ADAPTIVE MESH REFINEMENT SOLUTION TECHNIQUES FOR THE MULTIGROUP S_N TRANSPORT EQUATION USING A HIGHER-ORDER DISCONTINUOUS FINITE ELEMENT METHOD

A Dissertation

by

YAQI WANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2009

Major Subject: Nuclear Engineering

ADAPTIVE MESH REFINEMENT SOLUTION TECHNIQUES FOR THE MULTIGROUP S_N TRANSPORT EQUATION USING A HIGHER-ORDER DISCONTINUOUS FINITE ELEMENT METHOD

A Dissertation

by

YAQI WANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Jean C. Ragusa
Committee Members,	Marvin L. Adams
	Jim E. Morel
	Guido Kanschat
Head of Department,	Raymond Juzaitis

May 2009

Major Subject: Nuclear Engineering

ABSTRACT

Adaptive Mesh Refinement Solution Techniques
for the Multigroup S_N Transport Equation
Using a Higher-Order Discontinuous Finite Element Method. (May 2009)
Yaqi Wang, B.S., Tsinghua University;
M.S., Texas A&M University
Chair of Advisory Committee: Dr. Jean C. Ragusa

In this dissertation, we develop Adaptive Mesh Refinement (AMR) techniques for the steady-state multigroup S_N neutron transport equation using a higher-order Discontinuous Galerkin Finite Element Method (DGFEM). We propose two error estimations, a projection-based estimator and a jump-based indicator, both of which are shown to reliably drive the spatial discretization error down using h-type AMR. Algorithms to treat the mesh irregularity resulting from the local refinement are implemented in a matrix-free fashion. The DGFEM spatial discretization scheme employed in this research allows the easy use of adapted meshes and can, therefore, follow the physics tightly by generating group-dependent adapted meshes. Indeed, the spatial discretization error is controlled with AMR for the entire multigroup S_N-transport simulation, resulting in group-dependent AMR meshes. The computing efforts, both in memory and CPU-time, are significantly reduced. While the convergence rates obtained using uniform mesh refinement are limited by the singularity index of transport solution (3/2 when the solution is continuous, 1/2 when it is discontinuous), the convergence rates achieved with mesh adaptivity are superior. The accuracy in the AMR solution reaches a level where the solution angular error (or ray effects) are highlighted by the mesh adaptivity process. The superiority of higher-order calculations based on a matrix-free scheme is verified on modern computing

architectures.

A stable symmetric positive definite Diffusion Synthetic Acceleration (DSA) scheme is devised for the DGFEM-discretized transport equation using a variational argument. The Modified Interior Penalty (MIP) diffusion form used to accelerate the S_N transport solves has been obtained directly from the DGFEM variational form of the S_N equations. This MIP form is stable and compatible with AMR meshes. Because this MIP form is based on a DGFEM formulation as well, it avoids the costly continuity requirements of continuous finite elements. It has been used as a preconditioner for both the standard source iteration and the GMRes solution technique employed when solving the transport equation. The variational argument used in devising transport acceleration schemes is a powerful tool for obtaining transport-conforming diffusion schemes.

XUTHUS, a 2-D AMR transport code implementing these findings, has been developed for unstructured triangular meshes.

To my mother, Xiuhua Li

ACKNOWLEDGMENTS

First I would like to express my sincere gratitude to my advisor, Dr. Jean C. Ragusa. It is he who foresaw the importance of this research topic, namely controlling the spatial discretization error with AMR for the multigroup S_N transport equation. I appreciate his advice and help at every stage of this work. His dedication on research has always inspired me. I also express my gratitude to the committee members, Dr. Marvin L. Adams, Dr. Jim E. Morel, and Dr. Guido Kanschat, with whom I had many fruitful research discussions. I am also grateful to Dr. Jean-Luc Guermond, Dr. Richard Sanchez and Dr. Wolfgang Bangerth for their helpful advices.

I spent three summers at Oak Ridge National Laboratory. Thanks to my mentor, Dr. Phillip D. Ferguson, for accepting me into the NESLS program to work with his excellent team at the SNS facility. Thanks also go to the people at NSTD: Dr. Kevin T. Clarno, Dr. Mark D. DeHart, Dr. Thomas M. Evans, and Dr. Jess C. Gehin for their help with the SCALE package and for sharing their experiences regarding transport solvers.

Special thanks to my fellow graduate students: Vijay S. Mahadevan, David E. Ames II, Ayodeji B. Alajo, and Zeyun Wu. I had a lot of fun with them in the past five years.

Most importantly, I am deeply grateful for the support of my family. While my father Xiu Wang had always encouraged me into accomplishing something significant in my life, my mother Xiuhua Li placed more attention on telling me how to take care of myself. Their support should never be underestimated. Thanks to my sisters, Luqi Wang and Yuqi Wang. Their care to my parents in my absence is greatly appreciated. Thanks also to my wife Qun Shi. Her patience and care are the best gift to me.

NOMENCLATURE

Abbreviations

CSEWG	The Cross Section Evaluation Working Group
k_{eff}	Multiplication factor
AMPX	A Nuclear Data Processing System developed at ORNL
AMR	Adaptive Mesh Refinement
ARPACK	Arnoldi Package
C5G7	A 7-group benchmark problem with 5 different configurations
CERFACS	European Center for Research and Advanced Training in Scientific Computation
CFE	Continuous Finite Element
CG	Conjugate Gradient
CMFD	Coarse-Mesh Finite Difference
CPU	Central Processing Unit
DCF	Diffusion Conforming Form
DD	Domain Decomposition
DFE	Discontinuous Finite Element
DGFEM	Discontinuous Galerkin Finite Element Method

DSA	Diffusion Synthetic Accelerations
ENDF	Evaluated Nuclear Data File
FA	Fourier Analysis
FD	Finite Difference
GMRes	Generalized Minimal Residual
IAEA	International Atomic Energy Agency
IP	Interior Penalty
JEFF	The Joint Evaluated Fission and Fusion project
JENDL	Japanese Evaluated Nuclear Data Library
LANL	Los Alamos National Laboratory
LS	Level Symmetric
M4S	Modified 4 Step scheme
METIS	A set of serial programs for partitioning graphs
MFP	Mean Free Path
MIP	Modified Interior Penalty
MOX	Mixed Oxide
MPI	Message Passing Interface
NEWT	New Extended-step-characteristic-based Weighting Transport code (a SCALE module)

NITAWL	Nordheim Integral Treatment And Working Library Production (a SCALE module)
NJOY	A Nuclear Data Processing System developed at LANL
NSR	Numerical Spectral Radius
NSTD	The Nuclear Science and Technology Division of ORNL
ORNL	Oak Ridge National Laboratory
P1C	P_1 Conforming
P1M	Mixed P_1 form
PCG	Preconditioned Conjugate Gradient
pcm	per cent mille 10^{-5}
PD	Positive Definite
PDE	Partial Differential Equation
PHI	Periodic Horizontal Interface problem
PN	Spherical harmonics
SAAF	Self-Adjoint Angular Flux equation
SAF	Significant Angular Fluxes
SCALE	The Standardized Computer Analyses for Licensing Evaluation system
SI	Source Iteration

SN	Discrete ordinates
SPD	Symmetric Positive Definite
SPN	Simplified PN
SQMR	Symmetric Quasi-Minimal Residual method
SR	Spectral Radius
SSOR	Symmetric Successive Over-Relaxation algorithm
TAMU	Texas A&M University
TRIDENT	A 2-D neutron transport code
TRIPLET	A 2-D neutron transport code
TRITON	Transport Rigor Implemented with Time-dependent Operation for Neutronic depletion (a SCALE control module)
UOX	Uranium Oxide
WLA	Wareing-Larsen-Adams DSA scheme
XML	The Extensible Markup Language
XS	Cross Section

Symbols

α	Optional distance between two points
α	Threshold coefficient for refinement
α	Triangle angles

β	Boundary albedo
χ	Neutron fission spectrum
$\mathcal{D}$	Solution domain
δ	δ-function
δ	Difference between two successive iterations
$\ell(\cdot)$	Refinement level of an element
ϵ	A small number for convergence test
ϵ	Error distribution or global error with the numerical "exact" solution
ϵ	Error in angular flux
η	Error distribution or global error given by the jump-based error indicator
$\jmath$	Imaginary unit
$[\![\cdot]\!]$	Jump operator with respect to a streaming direction or a defined $\vec{n}$ on an edge
κ	Penalty coefficient
λ	Wave number in Fourier analysis
$(\cdot)$	Inner product
$\langle\cdot\rangle$	Inner product with streaming factor

$\mathbb{F}$	Collection of edges on subdomain interfaces
$\mathbb{R}$	Set of real number
$\mathbb{T}$	Mesh
$\mathbf{\Phi}$	Solution vector of flux moments or scalar flux
$\mathbf{\Psi}$	Solution vector of angular flux
$\mathbf{\Sigma}$	Global scattering mass matrix
ξ	Reference coordinate vector
ξ_1, ξ_2, ξ_3	Edge mapping for reference triangle
$\mathbf{A}$	Global diffusion matrix
$\mathbf{A}$	Matrix of local system
$\mathbf{B}$	Global diffusion matrix to construct the right-hand-side vector given a source vector
$\mathbf{b}$	Global transport right-hand-side vector
$\mathbf{b}$	Shape function vector
$\mathbf{D}$	Elementary matrix to obtain directional derivative
$\mathbf{D}$	Global direction-to-moment matrix with weightings
$\mathbf{d}$	Global diffusion right-hand-side vector
$\mathbf{E}$	Elementary edge matrix

e	Elementary edge vector whose inner product with an edge solution vector gives the integral of the solution over an edge
G	Elementary streaming matrix
H	Local transport edge matrix
I	Identity matrix
L	Global transport loss matrix
l	Right hand side of local system
M	Elementary mass matrix (used only in Appendix B)
M	Global moment-to-direction matrix
N	Elementary extraction matrix for outward norm derivative on an edge
N	Selecting matrix for the SAF
P	Prolongation from diffusion to transport
Q	External source vector
q	Surface source vector
R	Restriction from transport to diffusion
S	Elementary stiffness matrix
s	Elementary vector whose inner product with a solution vector gives the integral of the solution over the element

$\mathbf{T}$	Transport iteration matrix
$\mathbf{u}$	General symbol for a solution vector
$\mathbf{v}$	General symbol for a solution vector
$\mathbf{v}$	General transport solution vector
$\mathbf{w}$	General transport solution vector
$\mathbf{X}$	Elementary extraction matrix for x-derivative on an edge
$\mathbf{x}$	Coordinate vector
$\mathbf{x}$	General transport solution vector
$\mathbf{Y}$	Elementary extraction matrix for y-derivative on an edge
$\mathcal{E}$	Error in flux moments or scalar flux
$\mathcal{L}$	Maximum source iteration number
$\mathcal{M}$	SSOR preconditioner
D	Block diagonal matrix
E	Type-1 elementary edge matrix
G	Elementary reference mass matrix
L	Block lower-triangular matrix
P	Elementary projection matrix
P	Elementary prolongation matrix
T	Elementary extraction matrix for solution on an edge

tol	Tolerance
μ	Error distribution or global error given by the projection-based error estimator
$\{\!\{\cdot\}\!\}$	Average operator on an edge
$\nabla(\cdot)$	Gradient
$\nabla \cdot (\cdot)$	Divergence
ν	Average number of neutrons emitted per fission
$\overline{\mathbf{H}}$	Local transport upwind matrix
$\overline{\mathbf{L}}$	Upper-triangular part of $\mathbf{L}$ with minus sign
$\overline{\mu}$	Average scattering cosine
∂	Boundary of a domain
Φ	Flux moments or scalar flux
ϕ	Kernel function
ϕ	Scalar flux of the 1-D transport equation
Π	Projection operator
Ψ	Angular flux
ψ	Solution for the simple transport equation or angular flux of the 1-D transport equation
ρ	Spectral radius

σ	Cross section data
τ	A scalar value
$\underline{\cdot}$	Edge-related elementary matrix
$\underline{\mathbf{L}}$	Lower-triangular part of $\mathbf{L}$
$\underline{\mathrm{R}}$	Rotation matrix for an edge solution vector
ε	A small number
$\vec{\Upsilon}$	Higher moment of partial current
$\vec{e}$	Vector from one spatial point to another
$\vec{J}$	Current
$\vec{n}$	Unit norm vector defined for an edge in 2-D or a face in 3-D
$\vec{\Omega}$	Independent angular variable
$\vec{r}$	Independent space variable
$\widehat{\mathbf{b}}$	Reference shape function vector
$\widehat{b}$	Reference shape function
$\widehat{K}$	Reference element
$\widetilde{\cdot}$	A matrix applied with phase shift in Fourier analysis
ξ_1, ξ_2	Reference coordinates
A	Triangle area
$a(\cdot,\cdot)$	Full transport bilinear form

B	Transport albedo operator
$b(\cdot,\cdot)$	Bilinear form
C	Constant
c	Scattering ratio
$c(\cdot)$	Function for edge penalty
D	Diffusion coefficient
d	Dimension
d_{id}	Subdomain or processor ID
E	Energy variable
e	Edge index
E_h^i	Collection of all interior edge
EPS	A small real number for edge orientations
F	Function mapping reference coordinates to real coordinates
f	A general function
$F(\cdot,\cdot)$	Bilinear functional of a pair of primal and adjoint transport solution
G	Number of total energy groups
g	A general function
H	Transport scattering operator
h	Element size

h	Minimum distance for an edge to its opposite vertex in an element
I	Rank-2 identity tensor
J	Jacobin
J	Partial current
$j(\cdot)$	local edge ID of upwind element
K	Element
L	Lobatto function
L	Transport loss operator
L	Triangle edge length
$l(\cdot)$	Linear functional
M	Number of total streaming directions in angular quadrature
m_{id}	Material ID
$N(\cdot)$	Function of local number of unknowns
N_a	Truncation order of the P_N approximation
N_f	Spherical harmonic expansion of the angular flux
N_s	Legendre expansion of the scattering cross section
$N_{\mathbf{SAF}}$	Number of degrees of freedom for the SAF
N_{dof}	Total number of degrees of freedom
N_D	Total number of subdomains or processors

N_{el}	Total number of active elements
N_{mom}	Total number of flux moments
N_M	Total number of materials
N_R	Total number of regions
N_S	Total number of sources
np	Number of processors or subdomains
P	Polynomial function space for a reference simplex
P	Transport fission operator
p	Polynomial order
Q	External source moment
Q	Polynomial function space for a reference hypercube
q	Convergence rate
r	Solution regularity index
r	Surface-volume ratio
$R(\cdot)$	Linear functional of a transport solution
r_{id}	Region ID
S	Angular external source
s	A scalar value
s	Edge integral variable

S^2	2-D unit sphere
s_{id}	Source ID
t	Triangle orientation
u	+1 or -1
u	General symbol for a solution
V	Polynomial function space
v	General symbol for a solution
W	Broken function space
w	Weight in angular quadrature
X, Y	Domain sizes in x- and y-direction
x, y	Coordinates
Y	Spherical harmonics

Superscripts

$+$	Downwind side with respect to a streaming direction or a defined $\vec{n}$
$-$	Upwind side with respect to a streaming direction or a defined $\vec{n}$
$*$	Adjoint or test function
ℓ	Source iteration index
C	Edge coupling
c	Robin or Cauchy boundary

cvg	Converged
d	Dirichlet boundary or vacuum boundary
g	Energy group index
inc	Incoming
IP	Interior Penalty
k	Cycle number
MIP	Modified Interior Penalty
n	Numerical solution
r	Reflecting boundary
ref	Reference solution

Subscripts

$-$	Reverse streaming direction
SAF	Significant angular flux
source	Source iteration
$\perp$	Perpendicular
$\vec{\mathbf{n}}$	Outward unit norm vector for an element
a	Absorption
AMR	Adaptive Mesh Refinement
b	Boundary

DCF	Diffusion Conforming Form
DSA	Diffusion Synthetic Acceleration
e	Edge index
e	Even parity
$exact$	Exact solution
f	Fission
$flux$	Flux of outer iteration
g	Energy group index
h	Discretization
h	Mesh at a refinement cycle
$h/2$	Global h-refined mesh at a refinement cycle
$inner$	Inner iteration
IP	Interior Penalty
j	Shape function or local edge index
K	Element
k	Shape function index
k	Spherical harmonics index
m	Streaming direction index
MIP	Modified Interior Penalty

n	Spherical harmonics index
n	Unit norm direction $\vec{n}$
o	Odd parity
$P1C$	P_1 Conforming
$P1M$	Mixed P_1 form
r	Reflecting
s	Scattering
t	Total collision
$thermal$	Thermal iteration
tr	Transport
k_{eff}	k-effective

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

A. Purpose

The use of Adaptive Mesh Refinement (AMR) techniques [1, 2, 3, 4, 5, 6] has become widespread in many science & engineering disciplines over the last decade or so. In this dissertation, we present advances in AMR for the steady-state multi-group discrete ordinates (S_N) transport equation with applications to neutron transport. This study is an effort to improve numerical simulations related to the linear Boltzmann, the integro-differential transport equation governing the phase-space distribution of neutral and charged particles interacting with a background medium. Solving the transport equation is of high importance and interest in many fields, and, could have numerous applications in science & engineering applications, such as nuclear reactor analysis [7], computed tomography [8], weather forecast [9], etc. It is important to note that solving for the particle distribution requires a discretization of the phase-space, which consists of the physical space and velocity space, or equivalently physical space, direction space and energy space. The objective of this dissertation is to apply *spatial* AMR techniques to the transport equation, using higher-order approximations on unstructured grids. In the following paragraphs, we illustrate the needs for this work and present the various topics discussed in the following chapters.

With AMR, not only can the discretization error be controlled, but the computational effort can also be reduced significantly. The basic idea of AMR is illustrated in Fig. I-1. The AMR procedure starts with a given initial mesh, which is usually

---

The journal model is *Nuclear Science and Engineering*.

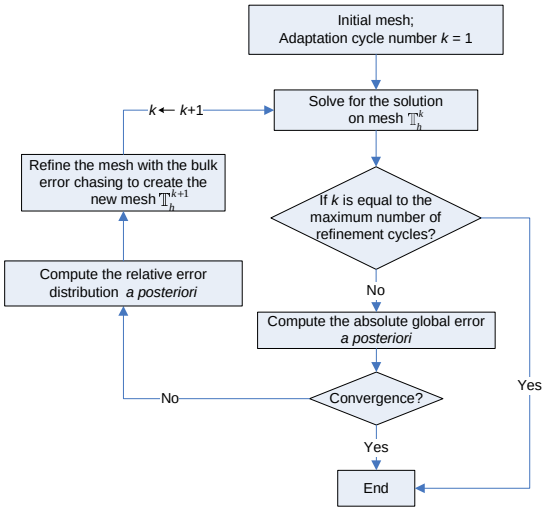


Fig. I-1. Principle of mesh adaptation.

coarse and can be determined simply by using material regions for instance. Generating the initial mesh does not require any knowledge about the physics, although it may be preferable to use a more sophisticated tool such as a mesh generator to create a initial mesh with better topological regularity. Then, a numerical solution is pursued with any adequate transport solver on this initial mesh. Once a numerical solution is available, the relative error distribution can be estimated in an *a posteriori* fashion, i.e., using the current numerical solution to obtain/generate information related to the spatial distribution of the numerical error. This technique is known as *a posteriori* error estimation. If the global error can be estimated accurately, the procedure can be terminated when the global error is smaller than a user-prescribed tolerance. Otherwise, the AMR procedure is usually terminated when the prescribed number of AMR iterations has been reached. Assessing the global error accurately may not be as important as obtaining a spatial distribution in terms of the relative error in order to reduce the number of spatial unknowns as much as possible. Once

the error distribution has been determined, a simple bulk-error-chasing strategy is performed, in which a fraction of cells whose errors are relatively large (i.e., above a certain threshold) is refined. This refined mesh is set as the new current mesh and the next cycle of mesh refinement proceeds until the prescribed tolerance for the discretization error is satisfied or the maximum number of cycles has been reached. In summary, AMR wraps mesh iterations around the traditional transport solver to drive the mesh adaptation.

The success of AMR depends on the following three facts:

1. The number of unknowns associated with the adapted mesh is orders of magnitude smaller than the one of a uniform mesh because

    - The smoothness of the solution is not uniform and uniform refinement may be *overkill* and
    - Engineering practice may not need a very accurate solution throughout the entire domain but only the precise knowledge of some quantities of interest, i.e., functionals of the solution over a small portion of the phase-space (this adaptivity technique is usually referred to as goal-oriented AMR) [10, 11].

2. The simple bulk-error-chasing strategy can deliver adapted meshes fairly close to the "optimal" mesh using a hierarchy of nested meshes referenced by their *levels of refinement* with respect to the initial mesh.

3. The relative error distribution of a mesh can be obtained with *a posteriori* techniques employing the current numerical solution on the mesh.

Because the number of unknowns associated with an adapted mesh is orders of magnitude smaller than the number of unknowns for a uniform mesh yielding about the

same required accuracy, the total computational time of the whole AMR procedure is typically smaller than the time needed for one calculation on a uniform fine mesh. The memory usage is reduced significantly as well.

We expect to control the *spatial* discretization error for the steady-state multigroup S_N neutron transport equations by devising a practical AMR technique for it. We believe once this is accomplished, it will shed some light on how to further apply AMR with respect to other phase-space variables, i.e., angular, energy and time variables. The computational effort using AMR is also expected to be reduced significantly so that analyzing large, realistic, multi-dimensional problems (simulations once impossible with uniform meshes) can be actively pursued.

B. Brief Background on the S_N Multigroup Neutron Transport

The present work deals with stationary S_N multigroup neutron transport because this contains the essence of all particle transport phenomena: the integro-differential nature of the PDE. Dealing with time variable and other kinds of particles [12] [13] [14] are important and challenging aspects as well, although they are not the focus of this research.

1. The Multigroup Procedure for the Energy Variable

The multigroup approximation will be employed to discretize the energy variable, mostly because it is the only widely used scheme in deterministic transport methods. With the multigroup approximation, one first breaks up the entire energy range into intervals or energy groups. The neutron transport equation is then integrated over each energy group in order to define appropriate average values of the various cross sections characterizing each group. After an assumed intra-group spectrum is used to compute these multigroup cross sections, a set of coupled multigroup equations are

formed. This procedure is explained in Appendix C.

Researchers have spent tremendous effort providing evaluated nuclear data libraries and updating them regularly. Currently, there are three major neutron data libraries: ENDF/B (Evaluated Nuclear Data File) [15], JENDL (Japanese Evaluated Nuclear Data Library) [16], and JEFF (the Joint Evaluated Fission and Fusion project) [17]. All energy-dependent data are stored in the internationally adopted format (ENDF-6) maintained by CSEWG (the Cross Section Evaluation Working Group) [18]. Covariance data are gradually put into the libraries to indicate the uncertainty of the data. Benchmark integral effect and separate effect experiments have been performed to validate the libraries and the transport codes employing them [19].

Existing codes such as NJOY [20] and AMPX [21] in SCALE [22] are used to process the continuous cross section data from the nuclear data library into the multigroup cross sections. Physical phenomena are properly accounted for, such as elastic and inelastic scattering, resonance self-shielding, Doppler effect to account for the thermal motion of the background nuclei, thermal equilibrium, etc.

There are data uncertainties associated with point-wise cross sections due to the experimental uncertainties and nuclear model uncertainties. In addition, the multigroup cross section process itself introduces additional uncertainties due to the presumed intra-group energy spectrum. How to evaluate these uncertainties is still an open problem [23] which will not be addressed in this dissertation. In this study, multigroup cross sections are assumed to be accurate. The propagation of the data uncertainties on the solution accuracy [24], which should eventually be used to control the termination of the AMR process, is not considered here.

2. Discrete Ordinates (S_N) Process for Angular Discretization

We will use the discrete ordinates method (S_N) for the angular discretization of the transport equation. In the S_N method, an angular quadrature set is chosen to evaluate all angular integrals and to solve the transport equation along the predetermined quadrature directions. The S_N method is a collocation discretization scheme, unlike a modal expansion scheme such as the P_N [25] or the Simplified P_N (SP_N) [26] methods. When using the S_N technique, the transport equation can

1. either be solved along a set of given trajectories (where a trajectory is a line determined by an entry point on the inflow boundary and a given direction of the angular quadrature set) throughout the entire computational domain, leading to the method of Characteristics or Long Characteristics [27],

2. or be solved from a mesh cell located on the inflow boundary, with radiation being followed inwards across all downwind (downstream) mesh cells. In this cell-by-cell approach, which we have adopted, the "short" Characteristics method and the Discontinuous Galerkin Finite Element Method (DGFEM) [28, 29, 30, 31, 32] are two widely used spatial discretization schemes.

In the cell-by-cell approach, an efficient procedure called "the transport sweep" is used to invert the streaming+total collision operator of the transport equation in a matrix-free fashion. Situations where solving one task (i.e., solving for radiation in a given cell for a given direction) can only be done after another task has been completed and vice versa, may exist. These situations are called cycles; cycles can be broken with the introduction of the significant angular flux variables and do not affect the matrix-free sweep. The details about the significant angular fluxes will be discussed in Chapter II. S_N methods have a long history that traces back to the 1950's [33] and are still widely in use nowadays. Nonetheless, S_N methods suffer

from ray effects [34], which could be resolved in the future using angular adaptivity. Currently, the ray effects are mitigated using first-collision source approaches [35, 36]. Since angular discretization is not the focus of this study, we refer the reader to the references cited later for additional details.

C. Approach for Applying AMR to the S_N Multigroup Transport Equation and Brief Literature Review

We have developed a methodology to perform AMR for the multigroup S_N transport solver using a higher-order DGFEM on *hp*-type unstructured meshes. In the following paragraphs, we briefly present four key aspects of this research and put them in context; these are:

1. higher-order DGFEMs: their current usage in the neutron transport community;
2. AMR on *hp*-type unstructured meshes with transport sweeps;
3. the *a posteriori* error estimators available for the transport equation;
4. the need to develop a methodology capable of performing Diffusion Synthetic Accelerations (DSA), especially on AMR unstructured meshes.

Some considerations regarding code development will also be provided. The following chapter will provide details on these aspects. Because the main subject in this research focuses on method development, the higher level of complexity in 3-D geometries will not be considered. We will deal with 2-D geometries in our implementation. The methodology and lessons on implementation learned in this work can be applied in 3-D.

1. Higher-Order Discontinuous Galerkin Finite Element Methods

The DGFEM is one of many various types of spatial discretization schemes for the first-order transport equation. Other spatial schemes for different forms (first-order, second-order, and integral) of the transport equation, including finite difference/volume methods, collision probability methods, long characteristic and short characteristic methods [37, 25, 31, 27, 33, 38, 39], will not be considered here. DGFEM is widely used to spatially discretize the discrete ordinates S_N transport equation. The method was originally derived for neutron/photon transport problems in the early 1970's [28, 29]. For instance, the computer code TRIPLET of Reed and Hill [40] used a DGFEM for 2-D regular triangular meshes with various polynomial order approximations. A few years later, the TRIDENT code for 2D multigroup triangular mesh S_N transport was released [41, 42, 43], but only employed linear basis functions. Similarly, most of the recent work related to the DGFEM discretization of the transport equation on unstructured meshes has been carried out using linear DGFEMs [31, 32, 44]. Even though the original work of Reed and Hill was not restricted to order-1 spatial representations, we note that there has not been much research carried out for higher-order finite element transport so far. Nonetheless, in other disciplines, e.g., fluid dynamics, where hyperbolic conservation laws are also present the development of higher-order DGFEM has significantly progressed, and it is important to bring and test these improvements for applications within the nuclear science and engineering community.

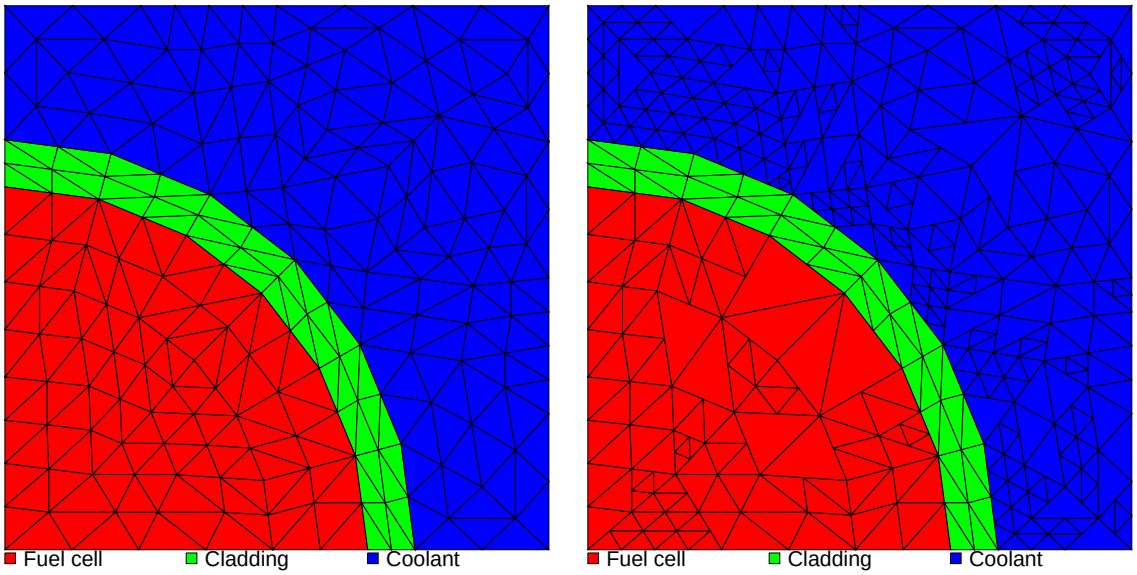
2. AMR on Unstructured Meshes

Unstructured meshes are widely used, allowing for the discretization of complicated geometries. The main feature of unstructured meshes is that the locations and con-

nections of all of the cells are described explicitly so that cells can be distributed freely. Unstructured meshes are naturally employed by *hp*-type AMR, where a local element can be refined either through cell sub-divisions (h-refinement) or by increasing the polynomial order of the approximation (p-refinement). h-refinement is appropriate for solutions with singularities, while p-refinement is good for smooth solutions. Their combination, the *hp*-type AMR, allows for the mesh to follow the physics tightly, reducing the number of unknowns significantly. *hp*-type meshes are easily supported in DGFEM, even for multi-dimensional geometries, thanks to the discontinuous nature of the numerical approximation. In DGFEM, continuity of the numerical solution is not required across elements, allowing for different levels of refinement in two adjacent elements. Likewise, the polynomial approximation across two elements can be arbitrary using a DGFEM. Continuous finite element approaches would require that the numerical solution be continuous, which adds a significant level of difficulty for *hp*-AMR meshes. Fig. I-2 shows an example of an unstructured mesh for a single fuel cell (rod+clad+moderator).

We will use triangular *hp*-type unstructured meshes in this study.

hp-type AMR emerged in the late 1980's [45, 46] and required the resolution of several formidable problems for an effective implementation: new data structures, efficient linear solvers, effective local error estimators. Applying *hp*-type AMR for hyperbolic equations, e.g., in Computational Fluid Dynamics, is an ongoing very active research field [47, 48, 49, 50]. In recent years, h-type and p-type AMR have been investigated for transport or diffusion calculations [51, 52, 53, 54, 55]. *hp*-type refinement can deliver exponential convergence for elliptic problems while h-type refinement only has algebraic convergence for the multigroup diffusion problem [56]. Investigating the performance of *hp*-type mesh refinement for the multigroup S_N equation is a very new and interesting topic.



(a) Regular mesh (without hanging nodes)

(b) Irregular mesh (with hanging nodes)

Fig. I-2. Example of unstructured meshes.

Finally, subdividing an element into smaller elements requires the ability to modify the element-by-element solution procedure (transport sweeps) to account for the presence of newly created cells in the domain. This is a straightforward issue, discussed in Chapter IV.

3. *A posteriori* Error Estimates

The *a posteriori* error estimation drives the AMR procedure by providing a measure of the local error based on the current numerical solution, i.e., *a posteriori* . The effectiveness of the *a posteriori* error estimates, i.e., the accuracy of the calculated error distribution and global error in some norm, is the key for the control of the discretization error. The development of *a posteriori* error estimators was reviewed in [57, 58], mostly for elliptic equations. Error estimators for the transport equation, and more generally for hyperbolic equations, are still under development. Currently, the state-of-the-art in the mathematical sciences regarding *a posteriori* error estima-

tors for hyperbolic equations rely on the use of an adjoint solution used as a weighting function in the derivation of the local error quantities [59, 60]. In S_N transport problems, this would require solving the problem in the reverse directions and retaining the direct and adjoint angular fluxes, which can be very costly. Instead, researchers have had recourses to semi-heuristical error indicators to drive the AMR procedure for transport problems. In [51], the gradient of the solution is employed to drive AMR for the radiation transport component in 2-D Cartesian geometries for one-group (one-frequency) equations; this estimator is known to be fairly accurate for low-order (e.g., first-order step) schemes but is overly conservative for higher-order schemes. A similar multiple-grid technique, with error estimation based on the gradient of the numerical solution, is also used by several other authors for photon transport applications; see, for instance [61, 62, 63]. In [64], a local refinement (cell-based AMR) technique is described for S_N transport, where the value of the neutron MFP (Mean-Free-Path) in a given cell is employed as a mesh refinement criterion. While this approach takes into account the size of potential internal layers at a given location in the domain, it does not account for the *actual* smoothness of the solution at these locations and is, therefore, far from optimal; for example, in optically thick media, the solution may well be approximated by a smooth spatial representation on coarse meshes despite the large optical size of the mesh. Hartmann and Leicht have monitored the inter-element jumps in the DGFEM solution [65, 66] since it was observed that the magnitude of these jumps diminished with refinement.

4. Diffusion Synthetic Acceleration

Diffusion Synthetic Acceleration, or DSA, is a very important part of any transport solver when dealing with highly diffusive media. For such media, the traditional Richardon iteration or Source Iteration (SI) is ineffective and preconditioning needs

to be performed in order to run simulations in reasonable CPU time. This preconditioner step can be of various natures, as described in the review article by Adams and Larsen [67], but the most popular choice consists in employing a diffusion solver to accelerate the transport solver in highly diffusive problems. Unfortunately, the discretized diffusion equations for the preconditioning step cannot be derived independently of the discretized transport equation for the preconditioning to be effective or even stable. For unstructured mesh DGFEM transport solvers in multi-dimensions, obtaining an effective and stable DSA scheme is an open and arduous problem. Although the modified-four-step (M4S) scheme [68][69] works well in 1-D, it suffers instability in 2-D unstructured meshes unless the cell size is either very large or small in terms of MFP. On the other hand, the "fully consistent" DSA scheme [70] is stable but not computationally effective due to the fact that a complicated mixed hybrid DGFEM for the P_1 equation needs to be solved at each Source Iteration. Two essential questions regarding DSA still persist:

- Does an unconditionally stable and effective DSA scheme exist in the case of unstructured meshes for which cell sizes may greatly vary, in the case of highly anisotropic scattering, and in the case of high-aspect ratio (sliver) cells?
- Can the resulting diffusion equation be solved efficiently, i.e., is the computing time spent in DSA comparable with the time spent in transport sweeps?

In this work, we have derived DSA equations which satisfy the vast majority of these needs.

5. Method and Code Development Aspects

In this research, XUTHUS, a reusable production code (as opposed to a mock up code), has been written to solve the S_N multigroup transport equation with spatial AMR. A

significant effort has been placed on both the quality of the code and on the methods themselves. The advantages of such a decision include: full control on the code implementation, which is currently almost impossible by using third-party libraries, and the personal satisfaction gained by putting a piece of software into production. Additionally, the developed code could be re-used for research activities in the future, including adaptivity for coupled electron-photon transport, for example. Therefore, in collaboration with the Nuclear Science and Technology Division (NSTD) of Oak Ridge National Laboratory (ORNL), this new transport solver has been planned to be put into a developmental version of SCALE [22], with the ultimate goal of releasing it as a new module within the TRITON lattice physics sequence [71], thus providing users an alternative to NEWT [72], the 2-D S_N module currently used within TRITON. Our initial objectives in the development of this new transport solver as a module in SCALE are:

- to implement arbitrary higher-order spatial shape functions in the DGFEM framework, where, as in the Extended Step Characteristic solver of NEWT, the scattering and fission source is assumed spatially constant in each polygonal element,
- to gain additional flexibility and robustness by having two distinct solvers,
- to control the spatial discretization error with a user-prescribed tolerance with AMR, and
- to extend SCALE on potential applications beyond traditional nuclear reactor fuel assembly calculations, including shielding or inverse problems.

Parallel domain decomposition has also been implemented in XUTHUS, using MPI (Message Passing Interface) to take the advantage of the development of supercom-

puters and to handle extremely large problems.

D. Organization of the Dissertation

The Chapters of this dissertation are organized as follows.

In Chapter II, we present the higher-order DGFEM for the multigroup S_N equation, investigate its convergence properties, and draw conclusions as to whether there is a need for higher-order methods in radiation transport using S_N solvers. The motivation behind using higher-order schemes for S_N transport calculations will be demonstrated with some sample problems.

In Chapter III we present our new DSA schemes obtained rigorously from the DGFEM S_N transport variational form. Both the stability and effectiveness issues of these DSA schemes are studied.

The h-type AMR for transport is presented in Chapter IV. Two different *a posteriori* error estimations have been devised for the multigroup S_N transport equation with DGFEM: a projection-based error estimator and a jump-based error indicator. The mesh coupling algorithm to deal with mesh irregularity (different levels of refinement in a computational domain) is presented. The issue of adaptivity in the context of multigroup equations is also presented and our technique to resolve this is explained. Convergence studies of h-type AMR are performed to demonstrate the validity of the approach. Finally, highly diffusive cases are presented to test our DSA schemes on AMR meshes.

Chapter V provides some implementation details regarding our new 2-D transport solver XUTHUS, such as its development history, data structure, matrix-free scheme, MPI parallelism with the domain decomposition, integration process into SCALE, etc.

Finally, we draw some conclusions and give recommendations for future devel-

opments in Chapter VI.

The detailed literature reviews, numerical results and conclusions related to each specific topic are distributed among Chapters II, III and IV.

Additional material, such as different forms of the transport equation, the derivation of the multigroup S_N transport equation and its iterative solver, the DGFEM for the transport equation in purely absorbing media, the Mathematica notebook file used to obtain all the local matrices, etc., are appended for completeness in Appendices A through F.

CHAPTER II

HIGHER-ORDER DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD FOR THE MULTIGROUP S_N EQUATIONS

A. Review of Higher-Order DGFEM and Application to the Multigroup S_N Equations

The Discontinuous Galerkin Finite Element Method (DGFEM) has been widely used to discretize in space hyperbolic partial differential equations (e.g., conservation laws) due to the relative simplicity of the scheme and its similarity with the Finite Volume Method; indeed, the lowest order of DGFEM is a Finite Volume method. In the transport community, predominantly linear DGFEM (i.e., DGFEM with linear shape functions) has been employed to solve the discrete ordinates S_N transport equation. The DGFEM was originally derived for neutron/photon transport problems in the early 1970's [28, 29]. For instance, in the computer code TRIPLET of Reed and Hill [40], the method was presented for 2-D regular triangular meshes with various polynomial order approximations. A few years later, the TRIDENT code for 2-D multigroup triangular mesh S_N transport was released [41, 42, 43]; TRIDENT was only based on a linear DGFEM method, which we denote hereafter by DGFEM(1). Similarly, most of the recent work related to the DGFEM discretization of the transport equation on unstructured meshes has been carried out using DGFEM(1). For instance, a linear function representation has been used for 3-D unstructured tetrahedral meshes in [31] and [32] and a trilinear representation for 3-D hexahedral meshes in [31]. These works have led to the creation of the DGFEM(1)-based Attila code [44], developed by Los Alamos National Laboratory and later Transpire company. Even though the original work of Reed and Hill was not restricted to order-1 spatial representations, we note

that there has not been much research carried out for higher-order finite element solution of the neutron transport equation so far. Several reasons can be put forth to explain this:

1. In the TRIPLET code manual, Reed and Hill recommended the use of "linear finite elements for problems where the computational time was a significant consideration"; furthermore, their higher-order basis functions used Lagrange polynomials based on equally-spaced points. Instead, we use a hierarchical basis, which allows for simpler coupling between AMR mesh cells, see Section B on higher-order hierarchical basis functions in this chapter and Section 5 in Chapter IV for further details on cell coupling with AMR;

2. Subsequently, the TRIDENT code was developed, employing only linear finite elements; the more recent works by McGhee, Wareing, Morel, and Warsa inherited the legacy of these seminal works with respect to their choice of linear spatial representation, focusing on the accuracy in the thick diffusive limit and on the robustness of the spatial discretization.

Nonetheless, in other disciplines, e.g., fluid dynamics, where hyperbolic conservation laws are also present, the development of higher-order DGFEM has significantly progressed. Higher-order basis functions for triangular meshes based on Jacobi polynomials have been derived and accurate solutions using higher-order functions can be obtained in fewer mesh cells (see [73, 74, 75, 4, 5, 4]). It is important to bring and test these improvements for applications within the nuclear science and engineering community.

The case for higher-order methods can be made based on (i) the higher enhanced convergence rates (i.e., better accuracy in the solutions) and (ii) on the fact that the solution time for higher-order methods may virtually come for free on modern com-

puter architectures. Today's computational bottleneck for neutron transport is predominantly memory-access related, and low-order spatial discretizations may require more elements in a computational domain, i.e., more solves of a small linear system, than higher-order approximations. We recall here that in S_N transport sweeps, one needs only to solve a small linear system, obtained by sweeping the mesh element by element. For linear finite elements on triangles, this system is of size 3×3, potentially leaving processors starved for data to compute, whereas higher-order approximations yield larger elementary systems, of size $N \times N$, with $N(p) = (p+1)(p+2)/2$ and where p is the polynomial order, thus providing more data to the CPU per solve. Therefore, a logical alternative to circumvent a memory-access constrained environment is to provide the CPU more data to compute at a single time by employing higher-order spatial representations.

Furthermore, another significant reason to consider higher polynomials is related to the fact that a DGFEM of order p, hereafter denoted by DGFEM(p), can yield convergence rates of order $q = \min(p + 1, r)$ for hyperbolic equations, where r is the solution regularity. Lesaint and Raviart, in 1974 [29], demonstrated that the convergence rate was p; later, Richter improved the theoretical result for topologically regular meshes to $p+1$ [76]. That is, the error term is of the form Ch^q, with C being a proportionality constant and h the typical mesh size.

The convergence of various spatial discretizations of the discrete ordinate transport equations has been the focus of many papers for over four decades. Earlier work mostly dealt with Finite Difference (FD) approximations (diamond differences, weighted diamond differences) (see, for instance, [77, 78, 79, 80, 81]). In [77, 78], Madsen established the stability and second-order convergence of a class of finite differences approximations (the diamond difference method of Carlson and weighted central differences) in a L_2-discrete norm. Madsen's proof required that the exact

solution possessed bounded third partial derivatives [77]. As noted by Arkuszewski [82] and Larsen [83], such conditions are seldom verified in realistic problems, where singular characteristics are present. This triggered several numerical studies of the convergence of FD schemes, typically in simple 1-D or 2-D configurations; see, for instance, [84, 83]. Larsen provided convergences rates of the numerical solutions in L_1, L_2, and L_∞ norms for various mesh refinements; in the 1-D case [84], these rates, measured in any norm, showed second-order convergence, whereas in 2-D, the rates depended on the norm utilized and were fractional powers of the mesh size. More recently, Azmy [85] numerically investigated the convergence rates of weighted Diamond Differences, nodal, and characteristics methods, using a variation of a test case proposed earlier by Larsen. The nodal and characteristics methods tested included constant, linear, and quadratic spatial approximations.

In this research, we implemented a DGFEM(p) for polynomial orders up to 4, tested it on several benchmark problems, and report the findings in terms of accuracy (convergence) as a function of both the total number of unknowns and the CPU time. Our choice of higher-order basis function consists of hierarchical functions. Such a set of functions is obtained by nesting the basis functions set, such that all lower-degree bases are included in the higher-degree bases. Such sets have attractive features, e.g., better conditioning properties [86] and ease of implementation, including nested elementary matrices and simple procedures to obtain edge values for inter-element communications. For instance, hierarchical bases have been used by [87] and [56] for multigroup diffusion applications. We studied the convergence properties of DGFEM applied to neutron transport for various orders of spatial approximation (from $p = 1$ to $p = 4$), and we compare the observed numerical rates with theoretical results.

The outline of this chapter is as follows. In Section B, we present the higher-order DGFEM for the multigroup S_N transport equation. We first give the variational

form with the upwind scheme, the definition of higher-order shape functions and the theoretical convergence rate for transport calculations. In Section C, we present the matrix-free transport sweep and the iterative solver based on it with both the Source Iteration (SI) and GMRes. Details on dealing with the sweeping cycles are explained with the introduction of the so-called Significant Angular Fluxes (SAF). In Section D, we present the results of higher-order transport calculations. Both the convergence rate with the uniform refinement and the grind time in the matrix-free transport sweep are presented. Finally, conclusions are provided in Section E.

B. DGFEM for the Multi-Group S_N Transport Equation

1. Multigroup S_N Transport Equations

The multigroup S_N transport equation is presented in detail in Appendix C. For the completeness, it is reproduced here. Given an angular quadrature set $\left\{\vec{\Omega}_m, w_m\right\}_{m=1,\ldots M}$ consisting of M directions $\vec{\Omega}_m$ and weights w_m, and a total number of G energy groups, the steady-state multigroup S_N transport equation in *one* direction indexed by m, for *one* group g, written for an open convex spatial domain $\mathcal{D}$ with boundary $\partial\mathcal{D}$ is,

$$\begin{aligned}\left(\vec{\Omega}_m \cdot \vec{\nabla} + \sigma_{t,g}\right)\Psi_{m,g}(\vec{r}) &= \sum_{g'=1}^{G}\sum_{n=0}^{N_a}\frac{2n+1}{4\pi}\sigma_{s,n}^{g'\to g}(\vec{r})\sum_{k=-n}^{n}\Phi_{n,k}^{g'}(\vec{r})Y_{n,k}(\vec{\Omega}_m) \\ &\quad + \frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\nu\sigma_{f,g'}(\vec{r})\Phi_{g'}(\vec{r}) + S_{m,g}^{ext}(\vec{r}) \qquad \text{for } \vec{r}\in\mathcal{D}\end{aligned} \tag{2.1}$$

with the general boundary condition

$$\begin{aligned}\Psi_{m,g}(\vec{r}_b) &= \Psi_{m,g}^{inc}(\vec{r}_b) + \sum_{g'=1}^{G}\sum_{\vec{\Omega}_{m'}\cdot\vec{n}_b>0}\beta_{m'\to m}^{g'\to g}(\vec{r}_b)\Psi_{m',g'}(\vec{r}_b) \\ &\qquad \text{for } \vec{r}_b\in\partial\mathcal{D}_m^- = \left\{\partial\mathcal{D},\ \vec{\Omega}_m\cdot\vec{n}_b<0\right\}\end{aligned} \tag{2.2}$$

Symbols used in the above equation are standard in textbooks and journals; their meanings are listed below for completeness:

$\vec{r}$	position variable [cm]
$\mathcal{D}$	$\in \mathbb{R}^d$ open convex spatial domain of dimension d
$\partial\mathcal{D}$	boundary of spatial domain $\mathcal{D}$
$\vec{n}_b$	$= \vec{n}(\vec{r}_b)$ outward unit normal vector on boundary $\partial\mathcal{D}$
$\vec{\Omega}_m$	unit vector for a steaming direction m in the angular quadrature set
m	index of streaming directions, from 1 to M
g	index of energy groups, from 1 to G

Angular fluxes and flux moments variables are given by:

$\Psi_{m,g}(\vec{r})$	$= \Psi_g(\vec{r}, \vec{\Omega}_m)$ neutron angular flux $[\frac{n}{cm^2 \cdot ster \cdot s}]$
$\Phi_g(\vec{r})$	$= \Phi^g_{0,0}(\vec{r}) = \sum_{m=1}^{M} w_m \Psi_{m,g}$ neutron scalar flux $[\frac{n}{cm^2 \cdot s}]$
$\Phi^g_{n,k}(\vec{r})$	$= \sum_{m=1}^{M} w_m Y_{n,k}(\vec{\Omega}_m)\Psi_{m,g}$ neutron flux moments $[\frac{n}{cm^2 \cdot s}]$
$Y_{n,k}(\vec{\Omega})$	spherical harmonics functions defined with Eq. (C.29) and Eq. (C.39)

The other terms are:

$S^{ext}_{m,g}(\vec{r})$ $= S^{ext}_{g}(\vec{r},\vec{\Omega}_m)$ external source $[\frac{n}{cm^2\cdot ster\cdot s}]$

$\sigma_{t,g}(\vec{r})$ macroscopic total cross section$[cm^{-1}]$

$\sigma^{g'\to g}_{s,n}(\vec{r})$ $= \int_{-1}^{1} d\mu\, \sigma_s^{g'\to g}(\vec{r},\mu)P_n(\mu)$ macroscopic scattering cross section $[cm^{-1}]$

N_a truncation order of the P_N approximation, see Section A of Appendix C for more details

$\nu\sigma_{f,g}(\vec{r})$ fission cross section times the average number of neutrons emitted per fission $[cm^{-1}]$

$\chi_g(\vec{r})$ neutron fission spectrum

$\Psi^{inc}_{m,g}(\vec{r}_b)$ non-homogeneous incoming angular flux on boundary $[\frac{n}{cm^2\cdot ster\cdot s}]$

$\beta^{g'\to g}_{m'\to m}(\vec{r}_b)$ boundary albedo, its definition depends on the quadrature set, also refer to the Eq. (C.28).

The multigroup equation is solved with the iterative solver presented in Section B of Appendix C, in which a sequence of one-group problems are solved. In addition, in the case of an eigenvalue problem, an additional (outer) iteration loop is added to update the eigenvalue estimate after each multigroup solve. For brevity's sake, it is suitable to solely present the spatial discretization and AMR technique method in the one-group case. The one-group S_N equation is,

$$\left(\vec{\Omega}_m\cdot\vec{\nabla}+\sigma_t(\vec{r})\right)\Psi_m(\vec{r}) \;=\; \sum_{n=0}^{N}\frac{2n+1}{4\pi}\sum_{k=-n}^{n} Y_{n,k}(\vec{\Omega}_m)\left[\sigma_{s,n}(\vec{r})\Phi_{n,k}(\vec{r})+Q_{n,k}(\vec{r})\right] \quad \text{for } \vec{r}\in\mathcal{D} \tag{2.3}$$

where the source terms (inscattering source, external source, fission source) have been lumped into a single source term $Q(\vec{r},\vec{\Omega})$ which has already been expanded on a spherical harmonics basis to yields the source moments $Q_{n,k}(\vec{r})$ in Eq. (2.3).

The general boundary condition for the one-group case is:

$$\begin{aligned} \Psi_m(\vec{r}_b) &= \Psi_m^{inc}(\vec{r}_b) + \sum_{\vec{\Omega}_{m'}\cdot\vec{n}_b>0} \beta_{m'\to m}(\vec{r}_b)\Psi_{m'}(\vec{r}_b) \\ &\text{for } \vec{r}_b \in \partial\mathcal{D}_m^- = \left\{\partial\mathcal{D}, \vec{\Omega}_m \cdot \vec{n}_b < 0\right\} \end{aligned} \tag{2.4}$$

If there are only Dirichlet-type and reflecting boundaries, the boundary condition can be written as

$$\begin{aligned} \Psi_m(\vec{r}_b) &= \begin{cases} \Psi_m^{inc}(\vec{r}_b), & \vec{r}_b \in \partial\mathcal{D}^d \\ \Psi_{m'}(\vec{r}_b), & \vec{r}_b \in \partial\mathcal{D}^r \end{cases} \\ &\text{for } \vec{r}_b \in \partial\mathcal{D}_m^- \end{aligned} \tag{2.5}$$

where $\partial\mathcal{D} = \partial\mathcal{D}^d \cup \partial\mathcal{D}^r$, and the reflecting direction is given by

$$\vec{\Omega}_{m'} = \vec{\Omega}_m - 2(\vec{\Omega}_m \cdot \vec{n}_b)\vec{n}_b \tag{2.6}$$

We suppose the angular quadrature set is designed to satisfy the following two conditions regarding any outward unit normal vector on reflecting boundaries $\partial\mathcal{D}^r$:

1. $\forall m = 1, \cdots, M$, the reflected direction $\vec{\Omega}_{m'}$ is also in the quadrature set for any location on $\partial\mathcal{D}^r$.

2. the weights of the incident and reflected directions must be equal, i.e., $w_m = w_{m'},\ m = 1, \cdots, M$

This is usually not an issue when reflecting boundaries are on the x, y, z axes. Nevertheless, even the reflecting boundaries are not perpendicular or parallel to these axes, for example, in the case of a 2-D hexagon fuel assembly with reflecting boundaries on all six sides, a proper angular quadrature set can still be chosen to meet the above two conditions (e.g., product quadrature set for hexagonal fuel lattices).

2. Local Weighted-Residual Formula and DGFEM

We consider the DGFEM for one angular direction $\vec{\Omega}_m$ and one spatial element K of an unstructured mesh $\mathbb{T}_h$, such that the union of the all elements fully covers $\mathcal{D}$, i.e., $\bigcup_{K\in\mathbb{T}_h} K = \mathcal{D}$. We denote the local polynomial function space as $V(K) = \{\text{all polynomials on } K \text{ of degree equal to or lesser than } p_K\}$. The dimension of this local space is determined by the polynomial order p_K. Note that all the polynomial orders $\{p_K, K \in \mathbb{T}_h\}$ do not have to be the same for all elements. We then multiply the transport equation Eq. (2.3) with a test function $\Psi^*_m \in V(K)$ and integrate it over element K. After integrating by parts the streaming term $(\vec{\Omega}_m \cdot \vec{\nabla})$ and employing *the upwind scheme* for the flux values on the upwind boundary of K, we obtain the Galerkin weighted-residual formula for a given angular direction $\vec{\Omega}_m$:

$$\begin{aligned}
&\text{Find} \quad \Psi_m \in V(K), \quad \text{such that} \quad \forall \Psi^*_m \in V(K), \\
&\big(\Psi_m, (-\vec{\Omega}_m \cdot \vec{\nabla} + \sigma_t)\Psi^*_m\big)_K + \big\langle \Psi^-_m, \Psi^{*-}_m \big\rangle_{\partial K^+} - \big\langle \Psi^-_m, \Psi^{*+}_m \big\rangle_{\partial K^- \setminus \partial\mathcal{D}} \\
&\qquad - \big\langle \Psi^{inc}_m, \Psi^{*+}_m \big\rangle_{\partial K^- \cap \partial\mathcal{D}^d} - \big\langle \Psi^+_{m'}, \Psi^{*+}_m \big\rangle_{\partial K^- \cap \partial\mathcal{D}^r} \\
&\qquad\qquad = \sum_{n=0}^{N_a} \sum_{k=-n}^{n} \frac{2n+1}{4\pi} Y_{n,k}(\vec{\Omega}_m)\big(\sigma_{s,n}\Phi_{n,k} + Q_{n,k}, \Psi^*_m\big)_K \qquad (2.7)
\end{aligned}$$

where ∂K^- is the inflow boundary, ∂K^+ is the outflow boundary. The traces f^+ and f^- are defined with respect to the particle direction $\vec{\Omega}_m$, i.e., on an inflow boundary, f^+ (resp. f^-) is the value of f taken from within element K (resp. from the upwind neighbor) and on an outflow boundary, f^+ (resp. f^-) is the value of f taken from the downwind neighbor (resp. from within element K). These definitions are expressed

as follows:

$$\Psi_m^+ \equiv \lim_{s\to 0^+} \Psi_m(\vec{r}+s\vec{\Omega}_m) \tag{2.8}$$

$$\Psi_m^- \equiv \lim_{s\to 0^-} \Psi_m(\vec{r}+s\vec{\Omega}_m) \tag{2.9}$$

$$(f,g)_K \equiv \int_K f\, g\, \mathrm{d}\vec{r} \tag{2.10}$$

$$\langle f,g\rangle_e \equiv \int_e |\vec{\Omega}_m\cdot\vec{n}(\vec{r})| f\, g\, \mathrm{d}s \tag{2.11}$$

$$\partial K^+ = \left\{\vec{r}\in\partial K : \vec{n}(\vec{r})\cdot\vec{\Omega}_m \geq 0\right\} \tag{2.12}$$

$$\partial K^- = \left\{\vec{r}\in\partial K : \vec{n}(\vec{r})\cdot\vec{\Omega}_m < 0\right\} \tag{2.13}$$

$\vec{n}$ is the outward unit normal vector of element K, e represent any edge (or face in 3-D) of K. Note that (i) in the case of straight element boundaries, the quantity $\vec{\Omega}_m\cdot\vec{n}(\vec{r})$ can be factored out of the boundary integrals and that (ii) $\Phi_{n,k}\in V(K)$ because $\Psi_m\in V(K)$. This is a Galerkin scheme because the function space in which the solution is sought is also the function space of the test functions.

It needs to be pointed out that while the test functions on the edges are always taken from within the element, the primal functions on the edges are taken based on the upwind (upstream) with respect to the streaming direction; thus, for outgoing edges, the angular flux is taken from within the element K whereas for incoming edges, its value is taken from the upwind neighbor element, leading to the so-called upwind scheme.

This weighted-residual formula states that once we know the solution values on the upwind elements, we can solve the local system for the flux value within element K. The local balance is conserved, i.e., the total collision in the element is equal to the total source, including the scattering and external source, plus the net in-leakage through the upwind sides minus the net out-leakage through the downwind sides. This simple upwind scheme is the essence of DGFEM; for fluid conservation laws,

a more complicated treatment of the numerical fluxes on the inter-element edges is performed.

Finally, for completeness we also provide another (more common) variant for the weighted-residual formula. The formula below and its associated definitions are *only* given here because they are widely used in numerical fluid flows and would help a reader familiar with that notation understand the notation we have chosen.

$$\begin{aligned}
&\text{Find} \quad \Psi_m \in V(K), \quad \text{such that} \quad \forall \Psi_m^* \in V(K), \\
&\left(\Psi_m, (-\vec{\Omega}_m \cdot \vec{\nabla} + \sigma_t)\Psi_m^*\right)_K + \left\langle \Psi_m^+, \Psi_m^{*+} \right\rangle_{\partial K^+} - \left\langle \Psi_m^-, \Psi_m^{*+} \right\rangle_{\partial K^- \setminus \partial \mathcal{D}} \\
&\qquad - \left\langle \Psi_m^{inc}, \Psi_m^{*+} \right\rangle_{\partial K^- \cap \partial \mathcal{D}^d} - \left\langle \Psi_{m'}^+, \Psi_m^{*+} \right\rangle_{\partial K^- \cap \partial \mathcal{D}^r} \\
&\qquad\qquad = \sum_{n=0}^{N_a} \sum_{k=-n}^{n} \frac{2n+1}{4\pi} Y_{n,k}(\vec{\Omega}_m)\left(\sigma_{s,n}\Phi_{n,k} + Q_{n,k}, \Psi_m^*\right)_K \qquad (2.14)
\end{aligned}$$

where ∂K^- is the inflow boundary, ∂K^+ is the outflow boundary, f^+ denotes the restriction (trace) of the function f taken from within the element K, and f^- represents the restriction of the function f taken from the neighboring element of K. These definitions are expressed as follows:

$$u = \text{sgn}\left(\vec{n}(\vec{r}) \cdot \vec{\Omega}_m\right) = +1 \text{ for outflow, or } -1 \text{ for inflow} \qquad (2.15)$$

$$\Psi_m^{\pm} \equiv \lim_{s \to 0^{\pm}} \Psi_m(\vec{r} - u\, s\, \vec{\Omega}_m) \qquad (2.16)$$

The differences between the two notations are related to the definitions of the numerical "traces". The advantage of Eq. (2.14) is that the + sign always denotes the value taken from within element K, be it for an inflow or outflow boundary, whereas this is not the case in Eq. (2.7). Again, Eqs. (2.14) through (2.16) will not be employed further here and have been given for completeness only.

3. Variational Form

Integrating by parts the streaming term of the local formula once again, multiplying the result with the angular weight w_m and summing over all elements and all directions, we obtain the variational form for the one-group S_N equation:

$$\begin{aligned}
&\sum_{m=1}^{M} w_m \left[\left((\vec{\Omega}_m \cdot \vec{\nabla} + \sigma_t)\Psi_m, \Psi_m^* \right)_{\mathcal{D}} + \left\langle [\![\Psi_m]\!], \Psi_m^{*+} \right\rangle_{E_h^i} \right] + \\
&\qquad \sum_{m=1}^{M} w_m \left\langle \Psi_m, \Psi_m^* \right\rangle_{\partial \mathcal{D}_m^-} - \sum_{m=1}^{M} w_m \left\langle \Psi_{m'}, \Psi_m^* \right\rangle_{\partial \mathcal{D}_m^{r-}} \\
&\quad = \sum_{m=1}^{M} w_m \sum_{n=0}^{N_a} \sum_{k=-n}^{n} \frac{2n+1}{4\pi} Y_{n,k}(\vec{\Omega}_m) \left(\sigma_{s,n} \Phi_{n,k} + Q_{n,k}, \Psi_m^* \right)_K + \\
&\qquad \sum_{m=1}^{M} w_m \left\langle \Psi_m^{inc}, \Psi_m^* \right\rangle_{\partial \mathcal{D}_m^{d-}}
\end{aligned} \tag{2.17}$$

with the following definitions

$$[\![\Psi_m]\!] = \Psi_m^+ - \Psi_m^- \tag{2.18}$$

$$(f,g)_{\mathcal{D}} \equiv \sum_{K \in \mathbb{T}_h} (f,g)_K \tag{2.19}$$

$$\langle f,g \rangle_{E_h^i} \equiv \sum_{e \in E_h^i} \langle f,g \rangle_e \,, \tag{2.20}$$

where $E_h^i = \cup_{K \in \mathbb{T}_h} \partial K \backslash \partial \mathcal{D}$ is the set of all interior edges (more generally, the dimension of the interior edges is $d-1$ for a computational domain of dimension d). For simplicity, we have dropped the $\pm$ superscript for the angular fluxes appearing in the boundary terms later because the angular fluxes inside the domain are always used. More detailed derivations regarding DGFEM for the first-order equation can be found in Section B of Appendix B. If we change the sequence of summation over directions

and summation over elements for the boundary terms, we obtain, after some algebra:

Find $\Psi_m \in W^h_{\mathcal{D}}, \quad m = 1, \cdots, M$ such that:

$$b(\Psi, \Psi^*) - \sum_{e \in \partial \mathcal{D}^r} \sum_{\vec{\Omega}_m \cdot \vec{n}_b < 0} w_m \left\langle \Psi_{m'}, \Psi^*_m \right\rangle_e - \sum_{n=0}^{N} \sum_{k=-n}^{n} \frac{2n+1}{4\pi} \left(\sigma_{s,n} \Phi_{n,k}, \Phi^*_{n,k}\right)_{\mathcal{D}} = l(\Psi^*) \tag{2.21}$$

$$\forall \Psi^*_m \in W^h_{\mathcal{D}}, \quad m = 1, \cdots, M$$

where the bilinear and linear forms are

$$\begin{aligned} b(\Psi, \Psi^*) = & \sum_{m=1}^{M} w_m \left((\vec{\Omega}_m \cdot \vec{\nabla} + \sigma_t)\Psi_m, \Psi^*_m\right)_{\mathcal{D}} + \sum_{m=1}^{M} w_m \left\langle [\![\Psi_m]\!], \Psi^{*+}_m \right\rangle_{E^i_h} \\ & + \sum_{e \in \partial \mathcal{D}} \sum_{\vec{\Omega}_m \cdot \vec{n}_b < 0} w_m \left\langle \Psi_m, \Psi^*_m \right\rangle_e \end{aligned} \tag{2.22}$$

$$l(\Psi^*) = \sum_{n=0}^{N} \sum_{k=-n}^{n} \frac{2n+1}{4\pi} \left(Q_{n,k}, \Phi^*_{n,k}\right)_{\mathcal{D}} + \sum_{e \in \partial \mathcal{D}^d} \sum_{\vec{\Omega}_m \cdot \vec{n}_b < 0} w_m \left\langle \Psi^{inc}_m, \Psi^*_m \right\rangle_e \tag{2.23}$$

and $W^h_{\mathcal{D}} = \{\Psi \in L^2(\mathcal{D});\ \Psi|_K \in V(K), \forall K \in \mathbb{T}_h\}$ is the function space in which the solution is sought. Note that the functions in this space may not be continuous across the interior edges. There are three terms on the left-hand-side of Eq. (2.21): the streaming operators from M simple transport (e.g., one direction) equations $b(\Psi_m, \Psi^*_m)$, the reflecting-boundary term and the scattering term. For notational simplicity, we define the following bilinear form,

$$\begin{aligned} a(\Psi, \Psi^*) = b(\Psi, \Psi^*) - & \sum_{e \in \partial \mathcal{D}^r} \sum_{\vec{\Omega}_m \cdot \vec{n}_b < 0} w_m \left\langle \Psi_{m'}, \Psi^*_m \right\rangle_e - \\ & \sum_{n=0}^{N} \sum_{k=-n}^{n} \frac{2n+1}{4\pi} \left(\sigma_{s,n} \Phi_{n,k}, \Phi^*_{n,k}\right)_{\mathcal{D}} \end{aligned} \tag{2.24}$$

The variational form for the multigroup S_N transport equation can be found in Section C of Appendix C.

With the same procedure, we obtain the variational form for the adjoint equation.

Find $\Psi_m^* \in W_{\mathcal{D}}^h, \quad m = 1, \cdots, M$ such that:

$$b^*(\Psi, \Psi^*) - \sum_{e \in \partial \mathcal{D}^r} \sum_{\vec{\Omega}_m \cdot \vec{n}_b > 0} w_m \left\langle \Psi_m, \Psi_{m'}^* \right\rangle_e - \sum_{n=0}^{N} \sum_{k=-n}^{n} \frac{2n+1}{4\pi} \left(\Phi_{n,k}, \sigma_{s,n} \Phi_{n,k}^* \right)_{\mathcal{D}} = l^*(\Psi) \tag{2.25}$$

$\forall \Psi_m \in W_{\mathcal{D}}^h, \quad m = 1, \cdots, M$

where

$$b^*(\Psi, \Psi^*) = \sum_{m=1}^{M} w_m \left(\Psi_m, (-\vec{\Omega}_m \cdot \vec{\nabla} + \sigma_t) \Psi_m^* \right)_{\mathcal{D}} - \sum_{m=1}^{M} w_m \left\langle \Psi_m^-, [\![\Psi_m^*]\!] \right\rangle_{E_h^i} + \sum_{e \in \partial \mathcal{D}} \sum_{\vec{\Omega}_m \cdot \vec{n}_b > 0} w_m \left\langle \Psi_m, \Psi_m^* \right\rangle_e \tag{2.26}$$

$$l^*(\Psi) = \sum_{n=0}^{N} \sum_{k=-n}^{n} \frac{2n+1}{4\pi} \left(\Phi_{n,k}, Q_{n,k}^* \right)_{\mathcal{D}} + \sum_{e \in \partial \mathcal{D}^d} \sum_{\vec{\Omega}_m \cdot \vec{n}_b > 0} w_m \left\langle \Psi_m, \Psi_m^{*inc} \right\rangle_e \tag{2.27}$$

In Section B of Appendix B, we prove that $b(\Psi_m, \Psi_m^*) = b^*(\Psi_m, \Psi_m^*)$. Furthermore, it is easy to note that the scattering terms are self adjoint. Finally, if the quadrature set is properly designed, the reflecting terms are also self-adjoint. So the bilinear forms for the primal equation and the adjoint equation are the same, i.e., $a(\Psi_m, \Psi_m^*) = a^*(\Psi_m, \Psi_m^*)$.

The only difference between the spatially discretized bilinear form $a(\Psi_m, \Psi_m^*)$ and the form stemming from the continuous one-group S_N equation is the additional jump term on all interior edges. It is easy to see that this term vanishes in the continuous case, i.e., jumps along the streaming directions are zero. So we have the Galerkin orthogonality (the jumps in the exact solution are zero)

$$a(\Psi - \Psi_{exact}, \Psi^*) = 0, \quad \forall \Psi_m^* \in W_{\mathcal{D}}^h, \quad m = 1, \cdots, M \tag{2.28}$$

Because of the positive definiteness of the form a, i.e.,

$$a(\Psi, \Psi) > 0 \tag{2.29}$$

we can define a so-called DG-norm

$$\|\!|\Psi|\!\|_{DG} = a(\Psi, \Psi) \tag{2.30}$$

Although we do not have to write down the variational form to solve the S_N equation with DGFEM, it is very useful for the study of theoretical convergence. It has been proved that with the uniform polynomial order, i.e., $p_K = p, \forall K \in \mathbb{T}_h$ [88]

$$\|\!|\Psi - \Psi_{exact}|\!\|_{DG} \leq C\frac{h^q}{(p+1)^q} \tag{2.31}$$

where $q = \min(p + 1/2, 1)$ for continuous transport solution, or $q = \min(p + 1/2, 0)$ if discontinuities are present in the transport solution. h is the maximum diameter of all elements, which is fixed for a given mesh. C is a constant independent of the mesh. It can be proved the convergence holds also for the scalar flux measured in the L_2-norm,

$$\|\Phi - \Phi_{exact}\|_2 \leq C\frac{h^{q+\frac{1}{2}}}{(p+1)^q} \tag{2.32}$$

The variational form derived here will be employed to obtain conforming preconditioners for the DSA schemes, in Chapter III.

4. Higher-Order Shape Functions

A good choice of shape functions is important for an easy and efficient implementation of higher-order DGFEM. Over the course of the last decade, higher-order and spectral finite elements have enjoyed many theoretical and applied developments; for instance,

in the computational fluid dynamics field, several novel computer code systems are based on DGFEM [75, 89, 90] and several authors describe basis function options for higher-order finite elements [5, 91, 4].

We have in mind the subsequent development of a mesh adaptive *hp*-code, where the mesh can either be locally subdivided (*h*-refinement) or the polynomial order can be locally increased (*p*-refinement). In such applications, the use of *hierarchical basis functions* has been shown to allow for an efficient implementation and a flexible polynomial order selection (in neutron diffusion codes, hierarchical functions have been employed for 2-D quadrilateral elements in [87, 56].) Hierarchical bases are sets of functions where bases of lower-order are successively nested in higher degree bases. Thus, an increase in the polynomial order means only adding new functions to the current set, not re-establishing a whole new set as would be the case with the traditional Lagrange polynomial bases. As a result, an elementary matrix for a polynomial approximation of order p contains, embedded in it, the matrices of order $p-1$, $p-2$, $\cdots$, 2, and 1, as shown on Fig. II-1 for $p = 4$.

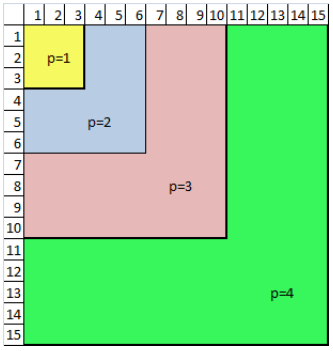


Fig. II-1. Hierarchical structure of elementary matrices.

The nested feature of the elementary matrices can be exploited for calculations where the polynomial order is not held constant throughout the domain. Even for

a constant polynomial order, hierarchical sets are attractive: they possess better condition numbers [5, 92, 86] and provide a simple representation of the solution on any edge; this latter point is important for inter-element communication in the upwinding procedure. In this chapter, we implemented and analyzed the convergence properties of hierarchical basis functions for 2-D unstructured geometries meshed with *triangular elements*. We assume the cross sections are *constant* per element. The unstructured meshes shown in the result sections are generated with the Triangle mesh generator [93]. Before providing the definition of the basis functions used, we need to note that the basis functions are typically not written for any element K of the physical geometry but for a reference element, obtained after applying an affine transformation that maps a physical element onto the reference element F_K^{-1}. Here, the reference element is the triangle $\widehat{K}$, with the three vertices $V_1 = (-1, -1)$, $V_2 = (1, -1)$ and $V_3 = (-1, 1)$; $\widehat{K}$ is defined as follows:

$$\widehat{K} = \left\{ \xi_1, \xi_2 \in \mathbb{R}^2; \ -1 < \xi_1, \xi_2 \ ; \ \xi_1 + \xi_2 < 0 \right\} \tag{2.33}$$

where we have introduced the reference coordinate system ξ_1, ξ_2. The reference triangle is shown in Fig. II-2.

The local ordering follows four rules, illustrated in Fig. II-2 and given below

1. the three vertices are numbered counter-clockwise.

2. edge 1 is the edge opposite of vertex 1 starting from vertex 2.

3. edge 2 is the edge opposite of vertex 2 starting from vertex 3.

4. edge 3 is the edge opposite of vertex 3 starting from vertex 1.

The linear (first-order) basis functions associated with three vertices are the

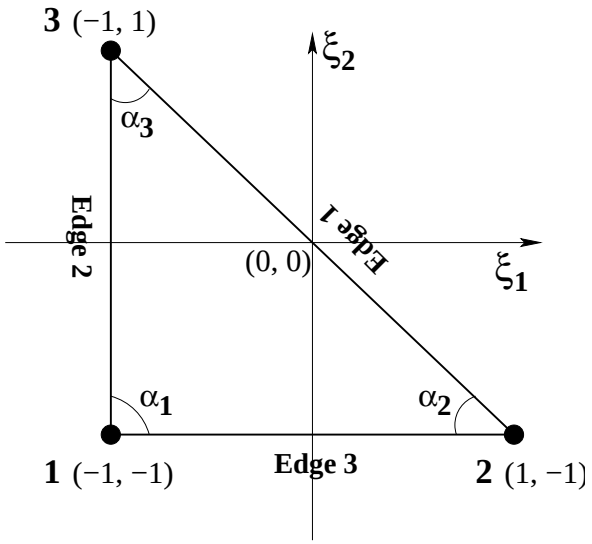


Fig. II-2. Reference triangular element.

standard ones:

$$\widehat{b}_1(\xi_1,\xi_2) = -\frac{\xi_1+\xi_2}{2} \tag{2.34}$$

$$\widehat{b}_2(\xi_1,\xi_2) = \frac{\xi_1+1}{2} \tag{2.35}$$

$$\widehat{b}_3(\xi_1,\xi_2) = \frac{\xi_2+1}{2} \tag{2.36}$$

The $\widehat{\ }$ symbol denotes that the functions are defined on the reference element $\widehat{K}$.

If the coordinates of the physical element K are (x_1, y_1), (x_2, y_2) and (x_3, y_3). Then, the affine transformation from the reference coordinate system to the physical coordinate system is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \widehat{b}_1(\xi_1,\xi_2)\begin{bmatrix} -1 \\ -1 \end{bmatrix} + \widehat{b}_2(\xi_1,\xi_2)\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \widehat{b}_3(\xi_1,\xi_2)\begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{2.37}$$

$$= \frac{1}{2}\begin{bmatrix} x_2+x_3 \\ y_2+y_3 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} x_2-x_1 & x_3-x_1 \\ y_2-y_1 & y_3-y_1 \end{bmatrix}\begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} \tag{2.38}$$

which can be written in matrix form as

$$\mathbf{x} = F(\xi) = \mathbf{x}_0 + J\xi \tag{2.39}$$

where J is the Jacobian matrix of the transformation. The determinant of J is half of the triangle area. Since the vertices are numbered counter clock-wise, the determinant is always positive.

The higher-order functions consist of edge functions (functions that are non-zero only on one of the edges) and interior functions (functions that are zero on all three edges). For a basis of order p, there are $p-1$ edge functions per edge and $(p-1)(p-2)/2$ interior functions. Therefore, the total number of shape functions, and thus of unknowns per triangle, is given by $N(p) = (p+1)(p+2)/2$. We have followed the definition by Solin and Sherwin [5, 4] for the hierarchical basis functions.

For edge 1, linking vertices V_2 and V_3, we have

$$\widehat{b}_{N(k-1)+1}(\xi_1,\xi_2) = \widehat{b}_2(\xi_1,\xi_2)\widehat{b}_3(\xi_1,\xi_2)\phi_{k-2}\left(\frac{2\xi_1+\xi_2+1}{2}\right) \text{ for } 2 \le k \le p \tag{2.40}$$

For edge 2, linking vertices V_3 and V_1, we have

$$\widehat{b}_{N(k-1)+2}(\xi_1,\xi_2) = \widehat{b}_3(\xi_1,\xi_2)\widehat{b}_1(\xi_1,\xi_2)\phi_{k-2}\left(\frac{\xi_2-\xi_1}{2}\right) \text{ for } 2 \le k \le p \tag{2.41}$$

For edge 3, linking vertices V_1 and V_2, we have

$$\widehat{b}_{N(k-1)+3}(\xi_1,\xi_2) = \widehat{b}_1(\xi_1,\xi_2)\widehat{b}_2(\xi_1,\xi_2)\phi_{k-2}\left(-\frac{\xi_1+2\xi_2+1}{2}\right) \text{ for } 2 \le k \le p \tag{2.42}$$

Finally, the interior functions are given by

$$\widehat{b}_{N(n_1+n_2)+n_1+3}(\xi_1,\xi_2) = \widehat{b}_1(\xi_1,\xi_2)\widehat{b}_2(\xi_1,\xi_2)\widehat{b}_3(\xi_1,\xi_2)\times \\ \phi_{n_1-1}\left(\frac{2\xi_1+\xi_2+1}{2}\right))\phi_{n_2-1}\left(-\frac{\xi_1+2\xi_2+1}{2}\right)$$

$$\text{for } 1 \leq n_1, n_2 \; n_1+n_2 \leq p-1 \tag{2.43}$$

We note that edge (resp. interior) functions only appear for $p > 1$ (resp. $p > 2$). The kernel functions $\phi_k(z)$ ($k \geq 0$) are given in terms of the 1-D Lobatto polynomials $L_{k+2}(x)$ of order $k+2$ defined on the $[-1,+1]$ interval:

$$\phi_k(z) = \frac{L_{k+2}(z)}{L_1(z)L_2(z)}, \; k \geq 0 \tag{2.44}$$

The shape functions on a physical element K are obtained as follows

$$\mathbf{b}_K = \widehat{\mathbf{b}}_{p_K} \circ F_K^{-1} \tag{2.45}$$

where $\hat{\mathbf{b}}_p = \mathbf{col}(\hat{b}_j)$ and p is the polynomial order of element K. For notational simplicity, we shall drop the subscript K.

The angular flux is, therefore, expanded on the set of local spatial shape functions b_j of order p,

$$\Psi_m(\vec{r}) = \sum_{j=1}^{N(p)} b_j(\vec{r})\Psi_{m,j} = \mathbf{b}^T\mathbf{\Psi}_m \tag{2.46}$$

with $\mathbf{\Psi}_m = \mathbf{col}(\Psi_{m,j})$ the vector of angular flux unknowns on cell K in direction m. An arbitrary test function can also be expanded as

$$\Psi^*_m(\vec{r}) = \sum_{j=1}^{N(p)} b_j(\vec{r})\Psi^*_{m,j} = \mathbf{b}^T\mathbf{\Psi}^*_m = \mathbf{\Psi}^{*T}_m\mathbf{b} \tag{2.47}$$

where $\mathbf{\Psi}^*_m = \mathbf{col}(\Psi^*_{m,j})$. Now we are ready to compute each elementary matrix on the reference triangle via a mapping from K to $\widehat{K}$.

a. Streaming Matrix

Let us first consider the cell integral of the streaming term $-(\Psi_m, \vec{\Omega}_m \cdot \vec{\nabla}\Psi_m^*)_K$ from Eq. (2.7). We define the streaming matrix as

$$\mathbf{G}_m = -\int_K (\nabla_{\mathbf{x}}\mathbf{b})\vec{\Omega}_m \, \mathbf{b}^T \, dxdy \tag{2.48}$$

where $\nabla_{\mathbf{x}} = [\partial_x, \, \partial_y]$ is the usual gradient operator in the physical space, and the cosine directors of $\vec{\Omega}_m = [\Omega_{m,x}, \Omega_{m,y}]^T$ are $\Omega_{m,x} = \vec{\Omega}_m \cdot \vec{e}_x$ and $\Omega_{m,y} = \vec{\Omega}_m \cdot \vec{e}_y$. The gradient operator is defined as a row vector so that the expression can be understood as normal matrix multiplications. We define ∇_ξ as the gradient in the reference space, with the local mapping from Eq. (2.38),

$$\nabla_\xi() \;=\; \nabla_{\mathbf{x}}()J \tag{2.49}$$

$$\nabla_{\mathbf{x}}() \;=\; \nabla_\xi()J^{-1} \tag{2.50}$$

The streaming matrix is bolded to indicate it depends on the shape of the element. We will use this notational rule as default for all elementary matrices that depend on the shape of the physical element. In the next section, we will see that the reference mass matrix is unbolded because it is shape-independent. With the change of variables theorem, we obtain

$$\mathbf{G}_m = \int_{\widehat{K}} \nabla_\xi \widehat{\mathbf{b}}(\xi_1, \xi_2) \left[-J^{-1} \cdot \vec{\Omega}_m \det(J)\right] \widehat{\mathbf{b}}^T(\xi_1, \xi_2) \, d\xi_1 d\xi_2 \tag{2.51}$$

We define orientations of three edges of the triangle as,

$$t_{m,i} = \frac{\vec{\Omega}_m \cdot \vec{n}_i L_i}{6}, \quad i = 1, \cdots, 3 \tag{2.52}$$

where L_i are lengths of the edges. Note that

$$t_{m,1} + t_{m,2} + t_{m,3} \equiv 0 \tag{2.53}$$

and that

$$-J^{-1}\cdot\vec{\Omega}_m \det(J) = 3\begin{bmatrix} t_{m,2} \\ t_{m,3} \end{bmatrix} \tag{2.54}$$

For simplicity, we only present $\mathbf{G}_m$ for $p = 2$. $\mathbf{G}_m$ with arbitrary polynomial orders can be generated from the Mathematica notebook given in Appendix F.

$$\begin{bmatrix} \begin{bmatrix} t_{m,1} & t_{m,1} & t_{m,1} \\ t_{m,2} & t_{m,2} & t_{m,2} \\ t_{m,3} & t_{m,3} & t_{m,3} \end{bmatrix} & -\frac{\sqrt{6}}{4}\begin{bmatrix} t_{m,1} & t_{m,1} & t_{m,1} \\ t_{m,2} & t_{m,2} & t_{m,2} \\ t_{m,3} & t_{m,3} & t_{m,3} \end{bmatrix} \\ \frac{\sqrt{6}}{4}\begin{bmatrix} t_{m,3}-t_{m,2} & t_{m,3}-t_{m,1} & t_{m,3} \\ t_{m,1} & t_{m,1}-t_{m,3} & t_{m,1}-t_{m,2} \\ t_{m,2}-t_{m,3} & t_{m,2} & t_{m,2}-t_{m,1} \end{bmatrix} & -\frac{3}{10}\begin{bmatrix} 2t_{m,3} & t_{m,3}-t_{m,1} & t_{m,3}-t_{m,2} \\ t_{m,1}-t_{m,3} & 2t_{m,1} & t_{m,1}-t_{m,2} \\ t_{m,2}-t_{m,3} & t_{m,2}-t_{m,1} & 2t_{m,2} \end{bmatrix} \end{bmatrix}$$

The streaming matrix is not symmetric and has units of length. With the streaming matrix, the streaming term

$$-(\Psi_m, \vec{\Omega}_m\cdot\vec{\nabla}\Psi_m^*)_K = \mathbf{\Psi}_m^{*T}\mathbf{G}_m\mathbf{\Psi}_m \tag{2.55}$$

where $\mathbf{\Psi}_m$ and $\mathbf{\Psi}_m^*$ are the local angular flux vector and the test function vector.

b. Mass Matrix

For the collision term $(\sigma_t\Psi_m, \Psi_m^*)_K$, we define reference mass matrix

$$\mathrm{M} = 6\int_{\widehat{K}} \widehat{\mathbf{b}}(\xi_1,\xi_2)\,\widehat{\mathbf{b}}^T(\xi_1,\xi_2)\,d\xi_1 d\xi_2, \tag{2.56}$$

and the local mass matrix is equal to

$$\sigma_{t,K}\int_K \mathbf{b}\,\mathbf{b}^T\,dxdy = \sigma_{t,K}\frac{\det(J)}{6}\mathrm{M} = \frac{\sigma_{t,K}A}{12}\mathrm{M} \tag{2.57}$$

with A the triangle area.

The reference mass matrix M, up to order 4, is given below

$$\begin{bmatrix}
2 & 1 & 1 & \frac{-2\sqrt{6}}{5} & \frac{-\sqrt{6}}{5} & \frac{-2\sqrt{6}}{5} & \frac{\sqrt{\frac{2}{5}}}{3} & 0 & \frac{-\sqrt{\frac{2}{5}}}{3} & \frac{2}{5} & \frac{\sqrt{\frac{2}{7}}}{5} & \frac{11}{30\sqrt{14}} & \frac{\sqrt{\frac{2}{7}}}{5} & \frac{2}{7\sqrt{15}} & \frac{-2}{7\sqrt{15}} \\
1 & 2 & 1 & \frac{-2\sqrt{6}}{5} & \frac{-2\sqrt{6}}{5} & \frac{-\sqrt{6}}{5} & \frac{-\sqrt{\frac{2}{5}}}{3} & \frac{\sqrt{\frac{2}{5}}}{3} & 0 & \frac{2}{5} & \frac{\sqrt{\frac{2}{7}}}{5} & \frac{\sqrt{\frac{2}{7}}}{5} & \frac{11}{30\sqrt{14}} & 0 & \frac{2}{7\sqrt{15}} \\
1 & 1 & 2 & \frac{-\sqrt{6}}{5} & \frac{-2\sqrt{6}}{5} & \frac{-2\sqrt{6}}{5} & 0 & \frac{-\sqrt{\frac{2}{5}}}{3} & \frac{\sqrt{\frac{2}{5}}}{3} & \frac{2}{5} & \frac{11}{30\sqrt{14}} & \frac{\sqrt{\frac{2}{7}}}{5} & \frac{\sqrt{\frac{2}{7}}}{5} & \frac{-2}{7\sqrt{15}} & 0 \\
\frac{-2\sqrt{6}}{5} & \frac{-2\sqrt{6}}{5} & \frac{-\sqrt{6}}{5} & \frac{4}{5} & \frac{2}{5} & \frac{2}{5} & 0 & \frac{-2}{7\sqrt{15}} & \frac{2}{7\sqrt{15}} & \frac{-4\sqrt{6}}{35} & \frac{-13}{20\sqrt{21}} & \frac{-13}{40\sqrt{21}} & \frac{-13}{40\sqrt{21}} & \frac{-1}{7\sqrt{10}} & 0 \\
\frac{-\sqrt{6}}{5} & \frac{-2\sqrt{6}}{5} & \frac{-2\sqrt{6}}{5} & \frac{2}{5} & \frac{4}{5} & \frac{2}{5} & \frac{2}{7\sqrt{15}} & 0 & \frac{-2}{7\sqrt{15}} & \frac{-4\sqrt{6}}{35} & \frac{-13}{40\sqrt{21}} & \frac{-13}{20\sqrt{21}} & \frac{-13}{40\sqrt{21}} & \frac{1}{7\sqrt{10}} & \frac{-1}{7\sqrt{10}} \\
\frac{-2\sqrt{6}}{5} & \frac{-\sqrt{6}}{5} & \frac{-2\sqrt{6}}{5} & \frac{2}{5} & \frac{2}{5} & \frac{4}{5} & \frac{-2}{7\sqrt{15}} & \frac{2}{7\sqrt{15}} & 0 & \frac{-4\sqrt{6}}{35} & \frac{-13}{40\sqrt{21}} & \frac{-13}{40\sqrt{21}} & \frac{-13}{20\sqrt{21}} & 0 & \frac{1}{7\sqrt{10}} \\
\frac{\sqrt{\frac{2}{5}}}{3} & \frac{-\sqrt{\frac{2}{5}}}{3} & 0 & 0 & \frac{2}{7\sqrt{15}} & \frac{-2}{7\sqrt{15}} & \frac{1}{7} & -\frac{1}{21} & -\frac{1}{21} & 0 & 0 & \frac{1}{24\sqrt{35}} & \frac{-1}{24\sqrt{35}} & \frac{1}{21\sqrt{6}} & \frac{-\sqrt{\frac{2}{3}}}{21} \\
0 & \frac{\sqrt{\frac{2}{5}}}{3} & \frac{-\sqrt{\frac{2}{5}}}{3} & \frac{-2}{7\sqrt{15}} & 0 & \frac{2}{7\sqrt{15}} & -\frac{1}{21} & \frac{1}{7} & -\frac{1}{21} & 0 & \frac{-1}{24\sqrt{35}} & 0 & \frac{1}{24\sqrt{35}} & \frac{1}{21\sqrt{6}} & \frac{1}{21\sqrt{6}} \\
\frac{-\sqrt{\frac{2}{5}}}{3} & 0 & \frac{\sqrt{\frac{2}{5}}}{3} & \frac{2}{7\sqrt{15}} & \frac{-2}{7\sqrt{15}} & 0 & -\frac{1}{21} & -\frac{1}{21} & \frac{1}{7} & 0 & \frac{1}{24\sqrt{35}} & \frac{-1}{24\sqrt{35}} & 0 & \frac{-\sqrt{\frac{2}{3}}}{21} & \frac{1}{21\sqrt{6}} \\
\frac{2}{5} & \frac{2}{5} & \frac{2}{5} & \frac{-4\sqrt{6}}{35} & \frac{-4\sqrt{6}}{35} & \frac{-4\sqrt{6}}{35} & 0 & 0 & 0 & \frac{6}{35} & \frac{\sqrt{\frac{7}{2}}}{30} & \frac{\sqrt{\frac{7}{2}}}{30} & \frac{\sqrt{\frac{7}{2}}}{30} & 0 & 0 \\
\frac{\sqrt{\frac{2}{7}}}{5} & \frac{\sqrt{\frac{2}{7}}}{5} & \frac{11}{30\sqrt{14}} & \frac{-13}{20\sqrt{21}} & \frac{-13}{40\sqrt{21}} & \frac{-13}{40\sqrt{21}} & 0 & \frac{-1}{24\sqrt{35}} & \frac{1}{24\sqrt{35}} & \frac{\sqrt{\frac{7}{2}}}{30} & \frac{3}{40} & \frac{13}{480} & \frac{13}{480} & \frac{1}{12\sqrt{210}} & 0 \\
\frac{11}{30\sqrt{14}} & \frac{\sqrt{\frac{2}{7}}}{5} & \frac{\sqrt{\frac{2}{7}}}{5} & \frac{-13}{40\sqrt{21}} & \frac{-13}{20\sqrt{21}} & \frac{-13}{40\sqrt{21}} & \frac{1}{24\sqrt{35}} & 0 & \frac{-1}{24\sqrt{35}} & \frac{\sqrt{\frac{7}{2}}}{30} & \frac{13}{480} & \frac{3}{40} & \frac{13}{480} & \frac{-1}{12\sqrt{210}} & \frac{1}{12\sqrt{210}} \\
\frac{\sqrt{\frac{2}{7}}}{5} & \frac{11}{30\sqrt{14}} & \frac{\sqrt{\frac{2}{7}}}{5} & \frac{-13}{40\sqrt{21}} & \frac{-13}{40\sqrt{21}} & \frac{-13}{20\sqrt{21}} & \frac{-1}{24\sqrt{35}} & \frac{1}{24\sqrt{35}} & 0 & \frac{\sqrt{\frac{7}{2}}}{30} & \frac{13}{480} & \frac{13}{480} & \frac{3}{40} & 0 & \frac{-1}{12\sqrt{210}} \\
\frac{2}{7\sqrt{15}} & 0 & \frac{-2}{7\sqrt{15}} & \frac{-1}{7\sqrt{10}} & \frac{1}{7\sqrt{10}} & 0 & \frac{1}{21\sqrt{6}} & \frac{1}{21\sqrt{6}} & \frac{-\sqrt{\frac{2}{3}}}{21} & 0 & \frac{1}{12\sqrt{210}} & \frac{-1}{12\sqrt{210}} & 0 & \frac{2}{105} & -\frac{1}{105} \\
\frac{-2}{7\sqrt{15}} & \frac{2}{7\sqrt{15}} & 0 & 0 & \frac{-1}{7\sqrt{10}} & \frac{1}{7\sqrt{10}} & \frac{-\sqrt{\frac{2}{3}}}{21} & \frac{1}{21\sqrt{6}} & \frac{1}{21\sqrt{6}} & 0 & 0 & \frac{1}{12\sqrt{210}} & \frac{-1}{12\sqrt{210}} & -\frac{1}{105} & \frac{2}{105}
\end{bmatrix}$$

M is symmetric positive definite.

c. Type-1 Edge Matrix

We now consider the edge integral terms. We first define local mapping for three edges:

$$\begin{aligned}
\xi_1 &= (-s, s) \\
\xi_2 &= (-1, -s) \\
\xi_3 &= (s, -1)
\end{aligned} \tag{2.58}$$

We then define the type-1 edge matrix (denoted by the superscript 1) for any of the three edges denoted by the edge subscript index i,

$$\mathrm{E}_i^1 = 3 \int_{-1}^{1} \widehat{\mathbf{b}}(\xi_i) \widehat{\mathbf{b}}^T(\xi_i) \, \mathrm{d}s \tag{2.59}$$

We call this the type-1 edge matrix because additional types will be introduced for the DGFEM formulation of the diffusion equation, in Chapter III. For the edge matrices, the affine transformation is simply a one-dimensional mapping from both extremities of the edge onto the interval $[-1; +1]$. The determinant of the 1-D Jacobian matrix for the three edges $i = 1, 2, 3$, is half of the edge length.

The edge terms in the local system

$$\mathbf{H}_{m,i} = t_{m,i} \mathrm{E}_i^1, \quad i = 1, 2, 3 \tag{2.60}$$

where $t_{m,i}$ are the three local orientations defined in the previous subsection. The

three type-1 edge matrices are

$$
\mathrm{E}_1^1 = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & 2 & 1 & 0 & -\sqrt{\tfrac{3}{2}} & 0\,0 & \tfrac{1}{\sqrt{10}} & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & 1 & 2 & 0 & -\sqrt{\tfrac{3}{2}} & 0\,0 & -\tfrac{1}{\sqrt{10}} & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & -\sqrt{\tfrac{3}{2}} & -\sqrt{\tfrac{3}{2}} & 0 & \tfrac{6}{5} & 0\,0 & 0 & 0\,0\,0 & \frac{-\sqrt{\frac{3}{7}}}{5} & 0\,0\,0 \\
0 & 0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & \tfrac{1}{\sqrt{10}} & -\tfrac{1}{\sqrt{10}} & 0 & 0 & 0\,0 & \tfrac{2}{7} & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0 & \frac{-\sqrt{\frac{3}{7}}}{5} & 0\,0 & 0 & 0\,0\,0 & \tfrac{2}{15} & 0\,0\,0 \\
0 & 0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0
\end{bmatrix}
$$

$$
\mathrm{E}_2^1 = \begin{bmatrix}
2 & 0 & 1 & 0\,0 & -\sqrt{\tfrac{3}{2}} & 0\,0 & -\tfrac{1}{\sqrt{10}} & 0\,0\,0 & 0 & 0\,0 \\
0 & 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0 \\
1 & 0 & 2 & 0\,0 & -\sqrt{\tfrac{3}{2}} & 0\,0 & \tfrac{1}{\sqrt{10}} & 0\,0\,0 & 0 & 0\,0 \\
0 & 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0 \\
0 & 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0 \\
-\sqrt{\tfrac{3}{2}} & 0 & -\sqrt{\tfrac{3}{2}} & 0\,0 & \tfrac{6}{5} & 0\,0 & 0 & 0\,0\,0 & \frac{-\sqrt{\frac{3}{7}}}{5} & 0\,0 \\
0 & 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0 \\
0 & 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0 \\
-\tfrac{1}{\sqrt{10}} & 0 & \tfrac{1}{\sqrt{10}} & 0\,0 & 0 & 0\,0 & \tfrac{2}{7} & 0\,0\,0 & 0 & 0\,0 \\
0 & 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0 \\
0 & 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0 \\
0 & 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0 \\
0 & 0 & 0 & 0\,0 & \frac{-\sqrt{\frac{3}{7}}}{5} & 0\,0 & 0 & 0\,0\,0 & \tfrac{2}{15} & 0\,0 \\
0 & 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0 \\
0 & 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0
\end{bmatrix}
$$

$$
\mathrm{E}_3^1 = \begin{bmatrix}
2 & 1 & 0 & -\sqrt{\tfrac{3}{2}} & 0\,0 & \tfrac{1}{\sqrt{10}} & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
1 & 2 & 0 & -\sqrt{\tfrac{3}{2}} & 0\,0 & -\tfrac{1}{\sqrt{10}} & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
-\sqrt{\tfrac{3}{2}} & -\sqrt{\tfrac{3}{2}} & 0 & \tfrac{6}{5} & 0\,0 & 0 & 0\,0\,0 & \frac{-\sqrt{\frac{3}{7}}}{5} & 0\,0\,0\,0 \\
0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
\tfrac{1}{\sqrt{10}} & -\tfrac{1}{\sqrt{10}} & 0 & 0 & 0\,0 & \tfrac{2}{7} & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
0 & 0 & 0 & \frac{-\sqrt{\frac{3}{7}}}{5} & 0\,0 & 0 & 0\,0\,0 & \tfrac{2}{15} & 0\,0\,0\,0 \\
0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0\,0 \\
0 & 0 & 0 & 0 & 0\,0 & 0 & 0\,0\,0 & 0 & 0\,0\,0\,0
\end{bmatrix}
$$

Note that the definition of our basis functions allows for an easy access to the edge values, regardless of the chosen polynomial order; indeed, the flux on any edge is obtained by using the two non-zero linear basis functions on that edge and all the edge bubble functions for that edge. This is the reason why most of the entries in these three matrices are zero. This is particularly useful to efficiently compute the

upwinding contributions for any polynomial order and is an attractive feature for subsequent mesh adaptivity developments using a matrix-free scheme. If we select and extract rows and columns corresponding to the vertex and edge shape functions for a given edge, the 1-D reference mass matrix will be obtained. More details will be given in Chapter IV.

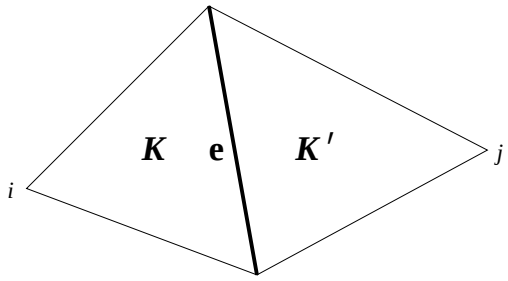


Fig. II-3. Interior edge definition: recall that for element K, the local edge ID is i because its opposite vertex is labeled i.

We conclude this section by describing the edge coupling in the upwind scheme. For now, let us assume that mesh is conforming (no local refinement), i.e., there is only one neighboring element on one side of any element. We define the coupling matrices for an interior edge showed in Fig. II-3 as

$$\mathrm{E}_{i,j}^{1C} = 3 \int_{-1}^{1} \widehat{\mathbf{b}}(\xi_i)\, \mathbf{b}^T(-\xi_j) ds \tag{2.61}$$

where j is the local edge ID of the upwind element K' that shares edge i of element K. Note that $\mathrm{E}_{i,i}^{1C}$ is not equal to E_i^1 because both elements are numbered counter-clock wise and we need to rotate the upwind solution vector to align it with the local solution vector on the common edge, i.e., we add a minus sign on ξ_j in equation

Eq. (2.61). One example for the type-1 edge coupling matrix is given below,

$$\mathrm{E}_{3,3}^{1C} = \begin{bmatrix}
1 & 2 & 0 & -\sqrt{\frac{3}{2}} & 0\ 0 & -\frac{1}{\sqrt{10}} & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
2 & 1 & 0 & -\sqrt{\frac{3}{2}} & 0\ 0 & \frac{1}{\sqrt{10}} & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & 0 & 0\ 0 & 0 & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
-\sqrt{\frac{3}{2}} & -\sqrt{\frac{3}{2}} & 0 & \frac{6}{5} & 0\ 0 & 0 & 0\ 0\ 0 & \frac{-\sqrt{\frac{3}{7}}}{5} & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & 0 & 0\ 0 & 0 & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & 0 & 0\ 0 & 0 & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
-\frac{1}{\sqrt{10}} & \frac{1}{\sqrt{10}} & 0 & 0 & 0\ 0 & -\frac{2}{7} & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & 0 & 0\ 0 & 0 & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & 0 & 0\ 0 & 0 & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & 0 & 0\ 0 & 0 & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & \frac{-\sqrt{\frac{3}{7}}}{5} & 0\ 0 & 0 & 0\ 0\ 0 & \frac{2}{15} & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & 0 & 0\ 0 & 0 & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & 0 & 0\ 0 & 0 & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & 0 & 0\ 0 & 0 & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0 \\
0 & 0 & 0 & 0 & 0\ 0 & 0 & 0\ 0\ 0 & 0 & 0\ 0\ 0\ 0
\end{bmatrix}$$

We can note a few sign changes with matrix E_3^1 due to the change in edge orientation.

Therefore, the upwind coupling matrices are

$$\overline{\mathbf{H}}_{i,K'} = t_{m,i}\mathrm{E}_{i,j(K')}^{1C} \tag{2.62}$$

with $j(K')$ the local edge ID of the upwind element K'.

d. Assembly Procedure for a Small Local Transport System

Considering three elements I, II and III in the domain shown in Fig. II-4. The local transport system matrix for element I, $\mathbf{A}^{(K=\mathrm{I})}$, is given by the streaming matrix, the mass matrix, the two downwind matrix

$$\mathbf{A}^{(K=\mathrm{I})} = -\mathbf{G}_m^{(K=\mathrm{I})} + \frac{(\sigma_t A)^{(K=\mathrm{I})}}{12}\mathrm{M} + t_{m,1}^{(K=\mathrm{I})}\mathrm{E}_1^1 + t_{m,2}^{(K=\mathrm{I})}\mathrm{E}_2^1$$

and the right-hand-side term $\mathbf{l}^{(K=\mathrm{I})}$ contains the volumetric (scattering+external) source and the upwind contribution

$$\mathbf{l}^{(K=\mathrm{I})} = t_{m,3}^{(K=\mathrm{I})}\mathrm{E}_3^1\mathbf{\Psi}_{m'}^{(K=\mathrm{I})} + \sum_{n=0}^{N_a}\sum_{k=-n}^{n}\frac{2n+1}{4\pi}Y_{n,k}(\vec{\Omega}_m)\frac{\mathrm{M}}{12}\left[A(\sigma_{s,n}\mathbf{\Phi}_{n,k} + \mathbf{Q}_{n,k})\right]^{(K=\mathrm{I})}$$

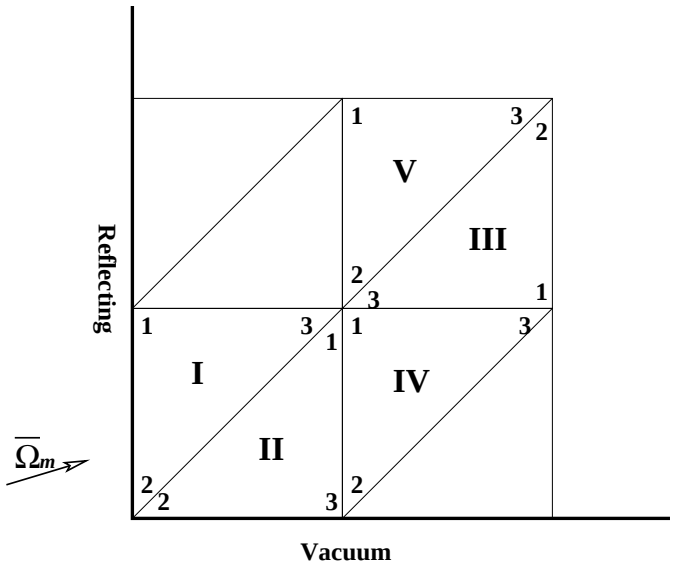


Fig. II-4. Sample transport domain.

