NUMERICAL METHODS TO SOLVE THE FUEL DEPLETION EQUATIONS FOR A NUCLEAR REACTOR

P.P. KRÜGER

Dissertation submitted in partial fulfilment of the requirements for the degree Magister Scientiae in Physics at the Potchefstroomse Universiteit vir Christelike Hoër Onderwys

Supervisor: Prof. H. Moraal (PU vir CHO, Potchefstroom)
Co-supervisor: Dr. W.R. Joubert (NECSA, Pelindaba)

2004
Potchefstroom

Abstract

Different numerical methods for solving the fuel depletion equations in the depletion module of a reactor core analysis system are compared with each other, in order to find the best method and the best implementation of this method to be used in NECSA's OSCAR system. It is assumed that the neutron flux is constant with time over each burnup step, giving a linear system of first order differential equations with constant coefficients that has to be solved. Using the special properties of the fuel depletion equations of stiffness, sparseness and essential-nonnegativity, it is shown how the various methods can be optimally implemented. By using a sample global reactor depletion and an assembly depletion problem, it is then shown that the Taylor expansion method, using a generalised uniformization technique, performs the best under most circumstances.

Acknowledgements

I want to thank Dr. Wessel Joubert and Dr. Djordje Tomašević who where always willing to help and to share their deep knowledge and experience in the field or reactor modelling. This work was done with the support of NECSA and I want to thank the people at NECSA, especially Coenie Stoker, who made this dissertation possible. Also to Zain Karriem and my colleagues at PBMR with whom I could work together.

Special thanks to Prof. Harm Moraal for his valuable advice and suggested corrections when I was writing this dissertation and especially the trouble he took when he was overseas.

To all my friends and family: I appreciate all your unlimited patience and support.

Above all I what to praise my Heavenly Father who gave me all the opportunities, talents and strength needed.

Paulus Krüger

Contents

Abstract

Different numerical methods for solving the fuel depletion equations in the depletion module of a reactor core analysis system are compared with each other, in order to find the best method and the best implementation of this method to be used in NECSA's OSCAR system. It is assumed that the neutron flux is constant with time over each burnup step, giving a linear system of first order differential equations with constant coefficients that has to be solved. Using the special properties of the fuel depletion equations of stiffness, sparseness and essential-nonnegativity, it is shown how the various methods can be optimally implemented. By using a sample global reactor depletion and an assembly depletion problem, it is then shown that the Taylor expansion method, using a generalised uniformization technique, performs the best under most circumstances.

Samevatting

Verskeie numeriese metodes, wat gebruik kan word om die brandstofbrandingsvergelykings in die verbrandingsmodule van 'n analisestelsel van 'n reaktorkern op te los, word met mekaar vergelyk om daardeur die beste metode en die beste implementering van die metode te bepaal. Deur te aanvaar dat die neutron vloed nie met tyd verander tydens elke verbrandingstap nie, word 'n lineêre stelsel van eerste orde differensiaalvergelykings met konstante koëffisiënte gevorm wat opgelos moet word. Deur die brandstofbrandingsvergelykings se spesiale eienskappe van styfheid, ylheid en essensieel-nie-negatiwiteit te gebruik, word aangetoon hoe die verskeie metodes optimaal geïmplementeer kan word. 'n Globale reaktorverbrandings en elementverbrandings probleem is as voorbeeld gebruik om aan te toon dat die Taylor uitbreidingsmetode, wat gebruik maak van die veralgemeende uniformisasietegniek, die beste is om onder die meeste omstandighede te gebruik.

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Nuclear energy industry

Since the first reactor went critical in 1942, nuclear reactors developed from expensive research tools into economically competitive power plants. Today there are 438 nuclear power plants in operation, producing 16% of the world's electricity. This is the largest share produced by any non-greenhouse-gas-emitting source. It is expected that in the future, nuclear energy will not only be able to meet the growing demand in electricity, but that it could also be used as an energy source in district heating, to generate hydrogen and to desalinate water (NERAC, 2002:1). Although the current generation of power plants produce electricity at competitive cost, they have very high construction costs (NERAC, 2002:5). South Africa is one of ten countries that are part of the Generation IV International Forum to develop future-generation nuclear energy systems. The main goals of this new generation of nuclear power plants are sustainability, economic viability, safety and reliability.

For the optimal design, construction, operation and decommissioning of a new nuclear power plant, a fundamental knowledge of the physical processes in the plant is necessary. Because of the complexity of the processes, a wide range of computer codes is used to help to understand, model and predict the behavior of nuclear power plants. To meet the four goals of sustainability, economic viability, safety and reliability, it is important to be able to model the behavior of a power plant as accurately as possible beforehand. For example, to meet the goal of safety, a minimum amount of radioactive fission products must be released to the outside. It is therefore important to model the amount of fission products in a reactor and their behavior under various accident conditions. Another example is that to be economical in operation, the power plant must make optimum use of the fuel and it is therefore important to be able to model the burnup of the fuel, the generation and transport of heat to the generator, and the efficiency of the generator in producing electricity.

South Africa is highly involved in the nuclear energy industry. It not only has a nuclear energy plant, Koeberg, but is also developing a Generation IV nuclear energy plant, called the Pebble Bed Modular Reactor (PBMR). To aid in the modelling of the reactors, NECSA (Nuclear Energy Corporation of South Africa) has a Radiation and Reactor Theory (RRT) group that is involved in theoretical reactor physics analysis. This group focuses on the research, development and implementation of

advanced calculation methods in the field of reactor physics. The RRT group has developed OSCAR, a reactor core analysis system, and is also working on defining and developing next generation core analysis tools for the PBMR.

A *reactor core analysis system* consists of a number of modules that solve different types of equations that describe the behavior of the core of a nuclear reactor. Some of the main modules (Duderstadt & Hamilton, 1975:461) are the thermal-hydraulic module that computes the temperature, pressure and density profiles in the core, the flux-power-reactivity module that solves the Boltzmann transport equation, and the depletion module that solves the equations which describe the amount of each isotope that is present in the reactor core.

The huge increase in computer speed and memory over the past few years, along with new numerical methods that are constantly being developed, makes it possible to model nuclear reactors, and therefore also nuclear power plants, in much more detail than a few years ago. To refine the modelling of reactors, the older computer codes have to be updated or rewritten. The purpose of this dissertation is to investigate various numerical methods that can be used to improve the depletion module in the OSCAR system.

First some terms, that will be used in the rest of the dissertation, are introduced in Section 1.2 before a general overview of the depletion module is given in Section 1.3. In Section 1.4 the purpose of the dissertation is given in more detail. Section 1.5 gives an overview of current computer codes and methods that are available and used in the depletion module, and the specific goals of the dissertation. Section 1.6 gives the criteria that will be used in comparing the different codes and methods and Section 1.7 gives the methodology that is followed in this dissertation.

1.2 Nuclear fission reactor core

A *nuclear power plant* utilizes the energy released by nuclear fission reactions inside a reactor core. The energy is released inside a large number of *fuel rods* that are surrounded by coolant that transports the heat away from the core. The *fission reactions* occur when a *fissile isotope*, like ^{235}U, splits or fissions into two lighter isotopes, called *fission products* (Stamm'ler & Abbate, 1983:378). These reactions normally occur when a fissile isotope captures a *free neutron*, but it can also occur spontaneously. In each fission reaction few free neutrons are released, which are necessary to establish a chain reaction. For a continuous release of energy, the number of free neutrons must stay constant over time, and the reactor is then called *critical*. If the number of neutrons increases or decreases over time, the reactor is called *super-critical* or *sub-critical* respectively. The number of neutrons is controlled by inserting or withdrawing isotopes, called *poison*, that absorb neutrons.

One type of fissile isotope can split into a large number of different fission products. The probability that a fissile isotope will give a certain fission product is called the *fission yield* of the specific fission product. Many of the fission products are unstable isotopes and undergo radioactive decay. When an isotope undergoes a reaction to form a new isotope, the original isotope is called the *parent isotope* or *precursor* of the new isotope, called the *daughter isotope*. Some of the fission products

undergo a long chain of radioactive decay before a stable isotope is reached. The fission products can also capture a free neutron forming a new isotope, which is usually unstable again. Some isotopes, called *fertile isotopes*, can become a fissile isotope after capturing a neutron. Because of all the possible reactions that an isotope can take part in, the reactor core contains a large number of different chemical elements and even a number of different isotopes of one element.

1.3 Depletion module

The change in the amount of fissile isotopes and fission products in a fuel element over time is referred to as fuel *burnup* or fuel *depletion*. It is measured as the amount of energy that is released for a given amount of initial fissile isotopes and the unit megawatt-days per ton (MWD/t) is typically used.

There are a number of reasons why it is important to keep track of the fuel burnup for each fuel element in the reactor core:

- Some fission products, like ^{135}Xe, can very easily capture neutrons (it has a large neutron capture cross section), which makes the reactor sub- or super-critical if the amounts of these isotopes change. It is important to monitor the build-up of these isotopes so that control adjustments can be made to keep the reactor critical.
- During burnup the amount of fissile isotopes decreases and the amount of fission products increases. After some time some fuel elements reach a burnup level where much more free neutrons are absorbed than produced in the fuel element and the fuel element must then be replaced.
- To obtain a uniform distribution of energy production in the core, fuel with different burnup levels is placed at different positions in the reactor core. This is done by placing fuel with a higher burnup level at places where there are more neutrons and fuel that has burnt less at places where there are fewer neutrons, in order to get the same fission rate per fuel element. It is therefore sometimes necessary to swap fuel elements with different burnup levels around inside the core to get the desired optimal distribution of energy production.
- Some of the fission products that are formed are in the gas phase. As these fission products build up, the fuel element can get pressurized. Certain fission products can also leak out of the fuel elements into the coolant. During the design of the fuel elements, it is therefore necessary to know how much of each fission product will be formed.

The depletion module in a reactor core analysis system gives the amount of each isotope as a function of time. This is done by solving the isotope depletion equations which describe the rate of change in the amount of each isotope in the core due to radioactive decay, neutron capture and

fission as described in Section 1.2. Using certain approximations, that will be described in Chapter 2, these depletion equations can be written as a system of ordinary first order differential equations

$$\frac{d\vec{N}(\vec{r},t)}{dt} = \mathbf{A}\vec{N}(\vec{r},t), \tag{1.1}$$

with the initial conditions at time $t = 0$

$$\vec{N}(\vec{r},0) = \vec{N}_0(\vec{r}), \tag{1.2}$$

where $\vec{N}$ is a vector containing the number densities of all the isotopes. The matrix $\mathbf{A}$ is called the coefficient matrix and it contains all the radioactive decay constants, the fission yields of the fissile isotopes, and the neutron capture terms. The main purpose of the depletion module is therefore to solve a system of fuel depletion equations having the form of (1.1) and (1.2). Because the purpose of this dissertation is to find an optimal method for the depletion module, various methods to solve such systems will be explored in this dissertation.

During reactor core analyses, the depletion module is used in two different types of calculations: global reactor calculations and assembly calculations.

1.3.1 Global reactor calculations

In *global reactor calculations*, also called *core calculations*, the properties and behaviour of the reactor core as a whole are calculated. In light water reactors a number of fuel rods and control rods are placed together into a *fuel assembly*. The reactor core is then made up of a number of such fuel assemblies placed next to each other. To simplify the analysis of the whole reactor core, the assemblies are not modelled in detail, but a simplified model is used for each assembly. The parameters of these simplified models are chosen such that the models give (almost) the same answer in the global reactor calculation that a detail model of the assembly would have given. These parameters are then called *equivalent parameters*.

The global reactor calculations are done over large time intervals, that can span the whole lifetime of the core. Because of the coupling between the neutron flux and rate of change of the isotope number densities (as will be explained in Chapter 2), large time intervals must be divided into smaller time intervals and the depletion module is then used to calculate the depletion in each small time interval. This results in the depletion module being called many times in a single core calculation, requiring that the depletion module should be able to solve the depletion equations very fast.

In a core calculation, all the isotopes are not important and the depletion module needs not keep track of all of them. The depletion module only has to account for the fuel burnup, the conversion of fertile isotopes to fissile isotopes and the build-up of fission products (Duderstadt & Hamilton, 1975:464). This requires only a few important isotopes to be modelled explicitly. The remaining isotopes are grouped together into a few pseudo-fission products (Stamm'ler & Abbate, 1983:378). Table B.1 gives an example of the isotopes in a typical core calculation.

1.3.2 Assembly calculations

In order to use simplified assembly models that behave the same as detailed assembly models, equivalent parameters must be calculated for each assembly. This is done by modelling each assembly in detail on its own. Because the behavior of the assemblies are not independent of each other, the effect of the other assemblies is approximated by suitable boundary conditions. As the burnup of the assembly changes, the equivalent parameters also change. The calculation of the parameters is therefore done at different fuel burnup levels and tabulated (Stamm'ler & Abbate, 1983:389). In the global reactor calculation, the necessary equivalent parameters are then interpolated between the tabulated values.

In the assembly calculations fewer isotopes, if any, are grouped together than in a global reactor calculation, resulting in a system of depletion equations that can consist of more than a hundred isotopes. Some of these isotopes have half-lives that are much shorter than the time over which the depletion equations are solved. When this is the case, the equation is called stiff and the stability and roundoff errors of numerical methods becomes a problem as will be explained in Chapters 3 and 5.

1.4 Motivation and purpose

The reactor core analysis systems that NECSA has developed uses the GAUGE code (Wagner, 1968) in its depletion module to solve the depletion equations. This code performs well in both global reactor and assembly calculations. However, the method imposes restrictions on the depletion mechanisms that can be described (Wagner, 1968:28). It must be possible to arrange the isotopes in a list, so that the daughter isotopes of all the reactions will be lower in the list than their parents. This can almost always be done for radio-active decay reactions, but when an isotope captures a neutron it can be its own precursor. For example if a neutron capture reaction is followed by a neutron decay reaction, the initial isotope is obtained again. When such reactions, called *feedback reactions*, are included, the isotopes can not be arranged in a downward list. With the GAUGE code it is therefore impossible to include any feedback reactions in the fuel depletion equations.

The purpose of this dissertation is to find a calculational method for the depletion module that performs as well, or better, as the current code with respect to accuracy, stability and speed, but without this restriction. In assembly calculations, where there are a large number of simultaneous equations to be solved, having a wide range of time constants, stability and roundoff errors become a problem. The new method must therefore be stable and accurate for stiff problems. In global reactor calculations, the number of equations is substantially less, but computational speed is critical. The new method must therefore also be fast, especially for small problems. These criteria will be discussed in Section 1.6. In the next section it is first shown which computer codes are currently available and what their shortcomings are.

1.5 Review of current methods and computer codes

Before reviewing current methods and computer codes that are used to solve the fuel depletion equations, the difference between the terms method, algorithm and implementation, as used in this dissertation, are first explained.

1.5.1 Definition of methods, algorithms and implementations

A *method* is a particular procedure for accomplishing or approaching something. In this dissertation a method is the particular procedure, consisting of a number of mathematic equations, that can be used to calculate the solution of the system of fuel depletion equations. These equations are mathematically derived by using mathematical identities and approximations. A method for solving the fuel depletion problem (1.1) and (1.2) is therefore some set of equations that can be used to calculate the isotope concentration $\vec{N}(\vec{r},t)$ at a time t, by using the coefficient matrix $\mathbf{A}$ and the initial isotope concentration $\vec{N}_0(\vec{r})$.

An *algorithm* is a process or set of rules to be followed in calculations and in this dissertation it describes the way and order in which the equations of a method must be computed. Because the order in which some equations are evaluated is not mathematically important, there are normally many different algorithms for the same method, differing only in the order in which the equations are evaluated. But, although these algorithms are mathematically equivalent, when computer time and roundoff errors are considered, they could behave differently.

A certain algorithm of a certain method is *implemented* (put into effect) by using a specific computer language, specific routines and a specific compiler, resulting in a piece of *computer code* that can be executed on a certain type of computer. During the implementation, important decisions, like how the computer memory is used and which type of floating point representation is used, must be made. These decisions could have a significant influence on the computer time required by the implementation, and the accuracy of the final results.

1.5.2 Methods

The solution of the system of first order linear differential equations, Eqs. (1.1) and (1.2), is formally

$$\vec{N}(\vec{r},t) = e^{\mathbf{A}t}\vec{N}_0(\vec{r}), \tag{1.3}$$

when the coefficient matrix $\mathbf{A}$ is independent of time. Solving the system of differential equations is therefore equivalent to computing the exponential of a matrix. Moler and Van Loan (1978) give a review of methods that can be used to compute the exponential of a matrix and conclude that all the methods are 'dubious' in the general case and that the best method will depend on the details of implementation and on the particular problem being solved (Moler & Van Loan, 1978:828). They also state (Moler & Van Loan, 1978:827) that the Padé approximation method using scaling and squaring is the best generally competitive series method when implemented properly.

A relatively new numerical method that is being used to compute the matrix exponential is the Krylov subspace approximation method. In this method the exponential of a large matrix is projected onto a small Krylov subspace. The matrix exponential calculation is then performed on the much smaller matrix (Saad, 1992:209). Saad (1992:226) showed that these techniques can provide an effective tool for approximating the matrix exponential. Modern routines aimed at computing the matrix exponential, like EXPOKIT, use a combination of Krylov subspace approximations and Padé approximations (Sidje, 1998).

Except for the newer Krylov subspace approximation, the methods that are used in this dissertation are those discussed by Moler and Van Loan (1978).

1.5.3 Implementations

Although there is a wide range of well known numerical methods to solve equations of the form of Eq. (1.1), the details of how the methods are implemented for the specific problem can have a large influence of the performance of these methods (Moler & Van Loan, 1978:801). Because general computer codes to solve these equations are far from optimal, it is necessary to develop calculation methods that are optimally implemented to solve the system of depletion equations (1.1).

The GAUGE code, developed by Gulf General Atomic in 1968 (Wagner, 1968), solves the fuel depletion equations analytically. A triangular form is assumed for the coefficient matrix, excluding all feedback reactions from the depletion chain (Wagner, 1968:28) as described in Section 1.4 above. This restricts the depletion mechanisms that can be described. A further problem of this analytical method is that when two eigenvalues of the coefficient matrix are the same (confluent) or nearly the same, large roundoff errors can be introduced as will be shown in Chapter 4.

The restriction of the GAUGE code can be overcome by triangulating the coefficient matrix or by treating the feedback terms as small perturbations. Moler and Van Loan (1978:823) show how a general matrix can be block triangulated using a Schur decomposition method. If the feedback coefficients in the coefficient matrix are treated as small perturbations, the preconditioning method of Castillo and Saad (1998) can be used to generalise the GAUGE method. These methods will be further discussed in Chapter 4 as no depletion code is known to the author that uses these two methods.

The well known ORIGEN-S code (Hermann & Westfall, 1998), developed by Oak Ridge National Laboratory, uses the Taylor series expansion method to solve the depletion equations. This method can not be used for stiff systems (Moler & Van Loan, 1978:807), making it necessary to remove the short half-life isotopes by assuming that they are in equilibrium. Because of this assumption, the computed isotope densities asymptotically approach the correct values over time as successive time intervals are executed, and at least five to ten time intervals must be used (Hermann & Westfall, 1998:F.7.2.7). Thomas and Barber (1994:309) showed that the ORIGEN-S code can have significant departures from the exact results at short depletion times because of the removal of the short half-life isotopes.

Although there are many, well analysed, methods that can be used to solve the fuel depletion problem, only a few of them are implemented in depletion modules. To find the best method to be used in the depletion module, which is the goal of this dissertation, it is therefore first necessary to develop implementations for various methods, in order to be able to compare the methods against each other.

1.6 Criteria used for comparing methods

In order to compare the different methods and their implementations, some criteria are necessary. Moler and Van Loan (1978) give the following attributes, listed in decreasing order of importance, to assess the value of a numerical method in calculating matrix exponentials:

Generality: The generality of a method is the range of classes of matrices for which the method can solve the matrix exponential. In solving the depletion equations, the generality will be the range of depletion mechanisms that can be included in the coefficient matrix.

Reliability: A reliable method gives an indication of the global error or warns if excessive errors are introduced. If large errors are introduced in the depletion module, the whole core analysis could be rendered useless. Even if the possibility for large errors is small, a method that can not warn the user of such errors will be unreliable.

Stability: A method is stable if small perturbations (for example roundoff errors) do not grow faster than what is inherent in the problem itself. Every numerical method has a region of stability and when a problem lies outside this region it will be unstable. When the depletion equations are stiff, a large region of stability will be necessary.

Accuracy: Errors can be introduced when infinite series are truncated or when numbers are rounded off due to the finite word length of a computer. Normally a tolerance is specified and it is required that these errors must be smaller than the given tolerance.

Efficiency: The efficiency of a method is the amount of computer time it requires to solve a particular problem. The computer time is measured in the amount of floating-point operations, or 'flops', executed by a method. A flop is defined as the computer time necessary to compute the FORTRAN statement:

```
A(I,J)=A(I,J)+T*A(I,K)
```
(1.4)

Most of the elements in the coefficient matrix are zero, making the matrix sparse. A method that operates only on the non-zero elements will be more efficient than a method that operates on all the elements.

Ease of use: The depletion module is part of a larger reactor core analysis system and is normally called from other modules. Because it is not directly called by the user, it will be the best if the minimum amount of user input is necessary for the method used in the depletion module. When a method requires input parameters, these parameters must preferably be automatically calculated as part of the algorithm used in the module.

These attributes will be mathematically defined in Chapter 5 where they are used.

1.7 Method of the investigation

In developing numerical algorithms and implementations that are optimized for solving the fuel depletion, the special properties of the system of depletion equations must be exploited. Firstly, in Chapter 2, the fuel depletion equations are derived. In the derivation the properties of the system are shown, like how the coefficient matrix depends on the neutron flux, that the matrix is sparse and essential nonnegative, and that the system of depletion equation can be stiff.

In Chapter 3 and 4 algorithms are given and explained for a number of different methods, including the methods mentioned in Section 1.5. Most of the algorithms are well known, although some small changes are made to some algorithms to optimise them for usage in the depletion module. Chapter 3 starts with a general overview of polynomial methods and then the Taylor expansion, Padé approximation, Krylov subspace approximation and Chebyshev approximation polynomial methods are discussed in more detail. Various algorithms that solve the depletion problem analytically are given in Chapter 4, including the GAUGE algorithm and Schur decomposition. In the second half of Chapter 4 algorithms are given that solve the depletion problem by using a solution of a similar depletion problem to precondition the system of depletion equations.

In Chapter 5, it is shown how the algorithms of Chapter 3 and 4 are implemented. By using a sample assembly calculation and sample global reactor calculation, the different implementations are compared against each other in order to find the best implementation for each method. The different methods are then compared against each other, using their best implementations, and against the criteria of Section 1.6. Chapter 6 then concludes the dissertation by stating which of these selected methods are the best to be used in the depletion module for solving the fuel depletion equations.

The goal of the dissertation, as stated earlier, is to find the best method to solve the depletion problem. The method of investigation that is followed to reach this goal can be summarised as follows:

1. Investigate the special properties of the system of fuel depletion equations (Chapter 2).
2. Give an overview of numerical methods and select a number of them (Chapter 3 and 4).
3. Using the special properties of the system, develop new or modify existing algorithms for each selected method where necessary (Chapters 3 and 4).
4. Implement each algorithm in one or more ways by using the special properties of the system (Chapter 5, Section 5.1).
5. Find the best implementation for each selected method by comparing the different implementations using a sample assembly and global reactor calculation (Chapter 5, Section 5.5).

6. Compare the selected methods, using their best implementation, against each other and against the GAUGE method using the criteria of Section 1.6 (Chapter 5, Section 5.6).

7. Conclude which single method will perform the best in the depletion module (Chapter 5, Section 5.7 and Chapter 6).

Chapter 2

Fuel Depletion Equations

To solve the fuel depletion equations optimally in the depletion module, the special properties and structure of the depletion equations must be exploited. First the definitions of neutron cross section and neutron flux are given Section 2.1, before the system of depletion equations is derived in Section 2.2. The various properties of the system of equations are discussed in Section 2.3 and it is shown how some of these properties can be exploited when implementing a numerical method.

2.1 Definitions

The *neutron flux* $\phi(\vec{r}, t, E)dE$ is the average number of neutrons per unit time that crosses a surface of unit area (of any orientation) at position $\vec{r}$ with an energy between E and $E + dE$ at time t,

$$\phi(\vec{r}, t, E)dE = \text{Number of incident neutrons/area/time.}$$

The *microscopic cross section* $\sigma_i(t, E)$ of an isotope i and for incident neutrons having an energy E can be formally defined as follows (Duderstadt & Hamilton, 1975:18)

$$\sigma_i(t, E) = \frac{\text{Number of reactions/nucleus/time}}{\text{Number of incident neutrons /cm}^2\text{/time}}.$$

In this sense, then, if the target has a total cross sectional area S, all of which is uniformly exposed to the incident beam, then

$$\frac{\sigma_i(t, E)}{S} = \begin{array}{l}\text{Probability per nucleus that a neutron in the} \\ \text{beam having an energy E will interact with it}\end{array}.$$

The *macroscopic cross section* $\Sigma_i(\vec{r}, t, E)$ can be interpreted as the probability per unit path length traveled that the neutron having an energy E will interact with a nucleus of the isotope i. The macroscopic cross section is related to the microscopic cross section by

$$\Sigma_i(\vec{r}, t, E) = N_i(\vec{r}, t)\sigma_i(t, E), \tag{2.1}$$

where $N_i(\vec{r},t)$ is the number density of isotope i, which is the number of atoms of the isotope in a unit volume, at time t. The reaction rate $R_i(\vec{r},t)$ is the number of neutron interactions per unit time in a unit volume:

$$\begin{aligned} R_i(\vec{r},t) &= \int_0^\infty \Sigma_i(\vec{r},t,E)\phi(\vec{r},t,E)dE \\ &= N_i(\vec{r},t)\overline{\sigma}_i(\vec{r},t)\phi(\vec{r},t), \end{aligned} \tag{2.2}$$

where

$$\phi(\vec{r},t) \equiv \int_0^\infty \phi(\vec{r},t,E)dE$$

is the flux of neutrons of all energies, and $\overline{\sigma}_i(\vec{r},t)$ is the flux weighted cross section, defined by

$$\overline{\sigma}_i(\vec{r},t) \equiv \frac{\int_0^\infty \sigma_i(E,t)\phi(\vec{r},t,E)dE}{\phi(\vec{r},t)}. \tag{2.3}$$

2.2 Derivation of the depletion equation

The depletion equation describes the rate of change of the number density $N_i(t)$ of isotope i:

$$\frac{dN_i(t)}{dt} = \text{gain rate - loss rate.} \tag{2.4}$$

In a reactor core there is a gain in the number of atoms of an isotope in a certain volume due to:

1. fission products produced by fission,
2. neutron capture of a parent isotope,
3. radio-active decay of a parent isotope,
4. movement of material from a neighboring volume into the volume.

For every gain term there is a corresponding loss term:

1. neutron capture that leads to fission,
2. neutron capture forming a daughter isotope,
3. radio-active decay into a daughter isotope,
4. outward movement of material into neighboring volume.

The next four subsections will show how each of these four types of gain and loss mechanisms give rise to terms in (2.4). It will be assumed that all the cross sections are flux weighted, using (2.3). The position dependence of all the variables will also not be shown.

2.2.1 Fission terms

A fission reaction is the reaction in which one fissile isotope fissions into two lighter fission products. For the fissile isotope i the loss rate is given by the fission reaction rate $R_{f,i}$:

$$-\left.\frac{dN_i(t)}{dt}\right|_{fission} = R_{f,i}(t) = \sigma_{f,i}(t)\phi(t)N_i(t) + \lambda_{f,i}N_i(t), \tag{2.5}$$

where the first term on the right hand side is fission due to the neutron capture of the fissile isotope having a microscopic fission cross section $\sigma_{f,i}$. The second term is fission due to spontaneous fission if the isotope has a decay constant of $\lambda_{f,i}$ where $\lambda_{f,i} = \ln 2/t_{1/2}$, with $t_{1/2}$ the half-life of the isotope for spontaneous fission.

The fission yield γ_{ji} gives the probability that a fissile material i gives rise to the fission product j. Because the fission yield for the neutron capture fission and spontaneous fission differ, the gain in the fission product j due to the fission of isotope i is

$$\left.\frac{dN_j(t)}{dt}\right|_{fission\,of\,i} = \gamma_{ji}\sigma_{f,i}(t)\phi(t)N_i(t) + \gamma'_{ji}\lambda_{f,i}N_i(t), \tag{2.6}$$

where γ_{ji} is the fission yield for neutron capture fission and γ'_{ji} the fission yield for spontaneous fission.

All the fissile isotopes are actinides which have a much higher atomic mass than the fission products. This makes it possible to divide all the isotopes into two groups, the actinides and the fission products. Other non-actinide isotopes that are present in the core are also included in the fission product group. Usually the n isotope number densities are written as a vector, $\vec{N}$, with the first n_a isotopes the actinides and the other $n - n_a$ isotopes the fission products. Using (2.5) for the loss rate of the actinides and (2.6) for the gain rate of the fission products, the equations can be written in matrix form as:

$$\left.\frac{d\vec{N}}{dt}\right|_{fission} = \left[\begin{array}{ccc|ccc} -\sigma_{f,1} & & 0 & 0 & \dots & 0 \\ & \ddots & & \vdots & \ddots & \vdots \\ 0 & & -\sigma_{f,n_a} & 0 & \dots & 0 \\ \hline \gamma_{n_a+1,1}\sigma_{f,1} & \dots & \gamma_{n_a+1,n_a}\sigma_{f,n_a} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \gamma_{n,1}\sigma_{f,1} & \dots & \gamma_{n,n_a}\sigma_{f,n_a} & 0 & \dots & 0 \end{array}\right] \phi\vec{N}$$

$$+ \left[\begin{array}{ccc|ccc} -\lambda_{1,f} & & 0 & 0 & \dots & 0 \\ & \ddots & & \vdots & \ddots & \vdots \\ 0 & & -\lambda_{n_a,f} & 0 & \dots & 0 \\ \hline \gamma'_{n_a+1,1}\lambda_{1,f} & \dots & \gamma'_{n_a+1,n_a}\lambda_{n_a,f} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \gamma'_{n,1}\lambda_{1,f} & \dots & \gamma'_{n,n_a}\lambda_{n_a,f} & 0 & \dots & 0 \end{array}\right] \vec{N} \tag{2.7}$$

$$= (\phi(t)\mathbf{F_1} + \mathbf{F_2})\vec{N}. \tag{2.8}$$

As shown in (2.8), the matrix can be divided into two matrices, one containing the neutron capture fission terms, $\mathbf{F}_1$, and the other one containing the spontaneous fission terms, $\mathbf{F}_2$. These two matrices can also be divided into four sub-matrices as shown in (2.7). Because these sub-matrices will later be further divided into sub-matrices, they will be called quadrants. The first quadrant is a n_a-by-n_a matrix containing the loss rates of the actinides and the reactions between the actinides. The fourth quadrant is a $(n-n_a)$-by-$(n-n_a)$ matrix containing the loss rates of the fission products and the reactions between the fission products. The second quadrant contains the reactions having a fission product as parent and an actinide as daughter and the third quadrant contains the reactions having an actinide as parent and a fission product as daughter.

For the fission reactions, the first quadrants are diagonal and contain all the loss terms of (2.5). The fission of a fissile isotope can produce many different isotopes, making the third quadrants of the matrices, that contains all the fission products, dense. Because a fission product can not be a fissile isotope or the parent of a fissile isotope, the second and fourth quadrants are zero.

