12-2017

Improvements to NESTLE: Cross Section Interpolation and *N*-Group Extension

William Matthews Kirkland

University of Tennessee, Knoxville, wkirklan@vols.utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by William Matthews Kirkland entitled "Improvements to NESTLE: Cross Section Interpolation and *N*-Group Extension." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.

Ondrej Chvala, Major Professor

We have read this thesis and recommend its acceptance:

G. Ivan Maldonado, Peter C. Lukens, Steven E. Skutnik

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Improvements to NESTLE: Cross Section Interpolation and *N*-Group Extension

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

William Matthews Kirkland

December 2017

for Nicholas, Ella, and Craig

Acknowledgements

I would like to thank my advisor, Dr. Ondřej Chvála, and my committee, especially Dr. Ivan Maldonado, for their extensive help and guidance at every step of my work. In addition, I thank my fellow graduate students: Dr. Nicholas Luciano, Dr. Cole Gentry, A.J. Pawel, and others, who have given me input files, references, coding recommendations, and many other useful ideas and materials to make this work easier.

I also would like to thank my employer, UT-Battelle, for their financial support of my studies. Particularly I want to acknowledge Mike Hechler and Dr. Peter Lukens for their personal and professional support in helping me pursue my studies, and Dr. Ken Tobin for getting me started in the first place.

Lastly and most importantly, to my family—to my parents Matt and Vicky Kirkland; to my grandparents Thomas and Mary Jo Morrow and Ned and Ona Mae Kirkland; to my brother Ben and my sister Molly; to my wife Maria, my son Nicholas, and my daughter Ella—thank you for your unconditional love and interminable support and patience, that I have continually relied upon for the past three years and throughout my life.

. . . in that Empire, the Cartographer's Art achieved such a Degree of Perfection that the Map of a single Province occupied an entire City, and the Map of the Empire, an entire Province. In time, these enormous Maps no longer sufficed, and the Guild of Cartographers struck a Map of the Empire whose size was that of the Empire, and which coincided with it point for point.

– Jorge Luis Borges, *"Del Rigor en la Ciencia"*

Abstract

The NESTLE program is a few-group neutron diffusion reactor core simulator code utilizing the nodal expansion method (NEM). This thesis presents two improvements made to NESTLE regarding cross-section interpolation and multigroup capability.

To quickly and accurately obtain cross sections from lattice physics input data, a new cross section interpolation routine was developed utilizing multidimensional radial basis function interpolation, also known as thin plate spline interpolation. Testing showed that, for existing NESTLE lattice physics input, accuracy was retained but not improved and processing time was longer. However, the new interpolation routine was shown allow much greater flexibility in the case matrix of the the lattice physics input file. This allows for much more detailed modeling of cross section variation at the expense of computation time.

The existing capability of NESTLE to use two or four neutron energy groups in the NEM calculation was supplemented with a new routine to allow the use of an arbitrary number of neutron energy groups by calling existing, widely used linear algebra libraries. This represents a significant expansion of NESTLE's capability to model a broader ranger of reactor types beyond typical light water reactors (LWRs). Testing revealed that the new NEM routines retained the accuracy and speed of the existing routines for two and four energy groups, while calculations with other numbers of energy groups had adequate accuracy and speed for practical use.

Table of Contents

List of Tables

List of Figures

Nomenclature

D	Diffusion coefficient
$\vec{J}$	Neutron current density
N	Matrix dimensionality (e.g., number of core physical parameters taken into account in a given diffusion code, number of neutron energy groups used)
N_{sp}	Soluble poison number density
P	Knot coordinate vector
T_c	Moderator/coolant temperature
T_F	Fuel temperature
$T_{F_{eff}}$	Effective fuel temperature
V	Internal strain energy
a_{gxn}	Nodal Expansion Method basis function expansion coefficients
$a_{j_{xg}}$	Cross section polynomial regression expansion coefficients
c_i	Radial basis function polynomial term coefficients
g	Neutron energy group (subscript)
k	Neutron multiplication factor

k_{eff}	Effective neutron multiplication factor
n	number (of neutrons, of interpolation points, etc.)
pcm	per cent mille (parts per 100,000)
r_0	Radial basis function distance scaling factor
$s\left(\vec{x}\right)$	Interpolating spline function
v	neutron velocity
Σ_a	Macroscopic absorption cross section
Σ_f	Macroscopic fission cross section
Σ_R	Macroscopic removal cross section
Σ_s	Macroscopic scattering cross section
Σ_t	Macroscopic total cross section
Σ_{tr}	Macroscopic transport cross section
Σ_x	Arbitrary macroscopic cross section or similar variable (e.g., Assembly Discontinuity Factor)
$\Phi\left(\vec{x}\right)$	Radial basis function kernel
λ_i	Radial basis function knot weights
ν	Neutrons produced per fission
ξ	In-node position variable, $\frac{x}{\Delta x}$
ρ_c	Moderator/coolant density
ϕ	neutron flux

$\varphi\left(\vec{x}\right)$	Airy stress function
χ	neutron fission energy spectrum

Chapter 1

Introduction

1.1 Nuclear Analysis and Neutron Transport

The calculation of the spatial and energy distribution of neutrons in a nuclear reactor is one of the central problems in nuclear reactor analysis. In a reactor, neutrons may be produced by fissions or by neutron sources, they may be removed by absorption or by escaping the confines of the reactor, and their energy and direction may be altered by interactions with fuel and non-fuel reactor materials. Careful consideration of these processes on a differential volume, and their time-dependent effects on neutron population, energy, and direction, allows for the development of an integro-differential equation describing the aggregate macroscopic behavior of neutrons in a reactor, known as the neutron transport equation.

The neutron transport equation is capable of modeling the neutronic behavior of a nuclear reactor to a high degree of accuracy. Thanks to the advances of modern supercomputing, in the foreseeable future it may be feasible to directly simulate the behavior of a nuclear reactor using direct solution of the neutron transport equation applied to a detailed physical model of a reactor, using finite difference, finite element, or Monte Carlo techniques. Nonetheless, direct solution of the neutron transport equation when applied to a large nonhomogeneous system such as a nuclear reactor is a daunting task. The neutron transport equation is an integro-differential equation in seven variables: neutron energy, the three dimensional neutron flux, the two dimensional neutron current density[1], and time. Furthermore, hundreds of nodes in each spatial variable are required to accurately represent any reactor's irregular distribution of fuel, moderator, neutron poisons, support structures, and other materials. Therefore, simplified, more tractable mathematical models have been the primary tool of the nuclear reactor analyst for decades, and these simplified approaches will continue to be useful in the years to come.

1.2 Simplifications to Neutron Transport

Perhaps the three most widespread and useful simplifications to neutronic analysis are energy grouping, the diffusion model, and spatial homogenization. Energy grouping consists of binning the continuous energy variable into a finite set of energy groups. Neutrons liberated due to fissions are placed into the various energy groups according to a pre-determined empirical distribution, and the neutrons then may move between energy groups due to scattering interactions.

The neutron diffusion model of a reactor allows the seven independent variables previously mentioned to be reduced to five. This is accomplished by the use of the Diffusion Approximation[2] drawn from the theory of mass transfer. With the Diffusion Approximation, rather than treating the two dimensional current density as an independent variable, neutrons are assumed to propagate from areas of high flux to areas of low flux in a quantity proportional to the gradient of the flux. Mathematically, this means that the current density independent variable can be replaced by the gradient of the flux multiplied by a proportionality constant known as the diffusion

[1]somewhat confusingly, it is convention in nuclear reactor theory that the three positional variables analogous to the concentration in kinetic theory are known as the "neutron flux", while the two directional variables analogous to flux in kinetic theory are known as the "neutron current density." This terminology is used throughout.

[2]also known as Fick's First Law or Fick's Law

coefficient [1],

$$\vec{J} = -D\nabla\phi$$

Note, however, that mass transfer theory tells us that this approximation is strictly valid only when the mean free path is much greater than the characteristic dimension of the system. Despite the computational benefits of this reduction in dimensionality, it is important to bear in mind that in typical nuclear reactors the neutron mean free path is on the order of the fuel pin diameter. Because of this, the diffusion approximation often requires refinement to accurately model reactor behavior.

Finally, spatial homogenization consists of "smearing" material properties over larger areas of the reactor, to reduce the number of nodes required for an accurate simulation. To accomplish this, the reactor analysis calculation is broken into several steps:

- First, cross section processing is performed. For deterministic calculations, a detailed analysis of flux distribution is performed for a single lattice cell, allowing the replacement of the continuous energy cross section data with discrete multigroup cross sections. For Monte Carlo calculations, the continuous energy cross section data is adjusted to the assumed operating temperatues, to account for the impact of Doppler broadening on the cross sections.

