UNIVERSIDADE FEDERAL DE MINAS GERAIS
Departamento de Engenharia Nuclear
Programa de Pós-Graduação em Ciências e Técnicas Nucleares

Wilmer Aruquipa Coloma

Estudo do padrão de recarga de um reator PWR com a arquitetura *Transformers*

Belo Horizonte
2022

Wilmer Aruquipa Coloma

Estudo do padrão de recarga de um reator PWR com a arquitetura *Transformers*

Versão final

Tese apresentada ao Programa de Pós-Graduação em Ciências e Técnicas Nucleares da Universidade Federal de Minas Gerais, como requisito parcial para à obtenção do título de Doutor em Ciências e Técnicas Nucleares.

Área de Concentração: Engenharia Nuclear e da Energia.
Orientador: Prof. Dr. Clarysson Alberto Mello da Silva
Co-orientadora: Profa. Dra. Claubia Pereira Bezerra Lima

Belo Horizonte
2022

Coloma, Wilmer Aruquipa.
C718e Estudo do padrão de recarga de um reator PWR utilizando a arquitetura *Transformers* [recurso eletrônico] / Wilmer Aruquipa Coloma. - 2022.
1 recurso online (113 f. : il., color.) : pdf.

Orientador: Clarysson Alberto Mello Silva.
Coorientadora: Claubia Pereira Bezerra Lima.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Anexos: f. 110-114.
Bibliografia: f. 104-108.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia nuclear - Teses. 2. Reatores de água pressurizada – Teses. 3. Seção de choque (Física Nuclear) – Teses. I. Silva, Clarysson Alberto Mello. II. Lima, Claubia Pereira Bezerra. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.039(043)

Ficha catalográfica elaborada pela Bibliotecária Letícia Alves Vieira - CRB-6/2337
Biblioteca Prof. Mário Werneck - Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS E TÉCNICAS NUCLEARES

PCTN

FOLHA DE APROVAÇÃO

ESTUDO DO PADRÃO DE RECARGA DE UM REATOR PWR UTILIZANDO A ARQUITETURA TRANSFORMERS

WILMER ARUQUIPA COLOMA

Tese submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em CIÊNCIAS E TÉCNICAS NUCLEARES, como requisito parcial para obtenção do grau de Doutor em CIÊNCIAS E TÉCNICAS NUCLEARES, área de concentração ENGENHARIA NUCLEAR E DA ENERGIA.

Aprovada em 26 de abril de 2022, pela banca constituída pelos membros:

Prof. Clarysson Alberto Mello da Silva - Orientador
Departamento de Engenharia Nuclear - UFMG

Prof(a). Claubia Pereira Bezerra Lima - Coorientdora
Departamento de Engenharia Nuclear - UFMG

Prof. Rochkhudson Batista de Faria
Universidade Federal do Espírito Santo

Dra. Patrícia Amélia de Lima Reis
Departamento de Engenharia Nuclear - UFMG

Dr. Fabiano Cardoso da Silva
UFMG

Prof. Dr. Giovanni Laranjo de Stefani
DNC - Escola Politécnica - UFRJ
SIAPE 3208356

Prof. Giovanni Laranjo de Stefani
Universidade Federal do Rio de Janeiro

Belo Horizonte, 26 de abril de 2022.

Este trabalho é dedicado a minha mãe, meu pai, minha irmã e a Lorena pela confiança que sempre depositaram em mim.

Agradecimentos

Agradeço a meus pais e irmã, Hilda, Leandro e Evelina, pelo apoio emocional, econômico, e muito mais, ao longo de minha vida e formação acadêmica; é por eles que consegui chegar até aqui.

A minha namorada Lorena por me apoiar e animar nos momentos que mais precisei.

A meu orientador Prof. Dr. Clarysson, pelo seu tempo e paciência.

A minha coorientadora Profa. Dr. Claubia pela ajuda, tempo e a disposição alongo da pós-graduação.

Aos membros da banca por seu tempo e valiosa contribuição.

A meus amigos Mario, Mabel, Kauan, Michel, Rodrigo, Eliane e Jéssica. E em especial a minha amiga Natália que me mostrou que no mundo existem pessoas com muita bondade.

A CNPq agradeço pela concessão da bolsa que permitiu minha dedicação integral aos estudos e à pesquisa.

