2-15-2008

The inverse kinetics method and PID compensation of the Annular Core Research Reactor

Benjamin Garnas

The Inverse Kinetics Method and PID Compensation of the Annular Core Research Reactor

by

Benjamin Garnas

B.S. General Engineering, New Mexico Highlands University, 2003

THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2007

Dedication

To my brother, Matthew Garnas, whose untimely death first catalyzed my desire to pursue an advanced degree, to my parents, Gil and Vicki, for their love, support, and encouragement even when it seems Matt and I would never amount to anything, and to God who has been with me every step of the way throughout my life even in the most troubling of times.

Acknowledgments

First and foremost I thank my employer, Sandia National Labs, for funding and the opportunity to further my education. I thank Ray Troughbridge for first helping me contact the ACRR group. I thank Robert Zaring, Lonie Martin, and Ron Farmer for the opportunity develop a controller for their ACRR. Many thanks to Dr. Ed Parma and Dr. Ron Knief, for the time spent working with me to understand the nuclear reactor theory. Additional thanks to Dr. Parma for all the time spent helping me to first developing an ACRR model. I would like to thank my present and past managers Bob Boney, Mary Gonzales, and Rich Kreutzfield for their continued support, flexibility, and patience while I pursued my educational goals. Finally I would like thank Ernie Wilson for covering for my responsibilities at work so that I was able to concentrate full-time on finishing this thesis.

The Inverse Kinetics Method and PID Compensation of the Annular Core Research Reactor

by

Benjamin Garnas

ABSTRACT OF THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2007

The Inverse Kinetics Method and PID Compensation of the Annular Core Research Reactor

by

Benjamin Garnas

B.S. General Engineering, New Mexico Highlands University, 2003

M.S., Electrical Engineering, University of New Mexico, 2007

Abstract

This thesis explores the development of a model describing the Annular Core Research Reactor (ACRR), the application of the inverse kinetics method to calculate the current reactivity level within the ACRR model, and the development of a PID compensator to automatically control the reactivity level within the model and thus controlling the power level.

Contents

List of Figures **xii**

List of Tables **xv**

1 Introduction **1**

1.1 Background 1

1.2 Overview of ACRR 3

1.2.1 Detailed Description of ACRR 4

1.3 Thesis Contribution 7

1.4 Organization of Thesis 7

2 Theory **8**

2.1 Overview 8

2.2 Nuclear Reaction Theory 8

2.2.1 Nuclear Fission 9

2.2.2 Neutron Population 10

2.2.3 Delayed Neutrons 12

2.2.4 Neutron Lifetime and Mean Neutron Generation Time 13

2.3 The Point Reactor Kinetics Equation 14

2.3.1 Derivation of the Inhour Equation 16

2.3.2 Stable Reactor Period 18

2.3.3 The Dollar 19

2.4 The Inverse Kinetics Method 20

2.5 Concluding Remarks 21

3 Reactor Model **22**

3.1 Overview 22

3.2 Solving the Kinetics Equation 23

3.2.1 Validation 24

3.3 Temperature Feedback 26

3.4 Sorting Data 29

3.5 Recreation 30

3.6 Solving the Inverse Equation 35

3.7 Concluding Remarks 39

4 Controller **41**

4.1 Overview 41

4.2 Overview of the Plant Model 41

4.2.1 State-Space Representation 43

4.3 Using the Inverse Method 46

4.4 Developing the PID compensator 51

4.4.1 Creation of the Desired Power vs. Time Path 52

4.4.2 Creation of the Desired Reactivity Path 53

4.4.3 Development of the Plant Input 54

4.4.4 Management of the Power Output 55

4.4.5 Treatment of the Error Signal 55

4.4.6 Control Rod Adjustments 56

4.4.7 Results 56

4.5 Concluding Remarks 70

5 Conclusions **71**

5.1 Review 71

5.2 Future Work 72

Appendices **74**

A Derivation of Numerical Integration Methods **75**

A.1 Trapezoidal 75

A.2 Parabolic 77

B Matlab Script and Function Files **79**

References **109**

List of Figures

1.1 Annular Core Research Reactor (ACRR). 2

1.2 ACRR Core and Central Irradiation Space. 5

1.3 Top of the ACRR Tank. 6

3.1 Power vs. Time for a $0.25 Reactivity Insertion. 24

3.2 Average ACRR Core Temperature vs. Steady-State Power Levels and Resulting Curve Fit. 28

3.3 Average Control Rod Position vs. Time for Steady-State Run 8512 (80%). 30

3.4 Power vs. Time for Steady-State Run 8512 (80%). 32

3.5 Power vs. Time for Steady-State Run 8435 (40%). 33

3.6 Power vs. Time for Steady-State Run 8031. 34

3.7 Delay Kernel Value vs. Time. 36

3.8 Inverse Calculated and Actual Reactivity Values vs. Time. 38

3.9 Error Between Calculated and Actual Reactivity Values vs. Time. . 39

List of Figures

4.1 Plant Block Diagram. 42

4.2 Power vs. Time for Steady-State Run 8512 (80%). 47

4.3 Actual and Simulation Effective Reactivity Input for Steady-State Run 8512 (80%). 48

4.4 Actual and Recreated Power Levels for Steady-State Run 8512 (80%). 49

4.5 Percent Error between Actual and Recreated Power Levels vs. Time. 50

4.6 Plant and Controller Block Diagram. 51

4.7 Desired Power vs. Time. 52

4.8 Desired Reactivity vs. Time. 53

4.9 Predetermined Control Rod Input. 54

4.10 Desired Power Level for an 80% Test Run (1.92 MW). 57

4.11 Desired and Actual Reactivity Levels vs. Time for 80% Run (1.92MW). 58

4.12 Reactivity Error vs. Time for 80% Run (1.92 MW). 59

4.13 Desired and Actual Power Levels vs. Time. 60

4.14 Power Percent Error vs. Time. 61

4.15 Control Rod Adjustments vs. Time for an 80% Run (1.92 MW). . . 62

4.16 Desired Power Trajectory for a Sine Oscillation About the 50% Power Level. 64

4.17 Desired and Actual Reactivity Levels vs. Time for Sine Run. 65

4.18 Reactivity Error vs. Time. 66

4.19 Desired and Actual Power Levels vs. Time. 67

4.20 Power Percent Error vs. Time. 68

4.21 Control Rod Adjustments vs. Time for the Sine Wave Run. 69

A.1 Trapezoidal Integration. 76

A.2 Parabolic Integration. 77

List of Tables

2.1 ACRR Values for λ_i's, β_i's, and l 15

3.1 Average Temperature Values at Selected Steady-State Power Levels 27

Chapter 1

Introduction

1.1 Background

The Annular Core Research Reactor (ACRR), shown in Figure 1.1, is a pool-type research reactor located at Sandia National Laboratories/New Mexico. It is able to operate in a pulse power or steady-state power mode. The majority of the experiments performed within the ACRR are done with the reactor in the pulse power mode, but there are instances when the reactor is required to operate in the steady-state power mode [1].

The current/previous steady-state controller, "Auto-Mode", performs so poorly that the majority of the time the reactor is operating in the steady-state configuration, the power level must be controlled manually by the operator. To this end, due to the infrequency of use, relative ease of manual operation in steady-state mode, and lack of time and resources, the need for a more improved "Auto-Mode" controller was not very high.

The new and improved "Auto-Mode" controller should be able to automatically

regulate the ACRR power level to various steady-state power levels within the operation power range of 1 kW to 2.4 MW. In addition, the controller should be able to respond to various ramp rates while still remaining within the physical limits of

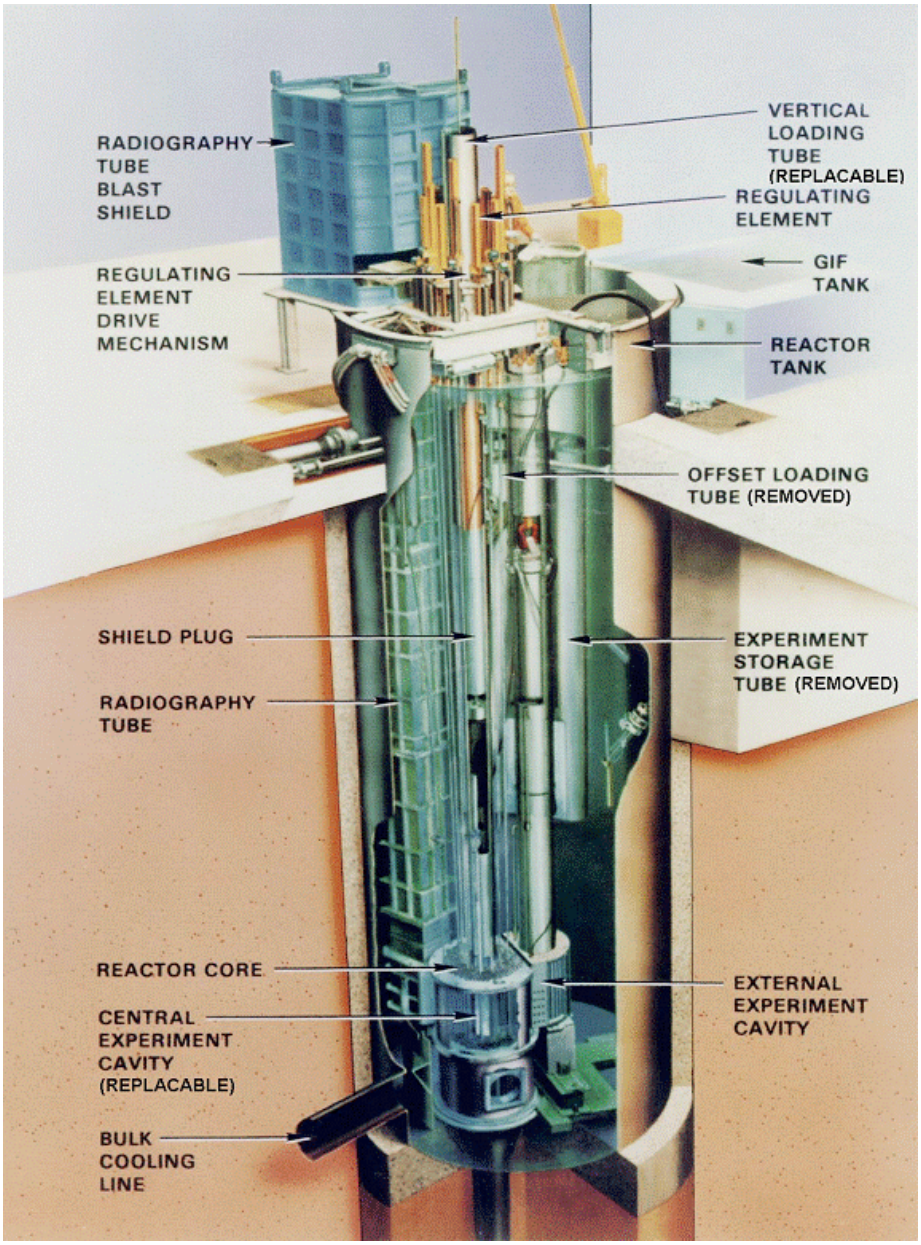

Figure 1.1: Annular Core Research Reactor (ACRR).

the system. It should also have the capability to compensate for power fluctuations within the ACRR due to the negative impact of temperature.

1.2 Overview of ACRR

The ACRR is a water-moderated low-power research reactor using an enriched driver fuel of cylindrical uranium dioxide-beryllium oxide ($UO_2 - BeO$). The ACRR contains a large dry central irradiation cavity at the center of the core that extends to the pool surface. Along with the central dry irradiation cavity the facility has additional experimental cavities, experiment setup areas, and storage locations [1].

The primary mission of the ACRR is to provide a means to subject various components or systems to pulse and steady-state neutron irradiation environments. The types of components and systems irradiated within the ACRR include:

- electronic circuit boards
- passive neutron and/or gamma dosimetry devices
- active neutron and/or gamma dosimetry devices
- arming, fusing and firing systems and their components
- explosive components
- radioactive materials
- experiment holding/positioning fixtures
- neutron spectrum modifying fixtures

The ACRR was designed to produce a high yield of high-energy neutrons in the central irradiation dry cavity and other experimental facilities over a very short-time

pulse. The dry central cavity is centrally located within the core to allow for the irradiation of such components [1].

1.2.1 Detailed Description of ACRR

The ACRR open-pool-type system utilizes light water as the moderator and coolant, with cylindrical fuel elements arranged in a triangular-pitch grid structure about the centrally located circular irradiation space. The fuel elements are cooled by natural convection and the pool water is cooled using a heat rejection system. The fuel elements consist of cylindrical uranium dioxide-beryllium oxide ($UO_2 - BeO$) fuel clad in stainless steel. The $UO_2 - BeO$ fuel, shown in Figure 1.2, is in the form of annular slotted disks stacked within the fluted, niobium fuel cups. The niobium fuel cups are then stacked within the fuel element stainless steel cladding. A BeO plug is located at each end of the fuel region and the overall length of the fuel element is 73 cm (28.7 in) [1].

In Figure 1.2, we also see the most prominent feature of the ACRR; the 233 mm (9.19 in) inside diameter central irradiation space, located at the center of the core. The annular core is formed by individual single fuel elements, having their longitudinal axes oriented vertically, in a hexagonal grid, around the central cavity. The core nominally consists of 200-250 fuel elements, with six fuel-followed control rods and two fuel-followed safety rods. The outer row of fuel elements are surrounded by nickel reflector elements. The high-heat-capacity $UO_2 - BeO$ fuel elements, take up approximately six rings of the hexagonal grid. Each of the fuel elements are individually positioned within the core by a pair of holes, one hole in an upper grid plate and the other hole in a lower grid plate. The core grid plates contains over 300 locations for fuel elements; however, only a maximum of 251 fuel elements are available for use within the ACRR [1].

When the ACRR is in the pulse-mode configuration, it uses 236 fuel elements and the three transient poison elements that are withdrawn pneumatically or by the use of high-speed stepper motors. In the steady-state mode, the six control rods are driven by six individual stepper motors. Each of the control rods may be driven or positioned independently, but they are typically driven together as bank controlling

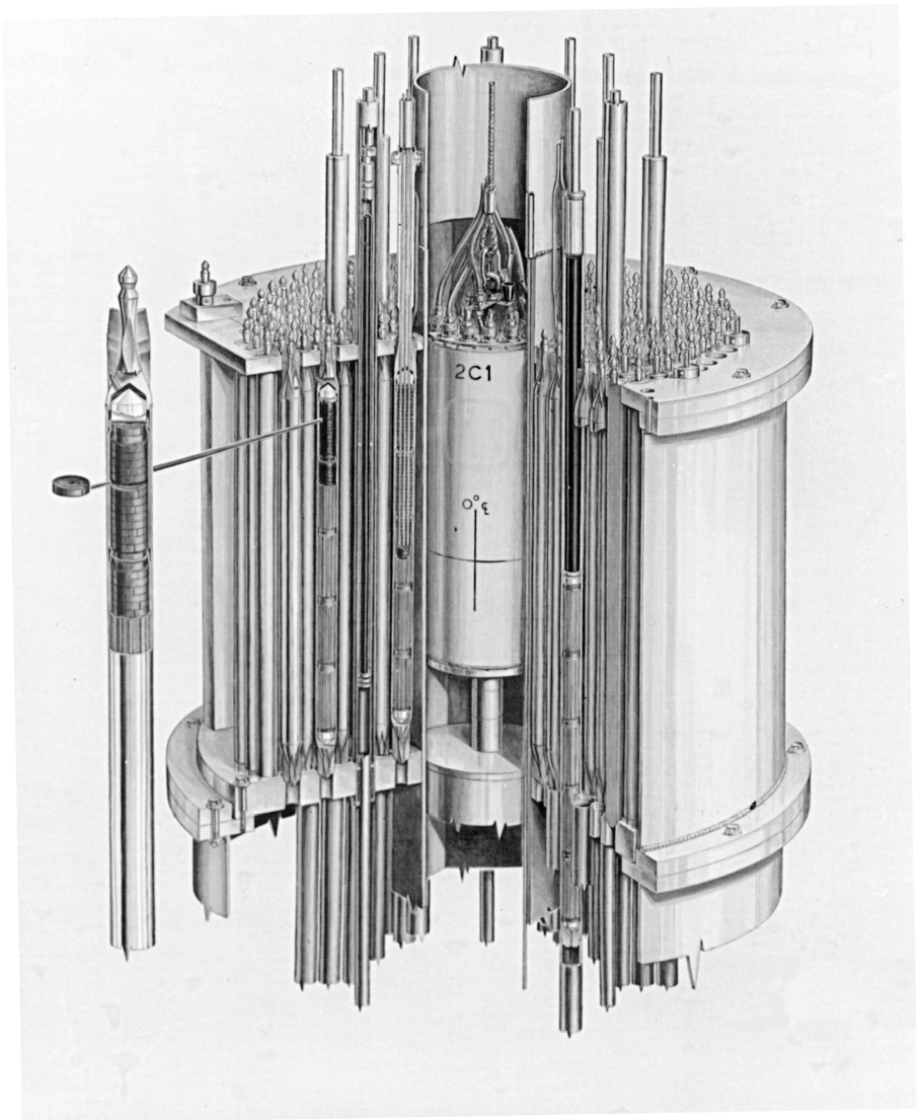

Figure 1.2: ACRR Core and Central Irradiation Space.

the ACRR reactivity levels [1].

The ACRR core is located near the bottom of a stainless-steel cylindrical vessel known as the ACRR tank. The upper portion of the tank and reactor bridge is shown in Figure 1.3. The ACRR tank is 3.05 m (10 ft) in diameter and 8.54 m (28 ft 0.25 in) in depth. The tank extends 0.91m (3 ft) above the ACRR High Bay floor. The top of the core upper grid plate is 6.89 m (22 ft 7.25 in) below the tank lip. The depth of the water between the top of the core upper grid plate and the surface of the pool is approximately 6.1 m (20 ft). At this depth, the saturated water temperature at the core exit is approximately 111^oC. The water also provides radiation shielding [1].

Figure 1.3: Top of the ACRR Tank.

1.3 Thesis Contribution

This thesis presents a theoretical model of the ACRR. This model of the ACRR is capable of recreating real-world power runs when subjected to similar reactivity inputs. The main assumption for this model is that the position of the control rods is the only controllable input.

The inverse kinetics method is applied to the power histories of the ACRR model. By comparing the reactivity levels over time, the method is shown to be valid. Then by utilizing the inverse kinetics method to calculate the reactivity levels over time from a desired power trajectory, we are given a desired reactivity trajectory. A PID compensator is then constructed to automatically control the ACRR reactivity levels such that it follows the desired reactivity trajectory.

1.4 Organization of Thesis

The relevant concepts of the theory of nuclear engineering are presented in chapter 2. The point reactor kinetics equations, the inhour equation, and the inverse kinetics method are also introduced in this chapter as well. The development of a reactor model representing the ACRR and the application of the inverse kinetics method are introduced in chapter 3. Chapter 4 covers the development of a state-space representation of the ACRR for the purpose of a more efficient model. Chapter 4 also covers the introduction and management of a pure time delay within the ACRR system that results in the adjustment of control rods every 0.5 seconds. Chapter 4 concludes with the creation and simulation results of a PID compensator to automatically adjust the position of the control rods at every 0.5 second interval. Chapter 5 contains the concluding remarks and future areas of study.

Chapter 2

Theory

2.1 Overview

This chapter gives a brief introduction in basic reactor theory that includes nuclear fission and neutron population. The point reactor kinetics equations and the inhour equation are introduced along with the discussion of the reactivity unit of measure "the *dollar*" and the concept of the stable reactor period. The chapter concludes with the introduction of the inverse method and the importance of this method when applied to the calculation of reactivity.

2.2 Nuclear Reaction Theory

A nuclear reaction is a process in which two nuclear particles collide to produce products different from the initial particles. Nuclear fission is the process of splitting the nucleus of an atom into smaller, lighter nuclei, along with the release of free neutrons, gamma rays, and other subatomic particles. The fission of heavier elements produces

a large amount of energy and more tightly bounded fission fragments (nuclei).

2.2.1 Nuclear Fission

We begin with the model of an atom, introduced by Niels Bohr in 1913. It consists of a heavy central nucleus surrounded by orbital electrons. The nucleus consists of two types of particles: protons and neutrons which are often referred to as nucleons. The proton and electron are of exactly opposite charge and the neutron is electrically neutral. Since protons and neutrons have roughly the same mass, the nucleus itself has a mass that is nearly proportional to the atomic mass number, defined as the total number of protons and neutrons [2].

According to the concept of the *the equivalence of mass and energy*, given in [3], the mass defect of an atom is a measure of the energy that would have to be supplied to break apart the nucleus. Therefore, the energy counterpart of the mass defect is known as the binding energy of the nucleus. In the nuclear physics and related fields, energy is often measured in terms of the electron volt, represented by eV; this is the energy acquired by a unit (electronic) charge which has been accelerated through a potential of 1 volt. Since one electronic charge is equal to 1.60 x 10^{-19} coulomb, it follows that 1 eV is equal to 1.60 x 10^{-19} joule, and 1 million electron volts (or 1 MeV) is equal to 1.60 x 10^{-13} joule [3].

As Duderstadt and Hamilton state in [4], "the binding energy per nucleon in the atomic nuclei reaches a maximum of 8.7 MeV for nuclei with a mass numbers of about 50". Therefore, we are able to produce more tightly bound nuclei and release energy, by causing a heavy nucleus to fission into two nuclei of intermediate mass numbers. For heavy nuclei, there exists a stability against spontaneous fission due to the short-range nuclear forces that are present within the nucleus. These forces act as a potential energy barrier that must be overcome before the nucleus will fission.

The range of this barrier occurs between 6-9 MeV in most heavy nuclei of interest. Therefore, if we wish to cause one of the heavier atomic nuclei to fission, we must add an ample amount of energy, to overcome the fission barrier [4].

This can be done in a number of different ways. One option is to allow the heavy nucleus to capture a neutron. Then the binding energy of the added neutron is enough to exceed the fission barrier and induce fission. Nuclides that can be induced to fission with the capture of a neutron of almost zero kinetic energy, or thermal neutrons which have small kinetic energies when compared to nuclear energies, are called fissile nuclides [4].

Of course with many heavy nuclides, the additional binding energy due to the capture of a neutron is still not enough to cause the heavy nucleus to break through the fission barrier. If this occurs we can add a little extra energy to the neutron by giving it a kinetic energy of one MeV or so, which is enough to push the nucleus through the barrier and cause fission. Nuclides that are fissioned with these "fast" neutrons are called fissionable nuclides. Since heavy nuclei are likely to induce fission by the capture of either low energy neutrons or "fast" neutrons, the probability that such a neutron will induce the heavy nuclei into fissioning (fission cross section) relies heavily on the neutron population within the nuclear reactor [4].

