to $c = 0.75$ problems due to ill-conditioning. For a low number of points, the weak form without SUPG stabilization ($c = 0.0$) produces acceptable results. As the number of points increases, the results without stabilization do not converge. When a direct solver is applied to the problems without stabilization, oscillations appear throughout the problems, again signifying ill-conditioning issues inherent to the numerical solution of advection problems without stabilization.

These results suggest that while the SUPG stabilization is important in achieving good conditioning and low errors, the selection of the parameter governing how much stabilization is added is not of paramount importance. These same results hold for other choices of Wendland function and number of neighbors.



(a) Radial problem

(b) Sinusoidal problem

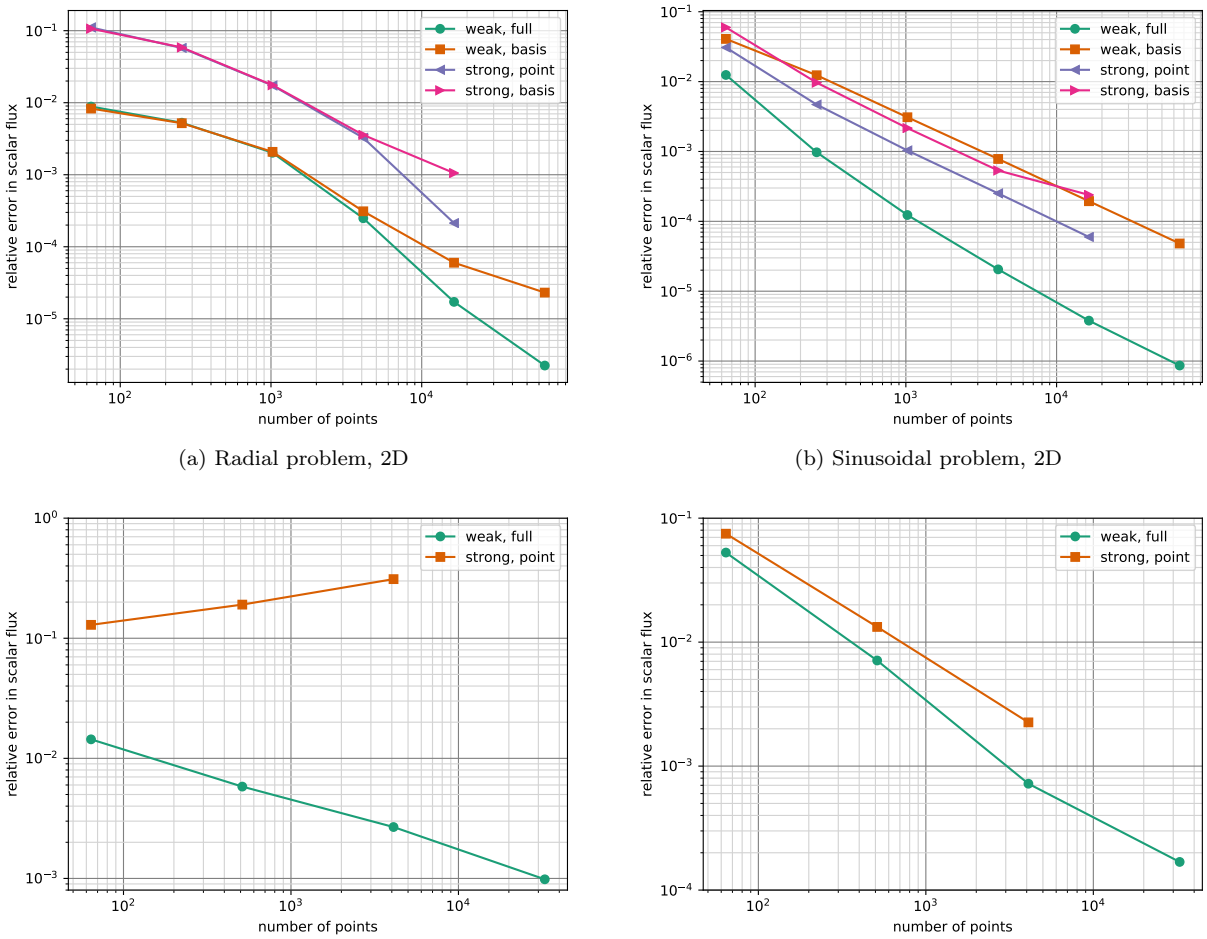
Figure 5.5: Dependence of the error of the 2D manufactured problems on the SUPG constant τ

5.3.3 Convergence results

Using functions and neighbor choices based on the parametric studies from this section, convergence results are calculated for each problem and discretization in two and three dimensions . The weak discretization uses the Wendland 31 function with an SUPG parameter of $c = 1.0$. The strong discretization uses the Wendland 33 function for the 2D results and the Wendland 31 function for the 3D results. For the 2D problems, 12 neighbors are used for each of the strong and weak discretizations. For 3D problems, the weak form uses 14 neighbors for the radial problem and 24 for the sinusoidal, while the strong form uses 20 neighbors for the radial problem and 24 for the sinusoidal. The strong-form results are limited to a relatively low number of points due to the high computational cost of the direct solve used in the $\mathcal{L}^{-1}$ operator.

In two dimensions, both the strong and weak discretizations converge with approximately second-order accuracy for the radial problem (Fig. 5.6a). The weak form converges for other functions and neighbors, while the strong form for other functions and neighbors often diverges. The problems with basis weighting of the cross sections converge more slowly than those with the full and point weighting for the weak and strong forms, respectively. The strong form with either cross section weighting performs over an order of magnitude worse than the weak form with full cross section weighting. The results for the sinusoidal problem in 2D (Fig. 5.6b) also indicate second-order convergence. As before, the strong discretization for the sinusoidal problem with either weighting has more than an order of magnitude higher error than the weak form with full cross section weighting.

For the 3D results, the weak form uses the full cross section weighting while the strong form uses the point cross section weighting. The results for the radial problem in 3D (Fig. 5.6c) do not show convergence



(a) Radial problem, 2D

(b) Sinusoidal problem, 2D

(c) Radial problem, 3D

(d) Sinusoidal problem, 3D

Figure 5.6: Convergence of the manufactured problems for the strong and weak discretizations

to the manufactured solution. The weak-form solution converges to 10^{-3} relative error at 32768 points. For the sinusoidal problem in 3D (Fig. 5.6d), the strong-form solution converges to around 3×10^{-3} relative error at 4096 points. The weak form converges more quickly to 7×10^{-4} error at 4096 points and less than 2×10^{-4} error at 32768 points.

5.3.4 Parameters for realistic problems

The manufactured solution parameter studies inform the choice of parameters for the problems in Chapters 6 and 7. For the MLPG equations, the Wendland 11 function is used as the weighting function for the MLS basis and weight functions for many of the problems, as it performs well even at a low number of neighbors. The number of neighbors is chosen as 8 for the 1D and 2D problems in Secs. 6.2 and 7.1 and 12 to 14 for the 3D problems in Secs. 6.3 and 7.4. The SUPG multiplication factor c [Eq. (3.1)] is set at 1.0 for most problems, or the lowest value for which both the manufactured problems are stable for a high number

of points. For some difficult problems, a multiplication factor of $c = 1.5$ helps the $\mathcal{L}^{-1}$ iterative solution converge more quickly.

Chapter 6

Results for steady-state problems

The steady-state problems included in this section include:

- A one-dimensional, homogeneous, purely-absorbing slab (Sec. 6.1);
- A one-dimensional, heterogeneous slab (Sec. 6.2); and
- A three-dimensional Kobayashi problem with a void region (Sec. 6.3).

The homogeneous slab is compared against an analytic solution . The heterogeneous slab is compared against a DFEM solution and a Monte Carlo solution. The three-dimensional Kobayashi problem is compared to analytic results for a purely-absorbing case and Monte Carlo results for a partially-scattering case.

The boundaries for the problems in this chapter and for the eigenvalue results in Chapter 7 are exclusively Cartesian planes. This is a simplification that reduces the complexity of reflection and integration. The 2D and 3D LDFE quadratures have quadrant and octant symmetry, respectively. For the reflection angle from Eq. (2.13), Cartesian boundaries guarantee that for any $\mathbf{\Omega}_n$, the reflection direction $\mathbf{\Omega}_{n_r}$ is also a member of the quadrature. The Cartesian boundaries also allow simple integration of the domain and boundaries using a Cartesian mesh. In addition, surfaces with a constant normal direction allow for direction-independent surface integration for the basis and weight functions. For arbitrary boundary surfaces, a more specialized integration mesh and interpolation between angular directions would be required.

The general parameters for the slab and void problems are shown in Table 6.1. As discussed in Chapter 5, to maintain a large enough radius for the basis and weight functions the number of neighbors for the radius calculation should be larger for a higher-dimensional problem.

6.1 One-dimensional, homogeneous, purely-absorbing slab

This slab-geometry problem is designed to test the ability of the MLPG discretization to calculate solutions in optically thick and optically thin problems. The problem consists of a homogeneous, one-group, purely-absorbing slab with a length of $L = 1.0$, an isotropic source of ψ_0 on the left side of the problem and no internal source. The transport equation for this problem with the appropriate boundary conditions can

Table 6.1: Solution parameters for steady-state problems

Parameter	Reference	Homogeneous	Heterogeneous	Kobayashi
Num. neighbors for radius calculation	Alg. 3.3	8		12–14
Multiplication factor for radius		1.0		
SUPG multiplication factor c	Eq. (3.1)	1.0		
$\mathcal{L}^{-1}$ convergence tolerance	Sec. 3.3	10^{-12}		10^{-10}
Solution convergence tolerance	Sec. 3.4	-	10^{-10}	10^{-8}
Number of angular directions	Sec. 3.1	8192	256	512
Wendland RBF used to create MLS	Sec. 2.10	31	11	

be written as

$$\mu \frac{\partial}{\partial x}\psi\left(x,\mu\right) + \Sigma_t \psi\left(x,\mu\right) = 0, \qquad 0 \leq x \leq L, \quad -1 \leq \mu \leq 1, \tag{6.1a}$$

$$\psi\left(x,\mu\right)\Big|_{x=0} = \psi_0, \qquad \mu > 0, \tag{6.1b}$$

$$\psi\left(x,\mu\right)\Big|_{x=L} = 0, \qquad \mu < 0, \tag{6.1c}$$

where μ is the angle cosine. The solution to this problem is

$$\psi_{\text{exact}}\left(x,\mu\right) = \begin{cases} \psi_0 \exp\left(-\dfrac{\Sigma_t x}{\mu}\right), & \mu > 0, \\ 0, & \mu < 0, \end{cases} \tag{6.2}$$

from which the scalar flux at any point in the problem can be calculated by integrating over the angular cosine,

$$\begin{aligned} \phi_{\text{exact}}\left(x\right) &= \int_{-1}^{1} \psi\left(x,\mu\right) d\mu \\ &= \int_{0}^{1} \psi_0 \exp\left(-\frac{\Sigma_t x}{\mu}\right) d\mu. \end{aligned} \tag{6.3}$$

This solution provides results to compare against for the numerical simulations.

The angular distribution of neutrons that reach the right edge of the slab at $x = L$ is forward peaked for an optically thick problem ($\Sigma_t \gg 1$), whereas for an optically thin problem ($\Sigma_t \ll 1$), the angular distribution of neutrons is relatively flat except near $\mu \ll 1$, for which the probability of a neutron reaching the far side of the slab is very low. These distributions are shown in Fig. 6.1 for various total cross section values. The angular integration of the $\Sigma_t = 1.0$ case is straightforward and does not require a large number of ordinates

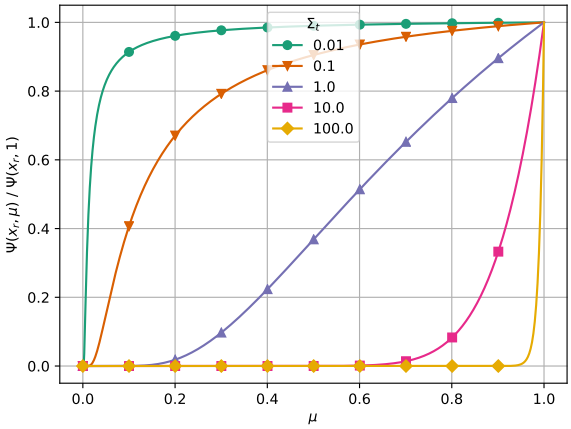


Figure 6.1: Normalized angular distribution of neutrons exiting purely absorbing slab for various cross sections

due to the approximately linear shape of the function. The cases of $\Sigma_t = 0.01$ and $\Sigma_t = 100$, however, have large gradients that require many ordinates to accurately integrate. Because of this, 8192 angular ordinates are used for this problem, as shown in Table 6.1, to limit the effect of the angular discretization on the accuracy of the MLPG solution.

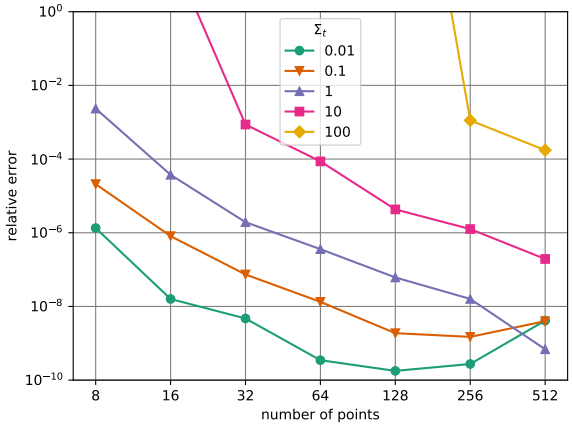
The problem is tested for the cross section cases of $\Sigma_t = 0.01,\ 0.1,\ 1.0,\ 10.0,\ 100.0$ and between 8 and 512 points. Because the problem is purely absorbing, the solution converges after a single outer iteration. The solution for the scalar flux at the right boundary of the problem, $\phi(L)$, is compared to the analytic solution from Eq. (6.3) using the relative error

$$\epsilon = \frac{|\phi_{\text{exact}}(L) - \phi(L)|}{\phi_{\text{exact}}(L)}. \tag{6.4}$$

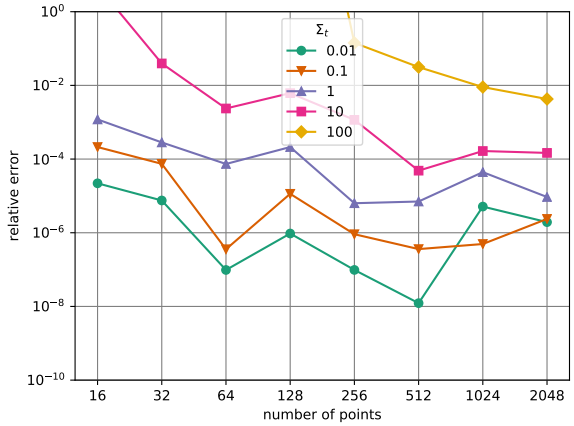
The values for the exact scalar flux at the right edge of the problem range from 3.64782×10^{-46} for $\Sigma_t = 100$ to 0.999037 for $\Sigma_t = 0.01$.

Figure 6.2 includes the error for MLPG, strong-form collocation and DFEM simulations. All three use the same angular discretization. The relative errors for the MLPG method are below 10^{-4} by 64 points for all cross section cases except $\Sigma_t = 100$, which has a flux value at the boundary that is 46 orders of magnitude lower than the initial boundary source and requires 512 points to reach a relative error of 1.7×10^{-4}. Relative errors are less than 10^{-8} by 512 points for cases with $\Sigma_t \leq 1$. The convergence rate of the MLPG solution appears to be around third-order. The solution converges much faster than this for a low number of points. For the optically thin cases, the conditioning of the problem appears to become worse for 256 to 512 points, causing the error to increase slightly.

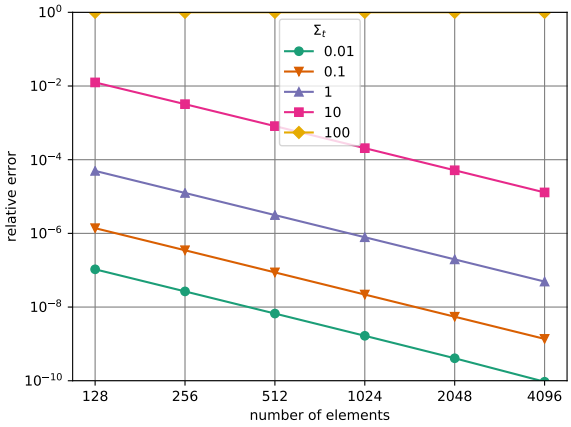
The relative error for the collocation method is around two to three orders of magnitude larger than the



(a) MLPG



(b) Strong-form collocation



(c) Linear DFEM, for comparison

Figure 6.2: Relative error in scalar flux for neutrons exiting purely absorbing slab for various cross sections

error for the MLPG method for a similar number of points. For a large number of points, the collocation method seems to not converge past 10^{-6} relative error, possibly due to numerical conditioning issues. For the optically-thick $\Sigma_t = 100$ case, the collocation method requires almost ten times as many points as the MLPG method to converge to 10^{-2} error. There is not a clear enough pattern in the results to calculate a convergence rate of the solution with any confidence.

The DFEM solution shows exactly second-order convergence for the $\Sigma_t \leq 10$ cases. For $\Sigma_t = 100$, the numerical solution is zero regardless of the number of points, which may be due to the mass matrix lumping technique employed to prevent negative solutions (See Ref. [40] for details). For a similar number of spatial unknown values (two per element for the DFEM solution), the MLPG solution has between two and four orders of magnitude lower error than the DFEM solution. The smallest two numbers of elements shown for the DFEM solution have the same number of unknowns as the largest two for the MLPG solution. For this problem, the errors are similar for 64 spatial points in the MLPG method and 2048 elements in the linear

DFEM method.

6.2 One-dimensional, heterogeneous slab reactor

The second problem is a one-dimensional, two-group, steady-state slab, with geometry shown in Fig. 6.3 and and cross sections listed in Table 6.2. The boundary conditions on the left and right are reflective and vacuum, respectively. The source region represents a multiplicative medium whose fast neutrons are moderated to thermal energies in the moderator region. The thermalized neutrons are then captured by the absorber. The solution for the problem in Fig. 6.4 is consistent with the expected birth of fast neutrons in the source region, the transfer to the thermal group in the moderator region and the preferential absorption of thermal neutrons in the absorber region.

The benchmark solution for the one-dimensional problem is generated using a DFEM code with 500000 elements and linear basis and weight functions within each element. The average solution $\bar{\phi}_{i,g}^{\text{bench}}$ over 1000 overlaid cells with index i for each energy group g is then calculated. After completion of the MLPG solution, the average scalar flux $\bar{\phi}_{i,g}$ across each of the same regions is calculated by integration with a 64-point Gauss-Legendre quadrature. Finally, the L_2 error of the meshless solution with respect to the benchmark solution

Table 6.2: Heterogeneous slab cross sections

Mat.	g	$\Sigma_{t;g}$	$\Sigma_{s;0,g\to1}$	$\Sigma_{s;0,g\to2}$	$\Sigma_{s;1,g\to1}$	$\Sigma_{s;1,g\to2}$	$\chi\nu\Sigma_{f;g\to1}$	$\chi\nu\Sigma_{f;g\to2}$	Q_g
Sou.	1	1.0	0.9	0.05	0.1	0.0	0.0	0.0	1.0
	2	2.0	0.05	0.8	0.0	0.0	1.0	0.0	0.0
Mod.	1	0.5	0.05	0.45	0.0	0.1	0.0	0.0	0.0
	2	1.0	0.0	1.0	0.0	0.01	0.0	0.0	0.0
Abs.	1	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	2	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

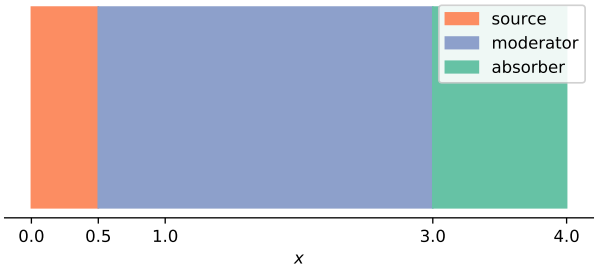


Figure 6.3: Heterogeneous slab geometry

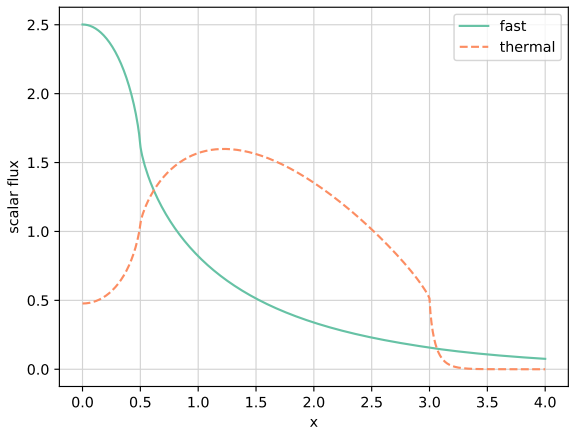


Figure 6.4: Heterogeneous slab solution, 5120 points

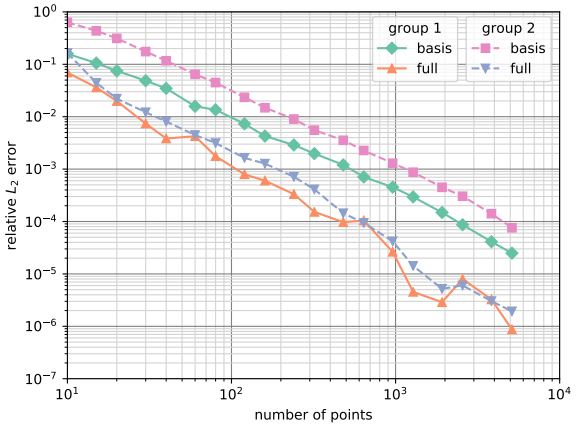


Figure 6.5: Heterogeneous slab convergence

is calculated as

$$(L_2 \text{ error})_g = \left(\frac{\sum_i \left(\bar{\phi}_{i,g} - \bar{\phi}_{i,g}^{\text{bench}} \right)^2}{\sum_i \left(\bar{\phi}_{i,g}^{\text{bench}} \right)^2} \right)^{1/2}. \tag{6.5}$$

With the solution parameters in Table 6.1, a convergence study for the number of points is performed for each of the basis and full cross section weightings as described in Sec. 2.8. The points are placed evenly throughout the problem. The convergence results in Fig. 6.5 show second-order convergence with the point spacing for both types of cross section weighting. Similarly to the MMS problems, the full cross section weighting has lower error than the basis weighting, in this case by more than and order of magnitude. The cellwise relative error is around an order of magnitude larger than the L_2 error near large gradients, for the fast group at the source/moderator boundary and for the thermal group at the moderator/absorber boundary.

For 5120 points and full cross section weighting, the L_2 error is 8.71×10^{-7} and 1.93×10^{-6} for the fast and thermal groups, respectively. For comparison, the L_2 errors of a 5000-element DFEM solution, which has almost twice as many degrees of freedom as the 5120-point meshless solution, are 1.45×10^{-6} and 2.71×10^{-6}. However, the DFEM solution requires about 12 seconds, compared to 32 or 50 for the MLPG solution, depending on whether a direct or iterative solver is used for the $\mathcal{L}^{-1}$ operator.

To quantify the expected error for the two and three dimensional problems in the following section that use Monte Carlo benchmarks, the convergence for this one-dimensional problem to the DFEM method is compared to the convergence to a Monte Carlo solution. The results of the MLPG method converge more quickly to a similar deterministic solution for a fixed angular quadrature rule than to a Monte Carlo solution. For this one-dimensional problem, a Monte Carlo calculation using the multigroup mode of OpenMC [62] is performed and tallies over the same 1000 cells as before with 100 generations and 10^7 particles per generation

are compared to the DFEM (500000 elements) and MLPG results (10240 points). The L_2 error from the Monte Carlo solution is the same for both to the fifth decimal place, at 4.9×10^{-4} for the fast and 2.1×10^{-4} for the thermal group. The standard deviation of the Monte Carlo flux tally does not exceed 1.6×10^{-4} for any tally cell or group and is, on average, 3.2×10^{-5} for the fast group and 6.2×10^{-5} for the thermal group. This indicates that the difference between the deterministic and Monte Carlo calculations is likely not random noise, but instead angular error introduced by the discrete ordinates approximation. For the two and three dimensional problems, good but not complete convergence of the MLPG method to the Monte Carlo solution is similarly expected.

6.3 Three-dimensional Kobayashi problem with void region

The three-dimensional Kobayashi benchmark problems [63] are designed to test deterministic codes in the presence of void regions. Problem 1 of the Kobayashi set, which is considered here, includes a cubic source at the center, a surrounding void region and a shield (Fig. 6.6 and Table 6.3). The cross sections are either purely absorbing or partially scattering. The boundary conditions for the planes at $x, y, z = 0$ cm are reflective, while the boundary conditions for the planes at $x, y, z = 100$ cm are vacuum.

Table 6.3: Kobayashi cross sections

Mat.	Σ_t	Σ_s (abs)	Σ_s (sca)	Q
Source	0.1	0.0	0.05	1.0
Void	0.0001	0.0	0.00005	0.0
Shield	0.1	0.0	0.05	0.0

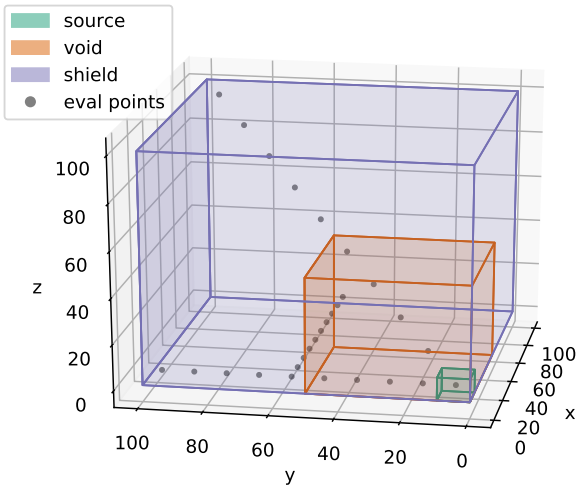


Figure 6.6: Kobayashi geometry and evaluation points

The centers for the Kobayashi problem are a simple Cartesian grid of points. The number of neighbors for the radius calculation is increased to between 12 and 14 for this 3D case to maintain the same general radius for a evenly-spaced points (Table 6.1), as discussed in Sec. 5.3.4. The scalar flux is calculated at specific points in the problem (which contrasts with the integral approach from Secs. 6.2 and 7.4) for comparison against the benchmark results at these same points from Kobayashi and Sugimura [63]. The given benchmarks are analytic for the purely absorbing case and are calculated by the Monte Carlo code GMVP for the scattering case. The overall L_2 error is calculated according to Eq. (6.5) with integrated values replaced by discrete values.

6.3.1 Convergence with spatial refinement

The convergence of these L_2 values for the absorbing and the scattering cases is shown in Fig. 6.7. The dependence on only a few pointwise quantities makes the L_2 error more volatile than previous problems, which use global or integrated quantities. The convergence plot shows several dips in the error at certain point values with full weighting, with minima at 16^3, 22^3 and 40^3 points. These irregularities in the convergence make determination of convergence rates difficult. An exponential function fit over the last ten data points (from 32^3 to 42^3 points) predicts at least second-order convergence with point spacing for all cases.

For the case with the most points, the largest relative error of around 100 percent occurs at the evaluation point (95 , 95, 95), which is at the furthest corner of the problem from the source and has a benchmark value of 3.01032×10^{-6} for the absorbing problem and 1.12892×10^{-5} for the scattering problem. This is also the point with the lowest absolute error. The evaluation point at (5, 5, 5), inside the source, has benchmark values of 5.95659 for the absorbing problem and 8.29260 for the scattering problem. This is the point with the highest absolute error but the lowest relative error, at around 0.15 percent and 0.10 percent for the

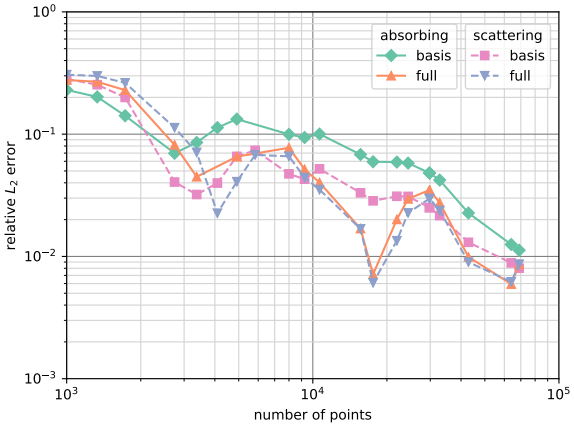


Figure 6.7: Kobayashi convergence with spatial refinement, 512 directions

absorbing and scattering problems, respectively.