Resumo

O problema de otimização da recarga do núcleo do reator de água pressurizada (PWR), embora seja facilmente explicado, está longe de ser facilmente resolvido. A tarefa do projetista é identificar o *layout* do combustível fresco, do parcialmente queimado e dos venenos queimáveis no núcleo do reator. Esta metodologia otimiza o desempenho do reator durante o ciclo operacional, até que seja necessário novamente reabastecê-lo, ao mesmo tempo garantindo que várias restrições de segurança sejam sempre atendidas. Neste trabalho, é abordado o estudo do padrão de recarga, a fim de desenvolver uma possível metodologia para resolver o problema de otimização, fazendo uso de algoritmos de *Maching Learnig* como a arquitetura *tranformers*. O objetivo é obter um padrão de recarga satisfatória, com relação à eficiência dos elementos combustíveis, dentro dos parâmetros impostos. Para o estudo foi considerado o benchmark BEAVRS, que especifica os detalhes de funcionamento, geometria, composição, etc. de um reator PWR da Westinghouse. Para atingir o objetivo, a metodologia adotada consiste em simular o núcleo do reator com os códigos WIMS-ANL e PARCS. Estes dois códigos são utilizados em conjunto, onde o código WIMS-ANL é utilizado para obter as seções de choque macroscópicas, que são posteriormente utilizadas no código PARCS. A simulação neutrônica foi feita com duas finalidades. A primeira foi validar a metodologia utilizada na simulação neutrônica. Essa validação permite verificar se o modelo é consistente com os dados reais auxiliando o desenvolvimento de outras simulações com dados confiáveis. A segunda finalidade foi para gerar um conjunto de dados de treinamento para o modelo usado neste trabalho. Com a simulação neutrônica foram obtidos 40000 conjuntos de dados de padrões de recarga e distribuições de potência, que foram utilizados no treinamento do modelo. A metodologia utilizada consistiu em propor uma distribuição de potência considerada "ideal" com os requisitos da teoria e, em seguida, utilizá-la como entrada no modelo para que retorne o padrão de recarga correspondente a essa distribuição de potência proposta. A ideia principal foi desenvolver um modelo baseado na arquitetura *transformer* que recebe uma distribuição de potência como entrada e retorna o padrão de recarga como saída. O modelo foi treinado com 40000 conjuntos de dados, usando a plataforma *GoogleColab* que levou aproximadamente 13 horas em finalizar o processo. Os resultados do modelo ofereceram resultados satisfatórios para distribuições de potência que foram geradas de forma aleatória. E para a distribuição de potência considerado "ideal" se conseguiu uma distribuição de potência que atende as restrições usadas neste trabalho.

Palavras-chave: Otimização da recarga, PWR, Seção de choque ,*Arquitetura Transformer*.

Abstract

The pressurized water reactor (PWR) core recharge optimization problem, while easily explained, is far from being easily solved. The designer's task is to identify the layout of fresh fuel, partially burned fuel, and burnable poisons in the reactor core. This methodology optimizes the performance of the reactor during the operational cycle, until it needs to be refilled again, while ensuring that various safety constraints are always met. In this work, the study of the recharge pattern is approached, with the objective of developing a possible methodology to solve the optimization problem, making use of Maching Learn algorithms such as the transformers architecture. The objective is to obtain a satisfactory recharge pattern, with respect to the efficiency of the fuel assemblies, within the imposed parameters. For the study, the BEAVRS benchmark was considered, which specifies the details of operation, geometry, composition, etc. from a Westinghouse PWR reactor. To achieve the objective, the adopted methodology consists of simulating the reactor core with the WIMS-ANL and PARCS codes. These two codes are used together, where the WIMS-ANL code is used to obtain the macroscopic cross-sections, which are later used in the PARCS code. The neutronic simulation was performed for two purposes. The first was to validate the methodology used in the neutronic simulation. This validation allows verifying if the model is consistent with the real data, helping the development of other simulations with reliable data. The second purpose was to generate a training dataset for the model used in this work. With the neutron simulation, 40000 datasets of recharge patterns and power distributions were obtained, which were used in the training of the model. The methodology used consisted of proposing a power distribution considered "ideal" with the requirements of the theory and then using it as an input in the model to return the recharge pattern corresponding to this proposed power distribution. The main idea was to develop a model based on the transformer architecture that receives a power distribution as an input and returns the recharge pattern as an output. The model was trained with 40000 datasets, using the GoogleColab platform, which took approximately 13 hours to complete the process. The model results offered satisfactory results for power distributions that were generated randomly. And for the power distribution considered "ideal", a power distribution that meets the restrictions used in this work was achieved.

