INSTITUTO DE ENGENHARIA NUCLEAR

ANDERSON DE MELO SILVA ALVES

DESENVOLVIMENTO DE UM SISTEMA COMPUTACIONAL PARA CÁLCULO DE INVENTÁRIO DO REATOR ARGONAUTA EM TEMPO REAL

Rio de Janeiro
2021

ANDERSON DE MELO SILVA ALVES

DESENVOLVIMENTO DE UM SISTEMA COMPUTACIONAL PARA CÁLCULO DE INVENTÁRIO DO REATOR ARGONAUTA EM TEMPO REAL

Dissertação apresentada ao Programa de Pós Graduação em Ciência e Tecnologia Nucleares do Instituto de Engenharia Nuclear da Comissão Nacional de Energia Nuclear como parte dos requisitos necessários para a obtenção do Grau de Mestre em Ciência e Tecnologia Nucleares - Profissional em Engenharia de Reatores

Orientadores: Prof. Dr. Celso Marcelo Franklin Lapa
Prof. Dr. Adino Américo Heimlich Almeida

Rio de Janeiro
2021

FICHA CATALOGRÁFICA

ALVE Silva Alves, Anderson de Melo

Desenvolvimento de um sistema computacional para cálculo de inventário do reator Argonauta em tempo real/ Anderson de Melo Silva Alves – Rio de Janeiro: CNEN/IEN, 2021.

xiii, 89f.: il.; 31cm

Orientadores: Celso Marcelo Franklin Lapa e
Adino Américo Heimlich Almeida

Dissertação (Mestrado em Ciência e Tecnologia Nucleares) - Instituto de Engenharia Nuclear, PPGIEN, 2021

Referências Bibliográficas: p. 60-62.

1. Concentrações isotópicas. 2. Termo Fonte 3. Argonauta 4. Equação de difusão de nêutrons 5. Exponencial da Matriz. 6. Método das diferenças finitas. 7. Raspberry Pi. 8. OPENMC.

DESENVOLVIMENTO DE UM SISTEMA COMPUTACIONAL PARA CÁLCULO DE INVENTÁRIO DO REATOR ARGONAUTA EM TEMPO REAL

Anderson de Melo Silva Alves

DISSERTAÇÃO SUBMETIDA AO PROGRAMA DE PÓS GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA NUCLEARES DO INSTITUTO DE ENGENHARIA NUCLEAR DA COMISSÃO NACIONAL DE ENERGIA NUCLEAR COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS E TECNOLOGIA NUCLEARES.

Aprovada por:

_____

Prof. Celso Marcelo Franklin Lapa, D. Sc.

_____

Prof. Adino Américo Heimlich Almeida, D. Sc.

_____

Prof. Francisco José de Oliveira Ferreira, D. Sc.

_____

Prof. Jonathan Marcello de Oliveira Pinto, D. Sc.

RIO DE JANEIRO, RJ - BRASIL
DEZEMBRO DE 2021

AGRADECIMENTOS

Primeiramente, agradeço a Deus, por todas as bençãos alcançadas até o momento e, mesmo pelas tribulações vividas que, permitem me fortalecer um pouco mais a cada dia.

A minha mãe Sandra pelo amor, dedicação e incentivo durante toda a trajetória de vida, graduação e pós graduação, sempre me motivando para buscar conhecimento, capacitação e a seguir meus objetivos.

Aos professores Dr. Celso Marcelo Franklin Lapa e Dr. Adino Américo Heimlich Almeida, primeiramente por terem aceitados me orientar no presente trabalho, pela paciência e, pelas significativas sugestões realizadas nas modificações desta dissertação.

Aos demais professores do curso de pós graduação em ciência e tecnologia nucleares do Instituto de Engenharia Nuclear (IEN), que compartilharam seus valiosos conhecimentos, me possibilitando assim, capacitar um pouco mais.

Aos amigos que fiz durante o curso de pós graduação no IEN, em especial, aos que me acompanharam durante o ciclo de aulas da área de reatores, Alessandro Mariano, Carla Gomes Tinoco, Jonathan Alves, José Rafael Nicolao, Luiz Carlos e Luiz Sergio Mendonça.

Aos amigos que trago desde a graduação em Física na UERJ, Bruno Ferraz e Vitor Hugo que, também como alunos do IEN, me acompanharam nessa árdua tarefa de mais alguns anos de estudo.

A todos os funcionários do IEN pela cordialidade, demonstração de sempre querer ajudar na resolução de problemas administrativos e pelas informações importantes.

Por fim, agradeço a bolsa de estudos concedida pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) durante o período de mestrado.

RESUMO

O trabalho apresenta a modelagem e simulação dos assemblys homogeneizados do reator Argonauta no OPENMC, a obtenção de seções de choque multigrupo, simulação de perfil de fluxo de nêutrons unidimensional e cálculo de depleção de 20 actinídeos e 31 produtos de fissão que, apresentam significativas seções de choque, alta radioatividade ou que, são de crucial conhecimento, devido a necessidade de armazenamento. De forma complementar, é feito um estudo de viabilidade técnica do uso do Raspberry Pi nas simulações citadas, bem como, é realizado uma descrição do desenvolvimento e implementação de uma interface eletrônica capaz de coletar dados de tensão dos sensores do reator Argonauta. A modelagem dos assemblys no OPENMC, foi feita com base em cálculos de homogeneização de reatores e na documentação oficial do código. A obtenção das seções de choque multigrupo homogeneizadas foi estimada por aritmética de tallys e com base no fluxo de nêutrons estimado via OPENMC. A simulação do perfil de fluxo de nêutrons do reator Argonauta foi baseada em simulação no DIF1D, escrito em FORTRAN, que estima o fluxo de nêutrons para um reator unidimensional. O modelo matemático presente no código DIF1D, baseia-se na discretização espacial da Equação de Difusão de Nêutrons com 2 grupos de energia, utilizando o método das diferenças finitas centrado na malha, bem como, no algoritmo de Thomas e no de potência, para assim, obter o fluxo de nêutrons. O cálculo e simulação da depleção de actinídeos e produtos de fissão foi feito com base em estudos, considerando mecanismos de ganho e perdas de cada elemento no tempo, levando-se em conta uma aproximação quase-estática e, para solucionar o sistema matricial de equações, foi desenvolvido o algoritmo expm, para resolver a matriz exponencial de depleção. A avaliação da viabilidade técnica foi feita através de aferição de tempo de processamento do código DIF1D, em FORTRAN e C++, executando no Raspberry Pi e em desktops tradicionais. A interface eletrônica desenvolvida foi feita com base em adaptações de algoritmo de projeto existente. Os resultados de tempo de simulação de perfil de fluxo de nêutrons, 0.11s, e de depleção, 0.14s, evidenciam a possibilidade do uso do Raspberry Pi em simulações ligados a área nuclear. Além disso, foi possível modelar e obter as seções de choque multigrupo através de simulação no OPENMC, mostrando-se uma alternativa as modelagens com uso de MCNPX. Os resultados simulados no código DIF1D foi satisfatório para um estudo preliminar, tendo em vista que, foram realizadas aproximações durante a simulação. Os resultados de depleção isotópica indicam que, o núcleo do Argonauta apresenta ainda grande quantidade de U^{235} e U^{238}. No entanto, existem elementos de alta atividade, como os isotopos de Iodo e Cs^{138}, necessitando a monitoração destes, para o armazenamento. O trabalho também mostra-se como uma salvaguarda para projetos de extensão de vida e das operações atuais do reator Argonauta.

Palavras-Chave: Concentrações isotópicas, Termo fonte, Argonauta, Equação de Difusão de Nêutrons, Método das diferenças finitas, Exponencial da Matriz, Raspberry Pi, OPENMC.

ABSTRACT

The work presents the modeling and simulation of the homogenized assemblies of the Argonauta reactor in OPENMC, the obtainment of multigroup cross sections, simulation of one-dimensional neutron flux profile, and calculation of depletion of 20 actinides and 31 fission products that present significant cross-section, high radioactivity or which are of crucial knowledge due to the need for storage. Complementarily, a study of the technical feasibility of using the Raspberry Pi in the aforementioned simulations is carried out, as well as a description of the development and implementation of an electronic interface capable of collecting voltage data from the Argonauta reactor sensors. The assembly modeling in OPENMC was done based on reactor homogenization calculations and official code documentation. The obtainment of the homogenized multigroup cross-sections was estimated by tally arithmetic and based on the neutron flux estimated via OPENMC. The simulation of the neutron flux profile of the Argonauta reactor was based on simulation in DIF1D, written in FORTRAN, which estimates the neutron flux for a one-dimensional reactor. The mathematical model present in the DIF1D code is based on the spatial discretization of the Neutron Diffusion Equation with 2 energy groups, using the finite difference method centered on the mesh, as well as on the Thomas algorithm and on the power one, for this, get the neutron flux. The calculation and simulation of the depletion of actinides and fission products were based on studies, considering the gain and loss mechanisms of each element in time, taking into account a quasi-static approximation and, to solve the matrix system of equations, was developed the algorithm expm, to solve exponential depletion matrix. The evaluation of technical feasibility is done by measuring the processing time of the DIF1D code, in FORTRAN and C ++, on Raspberry Pi and on traditional desktops. The electronic interface developed was based on adaptations of an existing design algorithm. The neutron flux profile simulation time results, 0.11s, and depletion, 0.14s, show the possibility of using the Raspberry Pi in simulations related to the nuclear area . Furthermore, it was possible to model and obtain a multigroup shock source through simulation in OPENMC, showing an alternative as modeling using MCNPX. The simulated results in the DIF1D code was satisfactory for a preliminary study, considering that approximations were performed during the simulation. The result of obtaining isotopic depletion is that the Argonauta core still presents large amounts of U^{235} and U^{238}. However, there are elements of high activity, such as Iodine isotopes and Cs^{138}, which need to be monitored for storage. The work also proves to be a safeguard for life extension projects and the current operations of the Argonauta reactor.

Keywords: Isotopic concentrations, Source term, Argonaut, Neutron diffusion equation, Finite difference method, Matrix exponential, Raspberry Pi, OPENMC.

Lista de Figuras

1 Núcleo do Reator Argonauta 16

2 Visão geral do reator Argonauta e seus sistemas de segurança 17

3 Raspberry Pi 4 modelo B 18

4 Discretização da variável energia em dois grupos 23

5 Discretização Espacial do Reator Argonauta 24

6 Cadeia de Actinídeos 29

7 Modos de transmutação de Actinídeos 30

8 Componentes utilizados no projeto 38

9 Implementação do MCP3008 no Raspberry Pi 3B 39

10 Flowchart do programa de simulação simplificada do reator Argonauta unidimensional 49

11 Perfil de Fluxo de Nêutrons Térmicos via DIF1D 55

12 Perfil de Fluxo de Nêutrons Rápido via DIF1D 55

13 Especificação das portas do ADC MCP3008 80

14 Esquema de ligações (conexões) entre o Raspberry Pi 3B e o ADC MCP3008 . 80

15 Esquema de ligações(conexões) entre o Raspbarry Pi 3B e o LCD LCM1602 . 81

Lista de Tabelas

1 Dados de meia-vida de actinídeos e suas relações de captura e decaimento . . . 29

2 Dados de meia-vida de produtos de fissão e suas relações de transmutação entre si, devido a decaimento e captura de nêutrons 33

3 Tempo de simulação (segundos), para simular um reator unidimensional da IAEA com 340 malhas e 9 regiões. 40

4 Parâmetros nucleares da Região 1(cerne ativo) via OPENMC, contendo U_3O_8, água e Alumínio 54

5 Parâmetros nucleares da Região 2(excesso de materiais) via OPENMC, contendo Alumínio e água 54

6 Parâmetros nucleares da Região 3(refletora) via OPENMC, contendo grafite . . 54

7 Parâmetros nucleares da Região 4(externa aos assemblys) via OPENMC, contendo grafite 54

8 Perfil de potência dos 6 elementos combustíveis do reator Argonauta, considerando operação em 500W. 55

9 Inventário de Actinídeos no núcleo do reator Argonauta até 2018 56

10 Inventário de isotopos de Iodo no núcleo do reator Argonauta até 2018 56

11 Inventário de produtos de fissão voláteis no núcleo do reator Argonauta até 2018 56

12 Inventório de produtos de fissão não voláteis no núcleo do reator Argonauta até 2018 57

13 Especificação das portas do Raspberry Pi, para os padrões BCM, Física e wiringPi(wPi) 79

LISTA DE ABREVIATURAS E SIGLAS

ADC	- Analog Digital Converter (Conversor Analógico para Digital)
ANL	- Argonne National Laboratory
API	- Application Programming Interface
ASSEMBLY	- Conjunto de elementos combustível presente no núcleo de um reator
CNEN	- Comissão Nacional de Energia Nuclear
CRON	- Agendador de Serviços dos Sistemas Operacionais Linux
DIF1D	- Conjunto de algoritmos, originalmente escrito em FORTRAN, que simula o perfil de fluxo de nêutrons de reatores unidimensionais baseado no método das diferenças finitas
GPIOs	- General-Purpose Input/Output
IAEA	- International Atomic Energy Agency
IEA	- International Energy Agency
OPENMC	- Código aberto, baseado no método de Monte Carlo, para simulação transporte de partículas e células/assembly de reatores nucleares
SBC	- Single Board Computer
TMI	- Three Mile Island

Sumário

1 INTRODUÇÃO **14**

1.1 Contextualização 14

1.2 Apresentação do Problema a ser Resolvido 15

1.2.1 **O Reator Argonauta** 15

1.2.2 **O Raspberry Pi** 18

1.2.3 **As Linguagens de Programação Escolhidas** 19

1.3 Objetivos do Trabalho 19

1.4 A Estrutura da Dissertação 20

2 FUNDAMENTAÇÃO TEÓRICA **21**

2.1 Um Breve Estudo de Nêutrons em Reatores 21

2.1.1 **A Equação de Difusão de Nêutrons Multigrupo** 22

2.1.2 **A Aproximação Quase-Estática** 23

2.1.3 **A Discretização da Variável Energia** 23

2.1.4 **A Discretização Espacial de Um Reator** 24

2.1.5 **Obtenção do Fluxo de Nêutrons** 25

2.2 A Depleção Isotópica Durante o Ciclo do Combustível 27

2.2.1 **Introdução ao Estudo das Variações das Concentrações Isotópicas no Tempo** 28

2.2.2 **Os Actinídeos Avaliados** 28

2.2.3 **Os Produtos de Fissão Avaliados** 31

2.2.4 **Forma Matricial das Equações de Depleção** 33

2.2.5 **Solução das Equações de Depleção com Método da Exponencial da Matriz** 34

2.3 Cálculo de Atividade de Radionuclídeos 35

3 METODOLOGIA **37**

3.1 Estudo da Viabilidade Técnica da Simulação das Concentrações Isotópicas do Reator Argonauta no Raspberry Pi 37

3.1.1 **Componentes Utilizados no Projeto** 37

3.1.2 **A Solução do Problema da Leitura de Medidas Analógicas Pelo Raspberry Pi** 38

3.1.3 **Benchmark para Aferir o Tempo de Processamento** 39

3.2 Modelagem do Assembly do reator Argonauta no OPENMC 40

3.2.1 **Descrição do OPENMC** 40

3.2.2 **A Modelagem da Célula do Argonauta no OPENMC** 41

4 DESCRIÇÃO DOS MÓDULOS DO SISTEMA **49**

4.1 O código lerSensores.c 50

4.2 O Código DIF1D 50

4.2.1 O Algoritmo LeituraDadosEntrada.f90 50

4.2.2 O Algoritmo Inicialização.f90 51

4.2.3 O Algoritmo CalculaMatrizesDiscretizacao.f90 51

4.2.4 O Algoritmo IteracoesExternas.f90 51

4.2.5 O Algoritmo CalculaPotencia.f90 51

4.2.6 O Algoritmo SendDataToHistoricoPotencia.py 52

4.2.7 O Algoritmo Decay.f90 52

4.2.8 O Algoritmo SendDataToConcentracoesIsotipicas.py 52

5 RESULTADOS **53**

5.1 Da viabilidade técnica 53

5.2 K_{eff} e parâmetros nucleares via OPENMC 53

5.3 K_{eff} e potência via DIF1D 54

5.4 Inventário isotópico e dados de atividade 56

6 CONCLUSÕES E PERSPECTIVAS FUTURAS **58**

6.1 Considerações gerais 58

6.2 Considerações finais 59

6.3 Sugestões de trabalhos futuros 59

REFERÊNCIAS **60**

APÊNDICES **63**

Apêndice A - Cálculos 63

Apêndice A.1 - Derivação da Equação de Difusão de Nêutrons Multigrupo . . . 63

Apêndice A.2 - Equações de Depleção de Actinídeos Expandidas 66

Apêndice A.3 - Cálculo da Célula do Argonauta 70

Apêndice B - Códigos, Especificações e Esquemas de Ligação 79

Apêndice B.1 - Especificação e Conexões do Raspberry Pi 3B, ADC MCP3008 e LCD LCM1602 79

Apêndice B.2 - Código em C, Para Leitura de Sensor Analógico com MCP3008 81

Apêndice B.3 - Código em Python, Para Envio de Dados de Potencia do Reator Para o Banco de Dados 86

Apêndice B.4 - Código em Python, Para Envio de Dados de Concentrações Isotópicas Para o Banco de Dados 88

1 INTRODUÇÃO

Este capítulo destina-se a fazer uma breve contextualização do problema a ser tratado neste trabalho, bem como, apresenta as motivações e objetivos que foram alcançados ao final do projeto. Ainda neste, é feito um resumo de como está estruturado os demais capítulos.

1.1 Contextualização

Após 70 anos da construção do primeiro reator nuclear, com a finalidade de produzir energia elétrica para o consumo, observa-se em um panorama histórico que, não houve um número significativo de acidentes nucleares de grande relevância, quando comparado a outros desastres naturais (ex.:Tsunami, Terremoto). Os três acidentes de maior destaque foram, o de Three Mile Island (TMI), em 1979, o de Chernobyl [1], em 1986 e o de Fukushima, em 2011, todos estes causando maiores impactos no aspecto psicológico do que ambiental [2]. No entanto, o medo social induzido [3] e, o aprendizado com os acidentes, motivaram uma grande evolução nos projetos de reatores nucleares, adotando-se sistemas de segurança passivos, bem como, tendo uma maior atenção ao inventário isotópico de elementos dentro do núcleo de reatores, de forma a prevenir e mitigar acidentes de grande proporções. Por exemplo, o conceito de um acidente de sítio no reator Argonauta, foi abordado em dois artigos [4] e [5] que, fazem estudo de caso, avaliando as concentrações isotópicas e o termo fonte, considerando um acidente hipotético no reator Argonauta, devido a queda da cobertura do reator, devido a falha do guindaste que, rotineiramente manipula a estrutura (cobertura), assim, promovendo um acidente, com exposição do núcleo e, liberação de radionuclídeos no prédio de contenção do reator. O artigo [4] em questão, foi a motivação inicial para a proposição deste projeto.

Por outro lado, instituições internacionais, tais como, a International Energy Agency (IEA), evidenciam a crescente demanda de energia [6]-[7] e, sugere que várias usinas nucleares sejam construídas até 2050, a fim de substituir as usinas que serão descomissionadas nesse período, bem como, em uma tentativa de frear (em parte) o aquecimento global, devido as emissões de gases poluentes(ex.:CO_2). Nesse aspecto, estudos de caso sobre a segurança operacional de reatores, atualização dos sistemas atuais [8] e das concentrações isotópicas [4],[9]-[11] são de crucial importância e, também já justificam (em parte) o presente trabalho.

1.2 Apresentação do Problema a ser Resolvido

Devido a necessidade de avaliar o inventário de reatores nucleares por motivo de segurança operacional, este projeto foi proposto. De forma sucinta, este trabalho consistiu em produzir um modelo simplificado dos *assemblys* (conjunto de elementos combustíveis) do reator Argonauta e, simular estes, para obter as seções de choque multigrupo que, foram utilizadas como parâmetros de entrada no código DIF1D (simula perfil de fluxo de nêutrons em reatores unidimensionais) e, assim, em ultima instancia, avaliar o inventário e o termo fonte do reator Argonauta em tempo real.

Por outro lado, este trabalho também foi motivado para obter um dispositivo capaz de capturar dados de tensão/corrente dos sensores do reator Argonauta, a fim dessas informações coletadas serem utilizados para estimar a potencia em cada assembly presente no reator.

As próximas seções apresentam com um pouco mais de detalhes as motivações do problema a ser solucionado, descrevendo uma justificativa para as linguagens de programação escolhidas e, para o dispositivo utilizado. Por fim, destaca os objetivos que foram alcançados no final do trabalho.

1.2.1 O Reator Argonauta

O Argonauta (IEN-R1) é um reator de pesquisa, moderado a água, situado no campus da Universidade Federal do Rio de Janeiro e, que foi projetado pelo Argonne National Laboratory (ANL, mesmo laboratório que projetou o primeiro reator nuclear). Ele é o único reator de pesquisa na cidade do Rio de Janeiro. O mesmo está em operação desde 1965 no Instituto de Engenharia Nuclear (IEN) e, fica a disposição para colaboração de outras instituições de ensino superior, tais como COPPE, IME e IRD. O projeto do reator contempla um fluxo de nêutrons térmicos de até 10^{10} nêutrons $cm^{-2} \cdot s^{-1}$ em 5kW de potência [12].

O formato do núcleo é composto por 2 cilindros de alumínio concêntricos, tendo raios interno de 304,8 mm e raio externo 457,2mm e, possuindo uma altura ativa de 60,2 cm [13]. O cilindro mais interno é completamente preenchido por grafite, que atua como refletor. Enquanto os arranjos de combustíveis são posicionados entre os cilindros, conforme pode ser visto na figura 1. Cada arranjo de combustível é composto por 17 lâminas de Alumínio com miolo de Alumínio e U_3O_8 enriquecido a 19.99%. Inicialmente existiam 6 arranjos de elemento combustível, totalizando uma massa de 10.5 Kg de U_3O_8. Separando os arranjos de combustível, existem cunhas de grafite, que atuam como refletores de nêutrons (com a finalidade de evitar a fuga de nêutrons em demasia), conforme pode ser visto na figura 1.

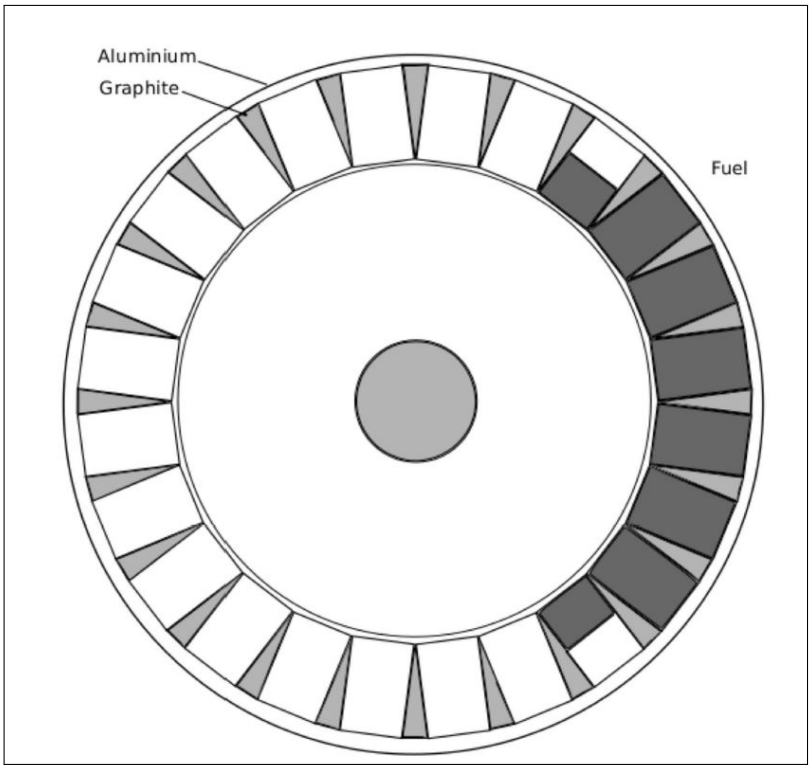

Figura 1: Núcleo do Reator Argonauta, fonte: [4]

Desde a sua primeira criticalidade, em 19 de fevereiro de 1965 até os dias atuais, não houve um levantamento do inventário isotópico e da queima contemplando vários actinídeos e produtos de fissão. Nesse sentido, os cadernos dos registros das operações do reator Argonauta, contendo dados como data, potência e tempo de operação, são utilizados atualmente para realizar cálculos de depleção de alguns poucos isótopos.