6.3.2 Convergence with angular refinement

In problems with low scattering ratios, the flux is mostly uncollided. For a problem in which a small detector (such as the point detectors in this problem) is far from the neutron source, the detector may only "see" the angular flux coming from a few directions. Rays coming from other directions do not lead back to the source, and so have values of zero. In Fig. 6.8, which is the Kobayashi geometry simplified to flatland geometry, only one of the rays that carries angular information to the point detector has a nonzero value for the analytic case. Using the CSG described in Appendix B and a given quadrature, the angular directions can be traced back to the original source region to see how many of these directions contribute to the value of the scalar flux in a purely absorbing medium.

Figure 6.9 shows the total number of directions with a nonzero value for a point detector at at various positions. To have even 10 nonzero quadrature points at the detector requires 8192 directions. As discussed in Chapter 4, integration quadratures do not perform optimally in the presence of discontinuities. Problems with ray effects have discontinuities in the angular flux values, which degrades the performance of the discrete ordinates solution. For the quadrature with 2048 points, even though the point at (95 , 95, 95) has only a single quadrature point that is nonzero for the purely absorbing case, the error should still decrease compared to the lower-order quadratures as the angular size of the source is known more accurately.

The L_2 error applied for the spatial convergence study in Sec. 6.3.1 is biased toward the regions of highest

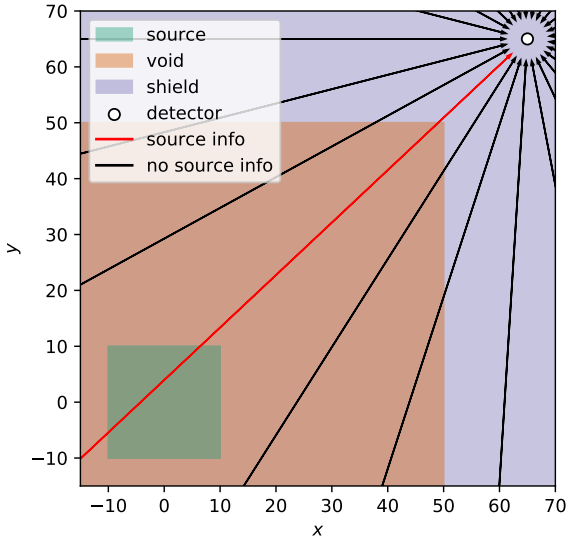


Figure 6.8: Ray effect diagram for the Kobayashi problem in a simplified flatland geometry

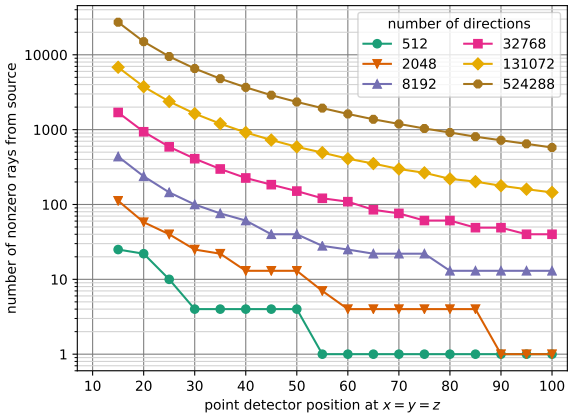
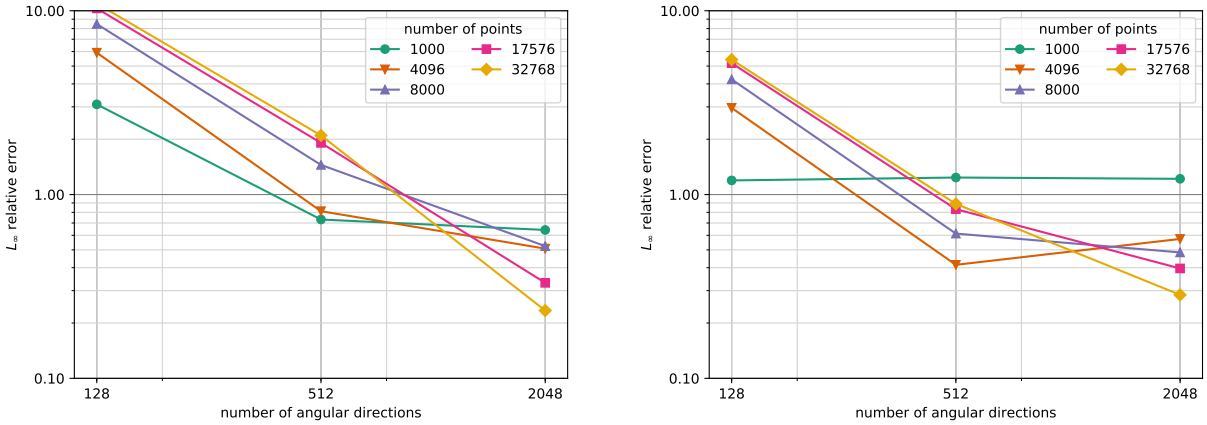


Figure 6.9: Number of nonzero angular quadrature points for a point detector in the Kobayashi problem



(a) Absorbing

(b) Partially scattering

Figure 6.10: Kobayashi convergence with angular refinement, full cross section weighting

flux, which includes the points nearest to the source that are easiest to converge. Using the the worst-case relative error,

$$L_\infty \text{ error} = \max_i \left(\left| \frac{\phi_i - \phi_i^{\text{bench}}}{\phi_i^{\text{bench}}} \right| \right), \tag{6.6}$$

however, the solution does not converge for an increasing number of points with a fixed angular quadrature rule. To show the effect of the angular error on the problem, the same scattering and absorbing problems are tested with between 10^3 and 32^3 points and 128 to 2048 directions (Fig. 6.10)

For 2048 directions the L_∞ error decreases as more points are added to the problem. For a 128 directions, however, the L_∞ error is lowest for the cases with the fewest spatial points. This is because the larger number of points causes accurate angular flux values to reach the furthest detectors for only one or two quadrature points, which the integration then incorrectly extrapolates to all nearby angular directions. The basis and weight functions for the smallest number of points (1000 points) have radii between 15 and 28 cm. This means the information from spatially integrating the basis and weight functions over the source region (which is 10 cm by 10 cm for the reflected problem) is included in basis and weight functions that have centers far from the source region. This blurring of the source may add a few more nonzero quadrature points to the integration, but also decreases the apparent source strength in the source region, which is part of why the cases with few spatial points perform poorly approximations of the closest points to the source region (see Sec. 6.3.1).

Chapter 7

Results for eigenvalue problems

The eigenvalue problems considered in these results include:

- A two-dimensional pressurized water reactor (PWR) pincell in two geometries (Secs. 7.1–7.3) and
- A three-dimensional reflected ellipsoid (Sec. 7.4).

Table 7.1 summarizes the solution parameters for each of the problems in this section, including the number of neighbors for the radius calculation, the SUPG multiplication factor, and the solution convergence tolerance. Table 7.4 shows the eigenvalue and eigenvector errors for the problems with the most spatial points. Table 7.3 and 7.2 show the multigroup and continuous-energy Monte Carlo eigenvalue results, which are used as benchmarks for the eigenvalue problems. As described for the steady-state problems in Chapter 6, these problems use Cartesian boundaries to simplify reflection at the boundaries and integration of the basis and weight functions.

Both problems use the same methodology to generate multigroup cross sections. The cross sections for use in the calculations are generated in OpenMC using ENDF/B-VII.1 cross sections [64] using the parameters from Table 7.2. The generated cross sections are used in the multigroup mode of OpenMC to calculate a benchmark k-eigenvalue solution (shown in Table 7.3). The associated eigenvector over a Cartesian tally is calculated in the same OpenMC simulation for comparison to the MLPG solution. During the MLPG

Table 7.1: Solution parameters for k-eigenvalue problems

Parameter	Reference	VERA 1B	VERA 1E	Ellipsoid
Num. neighbors for radius calculation	Alg. 3.3	8		12 / 14
Multiplication factor for radius		1.0		
SUPG multiplication factor c	Eq. (3.1)	1.0 / 1.5	1.5	1.0
$\mathcal{L}^{-1}$ convergence tolerance	Sec. 3.3	10^{-10}		
Solution convergence tolerance	Sec. 3.4	10^{-8}		
Number of angular directions	Sec. 3.1	16–4096		512
Wendland RBF used to create MLS	Sec. 2.10	11		11 / 31

Table 7.2: Continuous-energy Monte Carlo benchmark values for k-eigenvalue problems

	VERA 1B		VERA 1E		Ellipsoid
Value	600 K	1200 K	600 K	1200 K	
Generations	200 total: 150 active, 50 inactive				
Particles per generation	10^7				
k-eigenvalue	1.181827	1.163730	0.771338	0.761258	0.988033
k-eigenvalue std. dev. (pcm)	2.3	2.5	1.9	1.9	1.8
Difference from multigroup (pcm)	125.5	98.6	143.0	180.8	1172.2

Table 7.3: Multigroup Monte Carlo benchmark values for k-eigenvalue problems

		VERA 1B		VERA 1E		Ellipsoid
Value		600 K	1200 K	600 K	1200 K	
Generations		200 total: 150 active, 50 inactive				
Particles per generation		10^6				10^7
k-eigenvalue		1.180572	1.162744	0.772768	0.763066	0.999755
k-eigenvalue std. dev. (pcm)		4.0	4.0	4.2	3.9	2.0
Eigenvector std. dev., mean	Fast	0.000304	0.000307	0.000330	0.000333	0.000603
	Thermal	0.000177	0.000176	0.000174	0.000175	
Eigenvector std. dev., max.	Fast	0.000440	0.000426	0.000489	0.000476	0.002116
	Thermal	0.000267	0.000263	0.000280	0.000246	

Table 7.4: Best-case k-eigenvalue and normalized eigenvector error, full weighting

		VERA 1B		VERA 1E		Ellipsoid
Value		600 K	1200 K	600 K	1200 K	
Number of points		9825	11765	34086	34086	40078
Number of directions		4096	512	1024	1024	512
k-eigenvalue		1.180561	1.162572	0.772838	0.763150	0.998870
k-eigenvalue error (pcm)		1.0	17.2	7.0	8.5	88.5
Eigenvector L_2 error	Fast	0.000288	0.000983	0.000335	0.000327	0.001981
	Thermal	0.000374	0.000648	0.000748	0.000716	

solution, the average solution over these same elements is calculated using a Gauss-Legendre outer product quadrature [Eq. (4.19)]. The error for each group is then calculated using Eq. (6.5).

7.1 Introduction to two-dimensional VERA pincell problems

The two-dimensional pincell problems under consideration have materials and geometry as specified in Problem 1 of the VERA Core Physics Benchmark Problems [65] (see also Fig. 7.1) from the Watts Bar Unit 1 startup core. Problem 1B represents a pincell composed of UO_2 fuel with a hydrogen gap, Zircaloy-4 cladding and a light water moderator. The outer radii of the fuel, hydrogen gap and cladding are concentric circles centered at the origin with radii of 0.4096 cm, 0.418 cm and 0.475 cm, respectively. The moderator extends from the outer boundary of the cladding to the square defined by $-0.63 \leq x, y \leq 0.63$. Problem 1E has identical material properties except for a 10 µm ZrB_2 integral fuel burnable absorber (IFBA) between the fuel and the gap with an outer radius of 0.4106 cm. Both problems are considered for materials at two temperatures, 600 K and 1200 K.

The two-group cross sections at 600 K and 1200 K for use in the calculations are generated with isotopics from the VERA specifications. The 1200 K problem with the VERA 1B geometry corresponds to the VERA 1D problem from the VERA benchmark specifications. The generated cross sections (Tables 7.5 and 7.6) are used in the multigroup mode of OpenMC to calculate a benchmark k-eigenvalue solution (shown in Table 7.3). The associated eigenvector over a 10000-cell Cartesian tally with mesh spacing of $\Delta x = \Delta y = 0.0126$ is calculated in the same OpenMC simulation. After the MLPG solution, the average solution over these

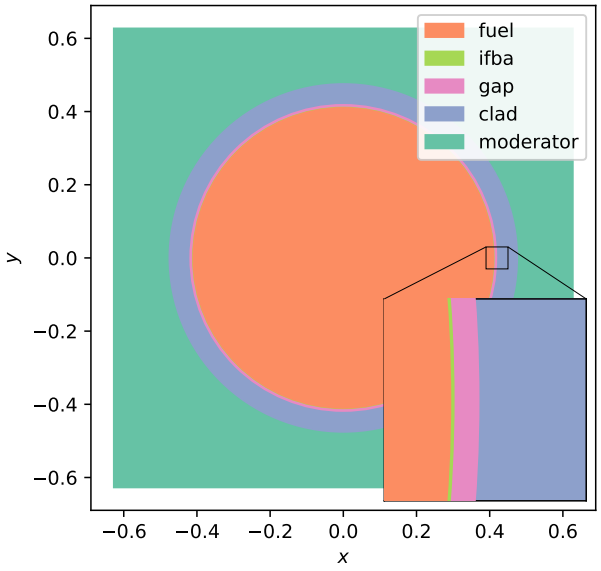
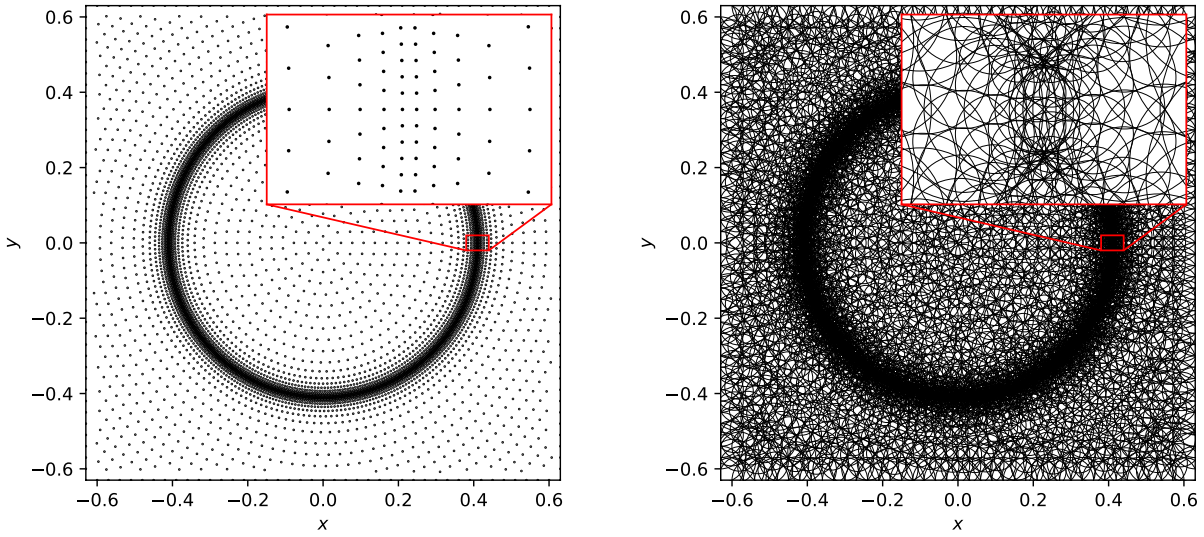


Figure 7.1: VERA pincell geometry



(a) The centers of the meshless functions

(b) The calculated radii of the meshless functions

Figure 7.2: VERA 1E meshless discretization for scaled point placement with 7644 points

10000 elements is calculated using a 1024-point Gauss-Legendre outer product quadrature.

Two separate layouts of points are tested for each problem, (1) a Cartesian grid of points and (2) a set of concentric cylindrical points with high density near the fuel boundary. The Cartesian sets have between 144 and 65536 evenly-spaced points, which corresponds to a spacing between points of 0.105 to 0.005 cm. The smallest point spacing of 0.005 cm is five times larger than the width of the IFBA region. The second set, which is referred to as the scaled set, begins near the fuel boundary (which has radius r_b) with two rings of points at $r_{1,\pm} = r_b \pm \Delta r_{min}/2$, where Δr_{min} is the desired minimum spacing between points in the problem. Further rings of points are added at $r_{n,\pm} = r_{n-1,\pm} \pm t^{n-1}\Delta r_{min}$, where t is a scaling factor. This continues until some maximum distance between points, Δr_{max} is reached, at which point the scaling stops but the concentric rings of points continue. Points outside the problem boundaries are discarded and the boundaries are lined with points approximately Δr_{max} apart. For the problem without IFBA (Problem 1B), the generation parameters are $t \approx 1.2$ and $\Delta r_{max} = 2\Delta r_{min}$ with Δr_{min} varied between 0.1 and 0.00625 cm, which results in 126 to 11765 points. For the problem with IFBA (Problem 1E), the parameters are $t \approx 1.3$ and $\Delta r_{max} = 10\Delta r_{min}$ with Δr_{min} varied between 0.006 and 0.001 cm, which corresponds to 3522 to 34086 points. The smallest Δr_{min} is equal to the length of the IFBA region, or five times smaller than the point spacing of the largest Cartesian set of points. The starting azimuth for the cylindrical points is randomized to prevent a set of radii along the starting axis that are closer together than at other azimuthal points. The other constants used in the simulation are shown in Table 7.1.

Figure 7.2 shows a set of basis and weight functions created for the VERA 1E problem using this algorithm

for $\Delta r_{min} = 0.003$ and $\Delta r_{max} = 10\Delta r_{min}$. The centers are concentrated at the boundary of the fuel, where the gradients due to the IFBA layer are highest. The radii are calculated using 8 neighbors and correspond to the distances to the nearest points. The radii are largest at the corners, where the points are far apart and there are fewer surrounding points due to the boundaries. The radii for the basis and weight functions for this example vary from around 0.004 cm near the fuel edge to 0.06 cm at the corners of the problem.

7.2 VERA 1B, standard pincell

The cross sections for VERA 1B at 600 K and 1200 K are shown in Table 7.5. For the VERA 1B problem, the eigenvectors for the fast and thermal groups (Figs. 7.3a and 7.3b) are consistent with the expected birth of the neutrons in the fuel, moderation in the water and reabsorption and fission in the fuel. The cladding area is not axially symmetric in these plots because of ray effects. For four times as many directions (1024 total), these visible ray effects go away but the error in the eigenvalue does not significantly decrease (see Sec. 7.2.2). Figure 7.4 shows the error in the eigenvector at 600 K for 256 and 4096 directions and a 3273 points.

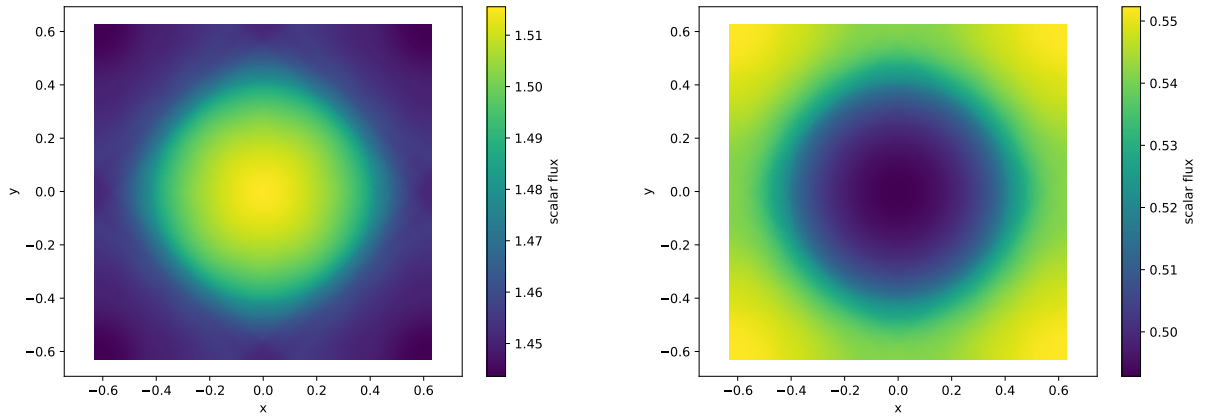
Table 7.5: VERA 1B cross sections

(a) 600 K

Mat.	g	$\Sigma_{t;g}$	$\Sigma_{s;0,g\to1}$	$\Sigma_{s;0,g\to2}$	$\Sigma_{s;1,g\to1}$	$\Sigma_{s;1,g\to2}$	$\chi\nu\Sigma_{f;g\to1}$	$\chi\nu\Sigma_{f;g\to2}$
Fuel	1	0.399697649	0.383829618	0.000830382	0.049482651	-0.000261476	0.013687612	0.000000015
	2	0.581826884	0.0	0.405420306	0.0	0.006013128	0.255838918	0.000000189
Gap	1	0.000060070	0.000059744	0.000000396	0.000011213	-0.000000063	0.0	0.0
	2	0.000021458	0.0	0.000021460	0.0	-0.000000063	0.0	0.0
Clad	1	0.319205599	0.317083075	0.000287583	0.052759371	-0.000078648	0.0	0.0
	2	0.297115812	0.0	0.294003048	0.0	0.002146731	0.0	0.0
Mod.	1	0.528320878	0.486251066	0.041923677	0.290403747	0.018069833	0.0	0.0
	2	1.316352822	0.000000031	1.301429236	0.000000031	0.523147177	0.0	0.0

(b) 1200 K

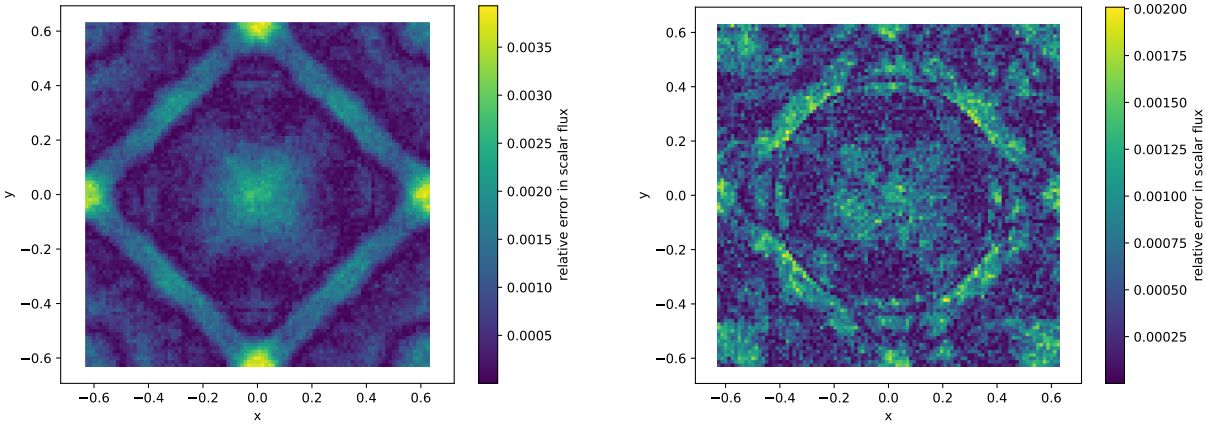
Mat.	g	$\Sigma_{t;g}$	$\Sigma_{s;0,g\to1}$	$\Sigma_{s;0,g\to2}$	$\Sigma_{s;1,g\to1}$	$\Sigma_{s;1,g\to2}$	$\chi\nu\Sigma_{f;g\to1}$	$\chi\nu\Sigma_{f;g\to2}$
Fuel	1	0.402896222	0.386358031	0.000813523	0.049553511	-0.000260073	0.013671378	0.000000014
	2	0.577168400	0.0	0.399441817	0.0	0.005555131	0.255515184	0.000000183
Gap	1	0.000060118	0.000059701	0.000000377	0.000011605	-0.000000078	0.0	0.0
	2	0.000022236	0.0	0.000022244	0.0	0.000003476	0.0	0.0
Clad	1	0.319199316	0.317088281	0.000285277	0.052818998	-0.000074498	0.0	0.0
	2	0.297107336	0.0	0.294001433	0.0	0.002160237	0.0	0.0
Mod.	1	0.527812983	0.486091382	0.041571306	0.290244792	0.017895899	0.0	0.0
	2	1.315303607	0.000000028	1.300403328	0.000000028	0.523131159	0.0	0.0



(a) Fast energy group

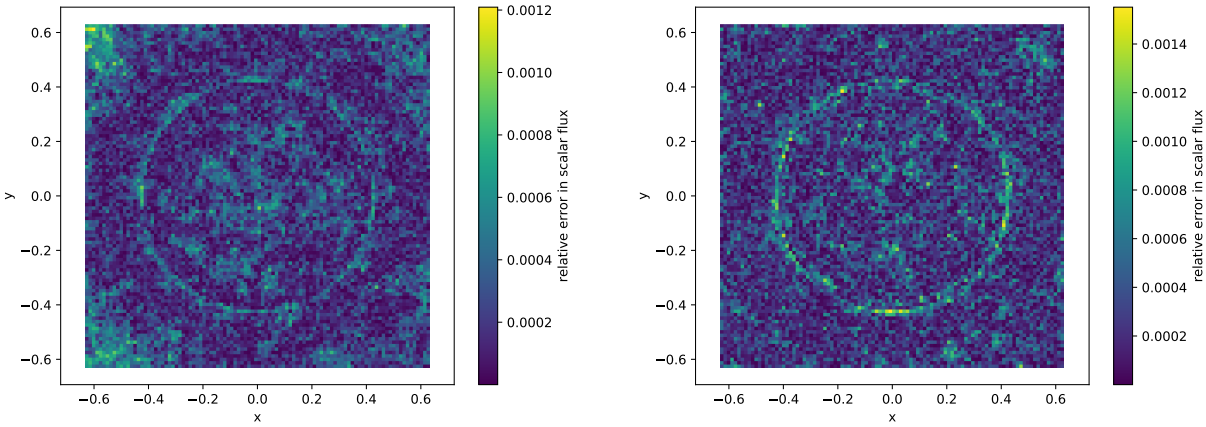
(b) Thermal energy group

Figure 7.3: VERA 1B eigenvector, 11765 points, 256 directions



(a) Fast energy group, 256 directions

(b) Thermal energy group, 256 directions



(c) Fast energy group, 4096 directions

(d) Thermal energy group, 4096 directions

Figure 7.4: VERA 1B normalized eigenvector error, 600 K, 3273 points

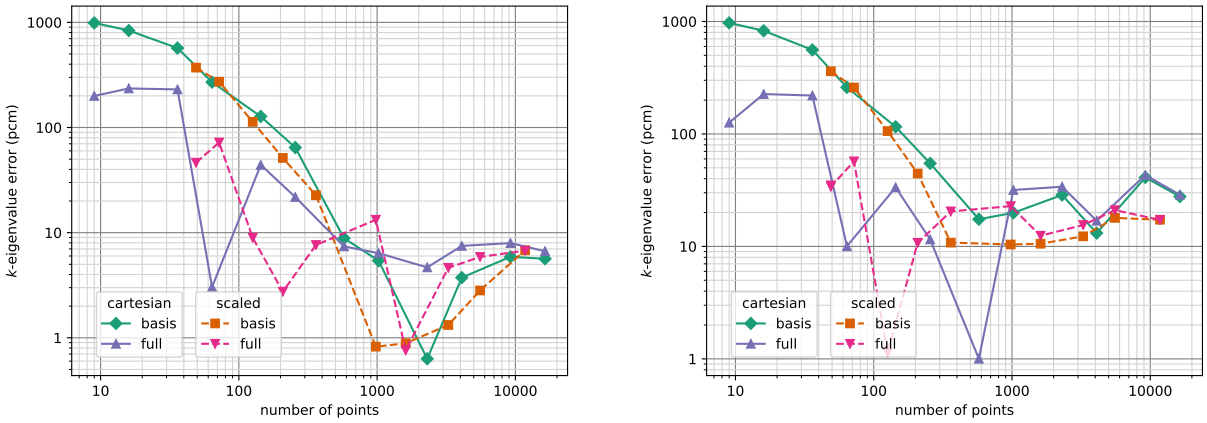
For the 256-direction problem, much of the error is in the ray effects around the edges of the pincell. For 4096 directions, the error from the angular discretization has subsided, which allows the error in the spatial discretization to be seen more clearly. The error in the spatial shape of the eigenvector is concentrated at the fuel edge, where the discontinuities are largest and the radii of the basis and weight functions are smallest. This is in line with the integration results from Sec. 4.5, which show that the integration is least accurate near the edge of the fuel boundary.

As shown in Table 7.2, difference between the continuous-energy and multigroup benchmark solutions is around 125.5 pcm for the 600 K solution and 98.6 pcm for the 1200 K solution. The energy discretization error for these problems is around ten times larger than the spatial discretization error. As the MLPG solutions are compared to multigroup results from OpenMC, this energy discretization error is not a factor in the convergence studies.

7.2.1 Convergence with spatial refinement

The convergence plot in Fig. 7.5 for 256 directions shows quick convergence to under 10 pcm with under 1000 points for both point geometries and both cross section weighting methods for the 600 K problem. For some point configurations the error drops below 1 pcm, but this is likely due to cancellation of error as all cases eventually converge to the same value of about 7 pcm error by 11765 points. For comparison, the standard deviation of the Monte Carlo benchmark solution is of approximately the same size at 4.0 pcm (Table 7.3). The convergence rate for the first four data points is at least second-order.