2.2.2 Radio-active decay terms

Many of the fission products are unstable so that the decay reactions (α), (γ), (β^+), (β^-), (β^-,n), (β^-,α), (β^+,n), (β^+,α), (β^-,p) and (p) are all possible inside a nuclear reactor (Forrest, 1997:11). The most common radio-active decay types are:

Alpha decay

$$^A_Z X \rightarrow {}^{A-4}_{Z-2}X + {}^4_2\mathrm{He}, \tag{2.9}$$

Beta decay

$$^A_Z X \rightarrow {}^A_{Z+1}X + \beta^- \tag{2.10}$$

or

$$ {}^{A}_{Z}X \rightarrow {}^{A}_{Z-1}X + \beta^{+}, \tag{2.11} $$

Neutron decay

$$ {}^{A}_{Z}X \rightarrow {}^{A-1}_{Z}X + n, \tag{2.12} $$

Gamma decay

$$ {}^{A}_{Z}X^{*} \rightarrow {}^{A}_{Z}X + \gamma, \tag{2.13} $$

where X^* is an excited state of the isotope. The changes in the mass number A and the atomic number Z are shown in the above reactions.

Normally the isotopes in the isotope vector $\vec{N}$ can be ordered such that the parent isotope's index for all the decay reactions is higher than the daughter's. The decay terms in the depletion equations are then

$$ \left.\frac{dN_i(t)}{dt}\right|_{decay} = -\lambda_i N_i + \sum_{j<i} \lambda_{ij} N_j. \tag{2.14} $$

The summation on the right hand side is the gain of the daughter isotope i due to the decay of the parent isotope j with λ_{ij} the decay constant of the reaction. The first term is the loss rate of isotope i due to all the decay reactions of which the isotope is the parent, with λ_i the total decay constant

$$ \lambda_i = \sum_{k>i} \lambda_{ik} = \frac{\ln 2}{t_{1/2}}. \tag{2.15} $$

Because there is no radio-active decay between an actinide and fission product, the decay chains of the fission products and actinides are independent. When all the parent isotope indices are higher than their daughter's indices, the decay coefficient matrix will be triangular. Separating the actinides and fission products, like in (2.7), the second and the third quadrants of the decay coefficient matrix will be zero and the first and the fourth quadrants triangular:

$$ \left.\frac{d\vec{N}}{dt}\right|_{decay} = \left[\begin{array}{ccc|ccc} -\lambda_1 & & 0 & 0 & \dots & 0 \\ \vdots & \ddots & & \vdots & \ddots & \vdots \\ \lambda_{n_a,1} & \dots & -\lambda_{n_a} & 0 & \dots & 0 \\ \hline 0 & \dots & 0 & -\lambda_{n_a+1} & & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \\ 0 & \dots & 0 & \lambda_{n,n_a+1} & \dots & -\lambda_n \end{array}\right] \vec{N} = \mathbf{V}\vec{N}. \tag{2.16} $$

Each isotope has only a few radio-active decay terms, making the decay coefficient matrix $\mathbf{V}$ sparse. Two decay chains are said to be independent if there is no isotope in one chain that is the parent or daughter of an isotope in the other chain. When there are a number of independent decay chains, the two triangular quadrants in $\mathbf{V}$ can be divided into smaller triangular matrices on the diagonal, as will be shown in Subsection 2.3.2.

2.2.3 Neutron capture terms

In neutron capture reactions a parent isotope i captures a neutron forming a daughter isotope j. The only exception is fission, where two daughter isotopes are formed for each neutron capture, which was treated separately in Subsection 2.2.1. The loss rate of the parent isotope and gain rate of the daughter isotope are given by the neutron capture reaction rate

$$R_{c,i\to j}(t) = \sigma_{c,i\to j}(t)\phi(t)N_i(t), \tag{2.17}$$

where $\sigma_{c,i\to j}$ is the microscopic cross section for the neutron capture reaction. Except for fission, there is only a small change in mass number between the parent and daughter isotopes in all neutron capture reactions and radio-active decay reaction. Some examples of neutron capture reactions, showing the changes in mass number A and atomic number Z, are

$$\begin{aligned}
{}^{A}_{Z}X_i + n &\rightarrow {}^{A+1}_{Z}X_{j_1} + \gamma, && (2.18)\\
{}^{A}_{Z}X_i + n &\rightarrow {}^{A-3}_{Z-2}X_{j_2} + \alpha, && (2.19)\\
{}^{A}_{Z}X_i + n &\rightarrow {}^{A}_{Z-1}X_{j_3} + p, && (2.20)\\
{}^{A}_{Z}X_i + n &\rightarrow {}^{A-1}_{Z}X_{j_4} + 2n. && (2.21)
\end{aligned}$$

Normally the isotopes are almost arranged by their mass numbers. This will result in a capture coefficient matrix, when written in four quadrants as in (2.7), with all the neutron capture terms close to the diagonal,

$$\left.\frac{d\vec{N}}{dt}\right|_{capture} = \left[\begin{array}{ccc|ccc} -\sigma_{c,1} & \ddots & 0 & 0 & \dots & 0 \\ \ddots & \ddots & \ddots & \vdots & \ddots & \vdots \\ 0 & \ddots & -\sigma_{c,n_a} & 0 & \dots & 0 \\ \hline 0 & \cdots & 0 & -\sigma_{c,n_a+1} & \ddots & 0 \\ \vdots & \ddots & \vdots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & 0 & \ddots & -\sigma_{c,n} \end{array}\right] \phi\vec{N} = \mathbf{C}\phi\vec{N}. \tag{2.22}$$

Because it is possible that an isotope can be its own precursor, feedback effects occur and the coefficient matrix can not always be written as a triangular matrix.

Although a large number of different reactions are possible, most isotopes are only involved in one or two types of neutron capture reactions. This makes the capture coefficient matrix $\mathbf{C}$ sparse.

2.2.4 Movement of material

If there is no movement of material between different regions, the system of depletion equations for the different regions will be loosely coupled through the neutron flux and can be solved independently in the depletion module. When movement of material is present, all the depletion equations of the different regions will be coupled, forming one very large system of first order differential equations.

These equations can be decoupled by assuming a constant rate of material movement between the regions in each time step. Let the gain or loss rate due to movement in a region be constant over a time step ΔT

$$\left.\frac{d\vec{N}(t)}{dt}\right|_{loss} = \frac{\Delta\vec{N}}{\Delta T} \equiv \vec{\alpha}_{loss}. \tag{2.23}$$

This will add a constant term to the depletion equation (1.1),

$$\frac{d\vec{N}(t)}{dt} = \mathbf{A}\vec{N}(t) + \vec{\alpha}_{loss}. \tag{2.24}$$

By defining

$$\vec{N}'(t) \equiv \vec{N}(t) + \mathbf{A}^{-1}\vec{\alpha}_{loss}, \tag{2.25}$$

the system of depletion equations (2.24) with initial conditions (1.2) can be rewritten as the system of equations

$$\frac{d\vec{N}'}{dt}(t) = \mathbf{A}\vec{N}'(t), \tag{2.26}$$

with initial conditions

$$\vec{N}'(0) = \vec{N}_0 + \mathbf{A}^{-1}\vec{\alpha}_{loss}. \tag{2.27}$$

This system has the same form as (1.1) and (1.2). It is therefore possible to eliminate all the constant gain and loss terms in the fuel depletion equation, by rewriting the system of equations (Wagner, 1968:33; Gallopoulos & Saad, 1992:1238).

2.2.5 Coefficient matrix

Combining the fission, the radio-active decay and the neutron capture gain and loss terms given by (2.8), (2.16) and (2.22), the system of depletion equations can be written as

$$\frac{d\vec{N}(t)}{dt} = \left.\frac{d\vec{N}(t)}{dt}\right|_{fission} + \left.\frac{d\vec{N}(t)}{dt}\right|_{decay} + \left.\frac{d\vec{N}(t)}{dt}\right|_{capture} \tag{2.28}$$

$$= [\phi(t)(\mathbf{F}_1 + \mathbf{C}) + \mathbf{F}_2 + \mathbf{V}]\,\vec{N}(t) \tag{2.29}$$

$$= \mathbf{A}(t)\vec{N}(t), \tag{2.30}$$

where $\mathbf{A}$ is the coefficient matrix as in (1.1). The properties of this equation are described in the next section.

2.3 Properties of the coefficient matrix

From the derivation of the system of depletion equations in Section 2.2, some properties of the system of equations can be deduced. The properties that will be discussed in this section are

the dependence of the coefficient matrix on the neutron flux (Subsection 2.3.1), the structure and sparseness of the coefficient matrix (Subsection 2.3.2), and the stiffness of the system of equations (Subsection 2.3.4). The coefficient matrix is also an essential nonnegative matrix (Subsection 2.3.5), and certain upper and lower limits can be given for the matrix norm (Subsection 2.3.3).

2.3.1 Constant flux approximation

In the depletion module, the microscopic neutron cross sections are assumed to be constant over the entire time step for which the depletion equations have to be solved. From the definition of the flux weighted cross section (2.3) it can be seen that this assumption will be valid as long as the neutron energy spectrum stays the same. Using this assumption, the coefficient matrix $\mathbf{A}$, as was constructed in (2.29), can be written as

$$\mathbf{A}(t) = \mathbf{A}_0 + \mathbf{A}_1\phi(t), \tag{2.31}$$

where the matrices $\mathbf{A}_0$ and $\mathbf{A}_1$ are independent of time.

To solve the fuel depletion equations, the neutron flux $\phi(t)$ must be known. The neutron flux is calculated in the flux-power-reactivity module by solving the Boltzmann transport equation (Stamm'ler & Abbate, 1983:29). The Boltzmann equation contains the macroscopic cross sections (2.1) of all the isotopes, which depend on the isotopic number densities. The depletion and Boltzmann equations are therefore coupled and can not be solved separately. To solve the depletion equations, the neutron flux $\phi(t)$ is usually assumed to be constant over a burnup time step ΔT. This constant flux assumption then gives the coefficient matrix that is independent of time

$$\mathbf{A} = \mathbf{A}_0 + \mathbf{A}_1\phi(0),\ 0 < t < \Delta T. \tag{2.32}$$

When the constant flux approximation can not be made, a non-linear method will have to be used in the depletion module, which is discussed in Chapter 6.

When a reactor is operating, the flux is normally varied to keep the power that the reactor produces constant. The power per unit volume is given by

$$P = \phi(t)\sum_i \sigma_{f,i}N_i(t)w_i + \sum_i \lambda_{f,i}N_i(t)w_i', \tag{2.33}$$

which is the sum of the capture fission reaction rate (2.5) for all fissile isotopes i multiplied by the energy released per neutron capture fission, w_i, and the spontaneous fission rate for all fissile isotopes i multiplied by the energy released per spontaneous fission event, w_i'.

When the fissile isotope concentrations decrease with ΔN_i, from $N_i(t_0)$ at time t_0 to $N_i(t_1)$ at time t_1,

$$N_i(t_1) = N_i(t_0) - \Delta N_i, \tag{2.34}$$

the flux must increase from $\phi(t_0)$ to $\phi(t_1)$ for the power to stay constant. When spatial dependence

is neglected, the change in neutron flux is given by

$$\phi(t_1) - \phi(t_0) = \frac{\phi(t_0)\sum_i \sigma_{f,i}\Delta N_i w_i + \sum_i \lambda_{f,i}\Delta N_i w_i'}{\sum_i \sigma_{f,i} N_i(t_1) w_i}. \tag{2.35}$$

The change in the coefficient matrix will then be

$$\Delta\mathbf{A} = \mathbf{A}(t_1) - \mathbf{A}(t_0) = \mathbf{A}_1 \left[\phi(t_1) - \phi(t_0)\right]. \tag{2.36}$$

To solve the fuel depletion equations at a constant power, the burnup time step is divided into a number of small time steps. Over each of these small time steps the neutron flux is assumed to be constant and the system (1.1) is solved. After each time step the neutron flux and coefficient matrix are updated using (2.35) and (2.36) to keep the power constant.

Some methods for solving the fuel depletion equations can make use of a previous solution if the change in the coefficient matrix is not large. These methods can be used when a constant power approximation is made by calculating the first time step fully and then using only the change in the coefficient matrix when solving the subsequent time steps. Especially when the system contains fast decaying isotopes, the matrix $\mathbf{A}_0$ will have large coefficients, which are not contained in $\Delta\mathbf{A}$. This has the effect of reducing the stiffness of the problem.

2.3.2 Sparseness and structure

The three types of reactions forming the coefficient matrix in (2.28), can all be written as matrices that are divided into four quadrants as explained in Subsections 2.2.1, 2.2.2 and 2.2.3. When the matrices given by Eqs. (2.7), (2.16) and (2.22) are added, the four quadrants of the coefficient matrix have the form (Wagner, 1968:28)

$$\mathbf{A} = \left[\begin{array}{ccc|ccc} -d_1 & \ddots & & 0 & \dots & 0 \\ \ddots & \ddots & \ddots & \vdots & \ddots & \vdots \\ & \ddots & -d_{n_a} & 0 & \dots & 0 \\ \hline \lambda_{n_a+1,1}F_1 & \cdots & \lambda_{n_a+1,n_a}F_{n_a} & -d_{n_a+1} & \ddots & \\ \vdots & \ddots & \vdots & \ddots & \ddots & \ddots \\ \lambda_{n,1}F_1 & \cdots & \lambda_{n,n_a}F_{n_a} & & \ddots & -d_n \end{array}\right] = \left[\begin{array}{c|c} \mathbf{B} & \mathbf{0} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right], \tag{2.37}$$

with n_a and n the number of actinides and the total number of isotopes, respectively. The fission yield terms, contained in the third quadrant, matrix $\mathbf{C}$, were shown to be a dense matrix in Subsection 2.2.1. The matrix $\mathbf{C}$ is therefore stored as a $n_a \times (n - n_a)$ matrix. The first and fourth quadrants, matrices $\mathbf{B}$ and $\mathbf{D}$, are sparse, with elements on the diagonal, and the radio-active decay and neutron capture terms close to and mainly below the diagonal (see Subsections 2.2.2 and 2.2.3). To effectively store and use these two sparse matrices, a band storage scheme, index storing scheme or sub-matrix storing scheme can be used. These three schemes are now described and explained by applying it to the coefficient matrix

$$\mathbf{A} = \begin{bmatrix} -1.40\text{x}10^{-8} & 5.60\text{x}10^{-13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2.79\text{x}10^{-9} & -2.45\text{x}10^{-9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3.87\text{x}10^{-10} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -3.97\text{x}10^{-8} & 4.20\text{x}10^{-13} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.43\text{x}10^{-8} & -3.96\text{x}10^{-8} & 2.80\text{x}10^{-13} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.93\text{x}10^{-8} & -3.95\text{x}10^{-8} & 4.90\text{x}10^{-13} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9.47\text{x}10^{-9} & -1.14\text{x}10^{-8} & 0 & 0 \\ 7.11\text{x}10^{-10} & 6.95\text{x}10^{-12} & 2.76\text{x}10^{-12} & 1.58\text{x}10^{-9} & 1.76\text{x}10^{-11} & 2.01\text{x}10^{-9} & 1.27\text{x}10^{-11} & -2.91\text{x}10^{-5} & 0 \\ 2.97\text{x}10^{-11} & 1.61\text{x}10^{-13} & 7.06\text{x}10^{-15} & 2.63\text{x}10^{-10} & 1.35\text{x}10^{-12} & 5.92\text{x}10^{-11} & 1.62\text{x}10^{-13} & 2.91\text{x}10^{-5} & -6.72\text{x}10^{-5} \end{bmatrix} s^{-1} \quad (2.38)$$

as an example.

2.3.2.1 Band storage scheme

Because all the elements in the matrices **B** and **D** are around the diagonal, they can be stored in a band storage scheme (Golub & Van Loan, 1990:21). If the furthest elements from the diagonal are p and q positions above and below the diagonal, all the elements can be stored in a $(p+q+1)$-by-n matrix. In an assembly calculation, for example, with about 130 isotopes, p and q are around 4 or 5 and in a core calculation with about 10 isotopes, p and q are normally 1 or 2. The band matrix is therefore much smaller than the full coefficient matrix. For example, the coefficient matrix (2.38) originated out of a core calculation and has $p = q = 1$, giving the band matrix shown in Figure 2.1. The advantage of the band storage method is that no index look up has to be done, as will be explained in the next paragraph. On the other hand, there may still be many zero elements in the band matrix that are being stored and used in calculations. Especially if there exists a reaction where the index of the parent and daughter isotope differ much, p or q will have to be large, making the band storage scheme less effective.

2.3.2.2 Index storage scheme

In an assembly calculation only about 3% of the matrix elements in **B** and **D** are non-zero and in a core calculation, about 30% of these elements are non-zero. Storing and using only the non-zero matrix elements in **B** and **D** will result in the minimum amount of flops. This can be done by storing, for each reaction, the index of the parent and daughter isotope and the reaction rate, in three vectors. The disadvantage of this storage scheme is that for every calculation a look up is required in the vectors containing the parent and daughter isotope index to find the position of the reaction in the coefficient matrix. In small matrices these look ups can take longer than just doing zero calculations. This will be shown in Chapter 5, where the same methods using different storage schemes are compared with each other.

$$p = 1;\ q = 1$$

$$\text{Band Matrix} = \begin{bmatrix} & 5.60\text{x}10^{-13} & 0 & 0 & 4.20\text{x}10^{-13} & 2.80\text{x}10^{-13} & 4.90\text{x}10^{-13} & 0 & 0 \\ 1.40\text{x}10^{-8} & 2.45\text{x}10^{-9} & 3.87\text{x}10^{-10} & 3.97\text{x}10^{-8} & 3.96\text{x}10^{-8} & 3.95\text{x}10^{-8} & 1.14\text{x}10^{-8} & 2.91\text{x}10^{-5} & 6.72\text{x}10^{-5} \\ 2.79\text{x}10^{-9} & 0 & 0 & 1.43\text{x}10^{-8} & 3.93\text{x}10^{-8} & 9.47\text{x}10^{-9} & 0 & 2.91\text{x}10^{-5} & \end{bmatrix}$$

$$\text{Fission Yield} = \begin{bmatrix} 7.11\text{x}10^{-10} & 6.95\text{x}10^{-12} & 2.76\text{x}10^{-12} & 1.58\text{x}10^{-9} & 1.76\text{x}10^{-11} & 2.01\text{x}10^{-9} & 1.27\text{x}10^{-11} \\ 2.97\text{x}10^{-11} & 1.61\text{x}10^{-13} & 7.06\text{x}10^{-15} & 2.63\text{x}10^{-10} & 1.35\text{x}10^{-12} & 5.92\text{x}10^{-11} & 1.62\text{x}10^{-13} \end{bmatrix}$$

Figure 2.1: Band storage for the core sample problem

$$
\begin{aligned}
\text{Diagonal} &= [\ 1.40\text{x}10^{-8}\ \ 2.45\text{x}10^{-9}\ \ 3.87\text{x}10^{-10}\ \ 3.97\text{x}10^{-8}\ \ 3.96\text{x}10^{-8}\ \ 3.95\text{x}10^{-8}\ \ 1.14\text{x}10^{-8}\ \ 2.91\text{x}10^{-5}\ \ 6.72\text{x}10^{-5}\ ] \\
\text{Number of parents} &= [\ 1\ \ 1\ \ 0\ \ 1\ \ 2\ \ 2\ \ 1\ \ 7\ \ 8\ ] \\
\text{Parent Index} &= [\ 2\ \ 1\ \ 5\ \ 4\ \ 6\ \ 5\ \ 7\ \ 6\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ ] \\
\text{Decay constants} &= \begin{bmatrix} 5.60\text{x}10^{-13} & 2.79\text{x}10^{-9} & 4.20\text{x}10^{-13} & 1.43\text{x}10^{-8} & 2.80\text{x}10^{-13} & 3.93\text{x}10^{-8} & 4.90\text{x}10^{-13} & 9.47\text{x}10^{-9} \\ 7.11\text{x}10^{-10} & 6.95\text{x}10^{-12} & 2.76\text{x}10^{-12} & 1.58\text{x}10^{-9} & 1.76\text{x}10^{-11} & 2.01\text{x}10^{-9} & 1.27\text{x}10^{-11} & 2.97\text{x}10^{-11} \\ 1.61\text{x}10^{-13} & 7.06\text{x}10^{-15} & 2.63\text{x}10^{-10} & 1.35\text{x}10^{-12} & 5.92\text{x}10^{-11} & 1.62\text{x}10^{-13} & 2.91\text{x}10^{-5} & \end{bmatrix}
\end{aligned}
$$

Figure 2.2: Index storage of parents in a vector for the core sample problem (2.38)

$$
\begin{aligned}
\text{Diagonal} &= [\ 1.40\text{x}10^{-8}\ \ 2.45\text{x}10^{-9}\ \ 3.87\text{x}10^{-10}\ \ 3.97\text{x}10^{-8}\ \ 3.96\text{x}10^{-8}\ \ 3.95\text{x}10^{-8}\ \ 1.14\text{x}10^{-8}\ \ 2.91\text{x}10^{-5}\ \ 6.72\text{x}10^{-5}\ ] \\
\text{Number of parents} &= [\ 1\ \ 1\ \ 0\ \ 1\ \ 2\ \ 2\ \ 1\ \ 0\ \ 1\ ] \\
\text{Parent Index} &= [\ 2\ \ 1\ \ 5\ \ 4\ \ 6\ \ 5\ \ 7\ \ 6\ \ 8\ ] \\
\text{Decay constants} &= [\ 5.60\text{x}10^{-13}\ \ 2.79\text{x}10^{-9}\ \ 4.20\text{x}10^{-13}\ \ 1.43\text{x}10^{-8}\ \ 2.80\text{x}10^{-13}\ \ 3.93\text{x}10^{-8}\ \ 4.90\text{x}10^{-13}\ \ 9.47\text{x}10^{-9}\ \ 2.91\text{x}10^{-5}\ ] \\
\text{Fission Yield} &= \begin{bmatrix} 7.11\text{x}10^{-10} & 6.95\text{x}10^{-12} & 2.76\text{x}10^{-12} & 1.58\text{x}10^{-9} & 1.76\text{x}10^{-11} & 2.01\text{x}10^{-9} & 1.27\text{x}10^{-11} \\ 2.97\text{x}10^{-11} & 1.61\text{x}10^{-13} & 7.06\text{x}10^{-15} & 2.63\text{x}10^{-10} & 1.35\text{x}10^{-12} & 5.92\text{x}10^{-11} & 1.62\text{x}10^{-13} \end{bmatrix}
\end{aligned}
$$

Figure 2.3: Index storage of parents in a vector and a fission yield matrix for the core sample problem (2.38)

$$
\begin{aligned}
\text{Diagonal} &= [\ 1.40\text{x}10^{-8}\ \ 2.45\text{x}10^{-9}\ \ 3.87\text{x}10^{-10}\ \ 3.97\text{x}10^{-8}\ \ 3.96\text{x}10^{-8}\ \ 3.95\text{x}10^{-8}\ \ 1.14\text{x}10^{-8}\ \ 2.91\text{x}10^{-5}\ \ 6.72\text{x}10^{-5}\ ] \\
\text{Number of parents} &= [\ 1\ \ 1\ \ 0\ \ 1\ \ 2\ \ 2\ \ 1\ \ 1\ \ 0\ ] \\
\text{Daugther Index} &= [\ 2\ \ 1\ \ 5\ \ 4\ \ 6\ \ 5\ \ 7\ \ 6\ \ 9\ ] \\
\text{Decay constants} &= [2.79\text{x}10^{-9}\ \ 5.60\text{x}10^{-13}\ \ 1.43\text{x}10^{-8}\ \ 4.20\text{x}10^{-13}\ \ 3.93\text{x}10^{-8}\ \ 2.80\text{x}10^{-13}\ \ 9.47\text{x}10^{-9}\ \ 4.90\text{x}10^{-13}\ \ 2.91\text{x}10^{-5}\ ] \\
\text{Fission Yield} &= \begin{bmatrix} 7.11\text{x}10^{-10} & 6.95\text{x}10^{-12} & 2.76\text{x}10^{-12} & 1.58\text{x}10^{-9} & 1.76\text{x}10^{-11} & 2.01\text{x}10^{-9} & 1.27\text{x}10^{-11} \\ 2.97\text{x}10^{-11} & 1.61\text{x}10^{-13} & 7.06\text{x}10^{-15} & 2.63\text{x}10^{-10} & 1.35\text{x}10^{-12} & 5.92\text{x}10^{-11} & 1.62\text{x}10^{-13} \end{bmatrix}
\end{aligned}
$$

Figure 2.4: Index storage of daughters in a vector and a fission yield matrix for the core sample problem (2.38)

$$
\begin{aligned}
\text{Number of actinide blocks} &= 3 \\
\text{Number of FP blocks} &= 1 \\
\text{Block size} &= [\ 2\ \ 1\ \ 4\ \ 2\ ] \\
\text{Fission Yield} &= \begin{bmatrix} 7.11\text{x}10^{-10} & 6.95\text{x}10^{-12} & 2.76\text{x}10^{-12} & 1.58\text{x}10^{-9} & 1.76\text{x}10^{-11} & 2.01\text{x}10^{-9} & 1.27\text{x}10^{-11} \\ 2.97\text{x}10^{-11} & 1.61\text{x}10^{-13} & 7.06\text{x}10^{-15} & 2.63\text{x}10^{-10} & 1.35\text{x}10^{-12} & 5.92\text{x}10^{-11} & 1.62\text{x}10^{-13} \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\text{Blocks} &= \left\{ \begin{bmatrix} -1.40\text{x}10^{-8} & 5.60\text{x}10^{-13} \\ 2.79\text{x}10^{-9} & -2.45\text{x}10^{-9} \end{bmatrix}; -3.87\text{x}10^{-10}; \begin{bmatrix} -3.97\text{x}10^{-8} & 4.20\text{x}10^{-13} & 0 & 0 \\ 1.43\text{x}10^{-8} & -3.96\text{x}10^{-8} & 2.80\text{x}10^{-13} & 0 \\ 0 & 3.93\text{x}10^{-8} & -3.95\text{x}10^{-8} & 4.90\text{x}10^{-13} \\ 0 & 0 & 9.47\text{x}10^{-9} & -1.14\text{x}10^{-8} \end{bmatrix}; \begin{bmatrix} -2.91\text{x}10^{-5} & 0 \\ 2.91\text{x}10^{-5} & -6.72\text{x}10^{-5} \end{bmatrix} \right\} \\
&= \begin{bmatrix} -1.40\text{x}10^{-8} & 5.60\text{x}10^{-13} & 2.79\text{x}10^{-9} & -2.45\text{x}10^{-9} & 3.87\text{x}10^{-10} \\ -3.97\text{x}10^{-8} & 4.20\text{x}10^{-13} & 0 & 0 & 1.43\text{x}10^{-8} \\ -3.96\text{x}10^{-8} & 2.80\text{x}10^{-13} & 0 & 0 & 3.93\text{x}10^{-8} \\ -3.95\text{x}10^{-8} & 4.90\text{x}10^{-13} & 0 & 0 & 9.47\text{x}10^{-9} \\ -1.14\text{x}10^{-8} & -2.91\text{x}10^{-5} & 0 & 2.91\text{x}10^{-5} & -6.72\text{x}10^{-5} \end{bmatrix}
\end{aligned}
$$

Figure 2.5: Block storage scheme for the core sample problem (2.38)

In the GAUGE code (Wagner, 1968:28) it is assumed that every isotope can have only one daughter due to radio-active decay and one due to neutron capture. Five vectors are used to store the decay constants, the fission cross sections, the neutron capture cross sections and the index of the decay and neutron capture daughter isotopes for each parent isotope respectively. The i-th element in these vectors contains the reactions having the i-th isotope as parent and may contains zeros if there are no reactions having this isotope as parent. This implementation does not require a vector containing the parent isotope's index and therefore one less look up is necessary for each calculation, but it places a severe restriction on the possible decay chains that can be described.

Another way to store the coefficient matrix is to store the parent or daughter isotopes for each isotope as shown in Figure 2.2. One vector is used to store the diagonal elements for each isotope. In the parent index storage scheme, a second vector is used to store the number of parent isotopes for each isotope. The index of these parent isotopes and the reaction rates are stored in another two vectors. The fission yield terms can also be stored separately as shown in Figures 2.3 and 2.4.

2.3.2.3 Sub-matrix decomposition

Another way to reduce the amount of zero calculations is to divide the elements around the diagonal into a number of square sub-matrices $\mathbf{D}_i$ (Wagner, 1968:29) on the diagonal

$$\mathbf{A} = \left[\begin{array}{ccc|ccc} \mathbf{D}_1 & & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ & \ddots & & \vdots & \ddots & \vdots \\ \mathbf{0} & & \mathbf{D}_d & \mathbf{0} & \cdots & \mathbf{0} \\ \hline \mathbf{F}_{d+1,1} & \cdots & \mathbf{F}_{d+1,d} & \mathbf{D}_{d+1} & & \mathbf{0} \\ \vdots & \ddots & \vdots & & \ddots & \\ \mathbf{F}_{n,1} & \cdots & \mathbf{F}_{n,d} & \mathbf{0} & & \mathbf{D}_n \end{array}\right]. \tag{2.39}$$

Each matrix $\mathbf{D}_i$ represents an independent decay chain. If there are no feedback reactions in a decay chain, the corresponding sub-matrix $\mathbf{D}_i$ will be triangular. In this scheme, the sub-matrices can be stored in one vector as shown in Figure 2.5, together with the size of each sub-matrix. This requires much fewer lookups than the index storage scheme, but more zero computations need to be done.

The advantage of this scheme, when compared to the band storage scheme, is that the block structure stays the same in matrix-matrix multiplication and matrix inversion. Also, if a sub-matrix $\mathbf{D}_i$ is triangular, it will stay triangular. This scheme can therefore be used when matrix-matrix calculations are done without a change of structure.

2.3.3 Norm

The i-th column in the coefficient matrix gives the loss rate of the i-th isotope at the diagonal element and the gain rate of its daughter isotopes at the off-diagonal elements. In fission reactions each individual fissile isotope gives rise to two fission products. In all the other reactions each

individual parent isotope gives rise to only one daughter isotope. For the fission product isotopes, the sum of all the elements in a column, except the diagonal element, is therefore always smaller than or equal to the absolute value of the diagonal element,

$$\sum_{i \neq j} a_{ij} \leq |a_{jj}| \text{ for all fission products } j. \tag{2.40}$$

For the fissile isotopes, the sum is always smaller than or equal to double the absolute value of the diagonal element,

$$\sum_{i \neq j} a_{ij} \leq 2\,|a_{jj}| \text{ for all fissile isotopes } j. \tag{2.41}$$

These properties can be used to calculate lower and upper limits for the norm of the coefficient matrix. The 1, 2 and ∞ matrix p-norms are defined as (Golub & Van Loan, 1990:56):

$$\|\mathbf{A}\|_1 \equiv \max_j \sum_i |a_{ij}|\,, \tag{2.42}$$

$$\|\mathbf{A}\|_2 \equiv \max_{\|x\|=1} \|\mathbf{A}x\|\,, \tag{2.43}$$

$$\|\mathbf{A}\|_\infty \equiv \max_i \sum_j |a_{ij}|\,. \tag{2.44}$$

The 2-norm has the property that (Golub & Van Loan, 1990:56)

$$\max_{i,j} |a_{ij}| \leq \|\mathbf{A}\|_2 \leq n \max_{i,j} |a_{ij}|\,. \tag{2.45}$$

Let $D_{max} = \max_i |a_{ii}|$. By using (2.41) and (2.40), the matrix norms of the coefficient matrix must satisfy

$$D_{max} \leq \|\mathbf{A}\|_1 \leq 3D_{max} \tag{2.46}$$

$$D_{max} \leq \|\mathbf{A}\|_2 \leq nD_{max} \tag{2.47}$$

$$D_{max} \leq \|\mathbf{A}\|_\infty \leq nD_{max}. \tag{2.48}$$

It is clear that the largest diagonal element of the coefficient matrix, D_{max}, determines the norm of the matrix.

