Université de Montréal

Application of the Collocation Technique to the Spatial Discretization of the Generalized Quasistatic Method for Nuclear Reactors

par

Alain Monier
Institut de génie énergétique
ÉCOLE POLYTECHNIQUE

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU GRADE DE PHILOSOPHIÆ DOCTOR (Ph.D.)
(Génie nucléaire)
December 1991

© droits réservés de Alain Monier 1991

UNIVERSITÉ DE MONTRÉAL
ÉCOLE POLYTECHNIQUE

Cette thèse intitulée:

Application of the Collocation Technique to the Spatial Discretization of the Generalized Quasistatic Method for Nuclear Reactors

présentée par: Alain Monier
en vue de l'obtention du grade de: Philosophiæ Doctor (Ph.D.)
a été dûment acceptée par le jury d'examen constitué de:

M. Rozon Daniel, Ph.D., président
M. Hébert Alain, Ph.D., membre et directeur de recherche
M. Koclas Jean, Ph.D., membre et co-directeur de recherche
M. Rouben Ben, Ph.D., membre
M. Marleau Guy, Ph.D., membre

à Nathalie

ABSTRACT

The analysis of nuclear reactor power transients which exhibit large spatial changes in flux shape is a formidable problem in computational physics. The objective of this thesis is to develop a computationally efficient numerical method for solving the time-dependant neutron diffusion equation. The two-group multidimensional diffusion equation is solved using the improved and generalized quasistatic methods with nodal-collocation spatial discretization and finite-difference temporal discretization. The quasistatic methods factorize the flux into an quickly varying amplitude and slowly varying shape function equation. Numerical techniques used to solve the shape function equation are based on a two-parameter variational acceleration technique in which the new trial directions are generated by pre-multiplying the residual by a preconditioning matrix. A new technique to solve the amplitude equations is presented and is based on approximating the integrating polynomial by a second-order polynomial for some of the equations. The effect of not applying the constraint, typically imposed in quasistatic method, was also investigated.

The result is the creation of XSTATIC code. A series of standard benchmark problems are presented and used to validate the code. The results produced by the XSTATIC program compare well to benchmark solutions.

SOMMAIRE

L'analyse de la cinétique espace-temps du flux neutronique est un problème numérique difficile dans le domaine de la physique des réacteurs. L'objectif de cette thèse est le développement d'un programme efficace pour résoudre l'équation de la diffusion neutronique dépendante du temps. L'équation de diffusion multidimensionnelle à deux groupes d'énergie est résolue par la méthode quasistatique améliorée et quasistatique généralisée. La variable spatiale est discrétisée par la méthode de collocation nodale et la variable temporelle par la méthode des différences-finies.

Pour les méthodes quasistatiques, on suppose que le flux peut être factorisé en une fonction de forme $\vec{\varphi}(\hat{r}, t)$, qui varie lentement, et une fonction d'amplitude $T(t)$ qui contient les variations rapides. Dans cette méthode numérique la fonction de forme est calculée moins souvent que la fonction d'amplitude. Pour les réacteurs de type CANDU, la fonction de forme varie lentement et les méthodes quasistatiques sont efficaces. De toutes les méthodes quasistatiques (approximation normale, approximation pseudoquasistatique et approximation adiabatique), seule la méthode améliorée ne néglige aucun aspect physique et si les intervalles de temps entre les calculs de la fonction de forme sont petits, la méthode donne la réponse exacte.

L'équation de la fonction de forme est dérivée en substituant le flux factorisé dans l'équation de diffusion neutronique dépendante du temps. L'équation de la fonction d'amplitude est dérivée par l'application d'une fonctionelle l à l'équation de forme factorisé. Comme dans la plupart des méthodes quasistatiques, la fonctionelle l consiste en une pondération par une fonction de poids, souvent le flux adjoint initial, et en une intégration sur le volume du réacteur. L'équation de la fonction d'amplitude ressemble à l'équation de la cinétique pontuelle.

La factorisation du flux permet plusieurs solutions de la forme $\vec{\varphi}_*(\hat{r},t) = \frac{\vec{\varphi}(\hat{r},t)}{c(t)}$ ou $T_*(t) = T(t)c(t)$. Toutes les méthodes quasistatiques à date déterminent une solution unique en imposant une contrainte où $l\{\vec{\varphi}(\hat{r},t)\}$ est constante. Dans cette recherche, la contrainte n'est pas imposée mais elle est laissée flottante. Ceci introduit un autre terme dans l'équation de la fonction d'amplitude. Si on impose la contrainte que ce terme est nul, on retrouve l'équation d'amplitude traditionelle.

La discrétisation de la variable spatiale est réalisée par la méthode de collocation nodale. La méthode de différences-finies centrée est équivalente à la méthode de collocation nodale de premier ordre. La discrétisation de la variable temporelle dans l'équation de la forme et de l'amplitude est réalisée par la méthode des différences-finies.

Après discrétisation de l'équation de la forme, le système matriciel est résolu par une technique d'accélération à deux paramètres avec un préconditionnement ADI (Alternating Direction Implicit).

L'équation de la fonction de l'amplitude est résolue par une nouvelle technique. Les équations qui donnent la dépendance de la fonction de forme possèdent un caractère "stiff" et sont résolues par la méthode de Gear. Les points (temps et amplitude) sont stockés durant l'intégration. La méthode d'intégration utilise un polynôme d'ordre élevé, qui passe à travers ces points, pour intégrer la fonction de forme. Les équations qui sont dépendantes de la fonction de l'amplitude sont obtenues en passant et en intégrant un polynôme de deuxième ordre à travers les points. Cette technique diminue le temps de calcul de trois à dix fois, car on réduit le nombre de calculs d'exposants qui sont très coûteux.

La solution des équations quasistatiques, qui comprend les équations de forme et d'amplitude, est faite par la méthode "fixed point". Cela signifie qu'on répète la résolution de la fonction de forme suivie de la résolution de la fonction d'amplitude jusqu'à ce que la différence entre deux itérations successives soit moins qu'un paramètre de convergence. Cette technique ne donne pas d'estimé de l'erreur mais

assure que la fonction de forme et d'amplitude sont consistants entre eux.

Quatre cas tests sont utilisés pour valider la méthode quasistatique améliorée. Les cas SIMPLE, CRKB et AECL sont des idéalisations 1, 2 et 3 dimensionnelles qui représentent des transitoires neutroniques dans un réacteur CANDU après un accident de perte de caloporteur. Le quatrième cas test LMW est une représentation idéalisée d'une transitoire opérationelle dans un réacteur PWR. Pour tous ces cas tests, la méthode quasistatique améliorée donne de très bon résultats, soit avec la contrainte imposée ou flottante. Pour les cas où l'intervalle entre les calculs de forme est grand, la méthode sans contrainte donne de meilleurs résultats. Ceci est dû au fait qu'un degré de liberté additionel permet à la méthode de mieux suivre la transitoire de réactivité. En étudiant ce phénomène, on remarque que la réactivité se recourbe lorsque le bout des barres d'arrêt traverse des lignes de réseau.

L'influence du choix de la fonction de poids sur la transitoire de puissance n'est pas très marqué pour les réacteurs de type CANDU. La réactivité et le temps de génération démontrent de grandes variations. Si on se sert du flux initial au lieu de l'adjoint initial comme fonction de poids pour le cas test AECL, on retrouve la même transitoire de puissance, mais la réactivité maximale est de 7.3 mk au lieu de 6.3 mk. Ceci est permis car la réactivité est un paramètre d'origine mathématique et n'est pas mesurable. Donc la réactivité n'est pas unique mais elle dépend de la fonction de poids.

La méthode quasistatique généralisée factorise le flux en une fonction de forme et une fonction d'amplitude dépendantes de l'énergie. Donc, chaque groupe d'énergie évolue avec sa propre fonction d'amplitude. La dérivation de cette méthode ressemble beaucoup à celle de la méthode quasistatique améliorée, sauf la fonctionelle l qui est seulement une intégration sur le volume du réacteur. Ceci a pour effect de transformer les paramètres scalaires comme la réactivité, le temps de génération et la contrainte dans une forme matricielle. Les méthodes numériques pour résoudre

les équations quasistatiques généralisées sont les mêmes que pour la méthode améliorée, sauf pour l'intégration de la fonction d'amplitude qui se fait au complet avec la méthode de Gear. Cette implantation de la méthode généralisée ne permet pas une contrainte flottante.

Les essais numériques portent sur les mêmes cas tests que pour la méthode quasistatique améliorée et démontrent que la méthode généralisée donne des résultats semblables à ceux de la méthode améliorée. Le temps de calcul pour la fonction de forme est réduit de 10%, mais le temps de calcul pour la fonction de l'amplitude augmente à 70% du temps total.

En résumant, nous avons développé une méthode quasistatique améliorée sans contrainte et une méthode quasistatique généralisée en se servant des nouvelles techniques numériques. Le résultat de cette thèse est un programme dénommé XSTATIC. La comparaison des solutions XSTATIC avec des réponses "benchmark" donne un très bon accord. Pour ces cas tests, les nouvelles méthodes numériques sont très fiables et rapides.

TABLE OF CONTENTS

DEDICATION iv

ABSTRACT v

SOMMAIRE vi

TABLE OF CONTENTS x

LIST OF FIGURES xiv

LIST OF TABLES xvi

LIST OF APPENDICES xviii

NOMENCLATURE xix

1 Introduction 1

1.1 Review of Solution Method 1

1.1.1 Point Kinetics Method 2

1.1.2 Modal Expansion Method 2

1.1.3 Improved Quasistatic Method 3

1.1.4 Generalized Quasistatic Method 4

1.2 Justification of the Quasistatic Method 4

1.3 Objective 5

2 Improved Quasistatic Method 7

2.1 Overview 7

2.2 Time-Dependent Neutron Diffusion Equation 7

2.3 Derivation of the Unconstrained Improved Quasistatic Method . . . 8
2.4 Reduced Two-Energy-Group Formulation of Quasistatic Equations . 13
2.5 Choice of a Weight Function 16
2.6 Discretization of Shape Function Equation 18
2.7 Discretization of Point Kinetics Equation 21
2.8 Discretization of Precursor Equation 22
2.9 Quasistatic Solution Strategy 23
2.9.1 Update Nuclear Properties - PHASE4 26
2.9.2 Shape Function Algorithm - FLXQSM 26
2.9.2.1 Iterative Strategy 27
2.9.2.2 Acceleration 28
2.9.2.3 Convergence Test 29
2.9.2.4 Application of the Constraint 29
2.9.3 Point Kinetics Solution - FASTQ 30
2.9.4 Precursor Update - PRECDY 33
2.9.5 IQS Convergence Test 33

3 Numerical Results for the Improved Quasistatic Method 35
3.1 SIMPLE Test Case 36
3.1.1 Performance of the FASTQ Algorithm 38
3.1.2 Spectral Radius of Residual Matrix **R** 42
3.1.3 Choice of the Weight Function 44
3.2 LMW Test Case 47
3.3 CRKB Test Case 52
3.3.1 Constrained XSTATIC Results 54
3.3.2 Unconstrained XSTATIC Results 55
3.4 AECL Test Case 57
3.4.1 Constrained XSTATIC Results 57

3.4.2 Linear Variation of Point Kinetics Parameters 58
3.4.3 Rod Cusping 63
3.4.4 Unconstrained XSTATIC Results 64
3.4.5 Galerkin Weighting 67
3.4.6 Computational Time 69

4 Generalized Quasistatic Method 72

4.1 Overview 72
4.2 Derivation of the Unconstrained Generalized Quasistatic Method . . 73
4.3 Reduced Two-Energy-Group Formulation of Quasistatic Equations . 77
4.4 Discretization of the Generalized Shape Function Equation 78
4.5 Discretization of Generalized Point Kinetics Equation 79
4.6 Discretization of Precursor Equation 80
4.7 Generalized Quasistatic Solution Strategy 81
4.7.1 Shape Function Algorithm - FLXGQS 82
4.7.1.1 Application of the Constraint 84
4.7.2 Generalized Point Kinetics Solution - DGEAR 85
4.7.3 Precursor Update - PRECDY 86
4.7.4 Convergence Test 87

5 Numerical Results for the Generalized Quasistatic Method 88

5.1 SIMPLE Test Case 89
5.1.1 System Poles 90
5.2 LMW Test Case 96
5.3 AECL Test Case 97
5.3.1 Galerkin Weighting 98
5.3.2 Computational Time 99

6 Conclusions **100**

6.1 IQS Method Conclusions 100

6.2 GQS Method Conclusions 101

6.3 Recommendations for Further Research 102

BIBLIOGRAPHY **103**

LIST OF FIGURES

2.1 Time Step Hierarchy 24

2.2 XSTATIC QUASISTATIC ALGORITHM 25

3.1 Variation of γ_3/T for a Step Perturbation 39

3.2 Variation of γ_3/T for a Ramp Perturbation 39

3.3 Error in γ_3/T for FASTQ 41

3.4 Error in γ_3/T for STIFFZ 41

3.5 Ratio of STIFFZ to FASTQ CPU times 42

3.6 Largest and Smallest Eignvalues of $\mathbf{R}$ 43

3.7 Largest and Smallest Eignvalues of $\mathbf{R}^*$ 44

3.8 Error in Fluxes Computed by Adjoint Weighting 45

3.9 Error in Fluxes Computed by Galerkin Weighting 46

3.10 LMW: Shape function at Several Times 51

3.11 AECL: Variation of Reactivity for CERBERUS like Definitions . 60

3.12 AECL: Variation of Generation Time for CERBERUS like Definitions 61

3.13 AECL: Variation of Reactivity During a Macro Time Step 63

3.14 AECL: Variation of Reactivity During a Macro Time Step 65

3.15 AECL: Variation of Constraint 66

3.16 AECL: Generation Time versus Time for Galerkin Weighting . . . 68

3.17 AECL: Reactivity versus Time for Galerkin Weighting 68

4.1 Generalized QUASISTATIC Algorithm 83

5.1 SIMPLE: Error in Poles Computed by Adjoint Weighting 94

5.2 SIMPLE: Error in Poles Computed by Galerkin Weighting 95

5.3 Error in Fluxes Computed by Adjoint Weighting 95

B.1 LMW GEOMETRY 116

B.2 CKRB-2 GEOMETRY 118

B.3 Vertical cross-section at Z = 0 cm showing region assignment for 0 < Z < 300 cm 124

B.4 Horizontal cross-section at Y = 390 cm showing region asssignment 125

B.5 Vertical cross-section at Z = 600 cm showing region assignment for 300 < Z < 600 CM 126

B.6 Vertical cross-section at Z = 0 CM illustrating grid layout in the XY plane 127

B.7 Horizontal cross-section at Z = 390 cm illustrating grid layout in the XZ plane 128

LIST OF TABLES

3.1 Parameters for SIMPLE Case 37

3.2 SIMPLE: XSTATIC and Reference Powers 40

3.3 LMW: Reference Total Power [Greenman] 47

3.4 LMW: XSTATIC Results for LAGR1 49

3.5 LMW: XSTATIC Results for MCFD2 49

3.6 LMW: Unconstrained XSTATIC Results 50

3.7 CRKB: CERKIN and ADEP Results for Average Thermal Flux . 53

3.8 CRKB: XSTATIC Results for Average Thermal Flux 53

3.9 CRKB: Results for Local Thermal Flux 54

3.10 CRKB: Unconstrained XSTATIC Results for Average Thermal Flux 56

3.11 CRKB: Reactivity Transients 56

3.12 AECL: Power versus Time 58

3.13 AECL: Constrained XSTATIC Results with CEBERUS Reference (tsplit=msplit=1) 59

3.14 AECL: XSTATIC Results with Linear Reactivity with CERBERUS Reference (tsplit=msplit=1) 61

3.15 AECL: Constrained XSTATIC Results (tsplit=msplit=1) 62

3.16 AECL: Unconstrained XSTATIC Results (tsplit = msplit = 1) . . 64

3.17 AECL: Constrained XSTATIC Results with Galerkin Weighting (tsplit= msplit=1) 69

3.18 AECL: Amplitude Function versus Time for Adjoint and Galerkin Weighting 70

3.19 AECL Computational Time for the IQS Method 71

5.1 SIMPLE:Average Fluxes for Reference Solution 90

5.2 SIMPLE:Average Fluxes for $\rho = 4.94$ mk 91
5.3 SIMPLE:Average Fluxes for $\rho = -29.2$ mk 91
5.4 System Poles for SIMPLE Test Case 93
5.5 LMW: GQS XSTATIC Results for LAGR1 96
5.6 AECL: Constrained GQS Results (tsplit=1,msplit=1) 97
5.7 AECL: Constrained GQS Results (tsplit=8,msplit=1) 98
5.8 AECL: GQS Results Using Galerkin Weighting (tsplit=1,msplit=1) 98
5.9 AECL: Computational Time for the GQS Method 99

A.1 SIMPLE TEST CASE DESCRIPTION 106

B.1 SIMPLE TEST CASE DESCRIPTION 113
B.2 LMW TEST CASE DESCRIPTION 117
B.3 CKRB-1 TEST CASE DESCRIPTION 121
B.4 AECL TEST CASE DESCRIPTION 122

LIST OF APPENDICES

A Programmer Guide to the Code POLES **106**

A.1 Static Equation 107

A.2 Adjoint Problem 107

A.3 Transient Solution 108

B Test Case Descriptions **111**

B.1 SIMPLE Test Case Description 111

B.1.1 Sample Input 111

B.2 LMW Test Case Description 114

B.2.1 Sample Input 114

B.3 CRKB-2 Test Case Description 118

B.3.1 Sample Input 118

B.4 AECL Test Case Description 122

B.4.1 Sample Input 129

C User Guide for XSTATIC **139**

C.1 Description of phase TRAN 139

C.2 Description of the phase PERT 141

NOMENCLATURE

Bold face quantities are matrices and $\vec{X}$ denotes a multigroup scalar quantity.

$\mathbf{A}$	neutron absorption matrix
$\mathbf{B}$	neutron production matrix
$\mathbf{C}$	precursor concentrations
$\mathbf{D}$	diagonal diffusion coefficients matrix
$\mathbf{S}$	scattering and removal matrix
$\mathbf{V}^{-1}$	diagonal inverse velocity matrix
$\vec{F}$	neutron production operator
G	number of energy groups
k_{eff}	effective multiplication factor
$\hat{r}$	position vector
N	number of precursor groups
T	IQS amplitude function
$\vec{T}, \mathbf{T}$	GQS amplitude function
t	time
$\vec{W}$	IQS weight Function
$\mathbf{W}$	GQS weight Function
β_i	delayed fraction for i^{th} precursor group
β	total delayed fraction
$\frac{\boldsymbol{\beta}}{\boldsymbol{\Lambda}}$	GQS total delayed neutron fraction
$\frac{\boldsymbol{\beta}_i}{\boldsymbol{\Lambda}}$	GQS delayed fraction for i^{th} precursor group
Λ	IQS generation time
λ_s, $\boldsymbol{\lambda}_s$	IQS and GQS constraint

ρ	reactivity
$\frac{\boldsymbol{\rho}}{\boldsymbol{\Lambda}}$	GQS reactivity
$\vec{\varphi}$, $\boldsymbol{\varphi}$	shape function
$\vec{\phi}$	flux
$\vec{\chi}_p$	prompt neutron fission spectrum
$\vec{\chi}_i$	delayed neutron fission spectrum for i^{th} precursor group
$\boldsymbol{\chi}$	discretized delayed neutron spectrum

CHAPTER 1

Introduction

The safety analysis and design of nuclear reactors require accurate knowledge of the time-dependent spatial flux distribution. The more accurate this information is, the more economical, reliable and safe will be the reactor design. However, the nature of the problem makes the computation of the time-dependent flux distribution difficult. Further, such calculations often require significant computer resources. Therefore, there exists a substantial economic incentive in reducing the cost of these calculations.

The transient behavior of the flux distribution in a nuclear reactor core is well modelled by the time-dependent neutron transport equation. This equation is however too complex to be solved analytically or even numerically. It is sufficient for most commercial reactors to use a low order angular approximation to model the angular variation of flux distribution. This simplifying assumption leads to the time-dependent neutron diffusion equation. The problem is further simplified by the use of multigroup theory to model the energy dependence.

1.1 Review of Solution Method

Solving even the simplified time-dependent neutron diffusion equation for a large homogeneous reactor is still a formidable task. There exist many different methods to treat this problem, some of which are specialized to a particular reactor design. Some methods apply brute force and try to solve the raw diffusion equation. Others methods attempt to approximate and/or manipulate their way to a cheaper and reasonably accurate solution. A summary of these sophisticated methods can be found in references.[1, 2, 3]

The following sections summarize some well known methods.

1.1.1 Point Kinetics Method

The basis of this method is the separation of the temporal and spatial variables in the flux $\vec{\phi}(\hat{r},t)$ [1] as:

$$\vec{\phi}(\hat{r},t) \simeq \vec{\varphi}(\hat{r},t_0)T(t) \tag{1.1}$$

where t_0 is the start of transient time and $\varphi(\hat{r},t_0)$ is usually the start of transient flux. An integration over the reactor volume is performed to eliminate the spatial dependence. As a result the reactor is treated as a point, thus the name point kinetics. This method illustrates well the principles of reactor dynamics. However, large and usually unacceptable errors are obtained even for relatively mild transients if the transient flux shape is significantly different from the initial flux shape.

The generalized point kinetics equation separates the temporal and spatial dependence of the flux $\vec{\phi}(\hat{r},t)$ as:

$$\vec{\phi}(\hat{r},t) \simeq \mathbf{T}(t)\,\vec{\varphi}(\hat{r},t_0) = \begin{pmatrix} \vdots \\ T_g(t)\varphi_g(\hat{r},t_0) \\ \vdots \end{pmatrix} \tag{1.2}$$

where $\mathbf{T}(t)$ is a $g \times g$ diagonal matrix, with elements $T_g(t)$ on the diagonal and 0 off the diagonal.

1.1.2 Modal Expansion Method

The modal expansion method attempts to drastically reduce the computational effort by expanding the flux $\vec{\phi}(\hat{r},t)$ into a series of trial functions giving:

$$\vec{\phi}(\hat{r},t) \simeq \sum_i \vec{\varphi}_i(\hat{r})T_i(t) \tag{1.3}$$

[1] $\vec{X}$ denotes a multigroup vector quantity and a bold quantity denotes a multigroup matrix quantity

where:

$\vec{\varphi}_i(\hat{r})$	predetermined flux shapes
$T_i(t)$	amplitude functions

The accuracy of this method depends, of course, on the number and on the choice of the trial functions $\vec{\varphi}_i(\hat{r})$. The trial functions can be chosen to be natural modes, lambda modes, Green's function modes, synthesis modes or the analytic modes. An example of a code using a modal expansion method is SMOKIN.[4] This code, developed by Ontario Hydro, uses lambda modes. Although these methods require less computational effort, their major drawback is a lack of error analysis, which limits the applicability of this method.

1.1.3 Improved Quasistatic Method

The improved quasistatic method (IQS) [5, 6] factors the flux $\vec{\phi}(\hat{r},t)$ into a slowly varying shape function $\vec{\varphi}(\hat{r},t)$ and a quickly varying amplitude function T(t). The equation has the following form:

$$\vec{\phi}(\hat{r},t) \quad = \quad \vec{\varphi}(\hat{r},t)T(t) \tag{1.4}$$

Note that this quasistatic factorization is mathematically correct, whereas the separation of variables performed by the point kinetics method is only approximate. This factorization splits the time-dependent diffusion equation into a point kinetics equation and a shape function equation.

Other forms of the quasistatic method are: the adiabatic method, the pseudo-quasistatic method, and the ordinary quasistatic method. The treatment of the temporal derivatives distinguishes the various quasistatic methods. Only the "improved" quasistatic method approach is guaranteed to approach the correct solution as the time step diminishes.

The IQS method is used by codes such as CERBERUS[7], to solve 3-dimensional 2-energy-group CANDU reactor problems.

1.1.4 Generalized Quasistatic Method

The generalized quasistatic method[8] (GQS) factors the flux $\vec{\phi}(\hat{r},t)$ in the same manner as the quasistatic method. However, the generalized quasistatic method seeks to reduce the overall computational effort further than the quasistatic methods by providing an amplitude function T_g for each energy group. The factorized flux can thus be expressed as:

$$\vec{\phi}(\hat{r},t) \quad = \quad \mathbf{T}(t)\,\vec{\varphi}(\hat{r},t) \quad = \begin{pmatrix} \vdots \\ T_g(t)\varphi_g(\hat{r},t) \\ \vdots \end{pmatrix} \tag{1.5}$$

where $\mathbf{T}(t)$ is a $g \times g$ diagonal matrix, with elements $T_g(t)$ on the diagonal and 0 off the diagonal.

The intent is that the increased computational effort in solving a more complicated set of point kinetics equations will be more than balanced by a decrease in the computational time spent solving the shape function equation.

1.2 Justification of the Quasistatic Method

The quasistatic method is based on the assumption that the flux can be factorized into a slowly varying shape function and a more rapidly varying amplitude function. Of course, the validity of this assumption varies with the type of reactor and the severity of the transient. In physical terms the quasistatic method works well for reactors for which

$$C/L \quad \text{is} \quad \text{small},$$

where C and L are the characteristic length of the reactor and the neutron diffusion length, respectively. If the quantity C/L is small, perturbations are felt throughout

the reactor forcing it to react as a unit. In other words, the spatial variations are small and the amplitude variations are large. However, if C/L is large, the effect of a perturbation is localized and as a result large spatial and small amplitude variations are observed. Large spatial variations usually violate the assumption of a slowly varying shape function, a basic premise of the quasistatic method, and hence make it less efficient in these cases.

Thus for LMFBRs and heavy water reactors (like CANDUs) where C/L is small, the quasistatic method is cost effective. In contrast, the method is less efficient for PWRs, as C/L is generally larger.

The performance of the quasistatic method is also dependent on the type of transient. Viewing the quasistatic method as a refined point kinetics method, the transients that are well modelled by the point kinetics method are equally well modelled by the quasistatic method. For a homogeneous one-speed reactor subject to a uniform perturbation the point kinetics solution is exact. Thus for little effort the quasistatic method also gives the exact solution. These types of transient rarely occur and are useful only for academic purposes. A more useful transient involves is the asymptotic behavior of the flux a "long time" after the last perturbation. The reactor is still evolving but changes in the shape function are small and tending toward zero. This case maximizes the performance of the quasistatic method.

1.3 Objective

The goal of this research is to develop a generalized quasistatic code, named XSTATIC, to simulate reactor excursions. This code is able to solve realistic 3-dimensional reactor problems quickly and accurately.

The first step is the development of an improved quasistatic code using the linear system techniques developed by A. Hebert[10]. When this is accomplished, a code similar to the CERKIN-CERBERUS family will have been developed. The

IQS version of XSTATIC will be validated via a series of benchmark problems.

The generalized quasistatic method will then be implemented into XSTATIC. It is hoped that the extra computational effort associated with several amplitude functions will result in higher accuracy, compared to the improved quasistatic method, for equal time steps or equal accuracy with larger time steps.

As always with any transient problem the stationary equation must first be solved. This is done with TRIVAC[11] which solves the static neutron diffusion equation using the nodal collocation method. TRIVAC is used in a modular fashion as a foundation on which to build the transient code.

CHAPTER 2

Improved Quasistatic Method

2.1 Overview

Most of the work, to date, in the area of the improved quasistatic method has involved a formulation where a constraint or a normalization is applied to the multigroup shape function [3, 5, 6]. In this chapter, a derivation of the unconstrained (or floating constraint) version of the improved quasistatic (IQS) method is presented. Both the constrained and unconstrained versions are used to solve the time-dependent neutron diffusion equation. Note that the constrained IQS method is obtained by assuming the constraint to be constant.

The resulting equations are then discretized using a hybrid nodal collocation - finite-element method for the shape function and finite-differences for the amplitude function. A fixed-point iteration scheme is used to simultaneously solve the shape and amplitude equations. New methods for solving the shape function equation using variational acceleration and for integrating the amplitude equations are also presented.

2.2 Time-Dependent Neutron Diffusion Equation

The transient behaviour of a thermal reactor core is well modelled by the time-dependent neutron diffusion equation, which is a low-order angular approximation to the transport equation. After energy-group discretization, the multigroup diffusion equation in G-energy-groups can be written in the following differential form:

$$\mathbf{V}^{-1}\frac{\partial}{\partial t}\vec{\phi}(\hat{r},t) = \left[\nabla\cdot\mathbf{D}(\hat{r},t)\nabla - \mathbf{S}(\hat{r},t) + (1-\beta)\vec{\chi}_p\vec{F}^T(\hat{r},t)\right]\vec{\phi}(\hat{r},t) + \sum_{i=1}^{N}\vec{\chi}_i\lambda_i C_i(\hat{r},t) \tag{2.1}$$

$$\frac{\partial}{\partial t}\vec{\chi}_i C_i(\hat{r},t) = \beta_i\vec{\chi}_i\vec{F}^T(\hat{r},t)\vec{\phi}(\hat{r},t) - \lambda_i\vec{\chi}_i C_i(\hat{r},t) \tag{2.2}$$

where:

$\mathbf{D}(\hat{r},t)$	diagonal diffusion coefficient matrix	$(G\times G)$
$\mathbf{S}(\hat{r},t)$	total removal minus scattering matrix	$(G\times G)$
$\mathbf{V}^{-1}$	diagonal inverse velocity matrix	$(G\times G)$
$\vec{\chi}_p$	prompt neutron fission spectrum	$(G\times 1)$
$\vec{\chi}_i$	delayed neutron fission spectrum for i^{th} precursor group	$(G\times 1)$
$\vec{F}(\hat{r},t)$	neutron production operator	$(G\times 1)$
β_i	delayed fraction for i^{th} precursor group	scalar
β	total delayed fraction	scalar
N	number of delayed neutron precursor groups	scalar

Equation 2.1 is known as the flux equation and 2.2 as the precursor equation.

2.3 Derivation of the Unconstrained Improved Quasistatic Method

The conceptual basis of the quasistatic method is the factorization of the flux $\vec{\phi}(\hat{r},t)$ into a shape function $\vec{\varphi}(\hat{r},t)$ and an amplitude function T(t). Mathematically, it is represented as:

$$\vec{\phi}(\hat{r},t) = T(t)\quad\vec{\varphi}(\hat{r},t) \tag{2.3}$$

where:

$\vec{\phi}(\hat{r},t)$	is the flux	$\in X$
$\vec{\varphi}(\hat{r},t)$	is the shape function	$\in X$
$T(t)$	is the amplitude function	$\in R$

and X spans the space of $\vec{\phi}(\hat{r},t)$.

From a numerical standpoint, this factorization permits the division of a large stiff system into a large regular system, the shape function equation, and a stiff system, the amplitude equation. The temporal dependence of the slowly varying shape equation is solved using a low-order approximation with a large time step. A high-order method with a small time step is used to solve the amplitude equation. Since the spatial aspect of this calculation is the most time consuming, a reduction in the frequency of its computation should result in faster execution times.

Starting from the time-dependent neutron diffusion equation, a substantial amount of mathematical manipulation must be performed to derive the shape function and amplitude function equations.

The shape function equation is obtained by substituting the factored flux $T(t)\vec{\varphi}(\hat{r},t)$ into the time-dependent flux 2.1, giving:

$$\begin{aligned}\mathbf{V}^{-1}\frac{\partial}{\partial t}\left[\vec{\varphi}(\hat{r},t)T(t)\right] &= \left[\nabla\cdot\mathbf{D}(\hat{r},t)\nabla-\mathbf{S}(\hat{r},t)+(1-\beta)\vec{\chi}_p\vec{F}^T(\hat{r},t)\right]\vec{\varphi}(\hat{r},t)T(t)\\ &+ \sum_{i=i}^{N}\vec{\chi}_i\lambda_i C_i(\hat{r},t) \qquad (2.4)\end{aligned}$$

Similarly the factored precursor equation is:

$$\frac{\partial}{\partial t}\vec{\chi}_i C_i(\hat{r},t)=\beta_i\vec{\chi}_i\vec{F}^T(\hat{r},t)\vec{\varphi}(\hat{r},t)T(t)-\lambda_i\vec{\chi}_i C_i(\hat{r},t) \qquad (2.5)$$

Having rather simply obtained the shape function and factored precursor equation, the more complicated amplitude equation will now be derived.

Define a functional l having the following property:

$$l : X \rightarrow R \quad or \quad l\{\Omega(\hat{r},t)\} = f(t), \tag{2.6}$$

The amplitude or point kinetics equations can be found by applying the functional l to the factored flux equation, leading to:

$$l\left\{\mathbf{V}^{-1}\frac{\partial}{\partial t}\left[\vec{\varphi}(\hat{r},t)T(t)\right]\right\} = \tag{2.7}$$

$$l\left\{\left[\nabla\cdot\mathbf{D}(\hat{r},t)\nabla - \mathbf{S}(\hat{r},t) + (1-\beta)\vec{\chi}_p\vec{F}^T(\hat{r},t)\right]\vec{\varphi}(\hat{r},t)T(t)\right\} + \sum_{i=i}^{N} l\left\{\vec{\chi}_i\lambda_i C_i(\hat{r},t)\right\}$$

This derivation makes reference to, but never describes explicitly the functional l. Mathematically the only condition on l is that it maps X onto R. In fact the functional l is generally chosen so that the shape function equation is slowly varying. In other words a functional l is chosen so that the fast poles of the system are contained in the amplitude equation. Typically the functional l is a weighted integral over the volume, of the form:

$$l\{\vec{\Omega}(\hat{r},t)\} = \int \vec{W}^t(\hat{r})\vec{\Omega}(\hat{r},t)d^3r = \left\langle \vec{W}(\hat{r}), \vec{\Omega}(\hat{r},t)\right\rangle \tag{2.8}$$

where $\vec{W}(\hat{r})$ is independent of time and the Dirac notation $<,>$ is used to express integration over the volume of the reactor and summation over all energy groups.