The 1200 K solution shows the same general trends as the 600 K solution but only converges to around 17 pcm error. Comparing the two convergence plots, the scaled discretizations both have the lowest error



(a) 600 K

(b) 1200 K

Figure 7.5: VERA 1B convergence of k-eigenvalue with spatial refinement, 256 directions

around 1000–2000 points, at under 1 pcm for the 600 K case and around 10 pcm for the 1200 K case. It is possible that the solution is actually best for these values and that then roundoff or other numerical errors cause the error to increase. However, it is more likely that the discrete solution at these values is simply not converged to the values a discrete benchmark with this angular discretization would provide. Based on the results from Sec. 6.2, the Monte Carlo and fully-converged discrete solutions are not expected to agree perfectly. If both the 600 K and 1200 K fully-converged discrete eigenvalues differ similarly from the Monte Carlo solutions, then it may be that the error around 1000–2000 points is passing through the Monte Carlo converged results on its way to the fully-converged discrete results.

The L_2 errors are low for the 1B case because the eigenvectors are normalized and relatively flat. For the solutions with the most points for 600 K, the L_2 error of around 0.0003 for both the fast and thermal groups (Table 7.4) is on the same order as the mean standard deviation in the Monte Carlo mesh tally of 0.0003 for the fast group and 0.0002 for the thermal group (Table 7.3). The 1200 K case, which is not converged in angle, has two to three times this error at 0.0009 for the fast group and 0.0006 for the thermal group.

7.2.2 Convergence with angular refinement

As previously discussed, ray effects make up a large part of the eigenvector error for the 256-ordinate calculations. To see what effect this eigenvector error has on the eigenvalue, an angular convergence study is performed for between 16 and 4096 directions and 230 to 9825 points for the 600 K case. Figure 7.6 shows that the effect of the angular discretization on this problem is significant at around 10–20 pcm. The problem with the most points and directions converges to around 1.0 pcm, which based on the trend of the lines for 4300 and 9825 points may be actual convergence and not numerical roundoff. As shown in Sec. 7.2.1, the discrete solution does not converge to the Monte Carlo solution when the number of points is increased

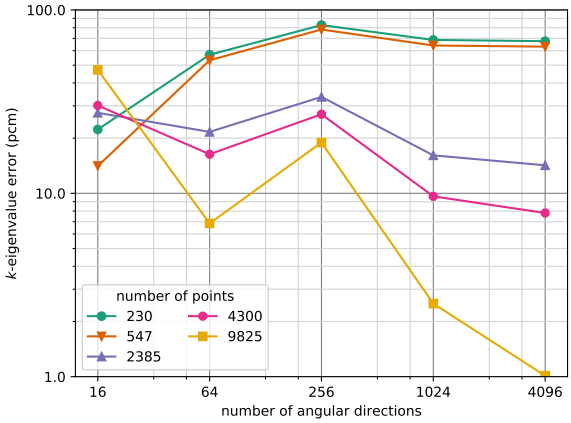


Figure 7.6: VERA 1B convergence of k-eigenvalue with angular refinement, 600 K, full weighting, scaled geometry

without also increasing the angular quadrature order. The error in the solution is below 20 pcm for this problem for any reasonable number of points and directions, but this is not the case the more difficult VERA 1E problem in Sec. 7.3.

7.3 VERA 1E, pincell with IFBA

The cross sections for VERA 1E at 600 K and 1200 K are shown in Table 7.6. The IFBA layer makes the VERA 1E problem difficult to solve for deterministic methods. The thermal absorption cross section elsewhere in the problem varies from $\Sigma_a \approx 0.0$ in the hydrogen gap to $\Sigma_a \approx 0.15$ in the fuel. In comparison, the IFBA thermal absorption cross section is around $\Sigma_a \approx 18.8$. This strong absorption means the addition of the IFBA, which only represents 0.16 percent of the problem volume, depresses the eigenvalue by over

Table 7.6: VERA 1E cross sections

(a) 600 K

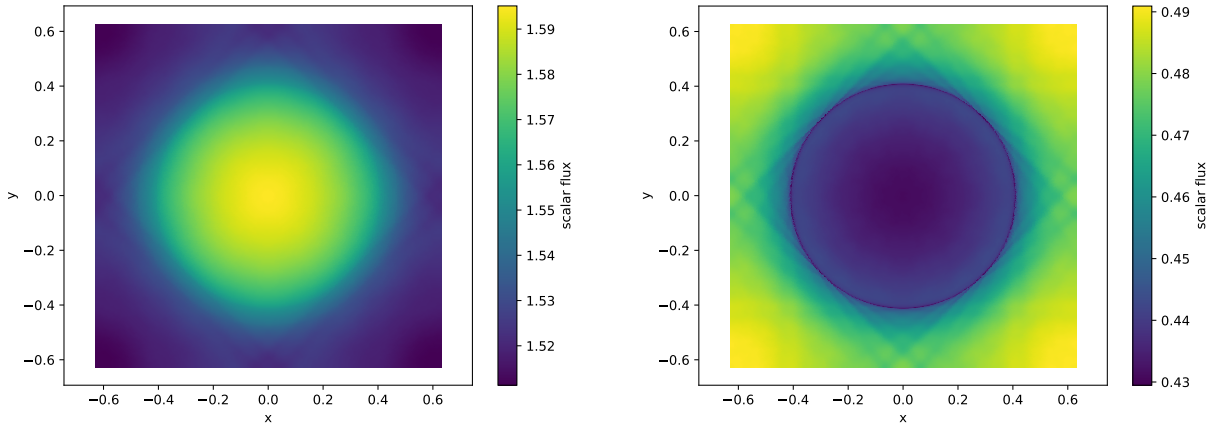
Mat.	g	$\Sigma_{t;g}$	$\Sigma_{s;0,g\to1}$	$\Sigma_{s;0,g\to2}$	$\Sigma_{s;1,g\to1}$	$\Sigma_{s;1,g\to2}$	$\chi\nu\Sigma_{f;g\to1}$	$\chi\nu\Sigma_{f;g\to2}$
Fuel	1	0.398430149	0.382535056	0.000821017	0.049926556	-0.000258533	0.013877940	0.000000015
	2	0.566537670	0.0	0.408203703	0.0	0.006196203	0.208179836	0.000000160
IFBA	1	0.400687673	0.272518929	0.001179693	0.038021479	-0.000343355	0.0	0.0
	2	18.807866148	0.0	0.284174375	0.0	0.009850593	0.0	0.0
Gap	1	0.000060712	0.000060487	0.000000400	0.000011767	-0.000000079	0.0	0.0
	2	0.000021233	0.0	0.000021243	0.0	0.000003495	0.0	0.0
Clad	1	0.318128762	0.316009612	0.000284867	0.052994156	-0.000078583	0.0	0.0
	2	0.296467308	0.0	0.293736124	0.0	0.002164244	0.0	0.0
Mod.	1	0.589533143	0.542576514	0.046780637	0.323899243	0.020173486	0.0	0.0
	2	1.404056519	0.000000034	1.389920798	0.000000034	0.598442391	0.0	0.0

(b) 1200 K

Mat.	g	$\Sigma_{t;g}$	$\Sigma_{s;0,g\to1}$	$\Sigma_{s;0,g\to2}$	$\Sigma_{s;1,g\to1}$	$\Sigma_{s;1,g\to2}$	$\chi\nu\Sigma_{f;g\to1}$	$\chi\nu\Sigma_{f;g\to2}$
Fuel	1	0.401619210	0.385045400	0.000805057	0.049995743	-0.000257614	0.013865154	0.000000015
	2	0.563859308	0.0	0.403512783	0.0	0.005676677	0.208251042	0.000000155
IFBA	1	0.400068065	0.272554529	0.001166271	0.038063324	-0.000339365	0.0	0.0
	2	18.782435513	0.0	0.285776128	0.000000000	0.009928280	0.0	0.0
Gap	1	0.000060758	0.000059793	0.000000401	0.000011277	-0.000000075	0.0	0.0
	2	0.000021811	0.0	0.000021778	0.000000000	0.000003630	0.0	0.0
Clad	1	0.318118514	0.316006772	0.000282946	0.053046979	-0.000074858	0.0	0.0
	2	0.296462951	0.0	0.293734642	0.0	0.002157170	0.0	0.0
Mod.	1	0.589009186	0.542417092	0.046425726	0.323740085	0.019997399	0.0	0.0
	2	1.403414263	0.000000039	1.389298953	0.000000039	0.598440853	0.0	0.0

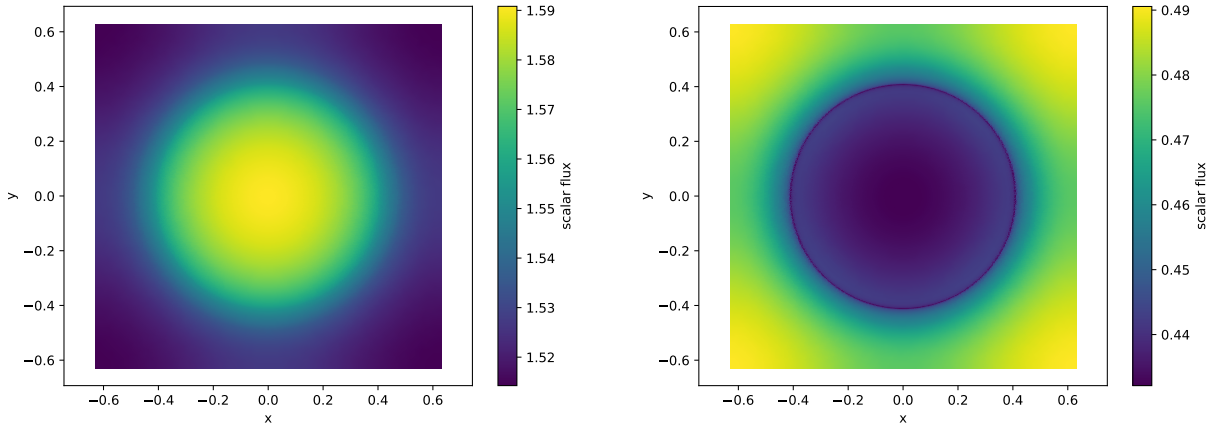
40000 pcm. As such, special attention must be paid to the IFBA region, both in integration and in selecting the locations for the basis and weight function centers. An adaptive quadrature is used for the integration as described in Eq. (4.20), with $N_{rad} = 32$. This guarantees that each basis and weight function is integrated at minimum by around 3000 quadrature points. For the second, scaled set of points described earlier, this also helps ensure that the strong cross section discontinuities at the edges of the IFBA region are accurately integrated. For more information on the integration of this problem, see Sec. 4.5.

The eigenvectors for the fast and thermal groups of VERA 1E show more defined ray effects (Figs. 7.7a and 7.7b) than VERA 1B, particularly in the thermal group. These ray effects, which are discussed in more detail in Sec. 6.3.2, cast shadows along the discrete directions in the locations where rays along these directions travel the furthest through the IFBA region. The ray effects shown for 256 directions are even clearer with 16 or 64 directions (not pictured) but improve with 1024 directions and disappear for 4096



(a) Fast group, 34086 points, 256 directions

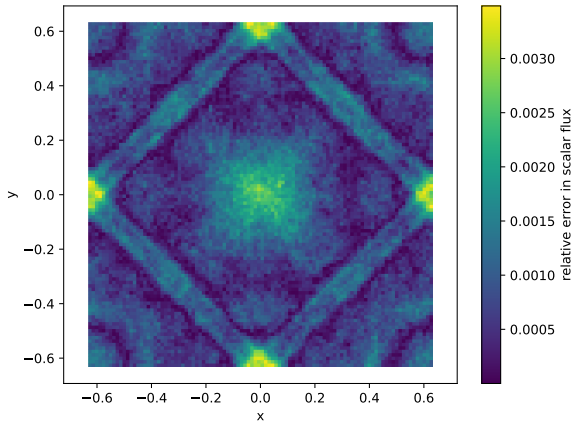
(b) Thermal energy group, 34086 points, 256 directions



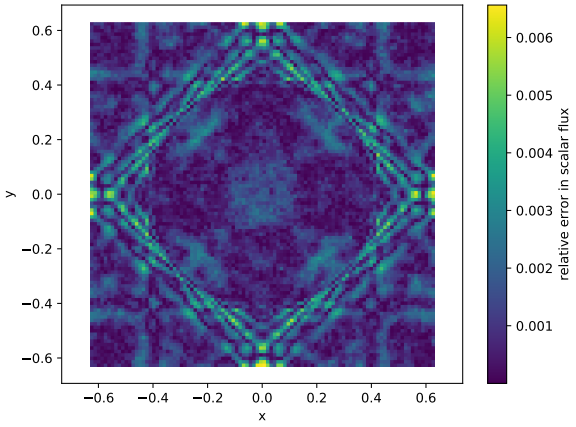
(c) Fast group, 12978 points, 4096 directions

(d) Thermal group, 12978 points, 4096 directions

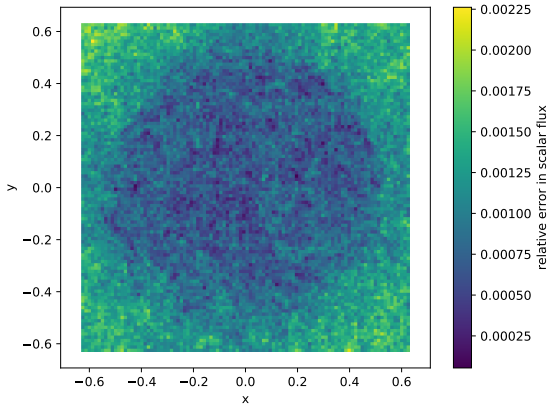
Figure 7.7: VERA 1E eigenvector



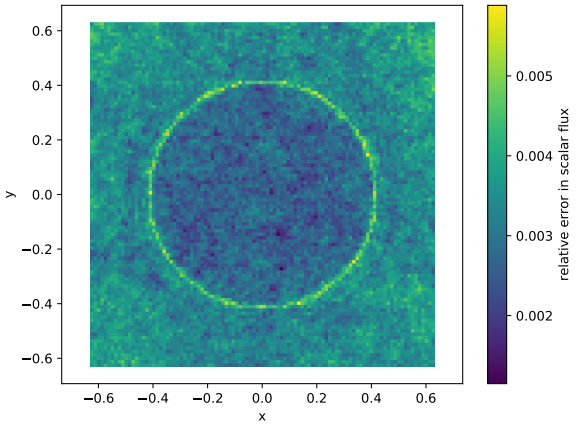
(a) Fast group, 34086 points, 256 directions



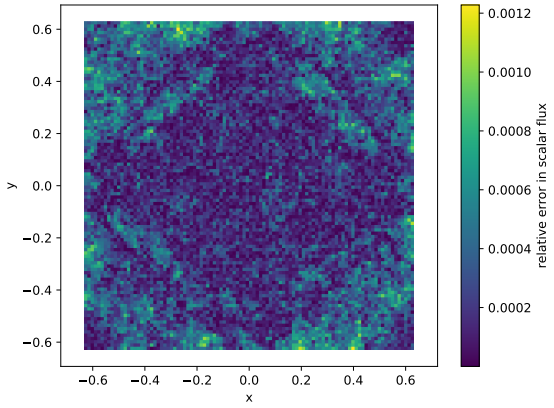
(b) Thermal group, 34086 points, 256 directions



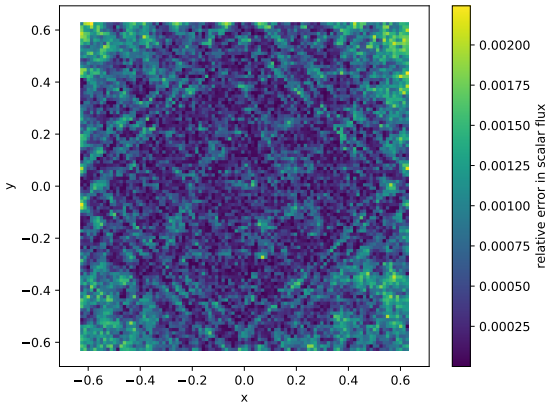
(c) Fast group, 8511 points, 4096 directions



(d) Thermal group, 8511 points, 4096 directions



(e) Fast group, 34086 points, 1024 directions



(f) Thermal group, 34086 points, 1024 directions

Figure 7.8: VERA 1E normalized eigenvector error

directions (Figs. 7.7c and 7.7d). For 4096 directions, the cladding area is axially symmetric for both groups and the ray effects due to the IFBA layer are no longer visible. For 34086 points, the magnitude of the worst-case relative error decreases from around 0.0034 for the fast group and 0.0062 for the thermal group for 256 directions to around 0.0012 for the fast group and 0.0021 for the thermal group.

For a problem that is converged in space but not in angle (34086 points, 256 directions), the largest error in the problem is due to these ray effects (Figs. 7.8a and 7.8b). For the fast group, the rays appear in the cladding region where the neutrons scatter from the fast group into the thermal group, making the cladding region appear to be absorptive in the fast group. The error due to the IFBA region is eclipsed by the error of the ray effects. For a problem that is converged in angle but not in space (8511 points, 4096 directions), error is concentrated at the edge of the fuel cell, where many points are required to converge the solution (Figs. 7.8c and 7.8d). Finally, for a problem with a large number of points and moderate number of directions (34086 points, 1024 directions), minimal ray effects appear but the solution appears to be converged below the background error of the Monte Carlo solution, even in the IFBA region (Figs. 7.8e and 7.8f).

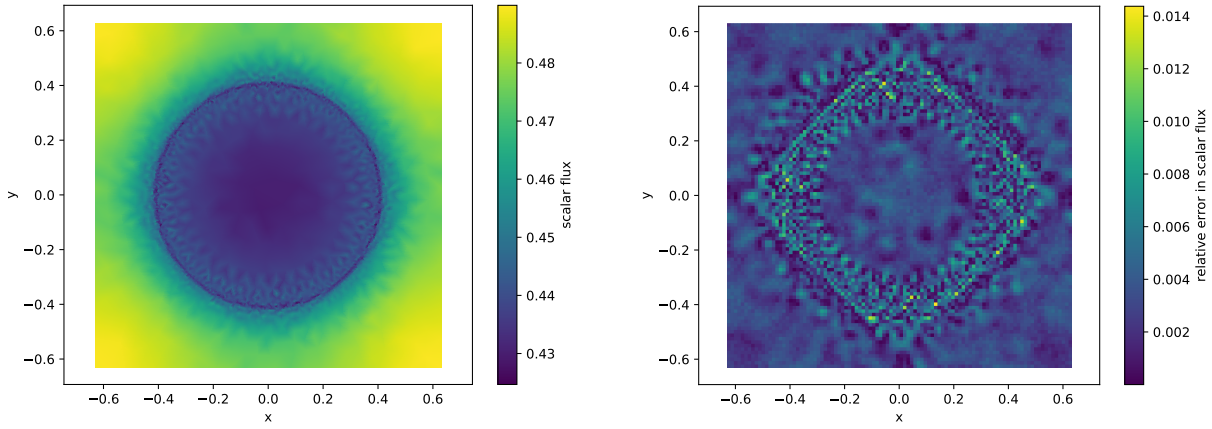
The stated purpose of the SUPG stabilization for the MLPG transport equations is to reduce the oscillations that would otherwise appear in the solution and to improve the conditioning of the problem. Figure 7.9a shows the eigenvector for the thermal group for 3518 points and 256 directions. Clear oscillations appear at the edges of the fuel that propagate into the fuel and cladding. Figure 7.9b shows the error for the thermal group for the same problem. The oscillations around the fuel edge are larger than the ray effects described previously. In order to generate this solution, the $\mathcal{L}^{-1}$ calculation (Sec. 3.3) has to be performed using a direct LU decomposition, as the system is too ill-conditioned to solve using either of the preconditioners. This dramatically increases the memory and computational cost of the solution, and the results are not physically accurate.

7.3.1 Convergence with spatial refinement

Due to the large ray effects in the VERA 1E solution, the spatial convergence study here uses 1024 directions instead of the 256 directions used for VERA 1B. Due to the large number of directions and points used in this problem, the SUPG stabilization parameter is set at 1.5 for this problem to ensure stable convergence. For a value of 1.0, the largest problems do not converge. This may be expected by extrapolating the results of the manufactured radial problem [Fig. 5.5a] to a problem with more directions and a wider range of radii for the basis and weight functions.

The results for both 600 K and 1200 K (Fig. 7.10) show incomplete convergence for the Cartesian points. This discretization is unable to resolve the localized error in the IFBA region for the tested number of points. A set of Cartesian points with spacing of the same size as the IFBA region would have approximately 1.6

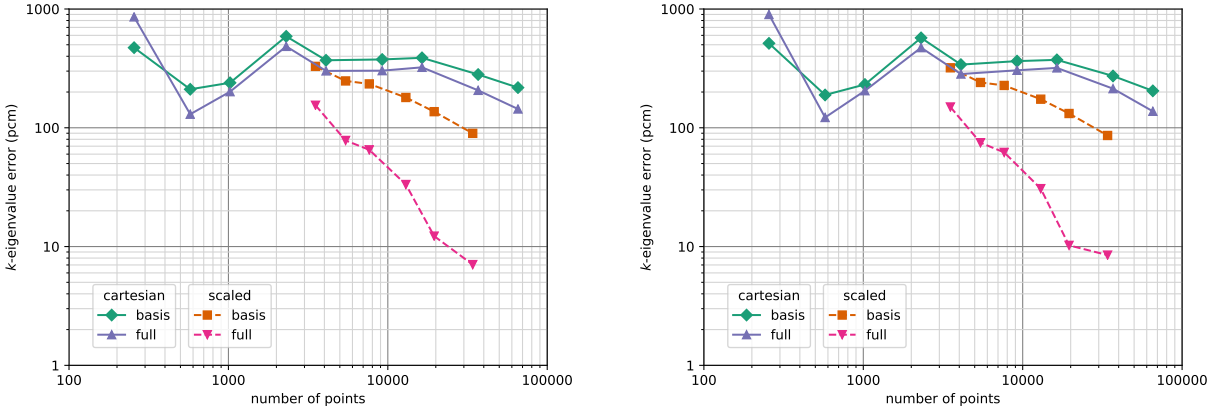
million points. For comparison, the scaled set of points with spacing of the same size as the IFBA region (shown in part in Fig. 4.7) has only 34086 points. The solution for both the 600 K and 1200 K cases doesn't converge below 90 pcm except for the scaled geometry with full cross section weighting, for which the solution converges to 7.0 pcm for 600 K and 8.5 pcm for 1200 K by 34086 points (Table 7.4). The rate of convergence for the last few points is second-order with point spacing for the full cross section method with the scaled geometry and around first-order for the other cases, which reflects the localized nature of the error. The relative L_2 error for 34086 points and 1024 directions for both the 600 K and 1200 K results is around 0.0003 for the fast group and 0.0007 for the thermal group, compared to the mean standard deviation of the Monte Carlo multigroup benchmark values of 0.0004 for the fast and 0.0003 for the thermal group.



(a) Eigenvector for thermal group

(b) Error in eigenvector for thermal group

Figure 7.9: VERA 1E eigenvector without SUPG stabilization, 3518 points, 256 directions



(a) 600 K

(b) 1200 K

Figure 7.10: VERA 1E convergence of k-eigenvalue with spatial refinement, 1024 directions

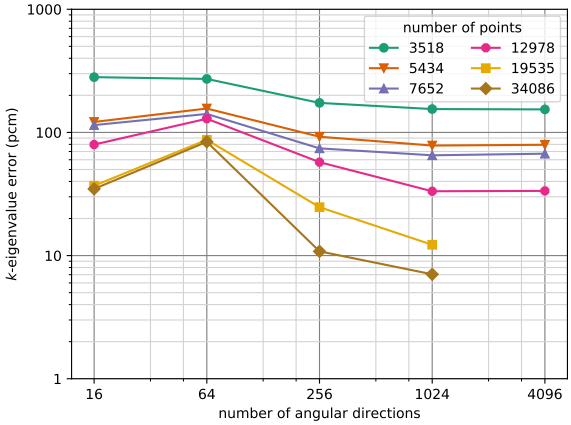


Figure 7.11: VERA 1E convergence of k-eigenvalue with angular refinement, 600 K

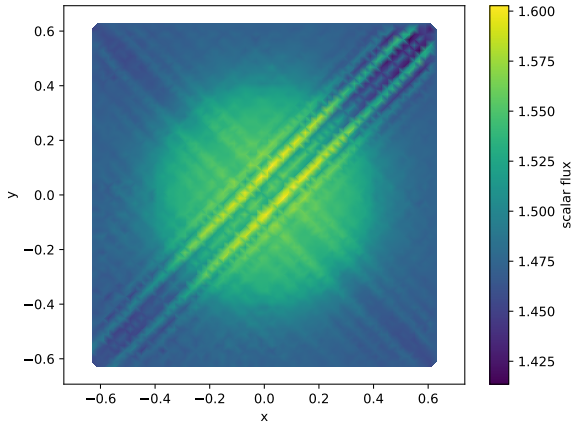
7.3.2 Convergence with angular refinement

To show the effect of the number of angles on the solution, an angular convergence for between 16 and 4096 directions and fixed values of points between 3518 and 34086 points is performed (Fig. 7.11). While a larger number of points reduces the error near the fuel boundary, a larger number of angles reduces the ray effects discussed previously. For 34086 points, the error decreases from 83 to 10 pcm from 64 to 256 directions and then to 7 pcm at 1024 directions.

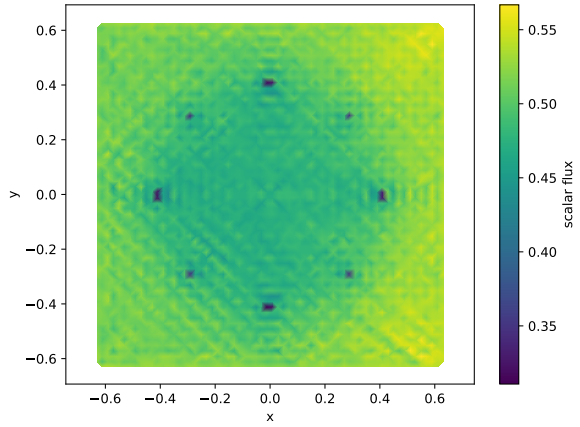
The convergence with the number of angular directions strongly depends on the number of points. For 3518 points, there is little benefit in using more than 256 directions. There is, however, a large benefit in increasing the number of directions at and over 12978 points. The angular error represents a much larger percentage of the total error compared to the spatial discretization error for a large number of points. The error decreases by about 70 pcm from 64 to 256 directions and by 10-20 pcm from 256 to 1024 directions for all numbers of points. For the cases with the most points, memory constraints (see Sec. 3.6) prevented testing 4096 directions.