Note that for the reflecting boundary, we use E^{1} instead of E^{1C} because we obtain the angular flux in the reflecting outgoing direction m' on the element itself.

The local transport system for element II is:

$$\mathbf{A}^{(K=\mathrm{II})} = -\mathbf{G}_m^{(K=\mathrm{II})} + \frac{(\sigma_t A)^{(K=\mathrm{II})}}{12}\mathrm{M} + t_{m,2}^{(K=\mathrm{II})}\mathrm{E}_2^1$$

$$\mathbf{l}^{(K=\mathrm{II})} = t_{m,3}^{(K=\mathrm{II})}\mathrm{E}_{3,1}^{1C}\mathbf{\Psi}_m^{(K=\mathrm{I})} + \sum_{n=0}^{N_a}\sum_{k=-n}^{n}\frac{2n+1}{4\pi}Y_{n,k}(\vec{\Omega}_m)\frac{\mathrm{M}}{12}\left[A(\sigma_{s,n}\mathbf{\Phi}_{n,k} + \mathbf{Q}_{n,k})\right]^{(K=\mathrm{II})}$$

Note that edge 2 is the upwind vacuum boundary, its contribution on the right hand side is zero. The local system of element III is:

$$\mathbf{A}^{(K=\mathrm{III})} = -\mathbf{G}_m^{(K=\mathrm{III})} + \frac{(\sigma_t A)^{(K=\mathrm{III})}}{12}\mathrm{M} + t_{m,3}^{(K=\mathrm{III})}\mathrm{E}_3^1$$

$$\mathbf{l}^{(K=\mathrm{III})} = t_{m,2}^{(K=\mathrm{III})}\mathrm{E}_{2,2}^{1C}\mathbf{\Psi}_m^{(K=\mathrm{IV})} + t_{m,1}^{(K=\mathrm{III})}\mathrm{E}_{1,1}^{1C}\mathbf{\Psi}_m^{(K=\mathrm{V})} + \sum_{n=0}^{N_a}\sum_{k=-n}^{n}\frac{2n+1}{4\pi}Y_{n,k}(\vec{\Omega}_m)\frac{\mathrm{M}}{12}\left[A(\sigma_{s,n}\mathbf{\Phi}_{n,k} + \mathbf{Q}_{n,k})\right]^{(K=\mathrm{III})}$$

Note that there are two upwind elements which are all inside the domain.

The local system matrices are simply assembled from templates of reference matrices (no numerical integration is used). This local system is of size $N(p) = \frac{(p+1)(p+2)}{2}$ and is always invertible. We have employed a simple Gaussian Elimination algorithm for the element solve, customizing it for each polynomial order by unrolling explicitly the loops. As we have proved in Section B of Appendix B,

$$\mathbf{G}_m + \mathbf{G}_m^T = \sum_{i=1}^{3} t_{m,i} \mathrm{E}_i^1 \tag{2.63}$$

C. Solution Procedure

1. Transport Sweeps: A Matrix-Free Inversion of the Streaming+Total Collision Operator

For 2-D triangular meshes, there are no dependencies in the graph of the sweep. So ,if we appropriately order all the tasks for all directions, the global matrix $\mathbf{L}$ resulting from the bilinear form $b(\Psi, \Psi^*)$ of Eq. (2.22) has a block lower-triangular structure. Such orderings of all elements are obviously direction-dependent. A simple ordering algorithm will be presented in Chapter V. As a result, the global matrix $\mathbf{L}$ is never formed for the transport calculations. But instead a proper sweeping through the elements K is prescribed so as to only invert the local systems for each element, knowing the inflow values from the upwind neighbors. In that sense, the *transport sweeps* are characterized as being performed in a *matrix-free* fashion.

Matrix-free schemes are preferable because:

- Today's computational bottleneck for neutron transport is fetching data from memory storage. Although matrices resulting from DGFEM are sparse, they are still significantly larger than the solution vectors.

- The iterative solver does not need the matrix explicitly, but only requires the

action of a matrix-vector product.

- A matrix-vector operation can be split into local operations on a single element. With properly designed higher-order shape functions, flops of local operations can be reduced further by investigating the structure of the local matrices as we will see in next sub-section.

- Matrix-free schemes favor the higher-order calculations as we explained in the introduction part.

The matrix-free scheme for the DGFEM diffusion problem and its implementation will be addressed in the next two chapters.

More generally, cycles can be present for 2-D quadrilateral and 3-D hexahedral and tetrahedral meshes (regardless of boundary conditions). In these cases, the **L** matrix is not strictly lower-triangular but the solution algorithms usually retain the sweeping process by splitting the matrix into a lower-triangular part and a strictly upper triangular part. We will discuss this in details in Sec. 3.

2. Source Iteration (SI)

In the preceding section, we avoided discussing the fact that the unknowns are present on both the left-hand-side (as angular fluxes) and on the right-hand-side (as flux moments). This section addresses this question and presents the popular Source Iteration method employed to iteratively solve the discretized transport equations.

With an appropriate numbering of the angular flux and flux moment unknowns into the $\mathbf{\Psi}$ solution vector for the angular fluxes and $\mathbf{\Phi}$ for the flux moments, we can

represent the discretized transport equation in the following matrix form:

$$\begin{aligned} \mathbf{L}\boldsymbol{\Psi} - \mathbf{M}\boldsymbol{\Sigma}\boldsymbol{\Phi} &= \mathbf{Q} \\ \boldsymbol{\Phi} &= \mathbf{D}\boldsymbol{\Psi} \end{aligned} \tag{2.64}$$

where $\mathbf{L}$ is the transport matrix from the streaming and the total collision term, $\boldsymbol{\Sigma}$ is the scattering mass matrix which operates on the flux moments, $\mathbf{M}$ is the moment-to-direction matrix, $\mathbf{D}$ is the direction-to-moment matrix. Here we use the directional external source $\mathbf{Q}$ for a more general expression rather than using the external source moments. It needs to be pointed that the spatial mass matrix associated with the scattering cross section can be applied on the flux moments before the evaluation of the directional scattering source. The dimensions of $\mathbf{L}$ are the number of spatial degrees of freedom times the number of directions, $(N_{dof} \times M)^2$; in a discontinuous method, the number of degrees of freedom is simply the number of elements N_{el} times the number of unknowns per element, i.e., $N_{dof} = N_{el} \times N(p)$, assuming a uniform polynomial order is employed. The dimensions of $\boldsymbol{\Sigma}$ are $(N_{dof} \times N_{mom})^2$, where N_{mom} is the number of moments employed in the spherical harmonics expansion of the angular fluxes; N_{mom} is equal to $\frac{(N_a+1)(N_a+2)}{2}$ in 2-D and equal to $(N_a+1)^2$ in 3-D for standard angular quadratures (as opposed to hybrid Galerkin quadratures for instance). $\mathbf{D}$ is of dimension $(N_{dof} \times N_{mom}) \times (N_{dof} \times M)$; $\mathbf{M}$ is of dimension $(N_{dof} \times M) \times (N_{dof} \times N_{mom})$. Usually, the number of flux moments is smaller than number of streaming directions M, and we can recast the problem in terms of flux moments:

$$\left[\mathbf{I} - \mathbf{D}\mathbf{L}^{-1}\mathbf{M}\boldsymbol{\Sigma}\right]\boldsymbol{\Phi} = \mathbf{D}\mathbf{L}^{-1}\mathbf{Q} \tag{2.65}$$

$\mathbf{L}$ can be inverted using a transport sweeps, direction by direction. A complete transport sweep requires $N_{el} \times M$ local solves.

One simple technique to invert Eq. (2.65) is the scattering *Source Iteration* (SI),

also known as *Richardson iteration*, where, given a previous guess for the flux moments, a complete transport sweep is performed to obtain a new angular flux, and thus new flux moments as follows:

$$\boldsymbol{\Psi}^{(\ell+1)} = \mathbf{L}^{-1}\mathbf{M}\left[\boldsymbol{\Sigma}\boldsymbol{\Phi}^{(\ell)} + \mathbf{Q}\right] \tag{2.66}$$

$$\boldsymbol{\Phi}^{(\ell+1)} = \mathbf{D}\boldsymbol{\Psi}^{(\ell+1)} \tag{2.67}$$

SI is guaranteed to converge (with some additional precautions in the case of highly forward peaked scattering) but can be excessively slow for highly diffusive media. SI can be accelerated (Preconditioned Richardson iteration), through a DSA scheme for instance, which will be presented in Chapter III. SI is terminated when the following criterion is satisfied:

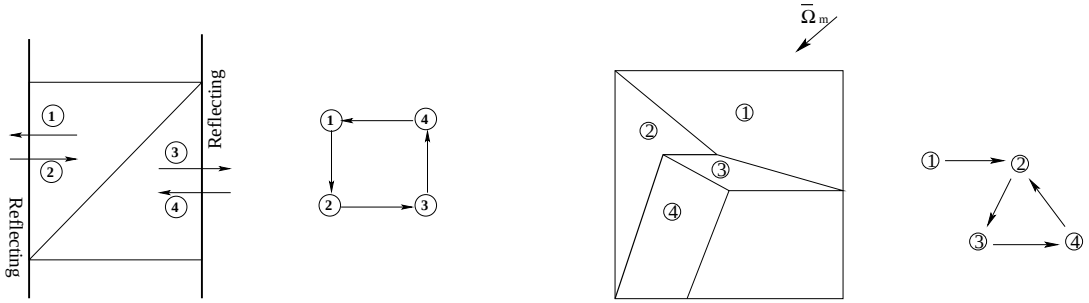
$$\frac{\left\|\Phi^{(\ell+1)} - \Phi^{(\ell)}\right\|}{\left\|\Phi^{(\ell+1)}\right\|} \leq \text{tol}_{source} \tag{2.68}$$

Each source iteration contains one transport sweep. During a transport sweep, we calculate the local source, solve the local system and update the local flux moments. No global matrices are assembled explicitly. Because the total number of flux moments is usually much smaller than the number of sweeping directions, we discard the angular fluxes which are no longer needed after each local solve, and only keep angular fluxes present in the sweeping front (to compute the upwind contributions). By doing this, significant memory savings are achieved.

3. Significant Angular Fluxes

The previous section alluded to the cases when cycles can be present in a transport sweep. A simple example of cycles is given in Fig. II-5a, where opposing edges of the domain are both reflecting edges.

Even though there are no cycles for tirangular 2-D geometries, cycles can be ob-



(a) Reflecting boundaries (b) Quadrilateral mesh

Fig. II-5. Cycles for the transport sweep.

tained in 2-D when employing quadrilaterals if their shapes are not properly designed, as shown in Fig. II-5b.

The presence of cycles in the transport sweeps will result in matrix $\mathbf{L}$ no longer being block lower-triangular. In order to retain the sweeping process, we first split $\mathbf{L}$ matrix as follows

$$\mathbf{L} = \underline{\mathbf{L}} - \overline{\mathbf{L}} \tag{2.69}$$

where $\underline{\mathbf{L}}$ is block lower-triangular and $\overline{\mathbf{L}}$ is strictly block upper-triangular (i.e., with all diagonal block matrices being zero.) We then select all angular fluxes which have at least one non-zero element in the corresponding columns of $\overline{\mathbf{L}}$ and denote them as Significant Angular Fluxes, i.e., angular fluxes that must be kept and stored in between cycles. Extracting the Significant Angular Fluxes (SAF) can be written as

$$\mathbf{\Psi_{SAF}} = \mathbf{N\Psi} \tag{2.70}$$

If the length of the SAF vector is $N_{\mathbf{SAF}}$, the dimension of matrix $\mathbf{N}$ is $N_{\mathbf{SAF}} \times (N_{dof} \times M)$. Note that $N_{\mathbf{SAF}}$ is generally significantly smaller than $(N_{dof} \times M)$, provided that a decent sweep ordering algorithm is employed. $\mathbf{NN}^T$ is an identity

matrix with dimension of $N_{\mathbf{SAF}}$. Note that

$$\overline{\mathbf{L}}\boldsymbol{\Psi} = \overline{\mathbf{L}}\mathbf{N}^T\mathbf{N}\boldsymbol{\Psi} = \overline{\mathbf{L}}\mathbf{N}^T\boldsymbol{\Psi}_{\mathbf{SAF}} \tag{2.71}$$

Now we have

$$\mathbf{L}\boldsymbol{\Psi} = \underline{\mathbf{L}}\boldsymbol{\Psi} - \overline{\mathbf{L}}\boldsymbol{\Psi} = \underline{\mathbf{L}}\boldsymbol{\Psi} - \overline{\mathbf{L}}\mathbf{N}^T\boldsymbol{\Psi}_{\mathbf{SAF}} - \mathbf{M}\boldsymbol{\Sigma}\boldsymbol{\Phi} = \mathbf{Q} \tag{2.72}$$

Moving the SAF components to right-hand-side yields

$$\begin{aligned} \boldsymbol{\Phi} &= \mathbf{D}\underline{\mathbf{L}}^{-1}\left[\overline{\mathbf{L}}\mathbf{N}^T\boldsymbol{\Psi}_{\mathbf{SAF}} + \mathbf{M}\boldsymbol{\Sigma}\boldsymbol{\Phi} + \mathbf{Q}\right] \\ \boldsymbol{\Psi}_{\mathbf{SAF}} &= \mathbf{N}\underline{\mathbf{L}}^{-1}\left[\overline{\mathbf{L}}\mathbf{N}^T\boldsymbol{\Psi}_{\mathbf{SAF}} + \mathbf{M}\boldsymbol{\Sigma}\boldsymbol{\Phi} + \mathbf{Q}\right] \end{aligned} \tag{2.73}$$

We obtain a linear system in which the unknowns are the flux moments and the SAF.

$$\begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Psi}_{\mathbf{SAF}} \end{bmatrix} = \begin{bmatrix} \mathbf{D}\underline{\mathbf{L}}^{-1}\mathbf{M}\boldsymbol{\Sigma} & \mathbf{D}\underline{\mathbf{L}}^{-1}\overline{\mathbf{L}}\mathbf{N}^T \\ \mathbf{N}\underline{\mathbf{L}}^{-1}\mathbf{M}\boldsymbol{\Sigma} & \mathbf{N}\underline{\mathbf{L}}^{-1}\overline{\mathbf{L}}\mathbf{N}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Psi}_{\mathbf{SAF}} \end{bmatrix} + \begin{bmatrix} \mathbf{D} \\ \mathbf{N} \end{bmatrix} \underline{\mathbf{L}}^{-1}\mathbf{Q} \tag{2.74}$$

or

$$\begin{bmatrix} \mathbf{I} - \mathbf{D}\underline{\mathbf{L}}^{-1}\mathbf{M}\boldsymbol{\Sigma} & -\mathbf{D}\underline{\mathbf{L}}^{-1}\overline{\mathbf{L}}\mathbf{N}^T \\ -\mathbf{N}\underline{\mathbf{L}}^{-1}\mathbf{M}\boldsymbol{\Sigma} & \mathbf{I} - \mathbf{N}\underline{\mathbf{L}}^{-1}\overline{\mathbf{L}}\mathbf{N}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Psi}_{\mathbf{SAF}} \end{bmatrix} = \begin{bmatrix} \mathbf{D} \\ \mathbf{N} \end{bmatrix} \underline{\mathbf{L}}^{-1}\mathbf{Q} \tag{2.75}$$

Now one transport sweep will update not only the flux moments, but also the SAF. The SI process is:

$$\begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Psi}_{\mathbf{SAF}} \end{bmatrix}^{(\ell+1)} = \begin{bmatrix} \mathbf{D}\underline{\mathbf{L}}^{-1}\mathbf{M}\boldsymbol{\Sigma} & \mathbf{D}\underline{\mathbf{L}}^{-1}\overline{\mathbf{L}}\mathbf{N}^T \\ \mathbf{N}\underline{\mathbf{L}}^{-1}\mathbf{M}\boldsymbol{\Sigma} & \mathbf{N}\underline{\mathbf{L}}^{-1}\overline{\mathbf{L}}\mathbf{N}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Psi}_{\mathbf{SAF}} \end{bmatrix}^{(\ell)} + \begin{bmatrix} \mathbf{D} \\ \mathbf{N} \end{bmatrix} \underline{\mathbf{L}}^{-1}\mathbf{Q} \tag{2.76}$$

So, simply by appending the SAF to the flux moment unknowns, we can still employ SI with the efficient matrix-free transport sweeps. We still use Eq. (2.68) to determine the convergence.

We can always break the proper sweeping order and artificially start the transport sweep on some tasks. Then, the angular fluxes associated with these tasks will be

labeled as significant. This situation happens when we parallelize the transport sweeps using domain decomposition with MPI and always start the transport sweeps on the sub-domain interfaces as illustrated in Fig. II-6 (synchronous start or Parallel Block Jacobi method).

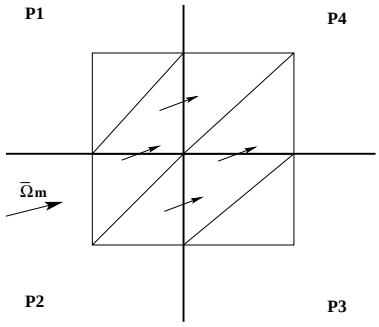


Fig. II-6. Domain decomposition with synchronous communication.

Finally, note that we can merge Eq. (2.65) and Eq. (2.75) to obtain a simpler and more general expression:

$$(\mathbf{I} - \mathbf{T})\mathbf{x} = \mathbf{b} \tag{2.77}$$

If there are no SAF, then

$$\mathbf{T} = \mathbf{D}\mathbf{L}^{-1}\mathbf{M}\boldsymbol{\Sigma} \tag{2.78}$$

$$\mathbf{x} = \boldsymbol{\Phi} \tag{2.79}$$

$$\mathbf{b} = \mathbf{D}\mathbf{L}^{-1}\mathbf{Q} \tag{2.80}$$

and we recover the simpler SI procedure. Otherwise we have:

$$\mathbf{T} = \begin{bmatrix} \mathbf{D}\underline{\mathbf{L}}^{-1}\mathbf{M}\boldsymbol{\Sigma} & \mathbf{D}\underline{\mathbf{L}}^{-1}\overline{\mathbf{L}}\mathbf{N}^T \\ \mathbf{N}\underline{\mathbf{L}}^{-1}\mathbf{M}\boldsymbol{\Sigma} & \mathbf{N}\underline{\mathbf{L}}^{-1}\overline{\mathbf{L}}\mathbf{N}^T \end{bmatrix} \tag{2.81}$$

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Psi}_{\mathbf{SAF}} \end{bmatrix} \tag{2.82}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{D} \\ \mathbf{N} \end{bmatrix} \underline{\mathbf{L}}^{-1}\mathbf{Q} \tag{2.83}$$

We point out that this splitting may increase the number of iterations [31, 32].

4. GMRes Solver

SI (or Richardson iteration) is not the only technique available to solve the global linear system formed by the discretized S_N transport equations. Since matrix $(\mathbf{I}-\mathbf{T})$ is not symmetric, we naturally have recourse to a GMRes technique [94]. With freely available quality open-source GMRes packages employing reverse communication (i.e., without the need to provide the global matrix, but just its action on a Krylov vector) [95], we can solve the linear system arising from the discretization of the linear transport equation by providing the following four operations:

1. Construct the right-hand-side $\mathbf{b}$:

   We need one transport sweep to construct the right-hand-side $\mathbf{b}$ with Eq. (2.80) or Eq. (2.83). This right-hand-side has the physical significance of the uncollided flux due to the presence of an external volumetric source or incident radiation (Dirichlet boundary conditions).

2. Provide the action of matrix $(\mathbf{I}-\mathbf{T})$ on a vector:

This matrix, without SAF, is $\mathbf{I} - \mathbf{D}\mathbf{L}^{-1}\mathbf{M}\boldsymbol{\Sigma}$. With SAF, it is

$$\begin{bmatrix} \mathbf{I} - \mathbf{D}\underline{\mathbf{L}}^{-1}\mathbf{M}\boldsymbol{\Sigma} & -\mathbf{D}\underline{\mathbf{L}}^{-1}\overline{\mathbf{L}}\mathbf{N}^T \\ -\mathbf{N}\underline{\mathbf{L}}^{-1}\mathbf{M}\boldsymbol{\Sigma} & \mathbf{I} - \mathbf{N}\underline{\mathbf{L}}^{-1}\overline{\mathbf{L}}\mathbf{N}^T \end{bmatrix}.$$

The matrix-vector product is simply equal to original vector of flux moments and SAF supplied, minus the updated flux moments and SAF after one transport sweep.

3. Determine convergence:

   The convergence criterion used in the GMRes solver is such that the norm of the normalized residual is smaller than a prescribed tolerance:

$$\frac{\|\mathbf{b} - [\mathbf{I} - \mathbf{T}]\,\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \leq \mathrm{tol}_{source} \tag{2.84}$$

4. Provide a preconditioner:

   This step is optional. We will see in Chapter III that DSA can be employed as a preconditioner in the context of GMRes, see also [96].

Details on the GMRes algorithm will not be discussed here but can be found in [96]. We have employed the GMRes solver developed by CERFACS, which can be obtained at: `http://www.cerfacs.fr/algor/`.

D. Numerical Results

We propose four test cases: two source-driven problem and two eigenproblems. The angular quadrature used is the S_8 level symmetric quadrature from the NEWT code of the SCALE package [72], unless otherwise noted.

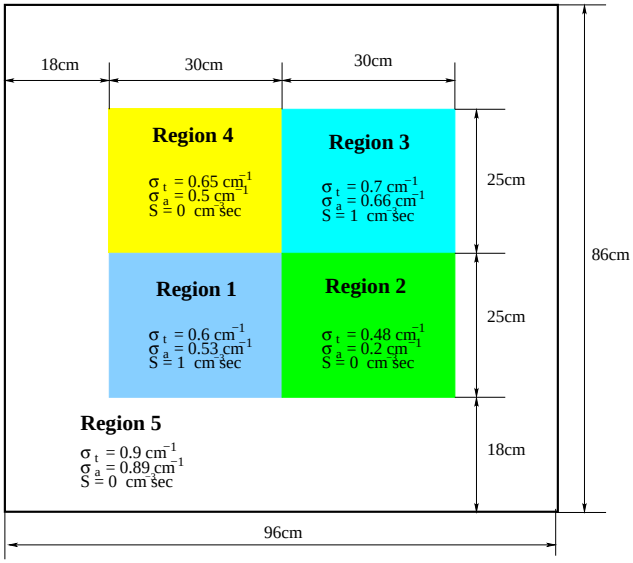
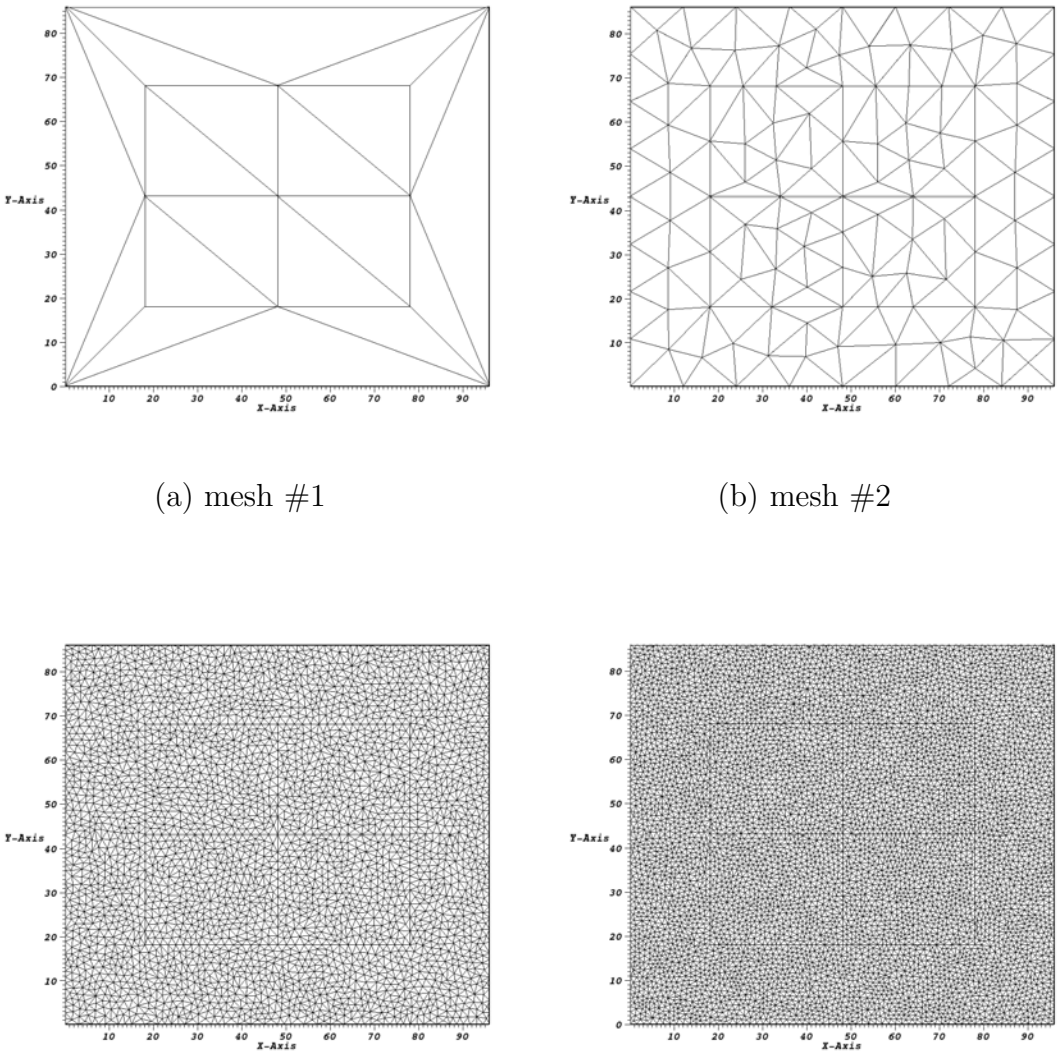


Fig. II-7. Geometry of the IAEA-EIR-2 benchmark.

1. One-Group Source Problem: The EIR-2 Benchmark

The definition (geometry and cross section data) of this problem can be found in several publications [97] [98]; it is commonly referred to as the "Stepanek problem" or the IAEA-EIR-2 problem. It consists of two source regions (regions #1 and 3) and two absorbing regions (regions #2 and 4). These four regions are arranged in a checkerboard fashion and are surrounded by region #5, see Fig. II-7. Both the external source and the scattering cross section are isotropic. Vacuum boundary conditions are applied. Various unstructured meshes were obtained using the Triangle mesh generator by employing different maximum element area constraints.

A total of seven meshes were created, with the following maximum triangle areas, in cm^2: 60, 30, 15, 5, 2, and 1, resulting in meshes having 20, 206, 431, 869, 2553, 6441, and 12893 elements, respectively. Fig. II-8 shows meshes #1, 2, 6, 7. Fig. II-9 presents the convergence rate for the reaction rate in region #3, as a function of the number of (scalar flux) unknowns. For unstructured meshes, plotting the convergence rate as a function of mesh size is delicate because the element mesh size is not uniform. Nonetheless, in this case, it is common practice to graph convergence rates as a function of the total number of elements in a mesh. Since the total number of elements in a mesh, N_{elem}, is proportional to the square of the typical element size, or $h \propto \sqrt{N_{el}}$, the slopes inferred from the loglog graphs simply have to be multiplied by 2 to retrieve the convergence rates as a function of the mesh size. Another, yet similar, option consists in graphing the convergence rates as a function of the number of unknowns, N_{dof}. In DGFEM, there is a simple relation between N_{dof} and N_{el} when the homogeneous polynomial order is employed: namely, for triangular meshes with basis functions of order p, we have $N_{dof} = \frac{(p+1)(p+2)}{2} N_{el}$; we have chosen this latter plotting convention because (i) the estimation of the convergence rates is not affected and (ii) such a convention accounts for the different memory requirements incurred by different polynomial orders. We can note that the various polynomial approximations in DGFEM converge at different rates; Table II-I provides the convergence rates seen on Fig. II-9, as well as the convergence rates in terms of mesh size. These values are in good agreement with the theoretical expected rates of $p + 1$ (for smooth enough solutions). For the same mesh, we also observe that the proportionality constant in the error term decreases as the order p is increased, and that these gains are diminishing as p increases. Clearly, DGFEM(1) has the poorest accuracy than any other methods. Fig. II-10 provides the loglog plot of the error versus CPU time, for region #2. A linear approximation requires about 10 times longer than a quadratic approximation



(a) mesh #1

(b) mesh #2

(c) mesh #3

(d) mesh #4

Fig. II-8. EIR-2 benchmark meshes.

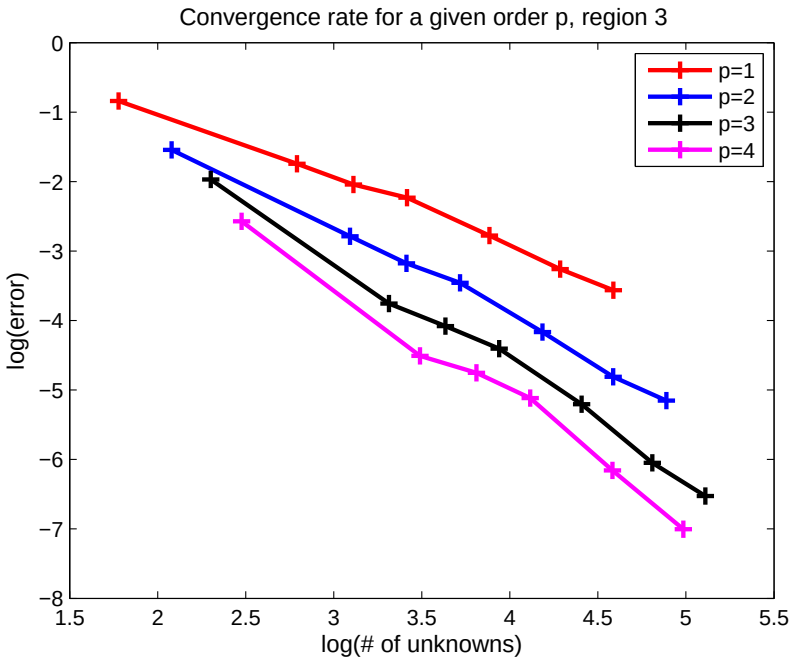


Fig. II-9. EIR-2 benchmark: Convergence rates for the average flux in region #3 as a function of the number of unknowns, approximations DGFEM(1) through DGFEM(4).

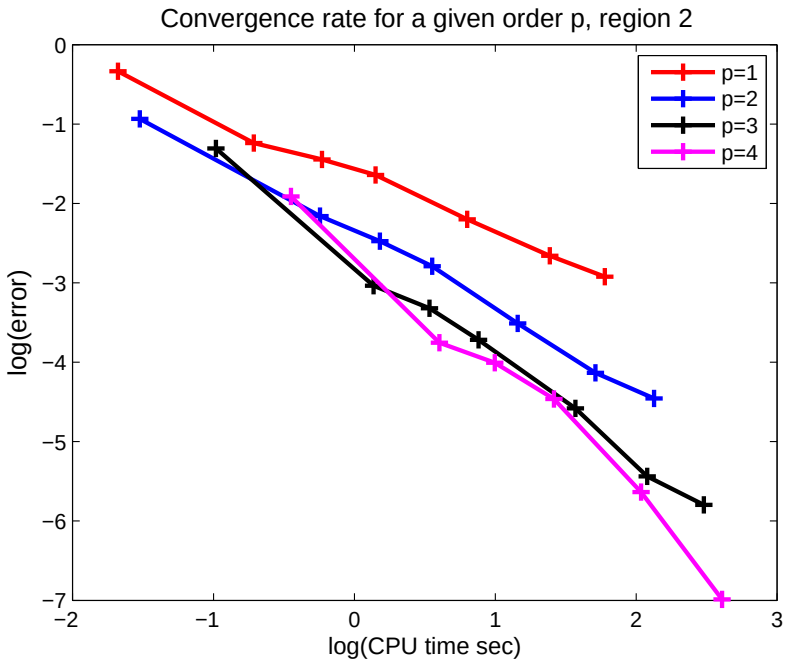


Fig. II-10. EIR-2 benchmark: Convergence rates for the average flux in region #2 as a function of CPU time, approximations DGFEM(1) through DGFEM(4).

Table II-I. EIR-2 benchmark: Convergence rates.

p	convergence of rate in N	convergence rate in h
1	1.12	2.24
2	1.47	2.94
3	1.84	3.68
4	2.14	4.28

in order to reach an error below 1%. The approximation $p = 3$ provides also some benefits over the quadratic approximation for higher accuracy. Finally, using $p = 4$ does not provide much CPU gain versus $p = 3$ for a given accuracy.

2. Four-Group Eigenproblem: The KNK Fast Reactor Benchmark

This benchmark problem, documented by Takeda [99], is a model of the KNK-II fast reactor core. The geometry, an hexagonal lattice, is given in Fig. II-11. The 4-group cross-sections can be found in [99] and are also listed in Table II-II.

The 2-D version of this problem has been recently used by [100] and [101] for nodal S_N methods; we present here the results related to the rodded case. The domain is meshed by subdividing every hexagon into six triangles, resulting in the coarsest mesh utilized here. Finer meshes were obtained by regularly subdividing every triangle into four smaller triangles. Four refinements in the initial mesh were used, thus, each hexagon was treated with either 6, 24, 96, 384, 1536 triangles. Table II-III presents the error, provided in pcm (per cent mille 10^{-5}), in the k_{eff} eigenvalue for various polynomial orders and uniform refinement levels. The reference k_{eff} was obtained with $p = 4$ and 1536 triangles per hexagon.

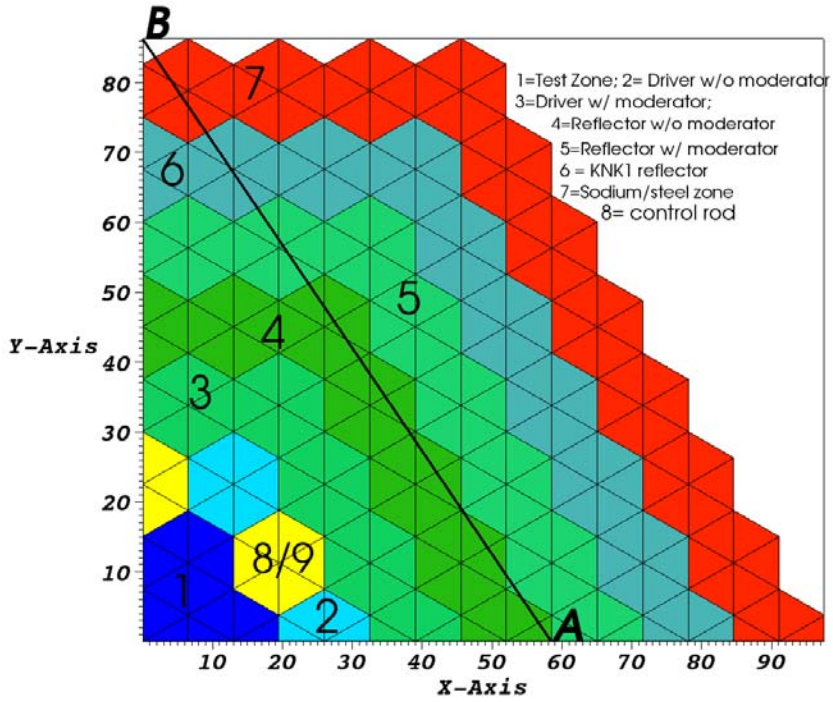


Fig. II-11. Geometry of the Takeda benchmark and initial triangular mesh.

Table II-II. Takeda benchmark: Material properties.

g	$\sigma_{t,g}$	$\sigma_{s,0}^{g\to 1}$	$\sigma_{s,0}^{g\to 2}$	$\sigma_{s,0}^{g\to 3}$	$\sigma_{s,0}^{g\to 4}$	$\nu\sigma_{f,g}$	χ_g
TEST ZONE							
1	1.24526E-01	1.05964E-01	1.12738E-02	1.46192E-04	9.62178E-07	1.79043E-02	0.908564
2	2.01025E-01	0.00000E-00	1.89370E-01	3.64847E-03	1.06888E-06	1.59961E-02	0.087307
3	2.86599E-01	0.00000E-00	0.00000E-00	2.70207E-01	1.80479E-03	2.40856E-02	0.004129
4	3.68772E-01	0.00000E-00	0.00000E-00	0.00000E-00	3.18960E-01	7.33104E-02	0.000000
DRIVER WITHOUT MODERATOR							
1	1.40226E-01	1.19887E-01	1.30790E-02	1.59938E-04	1.07166E-06	1.59878E-02	0.908564
2	2.28245E-01	0.00000E-00	2.15213E-01	4.00117E-03	1.82716E-06	1.64446E-02	0.087307
3	3.25806E-01	0.00000E-00	0.00000E-00	3.06885E-01	1.67341E-03	2.71541E-02	0.004129
4	4.18327E-01	0.00000E-00	0.00000E-00	0.00000E-00	3.60906E-01	8.45807E-02	0.000000
DRIVER WITH MODERATOR							
1	1.41428E-01	1.14337E-01	2.09664E-02	1.39132E-03	6.10281E-05	1.01663E-02	0.908564
2	2.45394E-01	0.00000E-00	2.12006E-01	2.67269E-02	1.08186E-03	9.46359E-03	0.087307
3	3.98255E-01	0.00000E-00	0.00000E-00	3.52093E-01	3.29030E-02	1.87325E-02	0.004129
4	4.35990E-01	0.00000E-00	0.00000E-00	0.00000E-00	3.70872E-01	8.25335E-02	0.000000
REFLECTOR WITHOUT MODERATOR							
1	1.59346E-01	1.47969E-01	1.06607E-02	2.49956E-04	1.82565E-06	-	-
2	2.16355E-01	0.00000E-00	2.10410E-01	5.46711E-03	1.00157E-06	-	-
3	3.48692E-01	0.00000E-00	0.00000E-00	3.42085E-01	5.36879E-03	-	-
4	6.24249E-01	0.00000E-00	0.00000E-00	0.00000E-00	6.19306E-01	-	-
REFLECTOR WITH MODERATOR							
1	1.39164E-01	1.05911E-01	2.96485E-02	3.06502E-03	1.41697E-04	-	-
2	2.46993E-01	0.00000E-00	1.84820E-01	5.91780E-02	2.69229E-03	-	-
3	4.52425E-01	0.00000E-00	0.00000E-00	3.73072E-01	7.81326E-02	-	-
4	5.36256E-01	0.00000E-00	0.00000E-00	0.00000E-00	5.12103E-01	-	-
KNK-1 REFLECTOR							
1	1.51644E-01	1.38427E-01	1.23901E-02	3.66930E-04	1.69036E-06	-	-
2	1.42382E-01	0.00000E-00	1.37502E-01	4.41927E-03	1.63280E-06	-	-
3	1.65132E-01	0.00000E-00	0.00000E-00	1.60722E-01	3.33075E-03	-	-
4	8.04845E-01	0.00000E-00	0.00000E-00	0.00000E-00	7.98932E-01	-	-
SODIUM STEEL ZONE							
1	9.65097E-02	8.83550E-02	7.73409E-03	1.94719E-04	8.89615E-07	-	-
2	9.87095E-02	0.00000E-00	9.52493E-02	3.22568E-03	7.98494E-07	-	-
3	1.34200E-01	0.00000E-00	0.00000E-00	1.30756E-01	2.90481E-03	-	-
4	4.12670E-01	0.00000E-00	0.00000E-00	0.00000E-00	4.09632E-01	-	-
CONTROL ROD							
1	1.39085E-01	1.17722E-01	1.26066E-02	1.33314E-04	1.08839E-06	-	-
2	2.28152E-01	0.00000E-00	1.94699E-01	4.32219E-03	1.85491E-07	-	-
3	3.18806E-01	0.00000E-00	0.00000E-00	2.44352E-01	3.68781E-04	-	-
4	6.27366E-01	0.00000E-00	0.00000E-00	0.00000E-00	3.14816E-01	-	-
CONTROL ROD FOLLOWER = STEEL							
1	9.83638E-02	9.06050E-02	7.42377E-03	1.18163E-04	8.25890E-07	-	-
2	1.35140E-01	0.00000E-00	1.30581E-01	4.35250E-03	3.41675E-07	-	-
3	2.24749E-01	0.00000E-00	0.00000E-00	2.19547E-01	4.64594E-03	-	-
4	2.83117E-01	0.00000E-00	0.00000E-00	0.00000E-00	2.80707E-01	-	-

Table II-III. Takeda benchmark: Error in the k_{eff}, in pcm, approximations DGFEM(1) through DGFEM(4), initial mesh and four uniformly refined meshes.

p	6 triangles	24 triangles	96 triangles	384 triangles	1536 triangles
1	355.161	68.200	12.620	2.166	0.342
2	11.534	2.489	0.364	0.042	0.004
3	2.534	0.336	0.037	0.004	0.0004
4	0.602	0.070	0.010	0.001	ref. $k_{\mathit{eff}} = 1.0108202$

Again, the convergence of the linear DGFEM seems relatively poor compared to higher-order approximations; it takes about four levels of uniform refinement for the linear approximation to yield the same accuracy of the fourth order method on the initial mesh or the cubic method on the once-refined mesh. This is also illustrated in Fig. II-12, where the convergence in k_{eff} is plotted as a function of the CPU time. In Fig. II-12, we note again the comparable effectiveness of the cubic and fourth order method, suggesting that approximations with orders greater than 4 may not be cost-effective in terms of CPU time.

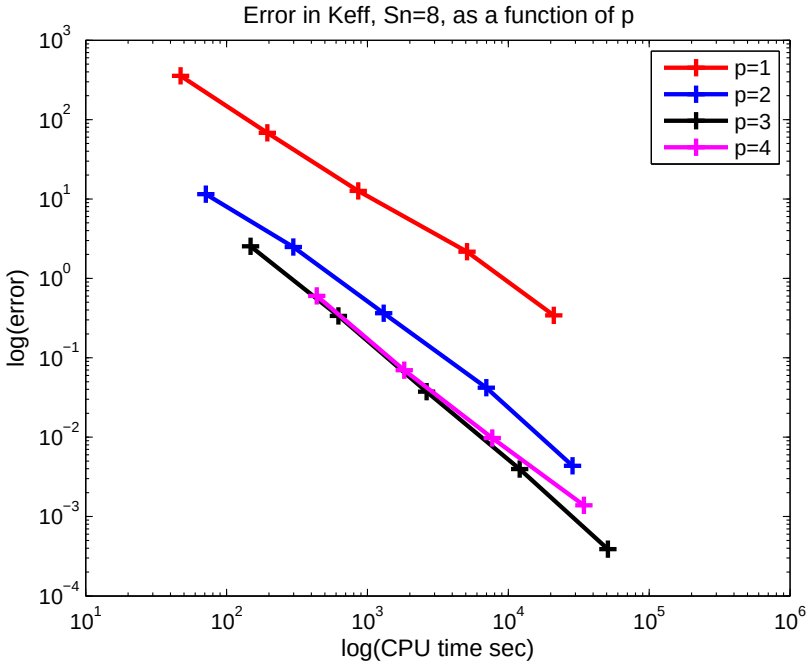


Fig. II-12. Takeda benchmark: Convergence rates in the k_{eff} as a function of CPU time, approximations DGFEM(1) through DGFEM(4).

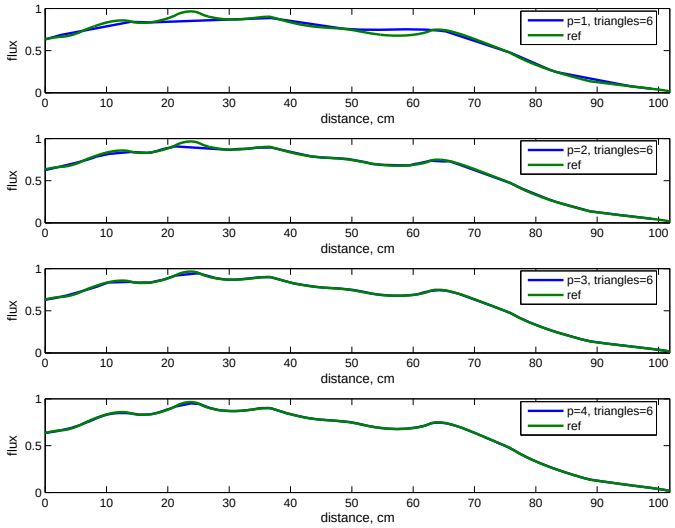
We also compared the detailed spatial solutions by graphing a 1-D cut throughout the geometry; this cut is represented by the AB line in Fig. II-11. Fig. II-13 provides the values along the AB line for the four polynomial orders as well as the first two

meshes (6 triangles/hexagon and 24 triangles/hexagon). The reference values are taken from a fourth order calculation, using 1536 triangles per hexagon. We note that on the initial mesh (6 triangles/hexagon), there is a good agreement for $p \geq 3$ and for the next once-refined mesh, a good agreement is achieved starting at $p \geq 2$, and, again, the linear solution shows the poorest results.

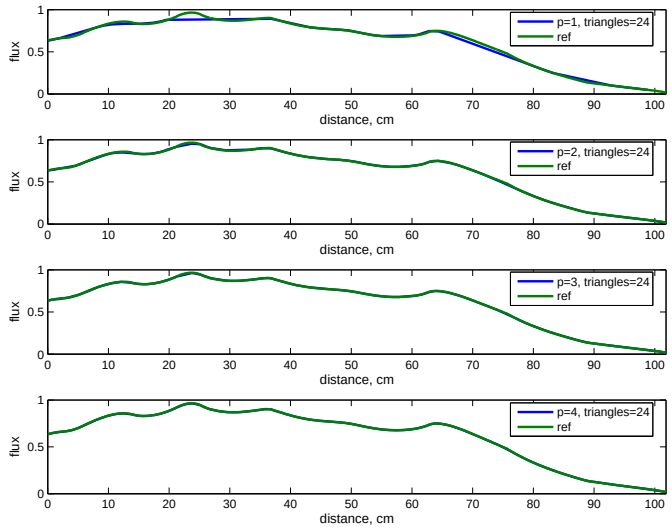
3. Seven-Group Eigen-problem: The UOX/MOX C5G7 Benchmark

This third test is the 2-D version of the UOX/MOX C5G7 benchmark, where the fuel pins are represented by cylinders (a homogenization of the fuel and its cladding) surrounded by water. The geometry consists of four $17x17$ fuel assemblies, each composed of 264 fuel pins and 25 water holes. These fuel assemblies are surrounded by a homogeneous reflector (water), resulting in a mini-core geometry of $3x3$ fuel assembly, see [102] for the complete description of the geometry and the cross section data. In our triangularization, each fuel pin is approximated by a regular dodecagon (12-sided polygon) whose side is such that the area of the fuel pin is preserved (a "conservation of fuel" principle). The resulting mesh, containing 39,633 elements, is shown in Fig. II-14. A Gauss-Chebyshev product quadrature (with 4 polar angles and 16 azimuthal angles per quadrant) is used. The k_{eff} using a linear approximation is 1.18641(47), whereas the cubic approximation yields 1.18641(90); the published reference value, obtained from a Monte Carlo calculation is 1.18655(0), [103]. Fig. II-15 provides the 1-D cross-sectional cut of the flux along from the main diagonal (from point (0,0) to point (64.26,64.26)), for groups 1 and 7 using the $p = 1$ and $p = 3$ methods. Even though the k_{eff} is well approximate using linear finite elements on this fine mesh, some discernible discrepancies in the local flux values can still be seen in Fig. II-15 for DGFEM(1) when compared to DGFEM(3).

Scalar fluxes of group 1 and group 7 are showed in Fig. II-16 with the 2-D



(a) Initial mesh.



(b) Once-refined mesh.

Fig. II-13. Takeda benchmark: Flux values along line AB for group #4, approximations DGFEM(1) through DGFEM(4).

pseudo-color plots.

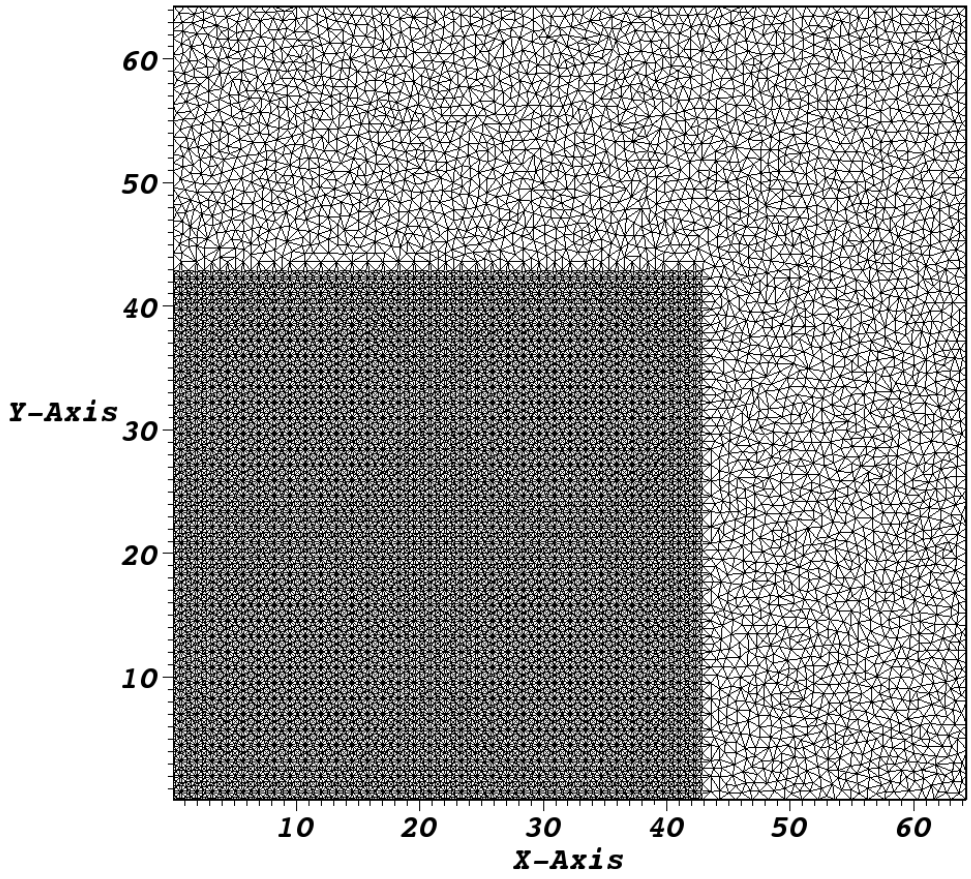
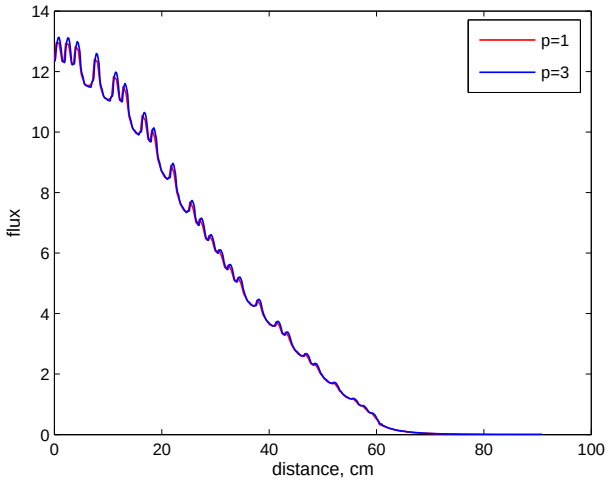
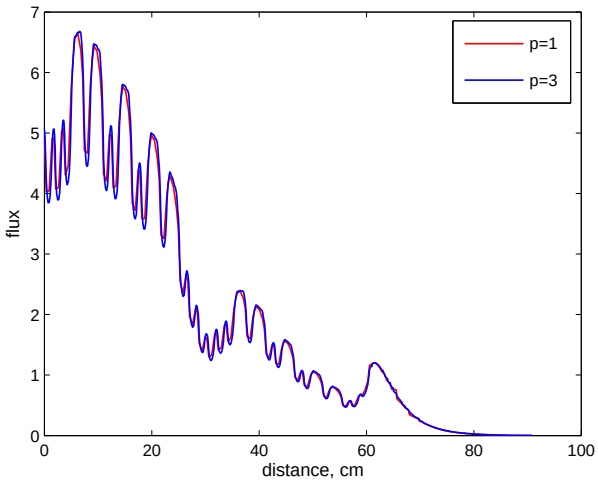


Fig. II-14. Mesh used for the C5G7 benchmark.

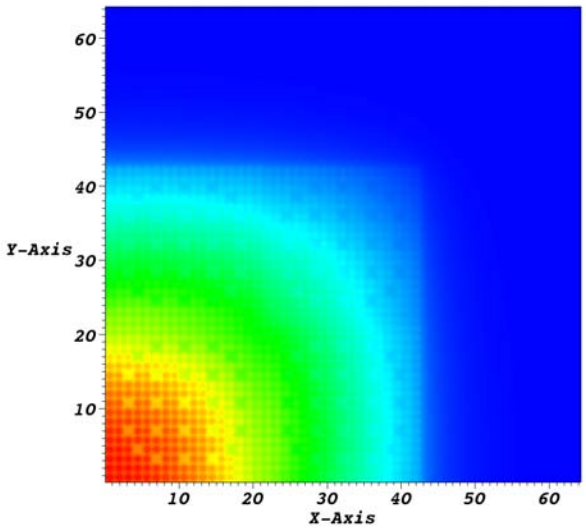


(a) Group #1

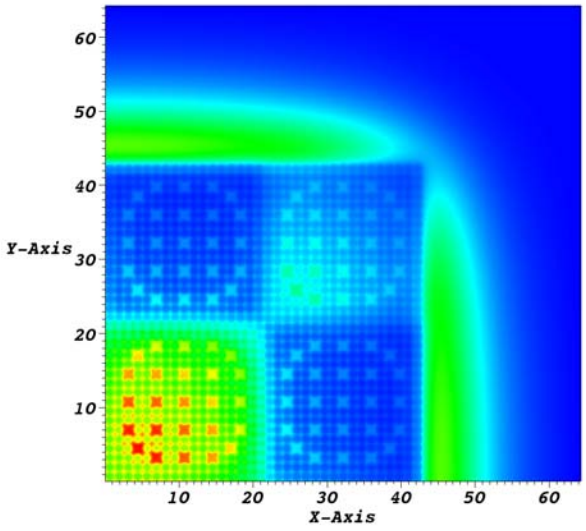


(b) Group #7

Fig. II-15. C5G7 benchmark: Flux values along the main diagonal (0,0)-(64.26,64.26), approximations DGFEM(1) and DGFEM(3).



(a) Group #1



(b) Group #7

Fig. II-16. C5G7 benchmark: Scalar fluxes.

4. Convergence Studies with a Simple Homogeneous Square Problem

Additional convergence results related to higher-order DGFEM are presented in this section. These test cases are inspired by Larsen's and Azmy's work [83, 85]. A square domain containing a homogeneous material is modeled. The total cross section, σ_t, is 1 cm^{-1}. We considered the cases of a pure absorber ($\sigma_{s,0} = 0$ cm^{-1}) and a scattering medium ($\sigma_{s,0} = 0.5$ cm^{-1}). Seven different square widths W were employed: 1, 5, 10, 20, 40, 60, and 100 cm, resulting in optical thicknesses ranging from 1 to 100 mean-free-paths (MFP). No volumetric external sources are present. Spatially uniform incident beam fluxes are applied either on the left face ($\Psi^{inc}(x = 0, y) = 1$) or on both left and bottom ($\Psi^{inc}(x = 0, y) = \Psi^{inc}(x, y = 0) = 1$) faces simultaneously. We considered two incident directions: a direction with exactly a 45-degree angle with respect to the axes and another direction with an angle of about 18.444 degrees with respect to the x-axis (an S_4 angular quadrature is used in these tests; its directions form angles of 18.444..., 45, and 71.556... degrees with respect to the x-axis in the first quadrant). The use of a 45-degree angle in a square domain makes it particularly easy to align the mesh with the singularity (incident beam direction stemming from the domain corner). The use of the other direction ($\sim$18.444... degrees) serves as a test case in which the mesh is not aligned with the incident boundary fluxes. When the flux is incident only on the left face, we should expect a more abrupt transition across the characteristic line separating the illuminated portion of the domain from the non-illuminated one (the flux is discontinuous in the case of a pure absorber). For an incoming flux, of equal value, incident on both the left and bottom faces, the flux inside the domain will present less singularities (in this case, the flux is indeed continuous inside the domain, even for a pure absorber medium).

We propose to analyze the effect of singularities in the transport solution on

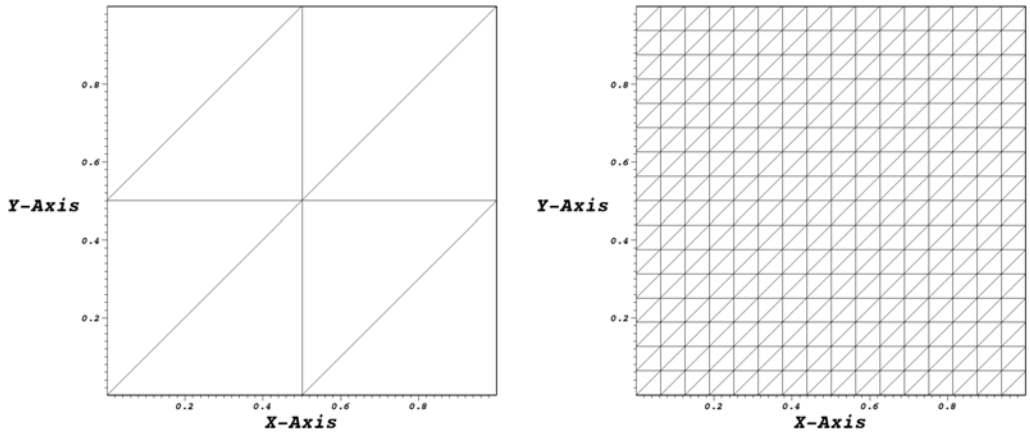
DGFEM(p) for (a) fluxes incident on one or two adjacent faces, (b) with meshes aligned or not with the incident direction, (c) for materials with or without scattering, (d) in configurations of various optical thicknesses, (e) on sequences of uniformly refined structured and unstructured meshes. Convergence results and some discussions are provided for these various cases. The initial structured and unstructured meshes are given on Fig. II-17 (left panes). After three uniform refinements, the meshes are graphed on Fig. II-17 (right panes). We have chosen to uniformly refine the original meshes, rather than re-generating new meshes with a smaller triangle area constraint. As we will see from the results, having grids with triangles of various sizes does not affect the convergence rates.

a. Flux Incident on the Left Face, Pure Absorber Case

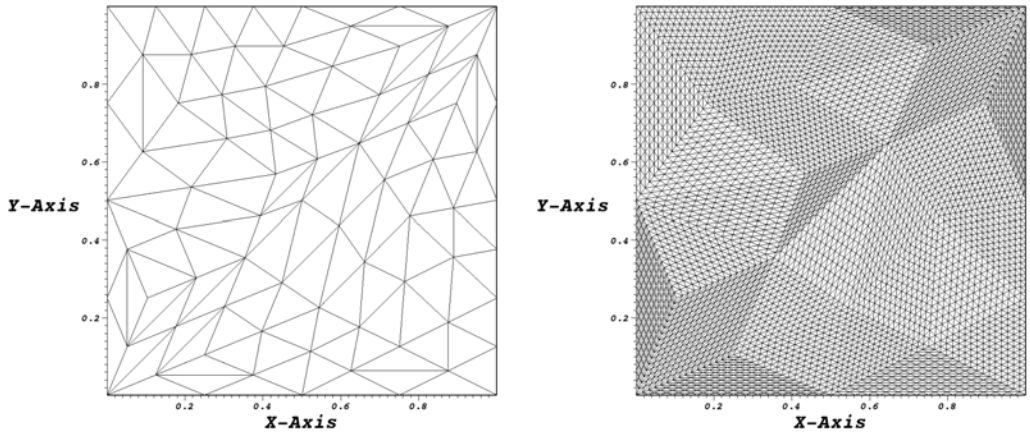
A left-incident beam is applied on the domain geometry. The transport solution in the domain is singular (with discontinuity) along the characteristic line emanating from the bottom-left corner in the direction of the incident particles. Mathematically, the transport solution is in the $H^{1/2-\varepsilon}(\mathcal{D})$ space.

For a 45-degree incident flux, the meshes employed are perfectly aligned with the discontinuity of the transport solution. The convergence rates for DGFEM(1) through DGFEM(4), measured in the L_2 norm, are plotted in Fig. II-18 (structured meshes aligned with the 45-degree singularity) and Fig. II-19 (unstructured meshes aligned with the 45-degree singularity). The plots are provided for domains of optical thicknesses of 1, 10, 40, and 100 MFP. We can clearly see that the DGFEM(p) method is converging at the theoretical rate of $p + 1$. In this case where the mesh is aligned with the singularity, the DGFEM "does not see" the lack of regularity of the solution.

We repeated these calculations but, this time, we employed an ∼18-degree inci-



(a) Structured mesh



(b) Unstructured mesh

Fig. II-17. Initial and three-time-refined meshes, (left column: initial meshes, right column: three-time-refined meshes).

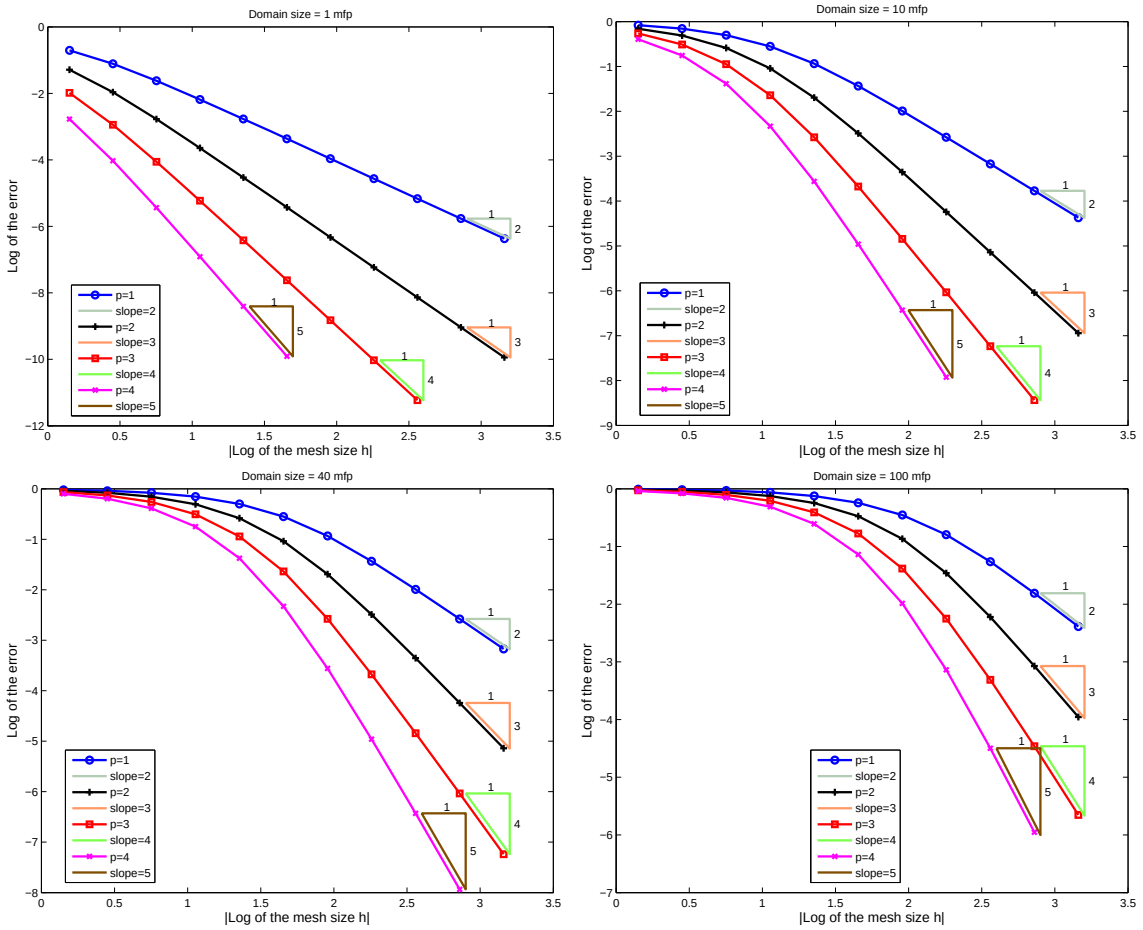


Fig. II-18. Convergence rates: pure absorber case with 45° left-face incidence, structured mesh aligned with incident beam, domain size = 1 (top left), 10 (top right), 40 (bottom left), and 100 (bottom right) MFP.

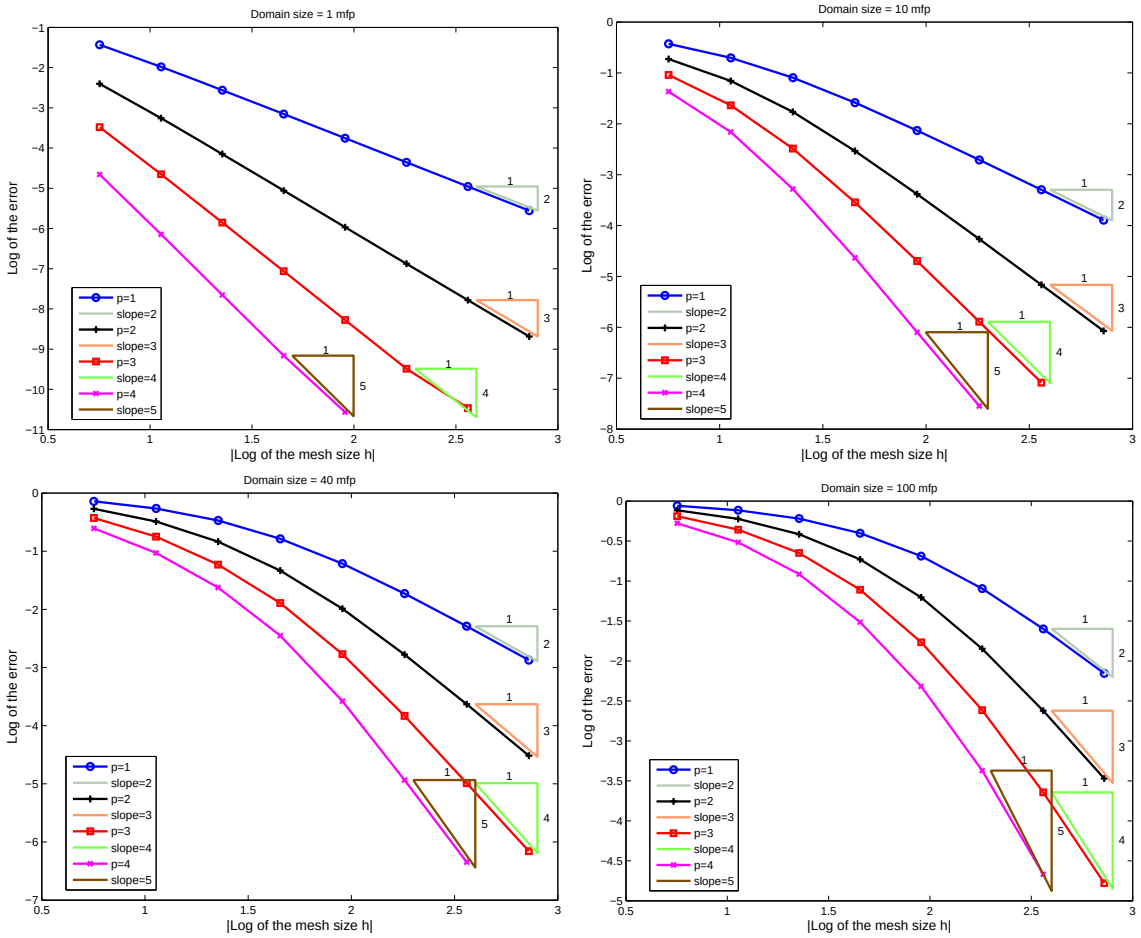


Fig. II-19. Convergence rates: pure absorber case with 45° left-face incidence, unstructured mesh aligned with incident beam, domain size = 1 (top left), 10 (top right), 40 (bottom left), and 100 (bottom right) MFP.

dent beam (i.e., the meshes are not aligned with the transport solution singularity). Fig. II-20 provides the convergence rates for the structured grids (domain thicknesses of 1, 10, 40, and 100 MFP) and Fig. II-21 gives the convergence rates for the unstructured grids (domain thicknesses of 1, 10, 40, and 100 MFP). We note that for optically thin domains (and thus optically thin meshes), the convergence rate is dictated by the regularity of the transport solution, i.e., all convergence slopes are equal to 1/2. For thicker domains, higher convergence rates are observed as long as the mesh width remains greater than the mean-free-path. In this case, we observe a pre-asymptotic region where the convergence rates tend to the higher theoretical value of $p+1$ without quite attaining that value. When mesh widths become optically thin, the regularity of the solution once again limits the convergence to a slope of 1/2.

We also note that, in any case, the error is always lower for higher polynomial orders. Thus, even if no enhanced convergence rates are observed, it may still be advantageous to use higher-order polynomials in order to have smaller error values.

b. Flux Incident on Both the Left and Bottom Faces, Pure Absorber Case

Two incident beams, of identical direction and intensity, are now applied to both the left and bottom faces. In this case, the transport solution is continuous inside the domain and belongs to the $H^{3/2-\varepsilon}(\mathcal{D})$ space. When using a 45-degree incident beam (i.e., grids are aligned with the singularity), we recover results identical to the above one-beam case (Section a), when meshes were aligned with the discontinuity. For brevity, we do not show any convergence plots for the case with alignment in this section but only state the conclusions: regardless of the domain or meshes optical thickness, DGFEM(p) converges at rate of $p+1$ measured in the standard L_2 norm, with lower absolute errors as the polynomial order is increased.

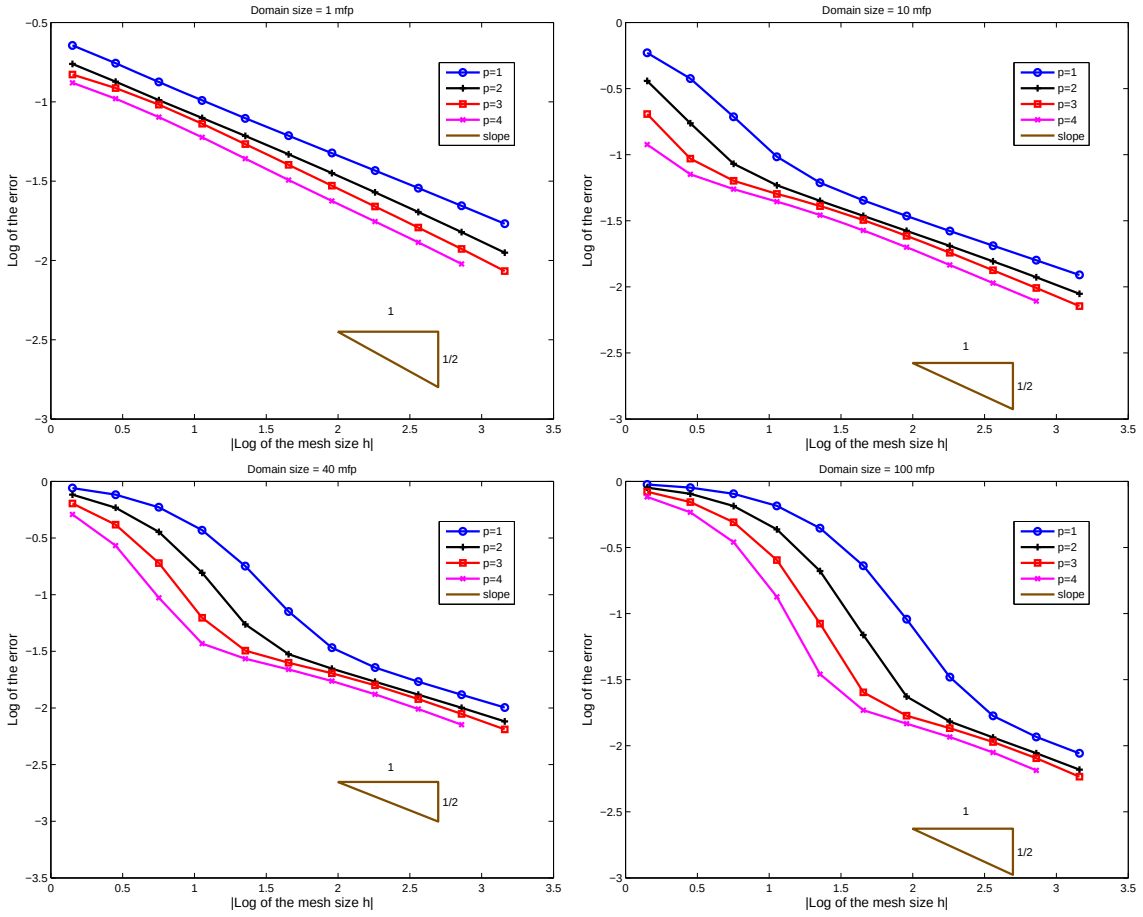


Fig. II-20. Convergence rates: pure absorber case with ∼18° left-face incidence, structured mesh not aligned with incident beam, domain size = 1 (top left), 10 (top right), 40 (bottom left), and 100 (bottom right) MFP.

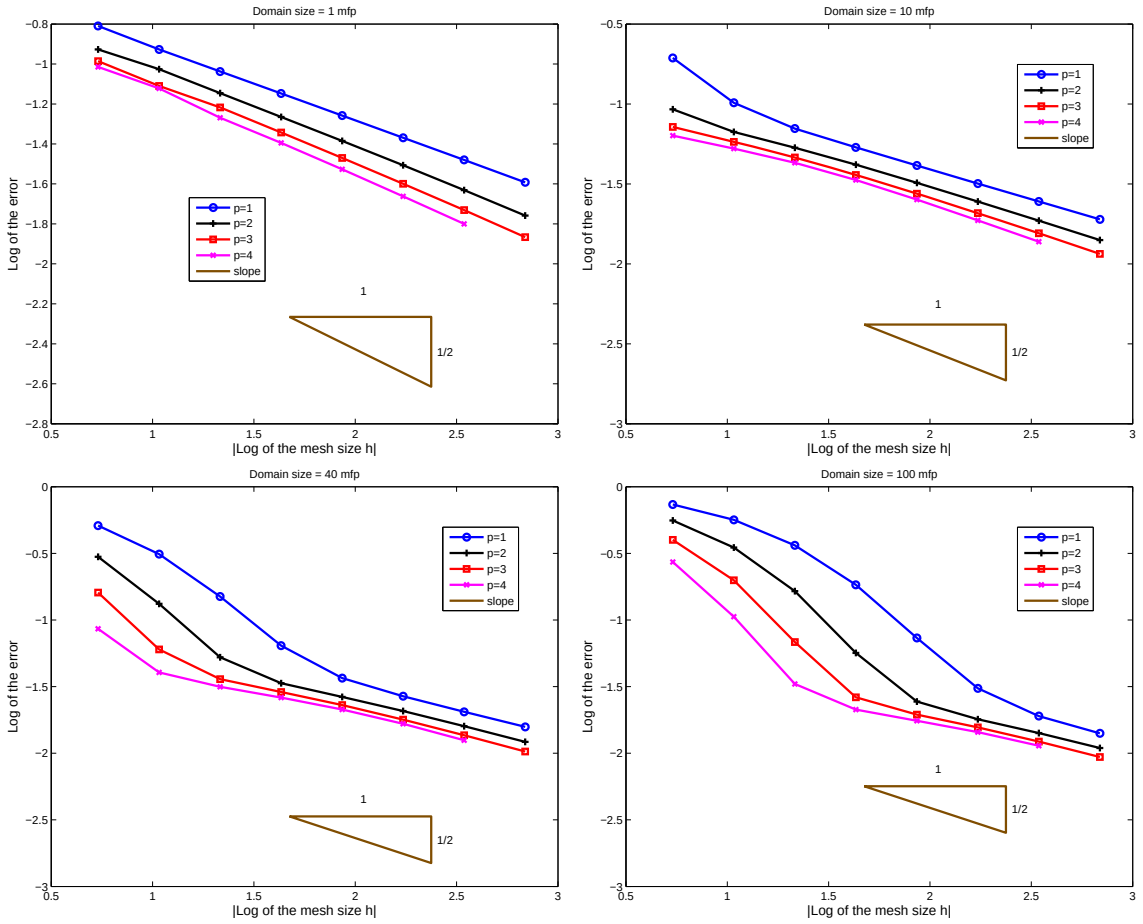


Fig. II-21. Convergence rates: pure absorber case with ∼18° left-face incidence, unstructured mesh not aligned with incident beam, domain size = 1 (top left), 10 (top right), 40 (bottom left), and 100 (bottom right) MFP.

In the case of an ∼18-degree incident beam, the meshes are no longer aligned with the singularity of the transport solution, but due to the higher regularity of the transport solution (which is now continuous), a convergence rate of 3/2 is achieved. As noted previously, for optically thick meshes (coarser meshes), the convergence rate is higher than the 3/2 value imposed by the solution regularity and is dependent upon p; for $p = 1, 2, 3$, the convergence rates observed tend to the value of $p + 1$; for $p = 4$, the regularity of the solution prevented the rate to fully reach a value of 5. As the meshes are refined, the convergence rates tend towards the asymptotic limit of 3/2, regardless of the polynomial bases used. Fig. II-22 gives the convergence rates for the structured grids (domain thicknesses of 1, 10, 40, and 100 MFP). Fig. II-23 shows the rates for the unstructured grid for a domain of 100 MFP-thick, and rates almost equal to $p + 1$ are observed for intermediate mesh sizes.

c. Flux Incident on the Left Face, Scatterer Material Case

In the case of a scatterer material, it is well known that the S_N transport solution will present singularities originating from the corners of the domain in each discrete ordinate direction [104]. It is of great practical interest to investigate how DGFEM(p) converges in this case, as many applications include scattering. We once again used the above-mentioned S_4 quadrature.

In this first series of tests, the incident beam is aligned along the 45-degree direction. When meshes are aligned with the incident particles, convergence rates obtained ranged from (i) a polynomial-order dependent value (close to but less than $p+1$) for optically thick meshes (in the pre-asymptotic range) to (ii) 3/2 for optically thin meshes. Again, it is not unexpected to fall short of the $p + 1$ rate for coarse cells, as the theoretical results holds in the asymptotic limit (i.e., as the mesh size tends towards 0). Here, in the asymptotic region, the rate is limited by the regularity

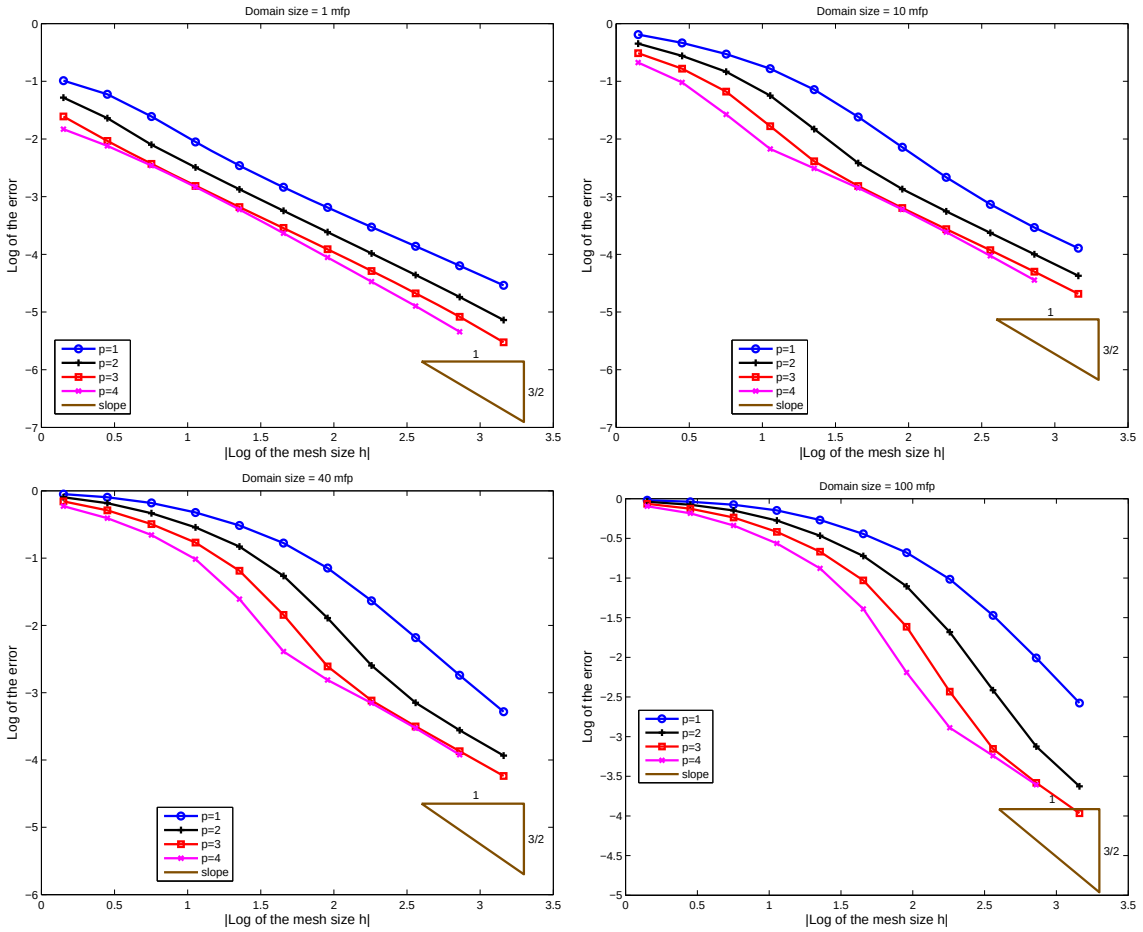


Fig. II-22. Convergence rates: pure absorber case with ∼18° (left+bottom)-face incidence, structured mesh not aligned with incident beam, domain size = 1 (top left), 10 (top right), 40 (bottom left), and 100 (bottom right) MFP.

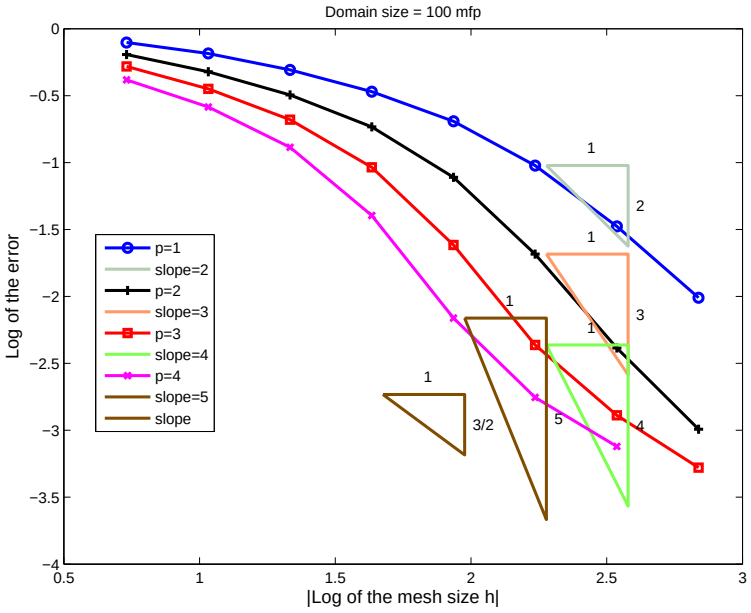


Fig. II-23. Convergence rates: pure absorber case with ∼18° (left+bottom)-face incidence, unstructured mesh not aligned with incident beam, domain size = 100 MFP.

of the solution and the rates graphed in Fig. II-24 shows the transitions from the higher-order convergence rate region (coarser meshes) to a convergence rate of 3/2 (finer/optically thin meshes). These figures are for unstructured meshes. Similar results, not presented here for brevity, were obtained for structured meshes. In the bottom right graph of Fig. II-24 (domain size = 100 MFP), the asymptotic slope of 3/2 and the pre-asymptotic slopes of $p + 1$ are shown.

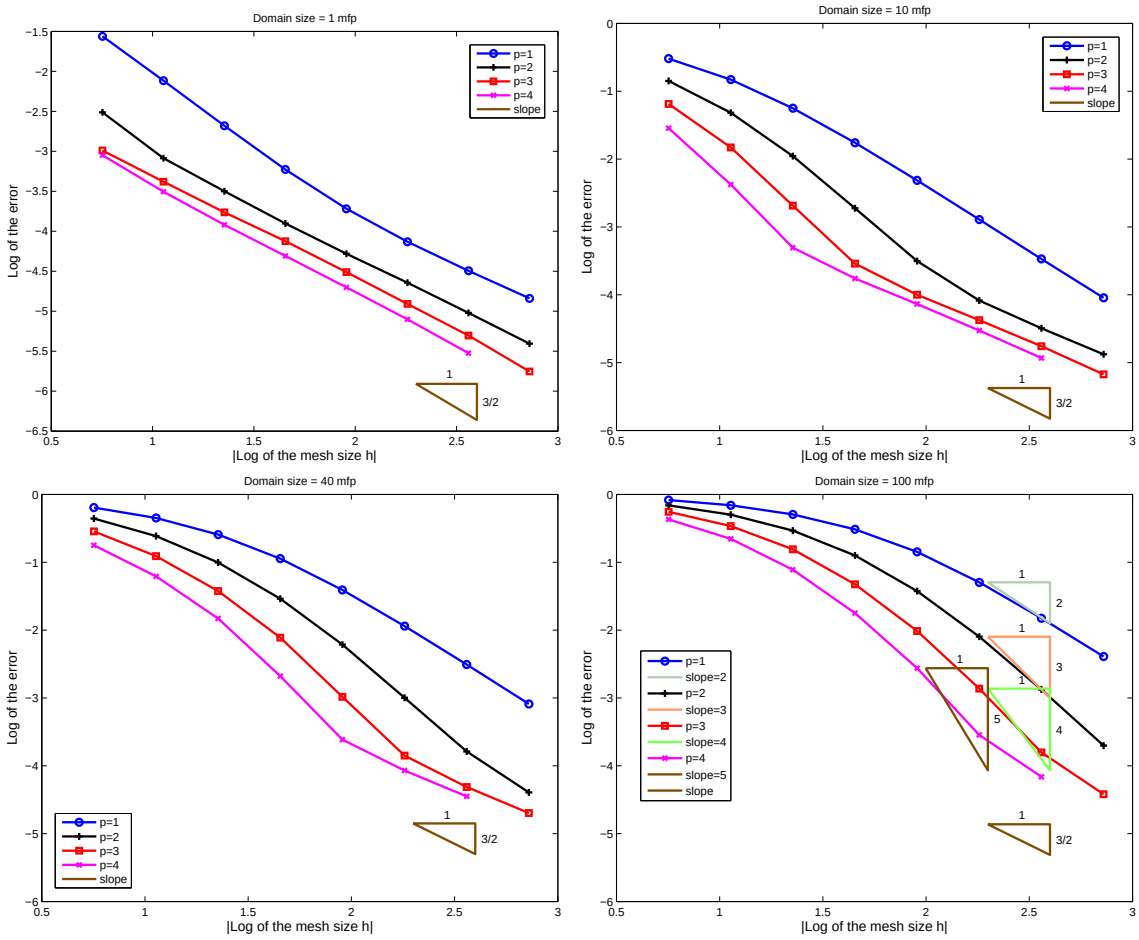


Fig. II-24. Convergence rates: scatterer case with 45° left-face incidence, unstructured mesh aligned with incident beam, domain size = 1 (top left), 10 (top right), 40 (bottom left), and 100 (bottom right) MFP.

For a beam incident at an angle of ∼18 degrees, we have again the case where the meshes are not aligned with the singularity of the transport solution. The observed convergence rates varied from a polynomial-order dependent high value to a regularity imposed rate of 1/2 (see Fig. II-25).

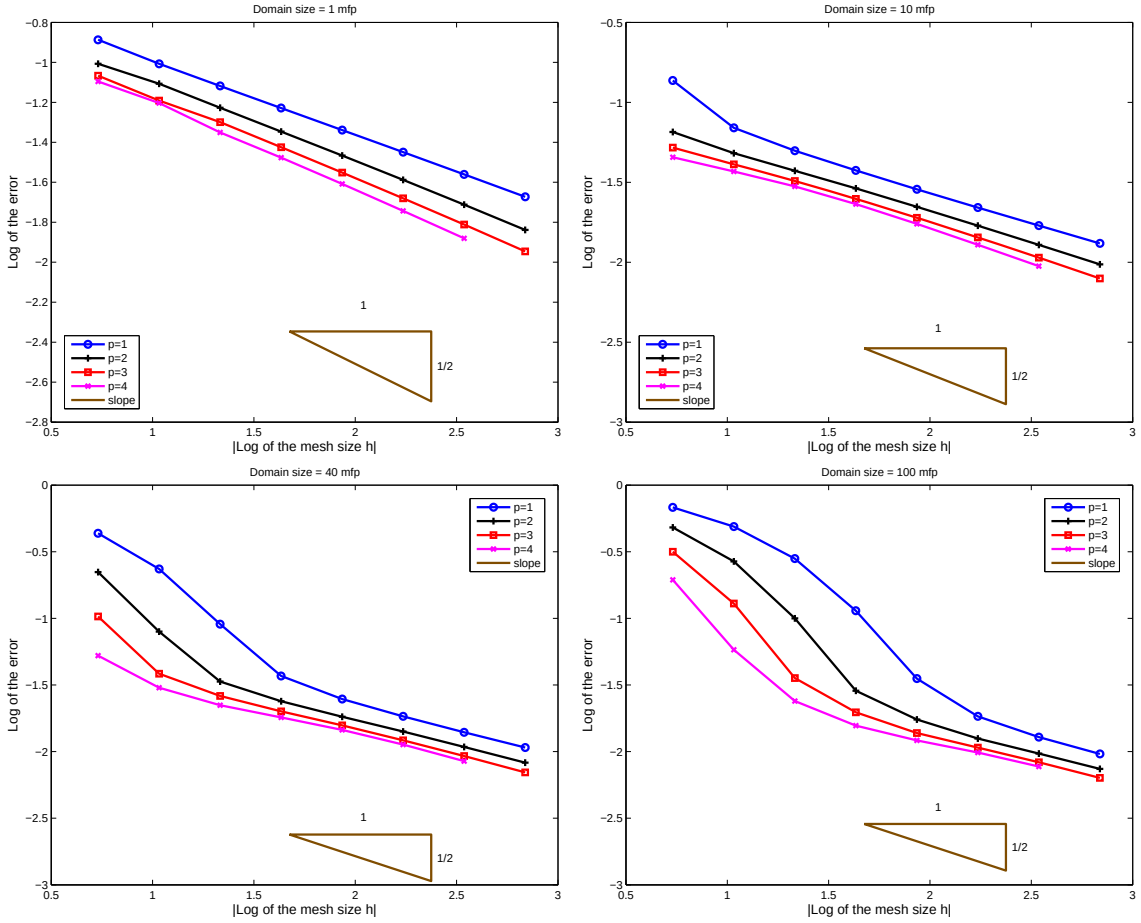
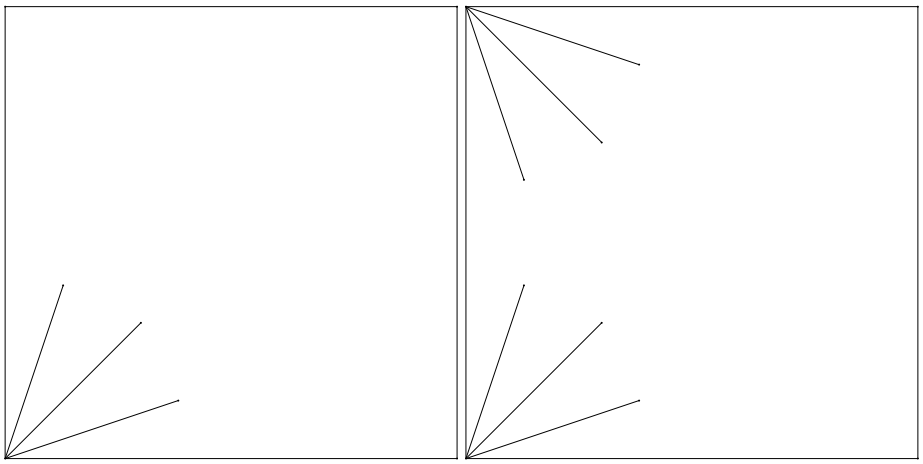


Fig. II-25. Convergence rates: scatterer case with ∼18° left-face incidence, unstructured mesh not aligned with incident beam, domain size = 1 (top left), 10 (top right), 40 (bottom left), and 100 (bottom right) MFP.

d. Flux Incident on the Left Face, Scatterer Material Case with Partial Mesh Alignment



(a) The bottom-left corner only (b) The bottom-left and top-left corners

Fig. II-26. Meshes partially aligned with singularities.

The requirement of having a mesh fully aligned with the singularities may seem a severe obstacle in general, and for adaptive mesh refinement in particular. Recognizing that mesh adaptation has been successfully applied in other engineering fields with hyperbolic equations [105], we propose a simple test, where the domain contains meshes that are partially aligned with the singularity (Fig. II-26). We use the scatterer test case of Section c, where the asymptotic convergence rate is 3/2 for meshes fully aligned with the singularity and 1/2 for unaligned meshes (see Figs. II-24 and II-25). In Section c, we have also noted that before reaching their asymptotic rates, rates close to $p+1$ can be attained. We now compare these previous results with the results obtained with partially aligned meshes and shown in Fig. II-26. In Fig. II-26 (left), the mesh is aligned with the 3 singularities (one per ordinate) present at the lower left

corner. In Fig. II-26 (right), the mesh is also aligned with the 3 singularities present at the upper left corner. For conciseness, we only show the mesh that is the input data for the Triangle mesh generator. We also generated meshes partially aligned with the singularities of all four corners.

We first carried out tests where singularities are partially meshed for the bottom left corner only. For very thin domains, rates of 1/2 are reached; the mesh alignment is too short (optically thin domain) to recover a higher convergence rate. Nonetheless, the errors obtained with the partially aligned meshes are smaller by a factor 3 to 5 compared with the case where the mesh is not aligned with the singularity (see Fig. II-27). For domains greater than 10 MFP (see top right and bottom left graphs on Fig. II-27), the asymptotic rate is now 3/2, which is a vast improvement over the 1/2 rate for unaligned meshes. The magnitude of the error has also been further decreased by two orders of magnitude, in comparison with the unaligned case.

When tests were repeated in which the singularities were partially meshed for both lower-left and upper-left corners, the observed asymptotic rates for thin domains were 3/2 (as compared to 1/2 previously). Thus, taking into account singularities from both corners allowed us to recover the maximum regularity-constrained theoretical rate. For thicker domains (greater than 20 MFP), the partial meshing of the singularities is enough to recover convergence rates of $p + 1$, i.e., the results are no longer constrained by the solution regularity. We present the results for a 40-MFP thick domain. The accuracy gains range from about 0.5 ($p = 1$) to almost 2 ($p = 4$) orders of magnitude (see bottom right graph on Fig. II-27). This is particularly noteworthy when considering the potential for adaptive mesh computations in transport. As we have seen in our tests, even if aligning the mesh was not always sufficient to augment the convergence rate, the magnitude of the error always decreased significantly. At that stage, and by surveying current practice in other disciplines, we can conjecture

that accuracy gains can be obtained by resolving, albeit partially, the singularity in the transport equation.

e. DG Norm Computations

We recalled in the theory Section 3 that the error, measured in the DG norm, converges at a rate that is reduced by 1/2 in comparison to the L_2 norm. We briefly provide numerical results that corroborate these theoretical facts. First, on Fig. II-28, we show the rates in the case of a pure absorber where the mesh is aligned with the singularity (domain size = 10 MFP, left graph on Fig. II-28). In the L_2 norm, rates were of $p + 1$ (see previous results), now we clearly observed rates of $p + 1/2$. Then, we tested a pure absorber case, where the mesh does not align with the singularity (domain size = 100 MFP, right graph on Fig. II-28). In this situation, the theoretical value of the error in the DG norm is 0, which can be noted in Fig. II-28 (right).

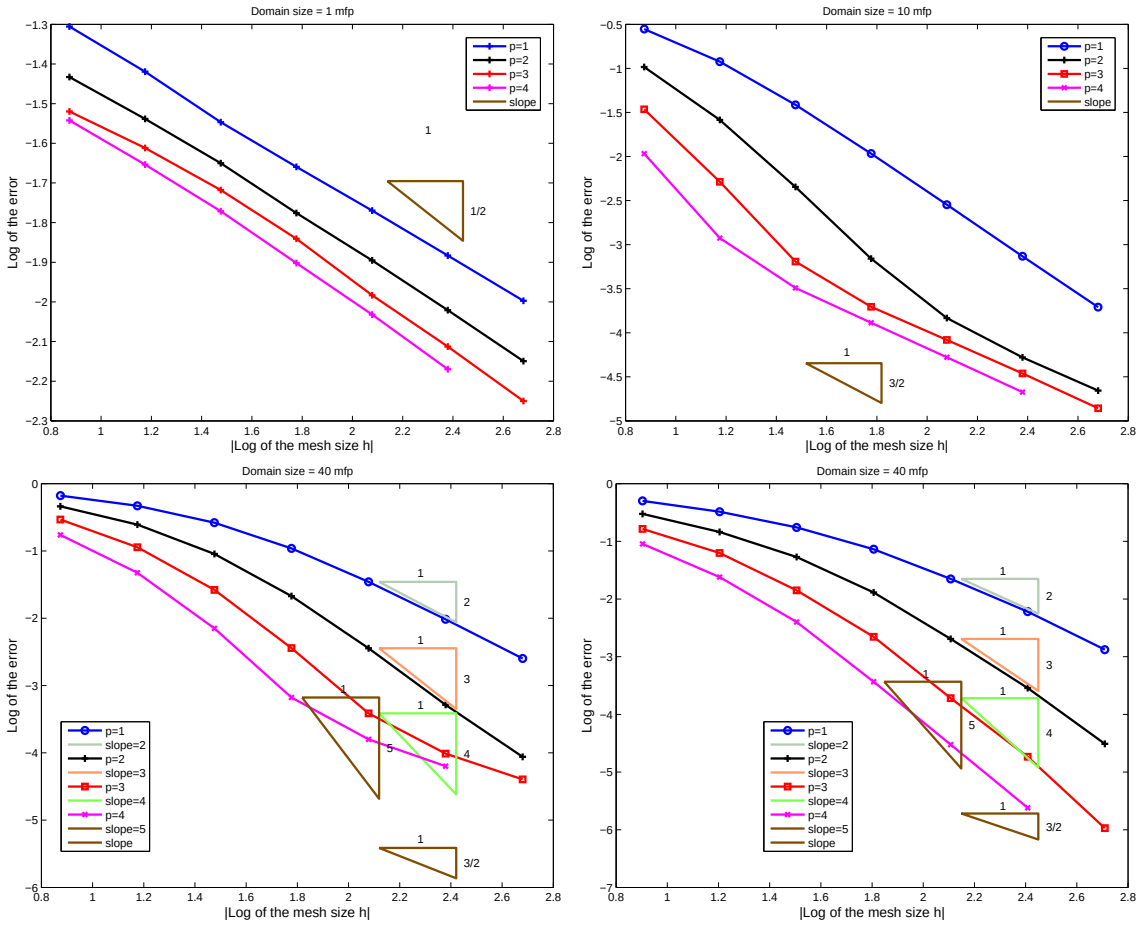


Fig. II-27. Convergence rates: scatterer case with ∼18° left-face incidence, mesh partially aligned (i) with singularities at the lower-left corner, domain size = 1 MFP (top-left graph), 10 MFP (top-right graph), 40 MFP (bottom-right graph) and (ii) singularities at both the bottom-left and upper-left corners (bottom-right graph).

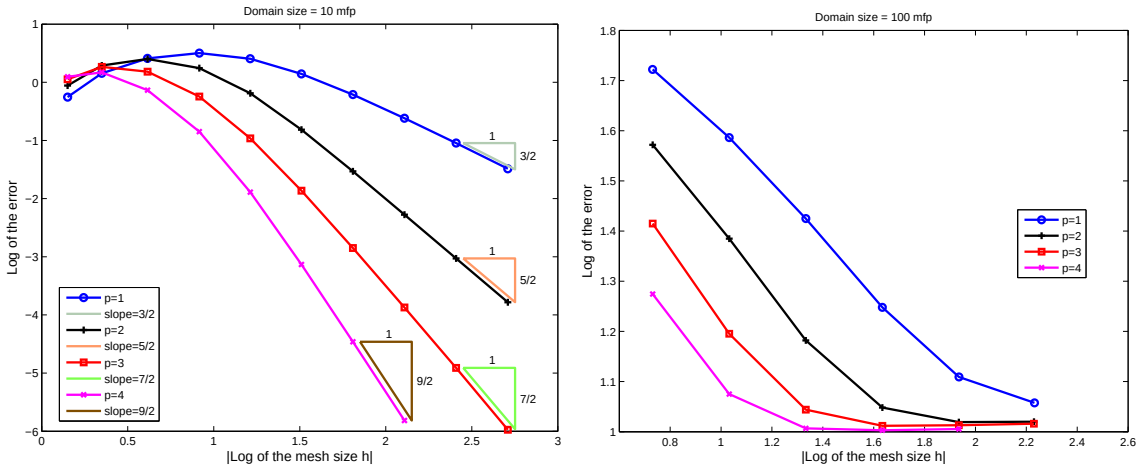


Fig. II-28. Convergence rates in the DG norm: pure absorber case with 45° left-face incidence, aligned with incident beam, (left: domain size = 10, structured mesh; right: domain size = 100 MFP, unstructured mesh).

E. Conclusions

We have presented higher-order numerical solutions to the S_N transport equation for unstructured 2D triangular meshes. A Discontinuous Galerkin Finite Element Method (DGFEM) was employed, with orders up to four. Hierarchical basis functions were chosen for the spatial representation of the solution. This basis set allows for an easy implementation of the upwinding procedure for any polynomial order and is widely used in other engineering disciplines for accurate higher-order solutions . We have numerically observed that the solutions converge at the theoretical rate of $p + 1$, where p is the order of the approximation. Notably, for a given mesh size, there are always significant gains in accuracy to be obtained from quadratic, cubic, and quartic approximations with respect to linear DGFEM. In terms of CPU time, cubic and quartic functions yield about the same accuracy per unit time, suggesting that, in 2D, the best accuracy/CPU time compromise may be attained for orders 3

and 4. Our results show that a linear representation of the solution is not optimal due to the lower convergence rate of the method, and we recommend that at least second-order functions be employed. Using the above framework, a mesh adaptive S_N transport solver is under development; the use of a DGFEM method will facilitate the coupling between elements of various refinement levels (in DGFEM, the solution is not required to be continuous across elements) and the hierarchical basis functions will allow for simpler inter-element communications.