- Second, a two-dimensional model is constructed containing fine detail of the composition of a fuel assembly. The macroscopic, averaged properties of the assembly are then calculated using deterministic or Monte Carlo techniques (e.g., CASMO, SERPENT, NEWT). The results of this *lattice physics calculation* are the flux-weighted averages of the various macroscopic cross sections of the assembly for each neutron energy group, at a variety of operating conditions (e.g., differing fuel and moderator conditions and neutron poison concentrations), and at various stages of fuel depletion.

- Third, these averaged cross sections are used in a neutron diffusion code (e.g., SIMULATE, NESTLE), along with core loading and geometry and coupled calculations such as thermal-hydraulic feedback, to simulate the reactor-level and time-dependent behavior of the system.

The current work pertains to the use of the lattice physics results in the diffusion code, i.e., the interrelation of the second and third steps above.

An additional simplification is often used when the time-scales of interest are long; specifically, when fuel depletion and isotope tracking are the primary time-dependent effects of interest. In such cases, rather than using a time-dependent form of the neutron diffusion equation, the system is analyzed as a series of steady-state operations at various points along the fuel depletion cycle. The point along the fuel depletion cycle is quantified as the time-integral of specific reactor operation power, referred to as "burnup", conventionally given in units of megawatt-days per metric ton heavy metal ($\frac{\text{MWd}}{\text{THM}}$, equivalent to $86.4 \cdot 10^6 \frac{\text{J}}{\text{kg U}}$). This quasi-steady-state approach is used throughout the current work.

1.3 The NESTLE Code

The NESTLE (**N**odal **E**igenvalue, **S**teady-state, **T**ransient, **L**ow-enriched core **E**valuator) code is a few-group, diffusion-based core simulation code that employs the Nodal Expansion Method (NEM). NESTLE has the capability to solve both steady-state and transient problems, using three-dimensions with either Cartesian or hexagonal geometry. NESTLE also has the capability to apply thermal-hydraulic feedback including single and two-phase flow. NESTLE was originally developed by Paul Turinsky and others at North Carolina State University [2]. A modern version of the NESTLE nodal-diffusion simulator is currently maintained and developed at the University of Tennessee [3].

Prior to the current work, NESTLE was capable of using two or four energy groups. Polynomial fits were used to calculate the effects of varying material temperatures, densities, and poison concentrations to the homogenized cross sections. This work was undertaken eliminate the restriction to two or four neutron energy groups, and provide the capability of performing neutron diffusion calculations with any arbitrary energy group structure. Additionally, this work seeks to provide an alternative interpolation method for determination of homogenized cross sections. This alternative method is more flexible than polynomial fits. It also introduces the capability to account for the simultaneous variation of reactor parameters (e.g., fuel temperature and coolant density) in a more complex manner than simple linear combinations.

Chapter 2

Cross Section Interpolation

2.1 Overview

Reactor simulations through computer codes such as NESTLE, which use nodal diffusion methods, are capable of accurately modeling many aspects of reactor behavior, both globally (on the scale of the reactor) and locally (on the scale of fuel assemblies or even fuel pins). However, the accuracy of these codes depends on the accurate calculation of homogenized lattice parameters (e.g., cross sections) [4]. These homogenized parameters are determined using a lattice physics code such as NEWT, CASMO, or SERPENT, and provide " 'equivalent' diffusion theory parameters which are spatially constant (or smoothly varying) over the entire cross sectional area of a fuel assembly" [4] for each group in the few-group energy structure.

The homogenized few-group cross sections must be evaluated many times during execution of the diffusion code, as the values of these constants generally depend on burnup, control rod state, and on a small number of physical core parameters (e.g., the temperature of the fuel and moderator, the density of the moderator, and the concentration of soluble neutron poisons). Typical practice is to generate a set of these constants at various values of these parameters using lattice physics codes, and then use interpolation to determine the group constants used during the execution of the diffusion code [1]. Several options exist for how this interpolation is carried out:

- which physical core parameters are taken into account,
- the number and structure of the core parameter points where the group constants are to be determined, and
- the interpolation method used.

The options used by NESTLE and other codes, and improvements that have been implemented in NESTLE in the course of this work, are described below.

2.2 Core Parameters

As described above, the values of the few-group cross sections are primarily determined by a relatively small set of physical core parameters, such as burnup, control rod state, and material temperatures and densities. The list of parameters having a significant effect on the few-group constants depends on the type of reactor, but are typically drawn from the following ([5], [6]):

- burnup
- control rod/blade state
- fuel temperature
- moderator temperature
- moderator density/void coefficient
- soluble poison concentration
- control rod/blade history
- fuel temperature history

- shutdown cooling history
- void coefficient (or neutron spectrum) history

It is important to note that this list excludes input parameters such as initial material composition and reactor geometry, which do not vary during the course of code execution and are handled separately in the input data.

Currently, for each material ("color") used, NESTLE interpolates cross sections based on the first six physical core parameters in the list above [7]. Other codes (e.g., PARCS) also include one or more of the history parameters. These history parameters may have a significant effect on the calculated cross sections [6]. Therefore, although incorporation of history compensations into the NESTLE code is beyond the scope of the current work, the selected interpolation scheme should be easily extendable to incorporate these effects.

2.3 The Case Matrix

The number and structure of the core parameter points at which the cross sections are determined by the lattice physics code is known as the *case matrix*. There exist many possible case matrix structures, which differ from code to code and among the various types of reactors [5]. The differences include both the choice of physical core parameters used and which combinations of values of those parameters are selected as calculation points in the lattice physics code. Figure 2.1 illustrates a very simple case matrix structure.

To give the simplest possible example, a case matrix could consist of a single line in parameter space: a base point where all parameters have their nominal values, and a second point at which all parameters vary from the base case. For the diffusion code to determine a cross section at a given operational point in parameter space, that point would be projected onto the case matrix line, and the cross section would

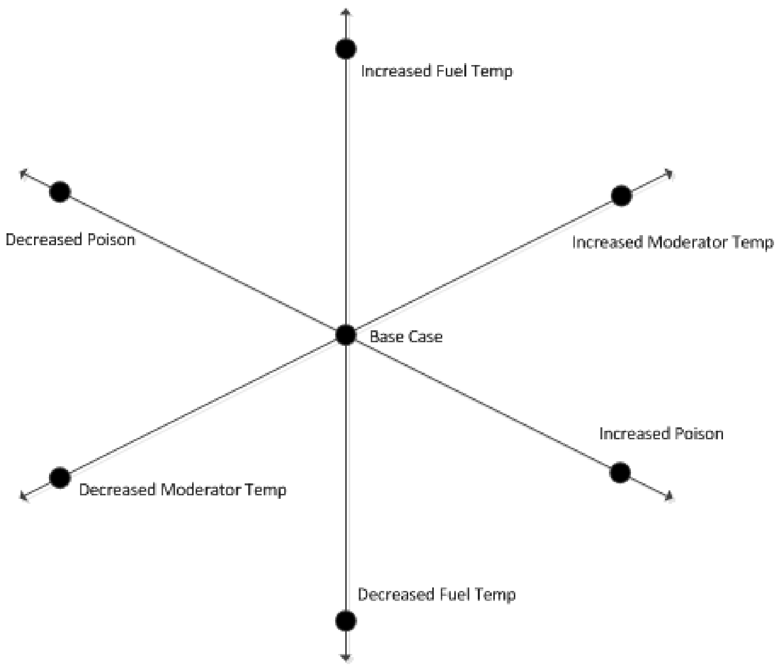

Figure 2.1: A simple case matrix

be interpolated along the line between the two points. Clearly, though, a one-dimensional, degenerate case matrix is of almost no value; to properly account for the influence of each of the physical core parameters on the homogenized cross sections, the case matrix must span the dimensionality of the parameter space. Therefore, for N physical core parameters used in the diffusion code, an N-dimensional interpolation will be required.

The simplest case matrix structure of practical use is to start at a base case and then vary each parameter individually, with one or more variations ("branches") in each parameter. For an operational state point that differs in several parameters from the base case, the effect of each variation is determined by interpolation or regression, and the effects are summed linearly to determine the cross section at that state point:

$$\Sigma_x = \Sigma_{x,base} + \sum_{i=1}^{N} \delta\Sigma_{x,i}$$

where $\Sigma_{x,base}$ is the base case cross section, $\delta\Sigma_{x,i}$ is the cross section difference calculated due to the variation in the ith parameter, and Σ_x is the output cross section at the operational state point.