Keywords: Reload optimization, PWR, Cross section, *Transformer architecture.*

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

BP	*Back Propagation*
EC	Elemento combustível
FFP	Fator de pico de potência
HFP	*Hot Full Power*
HZP	*Hot Zero Power*
MLP	*Multilayer Perceptron*
PWR	*Pressurized Water Reactor*
RNA	Rede neuronal artificial
VQ	Venenos queimáveis

Sumário

Parte I

Introdução

Capítulo 1

Introdução

Um reator nuclear de água pressurizada (PWR), é reabastecido por lotes, e normalmente opera entre doze e vinte e quatro meses entre as recargas (SECKER et al., 2005). Em teoria, um reator pode funcionar até que o fator de multiplicação seja menor igual a 1 quando ele se torna subcrítico. No entanto, é normal que a recarga de um reator seja realizado em uma parada programada.

A primeira etapa do gerenciamento da recarga, corresponde ao grupo de programação de funcionamento da usina, quem estabelece a janela de interrupção planejada para essa recarga e geram requisitos para o próximo lote em termos de requisitos de energia previstas. O grupo de aquisição de combustível irá gerar requisitos em termos de economia da usina, necessidade de energia e combustível disponível para a recarga. O grupo de física de reatores usam essas informações para gerar o melhor padrão de recarga possível, em um processo que envolve varias simulações. Uma vez que cada grupo esteja satisfeito com a solução proposta, uma simulação detalhada é realizada para fins regulatórios, e o regulador espera evidências de que o padrão de recarga proposto está dentro dos parâmetros operacionais seguros (GLASSTONE, 1994).

O principal objetivo do gerenciador de combustível é minimizar os custos com combustível, reduzindo assim ônus de geração de energia elétrica. Nesse contexto, procura-se atingir esse objetivo alterando os parâmetros permitidos sem violar nenhuma das restrições impostas por considerações operacionais e de segurança (KLERK et al., 1997).

Uma das atividades de gerenciamento de combustível, é o padrão de recarga do núcleo. O gerenciador de combustível deve decidir qual é o “melhor” padrão de recarga dos elementos combustíveis no núcleo, para que as restrições impostas a um determinado ciclo possam ser satisfeitas. As restrições de segurança são sempre as mesmas e estão presentes para todos os ciclos. Ciclos particulares, entretanto, podem ter restrições especiais; como o combustível deve atingir uma certa queima ou o ciclo deve ter uma certa duração (KALIVAS, 1995).

A determinação do melhor padrão de recarga do núcleo é um problema de otimização combinatória rígidos e restritos (KLERK et al., 1997). As características desse problema incluem um número extremamente grande de soluções possíveis, alta dimensionalidade, múltiplos mínimos locais, natureza não linear e regiões viáveis desconectadas. Para um típico PWR, constituído por 121 elementos combustíveis, o número de padrões de recarga possíveis é de cerca de $8,09 \times 10^{200}$, mas se for considerado uma simetria de 1/8 o número de padrões possíveis se reduz a 10^{25} (NICOLAU; SCHIRRU; Monteiro De Lima, 2012). A falta de informação e a alto custo computacional para avaliação de cada solução viável, agrava ainda mais a complexidade deste problema.

Portanto, é um dos problemas mais desafiadores na engenharia de reatores nucleares. Existem muitos métodos de otimização nos quais foram implementados na otimização do padrão de recarga durante as últimas décadas, como *backward diffusion calculation* (CHAO; HU; SUO, 1986), *simulated annealing* (ŠMUC; PEVEC; PETROVIĆ, 1994; PARK et al., 2009), algoritmos genéticos (YAMAMOTO, 1997; DO; NGUYEN, 2007; ALIM; IVANOV; LEVINE, 2008), redes neurais artificiais (FARIA; PEREIRA, 2003; ORTIZ; REQUENA, 2004), *integer programming* (SI, 2002), *characteristic statistic algorithm* (HU et al., 2006), *estimation of distribution algorithms* (JIANG et al., 2006), *harmony search algorithm* (POURSALEHI; ZOLFAGHARI; MINUCHEHR, 2013a), *differential harmony search algorithm* (POURSALEHI; ZOLFAGHARI; MINUCHEHR, 2013b), *evolutionary harmony search algorithm* (POURSALEHI; ZOLFAGHARI; MINUCHEHR, 2012), *particle swarm optimization* (BABAZADEH; BOROUSHAKI; LUCAS, 2009), *continues firefly algorithm* (POURSALEHI; ZOLFAGHARI; MINUCHEHR, 2013c), *bat algorithm* (KASHI et al., 2014), *ttabu search* (CASTILLO et al., 2007), etc.