Another norm that is sometimes used in the analysis of numerical methods is the Lapidus and Luus norm (Hermann & Westfall, 1998:F7.2.3):

$$[\mathbf{A}] \equiv \min \left\{ \max_i \sum_j |a_{ij}|\,, \max_j \sum_i |a_{ij}| \right\}. \tag{2.49}$$

For the coefficient matrix this norm has an upper limit

$$[\mathbf{A}] \leq 3D_{max}. \tag{2.50}$$

The largest diagonal element D_{max} in the coefficient matrix is given by the largest decay constant or the product of the largest neutron capture cross section and the neutron flux

$$D_{max} = \max\left\{\frac{\ln 2}{\min_i(t_{1/2,i})}, \max_i(\sigma_{c,i})\phi\right\}. \tag{2.51}$$

2.3.4 Stiffness

The following definition of stiffness is given by Shampine (1985):

> Consider the numerical solution of the initial value problem for a system of n first-order ordinary differential equations:
>
> $$\vec{y'} = \vec{f}(x, \vec{y}) \qquad a \leq x \leq b, \tag{2.52}$$
>
> with $\vec{y}(a)$ given. The standard assumption about $\vec{f}(x, \vec{y})$ is that it has many continuous derivatives and that it satisfies the Lipschitz condition:
>
> $$\left\|\vec{f}(x, \vec{u}) - \vec{f}(x, \vec{v})\right\| \leq L\left\|\vec{u} - \vec{v}\right\| \qquad \text{for all } a \leq x \leq b \text{ and all } \vec{u} \text{ and } \vec{v}, \tag{2.53}$$
>
> where L is a real constant independent of $\vec{u}$ and $\vec{v}$.
>
> A problem is termed stiff if $L(b-a)$ is large.

For the depletion equation (1.1) the condition (2.53) can be written as

$$\left\|\mathbf{A}\vec{N}\right\| \leq L\left\|\vec{N}\right\| \qquad \text{for all } \vec{N}.$$

Using Eq. (2.43), the Lipschitz constant L for the coefficient matrix can be given by

$$L = \|\mathbf{A}\|_2 .$$

A system of depletion equations is therefore stiff if $\|\mathbf{A}\|_2 \, \Delta t$ is large, with Δt being the burnup time step. It is immediately clear that the stiffness is directly proportional to the length of the time step. Because the 2-norm is always bigger than the maximum diagonal element D_{max}, according to (2.47), a lower limit for the stiffness can be given by the half-life of the element with the smallest half-life:

$$\text{(Stiffness)} = \|\mathbf{A}\|_2 \, \Delta t \geq D_{max}\Delta t \geq \frac{\Delta t \ln 2}{\min(t_{1/2})}. \tag{2.54}$$

When isotopes with a half-life of few order of magnitudes smaller than the burnup time step are present, the problem is very stiff. Stiffness has an influence on the stability and roundoff errors of numerical methods, as will be shown in Chapter 3.

2.3.5 Essential nonnegativity

The coefficient matrix has the property that all the diagonal elements are loss terms and therefore negative, and all the off-diagonal terms are gain terms and therefore positive. A matrix with nonnegative off-diagonal elements is called essential nonnegative. This property makes it possible to use some special numerical techniques, like the uniformization technique of Section 3.2.

In the next three chapters it will be shown how various numerical methods to solve the fuel depletion equations can be optimally implemented using the properties given in this section.

Chapter 3

Polynomial Expansion Methods

There is a wide range of known methods to solve a system of first order differential equations like (1.1). Moler and Van Loan (1978) give a review of methods that can be used and conclude that the Taylor and Padé methods are the most effective methods when properly implemented. In this chapter the family of methods that makes use of polynomial expansion, which includes these two methods, is investigated. In the next chapter two other methods, that are not polynomial expansion methods, are considered. The performance of the different methods which are investigated in these two chapters is then compared with each other in Chapter 5.

Section 3.1 first gives an overview of polynomial methods. This includes the methods given by Moler and Van Loan (1978), interpolation methods as introduced by Iserles (1981) and the newer Krylov subspace approximation (Gallopoulos & Saad, 1992). Four of these methods, the Taylor, Padé, Krylov and Chebyshev approximations, are then further investigated in this chapter. Of these four methods, only the Taylor method is currently used in depletion modules, for example in ORIGEN-S. In Section 3.2 a new, more efficient, implementation of the Taylor method is developed, using the uniformization technique suggested by Sidje and Stewart (1999). In Sections 3.3, 3.4 and 3.5 it is shown how the general Padé, Krylov and Chebyshev approximation methods, as implemented by Arioli *et. al.* (1996) and Sidje (1998), can be optimally implemented to solve the fuel depletion equations.

3.1 Overview of polynomial expansion methods

In the depletion module, the fuel depletion equation (1.1) has to be solved over a burnup time interval Δt, starting with an initial isotope concentration $\vec{N}_0 = \vec{N}(t_0)$ to give the final isotope concentration $\vec{N}_1 = \vec{N}(t_1)$, $t_1 = t_0 + \Delta t$. The coefficient matrix $\mathbf{A}$ is assumed to be constant over the burnup time interval, as explained in Subsection 2.3.1.

The simplest way to compute the final isotope concentration is to use a Taylor expansion of

$\vec{N}_1 = \vec{N}(t_0 + \Delta t)$ around t_0:

$$\vec{N}_1 = \vec{N}_0 + \frac{\Delta t}{1!}\frac{d\vec{N}(t)}{dt}\bigg|_{t=t_0} + \frac{(\Delta t)^2}{2!}\frac{d^2\vec{N}(t)}{dt^2}\bigg|_{t=t_0} + \ldots + \frac{(\Delta t)^i}{i!}\frac{d^{(i)}\vec{N}(t)}{dt^{(i)}}\bigg|_{t=t_0} + \ldots . \tag{3.1}$$

By calculating the time derivatives of (1.1), the i-th derivative of $\vec{N}(t)$ is found to be

$$\frac{d^{(i)}\vec{N}(t)}{dt^{(i)}} = \mathbf{A}^i\vec{N}(t). \tag{3.2}$$

This can be used in the Taylor expansion (3.1) to give

$$\vec{N}_1 = (\mathbf{I} + \frac{\Delta t}{1!}\mathbf{A} + \frac{(\Delta t)^2}{2!}\mathbf{A}^2 + \ldots + \frac{(\Delta t)^i}{i!}\mathbf{A}^i + \ldots)\vec{N}_0. \tag{3.3}$$

Alternatively, the Taylor expansion can be done for $\vec{N}_0 = \vec{N}(t_1 - \Delta t)$ around t_1, giving

$$(\mathbf{I} - \frac{\Delta t}{1!}\mathbf{A} + \frac{(\Delta t)^2}{2!}\mathbf{A}^2 - \ldots + \frac{(-\Delta t)^i}{i!}\mathbf{A}^i + \ldots)\vec{N}_1 = \vec{N}_0, \tag{3.4}$$

which can then be solved to compute $\vec{N}_1$.

Polynomial expansion methods to compute $\vec{N}_1$ are methods that can be written as

$$(c_0\mathbf{I} + c_1\Delta t\mathbf{A} + c_2\Delta t^2\mathbf{A}^2 + \ldots)\vec{N}_1 = (d_0\mathbf{I} + d_1\mathbf{A}\Delta t + d_2\Delta t^2\mathbf{A}^2 + \ldots)\vec{N}_0. \tag{3.5}$$

This includes the Taylor methods (3.3) and (3.4). This equation includes all the methods to calculate $\vec{N}_1$ that use the basic matrix operations on $\mathbf{A}$. In practice, the two summations must be truncated after a number of terms, say p and q terms,

$$\begin{aligned}(c_0\mathbf{I} + c_1\Delta t\mathbf{A} + c_2\Delta t^2\mathbf{A}^2 + \ldots + c_q\Delta t^q\mathbf{A}^q)\vec{N}_1 \quad &= \\ (d_0\mathbf{I} + d_1\Delta t\mathbf{A} + d_2\Delta t^2\mathbf{A}^2 + \ldots d_p\Delta t^p\mathbf{A}^p)\vec{N}_0,\end{aligned} \tag{3.6}$$

giving the two matrix polynomials $\mathbf{D}_p$ and $\mathbf{C}_q$:

$$\mathbf{C}_q(\Delta t\mathbf{A})\vec{N}_1 \quad = \quad \mathbf{D}_p(\Delta t\mathbf{A})\vec{N}_0. \tag{3.7}$$

Any method that has only the one term $c_0\vec{N}_1$ on the left hand side, like Eq. (3.3) with $c_0 = 1$, is called explicit. Explicit methods can be implemented using a number of matrix-vector multiplications to calculate $d_i\mathbf{A}^i\vec{N}_0$ and the computer time needed is of order $\mathcal{O}(n^2)$. Methods that have more terms on the left hand side, like Eq. (3.4), are called implicit and must be solved in two phases. First the two polynomials given in (3.6) have to be calculated and then the linear system (3.7) has to be solved (Gallopoulos & Saad, 1992:1237). This requires matrix-matrix computations, which requires computer time of order $\mathcal{O}(n^3)$.

There are a number of ways to chose the coefficients $c_0, \ldots, c_q$ and $d_0, \ldots, d_p$ so that $\vec{N}_1$ approximates the solution of the depletion equation. Some of the most common ways are by using a Taylor expansion, Padé approximation, Padé interpolation, Krylov approximation, Chebyshev rational approximation, or by computing the coefficients analytically using the Cayley-Hamilton theorem.

Each of these methods will now be introduced and some will then be described in more detail in the remainder of this chapter.

Taylor expansion: The truncation in Eq. (3.3) after k terms gives the k-th order Taylor expansion method (Moler & Van Loan, 1978:807) with the coefficients of (3.6) defined as

$$\begin{cases} d_i &= \frac{1}{i!} \quad \text{for } i = 0, 1, \ldots, k \\ c_0 &= 1 \\ c_i &= 0 \quad \text{for } i > 0 \end{cases} . \tag{3.8}$$

Padé approximation: The Padé approximation method is a generalisation of the Taylor expansion method, with the $(k, 0)$ Padé approximation being the above Taylor expansion, and the $(0, k)$ Padé approximation being the Taylor expansion using (3.4). The polynomial coefficients of the (p, q) Padé approximation are chosen to match the Taylor expansion up to order $p + q$ (Sidje, 1998:305; Gallopoulos & Saad, 1992:1243) for scalars,

$$\sum_{k=0}^{\infty} \frac{x^k}{k!} = \frac{D_p(x)}{C_q(x)} + \mathcal{O}(x^{p+q+1}). \tag{3.9}$$

The resulting coefficients are then (Moler & Van Loan, 1978:808)

$$\begin{cases} d_{i,pq} &= \frac{(p+q-i)!p!}{(p+q)!i!(p-i)!} \quad \text{for } i = 0, 1, \ldots, p \\ c_{i,pq} &= (-1)^i \frac{(p+q-i)!q!}{(p+q)!i!(q-i)!} \quad \text{for } i = 0, 1, \ldots, q \end{cases} . \tag{3.10}$$

When $p = q$ then $c_i = (-1)^i d_i$ and it is called the diagonal Padé approximation method.

Krylov subspace approximation: The Krylov subspace approximation method is an explicit method. This approximation reformulates the problem of finding the best coefficients d_i in (3.6) (with $c_0 = 1$ and $c_i = 0$, $i > 0$) to calculate $\vec{N}_1$, using only $p \ll n$ terms,

$$\vec{N}_1 \approx d_0\vec{N}_0 + d_1\mathbf{A}\vec{N}_0 + d_2\mathbf{A}^2\vec{N}_0 + \cdots + d_p\mathbf{A}^{p-1}\vec{N}_0, \tag{3.11}$$

to finding the element in the subspace

$$K_p = \text{span}\left\{\vec{N}_0,\ \mathbf{A}\vec{N}_0,\ \mathbf{A}^2\vec{N}_0,\ \cdots,\ \mathbf{A}^{p-1}\vec{N}_0\right\} \tag{3.12}$$

that best approximates $\vec{N}_1$ (Saad, 1992:210). This projects the large, sparse, n-by-n, coefficient matrix $\mathbf{A}$ onto a smaller, dense p-by-p matrix. The matrix exponential of this smaller matrix is then obtained with another method like the Padé approximation method or Chebyshev rational approximation method, to give an element in K_p that best approximates the matrix exponential.

Padé interpolation. The common feature of the Taylor, Padé and Krylov approximation methods is that they fit the matrix exponential (1.3) only at the origin. These three methods are therefore very accurate near the origin, but may be inaccurate far away from it (Gallopoulos & Saad, 1992:1242). Iserles (1981:1) gives interpolation methods that interpolate the exponential

along an equispaced mesh instead of using a high-degree interpolation at the origin. The polynomial coefficients of (3.6) are therefore chosen such that the exponential is calculated accurately at the mesh points. Although he found that the Padé interpolation is more accurate than the Padé approximation for stiff systems, it was found that the Padé interpolation was unstable due to oscillation of the interpolation error between the mesh points. This restricted the mesh size to small values (Iserles, 1981:9), giving a method that behaves much the same as the Padé approximation. This method will therefore not be considered further.

Chebyshev rational approximation. In the (p,q)-Chebyshev rational approximation method, the coefficients $d_0,\ldots,d_p$ and $c_i,\ldots,c_q$ are chosen such that

$$\max_{0\leq x<\infty}\left|\frac{d_0+d_1x+d_2x^2+\ldots+d_px^p}{c_0+c_1x+c_2x^2+\ldots+c_qx^q}-e^{-x}\right| \tag{3.13}$$

is a minimum (Sidje, 1998:133). Instead of fitting the exponential at the origin or at a number of fixed points, this approximation uses the best fit to e^{-x} over the interval $0\leq x<\infty$ (Moler & Van Loan, 1978:811; Gallopoulos & Saad, 1992:1242). It is also possible to fit the exponential over the interval (λ_1,λ_n) where λ_1 and λ_n are the largest and smallest eigenvalue of the coefficient matrix. Bergamaschi and Vianello (2000) show how the largest and smallest eigenvalue can be approximated and the coefficients d_i be calculated for a $(p,0)$ approximation. In this dissertation only the diagonal (p,p) Chebyshev approximations will be considered over the interval $0\leq x<\infty$, for they are more stable than the $(p,0)$ approximations.

Cayley-Hamilton theorem. The characteristic polynomial of an n-by-n matrix $\mathbf{A}$ is defined (Moler & Van Loan, 1978:815) as

$$z^n-\sum_{i=0}^{n-1}c_iz^i=c(z)\equiv\det(z\mathbf{I}-\mathbf{A}). \tag{3.14}$$

The Cayley-Hamilton theorem states that $c(\mathbf{A})=0$ and hence that

$$\mathbf{A}^n=\sum_{i=0}^{n-1}c_i\mathbf{A}^i. \tag{3.15}$$

Any power of $\mathbf{A}$ can therefore be expressed in terms of $\mathbf{I},\mathbf{A},\ldots,\mathbf{A}^{n-1}$

$$\mathbf{A}^k=\sum_{i=0}^{n-1}\beta_{ki}\mathbf{A}^i. \tag{3.16}$$

The infinite Taylor expansion (3.3) can then be written as

$$e^{\mathbf{A}}=D_{n-1}(\mathbf{A})=\sum_{j=0}^{n-1}\left[\sum_{k=0}^{\infty}\frac{\beta_{kj}}{k!}\right]\mathbf{A}^i. \tag{3.17}$$

Therefore, there exist coefficients d_i, $i=0,\ldots,n-1$ that give the matrix exponential exactly (Moler & Van Loan, 1978:815). There is a whole range of methods that make use of this fact, like Lagrange interpolation, Newton interpolation and inverse Laplace transform (Moler &

Van Loan, 1978:816). In these methods the characteristic polynomial, $c(z)$, must be known. A disadvantage that these methods share with the analytical methods of the next chapter, is that although they give $e^{\mathbf{A}} = D_n(\mathbf{A})$ exactly, it is possible that $D_n(\mathbf{A}+\mathbf{E})$, where $\mathbf{E}$ is a small perturbation, is very different from $e^{\mathbf{A}+\mathbf{E}}$. This makes these methods unstable in the presence of roundoff errors (Moler & Van Loan, 1978:816). Although these methods are equivalent to the analytical methods considered in the next chapter, they are much more expensive, $O(n^4)$, to implement and are, therefore, not further considered in this dissertation.

In the remainder of this chapter, algorithms are given to show how the Taylor, Padé, Krylov and Chebyshev approximation methods can be optimally implemented to solve the fuel depletion equation.

3.2 Taylor expansion method

The Taylor method is used in the well known depletion code ORIGEN-S (Hermann & Westfall, 1998). In Subsection 3.2.1 it is shown how the Taylor expansion is recursively implemented in this code. In Subsection 3.2.2 it is shown how roundoff errors may influence the Taylor expansion and the limit that is, therefore, imposed by the ORIGEN-S code. In Subsection 3.2.3 the number of terms that are necessary in the expansion is considered. This is then used in Subsection 3.2.4 to develop a new algorithm using the optimal integration step size.

Instead of using integration time steps as in the ORIGEN-S code, the Taylor method can be implemented using the uniformization technique of Sidje and Stewart (1999:351). In Subsection 3.2.5 it is shown how this technique can be generalized and then optimized, giving a new algorithm for the Taylor approximation method.

In Subsection 3.2.6 it is shown how the average isotope concentration can also be calculated by a slight modification to the Taylor algorithms, using almost no extra computations.

3.2.1 Recursive implementation

It is well known that the k-th order Taylor expansion can be calculated recursively (Hermann & Westfall, 1998:F7.2.3) by constructing the vectors $\vec{C}_i$:

$$\begin{aligned} \vec{C}_0 &= \vec{N}_0 \\ \vec{C}_i &= \frac{\mathbf{A}\Delta t}{i}\vec{C}_{i-1} \quad \text{for } i = 1 \ldots k. \end{aligned} \tag{3.18}$$

Then

$$\vec{N}_1 = \sum_{i=0}^{k} \vec{C}_i. \tag{3.19}$$

This can be seen by rewriting (3.3) as

$$\vec{N}_1 = \vec{N}_0 + \mathbf{A}\Delta t\left[\vec{N}_0 + \frac{\mathbf{A}\Delta t}{2}\left\{\vec{N}_0 + \frac{\mathbf{A}\Delta t}{3}(\vec{N}_0 + \frac{\mathbf{A}\Delta t}{4}(\vec{N}_0 + ...))\right\}\right] \tag{3.20}$$

Just one matrix-vector multiplication is necessary to calculate each $\vec{C}_i$, and no matrix-matrix operations are required. When the matrix $\mathbf{A}$ is sparse, the matrix-vector multiplication can be done efficiently by using an index storage scheme. If every depletion equation has on average m terms, then the coefficient matrix, consisting of n isotope depletion equations, will have mn non-zero elements. One matrix-vector multiplication will then need mn lookups and mn flops and the whole recursive algorithm of a k-th order Taylor expansion will need nmk flops.

3.2.2 Roundoff errors

When the system of equations is stiff, large roundoff errors may be introduced. To illustrate this, consider a single isotope with a decay constant R, described by

$$\frac{dN(t)}{dt} = -RN(t). \tag{3.21}$$

The Taylor expansion then consists of alternating negative and positive terms

$$N(\Delta t) = N(0)(1 - R\Delta t + \frac{(R\Delta t)^2}{2!} - \frac{(R\Delta t)^3}{3!} + \frac{(R\Delta t)^4}{4!} - ...). \tag{3.22}$$

If $\lfloor R\Delta t \rfloor$ is defined as the largest integer smaller than or equal to $R\Delta t$, the maximum term in the expansion cannot exceed

$$\frac{(R\Delta t)^{\lfloor R\Delta t \rfloor}}{\lfloor R\Delta t \rfloor!}. \tag{3.23}$$

If the computer uses D significant digits, a roundoff error of the order of $\varepsilon = (R\Delta t)^{\lfloor R\Delta t \rfloor}10^{-D}/\lfloor R\Delta t \rfloor!$ may be introduced. For example, having 13 significant figures, and $R\Delta t = 100$, the roundoff error can be as large as $\varepsilon = 1.07 \cdot 10^{42} \cdot 10^{-13} = 1.07 \cdot 10^{29}$.

Lapidus and Luus (Herman & Westfall, 1998:F7.2.3) have shown that for a matrix $\mathbf{A}$, the maximum term in the summation of any element in the Taylor expansion of $e^{\mathbf{A}\Delta t}$ cannot exceed

$$([\mathbf{A}]\,\Delta t)^n/n!, \tag{3.24}$$

where $n = \lfloor [\mathbf{A}]\,\Delta t \rfloor$ is the largest integer smaller than $[\mathbf{A}]\,\Delta t$. Roundoff errors therefore place a limit on the maximum step size Δt that can be used. In the ORIGEN-S code roundoff errors are limited (Herman & Westfall, 1998:F7.2.4) by requiring that

$$[\mathbf{A}]\,\Delta t \leq -2\ln(0.001) = 13.8155. \tag{3.25}$$

For stiff systems, $[\mathbf{A}]\,\Delta t \geq \max_i |a_{ii}|\,\Delta t \gg 0$, the above requirement must be satisfied by removing all the short half-life isotopes from the coefficient matrix, or by dividing the burnup time step Δt into smaller integration time steps Δt_i. In ORIGEN-S the short half-life isotopes are removed,

with the effect that significant departures from the exact result are possible for small burnup times (Thomas & Barber, 1994:309). This removal of isotopes can be used in conjunction with any of the methods described in this chapter. In order to compare the different methods, no isotopes are removed from the system of depletion equations, but the burnup time step is divided into smaller integration time steps. In Subsection 3.2.4 it is shown how to choose the integration step lengths.

3.2.3 Number of terms

Given a tolerance of accuracy for the local truncation error, $\varepsilon = 10^{-s}$, the Taylor expansion (3.3) must be continued for k terms, such that the $(k+1)$-st term will contribute less than ε. This implies that

$$\frac{(\mathbf{A}\Delta t)^{k+1}}{(k+1)!}\vec{N}_0 < \sum_{i=0}^{k}\frac{(\mathbf{A}\Delta t)^i}{i!}10^{-s}\vec{N}_0 \tag{3.26}$$

where the $<$ sign means that each element in the vector on the left hand side is smaller than the corresponding element in the right hand side vector. Using (3.19) this can also be stated as

$$\vec{C}_{k+1,j} < \sum_{i=0}^{k}\vec{C}_{i,j}10^{-s} \text{ for } j = 1\ldots n. \tag{3.27}$$

An upper limit, k_{max}, for the number of terms necessary to achieve (3.26) for all the isotopes is given by

$$\frac{([\mathbf{A}]\,\Delta t)^{k_{max}+1}}{(k_{max}+1)!} < 10^{-s}. \tag{3.28}$$

Although this means that

$$k_{max} > [\mathbf{A}]\,\Delta t - 1, \tag{3.29}$$

which may be quite large, the actual amount of terms necessary may be much smaller when the short half-life isotopes are in equilibrium. It is therefore much more efficient to test whether the given tolerance is reached, using (3.26), after each term is calculated and then to truncate the expansion when the tolerance is reached, rather than to use k_{max} terms.

A widely used stability criterion is that of A-stability, saying that a numerical method is A-stable if its region of absolute stability (defined in Subsection 5.6.4) contains the whole of the left-hand complex half-plane, $\mathrm{Re}(z) < 0$ (Cash 1985:71). For the scalar k-th order Taylor expansion, the complex number z will lie inside the region of stability if $|R(z)| < 1$, with $R(z)$ given by

$$R(z) = \sum_{i=0}^{k}\frac{z^i}{i!} \approx e^z - \frac{z^{k+1}}{(k+1)!}. \tag{3.30}$$

Assuming the coefficient matrix to be triangular, the k-th order Taylor expansion will be stable if all the eigenvalues, lying on the diagonal, satisfy $|R(z)| < 1$. Let $z = -\lambda_{max}\Delta t$ where λ_{max} is the largest decay constant. The k-th order Taylor expansion will then be A-stable if $k \geq k_{min}$, with k_{min} the smallest positive integer satisfying

$$\left|\frac{(\lambda_{max}\Delta t)^{k_{min}+1}}{(k_{min}+1)!}\right| < 1 - e^{-\lambda_{max}\Delta t}. \tag{3.31}$$

Table 3.1: Computer time and number of terms required in the Taylor expansion when solving the sample assembly problem using different step sizes

Step size ($\lambda_{max}\Delta t$)	Stiffness lower limit ($[A]\,\Delta t$)	Number of terms required (k)	Stability criteria (k_{min})	Number of time steps	Total Number of terms	Computer time (s x10^{-2})
1	1.36	4	1	375	1500	7.1
2	2.72	4	3	188	750	3.7
2.5	3.40	4	4	150	600	3.1
3	4.08	6	6	125	750	3.6
5	6.80	11	11	75	835	3.9
10	13.6	24	24	38	900	4.3
20	27.2	51	51	19	956	4.6

By approximating the right hand side of this inequality as one and using $\ln x! \geq x \ln x - x$, k_{min} is given as the smallest integer satisfying

$$k_{min} > \lambda_{max}\Delta te - 1. \tag{3.32}$$

Comparing Equations (3.29) and (3.32) it is clear that $k_{min} < k_{max}$ and the number of terms necessary must lie between these two limits.

3.2.4 Optimum step length

When large integration time steps are used, the number of terms in the Taylor expansion is limited by the stability criteria and when very small time steps are used, the number of terms is limited by the required accuracy. The optimum integration step size lies at the point where these two limitations meet. In the ORIGEN-S code a fixed step size is specified as given in (3.25). Table 3.1 gives an example how the computer time varies with different integration step lengths. This was calculated using the Taylor method as implemented in the ORIGEN-S code on the sample assembly problem, which will be described in Chapter 5, Subsection 5.2.1. For this test problem, using the ORIGEN-S limit (3.25), the computational time was 4.3x10^{-2} seconds, which is not far from the optimum 3.1x10^{-2} seconds. This optimum is found to be the smallest integration step size for which the number of terms required, k, is equal to the stability criteria minimum, k_{min}.

When a large integration time step is chosen, $[\mathbf{A}]\,\Delta t$ will be large. Although k_{max} will be large, after a few time steps the short half-life elements will be in equilibrium, requiring much less terms. k_{min} will also be large, so that a given tolerance, as specified by Eq. (3.26), is normally reached for $k < k_{min}$. The number of terms is then only limited by the stability criterion, $k = k_{min}$. If the time step Δt is divided into halves, the number of integration time steps will double, but the number of terms in each integration time step $k = k_{min}$ will be a bit less than half as can be seen from (3.32) and Table 3.1. Smaller time steps will therefore require a bit less computer time.

Algorithm 1 Taylor algorithm 1

1. Calculate [**A**] using (2.49) .
2. Do one Taylor step with $[\mathbf{A}]\,\Delta t = 1$ and let k_{min} be the number of terms in the expansion.
3. Use (3.32) to calculate the optimum step size Δt.
4. Do steps 5 and 6 for each time interval.
5. Let $\vec{N}_t = \vec{N}_{t-1}$.
6. Do steps 7 and 8 until $i \geq k_{min}$ and $C_{i,j} < N_{t,j}10^{-s}$, Eq. (3.27), for each isotope j.
7. Calculate $\vec{C}_i$ using (3.18).
8. Add every $\vec{C}_i$ to $\vec{N}_t$, Eq. (3.19): $\vec{N}_t = \vec{N}_t + \vec{C}_i$.

When $[\mathbf{A}]\,\Delta t$ is small (about one) the number of terms required is larger than k_{min} and therefore not determined by the stability criteria. As can be seen in Table 3.1 the number of terms, k, does not double if the time step doubles, but stays constant. Larger time steps will therefore, in this case, require much less flops.

Algorithm 1 shows how the Taylor method can be optimally implemented. One small integration time step, with $[\mathbf{A}]\,\Delta t = 1$, is done to find the minimum number of terms, k_1, that are necessary to obtain the required accuracy. The optimal integration time step Δt is then calculated by setting $k_{min} = k_1$ in (3.32).

3.2.5 Generalised uniformization technique

Because the coefficient matrix is essential non-negative, the uniformization technique of Sidje and Stewart (1999:351) can be used. In this technique the diagonal of the coefficient matrix is scaled with a constant $\alpha = D_{max} = \max_i(a_{ii})$ to form the non-negative matrix $\mathbf{P} = \mathbf{A} + \alpha\mathbf{I}$. The exponential of the coefficient matrix is then given by

$$e^{\mathbf{A}\Delta t}\vec{N}_0 = e^{-\alpha\Delta t}e^{\mathbf{P}\Delta t}\vec{N}_0. \tag{3.33}$$

The matrix exponential of $\mathbf{P}$ can be recursively calculated by computing $\vec{C}_i$,

$$\vec{C}_i = \frac{\mathbf{P}\Delta t}{i}\vec{C}_{i-1}, \tag{3.34}$$

which is combined with scaling the initial isotope densities,

$$\vec{C}_0 = e^{-\alpha\Delta t}\vec{N}_0, \tag{3.35}$$

or the final isotope densities,

$$e^{\mathbf{A}\Delta t}\vec{N}_0 = e^{-\alpha\Delta t}\sum_{i=0}^{k}\vec{C}_i, \tag{3.36}$$

by $e^{-\alpha\Delta t}$. In this method no integration time steps are necessary. Only when $e^{-\alpha\Delta t}$ is close to the smallest number that can be represented by the computer, the burnup time step has to be divided into smaller time steps. When double precision is used this will be necessary when $|\alpha\Delta t| > \ln(10^{300}) \simeq 700$.

This method of Sidje and Stewart can be generalised by using

$$\alpha = (1 - \beta)D_{max}, \tag{3.37}$$

with $0 < \beta < 1$. When $\beta = 1$, (3.33) reduces to the normal Taylor expansion, and when $\beta = 0$ the above uniformization technique is recovered. The higher the value of β, the smaller the maximum number of terms k_{max} will be and the fewer terms will be necessary in the expansion. If β is too high, roundoff errors will corrupt the answer.

Let $c = \beta\,[\mathbf{A}]\,\Delta t$. An upper limit for the roundoff error in the Taylor expansion of $e^{\mathbf{P}\Delta t}$, given by (3.23) with $n = \lfloor c \rfloor$, is

$$E_{max} = 2^{-D}\frac{c^c}{c!}e^{-\alpha\Delta t}, \tag{3.38}$$

with D the word length of the computer. Taking the logarithm, and making use of the fact that $\ln x! \geq x\ln x - x$, an upper limit for E_{max} is

$$\begin{aligned} \ln E_{max} &\leq \ln 2^{-D} + c\ln(c) - c\ln(c) + c - \alpha\Delta t \\ &= \ln 2^{-D} + (c - \alpha\Delta t), \end{aligned}$$

so that

$$E_{max} \leq 2^{-D}e^{(c-\alpha\Delta t)}. \tag{3.39}$$

Requiring that the roundoff errors will be smaller than 2^{-D}, (3.39) gives the requirement that $(c - \alpha\Delta t) \leq 0$, which can be rewritten as

$$\beta \leq \frac{1}{([\mathbf{A}]/D_{max} + 1)}. \tag{3.40}$$

Algorithm 2 shows how the generalised uniformization technique can be used to solve the fuel depletion problem. For the core and assembly depletion sample problems, that are considered in Chapter 5, the right-hand side of (3.40) is about 0.5 and 0.4 respectively. Using these values for β makes this new generalised method about twice as fast as the uniformization technique used by Sidje and Stewart, having $\beta = 0$.