In particular, if linear interpolation is used,

$$\Sigma_x = \Sigma_{x,base} + \sum_{i=1}^{N} \frac{x_i - x_{i,base}}{x_{i,branch} - x_{i,base}} \, \delta\Sigma_{i,branch}$$

where $\delta\Sigma_{i,branch}$ is the change in cross section from the base case to the ith branch case, the x_i's are the physical core parameters at the state point in question, and the $x_{i,base}$'s and $x_{i,branch}$'s are the values of those parameters at the base and branch points, respectively.

Case matrices that treat separately each change in cross section due to variation in a single physical core parameter are very common. The NESTLE code currently uses this sort of case matrix structure, where the changes in cross section due to each parameter are determined using a polynomial regression, then summed linearly. Similar equations are given by [5] for the cross section dependencies of both Boiling Water Reactors (BWRs) and Pressurized Water Reactors (PWRs). However, since the changes in cross section due to parameter changes occur due to a variety of physical effects (e.g., Doppler broadening, varying neutron moderation), there is no intrinsic reason to expect the cross section changes to combine in a linear manner; indeed, there is often a non-negligible interdependence of the thermal-hydraulic parameters [8]. For this reason, some codes include case matrix "cross-terms" that provide lattice physics cross sections at state points that vary from the base case in multiple parameters simultaneously [9]. These cross-terms result in a case matrix that has an N-dimensional grid or tree structure.

To reduce the number of grid points required to be provided by the lattice physics code, it is also possible to construct a *sparse grid* in which only a subset of grid points are used. As would be expected, a smaller subset of nodes allows for faster computation and less memory requirements, but with increases in expected interpolation error [10].

A further intriguing possibility is that the case matrix could consist of branch points scattered irregularly (i.e., not in a grid pattern) throughout the parameter space of interest. This case matrix structure may have some ability to account for cross-term effects while requiring fewer total branch cases to be run in the lattice physics input.

Given this diversity of potential case matrices, it is desirable that the selected interpolation method have the flexibility to utilize the existing NESTLE case matrix structure, as well as other, more complex structures.

2.4 Interpolation Techniques

2.4.1 Overview

A function that is known only from a set of numerical values at a given set of points can be used to produce a formula that can be evaluated at any arbitrary point using several different methods. Three widely used methods are

- polynomial interpolation,
- least-squares regression, and
- piecewise splines.

Each of these methods will be discussed in turn below.

Polynomial interpolation consists of constructing, for a set of n known points (or "knots"), a polynomial of degree n-1 which exactly reproduces values of the functions at the knots, while giving a polynomial curve between knot points. Although the method is conceptually simple and easy to evaluate, polynomial interpolation suffers from two major defects. First, extension to multiple dimensions is complex and unclear [11], and therefore the ability to account for cross-terms is reduced if this method were used for cross section interpolation. Second and perhaps even more seriously, for large n the interpolating polynomial will often oscillate wildly between the knot points rather than connecting the knot points in a smooth, intuitive manner. This effect, known as Runge's Phenomenon [12], means that polynomial interpolation can be used in practice only with extreme caution.

In least-squares regression, rather than forcing the interpolating function to evaluate exactly to the known values at the knot points, the form of the interpolating function (or, more properly in this case, the regression function) is taken as a given, with a number of parameters to be tuned to best match the data. These parameters of set to the best fit to the data by minimizing the square of the differences between the known function values at knot points and the regression values at those same points. Polynomials are often used as the regression function, but other functions are also commonly used (e.g., logarithmics, exponetials, and trigonometrics). The least-squares regression procedure can be used deterministically with a regression function using a linear combination of tuning parameters, and even non-linear combinations can be effectively used by applying an iterative method. A regression approach is currently used in NESTLE to determine cross sections, as is discussed below. In general, regression functions are simple to evaluate and avoid Runge's Phenomenon, but they are limited in their ability to reporduce cross-term effects and in their ability to reproduce complex or discontinuous underlying behavior.

The term "spline" referred originally to a thin strip (or lath) of wood or other flexible material that was used by drafters to create smooth curves between a set of points (see Figure 2.2). Lead weights (known as "dogs" or "ducks") were used to anchor the spline at the specified knot points, and the curve was traced along the course of the spline. By extension, the word is used to refer to the mathematical practice of creating an interpolating function using piecewise polynomials, which gives a similar results to a curve drawn using a physical spline [13]. Splines are easily extensible to multiple dimensions, fairly simple to evaluate, and produce plausible curves for even the most complex underlying phenomena, provided a sufficient number

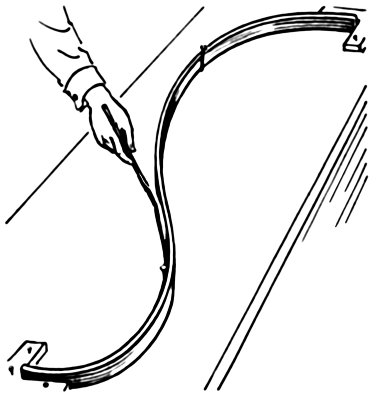

Figure 2.2: A drafting spline in use. [14]

of knot points are available. As described below, a type of multidimensional spline interpolation is implemented in this work.

2.4.2 NESTLE: Current Approach; Goals for Improvement

Currently, the NESTLE code implements a polynomial fit of cross section data, using up to seven polynomial terms as follows [7]:

$$\hat{\Sigma}_{xg} = a_{1_{xg}} + \sum_{n=1}^{2} a_{(n+1)_{xg}} (\Delta\rho_c)^n + a_{4_{xg}} \Delta T_c + a_{5_{xg}} \Delta\sqrt{T_{F_{eff}}} + \sum_{n=1}^{3} a_{(n+5)_{xg}} (\Delta N_{sp})^n$$

where $\hat{\Sigma}_{xg}$ is the evaluated cross section and the $a_{i_{xg}}$ terms are the seven polynomial fit terms. Taking the terms on the right hand side one by one, they represent:

- $a_{1_{xg}}$, the base case cross section (constant value)
- $\sum_{n=1}^{2} a_{(n+1)_{xg}} (\Delta\rho_c)^n$, the change in cross section due to coolant density changes (up to quadratic order)
- $a_{4_{xg}} \Delta T_c$, the change in cross section due to coolant temperature changes (linear order)

- $a_{5_{xg}}\Delta\sqrt{T_{F_{eff}}}$, the change in cross section due to fuel temperature changes (linear order with respect to the square root of the fuel temperature), and
- $\sum_{n=1}^{3} a_{(n+5)_{xg}}(\Delta N_{sp})^n$, the change in cross section due to soluble poison concentration changes (up to cubic order)

In current usage, the set of cross sections obtained from the lattice physics code is run through one of several regression-fitting programs collectively known as L2X (Lattice to XML) to produce an XML file which is read in to NESTLE through a X2N (XML to NESTLE) routine. For example, the process for converting output from the SERPENT 2 code to NESTLE format is shown in Figure 2.3.

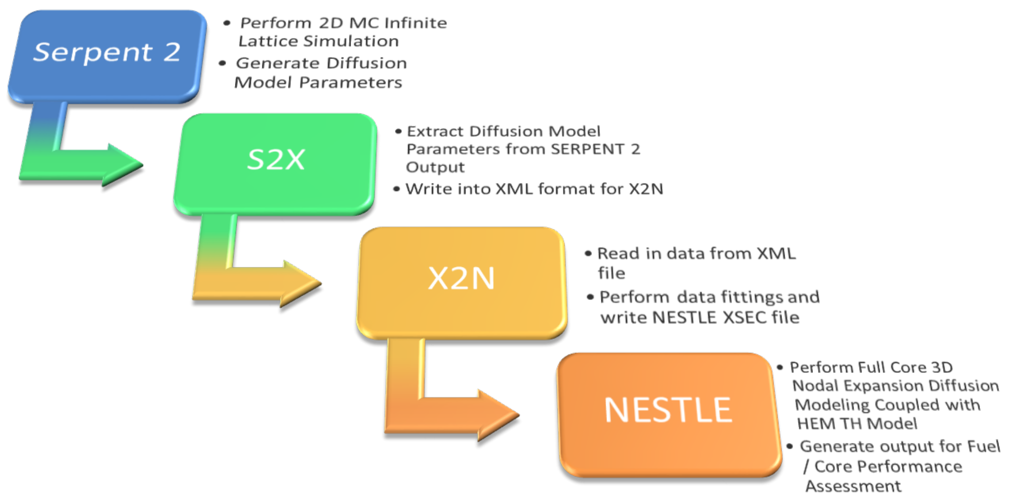
Serpent 2
• Perform 2D MC Infinite Lattice Simulation
• Generate Diffusion Model Parameters
S2X
• Extract Diffusion Model Parameters from SERPENT 2 Output
• Write into XML format for X2N
X2N
• Read in data from XML file
• Perform data fittings and write NESTLE XSEC file
NESTLE
• Perform Full Core 3D Nodal Expansion Diffusion Modeling Coupled with HEM TH Model
• Generate output for Fuel / Core Performance Assessment

Figure 2.3: Serpent 2 to NESTLE cross section file conversion process [15]

The existing polynomial regression routines work well in most typical current uses of NESTLE, when case matrix cross terms are not needed and when the cross section dependencies are fairly smooth. These existing routines can be left in place. However, additional flexibility in NESTLE's cross section interpolation would be a welcome improvement. The new interpolation routine should be able to deal with a wide variety of case matrices, not only case matrices structured especially for NESTLE.