Tendo em vista os problemas de otimização que existem; o objetivo principal deste trabalho, é desenvolver uma estratégia de recarga para um reator PWR usando a arquitetura de *machine learning* conhecido como *transformer*. O algoritmo é treinado com informações sobre o desempenho de um conjunto de configurações espaciais dos elementos combustíveis, utilizando-as posteriormente para gerar novas configurações. Os resultados do treinamento, serão avaliadas de acordo com o desempenho dos parâmetros obtidos por meio de simulações com códigos neutrônicos, neste caso em particular, com os códigos WIMS-ANL 5.08 (DEEN et al., 2004) e PARCS 2.4 (T. Downar, Y. Xu, 2006).

Como em todas as análises de simulação, primeiro deve-se validar o modelo e os resultados da simulação do reator. Nesse sentido, baseado no documento de referência BEAVRS (HORELIK et al., 2013), simula-se um reator PWR da Westinghouse em duas etapas. Simula-se um reator PWR em duas etapas. Na primeira fase, serão calculadas as seções de choque macroscópicas dos elementos combustíveis com o código WIM-AN; utilizando-se o modelo textitMULTICELL (DEEN et al., 2004). Na segunda etapa, as seções de choque obtidas serão usadas para simular o núcleo completo com o código

PARCS; estas simulações são feitas para o estado *Hot Zero Power* e para a queima do primeiro ciclo do reator.

Os *transformers* foram apresentados pelo artigo *"Attention Is All You Need"* (VASWANI et al., 2017). Esta é uma arquitetura para a tarefa de processamento de linguagem natural que usa modelos de sequência a sequência com atenção em muitas tarefas. Os *transformers* são uma arquitetura de modelo de sequência a sequência, eles têm um codificador e um componente de decodificador separados. A diferença de outros modelos sequenciais, está no uso de uma forma aprimorada de atenção conhecida como autoatenção, que além de seu poder expressivo tem a vantagem computacional de ser flexível como uma operação matricial. A arquitetura central consiste em uma pilha de codificadores totalmente conectados a uma pilha de decodificadores. Cada codificador consiste em dois blocos: um componente de autoatenção e uma rede de alimentação direta. Cada decodificador consiste em três blocos: um componente de autoatenção, um componente de atenção do codificador-decodificador e um componente de rede neural artificial *feed forward*. Apesar de ser uma arquitetura de processamento de linguagem natural, muitos trabalhos usam esta arquitetura na aplicação de diferentes áreas, como por exemplo a predição de estrutura de proteínas (VASWANI et al., 2017), predição de séries temporais (WU et al., 2020), processamento de imagens (CHEN et al., 2021), etc.

Para utilizar a arquitetura *transformer*, procedeu-se da seguinte forma; após a validação da simulação neutrônica o próximo passo consiste em simular um conjunto de 40000 diferentes padrões de recarga dos elementos combustíveis, gerados de forma aleatória, a fim de obter o conjunto de treinamento para a arquitetura *trasfomer*. Uma vez que o modelo foi treinado, dado uma distribuição de potência radial como *input*, espera-se que esse modelo seja capaz de predizer a configuração espacial dos elementos combustíveis para essa distribuição de potência. Nesse sentido, propõe-se uma distribuição de potencial radial considerado "ideal", para que o modelo retorne o padrão da distribuição espacial dos elementos combustíveis. A distribuição de potência, considerada "ideal", deverá cumprir as seguintes restrições: (a) o fator de pico de potência que correspondente à distribuição de potência, não deve ultrapassar um limite estabelecido; (b) a redução da fuga de nêutrons pela periferia do núcleo; (c) o achatamento da distribuição de potência na parte central do núcleo. A medida que os capítulos são apresentados, cada uma das explicações aprestadas nesta seção, serão detalhadas.