2.2.2 Neutron Population

For a reactor, we can describe the neutron population by the following equation given by Knief in [2] as:

$$Accumulation = production - absorption - leakage. \tag{2.1}$$

This equation dictates that neutrons, just like mass and energy, must be conserved [2].

If the neutron population within a reactor is at a nonzero steady-state level and the fission chain reaction within the reactor is self-sustaining, the reactor is said to be critical. Criticality may occur at any fission rate (or power level) just as long as neutron losses are equal to the neutron production [2].

If the neutron production within a reactor is greater than the neutron losses, the reactor is said to be supercritical and is characterized by increasing power levels. If a reactor is subcritical, the reactor has neutron losses that are greater than production and, thus, a decrease in power level occurs [2].

The multiplication factor, k, describes the tendency of the neutron population to change and is defined as:

$$\mathrm{k} = \frac{\text{neutron production}}{\text{neutron losses}}, \tag{2.2}$$

where the losses result from both absorption and leakage. Thus, the value of k is used to describe the state of the reactor where:

$\mathrm{k} = 1$ the reactor is critical

$\mathrm{k} > 1$ the reactor is supercritical

$\mathrm{k} < 1$ the reactor is subcritical.

The multiplication factor may be regulated by changing the production, absorption, and/or leakage of the neutron population [2].

We may now introduce the concept of reactivity, represented by ρ, defined as:

$$\rho(t) \equiv \frac{\mathrm{k}(t) - 1}{\mathrm{k}(t)}. \tag{2.3}$$

This quantity measures the deviation of the multiplication factor from its critical value $\mathrm{k} = 1$ [4].

2.2.3 Delayed Neutrons

The fact that several neutrons are produced in a fission reaction is just as significant as the energy released in the reaction. These neutrons may be used to create or sustain a fission chain reaction. As Duderstadt and Hamilton state in [4], "the majority of these fission neutrons appear essentially instantaneously (within 10^{-14} sec) of the fission event." The neutrons that appear almost instantaneously are called prompt neutrons. Of course a small number of neutrons, less than 1%, appear with a noticeable time delay from the decay of the radioactive fission products. Even though a small fraction of the fission neutrons are delayed, it is these delayed neutrons that allow us to effectively control the fission chain reaction [4].

The number of neutrons (both prompt and delayed) released in a fission reaction vary from reaction to reaction. Of course, in most nuclear applications we are only concerned with the average number of neutrons released per fission. The average depends on both the nuclear isotope involved and the energy of the neutron causing the fission reaction [4].

By way of example, let us consider a fission product decay scheme leading to the emission of a delayed neutron given first by the beta decay of ^{87}Br to ^{87}Kr*, followed by the subsequent decay of ^{87}Kr* to ^{86}Kr via neutron emission.[1] The time delay of this process is regulated by the half life of the beta-decay, which is around 55 sec. A fission fragment whose beta-decay yields a daughter nucleus which subsequently decays via delayed neutron emission is know as a *delayed neutron precursor*. At least 45 different delayed neutron precursor isotopes may be produced in a fission chain reaction. In reactor analysis the precursors are often grouped into six classes characterized by approximate half-lives of 55, 22, 6, 2, 0.5, and 0.2 sec, respectively,

[1] *beta decay* corresponds to the conversion of a neutron in the nucleus into a proton, generally accompanied by the emission of an electron and a neutrino. An asterisk is used to denote a nucleus in an excited state

where each precursor group contains a number of different isotopes [4].

Due to the fact that the relative isotopic yield per fission varies for different fuel isotopes, the detailed characteristics of the precursor groups are also isotope dependent. Therefore, we define:

- λ_i = Decay constant (beta-decay) of the ith precursor group
- β_i = Fraction of delayed neutrons emitted per fission that appear from the ith precursor group
- $\beta = \sum_i \beta_i$ = Total fraction of fission neutrons which are delayed.

In our case, we will take the number of precursor groups to be six, i.e. $i = 6$. In addition, the energy range of the delayed fission neutrons is significantly lower than prompt fission neutrons and also depends highly on both the delayed neutron group and fissioning isotope [4].

2.2.4 Neutron Lifetime and Mean Neutron Generation Time

Earlier we defined the multiplication factor, k, as:

$$\mathrm{k}(t) = \frac{\text{neutron production}}{\text{neutron losses}} = \frac{\mathrm{P}(t)}{L(t)}, \tag{2.4}$$

where $\mathrm{P}(t)$ represents the amount of neutron production within the reactor at time t and $L(t)$ represents the amount of neutron losses within the reactor at time t. Now, we define the neutron lifetime, l, as:

$$l = \frac{N(t)}{L(t)}, \tag{2.5}$$

where $N(t)$ is the total neutron population within the reactor at time t [4].

The mean neutron generation time, Λ, is defined as:

$$\Lambda \equiv \frac{l}{\mathrm{k}(t)}, \tag{2.6}$$

where l is the mean lifetime of a neutron in the a reactor and k is the multiplication factor. From equation (2.3) we can solve for k:

$$\mathrm{k}(t) = \frac{1}{1-\rho(t)}. \tag{2.7}$$

Using equation (2.7) we can rewrite equation (2.6) as:

$$\Lambda = l(1-\rho(t)). \tag{2.8}$$

As we can see the mean neutron generation time will vary with time as the multiplication factor or the reactivity within the reactor varies as well.

2.3 The Point Reactor Kinetics Equation

The base for a reactor model is a set of ordinary differential equations known as the point reactor kinetics equations described in equations (2.9) and (2.10). These equations are first order and linear, and essentially describe the change in the neutron density within the reactor due to a change in reactivity, $\rho(t)$. The equations are as follows,

$$\frac{dn}{dt} = \left[\frac{\rho(t)-\beta}{\Lambda}\right] n(t) + \sum_{i=1}^{6} \lambda_i C_i(t) \tag{2.9}$$

$$\frac{dC_i}{dt} = \frac{\beta_i}{\Lambda} n(t) - \lambda_i C_i(t), \qquad i = 1, \ldots, 6 \tag{2.10}$$

where:

- $n(t)$ - neutron density (total number of neutrons in the reactor at time t)
- $C_i(t)$ - effective precursor concentration for group i (expected number of precursors of ith kind that always decay by emitting a delayed neutron)
- $\rho(t)$ - reactivity
- Λ - mean neutron generation time
- β - total effective delayed neutron fraction ($\beta = \sum_{i=1}^{6} \beta_i$)
- β_i - effective delayed neutron fraction of group i
- λ_i - effective decay constant of group i

These seven coupled ordinary differential equations express the time-dependence of the neutron population and the decay of the delay neutron precursors within a reactor [4]. The values used for the variables λ_i's, β_i's, and l are specific for each reactor. The values for the ACRR are given in Table 2.1. Additionally the neutron density within a reactor is proportional to neutron flux, fission rate, and reactor power level [2].

Table 2.1: ACRR Values for λ_i's, β_i's, and l

i	β_i	λ_i	l
1	2.66 x 10^{-4}	1.27 x 10^{-2}	
2	1.492 x 10^{-3}	3.17 x 10^{-2}	
3	1.317 x 10^{-3}	0.115	
4	2.851 x 10^{-3}	0.311	24 x 10^{-6}
5	8.97 x 10^{-4}	1.40	
6	1.82 x 10^{-4}	3.87	
total	0.0073	na	

2.3.1 Derivation of the Inhour Equation

To derive the inhour equation, let us consider a simple situation in which we imagine a reactor operating at a initial neutron density level n_o prior to $t = 0$. Then, the reactivity is changed to a nonzero value ρ_o. To make this example a little more straightforward we will not attempt to solve the full set of point reactor kinetics equations, but instead we consider the case where all delayed neutrons are represented by one delayed group, characterized by a total yield fraction

$$\beta = \sum_i \beta_i \tag{2.11}$$

and an averaged decay constant

$$\lambda = \left[\frac{1}{\beta}\sum_i \frac{\beta_i}{\lambda_i}\right]^{-1}. \tag{2.12}$$

For this simplified case the point reactor kinetics equations, (2.9) and (2.10), may be rewritten as:

$$\frac{dn}{dt} = \left[\frac{\rho(t) - \beta}{\Lambda}\right] n(t) + \lambda C(t) \tag{2.13}$$

$$\frac{dC}{dt} = \frac{\beta}{\Lambda} n(t) - \lambda C(t) \qquad t \geq 0. \tag{2.14}$$

Prior to $t = 0$ the reactor is at a steady-state neutron density level n_o. Thus, we find that for $t < 0$ the following must hold:

$$\frac{dn}{dt} = \frac{dC}{dt} = 0 \rightarrow C_o = \frac{\beta}{\lambda\Lambda} n_o. \tag{2.15}$$

The equations in (2.15) result in the initial conditions for equations (2.13) and (2.14):

$$n(0) = n_o, \qquad C(0) = \frac{\beta}{\lambda\Lambda} n_o. \tag{2.16}$$

This initial value problem may be solved by seeking exponential solutions of the form:

$$n(t) = ne^{t\omega}, \qquad C(t) = Ce^{t\omega}, \tag{2.17}$$

where n, C, and ω are to be determined. If we substitute the initial conditions into equations (2.13) and (2.14) we arrive at following equations:

$$\omega n = (\frac{\rho_o - \beta}{\Lambda})n + \lambda C, \qquad \omega C = \frac{\beta}{\Lambda}n - \lambda C. \tag{2.18}$$

These equations have a solution if and only if

$$\Big[\omega - (\frac{\rho_o - \beta}{\Lambda})\Big](\omega + \lambda) = 0,$$

or

$$\Lambda\omega^2 + (\lambda\Lambda + \beta - \rho_o)\omega - \rho_o\lambda = 0 \text{ [4]}. \tag{2.19}$$

Equation (2.19) is the characteristic equation for the parameter ω. Using our definition of Λ in equation (2.8), we are able to write equation (2.19) as

$$\rho_o = \frac{\omega l}{\omega l + 1} + \frac{1}{\omega l + 1}(\frac{\omega\beta}{\omega + \lambda}). \tag{2.20}$$

Using equation (2.20) we can determine the decay constants ω for any constant reactivity ρ_o. Now let's generalize equation (2.20), for six groups delayed of neutrons:

$$\rho_o = \frac{\omega l}{\omega l + 1} + \frac{1}{\omega l + 1}\sum_{i=1}^{6}\frac{\omega\beta_i}{\omega + \lambda_i} \equiv \rho(\omega). \tag{2.21}$$

In reactor theory, equation (2.21) is known as the inhour equation. The roots of this equation, yield the seven decay constants ω_j that describe the time-behavior of the neutron density as,

$$n(t) = A_0e^{t\omega_0} + A_1e^{t\omega_1} + \ldots + A_6e^{t\omega_6} \text{ [4]}. \tag{2.22}$$

2.3.2 Stable Reactor Period

If reactivity ρ is positive, then k > 1, and six of the time constants from equation (2.22) are negative and one is positive. Therefore, in equation (2.22), we take ω_0 to be the one positive time constant, and ω_1, $\omega_2, \ldots, \omega_6$ are the negative time constants. The value of the six negative time constants are of the same order as one of the six decay constants, λ_i, of the delayed-neutron precursors. As t increases, the contributions of all negative time constants in equation (2.22) decrease rapidly to zero. Therefore, these negative time constants make an initial contribution to the neutron density, but soon become negligible compared to the first term, which increases due to the positive time value of ω_0. Therefore, after a short time, equation (2.22) reduces to,

$$n(t) = A_0 e^{t\omega_0} \text{ [3]}. \tag{2.23}$$

The reactor period, T, or e-folding time, is the the amount of time required for the neutron density to change by a factor e, so that

$$n = n_0 e^{t/T}. \tag{2.24}$$

By setting A_0 in equation (2.23) to be equal to n_0, we can set equations (2.23) and (2.24) equal to each other and solve for T. Therefore, we find that T is as follows,

$$T = \frac{1}{\omega_0}, \tag{2.25}$$

and is the stable reactor period. The $1/\omega_1$, $1/\omega_2, \ldots, 1/\omega_6$, values are sometimes referred to as transient periods, but since they are negative, they have no physical meaning as reactor periods in the same sense that $1/\omega_0$ does. Therefore, ω_1, $\omega_2, \ldots, \omega_6$ are merely the values that satisfy equation (2.19). They each affect the

rate of change in the neutron density, but only for a short amount time after the reactivity level has been increased [3].

If ρ is negative, then all seven values of ω are negative. Therefore, equation (2.22) consists of seven time constants, all with negative exponents and the magnitude of each term decreases over time. The time constant with the smallest numerical value, let's call it ω_0, decreases at a slower rate than the other six and yields a stable negative period equal to $1/\omega_0$ [3].

2.3.3 The Dollar

In nuclear engineering a unit commonly used to measure reactivity, is the *dollar*. The *dollar*, is measured in units of the total delayed neutron fraction β, and is defined by

$$\text{Reactivity in dollars} \equiv \frac{\rho}{\beta}. \tag{2.26}$$

The reason for this unit of measure is due to the prompt critical condition that occurs within the reactor when it is critical on prompt neutrons alone. The required condition for prompt criticality can be obtained from equation (2.9). If the delayed-neutron source term is ignored, so that only prompt fission neutrons are considered, equation (2.9) becomes,

$$\frac{dn}{dt} = \frac{\rho - \beta}{\Lambda} n. \tag{2.27}$$

If a reactor is critical, then dn/dt is equal to zero; since n and Λ can not equal zero, we see that the $\rho = \beta$ condition results in a reactor being prompt critical [3]. If $\rho = \beta$, then the amount of reactivity within the system measured in *dollars*, is \$1.00, one dollar.

If we pass the prompt-critical value of a reactor, the neutron density rapidly increases and results in a reactor that is difficult to control. Therefore, special

safety measures are taken to ensure that this condition does not occur during reactor operation [3].

2.4 The Inverse Kinetics Method

If we are given a certain $\rho(t)$ value, there are very few cases where it is possible to calculate an exact solution for $P(t)$. Actually it is more useful to invert the problem and determine the $\rho(t)$ that produces a desired $P(t)$ behavior, since this is more in line with the philosophy of reactor control [4].

In order to solve for $\rho(t)$ in terms of $P(t)$, let us derive yet another form of the kinetics equations. Let us begin by solving equation (2.10) in terms of $P(t)$:

$$C_i(t) = \int_{-\infty}^{t} \frac{\beta_i}{\Lambda} P(t') e^{-\lambda_i (t-t')} dt' = \int_0^{\infty} \frac{\beta_i}{\Lambda} e^{-\lambda_i \tau} P(t-\tau) d\tau, \tag{2.28}$$

where we assume that $C_i(t)e^{\lambda_i t} \to 0$ as $t \to -\infty$ and then let $\tau = t - t'$ to obtain the second integral of equation (2.28). We now place equation (2.28) into equation (2.9) and write

$$\frac{dP}{dt} = \left[\frac{\rho(t) - \beta}{\Lambda}\right] P(t) + \int_0^{\infty} \left[\sum_{i=1}^{6} \frac{\lambda_i \beta_i}{\Lambda} e^{-\lambda_i \tau}\right] P(t-\tau) d\tau. \tag{2.29}$$

By defining the delayed neutron kernel $D(\tau)$ as:

$$D(\tau) \equiv \sum_{i=1}^{6} \frac{\lambda_i \beta_i}{\beta} e^{-\lambda_i \tau}, \tag{2.30}$$

and substituting into equation (2.29), the result is an integro-differential form of the kinetics equations,

$$\frac{dP}{dt} = \left[\frac{\rho(t) - \beta}{\Lambda}\right] P(t) + \frac{\beta}{\Lambda} \int_0^{\infty} D(\tau) P(t-\tau) d\tau, \tag{2.31}$$

which describes the time-dependence of the power level of a reactor. If we rearrange this equation such that $\rho(t)$ is given in terms of $P(t)$ we arrive at the inverse equation:

$$\rho(t) = \beta + \Lambda \frac{d}{dt}[\ln P(t)] - \beta \int_0^\infty D(\tau) \frac{P(t-\tau)}{P(t)} d\tau. \tag{2.32}$$

This relationship is important since it can be used to calculate a specific reactivity input such that it yields a desired power response and to provide information about the feedback mechanism within the reactor [4].

2.5 Concluding Remarks

With the introduction of the point reactor kinetics equations and the inverse kinetics method behind us we now move on to applying them to the development of an ACRR model. In the next chapter, the kinetics equations is the cornerstone in creating such a model. The measurement of the reactor model's stable reactor period is used as a measure for comparing its operation with that of the actual ACRR. Last but not least, the inverse method is applied to calculating the reactivity levels from a power run of the ACRR model.

Chapter 3

Reactor Model

3.1 Overview

This chapter reviews the steps taken to develop a model describing the neutron population time-dependence of the ACRR by solving the point kinetics equations. Second we review the temperature vs. power equation construction that is used to determine the temperature feedback relationship. We present a discussion on the management of the ACRR data and the data manipulation that is done to create useful control rod inputs. Then by combining the model describing the neutron population and the temperature feedback relationship a model describing the ACRR is generated. Simulations recreating real world power runs are prepared by subjecting the ACRR model to similar control rod inputs. Then the simulation power output is compared to the real world power output to verify that the ACRR model does in-fact perform similarly to the actual ACRR in response to similar inputs.

The chapter concludes with the development of the inverse kinetics method which is used to determine the effective reactivity levels within the reactor. The validity of the inverse kinetics method is determined by comparing the actual simulated

reactivity levels with those found by the inverse kinetics method.

3.2 Solving the Kinetics Equation

The point kinetics equation, as we know, describes both the time-dependence of the neutron population within a reactor and the decay of the delayed neutron precursors. Since the neutron population of a nuclear reactor is proportional to its power level, we can rewrite equations (2.9) and (2.10) in terms of reactor power ($P(t)$) as,

$$\frac{dP}{dt} = \Big[\frac{\rho(t) - \beta}{\Lambda}\Big]P(t) + \sum_{i=1}^{6} \lambda_i C_i(t) \tag{3.1}$$

$$\frac{dC_i}{dt} = \frac{\beta_i}{\Lambda}P(t) - \lambda_i C_i(t), \qquad i = 1, \ldots, 6. \tag{3.2}$$

To solve the kinetics equations, we use Euler's method. To do this, we first have to determine the initial conditions of the effective precursor concentration for each group (the C_i's) due to an initial steady-state power level. By assuming the reactor is initially at a steady-state, i.e. $dC_i/dt = 0$, we can easily solve for each of the effective precursors at an initial steady-state power level by solving the following equations:

$$C_i(0) = \frac{\beta_i}{\lambda_i \Lambda}P(0), \qquad i = 1, \ldots, 6.$$

By using a relatively small time step (dt) of 0.001 seconds, in the script file *reactor1a.m*, we are able to solve the kinetics equations and determine the reactor power level at each time step due to a given input.[1] Figure 3.1 is an example of the reactor power level due to a positive \$0.25 reactivity insertion. We also simulated positive reactivity insertions of \$0.05 and \$0.50. In all cases the initial power level was assumed to be 3 kW.

[1]The script file *reactor1a.m* is located in Appendix B

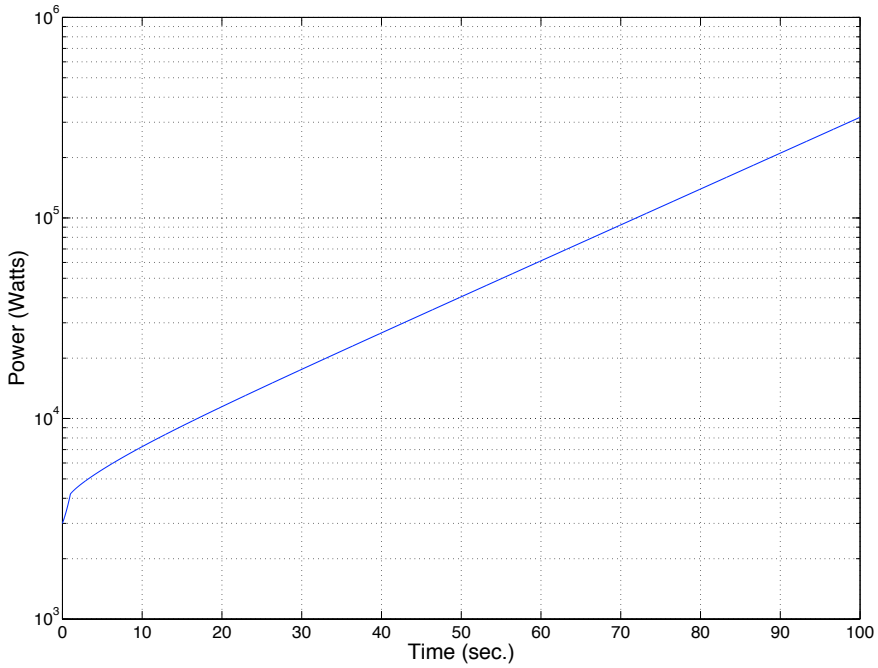

Figure 3.1: Power vs. Time for a \$0.25 Reactivity Insertion.

To clarify, the ACRR model at this point is at its simplest. The kinetics equations are only describing the time-dependence of the power level for the ACRR model. So far, there is no regulation of the power level. If a positive amount of reactivity, no mater how small, is added to the ACRR model, the power level will grow without bound. The power level, at first, will grow exponentially and then after some time (once the transients die out), will continue to grow asymprotically as it reaches its stable period.

3.2.1 Validation

From the calculated reactor power level vs. time we are be able to determine the stable reactor period for a given reactivity insertion. By comparing the stable reactor

period to known stable period values for the ACRR, we obtain a measure on how accurate a model of the time-dependence of the power level for the ACRR is.

In a reactor core, the growth or decay of the neutron population obeys an exponential law. Since the neutron population of a nuclear reactor is essentially proportional to its power level, we may consider the time behavior of the reactor power level as being exponential with a time constant (period) as well. Therefore, the power within the reactor can be described by equation (3.3):

$$P(t) = P_o e^{t/T}, \tag{3.3}$$

where P_o is the original power, t is time in seconds, and T is the instantaneous reactor period. Since we want to determine the instantaneous reactor period all we need to do is solve for T in equation (3.3). By doing so we end up with the expression,

$$T = \frac{t}{\ln(P(t)) - \ln(P_o)}. \tag{3.4}$$

In practice, measuring the stable reactor period is one of the simplest types of kinetic measurements used to determine the reactivity "worth". It is done by making a small positive perturbation in the core of a critical reactor. Here we are doing the opposite by adding a small amount of reactivity and then measuring the resulting reactor period.

In Figure 3.1 we can see that the power level after a positive \$0.25 reactivity insertion is growing exponentially, or linearly when plotted on a semi-log scale. After running the simulation for a significant amount of time, to insure the reactor model has reached its stable period, we calculate the period by the use of equation (3.4) and find it to be 24.30 seconds.

