HP-MESH ADAPTATION FOR 1-D MULTIGROUP NEUTRON DIFFUSION PROBLEMS

A Thesis

by

YAQI WANG

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

MASTER OF SCIENCE

December 2006

Major Subject: Nuclear Engineering

HP-MESH ADAPTATION FOR 1-D MULTIGROUP NEUTRON DIFFUSION PROBLEMS

A Thesis

by

YAQI WANG

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

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Jean C. Ragusa
Committee Members,	Marvin L. Adams
	Jean-Luc Guermond
Head of Department,	William E. Burchill

December 2006

Major Subject: Nuclear Engineering

ABSTRACT

hp-Mesh Adaptation for 1-D Multigroup Neutron

Diffusion Problems. (December 2006)

Yaqi Wang, B.S., Tsinghua University

Chair of Advisory Committee: Dr. Jean C. Ragusa

In this work, we propose, implement and test two fully automated mesh adaptation methods for 1-D multigroup eigenproblems. The first method is the standard *hp*-adaptive refinement strategy and the second technique is a goal-oriented *hp*-adaptive refinement strategy. The *hp*-strategies deliver optimal guaranteed solutions obtained with exponential convergence rates with respect to the number of unknowns. The goal-oriented method combines the standard *hp*-adaptation technique with a goal-oriented adaptivity based on the simultaneous solution of an adjoint problem in order to compute quantities of interest, such as reaction rates in a sub-domain or point-wise fluxes or currents. These algorithms are tested for various multigroup 1-D diffusion problems and the numerical results confirm the optimal, exponential convergence rates predicted theoretically.

To my mother

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Ragusa for his guidance and support throughout the course of this research. Thanks to my committee members, Dr. Adams and Dr. Guermond for their advices.

Thanks also to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my mother and father for their encouragement and to my wife for her patience and love.

TABLE OF CONTENTS

Page

ABSTRACT iii

DEDICATION iv

ACKNOWLEDGMENTS v

TABLE OF CONTENTS vi

LIST OF FIGURES ix

CHAPTER

I INTRODUCTION 1

II BRIEF INTRODUCTION TO FINITE ELEMENT METHOD 8

2.1 Preliminaries 8
2.2 Neutron balance equations 8
2.3 Formulations of elliptic partial differential equations 11
2.3.1 Classification of partial differential equations 11
2.3.2 Classical formulation of multidimensional elliptic boundary value problems 11
2.3.3 Variational formulation 14
2.3.4 Equivalence with a minimization problem 15
2.3.5 Sobolev spaces and well-posed VBVP 16
2.3.6 Continuity and coercivity of a bilinear form b(ϕ, φ) 17
2.4 Galerkin formulation 18
2.4.1 Galerkin method 18
2.4.2 Cea's lemma and *a priori* error estimations 19
2.5 Finite element method 20
2.5.1 Triangulation 21
2.5.2 Master element and shape functions 21
2.5.3 Real elements and mapping 23
2.5.4 Integral quadrature; element mass matrix; stiffness matrix; and load vector 25
2.5.5 Basis functions, connectivity and global assembly 27

CHAPTER Page

2.5.6 Boundary condition manipulation 29
2.6 Projection-based interpolation in 1-D 30
2.6.1 Definition 31
2.6.2 Some specific cases of projection 32
2.6.3 Projection of a solution from a mesh to another 39
2.7 A 1-D FEM neutron diffusion code in MATLAB 40
2.7.1 Features and limitations of the code 40
2.7.2 Some demonstration results 41
2.8 Conclusions of chapter II 53

III ADAPTIVE HP-REFINEMENT TECHNIQUES 54

3.1 Introduction: Motivations for mesh refinement and *hp*-adaptation 54
3.2 Principle of mesh adaptation 57
3.3 *A posteriori* error estimation 58
3.3.1 Notations 59
3.3.2 Computing a reference solution 60
3.3.3 Additional justifications for the error estimator 61
3.3.4 Interpolation based error estimator 64
3.3.5 New error estimator 64
3.4 Competitive *hp*-refinements 65
3.4.1 Competitive refinement choices 65
3.4.2 *hp*-refinement strategy 67
3.4.3 *h*-constraint and some numerical limitations 68
3.4.4 Expected convergence rates 71
3.5 The 1D-*hp* code (Demkowicz, UT-Austin) 73
3.5.1 Data structure for *hp*-refinement and some associated algorithms 73
3.5.2 Adaptive integration during assembly 77
3.5.3 Several enhancements to the original 1-D *hp*-FEM code 79
3.6 One-group fixed source diffusion problems 81
3.6.1 Example 1.A (A 1-D one-group neutron diffusion source problem) 81
3.6.2 Example 1.B (A 1-D one-group neutron diffusion source problem) 98
3.7 Conclusions of chapter III 100

IV MULTIGROUP ADAPTIVE HP-REFINEMENT STRATEGIES 102

4.1 Multigroup diffusion equations and iteration solver 103
4.1.1 Multigroup diffusion equations 103
4.1.2 Variational form of multi-group equations 107

CHAPTER Page

4.1.3 Description of the iteration solver for the multigroup diffusion equations 110
4.2 Multigroup energy-driven *hp*-adaptation 117
4.2.1 Basic considerations 117
4.2.2 Mesh iteration implemented around the one-group solver 118
4.2.3 Mesh iteration wrapped outside the power iteration loop 120
4.2.4 Construction of group source terms when different meshes are used . 124
4.3 Multigroup goal-oriented *hp*-adaptation 124
4.3.1 Detector response as a quantity of interest 125
4.3.2 Point value as quantities of interest 127
4.4 Data structure and algorithms 130
4.4.1 Modification of main data structure 130
4.4.2 BACKUP module and REFERENCE module 131
4.4.3 Goal-oriented calculation 131
4.5 Examples and discussions 132
4.5.1 Description of figures presented 132
4.5.2 Options of *hp*-adaptation 134
4.5.3 Two-group energy-driven source problem without fission or up-scattering 134
4.5.4 Two-group energy-driven source problem with fission 154
4.5.5 Two-group goal-oriented source problem 158
4.5.6 A two-group eigenvalue problem 168
4.5.7 Seven-group problems 180
4.5.8 Point-wise values as quantity of interest 202
4.6 Conclusions of chapter IV 211

V SUMMARY 213

REFERENCES 215

VITA 219

LIST OF FIGURES

Page

Fig. I-1. Schematics of mesh adaptation 5

Fig. II-1. Master elements with different definition domain 21

Fig. II-2. An example of FEM domain 28

Fig. II-3. Projection-based interpolation in the case of clustering 36

Fig. II-4. Projection-based interpolation in the case of splitting 39

Fig. II-5. An example of mesh projection 40

Fig. II-6. Condition number of local partial stiffness matrix and local mass matrix 43

Fig. II-7. Condition number of global stiffness matrix: (a) Lagrange polynomials, (b) modified Peano polynomials, (c) Peano polynomials and (d) Lobatto polynomials 44

Fig. II-8. Flux distribution with p=1: (a) 8 elements, (b) 16 elements, (c) 32 elements and (d) 64 elements 46

Fig. II-9. Flux distribution with p=2: (a) 16 elements, (b) 16 elements, (c) 32 elements and (d) 512 elements 48

Fig. II-10. Flux distributions for a sample eigenvalue problem: (a) 8 elements, (b) 16 elements, (c) 32 elements and (d) 64 elements 51

Fig. III-1. Schematics of mesh adaptation 58

Fig. III-2. *hp*-refinement strategy 69

Fig. III-3. Flow chart of one *hp*-adaptive mesh iteration step 70

Fig. III-4. Tree structure of *h*-refinement 77

Fig. III-5. Flowchart of adaptive integral algorithm 78

Fig. III-6. Solutions of example 1-A with tolerance=10%: (a) flux and (b) mesh structure 82

Fig. III-7. Solutions of example 1-A with tolerance=1%: (a) flux and (b) mesh structure 83

Page

Fig. III-8. Solutions of example 1-A with tolerance=10^{-5}%: (a) flux and (b) mesh structure 84

Fig. III-9. Convergence properties of *hp*-adaptive: (a) log-linear and (b) log-log 86

Fig. III-10. Computational cost 87

Fig. III-11. Convergence properties of: (a) *h*-adaptation and (b) uniform *h*-refinement 89

Fig. III-12. Convergence of uniform *p*-refinement 90

Fig. III-13. Convergence of different schemes 91

Fig. III-14. Convergence results of Lobatto and Peano shape functions 92

Fig. III-15. Convergence results with different maximum polynomial order 92

Fig. III-16. Influence on convergence sequence of *h*-constraint and initial mesh 94

Fig. III-17. Influence on resulting mesh of *h*-constraint and initial mesh distributions: (a) 7 initial elements and (b) 14 initial elements 95

Fig. III-18. Convergence paths of using density of interpolation error as refinement criteria 96

Fig. III-19. Results of using density of interpolation error as refinement criteria: (a) mesh structure, (b) local error distribution 97

Fig. III-20. Solution of example 1-B: (a) flux and (b) mesh structure 98

Fig. III-21. Convergence properties of *hp*-adaptation, example 1-B: (a) log-log and (b) log-linear 99

Fig. III-22. Convergence of uniform *p*-refinement 100

Fig. IV-1. General multigroup iteration solver 114

Fig. IV-2. Multigroup iteration solver with one thermal iteration 115

Fig. IV-3. Diagram of direct and adjoint source update 116

Fig. IV-4. Group sweep with mesh iteration wrapped around the one-group solver 118

Fig. IV-5. Schematics of mesh iteration wrapped outside power iteration 122

Fig. IV-6. Convergence sequence of example 2.A 136

Page

Fig. IV-7. Influence on thermal group of fast convergence 137

Fig. IV-8. Solutions of example 2.A: (a) flux and flux derivative with *tol*=10%, (b) flux and flux derivative with *tol*=1% 138

Fig. IV-9. Error distribution of fluxes with different tolerances, example 2.A 139

Fig. IV-10. Mesh structure when E_1 = 2.3×10^{-6}%; E_2 = 7.7×10^{-6}% of example 2.A 140

Fig. IV-11. Convergence sequence for example 2.B 141

Fig. IV-12. Flux distributions of example 2.B 142

Fig. IV-13. Mesh structure when E_1 = 5.3×10^{-6}%; E_2=4.9×10^{-6}% of example 2.B 143

Fig. IV-14. Error distribution for the fluxes of example 2.B 143

Fig. IV-15. Convergence sequence of example 2.C with α_2=0.1 145

Fig. IV-16. Convergence sequence for example 2.C with α_2=0.9 146

Fig. IV-17. Convergence sequence of example 2.C with mesh iteration outside compared with mesh-iteration-inside scheme 146

Fig. IV-18. Convergence sequence of example 2.C with mesh iteration outside 147

Fig. IV-19. Convergence sequence of example 2.C with different group weighting factors 148

Fig. IV-20. Convergence sequence of example 2.C with single mesh 149

Fig. IV-21. Flux distributions of example 2.C 149

Fig. IV-22. Mesh structure of example 2.C 150

Fig. IV-23. Convergence of uniform *p*-refinement of example 2.C 151

Fig. IV-24. Convergence of uniform *h*-refinement of example 2.C 152

Fig. IV-25. Convergence of adaptive *h*-refinement 152

Fig. IV-26. Convergence of different refinement strategies of example 2.C 153

Fig. IV-27. Convergence sequences of example 3 155

Fig. IV-28. Convergence results with different group weighting factors of example 3 156

Fig. IV-29. Solutions with E_1=7.2×10^{-5}% and E_2=8.0×10^{-5}% of example 3: (a) flux and (b) mesh structure 157

Page

Fig. IV-30. Error distribution of the thermal flux error, example 3 158

Fig. IV-31. Convergence sequence of goal-oriented calculation... 159

Fig. IV-32. Error distribution of thermal flux in a goal-oriented calculation 160

Fig. IV-33. Error distribution of the thermal flux in an energy-driven calculation 161

Fig. IV-34. Relationship between thermal flux accuracy and accuracy of quantity of interest 162

Fig. IV-35. Convergence sequence of goal-oriented calculation... 163

Fig. IV-36. Flux distribution from goal-oriented calculation 164

Fig. IV-37. Mesh structure from goal-oriented calculation 165

Fig. IV-38. Relationship between flux accuracy and accuracy of quantity of interest 165

Fig. IV-39. Convergence sequence of goal-oriented calculation... 166

Fig. IV-40. Result of goal-oriented calculation 167

Fig. IV-41. Flux distributions of example 5 170

Fig. IV-42. Convergence sequence of example 5 171

Fig. IV-43. Mesh structure of example 5... 172

Fig. IV-44. Comparison of *hp*-adaptive and uniform *h*-refinement of an eigenvalue problem.. 173

Fig. IV-45. Thermal flux error distributions, example 5 174

Fig. IV-46. Convergence sequence of goal-oriented calculation of example 5 175

Fig. IV-47. Comparison of goal-oriented and energy-driven calculations 176

Fig. IV-48. Mesh structure from goal-oriented calculation 176

Fig. IV-49. Two ways to calculate k-effective... 177

Fig. IV-50. “Exact” direct and adjoint flux distribution of example 6.A 181

Fig. IV-51 Flux distribution with different tolerance of example 6.A: (a) *tol*=10%, (b) *tol*=1%, (c) tol=0.1% and (d) *tol*=0.01% 182

Fig. IV-52. Number-based convergence path of example 6.A: (a) compared with error-based calculation, (b) compared with reference convergence path 185

Page

Fig. IV-53. Convergence paths with different *h*-refinement constrain of example 6.A 187

Fig. IV-54. Convergence paths of goal-oriented calculations of example 6.A.......... 188

Fig. IV-55. Mesh structure of example 6.A: (a) group #1, (b) group #2, (c) group #3, (d) group #4, (e) group #5, (f) group #6, (g) group #7 189

Fig. IV-56. Convergence paths of 7-group eigenvalue problem 194

Fig. IV-57. Flux distribution of example 6.B 195

Fig. IV-58. Adjoint flux distribution of example 6.B.......... 196

Fig. IV-59. Direct mesh structure of example 6.B 197

Fig. IV-60. Adjoint mesh structure of example 6.B 198

Fig. IV-61. Convergence paths of goal-oriented calculation of example 6.B 199

Fig. IV-62. Convergence paths of energy-driven and goal-oriented calculations of example 199

Fig. IV-63. Mesh structure of example 6.B 200

Fig. IV-64. Energy-driven calculation of adjoint eigenvalue problem of example 6.B 201

Fig. IV-65. Adjoint flux of example 7: (a) thermal group, (b) fast group 205

Fig. IV-66. Convergence paths of energy-driven calculation of example 7 206

Fig. IV-67. Mesh structure of example 7.......... 207

Fig. IV-68. Convergence paths of goal-oriented calculation of example 7 208

Fig. IV-69. Mesh structure of goal-oriented calculation of example 7.......... 209

Fig. IV-70. Convergence path of goal-oriented calculation of point flux as quantity of interest 209

Fig. IV-71. Adjoint flux of point flux as quantity of interest 210

Fig. IV-72. Mesh structure of point flux as quantity of interest 211

LIST OF TABLES

Page

TABLE III-I Main results of example 1.A… 86

TABLE III-II Influence of α_1 with different number of initial elements, (α_2=0.1)… 94

TABLE III-III Influence of α_2 with different number of initial elements, (α_1=1/3) 95

TABLE IV-I Influence of fast group convergence criterion on the thermal group convergence 138

TABLE IV-II Mesh structure of example 4.A 162

TABLE IV-III 7-group cross sections 179

TABLE IV-IV “Exact” solution of example 6.A 185

TABLE IV-V Results of example 6.A with *tol*=10% and 1% 185

TABLE IV-VI Results of example 6.A with *tol*=0.1% and 0.01% 185

TABLE IV-VII “Exact” solution of example 6.B 196

TABLE IV-VIII Mesh iteration with semi-H^1 norm of point current as quantity of interest 203

TABLE IV-IX Mesh iteration with L_2 norm of point current as quantity of interest 204

CHAPTER I

INTRODUCTION

The design, analysis and control of nuclear reactors require solving numerically the neutron transport equation (or an approximation of it) in order to determine the neutron distribution in the reactor, and hence validate and verify design and safety parameters. Unfortunately, obtaining a sufficiently accurate numerical solution can require a tremendous amount of floating point operations, taxing the computer's memory and speed. Even with nowadays computers, the explicit modeling of each fuel pin in, say, a light water reactor (LWR) is still prohibitive. For instance, we can roughly estimate how many unknowns would be required for the neutronics computation of a single state-point in a pressurized water reactor (PWR):

193	(fuel assemblies) ×
24	(axial planes in reactor model) ×
172	(pin cells in assembly) ×
32	(regions assuming for the pin cell spatial discretization) ×
256	(directions of neutron travel) ×
70	(energy groups)

≈

768 billions unknowns !

These numbers were adapted from a talk given by Kord Smith, the main author of the 2D lattice and 3D core neutronic codes CASMO and SIMULATE. Such larger numbers of unknowns were intractable several decades ago and still constitute a formidable challenge nowadays. In the earlier days of nuclear engineering, the process of solving for the neutron distribution in a nuclear reactor core was split into four sub-tasks (in a divide and conquer fashion) [1]: first, a small 2D geometrical motif of the core was solved with high fidelity,

This thesis follows the style of *Nuclear Science and Engineering.*

requiring the solution of the neutron transport equation; then the results of the previous stage were used to homogenize the geometry and material compositions; subsequently, the homogenized data for solved on 3D coarse meshes using the diffusion approximation of transport; finally, the coarse 3D results along with the 2D fine results are used to reconstruct the fine 3D results. This methodology is still prevalent nowadays but possesses inherent drawbacks found in its homogenization and reconstruction processes. It is commonly acknowledged [2] that overcoming these drawbacks will require solving the multigroup transport equation on the whole heterogeneous 3D geometry, with a large number of energy groups and angular directions or moments. A 3D solution will certainly not be feasible without improved algorithms such as *automatic mesh adaptation*, where the mesh cells are automatically and selectively refined in order to reduce the largest contributions to the total error. Even though the ultimate goal is the efficient resolution of the transport equation in 3D, there are some necessary issues which need to be resolved but can be understood and analyzed on a reduced scale. In this thesis, we investigate the behavior of mesh refinement techniques in the case of eigenproblems consisting systems of coupled equations (the multigroup equations). This analysis is carried out on a reduced framework, the 1-D multigroup diffusion setting, but the lessons drawn here will be useful for multigroup transport eigenproblems.

Historically, solutions to reactor core analysis problems were first obtained using traditional *finite difference methods* (FDM) [3-5] in the 1960's. FDM requires a large number of mesh points in order to represent accurately the spatial variation of the neutron flux. It is well-known that the finite difference mesh spacing must be on the order of the smallest group-wise diffusion length for correct results. The computational cost associated with FDM motivated the development of *modem transverse-integrated nodal methods* [6] [7]. These nodal methods reached maturity in the mid 1980's and are widely used for reactor physics design and on-site monitoring. Nowadays, 3D calculations for light water reactors with homogenized fuel assemblies can now be performed even in on-site fashion. However, the accuracy and the theoretical justification for the homogenization and reconstruction processes, which enabled the use of FDM and nodal methods, now hinders any further improvements to 3D solutions using these techniques. It is even believed that the success of modern nodal methods prohibited the further development of other spatial discretization techniques in reactor core analysis. However, with the emergence of new types of reactors [8] with more intricate geometries or more severe flux transients, the motivation to pursue more accurate numerical simulations is calling for finer geometrical details, increased number of energy groups and more angles or moments in the transport equation.

The *Finite element method* (FEM) [9], which had been introduced in nuclear engineering since as early as the mid 70's [10] [11] and gradually obtained more attentions [12] [13], is of special interest to us because it provides an efficient way to refine the mesh *non-uniformly* while delivering accurate solutions. The FEM is a computational technique for obtaining approximate solutions to the partial differential equations that arise in scientific and engineering applications. Rather than approximating the partial differential equations directly as for instance with finite difference, the finite element method utilizes a variational problem that involves an integral of the differential equation over the problem domain. This domain can contain complex geometries (and boundaries). FEM can easily handle such domains whereas FDM is restricted to handle only regular shapes and simple alterations of them. In FEM, the computational domain is divided into a number of sub-domains called finite elements and the solution of the partial differential equation is approximated by a polynomial function on each element. The division of domain can be arbitrary. The polynomial orders within each element can be of any value. Hence, FEM provides two options for refining a mesh non-uniformly. However, optimally distributing these approximation parameters - the sizes *h* of the elements and the orders *p* of the polynomial shape functions- represents a significant departure from the conventional finite element techniques. Such *hp*-refinements emerged in late 1980's and required the resolution of several formidable problems for an effective implementation [14-16]: new data structures, efficient linear solvers, effective local (*a posteriori*) error estimations. Note that in the recent years, *h*-refinement and *p*-refinement have been investigated for neutronics calculations [17] [18]. Nonetheless, neither the *h*-method nor the *p*-method yields optimal convergence rates.

Up to now, obtaining a solution to a desired tolerance is seldom addressed in nuclear science and engineering methodologies and tools. Simply attempting to converge a solution using uniform mesh refinement is impracticable because (1) this process soon comes to a halt due to the enormous increase in the number of unknowns, and, (2) it does not guarantee that a solution has been reached within a prescribed tolerance. Dealing with the approximation error $e_h = u - u_h$, i.e., the difference between the exact solution u and the numerical solution u_h, is a very arduous task because bounds of the approximation error are complex to obtain, with the added difficulty that they are problem-dependent. In the last decade, the theory of *a posteriori error estimations* [19] [20] has matured and allows the measure, control, and minimization of approximation errors. In this theory, the computed solution itself is used to inexpensively provide point-wise error estimations. These error estimators are called *a posteriori* because they are determined afterwards, once a solution has been obtained. By effectively estimating the error, the possibility of controlling the entire computational process emerges as the succession, within a single calculation, of adaptively refined meshes. Fig. I-1 depicts the error estimation and mesh refinement procedure. Once the solution has been computed on the i^{th} mesh, the error is estimated using the current solution u^i_h. Process termination is determined as follows:

- If the current solution has converged within the user's defined tolerance, the process is stopped.
- If the solution has not converged sufficiently, the error estimator is used to build a new mesh $i+1$ on which a new solution will be sought.

The entire process is achieved within a single calculation, comprising a set on successive meshes and successive solutions.

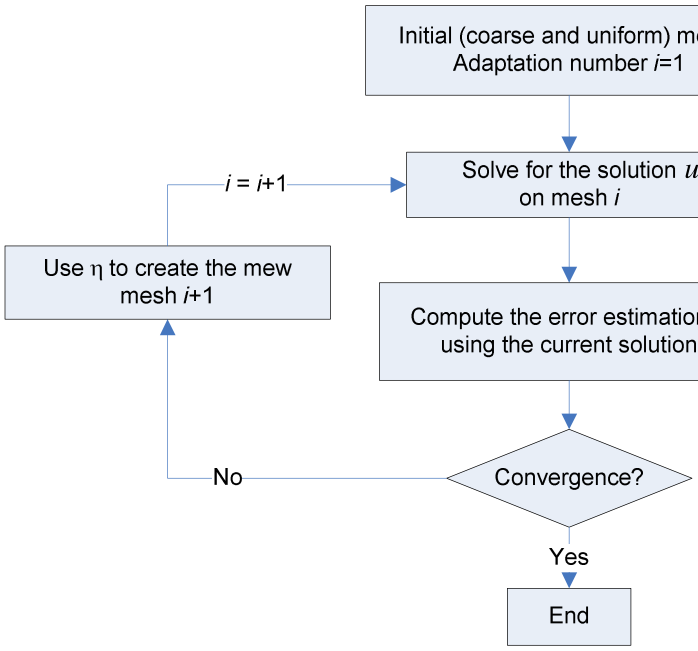

Initial (coarse and uniform) mesh; Adaptation number i=1
Solve for the solution u_h^i on mesh i
i = i+1
Use η to create the mew mesh i+1
Compute the error estimation η using the current solution u_h^i
Convergence?
No
Yes
End

Fig. I-1. Schematics of mesh adaptation

Obviously, we need to monitor and minimize the difference $\|u - u_h\|_{X,K}$ between the numerical solution and the exact solution *for each cell K at each iteration of the adaptation process* depicted in Figure 1 (X stands for an appropriate norm in the transport framework). There is no mathematical theory that provides a way for achieving this *locally*, i.e., cell by cell or element by element. But researchers have theoretically demonstrated [21] that the same goal can be achieved by minimizing a "reference" solution as follows:

$$\|u_{ref} - \Pi_h u_{ref}\|_{X,K}$$

where Π_h is the projection operator from a finer mesh solution to the coarser mesh solution. This theoretical result has nonetheless a very intuitive meaning: the error for any given cell is driven by the difference between the fine mesh solution and its projection on the coarse mesh (also known as the interpolation error). In the mathematical community, this error estimator is referred to as the "projection-based interpolation error of the reference solution" [22]. The *local* error estimator is therefore

$$\text{error estimator} = \left\| u_{ref} - \Pi_h u_{ref} \right\|_{X,K}$$

In order to obtain the "reference" solution u_{ref} needed at each stage i of the adaptation process, the current i^{th} mesh is globally refined and u_{ref} is sought on this temporary finer mesh. This is relatively inexpensive because the entire mesh adaptation process strives at delivering accuracy with the optimum mesh, i.e., with mesh containing the smallest number of cells. Adaptive meshes always contain far fewer cells than uniformly refined meshes. Note that the "reference" solution, obtained on a finer mesh, is also an approximation of the exact solution, but it is substantially more accurate than the approximation on the coarse mesh. The optimal meshes are obtained iteratively by minimizing the appropriate interpolation error in each step of the mesh adaptation until the user prescribed tolerance is reached.

There are several factors on which one can play to reduce the error in a cell chosen for refinement: (1) the cell can be subdivided in smaller cells, *h*-method, or (2) the polynomial order representation for the numerical solution of that cell can be increased, *p*-method. While both of these options perform better than uniform mesh refinement, neither are independently optimal [23].

- While *h-refinement* is indicated for regions where the solution is not smooth, such as domain corners or zones with significant material property discontinuities, it does not deliver the best convergence rate for regions where the solution is smooth.
- On the other hand, *p-refinement* is ideal for zones with a smooth solution but it should not be applied in regions where the solution is irregular, as near boundaries or material interfaces.

However, it is possible to combine the advantages of both methods into what is commonly termed the *hp*-refinement technique where the choice between a mesh subdivision and an increase in the polynomial order is based on a competitive minimization of local errors. Such *hp-methods* have been proven to *deliver exponential convergence* [24-26].

Moreover, *a posteriori* error estimator can be used to calculate the error of any quantities of interest. As a result, we can perform a so-called goal-oriented *hp*-adaptivity which reduces the total unknowns further [27] [28].

Computing with high spatial resolution accurate and converged solutions of the neutron balance equation is a challenging task. In this thesis, we introduce the *hp*-refinement techniques in FEM into the realm of nuclear engineering and investigate *hp*-strategies in the context of multigroup eigenvalue problems. We do so in the reduced framework of 1D multigroup diffusion as they will provide us with knowledge and experience on how the mesh adaptation procedure can be combined typical multigroup eigenproblem solvers. The lessons drawn from this work will prove useful when embarking on mesh adaptation for multigroup transport eigenproblems.

In Chapter II, we review the fundamental of the finite element method. In Chapter III, *hp*-adaptivity is introduced in the context of one-group calculations. In Chapter IV, we introduce new features in *hp*- multigroup diffusion, namely the embedding of the mesh adaptation process within the multigroup and eigenvalue problem context. We also extent the classical *hp*-adaptation technique to include Goal-oriented refinement in quantities of interest such as point wise fluxes, current, integrated reaction rates over specified zones, …

CHAPTER II

BRIEF INTRODUCTION TO FINITE ELEMENT METHOD

2.1 Preliminaries

The main objective of this chapter is to recall some fundamental concepts of Finite Element Method (FEM). As such, this chapter is not intended to be a exhaustive discussion regarding FEM; for more details, we refer the reader to these references [9] [29-31]; instead, we will reproduce here some basics aspects of FEM in a jargon which we believe is more accessible to the Nuclear Engineering community; furthermore, we will base our discussions on the multigroup neutron diffusion equation, whose application range is quite broad our field. Finally, we recognize upfront that the validity of using mesh refinement techniques with the neutron diffusion approximation rather than with the neutron transport equation may be arguably questionable when the mesh sizes become small. Nonetheless, the issue of investigating mesh adaptation in a multigroup context is a new and original topic; we foresee that lessons drawn from mesh adaptation in the multigroup diffusion setting will be the basis of mesh adaptation in the multigroup transport setting, where the variable depends not only upon space but upon space and angle. At the end of this chapter, we present a simple 1D multigroup MATLAB FEM demonstration code. For instance, a method, among others, used to deal with fission and scattering coupling between the various energy groups and based on a projective technique has been implemented in the above mentioned MATLAB code. The code supports different numbers of elements for different energy groups. Some calculation results are also included at the end of this chapter.

2.2 Neutron balance equations

The neutron transport equation, or Boltzmann equation, describes the neutron distribution in an elemental phase-space box consisting of (1) the usual physical space d^3r, (2) a solid angle

$d\Omega$ related to the neutron line of flight, and (3) an energy interval dE. The steady state neutron transport equation is given by:

$$\begin{aligned}\mathbf{\Omega}\cdot\vec{\nabla}\Psi(\mathbf{r},\mathbf{\Omega},E)+\Sigma(\mathbf{r},E)\Psi(\mathbf{r},\mathbf{\Omega},E) &= \int_{0}^{\infty} dE' \int_{(4\pi)} d\mathbf{\Omega}'\Sigma_s(\mathbf{r},\mathbf{\Omega}'\bullet\mathbf{\Omega},E'\to E)\Psi(\mathbf{r},\mathbf{\Omega}',E') \\ &+\frac{\chi(E)}{4\pi}\int_{0}^{\infty} dE'\nu\Sigma_f(\mathbf{r},E')\int_{(4\pi)} d\mathbf{\Omega}'\Psi(\mathbf{r},\mathbf{\Omega}',E') \\ &+S_{ext}(\mathbf{r},\mathbf{\Omega},E)\end{aligned}$$

The independent variables are: the neutron position $\mathbf{r}$, direction $\mathbf{\Omega}$, and energy E. The dependant variable is the neutron angular flux $\Psi(\mathbf{r},\mathbf{\Omega},E)$. The first and second terms on the left-hand-side represent the neutron losses due to, respectively, (1) leakage and (2) interaction with matter, where $\Sigma(\mathbf{r},E)$ is the neutron total cross section (probability of interaction per neutron track length). The three terms on the right-hand-side are, respectively, the gains of neutrons into the $d\Omega\ dE$ box due to (1) scattering, (2) fission, and (3) an extraneous source. Appropriate boundary conditions are required to close the system. If no (volumetric or boundary) external sources are present, the balance between losses and gains is enforced via the introduction of the eigenproblem such that:

$$\begin{aligned}\mathbf{\Omega}\cdot\vec{\nabla}\Psi(\mathbf{r},\mathbf{\Omega},E)+\Sigma(\mathbf{r},E)\Psi(\mathbf{r},\mathbf{\Omega},E) &= \int_{0}^{\infty} dE' \int_{(4\pi)} d\mathbf{\Omega}'\Sigma_s(\mathbf{r},\mathbf{\Omega}'\bullet\mathbf{\Omega},E'\to E)\Psi(\mathbf{r},\mathbf{\Omega}',E') \\ &+\frac{1}{\mathbf{K}_{eff}}\frac{\chi(E)}{4\pi}\int_{0}^{\infty} dE'\nu\Sigma_f(\mathbf{r},E')\int_{(4\pi)} d\mathbf{\Omega}'\Psi(\mathbf{r},\mathbf{\Omega}',E')\end{aligned}$$

If eigenvalue $\mathbf{K}_{eff}=1$, the system is said critical, if $\mathbf{K}_{eff}<1$ the system is said subcritical, if $\mathbf{K}_{eff}>1$, the system is said supercritical.

A widely used approximation to the neutron transport equation relies on the assumption of a linear angular dependence of the neutron angular flux and isotropic external volume sources, if any. Under these assumptions, the preceding eigenproblem is recast as follows, using the zero-th and first angular moments o the transport equation:

$$\vec{\nabla}\cdot\mathbf{J}(\mathbf{r},E)+\Sigma(\mathbf{r},E)\Phi(\mathbf{r},E)=\int_{0}^{\infty}dE\,\Sigma_s(\mathbf{r},E'\rightarrow E)\Phi(\mathbf{r},E')+\frac{1}{\mathbf{K}_{eff}}\chi(E)\int_{0}^{\infty}dE'\,\nu\Sigma_f(\mathbf{r},E')\Phi(\mathbf{r},E')$$

$$\frac{1}{3}\vec{\nabla}\cdot\Phi(\mathbf{r},E)+\Sigma(\mathbf{r},E)\mathbf{J}(\mathbf{r},E)=\int_{0}^{\infty}dE\,\Sigma_{s,1}(\mathbf{r},E'\rightarrow E)\mathbf{J}(\mathbf{r},E')$$

where the dependant variables are now the scalar flux and the scalar net current:

$$\Phi(\mathbf{r},E)=\int_{(4\pi)}d\mathbf{\Omega}'\,\Psi(\mathbf{r},\mathbf{\Omega}',E)$$

$$\mathbf{J}(\mathbf{r},E)=\int_{(4\pi)}d\mathbf{\Omega}'\;\mathbf{\Omega}\Psi(\mathbf{r},\mathbf{\Omega}',E)$$

The above equations are also known as the P_1 approximation to the neutron transport equation. After some further approximations regarding the second P_1 equation, we can arrive at Fick's law for the neutron current:

$$\mathbf{J}(\mathbf{r},E)=-\frac{1}{3\Sigma_{tr}(\mathbf{r},E)}\vec{\nabla}\cdot\Phi(\mathbf{r},E)=-\mathrm{D}(\mathbf{r},E)\vec{\nabla}\cdot\Phi(\mathbf{r},E)$$

$$\Sigma_{tr}(\mathbf{r},E)=\Sigma(\mathbf{r},E)-\int_{0}^{\infty}dE\,\Sigma_{s,1}(\mathbf{r},E'\rightarrow E)\frac{f(E')}{f(E)}$$

Customarily, Fick's law is substituted into the neutron balance equation, yielding an energy-dependant diffusion equation:

$$-\vec{\nabla}\cdot\mathrm{D}(\mathbf{r},E)\vec{\nabla}\cdot\Phi(\mathbf{r},E)+\Sigma(\mathbf{r},E)\Phi(\mathbf{r},E)=\int_{0}^{\infty}dE\,\Sigma_s(\mathbf{r},E'\rightarrow E)\Phi(\mathbf{r},E')$$
$$+\frac{1}{\mathbf{K}_{eff}}\chi(E)\int_{0}^{\infty}dE'\,\nu\Sigma_f(\mathbf{r},E')\Phi(\mathbf{r},E')$$

Finally, the energy variable is systematically treated by dividing the energy domain $\left[E_{\min,}E_{\max}\right]$ into smaller G energy intervals $\left[E_{g+1,}E_g\right]$, also called energy groups or simply groups, with $E_1=E_{\max}$ and $E_{G+1}=E_{\min}$. Integrating the energy-dependant equations on these energy intervals yields:

$$-\vec{\nabla}\cdot \mathrm{D}_{\mathrm{g}}(\mathbf{r})\vec{\nabla}\cdot\Phi_{g}(\mathbf{r})+\Sigma_{r,g}(\mathbf{r})\Phi_{g}(\mathbf{r})=\sum_{\mathrm{g'\neq g}}\Sigma_{s,g'\to g}(\mathbf{r})\Phi_{g'}(\mathbf{r})+\frac{1}{\mathbf{K}_{eff}}\chi_{g}\sum_{\mathrm{g'=1}}^{G}\nu\Sigma_{f,\mathrm{g'}}(\mathbf{r})\Phi_{\mathrm{g'}}(\mathbf{r})$$

$$\mathrm{g}=1,\ldots,G$$

The multigroup scalar fluxes $\Phi_g(\mathbf{r})$ are only space-dependent variables. The multigroup diffusion equations and the associated solver are described thoroughly in Chapter IV.

2.3 Formulations of elliptic partial differential equations

2.3.1 Classification of partial differential equations

By analogy with the conic sections – i.e., ellipse, parabola, and hyperbola - linear partial differential equations of second order have been classified as *elliptic*, *parabolic* and *hyperbolic* based on the determinant of their principal terms (i.e., the terms involving the highest order derivatives). If the operator of order two is positive-definite, the PDE is elliptic, if the determinant of the operator is zero, the PDE is parabolic, and in all other case, the PDE is said hyperbolic. One example of elliptic PDE is the steady state neutron diffusion equation. It should be noted that unsteady diffusion equation is parabolic and that the neutron transport equation is hyperbolic. We usually refer to the coefficients present in the PDE, the functions appearing in boundary and initial conditions, and the domain on which the PDE is required to hold as the *data of PDE problem*. A PDE problem is said to be *well-posed* if:

1. A solution to the problem exists,
2. The solution is unique, and,
3. The solution depends continuously on the problem data.

2.3.2 Classical formulation of multidimensional elliptic boundary value problems

We will use a multi-dimensional neutron diffusion source problem with one energy group as an example to describe the variational formulation used. Considering a source problem does not restrict our developments since eigenproblems are themselves solved with a power iteration technique which recasts the eigenproblems as an iterative scheme acting upon a source problem whose source is modified at each iteration. Even though the applications of the work presented in this thesis are one-dimensional (1-D), multi-dimensional equations will be employed to describe the problem as no theoretical assumptions will restrict this to 1-D. When needed, we will later simplify the equations into their 1-D form.

Let Ω be a bounded domain in R^{dim}, which is also open and connected. The boundary $\Gamma=\partial\Omega$ of the domain can be generally split into three disjoint parts Γ_d, Γ_{n}, Γ_{c}, on which Dirichlet, Neumann and Cauchy boundary conditions hold. (multigroup albedo and periodic boundary conditions are not discussed here because they can simply be recast as a Cauchy condition for a specific group, or be treated as constraints.)

A strong formulation of the problem is as follows:

Find $\phi(\mathbf{x})$, $\mathbf{x}$ belonging to the domain Ω, such that,

$$-\vec{\nabla}\cdot(D(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}))+\Sigma_r(\mathbf{x})\phi(\mathbf{x})=s(\mathbf{x}) \tag{2.1}$$

In reactor physics, the coefficients $D(\mathbf{x})$ and $\Sigma_r(\mathbf{x})$ are called diffusion coefficient and removal cross section and are medium-specific. Because the diffusion coefficient can be obtained from other neutron cross sections, we usually refer to all material data as *cross sections*. $s(\mathbf{x})$ is the fixed extraneous volumetric source. We call $\phi(\mathbf{x})$ *neutron flux*, and

$$\vec{J}(\mathbf{x})=-D(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}) \tag{2.2}$$

is the *neutron net current vector*. Comparing the neutron diffusion equation with general elliptic problems, the diffusion equation does not contain first order spatial derivative terms and the diffusion coefficient D is, in general, a scalar.

We describe now some typical boundary conditions (B.C.) :

- Dirichlet boundary conditions (B.C.) are:

$$\phi(\mathbf{x})=\phi_D(\mathbf{x}) \qquad on \quad \Gamma_D \tag{2.3}$$

- Neumann B.C. (*net incoming current*) are:

$$\mathbf{n}(\mathbf{x}) \cdot D(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}) = D(\mathbf{x})\frac{\partial \phi}{\partial n} = J_{-\mathbf{n}}(\mathbf{x}) = g(\mathbf{x}) \qquad on \;\; \Gamma_N \tag{2.4}$$

- Cauchy B.C. (*partial incoming current*) are:

$$\mathbf{n}(\mathbf{x}) \cdot D(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}) + \frac{\phi(\mathbf{x})}{2} = D\frac{\partial \phi}{\partial n} + \frac{\phi}{2} = 2J_{\mathbf{n}}^{-} = g(\mathbf{x}) \qquad on \;\; \Gamma_C \tag{2.5}$$

where $g(\mathbf{x})$ is a surface source on the boundary. **n** is outward unit normal vector defined on boundary.

Note: $J_{\vec{\mathrm{v}}}^{-}(\mathbf{x})$ represents the partial current at point **x** corresponding to the rates at which neutrons flow through a unit surface area from its negative side to its positive side ($\vec{\mathrm{v}}$ is the surface outward unit vector). $J_{\vec{\mathrm{v}}}^{+}(\mathbf{x})$ is the partial current at point **x** corresponding to the rates at which neutrons flow through a unit surface area from its side positive to its negative side. $J_{\vec{\mathrm{v}}}^{+}(\mathbf{x})$ and $J_{\vec{\mathrm{v}}}^{-}(\mathbf{x})$ are always positive quantities. $J_{\vec{\mathrm{v}}}(\mathbf{x})$ is the algebraic neutron net current (net flow of neutrons at a point **x** per unit area whose normal outward unit vector is $\vec{\mathrm{v}}$). This value can be negative. $\vec{J}(\mathbf{x}) \cdot \vec{\mathrm{v}} = J_{\vec{\mathrm{v}}}(\mathbf{x}) = J_{\vec{\mathrm{v}}}^{+}(\mathbf{x}) - J_{\vec{\mathrm{v}}}^{-}(\mathbf{x})$. All these currents are scalar values, different from the neutron current vector of Eq. (2.2).)

In reactor analysis, we often meet problems for which neutrons diffuse through a structure composed of several materials. Each material has positive constant cross sections, which results into piece-wise constant cross sections in the whole domain. In this work, *interface conditions* are added based on physical facts - *the continuity of the flux and of the normal component of the current*. In our work, we do not consider flux discontinuity factors, or assembly discontinuity factors, arising from the generalized equivalence theory and accounting for the discrepancy in approximating a transport problem with a diffusion problem.

The solution of the differential problem Eq. (2.1) is called a "strong" formulation because it demands the existence of second order derivatives. The solution at least belongs to second order Sobolev space H_E^2 constrained by boundary and interface conditions within each material.

2.3.3 Variational formulation

The first step in deriving a variational formulation of the diffusion problem consists of multiplying both sides of Eq. (2.1) by a test function $\varphi(\mathbf{x})$. The test function $\varphi(\mathbf{x})$ is equal to 0 on the Dirichlet boundary because we already known the solution there. Then, by integrating by parts (i.e., using Green's theorem or the divergence theorem), we obtain:

$$\begin{aligned}
&\int_\Omega -\vec{\nabla}\cdot(D(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}))\varphi(\mathbf{x})d\mathbf{x} \\
&\quad=\int_\Omega[\vec{\nabla}\varphi(\mathbf{x})\cdot(D(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}))-\vec{\nabla}\cdot(\varphi(\mathbf{x})D(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}))]d\mathbf{x}\quad (product\quad rule)\\
&\quad=\int_\Omega[D(\mathbf{x})\vec{\nabla}\varphi(\mathbf{x})\cdot\vec{\nabla}\phi(\mathbf{x})d\mathbf{x}-\vec{\nabla}\cdot(\varphi(\mathbf{x})D(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}))]d\mathbf{x}\quad (scalar\quad D)\\
&\quad=\int_\Omega D(\mathbf{x})\vec{\nabla}\varphi(\mathbf{x})\cdot\vec{\nabla}\phi(\mathbf{x})d\mathbf{x}-\oint_{\partial\Omega}\mathbf{n}(\mathbf{x})\cdot(\varphi(\mathbf{x})D(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}))d\mathbf{s}\quad (Green's\quad theorem)\\
&\quad=\int_\Omega D(\mathbf{x})\vec{\nabla}\varphi(\mathbf{x})\cdot\vec{\nabla}\phi(\mathbf{x})d\mathbf{x}-\oint_{\partial\Omega}\varphi(\mathbf{x})J_{-\mathbf{n}}(\mathbf{x})d\mathbf{s}\\
&\quad=\int_\Omega D(\mathbf{x})\vec{\nabla}\varphi(\mathbf{x})\cdot\vec{\nabla}\phi(\mathbf{x})d\mathbf{x}-\oint_{\Gamma_N}\varphi(\mathbf{x})g(\mathbf{x})d\mathbf{s}-\oint_{\Gamma_C}\varphi(\mathbf{x})g(\mathbf{x})d\mathbf{s}+\oint_{\Gamma_C}\varphi(\mathbf{x})\frac{\phi(\mathbf{x})}{2}d\mathbf{s}\\
&\quad=\int_\Omega D(\mathbf{x})\vec{\nabla}\varphi(\mathbf{x})\cdot\vec{\nabla}\phi(\mathbf{x})d\mathbf{x}-\oint_{\Gamma_N\cup\Gamma_C}\varphi(\mathbf{x})g(\mathbf{x})d\mathbf{s}+\oint_{\Gamma_C}\varphi(\mathbf{x})\frac{\phi(\mathbf{x})}{2}d\mathbf{s}
\end{aligned}$$

Finally, we get,

$$\int_\Omega[D\vec{\nabla}\phi\cdot\vec{\nabla}\varphi+\Sigma_r\phi\varphi]d\mathbf{x}+\frac{1}{2}\oint_{\Gamma_C}\phi\varphi d\mathbf{s}=\int_\Omega s\varphi d\mathbf{x}+\oint_{\Gamma_N\cup\Gamma_C}g\varphi d\mathbf{s}\tag{2.6}$$

The right-hand-side of this equation is identified as a linear functional $l(\varphi)$ of test function $\varphi(\mathbf{x})$. Similarly, the left-hand side is identified as a bilinear functional $b(\phi,\varphi)$. (Bilinear means with fixed ϕ, the left-hand side is linear in φ and, with fixed φ , is linear in ϕ.) Then an abstract variational formation can be written as

$$\begin{aligned}
&V=\{\varphi(\mathbf{x}):\varphi=0\quad on\quad \Gamma_D,\ \}\\
&\phi(\mathbf{x})\in\phi_D(\mathbf{x})+V\\
&b(\phi,\varphi)=l(\varphi),\quad\forall\varphi\in V
\end{aligned}\tag{2.7.a}$$

where $\phi_D(\mathbf{x})$ is called as a *lift function* which satisfies the Dirichlet boundary conditions.

We have:

$$\begin{aligned} b(\phi,\varphi) &= \int_\Omega [D\vec{\nabla}\phi \cdot \vec{\nabla}\varphi + \Sigma_r \phi\varphi] d\mathbf{x} + \frac{1}{2}\oint_{\Gamma_C} \phi\varphi d\mathbf{s} \\ l(\varphi) &= \int_\Omega s\varphi d\mathbf{x} + \oint_{\Gamma_N \cup \Gamma_C} g\varphi d\mathbf{s} \end{aligned} \tag{2.7.b}$$

This formulation is called *variational boundary value problem (**VBVP**).*

With some regularity conditions on geometry, material and source data, the variational formulation is equivalent to the classical formulation. Discussing the necessary conditions for the equivalence of these two forms is outside the scope of this document [32] [33].

2.3.4 Equivalence with a minimization problem

We may also write the neutron diffusion problem in the following form,

$$\begin{aligned} & F(\phi(\mathbf{x})) = \frac{1}{2} b(\phi,\phi) - l(\phi) \\ & \phi \in \phi_D + V \\ & F(\phi) \to \min \end{aligned} \tag{2.8}$$

where F is a functional. The variational problem is equivalent to a functional minimization problem if the bilinear form is *symmetric* and *positive-definite*. The symmetry condition

$$b(\phi,\varphi) = b(\varphi,\phi), \qquad \forall \phi,\varphi \in \phi_D + V \tag{2.9}$$

is equivalent to the assumption that the first order spatial derivative terms of the elliptic PDE vanish, which is always the case for neutron diffusion. Clearly, the bilinear form is always positive. For instance, in 2-D Cartesian geometry, we have

$$b(\phi,\phi) = \int_\Omega [D(\phi_x^2 + \phi_y^2) + \Sigma_r \phi^2] d\mathbf{x} + \frac{1}{2}\oint_{\Gamma_C} \phi^2 d\mathbf{s} = 0$$

meaning that

$$\int_\Omega D(\phi_x^2 + \phi_y^2) d\mathbf{x} = \int_\Omega \Sigma_r \phi^2 d\mathbf{x} = \oint_{\Gamma_C} \phi^2 d\mathbf{s} = 0$$

D is always greater than zero. If absorption is present in some region or if a fixed flux condition holds or if a partial incoming boundary condition holds, we can assure that if $b(\phi,\phi)=0$, then the flux ϕ is equal to 0 everywhere because b is positive-definite. If there are no absorption and only Neumann boundary conditions holds, another condition is needed to assure that the number of incoming and outgoing neutrons are identical (known as the compatibility condition). Again, this is only meaningful from a strict mathematical standpoint; in reality, absorption is always present in a real-life system.

Now we can expand our function space *V* to a space in which all functions satisfy

$$b(\phi,\phi)<\infty$$

We also call the square root of $b(\phi, \phi)$ the energy norm of ϕ.

$$\|\phi\|_E=\sqrt{b(\phi,\phi)} \tag{2.10}$$

It needs to be pointed out that the mathematics of variational boundary-value problems does not allow for imposing the Dirichlet BC at a single point, at a finite number of points or, in general, on a subset of zero-measure.

2.3.5 Sobolev spaces and well-posed VBVP

The Sobolev space H^k consists of functions u whose first kth derivatives belong to L^2. The space has the following inner product and norm:

$$\begin{aligned}(u,v)_k &:= \sum_{|\kappa|\le k}(D^\kappa u, D^\kappa v)\\ \|u\|_k &= \sqrt{(u,u)_k}\end{aligned} \tag{2.11}$$

where (in 2-D for example)

$$\boldsymbol{\kappa}=[\kappa_1,\kappa_2]^T, \qquad |\boldsymbol{\kappa}|=\kappa_1+\kappa_2$$

, with κ_1 and κ_2 non-negative integers, and

$$D^\kappa u:=\frac{\partial^{\kappa_1+\kappa_2}u}{\partial x^{\kappa_1}\partial y^{\kappa_2}}$$

In particular, the norm in space H^1 is

$$\|u\|_1=\left[\int_\Omega(u^2+u_x^2+u_y^2)dxdy\right]^{1/2} \tag{2.12a}$$

And the semi -norm in H^1 is

$$|u|_1 = \left[\int_\Omega (u_x^2 + u_y^2) dxdy \right]^{1/2} \tag{2.12b}$$

For k=0, the H^0 norm is,

$$\|u\|_0 = \left[\int_\Omega u^2 dxdy \right]^{1/2} \tag{2.13}$$

which is simply the L_2 norm.

A function u such that $u \in H^k$ implies that $u \in C^{k-1}$ in one dimension. The situation is not as simple in two and three dimensions. $u \in H^k$ can only imply that $u \in C^{k-2}$ in the multi-dimension case.

With the Sobolev space definitions now provided, we can make now describe precisely the variational formulation. The space of test functions V is a subspace of the Sobolev space consisting of functions vanishing at x=0,

$$\begin{aligned} &\phi(\mathbf{x}) \in \phi_D(\mathbf{x}) + H_0^1 \\ &b(\phi, \varphi) = l(\varphi), \quad \forall \varphi \in H_0^1 \end{aligned} \tag{2.14}$$

2.3.6 Continuity and coercivity of a bilinear form b(ϕ, φ)

A bilinear form $b(\phi, \varphi)$ is continuous in H^1 if there exists a constant α>0 such that

$$| b(\phi, \varphi) | \leq \alpha \|\phi\|_1 \|\varphi\|_1, \qquad \forall \phi, \varphi \in H^1 \tag{2.15}$$

A bilinear form $b(\phi, \varphi)$ is coercive in H^1 if there exists a constant β>0 such that

$$b(\phi, \phi) \geq \beta \|\phi\|_1^2, \qquad \forall \phi \in H^1 \tag{2.16}$$

(coercivity may also be described as H^1-ellipticity or positive-definiteness)

Actually, α and β provide a lower and an upper bound of the eigenvalue spectrum of the bilinear form.

With the continuity and coercivity, Lax-Milgram theorem implies that the VBVP is well-posed. These conditions are only sufficient.

2.4 Galerkin formulation

2.4.1 Galerkin method

The basic idea of *Galerkin's method* is to approximate a system with a reduced number of degrees of freedom (or DOF for short). In order to apply Galerkin's technique to a variational boundary value problem, let us construct N linear independent functions e_j, j=1,…,N belonging to V_{hp}, a finite subspace of V. We name these functions the *basis functions* and N the *global number of degrees of freedom*. Then, we test the equation with another N-dimensional space to find a solution in this functional space. We then arrive at an N-dimensional *approximation* to the *variational boundary value problem*. Because we usually employ the same functional space for the basis and the test spaces, the approximation problem is:

$$\begin{aligned} &\phi_{hp} \in \phi_D + V_{hp} \\ &b(\phi_{hp}, \varphi_{hp}) = l(\varphi_{hp}), \qquad \forall \varphi_{hp} \in V_{hp} \end{aligned} \tag{2.17}$$

Now, let us represent ϕ_{hp} by

$$\phi_{hp}(\mathbf{x}) = \phi_D(\mathbf{x}) + \sum_{j=1}^{N} \phi_j e_j(\mathbf{x}) \tag{2.18}$$

We get a more specific form of the problem

$$b(\phi_D(\mathbf{x}) + \sum_{j=1}^{N} \phi_j e_j(\mathbf{x}), \varphi_h) = l(\varphi_h), \qquad \forall \varphi_{hp} \in V_{hp}$$

Generally we use basis functions as *test functions* directly to keep symmetry of the resulting system of linear equations:

$$b(\phi_D(\mathbf{x}) + \sum_{j=1}^{N} \phi_j e_j(\mathbf{x}), e_i(\mathbf{x})) = l(e_i(\mathbf{x})), \qquad i = 1, 2, \cdots, N \tag{2.19}$$

The matrix form of above equations is,

$$\mathbf{Au=f} \tag{2.20.a}$$

Where,

$$\begin{aligned} a_{ij} &= b(e_j(\mathbf{x}), e_i(\mathbf{x})) \\ f_i &= l(e_i(\mathbf{x})) - b(\phi_D(\mathbf{x}), e_i(\mathbf{x})) \end{aligned} \tag{2.20.b}$$

A is called the *global stiffness matrix*, while **f** is the *global modified load vector*.

In principle, the approximate solution ϕ_{hp} depends only upon the space V_{hp} and is independent of the choice of basis functions e_j, as long as they span the same approximate subspace. In practice, however, the choice of the basis functions affects the conditioning of the global matrix **A** and, due to round-off errors, may influence on accuracy of approximate solution.

It is easy to demonstrate that in the case of symmetric and positive definite, Galerkin's method is equivalent to the *Ritz method*. The Galerkin method is also known as the *Bubnov-Galerkin method*. In a more general approach, known as Petrov-Galerkin method, the test functions are chosen in a different space than the basis function space.

From the Eq. (2.17), we can easily ontain the following Galerkin orthogonality relation:

$$b(\phi - \phi_{hp}, \varphi_{hp}) = b(e_{hp}, \varphi_{hp}) = 0, \qquad \forall \varphi_{hp} \in V_{hp} \tag{2.21}$$

The residual or approximation error e_{hp} is orthogonal to the test function space in the sense of bilinear form. If bilinear form is symmetric, we have another useful relation:

$$b(\phi - \phi_{hp}, \phi - \phi_{hp}) = b(\phi, \phi) - b(\phi_{hp}, \phi_{hp}) \tag{2.22}$$

2.4.2 Cea's lemma and *a priori* error estimations

If the exact solution ϕ is fully contained within the basis function space, then we will obtain the exact solution. However, this is rarely the case. What we want is that the larger space we apply, the better the approximation solution will be. Cea's lemma confirmed this:

If the bilinear form is both continuous and coercive as in Eqs. (2.15) and (2.16), the error ϕ-ϕ_{hp} is bounded by:

$$\left\| \phi - \phi_{hp} \right\|_E \le \frac{\alpha}{\beta} \min_{\psi_{hp} \in \phi_D + V_{hp}} \left\| \phi - \psi_{hp} \right\|_E \tag{2.23}$$

Especially when the bilinear form is SPD (symmetric and positive-definite), we have:

$$\left\|\phi-\phi_{hp}\right\|_E=\min_{\psi_{hp}\in\phi_D+V_{hp}}\left\|\phi-\psi_{hp}\right\|_E \tag{2.24}$$

Furthermore, based on interpolation theory, and under the following assumptions,

1. $\phi\in H_0^1$ and $\phi_h\in V_h\subset H_0^1$ are the exact solution and the approximate solutions, respectively, of the variational boundary value problem;
2. $b(\phi,\varphi)$ is a symmetric, continuous and H^1-elleptic bilinear form;
3. V_{hp} consists of complete piecewise-polynomial functions of degree p on a uniform family of meshes Δ_h;
4. $\phi\in H_0^1\cap H^{p+1}$;

we then have the following *a priori* error estimations (in the H^1 norm, H^1 semi-norm, energy norm and L_2 norm):

$$\left\|\phi-\phi_h\right\|_1\le Ch^p\left\|\phi\right\|_{p+1}$$
$$\left|\phi-\phi_h\right|_1\le Ch^p\left\|\phi\right\|_{p+1}$$
$$\left\|\phi-\phi_h\right\|_E\le Ch^p\left\|\phi\right\|_{p+1}$$
$$\left\|\phi-\phi_h\right\|_0\le Ch^{p+1}\left\|\phi\right\|_{p+1}$$

These error estimates provide us with the expected algebraic convergence rate (if the solution is smooth enough) but such error estimates are usually useless because (1) the constants C are extremely difficult to obtain in real-life situations and (2) these estimates are global quantities whereas we wish to determine errors *locally* in order to proceed with local refinements.

2.5 Finite element method

The Finite Element Method (FEM) is a special case of Galerkin's method. In FEM, the solution domain is partitioned into disjoint simple sub-domains called mesh cells or elements (a triangulation of the domain). For each element, we introduce polynomial shape functions, which are eventually glued together or expanded forming the globally defined continuous basis functions e_j. The support of finite element basis functions is always contained within a few adjacent elements.

2.5.1 Triangulation

The purpose of the domain triangulation is to map the initial domain with a finite number of non-overlapping polygonal cells. (There may be several types of cells involved, for example, triangles and quadrilaterals in 2-D, tetrahedrons, hexahedrons and prisms in 3-D…) Each cell possesses a *mapping* between the mesh cell itself and a unique reference master element. This mapping defines the global coordinates of all the vertices. Finally, we construct the *connectivity array*, i.e., the relationship between cells, faces, edges and vertex.

2.5.2 Master element and shape functions

The master element is a mean to describe various real cells with a single elementary master cell. In 1-D, we often find two kinds of master elements. One kind of master cell extends from -1 to 1 whereas the other kind spans 0 to 1 as illustrated in Fig. II-1. We can describe these in a mathematical way as an example in 1-D,

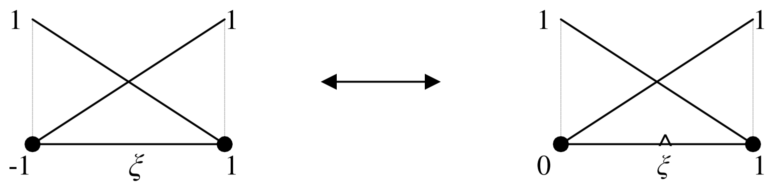


Fig. II-1. Master elements with different definition domain

$$
\begin{aligned}
&\xi = 2\hat{\xi} - 1; \qquad \hat{\xi} = \frac{\xi + 1}{2} \\
&l(\xi) = l(2\hat{\xi} - 1) = \hat{l}(\hat{\xi}) \\
&\frac{d\hat{l}}{d\hat{\xi}} = 2\frac{dl}{d\xi}
\end{aligned}
\tag{2.25}
$$

Various types of polynomial shape functions are available and a substantiated choice is required. Polynomials are often used because they are smooth functions, whose values and derivatives are easy to evaluate. A very common type of shape functions are the Lagrange equally spaced interpolation polynomials. However, hierarchical shape functions will prove more useful in our study. With hierarchical shape functions, there is no need to recalculate element matrix when modifying an element polynomial order p . This stems from the fact that hierarchical shape functions are appending higher order polynomials to the shape functions but not reconstruct all shape functions when local degrees of freedom is increased. In 1-D hierarchical shape functions, the first two shape functions are the linear shape functions, linking the two vertices of the element. For higher orders, p-1, the shape functions are also known as *bubble functions* because their values on two vertices vanish. Bubbles functions collocate all nodal values at fixed, polynomial order-independent locations. E.g., the nodal value locations for higher order polynomial shape functions in 1-D are simply the middle point of the element. This key property will lead to *significant simplifications* in the implementation of polynomial order refinement since all higher order nodal values will be collocated at the element middle point. Note that for the two linear shape functions, the nodal values are located at the vertices and represent values of the solution field itself. The mid-node values do not represent values of the solution field itself, but a linear combination of the solution field and its derivatives.

Below, we list three common 1-D shape functions. (given on the [-1,1] master element).

1) Lagrange shape functions,

$$l_k(\xi) = \prod_{l=0, i\neq k}^{p} \frac{\xi - \xi_l}{\xi_k - \xi_l}, \qquad k = 0,1,\ldots,p, \qquad \xi \in [-1,1]$$

$$l'_k(\xi) = \sum_{j=0, j\neq k}^{p} \frac{1}{\xi_k - \xi_j} \prod_{l=0, l\neq j,k}^{p} \frac{\xi - \xi_l}{\xi_k - \xi_l} \tag{2.26.a}$$

Nodes are usually even distributed between -1 to 1.

2) Lobatto shape functions (hierarchical functions),

$$
\begin{aligned}
&l_0(\xi)=\frac{1-\xi}{2},\ \ l_1(\xi)=\frac{1+\xi}{2},\ \ \xi\in[-1,1]\\
&l_0'(\xi)=-\frac{1}{2},\ \ l_1'(\xi)=\frac{1}{2},\ \ \xi\in[-1,1]\\
&l_k(\xi)=1-\frac{1}{\sqrt{2/(2k-1)}}\int_{-1}^{\xi}L_{k-1}(t)dt=\frac{L_k(\xi)-L_{k-2}(\xi)}{\sqrt{2/(2k-1)}},\ \ \xi\in[-1,1],\ \ k\geq 2\\
&l_k'(\xi)=-\frac{L_{k-1}(\xi)}{\sqrt{2/(2k-1)}},\ \ \xi\in[-1,1]
\end{aligned}
\tag{2.26.b}
$$

L_k(x) is the k^{th} order Legendre polynomial.

3) Peano shape functions (hierarchical),

$$
\begin{aligned}
&l_0(\xi)=\frac{1-\xi}{2},\ \ l_1(\xi)=\frac{1+\xi}{2},\ \ \xi\in[-1,1]\\
&l_0'(\xi)=-\frac{1}{2},\ \ l_1'(\xi)=\frac{1}{2},\ \ \xi\in[-1,1]\\
&l_2(\xi)=l_0(\xi)l_1(\xi) \qquad l_2'(\xi)=l_0'(\xi)l_1(\xi)+l_0(\xi)l_1'(\xi)\\
&l_k(\xi)=l_{k-1}(\xi)(l_1(\xi)-l_0(\xi)) \qquad k\geq 3\\
&l_k'(\xi)=l_{k-1}'(\xi)(l_1(\xi)-l_0(\xi))+l_{k-1}(\xi)(l_1'(\xi)-l_0'(\xi))
\end{aligned}
\tag{2.26.c}
$$

Ref. [29] and Ref.[34] propose different definitions of Peano shape functions. The above definition is taken from Ref. [29] The first two are linear function corresponding two vertices; nodal values of other shape functions l_n(x) are the n^{th} derivatives at the center. Yet, another type of shape functions, not mentioned in any references we were aware off but used by Prof. Demkowicz in its 1D-*hp* code [34] is as follows, it differs from the above definition of Peano shape functions only by the recursive relation, which is:

$$
l_k(\xi)=l_{k-1}(\xi)\frac{l_1(\xi)-l_0(\xi)}{2}
\tag{2.26.d}
$$

We designate it as modified Peano shape functions for convenience.

2.5.3 Real elements and mapping

The geometry of element is usually described with the same shape functions for the approximate solution. We can simply describe this with the following equation in 1-D,

$$x = \sum_{i=0}^{p} x_i l_i(\xi) \tag{2.27}$$

This is usually referred to as the isoparametric element [35]. The coefficients x_i are also known as geometry degree of freedom. This mapping must be invertible.

When, for instance, quadrilateral cells are used in 2-D, we usually use a bilinear transform mapping a global coordinate system to the local coordinate system for a element:

$$\begin{bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{bmatrix} = \sum_{i=1}^{4} \begin{bmatrix} x_i \\ y_i \end{bmatrix} l_i(\xi,\eta) \tag{2.28.a}$$

Here,

$$\begin{aligned} l_1(\xi,\eta) &= l_0(\xi)l_0(\eta); \quad l_2(\xi,\eta) = l_1(\xi)l_0(\eta) \\ l_3(\xi,\eta) &= l_0(\xi)l_1(\eta); \quad l_4(\xi,\eta) = l_1(\xi)l_1(\eta) \end{aligned} \tag{2.28.b}$$

are the shape functions corresponding to the four vertices and (x_i, y_i) are the coordinates of the i^{th} vertex.

$$\begin{bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \frac{(1-\xi)(1-\eta)}{4} + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \frac{(1+\xi)(1-\eta)}{4} + \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} \frac{(1-\xi)(1+\eta)}{4} + \begin{bmatrix} x_4 \\ y_4 \end{bmatrix} \frac{(1+\xi)(1+\eta)}{4}$$

$$x(\xi,\eta) = \frac{(1-\xi)(1-\eta)x_1 + (1+\xi)(1-\eta)x_2 + (1-\xi)(1+\eta)x_3 + (1+\xi)(1+\eta)x_4}{4}$$

$$y(\xi,\eta) = \frac{(1-\xi)(1-\eta)y_1 + (1+\xi)(1-\eta)y_2 + (1-\xi)(1+\eta)y_3 + (1+\xi)(1+\eta)y_4}{4}$$

The Jacobian matrix of the transformation is:

$$\mathrm{J}_e(\xi,\eta) = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}$$

$$\left(\mathrm{J}_e(\xi,\eta)\right)^{-1} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} = \frac{\begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix}}{\begin{vmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{vmatrix}} = \frac{\begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix}}{\det(\mathrm{J}_e(\xi,\eta))} \tag{2.29}$$

Specifically,

$$\mathrm{J}_e(\xi,\eta) = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} = \frac{1}{4} \begin{bmatrix} (1-\eta)(x_2-x_1)+(1+\eta)(x_4-x_3) & (1-\xi)(x_3-x_1)+(1+\xi)(x_4-x_2) \\ (1-\eta)(y_2-y_1)+(1+\eta)(y_4-y_3) & (1-\xi)(y_3-y_1)+(1+\xi)(y_4-y_2) \end{bmatrix}$$

The Jacobian matrix is position-dependent.

In the special case when x_1=x_3; x_2=x_4; y_1=y_2; y_3=y_4,

$$\mathrm{J}_e(\xi,\eta)=\begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}=\begin{bmatrix} \frac{h_x}{2} & 0 \\ 0 & \frac{h_y}{2} \end{bmatrix}$$

$$h_x = x_2 - x_1 = x_4 - x_3; \quad h_y = y_3 - y_1 = y_4 - y_2$$

It is a constant matrix.

2.5.4 Integral quadrature; element mass matrix; stiffness matrix; and load vector

Since Gauss-Legendre quadrature of order *p* can deliver exact integrals for polynomials of order up to 2*p*-1, it is widely used in calculating element matrix and element load vector.

Element matrix and load vector arise from following integrals:

$$b_e(\phi_h,\varphi_h)=\iint\limits_{\Omega_e} D_e[(\phi_h)_x(\varphi_h)_x+(\phi_h)_y(\varphi_h)_y]dxdy+\iint\limits_{\Omega_e}\Sigma_r^e\phi_h\varphi_h dxdy$$

$$(s,\varphi_h)_e=\iint\limits_{\Omega_e}\varphi_h s dxdy$$

ϕ_h is the approximation solution and φ_h is the trial function. *s* is the source term.

After coordinate transformation, we obtain:

$$\varphi_{h0}=\varphi_h(x(\xi,\eta),y(\xi,\eta)); \quad \phi_{h0}=\phi_h(x(\xi,\eta),y(\xi,\eta))$$

$$\begin{aligned} b_e(\phi_h,\varphi_h)=&\iint\limits_{\Omega_0} D_e\left[(\varphi_{h0})_\xi \quad (\varphi_{h0})_\eta\right](\mathrm{J}_e(\xi,\eta))^{-1}(\mathrm{J}_e^T(\xi,\eta))^{-1}\begin{bmatrix}(\phi_{h0})_\xi \\ (\phi_{h0})_\xi\end{bmatrix}\left|\mathrm{J}_e(\xi,\eta)\right|d\xi d\eta \\ &+\iint\limits_{\Omega_0}\Sigma_r^e\varphi_{h0}\phi_{h0}\left|\mathrm{J}_e(\xi,\eta)\right|d\xi d\eta \end{aligned} \tag{2.30}$$

$$s_0=s(x(\xi,\eta),y(\xi,\eta))$$

$$(s,\varphi_h)_e=\iint\limits_{\Omega_e}\varphi_{h0}s_0\left|\mathrm{J}_e(\xi,\eta)\right|d\xi d\eta$$

Generally, we need also perform the coordinate transformation for material data if they are not constant within the element *e*.