The user is still faced with the choice of a weight function. It has been shown that the use of the adjoint flux as a weight function reduces the error in the computation of the reactivity. [12]

Assuming a functional of the form 2.8, equation 2.7 can be written as:

$$\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\frac{\partial}{\partial t}\left[\vec{\varphi}(\hat{r},t)T(t)\right]\right\rangle =$$

$$\left\langle \vec{W}(\hat{r}), \left[\nabla\cdot\mathbf{D}(\hat{r},t)\nabla - \mathbf{S}(\hat{r},t) + (1-\beta)\vec{\chi}_p\vec{F}^T(\hat{r},t)\right]\vec{\varphi}(\hat{r},t)T(t)\right\rangle$$

$$+\sum_{i=1}^{N}\left\langle \vec{W}(\hat{r}), \vec{\chi}_i\lambda_i C_i(\hat{r},t)\right\rangle \tag{2.9}$$

Applying the product rule to the left hand side of the equation 2.9, the time derivative can be rewritten as:

$$\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\frac{\partial}{\partial t}\left[\vec{\varphi}(\hat{r},t)T(t)\right]\right\rangle = T(t)\frac{d}{dt}\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle + \left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle \frac{d}{dt}T(t) \tag{2.10}$$

Substituting equation 2.10 in 2.9 and adding then subtracting the term representing the delayed neutron production, the following is obtained:

$$T(t)\frac{d}{dt}\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle + \left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle \frac{d}{dt}T(t) = \tag{2.11}$$
$$\left\langle \vec{W}(\hat{r}), \left[\nabla\cdot\mathbf{D}(\hat{r},t)\nabla - \mathbf{S}(\hat{r},t) + \left\{(1-\beta)\vec{\chi}_p + \sum_{i=1}^{N}\beta_i\vec{\chi}_i\right\}\vec{F}^T(\hat{r},t)\right]\vec{\varphi}(\hat{r},t)\right\rangle T(t)$$
$$-\left\langle \vec{W}(\hat{r}), \sum_{i=1}^{N}\beta_i\vec{\chi}_i\vec{F}^T(\hat{r},t)\vec{\varphi}(\hat{r},t)\right\rangle T(t) + \sum_{i=i}^{N}\left\langle \vec{W}(\hat{r}), \vec{\chi}_i\lambda_i C_i(\hat{r},t)\right\rangle$$

A similar procedure must also be performed to transform the factored precursor equation into the amplitude precursor equation. Applying the functional l to equation 2.5 and dividing by $\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle$ gives:

$$\frac{\left\langle \vec{W}(\hat{r}), \frac{\partial}{\partial t}\vec{\chi}_i C_i(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle} = \beta_i\frac{\left\langle \vec{W}(\hat{r}), \vec{\chi}_i\vec{F}^t(\hat{r},t)\vec{\phi}(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle} - \lambda_i\frac{\left\langle \vec{W}(\hat{r}), \vec{\chi}_i C_i(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle} \tag{2.12}$$

The above equation is transformed via the following quotient rule for derivatives

$$\frac{d}{dt}\frac{\left\langle \vec{W}(\hat{r}), \vec{\chi}_i C_i(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle} = \frac{\left\langle \vec{W}(\hat{r}), \frac{\partial}{\partial t}\vec{\chi}_i C_i(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle} - \frac{\frac{d}{dt}\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle}\frac{\left\langle \vec{W}(\hat{r}), \vec{\chi}_i C_i(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle} \tag{2.13}$$

and the final form of the amplitude precursor equation is obtained:

$$\frac{d}{dt}\frac{\left\langle \vec{W}(\hat{r}), \vec{\chi}_i C_i(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V^{-1}}\vec{\varphi}(\hat{r},t)\right\rangle} = \frac{\beta_i \left\langle \vec{W}(\hat{r}), \vec{\chi}_i \vec{F}^t(\hat{r},t)\vec{\phi}(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V^{-1}}\vec{\varphi}(\hat{r},t)\right\rangle} - \left(\lambda_i + \frac{\frac{d}{dt}\left\langle \vec{W}(\hat{r}), \mathbf{V^{-1}}\vec{\varphi}(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V^{-1}}\vec{\varphi}(\hat{r},t)\right\rangle}\right)\frac{\left\langle \vec{W}(\hat{r}), \vec{\chi}_i C_i(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V^{-1}}\vec{\varphi}(\hat{r},t)\right\rangle} \tag{2.14}$$

Grouping the terms similar to the amplitude or point kinetics parameters (equations 2.18 to 2.21) in equations 2.12 and 2.14, equations 2.15 and 2.16 are obtained. These are the point kinetics or amplitude equations:

$$\frac{d}{dt}T(t) = \left(\frac{\rho}{\Lambda}(t) - \frac{\beta}{\Lambda}(t) - \lambda_s(t)\right)T(t) + \sum_{i=1}^{N}\lambda_i\xi_i(t) \tag{2.15}$$

$$\frac{d}{dt}\xi_i(t) = \frac{\beta_i}{\Lambda}(t)T(t) - (\lambda_i + \lambda_s(t))\,\xi_i(t) \tag{2.16}$$

where the point kinetics parameters are:

$$\frac{\rho}{\Lambda}(t) = \frac{1}{\left\langle \vec{W}(\hat{r}), \mathbf{V^{-1}}\vec{\varphi}(\hat{r},t)\right\rangle} \times \tag{2.17}$$

$$\left\langle \vec{W}(\hat{r})\ ,\ \left[\nabla\cdot\mathbf{D}(\hat{r},t)\nabla - \mathbf{S}(\hat{r},t) + \left\{(1-\beta)\vec{\chi}_p + \sum_{i=1}^{N}\beta_i\vec{\chi}_i\right\}\vec{F}^T(\hat{r},t)\right]\vec{\varphi}(\hat{r},t)\right\rangle$$

$$\frac{\beta_i}{\Lambda}(t) = \beta_i\frac{\left\langle \vec{W}(\hat{r})\ ,\ \vec{\chi}_i\vec{F}^T(\hat{r},t)\vec{\varphi}(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle} \tag{2.18}$$

$$\frac{\beta}{\Lambda}(t) = \sum_{i=1}^{N}\frac{\beta_i}{\Lambda}(t) \tag{2.19}$$

$$\xi_i(t) = \frac{\left\langle \vec{W}(\hat{r})\ ,\ \vec{\chi}_i C_i(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V^{-1}}\vec{\varphi}(\hat{r},t)\right\rangle} \tag{2.20}$$

$$\lambda_s(t) = \frac{\frac{d}{dt}\left\langle \vec{W}(\hat{r}), \mathbf{V^{-1}}\vec{\varphi}(\hat{r},t)\right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V^{-1}}\vec{\varphi}(\hat{r},t)\right\rangle} \tag{2.21}$$

Normally the point kinetics equations are expressed in terms of ρ,Λ and β. However the above equations are expressed in terms of the ratios ρ/Λ and β/Λ.

> "In fact it would avoid a lot of confusion in the field if one of the three parameters ρ,Λ or β were eliminated and we talked about the reactor only in terms of the remaining ratios." *A.F. Henry* [12]

This approach has the further benefit of allowing a smoother transition to generalized point kinetics where the scalar quantities ρ/Λ and β/Λ become matrices, and the quantity Λ is not defined.

In the constrained version of the IQS technique the term $\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t) \right\rangle$ is required to be constant in time and consequently $\lambda_s(\mathrm{t}) = 0$. The amplitude equation simplifies to the well-known point kinetics equation. However, here this condition is not imposed and the unconstrained version of the quasistatic formulation is retained.

In a further attempt to save some computational effort, equation 2.2 is integrated analytically from the previous time step t_{n-1} to time t. The result is a convolution which can be written as:

$$\vec{\chi}_i C_i(\hat{r},t) = \beta_i \int_{t_{n-1}}^{t} \vec{\chi}_i \vec{F}^T(\hat{r},\tau)\vec{\phi}(\hat{r},\tau)e^{(-\lambda_i(t-\tau))}d\tau + \vec{\chi}_i C_i(\hat{r},t_{n-1})e^{(-\lambda_i(t-t_{n-1}))} \tag{2.22}$$

2.4 Reduced Two-Energy-Group Formulation of Quasistatic Equations

For thermal reactors two-energy-group discretization is usually sufficient to model spectral effects. Thus, given a two-energy-group formulation (G=2), the matrices and vectors of equations 2.1 and 2.2 in a general form are:

$$\vec{\chi}_p = \begin{pmatrix} \chi_{p1} \\ \chi_{p2} \end{pmatrix} \tag{2.23}$$

$$\vec{\chi}_i = \begin{pmatrix} \chi_{i1} \\ \chi_{i2} \end{pmatrix} \tag{2.24}$$

$$\vec{F}(\hat{r},t) = \begin{pmatrix} \nu\Sigma_{f_1}(\hat{r},t) \\ \nu\Sigma_{f_2}(\hat{r},t) \end{pmatrix} \tag{2.25}$$

$$\mathbf{S}(\hat{r},t) = \begin{pmatrix} \Sigma_{r1}(\hat{r},t) & -\Sigma_{s12}(\hat{r},t) \\ -\Sigma_{s21}(\hat{r},t) & \Sigma_{r2}(\hat{r},t) \end{pmatrix} \tag{2.26}$$

$$\mathbf{D}(\hat{r},t) = \begin{pmatrix} D_1(\hat{r},t) & 0 \\ 0 & D_2(\hat{r},t) \end{pmatrix} \tag{2.27}$$

$$\mathbf{V}^{-1} = \begin{pmatrix} \frac{1}{V_1} & 0 \\ 0 & \frac{1}{V_2} \end{pmatrix} \tag{2.28}$$

To facilitate the development of a computer code and still solve realistic problems the following assumptions are made:

1. No neutron may be accelerated from the thermal group (group 2) to the fast group (group 1);

$$\Sigma_{s12}(\hat{r},t) = 0 \tag{2.29}$$

2. All secondary neutrons originate in the fast group (number 1);

$$\vec{\chi}_p = \vec{\chi}_i = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \vec{\chi} \tag{2.30}$$

3. No thermal or xenon feedback is present.

In order to express the reduced two-energy-group formulation in a concise fashion, the matrices **A** and **B** are defined as:

$$\mathbf{A}(\hat{r},t) = \nabla \cdot \mathbf{D}(\hat{r},t)\nabla - \mathbf{S}(\hat{r},t) \tag{2.31}$$

$$\mathbf{B}(\hat{r},t) = \vec{\chi}\vec{F}^T(\hat{r},t) \tag{2.32}$$

The quasistatic equations can now be rewritten in the following form. The reduced two-energy-group point kinetics parameters are:

$$\frac{\rho}{\Lambda}(t) = \frac{\left\langle \vec{W}(\hat{r}) \ , \ [\mathbf{A}(\hat{r},t) + \mathbf{B}(\hat{r},t)]\,\vec{\varphi}(\hat{r},t) \right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t) \right\rangle} \tag{2.33}$$

$$\frac{\beta_i}{\Lambda}(t) = \frac{\left\langle \vec{W}(\hat{r}) \ , \ \beta_i \mathbf{B}(\hat{r},t)\vec{\varphi}(\hat{r},t) \right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t) \right\rangle} \tag{2.34}$$

$$\frac{\beta}{\Lambda}(t) = \sum_{i=1}^{N} \frac{\beta_i}{\Lambda}(t) \tag{2.35}$$

$$\xi_i(t) = \frac{\left\langle \vec{W}(\hat{r}) \ , \ \vec{\chi}C_i(t) \right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t) \right\rangle} \tag{2.36}$$

$$\lambda_s(t) = \frac{\frac{d}{dt}\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t) \right\rangle}{\left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t) \right\rangle} \tag{2.37}$$

The point kinetics equations are:

$$\frac{d}{dt}T(t) = \left(\frac{\rho}{\Lambda}(t) - \frac{\beta}{\Lambda}(t) - \lambda_s(t)\right) T(t) + \sum_{i=1}^{N} \lambda_i \xi_i(t) \tag{2.38}$$

$$\frac{d}{dt}\xi_i(t) = \frac{\beta_i}{\Lambda}(t)T(t) - (\lambda_i + \lambda_s(t))\xi_i(t) \tag{2.39}$$

The reduced two-energy-group shape function equation is:

$$\begin{aligned} \mathbf{V}^{-1}\frac{\partial}{\partial t}\vec{\varphi}(\hat{r},t) + \mathbf{V}^{-1}\frac{1}{T(t)}\frac{d}{dt}T(t)\vec{\varphi}(\hat{r},t) &= \\ [\mathbf{A}(\hat{r},t) + (1-\beta)\mathbf{B}(\hat{r},t)]\,\vec{\varphi}(\hat{r},t) &+ \frac{1}{T(t)}\sum_{i=1}^{N} \lambda_i \vec{\chi}C_i(\hat{r},t) \end{aligned} \tag{2.40}$$

The reduced two-energy-group precursor equation is:

$$\begin{aligned} \vec{\chi}C_i(\hat{r},t) &= \beta_i \int_{t_{n-1}}^{t} \mathbf{B}(\hat{r},\tau)\vec{\phi}(\hat{r},\tau)e^{(-\lambda_i(t-\tau))}d\tau \\ &+ \vec{\chi}C_i(\hat{r},t_{n-1})e^{(-\lambda_i(t-t_{n-1}))} \end{aligned} \tag{2.41}$$

2.5 Choice of a Weight Function

Almost any choice for the weight function is permissible. We previously specified a $\vec{W}(\hat{r})$ independent of t, although theoretically a time-dependent weight function is permissible. Most methods, including this one, use a stationary value for $\vec{W}(\hat{r})$ corresponding to the initial adjoint flux. It can be shown that using the initial adjoint flux as a weight function reduces the error in the calculation of $\rho/\Lambda(t)$. Since the results are very sensitive to changes in $\rho/\Lambda(t)$, minimizing the error in the computation of this term is important.

The first step is to rewrite equation 2.33 as:

$$\frac{\rho}{\Lambda}(t) = \frac{1}{E(t)} \left\langle \vec{W}(\hat{r}), \mathbf{L}(\hat{r},t)\vec{\varphi}(\hat{r},t) \right\rangle \tag{2.42}$$

where:

$$\mathbf{L}(\hat{r},t) = \mathbf{A}(\hat{r},t) + \mathbf{B}(\hat{r},t)$$

$$E(t) = \left\langle \vec{W}(\hat{r}), \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t) \right\rangle$$

Closer examination of $\mathbf{L}(\hat{r},t)$ reveals that at t_0 it is the steady-state diffusion operator, thus for an initially critical reactor, we have

$$\mathbf{L}(\hat{r},t_0)\vec{\varphi}(\hat{r},t_0) = 0 \tag{2.43}$$

and consequently

$$\frac{\rho}{\Lambda}(t_0) = \frac{1}{E(t_0)} \left\langle \vec{W}(\hat{r}), \mathbf{L}(\hat{r},t_0)\vec{\varphi}(\hat{r},t_0) \right\rangle = 0 \tag{2.44}$$

The temporal dependence of both $\mathbf{L}$ and $\vec{\varphi}$ can be expressed as:

$$\mathbf{L}(\hat{r},t) = \mathbf{L}(\hat{r},t_0) + \delta\mathbf{L}(\hat{r},t) \tag{2.45}$$

and

$$\vec{\varphi}(\hat{r},t) = \vec{\varphi}(\hat{r},t_0) + \delta\vec{\varphi}(\hat{r},t) \tag{2.46}$$

Substituting equation 2.46 and 2.45 into 2.42 and multiplying out we obtain:

$$\begin{aligned}\frac{\rho}{\Lambda}(t) = \Big\{ \big\langle \vec{W}(\hat{r}), \mathbf{L}(\hat{r},t_0)\vec{\varphi}(\hat{r},t_0)\big\rangle \; + \; \big\langle \vec{W}(\hat{r}), \delta\mathbf{L}(\hat{r},t)\vec{\varphi}(\hat{r},t_0)\big\rangle + \\ \big\langle \vec{W}(\hat{r}), \mathbf{L}(\hat{r},t_0)\delta\vec{\varphi}(\hat{r},t)\big\rangle \; + \; \big\langle \vec{W}(\hat{r}), \delta\mathbf{L}(\hat{r},t)\delta\vec{\varphi}(\hat{r},t)\big\rangle \Big\} \times \\ 1 \; \Big/ \{E(t_0) + \delta E(t)\}\end{aligned} \tag{2.47}$$

If the changes in $\mathbf{L}$ are small, the second-order term $\delta\mathbf{L}\delta\vec{\varphi}$ is negligible. Since the initial shape function satisfies the initial flux equation 2.43, the first term on the right hand side is zero. Now only first-order perturbation terms remain in the numerator. When the perturbation term $\delta E(t)$ is brought from the denominator to the numerator using a truncated Taylor series, only second-order terms result. Therefore that term is dropped. Only two terms remain and the equation reduces to:

$$\frac{\rho}{\Lambda}(t) = \frac{1}{E(t_0)}\big\langle \vec{W}(\hat{r}), \delta\mathbf{L}(\hat{r},t)\vec{\varphi}(\hat{r},t_0)\big\rangle + \frac{1}{E(t_0)}\big\langle \vec{W}(\hat{r}), \mathbf{L}(\hat{r},t_0)\delta\vec{\varphi}(\hat{r},t)\big\rangle \tag{2.48}$$

Here it can easily be seen that any variation in $\vec{\varphi}(\hat{r},t)$ produces a first-order variation in ρ/Λ. Fortunately we can eliminate this dependence by choosing the adjoint flux $\vec{\varphi}^*$ as the weight function $\vec{W}$. The adjoint flux is defined as:

$$\mathbf{L}^*(\hat{r},t_0)\vec{\varphi}^*(\hat{r},t_0) = 0 \tag{2.49}$$

where the adjoint operator $\mathbf{L}^*$ is defined such that:

$$\langle \vec{u}(\hat{r}), \mathbf{L}^*(\hat{r},t_0)\vec{v}(\hat{r})\rangle = \langle \vec{v}(\hat{r}), \mathbf{L}(\hat{r},t_0)\vec{u}(\hat{r})\rangle \tag{2.50}$$

where $\vec{u}(\hat{r})$ and $\vec{v}(\hat{r})$ are arbitrary functions

Applying 2.50 to the second term in 2.48 we obtain:

$$\frac{\rho}{\Lambda}(t) = \frac{1}{E(t_0)}\left\langle \vec{W}(\hat{r}), \delta\mathbf{L}(\hat{r},t)\vec{\varphi}(\hat{r},t_0)\right\rangle + \frac{1}{E(t_0)}\left\langle \delta\vec{\varphi}(\hat{r},t_0), \mathbf{L}^*(\hat{r},t_0)\vec{W}(\hat{r})\right\rangle \quad (2.51)$$

If the adjoint flux is used as the weight function (ie. $\vec{W}(\hat{r}) = \vec{\varphi}^*(\hat{r},t_0)$ the second term in 2.51 vanishes and we obtain:

$$\frac{\rho}{\Lambda}(t) = \frac{1}{E(t_0)}\left\langle \vec{\varphi}^*(\hat{r}), \delta\mathbf{L}(\hat{r},t)\vec{\varphi}(\hat{r},t_0)\right\rangle \quad (2.52)$$

"Thus, by introducing $\vec{\varphi}^*(\hat{r})$ for the weight function, we have found an approximate method of computing reactivity which does not depend on knowing $\delta\vec{\varphi}(\hat{r},t)$, but in which, nevertheless, we can be certain that the terms neglected are of "lower order" (ie. contain a higher-order multiple of a perturbed quantity) than those retained."[12]

2.6 Discretization of Shape Function Equation

The time-dependent neutron diffusion equation has been transformed by the quasistatic method into a more practical set of equations. Before these equations can be solved they must first be discretized. A mixed method is used to perform this discretization.

The discretization of the spatial variable $\hat{r}$ was performed using the nodal collocation method. A very good description of this technique is given by A.Hébert[13]. Since the spatial discretization method is of little interest in the quasistatic method, only the discretization of the temporal variable will be discussed in detail. The following change in notation will be used to distinguish the continuous and discretized spatial variables and operators:

$$\text{continuous} \Rightarrow \text{discrete} \quad (2.53)$$

$$\mathbf{A}(\hat{r},t)\vec{\varphi}(\hat{r},t) \Rightarrow \mathbf{A}(t)\vec{\varphi}(t) \quad (2.54)$$

Assuming that there are G energy-groups and K^3 mesh points the discretization implies that

$$(G \times G)(G \times 1) \Rightarrow (GK^3 \times GK^3)(GK^3 \times 1) \tag{2.55}$$

Discretization of the unity matrix (or operator) **I** using the nodal collocation or finite-element method gives a unitary matrix **U** which is not necessarily equal to the identity matrix:

$$\mathbf{I}\vec{\varphi}(\hat{r},t) \Rightarrow \mathbf{U}\vec{\varphi}(t) = \begin{pmatrix} \mathbf{u} & 0 \\ 0 & \mathbf{u} \end{pmatrix} \begin{pmatrix} \vec{\varphi}_1(t) \\ \vec{\varphi}_2(t) \end{pmatrix} \tag{2.56}$$

The sub-blocks **u** of the unitary matrix are diagonal matrices and hence the unitary matrix **U** is also diagonal.

Since the discretized unitary matrix is not the identity matrix, the discretization of the inverse velocity matrix is represented as:

$$\mathbf{V}^{-1}\vec{\varphi}(\hat{r},t) \Rightarrow \begin{pmatrix} \frac{1}{V_1} & 0 \\ 0 & \frac{1}{V_2} \end{pmatrix} \mathbf{U}\vec{\varphi}(t) = \mathbf{V}^{-1}\mathbf{U}\vec{\varphi}(t) \tag{2.57}$$

Discretization of the precursor term $C_i(r,t)$ is more complicated. Since $C_i(r,t)$ always appears as a product with $\vec{\chi}$, the discretized form can be written as follows:

$$\vec{\chi}C_i(\hat{r},t) \Rightarrow \boldsymbol{\chi}\vec{C}_i(t) \tag{2.58}$$

$$(G \times 1)(1 \times 1) \Rightarrow (GK^3 \times K^3)(K^3 \times 1) \tag{2.59}$$

where $\boldsymbol{\chi}$ is

$$\boldsymbol{\chi} = \begin{pmatrix} 1\mathbf{u} \\ 0\mathbf{u} \end{pmatrix} = \begin{pmatrix} \mathbf{u} \\ \mathbf{0} \end{pmatrix} \tag{2.60}$$

Note that $\vec{C}_i(t)$ has only K^3 unknowns whereas the flux contains GK^3 unknowns.

The temporal variable is discretized by a finite-element like approach. The shape function $\vec{\varphi}(t)$ is assumed to have a linear variation over the interval as:

$$\vec{\varphi}(t) = f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1}) \tag{2.61}$$

$$f = \frac{t - t_{n-1}}{t_n - t_{n-1}} \tag{2.62}$$

At the start of any time step the initial shape function is known, and the end-of-time-step shape function is the unknown. The code marches forward by applying the above backward difference relationship over each time interval. A more mathematically rigorous derivation can be obtained by using a point collocation finite-element technique.

For the unconstrained version of the quasistatic method, multiple solutions $\varphi^{\dagger}(t)T^{\dagger}(t)$ of the form:

$$\vec{\varphi}(t)T(t) = \left[\frac{\vec{\varphi}(t)}{c(t)}\right][T(t)c(t)] = \vec{\varphi}^{\dagger}(t)T^{\dagger}(t) \tag{2.63}$$

must also be admitted. However, here a linear variation with time is imposed on the shape function. Therefore by construction the possibility of multiple solutions is eliminated.

For a ramp perturbation of the nuclear cross sections, the dependence of **A** and **B** with time are:

$$\mathbf{A}(t) = f\mathbf{A}(t_n) + (1-f)\mathbf{A}(t_{n-1}) \tag{2.64}$$

$$\mathbf{B}(t) = f\mathbf{B}(t_n) + (1-f)\mathbf{B}(t_{n-1}) \tag{2.65}$$

Similarily, for a step perturbation of the nuclear cross sections, the dependence of **A** and **B** with time are:

$$\mathbf{A}(t) = \mathbf{A}(t_n) \quad \text{for} \quad t_{n-1} \leq t \leq t_n \tag{2.66}$$

$$\mathbf{B}(t) = \mathbf{B}(t_n) \quad \text{for} \quad t_{n-1} \leq t \leq t_n \tag{2.67}$$

The discretized shape function equation, obtained by substituting equation 2.61 into equation 2.40 and evaluating it at t_n, is:

$$\begin{aligned} \mathbf{V}^{-1}\mathbf{U}\frac{\vec{\varphi}(t_n) - \vec{\varphi}(t_{n-1})}{\Delta t} + \mathbf{V}^{-1}\frac{1}{T(t_n)}\frac{d}{dt}T(t_n)\mathbf{U}\vec{\varphi}(t_n) &= \\ [\mathbf{A}(t_n) + (1-\beta)\mathbf{B}(t_n)]\,\vec{\varphi}(t_n) &+ \frac{1}{T(t_n)}\sum_{i=1}^{N}\chi\lambda_i\vec{C}_i(t_n) \end{aligned} \tag{2.68}$$

2.7 Discretization of Point Kinetics Equation

Before discussing the discretization of the amplitude equations, we shall first discretize the amplitude or point kinetics parameters. Equations 2.33 through 2.37 are discretized using the rules defined in the previous section. Starting from these discretized equations, the point kinetics parameters for a ramp perturbation are obtained by substituting equations 2.61 , 2.64 and 2.65 into the discretized version of equations 2.33 through 2.37, giving:

$$\frac{\rho}{\Lambda}(t) = \frac{1}{\vec{W}^t\mathbf{V}^{-1}\mathbf{U}\left[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})\right]} \times \tag{2.69}$$
$$\vec{W}^t\left[f(\mathbf{A}(t_n) + \mathbf{B}(t_n)) + (1-f)(\mathbf{A}(t_{n-1}) + \mathbf{B}(t_{n-1})\right]\left[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})\right]$$

$$\frac{\beta_i}{\Lambda}(t) = \beta_i\frac{\vec{W}^t\left[f\mathbf{B}(t_n) + (1-f)\mathbf{B}(t_{n-1})\right]\left[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})\right]}{\vec{W}^t\mathbf{V}^{-1}\mathbf{U}\left[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})\right]} \tag{2.70}$$

$$\frac{\beta}{\Lambda}(t) = \sum_{i=1}^{N}\frac{\beta_i}{\Lambda}(t) \tag{2.71}$$

$$\xi_i(t) = \frac{\vec{W}^t\,\chi\vec{C}_i(t)}{\vec{W}^t\mathbf{V}^{-1}\mathbf{U}\left[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})\right]} \tag{2.72}$$

$$\lambda_s(t) = \frac{1}{\Delta t}\frac{\vec{W}^t\mathbf{V}^{-1}\mathbf{U}\left[\vec{\varphi}(t_n) - \vec{\varphi}(t_{n-1})\right]}{\vec{W}^t\mathbf{V}^{-1}\mathbf{U}\left[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})\right]} \tag{2.73}$$

Similarly for step perturbations the discretized point kinetics parameters are obtained by substituting equation 2.61 , 2.66 and 2.67 into equation 2.33 through 2.37, giving:

$$\frac{\rho}{\Lambda}(t) = \frac{\vec{W}^t\left[\mathbf{A}(t_n) + \mathbf{B}(t_n)\right]\left[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})\right]}{\vec{W}^t\mathbf{V}^{-1}\mathbf{U}\left[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})\right]} \tag{2.74}$$

$$\frac{\beta_i}{\Lambda}(t) = \beta_i\frac{\vec{W}^t\mathbf{B}(t_n)\left[\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})\right]}{\vec{W}^t\mathbf{V}^{-1}\mathbf{U}\left[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})\right]} \tag{2.75}$$

$$\frac{\beta}{\Lambda}(t) = \sum_{i=1}^{N}\frac{\beta_i}{\Lambda}(t) \tag{2.76}$$

$$\xi_i(t) = \frac{\vec{W}^t \, \chi\vec{C}_i(t)}{\vec{W}^t\mathbf{V}^{-1}\mathbf{U}\,[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})]} \tag{2.77}$$

$$\lambda_s(t) = \frac{1}{\Delta t}\frac{\vec{W}^t\mathbf{V}^{-1}\mathbf{U}\,[\vec{\varphi}(t_n) - \vec{\varphi}(t_{n-1})]}{\vec{W}^t\mathbf{V}^{-1}\mathbf{U}\,[f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1})]} \tag{2.78}$$

The point kinetics or amplitude equation is independent of the spatial variable. Therefore, as stated earlier, a finite-difference method is used to discretize the temporal variable. Unfortunately this system of equations possesses a stiff character. Thus the scheme used is Gear's algorithm.[14]

2.8 Discretization of Precursor Equation

Using the technique applied to the shape function equation, the discretized precursor equation can be written as:

$$\chi\vec{C}_i(t_n) = \beta_i \int_{t_{n-1}}^{t_n} \mathbf{B}(\tau)\vec{\phi}(\tau)e^{(-\lambda_i(t_n-\tau))}d\tau + \chi\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n-t_{n-1}))} \tag{2.79}$$

For ramp perturbations, this equation is expanded by using equations 2.61 and 2.65, giving:

$$\begin{aligned}\chi\vec{C}_i(t_n) &= \beta_i \int_{t_{n-1}}^{t_n} (f\mathbf{B}(t_n) + (1-f)\mathbf{B}(t_{n-1}))(f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1}))e^{(-\lambda_i(t-\tau))}d\tau \\ &+ \chi\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n-t_{n-1}))}\end{aligned} \tag{2.80}$$

Multiplying out the product in the integral and performing some manipulation, the following equation is obtained:

$$\begin{aligned}\chi\vec{C}_i(t_n) &= \beta_i(\gamma_{i,3}\mathbf{B}(t_n)\vec{\varphi}(t_n) + \gamma_{i,2}\mathbf{B}(t_n)\vec{\varphi}(t_{n-1}) + \\ &\quad \gamma_{i,2}\mathbf{B}(t_{n-1})\vec{\varphi}(t_n) + \gamma_{i,1}\mathbf{B}(t_{n-1})\vec{\varphi}(t_{n-1})) + \\ &\quad \chi\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n-t_{n-1}))}\end{aligned} \tag{2.81}$$

where

$$\gamma_{i,3} = \int_{t_{n-1}}^{t_n} f(\tau)f(\tau)T(\tau)e^{-\lambda_i(t_n-\tau)}d\tau \tag{2.82}$$

$$\gamma_{i,2} = \int_{t_{n-1}}^{t_n} f(\tau)[1-f(\tau)]T(\tau)e^{-\lambda_i(t_n-\tau)}d\tau \tag{2.83}$$

$$\gamma_{i,1} = \int_{t_{n-1}}^{t_n} [1-f(\tau)][1-f(\tau)]T(\tau)e^{-\lambda_i(t_n-\tau)}d\tau \tag{2.84}$$

Performing the same calculation for a step perturbation, by substituting equations 2.61 and 2.67 into 2.79 gives again equation 2.81, where the γ's are now defined as:

$$\gamma_{i,3} = \int_{t_{n-1}}^{t_n} f(\tau)T(\tau)e^{-\lambda_i(t_n-\tau)}d\tau \tag{2.85}$$

$$\gamma_{i,2} = 0 \tag{2.86}$$

$$\gamma_{i,1} = \int_{t_{n-1}}^{t_n} [1-f(\tau)]T(\tau)e^{-\lambda_i(t_n-\tau)}d\tau \tag{2.87}$$

2.9 Quasistatic Solution Strategy

The shape function and amplitude function equations form a system of nonlinear equations. The relationship between the equations can be written concisely as:

$$\vec{\varphi}(\hat{r},t) = g(T(t)) \tag{2.88}$$

$$T(t) = h(\vec{\varphi}(\hat{r},t)) \tag{2.89}$$

A fixed-point iterative technique is used to solve these nonlinear equations. Thus an attempt is made to find a fixed vector $\vec{\varphi}^{(*)}(\hat{r},t)$ such that

$$\vec{\varphi}^{(*)}(\hat{r},t) = g(h(\vec{\varphi}^{(*)}(\hat{r},t))) \tag{2.90}$$

This solution scheme consists of iterating on the shape function using

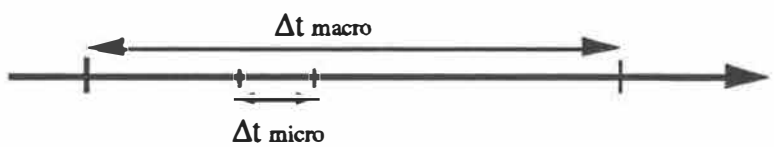
$$\vec{\varphi}^{(l+1)}(\hat{r},t) = g(h(\vec{\varphi}^{(l)}(\hat{r},t))) \tag{2.91}$$

until the errors between two consecutive iterations is smaller than a given convergence criteria. This scheme is not necessarily convergent. An attempt was made

to predict if the iterative scheme is convergent by computing the gradient of the function near the solution. However little insight was gained from this analysis.

In order to solve these equations simultaneously over different time steps, two integration time steps are used and are graphically represented as:

Figure 2.1: Time Step Hierarchy



The shape function is computed over macro time steps (Δt). At several and not necessarily equal time steps within the macro time step the point kinetics equations are solved. The size of the micro time step is controlled by the point kinetics solution algorithm. A micro time step may not cross a macro time step boundary. Thus the basic time unit is the macro time step upon which the quasistatic algorithm is based.

Given this time step hierarchy, the quasistatic algorithm can be easily illustrated. An iterative scheme over the macro time step is used due to the implicit nature of the coupled quasistatic equations. The iterative sequence is terminated when the quasistatic solution is self consistent over the macro time step. No guarantee of accuracy is possible but this implicit scheme is generally stable.

If the shape-function solution step is skipped, the shape function used throughout the transient is the steady-state flux and the IQS method reduces to point kinetics. A special keyword (PKIN) tells the program to execute without updating the shape function. It is convenient for academic exercises and for verifying the point kinetics aspects of the code.

The XSTATIC algorithm is expressed in flow-chart form in Figure 2.2. The functions of the major subroutines that make up the XSTATIC code are: PHASE4,

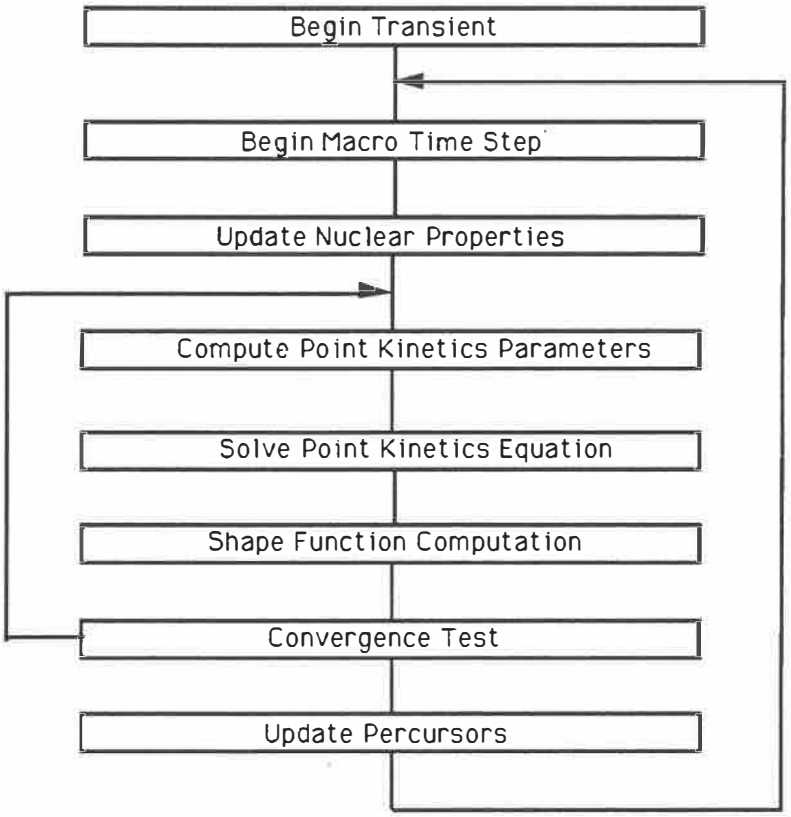


Figure 2.2: XSTATIC QUASISTATIC ALGORITHM

FLXQSM, FASTQ and PRECDY. These subroutines are discussed in the following sections.

2.9.1 Update Nuclear Properties - PHASE4

The module that updates the nuclear properties and the matrices which depend on them is PHASE4. User-supplied changes in nuclear properties are read in and passed to MATUPD. This TRIVAC subroutine assembles only the perturbed part of the matrices. The perturbed terms are then added to the original, unperturbed matrices to obtain the new matrices. Mathematically the updating of any matrix X is performed as:

$$\mathbf{X}(t_n) = \mathbf{X}(t_{n-1}) + \mathbf{\Delta X}(t_n) \tag{2.92}$$

2.9.2 Shape Function Algorithm - FLXQSM

The module FLXQSM solves the shape function equation. The shape function equation expressed in equation 2.68 is not solved directly, and some mathematical manipulations are still necessary. The term involving $\vec{C}_i(t)$ is an implicit function of $\varphi(t)$. Equation 2.81 is substituted in 2.68 to render this term explicit, leading to:

$$\begin{aligned}\mathbf{V}^{-1}\mathbf{U}\frac{\vec{\varphi}(t_n)-\vec{\varphi}(t_{n-1})}{\Delta t} + \frac{1}{T(t_n)}\frac{d}{dt}T(t_n)\mathbf{U}\vec{\varphi}(t_n) &= [\mathbf{A}(t_n)+(1-\beta)\mathbf{B}(t_n)]\,\vec{\varphi}(t_n)\\ +\frac{1}{T(t_n)}\sum_{i=1}^{N}\lambda_i\beta_i\{\gamma_{i,3}\mathbf{B}(t_n)\vec{\varphi}(t_n) &+ \gamma_{i,2}\mathbf{B}(t_n)\vec{\varphi}(t_{n-1}) + \gamma_{i,2}\mathbf{B}(t_{n-1})\vec{\varphi}(t_n) +\\ \gamma_{i,1}\mathbf{B}(t_{n-1})\vec{\varphi}(t_{n-1})\} &+ \frac{1}{T(t_n)}\,\chi\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n-t_{n-1})}\end{aligned} \tag{2.93}$$

Rewriting this equation with the $\vec{\varphi}(t_{n-1})$ and $\vec{C}_i(t_{n-1})$ terms on the right hand side yields:

$$\mathbf{H}(t_n)\vec{\varphi}^{(k)}(t_n) = \vec{S} \tag{2.94}$$

where

$$\begin{aligned}
\mathbf{H}(t_n) &= \Big[\mathbf{A}(t_n) + (1-\beta+\frac{\gamma_3}{T(t_n)})\mathbf{B}(t_n) + \frac{\gamma_2}{T(t_n)}\mathbf{B}(t_{n-1}) \\
&\quad -\mathbf{V}^{-1}\mathbf{U}\left\{\frac{1}{\Delta t} + \frac{1}{T(t_n)}\frac{d}{dt}T(t_n)\right\}\Big] && (2.95)\\
\vec{S} &= [\mathbf{V}^{-1}\mathbf{U}\frac{1}{\Delta t} + \frac{\gamma_1}{T(t_n)}\mathbf{B}(t_{n-1}) + \frac{\gamma_2}{T(t_n)}\mathbf{B}(t_n)]\vec{\varphi}(t_{n-1}) \\
&+ \frac{1}{T(t_n)}\,\chi\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n-t_{n-1}))} && (2.96)\\
\gamma_J &= \sum_{i=1}^{N}\lambda_i\beta_i\gamma_{i,J} && (2.97)
\end{aligned}$$

The above implicit shape function equation can be described as a fixed source problem. It is solved using a preconditioned iterative strategy and variational acceleration.

