THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Signal, Image, Acoustique et Optimisation

Présentée et soutenue par :

M. STEVEN MURPHY

le mercredi 26 août 2015

Titre :

METHODS FOR SOLVING DISCONTINUOUS-GALERKIN FINITE ELEMENT EQUATIONS WITH APPLICATION TO NEUTRON TRANSPORT

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Centre Européen de Recherche et Formation Avancées en Calcul Scientifique (CERFACS)

Directeur(s) de Thèse :

M. IAIN S. DUFF

M. KENNETH ANDREW CLIFFE

Rapporteurs :

M. MATTHEW HUBBARD, THE UNIVERSITY OF NOTTINGHAM GB

Mme JENNIFER RYAN, UNIVERSITY OF EAST ANGLIA NORWICH

Membre(s) du jury :

M. MATTHEW HUBBARD, THE UNIVERSITY OF NOTTINGHAM GB, Président

M. IAIN S. DUFF, RUTHERFORD APPLETON LABORATORY OXFORD, Membre

Résumé

Cette thèse traite des méthodes d'éléments finis Galerkin discontinus d'ordre élevé pour la résolution d'équations aux dérivées partielles, avec un intérêt particulier pour l'équation de transport des neutrons. Nous nous intéressons tout d'abord à une méthode de pré-traitement de matrices creuses par blocs, qu'on retrouve dans les méthodes Galerkin discontinues, avant factorisation par un solveur multifrontal. Des expériences numériques conduites sur de grandes matrices bi- et tri-dimensionnelles montrent que cette méthode de pré-traitement permet une réduction significative du 'fill-in', par rapport aux méthodes n'exploitant pas la structure par blocs. Ensuite, nous proposons une méthode d'éléments finis Galerkin discontinus, employant des éléments d'ordre élevé en espace comme en angle, pour résoudre l'équation de transport des neutrons. Nous considérons des solveurs parallèles basés sur les sous-espaces de Krylov à la fois pour des problèmes 'source' et des problèmes aux valeur propre multiplicatif. Dans cet algorithme, l'erreur est décomposée par projection(s) afin d'équilibrer les contraintes numériques entre les parties spatiales et angulaires du domaine de calcul. Enfin, un algorithme HP-adaptatif est présenté ; les résultats obtenus démontrent une nette supériorité par rapport aux algorithmes h-adaptatifs, à la fois en terme de réduction de coût de calcul et d'amélioration de la précision. Les valeurs propres et effectivités sont présentées pour un panel de cas test industriels. Une estimation précise de l'erreur (avec effectivité de 1) est atteinte pour un ensemble de problèmes aux domaines inhomogènes et de formes irrégulières ainsi que des groupes d'énergie multiples. Nous montrons numériquement que l'algorithme HP-adaptatif atteint une convergence exponentielle par rapport au nombre de degrés de liberté de l'espace éléments finis.

Abstract

We consider high order discontinuous-Galerkin finite element methods for partial differential equations, with a focus on the neutron transport equation. We begin by examining a method for preprocessing block-sparse matrices, of the type that arise from discontinuous-Galerkin methods, prior to factorisation by a multifrontal solver. Numerical experiments on large two and three dimensional matrices show that this preprocessing method achieves a significant reduction in fill-in, when compared to methods that fail to exploit block structures. A discontinuous-Galerkin finite element method for the neutron transport equation is derived that employs high order finite elements in both space and angle. Parallel Krylov subspace based solvers are considered for both source problems and k_{eff}-eigenvalue problems. An *a-posteriori* error estimator is derived and implemented as part of an *h*-adaptive mesh refinement algorithm for neutron transport k_{eff}-eigenvalue problems. This algorithm employs a projection-based error splitting in order to balance the computational requirements between the spatial and angular parts of the computational domain. An *hp*-adaptive algorithm is presented and results are collected that demonstrate greatly improved efficiency compared to the *h*-adaptive algorithm, both in terms of reduced computational expense and enhanced accuracy. Computed eigenvalues and effectivities are presented for a variety of challenging industrial benchmarks. Accurate error estimation (with effectivities of 1) is demonstrated for a collection of problems with inhomogeneous, irregularly shaped spatial domains as well as multiple energy groups. Numerical results are presented showing that the *hp*-refinement algorithm can achieve exponential convergence with respect to the number of degrees of freedom in the finite element space.

Notation

The following notation is employed.

	Domain and Variables
v	Velocity
t	Time
S_2	Angular domain: the unit sphere
$\boldsymbol{\Omega} \in S_2$	Angular variable
$\varphi \in (0, \pi)$	Polar coordinate
$\theta \in [0, 2\pi)$	Azimuthal coordinate
$E \in \mathbb{R}^+$	Energy variable
$\mathcal{D}$	The spatial domain
Γ	The spatial domain boundary
Γ_-	The spatial domain inflow boundary
Γ_+	The spatial domain outflow boundary
$\mathbf{r} \in \mathcal{D}$	Spatial variable
x, y	Spatial coordinates
∇	Spatial gradient function
	Neutron transport
ψ	The angular flux of neutrons
k_{eff}	Reactivity eigenvalue
ψ^k	Reactivity eigenfunction
α	Time decay eigenvalue
ψ^α	Time decay eigenfunction
ϕ	The scalar flux
ζ	The space averaged flux

Σ_s	Scattering cross section
Σ_f	Fission cross section
Σ_a	Absorption cross section
Σ_t	Total cross section
ν	The average number of neutrons appearing from fission
χ	The probability of a neutron appearing from fission
Q	Additional neutron source
D	Diffusion coefficient
c	Scattering ratio
	Multigroup approximation
g	Energy group
G	Number of energy groups
ψ_g	Angular flux in energy group g
ϕ_g	Scalar flux in energy group g
ζ_g	Space averaged flux in energy group g
Q_g	Neutron source flux in energy group g
$\Sigma_{s,g\to g'}$	Neutron cross section for scattering from group g to group g'
$\Sigma_{f,g}$	Fission cross section in energy group g
$\Sigma_{a,g}$	Absorption cross section in energy group g
$\Sigma_{t,g}$	Total cross section in energy group g
ν_g	The average number of neutrons from fission in energy group g
χ_g	The probability of a neutron from fission in energy group g
	Angular Discretisations
S_N	Discrete ordinates method
$\boldsymbol{\Omega}_i$	ith discrete ordinate
w_i	Weight on the ith discrete ordinate
N_O	Number of discrete ordinates
P_N	Spherical harmonics method
$Y_{l,m}$	The $(l,m)^{\text{th}}$ spherical harmonic function
	Vectors and matrices
N	Number of matrix rows/columns
NE	Number of matrix entries
$\mathbf{A}$, $\mathbf{M}$	Matrices

$\mathbf{P}$, $\mathbf{Q}$	Permutation matrices
$\mathbf{L}$, $\mathbf{U}$	Triangular matrices
$\mathbf{D}$	Block diagonal matrix
$\mathbf{A}_{i,j}$	$(i,j)^{\text{th}}$ block in $\mathbf{A}$
$\mathbf{T}$	Transport matrix
$\mathbf{S}$	Scattering matrix
$\mathbf{F}$	Fission matrix
$\boldsymbol{\psi}$	Vector of coefficients for the primal solution
$\mathbf{q}$	Discretised source term
$\boldsymbol{z}$	Vector of coefficients for the dual solution
	Finite element method
$\hat{g}$	Dirichlet boundary function
f	Forcing function
p	Order of polynomial approximation in space
q	Order of polynomial approximation in angle
$\mathcal{T}_S$	Spatial mesh
$\mathcal{T}_A$	Angular mesh
$\mathcal{T}$	Space-angle mesh
κ_S	Spatial element
κ_A	Angular element
h	Finite element diameter
$\hat{S}$	Canonical square
$\hat{T}$	Canonical triangle
F_{κ_S}	Map from canonical element to κ_S
F_{κ_A}	Map from canonical element to κ_A
$F_{\mathcal{T}_S}$	Set of F_{κ_S}, where $\kappa_S \in \mathcal{T}_S$
$F_{\mathcal{T}_A}$	Set of $F_{\mathcal{T}_A}$, where $\kappa_A \in \mathcal{T}_A$
$S_S^p(\mathcal{T}_S, F_{\mathcal{T}_S})$	Spatial finite element space
$S_A^q(\mathcal{T}_A, F_{\mathcal{T}_A})$	Angular finite element space
N_S	Number of degrees of freedom in $S_S^p(\mathcal{T}_S, F_{\mathcal{T}_S})$
N_A	Number of degrees of freedom in $S_A^q(\mathcal{T}_A, F_{\mathcal{T}_A})$
$\mathcal{V}_h^{p,q}$	Space-angle finite element space
$\mathcal{V}_h^{p+1,q+1}$	Enriched finite element space

$\mathcal{Q}_p$	Polynomial tensor product space of order p
$\mathcal{P}_p$	Restricted polynomial tensor product space of order p
$\mathcal{Q}_q$	Polynomial tensor product space of order q
$\mathbf{n}_{\kappa_S}$	Unit outward normal to κ_S
v^+	Interior trace of v
v^-	Exterior trace of v
$\mathcal{H}$	Numerical flux function
ψ_h	Discontinuous-Galerkin finite element solution
v	Test function
v_h	Finite element test function
$\xi^{\hat{T}}_{\alpha,\beta}$	Dubiner basis function
$\xi^{\hat{S}}_{\alpha,\beta}$	Legendre polynomial tensor product basis function
$P_n^{\alpha,\beta}$	Jacobi polynomial
ξ_j	j^{th} basis function in the finite element space
	Linear algebra
$\mathbf{r}_m$	m^{th} residual vector
$\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$	m^{th} Krylov subspace
$\mathbf{q}_i$	i^{th} orthogonal basis vector for $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$
$\mathbf{V}_m$	Orthogonal basis for the m^{th} Krylov subspace
$\mathbf{H}_{i+1}$	Upper Hessenberg matrix
λ_1	Largest eigenvalue
λ_2	Second largest eigenvalue
	Graph theory
$\tilde{G} = (\tilde{V}, \tilde{E})$	A graph
$\tilde{V}$	The set of vertices
$\tilde{E}$	The set of edges
$\tilde{C}$	A cycle
$\tilde{S}$	A connected component
SCC_i	i^{th} strongly connected component
	Functionals
$(T-S)(\cdot,\cdot)$	Transport and scattering functional
$F(\cdot,\cdot)$	Fission functional
$C(\cdot,\cdot)$	Nonlinear functional for normalisation
$\mathcal{N}(\cdot,\cdot)$	Semilinear functional

$J(\cdot)$	Target functional
$\overline{J}(\cdot)$	Mean value linearisation of $J(\cdot)$
$\mathcal{M}(\cdot,\cdot)$	Mean value linearisation of $\mathcal{N}(\cdot,\cdot)$
$\mathcal{N}'[\cdot](\cdot,\cdot)$	Frèchet derivative of $\mathcal{N}(\cdot,\cdot)$
$J'[\cdot](\cdot)$	Frèchet derivative of $J(\cdot)$
$\mathrm{Res}(\cdot,\cdot)$	Residual functional
	Refinement algorithms
$\underline{\psi} = \left(k_{\mathrm{eff}}^{\psi}, \psi\right)$	Analytical primal eigenpair
$\underline{v} = \left(k_{\mathrm{eff}}^{v}, v\right)$	Test eigenpair
$\underline{z} = \left(k_{\mathrm{eff}}^{z}, z\right)$	Analytical dual eigenpair
$\underline{\psi}_h = \left(k_{\mathrm{eff}}^{\psi}, \psi\right)$	DG primal eigenpair
$\underline{v}_h = \left(k_{\mathrm{eff},h}^{v}, v_h\right)$	DG test function
$\underline{z}_h = \left(k_{\mathrm{eff},h}^{z}, z_h\right)$	DG dual eigenpair
$\underline{\hat{z}} = \left(k_{\mathrm{eff}}^{\hat{z}}, \hat{z}\right)$	Dual eigenpair from enriched finite element space
Π_S	L_2-projection onto $S_S^p(\mathcal{T}_S, F_{\mathcal{T}_S})$
Π_A	L_2-projection onto $S_A^q(\mathcal{T}_A, F_{\mathcal{T}_A})$
η_S	Spatial error indicator
η_A	Angular error indicator
I_{eff}	Effectivity
α^{FF}	Fixed fraction refine percentage
α_S^{FF}	Refine percentage for spatial elements
α_A^{FF}	Refine percentage for angular elements
$\mathbf{p}$	Polynomial degree vector
p_{κ_S}	Order of polynomial approximation on κ_S
$\mathcal{V}_{hp}^{\mathbf{p},q}$	*hp*-DG finite element space
$\mathcal{V}_{hp}^{\mathbf{p}+\mathbf{1},q+1}$	Enriched *hp*-DG finite element space
$\underline{\psi}_{hp} = \left(k_{\mathrm{eff},hp}^{\psi}, \psi_{hp}\right)$	Primal *hp*-eigenpair
$\underline{v}_{hp} = \left(k_{\mathrm{eff},hp}^{v}, v_{hp}\right)$	Test *hp*-eigenpair
$\underline{z}_{hp} = \left(k_{\mathrm{eff},hp}^{z}, z_{hp}\right)$	Dual *hp*-eigenpair
ϑ	*hp*-steering parameter
ϑ_{κ_S}	Analyticity estimate on κ_S
$\vartheta_{\kappa_S}^{\psi}$	Primal analyticity estimate on κ_S
$\vartheta_{\kappa_S}^{z}$	Dual analyticity estimate on κ_S

Contents

CHAPTER 1

Introduction

Determining the solution to partial differential equations (PDEs) is essential for the modelling of physical phenomena in a wide variety of engineering and scientific disciplines. For the vast majority of the PDEs arising from real world problems, however, irregularly shaped domains and non-smooth problem data frequently mean it is not possible to find analytical solutions. This has led to the development of a wide variety of methods for the computation of approximate solutions to PDEs, including finite difference methods, finite volume methods, and continuous- and discontinuous-Galerkin finite element methods, for example. In this thesis we consider the application of the latter of these schemes, namely the discontinuous-Galerkin finite element method, also known as the DG method or the DGFEM, to problems arising in the field of nuclear engineering.

When utilising numerical methods for the solution of PDEs it is necessary to balance the need for accurate approximations with the available computational resources, see [1]. For DG methods, as is the case with other finite element methods, greater accuracy can be achieved through the introduction of more elements into the mesh (h-refinement), as well as the utilisation of higher order polynomials (p-refinement). In particular local mesh refinement and local polynomial enrichment may be combined (hp-refinement) in order to focus resources in those regions in the computational domain that significantly contribute to the total error in the computed finite element solution, when the error is measured with respect to a suitable norm or target functional of interest.

The main computational cost stemming from the application of DG methods, as indeed for other discretisation methods, to the numerical approximation of PDEs, lies in computing the solution of the underlying system of equations. In order to render such

systems tractable, it is necessary to design parallel linear solvers that are efficient in terms of both the number of floating point operations needed, as well as the amount of memory required. These solvers can be classified as direct linear solvers, which are usually based on computing a factorisation of the matrix, and iterative solvers, which include Krylov subspace based methods and stationary iterative schemes. Direct solvers benefit from considerable robustness and accuracy, however the increasing memory requirements as the problem size grows, particularly for higher dimensional equations, can render many problems intractable. Iterative solvers on the other hand provide the user with greater control on the memory requirements, at the cost of reduced stability, which necessitates the design of sophisticated, and frequently problem specific preconditioners.

For the safe design and operation of nuclear power stations it is necessary to find the solution to the integro-differential equation that governs the distribution of neutron flux inside the nuclear reactor: the neutron transport equation. The computation of accurate numerical approximations to the full seven dimensional neutron transport equation is not feasible in realistic geometries. Indeed, computing the key quantity of interest, the critical eigenvalue, for the reduced four and five dimensional versions of the neutron transport equation that nuclear engineers use to model modern nuclear reactors, presents a considerable computational challenge. In this thesis we shall develop an *hp*-adaptive algorithm for neutron transport criticality problems, that not only targets computational resources in those areas in the spatial domain that contribute the most error to the critical eigenvalue, but also balances the computational resources between the spatial and angular parts of the DG discretisation. Furthermore, we demonstrate the efficacy of our method by computing the solution to a series of industrially relevant benchmark problems. We present semilog convergence plots for these which show that the error appears to reduce exponentially quickly.

We begin by considering how the structure of the matrices arising from DG methods can be utilised to design efficient algorithms for their solution. We present a method for preprocessing matrices, such as DG matrices, that have the symmetric block sparse structure with dense blocks, in order to reduce the memory required, and number of operations, to compute a factorisation using a sparse direct solver. We then proceed to consider a high-order DG method for the neutron transport equation which exploits a DG method for both the spatial and the angular variables. We present an efficient Krylov subspace based solver for neutron transport criticality problems that utilises the

aforementioned preprocessing method as part of an efficient preconditioner. We then derive a computable *a posteriori* error representation formula for this scheme, which can be used to design adaptive algorithms for the neutron transport k_{eff}-eigenvalue problem. Finally, we implement an *hp*-version DG method for neutron transport criticality problems and demonstrate the rapid convergence that can be obtained by such a method with respect to the number of degrees of freedom required, for a series of challenging test problems.

We continue the present chapter with a brief discussion of the development of DG methods, followed by a description of the neutron transport equation. We then give an overview of the key methods currently employed in the literature for its numerical solution. The chapter is concluded with an outline of the remaining chapters in the thesis.

1.1 Development of DG methods

In this section we provide a brief review of the development of DG methods. For a more detailed history see the 1999 review paper by Cockburn, Karniadakis and Shu [2], or the 2002 article by Arnold, Brezzi, Cockburn and Marini [3] which presents a unified analysis of DG methods for elliptic problems. The important points from these articles are discussed in the following.

In PDEs where convection plays a significant role, it is known that the analytical solution can contain steep gradients and, for nonlinear problems, discontinuities. Moreover, close to these features, the solution can contain rich and complicated structures that need to be resolved in order to compute an accurate approximate solution. Conforming finite element methods are known to suffer from unphysical oscillations in the regions close to these features, which severely damage the quality of the computed numerical approximation, see [4]. These oscillations are a consequence of the fact that continuous-Galerkin (CG) methods attempt to represent discontinuous and rapidly varying solutions in a continuous finite element space which are not, in general, conservative across element boundaries. In order to negate this problem, the streamlined upwind Petrov Galerkin (SUPG) method has been developed [5], which includes an artificial diffusion term in the streamwise direction in order to damp any such oscillations. The development of DG methods however, which are locally conservative across

element boundaries, has enabled the computation of finite element approximations to this class of problems that are more robust with respect to such spurious oscillations, without the need to introduce artificial stabilisation terms. Though these oscillations can still occur, the permissibility of discontinuities across element boundaries allow for them to be damped by natural numerical dissipation.

DG methods were initially stimulated by the observation that the weak enforcement of Dirichlet boundary conditions employed in conforming finite element methods can be extended to enforce inter-element connectivity between elements. As such, it is possible to design discontinuous finite element spaces that can represent a high order approximation to the solution within finite elements, whilst permitting discontinuities across element boundaries in order to facilitate the representation of sharp features in the solution. Furthermore, such a framework results in a high degree of locality, enabling efficient parallel implementations. Indeed, the lack of a continuity restriction also means that DG methods can easily support local variations in the order of polynomial approximation, as well as supporting irregular meshes and meshes with hanging nodes. This enables the implementation of *hp*-adaptive algorithms and facilitates the application of DG methods to problems with complicated and irregular geometries.

The first implementation of a DG method was presented by Reed and Hill in 1973 [6], who developed a DG method for the spatial approximation of the neutron transport equation. In 1974 LeSaint and Raviart [7] published an *a priori* analysis of their method that demonstrated convergence of order $\mathcal{O}\left(h^p\right)$ in the L_2-norm for general triangulations, where h is a typical element diameter and p is the order of polynomial approximation. In 1983 Johnson and Pitkäranta [8], improved this to $\mathcal{O}\left(h^{p+\frac{1}{2}}\right)$ and this rate was shown to be optimal for general meshes by Peterson in 1991 [9]. In 1988 Richter [10], proved a rate of $\mathcal{O}\left(h^{p+1}\right)$ for some structured non-Cartesian grids. These results, however, are all for problems where the underlying solution is smooth. In [11], Houston, Schwab and Süli proved exponential convergence for the case when the underlying solution is piecewise analytic.

The *a posteriori* analysis for the DG method for linear hyperbolic problems was first presented by Stroubolis and Oden in [12]. Later, in [13], Bey and Oden produced the first *a priori* and *a posteriori* error estimates for the *hp*-version DG method, which reduces to the result of Johnson and Pitkäranta [8], when $h \to 0$ for fixed p. In 2002 Houston, Schwab and Süli [14] generalised these results to a framework containing second order

advection diffusion PDEs with non-negative characteristic form.

In 2003, Cockburn, Luskin, Shu and Süli [15], developed a method for postprocessing DG approximations to time dependent hyperbolic equations that increases the rate of convergence from $\mathcal{O}\left(h^{p+\frac{1}{2}}\right)$ to $\mathcal{O}\left(h^{2p+1}\right)$. In the same year, Ryan and Shu [16], further developed the method to apply to the entire domain and not just away from the boundaries, discontinuities or interfaces of different mesh sizes. In 2005, Ryan, Shu, and Atkins [17], extended this technique to equations in two spatial dimensions with variable and discontinuous coefficients. Furthermore, they extended the method to include superconvergence of the derivatives and to apply to multi-domains with different meshes.

Though originally presented in the context of neutron transport theory, the DG method of Reed and Hill was no more than a discretisation of the linear advection-reaction equation. Since then, DG methods have been successfully applied to a broad class of PDEs including hyperbolic, elliptic and parabolic equations, as well as nonlinear problems. Examples of application areas include viscoelastic flows by Fortin and Fortin in [18], magneto-hydrodynamics by Warburton and Karniadakis in [19] and to compressible viscous flows by Lomtev, Quillen and Karniadakis in [20].

1.2 The neutron transport equation

In the next three sections we introduce the neutron transport equation and describe the physical meaning of each of the terms that comprise it. We proceed to present an overview of the key mathematical problems arising in the field of neutronics and describe the key discretisation methods that are used in the literature to find approximate solutions to these. For a more comprehensive overview of the field we refer the reader to the books, [21] and [22]. For a more detailed derivation of the neutron transport equation we refer the reader to [23], and for consideration of the mathematical topics in the neutron transport theory we refer the reader to [24].

It is important to have an accurate model of the distribution and energy of neutrons inside nuclear reactors for the safe design and operation of modern nuclear power stations. This is because the rate of nuclear fission is determined by the distribution and energy levels of neutrons in and around the nuclear fuel. One of the most important equations for the modelling of this distribution is known as the neutron transport

equation, the Boltzmann equation or Boltzmann transport equation. Producing accurate numerical solutions of this equation is an active area of contemporary research. The neutron transport equation is a seven dimensional integro-differential equation in space, angle, time and energy which balances the number of neutrons being introduced into the domain with those leaving it. Before stating the equation we first introduce some modelling concepts.

When modelling a population of neutrons we need to have a way to represent the various interactions that might occur, as well as a way to represent the motion of the neutrons between those interactions. For sub-atomic particles such as neutrons this could potentially require an involved quantum mechanical model incorporating wave particle duality. However, neutrons moving at speeds typical for a nuclear reactor are best modelled as point particles moving ballistically between interactions. This is because, at the energies typical in a nuclear reactor, neutrons are moving too quickly for the neutron wavelength to be important. Also they rarely obtain speeds greater than 0.17% of the speed of light, therefore relativistic effects are not significant. As there are a large number of neutrons at motion throughout the system, we may model them as a continuum. We introduce the angular flux of neutrons at the point $\mathbf{r}$ in the direction $\boldsymbol{\Omega}$ with energy E at time t, denoted by

$$\psi(\mathbf{r}, \boldsymbol{\Omega}, E, t), \tag{1.2.1}$$

this is the key quantity of interest that is modelled by the neutron transport equation. Since ψ is a function of direction at each point in three dimensional space, it is necessary to define the two dimensional angular domain in which this set of directions exist. For that we introduce the concept of a solid angle. If we consider a point $\mathbf{r}$ in space, then the surface of any object in our domain may be considered to take up a certain proportion of the total view when looking out from that point. We call this quantity the *solid angle* subtended by that surface from $\mathbf{r}$. More precisely, we may describe the solid angle as the area taken up by the projection of that surface onto the unit sphere centred at $\mathbf{r}$. As such the total solid angle measured from any point is equal to $4\pi\, sr$, which is the surface area of the unit sphere. The S.I. unit of solid angle is the steradian, denoted sr, and like its two dimensional analogue, the radian, it is dimensionless. In the neutron transport equation, solid angles are used to represent the continuum of directions in which a flux of neutrons may be traveling from each point in the spatial domain.

Now that we have stated a suitable model for the motion of the neutrons between inter-

actions, we consider the types of interaction that can occur. The model that we are considering assumes that there are a very large number of stationary, inert atomic nuclei in our medium relative to the number of neutrons and therefore any neutron-neutron and nuclide-nuclide interactions may be ignored. All that remains are the neutron-nuclide interactions of which there are three types: scattering, capture, and fission. In order to represent these interactions we introduce their macroscopic neutron cross sections, which express the fractional probability of each reaction occurring per path length travelled, thereby providing a continuum characterisation of the point collisions occurring in the domain.

The first type of interaction that we consider is scattering, which happens when a neutron collides with a nuclide and is not absorbed but is instead deflected away. The differential scattering cross section represents the fractional probability that a neutron at $\mathbf{r}$ with energy E' traveling in direction $\boldsymbol{\Omega}'$ will be scattered to direction $\boldsymbol{\Omega}$ with energy E at time t; it is written as

$$\Sigma_s(\mathbf{r}, \boldsymbol{\Omega}', \boldsymbol{\Omega}, E', E, t). \tag{1.2.2}$$

The second type of interaction is absorption, which occurs when a neutron collides with a nuclide and is absorbed into it, increasing its nuclear mass number. The absorption cross section is given by

$$\Sigma_a(\mathbf{r}, E, t). \tag{1.2.3}$$

The third fundamental interaction is fission, which is the process that drives the nuclear reaction. Fission occurs when a large unstable nuclide splits into two or more smaller nuclei, thereby releasing additional neutrons into the system, as well as a very large amount of energy. The fission cross section quantifies the likelihood of a fission occurring at each given point in the domain; it is given by

$$\Sigma_f(\mathbf{r}, E, t). \tag{1.2.4}$$

As each fission can release many neutrons at various energies, two more functions are required to fully describe the effect of fission on the angular flux. These are the average number of neutrons produced per fission, which is denoted by $\nu(E)$, and the probability of a neutron being introduced with energy E, which is written as $\chi(E)$.

Before stating the neutron transport equation we introduce one further cross section which incorporates all neutrons lost from the system from absorption and scattering.

This is called the total cross section and is written as

$$\Sigma_t(\mathbf{r}, E, t). \tag{1.2.5}$$

Now that we have defined all the necessary terms we may state the full neutron transport equation which equates all neutrons lost to the angular flux on the left hand side with all neutrons gained on the right-hand side: find $\psi(\mathbf{r}, \boldsymbol{\Omega}', E', t)$ such that

$$\begin{aligned}\left(\frac{1}{v}\frac{\partial}{\partial t} + \boldsymbol{\Omega}\cdot\nabla + \Sigma_t(\mathbf{r}, E, t)\right)\psi(\mathbf{r}, \boldsymbol{\Omega}, E, t) \\ = \int_{\mathbb{R}^+}\int_{S_2}\Sigma_s(\mathbf{r}, \boldsymbol{\Omega}', \boldsymbol{\Omega}, E', E, t)\psi(\mathbf{r}, \boldsymbol{\Omega}', E', t)d\boldsymbol{\Omega}'dE' \\ + \frac{\chi(E)}{4\pi}\int_{\mathbb{R}^+}\int_{S_2}\nu(E')\Sigma_f(\mathbf{r}, E', t)\psi(\mathbf{r}, \boldsymbol{\Omega}', E', t)d\boldsymbol{\Omega}'dE' \\ + Q(\mathbf{r}, \boldsymbol{\Omega}, E, t).\end{aligned} \tag{1.2.6}$$

Here v is the neutron speed and S_2 is the surface of the unit sphere. The final term $Q(\mathbf{r}, \boldsymbol{\Omega}, E, t)$ is an artificial forcing function which accounts for any additional neutrons introduced into the system from other sources. For criticality problems, which comprise most of the problems considered in this thesis, this term is not present. When a known solution angular flux is required for testing purposes, this may be chosen so that the neutron transport equation will yield that specific solution. This is the strategy adopted in Chapter 3 when an analytical value for the solution is needed in order to investigate the order of convergence of the proposed DG method.

For all of the problems considered in this thesis, the nuclear cross sections are taken to be time invariant, therefore we may omit t from each of the cross sections in equation (1.2.6) to obtain

$$\begin{aligned}\left(\frac{1}{v}\frac{\partial}{\partial t} + \boldsymbol{\Omega}\cdot\nabla + \Sigma_t(\mathbf{r}, E)\right)\psi(\mathbf{r}, \boldsymbol{\Omega}, E, t) \\ = \int_{\mathbb{R}^+}\int_{S_2}\Sigma_s(\mathbf{r}, \boldsymbol{\Omega}', \boldsymbol{\Omega}, E', E)\psi(\mathbf{r}, \boldsymbol{\Omega}', E', t)d\boldsymbol{\Omega}'dE' \\ + \frac{\chi(E)}{4\pi}\int_{\mathbb{R}^+}\int_{S_2}\nu(E')\Sigma_f(\mathbf{r}, E')\psi(\mathbf{r}, \boldsymbol{\Omega}', E', t)d\boldsymbol{\Omega}'dE' \\ + Q(\mathbf{r}, \boldsymbol{\Omega}, E, t).\end{aligned} \tag{1.2.7}$$

1.3 Criticality problems

One key property of interest for engineers when considering a system of fissile nuclides is the criticality, that is whether the system is sub-critical, critical or super-critical. Intuitively this property corresponds to the behaviour of the system as $t \to \infty$ with the neutron population going to zero if the system is sub-critical, remaining constant if it is critical and increasing if the system is super-critical.

There are two versions of the criticality problem which both take the form of an eigenvalue problem. These are the time decay or α-eigenvalue problem and the reactivity or k_{eff}-eigenvalue problem. To obtain the α-eigenvalue problem we consider solutions of equation (1.2.7) of the form

$$\psi^{\alpha}(\mathbf{r}, \boldsymbol{\Omega}, E)e^{\alpha t}, \tag{1.3.1}$$

where $\psi^{\alpha}(\mathbf{r}, \boldsymbol{\Omega}, E)$ is the part of the angular flux which is time independent. Substituting this into equation (1.2.7) yields the following eigenvalue problem for the eigenpair $(\alpha, \psi^{\alpha}(\mathbf{r}, \boldsymbol{\Omega}, E))$

$$\begin{aligned} \alpha\psi^{\alpha}(\mathbf{r}, \boldsymbol{\Omega}, E) = &-v\boldsymbol{\Omega} \cdot \nabla\psi^{\alpha}(\mathbf{r}, \boldsymbol{\Omega}, E) - v\Sigma_t(\mathbf{r}, E)\psi^{\alpha}(\mathbf{r}, \boldsymbol{\Omega}, E) \\ &+ \int_{\mathbb{R}^+}\int_{S_2} v'\Sigma_s(\mathbf{r}, \boldsymbol{\Omega}', \boldsymbol{\Omega}, E', E)\psi^{\alpha}(\mathbf{r}, \boldsymbol{\Omega}', E')d\boldsymbol{\Omega}'dE' \\ &+ \frac{\chi(E)}{4\pi}\int_{\mathbb{R}^+}\int_{S_2} v'\nu(E')\Sigma_f(\mathbf{r}, E')\psi^{\alpha}(\mathbf{r}, \boldsymbol{\Omega}', E')d\boldsymbol{\Omega}'dE', \end{aligned} \tag{1.3.2}$$

where the source term has been omitted.

Given the possible set of solution eigenpairs $(\alpha_i, \psi^{\alpha_i}(\mathbf{r}, \boldsymbol{\Omega}, E))$, $i = 0, \cdots$, satisfying (1.3.2), we order them so that α_0 will be the eigenvalue with the largest real part. Writing the full time dependent angular flux as an expansion in terms of these eigenpairs, we note that as t increases the angular flux will be proportional to $e^{t\alpha_0}\psi^{\alpha_0}(\mathbf{r}, \boldsymbol{\Omega}, E)$. Thus we see that determining the criticality of the system is equivalent to determining the sign of α_0. For a sub-critical problem we have $\alpha_0 < 0$, for a super-critical problem we have $\alpha_0 > 0$, and for a critical problem we have $\alpha_0 = 0$.

The second formulation of the criticality problem is known as the effective multiplication factor, reactivity or k_{eff}-eigenvalue problem. Whereas the α-eigenvalue calculates whether a system is critical by considering what happens when $t \gg 0$, the k_{eff}-eigenvalue problem does so by considering how many neutrons would have to be released by every fission for the system to be critical. The eigenpairs $\left(k_{\text{eff}}, \psi^{k}(\mathbf{r}, \boldsymbol{\Omega}, E)\right)$

are computed by solving the eigensystem obtained by rewriting equation (1.2.7) with $\psi^k(\mathbf{r}, \boldsymbol{\Omega}, E)$ in place of $\psi(\mathbf{r}, \boldsymbol{\Omega}', E', t)$ and

$$\frac{\nu(E)}{k_{\text{eff}}} \tag{1.3.3}$$

in place of $\nu(E)$; i.e., we consider

$$\begin{aligned}(\boldsymbol{\Omega} \cdot \nabla + \Sigma_t(\mathbf{r}, E))\, \psi^k(\mathbf{r}, \boldsymbol{\Omega}, E) \\ = \int_{\mathbb{R}^+} \int_{S_2} \Sigma_s(\mathbf{r}, \boldsymbol{\Omega}', \boldsymbol{\Omega}, E', E) \psi^k(\mathbf{r}, \boldsymbol{\Omega}', E') d\boldsymbol{\Omega}' dE' \\ + \frac{1}{k_{\text{eff}}} \frac{\chi(E)}{4\pi} \int_{\mathbb{R}^+} \int_{S_2} \nu(E') \Sigma_f(\mathbf{r}, E') \psi^k(\mathbf{r}, \boldsymbol{\Omega}', E') d\boldsymbol{\Omega}' dE'. \end{aligned} \tag{1.3.4}$$

The existence in general of the eigenvalue k_{eff} and its associated eigenfunction $\psi^k(\mathbf{r}, \boldsymbol{\Omega}, E)$ are assumed for physical reasons. If there are fissions taking place in the system then we assume that by varying the number of neutrons released in each fission it is possible to force the system into a state of balance between the number of neutrons gained and the number of neutrons lost. A more detailed discussion of this eigenproblem is given in [21] including the existence of a discrete dominant k_{eff}-eigenvalue under the multigroup discretisation of the energy variable (see Section 1.5.1).

Whereas criticality was inferred from the α eigenvalue problem by checking the sign of α_0, criticality is inferred from the k_{eff} problem by comparing the dominant eigenvalue with unity. If k_{eff} is greater than one then fewer neutrons per fission would need to be released to create a critical system, therefore the system is super-critical. If k_{eff} is less than one then more neutrons would need to be released per fission, therefore the system is sub-critical. If k_{eff} is exactly one then the system is critical.

By comparing equations (1.3.2) and (1.3.4) we see that when the system is critical, with $\alpha_0 = 0$ and $k_{\text{eff}} = 1$, the eigenfunctions $\psi^\alpha(\mathbf{r}, \boldsymbol{\Omega}, E)$ and $\psi^k(\mathbf{r}, \boldsymbol{\Omega}, E)$ will be identical. When the system is not critical, however, they will in general be different, therefore when criticality needs to be verified, there is a choice to be made between solving these two problems. As stated in [21] the more common choice is to compute the k_{eff}-eigenvalue. This is because for sub-critical systems the α_0 problem can be more difficult to handle numerically, as well as the fact that for non critical systems the addition of $\frac{\alpha_0}{v}$ to the absorption term will change the spectrum. All of the criticality problems considered in this thesis are defined in terms of k_{eff}-eigenvalue problems.

1.4 Reducing the dimensionality

The full neutron transport equation is an integro-differential equation of seven dimensions (three in space, two in angle, one in energy and one in time) and as a consequence the number of degrees of freedom that would be required for an accurate discretisation of all the dimensions can grow extremely quickly. In order to mitigate this there are several assumptions that we may make in order to reduce the dimensionality of the mathematical problem that needs to be solved whilst maintaining a faithful representation of the physical system being modelled.

The first two assumptions that we may make are to assume that all nuclear cross sections are time independent and that the solution angular flux goes to zero as time goes to infinity, i.e.,

$$\lim_{t\to\infty} \psi(\mathbf{r}, \boldsymbol{\Omega}, E, t) = 0. \tag{1.4.1}$$

This enables us to re-write equation (1.2.7) as the steady state neutron transport source problem

$$\begin{aligned}(\boldsymbol{\Omega}\cdot\nabla + \Sigma_t(\mathbf{r}, E))\,\psi(\mathbf{r}, \boldsymbol{\Omega}, E) &\\ = \int_{\mathbb{R}^+}\int_{S_2} \Sigma_s(\mathbf{r}, \boldsymbol{\Omega}', \boldsymbol{\Omega}, E', E)\psi(\mathbf{r}, \boldsymbol{\Omega}', E')d\boldsymbol{\Omega}'dE' &\\ + \frac{\chi(E)}{4\pi}\int_{\mathbb{R}^+}\int_{S_2} \nu(E')\Sigma_f(\mathbf{r}, E')\psi(\mathbf{r}, \boldsymbol{\Omega}', E')d\boldsymbol{\Omega}'dE' &\\ + Q(\mathbf{r}, \boldsymbol{\Omega}, E), &\end{aligned} \tag{1.4.2}$$

where the time independent angular flux is given by the integral over time of the time dependent angular flux

$$\psi(\mathbf{r}, \boldsymbol{\Omega}, E) = \int_{\mathbb{R}^+} \psi(\mathbf{r}, \boldsymbol{\Omega}, E, t)dt. \tag{1.4.3}$$

The steady state source term $Q(\mathbf{r}, \boldsymbol{\Omega}, E)$ now incorporates any non-zero initial condition data.

Even the steady state neutron transport equation retains a high level of dimensionality, so it is frequently necessary to make further assumptions about the remaining dimensions in order to render it computationally tractable. The five dimensions in space and angle still remain, as well as the energy variable. In Section 1.5.1 we discuss the monoenergetic and the multigroup approximations for the energy spectrum. Furthermore, geometrical approximation may be employed to reduce the dimension of

the space-angle computational domain. We briefly describe the two dimensional case, which we will use for our calculations, and the 'slab' geometry which is a popular one dimensional approximation that we will refer to later in the thesis.

The two dimensional (2D) geometry, often referred to as pseudo-3D, is obtained by considering the spatial variable in Cartesian coordinates $\mathbf{r} = (x, y, z)^\top$ and requiring that the angular flux has no dependence on the z variable, which extends to infinity in either direction. As nuclear reactors are frequently designed with a high degree of symmetry in the z-direction, with geometries comprising long, narrow control rods and fuel rods, this geometry provides a good approximation for many existing nuclear reactors. All of the test problems in this thesis consider a two dimensional spatial geometry; this reduction reduces the neutron transport problem to a five dimensional problem with two spatial dimensions, two angular dimensions, and one dimension in energy.

In the early days of neutron transport theory even problems with two spatial dimensions were too large from a computational viewpoint. Therefore, further reduction to a single spatial dimension was frequently employed for computations. One way that this was achieved was by prescribing that the solution has no dependence on either the x or the y Cartesian variables in the $\mathbb{R}^2$ plane. Under this assumption the spatial domain is specified by a thickness in the z direction which is called the slab thickness. This symmetry in the spatial domain also leads to a reduction in the number of coordinates required to describe the angular variable, as the invariance in the (x, y) plane renders the polar coordinate superfluous. The result is a reduced problem with one spatial variable z and one angular variable $\mu = \cos(\varphi)$, where $\mu \in [-1, 1]$ and φ is the angle between the advective direction and the normal to the (x, y)-plane.

1.5 Discretisation methods

In this section we discuss a variety of deterministic methods that have been employed to discretise the neutron transport equation. Monte Carlo methods, which model a set of neutron histories using a random number generator to determine outcomes of collisions, are also widely used within the neutron transport field. In this thesis, however we shall focus on deterministic methods; for a detailed description of the various Monte Carlo methods that have been exploited to model the transport of neutrons, we refer the reader to [25].

We begin our consideration of deterministic methods for neutron transport with the multigroup approximation for energy, see [26] for a more comprehensive review of these. This is the most commonly used treatment for the energy variable in the literature and will be the energy approximation employed in all of our computations. We shall then proceed to discuss various schemes that have been used to discretise the spatial and angular variables.

1.5.1 Multigroup approximation for energy

There are two main approaches employed for the discretisation of the energy variable that are commonly used in neutron transport theory: the monoenergetic approximation and the multigroup approximation. The monoenergetic neutron transport may be derived directly from the physical system by assuming that all scattering is elastic (see [23] for an example of this derivation) or alternatively it can be arrived at as a special case of the multigroup approximation. As both of these approximations are used in this thesis, we first describe the multigroup approximation and leave the monoenergetic approximation to be deduced as the case where only a single energy group is considered.

The multigroup approximation provides a lightweight discretisation of the energy variable by dividing the energy spectrum into G discrete energy groups. Once divided, the neutron transport equation may be treated as a system of G transport problems of the remaining independent variables, coupled by a scattering term. To obtain the multigroup expansion, the full energy spectrum is first restricted to the range of interest, $[E_0, E_G] \subset \mathbb{R}^+$, which is then partitioned into G different intervals. Each energy group g, $g = 1, \cdots, G$, consists of the values of the energy variable in the interval $[E_{g-1}, E_g]$. By convention group 1 is the highest energy group. The multigroup approximation assumes that for each energy group, the angular flux and the source term may be separated into the product of its energy dependent and independent parts. For the time independent case and energy group g, we have that

$$\psi(\mathbf{r}, \boldsymbol{\Omega}, E) = f(E)\psi_g(\mathbf{r}, \boldsymbol{\Omega}), \tag{1.5.1}$$

$$Q(\mathbf{r}, \boldsymbol{\Omega}, E) = h(E)Q_g(\mathbf{r}, \boldsymbol{\Omega}), \tag{1.5.2}$$

where $f(E)$ and $h(E)$ are piecewise smooth functions on each interval $[E_{g-1}, E_g]$, for

each $g \in \{1, 2, \cdots, G\}$, that are normalised so that

$$\int_{E_{g-1}}^{E_g} f(E)dE \equiv 1 \quad \text{and} \quad \int_{E_{g-1}}^{E_g} h(E)dE \equiv 1. \tag{1.5.3}$$

To obtain the steady state multigroup equations we consider equation (1.4.2) and replace the integrals with respect to E' with the sum of the integrals over each energy sub-interval $[E_{g-1}, E_g]$, $g = 1, \cdots, G$. Furthermore, we integrate the resulting integro-differential equation with respect to E to obtain, for the interval $[E_{g-1}, E_g]$

$$\begin{aligned}
&\int_{E_{g-1}}^{E_g} \boldsymbol{\Omega} \cdot \nabla f(E)\psi_g(\mathbf{r}, \boldsymbol{\Omega})\, dE + \int_{E_{g-1}}^{E_g} \Sigma_t(\mathbf{r}, E) f(E)\psi_g(\mathbf{r}, \boldsymbol{\Omega})\, dE \\
&\qquad = \sum_{g'=1}^{G} \int_{E_{g'-1}}^{E_{g'}} \int_{E_{g-1}}^{E_g} \int_{S_2} \Sigma_s(\mathbf{r}, \boldsymbol{\Omega}', \boldsymbol{\Omega}, E', E)\psi_{g'}(\mathbf{r}, \boldsymbol{\Omega}') f(E')\, d\boldsymbol{\Omega}'\, dE\, dE' \\
&\qquad\quad + \sum_{g'=1}^{G} \int_{E_{g'-1}}^{E_{g'}} \int_{E_{g-1}}^{E_g} \frac{\chi(E)}{4\pi} \int_{S_2} \nu(E')\Sigma_f(\mathbf{r}, E')\psi_{g'}(\mathbf{r}, \boldsymbol{\Omega}') f(E')\, d\boldsymbol{\Omega}'\, dE\, dE' \\
&\qquad\quad + \int_{E_{g-1}}^{E_g} Q_g(\mathbf{r}, \boldsymbol{\Omega}) h(E)\, dE.
\end{aligned} \tag{1.5.4}$$

Now we utilise the normalisation properties of $f(E)$ and $h(E)$ to define the following group parameters

$$\Sigma_{t,g}(\mathbf{r}) = \int_{E_{g-1}}^{E_g} \Sigma_t(\mathbf{r}, E) f(E)\, dE, \tag{1.5.5}$$

$$\Sigma_{s,g'\to g}(\mathbf{r}, \boldsymbol{\Omega}', \boldsymbol{\Omega}) = \int_{E_{g'-1}}^{E_{g'}} \int_{E_{g-1}}^{E_g} \Sigma_s(\mathbf{r}, \boldsymbol{\Omega}', \boldsymbol{\Omega}, E', E) f(E')\, dE\, dE', \tag{1.5.6}$$

$$\chi_g = \int_{E_{g-1}}^{E_g} \chi(E)\, dE, \tag{1.5.7}$$

$$\nu_g \Sigma_{f,g}(\mathbf{r}) = \int_{E_{g-1}}^{E_g} \nu(E)\Sigma_f(\mathbf{r}, E) f(E)\, dE. \tag{1.5.8}$$

With this notation, we deduce the multigroup equation for energy group g, $1 \le g \le G$

$$\begin{aligned}
&\boldsymbol{\Omega} \cdot \nabla \psi_g(\mathbf{r}, \boldsymbol{\Omega}) + \Sigma_{t,g}(\mathbf{r})\psi_g(\mathbf{r}, \boldsymbol{\Omega}) \\
&\qquad = \sum_{g'=1}^{G} \int_{S_2} \Sigma_{s,g'\to g}(\mathbf{r}, \boldsymbol{\Omega}', \boldsymbol{\Omega})\psi_{g'}(\mathbf{r}, \boldsymbol{\Omega}')\, d\boldsymbol{\Omega}' \\
&\qquad\quad + \sum_{g'=1}^{G} \frac{\chi_g}{4\pi} \int_{S_2} \nu_{g'}\Sigma_{f,g'}(\mathbf{r})\psi_{g'}(\mathbf{r}, \boldsymbol{\Omega}')\, d\boldsymbol{\Omega}' + Q_g(\mathbf{r}, \boldsymbol{\Omega}).
\end{aligned} \tag{1.5.9}$$

In the case when $G = 1$, so that the interval $[E_0, E_1]$ contains all values of interest in the energy spectrum, we deduce the monoenergetic approximation of the neutron transport equation.

1.5.2 Angular approximation

Over the years several techniques have been developed for the discretisation of the angular variable; indeed, the development of angular discretisations remains the subject of current research, see [27], [28] and [29]. In this section, we discuss the treatment of the angular domain by the three most common methods: the discrete ordinates method, the spherical harmonics method and the reduction to a diffusion approximation. Other methods have been proposed, including the representation of the angular dependence by spherical wavelets (see Cho et al. in [28] and Buchan et al. in [30]) and the use of finite element discretisations in angle (see Martin et al. in [31], Briggs et al. in [32], Becker et al. in [33] and Kanschat in [34]).

Discrete ordinates methods

One of the simplest and most popular methods for discretising the angular variable is the discrete ordinates method (DOM), also known as the S_N method. It is a collocation method which approximates the solution by representing it at a finite number of points in the angular domain. This set of points, together with associated weights, on the surface of the sphere form a quadrature for the integral over the angular domain. If these points and weights are given by $\boldsymbol{\Omega}_i$ and w_i, $i = 1, \cdots, N_O$, respectively, then the integral approximation of a function $f(\boldsymbol{\Omega})$ over the unit sphere is given by

$$\int_{S_2} f(\boldsymbol{\Omega})\, d\boldsymbol{\Omega} \approx \sum_{i=1}^{N_O} w_i f(\boldsymbol{\Omega}_i). \tag{1.5.10}$$

Approximating the integrals in the neutron transport equation in this manner uncouples the streaming and absorption terms. This reduces the solution of the full space-angle equation to a series of spatial linear advection problems, which can be discretised and solved in parallel either as part of a stationary iterative scheme or as part of a preconditioner for a Krylov subspace method. Discrete ordinates methods were first proposed by Carlson in 1955 [35] and developed in greater detail by Chandrasekhar in [36]. Further early development of discrete ordinates methods was spurred by healthy competition between the team responsible for S_N methods and another team at Los Alamos who were working on Monte Carlo methods [37].

The main advantages of the DOM are the ease with which it can be implemented, as well as the relative ease of imposing boundary conditions. Also the high level of con-

currency makes the DOM very well suited to parallel architectures. For problems of neutron absorption the DOM reduces the neutron transport equation to a set of hyperbolic problems in the spatial domain. This reduction to linear hyperbolic problems, or coupled linear hyperbolic problems when there is scattering, also enables highly memory-efficient sweeping techniques for the solution of the spatial part of the problem. However, the DOM does have some significant disadvantages, not least of which is the occurrence of spurious oscillations in the angular solution, known as ray effects [38], which can seriously affect the quality of the solution. The most common way to mitigate these effects is through the introduction of additional ordinate directions, leading to significantly larger problems to be solved.

Ray effects are a fundamental shortcoming of the DOM as they are a direct consequence of the decision to restrict the angular flux to a discrete number of points. However, there have been several strategies developed to reduce their occurrence. One approach is to increase the number of discrete ordinates so that every element in the spatial domain is connected to the source regions along a discrete ordinate; another technique is to select a more rotationally invariant set of ordinate directions. It has been shown (in [39]) that though the introduction of additional directions was necessary, a much better improvement may be achieved by improving the set of chosen directions. Another strategy is to use artificial source terms which recast the discrete ordinates equations as the spherical harmonics equations. As the spherical harmonics equations are known to not suffer from ray effects, this eliminates the spurious fluctuations from the solution, at the expense of inducing a coupling over the streaming terms [40].

In [32] Briggs et al. used the fact that the DOM can be written as a DG finite element approximation in angle with piecewise constant basis functions in order to mitigate ray effects. They tested piecewise constant basis functions as well as piecewise bilinear finite elements and found that both schemes mitigated the ray effects when compared to discrete ordinates. In [41] Kanschat also used a piecewise constant finite element approximation in angle and presented a novel method of finding a high quality partition of the sphere into triangles, and consequently a new set of quadrature points for the discrete ordinates method. An angular finite element discretisation similar to these is implemented in this thesis, except that we permit the finite elements to have arbitrarily high order; in Chapter 3 we discuss further the reduction of this scheme to a system of discrete ordinates.

For the design of a set of discrete ordinates one seeks a set of points and weights that integrates the angular domain as accurately as possible, using as few points as possible to minimise computational requirements. For simplified geometries, such as one dimensional problems where the angular domain is reduced to the real interval $[-1, 1]$, the quadrature points and weights can be chosen to correspond to a Gauss-Legendre quadrature, which is the most accurate scheme for the integration of an unweighted analytic function on a real interval. When there is isotropic scattering, this choice can be shown to be equivalent to the spherical harmonics method in angle [21] and thus not subject to any ray effects. It is possible, for spatial domains consisting of a thin plate, to simplify further the 2D neutron transport problem so that the angular domain is reduced to a unit circle. This is equivalent to setting the polar coordinate to $\frac{\pi}{2}$ and is the strategy adopted by Johnson and Pitkäranta in [8], as well as Baker in [42] and Bennison in [43]. For the integration of a smooth function on a circle it is well known that evenly spaced, evenly weighted quadrature points provide exponential convergence (see [44] for a fascinating explanation of this phenomenon).