7.3.3 Strong-form discussion

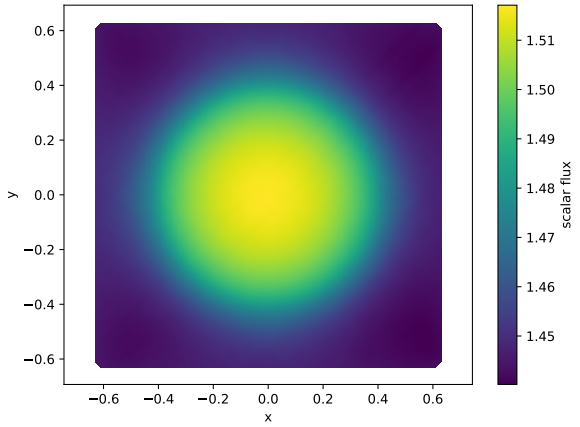
As mentioned in Sec. 5.3, the strong-form equations do not perform well for problems with discontinuous cross sections. For the VERA 1E problem with 4096 points and 256 directions, the point and basis cross section weighting (see Sec. 2.9) have 2953.5 and 16068.6 pcm error, respectively. For the same set of points, the basis and full cross section weighting for the weak form with SUPG stabilization have 376.1 and 302.6 pcm error, respectively. The requirement to use an LU decomposition instead of a preconditioned iterative solve for the $\mathcal{L}^{-1}$ operator means that the strong-form solution for a problem of this size requires a similar amount of computational time to the weak-form solution, at around 250 seconds on four processors. For



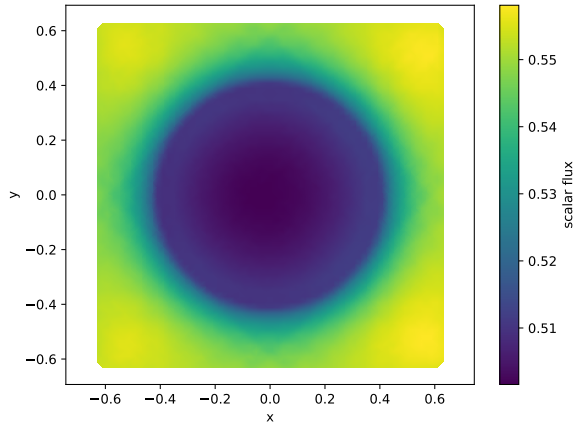
(a) Fast group, point cross section weighting



(b) Thermal energy group, point cross section weighting

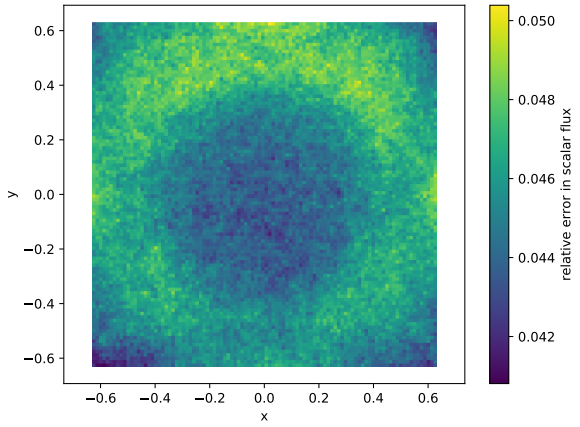


(c) Fast group, basis cross section weighting

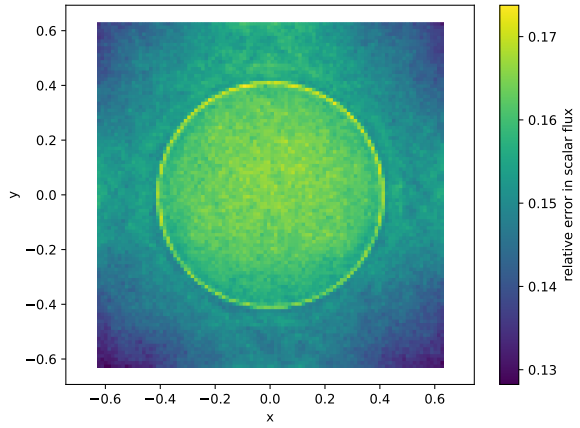


(d) Thermal group, basis cross section weighting

Figure 7.12: VERA 1E eigenvector for strong-form solution, 4096 points, 256 directions



(a) Fast group



(b) Thermal energy group

Figure 7.13: VERA 1E normalized eigenvector error for strong-form solution, 4096 points, 256 directions, basis cross section weighting

larger problems, the strong-form solution is much more expensive than the weak-form solution due to the poor scaling of the LU decomposition.

Figure 7.12 shows what happens when the strong-form collocation equations with each weighting type are applied to the IFBA pincell problem. As might be expected, the point-evaluated cross sections do not accurately represent the problem geometry. For the point cross section weighting, the basis functions that happen to be centered inside of the IFBA region simulate a disproportionate amount of absorption, while the basis functions with centers that are near the IFBA region but not inside of it do not simulate any of the expected absorption. For both the fast and thermal groups, the solution oscillates throughout the domain.

The basis function cross section weighting performs much better than the point cross sections and actually removes the visible oscillations in the solution. The IFBA layer in the solution is far wider and less prominent than for the weak-form solution (Fig. 7.7). Figure 7.13 shows the error in the normalized eigenvector for basis function weighting. The solution accurately models the axial symmetry of the pincell but does not resolve the IFBA layer. As a result, the IFBA absorption is broadened spatially and results in more absorption at the fuel edge and a lower flux inside the pincell for the thermal group than is physically accurate. While the basis function weighting does not resolve small details of the solution accurately, it is possible that for problems with simpler geometry the strong-form equations with basis function weighting could be used successfully.

7.4 Three-dimensional reflected ellipsoid

The final eigenvalue problem is a three-dimensional reflected ellipsoid centered at the origin. The ellipsoid has semi-axes of length $a_x = 4.3652$ cm, $a_y = 5.2382$ cm and $a_z = 6.5478$ cm, while the rectangular cuboid containing the ellipsoid has side lengths of $\ell_x = 20.2284$ cm, $\ell_y = 24.2741$ cm and $\ell_z = 30.3426$ cm (Fig. 7.14). The material isotopics are based off a critical thorium-reflected plutonium sphere at Los Alamos National Laboratory [66], which has the International Criticality Safety Benchmark Evaluation Project (ICSBEP) [67] evaluation identifier PU-MET-FAST-008. The thorium reflects some of the fission neutrons back to the plutonium, creating a near-critical system. The boundaries for the problem are vacuum. The multigroup cross sections and benchmarks are generated using the same procedure as in Sec. 7.1, but using 10^7 particles

Table 7.7: Ellipsoid cross sections

Mat.	Σ_t	$\Sigma_{s;0}$	$\Sigma_{s;1}$	$\nu\Sigma_f$
Pu	0.322644801	0.255040685	0.094644991	0.201891561
Th	0.290248656	0.284558789	0.060957181	0.000973016

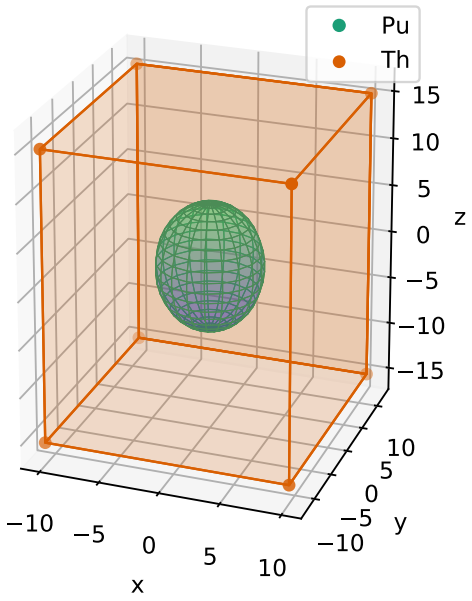


Figure 7.14: Ellipsoid geometry

per generation and an eigenvector tally of $20 \times 24 \times 30$ Cartesian cells. Due to the lack of thermal neutrons in the problem, a single group is used for the cross sections (Table 7.7). The L_2 error is calculated using the same procedure as in Sec. 7.1 over the same $20 \times 24 \times 30$ Cartesian mesh used for the Monte Carlo tally. The problem parameters are shown in Table 7.1. The multigroup and continuous-energy solutions do not agree as well for this problem as for the other eigenvalue problems, at 1172.2 pcm difference (Table 7.2). This may be because of the one-group assumption. For a solution that is closer to the continuous-energy solution, more energy groups would be required.

One set of centers for this problem is a Cartesian grid with constant spacing, similar to the Kobayashi problem (Sec. 6.3). Due to the ratios of the side lengths of the bounding cuboid, this results in approximately $N_x = \frac{5}{6}N_y = \frac{2}{3}N_z$ points in each dimension. The second set of centers is similar to the scaled set from the VERA problems (Sec. 7.1), with the same methodology to determine the radii at which rings of points are placed. Instead of concentric circles at each radius as for the 2D VERA problem, the ellipsoid problem uses concentric ellipsoids. Unlike for a circle, there does not exist a general method for placing points on a sphere or ellipsoid with exactly even spacing . One near-optimal method to produce evenly-spaced points on the sphere is the method of Fibonacci grids [68]. To apply the spherical method to the ellipsoidal problem, points are generated for the unit sphere with appropriate angular density and then mapped onto the ellipsoid. The starting point on the sphere for the point generation is randomized. The result is a set of concentric ellipsoids with high point density near the plutonium-thorium boundary. As in the VERA problems, a set of Cartesian points is placed on the surface of the cuboid boundary.

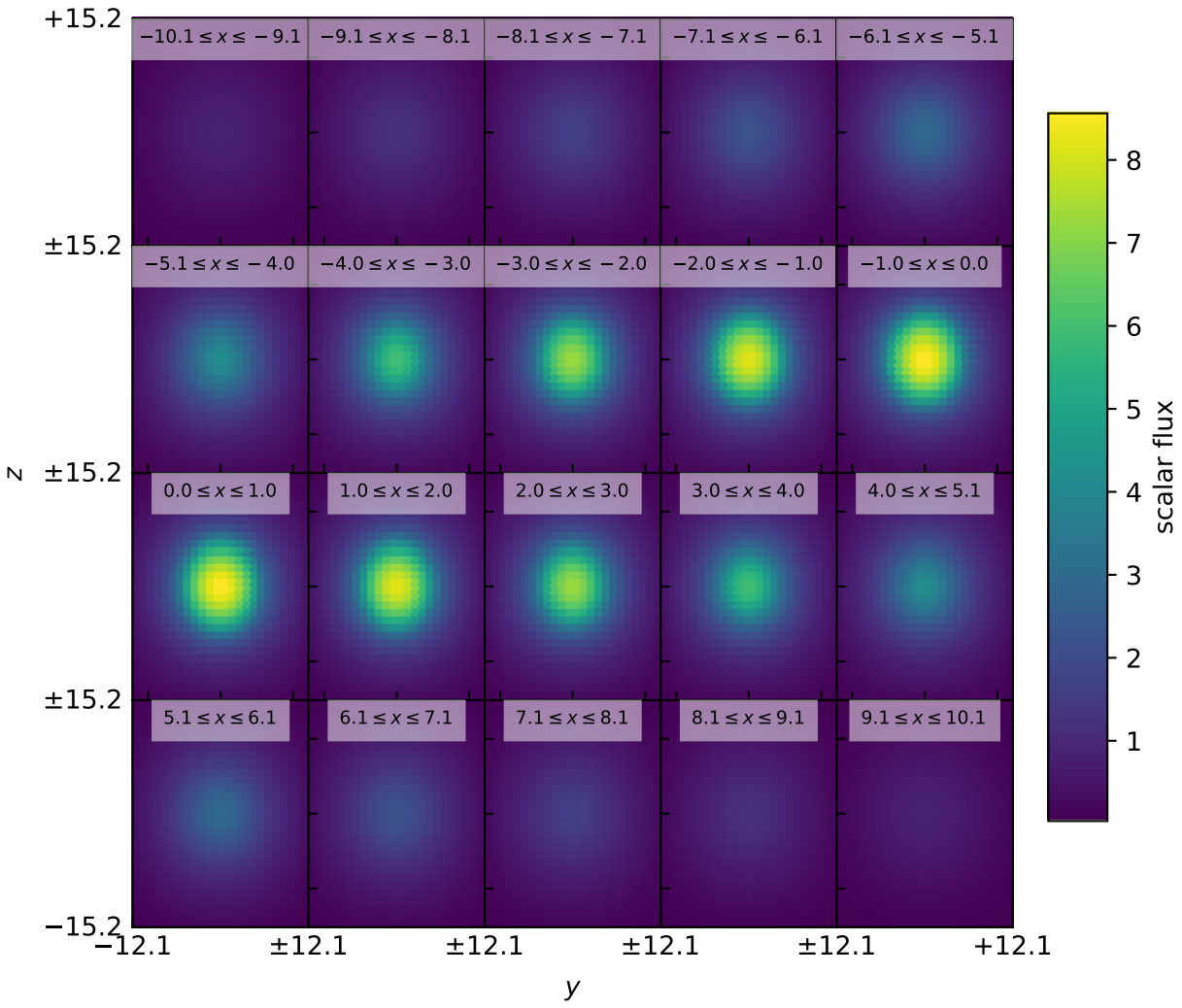


Figure 7.15: Ellipsoid eigenvector averaged over a 20x24x30 mesh, 58368 Cartesian points

The normalized eigenvector over the $20 \times 24 \times 30$ integration mesh in Fig. 7.15 shows a strong gradient in the neutron flux near the boundaries of the plutonium sphere. As might be expected from the results of the pincell problems, the error of the eigenvector compared to the Monte Carlo solution is largest at the edges of the plutonium sphere near the gradient in the solution and the material discontinuity (Fig. 7.16).

The convergence results in Fig. 7.17 show the k-eigenvalue and L_2 errors for both the basis and full cross section methods with the Cartesian and scaled meshes. The relative L_2 error is expected to be approximately 10^5 higher than the k-eigenvalue due to the units of pcm in the latter. The results for each cross section weighting type show that, adjusting for units, the k-eigenvalue and L_2 error agree well for this problem. The convergence rate is second-order with point spacing for all cases considered. The error of the k-eigenvalue is much larger, even for the largest problems (88.5 pcm at 40078 points for the scaled point geometry), than the standard deviation of the Monte Carlo solution of 2.1 pcm (Tabs. 7.4 and 7.3). The L_2 error for 58368

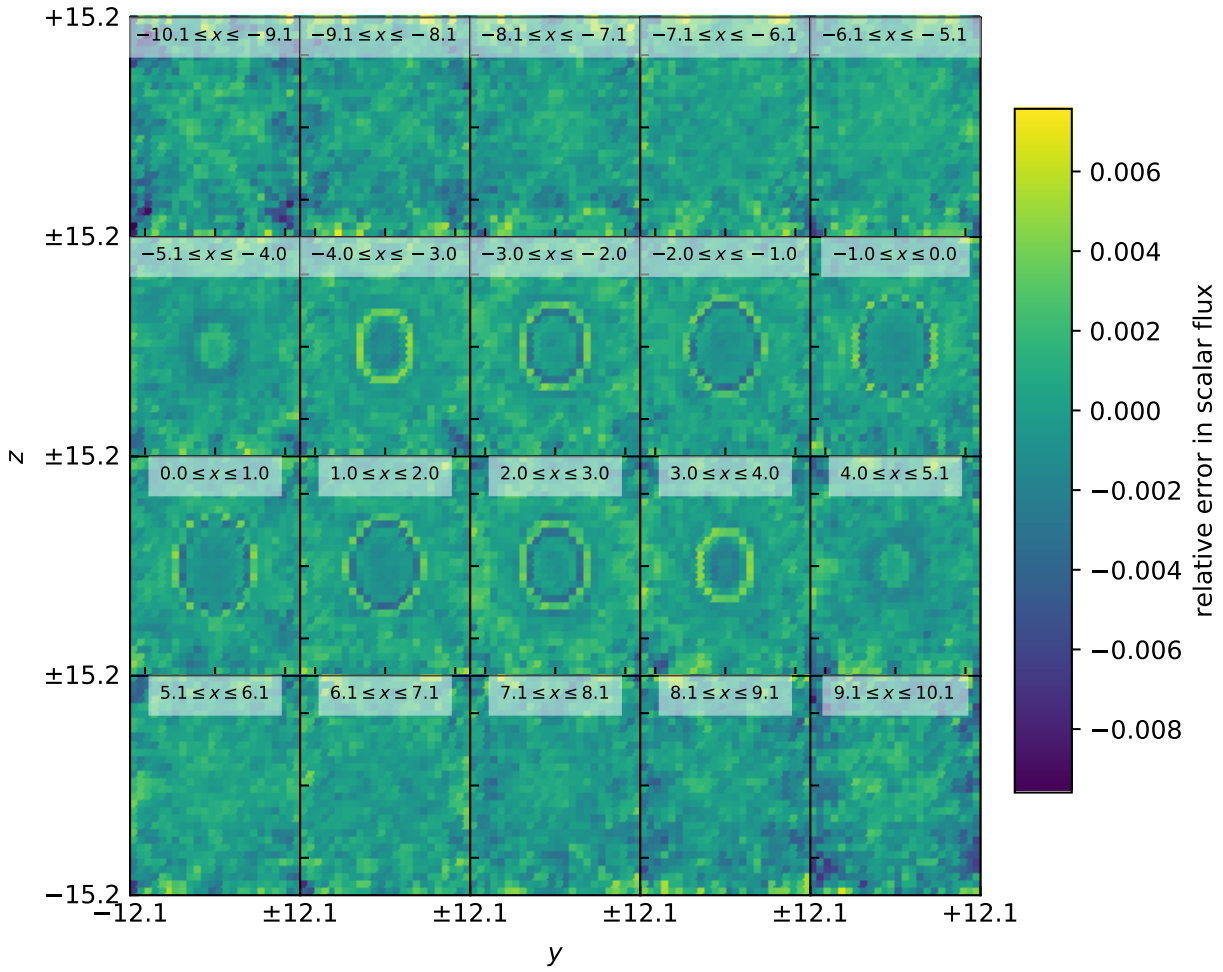
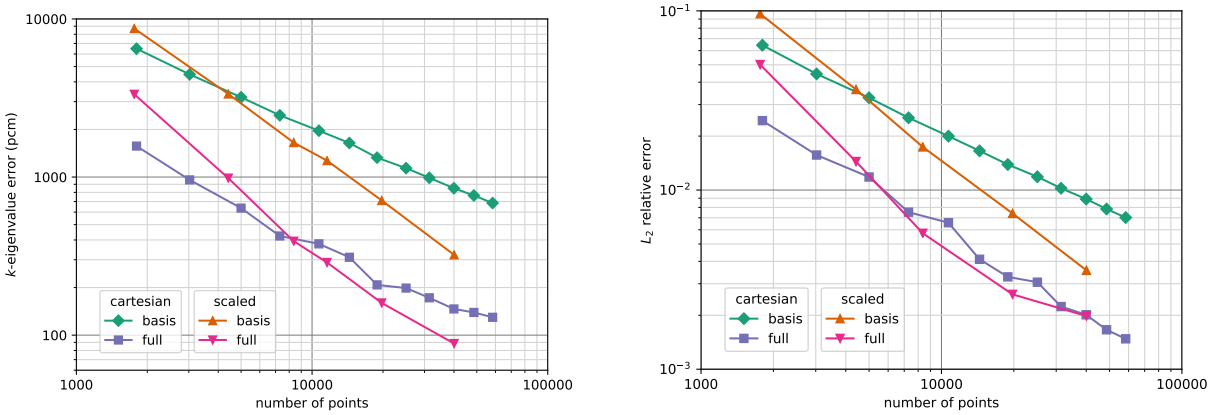


Figure 7.16: Ellipsoid relative error over a 20x24x30 mesh, 58368 Cartesian points



(a) k-eigenvalue error

(b) Relative L_2 error

Figure 7.17: Ellipsoid convergence with spatial refinement

Cartesian points of 0.0015 is 2.4 times higher than the mean standard deviation and 1.5 times lower than the maximum standard deviation for the Monte Carlo tallies. Further convergence of the solution is expected, but further refinement of the spatial solution is not performed here due to the high memory cost of running large simulations in three dimensions (see Sec. 3.6). The results for the largest number of points considered would have similar point density to a 1000-point solution in two dimensions, which for the IFBA pincell has more than 100 pcm error. The inherent scaling issues in three dimensions highlight the need for further performance improvements of the MLPG transport method, as is discussed in Sec. 9.3.

Chapter 8

Coupling of meshless heat transfer to meshless neutron transport

The heat conduction equation is a statement of energy conservation, specifically that the rate of change in the kinetic energy is proportional to the rate at which the energy is added into the system minus the rate at which the energy is removed from the system. The main mechanism of energy transfer in this equation is conduction, which is the transfer of energy by collisions between adjacent particles. The boundary conditions for the heat conduction equation can include convection, which is the transfer of heat by bulk motion of a fluid. At a convective boundary surface, heat is transferred to or from the fluid (depending on whether the fluid is cooler or warmer than the boundary surface, respectively) by conduction. The motion of the fluid moves these fluid particles in contact with the convective boundary continuously, which increases the efficiency of the heat transfer.

The heat equation has been successfully solved before using meshless methods, including by collocation [69] and using MLPG [70], which is the discretization used here for the heat transfer equations. For an overview of the application of meshless methods to heat transfer, see Ref. [71]. Due to the design requirements of nuclear reactors, the coupling of heat transfer and neutron transport is common. In environments such as CASL [72] and MOOSE [73, 74], the neutron transport is coupled to full thermal hydraulics codes, which adds important physics that the heat conduction code only approximates.

This section presents a loose coupling of the heat transfer equations and the neutron transport equations. The main difference between the methods presented in this section and previous approaches is the use of meshless functions to both solve the equations and transfer data between the heat transfer and neutron transport codes. The MLPG discretization of the heat transfer code is derived in Sec. 8.2. In Sec. 8.3 the code is verified by comparison to an analytic solution and by the method of manufactured solutions (MMS), similar to the MMS verification of neutron transport in Chapter 5. Finally, the coupling to neutron transport is described in Sec. 8.4 and implemented for a pincell problem in Sec. 8.5.

8.1 Heat conduction equation

The heat conduction equation is defined as

$$-\rho c_p \frac{\partial T}{\partial t} - \boldsymbol{\nabla} \cdot (k \boldsymbol{\nabla} T) = \dot{q}, \tag{8.1}$$

where c_p is the specific heat at constant pressure, ρ is the mass density, k is the thermal conductivity and $\dot{q}$ is the volumetric heat generation rate [75]. Each of the three terms in the equation has units of energy per time (e.g. watts per second). The time derivative term implies that the change in the temperature due to a change in energy is proportional to the density and specific heat of the material. The diffusion term states that the energy change due to convection for a certain region of the problem depends on the energy flow into and out of that region.

The convective boundary condition depends on the difference in the material temperature at the surface, T, and the fluid temperature T_∞,

$$k \boldsymbol{n} \cdot \boldsymbol{\nabla} T = h \left[T_\infty - T \right], \tag{8.2}$$

where $\boldsymbol{n}$ is the surface normal for the boundary surface. This equation describes the net direction of energy transfer for a convective boundary, from the hotter to the colder material. The convective boundary condition can be either an energy source to the system when $T > T_\infty$ or a sink when $T < T_\infty$. For a reflective boundary condition (i.e. no heat transfer in or out of a surface), the convection coefficient h can be set to zero.

The steady-state equation assumes that the temperature does not depend on time, which simplifies Eq. (8.1) to

$$-\boldsymbol{\nabla} \cdot (k \boldsymbol{\nabla} T) = \dot{q}. \tag{8.3}$$

Taken together with the boundary condition in Eq. (8.2), this steady-state equation states that the energy sources and sinks are balanced. Specifically, the energy flow out of any region in the problem equals the energy flow into and the generation rate inside of that region.

8.2 Heat equation discretization

The heat equation is discretized using the MLPG method, similarly to the neutron transport equation in Secs. 2.3 to 2.5 but with identical basis and weight functions. As the convection equation is a diffusion instead of an advection equation, no stabilization is required to prevent oscillations in the solution.

To derive the weak form of the heat equations, the steady-state conduction equation [Eq. (8.3)] is

multiplied by a series of test functions w_j and integrated over the support region for the test function, V_j,

$$-\int_{V_j} w_j \boldsymbol{\nabla} \cdot (k \boldsymbol{\nabla} T)\, dV = \int_{V_j} w_j \dot{q} dV, \qquad j = 1, \ldots, J. \tag{8.4}$$

The conduction term is integrated by parts to separate the integral into surface and volume-integrated terms,

$$-\int_{S_j} w_j \boldsymbol{n} \cdot (k \boldsymbol{\nabla} T)\, dS + \int_{V_j} (\boldsymbol{\nabla} w_j) \cdot (k \boldsymbol{\nabla} T)\, dV = \int_{V_j} w_j \dot{q} dV, \qquad j = 1, \ldots, J. \tag{8.5}$$

The integrand of the surface integral in Eq. (8.5) contains the gradient term in the convective boundary condition [(8.2)]. This term inside the surface integral is replaced by the boundary condition,

$$-\int_{S_j} w_j h \left[T_\infty - T(\boldsymbol{x})\right] dS + \int_{V_j} (\boldsymbol{\nabla} w_j) \cdot (k \boldsymbol{\nabla} T)\, dV = \int_{V_j} w_j \dot{q} dV, \qquad j = 1, \ldots, J. \tag{8.6}$$

Separating the surface integral into the known and unknown portions gives the final weak form of the heat equation,

$$\int_{S_j} w_j h T dS + \int_{V_j} (\boldsymbol{\nabla} w_j) \cdot (k \boldsymbol{\nabla} T)\, dV = \int_{V_j} w_j \dot{q} dV + \int_{S_j} w_j h T_\infty dS, \qquad j = 1, \ldots, J. \tag{8.7}$$

All of the terms containing the unknown temperature distribution T are on the right side of the equation. To solve for the temperature, a basis function expansion

$$T(\boldsymbol{x}) = \sum_{i=1}^{J} \zeta_i b_i(\boldsymbol{x}) \tag{8.8}$$

with the expansion coefficients ζ_i and basis functions b_i is inserted into Eq. (8.7) to get

$$\sum_i \left[\int_{S_j} w_j h b_i dS + \int_{V_j} (\boldsymbol{\nabla} w_j) \cdot (k \boldsymbol{\nabla} b_i)\, dV \right] \zeta_i = \int_{V_j} w_j \dot{q} dV + \int_{S_j} w_j h T_\infty dS, \qquad j = 1, \ldots, J. \tag{8.9}$$

Equation (8.9) represents a linear system of equations

$$\boldsymbol{A\zeta} = \boldsymbol{s} \tag{8.10}$$

with the matrix and source vector of

$$A_{i,j} = \int_{S_j} h w_j b_i dS + \int_{V_j} (\boldsymbol{\nabla} w_j) \cdot (k \boldsymbol{\nabla} b_i)\, dV, \tag{8.11a}$$

$$s_j = \int_{V_j} w_j \dot{q} dV + \int_{S_j} w_j h T_\infty dS. \tag{8.11b}$$

Unlike the transport equation, this equation contains no unknowns in the source term. Because of this, only a single linear solve is required to solve the system.

For cylindrical geometry without axial or longitudinal dependence, the boundary condition at $r = 0$ is reflective,

$$\frac{\partial T}{\partial r} = 0 \Big|_{r=0}, \tag{8.12}$$

and the heat transfer equation simplifies to

$$\left[r w_j h T\right]_{r=R} + \int_{V_j} \left(\frac{\partial w_j}{\partial r}\right) \left(k \frac{\partial T}{\partial r}\right) r dr = \int_{V_j} r w_j \dot{q} dr + \left[r w_j h T_\infty\right]_{r=R}, \tag{8.13}$$

where R is the outer radius of the cylinder. The linear system from Eqs. (8.11) becomes

$$A_{i,j} = \left[r w_j h b_i\right]_{r=R} + \int_{V_j} \left(\frac{\partial w_j}{\partial r}\right) \left(k \frac{\partial b_i}{\partial r}\right) r dr, \tag{8.14a}$$

$$s_j = \int_{V_j} r w_j \dot{q} dr + \left[r w_j h T_\infty\right]_{r=R}. \tag{8.14b}$$

To solve the heat transfer equations, the basis and weight functions are defined as in Sec. 2.10 to be compact. The volume and surface integrals in Eqs. (8.11) or Eqs. (8.14) are performed using a background mesh as described in Sec. 3.2. The convection and conduction coefficients, convective fluid temperature and heat source are assumed to be spatially-dependent. The linear system is then solved using a direct or iterative sparse linear algebra solver as described for the transport equation in Sec. 3.3.

8.3 Heat transfer verification

To verify that the heat transfer discretization appropriately solves the equations, two separate methods are employed:

1. An analytic solution to a slab-geometry problem (Sec. 8.3.1) and
2. A manufactured solution with dimensional dependence (Sec. 8.3.2).

For each case, the discretized system is solved using the Wendland 11 and Wendland 33 functions (Sec. 2.10) with a variable number of neighbors (Alg. 3.3) and linear MLS functions. The convergence of each method

is measured using the relative L_2 integral error,

$$\epsilon_{L_2} = \sqrt{\frac{\int_V \left(T - T_{ana}\right)^2 dV}{\int_V T_{ana}^2 dV}}. \tag{8.15}$$

As in Sec. 8.2, the integrals are performed using a background mesh.

8.3.1 Analytic solution

Analytic solutions of the heat equation can be derived for some simple problems. This slab-geometry problem over a rectangular cuboid has symmetry in the y and z axes. The limits of the cuboid are $-0.4 \leq x \leq 0.2$ and $-0.15 \leq y, z \leq 0.15$. The problem is tested in one, two and three dimensions with a Cartesian grid of points. For two and three dimensions, the number of points in the y and z directions is chosen to be half the number in the x direction due to the smaller length of the cuboid in these dimensions. The solution along the x axis does not change based on the dimensionality of the problem. For the discretized heat transfer equations, this corresponds to reflective boundaries ($h = 0$) on the boundaries with normal directions parallel to the y and z axes.

The source for the problem is

$$q\left(\boldsymbol{x}\right) = 100 + 200\sin^2\left(4x\right). \tag{8.16}$$

The material coefficients shown in Table 8.1 are piecewise constant from $-0.4 \leq x < 0$ and $0 \leq x \leq 0.2$ and dependent on only x. The slab-geometry problem with the given source and coefficients is solved analytically in Mathematica [61] for comparison against the numerical solution. The solution for two dimensions is shown in Fig. 8.1.

Figure 8.2 shows the spatial dependence of the error for one and two dimensions with 192 points along the x axis in the Cartesian grid of points. The figure shows that the error is approximately the same in one and two dimensions for a similar point spacing. The error is concentrated near the discontinuity in the conduction coefficient at $x = 0.0$. The convergence results for one, two and three dimensions in Figs. 8.3a, 8.3b and 8.3c, respectively, show only first-order convergence with the basis and weight center spacing Δx.