2.9.2.1 Iterative Strategy

The iterative strategy can be expressed as follows

$$\vec{g}^{(k)} = -\mathbf{M}\quad[\mathbf{H}\vec{\varphi}^{(k)} - \vec{S}^{(k)}] \tag{2.98}$$

where $\mathbf{M}$ is the preconditioning matrix.

The matrix $\mathbf{M}$ is arbitrary and it can be shown that the iterative sequence converges if the spectral radius of the residual matrix

$$\mathbf{R} = \mathbf{I} - \mathbf{MH} \tag{2.99}$$

is less than 1.

In order to take advantage of properties of the matrices $\mathbf{A}$ and $\mathbf{B}$, the preconditioning matrix $\mathbf{M}$ is constructed to resemble the inverse of $\mathbf{A}$. This choice of the preconditioning matrix is the same as for the static case [10]. Recall that the form of $\mathbf{A}$ is lower triangular, that is:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A_{11}} & 0 \\ \mathbf{A_{21}} & \mathbf{A_{22}} \end{pmatrix}. \tag{2.100}$$

Since the diagonal sub-blocks $\{\mathbf{A}_{gg}; g = 1, 2\}$ are symmetric positive definite, and diagonally dominant, the matrix $\mathbf{A}$ is therefore invertible :

$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{A}_{11}^{-1} & 0 \\ -\mathbf{A}_{22}^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{A}_{22}^{-1} \end{pmatrix}. \tag{2.101}$$

The matrix $\mathbf{M}$ is chosen to have the form:

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_{11}^{-1} & 0 \\ -\mathbf{M}_{22}^{-1}\mathbf{A}_{21}\mathbf{M}_{11}^{-1} & \mathbf{M}_{22}^{-1} \end{pmatrix}. \tag{2.102}$$

If $\mathbf{M}_{gg} = \mathbf{A}_{gg}^{-1}$ then the preconditioning matrix $\mathbf{M} = \mathbf{A}^{-1}$. However storage efficiency considerations forbid this. Instead an alternating direction implicit (ADI) procedure is used to approximate $\mathbf{A}_{gg}^{-1}$.[10]

2.9.2.2 Acceleration

The shape function solution algorithm is accelerated using a variational acceleration technique.[10] This technique is based on the introduction of two extrapolation factors, such that:

$$\vec{\varphi}^{(k+1)} = \vec{\varphi}^{(k)} + \alpha^{(k)} \left\{ \vec{g}^{(k)} + \beta^{(k)} \left[\vec{g}^{(k)} - \vec{g}^{(k-1)} \right] \right\} \tag{2.103}$$

Define the following functional F whose associated Euler equation is the shape function equation:

$$F\{\vec{\varphi}\} = \frac{1}{2} \left\langle \vec{\varphi}, \mathbf{H}^T\mathbf{H}\vec{\varphi} \right\rangle - \left\langle \vec{\varphi}, \mathbf{H}^T\vec{S} \right\rangle \tag{2.104}$$

The two extrapolation factors are chosen to make the functional stationary at the $(k+1)^{th}$ iteration.

$$\frac{dF\{\vec{\varphi}^{(k+1)}\}}{d\alpha^{(k+1)}} = 0 \tag{2.105}$$

$$\frac{dF\{\vec{\varphi}^{(k+1)}\}}{d\beta^{(k+1)}} = 0 \tag{2.106}$$

Solving these nonlinear systems gives $\alpha^{(k)}$, $\beta^{(k)}$. The acceleration is normally applied cyclically: 3 free iterations followed by 3 accelerated iterations.[10]

2.9.2.3 Convergence Test

The following convergence criterion is needed to ensure the proper convergence of the iterative algorithm:

$$\frac{\max_i \left|\varphi_i^{(k+1)}(t_n) - \varphi_i^{(k)}(t_n)\right|}{\max_i \left|\varphi_i^{(k+1)}(t_n)\right|} < \varepsilon_{\text{shape}} \tag{2.107}$$

Typically $\varepsilon_{\text{shape}}$ is between 10^{-4} and 0.5×10^{-5}. The iterative cycle is however terminated if the number of iterations (k) exceeds a user-given value (MMAX).

2.9.2.4 Application of the Constraint

For the constrained version of the IQS method, the constraint is imposed by multiplying the converged shape function by a constant c defined as:

$$c = \frac{\vec{W}^t \mathbf{V}^{-1} \mathbf{U} \vec{\varphi}(t_{n-1})}{\vec{W}^t \mathbf{V}^{-1} \mathbf{U} \vec{\varphi}(t_n)} \tag{2.108}$$

No further action anywhere else in the code is required, because the code assumes that the constraint is not imposed.

2.9.3 Point Kinetics Solution - FASTQ

The module FASTQ solves the point kinetics equations. The point kinetics are typically defined as in equations 2.38 and 2.39. These equations may be expressed in matrix form as:

$$\frac{d}{dt}\vec{Z} = \mathbf{P}(t)\vec{Z} \tag{2.109}$$

$$\vec{Z} = \begin{pmatrix} T \\ \xi_1 \\ \vdots \\ \xi_N \end{pmatrix} \tag{2.110}$$

$$\mathbf{P}(t) = \begin{pmatrix} \frac{\rho}{\Lambda} - \frac{\beta}{\Lambda} - \lambda_s & \lambda_1 & \dots & \lambda_N \\ \frac{\beta}{\Lambda}_1 & -(\lambda_1 + \lambda_s) & \dots & \\ \vdots & & \ddots & \vdots \\ \frac{\beta}{\Lambda}_N & & \dots & -(\lambda_N + \lambda_s) \end{pmatrix} \tag{2.111}$$

However in the quasistatic method, the γ_i terms, defined in equation 2.82 to 2.84, must be solved along with the point kinetics equation. This combined set or "augmented" point kinetics equation can be written in matrix form:

$$\frac{d}{dt}\vec{Z} = \mathbf{Q}(t)\vec{Z} \tag{2.112}$$

$$\vec{Z} = \begin{pmatrix} T \\ \xi_1 \\ \vdots \\ \xi_N \\ \gamma_{1,1} \\ \vdots \\ \gamma_{N,3} \end{pmatrix} \tag{2.113}$$

$$\mathbf{Q}(t) = \begin{pmatrix} \frac{\rho}{\Lambda} - \frac{\beta}{\Lambda} - \lambda_s & \lambda_1 & \dots & \lambda_N & \\ \frac{\beta}{\Lambda}_1 & -(\lambda_1 + \lambda_s) & \dots & & 0 \\ \vdots & & \ddots & \vdots & \\ \frac{\beta}{\Lambda}_N & & \dots & -(\lambda_N + \lambda_s) & \\ f^2 e^{-\lambda_1 \tau} & & & & \\ \vdots & & & & 0 \\ (1-f)^2 e^{-\lambda_N \tau} & & & & \end{pmatrix} \tag{2.114}$$

The introduction of the term "augmented" should remove some of the ambiguity associated with the term point kinetics.

XSTATIC uses the following two methods to solve the augmented point kinetics equations:

Method 1: DGEAR

The augmented point kinetics problem is solved by a brute force method using an International Mathematics Subroutine Library (IMSL) integration routine for stiff systems called DGEAR. This routine uses Gear's method to solve the amplitude equations. The amount of work involved in solving this system is quite large, making the solution of the point kinetics ordinary differential equation (ODE) system quite expensive due to the evaluation of many costly exponentials. The dimensions of the regular and augmented point kinetics system for NDG delayed precursor groups are NDG+1 and 4*NDG+1 respectively. Therefore for a 6-precursor-group case a 25X25 augmented system must be solved, instead of a 7X7 system.

Method 2: FASTQ

To avoid these problems, the following two-step procedure is proposed:

1. integrate the regular point kinetics equation using Gear's method over the interval, and store the result from each of the time step in an array of the form

$$(t_i, T_i) \quad i = 1, N \quad , \tag{2.115}$$

2. then separately integrate the 3*NDG $\gamma_{i,j}$ equations over the interval

Step 1 requires the integration of equation 2.109. The Gear's integration subroutine DGEAR was used to solve the point kinetics ODE but any set of canned routines (IMSL, ESSL, ...) can be used. The purpose here is not to rewrite highly perfected ODE solvers, but to break the augmented system into quickly solved sub-problems.

Step 2 requires the integration of $\gamma_{i,j}$'s and poses a rather special problem for which canned routines do not exist. The starting point is a set of solutions (t_i, T_i) that are generated by a polynomial integration method like Gear's. The polynomial approximation to the exact solution, which is generated by the integration method, can be reconstructed and integrated analytically. However this approach is time consuming. Alternately an accurate result for $\gamma_{i,j}$ can be obtained by approximating the polynomial solution by a polynomial of lower order and integrating analytically. This algorithm is presented without any further justification other than its success.

First a quadratic equation is fit to groups of three successive points, (t_{2j-1}, T_{2j-1}) through (t_{2j+1}, T_{2j+1}). This yields a piecewise quadratic polynomial representation of the amplitude. That is, the amplitude $T(t)$ can be expressed as:

$$T(t) = \sum_{j=1}^{J} \left[(a_j^{(2)} t^2 + a_j^{(1)} t^1 + a_j^{(0)}) \times H_j(t) \right] \tag{2.116}$$

$$H_j(t) = \begin{cases} 1 & \text{if} \quad t \in [t_{2j+1}, t_{2j-1}] \\ 0 & \text{elsewhere} \end{cases} \tag{2.117}$$

Substituting this polynomial representation of T(t) in the $\gamma_{i,j}$ equations (2.38 and 2.39) for ramp perturbation gives:

$$\gamma_{i,3} = \int_{t_{n-1}}^{t_n} f(\tau)^2 \sum_{j=1}^{J} \left[(a_j^{(2)} t^2 + a_j^{(1)} t^1 + a_j^{(0)}) \times H_j(t) \right] e^{-\lambda_i (t_n - \tau)} d\tau \tag{2.118}$$

$$\gamma_{i,2} = \int_{t_{n-1}}^{t_n} [1 - f(\tau)] f(\tau) \sum_{j=1}^{J} \left[(a_j^{(2)} t^2 + a_j^{(1)} t^1 + a_j^{(0)}) \times H_j(t) \right] e^{-\lambda_i (t_n - \tau)} d\tau \tag{2.119}$$

$$\gamma_{i,1} = \int_{t_{n-1}}^{t_n} [1 - f(\tau)]^2 \sum_{j=1}^{J} \left[(a_j^{(2)} t^2 + a_j^{(1)} t^1 + a_j^{(0)}) \times H_j(t) \right] e^{-\lambda_i (t_n - \tau)} d\tau \tag{2.120}$$

Similarly for step perturbations the $\gamma_{i,j}$ equations are:

$$\gamma_{i,3} = \int_{t_{n-1}}^{t_n} f(\tau) \sum_{j=1}^{J} \left[(a_j^{(2)} t^2 + a_j^{(1)} t^1 + a_j^{(0)}) \times H_j(t) \right] e^{-\lambda_i (t_n - \tau)} d\tau \tag{2.121}$$

$$\gamma_{i,2} = 0 \tag{2.122}$$

$$\gamma_{i,1} = \int_{t_{n-1}}^{t_n} [1 - f(\tau)] \sum_{j=1}^{J} \left[(a_j^{(2)} t^2 + a_j^{(1)} t^1 + a_j^{(0)}) \times H_j(t) \right] e^{-\lambda_i (t_n - \tau)} d\tau \quad (2.123)$$

2.9.4 Precursor Update - PRECDY

The precursor concentrations are updated at the end of each macro time step, before proceeding to the next step, by the PRECDY subroutine. For this purpose equation 2.80 is evaluated using the correct $\gamma_{i,j}$'s for either a step or ramp perturbation. For the sake of completeness the precursor equation is given again:

$$\begin{aligned} \chi \vec{C}_i(t_n) &= \beta_i(\gamma_{i,3}\mathbf{B}(t_n)\vec{\varphi}(t_n) + \gamma_{i,2}\mathbf{B}(t_n)\vec{\varphi}(t_{n-1}) + \\ &\quad \gamma_{i,2}\mathbf{B}(t_{n-1})\vec{\varphi}(t_n) + \gamma_{i,1}\mathbf{B}(t_{n-1})\vec{\varphi}(t_{n-1})) + \\ &\quad \chi \vec{C}_i(t_{n-1}) e^{(-\lambda_i (t_n - t_{n-1}))} \end{aligned} \quad (2.124)$$

Note that the $\gamma_{i,j}$'s are computed simultaneously with the solution of the point kinetics equations in the module PKIN.

2.9.5 IQS Convergence Test

The fixed-point iterative method is considered to have converged if the shape function calculation has converged and the following convergence criteria have been satisfied:

$$\rho^{(l+1)} - \rho^{(l)} < \epsilon_\rho \quad (2.125)$$

$$T^{(l+1)}(t_n) - T^{(l)}(t_n) < \epsilon_T \quad (2.126)$$

$$\frac{E^{(l+1)}(t_n) - E^{(0)}(t_n)}{E^{(0)}(t_n)} < \epsilon_E \quad (2.127)$$

where:

		Typical Values
ϵ_ρ	reactivity convergence criterion	0.01 mk
ϵ_T	amplitude convergence criterion	10^{-5}
ϵ_E	constraint convergence criterion	10^{-5}

These convergence criteria do not guarantee exactness of the solution but rather that a self-consistent solution has been achieved.

CHAPTER 3

Numerical Results for the Improved Quasistatic Method

The objective of this chapter is to validate and evaluate the performance of the IQS portion of the code XSTATIC, whose theoretical foundations were developed in the previous chapter. In order to do so, several reactor transients are simulated using both the constrained and unconstrained quasistatic methods. Emphasis is placed on the performance of the constrained version of the improved quasistatic. Results obtained using the unconstrained version of the quasistatic are presented only for insight into the constrained quasistatic method.

The following four test cases are analysed: SIMPLE, LMW, CRKB and AECL. The SIMPLE, CRKB and AECL test cases are respectively 1, 2 and 3 dimensional idealizations of a CANDU reactor transient following a large loss of coolant accident(LOCA). The LMW test case represents an idealized PWR reactor undergoing an operational transient. These cases are chosen because reference or reliable solutions are available. The SIMPLE test case is simple enough to allow reference solutions to be determined analytically. For the LMW test case a reference solution was generated by QUANDRY.[15] For the CRKB, which is a 2 dimensional CANDU reactor, a reference solution generated by ADEP[16] is available. Unfortunately no validated reference solution is available for the AECL test case, however a reliable CERBERUS[17] solution for the case exists.

In order to validate XSTATIC against a reference solution, a set of criteria is required. Let the reference regional power density P_i^* be defined as:

$$P_i^*(t) = \frac{1}{V_i}\int_{V_i} [H_1(\hat{r})\phi_1(\hat{r},t) + H_2(\hat{r})\phi_2(\hat{r},t)]d^3r \tag{3.1}$$

where $H_1(\hat{r})$ and $H_2(\hat{r})$ are power conversion factors and have units MeV/(cms).

Also let P_i be the computed regional power density. Also define the reference total power as:

$$P^*(t) = \sum_i P_i^*(t) V_i \tag{3.2}$$

Let $P(t)$ be the computed total power.

Using the above definitions, the following error criteria can be defined:

1. Total Power Error:

$$\varepsilon_{TP} = \frac{P(t) - P^*(t)}{P^*(t)} \tag{3.3}$$

2. Maximum Regional Power Density Error:

$$\varepsilon_{MAX}(t) = \max_i \frac{|P_i(t) - P_i^*(t)|}{P_i^*(t)} \quad \text{for } P_i^* \neq 0 \tag{3.4}$$

3. Average Regional Power Density Error:

$$\varepsilon_{AV}(t) = \frac{1}{V_{\text{core}}} \sum_i \frac{|P_i(t) - P_i^*(t)|}{P_i^*(t)} V_i \quad \text{for } P_i^* \neq 0 \tag{3.5}$$

where

$$V_{\text{core}} = \sum_i V_i \quad \text{for } P_i \neq 0 \tag{3.6}$$

3.1 SIMPLE Test Case

The SIMPLE test case is a bare 1-dimensional homogeneous 2-energy-group reactor with a zero flux boundary condition. A complete description of the test case is given in Appendix B.1. For this reactor configuration, simple analytic solutions exist if the transients are initiated by uniform step perturbations. The existence

Table 3.1: Parameters for SIMPLE Case

Parameter	Default Values
mesh split	1
time split	1
ε_ρ	0.5×10^{-5}
ε_T	0.5×10^{-5}
ε_E	0.5×10^{-5}
ε_{max}	0.5×10^{-5}
ε_{shape}	0.5×10^{-5}

of analytic solutions allows the point kinetics subroutines to be tested, and the convergence properties of the shape function solution algorithm to be investigated.

The analytic (or reference) solutions are generated by the code POLES. This code uses ESSL[1] routines to compute the eigenvector and eigenvalues of the system and then constructs the analytic solution (see Appendix A). An earlier version of the IMSL subroutine DGEAR, called STIFFZ, is used to generate the results in this section. The differences between the subroutines is minor and the two can be used interchangeably.

The steady-state results generated by XSTATIC are very close to the analytic results. A steady-state value for k_{eff} of 0.99806613 is obtained from XSTATIC, and the analytic k_{eff} given by POLES is 0.9981677, giving an error of -0.1 mk. The average and maximum power errors at t=0 compared to the analytic solution are 1.9×10^{-5} and 9.0×10^{-5} respectively. Table 3.1 contains relevant XSTATIC data input values used in all the SIMPLE test cases. A time split of N implies that the shape function calculation interval is subdivided into N intervals over which the complete shape function and point kinetics calculation are performed. A mesh split of N implies that all mesh spacings are reduced by N, and consequently the number of unknowns increases by N^3.

Two transient cases are run: a uniform step perturbation of the thermal absorption cross-section (Σ_{a2}) of $-.2 \times 10^{-4}$ and 1.0×10^{-4} cm^{-1} respectively. These

[1] ESSL is an IBM subroutine library

perturbations correspond to step reactivity insertions of 4.94 and -29.4 mk. Results for these runs using 8 macro time steps of 1 second are given in Table 3.2. The total power errors increase monotonically with time and are less than 0.2%. Errors of this magnitude are surprising because the ordinary differential equations integrators, either STIFFZ or FASTQ, are much more accurate than 0.2%. The error is due to a small error in the computation of the reactivity. For example during a zero transient ($\rho = 0.$ mk, ie. no perturbation of any property), the computed reactivity is roughly .01 mk ,instead of 0, and leads to an error of 0.12% after 10 seconds. This can be verified by evaluating the following prompt jump and constant delayed group approximation of the point kinetics equations at 8 seconds:

$$T(t) = \left(\frac{\beta}{\beta - \rho}\right) e^{\left(\frac{\rho\lambda}{\beta-\rho}\right)t} = e^{\left(\frac{\rho\lambda}{\beta}\right)t} \text{ for small } \rho \tag{3.7}$$

Another source of error is the round-off introduced as the code marches forward in time. The accumulation of error is evident from another zero transient run where a single time step from 0 to 8 seconds is more accurate than 8 one-second time steps.

3.1.1 Performance of the FASTQ Algorithm

The accuracy and performance of the FASTQ algorithm with respect to the STIFFZ algorithm are investigated. For step perturbations, the reference solution is constructed, using the code POLES by analytically integrating equation 2.85 for γ_3 using the analytic solutions for T(t) and summing over all precursor groups as in equation 2.97. (See Appendix A for more detail.) Shown in Figure 3.1 are plots of γ_3/T for step perturbations of -24.3, -4.0, 4.0, and 8.1 mk. The error in γ_3/T obtained using STIFFZ and FASTQ, with respect to the POLES solution (considered to be the reference) are shown in Figure 3.3 and 3.4 respectively. Also included for completeness is Figure 3.2 which illustrates the behavior of γ_3/T for

Figure 3.1: Variation of γ_3/T for a Step Perturbation

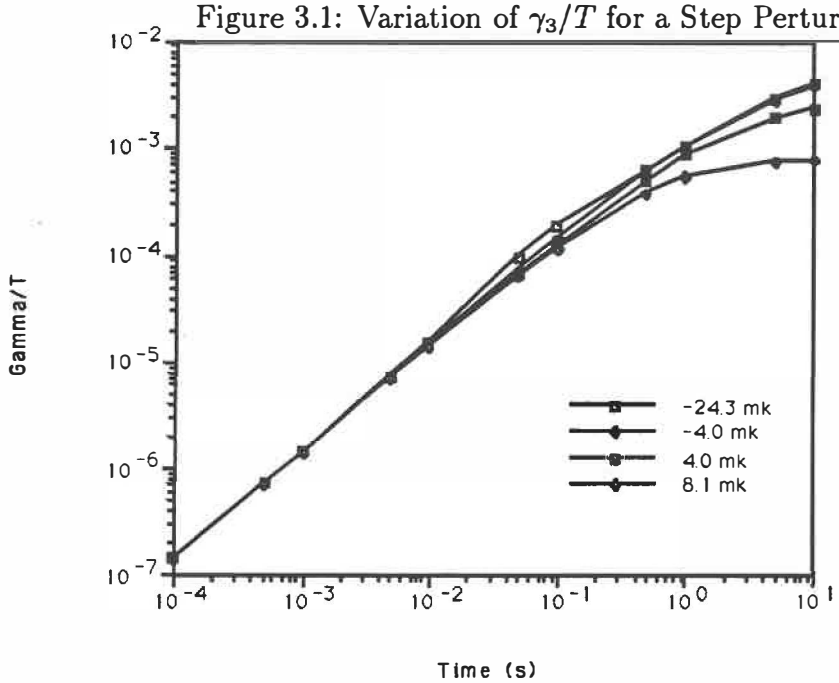


Figure 3.2: Variation of γ_3/T for a Ramp Perturbation

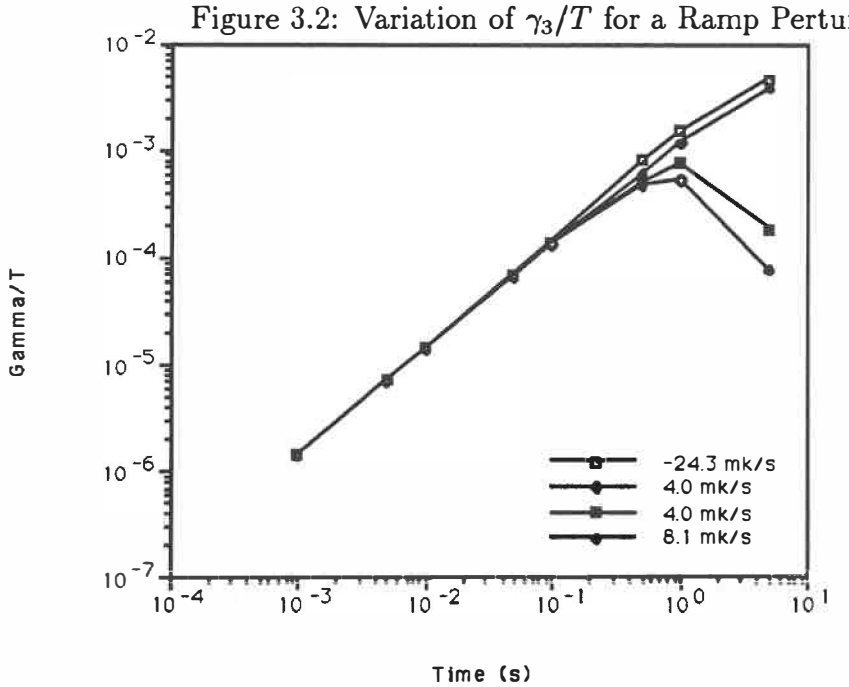


Table 3.2: SIMPLE: XSTATIC and Reference Powers

	ρ= 4.94 mk		ρ= -29.4 mk	
Time	XSTATIC	Reference	XSTATIC	Reference
0.0	1.000 (0.000%)	1.000	1.000 (0.000%)	1.000
1.00	4.31225 (0.009%)	4.31187	0.16834 (0.003%)	0.16834
2.00	7.80003 (0.041%)	7.7968	0.14379 (0.003%)	0.14379
4.00	21.8713 (0.087%)	21.8524	0.11325 (0.003%)	0.11325
6.00	59.0682 (0.13%)	59.0091	.093709 (0.003%)	.093709
8.00	158.333 (0.17%)	158.056	0.08014 (0.003%)	.08014

ramp perturbations of -24.3, -4.0, 4.0 and 8.1 mk/s. These results were generated using direct numerical integration of the point kinetics equation with a very stringent convergence criterion.

It is evident from Figures 3.3 and 3.4 that for most transients the FASTQ subroutine gives results comparable to or better than those of STIFFZ. If the macro time step is smaller than 5×10^{-2} seconds the error incurred using STIFFZ rises dramatically. Furthermore, FASTQ generated these results with an execution time between 3 and 10 times smaller than STIFFZ. Figure 3.5 shows a scatter plot of the ratio

$$R = \frac{CPU_{STIFFZ}}{CPU_{FAST}} \tag{3.8}$$

for many step and ramp transients.

Figure 3.3: Error in γ_3/T for FASTQ

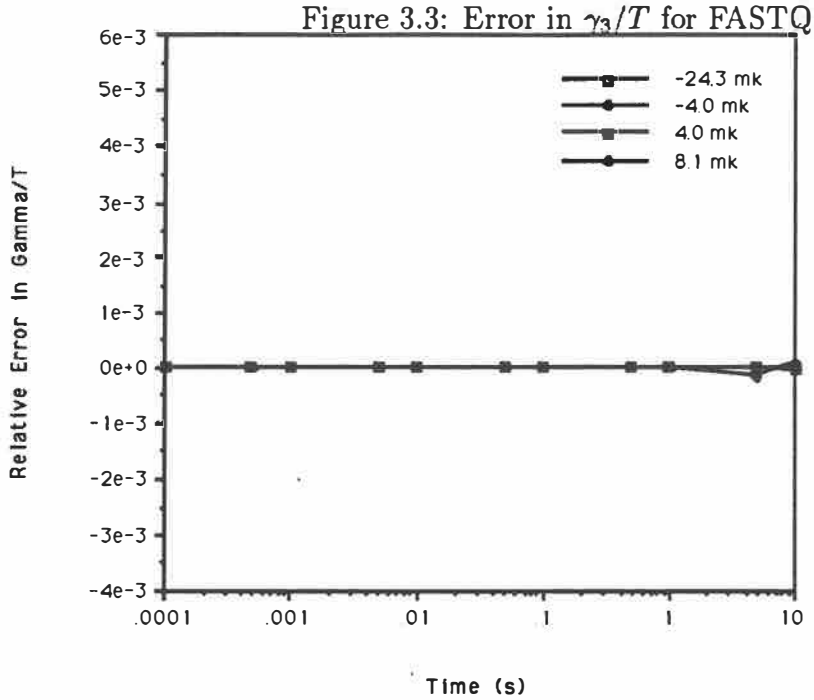


Figure 3.4: Error in γ_3/T for STIFFZ

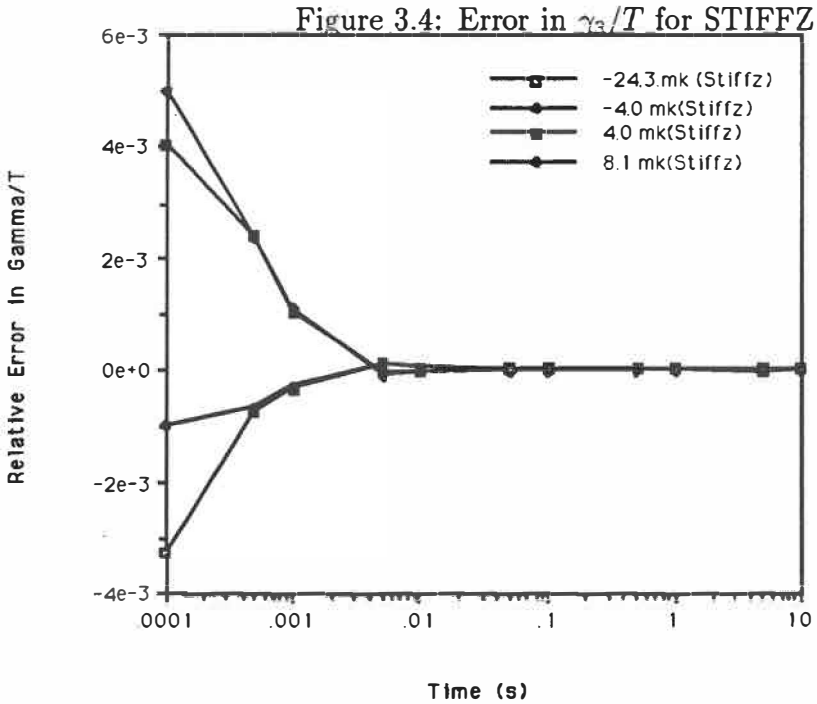
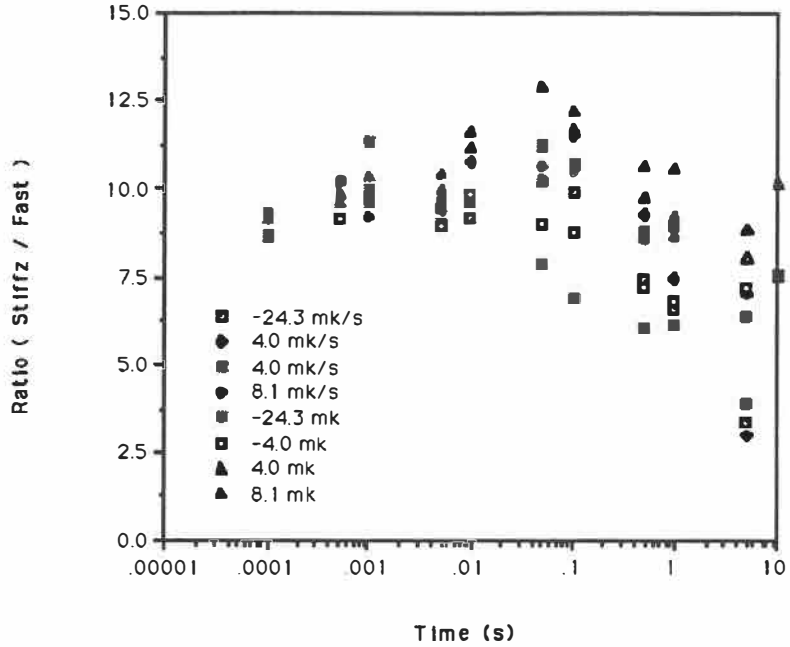


Figure 3.5: Ratio of STIFFZ to FASTQ CPU times



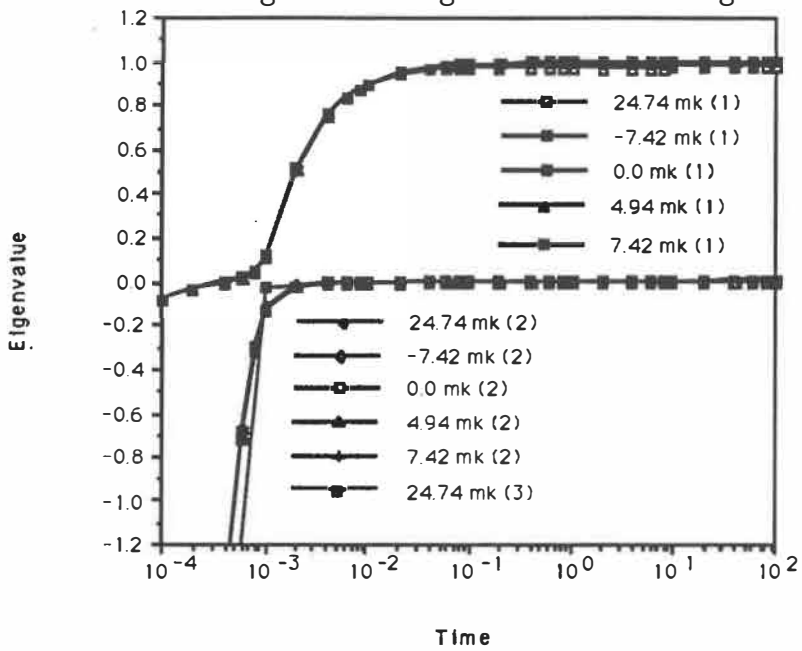
3.1.2 Spectral Radius of Residual Matrix R

As stated in section 2.9, the iterative strategy converges if the spectral radius of the residual matrix **R** is less than 1. Using this simple test case, the spectral radius of **R** was investigated. The iterative matrix given in equation 2.99 can be simplified using the fact that the ADI inversion of $\mathbf{A}_{gg}$ is exact for a one-dimensional problem such as SIMPLE. The preconditioning matrix **M** is then equal to $\mathbf{A}^{-1}$ and the iterative matrix can be rewritten as:

$$\mathbf{R} = \begin{pmatrix} \mathbf{A}_{11} & 0 \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}^{-1} \times \tag{3.9}$$
$$\left\{ (1-\beta+\frac{\gamma_3}{T}) \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} \frac{1}{v_1}[\frac{1}{\Delta t}+\frac{1}{T}\frac{dT}{dt}]\mathbf{u} & 0 \\ 0 & \frac{1}{v_2}[\frac{1}{\Delta t}+\frac{1}{T}\frac{dT}{dt}]\mathbf{u} \end{pmatrix} \right\}$$

The individual influence of the various parameters γ_3 and Δt now becomes more apparent. For example: as γ_3/T increases towards β, the spectral radius increases and as the macro time step Δt diminishes, the spectral radius diminishes. The spectral radius of the system was determined for a series of step perturbations of

Figure 3.6: Largest and Smallest Eignvalues of **R**



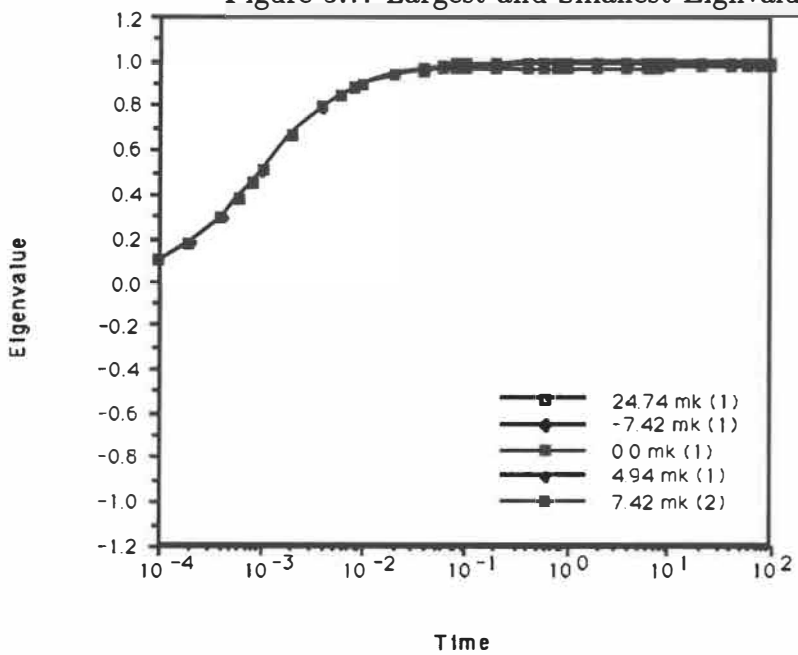
-24.3, -7.42, 0., 4.94, 7.42 mk in the thermal production cross-section and at various macro time step sizes using POLES (see Appendix B). The spectral radius is evaluated in POLES by computing the value of the parameters γ_3/T and $1/T(dT/dt)$ for a given reactivity transient and macro time step, and substituting these values into the equation for **R**. Finally, the eigenvalues are determined analytically. A graph of the spectral radius, in the form of the maximum and minimum eigenvalues of the iterative matrix, is shown in Figure 3.6.