3.2.6 Average isotope concentration

Sometimes it is required that the depletion module must give both the final isotope concentrations and the average isotope concentrations over the burnup time. The average isotope concentrations

Algorithm 2 Generalised uniformization Taylor algorithm

1. Calculate α using (3.40) and (3.37).
2. If $\alpha\Delta t > 700$ divide the total time step into smaller time steps such that $\alpha\Delta t < 700$.
3. Scale the initial isotope densities using (3.35).
4. Do steps 5 and 6 until $C_{i,j} < N_{1,j}10^{-s}$ for $j = 1 \ldots n$, (3.27).
5. Calculate $\vec{C}_i$ using (3.34) and index matrix storage.
6. Add $\vec{C}_i$ to $\vec{N}_1$, $\vec{N}_1 = \vec{N}_1 + \vec{C}_i$, as in (3.19).

can be computed, after the final isotope concentrations $\vec{N}(\Delta t)$ are calculated, with

$$\begin{aligned} \overline{\vec{N}} &= \frac{1}{\Delta t}\int_0^{\Delta t} e^{\mathbf{A}t'}\vec{N}_0 dt' \\ &= \frac{1}{\Delta t}\mathbf{A}^{-1}\left(e^{\mathbf{A}\Delta t} - \mathbf{I}\right)\vec{N}_0 \\ &= \frac{1}{\Delta t}\mathbf{A}^{-1}\left(\vec{N}(\Delta t) - \vec{N}_0\right). \end{aligned} \tag{3.41}$$

This formula has the disadvantage that it is expensive to invert large matrices and it is not always possible to invert a matrix exactly. If the isotope concentrations do not change much during an integration time step, the average concentrations can be approximated by

$$\overline{\vec{N}} = \frac{\vec{N}_0 + \vec{N}_1}{2}.$$

If the exact average concentrations are important, they can be directly calculated with a Taylor expansion

$$\overline{\vec{N}} = \frac{1}{\Delta t}\mathbf{A}^{-1}(e^{\mathbf{A}\Delta t} - \mathbf{I})\vec{N}(0) = \sum_{i=0}^{k}\frac{(\mathbf{A}\Delta t)^i}{(i+1)!}\vec{N}(0) \tag{3.42}$$

This is called the Volterra integral solution (Hermann & Westfall, 1998:F7.2.11). The final isotope concentrations can then be calculated as

$$\vec{N}_1 = \vec{N}_0 + \mathbf{A}\Delta t\overline{\vec{N}}. \tag{3.43}$$

In the Volterra integral solution k_{min} and k_{max} are one less than in the normal Taylor expansion and one less term will be necessary in (3.42). But the total number of matrix-vector multiplications will be the same because of the extra multiplication in (3.43).

3.3 Padé diagonal approximation

In this section it is shown how Arioli *et. al.* (1996:114) calculated the Padé approximation (Subsection 3.3.1), using the Horner algorithm (Subsection 3.3.2) and using scaling and squaring. In Subsection 3.3.3 it is shown how the implementation can be made more efficient by combining the

Algorithm 3 Padé algorithm 1(Arioli *et. al.*, 1996:114)

1. Determine the minimum integer $m = 2^k$ such that $\|A\Delta t\|_1 / m < 0.5$.
2. Compute $\mathbf{A} = \mathbf{A}\Delta t/m$.
3. Compute $\mathbf{C}_{qq}(\mathbf{A})$ and $\mathbf{D}_{qq}(\mathbf{A})$ using the Horner algorithm (Subsection 3.3.2).
4. Compute $e^A = \mathbf{C}_{qq}(\mathbf{A})^{-1}\mathbf{D}_{qq}(\mathbf{A})$, by solving $\mathbf{C}_{qq}(\mathbf{A})e^A = \mathbf{D}_{pq}(\mathbf{A})$, using Gaussian elimination.
5. Compute $e^{\mathbf{A}} = \left(e^{\mathbf{A}}\right)^{2^k}$ by squaring $e^{\mathbf{A}}$ k times.

scaling and squaring with time stepping. In Subsection 3.3.4 an upper limit for the roundoff errors of the Padé algorithms is given.

3.3.1 Implementation

Algorithm 3 shows how the diagonal Padé method was implemented by Arioli *et. al.*. The diagonal Padé approximation is the most efficient because if $p<q$, about qn^3 flops are required to calculate $\mathbf{C}_q^{-1}\mathbf{D}_p$ and the approximation is of order $p+q$. The diagonal Padé approximation $\mathbf{C}_q^{-1}\mathbf{D}_q$ uses the same number of flops, but has an order of $2q > p+q$ (Moler & Van Loan, 1978:808; Arioli *et. al.* 1996). The same argument can be used if $q > p$.

The method of scaling and squaring requires that $\mathbf{C}_q^{-1}\mathbf{D}_p$ be explicitly calculated. This makes it necessary to compute the two polynomials $\mathbf{C}_q$ and $\mathbf{D}_q$ explicitly and to invert the matrix $\mathbf{C}_q$. These full matrix computations are expensive for large systems. But it can be done more efficiently by using the block matrix storage scheme, given in Subsection 2.3.2.

3.3.2 Horner's algorithm

When a matrix polynomial has to be calculated explicitly, it can be done with the minimum amount of matrix-matrix multiplications by using the Horner algorithm.

Given a matrix polynomial $P(\mathbf{A})$ of order q,

$$P(\mathbf{A}) = b_0\mathbf{I} + b_1\mathbf{A} + b_2\mathbf{A}^2 + \ldots + b_q\mathbf{A}^q, \tag{3.44}$$

it can be computed using $s + r - 1$ matrix multiplications, where s is the largest integer smaller than $\sqrt{q}$ and r is the largest integer smaller than q/s (Golub and Van Loan, 1989:552). $s-1$ multiplications are used to calculate $\mathbf{A}^2, \ldots, \mathbf{A}^s$ and the r multiplications are used to calculate all

the products with the matrix $\mathbf{A}^s$:

$$\begin{aligned} P(\mathbf{A}) &= (b_0 + \ldots + b_{s-1}A^{s-1}) \\ &\quad + A^s\{b_s + \ldots + b_{2s-1}A^{s-1} \\ &\quad\quad + A^s[\ldots \\ &\quad\quad\quad + A^s(b_{rs} + \ldots + b_q A^q)\ldots]\}. \end{aligned} \tag{3.45}$$

3.3.3 Scaling and squaring

Because the Padé approximation, as given by (3.10), approximates the exponential at the origin, it has the same problem as the Taylor expansion with roundoff errors and stability when the problem is stiff. This is solved by scaling the coefficient matrix, computing the matrix exponential of the scaled matrix, and then squaring the matrix exponential:

$$e^{\mathbf{A}}\vec{N}_0 = \left(e^{\mathbf{A}/2^k}\right)^{2^k}\vec{N}_0.$$

Instead of squaring the matrix $e^{\mathbf{A}}$ k times and then multiplying it with the vector $\vec{N}_0$, which requires k full matrix operations, the matrix $e^{\mathbf{A}}$ can be multiplied 2^k times with the vector $\vec{N}_0$,

$$e^{\mathbf{A}}\vec{N}_0 = \underbrace{e^{\mathbf{A}/2^k}(e^{\mathbf{A}/2^k}(\ldots(e^{\mathbf{A}/2^k}}_{2^k \text{ times}}\vec{N}_0)\ldots)),$$

requiring 2^k matrix-vector operations. This effectively divides the burnup time into 2^k integration time steps.

If there are n^2 elements in the matrix $\mathbf{A}$ (or in the blocks of $\mathbf{A}$ when sub-matrices are used), squaring k times will need kn^3 flops. 2^k matrix-vector products will require about $2^k n^2$ flops and will therefore be better if

$$n > 2^k/k. \tag{3.46}$$

The two methods can be combined, as in Algorithm 4, by squaring k_1 times and multiplying 2^{k_2} times, with $k_1 + k_2 = k$ and k_2 the largest integer satisfying (3.46).

3.3.4 Roundoff errors

In this section it is shown how the repeated squaring and the matrix inversion of Algorithms 3 and 4 can introduce roundoff errors and an upper bound for the roundoff error is given.

When there are large transients at the beginning of the time interval Δt, but the isotopes are in equilibrium at the end of the time interval, it is possible that (Moler and Van Loan, 1978:804)

$$\left\|e^{\mathbf{A}\Delta t}\right\|_2 \ll \left\|e^{\mathbf{A}\Delta t/m}\right\|_2^m. \tag{3.47}$$

Algorithm 4 Padé algorithm 2 using integration time steps (Arioli *et. al.*, 1996:114)

1. Determine the minimum integer $m = 2^k$ such that $\|\mathbf{A}\Delta t\|_1 / m < 0.5$.
2. Compute $\mathbf{A} = \mathbf{A}\Delta t/m$.
3. Compute $\mathbf{C}_{qq}(\mathbf{A})$ and $\mathbf{D}_{qq}(\mathbf{A})$ using Horner's scheme.
4. Compute $e^A = \mathbf{C}_{qq}(\mathbf{A})^{-1}\mathbf{D}_{qq}(\mathbf{A})$, by solving $\mathbf{C}_{qq}(\mathbf{A})e^A = \mathbf{D}_{pq}(\mathbf{A})$ using Gaussian elimination.
5. Let k_2 be the largest integer satisfying (3.46) and $k_1 = m - k_2$.
6. Compute $e^{\mathbf{A}} = \left(e^{\mathbf{A}}\right)^{2^k}$ by squaring $e^{\mathbf{A}}$ k_1 times.
7. Compute $\vec{N} = e^{\mathbf{A}}\vec{N}$ by 2^{k_2} matrix-vector multiplications.

Then, although the roundoff errors introduced in Algorithm 3 are relative small compared to $\left\|e^{\mathbf{A}\Delta t/m}\right\|_2$, they may be relative large compared to $\left\|e^{\mathbf{A}\Delta t}\right\|_2$ because of the repeated squaring.

The inversion of the matrix $\mathbf{C}_q$ can also introduce roundoff errors. The condition of an invertible matrix $\mathbf{C}$ is denoted by cond($\mathbf{C}$) defined as (Moler & Van Loan, 1978:805)

$$\text{cond}(\mathbf{C}) \equiv \|\mathbf{C}\|_2 \left\|\mathbf{C}^{-1}\right\|_2 . \tag{3.48}$$

The larger the condition of a matrix is, the larger round errors can be introduced when the matrix is inverted.

Moler and Van Loan (1978:808) have shown that for large enough q,

$$\text{cond}(\mathbf{C}_q(\mathbf{A})) \geq e^{(\lambda_1 - \lambda_n)/2}, \tag{3.49}$$

where λ_1 and λ_n are the largest and smallest eigenvalue of $\mathbf{A}$. In Algorithms 3 and 4 the matrix is scaled such that

$$m \simeq 2\|\mathbf{A}\Delta t\|_1 \simeq 4D_{max}\Delta t, \tag{3.50}$$

from (2.46), where D_{max} is the maximum absolute diagonal element in $\mathbf{A}$. Because the coefficient matrix is nearly triangular, the diagonal elements can be taken as estimates for the eigenvalues. It then follows that the largest eigenvalue can be approximated by

$$\lambda_1 \simeq \frac{D_{max}\Delta t}{m} \simeq \frac{1}{4}, \tag{3.51}$$

thereby limiting the condition of $\mathbf{C}_q$.

When (3.50) is used to calculate the scaling factor m, Arioli *et. al.* (1996) have proven that for essentially non-negative matrices, an upper bound for the roundoff error due to the inversion and squaring of the matrices is:

$$\frac{\|fl\left(\mathbf{R}_q(\mathbf{A}\Delta t/m)^m\right) - \mathbf{R}_q(\mathbf{A}\Delta t/m)^m\|_1}{\|\mathbf{R}_q(\mathbf{A}\Delta t/m)^m\|_1} \leq um\left\{n + e^{1/2}\left[1 + q(n+1)\right]\left[1 + e^{3/2}\right]\right\} \tag{3.52}$$

where $\mathbf{R}_q(x) = \left(\mathbf{C}_q(x)\right)^{-1}\mathbf{D}_p(x)$ is the (q,q) Padé approximation computed with no roundoff errors

Algorithm 5 Krylov algorithm to compute $w(t) = e^{\mathbf{A}t}v$ (Sidje, 1998:135)

$\vec{w} = \vec{v}; t_k = 0$
$\mathbf{H} = \text{zeros}[m+2, m+2]$
while $t_k < t$ do
  $\vec{v} = \vec{w}$; $\beta = \|\vec{v}\|_2$
  $\vec{v}_1 = \vec{v}/\beta$
  for $j = 1 : m$ do *{Arnoldi process}*
    $\vec{p} = \mathbf{A}\vec{v}_j$
    for $i = 1 : j$ do
      $h_{ij} = \vec{v}_i^T \vec{p}$
      $\vec{p} = \vec{p} - h_{ij}\vec{v}_i$
    end
    $h_{j+1,j} = \|\vec{p}\|_2$
    if $h_{j+1,j} \leq tol \|\mathbf{A}\|$ happy-breakdown
    $\vec{v}_{j+1} = \vec{p}/h_{j+1,j}$
  end
  $h_{m+2,m+1} = 1$
  repeat
    $\tau =$ *step-size*
    $\mathbf{F} = e^{\tau \mathbf{H}}$ *{Padé approximation}*
    $\vec{w} = \beta \mathbf{V} \vec{F}_1$
    err_loc=local error estimate (Sidje, 1998:137)
  until *err_loc*$\leq \delta tol$
  $t_k = t_k + \tau$
end

and $fl(\mathbf{R}_q(x))$ the Padé approximation computed using floating point calculations with a relative precision of $u = 2^{-t}$.

For example, in an assembly depletion calculation, if the coefficient matrix is 100-by-100, and a (6,6) Padé approximation is used, using double precision ($u = 10^{-14}$), the upper bound for the roundoff error will be $5.6 \cdot 10^{-11} m$, where m is about $2\|\mathbf{A}\Delta t\|_1$.

3.4 Krylov subspace approximation

In this section an algorithm given by Sidje (1998) for using the Krylov subspace approximation is briefly described and an upper limit for the roundoff error of this algorithm is given.

3.4.1 Algorithm of Sidje

In the Krylov approximation, the element in the subspace K_p, given by (3.12), that best approximates $e^{\mathbf{A}\Delta t}\vec{N}_0$ is calculated. First an orthonormal basis $\mathbf{V}_p$ is calculated for K_p, using the Arnoldi algorithm (Saad, 1992:210; Gallopoulos & Saad, 1992:1238). The first basis vector $\vec{v}_1 = \mathbf{V}_p \vec{e}_1$ is taken as

$$\vec{v}_1 = \frac{\vec{N}_0}{\left\|\vec{N}_0\right\|}. \tag{3.53}$$

In the Arnoldi process an upper Hessenberg matrix $\mathbf{H}_p$ is formed, having the relation

$$\mathbf{H}_p = \mathbf{V}_p^T \mathbf{A} \mathbf{V}_p. \tag{3.54}$$

$\mathbf{H}_p$ represents the projection of $\mathbf{A}$ to the subspace K_p with respect to the basis $\mathbf{V}_p$. The optimal vector $\vec{N}_{opt}$ approximating $e^{\mathbf{A}\Delta t}\vec{N}_0$ is then (Steward & Leyk, 1996:360)

$$\begin{aligned} \vec{N}_{opt}(\Delta t) &= \mathbf{V}_p \mathbf{V}_p^T e^{\mathbf{A}\Delta t}\vec{N}_0 \\ &= \left\|\vec{N}_0\right\| \mathbf{V}_p \left(\mathbf{V}_p^T e^{\mathbf{A}\Delta t}\mathbf{V}_p\right)\vec{e}_1. \end{aligned} \tag{3.55}$$

$\mathbf{V}_p^T e^{\mathbf{A}\Delta t}\mathbf{V}_p$ can be approximated by $e^{\mathbf{H}_p\Delta t}$ (Gallopoulos & Saad, 1992:1240), giving the Krylov approximation of the matrix exponential

$$e^{\mathbf{A}\Delta t}\vec{N}_0 \approx \left\|\vec{N}_0\right\| \mathbf{V}_p e^{\mathbf{H}_p\Delta t}\vec{e}_1. \tag{3.56}$$

$\mathbf{H}_p$ is a dense matrix, having dimensions p-by-p. When p is much less than the dimension of $\mathbf{A}$, it will be much faster to calculate the matrix exponential of $\mathbf{H}_p$ in (3.56) than to calculate the matrix exponential of $\mathbf{A}$.

Algorithm 5 shows the algorithm used in EXPOKIT (Sidje, 1998) to calculated (3.56). Sidje (1998:11) also shows how (3.56) can be implemented using a time-stepping strategy. This strategy is more stable and accurate than using a single time step. The time-stepping is done by calculating a local error estimate at each step. If the error is larger than a given tolerance, the step size is decreased and the previous time step is repeated. If the error is much smaller, the step size is increased.

3.4.2 Accuracy

Because the n eigenvalues of $\mathbf{A}$ are approximated by the p eigenvalues of $\mathbf{H}_p$, Krylov subspace methods benefit from a clustering of eigenvalues in $\mathbf{A}$ (Hochbruck & Lubich, 1997:1918). A rough upper bound for the error introduced in the Krylov approximation is (Steward & Leyk, 1996:360; Saad, 1992:220)

$$\left\|e^{\mathbf{A}\Delta t}\vec{N}_0 - \mathbf{V}_p e^{\mathbf{H}_p\Delta t}\mathbf{V}_p^T\vec{N}_0\right\|_2 < \left\|\vec{N}\right\|_2 \frac{\|\mathbf{A}\Delta t\|_2^p}{p!2^{p-1}}. \tag{3.57}$$

Gallopoulos and Saad (1992:1242A) give a smaller upper bound, using the logarithmic norm, which is more difficult to calculate.

3.5 Chebyshev rational approximation

In this section it is shown how the Chebyshev rational approximation can be calculated using the Horner algorithm (Subsection 3.5.1), or by using partial fractions (Subsection 3.5.2) as implemented by Sidje (1998). In Subsection 3.5.3 the accuracy and stability of the algorithms are discussed.

Algorithm 6 Chebyshev rational approximation using Horner's algorithm

1. Calculate $\vec{N}_a(\Delta t) = \mathbf{D}_{14}(\mathbf{A}\Delta t)\vec{N}_0$, using 14 matrix-vector products.
2. Calculate $\mathbf{C}_{14}(\mathbf{A}\Delta t)$ with Horner's algorithm, using 6 matrix-matrix products.
3. Solve $\mathbf{C}_{14}(\mathbf{A}\Delta t)\vec{N}(\Delta t) = \vec{N}_a(\Delta t)$.

3.5.1 Horner's algorithm

Algorithm 6 shows how the Chebyshev rational approximation method, (3.7) with coefficients calculated using (3.13), can be implemented using the Horner algorithm. p matrix-vector multiplications are used to calculate $\vec{N}_a(\Delta t) = \mathbf{D}_p(\mathbf{A}\Delta t)\vec{N}_0$ and the Horner algorithm (Subsection 3.3.2) is used to calculate $\mathbf{C}_q(\mathbf{A}\Delta t)$. For $p = 14$, the Horner algorithm needs 6 matrix-matrix products to calculate $\mathbf{C}_{14}(\mathbf{A}\Delta t)$. Lastly, the system $\mathbf{C}_q(\mathbf{A}\Delta t)\vec{N}(\Delta t) = \vec{N}_a(\Delta t)$ is solved.

3.5.2 Partial fractions

When using partial fractions, the matrix $\mathbf{C}_q(\mathbf{A})$ does not need to be explicitly calculated, therefore having the advantage that no matrix-matrix calculations are necessary. The price to pay is that complex number computations have to be done. But even on scalar machines, the partial fraction methods represent the best way of computing $\vec{N}_1$, requiring fewer operations than a straight-forward calculation of (3.7) (Gallopoulos and Saad, 1992:1245). In the partial fraction implementations, the ratio of the two polynomials of (3.6), with $q = p$, is divided into partial fractions

$$\begin{aligned}\vec{N}(\Delta t) &= \left(c_0\mathbf{I} + c_1\Delta t\mathbf{A} + \ldots + c_p\Delta t^p\mathbf{A}^p\right)^{-1}\left(d_0\mathbf{I} + d_1\mathbf{A}\Delta t + \ldots + d_p\Delta t^p\mathbf{A}^p\right)\vec{N}_0 \\ &= \alpha_0\vec{N}_0 + \sum_{i=1}^{p}\alpha_i(\mathbf{A}\Delta t - \lambda_i\mathbf{I})^{-1}\vec{N}_0. \end{aligned} \tag{3.58}$$

This can be done for the Chebyshev approximation because all the roots λ_i are distinct (Saad, 1992:214). The complex coefficients α_i and λ_i are given by Gallopoulos and Saad (1992:1261) for $p = 10$ and 14. Because all the roots are complex they form complex conjugate pairs and it is therefore only necessary to use the real part of the fraction twice for one root of each pair:

$$\vec{N}(\Delta t) = \alpha_0\vec{N}_0 + 2\sum_{i=1}^{p/2}\mathbb{R}\left(\alpha_i(\mathbf{A}\Delta t - \lambda_i\mathbf{I})^{-1}\right)\vec{N}_0.$$

This can be computed by solving

$$(\mathbf{A}\Delta t - \lambda_i\mathbf{I})\vec{y}_i(\Delta t) = \vec{N}_0, \qquad i = 1\ldots p/2, \tag{3.59}$$

to compute $\vec{y}_i$ and then adding the terms

$$\vec{N}(\Delta t) = \alpha_0\vec{N}_0 + 2\sum_{i=1}^{p/2}\mathbb{R}\left(\alpha_i\vec{y}_i(\Delta t)\right). \tag{3.60}$$

Algorithm 7 Chebyshev rational approximation using partial fractions

1. Calculate $\vec{y}_i$ by solving the linear systems (3.59), using block matrix storage and $p = 14$.
2. Add the real parts of all the $\vec{y}_i$, Eq. (3.60).

This partial fraction algorithm is shown in Algorithm 7. Because no full matrix computations are used, the computer time is of order $\mathcal{O}(n^2)$. The linear equations (3.59) can be solved very fast when the block storage scheme and the property that the coefficient matrix is nearly triangular, are used.

3.5.3 Accuracy

In order to calculate the matrix exponential accurately using a polynomial method, two conflicting goals must be met. The first one is that the method must approximate the exponential of the eigenvalues to a high accuracy and secondly it must fit the exponential to a high order at the origin. This is proven in Appendix A. The Taylor and Padé approximation methods focused on the second goal in choosing the polynomial constants and use scaling to reach the first goal. In the *pq*-Chebyshev rational approximation the coefficients of (3.6) are chosen by minimising (3.13). This ensures that the first goal is met. By approximating the exponential on the whole interval $(-\infty, 0)$, the region of stability includes the whole negative real line and scaling is unnecessary for stiff systems. Cody *et. al.* (1969:60) have calculated that the maximum absolute difference between e^x and the value calculated by the *pp*-Chebyshev rational approximation in the interval $[-\infty, 0)$ is $1.8\text{x}10^{-14}$ for $p = 14$.

The Chebyshev rational approximation has the problem that the second goal of a high order fit at the origin is not fully met. The result is that it is possible that the matrix exponential will be less accurate if there are confluent or nearly confluent eigenvalues. The eigensystem may therefore have a subtle influence on this method, although it is not explicitly involved. This may make the Chebyshev approximation unreliable when the system of eigenvectors is poorly conditioned (Sidje, 1998:304). A further problem is that the Chebyshev approximation is not A-acceptable, because small eigenvalues near the origin may be amplified (Gallopoulos & Saad, 1992:1248).

3.6 Summary

Various algorithms were described in this chapter, using different polynomial methods, different ways to calculate the matrix polynomials, and different methods to handle stiff systems, as shown in Table 3.2. These algorithms were implemented in FORTRAN and compared with each other, as will be shown in Chapter 5, to determine which methods perform the best for different depletion problems.

Table 3.2: Summary of polynomial expansion algorithms given in Chapter 3

Algorithm	Polynomial approximation method	Technique used for stiff system	Polynomial evaluation
Algorithm 1	Taylor	Integration time steps	Recursive implementation
Algorithm 2	Taylor with uniformization	Very high order polynomial expansion	Recursive implementation
Algorithm 3	Padé	Scaling and squaring	Horner's scheme
Algorithm 4	Padé	Scaling, squaring and integration steps	Horner's scheme
Algorithm 5	Krylov	Scaling and squaring	Projection on small subspace
Algorithm 6	Chebyshev	-	Horner's scheme
Algorithm 7	Chebyshev	-	Partial fractions

Chapter 4

Analytical Methods

In Chapter 3 some algorithms using polynomial methods which can be used to solve the fuel depletion equation were given. In this chapter, algorithms for two other classes of methods are given, analytical methods and preconditioning methods. First it is shown in Section 4.1 how a triangular system can be solved analytically, as is done in the GAUGE code (Wagner, 1968), or by using the recursive method of Parlett (1976). In Section 4.2 it is then shown how a non-triangular system can be solved by using its Schur decomposition to make it triangular. Secondly, it is shown how the Runge-Kutta preconditioning method of Castillo and Saad (1998) can be used to solve the fuel depletion equations if the solution of a similar system of equations is known. Although this technique was already proposed by Lawson (1967), it is not a very well known technique.

4.1 Analytical solution of a triangular system

In this section, the source of roundoff errors in analytical methods is first discussed and two equations are derived to estimate the maximum roundoff error. It is then shown how the algorithms of GAUGE and Parlett can be used to solve a triangular system of fuel depletion equations.

4.1.1 Roundoff errors

To explain the roundoff errors that can be introduced in analytical methods due to the finite word length of a computer, consider the 3x3 triangular coefficient matrix

$$\mathbf{A} = \begin{bmatrix} -a_{11} & 0 & 0 \\ a_{21} & -a_{22} & 0 \\ a_{31} & a_{32} & -a_{33} \end{bmatrix}. \tag{4.1}$$

The matrix exponential of this matrix is

$$\mathbf{F} = e^{\mathbf{A}t} = \begin{bmatrix} e^{-a_{11}t} & 0 & 0 \\ a_{21}\frac{e^{-a_{11}t}-e^{-a_{22}t}}{a_{22}-a_{11}} & e^{-a_{22}} & 0 \\ a_{31}\frac{e^{-a_{11}t}-e^{-a_{33}t}}{a_{33}-a_{11}} + \frac{a_{32}a_{21}}{(a_{33}-a_{11})}\left[\frac{e^{-a_{11}t}-e^{-a_{22}t}}{(a_{22}-a_{11})} - \frac{e^{-a_{22}t}-e^{-a_{33}t}}{(a_{33}-a_{22})}\right] & a_{32}\frac{e^{-a_{22}t}-e^{-a_{33}t}}{a_{33}-a_{22}} & e^{-a_{33}} \end{bmatrix}. \tag{4.2}$$

In general, when two eigenvalues a_{ii} and a_{jj}, $i \neq j$, are close together, $a_{jj} = a_{ii}(1+\epsilon)$, $\epsilon \ll 1$, large roundoff errors may be introduced during the subtractions in the numerator and denominator (Moler & Van Loan, 1978:820; Thomas & Barber, 1993:408) when computing the element F_{ij},

$$F_{ij} = a_{ij}\frac{e^{-a_{ii}t} - e^{-a_{jj}t}}{a_{jj} - a_{ii}}. \tag{4.3}$$

Let the roundoff errors that are introduced be $e_1 = a_{ii}10^{-s}$ and $e_2 = e^{-a_{ii}t}10^{-s}$ respectively, with s being the available number of significant digits to represent a floating point value by the computer. These two roundoff errors then introduce an error $\Delta F_{ij}(e_1, e_2)$ in F_{ij} given by

$$F_{ij} + \Delta F_{ij}(e_1, e_2) = a_{ij}\frac{e^{-a_{ii}t} - e^{-a_{jj}t} + e_2}{a_{jj} - a_{ii} + e_1}. \tag{4.4}$$

The change in F_{ij} due to e_1 only is

$$\left|\frac{\Delta F_{ij}(e_1)}{F_{ij}}\right| = \frac{10^{-s}}{\epsilon + 10^{-s}} \simeq \frac{10^{-s}}{\epsilon}, \tag{4.5}$$

and due to e_2 only

$$\frac{\Delta F_{ij}(e_2)}{F_{ij}} = \frac{10^{-s}}{1 - e^{-\epsilon a_{ii}t}} \simeq \frac{10^{-s}}{\epsilon a_{ii}t}. \tag{4.6}$$

The first roundoff error, e_1, is independent of the burnup time step t and the size of the eigenvalue a_{ii}, but the second roundoff error, e_2, decreases with larger time steps and larger eigenvalues. Unlike numerical methods where roundoff errors build up over large time intervals, the roundoff errors in analytical methods can therefore decrease with increasing time step size. Eqs. (4.5) and (4.6) can be used to warn the user when roundoff errors larger than the specified tolerance can be expected.

In numerical methods that explicitly evaluate expression (4.3), the roundoff errors for nearly confluent eigenvalues $a_{jj} = a_{ii}(1+\epsilon)$, can be eliminated by using the truncated Taylor expansion

$$\frac{e^{-a_{ii}t} - e^{-a_{jj}t}}{a_{jj} - a_{ii}} \simeq te^{-a_{ii}t}(1 - \frac{\epsilon a_{ii}t}{2}), \tag{4.7}$$

which is accurate to order $\mathcal{O}(\epsilon^2)$.

4.1.2 GAUGE code

The GAUGE code uses the eigenvalues and eigenvectors (Wagner, 1968:33) of the triangular coefficient matrix to solve the fuel depletion equation (1.1). When **A** is triangular, the eigenvalues lie

Algorithm 8 GAUGE algorithm (Wagner, 1968:33)

1. Construct the triangular matrix $\mathbf{V}$ containing the eigenvectors, with the diagonal elements equal to one, $v_{ii} = 1$, by solving the eigenvalue problem (4.8):

$$v_{ij} = \frac{a_{ij} + \sum_{k=j+1}^{i-1} a_{ik} v_{kj}}{a_{jj} - a_{ii}}. \tag{4.11}$$

2. Calculate the diagonal elements of $\mathbf{B}$ with

$$b_{ii} = N_i - \sum_{j=1}^{i-1} v_{ij} N_j$$

and the triangular matrix $\mathbf{VB}$ with $b_{ij} = v_{ij} b_{jj}$.

3. Calculate the final isotope concentration using (4.10):

$$\vec{N}(t) = \sum_{j=1}^{N} \vec{B}_j e^{-a_{jj} t}, \tag{4.12}$$

where $\vec{B}_j$ is the j-th column of the triangular matrix $\mathbf{VB}$.

on the diagonal, and the eigenvectors can be computed by solving the eigenvalue problem

$$(\mathbf{A} - a_{ii}\mathbf{I})\vec{v}_i = 0. \tag{4.8}$$

The matrix exponential has the property that if $\mathbf{A} = \mathbf{VDV}^{-1}$, then $e^{\mathbf{A}} = \mathbf{V}e^{\mathbf{D}}\mathbf{V}^{-1}$(Moler & Van Loan, 1978:818). When $\mathbf{D}$ is a diagonal matrix with the eigenvalues a_{ii} on the diagonal and $\mathbf{V}$ is a triangular matrix containing the corresponding eigenvectors $\vec{v}_i$ of the triangular matrix $\mathbf{A}$, then (4.8) can be written as $\mathbf{A} = \mathbf{VDV}^{-1}$. The solution of the fuel depletion equation is then

$$\vec{N}_1 = \mathbf{V}e^{\mathbf{D}}\mathbf{V}^{-1}\vec{N}_0. \tag{4.9}$$

By constructing a diagonal matrix $\mathbf{B}$ having the elements of $\mathbf{V}^{-1}\vec{N}_0$ on its diagonal, it follows that $\mathbf{V}^{-1}\vec{N}(t_0) = \mathbf{B}\vec{I}$, where $\vec{I} = col\left[\begin{array}{cccc} 1 & 1 & \ldots & 1 \end{array}\right]$. Expression (4.9) can then be written as

$$\begin{aligned} \vec{N}(t_1) &= \mathbf{V}e^{\mathbf{D}}\mathbf{B}\vec{I} \\ &= (\mathbf{VB})e^{\mathbf{D}}\vec{I}. \end{aligned} \tag{4.10}$$

Algorithm 8 shows how (4.10) is solved in the GAUGE code.