Até o momento, existe somente a avaliação das concentrações de U^{235}, U^{238} e Pu^{239} para atender as exigências da Comissão Nacional de Energia Nuclear (CNEN) e, para atender ao acordo internacional de ceder informações sobre dados das concentrações isotópicas dos reatores e, ao tratado internacional de não proliferação de armas nucleares. Essa estimativa da concentração de apenas 3 nuclídeos foi motivado pelo alto custo computacional para realizar tais simulações e, em virtude da baixa potência de operação do reator durante as décadas. No entanto, tendo em vista a prolongação da vida útil do Argonauta e, que em breve o mesmo receberá novos arranjos de elemento combustível e, a redução do custo computacional para realizar tais simulações, foi um outro motivo para propor o presente projeto.

O reator Argonauta tem sistemas de segurança inerentes ao seu design, que o tornam intrinsecamente seguro. Por exemplo, existem vários detectores de radiação no salão do reator e, sensores de corrente no prédio de proteção de radiação, todos ligados ao painel de controle do reator Argonauta. Então, em uma eventual extrapolação da valores de segurança (prefixados

pelo operador), o reator é desligado automaticamente. Além disso, o reator conta com barras de controle que são acionadas por ação gravitacional, em uma possível falta de energia da rede externa a instalação, bem como, conta com um sistema de drenagem de água que, também atuam por ação gravitacional, sendo o refrigerante/moderador (água) rapidamente remanejado para um reservatório para armazenamento, localizado abaixo do nível do salão do reator. Por fim, existe um sistema de injeção de nitrogênio que aumenta o coeficiente de vazio, diminuindo a capacidade de moderação de nêutrons, fazendo com que, o reator rapidamente fique em condição de desligamento e, portanto, sob controle [12].

Por outro lado, segundo o artigo [14], os produtos de fissão mesmo que voláteis, tais como o Iodo, apresentam vaporização somente a partir de 500K, isto é, em caso de acidente com exposição do núcleo, a liberação do nuclídeo começa a ser verificada a partir de 500K e, atingindo um pico de liberação de radio-nuclídeos entre 1250K e 1800K. Desta forma, mesmo com a exposição do núcleo do reator Argonauta, existe uma tendencia de manter os nuclídeos próximo ao núcleo, com exceção de nuclídeos em formato de gás e extremamente voláteis, o que poderia ser entendido como mais uma segurança intrínseca ao projeto do reator.

Uma visão geral do reator Argonauta e alguns de seus sistema de segurança pode ser visto na figura 2.

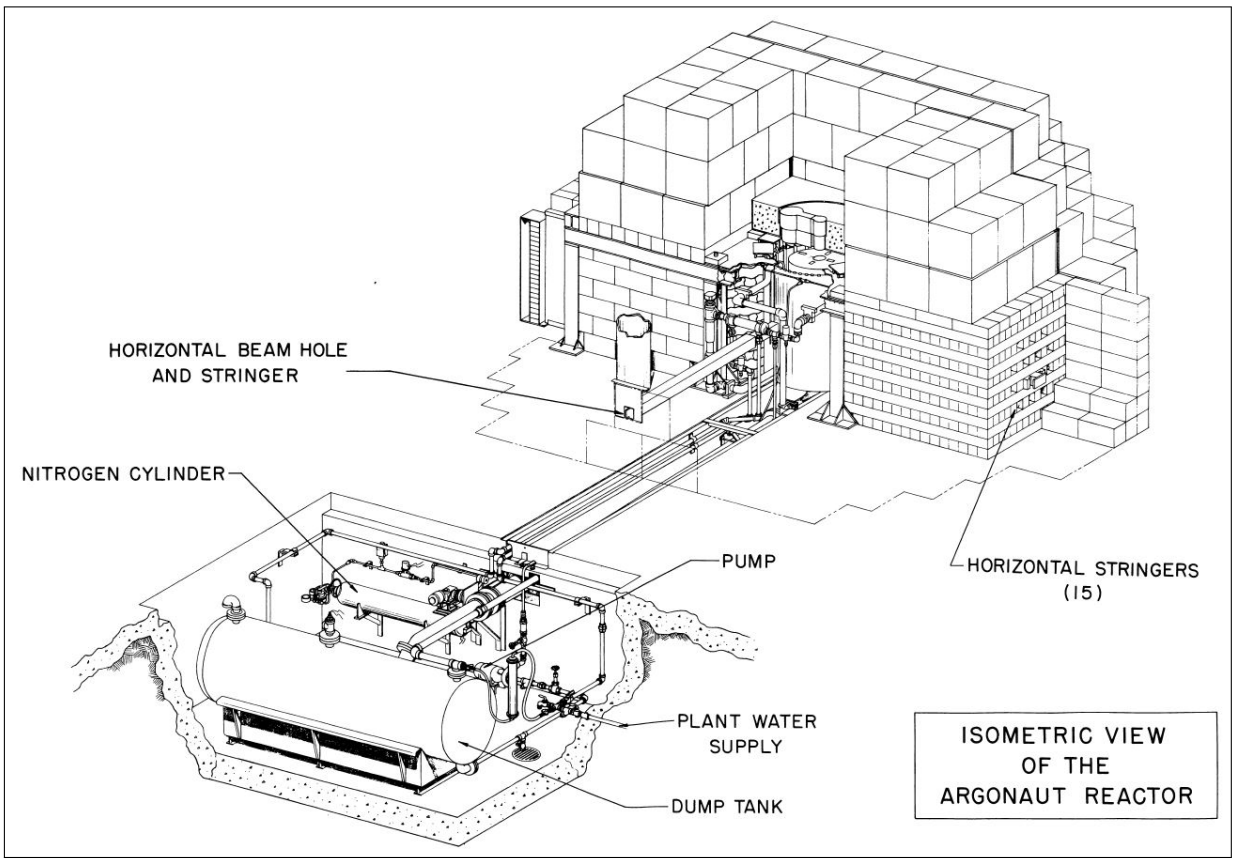

Figura 2: Visão geral do reator Argonauta e seus sistemas de segurança, fonte: [12]

1.2.2 O Raspberry Pi

O Raspberry Pi, figura 3, é uma Single Board Computer (SBC), isto é, uma placa de circuito único que, traz soldada nela: processador, memória, placa de vídeo e controlador de Input/Output(I/O). Ela tem aproximadamente o tamanho de um cartão de crédito e, algumas pessoas a consideram como um computador completo, devido a sua capacidade computacional similar aos desktops básicos. No entanto, ela traz uma grande vantagem sobre outros computadores tradicionais, que é a existência de 40 portas digitais, denominadas, General-Purpose Input/Output (GPIOs), que permitem a placa se comunicar com outros hardwares, através de vários protocolos, tais como, I2C e SPI. Essas características permitem que, ela seja usada em vários projetos de automação residencial e industrial, bem como, em auxiliar no aprendizado de eletrônica e programação [15]. As placas de Raspberry Pi foram pensadas e projetadas para custar $35 ou menos, a fim de facilitar no desenvolvimento de projetos. Atualmente já foram vendidas mais de 20 milhões de unidades pelo mundo e, devido a sua comunidade de desenvolvedores ativos, é a escolha certa para todos os tipos de projetos que requerem portabilidade, baixo custo financeiro e de energia [16].

A escolha do Raspberry Pi neste trabalho foi motivada pelo seu baixo custo e, pela necessidade de um dispositivo eletrônico que acoplas-se aos sensores de corrente do Argonauta, coletando esses dados, para serem utilizados como parâmetros de entrada para o código do simulador das concentrações isotópicas. Um estudo a viabilidade técnica da utilização deste tipo de dispositivo em tal função é descrito no capítulo 3.

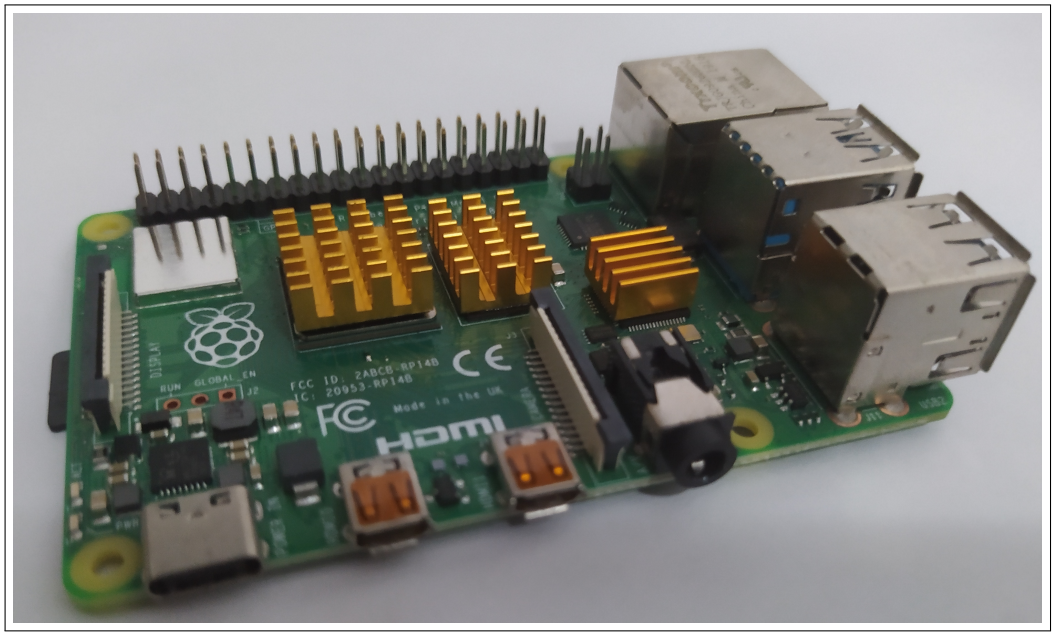

Figura 3: Raspberry Pi 4 modelo B, 2GB Ram, processador quad-core 1.5GHz, fonte: o autor, 2020

1.2.3 As Linguagens de Programação Escolhidas

Nesse trabalho foram utilizadas 3 linguagens de programação: o FORTRAN, o Python e o C++, por motivos diversos que serão descritos a seguir.

A escolha do FORTRAN foi motivada por esta ser uma linguagem de programação rápida, que consegue trabalhar de forma elementar com matrizes e vetores, sendo muito mais eficiente em tempo computacional que outras linguagens, como Python e Java [17]. Além disso, existem milhares de códigos da área nuclear, legados de décadas passadas, que estão somente implementados nesta linguagem. Outro fator relevante que motivou a escolha do FORTRAN, é que os códigos tradicionais que simulam o fluxo e/ou inventário de um reator, tais como o **CITATION**, **DIFF3D**, **SIMULATE** e **QUANDF**, são escritos em **FORTRAN IV**, **FORTRAN77** ou **FORTRAN90** [18].

O C/C++ foi utilizado neste trabalho como linguagem de programação para comparar o tempo de processamento com a linguagem o FORTRAN, pois ela é também eficiente e rápida e, consegue gerenciar memória e recursos hardwares em um baixo nível [19]. Nesse sentido, a linguagem C/C++ foi utilizada como um parâmetro de comparação para verificar a viabilidade técnica de realizar simulações computacionais em um Raspberry Pi, o que será melhor descrito no capítulo 3. Além disso, a linguagem C foi utilizada em conjunto com a biblioteca wiringPi para controlar a portas digitais(GPIOS) do Raspberry Pi, possibilitando a coleta de dados dos sensores do reator Argonauta.

O Python é utilizado neste trabalho devido ao OPENMC [20] ter uma *Application Programming Interface* (API) nesta linguagem que, facilita a modelagem de geometria de células e assemblys de reatores nucleares e, para produzir as seções de choque colapsadas em energia, para serem utilizadas em outros códigos (DIF1D). Maiores detalhes do uso do OPENMC e da API Python são abordados no capítulo 3.

1.3 Objetivos do Trabalho

Espera-se ao final deste trabalho o seguinte:

1. Verificar a viabilidade técnica de utilizar um Raspberry Pi em simulações computacionais ligados a área nuclear;
2. Implementar um sistema capaz de se acoplar ao sensores de corrente do Reator Argonauta para coletar informações de corrente do canal linear e logaritmo, a fim desses dados poderem serem utilizados como parâmetros para estimar a potencia do reator;

3. Produzir um modelo simplificado dos assemblys do Reator Argonauta no OPENMC e, a partir disso, obter as seções de choque multigrupo, colapsadas em 2 grupos de energia, considerando as diferentes regiões do reator Argonauta;

4. Implementar um código que seja capaz de simular em tempo real, as concentrações isotópicas e termo fonte do reator Argonauta, contemplando 20 actinídeos e 31 produtos de fissão, que devem ser avaliados em decorrência da sua alta volatilidade ou radioatividade;

5. Criar um sistema que sirva como mecanismo de salvaguarda para as operações do reator Argonauta.

1.4 A Estrutura da Dissertação

O restante da dissertação esta organizado da seguinte forma,

No capítulo 2, é apresentado os fundamentos matemáticos que são base para entender como os nêutrons se comportam dentro de um reator nuclear e, como o fluxo de nêutrons pode ser estimado através da Equação de Difusão de Nêutrons Multigrupo. Ainda neste, é descrito as aproximações utilizadas no trabalho e, é feita uma descrição da teoria da depleção de Actinídeos e produtos de Fissão, bem como, da solução de sistemas de equações diferenciais ordinárias de primeira ordem acopladas, pelo método de matriz exponencial.

No capítulo 3, são abordados um estudo de viabilidade técnica da implementação do Raspberry Pi 3B e 4B, em realizar a tarefa para obtenção de dados dos sensores de corrente do Argonauta. Ainda no capítulo, é abordado a modelagem do assembly do Argonauta com auxílio da API Python do OPENMC, a fim de obter as seções de choque colapsadas em dois grupos de energia, para utilizar como dados de entrada para o código DIF1D (simula perfil de fluxo) e estimar a potencia em cada assembly.

No capítulo 4, é feita uma breve descrição dos algoritmos utilizados, isto é, uma explicação do seu esquema modular, bem como, é descrito brevemente cada rotina.

No capítulo 5, são apresentados os resultados obtidos no trabalho.

Por fim, no capítulo 6, são apresentados as conclusões, considerações gerais e finais. Por outro lado, também são dadas sugestões de trabalho futuro, considerando os resultados obtidos.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentado a fundamentação teórica, isto é, a base necessária que foi empregada no desenvolvimento do trabalho. As próximas seções explicam as manipulações necessárias para obter o perfil de fluxo de nêutrons e, estimar as concentrações isotópicas em qualquer instante de tempo.

2.1 Um Breve Estudo de Nêutrons em Reatores

O nêutron é uma partícula sem carga, que esta presente em todos os materiais. Ela é a partícula fundamental que, possibilita o funcionamento de um reator nuclear. Devido a sua característica de ausência de carga, ela pode se aproximar dos núcleos de átomos pesados (ex.: U^{235} e Pu^{239}) sem grandes dificuldades e, assim, induzir uma fissão, isso é, uma divisão do átomo [18],[21].

Simultaneamente ao evento de fissão, outros nêutrons (1 a 4) são liberados, bem como, cerca de 200 MeV de energia é produzido por vários mecanismos físicos e químicos. No entanto, ao escapar do núcleo de um átomo, o nêutron fica livre para se movimentar pelas diversas regiões do reator, em movimento que se assemelha a difusão de um gás ideal, tendendo a promover deslocamento em direções de maior para menor concentração de nêutrons [18],[22]. Esse comportamento é matematicamente expresso através da conhecida lei de Fick, equação 1, que infere que as partículas sem carga, se movimentam de regiões de alta para baixa concentração, sendo este deslocamento proporcional ao coeficiente de difusão ($D = D(\vec{r}, t)$). Ainda sobre a equação 1, o termo $\vec{J}$ representa a corrente de nêutrons, enquanto o termo $\phi(\vec{r}, E, t)$ é o fluxo escalar de nêutrons, tendo ambos a mesma magnitude, isto é, $|\phi| = |\vec{J}| = \frac{\text{nêutrons}}{cm^2 \cdot s}$.

$$\vec{J} = -D\vec{\nabla}\phi(\vec{r}, E, t) \tag{1}$$

Além da fissão nuclear, nêutrons podem ser absorvidos dentro do núcleo do reator por outras reações nucleares, tais como, captura radioativa (n, γ) e absorção. Nesse sentido, um reator pode ser entendido com um sistema onde existe grande concorrência de eventos ao mesmo tempo, devido aos nêutrons se moverem pelo reator por caminhos que, a principio são muitos complexos. Entretanto, embora não seja trivial determinar o comportamento dos nêutrons em um reator, existe uma formulação matemática, denominada Equação de Difusão de Nêutrons, que será vista na próxima seção, que prediz muito bem o perfil do fluxo nêutrons, levando-se em conta todos os tipos de reação anteriormente citadas.

2.1.1 A Equação de Difusão de Nêutrons Multigrupo

O comportamento dos nêutrons no núcleo do reator pode ser descrito de forma muito simplificada, pela Equação de Difusão de Nêutrons Multigrupo, equação 2, que estima o fluxo de nêutrons dentro do reator nuclear. A descrição de como se obter a eq. 2, é mostrada no apêndice A.1 [18],[22], página 63.

$$\underbrace{-D_g\nabla^2\phi_g}_{1^\circ}+\underbrace{\Sigma_{ag}\phi_g}_{2^\circ}+\underbrace{\sum_{g'>g}\Sigma_{gg'}\phi_g}_{3^\circ}=\underbrace{\frac{1}{K}\sum_{g=1}^{G}\chi_g(\nu\Sigma_{fg}\phi_g)}_{4^\circ}+\underbrace{\sum_{g'<g}\Sigma_{g'g}\phi_g}_{5^\circ} \tag{2}$$

Na equação 2, o 1° representa a fuga, 2° a absorção, 3° o espalhamento *out-scattering*, 4° a produção de nêutrons e, por fim, o 5° termo, representa o espalhamento *in-scattering*. Nesse sentido, o estudo da equação 2 pode ser utilizado para estimar a distribuição do nêutrons no espaço e tempo, contemplando todo o reator ou um dado volume de controle arbitrário de interesse. Por outro lado, é importante destacar nesse momento que, a equação 2 é uma generalização considerando g grupos de energia.

Neste trabalho foi utilizado a equação de Difusão de Nêutrons com 2 grupos de energia, equação 3, porque esta é uma formulação eficiente para estimar o perfil do fluxo de nêutrons rápidos e térmicos, quando considerado estudo de caso em reatores térmicos[18],[22]. Muitos códigos computacionais ainda usam a implementação da Equação da Difusão de Nêutrons com 2 grupos de energia.

Opta-se por usar a formulação multigrupo ao invés do espectro continuo porque os computadores não conseguem trabalhar de forma eficiente com o espectro continuo, necessitando discretizar em pequenos grupos, onde os parâmetros nucleares são considerados constantes (durante o intervalo de simulação).

$$\begin{aligned} -D_g(\vec{r})\nabla^2\phi_g(\vec{r})+\Sigma_{ag}(\vec{r})\phi_g(\vec{r})+\sum_{g'>g}\Sigma_{gg'}(\vec{r})\phi_g(\vec{r}) \quad &= \\ =\frac{1}{K}\sum_{g=1}^{2}\chi_g(\nu\Sigma_{fg}(\vec{r})\phi_g)(\vec{r})+\sum_{g'<g}\Sigma_{g'g}(\vec{r})\phi_g(\vec{r}) & \end{aligned} \tag{3}$$

2.1.2 A Aproximação Quase-Estática

Um problema que surge analisando a equação 3 é que, o fluxo de nêutron, bem como, outros parâmetros nucleares variam no tempo. Isso dificulta os cálculos e, traz o problema da não linearidade do sistema de equações diferenciais de primeira ordem, utilizadas no estudo da variação das concentrações isotópicas dentro do reator. Isso ocorre porque, o fluxo de nêutrons depende do tempo e afeta as concentrações, porém, as próprias concentrações afetam o fluxo [9]-[10]. Logo, para contornar o problema da não linearidade, é comum recorre-se a aproximação quase-estática, que consiste em considerar que num breve intervalo de tempo ($\Delta t = t_l - t_{l-1}$) que, o fluxo de nêutrons e os parâmetros nucleares simulados, podem ser considerado constante. Isso torna os cálculos mais fáceis de serem solucionados. Então, a variável tempo é dividida em intervalos menores, onde os parâmetros nucleares podem ser considerados constantes.

2.1.3 A Discretização da Variável Energia

A energia é uma variável continua e, devido a essa característica, dificulta a implementação de códigos computacionais, pois os computadores não conseguem lidar muito bem com variáveis continuas, pois estas necessitam de muito esforço computacional (tempo e memória) para realizar as simulações pretendidas. Para contornar esse problema, adota-se uma abordagem similar a da soma de Riemann, isto é, divide-se o espectro continuo em intervalos finitos que, o computador consiga efetuar os cálculos. Logo, em suma, a variável de energia é dividida em G grupos de energia, conforme figura 4.

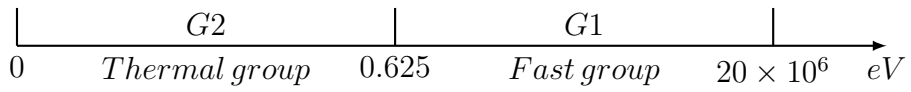

Figura 4: Discretização da variável energia em dois grupos

Os parâmetros nucleares multigrupo, tais como, seção de choque de reação do tipo $x(x = a, f, c, s)$, contido na equação 3, são estimados com base nas definições matemáticas a seguir [18],[22],

$$\phi_g = \int_{E_g}^{E_{g-1}} \phi(E)dE, \text{ se : } \begin{cases} \text{grupo térmico : } \begin{cases} E_g = 0eV, \\ E_{g-1} \approx 0.625eV \end{cases} \\ \text{grupo rápido : } \begin{cases} E_g = 0.625eV, \\ E_{g-1} = 20MeV \end{cases} \end{cases} \tag{4}$$

$$\Sigma_{xg} = \frac{\int_{E_g}^{E_{g-1}} \Sigma_x(E)\phi(E)dE}{\phi_g}, \text{ se : } \begin{cases} \text{grupo térmico : } \begin{cases} E_g = 0eV, \\ E_{g-1} \approx 0.625eV \end{cases} \\ \text{grupo rápido : } \begin{cases} E_g = 0.625eV, \\ E_{g-1} = 20MeV \end{cases} \end{cases} \tag{5}$$

2.1.4 A Discretização Espacial de Um Reator

Um reator nuclear é um sistema heterogêneo, isto é, que apresenta grandes variações de características (ex.:seção de choque) de uma região pra outra. Além disso, em um reator nuclear pode haver finitas descontinuidades devido a transição de materiais com seções de choque muito diferentes [13],[23]. Logo, também é necessário realizar uma discretização espacial, a fim de permitir o computador processar os cálculos, bem como, faz-se necessário uma homogeneização de cada região, levando se em consideração todos os materiais constituintes e seus respectivos pesos neutrônicos [13]. Nesse sentido, a discretização espacial do reator Argonauta é feito conforme ilustra a figura 5. Enquanto a homogeneização dos parâmetros nucleares das regiões é feito com algum programa apropriado (ex.: OPENMC), para colapsar os parâmetros nucleares em dois grupos de energia [20].

O reator Argonauta foi dividido em 32 regiões (4 tamanhos diferentes) homogeneizadas, contemplando de 4 tipos diferentes. A primeira região é o cerne ativo dos assemblys, contendo U_3O_8, Alumínio e Água. A segunda região, a parte inerte dos assemblys, contendo Alumínio e água. Região 3, possui apenas grafite (refletor). Enquanto a região 4, é o grafite externo ao reator. A figura 5, mostra a discretização da metade esquerda do reator Argonauta.