Another iterative scheme can be constructed by combining the $1/\Delta t\mathbf{U}$ term with the **A** matrix. The modified residual matrix $\mathbf{R}^*$ is:

$$\mathbf{R}^* = \left\{ \begin{pmatrix} \mathbf{A}_{11} & 0 \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} - \begin{pmatrix} [\frac{1}{v_1 \Delta t} + \frac{1}{T}\frac{dT}{dt}]\mathbf{U} & 0 \\ 0 & [\frac{1}{v_2 \Delta t} + \frac{1}{T}\frac{dT}{dt}]\mathbf{U} \end{pmatrix} \right\}^{-1} \times (1 - \beta + \frac{\gamma_3}{T}) \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ 0 & 0 \end{pmatrix} \tag{3.10}$$

The spectral radius of the modified iterative scheme versus time step is given in Figure 3.7. It is interesting to note the difference in the two plots. The modified

Figure 3.7: Largest and Smallest Eignvalues of $\mathbf{R}^*$

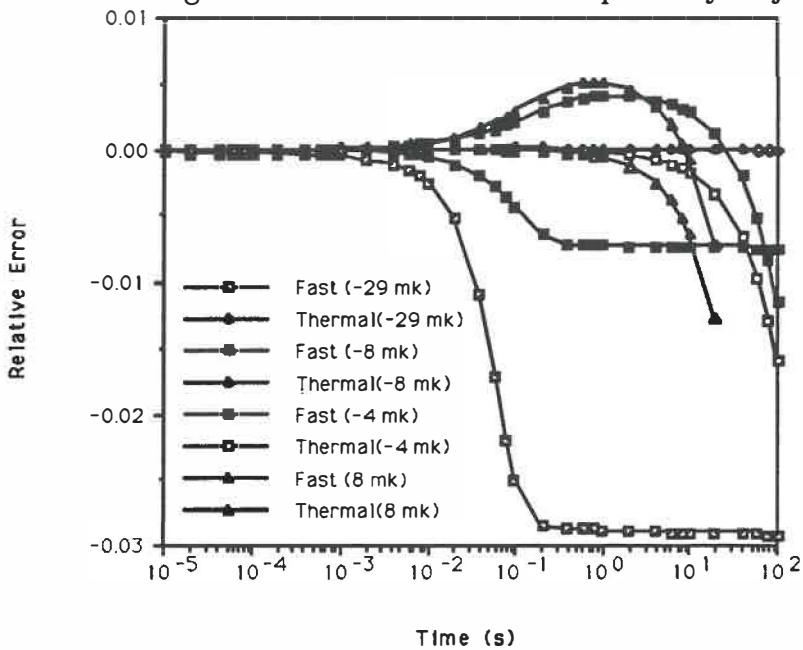


residual matrix $\mathbf{R}^*$ has N grid points and 2-energy-groups but only N eigenvalues. The introduction of $\frac{1}{\Delta t} + \frac{1}{T}\frac{dT}{dt}$ into the $\mathbf{B}$ matrix causes the eigenvalues of $\mathbf{R}$ to split. These new eigenvalues (labelled (2)) have little effect on the spectral radius of the system when the macro time step is large, because they are dominated by the fundamental spatial (no zero crossing) eigenvalue (labelled (1)) except when Δt is very small. The spectral radius of the iterative matrix $\mathbf{R}^*$ behaves much better for small Δt's. However for large Δts (of the order of 100 seconds) the spectral radius is larger than 1 and the system is unstable. For the range between .01 and 5 seconds there is no appreciable difference in the spectral radii.

3.1.3 Choice of the Weight Function

In this section, the effect of the choice of the weight function on the accuracy of the point kinetics solution is investigated. Typically three choices for the weight function are used: adjoint, Galerkin and unity weighting. Only the choice of the

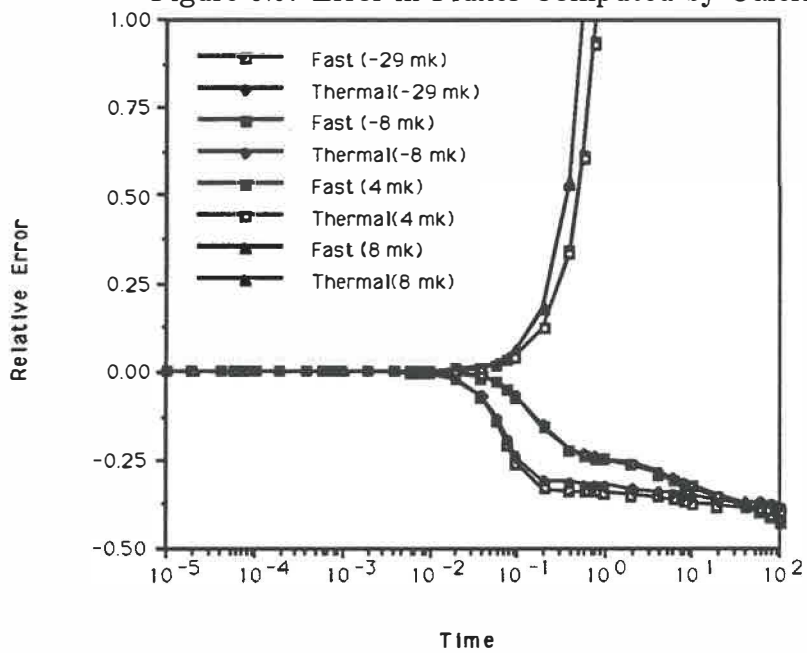
Figure 3.8: Error in Fluxes Computed by Adjoint Weighting



adjoint flux is based on theoretical considerations. As shown in section 2.5, it minimizes the error in the reactivity due to small errors in the flux. Computing the adjoint flux is expensive in comparison to a point kinetics calculation. This extra effort can be avoided, especially when a very accurate solution is not required, by using unity or Galerkin weighting. Although unity weighting is frequently used, only Galerkin and adjoint weighting is investigated here.

Several runs were used to establish the error incurred when using Galerkin instead of adjoint weighting. Figures 3.8 and 3.9 show the error in the average fast and thermal fluxes for several uniform step perturbations of the thermal absorption cross section, using adjoint and Galerkin weighting respectively. The reference solutions were generated by the code POLES which analytically solves the point kinetics equation. It is obvious form the Figures that adjoint weighting is far superior to Galerkin weighting for all cases. The extra effort required to obtain the adjoint flux is rewarded by a much more accurate point kinetics solution.

Figure 3.9: Error in Fluxes Computed by Galerkin Weighting



3.2 LMW Test Case

Further evaluation of the IQS method was obtained through the LMW test case, which can be characterized as an operational transient with large spatial variations involving an LWR type reactor. The original problem proposed by Lagenbuch, Mauer, and Werner[18] is a three-dimensional 1/8 core benchmark. Greenman[15] transformed it to a two-dimensional 1/4 core benchmark. The strength of the control rods were doubled to increase the spatial distortion. Appendix B.2 gives a full description of the transient.

Withdrawal of a rod bank initiates a transient which is terminated by the insertion of another rod bank. The power increases to about 450% full power at 10.0 seconds before being terminated by shutdown rod insertion, returning to approximately 1.0 at 25.0 seconds. From a spatial distortion point of view, the transient is quite severe. The maximum change in normalized power density is roughly 30%. Figure 3.10 shows the thermal shape function at various times and graphically illustrates the severity of the transient.

The results generated by QUANDRY[15] and presented by Greenman in his thesis are used as the reference for this analysis. Table 3.3 contains the reference total power versus time as given by Greenman. He also included regional powers at 0, 10. and 25. seconds.

The LMW test case was run using Lagrange linear (LAGR1) and second-order

Table 3.3: LMW: Reference Total Power [Greenman]

Time (s)	Power (-)
0	61600
5	92200
10	277500
15	160400
20	79020
25	53232

mesh-centered finite-difference (MCFD2) discretization for several mesh and time discretizations. The results are summarized in Tables 3.4 and 3.5. Using LAGR1 with no mesh or time splitting results in relatively large errors of roughly 6% at 10 and 25 seconds. Increasing the time splitting to 20 changes the results only slightly, indicating that the case is temporally converged and the errors result from an unconverged spatial representation. Increasing the mesh splitting from 1 to 4 reduces the average power error to 3.3% at 10 seconds. The MCFD cases are very similar. Increasing the time splitting leads to small drops in the average power error and increasing the mesh splitting reduces the error substantially.

Using the unconstrained version of XSTATIC gives the results shown in Table 3.6. The results obtained are very similar to those obtained using the constrained version. In the LAGR1 case the results of the unconstrained version of the code are more accurate than those of the constrained version for the peak power at 10 seconds. At 25 seconds, after shutdown is initiated, the situation is reversed with the constrained solution being more accurate than the unconstrained solution. This trend of the unconstrained version more correctly predicting the portion of the transient in which the rods are being driven into the core, is also true for the CRKB test case. The CRKB test case is much more severe and the differences between the constrained and unconstrained methods will become more apparent.

Table 3.4: LMW: XSTATIC Results for LAGR1

Type	LAGR1	LAGR1	LAGR1	LAGR1
Split	1	1	2	4
Step	1	20	20	20
P(0 s)	61600	61600	61600	61600
P(5 s)	94752	94744	93682	93168
P(10 s)	287792	287553	284966	283356
P(15 s)	160180	160235	162324	162387
P(20 s)	74484	74536	78075	79055
P(25 s)	53737	53779	57036	58021
$\epsilon_{AV}(0s)$	2.9	2.9	3.6	1.7
$\epsilon_{MAX}(0s$	8.2	8.2	10.1	5.1
$\epsilon_{TP}(10s)$	4.3	3.4	3.2	2.7
$\epsilon_{AV}(10s)$	5.4	5.4	4.7	3.3
$\epsilon_{MAX}(10s)$	13.3	13.3	13.9	7.9
$\epsilon_{TP}(25s)$	-7.7	6.7	3.9	-0.3
$\epsilon_{AV}(25s)$	6.7	6.7	3.9	1.8
$\epsilon_{MAX}(25s)$	12.7	12.7	8.2	4.8

Table 3.5: LMW: XSTATIC Results for MCFD2

Type	MCFD2	MCFD2	MCFD2	MCFD2
Split	1	2	1	2
Step	1	20	20	20
P(0 s)	61600	61600	61600	61600
P(5 s)	92653	92623	92626	92699
P(10 s)	280340	279616	279555	280062
P(15 s)	160337	160431	160230	160431
P(20 s)	78662	78581	78668	78581
P(25 s)	57925	57813	57931	57813
$\epsilon_{AV}(0s)$	1.7	0.9	1.7	0.9
$\epsilon_{MAX}(0s)$	4.0	2.2	4.0	2.2
$\epsilon_{TP}(10s)$	1.5	1.3	1.3	1.5
$\epsilon_{AV}(10s)$	1.9	1.2	1.9	1.3
$\epsilon_{MAX}(10s)$	3.2	2.3	2.8	2.4
$\epsilon_{TP}(25s)$	-0.5	-1.2	-0.5	-0.7
$\epsilon_{AV}(25s)$	1.7	1.6	1.7	1.2
$\epsilon_{MAX}(25s)$	4.6	3.5	4.6	3.0

Table 3.6: LMW: Unconstrained XSTATIC Results

Type Split Step	LAGR1 1 1	LAGR1 1 20	MCFD2 1 1
P(0 s)	61600	61600	61600
P(5 s)	94757	94748	92655
P(10 s)	287846	287556	280670
P(15 s)	160205	160238	160610
P(20 s)	74505	74537	78819
P(25 s)	53750	53780	58023
$\epsilon_{AV}(0s)$	2.9%	2.9%	1.7%
$\epsilon_{MAX}(0s)$	8.2%	8.2%	4.0%
$\epsilon_{MAX}(10s)$	13.%	13.%	3.2%
$\epsilon_{AV}(25s)$	6.7%	6.7%	1.6%
$\epsilon_{MAX}(25s)$	13.%	13.%	4.6%

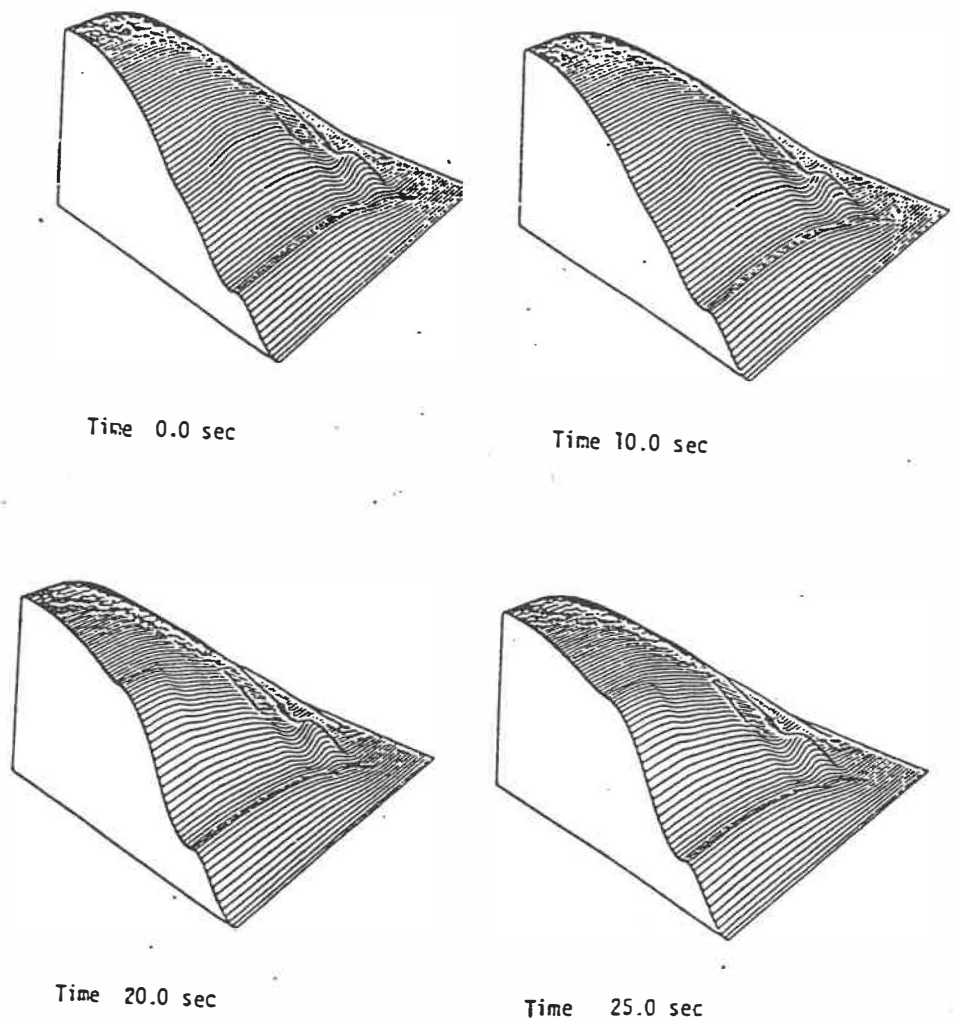


Figure 3.10: LMW: Shape function at Several Times

3.3 CRKB Test Case

This small but difficult 2-dimensional problem was proposed by McDonnell et al. [19] and is referred to as test case CRKB in this report. It is based on an idealized representation of a CANDU reactor. The transient is initiated by a uniform loss of coolant accident that increases the thermal neutron production cross section by 5% between 0 and 1 second. The reactor power rises to 2.7 times the original power before the transient is terminated by an asymmetric insertion of the shutoff rods between 1.5 and 2.9 seconds. Severe flux distortion causes the normalized power density to change by a factor of 3 in some regions. A full description of the problem is given in Appendix B.2.

McDonnell generated a reference solution using a two-dimensional "mesh-edge" finite-difference code called ADEP[16]. He used ADEP results to validate a three-dimensional MCFD improved quasistatic code called CERKIN. Since XSTATIC can also be run in the mesh-centered finite-difference mode, the two IQS codes can be compared using the same mesh spacing. This should eliminate the contribution of the spatial discretization error and allow a comparison of the different techniques used to solve the IQS equation in CERKIN and XSTATIC.

McDonnell states that the mesh spacings used were $\Delta x = 12.5cm$ and $\Delta y = 100.cm$. Using this mesh spacing XSTATIC gave a k_{eff} of 1.0075206 which is very close to the k_{eff} of 1.007489 and 1.007484 given by ADEP and CERKIN respectively.

CERKIN, unlike most codes, defines the variation of the thermal production cross-section with time to be

$$\nu\Sigma_f(t) = \frac{1}{k_{eff}}\{\nu\Sigma_f(t=0) + \Delta\nu\Sigma_f(t)\} \tag{3.11}$$

whereas XSTATIC, QUANDRY and other codes define the variation of neutron production cross section to be:

Table 3.7: CRKB: CERKIN and ADEP Results for Average Thermal Flux

	CERKIN				ADEP
Time	Split 1 $\bar{\phi}_2$	Split 2 $\bar{\phi}_2$	Split 4 $\bar{\phi}_2$	Split 8 $\bar{\phi}_2$	$\Delta t = 2.5ms$ $\bar{\phi}_2$
0.0	1.000	1.000	1.000	1.000	1.000
1.00	2.694	2.694	2.694	2.694	2.694
1.25	—	1.919	1.712	1.654	1.633
1.50	1.090	0.913	0.882	0.876	0.868
2.20	—	0.544	0.536	0.533	0.525
2.90	0.481	0.465	0.461	0.459	0.452

Table 3.8: CRKB: XSTATIC Results for Average Thermal Flux

Time	Split 1 $\bar{\phi}_2$	Split 2 $\bar{\phi}_2$	Split 4 $\bar{\phi}_2$	Split 8 $\bar{\phi}_2$	Split 20 $\bar{\phi}_2$
0.0	1.000	1.000	1.000	1.000	1.000
1.00	2.694	2.694	2.694	2.694	2.694
1.25	—	1.658	1.663	1.658	1.652
1.50	0.721	0.824	0.857	0.869	0.877
2.20	—	0.514	0.525	0.528	0.530
2.90	0.425	0.448	0.454	0.455	0.457

$$\nu\Sigma_f(t) = \frac{\nu\Sigma_f(t=0)}{k_{eff}} + \Delta\nu\Sigma_f(t) \tag{3.12}$$

Therefore, the perturbation in thermal production cross section given by McDonnell of $.2298 \times 10^{-4}$ was scaled down by 1.0075206 to $.22809 \times 10^{-4}$ for use in XSTATIC.

For all CERKIN and XSTATIC cases, the minimum number of shape function calculations is 3, these occur at 1.0, 1.5 and 2.9 seconds. A temporal split of N implies that the function is computed N times in the intervals between the minimum set of shape calculations, except for the interval between 0.0 and 1.0 second. For example a split of 2 implies that a total of 5 shape function calculations are performed at 1.0, 1.25, 1.50, 2.20 and 2.90 seconds.

Table 3.9: CRKB: Results for Local Thermal Flux

	CERKIN(Split 4)		ADEP($\Delta t = 2.5ms$)		XSTATIC(Split 4)	
Time	$\phi_2(225)$	$\phi_2(575)$	$\phi_2(225)$	$\phi_2(575)$	$\phi_2(225)$	$\phi_2(575)$
0.0	1.000	1.000	1.000	1.000	1.000	1.000
1.00	2.689	2.698	2.696	2.693	2.691	2.691
1.25	0.837	2.502	0.819	2.479	0.835	2.499
1.50	0.261	1.535	0.260	1.521	0.263	1.535
1.85	0.137	1.137	0.137	1.119	-	-
2.20	0.093	1.024	0.092	1.010	0.093	1.017
2.90	0.054	0.923	0.054	0.912	0.054	0.919

3.3.1 Constrained XSTATIC Results

The results obtained by McDonnell from CERKIN and ADEP are summarized in Table 3.7 which contains volume averaged thermal fluxes ($\bar{\phi}_2$) for various cases and Table 3.9 contains local thermal fluxes at $x = 225$ cm and $x = 575$ cm for $y = 400$ cm. In CERKIN the local fluxes were approximated by taking the average of mesh center fluxes on either side of these points.

The average thermal flux results obtained with the constrained version of XSTATIC using a MCFD1 spatial discretization and the same mesh spacing are shown in Table 3.8. For a temporal mesh split of 8 (ie. a total of 17 shape function evaluations during the transient), the CERKIN and XSTATIC average thermal flux results agree to within 0.6%. That is, for a small shape function (macro) time step, the two quasistatic MCFD codes give almost identical answers. For a temporal split of both 8 and 20, the maximum XSTATIC error relative to the ADEP solution is 1.2%. The error of the quasistatic codes relative to ADEP is slightly larger but this is probably due to different spatial discretization techniques used by the 2 codes (ie. finite-difference versus MCFD).

It is interesting to note the behavior of XSTATIC and CERKIN for large time

steps. For a mesh split of 2, CERKIN overestimates the average flux by 17% at 1.5 seconds, compared to only 5% by XSTATIC. In fact CERKIN has a larger error for almost all mesh splittings. The discrepancy, as expected, decreases as the mesh split increases.

3.3.2 Unconstrained XSTATIC Results

Table 3.10 show the results obtained for various time splits with the unconstrained versions of XSTATIC. For a time split of 20, the constrained and unconstrained versions of XSTATIC give the same results. That is, as the time step decreases, the effect of floating the constraint also decreases. For a mesh split of 1 the results given by the 2 methods vary considerably. The unconstrained version overestimates the average flux, while the constrained version underestimates it. This behavior is easily understood by examining the reactivity transient given in Table 3.11. The constrained method overestimates the reactivity insertion of the shutdown system at 1.5 s and consequently causes the power to be underestimated. In the constrained case, the reactivity is close to the reference value, leading to a better prediction of reactor power.

From the above results, it would appear that the unconstrained version of XSTATIC gives more accurate results than the constrained version when the time step is large (0.5 to 0.25 s). When the time step is 0.10 s or smaller, the two methods give the same results.

Table 3.10: CRKB: Unconstrained XSTATIC Results for Average Thermal Flux

Time	Split 1 $\bar{\phi}_2$	Split 2 $\bar{\phi}_2$	Split 4 $\bar{\phi}_2$	Split 8 $\bar{\phi}_2$	Split 20 $\bar{\phi}_2$
0.0	1.000	1.000	1.000	1.000	1.000
1.00	2.693	2.694	2.694	2.694	2.694
1.25	—	1.776	1.689	1.655	1.652
1.50	0.870	0.885	0.874	0.874	0.878
2.20	—	0.533	0.530	0.530	0.530
2.90	0.457	0.459	0.457	0.457	0.456

Table 3.11: CRKB: Reactivity Transients

	Split 1		Split 8	
Time	Unconstrained (mk)	Constrained (mk)	Unconstrained (mk)	Constrained (mk)
0.0	0.000	0.000	0.000	0.000
1.00	4.982	4.982	4.974	4.974
1.25	—	—	-0.811	-0.849
1.50	-2.976	-3.678	-2.948	-2.968
2.20	—	—	-5.990	-6.000
2.90	-7.330	-7.810	7.360	-7.370

3.4 AECL Test Case

This three-dimensional two-energy-group benchmark represents, in an idealized CANDU reactor, a half core LOCA followed by an asymmetric insertion of shutdown devices. This test case, which is very similar to a typical CANDU 6 large LOCA safety analysis transient, was proposed by AECL and is referred to as test case AECL. A full description of the problem is given in Appendix B. This problem is quite similar to the two-dimensional two-energy-group CRKB test case. Both are based on the same type of reactor accident scenario and both cause severe flux distortions.

The solutions presented by Rouben et al [17] were generated using a MCFD (mesh-centered finite-difference) code called CERBERUS. Table 3.12 gives the reference power transient generated by Rouben. The report also provides regional powers at various times and therefore an analysis of the spatial convergence is possible. The macro times steps for flux shape calculations were chosen as follows: For the initial and final parts of the transient (ie. before the absorbers start to move and after they are fully inserted) a uniform time interval of 0.1 s was chosen. For the period when the absorbers are moving into the core, the times are chosen as those instants when the leading edge of the absorbers coincides with consecutive mesh lines in the y-direction. This choice leads to a total of 27 shape function calculations (including t=0) for a transient that lasts 2.5 seconds.

3.4.1 Constrained XSTATIC Results

The results obtained using XSTATIC are shown in Table 3.13 and the reference results are assumed to be those given by CERBERUS. XSTATIC underestimates the total power by 10% (at 1.47 s) during the period when the insertion of the shutdown rods is causing the power to decrease rapidly. The total power error recovers at about 2 seconds because the reactor power is dependent mostly on long

Table 3.12: AECL: Power versus Time

Time (s)	*Power* ($-$)
0.0	1.0000
0.2	1.1999
0.6	2.8100
0.8885	3.6866
1.35	1.9648
1.4654	1.4140
2.1	0.3422
2.5	0.3250

k_{eff}= 1.003555

lived precursor groups and the reactivity worth of the shutdown system.

3.4.2 Linear Variation of Point Kinetics Parameters

For this case a very large error (the maximum regional power error is 10% and the total power error is -9.5%) exists at 1.47 seconds. Differences of this magnitude are large for 2 codes performing the same calculation in the same way. The difference was traced to the manner in which the point kinetics equations are derived. In XSTATIC, the temporal variation of reactivity $\frac{\rho}{\Lambda}$ across a macro time step, given in equation 2.69, is parabolic in the numerator and linear in the denominator. More specifically, the change in nuclear properties and the flux are assumed to vary linearly over a macro time step producing a non linear (parabolic) variation in the numerator. CERBERUS uses a linear variation of reactivity ρ between the endpoints of the macro interval and no variation for the Λ across a macro time step. The resulting equations are of the following form:

$$\rho(t) = \frac{\vec{W}^t\left[f(\mathbf{A}(t_n)+\mathbf{B}(t_n))\vec{\varphi}(t_n)+(1-f)(\mathbf{A}(t_{n-1})+\mathbf{B}(t_{n-1}))\vec{\varphi}(t_{n-1})\right]}{\vec{W}^t\mathbf{B}(t_n)\vec{\varphi}(t_n)} \quad (3.13)$$

Table 3.13: AECL: Constrained XSTATIC Results with CEBERUS Reference (tsplit=msplit=1)

Time (s)	ε_{TP} %	ε_{AV} %	ε_{MAX} %
0.00	0.00	0.01	0.04
0.20	-0.10	0.05	0.13
0.60	-0.14	0.08	0.18
0.89	-0.33	0.19	0.64
1.35	-8.86	4.99	9.33
1.47	-9.54	5.30	10.36
2.10	-2.36	1.15	3.11
2.50	-1.97	0.96	2.50

using CERBERUS reference

$$\Lambda(t) = \frac{\vec{W}^t \mathbf{B}(t_n)\vec{\varphi}(t_n)}{\vec{W}^t \mathbf{V}^{-1}\mathbf{U}\vec{\varphi}(t_n)} \tag{3.14}$$

The variation of $\frac{\rho}{\Lambda}$ over the macro time step is also linear and is given by:

$$\frac{\rho}{\Lambda}(t) = \frac{\vec{W}^t \left[f(\mathbf{A}(t_n)+\mathbf{B}(t_n))\vec{\varphi}(t_n) + (1-f)(\mathbf{A}(t_{n-1})+\mathbf{B}(t_{n-1}))\vec{\varphi}(t_{n-1})\right]}{\vec{W}^t \mathbf{V}^{-1}\mathbf{U}\vec{\varphi}(t_n)} \tag{3.15}$$

This interpretation can also be applied to simplify the precursor equation, where instead of integrating a parabolic variation in $\mathbf{B}\vec{\varphi}$, only a linear integration of the following form is needed:

$$\begin{aligned} \chi\vec{C}_i(t_n) &= \beta_i \int_{t_{n-1}}^{t_n} \left(f\mathbf{B}(t_n)\vec{\varphi}(t_n) + (1-f)\mathbf{B}(t_{n-1})\vec{\varphi}(t_{n-1})\right) e^{(-\lambda_i(t-\tau))} d\tau \\ &+ \chi\vec{C}_i(t_{n-1}) e^{(-\lambda_i(t_n - t_{n-1}))} \end{aligned} \tag{3.16}$$

Using this definition of reactivity, the results shown in Table 3.14 are obtained. The results given by XSTATIC and CERBERUS now agree very closely, with a maximum error of 1.6% on total power and 1.9% regional power. Figures 3.11 and 3.12 show the variation of reactivity and generation time, respectively, for both definitions of reactivity. Note the parabolic shape of the reactivity curves for the XSTATIC definition of reactivity.

Figure 3.11: AECL: Variation of Reactivity for CERBERUS like Definitions

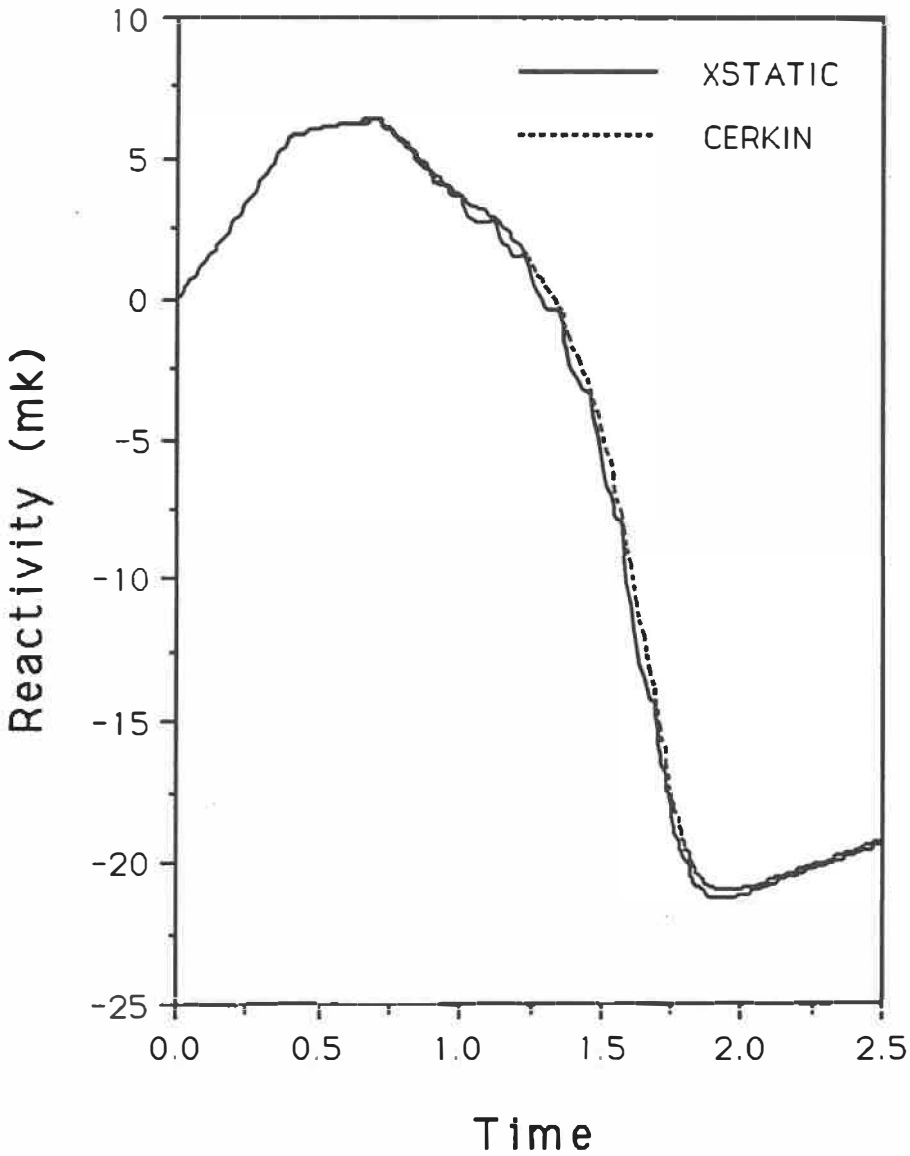


Table 3.14: AECL: XSTATIC Results with Linear Reactivity with CERBERUS Reference (tsplit=msplit=1)

Time (s)	ε_{TP} %	ε_{AV} %	ε_{MAX} %
0.00	0.00	0.01	0.04
0.20	0.01	0.01	0.04
0.60	0.04	0.02	0.09
0.89	1.46	0.88	1.89
1.35	0.55	0.33	0.62
1.47	0.59	0.33	1.47
2.10	0.29	0.17	0.52
2.50	0.38	0.21	0.48

using CERBERUS reference

Figure 3.12: AECL: Variation of Generation Time for CERBERUS like Definitions

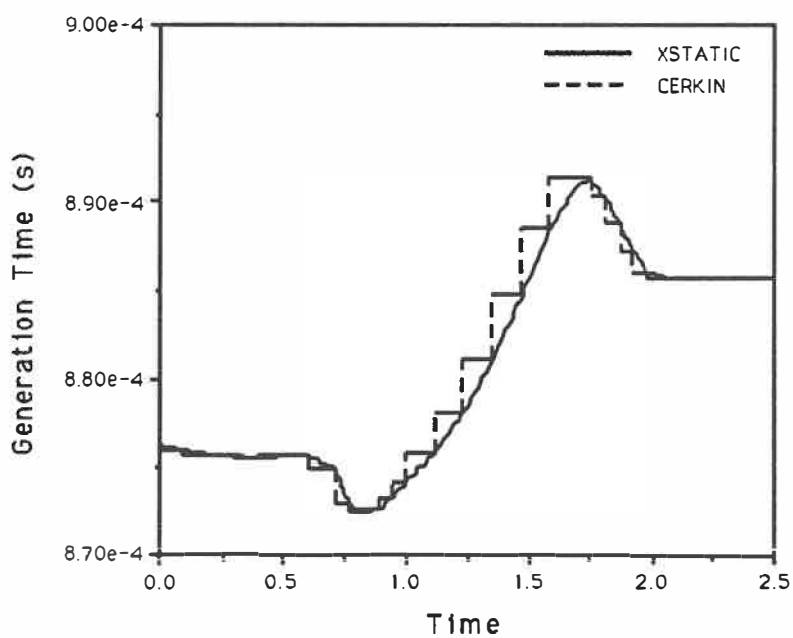


Table 3.15: AECL: Constrained XSTATIC Results (tsplit=msplit=1)

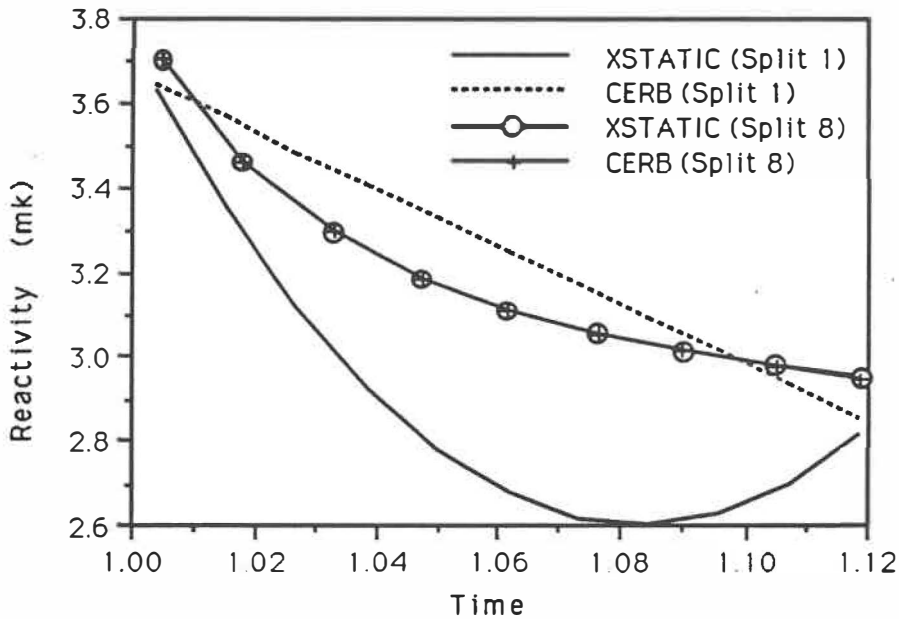
Time (s)	ε_{TP} %	ε_{AV} %	ε_{MAX} %
0.00	0.00	0.01	0.04
0.20	0.06	0.11	0.39
0.60	-0.24	0.14	0.51
0.89	-0.73	0.41	0.95
1.35	-6.74	3.67	7.57
1.47	-6.98	3.74	8.04
2.10	-1.89	0.91	2.29
2.50	-1.64	0.79	2.06

using XSTATIC reference

The test case was rerun with a time split of 8 using both the CERBERUS and XSTATIC definitions of reactivity. The maximum regional power difference between the cases is less than 0.2%. Since the results agree very closely, the XSTATIC result was chosen to be the new reference solution. For completeness the errors for time split of 1 (given in Table 3.13) were recomputed using this new reference solution and is given in Table 3.15.

Figure 3.13 shows the variation of the reactivity during the insertion of the shut off rod between 1.004 and 1.119 seconds. This interval is also a macro time step for a time split of 1. The Figure contains 4 reactivity curves: CERBERUS type reactivity for a time split of 1 and 8, and a XSTATIC reactivity for a time split of 1 and 8. As expected, the CERBERUS type reactivity predictions for a time split of 1 is linear over the macro time step. For a time split of 8, the CERBERUS reactivity predictions, which consists of 8 straight line segments, over the interval is parabolic in shape, and is close to the XSTATIC time split 8 result. The reactivity transient for time split 1, XSTATIC reaches a minimum before the end of the time step and appears strange. This type of behaviour is worrisome because it is probably not physical, but it does explain why XSTATIC underestimates the total power transients with respect to CERBERUS.