For the full neutron transport equation, the angular domain is the surface of a sphere, so there is no definitive method for the design of a system of discrete ordinates. This is because an optimal quadrature for the integration of a sphere is not known. Many methods have been proposed over the years and in [45], Koch and Becker analyse and evaluate the accuracy of several of the most popular of these. There is a collection of generally accepted design principles that all good sets of discrete ordinates try to meet in order to remain physically realistic, cf. [46]. These are listed below.

- Every ordinate direction must be on the unit sphere, i.e. $\forall i$, $|\boldsymbol{\Omega}_i| = 1$.
- Every ordinate weighting must be positive. This ensures that the error will be minimised.
- The number of neutrons must be conserved. This is equivalent to requiring that

$$\sum_{i=1}^{N_O} w_i \left(\mathbf{n} \cdot \boldsymbol{\Omega}_i\right) = 0, \tag{1.5.11}$$

  for all $\mathbf{n} \in S_2$.
- The set of ordinates must be invariant under any rotation about the centre of the sphere.

There are many choices of spherical quadrature schemes that will satisfy the first two of these conditions. The third condition is also easily satisfied if for each quadrature point there is a corresponding point with the same weighting at a rotation of π around S_2 from the first point. It is therefore how the user seeks to satisfy the final principle that distinguishes the various systems of discrete ordinates. It is clear that with a finite number of ordinate directions it is not possible to perfectly satisfy this condition. Instead one seeks a set of ordinate directions that is invariant with respect to some specially chosen symmetry groups, such as the symmetries of a regular octahedron aligned with the Cartesian coordinate axes and the set of cyclic rotations about some axis.

The angular finite element meshes exploited in this thesis will be constructed based on a rectangular partitioning principle similar to that described in [47]. This method was chosen because it respects the set of cyclic rotations about the z-axis, and is therefore well suited to 2D problems where there is no variation in the spatial solution along this axis. When a piecewise constant approximation is used in angle then the resulting finite element mesh reduces to a set of discrete ordinates that all have equal weightings. This is another desirable property that has been used to motivate the development of many discrete ordinate schemes, see [48]. See Figure 1.1 for an illustration of the first three meshes in the series utilised for the computations in this thesis. For these meshes the longitudinal separator lines are equally spaced around the azimuthal coordinate and the latitudinal separator lines are chosen so that all finite elements have equal area.

Spherical harmonics methods

The spherical harmonics, or P_N, treatment of the angular variable was first suggested for neutron transport in the 1940s by Mark in [49] and has since become, alongside the discrete ordinates schemes, one of the most popular methods. The main advantages of the P_N methods over the discrete ordinates methods are that they are invariant under rotation of the principal axes and are immune to ray effects. Therefore, they can produce a high quality approximation for the angular solution even when a relatively small number of basis functions are employed, particularly when the solution is close to isotropic. This is because they represent a continuous solution in the angular domain; in contrast, S_N methods select a set of discrete points on which to collocate the solution. They are also well suited to problems with anisotropic scattering, as the scattering kernel can be expanded exactly in terms of the spherical harmonics functions.

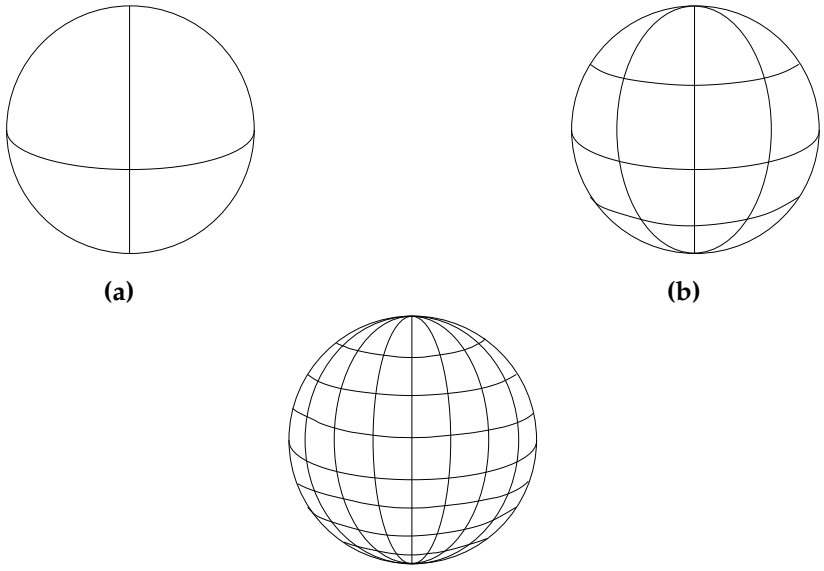


Figure 1.1: Three angular meshes constructed using the rectangular partitioning method. Mesh (a) has a single angular element in each principal triangle, mesh (b) has four and mesh (c) has 16.

For these reasons, the P_N methods can be used to accurately model a wide variety of neutron transport problems.

One arrives at the spherical harmonics equations by first expanding the angular flux as the sum over the set of solutions to Laplace's equation on the surface of the sphere. For the time invariant scalar flux this is given by

$$\psi(\mathbf{r}, \boldsymbol{\Omega}, E) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \psi_{l,m}(\mathbf{r}, E) Y_{l,m}(\varphi, \theta), \tag{1.5.12}$$

where φ is the polar coordinate and θ is the azimuthal coordinate. Here, the $\psi_{l,m}(\mathbf{r}, E)$, $l = 0, \cdots, \infty$, $m = -l, \cdots, l$, represent the angular flux at the angular moment associated with the spherical harmonic function indexed by l and m, i.e.,

$$Y_{l,m}(\varphi, \theta) = \sqrt{\frac{(2l+1)}{4\pi} \frac{(l-m)!}{(l+m)!}} P_l^m(\mu) e^{im\theta}. \tag{1.5.13}$$

Here, $P_l^m(\mu)$ is the associated Legendre function of the cosine of the polar variable, $\mu = \cos(\varphi)$. The square root term is a non-unique normalisation constant. These functions form a complete orthonormal basis for the angular solution space with

$$\int_{S_2} Y_{l,m}(\varphi, \theta) Y^*_{l',m'}(\varphi, \theta) \, d\varphi \, d\theta = \delta_{l,l'} \delta_{m,m'}, \tag{1.5.14}$$

where $Y^*_{l',m'}(\varphi,\theta)$ is the complex conjugate of $Y_{l',m'}(\varphi,\theta)$, and $\delta_{i,j}$ denotes the Dirac delta function. To obtain the discrete version of the P_N approximation, the sum to infinity in (1.5.12) is truncated with $l = N_{SH}$. The scattering cross section and the external source term are also expanded in terms of the spherical harmonics functions and both are substituted into the neutron transport equation. Finally, the resulting equation is multiplied by the complex conjugate of each $Y_{l,m}(\varphi,\theta)$ and integrated over the angular domain to yield a set of coupled equations to be solved for the flux moments $\psi_{l,m}(\mathbf{r},E)$.

The main disadvantage of the P_N methods when compared to discrete ordinates methods is that, even after utilising the orthogonality property of the spherical harmonics functions, the set of equations that results is still complicated and difficult to solve. This is due to the coupling that is introduced in the streaming term $\boldsymbol{\Omega}\cdot\nabla\psi(\mathbf{r},\boldsymbol{\Omega},E)$, as well as the fact that the number of angular moments grows with $\mathcal{O}(N^2_{SH})$. Furthermore, the fact that the flux at each angular moment is continuously represented for all directions on the sphere makes the imposition of vacuum boundary conditions difficult. This property also means that the direction of travel of information at each angular moment is not distinct, therefore the sweeping method of preconditioning (discussed in detail in Chapter 3) cannot be easily utilised. For a more comprehensive description of the P_N methods, see [21] and [22].

We note that there is a similarity between the numerical approximation of the neutron transport equation based on the P_N approach and the higher order finite element approximation proposed in this thesis. Indeed, when $m = 0$, the associated Legendre functions reduce to the Legendre polynomials which form the basis for the finite element method introduced in Chapter 3. The principal differences between these two methods are that, for the finite element approximation, the basis will be mapped onto a patch on the sphere, therefore losing its orthogonality property for the basis functions aligned in the polar direction. However, the finite element method in angle will allow the partition of the angular flux solution into piecewise continuous sets of directions on which the direction of travel of information is known; this will then facilitate the use of a sweeping preconditioner.

Diffusion approximation

The final method for treating the angular variable that we discuss is the reduction of the neutron transport equation to a diffusion equation. In certain situations the full integro-

differential equation in space and angle may be reduced to a diffusion equation in just the spatial dimensions. This approximation greatly reduces the computational requirements for determining a numerical solution to the neutron transport equation and, moreover, it can provide very accurate solutions for problems with highly isotropic angular solutions. For a diffusion approximation to be relevant, some key assumptions have to be made regarding the underlying physical problem. In particular the source term must be independent of angle and all scattering that occurs must be isotropic. Finally, the probability of a neutron being scattered must be much larger than the probability of it being absorbed (see [50]). If these conditions are met, then the steady state neutron diffusion equation can be derived from the P_1 spherical harmonics expansion of the steady state neutron transport equation, leading to

$$\nabla \cdot (D(\mathbf{r}, E)\nabla\phi(\mathbf{r}, E)) + \Sigma_t(\mathbf{r}, E) = Q(\mathbf{r}, E), \tag{1.5.15}$$

where the solution $\phi(\mathbf{r}, E)$ is the scalar flux which is independent of angle. See [21] for a detailed derivation of this equation. We have introduced the diffusion coefficient D, which includes a factor of $\frac{1}{\Sigma_t(\mathbf{r},E)}$. The presence of this term represents one of the drawbacks of the use of diffusion approximation: the difficulty of representing void regions in the domain. A further problem with the diffusion approximation is its unsuitability for modelling problems where the solution has a strong dependence on the angular variable. Also there is a risk that the solutions computed via the standard diffusion approximation can violate causality. In other words, $\phi(\mathbf{r}, E)$ could be greater than the neutron density multiplied by the maximum transport speed. Various methods have been developed to remedy this including the use of explicit flux limiting (as implemented by Levermore in [51]), as well as the re-derivation of the diffusion approximation to naturally include a limit on the scalar flux (see the work of Pomraning in [52]).

For problems where the solution to the diffusion approximation does not provide a good model for the neutron distribution, the neutron diffusion equation can still be useful as the basis of a preconditioner for the full neutron transport equation. This is the strategy of the diffusion synthetic acceleration method for solving source and eigenvalue problems which will be discussed again in Chapter 3. The inverse of the diffusion operator is particularly well suited to being the basis of a preconditioner because of the fact that it captures the effect of the scattering term in the full equation, which is not well approximated by the more common transport sweep method of preconditioning.

Solving the diffusion equation is also efficient in terms of memory requirements and operation counts which makes it suitable for repeated application.

1.5.3 Spatial discretisation methods

In general the spatial part of the solution to neutron transport problems is much more complicated than the solutions found in the angular domain. This is a consequence of the great variety in the spatial geometries on which one is called upon to solve problems in neutronics. Therefore, there has been considerable focus on providing accurate solutions to the spatial problem. This is in contrast to the angular discretisations where the focus has been more towards retaining accuracy whilst limiting the total problem size of the approximations. As a result, over the years many methods for the numerical solution of the spatial problem have been developed. In this section we consider deterministic methods in three broad categories: finite difference methods, characteristics based methods and finite element methods. Monte Carlo methods are also used in the field, where they are sometimes the only effective method for certain extreme geometries, such as pebble bed reactors which comprise a large number of tennis ball-sized 'pebbles' each containing thousands of micro-fuel particles. They are also well suited to parallelisation and there exist many well established codes that implement them such as the MCNPTM code by Briesmeister et al. [53] and the MCML code by Wang et al. [54]. On the other hand the slow convergence of Monte Carlo methods can lead to considerable computational requirements in order to compute sufficiently accurate solutions.

Finite difference methods

Some of the earliest attempts at solving the neutron transport equation utilised finite difference methods. Finite difference methods approximate the spatial solution on a highly structured set of points in the spatial domain, usually aligned with the Cartesian coordinate axes, [55]. This requirement for a highly structured grid means that the treatment of irregular spatial domains can be difficult and therefore the number of problems that can be solved with finite difference methods is somewhat limited, especially for problems in more than one dimension. However, when they are applicable, finite difference methods do yield very sparse and easy to solve linear systems.

Characteristics based methods

The method of characteristics for solving hyperbolic PDEs has been applied many times over the years to the solution of neutron transport problems from its first implementation on practical geometries by Askew [56] in 1972. For a review of many of this class of schemes, we refer to [57]. Characteristics methods are well suited to discrete ordinate approximations in angle; they are based on replacing the partial differential operator by a total derivative along the characteristic lines defined by the direction of travel of the neutrons. The resulting ODEs can then be solved by analytical methods and the solution in the various directions can be combined to build a numerical approximation of the flux within each element in a mesh over the spatial domain. Characteristics methods have the advantage over schemes such as the finite difference method, in that they are capable of computing solutions on irregular meshes, though computational challenges do arise for meshes with fine grained regions. This is because every element in the mesh must have at least one characteristic line passing through it for each of the selected characteristic directions; therefore, many closely packed, parallel characteristic lines are needed to resolve fine regions in the mesh. Moreover, determining the intersection of each characteristic line with every element that they encounter can be computationally expensive. Despite this, traditional (long) characteristics methods are used by several commercial codes, including the CACTUS module of Serco Assurance's WIMS software [58].

The challenges associated with the long characteristics method has led to work on the development of the method of short characteristics, which were first proposed by Takeuchi in [59]. Short characteristics methods differ from long characteristics methods in that the set of characteristics employed are defined element by element, rather than across the entire spatial domain. This means that for regions where a finer mesh is needed to compute an accurate representation of the solution, the introduction of additional elements is not inhibited by excessive computational requirements. Though the original work by Takeuchi was to create a linear approximation of the solution, the short characteristics method can be extended to higher order, see [42].

Finite element methods

Some of the most popular deterministic schemes for discretising the spatial part of the neutron transport equation are based on employing finite element methods. This is due to their accuracy, versatility and their ability to use unstructured meshes to accurately represent physical phenomena on realistic computational domains. Finite element methods approximate the solution in a finite element space consisting of the span of a set of piecewise polynomial functions, known as basis functions, defined on a mesh over the computational domain. It is the design of these spaces which distinguish the two broad categories of finite element methods: the continuous-Galerkin (CG) finite element method and the DG finite element method. In the CG finite element method, the finite element space consists of continuous piecewise polynomials, whereas the DG method permits discontinuities across inter element-boundaries.

CG methods were initially applied to two dimensional neutron diffusion in the early 1970s [60]. For the application of CG methods to hyperbolic problems, such as the full neutron transport equation, the quality of computed solution can be deteriorated by the occurrence of spurious oscillations [61]. One way to eliminate these is to employ streamline upwind Petrov-Galerkin methods, which mitigate these unphysical oscillations by introducing an artificial diffusion term in the streamwise direction. This method has proved very successful and is still commonly employed for neutron transport problems, see [62] and [63].

The DG finite element method was in fact originally developed for neutron transport by Reed and Hill in their influential 1973 paper [6]. This demonstrated the superiority of employing discontinuous finite elements in comparison to their continuous counterparts for two dimensional neutron transport calculations. Since then many authors have used DG methods for solving neutron transport problems; see for example [7], [64] and [65]. DG methods have the advantage over CG methods in that they are better at resolving steep gradients in the computed flux, which occur frequently in neutron transport problems, cf. the benchmarks computed later in this thesis. A key disadvantage of the DG method compared to the CG method is that there are more degrees of freedom when the same mesh and polynomial degree are employed. However, the flexibility and improved stability and robustness of DG methods makes them a highly desirable set of schemes for large classes of PDE problems, including the neutron transport problem. The DG finite element method is the main numerical method considered

in this thesis and as such we will revisit it in Chapter 3.

Ridgelet methods

In the last couple of years we have seen the development of a new method for the discretisation of linear transport problems based on ridgelets. Ridgelets belong to the class of multiscale systems known as 'α-molecules' that also includes wavelets, shearlets and curvelets, see [66] for a discussion of this framework. The functions in this group are known to each be particularly well suited to representing certain types of functions optimally, for example wavelets are known to accurately represent functions with point singularities and ridgelets are known to accurately represent functions with line singularities. In [67] Grohs and Obermeier proved that ridgelets are optimal for the representation of solutions to linear transport equations in the sense that the error decays optimally for an N term approximation. In [68] they applied ridgelets to the radiative transport equations which, like the neutron transport equation, can be represented as a series of linear transport equations via a discrete ordinates angular discretisation. In [69] they used a fast Fourier transform to develop a ridgelet based solver for radiative transport that results in uniformly well conditioned systems. Though it seems that these methods have not yet been applied in the field of neutronics, their successful utilisation in the radiative transport field and similarity between the radiative transport equation and the neutron transport equation suggests that they could have great potential for the development of future optimal neutron transport solvers.

1.6 Thesis outline

The thesis is structured as follows. In Chapter 2 we consider the direct solution of DG matrices for linear PDEs. In particular, we present a preprocessor that exploits the particular structure of such matrices in order to compute a more efficient factorisation when using sparse Gaussian elimination. This algorithm utilises the fact that the matrices arising from DG methods have a block sparse structure with dense blocks. The structure is determined by the underlying finite element mesh. Indeed the matrix possesses a dense block on the diagonal, which is associated with each element domain, together with further dense blocks above and below the diagonal associated with each interface between neighbouring elements. The preprocessor works by computing a

fill-in reducing ordering for the structure of the blocks and then expanding it to the full matrix. It is implemented for symmetric and non-symmetric solvers and tested on a series of high order DG matrices, including a large three dimensional problem. We demonstrate the benefits of such an approach both in terms of reduced memory requirements and reduced computational time.

In Chapter 3 we present a DG finite element method for the neutron transport equation. In contrast to many authors who exploit discrete ordinates methods or spherical harmonics methods in the angular variables, we employ a high order DG method in both the spatial and angular domain. This method provides an accurate discretisation for the angular variables, as well as facilitating the development of a reliable dual weighted residual based error estimator for the error arising in both the spatial and angular parts of the discretisation scheme. A variety of parallel algorithms are then tested for the solution to the neutron transport source and eigenvalue problems. These algorithms all utilise the preprocessing method for block matrices that was developed in Chapter 3. We demonstrate the advantage of utilising solvers based on Krylov subspace methods for high order problems. The chapter is concluded with results demonstrating the rates of convergence of the proposed DG method.

In Chapter 4 we consider neutron transport criticality problems and derive a computable *a posteriori* error representation formula for our DG method, which provides a high quality estimate for the error in the dominant eigenvalue. Moreover, we show how projections between the different finite element spaces can be employed to quantify the relative contributions to the error arising in the spatial and angular discretisations. We then develop an h-adaptive algorithm as well as an hp-adaptive algorithm for the computation of the multiplicative eigenvalue. We present results for a monoenergetic test problem, demonstrating that the effectivities are close to one for each algorithm. The efficiency of the hp method is illustrated by results displaying computed eigenvalues that are as accurate as for the h-method, but with an 80% reduction in the problem size.

In Chapter 5 we present results from a selection of challenging industrial benchmark problems that demonstrate the power and efficiency of the proposed adaptive algorithms. In particular we demonstrate that the hp-refinement algorithm can achieve exponential convergence with respect to the number of degrees of freedom in the finite element space. We conclude, in Chapter 6, with some final summarising remarks and

a discussion of possible directions for further research.

CHAPTER 2

Methods for Discontinuous-Galerkin Equations for General PDEs

In a wide variety of academic and industrial problems, discontinuous-Galerkin (DG) finite element methods are used to find the numerical solution to partial differential equations. Usually the most computationally intensive step in the application of such methods is the solution of a large, possibly unsymmetric, matrix with a symmetric block structure. In this chapter we consider a method for preprocessing a DG matrix prior to factorisation and solution by a sparse direct solver. The block structure of such a matrix corresponds directly to the mesh selected over the computational domain, with a diagonal block corresponding to each element in the mesh and off diagonal blocks corresponding to each interface between two neighbouring elements. The size and shape of the blocks is determined by the basis functions and order of approximation on each element as well as the numerical fluxes selected at the element boundaries. As a consequence, when a varying order of approximation is used on different elements the size of the square blocks on the diagonal can vary and the off-diagonal blocks become rectangular. We assume that these blocks have sufficiently few zeros to be treated as dense submatrices.

In this chapter we consider how this structure, common to all DG matrices, may be used to adapt existing direct linear solvers to solve the present class of problem at a greater speed and with reduced memory requirements. The two linear solvers consid-

ered in this chapter are the `MA57` symmetric solver and the `MA41` unsymmetric solver, which are both from the HSL Mathematical Software Library, [70]. Later in the thesis we will employ the block preprocessor for `MA41` as part of an efficient preconditioner for the high order DG discretisation of the neutron transport equation.

Before proceeding, we shall briefly present a description of how a matrix equation may be extracted from a DG method for a linear PDE. A more detailed description, specific to the neutron transport equation, is included in Section 3.2.1. If a DG finite element space, $\mathcal{V}_h$, is defined as the span of a set of basis functions $\mathcal{V}_h = \text{Span}(\xi_i)$, where $i = 1, \cdots, N$, then a DG method, such as those presented in [71], may be written in terms of the bilinear functional $a(\cdot,\cdot)$ and the linear functional $l(\cdot)$ as the following. Find $u_h \in \mathcal{V}_h$ such that:

$$a(u_h, v_h) = l(v_h), \tag{2.0.1}$$

for all $v_h \in \mathcal{V}_h$. Here $u_h = \sum_{j=1}^{N} \mathbf{u}[j]\,\xi_j$ is the DG solution, where $\mathbf{u} \in \mathbb{R}^N$. Then the DG solution can be found by solving the matrix equation $\mathbf{A}\mathbf{u} = \mathbf{b}$, where

$$\mathbf{A}[i][j] = a(\xi_j, \xi_i), \tag{2.0.2}$$

and

$$\mathbf{b}[i] = l(\xi_i). \tag{2.0.3}$$

We note that each ξ_i is nonzero only on the closure of a single finite element and that for the vast majority of combinations (i, j), i and j will index different finite elements which do not share any boundaries, therefore most values of $a(\xi_j, \xi_i)$ will be zero, so the matrix $\mathbf{A}$ will be sparse. $\mathbf{A}$ gets its block-sparse structure from the fact that, for high order methods, each finite element will have many basis functions associated to it. The indices of these may be ordered contiguously for each finite element so that the matrix comprises dense matrix blocks associated with each finite element and element interface.

2.1 Sparse Gaussian elimination

Before proceeding to look at how a preprocessor may be developed to enable `MA57` to solve the matrices from DG methods more efficiently, we will consider an overview of the three principle computational phases that most direct sparse solvers use to solve

the linear system. These phases are the preprocessing phase, `ANALYSE`, the factorisation phase, `FACTORISE` and the solution phase, `SOLVE`. For a more detailed overview of the theory, algorithms and data structures necessary for the implementation of direct methods for sparse linear systems, we suggest the books by Duff, Erisman and Reid [72] and Davis [73].

Nearly all direct linear solvers implement some version of Gaussian elimination to factorise the matrix and if they are to provide stable and accurate solutions, they must select a suitable pivot order. For sparse direct solvers the order in which the pivots are eliminated is also important for determining the number of floating-point operations and the amount of memory required to perform the factorisation. This is because a good ordering will reduce the amount of fill-in, that is the number of nonzero entries that are introduced during the factorisation. If the symmetric positive-definite matrix $\mathbf{A}$ is factorised via an LDLT decomposition, for example

$$\mathbf{PAP}^\top = \mathbf{LDL}^\top, \tag{2.1.1}$$

where $\mathbf{P}$ is a permutation matrix, $\mathbf{D}$ is a diagonal matrix and $\mathbf{L}$ is a unit lower triangular matrix, then the fill-in is the number of entries in the pattern of $\mathbf{L} + \mathbf{L}^\top$, that are not present in the pattern of $\mathbf{A}$. As the amount of memory required is crucial to the overall performance of the software, it is important that a good ordering is chosen. However, finding the optimal permutation for minimising fill-in is an NP-complete problem (see [74]), therefore direct solvers must use a heuristic method in order to obtain an ordering which reduces fill-in as much as possible.

Finding a fill-in reducing ordering is one of the key tasks performed by the `ANALYSE` phase of the solver. Other operations performed by `ANALYSE` are the transfer of user data into the internal data structures of the solver and the construction of data structures necessary for the following phases of the solver, such as the mappings and the assembly tree for the ordering that was computed. The `MA57` solver, [75], uses one of three well known algorithms to compute the pivot order: the minimum degree algorithm, the approximate minimum degree (AMD) algorithm and the nested dissection algorithm. The minimum degree algorithm works by, at each stage, looking at all of the rows that correspond to variables that have not yet been chosen as pivots and selecting the next pivot to be the diagonal entry in the row for which there is fewest entries, ignoring entries in columns corresponding to variables that have already been selected as pivots. As this can be computationally expensive, Amestoy et al. developed the

approximate minimum degree algorithm, in [76], which computes an ordering of similar quality, though requiring less computation. These two algorithms are examples of local strategies for choosing an ordering. That is because at each stage they only consider the rows and columns of the variables that have not already been chosen, trying to minimise the fill-in inherent from selecting the next pivot.

The nested dissection algorithm of George, [77], is an example of a global ordering strategy in that it seeks to minimise fill-in globally. In general it produces superior orderings to minimum degree methods when applied to finite element discretisations in two and three spatial dimensions (see [73]). It works by considering the symmetric matrix as an undirected graph, then recursively partitioning that graph into a hierarchy of subgraphs. At each stage in the recursion each subgraph is partitioned into two evenly sized subgraphs by removing a subset of its vertices, known as the node separator. It is known that the fill-in introduced by each stage of this procedure is limited to the square of entries corresponding to the node separator that is removed from each subgraph, together with the corresponding off-diagonals, thus nested dissection may be used to reduce fill-in, given that small node separators can be found. The asymptotic superiority of nested dissection for ordering regular finite element meshes in two dimensions has been proved by Lipton et al. in [78]. Their analysis was extended to three dimensions by Duff et al. in [79].

Following the `ANALYSE` phase, the `FACTORISE` phase then proceeds to copy the numerical values into the internal data structures and factorises them using the pivot order decided by `ANALYSE`. For matrices that are not symmetric positive definite the software must monitor the stability of the factorisation and make deviations from the pivot order computed during the matrix preprocessing if a zero, or relatively small, pivot is detected.

For the `MA57` solver the factorisation involves computing a decomposition of the form $\mathbf{PAP}^\top = \mathbf{LDL}^\top$, where $\mathbf{D}$ is a diagonal matrix if $\mathbf{A}$ is positive definite, or a block diagonal matrix if it is not. For the `MA41` solver the factorisation involves computing an LU decomposition, $\mathbf{PAQ}^\top = \mathbf{LU}$, where $\mathbf{U}$ is upper triangular and $\mathbf{P}$ and $\mathbf{Q}$ are both permutation matrices. See [72] and [73] for explicit details of these factorisations.

Finally the `SOLVE` phase permutes the right-hand side to the ordering used by the solver, then performs a forward substitution followed by a backward substitution with the factors computed in `FACTORISE`, before finally permuting the solution back to the user's

ordering and returning the solution.

We note that when the matrix $\mathbf{A}$ is symmetric or close to symmetric, it is usual practice to perform `ANALYSE` using just the sparsity pattern of $\mathbf{A}$ (or $\mathbf{A} + \mathbf{A}^\top$ when $\mathbf{A}$ is nearly symmetric) as opposed to requiring the numerical values of the entries in addition to the pattern. This is important for the present implementation because it enables us to treat the symmetric block structure as the pattern of a symmetric matrix of considerably reduced size. We may then use an existing `ANALYSE` subroutine to find a good ordering for the blocks before expanding the reduced ordering and `ANALYSE` output data to the original matrix. Codes for the analysis of unsymmetric matrices however frequently require numerical values on which to perform stability tests before selecting each pivot, thereby forming the factors as a byproduct of the `ANALYSE` phase. It is not obvious how efficient tests on blocks as opposed to individual entries could be designed, though if such a test was developed the use of optimized `BLAS` subroutines could potentially improve the performance of `ANALYSE` and `FACTORISE` subroutines for matrices with unsymmetric block structures.

Treating each block as an individual entry for the preprocessing and then expanding the output data structures forces entries from the same block to be kept together and eliminated at the same time during the factorisation. This was a key goal for the present work and it makes sense not just structurally in that it keeps the dense blocks together, but also geometrically in that each block corresponds to either a finite element or an interface between elements in the finite element mesh. It would seem intuitive that variables associated with the same finite element are eliminated at the same time, as opposed to variables from different parts of the domain being eliminated together. Simple tests on matrices from DG methods show that the orderings produced by sparse direct solvers do, in general, split up variables from the same block – and therefore from the same part of the computational domain – prior to factorisation. The preprocessing method presented in this chapter is designed to preserve the block structure. Furthermore, as blocks in the structure can be of size 25×25 or larger, and larger still for 3D problems, the size of the ordering problem tackled by the `ANALYSE` phase of the solve can be reduced to a fraction of its original size. This is indeed significant when we consider that the ordering algorithm used might involve $\mathcal{O}(N^2)$ operations, where N is the size of the matrix. The potential gain in time, as well as memory, of reducing N to $\frac{N}{25}$ might be considerable. In Section 2.3 we show plots displaying the speedup that can be realised in the `ANALYSE` phase.

2.2 Adapting MA57 for block matrices

The goal of the present chapter is to show how a specific property of the matrices being considered, the fact that all of their entries are contained within dense blocks in a block sparse structure, may be exploited to yield an improvement in the performance of two direct sparse linear solvers MA57 and MA41.

In order to make use of this block symmetric structure we will look specifically at the ANALYSE phase of the MA57 solver, adapting it so that it reorders the blocks, treating each block as if it were a single entry in a reduced matrix as opposed to a collection of entries. When this pivot ordering has been computed, it is then expanded to the full structure. Following this, all other information output by the ANALYSE routine must be expanded to correspond to the expanded ordering before proceeding to the FACTORISE and SOLVE phases of MA57. The ordering from the block ANALYSE version of MA57 may also be passed to the MA41 solver to find the solution to an unsymmetric system with the same symmetric block structure. MA41 can then use this ordering to prepare its own internal data structures. We note, however, that though the ordering passed to MA41 will have all the variables from each block in contiguous positions, the solver is not forced to assign them to the same node in the internal tree data structure. Indeed, it is well known that user supplied orderings in general perform less well than orderings computed internally within sparse direct solvers.

An illustration of the difference and some of the possible benefits of applying ANALYSE

block by block is illustrated by the following symmetric example matrix

$$\mathbf{A} = \left(\begin{array}{ccc|ccc|ccc|ccc} 1 & \times & \times & \times & \times & \times & \times & \times & \times & & & \\ & 2 & \times & \times & \times & \times & \times & \times & \times & & & \\ & & 3 & \times & \times & \times & \times & \times & \times & & & \\ \hline & & & 4 & \times & \times & & & & \times & \times & \times \\ & & & & 5 & \times & & & & \times & \times & \times \\ & & & & & 6 & & & & \times & \times & \times \\ \hline & & & & & & 7 & \times & \times & \times & \times & \times \\ & & & & & & & 8 & \times & \times & \times & \times \\ & & & & & & & & 9 & \times & \times & \times \\ \hline & & & & & & & & & 10 & \times & \times \\ & & & & & & & & & & 11 & \times \\ & & & & & & & & & & & 12 \end{array}\right), \tag{2.2.1}$$

with block structure

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \mathbf{A}_{1,3} & \\ & \mathbf{A}_{2,2} & & \mathbf{A}_{2,4} \\ & & \mathbf{A}_{3,3} & \mathbf{A}_{3,4} \\ & & & \mathbf{A}_{4,4} \end{pmatrix}. \tag{2.2.2}$$

The structure of $\mathbf{A}$ corresponds to the structure of a DG matrix with a $p = 1$ approximation, on a regular 2×2 mesh. Though it is too small to demonstrate potential performance advantages of the block method, it does demonstrate three benefits of utilising a block preprocessor: the potential to reduce both the amount of memory required and the operation count of the `ANALYSE` routine, as well as eliminating the possibility of unnecessary reordering within the blocks.

First consider the ordering computed by the METIS implementation of nested dissection, [80]. When the block structure is not taken into consideration, the pivot order computed is

$$\begin{bmatrix} 3 & 1 & 2 & 12 & 10 & 11 & 7 & 8 & 9 & 6 & 4 & 5 \end{bmatrix}, \tag{2.2.3}$$

which reorders the matrix to

$$\mathbf{A}_{\text{reord1}} = \left(\begin{array}{ccc|ccc|ccc|ccc} 3 & \times & \times & & & & \times & \times & \times & \times & \times & \times \\ & 1 & \times & & & & \times & \times & \times & \times & \times & \times \\ & & 2 & & & & \times & \times & \times & \times & \times & \times \\ \hline & & & 12 & \times & \times & \times & \times & \times & \times & \times & \times \\ & & & & 10 & \times & \times & \times & \times & \times & \times & \times \\ & & & & & 11 & \times & \times & \times & \times & \times & \times \\ \hline & & & & & & 7 & \times & \times & & & \\ & & & & & & & 8 & \times & & & \\ & & & & & & & & 9 & & & \\ \hline & & & & & & & & & 6 & \times & \times \\ & & & & & & & & & & 4 & \times \\ & & & & & & & & & & & 5 \end{array}\right). \tag{2.2.4}$$

The first thing we notice about this is that blocks $\mathbf{A}_{1,1}$ and $\mathbf{A}_{4,4}$ have been reordered internally by the software. This is in fact redundant, as `MA57` does not consider the numerical values when it analyses the matrix. In this case the blocks have not been split up, with the variables corresponding to each diagonal block, $\{1,2,3\}$, $\{4,5,6\}$, $\{7,8,9\}$ and $\{10,11,12\}$, respectively, remaining consecutive in the reordered matrix. This is merely good fortune on our part as there is nothing which would prevent the splitting of blocks for general matrices.

Now consider the ordering computed by METIS applied only to the block structure, and then expanded to the full matrix. The pivot order computed for the block structure is $\left[\begin{array}{cccc} \mathbf{A}_{2,2} & \mathbf{A}_{3,3} & \mathbf{A}_{4,4} & \mathbf{A}_{1,1} \end{array}\right]$ which expands to

$$\left[\begin{array}{cccccccccccc} 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 1 & 2 & 3 \end{array}\right]. \tag{2.2.5}$$

Then **A** reordered by blocks is

$$\mathbf{A}_{\text{reord2}} = \left(\begin{array}{ccc|ccc|ccc|ccc} 4 & \times & \times & & & & \times & \times & \times & \times & \times & \times \\ & 5 & \times & & & & \times & \times & \times & \times & \times & \times \\ & & 6 & & & & \times & \times & \times & \times & \times & \times \\ \hline & & & 7 & \times & \times & \times & \times & \times & \times & \times & \times \\ & & & & 8 & \times & \times & \times & \times & \times & \times & \times \\ & & & & & 9 & \times & \times & \times & \times & \times & \times \\ \hline & & & & & & 10 & \times & \times & & & \\ & & & & & & & 11 & \times & & & \\ & & & & & & & & 12 & & & \\ \hline & & & & & & & & & 1 & \times & \times \\ & & & & & & & & & & 2 & \times \\ & & & & & & & & & & & 3 \end{array}\right). \tag{2.2.6}$$

We notice that the pattern of the second ordering is precisely the same as the first, in other words we have achieved an equivalent outcome with fewer operations. With larger problems the second preprocessing method would have taken less time and memory than the first, though with extremely small problems such as this one the memory required to store the block structure and the operations required for the expansion negate these advantages. The blocks in the reordered matrix remain precisely the same despite being permuted, which is a property of the ordering that is preserved when moving to larger matrices with more irregular block structures. For larger DG matrices, ordering algorithms applied to the full matrix frequently break up the blocks, resulting in reordered systems where the variables from the same finite elements are in non-consecutive positions in the pivot order. As we see from the results in the next section, this leads to poorer quality orderings that lead to more fill-in than can be achieved by algorithms that force blocks to remain contiguous.

2.3 Results

To look at the difference that selecting a block based preprocessing strategy makes to `MA57` and `MA41`, we consider a simple interior penalty DG discretisation of Poisson's equation with Dirichlet conditions on the boundary of a two dimensional square do-

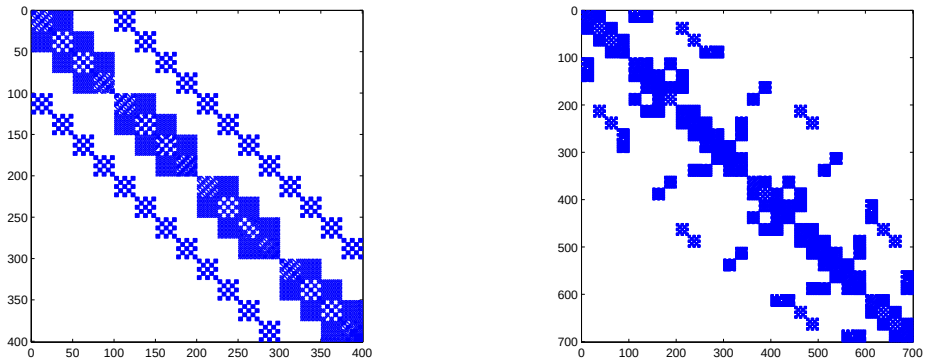


Figure 2.1: The pattern of the first two matrices in a series of matrices generated by a DG method for Poisson's equation.

main: find u such that

$$\nabla^2 u = f \qquad \text{on} \qquad \mathcal{D} = (0,1)^2 \tag{2.3.1}$$

$$u = \hat{g}(\mathbf{r}) \qquad \text{on} \qquad \partial\mathcal{D}, \tag{2.3.2}$$

where $\hat{g}(\mathbf{r})$ is a known function. See [71] for details of this scheme.

A series of matrices of increasing size, N, and varying block sizes were obtained by refining the spatial mesh and increasing the order of approximation on individual elements. The pattern of the first two matrices in this series is given by the two plots in Figure 2.1. Note the clear blocking in both structures and the fact that the second matrix has a less regular structure. The reduced regularity in the second matrix is a consequence of the adaptive mesh refinement that has taken place, with certain areas in the mesh requiring more refinement than others. Table 2.1 displays some information about the series of 2D matrices on which the solvers were tested. The reason for two columns for the number of entries is because `MA57` requires the matrix to be stored in a symmetric coordinate format as opposed to the regular coordinate format used in `MA41`. This assumption permitted the amount of data passed to `MA57` to be reduced considerably. An equivalent assumption was made for the storage of the block structure, with the entries below the diagonal not explicitly stored. Notice how much smaller the block structure is compared to the full matrix by comparing the third and sixth columns. This allows the ordering problem solved by the modified `ANALYSE` subroutine to be reduced by a factor close to the average block size.

Figure 2.2 contains plots illustrating the CPU time taken for the three phases of the `MA57` linear solver for the series of matrices in Table 2.1. Timings were taken both with the

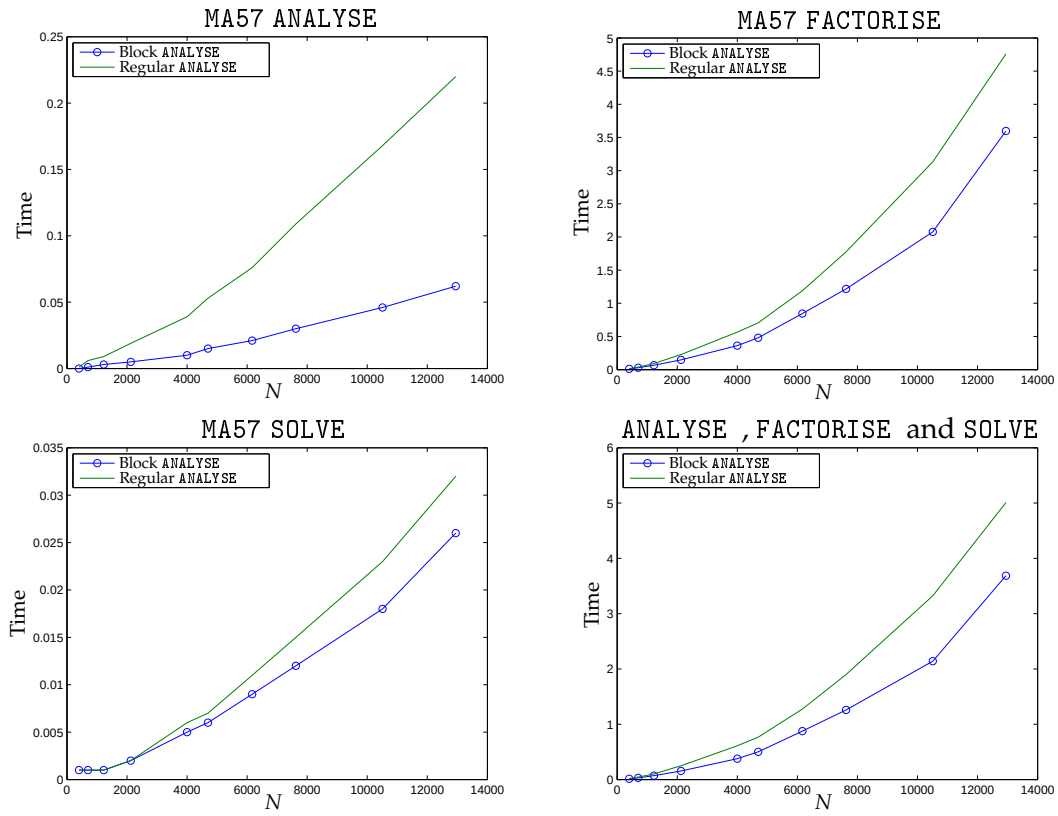


Figure 2.2: The time taken to solve a series of block matrices from a DG method with `MA57`. The data is split into four plots, the time taken (in seconds) for the modified `ANALYSE` phase, then the time taken for the `FACTORISE` and `SOLVE` phases and finally the total time taken to solve the linear system.

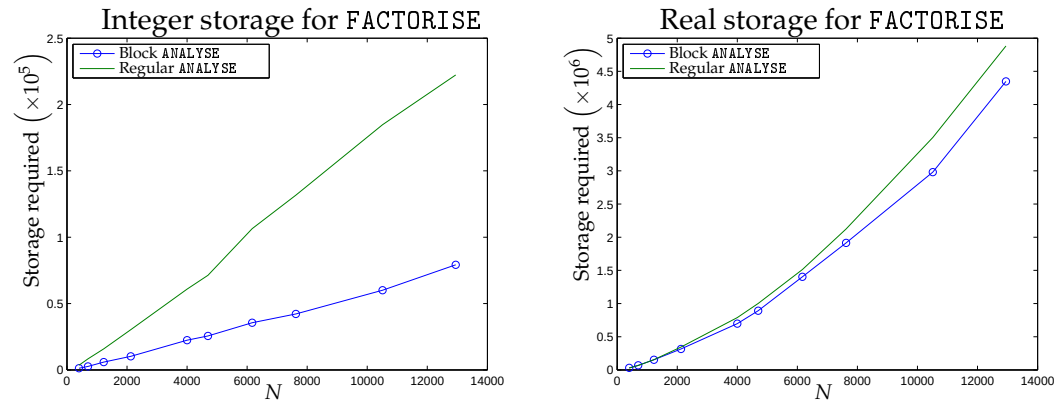


Figure 2.3: Plots displaying the reduction in the memory required to store the factors computed by the `MA57 FACTORISE` subroutine for the cases when a block version of `ANALYSE` preceded the call to `FACTORISE` and for the case when the unaltered version of `ANALYSE` was used.

Matrix	Size	Entries	Entries with symmetric storage	Order of block matrix	Entries in block structure	Largest block size	Average block size
1	400	20 186	10 293	16	40	25	25
2	700	52 012	26 356	28	78	25	25
3	1 225	90 361	45 793	49	140	25	25
4	2 125	186 327	94 226	85	254	25	25
5	4 000	326 548	165 274	160	468	25	25
6	4 696	449 908	227 302	172	508	36	27.3
7	6 168	649 654	327 911	208	610	36	29.7
8	7 624	927 500	467 562	244	734	49	31.2
9	10 511	1 449 359	729 935	319	975	49	32.9
10	12 946	1 929 784	971 365	379	1158	64	34.2

Table 2.1: This table gives information on the series of two dimensional matrices used to test the block version of the `MA57 ANALYSE` subroutine.

block version preprocessor as well as with the original. The AMD algorithm was used for both methods. Observe how the block version of `ANALYSE` performs considerably faster than the unblocked version. However when we consider the reduction in problem size (see the second and fifth columns of Table 2.1) we see that the improvement in speed is not proportional to the reduction in the size of the problem. This is because, following the computation of the ordering, the routine must necessarily proceed to expand the ordering to the full system as well as constructing the other necessary data structures before returning from the `ANALYSE` subroutine. There is also some significant improvement in the `FACTORISE` and `SOLVE` phases as well as in the total time. This is because of a reduction in the fill-in created by the factorisation and is a direct consequence of forcing the solver to keep the blocks together following the `ANALYSE` phase. Indeed when comparing the scales in these plots it is the improvement in the `FACTORISE` phase that makes the biggest difference to the overall performance of the solver.

The improvements in the fill-in introduced by `FACTORISE` are illustrated in Figure 2.3. Observe that although the reduction in the integer storage required appears more dramatic, it is in fact the 10% reduction in the real storage that has the greater effect on the total memory required. Indeed, for the largest matrix, we see a reduction in the storage requirements of approximately 150 000 four byte integers, whereas we see a reduction

of approximately 500 000 eight byte reals. This reduction in real storage corresponds directly to a reduction in fill-in introduced by the LDLT factorisation and consequently to the reduction to the total time to solve the linear system.

Figure 2.4 displays plots of the CPU time taken to perform the three phases of the linear solve, for the same series of matrices, for the `MA41` solver. These results for `MA41` used an AMD ordering on the full structure as well as an AMD ordering on the block structure. Observe that though there are improvements in the speed for each of the phases, the improvements are not as dramatic as those achieved by the block version of `MA57`. This is because, though the block structure was exploited to find an ordering which kept all blocks together, the `ANALYSE` subroutine in `MA41` was permitted to freely interpret the expanded ordering computed by the block `ANALYSE` routine and so constructed an elimination tree which did not keep the blocks together. The result was that the degrees of freedom associated with each finite element were not restricted to the same node of the elimination tree. Indeed, it is well known that user supplied orderings produce poorer results than solver generated orderings, however, despite this, there was still an improvement in the amount of fill-in produced by the factorisation. Figure 2.5 displays the reduction in the amount of integer and real storage required for the factorisation.

The codes were also tested on the largest matrices that could be factorised by `MA57` from DG methods of order $p = 1, \cdots, 7$. As the order of polynomial approximation grows so does the size of the blocks as well as the density of the linear systems, leading to more challenging matrices. It is the size of the arrays required to store the factors which limits the maximum problem size that can be solved with the release version of `MA57`. As the code always uses 32-bit integers to index the array containing the factors, the solvers cannot deal with matrices whose factors contain more than $2^{31} - 1$ entries. Tables 2.2 and 2.3 contain data on the largest matrices that could be solved by both the regular and block versions of `MA57`. These were computed with the most efficient available ordering algorithm, the METIS implementation of the nested dissection algorithm. Observe that the solvers can deal with problems more than five times as large when a lower order finite element approximation is used compared to when a higher order approximation is used. This is because the additional density of the matrices in the higher order problems leads to much more fill-in during factorisation. This table also shows the percentage improvement in the time and memory required to solve these matrices when using block preprocessing in place of the regular `MA57` preprocessor. The best improvements are made on the largest matrices, generated from the DG

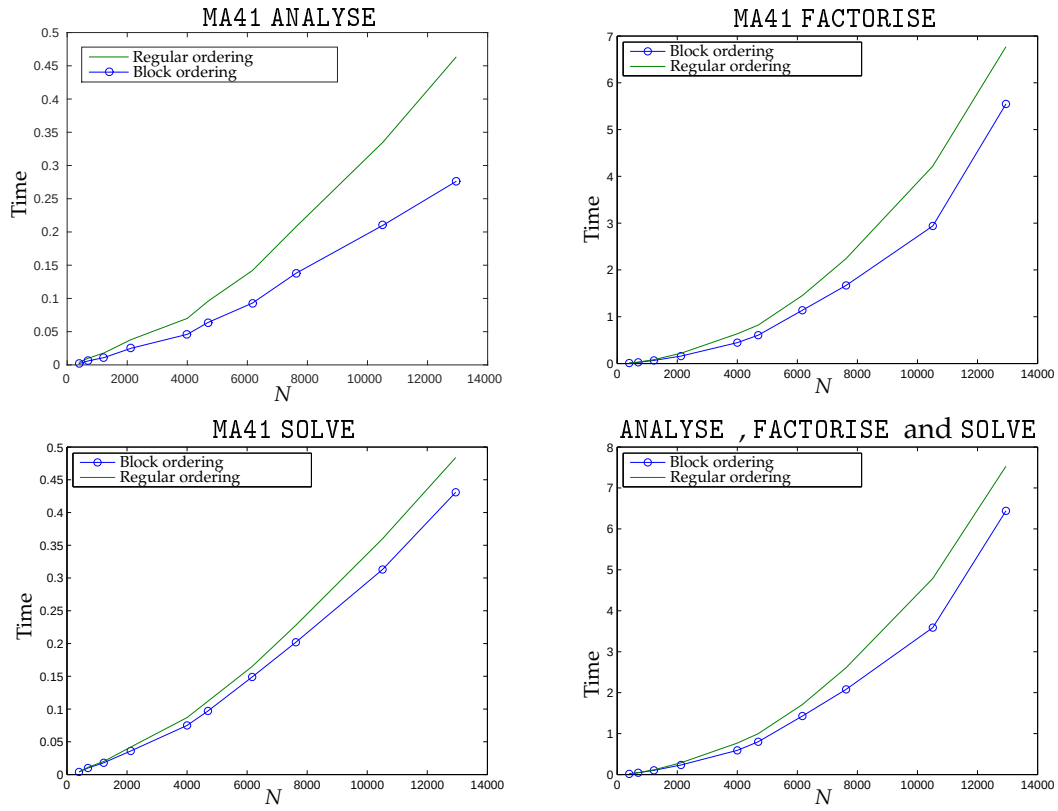


Figure 2.4: The time taken to for `MA41` to solve the series of block matrices from a DG method with regular and block preprocessing. The four plots show the time (in seconds) for the `ANALYSE` , `FACTORISE` , `SOLVE` and the total time to solve the linear system, respectively.