4	3	2	1	2	3	3	2	1	2	3	3	2	1	2	3

Figura 5: Discretização Espacial da metade esquerda do Reator Argonauta

2.1.5 Obtenção do Fluxo de Nêutrons

Segundo [23], a equação de difusão de nêutrons com dois 2 grupos de energia, considerando o caso unidimensional, pode ser discretizada numericamente, pelo método das diferenças finitas centrado na malha, através das equações 6 e 7.

$$-D_1\left(\frac{\phi^1_{i+1}-2\phi^1_i+\phi^1_{i-1}}{\Delta x^2}\right)+\Sigma_1\phi^1_i=\frac{1}{k_{eff}}\left[\nu\Sigma_{f1}\phi^1_i+\nu\Sigma_{f2}\phi^2_i\right] \tag{6}$$

$$-D_2\left(\frac{\phi^2_{i+1}-2\phi^2_i+\phi^2_{i-1}}{\Delta x^2}\right)+\Sigma_2\phi^2_i=\Sigma_{1\to2}\phi^1_i \tag{7}$$

As equações 6 e 7 devem ser resolvidas para cada malha da discretização do reator. Todavia, vale destacar nesse momento que, certas condições de contorno e continuidade de fluxo e de corrente de nêutrons devem ser atendidas em cada malha e interfaces. Enquanto a primeira e ultima malha geral do reator, devem atender a condições de contorno que podem ser: condição de fluxo nulo, de corrente nula e de albedo. No entanto, para manter o foco no projeto, sem estender demais sobre o assunto, a abordagem adotada neste trabalho foi a mesma descrita em [11], [23].

Por outro lado, as equações 6 e 7, podem ser rescritas de uma forma matricial, da forma,

$$A_1=\begin{bmatrix}\left(\frac{2D_1}{\Delta x^2}+\Sigma_1\right) & \left(\frac{-D_1}{\Delta x^2}\right) & 0 & \dots & 0\\ \left(\frac{-D_1}{\Delta x^2}\right) & \ddots & \ddots & \ddots & \vdots\\ 0 & \ddots & \ddots & \ddots & 0\\ \vdots & \ddots & \ddots & \ddots & \vdots\\ 0 & \dots & 0 & \ddots & \vdots\end{bmatrix} \tag{8}$$

$$A_2 = \begin{bmatrix} \left(\frac{2D_2}{\Delta x^2} + \Sigma_2\right) & \left(\frac{-D_2}{\Delta x^2}\right) & 0 & \dots & 0 \\ \left(\frac{-D_2}{\Delta x^2}\right) & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \ddots & \vdots \end{bmatrix} \tag{9}$$

$$E_{gg'} = \begin{bmatrix} \Sigma_{1\to 2} & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & \ddots \end{bmatrix} \tag{10}$$

$$F_1 = \begin{bmatrix} \nu\Sigma_{f1} & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & \ddots \end{bmatrix} \tag{11}$$

$$F_2 = \begin{bmatrix} \nu\Sigma_{f2} & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & \ddots \end{bmatrix} \tag{12}$$

As matrizes anteriores, podem ser compactadas em uma forma mais compacta, como se segue:

$$\begin{bmatrix} A_1 & 0 \\ E_{gg'} & A_2 \end{bmatrix} \begin{bmatrix} \phi^1 \\ \phi^2 \end{bmatrix} = \frac{1}{k_{eff}} \begin{bmatrix} F_1 & F_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \phi^1 \\ \phi^2 \end{bmatrix} \tag{13}$$

Ainda sobre a equação 13, tem-se que ela é um problema de autovalor, da forma como é descrito no apendice A.1. Nesse sentido, a equação 13 ainda pode ser reescrita da forma,

$$A\widetilde{\phi} = \frac{1}{k_{eff}} F\widetilde{\phi} \tag{14}$$

Então, a equação 14 é um problema de autovalor e autovetor a ser resolvido, que segundo [23], pode ser solucionado aplicando-se algum método iterativo, como o método das potencias. Resumidamente, este se baseia nos seguintes passos:

1°: Atribuir valores arbitrários para o vetor fluxo inicial ($\widetilde{\phi^0}$), como exemplo:

$$\widetilde{\phi^0} = \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ 1 \end{bmatrix} \tag{15}$$

2°: Realizar o produto $A \cdot \widetilde{\phi^0}$, de forma a obter o próximo vetor fluxo da interação $\widetilde{\phi^1}$;

3°: A partir do vetor anterior obtido, dividir o resultado pelo termo dominante do vetor (C_d), isto é, pelo maior elemento em módulo do vetor; Isso, promove obter o vetor fluxo da próxima iteração ($\widetilde{\phi^1}$);

4°: Repetir os passos 2° e 3° até que o vetor $\widetilde{\phi^{p+1}}$ convergir após n iterações.

2.2 A Depleção Isotópica Durante o Ciclo do Combustível

O estudo da depleção isotópica consiste na estimativa das variações das concentrações dos nuclídeos no tempo.

Durante o ciclo do combustível, muitos isótopos surgem, alguns inclusive, dependendo da concentração, podem dificultar a partida do reator ou, em caso mais graves, ocasionar acidentes, devido a mudança abrupta do estado de operação do reator (subcrítico ou supercrítico), como exemplo, o Xe^{135}.

2.2.1 Introdução ao Estudo das Variações das Concentrações Isotópicas no Tempo

Em um reator nuclear existem dois tipos de isótopos que podem surgir durante o ciclo do combustível que são: os Actinídeos e Produtos de Fissão (ou fragmentos de fissão). Os Actinídeos são elementos com $Z > 89$ [24] que, estão presente nas hastes de combustível frescas ou surgem por reações de absorção, captura radioativa (n, γ) e decaimento (β e α). Por outro lado, os produtos de fissão não estão presentes nas hastes de combustível fresco, eles surgem a partir da fissão de actinídeos físseis ou pelo decaimento de elementos instáveis que foram produzidos diretamente por fissão. Em suma, os produtos de fissão tem aproximadamente metade da massa do nuclídeo pai e, normalmente são instáveis e decaem para outro elemento após algum tempo, buscando uma configuração quântica mais estável (menor energia). Além disso, podem ocorrer reações de realimentação (n,2n) tanto em Actinídeos quanto produtos de fissão, mas isso não foi considerado neste trabalho. Portanto, o aumento e diminuição das concentrações isotópicas tanto de Actinídeos, quanto produtos de fissão tem caminhos complexos e, serão melhor entendidos com as próximas seções.

2.2.2 Os Actinídeos Avaliados

Neste trabalho, foram avaliados 5 Actinídeos (U, Np, Pu, Am e Cm) e, um total 20 elementos isótopos destes, a saber, 6 de Urânio (U^{234}, U^{235},U^{236},U^{237},U^{238} e U^{239}), 5 do Plutônio(Pu^{238}, Pu^{239}, Pu^{240}, Pu^{241} e Pu^{242}), 4 do Neptúnio (Np^{237}, Np^{238}, Np^{239} e Np^{240}), 3 do Amerício (Am^{241}, Am^{242} e Am^{243}) e, por fim, 2 do Cúrio (Cm^{242} e Cm^{243}).

As equações que descrevem as variações desses Actinídeos no tempo, encontra-se no apêndice A.2, página 66. Enquanto, a cadeia de decaimento desses elementos pode ser vista pela figura 6.

A tabela 1, traz informações resumidas sobre os actinídeos avaliados no trabalho e suas relações entre si, considerando reações de captura e decaimento.

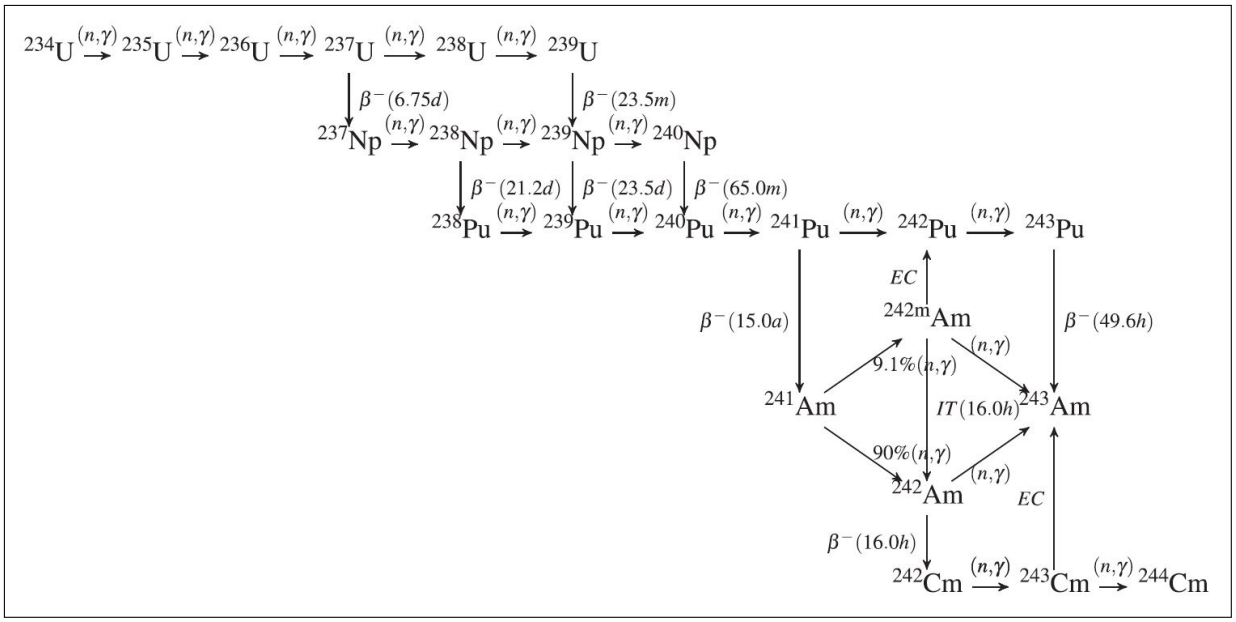

Figura 6: Cadeia de Actinídeos

Isótopo	Meia-vida	Índice	Isótopo Captura	Isótopo Decaimento
U^{234}	2.455×10^5 a	1	2	0
U^{235}	7.040×10^8 a	2	3	0
U^{236}	2.343×10^7 a	3	4	0
U^{237}	6.749×10^0 d	4	5	7
U^{238}	4.468×10^9 a	5	6	0
U^{239}	2.346×10^1 m	6	0	9
Np^{237}	2.144×10^6 a	7	9	0
Np^{238}	2.105×10^0 d	8	9	11
Np^{239}	2.356×10^0 d	9	10	12
Np^{240}	7.220×10^0 m	10	0	13
Pu^{238}	8.774×10^1 a	11	12	1
Pu^{239}	2.410×10^4 a	12	13	0
Pu^{240}	6.541×10^3 a	13	14	0
Pu^{241}	1.433×10^1 a	14	15	16
Pu^{242}	3.730×10^5 a	15	0	0
Am^{241}	4.326×10^2 a	16	17	7
Am^{242}	1.601×10^1 h	17	18	19
Am^{243}	7.367×10^3 a	18	0	9
Cm^{242}	1.628×10^2 d	19	20	15
Cm^{243}	2.890×10^1 a	20	0	18

Tabela 1: Dados de meia-vida de actinídeos e suas relações de captura e decaimento, legenda: m(minutos), h(horas), d(dias) e a(anos).

Um dado Actinídeo genérico E_x^A (ex.: U^{235}), com massa A, pode ter sua concentração alterada no tempo, devido ao decaimento (próprio e de um elemento mais pesado, E_y^{A+1}). Por

outro lado, a captura de nêutrons por elemento E_z, com massa $A-1$, pode resultar no aumento da concentração do elemento E_x^A. Todavia, se o elemento E_x^A absorve nêutrons e não fissiona, a sua concentração ira diminuir. Por fim, existem reações de realimentação (n,2n) que podem alterar as concentrações do elemento E_x^A. Todas essas ponderações são ilustradas na figura 7.

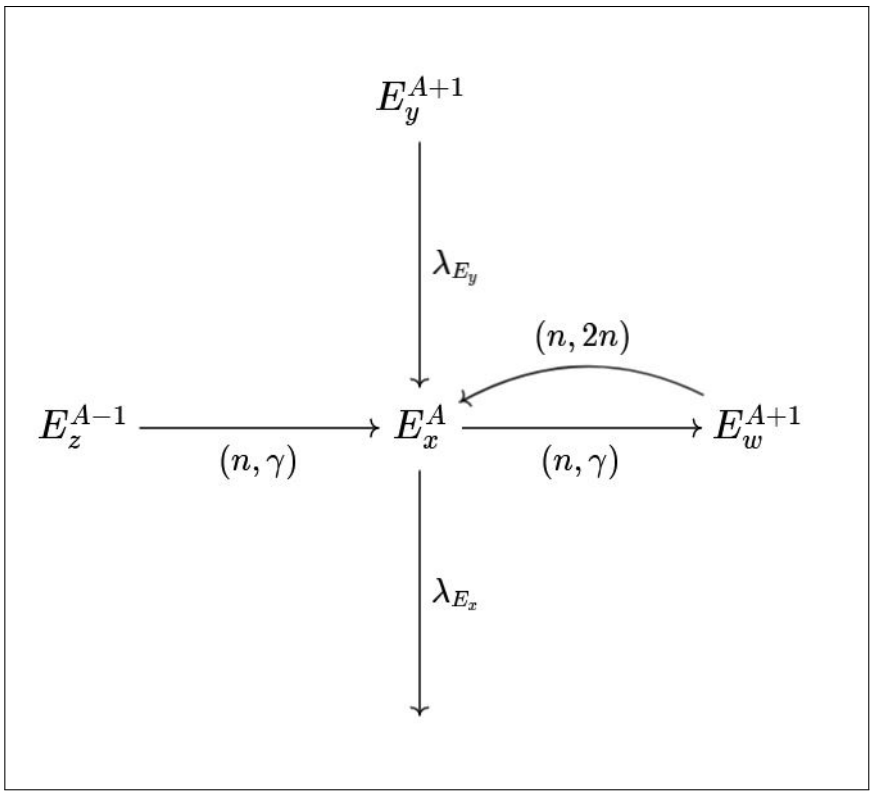

Figura 7: Modos de transmutação de Actinídeos, fonte: o autor, 2020

A variação da concentração no tempo, do Actinídeo genérico E_x, pode ser estimada com base na analise da figura 7 e, é matematicamente expressa pela equação 16,

$$\begin{aligned}\frac{\partial N_{E_x}(\vec{r},t)}{\partial t} = \quad & -N_{E_x}(\vec{r},t)\sum_{g=1}^{2}\sigma_{a,g}^{E_x}\phi_g(\vec{r},t) - \lambda_{E_x}N_{E_x}(\vec{r},t) \\ & +N_{E_z}(\vec{r},t)\sum_{g=1}^{2}\sigma_{\gamma,g}^{E_z}\phi_g(\vec{r},t) + N_{E_w}\sum_{g=1}^{2}\sigma_{(n,2n),g}^{E_w}\phi_g(\vec{r},t) \\ & +\lambda_{E_y}N_{E_y}(\vec{r},t)\end{aligned} \tag{16}$$

A equação 16 pode ser manipulada, de forma a evidenciar os termos comuns, conforme a expressão contida na equação 17.

$$\frac{\partial N^n_{E_x}}{\partial t}\Big|_{t=t_l} = -\left(\lambda_{E_x} + \sum_{g=1}^{2} \sigma^{E_x}_{a,g}\overline{\phi^n_g}\right) N^n_{E_x} + \left(\sum_{g=1}^{2} \sigma^{E_z}_{\gamma,g} N^n_{E_z} + \sigma^{E_w}_{(n,2n),g} N^n_{E_w}\right) + \lambda_{E_y} N^n_{E_y} \tag{17}$$

Onde:

$\sigma^u_{x,g} \equiv$ seção de choque microscópica para a reação do tipo $x (x = a, c, f, s)$, considerando o nuclídeo u e grupo de energia g;

$N^n_u \equiv$ concentração isotópica do nuclídeo u, na malha n, no instante de tempo analisado;

$\lambda_u \equiv$ constante de decaimento do nuclídeo u;

$\overline{\phi^n_g} \equiv$ fluxo médio de nêutrons na malha n, considerando o grupo de energia g.

2.2.3 Os Produtos de Fissão Avaliados

Neste trabalho, foram avaliados 13 produtos de fissão (Br, Kr, Sr, Y, Nb, Zr, Mo, Te, I, Cs, Ce, Xe e Sr) e, um total 30 isótopos, a saber, 2 do Bromo (Br^{85} e Br^{86}), 3 do Criptônio (Kr^{85m}, Kr^{85} e Kr^{88}), 4 do Ítrio (Y^{90}, Y^{91}, Y^{92} e Y^{93}), 2 do Nióbio (Nb^{95} e Nb^{96}), 1 do Molibdênio (Mo^{99}), 1 do Zircônio (Zr^{95}), 2 de Telúrio (Te^{133} e Te^{134}), 7 do Iodo (I^{130}, I^{131}, I^{132}, I^{133}, I^{134}, I^{135} e I^{136}), 3 do Césio (Cs^{134}, Cs^{137} e Cs^{138}), 1 Cério (Ce^{141}), 2 do Xenônio(Xe^{133} e Xe^{135}) e, por fim, 2 do Estrôncio (Sr^{91} e Sr^{92}) .

Um dado produto de fissão pode ser resultado do evento de fissão de diferentes Actinídeos, bem como, devido a captura de nêutrons por elementos menores ou, ainda pelo decaimento de elementos maiores. O rendimento de fissão, ou *fission yield* (Γ^i_j), é uma grandeza que estima a probabilidade do actinídeo j produzir o nuclídeo i em um evento de fissão, ou seja, há concorrência entre os produtos de fissão que surgem dentro do reator. Todas essas ponderações anteriores, podem ser matematicamente expressas por,

$$\frac{\partial N_i(\vec{r},t)}{\partial t} = \sum_{j=1}^{Actinides} \Gamma^i_j N_j(\vec{r},t) \sum_{g=1}^{2} \sigma^j_{f,g}\phi_g(\vec{r}),t + \lambda_{i-1}N_{i-1}(\vec{r},t) + -\left(\lambda_i + \sum_{g=1}^{2} \sigma^i_{a,g}\phi_g(\vec{r}),t\right) N_i(\vec{r},t) \tag{18}$$

Pode-se ainda simplificar a equação anterior, pois foi feita a homogeneização e adotou-se a aproximação quase-estática, desta forma tem-se que,

$$\frac{dN_i^n}{dt}\Big|_{t=t_l} = \sum_{j=1}^{Actinides} \Gamma_j^i N_j \sum_{g=1}^{2} \sigma_{f,g}^j \bar{\phi}_g^n\Big|_{t=t_l} + \lambda_{i-1} N_{i-1}^n\Big|_{t=t_l} - \left(\lambda_i + \sum_{g=1}^{2} \sigma_{a,g}^i \bar{\phi}_g^n\right) N_i^n\Big|_{t=t_l} \quad (19)$$

Onde:

$\Gamma_j^i \equiv$ Rendimento ou *fission yield*, que representa a porcentagem de produção do nuclídeo i, devido a fissão do actinídeo j;

$\lambda_i, \lambda_{i-1} \equiv$ Constante de decaimento do nuclídeo i e $i-1$, respectivamente;

$N_i^n, N_{i-1}^n \equiv$ Concentração do nuclídeo i e $i-1$ na malha n;

$\sigma_{x,g}^i \equiv$ Seção de choque microscópica para o tipo de reação x ($x =, a, c, f, s$), considerando o grupo de energia g e o nuclídeo i;

$\bar{\phi}_g^n \equiv$ Fluxo escalar médio de nêutrons na malha n, considerando o grupo de energia g.

As relações de transmutação entre os produtos de fissão, devido a captura de nêutrons e decaimento, bem como, o surgimento de cada produto, devido a fissão do U^{235} e Pu^{239} pode ser observado pelos dados da tabela 2.

Isótopo	Meia vida	Rendimento U^{235}	Rendimento Pu^{239}	Índice do Isótopo	Isótopo Decaimento	Isótopo Captura
Br^{85}	2.900×10^{0} m	2.3565100E-03	1.4804700e-03	1	3	2
Br^{86}	1.430×10^{1} s	4.6070700E-03	3.7812000e-03	2	31	3
Kr^{85}	1.073×10^{1} a	5.9235500E-05	1.2379430e-04	3	31	31
Kr_{m}^{85}	4.480×10^{0} h	3.1508250E-04	1.2379430e-04	4	31	31
Kr^{88}	2.825×10^{0} h	1.7368700E-02	7.4693500e-03	5	31	31
Sr^{91}	9.650×10^{0} h	2.5115600E-03	3.8600900e-03	6	9	7
Sr^{92}	2.611×10^{0} h	1.0780700E-02	1.0241700e-02	7	10	31
Y^{90}	6.405×10^{1} h	8.9934700E-08	1.9295910e-06	8	31	9
Y^{91}	5.851×10^{1} d	3.3086320E-06	2.4794590e-05	9	31	10
Y^{92}	4.200×10^{-6} s	7.1675000E-04	4.5069400e-04	10	31	11
Y^{93}	1.018×10^{1} h	1.0896040E-03	1.7401850e-03	11	31	31
Zr^{95}	6.403×10^{1} d	1.2758600E-03	1.2546000e-03	12	13	31
Nb^{95}	3.499×10^{1} d	1.3114280E-06	6.9685200e-06	13	31	14
Nb^{96}	2.335×10^{1} h	5.4542400E-06	3.6032400e-05	14	31	31
Mo^{99}	6.592×10^{1} h	4.2929100E-04	3.7499000e-04	15	31	31
Te^{133}	1.250×10^{1} m	4.1452900E-02	4.6560400e-02	16	21	17
Te^{134}	4.180×10^{1} m	6.2322100E-02	4.3964200e-02	17	22	31
I^{130}	1.236×10^{1} h	2.2217980E-06	4.6900000e-05	18	31	19
I^{131}	8.025×10^{0} d	3.9262400E-05	2.2929100e-04	19	31	20
I^{132}	2.295×10^{0} h	1.8310740E-04	2.6574100e-03	20	31	21
I^{133}	2.083×10^{1} h	1.6546840E-03	1.1072610e-02	21	25	22
I^{134}	5.250×10^{1} m	8.6674300E-03	2.6195700e-02	22	31	23
I^{135}	6.580×10^{0} h	2.9352400E-02	4.2865600e-02	23	26	24
I^{136}	8.340×10^{1} s	2.5775180E-02	2.8910830e-02	24	31	31
Xe^{133}	5.247×10^{0} d	2.5586840E-05	4.3237900e-04	25	31	26
Xe^{135}	9.140×10^{0} h	1.8192800E-03	1.0661980e-02	26	31	31
Cs^{134}	2.065×10^{0} a	7.7301800E-08	6.6985800e-06	27	31	31
Cs^{137}	3.008×10^{1} a	6.0160100E-04	5.9719700e-03	28	31	28
Cs^{138}	3.250×10^{1} m	4.6782300E-03	9.0088000e-03	29	31	31
Ce^{141}	3.251×10^{1} d	5.0030600E-08	2.2895100e-06	30	31	31
$Fict$	———————-	7.8200000E-01	7.4100000E-01	31	-1	31

Tabela 2: Dados de meia-vida de produtos de fissão e suas relações de transmutação entre si, devido a decaimento e captura de nêutrons. Legenda: s(segundos), m(minutos), h(horas), d(dias), a(anos).

2.2.4 Forma Matricial das Equações de Depleção

Os 20 Actinídeos e 30 produtos de fissão estudados neste trabalho, que são descritos pelas equações 17, 19 e que, foram mostrados nas duas ultimas seções, podem ser elegantemente compactados em uma forma matricial, da seguinte forma,

$$\frac{d}{dt}\begin{bmatrix} N_1^n(t) \\ N_2^n(t) \\ \vdots \\ N_N^n(t) \end{bmatrix} = \begin{bmatrix} M_{1,1}(t) & M_{1,2}^n(t) & \cdots & M_{1,N}^n(t) \\ M_{2,1}(t) & M_{2,2}^n(t) & \cdots & M_{2,N}^n(t) \\ \vdots & \vdots & \vdots & \vdots \\ M_{N,1}(t) & \cdots & \cdots & M_{N,N}^n(t) \end{bmatrix} \begin{bmatrix} N_1^n(t) \\ N_2^n(t) \\ \vdots \\ N_N^n(t) \end{bmatrix} \tag{20}$$

Onde, $\vec{N}$ é o vetor concentração, que contempla o total de elementos analisados (Actinídeos + Produtos de fissão), ou seja, a dimensão de matriz evolução, contido na equação 20, é de ordem quadrada e tem dimensão 50. A equação 20 pode ainda ser reescrita da seguinte forma,

$$\frac{d\vec{N}^n(t)}{dt} = \sum_{j=1}^{Act} M_{ij} N_j^n(t) \tag{21}$$

Onde:

$\vec{N}^n(t) \equiv$ É o vetor concentração dos nuclídeos na malha, e no instante de tempo t;

$M_{ij} \equiv$ É o elemento da matriz evolução do sistema.