Table 8.1: Analytic heat transfer material properties

Region	k	h	T_{inf}
$-0.4 \leq x < 0$	0.02	0.2	700
$0 \leq x \leq 0.2$	0.2	1.5	400

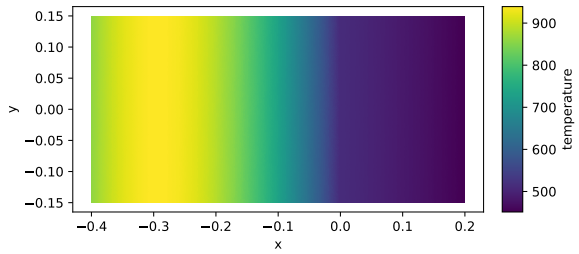


Figure 8.1: Solution of slab-geometry heat transfer problem

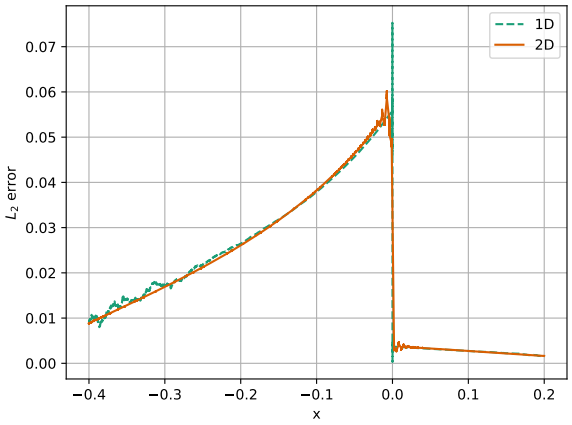
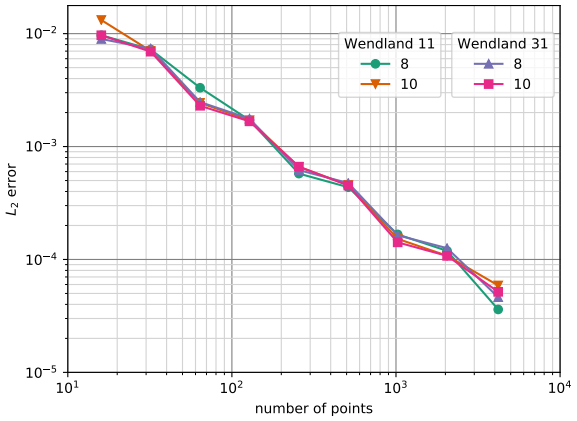
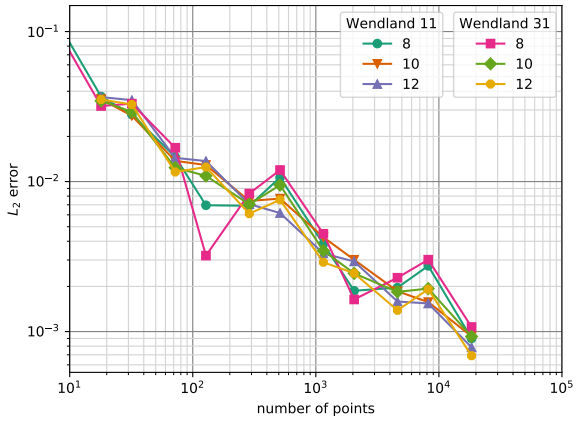


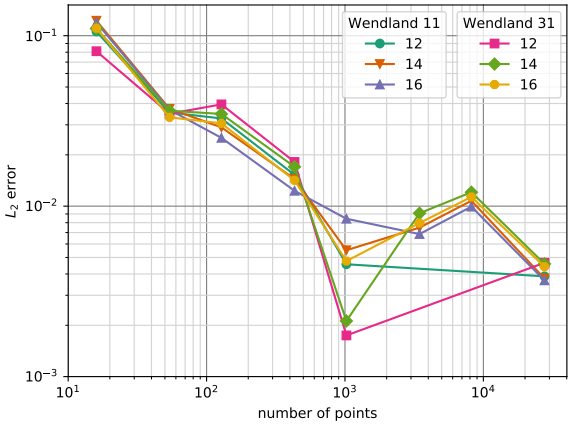
Figure 8.2: L_2 error at points along the x axis for the slab-geometry heat transfer solution in 1D and 2D using 192 points along x axis



(a) 1D



(b) 2D



(c) 3D

Figure 8.3: Convergence to analytic solution with spatial refinement for slab-geometry heat transfer problem

While the method with linear MLS functions should have second-order convergence for smooth coefficients and results (as for the manufactured results in Sec. 8.3.2), the discontinuity in the material coefficients causes a drop in the order of convergence. The smallest errors are around 3×10^{-5} in 1D for the Wendland 11 function with 8 neighbors and 4196 points, 6×10^{-4} in 2D for the Wendland 31 function with 12 neighbors and 18432 points, and 3×10^{-3} in 3D for the Wendland 11 function with any number of neighbors and 27648 points.

8.3.2 Manufactured solution

The method of manufactured solutions can be applied to the heat equation similarly to the transport equation (see Chapter 5). The solution for the temperature is assumed to be a known quantity, $T_m(\boldsymbol{x})$. With this solution fixed, the source from Eq. (8.3) becomes

$$\begin{aligned} \dot{q} &= -\boldsymbol{\nabla} \cdot (k \boldsymbol{\nabla} T_m) \\ &= -(\boldsymbol{\nabla} k) \cdot (\boldsymbol{\nabla} T_m) - k \nabla^2 T_m. \end{aligned} \tag{8.17}$$

For the boundary condition to satisfy the manufactured solution, the convective fluid temperature at the boundary is assumed to be variable spatially. The boundary condition is solved for the convective temperature given the manufactured solution,

$$T_\infty = \frac{k \boldsymbol{n} \cdot \nabla T_m}{h} + T_m. \tag{8.18}$$

In order to calculate the internal source and boundary temperature in Eqs. (8.17) and (8.18), the temperature should be function with at least a nonzero first derivative. If the conduction coefficient k is spatially-dependent, the derivative of k is also required. The conduction coefficient h may also be spatially dependent. See Ref. [76] for more details on appropriate choices for the temperature and coefficients for verification of the heat transfer code by the MMS. After the numerical solution is calculated using the calculated source and boundary temperature, the L_2 error from the manufactured solution is calculated using Eq. (8.15).

The manufactured solution and conduction and convection coefficients are chosen to be

$$T(\boldsymbol{x}) = 1.2 + \sin\left(x^2 + y^2 + z^2\right), \tag{8.19a}$$

$$k(\boldsymbol{x}) = 12.1 - \left(x^2 + y^2 + z^2\right), \tag{8.19b}$$

$$h(\boldsymbol{x}) = 1.1 + \sin(0.5xyz), \tag{8.19c}$$

with a domain of $-2 \leq \boldsymbol{x} \leq 2$. The temperature, conduction coefficient and heat transfer coefficient do not

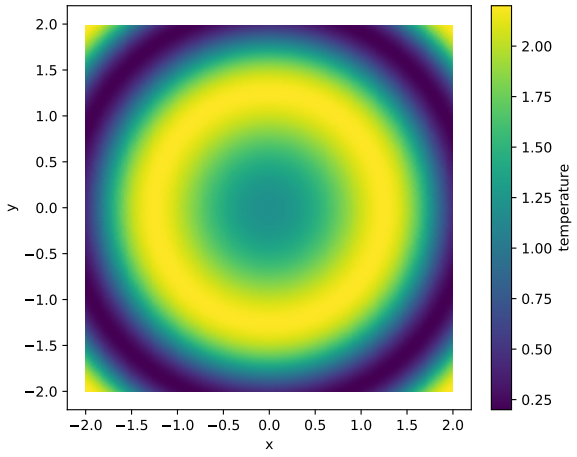
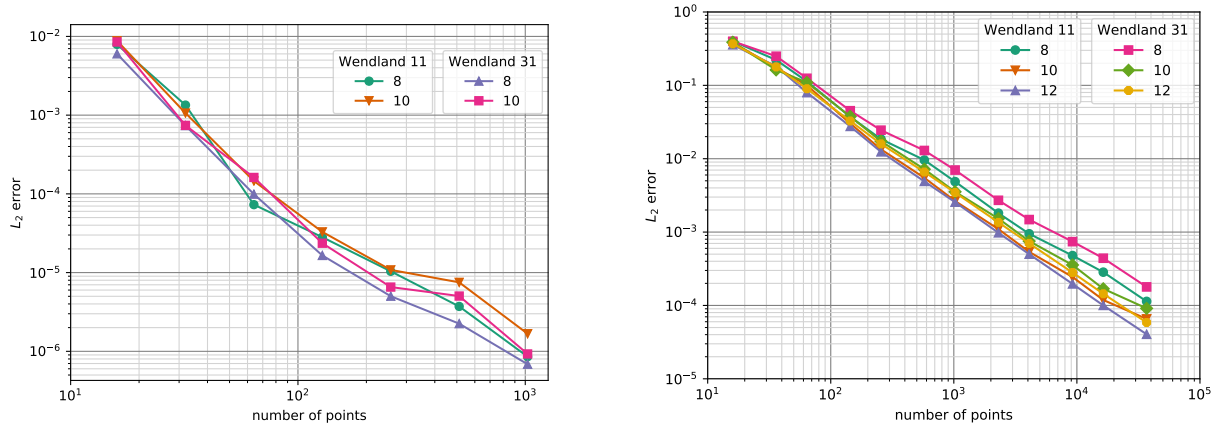
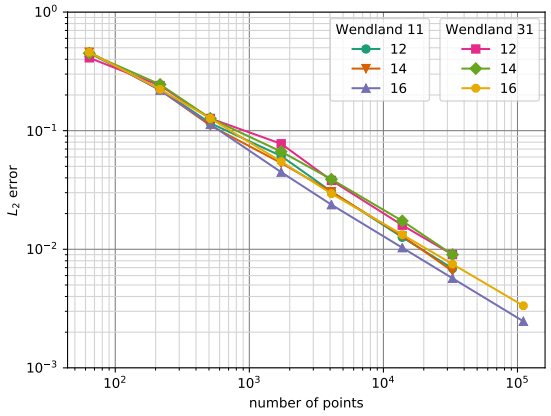


Figure 8.4: Solution of manufactured heat transfer problem



(a) 1D

(b) 2D



(c) 3D

Figure 8.5: Convergence to analytic solution with spatial refinement for manufactured heat transfer problem

go negative anywhere in the domain. The manufactured solution for the temperature is shown in Fig. 8.4. The same convergence study as in Sec. 8.3.1 is done for the linear MLS functions created using Wendland 11 and Wendland 31 functions.

The convergence for 1D (Fig. 8.5a), 2D (Fig. 8.5b) and 3D (Fig 8.5c) is second-order with point spacing for all cases considered. Unlike for the slab solution, the convergence is monotonic for all cases. For the 3D problems, he cases with 12 and 14 neighbors and more than 32768 points experienced ill-conditioning issues that prevented the solver from completing the simulation. The cases with 16 neighbors worked up to the maximum number of points tested, 110592. The smallest errors are around 7×10^{-7} in 1D for the Wendland 31 function with 8 neighbors and 1024 points, 4×10^{-5} in 2D for the Wendland 11 function with 12 neighbors and 36864 points, and 2×10^{-3} in 3D for the Wendland 11 function with 16 neighbors and 110592 points.

8.4 Coupling of heat conduction and neutron transport equations

Heat conduction and neutron transport are strongly linked in nuclear reactors. The fission of the nuclear fuel acts as an energy source that affects the temperature of the fuel. The heat from the fuel is transferred through through the gap and cladding of the fuel pin, after which convection by the water moves the heat away from the fuel pin. In the reverse direction, the temperature of the fuel, gap and cladding affects the cross sections for neutron transport as a result of Doppler broadening. In Sec. 8.5, the two-way coupling presented in this section is applied to a realistic pincell problem.

8.4.1 Temperature dependence of the cross section

The macroscopic cross section Σ is related to the microscopic cross section σ as

$$\Sigma \equiv \sigma N, \tag{8.20}$$

where N is the number density of the nuclei in the material. The dependence of the number density on temperature as result of thermal expansion is neglected in this section, which instead concerns the microscopic cross section changes in the laboratory frame as a result of Doppler broadening.

The microscopic cross sections depend on the relative velocity between the neutrons and the target nuclei. Doppler broadening of the cross sections is necessary because the neutron transport equation is solved in the laboratory frame of reference, not in the frame of reference of the nuclei. The nuclei move with respect to the laboratory frame due to the kinetic energy of the atoms [77]. The exact treatment of Doppler broadening by Cullen [78] calculates a microscopic cross section $\sigma(v, T)$ for a given temperature T and laboratory-frame neutron velocity v from base cross section data $\sigma(v_r, T_0)$ for a different temperature T_0 and the relative

neutron velocity v_r.

This exact treatment does not apply directly to the energy-integrated multigroup cross sections, given that the neutron velocity has already been integrated out of the cross section dependencies and the cross sections are assumed to already be in the laboratory frame. To approximate Doppler broadening, the "Combined Doppler broadening model" in Ref. [79] assumes an expansion of the microscopic cross sections in terms of temperature,

$$\sigma(T) = \sum_{i=0}^{\infty} \left[a_i T^{-i/2} + b_i T^{i/2} + c_i T^i \right], \tag{8.21}$$

where a_i, b_i and c are expansion coefficients. The c_i term represents cross section physics that are positively correlated with material temperature, while the a_i term represents the opposite correlation. The b_i term represents physics that are not monotonic with respect to temperature. To truncate the expansion, the a_i and b_i coefficients are assumed to be nonzero for $i = 1, \ldots, 6$ and y the c_i coefficients are assumed to have only a single term for $i = 0$.

For the problems in this section, two cross sections $\sigma(T_0) = \sigma_0$ and $\sigma(T_1) = \sigma_1$ are assumed to be known at temperatures below (T_0) and above (T_1) those expected in the physical system. The temperature is calculated as an interpolation between these two cross sections with a functional dependence based on Eq. (8.21). As the cross section interpolation for two points requires only two unknowns and should be monotonic, the b_i terms are neglected and the expansion is further truncated to include only the c_0 term and the a_1 term,

$$\sigma(T) = c_0 + a_1 \sqrt{\frac{1}{T}}. \tag{8.22}$$

To solve for the expansion coefficients, the constraints from the known cross sections are applied to get

$$\sigma(T) = \left(\sqrt{\frac{1}{T_0}} - \sqrt{\frac{1}{T_1}} \right)^{-1} \left[\left(\sqrt{\frac{1}{T}} - \sqrt{\frac{1}{T_1}} \right) \sigma_0 + \left(\sqrt{\frac{1}{T_0}} - \sqrt{\frac{1}{T}} \right) \sigma_1 \right]. \tag{8.23}$$

This is the function used to interpolate between two known values of the cross section.

8.4.2 Calculating heat generation from the neutron flux

For a nuclear reactor, the volumetric heat generation rate in the heat conduction equation is fission. The three components of prompt fission energy release are the kinetic energy of the fission products, neutrons and photons. The fission products and photons, which unlike neutrons do not travel far from the fission site, make up over 97 percent of the prompt fission energy release for the isotopes of uranium and plutonium used in nuclear reactors [80]. The remainder of the fission energy (around 10 percent total) is emitted by the decay of fission products, as anti-neutrinos or by the capture of fission neutrons [81]. The following equations

Algorithm 8.5 Coupling of heat conduction and neutron transport

initialize the temperature to a constant value
initialize the scalar flux
while scalar flux has not converged **do**
    perform the transport equation integration
        use the temperature from previous iteration to calculate cross sections
    run transport calculation
    if problem is k-eigenvalue **then**
        normalize the total power to a given value
    end if
    perform heat transfer integration
        use the current scalar flux to calculate heat generation rate
        use temperature from previous iteration to calculate conduction coefficients
    run heat conduction calculation
end while

assume that all of the energy created in a fission event is deposited at the site of fission.

The rate at which fission events occur in each energy group is calculated by multiplying the scalar flux $\phi^0_{0,g}$ [Eq. (2.11a)] by the fission cross section $\Sigma_{f;g}$. To calculate the total energy source from fission, this rate is multiplied by κ_g, the mean energy released in a fission event for energy group g, and summed over all energy groups,

$$\dot{q} = \sum_{g'=1}^{G} \kappa_{g'} \Sigma_{f;g'} \phi^0_{0,g'}, \tag{8.24}$$

In the discretized heat conduction equation [Eq. (8.7)], this energy generation rate is integrated over volume for each weighted equation. This requires spatially-dependent values of the cross sections and scalar flux, which can be calculated using an interpolation for the cross sections [Eq. (8.23)] and a basis expansion for the moments of the angular flux [Eq. (2.30b)].

A time-dependent coupling of the heat transfer and neutron transport equations would couple the time-dependent neutron transport equation [Eq. (2.6)] and the time-dependent heat transfer equation [Eq. (8.1)] directly through, for example, operator splitting. For the steady-state and k-eigenvalue equations [Eqs. (2.16a) and (2.17a)], this form of coupling can no longer be applied. Instead, an iterative procedure is applied to calculate the scalar flux and temperature profile, as shown in Alg. 8.5. The temperature and scalar flux are alternately calculated assuming the other is constant until the solution converges. If the thermal conductivity is chosen to be temperature-dependent, the temperature from the previous iteration is used to evaluate the conductivity.

For the k-eigenvalue equation [Eq. (2.17a)], the scalar flux is an eigenvector and therefore should be normalized to a physical value. Given the total volumetric heat generation Q for a given region V, the

spatially-dependent heat generation is normalized according to

$$\dot{q}_{norm} = \left(\frac{Q}{\int_V \dot{q} dV} \right) \dot{q}. \tag{8.25}$$

This normalized source is used in place of Eq. (8.24) for an eigenvalue transport problem to calculate the temperature distribution in Eq. (8.9).

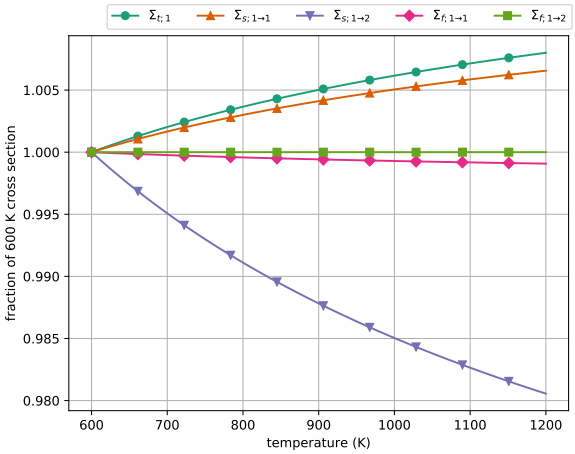
8.5 Coupled VERA pincell problem

One of the challenges in writing multiphysics codes is verification of the results. The coupling of neutron transport and material temperature in a nuclear reactor, for instance, changes the cross sections of the transport equation, which makes methods such as the MMS difficult to apply. As such, the results for this coupled problem are presented without direct verification. The neutron transport and heat transfer codes are separately verified to accurately solve problems with spatially-dependent sources and material properties in Secs. 5.2 and 8.3.2, respectively. As neither of the codes is altered to calculate these results, only additional sources of error in the equations are in the assumptions made for the temperature dependence of the cross sections [Eq. (8.23)] and the fission energy source [Eq. (8.24)]. Neither these results nor the results for the separate codes attempt to validate the code against physical results, which may require more complicated two-phase flow for the convective channels around the pincells and more energy groups for the neutron transport calculation. Instead, these results can be considered to be for an altered problem with simplified physics and the prescribed constants and coefficients. As this altered problem shares many physical characteristics with the original problem, some conclusions for the original problem can be inferred.

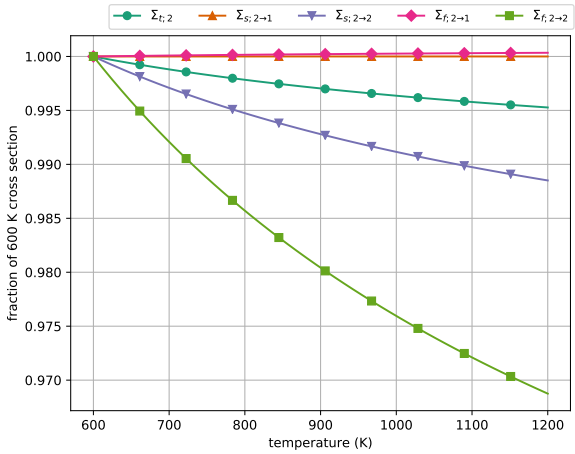
8.5.1 Physical data

The coupled problems under consideration are the VERA pincells without (1B) and with (1E) IFBA from Sec. 7.1. The cross sections for interpolation in Eq. (8.23) are the those with $T_0 = 600$ Kand $T_1 = 1200$ K for each problem from Table 7.5 for 1B and Table 7.6 for 1E. The interpolated temperature dependence of these cross sections is shown in Fig. 8.6.

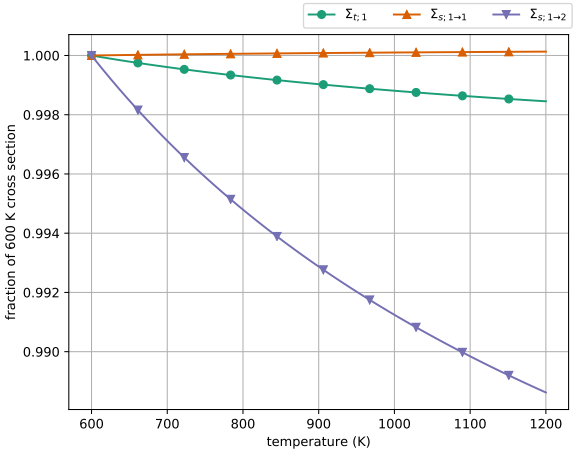
The temperature for a pincell is highest at the center of the fuel region and as such, changes in reactor power affect the cross sections of the fuel more than cross sections of the IFBA, gap or cladding. In the fuel region, an increase in temperature results in fewer neutrons in the thermal group as Doppler broadening increases the relative speed of the neutrons to the nuclei. In the fast group, with the increased temperature, neutrons have a higher probability of scattering into the fast group and a lower probability of scattering into the thermal group. This hardening of the spectrum effectively decreases the probability of fission, which is



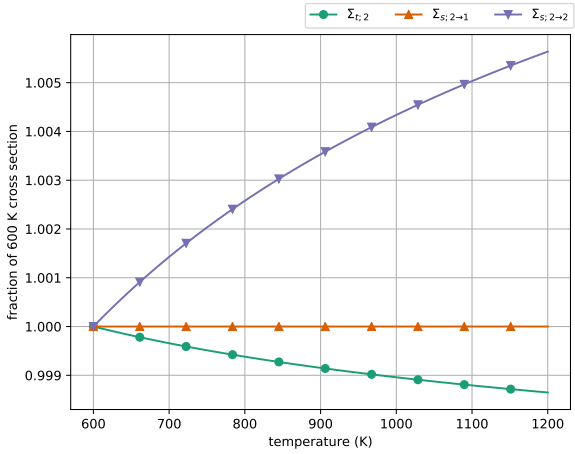
(a) Fuel, fast group



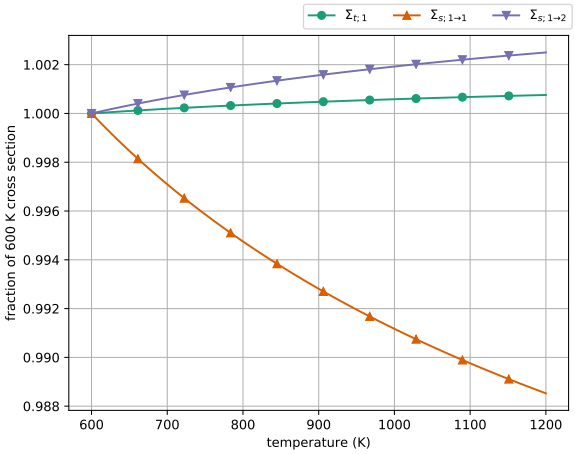
(b) Fuel, thermal group



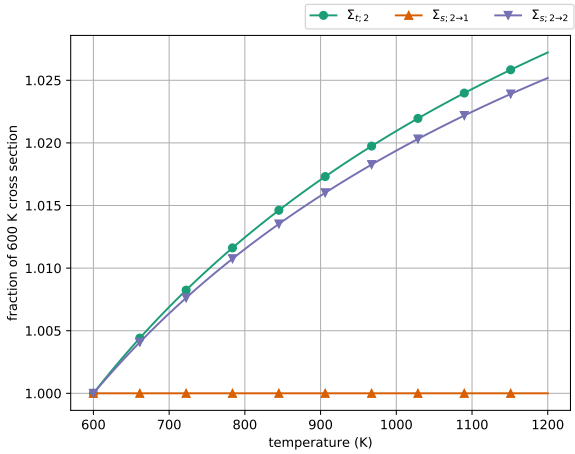
(c) IFBA, fast group



(d) IFBA, thermal group



(e) Gap, fast group



(f) Gap, thermal group

Figure 8.6: Interpolated values for the VERA 1E temperature-dependent cross sections

Table 8.2: Additional material properties for coupled VERA pincell problem

g	κ_g	ν_g
1	196.155 MeV	2.65063
2	193.083 MeV	2.43223

more likely to occur at thermal energies. The faster spectrum also results in a lower total cross section and an increase in the thermal scattering cross section for the IFBA, which absorbs neutrons preferentially in the thermal region. This has the reverse effect of the fuel cross section changes by increasing the probability that a neutron reaches the fuel region. Perhaps because of this, the eigenvalue change for the IFBA problem is smaller than the eigenvalue change for the problem without IFBA.

The data for the energy release by fission κ_g and the average number of neutrons released per fission event ν_g (Table 8.2) is generated using the same procedure as in Sec. 7.1 using the VERA 1B geometry. The data applies to only the fuel region and is assumed to be the same for the 1B and 1E problems, which have the same fuel isotopics, and to be independent of temperature. The calculation of the energy release data in Eq. (8.24) is performed using the tabulated $\chi\nu\Sigma_{f;g'\to g}$ values for each problem,

$$\left(\kappa\Sigma_f\right)_g = \frac{\kappa_g}{\nu_g}\sum_{g'}\chi\nu\Sigma_{f;g\to g'}. \tag{8.26}$$

From the VERA benchmark specifications [65], the rated core power for the benchmark problem is 3411 MW, with 193 assemblies and 264 fuel pins per assembly. As the problems are solved in two dimensions, the core volumetric heat generation is divided by the length of the fuel rod, 385 cm. Using these parameters, the total volumetric heat generation per centimeter at rated core power is

$$Q_{pin} = \frac{\text{3411 MW per core}}{\left(\text{193 assemblies per core}\right)\left(\text{264 fuel pins per assembly}\right)\left(\text{385 cm}\right)} = 173.884 \text{ W/cm per fuel pin.} \tag{8.27}$$

This value is used to normalize the fission energy source in Eq. (8.25).

The dependence of the thermal conductivity on the temperature is usually calculated using empirical equations from experimental results [82]. For this problem, three empirical formulas are used for each of the fuel, gas and cladding. The formulas depend on temperature and return the thermal conductivity in units of $\mathrm{W/m\cdot K}$. The IFBA thermal conductivity is neglected due to its small volume. The thermal conductivity of the fuel is evaluated using an equation from IAEA combined measurements [83],

$$k_{fuel} = \frac{100}{7.5408 + 17.692t + 3.6142t^2} + \frac{6400}{t^{5/2}}\exp\left(\frac{-16.35}{t}\right), \tag{8.28}$$

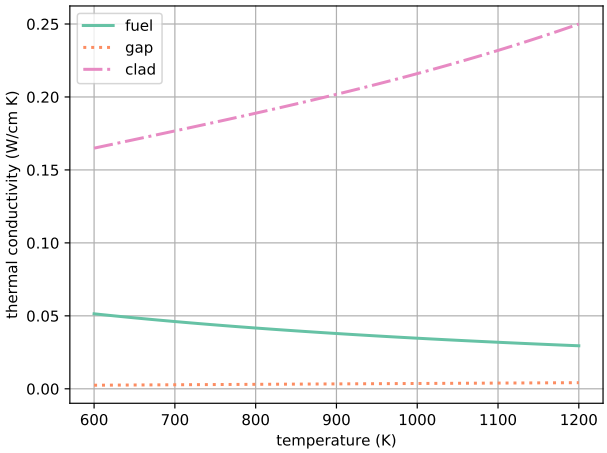


Figure 8.7: Temperature dependence of the thermal conductivity for the fuel, gap and clad for the coupled VERA pincell problem

with $t = T/1000$ K. The gap conductance is calculated using the URGAP model for helium from Lassmann and Hohlefeld [84],

$$k_{gap} = a_\lambda T^{b_\lambda}, \tag{8.29}$$

with $a_\lambda = 0.17632 \times 10^{-2}$ and $b_\lambda = 0.77163$. Finally, the cladding thermal conductivity is a correlation from the MATPRO library [77],

$$k_{clad} = 7.51 + 2.09 \times 10^{-2} T - 1.45 \times 10^{-5} T^2 + 7.69 \times 10^{-9} T^3. \tag{8.30}$$

These functions are evaluated for each quadrature point when performing the numerical integration of Eq. (8.11) or (8.14). The temperature dependence of these conductivities in units of $\mathrm{W/cm \cdot K}$ is shown in Fig. 8.7. The conductivity of the helium gap increases from 0.00245 to 0.00419 between 600 K and 1200 K. The cladding increases in conductivity as the temperature increases (0.16491 at 600 K and 0.24998 at 1200 K), while the fuel decreases in conductivity (0.05140 at 600 K and 0.02948 at 1200 K).