The reactor period was calculated again 2 more times for the positive reactivity insertions of \$0.05 and \$0.50. The stable periods were 217.58 seconds and 5.45

seconds respectively. These 3 stable period values were all very close to known stable period values for the ACRR for the given positive reactivity insertions. With these results sufficiently close to the known stable period values of the ACRR, we are confident that the simulation is correctly modeling the time-dependence of the power level for the ACRR.

3.3 Temperature Feedback

The temperature feedback in the ACRR is what makes it inherently stable. In order to increase the power in the ACRR one must add positive reactivity. The increases in power, in turn, increases the temperature in the ACRR. The increase in temperature then causes the effective amount of reactivity to decrease and therefore the change in power from one time step to the next decreases until a steady-state power level is reached in which the effective amount of reactivity acting on the system is equal to zero.

When combined with the kinetics equations the temperature feedback describes how the reactor reacts dynamically. Once a temperature feedback relationship is created we are able to take existing data from previous steady-state power runs of the ACRR and attempt to recreate the power histories using our model.

There are two routes in developing a description of the temperature feedback for the ACRR. The first is to develop a heat transfer model. Unfortunately this route proved to be much to difficult and time consuming. The second is to take the data collected from a safety analysis simulation and develop a temperature vs. power equation. The average temperature results from the simulation at selected power levels are shown in Table 3.1. The values given in Table 3.1 are then plotted against each other and by the use of a curve fitting tool, we develop a temperature vs. power equation by adding a trend line to the plot as shown in Figure 3.2. The resulting

Table 3.1: Average Temperature Values at Selected Steady-State Power Levels

Power	T(Ave) °C
1 W	20.1
1 kW	21.2
10 kW	26.7
100 kW	78.9
500 kW	244.4
1 MW	392.8
2 MW	607.8
3 MW	789.9
4 MW	955.0

temperature vs. power equation is given in equation (3.5) as,

$$T = 2.64P(t)^5 - 30.7P(t)^4 + 138P(t)^3 - 317P(t)^2 + 577P(t) + 21.2. \tag{3.5}$$

The temperature vs. power equation, (3.5), allows us to calculate the average reactor temperature at a given power level. By then using a ratio of $-\$0.006/°\text{C}$, we calculate the reactivity feedback due to the average temperature within the reactor.[2] The ratio of reactivity per degree Celsius was provided by the group that oversees the operation of the ACRR. It is a good estimate that is used to calculate the amount of negative reactivity present within the reactor during steady-state runs.

[2]The $-\$0.006/°\text{C}$ ratio is based on the U^{238}/U^{235} ratio and is an inherent property for the ACRR $UO_2 - BeO$ fuel

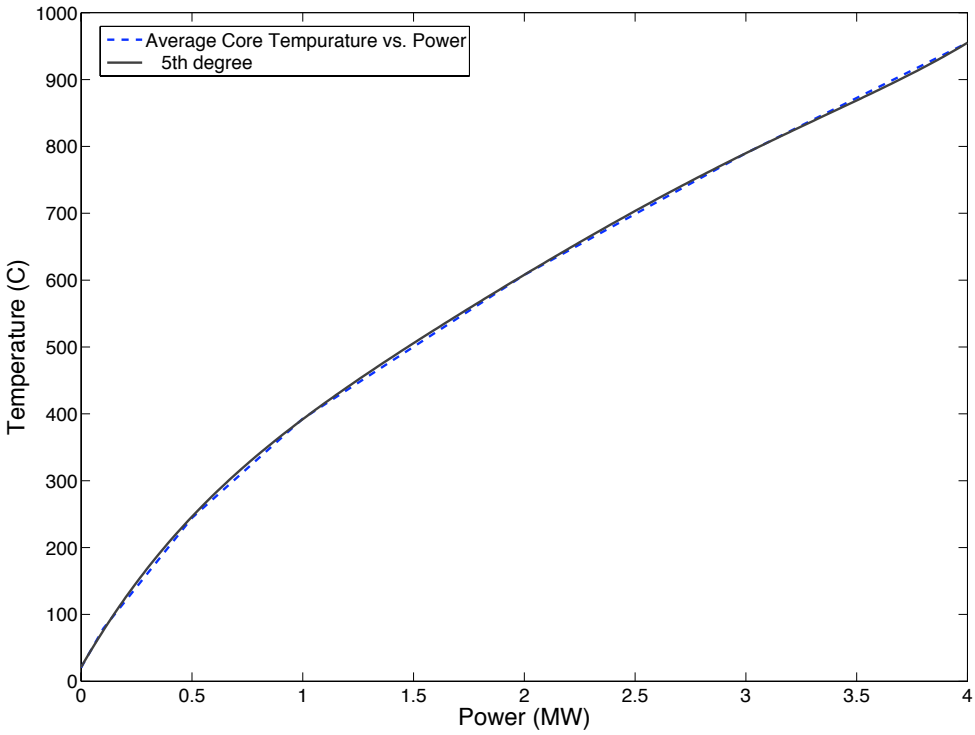

Figure 3.2: Average ACRR Core Temperature vs. Steady-State Power Levels and Resulting Curve Fit.

3.4 Sorting Data

First, we were given numerous EXCEL files containing the ACRR power level, control rod position, core temperature levels, etc. vs. time for steady-state runs that occurred from 2002 to 2006. The files were thoroughly sorted and only the steady-state runs that include a large power increase were used. Large power increases describe a steady-state run that included an increase in power above 20% of the reactor peak power. The ACRR peak power or 100% power is 2.4 MW. So basically any steady-state run that had the ACRR steady-state power at 20% or higher is used.

In order to save memory space, the data logs received from the steady state runs had the data logged every 6 seconds. This threw a minor "kink" in validating the model by re-creating the ACRR steady state runs, since the current model describing the time-dependence of the ACRR power level needs a reactivity value every 0.001 seconds. Luckily, by using the MATLAB function *spline* we are able to essentially fill in the gaps and have a reactivity value every 0.001 seconds.

The ACRR, when in steady-state power opertion, has 6 control rods to regulate the amount of reactivity in the system. In the data logs the rod position vs. time for each control rod is recorded in rod units. One rod unit is equal to 0.1 mm. The position of all the control rods is averaged, since they often move together but every once in a while a few might be off by a couple of rod units. The position of the control rods is estimated in-between the sample points by using the MATLAB function *spline* to include the rod position during the steady-state run at every 0.001 seconds. In Figure 3.3, we can see the average control rod position history plotted before and after the use of *spline*. As we can see the *spline* function does a good job filling in the gaps and creates a usable control rod input by calculating the average position of the control rods at every 0.001 second interval throughout the entire run.

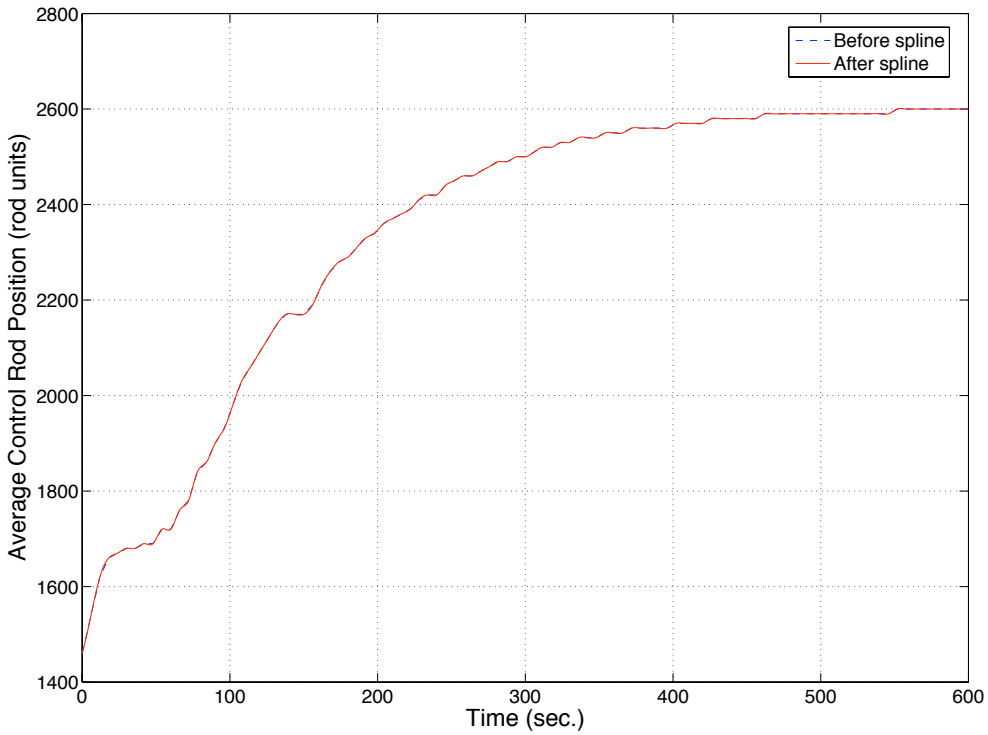

Figure 3.3: Average Control Rod Position vs. Time for Steady-State Run 8512 (80%).

3.5 Recreation

When the ACRR is used for a steady-state power run, the reactor operators almost always bring the reactor up from zero-power to a relatively low steady-state power level usually between 5% and 15% of the ACRR full power level. Then, some time after steady-state is reached at the lower power level, the reactor is taken up to its desired steady-state power level. Here in the recreation of these steady-state power runs we always begin the simulation just as the operator is beginning to take the reactor up to the desired steady-state power level from the lower steady-state power level. Therefore, the initial average control rod position and power level is never

equal to zero, but the initial amount of effective reactivity is always equal to zero.

In order to calculate the amount of reactivity added to the reactor by the control rods, we always assume that the reactor is critical, i.e. in steady-state, before the run begins. Therefore, the effective reactivity in the system is equal to zero and the initial average position of the control rods is used to calculate the amount of reactivity added to the reactor while the operator is taking the ACRR to its desired steady-state power level from its initial low level steady-state power level. The reactivity from the control rods is calculated using equation (3.6).

$$\rho_{add}(t) = (y(t) - y_o)(0.003), \tag{3.6}$$

where y_o is the average initial position of the control rods in rod units, $y(t)$ is the average position of the control rods in rod units over time, and $\rho_{add}(t)$ is the reactivity added by the control rods in dollars over time. The 0.003 multiplier is due to the relationship of \$0.003/rod unit for the collective control rod bank.

Now, to calculate the effective reactivity (ρ_{eff}) within the reactor, we simply add the reactivity from the control rods to the feedback reactivity due to the temperature within the reactor,

$$\rho_{eff} = \rho_{add} + \rho_f, \tag{3.7}$$

where we use the temperature vs. power equation to find the current temperature due to the current power level and then the $-\$0.006/^{\circ}\text{C}$ ratio to calculate ρ_f from the current temperature. The effective reactivity, ρ_{eff}, and the current power level are then used to find the resulting power by solving the kinetics equations.

Figure 3.4 shows a comparison of the actual power level over time (from the ACRR data) and the power level over time calculated by the ACRR model, given

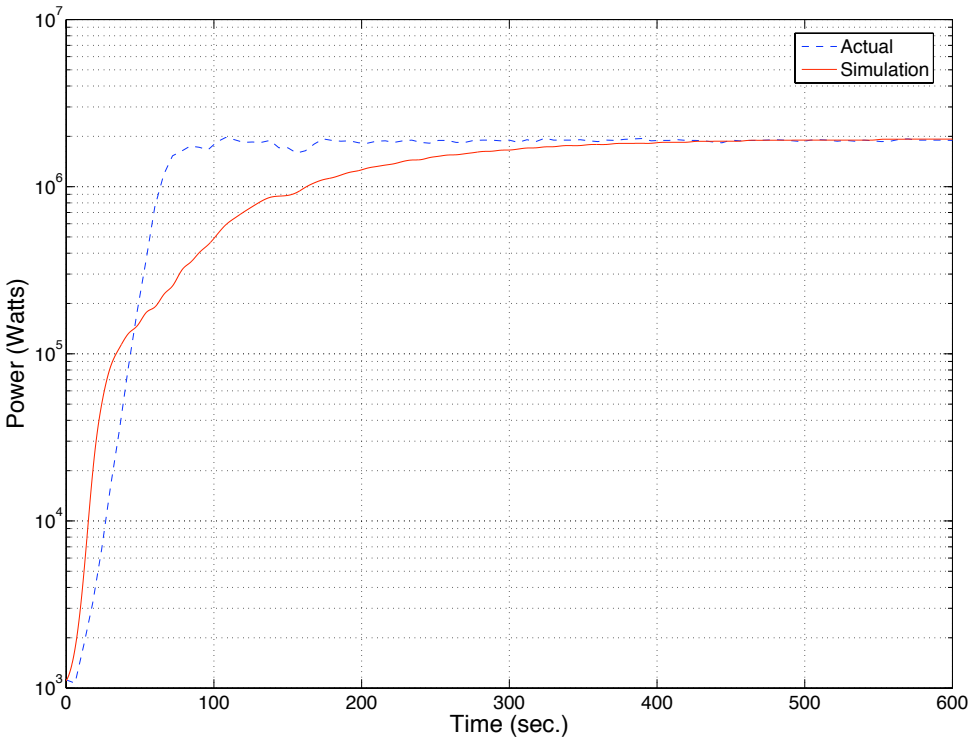

Figure 3.4: Power vs. Time for Steady-State Run 8512 (80%).

by the script file *reactor1b.m*, for an 80% power run.[3] We can see that the eventual steady-state power level is reached, but it does take longer for the simulated power level to reach the desired steady state level. This is due to the temperature vs. power equation, since it is calculated by fitting a polynomial equation to steady-state power and temperature values the temperature feedback is more aggressive than in the real world.

In the real world, the temperature feedback often takes more time to build up depending on the rate at which the reactivity is being inserted. This often allows the reactor operators to quickly get the ACRR up to the desired power level, and

[3]The script file *reactor1b.m* is located in Appendix B

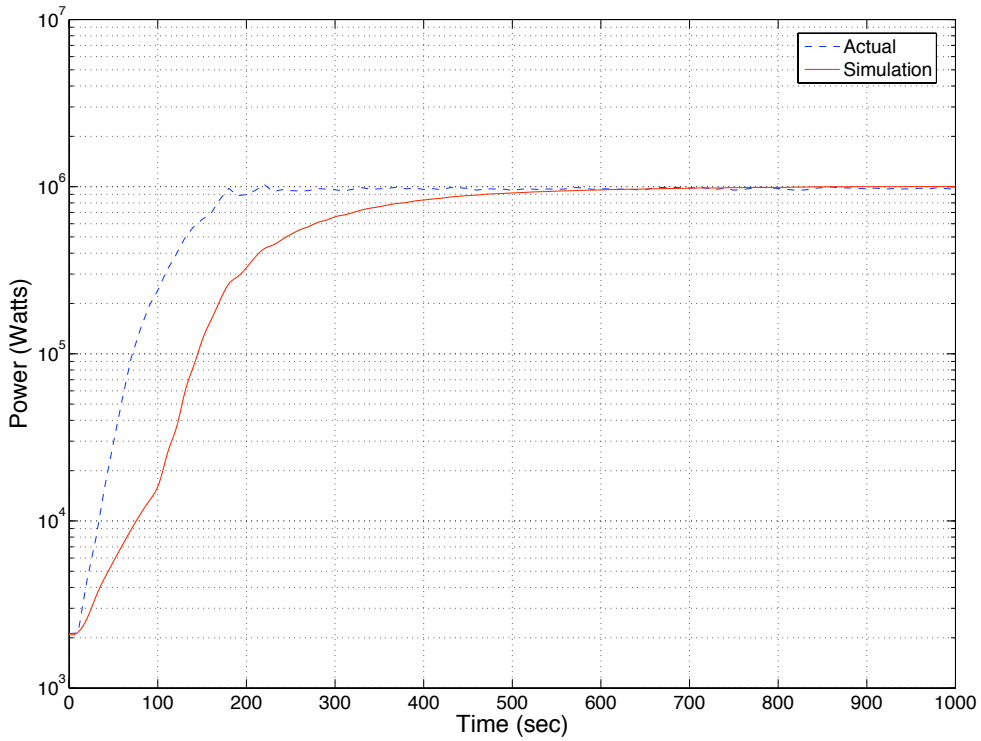

Figure 3.5: Power vs. Time for Steady-State Run 8435 (40%).

then adjust the reactivity levels to keep the ACRR at the desired power level as the temperature feedback becomes more pronounced.

After verifying that the model also adequately simulated 2 additional 80% power runs, we decide to try a 40% power run. Due to the inconsistency in the data given, we only had data that was useable from a few 80% and 40% power runs. After using the MATLAB function *spline* to calculate the position of the control rods for every 0.001 seconds for the 40% power run, we enter it into the ACRR model and the resulting power plot is shown in Figure 3.5.

As we can see from Figure 3.5 the steady-state power level from the model also reaches the desired steady-state power level. The power level for the 40% run does

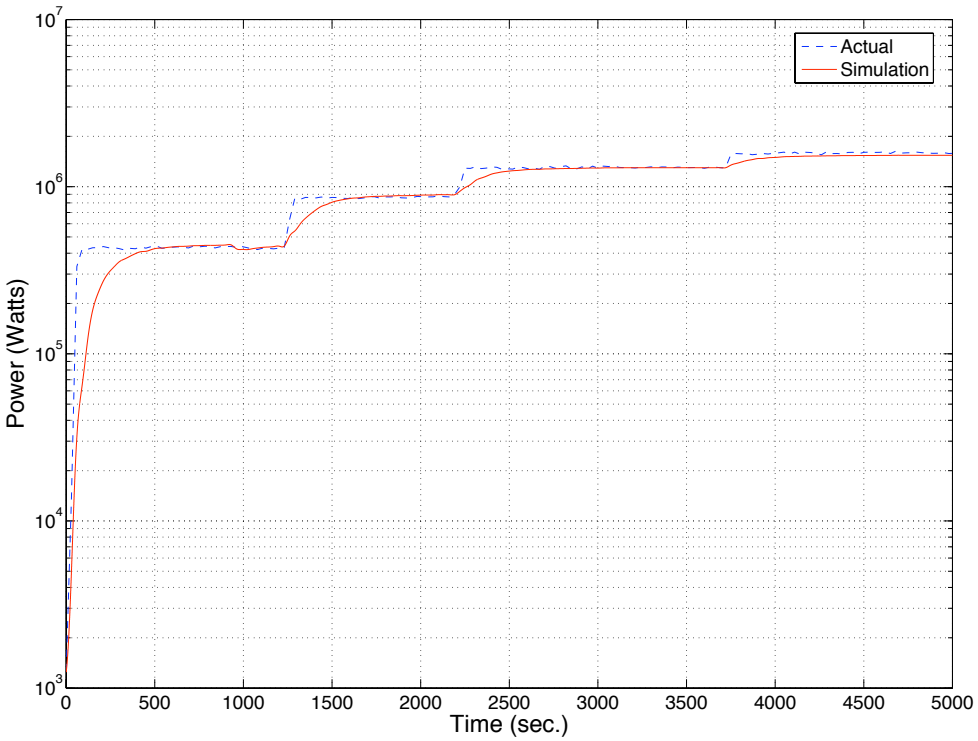

Figure 3.6: Power vs. Time for Steady-State Run 8031.

take a little longer to reach steady-state than the 80% run, but this is most likely due to the control rods being inserted at a slower rate. Another 40% run was also simulated and the ACRR model adequately recreated it as well.

With the ACRR model able to adequately recreate the 80% and 40% power runs using the same reactivity inputs that were given to the ACRR, it is safe to say that a reasonable model of the ACRR has been created, but just to make sure let us test the ACRR model further and see if it can reproduce a run that has numerous steady state power levels. The run starts from approximately the 0% power level and goes up to the 20% power level and holds for a while, then up to 35% and holds, then up to 55% and holds, and finally up to 65% and holds.

As we can see in Figure 3.6, the model reaches all the desired steady-state power levels. This is very good news. Not only are we able simulate reaching single steady-state power levels but, we are also able to reach multiple steady-state power levels within a single run.

3.6 Solving the Inverse Equation

Using the Inverse Equation after each run, we are able to determine the amount of effective reactivity within the reactor at each time step. By comparing the effective reactivity input from the simulation to the calculated effective reactivity, we can check the accuracy of the code.

The inverse equation given in (2.32) is repeated here for the sake of convenience

$$\rho(t) = \beta + \Lambda \frac{d}{dt}[\ln P(t)] - \beta \int_0^\infty D(\tau) \frac{P(t-\tau)}{P(t)} d\tau, \tag{3.8}$$

where the delayed neutron kernel, $D(\tau)$, is as follows,

$$D(\tau) \equiv \sum_{i=1}^{6} \frac{\lambda_i \beta_i}{\beta} e^{-\lambda_i \tau}. \tag{3.9}$$

Let us first apply the inverse kinetics method to solving the amount of effective reactivity within a reactor at a certain instant in time. Here we assume that we have just run a simulation that calculated the output power level of the ACRR from time $t = 0$ to time $t = 500$ sec, and that we have the access to the power level of the reactor at each time step throughout the simulation. Now, let's choose to calculate the amount of effective reactivity at a time $t = 340$ sec.