We have numerically analyzed the convergence properties of Discontinuous Galerkin Finite Elements, up to polynomial order 4, for the spatial discretization of the transport equation. Test cases were carried out with pure absorber media and scatterer media, for structured and unstructured triangular meshes. We have verified theoretical convergence results, namely that, in the L_2 norm, the solutions converge with a rate of $min(p+1, r)$, where p is the spatial approximation order and r the transport solution regularity. In the DG norm, the theoretical rate of $min(p+1/2, r-1/2)$ was recovered. For optically thin meshes, the convergence rate in the L_2-norm is always imposed by the solution regularity ($r = 1/2$ or $3/2$), but for thicker meshes, rates approaching $p + 1$ are observed. We note that for optically thick domains, the error is significantly reduced by the convergence rate in $p + 1$ before being limited by regularity for very fine meshes. In the cases where the meshes are partially aligned with a few singularities, convergence rates of $3/2$ (as opposed to $1/2$) can be attained. When more singularities are properly meshed, convergence at the rate of $p + 1$ can be observed.

CHAPTER III

DIFFUSION SYNTHETIC ACCELERATION SCHEMES FOR HIGH-ORDER DISCONTINUOUS FINITE ELEMENTS ON LOCALLY REFINED UNSTRUCTURED MESHES

A. Introduction

In this chapter, we develop and analyze Diffusion Synthetic Acceleration (DSA) schemes for higher-order discontinuous finite element (DFE) spatial discretizations of the S_N transport equation on 2-D, unstructured, locally refined meshes. The spatial discretization of the S_N transport equation on unstructured meshes has been described in Chapter II.

For problems with highly diffusive materials (i.e., with scattering ratios $c = \sigma_{s,0}/\sigma_t$ close to 1), the standard source iteration (SI) technique can become quite ineffective due to its slow convergence properties and DSA needs to be employed to accelerate the convergence of the SI process. It is well established that the spatial discretization of the DSA equations must be "consistent" with the one used for the S_N transport equations to yield unconditionally stable and effective DSA schemes [106, 107, 108, 70, 68, 67]. To date, the work by Warsa & Morel [70] regarding a fully-consistent DSA scheme for linear discontinuous discretizations on tetrahedrons is the only fully-consistent example of DSA for general meshes. Their DSA method was derived by using the zeroth and first angular moment of the discretized transport equation, resulting in a mixed-diffusion or P_1 system of equations, with a scalar continuity equation and a first moment vector equation. Even though their scheme achieved full consistency, the overall computational efficiency of their method only outperformed partially consistent schemes under certain circumstances (e.g., for

problems that are both highly diffusive and require high angular quadrature order). Partially consistent DSA schemes have been motivated by the difficulties associated with the algebraic elimination of the vector unknowns to yield an elliptic diffusion equations. With partial consistency, it is hoped that the reduction in the scheme's complexity outweighs the degradation in its effectiveness. Some partially consistent schemes have been analyzed for unstructured meshes [70]: the modified-four-step (M4S) scheme and the Wareing-Larsen-Adams (WLA) scheme. The M4S technique, though efficient in 1-D slab and 2-D rectangular geometries, was found to be divergent for 3-D tetrahedral meshes with linear discontinuous elements. The WLA scheme, based on the solution of a diffusion equation using continuous finite element (CFE) followed by a discontinuous update carried out cell-by-cell, was found to be stable and relatively effective, though the effectiveness degraded as the element sizes became more optically thick and highly diffusive. For the WLA scheme to be used on locally adapted meshes as obtained, for instance, when using AMR, the CFE diffusion equation must be solved on meshes containing hanging nodes (i.e., some nodal unknowns are only present on an element and not on its neighbor, with the consequence that these unknowns must be constrained in order to keep a numerical approximation that is continuous across elements). This can be a non-negligible task, especially for higher-order approximations on unstructured meshes. Some researchers [109] have presented DSA schemes based on linear DFE for Block-AMR meshes; our approach here allows for arbitrary mesh structures, arbitrary refinement level, and arbitrary polynomial order representation.

In our new approach, we have chosen to derive partially-consistent DSA schemes employing a DFE discretization by *directly* deriving them from the DFE discretization of the S_N transport equations. Our scheme belongs to the family of partially-consistent DSA methods because in our derivation, we only keep the zero-th moment

of the DFE transport equation and assume that Fick's law is verified point-wise to eliminate the current unknowns. We show that the resulting DFE discretization for the diffusion equation is remarkably similar to the Interior Penalty (IP) stabilization method for diffusion equations solved using a DFE Method (in the mathematical literature, such approximation is also referred to as Discontinuous Galerkin Finite Element Method, or DGFEM). Due to the discontinuous nature of the DGFEM approximation, it is particularly well suited for meshes arising in AMR calculations, i.e., hanging nodes are seamlessly incorporated into the DFE method. This property also leads to an easy implementation of higher-order test functions and in this work, we have employed test functions with polynomial orders up to 4.

The outline of this chapter is as follows.In Section B, we derive the DFE diffusion equation, starting directly from the DFE variational form of S_N transport equation. We link this so-called "conforming" diffusion form we arrive at with the standard IP DGFEM diffusion form. We label our DFE diffusion forms as "conforming" because they are derived directly from the S_N transport variational form. We also derive a P_1 conforming form and compare with the mixed P_1 form from Ref. [70]. We then present the local matrices obtained from our DFE diffusion form in the case of higher-order polynomial approximations, and describe how the DFE diffusion equations are solved in a matrix-free fashion using a preconditioned Conjugate Gradient (CG) method. (we have chosen SSOR as a preconditioner for CG.) In Section D, we show how the DFE diffusion forms can be used as DSA preconditioners to accelerate the SI and GMRes transport solves. In Section E, we perform a Fourier analysis for the various DSA schemes, for both homogeneous and heterogeneous medium configurations. In the "Results" Section F, we compare the spectral radius obtained from Fourier analysis with numerical estimates of the spectral radius from the XUTHUS S_N code and discuss the effectiveness and efficiency of DSA for various polynomial

orders. Finally, conclusions are provided in Section G.

B. Derivation of Discontinuous Finite Element Diffusion Forms

The starting point for the derivation of Discontinuous Finite Element (DFE) diffusion equations is the variational form for the S_N transport equation, Eq. (2.21) in Chapter II. To obtain the conforming diffusion form, we restrict the angular flux solution and angular test function spaces of a smaller subspace, where their angular dependence is only linear. Different approximations will lead to either the diffusion conforming form or the P_1 conforming DFE forms. Note that all terms in the equation will be multiplied by 4π in the following derivation.

1. The Diffusion Conforming Form, DCF

First let us assume a diffusion approximation for the primal and adjoint angular fluxes,

$$\Psi_m = \frac{1}{4\pi}(\Phi - 3D\vec{\nabla}\Phi \cdot \vec{\Omega}_m + 9D\vec{Q}_1 \cdot \vec{\Omega}_m) \tag{3.1}$$

$$\Psi^*_m = \frac{1}{4\pi}(\Phi^* + 3D\vec{\nabla}\Phi^* \cdot \vec{\Omega}_m) \tag{3.2}$$

Both the primal and adjoint angular fluxes are linearly anisotropic. We have used the following Fick's laws to eliminate the first angular moment (i.e., the net current) in the above expressions:

$$\vec{J} = -D\vec{\nabla}\Phi + 3D\vec{Q}_1 \tag{3.3}$$

$$\vec{J}^* = D\vec{\nabla}\Phi^* \tag{3.4}$$

The primal current satisfies a generalized Fick's law, where the linearly anisotropic

source is accounted for. We will later see that D will naturally turn out to be the standard diffusion coefficient; the change in signs between the angular fluxes and the test functions will make the reduced diffusion system symmetric. The first-angular moment vector $\vec{Q}_1$ is understood as containing the source term, weighted by the 3 first-order spherical harmonic functions, i.e., $\vec{Q}_1 = [\ Q_{1,-1} \quad Q_{1,0} \quad Q_{1,1}\ ]$. This term will be important when the anisotropy of scattering is strong. In the following pages, we derive the DFE diffusion conforming approximation, a reduced form that only contains the scalar flux as the unknown. The differences between the diffusion approximation and the P_1 approximation will be showed later.

We start the derivation by evaluating the simplest terms in the S_N variational form: the total and scattering reaction terms.

$$\begin{aligned}\sum_{m=1}^{M} 4\pi w_m \left(\sigma_t \Psi_m, \Psi_m^*\right)_{\mathcal{D}} &= \left(\sigma_t \Phi, \Phi^*\right)_{\mathcal{D}} - \left(3\sigma_t D\vec{\nabla}\Phi, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}} \\ &\quad + \left(9\sigma_t D\vec{Q}_1, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}} \qquad (3.5)\\ \sum_{n=0}^{N_a}\sum_{k=-n}^{n}(2n+1)\left(\sigma_{s,n}\Phi_{n,k}, \Phi_{n,k}^*\right)_{\mathcal{D}} &= \left(\sigma_{s,0}\Phi, \Phi^*\right)_{\mathcal{D}} - \left(3\sigma_{s,1} D\vec{\nabla}\Phi, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}} \\ &\quad - \left(9\sigma_{s,1} D\vec{Q}_1, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}} \qquad (3.6)\end{aligned}$$

Both terms are multiplied with 4π. The following properties of the angular quadrature

have been used:

$$\sum_{m=1}^{M} w_m = 4\pi \tag{3.7}$$

$$\sum_{m=1}^{M} w_m \vec{\Omega}_m = 0 \tag{3.8}$$

$$\sum_{m=1}^{M} w_m \vec{\Omega}_m \vec{\Omega}_m = \frac{4\pi}{3} I \tag{3.9}$$

$$\sum_{m=1}^{M} w_m \vec{\Omega}_m \vec{\Omega}_m \vec{\Omega}_m = 0 \tag{3.10}$$

and we recall the meaning of $(a,b)_{\mathcal{D}} = \sum_K (a,b)_K = \sum_K \int_K d^3r\, ab$. Merging these two terms and defining,

$$D = \frac{1}{3(\sigma_t - \sigma_{s,1})} = \frac{1}{3\sigma_{tr}} \tag{3.11}$$

$$\sigma_a = \sigma_t - \sigma_{s,0}, \tag{3.12}$$

we obtain,

$$(\sigma_a \Phi, \Phi^*)_{\mathcal{D}} - \left(\vec{\nabla}\Phi, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}} + \left(3\vec{Q}_1, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}} \tag{3.13}$$

Before processing the edge terms appearing in the DFE S_N variational form, we introduce the following edge definitions for the scalar flux on the interior edges (similar definitions for the derivatives of scalar flux on the edges can be inferred from these). Note the difference in the definition between the edge angular fluxes (where the $\pm$ superscript depended on the $\lim_{s\to 0^{\pm}} \Psi(\vec{r} + s\vec{\Omega}_m)$) and the scalar flux (there are no

specific directions associated with the scalar flux.)

$$\Phi^+ = \lim_{s\to 0^+} \Phi(\vec{r}+s\vec{n}) \tag{3.14}$$

$$\Phi^- = \lim_{s\to 0^-} \Phi(\vec{r}+s\vec{n}) \tag{3.15}$$

$$[\![\Phi]\!] = \Phi^+ - \Phi^- \tag{3.16}$$

$$\{\!\{\Phi\}\!\} = (\Phi^+ + \Phi^-)/2 \tag{3.17}$$

$\vec{n}_e(\vec{r})$ is a fixed normal unit vector of an edge e. The orientation of $\vec{n}$ on an interior edge is irrelevant. However, on the boundary edges, this vector must be oriented outward.

We now analyze the expression resulting from the S_N streaming term, when the primal and test functions are restricted to a linear angular dependence:

$$\begin{aligned}
&\sum_{m=1}^{M} 4\pi w_m \left(\vec{\Omega}_m \cdot \vec{\nabla}\Psi_m, \Psi_m^*\right)_{\mathcal{D}} \\
&\qquad = \left(\vec{\nabla}\Phi, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}} - \left(\vec{\nabla}\cdot D\vec{\nabla}\Phi, \Phi^*\right)_{\mathcal{D}} + \left(3\vec{\nabla}\cdot D\vec{Q}_1, \Phi^*\right)_{\mathcal{D}} \\
&\qquad = \left(\vec{\nabla}\Phi, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}} + \left(D\vec{\nabla}\Phi, \vec{\nabla}\Phi^*\right)_{\mathcal{D}} + \left(3\vec{\nabla}\cdot D\vec{Q}_1, \Phi^*\right)_{\mathcal{D}} \\
&\qquad\quad + \left(D\vec{\nabla}\Phi^+\cdot\vec{n}, \Phi^{*,+}\right)_{E_h^i} - \left(D\vec{\nabla}\Phi^-\cdot\vec{n}, \Phi^{*,-}\right)_{E_h^i} \\
&\qquad\quad - \left(D\vec{\nabla}\Phi\cdot\vec{n}, \Phi^*\right)_{\partial\mathcal{D}}
\end{aligned} \tag{3.18}$$

We note that one of the ($\vec{\nabla}$, $\vec{\nabla}$) term will cancel out an identical term in Eq. (3.13). Also note that integration by parts was applied in the last step of Eq. (3.18). Let us

now consider the term related to the interior edges:

$$\sum_{m=1}^{M} 4\pi w_m \left\langle [\![\Psi_m]\!], \Psi_m^{*+} \right\rangle_{E_h^i} = \sum_{e\in E_h^i} \sum_{m=1}^{M} 4\pi w_m |\vec{\Omega}_m \cdot \vec{n}_e| \left([\![\Psi_m]\!], \Psi_m^{*+} \right)_e$$

$$= \sum_{e\in E_h^i} \left[\begin{array}{l} \sum\limits_{\vec{\Omega}_m\cdot\vec{n}_e>0} \frac{w_m}{4\pi} |\vec{\Omega}_m \cdot \vec{n}_e| \\ \qquad \left([\![\Phi]\!] - 3[\![D\vec{\nabla}\Phi]\!] \cdot \vec{\Omega}_m + 9[\![D\vec{Q}_1]\!] \cdot \vec{\Omega}_m, \Phi^{*+} + 3D\vec{\nabla}\Phi^{*+} \cdot \vec{\Omega}_m \right)_e \\ - \sum\limits_{\vec{\Omega}_m\cdot\vec{n}_e<0} \frac{w_m}{4\pi} |\vec{\Omega}_m \cdot \vec{n}_e| \\ \qquad \left([\![\Phi]\!] - 3[\![D\vec{\nabla}\Phi]\!] \cdot \vec{\Omega}_m + 9[\![D\vec{Q}_1]\!] \cdot \vec{\Omega}_m, \Phi^{*-} + 3D\vec{\nabla}\Phi^{*-} \cdot \vec{\Omega}_m \right)_e \end{array} \right]$$

$$= \sum_{e\in E_h^i} \left[\begin{array}{l} \sum\limits_{\vec{\Omega}_m\cdot\vec{n}_e>0} \frac{w_m}{4\pi} |\vec{\Omega}_m \cdot \vec{n}_e| \\ \qquad \left([\![\Phi]\!] - 3[\![D\vec{\nabla}\Phi]\!] \cdot \vec{\Omega}_m + 9[\![D\vec{Q}_1]\!] \cdot \vec{\Omega}_m, \Phi^{*+} + 3D\vec{\nabla}\Phi^{*+} \cdot \vec{\Omega}_m \right)_e \\ - \sum\limits_{\vec{\Omega}_d\cdot\vec{n}_e>0} \frac{w_d}{4\pi} |\vec{\Omega}_m \cdot \vec{n}_e| \\ \qquad \left([\![\Phi]\!] + 3[\![D\vec{\nabla}\Phi]\!] \cdot \vec{\Omega}_m - 9[\![D\vec{Q}_1]\!] \cdot \vec{\Omega}_m, \Phi^{*-} - 3D\vec{\nabla}\Phi^{*-} \cdot \vec{\Omega}_m \right)_e \end{array} \right]$$

$$\begin{aligned} &= \frac{1}{4} \left([\![\Phi]\!], [\![\Phi^*]\!] \right)_{E_h^i} + \left([\![\Phi]\!], \{\!\{ D\vec{\nabla}\Phi^* \cdot \vec{n} \}\!\} \right)_{E_h^i} - \left([\![D\vec{\nabla}\Phi \cdot \vec{n}]\!], \{\!\{ \Phi^* \}\!\} \right)_{E_h^i} \\ &\quad - \frac{9}{16} \left([\![D\vec{\nabla}\Phi]\!], [\![D\vec{\nabla}\Phi^*]\!] \right)_{E_h^i} - \frac{9}{16} \left([\![D\vec{\nabla}\Phi \cdot \vec{n}]\!], [\![D\vec{\nabla}\Phi^* \cdot \vec{n}]\!] \right)_{E_h^i} \\ &\quad + \left([\![3D\vec{Q}_1 \cdot \vec{n}]\!], \{\!\{ \Phi^* \}\!\} \right)_{E_h^i} + \frac{9}{16} \left([\![3D\vec{Q}_1]\!], [\![D\vec{\nabla}\Phi^*]\!] \right)_{E_h^i} \\ &\quad + \frac{9}{16} \left([\![3D\vec{Q}_1 \cdot \vec{n}]\!], [\![D\vec{\nabla}\Phi^* \cdot \vec{n}]\!] \right)_{E_h^i} \end{aligned} \tag{3.19}$$

where we have used the following definition for the edge integral in the context of the diffusion equation:

$$\left(\Phi, \Phi^* \right)_e = \int_e \Phi\Phi^* \, \mathrm{d}s \tag{3.20}$$

(Recall that in the context of the transport equation, the edge integral contains a $|\vec{\Omega}_m \cdot \vec{n}_e|$ term.) We have also employed the following properties of the angular

quadrature and have assumed that if $\vec{\Omega}_m$ is in the angular quadrature set, so is $-\vec{\Omega}_m$.

$$\sum_{\vec{\Omega}_m\cdot\vec{n}>0} w_m|\vec{\Omega}_m\cdot\vec{n}| = \pi \tag{3.21}$$

$$\sum_{\vec{\Omega}_m\cdot\vec{n}>0} w_m|\vec{\Omega}_m\cdot\vec{n}|\vec{\Omega}_m = \frac{2\pi}{3}\vec{n} \tag{3.22}$$

$$\sum_{\vec{\Omega}_m\cdot\vec{n}>0} w_m|\vec{\Omega}_m\cdot\vec{n}|\vec{\Omega}_m\vec{\Omega}_m = \frac{\pi}{4}(I+\vec{n}\vec{n}) \tag{3.23}$$

Note that $\vec{n}\vec{n}$ is a rank-2 tensor. The boundary terms are treated next:

$$\begin{aligned}
&\sum_{e\in\partial\mathcal{D}}\sum_{\vec{\Omega}_m\cdot\vec{n}_e<0}\frac{w_m}{4\pi}|\vec{\Omega}_m\cdot\vec{n}_e|\left(\Phi-3D\vec{\nabla}\Phi\cdot\vec{\Omega}_m+9D\vec{Q}_1\cdot\vec{\Omega}_m,\Phi^*+3D\vec{\nabla}\Phi^*\cdot\vec{\Omega}_m\right)_e-\\
&\sum_{e\in\partial\mathcal{D}^r}\sum_{\vec{\Omega}_m\cdot\vec{n}_e<0}\frac{w_m}{4\pi}|\vec{\Omega}_m\cdot\vec{n}_e|\left(\Phi-(3D\vec{\nabla}\Phi-9D\vec{Q}_1)\cdot(\vec{\Omega}_m-2(\vec{\Omega}_m\cdot\vec{n}_e)\vec{n}_e),\Phi^*+3D\vec{\nabla}\Phi^*\cdot\vec{\Omega}_m\right)_e\\
&\quad=\sum_{e\in\partial\mathcal{D}^d}\sum_{\vec{\Omega}_m\cdot\vec{n}_e<0}\frac{w_m}{4\pi}|\vec{\Omega}_m\cdot\vec{n}_e|\left(\Phi-3D\vec{\nabla}\Phi\cdot\vec{\Omega}_m+9D\vec{Q}_1\cdot\vec{\Omega}_m,\Phi^*+3D\vec{\nabla}\Phi^*\cdot\vec{\Omega}_m\right)_e-\\
&\qquad\sum_{e\in\partial\mathcal{D}^r}\sum_{\vec{\Omega}_m\cdot\vec{n}_e<0}\frac{w_m}{4\pi}|\vec{\Omega}_m\cdot\vec{n}_e|\left(2(\vec{\Omega}_m\cdot\vec{n}_e)(3D\vec{\nabla}\Phi-9D\vec{Q}_1)\cdot\vec{n}_e,\Phi^*+3D\vec{\nabla}\Phi^*\cdot\vec{\Omega}_m\right)_e\\
&\quad=\sum_{e\in\partial\mathcal{D}^d}\sum_{\vec{\Omega}_m\cdot\vec{n}_e>0}\frac{w_m}{4\pi}|\vec{\Omega}_m\cdot\vec{n}_e|\left(\Phi+3D\vec{\nabla}\Phi\cdot\vec{\Omega}_m-9D\vec{Q}_1\cdot\vec{\Omega}_m,\Phi^*-3D\vec{\nabla}\Phi^*\cdot\vec{\Omega}_m\right)_e+\\
&\qquad\sum_{e\in\partial\mathcal{D}^r}\sum_{\vec{\Omega}_m\cdot\vec{n}_e>0}\frac{w_m}{4\pi}|\vec{\Omega}_m\cdot\vec{n}_e|\left(2(\vec{\Omega}_m\cdot\vec{n}_e)(3D\vec{\nabla}\Phi-9D\vec{Q}_1)\cdot\vec{n}_e,\Phi^*-3D\vec{\nabla}\Phi^*\cdot\vec{\Omega}_m\right)_e\\
&\quad=\frac{1}{4}\left(\Phi,\Phi^*\right)_{\partial\mathcal{D}^d}-\frac{1}{2}\left(\Phi,D\vec{\nabla}\Phi^*\cdot\vec{n}\right)_{\partial\mathcal{D}^d}+\frac{1}{2}\left(D\vec{\nabla}\Phi\cdot\vec{n},\Phi^*\right)_{\partial\mathcal{D}^d}\\
&\qquad-\frac{9}{16}\left(D\vec{\nabla}\Phi,D\vec{\nabla}\Phi^*\right)_{\partial\mathcal{D}^d}-\frac{9}{16}\left(D\vec{\nabla}\Phi\cdot\vec{n},D\vec{\nabla}\Phi^*\cdot\vec{n}\right)_{\partial\mathcal{D}^d}\\
&\qquad-\frac{1}{2}\left(3D\vec{Q}_1\cdot\vec{n},\Phi^*\right)_{\partial\mathcal{D}^d}+\frac{9}{16}\left(3D\vec{Q}_1,D\vec{\nabla}\Phi^*\right)_{\partial\mathcal{D}^d}+\frac{9}{16}\left(3D\vec{Q}_1\cdot\vec{n},D\vec{\nabla}\Phi^*\cdot\vec{n}\right)_{\partial\mathcal{D}^d}\\
&\qquad+\left(D\vec{\nabla}\Phi\cdot\vec{n},\Phi^*\right)_{\partial\mathcal{D}^r}-\frac{9}{4}\left(D\vec{\nabla}\Phi\cdot\vec{n},D\vec{\nabla}\Phi^*\cdot\vec{n}\right)_{\partial\mathcal{D}^r}\\
&\qquad-\left(3D\vec{Q}_1\cdot\vec{n},\Phi^*\right)_{\partial\mathcal{D}^r}+\frac{9}{4}\left(3D\vec{Q}_1\cdot\vec{n},D\vec{\nabla}\Phi^*\cdot\vec{n}\right)_{\partial\mathcal{D}^r}
\end{aligned} \tag{3.24}$$

Now, putting all terms in Eqs. (3.13), (3.18), (3.19) and (3.24) together, we obtain the following diffusion conforming form (DCF):

$$b\left(\Phi,\Phi^*\right)=l(\Phi^*) \tag{3.25}$$

where the bilinear form is given by:

$$\begin{aligned}
b\left(\Phi,\Phi^*\right) &= \left(\sigma_a\Phi,\Phi^*\right)_{\mathcal{D}} - \underline{\left(\vec{\nabla}\Phi, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}}} + \left(3\vec{Q}_1, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}} \\
&+ \underline{\left(\vec{\nabla}\Phi, D\vec{\nabla}\Phi^*\right)_{\mathcal{D}}} + \left(D\vec{\nabla}\Phi, \vec{\nabla}\Phi^*\right)_{\mathcal{D}} + \left(3\vec{\nabla}\cdot D\vec{Q}_1, \Phi^*\right)_{\mathcal{D}} \\
&+ \underline{\underline{\left(D\vec{\nabla}\Phi^+\cdot\vec{n}, \Phi^{*,+}\right)_{E_h^i} - \left(D\vec{\nabla}\Phi^-\cdot\vec{n}, \Phi^{*,-}\right)_{E_h^i}}} - \left(D\vec{\nabla}\Phi\cdot\vec{n}, \Phi^*\right)_{\partial\mathcal{D}} \\
&+ \frac{1}{4}\left(\left[\!\left[\Phi\right]\!\right], \left[\!\left[\Phi^*\right]\!\right]\right)_{E_h^i} + \left(\left[\!\left[\Phi\right]\!\right], \left\{\!\left\{D\vec{\nabla}\Phi^*\cdot\vec{n}\right\}\!\right\}\right)_{E_h^i} - \underline{\underline{\left(\left[\!\left[D\vec{\nabla}\Phi\cdot\vec{n}\right]\!\right], \left\{\!\left\{\Phi^*\right\}\!\right\}\right)_{E_h^i}}} \\
&- \frac{9}{16}\left(\left[\!\left[D\vec{\nabla}\Phi\right]\!\right], \left[\!\left[D\vec{\nabla}\Phi^*\right]\!\right]\right)_{E_h^i} - \frac{9}{16}\left(\left[\!\left[D\vec{\nabla}\Phi\cdot\vec{n}\right]\!\right], \left[\!\left[D\vec{\nabla}\Phi^*\cdot\vec{n}\right]\!\right]\right)_{E_h^i} \\
&+ \left(\left[\!\left[3D\vec{Q}_1\cdot\vec{n}\right]\!\right], \left\{\!\left\{\Phi^*\right\}\!\right\}\right)_{E_h^i} \\
&+ \frac{9}{16}\left(\left[\!\left[3D\vec{Q}_1\right]\!\right], \left[\!\left[D\vec{\nabla}\Phi^*\right]\!\right]\right)_{E_h^i} + \frac{9}{16}\left(\left[\!\left[3D\vec{Q}_1\cdot\vec{n}\right]\!\right], \left[\!\left[D\vec{\nabla}\Phi^*\cdot\vec{n}\right]\!\right]\right)_{E_h^i} \\
&+ \frac{1}{4}\left(\Phi,\Phi^*\right)_{\partial\mathcal{D}^d} - \frac{1}{2}\left(\Phi, D\vec{\nabla}\Phi^*\cdot\vec{n}\right)_{\partial\mathcal{D}^d} + \frac{1}{2}\left(D\vec{\nabla}\Phi\cdot\vec{n}, \Phi^*\right)_{\partial\mathcal{D}^d} \\
&- \frac{9}{16}\left(D\vec{\nabla}\Phi, D\vec{\nabla}\Phi^*\right)_{\partial\mathcal{D}^d} - \frac{9}{16}\left(D\vec{\nabla}\Phi\cdot\vec{n}, D\vec{\nabla}\Phi^*\cdot\vec{n}\right)_{\partial\mathcal{D}^d} \\
&- \frac{1}{2}\left(3D\vec{Q}_1\cdot\vec{n}, \Phi^*\right)_{\partial\mathcal{D}^d} + \frac{9}{16}\left(3D\vec{Q}_1, D\vec{\nabla}\Phi^*\right)_{\partial\mathcal{D}^d} + \frac{9}{16}\left(3D\vec{Q}_1\cdot\vec{n}, D\vec{\nabla}\Phi^*\cdot\vec{n}\right)_{\partial\mathcal{D}^d} \\
&+ \left(D\vec{\nabla}\Phi\cdot\vec{n}, \Phi^*\right)_{\partial\mathcal{D}^r} - \frac{9}{4}\left(D\vec{\nabla}\Phi\cdot\vec{n}, D\vec{\nabla}\Phi^*\cdot\vec{n}\right)_{\partial\mathcal{D}^r} \\
&- \left(3D\vec{Q}_1\cdot\vec{n}, \Phi^*\right)_{\partial\mathcal{D}^r} + \frac{9}{4}\left(3D\vec{Q}_1\cdot\vec{n}, D\vec{\nabla}\Phi^*\cdot\vec{n}\right)_{\partial\mathcal{D}^r}
\end{aligned} \tag{3.26}$$

and the linear functional is:

$$l(\Phi^*) = \left(Q_0,\Phi^*\right)_{\mathcal{D}} + \left(\vec{Q}_1, 3D\vec{\nabla}\Phi^*\right)_{\mathcal{D}} + \left(J^{inc},\Phi^*\right)_{\partial\mathcal{D}^d} - \left(\vec{\Upsilon}^{inc}, D\vec{\nabla}\Phi^*\right)_{\partial\mathcal{D}^d} \tag{3.27}$$

and the incident current and the next higher angular odd moment are defined as:

$$J^{inc} = \sum_{\vec{\Omega}_m \cdot \vec{n}(\vec{r}_b) < 0} w_m |\vec{\Omega}_m \cdot \vec{n}(\vec{r}_b)| \Psi_m^{inc} \tag{3.28}$$

$$\vec{\Upsilon}^{inc} = -\sum_{\vec{\Omega}_m \cdot \vec{n}(\vec{r}_b) < 0} 3 w_m \vec{\Omega}_m |\vec{\Omega}_m \cdot \vec{n}(\vec{r}_b)| \Psi_m^{inc} \tag{3.29}$$

The once-underlined terms in the bilinear form cancel out and the twice-underlined terms are merged together into $\left(\{\!\{ D\vec{\nabla}\Phi \cdot \vec{n} \}\!\}, [\![\Phi^*]\!] \right)_{E_h^i}$. Moving the $\vec{Q}_1$ terms into the linear functional and after some algebra, we obtain the final form of the DCF:

$$b_{DCF}(\Phi, \Phi^*) = l_{DCF}(\Phi^*) \tag{3.30}$$

where the bilinear form is

$$\begin{aligned} b_{DCF}(\Phi, \Phi^*) = & (\sigma_a \Phi, \Phi^*)_{\mathcal{D}} + \left(D\vec{\nabla}\Phi, \vec{\nabla}\Phi^* \right)_{\mathcal{D}} \\ & + \frac{1}{4} \left([\![\Phi]\!], [\![\Phi^*]\!] \right)_{E_h^i} + \left([\![\Phi]\!], \{\!\{ D\partial_n \Phi^* \}\!\} \right)_{E_h^i} + \left(\{\!\{ D\partial_n \Phi \}\!\}, [\![\Phi^*]\!] \right)_{E_h^i} \\ & + \frac{1}{4} (\Phi, \Phi^*)_{\partial\mathcal{D}^d} - \frac{1}{2} (\Phi, D\partial_n \Phi^*)_{\partial\mathcal{D}^d} - \frac{1}{2} (D\partial_n \Phi, \Phi)_{\partial\mathcal{D}^d} \\ & - \frac{9}{16} \left([\![D\vec{\nabla}\Phi]\!], [\![D\vec{\nabla}\Phi^*]\!] \right)_{E_h^i} - \frac{9}{16} \left([\![D\partial_n \Phi]\!], [\![D\partial_n \Phi^*]\!] \right)_{E_h^i} \\ & - \frac{9}{16} \left(D\vec{\nabla}\Phi, D\vec{\nabla}\Phi^* \right)_{\partial\mathcal{D}^d} - \frac{9}{16} (D\partial_n \Phi, D\partial_n \Phi^*)_{\partial\mathcal{D}^d} \\ & - \frac{9}{4} (D\partial_n \Phi, D\partial_n \Phi^*)_{\partial\mathcal{D}^r} \end{aligned} \tag{3.31}$$

and the linear form is

$$\begin{aligned} l_{DCF}(\Phi^*) = & (Q_0, \Phi^*)_{\mathcal{D}} - \left(3\vec{\nabla} \cdot D\vec{Q}_1, \Phi^* \right)_{\mathcal{D}} + \left(J^{inc}, \Phi^* \right)_{\partial\mathcal{D}^d} - \left(\vec{\Upsilon}^{inc}, D\vec{\nabla}\Phi^* \right)_{\partial\mathcal{D}^d} \\ & + \left(3\{\!\{ D\vec{Q}_1 \cdot \vec{n} \}\!\}, [\![\Phi^*]\!] \right)_{E_h^i} - \frac{1}{2} \left(3D\vec{Q}_1 \cdot \vec{n}, \Phi^* \right)_{\partial\mathcal{D}^d} \\ & - \frac{9}{16} \left([\![3D\vec{Q}_1]\!], [\![D\vec{\nabla}\Phi^*]\!] \right)_{E_h^i} - \frac{9}{16} \left([\![3D\vec{Q}_1 \cdot \vec{n}]\!], [\![D\partial_n \Phi^*]\!] \right)_{E_h^i} \\ & - \frac{9}{16} \left(3D\vec{Q}_1, D\vec{\nabla}\Phi^* \right)_{\partial\mathcal{D}^d} - \frac{9}{16} \left(3D\vec{Q}_1 \cdot \vec{n}, D\partial_n \Phi^* \right)_{\partial\mathcal{D}^d} \\ & - \frac{9}{4} \left(3D\vec{Q}_1 \cdot \vec{n}, D\partial_n \Phi^* \right)_{\partial\mathcal{D}^r} \end{aligned} \tag{3.32}$$

where the normal derivative notation

$$\partial_n \Phi = \vec{\nabla}\Phi \cdot \vec{n} \tag{3.33}$$

has been utilized to simplify the notations ($\vec{n}$ is arbitrarily fixed for any edge).

We note that the DCF is symmetric but not positive definite. Q_0 and $\vec{Q}_1$ are the volumetric source; J^{inc} and $\vec{\Upsilon}^{inc}$ are the non-homogeneous surface source. We will see in Sec. D that the significant angular flux on the reflecting boundaries will introduce a non-homogeneous surface source in the DSA calculations. Also note a non-homogeneous surface source in the DSA equations will be present on a sub-domain interface when a synchronous solve of all sub-domains is employed (domain decomposition with MPI); this term will be explained in Chapter V. Had we neglected the $\vec{Q}_1$ terms in Eq. (3.2) (Fick's law for the primal variable), all edge terms containing $\vec{Q}_1$ in the linear functional should be removed and volumetric $\vec{Q}_1$ term should be changed to $+\left(\vec{Q}_1, 3D\vec{\nabla}\Phi^*\right)_{\mathcal{D}}$.

All edge-integral terms in the DCF bilinear form are independent on the orientation of the normal unit vector $\vec{n}$ for the interior edges. This can be demonstrated as follows: if we define $\vec{\mathbf{n}}$ on a given edge of an element as the normal unit vector pointing outwards, i.e., we associate the vector $\vec{\mathbf{n}}$ with the local elements, we can easily see that:

$$\begin{aligned}
(\llbracket\Phi\rrbracket, \llbracket\Phi^*\rrbracket)_e &= (\{\!\{\vec{\mathbf{n}}\Phi\}\!\}, \{\!\{\vec{\mathbf{n}}\Phi^*\}\!\})_e \\
(\llbracket\Phi\rrbracket, \{\!\{\partial_n\Phi^*\}\!\})_e &= -(\{\!\{\vec{\mathbf{n}}\Phi\}\!\}, \{\!\{\vec{\mathbf{n}}\partial_{\mathbf{n}}\Phi^*\}\!\})_e \\
(\{\!\{\partial_n\Phi\}\!\}, \llbracket\Phi^*\rrbracket)_e &= -(\{\!\{\vec{\mathbf{n}}\partial_n\Phi\}\!\}, \{\!\{\vec{\mathbf{n}}\Phi^*\}\!\})_e \\
\left(\llbracket\vec{\nabla}\Phi\rrbracket, \llbracket\vec{\nabla}\Phi^*\rrbracket\right)_e &= \left(\{\!\{\vec{\mathbf{n}}\vec{\nabla}\Phi\}\!\}, \{\!\{\vec{\mathbf{n}}\vec{\nabla}\Phi^*\}\!\}\right)_e \\
(\llbracket\partial_n\Phi\rrbracket, \llbracket\partial_n\Phi^*\rrbracket)_e &= (\{\!\{\vec{\mathbf{n}}\partial_n\Phi\}\!\}, \{\!\{\vec{\mathbf{n}}\partial_{\mathbf{n}}\Phi^*\}\!\})_e
\end{aligned} \tag{3.34}$$

In the above expressions, swapping $\vec{n}$ for $-\vec{n}$ does not modify any of these terms.

2. The Interior Penalty (IP) Diffusion Form and a Variant of It

For comparison purposes, we write here the Interior Penalty (IP) DFE form for the following continuous diffusion problem:

$$-\vec{\nabla}\cdot D\vec{\nabla}\Phi + \sigma_a\Phi = Q_0 \qquad \text{for } \vec{r}\in\mathcal{D} \tag{3.35}$$

$$\Phi = \Phi^d \qquad \text{for } \vec{r}\in\partial\mathcal{D}^d \tag{3.36}$$

$$\partial_n\Phi = 0 \qquad \text{for } \vec{r}\in\partial\mathcal{D}^r \tag{3.37}$$

The IP formulation is one of the oldest techniques employed to solve the diffusion equation with discontinuous approximations across the mesh cells. It was first introduced by Nitsche [110] to weakly enforce Dirichlet boundary conditions on the boundary. Instead of enforcing that the approximation Φ was equal to the Dirichlet value Φ^d at any point on the boundary, Nitsche suggested to enforce the boundary condition as $\int_{\partial D^d}(\Phi-\Phi^d)v$, where v is any test function. Subsequently, by extending Nitsche's approach to all interior edges (or faces in 3D), the condition on the continuity of the approximation in between elements can be relaxed and satisfied weakly using $\int_e[\![\Phi]\!]v$, where e is an interior edge and $[\![\Phi]\!]=\Phi^+-\Phi^-$ is the inter-element jump with Φ^+, Φ^- the edge values in the two elements sharing edge e [111, 112]. The IP DFE form is given by:

$$\begin{aligned} b_{IP}(\Phi,\Phi^*) = {} & (\sigma_a\Phi,\Phi^*)_{\mathcal{D}} + \left(D\vec{\nabla}\Phi,\vec{\nabla}\Phi^*\right)_{\mathcal{D}} + \\ & \left(\kappa_e^{IP}[\![\Phi]\!],[\![\Phi^*]\!]\right)_{E_h^i} + \left([\![\Phi]\!],\{\!\{D\partial_n\Phi^*\}\!\}\right)_{E_h^i} + \left(\{\!\{D\partial_n\Phi\}\!\},[\![\Phi^*]\!]\right)_{E_h^i} \\ & + \left(2\kappa_e^{IP}\Phi,\Phi^*\right)_{\partial\mathcal{D}^d} - \left(\Phi,D\partial_n\Phi^*\right)_{\partial\mathcal{D}^d} - \left(D\partial_n\Phi,\Phi\right)_{\partial\mathcal{D}^d} \end{aligned} \tag{3.38}$$

$$l_{IP}(\Phi^*) = (Q_0,\Phi^*)_{\mathcal{D}} + \left(2\kappa_e^{IP}\Phi^d,\Phi^*\right)_{\partial\mathcal{D}^d} - \left(\Phi^d,D\partial_n\Phi^*\right)_{\partial\mathcal{D}^d} \tag{3.39}$$

where the stabilization parameter, κ_e^{IP}, is given by:

$$\begin{aligned} \kappa_e^{IP} &= \begin{cases} \frac{c(p^+)}{2}\frac{D^+}{h_\perp^+} + \frac{c(p^-)}{2}\frac{D^-}{h_\perp^-} & \text{on interior edges, i.e., } e \in E_h^i \\ c(p)\frac{D}{h_\perp} & \text{on boundary edges, i.e., } e \in \partial\mathcal{D} \end{cases} \\ \text{with } c(p) &= C\,p(p+1) \end{aligned} \tag{3.40}$$

C is a constant and should be equal to 1 when no sliver elements are present, to be safe, we use 2 [113]; p is the polynomial order; D the diffusion coefficient; $h_\perp$ is the length of the cell orthogonal to edge e, we use $h_\perp = \frac{2A}{L}$, with A the element area and L the edge length; the $+$ and $-$ signs represent the two sides of an edge. The penalty coefficient κ_e makes the bilinear form Symmetric Positive Definite (SPD). There is an extra 2 coefficient on the three Dirichlet boundary terms for optimum stabilization [113].

The non-homogeneous surface source on the Dirichlet boundary is determined by the boundary value Φ^d in the IP form, which is different from the DCF derived from the transport equation where the surface source J^{inc} and $\vec{\Upsilon}^{inc}$ are irrelevant in general. However, if we assume the Dirichlet boundary for the transport problem is non-homogeneous and isotropic, i.e.,

$$\Psi_m^{inc} = \frac{\Phi^d}{4\pi} \tag{3.41}$$

then, with the Eqs. (3.28) and (3.29), we have (using the previously mentioned quadrature properties):

$$J^{inc} = \frac{\Phi^d}{4} \tag{3.42}$$

$$\vec{\Upsilon}^{inc} = \frac{\Phi^d}{2}\vec{n}_b = 2J^{inc}\vec{n}_b \tag{3.43}$$

It is interesting that when the incoming angular flux is isotropic on the Dirichlet boundary, there are no boundary layer effects.

Finally, there are five differences between the DCF and the IP diffusion forms:

1. The penalty coefficient κ_e in the DCF form is fixed to $\frac{1}{4}$.

2. The additional factor 2 in the boundary stabilization term is absent from the DCF form.

3. There are two additional edge-terms in the DCF (terms starting with the coefficient $\frac{9}{16}$).

4. There is one additional term for the reflecting boundary (the term starting with $\frac{9}{4}$ in Eq. (3.31).)

5. There are no $\vec{Q}_1$ source terms in the IP form and the two non-homogeneous Dirichlet terms can be obtained by assuming isotropic incoming angular flux $\Psi^{inc} = \frac{1}{4\pi}\Phi^d$.

As will be clear from the DSA results, neither the DCF nor the IP form are stable for all optical thicknesses. We have devised a modified IP form, denoted by MIP, where, among other things, the stabilization parameter is a combination of the DCF and IP stabilization parameters. In detail, the MIP form can be obtained from the DCF form by:

1. Modifying the penalty coefficient.

2. Removing the boundary-stabilization factor 2 in the IP form.

3. Dropping all double-derivative terms in the bilinear form and dropping all corresponding source terms in the right hand side.

4. Dropping the $\vec{\Upsilon}^{inc}$ contribution on the right-hand-side for simplicity.

We then obtain the MIP form:

$$
\begin{aligned}
b_{MIP}(\Phi,\Phi^*) = &(\sigma_a\Phi,\Phi^*)_{\mathcal{D}} + \left(D\vec{\nabla}\Phi,\vec{\nabla}\Phi^*\right)_{\mathcal{D}} \\
&+ \left(\kappa_e^{MIP}[\![\Phi]\!],[\![\Phi^*]\!]\right)_{E_h^i} + \left([\![\Phi]\!],\{\!\{D\partial_n\Phi^*\}\!\}\right)_{E_h^i} + \left(\{\!\{D\partial_n\Phi\}\!\},[\![\Phi^*]\!]\right)_{E_h^i} \\
&+ \left(\kappa_e^{MIP}\Phi,\Phi^*\right)_{\partial\mathcal{D}^d} - \frac{1}{2}\left(\Phi,D\partial_n\Phi^*\right)_{\partial\mathcal{D}^d} - \frac{1}{2}\left(D\partial_n\Phi,\Phi^*\right)_{\partial\mathcal{D}^d} \qquad (3.44)\\
l_{MIP}(\Phi^*) = &(Q_0,\Phi^*)_{\mathcal{D}} - \left(3\vec{\nabla}\cdot D\vec{Q}_1,\Phi^*\right)_{\mathcal{D}} + \left(J^{inc},\Phi^*\right)_{\partial\mathcal{D}^d} \\
&+ \left(3\{\!\{D\vec{Q}_1\cdot\vec{n}\}\!\},[\![\Phi^*]\!]\right)_{E_h^i} - \frac{1}{2}\left(3D\vec{Q}_1\cdot\vec{n},\Phi^*\right)_{\partial\mathcal{D}^d} \qquad (3.45)
\end{aligned}
$$

with

$$
\kappa_e^{MIP} = \max\left(\kappa_e^{IP},\frac{1}{4}\right) \qquad (3.46)
$$

These modifications will be further described in Section F.

Note that the partial current on an edge e separating element K and K' is calculated as follows:

$$
\begin{aligned}
J_e^{out} &= \kappa_e^{MIP}\Phi_K - \frac{1}{2}D\partial_n\Phi_K \qquad (3.47)\\
J_e^{in} &= \kappa_e^{MIP}\Phi_{K'} - \frac{1}{2}D\partial_n\Phi_{K'} \qquad (3.48)
\end{aligned}
$$

where $\vec{n}$ is oriented from K to K'. The local balance is still preserved with the in-leakage and out-leakage calculated with the above equations. The MIP form is also SPD. Without $\vec{Q}_1$ terms in Eq. (3.2), the right-hand-side of the MIP form will not have the edge integral terms containing $\vec{Q}_1$ and the volumetric integral term should be changed to $+\left(\vec{Q}_1,3D\vec{\nabla}\Phi^*\right)_{\mathcal{D}}$.

3. The P_1 Conforming Form

In the derivation of the DCF form, we assume that (i) the primal and dual angular functions in the S_N variational form were limited to be linearly anisotropic *and* that

the currents satisfied a Fick's law. Removing the latter assumption, i.e., using,

$$\begin{aligned} \Psi_m &= \frac{1}{4\pi}(\Phi + 3\vec{J}\cdot\vec{\Omega}_m) \\ \Psi^*_m &= \frac{1}{4\pi}(\Phi^* + 3\vec{J}^*\cdot\vec{\Omega}_m) \end{aligned} \tag{3.49}$$

we can perform again a similar derivation and arrive at the following P_1 conforming (P1C) scheme:

$$b_{P1C}(\Phi, \vec{J}, \Phi^*, \vec{J}^*) = l(\Phi^*, \vec{J}^*) \tag{3.50}$$

with

$$\begin{aligned} b_{P1C}(\Phi, \vec{J}, \Phi^*, \vec{J}^*) &= (\sigma_a\Phi, \Phi^*)_{\mathcal{D}} + (3\sigma_{tr}\vec{J}, \vec{J}^*)_{\mathcal{D}} \\ &+(\vec{\nabla}\Phi, \vec{J}^*)_{\mathcal{D}} - (\vec{J}, \vec{\nabla}\Phi^*)_{\mathcal{D}} \\ &+\frac{1}{4}([\![\Phi]\!], [\![\Phi^*]\!])_{E_h^i} + ([\![\Phi]\!], \{\!\{\vec{J}^*\cdot\vec{n}\}\!\})_{E_h^i} - (\{\!\{\vec{J}\cdot\vec{n}\}\!\}, [\![\Phi^*]\!])_{E_h^i} \\ &+\frac{9}{16}([\![\vec{J}\cdot\vec{n}]\!], [\![\vec{J}^*\cdot\vec{n}]\!])_{E_h^i} + \frac{9}{16}([\![\vec{J}]\!], [\![\vec{J}^*]\!])_{E_h^i} \\ &+\frac{1}{4}(\Phi, \Phi^*)_{\partial\mathcal{D}^d} + \frac{1}{2}(\Phi, \vec{J}^*\cdot\vec{n})_{\partial\mathcal{D}^d} - \frac{1}{2}(\vec{J}\cdot\vec{n}, \Phi^*)_{\partial\mathcal{D}^d} \\ &+\frac{9}{16}(\vec{J}, \vec{J}^*)_{\partial\mathcal{D}^d} + \frac{9}{16}(\vec{J}\cdot\vec{n}, \vec{J}^*\cdot\vec{n})_{\partial\mathcal{D}^d} \\ &+\frac{9}{4}(\vec{J}\cdot\vec{n}, \vec{J}^*\cdot\vec{n})_{\partial\mathcal{D}^r} \end{aligned}$$

$$l_{P1C}(\Phi^*, \vec{J}^*) = (Q_0, \Phi^*)_{\mathcal{D}} + (3\vec{Q}_1, \vec{J}^*)_{\mathcal{D}} \tag{3.51}$$

The P1C form is PD (Positive Definite), i.e., $b_{P1C}(\Phi, \vec{J}, \Phi, \vec{J}) \geq 0$.

4. The Mixed P_1 Form from Warsa & Morel

For comparison purposes, we also provide the P_1 Mixed variational form (P1M), the so-called fully consistent DSA scheme from Warsa & Morel [70]. Starting with the

continuous P_1 equations,

$$\begin{aligned} \vec{\nabla}\Phi + 3\sigma_{tr}\vec{J} &= 3\vec{Q}_1 \\ \vec{\nabla}\cdot\vec{J} + \sigma_a\Phi &= Q_0 \end{aligned} \tag{3.52}$$

and testing it with discontinuous trial functions, we obtain

$$\begin{aligned} (\Phi^b, \vec{J}^{*,-}\cdot\vec{\mathbf{n}})_{\partial K} - (\Phi, \vec{\nabla}\cdot\vec{J}^*)_K + (3\sigma_{tr}\vec{J}, \vec{J}^*)_K &= (3\vec{Q}_1, \vec{J}^*)_K \\ (\vec{J}^b\cdot\vec{\mathbf{n}}, \Phi^{*,-})_{\partial K} - (\vec{J}, \vec{\nabla}\Phi^*)_K + (\sigma_a\Phi, \Phi^*)_K &= (Q_0, \Phi^*)_K \end{aligned} \tag{3.53}$$

where integration by parts was applied and the numerical traces are uniquely defined on the edges (here, $\vec{\mathbf{n}}$ is oriented outward locally. The numerical terms Φ^n and $\vec{J}^n$ on edges will be defined shortly. Summing over all elements K yields

$$-(\Phi^n, [\![\vec{J}^*\cdot\vec{n}]\!])_{E_h^i} + (\Phi^n, \vec{J}^*\cdot\vec{n})_{\partial\mathcal{D}} - (\Phi, \vec{\nabla}\cdot\vec{J}^*)_{\mathcal{D}} + (3\sigma_{tr}\vec{J}, \vec{J}^*)_{\mathcal{D}} = \quad (3\vec{Q}_1, \vec{J}^*)_{\mathcal{D}} \tag{3.54}$$

$$-(\vec{J}^n\cdot\vec{n}, [\![\Phi^*]\!])_{E_h^i} + (\vec{J}^n\cdot\vec{n}, \Phi^*)_{\partial\mathcal{D}} - (\vec{J}, \vec{\nabla}\Phi^*)_{\mathcal{D}} + (\sigma_a\Phi, \Phi^*)_{\mathcal{D}} = \quad (Q_0, \Phi^*)_{\mathcal{D}} \tag{3.55}$$

The numerical fluxes on interior edges are defined using two outgoing partial currents, (our notation is used, i.e., $\vec{n}$ is arbitrarily associated with edges.)

$$J^+ = \frac{1}{4}\Phi^+ - \frac{1}{2}\vec{J}^+\cdot\vec{n} \tag{3.56}$$

$$J^- = \frac{1}{4}\Phi^- + \frac{1}{2}\vec{J}^-\cdot\vec{n} \tag{3.57}$$

i.e.,

$$\begin{aligned}
\Phi^n &= 2(J^+ + J^-) = 2(\frac{1}{4}\Phi^+ - \frac{1}{2}\vec{J}^+ \cdot \vec{n} + \frac{1}{4}\Phi^- + \frac{1}{2}\vec{J}^- \cdot \vec{n}) \\
&= \{\!\{\Phi\}\!\} - [\![\vec{J} \cdot \vec{n}]\!] \qquad (3.58) \\
\vec{J}^n \cdot \vec{n} &= (J^- - J^+) = (\frac{1}{4}\Phi^- + \frac{1}{2}\vec{J}^- \cdot \vec{n} - \frac{1}{4}\Phi^+ + \frac{1}{2}\vec{J}^+ \cdot \vec{n}) \\
&= -\frac{1}{4}[\![\Phi]\!] + \{\!\{\vec{J} \cdot \vec{n}\}\!\} \qquad (3.59)
\end{aligned}$$

Note that the Φ^n does not depend on the orientation of $\vec{n}$, while $\vec{J}^n \cdot \vec{n}$ does. We now define the boundary numerical flux:

On Dirichlet boundaries $\partial\mathcal{D}^d$:

$$\begin{aligned}
\Phi^n &= 2(J^+ + J^-) = \Phi^d \qquad (3.60) \\
\vec{J}^n \cdot \vec{n} &= (J^- - J^+) = 2J^- - \frac{1}{2}\Phi^d \qquad (3.61)
\end{aligned}$$

On Neumann boundaries $\partial\mathcal{D}^n$:

$$\begin{aligned}
\Phi^n &= 2(J^+ + J^-) = 2J^{net} + 4J^- \qquad (3.62) \\
\vec{J}^n \cdot \vec{n} &= (J^- - J^+) = -J^{net} \qquad (3.63)
\end{aligned}$$

On Robin boundaries $\partial\mathcal{D}^c$:

$$\begin{aligned}
\Phi^n &= 2(J^{inc} + J^-) \qquad (3.64) \\
\vec{J}^n \cdot \vec{n} &= (J^- - J^{inc}) \qquad (3.65)
\end{aligned}$$

Substitute the numerical terms into the Eqs. (3.54) and (3.55) and sum these

two equations together, we obtain

$$
\begin{aligned}
&-\left(\{\!\{\Phi\}\!\}-[\![\vec{J}\cdot\vec{n}]\!],[\![\vec{J}^*\cdot\vec{n}]\!]\right)_{E_h^i}-\left(\Phi,\vec{\nabla}\cdot\vec{J}^*\right)_{\mathcal{D}}+\left(3\sigma_{tr}\vec{J},\vec{J}^*\right)_{\mathcal{D}}\\
&\quad-\left(-\frac{1}{4}[\![\Phi]\!]+\{\!\{\vec{J}\cdot\vec{n}\}\!\},[\![\Phi^*]\!]\right)_{E_h^i}-\left(\vec{J},\vec{\nabla}\Phi^*\right)_{\mathcal{D}}+(\sigma_a\Phi,\Phi^*)_{\mathcal{D}}\\
&\quad+\left(\Phi_{\mathcal{D}},\vec{J}^*\cdot\vec{n}\right)_{\partial\mathcal{D}^d}+\left(\frac{1}{2}\Phi+\vec{J}\cdot\vec{n}-\frac{1}{2}\Phi_{\mathcal{D}},\Phi^*\right)_{\partial\mathcal{D}^d}\\
&\quad+\left(2J^{net}+\Phi+2\vec{J}\cdot\vec{n},\vec{J}^*\cdot\vec{n}\right)_{\partial\mathcal{D}^n}-\left(J^{net},\Phi^*\right)_{\partial\mathcal{D}^n}\\
&\quad+\left(2J^{inc}+\frac{1}{2}\Phi+\vec{J}\cdot\vec{n},\vec{J}^*\cdot\vec{n}\right)_{\partial\mathcal{D}^c}+\left(\frac{1}{4}\Phi^-+\frac{1}{2}\vec{J}^-\cdot\vec{n}-J^{inc},\Phi^*\right)_{\partial\mathcal{D}^c}\\
&\qquad=(Q_0,\Phi^*)_{\mathcal{D}}+\left(3\vec{Q}_1,\vec{J}^*\right)_{\mathcal{D}}
\end{aligned}
\tag{3.66}
$$

Integrating by parts the term $-(\Phi,\vec{\nabla}\cdot\vec{J}^*)_{\mathcal{D}}$, we obtain, after some simplifications, the variational form

$$
\begin{aligned}
b_{P1M}(\Phi,\vec{J},\Phi^*,\vec{J}^*) &= \left(3\sigma_{tr}\vec{J},\vec{J}^*\right)_{\mathcal{D}}-\left(\vec{J},\vec{\nabla}\Phi^*\right)_{\mathcal{D}}+(\sigma_a\Phi,\Phi^*)_{\mathcal{D}}+\left(\vec{\nabla}\Phi,\vec{J}^*\right)_{\mathcal{D}}\\
&\quad+\frac{1}{4}\left([\![\Phi]\!],[\![\Phi^*]\!]\right)_{E_h^i}-\left(\{\!\{\vec{J}\cdot\vec{n}\}\!\},[\![\Phi^*]\!]\right)_{E_h^i}\\
&\quad+\left([\![\Phi]\!],\{\!\{\vec{J}^*\cdot\vec{n}\}\!\}\right)_{E_h^i}+\left([\![\vec{J}\cdot\vec{n}]\!],[\![\vec{J}^*\cdot\vec{n}]\!]\right)_{E_h^i}\\
&\quad+\frac{1}{2}\left(\Phi,\Phi^*\right)_{\partial\mathcal{D}^d}+\left(\vec{J}\cdot\vec{n},\Phi^*\right)_{\partial\mathcal{D}^d}-\left(\Phi,\vec{J}^*\cdot\vec{n}\right)_{\partial\mathcal{D}^d}\\
&\quad+2\left(\vec{J}\cdot\vec{n},\vec{J}^*\cdot\vec{n}\right)_{\partial\mathcal{D}^n}\\
&\quad+\frac{1}{4}\left(\Phi,\Phi^*\right)_{\partial\mathcal{D}^c}+\frac{1}{2}\left(\vec{J}\cdot\vec{n},\Phi^*\right)_{\partial\mathcal{D}^c}-\frac{1}{2}\left(\Phi,\vec{J}^*\cdot\vec{n}\right)_{\partial\mathcal{D}^c}\\
&\quad+\left(\vec{J}\cdot\vec{n},\vec{J}^*\cdot\vec{n}\right)_{\partial\mathcal{D}^c}\\
l_{P1M}(\Phi^*,\vec{J}^*) &= \left(Q_0,\Phi^*\right)_{\mathcal{D}}+\left(3\vec{Q}_1,\vec{J}^*\right)_{\mathcal{D}}\\
&\quad+\frac{1}{2}\left(\Phi_{\mathcal{D}},\Phi^*\right)_{\partial\mathcal{D}^d}-\left(\Phi_{\mathcal{D}},\vec{J}^*\cdot\vec{n}\right)_{\partial\mathcal{D}^d}\\
&\quad+4\left(J^{net},\frac{1}{4}\Phi^*-\frac{1}{2}\vec{J}^*\cdot\vec{n}\right)_{\partial\mathcal{D}^n}\\
&\quad+4\left(J^{inc},\frac{1}{4}\Phi^*-\frac{1}{2}\vec{J}^*\cdot\vec{n}\right)_{\partial\mathcal{D}^c}
\end{aligned}
\tag{3.67}
$$

P1M is also PD, which can be easily seen with $b_{P1M}(\Phi, \vec{J}, \Phi, \vec{J}) \geq 0$. The P1C and P1M forms present a few differences related to edge integrals. On interior edges, there is an additional term in the P1C form along the x- and y- directions. Also, the coefficients for these current terms are $\frac{9}{16}$ in the P1C form, whereas they are equal to 1 in the P1M form. Finally, current terms are present on vacuum boundaries in the P1C form but are absent in the P1M. The current coefficients for reflecting boundaries are $\frac{9}{4}$ in P1C while they are equal to 2 in P1M. Even though the two forms are relatively similar, significant differences in the spectral radius can be observed for highly linearly anisotropic scattering problems, see the results in Section F.

So far, we have describe five different forms for the DFE discretization of the diffusion equation:

1. the standard IP form used in the numerical analysis community;
2. the DCF form, derived from the DFE S_N form using a restriction linear in angle and Fick's law for the primal and dual functions;
3. a modified IP (MIP) form;
4. a P1C form, derived from the DFE S_N form using a restriction linear in angle;
5. a P1M form, derived from the continuous P_1 equations.

Unlike the variational form of the transport equation, all these diffusion forms can be directly used for coding. It also needs to be pointed out that all forms work on irregular meshes, i.e., meshes with hanging nodes.

C. Higher-Order Discontinuous Finite Element Method for the Diffusion Problem

1. Local Matrices

Having defining the various diffusion forms, we provide now the local matrices for these forms. The mass matrix and type-1 edge matrix are identical to the one used in the DFE S_N form and will not be repeated here. We use the same higher-order shape functions as in the case of the DFE S_N form.

a. Stiffness Matrix

We define the stiffness matrix for the term $(D\vec{\nabla}\Phi, \vec{\nabla}\Phi^*)_K$:

$$\mathbf{S} = \int_K \nabla_{\mathbf{x}}\mathbf{b}\nabla_{\mathbf{x}}\mathbf{b}^T \,\mathrm{d}x\,\mathrm{d}y \tag{3.68}$$

Using a change of variables to map onto the reference element, we obtain

$$\mathbf{S} = \int_{\widehat{K}} \nabla_{\xi}\widehat{\mathbf{b}} \left[J^{-1}J^{-T}\det(J)\right] \nabla_{\xi}\widehat{\mathbf{b}}^T \,\mathrm{d}\xi_1\,\mathrm{d}\xi_2 \tag{3.69}$$

Let us first simplify the $\left[J^{-1}J^{-T}\det(J)\right]$ term. We define the following ratio for any of the three edges of a triangle

$$r_i = \frac{L_i^2}{4A}, \quad i = 1, 2, 3. \tag{3.70}$$

Note that the three ratios are not independent; they abide to the following equality

$$2r_1r_2 + 2r_1r_3 + 2r_2r_3 - r_1^2 - r_2^2 - r_3^2 \equiv 1 \tag{3.71}$$

We then have

$$J^{-1}J^{-T}\det(J) = \begin{bmatrix} 2r_2 & r_1 - r_2 - r_3 \\ r_1 - r_2 - r_3 & 2r_3 \end{bmatrix} \tag{3.72}$$

and it is easy to prove that

$$\begin{aligned} \cot\alpha_1 &= -r_1 + r_2 + r_3 \\ \cot\alpha_2 &= -r_2 + r_1 + r_3 \\ \cot\alpha_3 &= -r_3 + r_1 + r_2 \end{aligned} \tag{3.73}$$

(We want to avoid sliver elements, where the angles are close to zero or π. In case, these ratios and cot-values could be very large.)

Finally, we have for the stiffness matrix (given below for $p = 2$)

$$\mathbf{S} = \frac{1}{2}\left[\begin{array}{ccc|ccc} 2r_1 & -\cot\alpha_3 & -\cot\alpha_2 & \frac{2r_1}{\sqrt{6}} & -\frac{\cot\alpha_3}{\sqrt{6}} & -\frac{\cot\alpha_2}{\sqrt{6}} \\ -\cot\alpha_3 & 2r_2 & -\cot\alpha_1 & -\frac{\cot\alpha_3}{\sqrt{6}} & \frac{2r_2}{\sqrt{6}} & -\frac{\cot\alpha_1}{\sqrt{6}} \\ -\cot\alpha_2 & -\cot\alpha_1 & 2r_3 & -\frac{\cot\alpha_2}{\sqrt{6}} & -\frac{\cot\alpha_1}{\sqrt{6}} & \frac{2r_3}{\sqrt{6}} \\ \hline \frac{2r_1}{\sqrt{6}} & -\frac{\cot\alpha_3}{\sqrt{6}} & -\frac{\cot\alpha_2}{\sqrt{6}} & r_1+r_2+r_3 & -\cot\alpha_3 & -\cot\alpha_2 \\ -\frac{\cot\alpha_3}{\sqrt{6}} & \frac{2r_2}{\sqrt{6}} & -\frac{\cot\alpha_1}{\sqrt{6}} & -\cot\alpha_3 & r_1+r_2+r_3 & -\cot\alpha_1 \\ -\frac{\cot\alpha_2}{\sqrt{6}} & -\frac{\cot\alpha_1}{\sqrt{6}} & \frac{2r_3}{\sqrt{6}} & -\cot\alpha_2 & -\cot\alpha_1 & r_1+r_2+r_3 \end{array}\right]$$

Note that local stiffness matrix is symmetric, dimensionless and singular. The local stiffness matrix is determined by the above-defined ratio for each element K. A bold letter is used to indicate this element-independence. As an example, when $r_1 = r_2 = r_3 = \frac{1}{\sqrt{3}}$ (i.e., equilateral triangle),

$$\mathbf{S} = \frac{1}{2\sqrt{3}}\begin{bmatrix} 2 & -1 & -1 & \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ -1 & 2 & -1 & -\frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ -1 & -1 & 2 & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} \\ \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & 3 & -1 & -1 \\ -\frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & -1 & 3 & -1 \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & -1 & -1 & 3 \end{bmatrix}$$

b. Type-2 Edge Matrix

We have presented the type-1 edge matrix in Chapter II to assemble the local transport problem. This type-1 edge matrix can also be used for the term $(\llbracket\Phi\rrbracket, \llbracket\Phi^*\rrbracket)_e$ in the diffusion system.

For the terms $(\llbracket\Phi\rrbracket, \{\!\{D\partial_n\Phi^*\}\!\})_e$ and $(\{\!\{D\partial_n\Phi\}\!\}, \llbracket\Phi^*\rrbracket)_e$, we need to define an additional edge matrix, the type-2 edge matrix:

$$\begin{aligned}\mathbf{E}_i^2 &= \int_{\partial K_i} (\nabla_{\mathbf{x}}\mathbf{b})\vec{n}_i\mathbf{b}^T \,\mathrm{d}s, \qquad i = 1,2,3 \\ &= \int_{-1}^{1} \nabla_\xi\hat{\mathbf{b}}(\xi_i)\left[J^{-1}\vec{n}_i\frac{L_i}{2}\right]\hat{\mathbf{b}}^T(\xi_i)\,\mathrm{d}s\end{aligned} \tag{3.74}$$

We note that

$$\begin{aligned}\frac{J^{-1}\vec{n}_1L_1}{2} &= \begin{bmatrix}\cot\alpha_3\\ \cot\alpha_2\end{bmatrix} \\ \frac{J^{-1}\vec{n}_2L_2}{2} &= \begin{bmatrix}-2r_2\\ \cot\alpha_1\end{bmatrix} \\ \frac{J^{-1}\vec{n}_3L_3}{2} &= \begin{bmatrix}\cot\alpha_1\\ -2r_3\end{bmatrix},\end{aligned} \tag{3.75}$$

and the three type-2 edge matrices (one matrix per edge) are (for $p = 2$)

$$\mathbf{E}_1^2 = \left[\begin{array}{ccc|ccc} 0 & -r_1 & -r_1 & \frac{2r_1}{\sqrt{6}} & 0 & 0 \\ 0 & \frac{\cot\alpha_3}{2} & \frac{\cot\alpha_3}{2} & -\frac{\cot\alpha_3}{\sqrt{6}} & 0 & 0 \\ 0 & \frac{\cot\alpha_2}{2} & \frac{\cot\alpha_2}{2} & -\frac{\cot\alpha_2}{\sqrt{6}} & 0 & 0 \\ \hline 0 & -\frac{2r_1+\cot\alpha_2}{\sqrt{6}} & -\frac{2r_1+\cot\alpha_3}{\sqrt{6}} & r_1 & 0 & 0 \\ 0 & \frac{2r_1}{\sqrt{6}} & \frac{4r_1}{\sqrt{6}} & -r_1 & 0 & 0 \\ 0 & \frac{4r_1}{\sqrt{6}} & \frac{2r_1}{\sqrt{6}} & -r_1 & 0 & 0 \end{array}\right]$$

$$\mathbf{E}_2^2 = \left[\begin{array}{ccc|ccc} \frac{\cot\alpha_3}{2} & 0 & \frac{\cot\alpha_3}{2} & 0 & -\frac{\cot\alpha_3}{\sqrt{6}} & 0 \\ -r_2 & 0 & -r_2 & 0 & \frac{2r_2}{\sqrt{6}} & 0 \\ \frac{\cot\alpha_1}{2} & 0 & \frac{\cot\alpha_1}{2} & 0 & -\frac{\cot\alpha_1}{\sqrt{6}} & 0 \\ \hline \frac{2r_2}{\sqrt{6}} & 0 & \frac{4r_2}{\sqrt{6}} & 0 & -r_2 & 0 \\ -\frac{2r_2+\cot\alpha_1}{\sqrt{6}} & 0 & -\frac{2r_2+\cot\alpha_3}{\sqrt{6}} & 0 & r_2 & 0 \\ \frac{4r_2}{\sqrt{6}} & 0 & \frac{2r_2}{\sqrt{6}} & 0 & -r_2 & 0 \end{array}\right]$$

$$\mathbf{E}_3^2 = \left[\begin{array}{ccc|ccc} \frac{\cot\alpha_2}{2} & \frac{\cot\alpha_2}{2} & 0 & 0 & 0 & -\frac{\cot\alpha_2}{\sqrt{6}} \\ \frac{\cot\alpha_1}{2} & \frac{\cot\alpha_1}{2} & 0 & 0 & 0 & -\frac{\cot\alpha_1}{\sqrt{6}} \\ -r_3 & -r_3 & 0 & 0 & 0 & \frac{2r_3}{\sqrt{6}} \\ \hline \frac{2r_3}{\sqrt{6}} & \frac{4r_3}{\sqrt{6}} & 0 & 0 & 0 & -r_3 \\ \frac{4r_3}{\sqrt{6}} & \frac{2r_3}{\sqrt{6}} & 0 & 0 & 0 & -r_3 \\ -\frac{2r_3+\cot\alpha_1}{\sqrt{6}} & -\frac{2r_3+\cot\alpha_2}{\sqrt{6}} & 0 & 0 & 0 & r_3 \end{array}\right]$$

The type-2 edge matrices E_i^2 $(i = 1, 2, 3)$ are not symmetric. All ratios r_i are evaluated on the local element K.

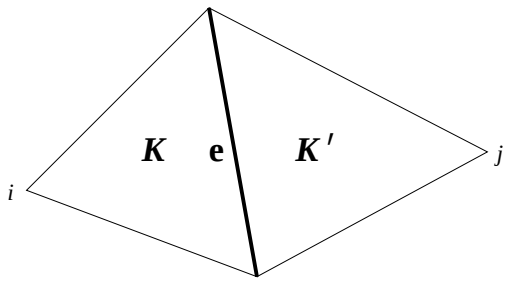


Fig. III-1. Interior edge definition: recall that for element K, the local edge ID is i because its opposite vertex is labeled i.

The type-2 edge coupling matrix for an interior edge showed in the Fig. III-1 is

defined as

$$\mathbf{E}^{2C}_{K,i,j} = \int_{-1}^{1} \nabla_\xi \hat{\mathbf{b}}(\xi_i) \left[\frac{(J^{-1}\vec{n}_i)_K L_i}{2} \right] \hat{\mathbf{b}}(-\xi_j)\, \mathrm{d}s. \tag{3.76}$$

$J^{-1}\vec{n}_i$ is evaluated on element K and not its neighbor K', which is why we give the subscript K in $\mathbf{E}^{2C}_{K,i,j}$. This coupling matrix $\mathbf{E}^{2C}_{K,i,j}$ operates on the solution vector of the neighboring element K'.

$$\mathbf{E}^{2C}_{K,1,1} = \left[\begin{array}{ccc|ccc} 0 & -r_1 & -r_1 & \frac{2r_1}{\sqrt{6}} & 0 & 0 \\ 0 & \frac{\cot\alpha_3}{2} & \frac{\cot\alpha_3}{2} & -\frac{\cot\alpha_3}{\sqrt{6}} & 0 & 0 \\ 0 & \frac{\cot\alpha_2}{2} & \frac{\cot\alpha_2}{2} & -\frac{\cot\alpha_2}{\sqrt{6}} & 0 & 0 \\ \hline 0 & \underline{-\frac{2r_1+\cot\alpha_3}{\sqrt{6}}} & \underline{-\frac{2r_1+\cot\alpha_2}{\sqrt{6}}} & r_1 & 0 & 0 \\ 0 & \underline{\frac{4r_1}{\sqrt{6}}} & \underline{\frac{2r_1}{\sqrt{6}}} & -r_1 & 0 & 0 \\ 0 & \underline{\frac{2r_1}{\sqrt{6}}} & \underline{\frac{4r_1}{\sqrt{6}}} & -r_1 & 0 & 0 \end{array}\right]$$

All entries that are different from those of $\mathbf{E}^2_1$ are underlined. Basically, the second and the third columns of $\mathbf{E}^{2C}_{K,1,1}$ are swapped. Also note that all ratios r_i are evaluated on the local element K.

Suppose that the local edge ID in element K' is changing from 1 to 2, the coupling matrix is

$$\mathbf{E}^{2C}_{K,1,2} = \left[\begin{array}{ccc|ccc} -r_1 & 0 & -r_1 & 0 & \frac{2r_1}{\sqrt{6}} & 0 \\ \frac{\cot\alpha_3}{2} & 0 & \frac{\cot\alpha_3}{2} & 0 & -\frac{\cot\alpha_3}{\sqrt{6}} & 0 \\ \frac{\cot\alpha_2}{2} & 0 & \frac{\cot\alpha_2}{2} & 0 & -\frac{\cot\alpha_2}{\sqrt{6}} & 0 \\ \hline -\frac{2r_1+\cot\alpha_2}{\sqrt{6}} & 0 & -\frac{2r_1+\cot\alpha_3}{\sqrt{6}} & 0 & r_1 & 0 \\ \frac{2r_1}{\sqrt{6}} & 0 & \frac{4r_1}{\sqrt{6}} & 0 & -r_1 & 0 \\ \frac{4r_1}{\sqrt{6}} & 0 & \frac{2r_1}{\sqrt{6}} & 0 & -r_1 & 0 \end{array}\right],$$

which simply requires to swap columns 2, 3 and 4 of $E^{2C}_{K,1,1}$ to columns 3, 1 and 5.

On the other hand, if the local edge ID of the element K is changing from 1 to

2, the coupling matrix is

$$\mathbf{E}^{2C}_{K,2,1} = \left[\begin{array}{ccc|ccc} 0 & \frac{\cot\alpha_3}{2} & \frac{\cot\alpha_3}{2} & -\frac{\cot\alpha_3}{\sqrt{6}} & 0 & 0 \\ 0 & -r_2 & -r_2 & \frac{2r_2}{\sqrt{6}} & 0 & 0 \\ 0 & \frac{\cot\alpha_1}{2} & \frac{\cot\alpha_1}{2} & -\frac{\cot\alpha_1}{\sqrt{6}} & 0 & 0 \\ \hline 0 & \frac{2r_2}{\sqrt{6}} & \frac{4r_2}{\sqrt{6}} & -r_2 & 0 & 0 \\ 0 & -\frac{2r_2+\cot\alpha_1}{\sqrt{6}} & -\frac{2r_2+\cot\alpha_3}{\sqrt{6}} & r_2 & 0 & 0 \\ 0 & \frac{4r_2}{\sqrt{6}} & \frac{2r_2}{\sqrt{6}} & -r_2 & 0 & 0 \end{array}\right],$$

this requires a swap from columns 3, 1 and 5 of $E^{2C}_{K,2,2}$ to columns 2, 3 and 4.

As an example, let us consider an interior edge showed in Fig. III-1.

$$\begin{aligned} (\llbracket\Phi\rrbracket, \{\!\{D\partial_n\Phi^*\}\!\})_e &= -(\Phi_{K'} - \Phi_K, [D\partial_{\mathbf{n}}\Phi^*]_{K'} - [D\partial_{\mathbf{n}}\Phi^*]_K)_e \\ &= -(\mathbf{\Phi}^{*T} D\mathbf{E}^2_i\mathbf{\Phi})_K - (\mathbf{\Phi}^{*T} D\mathbf{E}^2_j\mathbf{\Phi})_{K'} \\ &\quad + (\mathbf{\Phi}^{*T} D\mathbf{E}^{2C}_{i,j})_K\mathbf{\Phi}_{K'} + (\mathbf{\Phi}^{*T} D\mathbf{E}^{2C}_{j,i})_{K'}\mathbf{\Phi}_K \qquad (3.77) \\ (\{\!\{D\partial_n\Phi\}\!\}, \llbracket\Phi^*\rrbracket)_e &= -([D\partial_{\mathbf{n}}\Phi]_{K'} - [D\partial_{\mathbf{n}}\Phi]_K, \Phi^*_{K'} - \Phi^*_K)_e \\ &= -(\mathbf{\Phi}^{*T} D\mathbf{E}^{2T}_i\mathbf{\Phi})_K - (\mathbf{\Phi}^{*T} D\mathbf{E}^{2T}_j\mathbf{\Phi})_{K'} \\ &\quad + \mathbf{\Phi}^{*T}_{K'}(D\mathbf{E}^{2C,T}_{i,j}\mathbf{\Phi})_K + \mathbf{\Phi}^{*T}_K(D\mathbf{E}^{2C,T}_{j,i}\mathbf{\Phi})_{K'} \qquad (3.78) \end{aligned}$$

If we add these two terms together, we will have four block matrices corresponding to (K,K), (K,K'), (K',K) and (K',K'). It is easy to see that the block matrices for (K,K) and (K',K') are symmetric and the matrix of (K,K') is the transpose of the matrix of (K',K).

c. Type-3 Edge Matrices

For the term $([\![D\partial_n\Phi]\!],[\![D\partial_n\Phi^*]\!])_e$, we need yet an additional edge matrix (type-3), defined as follows

$$\begin{aligned}\mathbf{E}_i^3 &= \frac{L_i}{2}\int_{\partial K_i}(\nabla_{\mathbf{x}}\mathbf{b}\vec{n}_i)(\nabla_{\mathbf{x}}\mathbf{b}\vec{n}_i)^T\,\mathrm{d}s, \qquad i=1,2,3\\ &= \int_{-1}^{1}\nabla_\xi\hat{\mathbf{b}}(\xi_i)\left[\frac{J^{-1}\vec{n}_iL_i}{2}\left(\frac{J^{-1}\vec{n}_iL_i}{2}\right)^T\right]\nabla_\xi\hat{\mathbf{b}}^T(\xi_i)\,\mathrm{d}s\end{aligned} \tag{3.79}$$

The three local edge matrices are (for $p=2$)

$$\mathbf{E}_1^3=\left[\begin{array}{ccc|ccc}
2r_1^2 & -r_1\cot\alpha_3 & -r_1\cot\alpha_2 & \sqrt{6}r_1^2 & -\sqrt{6}r_1^2 & -\sqrt{6}r_1^2\\
-r_1\cot\alpha_3 & \frac{\cot^2\alpha_3}{2} & \frac{\cot\alpha_3\cot\alpha_2}{2} & -\frac{\sqrt{6}r_1\cot\alpha_3}{2} & \frac{\sqrt{6}r_1\cot\alpha_3}{2} & \frac{\sqrt{6}r_1\cot\alpha_3}{2}\\
-r_1\cot\alpha_2 & \frac{\cot\alpha_3\cot\alpha_2}{2} & \frac{\cot^2\alpha_2}{2} & -\frac{\sqrt{6}r_1\cot\alpha_2}{2} & \frac{\sqrt{6}r_1\cot\alpha_2}{2} & \frac{\sqrt{6}r_1\cot\alpha_2}{2}\\
\hline
\sqrt{6}r_1^2 & -\frac{\sqrt{6}r_1\cot\alpha_3}{2} & -\frac{\sqrt{6}r_1\cot\alpha_2}{2} & 4r_1^2-\cot\alpha_2\cot\alpha_3 & r_1\cot\alpha_2-4r_1^2 & r_1\cot\alpha_3-4r_1^2\\
-\sqrt{6}r_1^2 & \frac{\sqrt{6}r_1\cot\alpha_3}{2} & \frac{\sqrt{6}r_1\cot\alpha_2}{2} & r_1\cot\alpha_2-4r_1^2 & 4r_1^2 & 2r_1^2\\
-\sqrt{6}r_1^2 & \frac{\sqrt{6}r_1\cot\alpha_3}{2} & \frac{\sqrt{6}r_1\cot\alpha_2}{2} & r_1\cot\alpha_3-4r_1^2 & 2r_1^2 & 4r_1^2
\end{array}\right]$$

$$\mathbf{E}_2^3=\left[\begin{array}{ccc|ccc}
\frac{\cot^2\alpha_3}{2} & -r_2\cot\alpha_3 & \frac{\cot\alpha_1\cot\alpha_3}{2} & \frac{\sqrt{6}r_2\cot\alpha_3}{2} & -\frac{\sqrt{6}r_2\cot\alpha_3}{2} & \frac{\sqrt{6}r_2\cot\alpha_3}{2}\\
-r_2\cot\alpha_3 & 2r_2^2 & -r_2\cot\alpha_1 & -\sqrt{6}r_2^2 & \sqrt{6}r_2^2 & -\sqrt{6}r_2^2\\
\frac{\cot\alpha_1\cot\alpha_3}{2} & -r_2\cot\alpha_1 & \frac{\cot^2\alpha_1}{2} & \frac{\sqrt{6}r_2\cot\alpha_1}{2} & -\frac{\sqrt{6}r_2\cot\alpha_1}{2} & \frac{\sqrt{6}r_2\cot\alpha_1}{2}\\
\hline
\frac{\sqrt{6}r_2\cot\alpha_3}{2} & -\sqrt{6}r_2^2 & \frac{\sqrt{6}r_2\cot\alpha_1}{2} & 4r_2^2 & r_2\cot\alpha_1-4r_2^2 & 2r_2^2\\
-\frac{\sqrt{6}r_2\cot\alpha_3}{2} & \sqrt{6}r_2^2 & -\frac{\sqrt{6}r_2\cot\alpha_1}{2} & r_2\cot\alpha_1-4r_2^2 & 4r_2^2-\cot\alpha_1\cot\alpha_3 & r_2\cot\alpha_3-4r_2^2\\
\frac{\sqrt{6}r_2\cot\alpha_3}{2} & -\sqrt{6}r_2^2 & \frac{\sqrt{6}r_2\cot\alpha_1}{2} & 2r_2^2 & r_2\cot\alpha_3-4r_2^2 & 4r_2^2
\end{array}\right]$$

$$\mathbf{E}_3^3=\left[\begin{array}{ccc|ccc}
\frac{\cot^2\alpha_2}{2} & \frac{\cot\alpha_1\cot\alpha_2}{2} & -r_3\cot\alpha_2 & \frac{\sqrt{6}r_3\cot\alpha_2}{2} & -\frac{\sqrt{6}r_3\cot\alpha_2}{2} & \frac{\sqrt{6}r_3\cot\alpha_2}{2}\\
\frac{\cot\alpha_1\cot\alpha_2}{2} & \frac{\cot^2\alpha_1}{2} & -r_3\cot\alpha_1 & \frac{\sqrt{6}r_3\cot\alpha_1}{2} & \frac{\sqrt{6}r_3\cot\alpha_1}{2} & -\frac{\sqrt{6}r_3\cot\alpha_1}{2}\\
-r_3\cot\alpha_2 & -r_3\cot\alpha_1 & 2r_3^2 & -\sqrt{6}r_3^2 & -\sqrt{6}r_3^2 & \sqrt{6}r_3^2\\
\hline
\frac{\sqrt{6}r_3\cot\alpha_2}{2} & \frac{\sqrt{6}r_3\cot\alpha_1}{2} & -\sqrt{6}r_3^2 & 4r_3^2 & 2r_3^2 & r_3\cot\alpha_1-4r_3^2\\
\frac{\sqrt{6}r_3\cot\alpha_2}{2} & \frac{\sqrt{6}r_3\cot\alpha_1}{2} & -\sqrt{6}r_3^2 & 2r_3^2 & 4r_3^2 & r_3\cot\alpha_2-4r_3^2\\
-\frac{\sqrt{6}r_3\cot\alpha_2}{2} & -\frac{\sqrt{6}r_3\cot\alpha_1}{2} & \sqrt{6}r_3^2 & r_3\cot\alpha_1-4r_3^2 & r_3\cot\alpha_2-4r_3^2 & 4r_3^2-\cot\alpha_1\cot\alpha_2
\end{array}\right]$$

As an example, when $r_1=r_2=r_3=\frac{1}{\sqrt{3}}$, we obtain

$$\mathbf{E}_1^3+\mathbf{E}_2^3+\mathbf{E}_3^3=\begin{bmatrix}
1 & -\frac{1}{2} & -\frac{1}{2} & \frac{4}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}}\\
-\frac{1}{2} & 1 & -\frac{1}{2} & -\frac{2}{\sqrt{6}} & \frac{4}{\sqrt{6}} & -\frac{2}{\sqrt{6}}\\
-\frac{1}{2} & -\frac{1}{2} & 1 & -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{4}{\sqrt{6}}\\
\frac{4}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{11}{3} & -\frac{4}{3} & -\frac{4}{3}\\
-\frac{2}{\sqrt{6}} & \frac{4}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{4}{3} & \frac{11}{3} & -\frac{4}{3}\\
-\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{4}{\sqrt{6}} & -\frac{4}{3} & -\frac{4}{3} & \frac{11}{3}
\end{bmatrix}$$

The type-3 edge matrix is symmetric and positive definite.

The type-3 coupling matrix is

$$\begin{aligned}\mathbf{E}^{3C}_{K,i,j} &= \frac{L_i}{2}\int_{\partial K_i}(\nabla_{\mathbf{x}}\mathbf{b}\vec{n}_i)_K(\nabla_{\mathbf{x}}\mathbf{b}\vec{n}_j)^T_{K'}\,\mathrm{d}s, \qquad i=1,2,3;\ j=1,2,3\\ &= \int_{-1}^{1}\nabla_\xi\hat{\mathbf{b}}(\xi_i)\left[\frac{(J^{-1}\vec{n}_i)_K L_i}{2}\left(\frac{(J^{-1}\vec{n}_j)_{K'}L_i}{2}\right)^T\right]\nabla_\xi\hat{\mathbf{b}}^T(-\xi_j)\,\mathrm{d}s\end{aligned} \tag{3.80}$$

For example, we give $\mathbf{E}^{3C}_{K,1,1}$,

$$\left[\begin{array}{ccc|ccc}
2r_1r_1' & -r_1\cot\alpha_3' & -r_1\cot\alpha_2' & \sqrt{6}r_1r_1' & -\sqrt{6}r_1r_1' & -\sqrt{6}r_1r_1' \\
-\cot\alpha_3 r_1' & \frac{\cot\alpha_3\cot\alpha_3'}{2} & \frac{\cot\alpha_3\cot\alpha_2'}{2} & -\frac{\sqrt{6}\cot\alpha_3 r_1'}{2} & \frac{\sqrt{6}\cot\alpha_3 r_1'}{2} & \frac{\sqrt{6}\cot\alpha_3 r_1'}{2} \\
-\cot\alpha_2 r_1' & \frac{\cot\alpha_2\cot\alpha_3'}{2} & \frac{\cot\alpha_2\cot\alpha_2'}{2} & -\frac{\sqrt{6}\cot\alpha_2 r_1'}{2} & \frac{\sqrt{6}\cot\alpha_2 r_1'}{2} & \frac{\sqrt{6}\cot\alpha_2 r_1'}{2} \\
\hline
\sqrt{6}r_1r_1' & -\frac{\sqrt{6}r_1\cot\alpha_3'}{2} & -\frac{\sqrt{6}r_1\cot\alpha_2'}{2} & 3r_1r_1'-(r_2-r_3)(r_2'-r_3') & (\cot\alpha_3-4r_1)r_1' & (\cot\alpha_2-4r_1)r_1' \\
-\sqrt{6}r_1r_1' & \frac{\sqrt{6}r_1\cot\alpha_3'}{2} & \frac{\sqrt{6}r_1\cot\alpha_2'}{2} & r_1(\cot\alpha_3'-4r_1') & 2r_1r_1' & 4r_1r_1' \\
-\sqrt{6}r_1r_1' & \frac{\sqrt{6}r_1\cot\alpha_3'}{2} & \frac{\sqrt{6}r_1\cot\alpha_2'}{2} & r_1(\cot\alpha_2'-4r_1') & 4r_1r_1' & 2r_1r_1'
\end{array}\right]$$

We can see that this coupling matrix differs from the $\mathbf{E}^3_1$ matrix not only in the $'$ terms evaluated with data from K' but also the last two rows and last two columns are interchanged. Note that $\mathbf{E}^{3C}_{K,1,1}$ is not equal to $\mathbf{E}^{3C}_{K',1,1}$.

To see how indices i and j affect this matrix definition, let us see $\mathbf{E}^{3C}_{K,1,2}$ first,

$$\left[\begin{array}{ccc|ccc}
-r_1\cot\alpha_3' & 2r_1r_2' & -r_1\cot\alpha_1' & -\sqrt{6}r_1r_2' & \sqrt{6}r_1r_2' & -\sqrt{6}r_1r_2' \\
\frac{\cot\alpha_3\cot\alpha_3'}{2} & -\cot\alpha_3 r_2' & \frac{\cot\alpha_3\cot\alpha_1'}{2} & \frac{\sqrt{6}\cot\alpha_3 r_2'}{2} & -\frac{\sqrt{6}\cot\alpha_3 r_2'}{2} & \frac{\sqrt{6}\cot\alpha_3 r_2'}{2} \\
\frac{\cot\alpha_2\cot\alpha_3'}{2} & -\cot\alpha_2 r_2' & \frac{\cot\alpha_2\cot\alpha_1'}{2} & \frac{\sqrt{6}\cot\alpha_2 r_2'}{2} & -\frac{\sqrt{6}\cot\alpha_2 r_2'}{2} & \frac{\sqrt{6}\cot\alpha_2 r_2'}{2} \\
\hline
-\frac{\sqrt{6}r_1\cot\alpha_3'}{2} & \sqrt{6}r_1r_2' & -\frac{\sqrt{6}r_1\cot\alpha_1'}{2} & (\cot\alpha_2-4r_1)r_2' & 3r_1r_2'-(r_2-r_3)(r_3'-r_1') & (\cot\alpha_3-4r_1)r_2' \\
\frac{\sqrt{6}r_1\cot\alpha_3'}{2} & -\sqrt{6}r_1r_2' & \frac{\sqrt{6}r_1\cot\alpha_1'}{2} & 4r_1r_2' & r_1(\cot\alpha_1'-4r_2') & 2r_1r_2' \\
\frac{\sqrt{6}r_1\cot\alpha_3'}{2} & -\sqrt{6}r_1r_2' & \frac{\sqrt{6}r_1\cot\alpha_1'}{2} & 2r_1r_2' & r_1(\cot\alpha_3'-4r_2') & 4r_1r_2'
\end{array}\right],$$

which is obtained from $\mathbf{E}^{3C}_{K,1,1}$ by swapping columns from $(1,2,3,4,5,6)$ to $(3,1,2,6,4,5)$ first, then

$$\begin{aligned} r_1' &\to r_2', \qquad \alpha_1' \to \alpha_2' \\ r_2' &\to r_3', \qquad \alpha_2' \to \alpha_3' \\ r_3' &\to r_1', \qquad \alpha_3' \to \alpha_1' \end{aligned}$$

Now, let us look at $\mathbf{E}^{3C}_{K,2,1}$

$$\left[\begin{array}{ccc|ccc}
-\cot\alpha_3 r_1' & \frac{\cot\alpha_3\cot\alpha_3'}{2} & \frac{\cot\alpha_3\cot\alpha_2'}{2} & -\frac{\sqrt{6}\cot\alpha_3 r_1'}{2} & \frac{\sqrt{6}\cot\alpha_3 r_1'}{2} & \frac{\sqrt{6}\cot\alpha_3 r_1'}{2} \\
2r_2 r_1' & -r_2\cot\alpha_3' & -r_2\cot\alpha_2' & \sqrt{6}r_2 r_1' & -\sqrt{6}r_2 r_1' & -\sqrt{6}r_2 r_1' \\
-\cot\alpha_1 r_1' & \frac{\cot\alpha_1\cot\alpha_3'}{2} & \frac{\cot\alpha_1\cot\alpha_2'}{2} & -\frac{\sqrt{6}\cot\alpha_1 r_1'}{2} & \frac{\sqrt{6}\cot\alpha_1 r_1'}{2} & \frac{\sqrt{6}\cot\alpha_1 r_1'}{2} \\
\hline
-\sqrt{6}r_2 r_1' & \frac{\sqrt{6}r_2\cot\alpha_3'}{2} & \frac{\sqrt{6}r_2\cot\alpha_2'}{2} & r_2(\cot\alpha_2' - 4r_1') & 4r_2 r_1' & 2r_2 r_1' \\
\sqrt{6}r_2 r_1' & -\frac{\sqrt{6}r_2\cot\alpha_3'}{2} & -\frac{\sqrt{6}r_2\cot\alpha_2'}{2} & 3r_2 r_1' - (r_3 - r_1)(r_2' - r_3') & (\cot\alpha_1 - 4r_2)r_1' & (\cot\alpha_3 - 4r_2)r_1' \\
-\sqrt{6}r_2 r_1' & \frac{\sqrt{6}r_2\cot\alpha_3'}{2} & \frac{\sqrt{6}r_2\cot\alpha_2'}{2} & r_2(\cot\alpha_3' - 4r_1') & 2r_2 r_1' & 4r_2 r_1'
\end{array}\right],$$

which is obtained from $\mathbf{E}^{3C}_{K,1,1}$ by swapping rows from $(1,2,3,4,5,6)$ to $(3,1,2,6,4,5)$ first, then

$$\begin{aligned}
r_1 &\to r_2, \qquad \alpha_1 \to \alpha_2 \\
r_2 &\to r_3, \qquad \alpha_2 \to \alpha_3 \\
r_3 &\to r_1, \qquad \alpha_3 \to \alpha_1
\end{aligned}$$

d. Type-4 Edge Matrix

For the term $\left(\llbracket D\vec{\nabla}\Phi\rrbracket, \llbracket D\vec{\nabla}\Phi^*\rrbracket\right)_e$, we define the type-4 edge matrix as follows

$$\begin{aligned}
\mathbf{E}^4_i &= \frac{L_i}{2}\int_{\partial K_i} \nabla_{\mathbf{x}}\mathbf{b}\nabla_{\mathbf{x}}\mathbf{b}^T \,\mathrm{d}s, \qquad i = 1,2,3 \\
&= \int_{-1}^{1} \nabla_\xi \hat{\mathbf{b}}(\xi_i)\left[J^{-1}J^{-T}\frac{L_i^2}{4}\right]\nabla_\xi \hat{\mathbf{b}}^T(\xi_i)\,\mathrm{d}s
\end{aligned} \tag{3.81}$$

where,

$$J^{-1}J^{-T}\frac{L_i^2}{4} = 2r_i\begin{bmatrix} 2r_2 & -\cot\alpha_1 \\ -\cot\alpha_1 & 2r_3 \end{bmatrix} \tag{3.82}$$

The three local matrices are

$$\mathbf{E}^4_1 = r_1\left[\begin{array}{ccc|ccc}
2r_1 & -\cot\alpha_3 & -\cot\alpha_2 & \sqrt{6}r_1 & -\sqrt{6}r_1 & -\sqrt{6}r_1 \\
-\cot\alpha_3 & \underline{2r_2} & \underline{-\cot\alpha_1} & -\frac{\sqrt{6}\cot\alpha_3}{2} & \frac{\sqrt{6}\cot\alpha_3}{2} & \frac{\sqrt{6}\cot\alpha_3}{2} \\
-\cot\alpha_2 & \underline{-\cot\alpha_1} & \underline{2r_3} & -\frac{\sqrt{6}\cot\alpha_2}{2} & \frac{\sqrt{6}\cot\alpha_2}{2} & \frac{\sqrt{6}\cot\alpha_2}{2} \\
\hline
\sqrt{6}r_1 & -\frac{\sqrt{6}\cot\alpha_3}{2} & -\frac{\sqrt{6}\cot\alpha_2}{2} & \underline{2(r_1+r_2+r_3)} & \cot\alpha_2 - 4r_1 & \cot\alpha_3 - 4r_1 \\
-\sqrt{6}r_1 & \frac{\sqrt{6}\cot\alpha_3}{2} & \frac{\sqrt{6}\cot\alpha_2}{2} & \cot\alpha_2 - 4r_1 & 4r_1 & 2r_1 \\
-\sqrt{6}r_1 & \frac{\sqrt{6}\cot\alpha_3}{2} & \frac{\sqrt{6}\cot\alpha_2}{2} & \cot\alpha_3 - 4r_1 & 2r_1 & 4r_1
\end{array}\right]$$

$$\mathbf{E}_2^4 = r_2 \left[\begin{array}{ccc|ccc} \underline{2r_2} & -\cot\alpha_3 & \underline{-\cot\alpha_2} & \frac{\sqrt{6}\cot\alpha_3}{2} & -\frac{\sqrt{6}\cot\alpha_3}{2} & \frac{\sqrt{6}\cot\alpha_3}{2} \\ -\cot\alpha_3 & 2r_2 & -\cot\alpha_1 & -\sqrt{6}r_2 & \sqrt{6}r_2 & -\sqrt{6}r_2 \\ \underline{-\cot\alpha_2} & -\cot\alpha_1 & \underline{2r_3} & \frac{\sqrt{6}\cot\alpha_1}{2} & -\frac{\sqrt{6}\cot\alpha_1}{2} & \frac{\sqrt{6}\cot\alpha_1}{2} \\ \hline \frac{\sqrt{6}\cot\alpha_3}{2} & -\sqrt{6}r_2 & \frac{\sqrt{6}\cot\alpha_1}{2} & 4r_2 & \cot\alpha_1 - 4r_2 & 2r_2 \\ -\frac{\sqrt{6}\cot\alpha_3}{2} & \sqrt{6}r_2 & -\frac{\sqrt{6}\cot\alpha_1}{2} & \cot\alpha_1 - 4r_2 & \underline{2(r_1+r_2+r_3)} & \cot\alpha_3 - 4r_2 \\ \frac{\sqrt{6}\cot\alpha_3}{2} & -\sqrt{6}r_2 & \frac{\sqrt{6}\cot\alpha_1}{2} & 2r_2 & \cot\alpha_3 - 4r_2 & 4r_2 \end{array}\right]$$

$$\mathbf{E}_3^4 = r_3 \left[\begin{array}{ccc|ccc} \underline{2r_1} & \underline{-\cot\alpha_3} & -\cot\alpha_2 & \frac{\sqrt{6}\cot\alpha_2}{2} & -\frac{\sqrt{6}\cot\alpha_2}{2} & \frac{\sqrt{6}\cot\alpha_2}{2} \\ \underline{-\cot\alpha_3} & \underline{2r_2} & -\cot\alpha_1 & \frac{\sqrt{6}\cot\alpha_1}{2} & \frac{\sqrt{6}\cot\alpha_1}{2} & -\frac{\sqrt{6}\cot\alpha_1}{2} \\ -\cot\alpha_2 & -\cot\alpha_1 & 2r_3^2 & -\sqrt{6}r_3 & -\sqrt{6}r_3 & \sqrt{6}r_3 \\ \hline \frac{\sqrt{6}\cot\alpha_2}{2} & \frac{\sqrt{6}\cot\alpha_1}{2} & -\sqrt{6}r_3 & 4r_3 & 2r_3 & \cot\alpha_1 - 4r_3 \\ \frac{\sqrt{6}\cot\alpha_2}{2} & \frac{\sqrt{6}\cot\alpha_1}{2} & -\sqrt{6}r_3 & 2r_3 & 4r_3 & \cot\alpha_2 - 4r_3 \\ -\frac{\sqrt{6}\cot\alpha_2}{2} & -\frac{\sqrt{6}\cot\alpha_1}{2} & \sqrt{6}r_3 & \cot\alpha_1 - 4r_3 & \cot\alpha_2 - 4r_3 & \underline{2(r_1+r_2+r_3)} \end{array}\right]$$

$\mathbf{E_1^4}$ differs from $\mathbf{E_1^3}$ with three elements underlined in the above matrix.

As an example, when $r_1 = r_2 = r_3 = \frac{1}{\sqrt{3}}$,

$$\mathbf{E}_1^4 + \mathbf{E}_2^4 + \mathbf{E}_3^4 = \begin{bmatrix} 2 & -1 & -1 & \frac{4}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ -1 & 2 & -1 & -\frac{2}{\sqrt{6}} & \frac{4}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ -1 & -1 & 2 & -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{4}{\sqrt{6}} \\ \frac{4}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{14}{3} & -\frac{4}{3} & -\frac{4}{3} \\ -\frac{2}{\sqrt{6}} & \frac{4}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{4}{3} & \frac{14}{3} & -\frac{4}{3} \\ -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{4}{\sqrt{6}} & -\frac{4}{3} & -\frac{4}{3} & \frac{14}{3} \end{bmatrix}$$

We can see the difference between this summation of edge matrices and that of the type-3 edge matrices in Eq. (3.80) in the above result.

The type-4 edge-coupling matrix is defined as follows

$$\begin{aligned} \mathbf{E}_{K,i,j}^{4C} &= \frac{L_i}{2} \int_{\partial K_i} (\nabla_{\mathbf{x}} \mathbf{b})_K (\nabla_{\mathbf{x}} \mathbf{b}^T)_{K'} \, \mathrm{d}s, \qquad i = 1,2,3;\ j = 1,2,3 \\ &= \int_{-1}^{1} \nabla_\xi \hat{\mathbf{b}}(\xi_i) \left[(\frac{J^{-1} L_i}{2})_K (\frac{J^{-T} L_j}{2})_{K'} \right] \nabla_\xi \hat{\mathbf{b}}^T(-\xi_j) \, \mathrm{d}s \end{aligned} \tag{3.83}$$

where,

$$(\frac{J^{-1}L_i}{2})_K(\frac{J^{-T}L_j}{2})_{K'} = \frac{L_iL'_j}{4AA'}\begin{bmatrix} y_3-y_1 & x_1-x_3 \\ y_1-y_2 & x_2-x_1 \end{bmatrix}\begin{bmatrix} y'_3-y'_1 & y'_1-y'_2 \\ x'_1-x'_3 & x'_2-x'_1 \end{bmatrix} \tag{3.84}$$

The above equation can be further simplified given the numeral values of the indices i and j. For example, for $i = 2$ and $j = 2$ as showed in Fig. III-2, we obtain with some additional algebra

$$\left(\frac{J^{-1}L_i}{2}\right)_K\left(\frac{J^{-T}L_j}{2}\right)_{K'} = 4\begin{bmatrix} -r_2r'_2 & \frac{r_2\cot\alpha'_1}{2} \\ \frac{\cot\alpha_1 r'_2}{2} & -\frac{1+\cot\alpha_1\cot\alpha'_1}{4} \end{bmatrix} \tag{3.85}$$

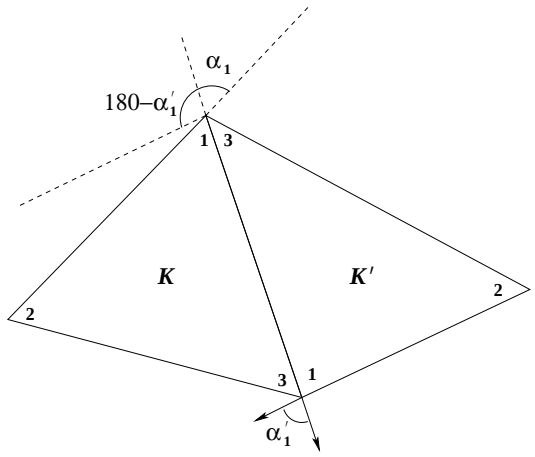


Fig. III-2. Element coupling through edge.