Expanding ϕ_{h0} and φ_{h0} with the element shape functions, we obtain:

$$\varphi_{h0} = \begin{bmatrix} d_1 & d_2 & \cdots & d_P \end{bmatrix} \begin{bmatrix} l_1(\xi,\eta) \\ l_2(\xi,\eta) \\ \vdots \\ l_P(\xi,\eta) \end{bmatrix} = \mathbf{d}^T \mathbf{L}(\xi,\eta)$$

$$\phi_{h0} = \begin{bmatrix} c_1 & c_2 & \cdots & c_P \end{bmatrix} \mathbf{L}(\xi,\eta) = \mathbf{c}^T \mathbf{L}(\xi,\eta) = \mathbf{L}^T(\xi,\eta)\mathbf{c}$$

$$\begin{bmatrix} (\varphi_{h0})_\xi & (\varphi_{h0})_\eta \end{bmatrix} = \mathbf{d}^T \begin{bmatrix} l'_{1\xi}(\xi,\eta) & l'_{1\eta}(\xi,\eta) \\ l'_{2\xi}(\xi,\eta) & l'_{2\eta}(\xi,\eta) \\ \vdots & \vdots \\ l'_{P\xi}(\xi,\eta) & l'_{P\eta}(\xi,\eta) \end{bmatrix}_{P\times 2} = \mathbf{d}^T \mathbf{X}(\xi,\eta)$$

$$\begin{bmatrix} (\phi_{h0})_\xi & (\phi_{h0})_\eta \end{bmatrix} = \mathbf{c}^T \mathbf{X}(\xi,\eta) = \mathbf{X}^T(\xi,\eta)\mathbf{c}$$

$$\mathbf{T}(\xi,\eta) = \mathbf{X}(\xi,\eta) \times (\mathrm{J}_e(\xi,\eta))^{-1}$$

$$b_e(\phi_h,\varphi_h) = \mathbf{d}^T \left[\iint_{\Omega_0} D_{e0} \mathbf{T}(\xi,\eta) \times \mathbf{T}^T(\xi,\eta) \left| \mathrm{J}_e(\xi,\eta) \right| d\xi d\eta \right] \mathbf{c}$$
$$+ \mathbf{d}^T \left[\iint_{\Omega_0} \Sigma^e_{r0} \mathbf{L}(\xi,\eta) \times \mathbf{L}^T(\xi,\eta) \left| \mathrm{J}_e(\xi,\eta) \right| d\xi d\eta \right] \mathbf{c}$$

$$b_e(\phi_h,\varphi_h) = \mathbf{d}^T \left[\mathbf{S}_{P\times P} + \mathbf{M}_{P\times P} \right]_e \mathbf{c}$$

Note: the nodes are numbered locally from 1 to *p*. $\boldsymbol{M_e}$ is local mass matrix, $\boldsymbol{S_e}$ is local stiffness matrix; they may vary with element because of different coordinate transformations and different material data. **c** is the local vector of degrees of freedom.

$$(s,\varphi_h)_e = \mathbf{d}^T \left[\iint_{\Omega_e} \mathbf{L}(\xi,\eta) \times s_0(\xi,\eta) \left| \mathrm{J}_e(\xi,\eta) \right| d\xi d\eta \right] = \mathbf{d}^T \mathbf{l}$$

$\mathbf{l}_\mathrm{e}$ is the local load vector.

Me and ***Se*** should be evaluated through numerical integration. We use tensor-product formulas here:

$$\begin{aligned}
\mathbf{M}_e &= \iint_{\Omega_0} D_{e0}\mathbf{T}(\xi,\eta)\times\mathbf{T}^T(\xi,\eta)\left|\mathrm{J}_e(\xi,\eta)\right| d\xi d\eta \approx \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} W_{i,j} D_{e0}\mathbf{T}(\xi_i,\eta_j)\times\mathbf{T}^T(\xi_i,\eta_j)\left|\mathrm{J}_e(\xi_i,\eta_j)\right| \\
\mathbf{S}_e &= \iint_{\Omega_e} \Sigma_{r0}^e\mathbf{L}(\xi,\eta)\times\mathbf{L}^T(\xi,\eta)\left|\mathrm{J}_e(\xi,\eta)\right| d\xi d\eta \approx \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} W_{i,j}\Sigma_{r0}^e\mathbf{L}(\xi_i,\eta_j)\times\mathbf{L}^T(\xi_i,\eta_j)\left|\mathrm{J}_e(\xi_i,\eta_j)\right| \qquad (2.31) \\
\mathbf{l} &= \iint_{\Omega_e} \mathbf{L}(\xi,\eta)\times s_0(\xi,\eta)\left|\mathrm{J}_e(\xi,\eta)\right| d\xi d\eta \approx \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} W_{i,j}\mathbf{L}(\xi_i,\eta_j)\times s_0(\xi_i,\eta_j)\left|\mathrm{J}_e(\xi_i,\eta_j)\right|
\end{aligned}$$

n_x, n_y is the number of quadrature points in the x- and y-directions.

Since the integral term of ***Se*** is proportional to $\xi^{2p_v+1}\eta^{2p_h+1}$ in general, we need $n_x=p_v+1$, $n_y=p_h+1$ to obtain its precise value with a Gauss quadrature. Evaluation of the integrals in the Mass matrix is more complex since the determinant of the Jacobian matrix is present in the denominator. The integral term is a polynomial fraction, so we cannot get an exact value. However when the global coordinates satisfy $x_1=x_3$; $x_2=x_4$; $y_1=y_2$; $y_3=y_4$,, the Jacobian matrix is constant and the integral term is proportional to $\xi^{2p_v}\eta^{2p_h}$, and we can still use $n_x=p_v+1$, $n_y=p_h+1$.

2.5.5 Basis functions, connectivity and global assembly

H^1 conformity requires that the basis functions are formed in following way:

1. bubble functions extend to solution domain with zero
2. glue face functions sharing same face in two adjacent elements in 3-D
3. glue edge functions of elements sharing same edge, 4 in 3-D and 2 in 2-D
4. glue vertex shape functions of elements sharing same vertex
5. basis functions is only corresponding to unconstrained nodes and continuous

If there are constraints present because of mesh irregularities (e.g., two adjacent elements which do not share a single whole common edge or a single whole common face due to non uniform *h*-refinement), the basis functions need to be modified. However, in the course of the 1-D work, we will not go into details regarding this topic.

The connectivity array data link the local (master element) DOF numbering with the global

DOF numbering. Different global numbering of basis functions will result into different connectivity data. For example, see the figure below.

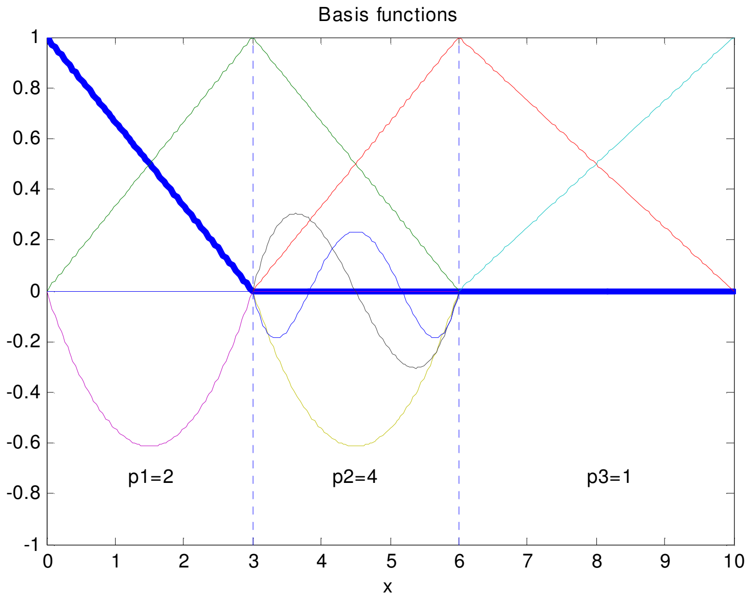


Fig. II-2. An example of FEM domain

There are three elements, whose orders are 2, 4 and 1 respectively in Fig. II-2. There are four basis functions corresponding to four vertices, one bubble function in element 1 and three bubble functions in element 2. Different colors represent different basis functions. We can number these basis functions in following rule:

- From left to right;
- Vertex first

Then we will get the connectivity:

E_1: [1 2 5]

E_2: [2 3 6 7 8]

E_3: [3 4]

However if we choose a different numbering rule:

- from left to right;

- element by element;
- vertex first within element

We will end up with a different data of connectivity:

E$_1$: [1 2 3]

E$_2$: [2 4 5 6 7]

E$_3$: [4 8]

Global numbering only influences the bandwidth of global stiffness matrix.

With the help of connectivity, we are ready to assembly the local matrix and local load vector together in order to obtain the global stiffness matrix and global load vector.

$$\begin{aligned} b(\phi_h,\varphi_h) &= \sum_{e=1}^{N_\Delta} b_e(\phi_h,\varphi_h) = \sum_{e=1}^{N_\Delta} \mathbf{d}_e^T\left[\mathbf{M}_e+\mathbf{S}_e\right]\mathbf{c}_e = \mathbf{d}^T\mathbf{A}\mathbf{c} \\ (s,\varphi_h) &= \sum_{e=1}^{N_\Delta}(s,\varphi_h)_e = \sum_{e=1}^{N_\Delta}\mathbf{d}_e^T\mathbf{l}_e = \mathbf{d}^T\mathbf{b} \end{aligned} \tag{2.32}$$

A is global stiffness matrices. **d** is test vector, **c** is field variable vector. During assembly, we do not have to be aware of the lift. The lift can be treated in applying Dirichlet boundary conditions.

2.5.6 Boundary condition manipulation

Boundary conditions are easily applied by modifying local matrices and load vectors on the fly while assembly the global matrix and load vector.

2.5.6.1 Dirichlet B.C.

Instructions to modify the load vector are: subtract load vector by the linear combination of columns vector and boundary value of the boundary nodes. Then let elements of boundary nodes equal to boundary value. Then modify local matrix **M+S**: let row and column vector of boundary nodes equal to zero, then set the diagonal elements of boundary nodes to 1.

Such modifications maintain the symmetry of the global stiffness matrix. Note that for eigenvalue problems, all Dirichlet B.C. must be homogeneous (zero flux condition).

2.5.6.2 Neumann B.C.

The result of applying the Neumann boundary conditions is the modification of local load vector which involves an integral on faces of element. In 1-D, element is segment, its faces reduce to a zero-dimensional point. We just need to add the corresponding element of local load vector with $J_{-\mathbf{n}}$. Here, we will not further describe the 2-D or 3-D case.

2.5.6.3 Robin B.C.

In this case, we need not only modify the load vector using partial incoming currents but also need modify the local mass matrix of boundary cells.

2.5.6.4 Periodical B.C.

For periodical BC, leakage on the two corresponding boundaries annihilate, so we need not to formulate out the equation of leakage. Boundary conditions can be implemented in following general form,

$$\mathbf{Tc}=\boldsymbol{\alpha} \tag{2.33}$$

$\mathbf{T}$ is a $l \times N$ matrix, $\boldsymbol{\alpha}$ is an l-vector. l is the number of nodes on the boundary. A simple way of treating problems with these boundary conditions is to use Lagrange multipliers.

$$\begin{bmatrix} \mathbf{A} & \mathbf{T}^T \\ \mathbf{T} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \boldsymbol{\alpha} \end{bmatrix} \tag{2.34}$$

This implementation will create zero elements on diagonal. However, it is simple and does not change the symmetry of the global matrix.

2.6 Projection-based interpolation in 1-D

For the coercise case, Cea's lemma implies that the actual error approximation can always be bounded by a mesh independent constant times the best approximation error. Thus, it is sufficient to estimate and control the best approximation error. By definition, the best approximation error is always bounded by the norm of the difference between the exact solution and any particular choice of a function that lives in the FE space. The choice made here, following Demkowicz [31], is the projection-based interpolant of the exact solution $\phi_{hp} = \Pi_{hp}\phi$.

We explain here the basis for the projection-based interpolation and detail some cases that will be needed subsequent when performing *hp*-mesh adaptation.

2.6.1 Definition

Let ϕ be a function defined on the interval [*a*, *b*]. We wish to find an approximation ϕ_{hp} of ϕ in a finite dimensional space V_{hp}, so that the residual $\phi_{hp} - \phi$ is minimum in the sense of semi H^1-seminorm (the H^1 norm or the energy norm are the 'natural' norms for elliptic problems, but thanks to the Poincare inequality, the H^1-seminorm is an equivalent norm).

In order to preserve locality and global continuity in the error estimates, we will require that the interpolation error is minimum and that the interpolant is equal to the exact solution at the vertices. This locality preservation is of paramount important to implement local refinement during the mesh adaptation. The problem is now:

$$\left|\phi_{hp} - \phi\right|_1 \to \min.$$

with $\phi_{hp}(a) = \phi(a)$ and $\phi_{hp}(b) = \phi(b)$.

If the basis functions of space V_{hp} are denoted by e_j, j=0,1,2,…,p, (e_1 and e_2 are equal to 1 at left and right vertex respectively, e_j, for j=2,…,p is equal to zero on two extremity vertices), then solving the problem minimization problem is equivalent to solving

$$\int_a^b (\phi(x) - \phi_{hp}(x))' e_i'(x)dx = 0, \quad i = 2,3,...,p$$

$$\phi_{hp}(a) = \phi(a) \qquad \phi_{hp}(b) = \phi(b)$$

, which means the interpolation error and the basis functions are orthogonal.

Letting $s_i = \int_a^b \phi'(x) e_i'(x)dx, \quad i = 2,3,...,p$

and $a_{i,j} = \int_a^b e_j'(x) e_i'(x)dx, \quad j = 0,1,...,p; \quad i = 2,...,p$

then the matrix form of the problem is given by:

$$\begin{bmatrix} 1 & & 0 & \\ & 1 & & \\ & a_{2,2} & \cdots & a_{2,p} \\ 0 & \vdots & \cdots & \vdots \\ & a_{p,2} & \cdots & a_{p,p} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} \phi(a) \\ \phi(b) \\ s_2 - a_{2,0}\phi(a) - a_{2,1}\phi(b) \\ \vdots \\ s_p - a_{p,0}\phi(a) - a_{p,1}\phi(b) \end{bmatrix}$$

with $\phi_{hp}(x) = \sum_{j=0}^{p} x_j e_j(x)$.

If ϕ smooth enough, we can prove that the projection operation is equivalent to solve a variational boundary value Poisson problem:

$$\begin{aligned} &\frac{d^2\phi_{hp}(x)}{dx^2} = \frac{d^2\phi(x)}{dx^2} \\ &\phi_{hp}(a) = \phi(a); \quad \phi_{hp}(b) = \phi(b); \end{aligned} \tag{2.35}$$

2.6.2 Some specific cases of projection

In the mesh adaptation process, the role of ϕ will be played by a finer numerical solution. In other to compute locally the error and to determine the optimum refinement sequence, projection-based interpolations will be required between the finer numerical solution ϕ and the coarser solution ϕ_{hp}. These projection operations will appear while discussing the *hp*-refinement techniques in next chapter.

2.6.2.1 Case 1: [*p* transformation]

We study here the projection for a given finite element [*a*, *b*] between two solutions defined on [*a*, *b*] as a whole but having different polynomial orders (*p* and *q*). We use hierarchical shape functions as the basis functions. The known (reference) function ϕ belongs to a space spanned by hierarchical shape functions $\chi_j(x)$ of polynomial order up to *q*. The (coarse) numerical solution spans a space of polynomial order up to *p*.

The minimization problem is as follows:

$$\int_a^b (\phi(x)-\phi_h(x))' e_i'(x)dx = 0, \quad i = 2,3,\ldots,p$$

$$\phi_h(x) = \sum_{j=0}^{p} x_j e_j(x)$$

$$\phi(x) = \sum_{j=0}^{q} s_j \chi_j(x)$$

where $e_j(x)$ and $\chi_j(x)$ are the basis functions.

Letting

$$a_{i,j} = \int_a^b e_j'(x)e_i'(x)dx = \frac{2}{b-a}\int_{-1}^{1} l_j'(\xi)l_i'(\xi)d\xi = \frac{2}{b-a}S_{ij}, \quad j = 0,1,\ldots,p; i = 2,3,\ldots,p$$

where $\xi = \frac{2}{b-a}(x - \frac{b+a}{2})$

and

$$b_{i,k} = \int_a^b \chi_k'(x)e_i'(x)dx = \frac{2}{b-a}\int_{-1}^{1} h_j'(\xi)l_i'(\xi)d\xi = X_{ij}, \quad k = 0,1,\ldots,q; i = 2,3,\ldots,p$$

We then can write the minimization problem in matrix form as follows:

$$\begin{bmatrix} 1 & & 0 & & \\ & 1 & & & \\ & & a_{2,2} & \cdots & a_{2,p} \\ 0 & & \vdots & \cdots & \vdots \\ & & a_{p,2} & \cdots & a_{p,p} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} 1 & & & & 0 \\ & 1 & & & \\ b_{2,0}-a_{2,0} & b_{2,1}-a_{2,1} & b_{2,2} & \cdots & b_{2,q} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{p,0}-a_{p,0} & b_{p,1}-a_{p,1} & b_{p,2} & \cdots & b_{p,q} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_q \end{bmatrix}$$

- Obviously if $p=q$ and the shape functions are identical, then the left-hand side matrix is equal to the right-hand side matrix, and thus $x_i=s_i$;
- if $p>q$ (p-refinement) and the shape functions are identical, we can just let $s_j = 0$ for $j=q+1,\ldots,p$ and let two matrix same, we can get solution: $x_i=s_i$, for $i\leq q$, and $x_i=0$ for $i>q$;
- we cannot expect a simple solution when $p<q$ (p-unrefinement), even with identical shape function. However, the Lobatto shape functions being orthogonal in the sense of semi-H^1 norm, we can just let $x_i=s_i$, when $i\leq p$, and just discard the higher order terms.

2.6.2.2 Case 2: [*h*-unrefinement (clustering)]

Consider a coarse element on [*a*, *b*] and its two son-elements [*a*, *c*] and [*c*, *b*]; the reference solution ϕ is a continuous function defined on [*a*, *c*] (with polynomial order p_l) and [*c*, *b*] (with polynomial order p_r). The minimization problem is:

$$\int_a^b (\phi(x)-\phi_{hp}(x))' e_i'(x)dx = 0, \quad i = 2,3,...,p$$

$$\phi(x) = \begin{cases} \sum_{j=0}^{p_l} s_j^l e_j^l(x), & x \in [a,c] \\ \sum_{j=0}^{p_r} s_j^r e_j^r(x), & x \in [c,a] \end{cases}$$

$$s_1^l = s_0^r = s_c$$

and $\phi_{hp}(x) = \sum_{j=0}^{p} x_j e_j(x)$ is the coarse solution defined on [*a*, *b*]. We have, for the left-hand side matrix,

$$a_{i,j} = \frac{2}{b-a} S_{ij}, \quad j = 0,1,...,p; i = 2,3,...,p$$

The right-hand side is more intricate:

$$\int_a^b \phi'(x)e_i'(x)dx = \int_a^c \sum_{j=0}^{p_l} s_j^l e_j'^l(x)e_i'(x)dx + \int_c^b \sum_{j=0}^{p_r} s_j^r e_j'^r(x)e_i'(x)dx, \quad i = 2,3,...,p$$

$$b_{ij}^l = \int_a^c e_j'^l(x)e_i'(x)dx = \frac{2}{b-a}\int_{-1}^1 l_j'(\xi)l_i'(\xi_l)d\xi = \frac{2}{b-a} L_{ij}$$

$$\xi_l = \frac{x - \frac{a+b}{2}}{\frac{b-a}{2}} \qquad \xi = \frac{x - \frac{a+c}{2}}{\frac{c-a}{2}}$$

$$\xi_l = \frac{\frac{c-a}{2}\xi + \frac{a+c}{2} - \frac{a+b}{2}}{\frac{b-a}{2}} = \frac{c-a}{b-a}\xi - \frac{b-c}{b-a}$$

$$b_{ij}^r = \int_c^b l_j'^r(x)e_i'(x)dx = \frac{2}{b-a}\int_{-1}^1 l_j'(\xi)l_i'(\xi_r)d\xi = \frac{2}{b-a} R_{ij}$$

$$\xi_r = \frac{x - \frac{a+b}{2}}{\frac{b-a}{2}} \qquad \xi = \frac{x - \frac{b+c}{2}}{\frac{b-c}{2}}$$

$$\xi_r = \frac{\frac{b-c}{2}\xi + \frac{b+c}{2} - \frac{a+b}{2}}{\frac{b-a}{2}} = \frac{b-c}{b-a}\xi + \frac{c-a}{b-a}$$

Writing down the right-hand side in matrix form yields:

$$\begin{bmatrix} b^l_{20} & b^l_{21} & \cdots & b^l_{2p_l} \\ \vdots & \vdots & \vdots & \vdots \\ b^l_{p0} & b^l_{p1} & \cdots & b^l_{pp_l} \end{bmatrix} \begin{bmatrix} s^l_0 \\ s^l_1 \\ \vdots \\ s^l_{p_l} \end{bmatrix} + \begin{bmatrix} b^r_{20} & b^r_{21} & \cdots & b^r_{2p_r} \\ \vdots & \vdots & \vdots & \vdots \\ b^r_{p0} & b^r_{p1} & \cdots & b^r_{pp_r} \end{bmatrix} \begin{bmatrix} s^r_0 \\ s^r_1 \\ \vdots \\ s^r_{p_l} \end{bmatrix}$$

Finally, we obtain the linear system:

$$\begin{bmatrix} 1 & & 0 & & \\ & 1 & & & \\ & & a_{2,2} & \cdots & a_{2,p} \\ 0 & & \vdots & \cdots & \vdots \\ & & a_{p,2} & \cdots & a_{p,p} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} 1 & & & & & 0 & & & \\ & 1 & & & & & & & \\ b^l_{20} - a_{2,0} & b^r_{21} - a_{2,1} & b^l_{21} + b^r_{20} & b^l_{22} & \cdots & b^l_{2p_l} & b^r_{22} & \cdots & b^r_{2p_r} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b^l_{p0} - a_{p,0} & b^r_{p1} - a_{p,1} & b^l_{p1} + b^r_{p1} & b^l_{p2} & \cdots & b^l_{pp_l} & b^r_{p2} & \cdots & b^r_{pp_r} \end{bmatrix} \begin{bmatrix} s^l_0 \\ s^r_1 \\ s_c \\ s^l_2 \\ \vdots \\ s^l_{p_l} \\ s^r_2 \\ \vdots \\ s^r_{p_r} \end{bmatrix}$$

As an example of clustering, we propose for the reference solution: $p_{left} = p_{right} = 1$; $s_0^{left} = 0; s_c = 2; s_1^{right} = 1$. Projection-based interpolation on [*a*, *c*] with different polynomial orders *p* and with different norms are shown in Fig. II-3.

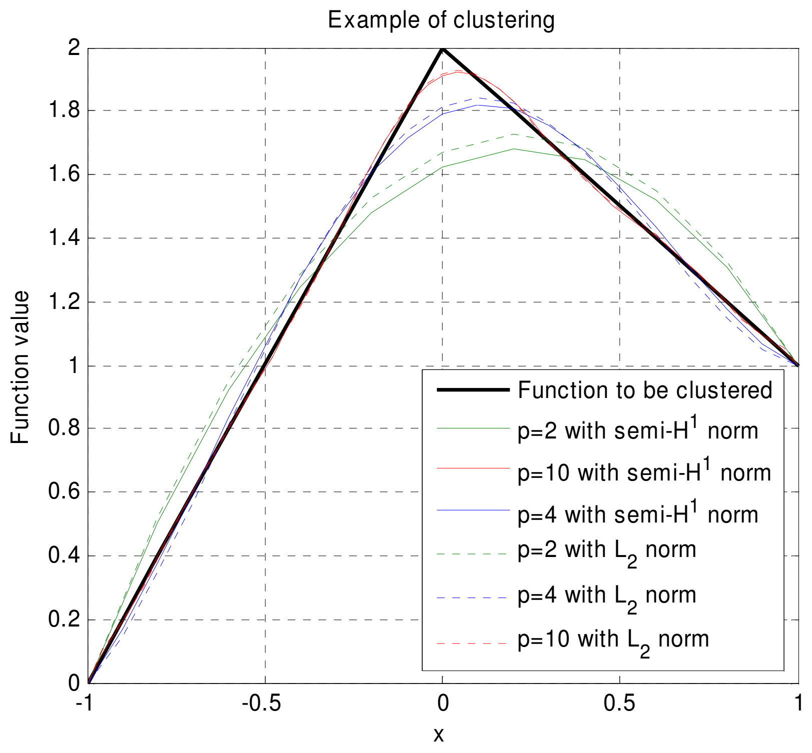


Fig. II-3. Projection-based interpolation in the case of clustering

The higher the polynomial order p is, the smaller difference between ϕ and ϕ_{hp} is. Note that the H^1 semi-norm and the L_2 norm give different results.

2.6.2.3 Case 3: [*h*-refinement (split)]

Consider two son-elements [a, c] and [c, b]; the reference solution ϕ belongs to a space spanned with shape functions on [a, b] on a whole. The coarse solution lives on [a, c] and [c, b].

After similar derivation, we obtain:

$$\int_a^b \phi'_{hp}(x) e_i'^l(x)dx = \int_a^c \phi'_{hp}(x) e_i'^l(x)dx$$

$$\int_a^b \phi'(x) e_i'(x)dx = \int_a^c \sum_{j=0}^{p_l} s_j^l e_j'^l(x) e_i'(x)dx + \int_c^b \sum_{j=0}^{p_r} s_j^r e_j'^r(x) e_i'(x)dx, \quad i = 2,3,...,p$$

$$\phi_{hp}(x) = \begin{cases} \sum_{j=0}^{p_l} x_j^l e_j^l(x), & x \in [a,c] \\ \sum_{j=0}^{p_r} x_j^r e_j^r(x), & x \in [c,b] \end{cases}$$

$$x_1^l = x_0^r = x_c$$

$$\int_a^b \phi'_{hp}(x) e_i'(x)dx = \int_a^c e_{i-2}^l(x) \sum_{j=0}^{p_l} x_j^l e_j^l(x) = \sum_{j=0}^{p_l} x_j^l \int_a^c e_{i-2}^l(x) e_j^l(x)dx$$

$$= \sum_{j=0}^{p_l} x_j^l \frac{2}{c-a} S_{i-2,j} = \sum_{j=0}^{p_l} x_j^l a_{i-2,j}^l, \qquad i = 4,5,..., p_l + 2$$

$$\int_a^b \phi'_{hp}(x) e_i'(x)dx = \int_c^b e_{i-p_l-1}^r(x) \sum_{j=0}^{p_r} x_j^r e_j^r(x) = \sum_{j=0}^{p_r} x_j^r \int_c^b e_{i-p_l-1}^r(x) e_j^r(x)dx$$

$$= \sum_{j=0}^{p_r} x_j^r \frac{2}{b-c} S_{i-p_l-1,j} = \sum_{j=0}^{p_r} x_j^r a_{i-p_l-1,j}^r, \qquad i = p_l + 3, p_l + 4,..., p_l + p_r + 1$$

$$\int_a^b \phi'_{hp}(x) e_3'(x)dx = \int_a^c e_1^l(x) \sum_{j=0}^{p_l} x_j^l e_j^l(x) + \int_c^b e_0^r(x) \sum_{j=0}^{p_r} x_j^r e_j^r(x)$$

$$= \sum_{j=0}^{p_l} x_j^l \frac{2}{c-a} S_{1,j} + \sum_{j=0}^{p_r} x_j^r \frac{2}{b-c} S_{0,j} = \sum_{j=0}^{p_l} x_j^l a_{1,j}^l + \sum_{j=0}^{p_r} x_j^r a_{0,j}^r$$

$$= x_0^l a_{1,0}^l + x_c (a_{1,1}^l + a_{0,0}^r) + x_1^r a_{0,1}^r + \sum_{j=2}^{p_l} x_j^l a_{1,j}^l + \sum_{j=2}^{p_r} x_j^r a_{0,j}^r$$

Simply, the left-hand side of equation, in matrix form, is:

$$\begin{bmatrix} 1 & & & & 0 & & & & \\ & 1 & & & & & & & \\ a_{1,0}^l & a_{0,1}^r & a_{1,1}^l + a_{0,0}^r & a_{1,2}^l & \cdots & a_{1,p_l}^l & a_{0,2}^r & \cdots & a_{0,p_r}^r \\ a_{2,0}^l & & a_{2,1}^l & a_{2,2}^l & \cdots & a_{2,p_l}^l & & & \\ \vdots & 0 & \vdots & \vdots & \vdots & \vdots & & 0 & \\ a_{p_l,0}^l & & a_{p_l,1}^l & a_{p_l,2}^l & \cdots & a_{p_l,p_l}^l & & & \\ & a_{2,1}^r & a_{2,0}^r & & & & a_{2,2}^r & \cdots & a_{2,p_r}^r \\ 0 & \vdots & \vdots & & 0 & & \vdots & \vdots & \vdots \\ & a_{p_r,1}^r & a_{p_r,0}^r & & & & a_{p_r,2}^r & \cdots & a_{p_l,p_r}^r \end{bmatrix} \begin{bmatrix} x_0^l \\ x_1^r \\ x_c \\ x_2^l \\ \vdots \\ x_{p_l}^l \\ x_2^r \\ \vdots \\ x_{p_r}^r \end{bmatrix}$$

It is exactly an assembly procedure for two finite elements.

The right-hand side is:

$$\phi(x)=\sum_{j=0}^{p}s_j h_j(x),\quad x\in[a,b]$$

$$\int_a^b \phi'(x)e_i'(x)dx=\int_a^c e_{i-2}'^{l}(x)\sum_{j=0}^{p}s_j h_j'(x)=\sum_{j=0}^{p}s_j\int_a^c e_{i-2}^{l}(x)h_j'(x)dx$$

$$=\sum_{j=0}^{p}s_j\frac{2}{b-a}\int_{-1}^{1}l_{i-2}'(\xi)l_j'(\xi_l)dx=\sum_{j=0}^{p}s_j\frac{2}{b-a}L_{j(i-2)}=\sum_{j=0}^{p}s_j b_{i-2,j}^{l},\qquad i=4,5,\ldots,p_l+2$$

$$\int_a^b \phi'(x)e_i'(x)dx=\int_c^b e_{i-p_l-1}^{r}(x)\sum_{j=0}^{p}s_j h_j'(x)=\sum_{j=0}^{p_r}s_j\int_c^b e_{i-p_l-1}^{r}(x)h_j'(x)dx$$

$$=\sum_{j=0}^{p}s_j\frac{2}{b-a}R_{j(i-p_l-1)}=\sum_{j=0}^{p}s_j b_{i-p_l-1,j}^{r},\qquad i=p_l+3,p_l+4,\ldots,p_l+p_r+1$$

$$\int_a^b \phi'(x)e_3'(x)dx=\int_a^c e_1^{l}(x)\sum_{j=0}^{p}s_j h_j'(x)+\int_c^b e_0^{r}(x)\sum_{j=0}^{p}s_j h_j'(x)$$

$$=\sum_{j=0}^{p}s_j b_{1,j}^{l}+\sum_{j=0}^{p}s_j b_{0,j}^{r}$$

Or in matrix form:

$$\begin{bmatrix} 1 & & & 0 & \\ & 1 & & & \\ b_{1,0}^{l}+b_{0,0}^{r} & b_{1,1}^{l}+b_{0,1}^{r} & b_{1,2}^{l}+b_{0,2}^{r} & \cdots & b_{1,p}^{l}+b_{0,p}^{r} \\ b_{2,0}^{l} & b_{2,1}^{l} & b_{2,2}^{l} & \cdots & b_{2,p}^{l} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ b_{p_l,0}^{l} & b_{p_l,1}^{l} & b_{p_l,2}^{l} & \cdots & b_{p_l,p}^{l} \\ b_{2,0}^{r} & b_{2,1}^{r} & b_{2,2}^{r} & \cdots & b_{2,p}^{r} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ b_{p_l,0}^{r} & b_{p_l,1}^{r} & b_{p_l,2}^{r} & \cdots & b_{p_l,p}^{r} \end{bmatrix}\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_p \end{bmatrix}$$

An example of projection-based interpolation in the case of splitting is shown with p=2; $s_0=s_1=0; s_2=-2$; (Lobatto shape functions are used here). Projections with different polynomials orders on the left and right elements (p_l and p_r) and with different norms are shown in Fig. II-4.

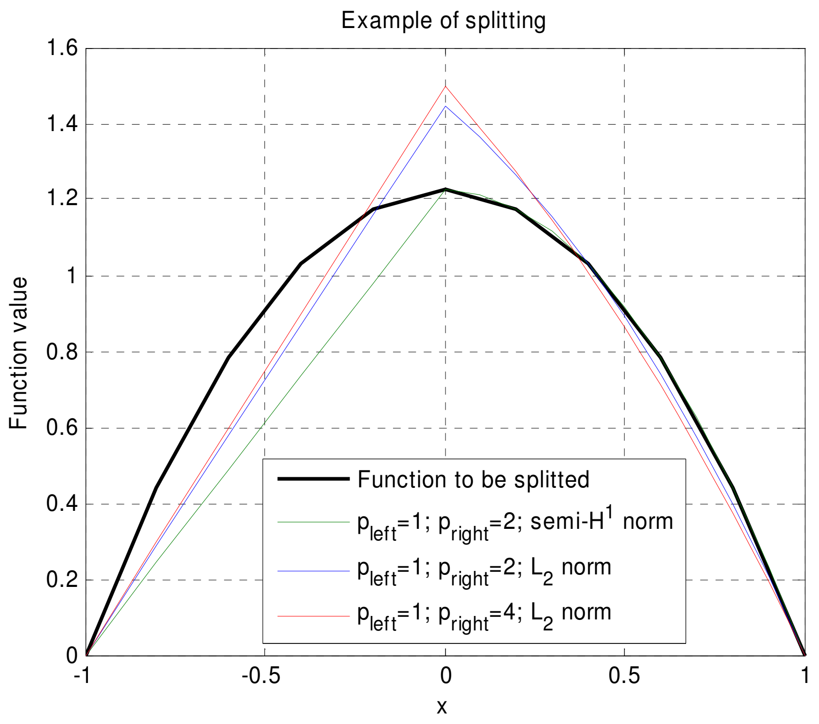


Fig. II-4. Projection-based interpolation in the case of splitting

It seems projection with H^1 semi-norm always deliver $\phi_{hp}=\phi$ at the center vertex though we did not try to prove it. Again, the higher polynomial order p_l or p_r is, the smaller difference between ϕ and ϕ_{hp}.

2.6.3 Projection of a solution from a mesh to another

Combining these projection operations, we can project a solution from one mesh to another mesh as illustrated in Fig. II-5. The initial mesh was composed of 3 elements. At some point during the computation, we have mesh 1 and we wish to unrefine its first three elements and to refine the other two elements.

In this example, we need two steps to pass from mesh 1 to mesh 2: in the first step, we cluster two son-elements of the right son-element of the first initial element; then, in a second step, we perform one cluster operation on the first element and two splits on the last two initial elements.

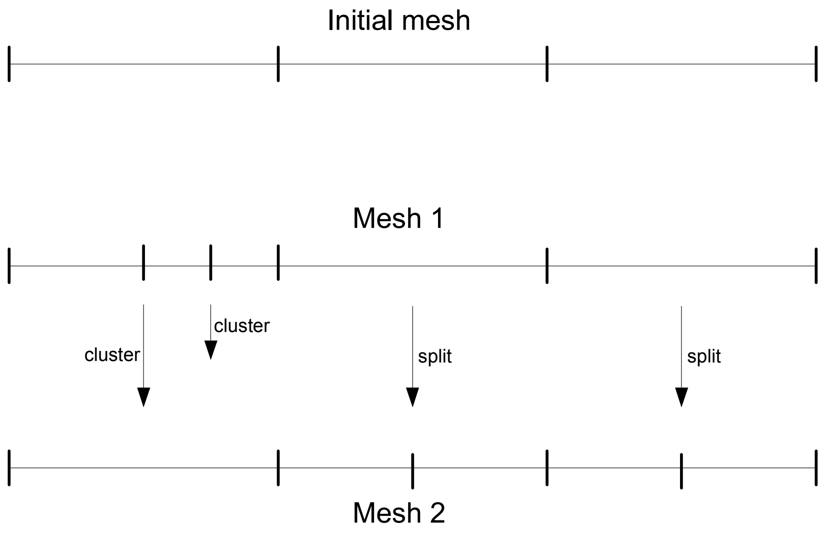


Fig. II-5. An example of mesh projection

Projection-based interpolation can provide us with a simple way to manipulate multi-group coupling source terms (i.e., fluxes) when using different meshes for different energy groups. We provide more details in Chapter IV regarding group coupling and will also propose another method to perform the global matrix assembly group by group directly using adaptive integration.

2.7 A 1-D FEM neutron diffusion code in MATLAB

A 1-D FEM multi-group neutron diffusion code with MATLAB is completed early during the research in order to demonstrate some basic ideas of FEM in a very convenient way. The multi-group diffusion problem is described with more details in Chapter IV. The MATLAB toy-code is available at http://nuclear.tamu.edu/~yaqiw. We present here some examples will be subsequently utilized in the one-group and multigroup *hp*-adaptation in Chapters II and III.

2.7.1 Features and limitations of the code

Some of the features of the toy-code are as follows:

1. Finite Element Method Coded in MATLAB

2. Supports multi-group neutron diffusion problem
3. Supports both source and eigenvalue problem
4. Supports Dirichlet, Neumann, Cauchy and periodical boundary conditions
5. Different energy groups can have different number of finite element (using projection operations)
6. FE polynomial order up to 20
7. Supports 4 different types of shape functions (this is extendable)
8. Supports 3 types of quadrature: Newton-Cotes, Gauss-Legendre and Gauss-Lobatto
9. Quadrature order can be adjusted independently
10. Supports different choices of global numbering

Limitations are:

1. One-dimension
2. All finite elements must have same polynomial order
3. Different energy groups may have different numbers of finite element but they must be 2^n multiple of each other
4. Piecewise constant volumetric source only
5. Unique fission spectrum
6. No up-scattering allowed (extendable)

2.7.2 Some demonstration results

We present three sets of results:

1. first, some considerations regarding the relation between the choice of shape functions and the condition number,
2. secondly, we present a one-group source problem,
3. and finally, we show a multigroup eigenproblem.

2.7.2.1 Condition number of local and global matrices

The approximation functional space V_{hp} in Eq. (2.17) is determined by mesh, which includes the information of domain triangulation and distribution of polynomial order in all cells or finite elements. The functional space is independent on the choice of type of shape functions because all kinds of shape functions or same order span the same polynomial space. However, the choice of shape functions will influence the condition number of local matrices $\mathbf{M_e}$, $\mathbf{S_e}$ and, hence, the condition number of the global matrix **A**. Some choices of shape functions may lead to ill-conditioned matrices (high condition number), thus the numerical solution may become quite sensitive to round errors (which are always present in numerical analysis).

We used the code to generate the global matrix **A** corresponding the operator

$$-D\frac{d^2}{dx^2}+\Sigma_r\,,$$

where D=0.4cm and Σ_r =0.1cm^{-1} are constant throughout a 400cm domain. Boundary conditions of the diffusion operator are homogeneous Dirichlet both on the left and on the right.

Before considering global matrix **A**, let us analyze the local (i.e., elemental) matrices because they are independent on a specific diffusion operator. The condition number of partial local stiffness matrix and local mass matrix with different polynomial order is illustrated in Fig. II-6. Because local stiffness matrix is singular, we only consider its sub-matrix formed with bubble functions (i.e., all shape functions except the first two linear shape functions corresponding to two vertex degrees of freedom). Based on condition number, Lobatto shape functions perform best among the four shape functions tested in 1-D. Note that the condition number for the modified Peano shape functions is much larger than the one for the Peano shape functions.

Now, we calculate condition numbers of the resulting global matrix **A** with different numbers of cells and different polynomial order (all elements have the same order). The plots of condition numbers of different types of shape functions are given in Fig. II-7.

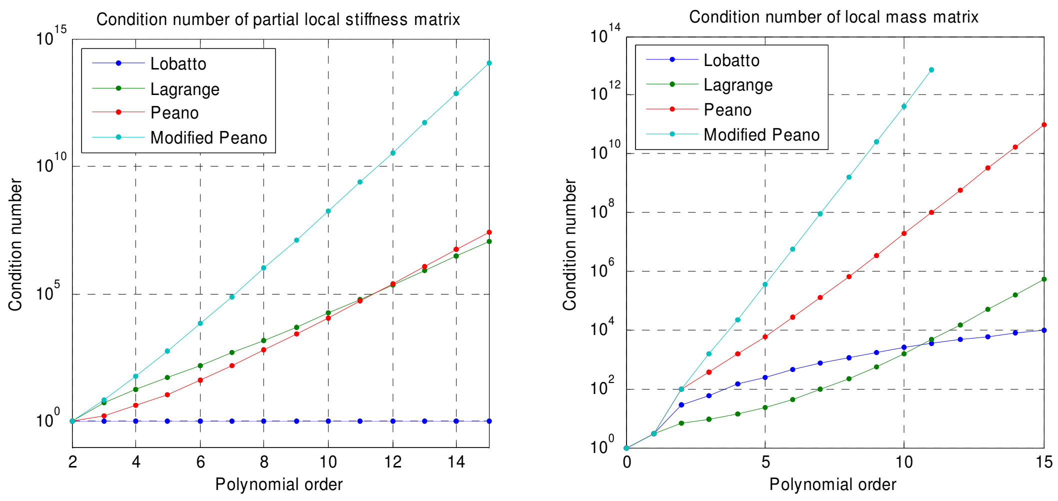


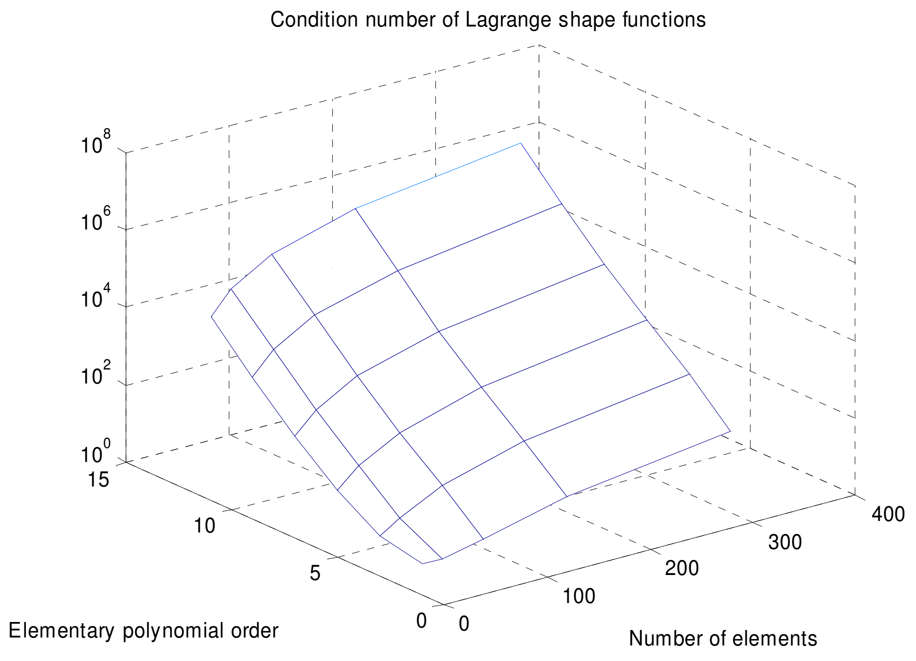
Fig. II-6. Condition number of local partial stiffness matrix and local mass matrix

Generally speaking, the condition number of global matrix is mainly determined by local stiffness matrix. Local stiffness matrices with larger condition numbers will produce larger condition number globally. The condition numbers vary little with number of elements but increase significantly with increasing polynomial order. Therefore, some shape functions should not be utilized with a higher order finite element method.

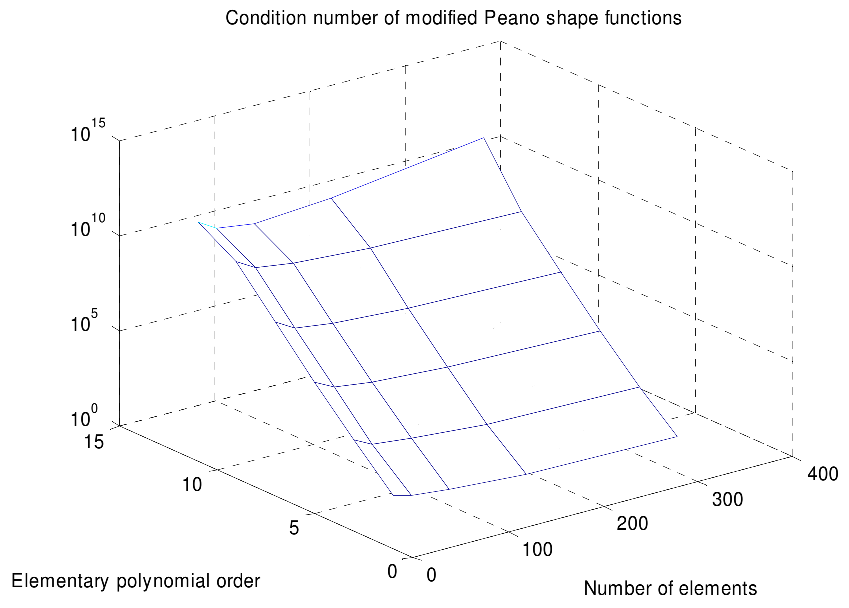
2.7.2.2 A simple one-group source problem

Three different materials are placed in 7 seven different regions as follows 1-2-3-2-3-3-2 (the number represent the material number). Each region is 100-cm thick. Zero-flux boundaries hold. For each material, diffusion coefficients are 0.333, 0.370 and 0.303 cm respectively; absorption cross sections are 0.02, 0.1, and 0.3 cm^{-1}; volumetric neutron source terms are 0.0, 1.5 and 1.8 n/cm3-sec.

Fig. II-8. are flux from calculations with same polynomial order 1 and different number of elements per assembly 8, 16, 32 and 64. We can see even with 32 elements there are still spikes in the curve.

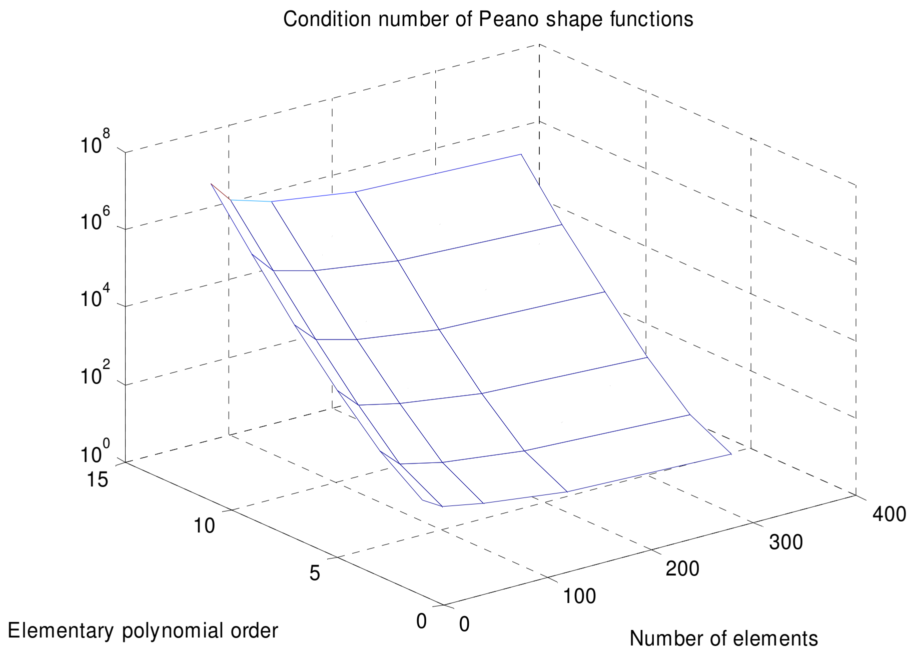


(a)

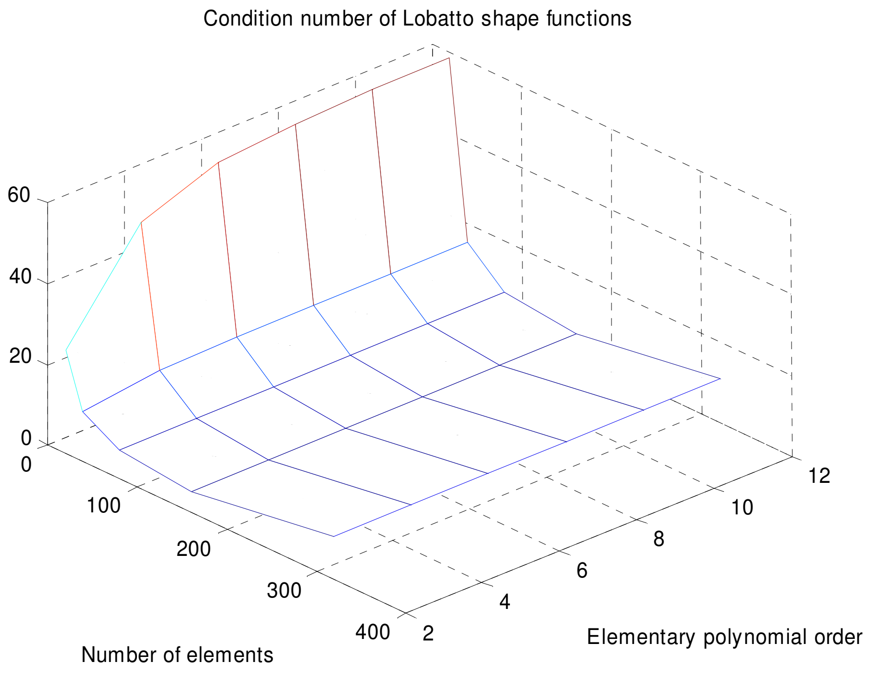


(b)

Fig. II-7. Condition number of global stiffness matrix: (a) Lagrange polynomials, (b) modified Peano polynomials



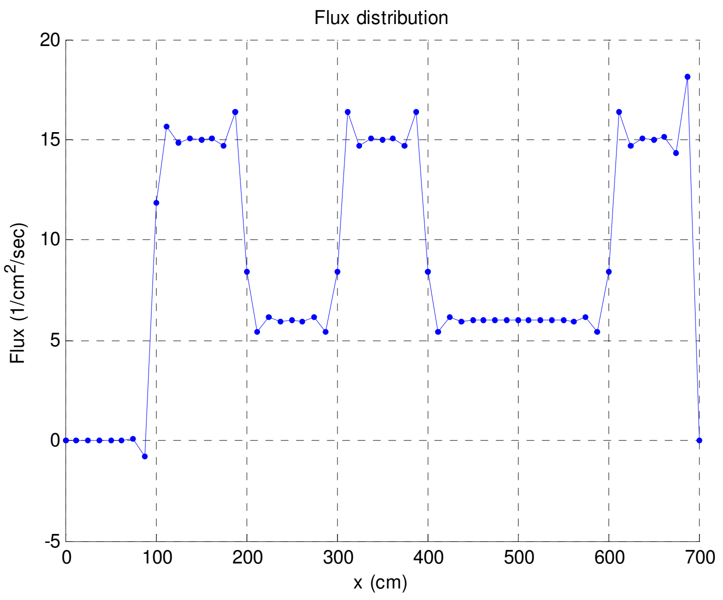
(c)



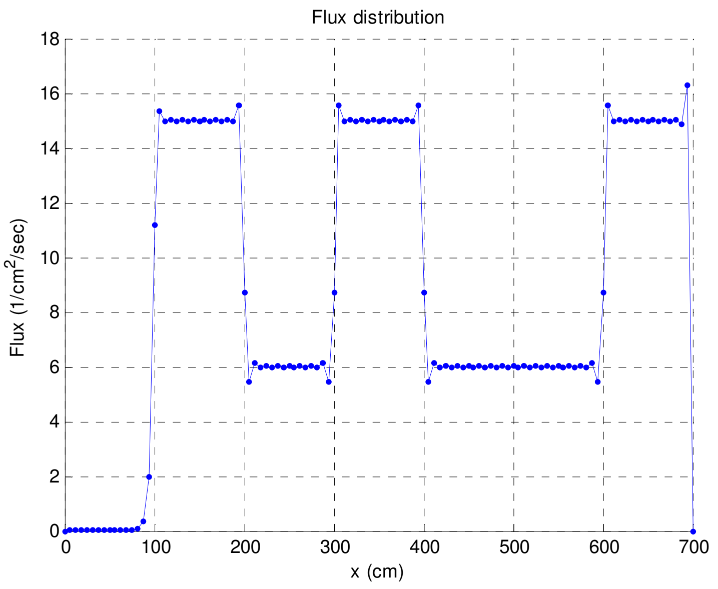
(d)

Fig. II-7. (Continued), (c) Peano polynomials and (d) Lobatto polynomials

We calculated with polynomial order 2, then got four graphs with different elements per assembly, 8, 16, 32 and 512 in Fig. II-9. The last one can act as a reference. We can see we can get better solutions with higher polynomial order with same global degrees of freedom.

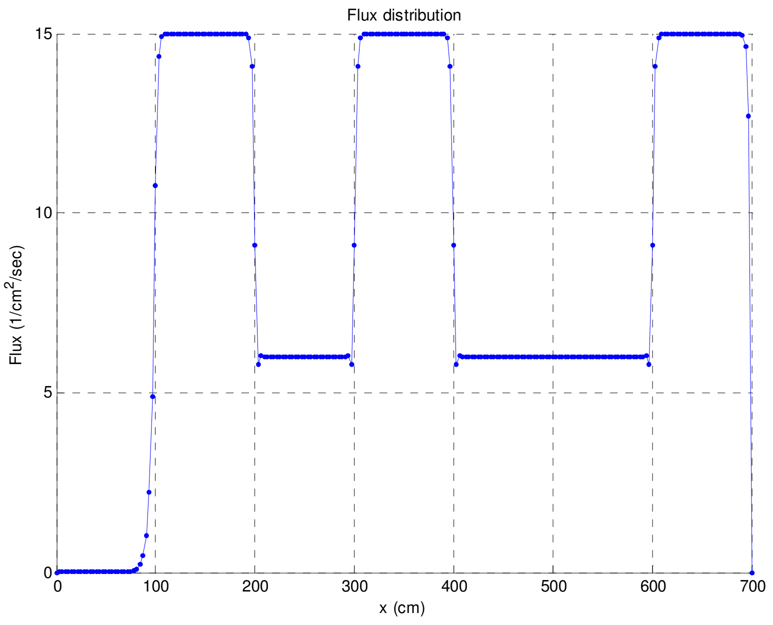


(a)

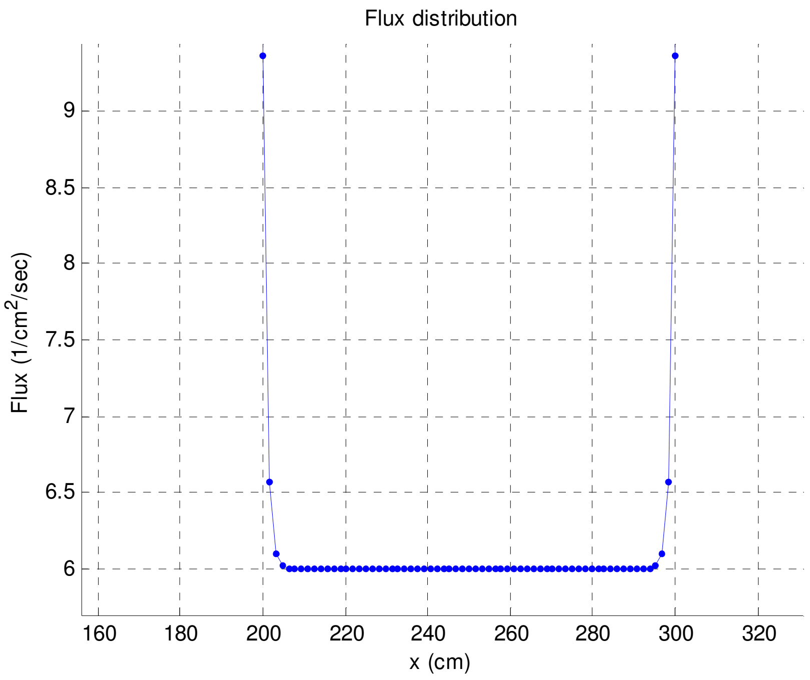


(b)

Fig. II-8. Flux distribution with *p*=1: (a) 8 elements, (b) 16 elements

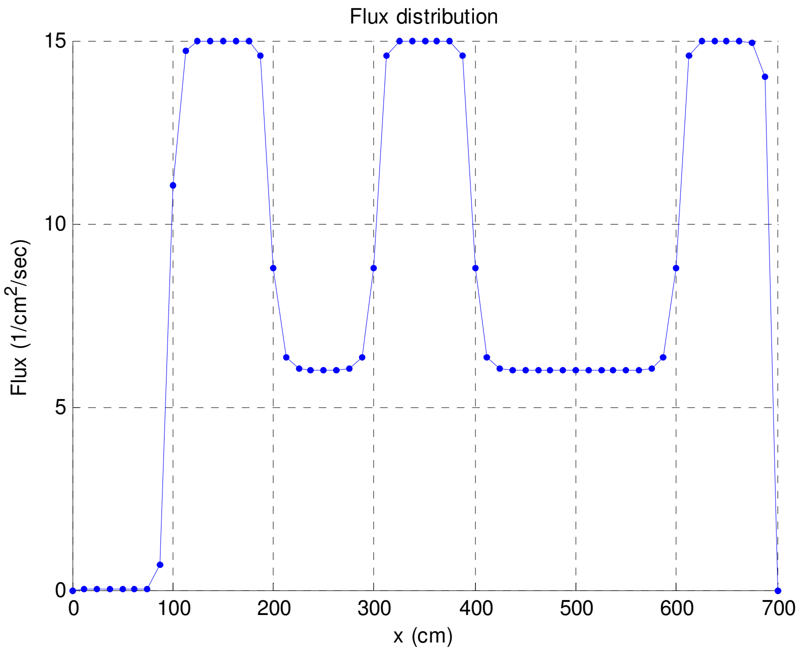


(c)

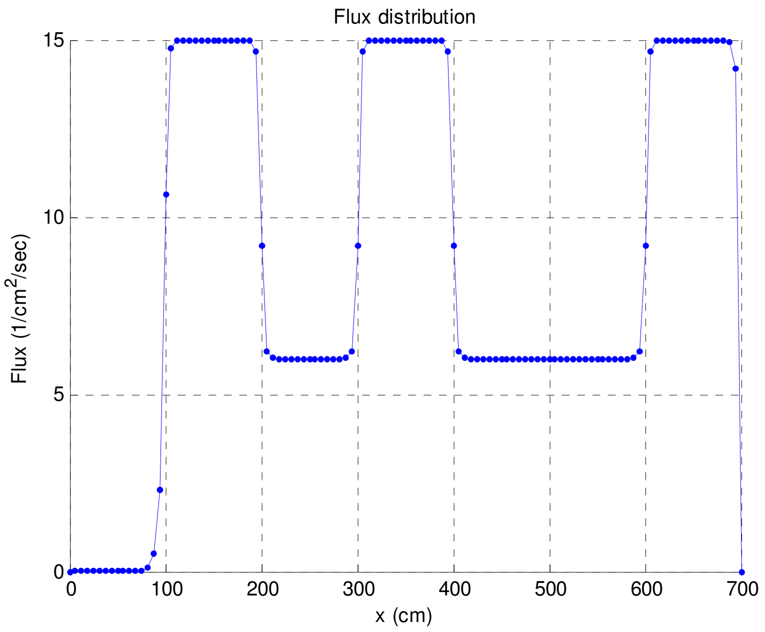


(d)

Fig. II-8. (Continued), (c) 32 elements and (d) 64 elements

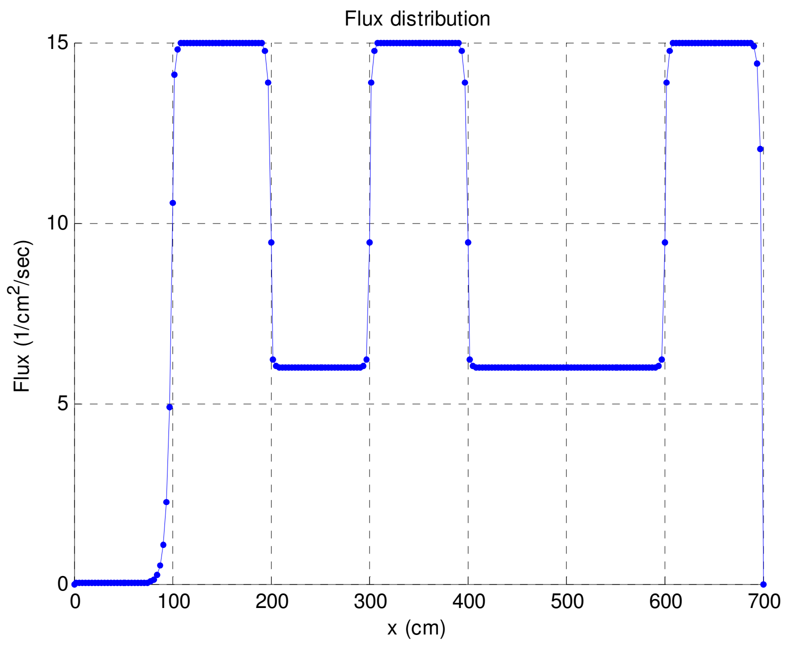


(a)

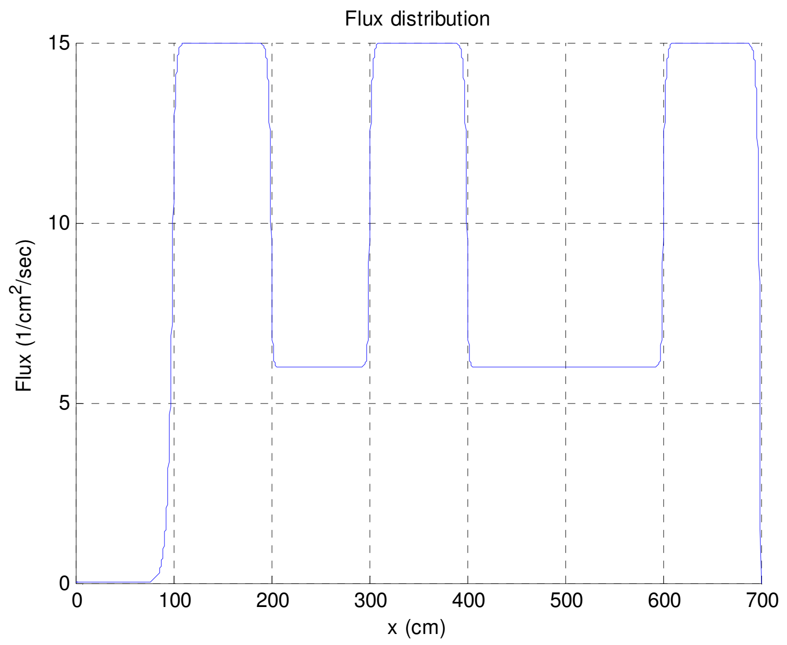


(b)

Fig. II-9. Flux distribution with *p*=2: (a) 16 elements, (b) 16 elements



(c)



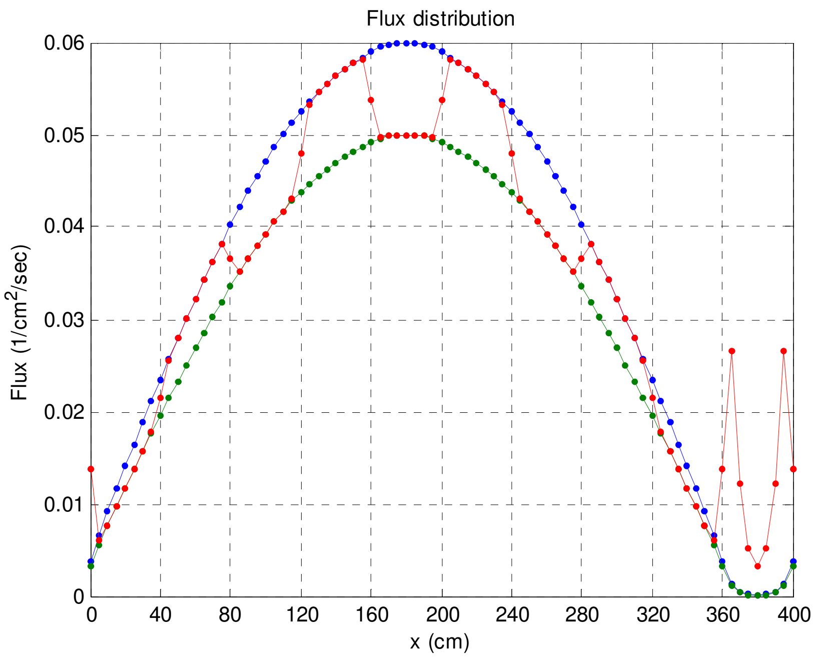
(d)

Fig. II-9. (Continued), (c) 32 elements and (d) 512 elements

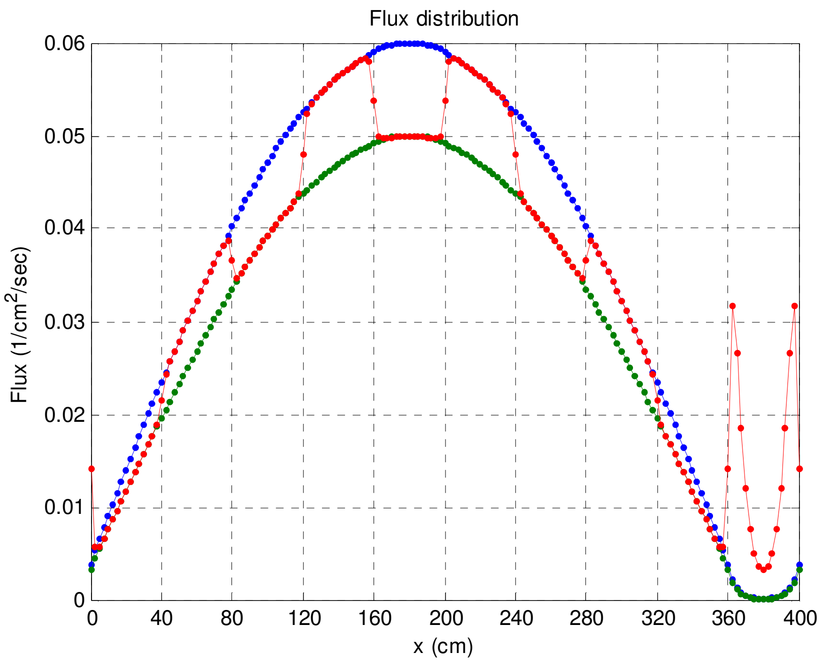
2.7.2.3 A 2-group eigenvalue problem

Again three different materials are placed in 10 regions as 1-2-1-2-1-2-1-2-1-3 (1 and 2 are fissile material, 3 is water acting as a reflector), each region is 40-cm thick. Periodic boundary conditions are applied. All materials have same fast diffusion coefficients 1.2cm, and material 1 and 2 have same thermal diffusion coefficients 0.4 cm, thermal diffusion coefficient of material 3 is 0.2cm; fast removal cross sections of fissile materials are 0.03 cm^{-1}, but they have different thermal removal (absorption) cross section 0.3 cm^{-1} and 0.25 cm^{-1}; Fast and thermal removal cross sections of reflector are 0.051 and 0.04 respectively; There is no up-scattering, and down-scattering cross section for fissile materials is 0.015 cm^{-1} and 0.05 cm^{-1} for reflector; Fast fission cross section times average number of neutron released per fission is 0.0075cm^{-1} for both fissile materials, but they have different thermal fission indicated as 0.045cm^{-1} and 0.0375cm^{-1} respectively; all neutron are born in fast.

Convergence is controlled by the error of k_{eff} between two successive power iterations being less than 10^{-10} and maximum power iteration number being less than 2000. Polynomial order 2 is applied. Fluxes are normalized with respect to the fast flux peak. Flux distributions with number of element per assembly being from 8 to 64 both of fast group and thermal group are in Fig. II-10. The red curves in the figures represent the thermal flux distribution. The blue and green curves are the fast flux plotted with different renormalization factors.