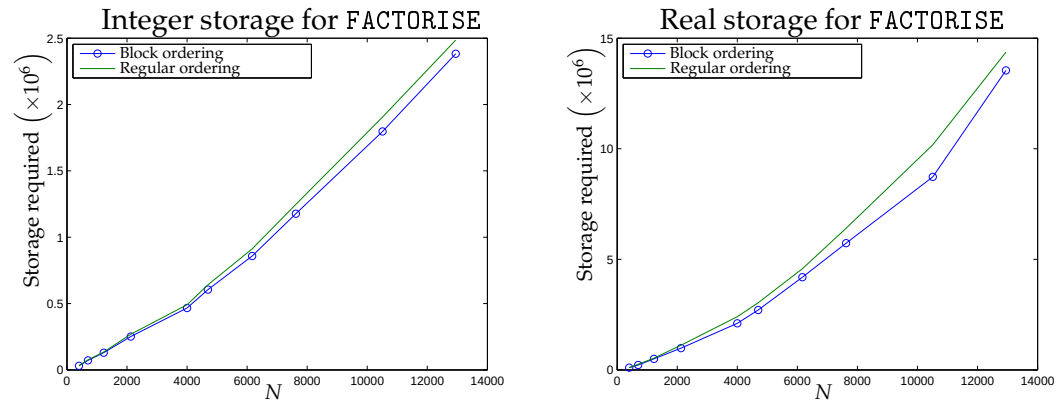


Figure 2.5: Plots displaying the reduction in the memory required to store the factors computed by `MA41` for a series of block matrices for the cases when the pivot order was computed using the block version of `MA57` and for when it was determined using the `MA41 ANALYSE` routine on the full matrix.

p	N	NE	Entries in factors (regular)	Entries in factors (block)	Total time (regular)	Total time (block)
1	5 008 644	42 979 904	1 184 466 559	969 943 114	7099.082	4361.738
2	4 004 001	86 664 261	1 625 511 039	1 321 414 443	10667.95	6183.452
3	2 509 056	94 256 589	1 539 811 502	1 313 303 424	11567.23	8554.046
4	2 002 225	122 906 040	1 584 765 379	1 405 605 175	12290.28	9669.956
5	1 512 900	132 366 556	1 542 669 556	1 390 159 818	13554.54	11203.12
6	1 002 001	121 072 385	1 173 537 254	1 112 921 516	10987.73	9609.322
7	1 000 000	154 360 936	1 570 107 115	1 361 385 760	17340.59	13632.63

Table 2.2: Information on the results found when testing the block `MA57` code until the length of the arrays containing the factors exceeded the 32-bit limit. Each line gives information on the largest matrix that could be solved by both the regular and block versions of `MA57` for a given order of approximation, p, in the finite element method.

methods with lowest polynomial order, though there is still a significant improvement for all polynomial orders.

In order to factorise matrices whose factors have more than $2^{31} - 1$ entries, code was written to enable `MA57` to interface with METIS 5, which has an option for the use of 64-bit indexing, see [81] for documentation. Then all of the Fortran codes were recompiled with a compiler flag to force all integers to be allocated 8 bytes of memory. These changes enabled the factorisation of matrices that were much more challenging than those that could be handled by the 32-bit code. A 64-bit integer can index an array containing up to $2^{63} - 1$ entries, therefore the new limit on problem size was imposed by the amount of virtual memory permitted by the computer operating system. Table 2.4 contains data on the memory required to factorise a matrix from a three dimensional finite element problem of size $N = 1\ 061\ 208$, an order of approximation of $p = 2$ and blocks of size 27×27. The matrix was obtained from a discretisation of equation (2.3.1), with the domain $\mathcal{D} = (0, 1)^3$, and the boundary condition $u = \hat{g}(\mathbf{r})$ on $\partial\mathcal{D}$.

The matrix was solved by regular and block versions of the software, with both the AMD ordering as well as the METIS 5 implementation of nested dissection. We note a substantial reduction in the amount of memory required for both real and integer storage when using the block version of either ordering. In particular, we observe a 35% reduction in the amount of memory required when moving from the regular version

		% improvement	
p	N	Memory	CPU Time
1	5 008 644	18.111	38.559
2	4 004 001	18.708	42.037
3	2 509 056	14.71	26.049
4	2 002 225	11.305	21.32
5	1 512 900	9.8861	17.348
6	1 002 001	5.1652	12.545
7	1 000 000	13.293	21.383

Table 2.3: The percentage improvement when using the block `ANALYSE` version of `MA57` in the memory and CPU time required to solve a series of matrices from DG methods with increasing order of polynomial approximation, p. These matrices were the largest that could be solved with 32 bit array indexing.

of nested dissection to the block version. Furthermore these data confirm the superiority of the nested dissection algorithm for computing orderings for three dimensional finite element problems, compared to AMD orderings. We see that the AMD ordering requires considerably less integer storage to compute its factorisation, however, this advantage is negated by a much greater quantity of fill-in compared to nested dissection.

These results hint at the improvements that can be made when using direct solvers to solve block sparse matrices with dense blocks, in particular those derived from DG finite element problems. Greater use of optimized `BLAS` subroutines may yield improvements. Also an improvement in the format used to store the block matrix could reduce the memory footprint significantly. As each block is dense, there is no need to store the coordinates of each entry within each block. The coordinates of the first entry and the size and shape of the block suffices to specify the position of all the remaining entries in that block, thus the full matrix structure could be stored as an array of the entries within the blocks together with a pointer to the start of each block, the size of each diagonal block and the pattern of the block structure. Though some improvement was realised by primarily considering the `ANALYSE` phase, possibly a more significant improvement could be made by tailoring the `FACTORISE` phase to take into account the block structure of the system. There is certainly an argument for this when we consider that Figures 2.2 and 2.4 show us how this stage in the solver dominates

Preprocessing algorithm	Storage	
	Integer	Real
AMD regular	22 082 190	8 942 957 652
AMD block	14 365 669	7 381 431 801
% improvement	34.945	17.461
METIS regular	93 711 047	7 734 840 693
METIS block	17 563 466	5 035 436 901
% improvement	81.258	34.899

Table 2.4: The amount of memory required to solve a large 3D matrix with $N = 1\,061\,208$ for four different methods of preprocessing. The preprocessing algorithms are the `MC47` implementation of the approximate minimum degree algorithm applied to the full matrix as well as the block structure, and the METIS implementation of the nested dissection algorithm applied to the full matrix as well as the block structure.

the total time. If this stage were to be tailored to utilise the block structures present in DG matrices, then improvements could be made by extracting further efficiencies from the optimized level 3 `BLAS` and `LAPACK` subroutines that the leading sparse direct solvers already employ. Problems may arise, however, if there were singular blocks on the diagonal, which could be possible for general block matrices, even if the full matrix is itself nonsingular. Consequently an efficient test for selecting a suitable block pivot would be essential.

CHAPTER 3

Discontinuous Galerkin Discretisation of the Neutron Transport Equation

In this chapter we introduce the discontinuous-Galerkin (DG) finite element discretisation of the neutron transport equation, which incorporates high order finite elements in both the spatial and the angular domain. To this end we outline both the underlying discrete scheme, together with potential solution strategies to solve the resulting linear system of equations. In particular, we present a preconditioner exploiting Tarjan's strongly connected components algorithm to find an ordered partition of the spatial elements for each element in the angular domain. This preconditioner employs a block preprocessor, developed in the previous chapter, in order to efficiently solve a reduced linear system on each of these strongly connected components. We then proceed to demonstrate the benefits of Krylov subspace methods when compared to simpler schemes for both the underlying source problem and the k_{eff}-criticality problem. The results from a parallel implementation of these solution algorithms are then presented. Finally, we numerically investigate the rates of convergence of the proposed finite element method by solving an artificially forced source problem. We conclude by providing results that demonstrate the benefits of employing higher order finite elements for the angular discretisation when compared to the more common discrete ordinates angular discretisation.

3.1 A high order DG method for neutron transport

In this section we outline a DG method for the monoenergetic neutron transport source problem. This discretisation may then readily be extended to the multigroup equations (see equation (1.5.9)) and and for the approximation of the k_{eff} and α neutron transport eigenvalue problems (see Section 1.3).

We begin by stating the steady state monoenergetic neutron transport source problem with isotropic scattering. Let the spatial domain $\mathcal{D}$ be an open Lipschitz domain in $\mathbb{R}^2$, with spatial variable $\mathbf{r} = (x, y)^\top \in \mathcal{D}$. Furthermore we let the angular domain consist of the surface of the unit sphere $S_2 \subset \mathbb{R}^3$ centred at the origin. The angular variable $\boldsymbol{\Omega}$ is parametrised by spherical polar coordinates:

$$\boldsymbol{\Omega} = \begin{pmatrix} \cos(\theta)\sin(\varphi) \\ \sin(\theta)\sin(\varphi) \\ \cos(\varphi) \end{pmatrix} \in \mathbb{R}^3, \tag{3.1.1}$$

where $\binom{\varphi}{\theta} \in (0, \pi) \times [0, 2\pi)$. The polar coordinate φ is measured from the north pole, $(0, 0, 1)^\top$ and the azimuthal coordinate is θ. We note that

$$\int_{S_2} d\boldsymbol{\Omega} = \int_0^{2\pi} \int_0^{\pi} \sin(\varphi)\, d\varphi\, d\theta. \tag{3.1.2}$$

The boundary of the spatial domain is denoted $\Gamma = \partial\mathcal{D}$ with unit outward normal $\mathbf{n}$. For a given direction in the angular domain $\boldsymbol{\Omega} \in S_2$ we denote the inflow and outflow parts of the spatial domain respectively by

$$\Gamma_- = \{\mathbf{r} \in \Gamma \mid \boldsymbol{\Omega} \cdot \mathbf{n} < 0\} \tag{3.1.3}$$

and

$$\Gamma_+ = \{\mathbf{r} \in \Gamma \mid \boldsymbol{\Omega} \cdot \mathbf{n} > 0\}. \tag{3.1.4}$$

By setting the number of groups to be one in equation (1.5.9) we obtain: find $\psi \in (\mathcal{D} \cup \Gamma_+) \times S_2 \to \mathbb{R}^2$ such that

$$\boldsymbol{\Omega} \cdot \nabla\psi(\mathbf{r}, \boldsymbol{\Omega}) + \Sigma_t(\mathbf{r})\psi(\mathbf{r}, \boldsymbol{\Omega}) = \frac{\Sigma_s(\mathbf{r}) + \nu\Sigma_f(\mathbf{r})}{4\pi}\phi(\mathbf{r}) + Q(\mathbf{r}, \boldsymbol{\Omega}), \tag{3.1.5}$$

when $(\mathbf{r}, \boldsymbol{\Omega}) \in \mathcal{D} \times S_2$ and

$$\psi(\mathbf{r}, \boldsymbol{\Omega}) = \hat{g}(\mathbf{r}, \boldsymbol{\Omega}), \tag{3.1.6}$$

when $(\mathbf{r}, \boldsymbol{\Omega}) \in \Gamma_- \times S_2$. The angular variable in the scattering cross section $\Sigma_s(\mathbf{r})$ has been omitted, thereby we assume that scattering occurs equally in all directions. Moreover we have introduced the scalar flux ϕ given by

$$\phi(\mathbf{r}) = \int_{S_2} \psi(\mathbf{r}, \boldsymbol{\Omega})\, d\boldsymbol{\Omega}. \tag{3.1.7}$$

To define the underlying space-angle finite element space, we proceed by first introducing meshes and corresponding finite element spaces over the spatial and angular domains respectively. The resulting space-angle mesh is defined based on employing a tensor product construction between these two meshes. The full space-angle finite element space is then defined as the tensor product between finite element spaces defined on each of the space and angle finite element meshes.

3.1.1 Spatial finite element space

Let $\mathcal{T}_S = \{\kappa_S\}$ be a shape regular subdivision of the spatial domain $\mathcal{D}$ into disjoint open elements κ_S such that

$$\overline{\mathcal{D}} = \bigcup_{\kappa_S \in \mathcal{T}_S} \overline{\kappa_S}. \tag{3.1.8}$$

We assume that each $\kappa_S \in \mathcal{T}_S$ is a smooth bijective image of a fixed reference element $\hat{\kappa}$, i.e.

$$\kappa_S = F_{\kappa_S}(\hat{\kappa}) \quad \forall \kappa_S \in \mathcal{T}_S, \tag{3.1.9}$$

where $F_{\kappa_S}(\cdot)$ is the smooth bijective mapping associated with the element κ_S. The reference element $\hat{\kappa}$ is taken to be either the reference square $\hat{S}$ or the reference triangle $\hat{T}$, i.e,

$$\hat{\kappa} \equiv \hat{S} = (-1,1)^2 \quad \text{or} \tag{3.1.10}$$

$$\hat{\kappa} \equiv \hat{T} = \left\{x, y \in \mathbb{R}^2 | \, x, y > 0 \, , \, x + y < 1\right\}, \tag{3.1.11}$$

respectively. Within this construction we admit 1-regular meshes; i.e., each face of $\kappa_S \in \mathcal{T}_S$ has at most one hanging node, typically located at the barycentre of the face. In order to define the finite element space in the spatial domain we first define the following polynomial spaces of order $p \geq 1$ on the reference element:

$$\mathcal{Q}_p = \text{Span}\left\{\hat{x}^\alpha \hat{y}^\beta \mid 0 \leq \alpha \leq p \, , \, 0 \leq \beta \leq p\right\}, \tag{3.1.12}$$

and

$$\mathcal{P}_p = \text{Span}\left\{\hat{x}^\alpha \hat{y}^\beta \mid \alpha \geq 0 \, , \, \beta \geq 0 \, , \, 0 \leq \alpha + \beta \leq p\right\}. \tag{3.1.13}$$

Writing $F_{\mathcal{T}_S} = \{F_{\kappa_S} \,|\, \kappa_S \in \mathcal{T}_S\}$, we define

$$S_S^p(\mathcal{T}_S, F_{\mathcal{T}_S}) = \{\psi_S \in L_2(\mathcal{D}) \quad | \quad \psi_S|_{\kappa_S} \circ F_{\kappa_S} \in \mathcal{R}\}, \tag{3.1.14}$$

where $\mathcal{R} = \mathcal{Q}_p$ if $\hat{\kappa} = F_{\kappa_S}^{-1}(\kappa_S)$ is a square and $\mathcal{R} = \mathcal{P}_p$ if $\hat{\kappa} = F_{\kappa_S}^{-1}(\kappa_S)$ is a triangle. Thereby $S_S^p(\mathcal{T}_S, F_{\mathcal{T}_S})$ consists of discontinuous piecewise polynomials of degree $p \geq 0$. Note that we make no requirement that the finite element space be continuous at the element boundaries, instead within the DG setting we enforce inter element continuity weakly through the use of an appropriate numerical flux function.

3.1.2 Angular finite element space

For the angular domain we consider a mesh over the parameter space $(0, \pi) \times [0, 2\pi)$ as opposed to the true angular domain S_2; thereby both the spatial and angular meshes comprise partitions of a bounded subset of $\mathbb{R}^2$ into spatial and angular elements κ_S and κ_A, respectively. We construct the angular mesh similarly to the spatial one with $(0, \pi) \times [0, 2\pi)$ in place of $\mathcal{D}$ and the map from the canonical square to each angular element κ_A given by $F_{\kappa_A}(\cdot)$. Thus we have

$$\mathcal{T}_A = \left\{ \kappa_A \,|\, \kappa_A = F_{\kappa_A}(\hat{S}) \right\}, \tag{3.1.15}$$

where

$$\bigcup_{\kappa_A \in \mathcal{T}_A} \overline{\kappa_A} = [0, \pi] \times [0, 2\pi]. \tag{3.1.16}$$

As before, $\mathcal{T}_A$ is a shape regular mesh consisting of potentially 1-irregular quadrilateral elements. Given $q \geq 0$ we again define

$$\mathcal{Q}_q = \text{Span}\left\{ \hat{\varphi}^\alpha \hat{\theta}^\beta \,|\, 0 \leq \alpha \leq q\,,\, 0 \leq \beta \leq q \right\}, \tag{3.1.17}$$

and the set of element maps $F_{\mathcal{T}_A} = \{F_{\kappa_A} \,|\, \kappa_A \in \mathcal{T}_A\}$. Then the angular finite element space of order q is given by

$$S_A^q(\mathcal{T}_A, F_{\mathcal{T}_A}) = \left\{ \psi_A \in L_2(S_2) \quad | \quad \psi_A|_{\kappa_A} \circ F_{\kappa_A} \in \mathcal{Q}_q \right\}, \tag{3.1.18}$$

which is the set of all square integrable functions that belong to the polynomial tensor product space of order q when they are restricted to a single angular element κ_A and projected onto the canonical element. As for the spatial finite element space we impose no continuity condition between angular elements, however unlike the spatial problem we will not later impose numerical flux conditions between the elements. This is justified on physical grounds. As described in Chapter 1 there is no interaction between the neutron flux in different directions and the modelling of any change in angle of

the neutrons is incorporated into the scattering cross-section term. This uncoupling of the angular elements will facilitate the development of a preconditioner for the Krylov methods considered later in this chapter, as well as enabling the implementation of the solvers in parallel.

3.1.3 Space-angle finite element space

To define a finite element space on the space-angle domain $\mathcal{D} \times S_2$ we first introduce the space-angle mesh $\mathcal{T}$ defined as the tensor product of the spatial and angular meshes, i.e.,

$$\mathcal{T} = \mathcal{T}_S \times \mathcal{T}_A = \{\kappa_S \times \kappa_A \,|\, \kappa_S \in \mathcal{T}_S \,,\, \kappa_A \in \mathcal{T}_A\}\,. \tag{3.1.19}$$

Then the corresponding finite element space is given by

$$\begin{aligned} \mathcal{V}_h^{p,q} = \{\psi_h \in L_2(\mathcal{D} \times S_2) \mid \psi_h = \psi_S \times \psi_A \,, \\ \psi_S \in S_S^p(\mathcal{T}_S, F_{\mathcal{T}_S}) \,,\, \psi_A \in S_A^q(\mathcal{T}_A, F_{\mathcal{T}_A})\}. \end{aligned} \tag{3.1.20}$$

Each function in $\mathcal{V}_h^{p,q}$ corresponds to a tensor product between a member of the angular finite element space and the spatial finite element space. From an implementational point of view this means that every degree of freedom in the angular finite element space will have a full set of spatial degrees of freedom associated with it, and, analogously, every degree of freedom in the spatial finite element space will have a full set of angular degrees of freedom associated with it.

3.1.4 Finite element discretisation

Now that we have defined a finite element space over the full computational domain we may proceed to derive the discretisation of the neutron transport equation. We begin by writing equation (3.1.5) in weak form. To this end, we multiply by an arbitrary smooth test function v and integrate over a single space-angle element $\kappa_S \times \kappa_A \in \mathcal{T}$,

$$\begin{aligned} &\int_{\kappa_A}\int_{\kappa_S} \boldsymbol{\Omega}\cdot\nabla\psi(\mathbf{r},\boldsymbol{\Omega})v(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega} + \int_{\kappa_A}\int_{\kappa_S} \Sigma_t(\mathbf{r})\psi(\mathbf{r},\boldsymbol{\Omega})v(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega} \\ &\quad = \int_{\kappa_A}\int_{\kappa_S} \frac{\Sigma_s(\mathbf{r}) + \nu\Sigma_f(\mathbf{r})}{4\pi}\phi(\mathbf{r})v(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega} + \int_{\kappa_A}\int_{\kappa_S} Q(\mathbf{r},\boldsymbol{\Omega})v(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega}. \end{aligned} \tag{3.1.21}$$

We assume that $v|_{\kappa_S} \in H^1(\kappa_S)$ for each $\kappa_S \in \mathcal{T}_S$ and we denote the spatial interior trace of v on $\kappa_S \times \kappa_A$ by v^+ and the spatial exterior trace of v on $(\partial\kappa_S \setminus \Gamma) \times \kappa_A$ by v^-. Then

we integrate the spatial part of the advection term by parts to get,

$$\begin{aligned}\int_{\kappa_A}\int_{\kappa_S} -\psi(\mathbf{r},\boldsymbol{\Omega})\boldsymbol{\Omega}\cdot\nabla v(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega} + \int_{\kappa_A}\int_{\partial\kappa_S}(\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S})\psi^+(\mathbf{r},\boldsymbol{\Omega})v^+(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega}\\ + \int_{\kappa_A}\int_{\kappa_S}\Sigma_t(\mathbf{r})\psi(\mathbf{r},\boldsymbol{\Omega})v(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega} = \int_{\kappa_A}\int_{\kappa_S}\frac{\Sigma_s(\mathbf{r})+\nu\Sigma_f(\mathbf{r})}{4\pi}\phi(\mathbf{r})v(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega}\\ + \int_{\kappa_A}\int_{\kappa_S}Q(\mathbf{r},\boldsymbol{\Omega})v(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega},\end{aligned} \tag{3.1.22}$$

where $\mathbf{n}_{\kappa_S}$ is the unit outward normal to κ_S and s parametrises the spatial element boundary. To obtain our DG finite element discretisation we replace ψ by the DG solution $\psi_h \in \mathcal{V}_h^{p,q}$ and v by $v_h \in \mathcal{V}_h^{p,q}$ and sum equation (3.1.22) over all spatial elements. As ψ_h is discontinuous over internal spatial element boundaries we replace $(\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S})\psi^+$ with a numerical flux function $\mathcal{H}(\psi_h^+,\psi_h^-,\mathbf{n}_{\kappa_S},\boldsymbol{\Omega})$ whenever $\partial\kappa_S$ is not on Γ. Thus we obtain the DG finite element discretisation of the neutron transport equation as follows: find $\psi_h \in \mathcal{V}_h^{p,q}$ such that

$$\begin{aligned}&\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S} -\psi_h(\mathbf{r},\boldsymbol{\Omega})\boldsymbol{\Omega}\cdot\nabla v_h(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega}\\ &\quad+\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\partial\kappa_S\setminus\Gamma}\mathcal{H}(\psi_h^+(\mathbf{r},\boldsymbol{\Omega}),\psi_h^-(\mathbf{r},\boldsymbol{\Omega}),\mathbf{n}_{\kappa_S},\boldsymbol{\Omega})v_h^+(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega}\\ &\quad+\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\partial\kappa_S\cap\Gamma_+}(\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S})\psi_h(\mathbf{r},\boldsymbol{\Omega})v_h(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega}\\ &\quad+\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S}\Sigma_t(\mathbf{r})\psi_h(\mathbf{r},\boldsymbol{\Omega})v_h(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega}\\ &=\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S}\frac{\Sigma_s(\mathbf{r})+\nu\Sigma_f(\mathbf{r})}{4\pi}\phi_h(\mathbf{r})v_h(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega}\\ &\quad-\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\partial\kappa_S\cap\Gamma_-}(\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S})\hat{g}(\mathbf{r},\boldsymbol{\Omega})v_h(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega}\\ &\quad+\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S}Q(\mathbf{r},\boldsymbol{\Omega})v_h(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega},\end{aligned} \tag{3.1.23}$$

for all $v_h \in \mathcal{V}_h^{p,q}$. Here we have included the Dirichlet boundary condition at Γ_- and an outflow term at Γ_+. The choice of the numerical flux is independent of the finite element space and can be any two-point monotone Lipschitz function that is consistent and conservative (see [82]). A numerical flux, $\mathcal{H}(\cdot,\cdot,\cdot,\cdot)$, is consistent if for each $\kappa_S \in \mathcal{T}_S$,

$$\mathcal{H}(v,v,\mathbf{n}_{\kappa_S},\boldsymbol{\Omega})|_{\partial\kappa_S} = (\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S})v. \tag{3.1.24}$$

It is conservative if for each pair of elements $\kappa_S, \kappa_S{}' \in \mathcal{T}_S$ where $\kappa_S \cap \kappa_S{}' \neq \emptyset$,

$$\mathcal{H}(v,w,\mathbf{n}_{\kappa_S},\boldsymbol{\Omega})|_{\partial\kappa_S} = -\mathcal{H}(w,v,-\mathbf{n}_{\kappa_S},\boldsymbol{\Omega})|_{\partial\kappa_S}, \tag{3.1.25}$$

where we note that $\mathbf{n}_{\kappa_S} = -\mathbf{n}_{\kappa_{S'}}$. For the present implementation we use the local Lax-Friedrichs flux which is defined by,

$$\mathcal{H}(\psi^+, \psi^-, \mathbf{n}_{\kappa_S}, \boldsymbol{\Omega}) = \frac{1}{2}\left(\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S} + |\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S}|\right)\psi^+ + \frac{1}{2}\left(\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S} - |\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S}|\right)\psi^-. \tag{3.1.26}$$

3.1.5 Relationship with the discrete ordinates method

When the order of approximation on each angular element is $q = 0$, the angular part of the above DG finite element method is very similar to the discrete ordinates method. However, the proposed DG scheme is not precisely the same due to the fact that we permit irregular meshes and curved boundaries in the spatial domain, as well as boundary conditions and a forcing function which are not constant with respect to angle. This can be seen by considering what happens to the various terms in (3.1.23) when a constant approximation order is employed in angle, i.e. when $q = 0$. In the terms containing the scattering and fission cross sections the scalar flux may be written as the weighted sum over N_O discrete ordinate directions, where N_O is the number of elements in the angular mesh, i.e.,

$$\begin{aligned}\phi_h(\mathbf{r}) &= \int_{S_2} \psi_h(\mathbf{r},\boldsymbol{\Omega})\, d\boldsymbol{\Omega} && (3.1.27)\\ &= \sum_{\kappa_A\in\mathcal{T}_A}\int_{\kappa_A} \psi_h(\mathbf{r},\boldsymbol{\Omega})\, d\boldsymbol{\Omega} && (3.1.28)\\ &= \sum_{i=1}^{N_O} w_i\psi_h(\mathbf{r},\boldsymbol{\Omega}_i), && (3.1.29)\end{aligned}$$

where the ordinate directions $\boldsymbol{\Omega}_i$ are the centroids of each κ_A on the surface of the sphere and the discrete ordinates weighting w_i is equal to the area of κ_A. The integration over the angular elements may be treated similarly in each of the terms in equation (3.1.23) that do not involve an integration over a spatial element boundary $\partial\kappa_S$ or the source term $Q(\mathbf{r},\boldsymbol{\Omega})$. For example, if i indexes the discrete ordinate direction and weighting associated with the angular element κ_A, we may rewrite the first term as

$$\begin{aligned}&\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S} -\psi_h(\mathbf{r},\boldsymbol{\Omega})\boldsymbol{\Omega}\cdot\nabla v_h(\mathbf{r},\boldsymbol{\Omega})\, d\mathbf{r}\, d\boldsymbol{\Omega}\\ &\qquad = \sum_{i=1}^{N_O} w_i \sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_S} -\psi_h(\mathbf{r},\boldsymbol{\Omega}_i)\boldsymbol{\Omega}_i\cdot\nabla v_h(\mathbf{r},\boldsymbol{\Omega}_i)\, d\mathbf{r}.\end{aligned} \tag{3.1.30}$$

If we require that $Q(\mathbf{r}, \boldsymbol{\Omega})$ is constant with respect to angle on each κ_A then we may write the final term as

$$\begin{aligned}\sum_{\kappa_A \in \mathcal{T}_A} \sum_{\kappa_S \in \mathcal{T}_S} \int_{\kappa_A} \int_{\kappa_S} Q(\mathbf{r}, \boldsymbol{\Omega}) v_h(\mathbf{r}, \boldsymbol{\Omega}) \, d\mathbf{r} \, d\boldsymbol{\Omega} \\ = \sum_{i=1}^{N_O} w_i \sum_{\kappa_S \in \mathcal{T}_S} \int_{\kappa_S} Q(\mathbf{r}, \boldsymbol{\Omega}_i) v_h(\mathbf{r}, \boldsymbol{\Omega}_i) \, d\mathbf{r}.\end{aligned} \tag{3.1.31}$$

If we further require that $\hat{g}(\mathbf{r}, \boldsymbol{\Omega})$ is constant with respect to angle on each κ_A and that there does not exist a pair (κ_S, κ_A) for which both $\partial\kappa_S \cap \Gamma_+ \neq \emptyset$ and $\partial\kappa_S \cap \Gamma_- \neq \emptyset$ for all $\boldsymbol{\Omega} \in \kappa_A$, then we may also write the terms involving the boundary of the spatial domain in terms of a sum over weighted discrete ordinate directions, i.e.,

$$\begin{aligned}\sum_{\kappa_A \in \mathcal{T}_A} \sum_{\kappa_S \in \mathcal{T}_S} \int_{\kappa_A} \int_{\partial\kappa_S \cap \Gamma_+} (\boldsymbol{\Omega} \cdot \mathbf{n}_{\kappa_S}) \psi_h(\mathbf{r}, \boldsymbol{\Omega}) v_h(\mathbf{r}, \boldsymbol{\Omega}) \, ds \, d\boldsymbol{\Omega} \\ = \sum_{i=1}^{N_O} w_i \sum_{\kappa_S \in \mathcal{T}_S} \int_{\partial\kappa_S \cap \Gamma_+} (\boldsymbol{\Omega}_i \cdot \mathbf{n}_{\kappa_S}) \psi_h(\mathbf{r}, \boldsymbol{\Omega}_i) v_h(\mathbf{r}, \boldsymbol{\Omega}_i) \, ds\end{aligned} \tag{3.1.32}$$

and

$$\begin{aligned}\sum_{\kappa_A \in \mathcal{T}_A} \sum_{\kappa_S \in \mathcal{T}_S} \int_{\kappa_A} \int_{\partial\kappa_S \cap \Gamma_-} (\boldsymbol{\Omega} \cdot \mathbf{n}_{\kappa_S}) \hat{g}(\mathbf{r}, \boldsymbol{\Omega}) v_h(\mathbf{r}, \boldsymbol{\Omega}) \, ds \, d\boldsymbol{\Omega} \\ = \sum_{i=1}^{N_O} w_i \sum_{\kappa_S \in \mathcal{T}_S} \int_{\partial\kappa_S \cap \Gamma_-} (\boldsymbol{\Omega}_i \cdot \mathbf{n}_{\kappa_S}) \hat{g}(\mathbf{r}, \boldsymbol{\Omega}_i) v_h(\mathbf{r}, \boldsymbol{\Omega}_i) \, ds.\end{aligned} \tag{3.1.33}$$

Finally, if we require that the spatial and angular meshes are chosen so that there does not exist an interface between two spatial elements κ_S and $\kappa_S{}'$ and two values of the angular variable on the same angular element $\boldsymbol{\Omega}, \boldsymbol{\Omega}' \in \kappa_A$ such that $\boldsymbol{\Omega} \cdot \mathbf{n}_{\kappa_S} > 0$ and $\boldsymbol{\Omega}' \cdot \mathbf{n}_{\kappa_S} < 0$, then we may write the numerical flux term as

$$\begin{aligned}\sum_{\kappa_A \in \mathcal{T}_A} \sum_{\kappa_S \in \mathcal{T}_S} \int_{\kappa_A} \int_{\partial\kappa_S \setminus \Gamma} \mathcal{H}(\psi_h^+(\mathbf{r}, \boldsymbol{\Omega}), \psi_h^-(\mathbf{r}, \boldsymbol{\Omega}), \mathbf{n}_{\kappa_S}, \boldsymbol{\Omega}) v_h^+(\mathbf{r}, \boldsymbol{\Omega}) \, ds \, d\boldsymbol{\Omega} \\ = \sum_{i=1}^{N_O} w_i \sum_{\kappa_S \in \mathcal{T}_S} \int_{\partial\kappa_S \setminus \Gamma} \mathcal{H}(\psi_h^+(\mathbf{r}, \boldsymbol{\Omega}_i), \psi_h^-(\mathbf{r}, \boldsymbol{\Omega}_i), \mathbf{n}_{\kappa_S}, \boldsymbol{\Omega}_i) v_h^+(\mathbf{r}, \boldsymbol{\Omega}_i) \, ds.\end{aligned} \tag{3.1.34}$$

We can then rewrite the space-angle finite element discretisation as a variational problem for the spatial solution on the ith discrete ordinate. Indeed writing the finite element space with piecewise constants in angle restricted to a single angular element by $\mathcal{V}_h^{p,0}\Big|_{\kappa_A}$ and the solution in the ith ordinate by $\psi_{h,i}(\mathbf{r}) \equiv \psi_h(\mathbf{r}, \boldsymbol{\Omega}_i)$, then we have the

following problem for each discrete ordinate: find $\psi_{h,i} \in \mathcal{V}_h^{p,0}\Big|_{\kappa_A}$ such that

$$\begin{aligned}
&\sum_{\kappa_S \in \mathcal{T}_S} \int_{\kappa_S} -\psi_{h,i}(\mathbf{r})\boldsymbol{\Omega}_i \cdot \nabla v_h(\mathbf{r})\, d\mathbf{r} + \sum_{\kappa_S \in \mathcal{T}_S} \int_{\partial\kappa_S \cap \Gamma_+} (\boldsymbol{\Omega}_i \cdot \mathbf{n}_{\kappa_S}) \psi_{h,i}(\mathbf{r}) v_h(\mathbf{r})\, ds \\
&+ \sum_{\kappa_S \in \mathcal{T}_S} \int_{\partial\kappa_S \setminus \Gamma} \mathcal{H}(\psi_{h,i}^+(\mathbf{r}), \psi_{h,i}^-(\mathbf{r}), \mathbf{n}_{\kappa_S}, \boldsymbol{\Omega}_i) v_h^+(\mathbf{r})\, ds + \sum_{\kappa_S \in \mathcal{T}_S} \int_{\kappa_S} \Sigma_t(\mathbf{r}) \psi_{h,i}(\mathbf{r}) v_h(\mathbf{r})\, d\mathbf{r} \\
&\qquad = \sum_{\kappa_S \in \mathcal{T}_S} \int_{\kappa_S} \frac{\Sigma_s(\mathbf{r}) + \nu\Sigma_f(\mathbf{r})}{4\pi} \phi_h(\mathbf{r}) v_h(\mathbf{r})\, d\mathbf{r} - \sum_{\kappa_S \in \mathcal{T}_S} \int_{\partial\kappa_S \cap \Gamma_-} (\boldsymbol{\Omega}_i \cdot \mathbf{n}_{\kappa_S}) \hat{g}(\mathbf{r}, \boldsymbol{\Omega}_i) v_h(\mathbf{r})\, ds \\
&\qquad + \sum_{\kappa_S \in \mathcal{T}_S} \int_{\kappa_S} Q(\mathbf{r}, \boldsymbol{\Omega}_i) v_h(\mathbf{r})\, d\mathbf{r}
\end{aligned} \tag{3.1.35}$$

for all $v_h \in \mathcal{V}_h^{p,0}\Big|_{\kappa_A}$. These equations are coupled by the sum over the ordinate directions implicit in $\phi(\mathbf{r})$. Here we have divided all terms by the constant w_i.

We have stated several assumptions for the finite element method with piecewise constants in angle to be the same as the discrete ordinates method. In practice however it is found that even when these requirements are not fully satisfied and general meshes are used in space and angle then the numerical results from these two methods are almost identical.

3.2 Developing a solver

Now that we have written down a discrete version of the neutron transport equation we consider how to solve the resulting system of linear equations in order to compute the numerical solution. In this section we begin by writing the neutron transport source problem and k_{eff}-eigenvalue problem in terms of matrices that represent the transport, scattering, fission and source terms, we then proceed to consider how the application of the inverse of the transport matrix may be computed using Tarjan's strongly connected components algorithm. We then compare two possible solution algorithms for each of the source and eigenvalue problems and give results from a parallel implementation of these strategies.

3.2.1 The discrete neutron transport equation in matrix form

We employ the finite element discretisation outlined in the previous section to write down the neutron transport source and eigenvalue problems in terms of three matri-

ces, $\mathbf{T}$, $\mathbf{S}$ and $\mathbf{F}$, as well as a vector representing the discretised source term $\mathbf{q}$. The three matrices will each be unassembled and will be implemented as the application of matrix vector product on a vector of length N, where N is the number of degrees of freedom in the finite element space $\mathcal{V}_h^{p,q}$. An algorithm for the application of the inverse of the transport matrix, $\mathbf{T}^{-1}$, will also be developed.

We define the polynomial basis functions for the finite element spaces on each of the canonical elements $\hat{S}$ and $\hat{T}$. For triangular elements we use Dubiner basis functions [83], which we denote $\xi_{\alpha,\beta}^{\hat{T}}$. These basis functions are constructed by taking a weighted product of Jacobi polynomials as

$$\xi_{\alpha,\beta}^{\hat{T}}(x,y) = P_\alpha^{0,0}\left(\frac{2x}{1-y} - 1\right)(1-y)^\alpha P_\beta^{2\alpha+1,0}(2y-1)\,. \tag{3.2.1}$$

Then the polynomial space $\mathcal{P}_p$ may be written as

$$\mathcal{P}_p = \text{Span}\left\{\xi_{\alpha,\beta}^{\hat{T}} \mid \alpha,\beta \geq 0\,,\ \alpha+\beta \leq p\right\}. \tag{3.2.2}$$

For square elements the basis functions, $\xi_{\alpha,\beta}^{\hat{S}}$, are constructed on $\hat{S}$ as the product of Legendre polynomials in either direction, i.e.,

$$\xi_{\alpha,\beta}^{\hat{S}}(x,y) = P_\alpha^{0,0}(x)\,P_\beta^{0,0}(y)\,. \tag{3.2.3}$$

Then the polynomial space $\mathcal{Q}_p$ may be written as

$$\mathcal{Q}_p = \text{Span}\left\{\xi_{\alpha,\beta}^{\hat{S}} \mid 0 \leq \alpha \leq p\,,\ 0 \leq \beta \leq p\right\}. \tag{3.2.4}$$

The angular polynomial space $\mathcal{Q}_q$ is constructed similarly. These sets of basis functions for $\mathcal{Q}_p$, $\mathcal{P}_p$ and $\mathcal{Q}_q$ together with the definitions of $S_S^p(\mathcal{T}_S, F_{\mathcal{T}_S})$, $S_A^q(\mathcal{T}_A, F_{\mathcal{T}_A})$ and $\mathcal{V}_h^{p,q}$ induce a set of basis functions, $\{\xi_j\}$, for the space-angle finite element space

$$\mathcal{V}_h^{p,q} = \text{Span}\{\xi_j\} \tag{3.2.5}$$

where $j = 1, \cdots, N$. Then the finite element solution $\psi_h \in \mathcal{V}_h^{p,q}$ is determined by a vector $\boldsymbol{\psi} \in \mathbb{R}^N$, where

$$\psi_h(\mathbf{r},\boldsymbol{\Omega}) = \sum_{j=1}^{N} \boldsymbol{\psi}[j]\xi_j(\mathbf{r},\boldsymbol{\Omega}). \tag{3.2.6}$$

The solution to the discrete neutron transport equation is the unique function $\psi_h \in \mathcal{V}_h^{p,q}$ for which equation (3.1.23) holds for all $v_h \in \mathcal{V}_h^{p,q}$. Equivalently we seek the vector $\boldsymbol{\psi}$ for which this equation holds for all test functions $\xi_j \in \mathcal{V}_h^{p,q}$. This enables the advection,

absorption, numerical flux and boundary terms to be written as an $N \times N$ sparse matrix $\mathbf{T}$ in terms of the functions ξ_i and ξ_j. $\mathbf{T}[i,j]$ is nonzero only when i and j index basis functions defined on the same space-angle element $\kappa_S \times \kappa_A$ or the same angular element κ_A and neighbouring spatial elements κ_S and $\kappa_S{}'$, where $\kappa_S \cap \kappa_S{}' \neq \emptyset$. If ξ_i is defined on κ_S then $\xi_i^+(\mathbf{r},\boldsymbol{\Omega})$ denotes the interior trace of ξ_i at $\mathbf{r} \in \partial\kappa_S$. Then when i and j index the same $\kappa_S \times \kappa_A$

$$\begin{aligned}\mathbf{T}[i,j] = &- \sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S} \xi_j(\mathbf{r},\boldsymbol{\Omega})\boldsymbol{\Omega}\cdot\nabla\xi_i(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega} \\ &+ \sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\partial\kappa_S\cap\Gamma_+} (\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S})\xi_j(\mathbf{r},\boldsymbol{\Omega})\xi_i(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega} \\ &+ \sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\partial\kappa_S\setminus\Gamma} \frac{1}{2}(\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S}+|\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S}|)\xi_j^+(\mathbf{r},\boldsymbol{\Omega})\xi_i^+(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega} \\ &+ \sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S} \Sigma_t(\mathbf{r})\xi_j(\mathbf{r},\boldsymbol{\Omega})\xi_i(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega} \\ &+ \sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\partial\kappa_S\cap\Gamma_-} (\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S})\hat{g}(\mathbf{r},\boldsymbol{\Omega})\xi_i(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega}.\end{aligned} \tag{3.2.7}$$

If i and j index the same angular element and neighbouring spatial elements then

$$\mathbf{T}[i,j] = \sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\partial\kappa_S\setminus\Gamma} \frac{1}{2}(\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S}-|\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S}|)\xi_j^+(\mathbf{r},\boldsymbol{\Omega})\xi_i^+(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega}. \tag{3.2.8}$$

$\mathbf{T}$ is a block diagonal matrix with a diagonal block corresponding to each angular element. This facilitates the parallel implementation of the action of $\mathbf{T}$ and $\mathbf{T}^{-1}$ on a vector. An OpenMP implementation of these operations will be presented later in this chapter.

We obtain matrices corresponding to the fission and scattering terms in the neutron transport equation in a similar manner to the transport matrix by selecting the appropriate terms in equation (3.1.23) and then re-writing the numerical solution as in equation (3.2.6). The $[i,j]$th entry of the matrices $\mathbf{S}$ and $\mathbf{F}$ are non zero only when i and j index basis functions on the same spatial element. Then the entries of the matrix associated with the scattering term are given by

$$\mathbf{S}[i,j] = \sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S} \frac{\Sigma_s(\mathbf{r})}{4\pi}\left(\sum_{\kappa_A{}'\in\mathcal{T}_A}\int_{\kappa_A{}'}\xi_j(\mathbf{r},\boldsymbol{\Omega}')\,d\boldsymbol{\Omega}'\right)\xi_i(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega}, \tag{3.2.9}$$

and the entries of the matrix associated with the fission term are given by

$$\mathbf{F}[i,j] = \sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S} \frac{\nu\Sigma_f(\mathbf{r})}{4\pi}\left(\sum_{\kappa_A{}'\in\mathcal{T}_A}\int_{\kappa_A{}'}\xi_j(\mathbf{r},\boldsymbol{\Omega}')\,d\boldsymbol{\Omega}'\right)\xi_i(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega}. \tag{3.2.10}$$

For the present case with a monoenergetic energy approximation and isotropic scattering both $\mathbf{S}$ and $\mathbf{F}$ have the same structure which is sparse, symmetric and singular.

These matrices couple together the angular elements and their action is fast to compute compared to the action of $\mathbf{T}$ and $\mathbf{T}^{-1}$. This is because, as the scattering cross section is isotropic, the spatial and angular basis functions may be separated prior to the computation of the integrals over each $\kappa_S \times \kappa_A$. The value of the integral over the full space-angle element may then be obtained by taking a tensor product afterwards. The spatial and angular variables may similarly be separated in the matrix associated with the fission term. For $\mathbf{T}$ however, the spatial and angular variables may not be separated in this fashion.

Finally, the source term may be written as a vector $\mathbf{q}$, with the ith entry given by

$$\mathbf{q}[i] = \sum_{\kappa_A \in \mathcal{T}_A} \sum_{\kappa_S \in \mathcal{T}_S} \int_{\kappa_A} \int_{\kappa_S} Q(\mathbf{r}, \boldsymbol{\Omega}) \xi_i(\mathbf{r}, \boldsymbol{\Omega}) \, d\mathbf{r} \, d\boldsymbol{\Omega}. \tag{3.2.11}$$

The matrix definitions for $\mathbf{T}$, $\mathbf{F}$ and $\mathbf{S}$, as well as the vector $\mathbf{q}$, describe the discretised form of the monoenergetic neutron transport equation with isotropic scattering, which is the simplest form of the neutron transport equation that we consider. The finite element discretisation of the neutron transport problem with anisotropic scattering, as well as the multigroup equations can be written in terms of matrices for the transport, scattering and fission terms, in an analogous manner. Whilst consideration of anisotropic scattering is not considered, we will solve problems with a multigroup discretisation in the energy spectrum. When a multigroup approximation is employed the transport matrix $\mathbf{T}$ retains the same structure and properties on each of the energy groups, with no coupling between energy groups. Under a multigroup approximation, the scattering and fission matrices couple together the energy group, however only the fission matrix retains its symmetry. The presence of the scattering cross section,

$$\Sigma_{s,g' \to g}(\mathbf{r}) \neq \Sigma_{s,g \to g'}(\mathbf{r}), \tag{3.2.12}$$

in the multigroup equations however means that the scattering matrix $\mathbf{S}$ will no longer be symmetric.

Utilising these matrix representations of the discrete equations we may now write the finite element discretisation of the neutron transport source problem more concisely as: find $\boldsymbol{\psi} \in \mathbb{R}^N$ such that

$$(\mathbf{T} - \mathbf{F} - \mathbf{S})\boldsymbol{\psi} = \mathbf{q}. \tag{3.2.13}$$

The discrete form of the k_{eff}-eigenvalue equation may be written as: find $\boldsymbol{\psi} \in \mathbb{R}^N$ and $k_{\text{eff},h} \in \mathbb{R}$ such that

$$(\mathbf{T} - \mathbf{S})\boldsymbol{\psi} = \frac{1}{k_{\text{eff},h}} \mathbf{F}\boldsymbol{\psi}. \tag{3.2.14}$$

Though we will not be solving the time decay eigenvalue problem, we include its discrete representation for completeness: find $\boldsymbol{\psi} \in \mathbb{R}^N$ and $\alpha \in \mathbb{R}$ such that

$$\alpha\boldsymbol{\psi} = (\mathbf{S} + \mathbf{F} - \mathbf{T})\boldsymbol{\psi}. \tag{3.2.15}$$

3.2.2 Inverting the transport matrix

In this thesis we consider two algorithms for the solution of the neutron transport source problem and two algorithms for the solution to the k_{eff}-eigenvalue problem. Each of these algorithms uses four different types of 'level 2' sparse matrix operation, as well as various 'level 1', vector-vector operations. In practice it is seen that the four 'level 2' operations dominate the computational time of each of the solution algorithms, with the application of $\mathbf{T}$ requiring more time than the application of $\mathbf{S}$ or $\mathbf{F}$, and the application of $\mathbf{T}^{-1}$ requires most time of all by a considerable margin. In this section we consider an algorithm for the application of $\mathbf{T}^{-1}$ based on the sweeping technique described in the literature for the S_N method for neutron transport and radiative transport.

The sweeping technique for inverting a matrix relies on finding an ordering of the spatial elements so that the matrix may be written in block triangular form followed by using a block forward substitution procedure to assemble and solve for each element without ever assembling the full matrix. Physically this ordering follows the characteristics of the differential equation being solved; for the present case the direction is determined by the angular variable. For the case when the angular part of our finite element method reduces to a discrete ordinates approximation the ordering is easily computed using a simple topological sorting algorithm. However, when higher order finite elements are used in angle we find that groups of spatial elements become coupled together and a more sophisticated algorithm is needed to identify groups of elements that need to be solved together. We begin by considering the simplest case and showing that a topological ordering will exist and give an algorithm to find it. We then proceed to the more challenging case and describe Tarjan's strongly connected components algorithm for identifying the required irreducible ordering for the matrix.

Before proceeding, we first revisit some graph theory. A directed graph $\tilde{G} = (\tilde{V}, \tilde{E})$, is composed of a set of vertices $\tilde{V}$, and a set of ordered pairs of vertices called edges,

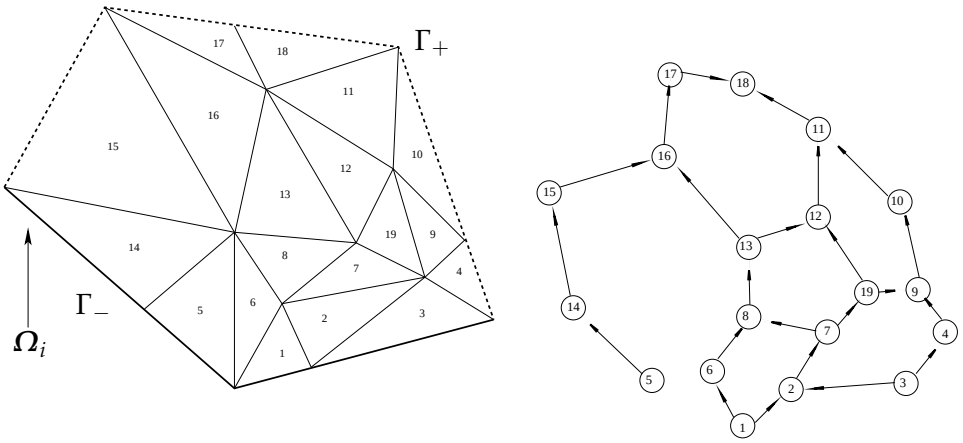


Figure 3.1: (a) A spatial mesh over some spatial domain with an ordinate direction $\boldsymbol{\Omega}_i$ with (b) the associated acyclic directed graph.

$(m, n) = e \in \tilde{E}$, where $m, n \in \tilde{V}$. A cycle is a set $\tilde{C}$

$$\tilde{C} = \{(n_1, n_2), (n_2, n_3), ..., (n_N, n_1)\} \subset \tilde{E}.$$

A graph is said to be acyclic if such a subset does not exist. A connected component $\tilde{S}$ is a subset of $\tilde{V}$ for which there is a path between any two nodes in $\tilde{S}$. $\tilde{S}$ is said to be a strongly connected component if there does not exist another connected component $\tilde{S}' \neq \tilde{S}$ such that $\tilde{S} \subset \tilde{S}'$. It is clear that any cycle is also a connected component, though not necessarily a strongly connected component, and that any directed graph may be uniquely partitioned into strongly connected components.

Given a characteristic direction $\boldsymbol{\Omega}_i$, the topological properties of our spatial finite element mesh $\mathcal{T}_S$ may be illustrated by representing it as a directed graph $\tilde{G}$. Each element in $\mathcal{T}_S$ corresponds to a vertex in $\tilde{V}$ and each element boundary which is not parallel to $\boldsymbol{\Omega}_i$ corresponds to an edge in $\tilde{E}$, with that edge directed in the direction of the characteristic flow across that boundary.

Figure 3.1 illustrates how a mesh may be represented as a graph in this manner. The further relationship between the graph and the structure of the stiffness matrix may be understood with reference to the numerical flux term in the discrete ordinates equation (3.1.35), which corresponds to an internal element boundary e, where $e = \partial\kappa_S \cap \partial\kappa_S{}'$ for two spatial elements κ_S and $\kappa_S{}'$. The term in the numerical flux involving $\psi^-_{h,i}(\mathbf{r})$ is responsible for the off-diagonal blocks in the angular element stiffness matrix. As this term considers the product of the basis functions on the element for which e is

an outflow boundary, and the test functions for the element on which e is an inflow boundary, this will yield a block below the diagonal if the elements are ordered so that the outflow element comes before the inflow element. If the ordering sets the inflow element before the outflow element, then the associated block will be above the diagonal.

Similarly, we may assert that an edge (m, n) on the graph corresponds to a lower triangular block in the stiffness matrix if m comes before n in the current ordering and an upper triangular block if n comes before m. Note that in the unlikely event that an element boundary is parallel to the characteristic direction, the aforementioned term will be zero and thus there will be no corresponding edge on the graph; in Figure 3.1 this occurs between elements 5 and 6.

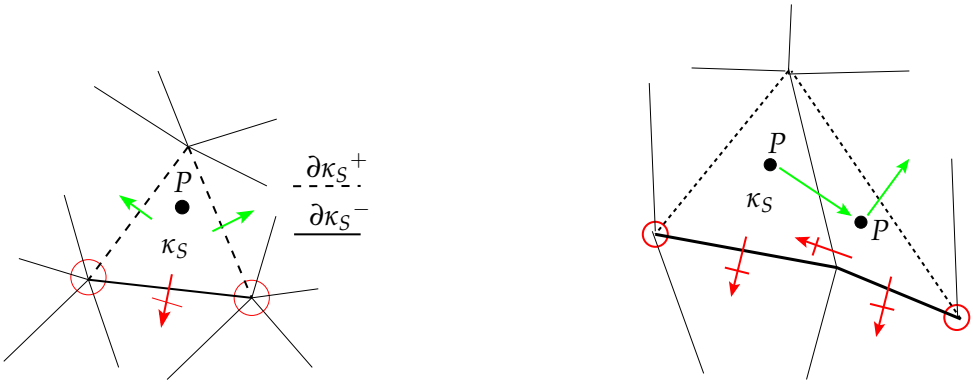
We now present a lemma which demonstrates that we may reorder the matrix $\mathbf{T}$ into a block triangular structure. This was first proved in my qualifying dissertation to the University of Nottingham, see [84].