In equation (3.8), the delayed neutron kernel $D(\tau) \to 0$ as $\tau \to \infty$. Thus, the first step in solving the inverse equation is to determine how large τ needs to be

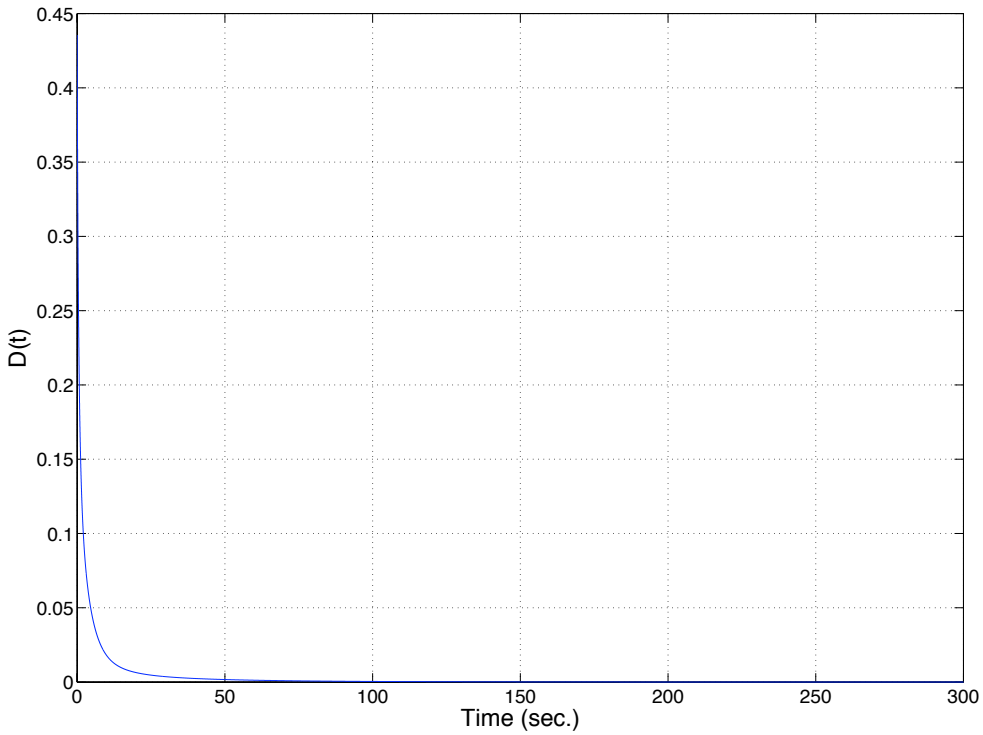

Figure 3.7: Delay Kernel Value vs. Time.

before $D(\tau) \approx 0$. In Figure 3.7, $D(\tau)$ is plotted from 0 to 300 seconds. The cut off point for $D(\tau)$ is chosen to be at 240 seconds, since the value of $D(\tau)$ drops below 3×10^{-5} sec^{-1}. Therefore, we assume the integral portion of equation (3.8) will only be relevant from $\tau = 0$ seconds to $\tau = 240$ seconds.

Now lets breakdown equation (3.8); first let's look at the integral portion of equation (3.8) when attempting to solve for $\rho(340)$:

$$\beta \int_0^{240} D(\tau) \frac{P(340-\tau)}{P(340)} d\tau.$$

For the $\frac{P(340-\tau)}{P(340)}$ portion within the integral, let's take $P(340)$ outside the integral, and treat it as a constant, rather than have it divide $P(340-\tau)$ at every integration

step. Then, we multiply the power history from $P(340)$ to $P(100)$ at each time step by the value of the delayed neutron kernel at each respective integration step and store the values in a column vector as shown below for $\tau = 0 : d\tau : 240$,

$$D(\tau)P(340-\tau) = \begin{bmatrix} D(0)P(340) \\ D(0.001)P(339.999) \\ D(0.002)P(339.998) \\ \vdots \\ D(240)P(100) \end{bmatrix}.$$

Therefore, as τ increases by an integration step, we take the power of the reactor at a later point in time and multiply it by the value of the delayed neutron kernel at τ. Here the value of the integration step was taken to be the same as the time step used for solving the kinetics equations, 0.001 seconds. To integrate, we take the column vector, shown above, that holds the values of $D(\tau)P(340-\tau)$ at each integration step, and use a backward integration method, given in [5], to find the value of the integral at time $t = 340$.

Now for the derivative portion of equation (3.8). Rather than numerically calculating the derivative of $\ln(P(340))$, we instead use $\frac{d}{dt}(\ln P(340)) = \frac{1}{P(340)}$, since $\frac{d}{dt}(\ln x) = \frac{1}{x}$.

To first test the algorithm to solve the inverse equation let us find the reactivity at a single point from an 80% run. This was done at different instances throughout the run and all were compared to the actual input for the simulation. In all cases the reactivity calculated from the inverse equation was off by only \$0.0003 of the actual input of the simulation.

The next step is to calculate the reactivity for the entire run by using just the power histories. In order to do so, we assume the reactor is critical before the input is changed (control rods are moved). Therefore, we assume that 240 seconds prior

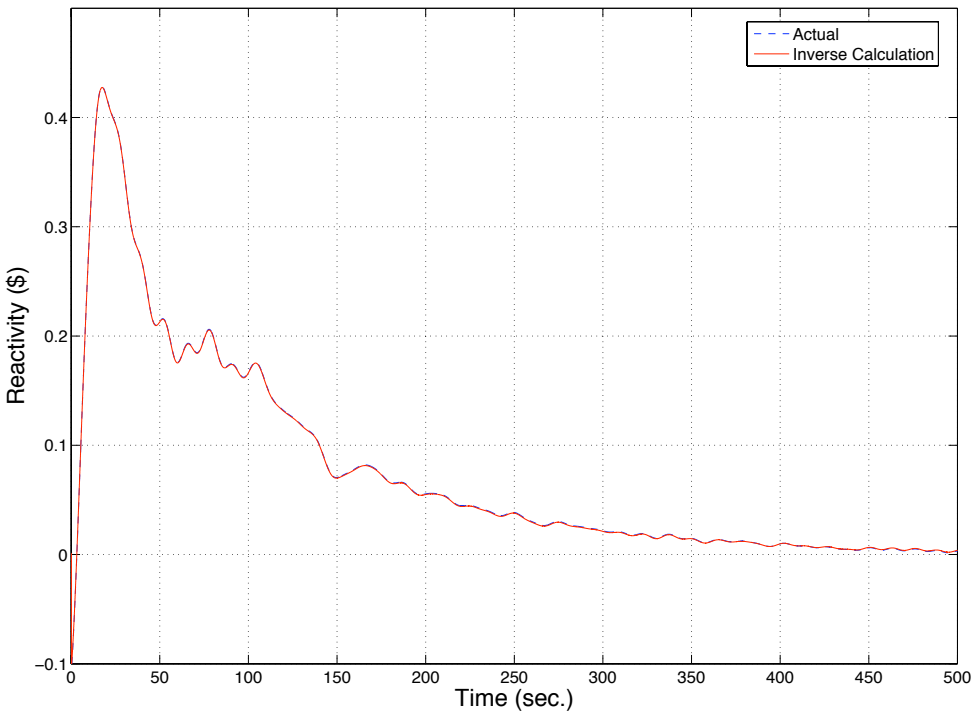

Figure 3.8: Inverse Calculated and Actual Reactivity Values vs. Time.

to, and up to the initial input, the power level within the reactor does not change.

As we can see in Figure 3.8, the calculated reactivity from the inverse equation and the simulation input are pretty much equal. Figure 3.9 plots the error between the two reactivity values shown in Figure 3.8 over time. For the entirety of the calculation, the percent error remains within the ±\$0.002 range. In fact, the RMS error between the two was calculated to be $\$5.25\times10^{-4}$ over the entire run. Therefore, we conclude that our algorithm for solving the inverse equation is working correctly.

Now, with this powerful tool, if we are given a power history for a reactor we can calculate the effective reactivity within the reactor throughout the run. The next

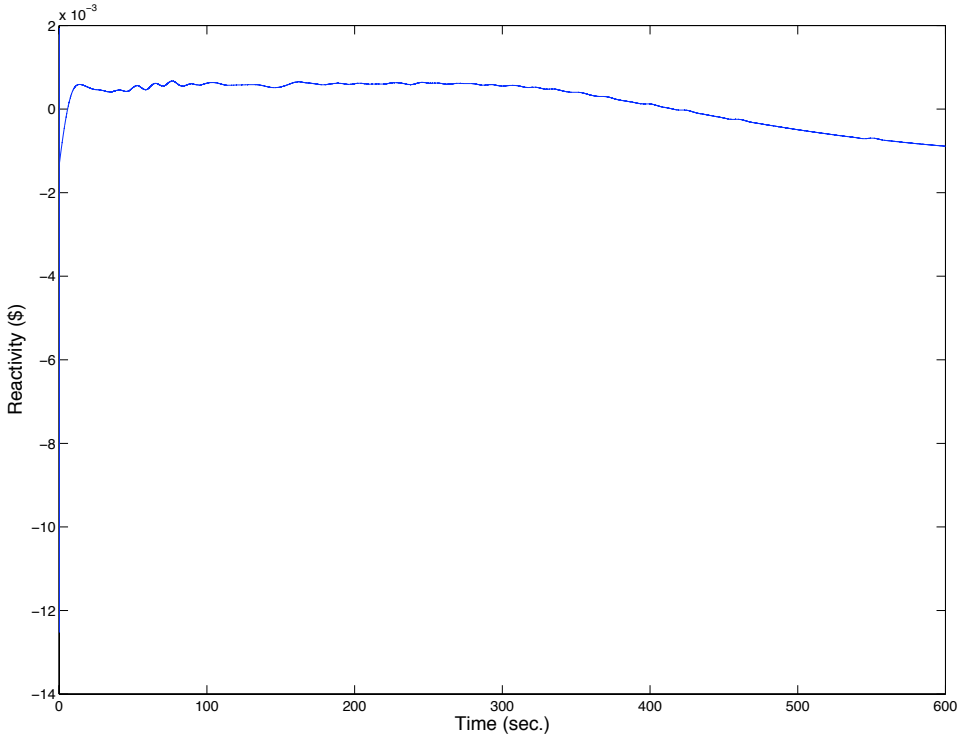

Figure 3.9: Error Between Calculated and Actual Reactivity Values vs. Time.

task is to use the inverse equation to help automatically control the power level of the ACRR.

3.7 Concluding Remarks

The model describing the time-dependent power level of the ACRR was validated by comparing its stable reactor period with that of the actual ACRR, for different reactivity inputs. The temperature vs. power equation when combined with the time-dependent power level model of the ACRR adequately recreated multiple real world ACRR power runs. In addition, the ACRR model was able to recreate a real

world power run containing multiple steady-state power levels.

The inverse equation was broken down into derivative and integral terms and methods for calculating each were described. Then, successful calculations of point reactivity levels and reactivity levels for entire runs confirmed the proper calculation of the inverse equation.

With the ACRR model describing both the temperature feedback and the time-dependent power level, we now move on to developing a controller to automatically control the power level of the ACRR model.

Chapter 4

Controller

4.1 Overview

Before the development of a controller is discussed, we review the current plant model and introduce the presence of a pure time delay within the ACRR system. Then, a state-space representation of the ACRR reactor model is developed to more efficiently simulate the ACRR. We then review the use of the inverse kinetics method to calculate the desired reactivity trajectory and the current amount of effective reactivity within the ACRR. Then, by the use of negative feedback and a PID compensator in the forward path we cause the ACRR model, with the presence of a pure time delay, to track the desired reactivity trajectory throughout the entire run.

4.2 Overview of the Plant Model

First, let us look at how the ACRR model operates as shown in Figure 4.1. As we can see, the inputs for the reactor model are the effective reactivity, ρ_{eff} and the

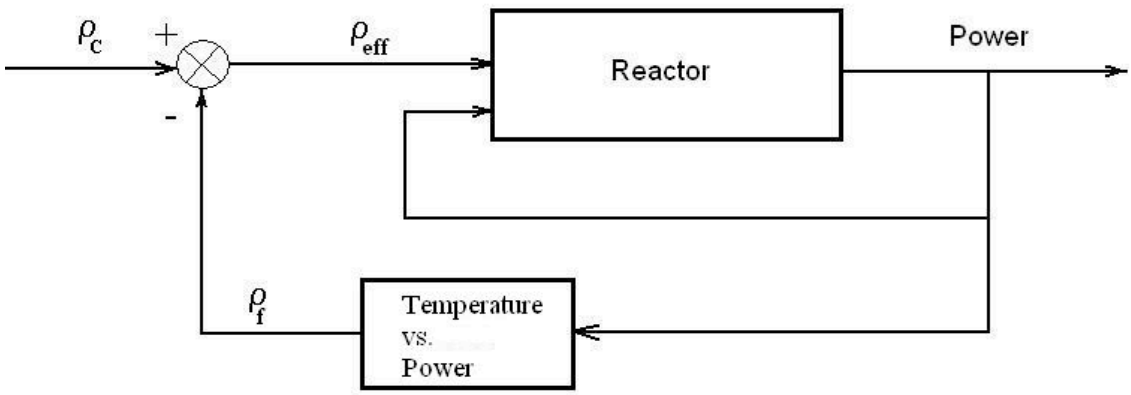

Figure 4.1: Plant Block Diagram.

current power level. The output is simply the new power level. From the new power level and the temperature vs. power equation the negative reactivity feedback, ρ_f, is calculated. Then with the negative reactivity feedback and the amount of reactivity due to the position of the control rods, ρ_c, we obain the amount of effective reactivity that will be added to the plant.

This is a rather simple breakdown of the reactor model but as we recall from chapter 3, this simple representation of the ACRR is sufficient to simulate previous runs reasonably well. The largest difference between the simplified model and the real world ACRR is in the calculation of the average reactor temperature and thus the negative reactivity. Of course this calculation is on the more conservative side since the temperature vs. power equation used to calculate the average reactor temperature is based on steady-state power and temperature values.

In order to have the simplified model better represent the real world ACRR a pure time delay, of about 0.5 seconds, will need to be added to the model. A time delay within a plant often occurs in the model of a system that involves the movement

of some substance, such as in this case, the movement of the control rods [6]. To account for such a delay, we assume that the desired destination of the control rods at the end of the time delay may only be changed every 0.5 seconds, rather than have an instantaneous adjustment every 0.5 seconds.

4.2.1 State-Space Representation

In an effort to more efficiently model the ACRR a state-space representation of the kinetics equation was developed. Also, we found that the time in-between samples may be increased from 0.001 seconds to 0.002 seconds without affecting the simulation results. The kinetics equations are repeated here for the sake of continuity, but in a discrete time representation:

$$\frac{P(k+1)-P(k)}{T} = \left[\frac{\rho(k)-\beta}{\Lambda}\right]P(k) + \sum_{i=1}^{6}\lambda_i C_i(k) \tag{4.1}$$

$$\frac{C_i(k+1)-C_i(k)}{T} = \frac{\beta_i}{\Lambda}P(k) - \lambda_i C_i(k), \qquad i = 1,\ldots,6, \tag{4.2}$$

where T is the time in-between sample k and sample $k+1$. For a state-space representation of the kinetics equations, one must first determine proper state variables and the proper input variables. But before we go any further we need to recall from section 2.3, that Λ is really a function of $\rho(k)$, therefore Λ will vary with time as well. Recall from equation (2.6) that the mean neutron generation time, Λ, is defined as:

$$\Lambda \equiv \frac{l}{\mathrm{k}}, \tag{4.3}$$

where l is the mean lifetime of a neutron in the ACRR and k is the multiplication factor given in equation (2.4). If we recall that the reactivity, ρ, was defined as:

$$\rho = \frac{\mathrm{k}-1}{\mathrm{k}}, \tag{4.4}$$

so that

$$\mathrm{k} = \frac{1}{1-\rho}. \tag{4.5}$$

Using equations (4.3) and (4.5), and moving all the terms that occur at sample k to the right hand side we can re-write equations (4.1) and (4.2) as:

$$P(k+1) = P(k) + \left[\frac{\mathrm{k}(k)(1-\beta)-1}{l}\right]P(k)T + \sum_{i=1}^{6}\lambda_i C_i(k)T \tag{4.6}$$

$$C_i(k+1) = C_i(k) + \frac{\mathrm{k}(k)\beta_i}{l}P(k)T - \lambda_i C_i(k)T, \qquad i = 1,\ldots,6. \tag{4.7}$$

Equations (4.6) and (4.7) will now be used to develop a proper state-space representation with $P(k), C_1(k), C_2(k), \ldots,$ and $C_6(k)$ being the state variables and $\mathrm{k}(k)$ being the input. Rewriting Equations (4.6) and (4.7) in a state-space representation we get the following equation:

$$\begin{bmatrix} C_1(k+1) \\ C_2(k+1) \\ C_3(k+1) \\ C_4(k+1) \\ C_5(k+1) \\ C_6(k+1) \\ P(k+1) \end{bmatrix} = \mathbf{A}\begin{bmatrix} C_1(k) \\ C_2(k) \\ C_3(k) \\ C_4(k) \\ C_5(k) \\ C_6(k) \\ P(k) \end{bmatrix} + \mathbf{N}\begin{bmatrix} C_1(k) \\ C_2(k) \\ C_3(k) \\ C_4(k) \\ C_5(k) \\ C_6(k) \\ P(k) \end{bmatrix}\mathrm{k}(k), \tag{4.8}$$

where,

$$\mathbf{A} = \begin{bmatrix} 1-\lambda_1 T & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-\lambda_2 T & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-\lambda_3 T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-\lambda_4 T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-\lambda_5 T & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-\lambda_6 T & 0 \\ \lambda_1 T & \lambda_2 T & \lambda_3 T & \lambda_4 T & \lambda_5 T & \lambda_6 T & 1-\frac{T}{l} \end{bmatrix} \tag{4.9}$$

and

$$\mathbf{N} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \frac{\beta_1 T}{l} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\beta_2 T}{l} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\beta_3 T}{l} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\beta_4 T}{l} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\beta_5 T}{l} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\beta_6 T}{l} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{(1-\beta) T}{l} \end{bmatrix}. \tag{4.10}$$

As an aside, if we look closely at equation (4.8) we can see that it is of the form

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \sum_{i=1}^{m} \mathbf{N}_i u_i(k)\mathbf{x}(k), \tag{4.11}$$

where $\mathbf{B} = [0]$ and $m = 1$. Equation (4.11) is state-space representation of a discrete-time invariant bilinear system. Bilinear systems are systems that are linear in control and in state but not jointly linear in state and control. Thus, the state-space model developed above is also a bilinear system. More precisely the system representing the ACRR is a homogenous bilinear system, due to the fact that $\mathbf{B}$ is a zero matrix [7].

4.3 Using the Inverse Method

Let us revisit the inverse method that was introduced at the end of chapters 2 and 3 in equations (2.32) and (3.8) respectively. The inverse equation is repeated here for convenience sake:

$$\rho(t) = \beta + \Lambda \frac{d}{dt}[\ln P(t)] - \beta \int_0^\infty D(\tau) \frac{P(t-\tau)}{P(t)} d\tau, \tag{4.12}$$

where the delayed neutron kernel $D(\tau)$ is as follows

$$D(\tau) \equiv \sum_{i=1}^{6} \frac{\lambda_i \beta_i}{\beta} e^{-\lambda_i \tau}. \tag{4.13}$$

As we know we can use equation (4.12) to calculate the amount of effective reactivity present within a reactor from the reactor power histories. So let us do so with an 80% run from chapter 3 shown in Figure 4.2.

Figure 4.2 compares the power vs. time plots for both the actual ACRR run and the simulated ACRR run. As we can see the steady-state values are very close, but up until the about the 350 second mark the plots are very different. This is due to the fact that both power runs were given entirely different effective reactivity inputs even though the simulated run was given the same control rod input as the actual run. This is due to, once again, the temperature vs. power equation that models the temperature feedback based on steady-state temperature and power values. Therefore, the simulated temperature feedback will likely be more pronounced in the earlier stages of the run since in the real-world the temperature feedback develops much more slowly over time.

Utilizing the inverse kinetics method we can determine effective reactivity values within the actual ACRR for the power run shown in Figure 4.2. Figure 4.3 plots the actual effective reactivity input to the ACRR, which was calculated by using the

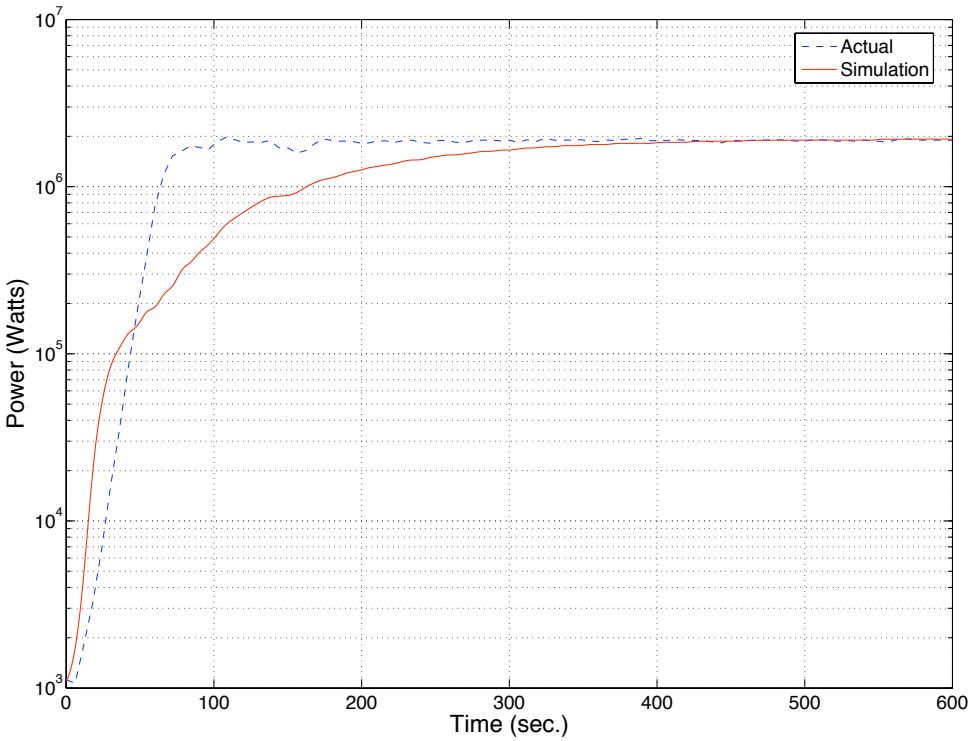

Figure 4.2: Power vs. Time for Steady-State Run 8512 (80%).

inverse kinetics method, and the simulation effective reactivity input to the ACRR model. As we can see the effective reactivity plots give us a lot insight into Figure 4.2. First the reactivity level for the simulation rises a lot faster than the actual reactivity level and thus the simulated power level rises a lot quicker than the actual power level. Then the reactivity level for the simulation quickly drops and thus causes the power level to increase at a much slower rate.