So, coupling matrix $\mathbf{E}^{4C}_{K,2,2}$ is

$$\left[\begin{array}{ccc|ccc} -\frac{1+\cot\alpha_3\cot\alpha'_3}{2} & \cot\alpha_3 r'_2 & \frac{1-\cot\alpha_3\cot\alpha'_1}{2} & -\frac{\sqrt{6}\cot\alpha_3 r'_2}{2} & \frac{\sqrt{6}\cot\alpha_3 r'_2}{2} & -\frac{\sqrt{6}\cot\alpha_3 r'_2}{2} \\ r_2\cot\alpha'_3 & -2r_2r'_2 & r_2\cot\alpha_1 & \sqrt{6}r_2r'_2 & -\sqrt{6}r_2r'_2 & \sqrt{6}r_2r'_2 \\ \frac{1-\cot\alpha_1\cot\alpha'_3}{2} & \cot\alpha_1 r'_2 & -\frac{1+\cot\alpha_1\cot\alpha'_1}{2} & -\frac{\sqrt{6}\cot\alpha_1 r'_2}{2} & \frac{\sqrt{6}\cot\alpha_1 r'_2}{2} & -\frac{\sqrt{6}\cot\alpha_1 r'_2}{2} \\ \hline -\frac{\sqrt{6}r_2\cot\alpha'_3}{2} & \sqrt{6}r_2r'_2 & -\frac{\sqrt{6}r_2\cot\alpha'_1}{2} & -2r_2r'_2 & r_2(4r'_2-\cot\alpha'_3) & -4r_2r'_2 \\ \frac{\sqrt{6}r_2\cot\alpha'_3}{2} & -\sqrt{6}r_2r'_2 & \frac{\sqrt{6}r_2\cot\alpha'_1}{2} & (4r_2-\cot\alpha_3)r'_2 & 1-3r_2r'_2+(r_1-r_3)(r'_1-r'_3) & (4r_2-\cot\alpha_1)r'_2 \\ -\frac{\sqrt{6}r_2\cot\alpha'_3}{2} & \sqrt{6}r_2r'_2 & -\frac{\sqrt{6}r_2\cot\alpha'_1}{2} & -4r_2r'_2 & r_2(4r'_2-\cot\alpha'_1) & -2r_2r'_2 \end{array}\right]$$

We will not give the matrices for all possible combinations of i and j. Note that all

these matrices depend on the ratios r_i of the two neighboring elements as the case of the type-3 edge matrices.

2. Local Diffusion System

We use three sample elements shown on Fig. III-3 to demonstrate that how the local diffusion system is assembled using the above the elementary matrices.

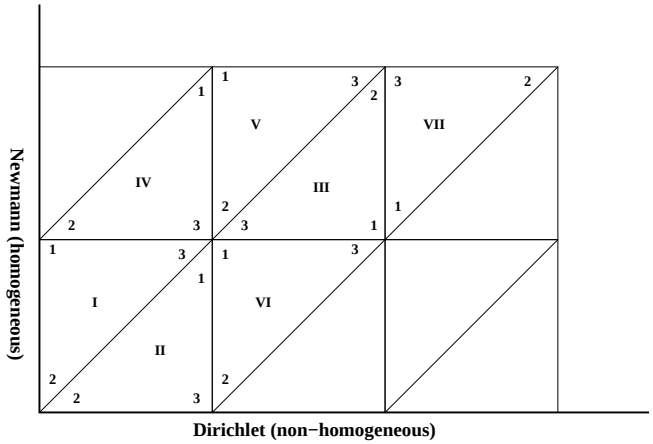


Fig. III-3. Sample diffusion domain.

We make the following assumptions: the $\vec{Q}_1$ term is zero; the Neumann boundary condition is homogeneous, i.e., reflecting; the Dirichlet boundary condition is non-homogeneous with a given Dirichlet scalar flux Φ^d. Let us suppose the volumetric source Q_0 and the surface scalar flux are properly assembled in vectors $\mathbf{Q}_0$ and $\mathbf{\Phi}^d$ (projection operations may be needed if Q_0 or Φ^d are not in the function space of piece-wise polynomials).

The DCF local system of element I is:

$$
\begin{aligned}
\mathbf{A}_{\mathrm{I}} &= \frac{(\sigma_a A)_{K=\mathrm{I}}}{12}\mathrm{M} + [D\mathbf{S}]_{K=\mathrm{I}} \\
&\quad + \frac{L_1^{K=\mathrm{I}}}{24}\mathrm{E}_1^1 - \frac{1}{2}\left[D(\mathbf{E}_1^2 + \mathbf{E}_1^{2T})\right]_{K=\mathrm{I}} - \frac{9}{8}\left[\frac{D^2}{L_1}(\mathbf{E}_1^3 + \mathbf{E}_1^4)\right]_{K=\mathrm{I}} \\
&\quad + \frac{L_2^{K=\mathrm{I}}}{24}\mathrm{E}_2^1 - \frac{1}{2}\left[D(\mathbf{E}_2^2 + \mathbf{E}_2^{2T})\right]_{K=\mathrm{I}} - \frac{9}{8}\left[\frac{D^2}{L_2}(\mathbf{E}_2^3 + \mathbf{E}_2^4)\right]_{K=\mathrm{I}} \\
&\quad - \frac{9}{2}\left[\frac{D^2}{L_1}\mathbf{E}_3^3\right]_{K=\mathrm{I}} \\
\mathbf{b}_{\mathrm{I}} &= \frac{1}{12}\mathrm{M}\left[A\mathbf{Q}_0\right]_{K=\mathrm{I}} \\
&\quad + \left\{ \begin{array}{l} \frac{L_1^{K=\mathrm{I}}}{24}\mathrm{E}_{1,3}^{1C} - \frac{1}{2}\left[D\mathbf{E}_{K,1,3}^{2C}\right]_{K=\mathrm{I}} - \frac{1}{2}\left[D\mathbf{E}_{K,3,1}^{2C,T}\right]_{K=\mathrm{II}} \\ + \frac{9}{8}D_{K=\mathrm{II}}\left[\frac{D}{L_1}(\mathbf{E}_{K,1,3}^{3C} + \mathbf{E}_{K,1,3}^{4C})\right]_{K=\mathrm{I}} \end{array} \right\} \mathbf{\Phi}_{K=\mathrm{II}} \\
&\quad + \left\{ \begin{array}{l} \frac{L_2^{K=\mathrm{I}}}{24}\mathrm{E}_{2,1}^{1C} - \frac{1}{2}\left[D\mathbf{E}_{K,2,1}^{2C}\right]_{K=\mathrm{I}} - \frac{1}{2}\left[D\mathbf{E}_{K,1,2}^{2C,T}\right]_{K=\mathrm{IV}} \\ + \frac{9}{8}D_{K=\mathrm{IV}}\left[\frac{D}{L_2}(\mathbf{E}_{K,2,1}^{3C} + \mathbf{E}_{K,2,1}^{4C})\right]_{K=\mathrm{I}} \end{array} \right\} \mathbf{\Phi}_{K=\mathrm{IV}} \\
&= \mathbf{q}_{K=\mathrm{I}} + \mathbf{A}_{\mathrm{I,II}}\mathbf{\Phi}_{K=\mathrm{II}} + \mathbf{A}_{\mathrm{I,IV}}\mathbf{\Phi}_{K=\mathrm{IV}}
\end{aligned}
$$

In this example, the reflecting boundary condition is homogeneous; thus, there are no boundary contributions to the right-hand-side of local edge 3.

We will use a local matrix T_i^T $(i = 1, 2, 3)$ to fill in zeros in the Dirichlet boundary source vectors $\mathbf{\Phi}^d$ corresponding to all interior shape functions and all shape functions for the other edges and the shape function for other vertex, so that we can use the edge matrices to assemble their contributions. The definition of T_i^T can be found in Sec. 5 of Chapter IV.

The local system of element II is,

$$
\begin{aligned}
\mathbf{A}_{\mathrm{II}} &= \frac{(\sigma_a A)_{K=\mathrm{II}}}{12}\mathrm{M} + [D\mathbf{S}]_{K=\mathrm{II}} \\
&\quad + \frac{L_2^{K=\mathrm{II}}}{24}\mathrm{E}_2^1 - \frac{1}{2}\left[D(\mathbf{E}_2^2 + \mathbf{E}_2^{2T})\right]_{K=\mathrm{II}} - \frac{9}{8}\left[\frac{D^2}{L_2}(\mathbf{E}_2^3 + \mathbf{E}_2^4)\right]_{K=\mathrm{II}} \\
&\quad + \frac{L_3^{K=\mathrm{II}}}{24}\mathrm{E}_3^1 - \frac{1}{2}\left[D(\mathbf{E}_3^2 + \mathbf{E}_3^{2T})\right]_{K=\mathrm{II}} - \frac{9}{8}\left[\frac{D^2}{L_3}(\mathbf{E}_3^3 + \mathbf{E}_3^4)\right]_{K=\mathrm{II}} \\
&\quad + \frac{L_1^{K=\mathrm{II}}}{24}\mathrm{E}_1^1 - \frac{1}{2}\left[D(\mathbf{E}_1^2 + \mathbf{E}_1^{2T})\right]_{K=\mathrm{II}} - \frac{9}{8}\left[\frac{D^2}{L_1}(\mathbf{E}_1^3 + \mathbf{E}_1^4)\right]_{K=\mathrm{II}} \\
\mathbf{b}_{\mathrm{II}} &= \frac{1}{12}\mathrm{M}\left[A\mathbf{Q}_0\right]_{K=\mathrm{II}} \\
&\quad + \left\{\frac{L_1^{K=\mathrm{II}}}{24}\mathrm{E}_1^1 - \frac{1}{2}\left[D\mathbf{E}_1^2\right]_{K=\mathrm{II}}\right\}\mathrm{T}_1^T\mathbf{\Phi}_{e=(\mathrm{II},1)}^d \\
&\quad + \left\{\begin{array}{l} \frac{L_2^{K=\mathrm{II}}}{24}\mathrm{E}_{2,3}^{1C} - \frac{1}{2}\left[D\mathbf{E}_{K,2,3}^{2C}\right]_{K=\mathrm{II}} - \frac{1}{2}\left[D\mathbf{E}_{K,3,2}^{2C,T}\right]_{K=\mathrm{VI}} \\ +\frac{9}{8}D_{K=\mathrm{VI}}\left[\frac{D}{L_2}(\mathbf{E}_{K,2,3}^{3C} + \mathbf{E}_{K,2,3}^{4C})\right]_{K=\mathrm{II}} \end{array}\right\}\mathbf{\Phi}_{K=\mathrm{VI}} \\
&\quad + \left\{\begin{array}{l} \frac{L_3^{K=\mathrm{II}}}{24}\mathrm{E}_{3,1}^{1C} - \frac{1}{2}\left[D\mathbf{E}_{K,3,1}^{2C}\right]_{K=\mathrm{II}} - \frac{1}{2}\left[D\mathbf{E}_{K,1,3}^{2C,T}\right]_{K=\mathrm{I}} \\ +\frac{9}{8}D_{K=\mathrm{I}}\left[\frac{D}{L_3}(\mathbf{E}_{K,3,1}^{3C} + \mathbf{E}_{K,3,1}^{4C})\right]_{K=\mathrm{II}} \end{array}\right\}\mathbf{\Phi}_{K=\mathrm{I}} \\
&= \mathbf{q}_{K=\mathrm{II}} + \mathbf{A}_{\mathrm{II,VI}}\mathbf{\Phi}_{K=\mathrm{VI}} + \mathbf{A}_{\mathrm{II,I}}\mathbf{\Phi}_{K=\mathrm{I}}
\end{aligned}
$$

It can be shown that the two block matrices $\mathbf{A}_{\mathrm{II,I}}$ and $\mathbf{A}_{\mathrm{I,II}}$ are identical and that both $\mathbf{A}_{\mathrm{I}}$ and $\mathbf{A}_{\mathrm{II}}$ are symmetric. Note that they may not positive definite.

The local matrix for interior element III is simpler to assemble,

$$
\begin{aligned}
\mathbf{A}_{\mathrm{III}} &= \frac{(\sigma_a A)_{K=\mathrm{III}}}{12}\mathrm{M} + [D\mathbf{S}]_{K=\mathrm{III}} \\
&\quad + \frac{L_1^{K=\mathrm{III}}}{24}\mathrm{E}_1^1 - \frac{1}{2}\left[D(\mathbf{E}_1^2 + \mathbf{E}_1^{2T})\right]_{K=\mathrm{III}} - \frac{9}{8}\left[\frac{D^2}{L_1}(\mathbf{E}_1^3 + \mathbf{E}_1^4)\right]_{K=\mathrm{III}} \\
&\quad + \frac{L_2^{K=\mathrm{III}}}{24}\mathrm{E}_2^1 - \frac{1}{2}\left[D(\mathbf{E}_2^2 + \mathbf{E}_2^{2T})\right]_{K=\mathrm{III}} - \frac{9}{8}\left[\frac{D^2}{L_2}(\mathbf{E}_2^3 + \mathbf{E}_2^4)\right]_{K=\mathrm{III}} \\
&\quad + \frac{L_3^{K=\mathrm{III}}}{24}\mathrm{E}_3^1 - \frac{1}{2}\left[D(\mathbf{E}_3^2 + \mathbf{E}_3^{2T})\right]_{K=\mathrm{III}} - \frac{9}{8}\left[\frac{D^2}{L_3}(\mathbf{E}_3^3 + \mathbf{E}_3^4)\right]_{K=\mathrm{III}} \\
\mathbf{b}_{\mathrm{III}} &= \frac{1}{12}\mathrm{M}\left[A\mathbf{Q}_0\right]_{K=\mathrm{III}} \\
&\quad + \left\{ \begin{array}{l} \frac{L_1^{K=\mathrm{III}}}{24}\mathrm{E}_{1,1}^{1C} - \frac{1}{2}\left[D\mathbf{E}_{K,1,1}^{2C}\right]_{K=\mathrm{III}} - \frac{1}{2}\left[D\mathbf{E}_{K,1,1}^{2C,T}\right]_{K=\mathrm{V}} \\ + \frac{9}{8}D_{K=\mathrm{V}}\left[\frac{D}{L_1}(\mathbf{E}_{K,1,1}^{3C} + \mathbf{E}_{K,1,1}^{4C})\right]_{K=\mathrm{III}} \end{array} \right\} \mathbf{\Phi}_{K=\mathrm{V}} \\
&\quad + \left\{ \begin{array}{l} \frac{L_2^{K=\mathrm{III}}}{24}\mathrm{E}_{2,2}^{1C} - \frac{1}{2}\left[D\mathbf{E}_{K,2,2}^{2C}\right]_{K=\mathrm{III}} - \frac{1}{2}\left[D\mathbf{E}_{K,2,2}^{2C,T}\right]_{K=\mathrm{VI}} \\ + \frac{9}{8}D_{K=\mathrm{VI}}\left[\frac{D}{L_2}(\mathbf{E}_{K,2,2}^{3C} + \mathbf{E}_{K,2,2}^{4C})\right]_{K=\mathrm{III}} \end{array} \right\} \mathbf{\Phi}_{K=\mathrm{VI}} \\
&\quad + \left\{ \begin{array}{l} \frac{L_3^{K=\mathrm{III}}}{24}\mathrm{E}_{3,2}^{1C} - \frac{1}{2}\left[D\mathbf{E}_{K,3,2}^{2C}\right]_{K=\mathrm{III}} - \frac{1}{2}\left[D\mathbf{E}_{K,2,3}^{2C,T}\right]_{K=\mathrm{VII}} \\ + \frac{9}{8}D_{K=\mathrm{VII}}\left[\frac{D}{L_3}(\mathbf{E}_{K,3,2}^{3C} + \mathbf{E}_{K,3,2}^{4C})\right]_{K=\mathrm{III}} \end{array} \right\} \mathbf{\Phi}_{K=\mathrm{VII}} \\
&= \mathbf{q}_{K=\mathrm{III}} + \mathbf{A}_{\mathrm{III,V}}\mathbf{\Phi}_{K=\mathrm{V}} + \mathbf{A}_{\mathrm{III,VI}}\mathbf{\Phi}_{K=\mathrm{VI}} + \mathbf{A}_{\mathrm{III,VII}}\mathbf{\Phi}_{K=\mathrm{VII}}
\end{aligned}
$$

Repeating this process for all elements, we assemble the DG-diffusion system for the entire computational domain:

$$
\mathbf{A}\mathbf{\Phi} = \mathbf{d} = \mathbf{B}\begin{bmatrix} \mathbf{Q}_0 \\ \mathbf{Q}_1 \\ \mathbf{q}_{edge} \end{bmatrix} \tag{3.86}
$$

We use $\mathbf{q}_{edge}$ to represent all edge sources including the surface source on the non-homogeneous boundaries. Multiplying the $\mathbf{B}$ matrix and the source vector $[\mathbf{Q}_0, \mathbf{Q}_1, \mathbf{q}_{edge}]^T$ gives the global right-hand-side of the system. We will see that there will be surface sources for DSA introduced by the significant angular flux in Sec. D in this chapter

and Sec. 3 in Chapter V.

3. Solving the DG-Diffusion Problem

Because the global system matrix $\mathbf{A}$ is symmetric and positive definite (SPD) for the IP and MIP, we can use a simple Preconditioned Conjugate Gradient (PCG) to solve the resulting linear system of equations. The system matrix $\mathbf{A}$ is split into $\mathrm{L} + \mathrm{D} + \mathrm{L}$, where the D is a positive block diagonal and L is the block-lower-triangular; this splitting is useful for the preconditioner stage. Each block corresponds to a local (active) element (the terminology active refers to their AMR process, to be detailed in Chapter IV). The dimension of each element block depends on the local polynomial order.

A simple SSOR (symmetric Successive Over-Relaxation) technique is used to precondition the diffusion equation with the over-relaxation factor 1, i.e., the preconditioner is given by $\mathcal{M} = (\mathrm{D} + \mathrm{L})\mathrm{D}^{-1}(\mathrm{D} + \mathrm{L}^T)$. This preconditioner can be easily implemented with the Eisenstat trick [114] presented in Appendix E. Only one forward and backward arbitrarily ordered element-based diffusion sweeps are needed. The chosen initial guess is zero and the recommended stopping criterion is,

$$\frac{\|\mathbf{A}\mathbf{\Phi} - \mathbf{d}\|_{\mathrm{M}}}{\|\mathbf{d}\|} \leqslant \epsilon \tag{3.87}$$

where the M-norm is defined as

$$\|\mathbf{\Phi}\|_{\mathrm{M}} = \sqrt{\mathbf{\Phi}^T \mathcal{M} \mathbf{\Phi}} \tag{3.88}$$

When the DCF form is used, the diffusion system should be solved with, for instance, an SQMR package [115] because the DCF form is symmetric but not PD. The P_1 forms have not been implemented in XUTHUS.

When solving a stand-alone diffusion problem, we need to provide the conver-

gence criterion:

$$\epsilon = \text{tol}_{inner} \tag{3.89}$$

D. DFE Diffusion Forms as Preconditioners to the DFE S_N Transport Form

The goal in deriving DFE forms for the diffusion equations was to employ them as preconditioning techniques to accelerate the DFE transport sweeps, which are solved either using SI or GMRes. We discuss these DSA preconditioners here.

1. DSA for SI

One step of SI for the DFE S_N transport form can be written as

$$\begin{aligned} b(\Psi^{(\ell+1/2)}, \Psi^*) =& l(\Psi^*) + \sum_{n=0}^{N_a} \sum_{k=-n}^{n} (2n+1)(\sigma_{s,n}\Phi_{n,k}^{(\ell)}, \Phi_{n,k}^*)_{\mathcal{D}} + \\ & \sum_{e\in\partial\mathcal{D}^r} \sum_{\vec{\Omega}_m\cdot\vec{n}_b<0} 4\pi w_m \left\langle \Psi_{m'}^{(\ell)}, \Psi_m^* \right\rangle_e \end{aligned} \tag{3.90}$$

Note that we start the transport sweeps on the reflecting boundaries; the incoming angular flux values on the reflecting boundaries are coming from the outgoing angular fluxes at the previous iteration. As a result, we can still enjoy a matrix-free ordered transport sweep. Note that SI not only converges the flux moments but also the outgoing angular fluxes on the reflecting boundaries, which we named as SAF in Chapter II.

Replacing the iterates ℓ and $\ell + 1/2$ with the converged solution, we have

$$\begin{aligned} b(\Psi^{cvg}, \Psi^*) =& l(\Psi^*) + \sum_{n=0}^{N_a} \sum_{k=-n}^{n} (2n+1)(\sigma_{s,n}\Phi_{n,k}^{cvg}, \Phi_{n,k}^*)_{\mathcal{D}} + \\ & \sum_{e \in \partial \mathcal{D}^r} \sum_{\vec{\Omega}_m \cdot \vec{n}_b < 0} 4\pi w_m \left\langle \Psi_{m'}^{cvg}, \Psi_m^* \right\rangle_e \end{aligned} \tag{3.91}$$

Subtracting these two equations, we obtain an equation for the iteration error,

$$\begin{aligned} b(\epsilon^{(\ell+1/2)}, \Psi^*) =& \sum_{n=0}^{N_a} \sum_{k=-n}^{n} (2n+1)(\sigma_{s,n}\mathcal{E}_{n,k}^{(\ell)}, \Phi_{n,k}^*)_{\mathcal{D}} + \\ & \sum_{e \in \partial \mathcal{D}^r} \sum_{\vec{\Omega}_m \cdot \vec{n}_b < 0} 4\pi w_m \left\langle \epsilon_{m'}^{(\ell)}, \Psi_m^* \right\rangle_e \end{aligned} \tag{3.92}$$

where the angular error and the error moments are, respectively,

$$\epsilon^{(\ell)} = \Psi^{cvg} - \Psi^{(\ell)} \tag{3.93}$$

$$\mathcal{E}_{n,k}^{(\ell)} = \Phi_{n,k}^{cvg} - \Phi_{n,k}^{(\ell)} \tag{3.94}$$

We have

$$\begin{aligned} \epsilon^{(\ell)} &= \epsilon^{(\ell+1/2)} + \left(\Psi^{(\ell+1/2)} - \Psi^{(\ell)}\right) = \epsilon^{(\ell+1/2)} + \delta\Psi^{(\ell)} \\ \mathcal{E}_{n,k}^{(\ell)} &= \mathcal{E}_{n,k}^{(\ell+1/2)} + \left(\Phi_{n,k}^{(\ell+1/2)} - \Phi_{n,k}^{(\ell)}\right) = \mathcal{E}_{n,k}^{(\ell+1/2)} + \delta\Phi_{n,k}^{(\ell)} \end{aligned}$$

Therefore, we have the following equation for the angular error at iteration $\ell + 1/2$ as a function of only the changes in angular fluxes $\delta\Psi^{(\ell+1/2)}$ and flux moments $\delta\Phi_{n,k}^{(\ell+1/2)}$,

$$\begin{aligned} a(\epsilon^{(\ell+1/2)}, \Psi^*) =& \sum_{n=0}^{N_a} \sum_{k=-n}^{n} (2n+1)(\sigma_{s,n}\delta\Phi_{n,k}^{(\ell)}, \Phi_{n,k}^*)_{\mathcal{D}} + \\ & \sum_{e \in \partial \mathcal{D}^r} \sum_{\vec{\Omega}_m \cdot \vec{n}_b < 0} 4\pi w_m \left\langle \delta\Psi_{m'}^{(\ell)}, \Psi_m^* \right\rangle_e \end{aligned} \tag{3.95}$$

Note the bilinear form $a(\cdot,\cdot)$ contains the removal bilinear form $b(\cdot,\cdot)$, and the scat-

tering and reflecting boundary terms. Refer the Sec. 3 of Chapter II for more details. We see that the source terms for the angular error $\epsilon^{(\ell+1/2)}$ is the difference between two successive SI values $\delta\Psi^{(\ell+1/2)}$ and $\delta\Phi_{n,k}^{(\ell+1/2)}$. If we define the isotropic and linearly anisotropic error source terms as

$$Q_0 = \sigma_{s,0}\delta\Phi^{(\ell)} \tag{3.96}$$

$$\vec{Q}_1 = \sigma_{s,1}\delta\vec{J}^{(\ell)} \tag{3.97}$$

and define the following surface sources on reflecting boundaries,

$$\delta J^{inc} = \sum_{\vec{\Omega}_m\cdot\vec{n}_b<0} w_m|\vec{\Omega}_m\cdot\vec{n}_b|\delta\Psi_{m'}^{(\ell)} = \sum_{\vec{\Omega}_m\cdot\vec{n}_b>0} w_m|\vec{\Omega}_m\cdot\vec{n}_b|\delta\Psi_{m}^{(\ell)} \tag{3.98}$$

$$\delta\vec{\Upsilon}^{inc} = -\sum_{\vec{\Omega}_m\cdot\vec{n}_b>0} 3w_m\vec{\Omega}_m|\vec{\Omega}_m\cdot\vec{n}_b|\delta\Psi_{m}^{(\ell)} \tag{3.99}$$

we can then solve an approximate equation for the angular error $\epsilon^{(l+1/2)}$, using for instance the MIP form of the DFE diffusion equation for the unknown $\Phi = \mathcal{E}^{(\ell+1/2)}$,

$$\begin{aligned} b_{MIP}(\Phi,\Phi^*) &= (\sigma_a\Phi,\Phi^*)_{\mathcal{D}} + \left(D\vec{\nabla}\Phi,\vec{\nabla}\Phi^*\right)_{\mathcal{D}} \\ &\quad + (\kappa_e[\![\Phi]\!],[\![\Phi^*]\!])_{E_h^i} + ([\![\Phi]\!],\{\!\{D\partial_n\Phi^*\}\!\})_{E_h^i} + (\{\!\{D\partial_n\Phi\}\!\},[\![\Phi^*]\!])_{E_h^i} \\ &\quad + (\kappa_e\Phi,\Phi^*)_{\partial\mathcal{D}^d} - \frac{1}{2}(\Phi,D\partial_n\Phi^*)_{\partial\mathcal{D}^d} - \frac{1}{2}(D\partial_n\Phi,\Phi^*)_{\partial\mathcal{D}^d} \end{aligned} \tag{3.100}$$

$$\begin{aligned} l_{MIP}(\Phi^*) &= (Q_0,\Phi^*)_{\mathcal{D}} - \left(3\vec{\nabla}\cdot D\vec{Q}_1,\Phi^*\right)_{\mathcal{D}} + \left(\delta J^{inc},\Phi^*\right)_{\partial\mathcal{D}^r} \\ &\quad + \left(3\{\!\{D\vec{Q}_1\cdot\vec{n}\}\!\},[\![\Phi^*]\!]\right)_{E_h^i} - \frac{1}{2}\left(3D\vec{Q}_1\cdot\vec{n},\Phi^*\right)_{\partial\mathcal{D}^d} \end{aligned} \tag{3.101}$$

Recall that in the MIP form, we dropped the $\vec{\Upsilon}$ term on the boundaries, assuming their effect is not significant. We also noticed the boundary condition on the vacuum boundary is of Dirichlet type and not of Robin type, as proposed in the asymptotic

analysis. With the DCF form, the source linear functional is,

$$\begin{aligned} l_{DCF}(\Phi^*) &= (Q_0,\Phi^*)_{\mathcal{D}} - \left(3\vec{\nabla}\cdot D\vec{Q}_1,\Phi^*\right)_{\mathcal{D}} + \left(\delta J^{inc},\Phi^*\right)_{\partial\mathcal{D}^r} \qquad (3.102)\\ &\quad - \left(\delta\vec{\Upsilon}^{inc},D\vec{\nabla}\Phi^*\right)_{\partial\mathcal{D}^r} + 2\left(\delta\vec{\Upsilon}^{inc}\cdot\vec{n},D\partial_n\Phi^*\right)_{\partial\mathcal{D}^r}\\ &\quad + \left(3\{\!\{D\vec{Q}_1\cdot\vec{n}\}\!\},[\![\Phi^*]\!]\right)_{E_h^i} - \frac{1}{2}\left(3D\vec{Q}_1\cdot\vec{n},\Phi^*\right)_{\partial\mathcal{D}^d} \qquad (3.103)\end{aligned}$$

where the higher moment terms with $\vec{\Upsilon}$ are kept. To obtain these two terms, Eq. (2.6) given the reflecting directions is used.

Once we obtain the scalar error $\mathcal{E}^{(\ell+1/2)}$, we need to prolongate it back and modify the $\ell+1/2$ transport solution. (Note that we need to update current and SAF unknowns consistently if we assumed no $\vec{Q}_1$ terms in the diffusion approximation.)

$$\Phi^{(\ell+1)} = \Phi^{(\ell+1/2)} + \mathcal{E}^{(\ell+1/2)} \qquad (3.104)$$

$$\vec{J}^{(\ell+1)} = \vec{J}^{(\ell+1/2)} + \frac{\sigma_{s,1}\delta\vec{J}^{(\ell)}}{\sigma_{tr}} - D\vec{\nabla}\mathcal{E}^{(\ell+1/2)} \qquad (3.105)$$

$$\Psi_m^{(\ell+1)} = \Psi_m^{(\ell+1/2)} + \frac{1}{4\pi}\left(\mathcal{E}^{(\ell+1/2)} + 3\left(\frac{\sigma_{s,1}\delta\vec{J}^{(\ell)}}{\sigma_{tr}} - D\vec{\nabla}\mathcal{E}^{(\ell+1/2)}\right)\cdot\vec{\Omega}_m\right) \qquad (3.106)$$

$$\text{on } \vec{r}_b\in\partial\mathcal{D}^r, \vec{\Omega}_m\cdot\vec{n}(\vec{r}_b)>0$$

The entire procedure can be written in a matrix form as follows. One SI is performed ,

$$\mathbf{x}^{(\ell+1/2)} = \mathbf{T}\mathbf{x}^{(\ell)} + \mathbf{b} \qquad (3.107)$$

where the notation is identical to the one used in Chapter II.

Finally, the diffusion correction to the transport solution, after each source iteration, is

$$\delta\mathbf{x}^{(\ell)} = \mathbf{P}\mathbf{A}^{-1}\mathbf{B}\mathbf{R}(\mathbf{x}^{(\ell+1/2)} - \mathbf{x}^{(\ell)}) \qquad (3.108)$$

where $\mathbf{R}$ is the restriction operation which gives the $[\mathbf{Q}_0,\mathbf{Q}_1,\mathbf{q}_{edge}]^T$ with Eqs. (3.96)

through (3.99). $\mathbf{P}$ is the prolongation operation which manipulates the scalar flux to obtain the correction of zero and first flux moments and SAF with Eqs. (3.104) through (3.106). $\mathbf{A}$ is the global DG-diffusion matrix and $\mathbf{B}$ is used to construct the right-hand-side of the DG-diffusion problem.

Thus,

$$\begin{aligned}
\mathbf{x}^{(\ell+1)} &= \mathbf{x}^{(\ell+1/2)} + \delta\mathbf{x}^{(\ell)} \\
&= \mathbf{T}\mathbf{x}^{(\ell)} + \mathbf{b} + \mathbf{P}\mathbf{A}^{-1}\mathbf{B}\mathbf{R}(\mathbf{x}^{(\ell+1/2)} - \mathbf{x}^{(\ell)}) \\
&= \mathbf{T}\mathbf{x}^{(\ell)} + \mathbf{b} + \mathbf{P}\mathbf{A}^{-1}\mathbf{B}\mathbf{R}(\mathbf{T}\mathbf{x}^{(\ell)} + \mathbf{b} - \mathbf{x}^{(\ell)}) \\
&= \mathbf{x}^{(\ell)} + (\mathbf{T} - \mathbf{I})\mathbf{x}^{(\ell)} + \mathbf{b} + \mathbf{P}\mathbf{A}^{-1}\mathbf{B}\mathbf{R}(\mathbf{T}\mathbf{x}^{(\ell)} + \mathbf{b} - \mathbf{x}^{(\ell)}) \\
&= \mathbf{x}^{(\ell)} - (\mathbf{I} + \mathbf{P}\mathbf{A}^{-1}\mathbf{B}\mathbf{R})(\mathbf{I} - \mathbf{T})\mathbf{x}^{(\ell)} + (\mathbf{I} + \mathbf{P}\mathbf{A}^{-1}\mathbf{B}\mathbf{R})\mathbf{b}
\end{aligned}$$

which is exactly a preconditioned Richardson iteration:

$$(\mathbf{I} + \mathbf{P}\mathbf{A}^{-1}\mathbf{B}\mathbf{R})(\mathbf{I} - \mathbf{T})\mathbf{x} = (\mathbf{I} + \mathbf{P}\mathbf{A}^{-1}\mathbf{B}\mathbf{R})\mathbf{b} \tag{3.109}$$

A loose criterion could degrade the spectral radius of DSA, while a tight criterion could augment excessively the CPU time spent in the DSA solves. As a compromise, the recommended stopping criterion is,

$$\epsilon = \text{tol}_{DSA} = 10^{-1} \tag{3.110}$$

This tolerance can be changed by the user. The initial guess for the DSA iterations is zero. Note that a tolerance $\text{tol}_{DSA} = 10^{-4}$ is chosen to analyze the spectral radius of the DSA schemes, in order to avoid creating numerical noise in the estimation of the spectral radius. When the SQMR solver is used for the DCF scheme, the convergence criteria of DSA needs to be set to 10^{-4}. Numerical results to validate these settings are presented in Section F.

2. DSA for GMRes

In the preceding section, we have demonstrated that the preconditioner for the transport system is $\mathbf{I}+\mathbf{PA}^{-1}\mathbf{BR}$. So we need to provide the action of matrix on a Krylov vector $\mathbf{v}$, i.e., $\mathbf{y}=(\mathbf{I}+\mathbf{PA}^{-1}\mathbf{BR})\mathbf{v}$. This matrix-vector operation can be done with two steps:

$$\mathbf{w} = \mathbf{PA}^{-1}\mathbf{BRv} \tag{3.111}$$

$$\mathbf{y} = \mathbf{w}+\mathbf{v} \tag{3.112}$$

So if we already have the DSA subroutine to accelerate SI, we can let $\mathbf{x}^{(\ell+1/2)}=\mathbf{v}$ and $\mathbf{x}^{(\ell)}=\mathbf{0}$, and reuse it, thus reducing the coding effort greatly.

Applying the preconditioner requires solving the diffusion system iteratively. As the result of iterative error, the preconditioning operation is inexact. Experiences showed that we can still achieve the converged transport solution while relaxing the convergence of the diffusion solver significantly. The strategy on the convergence tolerance of diffusion solver can affects the overall efficiency of the transport solver greatly. Studies on the inexact Krylov methods including the inexact preconditioning can be seen [116, 117]. The idea is that because at any particular GMRes iteration a Krylov subspace solver constructs a solution based on the solutions from previous iterations to that point, the preconditioning operation should be computed with a strict convergence tolerance in the early stages of the GMRes iterative solution and the tolerance can be relaxed as the GMRes iteration proceeds.

So the strategy is that

$$\epsilon = c\frac{\mathrm{tol}_{inner}}{error} \tag{3.113}$$

where *error* is the transport solution error of the previous GMRes iteration; c is a small constant to assure the convergence of the GMRes. We use $c=0.001$ and set

the initial guess of the diffusion system to zero.

E. Fourier Analysis (FA)

To analyze the performance of acceleration schemes, it is customary to carry out a Fourier Analysis (FA) on the discretized equations. A large body of work exists in the transport community regarding the application of FA to the study of acceleration of the SI scheme with DSA, both for the continuous and discretized equations [118, 68, 70]. Obviously, for numerical applications, the study of the discretized {transport + acceleration} solvers is of prime interest. Oftentimes, this is done for a periodic homogeneous domain, and in fewer cases for a periodic heterogeneous domain. We present here our FA of the various DGFEM diffusion schemes used to accelerated the DGFEM S_N transport equations. This study includes different geometries, media and mesh aspect ratios. First, we give the FA formulation. Basically, the error is decomposed into modes that are charactered by Fourier wave numbers. How these modes are damped during one step of the iterative method provides insight on the effectiveness of the acceleration method. This damping is characterized by the spectral radius, the largest attenuation factor for any wave number. It is important that the initial error (initial guess) contains all error modes. The slowest mode will dominate as the iteration proceeds and its damping rate (spectral radius) will eventually characterize the iteration procedure. The smaller spectral radius, the faster the iterations converge. If the spectral radius is greater than 1, the scheme is unstable.

Our periodic heterogeneous FA was performed on a Cartesian regular geometry, described by the cell widths in x $(\Delta x_1, \Delta x_2, \cdots, \Delta x_{N_x})$ and cell widths in y $(\Delta y_1, \Delta y_2, \cdots, \Delta y_{N_y})$, as shown in Fig. III-4. Each rectangular cell is cut into two triangular cells. Different triangular cells may contain different media. Periodic boundary conditions are applied on the domain boundary. While this geometrical

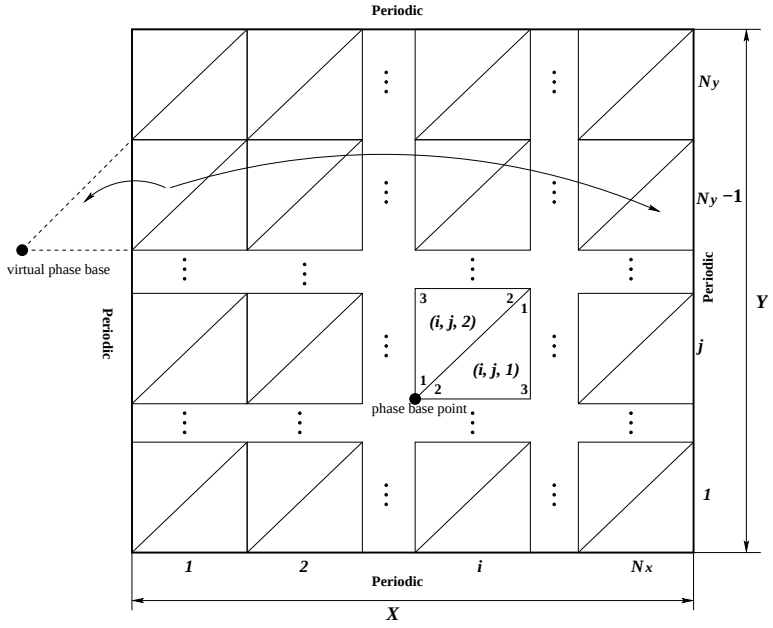


Fig. III-4. Domain for the Fourier analysis.

layout is simple, we were able to perform heterogeneous FA and analyze important problems such as the PHI (Periodic Horizontal Interface [119]) problems and situation where the elements present high aspect ratios. Our FA study was limited to linear shape functions, i.e., DGFEM(1), for simplicity. The transport system of equations is solved by direct inversion of the $\mathbf{L}$ operator in MATLAB, so no SAF are present.

For each error mode with the Fourier wave numbers $\lambda = [\lambda_x, \lambda_y]$, a diagonal phase matrix is associated with each cell $(i, j, 1)$:

$$e^{\jmath(\lambda_x x_i + \lambda_y y_j)} \begin{bmatrix} e^{\jmath(\lambda_x \Delta x + \lambda_y \Delta y)} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & e^{\jmath \lambda_x \Delta x} \end{bmatrix} \mathbf{\Phi}_{i,j,1} \tag{3.114}$$

and a diagonal phase matrix is associated with each cell $(i, j, 2)$:

$$e^{\jmath(\lambda_x x_i+\lambda_y y_j)}\begin{bmatrix}1 & 0 & 0\\ 0 & e^{\jmath(\lambda_x\Delta x+\lambda_y\Delta y)} & 0\\ 0 & 0 & e^{\jmath\lambda_y\Delta y}\end{bmatrix}\mathbf{\Phi}_{i,j,2} \tag{3.115}$$

with $\jmath=\sqrt{-1}$. When assembling $\mathbf{L}$ and $\mathbf{\Sigma}$ for the transport solver, and $\mathbf{A}$ and $\mathbf{B}$ for the diffusion solver, we apply these phase matrices on the corresponding elementary matrices. When assembling the edge coupling matrices in transport and diffusion with an edge located on the domain's boundary, the base phase of the cell of the other side of the periodic boundary edge is not used but instead the actual base of the virtual cell is employed, as shown in Fig. III-4.

Wave numbers are chosen in the interval $\left[0,\frac{2\pi}{X}\right)\otimes\left[0,\frac{2\pi}{Y}\right)$, where X and Y are the domain size in the x and y directions.

Finally, the resulting FA iteration matrix in the case of SI is

$$\mathbf{D}\widetilde{\mathbf{L}}^{-1}\mathbf{M}\widetilde{\mathbf{\Sigma}}, \tag{3.116}$$

where $\widetilde{\ }$ is used to represent a matrix to which the phase transformation was applied. The iteration matrix for SI+DSA is given by

$$\mathbf{I}-(\mathbf{I}+\mathbf{P}\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{B}}\mathbf{R})(\mathbf{I}-\mathbf{D}\widetilde{\mathbf{L}}^{-1}\mathbf{M}\widetilde{\mathbf{\Sigma}}). \tag{3.117}$$

The largest eigenvalue of these two iteration matrices is the spectral radius for a given specific wavelength number. This eigenmode can be calculated easily in MATLAB using the built-in function `eig`.

The Nelder-Mead simplex algorithm is used to find the maximum eigenvalue over all possible wavelength numbers in $\left[0,\frac{2\pi}{X}\right)\otimes\left[0,\frac{2\pi}{Y}\right)$ for a given problem configuration. This maximum will be the global spectral radius of the method for the problem under consideration.

F. Results

1. Fourier Analysis Results

a. Infinite Homogeneous Medium Case

We perform FA for a 1×1 Cartesian geometry (i.e., 2 triangles). The same medium in placed in these two elements. The domain is square, i.e., $X = Y$ and the mesh size X is varied from 2^{-10} MFP to 2^{10} MFP (Mean Free Path). Periodic boundary conditions are applied on all four sides. Scattering is isotropic with a scattering ratio fixed to $c = 0.9999$. Level-Symmetric (LS) angular quadrature sets with S_2, S_4, S_8 and S_{16} are used. Three DSA schemes are analyzed: DCF, IP and MIP. Figs. III-5 through III-7 show the spectral radius obtained using these three schemes. There are four curves on each plot corresponding to the four different angular quadrature sets. The x-axis of these three plots is the mesh size, measured in MFP; the y-axis is the spectral radius.

Both DCF and IP forms are not unconditionally stable: the DCF form is unstable in the intermediate range when the cell is a few MFP thick; the IP form is unstable for thick cells, when their cell sizes corresponds to an edge penalty factor of about 1/4. The MIP form is stable for all cell sizes, with the maximum spectral radius occuring in the intermediate MFP range: the maximum spectral radius is less than 0.5, except for S_2 where it is about 0.7. These results for MIP are very satisfactory and signify that the MIP DSA form is capable of providing good acceleration, at least in the case of an infinite homogeneous problem. Results employing the IP form will not longer be presented, since the MIP form is clearly better. We also note that

1. when the mesh size is very small, the spectral radius is approaching the theoretical value of $0.2247\,c$ (obtained in the case of a continuous (not discretized) DSA accelerator) as the number of directions increases in the angular quadrature. Values obtained for the four different S_N sets are 0.4999, 0.2689, 0.2401 and 0.2322, respectively.

2. when the mesh size is equal to 1, DCF gives a spectral radius of 1. (This is also the case in 1-D.)

3. the M4S method is also not stable in the intermediate MFP range, as reported [70], which is a behavior similar to that of the DCF form presented here.

Several optical thicknesses, marked on Fig. III-8 and numbered 1 through 6 from left to right, are further discussed in the case of DCF. Their 2-D wave number dependencies are plotted in Fig. III-9. The colorbars show the log value of the spectral radius.

The DSA DCF scheme becomes unstable when the cell size is equal to $\sqrt{2}$.

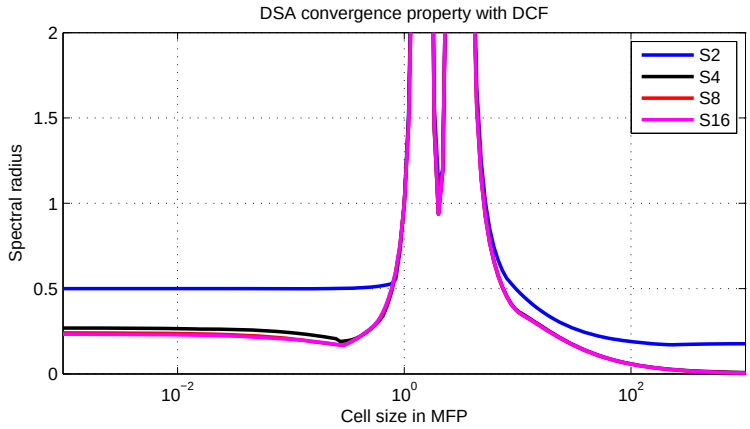


Fig. III-5. Fourier analysis for the DCF form as a function of the mesh optical thickness, homogeneous infinite medium case.

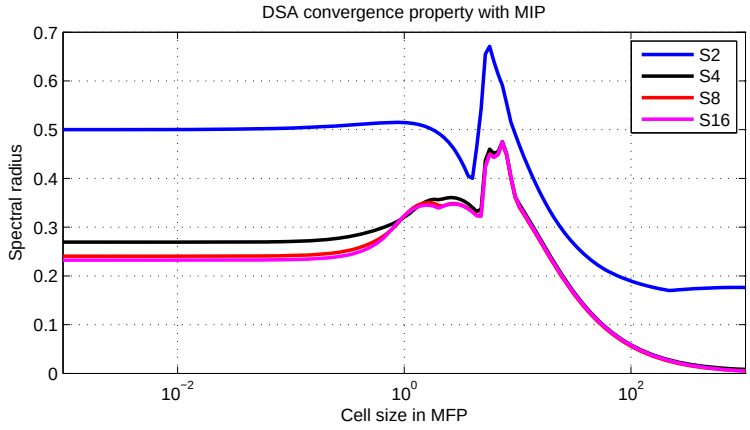


Fig. III-6. Fourier analysis for the MIP form as a function of the mesh optical thickness, homogeneous infinite medium case.

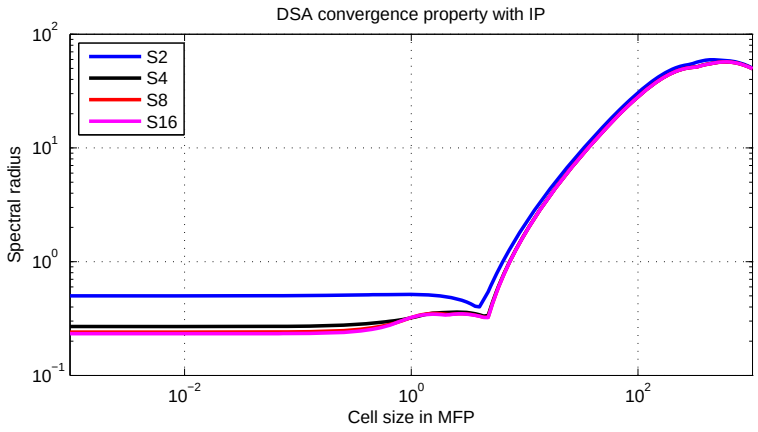


Fig. III-7. Fourier analysis for the IP form as a function of the mesh optical thickness, homogeneous infinite medium case.

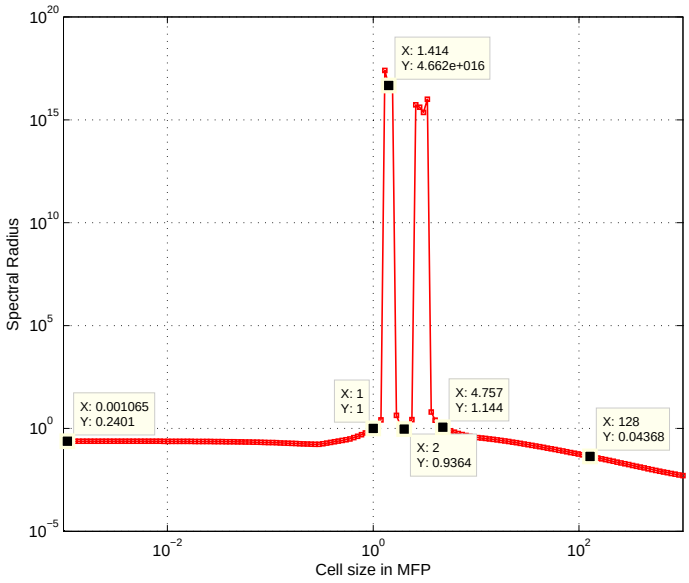
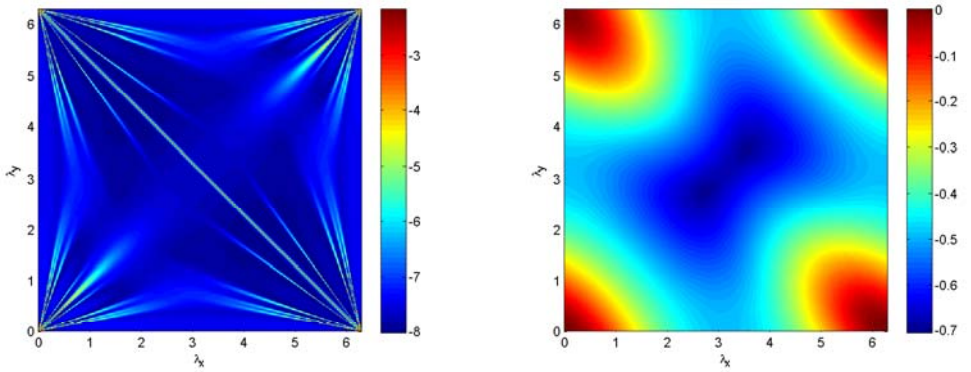
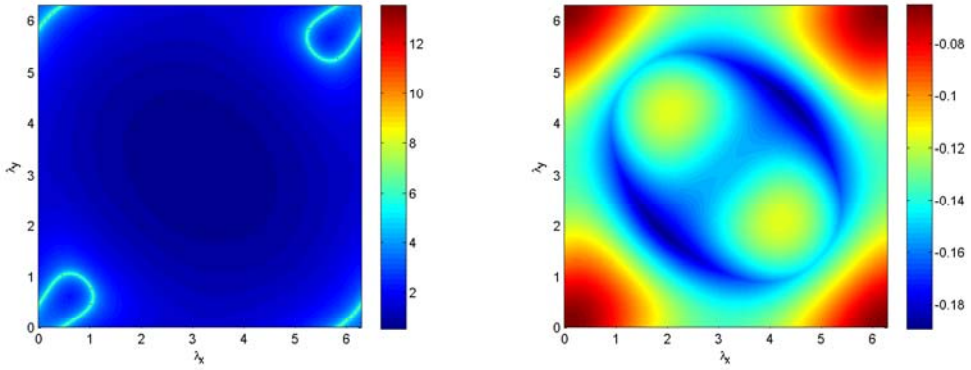


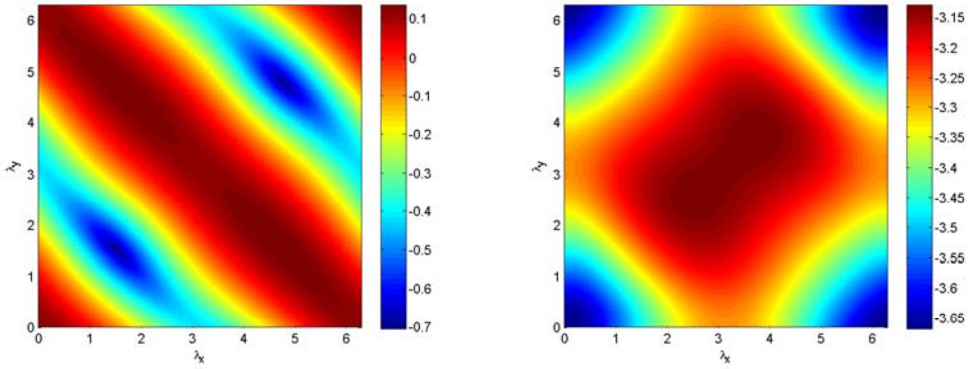
Fig. III-8. FA of the DCF form. Selected 6 points whose 2-D wave number dependencies are plotted in Fig. III-9.



(a) Point #1 (b) Point #2



(c) Point #3 (d) Point #4



(e) Point #5 (f) Point #6

Fig. III-9. 2-D wave number dependencies of the DCF form for the selected 6 optical thicknesses.

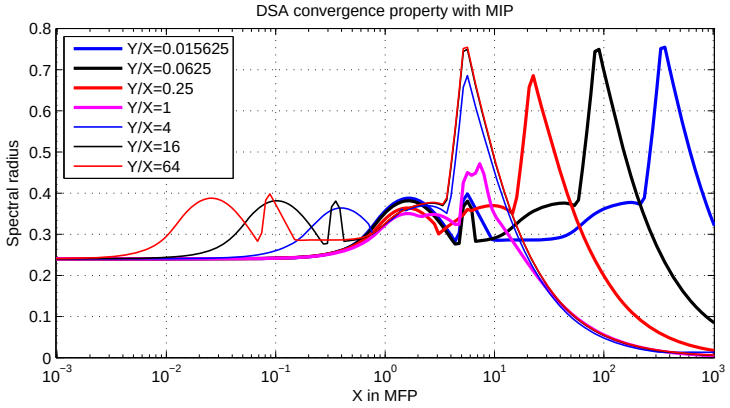


Fig. III-10. Spectral radius for the MIP form with different aspect ratios and using $C = 2$.

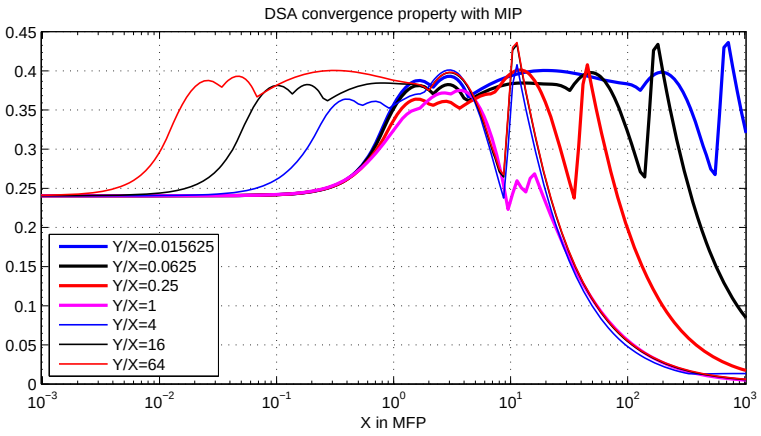


Fig. III-11. Spectral radius for the MIP form with different aspect ratios and using $C = 4$.

Finally, we analyze the MIP form for different cell aspect ratios by fixing $X = 1$ and changing the value of Y. Results are shown in Fig. III-10. They indicate that the MIP scheme converges, even for high aspect ratio cells. It was noted that the performance of the MIP acceleration scheme can be improved by adjusting the C constant in the penalty formula Eq. (3.40) by increasing it from 2 to 4. The Fourier Analysis results with this augmented coefficient are in presented Fig. III-11.

b. Periodic Horizontal Interface (PHI) Problem

The Periodic Horizontal Interface (PHI) problem [119] is a standard litmus test for DSA techniques, notably to assess the effectiveness of the acceleration for highly heterogeneous configurations. The PHI problem consists of horizontal stripes of alternating transparent and highly-diffusive media. In our test, two layers are employed. The first layer is optically thick and the other layer is optically thin. This is achieved by setting $\sigma_{t,1} = \sigma$ and $\sigma_{t,2} = 1/\sigma$ and increasing the value of σ. Strong material discontinuities are present in this problem when σ becomes large, which could potentially reduce the effectiveness of the DSA schemes. Again, different LS quadrature sets are utilized in our results. Various values of scattering ratios are chosen: $c = \{0.9, 0.99, 0.999, 0.9999, 0.99999, 0.999999\}$. The study was conducted with a sequence of $\sigma = \{10, 20, 40, 80, 160, 320, 640\}$. Tables III-I to III-VIII display the spectral radius results for different angular quadratures and different scattering ratios for the following two DSA schemes: MIP and DCF.

It can be seen the MIP DSA form loses effectiveness when the heterogeneity is strong, with a spectral radius tending towards c. However, the maximum spectral radius for the DCF DSA is close to 0.5 for all quadrature sets except for S_2 where it is about 0.69–0.78.

Recall that the edge penalty formula uses the average of the penalties computed

Table III-I. Spectral radius for the PHI problem, DCF form with LS-2.

	Scattering ratios					
σ	0.9	0.99	0.999	0.9999	0.99999	0.999999
10	0.2898	0.7374	0.7812	0.7860	0.7865	0.7865
20	0.3370	0.6572	0.7208	0.7278	0.7285	0.7286
40	0.2282	0.5890	0.6969	0.7105	0.7119	0.7120
80	0.1413	0.5031	0.6731	0.6982	0.7008	0.7011
160	0.0853	0.3974	0.6429	0.6892	0.6947	0.6953
320	0.0474	0.2833	0.5979	0.6806	0.6920	0.6933
640	0.0140	0.1864	0.5293	0.6682	0.6901	0.6929

Table III-II. Spectral radius for the PHI problem, DCF form with LS-4.

	Scattering ratios					
σ	0.9	0.99	0.999	0.9999	0.99999	0.999999
10	0.1724	0.2901	0.3215	0.3251	0.3255	0.3255
20	0.1093	0.2527	0.3388	0.3470	0.3484	0.4656
40	0.0831	0.2518	0.3502	0.3767	0.3796	0.4928
80	0.0543	0.2275	0.3504	0.3924	0.3985	0.5059
160	0.0319	0.1760	0.3479	0.3967	0.4086	0.5102
320	0.0175	0.1203	0.3211	0.3963	0.4130	0.5085
640	0.0092	0.0748	0.2710	0.3912	0.4137	0.5022

Table III-III. Spectral radius for the PHI problem, DCF form with LS-8.

	Scattering ratios					
σ	0.9	0.99	0.999	0.9999	0.99999	0.999999
10	0.1708	0.2772	0.3063	0.3525	0.3100	0.3100
20	0.1161	0.2739	0.3864	0.4074	0.4097	0.4099
40	0.0848	0.2824	0.3991	0.4395	0.4444	0.4449
80	0.0529	0.2479	0.4040	0.4540	0.4636	0.4646
160	0.0302	0.1850	0.3976	0.4571	0.4733	0.4753
320	0.0163	0.1212	0.3605	0.4599	0.4773	0.4808
640	0.0085	0.0723	0.2970	0.4520	0.4793	0.4839

Table III-IV. Spectral radius for the PHI problem, DCF form with LS-16.

	Scattering ratios					
σ	0.9	0.99	0.999	0.9999	0.99999	0.999999
10	0.1736	0.2902	0.3042	0.3074	0.3078	0.3078
20	0.1158	0.2794	0.4134	0.4423	0.4456	0.4460
40	0.0835	0.2891	0.4205	0.4745	0.4814	0.4821
80	0.0515	0.2498	0.4256	0.4869	0.5005	0.5019
160	0.0293	0.1835	0.4152	0.4873	0.5094	0.5123
320	0.0161	0.1186	0.3709	0.4904	0.5122	0.5175
640	0.0085	0.0701	0.3004	0.4792	0.5143	0.5201

Table III-V. Spectral radius for the PHI problem, MIP form with LS-2.

	Scattering ratios					
σ	0.9	0.99	0.999	0.9999	0.99999	0.999999
10	0.7334	0.7854	0.8655	0.8673	0.8675	0.8675
20	0.8029	0.9174	0.9354	0.9375	0.9377	0.9377
40	0.8029	0.9519	0.9574	0.9715	0.9717	0.9718
80	0.8328	0.9690	0.9781	0.9867	0.9870	0.9870
160	0.8640	0.9777	0.9878	0.9935	0.9938	0.9938
320	0.8686	0.9778	0.9947	0.9967	0.9970	0.9970
640	0.8711	0.9817	0.9966	0.9982	0.9985	0.9985

Table III-VI. Spectral radius for the PHI problem, MIP form with LS-4.

	Scattering ratios					
σ	0.9	0.99	0.999	0.9999	0.99999	0.999999
10	0.6910	0.8258	0.8521	0.8551	0.8554	0.8554
20	0.7749	0.9031	0.9272	0.9305	0.9308	0.9309
40	0.8175	0.9460	0.9635	0.9666	0.9670	0.9670
80	0.8430	0.9658	0.9814	0.9838	0.9841	0.9842
160	0.8579	0.9749	0.9902	0.9921	0.9923	0.9924
320	0.8657	0.9796	0.9943	0.9960	0.9963	0.9963
640	0.8697	0.9827	0.9963	0.9979	0.9981	0.9982

Table III-VII. Spectral radius for the PHI problem, MIP form with LS-8.

	Scattering ratios					
σ	0.9	0.99	0.999	0.9999	0.99999	0.999999
10	0.6868	0.8220	0.8541	0.8582	0.8586	0.8586
20	0.7716	0.9013	0.9278	0.9322	0.9327	0.9327
40	0.8167	0.9451	0.9629	0.9671	0.9676	0.9676
80	0.8434	0.9651	0.9811	0.9837	0.9841	0.9837
160	0.8580	0.9744	0.9900	0.9918	0.9922	0.9923
320	0.8656	0.9793	0.9942	0.9959	0.9962	0.9962
640	0.8696	0.9827	0.9962	0.9979	0.9981	0.9981

Table III-VIII. Spectral radius for the PHI problem, MIP form with LS-16.

	Scattering ratios					
σ	0.9	0.99	0.999	0.9999	0.99999	0.999999
10	0.6859	0.8189	0.8543	0.8593	0.8598	0.8599
20	0.7716	0.8996	0.9272	0.9326	0.9332	0.9333
40	0.8123	0.9442	0.9622	0.9671	0.9677	0.9678
80	0.8436	0.9647	0.9810	0.9835	0.9841	0.9838
160	0.8580	0.9742	0.9897	0.9917	0.9922	0.9923
320	0.8656	0.9793	0.9941	0.9958	0.9962	0.9962
640	0.8696	0.9828	0.9962	0.9978	0.9981	0.9981

on two neighboring cells, which may not be the best choice for this problem. The loss of effectiveness of the MIP DSA scheme suggests that the MIP form, though conditionally stable and quite effective for homogeneous configurations, may not yield the speed-ups necessary for highly heterogeneous and diffusive configurations.

2. Results for a Simple 2-Cell Problem

A Matlab code solving a 2-cell problem (shown in Fig. III-12) was written to test the five proposed DSA schemes: DCF, IP, MIP, P1C, P1M. To make the matrix assembly procedure simpler, all four boundaries are reflecting. This code has also been used to perform tests with highly anisotropic scattering; these results are presented later in this Section.

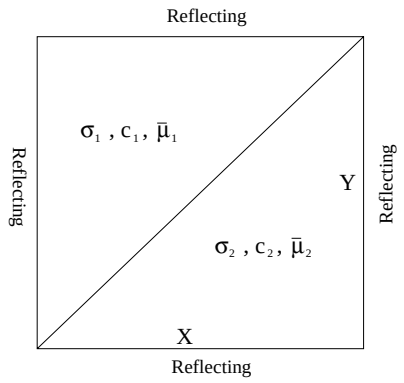


Fig. III-12. Geometry for a simple 2-cell problem.

The geometry is given in Fig. III-12. A rectangular area is cut into 2 triangular cells. The vertices of element 1 and element 2 are numbered so that the interface edge is from vertex 1 to vertex 2. Such numbering makes the r_i ratios of two elements identical,thus greatly simplifying the matrix assembly procedure. The size of the rectangle can be varied by changing X and Y. The total cross section, scattering ratio, and average scattering cosine can be different in the two elements. Only linear

elements (DGFEM(1)) are used, i.e., the unknowns are associated with vertices of two elements. The transport sweep is solved by direct inversion of the {streaming+loss} matrix $\mathbf{L}$, thus avoiding SAF in these calculations.

We first test the DCF with different angular quadratures with $\sigma_1 = \sigma_2 = 1$ cm^{-1}, $c_1 = c_2 = 0.9999$ and isotropic scattering. The domain size X is always equal to Y (square geometry), and its width varies from 2^{-8} cm to 2^{10} cm. The spectral radius is calculated as the maximum eigenvalue of the DSA iteration matrix. As shown in Fig. III-13, the angular quadrature does not have to exactly satisfy Eqs. (3.10) and (3.23) for an effective DSA scheme. The quadrature effect is only noticeable for large cell sizes. Fig. III-13 presents the spectral results using DCF, for which instability occurs for intermediate MFP values of the cell width, as observed earlier. Also note that the spectral radius goes to zero for small cell sizes, a trend that differs from the 2-D Fourier analysis results. The reason could be that some error modes are not present when using reflecting boundary conditions.

In Fig. III-14, we present the spectral radius results for the 5 DSA schemes using the LS-8 angular quadrature (hereafter, when no angular quadratures are specified, the LS-8 is assumed). We notice that the MIP is stable for the entire range of mesh size, with a maximum value of 0.358 attained for $X = 3.364$, and that the difference between P1M and P1C schemes is negligible, except maybe for very small cell sizes.

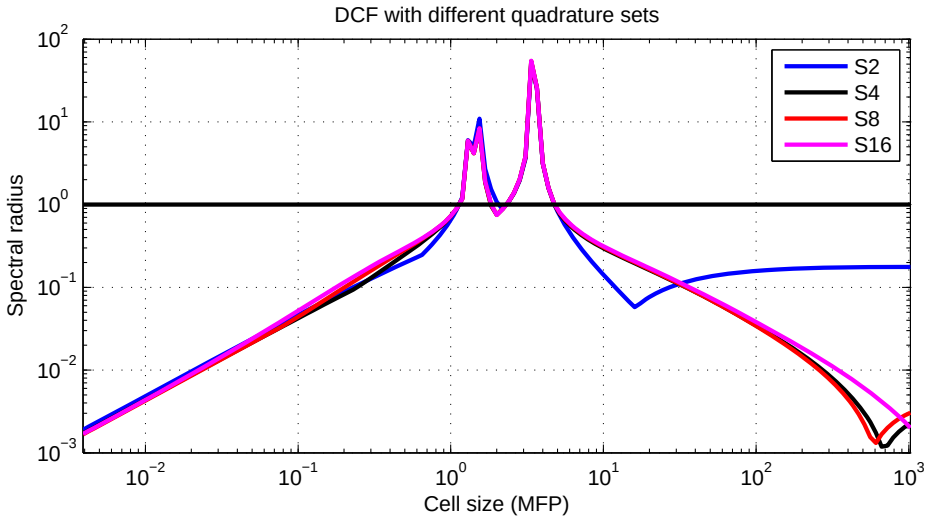


Fig. III-13. Spectral radius of the DCF form for the 2-cell problem with different quadrature sets.

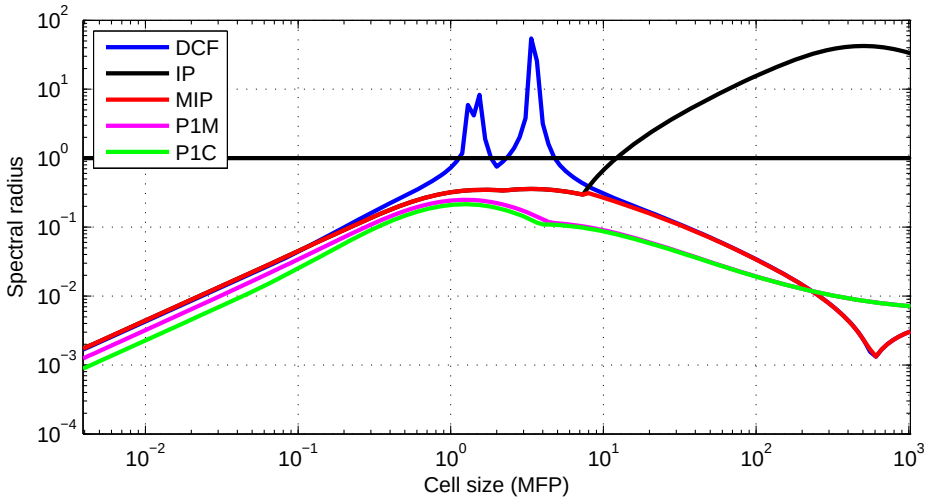


Fig. III-14. Spectral radius of different DSA forms for the 2-cell problem.

Different levels of anisotropic scatterings are presented. Linear anisotropic scattering is modeled using

$$\sigma_{s,1} = \overline{\mu}\sigma_{s,0} \tag{3.118}$$

where various values of $\overline{\mu}$ are chosen to increase the anisotropy. The DCF and MIP forms are tested with and without the $\vec{Q}_1$ terms present (recall that these terms are related to the anisotropic component in Fick's law and appear as anisotropic error source term in the DSA equations.) Without the $\vec{Q}_1$ terms, we observe in Fig. III-15 that the spectral radius is dependent upon the value of the average scattering cosine $\overline{\mu}$ for thick cells. The DCF form converges for thin cells, whereas the MIP form diverges for thin cells when the average scattering cosine is greater than 0.45.

With $\vec{Q}_1$ terms present, the spectral radius does not seem to be limited by the average scattering cosine for thick cells, as shown in Fig. III-16. For thin cells, we have also been able to reproduce exactly the spectral radius values $\frac{\overline{\mu}}{1-\overline{\mu}}$ published in the paper by Adams [120] on the effectiveness of DSA schemes for anisotropic scattering.

Furthermore, to stabilize DSA schemes in highly anisotropic situation, Adams suggested a simple remedy which consisted in performing several SI iterations before accelerating the transport solves with DSA. We have chosen to set the DSA frequency to once every 4 SI solves. The results are given in Fig. III-17. As we expected, convergence is restored when the cell size is small. But for highly forward-peaked scattering, i.e., $\overline{\mu}$ close to 1, the MIP scheme still fails for thick cells (the frequency of DSA acceleration should be further reduced.)

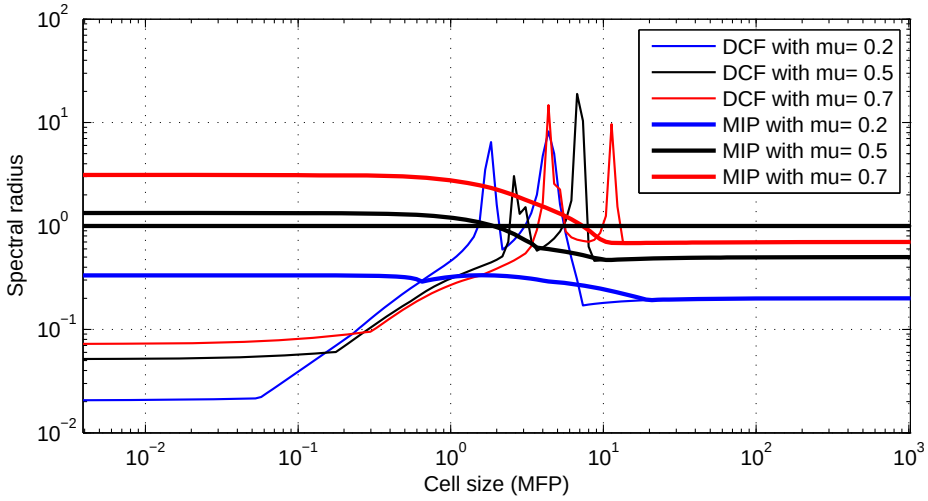


Fig. III-15. Spectral radius of different DSA forms for various degrees of anisotropic scattering without the Q_1 terms.

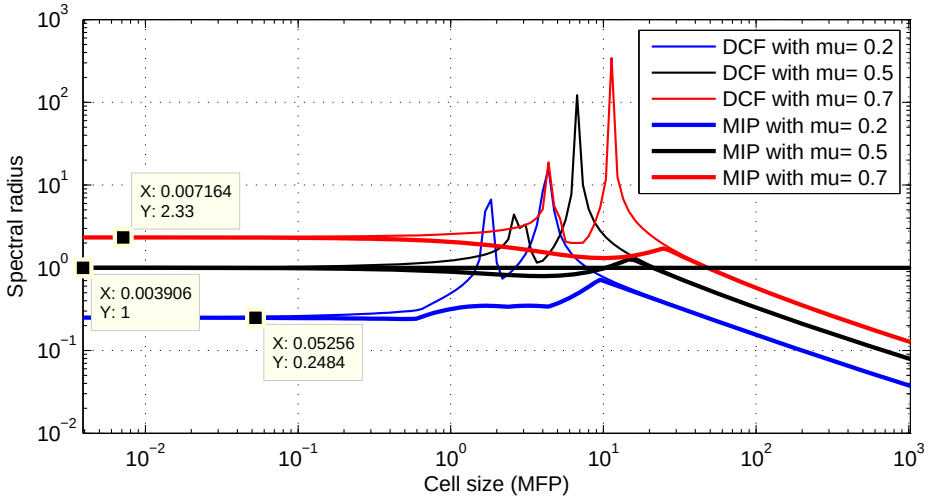


Fig. III-16. Spectral radius of different DSA forms for various degrees of anisotropic scattering with the Q_1 terms.

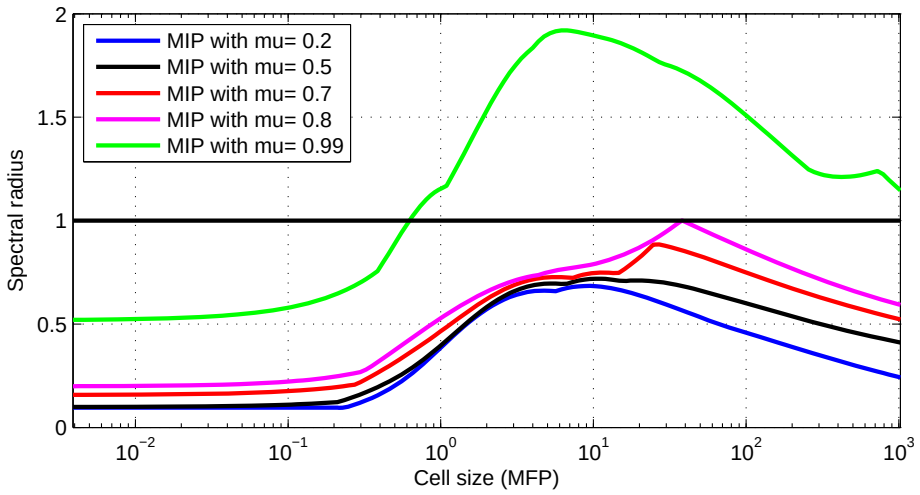


Fig. III-17. Spectral radius of the MIP form for various degrees of anisotropic scattering with Q_1 terms and the "anisotropic trick".

Finally, we have also tested the P1C and P1M schemes for this simple geometrical configuration. Both P1C and P1M provide good acceleration in the case of isotropic scattering (see Fig. III-18 and Fig. III-19 with mu= 0), but they behave differently in the case of anisotropic scattering. P1C is always stable and effective (spectral radius less then 0.5) even with highly forward-peaked anisotropic scattering; see Fig. III-18. P1M fails (similarly to MIP) for thin cells and strong anisotropy, as shown in Fig. III-19. These results suggest that the P1C scheme is superior than the P1M scheme. Further research should be performed regarding the P1C scheme (which is PD.)

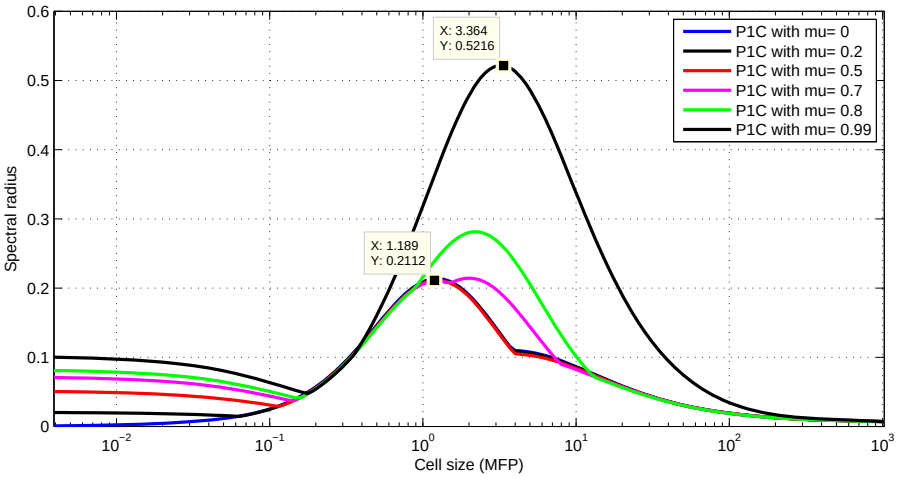


Fig. III-18. Spectral radius of the P1C form with anisotropic scattering.

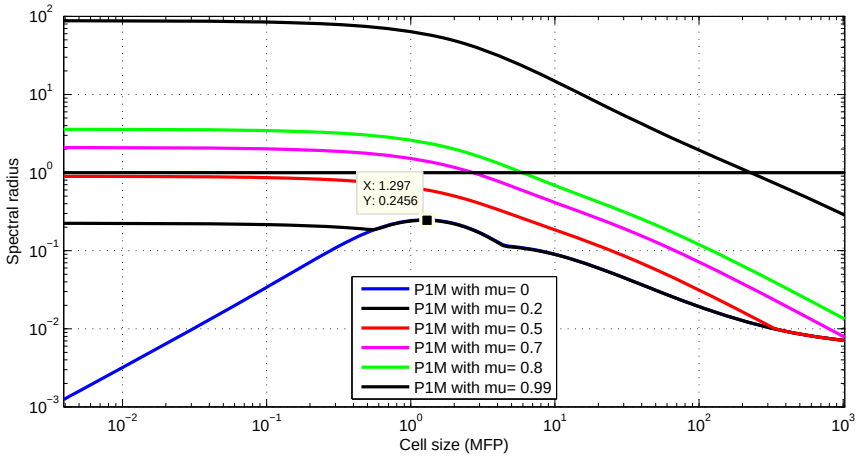


Fig. III-19. Spectral radius of the P1M form with anisotropic scattering.

3. Results Obtained Using XUTHUS

In this Section, we test three DSA schemes implemented in 2-D transport solver XUTHUS: the DSA-IP, the DSA-MIP, and the DSA-DCF. More detailed explanations about XUTHUS can be found in Chapter V. Problems with vacuum and reflecting boundaries, with heterogeneous material configurations and with unstructured irregular meshes stemming from adaptivity are utilized to test these schemes. Both the computing time and the spectral radius are provided.

a. Homogeneous Problem

The first test is a simple homogeneous problem with vacuum boundaries. The computational mesh is shown in Fig. III-20. Equal widths in x and y are used. Scattering is isotropic with a scattering ratio c being equal to 0.9999. All calculations are performed with the LS-8 angular quadrature.

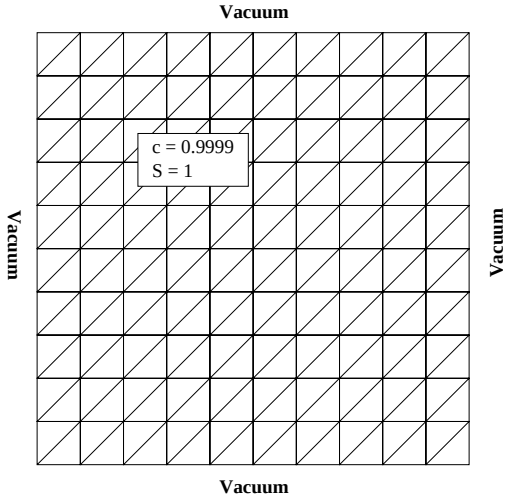


Fig. III-20. Domain of the homogeneous DSA test problem computed with XUTHUS.

We test DCF, IP and MIP as accelerator to SI for a wide range of cell sizes.

For cases with cells thicker than 1 MFP, a fixed number of elements is used. In this situation, the square domain is cut into 200 triangular elements and domain size is fixed to 10 cm $\times$ 10cm. The optical thickness of cells are changed by varying the total cross section σ_t from 1 cm^{-1} to 2^{10} cm^{-1}. For cases with cells thinner than 1 MFP and in order not to let the leakage affect the spectral radius much, we keep the total cross section equal to 1 cm^{-1} and the domain size is unchanged. By doing so, the domain size in MFP does not change, i.e., the leakage through the vacuum boundaries does not change. The cell size is reduced through uniform mesh refinements (during which each element is subdivided into 4 elements). Each refinement cycle decreases the cell size by a factor 2.

The spectral radius data is numerically obtained using the following equation, where $\mathcal{L}$ is the number of SI accomplished.

$$\rho = \begin{cases} \sqrt[4]{\dfrac{\left\|\Phi^{(\mathcal{L})} - \Phi^{(\mathcal{L}-1)}\right\|}{\left\|\Phi^{(\mathcal{L}-4)} - \Phi^{(\mathcal{L}-5)}\right\|}}, & \text{when } \mathcal{L} \geqslant 8 \\ \sqrt[2]{\dfrac{\left\|\Phi^{(\mathcal{L})} - \Phi^{(\mathcal{L}-1)}\right\|}{\left\|\Phi^{(\mathcal{L}-2)} - \Phi^{(\mathcal{L}-3)}\right\|}}, & \text{when } \mathcal{L} < 8 \end{cases} \tag{3.119}$$

The tolerance used in SI, tol_{inner}, is set to 10^{-10} and the maximum number of SI is 20. By doing this, numerical oscillations in calculating the spectral radius can be reduced. We later refer the spectral radius calculated with Eq. (3.119) as the *numerical spectral radius* or NSR.

The first investigation consisted in analyzing the effect of the DSA convergence tolerance, tol_{DSA}, to finely tune the performance of the DSA schemes. Different DSA tolerances are tested and the NSR results are shown in Fig. III-21. The MIP form is solved using PCG with the Eisenstat trick, while the DCF form is solved with the SQMR solver. We note that a coarse tolerance close to 1 is unacceptable because the DSA calculations do not accelerate SI at all. Although some oscillations in the

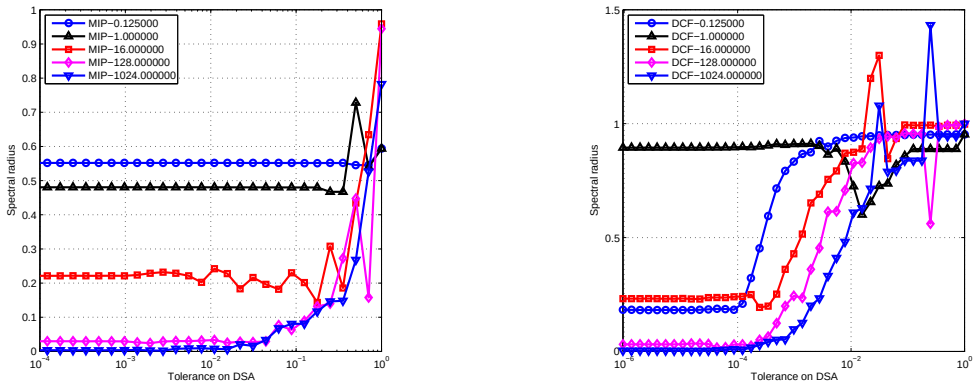


Fig. III-21. Dependence of the numerical spectral radius on the tolerance used in DSA.

convergence history still occur for a tolerance of 0.1 in the MIP form, the NRS are reasonably converged and there is no need to employ a tighter convergence criterion; therefore, it is a sensible choice to use 0.1 with the PCG solver. Due to the different criterion used in the SQMR solver, the right pane of Fig. III-21 shows that the tolerance needs to be set to 10^{-4} for DCF solved using SQMR. We note that because various convergence criteria are available, this type of curves needs to be generated to gain confidence in setting the tolerance criteria for any given iterative solver. For the purposes of generating NSR, we employ more stringent convergence tolerances as follows: 0.001 for the PCG solver and 10^{-6} for the SQMR solver. In routine calculations with XUTHUS or when the CPU time is of concern, we use the previously mentioned tolerances of 0.1 for the PCG solver and 10^{-4} for the SQMR solver.

With this preliminary remark on the convergence tolerances, we can now present the various NSR obtained for the different DSA, see Fig. III-22. It can been seen clearly that XUTHUS produces results very similar to the results obtained with Fourier Analysis. DCF diverges for cell sizes in the intermediate MFP range while IP is un-

stable with thick cells. The NSR of the MIP form is a combination of the NSR of DCF and IP, showing that MIP is stable in the entire range, from optically thin to optically thick cells. The presence of Dirichlet boundaries seems to degrade effectiveness of the MIP scheme for small cell sizes. The NSR for thin cells is now about 0.56, while, in the same range, DCF yields a NSR of 0.20, which is very close to the theoretical spectral radius. For large cell sizes, the NSR from MIP and DCF are almost identical. However, if we keep the additional factor 2 in the penalty coefficient for vacuum boundaries as in the IP form, the MIP spectral radius is significantly larger for thick cells, Fig. III-22.

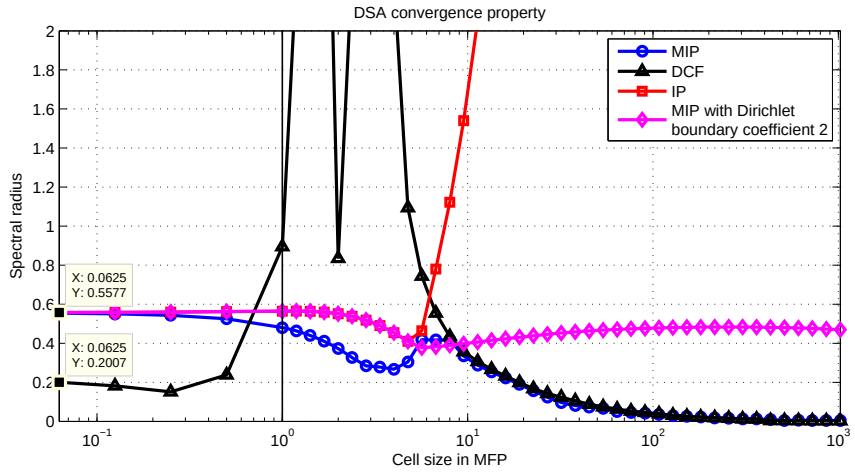


Fig. III-22. Numerical spectral radius computed with XUTHUS for various DSA schemes.

The effect of polynomial order on the MIP form is analyzed (recall that the penalty coefficient in the MIP quadratically depends on the polynomial order p.) For polynomial orders 1 through 4, the NSR results for DSA-MIP are plotted in Fig. III-23. The DSA MIP form is stable for all polynomial orders. By increasing the constant C in the penalty formula Eq. (3.40) to 4, the results shown in Fig. III-24 are obtained

and we can conclude that the default value of $C = 2$ is a good choice for the MIP stabilization terms.

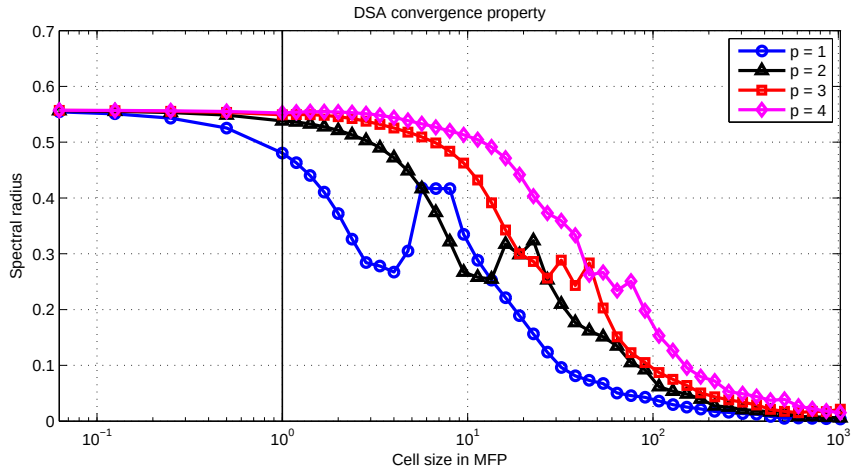


Fig. III-23. Numerical spectral radius computed with XUTHUS for the MIP form using different polynomial orders.

The next tests deal with the effect of reflecting boundaries. The MIP and DCF forms are used for a problem that is one quarter of the previous homogeneous problem, hence of size 5 cm $\times$ 5 cm with 50 triangles and reflecting boundaries on the left and bottom sides. The spectral radius data is shown in Fig. III-25, along with the data from the original homogeneous problem that used only Dirichlet boundaries. We see that neglecting the higher moment terms on the reflecting boundaries does not degrade the performance of the MIP form but DCF does not work as expected with reflecting boundaries for thin cells.

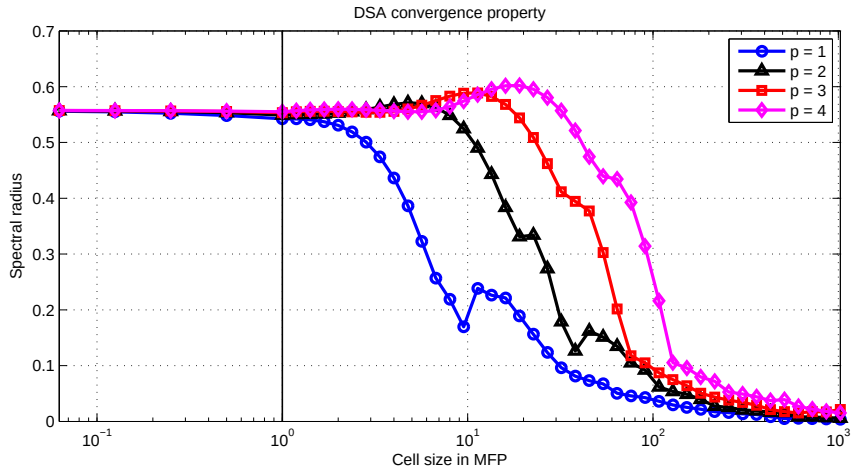


Fig. III-24. Convergence with different polynomial orders for the MIP form using $C = 4$.

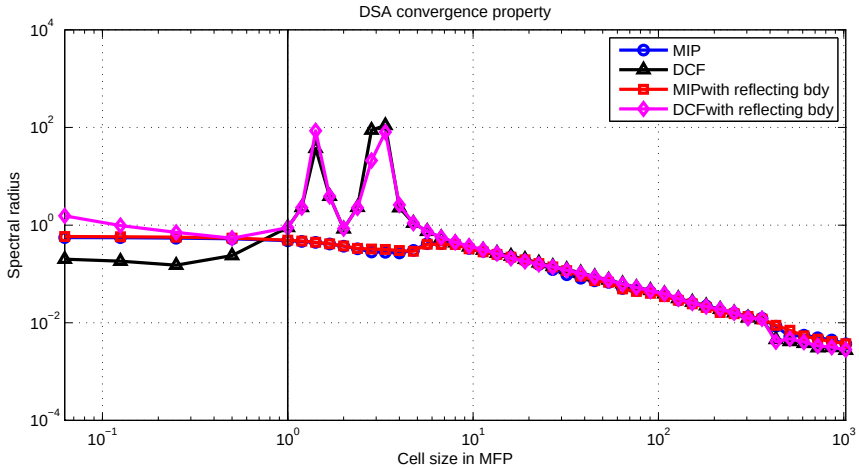


Fig. III-25. Spectral radius with reflecting boundaries.

b. Heterogeneous Problem

The CPU time for the MIP form is studied for a heterogeneous problem. This problem is a simplified version of a shielding problem. The external source and the scattering medium are separated by two shielding blocks and connected with a void channel. In this problem, the two shielding blocks are treated as a strong pure absorber, while the total cross section of the void channel is significantly smaller than the total cross section of the adjacent scattering and shielding medium. The geometrical descriptions can be found in Fig. III-26. The domain is triangularized with Triangle and the mesh is shown in the right pane of Fig. III-26.

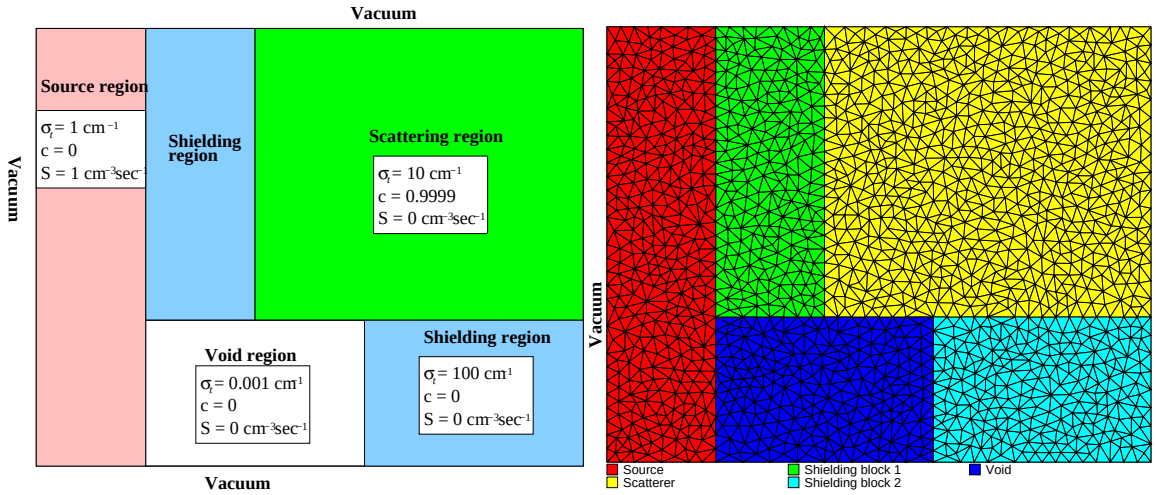


Fig. III-26. Non-homogeneous DSA test problem calculated with XUTHUS (left) and its initial mesh (right).

We are interested in how the choice of angular quadrature and the mesh refinement (both h- and p versions) affect the fraction of time spent in DSA with the MIP form. The fraction of DSA time is plotted in Fig. III-27 with different polynomial orders and different uniform mesh refinement levels.

The first plot in Fig. III-27 shows how uniform p-refinement affects the performance of DSA. The other three graphs correspond to three polynomial orders, from

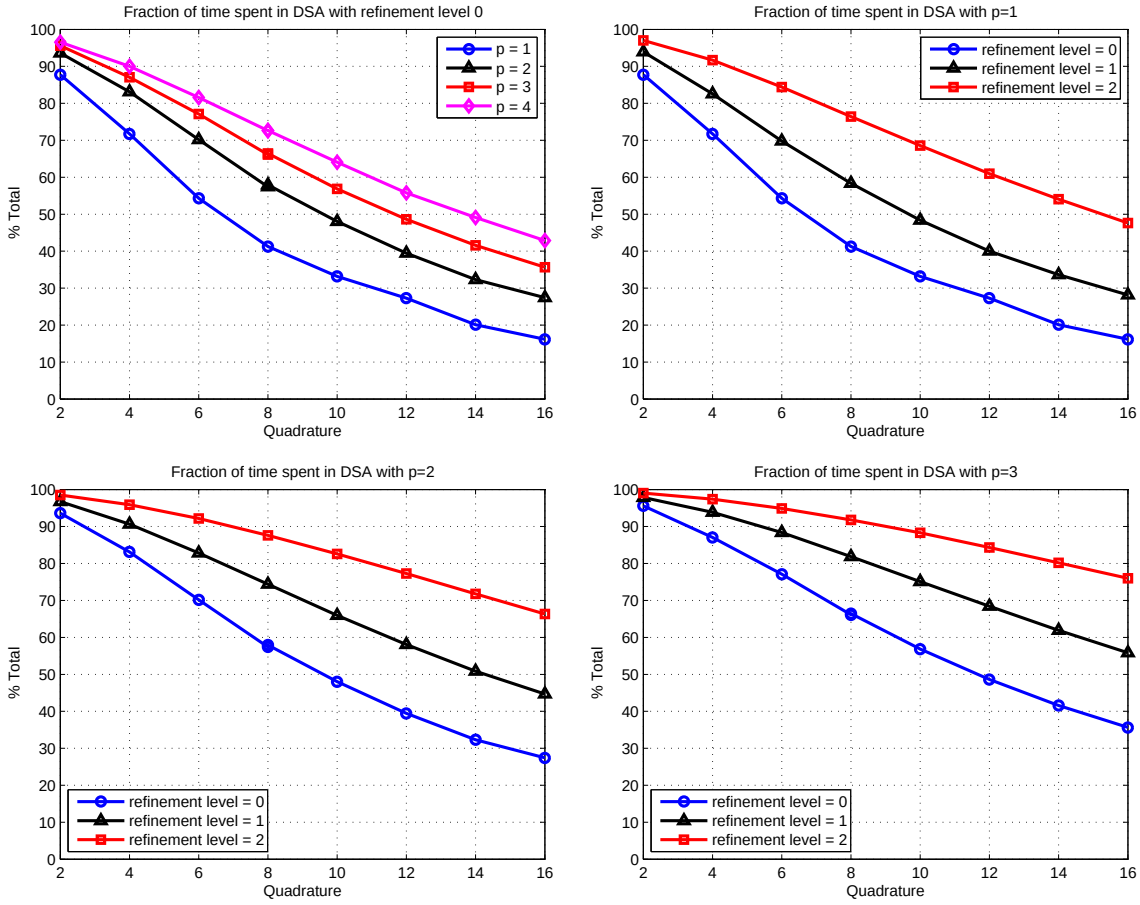


Fig. III-27. Fraction of time spent in DSA.

1 to 3, with different different uniform refinement levels. Refinement level 0 stands for the original mesh, shown in the right pane of Fig. III-26. After two levels of h-refinement, the time spent in DSA has about doubled with respect to p. Note that 0.1 tolerance on DSA is used for all calculations. Because the condition number of the global diffusion matrix $\mathbf{A}$ increases with p- and h-refinement, more iterations are required for the same DSA tolerance, although the time ratio of a diffusion sweep and a transport sweep does not change. These two figures confirmed this fact. The effect of DSA with MIP can be see in Table III-IX. We can note that MIP reduces the spectral radius from about 0.96 to 0.55. It is clear that MIP is effective for this heterogeneous problem.

Table III-IX. NSR with MIP for the heterogeneous problem.

(left number: NSR for SI, right number: NSR for SI+DSA)

Polynomial order	Initial mesh		Once uniformly refined mesh		Twice uniformly refined mesh	
1	0.9587	0.5542	0.9588	0.5319	0.9588	0.5149
2	0.9589	0.5291	0.9588	0.4885	0.9588	0.5213
3	0.9588	0.5293	0.9588	0.5430	0.9588	0.5199
4	0.9588	0.5301	0.9588	0.5455	0.9588	0.5088

The effect of using DSA-MIP as a preconditioner for the GMRes solver is tested and the results are shown in Table III-X, where the number of unpreconditioned and preconditioned GMRes iterations needed to reduce the error below 10^{-8} is given for different polynomial orders and different refinement cycles. Using DSA as a preconditioner reducing the number of GMRes iterations by a factor of 6.

Table III-X. Number of GMRes iterations with MIP for the heterogeneous problem.

(left number: GMRes, right number: DSA-preconditioned GMRes)

Polynomial order	Initial mesh		One uniformly refined mesh		two uniformly refined mesh	
1	55	9	56	9	55	9
2	55	9	55	9	55	9
3	54	9	54	9	55	9
4	54	10	55	9	55	9

c. Problem with Hanging Nodes

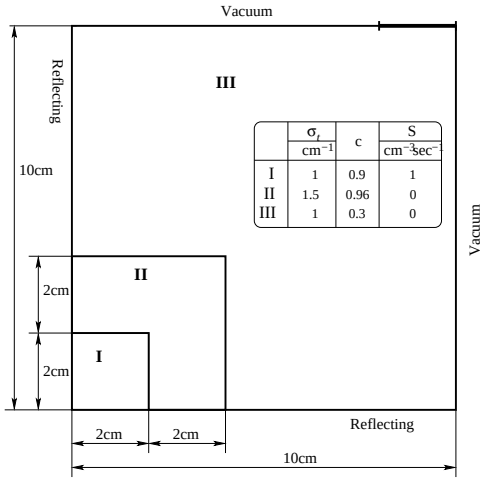


Fig. III-28. Geometry and material description.

The DGFEM diffusion schemes derived earlier can be used in a natural fashion with *hp*-type unstructured meshes. The performance of the MIP form on irregular meshes from AMR is analyzed using a simple transport benchmark problem. The geometry and material descriptions are shown in Fig. III-28. The initial computa-

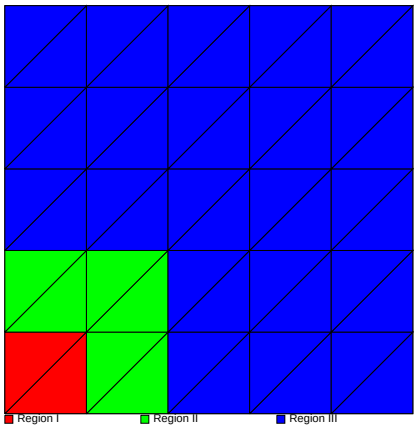


Fig. III-29. Initial mesh.

tional mesh is regular, as shown in Fig. III-29. The polynomial order used is 2 for all elements; the angular quadrature is LS-4; scattering is isotropic. The SI tolerance is set to $\text{tol}_{inner} = 10^{-8}$ and the AMR control parameter is $\alpha = 1/3$ (details regarding this parameter can be found in Section 3 of Chapter IV.) Different mesh irregularity settings are also tested: 1-irregularity, 2-irregularity and 3-irregularity. Here n-irregularity means the maximum refinement-level difference between two neighboring elements can be at most n. It is expected that a higher mesh irregularity is preferred for hyperbolic problems which can present strong singularities, but the point-wise errors at the irregular points, i.e., the hanging nodes, are significantly larger than the errors elsewhere in the domain. Thus, DSA could potentially lose accuracy due to this effect. For each mesh irregularity, two AMR runs driven by the projection-based error estimator $\mu_{g,K}^{k,ref}$ in Eq. (4.7) are presented: one run is performed using solution bootstrapping after each cycle of mesh adaptation, i.e., the numerical solution of the previous adapted mesh is projected onto the newly prescribed adapted mesh as an initial guess; the other run is performed without bootstrapping, i.e., the initial guess is reset to zero at each mesh adaptivity iteration.

Six tables for different mesh irregularities and different bootstrapping strategies were generated and are displayed as Tables III-XI through III-XVI. A total of 25 mesh adaptivity cycles are performed in each case. The second column in these tables shows the number of active elements at each adaptivity cycle. The third column is a mesh parameter measuring the mesh irregularity; this *irregularity index* is the average number of hanging nodes on all interior active edges. "Interior" in that context means that edges on boundaries or on sub-domain interface are not counted. An interior edge is regarded as active when following two conditions are met:

1. at least one neighboring element is active;

2. two neighboring elements have the same refinement level.

Here, the neighboring element of an edge is defined as the element containing the largest portion of that edge.

We can note that the NSR is decreasing as the AMR progresses and that the number of DSA iterations is significantly smaller with bootstrapping than with reinitialization. As the result, the total computing time with AMR is significantly reduced. The DSA preconditioner performs efficiently with AMR. The mesh irregularity index with the 2-irregularity constraint is larger than in the 1-irregularity case, while the difference between the results using a 2-irregularity or a 3-irregularity constraint is small. The resulting meshes are slightly different due to a slightly different convergence history for SI.

Several selected meshes marked in Table III-XV from the 2-irregularity run without bootstrapping are showed in Fig. III-30.

Table III-XI. Results with bootstrapping and 1-irregularity.