Lemma 3.2.1. *For the discrete ordinates equation with ordinate direction $\boldsymbol{\Omega}_i$, there exists an element ordering such that the spatial discontinuous Galerkin method will yield a block lower triangular stiffness matrix for any spatial mesh $\mathcal{T}_S$, if for all $\kappa_S \in \mathcal{T}_S$, the inflow boundary $\partial {\kappa_S}^-$ is a connected set.*

Proof. For a graph $\tilde{G} = (\tilde{V}, \tilde{E})$, we say that a vertex ordering is *topological* if for all $(m, n) \in \tilde{E}$, m precedes n in that ordering. If this is the case then the associated stiffness matrix will be block lower triangular. As at least one topological ordering of $\tilde{V}$ will exist for any acyclic graph it will suffice to show that the graph associated with any permissible mesh will not contain any cycles.

To show this, we define a particle P that exists inside the mesh, as in Figure 3.2a. The particle is permitted to move freely within each element, but it may only move to another element through an outflow boundary; see Figure 3.2b. It is clear that P may move from one element to another if and only if there exists an edge in E that connects those two elements in the associated graph. Therefore we may state that a cycle exists in the graph $\tilde{G}$ if and only if it is possible for P to re-enter an element that it has previously left.

To demonstrate that this may never happen, consider the limits of the inflow boundary for some element κ_S, these are denoted by red circles in Figure 3.2a. Without loss of



(a) **(b)**

Figure 3.2: (a) A particle, P, is defined inside an element κ_S. (b) P may only move to another element through an outflow boundary.

generality we have performed a coordinate change so that the characteristic vector field points upwards; i.e. $\boldsymbol{\Omega}_i = (0, 1)^\top$.

Now, as P may only re-enter κ_S through its inflow boundary, if it is able to return to κ_S it must take one of two possible routes. It must either circle round to the right of the right-hand circle illustrated in Figure 3.2a, or circle round to the left of the left circle in the same figure. We show that it may not take the right-hand route and the other follows by symmetry.

To follow the necessary path, P must move to a lower y-coordinate than the circled node. As P may not move to another element by leaving through an inflow boundary, the only way that it may move to a lower coordinate is within an element. However, if P enters such an element (see Figure 3.2b) we are presented with the same problem as for the previously circled node, but with the right-hand limit of the inflow boundary in the new element.

Thus, we will never be able to move P into a position from which it may re-enter κ_S because every time that we are in a position to move to a lower y-coordinate than the currently circled node we will be presented with the same problem, but for the limit of the inflow boundary on the element in which P currently resides.

So, as P may never leave and re-enter any element on any mesh composed of elements with connected inflow boundaries, the associated graph for those meshes must necessarily be acyclic. Therefore, a topological element ordering will exist, and the stiffness matrix for that ordering will be block lower triangular. □

Note that this lemma applies to all meshes with convex elements, as they will always have a single, connected inflow boundary. A simple extension of this lemma also applies to meshes that have elements with characteristic boundaries; if we specify that P may not cross any boundary that coincides with a characteristic direction, then the same proof may be employed.

Once it has been deduced that a topological ordering exists, we require an efficient method for finding it. Tarjan's algorithm (see [85]) will find such an ordering for the case where cycles might exist, however, as we have deduced that the associated graph will be acyclic, the 'topological sort' algorithm may be applied to the problem and will find the order in a comparable speed but with less memory required.

The topological sort works by gradually removing vertices that have no incoming edges from the graph, placing them in the ordering and then deleting their associated outgoing edges, until the empty graph is all that remains. As every vertex and edge is considered (and removed from) the graph once, the algorithm takes $\mathcal{O}(|\tilde{V}| + |\tilde{E}|)$ operations to determine the ordering.

Algorithm 3.2.2. (Topological Sort)
For a directed acyclic graph $\tilde{G} = (\tilde{V}, \tilde{E})$, the following pseudo-code will output an ordered list L, which gives a, possibly not unique, topological ordering. Let n, m be vertices in the graph, then the algorithm is as follows:

$L \leftarrow$ list to contain ordered vertices
$S \leftarrow$ set of nodes with no incoming edges
while $S \neq \emptyset$ ***do***
  $S \leftarrow S \setminus n$
  $L \leftarrow L + n$
  for $m \in \tilde{V}$ such that $(n, m) \in \tilde{E}$ ***do***
    $E \leftarrow E \setminus (n, m)$
    if $\nexists\, l \in \tilde{V}$ such that $(l, m) \in \tilde{E}$ ***then***
      $S \leftarrow S + m$
    end if
  end for
end while
output(Proposed order: 'L') □

This topological sorting algorithm is sufficient for the case when it is known that no

cycles occur in the directed graph associated with the current spatial mesh and angular element (or ordinate direction). However, when there is a possibility that cycles will occur we need a more sophisticated algorithm. Figure 3.3 provides an illustration of how such a situation may arise for a simple unstructured mesh and an angular element κ_A. In order to deal with such a scenario we utilise Tarjan's strongly connected components algorithm, Algorithm 3.2.3, which both partitions the graph into strongly connected components and outputs these strongly connected components in reverse topological order. With the output from Tarjan's strongly connected components algorithm it is possible to implement a version of the sweeping procedure to efficiently apply the inverse of $\mathbf{T}$ by performing linear solves on each strongly connected component in the reverse order that they are outputted from the algorithm.

The basic idea of the algorithm is to use a recursively implemented depth first search to visit every vertex in the graph, putting each onto a stack data structure. The stack contains all vertices that have already been visited but have not yet been assigned to a strongly connected component. The algorithm also requires an array of length $|\tilde{V}|$ which contains indices indicating the order in which each vertex is visited by the depth first search and an array of length $|\tilde{V}|$ which contains a pointer to the vertex with the lowest index that can be reached from each vertex on the stack. Any time a new vertex is visited it receives an index and a pointer and is pushed onto the stack. When the depth first search has finished recursing on a vertex, if its pointer points to itself then it is the root of a strongly connected component. Vertices are then popped off the top of the stack until it is no longer on the stack. As the algorithm progresses through the graph, no strongly connected component is output before its successors, and therefore a topological ordering of the strongly connected components is obtained.

Tarjan's Strongly Connected Components Algorithm is given by Algorithm 3.2.3 and the recursive procedure that it calls for each newly visited vertex is given by Algorithm 3.2.4 .

Algorithm 3.2.3. (Tarjan's Strongly Connected Components Algorithm)
For a directed graph $\tilde{G} = (\tilde{V}, \tilde{E})$, the following pseudo-code will identify the strongly connected components and output them in reverse topological order. Let $n, m \in \tilde{V}$ then the algorithm is as follows:

$Stack \leftarrow$ a stack to contain the vertices in the graph
$n.ind \leftarrow$ the index of vertex n

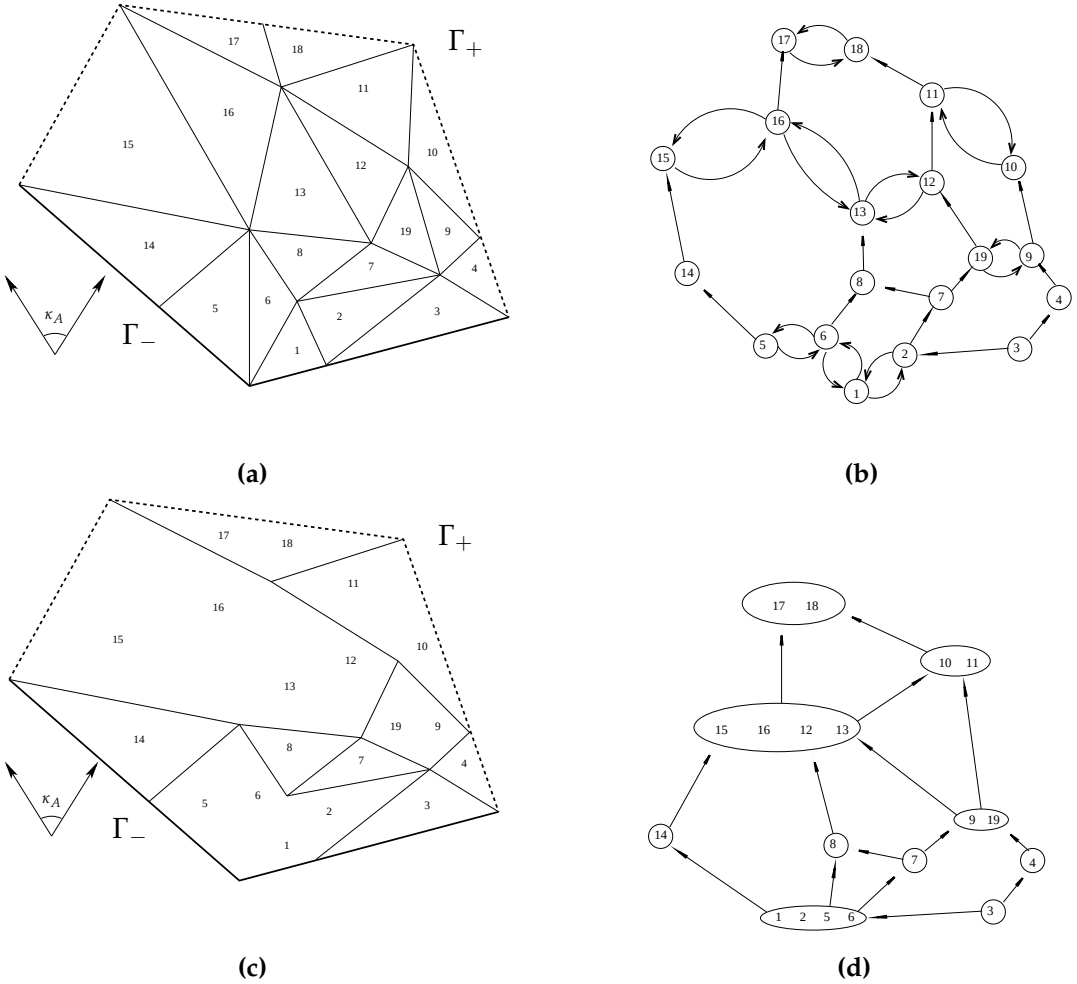


Figure 3.3: In (a) is the same spatial mesh as in Figure 3.1 with an angular element $\kappa_A \in \mathcal{T}_A$. As κ_A incorporates a continuum of values in its azimuthal coordinate it defines a graph that contains several nontrivial strongly connected components (see (b)). In (c) is the same mesh except the elements of each nontrivial strongly connected component are agglomerated so that an acyclic graph can be obtained (see (d)). Tarjan's strongly connected components algorithm will take (b) as an input and its output will tell us which elements need to be agglomerated in order to produce (d). It will also give us a topological ordering of (d) that will enable us to quickly apply $\mathbf{T}^{-1}$ to a vector.

$n.ptr \leftarrow$ the pointer to the lowest vertex reachable from n
$i = 0$
for $n \in \tilde{V}$ ***do***
  $n.ind = 0$
  $n.ptr = 0$
end for
for $n \in \tilde{V}$ ***do***
  if $n.ind = 0$ ***then***
    Call Algorithm 3.2.4 for vertex n
  end if
end for □

Algorithm 3.2.4. (TSCC Recursive Procedure)
A recursive procedure required by Algorithm 3.2.3 to process successor vertices to the current vertex $n \in \tilde{V}$. The algorithm proceeds as follows:

$i = i + 1$
$n.ind = i$
$n.ptr = i$
Push n onto *Stack*
for $m \in \tilde{V}$ such that $(n, m) \in \tilde{E}$ ***do***
  if $m.ind == 0$ ***then***
    Call Algorithm 3.2.4 for vertex m
    $n.ptr = \min(n.ptr , m.ptr)$
  else
    if $m \in Stack$ ***then***
      $n.ptr = \min(n.ptr , m.ptr)$
    end if
  end if
end for
if $n.ind == n.ptr$ ***then***
  Output the next strongly connected component
  repeat
    Pop m off *stack*
  until $n==m$
  End of current strongly connected component

end if □

As every vertex is visited once, when we call the recursive procedure, every edge is considered at most twice and we see that this algorithm will require $\mathcal{O}(|\tilde{V}| + |\tilde{E}|)$ operations. Due to its usefulness and popularity, there are many available implementations of Tarjan's strongly connected components algorithm. For the software developed in this thesis we use the version from the HSL Mathematical Software Library, see [86]. The routine used is the `MC13` subroutine which comes packaged as part of the `MA48` linear solver.

Now that we have the tools to write the matrix $\mathbf{T}$ in block triangular form, we can describe the sweeping procedure for applying $\mathbf{T}^{-1}$ to a vector without assembling either matrix. Algorithm 3.2.5 describes the procedure, which is equivalent to a block forward substitution algorithm for a block triangular matrix, in terms of our spatial finite element mesh and angular element.

Algorithm 3.2.5. (Sweeping Algorithm)
This algorithm takes a spatial finite element mesh $\mathcal{T}_S$, its directed graph $(\tilde{V}, \tilde{E})$ associated with an angular element κ_A and a topologically ordered list of its strongly connected components $SCC_i \subset \tilde{V}$, where $i = 1, \cdots, N_{SCC}$ and

$$\bigcup_i SCC_i = \tilde{V}$$

and sweeps through the spatial domain applying $\mathbf{T}^{-1}|_{\kappa_A}$ to a vector $\mathbf{v_1}$. The algorithm proceeds as follows:

Get maximum strongly connected component size
Allocate local matrix $\mathbf{A}$, solution $\mathbf{x}$ and right-hand side $\mathbf{b}$
Allocate data structures for the linear solver
Label all $\kappa_S \in \mathcal{T}_S$ ***UNSOLVED***
for $i = 1, \cdots, N_{SCC}$ ***do***
  Include values from $\mathbf{v_1}$ in $\mathbf{b}$
  for $\kappa_S \in SCC_i$ ***do***
    Compute volume integrals for κ_S
    Add to $\mathbf{A}$ and $\mathbf{b}$
    if $\partial\kappa_S \cap \Gamma_+ \neq \emptyset$ ***then***
      Compute face integrals for $\partial\kappa_S$ and include in $\mathbf{b}$

end if
end for
for $\kappa_S \cap \kappa_S{}' \in \tilde{E}$ ***do***
if $\{\kappa_S, \kappa_S{}'\} \subset SCC_i$ ***then***
Compute face integrals for $\kappa_S \cap \kappa_S{}'$
Add to $\mathbf{A}$ *and* $\mathbf{b}$
end if
if $\kappa_S \in SCC_i$ ***and*** $\kappa_S{}' \notin SCC_i$ ***then***
Compute face integrals for $\kappa_S \cap \kappa_S{}'$
if $\kappa_S{}'$ *is marked* ***SOLVED then***
Get solution values from $\kappa_S{}'$, $\mathbf{x}_{\kappa_S{}'}$
Build a block $\mathbf{B}_{\kappa_S \cap \kappa_S{}'}$ *from the values from equation* (3.2.8)
Subtract $\mathbf{B}_{\kappa_S \cap \kappa_S{}'} \mathbf{x}_{\kappa_S{}'}$ *from the* κ_S *values in* $\mathbf{b}$
else
Include $\partial\kappa_S \setminus \Gamma$ *face integrals in* $\mathbf{A}$
end if
end if
if $\kappa_S{}' \in SCC_i$ ***and*** $\kappa_S \notin SCC_i$ ***then***
Compute face integrals for $\kappa_S \cap \kappa_S{}'$
if κ_S *is marked* ***SOLVED then***
Get solution values from κ_S, $\mathbf{x}_{\kappa_S}$
Build a block $\mathbf{B}_{\kappa_S \cap \kappa_S{}'}$ *from the values from equation* (3.2.8)
Subtract $\mathbf{B}_{\kappa_S \cap \kappa_S{}'} \mathbf{x}_{\kappa_S}$ *from the* $\kappa_S{}'$ *values in* $\mathbf{b}$
else
Include $\partial\kappa_S{}' \setminus \Gamma$ *face integrals in* $\mathbf{A}$
end if
end if
end for
Solve $\mathbf{A}\mathbf{x} = \mathbf{b}$
Store solution values
for $\kappa_S \in SCC_i$ ***do***
Label κ_S ***SOLVED***
end for
end for

□

Note that at the beginning of the algorithm we allocate memory for the individual strongly connected component matrices and the linear solver. It is an advantage of this algorithm that we may allocate this memory only once for a very large number of linear solves as memory allocation can be a significant time bottleneck for computer software. It is an additional advantage that the maximum amount of memory required for these data structures is limited by the maximum size of a strongly connected component in the graph, which in practice is much smaller than the full problem size.

The choice of a linear solver to solve the strongly connected component matrix equation $\mathbf{Ax} = \mathbf{b}$ was found to have a significant effect on the performance of our implementation of this algorithm. The vast majority of these linear systems that need to be solved are small enough that they are most efficiently tackled by a dense solver, however it is possible that for some combinations of spatial and angular meshes linear systems could arise for which a sparse solver is preferable. This will be the case when one or more large strongly connected components are identified by Algorithm 3.2.3. For 2 dimensional problems this can occur for regular meshes, when the angular element couples together many adjacent spatial elements. See [87] for an example of when large strongly connected components arise in discrete ordinates methods for irregular 3 dimensional spatial meshes. It was decided that a sparse direct solver would be the most suitable choice of linear solver. This is because modern sparse direct solvers will automatically detect when a system would be better treated as dense and then switch to a dense factorisation implemented using efficient level 3 BLAS subroutines.

Two linear solvers were compared for the solution to the strongly connected component matrices, the `MA41` linear solver from the HSL library, together with the block preprocessor developed in Chapter 2, and the sequential version of the MUMPS linear solver. Though these pieces of software implement the same sparse direct method for the solution of linear systems, it was found that `MA41` was preferable for the present case for three reasons. Firstly, MUMPS allocates the memory that it uses for the *LU* factorisation dynamically whereas `MA41` requires the user to allocate arrays of integers and real numbers prior to using its interface subroutines. This control of memory allocation provided by `MA41` means that many systems can be solved without repeated deallocation and reallocation of memory, whereas MUMPS needs to be reinitialised before each solve. The second reason that `MA41` is preferable is because it is thread-safe

and can therefore be used inside an OpenMP loop. Finally, tests on a variety of structured and unstructured meshes for high order DG methods showed that `MA41` with block preprocessing was much faster than sequential MUMPS.

3.2.3 Solving the source problem

For the neutron transport source problem we seek to compute the vector of solution values $\boldsymbol{\psi}$ which solves equation (3.2.13) for a given spatial domain, material cross sections and source term. To this end we consider two algorithms: a stationary iterative procedure, referred to as the source iteration and the GMRES algorithm. The source iteration is the most commonly referred to solution strategy in the literature and is equivalent to a Jacobi iteration, though some authors have explored using various Krylov methods such as GMRES, Orthomin(k) and the conjugate gradient method (these are discussed below).

We explain the source iteration by first rewriting equation (3.2.13) as:

$$\mathbf{T}\boldsymbol{\psi}_i = (\mathbf{F} + \mathbf{S})\,\boldsymbol{\psi}_{i-1} + \mathbf{q}, \tag{3.2.16}$$

where $\boldsymbol{\psi}_i$ is the approximate solution vector at the ith iteration of this (or any) iterative procedure. We select an initial guess for the solution of $\boldsymbol{\psi}_0$ and then repeatedly solve equation (3.2.16) until:

$$\|\boldsymbol{\psi}_i - \boldsymbol{\psi}_{i-1}\| \leq \text{tol}, \tag{3.2.17}$$

where tol is a small positive real number. Note that solving equation (3.2.16) requires applying the inverse of the discrete transport matrix $\mathbf{T}^{-1}$ to the vector obtained by computing the right-hand side. In fact all of the solution procedures considered will utilise the application of this matrix, either directly or as a preconditioner for a Krylov method, which is why the efficient computation of its action discussed in Section 3.2.2 is essential.

The convergence properties of the source iteration are determined by the spectral radius of the system, which has been shown via the Fourier analysis of Reed to be equal to the scattering ratio $c = \Sigma_s/\Sigma_t$ (see [88]). As c becomes close to 1, as can happen in problems with large diffusive regions, the number of iterations required can become prohibitively large. In order to render such problems tractable many authors use diffusion synthetic acceleration (DSA) to reduce the number of iterations required for the solution procedure. DSA, described by Larsen in [89], works by alternating transport

sweeps with the multiplication by the inverse of a diffusion operator; this effectively reduces the spectral radius of the iterated system. When the discretisation of the diffusion operator is consistent with the discretisation of the transport operator it can be shown that the spectral radius of the DSA system is bounded above by $0.2247c$ for finite difference discretisations in one dimensional geometries (see [90]); this can lead to massive reductions in the number of iterations required for convergence. For the main results collected in this thesis we do not implement DSA but instead use a GMRES iteration preconditioned by transport sweeps; this has been shown by Patton and Holloway in [91] to perform comparably to DSA, at least in slab geometries.

The source iteration is not the only stationary iterative method that has been used for neutron transport. An improvement on it is the damped Richardson iteration which uses an appropriately chosen damping factor α to guarantee convergence. The ith damped Richardson iterate is given by:

$$\boldsymbol{\psi}_i = \boldsymbol{\psi}_{i-1} - \alpha \left[(\mathbf{T} - \mathbf{F} - \mathbf{S}) \, \boldsymbol{\psi}_{i-1} - \mathbf{q} \right]. \tag{3.2.18}$$

It can be shown that if $(\mathbf{T} - \mathbf{F} - \mathbf{S})$ is sufficiently well conditioned then the sequence $\boldsymbol{\psi}_{i-1}$ will converge geometrically, see [68]. Both the source iteration and the damped Richardson iteration compute the next approximation of the solution based only on the solution from the previous iteration via the product with a matrix. Whilst this may have the advantage of using a small amount of memory, there is a lot of information about the problem contained in each iterate of the solution which is discarded after every iteration. It is for this reason that one would expect a Krylov method could be utilised to speed up the convergence.

Several Krylov methods have been investigated in the literature for solving linear transport problems including the conjugate gradient method (applied to a symmetrised system in [92] and [93]), Orthomin(k) (see [94]), GMRES (see [88] and [95] amongst many others) and conjugate gradient squared [96]. As the matrix $\mathbf{T}$ induced by our discretisation is not symmetric we may not exploit the conjugate gradient method without computing an expensive symmetrisation. We may also dismiss Orthomin(k) as it is mathematically equivalent to GMRES but with worse numerical properties (see [97]). In order to decide between the remaining methods we make reference to the paper of Oliveira and Deng ([96]) which compares various Krylov methods for problems in a slab geometry with ILU and multigrid preconditioners and finds that GMRES and conjugate gradient squared solved their test problems with the lowest amount of CPU

time. Thus we choose GMRES as our Krylov method.

The generalised minimal residual of Saad and Shultz, or GMRES, method solves a linear system $\mathbf{A}\boldsymbol{\psi} = \mathbf{b}$ by constructing a succession of approximate solution vectors $\boldsymbol{\psi}_m$ that minimise the residual norm over $\boldsymbol{\psi}_0 + \mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$, where $\mathbf{A}$ is the matrix, $\mathbf{r}_m = \mathbf{b} - \mathbf{A}\boldsymbol{\psi}_m$ is the mth residual vector and the mth Krylov subspace is

$$\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0) = \text{span}\left\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \cdots, \mathbf{A}^{m-1}r_0\right\}. \tag{3.2.19}$$

Given an initial guess to the solution $\boldsymbol{\psi}_0$ we use the Arnoldi process to build $\mathbf{V}_m$ which is an orthonormal basis of the mth Krylov space, and then the mth approximation to the solution is given by

$$\boldsymbol{\psi}_m = \boldsymbol{\psi}_0 + \mathbf{V}_m\mathbf{y}_m, \tag{3.2.20}$$

where $\mathbf{y}_m$ is a real vector of length m that is chosen so that r_m is minimised over $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$. The orthogonal projection of $\mathbf{A}$ onto $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$ gives the upper Hessenberg matrix $\mathbf{H}_m = \mathbf{V}_m^\top\mathbf{A}\mathbf{V}_m$ of order m. We obtain $\mathbf{y}_m$ by minimising the functional

$$\begin{aligned} J(y) &= \|\mathbf{b} - \mathbf{A}\boldsymbol{\psi}\|_2 && (3.2.21)\\ &= \|\mathbf{b} - \mathbf{A}(\boldsymbol{\psi}_0 + \mathbf{V}_m\mathbf{y})\|_2 && (3.2.22)\\ &= \|\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}\|_2. && (3.2.23) \end{aligned}$$

From the Arnoldi process we have

$$\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{H}_m + h_{m+1,m}\mathbf{v}_{m+1}\mathbf{e}_m^\top, \tag{3.2.24}$$

where $\mathbf{e}_m$ is the mth canonical basis vector. If we write

$$\bar{\mathbf{H}} = \begin{bmatrix} \mathbf{H}_m \\ 0, \cdots, h_{m+1,m} \end{bmatrix}, \tag{3.2.25}$$

then

$$\mathbf{A}\mathbf{V}_m = \mathbf{V}_{m+1}\bar{\mathbf{H}}_m. \tag{3.2.26}$$

We can use this to rewrite the functional

$$\begin{aligned} J(y) &= \|\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}\|_2 && (3.2.27)\\ &= \|\beta\mathbf{v}_1 - \mathbf{V}_{m+1}\bar{\mathbf{H}}_m\mathbf{y}\|_2 && (3.2.28)\\ &= \|\mathbf{V}_{m+1}(\beta\mathbf{e}_1 - \bar{\mathbf{H}}_m\mathbf{y})\|_2, && (3.2.29) \end{aligned}$$

where $\beta = \|\mathbf{r}_0\|_2$ and $\mathbf{v}_1 = \mathbf{r}_0/\beta$. Then we can compute $\mathbf{y}_m$ by computing the following minimisation problem:

$$\mathbf{y}_m = \min_{\mathbf{y}\in\mathbb{R}^m} \|\beta\mathbf{e}_1 - \bar{\mathbf{H}}_m\mathbf{y}\|. \tag{3.2.30}$$

As this is a linear least squares problem of size $(m+1)\times m$, while m is relatively small the solution to this problem is quick to compute. With exact arithmetic the GMRES method is guaranteed to converge in N iterations, where N is the size of the system, so GMRES could potentially be used as a direct method. However, as the number of operations and the amount of memory required to store $\mathbf{V}_m$ increase with each iteration, this would rapidly become unfeasible for large sparse systems. In order to limit the memory requirements we employ the restarted GMRES method, or GMRES(M). GMRES(M) is exactly the same as GMRES except that the size of the orthonormal basis is limited to M. If the stopping criterion is not met after M iterations then the process is restarted with a new initial guess, $\boldsymbol{\psi}_0 \leftarrow \boldsymbol{\psi}_M$.

For the results collected in this thesis we use the CERFACS implementation of the GMRES(M) algorithm by Frayssé, Giraud, Gratton and Langou (see [98] for documentation). These routines use a backwards communication interface which allows the user to implement the most computationally intensive parts of the algorithm; the matrix vector products and the scalar products. As the GMRES(M) algorithm can be slow to converge we will utilise a preconditioner to speed convergence by solving a translated system with the same solution. For our preconditioner we use transport sweeps, that are equivalent to the application of $\mathbf{T}^{-1}$, which were found to perform well by Patton et al. in [91].

The solution of the source problem by the right preconditioned GMRES(M) method is then obtained by writing equation (3.2.13) as

$$(\mathbf{T} - \mathbf{F} - \mathbf{S})\,\mathbf{T}^{-1}\mathbf{y} = \mathbf{q}, \tag{3.2.31}$$

and solving $\mathbf{A}\mathbf{y} = \mathbf{b}$ where $\mathbf{A} = (\mathbf{T} - \mathbf{F} - \mathbf{S})\,\mathbf{T}^{-1}$, $\mathbf{b} = \mathbf{q}$ and $\mathbf{y} = \mathbf{T}\boldsymbol{\psi}$. The solution vector $\boldsymbol{\psi}$ may then be obtained by applying $\mathbf{T}^{-1}$ to $\mathbf{y}$.

Table 3.1 compares the number of iterations required to solve the artificially forced source problem that we will define in Section 3.4 by GMRES compared to using a source iteration. The data is for a series of five uniformly refined meshes, each for a $p = 1, q = 0$ approximation and a $p = 2, q = 1$ approximation. The convergence tolerances were set to 10^{-9}. We note that the GMRES algorithm uses far fewer iterations than the source

$q = 0, p = 1$			$q = 1, p = 2$		
N	GMRES iterations	Source iterations	N	GMRES iterations	Source iterations
64	7	19	576	7	21
1 024	7	21	9 216	7	23
16 384	7	24	147 456	8	25
262 144	8	26	2 359 296	8	26
4 194 304	8	27	-	-	-

Table 3.1: The total number of iterations required to solve the source problem from Section 3.4 using the source iteration and preconditioned GMRES for a series of five uniformly refined meshes.

$q = 0, p = 1$			$q = 1, p = 2$		
N	Source iteration	GMRES method	N	Source iteration	GMRES method
64	0.011041	0.052615	576	2.4137	1.4723
1 024	0.040212	0.10498	9 216	24.25	13.699
16 384	0.6375	1.5918	147 456	450.488	260.14
262 144	13.281	30.246	2 359 296	7 665.235	4 252.223
4 194 304	343.321	525.125	-	-	-

Table 3.2: The total time (in seconds) to complete a solve of the source problem from Section 3.4 for a series of five uniformly refined meshes.

iteration and that the number of iterations required for the convergence of the source iteration grows with problem size whereas it remains constant for GMRES. Table 3.2 gives the total time required for these problems. Note that though the source iteration solves the lower order problems more quickly, GMRES is the preferable algorithm for the higher order problems. As the $p = 1$, $q = 0$ solutions are all computed in under 10 minutes for both algorithms, we conclude that either algorithm is suitable for lower order computations. However, as GMRES performs considerably better for more time consuming higher order problems, we choose GMRES for all further source problem calculations.

3.2.4 Solving the eigenvalue problem

We also collect results from two candidate algorithms for the solution of the discrete k_{eff}-eigenvalue problem from equation (3.2.14). The first algorithm that we consider is a simple power iteration, which is the most commonly used method in the literature and the second is the Implicitly Restarted Arnoldi Method. The generalised eigenvalue problem in equation (3.2.14) may be written as a regular eigenvalue problem by first considering a series expansion of the $\mathbf{T} - \mathbf{S}$ term. By multiplying both sides of the equation by $\mathbf{T}^{-1}$ we obtain

$$\left(\mathbb{I} - \mathbf{T}^{-1}\mathbf{S}\right)\boldsymbol{\psi} = \frac{1}{k_{\text{eff},h}}\mathbf{T}^{-1}\mathbf{F}\boldsymbol{\psi}. \tag{3.2.32}$$

Then by multiplying through by $k_{\text{eff},h}\left(\mathbb{I} - \mathbf{T}^{-1}\mathbf{S}\right)^{-1}$ on both sides of the equation, we get

$$k_{\text{eff},h}\boldsymbol{\psi} = \left(\mathbb{I} - \mathbf{T}^{-1}\mathbf{S}\right)^{-1}\mathbf{T}^{-1}\mathbf{F}\boldsymbol{\psi}. \tag{3.2.33}$$

We then write $\left(\mathbb{I} - \mathbf{T}^{-1}\mathbf{S}\right)^{-1}$ as the Neumann series

$$\left(\mathbb{I} - \mathbf{T}^{-1}\mathbf{S}\right)^{-1} = \sum_{k=0}^{\infty}\left(\mathbf{T}^{-1}\mathbf{S}\right)^{k}, \tag{3.2.34}$$

to obtain

$$k_{\text{eff},h}\boldsymbol{\psi} = \mathbf{M}\boldsymbol{\psi}, \tag{3.2.35}$$

where

$$\mathbf{M} = \sum_{k=0}^{\infty}\left(\mathbf{T}^{-1}\mathbf{S}\right)^{k}\mathbf{T}^{-1}\mathbf{F}. \tag{3.2.36}$$

Then at each iteration of the power method the ith approximation of the dominant eigenvector is given by

$$\boldsymbol{\psi}_i = \frac{\mathbf{M}\boldsymbol{\psi}_{i-1}}{\|\mathbf{M}\boldsymbol{\psi}_{i-1}\|}, \tag{3.2.37}$$

and the ith approximation of the dominant eigenvalue is given by $\|\mathbf{M}\boldsymbol{\psi}_{i-1}\|$. The power iteration is terminated when the absolute value of the difference between successive eigenvalue approximations falls below a given outer tolerance tol_{out}. The application of the matrix $\mathbf{M}$ is computed by truncating the Neumann sum when the norm of the solution vector settles at a given value, within some inner tolerance tol_{in}. More precisely we approximate the application of $\mathbf{M}$ by

$$\mathbf{M}\boldsymbol{\psi}_{i-1} = \sum_{k=0}^{\infty} \left(\mathbf{T}^{-1}\mathbf{S}\right)^k \mathbf{T}^{-1}\mathbf{F}\boldsymbol{\psi}_{i-1} \tag{3.2.38}$$

$$\approx \sum_{k=0}^{K} \left(\mathbf{T}^{-1}\mathbf{S}\right)^k \mathbf{T}^{-1}\mathbf{F}\boldsymbol{\psi}_{i-1}, \tag{3.2.39}$$

where

$$\left\| \left(\mathbf{T}^{-1}\mathbf{S}\right)^K \mathbf{T}^{-1}\mathbf{F}\boldsymbol{\psi}_{i-1} \right\| \leq \text{tol}_{in}. \tag{3.2.40}$$

This is the simpler of the two eigenvalue algorithms and, like the fixed point iteration for the source iteration, it may be implemented as the repeated application of $\mathbf{T}^{-1}$ to a given vector. The convergence of this algorithm is determined by the ratio between the dominant eigenvalue and the next largest eigenvalue. If $\lambda_1 = k_{\text{eff},h}$ is the largest eigenvalue and $\lambda_2 < \lambda_1$ is the next largest eigenvalue then the error after i iterations is proportional to

$$\left|\frac{\lambda_2}{\lambda_1}\right|^i. \tag{3.2.41}$$

The ratio $\frac{\lambda_2}{\lambda_1}$ is referred to as the dominance ratio. For problems where the dominance ratio is close to one it is clear that the power iteration will take a very large number of iterations to converge. In [99] the authors use power iterations to compute approximations to k_{eff} for a series of three dimensional k_{eff}-eigenvalue problems with dominance ratio approaching 1 and found that power iteration could take more than 1000 outer iterations to compute the eigenvalue, even with the relatively modest tolerance of 10^{-4}. For many practical problems, however, the dominant eigenvalue is sufficiently isolated so that the power iteration will converge in an acceptable manner; hence many authors still use the power iteration for their computations.

The potentially high computational requirements of the power iteration has led many researchers to investigate how it may be accelerated. In [100] Adams et al. review several of the strategies that have been proposed. The power iteration can be accelerated by storing the vectors from previous iterations and combining them to produce the next approximate eigenvector. This is the strategy of the Chebyshev iteration, which uses a combination of all previous iterates weighted to correspond to a Chebyshev polynomial, see [101]. For problems where the dominance ratio approaches 1 the Chebyshev iteration can greatly reduce the number of iterations required. One advantage of the Chebyshev iteration is that not all of the previous iterates need to be stored in order to incorporate the information that they contained into the next iterate. This is achieved using the three term recursion formula for the Chebyshev polynomial, which means that only the previous two iterates need to be stored.

Other methods for accelerating power iteration use an assumed prior knowledge of the spectrum in order to accelerate the convergence. One such method is the shifted power iteration which works by shifting the spectrum by a positive constant $k_{\text{shift}} > k_{\text{eff}}$. If this shifting parameter is well chosen the shifted power iteration can converge very rapidly. In [102] an estimate of the eigenvalue is also used to accelerate convergence in the inverse power iteration method, which at each iteration applies the inverse of the difference between the original matrix and $k_{\text{est}}\mathbb{I}$, where k_{est} is an estimate of the eigenvalue. These methods yield convergence proportional to

$$\left| \frac{k_{\text{eff}} - k_{\text{shift}}}{\lambda_2 - k_{\text{shift}}} \right|, \tag{3.2.42}$$

and

$$\left| \frac{k_{\text{eff}} - k_{\text{est}}}{\lambda_2 - k_{\text{est}}} \right|, \tag{3.2.43}$$

respectively. The inverse power iteration method has the advantage over the shifted power iteration in that the estimated eigenvalue does not have to be larger than the solution eigenvalue but the disadvantage of increased computational cost per iteration. Both of these methods have the disadvantage that they require a reasonable estimate of the leading eigenvalue in order to be effective, which might not be available. Also as neutron transport problems in general have matrices that are not symmetric, the spectrum may extend into the complex plane, this can lead to complications when trying to compute optimal estimates of the required acceleration parameters, see [103].

These iterative eigenvalue algorithms can all be seen as Krylov subspace methods in that they look for their solutions in a space constructed by taking the span of the images

induced by the matrix on a vector. The power iteration, shifted power iteration and inverse power iteration look for their solutions in a Krylov space of order 1 and the Chebyshev iteration seeks a solution using all previous iterates. The next method that we consider can be seen as an improvement on these in that it uses the Krylov subspace constructed from the span of all previous images, but this time uses a more robust algorithm to compute the eigenvalue. This method is the Implicitly Restarted Arnoldi Method.

The basic Arnoldi eigenvalue algorithm works similarly to the GMRES method in that it uses a stabilised Gram-Schmidt iteration to build an orthogonal basis $\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_{i+1}$ of the present Krylov subspace:

$$\mathcal{K}_{i+1}(\mathbf{M}, \boldsymbol{\psi}_0) = \text{Span}\left\{\boldsymbol{\psi}_0, \mathbf{M}\boldsymbol{\psi}_0, \mathbf{M}^2\boldsymbol{\psi}_0, \cdots, \mathbf{M}^i\boldsymbol{\psi}_0\right\} \tag{3.2.44}$$

$$= \text{Span}\left\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_{i+1}\right\}. \tag{3.2.45}$$

It then considers the upper Hessenberg matrix $\mathbf{H}_{i+1}$ which is the orthogonal projection of $\mathbf{M}$ onto the Krylov subspace, represented by the basis $\{\mathbf{q}_i\}$. The eigenvalues of $\mathbf{H}_{i+1}$ are known as Ritz eigenvalues and converge to the extreme eigenvalues of $\mathbf{M}$. As we seek the dominant eigenvalue, no shifting of the spectrum will be required for the Ritz values to include $k_{\text{eff},h}$. As the matrix $\mathbf{H}_{i+1}$ is upper Hessenberg, its eigenvalues can be quickly computed using a version of the QR algorithm.

The Implicitly Restarted Arnoldi Method (IRAM) is an algorithmic variant on the basic Arnoldi eigenvalue algorithm which uses a restart in order to limit the amount of memory that will be used, at the cost of possibly increasing the number of iterations that will be needed for convergence. It can be viewed as a synthesis between the Arnoldi eigenvalue iteration and the Implicitly Shifted QR scheme. For the results collected in this thesis we use the ARnoldi PACKage (ARPACK) implementation of the Implicitly Restarted Arnoldi Method, see [104] for documentation.

The ARPACK software package has been used for k_{eff}-eigenvalue problems before by Warsa et al., in [99]. The authors compared the ARPACK implementation of IRAM with a basic power iteration and found it to be robust and extremely efficient for several challenging three dimensional test problems. Indeed they found that the additional computational cost per iteration of using the ARPACK routines was vastly outweighed by the convergence behaviour, particularly for problems with a dominance ratio close to 1.

Inner iterations

$q=0, p=1$			$q=1, p=2$		
N	GMRES	Neumann sum	N	GMRES	Neumann sum
480	151	597	4 224	149	441
7 680	150	467	67 584	148	399
122 880	149	420	1 081 344	150	357
1 966 080	150	378	-	-	-

Outer iterations

$q=0, p=1$			$q=1, p=2$		
N	IRAM	Power method	N	IRAM	Power method
480	15	23	4 224	15	21
7 680	15	21	67 584	15	21
122 880	15	21	1 081 344	15	21
1 966 080	15	21	-	-	-

Table 3.3: The total number of inner and outer iterations to solve the seventh Los Alamos criticality problem for four different iterative procedures on a series of uniformly refined meshes. The convergence tolerances were set at 10^{-9}.

To exploit ARPACK to compute the eigenvalue-eigenvector pair, the user must supply the action of a matrix on a vector. So in order to use ARPACK we once again rewrite equation (3.2.14) as

$$k_{\text{eff},h}\boldsymbol{\psi} = \mathbf{M}\boldsymbol{\psi}, \tag{3.2.46}$$

but this time instead of writing $\mathbf{M}$ as a Neumann sum, we write it as:

$$\mathbf{M} = (\mathbf{T} - \mathbf{S})^{-1}\,\mathbf{F}, \tag{3.2.47}$$

where the action of $(\mathbf{T} - \mathbf{S})^{-1}$ is computed using the GMRES method preconditioned by $\mathbf{T}^{-1}$. Note that though this is not precisely the same as in equation (3.2.33), its action on a vector will be the same.

Table 3.3 compares the number of iterations required to solve a k_{eff}-eigenvalue problem for a series of uniformly refined meshes for a $p = 1$, $q = 0$ approximation and a $p = 2$, $q = 1$ approximation. The table compares the stationary iterative procedure of the

$q=0, p=1$			$q=1, p=2$		
N	Power method	IRAM	N	Power method	IRAM
480	3.1728	1.2798	4 224	265.909	158.902
7 680	36.824	19.093	67 584	2 212.118	1 381.509
122 880	547.158	305.355	1 081 344	31 649.974	19 735.881
1 966 080	10 476.424	6 021.301	-	-	-

Table 3.4: The total time taken (in seconds) to perform the calculations from Table 3.3.

Neumann iteration nested inside a power iteration with the Krylov based algorithm of preconditioned GMRES nested inside the IRAM iteration. The total number of inner iterations for each procedure is given as well as the total number of outer iterations. Note that the Krylov based algorithm yields significant reductions in both the number of outer iterations, as well as the total number of inner iterations to be computed. As for the two source iterative procedures that we compared, this results in a significant reduction in the overall number of iterations required. Table 3.4 gives the time, in seconds, for each of these solves and the percentage reduction in the time required by the IRAM compared to the power method. Note that, in contrast to the algorithms tested for the source problem, the Krylov subspace based method out performs the stationary method for all problems, including small lower order problems. For this reason we select the Krylov based method for all further criticality computations in this thesis.

Notice that as the problem size grows the number of iterations for the Krylov based method remains constant, however the number of terms in the Neumann sum gets smaller. This is due to the weak stopping criterion for the Neumann iteration defined in equation (3.2.40). The more robust norm wise backwards error stopping criterion utilised for the GMRES method leads to the number of iterations remaining constant.

3.2.5 Parallel implementation

Each of the solution algorithms discussed in the previous two sections requires a very large number of floating-point operations to compute its solution. Indeed, in contrast to what might be expected for the discretisation of a higher dimensional integro-differential equation, it is the time required by the number of operations that typically

determines the largest problem size that can be tackled and not the limits on the available memory. Therefore, utilisation of modern parallel computer architectures is crucial if one wishes to compute the solution to challenging problems of industrial interest.

As the nuclear field and computational science have in many ways developed in synergy with one another it is not surprising that parallel computers have been used to tackle problems in neutronics almost as long as these architectures have been available. In [105], Azmy gives a review of the applications of parallel computing to neutronics up until the late 90s. For modern neutron transport codes it is almost assumed that the implementation will be parallel, whether on a shared memory, distributed memory or GPU environment. The most coarsely grained parallelism is best implemented on shared memory machines, for medium grained parallelism a network of CPUs with distributed memory is more suitable and for the most fine grained and highly regular computations GPU computing can be used.

In parallel computing there is a strong link between the architecture that is being used and the types of algorithms that will be most efficient. As such the choice of how to parallelise a code is determined by the computing resources available. For example in [87] Plimpton et al. discuss a distributed memory implementation of a code for source problems in radiative transport with unstructured meshes. In order to compute the mesh orderings and find any cycles prior to the sweeping procedure in parallel the authors replace Tarjan's strongly connected components algorithm with an alternative algorithm by Fleischer. This is because Tarjan's algorithm uses a depth first search which does not parallelise well over a distributed architecture so an alternative algorithm which uses a breadth first search had to be employed.

The present code was developed to run on a shared memory architecture with 8-16 physical CPU cores. In order to parallelise it we first need to decide which tasks to parallelise and how to partition the independent variables between the processors. As discussed in previous sections the tasks that form the greatest bottleneck in terms of computational time are the computation of the action of $\mathbf{T}$ and $\mathbf{T}^{-1}$. We have three options for splitting the independent variables between processors: we can divide them by energy group, by a partition of the angular domain or by a partition of the spatial domain.

The option to partition work between processors by dividing the energy groups would provide the coarsest grain parallelism. This partitioning would benefit from the fact

that the multigroup approximation leaves the energy groups uncoupled in the $\mathbf{T}$ matrix. However, though some early authors parallelised over the energy spectrum, it is not suitable for the present implementation because the typical number of energy groups in a multigroup approximation is lower than the number of available processors. This would lead to a severe load balancing problem as some processors would be left unoccupied at run time while others would be doing all the work.

The second option would be to partition the angular domain between processors. This would clearly be the best option as our decision to leave the angular elements in our mesh uncoupled by our finite element discretisation will allow parallel sweeps on different angular elements to happen concurrently between processors. This should lead to evenly balanced loading between processors because each sweep covers the same number of spatial elements, even on irregular meshes. The only potential drawback of partitioning by angular elements is if the code was ported onto a machine with many more CPU cores, there would not be enough angular elements to split loading between a large number of processors.

The third option of partitioning the spatial domain would have provided the finest grained parallelism. Decompositions of the spatial domain are more difficult to implement and require more communication between processors. Also, parallel efficiency can be difficult to maintain compared to angular decompositions. However, decompositions of the spatial domain are popular in the literature for the discretisation of large 3D problems where the size of the spatial part of the problem can get very large.

We now state our parallel algorithm for the application of the inverse of $\mathbf{T}$ in a shared memory environment.

Algorithm 3.2.6. (Parallel algorithm for the application of $\mathbf{T}^{-1}$)
For a spatial mesh $\mathcal{T}_S$ and an angular mesh $\mathcal{T}_A$ this algorithm applies the action of $\mathbf{T}^{-1}$ in parallel via a sweeping procedure on each angular element. The algorithm implements the following matrix vector product

$$\mathbf{v_2} \leftarrow \mathbf{T}^{-1}\mathbf{v_1} \tag{3.2.48}$$

and proceeds as follows:

Allocate shared data structures
Compute shared data
BEGIN PARALLEL REGION
Allocate private data structures

Threads	Speed up		
	$(\mathbf{T}-\mathbf{S})^{-1}\mathbf{F}$	$\mathbf{T}$	$\mathbf{T}^{-1}$
2	1.956	1.9806	1.9536
4	1.9829	1.9768	1.9948
8	1.897	1.9281	1.9329
16	1.8003	1.8653	1.8829

Table 3.5: This table gives the rate of speed up given by doubling the number of threads to the value in the leftmost column. These are for the application of the matrices $(\mathbf{T}-\mathbf{S})^{-1}\mathbf{F}$, $\mathbf{T}$ and $\mathbf{T}^{-1}$ and are computed from the same data as Figure 3.4. For the application of $\mathbf{T}^{-1}$ the speed up is close to perfect for up to 8 threads.

BEGIN DYNAMICALLY SCHEDULED PARALLEL LOOP
for $\kappa_A \in \mathcal{T}_A$ ***do***
Get data for κ_A
Build $G = (V, E)$ *for* $\mathcal{T}_S$ *and* κ_A
Call Algorithm 3.2.3 to get strongly connected components for G *and* κ_A
for $g = 1, \cdots, G$ ***do***
Perform sweep using Algorithm 3.2.5
end for
end for
END PARALLEL LOOP
END PARALLEL REGION □

This algorithm was implemented using OpenMP and then each of the various subroutines from the four solution algorithms were benchmarked against each other incorporating the parallel version of the product by $\mathbf{T}^{-1}$. Performing these tests revealed that the application of $\mathbf{T}$ had overtaken the application of $\mathbf{T}^{-1}$ as the most time consuming operation. An OpenMP implementation of the application of $\mathbf{T}$ was written in order to tackle this new bottleneck. The parallel implementation of the action of $\mathbf{T}$ was based on Algorithm 3.2.6, but with the construction of the graph and use of Tarjan's strongly connected components algorithm removed. The sweeping procedure was replaced with a loop over spatial elements and the use of the BLAS subroutine `DGEMV` to perform a matrix vector product on the individual blocks.

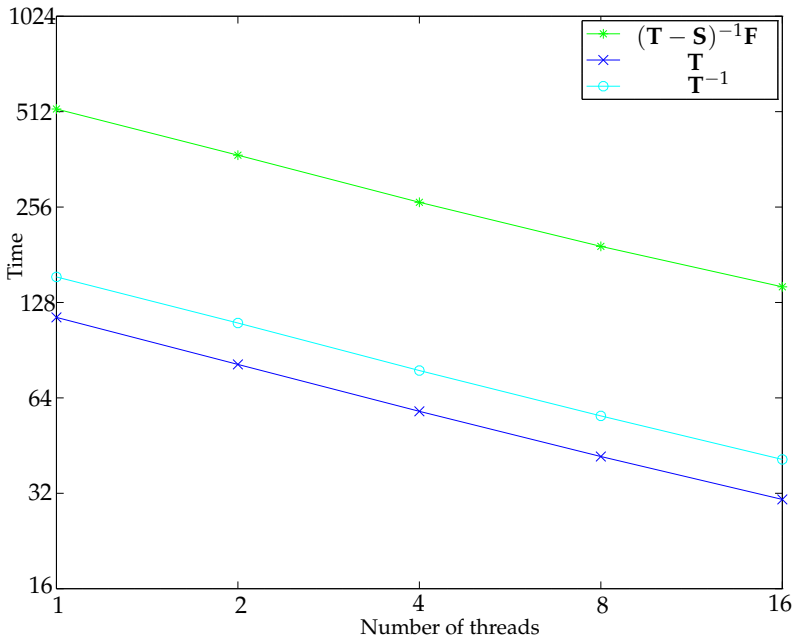


Figure 3.4: A log-log plot of the time taken to apply the action of three matrices against the number of OpenMP threads used. The matrix size was $N = 294\,912$ split into 144 by 144 blocks. The machine incorporated two Intel® Xeon® E5-2665 8-core processors (32 logical cores total) and 512Gb of RAM. We observe good speed up when increasing the number of threads up to 16, which is the number of physical cores present. The application of $(\mathbf{T}-\mathbf{S})^{-1}\mathbf{F}$ involves the repeated application of $\mathbf{F}$, $\mathbf{S}$, $\mathbf{T}$ and $\mathbf{T}^{-1}$. As $\mathbf{T}^{-1}$ is the most computationally challenging of these matrices to apply we observe that the performance of the application of $(\mathbf{T}-\mathbf{S})^{-1}\mathbf{F}$ degrades in the same way that it does for the application of $\mathbf{T}^{-1}$ when the limits of the machine are reached. Exact figures for the speed up are given in Table 3.5.

With this version of the code a set of tests were completed to examine the performance of the parallel code as more threads were added. The problem considered was the seventh criticality benchmark problem from the set compiled by Sood, Forster, and Parsons in [106]. For the discretisation we set the order of approximation in space to $p = 3$ and the order of approximation in angle to $q = 2$. A total of 2048 space-angle elements were used and the matrix size was $N = 294\,912$. Timings were taken for the application of $\mathbf{T}$ and $\mathbf{T}^{-1}$, as well as $(\mathbf{T} - \mathbf{S})^{-1}\mathbf{F}$ which requires the repeated application of both $\mathbf{T}$ and $\mathbf{T}^{-1}$. Figure 3.4 and Table 3.5 present the results.

These computations were completed on a machine incorporating 16 physical cores, or 32 logical cores. When increasing the number of threads up to eight we see a speed up factor close to the optimal factor of two for the application of $\mathbf{T}^{-1}$ and $\mathbf{T}$. This is very encouraging as it is these operations which provide the bottleneck in all of our computations. When doubling from 8 cores to 16 however, the speed up drops to 1.8. We believe that this is the result of congestion of memory access when all physical cores are in operation at the same time. A reduction in the speed up for $\mathbf{T}$ is also observed, though it is not as severe because the application of the transport matrix does not require as much memory as the application of its inverse.