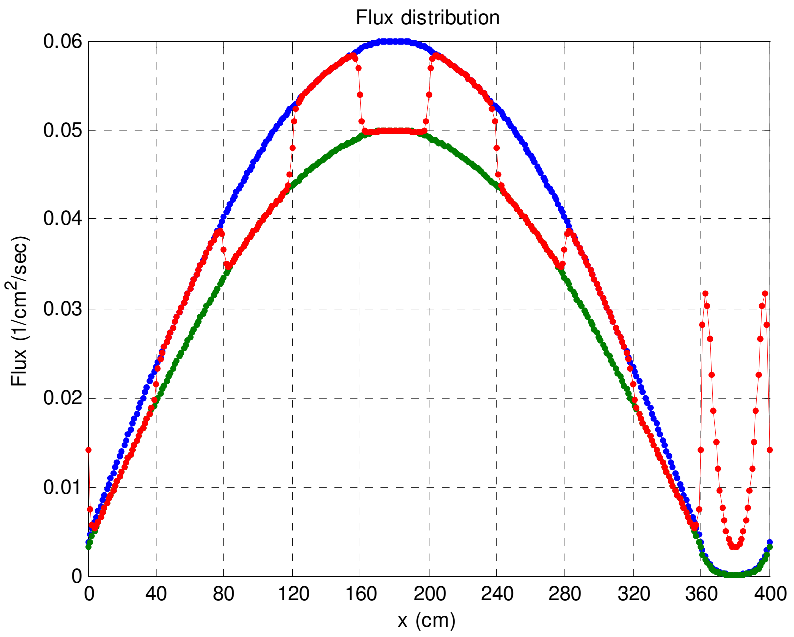


(a)

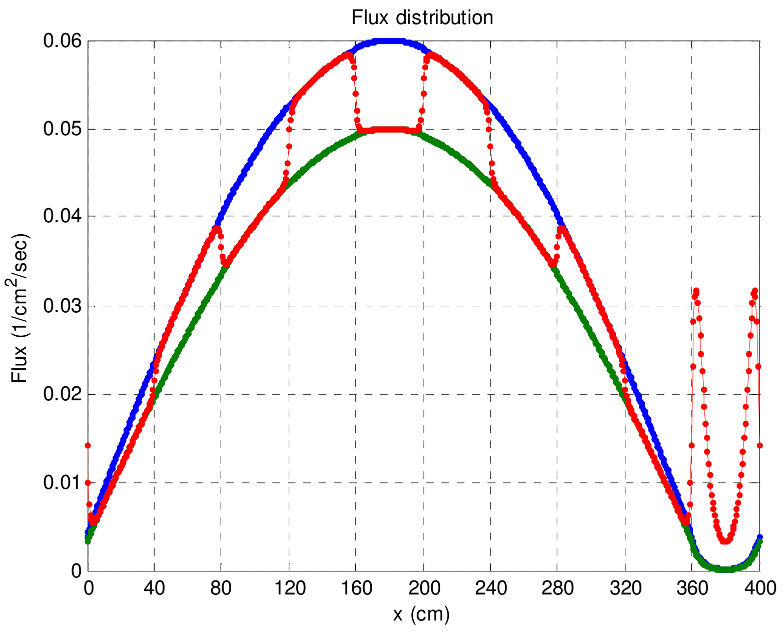


(b)

Fig. II-10. Flux distributions for a sample eigenvalue problem: (a) 8 elements, (b) 16 elements



(c)



(d)

Fig. II-10. (Continued), (c) 32 elements and (d) 64 elements

2.8 Conclusions of chapter II

We first presented some basics about variational boundary value problems and Galerkin method. Then we described some understandings of FEM. As a result, a 1-d MATLAB code was finished. All of these will give author the FEM background to pursue deeper topic of *hp*-adaptive of FEM. The 1-D MATLAB code has the capabilities to demonstrate FEM in the context of multi-group neutron diffusion problem. Besides these, some calculations with the code showed that non-uniform refinement, higher order FEM and different meshes for different energy groups are worthy to be considered in neutron diffusion. Some aspects about applying FEM were addressed for example, choice of shape function basing on condition number, projection of flux from one mesh to another different, influence on the spectrum of global matrix due to scattering term, etc.

CHAPTER III

ADAPTIVE HP-REFINEMENT TECHNIQUES

In this chapter, we present mesh refinement techniques applied to the one-group diffusion equation with a fixed source. The issues related to the coupling between groups or the interaction of the mesh adaptation procedure with the eigenvalue problem will be analyzed in Chapter IV. In essence, the chapter introduces the basic concepts related to mesh adaptation in the FEM setting for an elliptic PDE. We will first motivate the need for mesh adaptation, then present the principle of mesh adaptation based on a posteriori error estimation. Next, we discuss the various types of refinement available in FEM (either mesh subdivision or polynomial order increase). We then present an open-source 1-D *hp*-FEM code developed by Dr. Demkowicz (University of Texas, Austin) [34]; this code will serve as the basis for our development. Notably, we propose a new error estimator which possesses an equivalent reliability as the one actually present in the code but whose CPU cost is reduced. Some other minor enhancements to the code will also be discussed. Finally, we conclude by providing two 1-D one-group fixed source diffusion examples of computation.

3.1 Introduction: Motivations for mesh refinement and *hp*-adaptation

The presence of numerical error is intrinsic to computer simulation of physical phenomena. A remedy often used to circumvent possibly unacceptable discretization errors is to re-compute the problem using a finer mesh. Most of the times, the finer mesh is obtained by uniform refinement of the previous mesh. This process soon comes to a halt due to the enormous increase in the number of unknowns. Furthermore, uniform mesh refinement does not guarantee that a solution has been reached within a prescribed tolerance.

Another conceivable option would be to discretize the mesh based on some knowledge of the solution behavior: acquiring such knowledge is based on trail-and-error attempts and is obviously (a) time-consuming (the user has to iteratively design the mesh), (b) of limited validity (the user may not repeat this laborious process each time a change in material compositions or geometry occurs but may use some intuition regarding whether a given mesh can be utilized or not; unfortunately such a ‘domain of validity’ is often subjective based on the user’s experience), (c) prone to error (we rely on the user to verify that a solution has converged).

Yet another option to obtain a converged solution could consist in meshing the domain according to some physical length, characteristic of the problem: e.g., the diffusion length $\sqrt{D/\Sigma_a}$ in diffusion theory or the mean-free-path $1/\Sigma_t$ in transport theory. Such an approach seems reasonable to deliver acceptable results in terms of accuracy, but it is very far from being optimal in terms of number of unknowns: it does not account for the possible smoothness in the solution and will therefore arbitrarily and excessively over-refine the mesh in regions where it is unnecessary; we can give an simple example where such a mesh based on the problem characteristic length is ill-thought: consider an infinite domain containing an uniform source, the domain is composed of two different half-infinite media whose dimensions are much larger than the characteristic length of the problem: at the material interface, a sharp flux gradient may occur and fine meshes of the order of the characteristic length are needed to represent the flux accurately but away from for the interface, the flux will reach the spatially independent asymptotic solution S/Σ_a where very few meshes (one mesh) are required.

The need to reach a guaranteed convergence depending upon a prescribed user-defined tolerance with a more suitable usage of resources (CPU and memory) is, therefore, advised and desired. In other words, with a small computational budget, the effort (i.e., the meshes) should be put where needed. This can only be achieved with the knowledge of the local error and such knowledge will also permit to converge the solution to a user-specified tolerance. Simply put, we wish to automatically adapt the mesh to a user-prescribed tolerance, i.e., we wish to attain a guaranteed accuracy with a sensible usage of resources in a user-independent fashion. It is, therefore, most desirable to devise algorithms that can assess the local errors and adaptively refine the mesh only in areas where it is needed.

A priori error estimates, usually of the form $\left\|u-u_{hp}\right\|_1 \leq Ch^k \left\|u\right\|_{k+1}$, are of little use for mesh adaptation purposes because (1) they are global quantities over the entire solution domain whereas we wish to reduce the error locally (element by element), and (2) the constant C and the norm $\left\|u\right\|_{k+1}$ embedded in these a priori estimates are virtually impossible to obtain for real-life cases.

In the last decade, the theory of *a posteriori* error estimations [19] [36] [37] has matured and allows the measure, control and minimization of approximation errors. In this theory, the computed solution itself is used to provide inexpensively point-wise error estimations. In the framework of finite element methods, there are several factors on which one can play to reduce the error in a cell chosen for refinement: (1) the cell can be subdivided in smaller cells, *h*-method, or (2) the polynomial order representation for the numerical solution of that cell can be increased, *p*-method. While both of these options perform better than uniform mesh refinement, neither is independently optimal.

1. While *h*-refinement is indicated for regions where the solution is not smooth, such as domain corners or zones with significant material property discontinuities, it does not deliver the best convergence rate for regions where the solution is smooth.
2. On the other hand, *p*-refinement is ideal for zones with a smooth solution but it should not be applied in regions where the solution is irregular, as near boundaries or material interfaces.

However, it is possible to combine the advantages of both methods into what is commonly termed the *hp*-refinement technique where the choice between a mesh subdivision and an increase in the polynomial order is based on a competitive minimization of local errors. This refinement option augments somewhat the complexity of the computer code but an appropriate choice of basis functions can greatly reduce this complexity (this will be the case when using bubble functions which collocated all higher-order DOF at the same node, in 1-D this node is the element middle). Such *hp-methods* have been proven to be quasi-optimal and to *deliver exponential convergence* [24].

In this chapter, we present some crucial points regarding the *hp*-version applied to the one-group 1-D fixed source neutron diffusion problem (a simple elliptic PDE); Chapter IV will be devoted to the multigroup fixed-source problem and eigenproblem cases. The results presented here will demonstrate both the guaranteed convergence and the efficient implementation of *hp*-FEM.

The starting point of our work is an open source 1D-*hp* code from Dr. Demkowcicz (UT-Austin) which we will briefly present. A major improvement regarding the error estimator used in the *hp*-strategy will be presented as well as some minor enhancements to the code. Major improvements related to the multigroup problem will be described in the next chapter. Some sample 1-group results are provided in the chapter for illustrative purposes.

3.2 Principle of mesh adaptation

Dealing with the approximation error $e_h = u - u_h$, i.e., the difference between the exact solution u and the numerical solution u_h , is a very arduous task because bounds of the approximation error are complex to obtain, with the added difficulty that they are problem-dependent. In the last decade, the theory of a posteriori error estimations has matured and allows the measure, control, and minimization of approximation errors. In this theory, the computed solution itself is used to inexpensively provide point-wise error estimations. These error estimators are called a posteriori because they are determined afterwards, once a numerical solution has been obtained. By effectively estimating the error, the possibility of controlling the entire computational process emerges as the succession, within a single calculation, of adaptively refined meshes. Fig. III-1 depicts the error estimation (denoted by η) and the mesh refinement procedure. Once the solution has been computed on the ith mesh, the error is estimated using the current solution u_h^i. Process termination is determined as follows:

- If the current solution has converged within the user's defined tolerance, the process is stopped.
- If the solution has not converged sufficiently, the error estimator is used to build a new mesh *i*+1 on which a new solution will be sought.

The entire process is achieved within a single calculation, comprising a set of successively refined meshes and successive solutions.

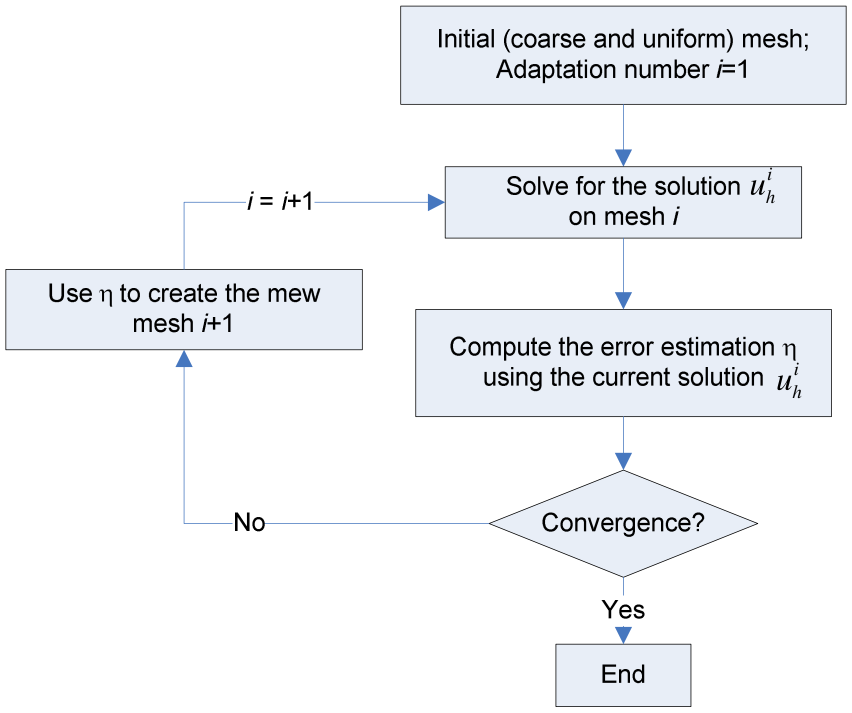

Initial (coarse and uniform) mesh; Adaptation number *i*=1
Solve for the solution u_h^i on mesh *i*
i = *i*+1
Use η to create the mew mesh *i*+1
Compute the error estimation η using the current solution u_h^i
Convergence?
No
Yes
End

Fig. III-1. Schematics of mesh adaptation

3.3 *A posteriori* error estimation

In general, the only piece of information (available to the analyst) which can provide some indication of the error is the approximate numerical solution itself. Thus, the challenge of obtaining estimates of the error in an a posteriori fashion (i.e., after the approximate solution has been obtained). The construction of a posteriori error estimates is a well studied field of numerical analysis. For survey of methods, please refer to Ref. [14] [19-20] [38-39]. Ideally, error estimates should satisfy the following conditions:

1. the error estimators should be computable from the given input data and an existing approximate solution;
2. the estimates should be lower and upper bounds of the true error in a suitable norm in the sense that constants C_1 and C_2 exist s.t. $C_1\eta \le \|e\| \le C_1\eta$. Asymptotically, as $h \to 0$ and $p \to \infty$, the estimates narrow down the error;
3. the estimates should exist *locally* in order to track down the mesh cells which contribute the most to the approximation error (bulk chasing).

Roughly speaking, the following classes of error estimators can be distinguished:

- sub-domain or element residual method: where the residual in a numerical method (the measure of how much the approximate solution fails to represent the true solution) is computed over each element or sub-domain;
- interpolation methods: these method use the interpolation theory of finite element in some Sobolev norm to produce estimates of the local error. Methods based upon the numerical Hessian for linear finite element may be also categorized as interpolation methods;
- post-processing methods: where a post-processed version of the approximate solution is compared with the approximation solution.

3.3.1 Notations

The exact solution of a 1-group fixed source diffusion problem satisfies

$$\begin{aligned} &\phi(\mathbf{x}) \in \phi_D(\mathbf{x}) + H_0^1 \\ &b(\phi,\varphi) = l(\varphi), \quad \forall \varphi \in H_0^1 \end{aligned} \tag{3.1}$$

A numerical approximation $\phi_{h,p}$ taken in an *hp*-FEM space satisfies:

$$\begin{aligned} &\phi_{h,p} \in \phi_0 + V_{h,p} \\ &b(\phi_{h,p}, \varphi_{h,p}) = l(\varphi_{h,p}), \qquad \forall \varphi_{h,p} \in V_{h,p} \end{aligned} \tag{3.2}$$

The subscripts *h,p* (the mesh size and polynomial order) are used to represent a current mesh. The approximation error is given by:

$$e_{h,p} = \phi - \phi_{h,p} \tag{3.3}$$

We have the Galerkin orthogonality relation:

$$b(e_{h,p}, \varphi_{h,p}) = 0 \tag{3.4}$$

In sections 3.3.2 and 3.3.3 we describe how to obtain an error estimation based on a reference solution. In sections 3.3.4 and 3.3.5, this error estimatate is further improved based on projection techniques.

3.3.2 Computing a reference solution

A reference solution ϕ_{ref} (which will act as the true solution ϕ in our error estimator) is an approximate solution which lies significantly closer to the exact solution ϕ than the approximation $\phi_{h,p}$ computed on the current finite element mesh (so, we will denote the current mesh as the coarse mesh and the reference mesh as the fine mesh). Usually we are interested in reference solutions that are at least by one order of accuracy better than the coarse mesh approximation.

For pure *h*-adaptivity, highly accurate approximation based on Babuska's Ref [40] extraction formulae can be used. More difficult are the *p*- and *hp*- adaptive methods, since the extraction techniques fail for higher polynomial orders. A robust way to obtain a reference solution is to *globally refine the current grid*. For *h*-adaptivity, this means that all current meshes (though of different sizes) will be subdivided in 2 (operation denoted by *h*→*h*/2). In the case of *p*-adaptive schemes, one increases the order of approximation in all elements by one (though this polynomial order may be different for all meshes); this operation is denoted by *p*→*p*+1). For *hp*-adaptive methods, we perform a *hp*→*h*/2,*p*+1 refinement, where both the mesh granularity and the polynomial order are increased. A more accurate approximate solution is sought on this finer mesh and we use:

$$\phi_{ref} = \phi_{h/2,p+1} \tag{3.5}$$

The reader may object that the computation of $\phi_{h/2,p+1}$ may become prohibitive in multi-D cases. However, *hp*-adaptivity is designed to reduce the number of unknowns by orders of magnitudes with respect to the standard *h*-adaptive schemes; this is possible because *hp* schemes deliver quasi optimal *exponential* convergence and the entire mesh adaptation process strives at delivering accuracy with the optimum mesh, i.e., with mesh containing the smallest number of cells.. Furthermore, it is possible to take advantage from the fact that the *h*/2,*p*+1 is a refinement of the coarser *h*,*p* mesh, unraveling the possibility of solving for the reference solution in a multigrid fashion. The optimal mesh is obtained iteratively by minimizing the appropriate interpolation error in each step of the mesh adaptation until the user prescribed tolerance is reached.

Note that the "reference" solution, obtained on a finer mesh, is also an approximation of the exact solution, but it is substantially more accurate than the approximation on the coarse mesh. Upon convergence of the mesh adaptation sequence, the latest reference solution is the final product and solution of the mesh adaptation scheme.

3.3.3 Additional justifications for the error estimator

Suppose we have a reference solution s.t.:

$$\begin{aligned} &\phi_{ref} \in \phi_0 + V_{ref} \\ &b(\phi_{ref}, \varphi_{ref}) = l(\varphi_{ref}), \qquad \forall \varphi_{ref} \in V_{ref} \end{aligned} \tag{3.6}$$

Its approximation error is given by:

$$e_{ref} = \phi - \phi_{ref} \tag{3.7}$$

Because $V_{h,p} \subset V_{ref}$, we also have,

$$\begin{aligned} &b(e_{ref}, \varphi_{ref}) = 0 \\ &b(e_{ref}, \varphi_{h,p}) = 0 \end{aligned} \tag{3.8}$$

For a one-group neutron fixed source neutron diffusion problem, the loss operator $-\nabla \cdot D(x)\nabla \bullet + \Sigma_r(x)\bullet$ is SPD, which results into an SPD bilinear form. The energy-norm (or equivalently the Sobolev (semi) norm of first order) of the approximation error between the reference approximate solution and the exact solution is:

$$\begin{aligned} \left\|e_{ref}\right\|_e^2 &= b(\phi - \phi_{ref}, \phi - \phi_{ref}) \\ &= b(\phi,\phi) + b(\phi_{ref},\phi_{ref}) - b(\phi,\phi_{ref}) - b(\phi_{ref},\phi) \qquad [bilinear] \\ &= b(\phi,\phi) + b(\phi_{ref},\phi_{ref}) - 2b(\phi,\phi_{ref}) \qquad [symmetric] \\ &= b(\phi,\phi) + b(\phi_{ref},\phi_{ref}) - 2b(\phi - e_{ref},\phi_{ref}) \qquad [orthogonality] \\ &= b(\phi,\phi) - b(\phi_{ref},\phi_{ref}) = \left\|\phi\right\|_e^2 - \left\|\phi_{ref}\right\|_e^2 \end{aligned} \tag{3.9.a}$$

Similarly, the energy-norm (or equivalently the Sobolev (semi) norm of first order) of the approximation error between the *hp* approximate solution and the exact solution is:

$$\left\|e_{h,p}\right\|_e^2 = \left\|\phi\right\|_e^2 - \left\|\phi_{h,p}\right\|_e^2 \tag{3.9.b}$$

And the energy norm of the difference between the reference and the *hp* approximate solutions is:

$$\begin{aligned} \left\|E_{h,p}\right\|_e^2 &= b(\phi_{ref} - \phi_{h,p}, \phi_{ref} - \phi_{h,p}) \\ &= b(\phi_{ref},\phi_{ref}) + b(\phi_{h,p},\phi_{h,p}) - b(\phi_{ref},\phi_{h,p}) - b(\phi_{h,p},\phi_{ref}) \quad [bilinear] \\ &= b(\phi_{ref},\phi_{ref}) + b(\phi_{h,p},\phi_{h,p}) - 2b(\phi_{ref},\phi_{h,p}) \qquad [symmetric] \\ &= b(\phi_{ref},\phi_{ref}) - b(\phi_{h,p},\phi_{h,p}) \qquad [orthogonality] \\ &= \left\|\phi_{ref}\right\|_e^2 - \left\|\phi_{h,p}\right\|_e^2 \\ &= \left\|e_{h,p}\right\|_e^2 - \left\|e_{ref}\right\|_e^2 \qquad [equation\ \ 2.12] \end{aligned} \tag{3.10}$$

$$\begin{aligned} &\left\|e_{h,p}\right\|_e^2 = \left\|e_{ref}\right\|_e^2 + \left\|E_{h,p}\right\|_e^2 \approx \left\|E_{h,p}\right\|_e^2 \\ &\left\|e_{h,p}\right\|_e \approx \left\|E_{h,p}\right\|_e \end{aligned} \tag{3.11}$$

Usually, the first order part of energy norm is dominant, so the H^1-semi norm is often used (due to the Poincare inequality)

$$\left|e_{h,p}\right|_1 \approx \left|E_{h,p}\right|_1 \tag{3.12}$$

For a non-symmetric case, as in the instance of a multi-group diffusion problem (PD case but not SPD case), we will need to make use of the coercivity of the (non symmetric) bilinear form. Letting $\forall \psi_{h,p}, \phi_{h,p} \in V_{h,p}$,

we have:

$$\begin{aligned}
\beta\left\|\phi-\phi_{h,p}\right\|^2 &\le b(\phi-\phi_{h,p},\phi-\phi_{h,p}) \qquad [coercivity] \\
&= b(\phi-\phi_{h,p},\phi-\psi_{h,p}) + b(\phi-\phi_{h,p},\psi_{h,p}-\phi_{h,p}) \qquad [bilinear] \\
&= b(\phi-\phi_{h,p},\phi-\psi_{h,p}) \qquad [orthogonality] \\
&\le \alpha\left\|\phi-\phi_{h,p}\right\|\left\|\phi-\psi_{h,p}\right\| \qquad [continuity]
\end{aligned}$$

and finally,

$$\left\|\phi-\phi_{h,p}\right\| \le \frac{\alpha}{\beta} \min_{\psi_{h,p}\in\phi_D+V_{h,p}} \left\|\phi-\psi_{h,p}\right\| \tag{3.13}$$

We also have,

$$\begin{aligned}
\beta\left\|\phi-\phi_{h,p}\right\|^2 &\le b(\phi-\phi_{h,p},\phi-\phi_{h,p}) = b(\phi-\phi_{h,p},\phi-\phi_{ref}) + b(\phi-\phi_{h,p},\phi_{ref}-\phi_{h,p}) \\
&\le \alpha\left\|\phi-\phi_{h,p}\right\|\left\|\phi-\phi_{ref}\right\| + \alpha\left\|\phi-\phi_{h,p}\right\|\left\|\phi_{ref}-\phi_{h,p}\right\|
\end{aligned}$$

$$\left\|\phi-\phi_{h,p}\right\| \le \frac{\alpha}{\beta}(\left\|\phi-\phi_{ref}\right\| + \left\|\phi_{ref}-\phi_{h,p}\right\|)$$

$$\left\|e_H\right\| \le \frac{\alpha}{\beta}(\left\|e_h\right\| + \left\|E_{h,p}\right\|) \approx \frac{\alpha}{\beta}\left\|E_{h,p}\right\| \tag{3.14}$$

The norm here is again unspecified, for elliptic problem the semi-H^1 norm is often employed. α and β are constants from the continuity and coercivity relations.

In summary, the quantity $E_{h,p}$ provides us with an estimation of the true error e_H. We define the total relative error estimate in semi-H^1 norm as follows:

$$E = \frac{\left|E_{h,p}\right|_1}{\left|\phi_{ref}\right|_1} = \frac{\left[\sum_K \left|\phi_{ref}-\phi_{h,p}\right|^2_{1,K}\right]^{1/2}}{\left|\phi_{ref}\right|_1} = \frac{\left[\sum_K \eta_K\right]^{1/2}}{\left|\phi_{ref}\right|_1} \tag{3.15.a}$$

where the local (element-wise) error estimate is:

$$\eta_K = \left|\phi_{ref} - \phi_{h,p}\right|_{1,K}^2 \tag{3.15.b}$$

3.3.4 Interpolation based error estimator

Under some regularity assumptions, the Galerkin method yields the best approximation (i.e., Galerkin optimality property) in the sense that there exists a positive constant C s.t. the actual approximation error is bounded by the best approximation error (Cea's lemma [32]):

$$\left\|\phi - \phi_h\right\| \le C \inf_{w_h \in \phi_D + V_h} \left\|\phi - w_h\right\|$$

where C is the ratio of the continuity and coercivity constants. For diffusion (elliptic) problems, the appropriate norm to be employed is the energy-norm (or, equivalently, the Sobolev (semi) norm of first order according to Poincare's inequality)

Cea's lemma implies that the actual error approximation can always be bounded by a mesh independent constant times the best approximation error. Thus, it is sufficient to estimate and control the best approximation error. By definition, the best approximation error is always bounded by the norm of the difference between the exact solution and any particular choice of a function that lives in the FE space. The choice made here, following Demkowicz, is the projection-based interpolant of the exact solution $\phi_{hp} = \Pi_{hp}\phi$. The symbol *hp* is a reminder that both the element size and the polynomial order will affect the interpolant.

$$\inf_{w_{hp} \in \phi_D + V_{hp}} \left\|\phi - w_{hp}\right\| \le \left\|\phi - \Pi_{hp}\phi\right\|$$

The right-hand side global interpolation error can obviously be broken into element contribution, thus providing *local* estimates of the approximation error.

$$\left\|\phi - \Pi_{hp}\phi\right\|^2 = \sum_K \left\|\phi - \Pi_{hp}\phi\right\|_K^2$$

3.3.5 New error estimator

The error estimate η_K possesses the undesirable feature that two numerical solutions are needed at each mesh adaptation step: the current (coarse) mesh solution $\phi_{h,p}$ and the reference (fine mesh) solution $\phi_{ref} = \phi_{h/2,p+1}$. We introduce here a new interpolation error to estimate the coarse mesh error:

$$\mu_K = \left| \phi_{ref,K} - \prod_{h,p_K} \phi_{ref,K} \right|_{1,K}^2 \tag{3.16}$$

Note that μ_K is different from η_K, but numerical results will show that their difference is very small, especially in the asymptotic range. Utilizing μ_K only requires one approximate solution and knowledge of the previous mesh. In order to use μ_K, we proceed as follows:

1. let a current *hp* mesh be given at any mesh adaptation stage,
2. we immediately proceed with the global mesh refinement *hp*→*h*/2,*p*+1 in order to compute the reference solution $\phi_{ref} = \phi_{h/2,p+1}$,
3. we project this reference solution back to the coarser *hp* mesh in order to compute the error μ_K,
4. based on μ_K, we either exit the mesh adaptation procedure or selectively refine the coarser *hp* mesh in order to obtain the next coarse *h'p'* mesh.

3.4 Competitive *hp*-refinements

If the total relative error *E* is less than the user-specified tolerance, then the mesh adaptation process has converged. Otherwise, a new mesh is needed. We present below the algorithm for selecting the elements to be refined as well as the method for selecting the type of refinement (*h* or *p*) to be performed for each element marked for refinement.

3.4.1 Competitive refinement choices

In pure *h*-refinement or *p*-refinement techniques, the local error estimator η_K may be used directly as a criterion to decide whether to refine the element or not. In *hp*-refinement, where for each element we have several competitive refinement choices (*h* and *p*), we need to define a technique for selecting the type of refinement to be performed. For this purpose, we utilize the reference solution, which not only provides us with a local error estimate, but can also be utilized to determine how to refine the elements marked for refinement.

Fist of all, we need a rule to limit the number of competitive refinement choices: for a given element, we will impose that the local number of degrees of freedom is only increased by 1 for each type of refinement and, in the case of *h*-refinement, we impose that the mesh be divided in two segments of equal lengths. This rule implies than we only have p_k+1 competitive choices (where p_k is element polynomial order for element *k*):

1. If *p*-refinement is chosen, we have: $h_k^{new} = h_k^{old};\ \ p_k^{new} = p_k^{old} + 1;$
2. If *h*-refinement is chosen, we have:

$$h_{k,left}^{new} = h_{k,right}^{new} = \frac{h_k^{old}}{2};\ \ p_{k,left}^{new} + p_{k,right}^{new} = p_k^{old} + 1\ \ .$$

Then, we will choose the option that will result into the highest element error decrease using the projection-based interpolation error Δerr_K. The projection-based interpolation error is defined as,

$$\begin{aligned} err_{h/2,p_{l,K}+p_{r,K}} &= \left\| \phi_{ref,K} - \prod_{h/2,p_{l,K}+p_{r,K}} \phi_{ref,K} \right\|_{1,K}^2 && \text{for } h\text{-refinement} \\ err_{h,p_K+1} &= \left\| \phi_{ref,K} - \prod_{h,p_K+1} \phi_{ref,K} \right\|_{1,K}^2 && \text{for } p\text{-refinement} \end{aligned} \tag{3.17}$$

(h/2,p_{lk}+p_{rk}) or (h,p_k+1) represents a local sub polynomial space which is a sub-space of the reference space. Π is the H^1 semi-norm projection operator described in Chapter II. We ignore superscript 1 by default.

The idea of optimal energy norm-driven refinement schemes is to equi-distribute the error as per DOF. A refinement strategy designed to reduce the error as much as possible would make the change in error per change in DOF as large as possible. For a given element marked for refinement, we should select the refinement option (*h* or *p*) that will lead to the largest error decrease per increase in the number of DOF: Δerror/Δndof. Since $\Delta\text{error}/\Delta\text{ndof}\big|_{h=const}$ and $\Delta\text{error}/\Delta\text{ndof}\big|_{p=const}$ are different, then the refinement type in the larger of these two should be adopted in searching for the optimum mesh. All refinement types considered here increase the number of DOF by exactly 1, so the rate of decrease in the interpolation error is,

$$\begin{aligned}\Delta err_K &= err_{h/2,p_{l,K}+p_{r,K}} - \mu_K \\ \text{or} \quad &= err_{h,p_K+1} - \mu_K, \text{ depending on the refinement choice.}\end{aligned} \tag{3.18}$$

The refinement choice, which maximizes the reduction of element interpolation error, is then chosen.

3.4.2 *hp*-refinement strategy

There is a balance between how far we would go in one mesh iteration step and how close our result mesh is to optimal mesh. If we choose refine more elements in one step, we can expect to reach our prescribed tolerance in fewer mesh iteration. But on the other hand, our results mesh may be far from optimal, an extreme case is that if we choose refine all elements for each step in *h*-refinement, we end up with uniform *h*-refinement.

In practice, we only refine elements, whose local interpolation error is greater than a coefficient α_1 times the maximum local interpolation error,

$$\mu_K > \alpha_1 \max_K \mu_K \tag{3.19.a}$$

A rule of the thumb for the choice of α_1 is:

$$\alpha_1 = 1/3 \tag{3.19.b}$$

Other rules can be imagined:

$$\eta_K > \alpha_1 \max_K \eta_K \quad \text{or} \tag{3.20.a}$$

$$\Delta err_K > \alpha_1 \max_K \Delta err_K \tag{3.20.b}$$

The current *hp*-code uses Eqs. (3.19a) and (b), and practices have proved this rule performs well. Eq. (3.20.a) (original error estimation in the *hp*-code) is almost identical to Eq. (3.19.a) but it demands the calculation on the coarse mesh. Numerical results showed Eq. (3.20.b) may cause the mesh adaptation to sometimes fail.

3.4.3 *h*-constraint and some numerical limitations

Another issue is that, when we are choosing to perform *h*-refinement, we may be losing the chance to get an optimal mesh. For example, suppose we only have one initial element and we know the exact solution to be a higher order shape function whose order p_1 is greater than the initial order p_0. We can imagine that at first *h*-refinement will decrease the error more prominently than *p*-refinement. If *h*-refinement is chosen, it will never deliver the optimal mesh here which is a single element whose polynomial order is p_1. We, therefore, implemented an additional restriction on *h*-refinements,

$$\Delta err_K > \alpha_2 \mu_K \tag{3.21}$$

, which means that *h*-refinement only wins when it produces relative big error drop. α_2 varies from 0 to 1. When α_2=1, we will never perform an *h*-refinement if we have the choice of *p*-refinement. α_2=0 means that there is no restriction on *h*-refinement. The higher accuracy we desire, the higher α_2 we may need.

Some numerical limitations may exist, for example the maximum of polynomial order. When p_{max} is reached, we have to enforce *h*-refinement regardless of the error drop. The flowchart of *hp*-refinement is given in Fig. III-2.

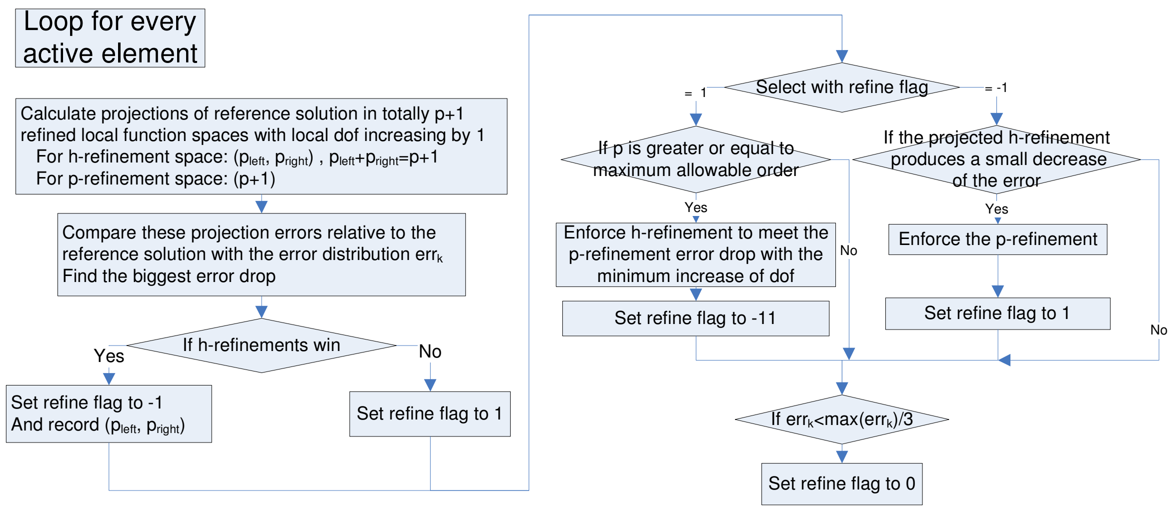

Loop for every active element
Calculate projections of reference solution in totally p+1 refined local function spaces with local dof increasing by 1
For h-refinement space: (p_left, p_right) , p_left+p_right=p+1
For p-refinement space: (p+1)
Compare these projection errors relative to the reference solution with the error distribution err_k
Find the biggest error drop
If h-refinements win
Yes
No
Set refine flag to -1
And record (p_left, p_right)
Set refine flag to 1
Select with refine flag
= 1
= -1
If p is greater or equal to maximum allowable order
Yes
No
Enforce h-refinement to meet the p-refinement error drop with the minimum increase of dof
Set refine flag to -11
If the projected h-refinement produces a small decrease of the error
Yes
No
Enforce the p-refinement
Set refine flag to 1
If err_k<max(err_k)/3
Set refine flag to 0

Fig. III-2. *hp*-refinement strategy

After a reference solution has been obtained during a mesh iteration, we start looping over all active elements. For each active element, we project the reference solution onto a series of subspaces corresponding to *h* and *p* refinement choices and to the coarse space. Then, we calculate the error drops for all competitive *hp* choices and select the one satisfying our criteria. We may need enforce *p*-refinement when *h*-constraint is met or *h*-refinement when $p_{\max}$ is reached. Finally we set the refinement flag based on the local interpolation error for the element under consideration.

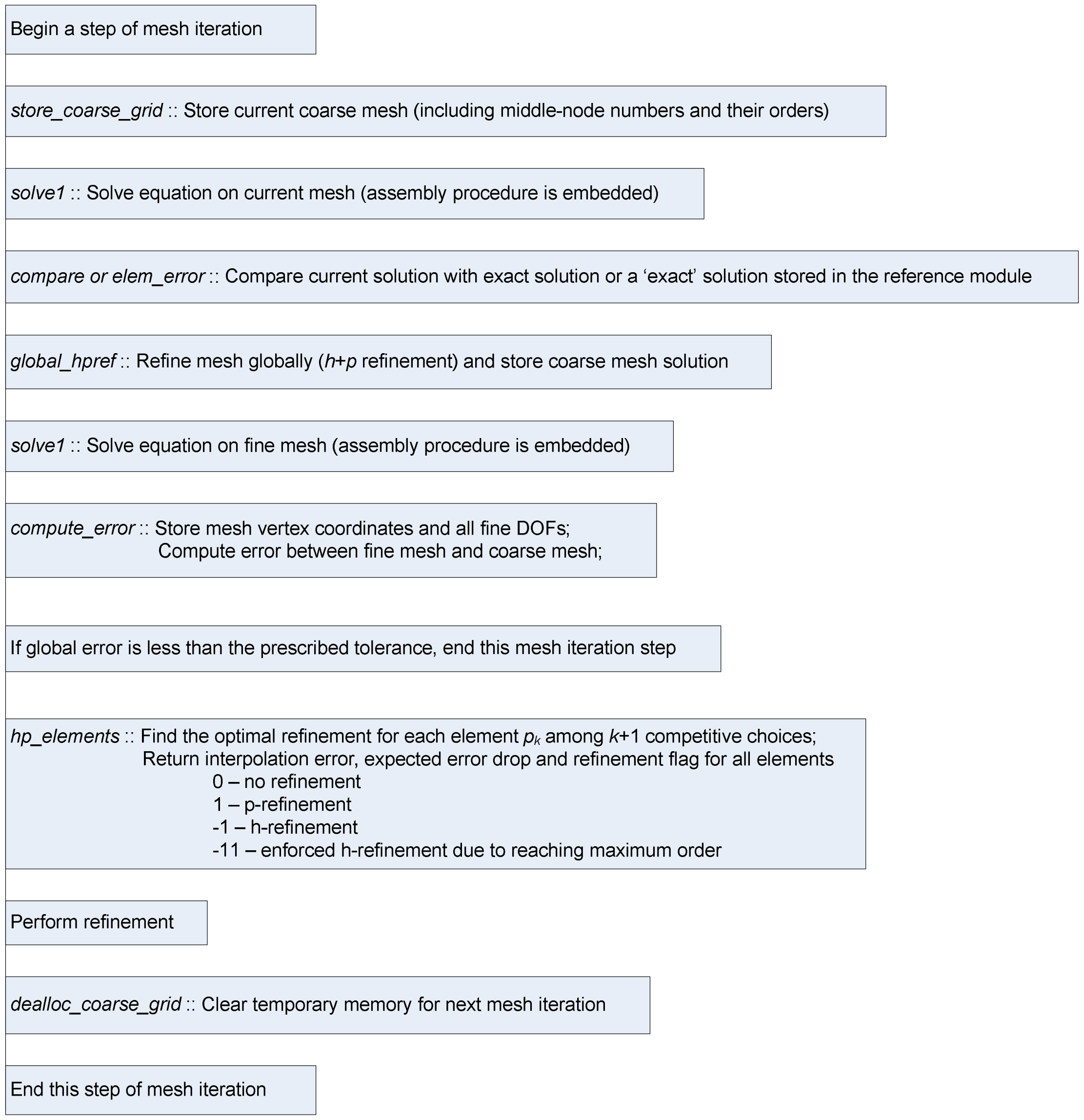

Begin a step of mesh iteration
store_coarse_grid :: Store current coarse mesh (including middle-node numbers and their orders)
solve1 :: Solve equation on current mesh (assembly procedure is embedded)
compare or elem_error :: Compare current solution with exact solution or a 'exact' solution stored in the reference module
global_hpref :: Refine mesh globally (*h*+*p* refinement) and store coarse mesh solution
solve1 :: Solve equation on fine mesh (assembly procedure is embedded)
compute_error :: Store mesh vertex coordinates and all fine DOFs; Compute error between fine mesh and coarse mesh;
If global error is less than the prescribed tolerance, end this mesh iteration step
hp_elements :: Find the optimal refinement for each element p_k among *k*+1 competitive choices; Return interpolation error, expected error drop and refinement flag for all elements
0 – no refinement
1 – p-refinement
-1 – h-refinement
-11 – enforced h-refinement due to reaching maximum order
Perform refinement
dealloc_coarse_grid :: Clear temporary memory for next mesh iteration
End this step of mesh iteration

Fig. III-3. Flow chart of one *hp*-adaptive mesh iteration step

The flowchart of one step of *hp*-adaptive mesh iteration is given in Fig. III-3. At each step, we calculate a coarse solution ϕ_{hp} and a reference solution ϕ_{ref}. With these two solutions we are able to calculate local error distribution η or μ. With the reference solution alone, we can perform *hp*-refinement to set the refinement flag and select the refinement type. The last item consists in calculating the total error: then, if convergence to the specified tolerance has been reached, the mesh adaptation process is terminated, otherwise the current mesh is refined and the next step of mesh iteration is scheduled. Note that we do not need to calculate the coarse solution ϕ_{hp}.

3.4.4 Expected convergence rates

It is customary to write the interpolation error estimator as follows (REF):

$$\left|\left(I-\Pi_{hp}\right)\phi\right|_{1,K} \leq C\frac{h^{\min(p,r)}}{p^{r}}\|\phi\|_{r+1,K}$$

, where p is the polynomial order, r the regularity index, h the mesh size. We now discuss the expected convergence rates for various mesh refinement options.

3.4.4.1 Uniform *h* refinements

This is the most classical usage of FD and FE methods. Starting with an initial mesh of uniform order p, and decreasing the mesh size h, we have:

$$\left\|\phi-\phi_{h}\right\|_{1} \leq Ch^{\min(p,r)}\|\phi\|_{r+1} = cN^{-\min(p,r)} \tag{3.22}$$

, where N is the total number of DOF (N is inversely proportional to the element size and each element contributes with p DOF: $N = p/h$). On a log-log scale, the plot of the error as a function of the number of degrees of freedom N is a straight line, whose the slope (i.e., the rate of convergence) equal to *min(p,r)*: $\log\left\|\phi-\phi_{h}\right\|_{1} \approx c+\min(p,r)\log h.$ The constant c hides all other constant quantities used in the error estimate.

The regularity r of a solution ϕ is defined as the maximum order of the Sobolev space the solution belongs to. Under the assumption of smooth data, 1-D neutron diffusion problems very often yield infinite regularity, so the convergence rate is only dependent upon the polynomial order p.

3.4.4.2 Uniform and adaptive *p*-refinements

In lieu of decreasing the element size, one may increase uniformly or adaptively the polynomial order of the approximation. This leads to the following estimates for uniform *p*-refinement:

$$\|\phi-\phi_h\|_1 \le Cp^{-r}\|\phi\|_{r+1} = cN^{-r} \tag{3.23}$$

Convergence rate is again determined by regularity. For smooth solutions, there is no limit on the convergence rate and exponential convergence is expected.

3.4.4.3 Adaptive *h*-refinements

Instead of uniformly subdividing the mesh (uniform *h*-refinement), we can refine only the elements where the error is large. Without going into the details of adaptive *h*-refinement (see REF), we stress out that adaptive *h*-refinement allow to eliminate the influence of the solution regularity from the convergence rate, yielding:

$$\|\phi-\phi_h\|_1 \le cN^{-p} \tag{3.24}$$

Hence, comparing with uniform *h* refinements, *h*-adaptivity eliminates the influence of regularity. However, for regular solutions, the *h*-adaptivity does not improve the rate of convergence (though it is expected that the constants in the estimates are smaller and hence the overall error, though decreasing at the same rate as uniform *h*-refinement, will be smaller). Another great advantage is that fewer DOF will be needed in *h*-adaptivity compared to uniform *h*-refinement.

3.4.4.4 Adaptive *hp*-refinements

In adaptive *hp*-refinement (the most flexible type of refinement), we can always obtain an exponential convergence,

$$\|\phi-\phi_h\|_1 \le Ce^{-\gamma N^{1/\alpha}} \tag{3.25}$$

for both regular and unsmooth (singular) solutions. Coefficient α depends on the dimension of the problem. In one dimensions α=1, two dimensions α=3 and three dimensions α=5. The coefficient γ>0 depends on the strength of the singularity of the solution in the neighborhood of the boundary and interfaces.

Summary: uniform and adaptive *h*-refinement schemes, though very popular, only deliver algebraic convergence rates.

3.5 The 1D-*hp* code (Demkowicz, UT-Austin)

The open-source 1-D *hp*-FEM code developed by Dr. Demkowicz at the University of Texas in Austin is available from the web at http://www.ticam.utexas.edu/~leszek/projects.html. This code will serve as the basis for our research in the multigroup eigenproblem setting (described thoroughly in Chapter IV). We present here some features of this code (data structure for adaptive mesh refinement, a useful integration procedure and some minor enhancements).

3.5.1 Data structure for *hp*-refinement and some associated algorithms

3.5.1.1 Initial mesh

Usually it determined by material composition, i.e., interfaces of different materials are always interfaces of initial element. And it is possible that initial mesh contains some information about our knowledge on the singularity of the solution. For very complicated geometry and/or applications, initial mesh can be generated with state-of-art mesh generators. Initial mesh should contain the element connectivity information and boundary information.

In the 1D-*hp* code, besides initial vertex and middle nodes, there is an array ELEMENTS which describes the initial mesh. Each entry of ELEMENTS corresponds to an initial element, which contains connectivity information such as the number of its two vertices, its middle node, initial neighbor elements. (0 initial neighbor means the element is adjacent to the boundary.)

3.5.1.2 Nodes

FEM researchers usually call all entries such as vertex and middle node are nodes in the viewpoint of coding. There are two main types of nodes in 1-D: vertex and middle node. The array NVERS is used to store the information for all vertices appearing in the mesh adaptation process. Each entry of NVERS corresponds to a vertex which includes its coordinate, its nodal value, its father element number and its boundary condition. A negative father number indicates that this vertex belongs to the initial mesh. The array NODES is used to store information related to the middle nodes appearing in the mesh adaptation process. Because each middle node corresponds to a finite element, we do not need an additional array to store the finite elements themselves. Each middle node has a father element and, if it is further refined. In this array, there are also the polynomial order and solution DOFs associated with middle nodes. Note that nodes are only a code concept so, even when the element polynomial order is equal to 1, a middle node still exists as a data.

3.5.1.3 Active finite element

All elements without son elements are called active finite elements. They compose the current computational mesh. If we look all nodes as a tree, active finite element are the leaves of the tree. The numbering of elements in the initial mesh (in 1-D generally from the left to the right) and the family tree structure induce the so-called natural order of active elements.

3.5.1.4 Tree structure of *h*-refinement

Let us see how this data structure represents the following refinement example illustrated in Fig. III-4. In this example, there are three initial element numbered from 1 to 3. So there are 3 initial middle nodes numbered from 1 to 3 and 4 initial vertices numbered 1 through 4. The father of initial vertex is the number of right initial element. A negative sign indicates a root vertex.

Initial element number	Left vertex	Right vertex	Middle node	Left initial element	Right initial element
1	1	2	1	0	2
2	2	3	2	1	3
3	3	4	3	2	0

Initial vertex number	Boundary flag	coordinate	Solution	Father element
1	1/2/3[a]	–	–	-1
2	0	–	–	-2
3	0	–	–	-3
4	1/2/3	–	–	-4

Initial middle node	Polynomial order	Father middle node	Left son middle node	Right son middle node	Son vertex	DOFs (p-1)
1	–	-1	**8**	**9**	**7**[b]	–
2	–	-2	**4**	**5**	**5**	–
3	–	-3	**6**	**7**	**6**	–

'–' means it is problem dependent

[a] Depends on type of boundary condition

[b] Number of son nodes are filled after mesh refinement

After the first mesh adaptation, two vertices and four middle nodes are added.

Vertex number	Boundary flag	coordinate	Solution	Father element
5	0	–	–	2
6	0	–	–	3

Middle node	Polynomial order	Father middle node	Left son middle node	right son middle node	Son vertex	DOFs (p-1)
4	–	2	**0**	**0**	**0**	–
5	–	2	**10**	**11**	**8**	–
6	–	3	**12**	**13**	**9**	–
7	–	3	**0**	**0**	**0**	–

After the second mesh adaptation, 3 vertices and 6 middle nodes are added.

Vertex number	Boundary flag	coordinate	Solution	Father element
7	0	–	–	1
8	0	–	–	5
9	0	–	–	6

Middle node	Polynomial order	Father middle node	Left son middle node	right son middle node	Son vertex	DOFs (p-1)
8	–	1	0	0	0	–
9	–	1	0	0	0	–
10	–	5	0	0	0	–
11	–	5	14	15	10	–
12	–	6	16	17	11	–
13	–	6	0	0	0	–

After the third mesh adaptation, i.e., the last mesh iteration in our example here, two vertices and four middle nods are added.

Vertex number	Boundary flag	coordinate	Solution	Father element
10	0	–	–	11
11	0	–	–	12

Middle node	Polynomial order	Father middle node	Left son middle node	right son middle node	Son vertex	DOFs (p-1)
14	–	11	0	0	0	–
15	–	11	0	0	0	–
16	–	12	0	0	0	–
17	–	12	0	0	0	–

Active elements are number 8, 9, 4, 10, 14, 15, 16, 17, 13 and 7 in nature order. There are total three generations and there are also three mesh iterations. But it needs to be pointed out that generation and mesh iteration concepts are different concepts; in this case, their numbers just occasionally agree.

Several representative algorithms operated on this data structure are listed here. One can get some idea about how the code works from their functionalities. For the completeness, one should refer to the code itself.

3.5.1.5 Some associated algorithms

break: *h*-refine one active element, create two son middle nodes and 1 son vertex

nelcon: find the next active element in the nature order of elements
history: create refinement history of a middle node
solelm: assembly solutions of a finite element, two vertex solutions plus p-1 middle node DOFs
nodmod: modify polynomial order of middle node

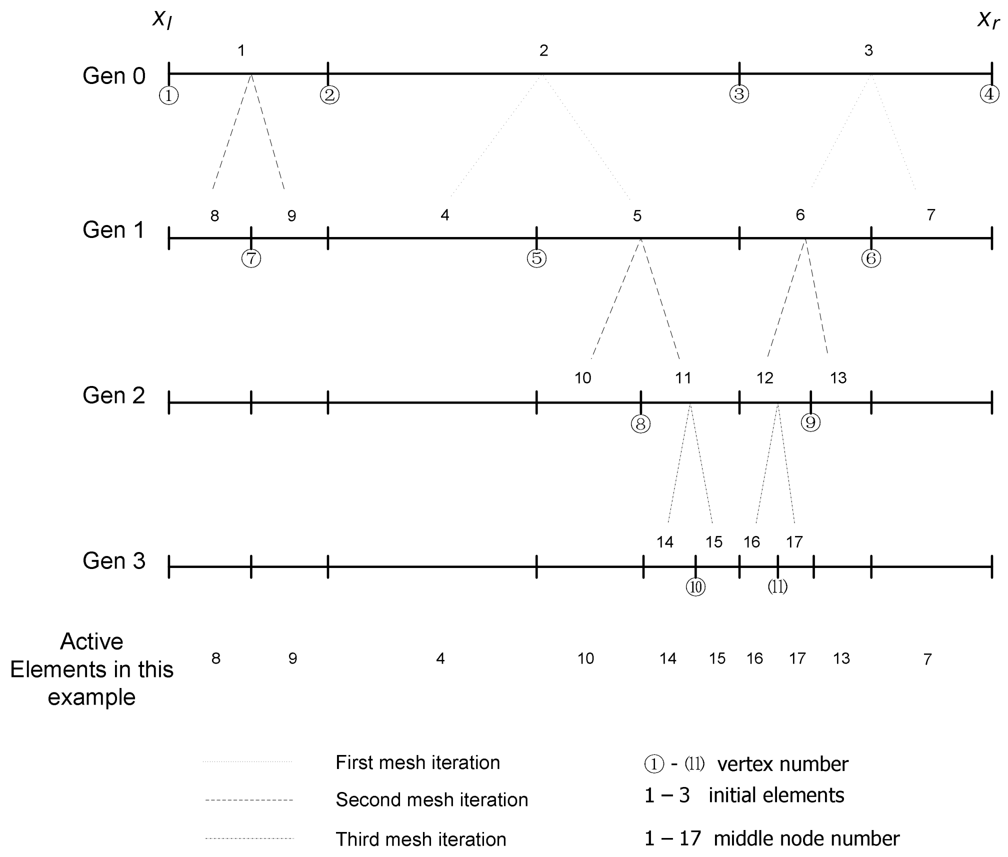


Fig. III-4. Tree structure of *h*-refinement

3.5.2 Adaptive integration during assembly

Adaptive integration was introduced to calculate accurately the load vector when a singularity in load or material data was present. In practice, adaptive integration can be used to manipulate group coupling with different group mesh and compare solutions with an "exact" numerical solution with highly refined mesh. So it is appropriate to present it in detail here. Its application regarding with multi-mesh coupling will be discussed in Chapter IV in detail.

Let us consider an integral to be calculated on a given element. We start the adaptive integration procedure by first creating a list of subintervals for the integration place the element in the list. We then retrieve the first subinterval from the list (which, at the beginning, is the whole element), and compute the integral twice, first on the whole subinterval and then by splitting the subinterval into two equal subintervals, integrating over the two subintervals separately, and adding up the two subinterval contributions.

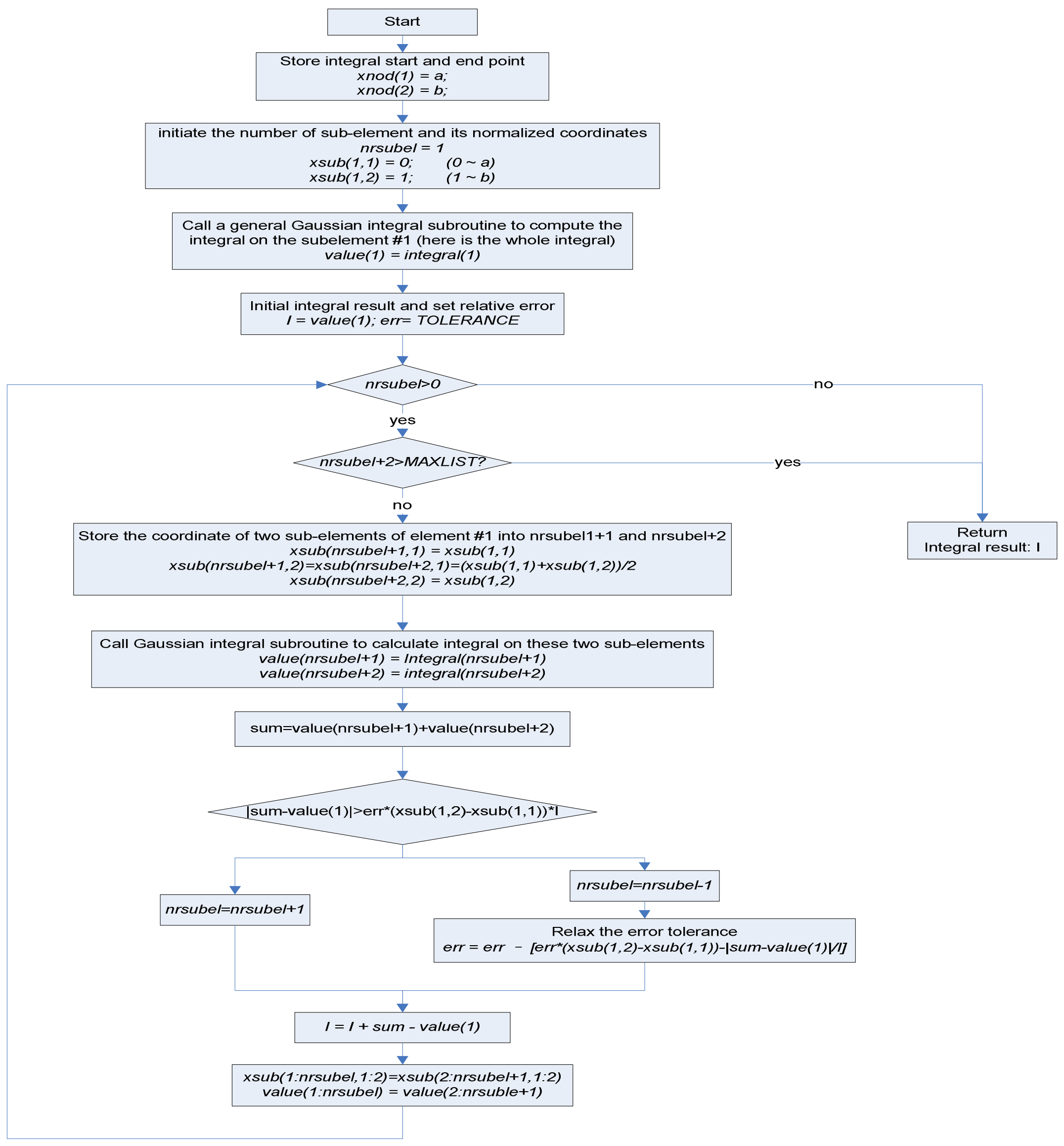

Start
Store integral start and end point
xnod(1) = a;
xnod(2) = b;
initiate the number of sub-element and its normalized coordinates
nrsubel = 1
xsub(1,1) = 0; (0 ~ a)
xsub(1,2) = 1; (1 ~ b)
Call a general Gaussian integral subroutine to compute the integral on the subelement #1 (here is the whole integral)
value(1) = integral(1)
Initial integral result and set relative error
I = value(1); err= TOLERANCE
nrsubel>0
no
yes
nrsubel+2>MAXLIST?
yes
no
Return
Integral result: I
Store the coordinate of two sub-elements of element #1 into nrsubel1+1 and nrsubel+2
xsub(nrsubel+1,1) = xsub(1,1)
xsub(nrsubel+1,2)=xsub(nrsubel+2,1)=(xsub(1,1)+xsub(1,2))/2
xsub(nrsubel+2,2) = xsub(1,2)
Call Gaussian integral subroutine to calculate integral on these two sub-elements
value(nrsubel+1) = Integral(nrsubel+1)
value(nrsubel+2) = integral(nrsubel+2)
sum=value(nrsubel+1)+value(nrsubel+2)
|sum-value(1)|>err*(xsub(1,2)-xsub(1,1))*I
nrsubel=nrsubel-1
nrsubel=nrsubel+1
Relax the error tolerance
err = err – [err*(xsub(1,2)-xsub(1,1))-|sum-value(1)|/I]
I = I + sum - value(1)
xsub(1:nrsubel,1:2)=xsub(2:nrsubel+1,1:2)
value(1:nrsubel) = value(2:nrsuble+1)

Fig. III-5. Flowchart of adaptive integral algorithm

Obviously, the integral value computed using two subintervals is more accurate. If the difference between the two results exceeds a tolerance level, we add both subintervals to the list; if the tolerance has been met, we accumulate (with the more accurate value) the subinterval contribution for subsequent aggregation with the integral over the whole initial element and remove the subinterval from the list. We then proceed with the next available subinterval in the list, until the list is empty. Fig. III-5 presents the flowchart for the adaptive integration procedure.

3.5.3 Several enhancements to the original 1-D *hp*-FEM code

3.5.3.1 Migration of the code from Linux to CVF (Compaq Visual Fortran) Windows

Though CVF may not be the most effective complier, it provides an integrated powerful debug environment.

3.5.3.2 Geometry description

In the problems of reactor physics, cross sections are usually piece-wise constant functions determined by material composition. Correspondingly, we must have every initial mesh only composed of one material. This extension allows flexibility in the code to describe any composition-based geometry. To avoid complexity, the code only supports piece-wise constant extraneous sources as of now.

3.5.3.3 Increase p_{max}

The maximum polynomial order p_{max} of the original code was increased from 8 to 20.

3.5.3.4 Add support of Lobatto shape functions

The original code did not have Lobatto shape functions. We added these shape functions in 1D.

3.5.3.5 A “exact” solution

In solving real problems, we do not have an exact solution. However, we can get a much more accurate solution by setting the user-tolerance to extremely low values. This ‘exact’ solution is still a numerical approximation but is highly accurate and can be used to analyze convergence properties. To add this support, a new module REFERENCE was created, in which an “exact” solution, including its mesh structure and DOFs, is stored. A subroutine COMPARE within the module compares any approximate numerical solution with the stored “exact” numerical solution. Because the mesh of the “exact” solution is much more refined, adaptive integration is used to calculate the error. We will explain this in detail in next chapter because adaptive integration will be a powerful tool in the context of multigroup equations where fluxes can be computed on different meshes.

3.5.3.6 Use different norm to evaluate interpolation error

In addition to using the H^1 semi-norm in Eqs. (3.15), (3.16) and (3.17), the L_2 and energy norms were also coded. Note this option does not affect the interpolation operation. We still use semi-H^1 norm to perform the interpolation but this additional feature allows us to verify and compare error estimations with several norms.

3.5.3.7 Options of *hp*-strategy

In section 3.4.2 we described a refinement strategy in which only elements, whose local interpolation error exceeds a certain value (fraction of the maximal error), are refined. We denote this as *error-based* refinement strategy. Other strategies could include:

3.5.3.7.1 Sorting-based refinement strategy

We can number elements with their local interpolation error descending I_k, k=1,2,…,*Nel*. *Nel* is the total number of elements. And only refine elements for which

$$I_k < \alpha_3 Nel \tag{3.26}$$

3.5.3.7.2 Cumulative error-based refinement strategy

We can a cumulative error (ε_k, k=1,2,…,*Nel*. *Nel* is the total number of elements) for a given element k, where the cumulative error is the accumulation of the local interpolation errors of all elements whose local errors is larger that the local error for the element under consideration. And we refine elements for which

$$\varepsilon_k < \alpha_4 \sum_{k=1}^{Nel} \mu_k \tag{3.27}$$

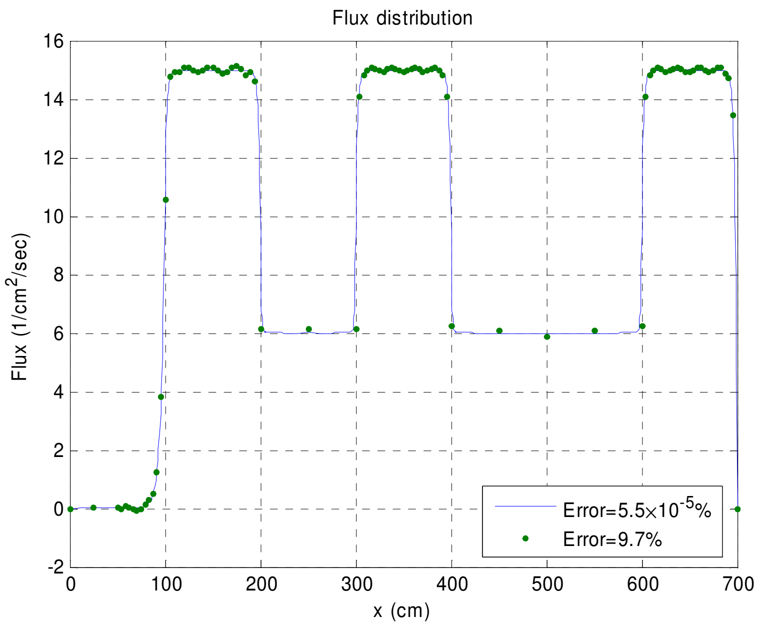
We will denote Eq. (3.26) as *sorting-based* refinement strategy and Eq. (3.27) as *cumulative error-based* refinement strategy.

3.6 One-group fixed source diffusion problems

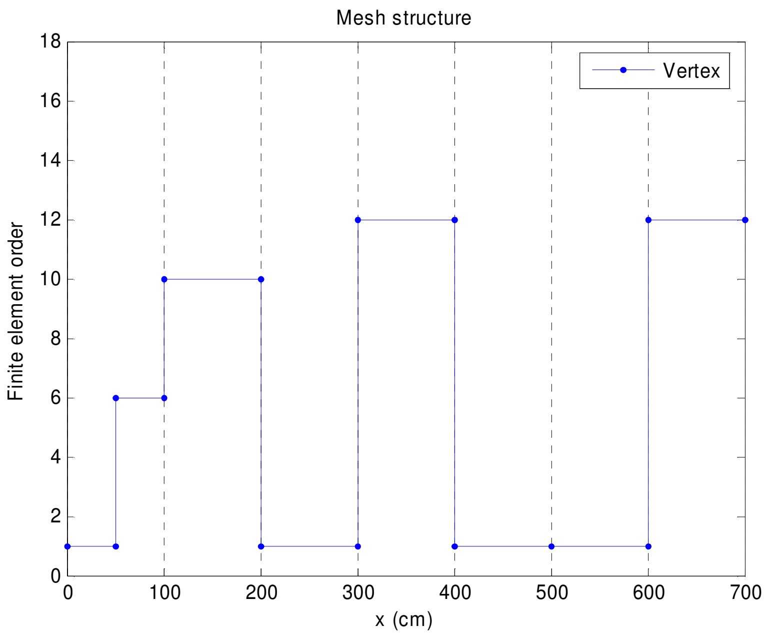
3.6.1 Example 1.A (A 1-D one-group neutron diffusion source problem)

This example is exactly same as the one-group example in Chapter II. Three different materials are placed in 7 seven regions as 1-2-3-2-3-3-2, each region is 100-cm thick. Both boundaries are Dirichlet zero flux conditions. For each material, the diffusion coefficients are 0.333, 0.370 and 0.303cm respectively; the absorption cross sections are 0.02, 0.1, and 0.3cm^{-1}; and the volumetric neutron source terms are 0.0, 1.5 and 1.8n/cm^{3}-sec.

The computation options are: 1 initial element per assembly; initial polynomial order 1; interpolation error distribution; α_1=1/3; α_2=0.9; p_{max} = 17. Uniform refinement produced exact same results with the 1-D FEM demonstration code.