2.2.5 Solução das Equações de Depleção com Método da Exponencial da Matriz

Foi desenvolvido na seção anterior a foma matricial do sistema de equações de depleção de actinídeos e produtos de fissão no tempo. Nesta seção, é mostrado a forma de resolver o sistema, através do método da exponencial da matriz. Primeiramente, reescreve-se a equação 21, da forma,

$$\frac{dN^n(t)}{dt} = E_l^n N^n(t), \quad t_{l-1} \leq t_l < t_{l+1} \tag{22}$$

Pode-se manipular a equação 22, isolando os termos iguais e integrando de ambos os lados, com os respectivos intervalos de integração, desta forma tem-se,

$$\int_{N^n(t=0)}^{N^n(t)} \frac{dN^n(t)}{N^n(t)} = \int_0^t E_l^n dt \tag{23}$$

Realizando a integração da equação 23, obtém-se,

$$ln\Big(\frac{N^n(t)}{N_0^n}\Big) = E_l^n t \rightarrow N^n(t) = e^{E_l^n t} \cdot N_0^n \tag{24}$$

Onde:

$N(t) \equiv$ É o vetor concentração no instante de tempo t;

$N_0^n \equiv$ É o vetor concentração inicial, durante o intervalo da queima, isto é, $N_0^n = N^n(t=0)$;

$E_l^n \equiv$ É a matriz de evolução da depleção;

$e^{E_l^n t} \equiv$ É a exponencial da matriz evolução no tempo, que será resolvida.

De forma sucinta, a equação 24 pode ser reescrita, na forma da equação 25, que pode ser resolvida por algum método de matriz exponencial. Por exemplo, em [27] são descrito pelo menos 19 modos de resolver essa matriz, considerando as vantagens e desvantagens de cada forma. No entanto, neste trabalho foi utilizado o algoritmo desenvolvido **expm** no IEN, para resolver esse sistema, considerando que a matriz $A(t)$, contida na equação 25, é altamente esparsa.

$$N^n(t) = A(t) \cdot N_0^n,\ A(t) = e^{E_l^n t} \tag{25}$$

2.3 Cálculo de Atividade de Radionuclídeos

A maioria dos actinídeos e produtos de fissão são instáveis e, após algum tempo tende a decair, isto é, transmutar em outro elemento por decaimento (α, β, γ), buscando uma configuração eletrônica mais estável(menor energia, ex.: núcleo par-par). Nesse aspecto, a instabilidade de um dado radionuclídeo no tempo, é medida pelo número de desintegrações por

segundo (Bequerel, unidade Bq), que é denominada por Atividade e, sendo expressa matematicamente por,

$$A_i = \frac{N_i}{M_i} \cdot \underbrace{6.022 \cdot 10^{23}}_{Avogrado} \cdot \frac{ln(2)}{\lambda_i} \tag{26}$$

Onde:

$N_i \equiv$ Concentração isotópica do nuclídeo i em quilogramas;

$M_i \equiv$ Massa molar do nuclídeo i;

$\lambda_i \equiv$ Meia-vida do nuclídeo i.

Então, resolvendo o sistema contido em 25, pode-se estimar as concentrações isotópicas em um instante de tempo t. A partir dessas informações, é possível calcular o termo fonte do reator, utilizando a equação 26.

3 METODOLOGIA

Neste capítulo é apresentado um estudo da viabilidade técnica do Raspberry Pi em realizar simulações ligados a área nuclear, destacando as especificações dos dispositivos e componentes utilizados, a fim de serem ligados ao sensores de corrente do reator Argonauta. Ainda no capítulo, é abordado como foi realizada a modelagem do assembly do reator Argonauta, seguindo as informações contidas em [13] e baseado na documentação do OPENMC.

3.1 Estudo da Viabilidade Técnica da Simulação das Concentrações Isotópicas do Reator Argonauta no Raspberry Pi

Para se verificar a viabilidade técnica, do Raspberry Pi em realizar simulações computacionais ligados a área nuclear, foi utilizado um Raspberry Pi 3B, com sistema operacional Raspbian instalado. Além disso, vários componentes eletrônicos e desktops, com diferentes níveis de capacidade computacional, rodando sistemas operacionais Windows e Linux, foram utilizados como parâmetros de comparação ao Raspberry Pi 3B. Para maior informações sobre a especificação dos componentes, consultar o apêndice B.1, contido na página 79.

3.1.1 Componentes Utilizados no Projeto

Os componentes eletrônicos utilizado nesse projeto em sua ampla maioria foram adquiridos através de importação isolada (Raspberry Pi) e através de Kit (Kit Stater Raspberry Pi da OSOYOO). A lista a seguir descreve os componentes utilizados no desenvolvimento deste trabalho:

1. Rapberry Pi 3B/4B;
2. Protoboard;
3. Cabo Extensor 40 vias;
4. Adaptador 40 vias para Protoboard;
5. Fios Jumpers macho/macho para interconexões na Protoboard;
6. Resistores 200 Ohms;
7. Leds 5mm de várias cores;

8. 2 Potenciômetros de 10K;
9. Analog to Digital Converter (ADC, Conversor Analógico para Digital), modelo MCP3008;
10. Matriz LCD;
11. Multimetro Digital.

A figura 8 ilustra os componentes que foram utilizados no projeto.

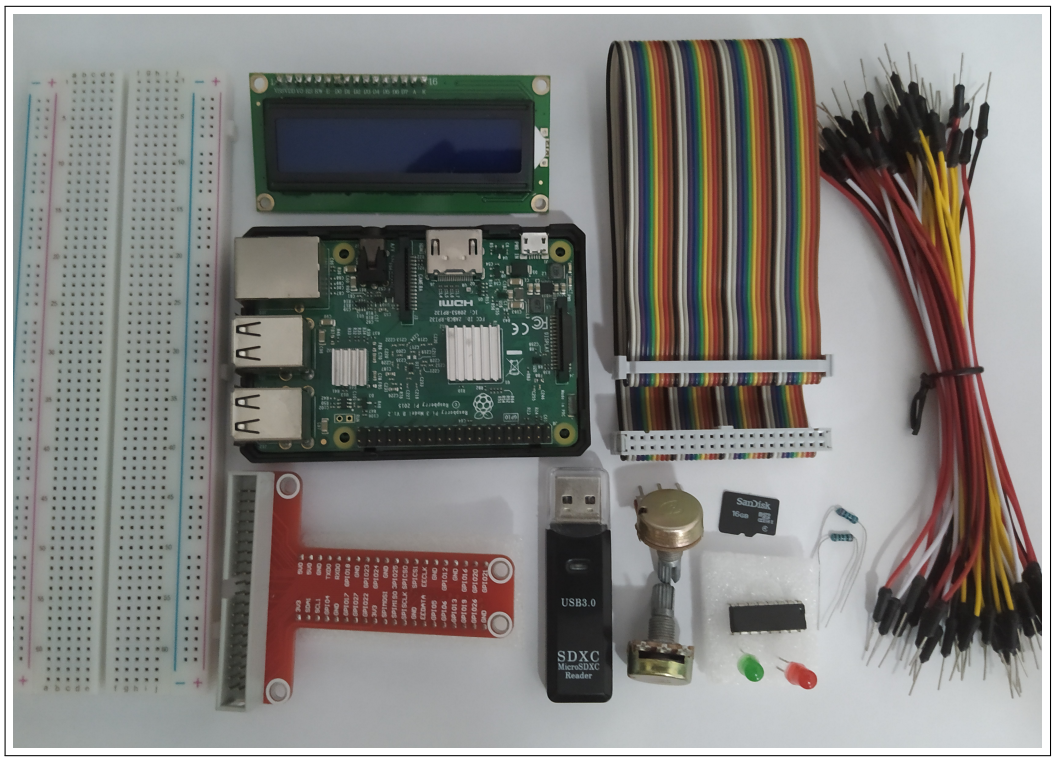

Figura 8: Componentes utilizados na verificação de viabilidade técnica em realizar simulações com um Raspberry Pi, fonte: o autor, 2020

3.1.2 A Solução do Problema da Leitura de Medidas Analógicas Pelo Raspberry Pi

O primeiro problema a se enfrentar na leitura dos sensores de corrente do reator Argonauta, é que essas medidas são grandezas analógicas e, o Raspberry Pi, diferente do ARDUINO, não traz embarcado nele, um conversor analógico para digital (ADC), que possibilite tais leituras. Necessitando, portanto, de um ADC externo a placa, para fazer tais aferições.

Nesse trabalho foi utilizado um ADC 3008, que acompanha o kit starter do Raspbery PI da OSOYOO. O esquema das portas do ADC podem ser vista no apêndice B.1, página 79. A implementação da leitura de corrente pelo Raspberry Pi foi feita através de modificação no código distribuído no Kit.

Para simular os dados de corrente do Argonauta, foi utilizado dois potenciômetros de 10K, que simbolicamente representam os dados do canal linear e logaritmo do reator Argonauta. O resultado de tal implementação pode ser visto na figura 9.

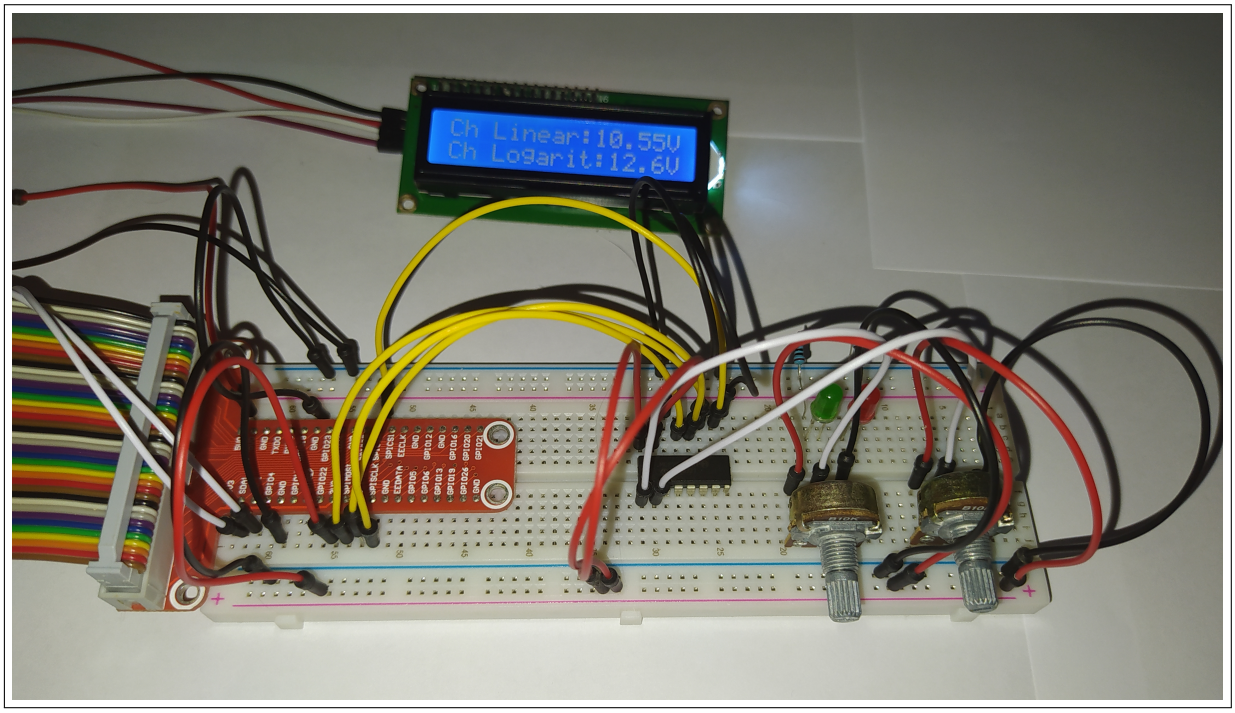

Figura 9: Implementação do MCP3008 no Raspberry Pi 3B, possibilitando a leitura de sensores analógicos. Os potênciometos ilustram os sensores de corrente do Argonauta, fonte: o autor, 2020

3.1.3 Benchmark para Aferir o Tempo de Processamento

Para conseguir estimar se seria possível realizar simulações computacionais ligados a área nuclear, em um pequeno Raspberry Pi 3B, o código DIF1D (simula o fluxo de nêutrons e K_{eff}, para um reator unidimensional), originalmente escrito em FORTRAN, foi convertido para a linguagem C++. O reator utilizado na simulação, foi um fornecido pela International Atomic Energy Agency (IAEA). A conversão do código foi motivada para comparar as duas linguagens de programação quanto ao tempo de processamento em realizar a tarefa de obter o fluxo de nêutrons e o k_{eff} do reator. Esse trabalho em questão, foi abordado em apresentação na International Nuclear Atlantic Conference (INAC) [25], em 2019. Resumidamente, o benchmark nas duas linguagens de programação foram executados no Raspberry Pi 3B e, em vários computadores tradicionais que, tinham capacidade computacionais diferentes.

Conforme resultados apresentados na tabela 3, é possível inferir que, o Raspberry Pi 3B, embora pequeno, apresenta performance computacional relativamente satisfatória em realizar a simulação do fluxo de nêutrons no caso de um reator unidimensional.

Motivado pelos resultados anteriores citados, foi adquirido a versão mais atual do Raspberry Pi comercializado, o modelo 4B, para dar continuidade ao projeto, de forma a melhorar os resultados demonstrados na tabela 3. Neste aspecto, com base nesse dados apresentados até aqui, é possível inferir que o, Raspberry Pi 4B demonstra-se um dispositivo capaz de realizar simulações de tarefas ligados a área nuclear com performance relativamente satisfatória, tendo como vantagem, a portabilidade, preço, baixo consumo de energia e, com possibilidade de escalabilidade (ex.: Uso de Clusters).

Hardware	**Software**	**PWR - IAEA**		
		FORTRAN90 (s)	C++ (s)	C++/FORTRAN90
i3-3250	Windows 10	—	0.11235	——
	Ubuntu	0.04708	0.08074	1.71495
	Xubuntu	0.04698	0.08007	1.70434
Athlon X4-840	Xubuntu	0.05908	0.08377	1.41791
Celeron N3060	Xubuntu	0.14718	0.27743	1.88497
Raspberry Pi 3B	Raspbian	0.55723	0.70682	1.26845
Raspberry Pi 4B	Raspbian	0.2087	0.2279	1.09199

Tabela 3: Tempo de simulação (segundos), para simular um reator unidimensional da IAEA com 340 malhas e 9 regiões, em diferentes hardwares

3.2 Modelagem do Assembly do reator Argonauta no OPENMC

3.2.1 Descrição do OPENMC

De forma sucinta, o OPENMC é um código computacional baseado no método estatístico de Monte Carlo. Ele permite simular o transporte de nêutrons e fótons, isto é, partículas sem carga. Seu desenvolvimento é aberto a comunidade e, foi originalmente desenvolvido pelo grupo de Física Computacional de Reatores do Massachusetts Institute of Technology no inicio de 2011. Neste código é possível simular termos fontes fixo, problemas de autovalor k, cálculos de fator de multiplicação subcrítico. Como pontos de destaques, o OPENMC permite representação em CAD ou em geometria solida construtiva. O OPENMC nativamente utiliza dados no formato HDF5 que pode ser obtidos da conversão de arquivos ACE produzidos por programas tradicionais na área nuclear, como o NJOY [20].

Os métodos estocásticos, como exemplo, o método de Monte Carlo, utilizado no OpenMC e em códigos como o MCNPX, estão sendo amplamente empregados para simular reatores completos, para obter parâmetros nucleares, como seções de choque multigrupo. Então, esses parâmetros nucleares podem ser utilizados em outros códigos que resolvem a equação de difusão de nêutrons multigrupo por métodos determinísticos. Isso vem ocorrido

com frequência, devido ao fato de as simulações em métodos determinísticos serem muito mais rápido do que as por métodos estocásticos.

3.2.2 A Modelagem da Célula do Argonauta no OPENMC

As seções de choque multigrupo são os dados primordiais para fazer uma simulação de reator nuclear. Esses dados podem ser obtidos por métodos determinísticos ou estocásticos [26], cada um possuindo suas vantagens e desvantagens. Os métodos determinísticos tem como vantagem depender do método empregado e serem rápido para convergir os resultados. No entanto, tem como desvantagem necessitar, a priori, do conhecimento do fluxo de nêutrons multigrupo . Por outro lado, os métodos estocásticos, como exemplo, o método de Monte Carlo, tem como vantagem conseguir trabalhar com especto continuo de energia e não depender do conhecimento prévio do fluxo de nêutrons. Todavia, tem como desvantagem um alto custo computacional, em termos de tempo e armazenamento de dados [20]. Nesse sentido, realizar a simulação de um reator em tempo real por métodos estocásticos é inviável. Porém, pode-se realizar apenas uma simulação com métodos estocásticos para obter a seções de choque multigrupo e, utilizar esses dados em outros códigos. Neste trabalho foi utilizado o OpenMC para obter as seções de choque da célula do Argonauta. Uma descrição detalhada dos cálculos de homogeneização do reator, para obter os dados utilizados no OPENMC, é descrito no apêndice A.3.

Como citado anteriormente na seção 3.2.1, o OPENMC utiliza uma modelagem baseado em geometria sólida construtiva ou orientada por malhas de polígonos. Nesse trabalho, utilizamos a primeira opção, para criar um célula representativa do reator Argonauta e, a partir desta o Assembly completo. Os valores contidos nos fragmentos de código subsequentes foram baseados em cálculos de homogeneização desenvolvidos no apêndice A.3, página 70.

Para obter as seções de choque macroscópicas através do OPENMC, foi necessário definir os materiais utilizados na simulação, isto é, delimitar (através de interseções e uniões) as diferentes regiões do reator ou célula. No caso do reator Argonauta, a célula é constituída de 5 regiões homogêneas, a saber, região 1 (material do miolo, contendo U_3O_8 e Alumínio), região 2 (material do clad, contendo Alumínio), região 3 (material dos canais de água adjacente as lâminas de combustível), região 4 (material em excesso, contendo água e Alumínio) e, região 5 (material refletor, contendo puramente grafite).

Um exemplo de como definir um material (ex.:U_3O_8) no OPENMC é feita no fragmento de código 1

```
fuel_mat = openmc.Material(1,name='20%_fuel_mat')
fuel_mat.set_density('g/cm3', 3.0687)
fuel_mat.add_nuclide('U235', 7.2348e+20, 'ao') # atomos/cm3
fuel_mat.add_nuclide('U238',2.8736e+21,'ao') # atomos/cm3
fuel_mat.add_nuclide('O16',9.5922e+21, 'ao') # atomos/cm3
fuel_mat.add_nuclide('Al27', 3.1486e+22,'ao') # atomos/cm3
fuel_mat.temperature = 294 # Em Kelvin
```

Código 1: Exemplo de como definir materiais no OPENMC

Um exemplo de como é feita a delimitação de regiões no OPENMC é feita nos fragmentos de código 2 e 3.

```
# Criar um dicionario para armazenar as celulas e universos necessarios
    para a simulacao:
celulas = {}
universos = {}
# Box delimitador da celula do Argonauta:
box = openmc.model.rectangular_prism(9.9616, 0.92) # cm
# Interfaces paralelas ao eixo y:
# Interface entre o grafite e o excesso de materiais:
int_esq_G_EM = openmc.XPlane(x0=-1*(6.576+1.0996)/2) # cm
# Interface entre o excesso de materais e a regiao ativa:
int_esq_EM_RA = openmc.XPlane(x0=-6.576/2) # cm
# Interface entre a regiao ativa e o excesso de materiais:
int_dir_RA_EM = openmc.XPlane(x0=6.576/2) # cm
# Interface entre o excesso de materiais e o grafite:
int_dir_EM_G = openmc.XPlane(x0=(6.576+1.0996)/2) # cm
```

Código 2: Delimitação dos planos paralelos ao eixo y

```
# Interface abaixo entre o canal de agua e o revestimento:
int_abaixo_mod_clad = openmc.YPlane(y0=-1*(0.0935+0.028)) # cm
# Interface abaixo entre o revestimento e o combustivel:
int_abaixo_clad_fuel = openmc.YPlane(y0=-0.0935) # cm
# Interface acima entre o combustivel e o revestimento:
int_acima_fuel_clad = openmc.YPlane(y0=0.0935) # cm
# Interface acima entre o revestimento e o canal de agua:
int_acima_clad_mod = openmc.YPlane(y0=(0.0935+0.028)) # cm
```

Código 3: Delimitação dos planos paralelos ao eixo X

Então, com os materiais e regiões definidos, pode-se atribuir os materiais as suas regiões correspondentes. Essa etapa é realizada conforme o exemplo do fragmento de código 4.

```
1    celulas['fuel_cell'] = openmc.Cell(name='fuel_cell')
2    celulas['fuel_cell'].fill = fuel_mat
3    celulas['fuel_cell'].region = \
4    +int_abaixo_clad_fuel & -int_acima_fuel_clad & \
5    +int_esq_EM_RA & -int_dir_RA_EM
```

Código 4: Exemplo de como atribuir um material à célula criada no OPENMC

Com as células prontas, é necessário criar uma malha para construir um universo contemplando todo o assemply do reator Argonauta. Essa etapa é feita como exemplifica o fragmento de código 5,

```
1    # Criar um dicionario de lattices:
2    lattices = {}
3    # Criar um registro do assembly Original:
4    lattices['Assembly original'] = openmc.RectLattice(name='Assembly
         original')
5    lattices['Assembly original'].dimension = [165, 14] # admensional
6    lattices['Assembly original'].lower_left = [-69.7312, -75.9] # cm
7    lattices['Assembly original'].pitch = [9.9616, 0.92] # cm
8    u = universos['celula_antiga']
9    #fn = universos['celula_fuelnat']
10   g = universos['celula_grafite']
11   cr = universos['celula_control_rod']
12   gt = universos['celula_guia'] # gt -> guide tube
13   v = universos['celula_vazia']

15   padrao_grafite = [g, g, g, g, g, g, g, g, g, g, g, g, g, g, g]
16   padrao_assembly = [g, g, g, gt, u, u, u, u, u, u, gt, g, g, g]
17   padrao_cadmio = [g, g, g, g, cr, cr, cr, cr, cr, cr, g, g, g, g]
18   padrao_vazio = [g, g, g, g, v, v, v, v, v, v, g, g, g, g]

20   lattices['Assembly original'].universes = \
21   [padrao_grafite, padrao_grafite, padrao_grafite, padrao_grafite,
         padrao_grafite, \
22   padrao_grafite, padrao_grafite, padrao_grafite, padrao_grafite,
         padrao_grafite, \
23   padrao_grafite, padrao_grafite, padrao_grafite, padrao_grafite,
         padrao_grafite, \
24   padrao_vazio, padrao_vazio, padrao_vazio, padrao_vazio, padrao_cadmio, \
25   padrao_cadmio, padrao_cadmio, padrao_cadmio, \
26   padrao_cadmio, padrao_cadmio, padrao_cadmio, padrao_cadmio,
         padrao_cadmio, \
```

```
27    padrao_cadmio , padrao_cadmio , padrao_cadmio , padrao_cadmio ,
          padrao_cadmio , \
28    padrao_assembly , padrao_assembly , padrao_assembly , padrao_assembly ,
          padrao_assembly , \
29    padrao_assembly , padrao_assembly , padrao_assembly , padrao_assembly ,
          padrao_assembly , \
30    padrao_assembly , padrao_assembly , padrao_assembly , padrao_assembly ,
          padrao_assembly , \
31    padrao_assembly , padrao_assembly , \
32    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
33    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
34    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
35    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
36    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
37    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
38    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
39    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
40    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
41    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
42    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
43    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
44    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
45    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
46    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
47    padrao_grafite , padrao_grafite , padrao_grafite , padrao_grafite ,
          padrao_grafite ,
48    padrao_grafite , padrao_grafite , \
49    padrao_cadmio , padrao_cadmio , padrao_cadmio , padrao_cadmio ,
          padrao_cadmio , \
50    padrao_cadmio , padrao_cadmio , padrao_cadmio , padrao_cadmio ,
          padrao_cadmio , \
51    padrao_cadmio , padrao_cadmio , padrao_cadmio , \
```

```
52      padrao_cadmio, padrao_vazio, padrao_vazio, padrao_vazio, padrao_vazio, \
53      padrao_grafite, padrao_grafite, padrao_grafite, padrao_grafite,
            padrao_grafite,
54      padrao_grafite, padrao_grafite, padrao_grafite, padrao_grafite,
            padrao_grafite,
55      padrao_grafite, padrao_grafite, padrao_grafite, padrao_grafite,
            padrao_grafite]
```