The parallel timing data for the computation of the application of $(\mathbf{T} - \mathbf{S})^{-1}\mathbf{F}$ is a good indicator of the overall performance of the algorithm. This is because it not only incorporates the two most demanding operations but also many of the sequential operations that need to be used in every iteration. The speed up factor of 1.8 for the application of this operator when doubling the number of processors corresponds closely to the overall speed up of the eigenvalue algorithm.

3.3 Implementation

The algorithms described in this thesis were implemented using a mixture of existing academic software libraries and newly developed code. In particular, the AptoFEM [107] finite element software was used to provide data structures for the storage of 2 dimensional meshes and solutions. Furthermore, extensive use was made of AptoFEM's routines for the generation of 2 dimensional basis functions, integration in the spatial domain and spatial mesh refinement. AptoFEM routines were also used for analyticity testing of finite element solutions, which will be discussed in the next chapter.

Other libraries that were used include the CERFACS GMRES library [98], the ARPACK eigenvalue code [104], MA41, MA57 and MC13 [86] as well as GotoBLAS [108] and the Triangle [109] 2 dimensional unstructured mesh generation software.

All other aspects of the software were originally implemented. This includes, but is not limited to, all of the code necessary to extend the aforementioned libraries to be applicable to the full 4 dimensional multienergetic problem as well as all of the mesh generation, integration and refinement routines in the angular domain. Furthermore, the routines necessary for the application of the matrices $\mathbf{T}$, $\mathbf{T}^{-1}$, $\mathbf{F}$ and $\mathbf{S}$ to a vector, the generation of structured 2 dimensional spatial meshes and the incorporation of problem data from the benchmark problems were all written by the author. All of the code necessary for the computation of global and local error estimators was also originally implemented. This includes routines for the computation of L_2-projections in 4 dimensions which are utilised for the restriction of the error estimators to the spatial and angular variables, as well as the computation of *a-posteriori* error representation formulas for the error in the critical eigenvalue.

3.4 Convergence results

In order to test the rates of convergence of the proposed DG finite element discretisation we consider a simple test problem for which the analytical solution is known. In order to obtain such a problem, we first specify an analytical value of the scalar flux at each point in the space-angle domain and then use this to determine the value of an artificial forcing function to be included in the equation to be discretised.

Consider a two dimensional monoenergetic neutron transport source problem defined on the following spatial domain

$$\mathcal{D} = \left[0, \frac{\pi}{2}\right]^2. \tag{3.4.1}$$

We specify that the analytical value for the angular flux is to be given by the following function:

$$\psi(\mathbf{r}, \boldsymbol{\Omega}) = xy\cos(x)\cos(y)\sin^2(\varphi)\sin^2(\theta). \tag{3.4.2}$$

Thus the analytical value for the scalar flux is given by

$$\phi(\mathbf{r}) = xy\cos(x)\cos(y)\frac{2\pi}{3}. \tag{3.4.3}$$

We also define the space averaged flux, which allows us to visualise the angular distribution of the solution by

$$\zeta(\boldsymbol{\Omega}) = \int_{\mathcal{D}} \psi(\mathbf{r}, \boldsymbol{\Omega})\, d\mathbf{r} = \frac{(\pi - 2)^2}{4} \sin(\varphi)^2 \sin(\theta)^2. \tag{3.4.4}$$

The forcing function is easily computed by substituting (3.4.2) into the neutron transport source problem. The source problem prescribed by this spatial domain and forcing function was solved on a series of uniformly refined meshes in order to investigate the order of convergence of the underlying DG finite element method.

3.4.1 Spatial convergence

When the finite element method presented in this chapter had been implemented tests were run to verify the rates of convergence achieved. As the spatial part of the discretisation is a standard higher order DG method for a first order hyperbolic problem the theoretical convergence rate is well known. In [110] Johnson and Pitkäranta proved that the error will decay at $\mathcal{O}(h^{p+\frac{1}{2}})$ in the L_2-norm and in [9] Peterson showed that this rate of convergence cannot be improved upon, even for problems with smooth exact solutions, without making some assumptions on the regularity of the finite element mesh being used. In [11] Houston et al. proved a similar bound in the DG-norm and displayed numerics demonstrating the exponential convergence at this rate that can be achieved for sufficiently analytical test problems. Finally in [111] Cockburn et al. showed that for a mesh of simplexes, each with a unique outflow face, the standard higher order DG method will converge like $\mathcal{O}(h^{p+1})$ in the L_2 norm. It is this rate of convergence that we demonstrate in the results collected here. In order to test specifically the convergence for the spatial scheme, an angular scheme with a fine mesh and high order of approximation was selected in order to render the angular contribution to the total error negligible. An initial spatial mesh of 4 elements was chosen and then uniformly refined. Table 3.6 contains the results and shows that the asymptotic order of convergence achieved by the present code matches the theoretical value.

3.4.2 Angular convergence

In contrast to the spatial part of the discretisation there is no literature on the use of a higher order DG finite element method for the approximation of the angular part of

Spatial mesh	Number of elements	$\|\|\phi(\mathbf{r}) - \phi_h(\mathbf{r})\|\|_{L_2(\mathcal{D})}$	$\|\|\zeta(\boldsymbol{\Omega}) - \zeta_h(\boldsymbol{\Omega})\|\|_{L_2(S_2)}$	Order of convergence in scalar flux
$p = 0$ spatial approximation, $q = 4$ angular approximation				
1	4	0.3558640	0.1566136	-
2	16	0.1634107	7.8206420E-02	1.1228
3	64	8.3881803E-02	4.1815333E-02	0.9621
4	256	4.3445326E-02	2.1997901E-02	0.9492
5	1 024	2.2260459E-02	1.1337632E-02	0.9647
6	4 096	1.1289563E-02	5.7635941E-03	0.9795
$p = 1$ spatial approximation, $q = 4$ angular approximation				
1	4	7.9475768E-02	1.2850358E-02	-
2	16	1.9341335E-02	1.7416134E-03	2.0388
3	64	4.7903312E-03	2.3518868E-04	2.0135
4	256	1.1943334E-03	5.9661892E-05	2.0039
5	1 024	2.9836851E-04	5.1859213E-05	2.0010
$p = 2$ spatial approximation, $q = 5$ angular approximation				
1	4	5.3683207E-03	1.4575111E-04	-
2	16	6.8253360E-04	7.9472275E-06	2.9755
3	64	8.5342872E-05	4.5220477E-07	2.9996
4	256	1.0659324E-05	2.5583245E-07	3.0012

Table 3.6: The error under spatial h-refinement. The angular schemes were chosen to to be very high order ($q = 4$ or 5) to render the error in the angular discretisation negligible. These results indicate that the spatial scheme converges at $\mathcal{O}(h^{p+1})$.

neutron transport problems. The only other attempt at a higher order DG approximation for the angular variables in neutron transport problems was by Bennison in [43]. However Bennison first projected the angular domain from the unit hemisphere onto the unit disk, therefore a factor of $\cos^{-1}(\varphi)$ needed to be included in all of his calculations. The inclusion of this factor, which introduces a coordinate singularity at the boundary of his angular domain, meant that he was unable to collect accurate data on the convergence of the angular part of his approximation.

Though there has been no analysis published for a higher order DG method in angle for neutron transport, there have been theoretical results published on the convergence of the discrete ordinates method. As the DG method considered here reduces to a scheme similar to a discrete ordinates method when the order of approximation is $q = 0$, we can use these results to confirm the precision of the implementation considered here, at least for a scheme with a constant approximation on each angular element. In [8] Johnson and Pitkäranta prove an error bound for the error in the scalar flux for a version of the neutron transport problem where the angular domain is restricted to the equator of S_2. This result was improved upon by Asadzadeh in [112] who proved an equivalent error bound for a discrete ordinates approximation on S_2. To prove this he used a discrete ordinates quadrature defined on the projection of S_2 onto a disk. This quadrature comprised $N_O = nm$ points, with m points in the azimuthal direction and n points in the radial (polar on S_2) direction. When the spatial error is small this error bound implies convergence of at least $\mathcal{O}(n^{-1} + m^{-1})$ in the angular variables. This discrete ordinates method can be written like a rectangular partitioned finite element mesh on S_2, with angular element diameters of $h \sim m^{-1}$. Then if $n = m$ we would expect convergence of at least $\mathcal{O}(h)$.

To test the order of convergence of the angular discretisation a series of progressively finer angular meshes was defined, each comprising angular elements of equal area. A spatial mesh was chosen which was sufficiently fine to render the spatial contribution to the error negligible compared to the angular error and the system was solved using finite element spaces of increasing order on the series of angular meshes. The results are given in Table 3.7. Note that for the $q = 0$ problem the rate of convergence is computed as $\mathcal{O}(h^2)$, which is better than the more pessimistic $\mathcal{O}(h)$ convergence predicted by the analysis of Asadzadeh.

Angular mesh	Number of elements	$\|\|\phi(\mathbf{r}) - \phi_h(\mathbf{r})\|\|_{L_2(\mathcal{D})}$	$\|\|\zeta(\boldsymbol{\Omega}) - \zeta_h(\boldsymbol{\Omega})\|\|_{L_2(S_2)}$	Order of convergence in scalar flux
$q = 0$ angular approximation				
1	4	9.6385024E-02	1.8744055E-02	-
2	16	2.4848044E-02	5.1731984E-03	1.9557
3	64	6.3705621E-03	1.4139627E-03	1.9636
4	256	1.6160379E-03	3.7319103E-04	1.9790
5	1024	4.0717906E-04	9.6241070E-05	1.9887
6	4096	1.0217800E-04	2.4444935E-05	1.9946
$q = 1$ angular approximation				
1	4	3.1649128E-03	3.2750253E-02	-
2	16	1.6101586E-03	2.1599442E-02	0.9750
3	64	3.4734685E-04	7.2245649E-03	2.2128
4	256	6.4424865E-05	2.5444836E-03	2.4307
5	1024	8.8051283E-06	9.0981479E-04	2.8712
6	4096	1.2965155E-06	3.2566828E-04	2.7637
$q = 2$ angular approximation				
1	4	3.9649173E-03	3.0746141E-02	-
2	16	1.3688076E-04	4.3400726E-03	4.8563
3	64	8.6517275E-06	7.3444797E-04	3.9838
4	256	9.1425471E-07	1.2617772E-04	3.2423

Table 3.7: The error under angular h-refinement. The spatial scheme was chosen to to be very high order ($p = 5$) to render the error in the spatial discretisation negligible. These results indicate that the angular scheme converges at $\mathcal{O}(h^2)$ for a $q = 0$ approximation.

CHAPTER 4

Adaptive Algorithms for Neutron Transport Criticality Problems

In the previous chapter we presented a high order *hp*-version discontinuous-Galerkin (DG) finite element discretisation for neutron transport criticality problems, as well as an efficient parallel solution algorithm for the resulting generalised eigenvalue problem. However the high dimensionality of problems arising in neutronics results in very large systems of equations that are extremely expensive to solve, even when employing a higher order numerical method and an efficient solution algorithm. Furthermore, once these calculations have been completed it will not be possible to quantify the size of the error in the computed eigenvalue.

In this chapter we will discuss how these deficiencies may be overcome using an *a posteriori* error estimator together with an adaptive mesh refinement algorithm. For an introduction to *a posteriori* methods we direct the reader to the reviews by Verfürth [113], and Ainsworth and Oden [114], as well as the book by Szabo and Babuška [115], and the article by Becker [116]. We shall present both an *h*-adaptive, as well as an *hp*-adaptive refinement algorithm for the numerical approximation of neutron transport k_{eff}-eigenvalue problems. These algorithms both feature a dual weighted residual (DWR) based *a posteriori* error representation formula, derived in Section 4.2, which provides reliable estimates of the error in the dominant eigenvalue. Both the *h*- and *hp*-refinement algorithms exploit a projection of the error onto the spatial degrees of freedom so that the relative contribution of the spatial and angular portions of the error may be quantified in order to balance the computational resources between the two computational domains. In Section 4.3 we present the *h*-refinement algorithm; here

we demonstrate the efficiency gains that can be attained compared to a more naive refinement strategy. In Section 4.4 we present the *hp*-refinement algorithm and include results which demonstrate the exponential convergence of this approach with respect to the number of degrees of freedom employed.

4.1 Adaptive mesh refinement for the neutron transport equation

In this thesis we are concerned with deriving a computable *a posteriori* estimate on the error committed by our DG finite element method in order to design an automatic adaptive mesh refinement algorithm. We elect to employ the dual weighted residual method which provides an estimate of the error in a given target functional of physical interest as opposed to a global norm of the error over the whole domain. This enables us to identify the regions in the computational domain that contribute most significantly to the error measured with respect to our quantity of interest. Thereby, only these regions will be refined, thus avoiding excessive computation in regions that have little effect on the target functional of interest. See [117] by Bangerth and Rannacher for a detailed description of the DWR method.

The DWR method has been applied to a wide variety of finite element methods, including continuous and discontinuous Galerkin methods for a range of partial differential equations, such as elliptic and hyperbolic equations, as well as for linear and non-linear conservation laws. Most relevant to the present work is the analysis of *a posteriori* methods for first order hyperbolic problems by Houston and Süli in [118], the application to nonlinear hyperbolic conservation laws by Hartmann et al. in [82], and the application to eigenvalue problems in incompressible flow problems in [119] by Cliffe et al., as well as to bifurcation problems in [120], [121] and [122], also by Cliffe and collaborators.

Such methods have recently received considerable interest in the fields of neutron transport; indeed DWR error estimation has been employed by several authors to control the error arising from the spatial discretisation of neutron transport problems. In particular Duo et al., in [123], followed on from the work of Houston and Süli [118] and implemented an adaptive algorithm for the spatial part of the S_N equations, controlling the error measured by the L_2-norm over the spatial domain. Fournier developed a similar algorithm for the nodal transport method in [124], again for an S_N angular dis-

cretisation and with error of the scalar flux measured in the L_2-norm. In [34] Kanschat applies a DWR method to obtain a mesh refinement algorithm for radiative transfer problems in astrophysics. In [125] Lathouwers employed a DWR approach to develop an adaptive spatial mesh refinement algorithm for the neutron transport source problem, controlling the error in a variety of functionals of practical interest. Furthermore, in [126] Lathouwers employed a DWR method, similar to the one derived in the next section, to control the spatial contribution to the error in the k_{eff}-eigenvalue, as well as developing an adaptive mesh refinement algorithm in the spatial variables.

Finally, in [65], Merton et al. employed an analysis based on a Taylor series expansion of the residual functional in order to obtain a spatial correction to the computed primal eigenvalue. Their error recovery formula utilises a convolution of the primal residual with a higher order dual eigenpair computed on the same spatial mesh, this is then subtracted from the computed primal eigenvalue. It was shown to accelerate convergence of the eigenvalue for both the neutron diffusion equation as well as the full neutron transport equation.

We remark that until now there has been very little attention devoted to *a posteriori* error estimation for the full space-angle discretisation of the neutron transport problem. Moreover, the work presented in this chapter represents the first attempt to develop *hp*-adaptive methods for neutron transport problems. In the following section we derive an error representation formula which provides a computable estimate of the error in the k_{eff}-eigenvalue. We then proceed to consider projections between finite element spaces in order to quantify the relative contribution to the error in the eigenvalue from each of the spatial and angular discretisations. We conclude the chapter by considering adaptive mesh refinement algorithms in both space and angle, as well as an *hp*-refinement algorithm.

4.2 An *a posteriori* error estimator for criticality computations

In this section we present a computable *a posteriori* error representation formula for the DG approximation of neutron transport criticality problems, which is derived in the book by Bangerth and Rannacher, [117]. In order to derive the error representation formula we first write equation (3.2.14) in terms of the bilinear forms $(T - S)(\cdot, \cdot)$ and $F(\cdot, \cdot)$. Here, $(T - S)(\cdot, \cdot)$ contains terms for the streaming, absorption, scattering and

boundary conditions and is defined by

$$\begin{aligned}(T-S)(\psi_h, v_h) = &\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S} -\psi_h(\mathbf{r},\boldsymbol{\Omega})\boldsymbol{\Omega}\cdot\nabla v_h(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega}\\ &+\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\partial\kappa_S\setminus\Gamma}\mathcal{H}(\psi_h^+(\mathbf{r},\boldsymbol{\Omega}),\psi_h^-(\mathbf{r},\boldsymbol{\Omega}),\mathbf{n}_{\kappa_S},\boldsymbol{\Omega})v_h^+(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega}\\ &+\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\partial\kappa_S\cap\Gamma_+}(\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S})\psi_h(\mathbf{r},\boldsymbol{\Omega})v_h(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega}\\ &+\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\kappa_S}\Sigma_t(\mathbf{r})\psi_h(\mathbf{r},\boldsymbol{\Omega})v_h(\mathbf{r},\boldsymbol{\Omega})\,d\mathbf{r}\,d\boldsymbol{\Omega}\\ &+\sum_{\kappa_A\in\mathcal{T}_A}\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_A}\int_{\partial\kappa_S\cap\Gamma_-}(\boldsymbol{\Omega}\cdot\mathbf{n}_{\kappa_S})\hat{g}(\mathbf{r},\boldsymbol{\Omega})v_h(\mathbf{r},\boldsymbol{\Omega})\,ds\,d\boldsymbol{\Omega}\\ &-\sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_S}\frac{\Sigma_s(\mathbf{r})}{4\pi}\left(\sum_{\kappa_A\in\mathcal{T}_A}\int_{\kappa_A}\psi_h(\mathbf{r},\boldsymbol{\Omega})\,d\boldsymbol{\Omega}\right)\left(\sum_{\kappa_A'\in\mathcal{T}_A}\int_{\kappa_A'}v_h(\mathbf{r},\boldsymbol{\Omega}')\,d\boldsymbol{\Omega}'\right)d\mathbf{r}.\end{aligned} \tag{4.2.1}$$

The fission terms are given by,

$$F(\psi_h, v_h) = \sum_{\kappa_S\in\mathcal{T}_S}\int_{\kappa_S}\frac{\nu\Sigma_f(\mathbf{r})}{4\pi}\left(\sum_{\kappa_A\in\mathcal{T}_A}\int_{\kappa_A}\psi_h(\mathbf{r},\boldsymbol{\Omega})\,d\boldsymbol{\Omega}\right)\left(\sum_{\kappa_A'\in\mathcal{T}_A}\int_{\kappa_A'}v_h(\mathbf{r},\boldsymbol{\Omega}')\,d\boldsymbol{\Omega}'\right)d\mathbf{r}. \tag{4.2.2}$$

With this notation, the DG approximation for the monoenergetic neutron transport k_{eff}-eigenvalue problem with isotropic scattering is given by: find $(k_{\text{eff},h}, \psi_h) \in \mathbb{R}\times\mathcal{V}_h^{p,q}$ such that,

$$k_{\text{eff},h}\,(T-S)(\psi_h, v_h) = F(\psi_h, v_h) + k_{\text{eff},h}^v(1-C(\psi_h)) \tag{4.2.3}$$

for all $(k_{\text{eff},h}^v, v_h) \in \mathbb{R}\times\mathcal{V}_h^{p,q}$. Here we have introduced the nonlinear functional, $C(\cdot)$, which is chosen so that $C(\psi) = C(\psi_h) = 1$, and an analogue to the eigenvalue for the test function, $k_{\text{eff},h}^v$, which will become the eigenvalue for the dual problem. We note that $C(\cdot)$ may be chosen to be any nonlinear functional, up to a scaling, such that $C(\psi_h) \neq 0$ and $C(\psi) \neq 0$. For the present numerical implementation we take the value of $C(\psi_h)$ to be the same as $\|\boldsymbol{\psi}\|_2$, where $\boldsymbol{\psi}$ is the vector of coefficients of the basis functions of ψ_h, as defined in Chapter 3.

Writing $\underline{v}_h = (k_{\text{eff},h}^v, v_h)$ and the DG solution eigenpair as $\underline{\psi}_h = (k_{\text{eff},h}^\psi, \psi_h)$, where $k_{\text{eff},h}^\psi = k_{\text{eff},h}$, we introduce the semilinear functional

$$\mathcal{N}\left(\underline{\psi}_h, \underline{v}_h\right) = k_{\text{eff},h}^\psi\,(T-S)(\psi_h, v_h) - F(\psi_h, v_h) + k_{\text{eff},h}^v(C(\psi_h)-1). \tag{4.2.4}$$

Thereby, the primal DG finite element problem may be written in the following equivalent form: find $\underline{\psi}_h \in \mathbb{R}\times\mathcal{V}_h^{p,q}$ such that,

$$\mathcal{N}\left(\underline{\psi}_h, \underline{v}_h\right) = 0 \tag{4.2.5}$$

for all $\underline{v}_h \in \mathbb{R} \times \mathcal{V}_h^{p,q}$.

If the analytical eigenpair is given by $\underline{\psi} = (k_{\text{eff}}^{\psi}, \psi)$ and if the eigenfunction ψ is sufficiently smooth, then the consistency property of the numerical flux implies that

$$\mathcal{N}\left(\underline{\psi}, \underline{v}_h\right) = 0, \tag{4.2.6}$$

for all $\underline{v}_h \in \mathbb{R} \times \mathcal{V}_h^{p,q}$. The Galerkin orthogonality property for the DG method defined by this weak problem is expressed as follows

$$\mathcal{N}\left(\underline{\psi}, \underline{v}_h\right) - \mathcal{N}\left(\underline{\psi}_h, \underline{v}_h\right) = 0 \tag{4.2.7}$$

for all $\underline{v}_h \in \mathbb{R} \times \mathcal{V}_h^{p,q}$.

Before we consider the *a posteriori* error estimator for the critical eigenvalue, we first outline the general framework for deriving a dual weighted residual (DWR) error representation formula for some target functional, $J(\cdot)$, of physical interest. Assuming $J(\cdot)$ is differentiable we define the mean value linearisation of $J(\cdot)$ between $\underline{\psi}$ and $\underline{\psi}_h$ by

$$\begin{aligned} \overline{J}\left(\underline{\psi}, \underline{\psi}_h; \underline{\psi} - \underline{\psi}_h\right) &= J\left(\underline{\psi}\right) - J\left(\underline{\psi}_h\right) \\ &= \int_0^1 J'\left[\chi\underline{\psi} + (1-\chi)\underline{\psi}_h\right]\left(\underline{\psi} - \underline{\psi}_h\right) d\chi, \end{aligned} \tag{4.2.8}$$

where $J'[\underline{w}](\cdot)$ denotes the Frèchet derivative evaluated at some $\underline{w} \in \mathbb{R} \times \mathcal{V}$, where $\mathcal{V} \supset \mathcal{V}_h^{p,q}$ is a suitably chosen space. Similarly, we write the mean value linearisation of the semilinear functional $\mathcal{N}(\cdot,\cdot)$ as

$$\begin{aligned} \mathcal{M}\left(\underline{\psi}, \underline{\psi}_h; \underline{\psi} - \underline{\psi}_h, \underline{v}\right) &= \mathcal{N}\left(\underline{\psi}, \underline{v}\right) - \mathcal{N}\left(\underline{\psi}_h, \underline{v}\right) \\ &= \int_0^1 \mathcal{N}'\left[\chi\underline{\psi} + (1-\chi)\underline{\psi}_h\right]\left(\underline{\psi} - \underline{\psi}_h, \underline{v}\right) d\chi, \end{aligned} \tag{4.2.9}$$

for all $\underline{v} \in \mathbb{R} \times \mathcal{V}$, where $\mathcal{N}'[\underline{w}](\cdot, \underline{v})$ is the Frèchet derivative in the first argument for some fixed $\underline{v} \in \mathbb{R} \times \mathcal{V}$, evaluated at $\underline{w} \in \mathbb{R} \times \mathcal{V}$. We define the formal dual problem as: find $\underline{z} \in \mathbb{R} \times \mathcal{V}$ such that

$$\mathcal{M}\left(\underline{\psi}, \underline{\psi}_h; \underline{v}, \underline{z}\right) = \overline{J}\left(\underline{\psi}, \underline{\psi}_h; \underline{v}\right) \tag{4.2.10}$$

for all $\underline{v} \in \mathbb{R} \times \mathcal{V}$.

Then the error in $J(\cdot)$ is given by

$$J\left(\underline{\psi}\right) - J\left(\underline{\psi}_h\right) = \overline{J}\left(\underline{\psi}, \underline{\psi}_h; \underline{\psi} - \underline{\psi}_h\right) \tag{4.2.11}$$

$$= \mathcal{M}\left(\underline{\psi}, \underline{\psi}_h; \underline{\psi} - \underline{\psi}_h, \underline{z}\right) \tag{4.2.12}$$

$$= \mathcal{N}\left(\underline{\psi}, \underline{z}\right) - \mathcal{N}\left(\underline{\psi}_h, \underline{z}\right). \tag{4.2.13}$$

Given $\underline{z}_h \in \mathbb{R} \times \mathcal{V}_h^{p,q}$, exploiting Galerkin orthogonality we obtain

$$J\left(\underline{\psi}\right) - J\left(\underline{\psi}_h\right) = \mathcal{N}\left(\underline{\psi}, \underline{z} - \underline{z}_h\right) - \mathcal{N}\left(\underline{\psi}_h, \underline{z} - \underline{z}_h\right). \tag{4.2.14}$$

Employing the consistency property of the DG method we obtain the following error representation formula

$$J\left(\underline{\psi}\right) - J\left(\underline{\psi}_h\right) = -\mathcal{N}\left(\underline{\psi}_h, \underline{z} - \underline{z}_h\right) \tag{4.2.15}$$

for all $\underline{z}_h \in \mathbb{R} \times \mathcal{V}_h^{p,q}$. To obtain an error representation formula for $k_{\text{eff},h}^{\psi}$ we set

$$J\left(\underline{\psi}\right) = k_{\text{eff}}^{\psi} C\left(\psi\right). \tag{4.2.16}$$

Then the error in $J(\cdot)$ between the true solution and the DG approximation is given by

$$J\left(\underline{\psi}\right) - J\left(\underline{\psi}_h\right) = k_{\text{eff}}^{\psi} C\left(\psi\right) - k_{\text{eff},h}^{\psi} C\left(\psi_h\right) = k_{\text{eff}}^{\psi} - k_{\text{eff},h}^{\psi}. \tag{4.2.17}$$

Assuming that $J(\cdot)$ is sufficiently smooth around $\underline{\psi}$, then for small $\underline{\psi} - \underline{\psi}_h$ we can make the following approximation

$$\overline{J}\left(\underline{\psi}, \underline{\psi}_h; \underline{\psi} - \underline{\psi}_h\right) \approx J'\left[\underline{\psi}\right]\left(\underline{\psi} - \underline{\psi}_h\right). \tag{4.2.18}$$

Analogously assuming that $\mathcal{N}(\cdot,\cdot)$ is sufficiently smooth, then for small $\underline{\psi} - \underline{\psi}_h$ we can make the following approximation at $\underline{\psi}$

$$\mathcal{M}\left(\underline{\psi}, \underline{\psi}_h; \underline{\psi} - \underline{\psi}_h, \underline{v}\right) \approx \mathcal{N}'\left[\underline{\psi}\right]\left(\underline{\psi} - \underline{\psi}_h, \underline{v}\right), \tag{4.2.19}$$

for all $\underline{v} \in \mathbb{R} \times \mathcal{V}$. Then, in place of the formal dual problem (4.2.10), we consider the following approximate dual problem: find $\underline{z} \in \mathbb{R} \times \mathcal{V}$ such that

$$\mathcal{N}'\left[\underline{\psi}\right](\underline{v}, \underline{z}) = J'\left[\underline{\psi}\right](\underline{v}), \tag{4.2.20}$$

for all $\underline{v} \in \mathbb{R} \times \mathcal{V}$. We note that the $\underline{z}$ considered here is not the same as the $\underline{z}$ from the formal dual problem, however the same symbol is employed for notational convenience. By differentiating $J(\cdot)$ and $\mathcal{N}(\cdot)$, (4.2.20) may be written in the following form: find $\underline{z} \in \mathbb{R} \times \mathcal{V}$ such that

$$k_{\text{eff}}^{v} C(\psi) + k_{\text{eff}}^{\psi} C'[\psi](v) = k_{\text{eff}}^{v}(T - S)(\psi, z) + k_{\text{eff}}^{\psi}(T - S)(v, z) - F(v, z) + k_{\text{eff}}^{z} C'[\psi](v) \tag{4.2.21}$$

for all $\underline{v} \in \mathbb{R} \times \mathcal{V}$. If the dual eigenfunction z is normalised so that $(T - S)(\psi, z) = C(\psi) = 1$ then (4.2.21) may be written in the following equivalent form: find $\underline{z} \in \mathbb{R} \times \mathcal{V}$ such that

$$k_{\text{eff}}^{\psi} C'[\psi](v) = k_{\text{eff}}^{\psi}(T - S)(v, z) - F(v, z) + k_{\text{eff}}^{z} C'[\psi](v). \tag{4.2.22}$$

for all $\underline{v} \in \mathbb{R} \times \mathcal{V}$. Since $F(\cdot,\cdot)$ is symmetric, we note that $k^z_{\text{eff}} = k^\psi_{\text{eff}}$. Thereby, the approximate dual eigenpair satisfies: find $\underline{z} \in \mathbb{R} \times \mathcal{V}$ such that

$$F(v,z) = k^z_{\text{eff}}(T-S)(v,z), \tag{4.2.23}$$

for all $v \in \mathcal{V}$. We may now write down our error representation formula for the primal k_{eff}-eigenvalue as

$$k^\psi_{\text{eff}} - k^\psi_{\text{eff},h} \approx -\mathcal{N}\left(\underline{\psi}_h, \underline{z}\right). \tag{4.2.24}$$

This formula requires knowledge of the adjoint eigenfunction to the continuous eigenproblem (4.2.23). As this will not be known in general we are required to compute a numerical approximation $\hat{z} \in \mathcal{V}_{\hat{h}}^{\hat{p},\hat{q}}$, where $\mathcal{V}_{\hat{h}}^{\hat{p},\hat{q}}$ is the space-angle finite element space over the mesh $\hat{\mathcal{T}}$ of granularity $\hat{h}$ with polynomials of order $\hat{p}$ and $\hat{q}$ in space and angle, respectively. Notice that Galerkin orthogonality necessitates that $\hat{z} \notin \mathcal{V}_h^{p,q}$, otherwise the error representation formula will evaluate identically to zero. Therefore we compute an approximation to z with $\hat{\mathcal{T}} = \mathcal{T}$, $\hat{p} = p+1$ and $\hat{q} = q+1$, i.e. we seek the solution to the following adjoint finite element problem: find $\underline{\hat{z}} \in \mathbb{R} \times \mathcal{V}_h^{p+1,q+1}$ such that

$$F(v_h,\hat{z}) = k^{\hat{z}}_{\text{eff}}(T-S)(v_h,\hat{z}), \tag{4.2.25}$$

for all $v_h \in \mathcal{V}_h^{p+1,q+1}$.

We may write (4.2.25) in matrix form by considering the set of N basis functions, $\{\xi_j\}$, of the finite element space $\mathcal{V}_h^{p+1,q+1}$, where each ξ_j is defined as in Section 3.2.1. Then $\hat{z} \in \mathcal{V}_h^{p+1,q+1}$ is determined by a vector $\boldsymbol{z} \in \mathbb{R}^N$, where

$$\hat{z}(\mathbf{r},\boldsymbol{\Omega}) = \sum_{j=1}^{N} \boldsymbol{z}[j]\,\xi(\mathbf{r},\boldsymbol{\Omega}). \tag{4.2.26}$$

The dual eigenpair can be computed as the solution to: find $\boldsymbol{z} \in \mathbb{R}^N$ and $k^{\hat{z}}_{\text{eff}} \in \mathbb{R}$ such that

$$k^{\hat{z}}_{\text{eff}}(\mathbf{T}-\mathbf{S})^\top \boldsymbol{z} = \mathbf{F}^\top \boldsymbol{z}, \tag{4.2.27}$$

where $\mathbf{T}$, $\mathbf{S}$ and $\mathbf{F}$ are defined as in Section 3.2.1. We remark that, by replacing $\mathbf{T}$, $\mathbf{S}$, $\mathbf{F}$ and $\mathbf{T}^{-1}$ by $\mathbf{T}^\top$, $\mathbf{S}^\top$, $\mathbf{F}^\top$ and $\mathbf{T}^{-\top}$, respectively, this discrete approximate dual eigenproblem may be solved by the same parallel Krylov subspace based method that was developed for the primal problem in Section 3.2.4. Indeed the fact that this solution algorithm was shown to be robust when raising the order of polynomial approximation in the finite element space is crucial for the efficient implementation of the mesh adaptation algorithms presented herein.

Utilising the approximate dual eigenpair, the computable error representation formula for the multiplicative eigenvalue in neutron transport criticality computations is given by

$$k^{\psi}_{\text{eff}} - k^{\psi}_{\text{eff},h} \approx -\mathcal{N}\left(\underline{\psi}_h, \hat{\underline{z}} - \underline{z}_h\right), \tag{4.2.28}$$

for all $\underline{z}_h \in \mathbb{R} \times \mathcal{V}_h^{p,q}$.

Note that this formula cannot be rearranged to provide a correction for the primal DG eigenvalue, as is the strategy of Merton et al., in [65]. Computing an equivalent correction for the error representation formula considered here leads to a corrected primal eigenvalue which is precisely the same as the computed dual eigenvalue $k^{\hat{z}}_{\text{eff}}$. We see this clearly by writing the above formula as

$$k^{\psi}_{\text{eff}} \approx k^{\psi}_{\text{eff},h} - \mathcal{N}\left(\underline{\psi}_h, \hat{\underline{z}} - \underline{z}_h\right) \tag{4.2.29}$$

$$= k^{\psi}_{\text{eff},h} - k^{\psi}_{\text{eff},h}\left(T - S\right)\left(\psi_h, \hat{z}\right) + F(\psi_h, \hat{z}) - k^{\hat{z}}_{\text{eff}}(C(\psi_h) - 1). \tag{4.2.30}$$

Utilising the fact that $C(\psi_h) = \left(T - S\right)\left(\psi_h, \hat{z}\right) = 1$, this becomes

$$k^{\psi}_{\text{eff}} \approx k^{\psi}_{\text{eff},h} - k^{\psi}_{\text{eff},h} + F(\psi_h, \hat{z}) \tag{4.2.31}$$

$$= k^{\hat{z}}_{\text{eff}}\left(T - S\right)\left(\psi_h, \hat{z}\right) \tag{4.2.32}$$

$$= k^{\hat{z}}_{\text{eff}}, \tag{4.2.33}$$

from the definition of the computed dual problem, 4.2.25, and the fact that $\psi_h \in \mathcal{V}_h^{p,q} \subset \mathcal{V}_h^{p+1,q+1}$.

An analogous error representation formula can be computed for the error in $\lambda^{\psi}_h = \frac{1}{k^{\psi}_{\text{eff},h}}$ if we consider the following semilinear functional in place of $\mathcal{N}(\cdot,\cdot)$:

$$\mathcal{N}_{\lambda}\left(\underline{\psi}_h, \underline{v}_h\right) = \left(T - S\right)\left(\psi_h, v_h\right) - \frac{1}{k^{\psi}_{\text{eff},h}}F(\psi_h, v_h) + \frac{1}{k^{v}_{\text{eff},h}}(1 - C(\psi_h)), \tag{4.2.34}$$

with the following functional,

$$J_{\lambda}\left(\underline{\psi}\right) = \frac{1}{k^{\psi}_{\text{eff}}}C\left(\psi\right). \tag{4.2.35}$$

In this case, we then normalise the dual eigenvalue so that $F(\psi, z) = 1$. Effectivities based on the error representation formulae for both the primal eigenvalue and its reciprocal are computed for all test problems.

4.3 *h*-refinement

In this section we employ the error representation formulas derived in the previous section to develop an adaptive mesh refinement (AMR) algorithm for k_{eff}-eigenvalue computations. At each iteration of this mesh refinement algorithm we seek to reduce the total error in the DG approximation to the critical eigenvalue by refining the finite element mesh $\mathcal{T}$ in those parts of the domain that contribute the most error to the computed eigenvalue. As $\mathcal{T}$ results from taking a tensor product between a mesh defined over the spatial domain $\mathcal{T}_S$ and a mesh defined over the angular domain $\mathcal{T}_A$, any mesh refinement algorithm requires a method for quantifying the relative contribution to the error from the spatial and angular parts of the problem in order to decide which to refine.

4.3.1 Space-angle error splitting

We describe a method for writing our error representation formula as a sum between the contributions from the spatial and angular variables respectively. First we write (4.2.15) in terms of a residual function $\text{Res}(\cdot,\cdot)$,

$$J\left(\underline{\psi}\right) - J\left(\underline{\psi}_h\right) = -\mathcal{N}\left(\underline{\psi}_h, \underline{z} - \underline{z}_h\right) \tag{4.3.1}$$

$$= -k^{\psi}_{\text{eff},h}\left(T - S\right)\left(\psi_h, z - z_h\right) + F\left(\psi_h, z - z_h\right) \tag{4.3.2}$$

$$\equiv \text{Res}(\underline{\psi}_h, z - z_h). \tag{4.3.3}$$

We write Π_S and Π_A to denote the L_2-projection operators onto the spatial and angular finite element spaces, $S^p_S(\mathcal{T}_S, F_{\mathcal{T}_S})$ and $S^q_A(\mathcal{T}_A, F_{\mathcal{T}_A})$, respectively, such that $z_h = \Pi_S\Pi_A z$. Then we have

$$J\left(\underline{\psi}\right) - J\left(\underline{\psi}_h\right) = \text{Res}\left(\underline{\psi}_h, z - \Pi_S\Pi_A z\right) \tag{4.3.4}$$

$$= \text{Res}\left(\underline{\psi}_h, z - \Pi_S z + \Pi_S z - \Pi_S\Pi_A z\right) \tag{4.3.5}$$

$$= \text{Res}\left(\underline{\psi}_h, z - \Pi_S z\right) + \text{Res}\left(\underline{\psi}_h, \Pi_S\left(z - \Pi_A z\right)\right). \tag{4.3.6}$$

This provides a splitting between the spatial and angular parts of the discretisation error, respectively. We can then write our computable error representation formula (4.2.28) as,

$$k^{\psi}_{\text{eff}} - k^{\psi}_{\text{eff},h} \approx \eta_S + \eta_A, \tag{4.3.7}$$

where the spatial error indicator is defined as

$$\eta_S = \text{Res}\left(\underline{\psi}_h, \hat{z} - \Pi_S \hat{z}\right), \tag{4.3.8}$$

and the angular error indicator is given by

$$\eta_A = \text{Res}\left(\underline{\psi}_h, \Pi_S\left(\hat{z} - \Pi_A \hat{z}\right)\right). \tag{4.3.9}$$

Note that the angular projection does not need to be computed explicitly, since $\eta_A = \text{Res}\left(\underline{\psi}_h, \Pi_S \hat{z}\right)$ due to Galerkin orthogonality. Furthermore we define the elementwise spatial error indicators as,

$$\eta_{\kappa_S} = \text{Res}\left(\underline{\psi}_h, \hat{z} - \Pi_S \hat{z}\right)\Big|_{\kappa_S} \tag{4.3.10}$$

for all $\kappa_S \in \mathcal{T}_S$, and the elementwise angular error indicators as

$$\eta_{\kappa_A} = \text{Res}\left(\underline{\psi}_h, \Pi_S(\hat{z} - \Pi_A \hat{z})\right)\Big|_{\kappa_A} \tag{4.3.11}$$

for all $\kappa_A \in \mathcal{T}_A$. In order to verify the accuracy of our *a posteriori* error estimator we introduce the effectivity index I_{eff}, which is the ratio between the true error in the eigenvalue and the value of the computed error representation formula, i.e.,

$$I_{\text{eff}} = \frac{\text{Res}\left(\underline{\psi}_h, \hat{z}\right)}{k^{\psi}_{\text{eff}} - k^{\psi}_{\text{eff},h}}. \tag{4.3.12}$$

An analogous definition holds when the error representation formula is computed for the reciprocal of k^{ψ}_{eff}.

4.3.2 Algorithm development

The following adaptive algorithms considered for k_{eff}-eigenvalue calculations follow the same basic structure. We begin with initial coarse spatial and angular meshes, then at each iteration we compute the primal and dual eigenpairs, $\underline{\psi}_h$ and $\underline{\hat{z}}$, respectively. Thereby $\text{Res}\left(\underline{\psi}_h, \hat{z}\right)$, η_S, η_A and all of the elementwise error indicators, η_{κ_S} and η_{κ_S} may be computed. If $\left|\text{Res}\left(\underline{\psi}_h, \hat{z}\right)\right|$ is less than a prescribed convergence tolerance the algorithm terminates. Otherwise, the spatial and angular meshes and finite element spaces undergo refinement (and/or derefinement) based on the computed error indicators and the next iteration is then performed.

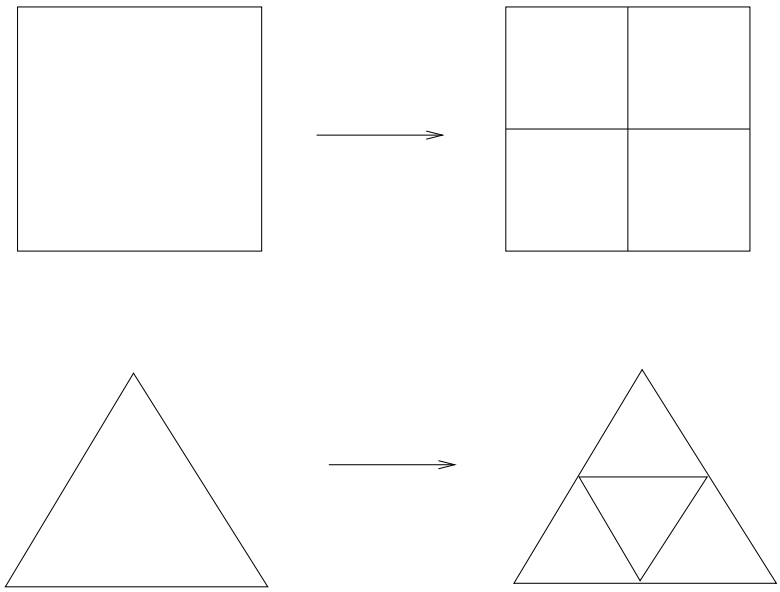


Figure 4.1: The process of isotropic refinement divides an element into four daughter elements by introducing a new mesh node at the centre of each face. This is illustrated for rectangular and triangular elements.

The primary difference between the algorithms is how the error indicators are employed to determine which elements in the spatial and angular meshes are refined. We consider two possible strategies for deciding which of the elements will be refined and which will be derefined. The first strategy is, at each iteration, to refine a proportion of the spatial elements where $|\eta_{\kappa_S}|$ is large and a proportion of the angular elements where $|\eta_{\kappa_A}|$ is large. These proportions are determined by dividing the 'total' amount of refinement α^{FF}, between the spatial and angular meshes. In particular, writing

$$\alpha_S^{\mathrm{FF}} = \frac{\alpha^{\mathrm{FF}} |\eta_S|}{|\eta_S| + |\eta_A|}, \tag{4.3.13}$$

and

$$\alpha_A^{\mathrm{FF}} = \frac{\alpha^{\mathrm{FF}} |\eta_A|}{|\eta_S| + |\eta_A|}, \tag{4.3.14}$$

then we refine α_S^{FF} percent of the spatial elements and α_A^{FF} percent of the angular elements. The elements to refine are selected as those that have the largest elementwise error indicators. When an element in $\mathcal{T}_S$ or $\mathcal{T}_A$ is marked for refinement it is divided into four sub-elements as illustrated in Figure 4.1. We now describe the first algorithm.

Algorithm 4.3.1. *h*-refinement algorithm 1

The following algorithm employs a DWR based error estimator to compute an approximation to the multiplicative eigenvalue in neutron transport criticality problems. The order of approximation in space and angle, p and q, respectively, the initial spatial and angular meshes, ${\mathcal{T}_S}^1$ and ${\mathcal{T}_A}^1$, respectively, and the total percentage to refine α^{FF} are taken as input.

for $i = 1, \cdots,$ *maximum number of meshes* ***do***
  Compute $\underline{\psi}_h$, $\underline{\hat{z}}$ *and Res* $\left(\underline{\psi}_h, \hat{z}\right)$
  if $\left|Res\left(\underline{\psi}_h, \hat{z}\right)\right| < tol$ ***then***
    Exit loop
  end if
  Compute $|\eta_{\kappa_S}|$ *for all* $\kappa_S \in {\mathcal{T}_S}^i$
  Compute $|\eta_{\kappa_A}|$ *for all* $\kappa_A \in {\mathcal{T}_A}^i$
  Compute η_S, η_A, α_S^{FF} *and* α_A^{FF}
  Mark α_S^{FF} *percent of* $\kappa_S \in {\mathcal{T}_S}^i$ *with the largest* $|\eta_{\kappa_S}|$ *for refinement*
  Mark α_A^{FF} *percent of* $\kappa_A \in {\mathcal{T}_A}^i$ *with the largest* $|\eta_{\kappa_A}|$ *for refinement*
  Build ${\mathcal{T}_S}^{i+1}$ *by refining marked elements in* ${\mathcal{T}_S}^i$
  Build ${\mathcal{T}_A}^{i+1}$ *by refining marked elements in* ${\mathcal{T}_A}^i$
end for
Output solution □

In order to study the performance of this algorithm, tests were run on the benchmark problem published by Maire and Talay in [127]. It had previously been proposed in a technical report for the International Atomic Energy Agency [128]. The spatial geometry is given in Figure 4.2 and incorporates a rectangular geometry divided into five different materials. At the centre of the domain are four adjacent rectangles of different materials, two of which are fissile. Surrounding these is an insulating layer of uniform thickness. The nuclear cross sections for the five regions in the spatial domain are given in Table A.1. The analytical value of this benchmark is not known, however the most accurate value computed by Maire and Talay is 1.00894. They also quoted a value of 1.00890 from Xavier Warin at EDF and the thesis of Baker suggests a value of 1.00888. Following our benchmarking of this problem we take a value of 1.00887 to be the true solution to five decimal places. Figure 4.3 contains plots of the scalar flux and the space averaged flux for the dominant eigenfunction, which provide a visualisation of the spatial and angular parts of the solution, respectively. The scalar flux is largest in Region 1, where the fissile material with the highest value of the fission cross section is located,

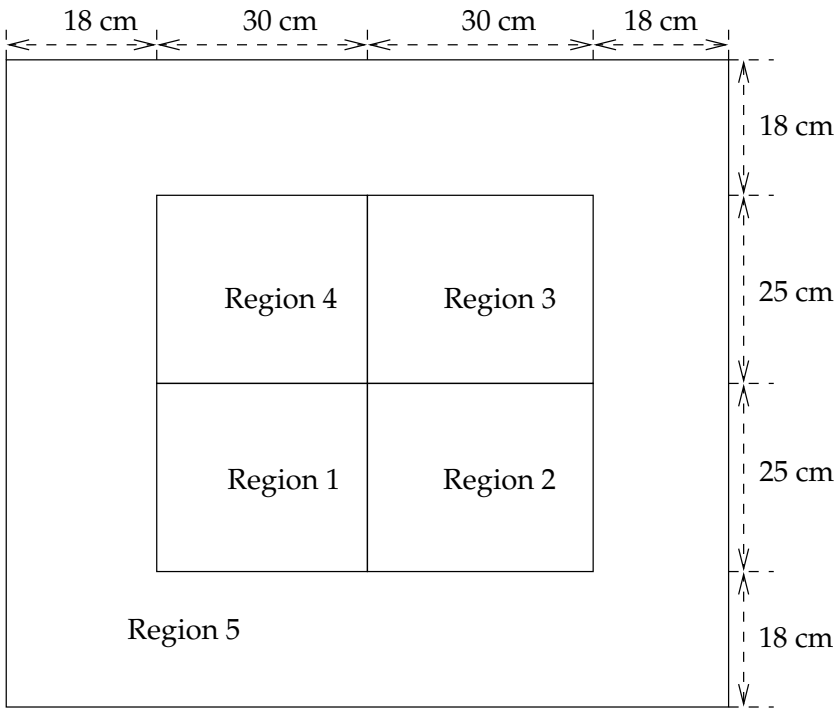


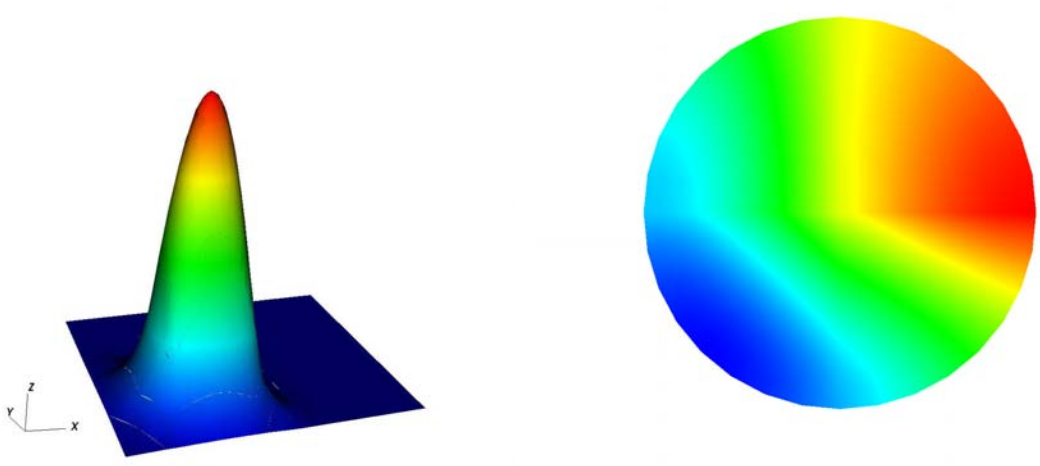
Figure 4.2: The domain for the Maire and Talay test problem. The material coefficients for the five regions marked here are given in Table A.1.

with relatively few neutrons elsewhere in the spatial domain. For this problem there is much more structure in the spatial solution than the angular solution. We note that, though there appears to be some structure to the angular part of the solution, the space averaged flux varies by no more than 1.08% from the mean, hence the angular solution varies very little throughout S_2, as expected for a problem with isotropic scattering. Table 4.1 contains the values for the critical eigenvalues computed for a series of uniformly refined meshes in space and angle at $p = q = 0, 1$ and 2. Note the rapid increase of the number of degrees of freedom required when uniformly refining the spatial and angular meshes. We shall see that the adaptive mesh refinement algorithms presented herein yield much more accurate computed eigenvalues, with many fewer degrees of freedom.

Algorithm 4.3.1 was applied to this problem with α^{FF} set to 25% and with orders of approximation chosen in space and angle to be $p = 1$ and $q = 0$, respectively. Table 4.2 contains the computed eigenvalues and effectivity indices at each iteration, together with the number of degrees of freedom in the primal and dual problems, respectively. We note that the computed effectivities are very close to one, which confirms the accu-

	$p=q=0$		$p=q=1$		$p=q=2$	
Mesh	N	$k_{\text{eff},h}$	N	$k_{\text{eff},h}$	N	$k_{\text{eff},h}$
1	256	0.7962051	4 096	0.9486699	20 736	1.007130
2	4 096	0.8351545	65 536	0.9993939	331 776	1.008381
3	65 536	0.8870401	1 048 576	1.006814	-	-
4	1 048 576	0.9334042	-	-	-	-

Table 4.1: Eigenvalues computed for the Maire and Talay benchmark under uniform refinement.



(a) **(b)**

Figure 4.3: The critical eigenfunction for the Maire and Talay benchmark problem. 4.3a is an elevated plot of the scalar flux. 4.3b is a polar plot of the space averaged flux with the polar coordinate plotted radially.