Cycle ID	Number of active cells	Irregularity index	NSR	Number of DSA iterations	CPU time in DSA (sec)	Time fraction of DSA	Relative error of scalar flux
0	50	0.0000	0.3157	75	0.01	0.500	1.167×10^{-2}
1	74	0.0412	0.2437	75	0.01	0.625	4.337×10^{-3}
2	104	0.0725	0.3408	65	0.02	0.529	2.624×10^{-3}
3	122	0.1118	0.3663	60	0.02	0.526	1.961×10^{-3}
4	206	0.1624	0.3205	90	0.05	0.667	9.156×10^{-4}
5	278	0.1220	0.3400	91	0.07	0.720	4.495×10^{-4}
6	356	0.1148	0.3105	112	0.11	0.711	3.593×10^{-4}
7	500	0.1759	0.3193	108	0.15	0.739	2.374×10^{-4}
8	824	0.1665	0.2701	92	0.21	0.732	1.284×10^{-4}
9	1136	0.1582	0.3687	150	0.46	0.790	9.440×10^{-5}
10	1586	0.1494	0.2276	73	0.32	0.703	5.768×10^{-5}
11	2216	0.1619	0.2833	138	0.83	0.787	3.962×10^{-5}
12	3188	0.1670	0.2038	61	0.56	0.695	2.521×10^{-5}
13	3974	0.1663	0.2071	65	0.75	0.708	1.930×10^{-5}
14	5318	0.1659	0.2118	59	1.00	0.684	1.430×10^{-5}
15	7004	0.1708	0.1983	56	1.31	0.666	1.054×10^{-5}
16	8918	0.1755	0.2014	45	1.41	0.613	7.725×10^{-6}
17	11492	0.1831	0.1994	34	1.46	0.551	6.228×10^{-6}
18	14492	0.1793	0.1343	19	1.09	0.469	4.788×10^{-6}
19	18296	0.1799	0.1359	23	1.65	0.512	3.650×10^{-6}
20	23522	0.1861	0.1408	22	2.01	0.503	2.986×10^{-6}
21	28214	0.1813	0.1393	23	2.52	0.513	2.407×10^{-6}
22	37394	0.1823	0.1350	25	3.63	0.533	1.811×10^{-6}
23	46940	0.1858	0.1352	22	4.13	0.507	1.424×10^{-6}
24	54146	0.1858	0.0977	17	3.64	0.507	1.222×10^{-6}

Table III-XII. Results with bootstrapping and 2-irregularity.

Cycle ID	Number of active cells	Irregularity index	NSR	Number of DSA iterations	CPU time in DSA (sec)	Time fraction of DSA	Relative error of scalar flux
0	50	0.0000	0.3157	75	0.01	0.500	1.167×10^{-2}
1	74	0.0412	0.2437	75	0.01	0.583	4.337×10^{-3}
2	104	0.0725	0.3408	65	0.02	0.500	2.624×10^{-3}
3	122	0.1118	0.3663	60	0.02	0.513	1.961×10^{-3}
4	206	0.1624	0.3205	90	0.05	0.684	9.156×10^{-4}
5	278	0.1220	0.3400	91	0.07	0.676	4.495×10^{-4}
6	356	0.1148	0.3105	112	0.11	0.745	3.593×10^{-4}
7	500	0.1759	0.3193	108	0.15	0.740	2.374×10^{-4}
8	794	0.1693	0.2917	93	0.20	0.739	1.342×10^{-4}
9	1112	0.1705	0.2790	117	0.35	0.772	8.982×10^{-5}
10	1526	0.1506	0.2268	71	0.30	0.710	5.693×10^{-5}
11	2174	0.1653	0.2786	142	0.84	0.804	3.916×10^{-5}
12	2942	0.1660	0.2022	65	0.53	0.719	2.592×10^{-5}
13	3716	0.1730	0.2051	67	0.71	0.728	1.965×10^{-5}
14	5144	0.1875	0.2063	67	1.07	0.713	1.430×10^{-5}
15	6848	0.1818	0.1978	38	0.85	0.588	9.766×10^{-6}
16	8096	0.1850	0.2022	28	0.78	0.516	7.737×10^{-6}
17	10328	0.1953	0.2038	38	1.35	0.579	6.113×10^{-6}
18	12746	0.1955	0.1401	19	0.89	0.471	4.651×10^{-6}
19	15476	0.1979	0.1365	25	1.39	0.530	3.654×10^{-6}
20	20834	0.2104	0.1402	27	2.03	0.549	2.758×10^{-6}
21	26060	0.2175	0.1477	22	2.26	0.505	2.147×10^{-6}
22	31748	0.2123	0.1414	25	3.08	0.533	1.697×10^{-6}
23	41138	0.2103	0.1461	26	4.18	0.543	1.284×10^{-6}
24	45116	0.2082	0.1032	19	3.37	0.533	1.173×10^{-6}

Table III-XIII. Results with bootstrapping and 3-irregularity.

Cycle ID	Number of active cells	Irregularity index	NSR	Number of DSA iterations	CPU time in DSA (sec)	Time fraction of DSA	Relative error of scalar flux
0	50	0.0000	0.3157	75	0.01	0.500	1.167×10^{-2}
1	74	0.0412	0.2437	75	0.01	0.609	4.337×10^{-3}
2	104	0.0725	0.3408	65	0.02	0.529	2.624×10^{-3}
3	122	0.1118	0.3663	60	0.02	0.564	1.961×10^{-3}
4	206	0.1624	0.3205	90	0.05	0.671	9.156×10^{-4}
5	278	0.1220	0.3400	91	0.07	0.699	4.495×10^{-4}
6	356	0.1148	0.3105	112	0.11	0.720	3.593×10^{-4}
7	500	0.1759	0.3193	108	0.15	0.744	2.374×10^{-4}
8	794	0.1693	0.2917	93	0.21	0.736	1.342×10^{-4}
9	1112	0.1705	0.2790	117	0.36	0.778	8.982×10^{-5}
10	1526	0.1506	0.2268	71	0.31	0.710	5.693×10^{-5}
11	2162	0.1663	0.2755	141	0.84	0.793	3.918×10^{-5}
12	2942	0.1660	0.2021	65	0.55	0.714	2.595×10^{-5}
13	3710	0.1737	0.2055	67	0.73	0.718	1.968×10^{-5}
14	5108	0.1871	0.2069	68	1.10	0.715	1.432×10^{-5}
15	6764	0.1838	0.2011	54	1.25	0.663	9.852×10^{-6}
16	8006	0.1883	0.2026	34	0.97	0.557	7.778×10^{-6}
17	10256	0.1968	0.2041	38	1.42	0.582	6.117×10^{-6}
18	12656	0.1992	0.1401	20	0.98	0.483	4.663×10^{-6}
19	15332	0.2003	0.1364	25	1.49	0.533	3.644×10^{-6}
20	20384	0.2116	0.1410	28	2.19	0.561	2.743×10^{-6}
21	24626	0.2161	0.1458	23	2.24	0.517	2.205×10^{-6}
22	27266	0.2167	0.1458	25	2.72	0.536	1.948×10^{-6}
23	33038	0.2157	0.1497	25	3.29	0.536	1.526×10^{-6}
24	42770	0.2180	0.1027	21	3.53	0.558	1.138×10^{-6}

Table III-XIV. Results with reinitialization and 1-irregularity.

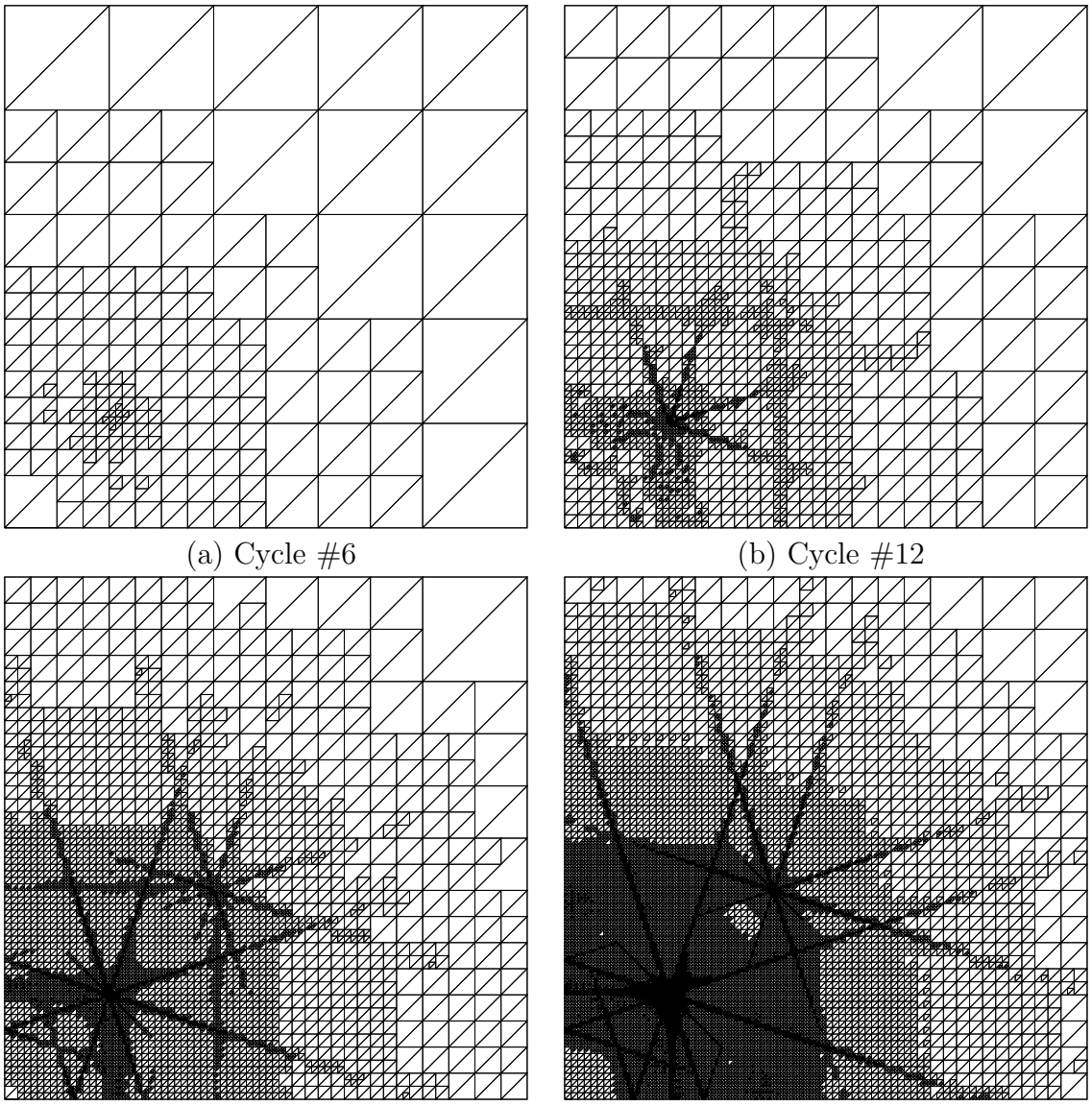
Cycle ID	Number of active cells	Irregularity index	NSR	Number of DSA iterations	CPU time in DSA (sec)	Time fraction of DSA	Relative error of scalar flux
0	50	0.0000	0.3157	75	0.01	0.478	1.167×10^{-2}
1	74	0.0412	0.3181	128	0.03	0.634	4.337×10^{-3}
2	104	0.0725	0.3108	133	0.04	0.656	2.624×10^{-3}
3	122	0.1118	0.3372	151	0.05	0.662	1.961×10^{-3}
4	206	0.1624	0.3046	238	0.13	0.749	9.156×10^{-4}
5	278	0.1220	0.3204	282	0.21	0.787	4.495×10^{-4}
6	356	0.1148	0.3411	318	0.29	0.806	3.593×10^{-4}
7	500	0.1759	0.3422	367	0.48	0.810	2.374×10^{-4}
8	824	0.1665	0.3504	469	1.02	0.849	1.284×10^{-4}
9	1136	0.1582	0.3464	545	1.62	0.868	9.440×10^{-5}
10	1586	0.1494	0.3441	657	2.76	0.874	5.768×10^{-5}
11	2216	0.1619	0.3593	814	4.74	0.895	3.962×10^{-5}
12	3188	0.1670	0.3957	957	8.10	0.912	2.521×10^{-5}
13	3974	0.1663	0.3576	1032	10.81	0.919	1.930×10^{-5}
14	5318	0.1659	0.3731	1194	17.52	0.923	1.430×10^{-5}
15	7004	0.1708	0.3761	1310	26.96	0.925	1.054×10^{-5}
16	8918	0.1755	0.3687	1561	42.92	0.932	7.725×10^{-6}
17	11486	0.1831	0.3844	1849	71.43	0.938	6.230×10^{-6}
18	14492	0.1793	0.3857	2162	107.53	0.947	4.788×10^{-6}
19	18296	0.1799	0.3795	2371	147.29	0.952	3.650×10^{-6}
20	23516	0.1860	0.3571	2617	204.23	0.957	2.986×10^{-6}
21	28226	0.1814	0.3788	3084	290.09	0.963	2.408×10^{-6}
22	37382	0.1824	0.3646	3266	412.18	0.965	1.813×10^{-6}
23	46994	0.1858	0.3750	3375	537.60	0.966	1.425×10^{-6}
24	54194	0.1858	0.4286	3819	703.51	0.969	1.223×10^{-6}

Table III-XV. Results with reinitialization and 2-irregularity.

Cycle ID	Number of active cells	Irregularity index	NSR	Number of DSA iterations	CPU time in DSA (sec)	Time fraction of DSA	Relative error of scalar flux
0	50	0.0000	0.3157	75	0.01	0.435	1.167×10^{-2}
1	74	0.0412	0.3181	128	0.03	0.619	4.337×10^{-3}
2	104	0.0725	0.3108	133	0.04	0.655	2.624×10^{-3}
3	122	0.1118	0.3372	151	0.05	0.636	1.961×10^{-3}
4	206	0.1624	0.3046	238	0.14	0.763	9.156×10^{-4}
5	278	0.1220	0.3204	282	0.21	0.791	4.495×10^{-4}
*6	356	0.1148	0.3411	318	0.30	0.809	3.593×10^{-4}
7	500	0.1759	0.3422	367	0.51	0.817	2.374×10^{-4}
8	794	0.1693	0.3403	464	1.00	0.852	1.342×10^{-4}
9	1112	0.1705	0.3430	546	1.64	0.874	8.982×10^{-5}
10	1526	0.1506	0.3964	637	2.61	0.888	5.693×10^{-5}
11	2174	0.1653	0.3471	782	4.61	0.892	3.916×10^{-5}
*12	2942	0.1660	0.3465	890	7.12	0.906	2.592×10^{-5}
13	3716	0.1730	0.3713	1005	10.30	0.909	1.965×10^{-5}
14	5144	0.1875	0.3454	1124	17.62	0.918	1.430×10^{-5}
15	6848	0.1818	0.3583	1286	27.96	0.926	9.766×10^{-6}
16	8096	0.1850	0.3715	1546	40.71	0.934	7.738×10^{-6}
17	10328	0.1953	0.3232	1617	56.29	0.939	6.113×10^{-6}
*18	12746	0.1955	0.3830	2021	88.13	0.946	4.651×10^{-6}
19	15470	0.1976	0.3281	2035	108.61	0.951	3.655×10^{-6}
20	20822	0.2100	0.3595	2401	175.32	0.958	2.758×10^{-6}
21	26060	0.2170	0.3610	2499	230.80	0.960	2.147×10^{-6}
22	31712	0.2120	0.3869	2862	322.72	0.961	1.700×10^{-6}
23	41210	0.2108	0.3707	3437	493.13	0.968	1.283×10^{-6}
*24	45140	0.2081	0.3746	3318	526.14	0.967	1.172×10^{-6}

Table III-XVI. Results with reinitialization and 3-irregularity.

Cycle ID	Number of active cells	Irregularity index	NSR	Number of DSA iterations	CPU time in DSA (sec)	Time fraction of DSA	Relative error of scalar flux
0	50	0.0000	0.3157	75	0.01	0.435	1.167×10^{-2}
1	74	0.0412	0.3181	128	0.03	0.619	4.337×10^{-3}
2	104	0.0725	0.3108	133	0.04	0.638	2.624×10^{-3}
3	122	0.1118	0.3372	151	0.05	0.697	1.961×10^{-3}
4	206	0.1624	0.3046	238	0.13	0.753	9.156×10^{-4}
5	278	0.1220	0.3204	282	0.21	0.787	4.495×10^{-4}
6	356	0.1148	0.3411	318	0.30	0.806	3.593×10^{-4}
7	500	0.1759	0.3422	367	0.50	0.812	2.374×10^{-4}
8	794	0.1693	0.3403	464	1.00	0.851	1.342×10^{-4}
9	1112	0.1705	0.3430	546	1.64	0.876	8.982×10^{-5}
10	1526	0.1506	0.3964	637	2.59	0.889	5.693×10^{-5}
11	2162	0.1663	0.3759	769	4.49	0.893	3.918×10^{-5}
12	2942	0.1660	0.3464	890	7.06	0.909	2.595×10^{-5}
13	3710	0.1737	0.3727	1005	10.56	0.913	1.968×10^{-5}
14	5108	0.1871	0.2838	1090	16.43	0.926	1.432×10^{-5}
15	6764	0.1838	0.3713	1248	25.91	0.924	9.852×10^{-6}
16	8006	0.1883	0.3740	1529	38.67	0.933	7.779×10^{-6}
17	10256	0.1968	0.3178	1663	56.12	0.941	6.117×10^{-6}
18	12656	0.1992	0.3810	2000	85.25	0.945	4.664×10^{-6}
19	15332	0.2003	0.3338	2021	105.80	0.951	3.645×10^{-6}
20	20384	0.2116	0.3680	2354	167.26	0.958	2.743×10^{-6}
21	24632	0.2160	0.3728	2614	225.27	0.958	2.203×10^{-6}
22	27104	0.2180	0.3773	2745	262.45	0.960	1.962×10^{-6}
23	34004	0.2165	0.4090	3110	364.24	0.964	1.479×10^{-6}
24	42044	0.2138	0.3683	3135	455.50	0.964	1.155×10^{-6}



(a) Cycle #6 (b) Cycle #12

(c) Cycle #18 (d) Cycle #24

Fig. III-30. Regular and irregular unstructured meshes.

G. Conclusions

In this chapter, we derived DSA conforming schemes for unstructured *hp*-type meshes with DGFEM using a variational principle. Both vacuum and reflecting boundaries are treated consistently with significant angular fluxes (SAF). We tested and analyzed five different schemes: DCF, MIP, IP, P1C and P1M. In conclusion,

1. All of these schemes can be used with AMR on unstructured meshes.

2. DCF is symmetric but not positive definite; like M4S schemes, it is unstable in the intermediate MFP range, although it works very well in the thin and thick cell limits. The reason why DCF is unstable has not been understood so far. DCF seems to work well with strong material discontinuities.

3. MIP is SPD, thus can be efficiently solved with PCG. It is stable over the entire MFP range; the presence of Dirichlet boundaries, i.e., vacuum boundaries, can degrade the performance of MIP slightly; like the WLA scheme, MIP loses its effectiveness when strong material discontinuities present. MIP behaves well for all polynomial orders and with distorted cells.

4. IP is not recommended because it is quite similar to MIP for thin cells but diverges for large optical thicknesses.

5. Although P1C has not been implemented with XUTHUS and no FA analysis was conducted so far, results based on the simple 2-cell problem suggest that it may be unconditionally stable and effective, even with strong anisotropic scattering. P1C is positive definite but is not symmetric, which can pose an issue for the numerical solution technique.

6. P1M is obtained directly by discretizing the P_1 equations with DGFEM, as

proposed by Warsa and Morel. Results from the 2-cell problem suggest that can loss effectiveness when highly anisotropic scattering is present, but is quite effective in the isotropic scattering case. We believe P1C to be superior than P1M for DSA.

Therefore, MIP is a reasonable choice when anisotropy is not too strong. The fully consistent P1C may be promising and we recommend further studies of it.

CHAPTER IV

SPATIAL ADAPTIVE MESH REFINEMENT

A. Introduction

In this chapter, we describe a cell-based spatial Adaptive Mesh Refinement (AMR) technique for the neutron transport equation. The numerical simulation of multi-dimensional deterministic particle transport processes remains a challenging issue in applied mathematics and engineering due to the high dimensionality of the phase-space. For large multi-dimensional realistic problems, containing heterogeneous materials of greatly varying opacities in complex geometrical configurations, an approach based on a uniformly distributed fine mesh can be too costly, in both memory and CPU time, to provide a reasonably accurate numerical solution. The concept of automatic mesh adaptation as a technique to efficiently obtain an accurate numerical solution to a partial differential equation (PDE) with fewer unknowns has been pioneered since the 1980's for finite volume and finite element techniques [121, 1]. We have explained and reviewed AMR in the first chapter. We will take a look at AMR from the implementation viewpoint and give more detailed reviews in this section.

The rationale for mesh adaptivity is based on the notion that a locally refined or adapted mesh, whose number of unknowns is significantly smaller than that of a uniform mesh, can yield the the same level of accuracy. This is based on the fact that solutions of PDEs

- can, on the one hand, present boundary layers, steep gradients, and discontinuities in some regions, where small mesh cells are required to resolve these aspects and provide a good numerical approximation,

- and, on the other hand, often exhibit a smoother behavior in other regions of the domain, where larger mesh cells can be employed while still yielding an accurate numerical approximation.

In order to prescribe the next adapted mesh on which a newer numerical solution is to be computed, mesh adaptivity procedures typically require the following tools:

1. a reliable local error estimate to assess the amount of error committed in a given cell; this estimate is usually obtained from the current numerical solution,

2. flexible geometrical data structures to follow the physics tightly and to allow the efficient implementation, and,

3. prolongation/restriction operators in order exchange data in between mesh cells of various refinement levels.

We now elaborate on these three above-mentioned aspects. First, an error estimator or indicator is necessary to determine the regions which will require further refinement. These zones are not known *a priori* and are obviously PDE- and problem-dependent. This naturally calls for error estimators based on a current numerical solution, a technique known and referred to as *a posteriori* error estimation [58]. With *a posteriori* error estimation, the zones with larger errors are selected for refinement. Hence, it is possible to control the numerical error in an automated succession of computations performed on locally adapted meshes. This leads to high-resolution numerical solutions that can be obtained with fewer unknowns and smaller CPU times than the more pedestrian approach based on uniform mesh refinement. Second, a flexible data structure is needed to handle the local refinement of the mesh. More specifically, the data structure needs to support *hp*-type unstructured meshes. In the context of multigroup approximations, this means that (i) energy groups may

have different meshes (hence, we need to handle group-dependent spatial meshes), (ii) elements may have different polynomial order (p-refinement type), and (iii) elements may have different refinement levels and histories. In cell-based discrete ordinate (S_N) transport sweeps, this notably requires the ability to insert additional newly refined cells in the sweep ordering and to efficiently obtain the radiation inflow values in between cells of various refinement levels. Finally, prolongation and restriction operations are needed (i) to compute the in-scattering and fission terms contribution to group g due to reactions that occurred in other groups $g' \neq g$ (mesh coupling) (ii) to compute inflow values in between zones of different refinements (mesh irregularity) and also (iii) to project the current solution onto the next adapted mesh in order to provide an good initial guess for the next computation and to reduce the computational time on the new mesh (e.g., bootstrapping the numerical solution on the newer mesh using a projection of the current solution). We also note that mesh coarsening (though not used in this Dissertation) can also be employed to decrease the resolution in areas where the mesh granularity was deemed too fine; examples of such situations include, for instance, transient problems with front wave propagations.

While mesh adaptivity is now widely used in many science and engineering fields (see, for instance, [122, 123] and recent textbooks such as [124, 125, 126]), only a limited number of references are available regarding the applications of mesh adaptivity to the transport equation. Most methods in place in production codes for computing the solution of the transport equation have been implemented for fixed computational meshes and cannot easily support a local refinement. It can be noted that some of the earliest work on mesh adaptive refinement for transport has occurred in the field of radiative transfer, where a transport solver was frequently coupled to AMR hydrodynamics codes [105, 127, 128], resulting in a natural tendency to implement AMR techniques in the transport solver.

In [51], a patch-AMR technique for the discrete ordinate transport equation has been devised and is based on a hierarchy of nested grids (see also the prior work of Berger and Oliger [129] and Berger and Colella for hyperbolic PDEs [105]). Patch-AMR can be relatively simple to implement in an already existing code that uses a fixed Cartesian grid; additionally, the various patches of a mesh can be readily distributed for parallel computations. Some of the drawbacks of patch-AMR may include the fact that the physics are not followed as closely as possible (the extent of refined patch being often too large), leading to more unknowns than needed, and the need to converge inflow/outflow values in between nested grids (a feature that is not present in cell-based AMR). In [51], the gradient of the solution is employed to drive the adaptive mesh refinement in 2-D Cartesian geometries for a one-group (one-frequency) transport equation. The gradient-based error estimator is known to be fairly accurate for low-order (e.g., first-order step) schemes but is overly conservative for higher-order schemes. A similar multiple-grid patch-AMR technique, with error estimation based on the gradient of the numerical solution, has also been more recently used by several other authors for photon transport applications, see, for instance, [130, 62, 63].

In [64], a local refinement (cell-based AMR) technique is described for S_N transport, where the value of the neutron MFP (Mean Free Path) in a given cell is employed as a mesh refinement criterion. While this approach takes into account the size of potential internal layers at a given location in the domain, it does not account for the *actual* smoothness of the solution at these locations and is, therefore, far from optimal; for example, in optically thick areas, the solution may well be approximated by a smooth spatial representation on coarse meshes despite the smallness of the MFP.

Kanschat et al. used fully adaptive finite element approximations to the stationary, monochromatic radiative transfer problem [131, 132] on 2-D structured Cartesian

grids. Klar et al. [133] considered a coupled radiation-temperature model, with a Simplified P_N treatment of the angular variable, leading to diffusion-like equations, and applied adaptive methods to their model to resolve the boundary layer of a hot, homogeneous body that is in thermal contact with a colder exterior.

In the field of neutronics and reactor applications, some authors have also considered mesh adaptivity for P_N and diffusion approximations. Lewis et al. presented a refinement technique for the inter-element approximation in a primal hybrid finite element technique; diffusion [134] and P_N [54] results were given for a 2-D one-group problem. Ragusa [55, 135] employed an error indicator based on estimating the second derivatives of the numerical solution and applied the resulting method to multigroup diffusion problems; their error indicator was based on the interpolation error of linear finite elements [136] and therefore limiting the application to these elements. More recently, Wang and Ragusa applied the *hp*-adaptation concept to the multigroup diffusion equations, where both the local polynomial degree of shape functions and the local cell size are selected adaptively [56]; nonetheless, error estimators for the combined *hp* adaptation are less mature and, as a consequence, these authors used the difference between a finer mesh solution Φ_{fine} and a coarser mesh solution Φ_{coarse} to drive the refinement; their results included mesh adaptation for 1-D multigroup and 2-D 1-group diffusion problems.

An outline of the remainder of this chapter is as follows: Section B presents the error indicator used to select mesh cells for refinement, describes the mesh refinement process and how multi-mesh coupling and mesh irregularity are dealt with. In Section C, 2-D results are presented for three examples: (i) a simple homogeneous problem employed to present in details the performance of AMR, (ii) a searchlight example, (iii) a problem with different material opacities. We conclude in Section D and present an outlook on open questions.

B. Mesh Adaptation

We briefly review next the principles of AMR, the *a posteriori* error estimates used to drive the AMR procedure for transport and describe some implementation details, namely the hierarchy of refined meshes, the edge flux mapping between elements of various refinement levels and the cell flux mapping between different meshes.

1. Principles of *a posteriori* error adaptivity and AMR

An AMR procedure generally starts with a given initial and coarse mesh. A mesh generator could be employed to generate a reasonable initial triangularization. On this mesh, a numerical solution is sought after using an appropriate solver for the PDE under consideration, here the multigroup S_N transport. With the use of *a posteriori* error estimates, the current numerical solution itself is employed to determine the regions where the spatial discretization errors are large. A fraction of the elements with the largest errors is selected for local refinement. The error estimates are said to be *a posteriori* because they are determined once a numerical solution is obtained. With elements flagged for refinement, a newly refined or adapted mesh is available and a new, more precise, numerical solution can be obtained. This process is repeated, as shown in Fig. IV-1, until a user-prescribed tolerance on the absolute global error is satisfied.

By effectively estimating the error, the entire simulation can be controlled: once a numerical solution has been computed on, say, the k-th mesh, the error is estimated using the current solution. If the solution has not converged sufficiently, the error estimator is used to build a new mesh $k+1$ on which a new solution will be sought. The entire process is achieved within a single calculation, comprising a set of successively adapted meshes and their solutions.

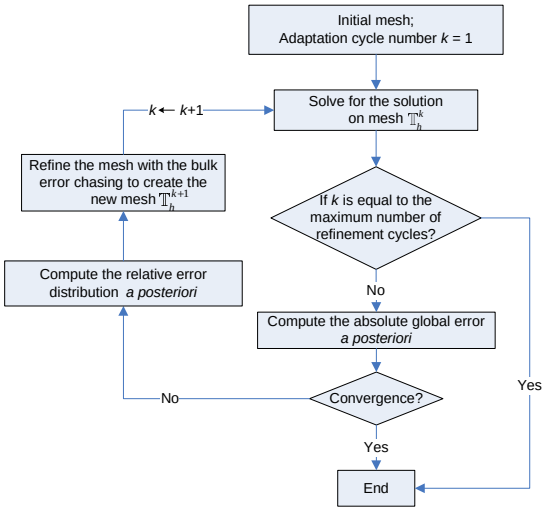


Fig. IV-1. Principle of mesh adaptation.

2. Error Estimations

Dealing with the approximation error, i.e., the difference between the exact solution and a numerical solution, is an arduous task because the bounds of the approximation error are complex to obtain, with the added difficulty that they are problem-dependent. Over the last two decades, the theory of *a posteriori* error estimations [58, 57] has significantly progressed to allow the measure and minimization of approximation errors. In this theory, the computed solution itself is used to inexpensively provide point-wise error estimations. In this section, we will present two error estimations: jump-based error indicator and projection-based error estimator. Their advantages and disadvantages will be discussed.

a. Jump-based Error Indicator

Some error estimators for transport problems have been theoretical derived (see, for instance, [137, 65, 138]), requiring the use of an adjoint calculation, which can

significantly increase the computational cost of the error estimator. Furthermore, this adjoint (dual) solution needs to be sought in a richer space than the primal solution. As a consequence, simpler techniques or considerations are frequently employed in order to obtain an error indicator; as noted in the introduction (Section A), several authors utilize the gradient of the numerical solution to drive the mesh refinement.

Here, we propose an error estimator that is both related to the formal mathematical derivation of the *a posteriori* error and intuitively practical. The fact that DG methods are discontinuous approximations, with the presence of jumps in the numerical solution at the interfaces between elements, can be used to monitor the approximation error. It has been observed that, as the mesh is refined, the magnitude of these jumps tends to zero, since the true solution is better approximated. Therefore, it is intuitive to use the jump values as an indication of the spatial error distribution and our mesh adaptive method will closely monitor these jumps. The *a posteriori* error estimators used in [137, 65, 138] is based on the the interior and interface (edges in 2-D) residuals. The integrated inter-element jumps are closely related to the interface residual and using them to control the error distribution in our simulations is, therefore, closely related to a more formal mathematical justification and can be a sensible (and less costly) alternative to adjoint-based error estimates.

Nonetheless, in the above discussion, the jumps are to be understood in the light of the S_N transport approximation, i.e., as direction-dependent (or angular) jumps. However, in most steady-state S_N algorithms, the angular flux is not kept (except on the domain's boundaries and the interfaces of its partitions) but is discarded after a transport sweep has been performed for a given direction. Thus, the information retained is usually limited to the flux moments, which are angle-integrated quantities, as shown in Eq. (C.35), because the number of moments is usually significantly smaller than the number of directions. This has led us to modify the error estimate in order

to only employ angle-integrated quantities and we finally arrive at the definition of the practical error indicator used in this dissertation:

$$\eta_{g,K}^{k} = \frac{1}{\left\|\Phi_g^k\right\|_2^2}\int_{\partial K}[\![\Phi_g^k]\!]^2\,ds = \frac{1}{\left\|\Phi_g^k\right\|_2^2}\int_{\partial K}\left(\sum_{m=1}^{M} w_m[\![\Psi_{g,m}^k]\!]\right)^2 ds \qquad (4.1)$$
$$\text{for } K \in \mathbb{T}_{g,h}^k, \quad g = 1, \cdots, G$$

where

$$[\![\Phi_g^k(\vec{r})]\!] = \Phi_g^{k,+}(\vec{r}) - \Phi_g^{k,-}(\vec{r}) \qquad (4.2)$$

and

$$\Phi_g^{k,-}(\vec{r}) = \lim_{s\to 0^-}\Phi_g^k(\vec{r}+s\vec{\mathbf{n}}) \qquad (4.3)$$

$$\Phi_g^{k,+}(\vec{r}) = \begin{cases} \lim_{s\to 0^+}\Phi_g^k(\vec{r}+s\vec{\mathbf{n}}) & \text{when } \vec{r}\notin\partial\mathcal{D} \\ \int_{\vec{\Omega}\cdot\vec{n}>0}\Psi_g^{k,-}(\vec{r},\vec{\Omega})\,d\Omega + \int_{\vec{\Omega}\cdot\vec{n}<0}\Psi_g^{inc}(\vec{r},\vec{\Omega})\,d\Omega & \\ = \sum_{\vec{\Omega}_m\cdot\vec{n}>0} w_m\Psi_{g,m}^{k,-}(\vec{r}) + \sum_{\vec{\Omega}_m\cdot\vec{n}<0} w_m\Psi_{g,m}^{inc}(\vec{r}) & \text{when } \vec{r}\in\partial\mathcal{D}^d \\ \Phi_g^{k,-}(\vec{r}) & \text{when } \vec{r}\in\partial\mathcal{D}^r \end{cases} \qquad (4.4)$$

$\vec{\mathbf{n}}$ is the unit norm outward vector wrt the element K on the element boundary ∂K; $\partial\mathcal{D}^d$ is the Dirichlet boundary; $\partial\mathcal{D}^r$ is the reflecting boundary. $\Psi_g^{k,-}(\vec{r},\vec{\Omega})$ is the angular flux defined on the element boundary. $\Psi_g^k(\vec{r})$ is the scalar flux. $\mathbb{T}_{g,h}^k$ denotes the mesh for the g-th group, at mesh adaptivity cycle k; note the use of the subscript g for each mesh triangularization to stress that different energy groups may have different meshes.

This error indicator for a given element $K \in \mathbb{T}_{g,h}^k$, is the integrated jump of the scalar flux $\Phi_g^k(\vec{x})$ along all interfaces ∂K. The error indicator is weighted by the norm of the scalar flux so that all energy groups can converge at the same rate. This indicator, denoted hereafter as *jump-based*, is simple and inexpensive to compute and

will be used to drive the mesh refinement.

b. Projection-based Error Estimator

This so-called *projection-based* error estimator was proposed in [125, 126] for elliptic and Maxwell's equations and was later extended in the context of the multigroup neutron diffusion equations by Wang and Ragusa [56].

Instead of computing the flux moments on the adapted mesh $\mathbb{T}^k_{g,h}$ at each mesh adaptation cycle k, they are solved for on a refined adapted mesh $\mathbb{T}^k_{g,h/2}$. $\mathbb{T}^k_{g,h/2}$ is obtained by refining *each* element K of the $\mathbb{T}^k_{g,h}$ mesh (in 2-D triangular geometries, this translates into subdividing each triangle K into 4 smaller triangles). We denote the solution on the refined mesh $\mathbb{T}^k_{g,h/2}$ as $\Phi^k_{g,h/2}$. This finer solution lives in a much richer function space than the solution of the $\mathbb{T}^k_{g,h}$ mesh, so it is significantly closer to the exact solution. We then project the finer solution back into the function space of the $\mathbb{T}^k_{g,h}$ mesh. The projection $\Pi_h \Phi^k_{g,h/2}$ is an fairly close approximation of the numerical solution on the coarse mesh $\mathbb{T}^k_{g,h}$. The L_2 norm of the difference between the solution computed on the finer mesh $\mathbb{T}^k_{g,h/2}$ and its projection onto $\mathbb{T}^k_{g,h}$ provides a good representation of the spatial error on $\mathbb{T}^k_{g,h}$. Following the practice employed in elliptic problems, where the semi-H_1 norm, i.e., the L_2 norm of the currents is used to compute the error estimate [139, 140, 141], we have chosen to compute the L_2 norm of the difference of the first (angular) flux moments (i.e., the x and y components of the net current) on the fine mesh and its projection on the coarse mesh. This difference is calculated on all elements of the coarse mesh, at adaptivity cycle k, as follows. For simplicity, we have dropped the mesh iteration superscript k in the following

equations.

$$\mu_{g,K}^{k} = \frac{\int_K \left[(\Pi_h J_{g,h/2}^{x} - J_{g,h/2}^{x})^2 + (\Pi_h J_{g,h/2}^{y} - J_{g,h/2}^{y})^2\right] \, d\vec{r}}{\int_{\mathcal{D}} \left[(\Pi_h J_{g,h/2}^{x})^2 + (\Pi_h J_{g,h/2}^{y})^2\right] \, d\vec{r}}, \qquad (4.5)$$
$$K \in \mathbb{T}_{g,h}^{k}; \quad g = 1, \cdots, G$$

In this equation and afterwards, the projection operation is defined as:
Find $\Pi_h \Phi \in W_{\mathcal{D}}^{h}$, such that,

$$\left(\Pi_h \Phi, \Phi^*\right)_{\mathcal{D}} = \left(\Phi, \Phi^*\right)_{\mathcal{D}}, \qquad \forall \Phi^* \in W_{\mathcal{D}}^{h} \qquad (4.6)$$

The solution vector of $\Pi_h \Phi$ is computed with a matrix-vector product of the inverse of the global mass matrix and the right-hand-side vector obtained with Φ, which could be in a richer function space than the DGFEM space $W_{\mathcal{D}}^{h}$. Because the continuity on the element interfaces of solutions belonging to space $W_{\mathcal{D}}^{h}$ is not required, the global mass matrix corresponding to $W_{\mathcal{D}}^{h}$ is block diagonal, where a block corresponds to an element. (It could even be strictly diagonal if we define shape functions which are orthogonal to each other on the reference element.) So the inverse of this matrix representing the projection operation can simply be done cell by cell; its cost is negligible compared to the cost of the solver. In contrast, in the continuous FEM setting, in order to avoid inverting the global mass matrix, which is not block diagonal in that case, the projection operation keeps the solution values on all element interfaces to preserve continuity and solutions are projected locally onto the span of the bubble interior functions only (since values of bubble functions on the element boundary are equal to zero).

In a preliminary version of the code developed, we tested the option of either choosing the first flux moments, i.e., the net currents, or the zeroth flux moment, i.e., the scalar flux, for calculating the error and results suggested that the first

flux moments were a better choice. This error estimator, valid on $\mathbb{T}^k_{g,h}$, requires the calculation of the numerical solution on the finer mesh $\mathbb{T}^k_{g,h/2}$ at every mesh adaptation cycle k, which may seem sub-optimal because (i) there are more unknowns in the finer mesh and (ii) only the error on the coarse mesh is obtained. We will discuss this point in Section C. The error distribution calculated is for the solution on the coarse mesh and not the fine mesh on which we sought the solution. Although a fixed ratio between these two errors may exist in the asymptotic range, this ratio could be problem-dependent and may not be easily obtained *a posteriori* . Nonetheless, upon convergence, the final product of this mesh adaptivity strategy *is* the numerical solution on the finer adapted mesh, which, as we have already mentioned, is much closer to the exact solution than the numerical solution on the coarser adapted mesh. Furthermore, this error estimate is, by construction, asymptotically exact. Accuracy and simplicity are among its more obvious advantages and it can also be used for *hp*-type AMR with minor modifications as presented in [125, 126].

c. A Two-Mesh Error Estimator

For debug purposes, when projection-based error estimator is used, the solution on the coarse mesh can also be computed at each adaptivity cycle (rather than approximation it by the projection of the finer mesh solution.) Then, an error estimation can be evaluated straightforwardly using the difference between the fine solution and the coarse solution directly.

$$\mu^{k,ref}_{g,K} = \frac{\int_K \left[(J^x_{g,h} - J^x_{g,h/2})^2 + (J^y_{g,h} - J^y_{g,h/2})^2 \right] \, d\vec{r}}{\int_{\mathcal{D}} \left[(J^x_{g,h})^2 + (J^y_{g,h})^2 \right] \, d\vec{r}} \tag{4.7}$$

In the asymptotic range (i.e., fine mesh limit), $\mu^{k,ref}_{g,K}$ and $\mu^k_{g,K}$ are very similar.

d. Reference Numerical Solution as Estimator

Finally, for debug purposes as well, an numerical "exact" solution to the S_N transport equations can be obtained and stored for verification of the various error estimates. We denote this solution as the *reference* solution and typically obtain it using the two-mesh error estimator $\mu_{g,K}^{k,ref}$.

Once this *reference* solution $\vec{J}_g^{ref}$ has been computed, we can drive the AMR procedure using the following "exact" spatial error distribution $\epsilon_{g,K}$.

$$\epsilon_{g,K}^k = \frac{\int_K \left[(J_{g,h}^x - J_g^{x,ref})^2 + (J_{g,h}^y - J_g^{y,ref})^2\right] \, d\vec{r}}{\int_{\mathcal{D}} \left[(J_{g,h}^x)^2 + (J_{g,h}^y)^2\right] \, d\vec{r}} \tag{4.8}$$

e. Closing Comments on the Error Estimates

Note the AMR processes based either on $\eta_{g,K}$, $\mu_{g,K}$, $\mu_{g,K}^{ref}$, or $\epsilon_{g,K}$ will yield different meshes and convergence histories.

In the results Section, we compare the error Eq. (4.8) as a function of the number of unknowns for the following three refinement strategies: (1) uniform mesh refinement, (2) adaptivity using the jump-based indicators of Eq. (4.2), and (3) adaptivity using the projection-based error of Eq. (4.8). The meshes resulting from the two AMR strategies will also be compared.

3. Refinement Strategy and Stop Criteria

The criterion for refinement is defined as follows: an element K of $\mathbb{T}_h^k$ is selected for refinement if

$$\eta_{g,K}^k \geq \alpha \max_{K' \in \mathbb{T}_{g,h}^k, 1 \leq g' \leq G} \left(\eta_{g',K'}^k\right), \tag{4.9}$$

where α is a user-defined fraction (we used $\alpha = 0.3$, unless otherwise noted). This criterion allows us to focus the computational effort on elements with the largest errors and tends to equi-distribute the spatial error. Note that this criterion does not mean that 30% of the elements are refined at each iteration; only the elements whose error is greater than or equal to 30% of the largest error are refined. XUTHUS was written to allwo *hp*-refinement but only *h*-adaptivity has been fully implemented so far, so there is need to specify an additional criterion to select beween *h*- (subdividing) or *p*- (increasing the polynomial order of) refinement at this point.

We can apply the above strategy with any other error estimates: $\mu^k_{g,K}$, $\mu^{k,ref}_{g,K}$, or $\epsilon^{k,ref}_{g,K}$. It can easily be seen that this strategy leads to group-dependent meshes in a natural way. It is possible to specify (user-input) that several energy groups share the same mesh, i.e., the number of meshes is not necessarily equal to the number of energy groups. In this case, the above refinement criterion should be understood as follows: if a cell is marked for refinement in any energy group sharing the same mesh, then it is going to be refined.

We obtain the global error index by summing the error estimates over all elements,

$$\eta^k_g = \sqrt{\sum_{K \in \mathbb{T}^k_{g,h}} \eta^k_{g,K}}, \qquad g = 1, \cdots, G \tag{4.10}$$

though this is only a reliable measure of the global error when the projection-based ($\mu^k_{g,K}$), two-mesh ($\mu^{k,ref}_{g,K}$), or reference error ($\epsilon^{k,ref}_{g,K}$) estimates are utilized.

The AMR iterations are stopped when

$$\eta^k_g < \mathrm{tol}_{AMR}, \qquad g = 1, \cdots, G \tag{4.11}$$

or when the total number of adaptivity cycles has been reached. In cases where the globally-refined solution is available at each cycle of the mesh adaptation, we can evaluate the relative errors of the scalar flux at little extra cost

$$\mu_g^{k,ref} = \frac{\left\|\Phi_{g,h} - \Phi_{g,h/2}\right\|_2}{\left\|\Phi_{g,h}\right\|_2} \tag{4.12}$$

and use this to control the termination of AMR.

4. *hp*-type Hierarchy of 2-D Unstructured Meshes

The Discontinuous Galerkin (DG) FEM supports *hp* unstructured meshes for AMR applications in a natural manner. By construction of the approximation space in DGFEM, the continuity of the numerical solution is *not* required across element boundaries. Therefore, an arbitrary difference in refinement level can be employed easily in between neighboring elements (for instance, see the Figures of Sections B and C). We refer this as mesh *multi-irregularity* later. Performing the upwind procedure to solve the transport equation using DGFEM on an AMR mesh is straightforward, even for higher-order finite element approximations. On *hp*-type meshes there are no constraints on the difference of polynomial orders between two neighbor elements as well. We do not need to constrain the edge and vertex shape functions to ensure the continuity in DGFEM. This concept of mesh irregularity applies independently to an energy group: depending on the local smoothness of the solution, the number of elements varies locally.

In addition, meshes with different numbers of unknowns (i.e., different mesh irregularities) are also desired in the context of multigroup calculations because the behavior of particles can vary greatly with their energy. For example, the total cross section of fast neutrons is smaller than the one of thermal neutrons; furthermore, fast groups are usually less diffusive than thermal group, and streaming is more

important in the fast portion of the energy spectrum. In short, the smoothness of the multigroup solution is rightfully expected to be quite different in between the multigroup components. This leads to another AMR concept: the *multi-mesh* concept, or group-dependent AMR meshes. Because the solution on any element is locally determined with DGFEM, it is easier to deal with the mesh coupling in between energy groups or to project the solution from one mesh onto another.

In our implementation in XUTHUS, the user can set

1. the maximum difference in refinement levels (mesh irregularity),
2. the maximum polynomial order to be used, and
3. the maximum level difference in between two elements located in different meshes that have a common parent element (or equivalently have the same initial element).

In addition, several energy groups can be tagged by the user so that they share the same adapted mesh (a useful feature to limit the total number of group-dependent meshes when the number of energy group is large).

It is important to note that all meshes are derived from the same initial (and usually) coarse mesh.

We denote by $\mathbb{T}^k_{g,h}$ a subdivision of domain $\mathcal{D}$ at the k-th mesh adaptivity cycle for one energy group g of all G energy groups. The number of elements on the mesh $\mathbb{T}^k_{g,h}$ is denoted by $N^k_{el,g}$. The *initial mesh*, $\mathbb{T}^0_{g,h}$, is a conforming triangular mesh, either obtained from a structured 2-D Cartesian mesh whose rectangles have been split into 2 triangles, or obtained from a 2-D Delaunay mesh generator (our initial unstructured meshes are generated with Triangle [93]). All energy groups are sharing the *same* initial mesh $\mathbb{T}^0_h$. All elements in the initial mesh are numbered with a fixed

ordering. This ordering will help to create the natural ordering, which will be used for mesh coupling and explained later. Any given mesh $\mathbb{T}^k_{g,h}$ $(k \geq 0; 1 \leq g \leq G)$ consists of disjoint open element cells K such that their union fully covers $\mathcal{D}$, i.e., $\bigcup_{K \in \mathbb{T}^k_{g,h}} K = \mathcal{D}$.

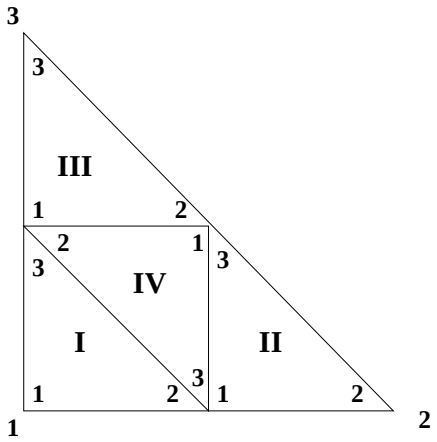


Fig. IV-2. Element refinement rules.

Once an element K has been flagged for refinement, it is subdivided into 4 equal smaller triangles (as seen on Fig. IV-2). All four child elements have the same shape. A subdivision into 4 smaller elements avoids the creation of sliver elements (which are frequently obtained when triangles are successively cut into 2 or 3 smaller triangles; we note that sliver elements could be mitigated using techniques such as edge swapping but we have not implemented such techniques and, therefore, only refine a given element into 4 children). The rules for element refinement are as follows:

1. Child element with rank I is placed near vertex 1.

2. Child element with rank II is placed near vertex 2.

3. Child element with rank III is placed near vertex 3.

4. Child element with rank IV is placed at the center.

5. The three vertices of child elements 1 to 3 common with the original element inherit the same local numbering.

6. The three vertices of center child element are numbered using the opposite vertex numbers of the original element.

7. Vertex and edge numbering of four child elements follow the rules presented in chapter 2.

These rules are illustrated in Fig. IV-2. These numbering rules ensure that the local vertex numbering is always counter-clockwise.

The children elements and the parent element remain related and the refinement process leads to a hierarchy of mesh cells that have all been obtained from subdivisions of cells from the initial mesh $\mathbb{T}_h^0$. All elements form a tree hierarchy structure. We denote as *active* an element that is not refined any further, i.e., it is a cell on which basis functions are defined. A cell becomes inactive after it has been refined. We number all active elements with the so-called natural ordering, explained below. This ordering (or sets of rules) allows us to reconstruct on the fly the tree structure (instead of storing it) with only the knowledge of the refinement level of all active elements.

1. First, the ordering is based on the ordering of initial elements; the number of an active element is smaller if its initial element has a smaller number.

2. Second, elements sharing the same closest ancestor element are ordered based on the rank of their ancestors which are the direct children of the closest common ancestor. Smaller rank has smaller numbering.

Natural ordering is illustrated in Fig. IV-3. This natural ordering proved to be useful for the mesh coupling because once it is set up, only refinement levels of all elements are needed to describe the partition of the domain.

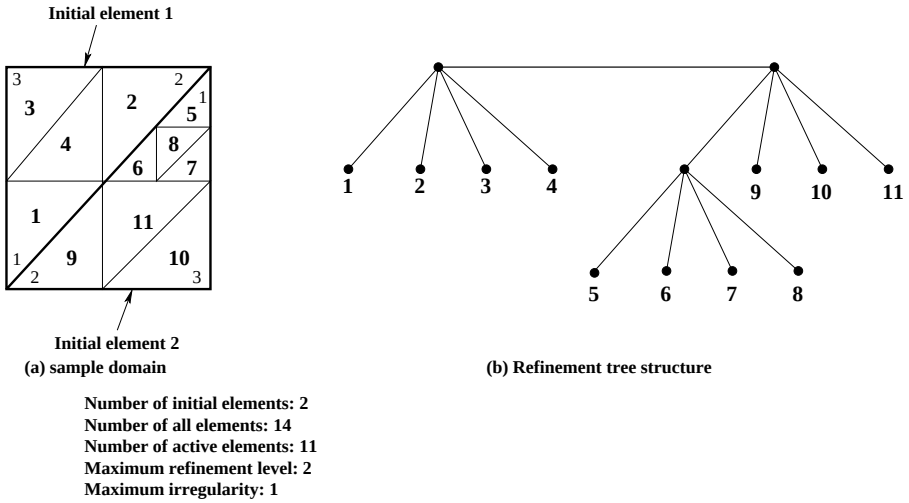


Fig. IV-3. Refinement tree.

We define the *refinement level* $\ell(K)$ for a given element K as the number of times a cell in the original mesh $\mathbb{T}_h^0$ has been refined to yield element K. Note that the the refinement level $\ell(K)$ is not the number of refinement cycles k since mesh cells are not necessarily refined at each adaptivity cycle. Practically, each time a cell is refined, its children inherit its refinement level, augmented by 1, (by convention, mesh cells in the initial mesh all have a refinement level of 0). Note that two neighboring elements, K_1 and K_2, that have the same refinement level ($\ell(K_1) = \ell(K_2)$) share a common edge in its entirety. This notion will prove useful to determine the upwind contribution, detailed next in Section 5. Once an element has been refined, it is removed from the sweep ordering and replaced by its 4 children, in the appropriate order for all directions. The sweep ordering algorithm will be detailed in Chapter V.

5. Edge Interface Flux Mapping

In a sweep-based solution procedure for transport or diffusion calculations (e.g., SSOR for diffusion), knowledge of the incoming angular fluxes from upwind neighboring elements or the scalar fluxes of the all neighboring elements are required to solve the local system in a given element.

We have three basic situations to deal with in the case of AMR, which are illustrated in Fig. IV-4: case (1) both elements sharing an edge have the same refinement level (this situation is identical to the case without refinement but will be explained in details as this will be helpful to understand cases 2 & 3); case (2) the edge contribution to element K comes from smaller elements; case (3) the edge contribution to element K comes from larger elements. There are two equivalent ways to compute the edge contribution: one manner employs 2-D prolongation and edge coupling matrices, the other manner uses only 1-D prolongation and edge operation matrices. Both ways are explained below for completeness. In transport calculations, the edge coupling only requires operations dealing with the basis functions that are nonzero on a given edge. This forms a subset of the element basis functions and can be dealt with as a 1-D problem only since the coupling amounts to computing a 1-D mass matrix (the basis functions not associated with that edge are exactly zero and do not contribute to the coupling). But, in the case of a diffusion solver, edge coupling also involve terms containing the derivative of the basis functions (and then any basis function may be nonzero on any given edge), requiring that all basis functions be taken into account in the implementation. Both coupling ways are presented here, although the second way employing all basis functions is recommended for a matrix-free algorithm due to its ease of implementation.

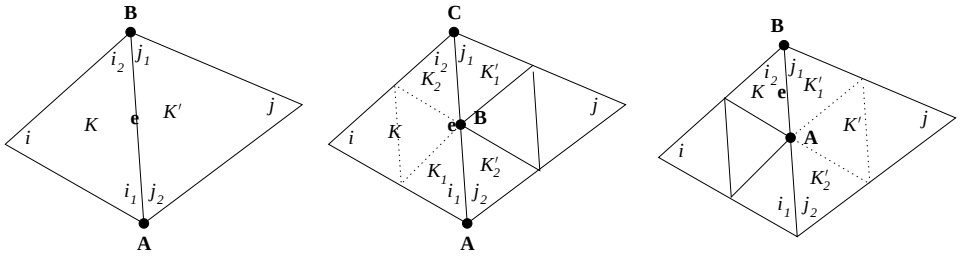


Fig. IV-4. Three cases of edge coupling.

a. With Cell Prolongation and Edge Coupling Matrices

First, we consider a given edge e separating element K and its neighbor K', with the assumption that these two elements possess the same refinement level, i.e., $\ell(K) = \ell(K')$ as shown in case 1 of Fig. IV-4. In such a situation, edge e is shared in its entirety by K and K' ($e = K \bigcap K'$). Let us consider the simplest edge coupling term (i.e., only functions and not their derivatives participate in the coupling terms), which is given by

$$\int_e u_{K'}(x,y) u_K^*(x,y)\, ds \tag{4.13}$$

where u_K^* is an arbitrary test function in element K and $u_{K'}$ is the solution on the neighboring element K'. We use the notation u to designate the solution variable, which, in transport calculations, is the angular flux and, in diffusion calculations, is the scalar flux.

Expanding the solution and test function with the shape functions and applying the change of variables to map onto the reference element, we have

$$u_K^*(x,y)\big|_e = \mathbf{u}_K^{*T}\mathbf{b}_K(x,y)\big|_e = \mathbf{u}_K^{*T}\widehat{\mathbf{b}}_K(\xi_1,\xi_2)\Big|_e = \mathbf{u}_K^{*T}\widehat{\mathbf{b}}(\xi_i) \tag{4.14}$$

$$u_{K'}(x,y)\big|_e = \mathbf{b}_{K'}^T(x,y)\mathbf{u}_{K'}\big|_e = \widehat{\mathbf{b}}^T(\xi_1,\xi_2)\mathbf{u}_{K'}\Big|_e = \widehat{\mathbf{b}}^T(-\xi_j)\mathbf{u}_{K'} \tag{4.15}$$

The definitions for $\xi_k,\ k = 1, 2, 3$ can be found in Chapter II. Here, i is the local edge ID of edge e with respect to element K and j is the local edge ID of e for element K'. Substituting the ξ_k into Eq. (4.13), we obtain

$$\begin{aligned} \int_e u_{K'}(x,y)u_K^*(x,y)\,ds &= \frac{L_{AB}}{6}\mathbf{u}_K^{*T}\left[3\int_{-1}^{1}\widehat{\mathbf{b}}(\xi_i)\widehat{\mathbf{b}}^T(-\xi_j)ds\right]\mathbf{u}_{K'} \\ &= \frac{L_{AB}}{6}\mathbf{u}_K^{*T}\mathrm{E}_{i,j}^{1C}\mathbf{u}_{K'}, \end{aligned} \tag{4.16}$$

where the type-1 edge coupling matrix $\mathrm{E}_{i,j}^{1C}$ was defined in Chapter II and L_{AB} is the length of edge $[AB]$. When assembling the global system, vectors $\mathbf{u}_{K'}$ and $\mathbf{u}_K^{*T}$ simply indicate which columns and rows of the global system should be used to insert the local matrix $\frac{L_{AB}}{6}\mathrm{E}_{i,j}^{1C}$. When assembling the right-hand-side vector, we have $\frac{L_{AB}}{6}\mathrm{E}_{i,j}^{1C}\mathbf{u}_{K'}$. $\mathbf{u}_K^{*T}$ simply tells us where the resulting local vector should be added on the global vector.

We now analyze the situation where the neighboring elements of K have been further refined. For simplicity, let us first consider case 2 of Fig. IV-4, where element K has an neighboring contribution along edge $[AC]$ from two smaller elements, K_1' and K_2'; note that there is only a refinement level difference of 1 between K and its two neighbors, i.e., $\ell(K_1') = \ell(K_2') = \ell(K) + 1$. The contribution to K, through edge $[AC]$, of elements K_1 and K_2 is given by:

$$\int_{[AB]} u_{K_2'}(x,y)u_K^*(x,y)\,ds + \int_{[BC]} u_{K_1'}(x,y)u_K^*(x,y)\,ds \tag{4.17}$$

Let us consider edge $[AB]$ first. Expanding the solution with the shape functions and applying the change of variables as in case 1, we have

$$u_{K_2'}(x,y)\big|_{AB} = \widehat{\mathbf{b}}^T(-\xi_j)\mathbf{u}_{K_2'} \tag{4.18}$$

Note that the (determinant of the) Jacobian of the coordinates transformation is $\frac{L_{AB}}{2}$ here. Again, we expand the test function with the shape functions and, before applying the change of variables, we virtually cut the element K and prolong the solution on element K to its children elements on the edge common with K_1. Because of the rule of refinement, this prolongation operation is only determined by the rank of the child element i_1. (Ranks of three corner child elements are equal to the corresponding vertex number of the parent element based on our refinement rules.)

$$\begin{aligned} u_K^*(x,y)|_{AB} &= \mathbf{u}_K^{*T}\mathbf{b}_K(x,y)\big|_{AB} = (\mathrm{P}_{i_1}\mathbf{u}_K^*)^T\mathbf{b}_{K_1}(x,y)\big|_{AB} \\ &= (\mathrm{P}_{i_1}\mathbf{u}_K^*)^T\widehat{\mathbf{b}}(\xi_i) \end{aligned} \tag{4.19}$$

Note that with our refinement rules, we have

$$\begin{aligned} i_1 &= \mod(i,3)+1 \\ i_2 &= \mod(i_1,3)+1 \end{aligned} \tag{4.20}$$

The $\mathrm{P}_k,\ k=1,2,3,4$ matrices we have introduced are cell prolongation matrices. They operate on the local solution vector and give the solution vectors on the four child elements. With our definition of shape functions and the refinement rules, they are independent of the element shape. These prolongation matrices for the four child elements are given below, for polynomial orders up to 4 (we have outlined the various parts according to the polynomial order):

$$P_1 = \left[\begin{array}{ccc|ccc|cccc|ccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & 0 & -\frac{\sqrt{6}}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{14}}{16} & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & -\frac{\sqrt{6}}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{14}}{16} & 0 & 0 \\
\hline
0 & 0 & 0 & \frac{1}{4} & 0 & 0 & -\frac{\sqrt{15}}{8} & 0 & 0 & 0 & \frac{\sqrt{21}}{16} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & 0 & -\frac{\sqrt{15}}{24} & 0 & \frac{\sqrt{15}}{24} & -\frac{\sqrt{6}}{8} & \frac{5\sqrt{21}}{384} & -\frac{\sqrt{21}}{64} & \frac{5\sqrt{21}}{384} & -\frac{\sqrt{10}}{32} & \frac{\sqrt{10}}{32} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & \frac{\sqrt{15}}{8} & 0 & 0 & 0 & \frac{\sqrt{21}}{16} & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 & 0 & -\frac{\sqrt{35}}{16} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 & \frac{\sqrt{35}}{128} & 0 & -\frac{\sqrt{35}}{128} & \frac{\sqrt{6}}{32} & \frac{\sqrt{6}}{32} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 & 0 & \frac{\sqrt{35}}{16} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & -\frac{5\sqrt{14}}{256} & 0 & -\frac{5\sqrt{14}}{256} & \frac{5\sqrt{15}}{96} & -\frac{5\sqrt{15}}{96} \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16}
\end{array}\right]$$

$$P_2 = \left[\begin{array}{ccc|ccc|cccc|ccccc}
\frac{1}{2} & \frac{1}{2} & 0 & -\frac{\sqrt{6}}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{14}}{16} & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & \frac{1}{2} & 0 & -\frac{\sqrt{6}}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{14}}{16} & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & \frac{1}{4} & 0 & 0 & \frac{\sqrt{15}}{8} & 0 & 0 & 0 & \frac{\sqrt{21}}{16} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & -\frac{\sqrt{15}}{8} & 0 & 0 & 0 & \frac{\sqrt{21}}{16} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{\sqrt{15}}{24} & -\frac{\sqrt{15}}{24} & 0 & -\frac{\sqrt{6}}{8} & \frac{5\sqrt{21}}{384} & \frac{5\sqrt{21}}{384} & -\frac{\sqrt{21}}{64} & 0 & -\frac{\sqrt{10}}{16} \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 & 0 & \frac{\sqrt{35}}{16} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 & 0 & -\frac{\sqrt{35}}{16} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & -\frac{\sqrt{35}}{128} & \frac{\sqrt{35}}{128} & 0 & -\frac{\sqrt{6}}{16} & \frac{\sqrt{6}}{32} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & -\frac{5\sqrt{14}}{256} & -\frac{5\sqrt{14}}{256} & 0 & 0 & \frac{5\sqrt{15}}{96} \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16}
\end{array}\right]$$

$$
\mathrm{P}_3 = \left[\begin{array}{ccc|ccc|cccc|cccc}
\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & -\frac{\sqrt{6}}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{14}}{16} & 0 & 0 \\
0 & \frac{1}{2} & \frac{1}{2} & 0 & -\frac{\sqrt{6}}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{14}}{16} & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{\sqrt{15}}{24} & -\frac{\sqrt{15}}{24} & -\frac{\sqrt{6}}{8} & -\frac{\sqrt{21}}{64} & \frac{5\sqrt{21}}{384} & \frac{5\sqrt{21}}{384} & \frac{\sqrt{10}}{32} & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & \frac{\sqrt{15}}{8} & 0 & 0 & 0 & \frac{\sqrt{21}}{16} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & -\frac{\sqrt{15}}{8} & 0 & 0 & 0 & \frac{\sqrt{21}}{16} & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 & 0 & 0 & -\frac{\sqrt{35}}{128} & \frac{\sqrt{35}}{128} & \frac{\sqrt{6}}{32} & -\frac{\sqrt{6}}{16} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 & 0 & \frac{\sqrt{35}}{16} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 & 0 & -\frac{\sqrt{35}}{16} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & -\frac{5\sqrt{14}}{256} & -\frac{5\sqrt{14}}{256} & -\frac{5\sqrt{15}}{96} & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16}
\end{array}\right]
$$

$$
\mathrm{P}_4 = \left[\begin{array}{ccc|ccc|cccc|cccc}
0 & \frac{1}{2} & \frac{1}{2} & 0 & -\frac{\sqrt{6}}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{14}}{16} & 0 & 0 & 0 \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & -\frac{\sqrt{6}}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{14}}{16} & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & 0 & -\frac{\sqrt{6}}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{14}}{16} & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{\sqrt{15}}{24} & -\frac{\sqrt{15}}{24} & -\frac{\sqrt{6}}{8} & -\frac{\sqrt{21}}{64} & \frac{5\sqrt{21}}{384} & \frac{5\sqrt{21}}{384} & \frac{\sqrt{10}}{32} & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & 0 & \frac{\sqrt{15}}{24} & 0 & -\frac{\sqrt{15}}{24} & -\frac{\sqrt{6}}{8} & \frac{5\sqrt{21}}{384} & -\frac{\sqrt{21}}{64} & \frac{5\sqrt{21}}{384} & -\frac{\sqrt{10}}{32} & \frac{\sqrt{10}}{32} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{\sqrt{15}}{24} & -\frac{\sqrt{15}}{24} & 0 & -\frac{\sqrt{6}}{8} & \frac{5\sqrt{21}}{384} & \frac{5\sqrt{21}}{384} & -\frac{\sqrt{21}}{64} & 0 & -\frac{\sqrt{10}}{32} \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & 0 & 0 & 0 & 0 & \frac{\sqrt{35}}{128} & -\frac{\sqrt{35}}{128} & -\frac{\sqrt{6}}{32} & \frac{\sqrt{6}}{16} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & 0 & 0 & -\frac{\sqrt{35}}{128} & 0 & \frac{\sqrt{35}}{128} & -\frac{\sqrt{6}}{32} & -\frac{\sqrt{6}}{32} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & 0 & \frac{\sqrt{35}}{128} & -\frac{\sqrt{35}}{128} & 0 & \frac{\sqrt{6}}{16} & -\frac{\sqrt{6}}{32} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & \frac{5\sqrt{14}}{128} & \frac{5\sqrt{14}}{128} & \frac{5\sqrt{14}}{128} & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16}
\end{array}\right]
$$

Note that P_4 is never used in the edge mapping, since it related to the interior child element but it will be employed for element coupling, for instance in the multi-group case.

Substitute Eqs. (4.18) and (4.19) into the integral,

$$\begin{aligned}\int_{[AB]} u_{K_2'}(x,y)u_K^*(x,y)\,ds &= \frac{L_{AB}}{6}(\mathrm{P}_{i_1}\mathbf{u}_K^*)^T\left[3\int_{-1}^{1}\widehat{\mathbf{b}}(\xi_i)\widehat{\mathbf{b}}^T(-\xi_j)ds\right]\mathbf{u}_{K_2'} \\ &= \mathbf{u}_K^{*T}(\mathrm{P}_{i_1}^T\frac{L_{AB}}{6}\mathrm{E}_{i,j}^{1C})\mathbf{u}_{K_2'} \qquad (4.21)\end{aligned}$$

Similarly, we obtain the contribution from edge $[BC]$:

$$\begin{aligned}\int_{[BC]} u_{K_1'}(x,y)u_K^*(x,y)\,ds &= \frac{L_{BC}}{6}(\mathrm{P}_{i_2}\mathbf{u}_K^*)^T\left[3\int_{-1}^{1}\widehat{\mathbf{b}}(\xi_i)\widehat{\mathbf{b}}^T(-\xi_j)ds\right]\mathbf{u}_{K_1'} \\ &= \mathbf{u}_K^{*T}(\mathrm{P}_{i_2}^T\frac{L_{BC}}{6}\mathrm{E}_{i,j}^{1C})\mathbf{u}_{K_1'} \qquad (4.22)\end{aligned}$$

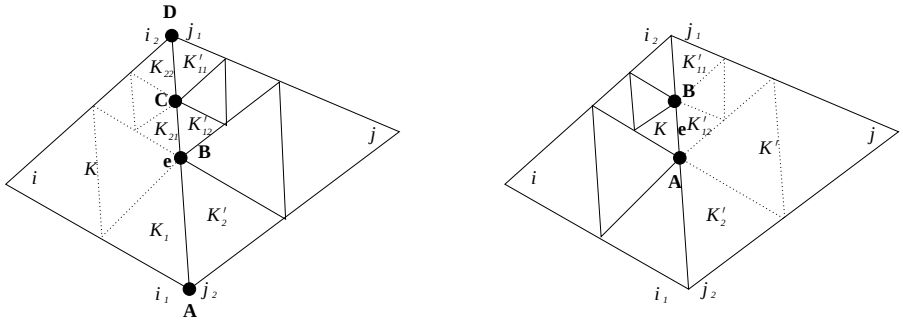
When the refinement levels are strictly greater than 1, as shown on the left pane of Fig. IV-5, we employ the fact that the meshes are nested and recursively use the above procedure, presented for the case of a refinement level difference of 1. For the situation shown on Fig. IV-5, the contribution through edge $[AD]$ to element K from the neighboring elements K_2', K_{12}', K_{11}' is:

$$(\mathrm{P}_{i_1}^T\frac{L_{AB}}{6}\mathrm{E}_{i,j}^{1C})\mathbf{u}_{K_2'} + (\mathrm{P}_{i_2}^T\mathrm{P}_{i_1}^T\frac{L_{BC}}{6}\mathrm{E}_{i,j}^{1C})\mathbf{u}_{K_{12}'} + (\mathrm{P}_{i_2}^T\mathrm{P}_{i_2}^T\frac{L_{CD}}{6}\mathrm{E}_{i,j}^{1C})\mathbf{u}_{K_{11}'} \qquad (4.23)$$

Refinement levels of any degree can be treated this way, using the cell prolongation matrices.

So far, we have described the contribution from smaller elements to larger elements. The case of neighboring contribution from larger elements to smaller elements is derived in an analogous fashion. Consider case 3 of Fig. IV-4. The contribution for element K is simply given by

$$\frac{L_{AB}}{6}\mathrm{E}_{i,j}^{1C}\mathrm{P}_{j_1}\mathbf{u}_{K'} \qquad (4.24)$$



(a) Levels of neighboring elements are higher (b) Level of neighboring element is lower

Fig. IV-5. Multi-irregularity, i.e., differences > 1 in refinement levels.

When the refinement levels are strictly greater than 1, we again employ the fact that the meshes are nested and recursively use the above procedure to determine the neighboring contributions.

Finally, considering the right pane of Fig. IV-5, the contributions from K' to K is,

$$\frac{L_{AB}}{6}\mathrm{E}_{i,j}^{1C}\mathrm{P}_{j_2}\mathrm{P}_{j_1}\mathbf{u}_{K'} \tag{4.25}$$

This procedure applies for all other edge coupling terms and is simply done by changing edge coupling matrix from $\mathrm{E}_{i,j}^{1C}$ to the appropriate matrices $\mathbf{E}_{K,i,j}^{2C}$, $\mathbf{E}_{K,i,j}^{3C}$ or $\mathbf{E}_{K,i,j}^{4C}$. Refer to Chapter III for more details regarding these additional edge coupling matrices.

b. With Edge Prolongation and Edge Operation Matrices

Edge flux mapping can also be perform by considering on the edge under consideration. In this case, the edge contribution reduces to a 1-D problem.

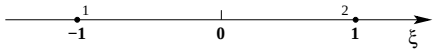


Fig. IV-6. 1-D reference element.

The 1-D reference element is represented by the $[-1;+1]$ interval on Fig. IV-6. Our choice of basis function for the triangle produces the Lobatto polynomials on the triangle's edges (see, e.g., pp. 27-27 and 56-57 in [5] for the definition of Lobatto polynomials).

Let us consider the type-1 edge coupling first. Since the view of edge coupling requires that we extract the the 1-D solution vector from the 2-D element solution

vector, we define the following extraction matrix, for any three local edges:

$$
\begin{aligned}
\mathrm{T}_1 &= \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix} \\
\mathrm{T}_2 &= \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix} \\
\mathrm{T}_3 &= \begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\end{aligned}
\tag{4.26}
$$

Because of the definition of 2-D and 1-D shape functions on their reference elements, the extraction operation simply takes a vector of size $(p+1)(p+2)/2$ (solution vector on an element) and extracts the nonzero components corresponding to a given edge (size of the vector solution on an edge is $p+1$). There are three extraction matrices for the three different edges.

Then, we need to rotate the neighboring edge solution vector to align it with the

orientation of the edge of the local element. The edge rotation matrix is

$$\underline{\mathrm{R}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This rotation operation simply swaps the first two degrees of freedom corresponding to the two edge vertices and adds minus sign on all the other terms with odd polynomial orders. We have used the "underlined" notations for all 1-D items: the rotation matrix, 1-D shape functions, reference mass matrix and 1-D edge prolongation matrices. Note that rotation on 2-D faces is more complicated and would be needed in 3-D calculations.

After we obtain the solution vectors on both sides of an edge, we apply the 1-D mass matrix to provide us with the integral of their inner product along that edge. The reference 1-D mass matrix is defined as

$$\underline{\mathrm{M}} = 3 \int_{-1}^{+1} \underline{\widehat{\mathbf{b}}}(\xi)\, \underline{\widehat{\mathbf{b}}}^{T}(\xi)\, d\xi \tag{4.27}$$

and, up to order 4, is given by

$$\underline{\mathrm{M}} = \begin{bmatrix} 2 & 1 & -\frac{\sqrt{6}}{2} & \frac{1}{\sqrt{10}} & 0 \\ 1 & 2 & -\frac{\sqrt{6}}{2} & -\frac{1}{\sqrt{10}} & 0 \\ -\frac{\sqrt{6}}{2} & -\frac{\sqrt{6}}{2} & \frac{6}{5} & 0 & -\frac{\sqrt{21}}{35} \\ \frac{1}{\sqrt{10}} & -\frac{1}{\sqrt{10}} & 0 & \frac{2}{7} & 0 \\ 0 & 0 & -\frac{\sqrt{21}}{35} & 0 & \frac{2}{15} \end{bmatrix} \tag{4.28}$$

Finally, for case 1 in Fig. IV-4, the neighboring contribution is $\frac{L_{AB}}{6}\mathrm{T}_i^T\underline{\mathrm{M}}\,\underline{\mathrm{R}}\,\mathrm{T}_j$. Because both the 1-D mass matrix and the rotation matrix are symmetric, their sequence can be exchanged. It is useful to note that the edge coupling matrix $\mathrm{E}_{i,j}^{1C}$ can be obtained again, since

$$\mathrm{E}_{i,j}^{1C} = \mathrm{T}_i^T\underline{\mathrm{M}}\,\underline{\mathrm{R}}\,\mathrm{T}_j \tag{4.29}$$

T_i and T_j are element independent, so is the type-1 edge coupling matrix $\mathrm{E}_{i,j}^{1C}$.

In the case of local refinement (edge irregularity), we need to use 1-D edge prolongation matrices defined as follows

$$\underline{\mathrm{P}}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & -\frac{\sqrt{6}}{4} & 0 & \frac{\sqrt{14}}{16} \\ 0 & 0 & \frac{1}{4} & -\frac{\sqrt{15}}{8} & \frac{\sqrt{21}}{16} \\ 0 & 0 & 0 & \frac{1}{8} & -\frac{\sqrt{35}}{16} \\ 0 & 0 & 0 & 0 & \frac{1}{16} \end{bmatrix} \tag{4.30}$$

$$\underline{\mathrm{P}}_2 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{\sqrt{6}}{4} & 0 & \frac{\sqrt{14}}{16} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & \frac{\sqrt{15}}{8} & \frac{\sqrt{21}}{16} \\ 0 & 0 & 0 & \frac{1}{8} & \frac{\sqrt{35}}{16} \\ 0 & 0 & 0 & 0 & \frac{1}{16} \end{bmatrix} \tag{4.31}$$

Multiplying these prolongation matrices with the 1-D solution vector of an edge gives the 1-D solution vector of the child elements connected to the edge.

For instance, in case 2 of Fig. IV-4, the edge contribution from elements K_1' and

K_2' to K is

$$\frac{L_{AB}}{6}(\mathrm{T}_i^T \underline{\mathrm{P}}_1^T \underline{\underline{\mathrm{M}}}\, \underline{\underline{\mathrm{R}}}\, \mathrm{T}_j)\mathbf{u}_{K_2'} + \frac{L_{BC}}{6}(\mathrm{T}_i^T \underline{\mathrm{P}}_2^T \underline{\underline{\mathrm{M}}}\, \underline{\underline{\mathrm{R}}}\, \mathrm{T}_j)\mathbf{u}_{K_1'} \tag{4.32}$$

If there are multiple hanging nodes on the edge (i.e., the level difference between the element and its neighbors is greater than 1), we again can recursively use the 1-D prolongation matrices. For example, the contribution for element K in Fig. IV-5 is,

$$\frac{L_{AB}}{6}(\mathrm{T}_i^T \underline{\mathrm{P}}_1^T \underline{\underline{\mathrm{M}}}\, \underline{\underline{\mathrm{R}}}\, \mathrm{T}_j)\mathbf{u}_{K_2'} + \frac{L_{BC}}{6}(\mathrm{T}_i^T \underline{\mathrm{P}}_2^T \underline{\mathrm{P}}_1^T \underline{\underline{\mathrm{M}}}\, \underline{\underline{\mathrm{R}}}\, \mathrm{T}_j)\mathbf{u}_{K_{12}'} + \frac{L_{CD}}{6}(\mathrm{T}_i^T \underline{\mathrm{P}}_2^T \underline{\mathrm{P}}_2^T \underline{\underline{\mathrm{M}}}\, \underline{\underline{\mathrm{R}}}\, \mathrm{T}_j)\mathbf{u}_{K_{11}'} \tag{4.33}$$

Finally, consider case 3 of Fig. IV-4. The contribution for element K is simply given by

$$\frac{L_{AB}}{6}(\mathrm{T}_i^T \underline{\underline{\mathrm{M}}}\, \underline{\mathrm{P}}_2\, \underline{\underline{\mathrm{R}}}\, \mathrm{T}_j)\mathbf{u}_{K_2'} \tag{4.34}$$

Considering the right pane of Fig. IV-5, a case with multi-irregularity, the contributions from K' to K is given by

$$\frac{L_{AB}}{6}(\mathrm{T}_i^T \underline{\underline{\mathrm{M}}}\, \underline{\mathrm{P}}_1\, \underline{\mathrm{P}}_2\, \underline{\underline{\mathrm{R}}}\, \mathrm{T}_j)\mathbf{u}_{K_2'} \tag{4.35}$$

We can define other edge operations to deal with the other edge coupling terms, needed in the DFEM diffusion solver. The following matrices given below should replace the use of the T_i $(i = 1, 2, 3)$ matrices above. To obtain the outward normal derivative on an edge, we need to define the following matrices (for conciseness, we only give the matrices up to order 3; the reader can generate the order-4 matrices

with the Mathematica notebook provided in Appendix F.)

$$\mathbf{N}_{K,1} = \frac{2}{L_1}\begin{bmatrix} -r_1 & \frac{\cot\alpha_3}{2} & \frac{\cot\alpha_2}{2} & -\frac{\sqrt{6}\cot\alpha_2}{2} & 0 & \sqrt{6}r_1 & \frac{\sqrt{10}\cot\alpha_2}{2} & 0 & \sqrt{10}r_1 & 0 \\ -r_1 & \frac{\cot\alpha_3}{2} & \frac{\cot\alpha_2}{2} & -\frac{\sqrt{6}\cot\alpha_3}{2} & \sqrt{6}r_1 & 0 & -\frac{\sqrt{10}\cot\alpha_3}{2} & -\sqrt{10}r_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{15}(r_3-r_2) & -\frac{\sqrt{15}r_1}{3} & \frac{\sqrt{15}r_1}{3} & \sqrt{6}r_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{N}_{K,2} = \frac{2}{L_2}\begin{bmatrix} \frac{\cot\alpha_3}{2} & -r_2 & \frac{\cot\alpha_1}{2} & \sqrt{6}r_1 & -\frac{\sqrt{6}\cot\alpha_3}{2} & 0 & \sqrt{10}r_2 & \frac{\sqrt{10}\cot\alpha_3}{2} & 0 & 0 \\ \frac{\cot\alpha_3}{2} & -r_2 & \frac{\cot\alpha_1}{2} & 0 & -\frac{\sqrt{6}\cot\alpha_1}{2} & \sqrt{6}r_2 & 0 & -\frac{\sqrt{10}\cot\alpha_1}{2} & -\sqrt{10}r_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{15}r_2}{3} & \sqrt{15}(r_1-r_3) & -\frac{\sqrt{15}r_2}{3} & \sqrt{6}r_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.36}$$

$$\mathbf{N}_{K,3} = \frac{2}{L_3}\begin{bmatrix} \frac{\cot\alpha_2}{2} & \frac{\cot\alpha_1}{2} & -r_3 & 0 & \sqrt{6}r_3 & -\frac{\sqrt{6}\cot\alpha_1}{2} & 0 & \sqrt{10}r_3 & \frac{\sqrt{10}\cot\alpha_1}{2} & 0 \\ \frac{\cot\alpha_2}{2} & \frac{\cot\alpha_1}{2} & -r_3 & \sqrt{6}r_3 & 0 & -\frac{\sqrt{6}\cot\alpha_2}{2} & -\sqrt{10}r_3 & 0 & -\frac{\sqrt{10}\cot\alpha_2}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{\sqrt{15}r_3}{3} & \frac{\sqrt{15}r_3}{3} & \sqrt{15}(r_2-r_1) & \sqrt{6}r_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Note this edge operation matrix depends on the shape of the real (physical) element. Next, we define some geometrical variables for a given triangle,

$$\begin{aligned} y_{12} &= \frac{y_1 - y_2}{2A}, \qquad x_{21} = \frac{x_2 - x_1}{2A} \\ y_{23} &= \frac{y_2 - y_3}{2A}, \qquad x_{32} = \frac{x_3 - x_2}{2A} \\ y_{31} &= \frac{y_3 - y_1}{2A}, \qquad x_{13} = \frac{x_1 - x_3}{2A}. \end{aligned} \tag{4.37}$$

In order to obtain the derivative in the x-direction, we define the following edge operation matrices:

$$\mathbf{X}_{K,1} = \begin{bmatrix} y_{23} & y_{31} & y_{12} & -\sqrt{6}y_{12} & 0 & -\sqrt{6}y_{23} & \sqrt{10}y_{12} & 0 & -\sqrt{10}y_{23} & 0 \\ y_{23} & y_{31} & y_{12} & -\sqrt{6}y_{31} & -\sqrt{6}y_{23} & 0 & -\sqrt{10}y_{31} & \sqrt{10}y_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{15}(y_{12}-y_{31}) & \frac{\sqrt{15}y_{23}}{3} & -\frac{\sqrt{15}y_{23}}{3} & -\sqrt{6}y_{23} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{X}_{K,2} = \begin{bmatrix} y_{23} & y_{31} & y_{12} & -\sqrt{6}y_{31} & -\sqrt{6}y_{23} & 0 & -\sqrt{10}y_{31} & \sqrt{10}y_{23} & 0 & 0 \\ y_{23} & y_{31} & y_{12} & 0 & -\sqrt{6}y_{12} & -\sqrt{6}y_{31} & 0 & -\sqrt{10}y_{12} & \sqrt{10}y_{31} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{\sqrt{15}y_{31}}{3} & \sqrt{15}(y_{23}-y_{12}) & \frac{\sqrt{15}y_{31}}{3} & -\sqrt{6}y_{31} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.38}$$

$$\mathbf{X}_{K,3} = \begin{bmatrix} y_{23} & y_{31} & y_{12} & 0 & -\sqrt{6}y_{12} & -\sqrt{6}y_{31} & 0 & -\sqrt{10}y_{12} & \sqrt{10}y_{31} & 0 \\ y_{23} & y_{31} & y_{12} & -\sqrt{6}y_{12} & 0 & -\sqrt{6}y_{23} & \sqrt{10}y_{12} & 0 & -\sqrt{10}y_{23} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{15}y_{12}}{3} & -\frac{\sqrt{15}y_{12}}{3} & \sqrt{15}(y_{31}-y_{23}) & -\sqrt{6}y_{12} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Similarly, for the derivative in the y-direction, we have

$$\mathbf{Y}_{K,1} = \begin{bmatrix} x_{32} & x_{13} & x_{21} & -\sqrt{6}x_{21} & 0 & -\sqrt{6}x_{32} & \sqrt{10}x_{21} & 0 & -\sqrt{10}x_{32} & 0 \\ x_{32} & x_{13} & x_{21} & -\sqrt{6}x_{13} & -\sqrt{6}x_{32} & 0 & -\sqrt{10}x_{13} & \sqrt{10}x_{32} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{15}(x_{21}-x_{13}) & \frac{\sqrt{15}x_{32}}{3} & -\frac{\sqrt{15}x_{32}}{3} & -\sqrt{6}x_{32} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{Y}_{K,2} = \begin{bmatrix} x_{32} & x_{13} & x_{21} & -\sqrt{6}x_{13} & -\sqrt{6}x_{32} & 0 & -\sqrt{10}x_{13} & \sqrt{10}x_{32} & 0 & 0 \\ x_{32} & x_{13} & x_{21} & 0 & -\sqrt{6}x_{21} & -\sqrt{6}x_{13} & 0 & -\sqrt{10}x_{21} & \sqrt{10}x_{13} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{\sqrt{15}x_{13}}{3} & \sqrt{15}(x_{32}-x_{21}) & \frac{\sqrt{15}x_{13}}{3} & -\sqrt{6}x_{13} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.39}$$

$$\mathbf{Y}_{K,3} = \begin{bmatrix} x_{32} & x_{13} & x_{21} & 0 & -\sqrt{6}x_{21} & -\sqrt{6}x_{13} & 0 & -\sqrt{10}x_{21} & \sqrt{10}x_{13} & 0 \\ x_{32} & x_{13} & x_{21} & -\sqrt{6}x_{21} & 0 & -\sqrt{6}x_{32} & \sqrt{10}x_{21} & 0 & -\sqrt{10}x_{32} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{15}x_{21}}{3} & -\frac{\sqrt{15}x_{21}}{3} & \sqrt{15}(x_{13}-x_{32}) & -\sqrt{6}x_{21} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

6. Mesh Coupling

We now address the issue related to having group-dependent adapted meshes with a multigroup solver. Multigroup equations are coupled via fission and scattering reactions, leading to coupling terms in between groups that require mass matrices, where the test function lives on one mesh, say g, and the shape function lives of the g' mesh.

First, let us describe the standard case, where a single mesh is employed for all energy groups. In this situation, we can simply follow the four steps given below to construct the (angular) directional source for any energy group g and direction m in the DGFEM transport sweep:

1. First, obtain the (n,k) moment of the source term due to energy transfers from all other groups

$$Q^g_{n,k}(\vec{r}) = \delta_{n,0}\chi_g(\vec{r})\sum_{g'=1}^{G}\nu\sigma_{f,g'}(\vec{r})\Phi^{g'}_{n,k}(\vec{r}) + \sum_{\substack{g'=1 \\ g'\neq g}}^{G}\sigma^{g'\to g}_{s,n}(\vec{r})\Phi^{g'}_{n,k}(\vec{r}) = \sum_{g'=1}^{G} f^{gg'}_{n,k}(\vec{r})\Phi^{g'}_{n,k}(\vec{r}) \tag{4.40}$$

2. Second, test the above source moment with all basis functions and carry out the inner product (integration) to form the (n, k) moment of the right-hand-side. This is equivalent to multiplying the global mass matrix with the source moment vector. This multiplication can be done element-wise in a matrix-free fashion.

3. Then, apply the spherical harmonics coefficient $\frac{2n+1}{4\pi} Y_{n,k}(\vec{\Omega}_m)$ to this right-hand-side source moment for the chosen direction m.

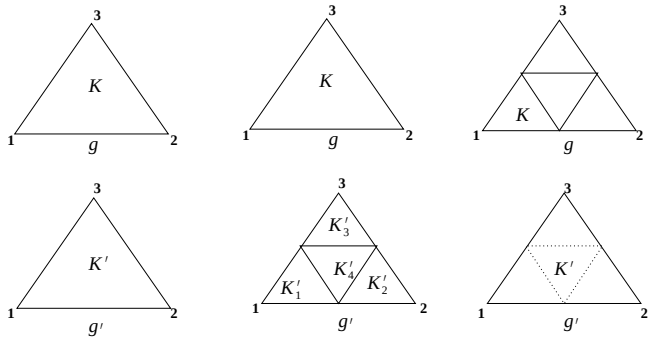
4. Finally, sum the contributions from all moments together ($\sum_{n=0}^{N} \sum_{k=-n}^{k=n} \ldots$).

In our implementation, the various flux moments and angular fluxes, for a given energy group, share the same mesh, so we can keep the last two steps. However, because different meshes for different energy groups are used, we had to combine the first two steps together in order to evaluate the right-hand-side source moment directly. This is explained below and constitutes what we refer to as "multi-mesh".

The idea to calculate the right-hand-side source moment for one group g with solutions of other groups on different meshes is pictured on Fig. IV-7. In general, integrating terms that involve functions defined on two entirely different meshes is an expensive procedure, since it involves finding the cell of one mesh in which a quadrature point defined on the other mesh lies. The integration will in this case have a complexity higher than $O(N)$, i.e., the number of operations will grow faster than linearly with the number of cells N. However, this problem can be avoided if we use hierarchical meshes that result from refinement of the same chosen initial coarse mesh.

With the common initial coarse mesh and the regular refinement, we can always find a set of cells, which we denote by $\mathbb{T}_{gg',h}$, that satisfy the following conditions:

- the union of the cells covers the entire domain, and



(a) Case 1 (a) Case 2 (a) Case 3

Fig. IV-7. Three cases of multi-mesh coupling.

- a cell in mesh $\mathbb{T}_{gg',h}$ is active on at least one of the two meshes $\mathbb{T}_{g,h}$ or $\mathbb{T}_{g',h}$.

Then for all of these cells, we will again have three basic situations to deal with, similar to the cases of edge flux mapping described in the preceding Section; the three cases of mesh-coupling are depicted in Fig. IV-7.

The first case is simple since there are no refinement and the shape and test functions live on the same local mesh common to both energy groups. The contribution from group g' to group g for the element K is

$$\frac{f^{gg'}_{K,n,k}A_K}{12}\mathrm{M}\mathbf{\Phi}^{g'}_{K,n,k}. \tag{4.41}$$

Using the same methodology developed for edge mapping, we deal with the second and the third cases of mesh-coupling next. The contribution group g' to element K in group g in case 2 and case 3 are, respectively

$$\sum_{i=1}^{4}\frac{f^{gg'}_{K,n,k}A_{K'_i}}{12}\mathrm{P}_i^T\mathrm{M}\mathbf{\Phi}^{g'}_{K'_i,n,k} \tag{4.42}$$

$$\frac{f^{gg'}_{K,n,k}A_K}{12}\mathrm{MP}_1\mathbf{\Phi}^{g'}_{K',n,k} \tag{4.43}$$

where A represent the triangle area.

In the cases where the difference in refinement levels is greater than 1, we can apply the prolongation matrices P_i, $i = 1, 2, 3, 4$ recursively for the cases where the maximum refinement level difference between any cells in a common cell of $\mathbb{T}_{gg',h}$ of the two meshes is greater than 1. The methodology applies to arbitrary level differences, although in practice, we have limited that level difference to 6.

Implementation-wise, we naturally order all active cells. Therefore, we do not need to visit the mesh information for each mesh coupling and the algorithm can be used indifferently in a matrix-free scheme or within a standard matrix assembly procedure. If the number of energy groups is smaller than the number of meshes, i.e., some energy groups share the same mesh, we can combine their solution vectors into the source moment first and then apply the mesh coupling algorithm. The mesh coupling operation needs to be done only for different meshes. Our numerical experiments have shown that mesh coupling can be efficiently implemented.

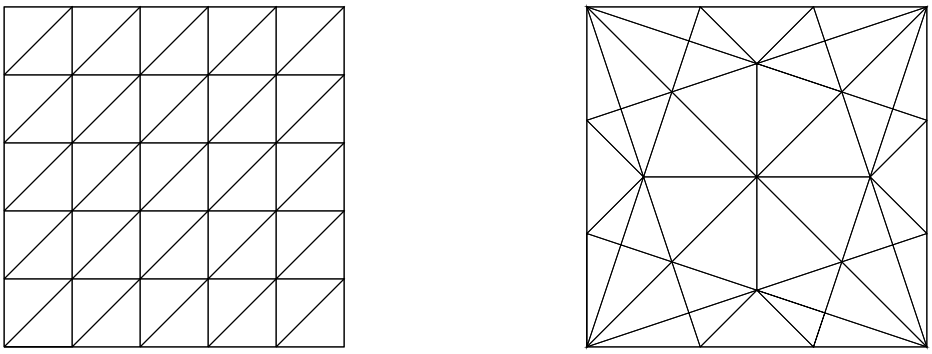
This algorithm can be extended for projecting the solution from one mesh to another easily because a DG method is used. The global mass matrix is block diagonal, hence, its inverse is easily computed element-wise. Projecting the solution onto another mesh is useful for bootstrapping the numerical procedure. Once a numerical solution has been obtained and its flux moments $\Phi_{n,k}$ computed, the spatial error distribution is assessed using the jump-based error estimated with Eq. (4.2) (or using any other error estimate described earlier) and a new adapted mesh $\mathbb{T}_h^{k+1}$ is prescribed. In order to bootstrap the solution procedure on that new mesh, the flux moments $\Phi_{n,k}$ are projected on $\mathbb{T}_h^{k+1}$. For bootstrapping, we also need to project the SAF on selected edges from one mesh to another. Again, if we number these edges appropriately, we have a similar 1-D algorithm to perform the projection.

C. Results

We present five examples to validate our approach.

1. Example 1: One-group Source Driven Problem

This first example consists of an homogeneous medium placed in vacuum. The domain size is $[10 \times 10]$ cm^2; the total cross section σ_t is varied from 0.1 to 100 cm^{-1} to model a wide range of domain optical thicknesses (from 1 MFP to 1000 MFP). The scattering ratio $c = \sigma_{s,0}/\sigma_t$ is chosen to be equal to 0.9. A volumetric uniform and isotropic source is imposed and a level-symmetric S_4 angular quadrature is used for the angular discretization. A 2-irregularity constraint is set in the AMR process. Two different initial meshes are chosen and shown on Fig. IV-8: a structured regular mesh and an unstructured mesh obtained with the Triangle mesh generator.



(a) Regular mesh (b) Aligned mesh

Fig. IV-8. Initial meshes for Example 1.

It is important to note that the unstructured mesh is aligned with the corner singularities of the S_N transport equation (using S_4 in 2-D leads to 3 singular lines per corner, thus a total of 12 singularity lines here). For a mesh aligned with the singularities, the convergence is not restricted by the regularity of the solution. The GMRes solver without preconditioning is used for this problem; tolerance tol_{source} is

set to 10^{-12}. $\mu_{g,K}^{k,ref}$ is used for the projection-based error estimator, i.e., two solutions on both the coarse mesh $\mathbb{T}_h^k$ and the finer mesh $\mathbb{T}_{h/2}^k$ are obtained at each cycle of mesh adaptation. The convergence plots are generated in the following way: we first use the projection-based error estimator to drive the AMR with highest polynomial order 4, and obtain a much accurate reference solution; then we can evaluate the numerical "exact" spatial error of the scalar flux of group g (here only one group) at any cycle k like Eq. (4.8),

$$\epsilon_g^{k,ref} = \frac{\left\|\Phi_{g,h} - \Phi_g^{ref}\right\|_2}{\left\|\Phi_{g,h}\right\|_2} \tag{4.44}$$

and plot them with respect to the number of unknowns of the mesh $\mathbb{T}_{h,g}^k$.

Fig. IV-9 shows the convergence rates as a function of the number of unknowns when utilizing the unstructured aligned mesh as initial mesh. Fig. IV-10 shows the convergence rates for various domain optical thicknesses as a function of the number of unknowns when utilizing the structured regular mesh as initial mesh. Each pane of Figs. IV-9 and IV-10 present 12 curves; the following three refinement strategies are plotted: uniform mesh refinement (black lines), AMR refinement driven by the jump-based estimator (red lines) and AMR driven by the projection-based estimator (blue lines); each strategy is displayed for polynomial orders 1 through 4 (squares for $p = 1$, circles for $p = 2$, crosses for $p = 3$ and diamonds for $p = 4$).

First, we discuss the results obtained using the initial mesh aligned with the transport singularities (Fig. IV-9). For uniform refinement, we note that the asymptotic convergence rates behave as $(p+1)/2$ as a function of the number of unknowns, which translates into orders of $p + 1$ as a function of mesh size (the number of mesh cells, thus of unknowns, is proportional to the square of the mesh size for uniform refinement). The exception is the case with domain size equal to 1 MFP. This may suggest that the 12 singularity lines may not be the only places where singularities

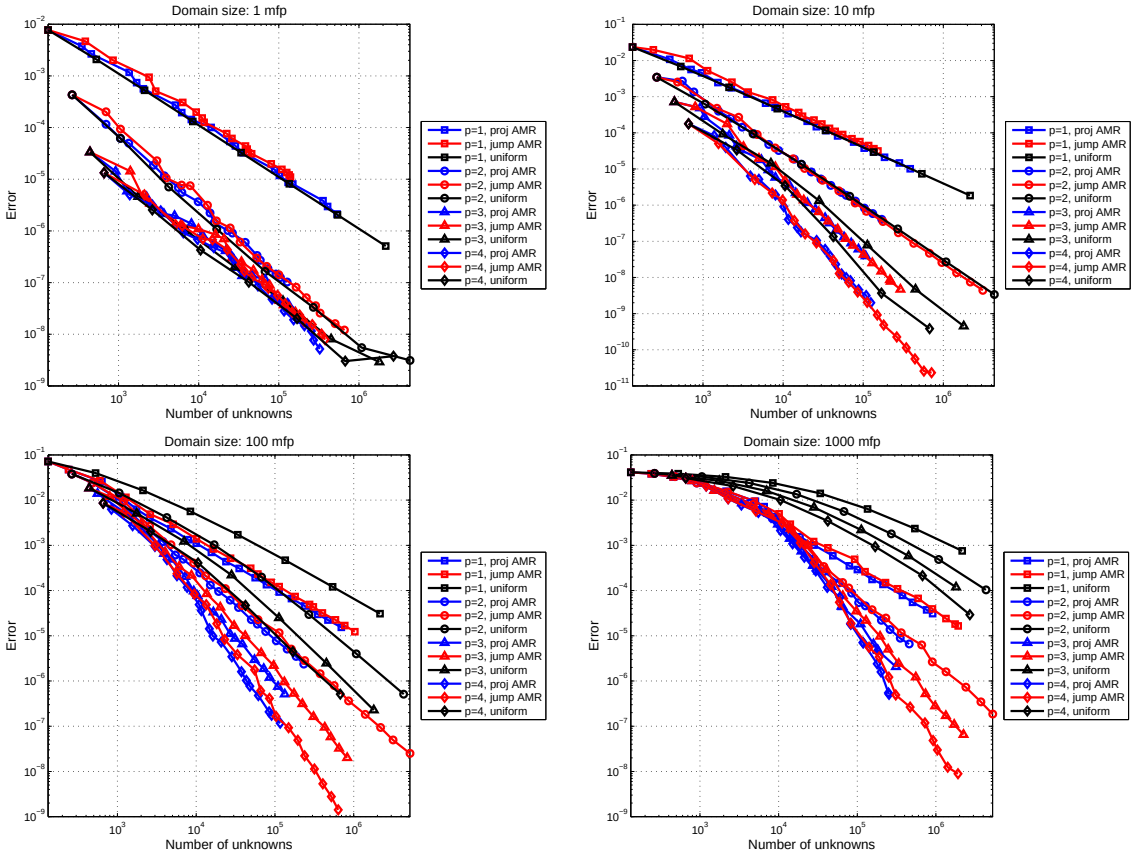


Fig. IV-9. Convergence rates for various domain thicknesses and various polynomial orders; case of the aligned initial mesh.

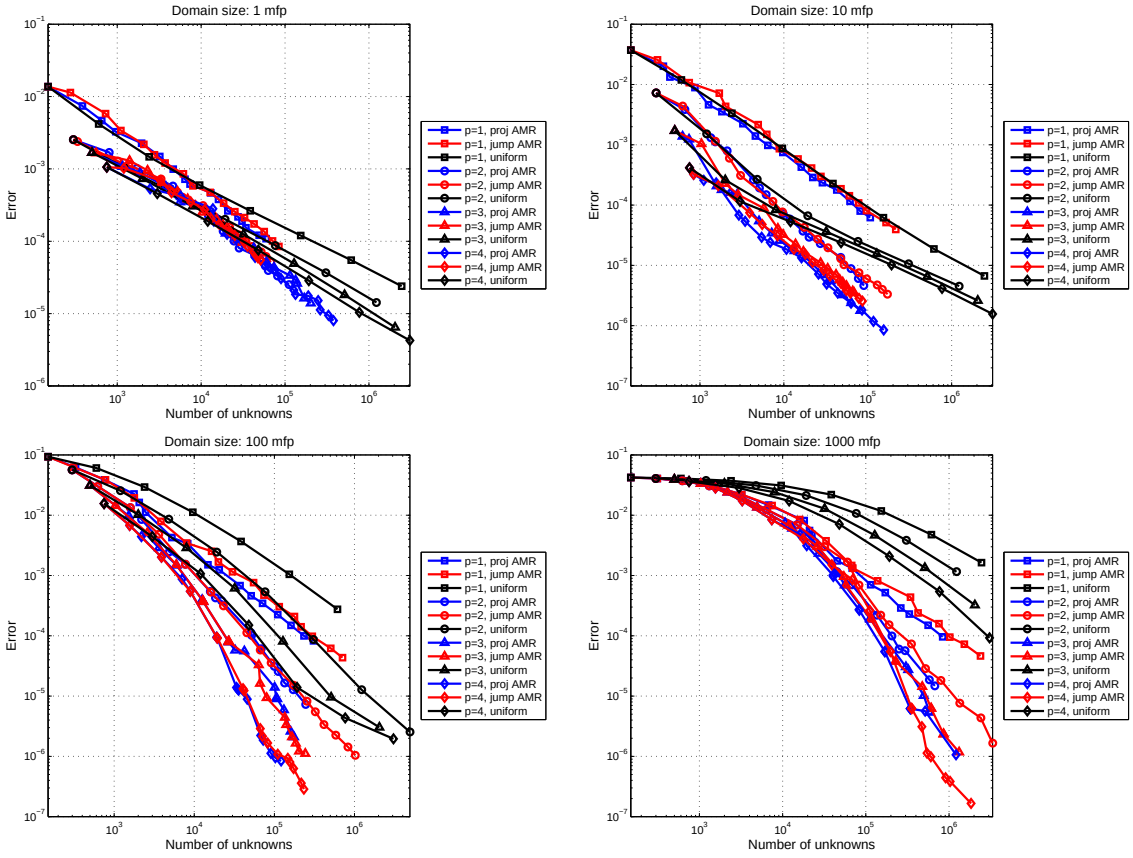


Fig. IV-10. Convergence rates for various domain thicknesses and various polynomial orders; case of the regular initial mesh.

are present. The results with AMR show a vast improvement over the uniform mesh approach for optically large domains. With AMR, the number of unknowns can be reduced by 1 to 1.5 orders of magnitude. It is clear that the higher-order results are significantly better in terms of number of unknowns.