Código 5: Padrão de Universos simulados no OPENMC

É necessário também delimitar o universo do assemply completo, conforme mostra o fragmento abaixo,

```
1   # Planos delimitadores do Assembly:
2   x_min = openmc.XPlane(x0=-69.7312, boundary_type='vacuum') # cm
3   x_max = openmc.XPlane(x0=69.7312,  boundary_type='vacuum') # cm
4   y_min = openmc.YPlane(y0=-75.9, boundary_type='reflective') # cm
5   y_max = openmc.YPlane(y0=75.9,  boundary_type='reflective') # cm
6   # Criar a celula raiz
7   root_cell = openmc.Cell(name='root cell')
8   root_cell.fill = lattices['Assembly original']
9   root_cell.region = +x_min & -x_max & +y_min & -y_max
10  # Criar o universo raiz
11  root_universe = openmc.Universe(name='root universe', universe_id=0)
12  root_universe.add_cell(root_cell)
```

Código 6: Configurações do limites do reator e as condigões de contorno

Tendo os universos criados e verificados, é necessário ajustar as configurações de simulação. Neste trabalho foram utilizados as configurações contidas no dois próximos fragmento de código,7 e 8.

```
1   batches = 50; inactive = 10; particles = 100000 # admensional
2   settings = openmc.Settings()
3   settings.batches = batches
4   settings.inactive = inactive
5   settings.particles = particles
6   settings.output = {'tallies': True}
7   # Criar uma fonte uniforme de neutrons:
8   bounds = [-69.7312, -75.9, -30.1, 69.7312, 75.9, 30.1] # cm
9   uniform_dist = openmc.stats.Box(bounds[:3], bounds[3:], only_fissionable
        =True)
10  settings.source = openmc.Source(space=uniform_dist)
11  settings.trigger_active = True
```

```
12    settings.trigger_max_batches = settings.batches * 4
13    settings.export_to_xml()
```

Código 7: Parâmetros de simulação no OPENMC

```
1     # Definindo os limites dos grupos de energia:
2     groups = mgxs.EnergyGroups([0., 0.625, 20.0e6]) # eV

4     tallies = openmc.Tallies()

6     # Limites globais do assembly:
7     x_min = -69.7312; x_max = 69.7312 # cm
8     y_min = -75.9; y_max = 75.9 # cm

10    # Filtro de energia que eh utilizado em todas as malhas:
11    filtro_energia = openmc.EnergyFilter([0., 0.625, 20.0e6]) # eV

13    # Regiao de grafite
14    malha_grafite_externo = openmc.RegularMesh()
15    malha_grafite_externo.dimension = [1,1] # admensional
16    malha_grafite_externo.lower_left = [-69.7312, -75.9] # cm
17    malha_grafite_externo.upper_right = [-29.5848, 75.9] # cm

19    filtro_malha1 = openmc.MeshFilter(malha_grafite_externo)

21    tally1 = openmc.Tally(name='registro_grafite_externo')
22    tally1.filters = [filtro_malha1, filtro_energia]
23    tally1.scores = ['flux', 'nu-fission', 'fission', 'absorption', 'scatter
          ', 'total']
24    tallies.append(tally1)
25    SigmaTr_grafite_externo = mgxs.TransportXS(domain_type='mesh', domain=
          malha_grafite_externo, groups=groups)
26    tallies += SigmaTr_grafite_externo.tallies.values()
27    # Regiao de Grafite interno:
28    malha_grafite_interno = openmc.RegularMesh()
29    malha_grafite_interno.dimension = [1,1]
30    malha_grafite_interno.lower_left = [-29.5848, -75.9] # cm
31    malha_grafite_interno.upper_right = [-28.4418 , 75.9] # cm
32    filtro_malha2 = openmc.MeshFilter(malha_grafite_interno)
33    tally2 = openmc.Tally(name='registro_grafite_interno')
34    tally2.filters = [filtro_malha2, filtro_energia]
35    tally2.scores = ['flux', 'nu-fission', 'fission', 'absorption', 'scatter
          ', 'total']
36    tallies.append(tally2)
37    SigmaTr_grafite_interno = mgxs.TransportXS(domain_type='mesh', domain=
          malha_grafite_interno, groups=groups)
```

```
38      tallies += SigmaTr_grafite_interno.tallies.values()
39      # Regiao de excesso de materiais:
40      malha_excesso_materiais = openmc.RegularMesh()
41      malha_excesso_materiais.dimension = [1,1] # admensional
42      malha_excesso_materiais.lower_left = [-28.4418, -75.9] # cm
43      malha_excesso_materiais.upper_right = [-27.892, 75.9] # cm
44      filtro_malha3 = openmc.MeshFilter(malha_excesso_materiais)
45      tally3 = openmc.Tally(name='registro_excesso')
46      tally3.filters = [filtro_malha3, filtro_energia]
47      tally3.scores = ['flux', 'nu-fission', 'fission', 'absorption', 'scatter
            ', 'total']
48      tallies.append(tally3)
49      SigmaTr_excesso = mgxs.TransportXS(domain_type='mesh', domain=
            malha_excesso_materiais, groups=groups)
50      tallies += SigmaTr_excesso.tallies.values()
51      # Regiao de cerne ativo:
52      malha_ativa = openmc.RegularMesh()
53      malha_ativa.dimension = [1,1] # admensional
54      malha_ativa.lower_left = [-27.892, -75.9] # cm
55      malha_ativa.upper_right = [-21.316, 75.9] # cm
56      filtro_malha4 = openmc.MeshFilter(malha_ativa)
57      tally4 = openmc.Tally(name='registro_cerne_ativo')
58      tally4.filters = [filtro_malha4, filtro_energia]
59      tally4.scores = ['flux', 'nu-fission', 'fission', 'absorption', 'scatter
            ', 'total']
60      tallies.append(tally4)
61      SigmaTr_cerne = mgxs.TransportXS(domain=malha_ativa, domain_type='mesh',
             groups=groups)
62      tallies += SigmaTr_cerne.tallies.values()
63      chi_cerne = mgxs.Chi(domain_type='mesh', domain=malha_ativa, groups=
            groups)
64      tallies += chi_cerne.tallies.values()
65      tallies.export_to_xml()
```

Código 8: Configurações de filtro de energia e malha, considerando diferentes tipos de reação

Neste ponto, já é possível realizar a simulação do reator completo no OPENMC, bastando executar o comando run, conforme ilustrado no fragmento a seguir,

```
1   !rm 'summary.h5' # Para evitar erros na execucao.
2   !rm 'statepoint.*'
3   openmc.run()
```

Código 9: Comando para executar a simulação no OPENMC

Ao final da simulação, o OPENMC retorna dados básicos como k_{eff}, fração de fuga de nêutrons, bem como, os erros associados a esses valores. No entanto, como foram definidos filtros de energia e de malhas, é possível recuperar maiores informações como seções de choque em cada região do reator, a fim de ser utilizado em outro programa de simulação de reator, como o DIF1D (código em FORTRAN para simular o fluxo de nêutrons em um reator unidimensional). Isso foi obtido através dos seguintes comandos,

```
1  sp = openmc.StatePoint('statepoint.050.h5')
2  registro_grafite_externo = sp.get_tally(name='registro_grafite_externo')
3  fluxo_grafite_externo = registro_grafite_externo.get_slice(scores=['flux
       '])
4  fluxo_grafite_externo.get_pandas_dataframe()
5  absorcao_grafite_externo = registro_grafite_externo.get_slice(scores=['
       absorption'])
6  SigmaAbsorcao_grafite_externo = absorcao_grafite_externo /
       fluxo_grafite_externo
7  SigmaAbsorcao_grafite_externo.get_pandas_dataframe()
8  scatter_grafite_externo = registro_grafite_externo.get_slice(scores=['
       scatter'])
9  SigmaScatter_grafite_externo = scatter_grafite_externo /
       fluxo_grafite_externo
10 SigmaScatter_grafite_externo.get_pandas_dataframe()
11 total_grafite_externo = registro_grafite_externo.get_slice(scores=['
       total'])
12 SigmaTotal_grafite_externo = total_grafite_externo /
       fluxo_grafite_externo
13 SigmaTotal_grafite_externo.get_pandas_dataframe()
14 SigmaTr_grafite_externo = (total_grafite_externo -
       scatter_grafite_externo) / fluxo_grafite_externo
15 SigmaTr_grafite_externo.get_pandas_dataframe()
```

Código 10: Comandos para obter as seções de choque multigrupo no OPENMC

4 DESCRIÇÃO DOS MÓDULOS DO SISTEMA

Neste capítulo é feito uma descrição de cada modulo que integra o conjunto de códigos para coletar dados de voltagens, simular o fluxo de nêutrons em dois grupos e a depleção isotópica no tempo.

A integração entre os códigos ocorre através do agendador de serviços/tarefas do sistema operacional LINUX, o CRON, que invoca os códigos em uma sequencia definida pelo usuário do sistema, desta forma, as sub-rotinas são chamadas em uma sequencia especifica e, a troca de dados entre algoritmos de diferentes linguagens é realizada por arquivo de dados com a extensão .dat. A figura 10 é um flowchart que visa facilitar o entendimento do fluxo do sistema. Todos os módulos contidos na figura 10 são executados em um Raspberry Pi. A descrição de cada etapa contida no Flowchart é feita nas próximas seções.

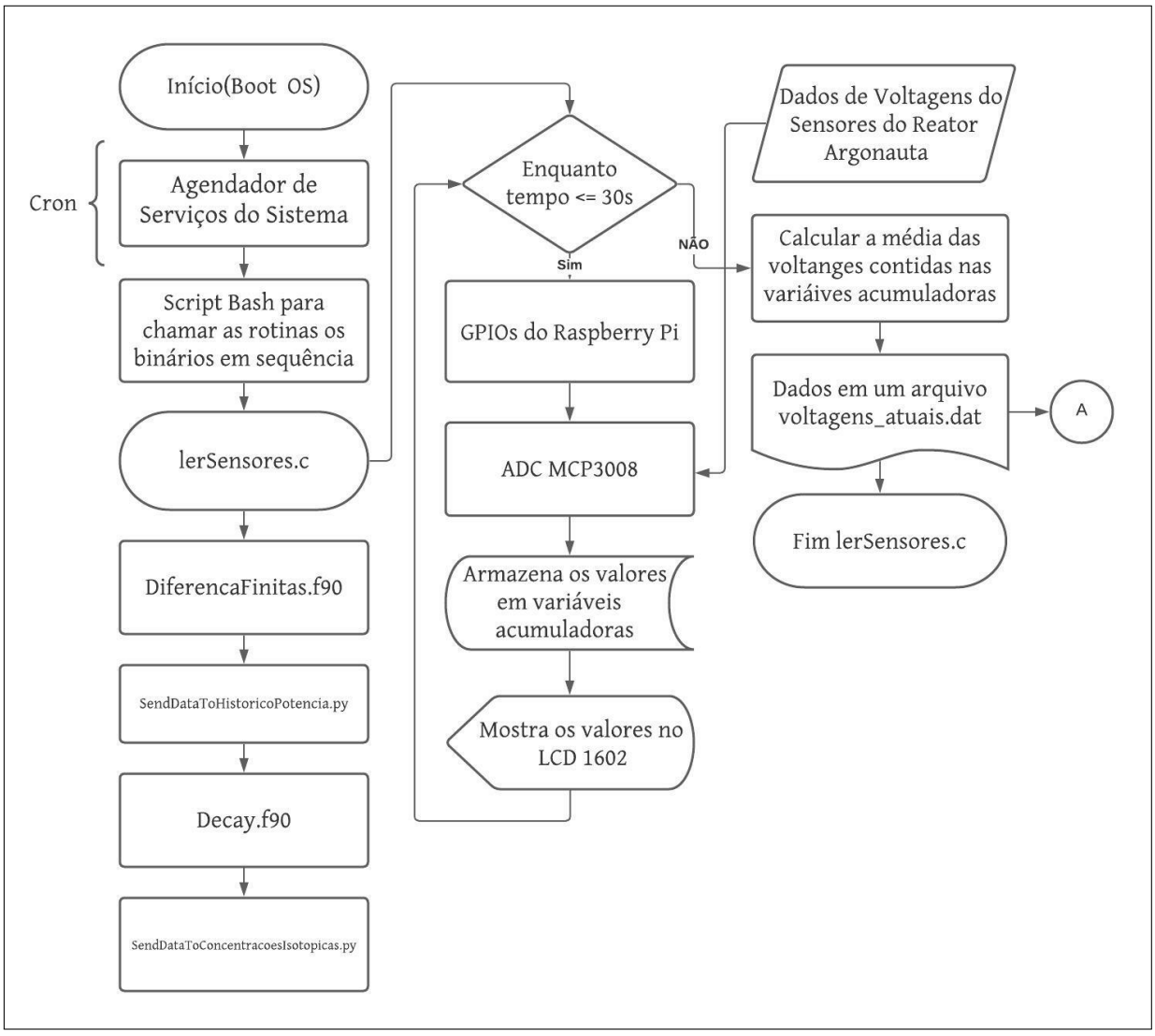

Figura 10: Flowchart do programa de simulação simplificada do reator Argonauta unidimensional

4.1 O código lerSensores.c

O código lerSensores.c, é uma sub-rotina que tem como função principal permitir a leitura de dados de tensão dos sensores analógicos do reator Argonauta. Devido a dificuldade imposta pelo afastamento social, devido a pandemia, optou-se por utilizar de forma representativa dois potenciômetros, que simbolicamente representam os dados aferidos pelos sensores linear e logaritmo do reator Argonauta. Desta forma, mesmo longe do reator é possível avaliar o funcionamento do sistema de captura de dados.

De forma sucinta, o código fonte do algoritmo, consiste em um loop de 30 segundos, com delay de 200ms que, fica monitorando os dados de tensão. A partir desses, é calculado a média, sendo essa informação colocada em um arquivo de texto com extensão .dat, para ser utilizado na rotina de decaimento para estimativa da potência total e em cada Assembly, pois há uma diferença na intensidade da depleção em virtude do fluxo presente em cada região do reator.

4.2 O Código DIF1D

O Dif1D é um conjunto de códigos fontes, escritos na linguagem de programação FORTRAN 90, que tem como objetivo principal, simula um reator unidimensional em geometria cartesiana, obtendo como resultado, o Keff do reator, o fluxo de nêutrons e a distribuição de potência ao longo da extensão do reator.

O Dif1D é baseado no tradicional e, amplamente utilizado método das diferenças finitas centrado nas malhas, no algoritmo de Thomas e no método das potências. O DIF1D é dividido em vários módulos, que tem como finalidade, deixar o código mais fácil de manter e entender, permitindo que cada algoritmo realize uma pequena tarefa especifica. Basicamente o código DIF1D é constituído de um programa principal, em FORTRAN 90, denominado DiferencaFinitas.f90, que invoca as demais sub-rotinas em uma sequência. As subseções a seguir descrevem cada um desses algoritmos com um pouco mais de detalhe.

4.2.1 O Algoritmo LeituraDadosEntrada.f90

Este algoritmo é a primeira sub-rotina a ser invocada pelo código DIF1D. Em suma, neste algoritmo é feita a leitura dos parâmetros geométricos e nucleares do reator a ser simulado. No caso deste trabalho, o algoritmo lerá o arquivo Argonauta.dat, que contem informações como número de tipos de região, número de regiões, dimensão do núcleo, bem como, os

parâmetros nucleares multigrupos associados a cada tipo de região, isto é, coeficiente de difusão, espectro de fissão, seções de choque macroscópica de absorção, espalhamento e fissão, obtidos previamente via simulação da geometria do reator no OPENMC.

4.2.2 O Algoritmo Inicialização.f90

Apos finalizar o algoritmo LeituraDadosEntrada, a sub-rotina Inicialização é chamada. Este código tem por finalidade inicializar os valores contido em arrays e em variáveis, tais como $K_{eff} = 1.0$ e somada fonte fissão (vetor inicialmente preenchido com 0.0).

4.2.3 O Algoritmo CalculaMatrizesDiscretizacao.f90

Após o algoritmo de Inicialização finalizar, o código CalculaMatrizesDiscretizacao é chamado. Nesta rotina, as diagonais da matriz para estimar o fluxo de nêutrons é preenchido, baseado na discretização da Equação da Difusão de Nêutrons em dois grupos de energia. Além disso, ainda na rotina é atribuído as condições de contorno a esquerda e a direita, considerando a primeira e ultima malha do reator, bem como são preenchidas a matriz de fissão e espalhamento.

4.2.4 O Algoritmo IteracoesExternas.f90

Este algoritmo tem por finalidade estimar o fluxo médio em cada malha do reator e o respectivo k_{eff} quando as condições de contorno são atendidas. Por ser um código iterativo, demanda um certo tempo de processamento. Vale destacar que mais de 80% do tempo de processamento do Dif1D está associado a esse algoritmo. De forma sucinta, o algoritmo consiste em um loop iterativo que fica estimando a fonte de fissão e o fluxo médio até que um dos critérios de convergência sejam atendidos ou que, o número máximo de iterações previamente estipulado seja alcançado.

4.2.5 O Algoritmo CalculaPotencia.f90

Este é o ultimo Algoritmo do conjunto de códigos do DIF1D. O código tem por finalidade estimar a potência em cada região do reator e, informar um perfil do fluxo de nêutrons multigrupo ao longo das malhas.

4.2.6 O Algoritmo SendDataToHistoricoPotencia.py

Este código foi escrito em Python e, é executado após a simulação do código Dif1D. Em suma, ele recupera os dados contidos nos arquivos voltagens_atuais.dat (obtido com lerSensores.c) e Fluxo_Potencia_Atuais.dat e, envia essas informações juntamente com data e hora, para um banco de dados denominado HistoricoPotenciaArgonauta.db (criado em SQLite3). O código fonte do algoritmo pode ser visto no apêndice B.3, na página 86.

4.2.7 O Algoritmo Decay.f90

Este algoritmo é invocado apos a sub-rotina SendDataToHisoticoPotencia ser finalizado. O objetivo deste algoritmo é estimar o decaimento dos actinídeos e produtos de fissão, para realizar essa tarefa, o código resolve a matriz exponencial, que representa a evolução do sistema em cada um dos 6 Assemblys do reator. O algoritmo deste código foi desenvolvido no IEN e, resolve a matriz exponencial do sistema de equações de depleção dos Actinídeos e produtos de fissão. Os dados de seções de choque microscópicas multigrupo utilizadas neste código foram obtidos via simulação no OPENMC.

4.2.8 O Algoritmo SendDataToConcentracoesIsotipicas.py

Este é um código em Python que, tem por finalidade enviar dados das concentrações isotópicas (Actinídeos e produtos de fissão) estimadas pela sub-rotina anterior (Decay.f90) para o banco de dados CocentracoesIsotopicaArgonauta.db (em SQLite3). O código fonte pode ser visto no apêndice B.4, página 88.

5 RESULTADOS

Neste capítulo é apresentado os resultados obtidos no decorrer do desenvolvimento do trabalho, quanto ao estudo da viabilidade técnica de usar o Raspberry Pi em simulações na área nuclear, quanto ao funcionamento da parte eletrônica feita para capturar os dados de tensão do reator Argonauta. Além disso, o capítulo ainda traz os resultados de parâmetros nucleares importantes, tais como, k_{eff} e seções choque macroscópicas para as diferentes regiões do reator Argonauta, obtidos via simulação no OPENMC e no DIF1D.

5.1 Da viabilidade técnica

Um estudo preliminar sobre a possibilidade de realizar simulação de um reator unidimensional no Raspberry Pi, foi feito considerando um reator da IAEA e o Argonauta. Em ambos os casos, o algoritmo DIF1D terminou a tarefa de simular o perfil do fluxo de neutros com dois grupos, considerando uma estrutura de malha fina(ex.: 340 e 1000 malhas) em menos de 0.3 segundos. É importante frisar que, a simulação do reator Argonauta foi muito rápido (0.118 s), quando comparada a simulação do reator da IAEA, exigindo apenas 16 iterações, em contraposição as 885, requeridas pelo reator PWR de referência.

O esquema eletrônico (página 79) para coletar dados de tensão dos sensores do reator Argonauta foi testado com uso de 2 potenciômetros. Durante os testes não houve percepção de delay e, a chamada do algoritmo lerSensores.c foi invocada corretamente pelo CRON (agendador de tarefas do LINUX).

5.2 K_{eff} e parâmetros nucleares via OPENMC

O reator Argonauta foi modelado e simulado no OPENMC para obter o K_{eff} e as seções de choque macroscópicas da diferentes regiões do reator Argonauta. O tempo médio das simulações ficou em torno de 30/35 minutos. O K_{eff} obtido foi igual a 1.02162. Enquanto os parâmetros nucleares de seções de choque das diferentes regiões do reator obtidos podem ser vistos nas tabelas de 4 a 7.

Região 1	g1(0.625 - 2.0E+7)eV	g2(0.0 - 0.625)eV
$\Sigma_a(cm^{-1})$	0.003129	0.012048
$\nu\Sigma_f(cm^{-1})$	0.001237	0.017442
$\Sigma_s(cm^{-1})$	0.456430	0.652120
Coef. Difusão(cm)	0.725333	1.703374

Tabela 4: Parâmetros nucleares da Região 1(cerne ativo) via OPENMC, contendo U_3O_8, água e Alumínio

Região 2	g1(0.625 - 2.0E+7)eV	g2(0.0 - 0.625)eV
$\Sigma_a(cm^{-1})$	0.003030	0.009735
$\nu\Sigma_f(cm^{-1})$	0.000000	0.000000
$\Sigma_s(cm^{-1})$	0.471275	0.625787
Coef. Difusão(cm)	0.702782	2.029451

Tabela 5: Parâmetros nucleares da Região 2(excesso de materiais) via OPENMC, contendo Alumínio e água

Região 3	g1(0.625 - 2.0E+7)eV	g2(0.0 - 0.625)eV
$\Sigma_a(cm^{-1})$	0.002554	0.002946
$\nu\Sigma_f(cm^{-1})$	0.000000	0.000000
$\Sigma_s(cm^{-1})$	0.360920	0.443825
Coef. Difusão(cm)	0.917073	3.882695

Tabela 6: Parâmetros nucleares da Região 3(refletora) via OPENMC, contendo grafite

Região 4	g1(0.625 - 2.0E+7)eV	g2(0.0 - 0.625)eV
$\Sigma_a(cm^{-1})$	0.000091	0.000643
$\nu\Sigma_f(cm^{-1})$	0.000000	0.000000
$\Sigma_s(cm^{-1})$	0.345679	0.395153
Coef. Difusão(cm)	0.964031	6.737545

Tabela 7: Parâmetros nucleares da Região 4(externa aos assemblys) via OPENMC, contendo grafite

5.3 K_{eff} e potência via DIF1D

Os dados contidos nas tabelas de 4 a 7 foram utilizados como parâmetros de entrada para o algoritmo DIF1D, a fim de simular o reator Argonauta de forma unidimensional. A simulação do reator, considerando 32 regiões e, um total de 1000 malhas, foi finalizada em 0.118 s.

O K_{eff} estimado pelo DIF1D foi de 1.09940. Enquanto os valores de potência para cada um dos 6 Assembly do Argonauta, estão contidos na tabela 8.