The current work was undertaken to develop a new cross section interpolation routine, with the priorities being:

1. Flexibility (accepting a wide variety of case matrices)
2. Extensibility (able to incorporate additional parameters in the future)
3. Speed (no excessive increase in processing time)

2.4.3 Scattered Data Interpolation

Given the many potential case matrix structures and the desire for compatibility with both newly-developed and previously existing lattice physics output files for NESTLE, the most straightforward approach to interpolation is to put no *a priori* requirements on the case matrix sturcture. This type of interpolation, with no assumptions about the spacing or density of the known points ("knots"), is known as *scattered data interpolation* [16].

In his classic review article, Franke [16] provides a wide-ranging listing of scattered data interpolation methods, along with classifications measuring both numerical accuracy, visual smoothness, and implementation concerns such as storage (memory) requirements and complexity. Of the methods tested by Franke, his method #23, Duchon's Thin Plate Splines (TPS), stands out as well-suited for our purposes for several reasons:

- The method receives the highest rating ("A") for complexity, accuracy, and visual smoothness.
- The method does not require external "tuning" parameters, and so can be implemented without foreknowledge of the arrangement and spacing of the knots.

- It is a global method and can be implemented fairly simply using linear algebra operations for both developing the interpolant and reading the interpolated value.

Franke also lists several disadvantages of the TPS method (its speed is rated "C/D"; the storage requirements are quadratic with the number of knots, while many other methods are linear). But, these disadvantages are relatively minor for use in a diffusion code such as NESTLE, because they come in to play primarily with huge matrices (i.e., thousands of knots), which is rarely applicable to case matrices used in diffusion codes.

2.4.4 Thin Plate Splines and Radial Basis Function Interpolation

Thin Plate Splines

As described above, the word "spline" originally meant a thin flexible strip used to draw a smooth curve between points. From the perspective of Bernoulli-Euler beam theory, this arrangement, where the spline is loaded with point loads at the knot locations, results in a piecewise, linearly varying bending moment. Since bending moment is the second derivative of displacement, the shape of a physical spline in use can be represented mathematically by a piecewise cubic polynomial. These piecewise interpolating cubic polynomials are known as cubic splines, and are ubiquitous in one dimensional interpolation. Note too that, in accordance with Castigliano's Theorem, this displacement represents a configuration that minimizes the internal strain energy in the beam [13][17].

The name "thin plate splines" is used because, just as a one-dimensional cubic spline is a representation of the bending of a lath to fit the knot points, in two dimensions a TPS is equivalent to pressing a thin sheet of metal or other flexible material over a scattered array of knot points of varying elevations. Again, in accordance with Castigliano's Theorem, the sheet forms to a shape which minimizes its internal strain energy. Up to a constant multiplicative factor, for a two-dimensional rectangular sheet or plate, the internal strain energy is given by [17],

$$V = \int\int\left(\left(\frac{\partial^2\varphi}{\partial x^2}\right)^2 + \left(\frac{\partial^2\varphi}{\partial y^2}\right)^2 + 2\left(\frac{\partial^2\varphi}{\partial x\partial y}\right)^2\right) dx\ dy$$

where $\varphi(x, y)$ is the Airy stress function. For a physical sheet held to a certain set of points of fixed displacement, the shape of the sheet will be such that $\frac{\partial V}{\partial x_i} = 0$ for all x_i. Just as in the case of the one dimensional cubic spline, TPS interpolation is mathematically equivalent to simply replacing the stress function with the dependent variable of interest.

TPS interpolation was first developed in 1972 by Harder and Desmarais [18] under the name of surface splines. They applied the method in two dimensions, for use in aeronautical engineering. Duchon [19] developed the method from a mathematical (as opposed to engineering) perspective, and noted in the same paper that the method was also directly applicable to higher dimensions.

Looking again at the equation for internal strain energy in a thin plate given above, the energy-minimizing solution in two dimensions for n points is given by Duchon [20]:

$$s(x, y) = \sum_{i=1}^{n} \lambda_i \left[(x - x_i)^2 + (y - y_i)^2\right] \ln\left[\sqrt{(x - x_i)^2 + (y - y_i)^2}\right] + c_0 + c_1 x + c_2 y$$

which is clearly generalizable to N dimensions as

$$s(\vec{x}) = \sum_{i=1}^{n} \lambda_i ||\vec{x} - \vec{x_i}||^2 \ln ||\vec{x} - \vec{x_i}|| + c_0 + \vec{c_1} \cdot \vec{x}$$

or, in terms of the distances r_i between x and each x_i,

$$s(\vec{x}) = \sum_{i=1}^{n} \lambda_i {r_i}^2 \ln r_i + c_0 + \vec{c_1} \cdot \vec{x} \tag{2.1}$$

where the λ_i terms are the fitted spline weights and the c_i *polynomial terms* constitute a multilinear best fit to the data. If the physical analog of the fitted spline weights is forming a thin flexible plate over a set of fixed points, the physical analog of the polynomial terms would be lifting and tilting a thin plate to minimize the amount of displacement required, rather than holding the plate rigidly flat along the $z = 0$ plane.

The factors λ_i and c_i can be found by solving the linear system [21]

$$\left[\begin{array}{c:c} \Phi\left(\vec{x_i}, \vec{x_j}\right) & \underline{\underline{P}} \\ \hdashline \underline{\underline{P}}^T & \underline{\underline{0}} \end{array}\right] \left[\begin{array}{c} \vec{\lambda} \\ \hdashline \vec{c} \end{array}\right] = \left[\begin{array}{c} f\left(\vec{x_i}\right) \\ \hdashline \vec{0} \end{array}\right]$$

where $\Phi\left(\vec{x_i}, \vec{x_j}\right)$ is the *kernel function* $\Phi\left(r\right) = r^2 \ln r$ applied to each pair of n knot points, and each row $\vec{P_i}$ of $\underline{\underline{P}}$ is the $\{n+1\}$-dimensional vector $[1, \vec{x_i}]$ (the dashed lines represent different regions which are concatenated into a single matrix or vector). Once the λ_i and c_i are determined, the interpolated value can be determined using Equation 2.1 above.

Radial Basis Functions; Multiquadrics

TPS interpolation belongs to a larger class of interpolation methods known as Global Basis Function methods, where a kernel function $\Phi\left(\vec{x}\right)$ is chosen and coefficients λ_i are then determined such that the function $s\left(\vec{x}\right) = \sum_i \lambda_i \Phi\left(\vec{x}\right)$ interpolates the data points [16]. In particular, if the kernel function is selected to be radially symmetric, i.e., $\Phi\left(\vec{x}\right) = \Phi\left(||\vec{x}||\right) = \Phi\left(r\right)$, then the interpolation method is known as a Radial Basis Function (RBF) method. For example, neglecting the polynomial terms, TPS interpolation is an RBF method with kernel $\Phi\left(r\right) = r^2 \ln r$

In addition to TPS interpolation, another widely used RBF interpolation method is the "Multiquadric" method of Hardy [22]. For Multiquadric interpolation, the kernel function is $\Phi\left(r\right) = \sqrt{r^2 + r_0^2}$ [22], while for Reciprocal (or Inverse) Multiquadric interpolation, the kernel function is $\Phi\left(r\right) = \left(r^2 + r_0^2\right)^{-1/2}$ [16], where in both cases r_0 is a scaling parameter selected by the user. Although these RBF interpolations do not have the clear physical interpretation of thin plate splines, they do yield resulting interpolation results that are as good as TPS interpolation in terms of complexity, accuracy, and visual evaluation [16]. Gaussian kernels of the form $\Phi(r) = \exp -\frac{r^2}{r_0^2}$ are also used [23].

Note that, since TPS and other RBF interpolation methods differ only in the definition of the kernel function, a general computer implementation of RBF interpolation can be constructed such that the kernel functions are easily swapped and modified. Doing this allows the programmer to select the type of RBF that produces the best results for the problem at hand with little extra effort.