In light of these explanations, we proceed with the discussion of Fig. IV-10 showing the convergence history for meshes that are not aligned with the singularities. Here, the regularity in the transport solution constrains the convergence rates for optically thin domain (see also a similar discussion in Chapter II) in the uniform refinement case. For large domain sizes, the AMR solutions are very accurate before the convergence rates enter the singularity-constrained asymptotic range. With adapted meshes, not only the convergence rates at the asymptotic range are improved, but also the numbers of unknowns are much smaller than the ones of uniform meshes to obtain the same level of accuracy. The projection-based error estimator delivers meshes slightly better than with the jump-based error indicator but their differences are small similar as shown in the results plotted in Fig. IV-9.

At this stage, we need to point out that, for the projection-based AMR, *only* the number of unknowns of mesh $\mathbb{T}_h^k$ at the k-th adaptivity cycle has been graphed. The error of adapted solutions on both the coarse mesh $\mathbb{T}_h^k$ and the finer mesh $\mathbb{T}_{h/2}^k$ for all refinement cycles are compared in Figs IV-11 and IV-12. We can see that when the AMR strategy reaches the asymptotic range, the convergence curves of the coarse solution and the finer solution almost overlap, which means that the finer meshes are almost as good as the coarse mesh. The coarse meshes are better when the refinement is not at the asymptotic range, which can easily been seen in the 100-MFP and 1000-MFP plots.

To complete the study of the convergence histories, we graph the accuracy reached as a function of the CPU time on Figs. IV-13 through IV-14 for the two

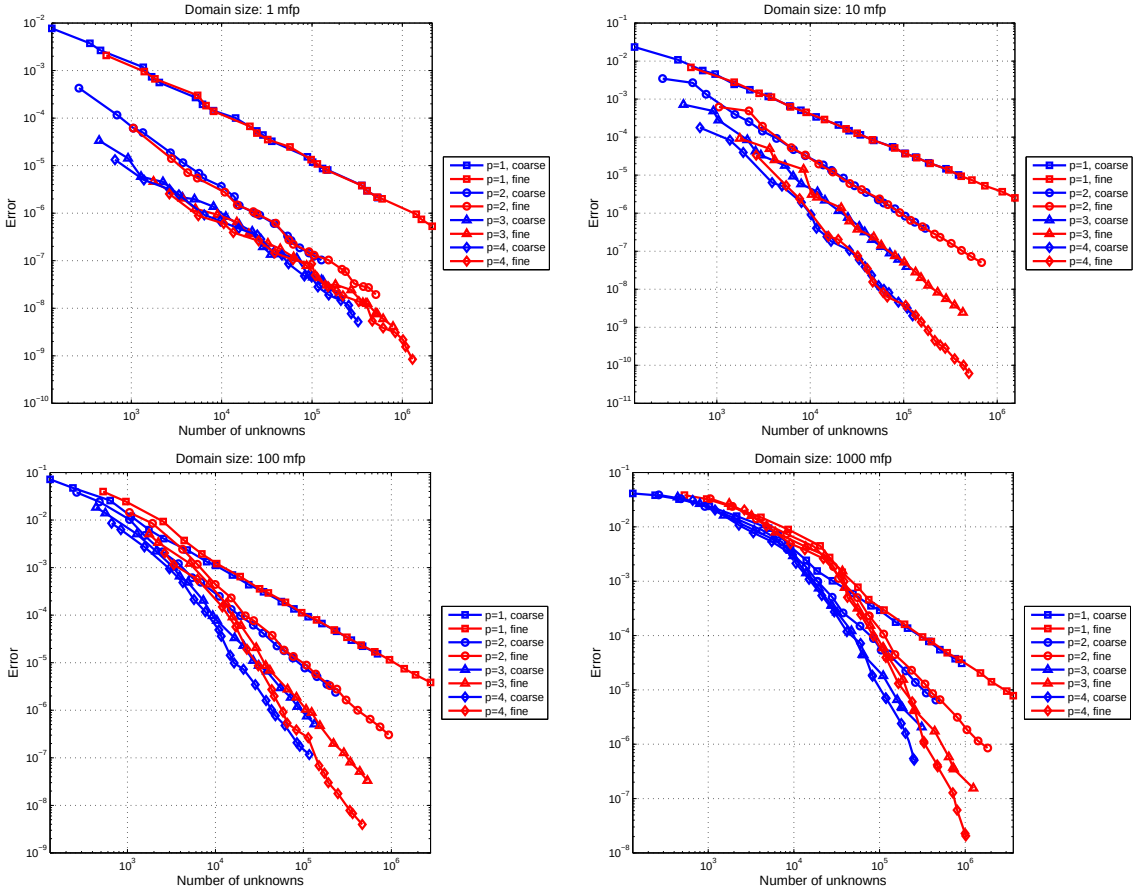


Fig. IV-11. Comparison of the coarse and finer solutions with the aligned initial mesh.

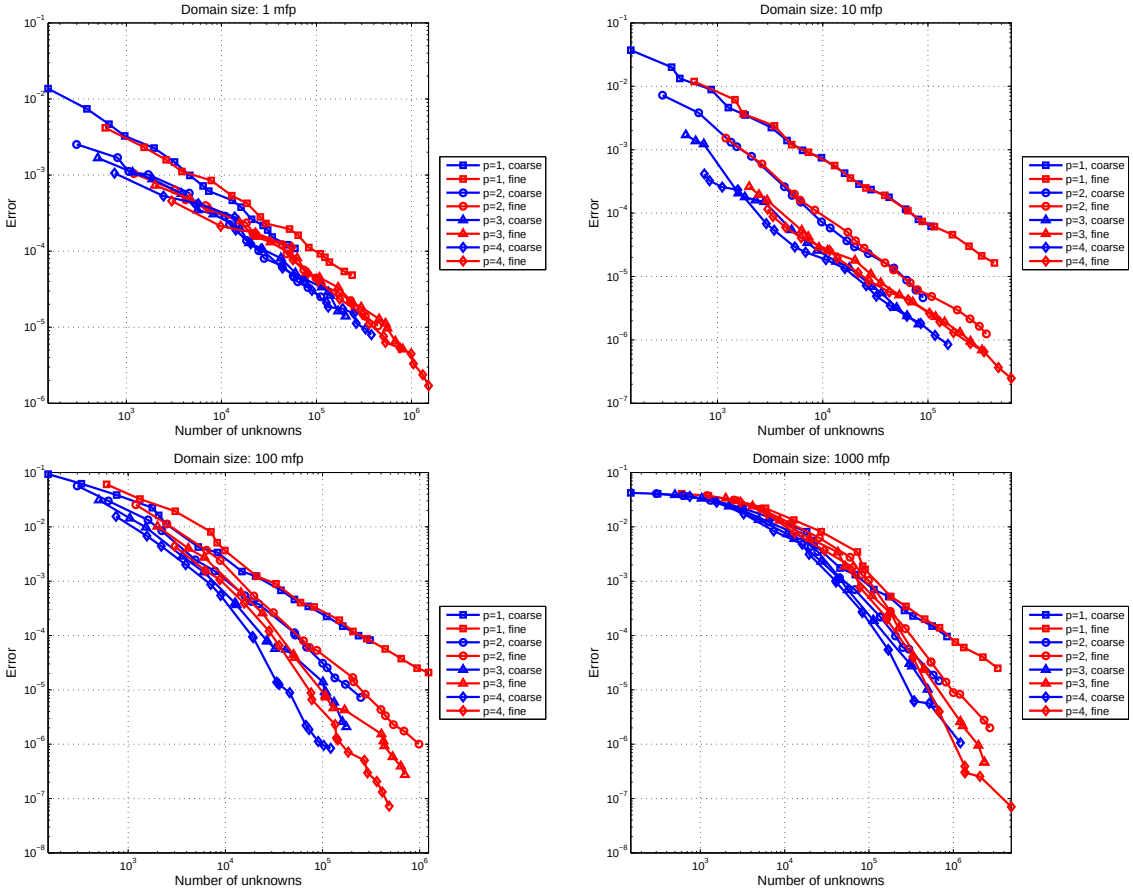


Fig. IV-12. Comparison of the coarse and finer solutions with the regular initial mesh.

initial meshes. At each cycle, the computing time is the accumulated time of all previous cycles. For the projection-based error estimator, the finer error is plotted with the computing time of the two solves per cycle together. This is the reason why the computing times are slightly larger with the projection-based estimator than the times with the jump-based indicator. For very thin domains, there is a modest gain using AMR techniques. As the domain thickness increases, both AMR techniques utilized present far better results than a uniform mesh refinement approach. For instance, AMR with DGFEM(1) is about one order of magnitude less costly in number of unknowns for the same accuracy as uniform refinement. In addition, AMR with quadratic, cubic or quartic polynomials is about two times better than DGFEM(1) with AMR.

Finally, we also compare the adaptive meshes generated using the two AMR approaches in Fig. IV-15. The left mesh is the adapted mesh at cycle 14 with the projection-based error estimator and $\alpha = 1/3$. Number of active elements in this mesh is 5042. The right mesh is the adapted mesh at cycle 13 with the jump-based error indicator and $\alpha = 0.2$. Number of active elements in this mesh is 5024. Numbers of active elements of these two meshes are about the same and both meshes have the uniform polynomial order 2. The numerically "exact" relative error of scalar flux in L_2 norm on the projection-based mesh is 3.20×10^{-6} while the error on the jump-based mesh is 5.22×10^{-6}. Both the projection-based and the jump-based error estimations are able to detect the singularity lines when the mesh is not aligned with the singularities.

Grind time are shown in Tables IV-I through IV-IV with four different polynomial orders. Only the case with domain size 10-by-10 MFP and projection-based AMR is considered. The grind time is defined as the average computing time per unknown for solving one single task in the transport sweeps. This grind time includes the time

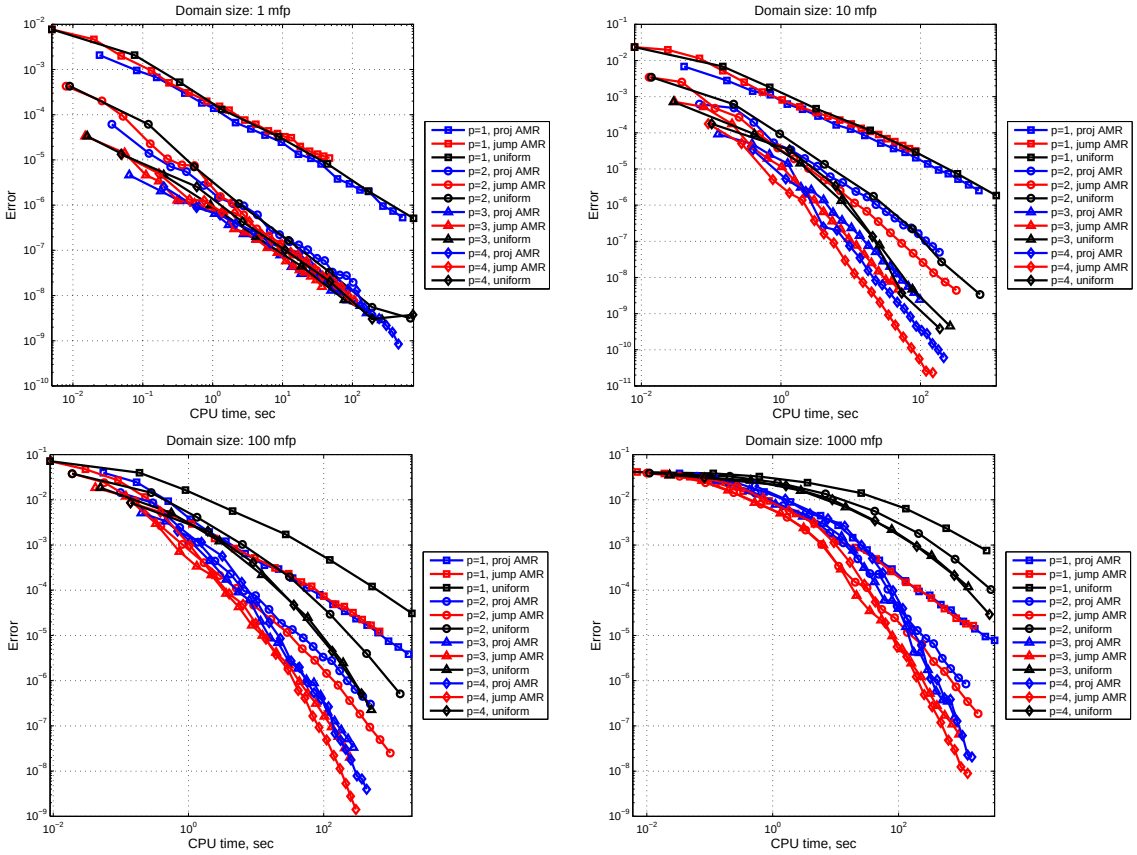


Fig. IV-13. CPU time of AMR with the aligned initial mesh.

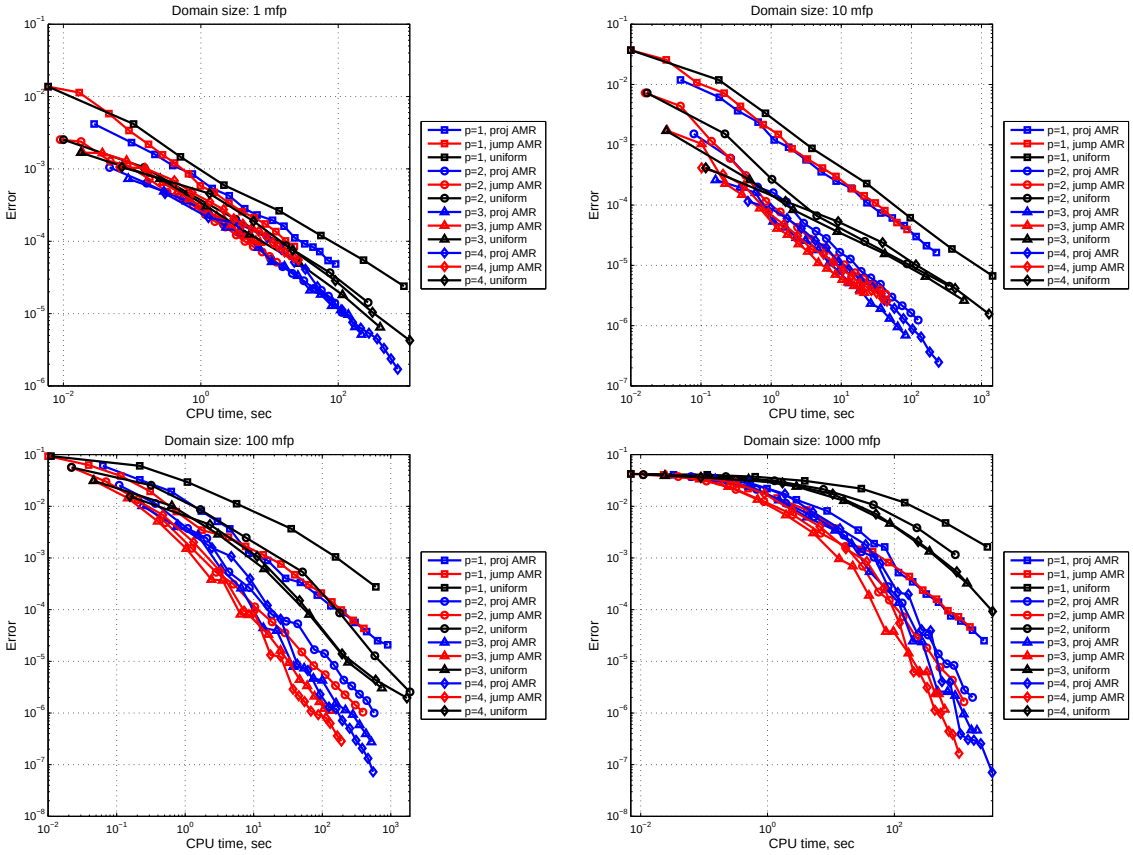
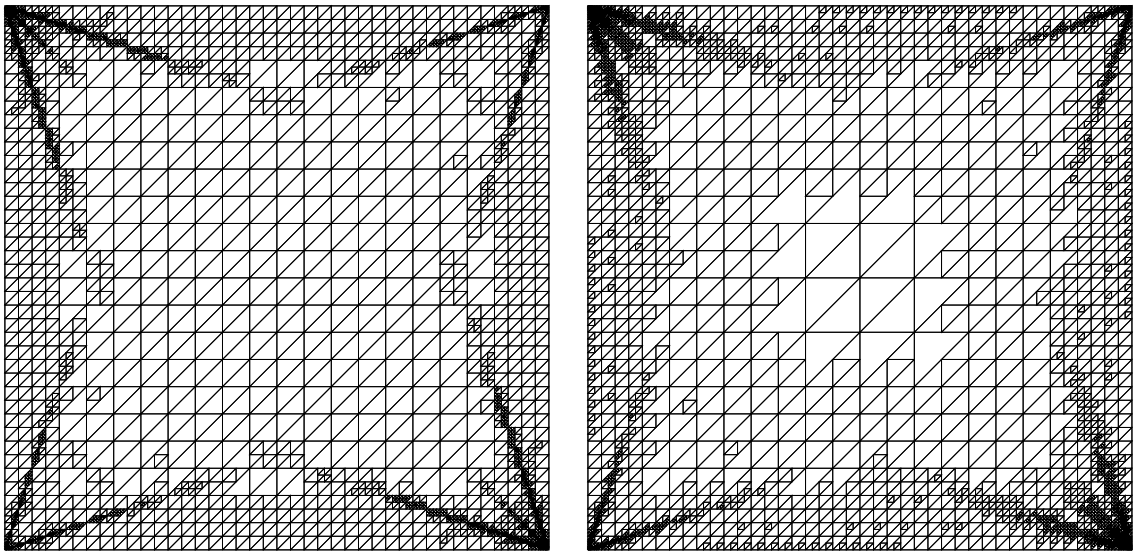


Fig. IV-14. CPU time of AMR with the regular initial mesh.



(a) Projection-based (b) Jump-based

Fig. IV-15. Meshes with different error estimations.

needed to assemble the local system, solving it and updating the flux moments. The time on construction of the source moments is not counted in the grind time but it is usually a small fraction of the computing time in the transport sweeps.

Results show that the grind times of quadratic and cubic elements are even smaller than the linear element although the numbers of local operations per unknown are much larger than the one with the linear element. These results suggest the computing time is dominated by accessing data from the memory. The grind time with polynomial order 4 is significantly larger than others may be due to cache missing with a larger dimension of local system ($N_p(4) = 15$). The presence of mesh irregularity does not impact much the grind time.

2. Example 2: Search-light Problem

This example models a searchlight problem, where an incident beam of radiation is propagated through in a vacuum. For instance, a similar problem has been studied in

Table IV-I. Grind time of polynomial order 1.

Cycle ID	Number of active cells	Irregularity index	Number of sweeps	Number of tasks on $\mathbb{T}_h^k$	CPU time in sweeps (sec)	Average grind time (μs/unknown)	Number of sweeps	Number of tasks on $\mathbb{T}_{h/2}^k$	CPU time in sweeps (sec)	Average grind time (μs/unknown)
0	50	0.0000	21	12600	0.01	0.132	24	50400	0.02	0.132
1	122	0.1019	22	32208	0.01	0.114	25	128832	0.05	0.140
2	146	0.0622	23	40296	0.01	0.124	24	161184	0.07	0.141
3	290	0.0720	23	80040	0.03	0.129	24	320160	0.14	0.144
4	422	0.0662	23	116472	0.04	0.120	23	465888	0.20	0.142
5	608	0.0327	22	160512	0.06	0.133	23	642048	0.29	0.151
6	1082	0.0710	23	298632	0.12	0.129	23	1194528	0.59	0.164
7	1526	0.0512	22	402864	0.16	0.129	22	1611456	0.91	0.188
8	2144	0.0325	22	566016	0.24	0.140	22	2264064	1.48	0.218
9	3242	0.0535	22	855888	0.41	0.160	22	3423552	2.46	0.239
10	5378	0.0428	21	1355256	0.74	0.183	21	5421024	4.14	0.254
11	7364	0.0349	20	1767360	1.12	0.211	21	7069440	5.74	0.270
12	9596	0.0403	20	2303040	1.61	0.232	20	9212160	7.03	0.254
13	14240	0.0547	20	3417600	2.57	0.251	20	13670400	10.68	0.260
14	20474	0.0355	19	4668072	3.59	0.256	20	18672288	15.26	0.272
15	27050	0.0316	19	6167400	4.71	0.255	19	24669600	19.32	0.261
16	35714	0.0320	19	8142792	6.28	0.257	19	32571168	25.46	0.261

Table IV-II. Grind time of polynomial order 2.

Cycle ID	Number of active cells	Irregularity index	Number of sweeps	Number of tasks on $\mathbb{T}_h^k$	CPU time in sweeps (sec)	Average grind time (μs/unknown)	Number of sweeps	Number of tasks on $\mathbb{T}_{h/2}^k$	CPU time in sweeps (sec)	Average grind time (μs/unknown)
0	50	0.0000	23	13800	0.01	0.133	23	55200	0.04	0.127
1	110	0.1805	23	30360	0.02	0.126	24	121440	0.09	0.128
2	224	0.2435	22	59136	0.04	0.116	23	236544	0.19	0.136
3	254	0.2330	22	67056	0.05	0.122	23	268224	0.20	0.126
4	350	0.1806	22	92400	0.06	0.114	22	369600	0.29	0.131
5	722	0.1998	21	181944	0.14	0.125	21	727776	0.60	0.138
6	854	0.1745	21	215208	0.17	0.130	20	860832	0.69	0.135
7	1028	0.1500	20	246720	0.19	0.129	20	986880	0.90	0.152
8	1628	0.1447	20	390720	0.31	0.133	20	1562880	1.63	0.174
9	1958	0.1415	19	446424	0.37	0.137	19	1785696	1.95	0.182
10	2822	0.1897	18	609552	0.54	0.148	18	2438208	2.79	0.191
11	3332	0.1804	18	719712	0.66	0.154	18	2878848	3.32	0.192
12	4526	0.2035	18	977616	0.94	0.160	18	3910464	4.59	0.196
13	7922	0.2026	17	1616088	1.80	0.186	17	6464352	7.53	0.194
14	10478	0.2190	16	2011776	2.32	0.192	16	8047104	9.53	0.197
15	12848	0.2277	15	2312640	2.74	0.198	15	9250560	10.94	0.197
16	14978	0.2165	14	2516304	3.00	0.199	14	10065216	12.06	0.200

Table IV-III. Grind time of polynomial order 3.

Cycle ID	Number of active cells	Irregularity index	Number of sweeps	Number of tasks on $\mathbb{T}_h^k$	CPU time in sweeps (sec)	Average grind time (μs/unknown)	Number of sweeps	Number of tasks on $\mathbb{T}_{h/2}^k$	CPU time in sweeps (sec)	Average grind time (μs/unknown)
0	50	0.0000	23	13800	0.03	0.181	23	55200	0.10	0.174
1	62	0.0506	23	17112	0.03	0.164	23	68448	0.11	0.155
2	74	0.0860	21	18648	0.03	0.166	23	74592	0.14	0.182
3	158	0.0773	19	36024	0.06	0.161	21	144096	0.26	0.183
4	182	0.0844	20	43680	0.07	0.153	20	174720	0.29	0.168
5	230	0.1492	20	55200	0.09	0.159	20	220800	0.42	0.191
6	278	0.1457	20	66720	0.11	0.166	20	266880	0.46	0.171
7	506	0.1323	20	121440	0.19	0.156	20	485760	0.88	0.180
8	716	0.1345	18	154656	0.27	0.177	17	618624	1.10	0.178
9	890	0.1501	17	181560	0.32	0.177	17	726240	1.40	0.193
10	1340	0.1602	17	273360	0.49	0.178	17	1093440	2.29	0.210
11	1796	0.2130	16	344832	0.67	0.194	16	1379328	2.97	0.215
12	2960	0.2068	14	497280	0.96	0.193	14	1989120	4.37	0.219
13	3608	0.2241	14	606144	1.21	0.200	14	2424576	5.30	0.218
14	5042	0.2210	12	726048	1.53	0.211	13	2904192	6.90	0.238
15	6344	0.2554	12	913536	1.97	0.216	12	3654144	8.08	0.221
16	8162	0.2483	11	1077384	2.35	0.218	12	4309536	10.37	0.241

Table IV-IV. Grind time of polynomial order 4.

Cycle ID	Number of active cells	Irregularity index	Number of sweeps	Number of tasks on $\mathbb{T}_h^k$	CPU time in sweeps (sec)	Average grind time (μs/unknown)	Number of sweeps	Number of tasks on $\mathbb{T}_{h/2}^k$	CPU time in sweeps (sec)	Average grind time (μs/unknown)
0	50	0.0000	23	13800	0.09	0.425	21	55200	0.32	0.390
1	56	0.0282	21	14112	0.10	0.454	21	56448	0.37	0.433
2	74	0.0860	21	18648	0.12	0.425	21	74592	0.49	0.438
3	104	0.1085	21	26208	0.17	0.443	21	104832	0.70	0.443
4	194	0.0949	18	41904	0.28	0.452	18	167616	1.13	0.449
5	230	0.1070	18	49680	0.33	0.441	18	198720	1.29	0.433
6	362	0.1268	18	78192	0.51	0.437	18	312768	2.07	0.441
7	458	0.1915	17	93432	0.61	0.436	17	373728	2.56	0.457
8	716	0.2481	17	146064	0.97	0.444	17	584256	4.10	0.467
9	1082	0.2570	17	220728	1.48	0.446	17	882912	6.41	0.484
10	1724	0.2445	15	310320	2.10	0.451	15	1241280	9.02	0.484
11	2156	0.2446	14	362208	2.48	0.456	14	1448832	10.59	0.487
12	2918	0.2741	13	455208	3.23	0.473	13	1820832	13.40	0.491
13	4214	0.2983	12	606816	4.43	0.486	12	2427264	17.98	0.494
14	5702	0.2990	11	752664	5.57	0.493	11	3010656	22.32	0.494
15	7790	0.3037	11	1028280	7.66	0.497	11	4113120	30.41	0.493
16	10322	0.3071	11	1362504	10.34	0.506	11	5450016	41.62	0.509

[53]. The spatial discretization causes the radiation to be distributed to all downwind edges of a cell, leading to numerical dispersion. In this example, a domain of size $[0, 1]^2$ is chosen and an incoming radiation impinges the left face for $0.25 \leq y \leq 0.35$. For the chosen direction, the analytical solution would cause the radiation to leave from the right edge for $0.583505568402405 \leq y \leq 0.683505568402405$. Any amount of radiation leaving from other values of y are due to the numerical spreading of the beam. When the projection-based error estimator is used, α is set 1/3. When the jump-based error indicator is used, $\alpha = 0.2$ for the linear elements and $\alpha = 0.1$ for all other polynomial orders. In Fig. IV-16, we show the relative error in the angular flux as a function of unknowns for the projection-based and jump-based adaptive refinement strategies and uniform refinement. In Fig. IV-17, we show the relative error in the right edge leakage on $0.583505568402405 \leq y \leq 0.683505568402405$ (in %) as a function of unknowns for the three strategies.

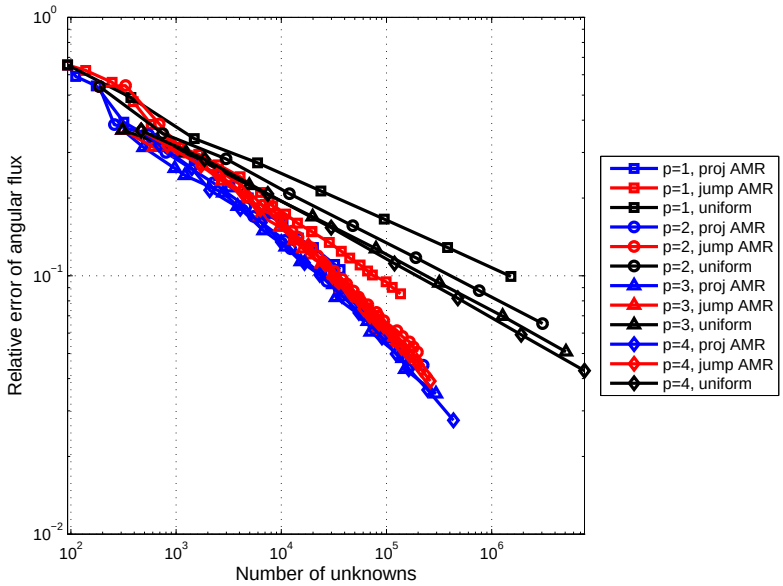


Fig. IV-16. Convergence history of the angular flux, Example 2.

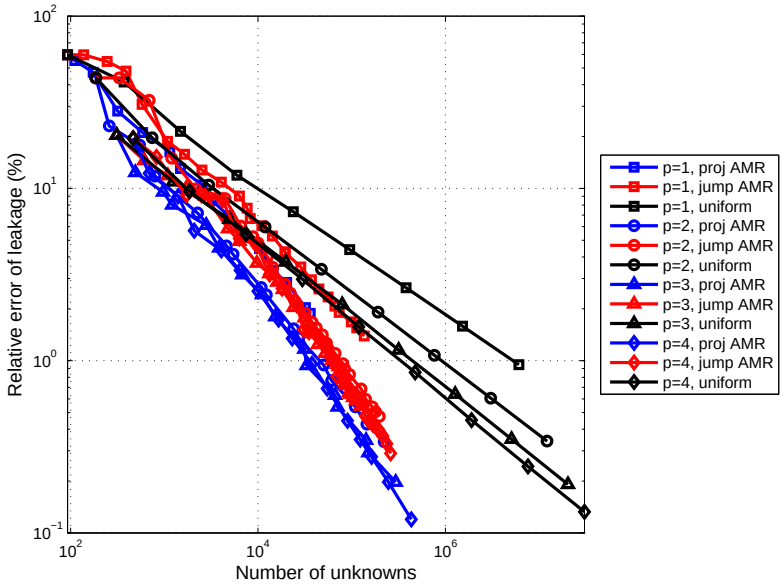


Fig. IV-17. Convergence history of the out-leakage on the right boundary with $0.583505568402405 \leq y \leq 0.683505568402405$.

We can see that both jump-based and projection-based AMR solutions are significantly more accurate than the solution obtained with uniform refinement. The convergence rates of solution employing AMR are significantly enhanced compared to the convergence rates obtained from the uniform refinement procedure. It also needs to be pointed out that the solution computed with quadratic elements is more precise than the solution obtained with linear element while there are not much gain with higher polynomial orders greater than 2.

We then plot the angular flux along the right boundary to see the numerical dispersion in Figs. IV-18 and IV-19 for the projection-based AMR and the uniform refinement with polynomial orders from 1 to 4.

To see how the polynomial order affects the transition at point (1, 0.683505568402405) along the right boundary, we plotted the angular flux around this point in the y-

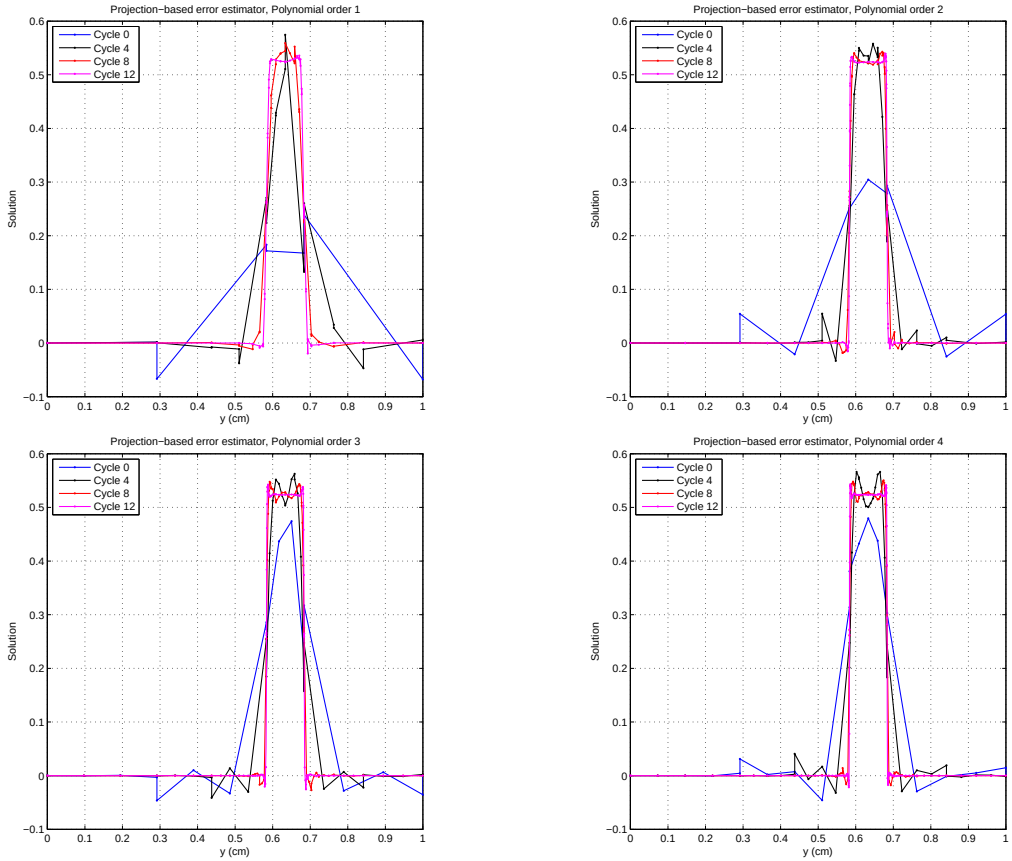


Fig. IV-18. Angular flux on the right boundary with the projection-based AMR.

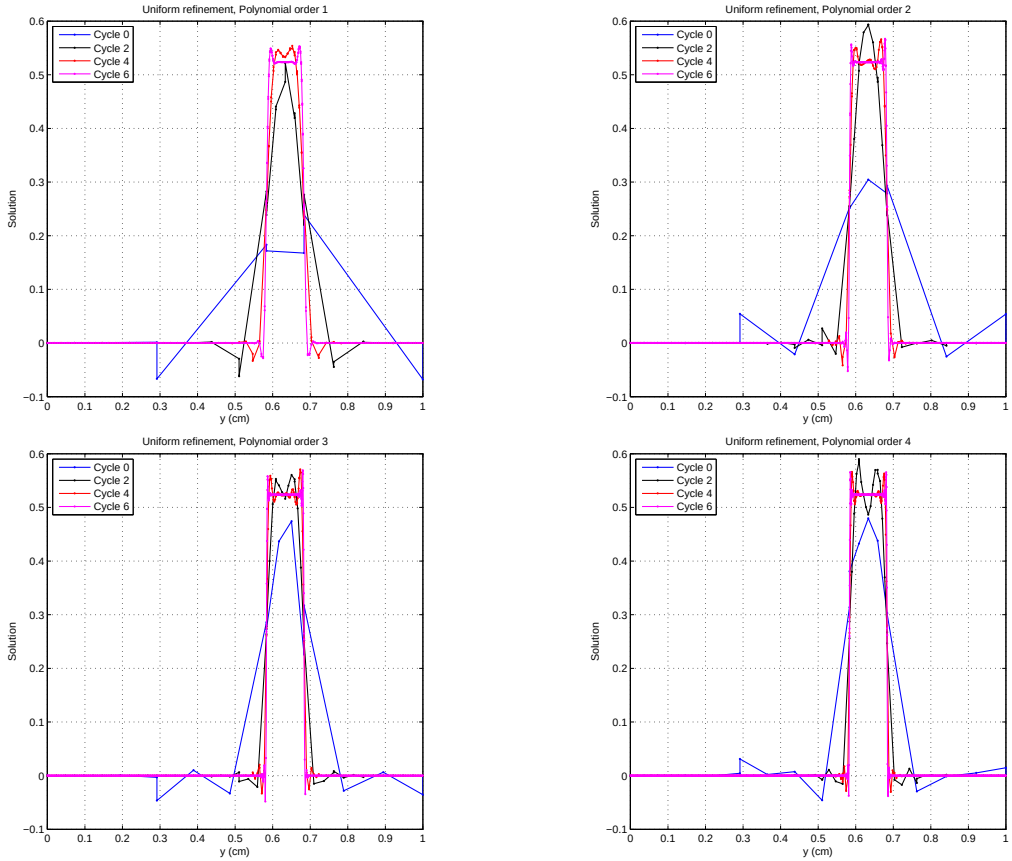
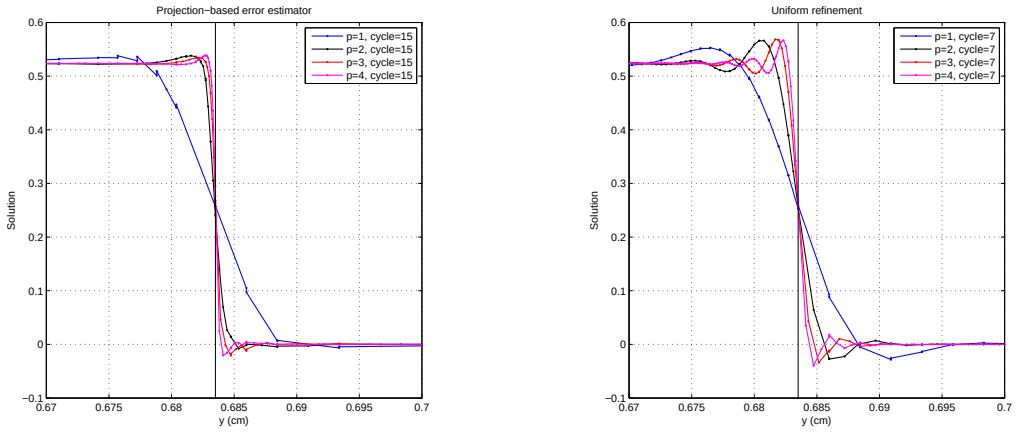


Fig. IV-19. Angular flux on the right boundary with the uniform refinement.

direction with the projection-based AMR and the uniform refinement in Fig. IV-20. Results of cycle 15 for the projection-based AMR and of cycle 7 for the uniform refinement are shown.



(a) h-type AMR with the projection-based error estimator

(b) Uniform refinement

Fig. IV-20. Angular flux on the right boundary around point (1, 0.683505568402405).

We plotted the resulting mesh obtained with the projection-based AMR at cycle 12 (4840 active elements) and the angular flux in Fig. IV-21. For comparison, we also plotted the results with the jump-based AMR at cycle 13 (4750 active elements) in Fig. IV-22. Although both estimations are able to capture the solution discontinuity along the streaming direction, the location of the discontinuity is better resolved by the projection-based error estimator: by looking at Fig. IV-21 more closely, we can notice the double-line feature along the two discontinuity lines. Similar behaviors are also seen for higher polynomial orders.

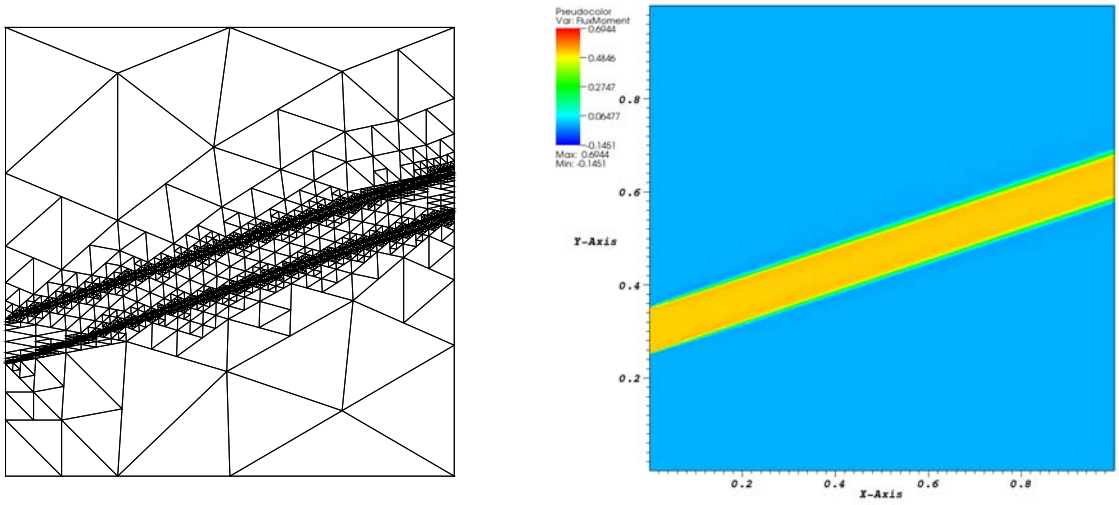


Fig. IV-21. Angular flux with the projection-based error estimator at cycle 12.

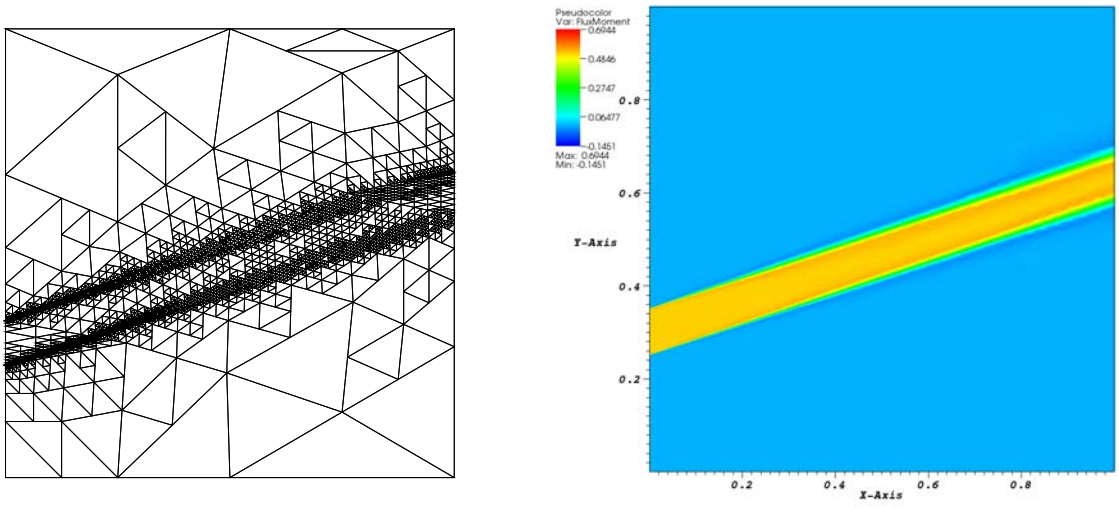


Fig. IV-22. Angular flux with the jump-based error indicator at cycle 13.

3. Example 3: 2-group Eigenproblem

This example is a 2-group eigenvalue problem to show that the AMR methodology proposed in this chapter can also be applied to eigenvalue problems. Geometry is described in Fig. IV-23. Material data are shown in Table IV-V.

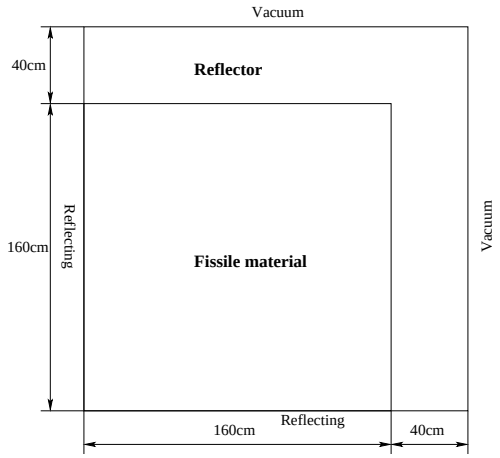


Fig. IV-23. Geometry of the 2-g eigenvalue problem.

Convergence studies are conducted with this eigenvalue problem. Fig. IV-24 shows the convergence history for the two energy groups using different polynomial orders and the projection-based error estimator $\mu_{g,K}^{k,ref}$.

We can note that the solution in each group converges in the same rate. The thermal group requires more degrees of freedom to reach the same accuracy as the fast group. The convergence in k_{eff} with different polynomial orders is plotted in Fig. IV-25, in which the x-coordinates represents the total number of degrees of freedom for the two groups. Adapted meshes at cycle 8 with linear and quadratic elements are given in Fig. IV-26 for the fast and thermal groups.

As we can see, regions at the material discontinuity are more refined by AMR automatically.

Table IV-V. Material properties of the 2-group eigenvalue problem.

		Fissile material		Reflector	
		$g=1$	$g=2$	$g=1$	$g=2$
Total XSs $\sigma_{t,g}$ (cm^{-1})		0.55	1.1	0.561	2.34
Fission XSs $\nu\sigma_{f,g}$ (cm^{-1})		0.005	0.125	-	-
Fission spectrum χ_g		0.99	0.01	-	-
Scattering matrix	$g'=1$	0.52	0.0	0.51	0.0
$\sigma_{s,0}^{g\to g'}$ (cm^{-1})	$g'=2$	0.02	1.0	0.05	2.3

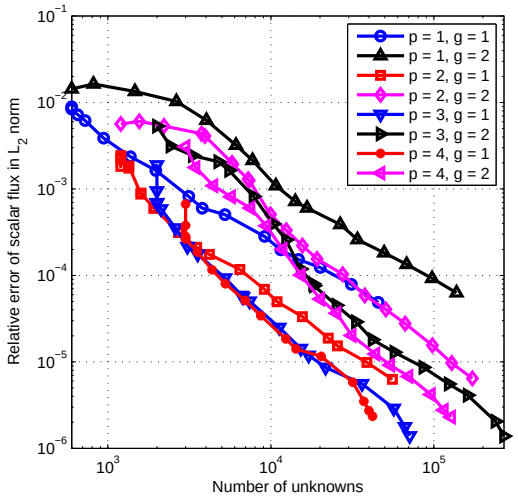


Fig. IV-24. AMR convergence of the 2-g eigenvalue problem.

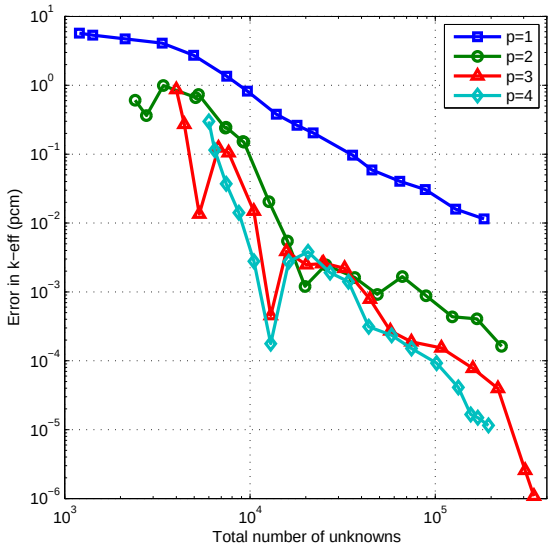


Fig. IV-25. AMR convergence of the 2-group eigenvalue problem with k_{eff}.

4. Example 4: Takeda Benchmark

The Takeda benchmark problem, a 4-group eigenvalue problem, has been described in Chapter II. We solve it here using AMR to demonstrate the multi-mesh calculation and solution singularities in a large domain. Projection-based error estimator and LS-16 angular quadrature are used. Four different meshes are assigned to four energy groups. The convergence histories employing polynomial order 1 to 3 are plotted in Figs. IV-27 and IV-28. The adapted meshes obtained after 15 cycles of refinement for the four energy groups are shown in Fig. IV-29 where a polynomial order 3 is used. The number of active elements for the four groups are 3516, 4347, 6384 and 8544, respectively. The error in k_{eff} for this cycle is 0.0032 pcm. More refinements are observed at the reentering boundary corner. AMR captures the material discontinuities at the annular interfaces of the core.

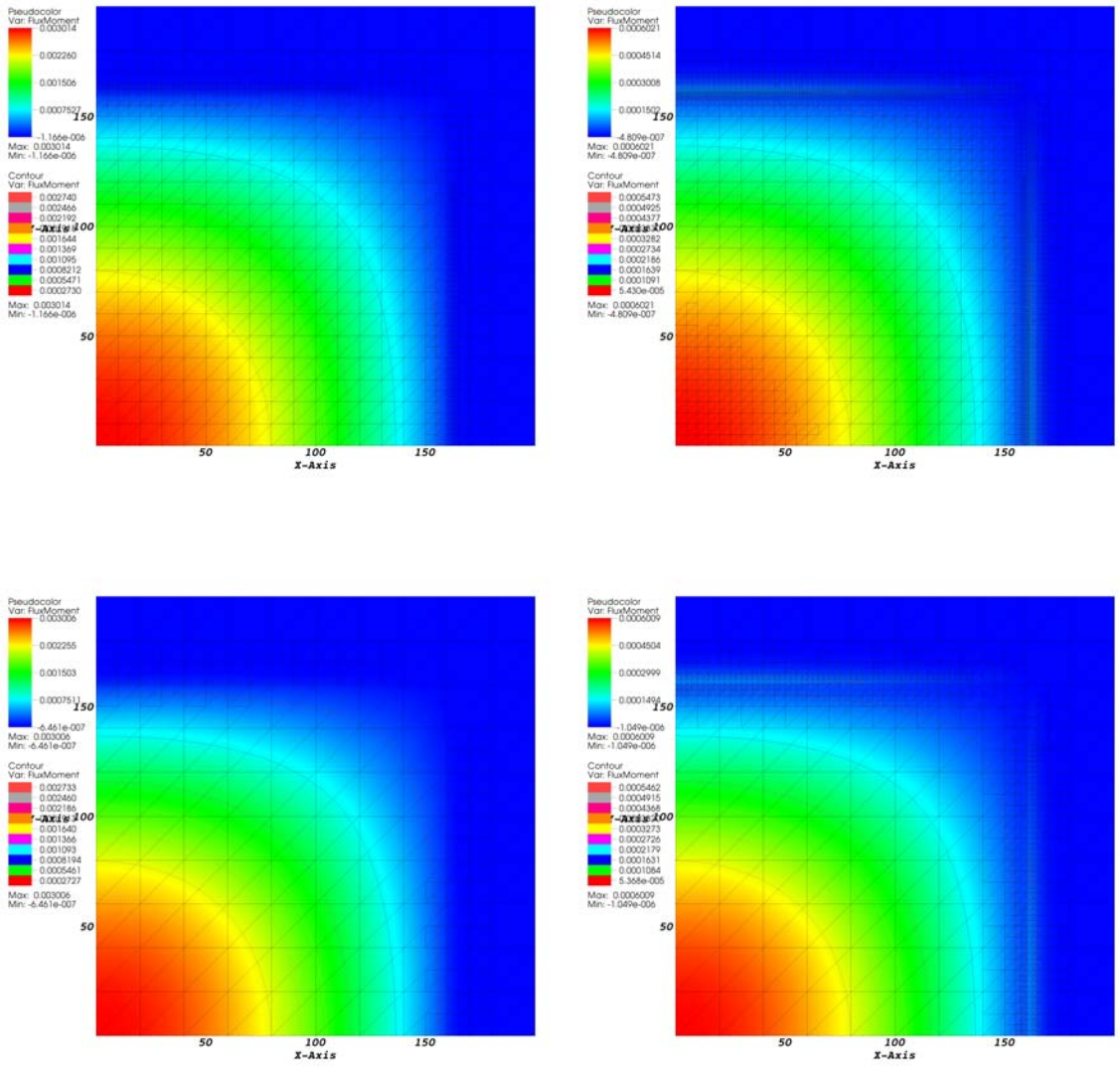


Fig. IV-26. Meshes of the 2-group eigenvalue problem.

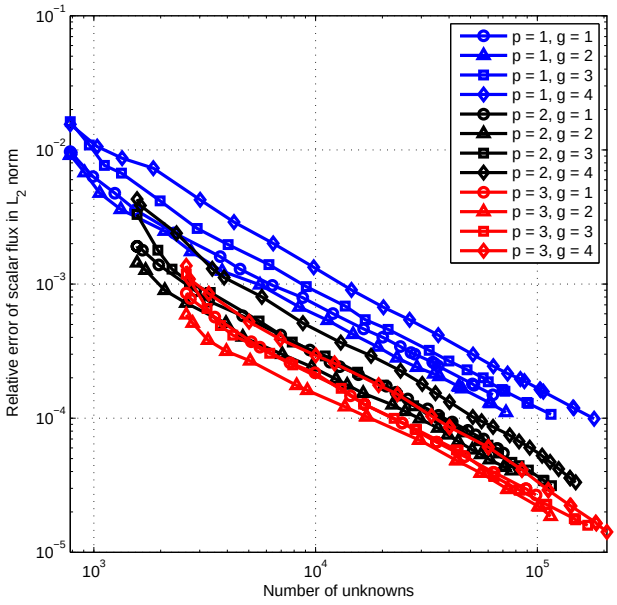


Fig. IV-27. Convergence in the flux for the Takeda benchmark problem.

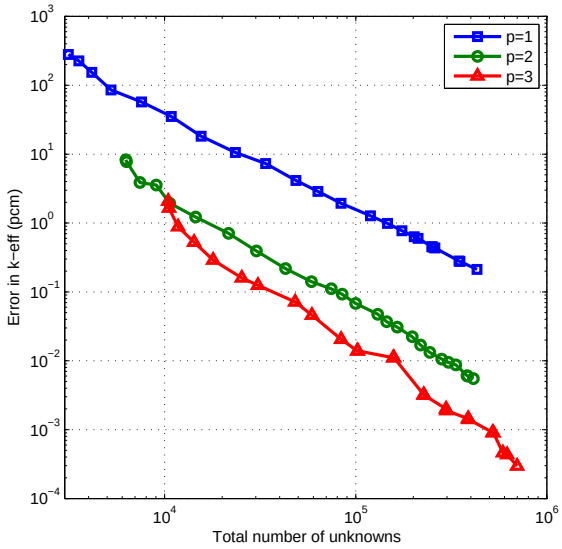
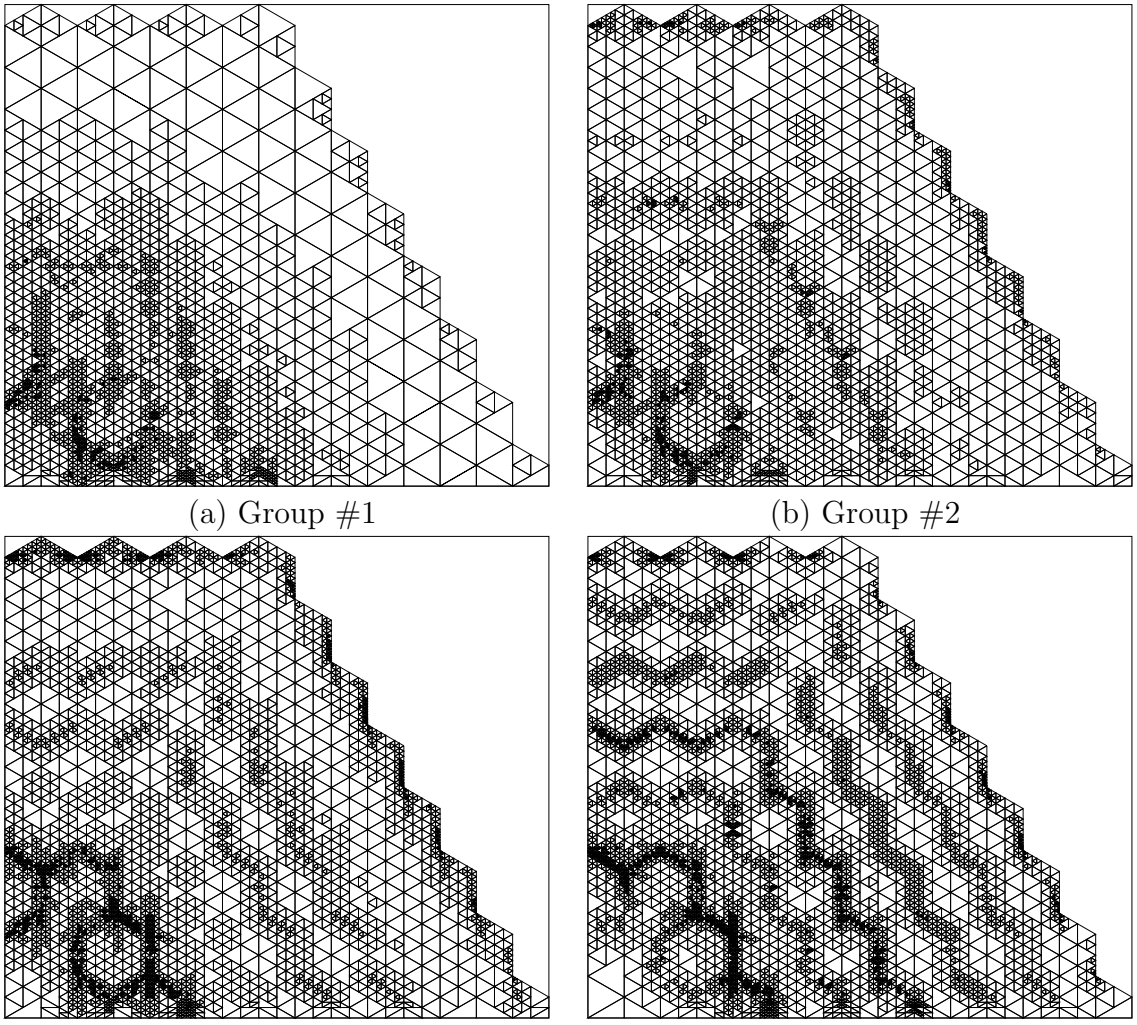


Fig. IV-28. Convergence in k_{eff} for the Takeda benchmark problem.

5. Example 5: 44-group Pin Cell Problem

A fuel cell surrounded with cladding is one of the basic structures of LWR (Light Water Reactor). We arranged fuel cells into an infinite lattice as shown in Fig. IV-30. Then, a geometry consisting of a few fuel elements is chosen for the calculations, in which all four boundaries are set to be reflecting. In Fig. IV-30, we indicate that there are two ways to select such a fuel element geometry: using a 0 and 45 degree rotation with respect to the x-direction.

The 44-group cross section data of the fuel cell problem are obtained with SCALE package. The cross sections for the example 4 in the NEWT manual [72] are used, which are generated with the T-XSEC sequence of TRITON [71]. 22 groups are in the thermal range. Maximum anisotropic scattering order N_a is set to 2 in this calculations. A LS-4 quadrature is used in this calculation to make it clearer (fewer singularities) that the singularities of the S_N solution are also present in eigenvalue



(a) Group #1 (b) Group #2

(c) Group #3 (d) Group #4

Fig. IV-29. Adapted meshes of the four energy groups, after 15 cycles of mesh adaptation.

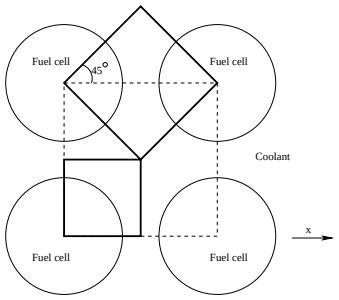
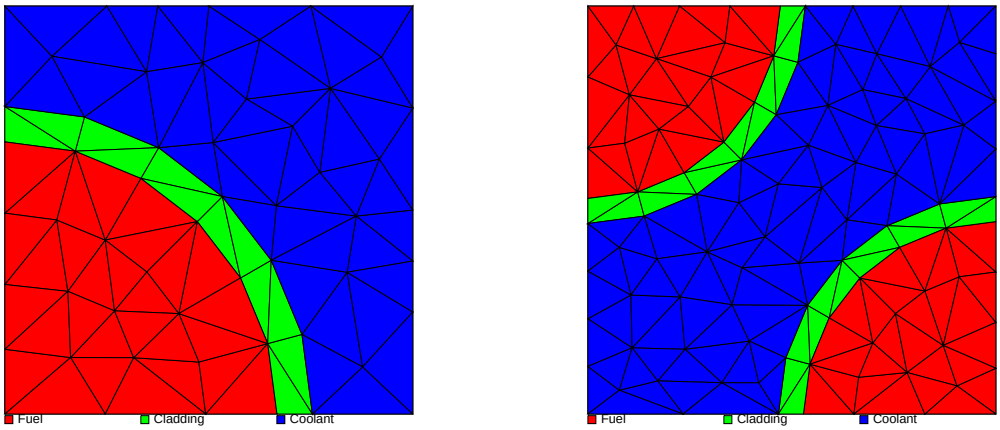


Fig. IV-30. Fuel lattice.

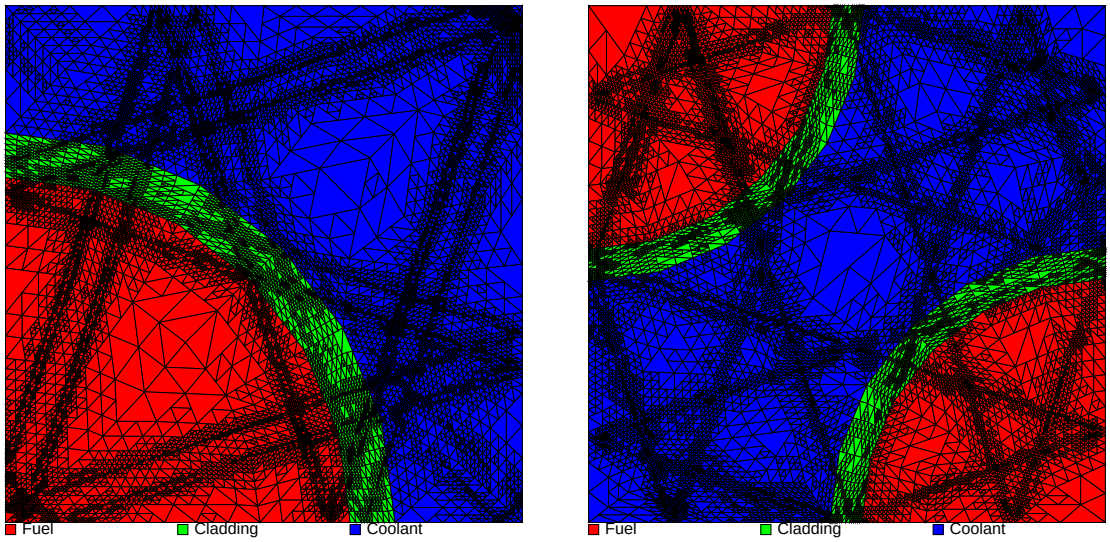
problems. In real-world reactor analysis, a higher quadrature (such as the product quadrature with many more sweeping directions in the azimuthal direction) should be employed. Calculations with AMR are conducted with the two fuel elements. The initial meshes are shown in Fig. IV-31. After 40 cycles, we obtain the two different adapted meshes given in Fig. IV-32.



(a) 0° rotation (b) 45° rotation

Fig. IV-31. Initial meshes for the 44-group pin problem.

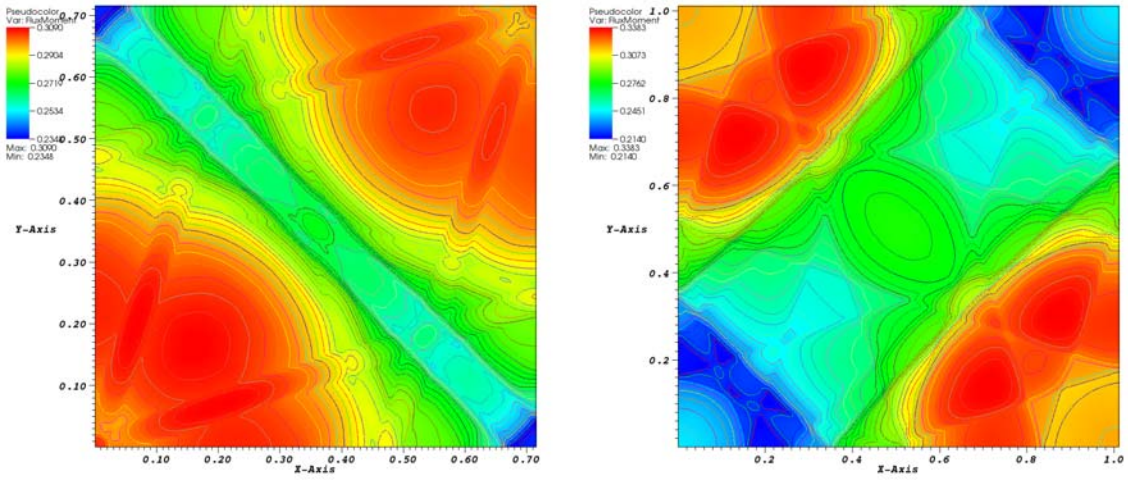
Singularity lines are observed in both calculations. the solutions of the continuous transport equation should be exactly the same for these two fuel elements. However, although we can control the spatial discretization error, the numerical solutions are



(a) 0° rotation (b) 45° rotation

Fig. IV-32. Adapted meshes for the 44-group pin problem.

different due to the angular discretization error. For example, the scalar flux in the first group (i.e., with the highest neutron energy) is very different as shown in Fig. IV-33. The singularities in the thermal groups are not as strong as the ones of the fast groups. The k_{eff} values for two calculations are 1.18874 and 1.19652, or about 800pcm difference. Such solutions are clearly unacceptable due to the poor choice of the angular quadrature, which was later verified by employing a higher-order product quadrature set.



(a) 0° rotation (b) 45° rotation

Fig. IV-33. Scalar fluxes of the first group of two fuel elements.

D. Conclusions

In this chapter, we proposed two error estimations to drive the AMR applied to the S_N transport equation: the projection-based and the jump-based error estimations. Tests were performed using both source problems and eigenvalue problems. In conclusion, we find that:

1. A cell-based AMR is feasible for higher-order DGFEM;
2. AMR can significantly reduce the number of unknowns and the CPU time requires to obtain an accurate solution. Notably, it pays to use, at least, quadratic basis functions;
3. Jump-based error estimation is cheaper and good for a variety of problems;
4. Singularities in the transport solutions can be captured with both error estimations;

5. Although we can align the mesh with the solution singularities due to the S_N quadrature once an angular quadrature has been chosen, such an alignment may not important because whenever AMR improves the solution significantly, the angular discretization error become dominant in the numerical solution.

6. We were able to reduce significantly the spatial error, to the point where the angular error (ray effects) were dominant. The next logical step would be to address the question adaptivity in both space and angle.

CHAPTER V

XUTHUS, A 2-D AMR TRANSPORT SOLVER

XUTHUS is the 2-dimensional multigroup S_N-transport solver with Adaptive Mesh Refinement (AMR) developed in Texas A&M University (TAMU) as a part this Ph.D. dissertation. The main feature of XUTHUS is that the spatial discretization error is controlled using AMR the first time for the multigroup S_N-transport calculations, including acceleration solvers. Guaranteed numerical solutions, with error below a prescribed tolerance, are provided in an efficient manner through AMR. The spatial discretization scheme employed in XUTHUS, the *hp*-version of the Discontinuous Galerkin Finite Element Method (DGFEM) on unstructured triangular meshes, allows for a flexible and non-uniform distribution of the degrees of freedom throughout the computational domain. Group-dependent adapted meshes (referred to as "multi-mesh"), with arbitrary refinement level differences between elements (aka "multi-irregularity") and non-homogeneous polynomial order are possible with *hp*-type meshes. Since the adapted mesh delivered with the AMR procedure can follow the physics closely, the computing effort can be significantly reduced. In addition, the reduction in memory needs with AMR enables the accurate solution of problems that were once impossible with a uniform mesh because of the prohibitive computational effort of the uniform refinement, both in CPU and memory. Furthermore, a stable conforming Diffusion Synthetic Acceleration (DSA) scheme makes XUTHUS effective for a wide range of highly-diffusive problems. A parallelization with spatial domain decomposition has been implemented with MPI (Message Passing Interface) and allows XUTHUS to take the advantage of the development of supercomputer architectures in order to handle extremely large problems.

In collaboration with the Nuclear Science and Technology Division (NSTD) of the Oak Ridge National Laboratory (ORNL), XUTHUS may ultimately be distributed as a released module within the SCALE (Standardized Computer Analysis for Licensing Evaluation) code system, providing users with an alternative to the current transport solver NEWT (New Extended-step-characteristic-based Weighting Transport code). All of the cross section processing options of SCALE are seamlessly integrated into a XUTHUS-based sequence. The powerful yet easy-to-use geometry description in SCALE, together with the Triangle mesh generator provides XUTHUS with the capability to solve problems containing sophisticated geometries.

XUTHUS can be used for both source- and eigenvalue-problems and can function as both a forward and an adjoint solvers. XUTHUS can potentially be applied beyond the traditional nuclear fuel assembly calculations, e.g., for shielding or inverse calculations, due to all of these features. In short, XUTHUS's characteristics are:

- Fortran 90 Language;
- Unstructured triangular meshes;
- *hp*-type DGFEM with polynomial order up to 4;
- Multi-mesh, multi-irregularity and non-homogeneous polynomial order;
- *h*-type AMR driven by either projection-based or jump-based error estimators;
- Standalone DG-diffusion calculation;
- Conforming stable DSA;
- Krylov solver and SI (Source Iteration);
- "Integratable" into SCALE;
- Domain decomposition with MPI, with synchronous or asynchronous transport sweeps.

A. Development History of XUTHUS

Development of XUTHUS can be roughly divided into four stages.

Initial stage, October 2006 to June 2007. Upon completion of a M.S. thesis [142] dealing with AMR applied to the multigroup diffusion equations, the following conclusions were drawn:

1. Exponential spatial convergence rate can be obtained for the multigroup diffusion equations using p-type or hp-type mesh adaptation, while h-type adaptivity alone only yields algebraic convergence rates.

2. Projection-based error estimator can be used to drive AMR without the need of a finely resolved initial mesh.

3. Using different meshes for different energy groups can further reduce the number of unknowns in the multigroup scenario. Mesh coupling in dependent group-dependent meshes can be done either either using an adaptive integral technique or by visiting the tree structure of the refinement history.

The experience gained by programming of hp-type mesh refinement with continuous FEM was successful and it was proposed to apply the hp-version AMR to the multigroup S_N-transport equation, which is a better mathematical model to describe particle transport. The model error is greatly reduced compared to suing a diffusion model, which was one of the major critique of performing AMR for a diffusion model of particle transport. As explained in Chapter I, we have not considered angular adaptation nor the accuracy of the multigroup approximation in the present work, although we recognize that discretization errors are also associated with both energy and angle variables.

First summer at ORNL, June 2007 to August 2007. Both Dr. Ragusa and I were at ORNL during this period; Fortran 90 was chosen as the programming language. We also decided that the new code must support MPI because problem sizes were becoming larger for Hi-Fi (High-Fidelity) calculations. We planned to write a reusable production code but not a toy code in this Ph.D. research, which meant that a significant amount of effort was targeted at code quality (use of modern version-control tools, documentation, verification). In this stage, the frame of the code was formed: data structures for multiple unstructured meshes, MPI, transport-sweep ordering, *hp*-type mesh refinement capability, iterative solver. Extensive discussions on the design of modules were conducted and module initializers were rewrote several times to make sure their logic were clearly defined. Common interests are identified through talks with people at NSTD. Before we left ORNL, we had named the code XUTHUS and it was running with unstructured regular meshes, i.e., without *hp*-refinement and without accelerations. The grind time of the transport sweep was in a good range compared with equivalent codes. We were facing with new challenges: how to deal with mesh coupling effectively both for multi-mesh and multi-irregularity, how would the performance of higher order elements affect runtime? How does AMR work with S_N-transport? And MPI with adaptive mesh refinement is far from trivial.

Development stage, August 2007 to December 2007. Mesh coupling algorithms were mastered and coded in a matrix-free scheme. Multi-mesh and multi-irregularity were implemented without much performance loss due to the fact that fetching data from memory is the bottle neck of modern computing but not floating-point operations in the CPU. Also, for the same reason, the grind time per unknown remained almost constant for different polynomial orders. The grind time of $p = 2$ was even smaller than linear grind time because of the cache capacity. We noticed

that higher-order calculation were still effective for calculations where boundary-layer effect was significant. Full *hp*-mesh capability were coded. We were thrilled that the S_N singularities were isolated with the *h*-type AMR driven by the projection-based error estimator. Also, MPI was implemented for AMR.

Improvement period, end of 2007 to present. We were facing a fork in development options: one option was to continue the development of AMR, including full implementation of *hp*-type and goal-oriented refinements; the other was to enhance the solver by including Krylov solvers, eigensolvers and acceleration techniques such as DSA. We decided to go in the second direction. The DGFEM for the diffusion turned out to be much more complicated to implement than for the transport. We first implemented the well-known IP form for elliptic problems and used it to perform DSA. We noticed immediately that this scheme was not stable for large cell sizes. By keeping the IP penalty below 0.25, the scheme remained stable. To understand this unexpected phenomenon, we tried several the variational derivations and a new conforming DSA scheme was proposed. It turned out that the modified IP form is a stabilization scheme of this conforming scheme, which also suffered instabilities in the intermediate mean-free-path range . We also develop a means of dealing with the significant angular fluxes and anisotropic scattering within these new DSA schemes. GMRes for one-group transport problem is implemented with an open-source software package. Preconditioned CG with the Eisenstat "trick" was also coded for the DG-diffusion calculation with the modified IP form. For all new spatial schemes, we made sure that AMR and MPI were working properly. SQMR solver for the DG-diffusion was employed for the diffusion conforming scheme, which is symmetric but not positive definite. Integration of XUTHUS within SCALE was considered during the summer 2008. The jump-based error estimator was implemented, while testing

the convergence properties of DG-FEM. Chebyshev acceleration for the power iteration was implemented and the ARPACK solver was added. Meanwhile, Dr. Ragusa and several students added several angular quadratures, which can be used for highly forward peaked scattering calculations.

Now, XUTHUS is in a stable version. Future developments may include:

- Finalize the integration of XUTHUS into SCALE ;
- Add METIS to perform domain decomposition on unstructured meshes;
- Perform a load rebalance after mesh adaptation;
- Add more post-processing options so that XUTHUS can be coupled with other modules in the TRITON sequence to perform depletion calculations;
- Implement an unstructured Coarse-Mesh Finite Difference (CMFD) solver within XUTHUS to accelerate power iterations;
- Implement full *hp*-adaptivity;
- Implement goal-oriented calculations;
- Optimize memory management in transport sweeps;
- Add P_1 conforming DSA;
- Optimize the design of shape functions and implementation of polynomial for orders greater than 4;
- Make the solver and AMR dimension-independent;
- Add cycle detection in the transport sweeps.

B. Implementation Details

1. Data Structure for the Unstructured Mesh

The spatial coordinates of all vertices are needed to locate all elements covering the domain. It is not suggested to describe the coordinates of vertices separately for all elements. If, for example, there are on average five elements connecting to a vertex, then four times the memory would be needed to store the same coordinates. So *a vertex array* is used, where each entry in the array containing the coordinates of a vertex and the entry index will is as the vertex ID. We can access the vertex coordinates through a vertex ID.

All local operations with DGFEM are associated with elements, so *an element array* is naturally needed to describe all elements. IDs of the three counter clock-wise numbered vertices must given for each element. The vertex array and the element array with vertex IDs are the minimum data required to describe a geometry. However, for simpler coding and better run-time performance, redundant geometrical data is also stored. e.g., the three neighboring element IDs of a given element are needed to easily access the neighbors of a element or to use them to describe the type of boundary condition when a element lies on the boundary.

We also maintained *an edge array* (in 2-D, this would be a face array). Our argument to have this redundant array is that edges form the domain boundary and the sub-domain interfaces, where significant angular fluxes are required in transport sweeps and for inter-processor communications. In addition, regional particle balance (in- and out-leakage) needs also to be evaluated along a region's edges. Our basic principle for this edge array is to setup the bi-directional connections between elements and edges, i.e., add an entry in the element type containing all of its edge IDs and add two entries in the edge type, which are the edge's left and right elements. We

also have two more entries for the edge type to locate the edge in space: the starting and ending vertex ID. Note that an edge has its own orientation defined by its two vertices; this orientation may not be the same as the one defined by the element local numbering. So, in the element type, we need have one more entry in general to state how the edge is oriented in that element. In 2-D, we can simply add a minus sign on the edge IDs if the two orientations are opposite. It is also convenient that add two other entries for the edge type, which give the local edge ID in its left and right elements. If a left or right element does not exist, i.e., the edge is a boundary edge, we can use the element ID as the boundary type or the virtually connected element ID in the case of periodic boundaries.

For h-type AMR, we need to add more entries in the element type to describe *the hierarchical mesh refinement structure*. To be able to visit the refinement tree structure from top to down, we need an entry containing the IDs of the four child elements. The entry is set to zero if the element is not refined. On the other hand, we need an entry containing its parent's ID to visit the tree structure from the bottom up. If an element is part of the initial mesh, this ID is zero. So far, these two entries are sufficient for the purpose of h-refinement calculations. However, some additional information has been stored to minimize the coding efforts and maximize the run-time performance. The rank of the element among its siblings and its refinement level in the tree have been added. An element is active when it is not further refined, i.e., all its child elements are zero. An active element has another ID, which we called active element ID, from the natural ordering we described in Chapter IV. So we need an entry for all elements to store this active element ID. If an element is not active, this entry is zero. To be able to access the element array through the active element ID, we need a *mapping array*, whose entries are simply the elements' ID numbers and whose length is the number of active elements, in order to map the active ID to the

element array ID.

When performing AMR, we need to temporarily increase the size of all arrays. To manage empty entries, we create *chain lists* for each array. Once a new entry needs to be added, the header of the corresponding chain list is accessed. Upon completion of one adaptivity iteration, all the arrays are resized to minimize the memory requirements. Note that because all these operations are not within the iterative solver, they are not critical for runtime efficiency.

All of the above arrays form one mesh structure. In multigroup calculations, there are mesh ID numbers for each group. The number of group-dependent meshes does not need to be equal to the number of energy groups, but the user can control which energy groups employ the same adapted mesh. When several meshes are used, we simply keep as many (*number-of-mesh copies*) adapted meshes as required.

It is also helpful to setup at the beginning of a run the following data: a *list of initial elements*, a *list of boundary edges* and *lists of edges on interfaces between subdomains*. This information is useful for projecting solutions from one mesh to another and for the initialization of computing modules. Note that the edges on the subdomains' interfaces need to be ordered properly to assure consistency among processors.

2. Transport Sweep Ordering

Each combination of an element and a streaming direction is labeled as a *task*. We define the *incoming degree* of a task as the number of upwind elements, i.e., the number of dependent tasks. The *incoming degree* is always less than the number of sides (edges) of an. When an edge is *parallel* or almost parallel with a streaming direction, the neighboring element on its other side is neither upwind nor downwind. Numerically we set a small real number EPS to determine this as follows: when the

Table V-I. Subroutines to form the topological relations in between all transport tasks.

Get_Incoming_Degree(g,itask)	Get the incoming degree of a task in energy group g
Get_UpWind_Tasks(g,itask)	Obtain a list of all upwind tasks for a task in group g
Get_DownWind_Tasks(g,itask)	Obtain a task list of all downwind tasks of a task of group g
Modify_DownWind_Degree(g,itask)	Modify incoming degrees of all downwind tasks
Modify_Edge_DownWind_Degree(g,edg,dir)	Modify incoming degrees of all downwind tasks of a single edge wrt the direction

inner product of the streaming vector $\vec{\Omega}_m$ and the unit outward norm vector of a side $\vec{n}_i$ is too small, i.e.,

$$\vec{\Omega}_m \cdot \vec{n}_i < EPS \tag{5.1}$$

this side is considered as being parallel to the streaming direction. In the above equation, i is the local side ID of the element. In our implementation, EPS is 10^{-8}. All tasks are managed through a *task list* or list of tasks. The total number of tasks for a given mesh is the number of active elements times the number of streaming directions. Each entry in the task list provides an active element ID, a streaming direction ID and the incoming degree of a task. At the initialization stage, this task list is not ordered. A few subroutines in Table V-I are used to setup the topological relations in between all tasks, which are collected in one Fortran 90 module.

Note that the number of upwind tasks of a task which is on the downwind side of a subdomain interface (wrt the streaming direction) depends on how we perform communications in the paralle setup. If we always start sweeping from the subdomain interface (asynchronous mode), the incoming degree of these tasks need to be

decreased in the subroutine.

With these subroutines, we can order the task list using a chain list, where all entries are accessed with the header pointer *hp*. The algorithm is presented in Algorithm 1.

Algorithm 1 Transport sweep ordering.

1: Insert IDs of all tasks whose incoming degrees are zero into the chain list one by one with a specific priority rule ‡.
2: Initial number of tasks to be solved: $itask \leftarrow 0$.
3: Allocate memory for the new ordered task list.
4: **while** the chain list is not empty **do**
5: Get a task ID $task\_id$ with the pointer hp from the chain list until the list is empty.
6: $itask \leftarrow itask + 1$
7: Put the task with $task\_id$ into the ordered task list.
8: Call Modify_DownWind_Degree(g,$task\_id$) to decrement the incoming degrees of all downwind tasks by 1.
9: Call Get_DownWind_Tasks(g,$task\_id$) to obtain all downwind task $dtask(j), j = 1, \cdots, ndtask$.
10: **for** $j = 1$ to $ndtask$ **do**
11: Call Get_Incoming_Degree(g,$dtask(j)$) to check if its incoming degree is zero or not.
12: **if** the incoming degree is zero **then**
13: Insert $dtask(j)$ into the chain list with the specific priority rule ‡.
14: **end if**
15: **end for**

16: **end while**

17: Check *itask*, see if it is equal to the number of total tasks *ntask*.

18: Deallocate memory for the old task list.

‡ Note that different *priority rules* can be applied. We did not try to optimize the priority rule to minimize memory cost. Theoretically, memory needs to be allocated solely for *the sweep front*, which is significantly smaller than the memory cost for storing angular fluxes for all tasks. As of now, the best we have is that setting the priority with the ID of the streaming direction, and only keeping the angular fluxes for one direction of all active elements.

Although we do not need to order all tasks if we directly use the above algorithm to perform a transport sweep, it is still preferred to do so for of cache optimization.

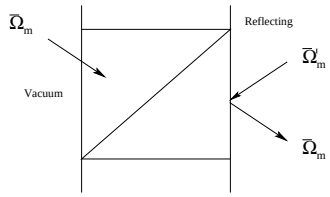


Fig. V-1. Significant angular flux update.

The Significant Angular Fluxes (SAF) are stored in another Fortran 90 module. We have two options on how to utilize the SAF. The first manner is as follows: before performing a transport sweep, the SAF values are copied to a "work" memory location and during the sweep, the SAF values are taken from and updated to these memory slots (the SAF may be updated during the sweep as shown in Fig. V-1). By doing so, we are in fact applying a Gauss-Seidel scheme for the SAF. The second manner is that we always directly take the SAF values from the Fortran module where they are stored. After a transport sweep, the SAF from the working array is copied to

the SAF module (Jacobi iterative approach). Communication could be done at this time if the sweeps are performed synchronously across subdomains' interfaces. It is important for the DSA algorithm that we employ the second option in the sweeps. More details on the parallelization are presented in the next section.

3. Spatial Domain Decomposition with MPI

The only parallel technique available in XUTHUS so far is a spatial domain decomposition, where communication across subdomains are performed using MPI. The entire spatial domain is divided into *subdomains* with the number of subdomains is equal to the number of processors *np*. The domain partition is done at the user-level as of now and, in the future, should be automatically performed by calling a domain decomposition package such as METIS [143]. A entry *domain_id* in the element type must present for all elements to differentiate between an element on a subdomain and a *ghost element*; a ghost element is an element belonging to other subdomains and attached to the current subdomain through an edge on the subdomain's interfaces. Note that not all the children of an initial ghost element are ghost elements because, at least, the central (fourth) child element is not connected to any interfaces. Like active elements, there are also *active ghost elements* which are assigned an active ID sequentially after the standard active elements.

Processors need to communicate the outgoing angular fluxes on their subdomain interfaces for transport calculation. Sweeping through a subdomain requires knowledge of the incoming angular fluxes (provided by the active ghost elements) on the interfaces; these flux values are obviously outgoing angular fluxes from other subdomains. There are essentially two different ways in XUTHUS for setting up the communications in between subdomains: asynchronously and synchronously.

In the asynchronous communication mode, we did not modify the sweeping order;

an interface task and all its downwind tasks in a given subdomain can not be initiated until its upwind task located in other subdomains are solved and the angular fluxes of these tasks are obtained through communication. Obviously, the sequential feature of the asynchronous transport sweep and the parallelization needs are in conflict, especially for unstructured meshes; refer to [144] for additional discussions. We did not optimize our implementation but simply used the sweeping algorithm presented in the previous section and modified it because of communications requirements. Note that because the communications are started with tasks and occur during a sweep, the solution sequence is not known before run-time; we cannot pre-order all tasks and must update dynamically the incoming degrees before performing a given task.

Although only the angular fluxes along the interface edges are needed, it is convenient to communicate the entire angular flux vector of the sending element. We need to obtain a tag or sending port with the edge ID and the direction through the proper ordering of the interfaces edges. Only when the orderings of any pair of two adjacent processors for their common edges are the same can the two processors be allowed to communicate.

Algorithm 2 Transport sweep with asynchronous communication.

1: Start listening of all receiving ports (MPI functionality).
2: Insert IDs of all tasks whose incoming degrees are zero into the chain list one by one with a specific priority rule ‡.
3: Initial number of tasks to be solved: $itask \leftarrow 0$.
4: **while** $itask$ is not equal to the total number of tasks $ntask$ on the local processor **do**
5: **while** the chain list is not empty **do**
6: Get a task ID $task\_id$ with the pointer hp from the chain list until the list is

empty.

7: $itask \leftarrow itask + 1$

8: Put the task with $task\_id$ into the ordered task list.

9: Call Modify_DownWind_Degree(g,$task\_id$) to decrement the incoming degrees of all downwind tasks by 1. Check all downwind tasks one by one, if it is in another domain, a non-blocking communication with the solution of the current task is started (MPI functionality).

10: Call Get_DownWind_Tasks(g,$task\_id$) to obtain all downwind task $dtask(j), j = 1, \cdots, ndtask$ in the current subdomain.

11: **for** $j = 1$ to $ndtask$ **do**

12: Call Get_Incoming_Degree(g,$dtask(j)$) to check if its incoming degree is zero or not.

13: **if** the incoming degree is zero **then**

14: Insert $dtask(j)$ into the chain list with the specific priority rule ‡.

15: **end if**

16: **end for**

17: **end while**

18: Test all left receiving ports (MPI functionality).

19: **for all** receiving ports just cleared **do**

20: Get the interface edge and call Modify_Edge_DownWind_Degree to decrement the incoming degrees of all its downwind tasks by 1. This subroutine will also give a downwind task list $dtask(j), j = 1, \cdots, ndtask$.

21: **for** $j = 1$ to $ndtask$ **do**

22: Call Get_Incoming_Degree(g,$dtask(j)$) to check if its incoming degree is zero or not.

23: **if** the incoming degree is zero **then**

24: Insert $dtask(j)$ into the chain list with the specific priority rule ‡.
25: **end if**
26: **end for**
27: **end for**
28: **end while**

‡: same remark as in the previous algorithm.

The communication mode is a synchronous communication, where sweep are broken on the subdomains' interfaces and each subdomain starts their portion of the sweep at the same time (parallel block Jacobi). Ghost angular fluxes from the previous source iteration or from the prolongation of a previous AMR solution are used. Communications are done synchronously, after all processors finish all their sweeping tasks. All processors copy their outgoing angular fluxes on the interfaces from the working memory and send them to their neighboring processors.

These two communication modes are radically different. If the problem is mostly absorbing (i.e., low scattering cross section), there is no point in doing synchronous sweeps as the transport effect dominates the solutions are the flow of information (upwind to downwind) is important. Such a situation is better handled with asynchronous communications.On the contrary, if scattering is strong, diffusion processed tend to dominate and there are no preferred flows of information, favoring a synchronous communication mode. It has also been proposed that the uncollided flux be solved with asynchronous communications (pure absorber situation) and subsequently use it to construct the first-collision source and finish solving the problem with synchronous communication. We have not yet added this automatic switch in XUTHUS.

We now present the DSA acceleration when performed with the *synchronous*

communication mode. The transport bilinear form is

$$b(\Psi,\Psi^*)=\sum_{m=1}^{M} w_m\big((\vec{\Omega}_m\cdot\vec{\nabla}+\sigma_t)\Psi_m,\Psi_m^*\big)_{\mathcal{D}}+ \sum_{m=1}^{M} w_m\left\langle[\![\Psi_m]\!],\Psi_m^{*+}\right\rangle_{E_h^i}+\sum_{e\in\partial\mathcal{D}}\sum_{\vec{\Omega}_m\cdot\vec{n}_b<0} w_m\left\langle\Psi_m,\Psi_m^*\right\rangle_e \tag{5.2}$$

Let us define the interface between subdomain i and subdomain j as $\mathbb{F}_{i,j}, i=1,...,np; j=1,...,np$, where np is the number of processors, i.e., the number of subdomains. All $\mathbb{F}_{i,i}, i=1,...,np$ are null sets. If two subdomains i and j are not adjacent with common edges, their interface $\mathbb{F}_{i,j}$ is also a null set. Denote the set of all interfaces as $\mathbb{F}=\cup_{i=1}^{np}\cup_{j=1}^{np}\mathbb{F}_{i,j}$ and denote the set of all the interfaces for a given subdomain i as $\mathbb{F}_i=\cup_{j=1}^{np}\mathbb{F}_{i,j}$. Let us also re-define the set of interior edges as $E_h^i=E_h^i\backslash\mathbb{F}$ (i.e., excluding edges belonging the subdomain interfaces). The bilinear form can then be re-written as,

$$\begin{aligned} b(\Psi,\Psi^*)=&\sum_{m=1}^{M} w_m\big((\vec{\Omega}_m\cdot\vec{\nabla}+\sigma_t)\Psi_m,\Psi_m^*\big)_{\mathcal{D}}+ \\ &\sum_{m=1}^{M} w_m\left\langle[\![\Psi_m]\!],\Psi_m^{*+}\right\rangle_{E_h^i}+\sum_{e\in\partial\mathcal{D}}\sum_{\vec{\Omega}_m\cdot\vec{n}_b<0} w_m\left\langle\Psi_m,\Psi_m^*\right\rangle_e+ \\ &\sum_{i=1}^{np}\left[\sum_{e\in\mathbb{F}_i}\sum_{\vec{\Omega}_m\cdot\vec{n}^i<0} w_m\left\langle\Psi_m^+,\Psi_m^{*+}\right\rangle_e-\sum_{e\in\mathbb{F}_i}\sum_{\vec{\Omega}_m\cdot\vec{n}^i<0} w_m\left\langle\Psi_m^-,\Psi_m^{*+}\right\rangle_e\right], \end{aligned} \tag{5.3}$$

where the $\vec{n}^i$ is the outward normal unit vector on $\mathbb{F}_i$ with respect to subdomain i.

We can then define the bilinear form with domain decomposition (DD) as follows

$$
\begin{aligned}
b_{DD}(\Psi,\Psi^*) = & \sum_{m=1}^{M} w_m \big((\vec{\Omega}_m\cdot\vec{\nabla}+\sigma_t)\Psi_m,\Psi_m^*\big)_{\mathcal{D}} + \\
& \sum_{m=1}^{M} w_m \left\langle [\![\Psi_m]\!],\Psi_m^{*+}\right\rangle_{E_h^i} + \sum_{e\in\partial\mathcal{D}}\sum_{\vec{\Omega}_m\cdot\vec{n}_b<0} w_m \left\langle \Psi_m,\Psi_m^*\right\rangle_e + \\
& \sum_{i=1}^{np}\sum_{e\in\mathbb{F}_i}\sum_{\vec{\Omega}_m\cdot\vec{n}^i<0} w_m \left\langle \Psi_m^+,\Psi_m^{*+}\right\rangle_e
\end{aligned}
\tag{5.4}
$$

It can be easily seen that the system corresponding this bilinear form can be divided into np separate systems which can then be solved separately. When we use synchronous communications, we basically move the term $\sum_{i=1}^{np}\sum_{e\in\mathbb{F}_i}\sum_{\vec{\Omega}_m\cdot\vec{n}^i<0} w_m \left\langle \Psi_m^-,\Psi_m^{*+}\right\rangle_e$ to the right-hand-side, like in the case of reflecting boundaries. The subdomain interface is treated like a reflecting boundaries. We define the surface source on the subdomain interface $\mathbb{F}_i, i=1,...,np$ for DSA as

$$
\delta J^{inc}\Big|_{\vec{r}\in\mathbb{F}_i} = \sum_{\vec{\Omega}_m\cdot\vec{n}^i<0} w_m|\vec{\Omega}_m\cdot\vec{n}^i|\delta\Psi_m^{-(\ell)} \tag{5.5}
$$

$$
\delta\vec{\Upsilon}^{inc}\Big|_{\vec{r}\in\mathbb{F}_i} = -\sum_{\vec{\Omega}_m\cdot\vec{n}^i<0} 3w_m\vec{\Omega}_m|\vec{\Omega}_m\cdot\vec{n}^i|\delta\Psi_m^{-(\ell)}, \tag{5.6}
$$

where $\delta\Psi_m^{-(\ell)}$ is the difference in the upwind (indicated by $'-'$ superscript) angular fluxes obtained from the active ghost elements before and after one transport sweep at the ℓ^{th} iteration; the value after the transport sweep is obtained through synchronous communications. Note that we will need to deal with the mesh irregularity for these interfaces, which is different from the reflecting boundary situation where irregularity does not exist.

After we obtain the scalar error $\mathcal{E}^{(\ell+1/2)}$, we can accelerate the outgoing angular

fluxes on all active elements on the interfaces:

$$\begin{aligned}\Psi_m^{+(\ell+1)} =& \Psi_m^{+(\ell+1/2)} + \frac{1}{4\pi}(\mathcal{E}^{(\ell+1/2)} - 3D\vec{\nabla}\mathcal{E}^{(\ell+1/2)} \cdot \vec{\Omega}_m) \\ & \text{on} \quad \vec{r} \in \mathbb{F}_i, \vec{\Omega}_m \cdot \vec{n}^i > 0.\end{aligned} \tag{5.7}$$

Through one synchronous communication, these accelerated values reach the neighboring processors as ghost incoming angular fluxes for the next iteration. We have discarded the $\vec{Q}_1$ contribution in the above equation.

Note that like flux moments from a previous AMR iteration were used to bootstrap the numerical solution at the next iteration, we use the ghost incoming angular fluxes of the previous cycle as the initial guess. Because all edges on the subdomains' interfaces possess a natural ordering like all the active elements, the solution on the edges can be projected with the simple algorithm presented in Chapter IV. It is possible that an element in subdomain i may have two sides on the interface $\mathbb{F}_{i,j}$. This situation is illustrated in Fig. V-2. We may have two copies of the angular fluxes of the ghost element. In this case, the angular fluxes in subdomain j will be projected twice because the projections are done edge-by-edge, and we may have two different solutions on the common vertex if the element solution to be projected is in a richer solution space. However, the difference between these two edge-projections on the common vertex should be negligible when the coarse mesh is good enough. Finally, we always have the same copies when one source iteration converges and communications have just been performed.

Parallelization of the diffusion calculation is straightforward if we only have to perform the matrix-vector product, because no ordering is required in this operation. The data needed to be communicated are the scalar fluxes and currents of active ghost elements. Note that that constructing the source requires the $\vec{Q}_1$ term on the active ghost elements, which can only be obtained through communications. For simplicity,

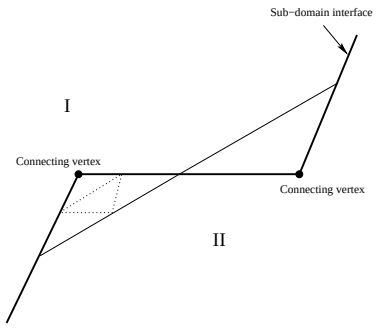


Fig. V-2. Elements have two edges on the one subdomain interface.

we do not do the $\vec{Q}_1$ communications now but assume that the $\vec{Q}_1$ jumps on the subdomain interfaces are always zero.

With the SSOR preconditioner for DSA, all local solutions need to be ordered again. Fortunately this ordering is much simpler than the transport ordering. The majority of local solutions, i.e., the solutions for all elements whose edges are not part of the subdomain interfaces can be done independently in any arbitrary order. For the rest of elementary solutions, we simply split them into several stages based on the connectivities of all subdomains. Communications are performed after each stage. For example, in the domain configuration of Fig. V-3, there will be two stages: all solutions on the subdomains of processors I and IV can be done completely in the first stage while only solutions not on the domain interfaces are processed for processors II and III; then processors I and IV send their results on the active interface elements to their neighboring processors II and III; in the second stage, processors II and III finish all their left local solutions. Note that each SSOR solve is composed of two sweeps: a forward and a backward solve. We will need to do above solution inversely as shown in the right pane of Fig. V-3.

Processor load could be severely unbalanced when using AMR. This problem

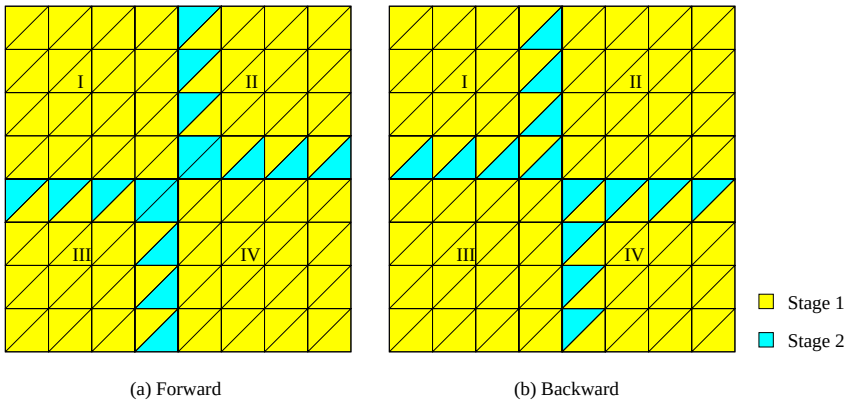


Fig. V-3. Stages of SSOR.

can only be solved by re-partitioning the domain and re-distributing the unknowns accordingly. The idea on how re-partitioning is performed with AMR can be found in [145]. Load rebalancing after AMR is not currently considered in XUTHUS. After the refinement flag is obtained with an *a posteriori* error estimator, communications are performed among all processors whose active elements located along the subdomain interfaces have received the refinement flags. In XUTHUS, the element irregularity constraints (either within one adapted mesh or across group-dependent meshes) are enforced across subdomain interfaces. For example, if 1-irregularity is imposed, Fig. V-4 shows that elements II and III need to be refined due to the refinement flag of element I on the subdomain interface.

4. Matrix-free Scheme

For each type of element, i.e., shape of an element + basis functions defined on the reference element, we can create a collection of local operations. We do not need to use a numerical quadrature to compute these operations, but instead computed their results symbolically and implemented them directly to reduce the flops by exploring

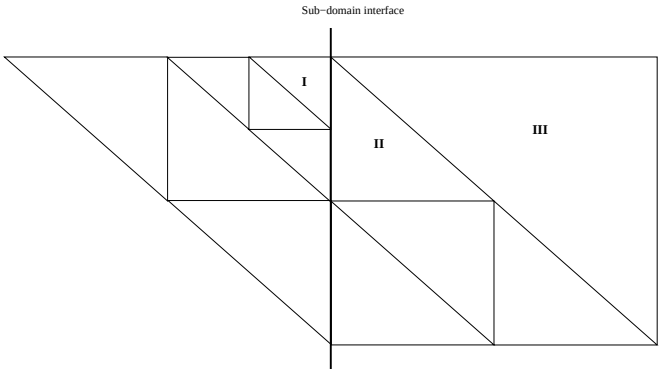


Fig. V-4. Irregularity constraint with domain decomposition.

the known structure of the involved local matrices. Table V-II lists all the local operations. The "importance" of these operations is labeled according to where the operations are used: if they are called inside the one-group transport solver, they have an importance level of 1; if they are called in other parts of multigroup transport solver, they have an importance level of 2; otherwise have an importance level of 3.

In the table, $\mathbf{u}$ is the input solution vector of an element. $\underline{\mathbf{u}}$ is the input solution vector on a local edge. The reference mass matrix M, element prolongation matrices $\mathrm{P}_i, i = 1, 2, 3, 4$, stiffness matrix $\mathbf{S}$ and transport upwind coupling matrices $\overline{\mathbf{H}}_{i,K_i}$ have been defined in previous chapters. The reference 1-D mass matrix $\underline{\mathrm{M}}$, 1 -D prolongation matrices $\underline{\mathrm{P}}_i, i = 1, 2$, rotation matrix $\underline{\mathrm{R}}$ and edge operation matrices $\mathrm{T}_i, \mathbf{N}_i, \mathbf{X}_i, \mathbf{Y}_i, i = 1, 2, 3$ have also been defined previously. A is the triangle area; $L_i, i = 1, 2, 3$ are the lengths of the three edges. We have a few new notations:

Table V-II. List of elementary operations.

Subroutine name	imp.	Operation	Description
ASSEMBLE_CELL_MATRIX	1		Assemble the local element matrix for both transport and diffusion
ADD_UPWIND_RHS_STEP	1	$\mathbf{v} \leftarrow \mathbf{v} + \overline{\overline{\mathbf{H}}}_{i,K_i}\mathbf{u}$	Add upwind contribution to the right-hand-side
PROLONG_SOLUTION_TO	3	$\mathbf{v} \leftarrow \mathrm{P}_i\mathbf{u}$	Cell prolongation
PROLONG_SOLUTION_TADD	3	$\mathbf{v} \leftarrow \mathbf{v} + \mathrm{P}_i^T\mathbf{u}$	Cell transpose prolongation
ELEMENT_L2_NORM	2	$\frac{A}{12}\mathbf{u}^T\mathrm{M}\mathbf{u}$	Evaluate square of the L_2 norm for the solution over an element
ELEMENT_INNER_PROD	2	$\frac{A}{12}\mathbf{v}^T\mathrm{M}\mathbf{u}$	Evaluate the element inner product between two solutions
ELEMENT_VOL_INTEGRAL	2	$\frac{A}{2}\mathbf{s}^T\mathbf{u}$	Evaluate the integral of a solution over an element
ELEMENT_MASS_PADD	2	$\mathbf{v} \leftarrow \mathbf{v} + \frac{A}{12}\mathrm{M}\mathbf{u}$	Multiply the element local mass matrix by a solution vector
MULT_STIFF_PADD	2	$\mathbf{v} \leftarrow \mathbf{v} + \mathbf{S}\mathbf{u}$	Multiply the element local stiffness matrix by a solution vector
GET_CELL_GRAD	2	$\begin{bmatrix}\mathbf{D}_x\\ \mathbf{D}_y\end{bmatrix}\mathbf{u}$	Get the element gradient
GET_CELL_DIV	2	$\begin{bmatrix}\mathbf{D}_x & \mathbf{D}_y\end{bmatrix}\begin{bmatrix}\mathbf{u}_x\\ \mathbf{u}_y\end{bmatrix}$	Get the element divergence
GET_CELL_DERIVATIVE	2	$\mathbf{D}_x\mathbf{u}$ or $\mathbf{D}_y\mathbf{u}$	Get the element derivative in the x- or y- direction
MULT_CELL_TADD_DER	2	$\mathbf{v} \leftarrow \mathbf{v} + \mathbf{D}_x^T\mathbf{u}$ or $\mathbf{D}_y^T\mathbf{u}$	The transpose operation of getting the derivative in the element
ELEMENT_INTEGRAL	1	$\frac{L_i}{2}\mathbf{e}_i^T\mathbf{u}$	Evaluate the solution integral on a local edge
LOCAL_PROJECTION	3	$\mathbf{v} \leftarrow \mathrm{P}\mathbf{u}$	Project a solution vector with one number of DoFs onto a solution vector with another number of DoFs
MULT_INVERSE_MASS	3	$\mathbf{v} \leftarrow \frac{12}{A}\mathrm{M}^{-1}\mathbf{u}$	Multiply the inverse of the element local mass matrix by a solution vector
GET_SIDE_SOLUTION	1	$\underline{\mathbf{v}} \leftarrow \mathrm{T}_i\mathbf{u}$	Extract the solution on a side of a element
ROT_SIDE_SOLUTION	1	$\underline{\mathbf{v}} \leftarrow \underline{\mathrm{R}}\,\underline{\mathbf{u}}$	Rotate the side solution according to the orientation
PUT_SIDE_SOLUTION	1	$\mathbf{v} \leftarrow \mathbf{v} + \mathrm{T}_i^T\underline{\mathbf{u}}$	Put a side solution back to the solution vector of a element

Table V-II. List of elementary operations (Continued).