Now let us take the effective reactivity found from the inverse method, shown in Figure 4.3 and have it be the effective reactivity input to the ACRR model. Figure 4.4 compares the actual power levels and the recreated power levels that were generated using the effective reactivity inputs. In Figure 4.4, we see that there is a

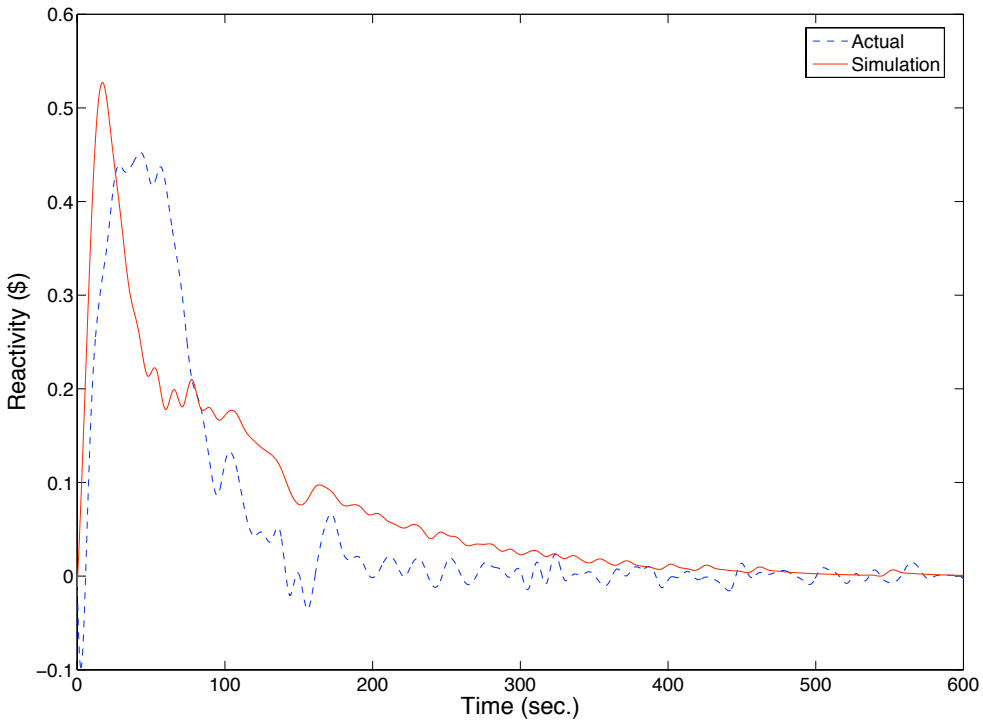

Figure 4.3: Actual and Simulation Effective Reactivity Input for Steady-State Run 8512 (80%).

power difference between the actual and recreated power levels and Figure 4.5 plots the percent error between the two power levels over time. As we can see the percent error between the two power levels remains within the ±5% range for the entire run. The power level difference, is likely due to the fact that the inverse kinetics method is assuming six delayed neutron precursor groups to calculated the effective reactivity input. If we were to use more delayed neutron precursor groups to calculate the input and model the ACRR, the recreated power trajectory would more closely resemble the actual power run.

As we can see, if we are able to input the same effective reactivity values as the actual run, we can recreate a similar power trajectory by using the ACRR model.

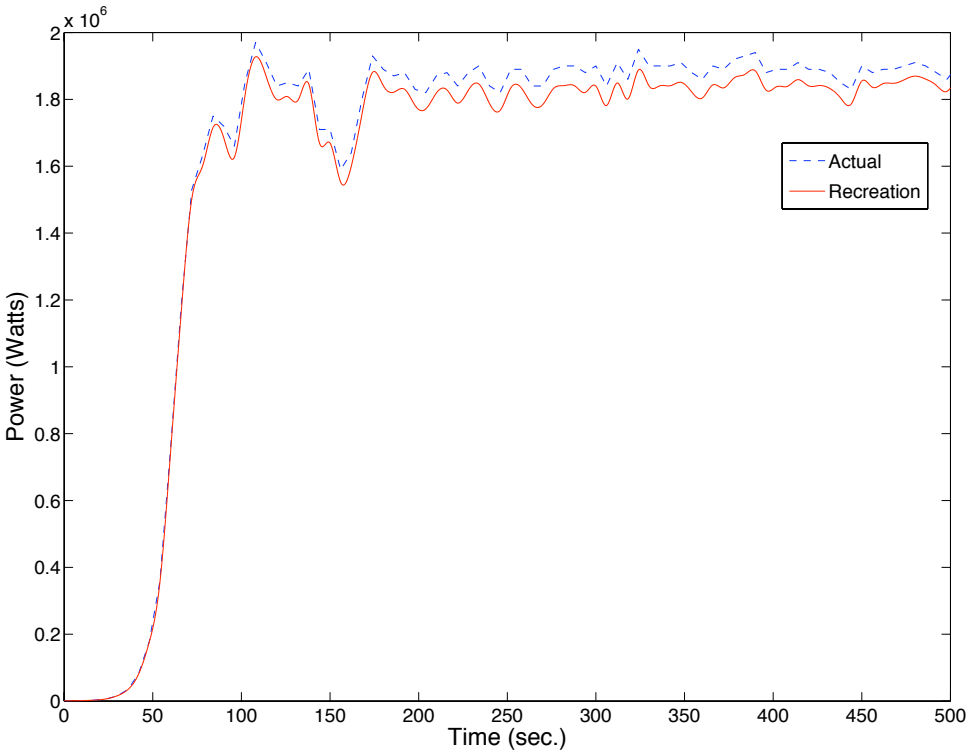

Figure 4.4: Actual and Recreated Power Levels for Steady-State Run 8512 (80%).

Therefore, we need to determine how to calculate the current amount of reactivity within the ACRR model and automatically adjust the position of the control rods to meet the required effective reactivity levels throughout the run. The inverse method can be used to calculate the current amount of reactivity within the ACRR model from the power output histories. By the use of negative feedback and a PID compensator in the forward path we can cause the ACRR model to effectively track the desired reactivity trajectories throughout the entire run.

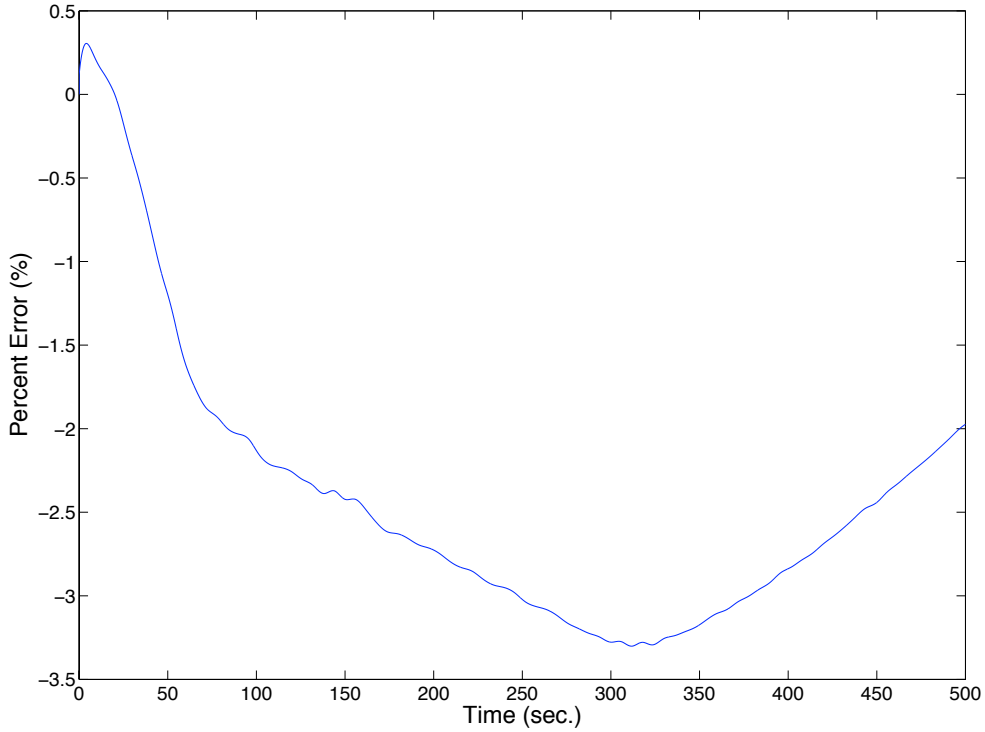

Figure 4.5: Percent Error between Actual and Recreated Power Levels vs. Time.

4.4 Developing the PID compensator

Figure 4.6 is a block diagram of the plant and the PID compensator. In this portion we discuss the needed steps before the PID compensator may be tested. Creation of the desired power trajectory, calculation of the proper desired reactivity trajectory, the development of the plant input, the proper management of the power output, the treatment of the error signal, and the management of the position of the control rods will all be discussed. The PID output dictates the proper reactivity adjustments needed for the plant to follow the desired reactivity trajectory. The reactivity adjustments are converted to control rod adjustments by the \$0.003/rod unit ratio. Then, the control rod adjustments are made to the predetermined plant input to ensure the proper reactivity levels are met.

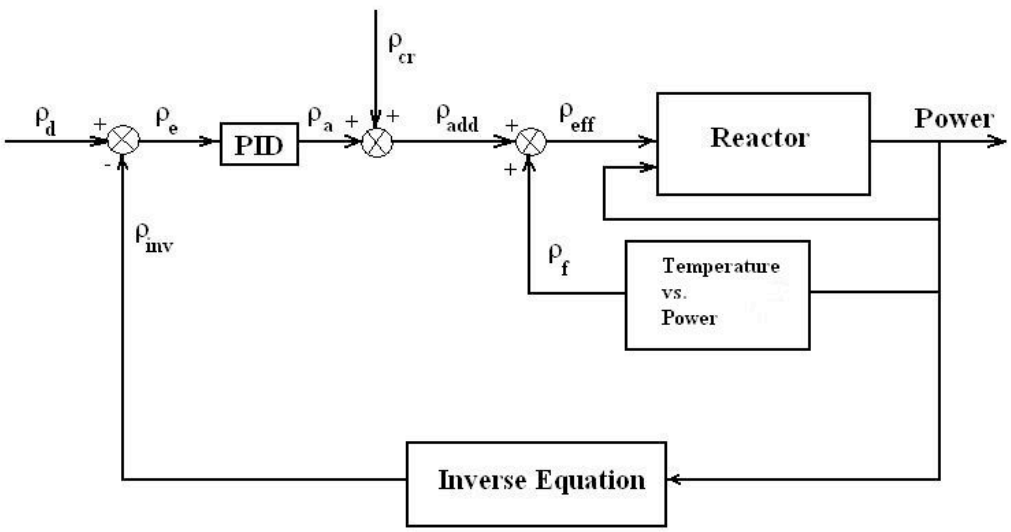

Figure 4.6: Plant and Controller Block Diagram.

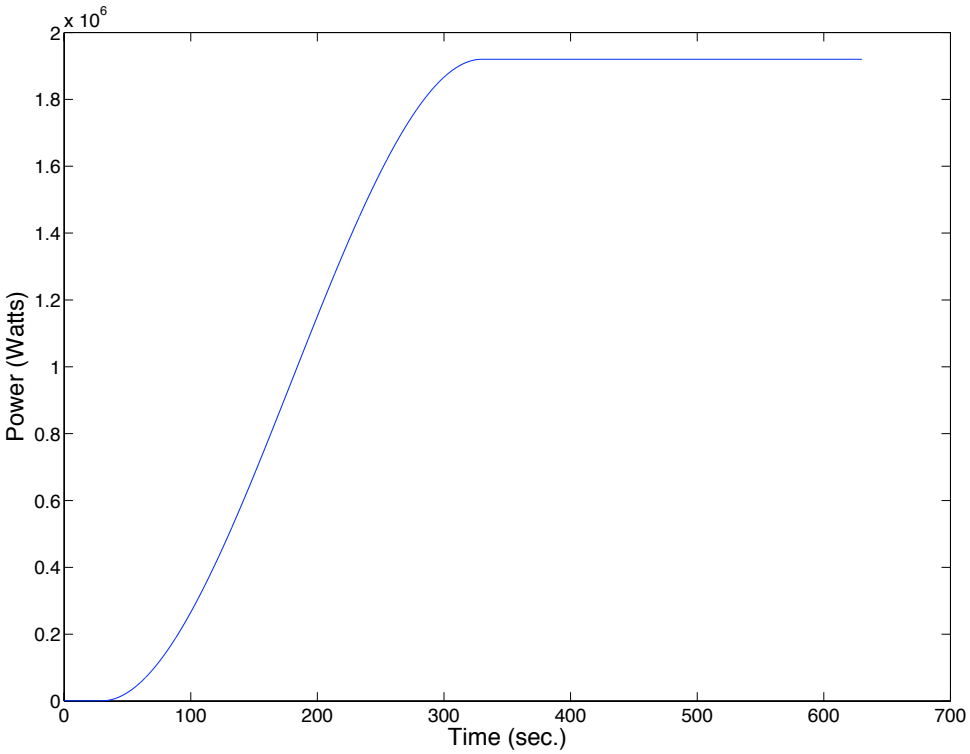

Figure 4.7: Desired Power vs. Time.

4.4.1 Creation of the Desired Power vs. Time Path

For this step we create a MATLAB function, *spline4*, that creates a cubic polynomial trajectory from an input of an initial power level, a final power level, a time step, and the total time to travel from the initial power level to the final power level.[1] We chose a cubic polynomial trajectory since it is a relatively easy input to derive since the rate of change at the beginning and end of the trajectory is assumed to be zero [8]. Figure 4.7 is an example of a power plot that starts at 1 kW and ends at 1.92 MW.

[1]The function *spline4* is given in Appendix B

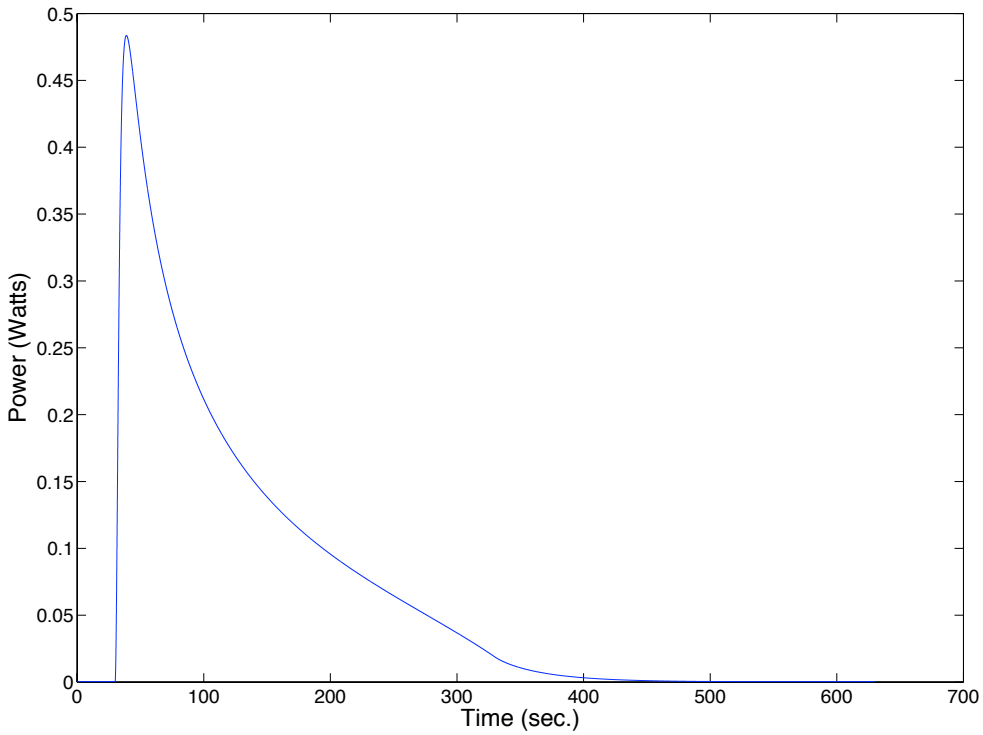

Figure 4.8: Desired Reactivity vs. Time.

4.4.2 Creation of the Desired Reactivity Path

The determination of the desired reactivity, at each time step, was accomplished by using the inverse equation to calculate the amount of reactivity within the reactor from the desired power trajectory. In order to properly use the inverse equation, we assume that the reactor is at steady-state for about 4 minutes before the power level begins to rise. This means that we need to add an extra 240 seconds of a constant initial power level to the beginning of the power vs. time history. Figure 4.8 is the resulting desired reactivity path, calculated by the script file *inverse2.m*, for the desired power level shown in Figure 4.7.[2]

[2]The script file *inverse2.m* is given in Appendix B.

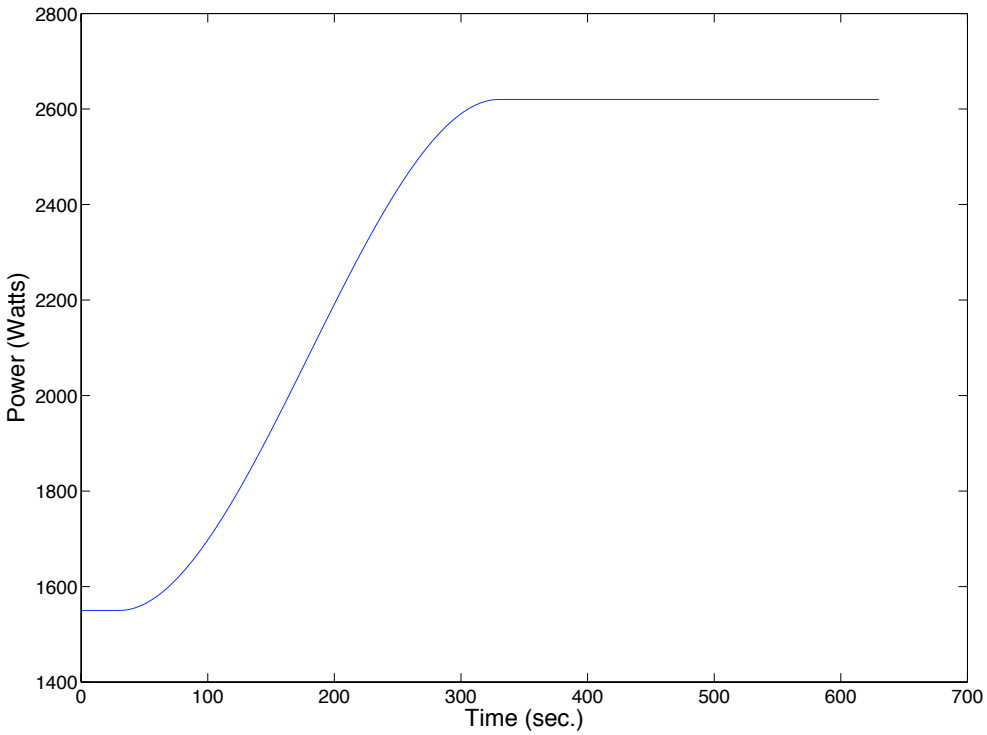

Figure 4.9: Predetermined Control Rod Input.

4.4.3 Development of the Plant Input

For a simple test run, such as the one in Figure 4.7, the initial and final positions of the control rods are determined from the real world steady-state control rod positions of ACRR runs at a similar power level. Then we use the function *spline5* to create a cubic polynomial control rod trajectory.[3] The goal here is to develop a predetermined control rod path and have the compensator make the necessary adjustments to the position of control rods as the plant reactivity deviates from its desired path. Figure 4.9 is an example of the predetermined plant input for a test run that takes the reactor from 1 kW up to 1.92 MW.

[3]The function *spline5* is given in Appendix B

4.4.4 Management of the Power Output

For the model output, an initial power history needs to be created. Again, the assumption is that the plant has been at steady-state for a least 4 minutes. This allows for a proper calculation of the inverse equation so that the necessary current plant reactivity level may be determined even at the beginning of the simulation. The output of the plant is stored in a column vector and by using the function *invequ*, we calculate the ACRR's current reactivity level at every 0.5 second interval throughout the simulation.[4]

4.4.5 Treatment of the Error Signal

The reactivity error is found every 0.5 seconds due to the pure time delay within the ACRR system. The desired reactivity trajectory is calculated every 0.002 seconds but during the run the error value is calculated at a 0.5 second interval along the desired reactivity trajectory.

The integral and derivative of the reactivity errors versus time are calculated by simple numerical methods. The derivative of the error is calculated by taking the difference of the current error and the previous error value in time and then dividing the difference by the time step in-between each error calculation.

The integration of the error signal requires a little more work, since the 0.5 second delay between error calculations did not allow for the familiar backward integration method to be used. Therefore, a trapezoidal integration method is used.[5] The trapezoidal method is essentially approximating the area underneath the error function from sample $k-1$ to k as a trapezoid [5]. Then through trial and error runs the respective proportional, derivative, and integral gains for the PID compensator

[4]The function *invequ* is given in Appendix B.

[5]For a detailed description on trapezoidal integration see Appendix A.1.

are found to be 0.5, 0.35, and 3.7 respectively.

4.4.6 Control Rod Adjustments

Let's review, due to the pure time delay within the ACRR system, we assume that we can only make an adjustment to the control rods final destination or desired position, every 0.5 seconds, rather than have an instantaneous adjustment every 0.5 seconds and track the progress over the delay.

We begin with a predetermined control rod trajectory. Then, due to the compensator output, we adjust the desired control rod position for the next sample. To accomplish this, let us again employ the use the MATLAB function *spline*. From the current position, the adjusted future position, and the time period of the delay; the trajectory for the control rods at every 0.002 second interval over the delay period is calculated. This assumes the control rods will follow the calculated cubic spline trajectory over the delay period. In essence, what we are telling the control rods, is go from current position x to final position y in 0.5 seconds and we are assuming that the control rods will follow the piecewise polynomial trajectory created by the *spline* function.

4.4.7 Results

Let us now see how the PID compensator performs by using the script file *reactor6.m*, to simulate the automatic control of the ACRR.[6] In this section we will review an 80% power run and a special 50% power run that includes an additional sine power curve that makes one ± 0.24 MW oscillation about the 50% power level (1.2 MW) once it reaches steady-state.

[6]The script file *reactor6.m* is given in Appendix B.

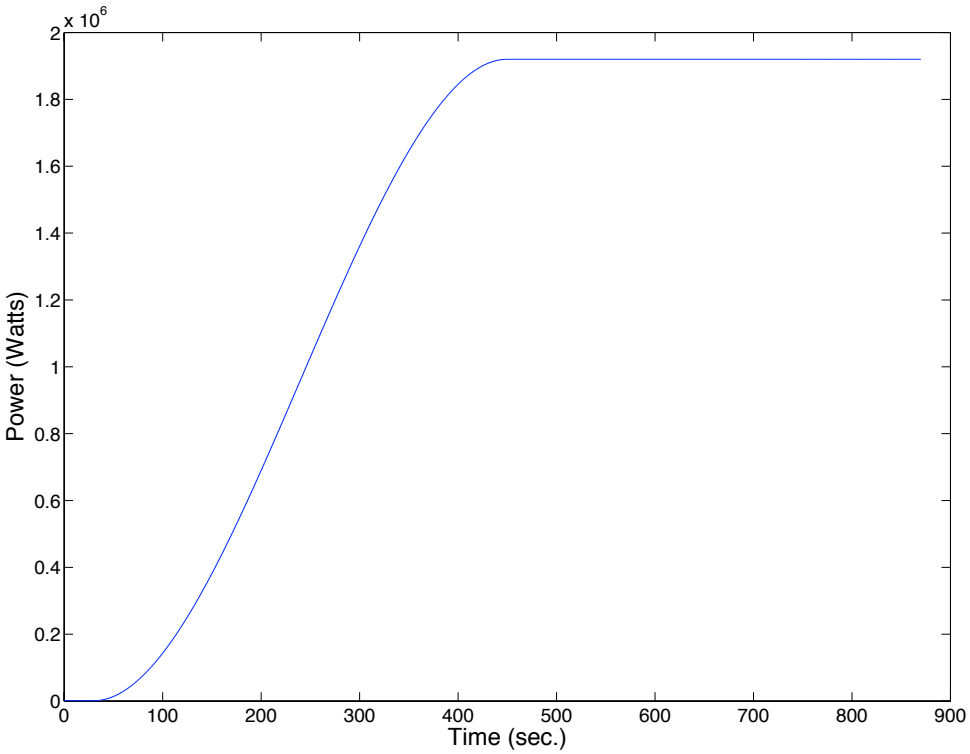

Figure 4.10: Desired Power Level for an 80% Test Run (1.92 MW).