When two eigenvalues are close together, the roundoff errors e_1 and e_2 will be introduced in (4.11) and (4.12) respectively. Because they are not introduced together as in (4.4), (4.7) can not be used to eliminate the roundoff errors. When the matrix $\mathbf{A}$ does not have a complete set of linearly independent eigenvectors, $\mathbf{V}^{-1}$ can not be calculated (Moler & Van Loan, 1978:820) and the GAUGE method can not be used. When cond($\mathbf{V}$) is large, the computation of $\mathbf{B}$, which implicitly involve $\mathbf{V}^{-1}$, may introduce large roundoff errors.

Algorithm 9 Parlett recursive algorithm (Assuming a lower triangular coefficient matrix **A**)

1. Calculate the diagonal elements of the matrix exponential, $\mathbf{F}_{rr} = e^{\mathbf{A}_{rr}t}$, $r = 1 \ldots n$.
2. For each sub-diagonal below the main diagonal, starting with the first sub-diagonal and going downward, do step 3.
3. For each element in the sub-diagonal calculate $\mathbf{F}_{rs}$ using (4.15).
4. Calculate the matrix-vector product $\vec{N}(t) = \mathbf{F}\vec{N}_0$.

4.1.3 Parlett recursive algorithm

Parlett (1976) showed that a simple relation exists among the elements of $\mathbf{F} = f(\mathbf{A})$ when f is an analytic function and matrix $\mathbf{A}$ is triangular. When $\mathbf{A}$ is a lower block triangular matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & 0 & \cdots & 0 \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{n1} & \mathbf{A}_{n2} & \cdots & \mathbf{A}_{nn} \end{bmatrix}, \tag{4.13}$$

the elements of the matrix $\mathbf{F} = f(\mathbf{A})$ can be calculated recursively with the formula

$$\mathbf{A}_{rr}\mathbf{F}_{rs} - \mathbf{F}_{rs}\mathbf{A}_{ss} = \sum_{k=0}^{r-s-1} (\mathbf{F}_{r,r-k}\mathbf{A}_{r-k,s} - \mathbf{A}_{r,s+k}\mathbf{F}_{s+k,s}), \tag{4.14}$$

for $r > s$. We start with the diagonal elements $\mathbf{F}_{rr}$, $r = 1, \ldots, n$ and end with $\mathbf{F}_{n1}$. When $\mathbf{A}$ is lower triangular, the above equation reduces to the scalar equation

$$F_{rs} = \frac{\sum_{k=0}^{r-s-1}(F_{r,r-k}A_{r-k,s} - A_{r,s+k}F_{s+k,s})}{A_{rr} - A_{ss}}, \tag{4.15}$$

which can also be expressed as

$$F_{rs} = \frac{A_{rs}(F_{rr} - F_{ss}) + \sum_{k=1}^{r-s-1}(F_{r,s+k}A_{s+k,s} - A_{r,s+k}F_{s+k,s})}{A_{rr} - A_{ss}}. \tag{4.16}$$

Using $\mathbf{F} = e^{\mathbf{A}}$, the diagonal elements, $F_{ii} = e^{A_{ii}}$, are first computed and one of the above two expressions is then used to calculate the other elements of the matrix exponential, as shown in Algorithm 9. The Parlett recursive method is mathematically equivalent to the method used in the GAUGE code. When two eigenvalues, A_{rr} and A_{ss}, are nearly confluent, Eq. (4.7) can be used for the first term in (4.16), but the second term may still introduce roundoff errors. This can be seen when a_{11} is close to a_{33} in the 3x3 example (4.1) by rewriting the second term of $\mathbf{F}_{31}$ in (4.2) as

$$\begin{aligned} & \frac{a_{32}a_{21}}{(a_{33} - a_{11})}\left[\frac{e^{-a_{11}t} - e^{-a_{22}t}}{(a_{22} - a_{11})} - \frac{e^{-a_{22}t} - e^{-a_{33}t}}{(a_{33} - a_{22})}\right] \\ = & \frac{a_{32}a_{21}}{(a_{33} - a_{22})}\left[\frac{e^{-a_{11}t} - e^{-a_{22}t}}{(a_{22} - a_{11})} - \frac{e^{-a_{11}t} - e^{-a_{33}t}}{(a_{33} - a_{11})}\right]. \end{aligned}$$

Algorithm 10 QR iterations

$\mathbf{T}_0 = \mathbf{A}$
for k=1,2,...
  Compute orthogonal $\mathbf{U}_k$ and triangular $\mathbf{R}_k$ such that $\mathbf{T}_{k-1} = \mathbf{U}_k\mathbf{R}_k$, using QR factorisation.
  $\mathbf{T}_k = \mathbf{R}_k\mathbf{U}_k$

The last term has the same form as (4.3) and can introduce roundoff errors.

As in the GAUGE method, the block storage scheme can also be used in the Parlett method to store both matrices **A** and **F**. This speeds up the summations in (4.15) and (4.16) because only the terms that use elements inside the blocks are non-zero.

4.2 Schur decomposition method

In this section two algorithms are shown which can be used to solve the fuel depletion equations analytically when they include feedback reactions. It is first shown how it is possible to transform the system of equations to a triangular system without feedback reactions by using QR iterations. Then it is shown how the method can be implemented using the block matrix storage scheme.

4.2.1 QR factorisation

The coefficient matrix **A** can be transformed to another matrix **T** by using the property of the matrix exponential (Moler & Van Loan, 1978:818) that

$$e^{\mathbf{A}} = \mathbf{Q}e^{\mathbf{T}}\mathbf{Q}^{-1} \quad \text{with} \quad \mathbf{T} = \mathbf{Q}^{-1}\mathbf{A}\mathbf{Q}, \tag{4.17}$$

for any invertible matrix **Q**. By using an orthogonal matrix **Q**, having the property that $\mathbf{Q}^{-1} = \mathbf{Q}^T$, no matrix inversions have to be done.

Golub and Van Loan (1989:335) have proven that for any real n-by-n matrix **A**, there exist an orthogonal matrix **S** such that $\mathbf{A} = \mathbf{S}\mathbf{T}\mathbf{S}^T$, where **T** is a quasi-triangular matrix, with the real eigenvalues of **A** on the diagonal and the complex eigenvalues represented by 2x2 blocks on the diagonal. The matrices **S** and **T** are called the Schur decomposition of **A** and the columns of **S**, $\vec{S}_i$, are referred to as Schur vectors. They also showed that the matrices $\mathbf{T}_k$ constructed in the QR iterations, Algorithm 10, converge to an upper triangular matrix **T**, giving the orthogonal Schur matrix $\mathbf{S} = \mathbf{U}_0\mathbf{U}_1...\mathbf{U}_k$ (Golub & Van Loan, 1989:351). It is also shown that each iteration in the QR factorization needs $O(n^3)$ flops for a general matrix **A**, but only $O(n^2)$ when **A** is in Hessenberg form (triangular but with non-zero elements on the first sub-diagonal). Any matrix can be transformed to Hessenberg form by using Givens rotations, where each rotation is used to make one element above the first sub-diagonal zero, or by using Householder reflections, where each reflection can transform a whole column to zeros above the first sub-diagonal (Golub & Van Loan, 1989:195-208).

Algorithm 11 Schur decomposition algorithm using GAUGE routine

1. For each diagonal block matrix $\mathbf{D}_i$ reduce the matrix to a Hessenberg matrix $\mathbf{H}_i$ using the Givens rotations $\mathbf{G}_i$:
$$\mathbf{H}_i = \mathbf{G}_i^T \mathbf{D}_i \mathbf{G}_i,$$
using QZHES.

2. Compute the Schur decomposition of each diagonal block matrix $\mathbf{D}_i$,
$$\mathbf{H}_i = \mathbf{S}_i \mathbf{T}_i \mathbf{S}_i^T,$$
by using the QR algorithm DHSEQR, where $\mathbf{S}_i$ is an orthogonal matrix containing the Schur vectors and $\mathbf{T}_i$ is a quasi-triangular matrix . Instead of forming $\mathbf{S}_i$ explicitly, the QR algorithm can be implemented to form the product $\mathbf{Q}_i = \mathbf{G}_i \mathbf{S}_i$. Then $\mathbf{T}_i = \mathbf{Q}_i^T \mathbf{D}_i \mathbf{Q}_i$.

3. For the fission product block matrices $\mathbf{F}_{ij}$ compute $\mathbf{T}_{ij} = \mathbf{Q}_j^T \mathbf{F}_{ij} \mathbf{Q}_i$.

4. Compute the matrix-vector product $\vec{N}_1 = \mathbf{Q}\vec{N}(t_0)$.

5. Solve $\vec{N}_2 = e^{\mathbf{T}\Delta t}\vec{N}_1$ using the GAUGE code, Algorithm 8 (or Parlett's method).

6. Compute the matrix-vector product $\vec{N}(t_1) = \mathbf{Q}^T \vec{N}_2$.

Algorithm 11 shows how the Schur decomposition is normally implemented by first transforming the coefficient matrix, using the notation of Eq. (2.39), to Hessenberg form before doing the Schur decomposition. Because the coefficient matrix has only a few terms above the diagonal, and fewer above the first sub-diagonal, it is more effective to use Givens rotations to transform the matrix to Hessenberg form than Householder reflections.

In Algorithm 11 the transformation is done by the subroutine QZHES that form part of the package of eigenvalue routines EISPACK (Smith *et. al.*, 1976) and the Schur decomposition is done by the subroutine DHSEQR of the package LAPACK (Anderson *et. al.*, 1999). The depletion problem can then be solved by computing
$$e^{\mathbf{A}}\vec{N}_0 = \mathbf{Q}\left(e^{\mathbf{T}}(\mathbf{Q}^T\vec{N}_0)\right), \tag{4.18}$$
using matrix-vector multiplications and computing $e^{\mathbf{T}}$ using one of the algorithms of Section 4.1.

Instead of using (4.10) when solving the triangular system, (4.9) can also be used if the eigenvectors are known. This is used in Algorithm 12 which is mathematically equivalent to Algorithm 11. In Algorithm 12 the program HQR from the LAPACK package is used to calculate both the Schur decomposition and eigenvectors of the coefficient matrix, giving $\mathbf{A} = \mathbf{QVDV}^{-1}\mathbf{Q}^T$, where $\mathbf{V}$ is triangular and $\mathbf{D}$ diagonal.

The QR factorisation gives $\mathbf{S} = \mathbf{QR}$, where $\mathbf{Q}$ is orthogonal and $\mathbf{R}$ is triangular. The condition of $\mathbf{S}$ is the same as the condition of $\mathbf{R}$ (Moler & Van Loan, 1978:822) which can be estimated reliably and efficiently in order to estimate the roundoff errors that can be introduced in solving the triangular system.

The Schur decomposition, using QR iterations, requires about $25n^3$ flops and the error introduced, $\mathbf{E}$, defined by $\mathbf{Q}^T(\mathbf{A}+\mathbf{E})\mathbf{Q} = \mathbf{T}$, is about $\|\mathbf{E}\|_2 = u\,\|\mathbf{A}\|_2$ which is well behaved (Golub & Van

Algorithm 12 Schur decomposition algorithm using eigenvectors

1. Reduce $\mathbf{A}$ to a Hessenberg matrix $\mathbf{H}$ using the Givens rotations $\mathbf{G}$, giving $\mathbf{A} = \mathbf{GHG^T}$, by using QZHES.
2. Compute the Schur decomposition $\mathbf{H} = \mathbf{STS}^T$, and form the product $\mathbf{Q} = \mathbf{GS}$ explicitly using HQR.
3. Compute the eigenvectors $\vec{V}_i$ of $\mathbf{T}$ using Eq. (4.11) with HQR, giving $\mathbf{A} = \mathbf{QVDV}^{-1}\mathbf{Q}^T$.
4. Compute the matrix-vector product $\vec{N}_1 = \mathbf{Q}^T\vec{N}(t_0)$.
5. Solve the triangular system $\vec{N}_2 = \mathbf{V}^{-1}\vec{N}_1$.
6. Calculate the exponential of each eigenvalue $N_{3,i} = e^{D_{ii}}N_{2,i}$.
7. Compute the triangular matrix-vector product $\vec{N}_4 = \mathbf{V}\vec{N}_3$.
8. Compute the matrix-vector product $\vec{N}(t_1) = \mathbf{Q}\vec{N}_4$.

Loan, 1989:380). The accuracy of the two algorithms is therefore mainly determined by the roundoff errors that are introduced in solving the triangular system as was explained in Section 4.1.

4.2.2 Block storage scheme

The block structure of the coefficient matrix can also be exploited when computing the Schur decomposition as shown in Algorithm 11. If the coefficient matrix has the form as in (2.39), each diagonal block $\mathbf{D}_i$ can be transformed to a triangular matrix $\mathbf{T}_i = \mathbf{Q}_i^T\mathbf{D}_i\mathbf{Q}_i$. Because the Schur decomposition needs $\mathcal{O}(n^3)$ flops, it will be much faster to transform all the blocks separately than transforming the whole n-by-n coefficient matrix. The blocks containing the fission products, $\mathbf{F}_{ij}$, can then be transformed by $\mathbf{T}_{ij} = \mathbf{Q}_j^T\mathbf{F}_{ij}\mathbf{Q}_i$, giving the triangular matrix

$$\mathbf{T} = \left[\begin{array}{ccc|ccc} \mathbf{Q}_1^T\mathbf{D}_1\mathbf{Q}_1 & & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ & \ddots & & \vdots & \ddots & \vdots \\ \mathbf{0} & & \mathbf{Q}_d^T\mathbf{D}_d\mathbf{Q}_d & \mathbf{0} & \cdots & \mathbf{0} \\ \hline \mathbf{Q}_{d+1}^T\mathbf{F}_{d+1,1}\mathbf{Q}_1 & \cdots & \mathbf{Q}_{d+1}^T\mathbf{F}_{d+1,d}\mathbf{Q}_d & \mathbf{Q}_{d+1}^T\mathbf{D}_{d+1}\mathbf{Q}_{d+1} & & \mathbf{0} \\ \vdots & \ddots & \vdots & & \ddots & \\ \mathbf{Q}_n^T\mathbf{F}_{n,1}\mathbf{Q}_1 & \cdots & \mathbf{Q}_n^T\mathbf{F}_{n,d}\mathbf{Q}_d & \mathbf{0} & & \mathbf{Q}_n^T\mathbf{D}_n\mathbf{Q}_n \end{array}\right]. \tag{4.19}$$

When the block storage scheme is used to implement Algorithm 12, step 3 of Algorithm 11 must be included.

4.3 Preconditioning methods

The following question is asked in this section: If we know the solution of a similar system of depletion equations, is it then possible to use this knowledge to solve the fuel depletion equations

more efficiently? For example, can the triangular analytical methods be used to solve the lower triangle of the system of equations and then some correction technique used for the feedback terms above the diagonal? Because the fuel depletion equations are coupled to the neutron transport equations, the depletion module is called many times with only slight changes in the neutron flux and therefore only small changes in the coefficient matrix. This will happen, for example, when the reactor is operated at constant power, as shown in Subsection 2.3.1. Can the matrix exponential of the coefficient matrix then be calculated only the first time, and the next time only some correction technique be used?

In order to motivate the discussion, two very inefficient transformations will first be introduced. Then the preconditioning method of Lawson (1967) is discussed.

Consider the coefficient matrix $\mathbf{A}$ of the system of depletion equations (1.1). Suppose there is a similar matrix $\mathbf{B}$, such that $\mathbf{E} = \mathbf{A} - \mathbf{B}$ has a much smaller norm and much smaller eigenvalues than $\mathbf{A}$, and that $\mathbf{F} = e^{t\mathbf{B}}$ is already computed.

Since the two systems with coefficient matrices $\mathbf{A}$ and $\mathbf{B}$ are similar, it is tempting to construct a variable $\vec{z}(t) = \vec{N}(t) - e^{t\mathbf{B}}\vec{N}_0$, which would not change much over time. The system of depletion equations then becomes

$$\begin{aligned} \frac{d\vec{z}(t)}{dt} &= \mathbf{A}\vec{z}(t) + (\mathbf{A} - \mathbf{B})e^{t\mathbf{B}}\vec{N}_0 \\ \vec{z}_0 &= \vec{0}. \end{aligned}$$

Using the Taylor expansion to calculate $\vec{z}$ it is found that

$$\begin{aligned} \vec{z}(t) &= \left(\frac{t\mathbf{A} - t\mathbf{B}}{1!} + \ldots + \frac{(t\mathbf{A})^i - (t\mathbf{B})^i}{i!} + \ldots \right) \vec{N}_0 \\ &= \left(e^{t\mathbf{A}} - e^{t\mathbf{B}}\right) \vec{N}_0. \end{aligned}$$

Therefore, to calculate $\vec{z}$ it is necessary to calculate both $e^{t\mathbf{A}}$ and $e^{t\mathbf{B}}$. The number of terms required will not necessarily be smaller than those required in (3.3). This can be seen by noting that, for any eigenvalue λ,

$$(\lambda(1+\varepsilon))^n - \lambda^n \simeq n\varepsilon\lambda^n,$$

when the change in eigenvalue $\varepsilon \ll 1$, and there will only be a reduction in the number of terms if $\varepsilon < 1/\lambda_{maks}$. Since double the amount of work is necessary per term, it make this method more expensive than a normal Taylor expansion to calculate $e^{\mathbf{A}t}$.

Another possibility is to try to calculate $e^{t(\mathbf{B}+\mathbf{E})}$ using the Taylor expansion:

$$\begin{aligned} e^{\mathbf{A}} &= e^{(\mathbf{B}+\mathbf{E})} \\ &= \mathbf{I} + \mathbf{A} + \frac{\mathbf{B}^2 + \mathbf{BE} + \mathbf{EB} + \mathbf{E}^2}{2!} \\ &\quad + \frac{\mathbf{B}^3 + \mathbf{B}^2\mathbf{E} + \mathbf{BEB} + \mathbf{EB}^2 + \mathbf{E}^2\mathbf{B} + \mathbf{EBE} + \mathbf{BE}^2 + \mathbf{E}^3}{3!} + \ldots \\ &= e^{\mathbf{B}} + \mathbf{E}\left(\mathbf{I} + \frac{\mathbf{B}}{2!} + \frac{\mathbf{B}^2}{3!} + \ldots\right) + \left(\frac{\mathbf{B}}{2!} + \frac{\mathbf{B}^2}{3!} + \ldots\right)\mathbf{E} + \ldots \end{aligned}$$

Algorithm 13 Fourth order general Runge-Kutta algorithm (Castillion & Saad, 1997:9)

1. Calculate $\mathbf{F} = e^{-\frac{t}{2}\mathbf{B}}$
2. $\vec{Y}_1 = \vec{N}_0$
3. $\vec{w} = \mathbf{F}\vec{Y}_1$
4. $\vec{q} = \mathbf{F}t\mathbf{E}\vec{Y}_1$
5. $\vec{Y}_2 = \vec{w} + \frac{1}{2}\vec{q}$
6. $\vec{Y}_3 = \vec{w} + \frac{1}{2}t\mathbf{E}\vec{Y}_2$
7. $\vec{Y}_4 = \mathbf{F}(\vec{w} + t\mathbf{E}\vec{Y}_3)$
8. $\vec{N}(t) = \mathbf{F}\left(\vec{w} + \frac{1}{6}\vec{q} + \frac{1}{3}t\mathbf{E}(\vec{Y}_2 + \vec{Y}_3)\right) + \frac{1}{6}t\mathbf{E}\vec{Y}_4$

Because there is no efficient way to calculate this expansion, it is impractical to use this possibility.

A third possibility is to consider the function $\vec{z}(t) = e^{-t\mathbf{B}}\vec{N}(t)$, as was suggested by Lawson (1967:372), giving the nonlinear initial value problem

$$\begin{aligned} \frac{d\vec{z}(t)}{dt} &= e^{-t\mathbf{B}}\mathbf{E}e^{t\mathbf{B}}\vec{z}(t) \\ \vec{z}(0) &= \vec{N}_0. \end{aligned} \tag{4.20}$$

The eigenvalues of $e^{-t\mathbf{B}}\mathbf{E}e^{t\mathbf{B}}$ are those of $\mathbf{E}$ (Lawson, 1967:373). When the matrix $\mathbf{B}$ is very similar to $\mathbf{A}$, $\mathbf{E}$ will have small eigenvalues, and the resulting system will be non-stiff even if the system of fuel depletion equations is stiff. To solve this non-linear system, a whole range of ODE methods can be used. Castillion and Saad (1997) show how the fourth order general Runge-Kutta method can be used to solve (4.20):

$$\begin{aligned} \vec{Y}_1 &= \vec{N}_0 \\ \vec{Y}_2 &= e^{-\frac{t}{2}\mathbf{B}}\vec{N}_0 + \frac{t}{2}e^{-\frac{t}{2}\mathbf{B}}\mathbf{E}\vec{Y}_1 \\ \vec{Y}_3 &= e^{-\frac{t}{2}\mathbf{B}}\vec{N}_0 + \frac{t}{2}\mathbf{E}\vec{Y}_2 \\ \vec{Y}_4 &= e^{-t\mathbf{B}}\vec{N}_0 + te^{-\frac{t}{2}\mathbf{B}}\mathbf{E}\vec{Y}_3 \\ \vec{N}_1 &= e^{-t\mathbf{B}}\vec{N}_0 + \frac{t}{6}\left(e^{-t\mathbf{B}}\mathbf{E}\vec{Y}_1 + 2e^{-\frac{t}{2}\mathbf{B}}\mathbf{E}\vec{Y}_2 + 2e^{-\frac{t}{2}\mathbf{B}}\mathbf{E}\vec{Y}_3 + \mathbf{E}\vec{Y}_4\right). \end{aligned} \tag{4.21}$$

An optimal algorithm for calculating (4.21) is given in Algorithm 13. This algorithm requires 9 matrix-vector multiplications per time step. This is almost double the amount that is necessary in a fourth order Taylor expansion, but the system is much less stiff, making it possible to use larger integration time steps.

Algorithm 14 shows how Algorithm 13 can be combined with the Parlett method to solve the depletion equations. The Parlett method is used to solve the lower triangle of the coefficient matrix and a number of correction steps are done for the feedback terms above the diagonal by repeating the Runge-Kutta algorithm 13.

Algorithm 14 Parlett and fourth order general Runge-Kutta algorithm

1. Let $\mathbf{B}$ be the lower triangle of the coefficient matrix $\mathbf{A}$, stored using the block storage scheme and $\mathbf{E}$ be the feedback terms of $\mathbf{A}$, stored using index notations.
2. Determine the amount of correction steps s needed.
3. Scale $\mathbf{E}$ with t/s: $\mathbf{E} = \frac{t}{s}\mathbf{E}$
4. Calculate $\mathbf{F} = e^{\frac{t}{2s}\mathbf{B}}$using the recursive Parlett algorithm, Algorithm 9.
5. Repeat steps 6 to 11 s times, starting with $\vec{N} = \vec{N}_0$, and ending with $\vec{N}(t) = \vec{N}$.
6. $\vec{w} = \mathbf{F}\vec{N}$
7. $\vec{q} = \mathbf{F}\mathbf{E}\vec{N}$
8. $\vec{Y}_2 = \vec{w} + \frac{1}{2}\vec{q}$
9. $\vec{Y}_3 = \vec{w} + \frac{1}{2}\mathbf{E}\vec{Y}_2$
10. $\vec{Y}_4 = \mathbf{F}(\vec{w} + \mathbf{E}\vec{Y}_3)$
11. $\vec{N} = \mathbf{F}\left(\vec{w} + \frac{1}{6}\vec{q} + \frac{1}{3}\mathbf{E}(\vec{Y}_2 + \vec{Y}_3)\right) + \frac{1}{6}\mathbf{E}\vec{Y}_4$

Algorithm 15 Fourth order general Runge-Kutta algorithm using a previous solution

1. The first time the depletion module is called, calculate $\mathbf{F} = e^{\frac{t}{s}\mathbf{A}}$ using Algorithm 3, store both $\mathbf{F}$ and $\mathbf{A}_1 = \mathbf{A}$, calculate the final isotope concentration $\vec{N}(t) = \mathbf{F}^s\vec{N}$ and exit.
2. Calculate $\mathbf{E} = \frac{1}{s}(\mathbf{A} - \mathbf{A}_1)$, with $\mathbf{A}$ the coefficient matrix and $\mathbf{A}_1$ the previous coefficient matrix used in step 1.
3. If $\mathbf{E}$ is too large, do step 1.
4. Repeat steps 5 to 10 s times, starting with $\vec{N} = \vec{N}_0$, and ending with $\vec{N}(t) = \vec{N}$.
5. $\vec{w} = \mathbf{F}\vec{N}$
6. $\vec{q} = \mathbf{F}\mathbf{E}\vec{N}$
7. $\vec{Y}_2 = \vec{w} + \frac{1}{2}\vec{q}$
8. $\vec{Y}_3 = \vec{w} + \frac{1}{2}\mathbf{E}\vec{Y}_2$
9. $\vec{Y}_4 = \mathbf{F}(\vec{w} + \mathbf{E}\vec{Y}_3)$
10. $\vec{N} = \mathbf{F}\left(\vec{w} + \frac{1}{6}\vec{q} + \frac{1}{3}\mathbf{E}(\vec{Y}_2 + \vec{Y}_3)\right) + \frac{1}{6}\mathbf{E}\vec{Y}_4$

When the depletion module is called many times with only small variations in the neutron flux, the Runge-Kutta method can be combined with the Padé approximation as shown in Algorithm 15. The first time the depletion module is called, the Padé approximation is used and the resulting matrix exponential is stored. When the depletion module is called again, the changes in the coefficient matrices are determined and if they are small, Algorithm 13 is used.

4.4 Summary

The matrix exponential of a triangular coefficient matrix can be determined by computing the eigenvalues and eigenvectors, which is done optimally in the GAUGE code, Algorithm 8, or by using the Parlett recursive function evaluation, Algorithm 9.

When the coefficient matrix is non-triangular, it can be transformed to a triangular matrix by using QR iterations to compute the Schur form of the matrix. Algorithms (11) and (12) show how this can be implemented using EISPACK and LAPACK routines. When eigenvalues are nearly confluent, large roundoff errors can be introduced in the analytical methods. If the eigenvalues are known, these roundoff errors can be estimated using (4.5) and (4.6).

When the solution of a similar fuel depletion problem is known, this solution can be used to precondition the fuel depletion equation, giving a non-stiff nonlinear system of equations. Algorithm (13) then shows how this system can be solved efficiently using the fourth order Runge-Kutta method.

In the next chapter the algorithms described in Chapters 3 and 4 will be compared with each other to determine which algorithm and which method will perform the best in the depletion module under various conditions.

Chapter 5

Comparison of Selected Methods

In this chapter the different methods, given in Chapters 3 and 4, to solve the fuel depletion equation are compared with each other. In Section 5.1 it is shown how these methods were implemented and in Sections 5.2 to 5.4 the performance of the different implementations and different algorithms of the same method are compared with each other in order to find the fastest implementation for each method. The fastest implementation for each method is then compared with each other in Section 5.5. In Section 5.6 the methods are evaluated using the criteria mentioned in Section 1.6. In Section 5.7 it is summarised which method is the best to be used in the depletion module under various conditions.

5.1 Implementation of algorithms

In Chapters 3 and 4 eight methods were described to solve the system of fuel depletion equations. In Chapter 3 four polynomial methods where described: The Taylor, Padé, Krylov and Chebyshev approximation methods. In Chapter 4 the Gauge and Parlett triangular methods, the Schur decomposition method and the Runge-Kutta preconditioning method were introduced. For each of these eight methods one or more algorithms were given.

Table 5.1 gives a short description of the algorithms that were given in Chapters 3 and 4. Each of these algorithms was implemented using one of the three storage schemes of Subsection 2.3.2, the full matrix storage scheme, the block matrix storage scheme or the index storage scheme. Table 5.2 shows the thirteen algorithms and which storage scheme was used for each algorithm. In order to compare the different storage schemes, some algorithms were implemented more than once using different schemes.

The EXPOKIT (Sidje, 1998) FORTRAN 77 code was used for both Padé algorithms, the Krylov algorithm and the partial fraction Chebyshev algorithm. Because this code uses full matrix calculations, all the matrix calculations in the code were changed to support the block matrix scheme or index matrix scheme. LAPACK and EISPACK routines were used to do the Schur decomposition in Algorithms 11 and 12. All the other algorithms were coded by the author in FORTRAN 90. All

Table 5.1: Description of algorithms given in Chapters 3 and 4

Algorithm	Method	Description	Reference
1 on page 34	Taylor 1	Taylor expansion for each integration time step.	ORIGEN-S (Hermann & Westfall, 1998)
2 on page 36	Taylor 2	Single step Taylor expansion with scaled diagonal (uniformization technique).	This dissertation. Generalisation of Sidje and Steward (1999)
3 on page 37	Padé 1	Padé method using Horner's algorithm with scaling and squaring.	Arioli *et. al.* (1996:114), Moler & Van Loan (1978:808)
4 on page 39	Padé 2	Padé method using Horner's algorithm with scaling, squaring and matrix-vector multiplications.	This dissertation
5 on page 40	Krylov	Krylov method using the Arnoldi process and the Padé approximation.	Sidje (1998:135), Saad (1992:210), Gallopoulos and Saad (1992:1238)
6 on page 42	Chebyshev 1	Chebyshev method using Horner's algorithm.	This dissertation
7 on page 43	Chebyshev 2	Chebyshev method using partial fractions.	Sidje (1998:135)
8 on page 47	GAUGE	Solving a triangular system of equations using eigenvalues and eigenvectors.	Wagner (1968:33)
9 on page 48	Parlett	Solving a triangular system of equations using Parlett's recursive formula.	Parlett (1976)
11 on page 50	Schur decomposition 1	Triangulate system using its Schur decomposition and solve using Algorithm 8.	This dissertation. Modification of Golub and Van Loan (1989:380)
12 on page 51	Schur decomposition 2	Triangulate system using its Schur decomposition and solve using eigenvectors.	Golub and Van Loan (1989:380)
14 on page 54	Parlett & Runge-Kutta	Solve lower triangle of system with Algorithm 9 and use Runge-Kutta method for feedback terms.	This dissertation. Modification of Castillo and Saad (1997:9)
15 on page 54	Runge-Kutta	Use a previous solution, calculated by Algorithm 3, and the Runge-Kutta method to make corrections for the difference.	This dissertation. Modification of Castillo and Saad (1997:9)

Table 5.2: Implementations of each algorithm in FORTRAN

Method	Algorithm	Matrix Storage Scheme	Language	Reference
Taylor 1	1 on page 34	Full, Block, Index	FORTRAN 90	This dissertation
Taylor 2	2 on page 36	Index	FORTRAN 90	This dissertation
Padé 1	3 on page 37	Full	FORTRAN 77	EXPOKIT
Padé 2	4 on page 39	Full, Block	FORTRAN 77	EXPOKIT
Krylov	5 on page 40	Index	FORTRAN 77	EXPOKIT
Chebyshev 1	6 on page 42	Full	FORTRAN 90	This dissertation
Chebyshev 2	7 on page 43	Full, Block	FORTRAN 77	EXPOKIT
GAUGE	8 on page 47	Full, Block	FORTRAN 90	GAUGE
Parlett	9 on page 48	Full, Block	FORTRAN 90	This dissertation
Schur decomposition 1	11 on page 50	Full, Block	FORTRAN 90	LAPACK routines
Schur decomposition 2	12 on page 51	Full	FORTRAN 90	LAPACK routines
Parlett & Runge-Kutta	14 on page 54	Block & Index	FORTRAN90	This dissertation
Runge-Kutta	15 on page 54	Index	FORTRAN 90	This dissertation

the implementations use double precision floating point computations, with only the partial fraction Chebyshev algorithm using complex numbers.