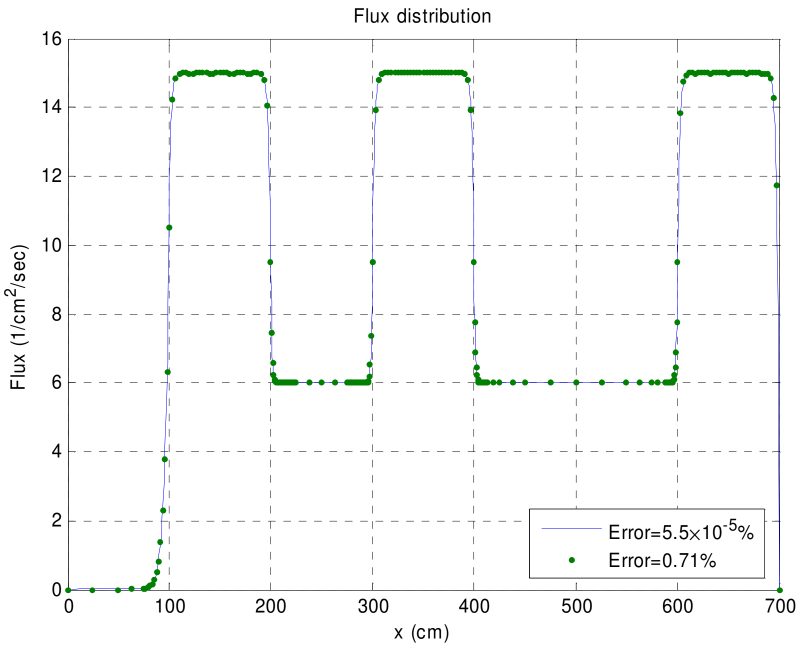


(a)

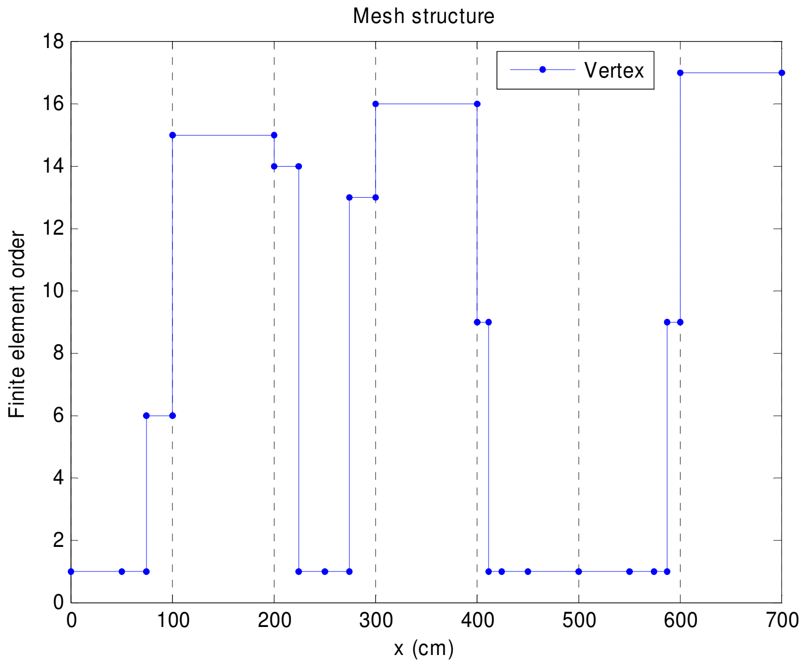


(b)

Fig. III-6. Solutions of example 1-A with tolerance=10%: (a) flux and (b) mesh structure

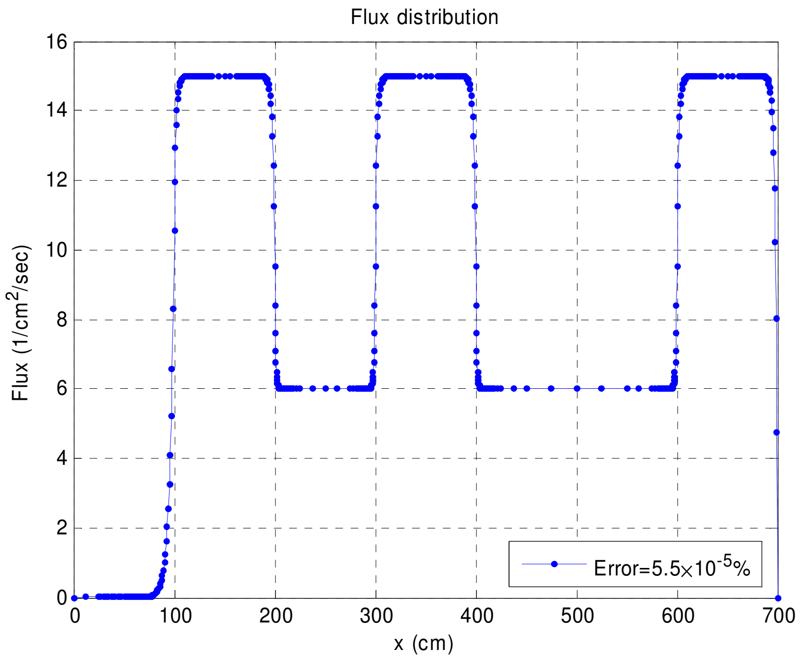


(a)

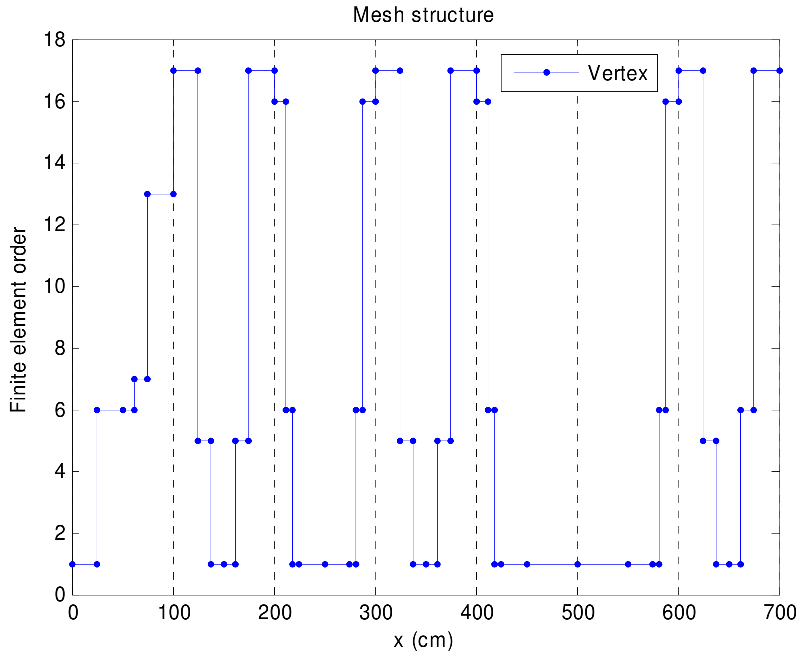


(b)

Fig. III-7. Solutions of example 1-A with tolerance=1%: (a) flux and (b) mesh structure



(a)



(b)

Fig. III-8. Solutions of example 1-A with tolerance=10^{-5}%: (a) flux and (b) mesh structure

The main results with three different tolerances 10%, 1% and 10^{-5}% are shown in Fig. III-6 to Fig. III-8 and are listed in TABLE III-I.

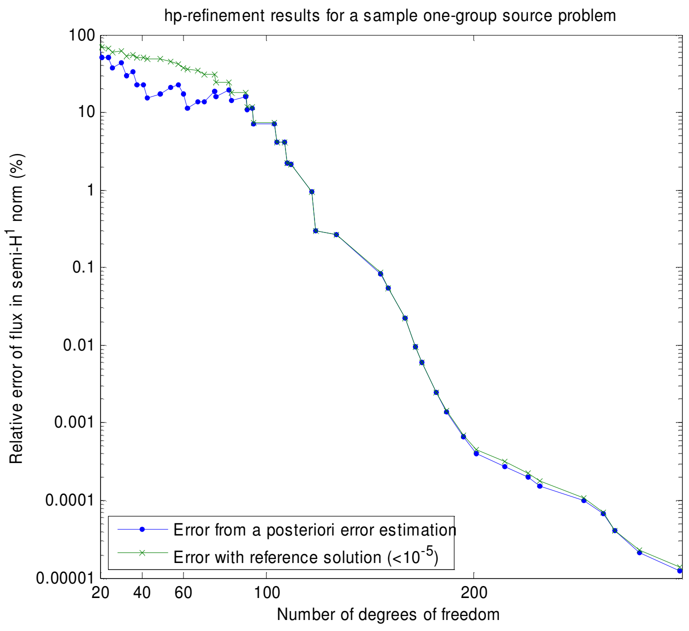
TABLE III-I

Main results of example 1.A

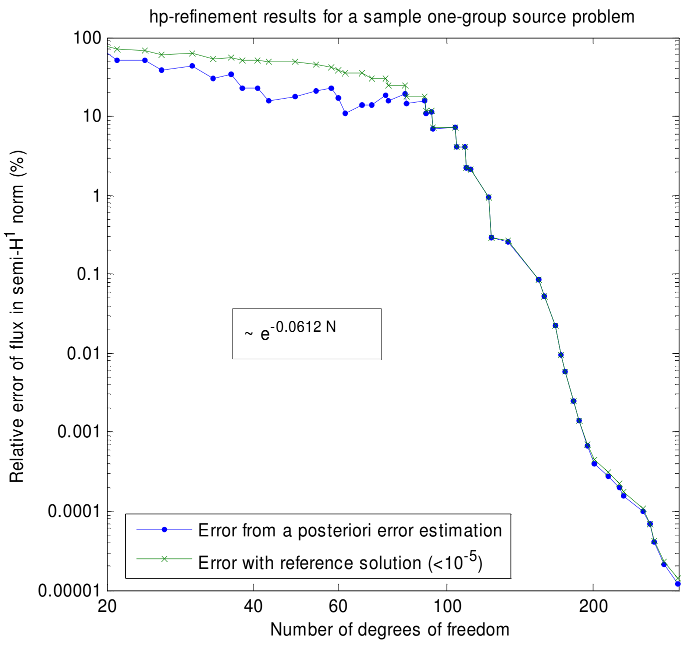
	Convergent error (%)		
	9.7	0.71	5.5×10^{-5}
Number of mesh iteration	14	34	54
Number of result finite elements	8	18	41
Global degree of freedom	45	110	271

Figs. III-6(a), III-7(a) and III-8(a) present the flux distribution with different error tolerances. To plot the flux, we used $2*p_k+1$ points for each elements. Mesh structures are shown in Figs. III-6(b), III-7(b) and III-8(b). (Points shown are the vertices of finite elements.) It is obvious that elementary orders are larger and sizes are smaller in the transient zone.

We can see the convergence properties in Fig. III-9. The only difference between Figs. III-9(a) and III-9(b) is that we are using linear *x* coordinate in (a) *log x* coordinate in (b). Each point on the curve corresponds to one step of the mesh adaptation procedure. In Fig. III-9, we can see that we obtain exponential convergence. The fitted slop is equal to 0.0612. The green curves are the errors with an "exact solution" whose accuracy is less than 10^{-5}%. The blue line is obtained from the *a posteriori* error estimator. Comparing these two curves, we can make a conclusion the local effectivity index in the asymptotic range is nearly equal to 1. And in the pre-asymptotic range, we can see the estimated errors may temporarily increase with refinement while they are in reality decreasing. Even starting with a relative coarse mesh, we still can converge the *hp* mesh adaptation scheme properly.



(a)



(b)

Fig. III-9. Convergence properties of *hp*-adaptive: (a) log-linear and (b) log-log

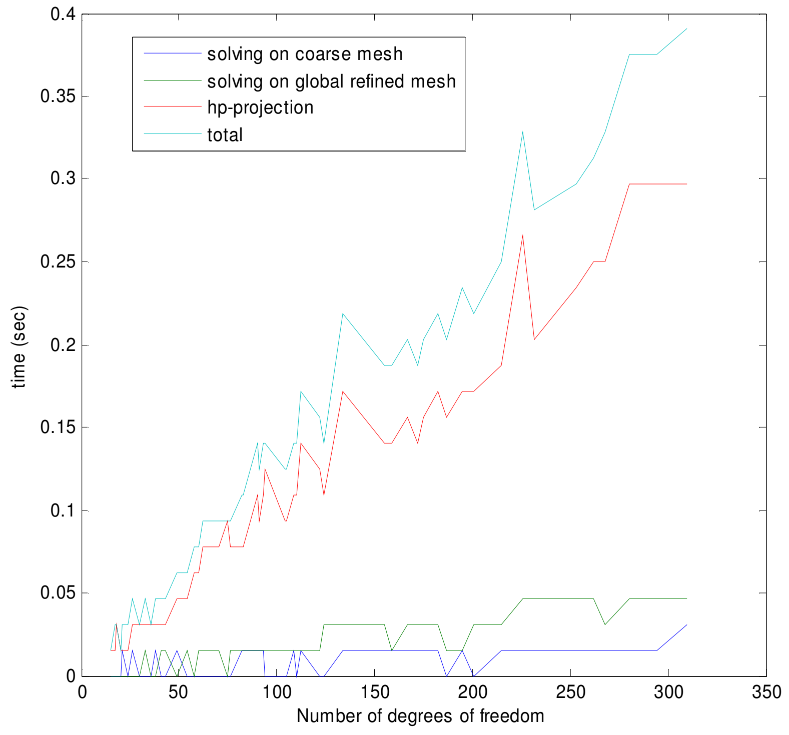
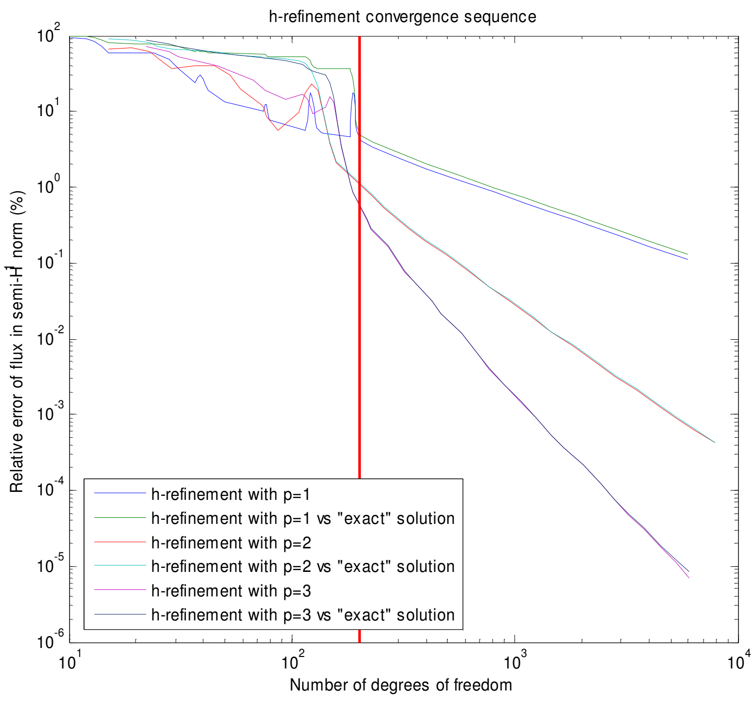


Fig. III-10. Computational cost

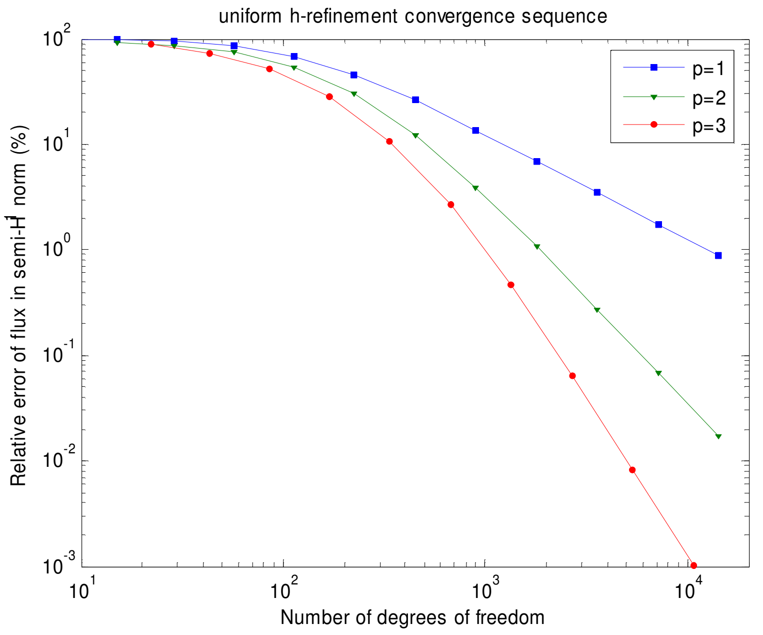
After implementing some time measurement functions in the code, we noticed three contributors to the CPU cost due to the mesh iteration process: solving on the current mesh, solving on the fine mesh and *hp*-projection (this was performed before switching the error estimator from η to μ). Fig. III-10 gives an example of the CPU cost. We notice that in one-dimension, the solver cost is smaller time than the *hp*-projection itself. It seems *hp*-projection operation increase linearly with number of degrees of freedom. Because computing effort of linear solver generally is proportional to N^3, so we can expect that in multi-dimension when the number DOF is larger, *hp*-projection CPU time will only be a small fraction of solver CPU time. Anyway, we did not try to optimize the *hp*-projection at this stage. This issue needs to be considered carefully for multidimensional studies.

Currently for *p*=2 uniform refinement, when global DOFs=7169, solution error 0.0689 is below 0.1. Only the solver for this single step costs about 2.20 second on our 3.4GHz Pentium IV processor. And for *hp*-refinement after 41mesh iterations, error is 0.00589. Time cost of solver total including fine mesh solver is 0.80 second. Together with *hp*-projection, time is 4.16 second.

We also analyzed the convergence properties of *h*-refinement. These convergence properties are shown on Fig. III-11. The reference solution is obtained with global *h*-refinement. For the linear finite element we can see the effectivity index is not equal to 1 exactly but the trend is still same. For higher uniform polynomial approximations, the effectivity index reached 1. 200 DOFs seems to be the threshold to reach the asymptotic range in our example. Uniform *h*-refinement was also considered. We saw that the higher polynomial order, the higher convergence rate will be. We clearly note the *algebraic convergence* of *h*-uniform and *h*-adaptive strategies (on a log-log plot, the slope of the error, measure in the semi H^1 norm is equal to the constant polynomial order used: p). Note that *h*-strategies can be easily implemented in an *hp*-code by switching off the *p*-refinement option.



(a)



(b)

Fig. III-11. Convergence properties of: (a) *h*-adaptation and (b) uniform *h*-refinement

We also analyzed the convergence of uniform *p*-refinement scheme, shown on Fig. III-12. As theoretically expected, *p*-adaptive schemes also yield exponential convergence, but with a lower slope (0.038 on Fig. III-12). We, therefore, note that *hp*-adaptation provides a better mesh.

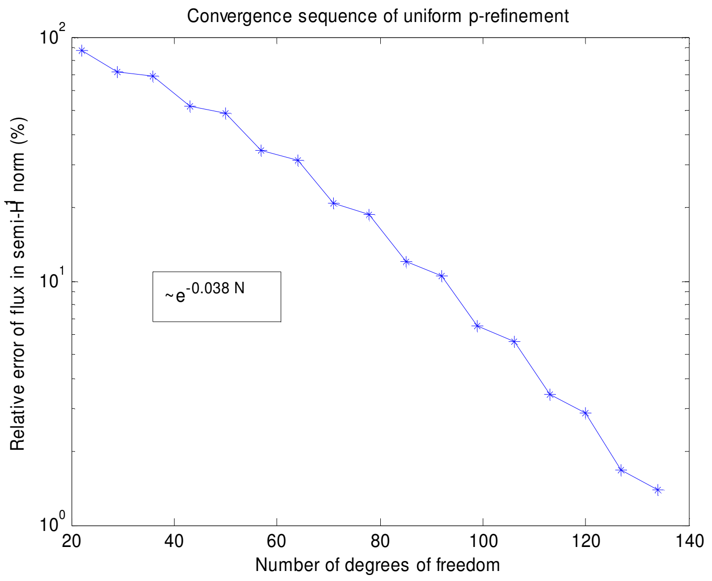


Fig. III-12. Convergence of uniform *p*-refinement

For exhaustive comparison, we provide all convergence analysis results in Fig. III-13. For *hp*-refinement, we even can reach an extremely low tolerance of 10^{-6}% with an amazingly small number of degrees of freedom.

Summarizing the results shown on Fig. III-13, we note that *h*-uniform and *h*-adaptive strategies yield algebraic convergence (equal to the polynomial order when the error is measure in the H^1 or semi H^1 norms). *hp*-refinement yields an exponential convergence behavior.

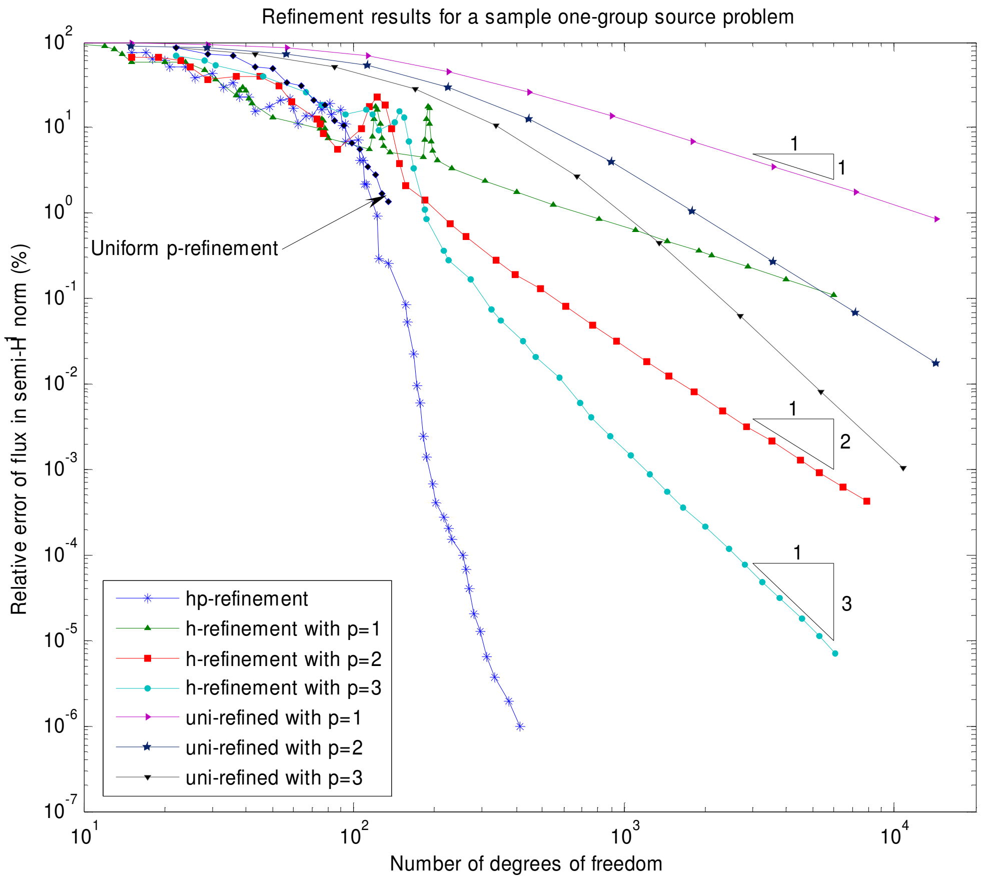


Fig. III-13. Convergence of different schemes

We then analyzed the effect of various shape functions on the performance of the *hp*-strategy. For tolerances greater than 10^{-5}%, the modified Peano shape functions give same results as Lobatto shape functions. However, the modified Peano functions didn't yield improve accuracy when the tolerance level was set below 10^{-5}% (see Fig. III-14). The higher performance of the Lobatto functions is easily understood due to their better behavior (smaller condition number for matrices resulting form the use of Lobatto shape functions).

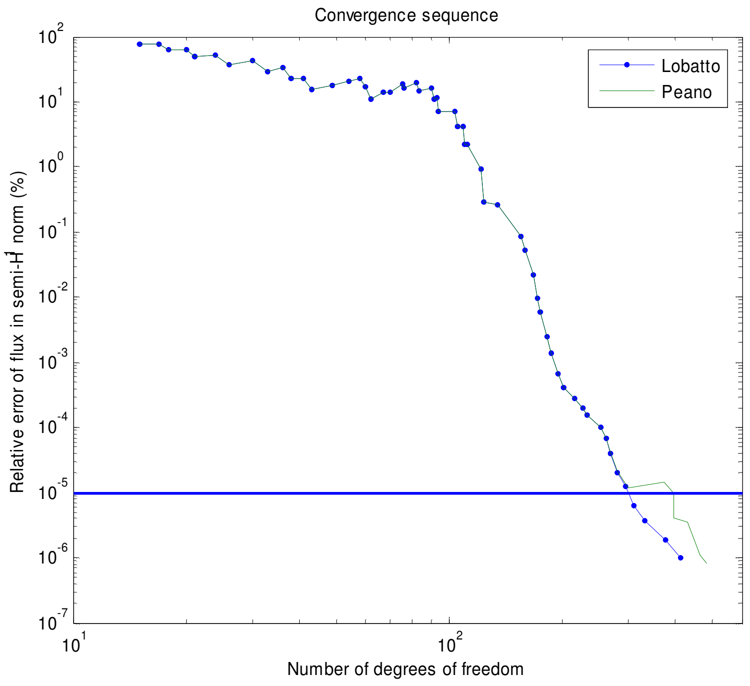


Fig. III-14. Convergence results of Lobatto and Peano shape functions

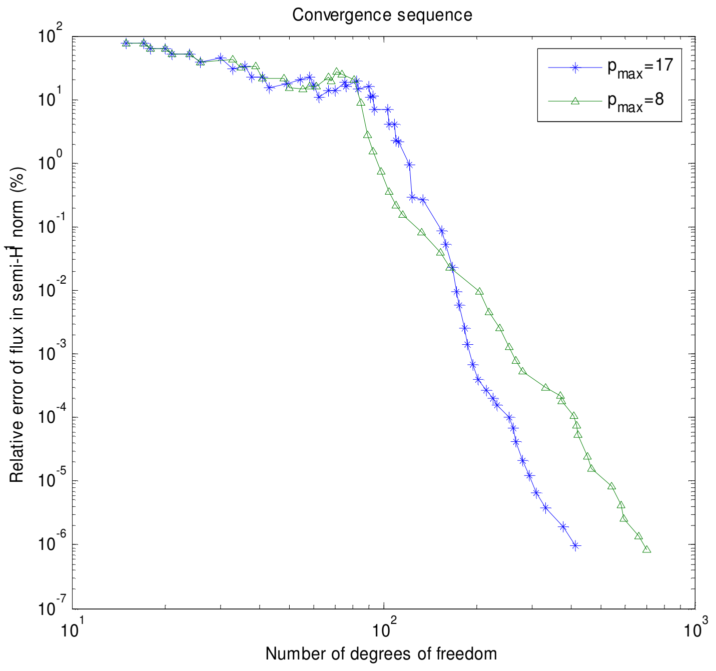


Fig. III-15. Convergence results with different maximum polynomial order

Then different maximum polynomial orders are tested. The results shown in Fig. III-15 prove that when the maximum polynomial order have been reached (and therefore, only *h*-refinements are conducted), the convergence is slightly slower. In other words, when we have to enforce *h*-refinement, more DOFs are needed to reach a prescribed tolerance in general and a small value for p_{max} may limit the exponential convergence of *hp*-schemes.

The influence of α_1 on convergence is shown in TABLE III-II. We can see that iteration numbers vary with respect to the value of α_1 and that only when α_1 is smaller than 1/40, it affects the final DOFs. We notice that different number of initial elements deliver different final meshes.

TABLE III-II

Influence of α_1 with different number of initial elements, (α_2=0.1)

α_1	7 initial elements			14 initial elements		
	Iteration number	Element number	Global DOFs	Iteration number	Element number	Global DOFs
1/2	77	59	302	53	83	433
1/3	**63**	**59**	**310**	**44**	**83**	**438**
1/4	58	59	303	38	83	442
1/5	55	59	307	34	83	447
1/6	46	59	309	29	83	432
1/8	48	60	328	28	83	442
1/10	45	60	331	26	83	433
1/20	39	59	309	24	83	446
1/40	30	60	319	19	83	453
1/80	29	64	344			
1/100	29	67	377	18	83	446
1/120	29	68	385			
1/1000				17	83	487

The influence of α_2 on convergence is shown in TABLE III-III. The bigger α_2 is, the smaller number of elements in the resulting mesh is ($\alpha_2 \leq 0.91$). When α_2 is equal to 1.0, the final mesh is independent of the choice of initial mesh. In our problem, we need higher *h*-refinement constraint to obtain a better mesh. α_2=2 could be a good choice. We will have to pay attention to this for multi-dimensional applications.

TABLE III-III

Influence of α_2 with different number of initial elements, (α_1=1/3)

α_2	7 initial elements				14 initial elements		
	Iteration number	Element number	Global DOFs		Iteration number	Element number	Global DOFs
0.01	63	59	310		44	83	438
0.05	63	59	310		44	83	438
0.1	63	59	310		44	83	438
0.2	63	59	310		44	83	438
0.4	63	59	310		44	83	438
0.5	63	57	298		43	79	398
0.8	64	50	283		45	67	363
0.9	**65**	**41**	**266**		**43**	**61**	**357**
0.91	63	40	262		45	56	337
0.95	63	40	264		44	46	295
0.99	67	34	266		41	35	271
1.0	69	28	267		46	28	267

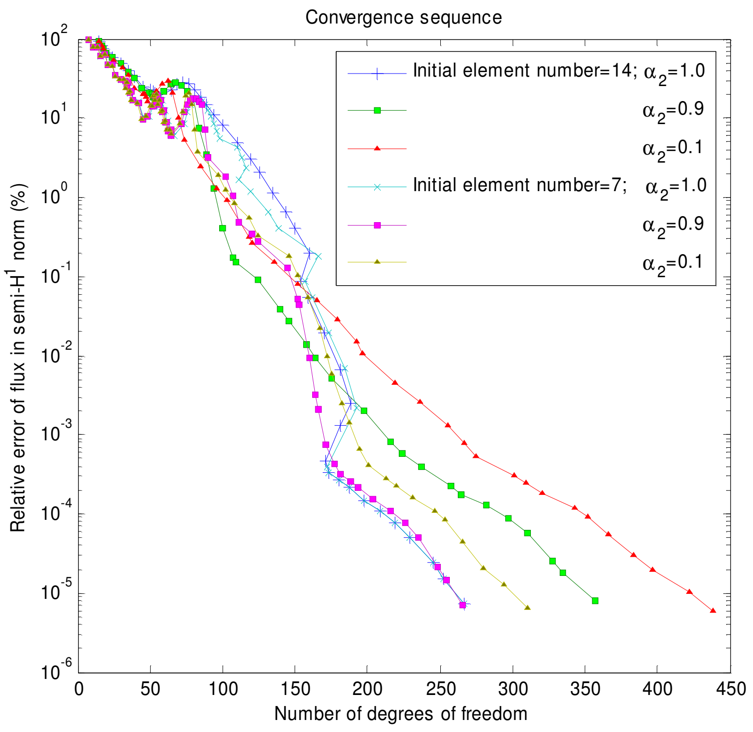
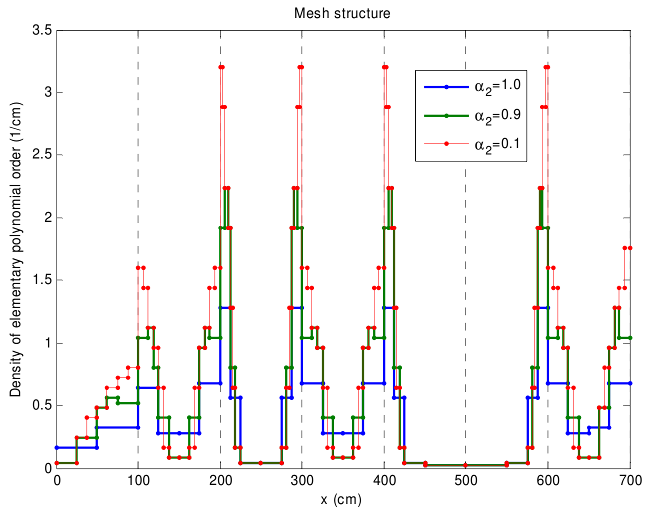
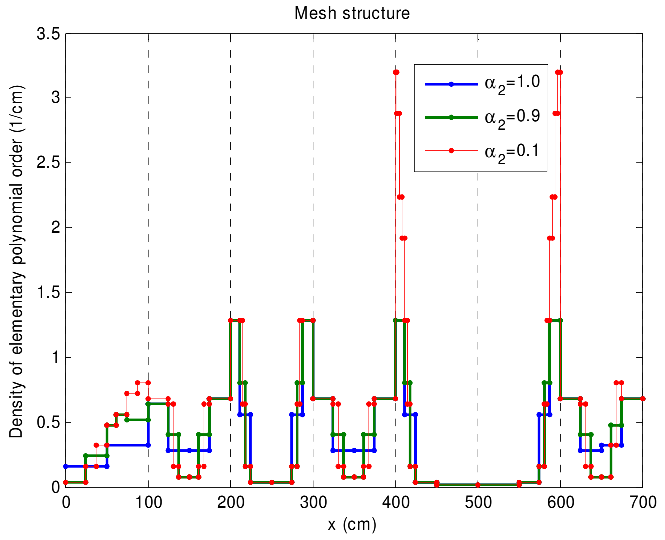


Fig. III-16. Influence on convergence sequence of *h*-constraint and initial mesh

The convergence sequences with different α_2 and different initial meshes are shown on Fig. III-16. In low accuracy range, *h*-refinement is the better choice. But for higher accuracy, *p*-refinement is better. The resulting meshes corresponding to the six curves in Fig. III-16 are plotted on Fig. III-17. This also explained Fig. III-15 why p_{max}=8 was better for lower accuracy.



(a)



(b)

Fig. III-17. Influence on resulting mesh of *h*-constraint and initial mesh distributions: (a) 7 initial elements and (b) 14 initial elements

Originally, α_2 =0.1 was used in the code, but it seems form this example that α_2=0.9 is a good balance over the overall error range.

We also used the density of the interpolation error as a refinement criterion instead of Eq. (3.19.a). The density of the interpolation error is defined as:

$$\frac{\mu_K}{L_K} > \alpha_1 \max_K \frac{\mu_K}{L_K} \tag{3.28}$$

, where L_K is the length of element K. The resulting convergence sequence is plotted in Fig. III-18. And the mesh, error distribution are plotted in Fig. III-19.

Regardless of the definition used for the refinement criterion (Eq. (3.19.a) or Eq. (3.28)), the convergence sequences are nearly identical. The final mesh tends to even distribute the error density.

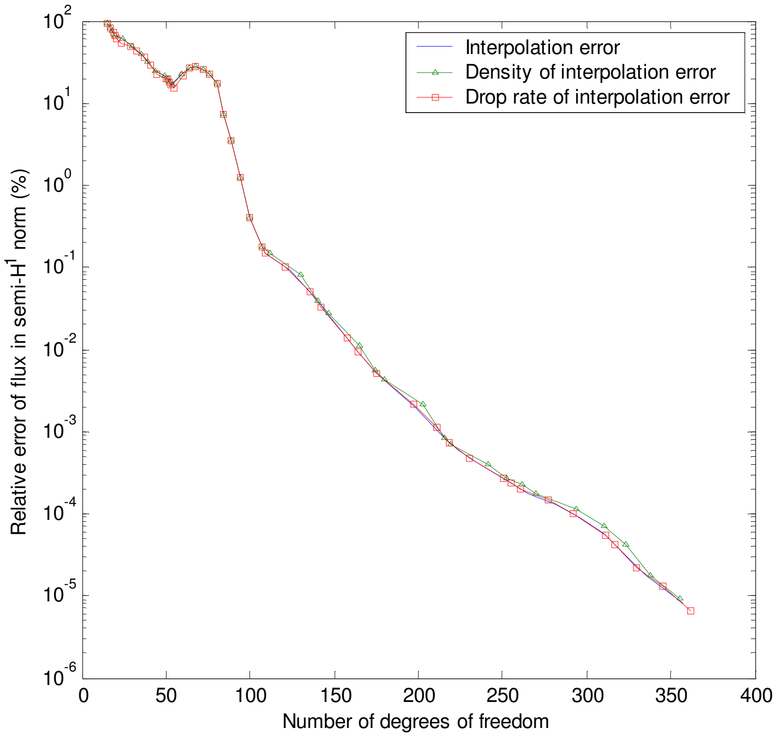
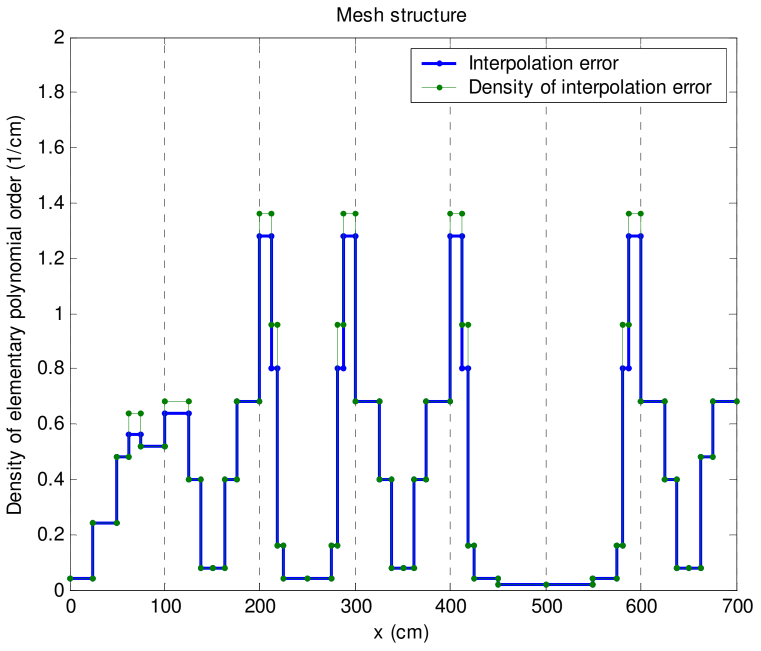
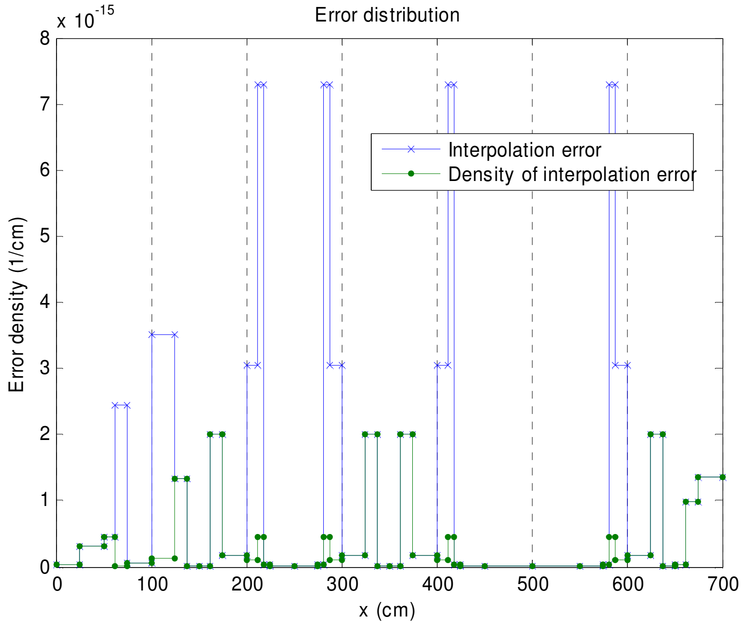


Fig. III-18. Convergence paths of using density of interpolation error as refinement criteria



(a)

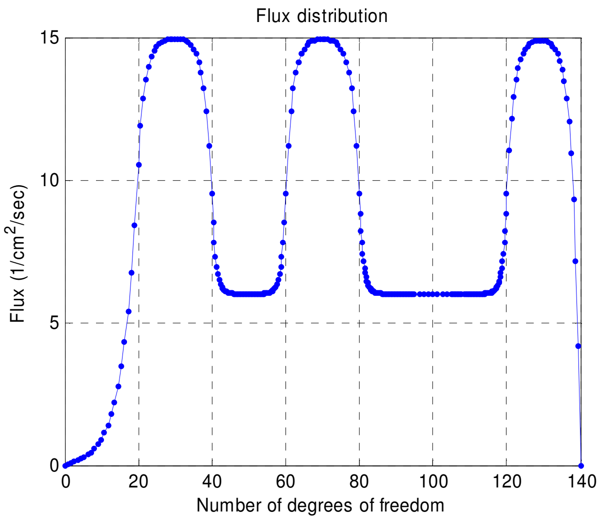


(b)

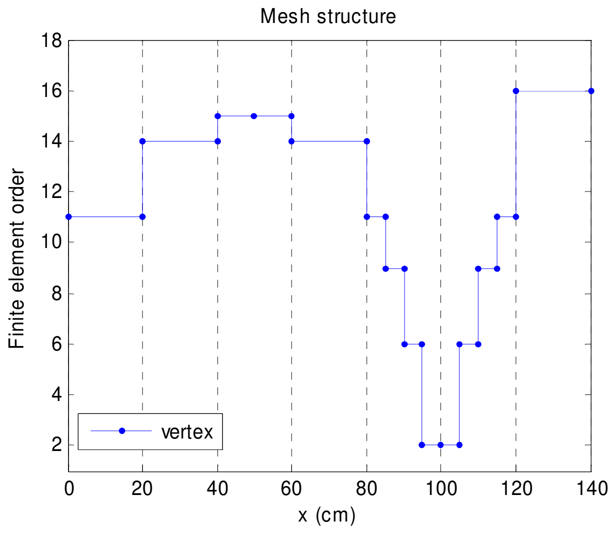
Fig. III-19. Results of using density of interpolation error as refinement criteria: (a) mesh structure, (b) local error distribution

3.6.2 Example 1.B (A 1-D one-group neutron diffusion source problem)

The data for this example is almost identical to the data for example 1.A except that the assembly size is reduced from 100cm to 20cm. Results are shown on Fig. III-20.



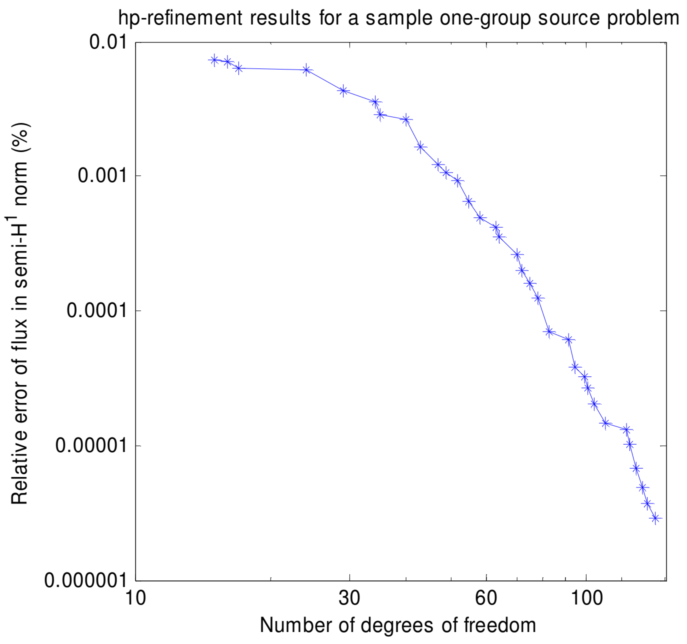
(a)



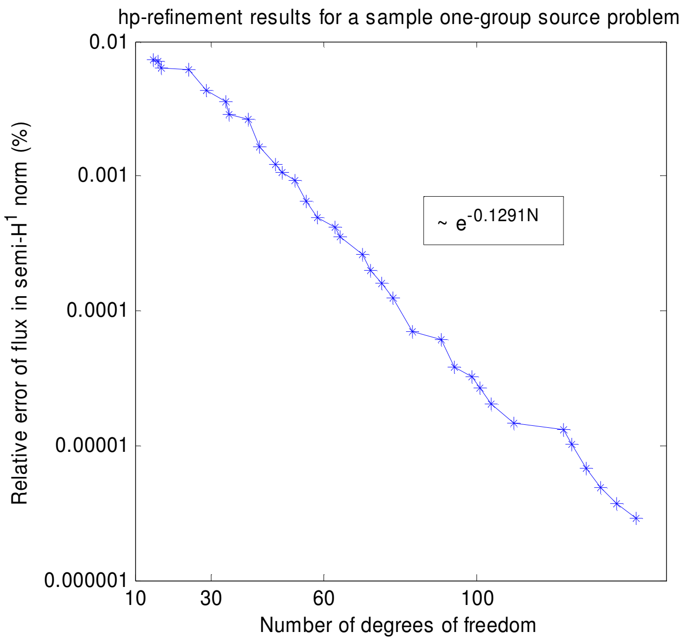
(b)

Fig. III-20. Solution of example 1-B: (a) flux and (b) mesh structure

This time *p*-refinement won all the time over *h*-refinement except between 80cm and 120cm. We noted that there were no enforced *h*-refinements due to the reach of $p_{\max}$.



(a)



(b)

Fig. III-21. Convergence properties of *hp*-adaptation, example 1-B: (a) log-log and (b) log-linear

Again, exponential convergence was attained, with a slope 0.1291, which is steeper than the one for example 1.A as seen in Fig. III-21.

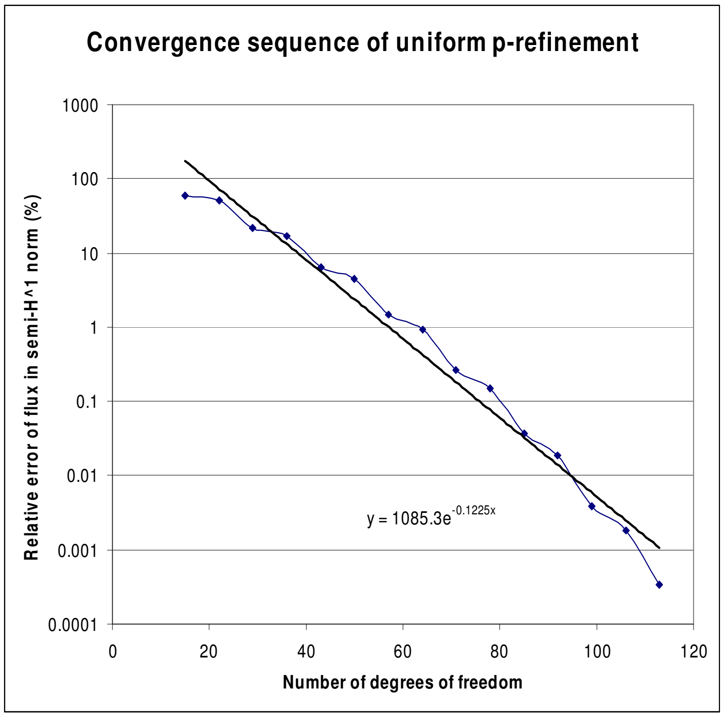


Fig. III-22. Convergence of uniform *p*-refinement

For comparison purposes, we tested uniform *p*-refinement again. The convergence path is in Fig. III-22. We can see that the convergence slope of uniform *p*-refinement is nearly same as *hp*-refinement. Anyway, the number of *hp*-refinement is smaller.

3.7 Conclusions of chapter III

This chapter introduced the concept of mesh adaptation, including the *h*-adaptive, *p*-adaptive and *hp*-adaptive strategies. We propose the following conclusions

1. *hp*-refinement converges exponentially for one-group neutron diffusion source problem;
2. *hp*-refinement can deliver the optimal mesh automatically starting from an extremely coarse initial mesh based on material compositions;

3. *hp*-refinement can deliver solution with extremely high accuracy using the smallest number of unknowns compared to other types of refinement schemes;
4. uniform *h*, and adaptive *h*-refinement yield algebraic convergence;
5. uniform *p*-refinement yields exponential convergence, but the convergence depends on the initial mesh and is slower than for *hp*-refinement;
6. Lobatto functions is a better choice of shape functions due to their lower condition number;
7. For high regularity problems, we need a relative high maximum polynomial order (we switch from 8 to 20) and a higher constraint on when to allow *h*-refinement to proceed;
8. The computational cost of the *hp*-projection itself will require attention for higher dimensions.

CHAPTER IV

MULTIGROUP ADAPTIVE HP-REFINEMENT STRATEGIES

This chapter presents the core of this thesis. After having analyzed the convergence properties of *hp*-refinement for one-group fixed source problems in the previous chapter, we tackle here the issues related to:

1. the coupling between groups due to scattering and fission events, and
2. the interaction of the mesh adaptation iterative procedure with the eigenvalue iterative solution (in the case of eigenvalue problems).

We note that it seems natural and legitimate to compute each multigroup flux with as little as DOFs as possible. For instance, it is well known that by nature, the fast flux in a thermal reactor is a significantly smoother function than the thermal flux. This rules out the idea of using a single mesh for all multigroup fluxes, as such a single mesh would be very far from being optimal for certain energy ranges. Consequently, each multigroup flux will be solved on its own (group-dependent) mesh. Therefore, we will have to consider how to treat the coupling terms such as

$$\Sigma_{s,g'\to g}\phi_{g'} \text{ or } \chi_g \nu\Sigma_{f,g'}\phi_{g'}$$

when $g' \neq g$.

Similarly, when computing a new iterate during the *n*+1-th power iteration, the fission integral term originates from the previous power iteration *n*. Since the mesh adaptation procedure can modify the multigroup meshes between the two power iterations, this will lead to a treatment similar to the one of group coupling (integration of functions of various orders defined on different meshes). Additionally, the question of where in the multigroup power iteration solver should the mesh adaptation procedure be plugged in will have to be analyzed.

We will analyze two main types of *hp*-refinement applied to the multigroup diffusion equations. The first one, already seen in Chapter III, deals with finding a numerical solution within a prescribed user tolerance everywhere in the computational domain. This is usually referred to as 'energy-driven' refinement [30] because the energy norm of the residual is minimized (we used the semi H^1 norm for simplicity; invoking Poincare inequality, the semi H^1 norm is equivalent to the energy norm). Engineering practice usually focuses on quantities of interest (a linear functional of the solution) in a sub-portion of the computational domain. Rather than finding an accurate solution everywhere in the domain, as it is the case for energy-driven refinement, it is advised to derive *hp*-refinement strategies that can accommodate these quantities of interest in a subdomain. This type of *hp*-refinement strategies, hereafter denoted by goal-oriented *hp*-strategies, will be investigated in the multigroup setting.

This chapter is organized as follows: first, we recall the multigroup direct and adjoint diffusion equations; secondly, we present energy-driven *hp*-refinement strategies in the multigroup setting; we then present goal-oriented *hp*-refinement strategies and propose extensive numerical verifications.

4.1 Multigroup diffusion equations and iteration solver

4.1.1 Multigroup diffusion equations

In reactor analysis, we usually solve the following steady state multi-group eigenvalue problem,

$$\begin{gathered} -\vec{\nabla}\cdot(D_g(\vec{r})\vec{\nabla}\phi_g(\vec{r}))+\Sigma_{r,g}(\vec{r})\phi_g(\vec{r})=\frac{1}{k_{eff}}\chi_g\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r})\phi_{g'}(\vec{r})+\sum_{g'\neq g}\Sigma_{s,g'\to g}(\vec{r})\phi_{g'}(\vec{r}) \\ g=1,2,\cdots,G \end{gathered} \tag{4.1}$$

where the diffusion coefficient D, the removal cross sections Σ_r, the fission cross sections $\nu\Sigma_f$, the scattering cross sections Σ_s and the fission spectrum χ are known material data. g is the energy group index and G is the total number of energy groups. We usually number the energy groups from the high energy range to the low energy range. The largest eigenvalue k_{eff} is called the neutron multiplication factor. The associated eigenvector is called the fundamental mode. Fission and scattering terms on the right hand side of Eq. (4.1) represent group couplings. The form of fission coupling term $\chi_g \nu\Sigma_{f,g'}(\vec{r})$ is different than the one of the scattering term $\Sigma_{s,g'\to g}(\vec{r})$, because it is assumed that the fission spectrum is independent of the incident neutron energy. There are no extraneous sources or inhomogeneous boundary conditions for eigenvalue problems.

Another class of problems consists of fixed source multigroup diffusion problems, whose equations are given below:

$$\begin{gathered}-\vec{\nabla}\cdot(D_g(\vec{r})\vec{\nabla}\phi_g(\vec{r}))+\Sigma_{r,g}(\vec{r})\phi_g(\vec{r})=\chi_g\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r})\phi_{g'}(\vec{r})+\sum_{g'\neq g}\Sigma_{s,g'\to g}(\vec{r})\phi_{g'}(\vec{r})+s_{ext,g}(\vec{r}) \\ g=1,2,\cdots,G\end{gathered} \tag{4.2}$$

In order to have non-trivial (i.e., nonzero) stable solution, the system must be sub-critical and source terms must present either in the form of an extraneous volumetric source $s_{ext,g}$ or a boundary source.

In our 1-D study, these multigroup equations are written as:

$$\begin{gathered}-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g(x))+\Sigma_{r,g}(x)\phi_g(x)=\frac{1}{k_{eff}}\chi_g\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(x)\phi_{g'}(x)+\sum_{g'\neq g}\Sigma_{s,g'\to g}(x)\phi_{g'}(x) \\ g=1,2,\cdots,G\end{gathered} \tag{4.3}$$

$$\begin{gathered}-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g(x))+\Sigma_{r,g}(x)\phi_g(x)=\chi_g\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(x)\phi_{g'}(x)+\sum_{g'\neq g}\Sigma_{s,g'\to g}(x)\phi_{g'}(x)+s_{ext,g}(x) \\ g=1,2,\cdots,G\end{gathered} \tag{4.4}$$

with appropriate boundary conditions for all energy groups. We usually define group fission source terms and in-group scattering terms as,

$$\begin{aligned}P_g(x)&=\chi_g\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(x)\phi_{g'}(x)=\chi_g F(x)\\ H_g(x)&=\sum_{g'\neq g}\Sigma_{s,g'\to g}(x)\phi_{g'}(x)\end{aligned} \tag{4.5}$$

$F(x)$ is the total fission neutron source. For convenience of derivation, we want to write multigroup equation in operator form. So, we define the following operators,

$$L = \begin{bmatrix} -\vec{\nabla}\cdot(D_1\vec{\nabla}) + \Sigma_{r,1} & & & o \\ & -\vec{\nabla}\cdot(D_2\vec{\nabla}) + \Sigma_{r,2} & & \\ & & \ddots & \\ o & & & -\vec{\nabla}\cdot(D_G\vec{\nabla}) + \Sigma_{r,G} \end{bmatrix}$$

$$H = \begin{bmatrix} 0 & \Sigma_{s,2\to1} & \Sigma_{s,3\to1} & \cdots & \Sigma_{s,G\to1} \\ \Sigma_{s,1\to2} & 0 & \Sigma_{s,3\to2} & \cdots & \Sigma_{s,G\to2} \\ \Sigma_{s,1\to3} & \Sigma_{s,2\to3} & 0 & \cdots & \Sigma_{s,G\to3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma_{s,1\to G} & \Sigma_{s,2\to G} & \Sigma_{s,3\to G} & \cdots & 0 \end{bmatrix} \tag{4.6.a}$$

$$P = X^T F = \begin{bmatrix} \chi_1 \\ \chi_2 \\ \vdots \\ \chi_G \end{bmatrix} \begin{bmatrix} \nu\Sigma_{f,1} & \nu\Sigma_{f,2} & \cdots & \nu\Sigma_{f,G} \end{bmatrix}$$

L is loss operator including neutron net leakage out of the system and removal from the group (either by absorption or out-scattering). *H* is the scattering matrix. Its diagonal entries are zeroes (because the within-group scattering has already been accounted for in the removal cross sections) The upper triangular part of *H* corresponds to up-scattering terms and the lower triangular part corresponds to down-scattering terms. Up-scattering occurs when an incident neutron can gain kinetic energy during a collision. This phenomenon can only occur in the thermal energy range where the thermal movement of the target nuclei can no longer be neglected compared to the neutron kinetic energy. *P* is fission operator, which is the product of the fission spectrum operator *X* and the fission cross section operator *F*. We define the flux and source vectors as:

$$\begin{aligned} \phi &= \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_G \end{bmatrix}^T \\ s_{ext} &= \begin{bmatrix} s_{ext,1} & s_{ext,2} & \cdots & s_{ext,G} \end{bmatrix}^T \end{aligned} \tag{4.6.b}$$

The multigroup equations can be recast as,

$$\begin{aligned} L\phi &= \frac{1}{k_{eff}} P\phi + H\phi \quad \text{(eigenproblem)} \\ L\phi &= P\phi + H\phi + s_{ext} \quad \text{(fixed source problem)} \end{aligned} \tag{4.6.c}$$

Their adjoint equations are:

$$\begin{aligned} L^*\phi^* &= \frac{1}{k_{eff}} P^*\phi^* + H^*\phi^* \\ L^*\phi^* &= P^*\phi^* + H^*\phi^* + s^*_{ext} \end{aligned} \tag{4.7.a}$$

*L**, *P**, *H** are the corresponding adjoint operators. The physical meaning of adjoint flux is that of neutron importance. An example of an adjoint source is typically a detector response. Direct and adjoint eigenvalue systems have the same eigenvalue. In the adjoint equations, the adjoint flux and adjoint source vectors are:

$$\begin{aligned} \phi^* &= \begin{bmatrix} \phi^*_1 & \phi^*_2 & \cdots & \phi^*_G \end{bmatrix}^T \\ s^*_{ext} &= \begin{bmatrix} s^*_{ext,1} & s^*_{ext,2} & \cdots & s^*_{ext,G} \end{bmatrix}^T \end{aligned} \tag{4.7.b}$$

It is easy to prove that,

$$L^* = L = \begin{bmatrix} -\vec{\nabla}\cdot(D_1\vec{\nabla}) + \Sigma_{r,1} & & & o \\ & -\vec{\nabla}\cdot(D_2\vec{\nabla}) + \Sigma_{r,2} & & \\ & & \ddots & \\ o & & & -\vec{\nabla}\cdot(D_G\vec{\nabla}) + \Sigma_{r,G} \end{bmatrix}$$

$$H^* = \begin{bmatrix} 0 & \Sigma_{s,1\to 2} & \Sigma_{s,1\to 3} & \cdots & \Sigma_{s,1\to G} \\ \Sigma_{s,2\to 1} & 0 & \Sigma_{s,2\to 3} & \cdots & \Sigma_{s,2\to G} \\ \Sigma_{s,3\to 1} & \Sigma_{s,3\to 2} & 0 & \cdots & \Sigma_{s,3\to G} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma_{s,G\to 1} & \Sigma_{s,G\to 2} & \Sigma_{s,G\to 3} & \cdots & 0 \end{bmatrix} \tag{4.7.c}$$

$$P^* = F^T X = \begin{bmatrix} \nu\Sigma_{f,1} \\ \nu\Sigma_{f,2} \\ \vdots \\ \nu\Sigma_{f,G} \end{bmatrix} \begin{bmatrix} \chi_1 & \chi_2 & \cdots & \chi_G \end{bmatrix}$$

Note that the loss operator is self-adjoint and symmetric positive definite.

Explicitly, the adjoint equations are:

$$\begin{aligned} -\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi^*_g(x)) + \Sigma_{r,g}(x)\phi^*_g(x) &= \frac{1}{k_{eff}} \nu\Sigma_{f,g}(x)\sum_{g'=1}^{G} \chi_{g'}\phi^*_{g'}(x) + \sum_{g'\neq g} \Sigma_{s,g\to g'}(x)\phi^*_{g'}(x) \\ g &= 1,2,\cdots,G \end{aligned} \tag{4.8}$$

$$-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g^*(x))+\Sigma_{r,g}(x)\phi_g^*(x)=\nu\Sigma_{f,g}(x)\sum_{g'=1}^{G}\chi_{g'}\phi_{g'}^*(x)+\sum_{g'\neq g}\Sigma_{s,g\to g'}(x)\phi_{g'}^*(x)+s_{ext,g}^*(x)$$
$$g=1,2,\cdots,G \tag{4.9}$$

4.1.2 Variational form of multi-group equations

The variational form of multi-group source problem is,

$$\begin{aligned}&\phi(x)\in\phi_D(x)+H_0^1\\&b(\phi,\phi^*)=l(\phi^*),\quad\forall\phi^*\in H_0^1\end{aligned} \tag{4.10.a}$$

$\phi_D(x)$ is a lift due to non-homogenous Dirichlet boundary conditions.

$$l(\phi^*)=(s_{ext},\phi^*)+\begin{cases}\sum_{g=1}^{G}2J_{g,in}(x_b)\phi_g^*(x_b), & \text{Cauchy B.C.}\\ \sum_{g=1}^{G}J_g(x_b)\phi_g^*(x_b), & \text{Neumann B.C.}\end{cases} \tag{4.10.b}$$

$$(s_{ext},\phi^*)=\sum_{g=1}^{G}\int_\Omega s_{ext,g}(x)\phi_g^*(x)dx \tag{4.10.c}$$

x_b is the coordinate of the boundary where a Cauchy or Neumann condition holds. $J_g(x_b)$ is net incoming current at x_b.

There are three terms included in the bilinear form for a source problem, (loss+scattering+fission, assuming there is no group coupling on boundary conditions)

$$
\begin{aligned}
&b(\phi,\phi^*) = (L\phi,\phi^*) - (X\phi,\phi^*) - (F\phi,\phi^*) \\
&\phi^{*T} L\phi = \begin{bmatrix}\phi_1^* & \phi_2^* & \cdots & \phi_G^*\end{bmatrix}\begin{bmatrix}-\nabla\cdot(D_1\nabla)+\Sigma_{r,1} & & & o \\ & -\nabla\cdot(D_2\nabla)+\Sigma_{r,2} & & \\ & & \ddots & \\ o & & & -\nabla\cdot(D_G\nabla)+\Sigma_{r,G}\end{bmatrix}\begin{bmatrix}\phi_1\\ \phi_2\\ \vdots\\ \phi_G\end{bmatrix} \\
&(L\phi,\phi^*) = \sum_{g=1}^{G}\left[\int_\Omega [D_g\nabla\phi_g^*\cdot\nabla\phi_g + \Sigma_{r,g}\phi_g^*\phi_g]dx + \frac{1}{2}\phi_g^*(x_b)\phi_g(x_b)\right] \\
&\phi^{*T} H\phi = \begin{bmatrix}\phi_1^* & \phi_2^* & \cdots & \phi_G^*\end{bmatrix}\begin{bmatrix}0 & \Sigma_{s,1\to2} & \Sigma_{s,1\to3} & \cdots & \Sigma_{s,1\to G}\\ \Sigma_{s,2\to1} & 0 & \Sigma_{s,2\to3} & \cdots & \Sigma_{s,2\to G}\\ \Sigma_{s,3\to1} & \Sigma_{s,3\to2} & 0 & \cdots & \Sigma_{s,3\to G}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \Sigma_{s,G\to1} & \Sigma_{s,G\to2} & \Sigma_{s,G\to3} & \cdots & 0\end{bmatrix}\begin{bmatrix}\phi_1\\ \phi_2\\ \vdots\\ \phi_G\end{bmatrix} \qquad (4.10.d) \\
&(H\phi,\phi^*) = \sum_{g=1}^{G}\sum_{g'\neq g}\int_\Omega \Sigma_{s,g\to g'}\phi_{g'}^*\phi_g dx \\
&\phi^{*T} P\phi = \begin{bmatrix}\phi_1^* & \phi_2^* & \cdots & \phi_G^*\end{bmatrix}\begin{bmatrix}\chi_1\\ \chi_2\\ \vdots\\ \chi_G\end{bmatrix}\begin{bmatrix}v\Sigma_{f,1} & v\Sigma_{f,2} & \cdots & v\Sigma_{f,G}\end{bmatrix}\begin{bmatrix}\phi_1\\ \phi_2\\ \vdots\\ \phi_G\end{bmatrix} \\
&(P\phi,\phi^*) = \sum_{g=1}^{G}\sum_{g'=1}^{G}\int_\Omega \chi_{g'} v\Sigma_{f,g}\phi_{g'}^*\phi_g dx = \int_\Omega\left[\sum_{g=1}^{G} v\Sigma_{f,g}\phi_g\right]\left[\sum_{g=1}^{G}\chi_g\phi_g^*\right]dx = \int_\Omega F(x)F^*(x)dx
\end{aligned}
$$

$F^*(x)$ is the adjoint fission source. The second term in the summation of $(L\phi,\phi^*)$ arises from the Cauchy boundary conditions.

Note that:

$$(L\phi,\phi^*) = \phi^T L^*\phi^* = \phi^{*T}L\phi,\ \ (P\phi,\phi^*) = \phi^T P^*\phi^* = \phi^{*T}P\phi,\ \ (H\phi,\phi^*) = \phi^T H^*\phi^* = \phi^{*T}H\phi$$

(ϕ^* does not have to be the solution of adjoint problem.)

The variational form of an eigenvalue problem is,

$$
\begin{aligned}
&\phi(x)\in H_0^1 \\
&b(\phi,\phi^*) = \frac{1}{k_{eff}}(P\phi,\phi^*),\ \ \forall\phi^*\in H_0^1
\end{aligned} \qquad (4.11)
$$

Here, there only two terms are present in the bilinear form (the loss and scattering terms).

The variational form of adjoint source problem is,

$$\begin{aligned} &\phi^*(x) \in \phi_D^*(x) + H_0^1 \\ &b(\phi,\phi^*) = l^*(\phi), \quad \forall \phi \in H_0^1 \end{aligned} \tag{4.12.a}$$

where,

$$l^*(\phi) = (s_{ext}^*,\phi) - \begin{cases} \sum_{g=1}^{G} 2J_{g,out}^*(x_b)\phi_g(x_b), & \text{Cauchy B.C.} \\ \sum_{g=1}^{G} J_g^*(x_b)\phi_g(x_b), & \text{Neumann B.C.} \end{cases} \tag{4.12.b}$$

with $\quad (s_{ext}^*,\phi) = \sum_{g=1}^{G} \int_\Omega s_{ext,g}^*(x)\phi_g(x)dx$

$J_g^*(x_b) = \left[D_g \frac{d\phi_g^*}{dx} \right]_{x=x_b}$; $\quad J_{g,out}^*(x_b) = \frac{1}{2}\left[\frac{\phi_g^*}{2} + D_g \frac{d\phi_g^*}{dx} \right]_{x=x_b}$ when x_b is the right boundary (4.10.c)

$J_g^*(x_b) = -\left[D_g \frac{d\phi_g^*}{dx} \right]_{x=x_b}$; $\quad J_{g,out}^*(x_b) = \frac{1}{2}\left[\frac{\phi_g^*}{2} - D_g \frac{d\phi_g^*}{dx} \right]_{x=x_b}$ when x_b is the left boundary

$J_g^*(x_b)$ is the net outgoing adjoint current at the boundary. $J_{g,out}^*(x_b)$ is the partial outgoing adjoint current. Note that the adjoint Fick's law is: $\vec{J}_g^*(\mathbf{x}) = D_g(\mathbf{x})\vec{\nabla}\phi_g^*(\mathbf{x})$.

The variational form of the adjoint eigenvalue problem is,

$$\begin{aligned} &\phi^*(x) \in H_0^1 \\ &b(\phi,\phi^*) = \frac{1}{k_{eff}}(F\phi,\phi^*), \quad \forall \phi \in H_0^1 \end{aligned} \tag{4.13}$$

Again, there are only two terms in the bilinear form for the adjoint eigenvalue problem (loss+scattering).

Note that: $b(\phi,\phi^*)$ is symmetric only for one group problem. In general, we have $b(\phi,\phi^*) \neq b(\phi^*,\phi)$ due to the scattering and fission operators.

After applying Galerkin's method to the multigroup equations, we obtain a system of linear equations, which have same form as equation 3.6.c and equation 3.7.a. But now, the operators *L*, *P*, *H* and L^*, P^*, H^* are block matrices and ϕ, ϕ^*, s_{ext}, s_{ext}^* are block vectors, where a 'block' represents an energy group. We have: $L^*=L$, $P^*=P^T$ and $H^*=H^T$.

4.1.3 Description of the iteration solver for the multigroup diffusion equations

Now let us consider some specialized forms of the multigroup equations for completeness:

4.1.3.1 Source problem without up-scattering and without fission

$$\begin{gathered}
-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g(x))+\Sigma_{r,g}(x)\phi_g(x)=\sum_{g'<g}\Sigma_{s,g'\to g}(x)\phi_{g'}(x)+s_{ext,g}(x)\\
g=1,2,\cdots,G\\
-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g^*(x))+\Sigma_{r,g}(x)\phi_g^*(x)=\sum_{g'>g}\Sigma_{s,g\to g'}(x)\phi_{g'}^*(x)+s_{ext,g}^*(x)\\
g=1,2,\cdots,G
\end{gathered}\tag{4.14}$$

We can assembly and solve source problem from group 1 to group *G*, group by group with solely the loss operator on the left-hand-side. The solution is attained in one group sweep. The solution of adjoint problem is similar; with the only difference is that the group sweep is performed in reverse order, from group *G* to 1. Since lost operator *L* is ***SPD*** (symmetric positive definite), it is generally much easier to solve this set of equations group-by-group rather than solving the whole multigroup system at once.

4.1.3.2 Source problem with up-scattering and without fission

At this time, we still keep the group sweep process outlined above, but introduce an iteration for solving a series one group source problem in the thermal energy range (where up-scattering occurs). With an initial flux guess given by

$$\phi^{(0)}=\begin{bmatrix}\phi_1^{(0)} & \phi_2^{(0)} & \cdots & \phi_G^{(0)}\end{bmatrix}^T,$$

which, for example, could be zero, we can update the fluxes from group 1 to group *G* with the up-scattering terms calculated from the previous iteration or initial guess.

$$\begin{gathered}
-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g^{(n)}(x))+\Sigma_{r,g}(x)\phi_g^{(n)}(x)=\\
\sum_{g'<g}\Sigma_{s,g'\to g}(x)\phi_{g'}^{(n)}(x)+\sum_{g'>g}\Sigma_{s,g'\to g}(x)\phi_{g'}^{(n-1)}(x)+s_{ext,g}(x)\\
g=1,2,\cdots,G
\end{gathered}\tag{4.15}$$

$$-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g^{*(n)}(x))+\Sigma_{r,g}(x)\phi_g^{*(n)}(x)=$$
$$\sum_{g'>g}\Sigma_{s,g\to g'}(x)\phi_{g'}^{*(n)}(x)+\sum_{g'<g}\Sigma_{s,g\to g'}(x)\phi_{g'}^{*(n-1)}(x)+s_{ext,g}^{*}(x)$$
$$g=1,2,\cdots,G$$

Once the new fluxes $\phi_g^{(n)}(x)$ are calculated, they will be used to construct the down-scattering term on the right hand side immediately. Obviously one group sweep is no longer enough and we need to iterate this process to converge the multigroup fluxes. This process is usually referred to as *thermal iterations*. Note that during one thermal iteration, we do not have to backup the previous fluxes. However, we may still want to do so for the comparison between two successive iterates to determine whether the thermal iteration process has converged.