80% Power Run

The power trajectory we want to follow is shown in Figure 4.10. In this case we are keeping the reactor at steady-state for 30 seconds in the beginning to ensure equilibrium power and temperature levels are maintained before increasing the power level. Then, the power follows a cubic trajectory from 1 kW to 1.92 MW over a period of 7 minutes. The reactor is then held at 1.92 MW for another 7 minutes to complete the run.

Figure 4.11 shows the desired reactivity trajectory that results in the desired power trajectory and the actual reactivity levels throughout the run. Figure 4.12

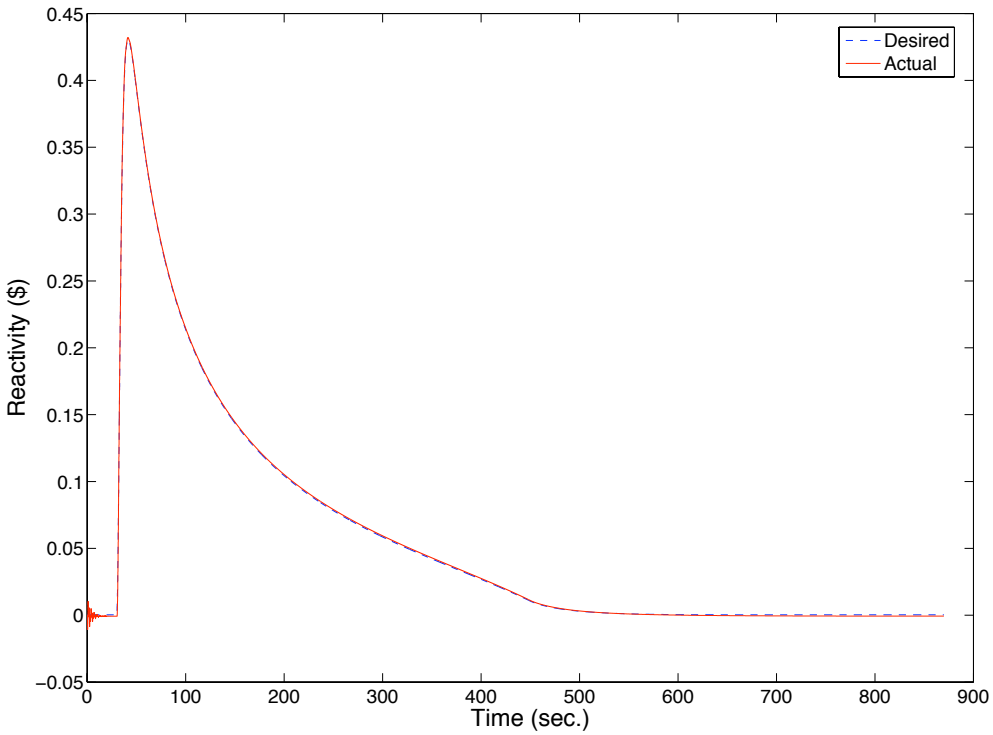

Figure 4.11: Desired and Actual Reactivity Levels vs. Time for 80% Run (1.92MW).

plots the error between the desired and actual reactivity levels throughout the run. As we can see, in Figure 4.11, the compensator is able to closely follow the desired reactivity trajectory throughout the run. In Figure 4.12, the error is relatively large during the initial steady-state portion of the run. These larger error values are due to the relatively small reactivity level we are trying to track and the fact that the temperature feedback is not considered until after the simulation begins.

Once the simulation begins, the reactivity feedback begins to degrade the initial steady-state effective reactivity level. Thus, the initial effective reactivity level declines throughout the first half second of the simulation until the compensator can begin tracking the effective reactivity level. In the real world ACRR, this initial

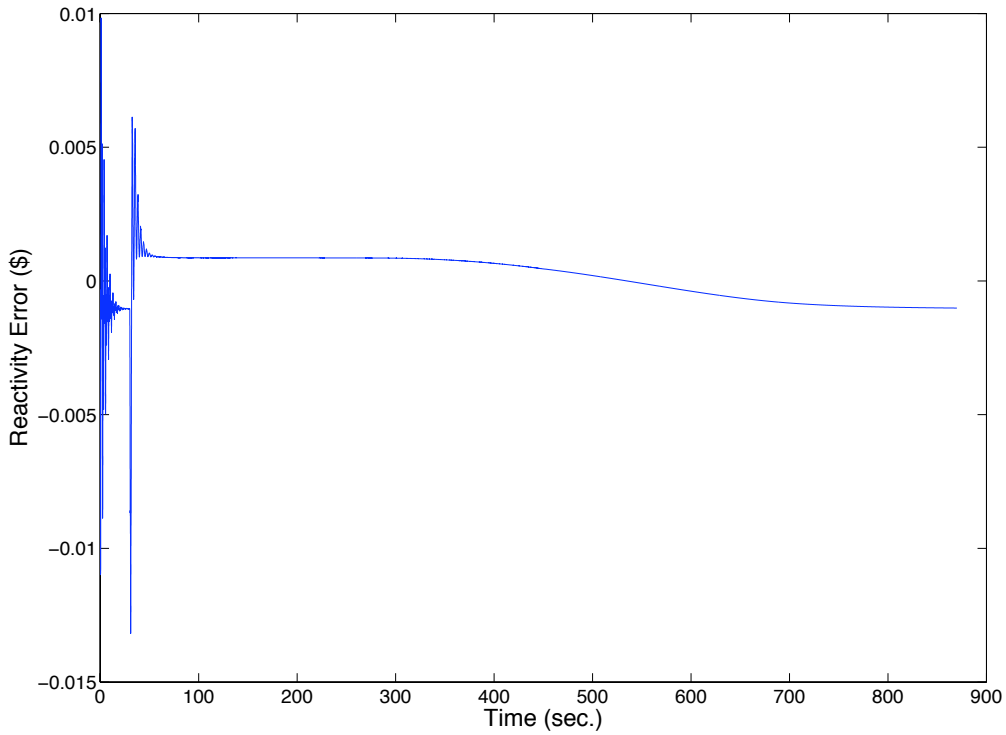

Figure 4.12: Reactivity Error vs. Time for 80% Run (1.92 MW).

steady-state offset would not occur since we are assuming the reactor operator has already brought the ACRR to an initial steady-state power level and then activates the inverse method and the PID compensator to automatically control the ACRR reactivity level. Of course, the reactor operator may have to wait a long time for the ACRR to reach an initial steady-state power level. Therefore, it is likely some initial offset between the ACRR reactivity levels may be present, but for the given run, this is most likely a "worst case" example.

Let us now go back to Figure 4.12. As we can see, once the reactivity levels begin to climb (the 30 second mark) the compensator does a good job regulating the error between the actual and desired reactivity levels to nearly zero. For the

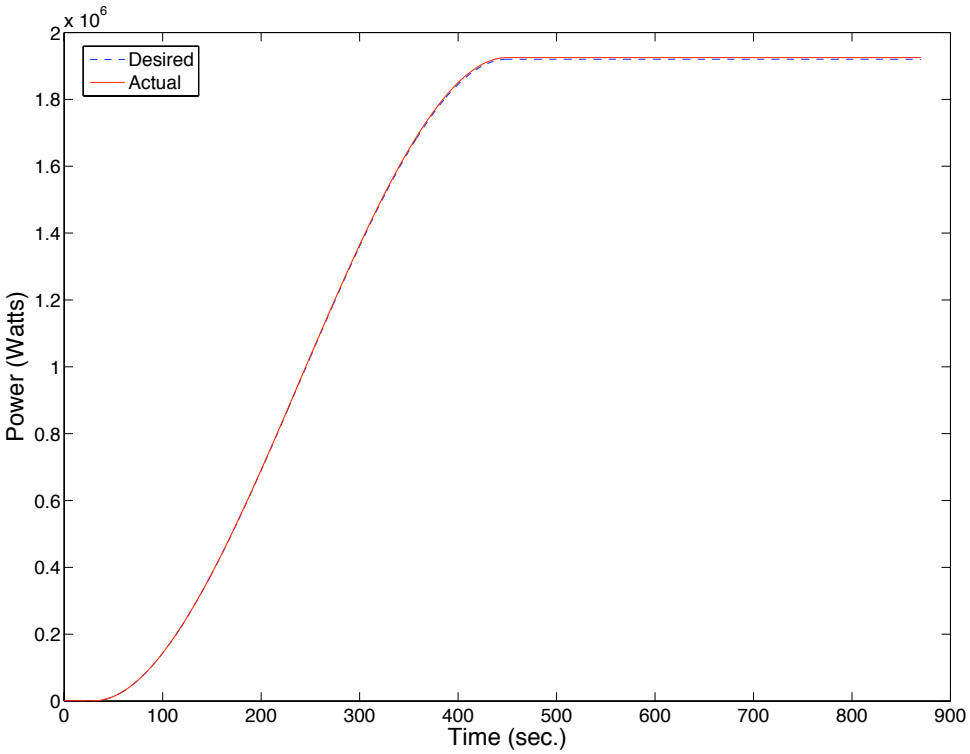

Figure 4.13: Desired and Actual Power Levels vs. Time.

majority of the run, the error is well within the ±\$0.005 range. In fact, the RMS error between the desired and actual reactivity levels is only $\$6.6 \times 10^{-4}$ for the entire run. Unfortunately, even a relatively small RMS error such as this still has a negative impact on the desired power level.

Figure 4.13 shows the comparison between the desired and actual ACRR power levels and Figure 4.14 plots the percent error between the two power levels vs. time for the 80% power run. As we can see in Figure 4.14, the reactivity error in the beginning of the run does causes the error between the desired and actual ACRR power levels to be off, but due to the relatively low desired reactivity levels, during this initial steady-state portion, the resulting power error is minimal. The actual

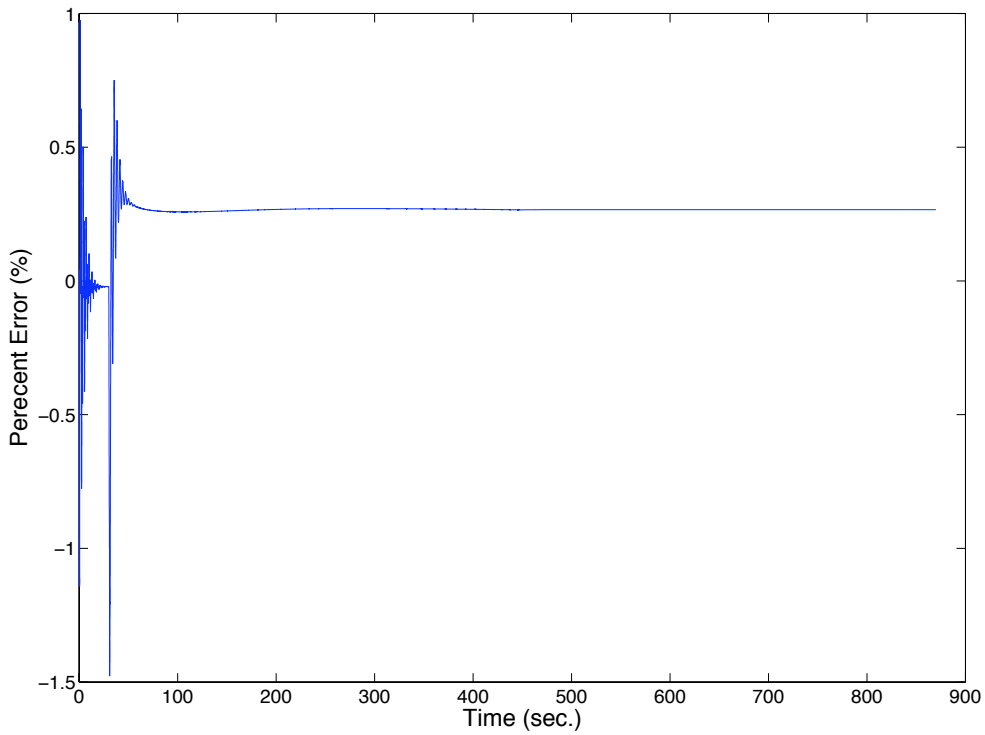

Figure 4.14: Power Percent Error vs. Time.

ACRR power level, for the majority of the run, is often within ±1% of its desired power level. Of course as we can see, since we are only regulating the reactivity level within the ACRR, there is always some amount of error present between the desired and actual ACRR power levels. For this run, the RMS error is approximately 4.2 kW over the entire run and the resulting steady-state error is approximately 5 kW or +0.3% of the desired steady-state power level. Therefore, errors in the overall reactivity levels result in a deviation between the desired and the actual ACRR power levels, but in this case the difference is negligible.

Figure 4.15 plots the control rod adjustments over time. As we can see the adjustments are relatively small and other than the sharp increase right when the

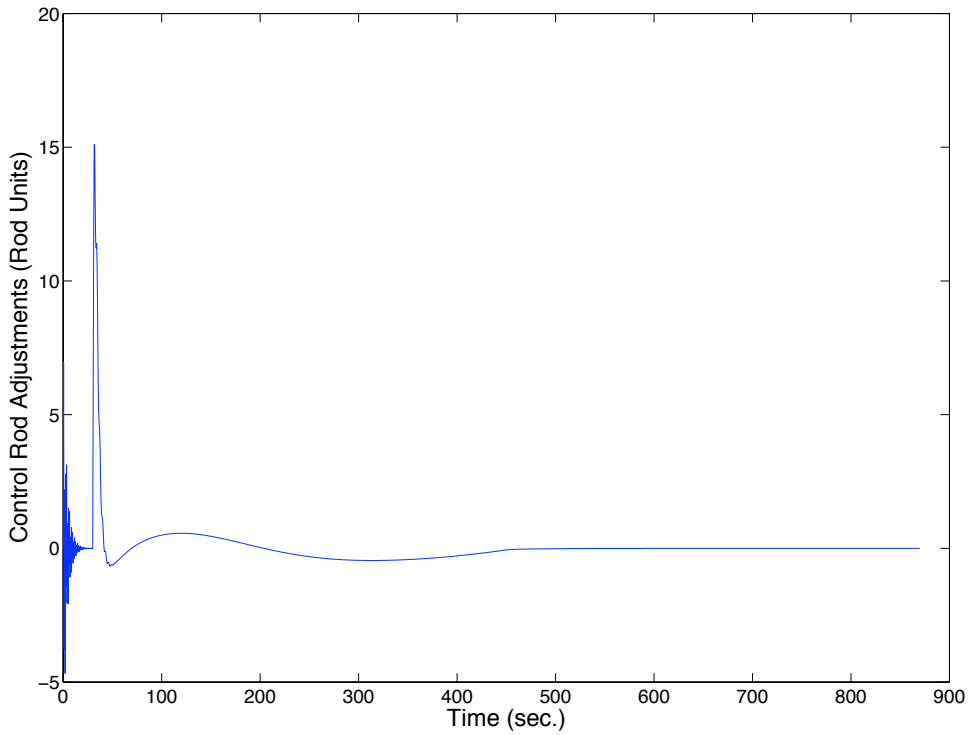

Figure 4.15: Control Rod Adjustments vs. Time for an 80% Run (1.92 MW).

desired reactivity starts to increase, we have a fairly smooth adjustment plot. The large sharp increase right around 30 seconds is due to the delay in the plant, since the desired reactivity is synced up directly with the delay in the plant, i.e. the change in desired reactivity occurs just as the next delay in the plant begins. Therefore, we are stuck with waiting a whole 0.5 seconds before the error between the desired and actual reactivity levels is noticed. To help lessen the sharp change, we could alter the desired reactivity path so that the initial reactivity change does not happen at a 0.5 second interval.

Also, in Figure 4.15, we can see that there is a lot of effort in the beginning of the run to regulate the ACRR to its initial steady-state reactivity level. This is again

due to the fact that the temperature or reactivity feedback is not considered until after the simulation begins.

One of the concerns in implementing this compensator into the real ACRR, is the rate of change in the control rods. The maximum rate of change allowed is approximately 40 rod units/second. The maximum rate of change for the 80% power run shown in Figure 4.13 is about 30 rod units/second. If there was a concern that the control rod rate might exceed the maximum value it could be easily compensated for by simply lengthening the amount of time it takes to reach the desired steady-state power level. In fact, the desired power trajectory shown in Figure 4.7 was first used, but when the maximum rate of change for the control rods exceeded 40 rod units/second, the time to full power was increased from 5 minutes to 7 minutes to create the power trajectory given in Figure 4.10.

In addition, we also experimented with three different numerical integration methods to determine the integral of the error signal: backward rectangular, trapezoidal, and parabolic.[7] The backward rectangular method was not very effective at all due to the large time step in-between the calculation of the error. The trapezoidal and parabolic methods both worked well and there was little to no improvement in the RMS error for the reactivity levels between the trapezoidal and parabolic methods.

[7] A detailed derivation of the parabolic integration method is given in Appendix A.2.

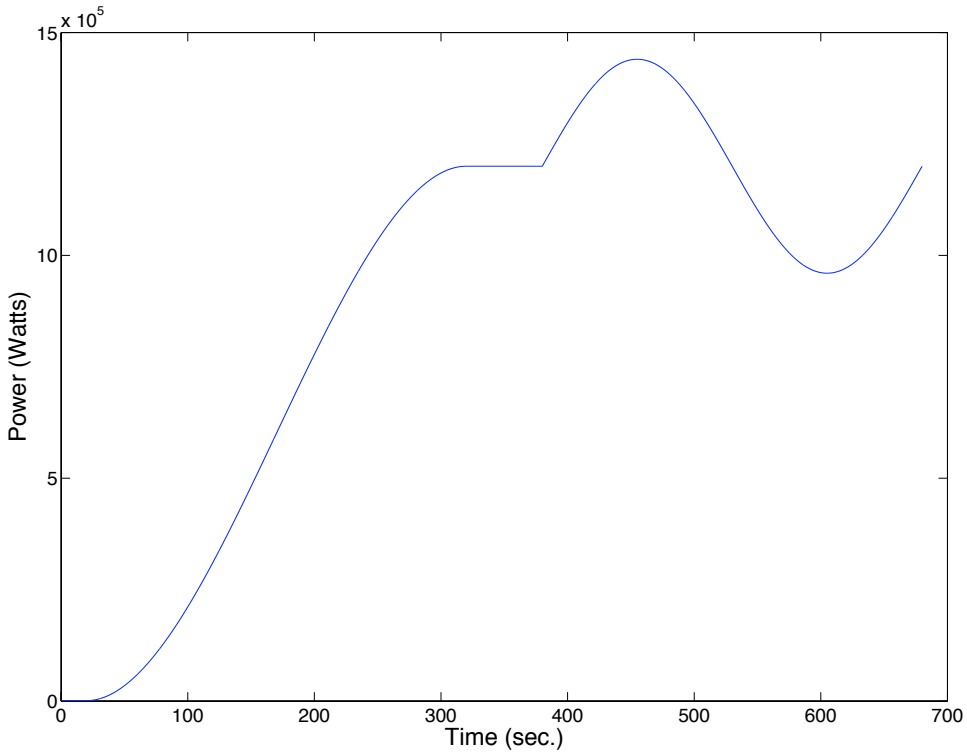

Figure 4.16: Desired Power Trajectory for a Sine Oscillation About the 50% Power Level.

50% Power Run and Sine Wave

The power trajectory we wish to follow is shown in Figure 4.16. In this run we are keeping the reactor at steady-state for 20 seconds in the beginning to ensure equilibrium power and temperature levels are maintained before increasing the power level. Then, the power follows a cubic trajectory from 1 kW to 1.2 MW over a period of 5 minutes. The reactor is then held at 1.2 MW for one minute before it oscillates about the 1.2 MW power level up to 1.44 MW, down to 0.96 MW, and then back to 1.2 MW to complete the run.

Figure 4.17 shows the desired reactivity trajectory that results in the desired

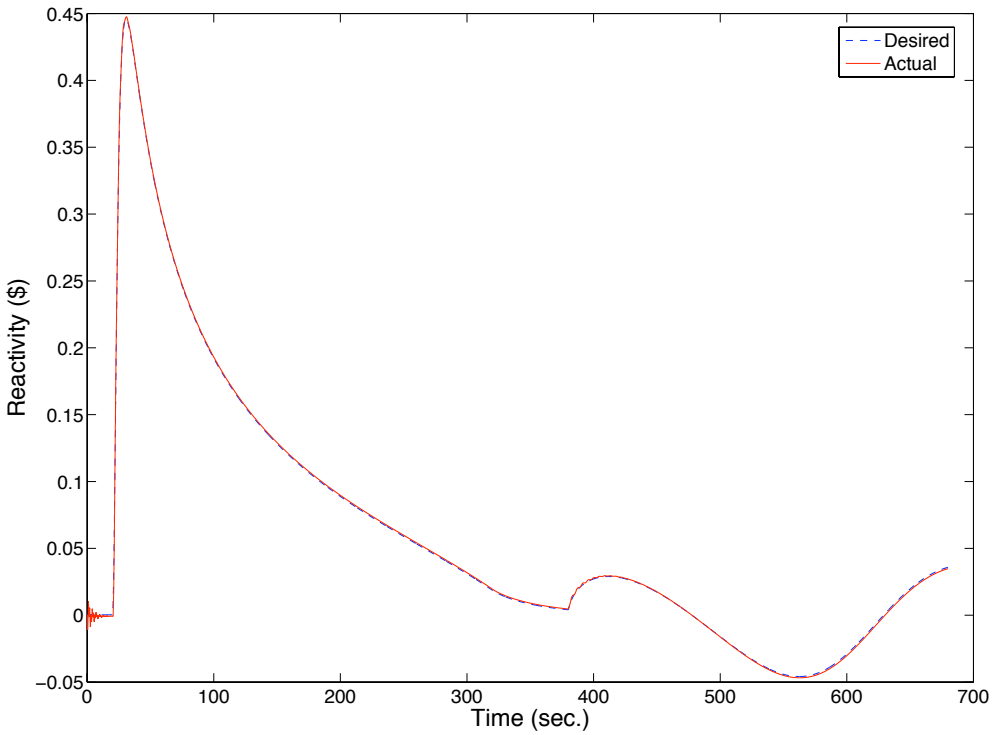

Figure 4.17: Desired and Actual Reactivity Levels vs. Time for Sine Run.

power trajectory and the actual reactivity trajectory. Figure 4.18 plots the percent error between the desired and actual reactivity levels throughout the test run. As we can see the compensator does a good job of following the desired reactivity trajectory. From Figure 4.18, we can see, once again, the compensator has a little trouble tracking the reactivity levels due to the initial steady-state error. This is once again due to the small magnitude of the initial reactivity value and the effect of temperature feedback.