O perfil do fluxo de nêutrons térmicos de rápidos obtidos via simulação no DIF1D, pode ser vistos nas figuras 11 e 12.

Nº Elemento Combustível	Potência (Watts)
1	78.24
2	84.32
3	87.43
4	87.43
5	84.32
6	78.24

Tabela 8: Perfil de potência dos 6 elementos combustíveis do reator Argonauta, considerando operação em 500W.

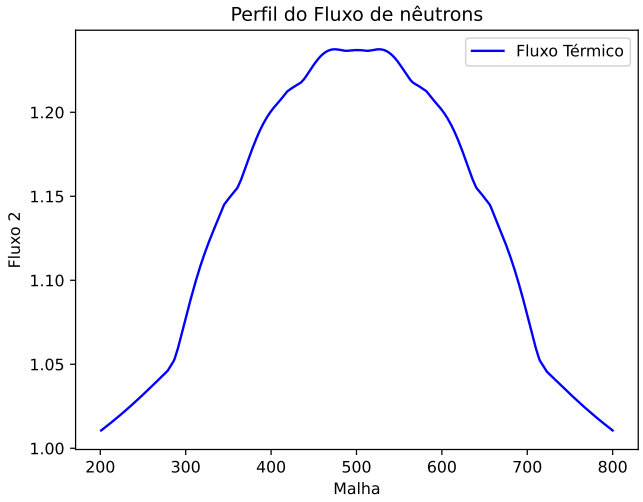

Figura 11: Perfil de Fluxo de Nêutrons Térmicos via DIF1D

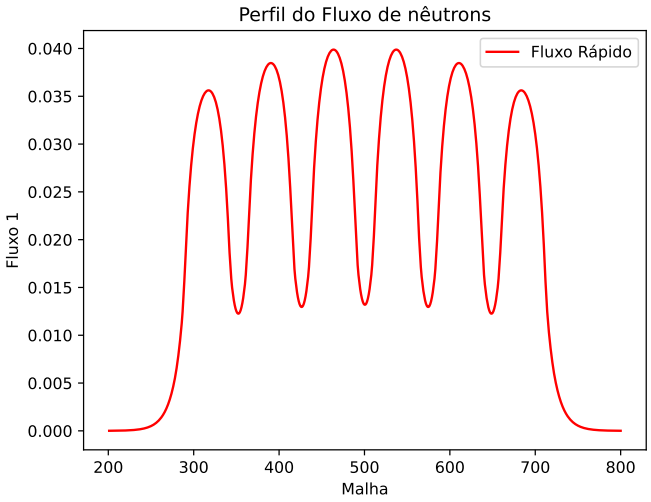

Figura 12: Perfil de Fluxo de Nêutrons Rápido via DIF1D

5.4 Inventário isotópico e dados de atividade

A simulação da depleção do reator Argonauta foi simulada pelo algoritmo Decay.f90 (usa o algoritmo **expm** desenvolvido no IEN), considerando 6 Assemblys. O tempo para realizar a simulação, foi de 0.14 s, em um Raspberry Pi 3B.

Por outro lado, o inventário isotópico do reator Argonauta até 2018, foi estimado em [4] via simulação, considerando 9781 horas de operação a 170W e, 6504 horas de operação a 340W. Os dados podem ser vistos nas tabelas de 9 a 13.

Nuclídeo	Massa(g)	Nuclídeo	Massa(g)
U^{235}	2.090e+03	Np^{240}	8.262e-16
U^{236}	8.742e-04	Pu^{238}	1.651e-18
U^{237}	1.531e-12	Pu^{239}	9.228e-05
U^{238}	8.409e+03	Pu^{240}	5.174e-11
U^{239}	5.869e-08	Pu^{241}	2.056e-17
Np^{237}	7.867e-12	Pu^{242}	7.703e-24
Np^{238}	1.558e-19	Am^{241}	7.458e-20
Np^{239}	2.959e-06	Cm^{242}	0.000e+00

Tabela 9: Inventário de Actinídeos no núcleo do reator Argonauta até 2018, fonte: [4]

Nuclídeo	Massa(g)	Atividade(Bq)
I^{130}	1.5562e-06	1.1230e+11
I^{131}	2.4315e-05	1.1174e+11
I^{132}	2.8998e-07	1.1099e+11
I^{133}	2.6579e-06	1.1124e+11
I^{134}	1.1476e-07	1.1349e+11
I^{135}	1.0213e-06	1.3331e+11
I^{136}	3.5978e-09	1.3240e+11

Tabela 10: Inventário de isotopos de Iodo no núcleo do reator Argonauta até 2018, fonte: [4]

Nuclídeo	Massa(g)	Atividade(Bq)
Kr_m^{85}	2.162e-08	6.584e+09
Kr^{85}	7.047e-06	4.240e-06
Br^{86}	7.908e-11	2.684e+10
Te^{132}	3.525e-06	4.032e+10
Te^{133}	9.842e-09	4.118e+10
Te^{134}	6.053e-08	7.518e+10
Xe^{133}	3.619e-07	2.505e+09
Xe_m^{135}	2.540e-05	1.749e+09

Tabela 11: Inventário de produtos de fissão voláteis no núcleo do reator Argonauta até 2018, fonte: [4]

Nuclídeo	Massa(g)	Atividade(Bq)
Sr^{91}	1.780e-08	2.350e+09
Sr^{92}	2.335e-08	1.106e+10
Y^{90}	6.586e-07	1.324e+10
Y^{91}	8.194e-06	7.435e+09
Y^{92}	2.231e-08	7.943e+09
Y^{93}	7.274e-08	8.909e+09
Zr^{95}	5.007e-06	3.977e+09
Nb^{95}	5.470e-06	7.590e+09
Nb^{96}	1.509e-07	7.805e+09
Mo^{99}	2.174e-08	3.859e+08
Pr^{144}	4.360e-15	3.158e+06
Ru^{106}	8.914e-08	1.092e+07
Cs^{134}	7.197e-11	1.495e+09
Cs^{136}	1.317e-08	1.302e+07
Cs^{138}	7.020e-10	9.402e+11
Ce^{141}	2.532e-11	2.204e+08

Tabela 12: Inventório de produtos de fissão não voláteis no núcleo do reator Argonauta até 2018, fonte: [4]

6 CONCLUSÕES E PERSPECTIVAS FUTURAS

6.1 Considerações gerais

Os resultados de tempo de processamento, para simular o fluxo de nêutrons unidimensional (0.11s) e, a depleção isotópica (0.14s), apresentados neste trabalho, demonstram a viabilidade do uso do Raspberry Pi em simulações ligados a área nuclear.

Uma interface eletrônica foi desenvolvida para solucionar o problema de obter dados de corrente dos sensores do reator Argonauta, a fim desses serem utilizados para estimar o perfil de potencia nos diferentes Assembly existentes. Além disso, foram desenvolvidos algoritmos para envio dos dados (corrente, potência e concentrações isotópicas) para um banco de dados, a fim de no futuro essas informações poderem serem recuperadas para uso em algum projeto.

Cálculos de homogeneização do Assembly completo do reator foram feitos, para a construção e simulação de um modelo bidimensional muito simplificado (geometria cartesiana e sem retroalimentação termo-hidráulica) do reator Argonauta, utilizando o OPENMC.

As seções de choque macroscópicas das diferentes regiões do reator Argonauta, foram obtidas via simulação em OPENMC, sendo os dados utilizados para alimentar outro código, o DIF1D, no intuito de se obter o perfil de potência e fluxo de nêutrons. Nesse aspecto, o estudo desenvolvido neste trabalho evidencia um potencial uso do OPENMC em obter parâmetros nucleares para serem utilizados em outros projetos (algoritmos de simulação). Nesse sentido, é possível inferir que, o OPENMC mostra-se uma ferramenta útil para modelagem e simulações de reatores, sendo alternativa ao uso do MCNPX.

Os valores de K_{eff} obtidos no OPENMC e via DIF1D mostram-se suavemente afastados (devido a base de dados diferente). No entanto, sugerindo que, o modelo simplificado do reator Argonauta (geometria cartesiana), precisa ser aperfeiçoado, necessitando portanto, talvez fazer um novo modelo, isto é, na geometria original do reator (geometria cilíndrica 2D ou 3D), levando-se em consideração ainda, a parte termo-hidráulica e, talvez fazendo uso de discretização por elementos finitos.

Os dados apresentados de inventário isotópico de Actinídeos e produtos de fissão, demonstram que, existe ainda grande quantidade de massa de U^{235} e U^{238} no núcleo do Argonauta. Por outro lado, a atividade de alguns nuclídeos, como exemplo, os isotopos de Iodo e de Cs^{138} sugerem que, um monitoramento de radionuclídeos se faz necessário para avaliação de dose em um acidente postulado, bem como, para um futuro armazenamento adequado dos resíduos nucleares, decorrentes da substituição de elementos combustíveis no núcleo do reator.

6.2 Considerações finais

Os objetivos inicialmente propostos para o trabalho foram alcançados. Um sistema computacional para coletar e processar dados do reator Argonauta foi obtido, para em ultima instancia, avaliar o inventário isotópico, potências e fluxo de nêutrons nos 6 Assemblys presentes no núcleo. No entanto, até a entrega deste trabalho, por motivos de força maior e de tempo, não foi possível testar presencialmente o funcionamento da implementação, isto é, os algoritmos desenvolvidos ou utilizados neste trabalho, foram testados de forma remota e isolada do reator Argonauta, através do uso de potenciômetros.

6.3 Sugestões de trabalhos futuros

A partir do que foi descrito neste trabalho, é possível recomendar o seguinte,

- Continuar o projeto no que tange a parte de testar presencialmente a implementação desenvolvida, de forma a validar todo o trabalho aqui apresentado;
- Aperfeiçoar o modelo do reator Argonauta no OPENMC, para obter as seções de choque em uma malha bidimensional;
- Implementar um algoritmo de simulação de fluxo de nêutrons que leve em consideração a parte termo-hidráulica do reator, considerando ainda um modelo em geometria cilíndrica em 2D ou 3D;
- Utilizar o OPENMC na simulação de depleção isotópica, a fim de servir como parâmetro de comparação de trabalhos futuros;
- Desenvolver se possível, uma interface com dashboard em PHP ou Python que, recupere as informações do banco de dados, referentes as simulações (K_{eff}, potência, concentrações isotópicas), disponibilizando esses dados de forma remota (através da internet).

Referências

[1] HIGGINBOTHAM, A. **Midnight in Chernobyl: The Untold Story of the World's Greatest Nuclear Disaster**. New York: Simon & Schuster, 2019.

[2] STEINHAUSER, G.; BRANDL, A.; JOHNSON, T. E. Comparison of the Chernobyl and Fukushima accidents: A review of the environmental impacts. **Science of the total environment**, United States. v. 470, p. 800-817, 2014.

[3] HIRSCH, H et al. Perigos dos reatores nucleares: Riscos na operação da tecnologia nuclear no século XXI. **Estudos Avançados**, São Paulo, v. 21, n. 59, p. 253-257 , 2007.

[4] ALVES, A. M. S.; HEIMLICH, A.; LAMEGO, F.; LAPA, C. M. F. The Inventory and Source Term Simulation of the Argonaut Nuclear Reactor Inside a Severe Accident. **Nuclear Technology**, United States. n. 2, v. 207, p. 316-322, 2020.

[5] LAMEGO, F.; ALVES, A. M.; HEIMLICH, A.; LAPA, C. M. F. The Inventory Simulation of the Argonaut Nuclear Core Reactor. **Instituto de Engenharia Nuclear: Progress Report**, Rio de Janeiro. n. 4, 2021.

[6] MADLENER, R.; SUNAK, Y. Impacts of urbanization on urban structures and energy demand: What can we learn for urban energy planning and urbanization management?. **Sustainable Cities and Society**, Germany. n. 1, v. 1, p. 45-53, 2011

[7] PRAVALIE, R.; BANDOC, G. Nuclear energy: Between global electricity demand, worldwide decarbonisation imperativeness, and planetary environmental implications. **Journal of Environmental Management**, Bucharest. v. 209, p. 81-92, mar. 2018.

[8] CARNEIRO, J. R. N. **Estudo Numérico da Solução das Equações da Cinética Espacial Para Transientes em Reatores Subcríticos (ADS) Multidimensionais**. 2020. 74 f. Dissertação (Mestrado em Engenharia Nuclear) - Instituto de Engenharia Nuclear, Rio de Janeiro, 2020.

[9] ALMEIDA, A. A. H., **Solução das Equações Isotópica do Combustível Nuclear por Computação de Alto Desempenho**. 2016. 144 f. Tese (Doutorado em Engenharia Nuclear) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2016.

[10] PRATA, F. S. **Solução das Equações de Depleção Isotópica Usando o Método da Decomposição e Soluções Analíticas**. 2011. 86 f. Dissertação (Mestrado em Engenharia Nuclear) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2011.

[11] MENEZES, H. P. **Cálculo de Depleção Isotópica em um Reator Nuclear Unidimensional para Comparativo de Performance entre as Linguagens de Programaçãao LUA e FORTRAN**. 2019. 57 f. Monografia (Graduação em Engenharia Nuclear) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2019.

[12] ARMSTRONG, R. H.; KOLB, W. L.; LENNOX, D. H. Argonaut: Engeneering Construction and Costs. Chicago: Argonne National Laboratory; mar. 1957. ANL-5704.

[13] AGHINA, L. Cadernos do Reator Argonauta: Célula do Reator Argonauta. Rio de Janeiro: Instituto de Engenharia Nuclear; 20 nov. 2003. CRA007-R1 (Publicação interna)

[14] HIERNAUT, J. P.; WISS, T.; PAPAIOANNOU, D.; KONINGS, R. J. M.; RONDINELLA, V. V. Volatile fission product behaviour during thermal annealing of irradiated UO2 fuel oxidised up to U3O8. **Journal of Nuclear Materials**, Germany. n. 2-3, v. 372, p. 215-225, mar 2007.

[15] UPTON, E.; HALFACREE, G. **Raspberry Pi user guide**. United Kingdom: John Wiley & Sons, 2014.

[16] MONK, S. **Raspberry Pi Cookbook: Software and Hardware Problems & Solutions**. United States of America: O'REILLY, 2020.

[17] CHAPMAN, S. J. **Fortran for Scientists and Engineers**. United States: McGraw-Hill Education, 2018.

[18] MASTERSON, R. E.**Introduction to Nuclear Reactor Physics**. United States of America: CRC Press, 2018.

[19] BACKES, A. **Linguagem C: Completa e Descomplicada**. Rio de Janeiro: Elsevier, 2013.

[20] ROMANO, P. K. et al. OpenMC: A State-of-Art Monte Carlo Code for Research and Development. **Annal of Nuclear Energy**, United States. v. 82, p. 90-97, ago. 2015.

[21] CHUNG, K. C. **Introdução à Física Nuclear**. Rio de Janeiro: edUERJ, 2001.

[22] DUDERSTADT, J. J; HAMILTON, L. J. **Nuclear Reactor Analysis**. United States of America: John Wiley & Sons, 1976.

[23] ALVIM, A. C. M. **Métodos Numéricos em Engenharia Nuclear**. Curitiba: CERTA, 2007.

[24] MORSS, L. R; EDELSTEIN, N. M; FUGER, J. **The Chemistry of the Actinide and Transactinide Elements**. Netherlands: Springer, 2010.

[25] ALVES, A. M. S.; HEIMLICH, A.; LAPA, C. M. F. The Inventory And Source Term Simulation Using The New Generation Of Computational Appliances. In: INTERNATIONAL NUCLEAR ATLANTIC CONFERENCE, XXI, 2019, Santos. Anais ... Santos: INAC, 2019

[26] POUNDERS, J. M. **Stochastically Generated Multigroup Diffusion Coeficients**. 2006. 51 f. Dissertação (Master of Science in Nuclear Engineering) - Georgia Institute of Technology, Georgia, USA, 2006.

[27] MOLER, C.; LOAN, C. V. Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later. **SIAM Review**, v. 45, n. 1, p. 3-49 , jan 2006.

[28] SANTOS, T. D. S.; HAYDT, H. M.; FREITAS, C. T. Fabricação de elementos combustíveis para o reator Argonauta do Instituto de Engenharia Nuclear. São Paulo: Instituto de Energia Atômica; maio. 1965. (Publicação I.E.A nº 95)

APÊNDICES

Apêndice A - Cálculos

Apêndice A.1 - Derivação da Equação de Difusão de Nêutrons Multigrupo

Simultaneamente ao processo de fissão, são liberados de 1 a 4 nêutrons que, ficam livres para se movimentar por todo o reator, seguindo o que prediz a lei de Fick, eq. 1. Então, esses nêutrons livres deslocam-se de regiões de alta para baixa concentrações, criando uma distribuição de fluxo de nêutrons que, a priori, não é uniforme. Além disso, alguns nêutrons podem escapar dos limites geométricos do núcleo. No entanto, analisando-se um volume arbitrário ΔV que tenha, por exemplo, coordenadas cartesianas, isto é, dimensões Δx, Δy, Δz e, que tenha uma densidade de nêutrons $n(\vec{r}, E, t)$, haverá uma mudança na densidade de nêutrons no tempo, devido a reações nucleares que ocorrem dentro do volume, tais como, fissão, absorção e captura.

Por outro lado, haverá também variação na densidade de nêutrons, no volume de controle, devido as correntes de nêutrons nas 6 faces (2 $A_x = \Delta y \Delta z$, 2 $A_y = \Delta x \Delta z$ e 2 $A_z = \Delta x \Delta y$). Então, pode-se compactar as considerações anteriores em uma expressão matemática da forma,

$$\Delta x \Delta y \Delta z \frac{\partial n}{\partial t} = \Delta y \Delta z \Delta J_x + \Delta x \Delta z \Delta J_y + \Delta x \Delta y \Delta J_z + \Delta x \Delta y \Delta z \big(\nu \Sigma_f - \Sigma_a\big)\phi \qquad (27)$$

Divide-se a equação 27 pelo volume($\Delta x \Delta y \Delta z$) e, considera-se que o fluxo de nêutrons (ϕ), pode ser expresso pelo produto: $\phi(\vec{r}, E, t) = n(\vec{r}, E, t) \cdot v(E)$. Assim, reorganizando os termos, obtém-se uma expressão no limite infinitesimal,

$$\frac{1}{v(E)} \frac{\partial \phi(\vec{r}, E, t)}{\partial t} = D \nabla^2 \phi(\vec{r}, E, t) + \big(\nu \Sigma_f - \Sigma_a\big)\phi(\vec{r}, E, t) \qquad (28)$$

A equação 28 é conhecida como a Equação de Difusão de Nêutrons monoenergética, dependente do tempo, onde o termo $v(E)$ é a velocidade do média dos nêutrons, $\phi(\vec{r}, E, t)$ é o fluxo escalar de nêutrons, $D = D(\vec{r})$ é o coeficiente de difusão do meio, $\nu = \nu(E)$ é o número de nêutrons liberados pela fissão do actinídeo k e, Σ_f e Σ_a são as seções de choque de fissão e absorção, respectivamente.

A Equação de Difusão de Nêutrons multigrupo é uma extensão quase que direta da equação 28 e, deve ser realizada para o ganho de precisão nas simulações, principalmente para estimar com melhor qualidade, o formato do fluxo rápido e térmico.

Para obter a formulação multigrupo da Equação de Difusão de Nêutrons, considera-e que existam 2 grupos de energia, grupo 1(nêutrons rápidos, i.e, $E_0 = 0.625eV$ e $E_1 = 20MeV$) e, grupo 2 (nêutrons térmicos, i.e, $E_0 = 0eV$ e $E_1 = 0.625eV$). Desta forma, as equações 29 e 30, representam a equação de difusão para o grupo térmico e rápido, respectivamente.

$$-D_2\nabla^2\phi_2 + \Sigma_{a2}\phi_2 = \frac{1}{K}\chi_2\big(\nu_1\Sigma_{f1}\phi_1 + \nu_2\Sigma_{f2}\phi_2\big) + \Sigma_{12}\phi_1 \tag{29}$$

$$-D_1\nabla^2\phi_1 + \Sigma_{a1}\phi_1 + \Sigma_{12}\phi_1 = \frac{1}{K}\chi_1\big(\nu_1\Sigma_{f1}\phi_1 + \nu_2\Sigma_{f2}\phi_2\big) \tag{30}$$

As equações 29 e 30 estão acopladas pelo termo Σ_{12}, que representa a seção de choque de espalhamento do grupo rápido (g=1), para o térmico (g=2). O termo K é o fator de multiplicação do reator, enquanto os termos χ_1 e χ_2 representam o espectro de fissão do grupo rápido e térmico, respectivamente. É importante destacar que, é comum igualar χ_2 a 0. Portanto, a equação 29 poderia ser reescrita da forma,

$$-D_2\nabla^2\phi_2 + \Sigma_{a2}\phi_2 - \Sigma_{12}\phi_1 = 0 \tag{31}$$

Enquanto a equação 30, pode ser reduzida, considerando que, a seção de choque de remoção do grupo rápido como sendo: $\Sigma_{R1} = \Sigma_{a1} + \Sigma_{12}$. Assim, a equação 30, seria matematicamente expressa por,

$$-D_1\nabla^2\phi_1 + \Sigma_{R1}\phi_1 = \frac{1}{K}\chi_1\big(\nu_1\Sigma_{f1}\phi_1 + \nu_2\Sigma_{f2}\phi_2\big) \tag{32}$$

Então, soma-se as equações 31 e 32, para obter apenas uma expressão matemática da forma da equação 33.

$$-\big[D_1\nabla^2\phi_1 + D_2\nabla^2\phi_2\big] + \big[\Sigma_{a1}\phi_1 + \Sigma_{a2}\phi_2\big] = \frac{1}{K}\chi_1\big[\nu_1\Sigma_{f1}\phi_1 + \nu_2\Sigma_{f2}\phi_2\big] \tag{33}$$

Ao ser manipulada a equação 33, retorna-se a forma da Equação de Difusão de Nêutrons

Estacionária monoenergética. No entanto, a forma para mais de 2 grupos de energia da Equação de Difusão de Nêutrons, pode ser obtida, a partir da equação 33, sendo expressa algebricamente por:

$$-D_g\nabla^2\phi_g + \Sigma_{ag}\phi_g + \sum_{g'>g}\Sigma_{gg'}\phi_g = \frac{1}{K}\sum_{g=1}^{G}\chi_g(\nu\Sigma_{fg}\phi_g) + \sum_{g'<g}\Sigma_{g'g}\phi_g \tag{34}$$

A equação **??** contempla os termos de fuga($-D_g\nabla^2\phi_g$), aborção($\Sigma_{ag}\phi_g$), *out-scattering* ($\sum_{g'>g}\Sigma_{gg'}\phi_g$), produção($\sum_{g=1,G}\chi_g(\nu\Sigma_{fg}\phi_g)$) e *in-scattering*($\sum_{g'<g}\Sigma_{g'g}\phi_g$) de nêutrons que, podem ocorrer dentro de um volume de controle arbitrário.

Em sua forma matricial com 2 grupos de energia, a equação 34 fica da forma,

$$\begin{bmatrix} -D_1\nabla^2 + \Sigma_{R1} & 0 \\ -\Sigma_{12} & -D_2\nabla^2 + \Sigma_{a2} \end{bmatrix}\begin{bmatrix}\phi_1 \\ \phi_2\end{bmatrix} = \frac{1}{K}\begin{bmatrix}\chi_1\nu_1\Sigma_{f1} & \chi_1\nu_2\Sigma_{f2} \\ \chi_2\nu_1\Sigma_{f1} & \chi_2\nu_2\Sigma_{f2}\end{bmatrix}\begin{bmatrix}\phi_1 \\ \phi_2\end{bmatrix}$$

Ou ainda,

$$\begin{bmatrix} -D_1\nabla^2 + \Sigma_{R1} & 0 \\ 0 & -D_2\nabla^2 + \Sigma_{a2} \end{bmatrix}\begin{bmatrix}\phi_1 \\ \phi_2\end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\Sigma_{12} & 0 \end{bmatrix}\begin{bmatrix}\phi_1 \\ \phi_2\end{bmatrix} = \frac{1}{K}\begin{bmatrix}\chi_1\nu_1\Sigma_{f1} & \chi_1\nu_2\Sigma_{f2} \\ \chi_2\nu_1\Sigma_{f1} & \chi_2\nu_2\Sigma_{f2}\end{bmatrix}\begin{bmatrix}\phi_1 \\ \phi_2\end{bmatrix}$$

Logo, a equação anterior(Equação de difusão de Nêutrons Multrigrupo) é na verdade um problema de autovalor e autovetor a ser resolvido por algum método numérico, como exemplo, método das diferenças finitas [23].