Figure 3.13: AECL: Variation of Reactivity During a Macro Time Step



3.4.3 Rod Cusping

The reactivity curves from the previous section indicate the reactivity is cusped at times when the rod tip crosses a mesh line. This behaviour is not physical because mother nature does not know were the mesh lines are. The cusping is due to the representation of the continuous movement of the shutdown system rod through a mesh cell as a patch which becomes progressively darker (more absorbant). Using a parabolic variation of the reactivity during the time it takes the rod to cross the mesh cell or patch leads to cusping at the endpoints. It was also shown in the previous section that using a linear variation of reactivity and allowing a shape function calculation at times when the rod is only partially inserted in the cell also leads to reactivity cusping.

Other codes, such as for example QUANDRY, pro rate the change in cross sections as a linear function of the volume that the rod occupies over the total cell

Table 3.16: AECL: Unconstrained XSTATIC Results (tsplit = msplit = 1)

Time (s)	ε_{TP} %	ε_{AV} %	ε_{MAX} %
0.00	0.00	0.01	0.04
0.20	0.06	0.11	0.39
0.60	-0.23	0.14	0.51
0.89	-0.57	0.33	0.84
1.35	-3.97	2.18	4.39
1.47	-4.26	2.28	4.88
2.10	-1.24	0.61	1.44
2.50	-1.07	0.51	1.31

using XSTATIC reference

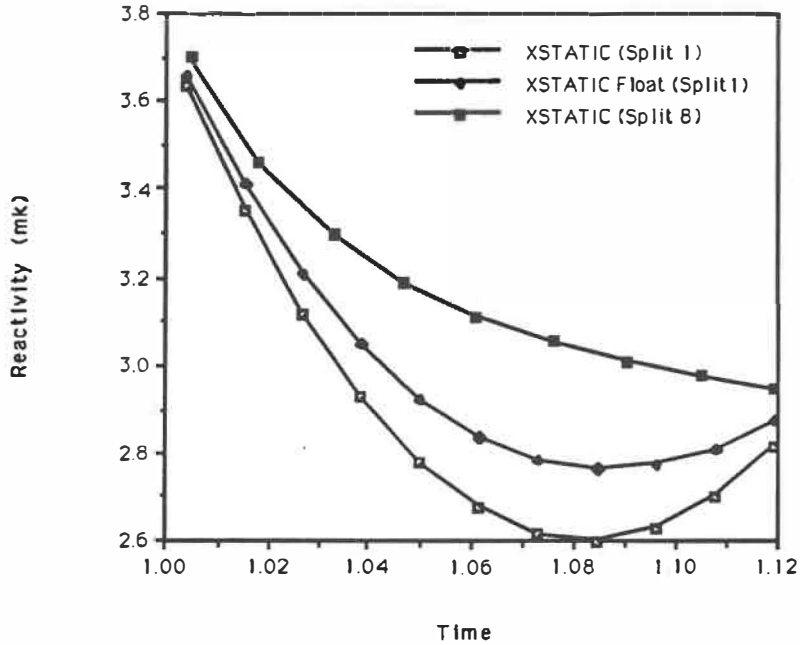
volume. This assumption is equivalent to the linear variation in nuclear properties assumed here. QUANDRY uses a brute force solution technique with small time steps and hence experiences rods cusping. Consequently to generate solutions that agree with QUANDRY rod cusping must the allowed. However, it is probably more correct to use a linear variation of the reactivity and thus eliminate rod cusping. An even better approach would be to compute the nuclear cross sections for the rod when it is partially inserted in the cell using transport methods.

3.4.4 Unconstrained XSTATIC Results

Permitting the constraint to vary allows another degree of freedom that may lead to a more a precise solution. The floating constraint results for a time split of 8 are very close (ϵ_{MAX} <0.1%) to the reference solution generated by the constrained method. For a time split of 1, the results obtained are shown in Table 3.16. These results should be compared with a similar constrained case shown in Table 3.15. The unconstrained case gives power errors roughly half the size.

Figure 3.14 shows the variation of the reactivity over the macro time step (from 1.004 seconds to 1.19 seconds) for the constrained and unconstrained methods with a time split of 1 and the reference solution (tsplit=8). It is apparent that the effect

Figure 3.14: AECL: Variation of Reactivity During a Macro Time Step

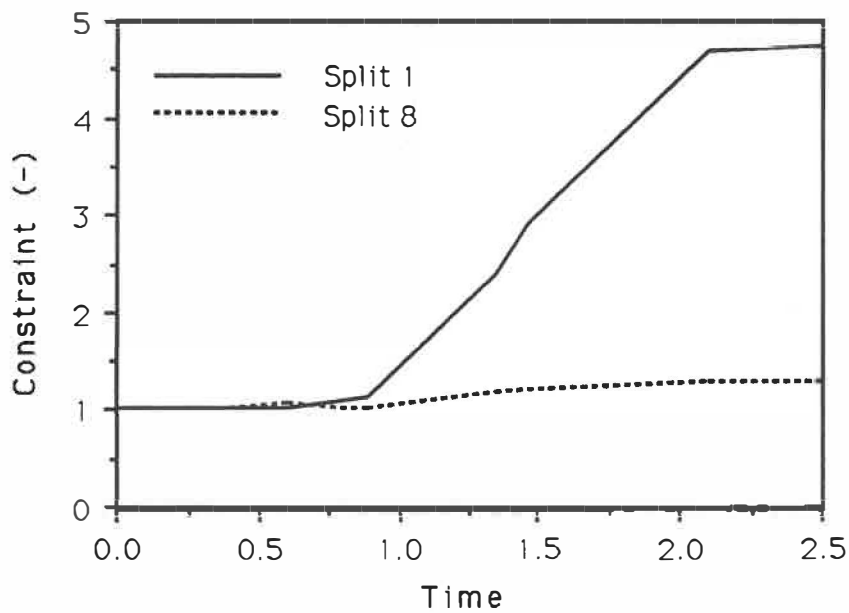


of the constraint is to reduce the non linearity in the reactivity for large time steps. For small time steps (msplit=8), the CERBERUS and XSTATIC reactivity predictions are very close.

The variation of the constraint with time is shown in Figure 3.15 for a time split of 1 and 8. When the time step is small (time split =8), the variation in the constraint is also small. It is probably true that given a small enough time step the constraint would remain constant. For larger time steps, the increase in the constraint is more dramatic. In both cases, the constraint increases during the rod insertion and is flat on either side.

From the above discussion, it is obvious that the effect of the floating constraint is to reduce the curvature or cusping of the reactivity within a macro time step. Mathematically, the effect of an increasing constraint can be interpreted by substituting in 2.69, $\alpha\vec{\varphi}$ for $\vec{\varphi}$ and taking the limit as $\alpha \rightarrow \infty$ giving:

Figure 3.15: AECL: Variation of Constraint



$$\lim_{\alpha\to\infty} \frac{\rho}{\Lambda}(t) = \frac{\vec{W}^t\left[f(\mathbf{A}(t_n)+\mathbf{B}(t_n))+(1-f)(\mathbf{A}(t_{n-1})+\mathbf{B}(t_{n-1})\right]\vec{\varphi}(t_{n-1})}{\vec{W}^t\mathbf{V}^{-1}\vec{\varphi}(t_n)} \tag{3.17}$$

The variation of $\frac{\rho}{\Lambda}$ is linear over the macro time step except near t_{n-1}. The extra degree of freedom allowed by floating the constraint has been used by the code to correct the reactivity transient.

3.4.5 Galerkin Weighting

This test case was rerun with a time split of 1 and Galerkin weighting. That is, the steady-state flux instead of adjoint weighting is used in the computation of the point kinetics parameters. The results for this case are presented in Table 3.17 and are to be compared to Table 3.15. The errors are similar with the Galerkin weighting results being slightly more accurate, and indicate that changing the weighting function does not significantly change the results. However, the reactivity and generation time transient is significantly different as can be seen in Figure 3.16 and 3.17. For example the peak reactivity for the Galerkin weighting is 7.32 mk at 0.4 seconds, and for adjoint weighting it is 6.31 mk at 0.715 seconds. Both codes predicts the same results even though the peak reactivity differs by 1.01 mk and occur at a times differing by 0.3 seconds.

If the time split is increased to 8, the Galerkin results agree with the reference result to within 0.1%. The reactivity transient remains unchanged as the time split is increased. Table 3.18 shows the value of the amplitude function at different times in the transient for Galerkin and adjoint weighting. Surprisingly they are almost identical. This indicates that although the reactivity transient is different the other point kinetics parameter are also different and yield the same solution.

Consider the following problem. The IQS method is used to solve a given transient for which the Δt is small. Remember that the IQS method is exact for sufficiently small Δt, regardless of the weight function. The reactivity can be

Figure 3.16: AECL: Generation Time versus Time for Galerkin Weighting

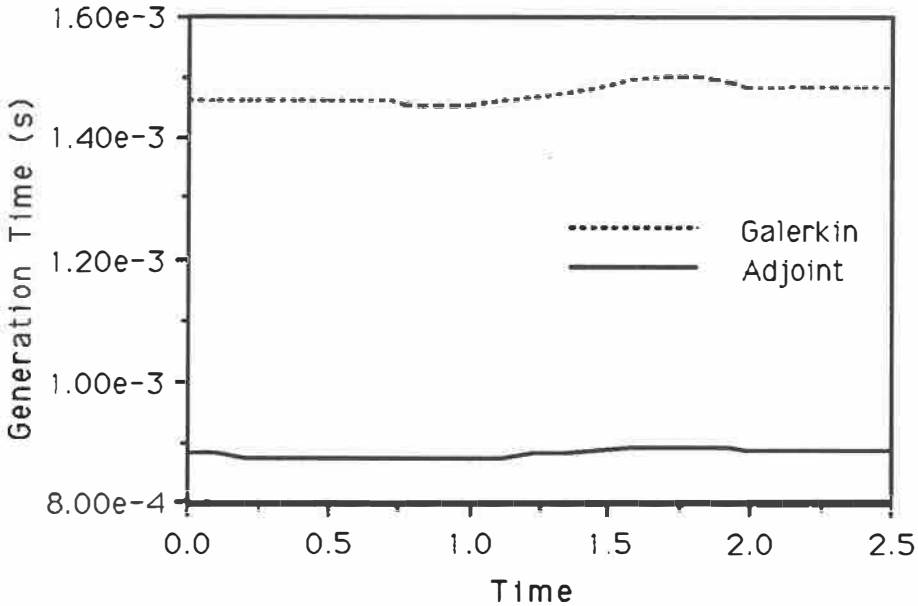


Figure 3.17: AECL: Reactivity versus Time for Galerkin Weighting

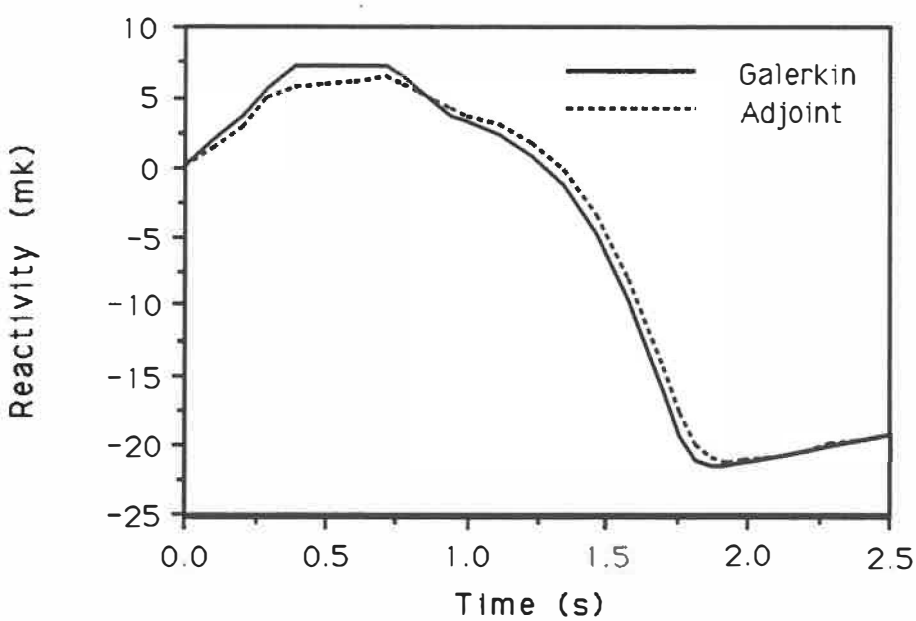


Table 3.17: AECL: Constrained XSTATIC Results with Galerkin Weighting (tsplit= msplit=1)

Time (s)	ε_{TP} %	ε_{AV} %	ε_{MAX} %
0.00	-0.00	0.00	0.00
0.20	-0.08	0.11	0.39
0.60	-0.10	0.13	0.34
0.89	-0.76	0.43	0.98
1.35	-6.26	3.40	7.07
1.47	-6.31	3.36	7.29
2.10	-1.85	0.89	2.24
2.50	-1.53	0.74	1.91

using XSTATIC reference

expressed as:

$$\frac{\rho}{\Lambda}(t) = \frac{\vec{W}^t(\mathbf{A}(t)+\mathbf{B}(t))\vec{\varphi}(t)}{\vec{W}^t\mathbf{V}^{-1}\vec{\varphi}(t)} = \frac{\vec{W}^t(\mathbf{A}(t)+\mathbf{B}(t))\vec{\phi}(t)}{\vec{W}^t\mathbf{V}^{-1}\vec{\phi}(t)} \tag{3.18}$$

The flux can be substituted in place of the shape function because the amplitude function can be divided out of the numerator and denominator. Since for both cases the flux solution is identical, the reactivity is obviously different. Therefore the reactivity transient is dependent on the weight function but the power transient is not. Reactivity is a mathematical quantity which is not observable and this apparent contradiction of 2 different reactivity transients for the same case is understandable.

3.4.6 Computational Time

The time the code spends in any routine is extremely dependent on the test case. For a test case with a small number of mesh points solving amplitude equations via (STIFFZ or FASTQ) requires a substantial fraction of the computational time. However as the number of mesh points increases the fraction of the time spent in solving the amplitude decreases and the fraction of the time spent solving the

Table 3.18: AECL: Amplitude Function versus Time for Adjoint and Galerkin Weighting

Time	Adjoint	Galerkin
0.00	1.000	1.000
0.20	1.198	—
0.60	2.812	2.811
0.89	3.692	3.682
1.35	1.920	1.927
1.47	1.375	1.384
2.10	0.338	0.340
2.50	0.321	.323

shape function equation increases. The time spent doing different calculations in the AECL test cases presented in Tables 3.14, 3.14 and 3.15 is shown in Table 3.19. The cases were executed on an APOLLO DN10000. The Table also contains a short description of the key inputs. For a typical production case (column 1) the code spends the vast majority (about 80%) of its effort solving the shape function equation and only about 5% solving the point kinetics equation. The same is true of the floating constraint (column 2) case which executes slightly more quickly (4% faster). If the time step is divided by 8, the total CPU time increases by a factor of 5.5. The factor is less than 8, because the number of outer (point kinetics and shape) iterations is less when the time step is smaller. Also the number of iterations per shape function computation is also smaller, largely due to a smaller time step reducing the spectral radius. However the total time is larger because more time steps were performed.

Comparing column 1 and 4 shows that using Galerkin weighting instead of adjoint weighting does slightly increase the computational time by about 10% but does not affect the computational accuracy (see previous section). For small time steps (tsplit=8), using Galerkin weighting has little effect on speed or accuracy.

Using DGEAR instead of FASTQ increases the time spent in solving the point

Table 3.19: AECL Computational Time for the IQS Method

Case Description					
Time Split	1	1	8	1	8
Mesh Split	1	1	1	1	1
Constrained	yes	no	yes	yes	yes
Method	IQS	IQS	IQS	IQS	IQS
Weight Function	Adjoint	Adjoint	Adjoint	Galerkin	Galerkin
Maximum Error	8.0%	4.8%	<0.5%	7.2%	<0.5%
CPU Time					
Area	Time	Time	Time	Time	Time
Shape Function	81%	80%	77%	82%	52%
(CPU s)	972	933	3914	1069	3990
Point Kinetics	4%	5%	2%	5%	2%
Assemble Matrices	3%	4%	6%	3%	6%
Other	12%	11%	15%	10%	15%
Total (CPU s)	1200	1167	5066	1290	5100

kinetics equations from 98 s (4% of total time) to 695 s (35% of the total time). This comparison is not very good because is it difficult to adjust the convergence criteria to ensure that both results have roughly the same error.

CHAPTER 4

Generalized Quasistatic Method

4.1 Overview

The generalized quasistatic method factors the flux into a shape function and an energy dependent amplitude function, ie one amplitude function for each energy group. This is in contrast to the improved quasistatic method where only one amplitude function is used for all energy groups. Hopefully this more consistent treatment given by the generalized quasistatic (GQS) method will reduce substantially the time needed to solve the shape function equation. However using an energy dependent amplitude function causes several problems: the generalized point kinetics equations are very stiff and the programming is more cumbersome because many previously scalar variables, such as reactivity, are now matrices.

Most of the work, in the area of the generalized quasistatic (GQS) method, as in the IQS method, involves a formulation where a constraint or a normalization is applied to each energy group of the multigroup shape function. For completeness the derivation of the unconstrained (or floating constraint) version of the GQS method will be presented. The same theoretical basis used to derive the floating constraint IQS equations is used for the GQS method. Unlike the presentation in chapter 2, only the constrained version of the GQS method will be used to solve time-dependent neutron diffusion equation.

As for the improved quasistatic method, the resulting equations are then discretized using a mixed nodal collocation - finite-element method for the shape function and finite-differences for the amplitude function. A fixed-point iteration scheme is used to simultaneously solve the shape and amplitude equations. The

same acceleration strategy that was applied to the IQS shape function equation will be applied to the GQS shape equations.

4.2 Derivation of the Unconstrained Generalized Quasistatic Method

The starting point is the time-dependent neutron diffusion equation which is repeated here for completeness:

$$\begin{aligned} \mathbf{V}^{-1}\frac{\partial}{\partial t}\vec{\phi}(\hat{r},t) &= \left[\nabla\cdot\mathbf{D}(\hat{r},t)\nabla - \mathbf{S}(\hat{r},t) + (1-\beta)\vec{\chi}_p\vec{F}^T(\hat{r},t)\right]\vec{\phi}(\hat{r},t) \\ &+ \sum_{i=1}^{N}\vec{\chi}_i\lambda_i C_i(\hat{r},t) \qquad (4.1) \\ \frac{\partial}{\partial t}\vec{\chi}_i C_i(\hat{r},t) &= \beta_i\vec{\chi}_i\vec{F}^T(\hat{r},t)\vec{\phi}(\hat{r},t) - \lambda_i\vec{\chi}_i C_i(\hat{r},t) \qquad (4.2) \end{aligned}$$

The conceptual basis of the quasistatic method is the factorization of the flux $\vec{\phi}(\hat{r},t)$ into the shape function $\vec{\varphi}(\hat{r},t)$ and the amplitude function $\mathbf{T}(t)$. Mathematically, it can be represented in two equivalent forms for a two-energy-group case:

$$\vec{\phi}(\hat{r},t) = \mathbf{T}(t)\vec{\varphi}(\hat{r},t) = \begin{pmatrix} T_1(t) & 0 \\ 0 & T_2(t) \end{pmatrix}\begin{pmatrix} \varphi_1(\hat{r},t) \\ \varphi_2(\hat{r},t) \end{pmatrix} \qquad (4.3)$$

$$= \varphi(\hat{r},t)\vec{T}(t) = \begin{pmatrix} \varphi_1(\hat{r},t) & 0 \\ 0 & \varphi_2(\hat{r},t) \end{pmatrix}\begin{pmatrix} T_1(t) \\ T_2(t) \end{pmatrix} \qquad (4.4)$$

where:

$\vec{\phi}_i(\hat{r},t)$	is the flux for the ith energy group	$\in X$
$\vec{\varphi}_i(\hat{r},t)$	is the shape function for the ith energy group	$\in X$
$T_i(t)$	is the amplitude function for the ith energy group	$\in R$

Both these factorization definitions are needed to derive the GQS equation: the first definition is used to derive the shape equation and the second is used to obtain the generalized amplitude equations.

As stated in chapter 2, these factorizations permit the division of a large stiff system into a large regular system (the shape function equation), and a stiff system (the generalized point kinetics equation). The temporal dependence of the slowly varying shape equation is discretized using a low order approximation with a large time step. A high order method with a small time step is used to discretize the amplitude equation. Since the spatial aspect of the calculation is the most time consuming, a reduction in the frequency of its computation should result in faster execution times.

The derivation of the generalized shape function and amplitude functions using the GQS factorisation of the flux is similar to the IQS derivation of chapter 2. Accordingly, only the main points of the derivation will be presented in this chapter and differences will be pointed out as they occur.

The shape function equation is obtained by substituting the factored flux $\mathbf{T}(t)\vec{\varphi}(\hat{r},t)$ into the time-dependent flux equation 4.1, giving:

$$\begin{aligned}\mathbf{V}^{-1}\frac{\partial}{\partial t}\left[\mathbf{T}(t)\vec{\varphi}(\hat{r},t)\right] &= \left[\nabla\cdot\mathbf{D}(\hat{r},t)\nabla-\mathbf{S}(\hat{r},t)+(1-\beta)\vec{\chi}_p\vec{F}^t(\hat{r},t)\right]\mathbf{T}(t)\vec{\varphi}(\hat{r},t)\\ &+ \sum_{i=1}^{N}\vec{\chi}_i\lambda_i C_i(\hat{r},t) \qquad (4.5)\end{aligned}$$

Similarly the factored precursor equation is obtained by substituting the factored flux into 4.2, yeilding:

$$\frac{\partial}{\partial t}\vec{\chi}_i C_i(\hat{r},t)=\beta_i\vec{\chi}_i\vec{F}^T(\hat{r},t)\mathbf{T}(t)\vec{\varphi}(\hat{r},t)-\lambda_i\vec{\chi}_i C_i(\hat{r},t) \qquad (4.6)$$

Define a functional $\vec{\mathbf{I}}$ that maps X onto R^2 to be :

$$\vec{\mathbf{I}}\{\vec{\Omega}\}=\int\mathbf{W}\vec{\Omega}(\hat{r},t)d^3r=\left\langle\mathbf{W},\vec{\Omega}(\hat{r},t)\right\rangle \qquad (4.7)$$

where $\mathbf{W}$ is independent of time and the Dirac notation $<,>$ is used to express only integration over the volume of the reactor. The functional used in the IQS method sums over the energy groups.

The generalized point kinetic equations are obtained by applying this functional to equation 4.5 with the flux factored as $\varphi(\hat{r},t)\vec{T}(t)$, the generalized point kinetics parameters are then obtained. After applying the product rule adding and subtracting the term representing the delayed neutron production, the following equation is obtained:

$$\frac{d}{dt}\left\langle \mathbf{W}, \mathbf{V}^{-1}\varphi(\hat{r},t)\right\rangle \vec{T}(t) + \left\langle \mathbf{W}, \mathbf{V}^{-1}\varphi(\hat{r},t)\right\rangle \frac{d}{dt}\vec{T}(t) = \tag{4.8}$$
$$\left\langle \mathbf{W}, \left[\nabla\cdot\mathbf{D}(\hat{r},t)\nabla - \mathbf{S}(\hat{r},t) + \left\{(1-\beta)\vec{\chi}_p + \sum_{i=1}^{N}\beta_i\vec{\chi}_i\right\}\vec{F}^t(\hat{r},t)\right]\varphi(\hat{r},t)\right\rangle \vec{T}(t)$$
$$-\left\langle \mathbf{W}, \sum_{i=1}^{N}\beta_i\vec{\chi}_i\vec{F}^t(\hat{r},t)\varphi(\hat{r},t)\right\rangle \vec{T}(t) + \sum_{i=1}^{N}\left\langle \mathbf{W}, \vec{\chi}_i\lambda_i C_i(\hat{r},t)\right\rangle$$

A similar procedure is also used to transform the factored precursor equation into the point kinetics precursor equation. Applying the functional l to the equation 4.6 with the flux factored as $\varphi(\hat{r},t)\vec{T}(t)$ and multiplying by $\langle \mathbf{W}, \mathbf{V}^{-1}\varphi(\hat{r},t)\rangle^{-1}$ and using $\frac{d}{dt}\mathbf{J}^{-1} = -\mathbf{J}^{-1}\left(\frac{d}{dt}\mathbf{J}\right)\mathbf{J}^{-1}$ [1] to evaluate the derivative of the inverse of a matrix gives the following precursor equation:

$$\left\langle \mathbf{W}, \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle^{-1}\left\langle \mathbf{W}, \frac{\partial}{\partial t}\vec{\chi}_i C_i(\hat{r},t)\right\rangle = \tag{4.9}$$
$$\beta_i\left\langle \mathbf{W}, \mathbf{V}^{-1}\varphi(\hat{r},t)\right\rangle^{-1}\left\langle \mathbf{W}, \vec{\chi}_i\vec{F}^t\varphi(\hat{r},t)\right\rangle$$
$$-\left(\lambda_i\mathbf{I} + \left\langle \mathbf{W}, \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle^{-1}\frac{\partial}{\partial t}\left\langle \mathbf{W}, \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle\right)\times$$
$$\left\langle \mathbf{W}, \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle^{-1}\left\langle \mathbf{W}, \vec{\chi}_i C_i(\hat{r},t)\right\rangle$$

Grouping the terms similar to the point kinetic parameters in equation 4.8 and 4.9 equations, yields the point kinetic equations:

$$\frac{d}{dt}\vec{T}(t) = \left(\frac{\boldsymbol{\rho}}{\boldsymbol{\Lambda}}(t) - \frac{\boldsymbol{\beta}}{\boldsymbol{\Lambda}}(t) - \boldsymbol{\lambda}_s(t)\right)\vec{T}(t) + \sum_{i=1}^{N}\lambda_i\vec{\xi}_i(t) \tag{4.10}$$

[1] Using $\frac{d}{dt}\mathbf{I} = 0 = \frac{d}{dt}\left(\mathbf{J}^{-1}\mathbf{J}\right) = \frac{d}{dt}\left(\mathbf{J}^{-1}\right)\mathbf{J} + \mathbf{J}^{-1}\frac{d}{dt}\mathbf{J}$ and then solving for $\frac{d}{dt}\mathbf{J}^{-1}$ gives the quoted rule.

$$\frac{d}{dt}\vec{\xi}_i(t) = \frac{\boldsymbol{\beta}_i}{\boldsymbol{\Lambda}}(t)\vec{T}(t) - (\lambda_i\mathbf{I} + \boldsymbol{\lambda}_s(t))\,\vec{\xi}_i(t) \tag{4.11}$$

The generalized point kinetics parameters, which are G $\times$ G matrices, are:

$$\frac{\boldsymbol{\rho}}{\boldsymbol{\Lambda}}(t) = \left\langle \mathbf{V}^{-1}\boldsymbol{\varphi}(\hat{r},t)\right\rangle^{-1} \langle \mathbf{W}\ ,\ [\nabla\cdot\mathbf{D}(\hat{r},t)\nabla - \mathbf{S}(\hat{r},t) + \left\{(1-\beta)\vec{\chi}_p + \sum_{i=1}^{N}\beta_i\vec{\chi}_i\right\}\vec{F}^t(\hat{r},t)\Big]\,\boldsymbol{\varphi}(\hat{r},t)\Big\rangle \tag{4.12}$$

$$\tag{4.13}$$

$$\frac{\boldsymbol{\beta}_i}{\boldsymbol{\Lambda}}(t) = \left\langle \mathbf{W}, \mathbf{V}^{-1}\boldsymbol{\varphi}(\hat{r},t)\right\rangle^{-1} \left\langle \mathbf{W}\ ,\ \vec{\chi}_i\vec{F}^t(\hat{r},t)\boldsymbol{\varphi}(\hat{r},t)\right\rangle \tag{4.14}$$

$$\frac{\boldsymbol{\beta}}{\boldsymbol{\Lambda}}(t) = \sum_{i=1}^{N}\frac{\boldsymbol{\beta}_i}{\boldsymbol{\Lambda}}(t) \tag{4.15}$$

$$\boldsymbol{\lambda}_s(t) = \left\langle \mathbf{W}, \mathbf{V}^{-1}\boldsymbol{\varphi}(\hat{r},t)\right\rangle^{-1} \frac{\partial}{\partial t}\left\langle \mathbf{W}\ ,\ \mathbf{V}^{-1}\boldsymbol{\varphi}(\hat{r},t)\right\rangle \tag{4.16}$$

$$\vec{\xi}_i(t) = \left\langle \mathbf{W}, \mathbf{V}^{-1}\boldsymbol{\varphi}(\hat{r},t)\right\rangle^{-1} \langle \mathbf{W}\ ,\ \vec{\chi}_i C_i(\hat{r},t)\rangle \tag{4.17}$$

Note that the generation time $\boldsymbol{\Lambda}$ is defined only for the IQS formulation. By analogy with the IQS derivation, this quantity could exist in the GQS matrix formulation only if $\left\langle \mathbf{W}, \vec{\chi}_i\vec{F}^t\boldsymbol{\varphi}(\hat{r},t)\right\rangle$ is nonsingular. However in this derivation $\boldsymbol{\Lambda}$ is a dyadic product, and is by definition singular.

In a further attempt to save some computational effort, equation 4.2 is integrated analytically. The result is a convolution which can be written as:

$$\vec{\chi}C_i(\hat{r},t) = \beta_i\int_{t_0}^{t}\vec{\chi}\vec{F}^t(\hat{r},\tau)\vec{\phi}(\hat{r},\tau)e^{(-\lambda_i(t-\tau))}d\tau + \vec{\chi}C_i(\hat{r},t_0)e^{(-\lambda_i(t-t_0))} \tag{4.18}$$

In summary, the GQS equations (4.10 to 4.18 and 4.5) must be solved simultaneously.

4.3 Reduced Two-Energy-Group Formulation of Quasistatic Equations

The assumptions for the reduced two-energy-group formulation are given in section 2.4. Applying these to the generalized quasistatic formulation gives the following equations:

Reduced generalized point kinetics equations:

$$\frac{d}{dt}\vec{T}(t) = \left(\frac{\boldsymbol{\rho}}{\boldsymbol{\Lambda}}(t) - \frac{\boldsymbol{\beta}}{\boldsymbol{\Lambda}}(t) - \boldsymbol{\lambda}_s(t)\right)\vec{T}(t) + \sum_{i=1}^{N}\lambda_i\vec{\xi}_i(t) \tag{4.19}$$

$$\frac{d}{dt}\vec{\xi}_i(t) = \frac{\boldsymbol{\beta}_i}{\boldsymbol{\Lambda}}(t)\vec{T}(t) - (\lambda_i\mathbf{I} + \boldsymbol{\lambda}_s(t))\,\vec{\xi}_i(t) \tag{4.20}$$

Reduced generalized point kinetics parameters:

$$\frac{\boldsymbol{\rho}}{\boldsymbol{\Lambda}}(t) = \left\langle \mathbf{W}, \mathbf{V}^{-1}\varphi(\hat{r},t)\right\rangle^{-1}\langle \mathbf{W}(\hat{r})\ ;\ [\mathbf{A}(\hat{r},t) + \mathbf{B}(\hat{r},t)]\,\varphi(\hat{r},t)\rangle \tag{4.21}$$

$$\frac{\boldsymbol{\beta}_i}{\boldsymbol{\Lambda}}(t) = \left\langle \mathbf{W}, \mathbf{V}^{-1}\varphi(\hat{r},t)\right\rangle^{-1}\langle \mathbf{W}(\hat{r})\ ,\ \beta_i\mathbf{B}(\hat{r},t)\varphi(\hat{r},t)\rangle \tag{4.22}$$

$$\frac{\boldsymbol{\beta}}{\boldsymbol{\Lambda}}(t) = \sum_{i=1}^{N}\frac{\boldsymbol{\beta}_i}{\boldsymbol{\Lambda}}(t) \tag{4.23}$$

$$\vec{\xi}_i(t) = \left\langle \mathbf{W}, \mathbf{V}^{-1}\vec{\varphi}(\hat{r},t)\right\rangle^{-1}\langle \mathbf{W}(\hat{r})\ ,\ \vec{\chi}C_i(\hat{r},t)\rangle \tag{4.24}$$

$$\boldsymbol{\lambda}_s(t) = \left\langle \mathbf{W}, \mathbf{V}^{-1}\varphi(\hat{r},t)\right\rangle^{-1}\frac{\partial}{\partial t}\left\langle \mathbf{W}\ ,\ \mathbf{V}^{-1}\varphi(\hat{r},t)\right\rangle \tag{4.25}$$

Reduced generalized shape function equation:

$$\begin{aligned}\mathbf{V}^{-1}\frac{\partial}{\partial t}\vec{\varphi}(\hat{r},t) + \mathbf{T}^{-1}(t)\frac{d}{dt}\mathbf{T}\vec{\varphi}(\hat{r},t) &= \\ \mathbf{T}^{-1}(t)\,[\mathbf{A}(\hat{r},t) + (1-\beta)\mathbf{B}(\hat{r},t)]\,\mathbf{T}(t)\vec{\varphi}(\hat{r},t) &+ \mathbf{T}^{-1}(t)\sum_{i=1}^{N}\vec{\chi}\lambda_i C_i(\hat{r},t)\end{aligned} \tag{4.26}$$

Reduced generalized precursor equation:

$$\vec{\chi}C_i(\hat{r},t) = \beta_i\int_{t_0}^{t}\mathbf{B}(\hat{r},\tau)\vec{\phi}(\hat{r},\tau)e^{(-\lambda_i(t-\tau))}d\tau + \vec{\chi}C_i(\hat{r},t_0)e^{(-\lambda_i(t-t_0))} \tag{4.27}$$

4.4 Discretization of the Generalized Shape Function Equation

The discretization of the generalized equations is similar to that of the IQS equations in chapter 2. Only the differences will be highlighted in this section.

The spatial variable r is discretized using the nodal collocation method[13]. The effect of discretization can be expressed as follows:

$$\begin{aligned} \text{continuous} &\Rightarrow \text{discrete} \\ \mathbf{a}(\hat{r},t)\vec{\varphi}(\hat{r},t) &\Rightarrow \mathbf{a}(t)\vec{\varphi}(t) \\ (G\times G)(G\times 1) &\Rightarrow (GK^3\times GK^3)(GK^3\times 1) \end{aligned}$$

The temporal variable is discretized by a finite-element like approach. The shape function $\vec{\varphi}(\hat{r},t)$ is assumed to have a linear variation over the interval, that is:

$$\vec{\varphi}(t) = f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1}) \tag{4.28}$$

$$f = \frac{t-t_{n-1}}{t_n - t_{n-1}} \tag{4.29}$$

For ramp perturbations of the nuclear cross sections, the dependence of $\mathbf{A}$ and $\mathbf{B}$ with time are:

$$\mathbf{A}(t) = f\mathbf{A}(t_n) + (1-f)\mathbf{A}(t_{n-1}) \tag{4.30}$$

$$\mathbf{B}(t) = f\mathbf{B}(t_n) + (1-f)\mathbf{B}(t_{n-1}) \tag{4.31}$$

Similarly, for step perturbations of the nuclear cross sections, the dependence of $\mathbf{A}$ and $\mathbf{B}$ with time are:

$$\mathbf{A}(t) = \mathbf{A}(t_n) \tag{4.32}$$

$$\mathbf{B}(t) = \mathbf{B}(t_n) \tag{4.33}$$

The discretized shape function equation obtained by substituting equation 4.28 into equation 4.26 and evaluating at t_n is:

$$\mathbf{V}^{-1}\frac{\vec{\varphi}(t_n)-\vec{\varphi}(t_{n-1})}{\Delta t}+\mathbf{V}^{-1}\mathbf{T}^{-1}(t_n)\frac{d}{dt}\mathbf{T}(t_n)\vec{\varphi}(t_n) = \tag{4.34}$$

$$\mathbf{T}^{-1}(t_n)\left[\mathbf{A}(t_n)+(1-\beta)\mathbf{B}(t_n)\right]\mathbf{T}(t_n)\vec{\varphi}(t_n) + \mathbf{T}^{-1}(t_n)\sum_{i=1}^{N}\chi\lambda_i\vec{C}_i(t_n)$$

4.5 Discretization of Generalized Point Kinetics Equation

The discretization of the point kinetics equation uses the second factorization presented in equation 4.4. The spatial discretization of the factored flux can be expressed as follows:

$$\begin{aligned}
\text{continuous} &\Rightarrow \text{discrete} \\
\varphi(\hat{r},t)\vec{T}(t) &\Rightarrow \varphi(t)\vec{T}(t) && (4.35)\\
(G\times G)(G\times 1) &\Rightarrow (GK^3\times G)(G\times 1) && (4.36)\\
&\Rightarrow \begin{pmatrix}\vec{\varphi}_1(t) & 0\\ 0 & \vec{\varphi}_2(t)\end{pmatrix}\begin{pmatrix}T_1(t)\\ T_2(t)\end{pmatrix} && (4.37)
\end{aligned}$$

The discretization of point kinetics parameters is more complex. These parameters are the product of G × G matrices whose elements are bilinear products. The discretization of a fictitious bilinear product involving operator **P** is expressed as:

$$\begin{aligned}
\text{continuous} &\Rightarrow \text{discrete} \\
\langle\mathbf{W}(\hat{r})\ ,\ \mathbf{P}(\hat{r},t)\varphi(\hat{r},t)\rangle &\Rightarrow \mathbf{W}^t\mathbf{P}(t)\ \varphi(t) && (4.38)\\
&\Rightarrow \begin{pmatrix}W_1^t & 0\\ 0 & W_2^t\end{pmatrix}\begin{pmatrix}\mathbf{P}_{11} & \mathbf{P}_{12}\\ \mathbf{P}_{21} & \mathbf{P}_{22}\end{pmatrix}\begin{pmatrix}\vec{\varphi}_1 & 0\\ 0 & \vec{\varphi}_2\end{pmatrix} && (4.39)\\
(G\times G)(G\times G)(G\times G) &\Rightarrow (G\times GK^3)(GK^3\times GK^3)(GK^3\times G) && (4.40)
\end{aligned}$$

The shape function is assumed to have the following linear variation over the interval:

$$\varphi(t) = f\ \varphi(t_n)+(1-f)\ \varphi(t_{n-1}) \tag{4.41}$$

The above equation is identical to equation 4.28, except that $\vec{\varphi}$ is now a matrix.