GET_SIDE_DSOLUTION	1	$\underline{\mathbf{v}} \leftarrow \mathbf{N}_i\mathbf{u}$	Extract the norm derivative of solution on a side of a element
SIDE_DSOL_TADD	1	$\mathbf{v} \leftarrow \mathbf{v} + \mathbf{N}_i^T\underline{\mathbf{u}}$	Transpose operation of GET_SIDE_DSOLUTION
GET_SIDE_DERIVATIVE	1	$\underline{\mathbf{v}} \leftarrow \mathbf{X}_i\mathbf{u}$ or $\mathbf{Y}_i\mathbf{u}$	Extract the derivative of solution in the x- or y- direction on a side of a element
MULT_SIDE_TADD_DER	1	$\mathbf{v} \leftarrow \mathbf{v} + \mathbf{X}_i^T\underline{\mathbf{u}}$ or $\mathbf{Y}_i^T\underline{\mathbf{u}}$	Transpose operation of GET_SIDE_DERIVATIVE
SIDE_MASS_PADD	1	$\underline{\mathbf{v}} \leftarrow \underline{\mathbf{v}} + \frac{L}{6}\underline{\mathrm{M}}\,\underline{\mathbf{u}}$	Multiply the side mass matrix by a solution vector
INV_SIDE_MASS	3	$\underline{\mathbf{v}} \leftarrow \frac{6}{L}\underline{\mathrm{M}}^{-1}\,\underline{\mathbf{u}}$	Multiply the inverse of the side local mass matrix
PROLONG_SOLUTION1D_TO	1	$\underline{\mathbf{v}} \leftarrow \underline{\mathrm{P}}_i\underline{\mathbf{u}}$	Prolongate the solution onto a side
PROLONG_SOLUTION_TADD1D	1	$\underline{\mathbf{v}} \leftarrow \underline{\mathbf{v}} + \underline{\mathrm{P}}_i^T\underline{\mathbf{u}}$	Transpose operation of PROLONG_SOLUTION1D_TO
SIDE_L2_NORM	3	$\frac{L}{6}\underline{\mathbf{u}}^T\underline{\mathrm{M}}\,\underline{\mathbf{u}}$	Evaluate the integral of solution squared on one side
CELL_POINT_SOLUTION	3	$\mathbf{b}^T(\xi_1,\xi_2)\mathbf{u}$	Get a point solution inside a element with barycentric coordinates
ELEMENT_L1_NORM	3	$\int_K \lvert u(x,y)\rvert\, dxdy$	Evaluate the integral of absolute of solution over an element (Numerical quadrature is used)
GET_SHAPE_FUNCTION	3	$\left[\mathbf{b},\quad \mathbf{D}_x\mathbf{b},\quad \mathbf{D}_y\mathbf{b}\right](\xi_1,\xi_2)$	Get the values for all shape functions and their x,y derivatives on a set of points

$$\mathbf{s} = \int_{\widehat{K}} \widehat{\mathbf{b}}(\xi_1, \xi_2)\, d\xi_1 d\xi_2 \tag{5.8}$$

$$\mathbf{D}_x = \frac{2}{A}\mathrm{M}^{-1} \int_K \mathbf{b}\frac{\partial}{\partial x}\mathbf{b}^T\, dxdy = \frac{1}{A}\mathrm{M}^{-1} \int_{\widehat{K}} \widehat{\mathbf{b}}\vec{\nabla}_\xi \widehat{\mathbf{b}}^T \begin{bmatrix} y_3 - y_1 \\ y_1 - y_2 \end{bmatrix} d\xi_1 d\xi_2 \tag{5.9}$$

$$\mathbf{D}_y = \frac{2}{A}\mathrm{M}^{-1} \int_K \mathbf{b}\frac{\partial}{\partial x}\mathbf{b}^T\, dxdy = -\frac{1}{A}\mathrm{M}^{-1} \int_{\widehat{K}} \widehat{\mathbf{b}}\vec{\nabla}_\xi \widehat{\mathbf{b}}^T \begin{bmatrix} x_3 - x_1 \\ x_1 - x_2 \end{bmatrix} d\xi_1 d\xi_2 \tag{5.10}$$

$$\mathbf{e}_i = \int_{-1}^{+1} \widehat{\mathbf{b}}(\xi_i)\, ds \tag{5.11}$$

$$\mathrm{P} = \mathrm{M}^{-1}_{N(p_1),N(p_1)} \mathrm{M}_{N(p_1),N(p_2)} \tag{5.12}$$

When p_2 is less then p_1, operating with the projection matrix P simply means appending $N(p_1) - N(p_2)$ zeros on the input vector $\mathbf{u}$. But when p_2 is greater then p_1, higher-order terms have non-zero projections on the low-order terms. We used

$$J^{-1} \det(J) = \frac{1}{2} \begin{bmatrix} y_3 - y_1 & x_1 - x_3 \\ y_1 - y2 & x_2 - x_1 \end{bmatrix} \tag{5.13}$$

in the $\mathbf{D}_x$ and $\mathbf{D}_y$ formula. Other notations in the table are straightforward.

In Fig. V-5, we provide, as an example, a piece of pseudo-code that demonstrates how the DCF edge terms are assembled in a matrix-free fashion for an interior edge:

```
! au1 - rhs vector of left element (Output)
! au2 - rhs vector of right element (Output)
! u1 - scalar flux vector of left element (Input)
! u2 - scalar flux vector of right element (Input)
! elm1 - left element ID (Input)
! elm2 - right element ID (Input)
! ied1 - local edge ID in left element (Input)
! ied2 - local edge ID in right element (Input)
! kh - penalty of the edge (Input)
! slen - edge length (Input)
! ------------------------------------------------------
! Following variables can be retrieved with element ID
! ndofs1 - length of u1
! ndofs2 - length of u2
! dc1 - diffusion coefficient of left element
! dc2 - diffusion coefficient of right element
! s2v1 - surface-volume ratio of left element
```

```
! s2v2 - surface-volume ratio of right element
! x1 - x-coordinate of left element
! x2 - x-coordinate of right element
! y1 - y-coordinate of left element
! y2 - y-coordinate of right element
! vol1 - area of left element
! vol2 - area of right element
! ndofside1 - length of left solution vector on an edge
! ndofside2 - length of right solution vector on an edge
! s1 - left scalar flux on the edge
! s2 - right scalar flux on the edge
! ds1 - left norm derivative on the edge
! ds2 - right norm derivative on the edge
! rs1, rs2, t1, t2, tt1, tt2, tx1, tx2, ty1, ty2 - temporary working arrays
! get solutions and normal derivatives on the side into s1,ds1 and s2,ds2
CALL GET_SIDE_SOLUTION (ndofs1,u1,ied1,ndofside1,s1)
CALL GET_SIDE_DSOLUTION(s2v1,slen,ndofs1,u1,ied1,ndofside1,ds1)
CALL GET_SIDE_SOLUTION (ndofs2,u2,ied2,ndofside2,s2)
CALL GET_SIDE_DSOLUTION(s2v2,slen,ndofs2,u2,ied2,ndofside2,ds2)
! get [[u]] with respect to elm2
CALL ROT_SIDE_SOLUTION(ndofside1, s1, -1, rs1)
t1 = zero
CALL SIDE_MASS_PADD(-slen, ndofside1, rs1, ndofside2, t1)
CALL SIDE_MASS_PADD( slen, ndofside2, s2 , ndofside2, t1)
! get [[D*du/dn]]/2 with respect to elm2
CALL ROT_SIDE_SOLUTION(ndofside1, ds1, -1, rs2)
t2 = zero
CALL SIDE_MASS_PADD(-slen*dc1/two, ndofside1, rs2, ndofside2, t2)
CALL SIDE_MASS_PADD( slen*dc2/two, ndofside2, ds2, ndofside2, t2)
! modify rhs coresponding to elm2
t2(1:ndofside2) = t1(1:ndofside2)*kh - t2(1:ndofside2)
CALL PUT_SIDE_SOLUTION(ndofside2, t2, ied2, ndofs2, au2)
t1(1:ndofside2) = -t1(1:ndofside2)*dc2/two
CALL SIDE_DSOL_TADD(s2v2, slen, ndofside2, t1, ied2, ndofs2, au2)
! get [[u]] with respect to elm1
CALL ROT_SIDE_SOLUTION(ndofside2, s2, -1, rs1)
t1 = zero
CALL SIDE_MASS_PADD(-slen, ndofside2, rs1, ndofside1, t1)
CALL SIDE_MASS_PADD( slen, ndofside1, s1,  ndofside1, t1)
! get [[D*du/dn]]/2 with respect to elm1
CALL ROT_SIDE_SOLUTION(ndofside2, ds2, -1, rs2)
t2 = zero
CALL SIDE_MASS_PADD(-slen*dc2/two, ndofside2, rs2, ndofside1, t2)
CALL SIDE_MASS_PADD( slen*dc1/two, ndofside1, ds1, ndofside1, t2)
! modify rhs coresponding to elm1
t2(1:ndofside1) = t1(1:ndofside1)*kh - t2(1:ndofside1)
CALL PUT_ADD_SIDE_SOLUTION(ndofside1, t2, ied1, ndofs1, au1)
t1(1:ndofside1) = -t1(1:ndofside1)*dc1/two
CALL SIDE_DSOL_TADD(s2v1, slen, ndofside1, t1, ied1, ndofs1, au1)
! get x and y derivatives on the side
CALL GET_SIDE_DERIVATIVE(vol1, y1, ied1, ndofs1, u1, ndofside1, dx1)
CALL GET_SIDE_DERIVATIVE(vol1,-x1, ied1, ndofs1, u1, ndofside1, dy1)
CALL GET_SIDE_DERIVATIVE(vol2, y2, ied2, ndofs2, u2, ndofside2, dx2)
CALL GET_SIDE_DERIVATIVE(vol2,-x2, ied2, ndofs2, u2, ndofside2, dy2)
hk = 0.75_8
cc1 = dc1*hk; cc2 = dc2*hk
! get [[dxu]] with respect to elm2
CALL ROT_SIDE_SOLUTION(ndofside1, dx1, -1, rs2)
tx2 = zero
CALL SIDE_MASS_PADD( slen*cc1*cc2, ndofside1, rs2, ndofside2, tx2)
CALL SIDE_MASS_PADD(-slen*cc2*cc2, ndofside2, dx2, ndofside2, tx2)
```

```
CALL MULT_SIDE_TADD_DER(vol2, y2, ied2, ndofside2, tx2, ndofs2, au2)
! get [[dyu]] with respect to elm2
CALL ROT_SIDE_SOLUTION(ndofside1, dy1, -1, rs2)
ty2 = zero
CALL SIDE_MASS_PADD( slen*cc1*cc2, ndofside1, rs2, ndofside2, ty2)
CALL SIDE_MASS_PADD(-slen*cc2*cc2, ndofside2, dy2, ndofside2, ty2)
CALL MULT_SIDE_TADD_DER(vol2,-x2, ied2, ndofside2, ty2, ndofs2, au2)
! get [[dxu]] with respect to elm1
CALL ROT_SIDE_SOLUTION(ndofside2, dx2, -1, rs1)
tx1 = zero
CALL SIDE_MASS_PADD( slen*cc2*cc1, ndofside2, rs1, ndofside1, tx1)
CALL SIDE_MASS_PADD(-slen*cc1*cc1, ndofside1, dx1, ndofside1, tx1)
CALL MULT_SIDE_TADD_DER(vol1, y1, ied1, ndofside1, tx1, ndofs1, au1)
! get [[dyu]] with respect to elm2
CALL ROT_SIDE_SOLUTION(ndofside2, dy2, -1, rs1)
ty1 = zero
CALL SIDE_MASS_PADD( slen*cc2*cc1, ndofside2, rs1, ndofside1, ty1)
CALL SIDE_MASS_PADD(-slen*cc1*cc1, ndofside1, dy1, ndofside1, ty1)
CALL MULT_SIDE_TADD_DER(vol1,-x1, ied1, ndofside1, ty1, ndofs1, au1)
! get {{D*du/dn}} with respect to elm1
CALL ROT_SIDE_SOLUTION(ndofside2, ds2, -1, rs1)
tt1 = zero
CALL SIDE_MASS_PADD(-slen*cc2*cc1, ndofside2, rs1, ndofside1, tt1)
CALL SIDE_MASS_PADD(-slen*cc1*cc1, ndofside1, ds1, ndofside1, tt1)
CALL SIDE_DSOL_TADD(s2v1, slen, ndofside1, tt1, ied1, ndofs1, au1)
! get {{D*du/dn}} with respect to elm2
CALL ROT_SIDE_SOLUTION(ndofside1, ds1, -1, rs2)
tt2 = zero
CALL SIDE_MASS_PADD(-slen*cc1*cc2, ndofside1, rs2, ndofside2, tt2)
CALL SIDE_MASS_PADD(-slen*cc2*cc2, ndofside2, ds2, ndofside2, tt2)
CALL SIDE_DSOL_TADD(s2v2, slen, ndofside2, tt2, ied2, ndofs2, au2)
```

Fig. V-5. Pseudo-code used to assemble the DCF edge terms in the matrix-free fashion.

Since compilers can in-line these small local function calls, there is no efficiency penalty associated with them.

C. Integration into SCALE

1. Procedure

In collaboration with the Nuclear Science and Technology Division (NSTD) of the Oak Ridge National Laboratory (ORNL), TAMU will be integrating XUTHUS into a

developmental version of SCALE, which may ultimately be distributed as a released module within the TRITON lattice physics sequence [71], providing users with an alternative to NEWT, the 2-D S_N module currently used with TRITON. Our initial objectives in the development of XUTHUS as a module for SCALE are fulfilled or verified: (a) arbitrary high-order spatial shape functions can be and have been implemented efficiently in the DGFEM framework, whereas in the Extended Step Characteristic solver of NEWT, the scattering and fission source is assumed spatially constant in each polygonal element, (b) additional flexibility and robustness is gained by having two distinct solvers, (c) the spatial discretization error is controlled with a user-prescribed tolerance with AMR, and (d) TRITON-XUTHUS sequence may be accurately utilized for problems beyond traditional nuclear reactor fuel assembly calculations, including shielding or inverse problems.

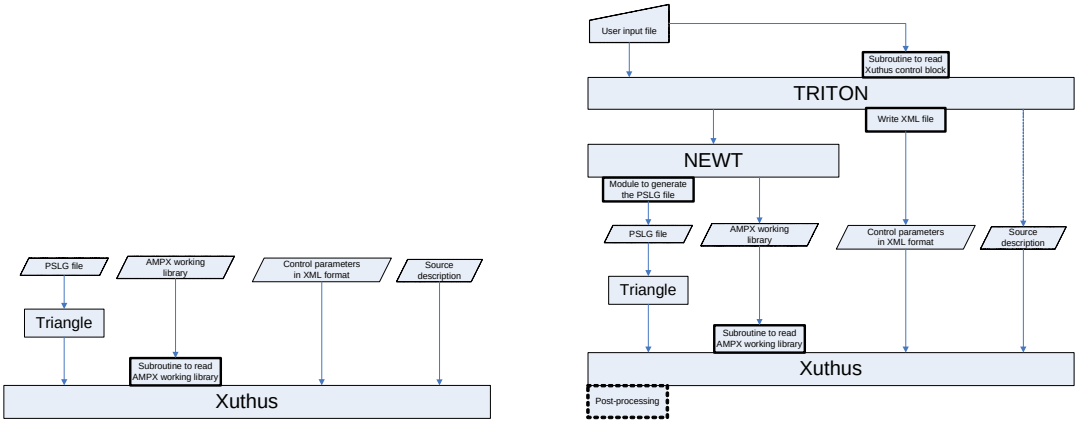


Fig. V-6. Integration of XUTHUS into SCALE: stand-alone mode (left), integration mode (right).

Like all other modules in SCALE, XUTHUS can be run as a standalone module or as part of a SCALE sequence. While XUTHUS provides new capabilities of transport calculation for SCALE, it benefits from other modules: all cross section processing options and the powerful yet easy-to-use geometry description provided by SCALE.

The relations between XUTHUS and other modules are illustrated in Fig. V-6. XUTHUS is loosely coupled with other modules through binary files, like all modules in SCALE.

In standalone mode, XUTHUS accepts a single XML (the Extensible Markup Language) input file, which provides access to all the control parameters, simple material data, regular geometry configurations and external source descriptions for a source problem. This file can be generated easily by hand, which makes it very useful for simple benchmark problems and for the proof-of-principle calculations. XUTHUS can also employ far more complicated problems in the standalone mode, for which a geometry file and/or a cross section library file (in a format readable by XUTHUS) are referenced in the input deck. A subroutine in XUTHUS has been created to read the AMPX-format cross section library. More subroutines can be added later for other types of libraries. XUTHUS accepts files for unstructured meshes generated by the open source triangularization packages Triangle. Triangle used the PSLG (Planar Straight Line Graph) format file (by definition, a PSLG is just a list of vertices and segments), as input. Also note that XUTHUS currently only allows triangular meshes and that boundaries and interfaces of the initial mesh are not changed in the AMR procedure.

The XUTHUS-based sequence within the TRITON control module of SCALE utilizes the automatically generated cross sections. In this sequence, TRITON reads the cross sections of all isotopes generated by the SCALE cross section processing routines and uses the SCALE ICE module to generate an AMPX-format working library of mixed macroscopic cross sections. For the details of the format of AMPX working library, see the manual of NITAWL, section F2.4. By doing this, all SCALE cross section processing options are seamlessly integrated into a XUTHUS-based sequence.

The XUTHUS-based sequence can also be used to automatically generate the PSLG file. The grid generation capabilities in NEWT (based on the combinato-

rial SCALE Generalized Geometry Package used by KENO VI and Monaco within SCALE) provide a simplified user input specification in which elementary bodies can be defined and placed within a problem domain. This body description is transformed into meshes of arbitrary polygons and can be used to closely approximate curved or irregular surfaces with volume preservation. A module has been created for NEWT to output the mesh into a PSLG which is then processed using the Triangle mesh generator. This module will be discussed in more detail later.

The XUTHUS-based sequence also provide a free-form and keyword-based input, similar in form to the input for many other modules in the SCALE code package. A subroutine for TRITON has been created to parse the input block specifically for XUTHUS control parameters. The *Control* module in XUTHUS can then be linked with TRITON, in which a subroutine to set default values for all control parameters and a subroutine to output parameters into a XML file are provided. TRITON then uses these subroutines to generate the XML file for XUTHUS.

So far, only a few things are missing to complete the integration into SCALE:

- Integrate the external source description for source problems.
- Form the sequence to call NEWT, XUTHUS and Triangle automatically.
- Add a post-processing module to XUTHUS so that the XUTHUS-based sequence can also perform 2-D depletion calculations.

2. Creation of a .poly File from NEWT

A .poly file, the input file required for the 2-D triangulation software - Triangle, contains a PSLG, as well as some additional information. By definition, a PSLG is just a list of vertices and segments. A .poly file can also contain information about holes in the domain, as well as regional attributes and constraints on the

maximum triangle areas. The format of .poly file for Triangle can be found in: `http://www.cs.cmu.edu/~quake/triangle.html`.

NEWT possesses its own mesh generator with the combinatorial SCALE Generalized Geometry Package used by KENO VI and its own compact data structure to describe polygons which are formed by the body interfaces and base grid-lines. In order to create a .poly file understandable by Triangle, we need to expand this data structure with additional information. The unstructured triangular mesh created by Triangle can then be read into XUTHUS for further computations. The main complexity in creating a PSLG .poly file from NEWT is the requirement to find an arbitrary point inside any polygon that is not necessarily convex (in order to assign a regional attribute index).

a. NEWT's Data Structure

Each polygon is called a element in NEWT. The user-defined type of element is as follows:

```
type elementstr
  integer :: material
  integer :: numsides
  integer, pointer :: side(:)
end type
```

Note that only entries relevant in creating a .poly file are listed. A global variable *numelements* described many elements or polygons are on the computational domain and the size of the array element. A polygon is composed of *numsides* number of sides, stored in the entry *side*. Using these side numbers, we can retrieve their left and right element numbers with another user-defined type:

```
type lineseg
  integer :: left
  integer :: rite
end type lineseg
```

Total number of sides is stored in the global variable *numlines*. Additional information about all lines (coordinates of their beginning and ending points,; whether they are on the boundary and the boundary type) can be obtained with the *endpts*array which has the user-defined type:

```
type lineends
  type (point) :: beg
  type (point) :: end
  integer :: boundtype
end type lineends
```

where the user-defined type *point* is:

```
type point
  double precision :: x
  double precision :: y
end type point
```

This data structure is very compact. Another remarkable feature is that there are no numberings of the vertices. Note that NEWT has its own encoding of boundary type. We need a small subroutine to transform the `boundtype` into XUTHUS's types of boundaries such as vacuum, reflecting, etc.

b. The Expanded Data Structure

To create the expanded data structure, we first collect all vertices into the array *verts* of data type *point*, count number of vertices into the global variable *nrvers*. With these numbered vertices, we can then create another array, *edges*, for all sides. Each entry of *edges* contains the indices of the beginning and the ending vertices. With this information, we can re-describe all elements with their numbered vertices and their sides in the array *elems*. Each entry of *elems* is the collection of side numbers and vertex numbers. Here, all vertices and sides of a element are arranged clockwise. It is possible that the orientation of a side in a element is in the opposite direction of the one determined by the *edges* array. We use a negative number to indicate this situation.

The algorithm to collect all vertices and form the array *edges* and *elems* is straightforward: we loop over all elements, and then consider its vertices and edges clockwise one by one. The way to determine if a vertex has already been in the array *verts* or if we need to add a new vertex into the array *verts* is by inspecting all elements connected with this vertex and check whether any one of them has been visited. All sides have already been numbered in NEWT.

c. Algorithm to Find an Arbitrary Point inside a Non-convex Polygon

Creating a .poly file with the expanded data structure is simple. The only difficulty resides in finding an arbitrary point inside a polygon. With a polygon whose vertices are numbered clockwise and all sides form a closed area which is nonzero, the algorithm for obtaing a point within that polygon is given below:

1. Identify a convex vertex v: letting its adjacent vertices be a and b, we make sure that$\widehat{(avb)} < \pi$; note this is a strict inequality, i.e., a, b and v are not on a

line.

2. Algorithm fails when no v can be identified.

3. For each other vertex q do:

   - if q is on any edge of triangle avb, find a new convex vertex v by restarting at step 1
   - if q is inside triangle avb, compute distance to v (orthogonal to line ab).
   - save point q if distance is a new minimum.

4. If no points are inside, return the midpoint of ab, or centroid of avb.

5. Else if some point is inside, qv is internal: return its midpoint.

Although the algorithm geometrically makes sense, we need be careful in step 1 and step 3.a. (see the additional comments in the parenthesis.) Machine round off error may be significant in some very extreme cases.

3. Dealing of Polygon Attributes

Another problem is that Triangle only supports one single zonal attribute whereas we need to have multiple attributes to describe the material ID, the external source ID (source problem only), the subdomain ID (parallel calculation only) and/or region ID (for the purpose of computing regional particle rebalance). To solve this issue, we encode all thse required data into a single attribute with the following equation:

$$code_{polygon} = m_{id}+s_{id}\times(N_M+1)+d_{id}\times(N_M+1)\times(N_S+1)+(r_{id}-1)\times(N_M+1)\times(N_S+1)\times N_D \tag{5.14}$$

where

m_{id} the material ID (≥ 0)

N_M the total number of materials

s_{id} the external source ID (≥ 0)

N_S the total number of external sources

d_{id} the subdomain ID (≥ 0)

N_D the total number of subdomains

r_{id} the region ID (≥ 1)

N_R the total number of regions

Material ID 0 is reserved for void. Material ID and region ID for users start at 1 while source ID and subdomain ID start at 0. A source with ID=0 means no external source. Note that the number of materials does not include the void medium. Note also that number of sources does not include the non-sources. The maximum subdomain ID must be less than the number of subdomains.

Then the first line of the .poly file should contain one line generated with

```
WRITE(*,'(A,4I,A)') '#code:', nmat, nsour, ndomain, nregion, ' '
```

This line will be treated as a comment line by Triangle because of the leading pound sign (#), but later can be read by XUTHUS and help decode the multiple attributes. The last space character is used to avoid text formating between Unix and DOS/Windows.

4. A Sample Triangular Mesh Created with NEWT

The left pane of Fig. V-7 is the grid structure of the example 4 in NEWT's manual [72]. This problem illustrates a calculation with one-forth of a PWR fuel assembly. A PSLG file is generated with the above procedure to describe this grid structure. Then the

polygon grid is further processed with Triangle into the triangular mesh shown in the right pane of Fig. V-7.

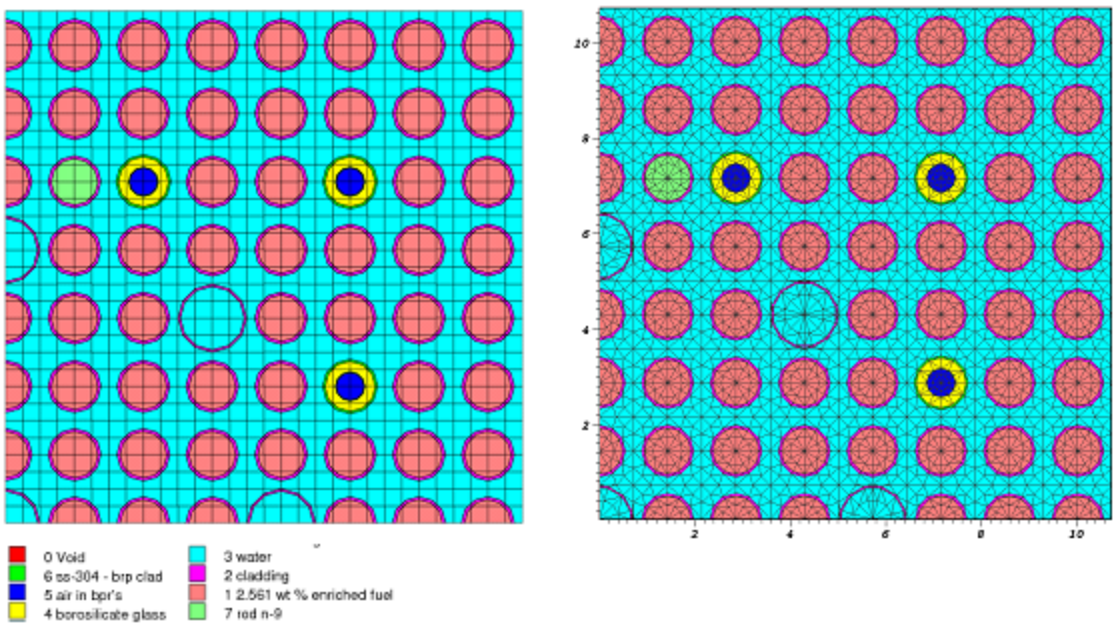


Fig. V-7. NEWT's polygon grid and XUTHUS's triangular mesh.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

We have developed a new 2-D multigroup S_N transport solver XUTHUS for unstructured triangular meshes. The spatial discretization has been carried out using a high-order Discontinuous Galerkin Finite Element Method (DGFEM), and Adaptive Mesh Refinement (AMR) techniques have been implemented and tested with this solver. Two error estimations, a projection-based error estimator and a jump-based error indicator, have been devised and implemented to drive the mesh adaptation procedure so as to deliver adapted meshes that tightly follow the physics. Different adapted meshes for different energy groups are automatically generated in XUTHUS, leading to group-dependent adapted meshes (the concept of "multi-mesh" developed in this dissertation.) Furthermore, within a given adapted mesh, the difference in refinement levels between two adjacent elements can be arbitrarily greater than one (the concept of "multi-irregularity"). Algorithms to deal with the multi-mesh couplings and the mesh multi-irregularity have been designed, implemented, and tested for the multigroup S_N equations. For the first time, the spatial discretization error is controlled for the S_N transport equation in the multigroup setting, i.e., the relative error in the multigroup scalar fluxes is assured to be smaller than a user-prescribed tolerance. Convergence studies with both the h-adaptation and the uniform refinement have been conducted, proving the superiority of mesh adaptivity, both in terms of number of unknowns and CPU time. For an efficient solution of the transport equation, especially for geometries containing highly diffusive media, a stable Symmetric Positive Definite (SPD) Diffusion Synthetic Acceleration (DSA) scheme, based on a modification of the Interior Penalty method, has been devised. This Modified

IP (MIP) DSA can be employed as an accelerator for the standard Source Iteration procedure or as a preconditioner for the GMRes solver applied to the transport equation. This MIP DSA is derived from the discretized DG transport equation using a variational argument, resulting in a DG diffusion solver that can handle hanging nodes (multi-irregular mesh)very efficiently and straightforwardly. All algorithms are implemented in a matrix-free fashion.

We tested our implementations extensively with sample problems presented in Chapters II, III and IV. Our conclusions are listed below:

1. The spatial discretization error for the S_N transport equation can be controlled for unstructured meshes using a DGFEM AMR technique. Two error estimates, a projection-based estimator and a jump-based indicator, have been devised and used to drive the h-adaptation reliably. This technique is easily applied to basis functions of degree greater than 1. We have utilized polynomial orders up to 4. The number of unknowns used in meshes stemming from adaptivity is significantly smaller than the number needed with uniform mesh refinement for the same level of accuracy. Both the computing time and memory cost are greatly reduced. The concepts of arbitrary irregularity and group-dependent meshes can be effectively implemented, leading to an adaptive technique that follows closely the physics of the problem under consideration, both within one group and among groups.

2. With h-adaptivity, we can capture the singularities in the S_N solution, i.e., the ray effects, and regions of material discontinuities are refined more automatically. The methodology implemented in this research can apply to both source and eigenvalue problems.

3. The solution of the S_N transport equation belongs in the $\mathrm{H}^{3/2}$ space when it is

continuous or in the $H^{1/2}$ space when it is discontinuous. The regularity index r is then 3/2 and 1/2, respectively. With uniform refinement, the convergence rates for the scalar flux are known to be equal to $min(r, p+1)$ in the L_2 norm. When the mesh is aligned with the known singularity lines of the S_N solution, a convergence order of $p+1$ can be restored. In the more general situation, when the mesh is not aligned with the singularities, the AMR technique delivers an accurate solution faster than uniform mesh refinement.

4. It is observed that the spatial discretization errors obtained with the same numbers of unknowns are smaller when higher-order basis functions are used, for both source problems and eigenvalue problems, establishing the advantage of higher-order calculations.

5. A modified IP form for DSA has been devised for AMR meshes. This MIP-DSA is SPD (thus can be solved effectively with preconditioned CG method). A Fourier Analysis has determined that the form is stable but its efficacy degrades in the cases where (i) materials of greatly disparate cross section are present and (ii) scattering anisotropy is strong. In such cases, another scheme, the positive definite P1C (P_1 Conforming), has been proposed although more numerical verifications need to be conducted.

6. The grind time (i.e., the average time needed to solve a local transport problem in one direction for one element) of the transport sweeps is about $0.2\mu s$ per unknown, which is in the typical range for similar codes. This grind time is determined not only by the number of elementary operations but also by memory access delays. With current cache capacity, the grind time for quadratic elements is even smaller than the one for linear elements, although the number of elementary operations is roughly four times larger. When the element poly-

nomial order increases, the number of local operations becomes dominant and the grind time increases. Nevertheless, the grind times of different polynomial orders are about the same.

This study opens several perspectives for continued research:

1. Full *hp*-adaptation

   Although the *hp*-mechanism has been implemented in our solver XUTHUS, the error estimation to drive *hp*-type mesh adaptation is absent, i.e., we have not implemented a mechanism to determine, at each refinement cycle, not only which elements need to be refined but also how these selected elements are to be refined: either by subdivision (*h*-refinement) or by increasing the polynomial order (*p*-refinement). Nonetheless, for this purpose, the projection-based error estimator can be easily extended to deliver *hp*-adaptation as shown, for instance, in [125, 5, 56].

2. Goal-oriented calculations

   From an engineering point of view, in most cases, only some localized quantities of interest may be needed, rather than the full detailed solution over the entire domain. These quantities of interest are specific properties of the solution and can usually be represented by locally bounded linear functionals of the solution. The mesh, adapted for the overall solution, may not be a good enough choice for such goals (too many unknowns may be wasted in regions that bear no importance to the quantity of interest, not enough refinement may be performed in the zones of interest, ...). To demonstrate this, let us consider the problem described in Chapter III, page 159. As our quantity of interest, we use the leakage through a part of the boundary at the right-top corner of the domain. This zone is marked on Fig. III-28. The calculated leakage obtained with

Table VI-I. Leakage convergence with uniform refinement.

Refinement level	Number of active elements	Leakage (n/cm^2/sec)
0	50	$2.8444426472 \times 10^{-6}$
1	200	$2.1614548069 \times 10^{-6}$
2	800	$2.1358961790 \times 10^{-6}$
3	3200	$2.1353830681 \times 10^{-6}$
4	12800	$2.1353634532 \times 10^{-6}$
5	51200	$2.1353630483 \times 10^{-6}$
6	204800	$2.1353630115 \times 10^{-6}$
7	819200	$2.1353630135 \times 10^{-6}$
8	3276800	$2.1353630124 \times 10^{-6}$

uniform refinement and h-adaptation is listed in Tables VI-I and VI-II. The calculation conditions are: uniform polynomial order 2, LS-4, $\text{tol}_{source} = 10^{-8}$, 2-irregularity and projection-based error estimator $\mu_{g,K}^{k,ref}$. We plot these results on Fig. VI-1 by taking the reference leakage solution to be $2.135363011 \times 10^{-6}$ (n/cm^2/sec).

The results clearly show that adapted meshes produce a less accurate result than uniform meshes in terms of the number of unknowns for the same accuracy. This is due to the fact that elements on the boundary edge are important for evaluating our quantity of interest, yet they are not refined in h-adaptation due to the small absolute flux values in these elements. This example does not imply that uniform refinement is the best approach to follow for goal-oriented calculations. Since the quantities of interest are usually local quantities, i.e., only local refinements are needed, a procedure that combines mesh adaptivity

Table VI-II. Leakage convergence with h-adaptation.

Refinement level	Number of active elements	Leakage (n/cm/sec) mesh $\mathbb{T}_h^k$	mesh $\mathbb{T}_{h/2}^k$
0	50	$2.8444426472 \times 10^{-6}$	$2.1614548069 \times 10^{-6}$
1	74	$2.9042896273 \times 10^{-6}$	$2.1619893940 \times 10^{-6}$
2	104	$2.5920154141 \times 10^{-6}$	$2.1556978825 \times 10^{-6}$
3	122	$2.5918812437 \times 10^{-6}$	$2.1556963968 \times 10^{-6}$
4	206	$2.9580990894 \times 10^{-6}$	$2.1477897298 \times 10^{-6}$
5	278	$2.9583467260 \times 10^{-6}$	$2.1477923789 \times 10^{-6}$
6	356	$2.9502350894 \times 10^{-6}$	$2.1476949285 \times 10^{-6}$
7	500	$2.9453945107 \times 10^{-6}$	$2.1476837935 \times 10^{-6}$
8	794	$2.6263097588 \times 10^{-6}$	$2.1470871126 \times 10^{-6}$
9	1112	$2.6171190140 \times 10^{-6}$	$2.1470138746 \times 10^{-6}$
10	1526	$2.6247919297 \times 10^{-6}$	$2.1467669171 \times 10^{-6}$
11	2174	$2.5085257044 \times 10^{-6}$	$2.1402683204 \times 10^{-6}$
12	2942	$2.5160732394 \times 10^{-6}$	$2.1402416073 \times 10^{-6}$
13	3716	$2.5149788239 \times 10^{-6}$	$2.1402312038 \times 10^{-6}$
14	5144	$2.1679688290 \times 10^{-6}$	$2.1364072688 \times 10^{-6}$
15	6848	$2.1652010997 \times 10^{-6}$	$2.1364140375 \times 10^{-6}$
16	8096	$2.1658738860 \times 10^{-6}$	$2.1363998825 \times 10^{-6}$
17	10328	$2.1619951446 \times 10^{-6}$	$2.1362326415 \times 10^{-6}$
18	12746	$2.1615735692 \times 10^{-6}$	$2.1362572581 \times 10^{-6}$
19	15476	$2.1615961869 \times 10^{-6}$	$2.1362548548 \times 10^{-6}$
20	20834	$2.1607686278 \times 10^{-6}$	$2.1362514431 \times 10^{-6}$
21	26060	$2.1395352866 \times 10^{-6}$	$2.1355711575 \times 10^{-6}$
22	31748	$2.1395971343 \times 10^{-6}$	$2.1355733854 \times 10^{-6}$
23	41138	$2.1406683718 \times 10^{-6}$	$2.1353756881 \times 10^{-6}$
24	45116	$2.1406157596 \times 10^{-6}$	$2.1353731232 \times 10^{-6}$

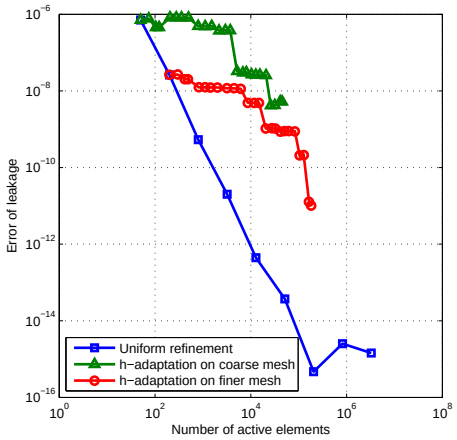


Fig. VI-1. Convergence of the boundary leakage with uniform refinement and *h*-adaptation.

and importance towards a given goal must to be employed. This technique is usually referred to as goal-oriented mesh adaptivity.

3. Angular adaptivity

   Singularity lines of the S_N equations (ray effects) have been noted in several examples in Chapters III and IV. Solutions in regions where these lines exist are not accurate due to the presence of a significant angular discretization error. Although solutions in regions far from the S_N singularities are more reliable, how the angular discretization error propagates through the domain is not clear. It is always possible to increase the number of streaming directions in order to reduce the angular error, but such a uniform angular approach soon becomes costly due to the rapid increase in computing efforts. We believe that the issues of both the spatial and angular discretization errors should be further tackled with a space/angle adaptivity technique.

4. Error control for the multigroup approximation

   It is difficult to quantify the error introduced by the multigroup approximation because of the complicated energy dependence of nuclear data. On the one hand, we can not afford a huge number of energy groups because the scattering kernel is tightly dependent on the energy discretization and the uncertainty of the nuclear data may make such efforts unnecessary. On the other hand, the more groups we employ, the smaller the dependence on the accuracy of the intra-group energy spectrum required. The energy discretization error needs to be investigated in future and these investigations should be tightly coupled with angular discretization through the scattering kernel.

5. Implementation of P_1 conforming DSA scheme

   Our preliminary results with the simple 2-cell problem of Chapter III suggest that the P1C scheme is very promising. However, Fourier Analysis and numerical verifications are needed to fully establish the properties of such a scheme.

6. Load re-balance issues for parallel computing using mesh refinement

   As of now, once the initial domain has been partitioned, each processor owns one of the partitions called sub-domains. If these subdomains undergo different refinements, the processors' loads may become severely unbalanced. Because such a situation depends on the physics of the solution, which we do not know beforehand, we would have to dynamically re-distribute the mesh among processors after mesh refinement in order to retain a load balance among all the processors for the next mesh adaptivity cycle. More robust data management is required for this matter.

7. Multi-physics, non-linear coupling with curved geometries

   XUTHUS has only been applied for the neutron transport equation as of now.

Piece-wise constant cross sections are assumed in XUTHUS and curved geometries are approximated by polygons in the initial mesh. To be able to couple other physics to XUTHUS, we should be able to deal with cross sections that can vary spatially based on variables from other physics such as temperature. Additionally, support for isoparametric elements in order to describe curved element faces may also be required.

REFERENCES

[1] Tomasz Plewa, Timur Linde, and V. Gregory Weirs, editors. *Adaptive Mesh Refinement - Theory and Applications*. Springer, Berlin, 2005.

[2] Graham F. Carey. *Computational Grids: Generations, Adaptation & Solution Strategies*. Computational and Physical Processes in Mechanics and Thermal Sciences. Taylor & Francis, Bristol, PA, 1997.

[3] Ekkehard Ramm, E. Rank, R. Rannacher, K. Schweizerhof, E. Stein, W. Wendland, G. Wittum, Peter Wriggers, and Walter Wunderlich. *Error-controlled Adaptive Finite Elements in Solid Mechanics*. Wiley, Chichester, England, 2003.

[4] George Em Karniadakis and Spencer J. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, New York, NY, 2nd edition, 2005.

[5] Pavel Solin, Karel Segeth, and Ivo Dolezel. *Higher-Order Finite Element Methods*. Chapman & Hall/CRC, New York, NY, 2003.

[6] C. Schwab. *p- and hp- Finite Element Methods: theory and applications in solid and fluid mechanics*. Clarendon Press, Oxford, 1998.

[7] James J. Duderstadt and Louis J. Hamilton. *Nuclear Reactor Analysis*. John Wiley & Sons, Inc., New York, NY, 1976.

[8] D. S. Anikonov, A. E. Kovtanyuk, and Iu. V. Prokhorov. *Transport Equation and Tomography*. Brill Academic Publishers, New York, NY, 2002.

[9] K. N. Liou. *An Introduction to Atmospheric Radiation*. Academic Press, New York, NY, 2nd edition, 2002.

[10] J. T. Oden and S. Prudhomme. Goal-oriented error estimation and adaptivity for the finite element method. *Comput. Math. Appl.*, 41:735–756, 2001.

[11] P. Solin and L. Demkowicz. Goal-oriented *hp*-adaptivity for elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 193(6-8):449–468, 2004.

[12] J. E. Morel. On the validity of the extended transport cross-section correction for low-energy electron transport. *Nucl. Sci. Eng.*, 71:64–71, 1979.

[13] Dan Ilas, Mark L. Williams, Douglas E. Peplow, and Bernadette L. Kirk. Multidimensional coupled photon-electron transport simulations using neutral particle S_N codes. In *Computational Medical Physics Working Group Workshop II*, September 30–October 3, 2007.

[14] David L. Hetrick. *Dynamics of Nuclear Reactors*. USA : American Nuclear Society, La Grange Park, IL, 3rd edition, 1993.

[15] M.B. Chadwick, P. Obložinský, M. Herman, N.M. Greene, R.D. McKnight, D.L. Smith, P.G. Young, R.E. MacFarlane, G.M. Hale, S.C. Frankle, A.C. Kahler, T. Kawano, R.C. Little, D.G. Madland, P. Moller, R.D. Mosteller, P.R. Page, P. Talou, H. Trellue, M.C. White, W.B. Wilson, R. Arcilla, C.L. Dunford, S.F. Mughabghab, B. Pritychenko, D. Rochman, A.A. Sonzogni, C.R. Lubitz, T.H. Trumbull, J.P. Weinman, D.A. Brown, D.E. Cullen, D.P. Heinrichs, D.P. McNabb, H. Derrien, M.E. Dunn, N.M. Larson, L.C. Leal, A.D. Carlson, R.C. Block, J.B. Briggs, E.T. Cheng, H.C. Huria, M.L. Zerkle, K.S. Kozier, A. Courcelle, V. Pronyaev, and S.C. van der Marck. ENDF/B-VII.0: Next generation

evaluated nuclear data library for nuclear science and technology. *Nuclear Data Sheets*, 107(12):2931–3118, December 2006.

[16] K. Shibata, T. Kawano, T. Nakagawa, O. Iwamoto, J. Katakura, S. Chiba T. Fukahori, A. Hasegawa, H. Matsunobu T. Murata, T. Ohsawa, T. Yoshida Y. Nakajima, A. Zukeran, M. Kawai, M. Baba, M. Ishikawa, T. Watanabe T. Asami, Y. Watanabe, M. Igashira, N. Yamamuro, N. Yamano H. Kitazawa, and H. Takano. Japanese Evaluated Nuclear Data Library Version 3 Revision-3: JENDL-3.3. *J. Nucl. Sci. Technol.*, 39:1125, 2002.

[17] Arjan Koning, Robin Forrest, Mark Kellett, Robert Mills, Hans Henriksson, and Yolanda Rugama. The JEFF-3.1 nuclear data library. Research Report 6190, Nuclear Energy Agency, Organisation for Economic Co-operation and Development, Paris, France, 2006.

[18] The Members of the Cross Section Evaluation Working Group. ENDF-6 formats manual. Research Report 44945-05-Rev, National Nuclear Data Center, Brookhaven National Laboratory, Upton, N.Y., 2005.

[19] Maurice Greene Edward Lent Robert MacFarlane Scott McKinley Ernest F. Plechaty Jean Christophe Sublet Dermott E. Cullen, Roger N. Blomquist. How accurately can we calculate neutrons slowing down in water? Research report, Lawrence Livermore National Laboratory, Livermore, CA, April 2006.

[20] Njoy99, nuclear data processing system. Accessed on http://t2.lanl.gov/codes/njoy99/index.html, 2007.

[21] M. E. Dunn and N. M. Greene. AMPX-2000: Cross-section processing system for generating nuclear data for criticality safety applications. *Trans. Am. Nucl. Soc.*, 86:118–119, 2002.

[22] SCALE: A modular code system for performing standardized computer analyses for licensing evaluation. Research Report ORNL/TM-2005/39, Version 5.1, Vols. I-III, Oak Ridge, TN, November 2006.

[23] G. Aliberti, G. Palmiotti, M. Salvatores, T.K. Kim, T.A. Taiwo, M. Anitescu, I. Kodeli, E. Sartori, J.C. Bosq, and J. Tommasi. Nuclear data sensitivity, uncertainty and target accuracy assessment for future nuclear systems. *Ann. Nucl. Energy*, 33(8):700–733, 2006.

[24] Matthew Anderson Jessee. *Cross-Section Adjustment Techniques for BWR Adaptive Simulation*. PhD thesis, North Carolina State University, 2008.

[25] G. J. Bell and S. Glasstone. *Nuclear Reactor Theory*. Van Nostrand Reinhold Company, New York, NY, 1970.

[26] E. M. Gelbard. Application of spherical harmonics method to reactor problems. Technical report, WAPD-BT-20, 1960.

[27] R. E. Alcouffe and E. W. Larsen. A review of characteristic methods used to solve the linear transport equation. In *Proc. Int. Topical Meeting Advances in Mathematical Methods for the Solution of Nuclear Engineering Problems*, volume 1, Munich, FRG, Apirl 27-29, 1981. Fachinformationszentrum Energie, Physik, Mathematik GmbH, Karlsruhe.

[28] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation, Los Alamos Scientific Laboratory Report. Technical Report LA-UR-73-479, Los Alamos National Laboratory, 1973.

[29] P. Lesaint and P. A. Raviart. On a finite element method for solving the neutron transport equation. In *Mathematical Aspects of Finite Elements in Partial*

Differential Equations, pages 89–145, C.A. deBoor (Ed.), 1974. Academic Press, New York.

[30] C. Johnson, U. Navert, and J. Pitkaranta. Finite element methods for linear hyperbolic problems. *Comp. Meth. Appl. Mech. Engr.*, 45:285–312, 1984.

[31] Todd A. Wareing, John M. McGhee, Jim E. Morel, and Shawn D. Pautz. Discontinuous finite element S_N methods on three-dimensional unstructured grids. *Nucl. Sci. Eng.*, 138:256–268, 2001.

[32] Jim E. Morel and James S. Warsa. An S_N spatial discretization scheme for tetrahedral meshes. *Nucl. Sci. Eng.*, 151:157–166, 2005.

[33] B. G. Carlson and K. D. Lathrop. *Computing Methods in Reactor Physics: Transport Theory-The Method of Discrete Ordinates*. Gordon and Breach Science Publishers, Inc., New York, NY, 1968.

[34] K. D. Lathrop. Ray effects in discrete ordinates equation. *Nucl. Sci. Eng.*, 32:357–369, 1968.

[35] K. D. Lathrop. Remedies for ray effect. *Nucl. Sci. Eng.*, 45:255–268, 1971.

[36] J. E. Morel, T. A. Wareing, R. B. Lowrie, and D. K. Parsons. Analysis of ray-effect mitigation techniques. *Nucl. Sci. Eng.*, 144:1–22, 2003.

[37] R. Sanchez. Review of neutron transport approximations. *Nucl. Sci. Eng.*, 80:481–535, 1982.

[38] J. E. Morel and J. M. McGhee. A self-adjoint angular flux equation. *Nucl. Sci. Eng.*, 132:312–325, 1999.

[39] Jim E. Morel, B. Todd Adams, Taewan Noh, John M. McGhee, Thomas M. Evans, and Todd J. Urbatsch. Spatial discretizations for self-adjoint forms of the radiative transfer equations. *J. Comput. Phys.*, 214:12–40, 2006.

[40] W. H. Reed, T. R. Hill, F. W. Brinkley, and K. D. Lathrop. TRIPLET: A two-dimensional, multigroup, triangular mesh, planar geometry, explicit transport code. Technical Report LA–5428-MS, Los Alamos National Laboratory, 1973.

[41] W. H. Reed, T. R. Hill, F. W. Brinkley, and K. D. Lathrop. TRIDENT: A two dimensional multigroup, triangular mesh, explicit neutron transport code. Technical Report LA-6735-MS, Los Alamos National Laboratory, 1977.

[42] T. J. Seed, W. F. Miller, and F. W. Brinkley. TRIDENT: A two-dimensional, multigroup triangular discrete ordinates, explicit neutron transport code. Technical Report LA–5428-M, Los Alamos National Laboratory, 1977.

[43] T. J. Seed, W. F. Miller, and G. E. Bosler. TRIDENT: A new triangular mesh discrete ordinates code. In *Topical Meeting on Advances in Reactor Physics*, Gatlinburg, TN, April 9–12 1978.

[44] John M. McGhee and Todd A. Wareing. Attila version 2: User's manual. Technical Report LA-UR-00-5778, Los Alamos National Laboratory, Los Alamos, NM, 2000.

[45] B. Guo and I. Babuska. The *h-p* version of the finite element method. *Computational Mechanics*, 1:21–41, 203–220, 1986.

[46] L. F. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy. Toward a universal *h-p* adaptive finite element strategy, part I, II and III. *Comput. Meth. Appl. Mech. Engrg.*, 77:79, 1989.

[47] J. Tinsley Oden, Ivo Babuska, and Carlos Erik Baumann. A discontinuous *hp* finite element method for diffusion problems. *J. Comput. Phys.*, 146:491–519, 1998.

[48] C. Schwab and M. Süli. The p and hp versions of the finite element method for problems with boundary layers. *MATHEMATICS OF COMPUTATION*, 65(216):1403–1429, 1996.

[49] Paul Houston, Max Jensen, and Endre Süli. *hp*-discontinuous galerkin finite element methods with least-squares stabilization. *J. Sci. Comput.*, 17(1-4):3–25, 2002.

[50] P. Houston and E. Süli. A note on the design of *hp*-adaptive finite element methods for elliptic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194(2-5):229–243, 2005.

[51] J. P. Jessee, W. A. Fiveland, L. H. Howell, P. Colella, and R. B. Pember. An adaptive mesh refinement algorithm for the radiative transport equation. *J. Comput. Phys.*, 139(2):380–398, 1998.

[52] J. Warsa and A. Prinja. p-adaptive numerical methods for particle transport. *Transp. Theory Stat. Phys.*, 28:29, 1999.

[53] A. Dedner and P. Vollmoller. An adaptive higher order method for solving the radiation transport equation on unstructured grids. *J. Comput. Phys.*, 178(2):263–289, 2002.

[54] H. Zhang and E. E. Lewis. Spatial adaptivity applied to the variational nodal P_N equations. *Nucl. Sci. Eng.*, 142:57, 2002.

[55] J. Ragusa. 3-D adaptive solution of the multigroup diffusion equation on irregular structured grids using a conforming finite element method formulation. In *Proc. PHYSOR 2004*, Chicago, IL, April 25–29, 2004. American Nuclear Society.

[56] Yaqi Wang and Jean Ragusa. Application of *hp*-adaptivity to the multigroup diffusion equations. *Nucl. Sci. Eng.*, 161:1, 2009.

[57] R. Verfurth. *A Review of a posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*. Wiley & Teubner, 1996.

[58] M. Ainsworth and J. T. Oden. *A posteriori Error Estimation in Finite Element Analysis*. John Wiley & Sons, New York, NY, 2000.

[59] Kenneth Eriksson, Don Estep, Peter Hansbo, and Claes Johnson. Introduction to adaptive methods for differential equations. *Acta Numerica*, 4:105–158, 1995.

[60] R. Becker and R. Rannacher. An optimal control approach to error estimation and mesh adaptation in finite element methods. *Acta Numerica*, 10:1–102, 2001.

[61] R.M. Shagaliev, A.V. Alekseev, I.M. Beliakov, A.V. Gichuk, Nuzhdin A.A., and Rezchikov V.Yu. *Different Algorithms of 2D Transport Equation Parallelization on Random Non-Orthogonal Grids*. Lecture Notes in Computational Science and Engineering , Vol. 48. Springer, Berlin, 2006.

[62] F. Ogando and P. Velarde. Development of a radiation transport fluid dynamic code under amr scheme. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 71(2-6):541–550, 2001.

[63] Randal S. Baker. A block adaptive mesh refinement algorithm for the neutral particle transport equation. *Nuclear Science & Engineering*, 141(1):1–12, 2002.

[64] C. Aussourd. A multidimensional AMR S_N scheme. *Nucl. Sci. Engr.*, 143:281–290, 2003.

[65] R. Hartmann. *Adaptive Finite Element Methods for the Compressible Euler Equations*. PhD thesis, University of Heidelberg, 2002.

[66] T. Leicht and R. Hartmann. Anisotropic mesh refinement for discontinuous galerkin methods in two-dimensional aerodynamic flow simulations. *Int. J. Numer. Methods Fluids*, 56(11):2111–2138, 2007.

[67] Marvin L. Adams and Edward W. Larsen. Fast iterative methods for discrete-ordinates particle transport calculations. *Prog. Nucl. Energy*, 40(1):3–159, 2002.

[68] M. L. Adams and W. R. Martin. Diffusion synthetic acceleration of discontinuities finite element transport iterations. *Nucl. Sci. Eng.*, 111:145–167, 1992.

[69] E. W. Larsen. Unconditionally stable diffusion-synthetic acceleration methods for slab geometry discrete ordinates equations. *Nucl. Sci. Eng.*, 82:47, 1982.

[70] James S. Warsa, Todd A. Wareing, and Jim E. Morel. Fully consistent diffusion synthetic acceleration of linear discontinuous S_N transport discretizations on unstructured tetrahedral meshes. *Nucl. Sci. Eng.*, 141:236–251, 2002.

[71] M. D. DeHart. *TRITON: A Two-Dimensional Transport and Depletion Module for Characterization of Spent Nuclear Fuel*. ORNL/TM-2005/39, Version 5.1, Vol. I, Book 3, Section T1, November 2006.

[72] M. D. DeHart. *NEWT: A new transport algorithm for two-dimensional discrete ordinates analysis in non-orthogonal geometries*. ORNL/TM-2005/39, Version 5.1, Vol. II, Book 4, Section F21, November 2006.

[73] Bernardo Cockburn, G. Karniadakis, and C. Shu, editors. *Discontinuous Galerkin Methods: Theory, Computation and Applications, (Lecture Notes in Computational Science and Engineering, volume 11)*. Springer-Verlag, New York, NY, 2000.

[74] Timothy J. Barth. *High-Order Methods for Computational Physics (Lecture Notes in Computational Science and Engineering, volume 9)*. Springer-Verlag, Berlin, 1999.

[75] Nektar code, a Navier-Stokes spectral/*hp* solver. Available at: http://www2.imperial.ac.uk/ssherw/spectralhp/nektar/.

[76] Gerard R. Richter. An optimal-order error estimate for the discontinuous galerkin method. *Mathematics of Computation*, 50(181):75–88, 1988.

[77] N. K. Madsen. Convergence of singular difference application for the discrete ordinate equations in x-y geometry. *Mathematics of Computation*, 26:45–50, 1972.

[78] N. K. Madsen. Convergent centered difference schemes for the discrete ordinate neutron transport equations. *SIAM J. Numer. Anal.*, 12:164–176, 1975.

[79] E. M. Gelbard, J. A. Davis, and L. A. Hageman. Solution of the discrete ordinate equations in one and two dimensiions. In R. Bellman, G. Birckhoff, and I. Abu-Shumays, editors, *Transport Theory*, volume 1, pages 129–158, Providence, R.I., 1969. SIAM-AMS.

[80] K. D. Lathrop. Spatial differencing of the transport equation: Positivity vs. accuracy. *J. Comp. Phys.*, 4:475–498, 1969.

[81] W. H. Reed. New difference equations for the neutron transport equation. *Nucl. Sci. Eng.*, 45:309–314, 1971.

[82] J. Arkuszewski, T. Kulikowska, and J. Mika. Effects of singularities on approximation in S_N methods. *Nucl. Sci. Eng.*, 49:20–26, 1972.

[83] Edward W. Larsen. Spatial convergence properties of the diamond difference method in x, y geometry. *Nucl. Sci. Eng.*, 80:710–713, 1982.

[84] E. W. Larsen and Jr. W. F. Miller. Convergence rates of spatial difference equations for the discrete-ordinates neutron transport equation in slab geometry. *Nucl. Sci. Eng.*, 73:76, 1980.

[85] J. I. Duo and Y. Y. Azmy. Error comparison of diamond difference, nodal, and characteristic methods for solving multi-dimensional transport problems with the discrete ordinates approximate. *Nucl. Sci. Eng.*, 156:139–153, 2007.

[86] I. Babuska, M. Griebel, and J. Pitkaranta. The problem of selecting the shape functions for a p-type finite element. *Internat. J. Numer. Methods Eng.*, na:1891–1908, 1989.

[87] D.C. Carpenter. Two-dimensional finite element neutron diffusion analysis using hierarchic shape functions. In *Proc. Joint Int. Conf. Mathematical Methods and Supercomputing Applications*, Saratoga Springs, New York, 1997. ANS.

[88] P. Houston, C. Schwab, and E. Süli. Stabilized hp-finite element methods for first-order hyperbolic problems. *SIAM J. Numer. Anal.*, 37(5):1618–1643, 2000.

[89] P. F. Fischer, G. W. Kruse, and F. Loth. Spectral element methods for transitional flows in complex geometries. *J. Sci. Comput.*, 17(1-4):81–98, 2002.

[90] P. F. Fischer and E. M. Ronquist. Spectral element methods for large scale parallel Navier-Stokes calculations. *Comp. Meth. Appl. Mech. Engr.*, 116:69–76, 1994.

[91] Jan S. Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Springer, Berlin, 2007.

[92] Leszek Demkowicz. *Computing with hp-Adaptive Finite Elements, Vol. 1: One and Two Dimensional Elliptic and Maxwell Problems*. applied mathematics and nonlinear science series. Chapman & Hall/CRC, Boca Raton, FL, 2007.

[93] Jonathan Richard Shewchuk. *Applied Computational Geometry: Towards Geometric Engineering: Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, volume 1148 of Lecture Notes in Computer Science of *the First ACM Workshop on Applied Computational Geometry*. Springer-Verlag, Berlin, May 1996.

[94] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[95] V. Fraysse, L. Giraud, S. Gratton, and J. Langou. A set of GMRES routines for real and complex arithmetics on high performance computers. Technical Report TR/PA/03/3, CERFACS, 2003.

[96] 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. *Nucl. Sci. Eng.*, 147:218–248, 2004.

[97] H. Khalil. A nodal diffusion technique for synthetic acceleration of nodal S_N calculations. *Nucl. Sci. Eng.*, 90:263, 1985.

[98] A. Kavenoky, J. Stepanek, and F. Schmidt. "Benchmark problems" , transport theory and advanced reactor simulations. Technical Report IAEA-TECDOC-254, IAEA, Vienna, Austria, 1979.

[99] T. Takeda and H. Ikeda. 3-D neutron transport benchmarks. Technical Report NEACRP-L-330, OECD/NEA Committee on Reactor Physics, March 1991.

[100] T. H. Kim and N. Z. Cho. Source projection analytic nodal S_N method for hexagonal geometry. *Ann. Nucl. Energy*, 23(2):133–143, 1996.

[101] Haoliang Lu, Hongchun Wu, Liangzhi Cao, Yongqiang Zhou, Chunyu Xian, and Dong Yao. Two-dimensional nodal transport method for triangular geometry. *Ann. Nucl. Energy*, 34:424–432, 2007.

[102] E. E. Lewis, M. A. Smith, G. Palmiotti, T. A. Taiwo, and N. Tsoul-FANIDIS. Benchmark specification for deterministic 2-D/3-D MOX fuel assembly transport calculations without spatial homogenisation (C5G7 MOX). Technical Report NEA/NSC/DOC(2001)4, OECD/NEA Expert Group on 3-D Radiation Transport Benchmarks, 2001.

[103] M. Smith, E. Lewis, and B. Na. Benchmark on deterministic 2-D MOX fuel assembly transport calculations without spatial homogenization. *Prog. Nucl. Energy*, 45(2-4):107–118, 2004.

[104] R. B. Kellogg. *Numerical analysis of the neutron transport equation.* Numerical Solution of Partial Differential Equations- III, SYNSPADE 1975. Academic Press, San Diego, CA, 1976.

[105] M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82(1):64–84, 1989.

[106] R. E. Alcouffe. Diffusion synthetic acceleration methods for the diamond-differenced discrete-ordinates equations. *Nucl. Sci. Eng.*, 64:344, 1977.

[107] E. W. Larsen. Unconditionally stable diffusion-synthetic acceleration methods for slab geometry discrete ordinates equations. part I: Theory. *Nucl. Sci. Eng.*, 82:47, 1982.

[108] D. R. McCoy and E. W. Larsen. Unconditionally stable diffusion-synthetic acceleration methods for slab geometry discrete ordinates equations. part II: Numerical results. *Nucl. Sci. Eng.*, 82:64, 1982.

[109] Robert C. Ward, Randal S. Baker, and Jim E. Morel. A diffusion synthetic acceleration method for block adaptive mesh refinement. *Nucl. Sci. Eng.*, 152(2):164–179, 2006.

[110] J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei der Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abh. Math. Univ. Hamburg*, 36:9–15, 1971.

[111] M. F. Wheeler. An elliptic collocation finite element method with interior penalties. *SIAM J. Numer. Anal.*, 15:152–161, 1978.

[112] D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM J. Numer. Anal.*, 19:742–760, 1982.

[113] Guido Kanschat. *Discontinuous Galerkin Methods for Viscous Incompressible Flow*. Deutscher Universit?sverlag, Wiesbaden, Germany, 2007.

[114] S. C. Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM J. Sci. Stat. Comput.*, 2(1):1–4, 1981.

[115] Roland W. Freund and Noël M. Nachtigal. QMRPACK: A package of qmr algorithms. *ACM Trans. Math. Softw.*, 22(1):46–77, 1996.

[116] Amina Bouras and Valerie Fraysse. Inexact matrix-vector products in Krylov methods for solving linear systems: A relaxation strategy. *SIAM J. Matrix Anal. Appl.*, 26:660–678, 2005.

[117] Valeria Simoncini and Daniel B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25:454–477, 2003.

[118] E.W. Larsen. Diffusion-synthetic acceleration methods for discrete-ordinates problems. *Transp. Theor. Stat. Phys*, 13:107–26, 1984.

[119] Y. Y. Azmy. Unconditionally stable and robust adjacent-cell diffusive preconditioning of weighed-difference particle transport method is impossible. *J. Comput. Phys.*, 182:213–233, 2002.

[120] M. L. Adams and T. A. Wareing. Diffusion-synthetic acceleration given anisotropic scattering, general quadratures, and multiple dimensiions. *Trans. Am. Nucl. Soc.*, 68:203, 1993.

[121] L. Babuska and W. C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM J. Numer. Anal.*, 15:736–754, 1978.

[122] Ekkehard Ramm, E. Rank, R. Rannacher, K. Schweizerhof, E. Stein, W. Wendland, G. Wittum, Peter Wriggers, Walter Wunderlich, and Erwin Stein. *Error-*

controlled Adaptive Finite Elements in Solid Mechanics. Wiley, Chichester, England, 2003.

[123] Plewa Tomasz, Linde Timur, and V. Gregory Weirs, editors. Lecture Notes in Computational Science and Engineering , Vol. 41. Springer, 2005.

[124] O. C. Zienkiewicz and Z. Zhu R. L. Taylor. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier Butterworth-Heinemann, Burlington, MA, 2005.

[125] Leszek Demkowicz. *Computing with hp-Adaptive Finite Elements, Vol. 1: One and Two Dimensional Elliptic and Maxwell Problems*. applied mathematics and nonlinear science series. Chapman & Hall/CRC, Boca Raton, FL, 2007.

[126] Leszek Demkowicz, Jason Kurtz, David Pardo, Maciej Paszynski, Waldemar Rachowicz, and Adam Zdunek. *Computing with hp-Adaptive Finite Elements, Vol.2: Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications*. applied mathematics and nonlinear science series. Chapman & Hall/CRC, Boca Raton, FL, 2007.

[127] L. H. Howell, R. B. Pember, P. Colella, and J.P. Jessee. A conservative adaptive-mesh algorithm for unsteady, combined-mode heat transfer using the discrete ordinates method. *Numerical Heat Transfer: Fundamentals*, 35:407–430(24), 1 June 1999.

[128] Hervé Jourdren. HERA: A hydrodynamic AMR platform for multi-physics simulations. In Plewa Tomasz, Linde Timur, and V. Gregory Weirs, editors, *Proceedings of the Chicago Workshop on Adaptive Mesh Refinement Methods, Sept. 3-5, 2003*, Lecture Notes in Computational Science and Engineering , Vol. 41, pages 283–294, Berlin, 2005. Springer.

[129] Marsha J Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53(3):484–512, 1984.

[130] R.M. Shagaliev, A.V. Alekseev, I.M. Beliakov, A.V. Gichuk, A.A. Nuzhdin, and V.Yu. Rezchikov. Different algorithms of 2D transport equation parallelization on random non-orthogonal grids. In Frank Graziani, editor, *Computational Methods in Transport, Granlibakken 2004*, Lecture Notes in Computational Science and Engineering , Vol. 48, pages 235–254. Springer, 2006.

[131] S. Richling, E. Meinköhn, N. Kryzhevoi, and G. Kanschat. Radiative transfer with finite elements. *Astron. Astroph.*, 380:776–788, 2001.

[132] Guido Kanschat. *Parallel and Adaptive Galerkin Methods for Radiative Transfer Problems*. PhD thesis, University of Heidelberg, 1996.

[133] A. Klar, J. Lang, and M. Seaïd. Adaptive solutions of SP_N-approximations to radiative heat transfer in glass. *Intern. J. Therm. Sc.*, 44:1013–1023, 2005.

[134] H. Zhang and E. E. Lewis. An adaptive approach to variational nodal diffusion problems. *Nucl. Sci. Engr.*, 137:14–22, 2001.

[135] J. Ragusa. A simple Hessian-based 3D mesh adaptation technique with applications to the multigroup diffusion equations. *Ann. Nucl. Energy*, 35:2006–2018, 2008.

[136] P.J. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49):5068–5082, 2005. Unstructured Mesh Generation.

[137] C. Führer and G. Kanschat. A posteriori error control in radiative transfer. *Computing*, 58(4):317–334, 1997.

[138] R. Hartmann and P. Houston. Adaptive discontinuous Galerkin finite element methods for nonlinear hyperbolic conservation laws. *SIAM J. Sci. Comput.*, 24:979–1004, 2002.

[139] Jpdesr Gago, D. W. Kelly, O. C. Zienkiewicz, and I. Babuska. A posteriori error analysis and adaptive processes in the finite element method: part II d adaptive mesh refinement. *Int. J. Num. Meth. Eng.*, 19:1621–1656, 1983.

[140] W. Bangerth, R. Hartmann, and G. Kanschat. `deal.II` *Differential Equations Analysis Library, Technical Reference*. `http://www.dealii.org`.

[141] Yaqi Wang, Wolfgang Bangerth, and Jean Ragusa. Three-dimensional *h*-adaptivity for the multigroup neutron diffusion equations. *Prog. Nucl. Energy*, 51:543–555, 2008.

[142] Yaqi Wang. *hp*-mesh adaptation for 1-D multi-group neutron diffusion problems. Master's thesis, Texas A&M University, 2006.

[143] G. Karypis and V. Kumar. METIS a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices version 4.0. *University of Minnesota, Department of Comp*, 1998.

[144] S. Plimpton, B. Hendrickson, S. Burns, and W. McLendon III. Parallel algorithms for radiation transport on unstructured grids. In *Supercomputing, ACM/IEEE 2000 Conference*, pages 25–25, 2000.

[145] E. Seegyoung Seol and Mark S. Shephard. Efficient distributed mesh data structure for parallel automated adaptive analysis. *Eng. with Comput.*, 22(3):197–213, 2006.

[146] James A. Davis. Variational vacuum boundary condition for a P_N approximation. *Nucl. Sci. Eng.*, 25:189–197, 1966.

APPENDIX A

DIFFERENT FORMS FOR THE STEADY-STATE ENERGY-DEPENDENT NEUTRON TRANSPORT EQUATION

A. Partial Differential Equations with Boundary Conditions

The steady-state energy-dependent neutron transport equation with $\vec{r} \in \mathcal{D}$, $\vec{\Omega} \in S^2$, $E \in \mathbb{R}^+$ is given by,

$$\left(\vec{\Omega}\cdot\vec{\nabla}+\sigma_t\right)\Psi = S_{ext}+\frac{\chi}{4\pi}\int_0^\infty \nu\sigma_f(E')\Phi(E')dE'+\int_0^\infty\int_{4\pi}\sigma_s(E'\to E,\vec{\Omega}'\cdot\vec{\Omega})\Psi(\vec{\Omega}',E')d\Omega'dE' \tag{A.1}$$

with the general boundary condition

$$\Psi(\vec{r}_b,\vec{\Omega},E) = \Psi^{inc}(\vec{r}_b,\vec{\Omega},E) + \int_0^\infty\int_{\vec{\Omega}'\cdot\vec{n}_b>0}\beta(\vec{r}_b,E'\to E,\vec{\Omega}'\to\vec{\Omega})\Psi(\vec{r}_b,\vec{\Omega}',E')d\Omega'dE' \tag{A.2}$$

on $\vec{r}_b \in \partial\mathcal{D}$, $E \in \mathbb{R}^+$ and $\vec{\Omega}\cdot\vec{n}_b < 0$

Symbols used in the equation are standard in text, their meanings are listed below:

$\vec{r}$	position variable [cm]
$\mathcal{D}$	$\in \mathbb{R}^d$ open convex space domain, d is the spatial dimension
$\partial\mathcal{D}$	boundary of spatial domain
$\vec{n}_b$	$= \vec{n}(\vec{r})$outward unit normal vector on the boundary
$\vec{\Omega}$	angular variable
S^2	2-dimensional unit sphere
E	energy [MeV], usually in range of [0, 20] MeV
$\mathbb{R}^+$	set of positive real number
$\Psi(\vec{r}, \vec{\Omega}, E)$	$= n(\vec{r}, \vec{\Omega}, E)v$ neutron density in phase space times speed also called neutron angular flux $[\frac{n}{cm^2 \cdot MeV \cdot ster \cdot s}]$
$\Phi(\vec{r}, E)$	$= \int_{4\pi} \Psi d\Omega$ neutron scalar flux $[\frac{n}{cm^2 \cdot MeV \cdot s}]$
$S_{ext}(\vec{r}, \vec{\Omega}, E)$	external source $[\frac{n}{cm^2 \cdot MeV \cdot ster \cdot s}]$
$\sigma_t(\vec{r}, E)$	macroscopic total cross section $[cm^{-1}]$
$\sigma_s(\vec{r}, E' \to E, \vec{\Omega}' \cdot \vec{\Omega})$	differential scattering cross section depending only on the cosine of scattering angles $[\frac{1}{cm \cdot MeV \cdot ster}]$
$\sigma_f(\vec{r}, E)$	fission cross section $[cm^{-1}]$
$\chi(\vec{r}, E)$	neutron fission spectrum$[\frac{1}{MeV}]$
$\nu(\vec{r}, E)$	average number of neutrons emitted per fission
$\Psi^{inc}(\vec{r}_b, \vec{\Omega}, E)$	incoming angular flux on the boundary $[\frac{n}{cm^2 \cdot MeV \cdot ster \cdot s}]$
$\beta(\vec{r}_b, E' \to E, \vec{\Omega}' \to \vec{\Omega})$	boundary albedo $[\frac{1}{MeV \cdot ster}]$

When $\Psi^{inc}(\vec{r}_b, \vec{\Omega}, E)$ is equal to zero, the boundary condition is homogeneous:

$$\Psi(\vec{r}_b, \vec{\Omega}, E) = \int_0^\infty \int_{\vec{\Omega}' \cdot \vec{n}_b > 0} \beta(\vec{r}_b, E' \to E, \vec{\Omega}' \to \vec{\Omega}) \Psi(\vec{r}_b, \vec{\Omega}', E') d\Omega' dE' \tag{A.3}$$

$$\text{on } \vec{r}_b \in \partial\mathcal{D},\ E \in \mathbb{R}^+ \text{ and } \vec{\Omega} \cdot \vec{n}_b < 0$$

In addition, when β is zero, boundary conditions are vacuum.

For reflective boundary conditions, we have $\Psi^{inc}(\vec{r}_b, \vec{\Omega}, E) = 0$ and

$$\beta(\vec{r}_b, E' \to E, \vec{\Omega}' \to \vec{\Omega}) = \delta(E' - E)\delta(\vec{\Omega}' - \vec{\Omega}_r) \tag{A.4}$$

where

$$\vec{\Omega}_r = \vec{\Omega} - 2(\vec{\Omega} \cdot \vec{n}_b(\vec{r}_b))\vec{n}_b(\vec{r}_b). \tag{A.5}$$

Define following function space (not very strict here):

$$W \equiv \mathcal{D} \cup \mathbb{R}^+ \cup S^2 = \left\{ \Psi(\vec{r}, \vec{\Omega}, E) | \vec{r} \in \mathcal{D},\ E \in \mathbb{R}^+ \text{ and } \vec{\Omega} \in S^2 \right\} \tag{A.6}$$

$$W^+ \equiv \partial\mathcal{D} \cup \mathbb{R}^+ \cup S^{2+} = \left\{ \Psi(\vec{r}_b, \vec{\Omega}, E) | \vec{r}_b \in \partial\mathcal{D},\ E \in \mathbb{R}^+ \text{ and } \vec{\Omega} \cdot \vec{n}_b > 0 \right\} \tag{A.7}$$

$$W^- \equiv \partial\mathcal{D} \cup \mathbb{R}^+ \cup S^{2-} = \left\{ \Psi(\vec{r}_b, \vec{\Omega}, E) | \vec{r}_b \in \partial\mathcal{D},\ E \in \mathbb{R}^+ \text{ and } \vec{\Omega} \cdot \vec{n}_b < 0 \right\} \tag{A.8}$$

and the following linear operators,

$$L\Psi \equiv \left(\vec{\Omega} \cdot \vec{\nabla} + \sigma_t(\vec{r}, E)\right)\Psi(\vec{r}, \vec{\Omega}, E) \tag{A.9}$$

$$P\Psi \equiv \frac{\chi(\vec{r}, E)}{4\pi} \int_0^\infty \nu\sigma_f(\vec{r}, E') \left[\int_{4\pi} \Psi(\vec{r}, \vec{\Omega}', E') d\Omega' \right] dE' \tag{A.10}$$

$$H\Psi \equiv \int_0^\infty \int_{4\pi} \sigma_s(\vec{r}, E' \to E, \vec{\Omega}' \cdot \vec{\Omega}) \Psi(\vec{r}, \vec{\Omega}', E') d\Omega' dE' \tag{A.11}$$

$$B\Psi^+ \equiv \int_0^\infty \int_{\vec{\Omega}' \cdot \vec{n}_b > 0} \beta(\vec{r}_b, E' \to E, \vec{\Omega}' \to \vec{\Omega}) \Psi(\vec{r}_b, \vec{\Omega}', E') d\Omega' dE' \tag{A.12}$$

$$\text{on } \vec{r}_b \in \partial\mathcal{D} \text{ and } \vec{\Omega} \cdot \vec{n}_b < 0$$

Ψ is a function defined on the solution space W. L, P, and H are the streaming-collision, fission production and scattering operators respectively. Ψ^+ is the function defined in the space W^+. We can think it as the result of trace operation on Ψ. Similarly, Ψ^- is the result of trace operation on Ψ. B operator maps Ψ^+, all angular fluxes for outgoing directions and all energies on the boundary, to a function in the

space W^-. The transport equation Eq. (A.1) can be written into a shorter form

$$L\Psi = S_{ext} + H\Psi + P\Psi \tag{A.13}$$

with boundary condition

$$\Psi^- = \Psi^{inc} + B\Psi^+ \tag{A.14}$$

Note: Ψ^{inc} is in the space W^-.

If we define following correspondingly linear adjoint operators,

$$P^*\Psi^* \equiv \frac{\nu\sigma_f(\vec{r},E)}{4\pi}\int_0^\infty \chi(\vec{r},E')\left[\int_{4\pi}\Psi^*(\vec{r},\vec{\Omega}',E')d\Omega'\right]dE' \tag{A.15}$$

$$H^*\Psi^* \equiv \int_0^\infty\int_{4\pi}\sigma_s(\vec{r},E\rightarrow E',\vec{\Omega}'\cdot\vec{\Omega})\Psi^*(\vec{r},\vec{\Omega}',E')d\Omega' dE' \tag{A.16}$$

$$L^*\Psi^* \equiv \big(-\vec{\Omega}\cdot\vec{\nabla}+\sigma_t(\vec{r},E)\big)\Psi^*(\vec{r},\vec{\Omega},E) \tag{A.17}$$

$$B^*\Psi^- \equiv \int_0^\infty\int_{\vec{\Omega}'\cdot\vec{n}_b<0}\beta(\vec{r}_b,E\rightarrow E',\vec{\Omega}\rightarrow\vec{\Omega}')\Psi(\vec{r}_b,\vec{\Omega}',E')d\Omega' dE' \tag{A.18}$$

$$\text{on } \vec{r}_b\in\partial\mathcal{D} \text{ and } \vec{\Omega}\cdot\vec{n}_b>0$$

we can write the adjoint equation into a simpler form

$$L^*\Psi^* = S^*_{ext} + H^*\Psi^* + P^*\Psi^* \tag{A.19}$$

with the general boundary condition

$$\Psi^{*+} = \Psi^{*out} + B^*\Psi^{*-} \tag{A.20}$$

Ψ^{*out} is in the space W^+.

B. Integral Equations

By inverting the streaming-collision operator, we obtain the integral equation for angular flux

$$\begin{aligned}\Psi(\vec{r},E,\vec{\Omega}) =&\Psi(\vec{r}-\tau\vec{\Omega},E,\vec{\Omega})e^{-\int_0^\tau \sigma_t(\vec{r}-\tau'\vec{\Omega})d\tau'}+\\ &\int_0^\tau ds\left[(H+P)\Psi+S_{ext}\right](\vec{r}-s\vec{\Omega})e^{-\alpha(\vec{r}-s\vec{\Omega},\vec{r},E)}\end{aligned} \tag{A.21}$$

where the optical distance between $\vec{r}_1$ and $\vec{r}_2$ (number of MFP) is

$$\alpha(\vec{r}_1,\vec{r}_2,E)=\int_0^{|\vec{r}_1-\vec{r}_2|} ds\,\sigma_t(\vec{r}+s\vec{e},E) \tag{A.22}$$

with

$$\vec{e}=\frac{\vec{r}_2-\vec{r}_1}{|\vec{r}_2-\vec{r}_1|} \tag{A.23}$$

Note: $\vec{r}-\tau\vec{\Omega}\in\partial\mathcal{D}$. The integral form of the transport equation has a clear physical interpretation and is the basis of a family of computational transport methods called characteristic methods.

If we have following assumptions:

1. vacuum boundary (no incident flux)

2. isotropic total source (isotropic scattering and isotropic external source)

We then integrate the integral equation over all directions

$$\Phi(\vec{r},E)=\int_{\mathcal{D}} d\vec{r}'\left[(H+P)\Phi+S_{ext}\right](\vec{r}',E)\frac{e^{-\alpha(\vec{r}',\vec{r},E)}}{4\pi\left|\vec{r}'-\vec{r}\right|^2} \tag{A.24}$$

This is the integral equation for the scalar flux, which is all called Peierl's equation. Discretizing this equation directly results into a linear system with the full matrix.

C. Variational Form of the PDE

We can easily prove that,

$$(P\Psi, \Psi^*) = (\Psi, P^*\Psi^*) \tag{A.25}$$

$$(H\Psi, \Psi^*) = (\Psi, H^*\Psi^*) \tag{A.26}$$

$$(L\Psi, \Psi^*) + \langle \Psi, \Psi^* \rangle^- = (\Psi, L^*\Psi^*) + \langle \Psi, \Psi^* \rangle^+ \tag{A.27}$$

$$\left\langle B\Psi^+, \Psi^* \right\rangle^- = \left\langle \Psi, B^*\Psi^{*-} \right\rangle^+ \tag{A.28}$$

with,

$$\langle \Psi, \Psi^* \rangle^- \equiv \int_{\partial\mathcal{D}} ds \int_0^\infty dE \int_{\vec{\Omega}\cdot\vec{n}_b<0} d\Omega \, |\vec{\Omega} \cdot \vec{n}_b(\vec{r}_b)| \Psi^*(\vec{r}_b, \vec{\Omega}, E)\Psi(\vec{r}_b, \vec{\Omega}, E) \tag{A.29}$$

$$\langle \Psi, \Psi^* \rangle^+ \equiv \int_{\partial\mathcal{D}} ds \int_0^\infty dE \int_{\vec{\Omega}\cdot\vec{n}_b>0} d\Omega \, |\vec{\Omega} \cdot \vec{n}_b(\vec{r}_b)| \Psi^*(\vec{r}_b, \vec{\Omega}, E)\Psi(\vec{r}_b, \vec{\Omega}, E) \tag{A.30}$$

Multiply the transport equation with Ψ^*, and integrate both sides over the phase space, and substitute the boundary condition we obtain

$$(\Psi, L^*\Psi^*) + \langle \Psi, \Psi^* \rangle^+ - \left\langle B\Psi^+, \Psi^* \right\rangle^- - (H\Psi, \Psi^*) - (P\Psi, \Psi^*) = (S_{ext}, \Psi^*) + \left\langle \Psi^{inc}, \Psi^* \right\rangle^- \tag{A.31}$$

Similarly we can obtain the equation for the adjoint equation

$$(L\Psi, \Psi^*) + \langle \Psi, \Psi^* \rangle^- - \left\langle \Psi, B^*\Psi^{*-} \right\rangle^+ - (\Psi, H^*\Psi^*) - (\Psi, P^*\Psi^*) = (\Psi, S^*_{ext}) + \left\langle \Psi, \Psi^{*out} \right\rangle^+ \tag{A.32}$$

Define

$$b(\Psi,\Psi^*) \equiv (\Psi, L^*\Psi^*) + \langle \Psi,\Psi^*\rangle^+ - \left\langle B\Psi^+,\Psi^*\right\rangle^- - (H\Psi,\Psi^*) - (P\Psi,\Psi^*) \quad \text{(A.33)}$$

$$R(\Psi^*) \equiv (S_{ext},\Psi^*) + \left\langle \Psi^{inc},\Psi^*\right\rangle^- \quad \text{(A.34)}$$

$$b^*(\Psi,\Psi^*) \equiv (L\Psi,\Psi^*) + \langle \Psi,\Psi^*\rangle^- - \left\langle \Psi, B^*\Psi^{*-}\right\rangle^+ - (\Psi,H^*\Psi^*) - (\Psi,P^*\Psi^*) \quad \text{(A.35)}$$

$$R^*(\Psi) \equiv (\Psi, S^*_{ext}) + \left\langle \Psi,\Psi^{*out}\right\rangle^+ \quad \text{(A.36)}$$

We immediately notice that

$$b(\Psi,\Psi^*) \equiv b^*(\Psi,\Psi^*) \quad \text{(A.37)}$$

The variational form of the transport equation is:

Find $\Psi \in W$, such that

$$b(\Psi,\Psi^*) = R(\Psi^*) \qquad \forall\Psi^* \in W \quad \text{(A.38)}$$

The variational form of the adjoint transport equation is:

Find $\Psi^* \in W$, such that

$$b(\Psi,\Psi^*) = R^*(\Psi) \qquad \forall\Psi \in W \quad \text{(A.39)}$$

[If there is no energy dependence, i.e. the equation is one-group, we can use Legendre polynomial to expand the differential scattering cross section and expand the angular flux with the spherical harmonics. And because $\sigma_{s,n} \le \sigma_{s,0} < \sigma_t, n = 1,\ldots,\infty$, $(\Psi,\sigma_t\Psi) > (H\Psi,\Psi)$. So if there is no fission, $b(\Psi,\Psi) > 0$. The function space W is not cared much by our engineers. We can simply think that functions in W are square integrable and the operation of $\vec{\Omega}\cdot\vec{\nabla}$ is meaningful.] Apparently, $b(\Psi,\Psi^*)$ is not symmetric.

We can define the functional

$$F(\Psi, \Psi^*) = R(\Psi^*) + R^*(\Psi) - b(\Psi, \Psi^*) \tag{A.40}$$

whose stationary points are the solutions of the normal and adjoint transport equation.

D. Parity Form

By changing variable $\vec{\Omega}$ to $-\vec{\Omega}$, we obtain

$$-\vec{\Omega} \cdot \vec{\nabla}\Psi(\vec{r}, -\vec{\Omega}, E) + \sigma_t \Psi(\vec{r}, -\vec{\Omega}, E) = S_{ext}(\vec{r}, -\vec{\Omega}, E) + H_- \Psi + P\Psi \tag{A.41}$$

where

$$H_- \Psi \equiv \int_0^\infty \int_{4\pi} \sigma_s(\vec{r}, E' \to E, -\vec{\Omega}' \cdot \vec{\Omega}) \Psi(\vec{r}, \vec{\Omega}', E') d\Omega' dE' \tag{A.42}$$

Define

$$\widehat{\Psi}(\vec{r}, \vec{\Omega}, E) = \Psi(\vec{r}, -\vec{\Omega}, E) \tag{A.43}$$

sum the equation with Eq. (A.13) we get

$$\vec{\Omega} \cdot \vec{\nabla}\Psi_o + \sigma_t \Psi_e = S_{ext,e} + H_e \Psi_e + P\Psi_e \tag{A.44}$$

Note that

$$\Psi_e = \frac{1}{2}(\Psi + \widehat{\Psi}) \tag{A.45}$$

$$\Psi_o = \frac{1}{2}(\Psi - \widehat{\Psi}) \tag{A.46}$$

$$H_e = \frac{1}{2}(H + H_-) \tag{A.47}$$

$$H_e \Psi_o = 0 \tag{A.48}$$

$$P\Psi_o = 0 \tag{A.49}$$

Subtract the equation with Eq. (A.13) we get

$$\vec{\Omega}\cdot\vec{\nabla}\Psi_e+\sigma_t\Psi_o=S_{ext,o}+H_o\Psi_o \tag{A.50}$$

Note that

$$H_o=\frac{1}{2}(H-H_-) \tag{A.51}$$

$$H_o\Psi_e=0 \tag{A.52}$$

Boundary condition is cast into

$$\Psi_e-\Psi_o=0 \ \text{ on } \ W^+ \tag{A.53}$$

$$\Psi_e+\Psi_o=0 \ \text{ on } \ W^- \tag{A.54}$$

To make life easier, vacuum boundary conditions are used hereafter. Transform the Eq. (A.50)

$$\Psi_o=\frac{1}{\sigma_t}\left[S_{ext,o}+H_o\Psi_o-\vec{\Omega}\cdot\vec{\nabla}\Psi_e\right]$$

and substitute it into Eq. (A.44), we obtain

$$-\vec{\nabla}\cdot\left[\frac{1}{\sigma_t}\vec{\Omega}\vec{\Omega}\cdot\vec{\nabla}\Psi_e\right]+\sigma_t\Psi_e=S_{ext,e}+H_e\Psi_e+P\Psi_e-\vec{\Omega}\cdot\vec{\nabla}\left[\frac{1}{\sigma_t}S_{ext,o}+\frac{1}{\sigma_t}H_o\Psi_o\right] \tag{A.55}$$

Similarly we can obtain

$$-\vec{\nabla}\cdot\left[\frac{1}{\sigma_t}\vec{\Omega}\vec{\Omega}\cdot\vec{\nabla}\Psi_o\right]+\sigma_t\Psi_o=S_{ext,o}+H_o\Psi_o-\vec{\Omega}\cdot\vec{\nabla}\left[\frac{1}{\sigma_t}S_{ext,e}+\frac{1}{\sigma_t}H_e\Psi_e+\frac{1}{\sigma_t}P\Psi_e\right] \tag{A.56}$$

sum the two equations,

$$-\vec{\nabla}\cdot\left[\frac{1}{\sigma_t}\vec{\Omega}\vec{\Omega}\cdot\vec{\nabla}\Psi\right]+\sigma_t\Psi=(1-\frac{1}{\sigma_t}\vec{\Omega}\cdot\vec{\nabla})\left[S_{ext,e}+H\Psi+P\Psi\right] \tag{A.57}$$

This is the SAAF (Self-Adjoint Angular Flux equation). Boundary conditions for this form is not quite clear. Because the coefficient matrix $\frac{1}{\sigma_t}\vec{\Omega}\vec{\Omega}$ is singular, so the

actual system is not elliptic, values can not be specified on all the boundaries.

We can also write down the parity form for the adjoint equation

$$-\vec{\Omega}\cdot\vec{\nabla}\Psi_o^* + \sigma_t\Psi_e^* = S_{ext,e}^* + H_e^*\Psi_e^* + P\Psi_{*e}^* \tag{A.58}$$

$$-\vec{\Omega}\cdot\vec{\nabla}\Psi_e^* + \sigma_t\Psi_o^* = S_{ext,o}^* + H_o^*\Psi_o^* \tag{A.59}$$

Boundary condition is cast into

$$\Psi_e^* + \Psi_o^* = 0 \quad \text{on} \quad W^+ \tag{A.60}$$

$$\Psi_e^* - \Psi_o^* = 0 \quad \text{on} \quad W^- \tag{A.61}$$

E. Parity Variational Form

To get the bilinear form, we multiply the normal parities with the adjoint parity equations

$$(\vec{\Omega}\cdot\vec{\nabla}\Psi_e, \Psi_o^*) + (\sigma_t\Psi_e, \Psi_e^*) - \boxed{\langle\Psi_e, \Psi_o^*\rangle^+} + \boxed{\langle\Psi_e, \Psi_o^*\rangle^-} = (\Psi_e, S_{ext,e}^*) + (\Psi_e, H_e^*\Psi_e^* + P^*\Psi_e^*) \tag{A.62}$$

$$(\vec{\Omega}\cdot\vec{\nabla}\Psi_o, \Psi_e^*) + (\sigma_t\Psi_o, \Psi_o^*) - \boxed{\boxed{\langle\Psi_o, \Psi_e^*\rangle^+}} + \langle\Psi_o, \Psi_e^*\rangle^- = (\Psi_o, S_{ext,o}^*) + (\Psi_o, H_o^*\Psi_o^*) \tag{A.63}$$

Apply boundary conditions on the two single-boxed terms to remove the adjoint odd parity in the boundary integrals,

$$(\vec{\Omega}\cdot\vec{\nabla}\Psi_e,\Psi_o^*)+(\sigma_t\Psi_e,\Psi_e^*)+\langle\Psi_e,\Psi_e^*\rangle^+ +\langle\Psi_e,\Psi_e^*\rangle^- -(\Psi_e,H_e^*\Psi_e^*+P^*\Psi_e^*)=(\Psi_e,S_{ext,e}^*) \tag{A.64}$$

$$(\vec{\Omega}\cdot\vec{\nabla}\Psi_o,\Psi_e^*)+(\sigma_t\Psi_o,\Psi_o^*)-\langle\Psi_o,\Psi_e^*\rangle^+ +\langle\Psi_o,\Psi_e^*\rangle^- -(\Psi_o,H_o^*\Psi_o^*)=(\Psi_o,S_{ext,o}^*) \tag{A.65}$$

Sum these two together, we obtain the variational form:

Find $\Psi_e^*\in W_e$ and $\Psi_o^*\in W_o$, such that

$$b^*(\Psi_e,\Psi_o,\Psi_e^*,\Psi_o^*)=R^*(\Psi_e,\Psi_o)\quad\forall\Psi_e\in W_e \text{ and } \Psi_o\in W_o \tag{A.66}$$

where

$$\begin{aligned} b^*(\Psi_e^*,\Psi_o^*,\Psi_e,\Psi_o)=&(\vec{\Omega}\cdot\vec{\nabla}\Psi_e,\Psi_o^*)+(\sigma_t\Psi_e,\Psi_e^*)-(\Psi_e,H_e^*\Psi_e^*+P^*\Psi_e^*)+\\ &(\vec{\Omega}\cdot\vec{\nabla}\Psi_o,\Psi_e^*)+(\sigma_t\Psi_o,\Psi_o^*)-(\Psi_o,H_o^*\Psi_o^*)+\\ &\langle\Psi_e-\Psi_o,\Psi_e^*\rangle^+ +\langle\Psi_e+\Psi_o,\Psi_e^*\rangle^- \end{aligned} \tag{A.67}$$

$$R^*(\Psi_e,\Psi_o)=(\Psi_e,S_{ext,e}^*)+(\Psi_o,S_{ext,o}^*) \tag{A.68}$$

Note: This bilinear form is not equal to the bilinear form of the one in the Sec. C. If we choose apply boundary condition on the double-boxed term, then we get the exactly same form.

We multiply the adjoint parities with the parity equations

$$(\Psi_o,-\vec{\Omega}\cdot\vec{\nabla}\Psi_e^*)+(\sigma_t\Psi_e,\Psi_e^*)+\boxed{\langle\Psi_o,\Psi_e^*\rangle^+}-\boxed{\langle\Psi_o,\Psi_e^*\rangle^-}=(S_{ext,e},\Psi_e^*)+(H_e\Psi_e+P\Psi_e,\Psi_e^*) \tag{A.69}$$

$$(\Psi_e,-\vec{\Omega}\cdot\vec{\nabla}\Psi_o^*)+(\sigma_t\Psi_o,\Psi_o^*)+\boxed{\boxed{\langle\Psi_e,\Psi_o^*\rangle^+}}-\langle\Psi_e,\Psi_o^*\rangle^-=(S_{ext,o},\Psi_o^*)+(H_o\Psi_o,\Psi_o^*) \tag{A.70}$$

Apply boundary conditions to remove the odd parity in the boundary integrals,

$$(\Psi_o, -\vec{\Omega}\cdot\vec{\nabla}\Psi_e^*) + (\sigma_t\Psi_e, \Psi_e^*) + \langle\Psi_e, \Psi_e^*\rangle^+ + \langle\Psi_e, \Psi_e^*\rangle^- = (S_{ext,e}, \Psi_e^*) + (H_e\Psi_e + P\Psi_e, \Psi_e^*) \tag{A.71}$$

$$(\Psi_e, -\vec{\Omega}\cdot\vec{\nabla}\Psi_o^*) + (\sigma_t\Psi_o, \Psi_o^*) + \langle\Psi_e, \Psi_o^*\rangle^+ - \langle\Psi_e, \Psi_o^*\rangle^- = (S_{ext,o}, \Psi_o^*) + (H_o\Psi_o, \Psi_o^*) \tag{A.72}$$

Sum these two together, we obtain the variational form:

Find $\Psi_e \in W_e$ and $\Psi_o \in W_o$, such that

$$b(\Psi_e, \Psi_o, \Psi_e^*, \Psi_o^*) = R(\Psi_e^*, \Psi_o^*) \quad \forall\Psi_e^* \in W_e \text{ and } \Psi_o^* \in W_o \tag{A.73}$$

where

$$\begin{aligned} b(\Psi_e^*, \Psi_o^*, \Psi_e, \Psi_o) =& (\Psi_o, -\vec{\Omega}\cdot\vec{\nabla}\Psi_e^*) + (\sigma_t\Psi_e, \Psi_e^*) - (H_e\Psi_e + P\Psi_e, \Psi_e^*) + \\ & (\Psi_e, -\vec{\Omega}\cdot\vec{\nabla}\Psi_o^*) + (\sigma_t\Psi_o, \Psi_o^*) - (H_o\Psi_o, \Psi_o^*) + \\ & \langle\Psi_e, \Psi_e^* + \Psi_o^*\rangle^+ + \langle\Psi_e, \Psi_e^* - \Psi_o^*\rangle^- \end{aligned} \tag{A.74}$$

$$R(\Psi_e^*, \Psi_o^*) = (S_{ext,e}, \Psi_e^*) + (S_{ext,o}, \Psi_o^*) \tag{A.75}$$

Again, we can easily prove that

$$b(\Psi_e, \Psi_o, \Psi_e^*, \Psi_o^*) = b^*(\Psi_e, \Psi_o, \Psi_e^*, \Psi_o^*) \tag{A.76}$$

We can write the bilinear form in form of normal and adjoint angular fluxes

$$\begin{aligned} b(\Psi, \Psi^*) =& (\Psi, L^*\Psi^*) - (H\Psi + P\Psi, \Psi^*) + \\ & \frac{1}{2}\left\langle \Psi + \widehat{\Psi}, \Psi^* \right\rangle^+ + \frac{1}{2}\left\langle \Psi + \widehat{\Psi}, \widehat{\Psi}^* \right\rangle^- \end{aligned} \tag{A.77}$$

Note that this form is important to derive the Marshark boundary condition for the P_N equations [146].

APPENDIX B

FORMS FOR THE SIMPLE TRANSPORT EQUATION

A. Variational Form for the Simple Transport Equation

Differential equation of the simple transport equation

$$\vec{\Omega} \cdot \vec{\nabla}\psi + \sigma\psi(\vec{r}) = S(\vec{r}) \tag{B.1}$$

$\vec{\Omega}$ is a constant vector in our study. With boundary condition

$$\psi(\vec{r}_b) = \psi^{inc}(\vec{r}_b) \quad \text{on} \quad \partial\mathcal{D}^- = \left\{\vec{r}_b \in \partial\mathcal{D}, \vec{\Omega} \cdot \vec{n}(\vec{r}_b) < 0\right\} \tag{B.2}$$

$\vec{n}$ is the normal unit outward vector on the domain boundary. Its variational form:

Find $\psi \in W_{\mathcal{D}}$, such that

$$b_{\vec{\Omega}}(\psi, \psi^*) = R(\psi^*) \qquad \forall\psi^* \in W_{\mathcal{D}} \tag{B.3}$$

where

$$b_{\vec{\Omega}}(\psi, \psi^*) \equiv (\vec{\Omega} \cdot \vec{\nabla}\psi + \sigma\psi, \psi^*)_{\mathcal{D}} + \langle\psi, \psi^*\rangle^- \tag{B.4}$$

$$R(\psi^*) \equiv (S, \psi^*)_{\mathcal{D}} + \left\langle\psi^{inc}, \psi^*\right\rangle^- \tag{B.5}$$

$$W_D = \left\{\psi \in L^2(\mathcal{D}) | \vec{\Omega} \cdot \vec{\nabla}\psi \in L^2(\mathcal{D})\right\} \tag{B.6}$$

$$(f, g)_{\mathcal{D}} = \int_{\mathcal{D}} f\ g\ d\vec{r} \tag{B.7}$$

$$\langle f, g\rangle^- = \int_{\partial\mathcal{D}^-} |\vec{\Omega} \cdot \vec{n}_b| f\ g\ ds \tag{B.8}$$

Variational form for the adjoint equation

Find $\psi^* \in W_{\mathcal{D}}$, such that

$$b^*_{\vec{\Omega}}(\psi, \psi^*) = R^*(\psi) \qquad \forall\psi \in W_{\mathcal{D}} \tag{B.9}$$

where

$$b^*_{\vec{\Omega}}(\psi,\psi^*) \equiv (\psi, -\vec{\Omega}\cdot\vec{\nabla}\psi^* + \sigma\psi^*)_{\mathcal{D}} + \langle\psi,\psi^*\rangle^+ \tag{B.10}$$

$$R^*(\psi) \equiv (\psi, S^*)_{\mathcal{D}} + \left\langle\psi,\psi^{*out}\right\rangle^+ \tag{B.11}$$

$$\langle f,g\rangle^+ = \int_{\partial\mathcal{D}^+} |\vec{\Omega}\cdot\vec{n}_b| f\ g ds \tag{B.12}$$

$$\partial\mathcal{D}^+ = \left\{\vec{r}_b \in \partial\mathcal{D},\ \vec{\Omega}\cdot\vec{n}_b(\vec{r}_b) > 0\right\} \tag{B.13}$$

We can prove:

$$b_{\vec{\Omega}}(\psi,\psi^*) = b^*_{\vec{\Omega}}(\psi,\psi^*) \tag{B.14}$$

$$b_{\vec{\Omega}}(\psi,\psi^*) \neq b_{\vec{\Omega}}(\psi^*,\psi) \tag{B.15}$$

$$b_{\vec{\Omega}}(\psi,\psi) = (\sigma\psi,\psi) + \frac{\langle\psi,\psi\rangle^+ + \langle\psi,\psi\rangle^-}{2} \geq 0 \tag{B.16}$$

The bilinear form is not symmetric but positive definite. Transport solution is the stationary point of the following functional

$$F(\psi,\psi^*) = R(\psi^*) + R^*(\psi) - b_{\vec{\Omega}}(\psi,\psi^*) \tag{B.17}$$

B. DGFEM for the Simple Transport Equation

We then consider the spatial discretization with the *hp*-version of the Discontinuous Galerkin Finite Element Method (DGFEM). Let $\mathbb{T}_h$ be a subdivision of $\mathcal{D}$ into disjoint open elements K such that $\mathcal{D} = \cup_{K\in\mathbb{T}_h}$, where $\mathbb{T}_h$ could be multi-irregular, i.e. an element may have more than one or two neighbors on one of its side. We assume that all elements are shape-regular i.e. they are affine images of a fixed master element $\widehat{K}$, $K = F_K(\widehat{K})$. If mixed types of elements are used, $\widehat{K}$ is either the unit simplex or the open unit hypercube in $\mathbb{R}^d$. The number of local faces varies with the type of

element: 3 for 2-D triangle, 4 for 2-D quadrilateral and for 3-D tetrahedron, 6 for 3-D hexahedron. For a nonnegative integer p, we denote by $P_p(\widehat{K})$ the set of polynomials of total degree p on $\widehat{K}$ if the $\widehat{K}$ is a simplex. We use $Q_p(\widehat{K})$ to denote the function space of all tensor-product polynomials on hypercubes in all coordinate direction. We define the local polynomial function space

$$V_p(K) = \left\{ \psi \in L^2(K) \;\middle|\; \begin{array}{ll} \psi \circ F_K \in P_p(\widehat{K}) & \text{if } K \text{ is a simplex} \\ \psi \circ F_K \in Q_p(\widehat{K}) & \text{if } K \text{ is a hypercube} \end{array} \right\} \tag{B.18}$$

More types of elements can be used. We then define the finite element space $W_{\mathcal{D}}^h(\mathcal{D}, \mathbb{T}_h, \mathbf{p}) = \left\{ \psi \in L^2(D) | \psi_{|K} \in V_{p_K}(K) \right\}$, where $\mathbf{p}$ is a vector $\{p_K | K \in T_h\}$ with the dimension being the total number of elements. Here the superscript h means the space is finite-dimensional with the discretization.