In Figure 4.18, we can see once the simulation reaches the 20 second mark the compensator does a good job regulating the error to nearly zero. Once it reaches the sine portion of the run the compensator has some trouble tracking the desired

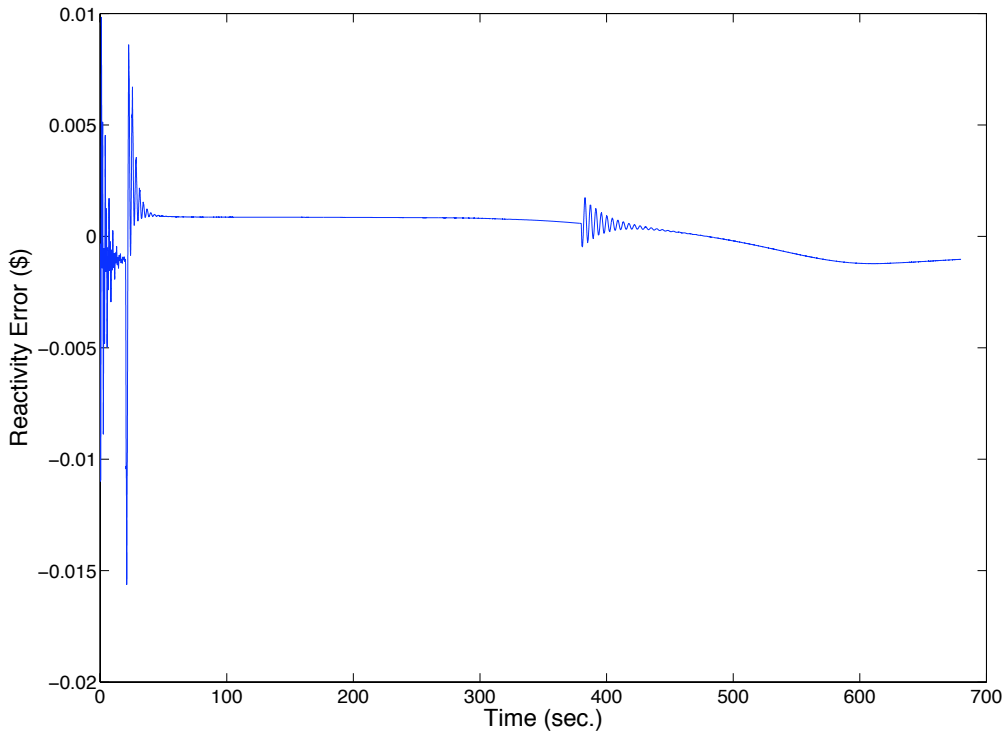

Figure 4.18: Reactivity Error vs. Time.

trajectory but it quickly recovers. For the majority of the run the error is between the ±\$0.005 range, even during the sine portion of the run. In addition, the RMS error between the desired and actual reactivity levels is $\$8.4 \times 10^{-4}$ over the entire run.

Figure 4.19 shows the comparison between the desired and actual ACRR power levels and Figure 4.20 plots the percent error between the two power levels for the sine wave power run. As we can see the percent error plot is similar to the one for the 80% power run. For the majority of the run, the actual ACRR power level is within ±1% of the desired power level and even during the sine portion of the run, the actual power level is within a close proximity of the desired power level. For this

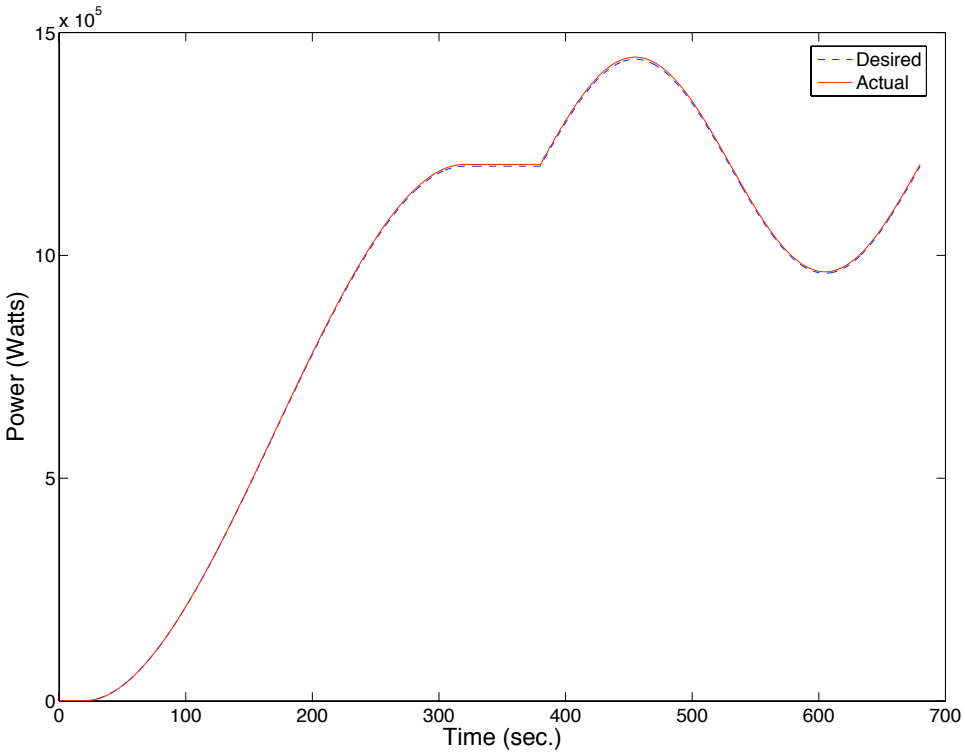

Figure 4.19: Desired and Actual Power Levels vs. Time.

run the RMS power error is approximately 3.4 kW and the resulting steady-state error, once the reactor reaches the 1.2 MW level, is approximately 4 kW or +0.3% of the desired steady-state power level. Once again, we can see in Figure 4.20, the resulting error in tracking the desired reactivity levels does have a negative impact on the desired power trajectory, but since a resolution of ±1%, for the majority of the run, is likely sufficient for this application, we can conclude that the compensators performance is respectable.

Figure 4.21 plots the control rod adjustments that were made throughout the run in order to follow the desired reactivity trajectory. As we can see in the beginning a lot of effort is made in driving the ACRR reactivity level to zero. Then, we see the

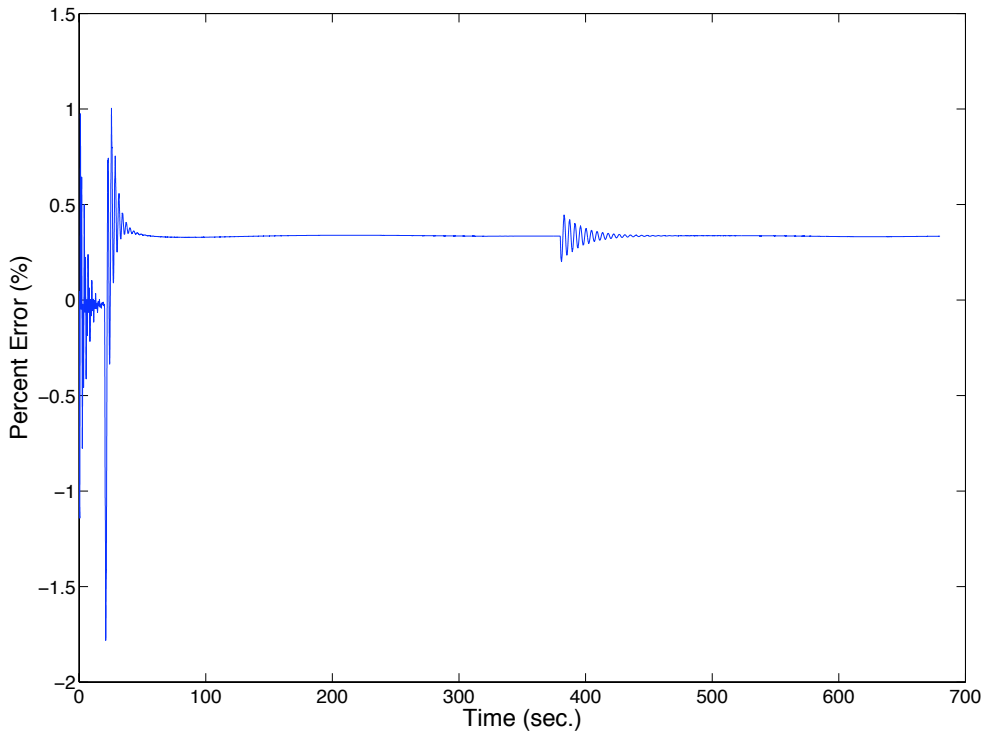

Figure 4.20: Power Percent Error vs. Time.

large increase right at 20 seconds due to the compensator making a large adjustment to catch up with the desired reactivity trajectory. After the large spike, the control rods adjustments are rather smooth unit we encounter the sine wave portion of the reactivity trajectory. As we see the compensator has a little trouble following the desired trajectory at the beginning of the sine curve, but it soon adjusts and begins following the trajectory rather well. On a separate note, we can see that in the beginning of the simulation it takes a rather large amount of time and effort to regulate the ACRR reactivity level to the desired steady-state level. One option would be to add an additional 10 seconds to the time that the reactor is initially at steady-state, much like the previous 80% power run. But, as we noted before, in the previous power run, this is most likely a worst case example, since the reactivity

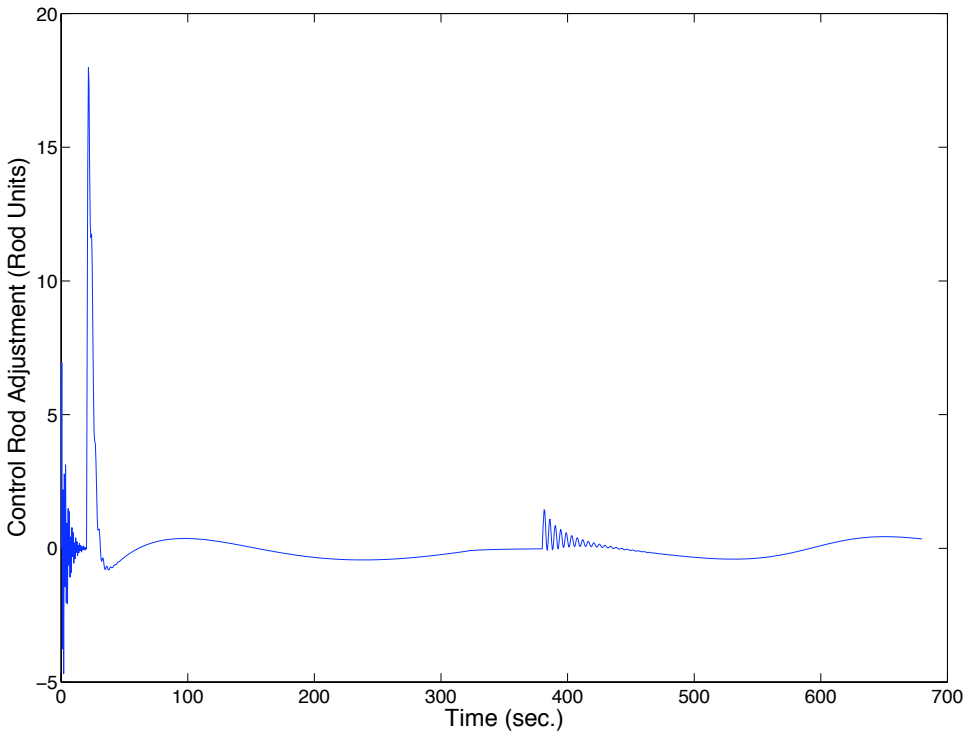

Figure 4.21: Control Rod Adjustments vs. Time for the Sine Wave Run.

feedback is not accounted for until the simulation begins.

As we noted before, an area of concern is keeping the velocity of the control rods within their physical limits. As we know the maximum velocity of the control rods is about 40 rod units per second. For the sine wave run the maximum velocity of the control rods was 36 rod units per second.

4.5 Concluding Remarks

By utilizing the inverse kinetics method to calculate the current reactivity levels within the ACRR and the use of PID compensation in the forward path, we are able to automatically control the ACRR power level by tracking a predetermined reactivity trajectory. Even with the presence of a time delay within the plant, the compensator is able to track the desired reactivity level quite closely. The small tracking error, unfortunately, does have a negative effect on the desired steady-state power level. Of course, due to the relatively small amount of steady-state error, the hope is that once the reactor reaches its final power level after being driven by the compensator, the difference will be negligible or the reactor operator may make the necessary adjustments to reach the final desired power level.

Chapter 5

Conclusions

5.1 Review

Using the point kinetics equations introduced in chapter 2 and the temperature vs. power equation (3.5), we were able to create an adequate model describing the ACRR. Similar real world inputs were used to drive the ACRR model and the resulting steady-state power levels were able to be reproduced. The inverse kinetics method was then used to calculate the reactivity over time for a re-created power run and resulted in an almost exact calculation of the original simulation input.

The inverse kinetics method was then applied to determine the required reactivity levels over time for a desired power trajectory. This desired reactivity trajectory was checked against the current reactivity levels within the ACRR model that were again calculated by the inverse kinetics method. The error in the reactivity levels was feed to a PID compensator which output the required control rod adjustments to meet the desired reactivity levels.

The PID compensator created to automatically control the ACRR model did an

acceptable job reaching the desired power levels. Due to the time delay, control rod adjustments were only able to be made every half second. The cumulative error between the desired and actual reactivity levels did have a negative impact on the final desired steady-state power level but in both cases that were reviewed the difference in power level was negligible.

Some limitations do exist in using this proposed controller. One is the assumption is that the reactor operator can successfully reach the initial predetermined steady-state power level. This often is not the case with the real ACRR. The initial power level is often within a range of five to fifteen percent of the peak reactor power. If the initial power level were off to start with, this could cause the controller to over or undershoot the final desired steady-state power level due to the continued build up of the power level error, since we are only tracking the effective reactivity levels.

Another limitation is the assumption that the control rods will follow a cubic spline trajectory when traveling from the current control rod position to the final control rod position over the half second delay. The control rods may in fact follow a linear or even a higher order polynomial trajectory over the delay, which could alter the performance of the PID compensator to regulate the effective reactivity level.

5.2 Future Work

Of course, the next step will be to actually apply the PID compensator and the inverse kinetics method to the automatic control of the actual ACRR. Some of the challenges will include implementing the kinetics method and ensuring that its calculation can be made in a sufficient amount of time so as not to increase the time delay within the plant. Another challenge will be ensuring that the compensator correctly adjusts the control rods and ensuring that the velocity of the control rods remains within its physical limits.

Another interesting application could include direct power feedback of the power levels over the time of the plant delay to automatically control the ACRR. The theory developed by Calafiore, Dabbene, and Tempo to use randomized algorithms to calculate the proper gain values for desired control [9] is a possible avenue of study.

Appendices

A Derivation of Numerical Integration Methods **4**

B Matlab Script and Function Files **5**

Appendix A

Derivation of Numerical Integration Methods

A.1 Trapezoidal

We want the integral of the function $e(t)$ from $t = 0$ to t as given by

$$I = \int_0^t e(t)dt$$

using only the samples $e_0, e_1, \ldots, e_{k-1}, e_k$. We assume that the integral from $t = 0$ to $t = t_{k-1}$ is known, and is represented as u_{k-1}. Thus we are trying to find a procedure to calculate the "next step." Here we will use trapezoidal integration, where we approximate the integral by computing the area A of the trapezoid in Figure A.1. Therefore,

$$A = \frac{t_k - t_{k-1}}{2}(e_k + e_{k+1}) \tag{A.1}$$

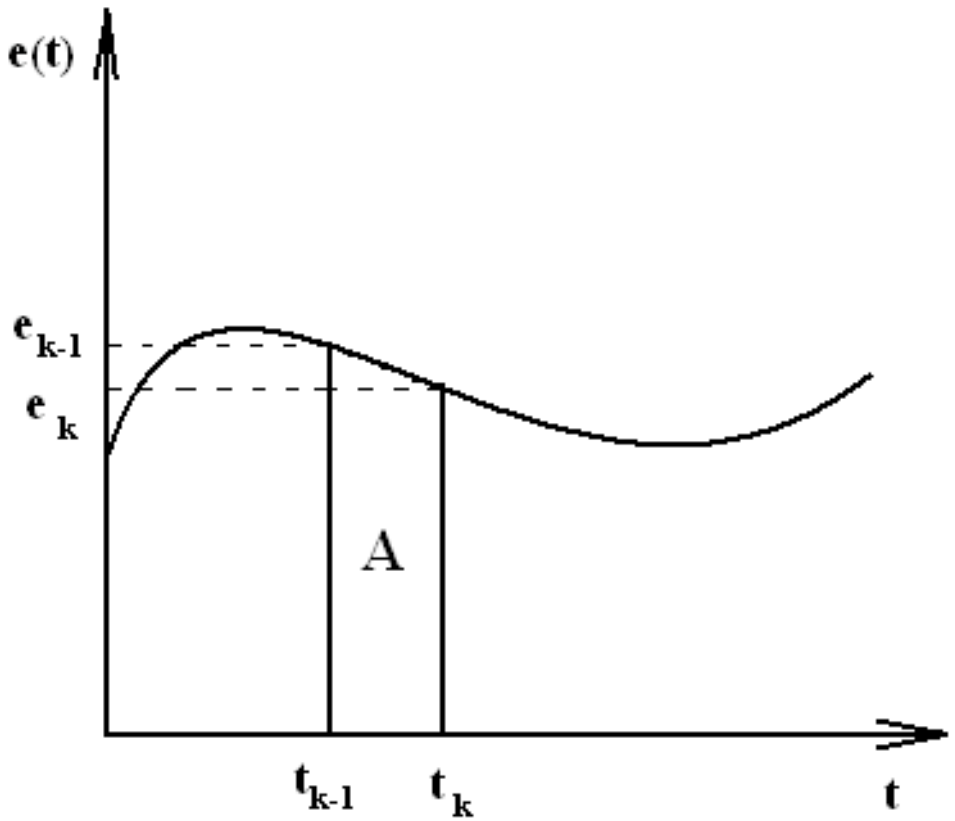

Figure A.1: Trapezoidal Integration.

By assuming a constant step-size, where $t_k - t_{k-1} = T$, we arrive at the linear difference equation for trapezoidal integration as [5]:

$$u_k = u_{k-1} + \frac{T}{2}(e_k + e_{k+1}) \tag{A.2}$$

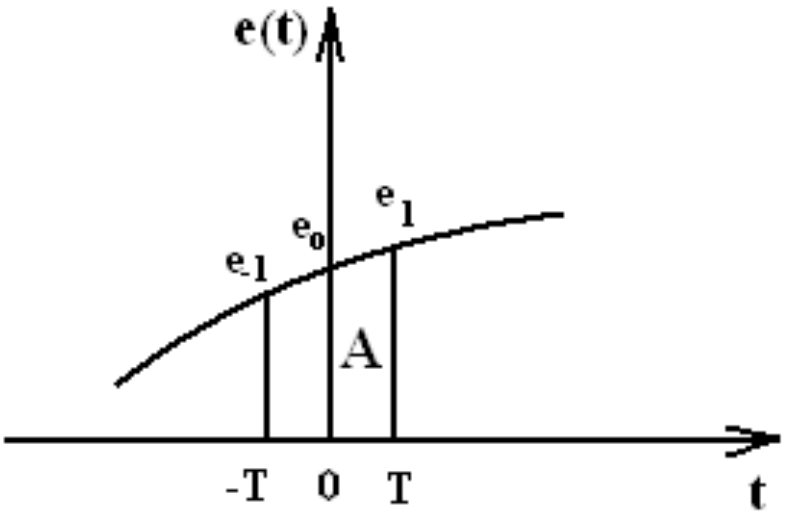

Figure A.2: Parabolic Integration.