2.5 Implementation in NESTLE

2.5.1 Selection and Modification of RBF Library

For NESTLE, the desired interpolating subroutine needed to be written for interpolation in four to six dimensions: the effects of fuel temperature, moderator temperature, moderator density, and soluble poison concentration are always included; the effects of control rod state and fuel burnup may be either included in the main interpolation or calculated separately. The use of high quality, well-established libraries of Fortran or C software was considered preferable to development of custom routines.

Given the kernel interchangeability of the various RBF interpolation methods, it was determined to begin work using TPS interpolation, then to test other RBF interpolation methods for any available improvements in accuracy or robustness. The program `RBF_INTERP_ND`—*Multidimensional Interpolation with Radial Basis Functions* by John Burkardt [23], a publicly-available, high-quality routine for RBF interpolation, was selected for use.

The program `RBF_INTERP_ND` does not contain a polynomial term, and preliminary testing on a prototype cross-section interpolation program made clear that a polynomial term is needed to yield accurate interpolation results for this application. Therefore, the `RBF_INTERP_ND` source code was modified to include this polynomial term. The library was also modified to use the LAPACK linear algebra library [24], rather than using standalone linear algebra code.

2.5.2 Interpolation Approach

Given the inherent flexibility of RBF interpolation, a variety of approaches to interpolation were tested. Some questions to be answered were:

- Should the case matrix parameters be scaled, and if so, how?
- Should control rod input be taken as an interpolation variable, or should control rod in and control rod out be treated as two different case matrices?
- Should the thin plate spline kernel continue to be used, or does another RBF kernel produce better results?
- Should the interpolation be carried out in one single step, or should the interpolation be broken into a multi-step process?

A variety of prototype interpolation functions were written and tested by comparison against cross section values calculated by lattice physics codes. This testing showed that the case matrix parameters should be re-scaled to a dimensionless value between zero and one to produce the most reliable interpolation, a process known as unit basis renormalization. The parameters vary in typical magnitude in their traditionally-used units; for example, temperatures are often hundreds of degrees Fahrenheit, while densities expressed as specific gravities are an order of magnitude or more smaller. If rescaling is not performed, the calculation of radial distance in parameter space can give undue weight to variation in some parameters while neglecting others. Keeping the case matrix confined to the unit hypercube in parameter space decreases this risk.

The magnitude of the control rod effect on the cross sections is typically much greater than the magnitude of the effects of the other parameters. Because of this, the impact of case matrix points with opposite control rod state is minimal within an interpolation. The various control rod states are therefore broken into into separate case matrices and the interpolation is performed independently. This reduces the size of the case and interpolation matrices by a factor of two, while causing little or no loss of accuracy.

A comparison of the various radial basis function kernels is shown in Figure 2.4, for a simple test problem with thermal-hydraulic feedback enabled, at various arbitrary points in parameter and burnup space. Since the Multiquadric kernel often produces the best fit ([16] describes Inverse Multiquadrics and TPS as producing fits "nearly as good" as Multiquadrics), and because it visually appears to stay close to the average of the kernels while some other methods produce occasional outliers, the Multiquadric kernel has been used as the default in the NESTLE implementation. However, the TPS kernel, which requires no scaling parameter [16] and therefore may be more robust to a variety of case matrices, has been left in the source code and can easily be reinstated. The Inverse Multiquadric and Gaussian kernels are also left in the source code (they were included in the `RBF_INTERP_ND` code and there was little need to remove them), though these kernels were not found to give any significant improvement in accuracy or performance across the range of parameters investigated. Note that, except in one outlying case, the results of TPS, Multiquadrics, and Inverse Multiquadrics were similar in each case.

Finally, several approaches to interpolation order were tried. Looking at the results qualitatively, the most satisfactory results were obtained by interpolating over all physical state parameters in a single step, then performing a separate, one-dimensional interpolation for burnup. Most likely, this two-step interpolation process produces better results because a typical lattice physics input file contains a considerably larger number of burnup steps than the number of branches in the various local reactor conditions.

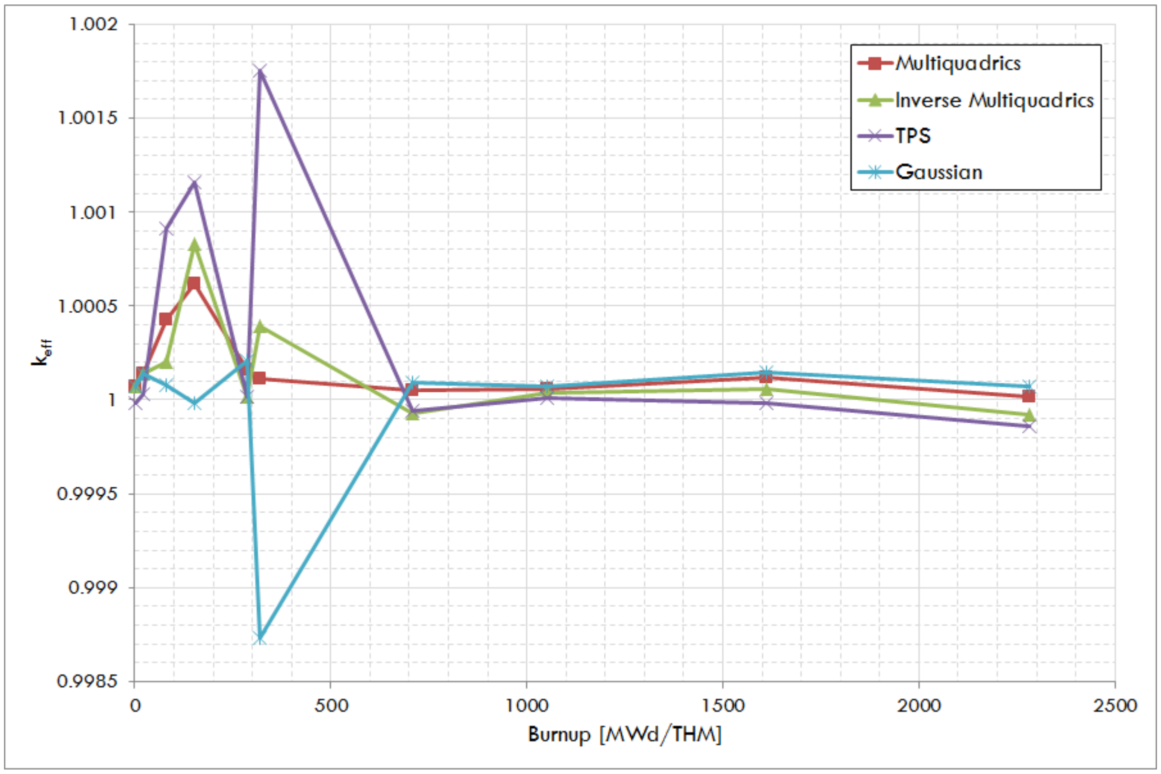

Figure 2.4: Comparison of test problem results using several RBF kernels

Performing the interpolation in this order may seem counterintuitive, since the local reactor conditions change during thermal-hydraulic feedback calculations while the burnup remains constant. But, because a system of linear equations can be solved once for any number of right-hand sides with little additional computation required, the interpolation can be performed for all burnup steps simultaneously. A simple one-dimensional cubic spline interpolation can then be used to calculate the cross section of interest at a given burnup value. The cubic interpolation routines included in the GNU Scientific Library [25] were used in the current work, with provision for simple linear extrapolation for material burnups outside the range of the lattice physic input.

2.5.3 Results

A new XML format for cross section input, developed by Cole Gentry [15], was chosen as the basis for the new RBF interpolator. An XML schema was developed for the new input format, and code was written to validate the XML file and import the data structure into NESTLE.

The additional interpolation capability was incorporated into NESTLE by creating two new modules. The module `case_matrix_mod` reads the XML file, develops the case matrix of branch states, and calculates the RBF and polynomial weighting terms. The `get_cross_section_mod` module is a function that returns a requested cross section at a given point in case matrix parameter space and burnup. The modified `RBF_INTERP_ND` module and the GSL are used for computation as described above, and the libxml2 [26] and FoX [27] libraries are used for XML processing.

A simple test case was run without thermal-hydraulic feedback and with the values of physical parameters selected arbitrarily away from knot points. A comparison of the values obtained with the RBF interpolator and the values obtained using the existing seven term polynomial regression method are compared in Table 2.2.