Up-scattering occurs at thermal energy range where the kinetic energy of surrounding atoms is on the same order as the energy of incident neutrons in reality. More precisely, the scattering matrix appearing in equation 3.6.a has the following form

$$X=\begin{bmatrix} 0 & & & & & \\ \Sigma_{s,1\to 2} & \ddots & & & 0 & \\ \vdots & & & & & \\ \Sigma_{s,1\to U} & \cdots & 0 & \Sigma_{s,U+1\to U} & \cdots & \Sigma_{s,G\to U} \\ \Sigma_{s,1\to U+1} & \cdots & \Sigma_{s,U\to U+1} & 0 & \cdots & \Sigma_{s,G\to U+1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma_{s,1\to G} & \cdots & \Sigma_{s,U\to G} & \Sigma_{s,U+1\to G} & \cdots & 0 \end{bmatrix},$$

, where U is the first thermal energy group, i.e., the first energy group for which up-scattering occurs. Now, we see that the whole problem can de decomposed into two steps: first, we solve the fast region domain (from energy group 1 to U-1); then, we proceed with the thermal iterations to solve energy groups U to G. It is similar to solve adjoint problem, except that the groups are swept in reverse order.

4.1.3.3 Source problem with fission present

In this case, we can manipulate the fission source terms and embed them with the scattering terms, which is the more conventional way to proceed for fixed-source problems. But in our code, we use the following scheme,

$$\begin{aligned}
&-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g^{(m)}(x))+\Sigma_{r,g}(x)\phi_g^{(m)}(x)= \\
&\qquad \chi_g\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(x)\phi_{g'}^{(m-1)}(x)+\sum_{g'<g}\Sigma_{s,g'\to g}(x)\phi_{g'}^{(m)}(x)+\sum_{g'>g}\Sigma_{s,g'\to g}(x)\phi_{g'}^{(m)}(x)+s_{ext,g}(x) \\
&\qquad g=1,2,\cdots,G
\end{aligned} \tag{4.16}$$

$$\begin{aligned}
&-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g^{*(m)}(x))+\Sigma_{r,g}(x)\phi_g^{*(m)}(x)= \\
&\qquad \nu\Sigma_{f,g}(x)\sum_{g'=1}^{G}\chi_{g'}\phi_{g'}^{*(m-1)}(x)+\sum_{g'>g}\Sigma_{s,g\to g'}(x)\phi_{g'}^{*(m)}(x)+\sum_{g'<g}\Sigma_{s,g\to g'}(x)\phi_{g'}^{*(m)}(x)+s_{ext,g}^{*}(x) \\
&\qquad g=1,2,\cdots,G
\end{aligned}$$

In this scheme, we have an additional iteration to renew the fission source (similarly to an *outer iteration* or a *power iteration* for eigenproblems; this will be removed in the future to treat fixed source problems and eigenproblems differently and in a fashion that it the most appropriate for each of these problems). In lots of cases especially for fewer group calculations, we can just do inner or thermal iteration once per power iteration. At this time, iteration scheme is as following,

$$\begin{aligned}
&-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g^{(n)}(x))+\Sigma_{r,g}(x)\phi_g^{(n)}(x)= \\
&\qquad \chi_g\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(x)\phi_{g'}^{(n-1)}(x)+\sum_{g'<g}\Sigma_{s,g'\to g}(x)\phi_{g'}^{(n)}(x)+\sum_{g'>g}\Sigma_{s,g'\to g}(x)\phi_{g'}^{(n-1)}(x)+s_{ext,g}(x) \\
&\qquad g=1,2,\cdots,G \\
&-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g^{*(n)}(x))+\Sigma_{r,g}(x)\phi_g^{*(n)}(x)= \\
&\qquad \nu\Sigma_{f,g}(x)\sum_{g'=1}^{G}\chi_{g'}\phi_{g'}^{*(n-1)}(x)+\sum_{g'>g}\Sigma_{s,g\to g'}(x)\phi_{g'}^{*(n)}(x)+\sum_{g'<g}\Sigma_{s,g\to g'}(x)\phi_{g'}^{*(n-1)}(x)+s_{ext,g}^{*}(x) \\
&\qquad g=1,2,\cdots,G
\end{aligned} \tag{4.17}$$

We need to store the multigroup fluxes or more exactly store director adjoint fission source for each outer iteration. The stored fission source can serve as our convergence criteria

4.1.3.4 Eigenvalue problem

The solution of eigenvalue problem is similar to the source problem with fission we described above. The differences are that we need to update the eigenvalue (*k*-effective) and normalize the flux at each power iteration. The principle is as follows:

Solve the direct problem from group 1 to *G* first,

$$-\vec{\nabla}\cdot(D_g(x)\vec{\nabla}\phi_g^{(n)}(x))+\Sigma_{r,g}(x)\phi_g^{(n)}(x)=$$
$$\chi_g F^{(n-1)}+\sum_{g'<g}\Sigma_{s,g'\to g}(x)\phi_{g'}^{(n)}(x)+\sum_{g'>g}\Sigma_{s,g'\to g}(x)\phi_{g'}^{(n-1)}(x) \qquad (4.18.a)$$
$$g=1,2,\cdots,G$$

Then update k_{eff} and fission source using

$$k_{eff}^{(n)}=\int_\Omega\left[\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(x)\phi_{g'}^{(n)}(x)\right]dx$$
$$F^{(n)}=\frac{1}{k_{eff}^{(n)}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(x)\phi_{g'}^{(n)}(x) \qquad (4.18.b)$$

This automatically normalizes flux with total fission neutron source equal to 1.

The adjoint eigenproblem is solved in a similar fashion:

$$-\nabla\cdot(D_g(x)\nabla\phi_g^{*(n)}(x))+\Sigma_{r,g}(x)\phi_g^{*(n)}(x)$$
$$=\nu\Sigma_{f,g}(x)F^{*(n-1)}+\sum_{g'>g}\Sigma_{s,g\to g'}(x)\phi_{g'}^{*(n)}(x)+\sum_{g'<g}\Sigma_{s,g\to g'}(x)\phi_{g'}^{*(n-1)}(x)$$
$$g=1,2,\cdots,G \qquad (4.18.c)$$
$$F^{*(n)}=\frac{1}{k_{eff}}\sum_{g'=1}^{G}\chi_{g'}\phi_{g'}^{*(n)}(x)$$

Note that equations 3.18.c are solved from group G to 1. Finally we normalize adjoint flux with

$$(F\phi,\phi^*)=\int_\Omega\phi^*F\phi dx=\int_\Omega F^*(x)F(x)dx=1 \qquad (4.18.d)$$

Other normalizations are possible.

The schematics of power iteration and flux iteration are shown on Fig. IV-1. We need to store the solutions for each power iteration and flux iteration separately to determine their respective convergence. If we only perform a flux iteration once, the schematics is simplified as shown in Fig. IV-2. For all the cases, we have a unique source construction module illustrated in Fig. IV-3.

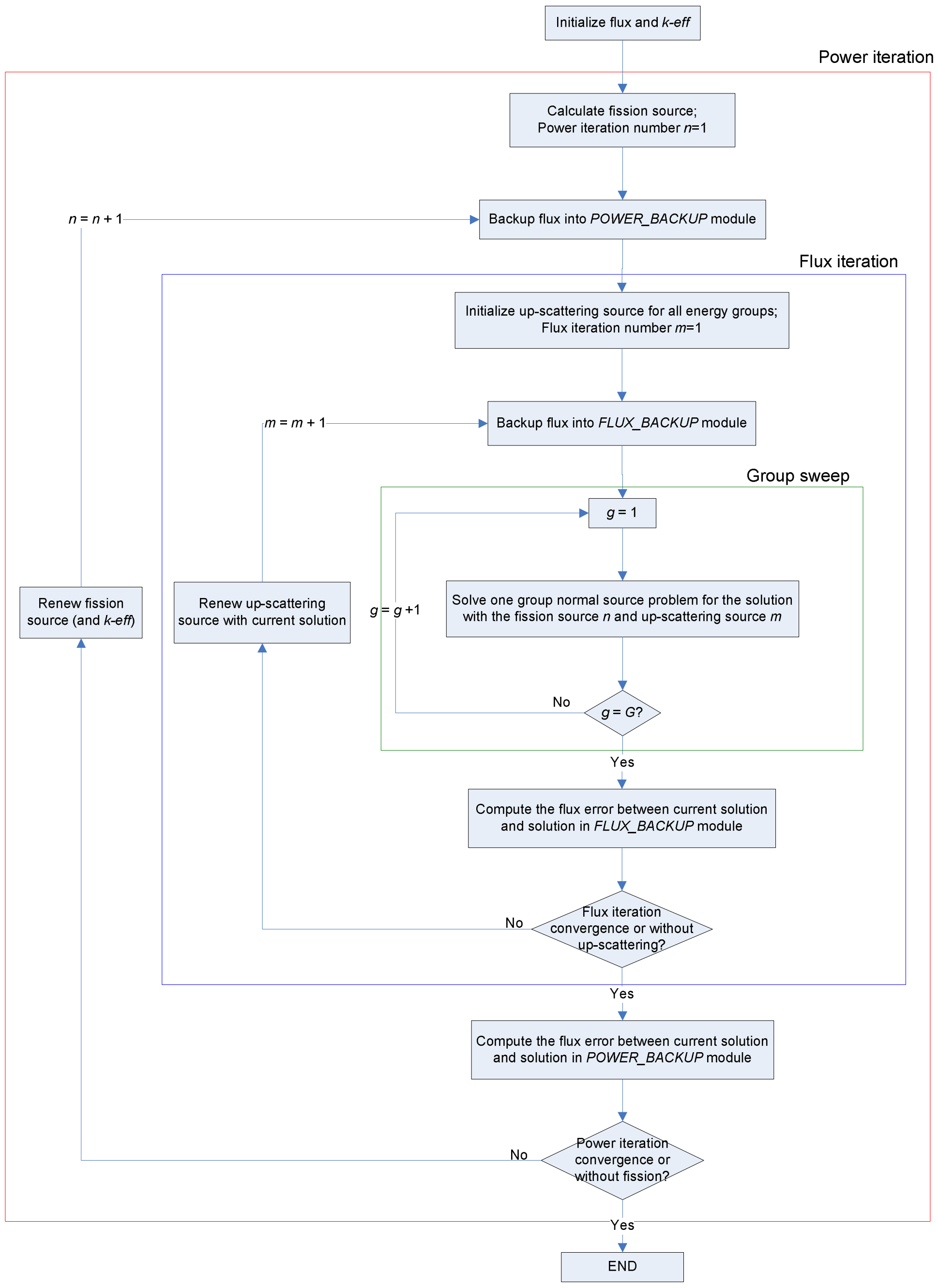
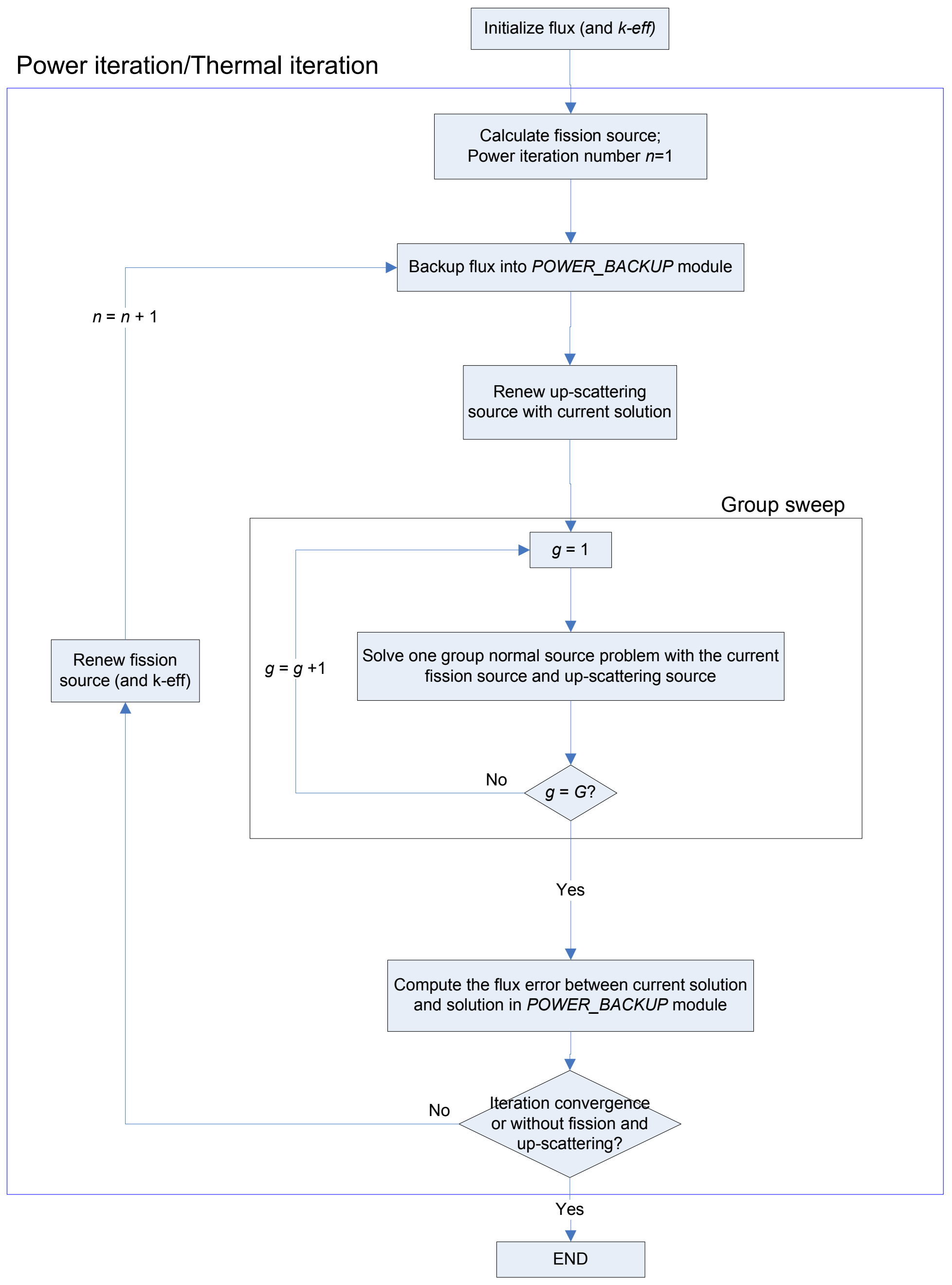
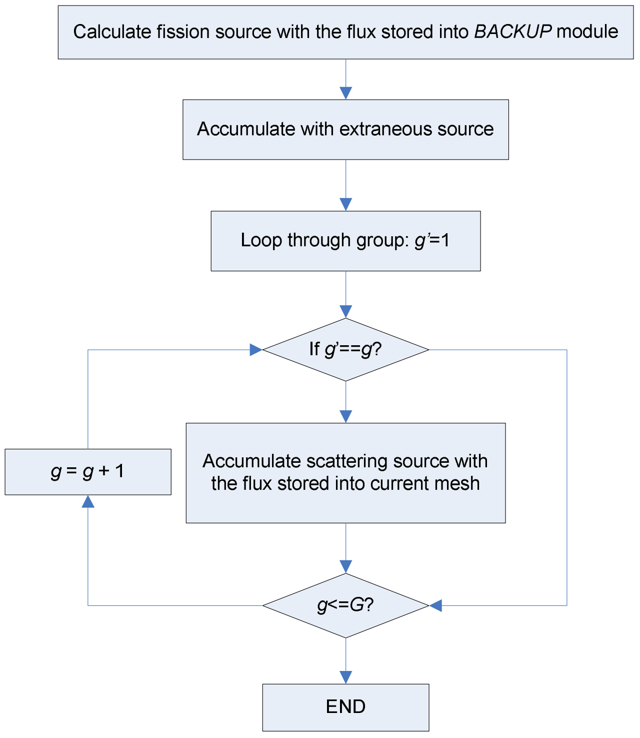

Initialize flux and *k-eff*
Power iteration
Calculate fission source; Power iteration number *n*=1
n = *n* + 1
Backup flux into *POWER_BACKUP* module
Flux iteration
Initialize up-scattering source for all energy groups; Flux iteration number *m*=1
m = *m* + 1
Backup flux into *FLUX_BACKUP* module
Group sweep
g = 1
g = *g* +1
Renew fission source (and *k-eff*)
Renew up-scattering source with current solution
Solve one group normal source problem for the solution with the fission source *n* and up-scattering source *m*
No
g = *G*?
Yes
Compute the flux error between current solution and solution in *FLUX_BACKUP* module
No
Flux iteration convergence or without up-scattering?
Yes
Compute the flux error between current solution and solution in *POWER_BACKUP* module
No
Power iteration convergence or without fission?
Yes
END

Fig. IV-1. General multigroup iteration solver


Initialize flux (and k-eff)
Power iteration/Thermal iteration
Calculate fission source; Power iteration number n=1
Backup flux into POWER_BACKUP module
n = n + 1
Renew up-scattering source with current solution
Group sweep
g = 1
g = g +1
Solve one group normal source problem with the current fission source and up-scattering source
No
g = G?
Yes
Renew fission source (and k-eff)
Compute the flux error between current solution and solution in POWER_BACKUP module
No
Iteration convergence or without fission and up-scattering?
Yes
END

Note: the fluxes used in calculating the fission source and scattering source are stored separately

Fig. IV-2. Multigroup iteration solver with one thermal iteration


Calculate fission source with the flux stored into *BACKUP* module
Accumulate with extraneous source
Loop through group: *g'*=1
If *g'*==*g*?
g = *g* + 1
Accumulate scattering source with the flux stored into current mesh
g<=*G*?
END

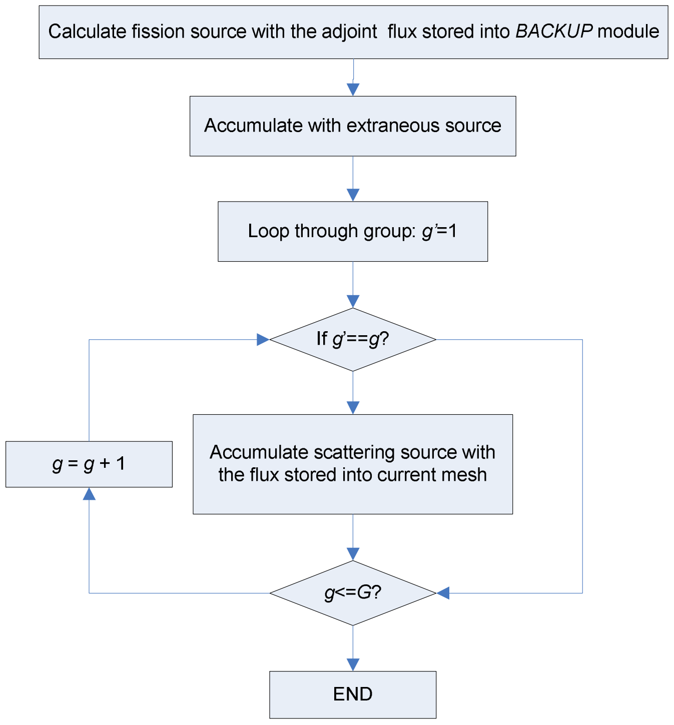

Calculate fission source with the adjoint flux stored into *BACKUP* module
Accumulate with extraneous source
Loop through group: *g'*=1
If *g'*==*g*?
g = *g* + 1
Accumulate scattering source with the flux stored into current mesh
g<=*G*?
END

Fig. IV-3. Diagram of direct and adjoint source update

4.2 Multigroup energy-driven *hp*-adaptation

We present here the concepts of energy-driven mesh adaptation in the context of multigroup equations.

4.2.1 Basic considerations

We start our work on *hp*-adaptation related to multigroup equations by making the following two statements:

1) We wish to keep the thermal and power iterations. This is motivated by the fact that we wish to conserve the sequence of one-group problems, for which the classical *hp*-strategy is readily applicable. By solving a series of one-group source problem, we keep the SPD nature of the matrices to be inverted. Although we do not obtain a solution in one step, the thermal iteration loop usually converges rapidly and some classical techniques are available to accelerate both the thermal iteration and power iteration schemes (e.g., thermal rebalancing and Chebyshev acceleration).

2) Multigroup fluxes should be solved on different meshes. This idea is quite natural since the smoothness of a solution for a given energy group is strongly dependent on the group data. We will therefore have to store one energy mesh per energy group and will need to investigate how to calculate fission and scattering source contribution due to fluxes computed on different meshes. However, such tactics will lead to optimal meshes in all groups, and the price paid for this solely consists in computing integrals of polynomial functions of various orders defined on different meshes.

Essentially, *hp*-adaptation introduces a new iteration (*mesh iteration*), during which meshes are refined non-uniformly in an optimal way (in the sense that the number of unknowns is minimized for a given prescribed tolerance). We propose two implementations of the mesh iteration loop within the context of an existing multigroup eigensolver: in the first implementation, the mesh iterations are wrapped immediately around the one-group solver (the inner iteration solver). In the second implementation, the mesh iterations are wrapped around the most outer loop, i.e., around the power iteration loops. We present these two implementations now.

4.2.2 Mesh iteration implemented around the one-group solver

This implementation is illustrated in Fig. IV-4.

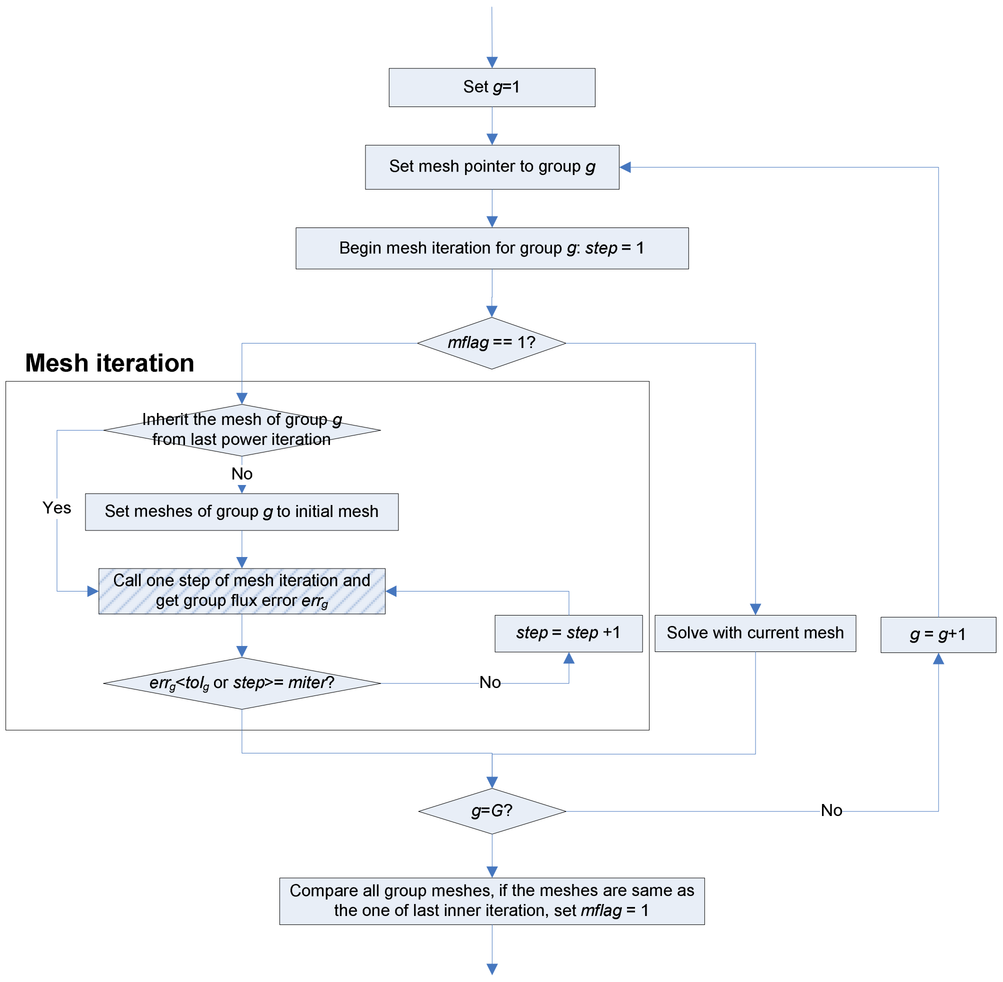

Set g=1
Set mesh pointer to group g
Begin mesh iteration for group g: step = 1
mflag == 1?
Mesh iteration
Inherit the mesh of group g from last power iteration
No
Yes
Set meshes of group g to initial mesh
Call one step of mesh iteration and get group flux error err_g
step = step +1
Solve with current mesh
g = g+1
err_g<tol_g or step>= miter?
No
g=G?
No
Compare all group meshes, if the meshes are same as the one of last inner iteration, set mflag = 1

Fig. IV-4. Group sweep with mesh iteration wrapped around the one-group solver

In this implementation, within one mesh iteration, we are solving a one-group problem (regardless of the fact that this one-group problem is subsequently coupled to other one-group problems via the thermal and power iterations). We can easily perform *hp*-refinement for all groups separately with any user specified tolerances tol_g (note that the tolerances can be group-dependent). We terminate the mesh iteration loop when the following conditions are fulfilled for all groups:

$$E_g = \frac{\left\| E_{g,hp} \right\|_1}{\left\| \phi_{g,hp} \right\|_1} = \frac{\left[\sum_{K_g} \left\| \phi_{g,hp} - \Pi_{g,h^{coarse} p^{coarse}} \phi_{g,hp} \right\|_{1,k}^2 \right]^{1/2}}{\left\| \phi_{g,hp} \right\|_1} = \frac{\left[\sum_{K_g} \mu_{g,k} \right]^{1/2}}{\left\| \phi_{g,hp} \right\|_1} < tol_g \tag{4.19}$$

Obviously, the multigroup fluxes will all have different meshes using this implementation.

An issue to be dealt with is related to the fact that at any given power or thermal iteration, the source terms appearing in the one-group problem are not converged (they may be modified at the next thermal or power iteration). Hence, the optimal *hp*-mesh coming from this implementation may change after each power/thermal iteration, which results into different optimal meshes for all energy groups at each power/thermal iteration. It is possible that, if little care is taken, the overall resulting multigroup meshes are not the optimal meshes because we do not have implemented an un-refinement scheme. If we want to obtain the optimal meshes, we may need to *reinitialize the mesh* at each mesh iteration and perform the mesh adaptation from scratch each time. This process may not be optimal in term of CPU cost. Alternatively, we can simply choose to *inherit the mesh* from last power/thermal iteration; however, our final mesh may not be optimal. A simpler way to reach the optimal mesh is to iteratively adjust (tighten) the desired accuracy tol_g according to the convergence of power/thermal iterations. For example, at the beginning of the power iteration process, we may choose a relaxed tolerance for the mesh adaptation.

Aside from the possibility of inheriting the *hp*-mesh from the previous power/thermal iteration, another very effective scheme stems from the fact that the mesh iterations converge rapidly with respect to the power/thermal iterations. So we can compare the optimal mesh obtained from current power iteration with the one from previous power iteration. If these two meshes are identical, we accept the current mesh as the resulting optimal mesh and switch off the mesh adaptation procedure to solve solely a one-group source problem with the known mesh directly. This *two-step iteration scheme* has been proved very effective in numerical experiments.

We can also control the mesh iteration accuracy tol_g for each group separately. But, because of the existence of group coupling due to scattering and fission, we need to evaluate *the influence of the accuracy from one group to another group*. It is meaningless to set, for instance say, tol_2=0.001% while tol_1=10% in a two-group problem where the source of second group (thermal group) is mainly due to slowing-down from the first group (fast group). Meanwhile, we may want to know how large the fast group tolerance could be for a given thermal tolerance to be fulfilled.

4.2.3 Mesh iteration wrapped outside the power iteration loop

Our second implementation puts the mesh adaptation process outside of the power iteration loop. Hence, all the multigroup eigen-equations are fully solved before the next mesh iteration, i.e., the error is only estimated after the power iterations have converged.

When goal-oriented mesh adaptation will be considered, this mesh adaptation implementation will be the only viable option, because calculating quantities of interest will require knowledge of solutions for all groups in order to refine meshes; (for goal-oriented calculations, placing the mesh iteration right outside the one-group solver will not make sense).

Because of the fission and scattering coupling terms, the bilinear form in this implementation is no longer symmetric, but by assuming continuity and coercivity, as has already been proved in Chapter III, we still have:

$$\|e_H\| \le \frac{\alpha}{\beta}(\|e_h\| + \|E_{h,p}\|) \approx \frac{\alpha}{\beta}\|E_{h,p}\| \tag{4.20}$$

This suggests that the difference between current solution and reference solution will still provide a good estimation of the error and can, therefore, drive the mesh iteration in an effective manner. It is possible that the error effectivity index be different than 1.0 but the asymptotic performance should not be degraded. Numerical experiments show that the ratio α/β is nearly equal to 1.0 for multigroup equations.

We can calculate global relative error E and group relative error E_g after obtaining solutions of all groups on current meshes and corresponding reference solutions using the following equations:

$$E = \frac{\left\| E_{hp} \right\|_1}{\left\| \phi_{hp} \right\|_1} = \frac{\left[\sum_{g=1}^{G} \sum_{K_g} \left\| \phi_{g,hp} - \Pi_{g,h^{coarse}p^{coarse}} \phi_{g,hp} \right\|_{1,K_g}^2 \right]^{1/2}}{\sqrt{\sum_{g=1}^{G} \left\| \phi_{g,hp} \right\|_1^2}} = \frac{\left[\sum_{g=1}^{G} \sum_{K_g} \mu_{g,k} \right]^{1/2}}{\sqrt{\sum_{g=1}^{G} \left\| \phi_{g,hp} \right\|_1^2}} \qquad (4.21.a)$$

$$E_g = \frac{\left\| E_{g,hp} \right\|_1}{\left\| \phi_{g,ref} \right\|_1} = \frac{\left[\sum_{K_g} \left\| \phi_{g,ref} - \phi_{g,hp} \right\|_{1,k}^2 \right]^{1/2}}{\left\| \phi_{g,ref} \right\|_1} = \frac{\left[\sum_{K_g} \eta_{g,k} \right]^{1/2}}{\left\| \phi_{g,ref} \right\|_1} \qquad (4.21.b)$$

, where the local error distribution,

$$\eta_{g,k} = \left\| \phi_{g,ref} - \phi_{g,hp} \right\|_{1,k}^2 \qquad (4.21.c)$$

Note the reference solution in Eq. (4.21) is obviously different from the reference solution in Eq. (4.19).

Using instead the projection-based error distribution, we have:

$$\mu_{g,k} = w_g \left\| \phi_{g,ref,k} - \Pi_{h,p_{g,K}} \phi_{g,ref,k} \right\|^2 \qquad (4.22)$$

w_g in Eq. (4.22) is a group weighting factors. Π is the semi-H^1 projection operator. The larger the weighting factor, the greater chance that the group be refined.

Like in the one group case, we only refine the mesh where error satisfies:

$$\mu_{g,k} > \alpha_1 \max_{(K_1,K_2,\cdots,K_G)} \mu_{g,k} \qquad (4.23)$$

Or we can perform sorting-based or cumulative error-based refinement with parameters α_3 and α_4.

(Refer to Chapter III for the definition of error-based, sorting-based, and cumulative error-based refinements.) Regarding the refinement option (*h* or *p*), the maximum polynomial order constraint and the *h*-refinement constraint in any given element are identical to the one-group mesh refinement case. We will not discuss them in detail here.

This multigroup refinement strategy will also deliver different meshes for different energy group. In this scheme, we control the convergence using

$$E < tol \tag{4.24}$$

The diagram of mesh iteration loop wrapped around the power iteration is given in Fig. IV-5. When mesh iteration is outside of multigroup iteration solver, there is no assembly of global stiffness matrix needed for each power or flux iteration, which constitutes a significant saving. The convergence of power iteration can and should vary with the parameters E and E_g.

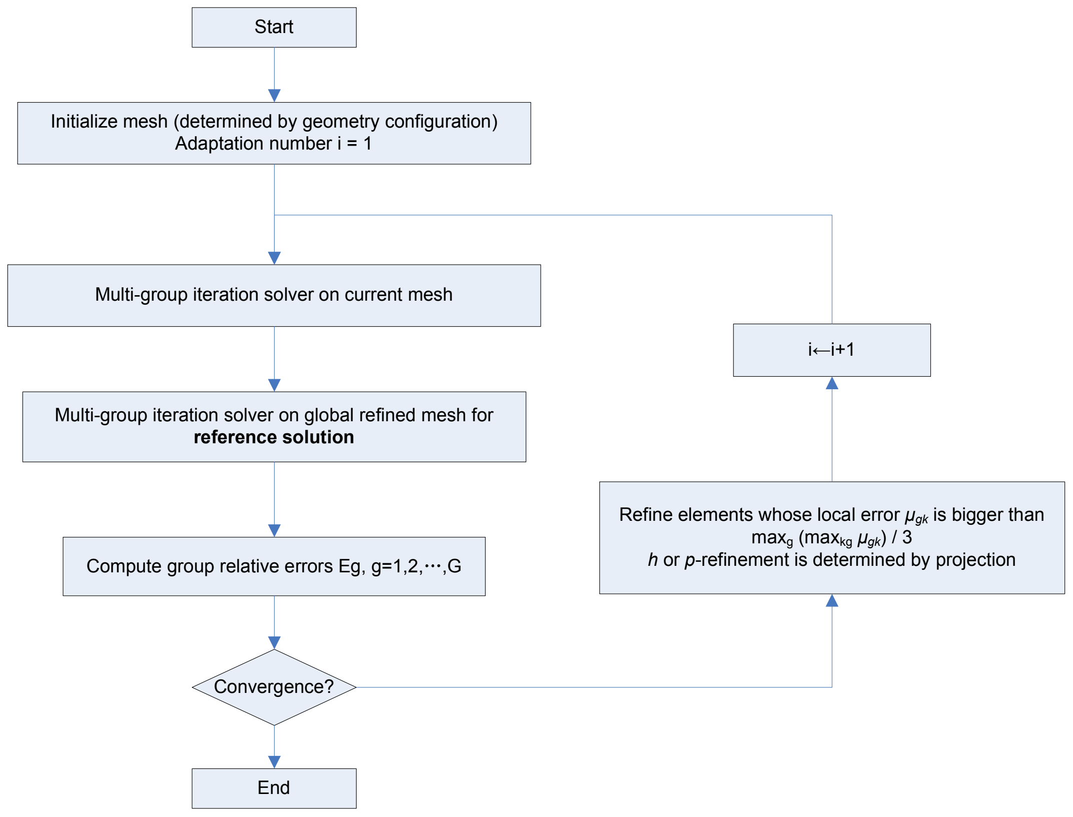

Start
Initialize mesh (determined by geometry configuration) Adaptation number i = 1
Multi-group iteration solver on current mesh
Multi-group iteration solver on global refined mesh for **reference solution**
Compute group relative errors Eg, g=1,2,⋯,G
Convergence?
End
Refine elements whose local error μ_{gk} is bigger than $\max_g(\max_{kg}\mu_{gk})/3$ *h* or *p*-refinement is determined by projection
i←i+1

Fig. IV-5. Schematics of mesh iteration wrapped outside power iteration

One question arises: how do we choose the group weighting factors w_g to synchronize the convergence rate of different groups? Simply put, we choose w_g in order to obtain our calculation objective which is let all E_g, g=1,2,…, G be smaller than their prescribed tolerances respectively with minimum total number of unknowns. If there are no group couplings, different choices of w_g will just yield different E_g. The meshes would still be optimal for their resulting E_g though E_g for some groups may be much smaller than *tol*. We should be aware that there are possibilities of inappropriate choices of w_g which make a mesh for a given group be over-refined because all energy groups are coupled together through fission and scattering. Let us define the convergence path f_g for any energy group:

$$E_g = f_g(N_g), \qquad g = 1,2,\cdots,G$$

N_g is the total number of unknowns of energy group g. And we can also define a "reference" convergence path $f_{g,reference}$ which is the convergence path with all other groups having over-refined meshes all the time of calculation. Because this optimal convergence path always expects the smallest N_g with same accuracy E_g, we want to change our objective to achieve a convergence path which approaches the optimal path as closely as possible by setting w_g for all groups.

Numerical experiments show that this objective is achievable in multigroup neutron diffusion problem by setting

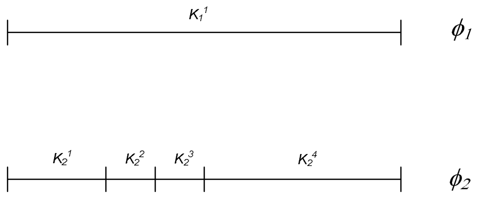
$$w_g = \frac{1}{\left\|\phi_{g,ref}\right\|_1^2} \tag{4.25}$$

This weighting does out damage the optimal convergence path and tends to result in equivalent group relative error E_g (equal distribution between groups). Note that because the mesh iterations are performed outside the power/thermal iterations, acceleration techniques are easily incorporated.

The numerical comparisons between these two implementations, where the mesh iteration loop is located at different locations with respect to the multigroup solver, will also give us insights regarding the behavior of *hp*-adaptation in multigroup diffusion problems.

4.2.4 Construction of group source terms when different meshes are used

In this work, we employ an adaptive integration method to compute the source terms (mass matrix elements) resulting from the product of functions of various polynomial orders defined on various meshes. The adaptive integration process was explained previously in Chapter III. It may not be the optimal method, but it works. For example, consider two meshes for a two group problem illustrated in following graph.



When evaluating source vector of the first group during assembly, we need calculate

$$\int_{K_1^1} \varphi_1 \phi_2 dx \quad ,$$

where φ_1 is shape functions defined on K_1^1. With adaptive integration algorithm, this integral can be automatically transformed into:

$$\int_{K_2^1+K_2^2+K_2^3} \varphi_1\phi_2 dx + \int_{K_2^4} \varphi_1\phi_2 dx = \int_{K_2^1} \varphi_1\phi_2 dx + \int_{K_2^2+K_2^3} \varphi_1\phi_2 dx + \int_{K_2^4} \varphi_1\phi_2 dx$$
$$= \int_{K_2^1} \varphi_1\phi_2 dx + \int_{K_2^2} \varphi_1\phi_2 dx + \int_{K_2^3} \varphi_1\phi_2 dx + \int_{K_2^4} \varphi_1\phi_2 dx$$

On each son elements, the integral can be evaluated exactly with Gaussian quadrature.

A similar situation arises when comparing fluxes computed on different meshes:

$$\int_{K_1^1} (\phi_1 - \phi_2)^2 dx$$

4.3 Multigroup goal-oriented *hp*-adaptation

Even though energy-driven mesh refinement techniques makes a sensible usage of resources (CPU and memory), obtaining a highly accurate solution in every single mesh cell of the computational domain may not be needed from a practical engineering point of view. Moreover, the ultimate answer sought may not be the solution itself but rather a functional of the solution such as a reaction rate integrated over a given volume, or a particle current at a given point. This motivates improvements to the previous algorithms in order to bring together (1) the need for accurate solution in quantities of interest and (2) *hp*-adaptivity, hereby proposing goal-oriented *hp*-adaptivity. Goal-oriented adaptivity should only refine the mesh such that specified quantities of interest are computed within the user prescribed tolerance. One can expect that goal-oriented adaptive techniques will calculate the quantities of interest with the same precision as the energy-driven adaptive scheme but with fewer degrees of freedom.[27] [28] We will develop this goal-oriented approach in the context of multigroup diffusion problem. As we will notice, the goal-oriented approach requires the solution of a dual or adjoint problem.

4.3.1 Detector response as a quantity of interest

The quantities of interest can be represented in the following form (a functional of the solution itself),

$$I(\phi) = \sum_{g=1}^{G} \int_{\Omega} \Sigma_{g,d}(x)\phi_g(x)dx \tag{4.28}$$

Here, subscript *d* means detector. We have assumed here that the quantity of interest is a type of response of a detector. We will later provide some additional types of quantities of interest.

If we let the adjoint source

$$s^*_{ext,g}(x) = \Sigma_{g,d}(x), \qquad g = 1,2,\cdots,G \tag{4.29}$$

we can then seek an adjoint solution such that, for any flux distribution ϕ, the adjoint flux verifies: $b(\phi,\phi^*) = l^*(\phi) = I(\phi)$.

We define the error as

$$\begin{aligned} e_{hp} = \left| I(\phi) - I(\phi_{hp}) \right| &= \sum_{g=1}^{G} \int_{\Omega} \Sigma_{g,d}(x) \left| \phi_g(x) - \phi_{g,hp}(x) \right| dx \\ &= \sum_{g=1}^{G} \sum_{K_g} \int_{\Omega_K} \Sigma_{g,d}(x) \left| \phi_g(x) - \phi_{g,hp}(x) \right| dx \end{aligned} \tag{4.30}$$

which is equal to,

$$e_{hp} = \left| I(\phi) - I(\phi_{hp}) \right| = \left| b(\phi - \phi_{hp}, \phi^*) \right|$$

Because ϕ is the exact solution and ϕ_{hp} is the solution on the current mesh, we have orthogonality, and

$$b(\phi - \phi_{hp}, \phi_{hp}^*) = 0,$$

where ϕ_{hp}^* is the adjoint approximation defined on the same mesh as ϕ_{hp}.

Therefore, we have,

$$e_{hp} = \left| b(\phi - \phi_{hp}, \phi^* - \phi_{hp}^*) \right| \tag{4.31}$$

Similarly to the energy-driven equation, the goal-oriented equation is given by:

$$e_{hp} = \left| b(\phi - \phi_{hp}, \phi^* - \phi_{hp}^*) \right| \leq C \sum_{g=1}^{G} \sum_{K_g} \left\| \phi_{g,k} - \phi_{g,k,hp} \right\|_1 \left\| \phi_{g,k}^* - \phi_{g,k,hp}^* \right\|_1 \tag{4.32}$$

The subscript *k, g* represent the solution on the element number *k* and group number *g*. Continuing along the lines of energy-driven strategies, we define,

$$\eta_{g,k} = \left\| \phi_{g,ref,k} - \phi_{g,k,hp} \right\|_1 \left\| \phi_{g,ref,k}^* - \phi_{g,k,hp}^* \right\|_1 \tag{4.33}$$

Thus, the error estimation si given by:

$$e_{hp} \leq C \sum_{g=1}^{G} \sum_{K_g} \eta_{g,k}$$

Note different groups may have different number of finite elements. Similarly to the energy-driven scheme, we define,

$$\mu_{g,k} = \left\| \phi_{g,ref,k} - \prod_{h,p_{g,K}} \phi_{g,ref,k} \right\|_1 \left\| \phi^*_{g,ref,k} - \prod_{h,p_{g,K}} \phi^*_{g,ref,k} \right\|_1 \tag{4.34}$$

Instead minimizing the error with semi-H^1 norm between reference solution and its project-based interpolation in energy-driven scheme, we minimize the product of the error in the direct solution and the error in the adjoint solution. We choose to refine only the elements satisfying

$$\mu_{g,k} = \alpha_1 \max_{\substack{1 \le g \le G \\ k \in K_g}} \mu_{g,k} \tag{4.35}$$

As before, we select the refinement choice which maximizes the decrease of elementary interpolation error. Note that we use the same multigroup meshes for the direct and adjoint solutions. The details regarding the *h*-refinement constraint and p_{max} are similar to those of the energy-driven schemes.

We define following terms of error tolerance,

$$err = \sqrt{\left| \frac{I(\phi) - I(\phi_{hp})}{I(\phi)} \right|} \tag{4.36.a}$$

$$\overline{err} = \sqrt{\left| \frac{b(\phi_{ref}, \phi^*_{ref}) - b(\phi_{hp}, \phi^*_{hp})}{b(\phi_{ref}, \phi^*_{ref})} \right|} \tag{4.36.b}$$

$$\overline{\overline{err}} = \sqrt{\frac{\sum_{g=1}^{G} \sum_{K_g} \left\| \phi - \phi_{hp} \right\|_1 \left\| \phi^* - \phi^*_{hp} \right\|_1}{\sum_{g=1}^{G} \sum_{K_g} \left\| \phi \right\|_1 \left\| \phi^* \right\|_1}} \tag{4.36.c}$$

For eigenvalue problems, the quantity of interest is simply the inverse of *k*-effective.

$$b(\phi, \phi^*) = \frac{(F\phi, \phi^*)}{k_{eff}} = \frac{1}{k_{eff}} \tag{4.37}$$

4.3.2 Point value as quantities of interest

We may also wish to obtain other types of quantities of interest, such as a point wise flux or a point wise current. In these cases, the adjoint source or even the adjoint flux will be discontinuous, and special considerations regarding the assembly procedure are needed.

For a current as quantity of interest, we can set adjoint source as,

$$\begin{aligned} s^*_{ext,g} &= \vec{n}\cdot\vec{\nabla}(D_g\delta(\vec{r}-\vec{r}_0)) \\ s^*_{ext,g'} &= 0 \qquad g'\neq g, 1\le g'\le G \end{aligned} \tag{4.38}$$

G is the total number of energy groups. $\vec{r}_0$ is a point in the solution domain *V*. Boundary conditions of the adjoint problem are homogeneous. Then the quantity of interest is

$$I=\sum_{g'=1}^{G}(\phi_{g'},s^*_{ext,g'})=(\phi_g,s^*_{ext,g})=\int_V d\vec{r}\phi_g\vec{n}\cdot\vec{\nabla}(D_g\delta(\vec{r}-\vec{r}_0)) \tag{4.39}$$

Using integration by parts,

$$\vec{\nabla}\cdot(a\vec{b})=\vec{\nabla}a\cdot\vec{b}+a\vec{\nabla}\cdot b$$

Letting

$$\begin{aligned} \vec{b} &= \phi\vec{n} \\ a &= D\delta(\vec{r}-\vec{r}_0) \end{aligned}$$

We get:

$$\begin{aligned} I &= \int_V d\vec{r}\phi_g\vec{n}\cdot\vec{\nabla}(D_g\delta(\vec{r}-\vec{r}_0))=\int_V d\vec{r}\vec{b}\cdot\vec{\nabla}a=\int_V d\vec{r}(\vec{\nabla}\cdot(a\vec{b})-a\vec{\nabla}\cdot\vec{b})=\oint_{\partial V}d\vec{s}a\vec{b}-\int_V d\vec{r}a\vec{\nabla}\cdot\vec{b} \\ &= 0-\int_V d\vec{r}D_g\delta(\vec{r}-\vec{r}_0)\vec{\nabla}\cdot(\phi_g\vec{n}) \\ &= \int_V d\vec{r}\delta(\vec{r}-\vec{r}_0)[-D_g\vec{n}\cdot\vec{\nabla}\phi_g]=\int_V d\vec{r}\delta(\vec{r}-\vec{r}_0)J_{\vec{n},g}(\vec{r}) \\ &= J_{\vec{n},g}(\vec{r}_0) \end{aligned} \tag{4.40}$$

This means that if we set adjoint source as in Eq. (4.38), the quantity of interest is the directional current at a specific point $\vec{r}_0$ in a specific direction $\vec{n}$ for a specific group *g*.

In 1-D, this yields

$$\begin{aligned} s^*_{ext,g} &= \frac{d}{dx}(D_g(x)\delta(x-x_0)) \\ s^*_{ext,g'} &= 0 \qquad g'\neq g, 1\le g'\le G \end{aligned} \tag{4.41}$$

The quantity of interest is,

$$I=\int_V dx\phi_g(x)\frac{d}{dx}(D_g(x)\delta(x-x_0))=\left[-D_g(x)\frac{d\phi_g(x)}{dx}\right]_{x=x_0}=J_g(x_0) \tag{4.42}$$

We need to point out that the adjoint source is discontinuous and even the adjoint flux is not continuous.

To perform the calculation using this adjoint source Eq. (4.42), we need calculate the local load vector of the element which contains the point x_0. The local load vector of the element e $[x_1, x_2]$ where $x_0<x_2$ and $x_0>x_1$ is given by:

$$\begin{aligned}(L_i(x), s^*_{ext,g})_e &= \left[-D_g(x)\frac{dL_i(x)}{dx}\right]_{x=x_0} = \left[-D_g(x)\frac{dL_i(x(\xi))}{d\xi}\frac{d\xi}{dx}\right]_{x=x_0} \\ &= -\frac{D_g(x_0)l_i'(\xi_0)}{x_2-x_1}, \qquad i=0,1,\dots,p_e\end{aligned} \tag{4.43}$$

p_e is the polynomial order of element e; and

$$\xi = \frac{x-x_1}{x_2-x_1}; \qquad \xi_0 = \frac{x_0-x_1}{x_2-x_1}; \qquad L\cdot x(\xi) = l(\xi)$$

l is the shape function on the master element [0,1].

If $x_0=x_1$, then

$$(L_i(x), s^*_{ext,g})_e = -D_g(x_0)\delta(0)l_i(\xi=0) + \left[-\frac{1}{2}D_g(x)\frac{dL_i(x)}{dx}\right]_{x=x_0} = -\frac{D_g(x_0)l_i'(0)}{2(x_2-x_1)}$$
$$i=1,2,\dots,p_e$$
$$(L_0(x), s^*_{ext,g})_e = -D_g(x_0)\delta(0) + \frac{D_g(x_0)}{2(x_2-x_1)}$$

There must be another element e' for which $x_2=x_0$, then

$$(L_i(x), s^*_{ext,g})_{e'} = D_g(x_0)\delta(0)l_i(\xi=1) + \left[-\frac{1}{2}D_g(x)\frac{dL_i(x)}{dx}\right]_{x=x_0} = -\frac{D_g(x_0)l_i'(1)}{2(x_2-x_1)}$$
$$i=0,2,3,\dots,p_e$$
$$(L_1(x), s^*_{ext,g})_{e'} = D_g(x_0)\delta(0) - \frac{D_g(x_0)}{2(x_2-x_1)}$$

When assembling these two local vectors, $D_g(x_0)\delta(0)$ cancel. So when x_0 is on the interface of two element, local load vector can written into,

$$(L_i(x), s^*_{ext})_e = -\frac{1}{2}\frac{D(x_0)l_i'(\xi_0)}{(x_2-x_1)}, \qquad i=0,1,\dots,p_e \tag{4.44}$$

With the capability of solving the adjoint problem with a given mesh, ideas of goal-oriented *hp*-refinement in previous section can then be applied.

In order to obtain a point wise flux, we may also set adjoint source as

$$\begin{aligned} s^{*}_{ext,g} &= \delta(\vec{r}-\vec{r}_0) \\ s^{*}_{ext,g'} &= 0 \qquad g' \neq g, 1 \leq g' \leq G \end{aligned} \tag{4.45}$$

to make the flux at $\vec{r}_0$ of group g as the quantity of interest. It is

$$\begin{aligned} s^{*}_{ext,g} &= \delta(x-x_0) \\ s^{*}_{ext,g'} &= 0 \qquad g' \neq g, 1 \leq g' \leq G \end{aligned} \tag{4.46}$$

in 1D. The adjoint source is discontinuous. However, adjoint flux is continuous.

Local load vector is

$$l_i(\xi_0), \qquad i = 0,1,\ldots, p_e \quad \text{or} \tag{4.47.a}$$

$$\frac{l_i(\xi_0)}{2}, \qquad i = 0,1,\ldots, p_e \tag{4.47.b}$$

for the element which contains x_0 or elements whose one of the vertices is x_0.

4.4 Data structure and algorithms

4.4.1 Modification of main data structure

Except initial element and some numerical parameters for example, maximum number of vertex/middle nodes, in module DATA_STRUCTURE1D, others are changed into pointers. Two new modules GROUPMESH and AGROUPMESH are added to store the mesh structure, refinement history and solutions. GROUPMESH stores the mesh structure and solutions of direct problem for all groups. In this module, methods GMESHGEN and GMESHINIT(IG) initialize all group meshes with initial mesh and zero flux or for a specific group only. Another subroutine in this module is SHIFT_MAIN_DATA(IG), which makes the pointers in DATA_STRUCTURE1D point to the specific group in GROUPMESH. Once the real arrays and pointers are associated, all existing algorithms operating on DATA_STRUCTRUE1D are operating on the pointed mesh. This mechanism gives a minimum code modification. Same module and methods exist for adjoint problem, which are AGROUPMESH, AGMESHGEN, AGMESHINIT(IG) and ASHIFT_MAIN_DATA(IG).

4.4.2 BACKUP module and REFERENCE module

BACKUP module provides a place to store the current solutions including mesh structure of active elements and associated DOFs. Operations in this module include storing solutions in the main data structure into the module, comparing current solutions with the stored results, backup module initialization, and etc. Solutions in BACKUP module are used to construct the multi-group source terms and to calculate the error between two successive power or flux iterations. REFERENCE module is another place to store the solutions. It contains same data and methods of BACKUP. The REFERENCE module is use to provide an “exact” solution.

4.4.3 Goal-oriented calculation

To support the implementation of the mesh adaptation wrapped outside the power iteration and the implementation of goal-oriented adaptation, nearly everything in HP_STRATEGY from the original code was rearranged and enhanced.

The *hp*-adaptive procedure for mesh iteration outside or goal-oriented scheme is:

1) Set mesh iteration step=0 and initialize all group meshes with the same initial mesh

2) Store current coarse meshes for all energy groups (middle-node numbers and their orders)

3) Solve multi-group equations on current mesh iteratively (assembly procedure is embedded)

4) (Optional) compare current solution with an “exact” solution stored in the reference module

5) Refine mesh globally and store coarse mesh solution for all groups

6) Solve multi-group equation on finer mesh (assembly procedure is embedded)

7) Store mesh vertex coordinates and all fine DOFs of all groups for projection

8) Compute error either for direct, adjoint flux or of quantity of interest between

reference and current solutions

9) If estimated errors are less than the prescribed tolerance, stop

10) Find the optimal refinement for each element k among p_k+1 competitive choice for all groups while obtaining interpolation error, expected error drop and refinement flags

11) Perform refinement according to the refinement flag

12) Clear temporary memory for next mesh iteration and set step = step+1

13) Go to 2)

Again, in 10) we may have either enforced h-refinement due to the limitation on p_{max} or enforced p-refinement due to relatively small error drop of h-refinement.

There are several new subroutines, for example BILINEAR to calculate $b(\phi_{hp},\phi_{hp}^*)$, TOTALPOWER to calculate $\int_\Omega \left[\sum_{g'=1}^{G} v\Sigma_{f,g'}(x)\phi_{g',hp}(x)\right]dx$ and ADJOINTNORMAL for $(F\phi_{hp},\phi_{hp}^*)$. Currently, we did not focus on the efficiency of the code. All iterations or numerical integrals are calculated with adaptive integral method with sufficient accuracy.

4.5 Examples and discussions

4.5.1 Description of figures presented

Several main types of figures are used extensively in presenting the results below. It is convenient to describe the various types of figures beforehand in a generic fashion.

4.5.1.1 Convergence path of mesh iteration

Mesh iteration delivers a series of approximate solutions, $\phi_{hp}^i(x), i=1,2,...,N$. N is the total number of refinement steps. x-coordinate in this figure is the number of degrees of freedom , which is equal to $1+\sum_{K_g} p_{g,k}$, and y-coordinate is the relative error of solution in a specific norm (semi-H^1 or L_2) E_g for a given energy group in energy-driven calculation. In goal-oriented calculation we plot the convergence path with x-coordinate being the total number of degrees of freedom of all energy groups $\sum_{g=1}^{G}(1+\sum_{K_g} p_{g,k})$ and with y-coordinate being the square root of relative error of the quantity of interest. Each point in these curves represents a step of mesh adaptation. One can easily see how many degrees of freedom are needed to reach a certain tolerance with this figure. Moreover this figure can also show the different convergence properties among energy groups clearly. It is also used to show roughly how large the differences are between different *hp*-adaptive options (for example, different coefficients of error-based strategy α_1 or different *h*-constraint α_2, etc…)

4.5.1.2 Flux or adjoint flux distribution

This is the plot of solution in 1D space. x-coordinate is position usually in cm. y-coordinate is flux in 1/cm2/sec or dimensionless adjoint flux. To have sufficient spatial resolution, flux in each element are plotted with 2**p*+1 even-distributed points, where *p* is the polynomial order of the element. As another result we can see the mesh density with the dotted flux curves. Sometimes the solution is too accurate or there are too many points, dots in curves are not attached.

4.5.1.3 Mesh structure

In FEM mesh are represented with coordinates of all vertices of elements and the associated polynomial order. So we can plot elementary polynomial order with element vertices to show the mesh structure. It could be more meaningful to plot density of elementary polynomial order because this density is just the local density of degrees of freedom.

4.5.1.4 Error distribution

Similarly to the flux or adjoint flux distribution plot, the only difference in this plot is that the y-coordinate is the difference between approximation solution and analytical value or “exact” numerical value at the points.

4.5.2 Options of *hp*-adaptation

Even though we have explained *hp*-adaptation thoroughly in Chapter III and in previous sections of this chapter, we feel it is better to summarize all possible options of *hp*-adaptation again. Besides demonstrating that the calculations are successful, these results in the following section provide insight regarding how these options affect the *hp*-adaptation in multigroup diffusion.

1. Energy-driven or goal-oriented calculation
2. Mesh iteration inside or outside power/thermal iteration
3. Group weighting: norm of group flux or 1 for all groups
4. Refinement strategy: error-based, sorting-based or cumulative error-based, which associate a constant parameter α_1, α_3 or α_4
5. *h*-constraint: the higher α_2 is, the more unlikely *h*-refinement will be chosen
6. Initial mesh: including number of elements and their initial polynomial order
7. Maximum elementary polynomial order p_{max}
8. Shape functions: Lobatto or Peano
9. Norm to calculate interpolation error: semi-H^1, L_2 or energy norm
10. Error tolerance tol_g or tol

Much more attention is paid to entries 1 to 5. *hp*-refinement always converges, no matter what initial mesh is chosen. We usually use a initial mesh determined by material compositions. We keep using p_{max}=17 and Lobatto shape functions. We did not investigate entry 9 thoroughly since semi-H^1 norm seems to be working extremely well in multigroup diffusion problems. We choose tolerances small enough to show the property of *hp*-adaptation.

4.5.3 Two-group energy-driven source problem without fission or up-scattering

4.5.3.1 Example 2.A (Uniform source problem with analytical solution)

Description of the example:

It is a 2-group 1-D source problem. A 80-cm thick slab is composed of a single material, whose D_1=1.2cm, D_2=0.4cm, Σ_{r1}=0.03cm^{-1}, Σ_{r2}=0.1cm^{-1} and $\Sigma_{s1\to 2}$=0.02cm^{-1}. There are no fissions and up-scattering i.e. $\nu\Sigma_{f1}= \nu\Sigma_{f2}= \Sigma_{s2\to 1}=0$. The left boundary is homogeneous Neumann (reflection) and the right boundary condition is zero flux. There is an even-distributed volumetric fast extraneous source of intensity 1.5cm^{-3}sec^{-1}. The problem can be described by following equations.

$$-D_1\frac{d^2\phi_1(x)}{dx^2}+\Sigma_{r1}\phi_1(x)=Q$$
$$-D_2\frac{d^2\phi_2(x)}{dx^2}+\Sigma_{a2}\phi_2(x)=\Sigma_{12}\phi_1(x)$$
$$\left.\frac{d\phi_1(x)}{dx}\right|_{x=0}=\left.\frac{d\phi_2(x)}{dx}\right|_{x=0}=0 \qquad \phi_1(a)=\phi_2(a)=0$$

a is the thickness of the slab. This problem has the following analytical solution:

$$\phi_1(x)=\frac{Q}{\Sigma_{r1}}(1-\frac{\cosh(x/L_1)}{\cosh(a/L_1)})$$
$$\phi_2(x)=\frac{Q\Sigma_{12}}{\Sigma_{r1}\Sigma_{a2}}(1-\frac{L_1^2}{L_1^2-L_2^2}\frac{\cosh(x/L_1)}{\cosh(a/L_1)}+\frac{L_2^2}{L_1^2-L_2^2}\frac{\cosh(x/L_2)}{\cosh(a/L_2)})$$
$$L_1=\sqrt{\frac{D_1}{\Sigma_{r1}}} \qquad L_2=\sqrt{\frac{D_2}{\Sigma_{a2}}}$$

$$\frac{d\phi_1(x)}{dx}=-\frac{Q}{\Sigma_{r1}L_1}\frac{\sinh(x/L_1)}{\cosh(a/L_1)})$$
$$\frac{d\phi_2(x)}{dx}=\frac{Q\Sigma_{12}}{\Sigma_{r1}\Sigma_{a2}}(\frac{L_2}{L_1^2-L_2^2}\frac{\sinh(x/L_2)}{\cosh(a/L_2)}-\frac{L_1}{L_1^2-L_2^2}\frac{\sinh(x/L_1)}{\cosh(a/L_1)})$$

Calculation conditions:

Initial meshes of both groups have only one linear element; p_{max}=17; error-based refinement strategy with α_1=1/3 is used; the implementation of mesh adaptation wrapped around the one-group solver is tested, an energy-driven type of calculation is performed.

Results and discussions:

The *convergence path* is shown in Fig. IV-6. Various α_2 were tested.

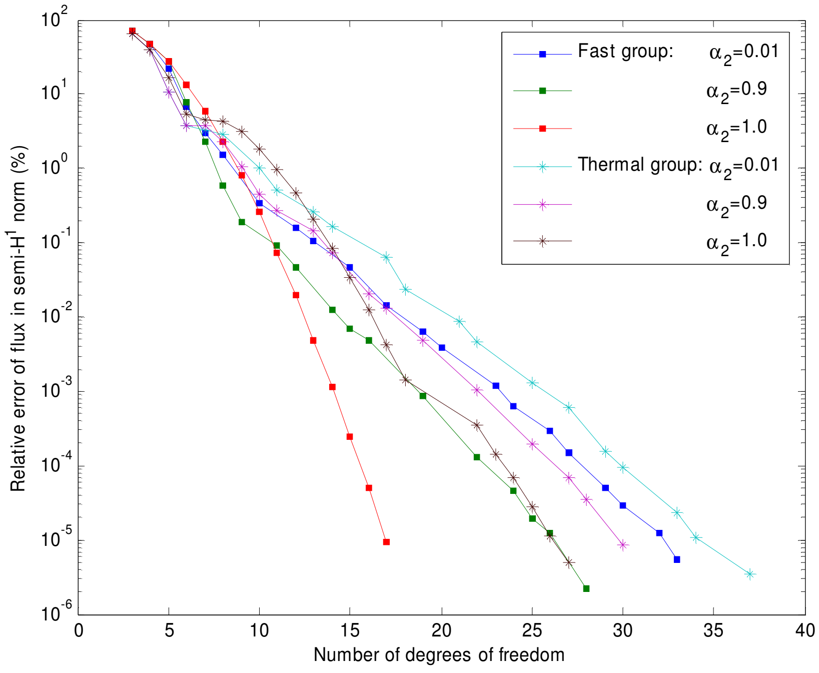


Fig. IV-6. Convergence sequence of example 2.A

We obtain exponential convergence (semi-log plot in Fig. IV-6). The two-group fluxes have different converging rates. The thermal energy group flux (group number 2) requires more DOFs for the same accuracy. We need a higher *h*-constraint to reduce further the number of DOFs for solutions without singularities. If not specifically mentioned, we choose α_2=0.9 from now on.

The influence of the fast flux tolerance on the thermal flux is intricate theoretically but we can observe it numerically.

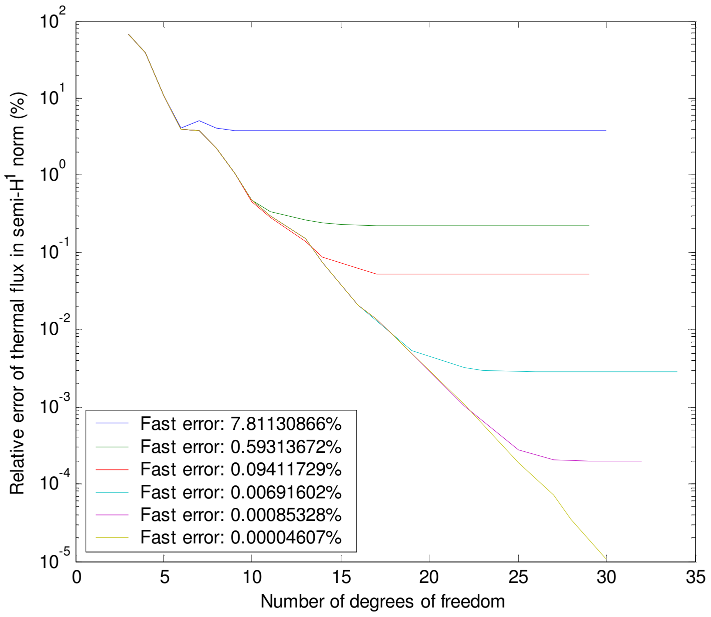


Fig. IV-7. Influence on thermal group of fast convergence

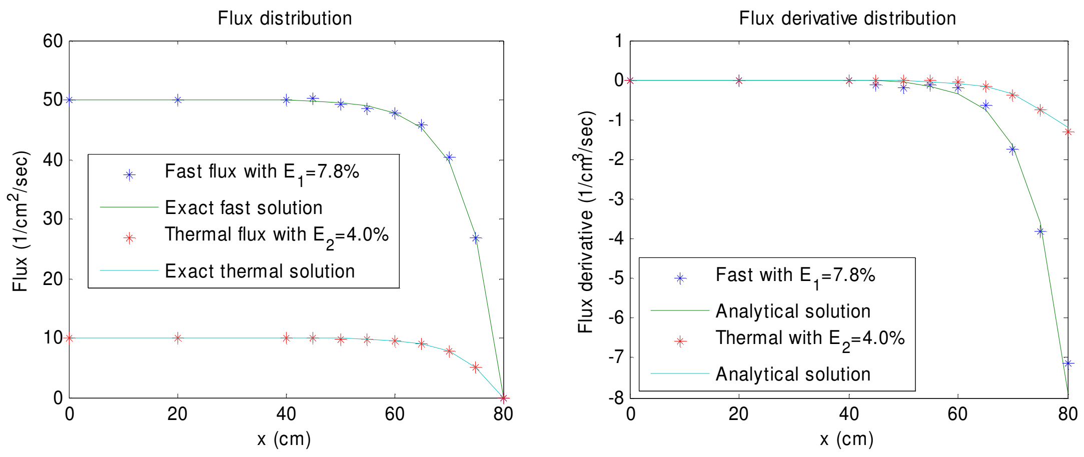
TABLE IV-I

Influence of fast group convergence criterion on the thermal group convergence

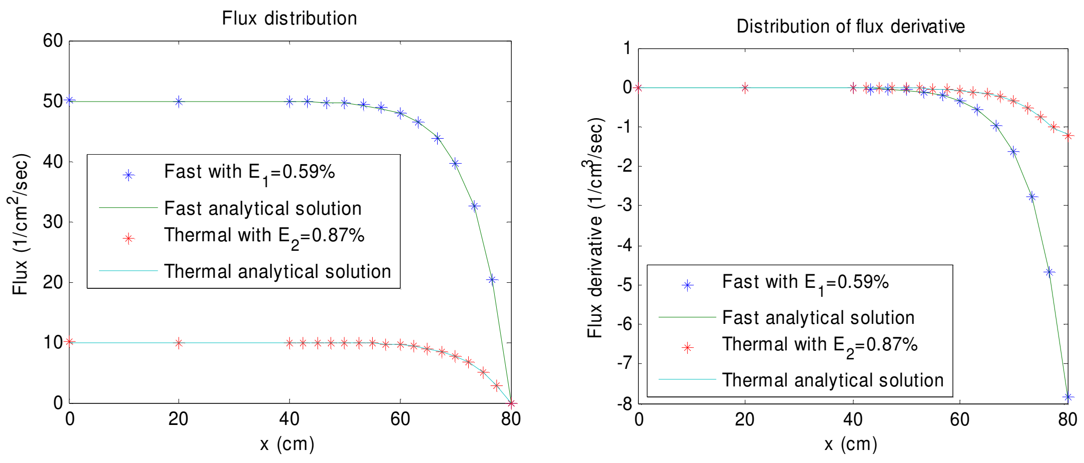
Fast error	Original thermal diffusion length		Half thermal diffusion length		Double diffusion length	
	Thermal error	ratio	Thermal error	ratio	Thermal error	ratio
7.81130866	5.07854942	1.54	6.58622472	1.19	3.00790095	2.60
0.59313672	0.31872578	1.86	0.45299501	1.31	0.17571701	3.38
0.09411729	0.06342188	1.48	0.07678684	1.23	0.04513204	2.09
0.00691602	0.00390303	1.77	0.00523254	1.32	0.00234359	2.95
0.00085328	0.00032143	2.65	0.00051973	1.64	0.00014888	5.73
0.00004607	0.00001215	3.79	0.00002421	1.90	0.00000475	9.70

The above curves of thermal group in Fig. IV-7 are obtained by comparing with an “exact” thermal solution, which was converged with sufficient accuracy in both the fast and in thermal group. We can see clearly that the fast group accuracy does impact the thermal convergence. Fully-converged thermal errors with different fast group errors for different thermal diffusion length are listed in TABLE IV-I. The ratio between fast error and thermal error only depends on diffusion length. It seems like the error of the fast group flux is damped in the thermal flux calculation.