Substituting equations 4.41, 4.30 and 4.31 into the point kinetic parameter equations (4.21 through 4.25) gives the following discretized equations:

$$\begin{aligned}
\frac{\boldsymbol{\rho}}{\boldsymbol{\Lambda}}(t) &= \left\{\mathbf{W}\mathbf{V}^{-1}\mathbf{U}\left[f\ \boldsymbol{\varphi}(t_n) + (1-f)\ \boldsymbol{\varphi}(t_{n-1})\right]\right\}^{-1} && (4.42)\\
&\quad \mathbf{W}\left[f(\mathbf{A}(t_n) + \mathbf{B}(t_n)) + (1-f)(\mathbf{A}(t_{n-1} + \mathbf{B}(t_{n-1})\right] \times \\
&\quad \left[f\ \boldsymbol{\varphi}(t_n) + (1-f)\ \boldsymbol{\varphi}(t_{n-1})\right] \\
\frac{\boldsymbol{\beta}_i}{\boldsymbol{\Lambda}}(t) &= \beta_i \left\{\mathbf{W}\mathbf{V}^{-1}\mathbf{U}\left[f\ \boldsymbol{\varphi}(t_n) + (1-f)\ \boldsymbol{\varphi}(t_{n-1})\right]\right\}^{-1} && (4.43)\\
&\quad \mathbf{W}\left[f\mathbf{B}(t_n) + (1-f)\mathbf{B}(t_{n-1})\right]\left[f\ \boldsymbol{\varphi}(t_n) + (1-f)\ \boldsymbol{\varphi}(t_{n-1})\right] \\
\frac{\boldsymbol{\beta}}{\boldsymbol{\Lambda}}(t) &= \sum_{i=1}^{N} \frac{\boldsymbol{\beta}}{\boldsymbol{\Lambda}} i(t) && (4.44)\\
\vec{\xi}_i(t) &= \left\{\mathbf{W}\mathbf{V}^{-1}\mathbf{U}\left[f\ \boldsymbol{\varphi}(t_n) + (1-f)\ \boldsymbol{\varphi}(t_{n-1})\right]\right\}^{-1}\ \mathbf{W}\ \boldsymbol{\chi}\vec{C}_i(t) && (4.45)\\
\boldsymbol{\lambda}_s(t) &= \left\langle \mathbf{W}, \mathbf{V}^{-1}\mathbf{U}\boldsymbol{\varphi}(\hat{r},t)\right\rangle^{-1} \frac{\partial}{\partial t}\left\langle \mathbf{W}\ ,\ \mathbf{V}^{-1}\mathbf{U}\boldsymbol{\varphi}(\hat{r},t)\right\rangle && (4.46)
\end{aligned}$$

The point kinetics or amplitude equation is independent of the spatial variable. Therefore, as stated earlier, these equations are discretized using a finite-difference method. Unfortunately this system of equations is very stiff and consequently the Gear's integration algorithm[14] is used.

4.6 Discretization of Precursor Equation

Using the technique applied to the shape function equation, the discretized precursor equation can be written as:

$$\boldsymbol{\chi}\vec{C}_i(t_n) = \beta_i \int_{t_{n-1}}^{t_n} \mathbf{B}(\tau)\vec{\phi}(\tau)e^{(-\lambda_i(t_n-\tau))}d\tau + \ \boldsymbol{\chi}\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n-t_{n-1}))} \qquad (4.47)$$

For ramp perturbations, this equation will be expanded using equations 4.28 and 4.31, giving:

$$\begin{aligned}\chi\vec{C}_i(t_n) &= \beta_i \int_{t_{n-1}}^{t_n} (f\mathbf{B}(t_n) + (1-f)\mathbf{B}(t_{n-1}))\mathbf{T}(t) \times \\ &\quad (f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1}))e^{(-\lambda_i(t-\tau))}d\tau \\ &+ \quad \chi\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n-t_{n-1}))} \end{aligned} \tag{4.48}$$

Multiplying out the product in the integral and performing some manipulation, the following is obtained:

$$\begin{aligned}\chi\vec{C}_i(t_n) &= \beta_i(\mathbf{B}(t_n)\ \gamma_{i,3}\vec{\varphi}(t_n) + \mathbf{B}(t_n)\ \gamma_{i,2}\vec{\varphi}(t_{n-1}) + \\ &\quad \mathbf{B}(t_{n-1})\ \gamma_{i,2}\vec{\varphi}(t_n) + \mathbf{B}(t_{n-1})\ \gamma_{i,1}\vec{\varphi}(t_{n-1})) + \\ &\quad \chi\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n-t_{n-1}))} \end{aligned} \tag{4.49}$$

where

$$\gamma_{i,k} = diag\{\gamma_{i,j,k}\} \tag{4.50}$$

$$\gamma_{i,3,k} = \int_{t_{n-1}}^{t_n} f(\tau)f(\tau)T_k(\tau)e^{-\lambda_i(t_n-\tau)}d\tau \tag{4.51}$$

$$\gamma_{i,2,k} = \int_{t_{n-1}}^{t_n} f(\tau)[1-f(\tau)]T_k(\tau)e^{-\lambda_i(t_n-\tau)}d\tau \tag{4.52}$$

$$\gamma_{i,1,k} = \int_{t_{n-1}}^{t_n} [1-f(\tau)][1-f(\tau)]T_k(\tau)e^{-\lambda_i(t_n-\tau)}d\tau \tag{4.53}$$

for k=1,2.

Performing the same calculation for a step perturbation, by substituting 4.28 and 4.33 into 4.47 gives equation 4.49 where the γ's are now defined as:

$$\gamma_{i,3,k} = \int_{t_{n-1}}^{t_n} [1-f(\tau)]T_k(\tau)e^{-\lambda_i(t_n-\tau)}d\tau \tag{4.54}$$

$$\gamma_{i,2,k} = 0 \tag{4.55}$$

$$\gamma_{i,1,k} = \int_{t_{n-1}}^{t_n} f(\tau)T_k(\tau)e^{-\lambda_i(t_n-\tau)}d\tau \tag{4.56}$$

4.7 Generalized Quasistatic Solution Strategy

The shape function and amplitude function equations form a system of nonlinear equations. Although $\mathbf{T}$ is a matrix the iterative scheme can still be written as:

$$\vec{\varphi}(\hat{r},t) = \mathbf{g}(\mathbf{T}(t)) \tag{4.57}$$

$$\mathbf{T}(t) = \mathbf{h}(\vec{\varphi}(\hat{r},t)) \tag{4.58}$$

The same fixed-point iterative technique, used in chapter 2, will be used to solve these nonlinear equations, which consists of iterating on the shape function

$$\vec{\varphi}^{(l+1)}(\hat{r},t) = \mathbf{g}(\mathbf{h}(\vec{\varphi}^{(l)}(\hat{r},t))) \tag{4.59}$$

until the error between two consecutive iterations is smaller than a given convergence criterion.

The GQS algorithm is the same as IQS algorithm and is given in Figure 4.1 for completeness. However, only the constrained version is implemented in XSTATIC. Consequently λ_s is zero and no longer appears in any of the equations in this section. The constraint is imposed in the FLXGQS subroutine and is explained in the next section

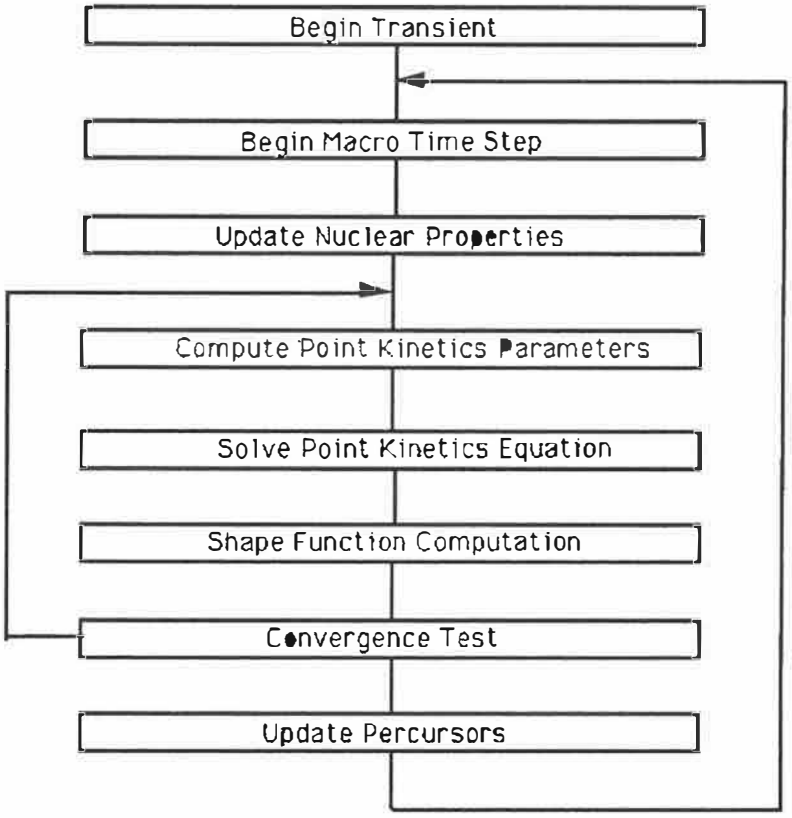
If the shape function solution step is skipped, the shape function used through out the transient is the steady state flux and the GQS method reduces to generalized point kinetics. As before a special keyword (PKIN) tells the program to execute without updating the shape function.

Since PHASE4 is unchanged from chapter 2 it will not be discussed here.

4.7.1 Shape Function Algorithm - FLXGQS

The module FLXGQS solves the generalized shape function equation. The shape function equation expressed in equation 4.34 must be solved, and again some mathematical manipulations are still necessary. The term involving $\vec{C}_i(t)$ is an implicit function of $\vec{\varphi}(t)$. Equation 4.48 is substituted in 4.34 to render this term explicit.

Figure 4.1: Generalized QUASISTATIC Algorithm



$$\mathbf{V}^{-1}\mathbf{U}\frac{\vec{\varphi}(t_n)-\vec{\varphi}(t_{n-1})}{\Delta t}+\mathbf{V}^{-1}\mathbf{T}^{-1}(t_n)\frac{d}{dt}\mathbf{T}(t_n)\vec{\varphi}(t_n) = \tag{4.60}$$
$$\mathbf{T}^{-1}(t_n)\left[\mathbf{A}(t_n)+(1-\beta)\mathbf{B}(t_n)\right]\mathbf{T}(t_n)\vec{\varphi}(t_n)$$
$$+\mathbf{T}^{-1}(t_n)\sum_{i=1}^{N}\lambda_i\beta_i\Big\{\mathbf{B}(t_n)\ \gamma_{i,3}\vec{\varphi}(t_n)+\mathbf{B}(t_n)\ \gamma_{i,2}\vec{\varphi}(t_{n-1})+\mathbf{B}(t_{n-1})\ \gamma_{i,2}\vec{\varphi}(t_n)+$$
$$\mathbf{B}(t_{n-1})\ \gamma_{i,1}\vec{\varphi}(t_{n-1})\Big\}+\mathbf{T}^{-1}(t_n)\ \chi\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n-t_{n-1}))}$$

Rewriting this equation with the $\vec{\varphi}(t_{n-1})$ and $\vec{C}_i(t_{n-1})$ terms on the right hand side yields:

$$\mathbf{H}(t_n)\vec{\varphi}^{(k)}(t_n)=\vec{S} \tag{4.61}$$

where

$$\begin{aligned}\mathbf{H}(t_n) &= \Big[\mathbf{T}^{-1}(t_n)\left[\mathbf{A}(t_n)+(1-\beta)\mathbf{B}(t_n)\right]\mathbf{T}(t_n)+\mathbf{T}^{-1}(t_n)\mathbf{B}(t_n)\ \gamma_3\\ &+ \mathbf{T}^{-1}(t_n)\mathbf{B}(t_{n-1})\ \gamma_2+\mathbf{V}^{-1}\left\{\frac{1}{\Delta t}-\mathbf{T}^{-1}(t_n)\frac{d}{dt}\mathbf{T}(t_n)\right\}\Big] \qquad (4.62)\\ \vec{S} &= \left[\mathbf{V}^{-1}\frac{1}{\Delta t}+\mathbf{T}^{-1}(t_n)\mathbf{B}(t_{n-1})\ \gamma_1+\mathbf{T}^{-1}(t_n)\mathbf{B}(t_n)\ \gamma_2\right]\vec{\varphi}(t_{n-1})\\ &+ \mathbf{T}^{-1}(t_n)\ \chi\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n-t_{n-1}))} \qquad (4.63)\\ \gamma_J &= \sum_{i=1}^{N}\lambda_i\beta_i\ \gamma_{i,J} \qquad (4.64)\end{aligned}$$

The shape function equation is solved using a preconditioned iterative strategy and variational acceleration. The techniques used here are identical to those described in section 2.9.2.

4.7.1.1 Application of the Constraint

For the constrained version of the GQS method, the constraint is imposed by multiplying the ith energy group of the converged shape function by a constant c_i defined as:

$$c_i=\frac{\vec{W}_i\mathbf{u}\vec{\varphi}_i(t_{n-1})}{\vec{W}_i\mathbf{u}\vec{\varphi}_i(t_n)} \tag{4.65}$$

The other GQS subroutines assume that the contraint has been imposed.

4.7.2 Generalized Point Kinetics Solution - DGEAR

The module DGEAR solves the point kinetics equations whose parameters were determined by the module TRANCA. The generalized point kinetics are typically defined by equations 4.21 through 4.25. These equation may be expressed in matrix form:

$$\frac{d}{dt}\vec{Z} = \mathbf{P}(t)\vec{Z} \tag{4.66}$$

$$\vec{Z} = \begin{pmatrix} T_1 \\ T_2 \\ \xi_1 \\ \vdots \\ \xi_N \end{pmatrix} \tag{4.67}$$

$$\mathbf{P}(t) = \begin{pmatrix} \frac{\rho}{\Lambda_{11}} - \frac{\beta}{\Lambda_{11}} & \frac{\rho}{\Lambda_{12}} - \frac{\beta}{\Lambda_{12}} & \lambda_1 & \cdots & \lambda_N \\ \frac{\rho}{\Lambda_{21}} - \frac{\beta}{\Lambda_{21}} & \frac{\rho}{\Lambda_{22}} - \frac{\beta}{\Lambda_{22}} & 0 & \cdots & 0 \\ \frac{\beta}{\Lambda_{11}} & \frac{\beta}{\Lambda_{11}} & -\lambda_1 & 0 & 0 \\ \vdots & 0 & & \ddots & 0 \\ \frac{\beta}{\Lambda_{N1}} & \frac{\beta}{\Lambda_{N2}} & 0 & 0 & -\lambda_N \end{pmatrix} \tag{4.68}$$

In the generalized quasistatic method, the γ_i terms, defined in equations 4.51 to 4.53, must also be solved along with the generalized point kinetics equation. This combined set or "augmented" point kinetics equation can be written in matrix form as:

$$\frac{d}{dt}\vec{Z} = \mathbf{Q}(t)\vec{Z} \tag{4.69}$$

$$\vec{Z} = \begin{pmatrix} T \\ \xi_1 \\ \vdots \\ \xi_N \\ \gamma_{1,1} \\ \vdots \\ \gamma_{N,3} \end{pmatrix} \tag{4.70}$$

$$\mathbf{Q}(t) = \begin{pmatrix} \frac{\rho}{\Lambda_{11}} - \frac{\beta}{\Lambda_{11}} & \frac{\rho}{\Lambda_{12}} - \frac{\beta}{\Lambda_{12}} & \lambda_1 & \cdots & \lambda_N & \\ \frac{\rho}{\Lambda_{21}} - \frac{\beta}{\Lambda_{21}} & \frac{\rho}{\Lambda_{22}} - \frac{\beta}{\Lambda_{22}} & 0 & \cdots & 0 & \\ \frac{\beta}{\Lambda_{11}} & \frac{\beta}{\Lambda_{11}} & -\lambda_1 & 0 & 0 & 0 \\ \vdots & 0 & & \ddots & 0 & \\ \frac{\beta}{\Lambda_{N1}} & \frac{\beta}{\Lambda_{N2}} & 0 & 0 & -\lambda_N & \\ e^{\lambda_1 \tau} & & & & & \\ \vdots & & & & & 0 \\ \tau^2 e^{\lambda_N \tau} & & & & & \end{pmatrix} \tag{4.71}$$

The introduction of the term "augmented" removes some of the ambiguity associated with the term point kinetics. This set of equations is solved using the canned ordinary differential integration routine DGEAR. This algorithm incorporates, into a sophisticated program, an Adams-Bashfort predictor and an Adams-Moulton corrector. A variable order method with variable step size regulates itself so as to minimize the computational effort required to achieve a given local error.

4.7.3 Precursor Update - PRECDY

The precursor concentrations are updated at the end of each macro time step by the PRECDY subroutine. For this purpose equation 4.48 is evaluated using the correct $\gamma_{i,j}$'s for either a step or ramp perturbation. For the sake of completeness the precursor equation is given again:

$$\begin{aligned} \chi\vec{C}_i(t_n) &= \beta_i \int_{t_{n-1}}^{t_n} (f\mathbf{B}(t_n) + (1-f)\mathbf{B}(t_{n-1}))\mathbf{T}(t) \times \\ &\quad (f\vec{\varphi}(t_n) + (1-f)\vec{\varphi}(t_{n-1}))e^{(-\lambda_i(t-\tau))}d\tau \\ &+ \chi\vec{C}_i(t_{n-1})e^{(-\lambda_i(t_n - t_{n-1}))} \end{aligned} \tag{4.72}$$

where the coefficients $\gamma_{i,j}$ are given by equation 4.51 to 4.53. They are computed simultaneously with the solution of the point kinetics parameters in the module PKIN.

4.7.4 Convergence Test

The fixed-point iterative method is considered to have converged if the shape function calculation has converged and the following convergence criteria have been satisfied:

$$T_i^{(l+1)}(t_n) - T_i^{(l)}(t_n) < \epsilon_T \text{ i=1,2} \tag{4.73}$$

$$\frac{E_i^{(l+1)}(t_n) - E_i^{(0)}(t_n)}{E_i^{(0)}(t_n)} < \epsilon_E \text{ i=1,2} \tag{4.74}$$

where:

		Typical Values
ϵ_T	amplitude convergence criterion	10^{-5}
ϵ_E	constraint convergence criterion	10^{-5}

Note that the reactivity is no longer used as a convergence criteria. These convergence criteria do not guarantee the validity of the solution, but rather guarentee a self-consistent solution to the equation.

Chapter 5

Numerical Results for the Generalized Quasistatic Method

In the previous chapter, the theoretical foundations of the generalized quasistatic method were derived. The objective of this chapter is to validate the GQS portion of the XSTATIC code and to determine if the use of energy dependent amplitude functions increases the performance of the code. Three of four test cases (SIMPLE, LMW, AECL) used to validate the IQS portion of XSTATIC are also used to validate the GQS portion. Results are given only for the constrained version of the generalized quasistatic method.

The GQS method is expected to either perform fewer shape function calculations (larger time step size) with the same accuracy, or perform the same number of shape function calculations with greater accuracy. This statement particularly applies to large time steps. Small time intervals between shape function calculations allows the IQS method to follow the transient well. For larger time steps, the effects of the more rigorous GQS treatment of a multigroup shape function will be greater felt. Hence in this chapter large time step cases are presented to test the performance of the code and very small time step cases are used to verify the temporal convergence of the method.

Comparing reactivity transients is much more difficult in the GQS method because reactivity is a matrix, instead of scalar quantity. An equivalent scalar reactivity is not easily calculated. For this reason no reactivity transient are given in this chapter.

5.1 SIMPLE Test Case

A short description is given in section 3.1 and a complete description of the SIMPLE test case is given in Appendix B.

In section 3.1, the accuracy of the quasistatic option of the code was analyzed using the total power error as a criterion. Since total reactor power is primarily a function of the thermal flux transient (this case has a small fast production cross-section) this criterion is not useful to examine fast flux changes. Hence, the error in the average fast and thermal flux will be used to gauge the performance of the different methods, with a semi-analytic code called POLES (see Appendix A) used to generate reference solutions.

Uniform step perturbations in the thermal absorption cross-section of (Σ_{a2}) of $-.2 \times 10^{-4}$ and 1.0×10^{-4} cm^{-1} corresponding to step reactivity insertions of 4.94 and -29.4 mk, respectively, are used to test the generalized point kinetics in XSTATIC. The reference average fast and thermal fluxes generated by POLES are shown in Table 5.1. These two cases were run using the IQS method, and the IQS and GQS methods with no shape calculation updates. The last two methods are equivalent to the point kinetics (PK) and generalized point kinetics (GPK). All the above XSTATIC calculations were performed with the following input data

Parameter	Default Values
mesh split	1
time split	1
ε_ρ	0.5×10^{-5}
ε_T	0.5×10^{-5}
ε_E	0.5×10^{-5}
ε_{max}	0.5×10^{-5}
ε_{shape}	0.5×10^{-5}

and the results obtained are shown in Table 5.2 and Table 5.3.

For the above uniform perturbation transients, there is no spatial distortion of the shape function but rather a movement of the fast to thermal flux ratio. Generalized point kinetics, for these cases, is an exact solution to the time dependent

Table 5.1: SIMPLE:Average Fluxes for Reference Solution

	ρ= -29.4 mk		ρ= 4.94 mk	
Time	ϕ_1	ϕ_2	ϕ_1	ϕ_2
0.	0.57236	1.0000	0.57236	1.0000
1.0	.084808	.14390	2.4576	4.3123
2.0	.072095	.12232	4.4432	7.7977
4.0	.056396	.09567	12.453	21.855
6.0	.046439	.07878	33.626	59.016
8.0	.039580	.06715	90.067	158.07

neutron diffusion equation. The errors in the GPK solution ($< 0.4\%$) are due either to errors in the computation of the point kinetics parameters or the integration of the point kinetics equations.

Using the IQS method with a time step of 1.0 second and adjoint weighting gives very good results. In fact the IQS results are slightly better than the GQS results at 8 seconds for the -29.4 mk case and this is probably due to computational error. As expected the point kinetic results are the worst of the three but still respectable due to the uniform nature of the perturbation.

As a test, the GQS method with shape function updates was simulated to ensure that the results remain unchanged and that the code performed no shape function iterations. This conclusion was verified with XSTATIC for both positive and negative reactivity insertions.

5.1.1 System Poles

This section presents a more in depth analysis of the SIMPLE test from the viewpoint of eigenvector and eigenvalues to determine the effectiveness of the GQS method for CANDU type problems.

The time-dependent neutron diffusion equation can be viewed as a set of equations whose behaviour is described by a set of eigenvalues and eigenvectors. In the SIMPLE test case (homogeneous core with uniform perturbation), 8 poles (or

Table 5.2: SIMPLE:Average Fluxes for $\rho = 4.94$ mk

	GQS		IQS		Point Kinetics	
Time	$\bar{\phi}_1$	$\bar{\phi}_2$	$\bar{\phi}_1$	$\bar{\phi}_2$	$\bar{\phi}_1$	$\bar{\phi}_2$
0.	.5724	1.000	.5724	1.000	.5724	1.000
1.0	2.462	4.320	2.461	4.319	2.472	4.318
2.0	4.451	7.811	4.455	7.819	4.473	7.816
4.0	12.46	21.87	12.53	21.96	12.56	21.95
6.0	33.61	58.98	33.87	59.44	33.98	59.37
8.0	89.90	157.8	90.97	159.5	91.19	159.3
ϵ_{MAX}	-0.18%	-0.17%	1.0%	0.9%	1.2%	0.78%

Table 5.3: SIMPLE:Average Fluxes for $\rho = -29.2$ mk

	GQS		IQS		Point Kinetics	
Time	$\bar{\phi}_1$	$\bar{\phi}_2$	$\bar{\phi}_1$	$\bar{\phi}_2$	$\bar{\phi}_1$	$\bar{\phi}_2$
0.	.57236	1.000	.57236	1.000	.57236	1.000
1.0	.08482	.1439	.08482	.1439	.08238	.1439
2.0	.07217	.1224	.07210	.1223	.07002	.1223
4.0	.05653	.09591	.0564	.09569	.05477	.09570
6.0	.04659	.07904	.04645	.0788	.0451	.0788
8.0	.03947	.06741	.03959	.06716	.03844	.06716
ϵ_{MAX}	-.27%	.39%	0.02%	.014%	-2.8%	0.01%

eigenvalues) and eigenvectors describe the behaviour of the system. In fact, an infinite set exists, but the rest are disregarded because they do not satisfy the initial conditions. The process of applying the functional $l\{\}$ to the time-dependent neutron diffusion equation can be viewed as a mapping or projection of the system poles. The point kinetics (PK) equations are obtained if $l\{\}$ is defined as in chapter 2, which sums over the energy groups. The generalized point kinetics equations are defined in chapter 4 and do not involve summing over all energy groups. Some information about the original system is lost in the PK projection. Normally this would be the case even for GPK, but the poles of the SIMPLE test case are the same as the poles of time-dependent neutron diffusion equation due to the uniform nature of the perturbation. For this type of problem the poles generated by the GPK are independent of the weight function.

Table 5.4 shows the poles, which were generated by the code POLES, of the GPK and PK systems for step reactivity insertions worth -29.4 and 4.94 mk in the thermal absorption cross-section. Adjoint weighting was used for the PK case, while no special weighting function is necessary for the GPK case. The introduction of a fast flux amplitude function into the GPK equation results in the addition of a pole (8) at -99840., making the system extremely stiff. The other poles are well predicted (error $< 0.1\%$) using PK with adjoint weighting.

Another choice for weight function in l is the flux itself. This is called Galerkin weighting. Plotting the error in the PK poles using adjoint and Galerkin weighting for various perturbations in the thermal absorption cross-section gives Figures 5.1 and 5.2, respectively. Errors of 40% in the poles predicted using Galerkin weighting lead to extremely large errors as the transient progresses. Hence using Galerkin weighting for point kinetics calculations will lead to large errors, which are attributable to a poor description of the poles of the system.

In contrast, PK using adjoint weighting very accurately predicts the poles (Table 5.4) and the flux transients (Table 5.2 and 5.3). Even though the error in the

Table 5.4: System Poles for SIMPLE Test Case

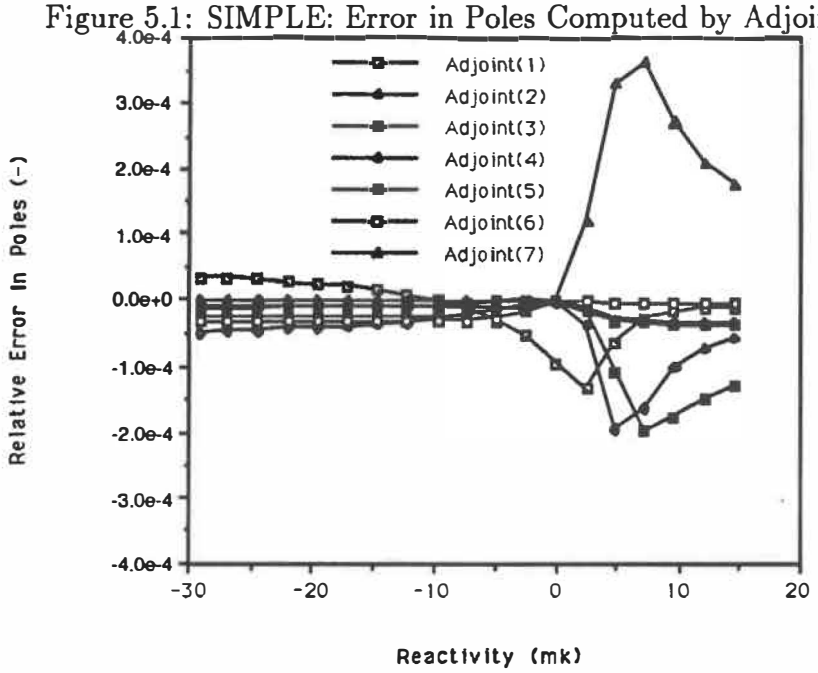
	ρ= -29. mk		ρ= 4.94 mk	
Pole	Exact (GPK)	Point Kinetics (Adj. Weighting)	Exact (GPK) (Adj. Weighting)	Point Kinetics
1	-.012	-.012	0.49	0.49
2	-.030	-.030	-0.13	-0.13
3	-0.11	-0.11	-0.14	-0.14
4	-0.28	-0.28	-0.41	-0.41
5	-1.36	-1.36	-0.89	-0.89
6	-3.85	-3.85	-2.58	-2.58
7	-36.39	-36.39	-4.22	-4.22
8	-99480	NA	-99480.	NA

poles are very small, small errors in the fast flux exist for the PK cases. These errors are due to the inability of the point kinetics to track changes in the ratio of the fast to thermal flux amplitudes. The change in this ratio can be determined by finding the dominant (ie. least negative) pole and obtaining the ratio of the fast to thermal amplitudes from the corresponding eigenvector. Figure 5.3, which was generated using this method, shows that for negative reactivity insertions, only perturbations to the thermal production (SF2) or the fast to thermal scattering cross-section (ST) change the fast to thermal ratio. However for positive reactivity insertions perturbing any property has an effect on this ratio.

Part of the ability of the GQS method to improve the computational speed and accuracy of the code lies in its ability to track this change in the fast to thermal flux ratio. For a reactivity insertion of -30.0 mk the change in the ratio is only 3%. This changes is relatively small and should not by itself make the GQS method superior to the IQS.

Figures 5.1 and 5.2 show that the point kinetics error in the amplitude is dependent on the weight function, and that the GPK equations are not as sensitive. For a reactor transient sensitive to changes in the time-dependent adjoint or for a case where the adjoint has not been computed, the GQS method should perform

Figure 5.1: SIMPLE: Error in Poles Computed by Adjoint Weighting



better than the IQS method.

Figure 5.2: SIMPLE: Error in Poles Computed by Galerkin Weighting

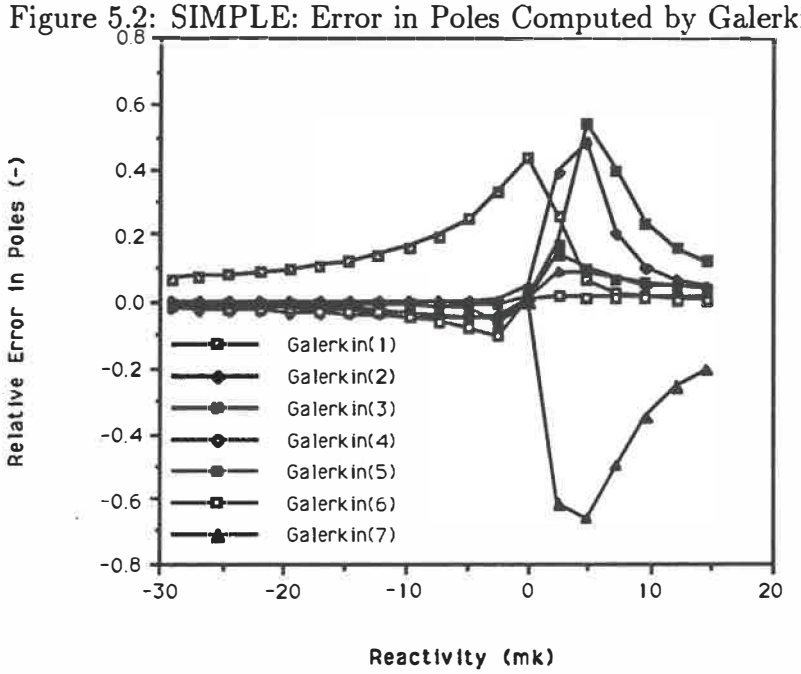
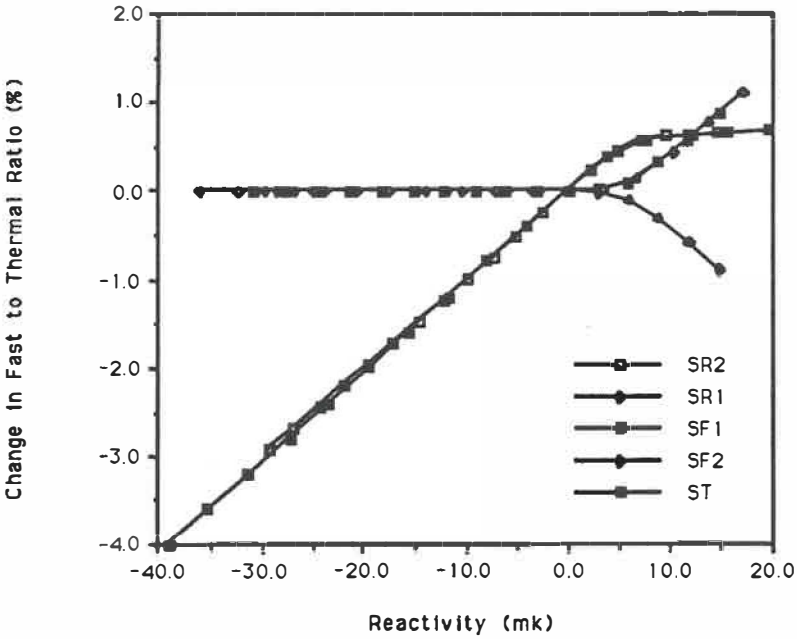


Figure 5.3: Error in Fluxes Computed by Adjoint Weighting



5.2 LMW Test Case

A short description is given in section 3.2 and a complete description of the LMW test case is given in Appendix B.2.

The results obtained using the GQS method are shown in Table 5.5. The GQS option of XSTATIC obtains better results for the power at 10. seconds than does the IQS option. For MCFD2 with a mesh split=2 and time split = 20 case, the maximum error is reduced from 2.4 to 1.5%. This decrease in error is typical of the other cases. At 25. seconds, the two methods give roughly the same results. This recovery of the error is probably due to the strong dependence on the precursors when the rods are fully inserted.

Table 5.5: LMW: GQS XSTATIC Results for LAGR1

Type	LAGR1	LAGR1	MCFD2	MCFD2
Split	1	1	2	2
Step	1	20	1	20
P(5 sec)	94756	94705	92627	92630
P(10 sec)	287733	286798	279552	279382
P(15 sec)	159979	159672	159542	159905
P(20 sec)	74370	74374	78089	78426
P(25 sec)	53688	53726	57490	57770
$\epsilon_{AV}(0)$	2.9%	2.9%	0.9%	0.9%
$\epsilon_{MAX}(0)$	8.2%	8.2%	2.2%	2.2%
$\epsilon_{TP}(10)$	4.4%	3.4%	0.7%	0.7%
$\epsilon_{AV}(10)$	5.3%	4.5%	1.1%	1.1%
$\epsilon_{MAX}(10)$	13.2%	12.3%	1.6%	1.5%
$\epsilon_{TP}(25)$	—%	-7.7%	-1.3%	-0.7%
$\epsilon_{AV}(25)$	3.0%	6.8%	1.7%	1.2%
$\epsilon_{MAX}(25)$	8.7%	12.8%	3.5%	3.1%

Comparing Table 5.5 with Tables 3.4 and 3.5 show the accuracy of the two methods. In most cases the GQS method is only slightly more accurate. The largest improvement in accuracy for the GQS method occurs for the LAGR1 case with a time and mesh split of 1, where the maximum error is 8.7% compared to

12.7% for the IQS method at 25 s. This increase in accuracy is probably due to the large change in the fast and thermal amplitudes caused by insertion of control rods. The improvements also come in a portion of the transient which is of little interest. It is hard to justify the extra complications of the GQS for such a small improvement in the accuracy of the results.