Table 2.1: Comparison of RBF interpolator and existing regressions, without thermal-hydraulic feedback

Burnup [MWd/THM]	Soluble Poison [ppm]	k_{eff} using RBF Interpolator	k_{eff} using Regressions	Difference [pcm]
1	200	1.03099	1.03086	+12.61
100	200	1.02713	1.02702	+10.71
500	200	1.02964	1.02951	+12.63
1000	200	1.03332	1.03319	+12.58
2000	200	1.03289	1.03278	+10.65
5000*	200	1.01496	1.07395	-5647

*Extrapolation beyond lattice physics file burnup values

The maximum difference for burnups within the range of the lattice physics file is not a significant deviation and verifies that the RBF interpolator closely reproduces the behavior of the existing regressions for the same existing input files. The deviation of 5647 pcm at a burnup of 5000 MWd/THM *is* a significant deviation, but the value of k_{eff} calculated with the RBF interpolator appears more reasonable than the value using the polynomial regression routines. Although accurate results should never be expected when extrapolating significantly outside of the lattice physics data, this last result shows that the use of multi-dimensional splines such as the RBF interpolator should produce a more realistic extrapolation than polynomial regression. This is likely to prove useful when a slight extrapolation beyond the lattice physics data is required.

When thermal-hydraulic feedback is included, the differences increase substantially. The results are shown in Table 2.2. Neglecting the extrapolation at 5000 MWd/THM, the difference is as high as 1017 pcm. Nonetheless, the RBF interpolator results appear to be more consistent with the calculated values of k_{inf} from the lattice physics input than the polynomial regression results. It is possible that the thermal-hydraulic feedback is driving an instantaneous state point beyond the range of the lattice physics file. The instability of polynomial regressions in extrapolation may then be causing the difference.

Table 2.2: Comparison of RBF interpolator and existing regressions, with thermal-hydraulic feedback

Burnup [MWd/THM]	Poison [ppm]	k_{eff} using RBF Interpolator	k_{eff} using Regressions	Difference [pcm]
1	318	1.00349	0.99515	-834.6
100	300	0.99998	0.99671	-327.5
500	310	1.00253	0.99679	-574.2
1000	330	1.00630	0.99612	-1017
2000	330	1.00662	0.99771	-889.1
5000*	280	1.04012	0.99701	-4232

*Extrapolation beyond lattice physics file burnup values

Chapter 3

N-Group Extension

3.1 Introduction

Beginning with the development of the PDQ code sixty years ago [28], multigroup neutron diffusion calculations based upon the finite difference method have been and remain the industry standard for steady-state reactor simulations. Traditionally, two or four energy groups have been used for light water reactor calculations [1]. Prior to this work, NESTLE had the capability of operating with either two or four energy groups.

There are many reactor calculations, however, for which the capability of operating with a number of energy groups greater than four proves useful. Gas-cooled reactors are typically evaluated using seven to nine energy groups, while fast reactor calculations may require twenty groups or more [1]. The trend in recent years has been to perform reactor calculations with increasing numbers of energy groups.

The physical and mathematical basis behind the generalization of fixed-group finite-difference diffusion theory into arbitrary number of energy groups N is a well-known problem [1]. This work deals specifically with the generalization of the Nodal Expansion Method (NEM), and the methodological and computational challenges encountered and solved during the implementation of this generalization in NESTLE.

3.2 Theory

3.2.1 Neutron Diffusion and the Nodal Expansion Method

As described in Chapter 1, a careful accounting of the creation and absorption of neutrons in a reactor allows the derivation of the neutron transport equation [1], given in its most general form (and with functional dependencies of position, direction, and energy suppressed for clarity) as:

$$\frac{\partial n}{\partial t} + v\hat{\Omega}\cdot\nabla n + v\Sigma_t n = \int_{4\pi}\int_0^\infty \left[v'\Sigma_s\left(E' \to E, \hat{\Omega}' \to \hat{\Omega}\right) n + s\right] dE' d\hat{\Omega}'$$

Implementing energy grouping and the diffusion approximation, as described in Chapter 1, as well as integrating out time dependence, produces the steady-state multigroup neutron diffusion equation:

$$-\nabla \cdot D_g \nabla \phi_g + \Sigma_g^r \phi_g = \sum_{g'=1}^{G} \Sigma_{gg'}^s \phi_{g'} + \frac{\chi_g}{k} \sum_{g'=1}^{G} \nu_{g'} \Sigma_{g'}^f \phi_{g'}$$

This equation is a partial differential equation (PDE) with up to three independent spatial variables as well as set number of energy groups. Like many PDE systems encountered in engineering analysis, closed-form analytical solutions to the multigroup diffusion equation are rarely available except in a limited number of simple geometries. Instead, numerical methods are typically used. Many standard techniques for solving PDEs can be applied to the neutron diffusion equation, including the finite difference method (FDM), the finite element method, and the Monte Carlo method. Historically, the most common approach has been the application of FDM and various refinements thereof. Nodal methods are one such refinement.

Nodal methods were developed to reduce the computational expense required in direct application of the FDM, where a very fine mesh size is required to obtain accurate simulations on the scale of a reactor. Nodal methods consist of dividing the reactor or system of interest into a number of spatial nodes, then integrating the diffusion equation over each node [29]. The technique of spatial homogenization, described in Chapter 1, is used to obtain the equivalent nuclear and physical properties at each node. In exchange for using a courser mesh than would be required in direct FDM, the direct FDM assumption of a flux varying linearly between node points must be replaced with a more sophisticated model. Polynomial methods, including the NEM used in NESTLE, involve representing the flux profile in each node as a sum of basis polynomials, i.e., for the ith node in the x-direction [7, 30],

$$\phi_g^i(x) = \overline{\phi}_g^i + \sum_{n=1}^{N} a_{gxn}^i f_n(x)$$

Clearly, since the basis polynomials are adjustments to the node-averaged flux, they must be selected to be orthogonal and to integrate to zero over the width of the node. For NEM, the following quartic polynomials are used as basis functions [29]:

$$\begin{aligned}
f_1(x) &= \frac{x}{\Delta x} \equiv \xi \\
f_2(x) &= 3\xi^2 - \frac{1}{4} \\
f_3(x) &= \xi\left(\xi - \frac{1}{2}\right)\left(\xi + \frac{1}{2}\right) \\
f_4(x) &= \left(\xi^2 - \frac{1}{20}\right)\left(\xi - \frac{1}{2}\right)\left(\xi + \frac{1}{2}\right)
\end{aligned}$$

The lower order expansion coefficients (a_{gx1}^i and a_{gx2}^i) terms can be obtained by simple physical constraints: balance of current densities over the node surface and continuity of flux and partial current density at the node boundaries. For the higher order expansion coefficients (a_{gx2}^i and a_{gx3}^i), however, additional constraints are needed. These are provided by applying the method of weighted residuals; in particular, the scheme known as method of moments weighting [29] [31].

3.2.2 Two-Node Problem

The heart of the non-linear iterative application of the NEM is that, during iteration of the standard coarse-mesh finite difference algorithm, to periodically solve the *two-node problem* for every interface between two nodes. For N energy groups, solving the two-node problem requires solving an $8N \times 8N$ linear system at each node interface. The $8N$ equations are those described above. Specifically, for each energy group,

- Nodal neutron balance at each of the two nodes ($2N$ equations)
- Continuity of current density at node boundary ($1N$ equations)
- Continuity (or known discontinuity) of flux at node boundary ($1N$ equations)
- Weighted residuals first moment for each node ($2N$ equations)
- Weighted residuals second moment for each node ($2N$ equations)

This linear system is not especially sparse compared to typical engineering linear algebra applications: it has a density of $\frac{N+2}{8N}$, giving a maximum density of $\frac{3}{8}$ for $N = 1$, a density of $\frac{1}{4}$ when $N = 2$, and a minimum density limit of $\frac{1}{8}$ as the number of groups increases. Additionally, although many of the non-zero entries appear near the diagonal, the system does not have a diagonal or banded structure that can be easily exploited. The structure of this matrix is shown in Figure 3.1 for $N = 2$.

NESTLE currently uses explicit solution routines for this $8N \times 8N$ linear system, which were generated using symbolic manipulation software specifically for the $N = 2$ and $N = 4$ cases [30]. The non-zero terms of the $N \times N$ matrix is stored internally in a one-dimensional packed storage array. This approach is very efficient in both computation and storage, but it limits the code to using either two or four groups, and presents few opportunities for optimization using the multi-threading and single instruction multiple data (SIMD) capabilities of modern computers.

3.3 Current Work

As an alternative to the explicit solvers already included in NESTLE, an additional set of solvers was added as part of the current work [32]. These new solvers call the general matrix solution routines of the LAPACK linear algebra library [24]. The use of LAPACK provides several advantages over the explicit solution routines:

- The solution of the NEM system is no longer limited to $N = 2, 4$.
- LAPACK utilizes, as much as possible, the Basic Linear Algebra Subprograms (BLAS) [33], for which a variety of highly optimized implementations are available for various computer architectures (e.g., [34, 35]), exploiting the multi-core nature of modern processors, chip caches, and SIMD instruction sets.
- Parallel computing paradigms including distributed memory and accelerators can be easily utilized by popular derivatives of LAPACK, i.e. ScaLAPACK, PLASMA, and MAGMA.