For the comparison of the various methods, a tolerance of 10^{-10} was specified. This required a $(6, 6)$ Padé and a $(14, 14)$ Chebyshev approximation. For the Krylov subspace approximation, a subspace of dimension 30 was used as suggested by Sidje (1998). The efficiency of the codes was measured by comparing the computer time needed to solve a system of depletion equations on an Intel Pentium III 1 GHz processor with 128MB RAM when compiled and optimised by Fortran Power Station 4.0 of Microsoft Developer Studio.

5.2 General methods

In order to compare the different implementations of the same method, the computer time needed by the implementations to solve some sample depletion problems was measured and compared.

The Taylor, Padé, Krylov and Chebyshev approximation methods, together with the Schur decomposition analytical method can be used to solve any depletion problem. In this section the different implementations of these methods are compared with each other using the sample problems introduced in Subsection 5.2.1. Subsection 5.2.2 gives the resulting computer time needed by each algorithm. These results are discussed in Subsection 5.2.3 to 5.2.6.

Because different sample problems were used to compare the implementations of the Runge-Kutta method and the Gauge and Parlett methods, these comparisons are discussed separately in Sections 5.3 and 5.4.

5.2.1 Test problems

To test the performance of the various implementations in solving the fuel depletion equations under different conditions, a small system with 9 isotopes and a large system with 126 isotopes were considered. These two systems are examples of how the system of fuel depletion equations will typically look like in a global reactor calculation and in an assembly calculation.

5.2.1.1 Core calculation

As explained in Subsection 1.3.1 only the burnup of the fuel and the most important fission products are included in the depletion equations during a global reactor calculation. In the core sample problem the actinides U^{235}, U^{236}, U^{238}, Pu^{239}, Pu^{240}, Pu^{241} and Pu^{242}, and the two fission products I^{135} and Xe^{135} were included. In Table B.1 the initial isotope number densities are listed. The fission cross sections and the fission yields of the two fission products for the 7 fissile isotopes are listed in Table B.2. The decay constants of Pu^{241}, I^{135} and Xe^{135} are given in Table B.3 and the neutron capture cross sections and the corresponding daughter products are given in Table B.4. A neutron flux of $\Phi = 3.892\text{x}10^{14}$ neutrons.s^{-1}.cm^{-2} was assumed to be present. Using these nuclear data, the coefficient matrix $\mathbf{A}$ can be constructed as explained in Section 2.2. Expression (2.38) on page 20 shows the resulting coefficient matrix, in units of s^{-1}.

Some properties of coefficient matrix (2.38) are that the Lapidus and Luus norm (2.49) and the 1-norm (2.42) are given by the maximum diagonal element

$$D_{max} = [\mathbf{A}] = \|\mathbf{A}\|_1 = 6.72\text{x}10^{-5}s^{-1}, \tag{5.1}$$

and that the matrix infinity norm (2.44) is

$$\|\mathbf{A}\|_\infty = 9.63\text{x}10^{-5}s^{-1}. \tag{5.2}$$

The stiffness of the core sample problem is then given by (2.54). For example, if the burnup time step over which the system must be solved is 10 days, then the stiffness given by (2.54) is 58.1, which is of moderate stiffness.

When using the index storage scheme, there are 32 non-zero elements as shown in Figures 2.2 to 2.4, and when using the block storage scheme, shown in Figure 2.5, there are 3 fissile isotope blocks and one fission product block.

5.2.1.2 Assembly calculation

To test the performance of the methods in solving the fuel depletion equations during an assembly calculation, a sample system of depletion equations was used. This system contains 126 isotopes of which 22 are actinides, as listed in Table B.5. This was taken out of an assembly calculation, as explained in Subsection 1.3.2, for a reactor core that had already burned for some time with a

neutron flux of $3.69\text{x}10^{13}$ neutrons.s^{-1}.cm^{-2}. All the fast decaying isotopes are therefore already in semi-equilibrium with their parent isotopes. The maximum diagonal element of the coefficient matrix is

$$D_{max} = 4.338\text{x}10^{-5}s^{-1}, \tag{5.3}$$

giving a stiffness (2.54) of greater than 37.5 for a burnup time step of 10 days.

The coefficient matrix has 1751 non-zero terms, which is much less than the total number of terms in the matrix, $126^2 = 15876$, resulting in a very sparse coefficient matrix. When using the block storage scheme, (2.39), the actinides are all in one group and the fission products can be divided into 8 independent decay chains and 15 single isotopes. These 15 isotopes form 15 blocks of size one. The 126 isotopes, as divided into these 24 groups, are listed in Table B.5. The blocks contain 1751 non-zero and 257 zero elements, which is a large reduction in the number of zero elements that have to be stored.

5.2.2 Results

In solving the two sample problems, a number of different burnup times, ranging from 2.4 hours (0.1 days) to 2.8 years (1000 days), were considered. Because the stiffness of the system is directly proportional to the burnup time step, (2.54), the effect of stiffness on a particular method can be seen by looking at the effect of the burnup time.

For each burnup time, a reference solution was calculated using the Taylor expansion with small integration time steps. The integration step size was chosen such that halving the step size would have no influence on the result within the tolerance of 10^{-12}. In the core problem the analytical methods gave results that differed less than 10^{-11} for each isotope, which is smaller than the specified tolerance of 10^{-10}. In the assembly test problem, the analytical methods also gave results that differed less than 10^{-11} for most of the isotopes. This confirms that the reference solution is indeed correct. The reference solutions of the core sample problem when burned for 100 and 1000 days are given in Table B.1.

To compare the accuracy of the various implementations, the maximum relative difference between the final isotope number densities and the reference isotope number densities was calculated. Isotopes having a number density smaller than 10^{-9} times the largest number density were neglected.

The computer time needed to solve the two sample depletion problems and the maximum relative difference in the final isotope number densities, for the Taylor, Padé, Krylov, Chebyshev and Schur implementations, are given in Tables B.6 and B.7. These results will be discussed in the remainder of this section.

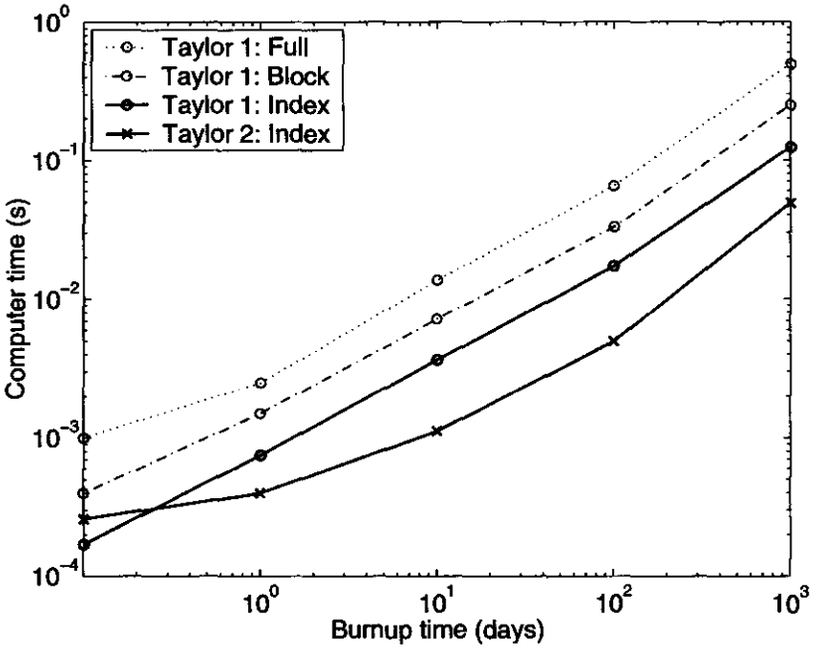

Figure 5.1: Computer time used by the Taylor implementations to solve the assembly depletion problem

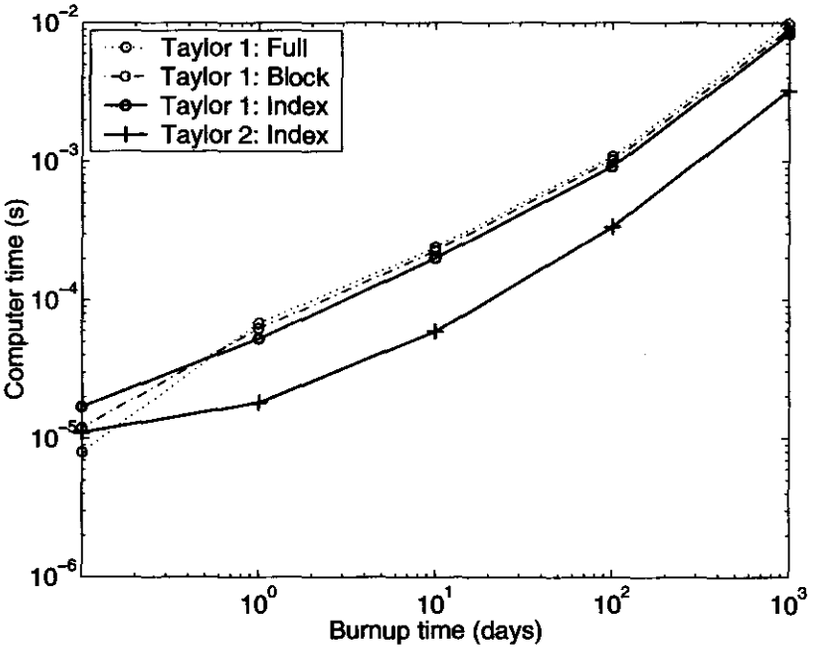

Figure 5.2: Computer time used by the Taylor implementations to solve the core depletion problem

5.2.3 Taylor implementations

Fastest implementation

The two Taylor algorithms, Taylor 1 and 2, as described in Section 3.2 and implemented as described in Section 5.1 were used to solve the two sample depletion problems described in Subsection 5.2.1. Figures 5.1 and 5.2 show the computer time, taken from Tables B.6 and B.7 respectively, needed for solving the two sample depletion problems for different burnup time lengths.

The first Taylor algorithm was implemented three times, using full matrix computations, index matrix storage and block matrix storage as explained in Subsection 2.3.2. Comparing the speed of the three implementations in solving the large assembly problem, shown in Figure 5.1, it is found that the index storage scheme is about four times faster than the full matrix scheme, with the block matrix scheme lying between the other two. For large systems, which typically have a very sparse coefficient matrix, using index storage will therefore be the fastest.

In the smaller core problem the three implementations of the first Taylor algorithm use about the same amount of computer time as can be seen in Figure 5.2. The reason for this is that the coefficient matrix in the core sample problem is not very sparse and that the time gained by doing less zero computations in the block and index schemes is lost with the look up that has to be done.

In order to compare the standard Taylor algorithm 1 as used in the ORIGEN-S code against the second Taylor algorithm using the uniformization technique, both algorithms were implemented using the index storage scheme. As shown in Figures 5.1 and 5.2, the uniformization technique is about two to three times faster in solving both sample problems for burnup times longer than one day. For small burnup time steps, when the system of fuel depletion equations is non-stiff, both algorithms will need only one integration time step with only a few terms in the Taylor expansion. Because the uniformization method has to scale the diagonal of the coefficient matrix and the initial number densities, which is not needed in the normal Taylor algorithm, it is a bit slower than the normal Taylor algorithm at very small burnup time steps.

The fastest implementation for the Taylor method will therefore be the second Taylor algorithm implemented using index matrix storage.

Effect of stiffness on speed

The size of the integration time steps in the first Taylor algorithm depends on the norm of the coefficient matrix, as explained in Subsection 3.2.4, which is independent of the total burnup time. The number of integration time steps is therefore directly proportional to the burnup time. If the same number of terms in the Taylor expansion would have been used for each integration time step, the computer time would also be directly proportional to the burnup time. But, as can be seen from Figures 5.1 and 5.2, the first few integration time steps need more computer time, meaning that more terms are necessary to obtain the required tolerance. This is because all the short-lifetime isotopes are not fully in equilibrium with their parent isotopes at the beginning of the burnup time.

As the short-lifetime isotopes reach equilibrium, the number of terms in each expansion becomes constant and the computer time becomes almost proportional to the burnup time.

Although only one integration time step is used in the uniformization Taylor algorithm, it seems to have the same behaviour as the first Taylor algorithm for large burnup times, i.e. the computer time becomes almost proportional to the burnup time. This is suggested by the linear dependence of both the upper limit, (3.29), and the lower limit, (3.32), of number of terms in the expansion on the burnup step size.

Because the stiffness of the depletion problem is directly proportional to the burnup time, (2.54), both algorithms of the Taylor method have the drawback that the computer time depends strongly on the stiffness, making the Taylor algorithms slow for very stiff problems.

5.2.4 Padé implementations

Fastest implementation

As shown in Table 5.2, Padé algorithm 1 was implemented using full matrix calculations, and Padé algorithm 2 was implemented twice, using full matrix and block matrix calculations. The computer time needed by these three implementations to solve the two sample depletion problems for different burnup times is shown in Figures 5.3 and 5.4.

The only difference in the two Padé algorithms is that some of the matrix squaring in the first algorithm is replaced by matrix-vector products in the second algorithm. For non-stiff problems, for example the assembly problem with a burnup time of 0.1 days, no scaling is necessary and the two methods are equivalent. In solving stiff problems, the second algorithm is slightly faster than the first. This can be seen from the blue and red solid lines in Figures 5.3 and 5.4, where both methods were implemented using full matrix computations.

The second Padé algorithm was also implemented using block matrix storage. As can be seen in Figures 5.3 and 5.4 the block storage scheme (dotted red line) is about 3.5 times faster in solving the assembly problem and about 1.5 times slower in solving the core sample problem, than the full matrix implementation (solid red line). Because the core sample problem is small, containing only 9 isotopes, and not very sparse, it is faster to do one matrix calculation on the full matrix than doing matrix calculations with the individual blocks.

The fastest implementation for the Padé method will therefore be the second Padé algorithm using full matrix calculations for small problems and using block matrix calculations for large problems.

Effect of stiffness on speed

Because of the scaling and squaring used in the Padé algorithms, one extra matrix-matrix product is necessary each time the burnup time doubles. The computer time therefore increases logarithmically with the burnup time, as can be seen in Figures 5.3 and 5.4, for burnup times longer than one day. Both Padé algorithms are therefore not very sensitive to the stiffness of the system.

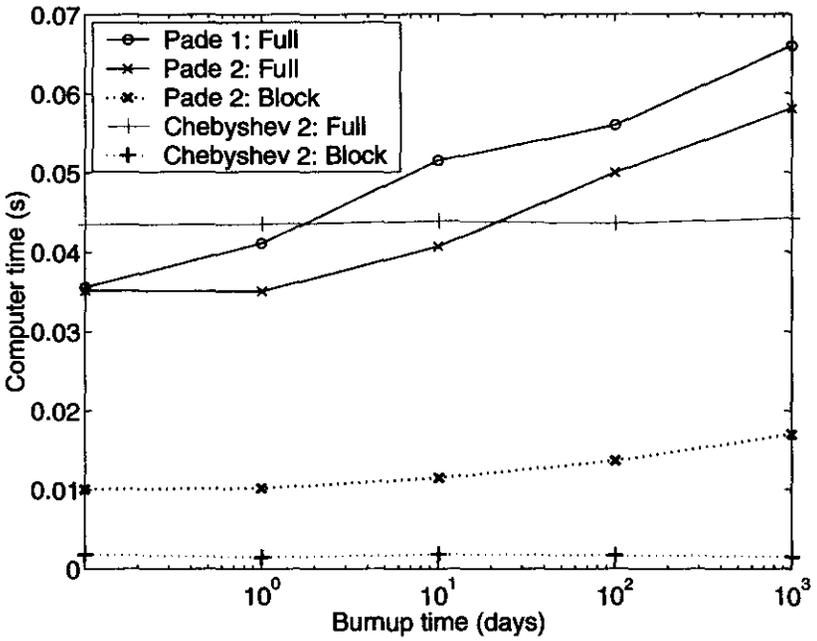

Figure 5.3: Computer time used by the Padé and Chebyshev implementations to solve the assembly depletion problem

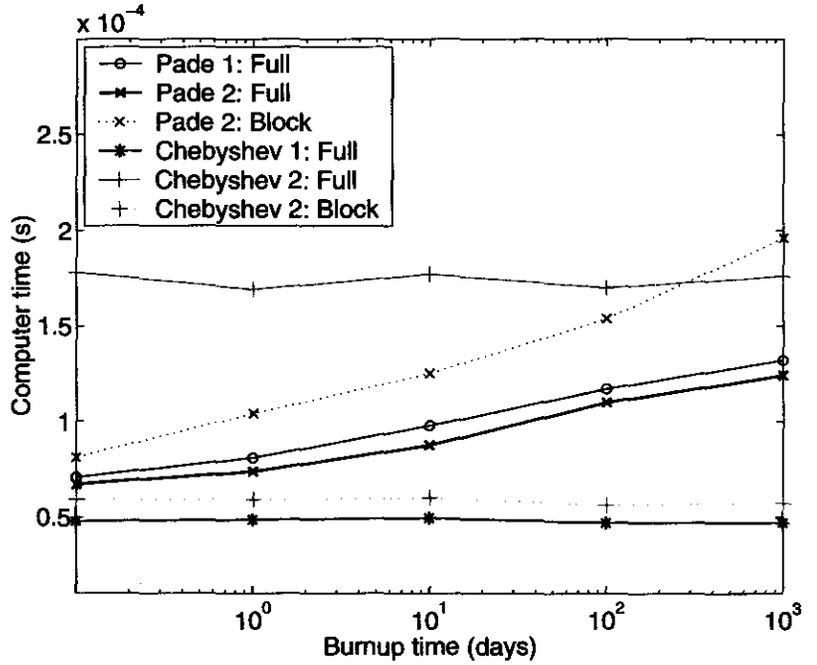

Figure 5.4: Computer time used by the Padé and Chebyshev implementations to solve the core depletion problem

5.2.5 Chebyshev implementations

Fastest implementation

The two Chebyshev algorithms, Algorithms 6 on page 42 and 7 on page 43, use no scaling or integration time steps and the computer time used is therefore independent of the burnup time. Unlike the Padé implementations, where the full matrix implementation is faster than the block matrix implementation in solving the core problem, the block implementation (dotted green line) of the partial fraction Chebyshev algorithm, Chebyshev 2, is three times faster than the full matrix implementation (solid green line), as shown in Figure 5.4. This is because the partial fraction algorithm uses complex number operations which each need four flops, making it four times more expensive to do calculations.

The first Chebyshev algorithm, which uses real matrix computations and a matrix inversion, has the same characteristics as the Padé algorithms for different matrix storage schemes. Because full matrix computations are much slower than matrix-vector calculations for large matrices, the first Chebyshev algorithm will only be competitive in solving small systems and, like the Padé algorithms, the full matrix implementation will then be faster than the block matrix implementation.

Comparing the speed of the first Chebyshev algorithm using full matrix computations (solid black line) with the second Chebyshev algorithm using block matrix computations (solid green line), as shown in Figure 5.4, it is found that the first method is only slightly faster. For larger problems, the second partial fraction algorithm will be a lot faster than the first algorithm and it is therefore generally the best to use the second Chebyshev algorithm, implemented with block matrix computations.

Effect of stiffness on accuracy

As mentioned above, the burnup time, and therefore the stiffness of the depletion problem, does not influence the speed of the Chebyshev algorithms, because only one time step is used, but the accuracy is influenced by the stiffness.

The (14,14) Chebyshev method is slightly less accurate than the (6,6) Padé method. For the very long burnup time of 1000 days there is a loss of accuracy from 10^{-11} to 10^{-7} in the assembly example and from 10^{-13} to 10^{-11} in the core example, as given in Tables B.6 and B.7. The reason for this loss of accuracy is that the Chebyshev approximation to e^{-x} is less accurate for large x.

5.2.6 Non-triangular analytical implementations

Fastest implementation

Table 5.3 shows the computer time needed by the two Schur decomposition algorithms to solve the assembly and core problems. The first Schur decomposition algorithm, Algorithm 11 on page 50,

Table 5.3: Computer time used by the Schur decomposition implementations to solve the assembly and core depletion problems

Algorithm	Storage Scheme	Assembly calculation Computer time (s $\times 10^{-3}$)	Core calculation Computer time (s $\times 10^{-6}$)
Schur 1	Full	12.6	79.5
	Block	4.0	50.7
Schur 2	Full	7.4	47.3

was implemented twice, using full matrix and block matrix computations, and the second Schur decomposition algorithm, Algorithm 12 on page 51, was implemented once, using full matrix computations.

As with the triangular analytical methods, which will be discussed in Section 5.4, the block implementation of the first Schur method is faster than the full matrix implementation for both sample problems. The second Schur decomposition algorithm uses about 40% less computer time than the first algorithm when both are implemented using full matrix calculations, but is slower in the assembly calculation than the first algorithm when the first algorithm is implemented using block matrix calculations. The fastest implementation for the Schur decomposition method during an assembly calculation is the first algorithm using block matrix calculations, and during a core calculation the second algorithm using full matrix calculations.

5.3 Preconditioning methods

The preconditioning methods can only be used if the preconditioning matrix is close to the coefficient matrix. These methods are therefore less general than the methods discussed in the previous section, and the sample problems of the previous section can therefore not directly be used to compare the implementations of these methods. In Subsection 5.3.1 it is shown how the two sample problems of Subsection 5.2.1 can be slightly modified, so that it can be used in Subsections 5.3.2 and 5.3.3 to compare the Runge-Kutta implementations.

5.3.1 Test problems

Because the order of the Runge-Kutta algorithm, Algorithm 15, is fixed to 4 and can not easily be varied as with the Taylor algorithms, very small time steps will be necessary at the beginning of the burnup time step when the initial isotope concentrations are not in equilibrium with their precursors. Because Algorithm 15 does not have step-size control, the core sample problem of Subsection 5.2.1 had to be burned for 10 days, using a 1% and 0.1% smaller neutron flux, to ensure that all the fast transients have died out. The final isotope concentrations which are obtained, for the 1% and 0.1% smaller neutron flux, are then used as initial isotope concentrations, giving two new core problems. These two core problems have the same coefficient matrix as the initial core

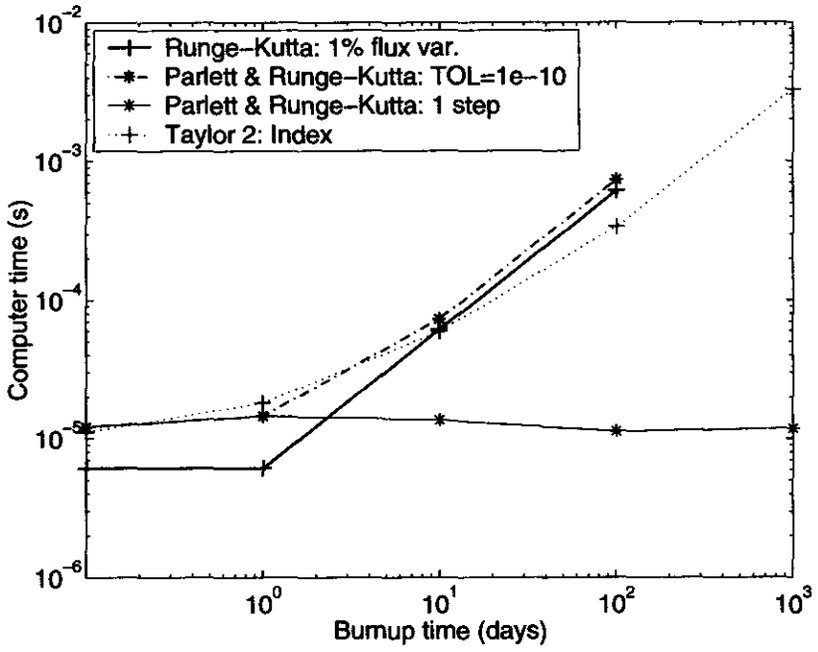

Figure 5.5: Computer time used by the Runge-Kutta implementations to solve the core depletion problem

problem and only slightly different initial isotope concentrations, because the first initial isotope concentrations were already close to equilibrium. Using the new initial isotope concentrations, the reference solutions were calculated again as explained in Subsection 5.2.2.

5.3.2 Runge-Kutta implementation using previous solution

The matrix exponential of the preconditioning coefficient matrices, having a 1% and 0.1% smaller neutron flux, was calculated using the Padé method, and used to precondition the coefficient matrix when solving the two core problems with the Runge-Kutta algorithm, Algorithm 15 on page 54.

Because the Runge-Kutta method uses vector-matrix multiplications, which were also used in the Taylor algorithms, the index matrix storage implementation was used for the Runge-Kutta algorithm, because it was found to be the fastest implementation for the Taylor algorithms.

Accuracy

The solid blue line in Figure 5.5 shows the computer time needed by the Runge-Kutta implementation to solve the depletion problem when the flux was changed by 1% and the tolerance was set at 10^{-8}. If the computer time is fixed, the accuracy of the Runge-Kutta algorithm depends on the change in the neutron flux, because the largest eigenvalue of the preconditioned system depends on the neutron flux. As can be seen in Table B.7, when the neutron flux changes by 0.1% the same computer time will give an accuracy of 10^{-9}.

Table 5.4: The effect of one fourth order Runge-Kutta step for the feedback terms in the core problem

Burnup time (days)	Accuracy when neglecting feedback terms	Accuracy with one Runge-Kutta step
0.1	$1.9\text{x}10^{-9}$	$8.9\text{x}10^{-12}$
1	$1.9\text{x}10^{-8}$	$1.5\text{x}10^{-10}$
10	$1.9\text{x}10^{-7}$	$1.4\text{x}10^{-8}$
100	$2.0\text{x}10^{-6}$	$2.2\text{x}10^{-7}$
1000	$3.7\text{x}10^{-5}$	$5.4\text{x}10^{-6}$

As shown in Subsection 2.3.1, the radio-active decay terms will not be included in the preconditioned system. In the core sample problem the largest eigenvalue of the coefficient matrix is given by the neutron capture rate (2.17) of Xe^{135}, which depends linearly on the neutron flux. Therefore, if the flux increases by 1%, the largest eigenvalues of the coefficient matrix will increase by 1% and the largest eigenvalue of the preconditioned system will be 1% of the largest eigenvalue of the coefficient matrix. The stiffness (2.54) of the preconditioned system will therefore be 1% of the stiffness of the coefficient matrix. The accuracy therefore depends on the largest eigenvalue of the preconditioned system, which depends on the largest neutron capture rate and the change in neutron flux.

Computer time

Because both the step size and the order of the Runge-Kutta algorithm are fixed, the computer time needed is directly proportional to the burnup time for times longer than one day. For shorter burnup times the required tolerance is met with only one Runge-Kutta time step. Although the fourth order Runge-Kutta method requires about double the amount of calculations per time step than a fourth order Taylor expansion, it can use larger integration time steps, because of reduced stiffness of the problem it must solve. Therefore, for a burnup time of 0.1 to 1 days, for which the core problem is not very stiff, the fourth order Runge-Kutta algorithm (solid blue line) needs only one time step, making it faster than the fastest Taylor algorithm (dotted red line), as shown in Figure 5.5.

5.3.3 Runge-Kutta implementation using lower triangular solution

Accuracy

When the feedback terms, the terms above the diagonal of the coefficient matrix, are small compared to the diagonal terms, which is the case for the core problem as can be seen in (2.38), the Parlett & Runge-Kutta algorithm, Algorithm 14 on page 54, can be used.

Table 5.4 shows the maximum relative difference in final isotope concentrations when the feedback terms are neglected and when one Runge-Kutta correction step is used. In depletion modules that

use triangular codes, like the GAUGE code, the accuracy will be much improved at small burnup times by adding a single Runge-Kutta correction step.

Computer time

In the Parlett & Runge-Kutta implementation, the lower triangle of the coefficient matrix was stored in block notation, because the Parlett algorithm is used to solve the lower triangle, and the feedback terms were stored using the index storage scheme, which is the fastest for vector-matrix calculations.

For long burnup times, the preconditioned system also becomes stiff and only one fourth order Runge-Kutta step becomes insufficient when a high accuracy is required. Figure 5.5 shows the computer time needed by the Parlett & Runge-Kutta implementation when only one Runge-Kutta step is done (solid black line) and when more steps are done for longer burnup times to obtain an accuracy of 10^{-10} (dotted black line). For long burnup times the computer time used by this Runge-Kutta algorithm, like the Runge-Kutta algorithm of the previous subsection, increases linearly with time, making the algorithm slow for stiff systems.

5.4 Triangular methods

In this section the computer speed and accuracy of the GAUGE and Parlett implementations are discussed. In Subsection 5.4.1 some sample triangular problems are constructed and the performance of these implementations in solving the problems is discussed in Subsection 5.4.2.

5.4.1 Test problems

Because both the analytical GAUGE and Parlett methods, described in Chapter 4, can only solve systems with a lower triangular coefficient matrix, the implementations of these two methods were compared by solving the systems containing the lower triangular terms of the core and assembly examples of Subsection 5.2.1. Because the terms above the diagonal of the coefficient matrices are small and few compared to the terms on and below the diagonal, the eigenvalues and stiffness of the triangular depletion problems are almost the same as those of the full depletion problems.

To compare the effect of confluent eigenvalues on the analytical methods, three simple test depletion systems, having no feedback terms, were considered. The reference solution was calculated analytically using (4.2). The initial isotope number densities were assumed to be zero except for the first isotope with the number density normalised to one. The test systems, described by their coefficient matrices **A**, are:

Test 1:

$$\mathbf{A} = \begin{bmatrix} -1 & 0 \\ 1 & -1-\epsilon \end{bmatrix}, \tag{5.4}$$

which describes two isotopes with almost the same decay constants, -1 and $-1-\epsilon$, with the first isotope the parent of the second isotope. The eigenvectors of the coefficient matrix are the columns of the matrix

$$\mathbf{V} = \begin{bmatrix} 0 & \epsilon \\ 1 & 1 \end{bmatrix}, \tag{5.5}$$

having cond($\mathbf{V}$)=$\mathcal{O}(\frac{1}{\epsilon})$. From (4.1) and (4.2), the solution of this test problem is

$$e^{\mathbf{A}t} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} e^{-t} \\ \frac{e^{-t}-e^{-t-\epsilon t}}{\epsilon} \end{bmatrix}. \tag{5.6}$$

Test 2:

$$\mathbf{A} = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1-\epsilon & 0 \\ 0 & 1 & -1-2\epsilon \end{bmatrix}, \tag{5.7}$$

describing three isotopes with almost the same decay constants where the first isotope is the parent of the second isotope and the second isotope is the parent of the third isotope. The solution of this system is

$$e^{\mathbf{A}t} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} e^{-t} \\ \frac{e^{-t}-e^{-t-\epsilon t}}{\epsilon} \\ \frac{e^{-t}-e^{-t-2\epsilon t}}{2\epsilon} + \frac{1}{2\epsilon^2}\left[e^{-t} - 2e^{-t-\epsilon t} + e^{-t-2\epsilon t}\right] \end{bmatrix}. \tag{5.8}$$

Test 3:

$$\mathbf{A} = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1000 & 0 \\ 0 & 1000 & -1-\epsilon \end{bmatrix}, \tag{5.9}$$

which describes two isotopes with almost the same decay constant, with a fast decaying isotope in-between them in the decay chain. The solution of this system is

$$e^{\mathbf{A}t} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} e^{-t} \\ \frac{e^{-t}-e^{-1000t}}{999} \\ \frac{1}{\epsilon}\left[\frac{e^{-t}-e^{-1000t}}{999} + \frac{e^{-1000t}-e^{-t-\epsilon t}}{1000-\epsilon}\right] \end{bmatrix}. \tag{5.10}$$

The roundoff errors introduced by the analytical methods in solving these three systems are discussed in the next subsection.