Apêndice A.2 - Equações de Depleção Actinídeos Expandidas

Neste apêndice são apresentadas as equações de depleção expandidas, referentes aos 21 actinídeos abordados no trabalho.

Urânio:

$$\frac{dN^n_{U^{235}}}{dt}\Big|_{t=t_l} = -N^n_{U^{235}} \sum_{g=1}^{2} \sigma^{U^{235}}_{a,g} \bar{\phi}^n_g \Big|_{t=t_l} \tag{35}$$

$$\frac{dN^n_{U^{236}}}{dt}\Big|_{t=t_l} = -N^n_{U^{236}} \sum_{g=1}^{2} \sigma^{U^{236}}_{a,g} \bar{\phi}^n_g \Big|_{t=t_l} + N^n_{U^{235}} \sum_{g=1}^{2} \sigma^{U^{235}}_{\gamma,g} \bar{\phi}^n_g \Big|_{t=t_l} \tag{36}$$

$$\frac{dN^n_{U^{237}}}{dt}\Big|_{t=t_l} = -N^n_{U^{237}} \Big(\lambda_{U^{237}} + \sum_{g=1}^{2} \sigma^n_{a,g} \bar{\phi}^n_g\Big)\Big|_{t=t_l} + N^n_{U^{236}} \sum_{g=1}^{2} \sigma^{U^{236}}_{\gamma,g} \bar{\phi}^n_g \Big|_{t=t_l} \tag{37}$$

$$\frac{dN^n_{U^{238}}}{dt}\Big|_{t=t_l} = -N^n_{U^{238}} \sum_{g=1}^{2} \sigma^{U^{235}}_{a,g} \bar{\phi}^n_g \Big|_{t=t_l} + N^n_{U^{237}} \sum_{g=1}^{2} \sigma^{U^{237}}_{\gamma,g} \bar{\phi}^n_g \Big|_{t=t_l} \tag{38}$$

$$\frac{dN^n_{U^{239}}}{dt}\Big|_{t=t_l} = -N^n_{U^{239}} \Big(\lambda_{U^{239}} + \sum_{g=1}^{2} \sigma^{U^{239}}_{a,g} \bar{\phi}^n_g\Big)\Big|_{t=t_l} + N^n_{U^{238}} \sum_{g=1}^{2} \sigma^{U^{238}}_{\gamma,g} \bar{\phi}^n_g \Big|_{t=t_l} \tag{39}$$

Neptúnio:

$$\frac{dN^n_{Np^{237}}}{dt} = -N^n_{Np^{237}} \sum_{g=1}^{2} \sigma^{Np^{237}}_{a,g} \bar{\phi}^n_g \Big|_{t=t_l} + \lambda_{U^{237}} N^n_{U^{237}} \Big|_{t=t_l} \tag{40}$$

$$\frac{dN^n_{Np^{238}}}{dt} = -N^n_{Np^{238}}\left(\lambda_{Np^{238}} + \sum_{g=1}^{2}\sigma^{Np^{238}}_{a,g}\bar{\phi}^n_g\right)\Big|_{t=t_l} + N^n_{Np^{237}}\sum_{g=1}^{2}\sigma^{Np^{237}}_{\gamma,g}\bar{\phi}^n_g\Big|_{t=t_l} \tag{41}$$

$$\begin{aligned}\frac{dN^n_{Np^{239}}}{dt} = \; & -N^n_{Np^{239}}\left(\lambda_{Np^{239}} + \textstyle\sum_{g=1}^{2}\sigma^{Np^{237}}_{a,g}\bar{\phi}^n_g\right)\Big|_{t=t_l} + \\ & \lambda_{U^{239}}N^n_{U^{239}} + N^n_{Np^{238}}\textstyle\sum_{g=1}^{2}\sigma^{Np^{238}}_{\gamma,g}\bar{\phi}^n_g\Big|_{t=t_l}\end{aligned} \tag{42}$$

$$\frac{dN^n_{Np^{240}}}{dt} = -\lambda_{Np^{240}}N^n_{Np^{239}}\Big|_{t=t_l} + N^n_{Np^{239}}\sum_{g=1}^{2}\sigma^{Pu^{239}}_{\gamma,g}\bar{\phi}^n_g\Big|_{t=t_l} \tag{43}$$

Plutônio:

$$\frac{dN^n_{Pu^{238}}}{dt}\Big|_{t=t_l} = -N^n_{Pu^{238}}\sum_{g=1}^{2}\sigma^{Pu^{238}}_{a,g}\bar{\phi}^n_g\Big|_{t=t_l} + \lambda_{Np^{238}}N^n_{Np^{238}}\Big|_{t=t_l} \tag{44}$$

$$\begin{aligned}\frac{dN^n_{Pu^{239}}}{dt}\Big|_{t=t_l} = \; & -N^n_{Pu^{239}}\textstyle\sum_{g=1}^{2}\sigma^{Pu^{239}}_{a,g}\bar{\phi}^n_g\Big|_{t=t_l} + N^n_{Pu^{238}}\textstyle\sum_{g=1}^{2}\sigma^{Pu^{238}}_{\gamma,g}\bar{\phi}^n_g\Big|_{t=t_l} + \\ & +\lambda_{Np^{240}}N^n_{Np^{240}}\Big|_{t=t_l}\end{aligned} \tag{45}$$

$$\begin{aligned}\frac{dN^n_{Pu^{240}}}{dt}\Big|_{t=t_l} = \; & -N^n_{Pu^{240}}\textstyle\sum_{g=1}^{2}\sigma^{Pu^{240}}_{a,g}\bar{\phi}^n_g\Big|_{t=t_l} + N^n_{Pu^{239}}\textstyle\sum_{g=1}^{2}\sigma^{Pu^{239}}_{\gamma,g}\bar{\phi}^n_g\Big|_{t=t_l} + \\ & +\lambda_{Np^{239}}N^n_{Np^{239}}\Big|_{t=t_l}\end{aligned} \tag{46}$$

$$\frac{dN^n_{Pu^{241}}}{dt}\Big|_{t=t_l} = -N^n_{Pu^{241}}\left(\lambda_{Pu^{241}} + \sum_{g=1}^{2}\sigma^{Pu^{241}}_{a,g}\bar{\phi}^n_g\right)\Big|_{t=t_l} + N^n_{Pu^{240}}\sum_{g=1}^{2}\sigma^{Pu^{240}}_{\gamma,g}\bar{\phi}^n_g\Big|_{t=t_l} \tag{47}$$

$$\frac{dN^n_{Pu^{242}}}{dt}\Big|_{t=t_l} = -N^n_{Pu^{242}} \sum_{g=1}^{2} \sigma^{Pu^{\bar{2}42}}_{a,g} \phi^n_g \Big|_{t=t_l} + \tag{48}$$

$$\frac{dN^n_{Pu^{243}}}{dt}\Big|_{t=t_l} = -\lambda_{Pu^{243}} N^n_{Pu^{243}}\Big|_{t=t_l} + N^n_{Pu^{242}} \sum_{g=1}^{2} \sigma^{Pu^{242}}_{\gamma,g} \bar{\phi}^n_g \Big|_{t=t_l} \tag{49}$$

Amerício:

$$\frac{dN^n_{Am^{241}}}{dt}\Big|_{t=t_l} = -N^n_{Am^{241}} \sum_{g=1}^{2} \sigma^{Am^{241}}_{a,g} \bar{\phi}^n_g \Big|_{t=t_l} + \lambda_{Pu^{241}} N^n_{Pu^{241}}\Big|_{t=t_l} \tag{50}$$

$$\frac{dN^n_{Am^{242}}}{dt}\Big|_{t=t_l} = +N^n_{Am^{241}} \sum_{g=1}^{2} \sigma^{Am^{241}}_{\gamma,g} \bar{\phi}^n_g \Big|_{t=t_l} - \left(\lambda_{Am^{242}} + \sum_{g=1}^{2} \sigma^{Am^{242}}_{a,g} \bar{\phi}^n_g\right) N^n_{Am^{242}}\Big|_{t=t_l} \tag{51}$$

$$\frac{dN^n_{Am^{243}}}{dt}\Big|_{t=t_l} = \lambda_{Pu^{243}} N^n_{Pu^{243}}\Big|_{t=t_l} + N^n_{Am^{242}} \sum_{g=1}^{2} \sigma^{Am^{\bar{2}42}}_{\gamma,g} \phi^n_g \Big|_{t=t_l} \tag{52}$$

Cúrio:

$$\frac{dN^n_{Cm^{242}}}{dt}\Big|_{t=t_l} = -N^n_{Cm^{242}} \sum_{g=1}^{2} \sigma^{Cm^{242}}_{a,g} \bar{\phi}^n_g \Big|_{t=t_l} + \lambda_{Am^{242}} N^n_{Am^{242}}\Big|_{t=t_l} \tag{53}$$

$$\frac{dN^n_{Cm^{243}}}{dt}\Big|_{t=t_l} = -N^n_{Cm^{243}} \sum_{g=1}^{2} \sigma^{Cm^{243}}_{a,g} \bar{\phi}^n_g \Big|_{t=t_l} + N_{Cm^{243}} \sum_{g=1}^{2} \sigma^{Cm^{242}}_{\gamma,g} \bar{\phi}^n_g \Big|_{t=t_l} \tag{54}$$

$$\frac{dN^n_{Cm^{244}}}{dt}\Big|_{t=t_l} = -N^n_{Cm^{243}} \sum_{g=1}^{2} \sigma^{Cm^{244}}_{a,g} \bar{\phi}^n_g \Big|_{t=t_l} + N_{Cm^{243}} \sum_{g=1}^{2} \sigma^{Cm^{243}}_{\gamma,g} \bar{\phi}^n_g \Big|_{t=t_l} \tag{55}$$

Apêndice A.3 - Cálculo da Célula do Reator Argonauta

A homogeneização de uma célula de um reator é o primeiro passo para se realizar a simulação de uma célula ou de um reator por completo. Essa homogeneização é feita através de cálculos neutrônicos, considerando a composição de materiais estruturantes do núcleo.

Como um reator nuclear é uma estrutura bastante heterogênea, a primeira tarefa ao tentar iniciar a simulação de um reator é tentar através de cálculos, transformar um reator heterogêneo em homogêneo, para assim, simular este em algum programa de cálculos neutrônicos pertinentes, tais como o HAMMER ou o OpenMC.

O HAMMER é um programa de simulação de células de reatores, que é escrito na linguagem de programação FORTRAN. O mesmo foi utilizado no passado pelo IEN, para simular as células do reator Argonauta. No entanto, devido a ser um código legado, com dificuldade de achar manuais de referencia atualizados e, que dificilmente simula um reator por completo, optou-se pelo uso de um código mais atual, denominado de OpenMC.

O OpenMC é um código baseado no método de Monte Carlo, que foi desenvolvido pelo grupo de Física de Reatores do Instituto de Massachussets, no inicio de 2011 e, que atualmente tem padrão de desenvolvimento aberto a comunidade. Este código possibilita simular Assemblys de reatores completos. O núcleo do código é escrito em C++ para deixar o algoritmo rápido. Além disso, o código contempla ainda uma rica API em Python que possibilita realizar construção e simulação de reatores nucleares reais com uma relativa facilidade. No entanto, o OpenMC, bem como, tantos outros códigos de simulação de reatores tem suas certas limitações, por exemplo, a maioria dos códigos que simula Assembly de reatores nucleares implementa simulações em geometria cartesiana ou hexagonal, pois a maioria dos reatores de potência utiliza esse tipo de geometria.

O reator Argonauta traz uma dificuldade adicional, por apresentar uma geometria cilíndrica. Essa dificuldade pode ser contornada transformando a geometria do reator para geometria cartesiana, a exemplo do procedimento adotado no relatório interno do cálculo da célula do reator Argonauta [13].

Considerações iniciais:

Segundo o relatório interno do IEN [13], o reator Argonauta é bem heterogêneo, necessitando de uma transformação de geometria, para uma mais simples, para que se possa simula-lo em algum programa de simulação de células de reatores. Essa transformação leva em conta as seções retas de cada parte do reator, bem como, os respectivos pesos neutrônicos. Nesse sentido, a parte anular do reator Argonauta é transformado em uma região de retângulo, onde as

lâminas de U_3O_8 fiquem paralelas entre si, e espaçadas pelas cunhas de grafite.

Os elementos combustíveis na parte anelar entre os cilindros concêntricos, são divididos em 3 macro células:

- Região A: parte ativa do reator, contendo o miolo(U_3O_8 + Al), revestimento (Al) e água (H_2O) na região paralela as lâminas;
- Regiao V: parte inerte do reator, contendo as pontas de Al das lâminas e água (H_2O) entre a parte ativa do reator e a face das cunhas de grafite laterais;
- Região M: parte refletora do reator, contendo $\frac{1}{2}$ cunhas de grafite laterais a parte ativa.

As regiões anteriormente mencionadas possuem as seguintes áreas [13]:

- Região A $= 50.174\, cm^2$;
- Regiao V $= 8.390\, cm^2$;
- Regiao M $= 17.442\, cm^2$.

A soma das áreas das regiões resulta em uma macro célula C de área igual a $76.006\, cm^2$. No entanto, para este trabalho foi necessário duplicar os valores anteriormente informados, pois no relatório interno do IEN [13] foi simulado apenas a metade de um Assembly, devido a simetria. Então, como o objetivo inicial desse trabalho é construir uma célula do reator Argonauta, para depois construir o reator por completo, foi necessário essas adaptações. Desta forma, tem-se,

- A $= 100.348\, cm^2$;
- V $= 16.78\, cm^2$;
- M $= 34.884\, cm^2$;
- C $= 152.012\, cm^2$.

Uma alternativa a obtenção da área da macro célula C

Uma forma de verificar se o valor da área da macro célula C foi estimado de forma correta, é através do calculo de área total anelar que, pode ser obtido pela diferença entre as área dos cilindros, da seguinte forma:

$$A_{anelar} = \pi \times (45.72^2 - 30.48^2)\, cm^2 = 3648.29385\, cm^2 \tag{56}$$

A seção reta anelar é a área total disponível para inserção dos Elementos combustível com as cunhas de grafite espaçadoras. No entanto, é necessário saber apenas a área de um EC, que pode ser estimado pela equação 57, levando em consideração que 6ECs ocupam um quarto da área anelar. Assim, segue-se que, á área de 1 EC é:

$$A_{EC} = 3648.29385 \cdot \frac{1}{4} \cdot \frac{1}{6}\, cm^2 = 152.0122437\, cm^2 \tag{57}$$

Percebe-se que o valor obtido pela eq. 57 é muito próximo ao descrito pelo relatório [13], com uma pequena diferença devido a possíveis aproximações de casas decimais durante o processo de realização dos cálculos.

O comprimento normalizador dos elementos combustíveis

O comprimento normalizador é um valor para ser utilizado como referencia na transformação de uma geometria complexa para uma mais simples. Em [13], o comprimento normalizador é a largura ativa das lâminas de combustível, isto é, $C_N = 6.575cm$, pois a intenção era simular uma célula. No entanto, nosso objetivo é simular o reator Argonauta por completo. Nesse sentido, o comprimento normalizador adotado neste trabalho, é dado pela diferença entre os raios dos cilindros, isto é:

$$C_N = (45.72 - 30.48)\, cm = 15.24\, cm \tag{58}$$

A Largura dos Assemblys

A largura dos Assemblys é calculado da seguinte forma:

$$L_{EC} = \frac{152.012\, [cm^2]}{C_N\, [cm]} = 9.974540682\, cm \tag{59}$$

Largura da Regiao M

A largura da Região M, referente ao grafite é calculado da seguinte forma:

$$L_G = \frac{34.884\,[cm^2]}{C_N\,[cm]} = 2.288976378\,cm \tag{60}$$

Por outro lado, esse valor será partilhado em duas metades, isto é:

$$L_{1/2G} = \frac{2.288976378\,cm}{2} = 1.144488189\,cm \tag{61}$$

Largura da Região V

A largura da Região V, referente a região interne das pontas de Alumínio e Água em excesso, é calculada da seguinte forma:

$$L_V = \frac{16.78\,[cm^2]}{C_N\,[cm]} = 1.101049869\,cm \tag{62}$$

Por outro lado, o valor imediatamente anterior é partilhado em duas metades.

$$L_{1/2V} = \frac{1.101049869\,cm}{2} = 0.5505249344\,cm \tag{63}$$

Cálculo do Área e Volumes das Diferentes Regiões do Reator

O Assembly do reator Argonauta, contendo 17 Lâminas de combustível, deve-ser entendido como um conjunto de células individuais que, ainda podem ser subdivididas em 5 regiões diferentes, a saber:

1. Região 1: Parte onde fica o miolo do combustível, constituída de U_3O_8(natural ou enriquecido a 20% no isotopo U^{235}) e Alumínio. Esta região tem dimensões 0.187 cm por 6.575 cm;

2. Região 2: Revestimento ativo do miolo, composto puramente de liga de Alumínio 1100. Essa região é composta de duas metades, cada uma com as seguintes dimensões, 0.028 cm por 6.575 cm;

3. Região 3: Constituída puramente de água entre as placas paralelas. Essa região é composta de duas metades, cada uma tendo as seguintes dimensões, 6.575 cm por 0.32673 cm;

4. Região 4: Composta de Alumínio inerte das pontas da lâminas e, água entre a região ativa

do reator e as faces das cunhas de grafite adjacente as lâminas. Essa região é composta de duas metades, cada uma com as seguintes dimensões, 0.8964705882352941 cm por 0.5505249344 cm;

5. Região 5: Composta puramente duas metades de cunha de grafite, cada uma tendo as seguintes dimensões, 1.144488189 cm por 0.8964705882352941 cm.

Então, o volume respectivo a cada região é baseado nas informações anteriores e, tendo como referencia que a altura ativa do reator é de 60.2 cm [13]. Assim, o volume de cada região é calculado pelas equações de 64 a 68.

Volume da Região 1:

$$V_{R1} = 0.187 \cdot 6.575 \cdot 60.2\, cm^3 = 74.017405\, cm^3 \tag{64}$$

Volume da Região 2:

$$V_{R2} = 2 \cdot 0.028 \cdot 6.575 \cdot 60.2\, cm^3 = 22.16564\, cm^3 \tag{65}$$

Volume da Região 3:

$$V_{R3} = 2 \cdot 0.3267352941176471 \cdot 6.575 \cdot 60.2\, cm^3 = 258.6534609\, cm^3 \tag{66}$$

Volume da Região 4:

$$V_{R4} = 2 \cdot 0.5505249344 \cdot 0.8964705882352941 \cdot 60.2\, cm^3 = 59.42094118\, cm^3 \tag{67}$$

Volume da Região 5:

$$V_{R5} = 2 \cdot 0.8964705882352941 \cdot 1.144488189 \cdot 60.2\, cm^3 = 123.5304\, cm^3 \tag{68}$$

Cálculo dos Pesos Neutrônicos de Cada Região da Célula

O peso neutrônico de cada região é estimado pela equação 69 e, consiste em obter a densidade média de nêutrons de um dado nuclídeo dentro do volume analisado.

$$N_i = \frac{m_i \times N_a}{V_R \times A_i} \, [atomos/cm^3] \tag{69}$$

Onde:

N_i	Densidade do nuclídeo i no volume da região R, dado em átomos / cm^3;
m_i	Massa em gramas do nuclídeo i dentro do volume da região R;
N_a	Constante de Avogrado, igual a $6.0225 \times 10^{23} atomos/mol$;
V_R	Volume da região R, dado em cm^3;
A_i	Massa molar do nuclídeo ou molécula i, dado em g/mol.

Uma rápida análise dimensional da equação 69, percebe-se que a densidade de átomos no volume é de fato dado em átomos por cm^3.

Peso neutrônico da região 1

O Peso neutrônico desta região deve ser calculado duas vezes, porque um dos resultados será utilizado na criação da célula com enriquecimento de 20% e o outro resultado será utilizado para a criação da célula sem enriquecimento.

Peso neutrônico da região 1 considerando o enriquecimento de U^{235}

Segundo o relatório de fabricação das lâminas utilizado nos assemblys do Argonauta [28], cada lâmina contem 21 g de U^{235}. Além disso, também é informado que o miolo é composto de 54.36% fração de massa de U_3O_8. Enquanto, o restante 45.64% da massa total, é constituída de Alumínio que serve para que, o dióxido de Urânio aderir melhor ao revestimento, promovendo uma menor possibilidade de fuga de material radioativo para a água. Baseado nesses dados, segue os cálculos de peso neutrônico considerando o enriquecimento de 20% de U^{235}

Massa total de Urânio no miolo (m_{Uranio}):

$$m_{Uranio} = \frac{21\,g}{0.1991} = 105.4746359\,g \tag{70}$$

Massa total de Urânio 238 no miolo ($m_{U_{238}}$):

$$m_{U_{238}} = (105.4746359 - 21)g = 84.47463586g \tag{71}$$

Densidade de Urânio 235:

$$N_{U235} = \frac{21 \cdot 6.0225 \times 10^{23}}{74.017405 \cdot 235} = 7.2710E + 20\,atomos/cm^3 \tag{72}$$

Densidade de Urânio 238:

$$N_{U238} = \frac{84.47463586 \cdot 6.0225 \times 10^{23}}{74.017405 \cdot 238} = 2.8879E + 21\,atomos/cm^3 \tag{73}$$

Densidade total de Urânio no miolo:

$$N_U = N_{U235} + N_{U238} = 3.6150E + 20\,atomos/cm^3 \tag{74}$$

Analisando a formula do U_3O_8, percebe-se que para cada 3 átomos de Urânio no volume da região 1, temos 8 átomos de Oxigênio. Logo, pode-se a densidade de átomos de Oxigênio, pela densidade de átomos de Urânio, pela formula a seguir,

$$N_{O16} = \frac{8.0}{3.0} \cdot N_U\,atomos/cm3 = 9.6401E + 21\,atomos/cm^3 \tag{75}$$

Manipulando a formula da equação 69, pode-se estimar a massa de O^{16}, da seguinte forma,

$$m_{O^{16}} = \frac{N_{O16} \cdot V_{R1} \times A_{O16}}{N_a}\,g = 18.95667827\,g \tag{76}$$

Desta forma temos uma massa de U_3O_8 equivalente a,

$$m_{U_3O_8} = (m_U + m_{O16})\, g = 124.4313142\, g \tag{77}$$

Com o valor da massa de U_3O_8, pode-se estimar a massa de Alumínio no miolo das células de combustível.

$$m_{Al27} = \frac{0.4564 \cdot m_{U_3O_8}}{1 - 0.4564} = \frac{0.4564 \cdot 124.4313142}{0.5436} = 104.4710298\, g \tag{78}$$

A densidade de Al^{27} dentro do miolo da célula de combustível:

$$N_{Al27} = \frac{104.4710298 \cdot 6.0225 \times 10^{23}}{74.017405 \cdot 26.9815}\, atomos/cm^3 = 3.1504E + 22\, atomos/cm^3 \tag{79}$$

Peso neutrônico da região 1 considerando Urânio natural

2 dos elementos combustíveis (EC) do reator Argonauta são constituídos de 8 Lâminas U_3O_8 com Urânio na sua concentração natural. De acordo com [28], cada lâmina dessas estruturas contem em média $121.8\, g$ de U_3O_8. Baseado nessas informações a nas concentrações naturais dos isotopos do Urânio contida em [18] é possível estimar o peso neutrônico da célula representativa para essas lâminas.