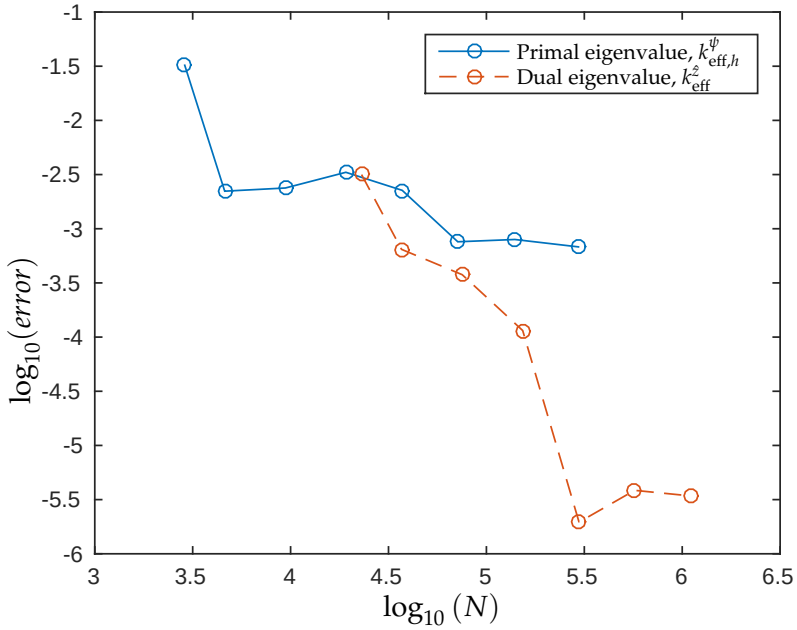


Figure 4.4: h-refinement data from algorithm 4.3.1 for the Maire and Talay benchmark problem with $p = 1$, $q = 0$ in the primal problem.

racy of the computed error representation formula. Figure 4.4 contains a log-log plot of the absolute value of the error in the primal and dual eigenvalues against the size of the linear systems. We remark that the eigenvalues computed achieve significantly improved accuracy, for fewer degrees of freedom, compared to the values computed under uniform refinement. Furthermore, we observe that the primal eigenvalue does not converge monotonically to the computed true value, k^{ψ}_{eff}. Indeed, $k^{\psi}_{\text{eff},h}$ begins beneath the true value before overshooting it and then settling down towards the true value. From the spatial and angular error indicators we see that this is due to persistent error in the angular variables. Table 4.3 presents the values of the spatial and angular contributions to the error estimates, denoted η_S and η_A respectively, and the refinement percentages for this problem. The number of spatial and angular degrees of freedom in each energy group are denoted N_S and N_A, respectively, where the $N = G \times N_S \times N_A$, and $G = 1$, as the present problem is monoenergetic. Here, we observe that despite significant refinement taking place in the angular mesh following the second, third and fourth iterations, $|\eta_A|$ remains relatively large.

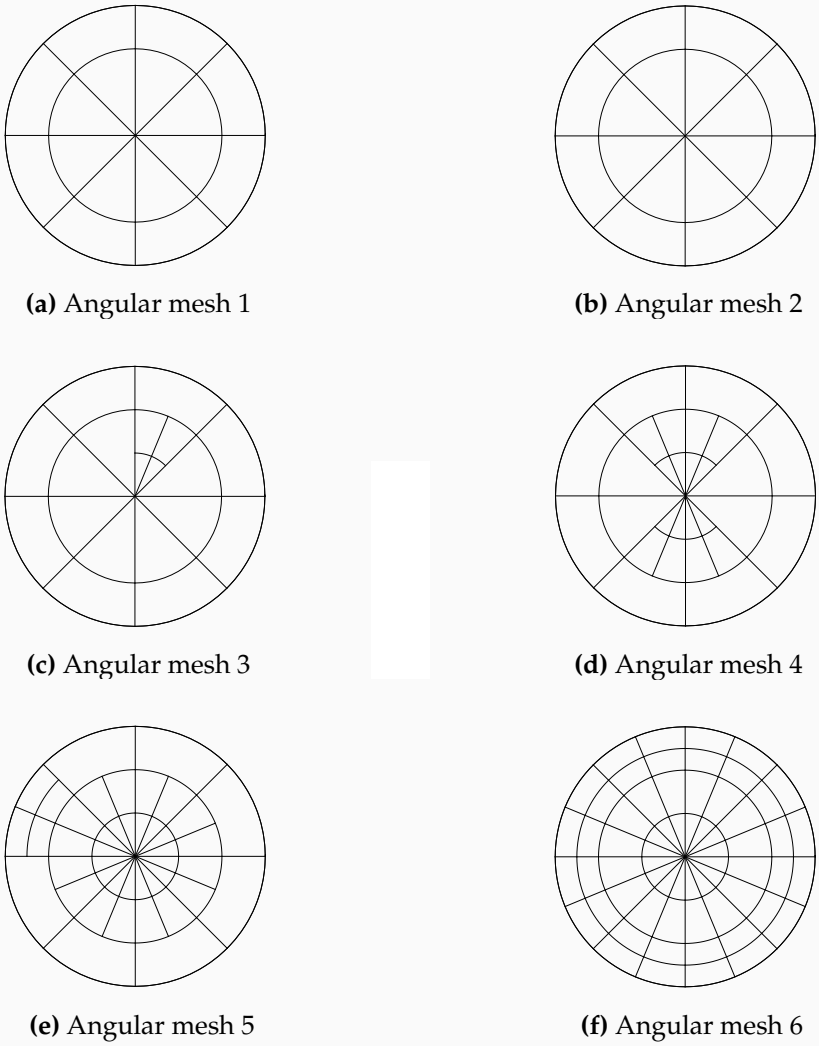
Figure 4.5 contains polar plots of the upper hemisphere of the first six angular meshes, with the polar coordinate plotted radially. Notice that the sixth mesh is the same as the

			Primal		Dual	
Mesh	I_{eff} for $\frac{1}{k_{\text{eff}}}$	I_{eff} for k_{eff}	N	$k^{\psi}_{\text{eff},h}$	N	$k^{\hat{z}}_{\text{eff}}$
1	0.90482	0.90199	2880	0.9766	23040	1.0057
2	0.70898	0.70853	4608	1.0066	36864	1.0082
3	1.1605	1.1601	9405	1.0112	75240	1.0085
4	1.0358	1.0357	19152	1.0122	153216	1.0088
5	1.0006	1.0006	37152	1.0111	297216	1.0089
6	1.0096	1.0096	71424	1.0096	571392	1.0089
7	1.0000	1.0000	139590	1.0097	1116720	1.0089

Table 4.2: The eigenvalues and effectivities for the Maire and Talay benchmark with $p = 1, q = 0$ in the primal problem under adaptive Algorithm 4.3.1.

		Spatial			Angular	
Mesh	N_S	η_S	α_S^{FF}	N_A	η_A	α_A^{FF}
1	180	0.035158	21.5	16	-0.005722	3.5
2	288	0.008243	13.95	16	-0.006527	11.05
3	495	0.003638	9.13	19	-0.006319	15.87
4	684	0.001948	6.67	28	-0.00535	18.33
5	864	0.001498	7.16	43	-0.003734	17.84
6	1116	0.001072	9.27	64	-0.001817	15.73
7	1485	0.000677	7.93	94	-0.001457	17.07

Table 4.3: The error indicators and proportion of elements refined for each iteration of adaptive Algorithm 4.3.1 for the problem in Table 4.2.



(a) Angular mesh 1

(b) Angular mesh 2

(c) Angular mesh 3

(d) Angular mesh 4

(e) Angular mesh 5

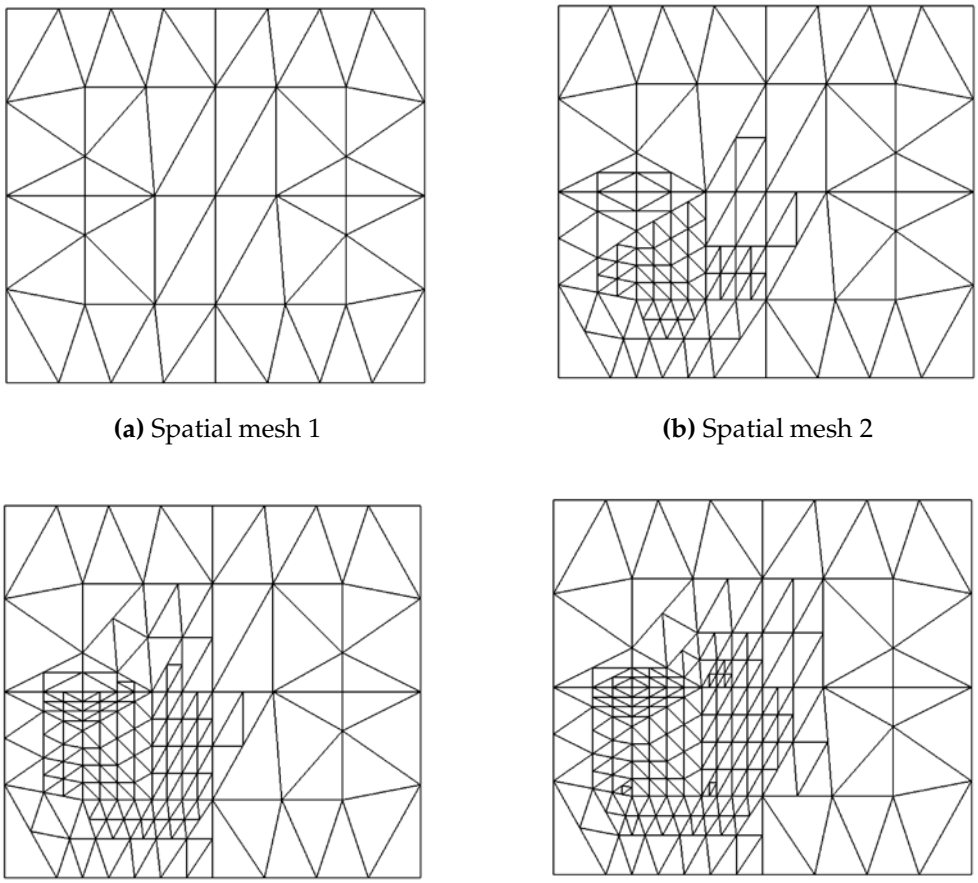
(f) Angular mesh 6

Figure 4.5: A series of angular meshes under fixed fraction refinement for the Maire and Talay benchmark problem, $p = 1, q = 0$ approximation.

initial mesh following a uniform refinement. By comparing the values of η_A with these meshes we see that the greatest reduction in the angular error occurs when moving from the fifth to the sixth mesh. This suggests that the error in the angular variables is evenly distributed throughout S_2. Analysis of the values of the elementwise angular error indicators confirm this hypothesis. The mean and standard deviation of all of the local error indicators, $|\eta_{\kappa_A}|$ and $|\eta_{\kappa_S}|$ were computed at each iteration. It was observed that, for every angular mesh, except mesh 5, the extreme values of $|\eta_{\kappa_A}|$ were within two standard deviations of the mean. For angular mesh 5 all error indicators were within 2.5 standard deviations of the mean. This contrasts with the elementwise spatial error indicators, for which the extreme values were at least 3.75 standard deviations from the mean for all meshes. Similar analyses were conducted for the elementwise error indicators for a variety of benchmark problems and analogous results were found. Indeed this is expected for all of the problems considered in this thesis, since for problems with isotropic scattering the neutrons are scattered with equal likelihood in all directions. This leads to very smooth solutions in the angular domain and therefore we expect the error to be evenly distributed throughout S_2.

We note that the improved accuracy obtained when using regular angular meshes in the present order 0 DG method supports the use of the 'equal weight' quadrature schemes, as is common in the discrete ordinates literature, see [48]. However, considering the data from the second, third, fourth and fifth angular meshes in Figure 4.4 we note that while the error stays relatively constant in the primal problem, the dual eigenvalue benefits from a significant improvement in its accuracy despite the use of irregular angular meshes. This suggests that regular meshes in the angular domain may not be necessary for higher order DG angular discretisations. This could make high order DG methods in angle useful for problems with highly structured angular solutions, such as problems with anisotropic scattering.

Figures 4.6 and 4.7 plot the spatial meshes that were computed. As the method progresses, the algorithm targets the region in the spatial domain where the scalar flux of the dominant eigenvector takes its largest values (see Figure 4.3). We observe that the area targeted is centred on Region 1, the region of fissile material at the centre of the spike in the scalar flux. Furthermore, we note that there is very little refinement in Region 3, where the remainder of the fissile material is located. This contrasts with the mesh that would be obtained by utilising a mesh design-heuristic that was based on considering the material cross sections.



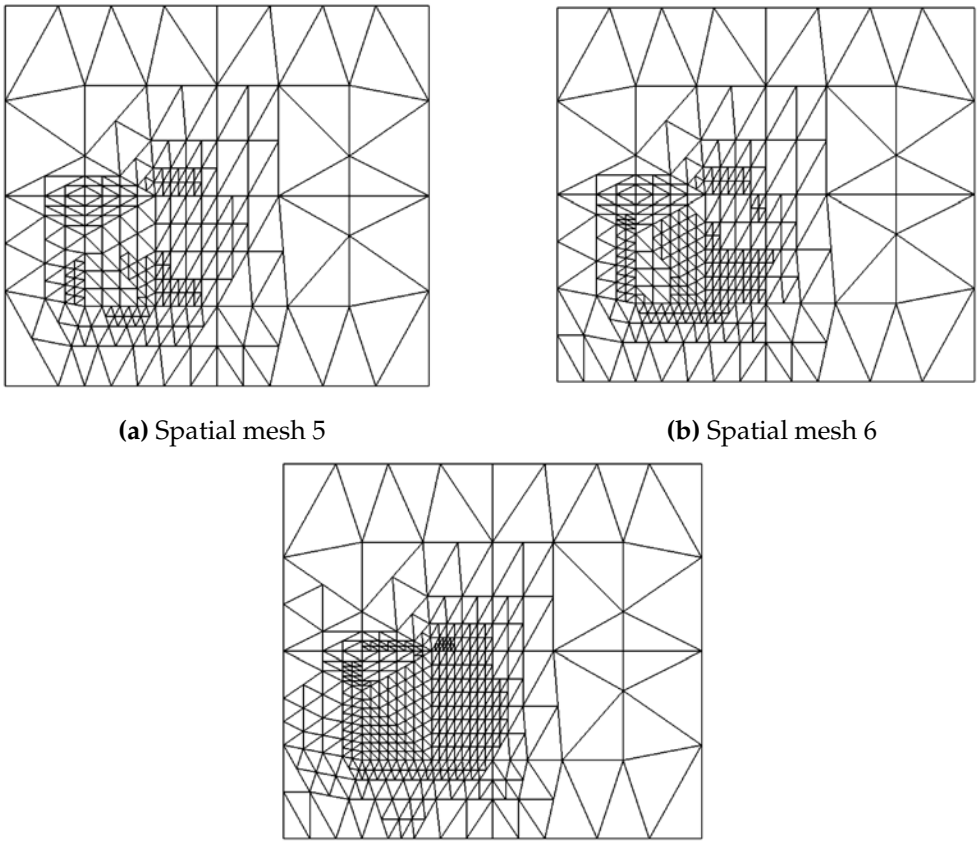
(a) Spatial mesh 1

(b) Spatial mesh 2

(c) Spatial mesh 3

(d) Spatial mesh 4

Figure 4.6: A series of spatial meshes under fixed fraction refinement for the Maire and Talay benchmark problem, $p = 1$, $q = 0$ approximation. We note with reference to Figure 4.3a that the algorithm quickly identifies the region in the domain with the highest values of the scalar flux as the area which contains the most error.



(a) Spatial mesh 5

(b) Spatial mesh 6

(c) Spatial mesh 7

Figure 4.7: A series of spatial meshes under fixed fraction refinement for the Maire and Talay benchmark problem, $p = 1, q = 0$ approximation.

These results prompt us to suggest an improvement to the *h*-refinement algorithm 4.3.1: we propose an adaptive algorithm that decides at each iteration whether to perform refinement in space or angle based on the relative absolute size of $|\eta_S|$ and $|\eta_A|$. If spatial refinement is selected then the algorithm refines a fixed fraction of the spatial elements. If angular refinement is selected then we move to the next angular mesh in a predetermined series of uniform angular meshes. The angular meshes will be based on an equal weighted rectangular partition of S_2. The first mesh ${\mathcal{T}_A}^1$, will have an element in each principal triangle of the unit hemisphere. Each further angular mesh will have four evenly sized angular elements for each element in the previous mesh. The new algorithm is as follows.

Algorithm 4.3.2. *h*-refinement algorithm 2

The following algorithm employs a DWR based error estimator to compute an approximation to the multiplicative eigenvalue in neutron transport criticality problems. The order of approximation in space and angle, p and q, respectively, the initial spatial mesh, ${\mathcal{T}_S}^1$, the refine percentage for the spatial mesh, α^{FF}, and a series of uniform angular meshes ${\mathcal{T}_A}^i$, are taken as input.

for $i = 1, \cdots,$ *maximum number of meshes* ***do***
  Compute $\underline{\psi}_h$, $\underline{\hat{z}}$ *and Res* $\left(\underline{\psi}_h, \hat{z}\right)$
  if $\left|Res\left(\underline{\psi}_h, \hat{z}\right)\right| < tol$ ***then***
    Exit loop
  end if
  Compute η_S *and* η_A
  if $|\eta_S| \geq |\eta_A|$ ***then***
    Compute $|\eta_{\kappa_S}|$ *for all* $\kappa_S \in {\mathcal{T}_S}^i$
    Mark α^{FF} *percent of* $\kappa_S \in {\mathcal{T}_S}^i$ *with the largest* $|\eta_{\kappa_S}|$ *for refinement*
    Build ${\mathcal{T}_S}^{i+1}$ *by refining marked elements in* ${\mathcal{T}_S}^i$
    Set ${\mathcal{T}_A}^{i+1}$ *as* ${\mathcal{T}_A}^i$
  end if
  if $|\eta_S| < |\eta_A|$ ***then***
    Move to mesh ${\mathcal{T}_A}^{i+1}$
    Set ${\mathcal{T}_S}^{i+1}$ *as* ${\mathcal{T}_S}^i$
  end if
end for
Output solution □

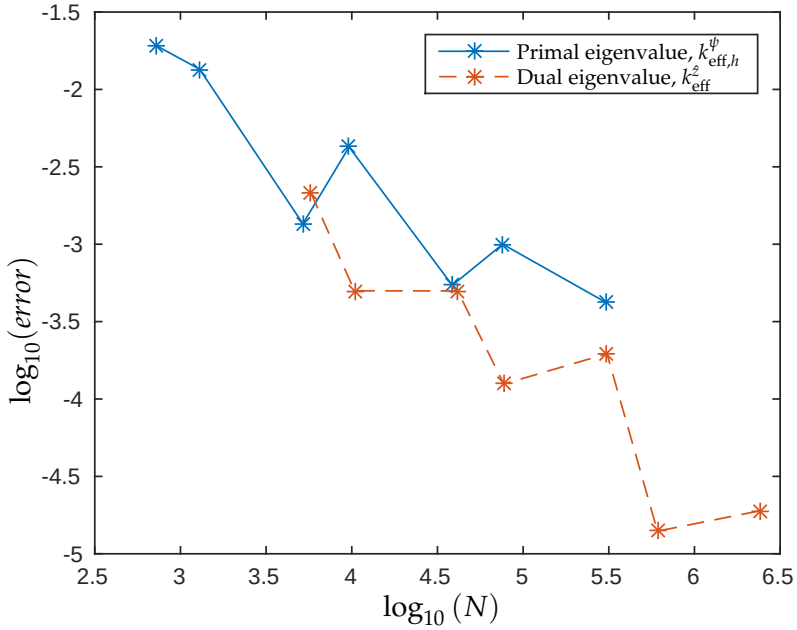


Figure 4.8: h-refinement data from algorithm 4.3.2 for the Maire and Talay benchmark problem with $p = 1, q = 0$ in the primal problem.

This algorithm was tested on the Maire and Talay benchmark with the same initial spatial mesh as for the previous set and the fixed fraction refinement percentage in the spatial variables was chosen to be $\alpha^{\text{FF}} = 25\%$. The computed eigenvalues and effectivities are given in Tables 4.4 and 4.5. Figure 4.8 contains a log-log plot of the absolute values of the error in the eigenvalue against the number of degrees of freedom. We note a much more accurate value of the final primal eigenvalue compared to the previous strategy. Also note that each time the algorithm decides to refine in either space or angle, there is a significant reduction in either η_S or η_A, respectively, at the next iteration. This contrasts with the previous algorithm where refinement was taking place in the angular variables without a significant reduction in the angular error.

Finally, this set of results illustrates an important property of the type of error control employed here. As we are controlling the error in a functional, as opposed to a norm, we do not expect that the error will reduce monotonically under refinement. The non-monotonic convergence of general target functionals is expected and observed by many other authors. Since functionals are not metrics, we can have positive and negative contributions to the error in different parts of the computational domain. In particular, consider the values of η_S and η_A for the third and fourth mesh in Table 4.5 and

			Primal		Dual	
Mesh	I_{eff} for $\frac{1}{k_{\text{eff}}}$	I_{eff} for k_{eff}	N	$k^{\psi}_{\text{eff},h}$	N	$k^{\hat{z}}_{\text{eff}}$
1	0.89044	0.88855	720	0.98963	5760	1.0067
2	0.96111	0.9616	1296	1.0224	10368	1.0094
3	0.64124	0.64093	5184	1.0075	41472	1.0084
4	1.025	1.0249	9648	1.0131	77184	1.0087
5	0.66214	0.66202	38592	1.0083	308736	1.0087
6	0.99527	0.99528	76032	1.0099	608256	1.0089
7	1.0000	1.0000	304128	1.0084	2433024	1.0088

Table 4.4: The eigenvalues and effectivities for the Maire and Talay benchmark with $p = 1, q = 0$ in the primal problem under adaptive Algorithm 4.3.2.

	Spatial		Angular			
Mesh	N_S	η_S	N_A	η_A	Refine space	Refine angle
1	180	0.034697	4	-0.017625	.TRUE.	.FALSE.
2	324	0.007316	4	-0.020305	.FALSE.	.TRUE.
3	324	0.007396	16	-0.006538	.TRUE.	.FALSE.
4	603	0.002321	16	-0.006691	.FALSE.	.TRUE.
5	603	0.002317	64	-0.001969	.TRUE.	.FALSE.
6	1188	0.000967	64	-0.001981	.FALSE.	.TRUE.
7	1188	0.000966	256	-0.000563	.TRUE.	.FALSE.

Table 4.5: The spatial and angular error indicators for each iteration of adaptive Algorithm 4.3.2 for the problem in Table 4.4.

compare them with the primal eigenvalues from Table 4.4. For the third mesh we have $\eta_A \approx -\eta_S$, and as a consequence the computed eigenvalue is accurate to within 0.0013 of the true value. When the problem then undergoes spatial refinement, the spatial error is reduced and contribution to the error from the spatial and angular parts of the problem no longer cancel each other out. The result is that the error as measured by the eigenvalue increases to 0.0042. Note also how this affects the effectivity index. When there is significant error cancelling, as for meshes 3 and 5, our error representation formula slightly underestimates the total error compared to the other meshes. Despite this, the computed effectivities remain close to one, again confirming the quality of the computed error representation formula.

In order to choose which algorithm to use for our remaining computations we consider the total time required to compute the primal eigenvalue, compared to the error. Both sets of computations were completed on the same server and were both allocated 8

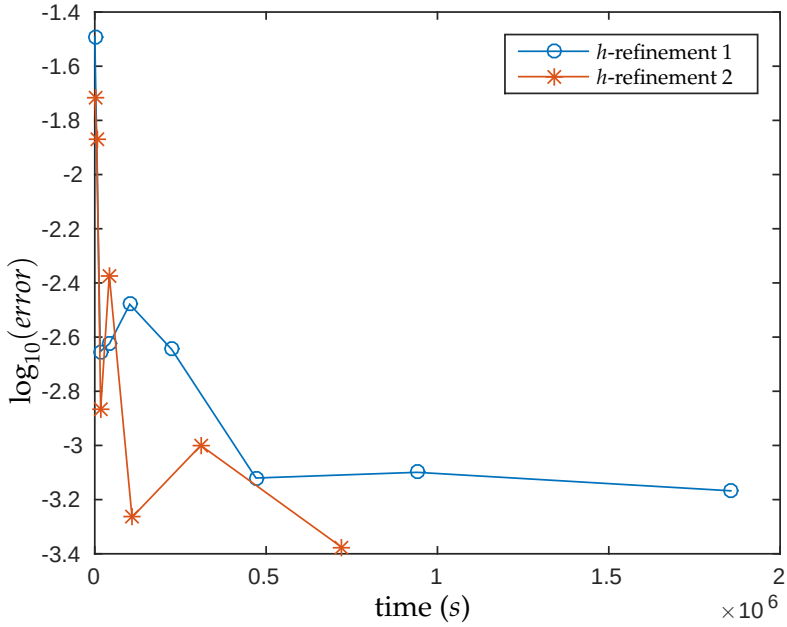


Figure 4.9: Cumulative time to compute the primal eigenvalues under h-refinement for Algorithm 4.3.1 and Algorithm 4.3.2. Data from the Maire and Talay benchmark problem with $p = 1$, $q = 0$ in the primal problem. Computed on 8 OpenMP threads.

OpenMP threads. Figure 4.9 plots the series of primal eigenvalues against the total time that the code had been running when they were computed. We note that Algorithm 4.3.2 computes its values considerably quicker than Algorithm 4.3.1. Consequently, all further h-refinement computations in this thesis will be computed with Algorithm 4.3.2. We remark that if we compare the dual eigenvalues computed by either algorithm, it is the first algorithm that achieves the more accurate results. However, as our error representation formula was derived for the error in the primal eigenvalue, it is by comparing $k^{\psi}_{\text{eff},h}$ that we obtain the most pertinent comparison.

4.4 An *hp*-refinement algorithm

In this section we present an *hp*-adaptive refinement algorithm for the k_{eff}-eigenvalue problem. The exploitation of *hp*-methods are motivated by the fact that when the underlying solution is sufficiently smooth, or indeed analytic, p-adaptivity can achieve convergence which is exponential with respect to the number of degrees of freedom in the finite element space. However, for many problems of practical interest the solution

is only piecewise analytic and consequently a combination of *h*- and *p*-refinement is required in order to accurately represent the solution. An *hp*-adaptive algorithm provides the flexibility to select the type of refinement most appropriate to each region in the domain, thus recovering exponential convergence even for non smooth problems whose solution is piecewise analytic.

We choose *hp*-refinement for the spatial part of the neutron transport equation because we need to treat irregularly shaped domains featuring discontinuities in the nuclear cross sections. Hence we expect that the underlying solutions will not be analytic everywhere in the domain. The *hp*-version of the DGFEM was first introduced for linear hyperbolic problems by Bey and Ogen in [13]. They proved *a priori* and *a posteriori* error estimates for their method that reduce to the optimal $\mathcal{O}(h^{p+\frac{1}{2}})$ estimates for the *h*-DGFEM when p is fixed, with respect to a mesh dependent norm. Houston, Schwab and Süli, in [11], extended this analysis to obtain bounds that are also optimal in p, and derived an exponential convergence estimate for problems with an elementwise analytic solution. To the author's knowledge, this is the first attempt at employing *hp*-finite element methods to criticality problems in neutron transport. As we consider problems with isotropic scattering we expect the analytical solutions in the angular domain to be extremely smooth; thereby we employ *q*-refinement in angle.

4.4.1 *hp*-finite element space

We define the *hp*-finite element space in which we will seek our DG solutions. We utilise the spatial finite element mesh defined in Chapter 3, $\mathcal{T}_S$, as well as the polynomial spaces, $\mathcal{P}_p$ and $\mathcal{Q}_p$, and the spatial element maps to the canonical elements, $F_{\kappa_S}(\cdot)$. Let

$$\mathbf{p} = \{\, p_{\kappa_S} \;\mid\; \kappa_S \in \mathcal{T}_S \} \tag{4.4.1}$$

be a polynomial degree vector on $\mathcal{T}_S$. Then the discontinuous spatial *hp*-finite element space is

$$S_S^{\mathbf{p}}(\mathcal{T}_S, F_{\mathcal{T}_S}) = \left\{ \psi_S \in L_2(\mathcal{D}) \;\mid\; \psi_S|_{\kappa_S} \circ F_{\kappa_S} \in \mathcal{R}_{p_{\kappa_S}} \right\}, \tag{4.4.2}$$

where $\mathcal{R}_{p_{\kappa_S}} = \mathcal{Q}_{p_{\kappa_S}}$ if $\hat{\kappa} = F_{\kappa_S}^{-1}(\kappa_S)$ is a square and $\mathcal{R}_{p_{\kappa_S}} = \mathcal{P}_{p_{\kappa_S}}$ if $\hat{\kappa} = F_{\kappa_S}^{-1}(\kappa_S)$ is a triangle. We note that when the polynomial degrees are uniform, namely when $p_{\kappa_S} = p$ for all κ_S, then this is the same as the *h*-adaptive finite element space $S_S^p(\mathcal{T}_S, F_{\mathcal{T}_S})$, which was defined in the previous chapter.

We combine this with the angular finite element space $S_A^q(\mathcal{T}_A, F_{\mathcal{T}_A})$, to define the full space-angle *hp*-finite element space by

$$\begin{aligned}\mathcal{V}_{hp}^{\mathbf{p},q} = \{\psi_h \in L_2(\mathcal{D}\times S_2) \mid \psi_h = \psi_S \times \psi_A\,, \\ \psi_S \in S_S^{\mathbf{p}}(\mathcal{T}_S, F_{\mathcal{T}_S})\,,\ \psi_A \in S_A^q(\mathcal{T}_A, F_{\mathcal{T}_A})\}.\end{aligned} \tag{4.4.3}$$

Then the *hp*-DG approximation to the k_{eff}-eigenvalue problem is given by: find $\underline{\psi}_{hp} \in \mathbb{R}\times\mathcal{V}_{hp}^{\mathbf{p},q}$ such that

$$\mathcal{N}\left(\underline{\psi}_{hp}, \underline{v}_{hp}\right) = 0, \tag{4.4.4}$$

for all $\underline{v}_{hp} \in \mathbb{R}\times\mathcal{V}_{hp}^{\mathbf{p},q}$. We introduce the dual *hp*-DG space, $\mathcal{V}_{hp}^{\mathbf{p}+\mathbf{1},q+1}$, where $\mathbf{1}$ is a vector of 1s. Then the computable dual *hp*-DG solution is given by: find $(k_{\text{eff},hp}^z, z_{hp}) \in \mathbb{R}\times\mathcal{V}_{hp}^{\mathbf{p}+\mathbf{1},q+1}$ such that

$$F\left(v_{hp}, z_{hp}\right) = k_{\text{eff},hp}^z\left(T-S\right)\left(v_{hp}, z_{hp}\right) \tag{4.4.5}$$

for all $v_{hp} \in \mathcal{V}_{hp}^{\mathbf{p}+\mathbf{1},q+1}$. We compute the error representation formula, as well as spatial, angular and elementwise error indicators analogously to the case when *h*-refinement was employed. Furthermore, we define the scalar flux in the *hp* framework for the primal eigenfunction as $\phi^{\psi} \in S_S^{\mathbf{p}}(\mathcal{T}_S, F_{\mathcal{T}_S})$, where,

$$\phi^{\psi}\left(\mathbf{r}\right) = \int_{S_2} \psi_{hp}\left(\mathbf{r}, \boldsymbol{\Omega}\right)\, d\boldsymbol{\Omega}, \tag{4.4.6}$$

and the scalar flux in the *hp* framework for the dual eigenfunction as $\phi^z \in S_S^{\mathbf{p}+\mathbf{1}}(\mathcal{T}_S, F_{\mathcal{T}_S})$, where,

$$\phi^{z}\left(\mathbf{r}\right) = \int_{S_2} z_{hp}\left(\mathbf{r}, \boldsymbol{\Omega}\right)\, d\boldsymbol{\Omega}. \tag{4.4.7}$$

The final ingredient necessary for the *hp*-refinement algorithm is a decision mechanism for determining whether the spatial elements that are marked for refinement undergo *h*-refinement, whereby the present element is subdivided into sub-elements, or *p*-refinement, where the local finite element space is enriched to a higher order approximation. Several methods for determining the refinement type have been proposed in the literature, a comparison of which can be found in [129]. For the present implementation we employ the *hp*-refinement criterion developed by Houston and Süli in [130]. This criterion assesses the local smoothness of the underlying solution within each finite element based on considering the local decay rate of the Legendre coefficients which comprise the finite element solution in each dimension. This is justified with reference to the result of Eibner and Melenk, in [131], which states that the decay

rate of the coefficients of a function expanded in orthogonal polynomials on a triangle will be exponential if and only if the function is analytic. In particular, for each element that has been marked for refinement κ_S, the test of Houston and Süli uses the solution coefficients to compute an estimate of the size of the domain of analyticity, ϑ_{κ_S}. This estimate is then compared to a steering parameter ϑ, if $\vartheta_{\kappa_S} \leq \vartheta$ then the local solution is judged to be locally 'smooth', otherwise the local solution is judged to not be locally 'smooth'.

We utilise this test on the primal and dual scalar fluxes, ϕ^{ψ} and ϕ^{z}, respectively, in order to determine the analyticity of the computed solutions on each spatial element that has been marked for refinement. If either of ϕ^{ψ} or ϕ^{z} is locally 'smooth' on an element κ_S, then p-refinement will be more effective than h-refinement, since the error will be expected to decay exponentially as p_{κ_S} is increased. However, if ϕ^{ψ} and ϕ^{z} are not locally 'smooth', then h-refinement will be necessary in order to isolate those areas in $\mathcal{D}$ where the primal and dual solutions are non-'smooth' from the 'smooth' areas. In this way we seek to reduce the influence of singularities and discontinuities on the computed solutions, thereby enhancing the convergence achieved by the p-refinement in the 'smooth' regions.

We propose an algorithm which incorporates an hp-refinement in the spatial domain whenever the error in the spatial variables exceeds the angular error. As we expect highly unstructured angular solutions in all of the benchmarks computed in this thesis, we select a scheme of uniform q-refinement in the angular variables whenever the angular error exceeds the spatial error. The hp-refinement algorithm is as follows:

Algorithm 4.4.1. *hp*-refinement algorithm
The following algorithm employs a DWR based error estimator to compute an approximation to the multiplicative eigenvalue in neutron transport criticality problems. An angular mesh $\mathcal{T}_A$, an initial spatial mesh, $\mathcal{T}_S^1$, the spatial refinement percentage, α^{FF}, the hp*-steering parameter ϑ, and initial orders of polynomial approximation in space $\mathbf{p}^1$, and angle q^1 are taken as input.*

for $i = 1, \cdots,$ *maximum number of meshes* ***do***
  Compute $\underline{\psi}_{\mathrm{hp}}$, $\underline{z}_{\mathrm{hp}}$ *and Res* $\left(\underline{\psi}_{\mathrm{hp}}, z_{\mathrm{hp}}\right)$
  if $\left|Res\left(\underline{\psi}_{\mathrm{hp}}, z_{\mathrm{hp}}\right)\right| < tol$ ***then***
    Exit loop
  end if

Set $\mathbf{p}^{i+1} \leftarrow \mathbf{p}^i$
Compute η_S *and* η_A
if $|\eta_S| \geq |\eta_A|$ ***then***
Compute $|\eta_{\kappa_S}|$ *for all* $\kappa_S \in \mathcal{T}_S{}^i$
Mark α^{FF} *percent of* $\kappa_S \in \mathcal{T}_S{}^i$ *with the largest* $|\eta_{\kappa_S}|$ *for refinement*
Compute the scalar fluxes, ϕ^ψ *and* ϕ^z
for $\kappa_S \in \mathcal{T}_S{}^i$ ***do***
if κ_S *has been marked for refinement* ***then***
Compute $\vartheta^\psi_{\kappa_S}$ *for* $\phi^\psi|_{\kappa_S}$
Compute $\vartheta^z_{\kappa_S}$ *for* $\phi^z|_{\kappa_S}$
if $\vartheta^\psi_{\kappa_S} > \vartheta$ ***and*** $\vartheta^z_{\kappa_S} > \vartheta$ ***then***
Mark κ_S *for h-refinement*
end if
if $\vartheta^\psi_{\kappa_S} \leq \vartheta$ ***or*** $\vartheta^z_{\kappa_S} \leq \vartheta$ ***then***
Set $p_{\kappa_S} \leftarrow p_{\kappa_S} + 1$ *in* $\mathbf{p}^{i+1}$
end if
end if
end for
Build $\mathcal{T}_S{}^{i+1}$ *by refining elements marked for h-refinement in* $\mathcal{T}_S{}^i$
Set $q^{i+1} \leftarrow q^i$
end if
if $|\eta_S| < |\eta_A|$ ***then***
Set $q^{i+1} \leftarrow q^i + 1$
Set $\mathcal{T}_S{}^{i+1}$ *as* $\mathcal{T}_S{}^i$
end if
end for
Output solution □

This algorithm was tested on the Maire and Talay benchmark problem with the same initial spatial mesh as for the *h*-refinement Algorithm 4.3.2. The angular mesh was selected to be as coarse as possible, with just a single angular element in each of the four polar triangles in the hemisphere (see Figure 1.1a). α^{FF} was set at 25%, q^1 was set to 0 and $\mathbf{p}^1$ was set to $\mathbf{1}$. The computed eigenvalues and effectivities are given in Table 4.6. Note the accuracy of the primal eigenvalues, which are considerably more accurate than those values obtained by previous algorithms. This table also displays

			Primal		Dual	
Mesh	I_{eff} for $\frac{1}{k_{\text{eff}}}$	I_{eff} for k_{eff}	N	$k^{\psi}_{\text{eff},h}$	N	$k^{\hat{z}}_{\text{eff}}$
1	0.88959	0.88768	720	0.98963	5760	1.0067
2	0.96627	0.96682	1584	1.0259	11904	1.0094
3	0.86001	0.85958	6336	1.0053	26784	1.0084
4	1.5931	1.5929	10032	1.009	42408	1.0088
5	0.88904	0.88895	22572	1.008	75392	1.0088
6	1.0000	1.0000	39672	1.0086	125312	1.0089
7	-	-	60984	1.00883	-	-

Table 4.6: The eigenvalues and effectivities for the Maire and Talay benchmark under the *hp*-refinement algorithm.

	Spatial		Angular			
Mesh	N_S	η_S	N_A	η_A	Refine space	Refine angle
1	180	0.034697	4	-0.017625	.TRUE.	.FALSE.
2	396	0.004111	4	-0.020627	.FALSE.	.TRUE.
3	396	0.004141	16	-0.001048	.TRUE.	.FALSE.
4	627	0.000808	16	-0.001068	.FALSE.	.TRUE.
5	627	0.000811	36	-4×10^{-06}	.TRUE.	.FALSE.
6	1102	0.000273	36	-4×10^{-06}	.TRUE.	.FALSE.

Table 4.7: The spatial and angular error indicators for each iteration of the *hp*-refinement algorithm.

effectivities close to one for most meshes, confirming the quality of the error estimates given by our error representation formula. However we observe that the effectivities for mesh 4 show a slight overestimation of the error. We remark that this coincides with significant error cancelling between the spatial and angular solutions.

Details of the spatial and angular error estimates at each mesh are given in Table 4.7. The values of η_A demonstrate the benefit of using a higher order finite element method in the angular variables. Despite using an extremely coarse mesh containing only four angular elements, we still obtain a very accurate angular solution by using a $q = 2$ approximation. Indeed, we see the angular error estimate reduced to -4×10^{-06} for an approximation containing only 36 degrees of freedom. Figure 4.10 contains a semi-log plot of the error in k_{eff} against the number of degrees of freedom in the primal finite element space. In addition to the *hp*-refinement convergence we plot two sets of results from Algorithm 4.3.2; the $p = 1$, $q = 0$ results from Tables 4.4 and 4.5, and results from

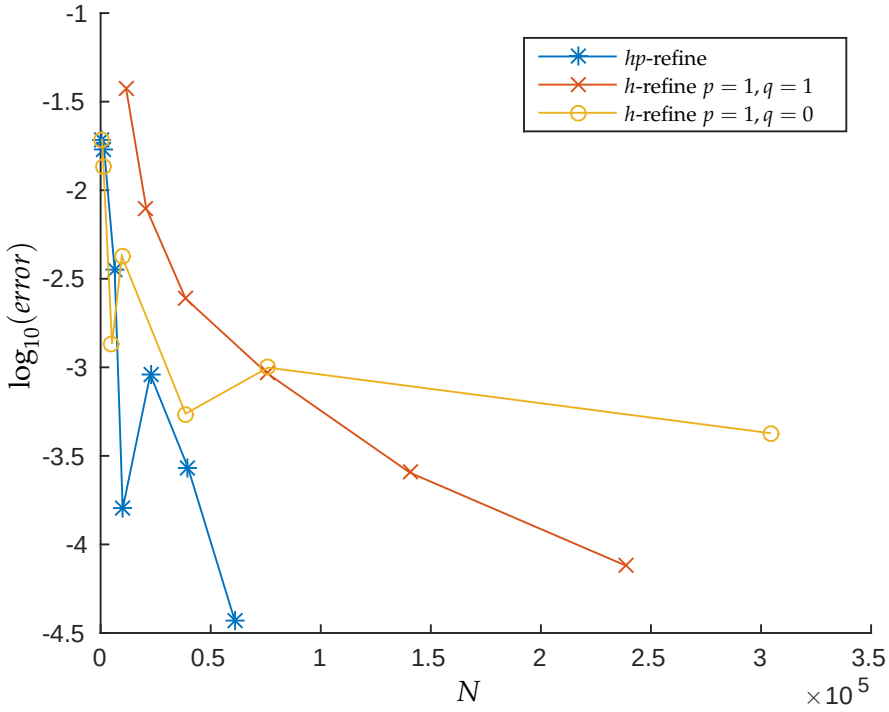


Figure 4.10: The log of the error in the primal eigenvalue against the number of degrees of freedom for the Maire and Talay benchmark problem.

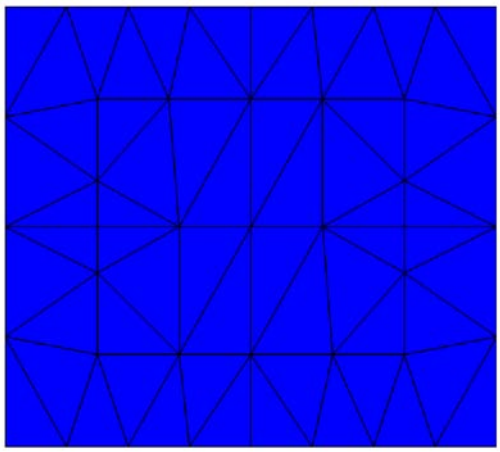
a run with the same initial meshes and a $p = 1, q = 1$ approximation. We note that the *hp*-refinement data achieves the same accuracy as the most accurate primal eigenvalue computed by the *h*-refinement algorithm, but with approximately 80% fewer degrees of freedom required. Furthermore, we note an improvement in the accuracy of more than an order of magnitude when comparing eigenvalues computed using finite element spaces with the same number of degrees of freedom.

Though Figure 4.10 only compares the algorithms in terms of the number of degrees of freedom, we note that the *hp*-refinement algorithm also outperforms the *h*-refinement algorithm both in terms of the total memory required as well as the total CPU time required. As each of the algorithms was implemented without fully assembling the matrices, the amount of memory required is dominated by the memory required by the two Krylov subspace based algorithms. Therefore, as the software that implement these specify the memory that they need in terms of a multiple of the system size (see

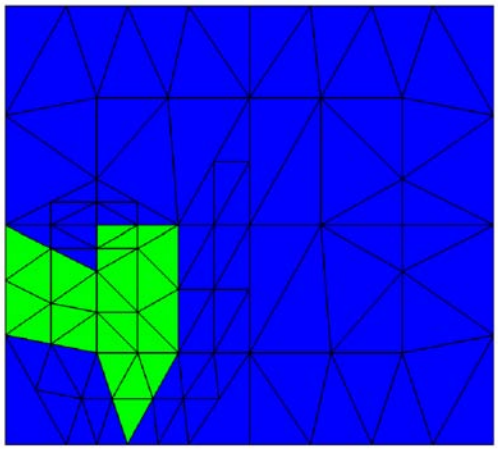
[98] and [104]), Figure 4.10 also tells us that the *hp*-refinement algorithm is highly memory efficient compared to the other two algorithms. Furthermore, we note that the seven data points obtained by the algorithm took approximately the same amount of CPU time to compute as the seven data points computed by the $p = 1$, $q = 0$ h-refinement algorithm, despite computing the eigenvalue to a much greater accuracy. The $p = 1$, $q = 1$ h-refinement algorithm took approximately 3 times as much CPU time as these, and whilst it did achieve more accurate results than the $p = 1$, $q = 0$ h-refinement algorithm, it was easily outperformed by the *hp*-refinement algorithm.

Figures 4.11 and 4.12 contain the seven spatial meshes that were computed, with each element, κ_S, coloured to indicate the order of polynomial approximation employed there, p_{κ_S}. We note a mixture of h- and p-refinement in the bottom left of the spatial domain, where the scalar flux is largest. This refined region is centred on Region 4 from Figure 4.2, where the fissile material with the highest fission cross section is located. We note that, at the centre of this region, slightly less refinement is required compared to its edges. Indeed, it is the finite elements close to the material discontinuities that receive the most refinement. Furthermore, the elements where h-refinement takes place but p-refinement does not are all close to these material discontinuities. We remark that the computed solution must manifest a lack of smoothness in these parts of the domain, and the algorithm is h-refining in order to isolate the non-smooth region from those regions that exhibit greater regularity.

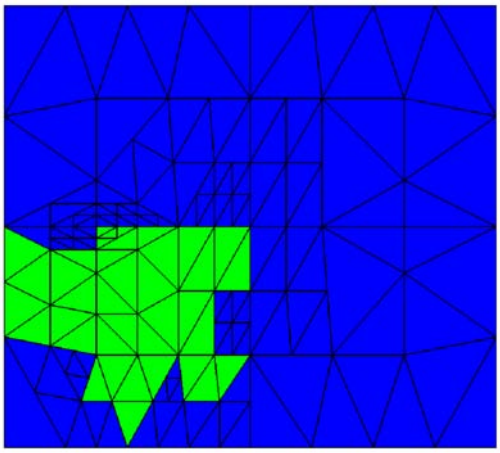
We note that though we have presented results for both h and *hp*-refinement in the spatial variables, we have not studied pure p-refinement in space. The decision not to study p-refinement independently of *hp*-refinement was made with reference to the literature on DG methods for hyperbolic problems, in particular [130] and [11], which give examples of how the DG solution to hyperbolic equations can become locally non-smooth in certain areas of the domain. Furthermore we can see in Figure 4.12 an example of how the analyticity testing procedure used for the present calculations locates the parts of the domain where the DG solution is not analytic. If a pure p-refinement algorithm were used to attempt to reduce the error in these non-smooth regions we would expect little or no reduction. We might even see the error increasing under refinement, whilst at the same time increasing the computational requirements of finding the solution.



(a) Spatial mesh and p_{κ_S}, mesh 1



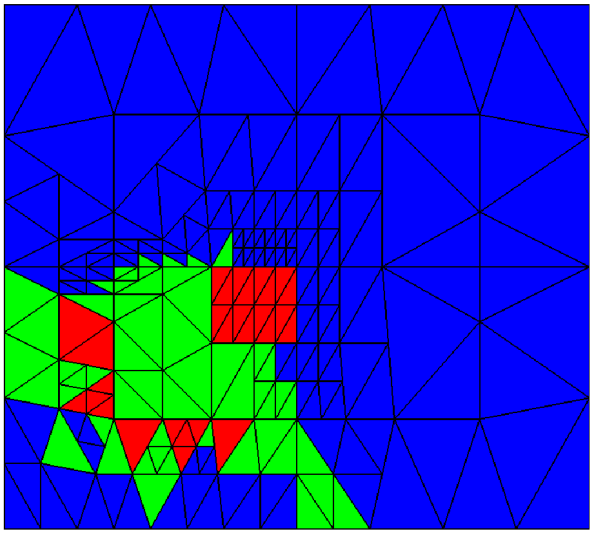
(b) Iterations 2 and 3



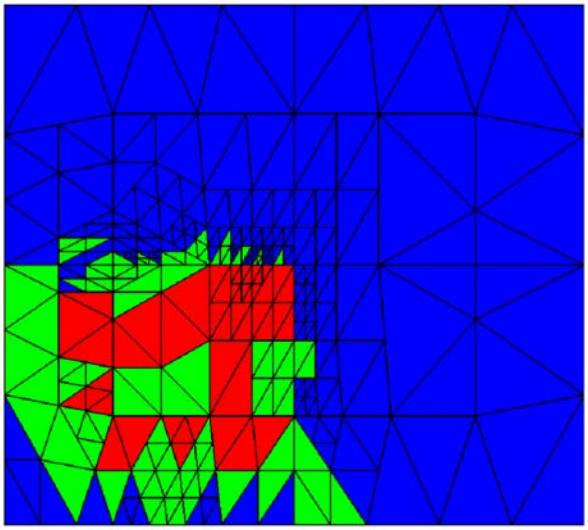
(c) Iterations 4 and 5

$p_{\kappa_S} = 1$ $p_{\kappa_S} = 2$ $p_{\kappa_S} = 3$

Figure 4.11: The polynomial vector $\mathbf{p}^i$ plotted on the spatial meshes for the Maire and Talay benchmark under *hp*-refinement.



(a) Iteration 6



(b) Iteration 7

$p_{\kappa_S} = 1$ $p_{\kappa_S} = 2$ $p_{\kappa_S} = 3$

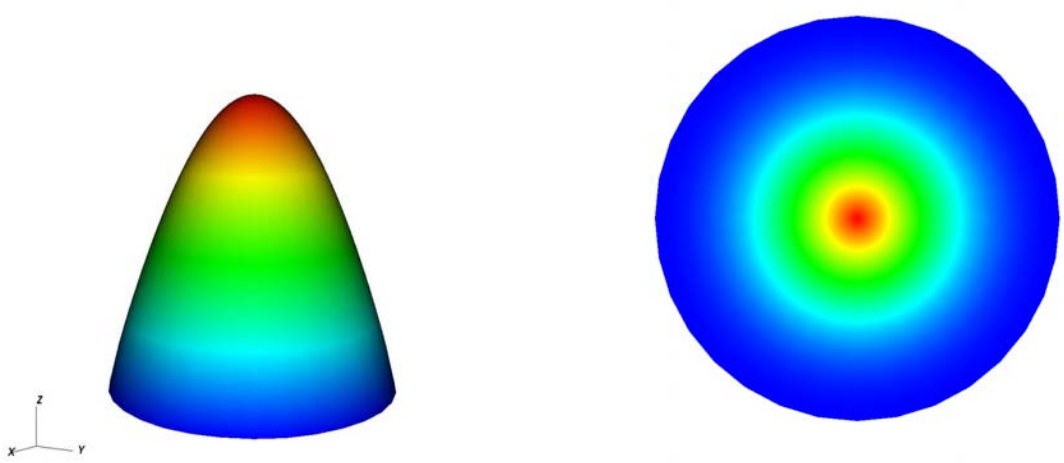
Figure 4.12: The polynomial vector $\mathbf{p}^i$ plotted on the spatial meshes for the Maire and Talay benchmark under *hp*-refinement.

CHAPTER 5

Application to Industrial Problems

In order that we may have full confidence in the discontinuous-Galerkin (DG) neutron transport criticality solver developed in this thesis, it is necessary to compute the solution to a variety of problems with known solutions. In Chapter 3 we considered a simple, artificially forced source problem that was solved in order to compute the order of convergence of the DG method employed. In Chapter 4 we used the monoenergetic criticality problem of Maire and Talay [127] to develop *h*- and *hp*-adaptive algorithms, finding that our most accurate eigenvalues matched the values of k_{eff} quoted in the literature, up to four decimal places. In this chapter we further verify the code by computing a set of challenging, and industrially relevant, benchmarks, including problems that incorporate inhomogeneous spatial domains as well as multigroup approximations for the energy spectrum.