5.3 AECL Test Case

The description of the AECL test case is given in Appendix B.4 and a short description is given in section 3.4.

To test the temporal convergence of the GQS method in XSTATIC, this case was run with a time split of 8 and no mesh splitting. The results are shown in Table 5.7. The magnitude of the error shows that it is very close to the reference solution. Decreasing the time split to 1, gives the results shown in Table 5.6. Comparing this result with equivalent IQS results (see Table 3.15) shows that the result is quite comparable and little was gained in terms of accuracy using the GQS method even for large time steps. Unfortunately, this test case shows that little improvement in accuracy is gained using the GQS method for a real CANDU test case.

Table 5.6: AECL: Constrained GQS Results (tsplit=1,msplit=1)

Time	ε_{TP} %	ε_{AV} %	ε_{MAX} %
0.00	0.00	0.00	0.00
0.20	0.00	0.12	0.38
0.60	-0.25	0.16	0.54
0.89	-0.79	0.44	1.02
1.35	-6.86	3.74	7.72
1.47	-7.14	3.83	8.20
2.10	-2.13	1.04	2.57
2.50	-1.83	0.89	2.26

Table 5.7: AECL: Constrained GQS Results (tsplit=8,msplit=1)

Time	ε_{TP} %	ε_{AV} %	ε_{MAX} %
0.00	0.00	0.00	0.00
0.20	0.00	0.00	0.03
0.60	-0.03	0.02	0.04
0.89	-0.11	0.07	0.17
1.35	-0.12	0.08	0.22
1.47	-0.16	0.10	0.24
2.10	-0.28	0.15	0.33
2.50	-0.22	0.12	0.27

Table 5.8: AECL: GQS Results Using Galerkin Weighting (tsplit=1,msplit=1)

Time	ε_{TP} %	ε_{AV} %	ε_{MAX} %
0.00	0.00	0.01	0.03
0.20	-0.05	0.13	0.38
0.60	-0.24	0.15	0.50
0.89	-0.93	0.52	1.15
1.35	-6.55	3.56	7.43
1.47	-6.78	3.63	7.82
2.10	-2.20	1.07	2.66
2.50	-1.89	0.92	2.36

5.3.1 Galerkin Weighting

For a time split of 8, the Galerkin weighting results are within 0.5% of the reference results, indicating that for small time steps Galerkin weighting gives the same results as adjoint weighting. If the time split is decreased to 1, the results shown in Table 5.8 are obtained. Comparing these results to the equivalent IQS results given in Table 3.17 shows that the GQS and IQS method give roughly the same results for this case also. This test case shows that the accuracy of neither the IQS northe GQS method is affected by the weight function for a typical CANDU problem. A larger difference would probably be evident for larger time steps.

5.3.2 Computational Time

The time spent doing different calculations is shown in Table 5.9. Solution of the stiff generalized point kinetics required almost 70% of the CPU time. This is unrealistically high and can probably be substantially reduced by changing the solution algorithm or reducing the convergence criteria. Consequently, the time spent solving the shape function would become more important.

Comparing the computational time needed by the IQS method to solve the shape function equation, given in Table 3.19, show a 10% reduction for the GQS method for small time steps (tsplit=1). Note also that the CPU time for the Galerkin and adjoint cases are very similar for the GQS method while for the IQS method a 10% increase is seen.

Table 5.9: AECL: Computational Time for the GQS Method

Case Description			
Time Split	1	8	1
Mesh Split	1	1	1
Method	GQS	GQS	GQS
Constrained	yes	yes	yes
Weight Function	Adjoint	Adjoint	Galerkin
Maximum Error	8.2%	<0.5%	7.8%
CPU Time			
Area	Time	Time	Time
Shape Function	25%	17%	25%
CPU sec	888	5400	890
Point Kinetics	70%	74%	71%
Assemble Matrices	1%	1%	1%
Other	4%	8%	3%
Total	3622	30299	3607

CHAPTER 6

Conclusions

The objective of this thesis was the development of the Improved Quasistatic (IQS) method and the Generalized Quasistatic (GQS) method and their implementation in TRIVAC as a module called XSTATIC. The development of the IQS and GQS methods was outlined in chapters 2 and 4, respectively. The solution of several benchmark kinetics problems by both methods is presented in chapter 3 and 5.

The test results show that either method can be used to generate results very close to the benchmark solutions. However, at present, the more traditional and less rigorous IQS method is preferred for solving CANDU-type kinetics problems, due to faster execution times and comparable accuracy.

6.1 IQS Method Conclusions

The following conclusions summarize chapter 3:

1. The SIMPLE test case verifies the point kinetics routines in XSTATIC. A total power error of at least 0.2% for a 10 second transient is to be expected.

2. The LMW test case shows that the code is well-behaved for all the spatial discretization option (MCFD1, MCFD2, LAGR1, LAGR2). Well-behaved means that the error decreases as Δx and Δt decrease.

3. The FASTQ solution of the point kinetics equations is much faster and more accurate than the STIFFZ version.

4. Galerkin weighting leads to large errors when used in the point kinetics equations, but produces respectable results when used in the IQS method.

5. The spectral radius of the residual matrix ($\mathbf{R}$) of the iterative system was computed and shown to be very close to 1 for large Δt (>.1 s).

6. The unconstrained version of XSTATIC gives the most accurate results over the full range of time step sizes (about 1/2 the error). The constrained version is the next best and the linear reactivity version (CERBERUS like) is the least accurate.

7. Changing the weight function can significantly affect the reactivity, but has little effect on the power transient for small Δt

8. The effect of rod cusping on the power and reactivity transient is substantial. The phenomenon challenges the fundamental correctness of the solution obtained.

9. Using Galerkin weighting increases the time spent solving the shape function equation by about 10%.

10. The unconstrained version of XSTATIC has the fastest computational time of any of the methods.

6.2 GQS Method Conclusions

The following conclusions summarize chapter 5:

1. The SIMPLE test case verifies the generalized point kinetics routines in XSTATIC.

2. The LMW test case shows that the code is also well behaved (the error decreases as Δx and Δt decrease) for all spatial discretization option (MCFD1,MCFD2,LAGR1,LAGR2)

3. The use of the GQS method speeds up the shape function calculation by about 10%. However, the solution of the generalized point kinetics equations requires 70% of the CPU time, because they are not optimized.

4. Using Galerkin instead of adjoint weighting does not seem to affect the accuracy or speed of the GQS method.

6.3 Recommendations for Further Research

Further research should be done in the following areas:

1. The GQS method is mathematically preferable to the IQS method, however the substantial increase in computational time spent solving the point kinetics equations makes this option unsuitable for reactor calculations. Applying the techniques used to integrate the IQS method (FASTQ in chapter 2) would substantially reduce the computational time. The execution times required by the codes would then be very comparable, with the GQS method slightly faster.

2. The use of a parabolic variation of the reactivity during a macro time step, which leads to rod cusping, may not be more accurate than the linear variation. Some research into whether linear or parabolic variation of reactivity better models the movement of a shut off rod in the reactor is needed. Also the typical modelling of shut off rod movements does not take into account the perturbation felt by the next mesh cell (or next channel), because the rod is modelled as only changing the mesh cell which contains the tip of the rod.

BIBLIOGRAPHY

[1] STACEY Jr. W.M., "Space-Time Reactor Kinetics", Academic Press, New York, 1969.

[2] BELL G.I. and GLASSTONE S., "Nuclear Reactor Theory",Von Nostrand, Princeton, N.J. (1970), pp. 532-555.

[3] OTT K.O. and MENELEY D.A., Nucl. Sci. Eng. 36, 402 (1969).

[4] GOLD M. and LUXAT J.C., "SMOKIN - A Code for Time-Dependent Three-Dimensional Neutronics Calculations in CANDU-PHW Reactors Based on Modal Kinetics Theory. Application to analysis of Loss of Coolant Accidents", presented at the International Conference on the Physics of Reactors: Operations, Design and Computation, Marseille, France, April 23-27, 1990.

[5] MENELEY D.A., OTT K.O. and WIENER E.S., "Fast Reactor Kinetics - The QX-1 code", ANL 7769, Argonne National Laboratory, 1971.

[6] DODDS H.L., Nucl. Sci. Eng. 59, 271 (1976).

[7] DASTUR A.R., ROUBEN B., BUSS D.B., "CERBERUS - A Computer Program for Solving the Space-Dependent Neutron Kinetics Equations in Three Dimension", AECL Report TDAI-176, December 1979.

[8] DEVOOGHT J., "Generalized Quasistatic Method for Nuclear Reactor Space Time Kinetics", Nuclear Science and Engineering, June 1980.

[9] HÉBERT A., "Application of the Hermite Method for Finite Element Reactor Calculations", Nucl. Sci. Eng. 91, 34 (1985).

[10] HÉBERT A., "Variational Principles and Convergence Acceleration Strategies for the Neutron Diffusion Equations", Nucl. Sci. Eng. 91, 414 (1985).

[11] HÉBERT A., "TRIVAC, A Modular Diffusion Code for Fuel Management and Design Applications", Nucl. J. of Canada, Vol. 1, No. 4, 325 (1987).

[12] HENRY A.F., "Nuclear Reactor Analysis", M.I.T. Press, Cambridge, Mass. (1975), pp. 347-351.

[13] HÉBERT A., "Development of the Nodal Collocation Method for Solving the Neutron Diffusion Equation", Ann. Nucl. Energy, Vol 14, No. 10, 527(1987).

[14] LAMBERT J.D., "Computational Methods in Ordinary Differential Equations", Wiley, 1973.

[15] GREENMAN G., "A Quasistatic Flux Synthesis Temporal Integration Scheme for an Analytic Nodal Method", Massachusetts Institute of Technology PHD Thesis, 1980.

[16] DENNING R.S.,"ADEP: One- and Two-Dimensional Few-Group Kinetics Code", 3MI-1911, Battelle Memorial Institute, Columbus (1971).

[17] ROUBEN B., AUGUSTINE M., WALK K., JUDD R.A., "CERBERUS Solution of a 3-D Kinetics Benchmark Problem in a Heavy Water Reactor", TDAI-260, September 1981.

[18] LANGENBUCH S., MAUER W., WERNER W., "Coarse-Mesh Nodal Diffusion Method for the Analysis of Space-Time Effects in Large Light Water Reactor", Nucl. Sci. Eng. 63, 437-456.

[19] McDONNELL F.N., BAUDOIN A.P., GARVEY P.M. and LUXAT J.C.," CANDU Reactor Kinetics Benchmark Activity",Nucl. Sci. Eng. 64, 95 (1977).

[20] HÉBERT A., "Guide de lusager du code TRIVAC-2 Version 1.3", Rapport GAN-145 R2, Ecole Polytechnique de Montréal, Mai 1988. No. 10, 527(1987).

Appendix A

Programmer Guide to the Code POLES

The code POLES solves semi-analytically the time-dependent neutron diffusion equation for a 2-energy-group homogeneous 1-dimensional reactor (shown in Table A.1) with uniform step perturbation of any properties. The codes permits up to 10 precursor groups. As output the code provides:

1. the analytic solution of the transient
2. the point kinetics solution of the transient
3. the eigenvalue and eigenvectors of the system
4. the spectral radius of the iterative system

The purpose of this appendix is to derive the equations and show the solution technique that POLES uses to solve the time dependent neutron diffusion equation.

Table A.1: SIMPLE TEST CASE DESCRIPTION

Boundary Conditions
$\phi(0.) = 0 \qquad \phi(L) = 0$

A.1 Static Equation

Before solving the time-dependent problem, a solution to the static problem must first be determined. The two group diffusion equation to this problem are:

$$(D_1\nabla^2 - \Sigma_{a1})\phi_1(x) \quad + \quad \frac{1}{k_{eff}}(\nu\Sigma_{f1}\phi_1(x) + \nu\Sigma_{f2}\phi_2(x)) = 0 \tag{A.1}$$

$$\Sigma_{21}\phi_1(x) \quad + \quad (D_2\nabla^2 - \Sigma_{a2})\phi_2(x) = 0 \tag{A.2}$$

A zero flux boundary conditions at x = 0 and at x = L imply a solution of the form:

$$\vec{\phi} = \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = \begin{pmatrix} Asin(Bx) \\ sin(Bx) \end{pmatrix} \tag{A.3}$$

where B= $\frac{\pi}{L}$

Substituting equation A.3 into equations A.1 and A.2 gives the following equations for A and k_{eff}

$$(-D_1B^2 - \Sigma_{a1})A \quad + \quad \frac{1}{k_{eff}}(\nu\Sigma_{f1}A + \nu\Sigma_{f2}) = 0 \tag{A.4}$$

$$\Sigma_{21}A \quad + \quad (-D_2B^2 - \Sigma_{a2}) = 0 \tag{A.5}$$

Solving equation A.5 for A and k_{eff} gives the following equations

$$A \quad = \quad \frac{D_2B^2 + \Sigma_{a2}}{\Sigma_{21}B^2} \tag{A.6}$$

$$k_{eff} \quad = \quad \frac{(\nu\Sigma_{f1}A + \nu\Sigma_{f2})}{(D_1B^2 + \Sigma_{a1})A} \tag{A.7}$$

A.2 Adjoint Problem

The equations for the adjoint problem are:

$$(D_1\nabla^2 - \Sigma_{a1})\phi_1(x) \quad + \quad \Sigma_{21}\phi_1(x) = 0 \tag{A.8}$$

$$\frac{1}{k_{eff}}(\nu\Sigma_{f1}\phi_1(x) + \nu\Sigma_{f2}\phi_2(x)) \quad + \quad (D_2\nabla^2 - \Sigma_{a2})\phi_2(x) = 0 \tag{A.9}$$

Solution of the steady state adjoint problem is quite similar to the direct problem. Assuming a solution of the form

$$\vec{\phi} = \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = \begin{pmatrix} A^* sin(Bx) \\ sin(Bx) \end{pmatrix} \tag{A.10}$$

where B= $\frac{\pi}{L}$ and substituting into equation A.8 and A.9 gives:

$$(-D_1B^2 - \Sigma_{a1})A^* \quad + \quad \Sigma_{21} = 0 \tag{A.11}$$

$$\frac{1}{k_{eff}}(\nu\Sigma_{f1}A^* + \nu\Sigma_{f2}) \quad + \quad (-D_2B^2 - \Sigma_{a2}) = 0 \tag{A.12}$$

The k_{eff} is the same as for the direct flux equation, which can easily be verified. Solving for A^* gives the following equation.

$$A^* = \frac{\Sigma_{21}}{D_1B^2 + \Sigma_{a1}} \tag{A.13}$$

$$k_{eff} = \frac{(\nu\Sigma_{f1}A^* + \nu\Sigma_{f2})}{D_2B^2 + \Sigma_{a2}} \tag{A.14}$$

A.3 Transient Solution

The solution of the time-dependent neutron diffusion equation is much more complex. Although it can be completely solved by analytic techniques for the SIMPLE test case, a limited part is solved numerically to avoid round-off error. This numerical round-off is the result of evaluating an ill-conditioned equation that is obtained from an analytic solution. The two-energy-group time-dependent neutron diffusion equation can be expressed as:

$$\frac{1}{v_1}\frac{\partial}{\partial t}\phi_1(x,t) = (D_1\nabla^2 - \Sigma_{a1} + \nu\Sigma_{f1})\phi_1(x,t) + \nu\Sigma_{f2}\phi_2(x,t) + \sum_i \lambda_i C_i(x,t) \tag{A.15}$$

$$\frac{1}{v_2}\frac{\partial}{\partial t}\phi_2(x,t) = \Sigma_{21}\phi_1(x,t) + (D_2\nabla^2 - \Sigma_{a2})\phi_2(x,t) \tag{A.16}$$

$$\frac{\partial}{\partial t}C_i(x,t) = \beta_i\nu\Sigma_{f1}\phi_1(x) + \nu\Sigma_{f2}\phi_2(x,t) - \lambda_i C_i(x,t) \tag{A.17}$$

Assuming a separation of the temporal and spatial variables on equation A.3, the flux can be written as:

$$\vec{\phi} = \begin{pmatrix} \phi_1(x,t) \\ \phi_2(x,t) \end{pmatrix} = \begin{pmatrix} T_1(t)sin(Bx) \\ T_2(t)sin(Bx) \end{pmatrix} \tag{A.18}$$

Substituting the above factorized flux into equations A.13 to A.15 the equations can be expressed in matrix form:

$$\frac{d}{dt}\mathbf{V^{-1}}\vec{Z} = \mathbf{Q}`(t)\vec{Z} \tag{A.19}$$

$$\vec{Z} = \begin{pmatrix} T_1 \\ T_2 \\ C_1 \\ \vdots \\ C_N \end{pmatrix} \tag{A.20}$$

$$\mathbf{Q}`(t) = \begin{pmatrix} -D_1B^2 - \Sigma_{a1} + \nu\Sigma_{f1} & \nu\Sigma_{f2} & \lambda_1 & \dots & \lambda_N \\ \Sigma_{21} & -D_2B^2 - \Sigma_{a2} & & & \\ \beta_1\nu\Sigma_{f1} & \beta_1\nu\Sigma_{f1} & -\lambda_1 & \dots & \\ \vdots & \vdots & & \ddots & \vdots \\ \beta_N\nu\Sigma_{f1} & \beta_N\nu\Sigma_{f2} & & \dots & -\lambda_N \end{pmatrix} \tag{A.21}$$

$$\mathbf{V^{-1}} = \begin{pmatrix} \frac{1}{v_1} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{v_2} & & & \\ 0 & 0 & 1 & \dots & \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & & \dots & 1 \end{pmatrix} \tag{A.22}$$

Multiplying equation A.21 by $\mathbf{V}$ gives:

$$\frac{d}{dt}\vec{Z} = \mathbf{V}\mathbf{Q}`(t)\vec{Z} = \mathbf{Q}(t)\vec{Z} \tag{A.23}$$

The solution for this system of equations is simply:

$$\frac{d}{dt}\vec{Z}(t) \;=\; e^{\mathbf{Q(t)}t}\vec{Z}(t=0) \tag{A.24}$$

This code is limited to solving for step variation in nuclear properties at t=0, and hence $\mathbf{Q}(t) = \mathbf{Q}$. Evaluation of the transition matrix $e^{\mathbf{Q}t}$ for this linear system is done via the straightforward technique of diagonalization. That is to say, there exists a matrix $\mathbf{S}$ such that $\mathbf{\Lambda} = \mathbf{S^{-1}QS}$ is diagonal matrix and hence

$$\frac{d}{dt}\vec{Z}(t) = \mathbf{S^{-1}}e^{\mathbf{SQS^{-1}}t}\mathbf{S}\vec{Z}(t=0) \tag{A.25}$$

Construction of the matrix $\mathbf{S}$ requires the knowledge of the eigenvalues and eigenvectors of the system. For the 1-precursor group case the eigenvalues are the roots of a 3rd order characteristic polynomial. However the round-off error introduced by direct evaluation of the analytic roots requires quadruple precision (128 byte) mathematics for an accurate answer. As an alternative the ESSL eigenvalue routine DGEEV was used. This routine computes the eigenvalues by balancing, then transforms the system to Hessenberg form and uses the QR algorithm with an implicit shift to solve for the eigenvalues. The results obtained are extremely accurate even for a system as stiff as this one. The ESSL results were of comparable accuracy with those obtained from the quadruple precision analytic case.

The above analytic solutions can be used to compute the spectral radius of the residual matrix $\mathbf{R}$ in equation 3.9. A given time step Δt is assumed and analytic solutions are used to compute γ and $\frac{1}{T}\frac{dT}{dt}$. The spectral radius of $\mathbf{R}$ is determined from a simple 2×2 eigenvalue calculation.

Appendix B

Test Case Descriptions

B.1 SIMPLE Test Case Description

This simple test case was developed to qualify XSTATIC against analytic solutions. No other sources of analytic solutions to this problem exist except our own.

B.1.1 Sample Input

```
' VERIFICATION TEST FOR POINT KINETICS OF XSTATIC '
MAXIMA X1D LAGR 1 40 FIND
GEOM   IMPR 1 PRED 10
       ZERO XSUP ZERO XINF
            1   1   1    1    1    1    1    1    1    1
       0.0  39. 78. 117. 156. 195. 234. 273. 312. 351. 390.
      SPLIT 4   4   4    4    4    4    4    4    4    4
       FIND
FLUX  PREC .5E-5 ADJ IMPR 3 SGD 2
       (  1)
   1.2000E+00 8.00000E-03 2.00000E-04 2.00000E-04 *
   1.0000E-01 4.00000E-03 4.50000E-03 4.50000E-03 T 7.00000E-03 *
   * FIND
   SORTIE POWR 0.46147
   INTG # POWD IMPR FIND
TRAN 6
       0.000247   0.0127
       0.0013845  0.0317
       0.001222   0.115
       0.0026455  0.311
       0.000832   1.40
       0.000169   3.87
       1.25E7     2.5E5
       CONV 1.0E-7  0.1E-1  0.5E-5  10 MAX 40
       CON2 .5E-4 5.E-6 1.
       OPTION  RENORM      *
       PKIN    QSM       POWER 0.46147
       *
(T=1)  PERT 1 STEP 1.00 SHOW REACT 5 *
       DSGD 2 1
       0.0E+00 0.0E-03 0.0E-04 0.0E-04 *
```

```
        0.0E-01 -.2E-04 0.0E-03 0.0E-03 T 0.0E-03 *
        *
        END
        OUTPUT INTG # POWD IMPR FIND

(T=2)   PERT 1 STEP 1.E00 SHOW REACT 5 *
        DSGD 2 1   0. 0. 0. 0. * 0. 0. 0. 0. T 0. *  * END
        OUTPUT INTG # POWD IMPR FIND
(T=3)   PERT 1 STEP 1.E00 SHOW REACT 5 *
        DSGD 2 1   0. 0. 0. 0. * 0. 0. 0. 0. T 0. *  * END
(T=4)   PERT 1 STEP 1.E00 SHOW REACT 5 *
        DSGD 2 1   0. 0. 0. 0. * 0. 0. 0. 0. T 0. *  * END
        OUTPUT INTG # POWD IMPR FIND
(T=5)   PERT 1 STEP 1.E00 SHOW REACT 5 *
        DSGD 2 1   0. 0. 0. 0. * 0. 0. 0. 0. T 0. *  * END
(T=6)   PERT 1 STEP 1.E00 SHOW REACT 5 *
        DSGD 2 1   0. 0. 0. 0. * 0. 0. 0. 0. T 0. *  * END
        OUTPUT INTG # POWD IMPR FIND
(T=7)   PERT 1 STEP 1.E00 SHOW REACT 5 *
        DSGD 2 1   0. 0. 0. 0. * 0. 0. 0. 0. T 0. *  * END
(T=8)   PERT 1 STEP 1.E00 SHOW REACT 5 *
        DSGD 2 1   0. 0. 0. 0. * 0. 0. 0. 0. T 0. *  * END
        OUTPUT INTG # POWD IMPR FIND
(T=9)   PERT 1 STEP 1.E00 SHOW REACT 5 *
        DSGD 2 1   0. 0. 0. 0. * 0. 0. 0. 0. T 0. *  * END
(T=10)  PERT 1 STEP 1.E00 SHOW REACT 5 *
        DSGD 2 1   0. 0. 0. 0. * 0. 0. 0. 0. T 0. *  * END
        OUTPUT INTG # POWD IMPR FIND
FIND
FIN
```

Table B.1: SIMPLE TEST CASE DESCRIPTION

Boundary Conditions
$\phi(0.) = 0 \qquad \phi(390.) = 0$

Region	Group_g	D_g (cm)	Σ_{r_g} (cm^{-1})	$\nu\Sigma_{f_g}$ (cm^{-1})	Σ_{21} (cm^{-1})
1	1	1.2	0.008	0.0002	0.007
	2	0.1	0.004	0.0045	

$\chi_1 = 1.0 \quad \chi_2 = 0.0$
$\nu = 2.5$
$v_1 = 1.25 \times 10^7 \frac{cm}{s} \quad v_2 = 5 \times 10^5 \frac{cm}{s}$

Family_d	β_d	$\lambda_d(s^{-1})$
1	0.000247	0.0127
2	0.0013845	0.0317
3	0.001222	0.115
4	0.0026455	0.311
5	0.000832	1.40
6	0.000169	3.87

B.2 LMW Test Case Description

This is a modified version the Lagenbuch, Mauer, and Werner[18] test case, which is a three dimensional 1/8 core benchmark. Greenman [15] transformed it to a two dimensional 1/4 core benchmark. The strength of the control rods were doubled to increase the spatial distortion. Figure B.1 and Table B.2 give a full description of the transient.

B.2.1 Sample Input

```
  'LMW 2-D NON SYMETRIC'
  MAXIMA  XY2D MCFD 2 12 12 FIND
  GEOM PRED 6 6 REFL XINF YINF ZERO XSUP YSUP
       1   1   1   2   3   4
       1   1   1   1   3   4
       1   1   5   1   3   4
       6   1   1   3   3   4
       3   3   3   3   4   4
       4   4   4   4   4   4
       0.0 10.  30.  50.  70.  90.  110. BIS
 SPLIT 2   2   2   2   2   2
       2   2   2   2   2   2
  FIND
  FLUX PREC .5E-5 ADJ SGD 2
         (  1)
.142391E+1 .2795756E-1 .6477691E-2 .259107E-2 *
.356306E+0 .8766217E-1 .1127328E+0 .450931E-1 T .175555E-1 *
         (  2)
.142391E+1 .2850756E-1 .6477691E-2 .259107E-2 *
.356306E+0 .9146217E-1 .1127328E+0 .450931E-1 T .175555E-1 *
         (  3)
.142561E+1 .2817031E-1 .7503284E-2 .3001310E-2 *
.350574E+0 .9925634E-1 .1378004E+0 .5512106E-1 T .1717768E-1 *
         (  4)
.163422E+1 .3025750E-1 .00000E+0  .00000E+0 *
.264002E+0 .4936351E-1 .00000E+0  .00000E+0 T  .2759693E-1 *
         (  5)
.142391E+1 .2795756E-1 .6477691E-2 .259107E-2 *
.356306E+0 .8766217E-1 .1127328E+0 .450931E-1 T .175555E-1 *
         (  6)
.142391E+1 .2850756E-1 .6477691E-2 .259107E-2 *
.356306E+0 .9146217E-1 .1127328E+0 .450931E-1 T .175555E-1 *
  * FIND

  SORTIE INTG # POWD  FIND

  TRANSIENT 6
      0.000247   0.0127
```

```
      0.0013845  0.0317
      0.001222   0.115
      0.0026455  0.311
      0.000832   1.40
      0.000169   3.87
      1.25E7   2.5E5
      CONV 1.0E-7  0.1E-1  0.5E-5  10 MAX 40
      CON2 .5E-4 5.E-6 1.
      QSM   OPTION RENORM *
      POWE  61600. IMPR 1
      *

  PERT  1  RAMP   5.0
        DSGD 2 6
    .00000E+0  -.05500E-2  .00000E+0  .00000E+0 *
    .00000E+0  -.38000E-2  .00000E+0  .00000E+0 *
     *
  END
  OUTPUT INTG # POWD  FIND
  PERT   1 RAMP   5.0
        DSGD 2 6
    .00000E+0  -.05500E-2  .00000E+0  .00000E+0 *
    .00000E+0  -.38000E-2  .00000E+0  .00000E+0 *
     *
  END
  OUTPUT INTG # POWD  FIND

  PERT  1  RAMP   5.0
        DSGD 2 5
    .00000E+0   .05500E-2  .00000E+0  .00000E+0 *
    .00000E+0   .38000E-2  .00000E+0  .00000E+0 *
     *
  END
  OUTPUT INTG # POWD  FIND
  PERT  1  RAMP   5.0
        DSGD 2 5
    .00000E+0   .05500E-2  .00000E+0  .00000E+0 *
    .00000E+0   .38000E-2  .00000E+0  .00000E+0 *
     *
  END
  OUTPUT INTG # POWD  FIND
  PERT  1  RAMP    5.0
        DSGD 2 5
    .00000E+0   .00000E-2  .00000E+0  .00000E+0 *
    .00000E+0   .00000E-2  .00000E+0  .00000E+0 *
     *
  END
  OUTPUT INTG # POWD  FIND 1
FIN
```

Figure B.1: LMW GEOMETRY

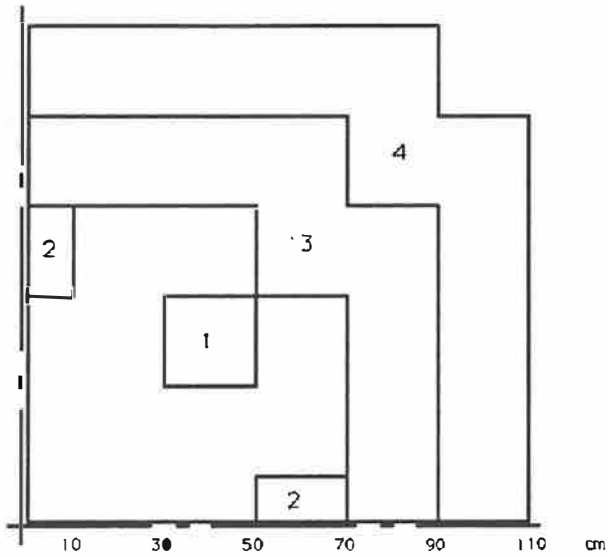


Table B.2: LMW TEST CASE DESCRIPTION

Region(Group)	D_g (cm)	Σ_{r_g} (cm^{-1})	$\nu\Sigma_{r_g}$ (cm^{-1})	Σ_{21} (cm^{-1})
1 (1)	1.423913	0.02795756	0.006477691	0.0175555
(2)	0.356306	0.08766217	0.1127328	
2 (1)	1.423913	0.02850756	0.006477691	0.0175555
(2)	0.356306	0.09146217	0.1127328	
3 (1)	1.425611	0.02817631	0.007503284	0.01717768
(2)	0.350574	0.099925634	0.13708004	
4 (1)	1.634227	0.03025703	0.0	0.02759693
(2)	0.264002	0.04936351	0.0	

$\chi_1 = 1.0 \quad \chi_2 = 0.0$

$\nu = 2.5$

$v_1 = 1.25 \times 10^7 \frac{cm}{s} \quad v_2 = 5 \times 10^5 \frac{cm}{s}$

Family_d	β_d	$\lambda_d(s^{-1})$
1	0.000247	0.0127
2	0.0013845	0.0317
3	0.001222	0.115
4	0.0026455	0.311
5	0.000832	1.40
6	0.000169	3.87

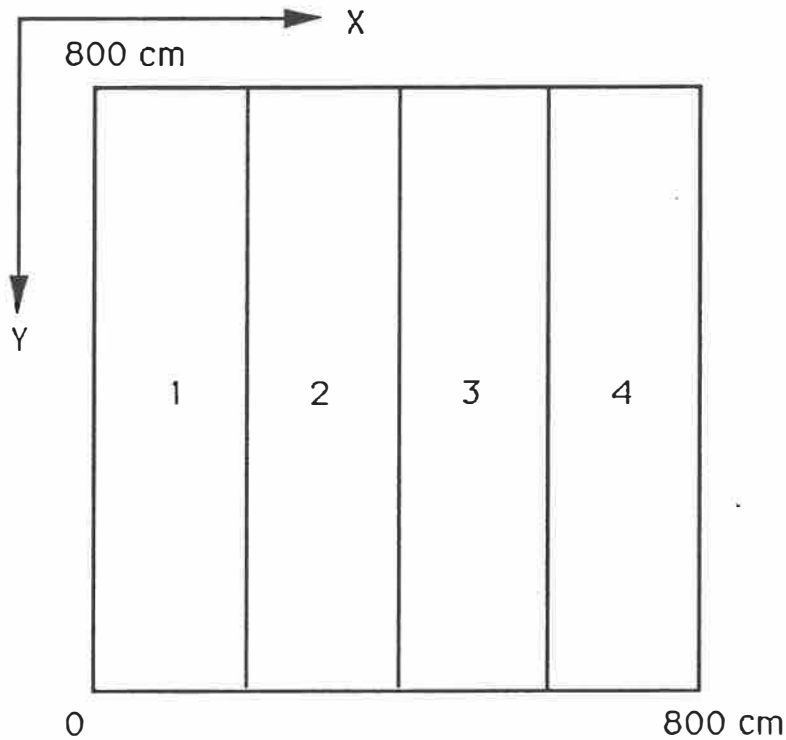


Figure B.2: CKRB-2 GEOMETRY

B.3 CRKB-2 Test Case Description

B.3.1 Sample Input

```
 'CRKB 2-D SIMPLIFIED LOCA TRANSIENT'
 MAXIMA  XY2D  MCFD 1 128 8 FIND
 GEOM PRED 10 1 ZERO XINF YINF XSUP YSUP
      1   2   2   2   2   3   3   3   3   4
      0.  200.  212.5  225. 237.5  400.  562.5
          575.  587.5  600.  800.
      0.  800.
(* STRANGE SPLITTING NEEDED FOR COMPARISON WITH ADEP
SPLIT 16   1   1   1  13  13   1   1   1  16
      8 FIND
 FLUX PREC .5E-5 ADJ IMPR 1 SGD 2
        (  1)
 1.2   .01     0.0          0.0        *
```

```
0.9    .004     0.00459653    0.00459653 T  0.009      *
        (  2)
1.2    .01      0.0           0.0        *
0.9    .004     0.00450156    0.00450156 T  0.009      *
        (  3)
1.2    .01      0.0           0.0        *
0.9    .004     0.00450156    0.00450156 T  0.009      *
        (  4)
1.2    .01      0.0           0.0        *
0.9    .004     0.00459653    0.00459653 T  0.009      *
* FIND

SORTIE  INTG # IMPR
FIND
TRANSIENT 1
    0.00572    0.08
    7.16+6   2.86+5
    CONV .50e-6  0.1-2  0.5E-5  20 MAX  40
    CON2 .50e-5 1.E-5 1.            IMPR 2  (ADI 3)
    QSM  OPTION   FASTQ RENORM   *
*

PERT  1       RAMP      1.0000
  DSGD 2 1
  .00000E+0  -.0000E-2   .00000E-3 .00000E-3 *
(*  .00000E+0  -.0000E-2   .02298E-3 .02298E-3 *
  .00000E+0  -.0000E-2   .022809E-3 .02298E-3 *
         2
  .00000E+0  -.0000E-2   .00000E-3 .00000E-3 *
(*  .00000E+0  -.0000E-2   .02250E-3 .02250E-3 *
  .00000E+0  -.0000E-2   .022333E-3 .02250E-3 *
         3
  .00000E+0  -.0000E-2   .00000E-3 .00000E-3 *
(*  .00000E+0  -.0000E-2   .02250E-3 .02250E-3 *
  .00000E+0  -.0000E-2   .022333E-3 .02250E-3 *
         4
  .00000E+0  -.0000E-2   .00000E-3 .00000E-3 *
(*  .00000E+0  -.0000E-2   .02298E-3 .02298E-3 *
  .00000E+0  -.0000E-2   .022809E-3 .02298E-3 *
  *
  END
OUTPUT INTG # IMPR FIND
        ( TIME = 1.50 )
PERT  1     RAMP      0.5000
  DSGD 2 1
  .00000E+0   .0000E-2   .000000E-3 .000000E-3 *
  .00000E+0   .1600E-3   .000000E-3 .000000E-3 *
         2
  .00000E+0   .0000E-2   .000000E-3 .000000E-3 *
  .00000E+0   .1600E-3   .000000E-3 .000000E-3 *
  *
  END
OUTPUT INTG # IMPR FIND
        ( TIME = 2.90 )
PERT   1      RAMP      1.4000
```

```
  DSGD 2 1
  .00000E+0   .0000E-2   .000000E-3 .000000E-3 *
  .00000E+0   .3400E-3   .000000E-3 .000000E-3 *
        2
  .00000E+0   .0000E-2   .000000E-3 .000000E-3 *
  .00000E+0   .3400E-3   .000000E-3 .000000E-3 *
  *
  END
OUTPUT INTG # IMPR FIND
```

Table B.3: CKRB-1 TEST CASE DESCRIPTION

Region(Group)	D_g (cm)	Σ_{r_g} (cm^{-1})	$\nu\Sigma_{r_g}$ (cm^{-1})	Σ_{21} (cm^{-1})
1 (1)	1.2	0.01	0.00	0.009
(2)	0.9	0.004	0.00459653	
2 (1)	1.2	0.01	0.00	0.009
(2)	0.9	0.004	0.00450156	
3 (1)	1.2	0.01	0.00	0.009
(2)	0.9	0.004	0.00450156	
4 (1)	1.2	0.01	0.00	0.009
(2)	0.9	0.004	0.00459653	

$\chi_1 = 1.0 \quad \chi_2 = 0.0$
$\nu = 2.5$
$v_1 = 7.16 \times 10^6 cm/s \quad v_2 = 2.86 \times 10^5 cm/s$

Family_d	β_d	$\lambda_d(s^{-1})$
1	0.00572	0.08

Transient Information

Time	Perturbation		
$0 < t < 1.0$	$\Delta\nu\Sigma_{f2} = 0.022809 \times 10^{-3}$	Ramp Region 1&4	
$1.0 < t < 1.5$	$\Delta\Sigma_{a2} = 0.160000 \times 10^{-3}$	Ramp Region 1&2	
$1.5 < t < 2.9$	$\Delta\Sigma_{a2} = 0.340000 \times 10^{-3}$	Ramp Region 1&2	(B.1)

B.4 AECL Test Case Description

The original description of this test case can be found in reference [19]

Table B.4: AECL TEST CASE DESCRIPTION

Region(Group)	D_g (cm)	Σ_{r_g} (cm^{-1})	$\nu\Sigma_{r_g}$ (cm^{-1})	Σ_{21} (cm^{-1})
1,2,3,4 (1)	1.3100	1.0180E-2	0.00	1.018E-2
13,14,15,16 (2)	0.8695	2.1170E-4	0.0	
5,6,7,8,9 (1)	1.2640	8.1540E-3	0.0	7.3680e-3
10,17,18,19 (2)	0.9328	4.0140E-3	4.7230E-3	
20,21,22				
10,17,18,19 (2)	0.9328	4.0140E-3	4.7230E-3	
11,12,23,24 (1)	1.2640	8.1540E-3	0.0	7.3680e-3
(2)	0.9328	4.1000E-3	4.5620E-3	

$\chi_1 = 1.0 \quad \chi_2 = 0.0$
$\nu = 2.5$
$v_1 = 1.0 \times 10^7 cm/s \quad v_2 = 3.0 \times 10^5 cm/s$

Family_d	β_d	$\lambda_d(s^{-1})$
1	4.170E-4	1.244E-2
2	1.457E-3	3.063E-2
3	1.339E-3	1.139E-1
4	3.339E-3	3.079E-1
5	8.970E-4	1.198E+0
6	3.200E-4	3.212E+0

Transient Information

Σ_2, in regions 5, 6, 10, 11, 17, 12, 22 and 23 (Figures B.2, B.3 and B.4), varies linearly in time, with

$$\begin{aligned} \frac{\partial}{\partial t}\Sigma_2 &= -1.0e^{-4}(cms)^{-1}, for\ t \leq 0.4s \\ &= -8.8889e^{-6}(cms)^{-1}, for\ t > 0.4s \end{aligned} \tag{B.2}$$

$\Delta\Sigma_2$	$6.150 \times 10^{-4} cm^{-1}$
Insertion starts at	0.6 s
Absorber velocity	520. cm/ s

An incremental cross section, $\Delta\Sigma_2$, is added to regions 2, 4, 7, 9, 12, 14, 16, 18, 19, 21, 22, 23 and 24 to simulate asymmetric insertion of absorbers. The absorbers are inserted at constant velocity in the y-direction. The moving absorber boundary is parallel to the x-z plane and moves at the following rate:

Whenever perturbation overlap the rates of change of $\Delta\Sigma_2$ are additive.