At the boundary, the temperature of the moderator is assumed to stay at a constant $T_\infty = 600$ K. The convective heat transfer coefficient is chosen to be a constant of

$$h = 3.0 \ \mathrm{W/cm^2 \cdot K}, \tag{8.31}$$

as recommended by Ref. [85] for single-phase forced convection.

8.5.2 Results

Given the physical data from Sec. 8.5.1, the only remaining problem specifications needed to solve the coupled problem are the spatial and angular discretization options. For the transport equation, the spatial discretization for these results uses the 11765-point set for VERA 1B (from Sec. 7.2) and the 12974-point set for VERA 1E (Sec. 7.3), while the angular discretization uses 256 directions. Cylindrical-geometry heat equations [Eqs. (8.14)] are used chosen for this problem for simplicity in specification of the cylindrical convective boundaries. For the heat transfer calculation, 1001 evenly-spaced points are used in cylindrical geometry.

The coupling of the heat transfer and transport calculations works as follows. After the two-dimensional neutron transport calculation, the energy source is converted to radial form by averaging the source angularly,

$$\dot{q}_{rad}\left(r\right) = \frac{1}{2\pi}\int_0^{2\pi} \dot{q}\left(x\left(r,\theta\right), y\left(r,\theta\right)\right) d\theta,$$

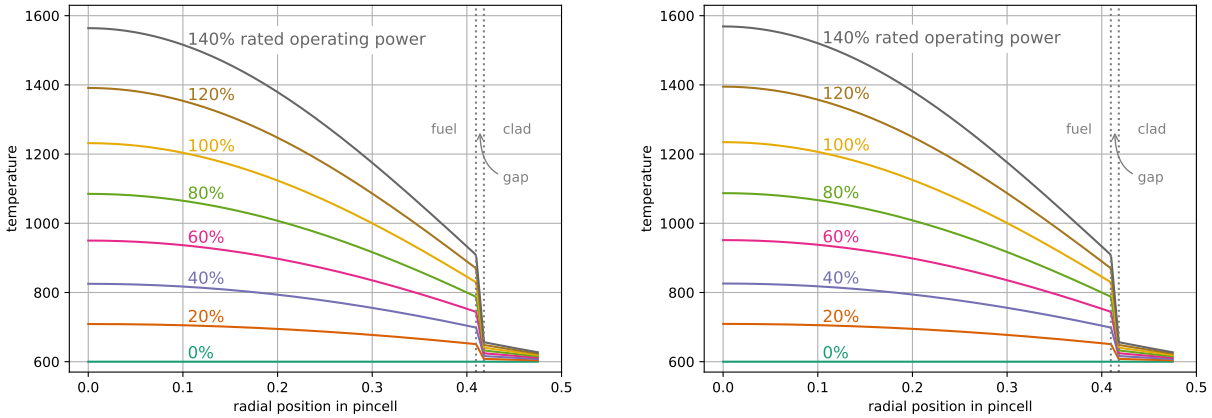
and normalized [Eq. (8.25)] according to

$$\dot{q}_{norm}\left(r\right) = \left(\frac{Q}{2\pi\int_0^R \dot{q}_{rad}\left(r\right) r dr}\right) \dot{q}_{rad}\left(r\right). \tag{8.32}$$

At the end of the heat transfer calculation, the temperature dependence for the transport integrals is calculated by converting the cylindrically-dependent temperature back to Cartesian coordinates.

To generate results for different power levels of the reactor, the power is varied linearly from zero-power conditions up to 140 percent of the rated core power. The eigenvalue and spatial dependence of the temperature and eigenvector are recorded for each case. For both VERA 1B and VERA 1E and for all power levels, the eigenvalue converges to less than 0.1 pcm within five coupled transport/heat iterations.

The radial temperature profiles for each of the two cases are shown in Fig. 8.8. The temperature dependence of the pincell shows the effect of varying heat conduction coefficients on the solution. In the cladding, the high conduction coefficient leads to good heat transfer and a relatively flat solution. The hydrogen gap, on the other hand, acts to insulate the fuel from the cladding due to poor heat transfer. The fuel conduction falls in between these two extremes. The 1200–1300 K temperatures at the centerline of the fuel at 100 percent rated operating power are consistent with the centerline temperatures at startup reported in a VERA simulation for Watts Bar Unit 1 [86]. Because the eigenvectors are normalized to the same pin power Q_{pin} for VERA 1B and 1E, the temperature profiles are very similar. Due to the IFBA on the fuel edge for the 1E problem, the power would be lower for that particular pincell than the average pincell. It would be anticipated that the temperature would accordingly be lower.



(a) VERA 1B

(b) VERA 1E

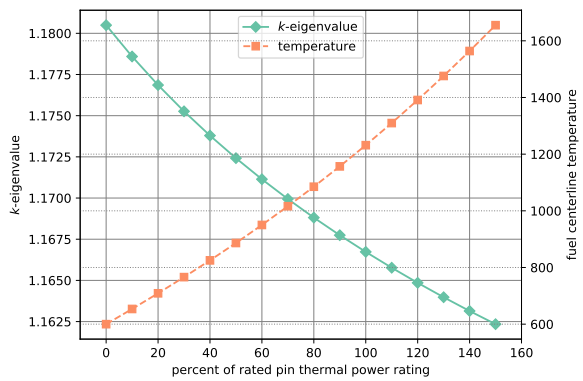
Figure 8.8: Radial temperature profile for coupled VERA pincell problems at various power levels

The dependence of the eigenvalue and the fuel centerline temperature on the percent rated power is shown in Fig. 8.9. As expected, the fuel centerline temperature increases as the power increases. The eigenvalue decreases as the temperature increases. In a continuous-energy simulation, the decrease in the eigenvalue would be an effect of the Doppler broadening of the absorption resonances as the temperature increases, resulting in fewer neutrons reaching thermal energies. For this multigroup simulation, as discussed in Sec. 8.5.1, these physics are manifested in the lower scattering scattering cross section from the fast group to the thermal group and a corresponding higher absorption cross section representing the higher probability of resonance absorption. The eigenvalue decreases by 1376.3 pcm for the pincell without IFBA and by 791.2 pcm for the pincell with IFBA from zero-power conditions to full rated power. The smaller change in the eigenvalue for the IFBA problem may be due to a lower IFBA absorption cross section at higher temperatures, which has a positive effect on the reactivity.

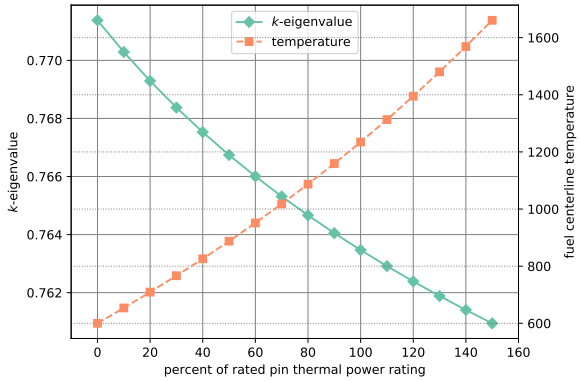
The spatially-dependent relative difference between the flat-temperature 600 K solution and the solution at full rated power is shown in Fig. 8.10. As expected, the normalized eigenvector increases in the fast energy group and decreases in the thermal energy group. Relative to the center of the fuel, where the Doppler broadening is most pronounced, the edge of the fuel produces a higher fraction of the fission neutrons as the power of the reactor increases. In the thermal group, the eigenvector decreases least in the moderator, which is assumed to have a constant temperature of 600 K, and decreases most where the temperature is highest at the center of the pincell (see Fig. 8.8 for the temperature distribution).

Reactivity is a measure of the deviance of the k-eigenvalue from criticality,

$$\rho \equiv \frac{k-1}{k}. \tag{8.33}$$

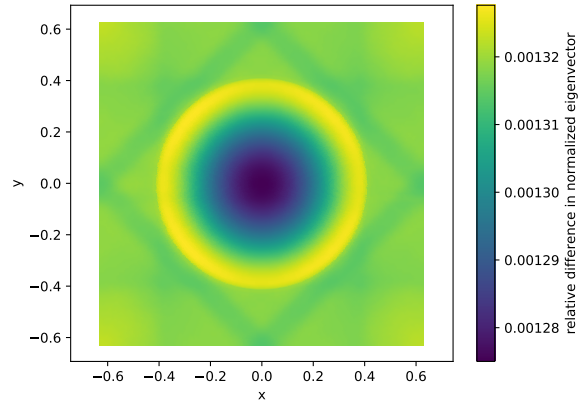


(a) VERA 1B

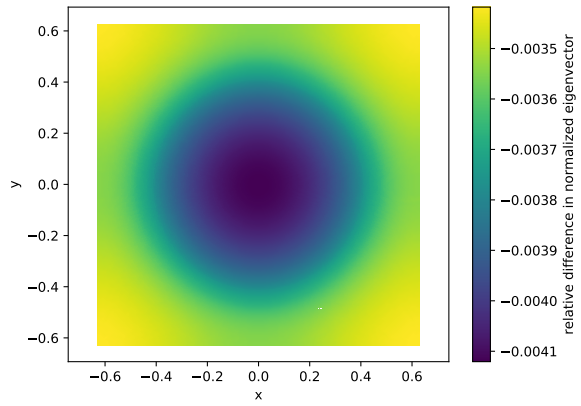


(b) VERA 1E

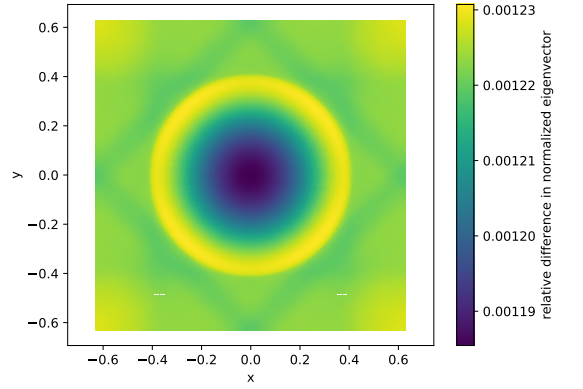
Figure 8.9: Dependence of k-eigenvalue and pincell centerline temperature on reactor power level for VERA pincell problems



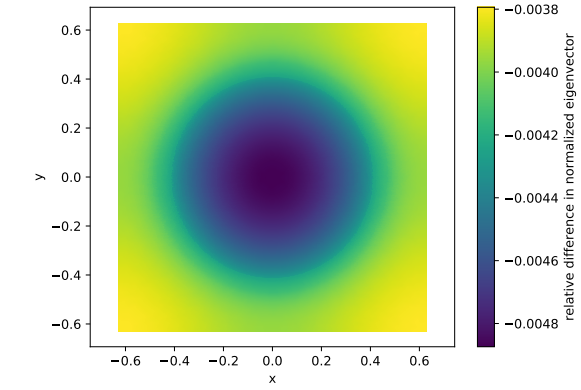
(a) VERA 1B, fast group



(b) VERA 1B, thermal group

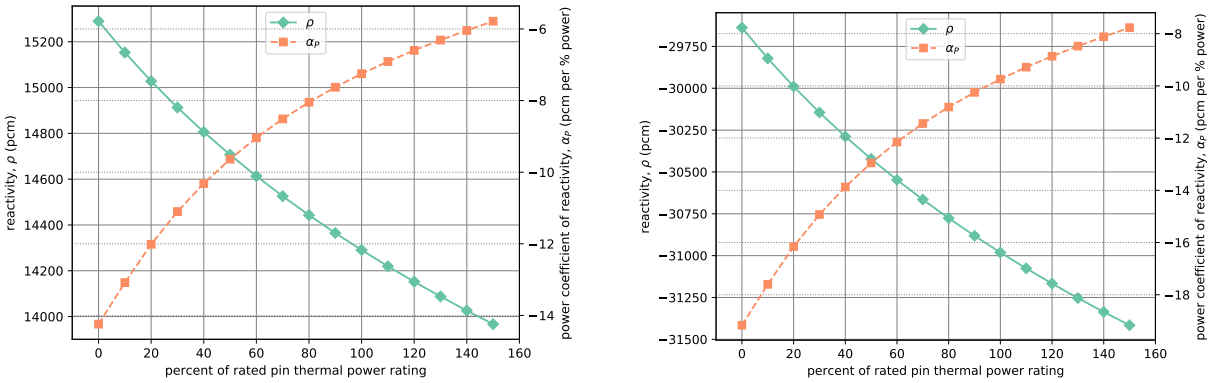


(c) VERA 1E, fast group



(d) VERA 1E, thermal group

Figure 8.10: Relative difference in normalized eigenvector between zero and full power conditions for VERA pincell problems



(a) VERA 1B

(b) VERA 1E

Figure 8.11: Dependence of reactivity and power coefficient of reactivity on reactor power level for VERA pincell problems

The power coefficient of reactivity, or differential of reactivity with respect to the power, is defined as [87]

$$\alpha_P \equiv \frac{\partial \rho}{\partial P}, \tag{8.34}$$

where P is the power level of the reactor. The power coefficient of reactivity is commonly used for the evaluation of the safety of a nuclear reactor. If the power coefficient of reactivity is positive, an increase in reactivity (e.g. by removing a control rod), may trigger a feedback loop that further increases power and the reactivity further.

To calculate the power coefficient of reactivity for the coupled VERA problem, the reactivity is calculated at each power level and the derivative is calculated using second-order finite differences. Figure 8.11 shows the reactivity in pcm ($10^{-5}\rho$) and the power coefficient of reactivity in pcm per percent power. The reactivity decreases similarly to the eigenvalue. The power coefficient of reactivity increases as the power (and temperature) increases from -14 (at 0% power) to -7 (at 100% power) pcm per percent power for VERA 1B and from -19 to -10 pcm per percent power for VERA 1E. These values are consistent with the full-core power coefficients reported for Watts Bar Unit 2 of -15 to -10 pcm per percent power at beginning of life [88]. The shape of the power coefficient of reactivity is also consistent with the Watts Bar Unit 2 solution due to the $1/\sqrt{T}$ dependence of the cross sections in Eq. (8.23). The difference between these pincell simulations and the full-core Watts Bar result is at most 4 pcm per percent power.

Chapter 9

Summary and future work

Meshless methods can simplify the process of discretizing the problem domain for PDEs. Meshless methods have been applied to the radiative transport equation previously, but simplifications such as homogeneous materials, directionally-dependent test functions and energy independence makes the direct application of these methods to neutron transport difficult. Work on meshless methods for the neutron transport equation in the past has been limited to simple geometries with homogeneous materials and little spatial variation in the solution. Here, the neutron transport equation is discretized with no constraints on the location or number of points or the geometry inside the domain. The implementation of this discretization is verified for geometrically-challenging problems with strong spatial dependence.

9.1 Weak-form meshless transport

To solve the neutron transport equation accurately and efficiently using the MLPG method, several conditions are enforced. The integration of the basis and weight functions should be independent of direction due to the high number of directions required for many problems. The equations should include stabilization, which prevents oscillations in the solution. Finally, the discretization should enforce global particle balance. The presented discretization of the neutron transport equation with SUPG stabilization and MLS basis and weight functions permits efficient, directionally-independent integration, satisfies the neutron balance and dampens oscillations. To account for the spatial variation of neutron cross sections, two cross section methods are presented, both of which are informed by and satisfy neutron conservation.

The efficient implementation of the neutron transport equation involves use of Krylov iterative methods to invert the streaming term and converge on the scattering source. The application of preconditioners to the streaming term permits the solution using GMRES. This, in turn, allows for application of the MLPG method to problems with tens to hundreds of millions of unknowns. The main drawback of the MLPG method as presented here is the high memory and computational cost associated with the simulations.

Integration of the MLPG system of equations is performed using either a fully-meshless approach or a background mesh, which is agnostic to the geometry of the problem and the location of the basis and

weight functions. For large problems with geometric complexity, the background mesh integration is more efficient as all the interdependent MLS functions at a point can be calculated simultaneously. The cost savings permits use of more quadrature points per unit volume to more accurately integrate heterogeneous domains. For integration of standard RBF functions on problems with smooth cross sections, the meshless integration technique converges to the analytic integrals with similar speed to the background mesh, while better accounting for variability in the basis and weight function radii.

The MLPG transport equations are verified using the method of manufactured solutions for problems with continuously-variable and discontinuous cross sections. The equations are stable for a wide range of basis and weight function radii and SUPG parameters. The unmodified cross sections (with basis and weight function dependence) are more accurate than the approximate cross sections (with only basis function dependence) but are also more expensive to compute and store. The results for the manufactured problems in two and three dimensions show second-order convergence to the manufactured solutions.

For a homogeneous, purely-absorbing slab, the MLPG equations show third-order convergence the analytic solution for optically thick and optically thin problems. For a heterogeneous slab, the MLPG equations converge to 10^{-6} relative L_2 error for the full cross sections by 5000 points and 256 directions. The three-dimensional Problem 1 from the Kobayashi set, which is designed to simulate void regions, converged to under 10^{-2} L_2 error by 64000 points and 512 directions for both the absorbing and scattering case and both cross section representations when compared to the values for a Monte Carlo point estimator. The worst-case error is much higher due to ray effects but improves with additional directions to around 0.3 L_∞ error for 32768 points and 2048 directions.

The eigenvalue problems have multigroup cross sections generated from continuous-energy Monte Carlo simulations and benchmark results generated from multigroup Monte Carlo simulations. For a simple pincell, the k-eigenvalue converges to approximately 10 pcm at around 1000 points and 256 directions. As more directions are added, the solution converges further. For 9825 points and 4096 directions, the problem has around 1 pcm error. The IFBA pincell problem is much more challenging due to the strong discontinuities in the cross sections and solution near the fuel boundary. The solution requires a large number of both points and directions to converge due to ray effects and the small relative volume of the IFBA region. For 34086 points and 1024 directions, the solution has 7 pcm error. The largest error in the solution occurs at the pincell boundary near the IFBA layer. When applied to the IFBA pincell problem, the weak-form discretization without SUPG stabilization requires a direct LU decomposition for the streaming operator due to ill-conditioning and exhibits oscillations in the final solution.

A three-dimensional plutonium ellipsoid reflected by thorium converges to 89 pcm error for 40078 points and 512 directions. Similar to the pincell problem, the error in the solution is concentrated at material

discontinuity near the edge of the ellipsoid. A set of points with higher density near this gradient achieves better convergence. The three-dimensional pincell is not converged to a similar error to the pincell problems due to the high cost of three-dimensional simulations, which inherently require many more spatial points to converge than similar two-dimensional results.

Finally, a MLPG discretization of the heat equations is derived. The implementation of this discretization is verified by the method of manufactured solutions and by comparison to an analytic solution. A coupling scheme for the temperature from the heat transfer equations and the flux from the neutron transport equation is applied to a realistic pincell problem with temperature-dependent cross sections and material conductivities. The results for the power coefficient of reactivity show a maximum of 4 pcm difference from published values for the full reactor core.

9.2 Strong-form meshless transport

The strong-form collocation neutron transport equation is derived from the weak-form MLPG equation, which allows the application of the basis function cross section weighting method from the MLPG equations. Unlike the weak-form equations, the strong-form equations do not allow stabilization (which results in oscillations and poorly-conditioned streaming matrices), do not enforce particle balance (which produces unpredictable results), and do not as accurately model the spatial dependence of cross sections. However, the implementation of strong-form neutron transport is less complicated. For problems with simple geometries, the strong-form discretization can work well with an informed choice of basis function parameters.

The same manufactured problems as in the weak case are applied to the collocation equations. For a problem with a sinusoidal solution shape and sinusoidal cross sections, the strong-form equations are not highly parameter-dependent. To achieve good error properties, the basis functions need to have larger radii than the functions for the MLPG equations, which contributes to the ill-conditioning issues of the streaming operator. For the problem with discontinuous cross sections, the strong-form equations behave much more erratically. The results are highly dependent on the radii of the basis and weight functions and do not converge for the three-dimensional problem.

When applied to the IFBA pincell problem, the collocation equations with the standard point evaluation of the cross sections do not calculate the correct solution due to the low amount of geometric information in the equations. The k-eigenvalue has 16069 pcm error, the solution has unphysical oscillations and the eigenvector does not agree with the Monte Carlo solution. The basis function cross section weighting performs much better. The solution exhibits the correct symmetries and does not oscillate. The eigenvalue error of 2956 pcm is still large compared to the weak-form solution for the same set of points (303 pcm).

For a problem without discontinuities in the material properties, the strong-form equations provide

accurate solutions. However, the solutions are less accurate than the weak-form equations for every case tested here. The only additional computational cost of the MLPG equations compared to the collocation equations is in the integration step. For problems with a large number of spatial points, the ill-conditioning of the streaming operator in the collocation method adds more computational cost than is saved by the lack of integration, and for geometrically difficult problems, integral weighting of the cross sections may still be needed.

9.3 Future work

9.3.1 Choice of methods and parameters for the MLPG discretization

The parameter space of the meshless discretizations presented here is simultaneously a feature and a complication. There are many choices of RBFs not explored here, including RBFs developed specifically for the PDE being solved [10]. The choices of which RBFs to use, how to weight the cross sections, what kind of stabilization to use and with what parameters, how to calculate the radii of the basis and weight functions, and how to integrate the basis and weight functions are explored here, but there are likely other choices that work equally well or better. The optimization of meshless methods remains an open problem, particularly for equations such as the neutron transport equation that have not been extensively studied for these methods.

The main drawback of the MLPG technique is its high computational cost. As such, many of the recommendations for future work included here involve methods to reduce the memory and time requirements to complete simulations. One of the simplest ways to improve the cost of the MLPG method presented here would be through better preconditioners for the streaming operator of the transport equation. It is possible that other preconditioners, such as multigrid preconditioners [89] that have been successfully applied to CFEM problems [90], would be more efficient than than the two presented here. The direction and energy-dependent ILUT preconditioners for the MLPG method require a large amount of memory and restrict refinement in space, angle and energy. A preconditioner without the memory or computational limitations of those presented here would significantly expand the range of problems to which this method can be applied.

The integration approach used here could be described as brute-force: if enough integration points are applied to the integral of a discontinuous function, the integral should eventually converge. However, the integration could be significantly reduced in cost with the implementation of integration methods designed specifically for discontinuous functions [91] near material discontinuities and for efficient integration of MLS functions away from discontinuities [92].

9.3.2 Application to problems with irregular geometries

The motivation for this research is to solve the neutron transport equation for realistic problems using meshless methods without subdivision of the problem. Much of this dissertation is focused on verifying that the meshless discretization as derived in Sec. 2 accurately solves the transport equation for problems of various dimensions and geometries. This should allow future application of this method to problems that include irregular geometry features, such as cracks in fuel and buildup of CRUD (Chalk River Unidentified Deposits) on the outer perimeter of the cladding, for which the meshless approach may perform better than current mesh-based methods.

The independence of the solution nodes and the material properties introduces flexibility that is uncommon in mesh-based methods. For problems in which the material is stochastic in nature, the discretization could account for a realization of the geometry without changing the location of the solution nodes. Instead, inclusions in a material would be accounted for in the integration step. This approach could likewise be used to solve the problem on geometries with small or irregular features with known positions.

9.3.3 Time-dependent transport

Much of the cost of the MLPG transport method is in the integration of the basis and weight functions and material parameters and in the initialization of the preconditioner matrices. For problems in which the cross sections are not temporally dependent but the neutron flux is, the integrals and preconditioners would be reusable at each time step of the transport equation. For steady-state problems of moderate size (10000 to 20000 points) in three dimensions, the solution of the transport equation once the integrals and preconditioners are initialized represents between a tenth and a fourth of the total computational cost, depending on the number of outer iterations required to converge the scattering source. A time-dependent problem is approximately equivalent in cost to a steady-state problem at each time step. For around 100 time steps, the total cost of the code would increase by between 10 and 40 times compared to a steady-state simulation. For smaller problems or problems within a DFEM mesh (see Sec. 9.3.5), the cost of the solution is much smaller in comparison to the initialization steps and the time required for a time-dependent solution would accordingly be lower.

9.3.4 Coupling to Monte Carlo

While the MLPG method is difficult to implement, it is not difficult to use. The ability to represent the spatial shape of a solution with a geometric representation that does not conform to the materials in the problem makes the MLPG method attractive for use in calculating weight windows using the adjoint flux

[93]. As discussed in Sec. 6.1, the MLPG equations appears to be able to resolve the solution in problems that are very optically thick and benefit from the application of weight windows in a Monte Carlo simulation.

The CSG used to represent materials for the MLPG results (see Appendix B) in Chapters 6 and 7 is very similar to the CSG that would be found in a Monte Carlo code. Informed by the general choices of parameters from these results, the only additional parameter a MLPG simulation would need to calculate the adjoint flux would be the centers of the basis and weight function centers, which only have to conform to the problem boundaries, not the material discontinuities inside the problem. For these calculations, high accuracy may not be needed, in which case the number of points to needed solve the transport equation and the number of integration ordinates for the integration of the basis and weight functions could be much lower than the number needed to fully resolve the solution.

9.3.5 Applications to the DFEM

The MLPG method is mathematically identical to the DFEM inside a single element with a specialized class of basis and weight functions. The method of using these functions in a finite element method [26] could be extended to the use of the more general MLPG equations derived in Chapter 2 with a few modifications. The MLPG neutron transport equations [Eqs. (2.32)] include only one term that couples the equation to physics outside of the problem boundary, specifically the boundary source term,

$$\int_{\boldsymbol{\Omega}_n \cdot \boldsymbol{n} < 0} |\boldsymbol{\Omega}_n \cdot \boldsymbol{n}| \, \psi^{ext}_{n,g} w_j dS. \tag{9.1}$$

For an interior element in a DFEM, this source represents the integral over the incoming flux from the upwind elements. Letting the known angular flux for each upwind element with index e have the expansion

$$\psi^{up}_{e,n,g} = \sum_i b^{up}_{e,i} \alpha^{up}_{e,i,n,g}, \tag{9.2}$$

it follows that the integral for the incoming source on the boundary for the current element e_0 is

$$\sum_{e:\boldsymbol{\Omega}_n \cdot \boldsymbol{n}_e < 0} \sum_i \alpha^{up}_{e,i,n,g} \, |\boldsymbol{\Omega}_n \cdot \boldsymbol{n}| \int_{S_{e_0,e}} b^{up}_{e,i} w_{e_0,j} dS. \tag{9.3}$$

The connectivity of the two elements would be determined by the basis and weight functions that intersect on the shared element face. This approach of applying the MLPG equations to each element of a DFEM mesh would allow for heterogeneous elements that do not conform as tightly to the problem geometry. By splitting the problem in this way, the cost of the MLPG method would decrease, although there would likely be a drop in accuracy across the discontinuous solutions at the element faces. This would also lessen the

need to parallelize the MLPG discretization using a general MPI method, as each element would presumably be small enough to be calculated on a single node and the elements would only be connected on the element faces.

As mentioned in the introduction (Chapter 1), spatially-dependent cross sections can present problems for mesh-based methods implemented with the assumption that cross sections are constant inside each cell. The cross section approximation methods from Sec. 2.8 are directly applicable to a DFEM without modification. The full weighting of the cross sections in an element would multiply the cross section storage cost by the number of basis functions times the number of weight functions, while the basis function weighting of the cross sections would only multiply the cross section storage cost by the number of basis functions. For instance, for a finite element mesh composed of linear rectangular prisms, there are 8 basis and weight functions, and so the full and basis weighting options would, respectively, multiply the storage cost of cross sections by 64 and 8 times. For a mesh with linear tetrahedral elements, the storage cost would be, respectively, 16 and 4 times the original. For problems for which the full cross section method would be prohibitive in terms of memory cost, the basis-weighted cross sections may be a good compromise between cost and accuracy.