Let us plot the flux distributions with different tolerances and compare them with analytical results in Fig. IV-8.



(a)



(b)

Fig. IV-8. Solutions of example 2.A: (a) flux and flux derivative with *tol*=10%, (b) flux and flux derivative with *tol*=1%

We can see that for tolerances below 1%, the error is unnoticeable to the eye anymore.

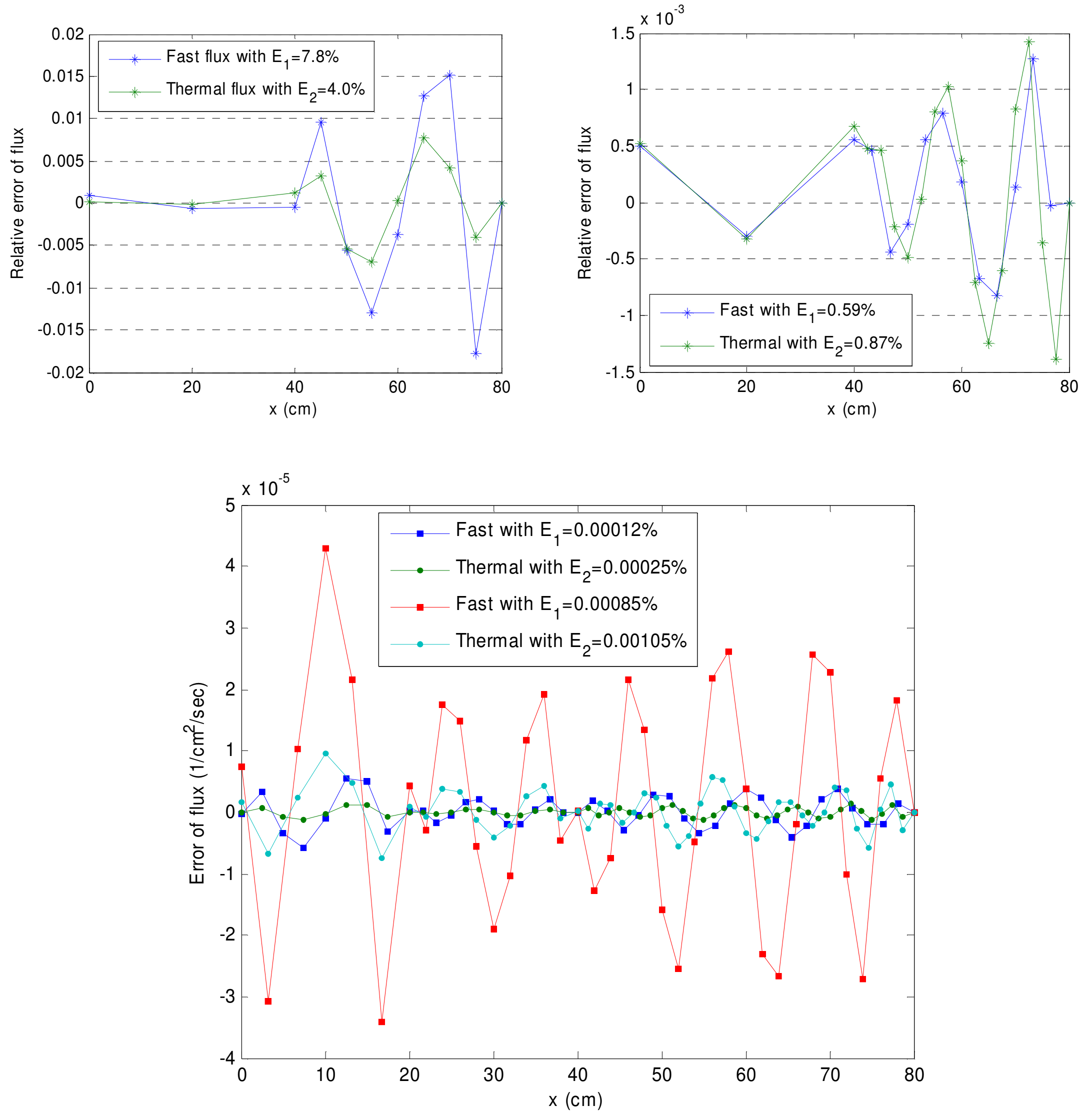


Fig. IV-9. Error distribution of fluxes with different tolerances, example 2.A

Error distributions are also plotted on Fig. IV-9, this confirms that the error decreases proportionally with an increase of the convergence accuracy. Furthermore, the error distribution tends to be even-distributed.

Final mesh structure is given in Fig. IV-10.

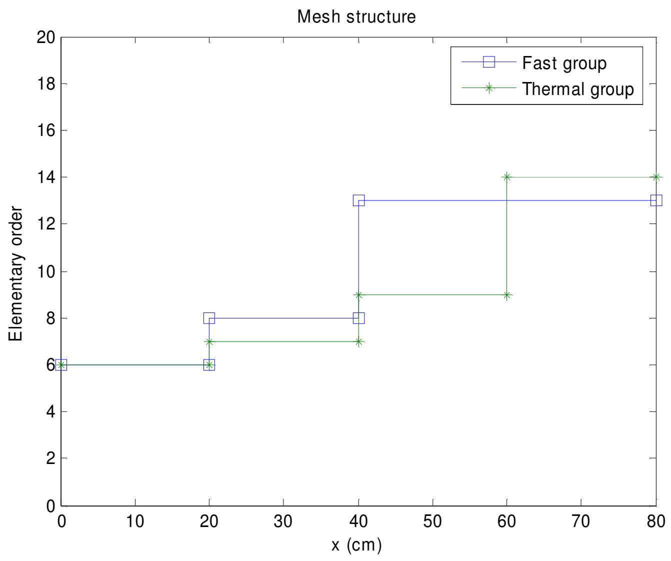


Fig. IV-10. Mesh structure when $E_1 = 2.3\times10^{-6}\%$; $E_2 = 7.7\times10^{-6}\%$ of example 2.A

We can see in Fig. IV-10, 3 *h*-refinements were performed in the thermal group and 2 *h*-refinements in the fast group. More DOFs near zero-flux boundary are needed.

4.5.3.2 Examples 2.B (Surface source problem with analytical solution)

Problem description:

- 2-group 1-D source problem;
- 80-cm thick slab with single material;
- No fissions and up-scattering;
- No volumetric extraneous source;
- Right zero flux of both groups and left net surface fast source 1.5cm^{-2}sec^{-1} and thermal reflectory boundary
- Material properties:

D_1 (cm)	D_2 (cm)	Σ_{r1} (cm^{-1})	Σ_{r2} (cm^{-1})	$\Sigma_{s1\rightarrow2}$ (cm^{-1})
1.2	0.4	0.03	0.1	0.02

Governing equations:

$$-D_1 \frac{d^2\phi_1(x)}{dx^2} + \Sigma_{r1}\phi_1(x) = 0$$
$$-D_2 \frac{d^2\phi_2(x)}{dx^2} + \Sigma_{a2}\phi_2(x) = \Sigma_{12}\phi_1(x)$$
$$D_1 \frac{d\phi_1(x)}{dx}\bigg|_{x=0} = S; D_2 \frac{d\phi_2(x)}{dx}\bigg|_{x=0} = 0 \qquad \phi_1(a) = \phi_2(a) = 0$$

Analytical solution:

$$\phi_1(x) = \frac{SL_1}{D_1}\frac{\sinh(\frac{a-x}{L_1})}{\cosh(\frac{a}{L_1})}; \qquad \phi_2(x) = \frac{S\Sigma_{12}}{D_1\Sigma_2}\frac{L_1^2}{L_1^2 - L_2^2}\left(\frac{L_1\sinh(\frac{a-x}{L_1})}{\cosh(\frac{a}{L_1})} - \frac{L_2\sinh(\frac{a-x}{L_2})}{\cosh(\frac{a}{L_2})}\right)$$
$$L_1 = \sqrt{\frac{D_1}{\Sigma_{r1}}} \qquad L_2 = \sqrt{\frac{D_2}{\Sigma_{a2}}}$$

Calculation conditions: 1 linear initial element for both groups; p_{max}=17; error-based refinement strategy with α_l=1/3; mesh iteration wrapped around he one-group solver; energy-driven case.

Again, we observed exponential convergence in Fig. IV-11. Different groups have different converging rates. The thermal energy group needs more DOFs for the same accuracy. A larger h-constraint coefficient of α_2=0.9 is preferred.

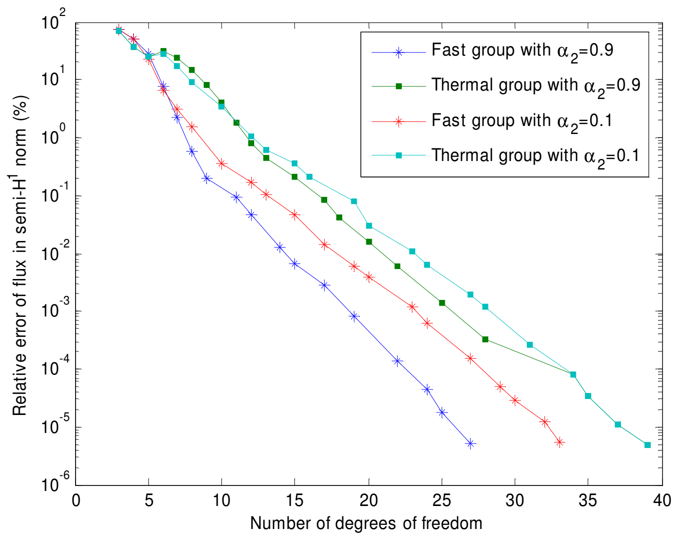
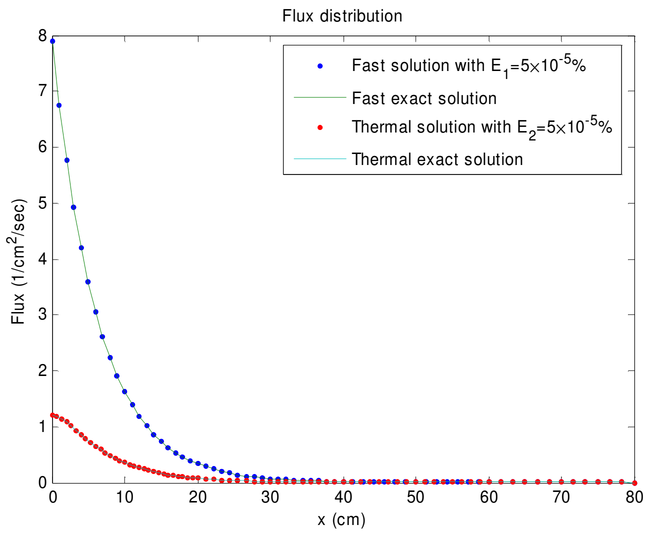
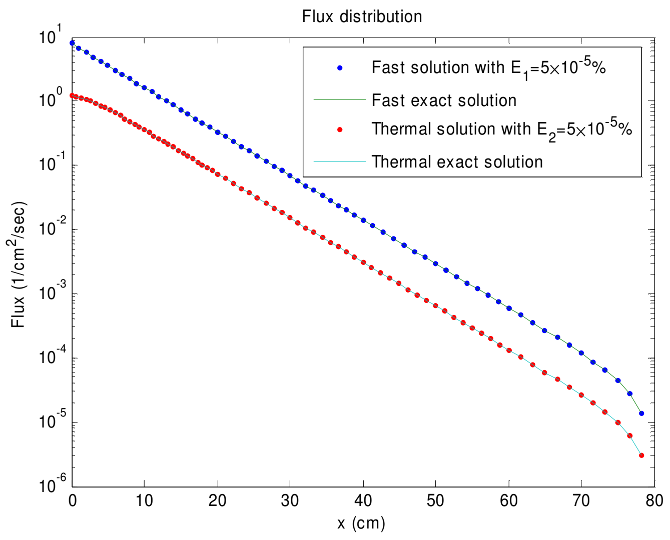


Fig. IV-11. Convergence sequence for example 2.B



(a) linear-linear



(b) log-linear

Fig. IV-12. Flux distributions of example 2.B

Thermal flux is nearly proportional to fast flux after several thermal diffusion lengths as illustrated in Fig. IV-12.

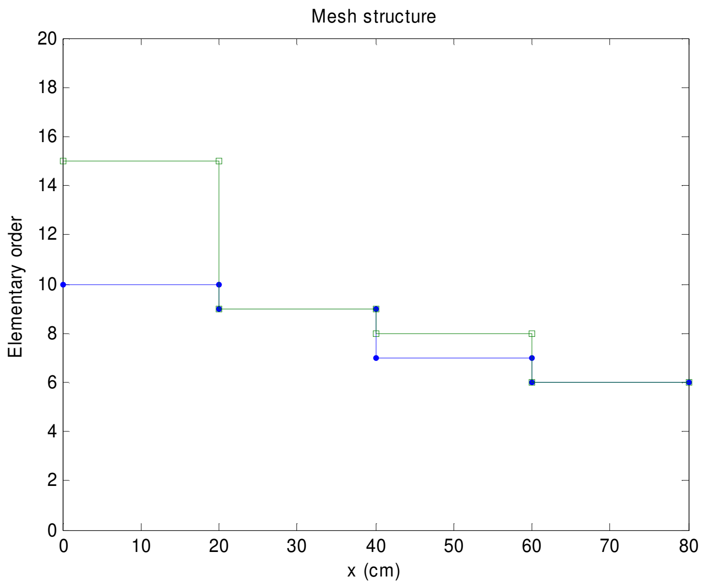


Fig. IV-13. Mesh structure when E_1 = 5.3×10^{-6}%; E_2=4.9×10^{-6}% of example 2.B

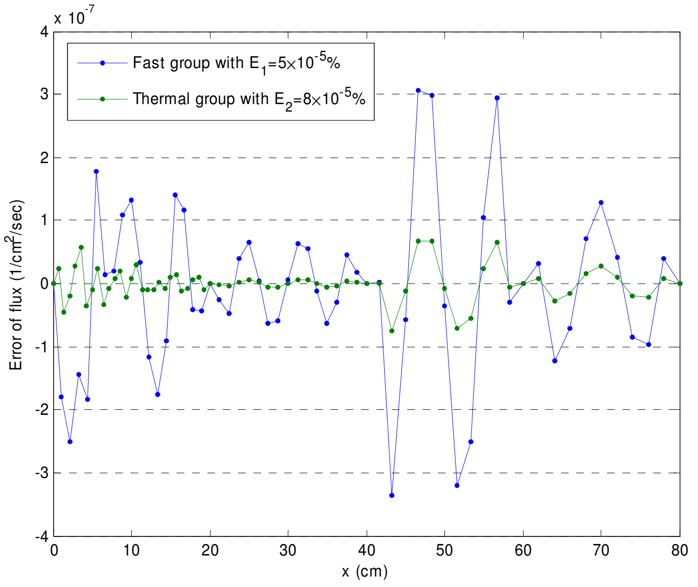


Fig. IV-14. Error distribution for the fluxes of example 2.B

We can obtain really accurate results within few mesh iterations as shown in Fig. IV-13 and Fig. IV-14.

4.5.3.3 Example 2.C (Piece-wise constant material properties and extraneous source)

Problem description:

- 2-group 1-D source problem;
- Seven 100-cm thick assemblies with three types are arranged in 1-2-3-2-3-3-2;
- Without fission and up-scattering;
- Each type of assembly corresponding to a single material numbered from 1 to 3;
- Assembly-wide piecewise constant fast volumetric source 0.0-1.5-1.8-1.5-1.8-1.8-1.5 cm^{-3}sec^{-1};
- Zero flux are at both right and left boundaries;
- Material properties:

Material #	D_1 (cm)	D_2 (cm)	Σ_{r1} (cm^{-1})	Σ_{r2} (cm^{-1})	$\Sigma_{s1\to 2}$ (cm^{-1})
1	1.2	0.4	0.03	0.1	0.02
2	1.2	0.4	0.03	0.2	0.015
3	1.2	0.4	0.03	0.25	0.015

Governing equations:

No theoretical solution is available but several diffusion lengths inside a material we can ignore the leakage terms in the two group equations: (Only a fast source is present in the problem.)

$$\cancel{-D_1\frac{d^2\phi_1(x)}{dx^2}}+\Sigma_{r1}\phi_1(x)=S$$

$$\cancel{-D_2\frac{d^2\phi_2(x)}{dx^2}}+\Sigma_{a2}\phi_2(x)=\Sigma_{12}\phi_1(x)$$

Then we can get a solution in an infinite homogeneous media,

$$\phi_1=\frac{S}{\Sigma_{r1}};\qquad \phi_2=\frac{\Sigma_{12}}{\Sigma_{a2}}\phi_1$$

The transition area at the interface of two different materials can be also analyzed. In general the larger diffusion length, the bigger transit area of the material will have.

Calculation conditions:

1 linear initial element per assembly; p_{max}=17; α_1=1/3; energy-driven case.

Results:

In the mesh-iteration-inside calculation with α_2=0.1, 33 iterations for the fast group and 72 iterations for the thermal group are needed to obtain relative errors less than 10^{-4}%. Among the 72 iterations, only 20 iterations are in the error asymptotic range. The convergence sequence is illustrated in Fig. IV-15. The 'exact' numerical solution is obtained with a tolerance of 10^{-5}%.

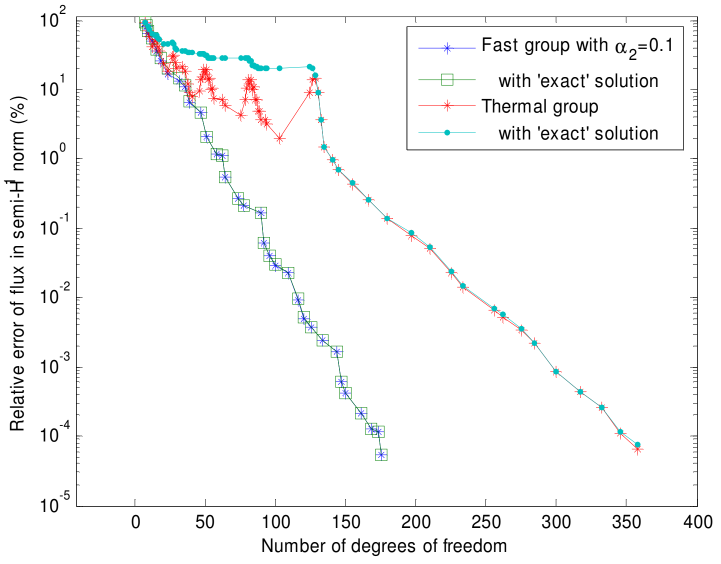


Fig. IV-15. Convergence sequence of example 2.C with α_2=0.1

Again we obtain exponential convergence for both groups. The different groups have different convergence rates. The error calculated with the reference solution is very accurate in the asymptotic range.

Repeating the same calculation with α_2=0.9, we obtain the following results: 31 iterations for the fast group and 76 iterations for the thermal group are needed to achieve a tolerance less than 10^{-4}%.

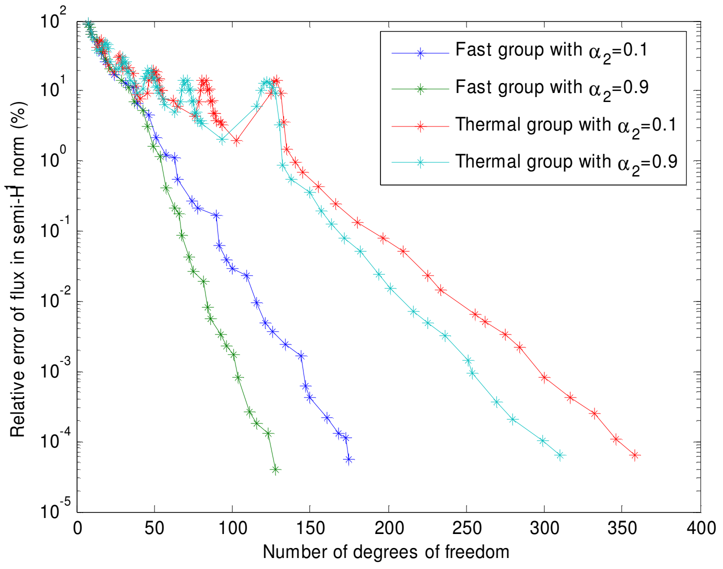


Fig. IV-16. Convergence sequence for example 2.C with α_2=0.9

We obtain a similar exponential convergence but with fewer DOFs as illustrated in Fig. IV-16, which signifies that for this neutron diffusion problem, a higher *h*-constraint is better.

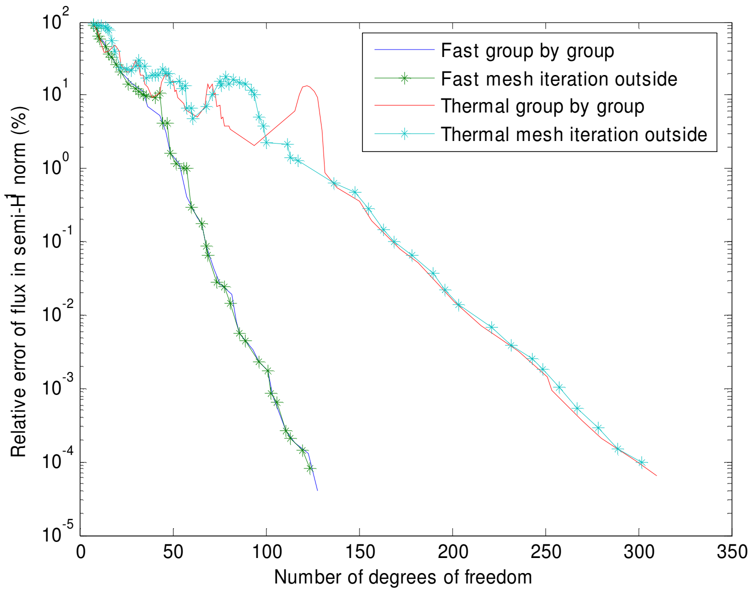


Fig. IV-17. Convergence sequence of example 2.C with mesh iteration outside compared with mesh-iteration-inside scheme

We then calculated the same case, but with the implementation of the mesh adaptation loop wrapped around the power iteration solver. In this calculation, the square of flux semi-H^1 norm was chosen as a group weighting factor. We needed 64 iterations to obtain relative error below $10^{-4}\%$ with α_2=0.9. The convergence sequence comparing the two mesh adaptation implementations is given in Fig. IV-17. Note that the numbers of points in the thermal and fast group curves are the same for mesh-iteration-outside scheme. The two convergence sequences are extremely coherent with each other, especially in the asymptotic range. This means that no matter what schemes we adopt, they will deliver about the same final mesh structures.

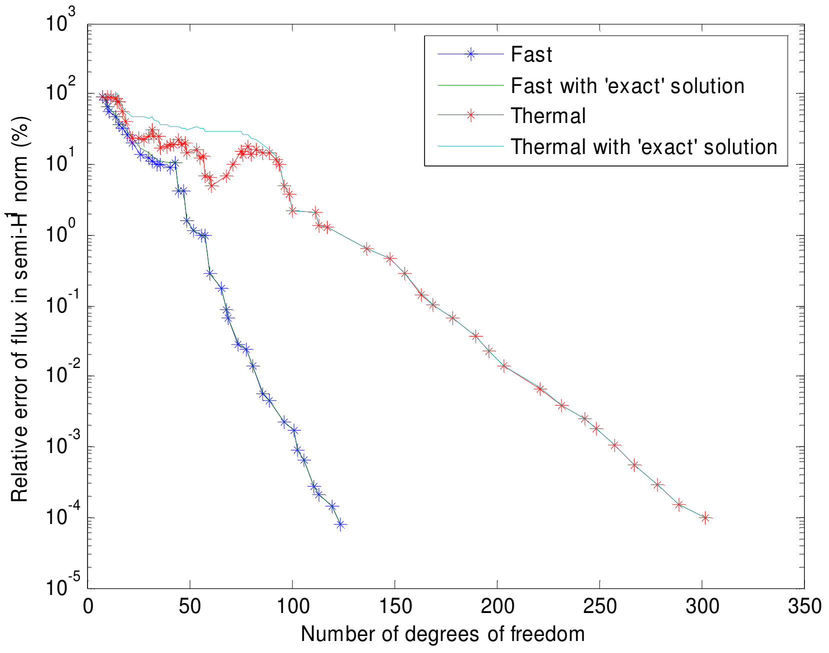


Fig. IV-18. Convergence sequence of example 2.C with mesh iteration outside

Comparing the convergence sequence with the ‘exact’ solution in Fig. IV-18, we can see for these two-group source problems, the error estimator still provides an accurate result even though the bilinear form is no longer symmetric. Different group weighting factors are also tested and the results are presented n Fig. IV-19.

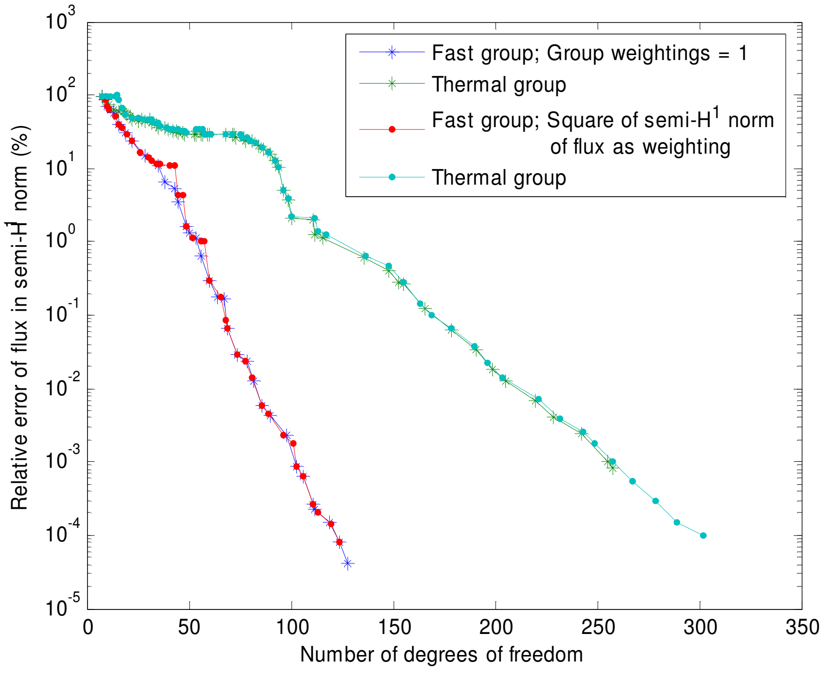


Fig. IV-19. Convergence sequence of example 2.C with different group weighting factors

These two different sets of weighting factors give same mesh convergence sequence. They only have slightly different convergent paths. This also means we do not have to converge fast flux too much to obtain an optimal thermal mesh.

We also verified that using a unique mesh for both groups was not optimal. The blue curve in Fig. IV-20 is the error of $\phi_1 + \phi_2$, the green curve is for ϕ_1 only; the red curve is for ϕ_2. We can see that with same triangulation, the fast group is much more accurate than thermal, mainly because of its bigger magnitude. At the asymptotic range, both errors decrease with same rate. Refining in the thermal group does not automatically produce the best error reduction in the fast group. The mesh based on the sum of the errors in both fluxes is not optimal for neither the thermal nor the fast group.

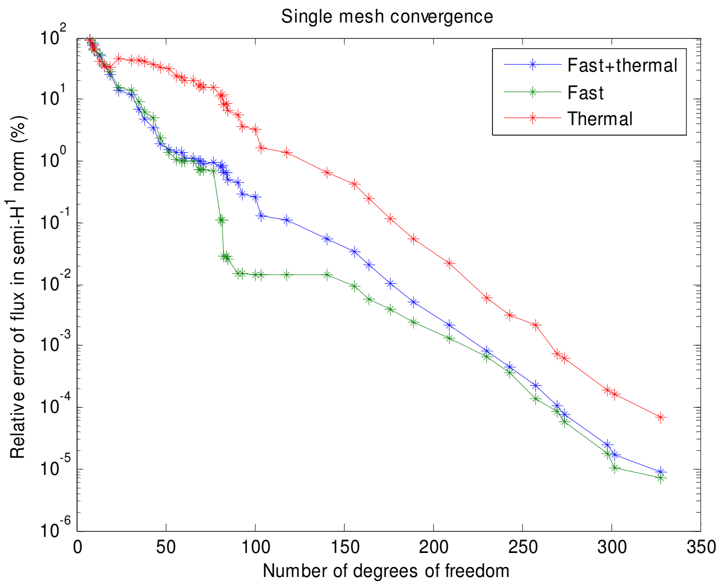


Fig. IV-20. Convergence sequence of example 2.C with single mesh

The flux distributions are given in Fig. IV-21.

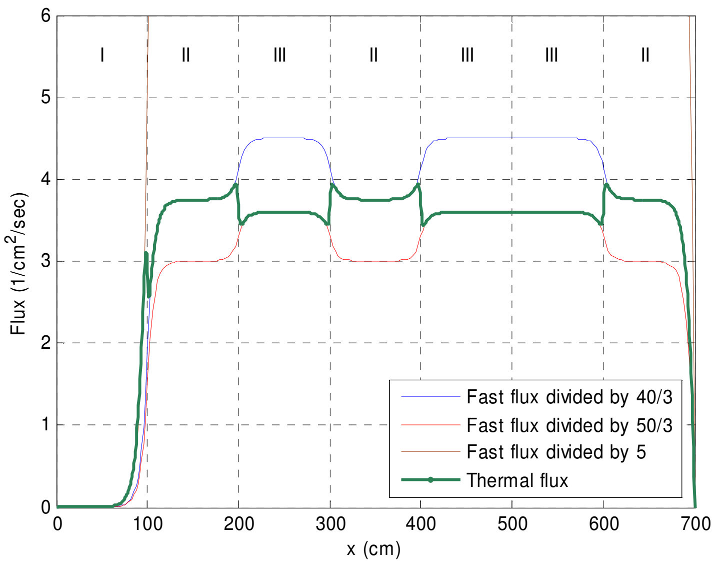
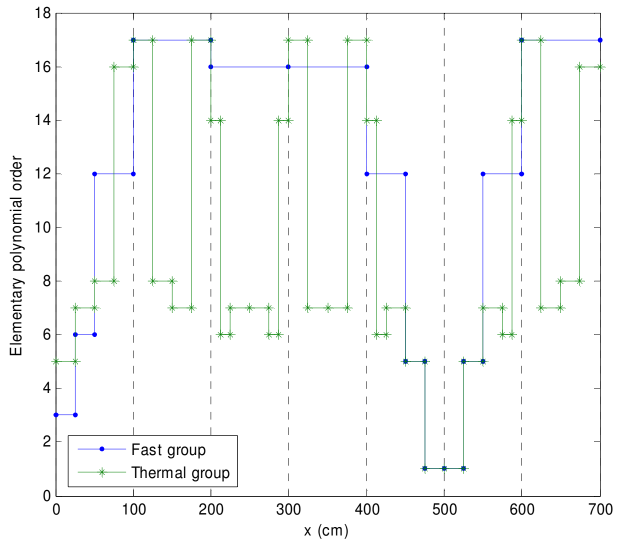
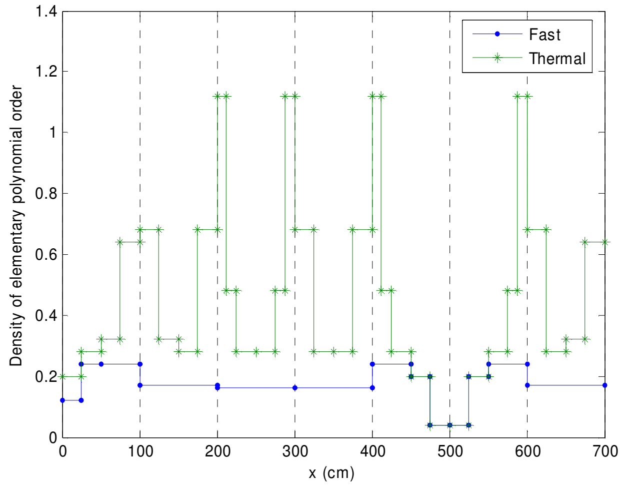


Fig. IV-21. Flux distributions of example 2.C

Results showed that as long as the calculation ended in the asymptotic range, it captures the thermal flux spatial transition extremely accurately. Mesh structures are provided in Fig. IV-22. with E_1=8.1×10^{-5}% and E_2=1.0×10^{-4}%. It is more useful to plot the density of element polynomial order. Generally the thermal transition areas require more meshes.



(a) polynomial order



(b) density of polynomial order

Fig. IV-22. Mesh structure of example 2.C

We then compared the *hp*-adaptive results with other types of mesh refinements.

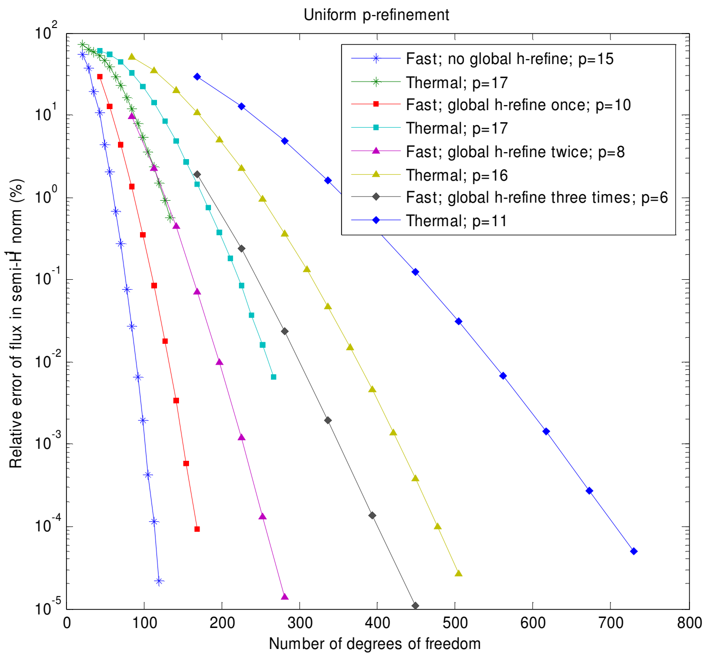


Fig. IV-23. Convergence of uniform *p*-refinement of example 2.C

For the fast group, we do not have to perform global *h*-refinement along with uniform *p*-refinement, but for the thermal group, if we wish the error to be below $10^{-4}\%$, we have to perform some global *h*-refinement twice before proceeding with the *p*-refinement. We obtain exponential convergence for uniform *p*-refinement as shown in Fig. IV-23. However, number of degrees of freedom is larger than what was need for *hp*-adaptation to reach the same accuracy and it is larger than in the *hp*-adaptation case where we did not have to know how many initial elements were required to yield a good enough starting point for the computation.

As expected, uniform *h*-refinement can only deliver algebraic convergence. From Fig. IV-24 we can see that using *p*=3 yields obviously a much better convergence than linear elements, but it still needs one order of magnitude more DOFs to reach a similar convergence of 0.1% compared with *hp*-refinement. The advantages of *hp*-refinement compared to uniform *h*-refinement in this example are obvious.

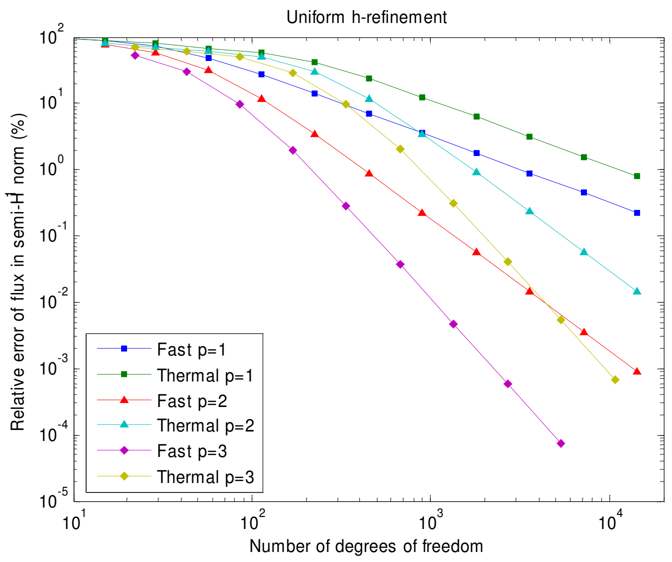


Fig. IV-24. Convergence of uniform *h*-refinement of example 2.C

The results of *h*-refinement in Fig. IV-25 confirm their algebraic convergence.

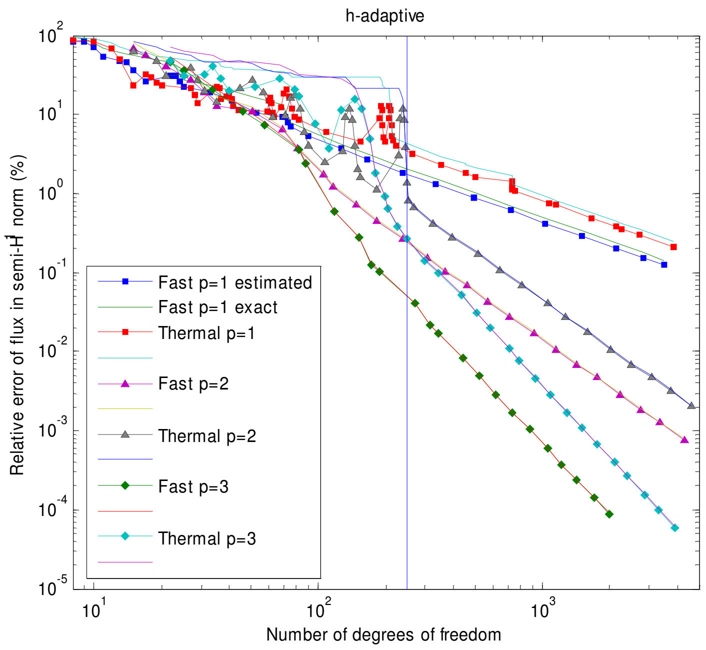


Fig. IV-25. Convergence of adaptive *h*-refinement

The local effectivity index of the error estimator, when using a global *h*-refined solution as reference, is not equal to 1 for linear elements. But, in the asymptotic range, the error estimator can still delivers a good performance.

Put all these convergence sequences together on Fig. IV-26, we obtain:

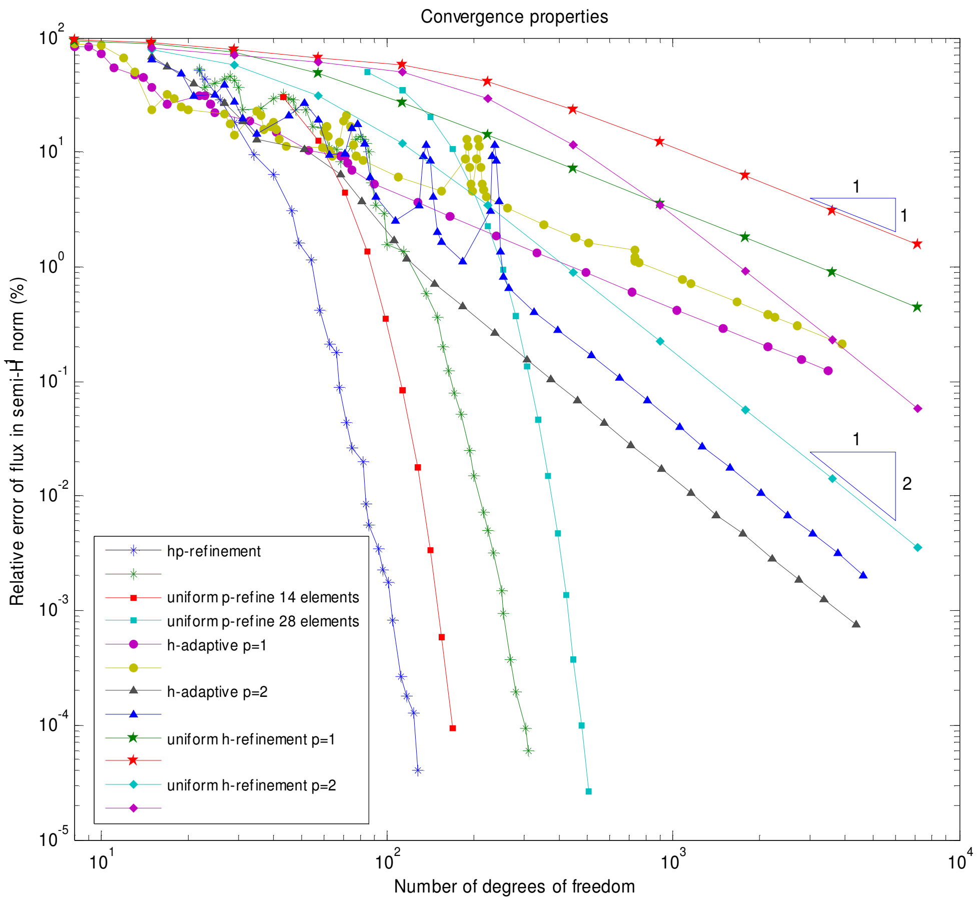


Fig. IV-26. Convergence of different refinement strategies of example 2.C

The advantage of *hp*-refinement is clear. It needs to be pointed out that *hp*-adaptive calculations make highly accurate solution possible.

4.5.4 Two-group energy-driven source problem with fission

4.5.4.1 Example 3 (Problem with piece-wise constant material properties and extraneous source)

Description:

Same configuration as in example 2.C, except that this example includes fission. All fission neutrons are born in fast i.e. χ_1=1.0, χ_2=0.0.

Material properties:

Material #	D_1 (cm)	D_2 (cm)	Σ_{r1} (cm^{-1})	Σ_{r2} (cm^{-1})	$\Sigma_{s1\to2}$ (cm^{-1})	$\nu\Sigma_{f1}$ (cm^{-1})	$\nu\Sigma_{f2}$ (cm^{-1})
1	1.2	0.4	0.03	0.1	0.02	0.005	0.1
2	1.2	0.4	0.03	0.2	0.015	0.0075	0.1
3	1.2	0.4	0.03	0.25	0.015	0.0075	0.1

Again, we have,

$$\xcancel{-D_1\frac{d^2\phi_1(x)}{dx^2}}+\Sigma_{r1}\phi_1(x)=S+\nu\Sigma_{f1}\phi_1(x)+\nu\Sigma_{f2}\phi_2(x)$$

$$\xcancel{-D_2\frac{d^2\phi_2(x)}{dx^2}}+\Sigma_{a2}\phi_2(x)=\Sigma_{12}\phi_1(x)$$

$$\phi_1=\frac{S}{\Sigma_{r1}-\nu\Sigma_{f1}-\dfrac{\nu\Sigma_{f2}\Sigma_{12}}{\Sigma_{a2}}};\qquad \phi_2=\frac{\Sigma_{12}}{\Sigma_{a2}}\phi_1$$

Calculation conditions:

1 linear initial element per assembly; p_{max}=17; α_1=1/3, α_2=0.9; energy-driven case.

Results:

Because of the existence of fission, power iterations are needed. After 8 power iterations, the mesh iterations converged to the same mesh within a tolerance of 10^{-5}% with 37 iterations for the fast flux and 63 for thermal flux. Convergence sequence of the 8th power iteration is illustrated in Fig. IV-27.

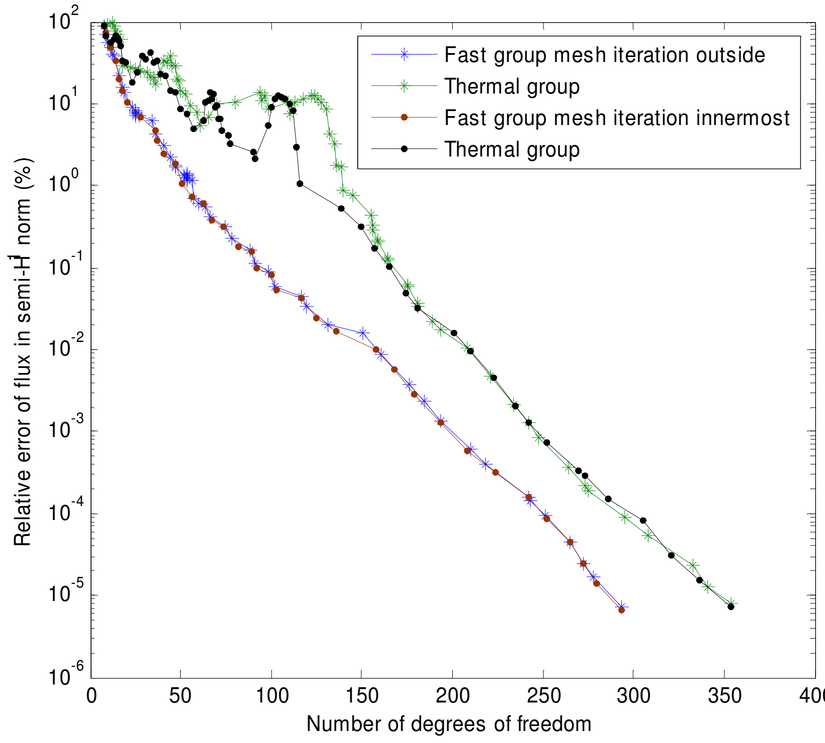


Fig. IV-27. Convergence sequences of example 3

In the case where the mesh iteration outside is implemented outside the power iteration, 76 iterations are needed to converge the fluxes below 10^{-5}%, while 58 iterations are needed in the pre-asymptotic range. No matter where the mesh iteration is implemented, the resulting meshes at all steps provide almost exactly the same convergence path.

We also tested different weighting factors. Putting higher weighting factor in the thermal group results in a significant increase of DOFs for the thermal group for the same error tolerance, whereas this influence on the fast group is not so obvious as in the red curve in Fig. IV-28. Using the group flux norm as weighting factors is a good balance which can make both groups converge along the nearly optimal curve.

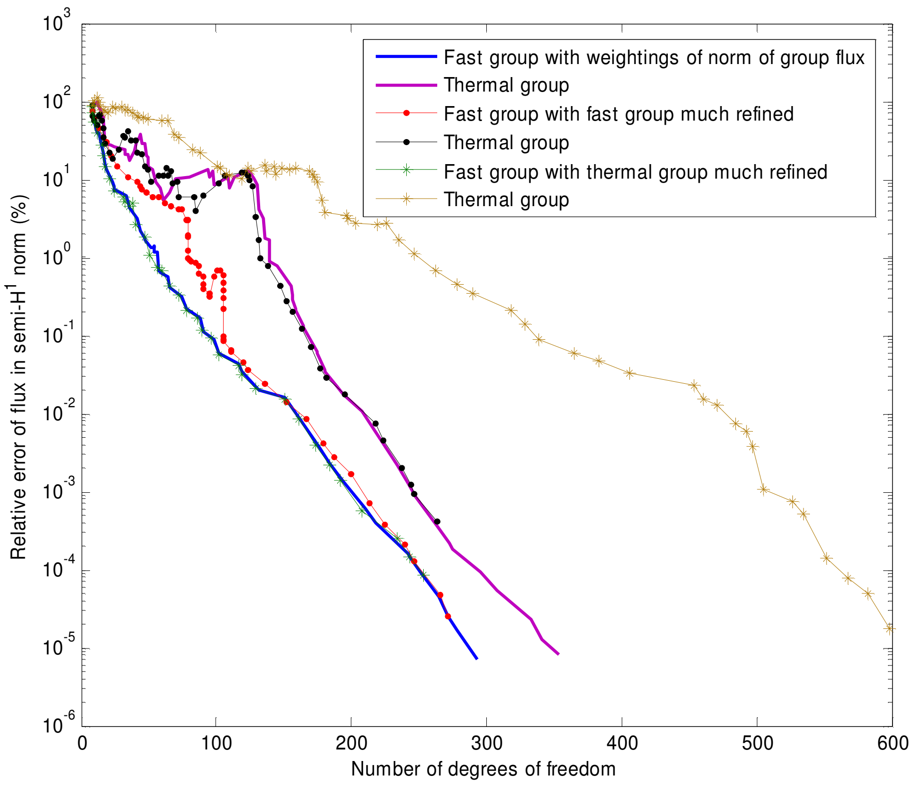
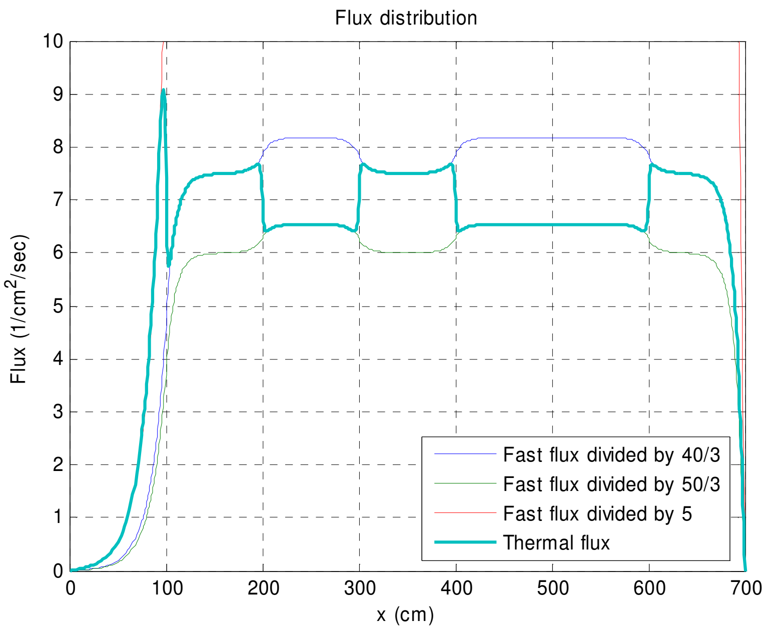


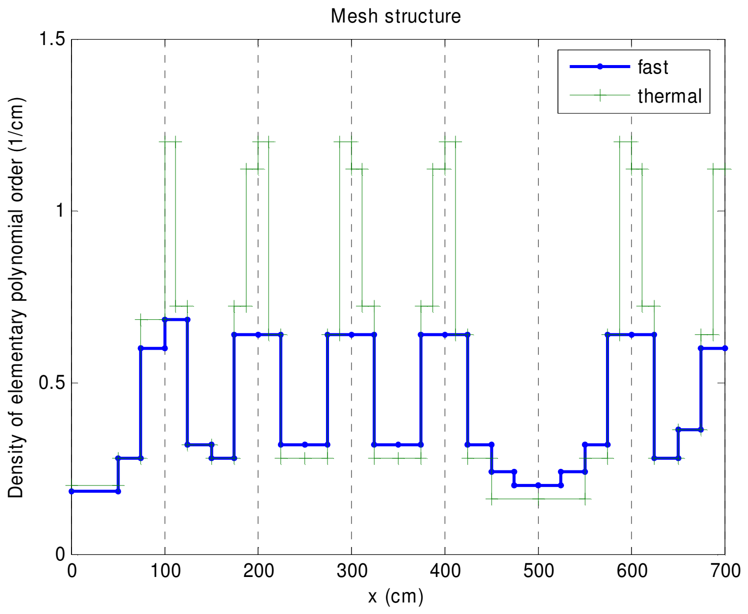
Fig. IV-28. Convergence results with different group weighting factors of example 3

Flux distribution and mesh structure are given in Fig. IV-29.

Comparing with example 2.C, this problem is more tightly coupled. A greater number of DOFs density is needed for both the fast and thermal groups of this problem. Again, the errors tend to be even distributed as illustrated in Fig. IV-30.



(a)



(b)

Fig. IV-29. Solutions with E_1=7.2×10^{-5}% and E_2=8.0×10^{-5}% of example 3: (a) flux and (b) mesh structure

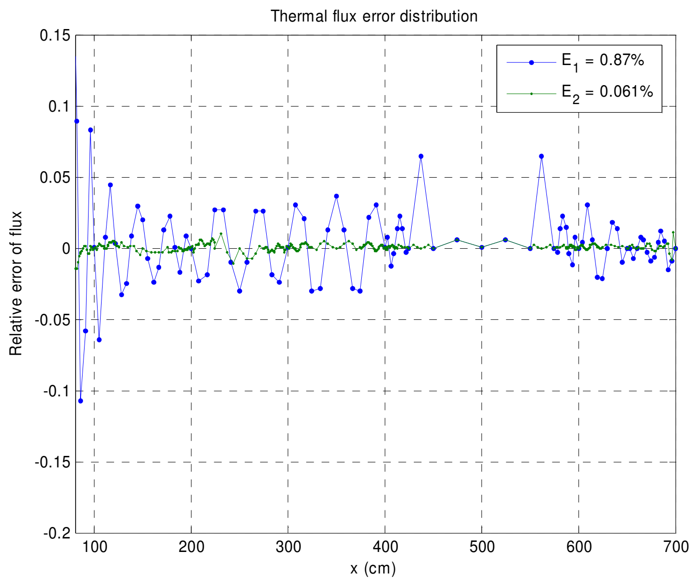


Fig. IV-30. Error distribution of the thermal flux error, example 3

4.5.5 Two-group goal-oriented source problem

4.5.5.1 Example 4.A (A problem with analytical solution)

Description:

Same as example 2.A, but here we are only concerned with the thermal flux integrated between a specified spatial range [a_1, a_2].

We know the analytical solution, given below:

$$\phi_1(x) = \frac{Q}{\Sigma_{r1}}(1 - \frac{\cosh(x/L_1)}{\cosh(a/L_1)})$$

$$\phi_2(x) = \frac{Q\Sigma_{12}}{\Sigma_{r1}\Sigma_{a2}}(1 - \frac{L_1^2}{L_1^2 - L_2^2}\frac{\cosh(x/L_1)}{\cosh(a/L_1)} + \frac{L_2^2}{L_1^2 - L_2^2}\frac{\cosh(x/L_2)}{\cosh(a/L_2)})$$

$$L_1 = \sqrt{\frac{D_1}{\Sigma_{r1}}} \qquad L_2 = \sqrt{\frac{D_2}{\Sigma_{a2}}}$$

Then the quantity of interest is:

$$I(\phi) = \int_{a_1}^{a_2} \phi_2(x)dx = \frac{Q\Sigma_{12}}{\Sigma_{r1}\Sigma_{a2}}(x - \frac{L_1^3}{L_1^2 - L_2^2}\frac{\sinh(x/L_1)}{\cosh(a/L_1)} + \frac{L_2^3}{L_1^2 - L_2^2}\frac{\sinh(x/L_2)}{\cosh(a/L_2)})\Big|_{a_1}^{a_2}$$

Substituting material properties and letting a_1=60cm and a_2=80cm, yields an exact value of

134.9238787715397.

Calculation conditions:

Two linear initial mesh [0 60]cm and [60 80]cm; p_{max}=17; α_1=1/3, α_2=0.9; goal-oriented case.

Results:

The "exact" value calculated from Eq. (4.28) with an energy-driven solution is 134.9238787714360, with flux errors $E_1 = 3.5\times10^{-6}\%$ $E_2 = 5.3\times10^{-6}\%$, which has 13 digits accuracy. Its error is $8.8\times10^{-5}\%$. With same number of DOFs, a goal-oriented calculation reached an error of $4.4\times10^{-6}\%$ and gave a value equal to 134.923878771539. The convergence sequence is provided in Fig. IV-31.

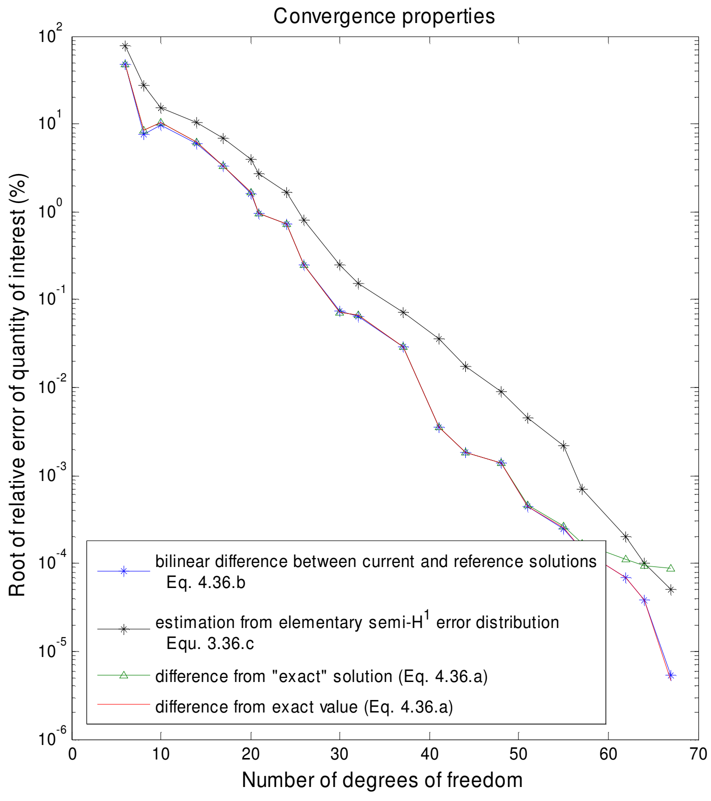


Fig. IV-31. Convergence sequence of goal-oriented calculation

Black points represent the error calculated with Eq. (4.36.c). Blue points are the error with Eq. (4.36.b). And green and red points are obtained with Eq. (4.36.a). The ratio between $\overset{\frown}{err}$ and $\overline{err}$ is nearly constant in the asymptotic range. This calculation showed that the elementary error of Eq. (4.34) can effectively control the convergence process.

The error distribution for the goal-oriented calculation is given in Fig. IV-32.

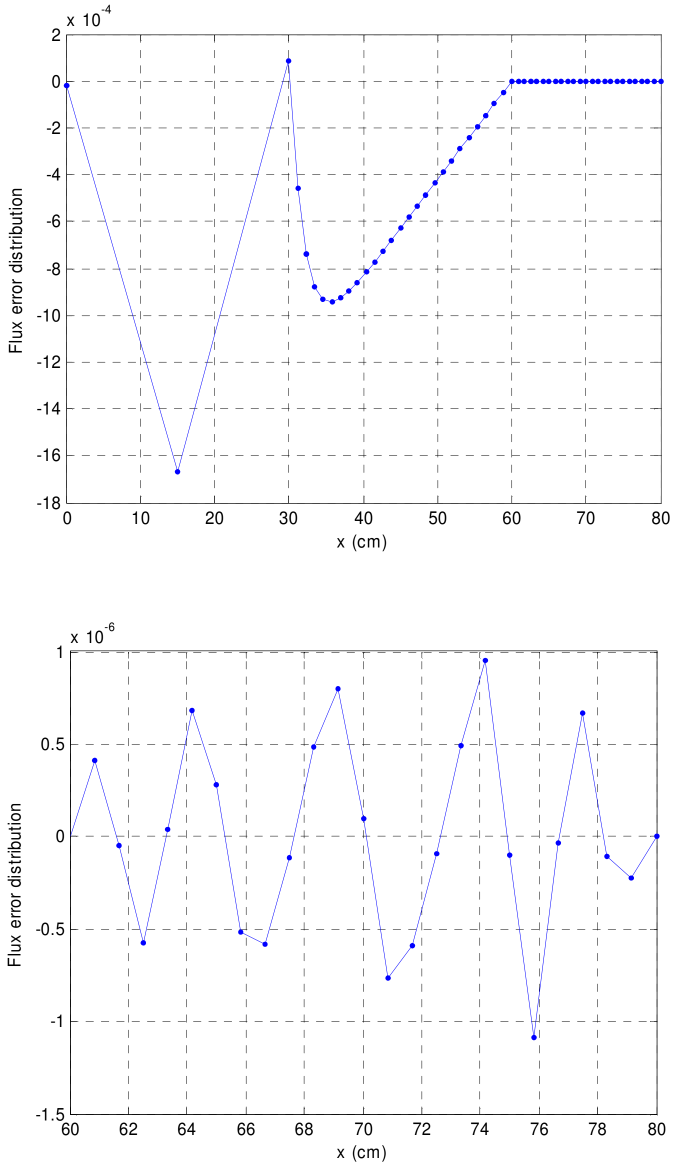


Fig. IV-32. Error distribution of thermal flux in a goal-oriented calculation

We can see clearly the goal-oriented calculation only yield accurate results in the region of interest, while the error of the thermal flux in energy-driven calculation tends to be even-distributed in the whole domain, as shown in Fig. IV-33.

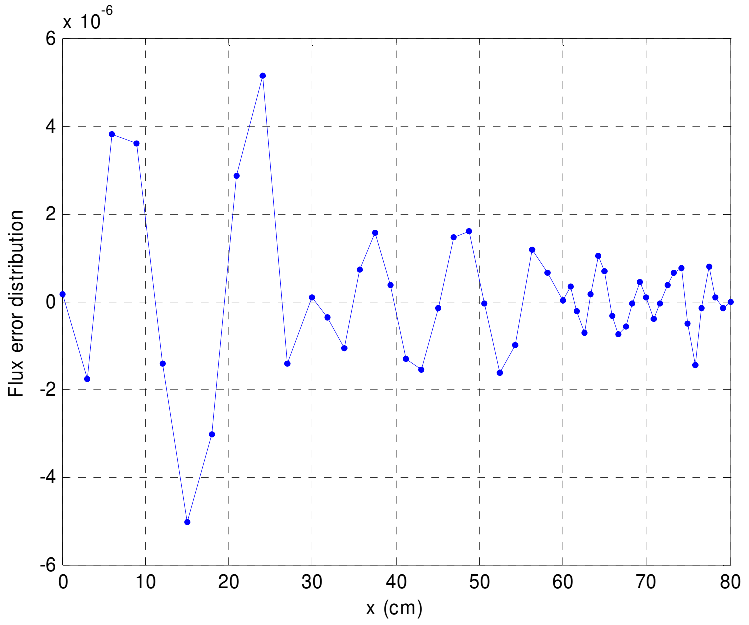


Fig. IV-33. Error distribution of the thermal flux in an energy-driven calculation

We can also see the difference of mesh structure between these two calculations in TABLE IV-II. The region spanning from 30 to 90 cm received more refinements.

TABLE IV-II

Mesh structure of example 4.A

Energy group	Left vertex	Right vertex	Polynomial order	
			Energy-driven	Goal-oriented
1	0	30	5	5
	30	60	8	9
	60	80	8	9
2	0	30	5	1
	30	60	8	10
	60	80	12	12

A series of energy-driven calculation was conducted with the fast group fully converged to investigate how the thermal flux accuracy affects the accuracy of quantity of interest. Fig. IV-34 illustrates their relationship. These two accuracies are nearly proportional to each other.

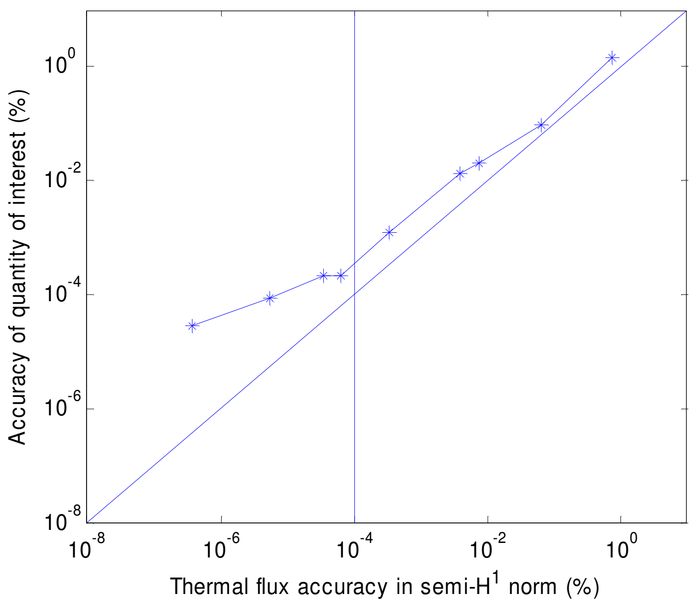


Fig. IV-34. Relationship between thermal flux accuracy and accuracy of quantity of interest

Experiments showed that with double precision the accuracy of quantity can reach 15 digits.

4.5.5.2 Example 4.B (Piece-wise constant material properties and extraneous source)

Description:

2-group 1-D source problem same as in example 2.C; Assembly-wide piecewise constant thermal adjoint volumetric source is 0-1-1-0-0-0-0 cm^{-1} and fast adjoint source is zero everywhere, which means we are only concerned with the integral of the thermal flux in the second and third assembly.

Calculation conditions:

1 linear initial cell per assembly; p_{max}=17; α_1=1/3, α_2=0.9; goal-oriented case.

Results:

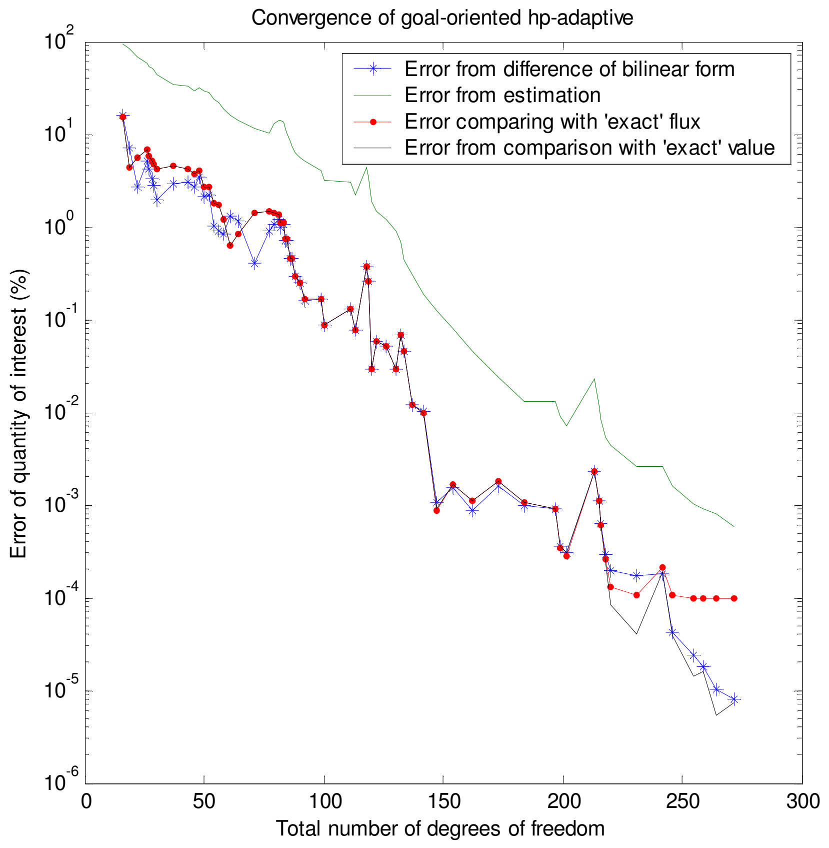
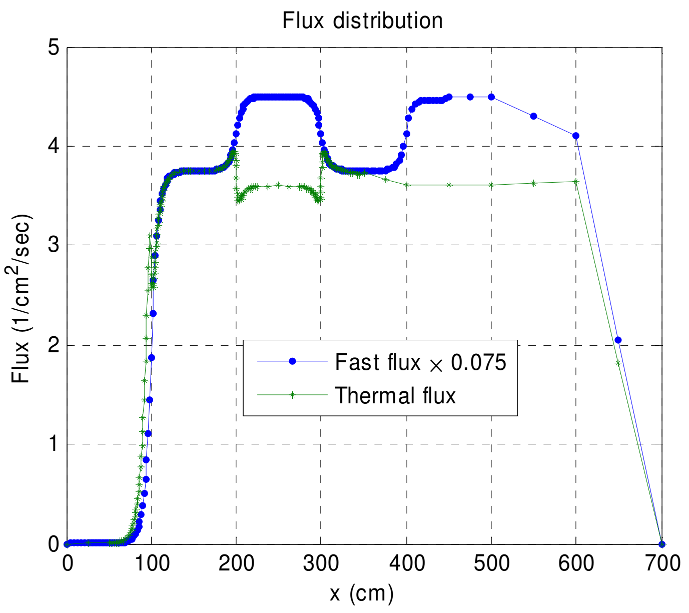


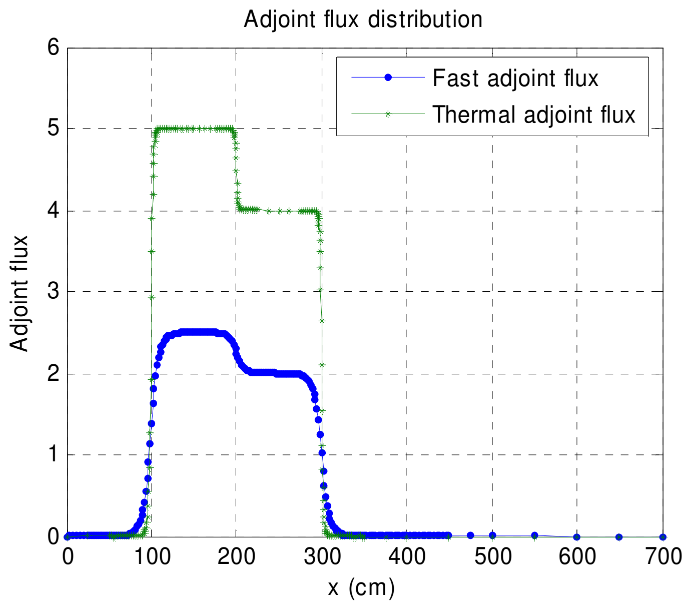
Fig. IV-35. Convergence sequence of goal-oriented calculation

In Fig. IV-35, the ratio between green curve and blue curve is nearly constant. We can control the adaptive procedure with the error estimation Eq. (4.32). Bilinear estimation is indeed a good estimation of real error of quantity of interest. The accuracy calculated with reference flux is about 10^{-4}% with flux tolerance E_1=7.2×10^{-6}%, E_2=7.6×10^{-6}%. At this time the number of DOFs is 159+377, however with goal-oriented calculation, we only need 104+138 to get accuracy of 4.3×10^{-5}%.