Multiply the transport equation with $\forall \psi^* \in V_p(K)$ and integrate over one element K, with integration by parts we get

$$(\psi, -\vec{\Omega} \cdot \vec{\nabla} \psi^*)_K + \boxed{\left\langle \psi^n, \psi^{*-} \right\rangle_{\partial K^+} - \left\langle \psi^n, \psi^{*+} \right\rangle_{\partial K^-}} + (\sigma\psi, \psi^*)_K = (S, \psi^*)_K \tag{B.19}$$

Values on the two sides of faces could be different. To avoid ambiguity we define:

$$\begin{aligned} f^+ &= \lim_{s \to 0^+} f(\vec{r} + s\vec{\Omega}) \\ f^- &= \lim_{s \to 0^-} f(\vec{r} + s\vec{\Omega}) \end{aligned} \tag{B.20}$$

After applying *the upwind scheme* to the numerical flux ψ^n

$$\psi^n = \psi^- \text{ on } \partial K \backslash \partial \mathcal{D}^- \tag{B.21}$$

$$\psi^n = \psi^{inc} \text{ on } \partial \mathcal{D}^- \tag{B.22}$$

to the face terms in the Eq. (B.19), we obtain

$$\begin{aligned}(\psi, -\vec{\Omega}\cdot\vec{\nabla}\psi^*)_K + \left\langle \psi^-, \psi^{*-}\right\rangle_{\partial K^+} + (\sigma\psi, \psi^*)_K \\ = (S, \psi^*)_K + \left\langle \psi^-, \psi^{*+}\right\rangle_{\partial K^- \setminus \partial\mathcal{D}^-} + \left\langle \psi^{inc}, \psi^{*+}\right\rangle_{\partial K^- \cap \partial\mathcal{D}^-}\end{aligned} \tag{B.23}$$

We have *the local conservation* with the upwind scheme by substituting a constant test function

$$J_K^{out} + R_K = S_K + J_K^{in} \tag{B.24}$$

with

$$\begin{aligned}J_K^{out} &= \int_{\partial K^+} |\vec{\Omega}\cdot\vec{n}|\psi^-\ ds \\ J_K^{in} &= \int_{\partial K^-} |\vec{\Omega}\cdot\vec{n}|\psi^n\ ds \\ R_K &= \int_K \sigma\psi\ d\vec{r} \\ S_K &= \int_K S\ d\vec{r}\end{aligned} \tag{B.25}$$

We have the freedom to define the shape functions on the different types of master elements

$$\widehat{\mathbf{b}}_p(\vec{r}) = \begin{bmatrix}\widehat{b}_1 & \widehat{b}_2 & \cdots & \widehat{b}_{N(p)}\end{bmatrix}^T \tag{B.26}$$

$N(p)$ is a function of polynomial order p which gives the dimension of the local polynomial function space. The shape functions on an element K

$$\mathbf{b}_K = \widehat{\mathbf{b}}_{p_K} \circ F_K^{-1} \tag{B.27}$$

Expand the solution and the test function on the element K with the basis functions

with dimension being $N_K = N(p_K)$

$$\begin{aligned}
\psi(\vec{r})|_{\vec{r}\in K} &= \mathbf{b}_K^T(\vec{r}) \cdot \begin{bmatrix} \psi_{K,1} \\ \psi_{K,2} \\ \vdots \\ \psi_{K,N_K} \end{bmatrix} = \mathbf{b}_K^T(\vec{r}) \cdot \psi_K \\
\psi^*(\vec{r})|_{\vec{r}\in K} &= \psi_K^{*T} \cdot \mathbf{b}_K(\vec{r}) \\
S(\vec{r})|_{\vec{r}\in K} &= \mathbf{b}_K^T(\vec{r}) \cdot q_K
\end{aligned} \tag{B.28}$$

q_K is the external source vector. For the sake of simplicity, we shall suppress subscript K in the equations later. Substitute them into Eq. (B.23), we get

$$\begin{aligned}
&\psi^{*T}\big(-\mathbf{G} + \sum_{\partial K_i \in \partial K^+} \mathbf{H}_i + \sigma\mathbf{M}\big)\psi = \\
&\qquad \psi^{*T}\big(\mathbf{M}q + \sum_{\partial K_i \in \partial K^- \setminus \partial\mathcal{D}} \sum_{K' \in B_i} \overline{\mathbf{H}}_{i,K'}\psi_{K'} + \sum_{\partial K_i \in \partial\mathcal{D}} \mathbf{H}_i\psi_i^{inc}\big)
\end{aligned}$$

where

$$\mathbf{M} = \int_K \mathbf{b}\mathbf{b}^T \, d\vec{r} \tag{B.29}$$

$$\mathbf{G} = \int_K \left[\vec{\nabla}\mathbf{b}\vec{\Omega}\right] \mathbf{b}^T \, d\vec{r} \tag{B.30}$$

$$\mathbf{H}_i = \int_{\partial K_i} |\vec{\Omega}\cdot\vec{n}_i| \mathbf{b}\mathbf{b}^T \, ds, \qquad i = 1, \cdots, N_e \tag{B.31}$$

$$\overline{\mathbf{H}}_{i,K'} = \int_{\partial K_i \cap \partial K'} |\vec{\Omega}\cdot\vec{n}_i| \mathbf{b}\mathbf{b}_{K'}^T \, ds, \ i = 1, \cdots, N_e \tag{B.32}$$

B_i is the set of neighboring elements which is on the side i. N_e is the number of sides of element K. ψ_i^{inc} is the vector of the boundary incoming flux. Note that $\vec{\Omega} = \{\Omega_x \ \Omega_y \ \Omega_z\}^T$ and $\vec{\nabla} = \left\{\frac{\partial}{\partial x} \ \frac{\partial}{\partial y} \ \frac{\partial}{\partial z}\right\}$ in 3-D or $\vec{\Omega} = \{\Omega_x \ \Omega_y\}^T$ and $\vec{\nabla} = \left\{\frac{\partial}{\partial x} \ \frac{\partial}{\partial y}\right\}$ in 2-D. If there is no h-refinement on the neighboring element on the side i, the number of elements in B_i is 1. It could be more than two with multi-irregularity.

Because the test vector is arbitrary, the above equation is equivalent with solving

$$\mathbf{A}\psi = \mathbf{l} \tag{B.33}$$

where

$$\mathbf{A} = -\mathbf{G} + \sum_{\partial K_i \in \partial K^+} \mathbf{H}_i + \sigma \mathbf{M} \tag{B.34}$$

$$\mathbf{l} = \mathbf{M}q + \sum_{\partial K_i \in \partial K^- \backslash \partial \mathcal{D}} \sum_{K' \in B_i} \overline{\mathbf{H}}_{i,K'} \psi_{K'} + \sum_{\partial K_i \in \partial \mathcal{D}} \mathbf{H}_i \psi_i^{inc} \tag{B.35}$$

Once we have all solutions of all upwind elements, we can solve this local system.

If we apply the integration by parts once again from the Eq. (B.23), we get

$$\begin{aligned} &(\vec{\Omega} \cdot \vec{\nabla}\psi, \psi^*)_e + \left\langle \psi^+, \psi^{*+} \right\rangle_{\partial e^-} + (\sigma\psi, \psi^*)_e \\ &\quad = (S, \psi^*)_e + \left\langle \psi^-, \psi^{*+} \right\rangle_{\partial e^- \backslash \partial \mathcal{D}^-} + \left\langle \psi^{inc}, \psi^{*+} \right\rangle_{\partial e^- \cap \partial \mathcal{D}^-} \end{aligned} \tag{B.36}$$

The local system is

$$(\mathbf{G}^T - \sum_{\partial K_i \in \partial K^-} \mathbf{H}_i + \sigma \mathbf{M})\psi = \mathbf{l} \tag{B.37}$$

We see that

$$-\mathbf{G} + \sum_{\partial K_i \in \partial K^+} \mathbf{H}_i = \mathbf{G}^T - \sum_{\partial K_i \in \partial K^-} \mathbf{H}_i \tag{B.38}$$

i.e.,

$$\mathbf{G} + \mathbf{G}^T = \mathbf{H} \equiv \sum_{i=1}^{N_e} \mathbf{H}_i \tag{B.39}$$

It can be proved the local system is always invertible. Sum the left hand side of Eq. (B.23) and Eq. (B.36)

$$b_K(\psi, \psi) = (\sigma\psi, \psi)_K + \left\langle \psi^+, \psi^+ \right\rangle_{\partial K^-} + \left\langle \psi^-, \psi^- \right\rangle_{\partial K^+} \geq 0$$

Sum up Eq. (B.36) over all elements we get the variational form with DGFEM:

Find $\psi_h \in W_{\mathcal{D}}^h$, such that

$$b_{\vec{\Omega},h}(\psi_h, \psi_h^*) = R(\psi_h^*) \qquad \forall \psi_h^* \in W_{\mathcal{D}}^h \tag{B.40}$$

where

$$b_{\vec{\Omega},h}(\psi_h, \psi_h^*) \equiv (\vec{\Omega} \cdot \vec{\nabla}\psi_h + \sigma\psi_h, \psi_h^*)_{\mathcal{D}} + \left\langle [\![\psi_h]\!], \psi_h^{*+} \right\rangle_{E_h^i} + \left\langle \psi_h^+, \psi_h^{*+} \right\rangle_{\partial\mathcal{D}^-} \tag{B.41}$$

$$R(\psi_h^*) \equiv (S, \psi_h^*)_{\mathcal{D}} + \left\langle \psi^{inc}, \psi_h^{*+} \right\rangle_{\partial\mathcal{D}^-} \tag{B.42}$$

$$W_{\mathcal{D}}^h = \left\{ \psi \in L^2(\mathcal{D}) | \psi_{|K} \in V(K),\ \forall K \in T_h \right\} \tag{B.43}$$

$$(f, g)_D = \sum_{K \in \mathbb{T}_h} \int_K f \cdot g \ d\vec{r} \tag{B.44}$$

$$\langle f, g \rangle_{E_h^i} = \sum_{e \in E_h^i} \int_e |\vec{\Omega} \cdot \vec{n}_e| f \cdot g \ ds \tag{B.45}$$

$$E_h^i = \cup_{\partial K} \backslash \partial\mathcal{D} \tag{B.46}$$

We use the subscript h to make clear the difference between Eq. (B.40) and Eq. (B.3). The operator $[\![\cdot]\!]$ is defined

$$[\![f]\!] = f^+ - f^- \tag{B.47}$$

This system can be solved with one sweep of all elements with proper ordering. This ordered solving sweep is usually called the *transport sweep*. Because the exact solution is continuous along $\vec{\Omega}$, it satisfies the Eq. (B.40). So we have the Galerkin orthogonality

$$b_{\vec{\Omega},h}(\psi_h - \psi, \psi_h^*) = 0 \qquad \forall \psi_h^* \in W_{\mathcal{D}}^h \tag{B.48}$$

DGFEM scheme is consistent because $\psi_h \to \psi$ when the mesh size $h \to 0$.

We can also write down the adjoint form of equations

$$\begin{aligned} &(\vec{\Omega} \cdot \vec{\nabla}\psi, \psi^*)_K + \left\langle \psi^+, \psi^{*+} \right\rangle_{\partial K^-} + (\sigma\psi, \psi^*)_K \\ &\quad = (\psi, S^*)_K + \left\langle \psi^-, \psi^{*+} \right\rangle_{\partial K^+ \backslash \partial\mathcal{D}^+} + \left\langle \psi^-, \psi^{*out} \right\rangle_{\partial K^+ \cap \partial\mathcal{D}^+} \end{aligned} \tag{B.49}$$

Apply integration by parts again,

$$\begin{aligned}(\psi, -\vec{\Omega}\cdot\vec{\nabla}\psi^*)_K + \left\langle \psi^-, \psi^{*-}\right\rangle_{\partial K^+} + (\sigma\psi, \psi^*)_K \\ = (\psi, S^*)_K + \left\langle \psi^-, \psi^{*+}\right\rangle_{\partial K^+\setminus\partial\mathcal{D}^+} + \left\langle \psi^-, \psi^{*out}\right\rangle_{\partial K^+\cap\partial\mathcal{D}^+}\end{aligned} \tag{B.50}$$

Sum over all element,

$$b^*_{\vec{\Omega},h}(\psi_h, \psi^*_h) \equiv (\psi_h, -\vec{\Omega}\cdot\vec{\nabla}\psi^*_h + \sigma\psi^*_h)_{\mathcal{D}} - \left\langle \psi^-_h, [\![\psi^*_h]\!]\right\rangle_{E^i_h} + \left\langle \psi^-_h, \psi^{*-}_h\right\rangle_{\partial\mathcal{D}^+} \tag{B.51}$$

$$R(\psi_h) \equiv (\psi_h, S^*)_{\mathcal{D}} + \left\langle \psi^-_h, \psi^{*out}\right\rangle_{\partial\mathcal{D}^+} \tag{B.52}$$

Sum Eq. (B.41) and Eq. (B.51) and divide it by 2

$$b_{\vec{\Omega},h}(\psi_h, \psi_h) = \|\psi_h\|^2_{L^2(D)} + \frac{1}{2}\sum_{e\in E^i_h}\int_e |\vec{\Omega}\cdot\vec{n}_e|[\![\psi_h]\!]^2\, ds + \frac{1}{2}\int_{\partial\mathcal{D}} |\vec{\Omega}\cdot\vec{n}|\psi^2_h\, ds \geq 0 \tag{B.53}$$

With this property, we define the DG-norm

$$|||\phi_h|||^2_{DG} = b_{\vec{\Omega},h}(\psi_h, \psi_h) \tag{B.54}$$

APPENDIX C

MULTIGROUP S_N TRANSPORT EQUATIONS

A. Multigroup Transport Equations

Integrate Eq. (A.1) over number G of energy intervals or groups $[E_g, E_{g-1}]$, $g = 1, \cdots, G$, we obtain

$$\left(\vec{\Omega}\cdot\vec{\nabla}+\sigma_{t,g}\right)\Psi_g = S_{ext,g} + \frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\nu\sigma_{f,g'}\Phi_{g'} + \sum_{g'=1}^{G}\int_{4\pi}\sigma_s^{g'\to g}(\vec{\Omega}',\vec{\Omega})\Psi_{g'}(\vec{\Omega}')d\Omega' \quad \text{(C.1)}$$

where

$$\Psi_g(\vec{r},\vec{\Omega}) \equiv \int_{E_g}^{E_{g-1}} \Psi(\vec{r},\vec{\Omega},E)\ dE \quad \text{(C.2)}$$

$$\Phi_g(\vec{r}) \equiv \int_{E_g}^{E_{g-1}}\int_{4\pi} \Psi(\vec{r},\vec{\Omega},E)\ d\Omega dE \quad \text{(C.3)}$$

$$S_{ext,g}(\vec{r},\vec{\Omega}) \equiv \int_{E_g}^{E_{g-1}} S_{ext}(\vec{r},\vec{\Omega},E)\ dE \quad \text{(C.4)}$$

$$\sigma_{t,g}(\vec{r},\vec{\Omega}) \equiv \frac{\int_{E_g}^{E_{g-1}}\sigma_t(\vec{r},E)\Psi(\vec{r},\vec{\Omega},E)\ dE}{\int_{E_g}^{E_{g-1}}\Psi(\vec{r},\vec{\Omega},E)\ dE} \quad \text{(C.5)}$$

$$\nu\sigma_{f,g}(\vec{r}) \equiv \frac{\int_{E_g}^{E_{g-1}}\nu\sigma_f(\vec{r},E)\Phi(\vec{r},E)\ dE}{\int_{E_g}^{E_{g-1}}\Phi(\vec{r},E)\ dE} \quad \text{(C.6)}$$

$$\chi_g(\vec{r}) \equiv \int_{E_g}^{E_{g-1}} \chi(\vec{r},E)\ dE \quad \text{(C.7)}$$

$$\sigma_s^{g'\to g}(\vec{r},\vec{\Omega}',\vec{\Omega}) \equiv \frac{\int_{E_{g'}}^{E_{g'-1}}\left[\int_{E_g}^{E_{g-1}}\sigma_s(\vec{r},E'\to E,\vec{\Omega}'\cdot\vec{\Omega})\ dE\right]\Psi(\vec{r},\vec{\Omega}',E')\ dE'}{\int_{E_{g'}}^{E_{g'-1}}\Psi(\vec{r},\vec{\Omega}',E')\ dE'} \quad \text{(C.8)}$$

The above equations mean that we can obtain the multigroup cross sections exactly only after we know the continuous transport solution. Note that the total and scattering cross sections are angular dependent. However, the above equations allow us introduce the multigroup approximation:

$$\Psi(\vec{r},\vec{\Omega},E) = \hat{\Psi}(\vec{r},\vec{\Omega})f_{g,r}(E) \quad \text{(C.9)}$$

so

$$\Phi(\vec{r}, E) = \hat{\Phi}(\vec{r}) f_{g,r}(E) \tag{C.10}$$

r is the region ID under consideration. We may have many regions in the solution domain to reduce the approximation error induced by the multigroup approximation. $f_{g,r}(E)$ is called as the neutron spectrum of a region r and an energy group g. With this approximation,

$$\sigma_{t,g}(\vec{r}) \equiv \frac{\int_{E_g}^{E_{g-1}} \sigma_t(\vec{r}, E) f_{g,r}(E)\ dE}{\int_{E_g}^{E_{g-1}} f_{g,r}(E)\ dE} \tag{C.11}$$

$$\nu\sigma_{f,g}(\vec{r}) \equiv \frac{\int_{E_g}^{E_{g-1}} \nu\sigma_f(\vec{r}, E) f_{g,r}(E)\ dE}{\int_{E_g}^{E_{g-1}} f_{g,r}(E)\ dE} \tag{C.12}$$

$$\sigma_s^{g' \to g}(\vec{r}, \vec{\Omega}' \cdot \vec{\Omega}) \equiv \frac{\int_{E_{g'}}^{E_{g'-1}} \left[\int_{E_g}^{E_{g-1}} \sigma_s(\vec{r}, E' \to E, \vec{\Omega}' \cdot \vec{\Omega})\ dE \right] f_{g',r}(E')\ dE'}{\int_{E_{g'}}^{E_{g'-1}} f_{g',r}(E')\ dE'} \tag{C.13}$$

We expect when the regions covering the domain are getting smaller and energy groups are becoming thinner, the error caused by this approximation tends to be zero. The accuracy of the multigroup approximation depends on how the full range of energy is cut and number of regions are considered and also depends on how well the spectrum can be obtained.

Because the integral in energy will smooth the scattering kernel, and higher moments of angular flux are significantly smaller in most situations, we usually treat

the scattering term with a Legendre polynomial expansion:

$$
\begin{aligned}
&\int_{4\pi} \sigma_s^{g'\to g}(\vec{r},\vec{\Omega}'\cdot\vec{\Omega})\Psi_{g'}(\vec{r},\vec{\Omega}')\ d\Omega' \\
=\ &\int_{4\pi}\left[\sum_{n=0}^{N_s}\frac{2n+1}{2}\sigma_{s,n}^{g'\to g}(\vec{r})P_n(\vec{\Omega}'\cdot\vec{\Omega})\right]\left[\sum_{m=0}^{N_f}\sum_{l=-m}^{m}\Phi_{m,l}^{g'}(\vec{r})Y_{m,l}(\vec{\Omega}')\right]\ d\Omega' \\
=\ &\int_{4\pi}\left[\sum_{n=0}^{N_s}\frac{2n+1}{2}\sigma_{s,n}^{g'\to g}(\vec{r})\frac{2}{2n+1}\sum_{k=-n}^{n}Y_{n,k}(\vec{\Omega})Y_{n,k}^*(\vec{\Omega}')\right]\left[\sum_{m=0}^{N_f}\sum_{l=-m}^{m}\Phi_{m,l}^{g'}(\vec{r})Y_{m,l}(\vec{\Omega}')\right]\ d\Omega' \\
=\ &\int_{4\pi}\left[\sum_{n=0}^{N_s}\sigma_{s,n}^{g'\to g}(\vec{r})\sum_{k=-n}^{n}Y_{n,k}(\vec{\Omega})Y_{n,k}^*(\vec{\Omega}')\right]\left[\sum_{m=0}^{N_f}\sum_{l=-m}^{m}\Phi_{m,l}^{g'}(\vec{r})Y_{m,l}(\vec{\Omega}')\right]\ d\Omega' \\
=\ &\int_{4\pi}\left[\sum_{n=0}^{N_s}\sum_{k=-n}^{n}\sum_{m=0}^{N_f}\sum_{l=-m}^{m}\sigma_{s,n}^{g'\to g}(\vec{r})\Phi_{m,l}^{g'}(\vec{r})Y_{n,k}(\vec{\Omega})Y_{n,k}^*(\vec{\Omega}')Y_{m,l}(\vec{\Omega}')\right]\ d\Omega' \\
=\ &\sum_{n=0}^{N_s}\sum_{k=-n}^{n}\sum_{m=0}^{N_f}\sum_{l=-m}^{m}\sigma_{s,n}^{g'\to g}(\vec{r})\Phi_{m,l}^{g'}(\vec{r})Y_{n,k}(\vec{\Omega})\delta_{n,m}\delta_{k,l} \\
=\ &\sum_{n=0}^{\min(N_s,N_f)}\sum_{k=-n}^{n}\sigma_{s,n}^{g'\to g}(\vec{r})\Phi_{n,k}^{g'}(\vec{r})Y_{n,k}(\vec{\Omega})=\sum_{n=0}^{N}\sigma_{s,n}^{g'\to g}(\vec{r})\sum_{k=-n}^{n}\Phi_{n,k}^{g'}(\vec{r})Y_{n,k}(\vec{\Omega})
\end{aligned}
$$

In the derivation, we defined

$$\mu_0 \equiv \vec{\Omega}\cdot\vec{\Omega}' \tag{C.14}$$

$$\sigma_s^{g'\to g}(\vec{r},\vec{\Omega}\cdot\vec{\Omega}') \equiv \frac{1}{2\pi}\sigma_s^{g'\to g}(\vec{r},\mu_0) \tag{C.15}$$

$$P_n(\vec{\Omega}\cdot\vec{\Omega}') \equiv \frac{1}{2\pi}P_n(\mu_0) \tag{C.16}$$

$$\sigma_{s,n}^{g'\to g}(\vec{r}) \equiv \int_{-1}^{1}\sigma_s^{g'\to g}(\vec{r},\mu_0)P_n(\mu_0)\ d\mu_0 \tag{C.17}$$

$$\Phi_{n,k}^{g}(\vec{r}) \equiv \int_{4\pi}\Psi_g(\vec{r},\vec{\Omega})Y_{n,k}^*(\vec{\Omega})\ d\Omega \tag{C.18}$$

The spherical harmonics are,

$$Y_{n,k}(\vec{\Omega}) \equiv \sqrt{C_{n,k}} P_n^k(\mu) e^{ik\theta} \tag{C.19}$$

$$C_{n,k} \equiv \frac{(2n+1)(n-k)!}{4\pi(n+k)!} \tag{C.20}$$

Note these spherical harmonics are normalized. μ is the cosine of the polar (colatitudinal) angle. θ is the azimuthal (longitudinal) angle with $\theta \in [0, 2\pi)$. $P_n^k(\mu)$ is the associated Legendre polynomials. They have following properties,

$$P_n^0(\mu) = P_n(\mu) \tag{C.21}$$

$$P_n^{-k} = (-1)^k \frac{(n-k)!}{(n+k)!} P_n^k(\mu) \tag{C.22}$$

$P_n(\mu)$ are the Legendre polynomials. Note that $Y_{n,0} = \sqrt{\frac{2n+1}{4\pi}} P_n(\mu)$. Also note that with this definition of spherical harmonics

$$\Phi_g(\vec{r}) = \sqrt{4\pi} \Phi_{0,0}^g(\vec{r}) \tag{C.23}$$

In the above derivation, we used the spherical harmonic addition theorem

$$P_n(\vec{\Omega} \cdot \vec{\Omega}') \equiv \frac{1}{2\pi} P_n(\mu_0) = \frac{2}{2n+1} \sum_{k=-n}^{n} Y_{n,k}(\vec{\Omega}) Y_{n,k}^*(\vec{\Omega}') \tag{C.24}$$

In the above derivation, we truncate the Legendre expansion of the scattering cross section to N_s and the spherical harmonic expansion of the angular flux to N_f. This ends up with the truncation up to $N_a \equiv min(N_s, N_f)$ known as *the P_N approximation.* Note that if the scattering is isotropic

$$\int_{4\pi} \sigma_s^{g' \to g}(\vec{r}, \vec{\Omega}' \cdot \vec{\Omega}) \Psi_{g'}(\vec{r}, \vec{\Omega}') \, d\Omega' = \sigma_{s,0}^{g' \to g}(\vec{r}) \Phi_{0,0}^{g'}(\vec{r}) Y_{0,0}(\vec{\Omega}) = \frac{1}{4\pi} \sigma_{s,0}^{g' \to g}(\vec{r}) \Phi_{g'}(\vec{r}) \tag{C.25}$$

So the multigroup transport equation with the P_N approximation,

$$\left(\vec{\Omega}\cdot\vec{\nabla}+\sigma_{t,g}\right)\Psi_g = S_{ext,g}+\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\nu\sigma_{f,g'}\Phi_{g'}+\sum_{g'=1}^{G}\sum_{n=0}^{N_a}\sigma_{s,n}^{g'\to g}\sum_{k=-n}^{n}\Phi_{n,k}^{g'}(\vec{r})Y_{n,k} \quad \text{(C.26)}$$

with the boundary condition

$$\Psi_g(\vec{r}_b,\vec{\Omega}) = \Psi_g^{inc}(\vec{r}_b,\vec{\Omega})+\sum_{g'=1}^{G}\int_{\vec{\Omega}'\cdot\vec{n}_b>0}\beta^{g'\to g}(\vec{r}_b,\vec{\Omega}'\to\vec{\Omega})\Psi_{g'}(\vec{r}_b,\vec{\Omega}')\,d\Omega' \quad \text{(C.27)}$$

$$\text{on } \vec{r}_b\in\partial\mathcal{D} \text{ and } \vec{\Omega}\cdot\vec{n}_b<0$$

where

$$\beta^{g'\to g}(\vec{r}_b,\vec{\Omega}'\to\vec{\Omega}) = \frac{\int_{E_{g'}}^{E_{g'-1}}\left[\int_{E_g}^{E_{g-1}}\beta(\vec{r}_b,E'\to E,\vec{\Omega}'\to\vec{\Omega})\,dE\right]f_{g',r}(E')\,dE'}{\int_{E_{g'}}^{E_{g'-1}}f_{g',r}(E')\,dE'} \quad \text{(C.28)}$$

We may use another definition of spherical harmonics which is favorable for coding because there is not imagining number in it.

$$\begin{aligned} Y_{n,k}^{e}(\vec{\Omega}) &\equiv \sqrt{C_{n,k}}P_n^k(\mu)\cos(k\theta),\ k=0,\cdots,n \\ Y_{n,k}^{o}(\vec{\Omega}) &\equiv \sqrt{C_{n,k}}P_n^k(\mu)\sin(k\theta),\ k=1,\cdots,n \\ C_{n,k} &\equiv \frac{(n-k)!}{(n+k)!}(2-\delta_{k,0}) \end{aligned} \quad \text{(C.29)}$$

With this definition, we have the orthogonality properties,

$$\int_{4\pi}Y_{n,k}^{e}(\vec{\Omega})Y_{m,l}^{e}(\vec{\Omega})\,d\Omega = \frac{4\pi}{2n+1}\delta_{n,m}\delta_{k,l} \quad \text{(C.30)}$$

$$\int_{4\pi}Y_{n,k}^{o}(\vec{\Omega})Y_{m,l}^{o}(\vec{\Omega})\,d\Omega = \frac{4\pi}{2n+1}\delta_{n,m}\delta_{k,l} \quad \text{(C.31)}$$

$$\int_{4\pi}Y_{n,k}^{e}(\vec{\Omega})Y_{m,l}^{o}(\vec{\Omega})\,d\Omega = 0 \quad \text{(C.32)}$$

With this definition, the addition theorem is

$$2\pi P_n(\vec{\Omega}\cdot\vec{\Omega}') = P_n(\mu_0) = \sum_{k=0}^{n}Y_{n,k}^{e}(\vec{\Omega})Y_{n,k}^{e}(\vec{\Omega}')+\sum_{k=1}^{n}Y_{n,k}^{o}(\vec{\Omega})Y_{n,k}^{o}(\vec{\Omega}') \quad \text{(C.33)}$$

We then define the flux moments,

$$\Phi^g_{n,k,e}(\vec{r}) = \int_{4\pi} \Psi_g(\vec{r},\vec{\Omega}) Y^e_{n,k}(\vec{\Omega})\ d\Omega \tag{C.34}$$

$$\Phi^g_{n,k,o}(\vec{r}) = \int_{4\pi} \Psi_g(\vec{r},\vec{\Omega}) Y^o_{n,k}(\vec{\Omega})\ d\Omega \tag{C.35}$$

The angular flux can be expanded with the spherical harmonics,

$$\Psi_g(\vec{r},\vec{\Omega}) = \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} \left[\sum_{k=0}^{n} \Phi^g_{n,k,e}(\vec{r}) Y^e_{n,k}(\vec{\Omega}) + \sum_{k=1}^{n} \Phi^g_{n,k,o}(\vec{r}) Y^o_{n,k}(\vec{\Omega}) \right] \tag{C.36}$$

One nice thing about this definition is

$$\begin{aligned} Y^e_{0,0}(\vec{\Omega}) &= 1 \\ Y^e_{1,1}(\vec{\Omega}) &= \Omega_x \\ Y^o_{1,1}(\vec{\Omega}) &= \Omega_y \\ Y^e_{1,0}(\vec{\Omega}) &= \Omega_z \end{aligned} \tag{C.37}$$

Correspondingly

$$\begin{aligned} \Phi^g_{0,0,e} &= \Phi_g \\ \Phi^g_{1,1,e} &= J^x_g \\ \Phi^g_{1,1,o} &= J^y_g \\ \Phi^g_{1,0,e} &= J^z_g \end{aligned} \tag{C.38}$$

Now the scattering term

$$
\begin{aligned}
&\int_{4\pi} \sigma_s^{g'\to g}(\vec{r},\vec{\Omega}'\cdot\vec{\Omega})\Psi_{g'}(\vec{r},\vec{\Omega}')\ d\Omega' \\
= &\int_{4\pi} \frac{\left[\sum_{n=0}^{N_s}\frac{2n+1}{2}\sigma_{s,n}^{g'\to g}(\vec{r})P_n(\vec{\Omega}'\cdot\vec{\Omega})\right]\cdot}{\left[\sum_{m=0}^{N_f}\frac{2m+1}{4\pi}\left[\sum_{l=0}^{m}\Phi_{m,l,e}^{g'}(\vec{r})Y_{m,l}^{e}(\vec{\Omega}')+\sum_{l=1}^{m}\Phi_{m,l,o}^{g'}(\vec{r})Y_{m,l}^{o}(\vec{\Omega}')\right]\right]}\ d\Omega' \\
= &\int_{4\pi} \frac{\left[\sum_{n=0}^{N_s}\frac{2n+1}{4\pi}\sigma_{s,n}^{g'\to g}(\vec{r})\left[\sum_{l=0}^{m}Y_{m,l}^{e}(\vec{\Omega})Y_{m,l}^{e}(\vec{\Omega}')+\sum_{l=1}^{m}Y_{m,l}^{o}(\vec{\Omega})Y_{m,l}^{o}(\vec{\Omega}')\right]\right]\cdot}{\left[\sum_{m=0}^{N_f}\frac{2m+1}{4\pi}\left[\sum_{l=0}^{m}\Phi_{m,l,e}^{g'}(\vec{r})Y_{m,l}^{e}(\vec{\Omega}')+\sum_{l=1}^{m}\Phi_{m,l,o}^{g'}(\vec{r})Y_{m,l}^{o}(\vec{\Omega}')\right]\right]}\ d\Omega' \\
= &\sum_{n=0}^{\min(N_s,N_f)}\frac{2n+1}{4\pi}\sigma_{s,n}^{g'\to g}(\vec{r})\left[\sum_{k=0}^{n}\Phi_{n,k,e}^{g'}(\vec{r})Y_{n,k}^{e}(\vec{\Omega})+\sum_{k=1}^{n}\Phi_{n,k,o}^{g'}(\vec{r})Y_{n,k}^{o}(\vec{\Omega})\right] \\
= &\sum_{n=0}^{N_a}\frac{2n+1}{4\pi}\sigma_{s,n}^{g'\to g}(\vec{r})\left[\sum_{k=0}^{n}\Phi_{n,k,e}^{g'}(\vec{r})Y_{n,k}^{e}(\vec{\Omega})+\sum_{k=1}^{n}\Phi_{n,k,o}^{g'}(\vec{r})Y_{n,k}^{o}(\vec{\Omega})\right]
\end{aligned}
$$

For notational simplicity, let us re-define

$$
\begin{aligned}
Y_{n,k}(\vec{\Omega}) &\equiv Y_{n,k}^{e}(\vec{\Omega}),\ k=0,\cdots,n \\
Y_{n,-k}(\vec{\Omega}) &\equiv Y_{n,k}^{o}(\vec{\Omega}),\ k=1,\cdots,n \\
\Phi_{n,k}^{g}(\vec{r}) &\equiv \Phi_{n,k,e}^{g}(\vec{r}),\ k=0,\cdots,n \\
\Phi_{n,-k}^{g}(\vec{r}) &\equiv \Phi_{n,k,o}^{g}(\vec{r}),\ k=1,\cdots,n
\end{aligned}
\tag{C.39}
$$

Then the multigroup transport equation is,

$$
\left(\vec{\Omega}\cdot\vec{\nabla}+\sigma_{t,g}\right)\Psi_g = S_{ext,g}+\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\nu\sigma_{f,g'}\Phi_{g'}+\sum_{g'=1}^{G}\sum_{n=0}^{N_a}\frac{2n+1}{4\pi}\sigma_{s,n}^{g'\to g}\sum_{k=-n}^{n}\Phi_{n,k}^{g'}(\vec{r})Y_{n,k} \tag{C.40}
$$

In reactor analysis, we often solve the general eigenvalue (k-eigenvalue) problem

$$
\left(\vec{\Omega}\cdot\vec{\nabla}+\sigma_{t,g}\right)\Psi_g = \frac{1}{k_{eff}}\frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\nu\sigma_{f,g'}\Phi_{g'}+\sum_{g'=1}^{G}\sum_{n=0}^{N_a}\frac{2n+1}{4\pi}\sigma_{s,n}^{g'\to g}\sum_{k=-n}^{n}\Phi_{n,k}^{g'}(\vec{r})Y_{n,k} \tag{C.41}
$$

with the homogeneous boundary conditions

$$\Psi_g(\vec{r}_b,\vec{\Omega}) = \sum_{g'=1}^{G} \int_{\vec{\Omega}'\cdot\vec{n}_b>0} \beta^{g'\to g}(\vec{r}_b,\vec{\Omega}'\to\vec{\Omega})\Psi_{g'}(\vec{r}_b,\vec{\Omega}')d\Omega' \qquad \text{on } \vec{r}_b\in\partial\mathcal{D} \text{ and } \vec{\Omega}\cdot\vec{n}_b<0 \tag{C.42}$$

B. The Iterative Solver for the Multigroup Problem

The traditional iterative solver of the multigroup eigenvalue problem Eq. (C.41) and Eq. (C.42) is described in Algorithm 3.

Algorithm 3 Iterative solver for the multigroup problem.

1: **Initialization**:

$$\Phi_g^{(0)} = \Phi_{0,0}^{g,(0)} = 1,\ g = 1,\cdots,G$$
$$\Phi_{n,k}^{g,(0)} = 0,\ g = 1,\cdots,G;\ n = 1,\cdots,N;\ k = -n,\cdots,n$$
$$k_{eff}^{(0)} = \boxed{k_{eff,0}} \qquad \text{(usually 1.0)}$$

2: Calculate the total fission source:

$$F^{(0)} = \sum_{g=1}^{G} \int_{\mathcal{D}} \nu\sigma_{f,g}(\vec{r})\Phi_g^{(0)}(\vec{r})\,d\vec{r}$$

3: Set the convergence flag=.FALSE.

4: **for power iteration** (outer iteration) $l = 1 : \boxed{max\_outer}$ **do**

5: **for fast group sweep** $g = 1 : nfg$ **do**

6: Construct fast group source moment:

$$Q_{n,k}^{g,(l)}(\vec{r}) = \begin{cases} \frac{\chi_g(\vec{r})}{k_{eff}^{(\mathbf{l-1})}} \sum_{g'=1}^{G} \nu\sigma_{f,g'}(\vec{r})\Phi_{g'}^{(\mathbf{l-1})}(\vec{r}) + \sum_{g'=1}^{\mathbf{g-1}} \sigma_{s,0}^{g'\to g}\Phi_{0,0}^{g',(\mathbf{l})}(\vec{r}) & , n=k=0 \\ \sum_{g'=1}^{\mathbf{g-1}} \sigma_{s,n}^{g'\to g}(\vec{r})\Phi_{n,k}^{g',(\mathbf{l})}(\vec{r}) & , \text{otherwise} \end{cases}$$

7: Solve the one-group source problem:

$$\left(\vec{\Omega}\cdot\vec{\nabla}+\sigma_{t,g}\right)\Psi_g^{(l)}=\sum_{n=0}^{N_a}\frac{2n+1}{4\pi}\sum_{k=-n}^{n}\left[Q_{n,k}^{g,(l)}(\vec{r})+\sigma_{s,n}^{g\to g}(\vec{r})\Phi_{n,k}^{g,(l)}(\vec{r})\right]Y_{n,k}(\vec{\Omega})$$

8: **end for** fast group sweep

9: Construct fast-to-thermal scattering sources:

$$Q_{n,k,fast}^{g,(l)}(\vec{r})=\begin{cases}\frac{\chi_g(\vec{r})}{k_{eff}^{(\mathbf{l-1})}}\sum_{g'=1}^{G}\nu\sigma_{f,g'}(\vec{r})\Phi_{g'}^{(\mathbf{l-1})}(\vec{r})+\sum_{g'=1}^{\mathbf{nfg}}\sigma_{s,0}^{g'\to g}\Phi_{0,0}^{g',(\mathbf{l})}(\vec{r}) & ,n=k=0\\ \sum_{g'=1}^{\mathbf{nfg}}\sigma_{s,n}^{g'\to g}(\vec{r})\Phi_{n,k}^{g',(\mathbf{l})}(\vec{r}) & ,\text{otherwise}\end{cases}$$

$$g=nfg+1,\cdots,G$$

10: Initialize thermal flux moments:

$$\Phi_{n,k}^{g,(\mathbf{l},\mathbf{m=0})}(\vec{r})=\Phi_{n,k}^{g,(\mathbf{l-1})}(\vec{r}),\qquad g=nfg+1,\cdots,G$$

11: **for thermal iteration** $m=1:\boxed{max\_thermal}$ **do**

12: **for thermal group sweep** $g=nfg+1:G$ **do**

13: Construct thermal group source moment:

$$Q_{n,k}^{g,(l,m)}(\vec{r})=Q_{n,k,fast}^{g,(l)}(\vec{r})+\sum_{g'=nfg+1}^{g-1}\sigma_{s,n}^{g'\to g}(\vec{r})\Phi_{n,k}^{g',(\mathbf{l},\mathbf{m})}(\vec{r})+\sum_{g'=g+1}^{G}\sigma_{s,n}^{g'\to g}(\vec{r})\Phi_{n,k}^{g',(\mathbf{l},\mathbf{m-1})}(\vec{r})$$

14: Solve the one-group source problem:

$$\left(\vec{\Omega}\cdot\vec{\nabla}+\sigma_{t,g}\right)\Psi_g^{(l,m)}=\sum_{n=0}^{N_a}\frac{2n+1}{4\pi}\sum_{k=-n}^{n}\left[Q_{n,k}^{g,(l,m)}(\vec{r})+\sigma_{s,n}^{g\to g}(\vec{r})\Phi_{n,k}^{g,(l,m)}(\vec{r})\right]Y_{n,k}(\vec{\Omega})$$

15: **end for** thermal group sweep

16: Test thermal convergence:

$$\max_{g=nfg+1,\cdots,G}\frac{\left\|\Phi_g^{(l,m)}-\Phi_g^{(l,m-1)}\right\|}{\left\|\Phi_g^{(l,m)}\right\|}\leq\boxed{\text{tol}_{thermal}}\qquad\text{(C.43)}$$

17: **if** the criteria is satisfied **then**

18: Terminate the thermal iteration.

19: **end if**

20: **end for** thermal iteration

21: Set the latest thermal solutions for the current power iteration:

$$\Psi_g^{(l)} = \Psi_g^{(l,m)}, \quad g = nfg + 1, \cdots, G$$

22: Update total fission source and k_{eff}:

$$F^{(l)} = \sum_{g=1}^{G} \int_D \nu\sigma_{f,g}(\vec{r})\Phi_g^{(l)}(\vec{r})\,\mathrm{d}\vec{r}$$

$$k_{eff}^{(l)} = \frac{F^{(l)}}{F^{(l-1)}} k_{eff}^{(l-1)}$$

23: Test outer convergence:

$$\frac{\left|k_{eff}^{(l)} - k_{eff}^{(l-1)}\right|}{k_{eff}^{(l)}} \leq \boxed{\mathrm{tol}_{keff}} \tag{C.44}$$

$$\max_{1\leq g\leq G}\left[\max_{K\in T_h} \frac{\left\|\Phi_g^{(l)} - \Phi_g^{(l-1)}\right\|_{2,K}}{\left\|\Phi_g^{(l)}\right\|_{2,K}}\right] \leq \boxed{\mathrm{tol}_{flux}} \tag{C.45}$$

24: **if** the outer criteria is satisfied **then**

25: Set flag=.TRUE. and terminate the power iteration.

26: **end if**

27: **end for** power iteration

Remarks:

- Control parameters are boxed in the algorithm, including
  - *max_outer* Maximum number of outer iterations

  - $max\_thermal$ Maximum number of thermal iterations
  - $\text{tol}_{thermal}$ Tolerance of the thermal iteration
  - tol_{keff} Tolerance on k_{eff} for the outer iteration
  - tol_{flux} Tolerance on flux for the outer iteration
  - $k_{eff,0}$ Initial guess for the k-effective

- Number of fast groups nfg plus one is equal to the minimum number of energy group with up-scattering sources, i.e., sources from energy group(s) with number being larger than its number. Note that nfg does not necessarily mean the number of fast groups, for example, if we have a problem with only two groups, and the second group covers the entire thermal energy range, both energy groups do not have up-scattering and are called the "fast" group in the algorithm although the second energy group is thermal.

- $F^{(l)}k_{eff}^{(l)}$ is constant during the power iteration.

- If all energy groups are using the same mesh, the fission distribution

$$f(\vec{r}) = \sum_{g=1}^{G} \nu\sigma_{f,g}(\vec{r})\Phi_g^{(l)}(\vec{r}) \tag{C.46}$$

  can be calculated and stored at each power iteration to save time for computing the fission sources of all energy groups.

- Power iteration can be simply accelerated with techniques such as Chebyshev acceleration.

- Gauss-Seidel scheme on the thermal group sweep is applied in the algorithm. Thermal re-balance could be done after each thermal iteration.

- Solver for the one-group source problem with DGFEM is detailed in the body of this dissertation.
- For the multi-group source problems we simply remove the outer iteration.
- The whole procedure can be accelerated with CMFD (Coarse-Mesh Finite Difference).
- New solvers regarding large number of energy groups (over 1000) are under development.

C. Multigroup S_N Equations

Given an angular quadrature set $\left\{\vec{\Omega}_m, w_m\right\}_{m=1,\dots M}$ and number G of energy intervals or groups $[E_g, E_{g-1}]$, $g = 1, \cdots, G$, the steady-state multigroup S_N equation in one direction m and for one group g in the open convex space domain $\mathcal{D}$ with the boundary $\partial\mathcal{D}$ is

$$\left(\vec{\Omega}_m \cdot \vec{\nabla} + \sigma_{t,g}\right)\Psi_{m,g} = S^{ext}_{m,g} + \frac{\chi_g}{4\pi}\sum_{g'=1}^{G}\nu\sigma_{f,g'}\Phi_{g'} + \sum_{g'=1}^{G}\sum_{n=0}^{N_a}\frac{2n+1}{4\pi}\sigma^{g'\to g}_{s,n}\sum_{k=-n}^{n}\Phi^{g'}_{n,k}Y_{n,k}(\vec{\Omega}_m) \tag{C.47}$$

with the boundary condition

$$\Psi_{m,g}(\vec{r}_b) = \Psi^{inc}_{m,g}(\vec{r}_b) + \sum_{g'=1}^{G}\sum_{\vec{\Omega}_{m'}\cdot\vec{n}_b>0}\beta^{g'\to g}_{m'\to m}(\vec{r}_b)\Psi_{m',g'}(\vec{r}_b) \tag{C.48}$$

$$\text{on } \vec{r}_b \in \partial\mathcal{D} \text{ and } \vec{\Omega}_m \cdot \vec{n}_b < 0$$

Symbol meanings are listed below:

$\vec{r}$	position variable [cm]
$\mathcal{D}$	$\in \mathbb{R}^d$ open d-dimensional convex space domain
$\partial\mathcal{D}$	boundary of space domain
$\vec{n}_b$	$= \vec{n}(\vec{r}_b)$ outward unit normal vector on boundary
$\vec{\Omega}_m$	unit steaming direction vector in the angular quadrature set
m	index of streaming directions from 1 to M
g	index of energy groups from 1 to G, usually E_0=20MeV and E_G=0
$\Psi_{m,g}(\vec{r})$	$= \Psi_g(\vec{r}, \vec{\Omega}_m)$ neutron angular flux $[\frac{n}{cm^2 \cdot ster \cdot s}]$
$\Phi_g(\vec{r})$	$= \sum_{m=1}^{M} w_m \Psi_{m,g}$ neutron scalar flux $[\frac{n}{cm^2 \cdot s}]$
$\Phi^g_{n,k}(\vec{r})$	$= \sum_{m=1}^{M} w_m Y_{n,k}(\vec{\Omega}_m)\Psi_{m,g}$ neutron flux moments $[\frac{n}{cm^2 \cdot s}]$
$Y_{n,k}(\vec{\Omega})$	spherical harmonics defined with Eq. (C.29) and Eq. (C.39)
$S^{ext}_{m,g}(\vec{r})$	$= S^{ext}_g(\vec{r}, \vec{\Omega}_m)$ external source $[\frac{n}{cm^2 \cdot ster \cdot s}]$
$\sigma_{t,g}(\vec{r})$	macroscopic total cross section $[cm^{-1}]$
$\sigma^{g' \to g}_{s,n}(\vec{r})$	$= \int_{-1}^{1} \sigma^{g' \to g}_s(\vec{r}, \mu) P_n(\mu)\, d\mu$ macroscopic scattering cross sections $[cm^{-1}]$
N_a	truncation order of the P_N approximation, see Appendix A for more details
$\nu\sigma_{f,g}(\vec{r})$	fission cross section times the average number of neutrons emitted per fission $[cm^{-1}]$
$\chi_g(\vec{r})$	neutron fission spectrum
$\beta^{g' \to g}_{m' \to m}(\vec{r}_b)$	boundary albedo, its definition depends on the quadrature set, also refer to the Eq. (C.28)

Apparently to re-produce the multigroup solution, we need the quadrature set to satisfy the following orthogonal conditions:

$$\int_{4\pi} Y_{n,k}(\vec{\Omega})Y_{m,l}(\vec{\Omega})\ d\Omega = \frac{4\pi}{2n+1}\delta_{n,m}\delta_{k,l} \\ , \quad n=0,\cdots,N_s; \quad k=-n,\cdots,n \\ , \quad m=0,\cdots,N_f; \quad l=-n,\cdots,n \tag{C.49}$$

Because expansion order N_f of angular flux is infinite generally, S_N equation will not exactly equivalent with the transport equation in general and as the result, we can see the S_N singularities along some characteristic lines also known as the ray effects. The angular quadrature is designed to satisfy as many orthogonality properties as possible, at least for all with $n = 0, 1$. Currently, ray effects are mitigated using first-collision approaches or by making the S_N produce the P_N results. Ray effects will not be the focus here. Multigroup P_N equation will also not be presented.

D. Variational Form for the Multigroup S_N Equations with DGFEM

We will not give the details on how the variational form with DGFEM for the general multigroup S_N transport equation is obtained but present it directly.

First we define the function space,

$$W_{\mathcal{D}}^h = \left\{\Psi_{m,g} \in L^2(\mathcal{D});\ \Psi_{m,g}|_K \in V(K), \forall K \in \mathbb{T}_{g,h}, \quad m=1,\cdots,M;\ g=1,\cdots,G\right\} \tag{C.50}$$

Note that meshes for all G energy groups do not have to be the same. The variational form:

Find $\Psi \in W_{\mathcal{D}}^h$ such that:

$$a(\Psi,\Psi^*) = l(\Psi^*) \qquad \forall \Psi^* \in W_{\mathcal{D}}^h \tag{C.51}$$

where

$$a(\Psi,\Psi^*) = b(\Psi,\Psi^*) - \sum_{g=1}^{G}\sum_{g'=1}^{G}\sum_{e\in\partial\mathcal{D}}\sum_{\vec{\Omega}_m\cdot\vec{n}_b<0}\sum_{\vec{\Omega}_{m'}\cdot\vec{n}_b>0}\left\langle w_m\beta^{g'\to g}_{m'\to m}\Psi_{m',g'},\Psi^*_{m,g}\right\rangle_e -$$
$$\sum_{g=1}^{G}\sum_{g'=1}^{G}\frac{1}{4\pi}\left(\nu\sigma_{f,g'}\Phi_{g'},\chi_g\Phi^*_g\right)_{\mathcal{D}} -$$
$$\sum_{g=1}^{G}\sum_{g'=1}^{G}\sum_{n=0}^{N_a}\sum_{k=-n}^{n}\frac{2n+1}{4\pi}\left(\sigma^{g'\to g}_{s,n}\Phi^{g'}_{n,k},\Phi^{g*}_{n,k}\right)_{\mathcal{D}} \tag{C.52}$$

$$b(\Psi,\Psi^*) = \sum_{g=1}^{G}\sum_{m=1}^{M}w_m\left((\vec{\Omega}_m\cdot\vec{\nabla}+\sigma_{t,g})\Psi_{m,g},\Psi^*_{m,g}\right)_{\mathcal{D}} + \sum_{g=1}^{G}\sum_{m=1}^{M}w_m\left\langle[\![\Psi_{m,g}]\!],\Psi^{*+}_{m,g}\right\rangle_{E^i_{g,h}} +$$
$$\sum_{g=1}^{G}\sum_{e\in\partial\mathcal{D}}\sum_{\vec{\Omega}_m\cdot\vec{n}_b<0}w_m\left\langle\Psi_{m,g},\Psi^*_{m,g}\right\rangle_e \tag{C.53}$$

$$l(\Psi^*) = \sum_{g=1}^{G}\sum_{m=1}^{M}w_m\left(S^{ext}_{m,g},\Psi^*_{m,g}\right)_{\mathcal{D}} + \sum_{g=1}^{G}\sum_{e\in\partial\mathcal{D}}\sum_{\vec{\Omega}_m\cdot\vec{n}_b<0}w_m\left\langle\Psi^{inc}_{m,g},\Psi^*_{m,g}\right\rangle_e \tag{C.54}$$

Because the meshes could vary with energy groups, the interior edge sets $E^i_{g,h}$ are also group-dependent. Bilinear form $a(\Psi,\Psi^*)$ is not symmetric. If there is no fission and the scattering cross section is physically possible, the positiveness of the bilinear form can be proved. Again we can find the adjoint bilinear form and prove the primal and the adjoint are the same.

APPENDIX D

1-D LINEAR DGFEM FOR THE S_N TRANSPORT WITH DSA

1-D transport equation with isotropic scatterings with $x \in [0, a]$ and $\mu \in [-1, 1]$

$$\mu \frac{\partial \psi}{\partial x} + \sigma_t(x)\psi(x, \mu) = \frac{\sigma_s(x)}{2} \int_{-1}^{1} \psi(x, \mu')\, d\mu' + \frac{Q(x)}{2} \tag{D.1}$$

and the boundary conditions

$$\psi(x = 0, \mu) = \psi^{left}(\mu), \quad \text{where } \mu > 0 \tag{D.2}$$

$$\psi(x = a, \mu) = \psi^{right}(\mu), \quad \text{where } \mu < 0 \tag{D.3}$$

With a quadrature set $\{\mu_m, w_m\}_{m=1}^{M}$ on $[-1, 1]$, we have the S_N equation,

$$\mu_m \frac{\partial \psi_m}{\partial x} + \sigma_t(x)\psi_m(x) = \frac{\sigma_s(x)}{2} \sum_{m'=1}^{M} w_{m'}\psi_{m'}(x) + \frac{Q(x)}{2} \tag{D.4}$$

Arbitrarily distribute $N + 1$ points $\left\{x_{i+1/2}\right\}_{i=0}^{N}$ in the domain $[0, a]$, and they satisfy

$$\begin{aligned} &x_{1/2} = 0 \\ &x_{N+1/2} = a \\ &x_{1/2} < x_{3/2} < x_{5/2} < \cdots < x_{N-1/2} < x_{N+1/2} \end{aligned} \tag{D.5}$$

Suppose the cross sections are piece-wise constant, and the cross section discontinuity points are in the point set. And also suppose the source is piece-wise linear function, and the source discontinuity points are also in the set. These points form N elements $\{K_i\}_{i=1}^{N}$, where

$$K_i = \left[x_{i-1/2}, x_{i+1/2}\right] \tag{D.6}$$

We also define

$$\Delta x_i = x_{i+1/2} - x_{i-1/2} \tag{D.7}$$

$$x_i = \frac{x_{i+1/2} + x_{i-1/2}}{2} \tag{D.8}$$

With these definitions, the cross sections is constant in each element. Define shape functions on element K_i:

$$b_i^L(x) = b^L(\hat{x}) = b^L \circ F_i^{-1}(x) \tag{D.9}$$

$$b_i^R(x) = b^R(\hat{x}) = b^R \circ F_i^{-1}(x) \tag{D.10}$$

where

$$b^L(\hat{x}) = \frac{1-\hat{x}}{2} \tag{D.11}$$

$$b^R(\hat{x}) = \frac{1+\hat{x}}{2} \tag{D.12}$$

$$x = F_i(\hat{x}) = x_{i-1/2} b^L(\hat{x}) + x_{i+1/2} b^R(\hat{x}) = x_i + \frac{\Delta x_i}{2}\hat{x} \tag{D.13}$$

we get further

$$\hat{x} = F_i^{-1}(x) = \frac{2}{\Delta x_i}(x - x_i) \tag{D.14}$$

$$b_i^L(x) = \frac{x_{i+1/2} - x}{\Delta x_i} \tag{D.15}$$

$$b_i^R(x) = \frac{x - x_{i-1/2}}{\Delta x_i} \tag{D.16}$$

$$\frac{db_i^L}{dx}(x) = -\frac{1}{\Delta x_i}; \quad \frac{db_i^R}{dx}(x) = \frac{1}{\Delta x_i} \tag{D.17}$$

$$\mathbf{M}_i \equiv \int_{x_{i-1/2}}^{x_{i+1/2}} \begin{bmatrix} b_i^L(x) \\ b_i^R(x) \end{bmatrix} \begin{bmatrix} b_i^L(x) & b_i^R(x) \end{bmatrix} dx = \frac{\Delta x_i}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{D.18}$$

$$\mathbf{G}_i \equiv \int_{x_{i-1/2}}^{x_{i+1/2}} \frac{d}{dx} \begin{bmatrix} b_i^L(x) \\ b_i^R(x) \end{bmatrix} \begin{bmatrix} b_i^L(x) & b_i^R(x) \end{bmatrix} dx = \frac{1}{2} \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \tag{D.19}$$

Assume solution in K_i

$$\psi_{m,i}(x) = \psi_{m,i}^L b_i^L(x) + \psi_{m,i}^R b_i^R(x), \qquad m = 1, \cdots, M \tag{D.20}$$

Or use vertex-based definition

$$\psi_{m,i}(x) = \psi^R_{m,i-1/2} b^L_i(x) + \psi^L_{m,i+1/2} b^R_i(x), \qquad m = 1, \cdots, M \tag{D.21}$$

substitute it into Eq. (D.4) and test with two shape functions,

$$\begin{aligned} &\mu_m \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{\psi}_{m,i-1/2} \\ \hat{\psi}_{m,i+1/2} \end{bmatrix} - \frac{\mu_m}{2} \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \psi^L_{m,i} \\ \psi^R_{m,i} \end{bmatrix} + \frac{\sigma_{t,i}\Delta x_i}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} \psi^L_{m,i} \\ \psi^R_{m,i} \end{bmatrix} \\ &= \frac{\sigma_{s,i}\Delta x_i}{12} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \sum_{m'=1}^{M} w_{m'} \begin{bmatrix} \psi^L_{m',i} \\ \psi^R_{m',i} \end{bmatrix} + \frac{\Delta x_i}{12} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} Q^L_i \\ Q^R_i \end{bmatrix} \end{aligned} \tag{D.22}$$

Now apply the upwind scheme:

$$\hat{\psi}_{m,i+1/2} = \begin{cases} \psi^R_{m,i} & \text{if } i > 0 \\ \psi^{left}_m & \text{if } i = 0 \end{cases} \tag{D.23}$$

when $\mu_m > 0$.

$$\hat{\psi}_{m,i+1/2} = \begin{cases} \psi^L_{m,i+1} & \text{if } i < N \\ \psi^{right}_m & \text{if } i = N \end{cases} \tag{D.24}$$

when $\mu_m < 0$. This gives us the proper coupling.

We use the same variational derivation to obtain the conforming diffusion scheme for this 1-D problem. We write down the variational form for the 1-D transport

problem,

$$\sum_{m=1}^{M/2} 2w_m \sum_{i=1}^{N-1} |\mu_m| \, \psi^{*R}_{m,i+1/2}(\psi^R_{m,i+1/2} - \psi^L_{m,i+1/2}) + \sum_{m=1}^{M/2} 2w_m \, |\mu_m| \, \psi^{*R}_{m,1/2}\psi^R_{m,1/2} +$$
$$\sum_{m=M/2+1}^{M} 2w_m \sum_{i=1}^{N-1} |\mu_m| \, \psi^{*L}_{m,i+1/2}(\psi^L_{m,i+1/2} - \psi^R_{m,i+1/2}) +$$
$$\sum_{m=M/2+1}^{M} 2w_m \, |\mu_m| \, \psi^{*L}_{m,N+1/2}\psi^L_{m,N+1/2} +$$
$$\sum_{m=1}^{M} 2w_m \sum_{i=1}^{N} (\mu_m \frac{\partial \psi_{m,i}}{\partial x} + \sigma_t \psi_{m,i}, \psi^*_{m,i})_i = \sum_{m=1}^{M} 2w_m \sum_{i=1}^{N} (\frac{\sigma_s}{2}\phi_i + \frac{Q_i}{2}, \psi^*_{m,i})_i +$$
$$\sum_{m=1}^{M/2} 2w_m \, |\mu_m| \, \psi^{*R}_{m,1/2}\psi^{left}_m + \sum_{m=M/2+1}^{M} 2w_m \, |\mu_m| \, \psi^{*L}_{m,N+1/2}\psi^{right}_m$$

Then we directly write down the diffusion conforming form

$$b_{DCF}(\phi, \phi^*) = (\sigma_a\phi, \phi^*)_{\mathcal{D}} + (Dd_x\phi, d_x\phi^*)_{\mathcal{D}} +$$
$$\frac{1}{4}([\![\phi]\!], [\![\phi^*]\!])_{E_h^i} + ([\![\phi]\!], \{\!\{Dd_x\phi^*\}\!\})_{E_h^i} + (\{\!\{Dd_x\phi\}\!\}, [\![\phi^*]\!])_{E_h^i} - \frac{9}{8}([\![Dd_x\phi]\!], [\![Dd_x\phi^*]\!])_{E_h^i} +$$
$$\frac{1}{4}(\phi, \phi^*)^R_{1/2} + \frac{1}{2}(\phi, Dd_x\phi^*)^R_{1/2} + \frac{1}{2}(Dd\phi_x, \phi^*)^R_{1/2} - \frac{9}{8}(Dd_x\phi, Dd_x\phi^*)^R_{1/2} +$$
$$\frac{1}{4}(\phi, \phi^*)^L_{N+1/2} - \frac{1}{2}(\phi, Dd_x\phi^*)^L_{N+1/2} - \frac{1}{2}(Dd_x\phi, \phi^*)^L_{N+1/2} - \frac{9}{8}(Dd_x\phi, Dd_x\phi^*)^L_{N+1/2} \tag{D.25}$$

$$l(\phi^*) = (Q, \phi^*)_{\mathcal{D}} +$$
$$(J^{left}, \phi^*)^R_{1/2} + (\varsigma^{left}, Dd_x\phi^*)^R_{1/2} + (J^{right}, \phi^*)^L_{N+1/2} + (\varsigma^{right}, Dd_x\phi^*)^L_{N+1/2} \tag{D.26}$$

where

$$J^{left} = \sum_{m=1}^{M/2} w_m \left|\mu_m\right| \psi_m^{left} \tag{D.27}$$

$$\varsigma^{left} = \sum_{m=1}^{M/2} 3w_m \left|\mu_m\right| \mu_m \psi_m^{left} \tag{D.28}$$

$$J^{right} = \sum_{m=M/2+1}^{M} w_m \left|\mu_m\right| \psi_m^{right} \tag{D.29}$$

$$\varsigma^{right} = \sum_{m=M/2+1}^{M} 3w_m \left|\mu_m\right| \mu_m \psi_m^{right} \tag{D.30}$$

E_h^i is the point set $\left\{x_{i+1/2}\right\}_{i=1}^{N-1}$. We have following definitions to understand the

formula:

$$(f,g)_{\mathcal{D}} = \sum_{i=1}^{N} (f,g)_i \tag{D.31}$$

$$(f,g)_i = \int_{x_{i-1/2}}^{x_{i+1/2}} f \cdot g \; dx \tag{D.32}$$

$$(f,g)_{E_h^i} = \sum_{i=1}^{N-1} (f,g)_{i+1/2} \tag{D.33}$$

$$(f,g)_{i+1/2} = f_{i+1/2} g_{i+1/2} \tag{D.34}$$

$$(f,g)^R_{i+1/2} = f^R_{i+1/2} g^R_{i+1/2} \tag{D.35}$$

$$(f,g)^L_{i+1/2} = f^L_{i+1/2} g^L_{i+1/2} \tag{D.36}$$

$$[\![\phi]\!]_{i+1/2} = \phi^R_{i+1/2} - \phi^L_{i+1/2} \tag{D.37}$$

$$[\![\phi^*]\!]_{i+1/2} = \phi^{*R}_{i+1/2} - \phi^{*L}_{i+1/2} \tag{D.38}$$

$$[\![Dd_x\phi]\!]_{i+1/2} = \left[D\frac{d\phi}{dx}\right]^R_{i+1/2} - \left[D\frac{d\phi}{dx}\right]^L_{i+1/2} \tag{D.39}$$

$$[\![Dd_x\phi^*]\!]_{i+1/2} = \left[D\frac{d\phi^*}{dx}\right]^R_{i+1/2} - \left[D\frac{d\phi^*}{dx}\right]^L_{i+1/2} \tag{D.40}$$

$$\{\!\{Dd_x\phi\}\!\}_{i+1/2} = \frac{1}{2}\left(\left[D\frac{d\phi}{dx}\right]^R_{i+1/2} + \left[D\frac{d\phi}{dx}\right]^L_{i+1/2}\right) \tag{D.41}$$

$$\{\!\{Dd_x\phi^*\}\!\}_{i+1/2} = \frac{1}{2}\left(\left[D\frac{d\phi^*}{dx}\right]^R_{i+1/2} + \left[D\frac{d\phi^*}{dx}\right]^L_{i+1/2}\right) \tag{D.42}$$

We copy the multi-dimensional diffusion conforming form to here:

$$\begin{aligned} b_{DCF}(\Phi,\Phi^*) =&(\sigma_a\Phi,\Phi^*)_{\mathcal{D}} + (D\vec{\nabla}\Phi,\vec{\nabla}\Phi^*)_{\mathcal{D}} \\ &+\frac{1}{4}([\![\Phi]\!],[\![\Phi^*]\!])_{E_h^i} + ([\![\Phi]\!],\{\!\{D\partial_n\Phi^*\}\!\})_{E_h^i} + (\{\!\{D\partial_n\Phi\}\!\},[\![\Phi^*]\!])_{E_h^i} \\ &+\frac{1}{4}(\Phi,\Phi^*)_{\partial\mathcal{D}^d} - \frac{1}{2}(\Phi,D\partial_n\Phi^*)_{\partial\mathcal{D}^d} - \frac{1}{2}(D\partial_n\Phi,\Phi)_{\partial\mathcal{D}^d} \\ &-\frac{9}{16}([\![D\vec{\nabla}\Phi]\!],[\![D\vec{\nabla}\Phi^*]\!])_{E_h^i} - \frac{9}{16}([\![D\partial_n\Phi]\!],[\![D\partial_n\Phi^*]\!])_{E_h^i} \\ &-\frac{9}{16}(D\vec{\nabla}\Phi,D\vec{\nabla}\Phi^*)_{\partial\mathcal{D}^d} - \frac{9}{16}(D\partial_n\Phi,D\partial_n\Phi^*)_{\partial\mathcal{D}^d} \\ l_{DCF}(\Phi^*) =&(Q_0,\Phi^*)_{\mathcal{D}} + (J^{inc},\Phi^*)_{\partial\mathcal{D}^d} - (\vec{\Upsilon}^{inc},D\vec{\nabla}\Phi^*)_{\partial\mathcal{D}^d} \end{aligned} \tag{D.43}$$

where

$$\begin{aligned} J^{inc} &= \sum_{\vec{\Omega}_m\cdot\vec{n}(\vec{r}_b)<0} w_m \left|\vec{\Omega}_m\cdot\vec{n}(\vec{r}_b)\right| \Psi_m^{inc} \\ \vec{\Upsilon}^{inc} &= -\sum_{\vec{\Omega}_m\cdot\vec{n}(\vec{r}_b)<0} 3w_m\vec{\Omega}_m \left|\vec{\Omega}_m\cdot\vec{n}(\vec{r}_b)\right| \Psi_m^{inc} \end{aligned} \tag{D.44}$$

We can see that, with following

$$D\vec{\nabla}\Phi = Dd_x\phi \tag{D.45}$$

$$D\partial_n\Phi = Dd_x\phi \qquad \text{on } E_h^i \tag{D.46}$$

$$D\partial_n\Phi = Dd_x\phi \qquad \text{on } x_{N+1/2} \tag{D.47}$$

$$D\partial_n\Phi = -Dd_x\phi \qquad \text{on } x_{1/2} \tag{D.48}$$

We can obtain the 1-D form from the multi-dimensional formula.

I will present a different way to assemble the system. Let us consider two element

adjacent with each other noted with 1 for the left and 2 for the right.

$$
\begin{aligned}
(\sigma_a\phi, \phi^*)_D &= \phi_1^{*T}\sigma_{a,1}\Delta x_1 \mathbf{M}\phi_1 + \phi_2^{*T}\sigma_{a,2}\Delta x_2 \mathbf{M}\phi_2 \\
(Dd_x\phi, d_x\phi^*)_D &= \phi_1^{*T}\frac{D_1}{\Delta x_1}\mathbf{S}\phi_1 + \phi_2^{*T}\frac{D_2}{\Delta x_2}\mathbf{S}\phi_2 \\
\frac{1}{4}([\phi], [\phi^*])_{E_h^i} &= \frac{1}{4}(\phi_{3/2}^R - \phi_{3/2}^L)(\phi_{3/2}^{*R} - \phi_{3/2}^{*L}) \\
&= \frac{1}{4}\phi_{3/2}^L\phi_{3/2}^{*L} + \frac{1}{4}\phi_{3/2}^R\phi_{3/2}^{*R} - \frac{1}{4}\phi_{3/2}^R\phi_{3/2}^{*L} - \frac{1}{4}\phi_{3/2}^L\phi_{3/2}^{*R} \\
&= \frac{1}{4}\phi_1^{*T}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix}\phi_1 + \frac{1}{4}\phi_2^{*T}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix}\phi_2 \\
&\quad - \frac{1}{4}\phi_1^{*T}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix}\phi_2 - \frac{1}{4}\phi_2^{*T}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix}\phi_1 \\
&= \phi_1^{*T}\mathbf{E}_{11}^1\phi_1 + \phi_2^{*T}\mathbf{E}_{22}^1\phi_2 + \phi_1^{*T}\mathbf{E}_{12}^1\phi_2 + \phi_2^{*T}\mathbf{E}_{21}^1\phi_1
\end{aligned}
$$

$$
\begin{aligned}
(\{Dd_x\phi\}, [\phi^*])_{E_h^i} &= \frac{1}{2}(\left[D\frac{d\phi}{dx}\right]_{3/2}^R + \left[D\frac{d\phi}{dx}\right]_{3/2}^L)(\phi_{3/2}^{*R} - \phi_{3/2}^{*L}) \\
&= -\frac{1}{2}\left[D\frac{d\phi}{dx}\right]_{3/2}^L\phi_{3/2}^{*L} + \frac{1}{2}\left[D\frac{d\phi}{dx}\right]_{3/2}^R\phi_{3/2}^{*R} \\
&\quad - \frac{1}{2}\left[D\frac{d\phi}{dx}\right]_{3/2}^R\phi_{3/2}^{*L} + \frac{1}{2}\left[D\frac{d\phi}{dx}\right]_{3/2}^L\phi_{3/2}^{*R} \\
&= -\frac{1}{2}\phi_1^{*T}\begin{bmatrix}0\\1\end{bmatrix}\frac{D_1}{\Delta x_1}\begin{bmatrix}-1 & 1\end{bmatrix}\phi_1 + \frac{1}{2}\phi_2^{*T}\begin{bmatrix}1\\0\end{bmatrix}\frac{D_2}{\Delta x_2}\begin{bmatrix}-1 & 1\end{bmatrix}\phi_2 \\
&\quad - \frac{1}{2}\phi_1^{*T}\begin{bmatrix}0\\1\end{bmatrix}\frac{D_2}{\Delta x_2}\begin{bmatrix}-1 & 1\end{bmatrix}\phi_2 + \frac{1}{2}\phi_2^{*T}\begin{bmatrix}1\\0\end{bmatrix}\frac{D_1}{\Delta x_1}\begin{bmatrix}-1 & 1\end{bmatrix}\phi_1 \\
&= \phi_1^{*T}\frac{D_1}{\Delta x_1}\mathbf{E}_{11}^2\phi_1 + \phi_2^{*T}\frac{D_2}{\Delta x_2}\mathbf{E}_{22}^2\phi_2 + \phi_1^{*T}\frac{D_2}{\Delta x_2}\mathbf{E}_{12}^2\phi_2 + \phi_2^{*T}\frac{D_1}{\Delta x_1}\mathbf{E}_{21}^2\phi_1
\end{aligned}
$$

$$
\begin{aligned}
\left([\phi], \{Dd_x\phi^*\}\right)_{E_h^i} &= \frac{1}{2}(\phi_{3/2}^R - \phi_{3/2}^L)\left(\left[D\frac{d\phi^*}{dx}\right]_{3/2}^R + \left[D\frac{d\phi^*}{dx}\right]_{3/2}^L\right) \\
&= -\frac{1}{2}\left[D\frac{d\phi^*}{dx}\right]_{3/2}^L \phi_{3/2}^L + \frac{1}{2}\left[D\frac{d\phi^*}{dx}\right]_{3/2}^R \phi_{3/2}^R \\
&\quad + \frac{1}{2}\left[D\frac{d\phi^*}{dx}\right]_{3/2}^L \phi_{3/2}^R - \frac{1}{2}\left[D\frac{d\phi^*}{dx}\right]_{3/2}^R \phi_{3/2}^L \\
&= -\frac{1}{2}\frac{D_1}{\Delta x_1}\phi_1^{*T}\begin{bmatrix}-1\\1\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix}\phi_1 + \frac{1}{2}\frac{D_2}{\Delta x_2}\phi_2^{*T}\begin{bmatrix}-1\\1\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix}\phi_2 \\
&\quad - \frac{1}{2}\frac{D_1}{\Delta x_1}\phi_1^{*T}\begin{bmatrix}-1\\1\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix}\phi_2 + \frac{1}{2}\frac{D_2}{\Delta x_2}\phi_2^{*T}\begin{bmatrix}-1\\1\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix}\phi_1 \\
&= \phi_1^{*T}\frac{D_1}{\Delta x_1}\mathbf{E}_{11}^{2T}\phi_1 + \phi_2^{*T}\frac{D_2}{\Delta x_2}\mathbf{E}_{22}^{2T}\phi_2 + \phi_1^{*T}\frac{D_1}{\Delta x_1}\mathbf{E}_{21}^{2T}\phi_2 + \phi_2^{*T}\frac{D_2}{\Delta x_2}\mathbf{E}_{12}^{2T}\phi_1
\end{aligned}
$$

$$
\begin{aligned}
-\frac{9}{8}\left([Dd_x\phi], [Dd_x\phi^*]\right)_{E_h^i} &= -\frac{9}{8}\left(\left[D\frac{d\phi}{dx}\right]_{3/2}^R - \left[D\frac{d\phi}{dx}\right]_{3/2}^L\right)\left(\left[D\frac{d\phi^*}{dx}\right]_{3/2}^R - \left[D\frac{d\phi^*}{dx}\right]_{3/2}^L\right) \\
&= -\frac{9}{8}\left[D\frac{d\phi}{dx}\right]_{3/2}^L\left[D\frac{d\phi^*}{dx}\right]_{3/2}^L - \frac{9}{8}\left[D\frac{d\phi}{dx}\right]_{3/2}^R\left[D\frac{d\phi^*}{dx}\right]_{3/2}^R \\
&\quad + \frac{9}{8}\left[D\frac{d\phi}{dx}\right]_{3/2}^R\left[D\frac{d\phi^*}{dx}\right]_{3/2}^L + \frac{9}{8}\left[D\frac{d\phi}{dx}\right]_{3/2}^L\left[D\frac{d\phi^*}{dx}\right]_{3/2}^R \\
&= -\frac{9}{8}\frac{D_1}{\Delta x_1}\phi_1^{*T}\begin{bmatrix}-1\\1\end{bmatrix}\frac{D_1}{\Delta x_1}\begin{bmatrix}-1 & 1\end{bmatrix}\phi_1 - \frac{9}{8}\frac{D_2}{\Delta x_2}\phi_2^{*T}\begin{bmatrix}-1\\1\end{bmatrix}\frac{D_2}{\Delta x_2}\begin{bmatrix}-1 & 1\end{bmatrix}\phi_2 \\
&\quad + \frac{9}{8}\frac{D_1}{\Delta x_1}\phi_1^{*T}\begin{bmatrix}-1\\1\end{bmatrix}\frac{D_2}{\Delta x_2}\begin{bmatrix}-1 & 1\end{bmatrix}\phi_2 + \frac{9}{8}\frac{D_2}{\Delta x_2}\phi_2^{*T}\begin{bmatrix}-1\\1\end{bmatrix}\frac{D_2}{\Delta x_2}\begin{bmatrix}-1 & 1\end{bmatrix}\phi_1 \\
&= \phi_1^{*T}(\frac{D_1}{\Delta x_1})^2\mathbf{E}_{11}^3\phi_1 + \phi_2^{*T}(\frac{D_2}{\Delta x_2})^2\mathbf{E}_{22}^3\phi_2 \\
&\quad + \phi_1^{*T}\frac{D_1}{\Delta x_1}\frac{D_2}{\Delta x_2}\mathbf{E}_{12}^3\phi_2 + \phi_2^{*T}\frac{D_1}{\Delta x_1}\frac{D_2}{\Delta x_2}\mathbf{E}_{21}^3\phi_1
\end{aligned}
$$

where

$$\mathbf{M} = \frac{1}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$\mathbf{E}^1_{11} = \frac{1}{4}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{E}^1_{22} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{E}^1_{12} = -\frac{1}{4}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix}$$

$$\mathbf{E}^1_{21} = -\frac{1}{4}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{E}^2_{11} = \mathbf{E}^2_{12} = -\frac{1}{2}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} -1 & 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix}$$

$$\mathbf{E}^2_{22} = \mathbf{E}^2_{21} = \frac{1}{2}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} -1 & 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{E}^3_{11} = \mathbf{E}^3_{22} = -\frac{9}{8}\begin{bmatrix} -1 \\ 1 \end{bmatrix}\begin{bmatrix} -1 & 1 \end{bmatrix} = \frac{9}{8}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\mathbf{E}^3_{12} = \mathbf{E}^3_{21} = \frac{9}{8}\begin{bmatrix} -1 \\ 1 \end{bmatrix}\begin{bmatrix} -1 & 1 \end{bmatrix} = \frac{9}{8}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Note that $\mathbf{E}_{12}^{1} = \mathbf{E}_{21}^{1T}$. So the final system without the boundary treatment is

$$\begin{bmatrix} \phi_1^* & \phi_2^* \end{bmatrix} \mathbf{A} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \tag{D.49}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \tag{D.50}$$

$$\mathbf{A}_{11} = \sigma_{a,1}\Delta x_1 \mathbf{M} + \frac{D_1}{\Delta x_1}\mathbf{S} + \mathbf{E}_{11}^{1} + \frac{D_1}{\Delta x_1}(\mathbf{E}_{11}^{2} + \mathbf{E}_{11}^{2T}) + (\frac{D_1}{\Delta x_1})^2 \mathbf{E}_{11}^{3} \tag{D.51}$$

$$\mathbf{A}_{12} = \mathbf{E}_{12}^{1} + \frac{D_2}{\Delta x_2}\mathbf{E}_{12}^{2} + \frac{D_1}{\Delta x_1}\mathbf{E}_{21}^{2T} + \frac{D_1}{\Delta x_1}\frac{D_2}{\Delta x_2}\mathbf{E}_{12}^{3} \tag{D.52}$$

$$\mathbf{A}_{21} = \mathbf{E}_{21}^{1} + \frac{D_1}{\Delta x_1}\mathbf{E}_{21}^{2} + \frac{D_2}{\Delta x_2}\mathbf{E}_{12}^{2T} + \frac{D_1}{\Delta x_1}\frac{D_2}{\Delta x_2}\mathbf{E}_{21}^{3} \tag{D.53}$$

$$\mathbf{A}_{22} = \sigma_{a,2}\Delta x_2 \mathbf{M} + \frac{D_2}{\Delta x_2}\mathbf{S} + \mathbf{E}_{22}^{1} + \frac{D_2}{\Delta x_2}(\mathbf{E}_{22}^{2} + \mathbf{E}_{22}^{2T}) + (\frac{D_2}{\Delta x_2})^2 \mathbf{E}_{22}^{3} \tag{D.54}$$

Matrix $\mathbf{A}$ is symmetric. If there is another cell on the right of cell 2 denoted with 3, the resulting matrix will be

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{0} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{0} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} \tag{D.55}$$

where

$$\mathbf{A}_{11} = \sigma_{a,1}\Delta x_1 \mathbf{M} + \frac{D_1}{\Delta x_1}\mathbf{S} + \mathbf{E}^1_{11} + \frac{D_1}{\Delta x_1}(\mathbf{E}^2_{11} + \mathbf{E}^{2T}_{11}) + (\frac{D_1}{\Delta x_1})^2 \mathbf{E}^3_{11} \tag{D.56}$$

$$\mathbf{A}_{12} = \mathbf{E}^1_{12} + \frac{D_2}{\Delta x_2}\mathbf{E}^2_{12} + \frac{D_1}{\Delta x_1}\mathbf{E}^{2T}_{21} + \frac{D_1}{\Delta x_1}\frac{D_2}{\Delta x_2}\mathbf{E}^3_{12} \tag{D.57}$$

$$\mathbf{A}_{21} = \mathbf{E}^1_{21} + \frac{D_1}{\Delta x_1}\mathbf{E}^2_{21} + \frac{D_2}{\Delta x_2}\mathbf{E}^{2T}_{12} + \frac{D_1}{\Delta x_1}\frac{D_2}{\Delta x_2}\mathbf{E}^3_{21} \tag{D.58}$$

$$\mathbf{A}_{22} = \sigma_{a,2}\Delta x_2 \mathbf{M} + \frac{D_2}{\Delta x_2}\mathbf{S} + \mathbf{E}^1_{22} + \frac{D_2}{\Delta x_2}(\mathbf{E}^2_{22} + \mathbf{E}^{2T}_{22}) + (\frac{D_2}{\Delta x_2})^2 \mathbf{E}^3_{22} + \mathbf{E}^1_{11} + \frac{D_2}{\Delta x_2}(\mathbf{E}^2_{11} + \mathbf{E}^{2T}_{11}) + (\frac{D_2}{\Delta x_2})^2 \mathbf{E}^3_{11} \tag{D.59}$$

$$\mathbf{A}_{32} = \mathbf{E}^1_{21} + \frac{D_2}{\Delta x_2}\mathbf{E}^2_{21} + \frac{D_3}{\Delta x_3}\mathbf{E}^{2T}_{12} + \frac{D_2}{\Delta x_2}\frac{D_3}{\Delta x_3}\mathbf{E}^3_{21} \tag{D.60}$$

$$\mathbf{A}_{33} = \sigma_{a,3}\Delta x_3 \mathbf{M} + \frac{D_3}{\Delta x_3}\mathbf{S} + \mathbf{E}^1_{22} + \frac{D_3}{\Delta x_3}(\mathbf{E}^2_{22} + \mathbf{E}^{2T}_{22}) + (\frac{D_3}{\Delta x_3})^2 \mathbf{E}^3_{22} \tag{D.61}$$

This is one way where we basically are considering a bunch of small 2-by-2 system and then summing them together. We can also consider each cell with two side vertices, i.e., each row of the global system. 11 is right self-coupling, 22 is left self-coupling, 12 is right vertex coupling, 21 is left vertex coupling. This way is better in multi-dimensional situation.

$$\mathbf{E}_{11} = \mathbf{E}_R \tag{D.62}$$

$$\mathbf{E}_{22} = \mathbf{E}_L \tag{D.63}$$

$$\mathbf{E}_{12} = \mathbf{E}_{RC} \tag{D.64}$$

$$\mathbf{E}_{21} = \mathbf{E}_{LC} \tag{D.65}$$

APPENDIX E

PRECONDITIONED CG METHOD WITH EISENSTAT TRICK

For complete, we present the algorithm proposed by Eisenstat for the PCG (Preconditioned Conjugate Gradient) method [114].

Consider the linear system,

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{E.1}$$

where the matrix A is SPD (symmetric and positive definite).

Applying PCG with the preconditioner

$$\mathcal{M} = (\tilde{\mathrm{D}} + \mathrm{L})\tilde{\mathrm{D}}^{-1}(\tilde{\mathrm{D}} + \mathrm{L})^T \tag{E.2}$$

to the system is equivalent to applying PCG with

$$\mathcal{M} = \tilde{\mathrm{D}}^{-1} \tag{E.3}$$

to the following modified system

$$\left[(\tilde{\mathrm{D}} + \mathrm{L})^{-1}\mathbf{A}(\tilde{\mathrm{D}} + \mathrm{L})^{-T}\right]\left[(\tilde{\mathrm{D}} + \mathrm{L})^T\mathbf{x}\right] = (\tilde{\mathrm{D}} + \mathrm{L})^{-1}\mathbf{b} \tag{E.4}$$

or

$$\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}} \tag{E.5}$$

where L is lower-triangular and D is the positive block diagonal.

Now the algorithm of PCG is:

Algorithm 4 Preconditioned CG with Eisenstat's trick

1: $\hat{\mathbf{r}}_0 = (\tilde{\mathrm{D}} + \mathrm{L})^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_0)$

2: $\hat{\mathbf{p}}_0 = \hat{\mathbf{r}}'_0 = \tilde{\mathrm{D}}\hat{\mathbf{r}}_0$

3: **for** $k = 0 : \underline{maxiter}$ **do**

4: $\quad \hat{a}_k = \frac{(\hat{\mathbf{r}}_k, \hat{\mathbf{r}}'_k)}{(\hat{\mathbf{p}}_k, \hat{\mathbf{A}}\hat{\mathbf{p}}_k)}$

5: $\quad \mathbf{x}_{k+1} = \mathbf{x}_k + \hat{a}_k(\tilde{\mathrm{D}} + \mathrm{L})^{-T}\hat{\mathbf{p}}_k$

6: $\quad \hat{\mathbf{r}}_{k+1} = \hat{\mathbf{r}}_k + \hat{a}_k\hat{\mathbf{A}}\hat{\mathbf{p}}_k$

7: $\quad \hat{\mathbf{r}}'_{k+1} = \tilde{\mathrm{D}}\hat{\mathbf{r}}_{k+1}$

8: $\quad \hat{b}_k = \frac{(\hat{\mathbf{r}}_{k+1}, \hat{\mathbf{r}}'_{k+1})}{(\hat{\mathbf{r}}_k, \hat{\mathbf{r}}'_k)}$

9: $\quad$ **if** $\frac{(\hat{\mathbf{r}}_{k+1}, \hat{\mathbf{r}}'_{k+1})}{(\hat{\mathbf{r}}_0, \hat{\mathbf{r}}'_0)} < \underline{\mathrm{tol}}$ **then**

10: $\quad\quad$ Exit

11: $\quad$ **end if**

12: $\quad \hat{\mathbf{p}}_{k+1} = \hat{\mathbf{r}}'_{k+1} + \hat{b}_k\hat{\mathbf{p}}_k$

13: **end for**

The trick is that the matrix-vector product $\hat{\mathbf{A}}\hat{\mathbf{p}}_k$ can be computed efficiently by splitting it into two stages:

$$\hat{\mathbf{t}}_k = (\tilde{\mathrm{D}} + \mathrm{L})^{-T}\hat{\mathbf{p}}_k, \tag{E.6}$$

$$\hat{\mathbf{A}}\hat{\mathbf{p}}_k = \hat{\mathbf{t}}_k + (\tilde{\mathrm{D}} + \mathrm{L})^{-}(\hat{\mathbf{p}}_k - \mathrm{K}\hat{\mathbf{t}}_k) \tag{E.7}$$

where $\mathrm{K} = 2\tilde{\mathrm{D}} - \mathrm{D}$.

APPENDIX F

MATHEMATICA NOTEBOOK FOR ELEMENTARY MATRICES

Turn off spelling warnings

```
In[1]:= Off[General :: spell, General :: spell1]
```

Lobatto and kernel functions:

```
In[2]:= Lobatto[k_, x_] :=
          If[k == 0, (1 - x)/2, If[k == 1, (1 + x)/2, (LegendreP[k, x] - LegendreP[k - 2, x])/Sqrt[2 * (2 * k - 1)]]]

In[3]:= Kernel[k_, x_] := Lobatto[k + 2, x]/(Lobatto[0, x] * Lobatto[1, x])
```

Barycentric coordinates:

```
In[4]:= lamda3[x_, y_] := (1 + y)/2

In[5]:= lamda1[x_, y_] := -(x + y)/2

In[6]:= lamda2[x_, y_] := (1 + x)/2
```

Hierarchical basis functions on the reference triangle:

```
In[7]:= basis[p_, x_, y_] :=
          Module[{t, i, j, j2},
            Simplify[If[p == 0, {Kernel[0, 0]}, t = {lamda1[x, y], lamda2[x, y], lamda3[x, y]};
                For[i = 1, i < p,

                  t = Append[t, lamda2[x, y] * lamda3[x, y] *
                        Kernel[i - 1, lamda3[x, y] - lamda2[x, y]]];

                  t = Append[t, lamda3[x, y] * lamda1[x, y] *
                        Kernel[i - 1, lamda1[x, y] - lamda3[x, y]]];

                  t = Append[t, lamda1[x, y] * lamda2[x, y] *
                        Kernel[i - 1, lamda2[x, y] - lamda1[x, y]]];

                  For[j = 1, j < i, j2 = i - j;
                    t = Append[t, lamda1[x, y] * lamda2[x, y] * lamda3[x, y] *
                          Kernel[j - 1, lamda3[x, y] - lamda2[x, y]] *
                          Kernel[j2 - 1, lamda2[x, y] - lamda1[x, y]]]; j++]; i++]; {t}]]]
```

Get Jacobin:

J={{dx/dxix,dx/dxiy},{dy/dxix,dy/dxiy}}
(note that the determinit of Jacobin is equal to half of the triangle area)

```
In[8]:= coef = Solve[{x1 == a + b * (-1) + c * (-1), y1 == dd + ee * (-1) + ff * (-1),
            x2 == a + b * (1) + c * (-1), y2 == dd + ee * (1) + ff * (-1), x3 == a + b * (-1) + c * (1),
            y3 == dd + ee * (-1) + ff * (1)}, {a, b, c, dd, ee, ff}]
Out[8]= {{a → -1/2 (-x2 - x3), b → -1/2 (x1 - x2), c → -1/2 (x1 - x3),
            dd → -1/2 (-y2 - y3), ee → -1/2 (y1 - y2), ff → -1/2 (y1 - y3)}}

In[9]:= (Jcb = {{b, c}, {ee, ff}} /. coef[[1]]) //MatrixForm
```

```
Out[9]= ( -1/2 (x1 - x2)   -1/2 (x1 - x3) )
        ( -1/2 (y1 - y2)   -1/2 (y1 - y3) )
```

Reference mass matrix:

```
In[10]:= mmatrix[basis_] := 6 * ∫_-1^1 ∫_-1^-x Transpose[basis].basisdydx
```

Two freaquently used matrices:

```
In[11]:= (TT = Simplify[Inverse[Transpose[Jcb]] * Det[Jcb]])//MatrixForm
Out[11]= ( 1/2 (-y1 + y3)      (y1 - y2)/2 )
         ( (x1 - x3)/2      1/2 (-x1 + x2) )
```

T=Inverse[Jcb].Inverse[Transpose[Jcb]]*Det[Jcb]

```
In[12]:= T = ( 2 r2          r1 - r3 - r2 )//MatrixForm
             ( r1 - r3 - r2      2 r3    )
Out[12]= ( 2 r2          r1 - r2 - r3 )
         ( r1 - r2 - r3      2 r3    )
```

Streaming matrx :

Note : det[Jcb] $*$ (omgx omgy).$Transpose[Inverse[Jcb]] == -3(t2, t3)$

(Two minus sign will be cancelled in the streaming matrix)

```
In[13]:= gmatrix[basis_] := Simplify[
           ∫_-1^1 ∫_-1^-x 3 * Transpose[Join[∂_x basis, ∂_y basis]] . ( t2 ) .basisdydx
                                                                     ( t3 )

           , -t2 - t3 == t1]
```

Stiffness matrix:

```
In[14]:= smatrix[basis_] :=
           Simplify[ ∫_-1^1 ∫_-1^-x Transpose[Join[∂_x basis, ∂_y basis]].T.Join[∂_x basis, ∂_y basis]dydx]
```

The type-1 edge matrix:

```
In[15]:= ematrixSelf[basis_, id_] :=
           3*
             ∫_-1^1 ((Transpose[basis].basis)/.If[id == 3, {x → t, y → -1},
                       If[id == 1, {x → -t, y → t}, {x → -1, y → -t}]])dt

In[16]:= ematrix[basis_, id_, iu_] :=
           3*
             ∫_-1^1 (Transpose[basis]/.If[id == 3, {x → t, y → -1},
                       If[id == 1, {x → -t, y → t}, {x → -1, y → -t}]] ) .
                  ( basis/.If[iu == 3, {x → -t, y → -1}, If[iu == 1, {x → t, y → -t}, {x → -1, y → t}]])
                dt
```

1-D shape functions and reference mass, prolongation matrix:

```
In[17]:= basis1D[p_, x_] :=
           Simplify[If[p == 0, {Kernel[0, 0]}, {Table[Lobatto[i, x], {i, 0, p}]}]]

In[18]:= mmatrix1D[basis1d_] := ∫_-1^1 Transpose[basis1d].basis1ddx

In[19]:= bmatrix1D[basis1d_, id_] :=
           Inverse[mmatrix1D[basis1d]].
             ∫_-1^1 Transpose[(basis1d/.x → t)].(basis1d/.(If[id == 1, {x → (t - 1)/2}, {x → (t + 1)/2}])) dt
```

Edge operation to obtain the norm derivative:

Get directional derivative on outward norm direction of a side with I-D of all basis functions:

```
In[20]:= sideDnorm[basis_, id_] :=
           Simplify[
             If[id == 3, 2/√t3 {{r3 + r2 - r1, -2 r3}},
                  If[id == 1, 2/√t1 {{r2 + r1 - r3, r3 + r1 - r2}}, 2/√t2 {{-2 r2, r3 + r2 - r1}}]]

               .Join[∂_x basis, ∂_y basis]/.
             If[id == 3, {x → t, y → -1}, If[id == 1, {x → -t, y → t}, {x → -1, y → -t}]]]
```

Get directional derivative on outward norm direction of side with ID in terms of side basis functions for all cell basis functions (i.e. DoFs):

```
In[21]:= sideDSolution[basis_, basis1D_, id_] :=
           Simplify[Inverse[mmatrix1D[basis1D]].
              ∫_-1^1 Transpose[basis1D/.x → t].sideDnorm[basis, id]dt]
```

The type-2 edge matrix:

```
In[22]:= sideDSolFull[basis_, id_] :=
           ∫_-1^1 ((Transpose[
                       Simplify[If[id == 3, {{r3 + r2 - r1, -2 r3}},
                            If[id == 1, {{r2 + r1 - r3, r3 + r1 - r2}}, {{-2 r2, r3 + r2 - r1}}]]

                          .Join[∂_x basis, ∂_y basis]]].basis)/.
                  If[id == 3, {x → t, y → -1}, If[id == 1, {x → -t, y → t}, {x → -1, y → -t}]])dt

In[23]:= sideDSolFullt[basis_, id_, iu_] :=
           ∫_-1^1 (Transpose[
                    (Simplify[If[id == 3, {{r3 + r2 - r1, -2 r3}},
                             If[id == 1, {{r2 + r1 - r3, r3 + r1 - r2}}, {{-2 r2, r3 + r2 - r1}}]]

                            .Join[∂_x basis, ∂_y basis]])/.
                      If[id == 3, {x → t, y → -1}, If[id == 1, {x → -t, y → t}, {x → -1, y → -t}]]].
                   (basis/.If[iu == 3, {x → -t, y → -1}, If[iu == 1, {x → t, y → -t}, {x → -1, y → t}]]))
             dt
```

The type-3 edge matrix:

```
In[24]:= sideDnFull[basis_, id_] :=
          ∫_-1^1 ((Transpose[
                      Simplify[If[id == 3, {{r3 + r2 - r1, -2 r3}},
                           If[id == 1, {{r2 + r1 - r3, r3 + r1 - r2}}, {{-2 r2, r3 + r2 - r1}}]]

                          .Join[∂x basis, ∂y basis]]].
                     Simplify[If[id == 3, {{r3 + r2 - r1, -2 r3}},
                          If[id == 1, {{r2 + r1 - r3, r3 + r1 - r2}}, {{-2 r2, r3 + r2 - r1}}]]

                        .Join[∂x basis, ∂y basis]])/.
                   If[id == 3, {x → t, y → -1}, If[id == 1, {x → -t, y → t}, {x → -1, y → -t}]])dt

In[25]:= sideDnFullt[basis_, id1_, id2_] :=
          ∫_-1^1 (((Transpose[Simplify[If[id1 == 3, {{r13 + r12 - r11, -2 r13}},
                            If[id1 == 1, {{r12 + r11 - r13, r13 + r11 - r12}}, {{-2 r12, r13 + r12 - r11}}]]

                           .Join[∂x basis, ∂y basis]]])/.
                      If[id1 == 3, {x → t, y → -1}, If[id1 == 1, {x → -t, y → t}, {x → -1, y → -t}]]).
                   ((Simplify[If[id2 == 3, {{r23 + r22 - r21, -2 r23}},
                           If[id2 == 1, {{r22 + r21 - r23, r23 + r21 - r22}}, {{-2 r22, r23 + r22 - r21}}]]

                          .Join[∂x basis, ∂y basis]])/.
                      If[id2 == 3, {x → -t, y → -1}, If[id2 == 1, {x → t, y → -t}, {x → -1, y → t}]]))dt
```

Edge operation to obtain the derivative in x and y directions:

```
In[26]:= sideDx[basis_, basis1D_, id_] :=
          Simplify[
            2/area Inverse[mmatrix1D[basis1D]].
              ∫_-1^1 Transpose[basis1D/.x → t].
                   (((y3 - y1)/2   (y1 - y2)/2 ).Join[∂x basis, ∂y basis]/.
                      If[id == 3, {x → t, y → -1}, If[id == 1, {x → -t, y → t}, {x → -1, y → -t}]])dt]

In[27]:= sideDy[basis_, basis1D_, id_] :=
          Simplify[
            2/area Inverse[mmatrix1D[basis1D]].
              ∫_-1^1 Transpose[basis1D/.x → t].
                   (((x1 - x3)/2   (x2 - x1)/2 ).Join[∂x basis, ∂y basis]/.
                      If[id == 3, {x → t, y → -1}, If[id == 1, {x → -t, y → t}, {x → -1, y → -t}]])dt]
```

The type-4 edge matrix:

Because :

$$\begin{pmatrix} \frac{y3-y1}{2} \\ \frac{y1-y2}{2} \end{pmatrix}\cdot\begin{pmatrix} \frac{y3-y1}{2} & \frac{y1-y2}{2} \end{pmatrix}+\begin{pmatrix} \frac{x1-x3}{2} \\ \frac{x2-x1}{2} \end{pmatrix}\cdot\begin{pmatrix} \frac{x1-x3}{2} & \frac{x2-x1}{2} \end{pmatrix}$$

$$== \frac{1}{4}\begin{pmatrix} a3 & -\frac{a1+a3-a2}{2} \\ -\frac{a1+a3-a2}{2} & a1 \end{pmatrix}, \textit{ we can have the following for one side,}$$

```
In[28]:= sideDxyFull[basis_, id_] :=
           Simplify[2 * If[id == 1, r1, If[id == 2, r2, r3]]
             ∫_-1^1 ((Transpose[Join[∂_x basis, ∂_y basis]] . {{2 r2, -(r2 + r3 - r1)}, {-(r2 + r3 - r1), 2 r3}} .
                   Join[∂_x basis, ∂_y basis]) /.
                 If[id == 3, {x → t, y → -1}, If[id == 1, {x → -t, y → t}, {x → -1, y → -t}]]) dt]
```

For the general fourth coupling matrix, we need to use coordinates of all vertices of two elements:

```
In[29]:= sideDxyFullt[basis_, id_, iu_] :=
           Simplify[
             ∫_-1^1 (2/area1 2/area2 If[id == 1, a1, If[id == 2, a2, a3]]/2
                 ((Transpose[Join[∂_x basis, ∂_y basis]] /.
                     If[id == 3, {x → t, y → -1}, If[id == 1, {x → -t, y → t}, {x → -1, y → -t}]]) .
                   ((((y13 - y11)/2)/((y11 - y12)/2)) . ((y23 - y21)/2  (y21 - y22)/2) +
                     (((x11 - x13)/2)/((x12 - x11)/2)) . ((x21 - x23)/2  (x22 - x21)/2)) .
                   (Join[∂_x basis, ∂_y basis] /. If[iu == 3, {x → -t, y → -1},
                       If[iu == 1, {x → t, y → -t}, {x → -1, y → t}]])) dt]
```

A special case of the type-4 edge coupling matrix: id==2 and iu==2

```
In[30]:= sideDxyFullt2[basis_] :=
           Simplify[
             ∫_-1^1 ((Transpose[Join[∂_x basis, ∂_y basis]] /. {x → -1, y → -t}) .
                 {{-4 r2 t2, 2 r2 (t2 + t3 - t1)}, {2 (r2 + r3 - r1) t2, -(r2 + r3 - r1) (t2 + t3 - t1) - 1}} .
                 (Join[∂_x basis, ∂_y basis] /. {x → -1, y → t})) dt]
```

Inverse[Jcb]*Det[Jcb]//MatrixForm = $\begin{pmatrix} \frac{y3-y1}{2} & \frac{x1-x3}{2} \\ \frac{y1-y2}{2} & \frac{x2-x1}{2} \end{pmatrix}$==TT

Get cell gridient in x or y direction (1 or 2):

(invm is the inverse of mass matrix)

```
In[31]:= cellGridient[invm_, basis_, id_] :=
           Simplify[
             2/area
              invm.
                Integrate[1, {-1, 1}, {-1, -x}] Transpose[basis].
                    {(Transpose[Join[∂x basis, ∂y basis]].{{(y3 - y1)/2, (x1 - x3)/2}, {(y1 - y2)/2, (x2 - x1)/2}})[[All, id]]} dydx]
```

Cell prolongation matrix:

```
In[32]:= bmatrix[basis_, id_] :=
           Inverse[mmatrix[basis]].
             Integrate[1, {-1, 1}, {-1, -t1}] Transpose[(basis/.x → t1/.y → t2)].
                 (basis/.(If[id == 1, {x → (t1 - 1)/2, y → (t2 - 1)/2},
                       If[id == 2, {x → (t1 + 1)/2, y → (t2 - 1)/2},
                         If[id == 3, {x → (t1 - 1)/2, y → (t2 + 1)/2}, {x → -(t1 + 1)/2, y → -(t2 + 1)/2}]]])) dt2dt1

In[33]:= bmatrixinvm[basis_, id_, invm_] :=
           invm.
             Integrate[1, {-1, 1}, {-1, -t1}] Transpose[(basis/.x → t1/.y → t2)].
                 (basis/.(If[id == 1, {x → (t1 - 1)/2, y → (t2 - 1)/2},
                       If[id == 2, {x → (t1 + 1)/2, y → (t2 - 1)/2},
                         If[id == 3, {x → (t1 - 1)/2, y → (t2 + 1)/2}, {x → -(t1 + 1)/2, y → -(t2 + 1)/2}]]])) dt2dt1
```

END of formula

VITA

Yaqi Wang was born in 1973, in Inner Mongolia, China. He is the eldest son of Xiu Wang and Xiuhua Li. He grew up in Baoding, Hebei province. He enrolled in the Department of Engineering Physics of Tsinghua University in 1991 and obtained his B.S. degree in 1996. Upon graduation, he then worked at the Institute of Nuclear and New Energy Technology (INET) from 1996 to 2004 on research themes related with reactor operation. He married Qun Shi in September 2003 and moved to College Station, TX, USA in August 2004 to pursue a Ph.D. in the Department of Nuclear Engineering of Texas A&M University. He obtained his Master degree in December 2006 with a thesis topic related to *hp*-Mesh Adaptation for 1-D Multi-Group Neutron Diffusion Problems. In February 2008, he defended his Ph.D. research. He received his Ph.D. degree in May of 2009. Yaqi Wang's favorites include reading, chess, Xiangqi, etc. He can be reached at the following address: Department of Nuclear Engineering, Texas A&M University, 3133 TAMU, College Station, TX 77843-3133.