We consider three problems from the published literature followed by two problems from a technical report compiled by Albrecht Kyrieleis at Serco Assurance [132]. We begin with a simple monoenergetic problem and then proceed to consider a further four problems that incorporate a multigroup approximation. The problems have increasingly complicated spatial domains, beginning with a spatially homogeneous problem and proceeding to problems with up to 18 different spatial regions representing up to 8 different sets of material coefficients. We model reactors in a variety of shapes, beginning with a cylindrical reactor, for which we employ a mixture of triangular finite elements, quadrilateral finite elements and quadrilateral finite elements with polynomial shaped boundaries in order to accurately represent the geometry. We consider a geometry composed of a tessellation of hexagonal prisms, for which we employ triangular finite elements, and three cuboidal reactors, for which we employ unstructured trian-



(a) **(b)**

Figure 5.1: The critical eigenfunction for the Los Alamos benchmark problem. 5.1a is an elevated plot of the scalar flux. 5.1b is a polar plot of the space averaged flux with φ plotted radially.

gular and unstructured rectangular meshes. For the unstructured triangular meshes the Triangle mesh generator [109], is employed. For all other problems, meshes were generated manually.

5.1 Los Alamos benchmark

The first benchmark we consider is taken from a comprehensive problem set compiled by Sood, Forster, and Parsons from the Los Alamos National laboratory, NM [106]. Sood, Foster and Parsons present 75 different criticality problems in slab, spherical and cylindrical geometries, including problems with inhomogeneous spatial domains and anisotropic scattering in the angular domain. All of these problems are taken from reviewed journal articles and are known to have critical eigenvalues of 1, to at least 5 decimal places. The problem selected for this thesis was originally presented by Westfall, in [133], and is a monoenergetic problem which incorporates an infinite cylinder of Plutonium-239 (^{239}Pu) with vacuum boundary conditions. The cylinder is of radius 4.279960 centimetres and the material cross sections for ^{239}Pu are given in Table A.2. It is indexed as problem 7 in [106].

We note that though this benchmark is simpler than the problem of Maire and Talay

	$p = q = 0$		$p = q = 1$		$p = q = 2$	
Mesh	N	$k_{\text{eff},h}$	N	$k_{\text{eff},h}$	N	$k_{\text{eff},h}$
1	256	1.054552	3 968	1.001798	19 872	1.000181
2	4 096	1.011031	63 488	1.000118	317 952	0.999968
3	65 536	1.000611	1 015 808	0.999961	5 087 232	0.999996
4	1 048 576	0.998932	-	-	-	-

Table 5.1: Eigenvalues computed for the Los Alamos benchmark under uniform refinement.

that was approximated in the previous chapter, it does enable us to demonstrate a useful feature of the present solver: the ability to represent spatial domains with piecewise polynomial shaped boundaries. As the Los Alamos benchmark problem comprises a circle in the (x, y)-plane, a very large number of square or triangular finite elements would be required in order to accurately represent the shape of the physical boundary. Indeed, a much larger number of elements would be needed at the boundary than would otherwise be needed to resolve an accurate solution within the actual computational domain. The present computations avoid this pitfall, however, by deforming each spatial element κ_S, where $\partial\kappa_S \cap \Gamma \neq \emptyset$, so that the element boundary coincident with Γ is shaped as a parabola. As before, each κ_S that is deformed in this way is defined as a mapping from either the reference square or the reference triangle, though this mapping is no longer affine. When such an element is marked for h-refinement, a similar isotropic refinement process is undertaken as was illustrated in Figure 4.1, however, the new mesh node that is introduced on the boundary of κ_S is at the midpoint between the existing nodes along Γ, as opposed to the geometric midpoint between those nodes. This means that each h-refinement that occurs at the domain boundary leads to the replacement of a single quadratic boundary segment with two quadratic boundary segments, thus improving the accuracy with which the computational domain represents the physical domain. We remark that under p-refinement no such improvement takes place.

The scalar flux and the space averaged flux from the critical eigenfunction are shown in Figure 5.1. We notice that the solution appears to be very smooth in both space, as illustrated by the scalar flux, and angle, as illustrated by the space averaged flux. Consequently, we expect the hp-refinement algorithm to select p-refinement for those

elements that are marked for refinement. Table 5.1 gives the computed k_{eff} values for this problem under uniformly refined meshes, with order of approximation in the spatial and angular domain of $p = q = 0, 1$ and 2 respectively. We note that the most accurate eigenvalue computed here is within 10^{-5} of the value quoted by Sood et al. of $k_{\text{eff}} = 1$, which they claim is accurate to five decimal places. This value compares favourably with the most accurate values presented in [42] and [43], which computed values within 10^{-4} and 10^{-1} of the expected eigenvalue respectively.

The present problem was solved using the *h*-refinement algorithm with a $p = 1$, $q = 0$ method and a $p = 1$, $q = 1$ method, as well as by the *hp*-refinement algorithm. The same initial spatial and angular meshes were chosen for all three runs, with the spatial mesh comprising 32 elements, with four elements in the radial direction and eight in the azimuthal direction, and the initial angular mesh comprising 4 elements, with a single angular element in each principal triangle of the hemisphere. The fixed fraction refine percentage in the spatial problem was chosen as $\alpha^{\text{FF}} = 25\%$. Table 5.2 gives the total problem sizes and computed eigenvalues from the primal and dual solves at each iteration, together with the effectivities for both k_{eff} and its reciprocal.

We remark that nearly all of the effectivities are close to 1, especially for the *hp*-refinement algorithm. This confirms the accuracy of the computed error representation formula. However, some of the effectivities computed for the *h*-refinement algorithm show a slight over- or under-estimation of the error at certain stages in the computation. In particular we see an over-estimation of the error at the fourth iteration of the $p = 1$, $q = 0$ data and an under-estimation at the second and fifth iterations of the $p = 1$, $q = 1$ data. We note with reference to Table 5.3, which gives the spatial and angular error indicators, η_S and η_A respectively, that the iterations for which the values of I_{eff} are not close to one coincide with those for which $\eta_S \approx -\eta_A$. In other words, there is a cancellation between the error contained in the spatial discretisation and that contained in the angular discretisation.

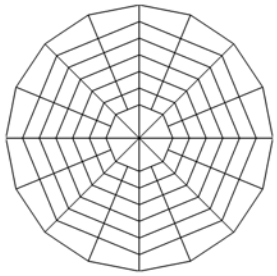
We observe that the *hp*-refinement algorithm does not converge to the same value as the *h*-refinement algorithm. This is because, for the *hp*-refinement data, the imperfect representation of the domain boundary provided by the initial coarse spatial mesh remains in the final mesh. We see this by considering Figure 5.2, which contains plots of the final meshes from each of the *h*-refinement runs, as well as a colour-plot of the distribution of $\mathbf{p}$ on the final *hp*-refinement mesh. The final mesh for the *hp*-refinement

			Primal		Dual	
Mesh	I_{eff} for $\frac{1}{k_{\text{eff}}}$	I_{eff} for k_{eff}	N	$k^{\psi}_{\text{eff},h}$	N	$k^{\hat{z}}_{\text{eff}}$
			h-refinement, $p = 1$, $q = 0$			
1	0.939785	0.943307	480	1.066059	4 224	1.003714
2	0.925801	0.926711	1 920	1.013384	16 896	1.000950
3	0.995781	0.995794	7 680	1.003023	67 584	0.999980
4	3.317106	3.316302	30 720	1.000071	270 336	0.999725
5	0.720393	0.720188	122 880	0.998950	1 081 344	0.999682
6	1.013495	1.013502	221 184	0.999445	1 966 080	0.999974
7	0.984741	0.984746	417 792	1.000296	3 735 552	0.999972
8	1.000000	1.000000	1 671 168	0.999861	14 942 208	0.999967
			h-refinement, $p = 1$, $q = 1$			
1	0.935931	0.936078	1 920	1.002450	9 504	1.000150
2	0.206110	0.206053	7 680	0.999641	38 016	0.999713
3	0.821873	0.821887	13 824	1.000091	69 120	1.000010
4	0.967777	0.967749	55 296	0.999101	276 480	0.999964
5	0.329697	0.329687	104 448	0.999947	525 312	0.999962
6	1.028219	1.028216	190 464	1.000089	960 768	0.999990
7	1.000000	1.000000	761 856	0.999830	3 843 072	0.999993
			hp-refinement			
1	0.935684	0.939446	480	1.066059	4 224	1.003714
2	0.834547	0.834927	1 920	1.002450	9 504	1.000150
3	0.968486	0.968456	4 320	0.998705	16 896	0.999664
4	0.976771	0.976760	5 760	0.999206	20 480	0.999684
5	1.049801	1.049787	7 200	0.999967	24 064	0.999682
6	1.016976	1.016980	12 800	0.999478	37 600	0.999699
7	1.076184	1.076188	15 360	0.999649	43 200	0.999699
8	1.000000	1.000000	18 944	0.999668	50 400	0.999695

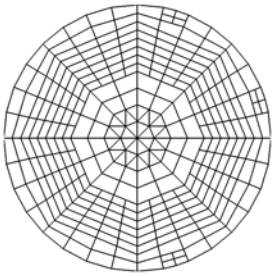
Table 5.2: The eigenvalues and effectivities for the Los Alamos benchmark problem.

	Spatial		Angular			
Mesh	N_S	η_S	N_A	η_A	Refine space	Refine angle
			h-refinement, $p = 1, q = 0$			
1	120	0.001976	4	-0.064321	.FALSE.	.TRUE.
2	120	0.001581	16	-0.014014	.FALSE.	.TRUE.
3	120	0.001440	64	-0.004483	.FALSE.	.TRUE.
4	120	0.001400	256	-0.001747	.FALSE.	.TRUE.
5	120	0.001393	1024	-0.000661	.TRUE.	.FALSE.
6	216	0.001186	1024	-0.000657	.TRUE.	.FALSE.
7	408	0.000334	1024	-0.000659	.FALSE.	.TRUE.
8	408	0.000333	4096	-0.000228	.TRUE.	.FALSE.
			h-refinement, $p = 1, q = 1$			
1	120	0.001399	16	-0.003699	.FALSE.	.TRUE.
2	120	0.001398	64	-0.001325	.TRUE.	.FALSE.
3	216	0.001197	64	-0.001277	.FALSE.	.TRUE.
4	216	0.001181	256	-0.000317	.TRUE.	.FALSE.
5	408	0.000333	256	-0.000318	.TRUE.	.FALSE.
6	744	0.000214	256	-0.000312	.FALSE.	.TRUE.
7	744	0.000214	1024	-0.000051	.TRUE.	.FALSE.
			hp-refinement			
1	120	0.001976	4	-0.064321	.FALSE.	.TRUE.
2	120	0.001399	16	-0.003699	.FALSE.	.TRUE.
3	120	0.001356	36	-0.000397	.TRUE.	.FALSE.
4	160	0.000980	36	-0.000502	.TRUE.	.FALSE.
5	200	0.000219	36	-0.000504	.FALSE.	.TRUE.
6	200	0.000201	64	0.000020	.TRUE.	.FALSE.
7	240	0.000030	64	0.000020	.TRUE.	.FALSE.
8	296	0.000025	64	0.000001	.TRUE.	.FALSE.

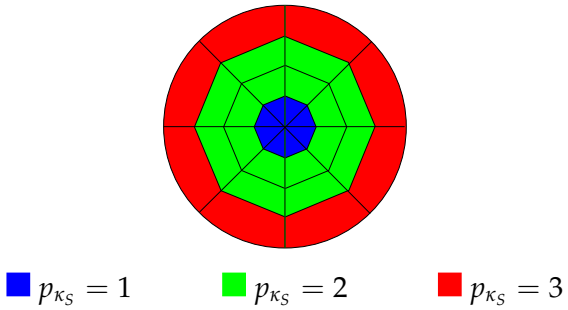
Table 5.3: The η_A and η_S for the Los Alamos benchmark problem.



(a) h-refinement, $p = 1, q = 0$.



(b) h-refinement, $p = 1, q = 1$.



(c) hp-refinement with p_{κ_S}.

Figure 5.2: The final spatial meshes from the Los Alamos benchmark.

algorithm is the same as the initial mesh, with eight spatial elements on the boundary. This is because, each time an element has been marked for refinement, the analyticity testing procedure, discussed in Section 4.4.1, has determined that either the primal or the dual solution is sufficiently smooth that p-refinement can be performed. The result is that the coarse representation of the domain boundary defined by the initial spatial mesh is retained throughout the computation. By contrast, when we consider the final spatial meshes from the h-refinement data we see that every element at the boundary has been h-refined at least once, resulting in a representation of Γ comprising at least 16 parabolic line segments, as opposed to 8 for the hp-refinement algorithm. Consequently, the meshes from the h-refinement data provide a more accurate representation of the true domain.

We remark that the hp-refinement algorithm does compute an accurate value for k_{eff}, for the spatial domain on which it computes its solution. This is evidenced by the fact that the primal eigenvalue converges towards the dual eigenvalue. Indeed the primal eigenvalue achieves four decimal places of accuracy, when referenced against the most accurate dual eigenvalue, with only 18 944 degrees of freedom in the finite

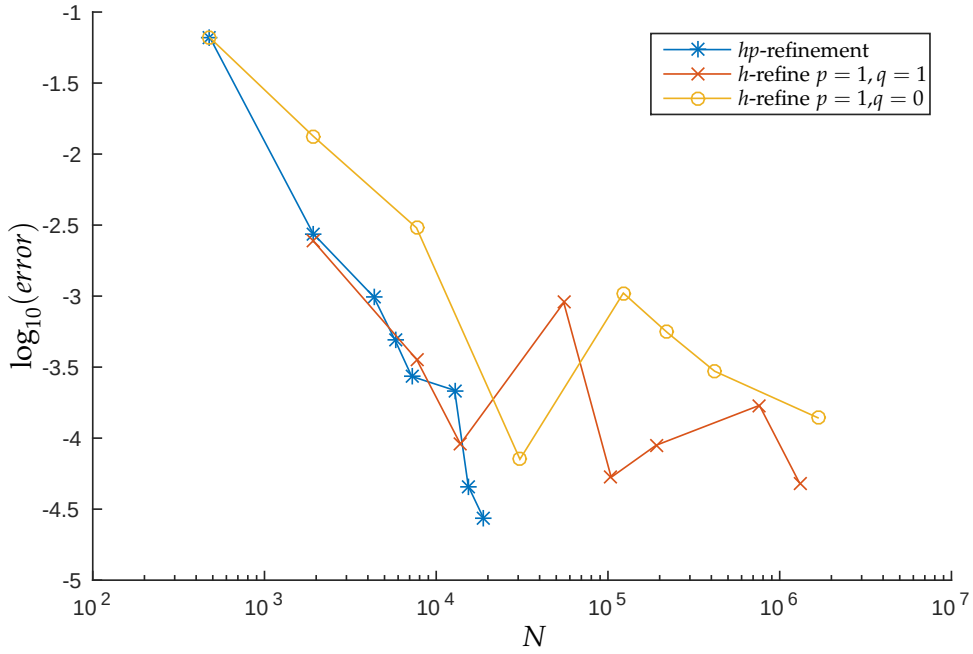


Figure 5.3: The rates of convergence for the *h*- and *hp*-refinement algorithms for the Los Alamos benchmark problem.

element space. This compares favourably with the problem size required to compute the eigenvalue to four decimal places for each of the *h*-refinement algorithms, as well as for the uniform refinement computations.

Figure 5.3 contains a log-log plot of the absolute value of the error contained in $k^{\psi}_{\text{eff},h}$ against the number of degrees of freedom in the finite element space for each of the three adaptive runs. In order to have a fair comparison between the *h*-refinement data and the *hp*-refinement data, the true solution is taken to be 0.999695 for the *hp*-refinement data, and 1.0 for the *h*-refinement data. We observe that, though eventually the *hp*-refinement algorithm outperforms the *h*-refinement algorithm, it does require seven meshes before it produces an eigenvalue that is more accurate than the third eigenvalue in the $p = 1$, $q = 1$ data. Furthermore, we note that the $p = 1$, $q = 0$ data requires a very large problem size before it achieves three decimal places of accuracy. These observations highlight the benefit of employing a high order method in angle, compared to using a zero order method such as the present $q = 0$ DG method, or the discrete ordinates method.

5.2 Larsen and Alcouffe benchmark

The second test problem considered in this chapter was proposed by Larsen and Alcouffe in a Los Alamos technical report on the long characteristics method, see [134]. This problem is considerably more challenging than the first two due to the fact that it incorporates a $G = 3$ multigroup approximation for the energy spectrum. As a consequence, three times as many degrees of freedom are required than would be required for an equivalent monoenergetic problem. The spatial domain is given in Figure 5.4. It comprises a square of fissile material at the centre, bounded on two sides by rectangular blocks of absorbing material, all surrounded by a uniform layer of shielding. The dimensions of the spatial domain are given in Figure 5.4 and the nuclear cross sections for the three regions are given in Table A.3. We note that, for multienergetic problems, when computing the dual eigenpair we must not only reverse the order of the ordered partition of the spatial elements computed by Algorithm 3.2.3, but we must also reverse the direction of scattering so that the scattering cross section $\Sigma_{s,g-1\to g}$ denotes scattering from group g to group $g-1$.

In [134], the authors quote a value of the critical eigenvalue of 0.6036, however they specify that this value is computed with a somewhat coarse S_4 approximation. In [42] the author computes a slightly more accurate value of 0.60074 for this benchmark, employing a rectangular partitioned discrete ordinates method comprising $24 \times 8 = 192$ ordinate directions. However, following our own computations, we believe that both of these approximations include a significant amount of angular error. For the computation of our convergence data we take the final dual eigenvalue computed by the *hp*-refinement algorithm to be the true value. It is $k_{\text{eff}}^{\hat{z}} = 0.599730$.

The present problem was solved using the *h*-refinement algorithm with a $p = 1$, $q = 0$ finite element space and a $p = 1$, $q = 1$ finite element space, as well as by the *hp*-refinement algorithm. The same initial spatial and angular meshes were chosen for all three runs. The initial spatial mesh is given in Figure 5.5a. The initial angular mesh comprised 4 elements, with a single angular element in each principal triangle of the hemisphere. The fixed fraction refine percentage in the spatial problem was chosen as $\alpha^{\text{FF}} = 25\%$. Table 5.4 gives the total problem sizes and computed eigenvalues from the primal and dual solves at each iteration, together with the effectivities for both k_{eff} and its reciprocal. Table 5.5 gives the spatial and angular error indicators, η_S and η_A respectively, together with the number of spatial and angular degrees of freedom in

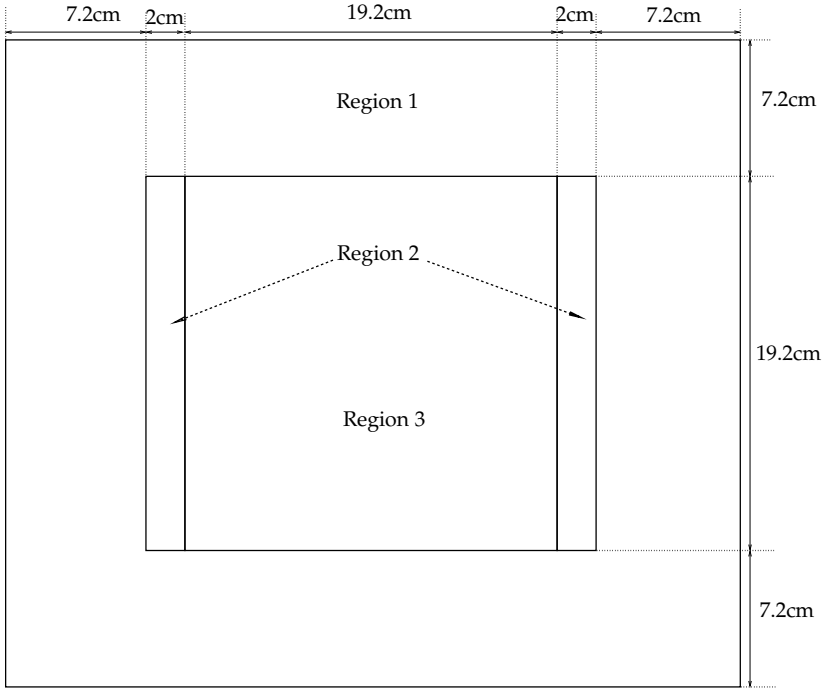
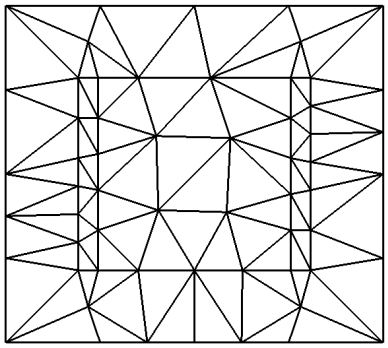


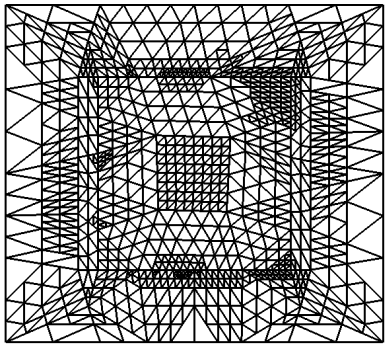
Figure 5.4: The domain for the Larsen and Alcouffe test problem. The material coefficients for the three regions marked here are given in Table A.3.

each energy group, N_S and N_A respectively. The total number of degrees of freedom in the primal problem is given by $3 \times N_S \times N_A$.

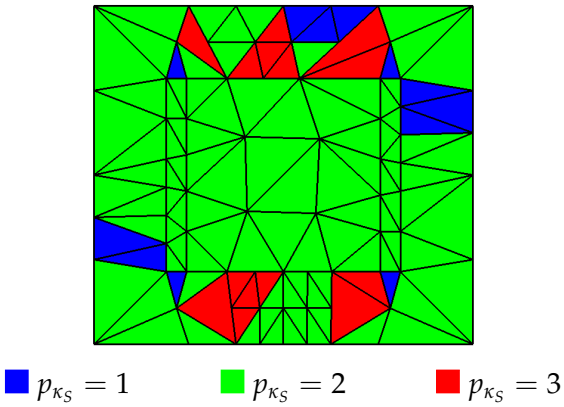
We note that all of the effectivites shown in Table 5.4 are very close to one, indicating that there is no significant over- or under-estimation of the error by our computed error representation formula. Furthermore, we note that the problem size required to compute the dual eigenpair for the present multigroup benchmark problem is considerably larger than for the previous, monoenergetic, problems. Indeed, for the *h*-refinement algorithm with $p = 1$ and $q = 0$, the dual problem size exceeds 24 000 000 degrees of freedom. Such computations are extremely time consuming, and would be infeasible without the efficient parallel solution algorithm that was developed in Chapter 3. When we compare this problem size to those from the *hp*-refinement algorithm, where the largest dual problem required fewer than 360 000 degrees of freedom, we see the benefit of utilizing a higher order method. In Table 5.5 we see the reason for such a large problem size. It is caused by the fact that there is a large amount of error in the angular variables, which necessitates repeated angular refinement. As the *hp*-refinement algorithm utilises *q*-refinement in the angular variables, it is able to eliminate the angular



(a)



(b)



(c)

Figure 5.5: Meshes from the Larsen and Alcouffe benchmark problem. 5.5a shows the initial spatial mesh employed for each algorithm. 5.5b shows the final mesh from the h-refinement algorithm with $p = 1$, $q = 1$. 5.5c shows the final mesh and distribution of $\mathbf{p}$ from the hp-refinement algorithm.

			Primal		Dual	
Mesh	I_{eff} for $\frac{1}{k_{\text{eff}}}$	I_{eff} for k_{eff}	N	$k^{\psi}_{\text{eff},h}$	N	$k^{\hat{z}}_{\text{eff}}$
			h-refinement, $p = 1, q = 0$			
1	0.961691	0.964638	3 456	0.651720	27 648	0.601585
2	0.986248	0.986416	13 824	0.607283	110 592	0.599850
3	0.886988	0.886451	55 296	0.596548	442 368	0.599384
4	1.022348	1.022280	100 224	0.601530	801 792	0.599708
5	0.922760	0.922571	400 896	0.598167	3 207 168	0.599625
6	0.964840	0.964851	767 232	0.599946	6 137 856	0.599755
7	1.000000	1.000000	3 068 928	0.598864	24 551 424	0.599748
			h-refinement, $p = 1, q = 1$			
1	0.957959	0.957559	13 824	0.593845	62 208	0.599497
2	1.100439	1.100598	25 920	0.598886	116 640	0.599834
3	0.757842	0.758113	50 976	0.600632	229 392	0.599961
4	1.021664	1.021694	203 904	0.598947	917 568	0.599765
5	0.863120	0.863152	430 272	0.599912	1 936 224	0.599770
6	1.000000	1.000000	1 721 088	0.599451	7 744 896	0.599747
7	-	-	3 518 208	0.599677	-	-
			hp-refinement			
1	0.961327	0.964302	3 456	0.651720	27 648	0.601585
2	0.960808	0.960435	13 824	0.593845	62 208	0.599497
3	1.212404	1.212591	17 568	0.599293	74 088	0.599822
4	0.833063	0.833382	21 888	0.601108	87 588	0.599959
5	1.069484	1.069567	49 248	0.599060	155 712	0.599776
6	1.197883	1.197973	65 772	0.599504	199 104	0.599774
7	0.786923	0.786958	77 976	0.599853	228 864	0.599756
8	1.000000	1.000000	138 624	0.599686	357 600	0.599730

Table 5.4: The eigenvalues and effectivities for the Larsen and Alcouffe benchmark problem.

	Spatial		Angular			
Mesh	N_S	η_S	N_A	η_A	Refine space	Refine angle
			h-refinement, $p = 1, q = 0$			
1	288	0.009386	4	-0.059520	.FALSE.	.TRUE.
2	288	0.007949	16	-0.015383	.FALSE.	.TRUE.
3	288	0.007591	64	-0.004755	.TRUE.	.FALSE.
4	522	0.003064	64	-0.004886	.FALSE.	.TRUE.
5	522	0.003000	256	-0.001542	.TRUE.	.FALSE.
6	999	0.001370	256	-0.001561	.FALSE.	.TRUE.
7	999	0.001359	1024	-0.000476	.TRUE.	.FALSE.
			h-refinement, $p = 1, q = 1$			
1	288	0.007489	16	-0.001837	.TRUE.	.FALSE.
2	540	0.002913	16	-0.001965	.TRUE.	.FALSE.
3	1062	0.001371	16	-0.002042	.FALSE.	.TRUE.
4	1062	0.001339	64	-0.000521	.TRUE.	.FALSE.
5	2241	0.000398	64	-0.000540	.FALSE.	.TRUE.
6	2241	0.000386	256	-0.000090	.TRUE.	.FALSE.
7	4581	-	256	-	-	-
			hp-refinement			
1	288	0.009386	4	-0.059520	.FALSE.	.TRUE.
2	288	0.007489	16	-0.001837	.TRUE.	.FALSE.
3	366	0.002483	16	-0.001953	.TRUE.	.FALSE.
4	456	0.000893	16	-0.002042	.FALSE.	.TRUE.
5	456	0.000877	36	-0.000161	.TRUE.	.FALSE.
6	609	0.000441	36	-0.000171	.TRUE.	.FALSE.
7	722	0.000078	36	-0.000175	.FALSE.	.TRUE.
8	722	0.000073	64	-0.000030	.TRUE.	.FALSE.

Table 5.5: The η_A and η_S for the Larsen and Alcouffe benchmark problem.

error much more efficiently. Indeed, we observe that the angular discretisation with 36 degrees of freedom in each energy group, from iterations 5,6 and 7 of the *hp*-refinement algorithm, achieves a more accurate angular solution than the finest angular scheme employed in the $p = 1$, $q = 0$ *h*-refinement algorithm, which employs 1024 degrees of freedom in each energy group.

Figure 5.5 contains plots of the final mesh from the *h*-refinement algorithm with $p = 1$ and $q = 1$, as well as a colour-plot of the distribution of the p_{κ_S} on the final mesh from the *hp*-refinement algorithm. We note that both of these algorithms target the border between the fissile material and the shielding material for the most refinement. Relatively less refinement occurs at the boundary with the absorbing material. Furthermore, we note that the *hp*-refinement algorithm primarily selects *p*-refinement, with only 5 spatial elements undergoing *h*-refinement. This signifies a significant amount of smoothness in the computed eigenfunction.

Figure 5.6 contains a log-log plot of the absolute value of the error in the primal eigenvalue against the total primal problem size for each of the runs. We observe that the *hp*-refinement algorithm is significantly more efficient than the *h*-refinement algorithm.

5.3 KNK fast reactor benchmark

The next test problem is a model of the *Kompakte Natriumgekühlte Kernreaktoranlage* (Compact Sodium Nuclear Reactor Plant) in Germany. We shall refer to it as the KNK fast reactor benchmark. It was originally published by Takeda and Ikeda in [135] and has since been computed by several authors, including Kim and Cho, in [136], Wang, in [137] and Baker, in [42]. The spatial domain is considerably more complicated than the previous problems, comprising a tessellation of 169 regular hexagons split into 18 regions with 8 distinct materials. The energy spectrum is discretised into $G = 4$ energy groups. The spatial domain is given in Figure 5.7. Takeda and Ikeda published three sets of data for the KNK fast reactor benchmark, modelling the case with control rods inserted, control rods half inserted and the unrodded case, respectively. We compute eigenvalues for the unrodded case, for which the material cross sections are given in Tables A.4 and A.5.

In [135], the authors published several approximate values of the critical eigenvalue, computed employing a variety of numerical methods, ranging between 1.0887 and

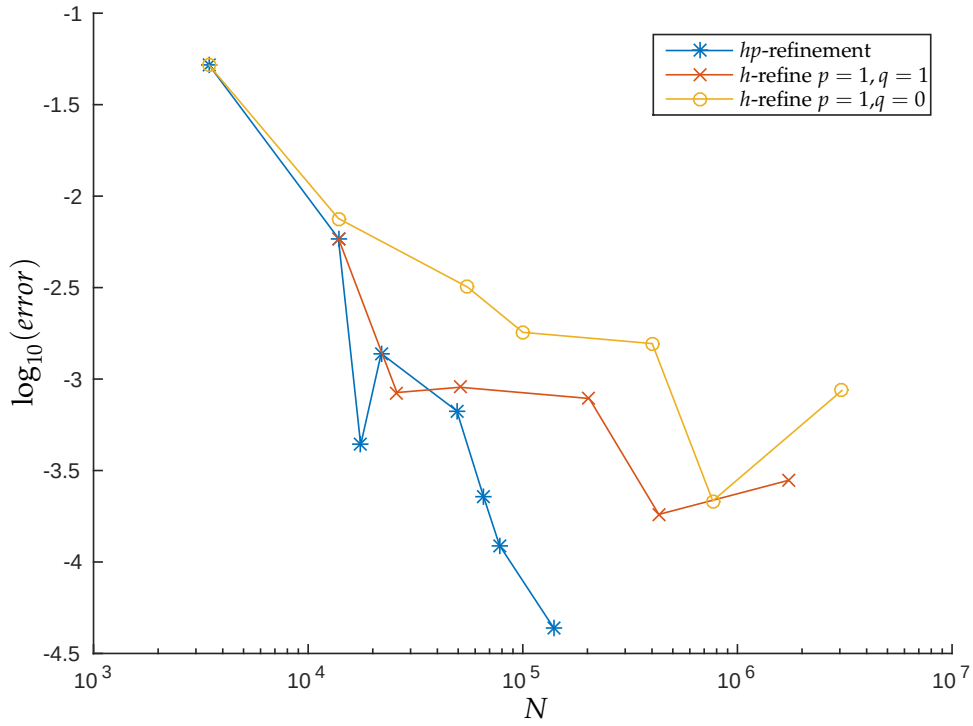


Figure 5.6: The rates of convergence for the *h*- and *hp*-refinement algorithms for the Larsen and Alcouffe benchmark problem.

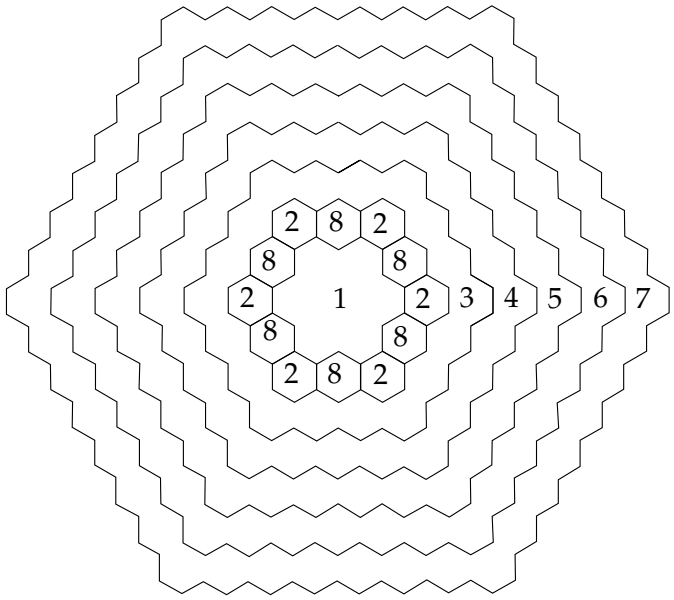
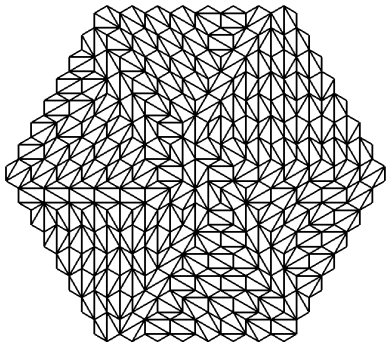


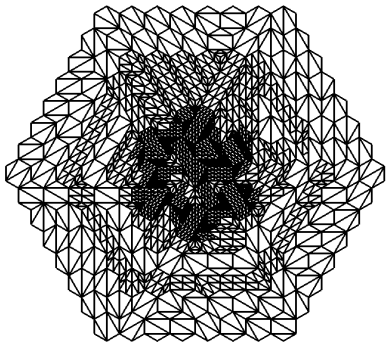
Figure 5.7: The domain for the KNK test problem. The domain is composed of a tessellation of regular hexagons, each of which has sides of length 7.5cm. The numbers mark the eight different regions, the nuclear cross sections for these are given in Tables A.4 and A.5.

1.0951. More recently, Wang published a value of 1.01082, Baker computed a value of 1.0105 and Kim and Cho published values ranging between 1.0094 and 1.01055. The most accurate dual eigenvalues computed by our codes are 1.010316 and 1.010370 in the $h-$ and $hp-$ refinement codes respectively. For the purposes of error quantification in the present code we take the true k_{eff} to be the most accurate dual eigenvalue computed by the hp solver, 1.010370. The critical eigenvalue for the KNK fast reactor benchmark was computed by the h-refinement algorithm with a $p = 1$, $q = 0$ finite element space, as well as by the hp-refinement algorithm. The same initial spatial and angular meshes were chosen for both runs. The initial spatial mesh is given in Figure 5.8a. The initial angular mesh comprised 4 elements, with a single angular element in each principal triangle of the hemisphere. The fixed fraction refine percentage in the spatial problem was chosen as $\alpha^{\text{FF}} = 25\%$.

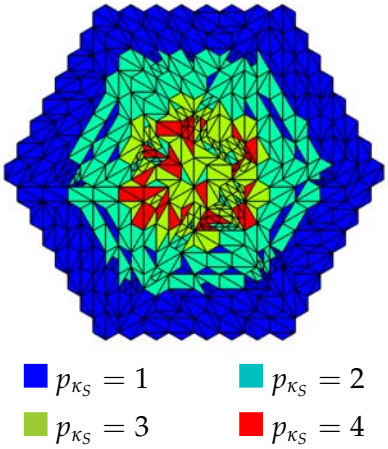
Table 5.6 gives the total problem sizes and computed eigenvalues from the primal and dual solves at each iteration, together with the effectivities for both k_{eff} and its reciprocal. Table 5.7 gives the spatial and angular error indicators, η_S and η_A respectively, together with the number of spatial and angular degrees of freedom in each energy



(a)



(b)



(c)

Figure 5.8: Meshes from the KNK fast reactor benchmark problem. 5.8a shows the initial spatial mesh employed for each algorithm. 5.8b shows the final mesh from the h-refinement algorithm with $p = 1$, $q = 0$. 5.8c shows the final mesh and distribution of $\mathbf{p}$ from the hp-refinement algorithm.

			Primal		Dual	
Mesh	I_{eff} for $\frac{1}{k_{\text{eff}}}$	I_{eff} for k_{eff}	N	$k^{\psi}_{\text{eff},h}$	N	$k^{\hat{z}}_{\text{eff}}$
h-refinement, $p = 1, q = 0$						
1	0.954984	0.956818	32 448	1.055231	259 584	1.012255
2	1.032756	1.032598	129 792	1.015041	1 038 336	1.010162
3	0.939630	0.939274	519 168	1.004013	4 153 344	1.009933
4	1.018679	1.018634	931 584	1.012680	7 452 672	1.010272
5	0.928592	0.928522	3 726 336	1.009261	29 810 688	1.010240
6	1.000000	1.000000	7 412 736	1.011165	59 301 888	1.010316
hp-refinement						
1	0.956193	0.957977	32 448	1.055231	259 584	1.012255
2	0.941291	0.940792	129 792	1.001320	584 064	1.009834
3	1.193111	1.192740	166 080	1.011999	701 568	1.010056
4	0.744608	0.744460	373 680	1.009584	1 247 232	1.010169
5	1.000000	1.000000	532 368	1.009999	1 671 936	1.010370
6	-	-	740592	1.010223	-	-

Table 5.6: The eigenvalues and effectivities for the KNK fast reactor benchmark.

group, N_S and N_A respectively. We note that much more refinement was required in the angular domain than in the spatial domain. This is because the initial spatial mesh needed to be quite fine in order to accurately represent the problem geometry, whereas the initial angular mesh was extremely coarse. We note that the effectivities presented in Table 5.6 are very close to one, once again confirming the accuracy of our DWR error representation formula. Furthermore, we note a rapid increase in the problem size as the h-refinement algorithm progresses, particularly for the dual problem. We note that the problem size increases less quickly as the hp-refinement algorithm progresses.

Figure 5.8b contains the refined spatial mesh from the h-refinement run. Figure 5.8c contains a colour plot of the distribution of $\mathbf{p}$ in the final mesh from the hp-refinement data. We note that both of these algorithms target most of the refinement at the central region. This is where the test zone, control rods and driver are located. There is also a limited amount of refinement in region 5, which contains reflecting material, with a moderator. We note that, once again, the majority of refinement undertaken by the hp-refinement algorithm is p-refinement, however there is some h-refinement, signifying that there are areas where the computed solution is not perfectly smooth.

Figure 5.9 contains a log-log plot of the absolute value of the error in the primal eigen-

	Spatial		Angular			
Mesh	N_S	η_S	N_A	η_A	Refine space	Refine angle
			h-refinement, $p = 1, q = 0$			
1	2 028	0.012217	4	-0.055193	.FALSE.	.TRUE.
2	2 028	0.011060	16	-0.015939	.FALSE.	.TRUE.
3	2 028	0.010608	64	-0.004688	.TRUE.	.FALSE.
4	3 639	0.002454	64	-0.004862	.FALSE.	.TRUE.
5	3 639	0.002417	256	-0.001438	.TRUE.	.FALSE.
6	7 239	0.000602	256	-0.001451	.FALSE.	.TRUE.
			hp-refinement			
1	2 028	0.012217	4	-0.055193	.FALSE.	.TRUE.
2	2 028	0.010654	16	-0.002139	.TRUE.	.FALSE.
3	2 595	0.000498	16	-0.002441	.FALSE.	.TRUE.
4	2 595	0.000481	36	0.000105	.TRUE.	.FALSE.
5	3 697	0.000256	36	0.000115	.TRUE.	.FALSE.

Table 5.7: The η_A and η_S for the KNK fast reactor benchmark problem.

value against the total primal problem size for each of the runs. We observe that the convergence of the hp-refinement algorithm reduces the error to just over 10^{-4}, whereas the h-refinement algorithm fails to achieve the same accuracy despite using 10 times as many degrees of freedom in its finite element space.

5.4 SILENE reactor

We now consider two benchmark problems from a set compiled by Albrecht Kyrieleis at Serco Assurance, [132]. The first of these is a model of an experimental reactor in France called the SILENE reactor. It comprises a steel cylindrical tank filled with uranyl nitrate with a cylindrical steel pipe positioned vertically down the middle through which a control rod may be inserted, see Figure 5.10 for the exact dimensions. We model the unrodded case with a $G = 2$ multigroup approximation for energy, the cross sections for which are reproduced in Table A.6. For this problem, the MONK Monte Carlo code computed a value of 1.1293 for the critical eigenvalue. The two most accurate dual eigenvalues that we computed were 1.129432 and 1.129459. For the purpose of computing the convergence data for our adaptive algorithms we take the most accurate dual eigenvalue from the hp solver to be the true eigenvalue: 1.129459.

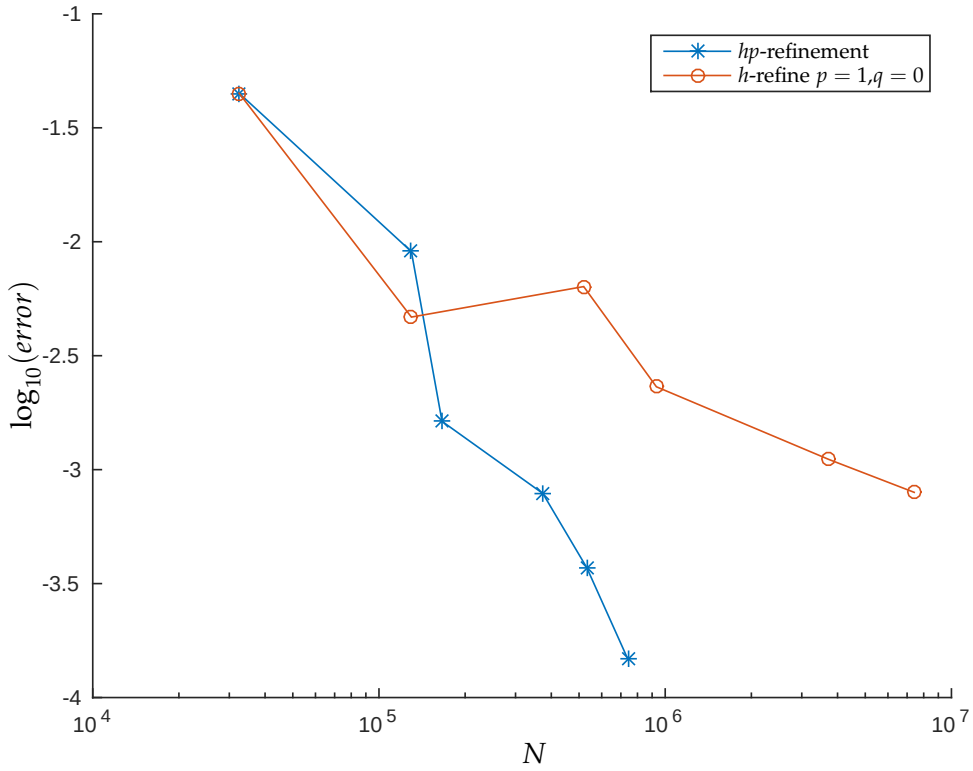


Figure 5.9: The rates of convergence for the *h*- and *hp*-refinement algorithms for the KNK benchmark problem.

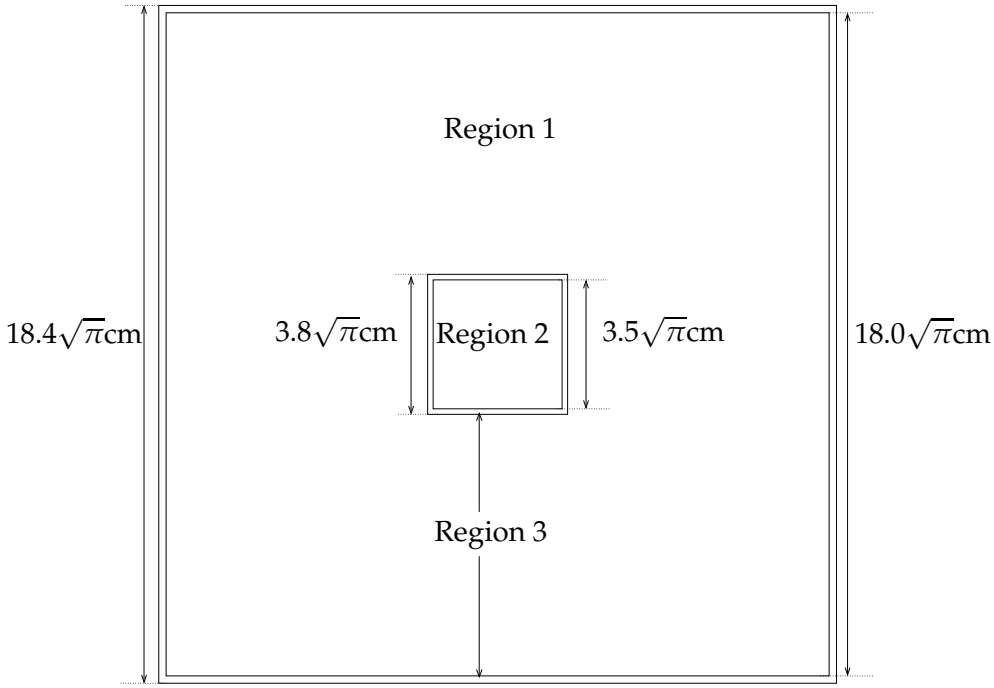
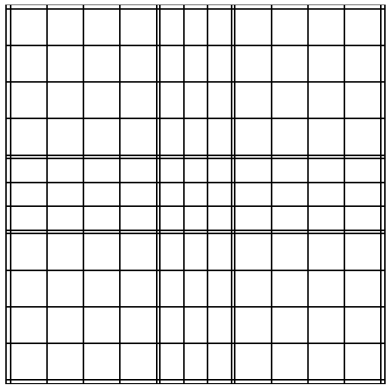
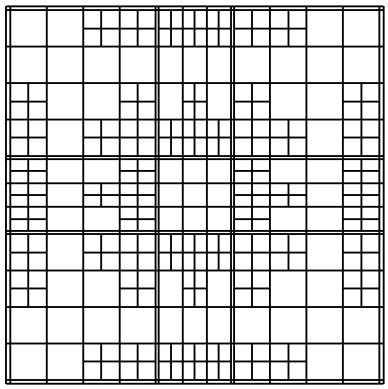


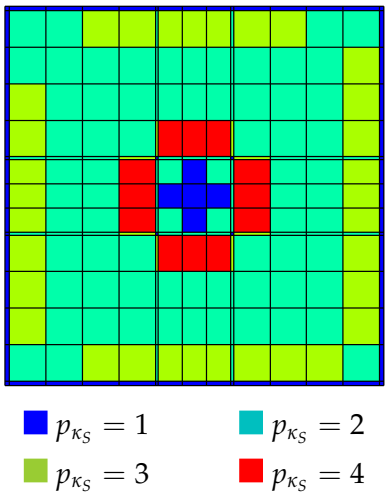
Figure 5.10: The domain for the Serco SILENE test problem. The region boundaries are all squares, therefore the horizontal dimensions are the same as the vertical ones. The material coefficients for the three regions are given in Table A.6.



(a)



(b)



(c)

Figure 5.11: Meshes from the SILENE reactor benchmark problem. 5.11a shows the initial spatial mesh employed for each algorithm. 5.11b shows the final mesh from the h-refinement algorithm with $p = 1, q = 0$. 5.11c shows the final mesh and distribution of $\mathbf{p}$ from the hp-refinement algorithm.

			Primal		Dual	
Mesh	I_{eff} for $\frac{1}{k_{\text{eff}}}$	I_{eff} for k_{eff}	N	$k^{\psi}_{\text{eff},h}$	N	$k^{\hat{z}}_{\text{eff}}$
h-refinement, $p = 1$, $q = 0$						
1	0.939108	0.942367	7 200	1.197436	64 800	1.133351
2	0.984227	0.984492	28 800	1.149016	259 200	1.129736
3	1.010127	1.010088	115 200	1.133840	1 036 800	1.129388
4	0.587143	0.587102	460 800	1.129241	4 147 200	1.129353
5	1.000000	1.000000	804 864	1.130390	7 243 776	1.129432
6	-	-	3 219 456	1.129013	-	-
hp-refinement						
1	0.939500	0.942738	7 200	1.197436	64 800	1.133351
2	1.083026	1.082883	28 800	1.131261	145 800	1.129309
3	0.947996	0.947900	64 800	1.127252	259 200	1.129344
4	0.957356	0.957328	84 960	1.128662	309 376	1.129425
5	0.960837	0.960828	105 696	1.129176	360 576	1.129448
6	0.945649	0.945645	128 160	1.129363	414 848	1.129454
7	1.000000	1.000000	156 384	1.129415	479 360	1.129459
8	-	-	278 016	1.129450	-	-

Table 5.8: The eigenvalues and effectivities for the Serco SILENE benchmark.

The critical eigenvalue for this benchmark was computed by the h-refinement algorithm with a $p = 1$, $q = 0$ finite element space, as well as by the hp-refinement algorithm. The same initial spatial and angular meshes were chosen for both runs. The initial spatial mesh is given in Figure 5.11a. Note that we have utilised long, thin elements to represent the thin layers of steel around the central region and at the domain boundary. The initial angular mesh comprised 4 elements, with a single angular element in each principal triangle of the hemisphere. The fixed fraction refine percentage in the spatial problem was chosen as $\alpha^{\text{FF}} = 25\%$. Table 5.8 gives the total problem sizes and computed eigenvalues from the primal and dual solves at each iteration, together with the effectivities for both k_{eff} and its reciprocal. As for all previous benchmark problems, we observe excellent effectivities. Table 5.9 gives the spatial and angular error indicators, η_S and η_A respectively, together with the number of spatial and angular degrees of freedom in each energy group, N_S and N_A respectively.

Figure 5.11b shows the final spatial mesh from the h-refinement algorithm. We note that all of the refinement has taken place in the uranyl nitrate, with no refinement taking place in the inner pipe or the steel shielding. This is in contrast to the hp-refinement

	Spatial		Angular			
Mesh	N_S	η_S	N_A	η_A	Refine space	Refine angle
h-refinement, $p = 1, q = 0$						
1	900	0.002228	4	-0.066313	.FALSE.	.TRUE.
2	900	0.002083	16	-0.021364	.FALSE.	.TRUE.
3	900	0.002056	64	-0.006509	.FALSE.	.TRUE.
4	900	0.002044	256	-0.001932	.TRUE.	.FALSE.
5	1572	0.000982	256	-0.001940	.FALSE.	.TRUE.
6	1572	-	1024	-	-	
hp-refinement						
1	900	0.002228	4	-0.066313	.FALSE.	.TRUE.
2	900	0.002052	16	-0.004003	.FALSE.	.TRUE.
3	900	0.002040	36	0.000051	.TRUE.	.FALSE.
4	1 180	0.000722	36	0.000040	.TRUE.	.FALSE.
5	1 468	0.000231	36	0.000040	.TRUE.	.FALSE.
6	1 780	0.000055	36	0.000036	.TRUE.	.FALSE.
7	2 172	0.000012	36	0.000031	.FALSE.	.TRUE.
8	2 172	-	64	-	-	-

Table 5.9: The η_A and η_S for the Serco SILENE benchmark problem.

data, where the error in the nuclear fuel was more quickly eliminated by *p*-refinement, with the algorithm then detecting and eliminating error in the inner tube and the steel pipe. Figure 5.11c contains a colour-plot of the distribution of $\mathbf{p}$ on the final iteration of the *hp*-refinement algorithm. Notice that this algorithm has exclusively selected *p*-refinement, with no *h*-refinement taking place. This is because, for each spatial element that is marked for refinement, either the local primal solution or the local dual solution was found to be smooth. However, the solution eigenvector for this problem does have some very sharp features. In particular, when moving from the inner region, through the steel pipe to the nuclear fuel, there is a sudden and extreme increase in the scalar flux. However, the fact that we have designed a mesh with narrow finite elements isolating the air from the fuel means that the numerical approximation remains smooth on either side of this sharp gradient. Indeed, this can be seen as an example of why *hp*-refinement is so efficient at finding numerical approximations to problems whose solutions contain sharp features: *h*-refinement enables the non-smooth parts of the solution to be isolated from the smooth parts, where *p*-refinement can be used to obtain locally exponential convergence.