With accuracy below 10^{-5}%, the number of DOFs is 116+156. The values of last two iteration are 722.961048869942 and 722.961048869944. We can get a really accurate estimation! (We let this value the “exact” quantity of interest.) Directand adjoint flux distributions are shown in Fig. IV-36.



(a)



(b)

Fig. IV-36. Flux distribution from goal-oriented calculation

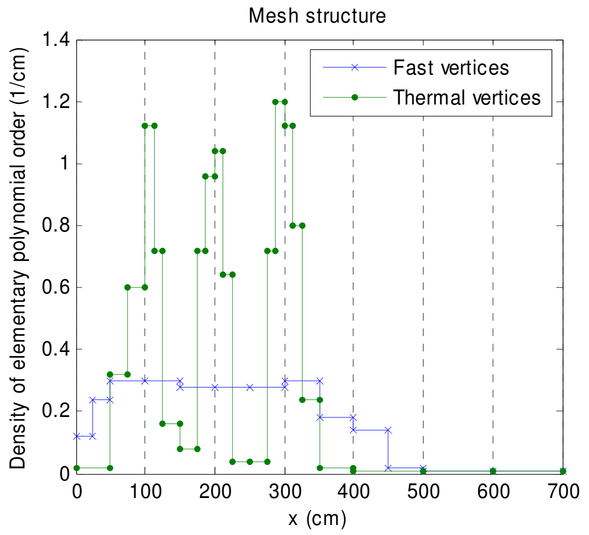


Fig. IV-37. Mesh structure from goal-oriented calculation

Mesh structure is in Fig. IV-37. Goal-oriented calculation indeed refined where really needed. And we can also notice we need relative more DOFs in fast group than the number of energy-driven calculation to obtain more accurate quantity of interest.

Then, a series of energy-driven calculations with different fast flux accuracy and thermal accuracy was conducted in order to show how these accuracies affect the accuracy of quantity of interest. Results are plotted in Fig. IV-38.

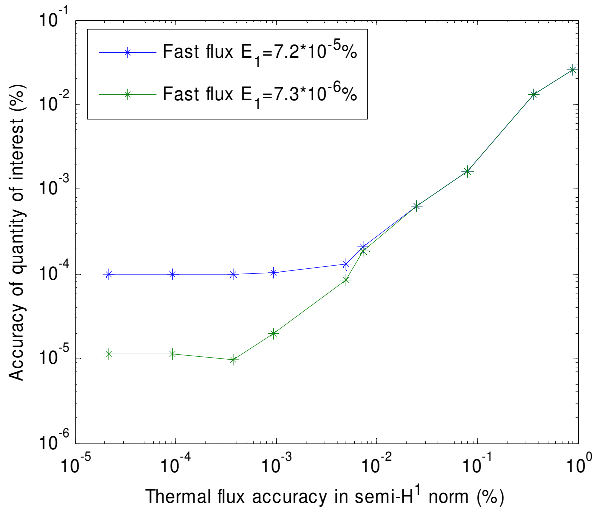


Fig. IV-38. Relationship between flux accuracy and accuracy of quantity of interest

Relative error of fast flux gives more influence on the accuracy of quantity of interest. This also explained why we need 116+156 but energy-driven only 159+377, ratio of fast group is much higher than thermal. Linear relationship between flux accuracy and accuracy of quantity of interest is also observed.

4.5.5.3 Example 4.C (Piece-wise constant material properties and external source with fission)

Description:

2-group 1-D source problem same as example 3; Assembly-wide piecewise constant thermal adjoint volumetric source 0-1-1-0-0-0-0 cm^{-1}, which means we are only concerned with the integral of the thermal flux in the second and third assemblies.

Calculation conditions: 1 linear initial mesh per assembly; p_{max}=17; α_1=1/3, α_2=0.9; goal-oriented case.

Results:

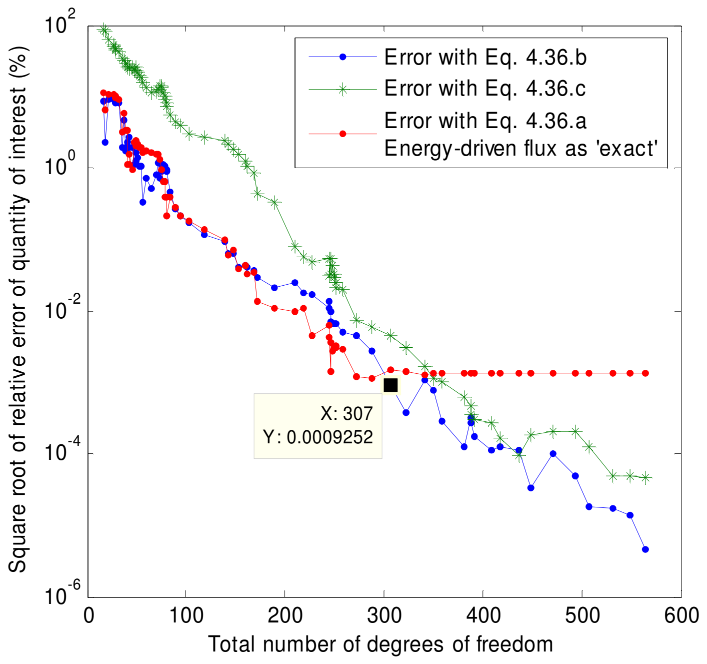
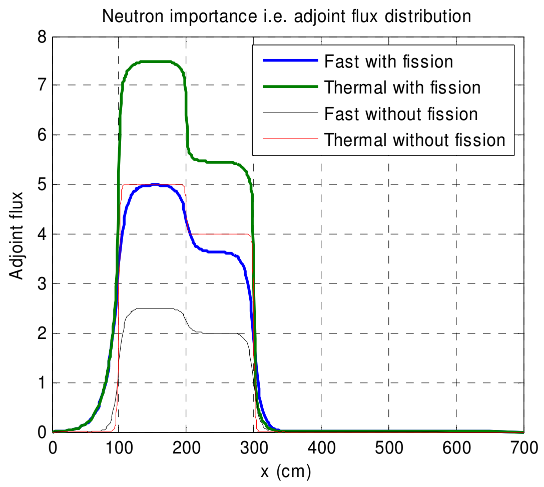


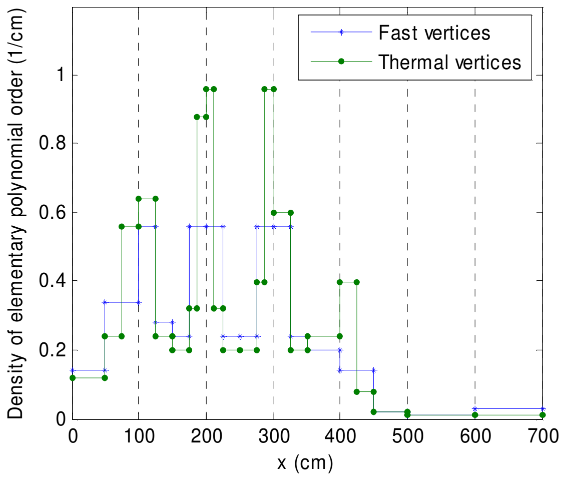
Fig. IV-39. Convergence sequence of goal-oriented calculation

The convergence sequence is shown in Fig. IV-39. Exponential convergence is observed. With the energy-driven calculation, we needed 293+354 DOFs to reach $10^{-3}\%$. With the goal-oriented calculation, the DOFs are 148+159 to reach a better solution of accuracy $9.3\times10^{-4}\%$. Note that the quantity of interest calculated with bilinear form is different from the flux integral due to the insufficient convergence of power iteration.

The adjoint flux and result mesh structure are provided in Fig. IV-40.



(a) adjoint flux



(b) mesh structure

Fig. IV-40. Result of goal-oriented calculation

Comparing with example 4.B where there was no fission, the thermal and fast neutron importance increased. Because of fission, the transition range of the thermal neutron importance is larger. The mesh density ratio between the fast and the thermal fluxes increased and more DOFs are needed in the thermal group at the adjacent assemblies 1 and 4.

4.5.6 A two-group eigenvalue problem

4.5.6.1 Example 5 (a two group eigenvalue problem)

Description:

2-group 1-D eigenvalue problem; ten 40-cm thick assemblies with three types are arranged in 1-2-1-2-1-2-1-2-1-3; all neutrons are born fast; each type of assembly corresponds to a single material numbered from 1 to 3; 1 and 2 are two different fuel assemblies (fissile material) and 3 is a reflector assembly; Homogeneous Neumann condition hold at the left, boundary condition at the right is zero flux;

Material properties:

Material #	D_1 (cm)	D_2 (cm)	Σ_{r1} (cm^{-1})	Σ_{r2} (cm^{-1})	$\Sigma_{s1\to 2}$ (cm^{-1})	$\nu\Sigma_{f1}$ (cm^{-1})	$\nu\Sigma_{f2}$ (cm^{-1})
1	1.2	0.4	0.03	0.3	0.015	0.0075	0.45
2	1.2	0.4	0.03	0.25	0.015	0.0075	0.375
3	1.2	0.2	0.051	0.04	0.05	0	0

No theoretical solution is available. Nonetheless, we can relate the fast and thermal fluxes ratios in the fundamental mode away from the interfaces.

$$\xcancel{-D_2\frac{d^2\phi_2(x)}{dx^2}} + \Sigma_{a2}\phi_2(x) = \Sigma_{12}\phi_1(x)$$

$$\phi_2(x) = \frac{\Sigma_{12}}{\Sigma_{a2}}\phi_1(x)$$

Calculation conditions:

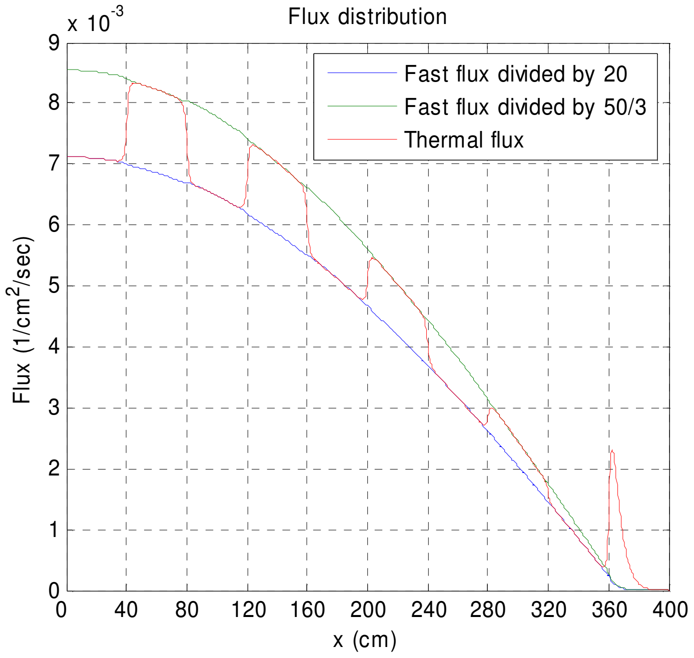
1 quadratic initial mesh per assembly; p_{max}=17; α_1=1/3, α_2=0.9

Results and discussion:

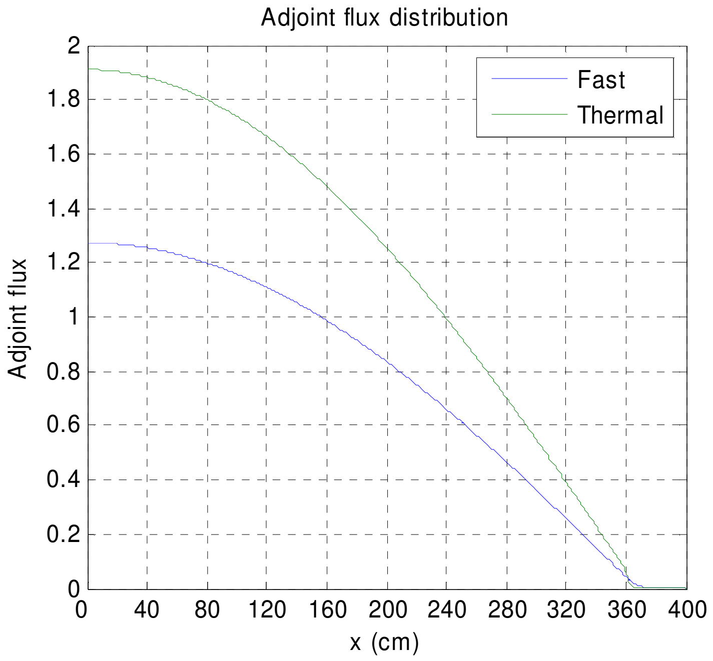
With mesh iteration implemented inside the power iteration (i.e., wrapped around the one-group solver), 2544 power iterations are needed to obtain a successive error in k_{eff} less than 10^{-1} . The final eigenvalue is k_{eff}=0.9992527638. At the 7th power iteration, we began generating the same mesh as in the 6th power iteration. 30 power iterations later, the mesh iteration result became fully stable. At this time, the number of DOFs of resulting fast mesh is 281 with error E_1=6.6×10^{-5}% and thermal 319 - 7.6×10^{-5}%.

For the implementation of the mesh iteration loop outside the power iteration, we obtained k_{eff}=0.99925276396 after 54 iterations, with number of DOFs 322 and 450 in fast and thermal group respectively with both relative error less than 5×10^{-6}%. Normalized flux distributions are given in Fig. IV-41.

The semi-H^1 norm of fast flux is 6.77×10^{-5}, while thermal norm is 3.44×10^{-6}. Their difference is similar as the one of L_2 or L_∞ norm.



(a) Direct flux



(b) adjoint flux

Fig. IV-41. Flux distributions of example 5

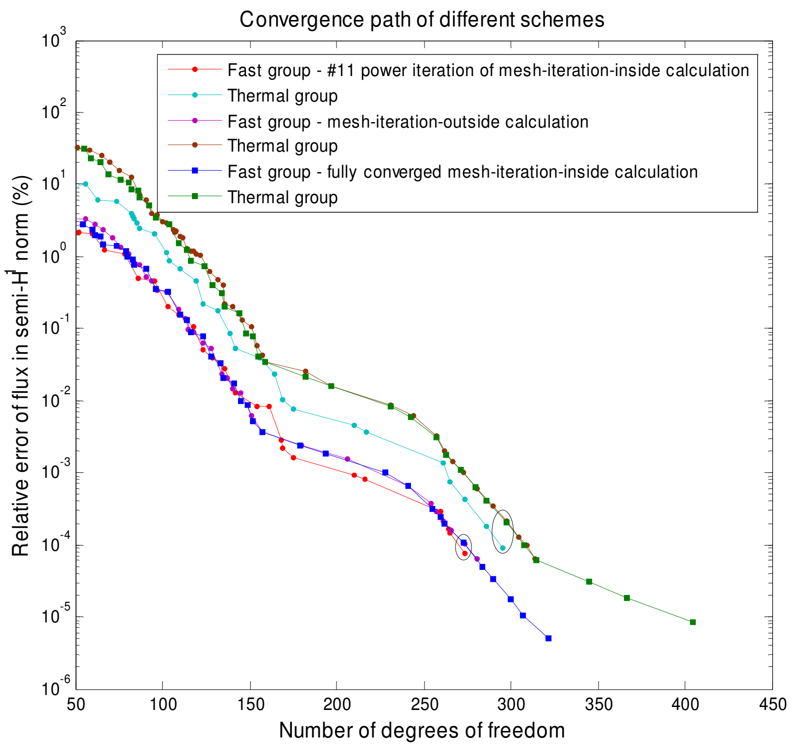
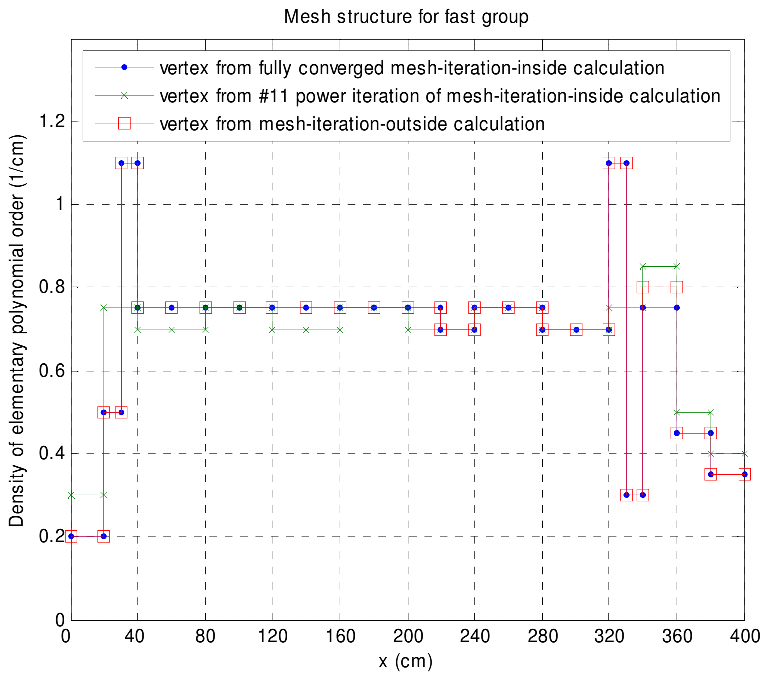
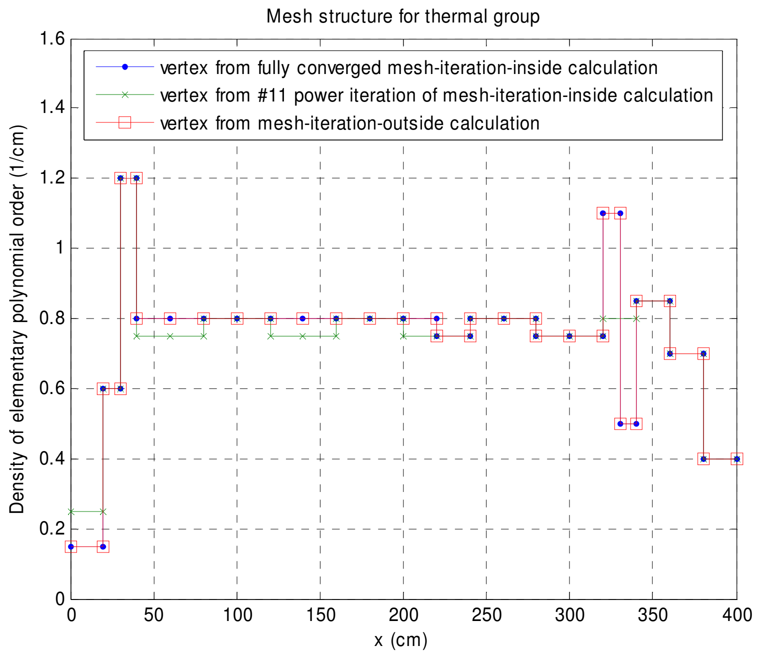


Fig. IV-42. Convergence sequence of example 5

Convergence sequence is shown in Fig. IV-42. We can see that mesh-iteration-inside scheme and mesh-iteration-outside scheme deliver same convergence sequence. When doing mesh iteration inside power iteration, meshes generated in #11 power iteration are slightly different from the converged meshes. We can see these in figure 42. Three sets of meshes are marked in Fig. IV-42. The two sets of meshes are nearly identical. We need more *h*-refinement in the central assembly and ay the assembly near the reflector. We can also see that from Fig. IV-43. more *h*-refinements are needed at the central assembly and at the assembly near the reflector.



(a) fast group



(b) thermal group

Fig. IV-43. Mesh structure of example 5

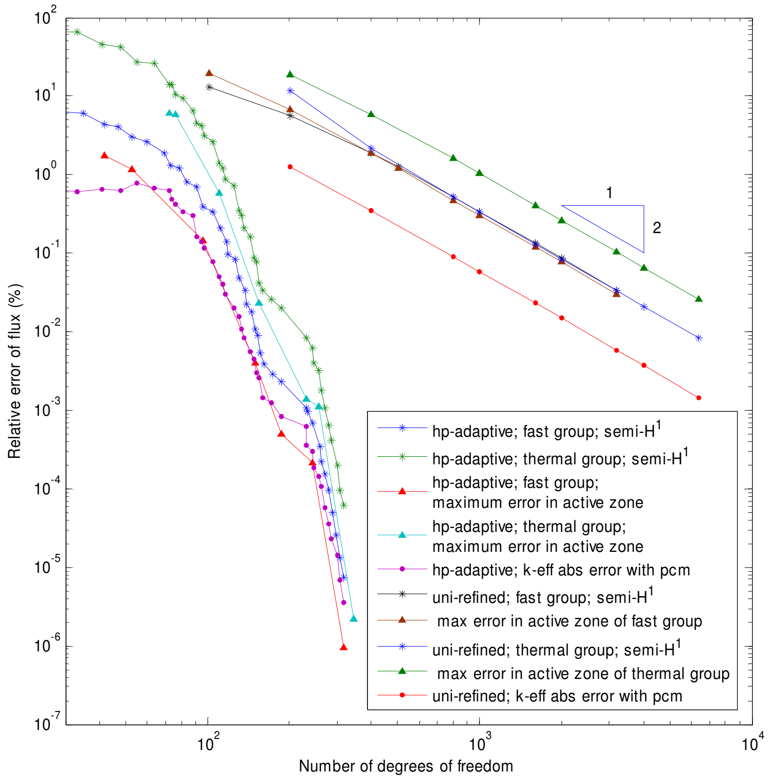
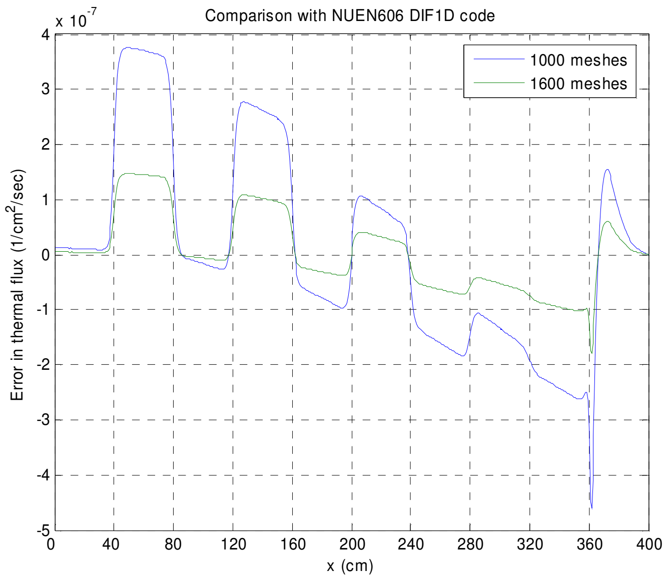


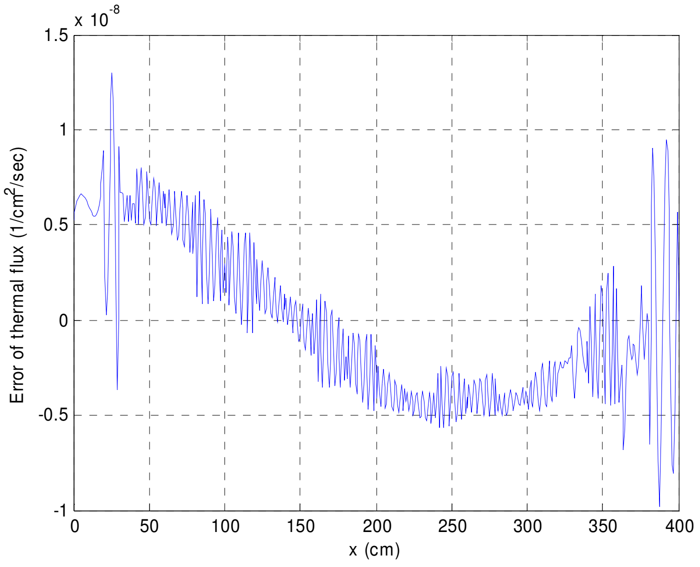
Fig. IV-44. Comparison of *hp*-adaptive and uniform *h*-refinement of an eigenvalue problem

Then, some comparisons with a 1-D diffusion code (DIFF1D) developed in the NUEN606 course DIFF1D were carried out. DIFF1D is a code developed in NUEN 606 with a coarse mesh method, in which quadratic polynomial are used to describe the fast flux distribution within an element, and the thermal flux in a mesh is represented with a quadratic particular solution corresponding to the fast flux and two homogeneous hyperbolic solutions corresponding to the interface fluxes. The meshes are uniformly refined in DIFF1D calculations.

We can clearly note with Fig. IV-44 that DIFF1D is a second order method. Again, *hp*-adaptive delivers exponential convergence. The thermal flux error distribution is shown in Fig. IV-45. The convergence of k_{eff} can not be treated as similar magnitude of convergence of flux everywhere, for example, when the fast number of degrees of freedom is equal to 100 and thermal one is equal to 200, maximum relative error of thermal flux in fuel assemblies is nearly equal to 20% which is large but the error in k_{eff} is only about 1 pcm.



(a) DIF1D



(b) *hp*-adaptation

Fig. IV-45. Thermal flux error distributions, example 5

We also note that some flux modes flux are damped out very slowly in uniform refinement. But in *hp*-adaptive calculations, the error tends to be even-distributed. This is be why maximum error in active zone (fuel assemblies) is larger than error in semi-H^1 norm in uniform refinement calculation but is smaller in *hp*-adaptation.

Goal-oriented calculation was also completed. The convergence sequence is shown in Fig. IV-46.

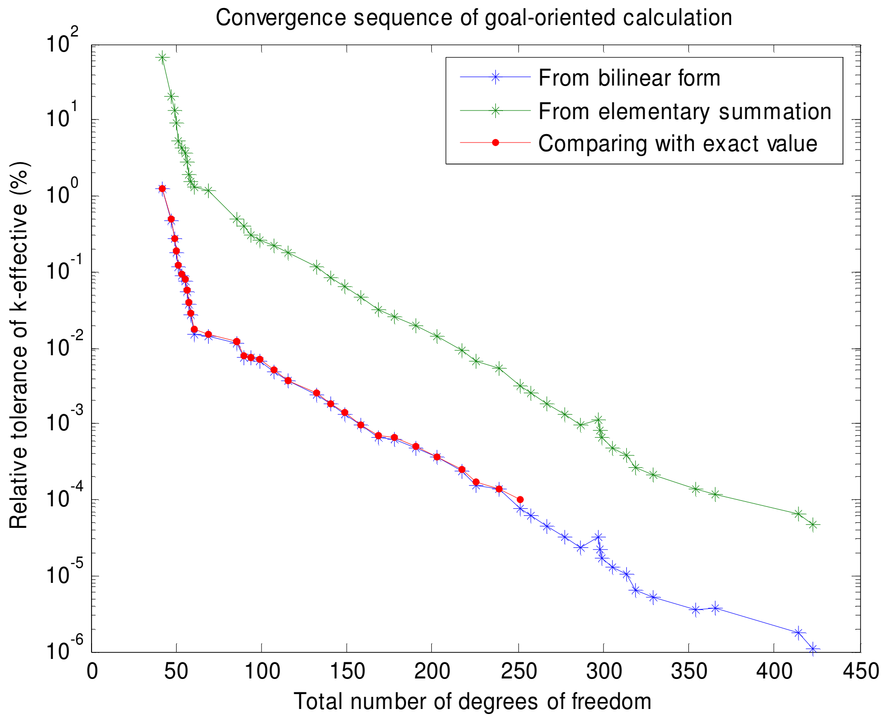


Fig. IV-46. Convergence sequence of goal-oriented calculation of example 5

The x-coordinate of Fig. IV-46 is the total number of degrees of freedom (thermal+ fast). Note that we can get k_{eff}=0.999252763970375 with 15 effective digits! The flux errors are compared with energy-driven calculation on Fig. IV-47.

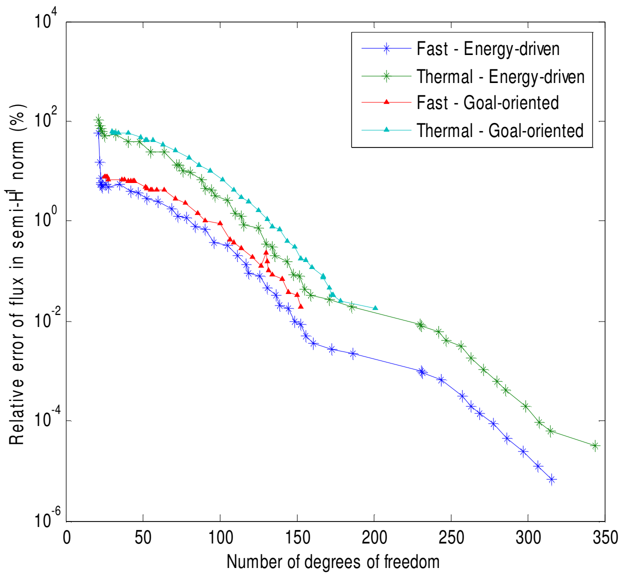


Fig. IV-47. Comparison of goal-oriented and energy-driven calculations

Goal-oriented calculation indeed did not create optimal mesh in sense of energy of flux but optimal for calculation of eigenvalue.

Meshes structure delivered with goal-oriented calculation is shown in Fig. IV-48. This mesh corresponds to the 41th iteration, which is the last number 5 in the Fig. IV-46. In this problem, we only need to refine two boundary assemblies more to get an optimal estimation of k_{eff}.

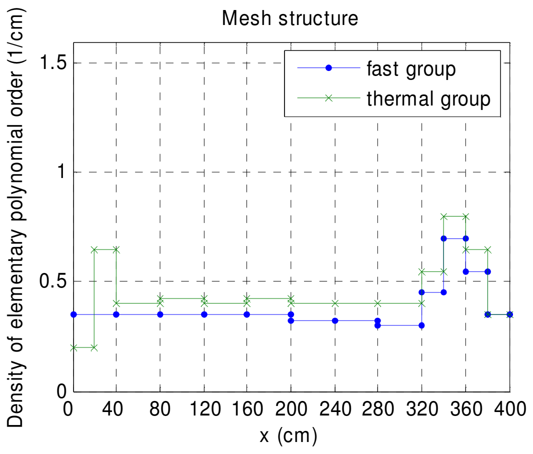


Fig. IV-48. Mesh structure from goal-oriented calculation

Another interesting fact is that the k_{eff} calculated with bilinear form is much more accurate than the one from the classical power iteration. There relationship is shown in Fig. IV-49.

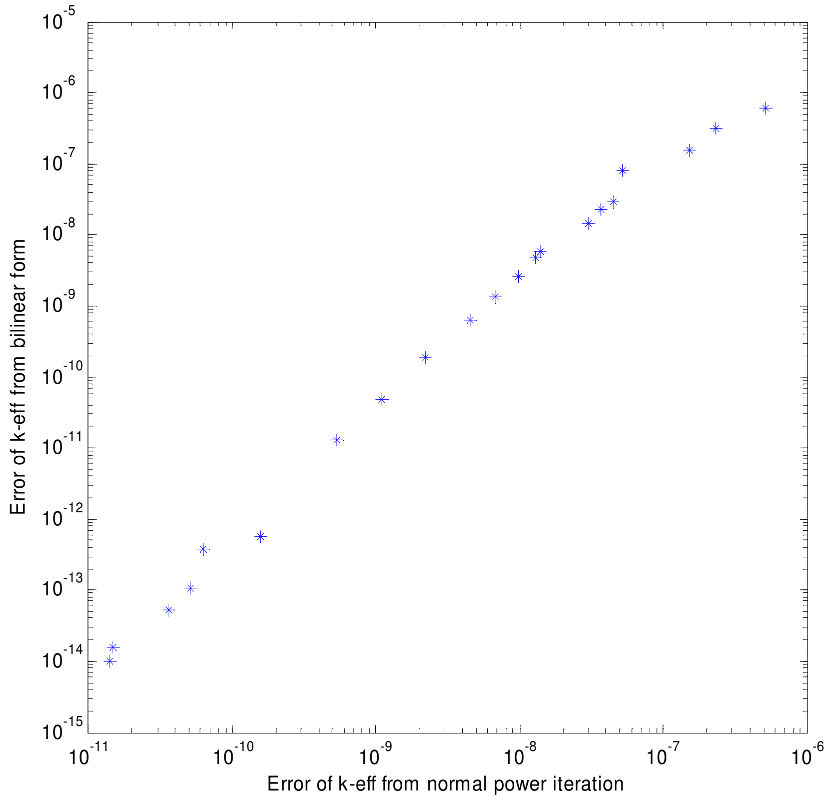


Fig. IV-49. Two ways to calculate k-effective

TABLE IV-III

7-group cross sections

Material number	Group number g	Diffusion Coeffs	Removal	Scattering matrix $\Sigma_{g \to g'}$ (cm^{-1})							Fission	Fission spectr.
		D_g (cm)	$\Sigma_{r,g}$ (cm^{-1})	$g'=1$	2	3	4	5	6	7	$\nu\Sigma_{f,g}$ (cm^{-1})	χ_g
1.	1	1.87320E+00	5.04122E-02	1.27537E-01	4.23780E-02	9.43740E-06	5.51630E-09	0	0	0	2.00600E-02	5.87910E-01
	2	1.01070E+00	5.34880E-03	0	3.24456E-01	1.63140E-03	3.14270E-09	0	0	0	2.02730E-03	4.11760E-01
UO2	3	6.93884E-01	2.94482E-02	0	0	4.50940E-01	2.67920E-03	0	0	0	1.57060E-02	3.39060E-04
fuel-clad	4	6.01286E-01	1.01802E-01	0	0	0	4.52565E-01	5.56640E-03	0	0	4.51830E-02	1.17610E-07
macroscopic	5	1.06906E+00	4.04003E-02	0	0	0	1.25250E-04	2.71401E-01	1.02550E-02	1.00210E-08	4.33421E-02	0
cross-sections	6	8.43523E-01	1.29366E-01	0	0	0	0	1.29680E-03	2.65802E-01	1.68090E-02	2.02090E-01	0
	7	5.90591E-01	2.91326E-01	0	0	0	0	0	8.54580E-03	2.73080E-01	5.25711E-01	0
2.	1	1.86500E+00	4.98551E-02	1.28876E-01	4.14130E-02	8.22900E-06	5.04050E-09	0	0	0	2.17530E-02	5.87910E-01
	2	1.00751E+00	5.39720E-03	0	3.25452E-01	1.63950E-03	1.59820E-09	0	0	0	2.53510E-03	4.11760E-01
4.3% MOX	3	6.89030E-01	3.05842E-02	0	0	4.53188E-01	2.61420E-03	0	0	0	1.62680E-02	3.39060E-04
fuel-clad	4	5.87970E-01	1.09749E-01	0	0	0	4.57173E-01	5.53940E-03	0	0	6.54741E-02	1.17610E-07
macroscopic	5	7.82056E-01	1.49413E-01	0	0	0	1.60460E-04	2.76814E-01	9.31270E-03	9.16560E-09	3.07241E-02	0
cross-sections	6	4.90920E-01	4.26035E-01	0	0	0	0	2.00510E-03	2.52962E-01	1.48500E-02	6.66651E-01	0
	7	4.88149E-01	4.17845E-01	0	0	0	0	0	8.49480E-03	2.65007E-01	7.13990E-01	0
3.	1	1.83834E+00	5.08662E-02	1.30457E-01	4.17920E-02	8.51050E-06	5.13290E-09	0	0	0	2.38140E-02	5.87910E-01
	2	9.96906E-01	5.94030E-03	0	3.28428E-01	1.64360E-03	2.20170E-09	0	0	0	3.85869E-03	4.11760E-01
7.0% MOX	3	6.75058E-01	3.54141E-02	0	0	4.58371E-01	2.53310E-03	0	0	0	2.41340E-02	3.39060E-04
fuel-clad	4	5.63810E-01	1.27507E-01	0	0	0	4.63709E-01	5.47660E-03	0	0	9.43662E-02	1.17610E-07
macroscopic	5	7.02941E-01	1.91885E-01	0	0	0	1.76190E-04	2.82313E-01	8.72890E-03	9.00160E-09	4.57699E-02	0
cross-sections	6	3.99872E-01	5.83850E-01	0	0	0	0	2.27600E-03	2.49751E-01	1.31140E-02	9.28181E-01	0
	7	3.90502E-01	5.94075E-01	0	0	0	0	0	8.86450E-03	2.59529E-01	1.04320E+00	0
4.	1	1.82105E+00	5.15409E-02	1.31504E-01	4.20460E-02	8.69720E-06	5.19380E-09	0	0	0	2.51860E-02	5.87910E-01
	2	9.89986E-01	6.30190E-03	0	3.30403E-01	1.64630E-03	2.60060E-09	0	0	0	4.73951E-03	4.11760E-01
8.7% MOX	3	6.65991E-01	3.87149E-02	0	0	4.61792E-01	2.47490E-03	0	0	0	2.94781E-02	3.39060E-04
fuel-clad	4	5.49897E-01	1.38153E-01	0	0	0	4.68021E-01	5.43300E-03	0	0	1.12250E-01	1.17610E-07
macroscopic	5	6.63015E-01	2.16983E-01	0	0	0	1.85970E-04	2.85771E-01	8.39730E-03	8.92800E-09	5.53030E-02	0
cross-sections	6	3.61914E-01	6.73414E-01	0	0	0	0	2.39160E-03	2.47614E-01	1.23220E-02	1.07500E+00	0
	7	3.48956E-01	6.99138E-01	0	0	0	0	0	8.96810E-03	2.56093E-01	1.23930E+00	0

TABLE IV-III Continued

7-group cross sections

Material number	Group number g	Diffusion Coeffs	Removal	Scattering matrix $\Sigma_{g\rightarrow g'}$ (cm^{-1})							Fission	Fission spectr.
		D_g (cm)	$\Sigma_{r,g}$ (cm^{-1})	$g'=1$	2	3	4	5	6	7	$\nu\Sigma_{f,g}$ (cm^{-1})	χ_g
5.	1	2.64483E+00	5.98661E-02	6.61659E-02	5.90700E-02	2.83340E-04	1.46220E-06	2.06420E-08	0	0	1.32340E-08	5.87910E-01
	2	1.13704E+00	5.27834E-02	0	2.40377E-01	5.24350E-02	2.49900E-04	1.92390E-05	2.98750E-06	4.21400E-07	1.43450E-08	4.11760E-01
Fission	3	1.17268E+00	1.00825E-01	0	0	1.83425E-01	9.22880E-02	6.93650E-03	1.07900E-03	2.05430E-04	1.12860E-06	3.39060E-04
chamber	4	1.18616E+00	2.01943E-01	0	0	0	7.90769E-02	1.69990E-01	2.58600E-02	4.92560E-03	1.27630E-05	1.17610E-07
macroscopic	5	9.96631E-01	2.34703E-01	0	0	0	3.73400E-05	9.97570E-02	2.06790E-01	2.44780E-02	3.53850E-07	0
cross-sections	6	5.89303E-01	2.48866E-01	0	0	0	0	9.17420E-04	3.16774E-01	2.38760E-01	1.74010E-06	0
	7	2.84380E-01	7.30370E-02	0	0	0	0	0	4.97930E-02	1.09910E+00	5.06330E-06	0
6.	1	2.64483E+00	5.98661E-02	6.61659E-02	5.90700E-02	2.83340E-04	1.46220E-06	2.06420E-08	0	0	0	
	2	1.13704E+00	5.27833E-02	0	2.40377E-01	5.24350E-02	2.49900E-04	1.92390E-05	2.98750E-06	4.21400E-07	0	
Guide tube	3	1.17272E+00	1.00943E-01	0	0	1.83297E-01	9.23970E-02	6.94460E-03	1.08030E-03	2.05670E-04	0	
macroscopic	4	1.18641E+00	2.02109E-01	0	0	0	7.88511E-02	1.70140E-01	2.58810E-02	4.92970E-03	0	
cross-sections	5	9.96691E-01	2.34703E-01	0	0	0	3.73330E-05	9.97372E-02	2.06790E-01	2.44780E-02	0	
	6	5.89303E-01	2.48875E-01	0	0	0	0	9.17260E-04	3.16765E-01	2.38770E-01	0	
	7	2.84378E-01	7.30340E-02	0	0	0	0	0	4.97920E-02	1.09912E+00	0	
7.	1	2.09372E+00	1.14728E-01	4.44777E-02	1.13400E-01	7.23470E-04	3.74990E-06	5.31840E-08	0	0	0	
	2	8.07161E-01	1.30636E-01	0	2.82334E-01	1.29940E-01	6.23400E-04	4.80020E-05	7.44860E-06	1.04550E-06	0	
Moderator	3	5.64675E-01	2.45054E-01	0	0	3.45256E-01	2.24570E-01	1.69990E-02	2.64430E-03	5.03440E-04	0	
macroscopic	4	5.70434E-01	4.93322E-01	0	0	0	9.10284E-02	4.15510E-01	6.37320E-02	1.21390E-02	0	
cross-sections	5	4.64253E-01	5.78862E-01	0	0	0	7.14370E-05	1.39138E-01	5.11820E-01	6.12290E-02	0	
	6	2.65721E-01	5.54537E-01	0	0	0	0	2.21570E-03	6.99913E-01	5.37320E-01	0	
	7	1.25768E-01	1.69679E-01	0	0	0	0	0	1.32440E-01	2.48070E+00	0	

4.5.7 Seven-group problems

4.5.7.1 Example 6.A (7-group source problem without fission)

Problem description:

- 7-group 1-D source problem;
- Four 100-cm thick assemblies with four different materials are arranged in 1-2-3-7;
- Each type of assembly corresponding to a single material numbered from 1 to 7; (Material properties are referred to TABLE IV-III.) Fission cross sections are set to zero;
- Volumetric extraneous source for the first group is 1.0cm^{-3}sec^{-1}; No sources for all other groups;
- Volumetric extraneous adjoint source in group 7 and the second assembly from 100cm to 200cm is 1.0; zero for all other assemblies and all other groups.
- Left homogeneous Neumann boundary condition and zero flux right boundary;

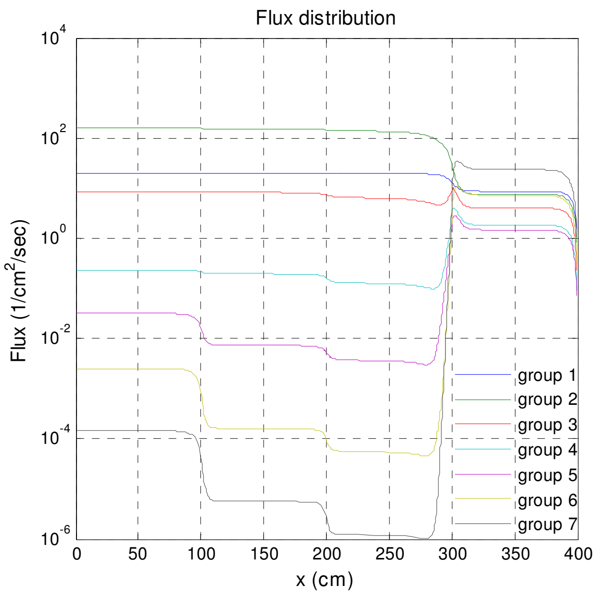
The material data are obtained from a benchmark problem on deterministic transport calculations, which was proposed by OECD/NEA Expert Group on 3-D Radiation Transport Benchmarks in 2001. This seven-group set of cross-sections was chosen to enhance the multigroup difficulties of heterogeneous problems. This example problem contains four different materials and was designed to show that the code can work with multigroup neutron diffusion and can deliver similar results as the results in 2-group calculations.

Calculation conditions:

Initial mesh: 4 linear elements; Lobatto shape functions; p_{max}=17; α_2=0.9; number-based refinement stratergy with α_3=1/4.

Results:

"exact" direct and adjoint flux are obtained with 62 and 59 mesh iterations. Direct and adjoint flux distributions are plotted in Fig. IV-50.



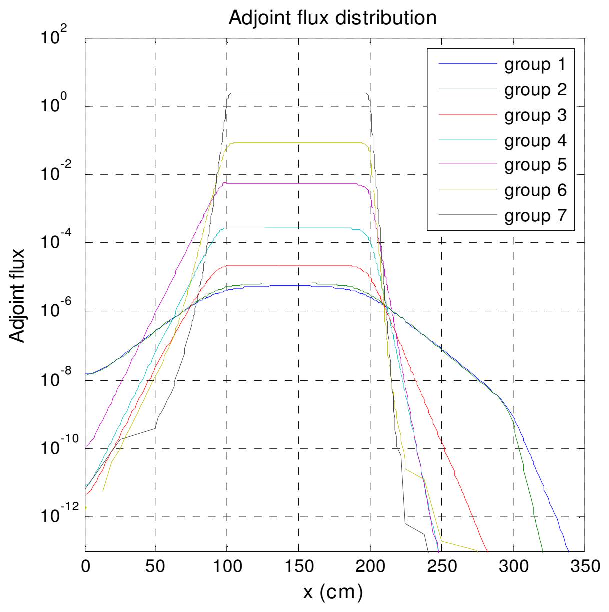
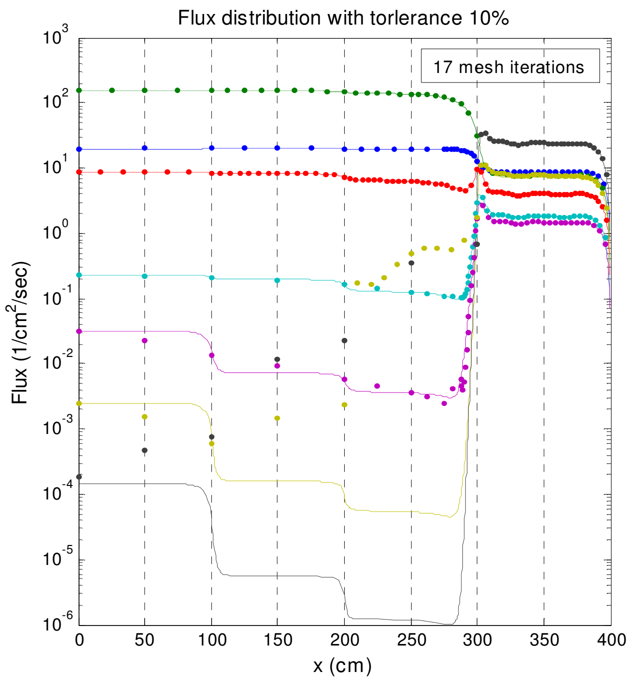
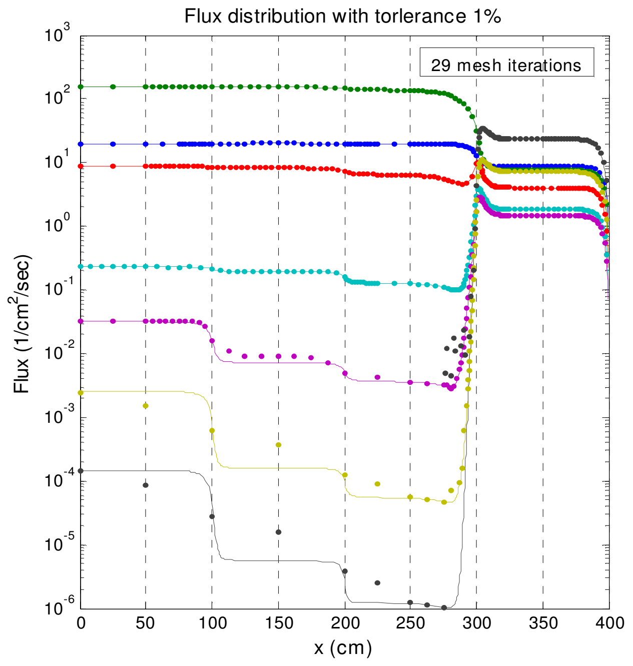


Fig. IV-50. “Exact” direct and adjoint flux distribution of example 6.A

Their number of degrees of freedom and accuracy are shown in TABLE IV-IV. And Flux distributions with different tolerances are shown in Fig. IV-51. Their number of degrees of freedom and accuracy are shown in TABLE IV-V and TABLE IV-VI.

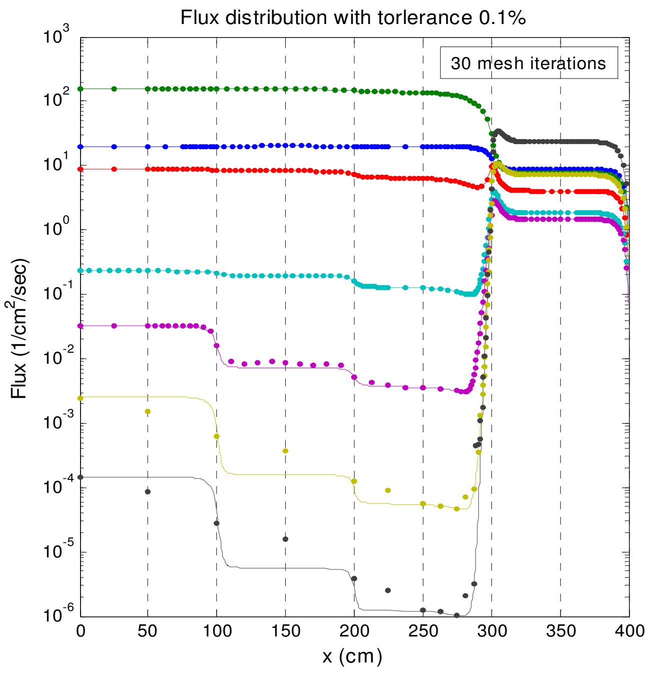


(a)

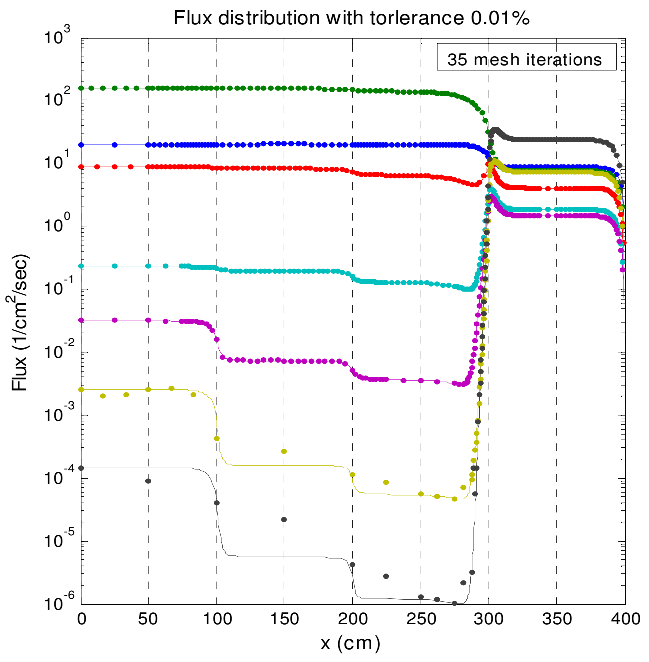


(b)

Fig. IV-51 Flux distribution with different tolerance of example 6.A: (a) *tol*=10%, (b) *tol*=1%



(c)



(d)

Fig. IV-51 (Continued), (c) *tol*=0.1% and (d) *tol*=0.01%

TABLE IV-IV

“Exact” solution of example 6.A

	direct			adjoint	
group	DoFs	accuracy		DoFs	accuracy
1	158	0.0000001		154	0.0000097
2	139	0.0000001		169	0.0000113
3	194	0.0000001		220	0.0000033
4	250	0.0000001		252	0.0000003
5	256	0.0000001		235	0.0000001
6	220	0.0000002		205	0.0000001
7	178	0.0000014		174	0.0000001

TABLE IV-V

Results of example 6.A with *tol*=10% and 1%

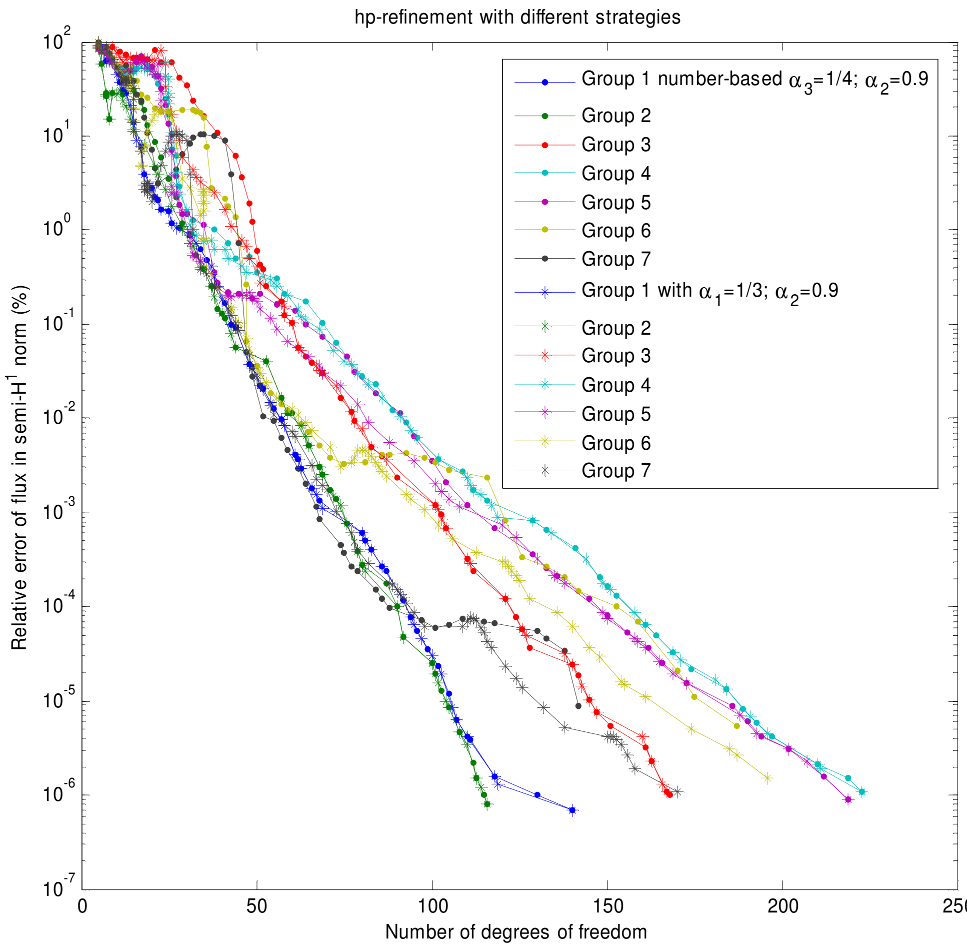
	tol=10% 17 mesh iterations			*tol*=1% 29 mesh iterations	
group	DoFs	accuracy		DoFs	Accuracy
1	23	1.6263666		48	0.0376078
2	23	5.8257019		53	0.0402451
3	39	10.9362557		62	0.0571557
4	26	10.5387235		64	0.1723995
5	25	13.7056138		51	0.2072578
6	25	18.1879218		47	0.0652564
7	20	7.0117013		45	0.7229557

TABLE IV-VI

Results of example 6.A with *tol*=0.1% and 0.01%

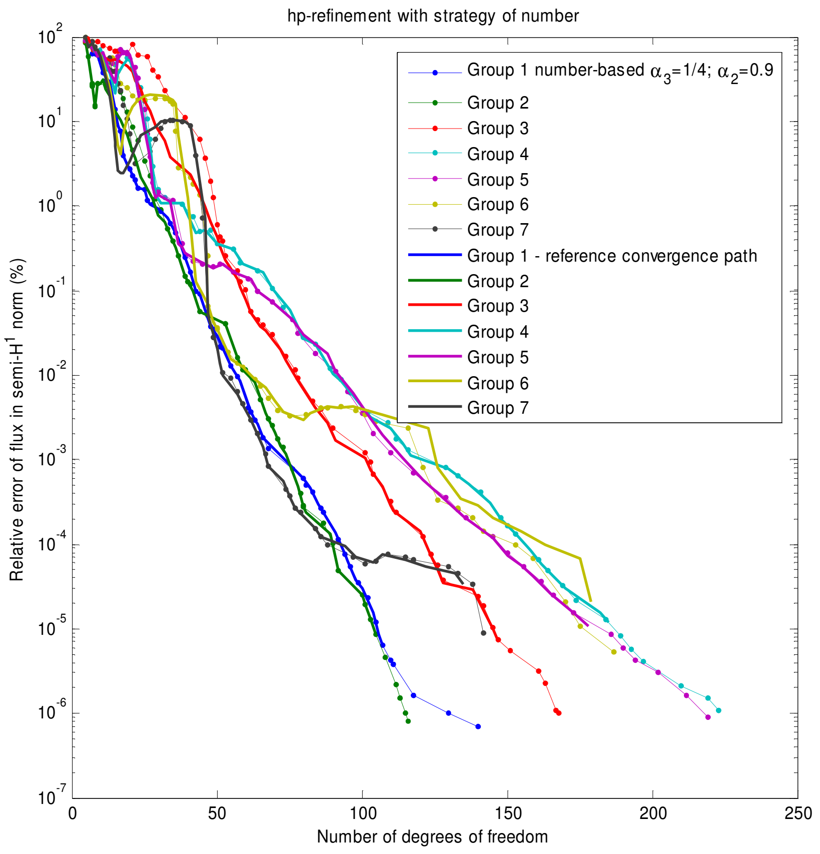
	tol=0.1% 30 mesh iterations			*tol*=0.01% 35 mesh iterations	
group	DoFs	accuracy		DoFs	Accuracy
1	49	0.0352224		61	0.0041080
2	53	0.0402415		65	0.0051888
3	64	0.0453939		78	0.0093476
4	69	0.1038216		93	0.0088737
5	56	0.1619451		78	0.0308484
6	50	0.0364227		63	0.0087919
7	47	0.0512048		57	0.0062672

Different refinement strategies were tried.



(a)

Fig. IV-52. Number-based convergence path of example 6.A: (a) compared with error-based calculation

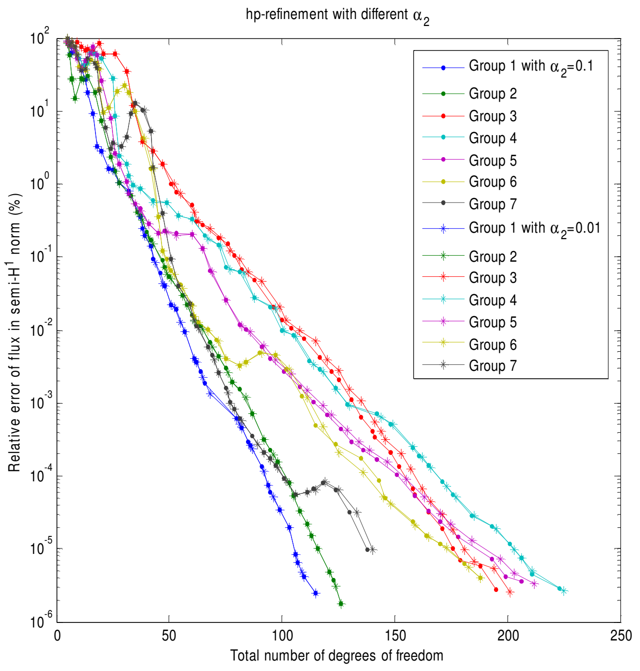


(b)

Fig. IV-52. (Continued), (b) compared with reference convergence path

Number-based refinement nearly delivers same convergence path as error-based refinement and the reference path as in Fig. IV-52. The number of mesh iteration needed to reach a certain tolerance is much smaller than error-based refinement with α_1=1/3. These figures also show that setting square of norm of group flux as the group weightings is appropriate.

Two error-based refinements with different α_1 are compared. Results show there is a broad range of choice of α_1. We can decrease α_1 to make mesh iteration converge faster. Different α_2 were also tried. Results are plotted in Fig. IV-53.



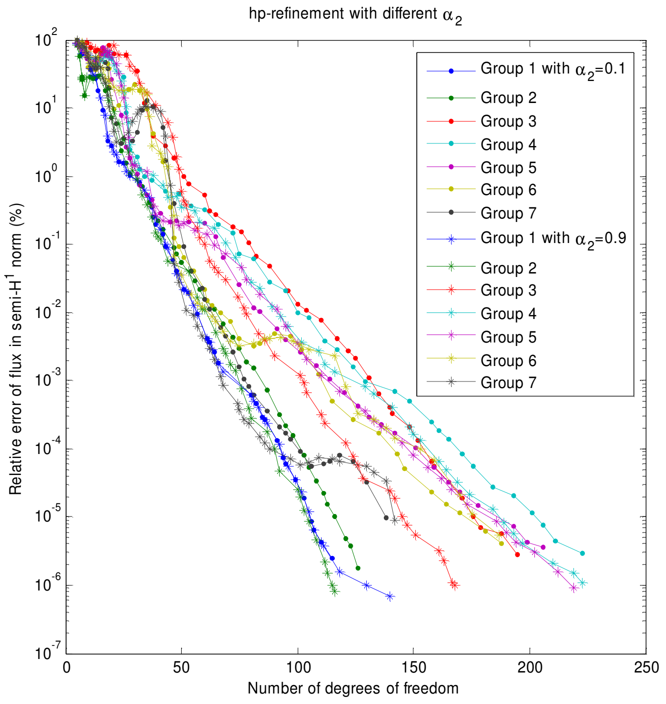
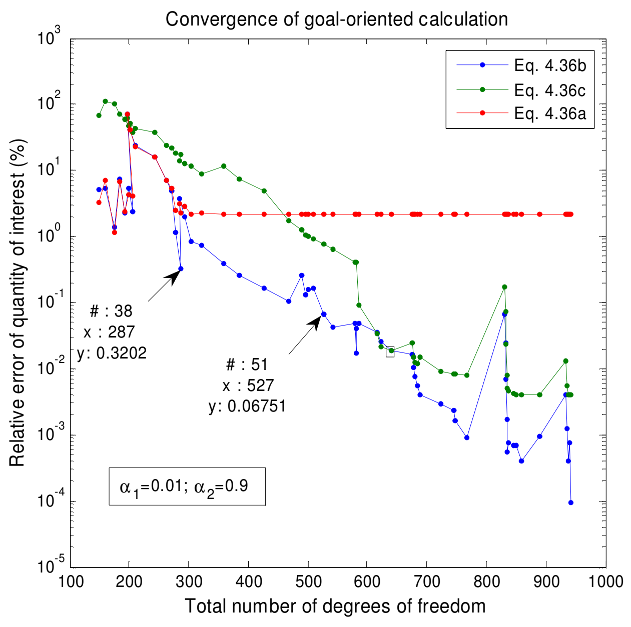


Fig. IV-53. Convergence paths with different *h*-refinement constrain of example 6.A

It shows again that higher values of α_2 are preferred for neutron diffusion problems.

Then goal-oriented calculations with different refinement strategies were tried. The quantity of interest is the integrated flux in the second assembly. Convergence paths are shown in Fig. IV-54.



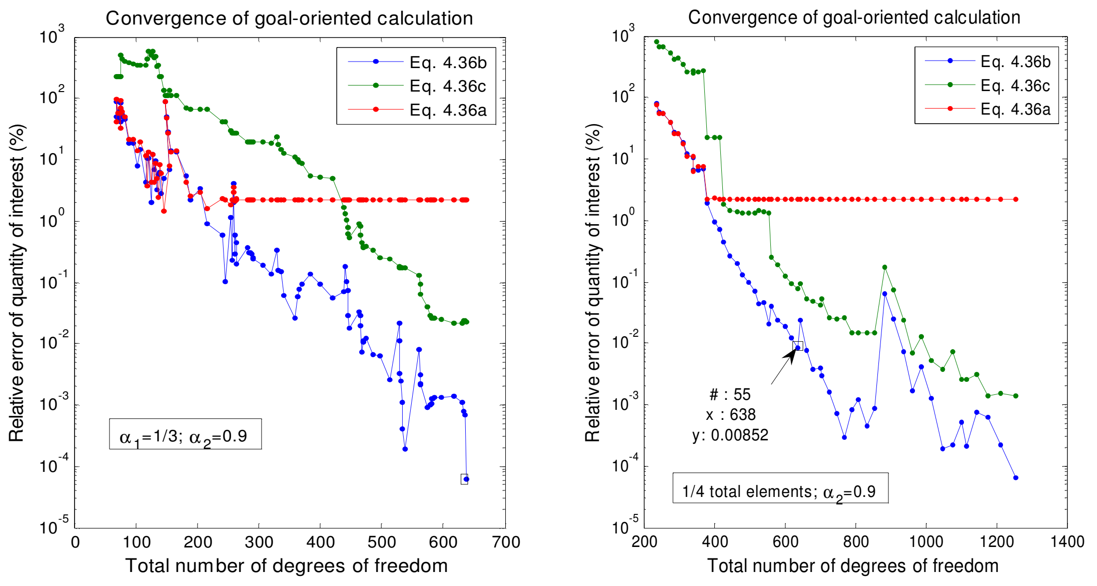
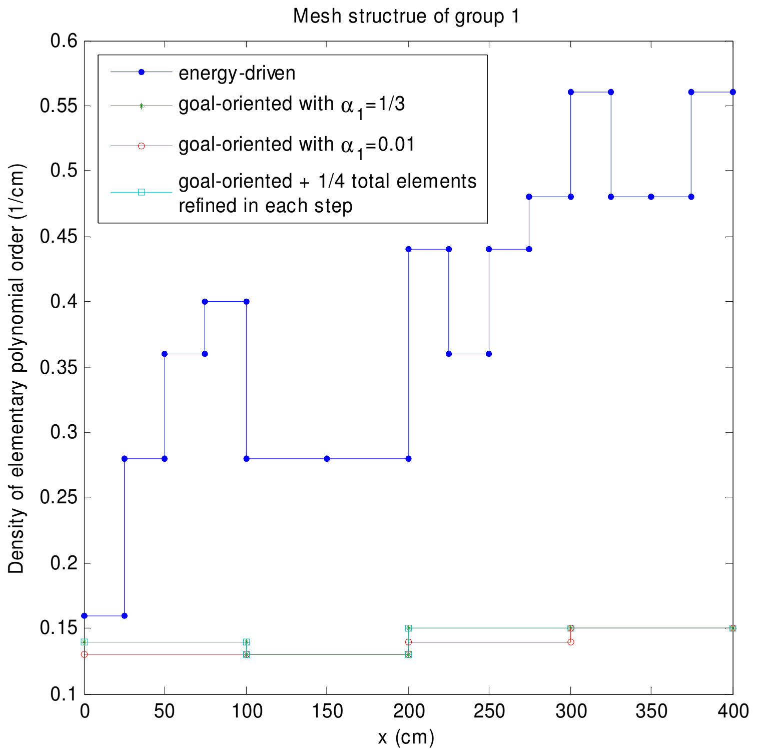


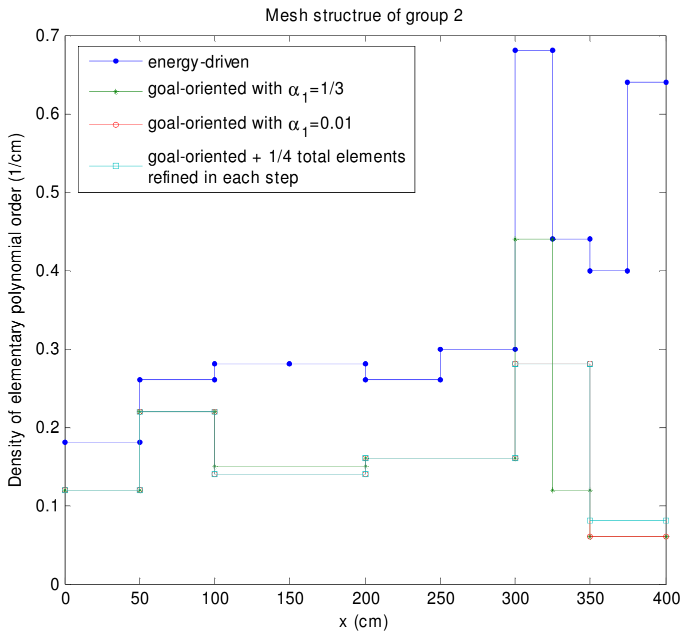
Fig. IV-54. Convergence paths of goal-oriented calculations of example 6.A

We observe that the energy-driven calculation using 1395 total DoFs can only give produce the quantity of interest with the relative error being about 2%. With goal-oriented calculation, 527 DoFs have led to an error below 0.1% with α_1=0.01 and α_2=0.9. This time, different parameters α_1 give different convergence slopes, which could be due to unnecessary refinement at boundary assembly. Mesh structure of "exact" energy-driven calculation and three goal-oriented calculations (marked in Fig. IV-54) for all 7 groups are plotted in Fig. IV-55.