Massa molar do Urânio natural:

$$M_{U_{NAT}} = 0.992263 \cdot 238 + 0.007196 \cdot 235 + 0.000539 \cdot 234 = 237.97578\, g/mol \tag{80}$$

Massa molar do U_3O_8:

$$M_{U_3O_8} = 3 \cdot 237.97578 + 8 \cdot 16 = 841.92734\, g/mol \tag{81}$$

Massa de Urânio no miolo contendo U_3O_8 natural:

$$m_U = \frac{3 \cdot 237.97578}{841.92734} \cdot 121.8\, g = 103.2824876\, g \tag{82}$$

A massa dos nuclídeos de Urânio na amostra de U_3O_8:

$$\begin{aligned} m_{U^{234}} &= 103.2824876 \cdot 0.000539\, g = 0.05566926082\, g \\ m_{U^{235}} &= 103.2824876 \cdot 0.007196\, g = 0.7432207808\, g \\ m_{U^{238}} &= 103.2824876 \cdot 0.992263\, g = 102.483391\, g \end{aligned} \tag{83}$$

Os pesos neutrônicos dos isotopos de Urânio são calculados por:

$$\begin{aligned} N_{U234} &= \frac{0.05566926082 \cdot N_a}{74.017405 \cdot 234} = 1.935720062E + 18\, atomos/cm^3 \\ N_{U235} &= \frac{0.7432207808 \cdot N_a}{74.017405 \cdot 235} = 2.573314906E + 19\, atomos/cm^3 \\ N_{U238} &= \frac{102.483391 \cdot N_a}{74.017405 \cdot 238} = 3.503640063E + 21\, atomos/cm^3 \\ N_U &= N_{U234} + N_{U235} + N_{U238} = 3.531308932E + 21\, atomos/cm^3 \end{aligned} \tag{84}$$

Por outro lado, o peso neutrônico do Oxigênio é calculado pela densidade total de atomos de Urânio, contido em 84 e, sabendo que, a formula do combustível utilizado é U_3O_8,

$$N_{O16} = \frac{8.0}{3.0} \cdot N_U = 9.416823819E + 21\, atomos/cm^3 \tag{85}$$

Massa de Alumínio utilizado em conjunto ao miolo de U_3O_8,

$$m_{Al} = \frac{0.4564 \cdot 121.8}{0.5436} = 102.2618102\, g \tag{86}$$

Peso neutrônico do Alumínio no miolo de U_3O_8 natural,

$$N_{Al} = \frac{102.2618102 \cdot N_a}{74.017405 \cdot 26.9815} = 3.083829367E + 22\, atomos/cm^3 \tag{87}$$

Apêndice B - Especificações, Esquemas de Ligação e Códigos

Apêndice B.1 - Especificação e Conexões do Raspberry Pi 3B, ADC MCP3008 e LCD LCM1602

O Raspberry Pi é uma Single Board Computer (SBC) que, possui 40 portas digitais que, possibilitam conectar outros periféricos, tais como, leds, servo motores e sensores. Todavia, devido a ampla gama de aplicações, linguagens de programação que o Raspberry Pi suporta, existem 3 padrões de numeração da portas de comunicação que são possíveis, a numeração física, de 1 a 40, a oficial do fabricante (Broadcom - BCM) e a wiringPi(wPi), sendo esta ultima mais utilizada por programadores em linguagem C.

A especificação de cada porta do Raspberry Pi 3B é descrita na tabela 13. Neste aspecto, é importante frisar que, este trabalho utilizou o esquema de numeração de portas wiringPi.

BCM	wPi	Nome	Modo	Porta Física		Modo	Nome	wPi	BCM
-	-	3.3v	-	1	2	-	5v	-	-
2	8	SDA.1	IN	3	4	-	5v	-	-
3	9	SCL.1	IN	5	6	-	0v	-	-
4	7	GPIO.7	IN	7	8	IN	TxD	15	14
-	-	0v	-	9	10	IN	RxD	16	15
17	0	GPIO.0	IN	11	12	IN	GPIO.1	1	18
27	2	GPIO.2	IN	13	14	-	0v	-	-
22	3	GPIO.3	IN	15	16	IN	GPIO.4	4	23
-	-	3.3v	-	17	18	IN	GPIO.5	5	24
10	12	MOSI	ALT	19	20	-	0v	-	-
9	13	MISO	ALT	21	22	IN	GPIO.6	6	25
11	14	SCLK	ALT	23	24	IN	CE0	10	8
-	-	0v	-	25	26	IN	CE1	11	7
0	30	SDA.0	IN	27	28	IN	SCL.0	31	1
5	21	GPIO.21	IN	29	30	-	0v	-	-
6	22	GPIO.22	IN	31	32	IN	GPIO.26	26	12
13	23	GPIO.23	IN	33	34	-	0v	-	-
19	24	GPIO.24	IN	35	36	IN	GPIO.27	27	16
26	25	GPIO.25	IN	37	38	IN	GPIO. 28	28	20
-	-	0v	-	39	40	IN	GPIO.29	29	21

Tabela 13: Especificação das portas do Raspberry Pi, para os padrões BCM, Física e WiringPi(wPi)

Para a leitura de sensores analógicos, como o utilizado no Argonauta, faz-se necessário a utilização de um ADC, isto é, um Analog Converter to Digital (Conversor de Analógico para

Digital), pois os Raspberry Pi, por padrão não traz essa funcionalidade de forma embarcada. A figura 13, mostra o esquema das portas do ADC MCP3008 que foi utilizado neste trabalho. A figura 14 mostra o esquema de ligação das Portas do Raspberry Pi às portas do ADC MCP3008. Enquanto a figura 15, mostra o esquema de ligação entre o Raspberry Pi 3B e o display LCD (LCM1602).

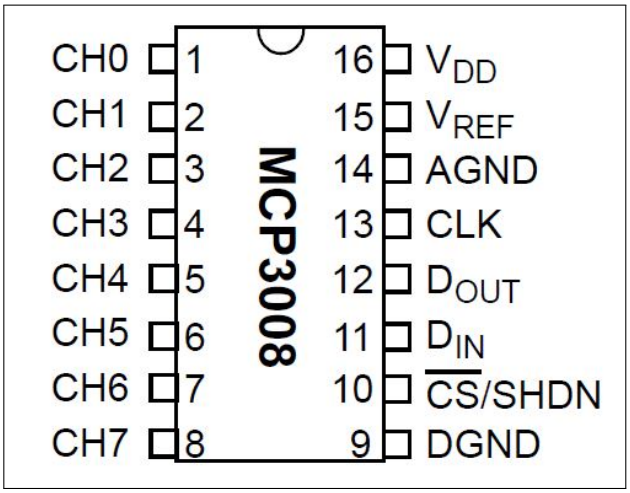

Figura 13: Especificação das portas do ADC MCP3008

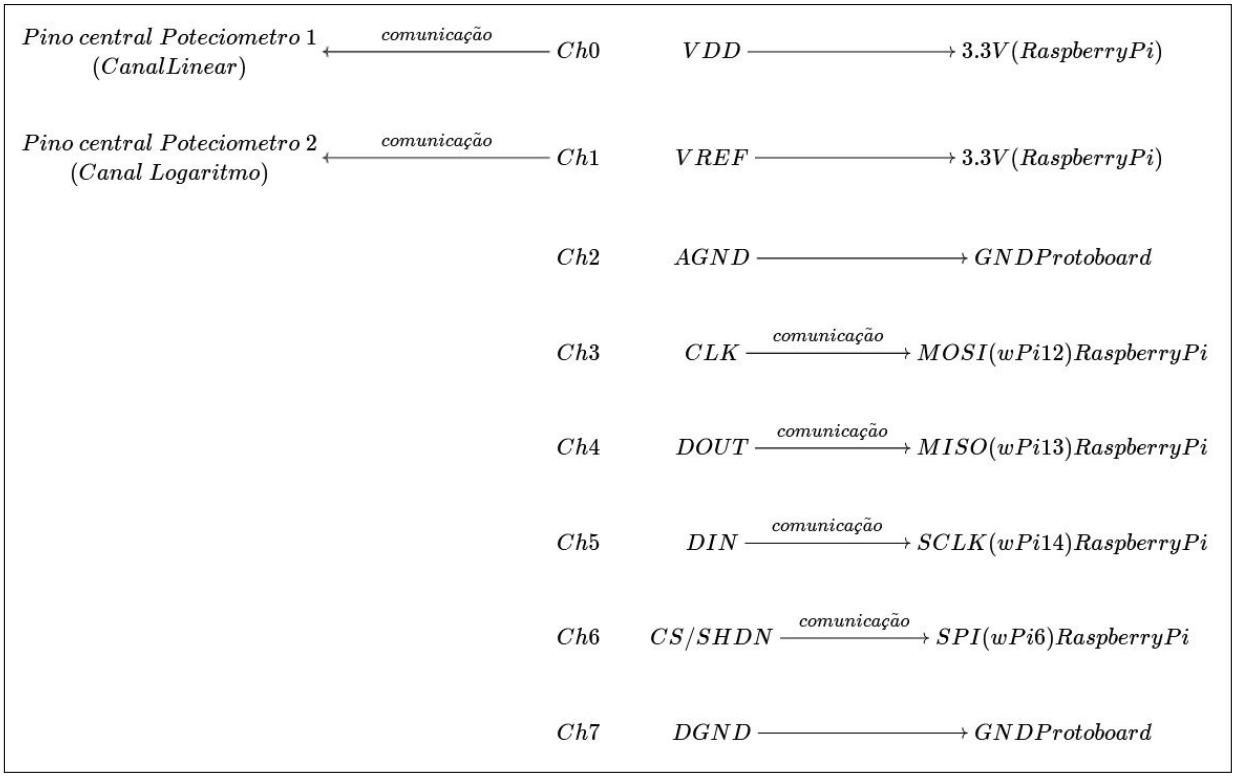

Figura 14: Esquema de ligações (conexões) entre o Raspberry Pi 3B e o ADC MCP3008

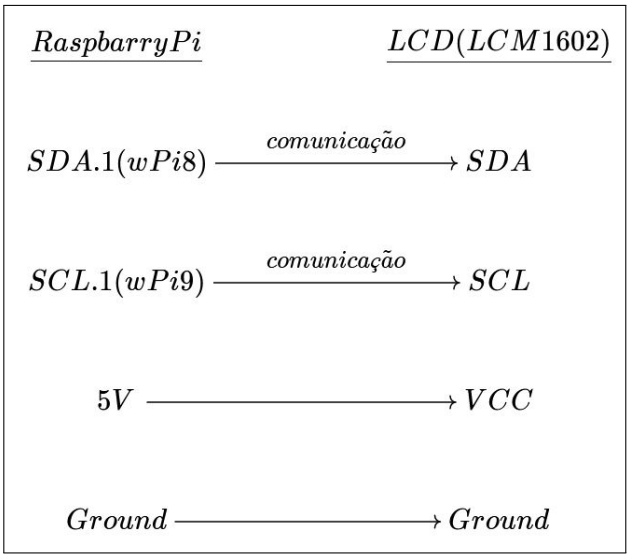

Figura 15: Esquema de ligações(conexões) entre o Raspbarry Pi 3B e o LCD LCM1602

Apêndice B.2 - Código em C, Para Leitura de Sensor Analógico com MCP3008

```
1   #include <stdint.h>
2   #include <string.h>
3   #include <errno.h>
4   #include <wiringPi.h>
5   #include <stdio.h>
6   #include <stdlib.h>
7   #include <wiringPiSPI.h>
8   #include <wiringPiI2C.h>
9
10  #define LCDADDR                 0x27  //IIC LCD address
11  #define BLEN                    1     //1--open backlight,0--close backlight
12
13  #define CHAN_CONFIG_SINGLE      8     //setup channel 0 as Single-ended input
14  #define SPICHANNEL              0     //MCP3008 connect to SPI0
15  #define ANALOGCHANNEL           0     //Potentiometer connect MCP3008 analog
        channel 0
16  #define ANALOGCHANNEL2          1
17
18  static int spifd ;
19  static int i2cfd;
20
21  void spiSetup (int spiChannel)
22  {
23      if ((spifd = wiringPiSPISetup (spiChannel, 10000)) < 0)
24      {
25          fprintf (stderr, "Can't open the SPI bus: %s\n", strerror (errno))
        ;
26          exit (EXIT_FAILURE) ;
```

```
27          }
28      }
29
30      int myAnalogRead(int spiChannel,int channelConfig,int analogChannel)
31      {
32          if(analogChannel<0 || analogChannel>7)
33          return -1;
34          unsigned char buffer[3] = {1}; // start bit
35          buffer[1] = (channelConfig+analogChannel) << 4;
36          wiringPiSPIDataRW(spiChannel, buffer, 3);
37          return ( (buffer[1] & 3 ) << 8 ) + buffer[2]; // get last 10 bits
38      }
39
40      //write a word to lcd
41      void write_word(int data){
42          int temp = data;
43          if ( BLEN == 1 )
44          temp |= 0x08;
45          else
46          temp &= 0xF7;
47          wiringPiI2CWrite(i2cfd, temp);
48      }
49
50      //send command to lcd
51      void send_command(int comm){
52          int buf;
53          // Send bit7-4 firstly
54          buf = comm & 0xF0;
55          buf |= 0x04;            // RS = 0, RW = 0, EN = 1
56          write_word(buf);
57          delay(2);
58          buf &= 0xFB;            // Make EN = 0
59          write_word(buf);
60
61          // Send bit3-0 secondly
62          buf = (comm & 0x0F) << 4;
63          buf |= 0x04;            // RS = 0, RW = 0, EN = 1
64          write_word(buf);
65          delay(2);
66          buf &= 0xFB;            // Make EN = 0
67          write_word(buf);
68      }
69
70      //send data to lcd
71      void send_data(int data){
72          int buf;
73          // Send bit7-4 firstly
```

```
74          buf = data & 0xF0;
75          buf |= 0x05;              // RS = 1, RW = 0, EN = 1
76          write_word(buf);
77          delay(2);
78          buf &= 0xFB;              // Make EN = 0
79          write_word(buf);
80
81          // Send bit3-0 secondly
82          buf = (data & 0x0F) << 4;
83          buf |= 0x05;              // RS = 1, RW = 0, EN = 1
84          write_word(buf);
85          delay(2);
86          buf &= 0xFB;              // Make EN = 0
87          write_word(buf);
88      }
89
90      //initialize the lcd
91      void init(){
92          send_command(0x33); // Must initialize to 8-line mode at first
93          delay(5);
94          send_command(0x32); // Then initialize to 4-line mode
95          delay(5);
96          send_command(0x28); // 2 Lines & 5*7 dots
97          delay(5);
98          send_command(0x0C); // Enable display without cursor
99          delay(5);
100         send_command(0x01); // Clear Screen
101         wiringPiI2CWrite(i2cfd, 0x08);
102     }
103
104     //clear screen
105     void clear(){
106         send_command(0x01); //clear Screen
107     }
108
109     //Print the message on the lcd
110     void write(int x, int y, char data[]){
111         int addr, i;
112         int tmp;
113         if (x < 0)  x = 0;
114         if (x > 15) x = 15;
115         if (y < 0)  y = 0;
116         if (y > 1)  y = 1;
117
118         // Move cursor
119         addr = 0x80 + 0x40 * y + x;
120         send_command(addr);
```

```
121
122         tmp = strlen(data);
123         for (i = 0; i < tmp; i++){
124             send_data(data[i]);
125         }
126     }
127
128     int main()
129     {
130         int adc;
131         int adc2;
132         int segundos;
133         float voltage;
134         float voltage2;
135         float acumulador_linear=0.0, acumulador_logaritmo=0.0;
136         float media_linear, media_logaritma;
137         char buf[5];
138
139         FILE *arquivo1, *arquivo2, *arquivo3;
140         if(wiringPiSetup() < 0)
141         {   fprintf(stderr,"Can't init wiringPi: %s\n",strerror(errno));
142             exit(EXIT_FAILURE);
143         }
144
145         spiSetup(SPICHANNEL);//init spi
146
147         i2cfd = wiringPiI2CSetup(LCDADDR);//init i2c
148         init();//init LCD
149         clear();//clear screen
150
151         arquivo1 = fopen("voltagens_atuais.txt","r");
152         if(arquivo1 == NULL){
153             printf("O Arquivo nao existe ainda!");
154         }
155
156         arquivo2 = fopen("historico_voltagens.txt","a");
157
158         for(segundos=0; segundos<30; segundos++)
159         {
160             adc = myAnalogRead(SPICHANNEL,CHAN_CONFIG_SINGLE,ANALOGCHANNEL);
161             adc2 = myAnalogRead(SPICHANNEL, CHAN_CONFIG_SINGLE, ANALOGCHANNEL2
    );
162             voltage = adc/1024.*20.0;
163             write(0,0,"Ch Linear:");
164             sprintf(buf,"%2.2f",voltage);//float change to string
165             write(10,0,buf);//print voltage on lcd
166             write(15,0,"V");//print unit
```

```
167
168             write(0,1,"Ch Logarit:");
169             voltage2 = adc2/1024.*20.0;
170             sprintf(buf,"%2.2f",voltage2);
171             write(11,1, buf);
172             write(16,1,"V");
173
174             acumulador_linear += voltage;
175             acumulador_logaritmo += voltage2;
176             // Espera 1 segundos entre as afericoes:
177             delay(200);
178         }
179
180         media_linear = acumulador_linear / 150.0;
181         media_logaritma = acumulador_logaritmo / 150.0;
182
183         if(arquivo1 == NULL){
184             printf("Arquivo nao foi fechado, porque nao foi aberto");
185         }else{
186             fclose(arquivo1); // Finaliza a leitura do arquivo1
187         }
188
189         // Agora abre o arquivo 1 em modo de escrita:
190         arquivo3 = fopen("voltagens_atuais.dat","w");
191
192         fprintf(arquivo3,"%.2f %.2f\n",media_linear, media_logaritma);
193         fprintf(arquivo2, "%.2f %.2f\n",media_linear, media_logaritma);
194
195         fclose(arquivo3);
196         fclose(arquivo2);
197
198         return 0;
199     }
```

Código 11: lerSensores.c

Apêndice B.3 - Código em Python, Para Envio de Dados de Potencia do Reator Para o Banco de Dados

```
# Importacao das bibliotecas necessarias:
from datetime import datetime
import sqlite3

# Obtendo a data e hora atual
data_hora_atual = datetime.now()

data_atual = data_hora_atual.strftime("%Y-%m-%d")
hora_atual = data_hora_atual.strftime("%H:%M:%S")

# Abrir arquivo com as voltagens atuais:
path_arquivo = '../Data/voltagens_atuais.dat'
arquivo =  open(path_arquivo,'r')

voltagens = arquivo.readline().split()

voltagem_canal_linear = float(voltagens[0])
voltagem_canal_logaritmo = float(voltagens[1])

arquivo.close()

# Estabelecendo conexao com o banco de dados:
path_database = '../Data/Databases/HistoricoPotenciaArgonauta.db'
conexao = sqlite3.connect(path_database)

# Criando cursor para operar no banco de dados:
cursor = conexao.cursor()

cursor.execute("""INSERT INTO voltagens(data, hora, canal_linear, \
canal_logaritmo) VALUES(
?, ?, ?, ?
);""", (data_atual, hora_atual, voltagem_canal_linear, \
voltagem_canal_logaritmo))

conexao.commit()

# Abrir o arquivo para Fluxo_E_Potencia_Atuais_Argonauta.dat:
path_arquivo = '../Data/Fluxo_Potencia_Atuais_Argonauta.dat'

arquivo = open(path_arquivo, 'r')

pot_total = arquivo.readline()
pot_total = float(pot_total)
```

```
45      # Ler linha em branco que sera descartada:
46      _ = arquivo.readline()

48      pot = [] # Lista contendo as respectivas potencias de
49      # cada Elemento Combustivel, estimada
50      # via simulacao do codigo Dif1D.

52      # Ler demais dados do arquivo:
53      for l in range(8):
54      linha = arquivo.readline().split()
55      pot.append(float(linha[2]))

57      arquivo.close()

59      campos = 'potencia_EC1, potencia_EC2, potencia_EC3, potencia_EC4, ' + \
60      'potencia_EC5, potencia_EC6, potencia_EC7, potencia_EC8, potencia_total'

62      valores = (data_atual, hora_atual, \
63      pot[0], pot[1], pot[2], pot[3], pot[4], pot[5], \
64      pot[6], pot[7], pot_total)

66      cursor.execute("""INSERT INTO potencias(data, hora, """ + \
67      campos + """) VALUES(
68      ?, ?,
69      ?, ?, ?, ?, ?, ?, ?, ?
70      );""", valores)

72      conexao.commit()
73      conexao.close()
```

Código 12: SendDataToHistoricoPotencia.py

Apêndice B.4 - Código em Python, Para Envio de Dados de Concentrações Isotópicas Para o Banco de Dados

```
# Algoritmo para enviar os dados dos arquivos de texto
# para o banco de dados:
# Importacao das bibliotecas necessarias:
from datetime import datetime
import sqlite3
# Abrindo o arquivo concentracoes_atuais.dat em modo leitura:
path_arquivo = '../Data/concentracoes_atuais.dat'
arquivo = open(path_arquivo,'r')
d = [] # d -> eh abreviacao para dados
# Vamos ler 51 linhas de concentracoes isotopicas:
for i in range(51):
# Ler uma linha contendo 8 dados.
# O split eh para criar uma lista com esses valores.
linha_dados = arquivo.readline().split()
# Os dados sao lidos como string.
# Entao convertemos os dados para numerico,
# para inserir no Banco de Dados adequadamente.
for j in range(8):
linha_dados[j] = float(linha_dados[j])
d.append(linha_dados)
arquivo.close()

######################################################################
# Obter a data e hora atual:
data_hora_atual = datetime.now()
data_atual = data_hora_atual.strftime('%Y-%m-%d')
hora_atual = data_hora_atual.strftime('%H:%M:%S')

######################################################################
# Transferindo os dados para o Banco de Dados:
# Estabelecendo conexao com o Banco de Dados:
path_database = '../Data/Databases/ConcentracoesIsotopicasArgonauta.db'
conexao = sqlite3.connect(path_database)
cursor = conexao.cursor()
# Sigla dos elementos, cujas as informacoes
# serao transferidas para o banco de dados:
nuclideos = 'U234, U235, U236, U237, U238, U239, ' + \
'Np237, Np238, Np239, Np240, ' + \
'Pu238, Pu239, Pu240, Pu241, Pu242, ' + \
'Am241, Am242, Am243, ' + \
'Cm242, Cm243, ' + \
'Br85, Br86, ' + \
'Kr85, Kr85m, Kr88, ' + \
'Sr91, Sr92, ' + \
```

```
45      'Y90, Y91, Y92, Y93, ' + \
46      'Zr95, ' + \
47      'Nb95, Nb96, ' + \
48      'Mo99, ' + \
49      'Te133, Te134, ' + \
50      'I130, I131, I132, I133, I134, I135, I136, ' + \
51      'Xe133, Xe135, ' + \
52      'Cs134, Cs137, Cs138, ' + \
53      'Ce141, ' + \
54      'Fict'

56      ECs = ['EC1', 'EC2', 'EC3', 'EC4', 'EC5', 'EC6', 'EC7', 'EC8']

58      for EC in range(8):
59          valores = (data_atual, hora_atual, \
60      d[0][EC] , d[1][EC] , d[2][EC] , d[3][EC] , d[4][EC] , d[5][EC] , \
61      d[6][EC] , d[7][EC] , d[8][EC] , d[9][EC], d[10][EC], d[11][EC], \
62      d[12][EC], d[13][EC], d[14][EC], d[15][EC], d[16][EC], d[17][EC], \
63      d[18][EC], d[19][EC], d[20][EC], d[21][EC], d[22][EC], d[23][EC], \
64      d[24][EC], d[25][EC], d[26][EC], d[27][EC], d[28][EC], d[29][EC], \
65      d[30][EC], d[31][EC], d[32][EC], d[33][EC], d[34][EC], d[35][EC], \
66      d[36][EC], d[37][EC], d[38][EC], d[39][EC], d[40][EC], d[41][EC], \
67      d[42][EC], d[43][EC], d[44][EC], d[45][EC], d[46][EC], d[47][EC], \
68      d[48][EC], d[49][EC], d[50][EC])

70      cursor.execute("""INSERT INTO """ + ECs[EC] + """(data, hora, """ + \
71      nuclideos + """) VALUES(
72      ?, ?,
73      ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
74      ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
75      ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?
76      );""", valores)

78      conexao.commit()
79      conexao.close()
```

Código 13: SendDataToConcentracoesIsotopicas.py