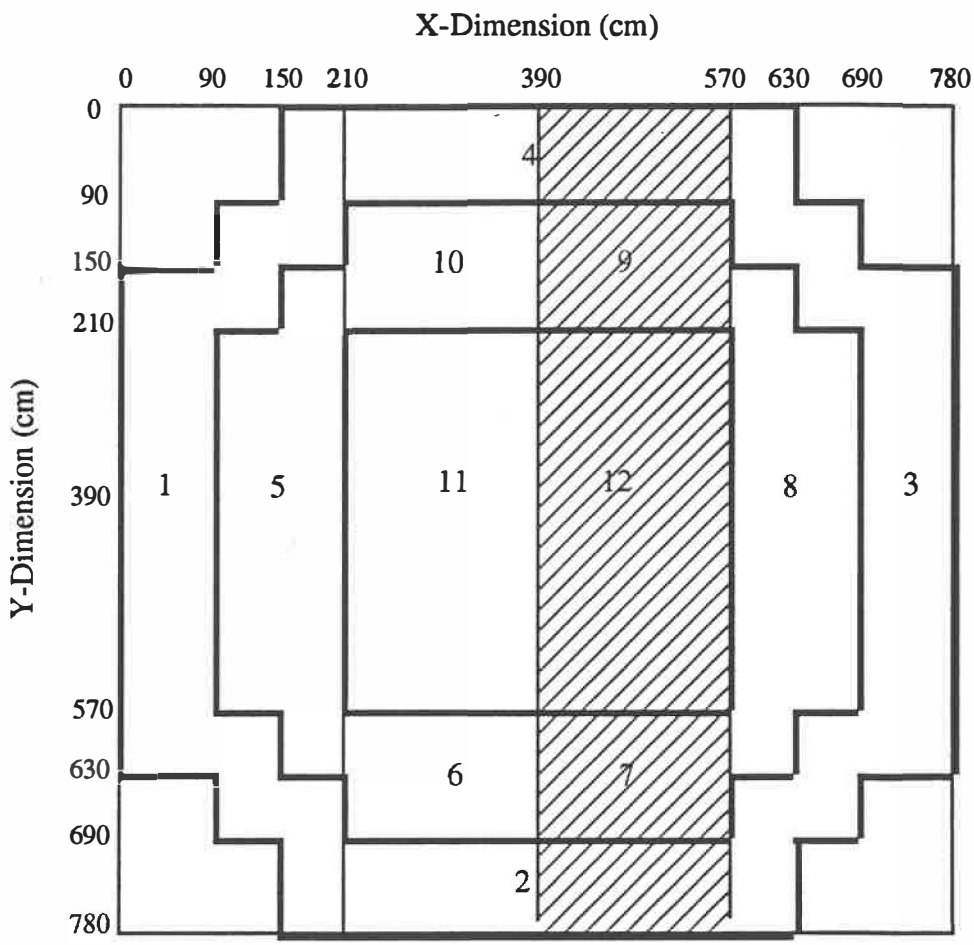


Figure B.3: Vertical cross-section at Z = 0 cm showing region assignment for 0 < Z < 300 cm

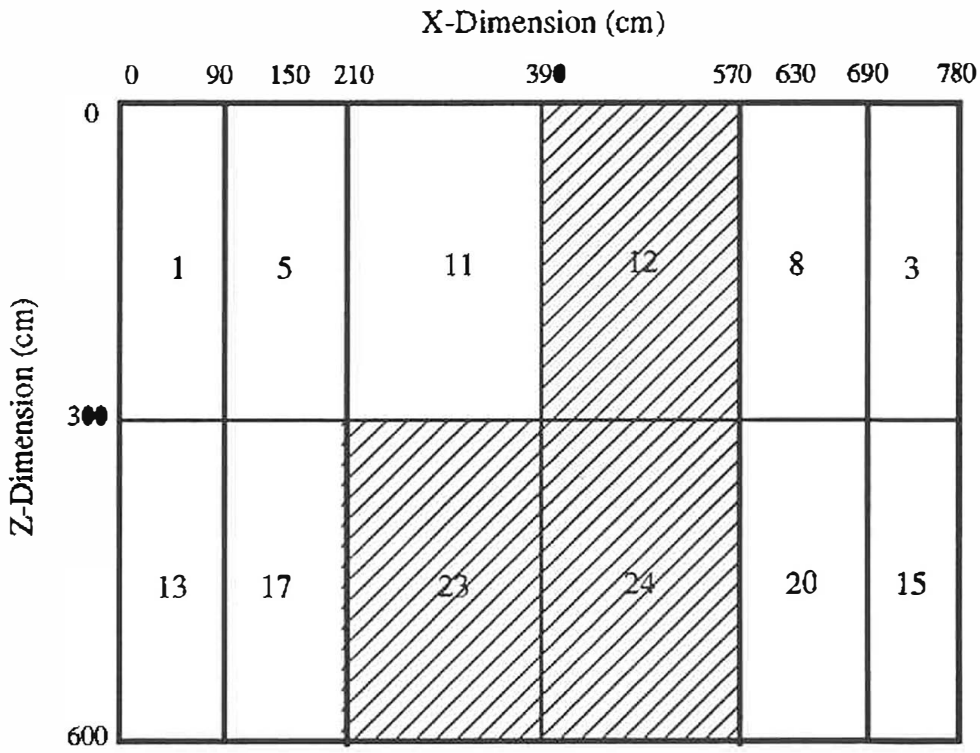


Figure B.4: Horizontal cross-section at Y = 390 cm showing region asssignment

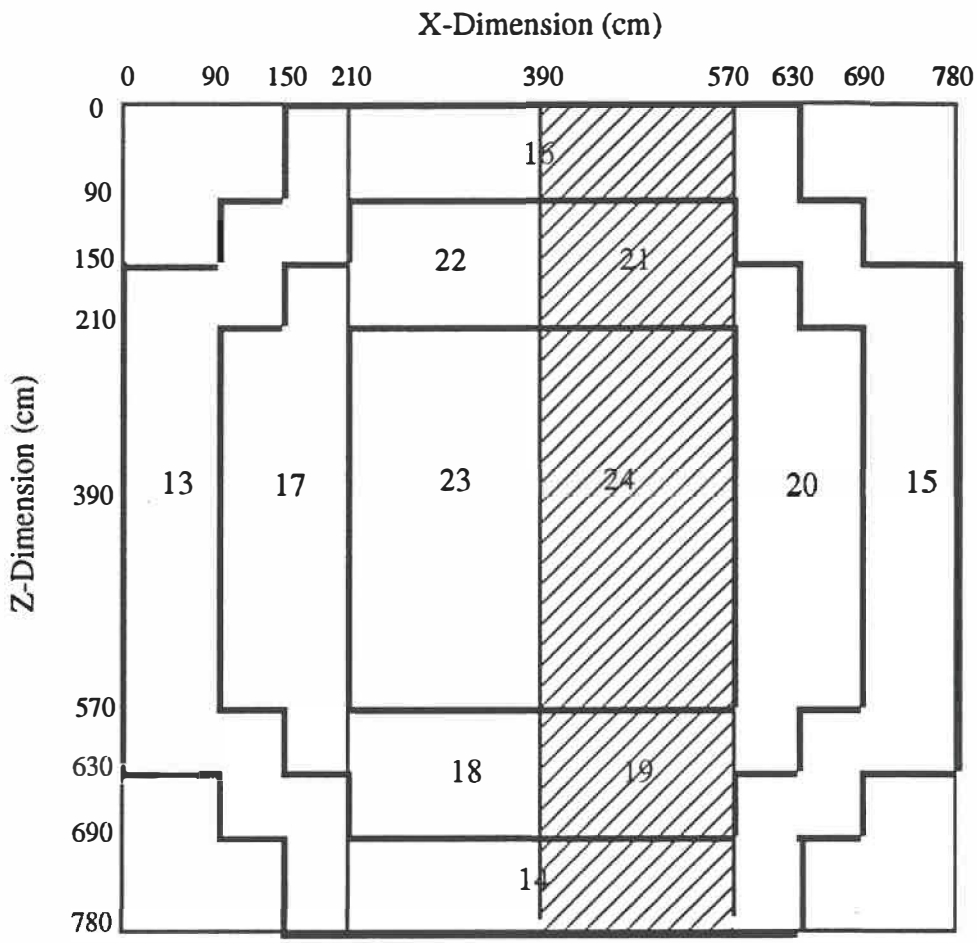


Figure B.5: Vertical cross-section at Z = 600 cm showing region assignment for 300 < Z < 600 CM

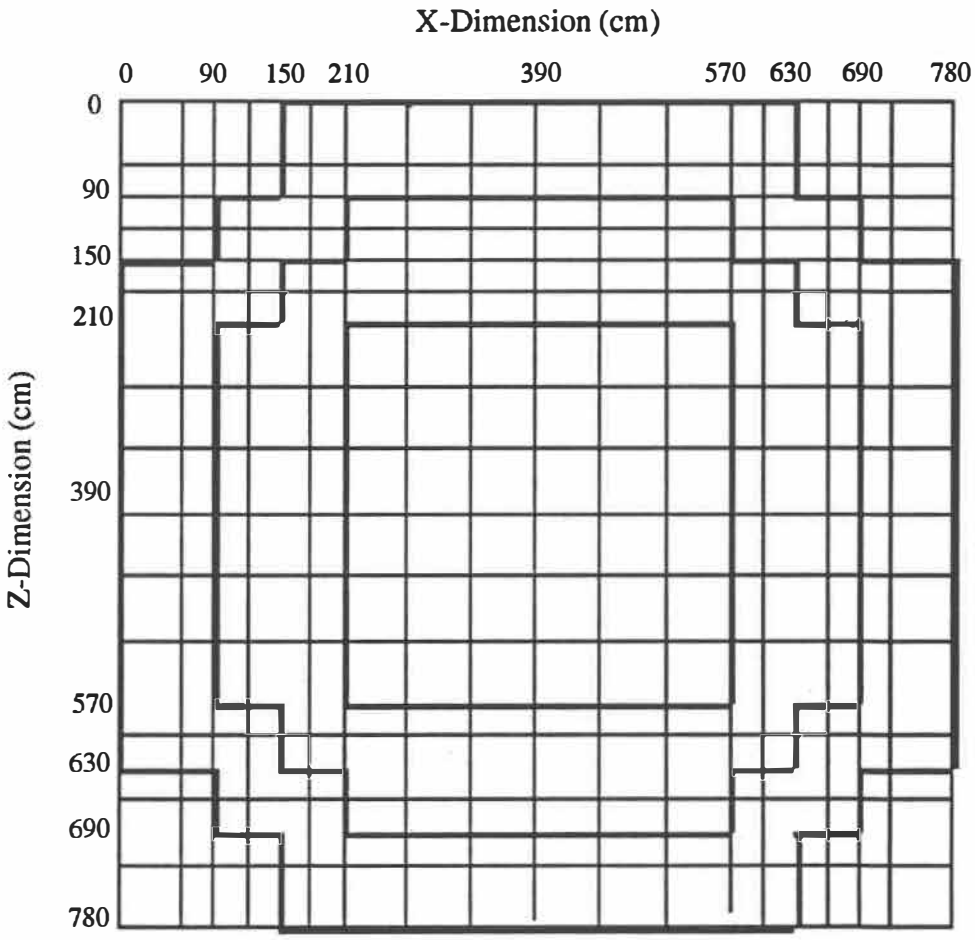


Figure B.6: Vertical cross-section at Z = 0 CM illustrating grid layout in the XY plane

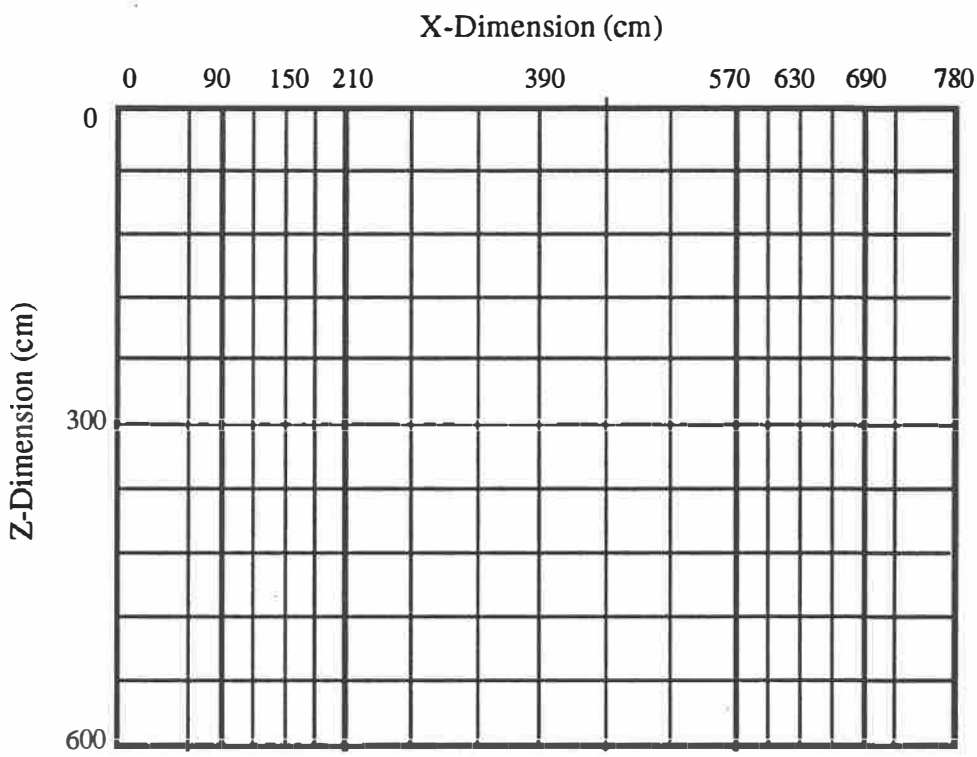


Figure B.7: Horizontal cross-section at Z = 390 cm illustrating grid layout in the XZ plane

B.4.1 Sample Input

The input data for the AECL test case is almost 2000 lines long and is not given here because of space considerations.

```
TRIVAC 1.4: CANDU 3-D AVEC TRANSITOIRE, AECL (LOCA)
MAXIMA     XYZ3 MCFD 1 18 1810      FIND
GEOM       PRED 18 18 2
   ZERO XINF XSUP ZERO YINF YSUP ZERO ZINF ZSUP
   (PLAN 1)
   0  0  0  0  1  1  1  1  1  5  5  5  1  1  0  0  0  0
   0  0  0  0  1  1  1  1  1  6  6  6  1  1  0  0  0  0
   0  0  1  1  1  1  3  3  3  7  7  7  1  1  1  1  0  0
   0  0  1  1  1  1  3  3  3  8  8  8  1  1  1  1  0  0
   1  1  1  1  3  3  3  3  3  9  9  9  2  2  1  1  1  1
   1  1  1  1  3  3  3  3  3 10 10 10  2  2  1  1  1  1
   1  1  3  3  3  3  4  4  4 11 11 11  2  2  2  2  1  1
   1  1  3  3  3  3  4  4  4 12 12 12  2  2  2  2  1  1
   1  1  3  3  3  3  4  4  4 13 13 13  2  2  2  2  1  1
   1  1  3  3  3  3  4  4  4 14 14 14  2  2  2  2  1  1
   1  1  3  3  3  3  4  4  4 15 15 15  2  2  2  2  1  1
   1  1  3  3  3  3  4  4  4 16 16 16  2  2  2  2  1  1
   1  1  1  1  3  3  3  3  3 17 17 17  2  2  1  1  1  1
   1  1  1  1  3  3  3  3  3 18 18 18  2  2  1  1  1  1
   0  0  1  1  1  1  3  3  3 19 19 19  1  1  1  1  0  0
   0  0  1  1  1  1  3  3  3 20 20 20  1  1  1  1  0  0
   0  0  0  0  1  1  1  1  1 21 21 21  1  1  0  0  0  0
   0  0  0  0  1  1  1  1  1 22 22 22  1  1  0  0  0  0
   (PLAN 2)
   0  0  0  0  1  1 23 23 23 23 23 23  1  1  0  0  0  0
   0  0  0  0  1  1 24 24 24 24 24 24  1  1  0  0  0  0
   0  0  1  1  1  1 25 25 25 26 26 26  1  1  1  1  0  0
   0  0  1  1  1  1 27 27 27 28 28 28  1  1  1  1  0  0
   1  1  1  1  3  3 29 29 29 30 30 30  2  2  1  1  1  1
   1  1  1  1  3  3 31 31 31 32 32 32  2  2  1  1  1  1
   1  1  3  3  3  3 33 33 33 34 34 34  2  2  2  2  1  1
   1  1  3  3  3  3 35 35 35 36 36 36  2  2  2  2  1  1
   1  1  3  3  3  3 37 37 37 38 38 38  2  2  2  2  1  1
   1  1  3  3  3  3 39 39 39 40 40 40  2  2  2  2  1  1
   1  1  3  3  3  3 41 41 41 42 42 42  2  2  2  2  1  1
   1  1  3  3  3  3 43 43 43 44 44 44  2  2  2  2  1  1
   1  1  1  1  3  3 45 45 45 46 46 46  2  2  1  1  1  1
   1  1  1  1  3  3 47 47 47 48 48 48  2  2  1  1  1  1
   0  0  1  1  1  1 49 49 49 50 50 50  1  1  1  1  0  0
   0  0  1  1  1  1 51 51 51 52 52 52  1  1  1  1  0  0
   0  0  0  0  1  1 53 53 53 53 53 53  1  1  0  0  0  0
   0  0  0  0  1  1 54 54 54 54 54 54  1  1  0  0  0  0
   0.0  60.0  90.0 120.0 150.0 180.0 210.0 270.0 330.0
 390.0 450.0 510.0 570.0 600.0 630.0 660.0 690.0 720.0
 780.0 BIS
   0.0 300.0 600.0
   SPLIT
    1  1  1  1  1  1  1  1  1
```

```
  1  1  1  1  1  1  1  1  1
 BIS
5  5 FIND

FLUX  IMPR 1 ADJ MAX 75 PREC 0.5E-5 SGD 2
  (   1; REF:   1 )
 1.3100  1.018E-2    0.0      0.0 *
 0.8695  2.117E-4    0.0      0.0    T    1.018E-2 *
  (   2; REF:   2 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
  (   3; REF:   3 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
  (   4; REF:   4 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
  (   5; REF:   7 )
 1.3100  1.018E-2    0.0      0.0 *
 0.8695  2.117E-4    0.0      0.0    T    1.018E-2 *
  (   6; REF:   7 )
 1.3100  1.018E-2    0.0      0.0 *
 0.8695  2.117E-4    0.0      0.0    T    1.018E-2 *
  (   7; REF:   8 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
  (   8; REF:   8 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
  (   9; REF:   8 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
  (  10; REF:   8 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *

  (  11; REF:   9 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
  (  12; REF:   9 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
  (  13; REF:   9 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
  (  14; REF:   9 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
  (  15; REF:   9 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
  (  16; REF:   9 )
 1.2640  8.154E-3    0.0      0.0 *
 0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
  (  17; REF:   8 )
```

```
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  18; REF:   8 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  19; REF:   8 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  20; REF:   8 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  21; REF:   7 )
1.3100  1.018E-2    0.0       0.0 *
0.8695  2.117E-4    0.0       0.0    T    1.018E-2 *
 (  22; REF:   7 )
1.3100  1.018E-2    0.0       0.0 *
0.8695  2.117E-4    0.0       0.0    T    1.018E-2 *
 (  23; REF:   7 )
1.3100  1.018E-2    0.0       0.0 *
0.8695  2.117E-4    0.0       0.0    T    1.018E-2 *
 (  24; REF:   7 )
1.3100  1.018E-2    0.0       0.0 *
0.8695  2.117E-4    0.0       0.0    T    1.018E-2 *
 (  25; REF:   5 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  26; REF:   8 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  27; REF:   5 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  28; REF:   8 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  29; REF:   5 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  30; REF:   8 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *

 (  31; REF:   5 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  32; REF:   8 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  33; REF:   6 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  34; REF:   9 )
1.2640  8.154E-3    0.0       0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  35; REF:   6 )
```

```
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  36; REF:   9 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  37; REF:   6 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  38; REF:   9 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  39; REF:   6 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  40; REF:   9 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *

 (  41; REF:   6 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  42; REF:   9 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  43; REF:   6 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  44; REF:   9 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.100E-3    4.562E-3 4.562E-3  T  7.368E-3 *
 (  45; REF:   5 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  46; REF:   8 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  47; REF:   5 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  48; REF:   8 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  49; REF:   5 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  50; REF:   8 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  51; REF:   5 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  52; REF:   8 )
1.2640  8.154E-3    0.0      0.0 *
0.9328  4.014E-3    4.723E-3 4.723E-3  T  7.368E-3 *
 (  53; REF:   7 )
```

```
   1.3100  1.018E-2    0.0      0.0 *
   0.8695  2.117E-4    0.0      0.0    T    1.018E-2 *
    (  54; REF:   7 )
   1.3100  1.018E-2    0.0      0.0 *
   0.8695  2.117E-4    0.0      0.0    T    1.018E-2 *
   * FIND

SORTIE INTG
    (PLAN 1)
    0  0  0  0  1  1  4  4  4  4  4  4  3  3  0  0  0  0
    0  0  0  0  1  1  4  4  4  4  4  4  3  3  0  0  0  0
    0  0  1  1  1  1 10 10 10  9  9  9  3  3  3  3  0  0
    0  0  1  1  1  1 10 10 10  9  9  9  3  3  3  3  0  0
    1  1  1  1  5  5 10 10 10  9  9  9  8  8  3  3  3  3
    1  1  1  1  5  5 10 10 10  9  9  9  8  8  3  3  3  3
    1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
    1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
    1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
    1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
    1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
    1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
    1  1  1  1  5  5  6  6  6  7  7  7  8  8  3  3  3  3
    1  1  1  1  5  5  6  6  6  7  7  7  8  8  3  3  3  3
    0  0  1  1  1  1  6  6  6  7  7  7  3  3  3  3  0  0
    0  0  1  1  1  1  6  6  6  7  7  7  3  3  3  3  0  0
    0  0  0  0  1  1  2  2  2  2  2  2  3  3  0  0  0  0
    0  0  0  0  1  1  2  2  2  2  2  2  3  3  0  0  0  0
    (PLAN 2)
    0  0  0  0 13 13 16 16 16 16 16 16 15 15  0  0  0  0
    0  0  0  0 13 13 16 16 16 16 16 16 15 15  0  0  0  0
    0  0 13 13 13 13 22 22 22 21 21 21 15 15 15 15  0  0
    0  0 13 13 13 13 22 22 22 21 21 21 15 15 15 15  0  0
   13 13 13 13 17 17 22 22 22 21 21 21 20 20 15 15 15 15
   13 13 13 13 17 17 22 22 22 21 21 21 20 20 15 15 15 15
   13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
   13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
   13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
   13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
   13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
   13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
   13 13 13 13 17 17 18 18 18 19 19 19 20 20 15 15 15 15
   13 13 13 13 17 17 18 18 18 19 19 19 20 20 15 15 15 15
    0  0 13 13 13 13 18 18 18 19 19 19 15 15 15 15  0  0
    0  0 13 13 13 13 18 18 18 19 19 19 15 15 15 15  0  0
    0  0  0  0 13 13 14 14 14 14 14 14 15 15  0  0  0  0
          0  0  0  0 13 13 14 14 14 14 14 14 15 15  0  0  0  0
IMPR FIND
TRAN   6
    ( BETA    LAM BDA )
    4.170E-4 1.244E-2
    1.457E-3 3.063E-2
    1.339E-3 1.139E-1
    3.339E-3 3.079E-1
    8.970E-4 1.198E+0
    3.200E-4 3.212E+0
```

```
( V1    V2 )
1.0E7 3.0E5
CONV 0.5E-6 5.0E-3 1.0E-5 15 IMPR 0 MAX 12
QSM  OPTION FASTQ RENORM *
*

      ($  1)
PERT 1 RAMP 0.10000E+00 DSGD 2  (T=0.1)
              3
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
         4
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        25
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        27
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        29
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        31
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        33
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        35
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        37
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        39
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        41
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        43
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        45
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        47
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        49
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        51
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
```

```
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
* SHOW REAC 1 * END
          (.$  1)
          ($  2)

PERT 1 RAMP 0.10000E+00 DSGD 2  (T=0.2)
        3
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
        4
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       25
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       27
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       29
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *

       31
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       33
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       35
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       37
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       39
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       41
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       43
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       45
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       47
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       49
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
       51
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
```

```
    * SHOW REAC 1 * END

OUTPUT INTG
     (PLAN 1)
     0  0  0  0  1  4  4  4  4  4  4  4  3  3  0  0  0  0
     0  0  0  0  1  4  4  4  4  4  4  4  3  3  0  0  0  0
     0  0  1  1  1  1 10 10 10  9  9  9  3  3  3  3  0  0
     0  0  1  1  1  1 10 10 10  9  9  9  3  3  3  3  0  0
     1  1  1  1  5  5 10 10 10  9  9  9  8  8  3  3  3  3
     1  1  1  1  5  5 10 10 10  9  9  9  8  8  3  3  3  3
     1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
     1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
     1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
     1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
     1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
     1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
     1  1  1  1  5  5  6  6  6  7  7  7  8  8  3  3  3  3
     1  1  1  1  5  5  6  6  6  7  7  7  8  8  3  3  3  3
     0  0  1  1  1  1  6  6  6  7  7  7  3  3  3  3  0  0
     0  0  1  1  1  1  6  6  6  7  7  7  3  3  3  3  0  0
     0  0  0  0  1  1  2  2  2  2  2  2  3  3  0  0  0  0
     0  0  0  0  1  1  2  2  2  2  2  2  3  3  0  0  0  0
     (PLAN 2)
     0  0  0  0 13 13 16 16 16 16 16 16 15 15  0  0  0  0
     0  0  0  0 13 13 16 16 16 16 16 16 15 15  0  0  0  0
     0  0 13 13 13 13 22 22 22 21 21 21 15 15 15 15  0  0
     0  0 13 13 13 13 22 22 22 21 21 21 15 15 15 15  0  0
    13 13 13 13 17 17 22 22 22 21 21 21 20 20 15 15 15 15
    13 13 13 13 17 17 22 22 22 21 21 21 20 20 15 15 15 15
    13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
    13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
    13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
    13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
    13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
    13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
    13 13 13 13 17 17 18 18 18 19 19 19 20 20 15 15 15 15
    13 13 13 13 17 17 18 18 18 19 19 19 20 20 15 15 15 15
     0  0 13 13 13 13 18 18 18 19 19 19 15 15 15 15  0  0
     0  0 13 13 13 13 18 18 18 19 19 19 15 15 15 15  0  0
     0  0  0  0 13 13 14 14 14 14 14 14 15 15  0  0  0  0
           0  0  0  0 13 13 14 14 14 14 14 14 15 15  0  0  0  0
IMPR FIND
              (.$  2)


              ($  3)
    PERT 1 RAMP 0.10000E+00 DSGD 2 (T=0.3)
           3
    0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
    0.00000E+00   -0.10000E-04    0.00000E+00    0.00000E+00   *
           4
    0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00   *
```

```
PERT 1 RAMP 0.10000E+00 DSGD 2 (T=2.5 S)
                3
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
         4
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        25
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        27
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        29
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        31
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        33
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        35
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        37
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        39
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        41
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        43
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        45
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        47
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        49
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
        51
0.00000E+00    0.00000E+00    0.00000E+00    0.00000E+00    *
0.00000E+00   -0.88889E-06    0.00000E+00    0.00000E+00    *
* SHOW REAC 1 * END
          (.$ 28)

OUTPUT INTG
     (PLAN 1)
```

```
 0  0  0  0  1  4  4  4  4  4  4  4  3  3  0  0  0  0
 0  0  0  0  1  4  4  4  4  4  4  4  3  3  0  0  0  0
 0  0  1  1  1  1 10 10 10  9  9  9  3  3  3  3  0  0
 0  0  1  1  1  1 10 10 10  9  9  9  3  3  3  3  0  0
 1  1  1  1  5  5 10 10 10  9  9  9  8  8  3  3  3  3
 1  1  1  1  5  5 10 10 10  9  9  9  8  8  3  3  3  3
 1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
 1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
 1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
 1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
 1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
 1  1  5  5  5  5 11 11 11 12 12 12  8  8  8  8  3  3
 1  1  1  1  5  5  6  6  6  7  7  7  8  8  3  3  3  3
 1  1  1  1  5  5  6  6  6  7  7  7  8  8  3  3  3  3
 0  0  1  1  1  1  6  6  6  7  7  7  3  3  3  3  0  0
 0  0  1  1  1  1  6  6  6  7  7  7  3  3  3  3  0  0
 0  0  0  0  1  1  2  2  2  2  2  2  3  3  0  0  0  0
 0  0  0  0  1  1  2  2  2  2  2  2  3  3  0  0  0  0
 (PLAN 2)
 0  0  0  0 13 13 16 16 16 16 16 16 15 15  0  0  0  0
 0  0  0  0 13 13 16 16 16 16 16 16 15 15  0  0  0  0
 0  0 13 13 13 13 22 22 22 21 21 21 15 15 15 15  0  0
 0  0 13 13 13 13 22 22 22 21 21 21 15 15 15 15  0  0
13 13 13 13 17 17 22 22 22 21 21 21 20 20 15 15 15 15
13 13 13 13 17 17 22 22 22 21 21 21 20 20 15 15 15 15
13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
13 13 17 17 17 17 23 23 23 24 24 24 20 20 20 20 15 15
13 13 13 13 17 17 18 18 18 19 19 19 20 20 15 15 15 15
13 13 13 13 17 17 18 18 18 19 19 19 20 20 15 15 15 15
 0  0 13 13 13 13 18 18 18 19 19 19 15 15 15 15  0  0
 0  0 13 13 13 13 18 18 18 19 19 19 15 15 15 15  0  0
 0  0  0  0 13 13 14 14 14 14 14 14 15 15  0  0  0  0
 0  0  0  0 13 13 14 14 14 14 14 14 15 15  0  0  0  0
IMPR
FIND
FIN
```

APPENDIX C

User Guide for XSTATIC

The input data for XSTATIC is represented as a sequence of structures or phases that define all aspects of the calculation. For the syntax and organization of a structure or phase not defined here, refer to the TRIVAC user guide [20].

The XSTATIC input data can be represented as:

TITRE

MAXI (group 1) FIND

GEOM (group 2) FIND

FLUX (group 3) FIND

TRAN (group 4) FIND

[[PERT (group 5) FIND [OUTP (group 6) FIND]]]

The first three phases (MAXI, GEOM, FLUX) define all the data necessary for the static calculation. The next phase TRAN sets the data that will remain invariant throughout the transients calculation. The next two phases, PERT and OUTP , define the transient and the output desired, respectively.

C.1 Description of phase TRAN

This compulsory substructure (group 4) defines the data that remains invariant through out the transient, ie: the number of delayed groups, the associated delayed fractions and decay constants, and the convergence criteria are set. This phase has the following form:

NDG ($\beta(i)$ $\lambda(i)$, i=1,NDG) V1 V2

[POWE PWRINT] [MAX MAXX]

[OPTI [FAST] [RENO] *]

[CONV EPSPK EPSRHO EPSKEF EPSFER LIMM]

where:

NDG	number of delayed groups
$\beta(i)$	delayed fraction for the i-th group
$\lambda(i)$	decay rate for the i-th group
V1	velocity of neutrons in the fast group
V2	velocity of neutrons in the thermal group

The optional keyword POWE allows the initial reactor power to be set to the arbitrary non zero constant PWRINT. The default value is 1.0.

The optional keyword MAX sets the maximum iterations per shape function calculation. The default value is 40.

The optional keyword OPTI permits the user to specify the FAST and RENO options. FAST forces the code to integrate the point kinetics using FASTQ and RENO forces the code to use the constrained version of the IQS method.

The optional keyword CONV permits the user to specify he following convergence criteria:

where:

		Default
EPSPK	maximum error in solution of point kinetics equations	10^{-4}
EPSRHO	convergence criteria for reactivity in mill- k	10^{-2}
EPSKEF	convergence criteria for k-eff in in shape function calculation	10^{-6}
EPSFER	convergence criteria for error in shape function calculation 10^{-4}	
LIMM	maximum number of reactivity iterations	6

C.2 Description of the phase PERT

This substructure (group 5) defines one step in the transient. A step is defined as a set of changes in nuclear properties and the time step over which they occur. Group 5 has the following structure

{ RAMP | STEP | CERB } ΔT

[DSGD

NG [[I (DD(I,IG) DSR(I,G) DSF(I,IG) DH(I,IG) [T DST(I,IG)] * ,IG=1,NG)]]

]

where:

RAMP	the perturbations are considered a ramp over this time step
CERB	the perturbations are considered a ramp over this time step using CERBERUS variation of the point kinetics parameters
STEP	the perturbation is considered a step
ΔT	the length of the time step

The variations in the nuclear properties, which define the transient, are given following the keyword DSGD . Should this keyword be absent, a null variation is assumed. For more information on the DSGD group see the TRIVAC user guide [20].

ÉCOLE POLYTECHNIQUE DE MONTRÉAL
3 9334 00291738 1