5.4.2 Analytical GAUGE and Parlett implementations

Computer time

Both the GAUGE algorithm, Algorithm 8 on page 47, and the Parlett algorithm, Algorithm 9 on page 48, were implemented twice, using the full matrix storage and the block matrix storage scheme respectively. Table 5.5 shows the computer time needed by these four implementations for solving

Table 5.5: Computer time used by the GAUGE and Parlett implementations to solve the core and assembly depletion problems having no feedback terms

Algorithm	Storage Scheme	Assembly calculation Computer time ($s \times 10^{-3}$)	Core calculation Computer time ($s \times 10^{-6}$)
GAUGE	Full	3.6	10.6
	Block	1.7	9.8
Parlett	Full	4.6	8.2
	Block	1.0	6.1

Table 5.6: Roundoff errors of analytical algorithms when solving test case 1, (5.4)

ϵ	GAUGE algorithm	Parlett algorithm 1	Parlett algorithm 2	Roundoff error Double precision ($s = 15$)
10^{-6}	3.9×10^{-11}	6.0×10^{-11}	6.0×10^{-11}	1.0×10^{-9}
10^{-8}	1.0×10^{-8}	2.5×10^{-9}	1.0×10^{-10}	1.0×10^{-7}
10^{-10}	1.4×10^{-7}	3.9×10^{-7}	1.0×10^{-10}	1.0×10^{-5}
10^{-12}	1.1×10^{-4}	4.3×10^{-5}	3.3×10^{-13}	1.0×10^{-3}

Table 5.7: Roundoff errors of analytical algorithms when solving test case 2, (5.7)

ϵ	GAUGE algorithm	Parlett algorithm 1	Parlett algorithm 2
10^{-6}	1.1×10^{-4}	5.3×10^{-5}	1.0×10^{-8}
10^{-8}	-	-	1.9×10^{-8}
10^{-10}	-	-	3.9×10^{-7}
10^{-12}	-	-	1.5×10^{-4}

Table 5.8: Roundoff errors of analytical algorithms when solving test case 3, (5.9)

ϵ	GAUGE algorithm	Parlett algorithm 1	Parlett algorithm 2
10^{-6}	2.0×10^{-11}	7.4×10^{-11}	7.4×10^{-11}
10^{-8}	1.0×10^{-9}	8.6×10^{-9}	8.6×10^{-9}
10^{-10}	8.6×10^{-7}	1.4×10^{-6}	1.4×10^{-6}
10^{-12}	5.5×10^{-5}	1.9×10^{-4}	1.9×10^{-4}

the triangular core and assembly sample problems. Because the computer time of the two analytical methods is independent of the length of the burnup time, these values were calculated by averaging the computer times given in Tables B.6 and B.7. It can be seen that the computer time of the two methods is very similar, with the Parlett method slightly faster than the GAUGE method. The block matrix implementations are faster than the full matrix implementations for both methods. Because the GAUGE method, implemented with block matrix computations, is currently used in depletion modules, it will be used in Section 5.5 where the different methods are compared against each other.

Accuracy

As shown in Table B.6, the roundoff errors introduced by the two analytical methods for the assembly sample problem are about 10^{-7}, which is much larger than the errors of around 10^{-11} that are introduced in the polynomial methods. This is due to nearly confluent eigenvalues.

To investigate the effect of nearly confluent eigenvalues on the roundoff errors, the three simple test cases, given in Subsection 5.4.1, were solved with the GAUGE and Parlett algorithms. The maximum relative roundoff errors introduced by the three algorithms in solving the three test problems, (5.4), (5.7) and (5.9), are given in Tables 5.6, 5.7 and 5.8 respectively. The closeness of the eigenvalues, ϵ, was changed from 10^{-6} to 10^{-12} to show how the roundoff error increases as the eigenvalues approach each other. From the three tables it can be seen that the roundoff errors of the GAUGE algorithm and first Parlett algorithm are nearly the same.

In Section 4.1 it was shown that the loss in accuracy is due to subtractions in the numerator and denominator in both methods. To show that this is indeed the case, a second Parlett algorithm was constructed. This second Parlett algorithm uses (4.16) which is equivalent to (4.15) used in the first Parlett algorithm. But, when the decay constant of a parent and daughter is very close, the second Parlett algorithm uses (4.7) to eliminate the roundoff errors introduced. In Table 5.6 it can be seen that the second Parlett algorithm is much more accurate then the first Parlett and GAUGE algorithms in solving the first test case. In this table it is also shown that the roundoff errors for the first Parlett and GAUGE algorithms are smaller than the theoretical maximum roundoff error, which was computed with (4.5).

When the one isotope, having almost the same decay constant as one of its precursors, is not the direct daughter of this precursor, (4.7) used in the second Parlett algorithms can not eliminate the roundoff error and the two Parlett algorithms are equivalent. This can be seen in Table 5.8 for the third test case (5.9).

When three eigenvalues in a decay chain are nearly confluent, as in the second test case (5.7) much larger roundoff errors are introduced. Table 5.7 shows how both the GAUGE and first Parlett implementations give totally wrong isotope concentrations for $\epsilon \leq 10^{-8}$ when solving the third test case.

All the analytical methods suffer the drawback that roundoff errors are introduced for nearly confluent eigenvalues. When the coefficient matrix is triangular, the eigenvalues are given by the decay

Table 5.9: Fastest algorithm and implementation for each method to solve the core and assembly problems

Method	Assembly problem	Core problem
Taylor expansion	Taylor 2: Index	Taylor 2: Index
Padé approximation	Padé 2: Block	Padé 2: Block
Krylov subspace approximation	Krylov: Index	-
Chebyshev	Chebyshev 2: Block	Chebyshev 1: Full
Schur decomposition 1	Schur 1: Block	Schur 2: Full
Runge-Kutta using lower triangular solution	-	Parlett & Runge-Kutta: Block & Index
Runge-Kutta using previous solution	-	Runge-Kutta: Index
GAUGE	GAUGE: Block	GAUGE: Block

constants and the roundoff errors can be estimated using (4.5) and (4.6). If these errors are larger than the specified tolerance, numerical methods have to be used.

5.5 Fastest implementations

In Sections 5.2 to 5.4 the different implementations and algorithms of the same method were compared with each other to find the fastest implementation for each method, as summarised in Table 5.9. In this section the computer times used by these fastest implementations are compared with each other. Although different implementations of the Krylov algorithms were not compared in the previous sections, the index implementation of Krylov Algorithm 5 on page 40, will be compared against the other methods in this section. Although slightly different problems were solved by the GAUGE and Runge-Kutta implementations, it is assumed that the slight change in the problems did not influence the speed of the implementations.

Figures 5.6 and 5.7 show the computer time used to solve the sample depletion problems for different burnup times for the implementations given in Table 5.9. The values are given in Tables B.6 and B.7.

For a large system, like the assembly depletion problem, the Taylor method is by far the fastest method for small burnup time intervals, when the system is non-stiff, as shown by the red line in Figure 5.6. For larger burnup times, when the system is stiff, the Chebyshev method (dotted blue line) performs the best, requiring about the same amount of time as the GAUGE method (dotted green line), because it has no matrix-matrix operations and the computer time is independent of the stiffness. The Schur method (solid green line) and Padé method (solid blue line) are slower because of the matrix-matrix operations that are necessary. Although the Krylov method (black line) is faster than the Padé method for small burnup time steps, it gets slower at larger burnup times because it has to use integration time steps. Therefore, by using the Taylor method for small

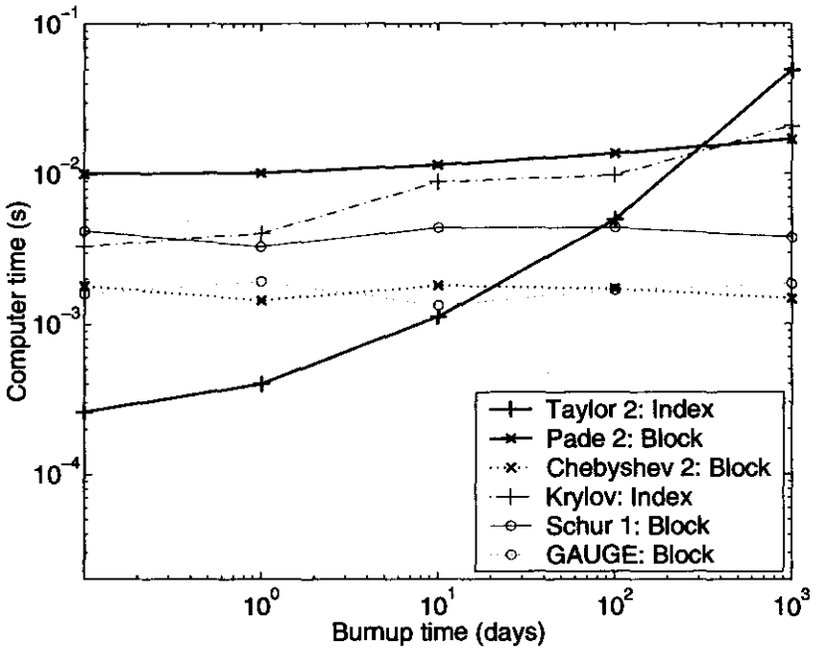

Figure 5.6: Computer time used by the fastest implementations to solve the assembly depletion problem

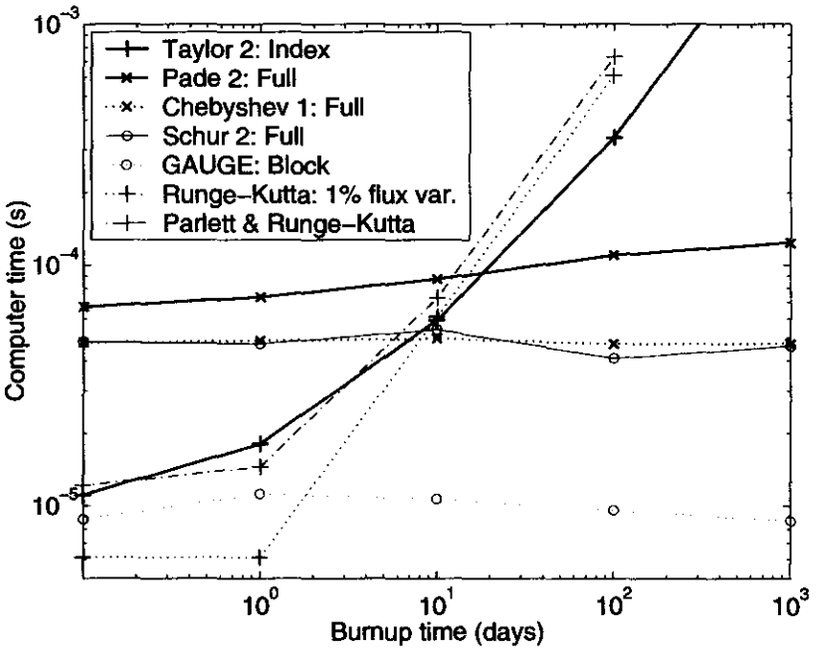

Figure 5.7: Computer time used by the fastest implementations to solve the core depletion problem

Table 5.10: Fastest method for different stiffness and sizes of the depletion problem

	Large system	Small system (high accuracy)	Small system (low accuracy)
Non-stiff	Taylor 2: Index	Taylor 2: Index	Runge-Kutta
stiff	Taylor 2: Index Chebyshev 2: Block	Chebyshev 1: Full	Parlett & Runge-Kutta
Very stiff	Chebyshev 2: Block	Chebyshev 1: Full	Parlett & Runge-Kutta

burnup times and the Chebyshev method for longer burnup times the same, or less, computer time will be necessary than the time used by the GAUGE code.

For small systems, as shown in Figure 5.7 for the core problem, the Schur (solid green line), Padé (solid blue line) and Chebyshev (dotted blue line) methods perform almost the same, with the Padé method about two times slower than the other two methods. For small burnup times, the Taylor method (solid red line) is faster than these three methods, but still slower than the analytical GAUGE method (dotted green line). For larger burnup times, the three methods are faster than the Taylor method, but about four times slower than the GAUGE code. When speed is important, the Runge-Kutta methods, which have a low accuracy, can be used. If a previous solution of the core problem is known for a 1% smaller neutron flux, and used to precondition the coefficient matrix, the Runge-Kutta algorithm (dotted red line) will solve the system in one time step for small burnup time, which is faster then the GAUGE method. When the preconditioned system becomes stiff, the computer time of the Runge-Kutta method increases linearly with time, making it much slower than the GAUGE method for long burnup times. When the Parlett & Runge-Kutta algorithm (black line) is used, the computer time is longer than the GAUGE method's time because the Parlett method alone takes the same amount of time as the GAUGE method. When accuracy is not very important only a few Runge-Kutta steps will be necessary, and this method will be the fastest general method for small, stiff systems.

Table 5.10 gives a summary of the fastest methods for different stiffness and size of the depletion problem. Before these methods can be accepted as the best, it is first necessary to investigate the generality, accuracy, reliability, stability, and ease of use of the methods, which will be done in the next section.

5.6 Comparison of methods against criteria

In Section 5.5 the different methods to solve the fuel depletion equations were compared against each other by measuring the computer time necessary to solve the two sample depletion problems. The computer time gives an indication of the efficiency of the methods, which is only one of the criteria mentioned in Section 1.6. In order to choose the best method, this section will evaluate the 8 fastest implementations, as listed in Table 5.9, according to the criteria mentioned in Section 1.6.

5.6.1 Generality

All the methods listed in Table 5.9, except the GAUGE method, can solve any linear system of first order differential equations having a coefficient matrix independent of time. This includes any system of the form (1.1) and (1.2), with **A** a constant matrix. This includes all feedback mechanisms where a daughter isotope may be its own precursor.

The implementations using the block storage scheme assume that it is possible to separate the actinides and fission products, which is a valid assumption as explained in Section 2.2. All the methods can be used for large and small depletion problems, except the Krylov method which is only applicable to large systems having more than 30 isotopes. For small matrices it reduces to the Padé approximation.

Therefore, unlike the GAUGE method in which the coefficient matrix must be triangular, none of the other 7 methods has any restrictions on the type of depletion mechanisms that can be described.

5.6.2 Efficiency

There are a number of factors that influence the computer time needed by the various methods:

Type of operations: Methods that use only matrix-vector operations, which use $\mathcal{O}(n^2)$ flops, will be faster than methods using matrix-matrix operation, which use $\mathcal{O}(n^3)$ flops, when the number of isotopes n is large. The Taylor, Chebyshev and Runge-Kutta algorithms are of order $\mathcal{O}(n^2)$ and the Schur, Padé and GAUGE algorithms are of order $\mathcal{O}(n^3)$ as shown in Table 5.11. The Krylov approximation use both matrix-vector multiplications and matrix-matrix computations on a dense 30-by-30 matrix.

Implicit or Explicit: Implicit numerical methods must have some sort of matrix inversion or must solve a linear system, which is not necessary in explicit methods.

Complex numbers: Methods using complex operations, like the Chebyshev method, are in general slower than methods using only real operations.

Zero calculations: When there are many elements in the coefficient matrix, methods that use a look up scheme to eliminate zero computations will be more efficient.

Dependence on stiffness: The computer time used by some methods is independent of the stiffness of the problem, like the Chebyshev and analytical methods, or increases logarithmically with stiffness, like the Padé method. When the problem is stiff, these methods will perform better than methods whose computing time increases almost linearly with the stiffness of the problem, like the Taylor method.

Table 5.11 gives a summary of which factors influence each method.

Table 5.11: Factors influencing the efficiency of each method

Method	Matrix-matrix computations	Complex computations	Number of calculations with zeros	Time dependence of method	Implicit method
Taylor	$\mathcal{O}(n^2)$	No	None	Almost linear for stiff systems	No
Padé	$\mathcal{O}(n^3)$	No	Some	Logarithmic	Yes
Krylov	$\mathcal{O}(n^2)+\mathcal{O}(30^3)$	No	None	Logarithmic	No
Chebyshev	$\mathcal{O}(n^2)$	Yes	Some	Constant	Yes
Schur	$\mathcal{O}(n^3)$	No	Some	Constant	No
Parlett & Runge-Kutta	$\mathcal{O}(n^3)$	No	Some	Linear	No
Padé & Runge-Kutta	$\mathcal{O}(n^2)$	No	None	Linear	No
GAUGE	$\mathcal{O}(n^3)$	No	Some	Constant	No

The efficiency depends strongly on how the methods are implemented, which type of compiler is used and on what type of computer the code is executed. For example, some computers can do complex number calculations in parallel, using the same time for a complex number operation than for a flop, and some computers can do vector calculations very efficiently by using pipelining. The best way to compare the efficiency is therefore to compare the computer time needed by various implementations to solve some sample problems, as was done in Section 5.5 and summarised in Table 5.10.

5.6.3 Accuracy and reliability

Accuracy is the difference between the true solution and the solution given by the numerical method. There are two sources of error:

Roundoff errors are errors that are introduced due to the finite word length of the computer. This is typically introduced when two large numbers are subtracted, like when a matrix having a small condition, is inverted.

Truncation errors are errors that are introduced when non-exact arithmetical expressions are used. For the analytical methods the truncation error is zero and the truncation errors for the polynomial methods are derived in Appendix A and shown in Table A.1.

When there is a possibility that large errors can be introduced without warning, a method is called unreliable. Even if such a method gives only very small roundoff errors for most problems, such a method can not be used. On the other hand, if a method can give a good estimate for the errors introduced or give a warning if some errors are above a certain tolerance, the method is reliable.

Table 5.12: Accuracy and reliability of each method

Method	Accuracy	Reliable
Taylor	Always	Yes
Padé	Always	Yes
Krylov	Not for very stiff systems	Yes
Chebyshev	Not for confluent eigenvalues	Yes
Schur	Not for confluent eigenvalues	Yes
Runge-Kutta	Depends on stiffness	Yes
GAUGE	Not for confluent eigenvalues	Yes

Table 5.12 gives a summary of the accuracy and reliability of each method. This table will be explained in the remainder of this subsection.

Taylor The truncation error of the Taylor expansion is given by (3.28) and can always be reduced below a given tolerance by using enough terms in the expansion. An upper limit for the roundoff error in the uniformization Taylor technique is given by (3.39), which can be reduced below the tolerance by choosing the right β in (3.40). The Taylor method is therefore always accurate and reliable when correctly implemented.

Padé The roundoff errors in the Padé methods are limited by scaling and squaring, as can be seen in the upper limit given by (3.52). By increasing the amount of scaling and squaring, the roundoff errors can thus be reduced below a given tolerance. The truncation errors of the (p,q)-Padé approximation are of the same magnitude as the truncation error for the $(p+q)$-th order Taylor expansion and can therefore also be limited below the tolerance by using a sufficiently high p and q. The Padé method is therefore also reliable.

Krylov Eq. (3.57) gives an upper limit for the truncation error in the Krylov approximation, which can be used to make the method reliable. When using the Krylov expansion with the subspace dimension of p, the roundoff error may increase like $(\Delta t)^p$ if the time increases by a factor Δt. For stiff systems the roundoff error may therefore become very large. These roundoff errors can be reduced by using a time stepping strategy as was done by Sidje (1998).

Chebyshev The maximum truncation error of the (14,14) Chebyshev approximation for scalars in the interval $(0,-\infty)$ is $1.8\text{x}10^{-14}$. When all the eigenvalues are distinct, roundoff errors are small and this is then also the accuracy of the matrix exponential. When confluent eigenvalues are present, the truncation error depends on the size of the confluent eigenvectors as shown in Table A.1. If the eigenvalues can be estimated by the diagonal of the coefficient matrix, it is possible to warn the user if there is a possibility of large confluent eigenvalues.

Analytical methods In the Schur and GAUGE methods, roundoff errors are introduced when eigenvalues are nearly confluent. Because the eigenvalues are known when using the GAUGE method and are calculated by the Schur decomposition, it is possible to estimate the roundoff errors using (4.5) and (4.6). The analytical methods are therefore reliable, although a given accuracy will not always be achievable.

Runge Kutta The accuracy of the Runge Kutta method was discussed in Subsection 5.3.2, and its reliability is similar to that of the Taylor method.

5.6.4 Stability

Although roundoff errors can amplify small perturbations, over many time steps these roundoff errors will tend to cancel each other out. Truncation errors are normally in one direction only, which could result in a build-up of errors. In order to evaluate this build-up of truncation errors in a numerical algorithm, a test equation $y' = \lambda y$ is used to compute the values y_n after each time interval t_n. The region of absolute stability (Gear, 1971:9) is then defined as the set of values of λ for which a perturbation in a single value y_n will produce a change in subsequent values which does not increase from step to step. A method solving the system of equations (1.1) will be absolutely stable if all the eigenvalues lie inside the region of absolute stability.

When a method is stable in the region $-z_{max} < z \leq 0$, it can be made stable for any system by scaling the system with a factor λ_{max}/z_{max}, where λ_{max} is the largest eigenvalue, and dividing the burnup time step ΔT into a number of integration time steps $\Delta t = z_{max}/\lambda_{max}\Delta T$.

A method is called A-stable if its region of absolute stability contains the whole of the left-hand half-plane (Cash, 1985:72). Gallopoulos and Saad (1992:1248) use the A-stability of the diagonal Padé approximations to prove that the Krylov method is also A-stable. They also state that the Chebyshev approximation can not be guaranteed to be stable because small eigenvalues near zero may be amplified. In Subsection 3.2.3 it was shown that the stability of the Taylor method can be ensured by specifying a minimum number of terms, using (3.32).

Because analytical methods have no truncation errors, they are all stable.

5.6.5 Ease of use

Table 5.13 gives the parameters used in each algorithm. Most of the parameters are automatically calculated, but some have to be specified by the user as shown. Normally these input parameters can be fixed, resulting in computer codes requiring no additional input parameters, which makes it easy to use it as part of a larger analysis system.

5.7 Summary

Among the eight methods that were considered, given in Table 5.9, only the GAUGE method has severe restrictions on the type of depletion mechanisms that can be described. All eight methods were also seen to be stable and reliable if properly implemented.

When accuracy is most important, the Taylor and Padé methods will be the best, with the Taylor method the fastest for non-stiff systems and the Padé method the fastest for stiff systems, as shown

Table 5.13: Parameters used and input parameters required by the different algorithms

Algorithm	Method	Required Parameters	Input Parameters
1 on page 34	Taylor 1	Integration step size Δt, accuracy $\varepsilon = 10^{-s}$	ε
2 on page 36	Taylor 2	Scale factor α, accuracy $\varepsilon = 10^{-s}$.	ε
3 on page 37	Padé 1	Scaling factor m, Order of expansion q.	-
4 on page 39	Padé 2	Scaling factor m, Number of matrix-vector multiplications k_2, Order of expansion q.	-
5 on page 40	Krylov	Subspace dimension m, accuracy $\varepsilon = 10^{-s}$	m, ε
6 on page 42	Chebyshev 1	Order of expansion p.	-
7 on page 43	Chebyshev 2	Order of expansion p.	-
8 on page 47	GAUGE	-	-
9 on page 48	Parlett	-	-
11 on page 50	Schur decomposition 1	Accuracy of Schur decomposition $\varepsilon = 10^{-s}$	ε
12 on page 51	Schur decomposition	Accuracy of Schur decomposition $\varepsilon = 10^{-s}$	ε
14 on page 54	Parlett & Runge-Kutta	Order of Runge-Kutta p, Number of steps s.	s
15 on page 54	Runge-Kutta	Order of Runge-Kutta p, Number of steps s.	s

Table 5.14: Best methods when accuracy is important for different stiffness and sizes of depletion problem

	Large system	Small system
Non-stiff	Taylor 2: Index	Taylor 2: Index
Stiff	Taylor 2: Index	Taylor 2: Index Padé 2: Full
Very stiff	Padé 2: Block	Padé 2: Full

Table 5.15: Best methods when speed is important for different stiffness and sizes of the depletion problem (copy of Table 5.10)

	Large system	Small system	Small system (low accuracy)
Non-stiff	Taylor 2: Index	Taylor 2: Index	Runge-Kutta
Stiff	Taylor 2: Index Chebyshev 2: Block	Chebyshev 1: Full	Parlett & Runge-Kutta
Very stiff	Chebyshev 2: Block	Chebyshev 1: Full	Parlett & Runge-Kutta

in Table 5.14. When speed is important, the Chebyshev method will be better than the Padé method for stiff systems, because it is faster but less accurate. The Taylor method will still be the best for non-stiff systems, as shown in Table 5.15. When a previous solution is known, the Runge-Kutta methods can be used for small systems to give a solution very fast, although the solution would not be very accurate. For small stiff systems, it is best to combine the Runge-Kutta method with an analytical method, like the Parlett method, and for large systems the best is to use a method like the Padé method which explicitly calculates the matrix exponential.

Chapter 6

Conclusion

In Chapters 3, 4 and 5 various methods to solve the fuel depletion equations (1.1) and (1.2), having a constant coefficient matrix **A**, were compared. In Section 5.7 the results were summarised and it was seen that different methods perform best under different conditions. In the depletion module, the method that performs the best under most conditions must be used, as will be discussed in Section 6.1.

In Section 6.2 it will be shown how the results of Chapter 5 can be used to qualitatively compare non-linear methods. Although only linear methods, requiring a time constant coefficient matrix, were investigated in this dissertation, non-linear methods may, in the future, become important.

6.1 Best linear method

The depletion module is typically called in two situations, as explained in Section 1.3. In both situations the constant flux approximation limits the maximum burnup time to less than 20 days, thereby limiting the stiffness of the system. In Chapter 5 the Taylor, Padé, Krylov and Chebyshev approximation methods, the Schur decomposition method and the Runge-Kutta preconditioning method were compared with each other, using the criteria of Section 1.6. Tables 5.14 and 5.15 give the best method under various conditions. The method that performed the best under most conditions is the Taylor method. This is because the Taylor method has the advantages that it can be guaranteed to be accurate within a given tolerance, it has no problem with confluent eigenvalues and it is stable when properly implemented. The only input parameter that is required is the tolerance, which can be fixed, thereby requiring no input from the user. The generalized uniformization algorithm, Algorithm 2 on page 36, which was found to be the fastest algorithm, is recommended. It is also recommended to use the index notation matrix storage scheme which is the fastest implementation, although any matrix storage scheme can be used. Because any matrix storage scheme can be used, it is easy to implement the method in an existing code system.

When speed is crucial, the Runge-Kutta method, using the triangular Parlett or GAUGE method as preconditioning, may be used, which is about four times faster than the Taylor method. The price

that must be paid is that this method is less accurate than the Taylor method and severe roundoff errors may be introduced with nearly confluent eigenvalues. This method can be investigated for specialised applications, such as core 'simulators', which normally treat a limited number of isotopes explicitly.

In their well-known article, Moler and Van Loan (1978:827) have compared 19 ways to calculate the matrix exponential, and concluded that all the methods are dubious in one way or another and that the performance will depend mainly on the type of problem and detail implementation of the methods. They also stated that the Taylor and Padé methods are the most effective methods we know of, when properly implemented. In a more recent paper, Sidje and Stewart (1999:367) have shown that for large, essential non-negative systems, the uniformization technique performs the best when the system is non-stiff and the Krylov subspace approximation is the best for stiff systems. Although the type of problems considered in these two papers are more general than the system of fuel depletion equations considered in this dissertation, their conclusions confirm that the uniformization Taylor method is one of the best methods to solve the fuel depletion equations.

6.2 Non-linear methods

When very accurate results of the depletion module are required, or when the neutron flux or isotope number densities are changing relatively fast in comparison with the burnup time step, like for a fresh core, the constant flux approximation breaks down. The fuel depletion equations then have the form

$$\frac{d\vec{N}(t)}{dt} = \mathbf{A}(t)\vec{N}(t), \tag{6.1}$$

where $\mathbf{A}(t) = \mathbf{A}(\Phi(t, E))$ depends on the neutron flux $\Phi(t, E)$. The neutron flux $\Phi(t, E)$ is calculated by solving the Boltzmann neutron transport equations and depends, among other time varying parameters, on the isotope number densities $\vec{N}(t)$. Eq. (6.1) can therefore be written as

$$\frac{d\vec{N}(t)}{dt} = \mathbf{A}(t, \vec{N}(t))\vec{N}(t). \tag{6.2}$$

When the isotope number densities $\vec{N}$ and the coefficient matrix $\mathbf{A}$ change relatively slow compared to the fastest decay constants, this system of ordinary differential equations (ODE) is stiff. Shampine and Gear (1979) give an overview of methods to solve stiff ODEs and Press *et. al.* (1993) give a number of methods for solving non-stiff ODEs. To individually discuss these methods is outside the scope of this dissertation, but some general similarities with the linear methods discussed in this dissertation will be shown.

Explicit ODE methods, like the well-known Runge-Kutta methods and a variety of multistep methods, behave like the Taylor method in that the integration step size is limited by stability and that the computation time increases almost linearly with stiffness. Explicit methods are therefore best

when solving non-stiff systems, as was seen with the Taylor method. If the system is not very stiff, a preconditioning method can be used to decrease the stiffness before an explicit method is used.

In the linear case, implicit methods like the Padé method were slow due to the fact that linear system had to be solved at each time step. In the nonlinear case, implicit methods require a non-linear system to be solved. This non-linear system is much slower to solve than the linear system as it includes the Boltzmann transport equation. The non-linear system must therefore be solved iteratively using some form of Newton iteration. The advantage of implicit methods is that they are A-stable and that the step size is therefore only determined by the accuracy and not by the stiffness. This was seen in the case of Chebyshev method, which required only one time step, and the Padé method where the step size was determined by the accuracy. Large step sizes can therefore be used, once all the fast decaying isotopes are in equilibrium with their precursors. For very stiff systems implicit methods will therefore be the best.

Multistep methods have the disadvantage that they are not A-stable and therefore do not share the advantage of the implicit methods, and are therefore in the same class as the explicit methods. The only exception is the two-step trapezoidal rule which is A-stable.

Press *et. al.* (1993:747) conclude in their paper that for low precisions, the explicit adaptive step size Runge-Kutta methods dominate, for high precisions and expensive function evaluations the implicit Bulirsh-Stoer methods dominate, and that the predictor-corrector methods are applicable between these two extremes. Because most of the systems which must be solved by the depletion module lie somewhere between non-stiff and very-stiff, the predictor-corrector methods seem like an attractive possibility. Most predictor-corrector methods consist of an explicit predictor equation and an implicit corrector equation. The explicit predictor is used to calculate a first approximation, which is then used (sometimes iteratively) in the corrector to evaluate the corrector explicitly. Predictor-corrector methods have better stability than explicit methods and do not require the solution of a non-linear system like implicit methods (Shampine & Gear 1979:9).

In the linear case it was found that analytical methods were not significantly faster than numerical methods, but that they had the drawback that confluent eigenvalues could introduce large roundoff errors. The Laplace transformation method, which is a more general analytical method, could be used, but as mentioned by Moler and Van Loan (1978:818) this method is $\mathcal{O}(n^4)$, which would make it much slower than the analytical methods considered, which were of $\mathcal{O}(n^3)$, and it could also be seriously affected by roundoff errors. It is therefore not expected that analytical methods will be better than well implemented numerical methods in both speed and accuracy, which was seen in this dissertation for the linear case.

6.3 Conclusion

When the time constant neutron flux approximation is used, the Taylor method, implemented with the uniformization algorithm and index matrix storage, was found to be the best method to use in the depletion module for solving the fuel depletion equations. Only for very long burnup time steps, which are not used in practise, do other numerical methods become competitive.

As the demand for more accurate reactor modelling increases, the constant flux approximation, which was necessary for a time constant coefficient matrix, has to be revised and general ODE methods will have to be used. Using the results obtained in this dissertation, it is expected that the preconditioned Runge-Kutta method or some predictor-corrector method will be the best ODE method to use in the depletion module.