Appendix A

Legendre polynomials and spherical harmonics

A.1 Legendre polynomials

The definition of the scattering moments in Eq. (2.4) and the scattering expansion in Eq. (2.5b) depend on the Legendre polynomials. The first few Legendre polynomials are

$$P_0(\mu) = 1, \tag{A.1a}$$

$$P_1(\mu) = \mu, \tag{A.1b}$$

$$P_2(\mu) = \frac{1}{2}\left(3\mu^2 - 1\right). \tag{A.1c}$$

A recursion relationship allows the calculation of higher-order Legendre polynomials,

$$P_\ell(\mu) = \frac{(2\ell-1)\,\mu P_{\ell-1}(\mu) - (\ell-1)\,P_{\ell-2}(\mu)}{\ell}. \tag{A.2}$$

The Legendre polynomials are orthogonal,

$$\int_{-1}^{1} P_\ell(\mu)\,P_m(\mu)\,dx = \frac{2}{2\ell+1}\delta_{\ell,m}, \tag{A.3}$$

where $\delta_{\ell,m}$ is the Kronecker delta function defined in Eq. (2.20).

A.2 Spherical harmonics

The standard spherical harmonics are complex. To avoid complex algebra, the spherical harmonics functions used to define the spherical harmonics moments in Eq. (2.3) are chosen to be the real (or tesseral) spherical harmonic functions. In terms of the direction unit vector with polar angle cosine $\mu = \cos\theta$ and azimuthal angle γ,

$$\mathbf{\Omega} = \left(\mu,\ \sqrt{1-\mu^2}\cos\gamma,\ \sqrt{1-\mu^2}\sin\gamma\right), \tag{A.4}$$

the real spherical harmonics functions have the definition [77]

$$Y_\ell^m(\mathbf{\Omega}) = N_\ell^m P_\ell^{|m|}(\mu) T_m(\gamma), \tag{A.5a}$$

where

$$N_\ell^m = \sqrt{(2-\delta_{m,0})\frac{(\ell-|m|)!}{(\ell+|m|)!}}, \tag{A.5b}$$

$$T_m(\gamma) = \begin{cases} \cos m\gamma, & m \geq 0, \\ \sin|m|\gamma, & \text{otherwise.} \end{cases} \tag{A.5c}$$

The P_ℓ^m are the associated Legendre polynomials. The associated Legendre polynomials can be calculated recursively as

$$P_\ell^m(\mu) = \begin{cases} (2\ell-1)!!\left(1-\mu^2\right)^{\ell/2}, & \ell = m, \\ \mu(2\ell-1)P_{\ell-1}^{\ell-1}(\mu), & \ell = m+1, \\ \dfrac{\mu(2\ell-1)P_{\ell-1}^m - (\ell+m-1)P_{\ell-2}^m(\mu)}{(\ell-m)}, & \ell \geq m+2. \end{cases} \tag{A.5d}$$

The first few associated Legendre functions are

$$P_0^0(\mu) = 1, \tag{A.6a}$$

$$P_1^0(\mu) = \mu, \tag{A.6b}$$

$$P_1^1(\mu) = -\left(1-\mu^2\right)^{1/2}, \tag{A.6c}$$

$$P_2^0(\mu) = \frac{1}{2}\left(3\mu^2-1\right), \tag{A.6d}$$

$$P_2^1(\mu) = -3\mu\left(1-\mu^2\right)^{1/2}, \tag{A.6e}$$

$$P_2^2(\mu) = 3\left(1-\mu^2\right). \tag{A.6f}$$

Associated Legendre functions with $m < 0$ also exist, but are not needed to calculate the spherical harmonics in Eq. (A.5a). In terms of the components of the direction vector $\mathbf{\Omega}$, the first few real spherical harmonics are

$$Y_0^0(\mathbf{\Omega}) = 1, \tag{A.7a}$$

$$Y_1^{-1}(\mathbf{\Omega}) = \Omega_z, \tag{A.7b}$$

$$Y_1^0(\mathbf{\Omega}) = \Omega_x, \tag{A.7c}$$

$$Y_1^1(\mathbf{\Omega}) = \Omega_y, \tag{A.7d}$$

$$Y_2^{-2}(\mathbf{\Omega}) = \sqrt{3}\Omega_y\Omega_z, \tag{A.7e}$$

$$Y_2^{-1}(\mathbf{\Omega}) = \sqrt{3}\Omega_x\Omega_z, \tag{A.7f}$$

$$Y_2^0(\mathbf{\Omega}) = \frac{1}{2}\left(3\Omega_x^2 - 1\right), \tag{A.7g}$$

$$Y_2^1(\mathbf{\Omega}) = \sqrt{3}\Omega_x\Omega_y, \tag{A.7h}$$

$$Y_2^2(\mathbf{\Omega}) = \frac{\sqrt{3}}{2}\left(\Omega_y^2 - \Omega_z^2\right). \tag{A.7i}$$

The separate components of the spherical harmonics are orthogonal,

$$\int_0^{2\pi} T_m(\gamma)\, T_{m'}(\gamma)\, d\gamma = \pi\left(1 + \delta_{m,0}\right)\delta_{m,m'}, \tag{A.8a}$$

$$\int_{-1}^{1} P_\ell^m(\mu)\, P_{\ell'}^m(\mu)\, d\mu = \frac{2}{2\ell+1}\frac{(\ell+m)!}{(\ell-m)!}\delta_{\ell,\ell'}. \tag{A.8b}$$

These equations can be used to calculate the original normalization constant N_ℓ^m that allows for the simple orthogonality relation of the spherical harmonics,

$$\int_{4\pi} Y_\ell^m(\mathbf{\Omega})\, Y_{\ell'}^{m'} = \frac{4\pi}{2\ell+1}\delta_{\ell,\ell'}\delta_{m,m'}. \tag{A.8c}$$

The addition theorem for the real spherical harmonics relates the spherical harmonics functions to the Legendre polynomials,

$$P_\ell(\mathbf{\Omega}\cdot\mathbf{\Omega}') = \sum_{m=-\ell}^{\ell} Y_\ell^m(\mathbf{\Omega})\, Y_\ell^m(\mathbf{\Omega}'). \tag{A.9}$$

Using the addition theorem, the Legendre expansion of the scattering cross section (Eq. (2.5b)) can be written as

$$\begin{aligned}\Sigma_s(\boldsymbol{x}, \mu_0 = \mathbf{\Omega}'\cdot\mathbf{\Omega}, E'\to E) &= \sum_{\ell=0}^{\infty}\frac{2\ell+1}{4\pi}\Sigma_{s;\ell}(\boldsymbol{x}, E'\to E)\, P_\ell(\mu_0) \\ &= \sum_{\ell=0}^{\infty}\frac{2\ell+1}{4\pi}\sum_{m=-\ell}^{\ell}\Sigma_{s;\ell}(\boldsymbol{x}, E'\to E)\, Y_\ell^m(\mathbf{\Omega})\, Y_\ell^m(\mathbf{\Omega}').\end{aligned} \tag{A.10}$$

This approximation and is inserted into the original scattering term in Eq. (2.1a),

$$\begin{aligned}
&\sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} \int_0^{\infty} \int_{4\pi} \Sigma_{s;\ell}\left(\boldsymbol{x}, E\right) Y_\ell^m\left(\boldsymbol{\Omega}\right) Y_\ell^m\left(\boldsymbol{\Omega}'\right) \psi\left(\boldsymbol{x}, \boldsymbol{\Omega}', E', t\right) d\Omega' dE' \\
&\quad = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} \int_0^{\infty} \Sigma_{s;\ell}\left(\boldsymbol{x}, E\right) Y_\ell^m\left(\boldsymbol{\Omega}\right) \left(\int_{4\pi} Y_\ell^m\left(\boldsymbol{\Omega}'\right) \psi\left(\boldsymbol{x}, \boldsymbol{\Omega}', E', t\right) d\Omega' \right) dE' \\
&\quad = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi} \sum_{m=-\ell}^{\ell} \int_0^{\infty} \Sigma_{s;\ell}\left(\boldsymbol{x}, E\right) Y_\ell^m\left(\boldsymbol{\Omega}\right) \phi_\ell^m\left(\boldsymbol{x}, E', t\right) dE', \qquad \text{(A.11)}
\end{aligned}$$

to get the scattering expansion for the transport equation.

In the MLPG code, the spherical harmonics up to $\ell = 3$ are calculated directly using Eqs. (A.7). For higher moments, Eqs. (A.5) are used to calculate the spherical harmonics. For the neutron transport applications considered in this dissertation, the maximum degree of the spherical harmonics is $L = 2$.

Appendix B

Constructive solid geometry equations

In a constructive solid geometry (CSG), regions of (usually constant) material are defined by analytic surfaces, such as planes, spheres and cylinders. The CSG can be used in Monte Carlo simulations for tracking particles through the problem domain by calculating which region a particle is located in, the distance in a given direction to the next surface, and the reflection angle off of a surface where applicable. For the MLPG transport equations, a CSG can provide similar data, such as the material at integration points and the reflection angle off of a reflective boundary surface for a given discrete direction.

To find the region in the problem given a position, the CSG calculates the aspect of the particle with respect to the surfaces in the problem. Each region is formed from one or more of these surfaces with a defined aspect for each surface. For instance, for a simple spherical problem, one region, which is outside the problem boundaries, would have a positive aspect with respect to the spherical surface. The second region would have a negative aspect with respect to the spherical surface, and would be inside the problem boundaries. For a point inside the problem, the code would calculate a negative aspect to the sphere and then identify that the aspect matches the second region.

For problems with more complicated geometries, the code checks through the surfaces to find the appropriate region. To find the distance to intersection of the particle, the distance to collision with each surface in the current region is calculated and the lowest distance to collision is the one used. If a ray has multiple collisions for a given surface (e.g. for a particle approaching a sphere), the lowest nonnegative collision is used. The algorithms to find a region given a position and to find an intersection given a position, region and direction are given in Alg. B.6.

In the following sections, the following equations are provided for surfaces represented by a plane, cylinder, sphere and ellipsoid:

- The normal vector of the surface at each point on the surface,
- The distance to intersection and point of intersection of a particle incident on the surface, and
- The aspect of a particle with respect to the surface.

Algorithm B.6 Constructive solid geometry functions

```
1: function FIND REGION(position x)
2:     initialize surface aspects
3:     for each region i do
4:         for each surface j in region i do
5:             if aspect is not stored then
6:                 calculate and store surface aspect with respect to x
7:             end if
8:             if aspect is incorrect for this region then
9:                 continue to next region
10:            end if
11:        end for
12:        return region i
13:    end for
14: end function
15:
16: function FIND INTERSECTION(region i, position x, direction Ω)
17:    initialize current best distance d_best to high value
18:    initialize current best surface s_best to none
19:    for each surface j in region i do
20:        calculate distance d to intersection given x and Ω
21:        if 0 ≤ d < d_best then
22:            set best distance d_best = d
23:            set best surface s_best = j
24:        end if
25:    end for
26:    return best distance d_best and surface s_best
27: end function
```

The aspect for each surface can be positive, negative or coincident. If the aspect is coincident, the particle is located on the surface. The equations are derived using vector notation for simplicity.

B.1 Plane

The general equation for a plane which includes point $\boldsymbol{x}_0$ and has the normal vector $\boldsymbol{n}$ is

$$(\boldsymbol{x} - \boldsymbol{x}_0) \cdot \boldsymbol{n} = 0. \tag{B.1}$$

A particle located at point $\boldsymbol{g}_0$ moving in direction $\boldsymbol{g}_1$ has the trajectory

$$\boldsymbol{x}_p(s) = \boldsymbol{g}_0 + \boldsymbol{g}_1 s, \tag{B.2}$$

where s is the distance traveled from the initial point $\boldsymbol{g}_0$. This equation is inserted into Eq. (B.1) and solved for s to get the distance to intersection for the particle,

$$s_{int} = \frac{(\boldsymbol{x}_0 - \boldsymbol{g}_0) \cdot \boldsymbol{n}}{\boldsymbol{g}_1 \cdot \boldsymbol{n}}. \tag{B.3}$$

The point of intersection can be found by advancing the particle by distance s_{int} in Eq. (B.2).

The geometric relationship between a particle at point $\boldsymbol{x}$ and a plane is defined as

$$\boldsymbol{n} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) \begin{cases} > 0, & \text{positive,} \\ < 0, & \text{negative,} \\ = 0, & \text{coincident.} \end{cases} \tag{B.4}$$

Similar equations are derived for each of the other shapes, with the addition of a calculation for the normal vector, which is a given in the plane equation.

B.2 Cylinder

The equation for a cylinder with a centerline axis in direction $\boldsymbol{\Omega}$ that runs through the point $\boldsymbol{x}_0$ can be written

$$\left\| \boldsymbol{x} - \boldsymbol{x}_0 - \boldsymbol{\Omega} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) \, \boldsymbol{\Omega} \right\|^2 = r^2, \tag{B.5}$$

where r is the radius of the cylinder and $\|(\cdot)\|$ calculates the length of the vector. The particle position is defined the same as before according to Eq. (B.2). Inserting this equation into Eq. (B.5), the cylindrical equation becomes (after some simplification)

$$\left\| \boldsymbol{k}_0 + \boldsymbol{k}_1 s \right\|^2 = r^2, \tag{B.6}$$

with the two constant vectors

$$\boldsymbol{k}_0 = \left[\boldsymbol{g}_0 - \boldsymbol{x}_0 - \boldsymbol{\Omega} \cdot (\boldsymbol{g}_0 - \boldsymbol{x}_0) \, \boldsymbol{\Omega} \right], \tag{B.7a}$$

$$\boldsymbol{k}_1 = \left[\boldsymbol{g}_1 - (\boldsymbol{\Omega} \cdot \boldsymbol{g}_1) \, \boldsymbol{\Omega} \right]. \tag{B.7b}$$

Here $\boldsymbol{k}_0$ represents the vector rejection of $\boldsymbol{g}_0 - \boldsymbol{x}_0$ onto $\boldsymbol{\Omega}$, or the part of the vector connecting the midpoint of the cylinder to the position of the particle that is perpendicular to the cylinder axis. $\boldsymbol{k}_1$ has a similar

geometric meaning for $\boldsymbol{g}_1$, the direction of the particle. The equation can further simplify to

$$\begin{aligned} r^2 &= (\boldsymbol{k}_0 + \boldsymbol{k}_1 s) \cdot (\boldsymbol{k}_0 + \boldsymbol{k}_1 s) \\ &= \boldsymbol{k}_0 \cdot \boldsymbol{k}_0 + 2\boldsymbol{k}_0 \cdot \boldsymbol{k}_1 s + \boldsymbol{k}_1 \cdot \boldsymbol{k}_1 s^2. \end{aligned} \tag{B.8}$$

Defining the constants

$$\ell_0 = \boldsymbol{k}_0 \cdot \boldsymbol{k}_0 - r^2, \tag{B.9a}$$

$$\ell_1 = \boldsymbol{k}_0 \cdot \boldsymbol{k}_1, \tag{B.9b}$$

$$\ell_2 = \boldsymbol{k}_1 \cdot \boldsymbol{k}_1, \tag{B.9c}$$

the solution to this equation for s is

$$s_{int} = \frac{-\ell_1 \pm \sqrt{\ell_1^2 - \ell_0 \ell_2}}{\ell_2}, \tag{B.10}$$

which is the distance to collision of the particle with the cylinder. If $l_2 = 0$, the particle continues along the axis of the cylinder without intersection. If $\ell_1^2 - \ell_0 \ell_2 < 0$, there are no intersections between the particle and the surface.

The aspect of a particle with respect to a cylinder can be found by setting $\boldsymbol{k}_1 = \boldsymbol{0}$ in Eq. (B.6),

$$\boldsymbol{k}_0 \cdot \boldsymbol{k}_0 = r^2, \tag{B.11}$$

and evaluating the sign of the equation for position $\boldsymbol{g}_0$ in terms of the constant ℓ_0 [Eq. (B.9a)],

$$\ell_0 \begin{cases} > 0, & \text{outside}, \\ < 0, & \text{inside}, \\ = 0, & \text{coincident}. \end{cases} \tag{B.12}$$

Finally, the normal direction of a particle on a cylinder at a point $\boldsymbol{g}_0$ can be found by normalizing the $\boldsymbol{k}_0$ vector,

$$\boldsymbol{n} = \frac{\boldsymbol{k}_0}{\|\boldsymbol{k}_0\|}. \tag{B.13}$$

B.3 Sphere

The equation of a sphere is

$$\|\boldsymbol{x} - \boldsymbol{x}_0\|^2 = r^2. \tag{B.14}$$

Inserting the particle position from Eq. (B.2) gives

$$\|\boldsymbol{g}_0 + \boldsymbol{g}_1 s - \boldsymbol{x}_0\|^2 = r^2,$$

which can be simplified by defining

$$\boldsymbol{k}_0 = \boldsymbol{g}_0 - \boldsymbol{x}_0, \tag{B.15a}$$

$$\boldsymbol{k}_1 = \boldsymbol{g}_1, \tag{B.15b}$$

to get

$$\|\boldsymbol{k}_0 + \boldsymbol{k}_1 s\|^2 = r^2. \tag{B.16}$$

This form of the equation is identical to the cylindrical equations and the results from the cylindrical equations follow, but the spherical equations can be simplified by noting that $\boldsymbol{g}_1 \cdot \boldsymbol{g}_1 = 1$, which results in a distance to intersection of

$$s_{int} = -\ell_1 \pm \sqrt{\ell_1^2 - \ell_0}, \tag{B.17}$$

with

$$\ell_0 = \boldsymbol{k}_0 \cdot \boldsymbol{k}_0 - r^2, \tag{B.18a}$$

$$\ell_1 = \boldsymbol{k}_0 \cdot \boldsymbol{k}_1. \tag{B.18b}$$

If $\ell_1^2 - \ell_0 < 0$, there are no intersections.

To check if a particle is inside of the sphere, the ℓ_0 constant is used to get

$$\ell_0 \begin{cases} > 0, & \text{outside}, \\ < 0, & \text{inside}, \\ = 0, & \text{coincident}, \end{cases} \tag{B.19}$$

as for the cylindrical case. The normal direction of a point on the sphere at point $\boldsymbol{g}_0$ is

$$\boldsymbol{n} = \frac{\boldsymbol{k}_0}{\|\boldsymbol{k}_0\|}. \tag{B.20}$$

B.4 Ellipsoid

The equations for the ellipsoid are similar to the spherical and cylindrical equations. Defining

$$\boldsymbol{q} = \left\{ \frac{1}{a}, \frac{1}{b}, \frac{1}{c} \right\}, \tag{B.21}$$

the equation for an ellipsoid with semiaxes a, b, c along the Cartesian axes is

$$\|(\boldsymbol{x} - \boldsymbol{x}_0) \circ \boldsymbol{q}\| = 1, \tag{B.22}$$

where $\circ$ represents the entrywise product. The equation for a traveling particle is inserted into this equation, which results in

$$\|(\boldsymbol{g}_0 + \boldsymbol{g}_1 s - \boldsymbol{x}_0) \circ \boldsymbol{q}\| = 1. \tag{B.23}$$

Letting

$$\boldsymbol{k}_0 = (\boldsymbol{g}_0 - \boldsymbol{x}_0) \circ \boldsymbol{q}, \tag{B.24a}$$

$$\boldsymbol{k}_1 = \boldsymbol{g}_1 \circ \boldsymbol{q}, \tag{B.24b}$$

the equations become

$$\|\boldsymbol{k}_0 + \boldsymbol{k}_1 s\| = 1. \tag{B.25}$$

The results from the equations for a cylinder with the same equation follow. The constants

$$\ell_0 = \boldsymbol{k}_0 \cdot \boldsymbol{k}_0 - 1, \tag{B.26a}$$

$$\ell_1 = \boldsymbol{k}_0 \cdot \boldsymbol{k}_1, \tag{B.26b}$$

$$\ell_2 = \boldsymbol{k}_1 \cdot \boldsymbol{k}_1, \tag{B.26c}$$

simplify solution of the equation for s,

$$s = \frac{-\ell_1 \pm \sqrt{\ell_1^2 - \ell_0 \ell_2}}{\ell_2}, \tag{B.27}$$

as before. This is the distance to collision with the ellipsoid. As for the cylinder, if $\ell_1^2 - \ell_0\ell_2 < 0$ there are no intersections. The aspect of the particle with respect to the surface is also the same,

$$\ell_0 \begin{cases} > 0, & \text{outside,} \\ < 0, & \text{inside,} \\ = 0, & \text{coincident.} \end{cases} \tag{B.28}$$

To find the normal direction for a point $\boldsymbol{g}_0$ on the surface of the ellipsoid, the property that the gradient vector of surface is perpendicular to the surface is employed. The gradient has the magnitude

$$\begin{aligned} \boldsymbol{N} &= \nabla \left[\|(\boldsymbol{x} - \boldsymbol{x}_0) \circ \boldsymbol{q}\| - 1 \right] \\ &= \frac{(\boldsymbol{g}_0 - \boldsymbol{x}_0) \circ \boldsymbol{q} \circ \boldsymbol{q}}{\|(\boldsymbol{g}_0 - \boldsymbol{x}_0) \circ \boldsymbol{q}\|} \\ &= \frac{\boldsymbol{k}_0 \circ \boldsymbol{q}}{\|\boldsymbol{k}_0\|}, \end{aligned} \tag{B.29}$$

which when normalized is the normal direction of the surface,

$$\boldsymbol{n} = \frac{\boldsymbol{N}}{\|\boldsymbol{N}\|}. \tag{B.30}$$

Appendix C

Alternate discretization of the strong form of the transport equation

The basis and weight functions for the strong-form transport equation in Sec. 2.9 are either compact, which produces a sparse transport operator ($\mathcal{L}^{-1}$), or global, which produces a dense transport operator. For problems with a large number of spatial points, the use of global basis functions makes the method very computationally expensive. For instance, the multiquadric function [Eq. (2.66)] actually grows larger as the distance from its center increases. The following is a method based on Ref. [10] for restricting the range of these global RBF functions by changing the solution variable of the $\mathcal{L}^{-1}$ calculation from the coefficients of angular flux expansion to the angular flux itself at the basis and weight function centers.

C.1 Derivation

The same strong-form equations with point cross section weighting [Eqs. (2.62)] are used for this discretization. This linear system can be localized by restricting the basis for each equation to a certain number of nearest neighbors, which creates a sparse matrix and a better-conditioned system. This would ordinarily create discontinuities in the solution where the basis functions end. However, if only the values at the basis function centers are needed, then the discontinuities do not present an issue. Letting G_j be the set of the chosen nearest basis functions to the point $\boldsymbol{x}_j$, the basis function expansion [Eq. (2.30a)] becomes

$$\psi_{n,g}\left(\boldsymbol{x}_j\right) = \sum_{i \in G_j} \alpha_{i,n,g} b_i\left(\boldsymbol{x}\right). \tag{C.1}$$

Letting $\boldsymbol{A}_{n,g}$ be the sparse $\mathcal{L}$ matrix (or streaming operator) for each ordinate n and group g, the linear system can be described by the equations

$$\boldsymbol{A}_{n,g}\boldsymbol{\alpha}_{n,g} = \boldsymbol{q}_{n,g}. \tag{C.2}$$

The values of the rows $\boldsymbol{a}_{j,n,g}$ of the matrix $\boldsymbol{A}_{n,g}$,

$$\boldsymbol{A}_{n,g} = \begin{bmatrix} \boldsymbol{a}_{1,n,g}^{\top} \\ \boldsymbol{a}_{2,n,g}^{\top} \\ \vdots \\ \boldsymbol{a}_{I,n,g}^{\top} \end{bmatrix}, \tag{C.3}$$

can be written as

$$\left(\boldsymbol{a}_{j,n,g}\right)_i = \begin{cases} \mathcal{L}_{n,g} b_i\left(\boldsymbol{x}_j\right), & \boldsymbol{x}_j \in V, \\ \mathcal{B}_{n,g} b_i\left(\boldsymbol{x}_j\right), & \boldsymbol{x}_j \in \partial V, \end{cases} \tag{C.4}$$

where $\mathcal{B}_{n,g}$ is the boundary operator representing Eq. (2.62d) and ∂V and V are the boundary and interior of the spatial domain, respectively.

The system in Eq. (C.2) could be solved directly, but the matrix would be ill-conditioned due to the use of RBF functions that may be very flat (see Chapter 1). Alternately, each row of this equation can be converted separately to produce a solution for the vector of angular flux values at the center points of the basis functions, $\boldsymbol{\psi}_{n,g}$. The basis function expansion [Eq. (C.1)] can be written as a linear system,

$$\boldsymbol{\psi}_{n,g} = \boldsymbol{\Gamma}_j \boldsymbol{\alpha}_{n,g}, \tag{C.5}$$

where $\boldsymbol{\Gamma}_j$ is the basis function distance matrix for the N local points for basis function j with indices j_n,

$$\boldsymbol{\Gamma}_j = \begin{bmatrix} b_{j_1}\left(\boldsymbol{x}_{j_1}\right) & \cdots & b_{j_N}\left(\boldsymbol{x}_{j_1}\right) \\ b_{j_1}\left(\boldsymbol{x}_{j_2}\right) & \cdots & b_{j_N}\left(\boldsymbol{x}_{j_2}\right) \\ \vdots & \ddots & \vdots \\ b_{j_1}\left(\boldsymbol{x}_{j_N}\right) & \cdots & b_{j_N}\left(\boldsymbol{x}_{j_N}\right) \end{bmatrix},$$

and where the $\boldsymbol{\alpha}_{n,g}$ vector is appropriately indexed to be of the same size as the number of local points. Applying the inverse of this matrix to Eq. (C.5), the equation for a single row of the matrix becomes

$$\boldsymbol{a}_{j,n,g}^{\top} \boldsymbol{\alpha}_{n,g} = \boldsymbol{a}_{j,n,g}^{\top} \boldsymbol{\Gamma}_j^{-1} \boldsymbol{\psi}_{n,g}. \tag{C.6}$$

From these equations, the matrix describing the $\mathcal{L}$ operator $\boldsymbol{B}_{n,g}$ for $\boldsymbol{\psi}_{n,g}$ can be written as

$$\boldsymbol{b}_{j,n,g} = \left(\boldsymbol{a}_{j,n,g}^{\top}\boldsymbol{\Gamma}_j^{-1}\right)^{\top} = \left(\boldsymbol{\Gamma}_j^{\top}\right)^{-1}\boldsymbol{a}_{j,n,g}, \tag{C.7}$$

which converts the linear system of equations to

$$\boldsymbol{B}_{n,g}\boldsymbol{\psi}_{n,g} = \boldsymbol{q}_{n,g}. \tag{C.8}$$

The $\boldsymbol{\Gamma}$ matrix is the same for every group and direction, and so the LU decomposition of the $\boldsymbol{\Gamma}_j$ matrix can be stored and reused, making the cost of inversion of $\boldsymbol{\Gamma}_j$ onto $\boldsymbol{a}_{j,n,g}$ comparable in time to the calculation of the values of the angular flux from the basis function expansion. The switch to a solution for $\boldsymbol{\psi}_{n,g}$ can produce a matrix $\boldsymbol{B}_{n,g}$ with a lower condition number than $\boldsymbol{A}_{n,g}$, while the switch to a local basis makes the condition number for $\boldsymbol{\Gamma}_j$ lower than that of either $\boldsymbol{A}_{n,g}$ or $\boldsymbol{B}_{n,g}$.

As global RBFs are used in this section, the RBF shape parameter is altered from Eq. (2.72) to be

$$\epsilon_i = \frac{k}{\Delta r}, \tag{C.9}$$

where Δr is the average distance to the nearest points (two in 1D, four in 2D, six in 3D) and k is a chosen constant.

Compared to the weak-form equations or even for the strong-form equations presented in Chapter 2, this discretization depends much more on the "correct" choice of parameters to get a good solution. This discretization does successfully solve the slab geometry problem in Sec. C.2, but struggles with problems in two and three dimensions, as shown in Sec. C.3.

C.2 Results for a slab-geometry problem

The slab-geometry results in this section are reproduced from Ref. [94]. The problem under consideration is a one-dimensional, isotropically-scattering slab with a reflective boundary on the left side of the problem, a fuel cell from 0 to X_{fuel}, a moderator cell from X_{fuel} to $X_{fuel}+X_{mod}$ and a vacuum boundary on the right side of the problem. The lengths of the fuel and moderator are $X_{fuel} = 1.0$ and $X_{mod} = 2.0$. The two-group cross section values for the problem are listed in Table C.1. The internal, isotropic source is given in moment form, so the discrete value of the internal source is $q = Q/2$ for the slab-geometry (azimuthally-integrated) problem. For all cases, the S_N approximation is used in angle with 16 Gauss-Legendre discrete ordinates.