A.2 Parabolic

As in the trapezoidal case we will assume we know the value of the integral from $t = 0$ to $t = k - 1$ and is represented as u_{k-1}. Therefore, as before we are looking for a procedure to calculate the "next step." In this case, we first assume that the three points we need for this method are centered about $t = 0$ and then we determine the more generalized case. Using the three samples shown in Figure A.2, along with the parabola,

$$\hat{e} = b_0 + b_1 t + b_2 t^2 \tag{A.3}$$

we fit the parabola by requiring

$$\hat{e}(-T) = b_0 - b_1 T + b_2 T^2 = e_{-1}, \tag{A.4}$$

$$\hat{e}(0) = b_0 = e_0, \tag{A.5}$$

$$\hat{e}(T) = b_0 + b_1 T + b_2 T^2 = e_1. \tag{A.6}$$

Solving equations (A.4)-(A.6) we get

$$b_0 = e_0, \qquad b_1 = \frac{e_1 - e_{-1}}{2T}, \qquad b_2 = \frac{e_{-1} - 2e_0 + e_1}{2T^2}. \tag{A.7}$$

Next we integrate the parabola given in equation (A.3) to find the area A:

$$A = \int_0^T (b_0 + b_1 t + b_2 t^2) dt = b_0 T + \frac{b_1 T^2}{2} + \frac{b_2 T^3}{3}, \tag{A.8}$$

by substituting the expressions from equations (A.7) into equation (A.8), we arrive at the expression for the area A as:

$$A = \left[\frac{5e_1 + 8e_0 - e_{-1}}{12}\right] T. \tag{A.9}$$

We can now generalize from samples -1, 0, 1 to samples $k-2, k-1$, and k, respectively, we arrive at the final form of the difference equation for parabolic integration as [5]:

$$u_k = u_{k-1} + \frac{T}{12}(5e_k + 8e_{k-1} - ek - 2) \tag{A.10}$$

Appendix B

Matlab Script and Function Files

reactor1a.m

```
%reactor1a.m
%
%1st attempt at a reactor model.
%This is the script that will calculate the power output of ACRR.
%This script does not encorperate the use of temperature feedback
%to dynamically describe the ACRR.  This script only simulates the
%time-dependence of the ACRR power level(hopefully) :D
%
%-- B.R.G ECE599 --


tic
RHO = 0.0;              %initial reactivity
DOTRHO = 2.5e-4;        %reactivity/sec
fin = 0.25;             %final amount of reactivity desired
```

```
Po = 3000               %initial power

%determines initial CI's and ACRR variables from initial steady-state
%power and effective reactivity levels
[LAMI,CI,CAPLAM,BETA,BI] = Inkin(Po,RHO);

tf = 100                %final time
dt = 0.001              %time step
t = 0.0:dt:tf;
t = t';

N = tf/dt               %total # of iteration steps

PWR = zeros(N+1,1);     %vector of power histories
PWR(1,1) = Po;

rho = zeros(N+1,1);     %vector of reactivity histories
rho(1,1) = RHO;


RHO = RHO + DOTRHO;

for i=1:N

    %solves the new CI values and the new ACRR power level for the
    %next time step
    rho(i+1,1) = RHO;
    [Po,CI] = kinetics(Po,CI,LAMI,CAPLAM,BETA,RHO,BI,dt);
```

```
    PWR(i+1,1) = Po;
    pwr2(q,1) = Po;

    %determines if the desired total amount of reactivity has been
    %entered into the system
    if RHO < fin
        RHO = RHO + DOTRHO;
    else
        RHO = RHO;
    end

end

%displays the final power level
Po

%calculates and displays the reactor period
tau = dt/(log(PWR(N+1,1))-log(PWR(N,1)))
toc
```

Inkin.m

```
%function [LAMI,CI,CAPLAM,BETA,BI] = Inkin(Po,RHO) calculates the
%initial values for the effective precursor concentrations for the
%six groups from an initial steady-state power level and reactivity
%level. This function also returns the LAMI's, BI's, BETA, and CAPLAM
%values for the ACRR.  The inputs are:
%Po - Initial Power level (Watts)
%RHO - Initial amount of effective reactivity ($)
%
%The outputs are:
%CI's - effective precursor concentration for the six groups
%LAMI's - effective decay constatns for the six groups (1/sec.)
%CAPLAM - mean neutron generation time (sec.)
%BETA - total effective delayed neutron fraction
%RHO - amount of effective reactivity ($)
%BI's - effective delayed neutron fraction of the six groups
%
%-- BRG ECE599 --


function [LAMI,CI,CAPLAM,BETA,BI] = Inkin(Po,RHO)

CAPLAM = 24e-6;             %mean neutron lifetime for ACRR
GEFF = 0.0073/0.007;

%CAPLAM = CAPLAM*GEFF;
BETA = 0.0073;
```

```
CI = zeros(1,6);

%six group beta's for ACRR
BI = [2.66e-4 1.492e-3 1.317e-3 2.851e-3 8.97e-4 1.82e-4];

%six group lamda for ACRR
LAMI = [1.27e-2 3.17e-2 1.15e-1 3.11e-1 1.40 3.87];

%BETA = BETA*GEFF;
BI = BI*GEFF;
%LAMI = LAMI*GEFF;

%initial conditions
RO = RHO*BETA;              %effective reactivity
CAPLAM1 = CAPLAM*(1-RO);    %mean neutron generation time

for j=1:6
    CI(1,j) = (Po*BI(j))/(CAPLAM1*LAMI(1,j));
end
```

kinetics.m

```
%function [Po,CI] = kinetics(Po,CI,LAMI,CAPLAM,BETA,RHO,BI,dt) solves
%the point reactor kinetics equations for six precursor groups by
%using Eulers method.  The inputs are:
%Po - Power (Watts)
%CI's - effective precursor concentration for the six groups
%LAMI's - effective decay constatns for the six groups (1/sec.)
%CAPLAM - mean neutron generation time (sec.)
%BETA - total effective delayed neutron fraction
%RHO - amount of effective reactivity ($)
%BI's - effective delayed neutron fraction of the six groups
%dt - time step between calculations (sec.)
%
%The outputs are:
%Po - New Power level (Watts)
%CI's - New effective precursor concentration for the six groups
%
%-- B.R.G. ECE599 --


function [Po,CI] = kinetics(Po,CI,LAMI,CAPLAM,BETA,RHO,BI,dt)

RO = RHO*BETA;              %effective amount of reactivity
current = Po;               %current power level (Watts)
CAPLAM1 = CAPLAM*(1-RO);    %mean neutron generation time (sec.)

%sum up the contributions from the delayed neutrons
```

```
S = sum(LAMI.*CI);

%calculate change in power
DP = (((RO - BETA)/CAPLAM1)*current + S)*dt;

%calculate new power level
Po = current + DP;

%calculate new CI's for the new power level
for i=1:6
    DCI(1,i) = ((BI(1,i)/CAPLAM1)*current - LAMI(1,i)*CI(1,i))*dt;
    CI(1,i) = CI(1,i) + DCI(1,i);
end
```

reactor1b.m

```
%reactor1b.m
%
%1st attempt at reactor model.
%This is the script that will simulate the dynamic response of the
%ACRR by solving the point reactor kinetics equations and the use
%of a temperature vs. power feedback equation.  This script was
%created to re-create actual real world ACRR power runs by giving
%the model similar real world inputs. (hopefully) :D
%
%-- B.R.G. ECE599 --

tic
RHO = 0.0;              %initial amount of effective reactivity
RHOF = 0.0;

%load power history for an actual ACRR power run
Pd = load('8512_Power_short.txt');
Po = Pd(1,1)

%determines initial CI's and ACRR variables from initial steady-state
%power and effective reactivity levels
[LAMI,CI,CAPLAM,BETA,BI] = Inkin(Po,RHO);

tf = 600                %final time
dt = 0.001              %time step
t = 0.0:dt:tf;
```

```
t = t';
Ts = 6;                 %time step inbetween logged data points

N = tf/dt               %total # of iteration steps
h = tf/Ts;

%load actual control rod path for re-creation of ACRR power run
y = load('CR_position_8512_short.txt');
ys = length(y);
tt = 0:Ts:(ys*Ts)-Ts;
yy = spline(tt,y,t);
yo = yy(1,1);           %initial position of the control rods


PWR = zeros(N+1,1);     %vector of power histories
PWR(1,1) = Po;

rho = zeros(N+1,1);     %vector of effective reactivity histories
rho(1,1) = RHO;
rhoadd = zeros(N+1,1);  %vector of reactivity added by control rods
rhof = zeros(N+1,1);    %vector of feedback reactivity due to temp.
TEMP = zeros(N+1,1);    %vector of core temperature history


for i=1:N
    %calculate the amount of reactivity added to the system in $ by
    %the position of the control rods
    RHOADD = (yy(i,1) - yo)*0.003;
```

```
    rhoadd(i+1,1) = RHOADD;

    %calculate the amount of effective reactivity
    RHO = RHOADD + RHOF;
    rho(i+1,1) = RHO;

    %solves the new CI values and the new ACRR power level for the
    %next time step
    [Po,CI] = kinetics(Po,CI,LAMI,CAPLAM,BETA,RHO,BI,dt);
    PWR(i+1,1) = Po;

    %calculates the amount of feedback reactivity and temperature
    %within the ACRR due to the current power level
    [RHOF,T] = feedback(Po);
    rhof(i+1,1) = RHOF;
    TEMP(i+1,1) = T;

end

%displays the final power level
Po

%calculates and displays the instantaneous reactor period
tau = dt/(log(PWR(N+1,1))-log(PWR(N,1)))
toc
```

feedback.m

```
%function [RHOF,T] = feedback(Po) calculates the amount of feedback
%reactivity within the ACRR due to temperature from a given power
%level.  The input is:
%Po - Power level (Watts)
%
%The outputs are:
%RHOF - feedback reactivity ($)
%T - core temperature (deg C)
%
%-- B.R.G ECE599 --


function [RHOF,T] = feedback(Po)

Po = Po/(1e6);                  %normalize power level to a 1MW scale

% Average from ACRR model
%T = 2.8108*Po^3 - 45.535*Po^2 + 369.23*Po + 36.345;

% Values given from measured data
%T = 29.643*Po^3 - 167.11*Po^2 + 553.19*Po + 35.311;

% Max values from ACRR model
%T = 80.201*Po^3 - 348.84*Po^2 + 800.8*Po + 27.049;

% Average from ACRR model (MATLAB fit)
```

```
%T = 21.7*Po^5 - 87.2*Po^4 + 115*Po^3 - 83.3*Po^2 + 377*Po + 520;
T = 2.64*Po^5 - 30.7*Po^4 + 138*Po^3 - 317*Po^2 + 577*Po + 21.2;
%T = 0.752*Po^5 - 12.5*Po^4 + 78.1*Po^3 - 239*Po^2 + 545*Po + 22.0;

%we assume at 20 deg C there is no reactivity feedback
RHOF = -0.006*T + 0.12;
```

spline4.m

```
%function [time,traj_p]=spline4(tf) calculates a desired cubic power
%trajactory for the ACRR.
%The input is:
%tf - final time (sec.)
%
%The outputs are:
%time - time vector
%traj_p - power trajectory
%
%-- BRG ECE 599 --

function [time,traj_p] = spline4(tf)

dt = 0.002;
ti = 30;

h = ti/dt;

t=[0.0:dt:tf]';
w = length(t);

rf = 1920000;       %final power
ri = 1000;          %initial power

a0 = ri;
a1 = 0;
```

```
a2 = (3/(tf^2))*(rf-ri);
a3 = -(2/(tf^3))*(rf-ri);

traj_2 = a0*t.^0 + a1*t + a2*t.^2 + a3*t.^3;
traj_3 = ones(w-1,1)*traj_2(w,1);
traj_1 = ones(h,1)*traj_2(1,1);

traj_p = [traj_1; traj_2; traj_3];

time = [0:dt:2*tf+ti]';
```

spline5.m

```
%function [time,traj_cr]=spline5(tf) calculates a desired cubic
%control rod trajectory for the ACRR.
%The input is:
%tf - final time (sec.)
%
%The outputs are:
%time - time vector
%traj_cr - trajectory of the control rods
%
%-- BRG ECE 599 --


function [time,traj_cr] = spline5(tf)

%dt = 0.001;
dt = 0.5;
ti = 30;

h = ti/dt;

t=[0.0:dt:tf]';
w = length(t);

rf = 2620;          %final control rod position (rod units)
                                                    %(80% = 2620)
                                                    %(40% = 2260)
```

```
                                                        %(50% = 2320)
                                                        %(60% = 2400)
ri = 1550;          %initial control rod position (rod units) good
%for 1 kW

a0 = ri;
a1 = 0;
a2 = (3/(tf^2))*(rf-ri);
a3 = -(2/(tf^3))*(rf-ri);

traj_2 = a0*t.^0 + a1*t + a2*t.^2 + a3*t.^3;
traj_3 = ones(w-1,1)*traj_2(w,1);
traj_1 = ones(h,1)*traj_2(1,1);

traj_cr = [traj_1; traj_2; traj_3];

time = [0:dt:2*tf+ti]';
```

inverse2.m

```
%inverse2.m
%
%This script calculates the amount of effective reactivity in the
%reactor over time from the loaded power history or desired
%power trajectory by solving equation 6-52 in Duderstadt and
%Hamilton (hopefully):D.
%
%-- BRG ECE 599 --


tic
dt = 0.002;             %time step
BETA = 0.0073;          %total delayed neutron fraction for the ACRR
CAPLAM = 24e-6;         %mean neutron generation time for the ACRR

M = 240/dt;             %total # of integration steps

%load power history or power trajectory
PWR = load('test8_Power.txt');
ps = length(PWR)

pwr = ones(M,1);
pwr = pwr*PWR(1,1);

pwr1 = [pwr; PWR];
```

```
l = length(pwr1);

pwr2 = flipud(pwr1);

%vector created to store effective reactivity levels to create a
%desired effective reactivity trajectory
inv_rho = zeros(ps,1);

%load delayed neutron kernel
D = load('delayKernel2.txt');

K = speye(M,length(D));
D = K*D;
d = length(D)

for i=0:ps-1

    %variable used for keeping track of the power vector
    q = ps-i;
    v = q+M-1;

    %power vector to be used in calculating the amount of effective
    %reactivity
    C = pwr2(q:v);

    mult = D.*C;

    %calculation of the integral portion
```

```
    intg = sum(mult);
    intg = intg*dt;

    %calculation of the derivative portion
    der = 1/C(1,1);

    %calculation of seperate numerator and denomenator portions
    num = BETA*C(1,1) + der*CAPLAM - BETA*intg;
    den = der*CAPLAM + C(1,1);

    %calculation of the amount of effective reactivity ($)
    inv_rho(i+1,1) = (num/den)*(1/BETA);
end
toc;
```

reactor6.m

```
%reactor6.m
%
%The reactor6.m script will simulate the operation of the inverse
%kinetics method and a PID compensator to automatically control
%the ACRR.  In this simulation the point reactor kinetics equations
%are expressed and solved in a state-space representation.  This
%simulation also models the pure time delay within the ACRR system.
%
%-- BRG ECE 599 -- 10/12/07 --

tic

RHO = 0.0;                  %initial amount of effective reactivity

%PID gains
Kp = 0.5
Kd = 0.35
Ki = 3.7

%Load Desired Power Trajectory
Pd = load('test7_Power.txt');
Po = Pd(1,1);                               %initial power level

%determines initial CI's and ACRR variables from initial steady-state
%power and effective reactivity levels
[LAMI,CI,CAPLAM,BETA,BI] = Inkin(Po,RHO);
```

```
%initial values for state variables
xo = [CI'; Po];

tf = 680;                    %final time
dt = 0.002                   %time step of simulation
z = 0.5;                     %delay of plant
n = z/dt;                    %number of plant delay itterations

%time vectors for plotting simulation results
t = 0.0:dt:tf;
t2 = 0.0:z:tf;
t3 = 0.5:z:tf;
t = t';
t2 = t2';


N = tf/dt                    %total # of iteration steps

Q = tf/z;
M = 240/dt;

%column vectors to hold variables values throughout simulation
rho = zeros(N+1,1);          %effective reactivity
rho(1,1) = RHO;

rhoadd = zeros(N+1,1);       %reactivity added by control rods
```

```
rhof = zeros(N+1,1);        %feedback reactivity due to temperature
TEMP = zeros(N+1,1);        %core temperature (deg C)
cra = zeros(Q+1,1);         %control rod adjustments (rod units)
inv_rho = zeros(N+1,1);     %effective reactivity calculated by
%inverse method

rhoe = zeros(Q+2,1);        %reactivity error between desired and
%actual

rhoa = zeros(Q+1,1);        %reactivity adjustments calculated from
%cra

r = ones(N+1,1);            %multiplication factor

%state space matrices -- xo = d*xo + h*xo*u
d = [1-LAMI(1,1)*dt 0 0 0 0 0 0;
     0 1-LAMI(1,2)*dt 0 0 0 0 0;
     0 0 1-LAMI(1,3)*dt 0 0 0 0;
     0 0 0 1-LAMI(1,4)*dt 0 0 0;
     0 0 0 0 1-LAMI(1,5)*dt 0 0;
     0 0 0 0 0 1-LAMI(1,6)*dt 0;
     LAMI(1,1)*dt LAMI(1,2)*dt LAMI(1,3)*dt...
     LAMI(1,4)*dt LAMI(1,5)*dt LAMI(1,6)*dt 1-dt/CAPLAM];

h = [0 0 0 0 0 0 BI(1,1)*dt/CAPLAM;
     0 0 0 0 0 0 BI(1,2)*dt/CAPLAM;
     0 0 0 0 0 0 BI(1,3)*dt/CAPLAM;
     0 0 0 0 0 0 BI(1,4)*dt/CAPLAM;
```

```
     0 0 0 0 0 0 BI(1,5)*dt/CAPLAM;
     0 0 0 0 0 0 BI(1,6)*dt/CAPLAM;
     0 0 0 0 0 0 (1-BETA)*dt/CAPLAM];

%create output matrix -- PWR = C*xo
C1 = [zeros(1,6) 1];



PWR = zeros(N+1,1);     %vector of power histories
PWR(1,1) = Po;          %initial power level
ps = length(PWR);

%create extended power history for use of the inverse kinetics method
pwr = ones(M,1);
pwr = pwr*PWR(1,1);

pwr1 = [pwr; PWR];
pwr2 = flipud(pwr1);

%load delay kernel value
D = load('delayKernel2.txt');
F = D;
GG = speye(M,length(D));
D = GG*D;

%load predetermined control rod trajectory
yyy = load('test7_crd.txt');
yo = yyy(1,1)                   %initial position of the control rods
```

```
y = yyy;                          %initial trajectory of control rods



k = 0;
aa = [0 z]';
ttt = (0:dt:z)';
int = 0.0;
RHOF = 0.0;

%load desired effective reactivity trajectory
rho_d = load('test7_rho.txt');

for i = 1:Q

    %determine control rod trajectory over plant delay period
    yyy(i+1,1) = yyy(i+1,1) + sum(cra);
    bb = [yyy(i,1) yyy(i+1,1)]';

    %assumes piece-wise cubic polynomial trajectory for control rods
    yy = spline(aa,bb,ttt);

    %this for loop simulates the ACRR progres over the plant delay
    for j = 1:n
        k = k+1;

        %variable used for keeping track of the proper power vector
        %for use in the inverse method
        q = ps-k;
```

```
v = q+M-1;

%calculate the amount of reactivity added to the system in $
%by the position of the control rods
RHOADD = (yy(j,1)-yo)*0.003;
rhoadd(k+1,1) = RHOADD;

%calculated the amount of effective reactivity present within
%the system ($)
RHO = RHOADD + RHOF;
rho(k+1,1) = RHO;

%converts effective reactivity in dollars to effective
%reactivity in a decimal representation
RO = RHO*BETA;

%calculates the value of the effective multiplication factor
r(k+1,1) = 1/(1-RO);

%solves the state-space equations and calculates the new CI
%values and new ACRR Power level
xo = d*xo + h*xo*r(k+1,1);

%power output
PWR(k+1,1) = C1*xo;
pwr2(q,1) = PWR(k+1,1);

%calculates the amount of feedback reactivity due to the
```

```
    %temperature within the ACRR
    [RHOF,T] = feedback(PWR(k+1,1));
    rhof(k+1,1) = RHOF;
    TEMP(k+1,1) = T;

end

%calculate the actual amount of effective reactivity within the
%ACRR by the use of the inverse method
C = pwr2(q:v);
INV_RHO = invequ(C,D,dt);
inv_rho(k+1,1) = INV_RHO;

%finds the error between the desired and actual amounts of
%effective reactivity
RHOE = rho_d(k+1,1) - INV_RHO;
rhoe(i+2,1) = RHOE;

%derivative of the error signal miltiplied by the gain value
DER = Kd*(rhoe(i+2,1)-rhoe(i+1,1))/z;

%%INTEGRATION METHODS%%
%backwards integration
%int = sum(rhoe)*z;
%trapizoidal integration
int = int + (rhoe(i+2,1)+rhoe(i+1,1))*z/2;
%parabolic integration
%int = int + (5*rhoe(i+2,1)+8*rhoe(i+1,1)-rhoe(i,1))*z/12;
```

```
    %multiplies gain values
    INT = Ki*int;
    PRO = Kp*RHOE;

    %calculates the needed amount of reactivity compensation
    RHOA = DER+INT+PRO;
    rhoa(i+1,1) = RHOA;

    %calculates the needed amount of control rod compensation
    CRA = (RHOA/0.003);
    cra(i+1,1) = CRA;

end

%calculates the velocity of the control rods over entire simulation
rod_vel_6 = RodV(yyy);
mrv = max(rod_vel_6)            %max positive control rod velocity
lrv = min(rod_vel_6)            %max negative control rod velocity

%calculates the RMS error between the desired and actual effective
%reactivity trajectoies ($)
erms_rho = sqrt(rhoe'*rhoe/length(rhoe))

PWR(k+1,1)                      %final ACRR power level
toc
```

invequ.m

```
%function [inv_rho] = invequ(C,D,dt) calculates the effective
%reactivity in the reactor at a single point in time from the power
%histories by using equation 6-52 in Duderstadt and Hamilton
%(hopefully):D.  The inputs are:
%C - vecotr containing the Reactor power histories (Watts)
%D - vector containing the values of the delayed neutron kernel
%dt - time step between Reactor power values (sec.)
%
%The output is:
%inv_rho - effective reactivity level ($)
%
%-- BRG ECE599 --

function [inv_rho] = invequ(C,D,dt)


BETA = 0.0073;          %total effective delayed neutron fraction
CAPLAM = 24e-6;         %mean neutron generation time

%calculates the product of the delayed neutron kernel with the power
%history
mult = D.*C;

%calculates the integral portion
intg = sum(mult);
intg = intg*dt;
```

```
%calculates the derivative portion
der = 1/C(1,1);

%calculation of seperate numerator and denomenator portions of the
%equation
num = BETA*C(1,1) + der*CAPLAM - BETA*intg;
den = der*CAPLAM + C(1,1);

%calculation of the amount of effective reactivity ($)
inv_rho = (num/den)*(1/BETA);
```

RodV.m

```
%function [rod_vel] = RodV(rod_pos) calculates the velocity of the
%control rods from a vector containing the position of the control
%rods over time.  The input is:
%rod_pos - vector containing control rods positions over time
%
%The output is:
%rod_vel - vector containing the velocity of the control rods over
%time
%
%-- BRG ECE599 --

function [rod_vel] = RodV(rod_pos)

dt = 0.5;                       %time step between values of the
%control rod positions

l = length(rod_pos);
rod_vel = zeros(l-1,1);

for i=1:(l-1)
    rod_vel(i,1) = (rod_pos(i+1,1)-rod_pos(i,1))/dt;
end
```

References

[1] James J. Dahl, Richar L. Coats, and Michael K. Black. Documented saftey analysis (dsa) for the annular core research reactor facility (acrrf), ucni. Technical report, Sandia National Laboratory, 2007.

[2] Ronald A. Knief. *Nuclear Engineering: Theory and Technology of Commercial Nuclear Power*. Hemisphere Publishing Corp, second edition, 1992.

[3] Samuel Glasstone and Alexander Sesonske. *Nuclear Reactor Engineering*. Van Nostrand Reinhold Company, 1967.

[4] James J. Duderstadt and Louis J. Hamilton. *Nuclear Reactor Analysis*. John Wiley and Sons, Inc., 1976.

[5] Gregory P. Starr. Introduction to applied digital control. Lecture Notes in Digital Control, November 2006.

[6] Valery D. Yurkevich. *Design of Nonlinear Control Systems with the Highest Derivative in Feedback*. World Scientific, 2004.

[7] Ronald R. Mohler. *Nonlinear Systems*, volume II Application to Bilinear Control. Prentice Hall, 1991.

[8] John J. Craig. *Introduction to Robotics Mechanics and Control*. Pearson Prentice Hall, third edition, 2005.

[9] Giuseppe Calafiore, Fabrizio Dabbene, and Roberto Tempo. A survey of randomized algorithms for control synthesis and performance verification. *Journal of Compexity*, 23:301–316, June 2007.