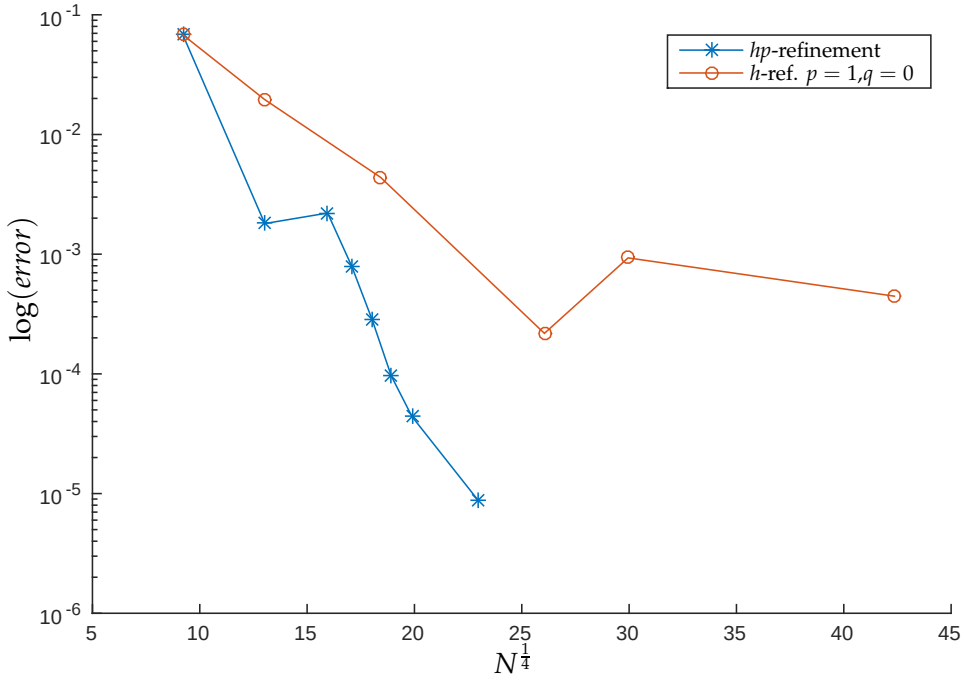


Figure 5.12: The $\log_{10}$ of the error against the fourth root of N for the h- and hp-refinement algorithms for the SILENE benchmark problem.

The hp-refinement algorithm seems to be particularly efficient for solving this benchmark problem. In order to investigate its convergence we consider a plot of the log of the error against N^s, where we take s to be $\frac{1}{4}$, see Figure 5.12. In such a plot, exponential convergence would be signified by a straight line with a negative gradient. Considering the third to the sixth data points of the hp refinement data, ignoring the second point where we can see from Table 5.9 there has been significant error cancelling, it does appear as if the line of convergence is indeed straight and with a negative gradient. Therefore we can conclude that this method does converge exponentially quickly, at least until the final point breaks the trend.

5.5 Water-cooled reactor

The second problem that we consider from [132] is a model of a water-cooled reactor. The spatial domain comprises a central region containing the nuclear fuel, surrounded by a steel reflector, which is itself surrounded by water, see Figure 5.13 for the exact dimensions. The model incorporates a $G = 2$ multigroup approximation for energy, the

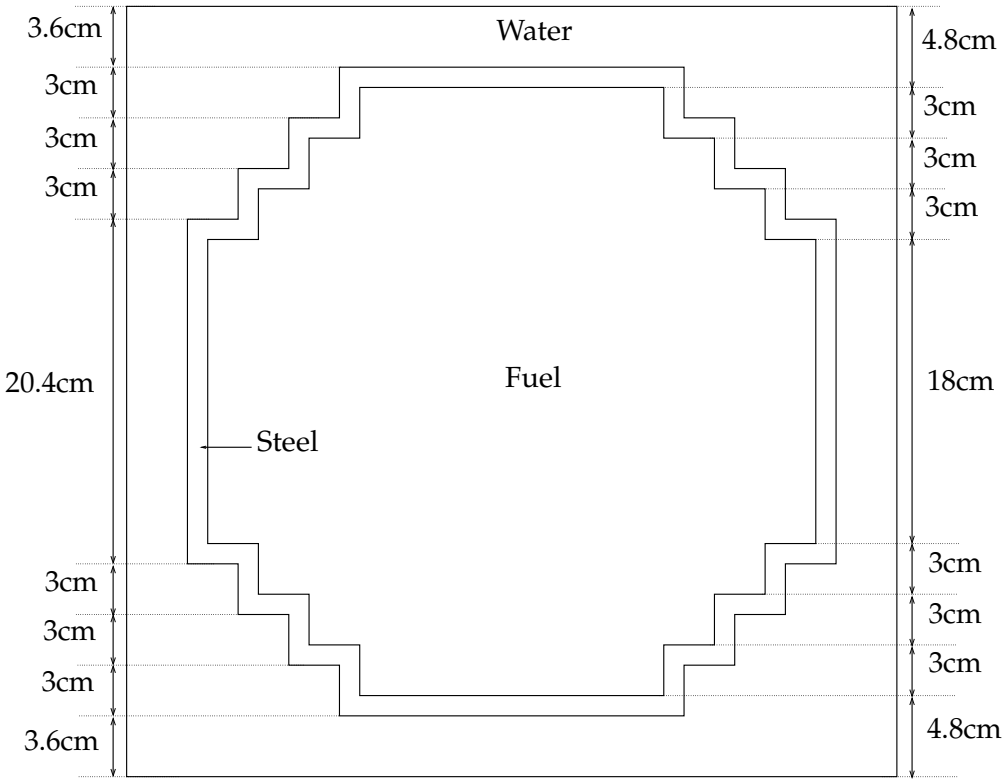
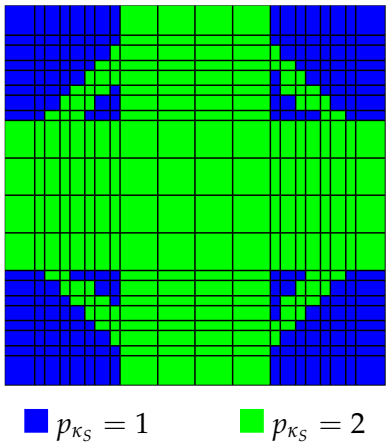


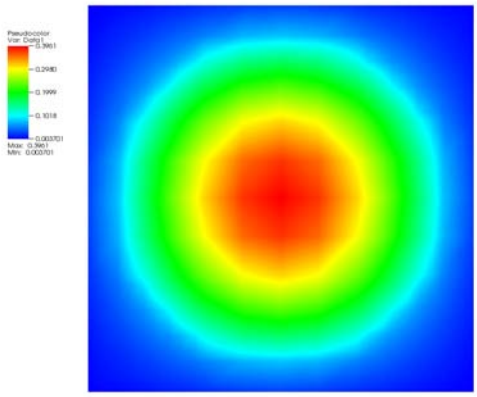
Figure 5.13: The domain for the water-cooled reactor test problem.

nuclear cross sections for which are given in Table A.7. When applied to this problem, the MONK Monte Carlo code computed a critical eigenvalue of 1.0118. The short characteristics code of Baker [42] computed a value of 1.0129. However, we find that our most accurate dual eigenvalues from each algorithm both take the value 1.012132. For the purposes of computing convergence data we take 1.012132 to be the true solution.

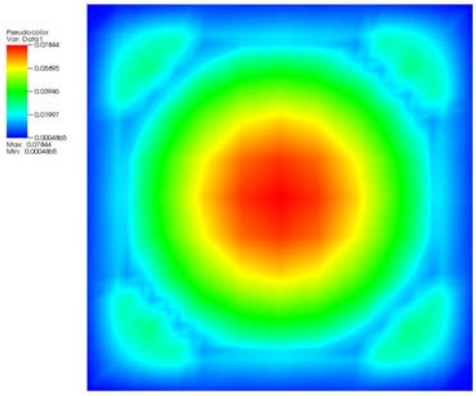
The critical eigenvalue for the water-cooled reactor was computed by the *h*-refinement algorithm with a $p = 1$, $q = 0$ finite element space, as well as by the *hp*-refinement algorithm. The same initial spatial and angular meshes were chosen for both runs. The initial spatial mesh is given in Figure 5.14a. The initial angular mesh comprised 4 elements, with a single angular element in each principal triangle of the hemisphere. The fixed fraction refine percentage in the spatial problem was chosen as $\alpha^{\mathrm{FF}} = 25\%$. Table 5.10 gives the total problem sizes and computed eigenvalues from the primal and dual solves at each iteration, together with the effectivities for both k_{eff} and its reciprocal. Notice that these effectivities are very close to unity, confirming the accuracy of our error estimation.



(a)



(b)



(c)

Figure 5.14: Water-cooled reactor benchmark problem. 5.14a shows the mesh and final distribution of **p** from the *hp*-refinement algorithm. As no *h*-refinement took place, this is the same as the original mesh. 5.14b and 5.14c contain the space averaged flux for the critical eigenfunction in the high and low energy group, respectively.

Mesh	I_{eff} for $\frac{1}{k_{\text{eff}}}$	I_{eff} for k_{eff}	Primal N	Primal $k^{\psi}_{\text{eff},h}$	Dual N	Dual $k^{\hat{z}}_{\text{eff}}$
h-refinement, $p = 1$, $q = 0$						
1	0.950727	0.953365	12 800	1.072370	115 200	1.014941
2	0.987397	0.987611	51 200	1.029824	460 800	1.012351
3	0.996965	0.996979	204 800	1.016790	1 843 200	1.012146
4	1.000000	1.000000	819 200	1.012910	7 372 800	1.012132
5	-	-	3 276 800	1.011785	-	-
hp-refinement						
1	0.950726	0.953365	12 800	1.072370	115 200	1.014941
2	1.015304	1.015273	51 200	1.014119	259 200	1.012101
3	0.993352	0.993347	115 200	1.011316	460 800	1.012126
4	1.000000	1.000000	151 200	1.011894	550 400	1.012132
5	-	-	187 200	1.012099	-	-

Table 5.10: The eigenvalues and effectivities for the water-cooled reactor benchmark.

Table 5.11 gives the spatial and angular error indicators, η_S and η_A respectively, together with the number of spatial and angular degrees of freedom in each energy group, N_S and N_A respectively. From the values of η_S we see that the initial spatial scheme provides a very accurate spatial approximation. Indeed, every spatial error indicator is computed to be less than 10^{-3}. As a consequence, the refinement for each algorithm is initially targeted towards the angular variables. In fact, for the h-refinement algorithm, spatial refinement is never selected. For the hp-refinement algorithm, where q-refinement eliminates the error much more efficiently, spatial refinement is not selected until the fourth iteration.

The scalar flux of the critical eigenfunction at energy groups 1 and 2 are given in Figures 5.14b and 5.14c, respectively. The spatial solution appears to be very smooth at the high energy group ($g = 1$), whereas there is more structure in the solution at the low energy group ($g = 2$). Furthermore, we note that the peak value of the scalar flux in the high energy group is five times higher than the peak value in the low energy group. Figure 5.14a shows the mesh and the distribution of $\mathbf{p}$ from the hp-refinement algorithm. We note that the hp-adaptive algorithm has chosen p-refinement for every element that was selected for refinement. This indicates that the computed primal and dual solutions are smooth within all of the spatial elements that were refined. Moreover, we note that the refinement algorithm has identified the structures in the solution at the low energy

	Spatial		Angular			
Mesh	N_S	η_S	N_A	η_A	Refine space	Refine angle
h-refinement, $p = 1, q = 0$						
1	1600	0.000854	4	-0.058283	.FALSE.	.TRUE.
2	1600	0.000806	16	-0.018279	.FALSE.	.TRUE.
3	1600	0.000795	64	-0.005439	.FALSE.	.TRUE.
4	1600	0.000791	256	-0.001570	.FALSE.	.TRUE.
hp-refinement						
1	1600	0.000854	4	-0.058283	.FALSE.	.TRUE.
2	1600	0.000791	16	-0.002808	.FALSE.	.TRUE.
3	1600	0.000789	36	0.000021	.TRUE.	.FALSE.
4	2100	0.000212	36	0.000026	.TRUE.	.FALSE.
5	2600	-	36	-	-	-

Table 5.11: The η_A and η_S for the Serco water-cooled reactor benchmark problem.

group, as well as the extreme values in the high energy group.

As for the SILENE problem, the *hp*-refinement algorithm seems to be highly efficient for solving this benchmark problem. In order to investigate whether we are achieving exponential convergence we consider a plot of the log of the error against N^s, where we take s to be $\frac{1}{4}$, see Figure 5.15. We note that the gradient of the line of convergence for the *hp*-refinement algorithm is increasing. Therefore, for the data points that we have, the algorithm is converging at a rate that is exponential.

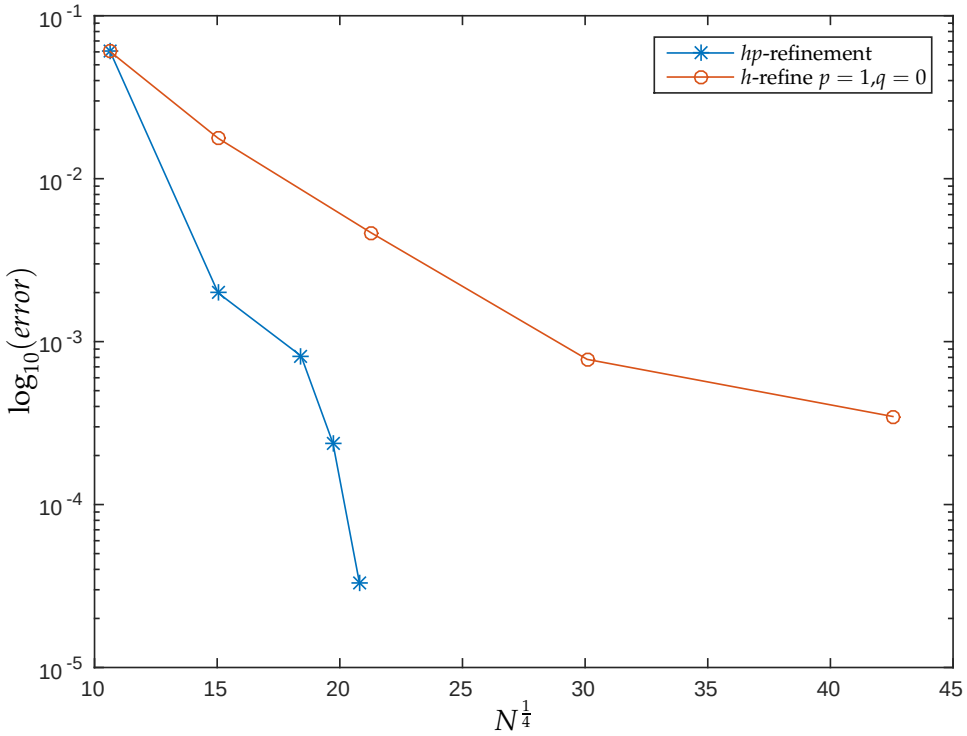


Figure 5.15: The $\log_{10}$ of the error against the fourth root of the total problem size for the *h*- and *hp*-refinement algorithms for the water-cooled reactor benchmark problem.

CHAPTER 6

Conclusions

In this thesis we have investigated the numerical approximation of partial differential equations by discontinuous-Galerkin (DG) finite element methods. We began by considering solution strategies that take advantage of the structure of the matrices which arise from the application of DG methods to general PDEs. We then focused on problems arising in the field of neutron transport. In particular, we focused on criticality eigenvalue problems. In this context we developed a high order discretisation of the Boltzmann transport equation and implemented *h*- and *hp*-adaptive algorithms for the computation of the effective multiplication factor, k_{eff}.

In Chapter 2 we considered a method for preprocessing DG matrices for sparse direct solvers. It was based on analysing the structure of the dense blocks associated with the finite element mesh, where each finite element and each interface between finite elements corresponds to one or two dense blocks, respectively. The computed block ordering was then expanded to correspond to a pivot order for the full linear system. Such a method encourages the solver to keep the degrees of freedom associated with each finite element together during the factorisation phase of the solve. It was tested for two direct solvers from the HSL software library [86]: the `MA57` symmetric linear solver and the `MA41` unsymmetric linear solver. Significant improvements in the performance of each solver was noted, both in terms of the amount of memory required, and the amount of time required to analyse, factorise and solve a selection of DG matrices. The `MA57` solver saw the greatest improvement in its performance when moving to the bespoke preprocessor. When tested on matrices arising from two dimensional geometries, a reduction in CPU time of between 12% and 39% was observed. The memory requirements were reduced by between 5% and 19%. For a large matrix arising from a

three dimensional geometry, memory requirements were reduced by 35%.

In the following chapter we derived the DG discretisation of the neutron transport equation which incorporated high order finite elements in both the spatial and the angular domains. The asymptotic rate of convergence for the spatial part of the discretisation error, measured in the L_2-norm of the scalar flux, was shown to be of order h^{p+1} in agreement with the theoretical value, where p is the order of polynomial approximation in the spatial variables. The angular discretisation was found to converge at the rate $\mathcal{O}(h^{q+2})$, when computed with respect to the same norm, where q is the order of polynomial approximation in the angular variables. Fast Krylov subspace based solvers for the neutron transport source and k_{eff}-eigenvalue problem were then investigated. These solvers utilise a preconditioner which was based on computing the action of the inverse of the matrix $\mathbf{T}$, which comprises the streaming, absorption and boundary terms from the discretisation. This preconditioner employed Tarjan's strongly connected components algorithm in order to determine an ordered partition of the spatial elements for each angular element. These orderings, together with the efficient preprocessor for block matrices developed in the previous chapter, enabled the parallel, unassembled computation of the action of $\mathbf{T}^{-1}$. The Krylov subspace solvers were shown to be robust when the problem size was increased by introducing more finite elements to the mesh, as well as when the order of polynomial approximation in the underlying finite element space was increased.

In Chapter 4 we derived a dual weighted residual error representation formula for neutron transport criticality problems. This, together with a projection-based error splitting, enabled the computation of accurate *a posteriori* indicators for the error stemming from the spatial and angular parts of the discretisation. The restriction of these quantities to individual elements in the spatial and angular meshes yielded elementwise spatial and angular error indicators, respectively, thus facilitating the development of an h-adaptive algorithm for the computation of the effective multiplication factor, k_{eff}. These error indicators were then used in conjunction with an analyticity testing procedure to develop an hp-adaptive algorithm for k_{eff} problems. Numerical experiments for a monoenergetic test problem led to effectivities of close to one for both of these algorithms, confirming the quality of the computed error estimation formula. Furthermore, these numerical tests demonstrated that the hp-adaptive method could compute the critical eigenvalue to the same accuracy as the h-adaptive method, but with a finite element space containing approximately 80% fewer degrees of freedom. Finally, in

Chapter 5 we presented results from a series of industrial benchmark problems. These incorporated multigroup discretisations for the energy variable that greatly increased the size and difficulty of the problems at hand. Despite this, accurate eigenvalues were computed and effectivities close to one were obtained.

6.1 Further work

The results collected in this thesis suggest several avenues of further research. Firstly, the efficiency gains made when solving DG matrices with `MA41` and `MA57` by utilising the underlying block structure during preprocessing could further be enhanced by exploiting these structures in the `FACTORISE` and `SOLVE` phases of the sparse direct solvers. In particular, if an efficient test could be developed for determining whether a diagonal block was singular, or close to singular, then it should be possible to construct a sparse block unsymmetric solver that factorises all degrees of freedom associated with each finite element at the same time. Such a solver could utilise efficient level 3 BLAS and LAPACK routines to achieve a highly efficient factorisation for high order DG matrices.

There are three principal ways in which the methods for neutron transport problems described herein could be further developed in order to more faithfully represent the physical systems being modelled. Firstly, incorporating the potential for anisotropic scattering of neutrons would lead to more physically relevant solutions, though this would come at the cost of increased computational expense for the computation of the action of the scattering matrix $\mathbf{S}$. Even more computationally demanding would be to extend the spatial domain to model the full three dimensional space, as in [87]. Such an extension would be necessary to obtain a truly accurate representation of the physical systems considered; however, the problem size could grow prohibitively large without significant enhancements to the present code. This is because we would now be required to solve, in the monoenergetic case, a five dimensional integro-differential eigenvalue equation. Indeed, in order to solve a source problem in the field of radiation transport, the authors of [87] needed to run their discrete ordinates solver in parallel on hundreds of cores. The computation of the k_{eff}-eigenvalue in such a geometry would be considerably more challenging. Finally, the recent development of *hp*-finite element methods with general polygonal elements, see [138], could be exploited in or-

der to retain a faithful representation of the physical domain for complicated reactor geometries, whilst reducing the overall problem size. Such methods define their basis functions in physical space on a bounding box containing the polygonal finite element, which is then restricted to the element. These methods permit the modelling of fine details in the spatial domain that could not otherwise be modelled without employing a large number of smaller elements.

The speed of the software could also potentially be improved in three ways. Firstly, the incorporation of reflective boundary conditions could enable a significant reduction in the size of the spatial domain, and hence the overall problem size, when permitted by symmetries in the spatial geometry. For example, the Larsen and Alcouffe benchmark problem from Section 5.2 could be computed to the same accuracy with a 75% reduction in the number of degrees of freedom required if reflective boundaries were placed along the horizontal and vertical lines of symmetry. Then the critical eigenvector could be computed in a quarter of the original spatial domain, yielding the same value for the critical eigenvalue with a large reduction in both the amount of memory and the number of floating point operations required. Secondly, if large three dimensional problems were to be tackled, it would be necessary to utilise distributed processing environments. For the two dimensional problems considered in this thesis, the spatial domains were sufficiently small that transport sweeps could be efficiently computed for the full spatial domain on a single processor for each angular element. However, in [87], the authors found that for large 3D problems, it was necessary to consider a decomposition of the spatial domain so that the full spatial geometry did not need to be stored on each processor. In this case multiple processors were used for the transport sweeps in each ordinate direction. Such a strategy raises significant computational challenges, including the necessity to time the transport sweeps between processors so that computational resources are not left idle while waiting for information from neighbouring subdomains. Finally, the number of iterations required by the Krylov subspace solvers could be reduced for some problems by using diffusion synthetic acceleration (see [89] and [90]). DSA preconditioners interleave transport sweeps with the application of an inverse diffusion operator derived from a discretisation of the neutron diffusion equation (see Section 1.5.2). Such methods capture not only the physical process of neutron advection, modelled by the transport sweeps, but also the scattering, modelled by the diffusion operator. Though the transport sweep based preconditioner considered in this thesis was found to be effective and scalable for the problems con-

sidered herein, for scattering dominated problems a DSA type preconditioner could provide significant efficiency gains.

Appendices

APPENDIX A

Problem data

In this appendix we give the nuclear cross sections from the benchmark problems that are included in this thesis.

A.1 Maire and Talay benchmark

Table A.1 gives the cross sections for the benchmark problem published by Maire and Talay in [127].

A.2 Los Alamos benchmark

Table A.2 gives the nuclear cross sections for the seventh benchmark problem from the set compiled by Sood, Forster, and Parsons from the Los Alamos National Laboratory,

Region	Σ_s	$\nu\Sigma_f$	Σ_t
1	0.53	0.079	0.6
2	0.20	0	0.48
3	0.66	0.043	0.70
4	0.50	0	0.65
5	0.89	0	0.90

Table A.1: The material coefficients for the five regions marked in the diagram in Figure 4.2 of the Maire and Talay benchmark problem.

Σ_t	Σ_s	Σ_f	ν
0.32640	0.225216	0.081600	2.84

Table A.2: The material coefficients for Plutonium-239 used in the computation of the LA7 benchmark.

Group (g)	χ	$\Sigma_{s,g\to g}$	$\Sigma_{s,g-1\to g}$	$\Sigma_{s,g-2\to g}$	$\nu\Sigma_f$	Σ_t
Region 1						
1	0.7	0.0871	0	0	0.0524	0.144
2	0.2	0.2486	0.0453	0	0.01	0.2591
3	0.1	0.3883	0.0387	0.0001	0.006	0.4062
Region 2						
1	0	0	0	0	0	0.1
2	0	0	0	0	0	0.3
3	0	0	0	0	0	5.0
Region 3						
1	0	0.176	0	0	0	0.2163
2	0	0.3236	0.0399	0	0	0.3255
3	0	0.9328	0	0	0	1.1228

Table A.3: The material coefficients for the three regions marked in the diagram in Figure 5.4 of the Larsen and Alcouffe benchmark problem.

NM [106].

A.3 Larsen and Alcouffe benchmark

Table A.3 gives the cross sections for the benchmark problem published by Larsen and Alcouffe in the Los Alamos technical report, [134].

g	Σ_t	$\Sigma_{s,g\to1}$	$\Sigma_{s,g\to2}$	$\Sigma_{s,g\to3}$	$\Sigma_{s,g\to4}$	$\nu\Sigma_f$	χ
				Region 1			
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
				Region 2			
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
				Region 3			
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
				Region 4			
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	-	-

Table A.4: The material coefficients for the first four regions marked in the diagram of the KNK benchmark problem from Figure 5.7.

A.4 KNK fast reactor benchmark

Tables A.4 and A.5 give the cross sections for the KNK fast reactor benchmark problem originally published by Takeda and Ikeda in [135]. This is for the unrodded case.

A.5 SILENE benchmark

Table A.6 gives the cross sections for the SILENE benchmark problem. This is from the technical report compiled by Albrecht Kyrieleis at Serco Assurance, [132].

g	Σ_t	$\Sigma_{s,g\to1}$	$\Sigma_{s,g\to2}$	$\Sigma_{s,g\to3}$	$\Sigma_{s,g\to4}$	$\nu\Sigma_f$	χ
			Region 5				
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	-	-
			Region 6				
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	-	-
			Region 7				
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	-	-
			Region 8				
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	-	-

Table A.5: The material coefficients for the final four regions marked in the diagram of the KNK fast reactor benchmark problem from Figure 5.7.

g	Σ_t	$\Sigma_{s,g\to1}$	$\Sigma_{s,g\to2}$	Σ_f	ν	χ
			Region 1, Uranyl Nitrate			
1	2.633073E-01	2.161793E-01	4.400309E-02	1.610373E-03	2.448503E+00	1.0
2	1.954845E+00	4.147318E-04	1.855253E+00	6.961299E-02	2.437871E+00	0.0
			Region 2, Air			
1	1.306052E-04	1.251169E-04	2.182610E-06	-	-	-
2	4.765118E-04	3.166587E-07	4.236728E-04	-	-	-
			Region 3, Steel			
1	3.354474E-01	3.287258E-01	1.357145E-03	-	-	-
2	1.045146E+00	8.654190E-04	8.624749E-01	-	-	-

Table A.6: The material coefficients for the three regions marked in Figure 5.10 for the SILENE benchmark problem.

A.6 Water-cooled reactor benchmark

Table A.7 gives the cross sections for the 2-group water-cooled reactor benchmark problem. This is from the technical report compiled by Albrecht Kyrieleis at Serco Assurance, [132].

g	Σ_t	$\Sigma_{s,g\to1}$	$\Sigma_{s,g\to2}$	Σ_f	ν	χ
			Region 1, Fuel			
1	2.905902E-01	2.563342E-01	2.354509E-02	3.108461E-03	2.551354E+00	1.0
2	1.309916E+00	7.488663E-04	1.193697E+00	7.984104E-02	2.438050E+00	0.0
			Region 2, Steel			
1	3.355475E-01	3.293120E-01	1.186761E-03	-	-	-
2	1.017820E+00	1.190808E-03	8.452682E-01	-	-	-
			Region 3, Water			
1	1.983542E-01	1.678229E-01	2.988528E-02	-	-	-
2	1.294089E+00	5.246971E-04	1.266390E+00	-	-	-

Table A.7: The material coefficients for the three regions marked in Figure 5.13 for the water-cooled reactor benchmark problem.

References

[1] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. Introduction to adaptive methods for differential equations. *Acta numerica*, 4:105–158, 1995.

[2] Cockburn B., Karniadakis G.E., and Shu C.-W. The development of discontinuous Galerkin methods. In *Discontinuous Galerkin Methods*, volume 11 of *Lecture Notes in Computational Science and Engineering*. Springer Berlin Heidelberg, 2000.

[3] Arnold D., Brezzi F., Cockburn B., and Marini L.D. Unified analysis of discontinuous Galerkin methods for Eliptic problems. *SIAM Journal on Numerical Analysis*, 39:1749–1779, 2002.

[4] Scott Congreve. *Two-grid hp-version discontinuous Galerkin finite element methods for quasilinear PDEs*. PhD thesis, University of Nottingham, 2014.

[5] Gerald John Le Beau and Tayfun E Tezduyar. *Finite element computation of compressible flows with the SUPG formulation*. Army High Performance Computing Research Center, 1991.

[6] Reed W.H. and Hill T.R. *Triangular mesh methods for the neutron transport equation*. Los Alamos Scientific Lab., N.Mex. (USA), 1973.

[7] Lesaint P. and Raviart P.-A. *On a finite element method for solving the neutron transport equation*, pages 89–123. Academic Press, Inc, 1974.

[8] Johnson C. and Pitkäranta J. Convergence of a fully discrete scheme for two-dimensional neutron transport. *SIAM journal on numerical analysis*, 20:951–966, 1983.

[9] Peterson T.E. A note on the convergence of the discontinuous Galerkin method for a scalar hyperbolic equation. *SIAM Journal on Numerical Analysis*, 28:133–140, 1991.

[10] Richter G.R. An optimal-order error estimate for the discontinuous Galerkin method. *Mathematics of Computation*, 50:75–88, 1988.

[11] Houston P., Schwab C., and Süli E. Stabilized *hp*-finite element methods for first-order hyperbolic problems. *SIAM Journal on Numerical Analysis*, 37:1618–1643, 2000.

[12] Strouboulis T. and Oden J.T. *A posteriori* error estimation of finite element approximations in fluid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 78:201–242, 1990.

[13] Bey K.S. and Oden J.T. *hp*-version discontinuous Galerkin methods for hyperbolic conservation laws. *Computer Methods in Applied Mechanics and Engineering*, 133:259–286, 1996.

[14] Houston P., Schwab C., and Süli E. Discontinuous *hp*-finite element methods for advection-diffusion-reaction problems. *SIAM Journal on Numerical Analysis*, 39: 2133–2163, 2002.

[15] Cockburn B., Luskin M., Shu C.-W., and Süli E. Enhanced accuracy by post-processing for finite element methods for hyperbolic equations. *Mathematics of Computation*, 72(242):577–606, 2003.

[16] Ryan J. and Shu C.-W. On a one-sided post-processing technique for the discontinuous Galerkin methods. *Methods and Applications of Analysis*, 10:295–308, 2003.

[17] Ryan J., Shu C.-W., and Atkins H. Extension of a post processing technique for the discontinuous Galerkin method for hyperbolic equations with application to an aeroacoustic problem. *SIAM Journal on Scientific Computing*, 26:821–843, 2005.

[18] Guénette R. and Fortin M. A new mixed finite element method for computing viscoelastic flows. *Journal of Non-Newtonian Fluid Mechanics*, 60:27–52, 1995.

[19] Warburton T.C. and Karniadakis G.E. A discontinuous Galerkin method for the viscous MHD equations. *Journal of Computational Physics*, 152:608–641, 1999.

[20] Lomtev I., Quillen C.B., and Karniadakis G.E. Spectral/*hp* methods for viscous compressible flows on unstructured 2D meshes. *Journal of Computational Physics*, 144:325–357, 1998.

[21] Bell G.I. and Glasstone S. *Nuclear reactor theory*. Van Nostrand Reinhold New York, 1970.

[22] Stacey W.M. *Nuclear reactor physics*. John Wiley & Sons, 2007.

[23] Ganapol B.D. *Analytical benchmarks for nuclear engineering applications: case studies in neutron transport theory*. Nuclear Energy Agency, 2008.

[24] M Kharroubi-Mokhtar. *Mathematical topics in neutron transport theory*. World Scientific, 1997.

[25] Carter L.L. and Cashwell E.D. Particle-transport simulation with the Monte Carlo method. Technical report, Los Alamos Scientific Lab., N. Mex.(USA), 1975.

[26] R Sanchez and N. McCormick. Review of neutron transport approximations. *Nuclear Science and Engineering*, 80, 1982.

[27] L. Cao, H. Wu, and Y. Zheng. Solution of neutron transport equation using daubechiesâĂŹ wavelet expansion in the angular discretization. *Nuclear engineering and design*, 238(9):2292–2301, 2008.

[28] Cho N.Z. and Cao L. Wavelet-theoretic method for solution of neutron transport equation. In *Transactions of the Korean Nuclear Society Spring Meeting, Chuncheon, Korea*, 2006.

[29] A. Buchan, C. Pain, M. Eaton, R. Smedley-Stevenson, and A. Goddard. Self-adaptive spherical wavelets for angular discretizations of the boltzmann transport equation. *Nuclear Science and Engineering*, 158(3):244–263, 2008.

[30]

[31] Martin W.R. and Duderstadt J.J. Finite element solutions of the neutron transport equation with applications to strong heterogeneities. *Nuclear Science and Engineering*, 62:371–390, 1977.

[32] Briggs L.L., Miller Jr W.F., and Lewis E.E. Ray-effect mitigation in discrete ordinate-like angular finite element approximations in neutron transport. *Nuclear Science and Engineering*, 57:205–217, 1975.

[33] Becker R., Koch R., Bauer H.-J., and Modest M.F. A finite element treatment of the angular dependency of the even-parity equation of radiative transfer. *Journal of Heat Transfer*, 132:023404, 2009.

[34] Kanschat G. A robust finite element discretization for radiative transfer problems with scattering. *East-West Journal of Numerical Mathematics*, 6:265–272, 1998.

[35] Carlson B.G. Solution of the transport equation by s_n approximations. Technical report, Los Alamos Scientific Lab., N. Mex., 1955.

[36] Chandrasekhar S. *Radiative transfer*. Dover Books on Intermediate and Advanced Mathematics. Dover Publications, 1960.

[37] Lathrop K.D. The early days of the s_n method. In *Invited talk presented at the American Nuclear Society/European Nuclear Society, Chicago, Illinois*, 1992.

[38] Chai J.C., Lee H.S., and Patankar S.V. Ray effect and false scattering in the discrete ordinates method. *Numerical Heat Transfer, Part B Fundamentals*, 24:373–389, 1993.

[39] Li H.-S., Flamant G., and Lu J.-D. Mitigation of ray effects in the discrete ordinates method. *Numerical Heat Transfer: Part B: Fundamentals*, 43:445–466, 2003.

[40] Miller Jr W.F. and Reed W.H. Ray-effect mitigation methods for two-dimensional neutron transport theory. *Nuclear Science and Engineering*, 62:391–411, 1977.

[41] Kanschat G. *Parallel and adaptive Galerkin methods for radiative transfer problems*. PhD thesis, Universität Heidelberg, 1996.

[42] Baker D.J. *Characteristic-based methods for modelling neutron transport*. PhD thesis, University of Nottingham, 2012.

[43] Bennison T.A.J. *Adaptive discontinuous Galerkin methods for the neutron transport equation*. PhD thesis, University of Nottingham, 2013.

[44] Trefethen L.N. and Weideman J.A.C. The exponentially convergent trapezoidal rule. *SIAM Review*, 56:385–458, 2013.

[45] Koch R. and Becker R. Evaluation of quadrature schemes for the discrete ordinates method. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 84:423–435, 2004.

[46] Koch R., Krebs W., Wittig S., and Viskanta R. Discrete ordinates quadrature schemes for multidimensional radiative transfer. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 53:353–372, 1995.

[47] Buchan A.G. *Adaptive spherical wavelets for the angular discretisation of the Boltzmann transport equation*. PhD thesis, Imperial College London (University of London), 2006.

[48] Carlson B.G. *Tables of equal weight quadrature EQn over the unit sphere*. Los Alamos Scientific Laboratory of the University of California, 1971.

[49] Carson M.J. *The spherical harmonic method*. National Research Council of Canada, Atomic Energy Project, Division of Research, 1947.

[50] Scheichl R. *Parallel solution of the transient multigroup neutron diffusion equations with multi-grid and preconditioned Krylov-subspace methods*. Trauner-Verlag, 1997.

[51] Levermore C.D. and Pomraning G.C. A flux-limited diffusion theory. *The Astrophysical Journal*, 248:321–334, 1981.

[52] Pomraning G.C. Flux limiters and Eddington factors. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 27:517–530, 1982.

[53] Briesmeister J.F. *MCNPTM-A general Monte Carlo N-particle transport code*, 2000.

[54] Wang L., Jacques S.L., and Zheng L. MCML–Monte Carlo modeling of light transport in multi-layered tissues. *Computer Methods and Programs in Biomedicine*, 47:131–146, 1995.

[55] P. Wheatley and C. Gerald. *Applied Numerical Analysis*. Addison-Wesley Publishing Company, 1984.

[56] Askew J.R. A characteristics formulation of the neutron transport equation in complicated geometries. Technical report, United Kingdom Atomic Energy Authority, Reactor Group, Winfrith (United Kingdom), 1972.

[57] Alcouffe R.E. and Larsen E.W. Review of characteristic methods used to solve the linear transport equation. Technical report, Los Alamos Scientific Lab., NM (USA), 1981.

[58] Powney D.J. and Newton T.D. Overview of the wims 9 resonance treatment. Technical report, ANSWERS/WIMS/TR. 26, 2004.

[59] Takeuchi K. A numerical method for solving the neutron transport equation in finite cylindrical geometry. *Journal of Nuclear Science and Technology*, 6:466–473, 1969.

[60] Ohnishi T. Finite element method applied to reactor physics problems. *Journal of Nuclear Science and Technology*, 8:717–720, 1971.

[61] Johnson C., Nävert U., and Pitkäranta J. Finite element methods for linear hyperbolic problems. *Computer Methods in Applied Mechanics and Engineering*, 45: 285–312, 1984.

[62] Pain C.C., Eaton M.D., Smedley-Stevenson R.P., Goddard A.J.H., Piggott M.D., and de Oliveira C.R.E. Streamline upwind Petrov–Galerkin methods for the steady-state Boltzmann transport equation. *Computer Methods in Applied Mechanics and Engineering*, 195:4448–4472, 2006.

[63] Pain C.C., Eaton M.D., Smedley-Stevenson R.P., Goddard A.J.H., Piggott M.D., and de Oliveira C.R.E. Space–time streamline upwind Petrov–Galerkin methods for the Boltzmann transport equation. *Computer Methods in Applied Mechanics and Engineering*, 195:4334–4357, 2006.

[64] de Oliveira C.R.E. An arbitrary geometry finite element method for multigroup neutron transport with anisotropic scattering. *Progress in Nuclear Energy*, 18:227–236, 1986.

[65] Merton S.R., Smedley-Stevenson R.P., Pain C.C., and Buchan A.G. Adjoint eigenvalue correction for elliptic and hyperbolic neutron transport problems. *Progress in Nuclear Energy*, 76:1–16, 2014.

[66] Grohs P., Keiper S., Kutyniok G., and Schäfer M. α-molecules. *arXiv preprint arXiv:1407.4424*, 2014.

[67] Grohs P. and Obermeier A. Optimal adaptive ridgelet schemes for linear transport equations. *arXiv preprint arXiv:1409.1881*, 2014.

[68] Grohs P. and Obermeier A. Ridgelet methods for linear transport equations. Technical report, ETH Zürich, 2014.

[69] Etter S., Grohs P., and Obermeier A. FFRT: A fast finite ridgelet transform for radiative transport. *Multiscale Modeling & Simulation*, 13:1–42, 2015.

[70] HSL. HSL 2013: A collection of Fortran codes for large scale scientific computation, 2013. `http://www.hsl.rl.ac.uk`.

[71] D. Arnold, F. Brezzi, B. Cockburn, and D. Marini. Discontinuous galerkin methods for elliptic problems. In *Discontinuous Galerkin Methods*, pages 89–101. Springer, 2000.

[72] Duff I.S., Erisman A.N., and Reid J.K. *Direct Methods for Sparse Matrices*. Oxford University Press, Oxford, England, 1986.

[73] Davis T.A. *Direct methods for sparse linear systems*, volume 2 of *Fundamentals of Algorithms*. SIAM, 2006.

[74] Yannakakis M. Computing the minimum fill-in is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 2:77–79, 1981.

[75] Duff I.S. MA57 – A code for the solution of sparse symmetric indefinite systems. *ACM Transactions on Mathematical Software (TOMS)*, 30:118–144, 2004.

[76] Amestoy P.R., Davis T.A., and Duff I.S. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17:886–905, 1996.

[77] George A. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10:345–363, 1973.

[78] Lipton R.J., Rose D.J., and Tarjan R.E. Generalized nested dissection. *SIAM Journal on Numerical Analysis*, 16:346–358, 1979.

[79] Duff I.S., Erisman A.L., and Reid J.K. On George's nested dissection method. *SIAM Journal on Numerical Analysis*, 13:686–695, 1976.

[80] Karypis G. and Kumar V. METIS version 4.0, 1998. University of Minnesota, Department of Computer Science, Minneapolis, MN.

[81] Karypis G. and Kumar V. METIS-a software package for partitioning unstructured graphs, meshes, and computing fill-reducing orderings of sparse matrices-version 5.0, 2011. University of Minnesota.

[82] Hartmann R. and Houston P. Adaptive discontinuous Galerkin finite element methods for nonlinear hyperbolic conservation laws. *SIAM Journal on Scientific Computing*, 24:979–1004, 2003.

[83] Dubiner M. Spectral methods on triangles and other domains. *Journal of Scientific Computing*, 6:345–390, 1991.

[84] Murphy S. First year report. Technical report, University of Nottingham, 2011. Submitted to the Department of Mathematical Sciences in accordance with the requirements of the PhD program.

[85] Duff I.S. and Reid J.K. An implementation of Tarjan's algorithm for the block triangularization of a matrix. *ACM Transactions on Mathematical Software (TOMS)*, 4:137–147, 1978.

[86] HSL. Collection of Fortran codes for large-scale scientific computation, 2007. See http://www.hsl.rl.ac.uk.

[87] Plimpton S., Hendrickson B., Burns S., and McLendon III W. Parallel algorithms for radiation transport on unstructured grids. In *Supercomputing, ACM/IEEE 2000 Conference*, page 25. IEEE, 2000.

[88] Patton B.W. and Holloway J.P. Application of preconditioned GMRES to the numerical solution of the neutron transport equation. *Annals of Nuclear Energy*, 29:109–136, 2002.

[89] Larsen E. Diffusion-synthetic acceleration methods for discrete-ordinates problems. *Transport Theory and Statistical Physics*, 13:107–126, 1984.

[90] Alcouffe R.E. Diffusion synthetic acceleration methods for the diamond differenced discrete-ordinates equations. *Nuclear Science and Engineering*, 64:344–355, 1977.

[91] Patton B.W. and Holloway J.P. Some remarks on GMRES for transport theory. In *Proceedings of M&C 2003–Nuclear Mathematical and Computational Sciences: A Century in Review-A Century Anew, Gatlinburg, USA, April 6–1*, 2003.

[92] Gupta A. and Modak R.S. On the use of the conjugate gradient method for the solution of the neutron transport equation. *Annals of Nuclear Energy*, 29:1933–1951, 2002.

[93] Sanchez R. and Santandrea S. Symmetrization of the transport operator and Lanzcos iterations. In *Proceedings of the 2001 International Meeting on Mathematical Methods for Nuclear Applications*, pages 9–13, 2001.

[94] Modak R.S. and Gupta A. New applications of orthomin(1) algorithm for *k*-eigenvalue problem in reactor physics. *Annals of Nuclear Energy*, 33:538–543, 2006.

[95] Warsa J.S., Wareing T.A., and Morel J.E. Krylov iterative methods applied to multidimensional S_N calculations in the presence of material discontinuities. In *Proceedings of M and C 2003–Nuclear Mathematical and Computational Sciences: A Century in Review-A Century Anew, Gatlinburg, USA, April 6–1*, 2003.

[96] Oliveira S. and Deng Y. Preconditioned Krylov subspace methods for transport equations. *Progress in Nuclear Energy*, 33:155–174, 1998.

[97] Saad Y. and Schultz M.H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

[98] Frayssé V., Giraud L., Gratton S., and Langou J. A set of GMRES routines for real and complex arithmetics. Technical report, CERFACS, Toulouse Cedex, France, 1997.

[99] Warsa J.S., Wareing T.A., Morel J.E., McGhee J.M., and Lehoucq R.B. Krylov subspace iterations for deterministic k-eigenvalue calculations. *Nuclear Science and Engineering*, 147:26–42, 2004.

[100] Adams M.L. and Larsen E.W. Fast iterative methods for discrete-ordinates particle transport calculations. *Progress in nuclear energy*, 40:3–159, 2002.

[101] Manteuffel T.A. The Tchebychev iteration for nonsymmetric linear systems. *Numerische Mathematik*, 28:307–327, 1977.

[102] Allen E.J. and Berry R.M. The inverse power method for calculation of multiplication factors. *Annals of Nuclear Energy*, 29:929–935, 2002.

[103] Varga R.S. *Matrix iterative analysis*, volume 27. Springer-Verlag Berlin Heidelberg, 2009.

[104] Lehoucq R.B., Sorensen D.C., and Yang C. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, volume 6. SIAM, 1998.

[105] Azmy Y.Y. Multiprocessing for neutron diffusion and deterministic transport methods. *Progress in Nuclear Energy*, 31:317–368, 1997.

[106] Sood A., Forster R.A, and Parsons D.K. Analytical benchmark test for criticality code verification. *Progress in Nuclear Energy*, 42:55–106, 2003.

[107] Giani S., Hall E., and Houston P. Aptofem users manual. Technical report, Technical report, University of Nottingham, 2009.

[108] Goto K. Gotoblas. *Texas Advanced Computing Center, University of Texas at Austin, USA. URL< http://www. otc. utexas. edu/ATdisplay. jsp*, page 21, 2007.

[109] Shewchuk J.R. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied computational geometry: towards geometric engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, 1996.

[110] Johnson C. and Pitkäranta J. An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation. *Mathematics of Computation*, 46:1–26, 1986.

[111] Cockburn B., Dong B., and Guzmán J. Optimal convergence of the original DG method for the transport-reaction equation on special meshes. *SIAM Journal on Numerical Analysis*, 46:1250–1265, 2008.

[112] Asadzadeh M. Analysis of a fully discrete scheme for neutron transport in two-dimensional geometry. *SIAM journal on numerical analysis*, 23:543–561, 1986.

[113] Verfürth R. *A review of a posteriori error estimation and adaptive mesh-refinement techniques*. Advances in Numerical Mathematics. Wiley-Teubner, 1996.

[114] Ainsworth M. and Oden J.T. *A posteriori* error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 142:1–88, 1997.

[115] Szabo B.A. and Babuška I. *Finite element analysis*. A Wiley-Interscience publication. John Wiley & Sons, 1991.

[116] Becker R. and Rannacher R. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica 2001*, 10:1–102, 2001.

[117] Bangerth W. and Rannacher R. *Adaptive finite element methods for differential equations*. Birkhäuser Verlag, 2003.

[118] Houston P. and Süli E. *hp*-adaptive discontinuous Galerkin finite element methods for first-order hyperbolic problems. *SIAM Journal on Scientific Computing*, 23: 1226–1252, 2001.

[119] Cliffe K.A., Hall E.J.C., and Houston P. Adaptive discontinuous Galerkin methods for eigenvalue problems arising in incompressible fluid flows. *SIAM Journal on Scientific Computing*, 31:4607–4632, 2010.

[120] Cliffe K.A., Hall E.J.C., Houston P., Phipps E.T., and Salinger A.G. Adaptivity and a posteriori error control for bifurcation problems I:Bratu problem. *Communications in Computational Physics*, 8:845–865, 2010.

[121] Cliffe K.A., Hall E.J.C., Houston P., Phipps E.T., and Salinger A.G. Adaptivity and a posteriori error control for bifurcation problems II: Incompressible fluid flow in open systems with z_2 symmetry. *Journal of Scientific Computing*, 47:389–418, 2011.

[122] Cliffe K.A., Hall E.J.C., Houston P., Phipps E.T., and Salinger A.G. Adaptivity and *a posteriori* error control for bifurcation problems III: Incompressible fluid flow in open systems with $o(2)$ symmetry. *Journal of Scientific Computing*, 52:153–179, 2012.

[123] Duo J.I., Azmy Y.Y., and Zikatanov L.T. A posteriori error estimator and AMR for discrete ordinates nodal transport methods. *Annals of Nuclear Energy*, 36:268–273, 2009.

[124] Fournier D., Le Tellier R., and Suteau C. Analysis of an *a posteriori* error estimator for the transport equation with S_N and discontinuous Galerkin discretizations. *Annals of Nuclear Energy*, 38:221–231, 2011.

[125] Lathouwers D. Goal-oriented spatial adaptivity for the S_N equations on unstructured triangular meshes. *Annals of Nuclear Energy*, 38:1373–1381, 2011.

[126] Lathouwers D. Spatially adaptive eigenvalue estimation for the S_N equations on unstructured triangular meshes. *Annals of Nuclear Energy*, 38:1867–1876, 2011.

[127] Maire S. and Talay D. On a Monte Carlo method for neutron transport criticality computations. *IMA Journal of Numerical Analysis*, 26:657–685, 2006.

[128] Kavenoky A., Stepanek J., and Schmidt F. Benchmark problems. Transport theory and advanced reactor simulations. Technical report, International Atomic Energy Agency, 1979.

[129] Mitchell W.F. and McClain M.A. A comparison of *hp*-adaptive strategies for elliptic partial differential equations. *ACM Transactions on Mathematical Software (TOMS)*, 41:2:1–2:39, 2014.

[130] Houston P. and Süli E. A note on the design of *hp*-adaptive finite element methods for elliptic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194:229–243, 2005.

[131] Eibner T. and Melenk J.M. An adaptive strategy for *hp*-FEM based on testing for analyticity. *Computational Mechanics*, 39:575–595, 2007.

[132] Kyrieleis A. Monk benchmarks using 2D reactor models. Technical report, Technical report, Serco Assurance, 2011.

[133] Westfall R.M. Benchmark solutions for the infinite critical cylinder. Technical report, Oak Ridge National Laboratory, 1983.

[134] Larsen E.W. and Alcouffe R.E. The linear characteristic method for spatially discretising the discrete-ordinates equations in (x,y)-geometry. Technical report, Los Alamos Scientific Laboratory, 1981.

[135] Takeda T. and Ikeda H. 3-D neutron transport benchmarks. *Journal of Nuclear Science and Technology*, 28:656–669, 1991.

[136] Kim T.H. and Cho N.Z. Source projection analytic nodal S_N method for hexagonal geometry. *Annals of Nuclear Energy*, 23:133–143, 1996.

[137] Wang Y. *Adaptive mesh refinement solution techniques for the multigroup S_N transport equation using a higher-order discontinuous finite element method*. PhD thesis, Texas A&M University, 2009.

[138] Cangiani A., Georgoulis E.H., and Houston P. *hp*-Version discontinuous Galerkin methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, 24:2009–2041, 2014.