In the fuel area, the dominant process is within-group scattering for both of the groups, with a comparatively higher fission and absorption cross section in group 2. In the moderator area, group 1 has a very high

Table C.1: Cross sections and internal source for strong-form slab results

Material	Group	$\Sigma_{t,g}$	$\Sigma_{s,g\to1}$	$\Sigma_{s,g\to2}$	χ_g	v_g	$\Sigma_{f,g}$	Q
Fuel	1	1.0	0.84	0.09	1.0	2.4	0.003	1.0
	2	2.0	0.0	0.4	0.0	2.4	1.1	0.0
Moderator	1	1.0	0.2	0.8	0.0	0.0	0.0	0.0
	2	2.0	0.0	1.9	0.0	0.0	0.0	0.0

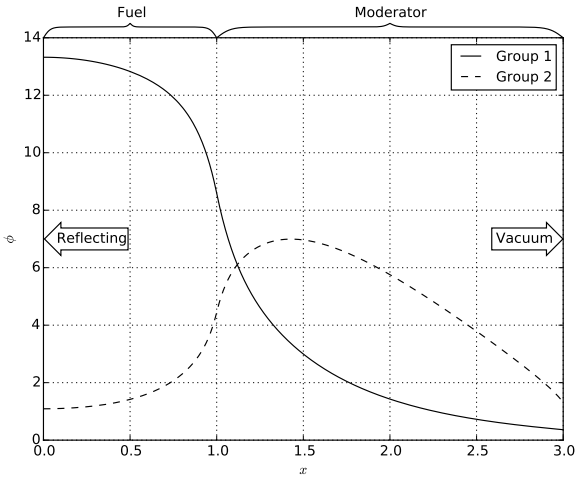


Figure C.1: Benchmark solution for strong-form slab results

probability of scattering into group 2, while group 2 has a very high probability of within-group scattering. This was chosen to represent the effects of fast neutrons being moderated into thermal energies.

The benchmark solution is calculated using a 1D discontinuous finite element (DFEM) code with linear basis and weight functions. The fuel and moderator regions have, respectively, 4000 and 8000 cells with two nodes per cell for the first set of tests. For the second set of tests, these numbers are 12000 and 24000 cells to accommodate the varying number of points in the RBF input. The benchmark solution can be seen in Fig. C.1. The DFEM and RBF solution routines use the same code except for the spatial discretization and $\mathcal{L}^{-1}$ calculation.

For the first set of tests, the number of spatial points for the RBF method is 1202, including the two boundary points. The number of neighbors is 50, including the current point. The shape multiplier k is altered logarithmically between $k = 0.05$ and $k = 3.0$ for each of the three global basis functions described in Sec. 2.10.1, including the multiquadric (MQ), Gaussian (GA) and inverse multiquadric (IMQ). The L_2 error in the scalar flux is then calculated using coincident points from the DFEM solution. The results can be seen in Fig. C.2. The MQ basis function produces the best and most reliable results. For the range between $k = 0.2$ and $k = 3.0$, the error in the solution is at below 1.69×10^{-4}. Between $k = 0.35$ and $k = 0.5$,

the maximum error found is 2.53×10^{-5}, with the minimum of $k = 2.38 \times 10^{-5}$ occurring at $k = 0.391$. The MQ basis with localized basis functions provides a wide range of good choices for the shape parameter. Comparatively, the GA and IMQ basis functions are not nearly so well-behaved. The minimum error of 8.41×10^{-4} for the GA basis occurs at $k = 0.428$ and the error is at or below 2.02×10^{-3} for the region from $k = 0.35$ to $k = 0.67$.

Particularly for the GA and IMQ basis functions, using a shape parameter higher or lower than the bounds of certain stability regions makes the error jump up relatively quickly. The errors for too high or low of shape parameters are two separate phenomena. In the limit of $k \to 0$, for a homogeneous problem the series expansion for the RBF solution should agree to spectral accuracy with the analytic solution to the problem. However, as the shape parameter decreases, the system becomes extremely ill-conditioned, causing more and more roundoff error. As the shape parameter gets large, even the first-order terms in the series expansion disagree with the analytic solution, creating a different type of error in the solution. The MQ basis functions cannot escape the ill-conditioning as $k \to 0$ but aren't nearly as affected by high values of k because they limit to a linear function as $r \to 0$, whereas the GA and IMQ functions limit to zero.

The number of iterations required to converge the first-flight flux $\mathcal{L}^{-1}q$ is three for all the cases. The number of iterations required to converge the full transport problem for source iteration and Krylov iteration for the DFEM method is 421 and 15, respectively. This number is identical to the number of iterations taken by an RBF solution with an appropriately-chosen shape parameter. As the error of the RBF method increases, the number of iterations required also increases. For the GA basis, problems with errors above 3.4×10^{-3} take between 16 and 110 Krylov iterations to converge. For the MQ basis, problems with errors of 7.3×10^{-5} and above take between 16 and 24 iterations, with the worst offenders being the problems with very small shape parameters. The IMQ produce similar numbers for problems with errors above 5.2×10^{-4}, for which the number of iterations varies between 16 and 25.

For the second set of tests, the shape parameter is held constant at 0.5 and the number of neighbors is held at 50 while the number of internal points is varied. Using fewer than about 50 neighbors tends to degrade the solution, while using more adds significant computational cost. Figure C.3 shows the error decreasing as the number of points increases. Applying a fit of the form ax^b to the MQ data, with x the number of points, the coefficients equal $a = 1.56$ and $b = -1.56$. Empirically, the order of the method as the distance between points increases lies somewhere between first and second-order globally. Performing the same fit on data for the MQ method with global basis functions, the coefficients become $a = 1.19$ and $b = -1.49$, which means that the order of the method is approximately the same. However, the time to perform the calculation is much higher.

The timing for the 600-point problem can be seen in Table C.2. Krylov iteration speeds up the solution

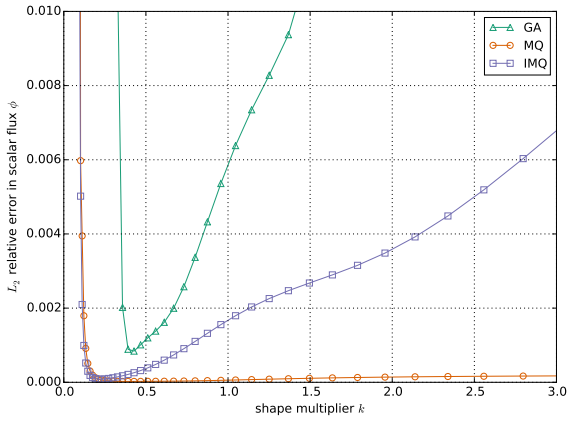


Figure C.2: L_2 relative error in the scalar flux with varying shape parameter for strong-form slab

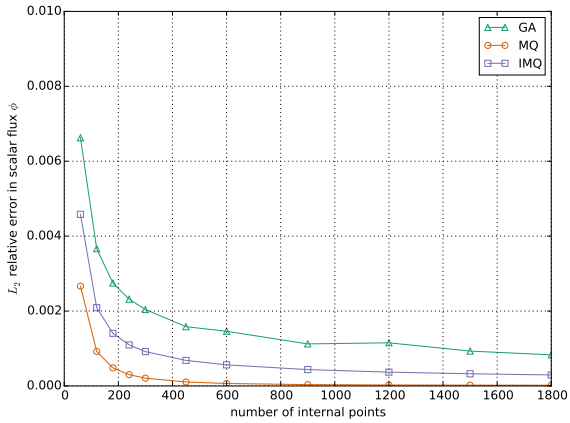


Figure C.3: L_2 relative error in the scalar flux with spatial refinement for strong-form slab

Table C.2: Total runtime for alternate strong-form results

	DFEM	RBF Global	RBF Local
SI	1.53 sec	664.56 sec	72.60 sec
KI	0.08 sec	63.15 sec	6.91 sec

procedure by more than an order of magnitude, while switching to localized basis functions speeds up the calculation by almost another order of magnitude. In the limit of many points, the localized method is second-order in time, while the global method is third-order in time, meaning the timing discrepancy only increases as more points are added. The global method does not have significantly better error properties for these test cases.

C.3 Qualitative results for a simple pincell

This problem is similar to the one-dimensional problem presented in Sec. C.2 and has the same materials from Table C.1, but uses the pincell geometry as shown in Fig. C.4. The fuel region has a radius of 1.0 cm, while the pincell length is 4.0 cm. The points are placed randomly within the problem, ensuring that no two points are too close. This is done to show the irregularities in the solution that emerge from solving the strong transport equations. The shape function parameter is set to $k = 2.0$ for these problems.

Figure C.5 shows the fast and thermal-group solutions for 4039 spatial points and 16 directions. The solution is asymmetric about the pincell center and and oscillates near the fuel/moderator boundary, which indicates poor conditioning of the solution due to the advection in the streaming operator. The oscillations near the pincell boundary have a wavelength approximately equal to two times the point spacing in the

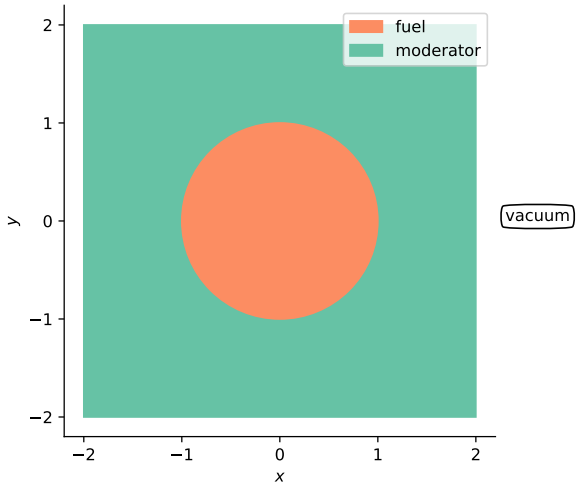
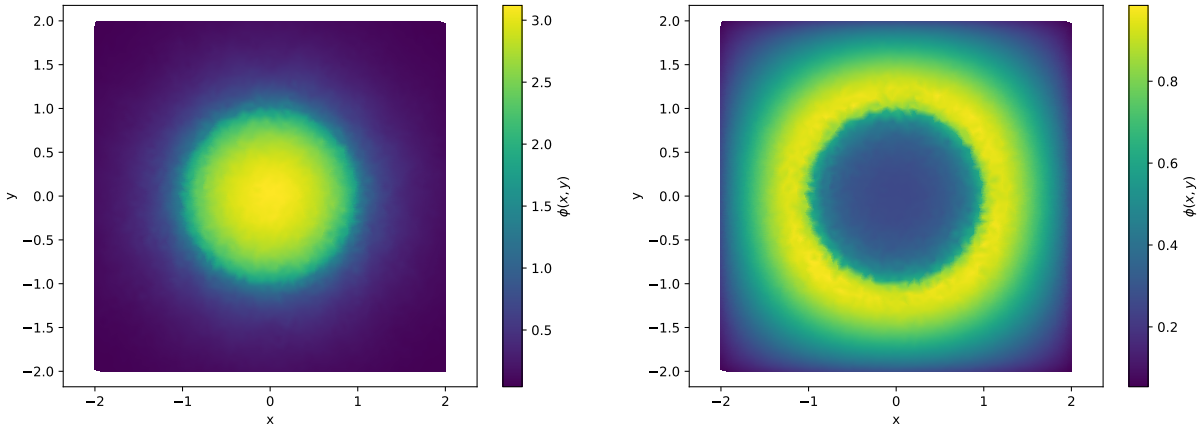


Figure C.4: Geometry for strong-form pincell results



(a) Fast group

(b) Thermal group

Figure C.5: Scalar flux for strong-form pincell problem

problem.

For one dimension, the strong-form equations can perform well if the "correct" shape parameter is chosen and enough points are used. In two dimensions, the additional geometric complexities and solution cost make the collocation method difficult to apply accurately.

References

[1] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013–1024, 1977.

[2] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, 1977.

[3] CW Hirt, Anthony A Amsden, and JL Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14(3):227–253, 1974.

[4] MB Liu and GR Liu. Smoothed particle hydrodynamics (SPH): an overview and recent developments. *Archives of Computational Methods in Engineering*, 17(1):25–76, 2010.

[5] Wing Kam Liu, Sukky Jun, Shaofan Li, Jonathan Adee, and Ted Belytschko. Reproducing kernel particle methods for structural dynamics. *International Journal for Numerical Methods in Engineering*, 38(10):1655–1679, 1995.

[6] E Onate, S Idelsohn, OC Zienkiewicz, and RL Taylor. A finite point method in computational mechanics: Applications to convective transport and fluid flow. *International Journal for Numerical Methods in Engineering*, 39(22):3839–3866, 1996.

[7] Gui-Rong Liu and YuanTong Gu. A point interpolation method for two-dimensional solids. *International Journal for Numerical Methods in Engineering*, 50(4):937–951, 2001.

[8] Edward J Kansa. Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—ii solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & Mathematics with Applications*, 19(8):147–161, 1990.

[9] Gregory E Fasshauer. Solving partial differential equations by collocation with radial basis functions. In *Proceedings of Chamonix*, volume 1997, pages 1–8. Citeseer, 1996.

[10] Wen Chen, Zhuo-Jia Fu, and Ching-Shyang Chen. *Recent advances in radial basis function collocation methods*. Springer, 2014.

[11] Bengt Fornberg and Julia Zuev. The Runge phenomenon and spatially variable shape parameters in RBF interpolation. *Computers & Mathematics with Applications*, 54(3):379–398, 2007.

[12] B Nayroles, G Touzot, and P Villon. Generalizing the finite element method: diffuse approximation and diffuse elements. *Computational Mechanics*, 10(5):307–318, 1992.

[13] Ted Belytschko, Yun Yun Lu, and Lei Gu. Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering*, 37(2):229–256, 1994.

[14] Satya N Atluri and Tulong Zhu. A new meshless local petrov-galerkin (mlpg) approach in computational mechanics. *Computational Mechanics*, 22(2):117–127, 1998.

[15] TP Fries and HG Matthies. A review of petrov-galerkin stabilization approaches and an extension to meshfree methods. informatikbericht-nr. 2004-01, technical university of braunschweig, 2004.

[16] H Lin and SN Atluri. Meshless local Petrov-Galerkin method for convection-diffusion problems. *CMES (Computer Modelling in Engineering & Sciences)*, 1(2):45–60, 2000.

[17] Gui-Rong Liu. *Meshfree methods: moving beyond the finite element method*. Taylor & Francis, 2009.

[18] Volker Springel. Smoothed particle hydrodynamics in astrophysics. *Annual Review of Astronomy and Astrophysics*, 48:391–430, 2010.

[19] Burke Ritchie, Pieter G Dykema, and Dennis Braddy. Use of fast Fourier transform computational methods in radiation transport. *Physical Review E*, 56(2):2217, 1997.

[20] Arnold D Kim and Miguel Moscoso. Chebyshev spectral methods for radiative transfer. *SIAM Journal on scientific computing*, 23(6):2074–2094, 2002.

[21] Hamou Sadat. On the use of a meshless method for solving radiative transfer with the discrete ordinates formulations. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 101(2):263–268, 2006.

[22] LH Liu and JY Tan. Least-squares collocation meshless approach for radiative heat transfer in absorbing and scattering media. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 103(3):545–557, 2007.

[23] Manuel Kindelan, Francisco Bernal, Pedro González-Rodríguez, and Miguel Moscoso. Application of the RBF meshless method to the solution of the radiative transport equation. *Journal of Computational Physics*, 229(5):1897–1908, 2010.

[24] Hamou Sadat, Cheng-An Wang, and Vital Le Dez. Meshless method for solving coupled radiative and conductive heat transfer in complex multi-dimensional geometries. *Applied Mathematics and Computation*, 218(20):10211–10225, 2012.

[25] LH Liu and JY Tan. Meshless local Petrov-Galerkin approach for coupled radiative and conductive heat transfer. *International journal of thermal sciences*, 46(7):672–681, 2007.

[26] S Kashi, A Minuchehr, A Zolfaghari, and B Rokrok. Mesh-free method for numerical solution of the multi-group discrete ordinate neutron transport equation. *Annals of Nuclear Energy*, 106:51–63, 2017.

[27] B Rokrok, H Minuchehr, and A Zolfaghari. Element-free Galerkin modeling of neutron diffusion equation in x–y geometry. *Annals of Nuclear Energy*, 43:39–48, 2012.

[28] Tayfun Tanbay and Bilge Ozgener. Numerical solution of the multigroup neutron diffusion equation by the meshless RBF collocation method. *Mathematical and Computational Applications*, 18(3):399–407, 2013.

[29] JY Tan, JM Zhao, LH Liu, and YY Wang. Comparative study on accuracy and solution cost of the first/second-order radiative transfer equations using the meshless method. *Numerical Heat Transfer, Part B: Fundamentals*, 55(4):324–337, 2009.

[30] Cheng-An Wang, Hamou Sadat, Vital Ledez, and Denis Lemonnier. Meshless method for solving radiative transfer problems in complex two-dimensional and three-dimensional geometries. *International Journal of Thermal Sciences*, 49(12):2282–2288, 2010.

[31] JM Zhao, JY Tan, and LH Liu. A second order radiative transfer equation and its solution by meshless method with application to strongly inhomogeneous media. *Journal of Computational Physics*, 232(1):431–455, 2013.

[32] Cheng-An Wang, Hamou Sadat, and Jian-Yu Tan. Meshless method for solving coupled radiative and conductive heat transfer in refractive index medium. In *Journal of Physics: Conference Series*, volume 676, page 012024. IOP Publishing, 2016.

[33] Rodolphe Turpault. A consistent multigroup model for radiative transfer and its underlying mean opacities. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 94(3-4):357–371, 2005.

[34] Forrest B Brown and William R Martin. Direct sampling of Monte Carlo flight paths in media with continuously varying cross-sections. In *Proc. ANS Mathematics & Computation Topical Meeting*, volume 2, 2003.

[35] Matthew Ellis, Derek Gaston, Benoit Forget, and Kord Smith. Preliminary coupling of the Monte Carlo code OpenMC and the Multiphysics Object-Oriented Simulation Environment for analyzing Doppler feedback in Monte Carlo simulations. *Nuclear Science and Engineering*, 185(1):184–193, 2017.

[36] Sebastian Schunert, Yaqi Wang, Vincent Laboure, Javier Ortensi, and Mark DeHart. Approximate transport sweeps for DFEM on higher order meshes with spatially varying cross sections. In *Transactions of the American Nuclear Society, Vol. 117*, 2017.

[37] Mehdi Dehghan and Ali Shokri. A meshless method for numerical solution of a linear hyperbolic equation with variable coefficients in two space dimensions. *Numerical Methods for Partial Differential Equations*, 25(2):494–506, 2009.

[38] Vinh Phu Nguyen, Timon Rabczuk, Stéphane Bordas, and Marc Duflot. Meshless methods: a review and computer implementation aspects. *Mathematics and Computers in Simulation*, 79(3):763–813, 2008.

[39] Jichun Li and YC Hon. Domain decomposition for radial basis meshless methods. *Numerical Methods for Partial Differential Equations*, 20(3):450–462, 2004.

[40] Marvin L Adams. Discontinuous finite element transport solutions in thick diffusive problems. *Nuclear Science and Engineering*, 137(3):298–333, 2001.

[41] Changyuan Liu. *Discrete ordinates methods for transport problems with curved spatial grids.* PhD thesis, University of Michigan, 2015.

[42] Douglas N Woods. High order finite elements S_N transport in X-Y geometry on meshes with curved surfaces in the thick diffusion limit. Master's thesis, Oregon State University, 2016.

[43] Yaqi Wang. Nonlinear diffusion acceleration for the multigroup transport equation discretized with S_N and continuous FEM with Rattlesnake. Technical report, American Nuclear Society, 2013.

[44] Elmer Eugene Lewis and Warren F Miller. *Computational methods of neutron transport.* John Wiley and Sons, Inc., New York, NY, 1984.

[45] James S Warsa, Todd A Wareing, and Jim E Morel. Krylov iterative methods and the degraded effectiveness of diffusion synthetic acceleration for multidimensional S_N calculations in problems with material discontinuities. *Nuclear Science and Engineering*, 147(3):218–248, 2004.

[46] James S Warsa, Todd A Wareing, Jim E Morel, John M McGhee, and Richard B Lehoucq. Krylov subspace iterations for deterministic k-eigenvalue calculations. *Nuclear Science and Engineering*, 147(1):26–42, 2004.

[47] Holger Wendland. Error estimates for interpolation by compactly supported radial basis functions of minimal degree. *Journal of Approximation Theory*, 93(2):258–272, 1998.

[48] Gui-Rong Liu and Yuan-Tong Gu. *An introduction to meshfree methods and their programming.* Springer Science & Business Media, 2005.

[49] Joshua J Jarrell and Marvin L Adams. Discrete-ordinates quadrature sets based on linear discontinuous finite elements. In *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011), Rio de Janeiro, RJ, Brazil*, 2011.

[50] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[51] Jose Luis Blanco. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with k-d trees, 2014.

[52] Michael A Heroux, Roscoe A Bartlett, Vicki E Howle, Robert J Hoekstra, Jonathan J Hu, Tamara G Kolda, Richard B Lehoucq, Kevin R Long, Roger P Pawlowski, Eric T Phipps, et al. An overview of the Trilinos project. 31(3):397–423.

[53] Raymond E Alcouffe. Diffusion synthetic acceleration methods for the diamond-differenced discrete-ordinates equations. *Nuclear Science and Engineering*, 64(2):344–355, 1977.

[54] Matthew T Calef, Erin D Fichtl, James S Warsa, Markus Berndt, and Neil N Carlson. Nonlinear krylov acceleration applied to a discrete ordinates formulation of the k-eigenvalue problem. *Journal of Computational Physics*, 238:188–209, 2013.

[55] Steven P Hamilton. *Numerical solution of the k-eigenvalue problem*. PhD thesis, Emory University, 2011.

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

[57] Suvranu De and Klaus-Jürgen Bathe. The method of finite spheres with improved numerical integration. *Computers & Structures*, 79(22-25):2183–2196, 2001.

[58] Pravin Madhavan. Numerical integration in meshfree methods. Master's thesis, University of Oxford, 2010.

[59] Donat Racz and Tinh Quoc Bui. Novel adaptive meshfree integration techniques in meshless methods. *International Journal for Numerical Methods in Engineering*, 90(11):1414–1434, 2012.

[60] Amir Khosravifard and Mohammad Rahim Hematiyan. A new method for meshless integration in 2d and 3d galerkin meshfree methods. *Engineering Analysis with Boundary Elements*, 34(1):30–40, 2010.

[61] Wolfram Research, Inc. Mathematica, Version 10.0. Champaign, IL, 2018.

[62] Paul K Romano and Benoit Forget. The OpenMC monte carlo particle transport code. *Annals of Nuclear Energy*, 51:274–281, 2013.

[63] Keisuke Kobayashi, Naoki Sugimura, and Yasunobu Nagaya. 3D radiation transport benchmark problems and results for simple geometries with void region. *Progress in Nuclear Energy*, 39(2):119–144, 2001.

[64] MB Chadwick, M Herman, P Obložinskỳ, Michael E Dunn, Y Danon, AC Kahler, Donald L Smith, B Pritychenko, Goran Arbanas, R Arcilla, et al. ENDF/B-VII. 1 nuclear data for science and technology: cross sections, covariances, fission product yields and decay data. 112(12):2887–2996.

[65] Andrew T Godfrey. Vera core physics benchmark progression problem specifications. *Consortium for Advanced Simulation of LWRs*, 2014.

[66] Roger W. Brewer. Benchmark critical experiment of a thorium reflected plutonium sphere. Los Alamos National Laboratory Report LA-UR-94-3275.

[67] J Blair Briggs, Lori Scott, and Ali Nouri. The International Criticality Safety Benchmark Evaluation Project. *Nuclear Science and Engineering*, 145(1):1–10, 2003.

[68] Richard Swinbank and R James Purser. Fibonacci grids: A novel approach to global modelling. *Quarterly Journal of the Royal Meteorological Society*, 132(619):1769–1793, 2006.

[69] M Zerroukat, H Power, and CS Chen. A numerical method for heat transfer problems using collocation and radial basis functions. *International Journal for Numerical Methods in Engineering*, 42(7):1263–1278, 1998.

[70] Xue-Hong Wu and Wen-Quan Tao. Meshless method based on the local weak-forms for steady-state heat conduction problems. *International Journal of Heat and Mass Transfer*, 51(11-12):3103–3112, 2008.

[71] WJ Minkowycz, Ephraim M Sparrow, Jayathi Y Murthy, and John P Abraham. *Handbook of numerical heat transfer*. John Wiley & Sons, Inc., 2009.

[72] Jin Yan, Brendan Kochunas, Mathieu Hursin, Thomas Downar, Zeses Karoutas, and Emilio Baglietto. Coupled computational fluid dynamics and MOC neutronic simulations of westinghouse PWR fuel assemblies with grid spacers. In *The 14th International Topical Meeting on Nuclear Reactor Thermalhydraulics*, 2011.

[73] Derek R Gaston, Cody J Permann, John W Peterson, Andrew E Slaughter, David Andrš, Yaqi Wang, Michael P Short, Danielle M Perez, Michael R Tonks, Javier Ortensi, et al. Physics-based multiscale coupling for full core nuclear reactor simulation. *Annals of Nuclear Energy*, 84:45–54, 2015.

[74] JD Hales, MR Tonks, FN Gleicher, BW Spencer, SR Novascone, RL Williamson, G Pastore, and DM Perez. Advanced multiphysics coupling for LWR fuel performance analysis. *Annals of Nuclear Energy*, 84:98–110, 2015.

[75] Theodore L Bergman, Frank P Incropera, David P DeWitt, and Adrienne S Lavine. *Fundamentals of heat and mass transfer*. John Wiley & Sons, 2011.

[76] Kambiz Salari and Patrick Knupp. Code verification by the method of manufactured solutions. Technical report, Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US), 2000.

[77] Dan Gabriel Cacuci. *Handbook of Nuclear Engineering: Vol. 1: Nuclear Engineering Fundamentals; Vol. 2: Reactor Design; Vol. 3: Reactor Analysis; Vol. 4: Reactors of Generations III and IV; Vol. 5: Fuel Cycles, Decommissioning, Waste Disposal and Safeguards*. Springer Science & Business Media, 2010.

[78] Dermott E Cullen and Charles R Weisbin. Exact doppler broadening of tabulated cross sections. *Nuclear Science and Engineering*, 60(3):199–229, 1976.

[79] Gokhan Yesilyurt, William R Martin, and Forrest B Brown. On-the-fly doppler broadening for monte carlo codes. *Nuclear Science and Engineering*, 171(3):239–257, 2012.

[80] DG Madland. Total prompt energy release in the neutron-induced fission of 235u, 238u, and 239pu. *Nuclear Physics A*, 772(3-4):113–137, 2006.

[81] Weston M Stacey. *Nuclear reactor physics*. John Wiley & Sons, 2018.

[82] R Byron Bird, Warren E Stewart, and Edwin N Lightfoot. *Transport Phenomena, 2nd Edition*. New York: Wiley, 2002.

[83] International Atomic Energy Agency. *Thermophysical properties database of materials for light water reactors and heavy water reactors*. International Atomic Energy Agency, 2006.

[84] K Lassmann and F Hohlefeld. The revised urgap model to describe the gap conductance between fuel and cladding. *Nuclear Engineering and Design*, 103(2):215–221, 1987.

[85] Igor Pioro. *Handbook of generation IV nuclear reactors*. Woodhead Publishing, 2016.

[86] Shane G Stimpson, Jeffrey J Powers, Kevin T Clarno, Roger Pawlowski, and Ryan Bratton. Assessment of pellet-clad interaction indicators in Watts Bar Unit 1 using the VERA Framework. Technical report, Oak Ridge National Laboratory (ORNL), 2015.

[87] James J Duderstadt and Louis J Hamilton. *Nuclear reactor analysis*, volume 1. Wiley New York, 1976.

[88] Watts Bar Unit. Watts Bar Nuclear Plant, Unit 2 - Amendment 99 to Final Safety Analysis Report (FSAR), Section 4, ML101610356. *URL: https://www.nrc.gov/docs/ML1016/ML101610356.pdf*, 2010.

[89] Van Emden Henson and Ulrike Meier Yang. Boomeramg: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155–177, 2002.

[90] Yaqi Wang, Mark David DeHart, Derek Ray Gaston, Frederick Nathan Gleicher, Richard Charles Martineau, John William Peterson, and Sebastian Schunert. Convergence study of Rattlesnake solutions for the two-dimensional C5G7 MOX benchmark. Technical report, Idaho National Laboratory (INL), Idaho Falls, ID (United States), 2015.

[91] SE Mousavi and N Sukumar. Generalized gaussian quadrature rules for discontinuities and crack singularities in the extended finite element method. *Computer Methods in Applied Mechanics and Engineering*, 199(49-52):3237–3249, 2010.

[92] Piotr Breitkopf, Alain Rassineux, and Pierre Villon. An introduction to moving least squares meshfree methods. *Revue Européenne des Éléments Finis*, 11(7-8):825–867, 2002.

[93] John C Wagner and Alireza Haghighat. Automated variance reduction of monte carlo shielding calculations using the discrete ordinates adjoint function. *Nuclear Science and Engineering*, 128(2):186–208, 1998.

[94] Brody Bassett and Brian Kiedrowski. Localized radial basis functions for radiation transport in slab geometry. In *Transactions of the American Nuclear Society, Vol. 115*, 2016.