Appendix A

Polynomial Approximation

When using a polynomial expansion to approximate the exponential of a scalar, the question arises whether this polynomial can also be used to compute the exponential of a matrix. In this appendix it is proven that:

1. If all the eigenvalues of a matrix are distinct, the polynomial expansion will approximate the matrix exponential to the same order of accuracy with which it approximates the exponential of the eigenvalues.
2. If eigenvalues are confluent, the order of accuracy depends on the order at which the polynomial expansion method fits the exponential at the origin.

In Section A.1 it is proven that if the above statements are true for all triangular matrices, they will be valid for all matrices. In Section A.2 the first statement is proven and in Section A.3 the second statement.

In this appendix the polynomial method to approximate the matrix exponential is given in terms of the two polynomials $D_p(x)$ and $C_q(x)$, of order p and q, giving

$$e^{-x} = \frac{D_p(x)}{C_q(x)} + \mathcal{O}(l) = R_{pq}(x) + \mathcal{O}(l), \tag{A.1}$$

in the scalar case and

$$e^{-\mathbf{A}} = C_q(\mathbf{A})^{-1} D_p(\mathbf{A}) = R_{pq}(\mathbf{A}), \tag{A.2}$$

to approximate the matrix exponential.

A.1 Triangularization

It is possible to translate the coefficient matrix $\mathbf{A}$ to a triangular matrix $\mathbf{T}$ by using an orthogonal transformation $\mathbf{Q}$:

$$\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^{-1}. \tag{A.3}$$

For any polynomial $P(\mathbf{A})$ it then follows that

$$P(\mathbf{A}) = \mathbf{Q}P(\mathbf{T})\mathbf{Q}^{-1}, \tag{A.4}$$

and because the matrix exponential can be written as a polynomial by using its Taylor expansion, the matrix exponential has the property that $e^{\mathbf{A}} = \mathbf{Q}e^{\mathbf{T}}\mathbf{Q}^{-1}$. To prove that $R_{pq}(\mathbf{A})$ indeed approximates $e^{\mathbf{A}}$, where $\mathbf{A}$ is any coefficient matrix, it is therefore sufficient to prove that $R_{pq}(\mathbf{T})$ approximates $e^{\mathbf{T}}$, where $\mathbf{T}$ is any triangular matrix, with negative eigenvalues.

A.2 Distinct eigenvalues

We use induction to prove (A.2). When the dimension of the n-by-n triangular matrix T is one, $n = 1$, (A.2) is the same as (A.1) and is therefore true. Let us state that (A.2) is true for all dimensions smaller than n. An n-by-n triangular matrix can be written as

$$\mathbf{T}_{nn} = \begin{bmatrix} \mathbf{T}_{rr} & 0 \\ \mathbf{T}_{rs} & \mathbf{T}_{ss} \end{bmatrix}, \tag{A.5}$$

where T_{rr} and T_{ss} are a r-by-r and a s-by-s triangular matrices, with $r + s = n$. Let

$$\mathbf{C}_q(\mathbf{T}_{nn}) = \begin{bmatrix} \mathbf{C}_{rr} & 0 \\ \mathbf{C}_{rs} & \mathbf{C}_{ss} \end{bmatrix},$$

and

$$\mathbf{D}_p(\mathbf{T}_{nn}) = \begin{bmatrix} \mathbf{D}_{rr} & 0 \\ \mathbf{D}_{rs} & \mathbf{D}_{ss} \end{bmatrix}. \tag{A.6}$$

Then we can write

$$\begin{aligned} \mathbf{C}_{rr} &= c_0 + c_1\mathbf{T}_{rr} + c_2\mathbf{T}_{rr}^2 + \ldots \\ \mathbf{C}_{ss} &= c_0 + c_1\mathbf{T}_{ss} + c_2\mathbf{T}_{ss}^2 + \ldots \end{aligned} \tag{A.7}$$

$$\begin{aligned} \mathbf{B}_{rr} &= b_0 + b_1\mathbf{T}_{rr} + b_2\mathbf{T}_{rr}^2 + \ldots \\ \mathbf{B}_{ss} &= b_0 + b_1\mathbf{T}_{ss} + b_2\mathbf{T}_{ss}^2 + \ldots. \end{aligned} \tag{A.8}$$

Eq. (A.2) can then be written as

$$\mathbf{F} = e^{-\mathbf{T}_{nn}} = \begin{bmatrix} \mathbf{C}_{rr}^{-1}\mathbf{D}_{rr} & 0 \\ \mathbf{C}_{ss}^{-1}\mathbf{D}_{rs} - \mathbf{C}_{ss}^{-1}\mathbf{C}_{rs}\mathbf{C}_{rr}^{-1}\mathbf{D}_{rr} & \mathbf{C}_{ss}^{-1}\mathbf{D}_{ss} \end{bmatrix} \tag{A.9}$$

$$= \begin{bmatrix} e^{-\mathbf{T}_{rr}} + O(l) & 0 \\ \mathbf{C}_{ss}^{-1}\mathbf{D}_{rs} - \mathbf{C}_{ss}^{-1}\mathbf{C}_{rs}\mathbf{C}_{rr}^{-1}\mathbf{D}_{rr} & e^{-\mathbf{T}_{ss}} + O(l) \end{bmatrix}. \tag{A.10}$$

$\mathbf{F}_{rr} = e^{-\mathbf{T}_{rr}} + \mathcal{O}(l)$ and $\mathbf{F}_{ss} = e^{-\mathbf{T}_{ss}} + \mathcal{O}(l)$ and, for distinct eigenvalues, the third term must be

(using Parlett recursive formula)

$$\mathbf{T}_{ss}\mathbf{F}_{rs} - \mathbf{F}_{rs}\mathbf{T}_{rr} = \mathbf{F}_{ss}\mathbf{T}_{rs} - \mathbf{T}_{rs}\mathbf{F}_{rr}. \tag{A.11}$$

It must therefore be proven that $\mathbf{F}_{rs}$ does indeed satisfy this formula. $\mathbf{F}_{rs}$ can be written as

$$\mathbf{F}_{rs} = \mathbf{C}_{ss}^{-1}(\mathbf{D}_{rs} - \mathbf{C}_{rs}\mathbf{C}_{rr}^{-1}\mathbf{D}_{rr}). \tag{A.12}$$

Constructing the sequence $\mathbf{S}_0 = 0$, $\mathbf{S}_1 = T_{rs}$, $\mathbf{S}_2 = (\mathbf{T}_{rs}\mathbf{T}_{rr} + \mathbf{T}_{ss}\mathbf{S}_1)$, $\mathbf{S}_i = (\mathbf{T}_{rs}\mathbf{T}_{rr}^{i-1} + \mathbf{T}_{ss}\mathbf{S}_{i-1})$, it follows that

$$\mathbf{D}_{rs} = d_0\mathbf{S}_0 + d_1\mathbf{S}_1 + d_2\mathbf{S}_2 + \ldots. \tag{A.13}$$

Using the property that $\mathbf{T}_{ss}\mathbf{S}_i - \mathbf{S}_i\mathbf{T}_{rr} = \mathbf{T}_{ss}^i\mathbf{T}_{rs} - \mathbf{T}_{rs}\mathbf{T}_{rr}^i$ it then follows that

$$\mathbf{T}_{ss}\mathbf{D}_{rs} - \mathbf{D}_{rs}\mathbf{T}_{rr} = \mathbf{D}_{ss}\mathbf{T}_{rs} - \mathbf{T}_{rs}\mathbf{D}_{rr}. \tag{A.14}$$

The same can be proven for $\mathbf{C}_{rs}$:

$$\mathbf{T}_{ss}\mathbf{C}_{rs} - \mathbf{C}_{rs}\mathbf{T}_{rr} = \mathbf{C}_{ss}\mathbf{T}_{rs} - \mathbf{T}_{rs}\mathbf{C}_{rr}. \tag{A.15}$$

It then follows that $\mathbf{T}_{ss}\mathbf{F}_{rs} - \mathbf{F}_{rs}\mathbf{T}_{rr}$ is given as follows:

$$\begin{aligned}
\mathbf{T}_{ss}\mathbf{F}_{rs} - \mathbf{F}_{rs}\mathbf{T}_{rr} &= \mathbf{T}_{ss}\mathbf{C}_{ss}^{-1}(\mathbf{D}_{rs} - \mathbf{C}_{rs}\mathbf{F}_{rr}) - \mathbf{C}_{ss}^{-1}(\mathbf{D}_{rs} - \mathbf{C}_{rs}\mathbf{F}_{rr})\mathbf{T}_{rr} \\
&= \mathbf{F}_{ss}\mathbf{T}_{rs} - \mathbf{T}_{rs}\mathbf{F}_{rr} \\
&= (e^{-\mathbf{T}_{ss}} + O(l))\mathbf{T}_{rs} - \mathbf{T}_{rs}(e^{-\mathbf{T}_{rr}} + O(l)).
\end{aligned} \tag{A.16}$$

The polynomial methods are therefore mathematically equivalent to the analytical Parlett method when eigenvalues are distinct, meaning that the polynomial does indeed approximate the matrix exponential.

A.3 Confluent eigenvalues

When there are confluent eigenvalues ($\mathbf{T}_{ss}\mathbf{T}_{rs} = \mathbf{T}_{rs}\mathbf{T}_{rr}$), the above proof breaks down and we must prove $\mathbf{F}_{rs}$ is equal to the true solution

$$\mathbf{F}_{rs,true} = \mathbf{T}_{rs}e^{\mathbf{T}_{rr}}. \tag{A.17}$$

When $\mathbf{T}_{ss}\mathbf{T}_{rs} = \mathbf{T}_{rs}\mathbf{T}_{rr}$, the sequence $\mathbf{S}_i$ reduces to $\mathbf{S}_i = i\mathbf{T}_{rs}(\mathbf{T}_{rr}^{i-1})$, and

$$\begin{aligned}\mathbf{F}_{rs} &= \mathbf{C}_{ss}^{-1}(\mathbf{D}_{rs} - \mathbf{C}_{rs}\mathbf{C}_{rr}^{-1}\mathbf{D}_{rr}) \\ &= \mathbf{T}_{rs}\left(\sum_{i=0}^{q} c_i \mathbf{T}_{rr}^{i}\right)^{-1}\left[\left(\sum_{i=1}^{p} i d_i \mathbf{T}_{rr}^{i-1}\right) - \left(\sum_{i=1}^{q} i c_i \mathbf{T}_{rr}^{i-1}\right)\mathbf{F}_{rr}\right].\end{aligned} \tag{A.18}$$

The difference between the true solution and the polynomial expansion is then

$$\mathbf{T}_{rs}\mathbf{E} = \mathbf{F}_{rs,true} - \mathbf{F}_{rs}$$

$$\mathbf{E}\left(\sum_{i=0}^{q} c_i \mathbf{T}_{rr}^{i}\right) = \left(\sum_{i=0}^{q} d_i \mathbf{T}_{rr}^{i}\right) - \left(\sum_{i=1}^{p} i d_i \mathbf{T}_{rr}^{i-1}\right) + \left(\sum_{i=1}^{q} i c_i \mathbf{T}_{rr}^{i-1}\right)\mathbf{F}_{rr}, \tag{A.19}$$

which can be written as

$$\mathbf{E} = \frac{d\mathbf{F}_{rr}}{d\mathbf{T}_{rr}} - \mathbf{F}_{rr}.$$

Now, if we introduce the time variable t

$$\mathbf{F}(t) = e^{t\mathbf{T}},$$

and suppose a method has an error of order p, $\mathbf{E} = c(t\mathbf{T})^p$, then

$$\mathbf{E} = c(t\mathbf{T})^p = \mathbf{T}^{-1}\frac{d\mathbf{F}(t)}{dt} - \mathbf{F}(t),$$

which can be used to calculate the derivatives of $\mathbf{F}(t)$

$$\begin{aligned}\frac{d\mathbf{F}(t)}{dt} &= \mathbf{T}\mathbf{F}(t) + c\mathbf{T}^{p+1}t^p \\ \frac{d^2\mathbf{F}(t)}{dt^2} &= \mathbf{T}^2\mathbf{F}(t) + ct^p\mathbf{T}^{p+1} + pc\mathbf{T}^{p+1}t^{p-1} \\ &\vdots \quad .\end{aligned}$$

The derivatives at $t = 0$ are then

$$\begin{aligned}\frac{d\mathbf{F}(t)}{dt}|_{t=0} &= \mathbf{T} \\ \frac{d^i\mathbf{F}(t)}{dt^i}|_{t=0} &= \mathbf{T}^i, \; i = 1, \ldots, p \\ \frac{d^{(p+1)}\mathbf{F}(t)}{dt^{(p+1)}}|_{t=0} &= \mathbf{T}^{p+1} + p!c\mathbf{T}^{p+1}.\end{aligned}$$

Because

$$\frac{d^i(e^{t\mathbf{A}})}{dt^i}|_{t=0} = \mathbf{A}^i,$$

this means that the first p derivatives must fit the exponential at the origin. Therefore, if we require a method to be accurate to order p when there are confluent eigenvalues, the polynomial must fit the exponential to degree p at the origin.

Table A.1: Truncation error due to confluent eigenvalues in polynomial methods

Method	p	q	Order of truncation error $\mathbf{E}$ for confluent eigenvalues	Order of truncation error (No confluent eigenvalues)
Taylor	p	0	$\mathbf{T}^p/p!$	$\mathbf{A}^{p+1}/(p+1)!$
Taylor	0	q	$\mathbf{T}^q/q!$	$\mathbf{A}^{q+1}/(q+1)!$
Padé	p	q	$\mathbf{T}^{p+q}/(p+q)!$	$\mathbf{A}^{p+q+1}/(p+q+1)!$
Chebyshev	14	14	$1.9{\cdot}10^{-14}$ $+2.1{\cdot}10^{-12}\mathbf{T}$ $+4.1{\cdot}10^{-11}\mathbf{T}^2$ $+3.2{\cdot}10^{-10}\mathbf{T}^3$ $+1.3{\cdot}10^{-9}\mathbf{T}^4$ $+3.5{\cdot}10^{-9}\mathbf{T}^5$ $+6.2{\cdot}10^{-9}\mathbf{T}^6$ $+8.2{\cdot}10^{-9}\mathbf{T}^7+\ldots$	$1.8\text{x}10^{-14}$

Table A.1 gives the truncation error of the Taylor, Padé and Chebyshev methods when there are confluent eigenvalues and when there are not. Note that the Taylor and Padé methods fit the exponential at the origin and confluent eigenvalues have the same effect as reducing the order of the approximation with one. The Chebyshev method fits the exponential on the whole interval $[0,-\infty)$ and is therefore always accurate when there are no confluent eigenvalues. Because it does not fit the exponential well at the origin, confluent eigenvalues could introduce large truncation errors.

Appendix B

Test Problems

In Table B.1 the initial and two final reference isotope number densities for the core calculation sample problem are listed. In Tables B.2, B.3 and B.4 the neutronic data, used in constructing the coefficient matrix (2.38), are given.

In Table B.5 the isotopes included in the assembly sample problem are listed, divided in one actinide group, eight fission product groups and 15 single isotopes.

Tables B.6 and B.7 show the computer time required and maximum errors introduced by the various implementations in solving the two sample problems for a burnup time ranging from 0.1 to 1000 days. Note that the GAUGE and Parlett methods solve a slightly different problem having no feedback term and the Runge-Kutta method, that exploits a small change in the neutron flux, solves the problem starting with equilibrium initial isotope concentrations. The results of these methods are therefore separated by double lines.

Because the timing routine used in FORTRAN does not give the CPU time with an accuracy better than 0.01 seconds, the computer time was measured by repeating a code until the CPU time used was longer than 1 second, and then dividing the CPU time by the number of repetitions.

Table B.1: Initial and final isotope number densities for the core depletion problem

Isotope	Initial number density (per unit volume)	Reference solution 100 days	Reference solution 1000 days
U^{235}	8.90×10^{-5}	7.89×10^{-5}	2.65×10^{-5}
U^{236}	2.35×10^{-5}	2.50×10^{-5}	3.00×10^{-5}
U^{238}	6.62×10^{-3}	6.60×10^{-3}	6.40×10^{-3}
Pu^{239}	2.55×10^{-5}	1.81×10^{-5}	8.25×10^{-7}
Pu^{240}	9.93×10^{-6}	9.28×10^{-6}	1.34×10^{-6}
Pu^{241}	7.80×10^{-6}	8.32×10^{-6}	3.11×10^{-6}
Pu^{242}	2.02×10^{-6}	2.46×10^{-6}	3.86×10^{-6}
I^{135}	4.70×10^{-9}	4.12×10^{-9}	1.52×10^{-9}
Xe^{135}	2.19×10^{-9}	1.90×10^{-9}	6.79×10^{-10}

Table B.2: Fission yields used in the core depletion problem

Isotope	Daughter	%	Fission cross section
	I^{135}	Xe^{135}	$(10^{-24}cm^2)$
U^{235}	6.326	0.264	28.9167
U^{236}	6.061	0.140	0.2945
U^{238}	6.552	0.017	0.1084
Pu^{239}	6.195	1.035	65.3574
Pu^{240}	7.031	0.537	0.6444
Pu^{241}	7.070	0.208	73.1890
Pu^{242}	6.836	0.087	0.4769

Table B.3: Decay constants used in the core depletion problem

Parent	Decay constant (s^{-1})	Daughter
Pu^{241}	$1.5253x10^{-9}$	-
I^{135}	$2.9129x10^{-5}$	Xe^{135}
Xe^{135}	$2.1182x10^{-5}$	-

Table B.4: Capture reactions included in the core depletion problem

Parent	Daughter	Capture cross section $(10^{-24}cm^2)$
U^{235}	U^{236}	7.166
U^{236}	-	6.008
U^{236*}	U^{235}	$1.439x10^{-3}$
U^{238}	-	0.8856
Pu^{239}	Pu^{240}	36.641
Pu^{240}	Pu^{241}	101.095
Pu^{240*}	Pu^{239}	$1.079x10^{-3}$
Pu^{241}	Pu^{242}	24.322
Pu^{241*}	Pu^{240}	$7.192x10^{-4}$
Pu^{242}	-	28.772
Pu^{242*}	Pu^{241}	$1.259x10^{-3}$
I^{135}	-	$1.359x10^{-3}$
Xe^{135}	-	$1.183x10^{5}$

Table B.5: Isotope blocks in the assembly depletion problem

Actinides	FP1	FP2	FP3	FP4	FP5	FP6	FP7	FP8	Single
Th^{232}	Zr95	I131	Ce141	Sm154	Dy161	Gd154	Er166	Hf176	Kr83
Pa^{233}	Zr96	I135	Ce142	Eu153	Dy162	Gd155	Er167	Hf177	Y89
U^{233}	Nb96	Xe131	Ce144	Eu154	Dy163	Gd156	Er168	Hf178	Zr91
U^{234}	Mo95	Xe132	Pr141	Eu155		Gd157	Er170	Hf179	Zr93
U^{235}	Mo96	Xe133	Pr143	Eu156		Gd158	Tm169	Hf180	Cd113
U^{236}	Mo97	Xe135	Nd143	Eu157			Tm171		In115
U^{237}	Mo98	Cs133	Nd144	Gd155			Yb171		I127
U^{238}	Mo100	Cs134	Nd145	Gd156					I129
Np^{237}	Tc99	Cs135	Nd146	Gd157					La139
Np^{239}	Ru100	Ba134	Nd147	Gd158					Tb159
Pu^{238}	Ru101		Nd148						Pseudo
Pu^{239}	Ru102		Nd150						B10
Pu^{240}	Ro103		Pm147						Ag107
Pu^{241}	Ru104		Pm148						Ag109
Pu^{242}	Ru105		Pm148*						Hf174
Am^{241}	Rh103		Pm149						
Am^{242*}	Rh105		Pm151						
Am^{243}	Pd104		Sm147						
Cm^{242}	Pd105		Sm148						
Cm^{243}	Pd106		Sm159						
Cm^{244}	Pd107		Sm150						
Cm^{245}	Pd108		Sm151						
	Ag109		Sm152						
	Cd110		Sm153						
	Cd111								

Table B.6: Computer time used by and accuracy of each implementation when solving the sample assembly depletion problem

Computer time (s)	$\times 10^{-3}$		$\times 10^{-3}$		$\times 10^{-3}$		$\times 10^{-3}$		$\times 10^{-3}$	
Max difference E		$\log(E)$		$\log(E)$		$\log(E)$		$\log(E)$		$\log(E)$
Burnup time: Days	1000		100		10		1		0.1	
Taylor 1: Full	495	*-13*	65.1	*-12*	13.7	*-11*	2.5	*-12*	1.0	*-12*
Taylor 1: Index	124	*-13*	17.2	*-12*	3.66	*-11*	0.751	*-12*	0.17	*-12*
Taylor 1: Block	250	*-13*	33.0	*-12*	7.21	*-11*	1.5	*-12*	0.4	*-12*
Taylor 2: Index	49.3	*-11*	5.0	*-12*	1.12	*-11*	0.4	*-11*	0.26	*-14*
Padé 1: Full	66	*-12*	56	*-12*	51.5	*-11*	41.1	*-11*	35.6	*-14*
Padé 2: Full	58.1	*-12*	50	*-12*	40.7	*-11*	35.1	*-11*	35.2	*-14*
Padé 2: Block	17	*-12*	13.7	*-12*	11.5	*-11*	10.2	*-11*	10.1	*-14*
Krylov	21	*-10*	9.8	*-12*	8.9	*-11*	4.0	*-11*	3.3	*-14*
Chebyshev 2: Full	44.2	*-7*	43.6	*-10*	43.9	*-11*	43.5	*-11*	43.5	*-11*
Chebyshev 2: Block	1.48	*-7*	1.72	*-10*	1.81	*-11*	1.44	*-11*	1.80	*-11*
Schur 1: Full	13	*-7*	13	*-7*	12	*-7*	13	*-8*	12	*-8*
Schur 1: Block	3.8	*-7*	4.4	*-7*	4.4	*-7*	3.3	*-8*	4.2	*-8*
Schur 2: Full	7.3	*-8*	7.6	*-7*	7.4	*-7*	7.5	*-7*	7.3	*-7*
GAUGE: Full	3.7	*-9*	3.6	*-7*	3.6	*-7*	3.7	*-7*	3.6	*-7*
GAUGE: Block	1.9	*-9*	1.7	*-7*	1.3	*-7*	1.9	*-7*	1.6	*-7*
Parlett: Full	4.6	*-8*	4.6	*-7*	4.6	*-8*	4.6	*-7*	4.6	*-7*
Parlett: Block	1.0	*-8*	1.1	*-7*	1.0	*-8*	1.0	*-7*	1.0	*-7*

Table B.7: Computer time used by and accuracy of each implementations when solving the sample core depletion problem

Computer time (s)	$\times 10^{-5}$		$\times 10^{-5}$		$\times 10^{-5}$		$\times 10^{-5}$		$\times 10^{-5}$	
Max difference E		log(E)		log(E)		log(E)		log(E)		log(E)
Burnup time: Days	1000		100		10		1		0.1	
Taylor 1: Full	981	-13	109	-14	24	-14	6.8	-12	0.8	-12
Taylor 1: Index	831	-13	92	-14	20	-14	5.2	-12	1.7	-12
Taylor 1: Block	891	-13	103	-14	23	-14	6.2	-12	1.2	-12
Taylor 2: Index	326	-11	34	-12	5.9	-12	1.8	-11	1.1	-13
Padé 1: Full	13	-12	12	-13	9.8	-13	8.1	-12	7.1	-14
Padé 2: Full	12	-12	11	-13	8.8	-13	7.4	-12	6.7	-14
Padé 2: Block	20	-12	15	-13	13	-13	10	-12	8.1	-14
Chebyshev 1: Full	4.7	-11	4.7	-13	5.0	-13	4.9	-12	4.8	-13
Chebyshev 2: Full	18	-11	17	-11	18	-11	17	-11	18	-11
Chebyshev 2: Block	5.7	-11	5.6	-11	6.0	-11	5.9	-11	5.9	-11
Schur 1: Full	7.4	-12	7.8	-11	7.9	-11	8.5	-11	8.1	-11
Schur 1: Block	5.1	-12	4.8	-11	5.6	-11	5.2	-11	4.6	-11
Schur 2: Full	4.6	-11	4.1	-11	5.4	-11	4.7	-11	4.8	-11
Parlett&Runge-Kutta: one step	1.2	-5	1.1	-7	1.4	-8	1.5	-10	1.2	-11
Parlett&Runge-Kutta: tol=10^{-10}	-		73	-10	7.3	-10	1.5	-10	1.2	-11
Runge-Kutta: 1% flux var.	-		61	-8	6.1	-8	0.6	-8	0.6	-8
Runge-Kutta: 0.1% flux var.	-		61	-9	6.1	-9	0.6	-9	0.6	-9
GAUGE: Full	1.2	-12	1.0	-11	1.2	-11	0.89	-11	1.0	-10
GAUGE: Block	0.87	-12	0.96	-11	1.1	-11	1.1	-11	0.88	-10
Parlett: Full	0.76	-12	0.71	-11	0.82	-12	1.0	-11	0.82	-12
Parlett: Block	0.52	-12	0.46	-11	0.70	-12	0.79	-11	0.55	-12

References

ANDERSON, E., BAI, Z., BISCHOF. C., BLACKFORD, S., DEMMEL, J., DONGARRA, J., DU CROZ, J., GREENBAUM, A., HAMMARLING, S., McKENNEY, A., OSTROUCHOV, S., SORENSEN, D. 1999. LAPACK Users' guide. 3rd ed. Philadelphia : SIAM.
[Web:] http://www.netlib.org/lapack/lug/lapack_lug.html

ARIOLI, M., CODENOTTI, B., FASSINO, C. 1996. The Pade method for computing the matrix exponential. *Linear Algebra and its Applications*, 240:111-130.

BERGAMASCHI, L., VIANELLO. M. 2000. Efficient computation of the exponential operator for large, sparse, symmetric matrices. *Numerical Linear Algebra with Applications*, 7(1):27-45.

CASH, J.R. 1985. Stiff methods (*In* Aiken, R.C., *ed.* Stiff computation. Great Britain : Oxford University Press. p. 70-90.)

CASTILLO, P., SAAD, Y. 1998. Preconditioning the matrix exponential operator with applications. *Journal of Scientific Computing*, 13(3):275-302, Sep.

CODY, W.J., MEINARDUS, G., VARGA, R.S. 1969. Chebyshev rational approximations to e^{-x} in [0,+∞) and applications to heat-conduction problems. *Journal of Approximation Theory*, 2:50-65.

DUDERSTADT, J.J., HAMILTON, L.J. 1975. Nuclear reactor analysis. N.Y. : Wiley. 672 p.

FAIR, W., LUKE, Y.L. 1969. Pade approximations to the operator exponential. *Numerical Mathematics*, 14:379-382.

FORREST, R.A. 1997. EASY-97: Overview. EASY Documentation series. Oxfordshire, UK : UKAEA Fusion. 19 p.

GALLOPOULOS, E., SAAD, Y. 1992. Efficient solution of parabolic equations by Krylov approximation methods. *SIAM Journal of Scientific and Statistical Computing*, 13(5):1236-1264, Sep.

GEAR, C.W. 1971. Numerical initial value problem in ordinary differential equations. Englewood Cliffs, N.J. : Prentice-Hall. 253 p.

GOLUB, G.H., VAN LOAN, C.F. 1989. Matrix computations. 2nd ed. London : Johns Hopkins. 642 p.

HERMANN, O.W., WESTFALL, R.M. 1998. ORIGEN-S: Scale system module to calculate fuel depletion, actinide transmutation, fission product buildup and decay, and associated radiation source term. Oak Ridge : Oak Ridge National Laboratory. NUREG/CR-0200 Rev 6 Vol 2 Sec F7.

HOCHBRUCK, M., LUBICH, C. 1997. On Krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 34(5):1911-1925, Oct.

ISERLES, A. 1981. Rational interpolation to exp(-x) with application to certain stiff systems. *SIAM Journal on numerical analysis*, 18(1):1-11.

LAMBERT, J.D. 1983. Computational methods in ordinary differential equations. Great Britian : Wiley. 278 p.

LAWSON, J.D. 1967. Generalized Runge-Kutta processes for stable systems with large Lipschitz constants. *SIAM Journal on Numerical Analysis*, 4(3):372-380, Sep.

LEVIS, A. H. 1969. Some computational aspects of the matrix exponential. *IEEE Transactions on Automatic Control*, 14(4):410-411, Aug.

MOLER, C.B, STEWART, G.W. 1973. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis*, 10(2):241-256, Apr.

MOLER, C., VAN LOAN, C. 1978. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20(4):801-836, Oct.

NERAC, 2002. A technology roadmap for generation IV nuclear energy systems: Ten nations preparing today for tomorrow's energy needs. U.S. Department of Energy's Nuclear Energy Research Advisory Committee. *GIF-002-00*. 91 p.
[Web:] http://gif.inel.gov/roadmap/pdfs/gen_iv_roadmap.pdf [2 Okt. 2003].

PARLETT, B.N. 1976. A recurrence among the elements of functions of triangular matrices. *Linear Algebra and its Applications*, 14:117-121.

PRESS, W.H., FLANNERY, B.P., TEUKOLSKY, S.A., VETTERLING, W.T. 1993. Numerical recipes in C: The art of scientific computing. 2nd ed. Cambridge University Press. 994 p.

SAAD, Y. 1992. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM Journal on numerical analysis*, 29(1):209-228, Feb.

SAAD, Y. 1990. Overview of Krylov subspace methods with applications to control problems, International symposium on the mathematical theory of networks and systems, Boston : Birkhauser, 401-410 p.

SHAMPINE, L.F., GEAR, C.W. 1979. A user's view of solving stiff ordinary differential equations. *SIAM Review*, 21(1):1-7, Jan.

SHAMPINE, L.F. 1985. What is stiffness? (*In* Aiken, R.C., *ed.* Stiff computation. Great Britain : Oxford University Press. p. 1-21.)

SIDJE, R.B. 1998. EXPOKIT: software package for computing matrix exponentials. *ACM - Transactions on Mathematical Software*, 24(1):130-156.

SIDJE, R.B., STEWART, W.J. 1999. A numerical study of large sparse matrix exponentials arising in Markov chains. *Computational Statistics & Data Analysis*, 29:345-368.

SMITH, B.T., BOYLE, J.M., DONGARRA, J.J., GARBOW, B.S., IKEBE, Y., KLEMA, V.S., MOLER, C.B. 1976. Matrix Eigensystem Routines - EISPACK guide. 2nd ed. Berlin : Springer-Verlag. 387 p.

STAMM'LER, R.J.J., ABBATE, M.J. 1983. Methods of steady-state reactor physics in nuclear design. Great Britain : Academic Press. 491 p.

STEWART, D.E., LEYK, T.S. 1996. Error estimates for Krylov subspace approximations of matrix exponentials. *Journal on Computational and Applied Mathematics*, 72:359-369.

SUBLET, J.-Ch., KOPECKY, J., FORREST, R.A. 1997. AEF-97: Cross section library. *UKAEA FUS 351.* Oxfordshire, UK : UKAEA Fusion. 108 p.

THOMAS, G.F., BARBER, D.H. 1993. Numerical solution of bateman systems. *Annals of Nuclear Energy*, 20(6):407-414, Jun.

THOMAS, G.F., BARBER, D.H. 1994. Stiffness of radioactive decay chains. *Annals of Nuclear Energy*, 21(5):309-320, May.

VAN LOAN, C. 1977. The sensitivity of the matrix exponential. *SIAM Journal on Numerical Analysis*, 14(6):971-981, Dec.

WAGNER, M.R. 1968. GAUGE: A two-dimensional few group neutron diffusion-depletion program for a uniform triangular mesh. San Diego : Gulf General Atomic Incorporated. AEC research and development report, SA-8307.