However, the use of LAPACK also has some disadvantages:

- The use of general linear solve routines means the coefficients are stored as a dense matrix. This increases storage requirements, although for $N \lesssim 50$ the storage is still considerably less than the typical L2 cache of modern processors.
- In non-parallel, non-SIMD operation, the general solution routines are unlikely to be as fast as the tailor-made explicit solvers.

Because of this last disadvantage, the use of the explicit solvers has been maintained as an option for two and four group problems.

3.4 Results

After the implementation of the changes described above, lattice physics cross section files were generated for a simple test case, and the general LAPACK solvers were

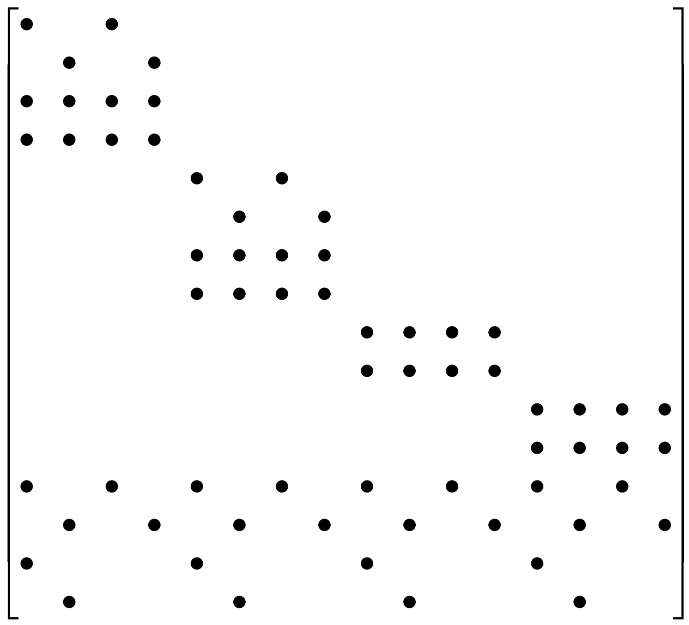

Figure 3.1: Non-zero entries in NEM Matrix ($N = 2$)

confirmed to work for test cases with various numbers of neutron energy groups. The results are shown in Table 3.1.

As expected, in a test case for $N = 2$, using a single computer core and no optimized BLAS library, the use of the general LAPACK solvers was slightly slower (average execution time of 0.63 s vs. 0.61 s, an increase of just over 3%). This difference is of little significance, and the use of additional cores or optimized BLAS routines may further improve the performance of the LAPACK solvers. However, given the small size of the two-node matrix for $N = 2$ and 4, the difference is likely to be commensurately small. Therefore, the code has been written to use the existing direct solvers for two and four group calculations unless directed via a command-line switch to use the LAPACK general solvers.

For the cases $N = 2$ and $N = 4$, when results could be compared directly, the calculated values for k_{eff} determined using the general solvers were identical (to round-off precision) to those calculated for direct solvers.

As can be seen in Figure 3.2, the processing time required for a given N appears to scale closely with the theoretical complexity of $\mathcal{O}(\frac{N^3}{3} + N^2)$ for Gaussian elimination

Table 3.1: Comparison of processing time and memory usage

	Processing Time [s]	Memory Usage [kB]	Calculated k_{eff}
$N = 2$ (prior version)	0.48	26868	1.00032
$N = 4$ (prior version)	0.60	27936	1.00042
$N = 2$ (explicit solver)	0.39	26756	1.00032
$N = 2$ (LAPACK solver)	0.41	26980	1.00032
$N = 3$	0.51	27512	0.99991
$N = 4$ (explicit solver)	0.56	27968	1.00042
$N = 4$ (LAPACK solver)	0.60	28232	1.00042
$N = 12$	9.07	35084	1.00002
$N = 18$	17.80	43028	1.00035
$N = 25$	45.84	55428	1.00045
$N = 40$	168.41	91816	1.00013
$N = 69$	1002.23	172204	0.99690

with scaled partial pivoting [36]. This is consistent with the method used by the selected LAPACK routines `dgetrf` and `dgetrs`. These routines implement lower-upper (LU) decomposition, which is computationally equivalent to Gaussian elimination.

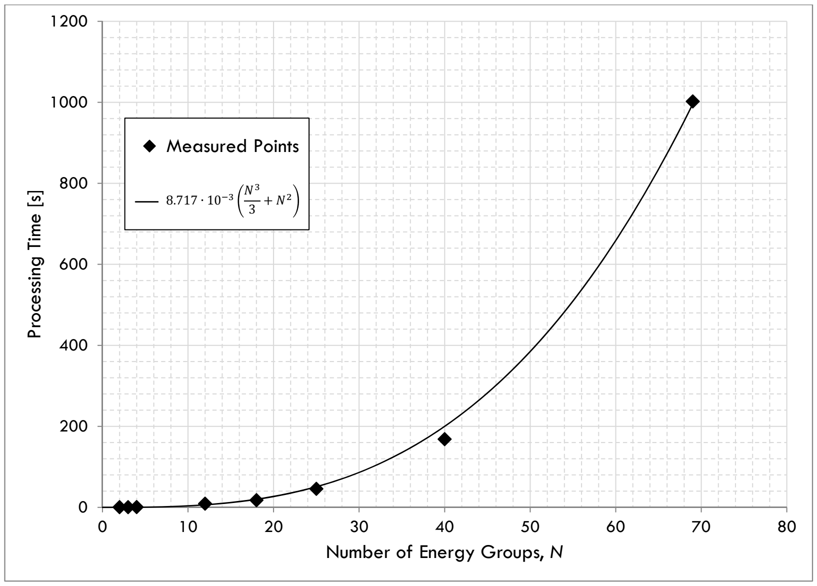

Figure 3.2: Processing time as a function of number of energy groups

Chapter 4

Conclusions and Future Work

4.1 Summary

Improved cross section interpolation and arbitrary N-group extension have been implemented in the NESTLE code without adversely affecting the existing code base and without breaking backward compatibility of input files. A variety of multidimensional interpolation methods potentially applicable to this field were examined, and among these the radial basis function interpolation method known as Hardy Multiquadrics was selected. This method is a close relative of the thin plate spline method, is extremely flexible in terms of the number and arrangement of lattice physics data points, and is well-suited to implementation in Fortran using robust linear algrebra libraries such as LAPACK and BLAS.

The extension of NESTLE to N-dimensions was accomplished by refactoring the subroutines used to solve the NEM two-node problem, vectorizing when possible, then by using the same linear algebra packages mentioned earlier to solve the linear system. This new capability has been shown to deal successfully with numbers of groups ranging from 2 to 69 while retaining the same accuracy exhibited by the existing NESTLE direct solvers.

4.2 Future Work

Several opportunities exist for future improvements and expansions related to the current work. The radial basis function cross section interpolator was implemented in the existing version of NESTLE, without replacing the existing routines which read and interpret the `NESTLE.XSEC.MACRO` file. The XML lattice physics input file contains nearly all of the same information. A future improvement is to obtain all system parameters currently in the `NESTLE.XSEC.MACRO` file directly from the XML file. This would eliminate the need to generate the `NESTLE.XSEC.MACRO` file with X2N.

The cross section interpolation routines were written to be easily expandable to include additional physical parameters. History effects, where the nuclear properties of a reactor material are dependent not only on their instantaneous physical state but on their service history, are often significant. The future incorporation of fuel temperature history, control rod history, or shutdown cooling effects may improve the accuracy of the code.

The opportunity exists for future work investigating the application sparse linear algebra libraries to the general solver routines developed in this work. In particular, a sparse approach that could take advantage of the specific NEM Matrix structure while utilizing the sparse BLAS and retaining the capability of parallel processing may present opportunities for performance improvement.

In certain cases where existing NESTLE appears to converge only slowly, the N-group formulation has shown a tendency to diverge away from the expected solution towards a non-physical configuration. This issue can be worked around by cutting off iteration before the divergence begins, but future work may be able to resolve the issue, which is likely related to the Chebyshev or Weilandt acceleration techniques used in the NEM solver. It is possible that the methods or parameters currently used are optimized for two- and four-group systems only.

Additionally, there may be additional performance improvement that can be gained through optimization of the routines created as part of this work. Some potential areas for improvement are:

- Vectorization of the `get_cross_section_mod` routine to return a vector of applicable cross sections at each call. The Fortran code could be rewritten in a vectorized form since cross sections are generally obtained one after the other with very little intermediate computation.