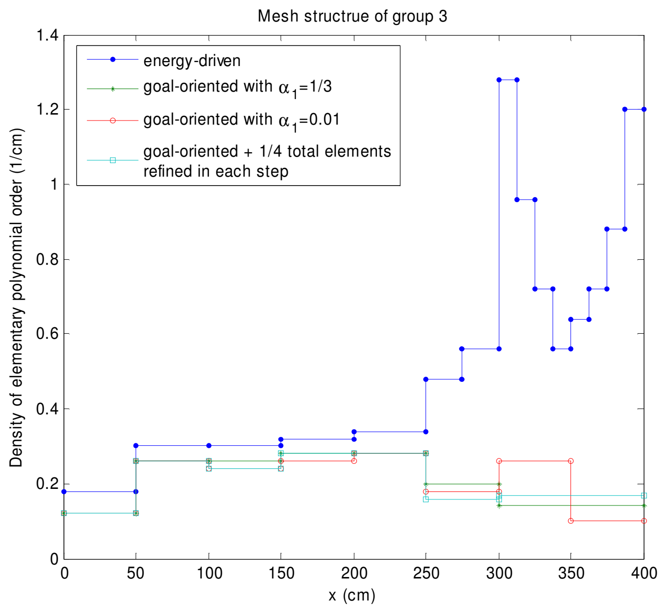


(a)

Fig. IV-55. Mesh structure of example 6.A: (a) group #1

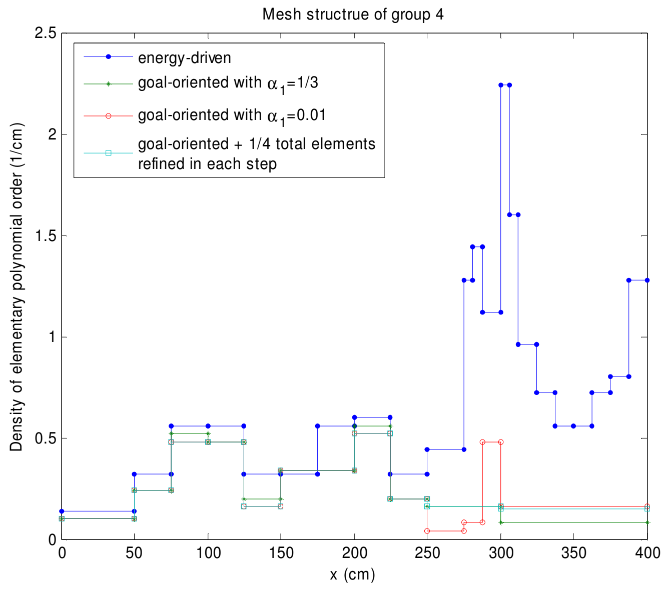


(b

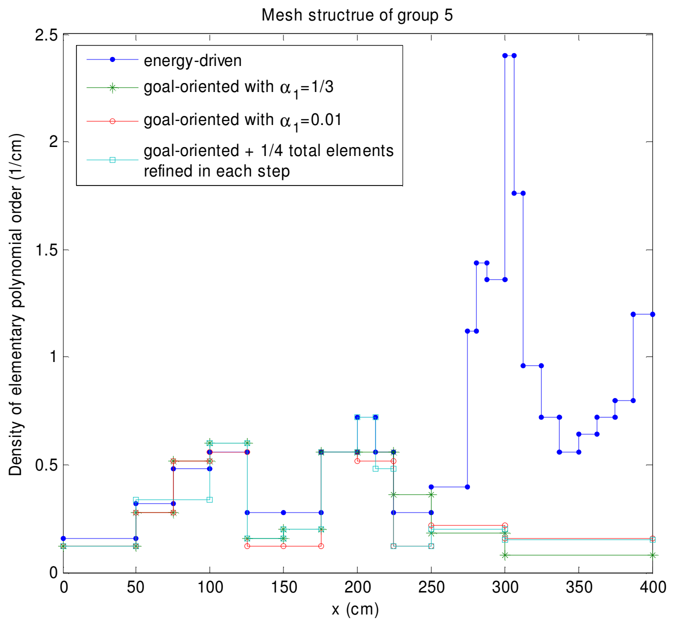


(c)

Fig. IV-55. (Continued), (b) group #2, (c) group #3

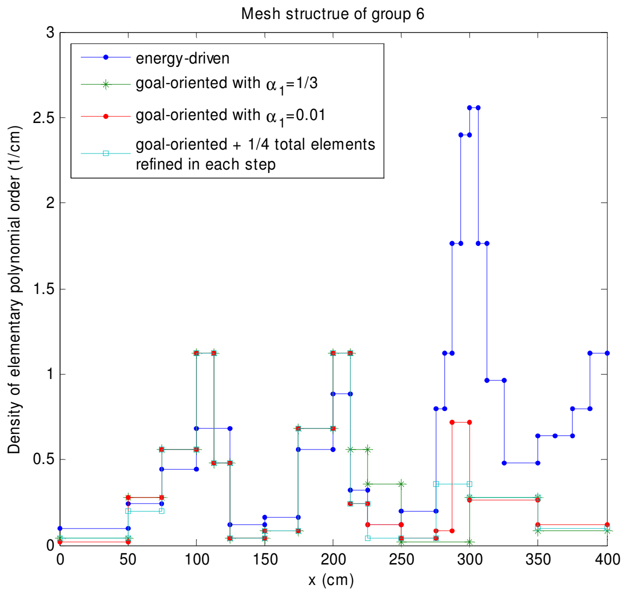


(d) group #4

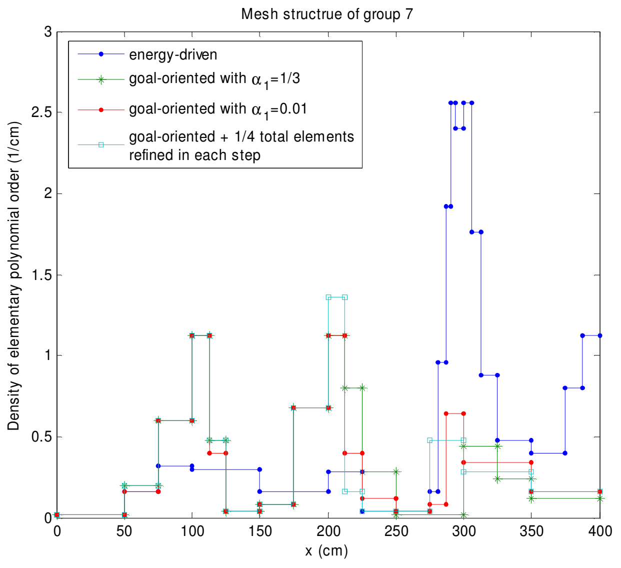


(e) group #5

Fig. IV-55. (Continued), (d) group #4, (e) group #5



(f) group #6



(g) group #7

Fig. IV-55. (Continued), (f) group #6, (g) group #7

We note that when comparing goal-oriented calculations, meshes in fast groups and right boundary assembly are much refined. Because there is a large flux gradient in group 7 at right boundary assembly, this assembly also receives a lot of mesh refinements in goal-oriented calculations which shows for some cases, we may need improve the Eq. (4.32).

4.5.7.2 Example 6.B (7-group eigenvalue problem)

Problem description:

There are three fuel assemblies and one reflector assembly each with 20cm width in this problem. Fuel assembly are arranged as follows: 1-7-1-7-1-7-1-7-1-7-1-7-1-7-1-7-1-7-1-7 and 2-7-2-7-2-7-2-7-2-7-2-7-2-7-2-7-2-7-2-7 and 3-7-3-7-3-7-3-7-3-7-3-7-3-7-3-7-3-7-3-7 lattices. The size of fuel pins is 1cm and the size of the moderator gap is also 1cm. Left homogeneous Neumann boundary condition and zero flux right boundary.

Conditions:

1 linear element of each fuel pin and moderator gap and 1 linear element of the reflector assembly; Lobatto shape functions; p_{max}=17; α_2=0.9;

Results:

Convergence paths with different refinement strategies are plotted in Fig. IV-56.

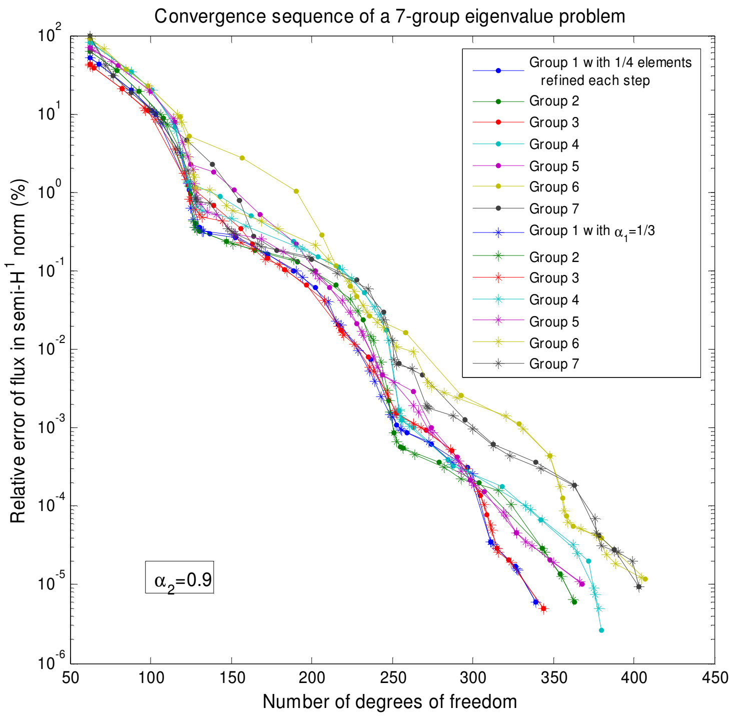


Fig. IV-56. Convergence paths of 7-group eigenvalue problem

Again we obtain exponential convergence and all groups converge well with the flux weightings.

Flux distributions with different tolerances are plotted in Fig. IV-57. “Exact” fluxes are plotted in solid lines.

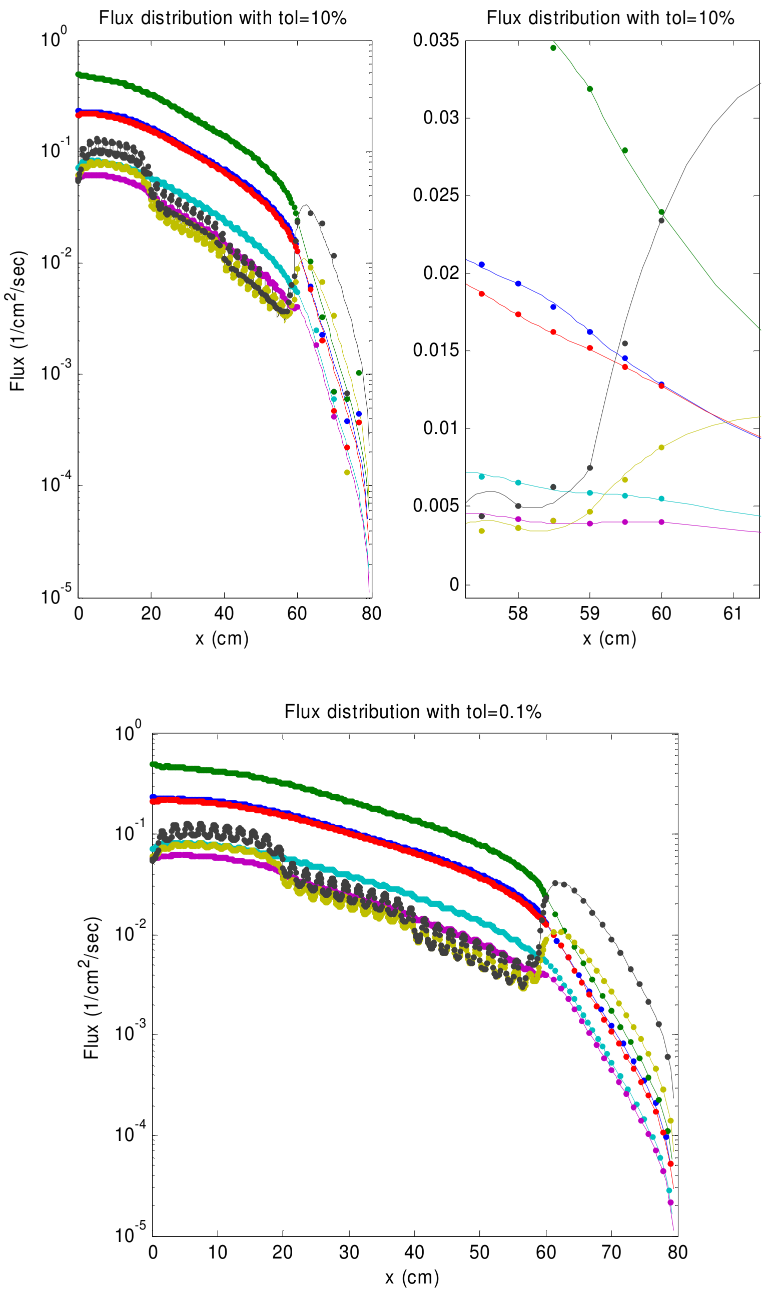


Fig. IV-57. Flux distribution of example 6.B

Due to the small size of fuel pin, even with 10% tolerance we obtain very accurate flux in fuel assemblies. Because linear elements are used, some points in the zoom of Fig. IV-57 depart from the exact value which is at middle of the elements. The adjoint flux is given in Fig. IV-58.

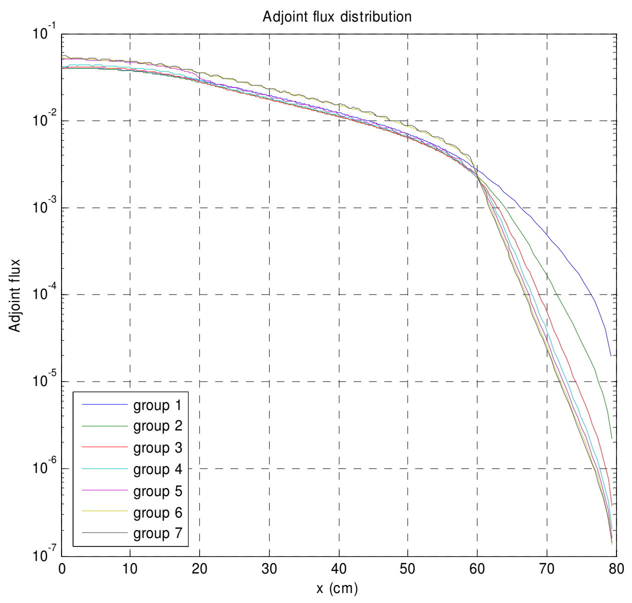


Fig. IV-58. Adjoint flux distribution of example 6.B

Both direct and adjoint "exact" fluxes have tolerance lower than 10^{-5}%. The number of degrees of freedom and accuracies for all groups are listed in TABLE IV-VII.

TABLE IV-VII

"Exact" solution of example 6.B

group	direct		adjoint	
	DoFs	$E_i \times 10^6$	DoFs	$E_i \times 10^6$
1	339	6.0	258	9.0
2	362	6.5	314	5.2
3	344	5.0	325	8.7
4	378	4.9	383	10.1
5	366	10.9	419	7.2
6	405	12.3	437	9.6
7	403	9.3	442	9.5

The mesh structure of the direct calculation is shown in Fig. IV-59.

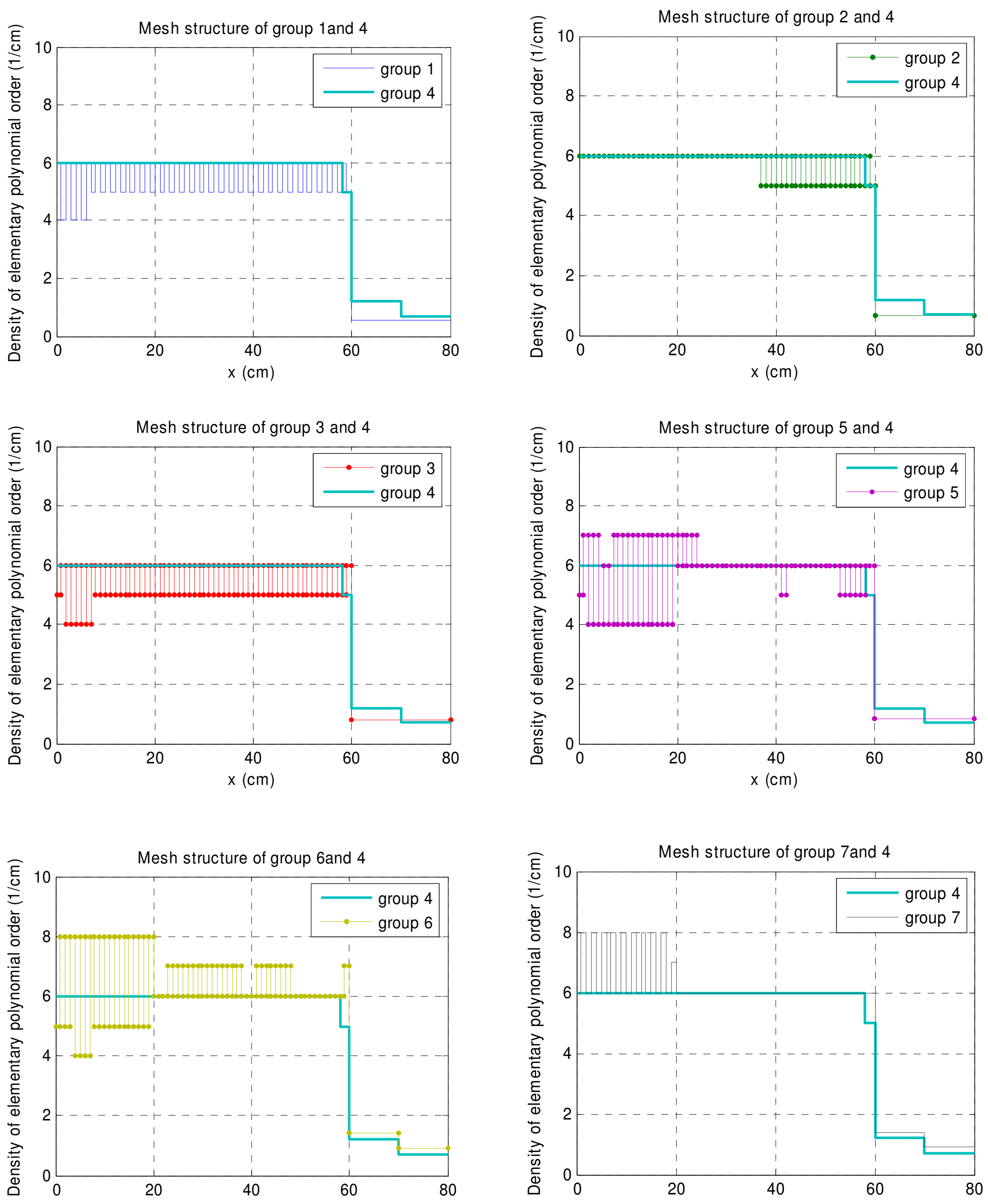


Fig. IV-59.   Direct mesh structure of example 6.B

The mesh structure of the adjoint problem is shown in Fig. IV-60. The thermal mesh densities tend to be higher.

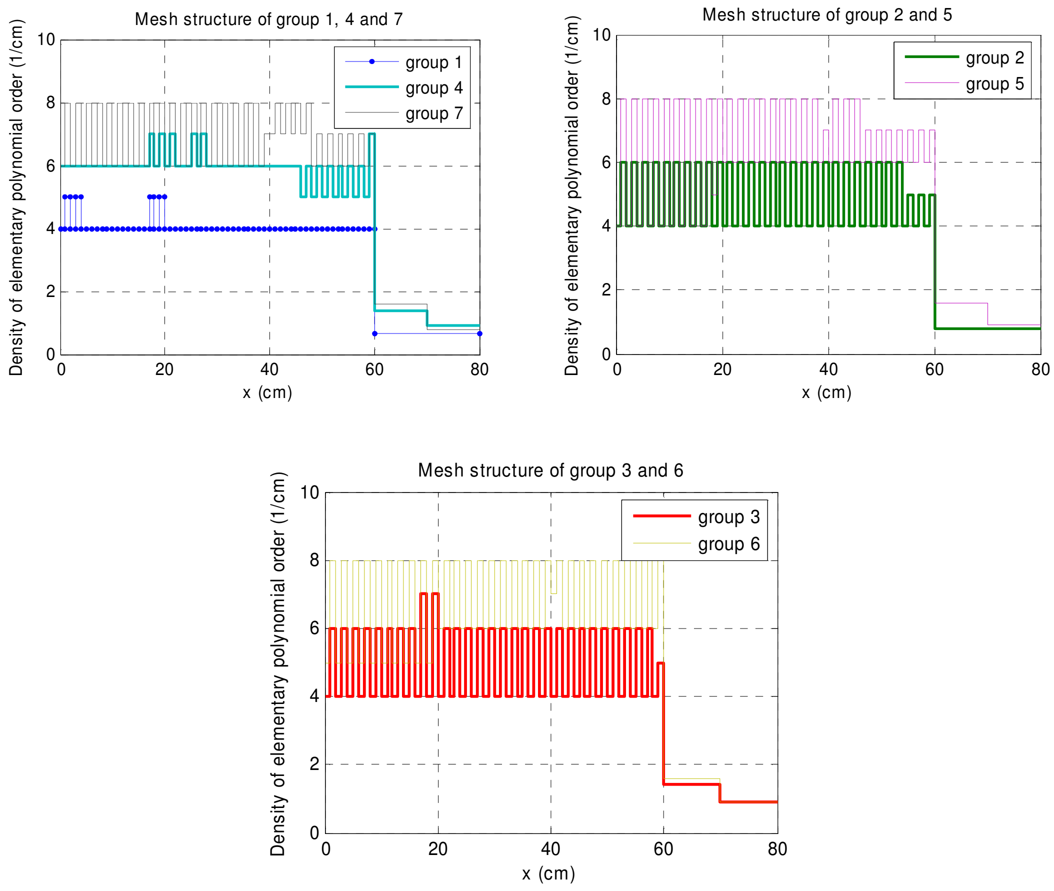


Fig. IV-60. Adjoint mesh structure of example 6.B

Using the sorting -based refinement, with 10 mesh iterations, accuracies below 0.1% and 21 iterations 10^{-5}% can be reached. For adjoint calculation, numbers are 8 and 21 respectively.

Then, goal-oriented calculations with different refinement strategies were performed. Results are shown in Fig. IV-61. Convergences of flux are also compared with results of energy-driven calculation in Fig. IV-62.

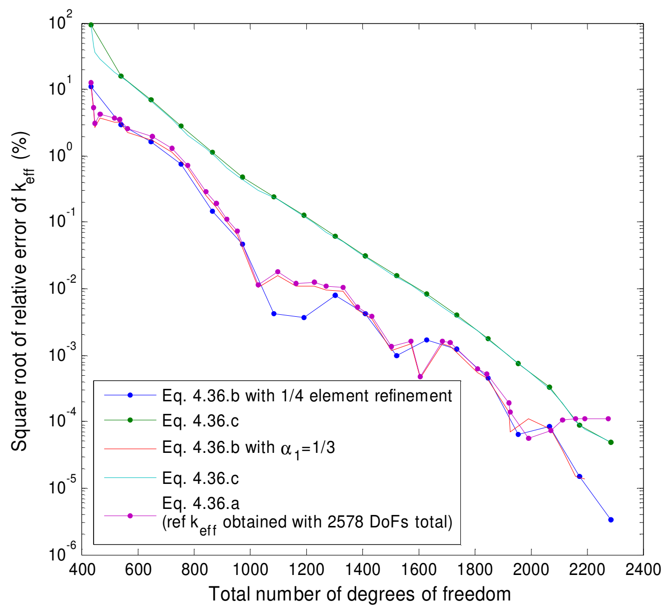


Fig. IV-61. Convergence paths of goal-oriented calculation of example 6.B

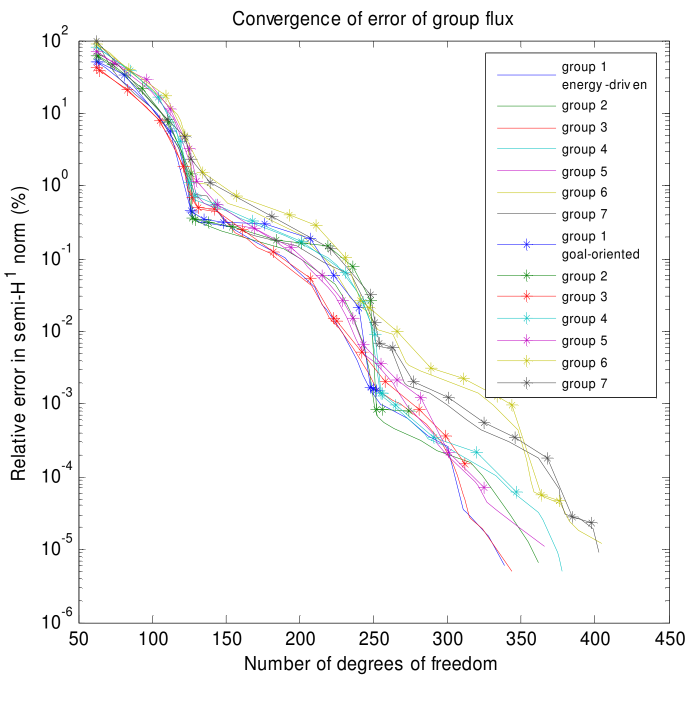


Fig. IV-62. Convergence paths of energy-driven and goal-oriented calculations of example 6.B

We obtain exponential convergence. For this problem, there is no much difference between energy-driven and goal-oriented calculation. Convergence path and mesh structures verified this.

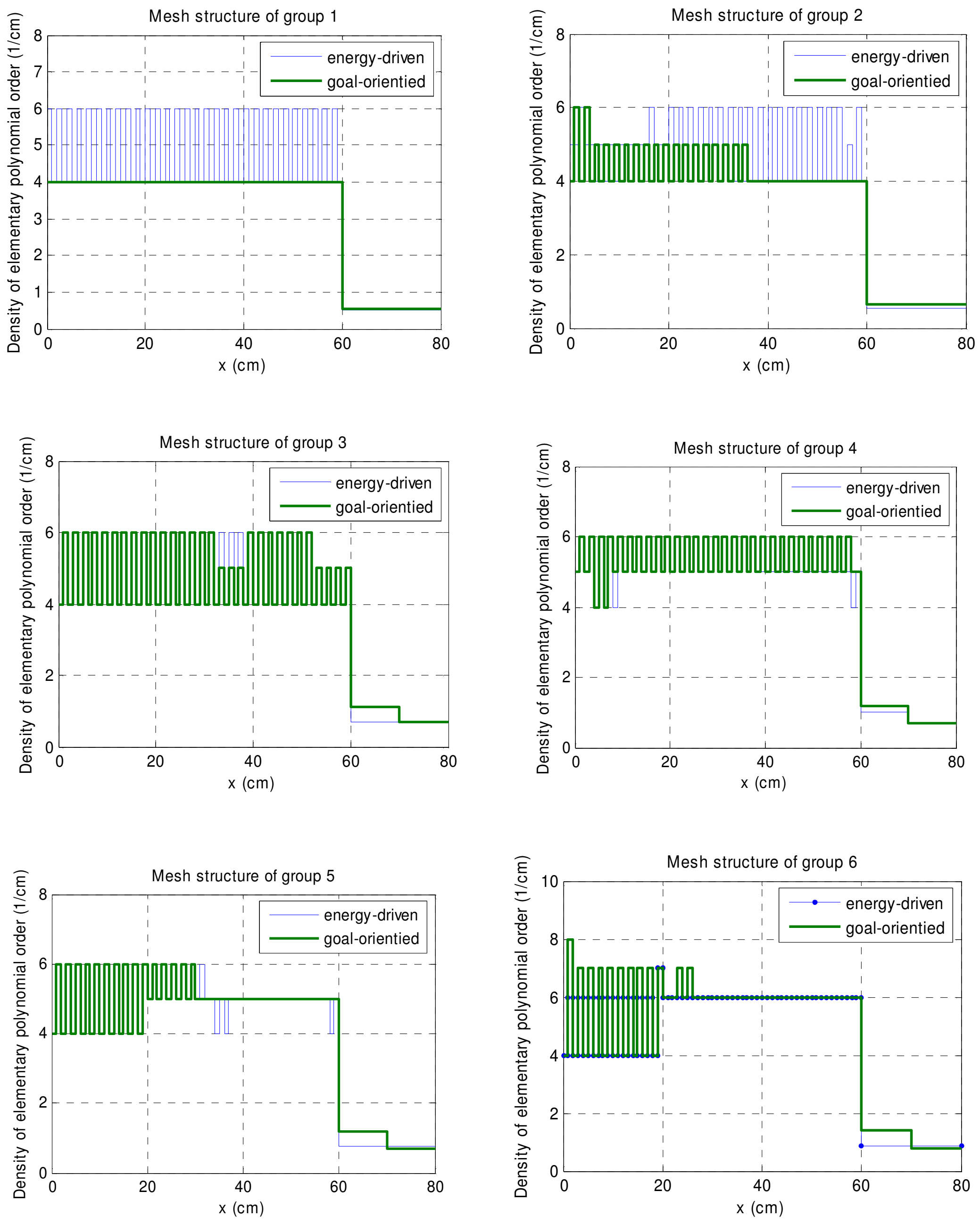


Fig. IV-63. Mesh structure of example 6.B

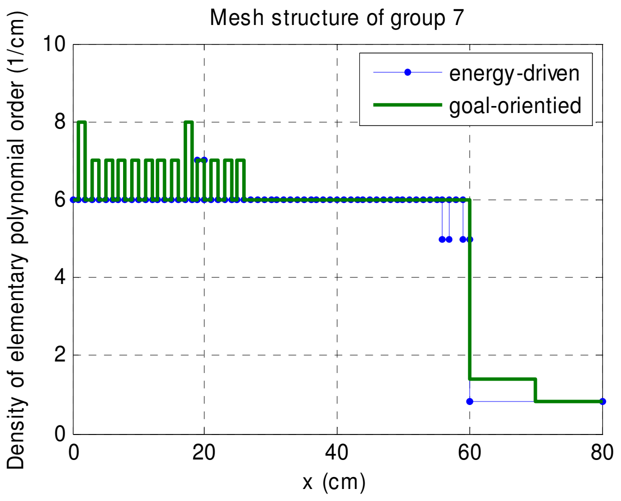


Fig. IV-63. (Continued) Mesh structure of example 6.B

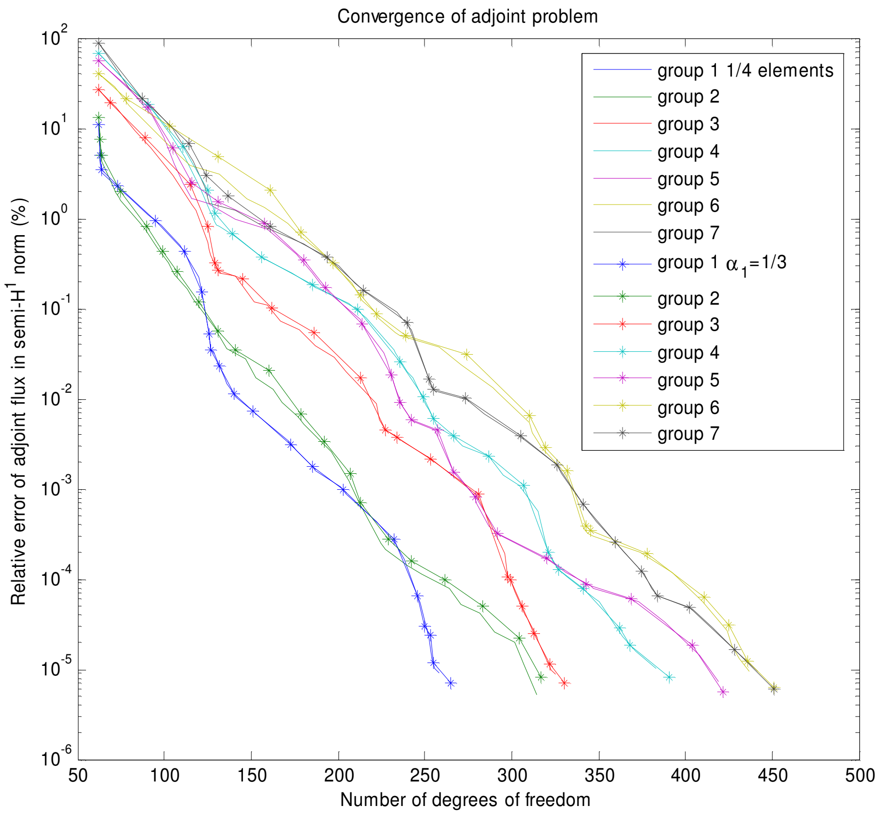


Fig. IV-64. Energy-driven calculation of adjoint eigenvalue problem of example 6.B

Mesh structures in Fig. IV-63 are those obtained at the 38th mesh iteration with α_1=1/3 and sorting-based goal-oriented calculation with relative error less than 10^{-5}%. The total numbers of degrees of freedom for these two meshes are nearly identical. The difference in k_{eff} between these two calculations is too small to be noticeable, although goal-oriented calculation refined more in thermal groups. Convergence path of adjoint calculation alone are plotted in Fig. IV-64.

4.5.8 Point-wise values as quantity of interest

4.5.8.1 Example 7 (A two group source problem with analytical solution)

Problem description:

Use example 2.A and set adjoint source for thermal group at some points and zero elsewhere. Adjoint source of fast group is zero. Left boundary condition of adjoint problem is homogeneous Neumann and right boundary condition is zero flux.

Calculation conditions:

Since we can expect singularities at the source points, we relax the *h*-refinement constraint α_2 to 0.01; Error-based strategy with α_2=1/3 is applied; 1 linear initial element from 0 to 80cm; mesh iteration outside implementation is used; Goal-oriented case.

Results:

TABLE IV-VIII lists the results of the adjoint source Eq. (4.34) at 75cm with g=2. We can see that using semi-H^1 norm to calculate the interpolation error failed. The more the mesh is refined, the larger the norm. *h*-refinement wins almost all the time close to the singular point. And E_2 converges to a non-zero value with mesh iteration.

TABLE IV-VIII

Mesh iteration with semi-H^1 norm of point current as quantity of interest

group	Mesh iteration #	DoFs	Solution semi-H^1 norm	E_2
2	1	3	1.75066E-04	98.89066433
2	2	4	1.96640E-04	92.79639703
2	3	5	1.52025E-02	100.40055667
2	4	6	3.28123E-02	100.16706496
2	5	7	1.17809E-01	94.00931831
2	6	8	1.16404E+00	98.29015646
2	7	9	1.61651E+00	91.04143018
2	8	10	3.22559E+00	88.18234032
2	9	11	6.43047E+00	86.53503597
2	10	12	1.28329E+01	85.66809918
2	11	13	2.56342E+01	85.22609952
2	12	14	5.12348E+01	85.00321545
2	13	15	1.02435E+02	84.89132027
2	14	16	2.04835E+02	84.83526011
2	15	17	4.09635E+02	84.80720186
2	16	18	8.19235E+02	84.79316568
2	17	19	1.63844E+03	84.78614583
2	18	20	3.27684E+03	84.78263545
2	19	21	6.55364E+03	84.78088016
2	20	22	1.31072E+04	84.78000248

If we use L_2 norm to evaluate elementary error and to make decision where to refine the mesh and how to refine and when to terminate mesh iteration, mesh iteration can converge properly. (Refer to Chapter III section 3.5.3 for details.)

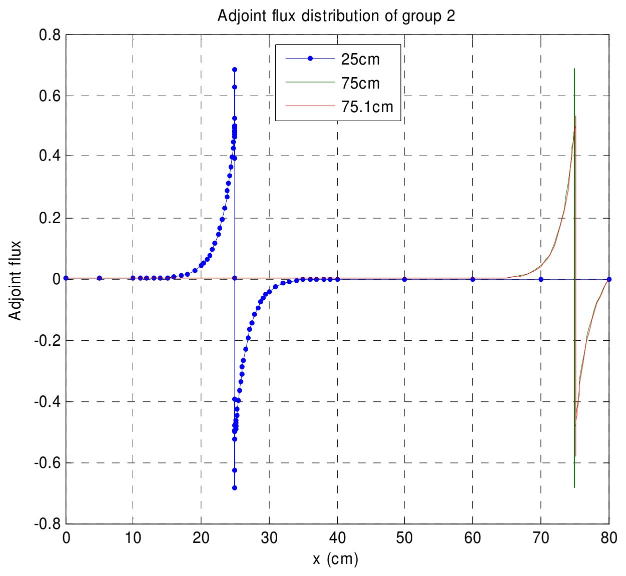
TABLE IV-IX is from an excerpt of the calculation at 75cm for group 2. Group 1 can always converged properly.

TABLE IV-IX

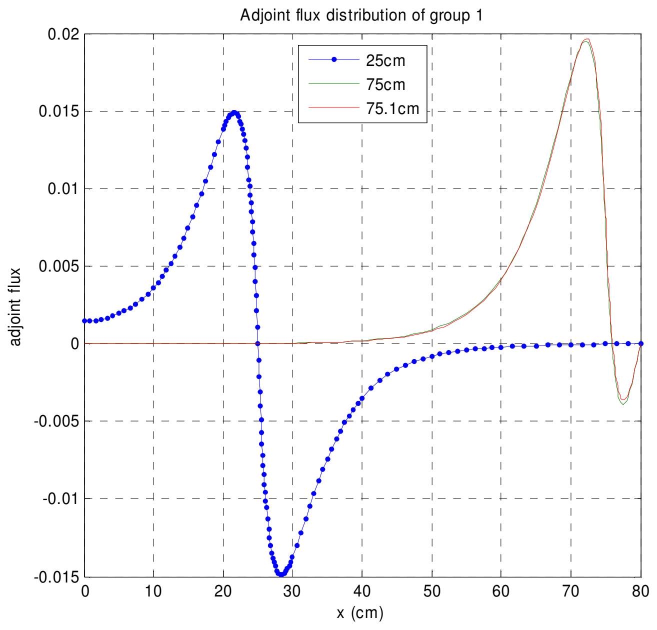
Mesh iteration with L_2 norm of point current as quantity of interest

group	Mesh iteration #	DoFs	Solution L_2 norm	E_2
2	1	3	1.12956E-01	98.85325535
2	2	4	1.26769E-01	92.69027294
2	3	5	2.43971E+00	100.37839586
2	4	6	5.26074E+00	100.15695430
2	5	7	4.73424E+00	93.94967826
2	6	8	1.17005E+01	98.22940082
2	7	9	4.09311E+00	90.85905116
2	8	10	2.47007E+00	89.87780699
2	9	12	2.75965E+00	91.67243942
2	10	14	4.03636E+00	88.73564264
2	11	15	2.42976E+00	86.06222298
2	12	16	1.52488E+00	84.31184602
2	13	18	8.05608E-01	82.10265883
2	14	20	4.38561E-01	78.79493742
2	15	22	2.53050E-01	73.54484235
2	16	24	1.59765E-01	65.79315801
2	17	26	1.12986E-01	55.89657473
2	18	28	8.95607E-02	45.29317363
2	19	30	7.78392E-02	35.67903344
2	20	32	7.19763E-02	28.08379814
2	21	34	6.90443E-02	22.70934126
2	22	37	6.75908E-02	16.71674314
2	23	40	6.68572E-02	12.02363684
2	24	43	6.64914E-02	8.90076985
2	25	48	6.63083E-02	5.14535707
2	26	50	6.62166E-02	4.06508374
2	27	52	6.61708E-02	3.39700960
2	28	57	6.61479E-02	2.11590762
2	29	61	6.61364E-02	1.27975940
2	30	63	6.61307E-02	1.00646054
2	31	65	6.61278E-02	0.83696570
2	32	69	6.61264E-02	0.66831253
2	33	76	6.61257E-02	0.35545215
2	34	78	6.61253E-02	0.29543788
2	35	82	6.61246E-02	0.19121665
2	36	85	6.61245E-02	0.14047806
2	37	88	6.61244E-02	0.10978462
2	38	94	6.61244E-02	0.06252990
2	39	97	6.61244E-02	0.04552543
2	40	99	6.61244E-02	0.03823861

The adjoint flux is plotted in Fig. IV-65.



(a) thermal group



(b) fast group

Fig. IV-65. Adjoint flux of example 7: (a) thermal group, (b) fast group

Convergence paths of energy-driven calculation with quantities of interest located at points at 25cm or 75cm or 75.1cm are plotted on Fig.IV-66.

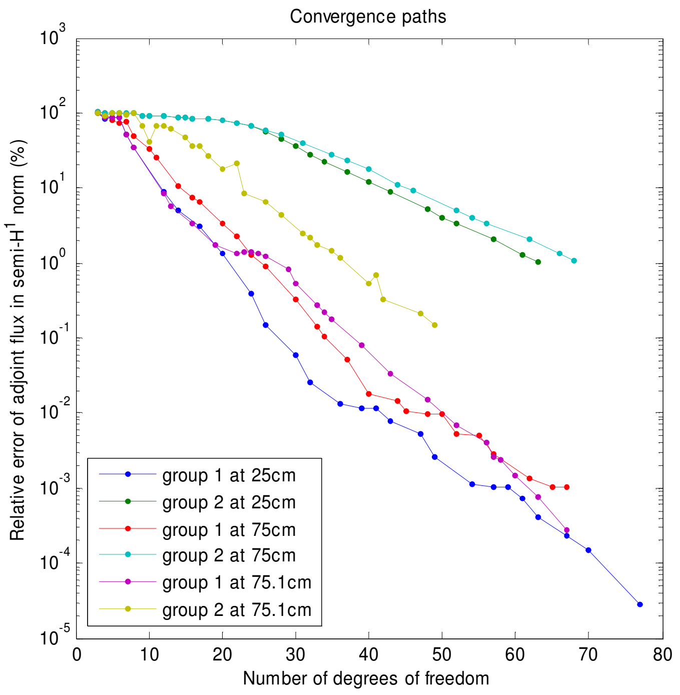
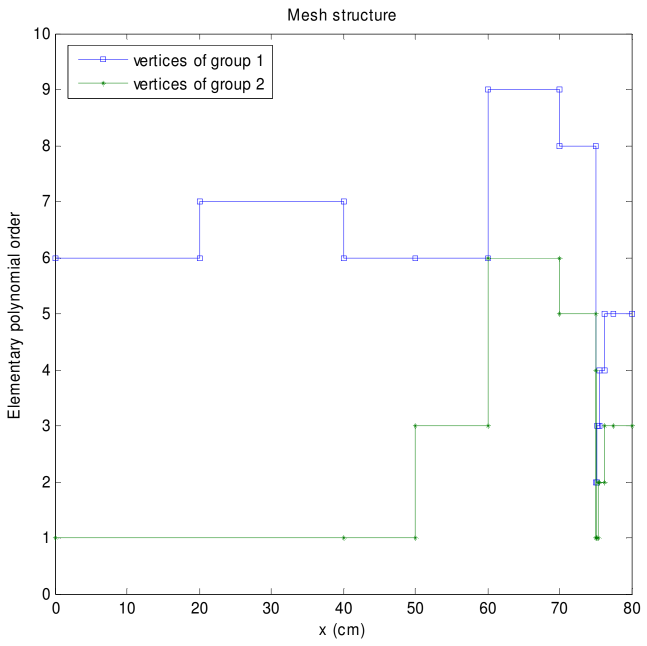


Fig. IV-66. Convergence paths of energy-driven calculation of example 7

hp-refinement can still deliver exponential convergence. If the point is at the vertices of element, we need *h*-refinement on both elements. That is why the slope of convergence path at 75.1 cm is about double of the slopes of the other two cases.

Mesh structures after 30 mesh iterations for two groups at 75.1cm (E_1=0.15% and E_2=2.7×10^{-4}%) are plotted in Fig. IV-67.



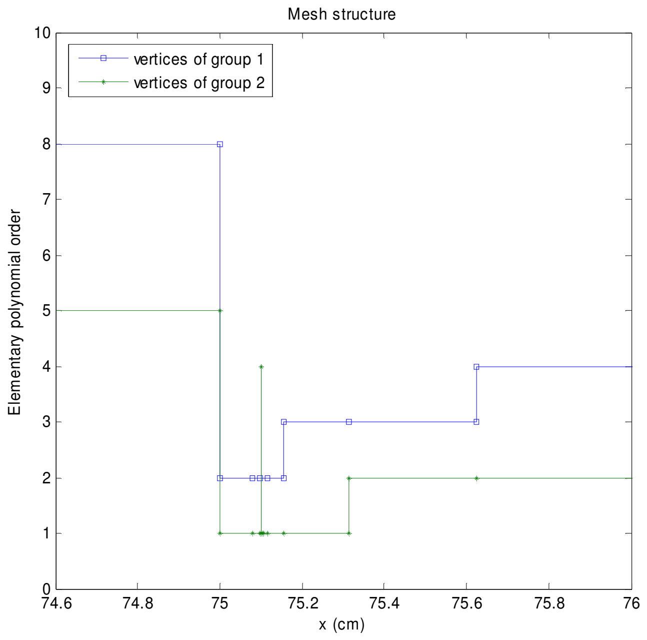


Fig. IV-67. Mesh structure of example 7

We note that at 75.1cm, the mesh is much refined, especially for the second group.

Then goal-oriented calculations are performed. The calculation converged properly, even when using the semi-H^1 norm. We obtain again exponential convergence and correct current values. Convergence paths are plotted in Fig. IV-68.

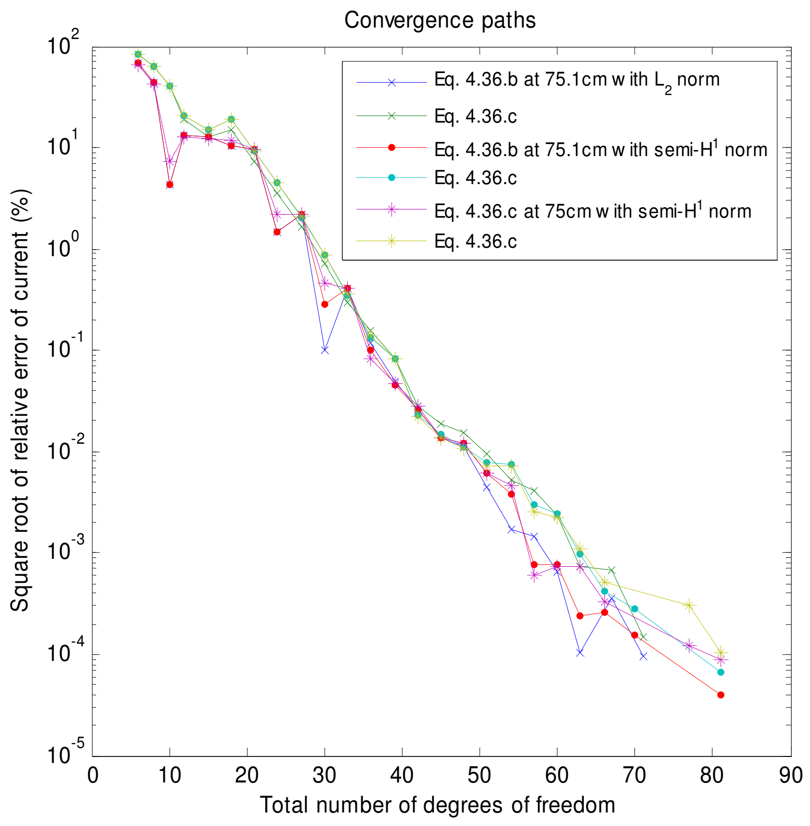


Fig. IV-68. Convergence paths of goal-oriented calculation of example 7

For example, after 24 mesh iterations, the square root of relative error drops below 0.0001% with the quantity of interest being 0.300506950285 at 75cm while the exact value is 0.30050695028506. Same for the point at 75.1cm with current being 0.304651612234 vs. the exact value 0.30465161223356.

Converged meshes for 75cm and 75.1cm of goal-oriented calculation are plotted in Fig. IV-69.

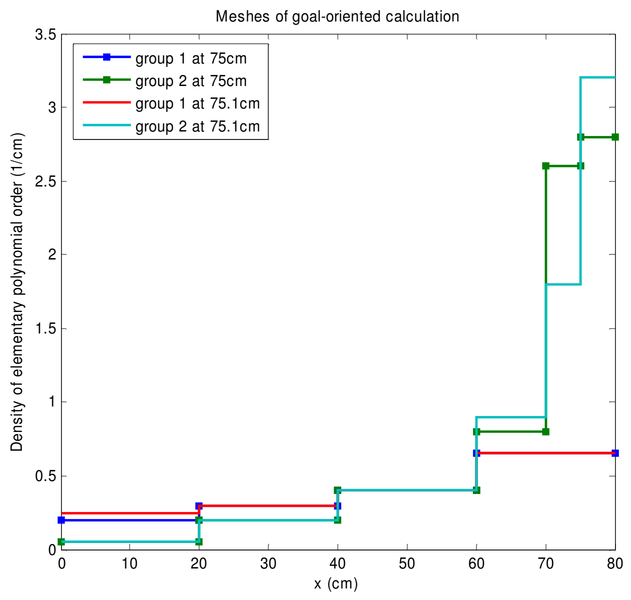


Fig. IV-69. Mesh structure of goal-oriented calculation of example 7

Then calculations with flux being the quantity of interest are also performed. Convergence path of goal-oriented calculation at 75cm is in Fig. IV-70.

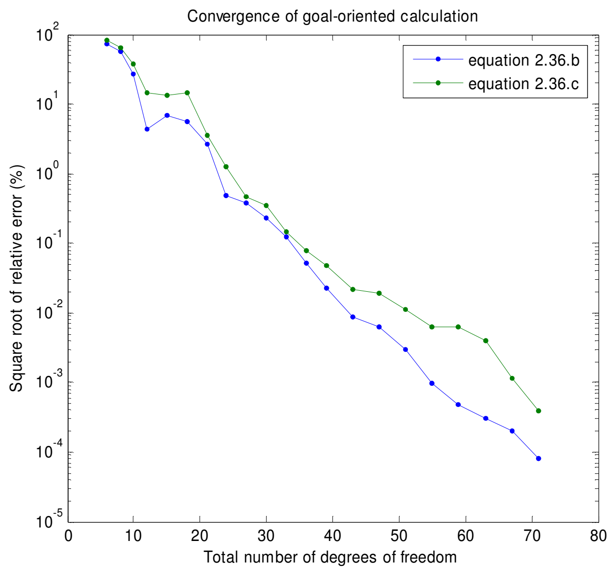
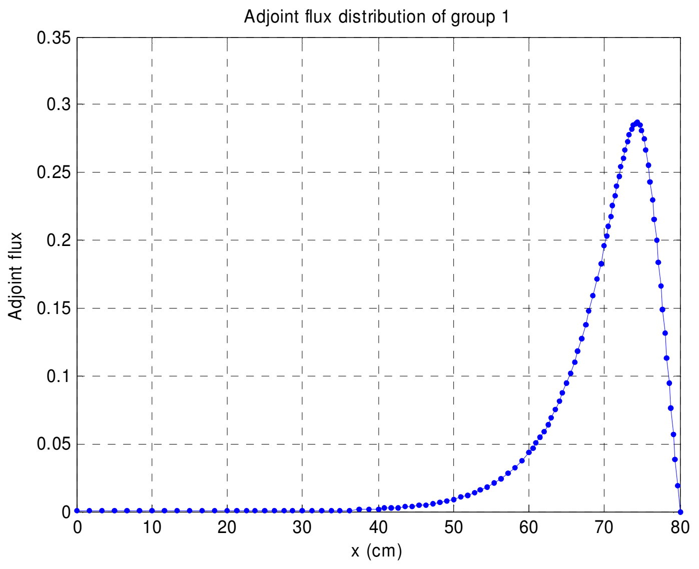
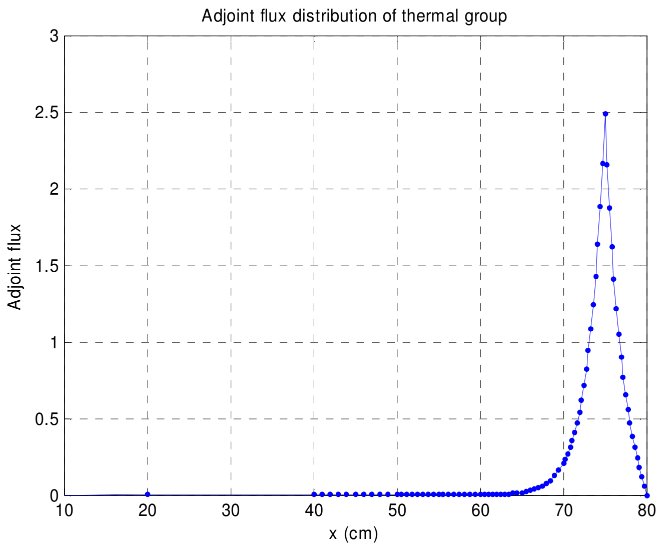


Fig. IV-70. Convergence path of goal-oriented calculation of point flux as quantity of interest

The adjoint flux is in Fig. IV-71.



(a) fast group



(b) thermal group

Fig. IV-71. Adjoint flux of point flux as quantity of interest

Mesh structure of goal-oriented calculation with *tol*=10^{-5}% is in Fig. IV-72.

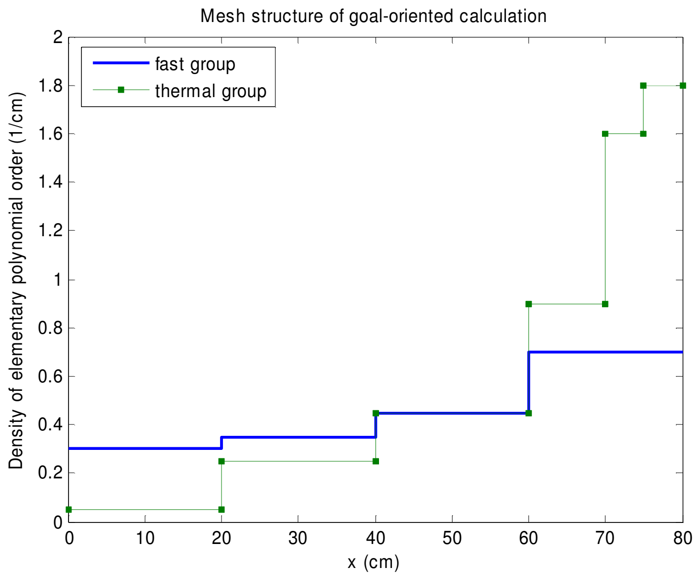


Fig. IV-72. Mesh structure of point flux as quantity of interest

4.6 Conclusions of chapter IV

The following conclusions can be drawn:

1. Both energy-driven and goal-oriented calculations for multigroup source problems and eigenvalue problems successfully delivered exponential convergence. Significantly smaller number of degrees of freedom is needed to reach a prescribed tolerance.
2. The convergence process is stable, and no special initial mesh generation is required.
3. The relationship between mesh iteration and power/flux iteration was considered in two ways: mesh iteration wrapped around the power iteration loop and mesh iteration wrapped around the one-group solver. Both implementations proved reliable and produced quasi-identical convergence sequences.
4. Different meshes for different groups are needed.
5. in order to converge all groups together in energy-driven calculation with the mesh iteration being wrapped around the power/thermal iterations, the square of norm of flux

is a good choice for the group weighting factors.

6. In neutron diffusion problem, higher *h*-refinement constraints are needed to higher efficiency.

CHAPTER V

SUMMARY

We have presented and implemented a fully automatic *hp*-refinement strategy in the framework of the 1-D the multigroup diffusion source problems and eigenproblems. The method guarantees convergence in the numerical solution with the smallest number of unknowns. The *hp*-strategy is a technique combining *h*-refinements (subdivision of mesh cell or element) and *p*-refinements (increase in the element polynomial order) in a competitive fashion, yielding a solution converged to the user-prescribed tolerance. The mesh adaptation is automatic and is based on an interaction between two meshes, a coarse *hp* mesh and a fine *hp* mesh obtained from refinement of the coarse *hp* mesh. Having solved the problem on the fine mesh, a new optimal coarse mesh is constructed by minimizing the coarse grid interpolation error. Theoretical considerations and numerical experiments proved that the *hp* mesh adaptation strategy delivers exponential convergence rate in the multigroup diffusion setting. Such a convergence rate is significantly higher than the algebraic convergence obtained with *h*-uniform and *h*-adaptive refinements. The *hp*-adaptation converges independently of the choice of initial mesh, which is an improvement over *p*-strategies.

We extensively studied the interaction of the mesh adaptation procedure with the multigroup solver and the eigenvalue solver. Two implementations were tested: the first one consisted in wrapping the mesh iterations around the one-group fixed source solver. The second implementation embedded the entire multigroup eigensolver within one mesh adaptation step. Both implementations yield similar optimal solutions and meshes.

Optimality for multigroup equations was attained by solving each multigroup flux on its own mesh, leading group-dependent meshes. This requires the effective treatment of coupling terms due the energy transfers between group via scattering and fission. Notably, mass matrices containing polynomial functions of various orders and defined on different meshes needed to be integrated adequately. Adaptive integration proved to solve this issue efficiently.

Another key element of our work includes the development of goal-oriented mesh adaptation applied to multigroup eigensolvers. The aim of this type of mesh adaptation was to bring together the advantages *hp*-adaptivity and goal-oriented adaptivity into a fully automatic goal-oriented *hp*-adaptive strategy for multigroup eigenproblems. The gist of goal-oriented computations is to provide accurate estimates of functional of the solution (e.g., g detector response, integrated reaction rates, point wise flux, and point wise current). Goal-oriented adaptation introduces the adjoint problem and utilized it in the mesh optimization algorithm.

The methods employ a few parameters which may require some tuning for certain class of problems in order to yield the optimal mesh in terms of number of unknowns. This tuning is mostly needed to get a little closer to the optimum mesh but does not hinder the overall performance of the methods.

The numerical results presented in the last two chapters demonstrate the advantages of our approach for multigroup problems with respect to both the goal-oriented *hp*-adaptive strategy and the standard *hp*-adaptive strategy in energy norm. Numerical validation was thorough and extensive; various one-group, two-group and seven-group problems were analyzed in 1-D.

In conclusion, we believe that the method present here for 1-D multigroup eigenproblems can be extended to multi-dimension cases and will help us develop future mesh adaptation for multigroup transport eigenproblems as well.

REFERENCES

1. K.S. Smith, “Assembly Homogenization Techniques for Light Water Reactor Analysis”, *Prog. Nucl. Energy*,**17**(3), 303, (1986).
2. Workshop on Simulation and Modeling for Advanced Nuclear Energy Systems, Washington, D.C. August 15-17, (2006).
3. G.G. Bilodeau, W.R. Cadwell, J.P. Dorsey, J.G. Fairey, R.S. Varga, “PDQ - An IBM-704 Code to Solve the Two-Dimensional Few-Group Neutron Diffusion Equations”, WAPD-TM-70, Bettis Atomic Power Laboratory, Westinghouse Electric Corp., Pittsburgh (1957).
4. D.R. Ferguson and K.L. Derstine, “Optimized Iteration Strategies and Data Management Considerations for Fast Reactor Finite Difference Diffusion Theory Codes”, *Nucl. Sci. Engi.*, **64**, 593, (1977).
5. K.L. Derstine, “DIF3D: A Code to Solve One-, Two and Three-Dimensional Finite-Difference Diffusion Theory Problems”, ANL-82-64, Argonne National Laboratory, Chicago (1984).
6. J.J. Doming, “Modem Coarse-Mesh Methods - A Development of the '70s”, *Proc. Topical Meeting on Computational Methods in Nuclear Engineering, 3,* p1-31, Williamsburg, Virginia, April 23-25, (1979).
7. R.D. Lawrence, “Progress in Nodal Methods for the Solution of the Neutron Diffusion and Transport Equations”, *Prog. Nucl. Energy*, **17**, 271, (1986).
8. U.S. DOE Nuclear Energy Research Advisory Committee and the Generation IV International Forum, “A Technology Roadmap for Generation IV Nuclear Energy Systems”, available at http://gif.inel.gov/roadmap/, December (2002).
9. O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, *The Finite Element Method*, 6 ed., Butterworth-Heinemann, Oxford (2005).

10. C.M. Kang, and K.E Hansen, “Finite Element Methods for Reactor Analysis”, *Nucl. Sci. Engi.,* **51**, 456, (1973).

11. A. Kavenoky and J.J. Lautard, “The Neutron Kinetics and Thermal-Hydraulic Transient Computational Module of the Neptune System: CRONOS”, *Proc. Topical Meeting on Advances in Reactor Physics and Core Thermal Hydraulics,* 781-792, Kiamesha Lake, NY, September 22-24, (1982).

12. M.L. Adams and W.R. Martin, “Diffusion Synthetic Acceleration of Discontinuous Finite Element Transport Iterations”, *Nucl. Sci. Eng.* **111**, 145, (1992).

13. T.A. Wareing, J.M. McGhee, J.E. Morel, and S.D. Pautz, “Discontinuous Finite Element SN Methods on Three-Dimensional Unstructured Grids”, *Nucl. Sci. Eng.* **138**, 256, (2001).

14. L.F. Demkowicz, J.T. Oden, W. Rachowicz, and O. Hardy, “Toward a Universal *h-p* Adaptive Finite Element Strategy, Part I. Constrained Approximation and Data Structure”, *Computer Methods in Applied Mechanics and Engineering*, **77**, 79, (1989).

15. J.T. Oden, L.F. Demkowicz, W. Rachowicz, and T.A. Westermann, “Toward a Universal *h-p* Adaptive Finite Element Strategy, Part II. A Posteriori Error Estimation”, *Computer Methods in Applied Mechanics and Engineering*, **77**, 113, (1989).

16. W. Rachowicz, L.F. Demkowicz, and J.T. Oden, “Toward a Universal *h-p* Adaptive Finite Element Strategy, Part III. Design of *h-p* Meshes”, *Computer Methods in Applied Mechanics and Engineering*, **77**, 181, (1989).

17. H. Zhang, E.E. Lewis, “Spatial Adaptivity Applied to the Variational Nodal Pn Equations,” *Nucl. Sci. & Eng.* **142**, 1-7 (2002).

18. J.S. Warsa and A.K Prinja, “*p*-Adaptive Numerical Methods for Particle Transport", *Trans. Theo. Stat. Phys.*, **28**, 229 (1999).

19. R. Verfurth, *A Review of a posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*, Wiley & Teubner, New York, Stuttgart (1996).

20. M. Ainsworth, J.T. Oden, "*a posteriori* Error Estimation in Finite Element Analysis", *Comput. Meth. Appl. Mech. Engrg.*, **142**, 1, (1997).
21. L.F. Demkowicz, P.B. Monk, L. Vardapetyan, and W. Rachowicz. "De Rham Diagram for *hp* Finite Element Spaces",*Computers & Mathematics with Applications*, **39**(7), 29, (2000).
22. L.F. Demkowicz, "Projection-Based Interpolation," ICES Report 04-03, University of Texas at Austin, (2004).
23. C. Schwab, *p and hp-Finite Element Methods*, Clarendon Press, Oxford, (1998).
24. B. Guo and I. Babuska, "The *h-p* Version of the Finite Element Method, Part 1. The Basic Approximation Results. Part 2. General Results and Applications", *Comput. Mech.*, **1**, 21-41, 203-220, (1986).
25. I. Babuska and M. Suri, "The *h-p* Version of the Finite Element Method with Quasi-Uniform Meshes", *RAIRO Math. Mod. And Numer.*, **21**(2), 199, (1987).
26. W. Gui and I. Babuska, "The *h*, *p* and *h-p* Versions of the Finite Element Method in One Dimension, part 1: The error analysis of the *p*-version, part 2: The error analysis of the *h*- and *hp*-versions. part 3: The adaptive *hp* vesrion", *Numer. Math*. **49**, 577, (1986).
27. J.T. Oden and S. Prudhomme, "Goal-Oriented Error Estimation and Adaptivity for the Finite Element Method," *Comput. Math. Appl.*, **41**, 735, (2001).
28. P. Solin, L.F. Demkowicz "A Goal-oriented *hp*-Adaptivity for Elliptic Problems", *Comput. Methods Appl. Mech. Engrg.,* **93**, 449, (2004).
29. P. Solin, K. Segeth, I. Dolezel, *Higher-Order Finite Element Methods*, Chapman & Hall/CRC, Boca Raton (2004).
30. J.E. Flaherty, Course notes of Finite Element Analysis, available at, (http://www.cs.rpi.edu/~flaherje/), (2006).
31. L.F. Demkowicz, Notes of Finite Elements course, available at, (http://www.ticam.utexas.edu/%7Eleszek/classes/EM394F/book.pdf), (2006).
32. I. Babuska, T. Strouboulis, *The Finite Element Method and its Reliability*, Oxford

University Press, Oxford (2001).

33. S.C. Brenner, L.R. Scott, “The Mathematical Theory of Finite Element Methods”, Springer Verlag, New York (2002).
34. L.F. Demkowicz, C.W. Kim, "1D *hp*-Adaptive Finite Element Package. Fortran 90 Implementation (1D*hp*90)", TICAM Report 99-38, University of Texas at Austin, (1999).
35. T.J. Oden, “Some historic comments on finite elements,” *Proceedings of the ACM Conference on History of Scientific and Numeric Computation*, Princeton, NJ, United States, May 13-15, (1987).
36. O.C. Zienkiewicz and J.Z. Zhu, “The Super-Convergent Patch Recovery and *a posteriori* Error Estimation. Part I The Recovery Technique”, *Int. J. Numer. Meth. Engrg.*, **33**, 1365, (1992).
37. O.C. Zienkiewicz and J.Z. Zhu, “The Super-Convergent Patch Recovery and *a posteriori* Error Estimation. Part II Error Estimates and Adaptivity”, *Int. J. Numer. Meth. Engrg.*, **33**, 1365, (1992).
38. R.E. Bank, A. Weiser, “Some A Posteriori Error Estimators for Elliptic Partial Differential Equations”, *Math. Comp.*, **44**, 283, (1985).
39. M. Ainsworth, J.T. Oden, “A Posteriori Error Estimation in Finite Element Analysis,” *Comput. Meth. Appl. Mech. Engrg.*, **142**, 1 (1997).
40. I. Babuska, R.B. Kellogg, and J. Pitkaranta, “Direct and inverse error estimates for finite elements with mesh refinement,” *Numer. Math.*, **33,** 447, (1979).

VITA

Name:	Yaqi Wang
Address:	Department of Nuclear Engineering
	Texas A&M University
	3133 TAMU
	College Station, TX 77843-3133
Email Address:	yaqiwang@tamu.edu
Education:	B.S., Nuclear Engineering, Tsinghua University, 1996
	M.S., Nuclear Engineering, Texas A&M University, 2006

Before starting graduate study in the Department of Nuclear Engineering of Texas A&M University Mr. Wang worked with Institute of Nuclear Energy and Technology (INET) in Beijing for eight years. He will continue pursuing PhD in TAMU. His research interests include finite element transport theory, nuclear reactor monitoring, analysis and design and high performance computing.