- Often identical or very similar physical properties (fuel and moderator temperatures, burnup, etc.) exist at nearby nodes. If the cross section retrieval function can be vectorized, there may be a potential speed improvement where the previous slate of cross sections can be returned directly if the state at the next node is at or very near the same point in parameter space as the previous node.

- The GSL call for the one dimensional cubic spline in the burnup dimension may have significant overhead for a rather simple spline that is only momentarily kept in computer memory. A simplified direct implementation of a similar spline may improve performance by requiring less memory allocation and deallocation.

4.3 Conclusion

The additions of the radial basis function cross section interpolator and N-group capability represent a significant expansion in NESTLE's flexibility and adaptability. These changes are important steps forward for the NESTLE's development, and make NESTLE increasingly applicable to new reactor types beyond light water reactors, and to new fuel types and compositions that may have more complex cross section behavior.

Furthermore, this work has been an exciting opportunity for the author to apply his coursework and to greatly expand his previous knowledge in nuclear reactor systems, in computer programming, in numerical approaches to the solutions of engineering problems. These skills have proven invaluable in both the author's academic and professional work, and will no doubt continue to do so for years to come.

Bibliography

[1] James J. Duderstadt and Louis J. Hamilton. *Nuclear Reactor Analysis*. John Wiley and Sons, Inc., New York, 1976. 3, 7, 25, 26

[2] P.J. Turinsky, R.M.K. Al-Chalabi, P. Engrand, H.N. Sarsour, F.X. Faure, , and W. Guo. Computer code abstract: Nestle. *Nuclear Science and Engineering*, 120(1):72–73, 1995. 4

[3] Nicholas P. Luciano et al. The NESTLE 3D nodal core simulator: Modern reactor models. In *Proc. M&C+SNA+MC (Nashville, TN)*. American Nuclear Society, 2015. 4

[4] Kord S. Smith. Assembly homogenization techniques for light water reactor analysis. *Progress in Nuclear Energy*, 17(3):303–335, 1986. 6

[5] Dave Knott and Akio Yamamoto. *Handbook of Nuclear Engineering – Volume 1: Nuclear Engineering Fundamentals*, chapter 9: Lattice Physics Computations. Springer, Berlin, 2010. 7, 8, 10

[6] D. Want, B.J. Ade, and A. M. Ward. Cross section generation guidelines for TRACE-PARCS, institution = U.S. Nuclear Regulator Commission, number = NUREG/CR-7164, year = 2013, month = June. Technical report. 7, 8

[7] North Carolina State University. *NESTLE Version 5.2.1 Manual – Few-Group Neutron diffusion Equation Solver Utilizing the Nodal Expansion Method for Eigenvalue, Adjoint, Fixed-Source Steady-State and Transient Problems*, July 2003. 8, 13, 27

[8] Justin K. Watson and Konstadin N. Ivanov. Improved cross-section modeling methodology for coupled three-dimensional transient simulations. *Annals of Nuclear Energy*, 29:937–966, 2002. 10

[9] Y. Xu and T. Downar. GenPMAXS – Code for Generating the PARCS Cross Section Interface File PMAXS. Technical Report PU/NE-00-20, Rev. 8, Purdue University School of Nuclear Engineering, November 2006. 10

[10] Danniëll Botes and Pavel M. Bokov. Hierarchical, multilinear representation of few-group cross sections on sprase grids. In *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)*, 2011. 11

[11] Stefano de Marchi. Lectures on multivariate polynomial interpolation. http://www.math.unipd.it/~demarchi/MultInterp/LectureNotesMI.pdf. accessed 2017-06-30. 11

[12] Carl Runge. Über empirische funktionen und die interpolation zwischen äquidistanten ordinaten. *Zeitschrift für Mathematik und Physik*, 46:224–243, 1901. 12

[13] J.H. Ahlberg, E.N. Nilson, and J.L. Walsh. *The Theory of Splines and Their Applications*. Academic Press, New York, 1967. 12, 16

[14] Image produced by Scott Foresman and Company. Public domain. https://commons.wikimedia.org/wiki/File:Spline_(PSF).png. accessed 2017-08-15. 13

[15] Cole Gentry. S2N Quick Reference. unpublished, December 2014. 14, 22

[16] Richard Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38(157):181–200, 1982. 15, 18, 19, 21

[17] Stephen Timoshenko and J.N. Goodier. *Theory of Elasticity*. McGraw-Hill, New York, 1951. 16, 17

[18] Robert L. Harder and Robert N. Desmarais. Interpolation using surface splines. *Journal of Aircraft*, 9(2):189–191, 1972. 17

[19] Jean Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *Revue Française d'Automatique, Informatique, Recherche Opèrationnelle: Analyse Numérique*, 10(3):5–12, 1976. 17

[20] Jean Duchon. *Splines Minimizing Rotation-Invariant Seminorms in Sobolev Spaces*. SpringerVerlag, Berlin, 1977. 17

[21] MJD Powell. Some algorithms for thin plate spline interpolation to functions of two variables. *Advances in Computational Mathematics, New Dehli, India, World Scientific, Singapore*, pages 303–319, 1994. 18

[22] Rolland L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research*, 76(8):1905–1915, 1971. 18

[23] John Burkardt. RBF_INTERP_ND – Multidimensional Interpolation with Radial Basis Functions. `https://people.sc.fsu.edu/~jburkardt/f_src/rbf_interp_nd/rbf_interp_nd.html`, September 2012. 19

[24] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999. 20, 29

[25] Mark Galassi et al. *GNU Scientific Library Reference Manual (3rd Ed.)*. 22

[26] Daniel Veillard. Libxml2: The XML C parser and toolkit of Gnome. `http://xmlsoft.org/`. 23

[27] A Walker. FoX, A Fortran XML Library. http://homepages.see.leeds.ac.uk/~earawa/FoX/, 2014. 23

[28] GG Bilodeau, WR Cadwell, JP Dorsey, JG Fairey, and RS Varga. PDQ – an IDM 704 code to solve the two-dimensional few-group neutron-diffusion equations. Technical Report WAPD-TM-70, Bettis Atomic Power Laboratory, November 1957. 25

[29] R.D. Lawrence. Pogress in nodal methods for the solution of the neutron diffusion and transport equations. *Progress in Nuclear Energy*, 17(3):271–301, 1986. 27

[30] G. Ivan Maldonado. *Non-Linear Nodal Generalized Perturbation Theory within the Framework of PWR In-Core Nuclear Fuel Management Optimization*. PhD thesis, North Carolina State University, 1993. 27, 28

[31] A. Finlayson and L.E. Scriven. The method of weighted residuals—a review. *Applied Mechanics Reviews*, 19(9), September 1966. 27

[32] William Kirkland, Ondrej Chvala, and G. Ivan Maldonado. Generalization of NESTLE into a multi-energy N-group formulation. In *Transactions of the American Nuclear Society*, 2017. 29

[33] Chuck L Lawson, Richard J. Hanson, David R Kincaid, and Fred T. Krogh. Basic linear algebra subprograms for Fortran usage. *ACM Transactions on Mathematical Software (TOMS)*, 5(3):308–323, 1979. 29

[34] Intel Math Kernel Library (MKL). https://software.intel.com/en-us/intel-mkl. accessed 2017-01-25. 29

[35] Zhang Xianyi, Wang Qian, and Werner Saar. OpenBLAS: An optimized BLAS library. http://www.openblas.net/. accessed 2017-01-26. 29

[36] Ward Cheney and David Kincaid. *Numerical Mathematics and Computing*. Thomson Brooks/Cole, Belmont, CA, fifth edition, 2004. 31

Appendices

Appendix A

Code Listing

The following computer codes developed for this work can be found at: https://github.com/willkirkland/masters-thesis.

case_matrix_mod.f90

generalSolver.f90

get_cross_section_mod.f90

rbf_interp_nd.f90

spline_mod.c

x2nSchema.xsd

x2nSchema_new.xsd

xmlValidate.c

In addition to the codes listed, an extensive amount of work was done to the existing NESTLE source code, including a complete refactoring of the NEM solvers and inserting calls to the cross section interpolator. These modifications can be found in the development version of the NESTLE source code.

Vita

William Matthews Kirkland was born in Charleston, W.Va., and raised in Vienna, W.Va. He received his diploma from Parkersburg High School in 2001. In 2005, Will graduated from the University of Kentucky with a B.S.M.E. in Mechanical Engineering with minors in physics and mathematics.

After graduation, he worked as a system engineer at the Y-12 National Security Complex, where he worked in special processing and microwave casting. In 2011, he left Y-12 to work at Oak Ridge National Laboratory as a vacuum systems mechanical engineer for ITER, a multi-national fusion energy reactor.

In the fall of 2014, with support from ORNL he entered the master's program in the Department of Nuclear Engineering at the University of Tennessee, Knoxville.