Parte II

Estado da arte

Capítulo 2

Gerenciamento do Combustível no Núcleo do Reator e Otimização

A fonte de energia de um reator é o combustível nuclear que está em seu núcleo, onde a energia é gerada a partir das fissões. Os elementos combustíveis representam um custo de investimento recorrente para o operador da planta. Ao contrário de outras fontes de energia, há implicações mais amplas para seu uso do que apenas a otimização econômica. Após a queima, o combustível nuclear apresenta alta atividade, a garantia de contenção segura dessa radioatividade é a principal preocupação do licenciamento da operação de usinas nucleares.

O gerenciamento de combustível no núcleo é uma análise o qual possibilita assegurar que: (1) o núcleo seja capaz de produzir a energia necessária continuamente durante o ciclo de operação do reator; (2) o núcleo opere dentro dos limites da segurança permitida (3) o inventário isotópico seja conhecido ao longo e no final do ciclo e (4) o padrão de recarga minimize os custos totais (IAEA, 2020). Assim, o principal objetivo do gerenciador de combustível é minimizar os custos com combustível, reduzindo assim o custo de geração de energia elétrica. O responsável por esta tarefa tenta atingir este objetivo alterando os parâmetros permissíveis sem violar nenhuma das restrições impostas por considerações operacionais e de segurança (IAEA, 1992). Algumas destas restrições são:

- Maximização da queima do combustível;
- Redução do tamanho do lote de alimentação;
- Recarga de baixa fuga, que estende a vida útil do vaso de pressão, minimizando a fuga de nêutrons do núcleo;
- Redução do enriquecimento, mas garantindo suficiente reatividade durante o ciclo;
- Minimização do fator de pico de potência (FPP) e outras métricas para obter um perfil de temperatura do refrigerante mais uniforme;

- Minimização do número dos Venenos Queimáveis (VQ), devido ao possível envenenamento residual de isótopos de alta captura de nêutrons que se acumulam durante a queima.

2.1 Recarga de combustível em PWR

O problema de otimização do núcleo de recarga do reator de água pressurizada, embora seja facilmente mencionado, está longe de ser facilmente resolvido. A tarefa do projetista é identificar a disposição do combustível fresco e parcialmente queimado e dos venenos queimáveis (material de controle) dentro do núcleo, que otimiza o desempenho do reator durante aquele ciclo operacional, até que a recarga seja novamente necessária.

Um núcleo PWR típico contém, dependendo de sua potência térmica nominal, entre 121 e 241 elementos combustíveis dispostos com simetria de um quarto ou um oitavo de núcleo (reflexivo ou rotacional). Em cada recarga, entre um terço ou um quarto desses conjuntos podem ser substituídos por um lote de combustível novo. Para obter uma melhor utilização do combustível, um determinado lote pode ser subdividido de acordo com os diferentes enriquecimentos. É uma prática comum que (alguns dos) elementos combustíveis possuam varetas de venenos queimáveis, dos quais podem haver diferentes números e/ou concentração de isótopos absorvedores de nêutrons. Também é comum embaralhar (alguns) os elementos combustíveis queimados restantes para melhorar as características do novo núcleo. Esta combinação pode envolver: (a) a troca de conjuntos correspondentes entre os quadrantes, ou (b) a troca de conjuntos diferentes, que também mudam suas localizações e possivelmente suas orientações. Portanto, puramente do ponto de vista de otimização, a busca pelo melhor padrão de carga representa um problema combinatório de grande magnitude.

Os reatores nucleares têm inerentemente algumas características altamente não lineares, o que significa que qualquer que seja a função objetivo (há várias possibilidades), as restrições são usadas para definir e quantificar padrões de carga aceitáveis, algumas dessas variáveis do sistema serão inevitavelmente funções não lineares. Exemplos de tais não linearidades incluem os efeitos de retroalimentação local térmico e hidráulico e a dependência do tempo que resulta da queima de combustível. Particularmente com relação a este último, o gasto computacional associado à análise de uma única solução pode ser substancial. Quando considerado no contexto de dezenas de milhares de soluções, como geralmente é necessário para o uso de rotinas de otimização modernas, o custo do tempo de execução da CPU torna-se muito alto.

Fica evidente que o progresso neste campo de pesquisa sempre esteve intimamente ligado ao do campo da tecnologia da computação. Esta é talvez a principal razão pela qual, embora um progresso substancial tenha sido feito nos últimos anos, os pesquisadores

que buscaram projetar núcleos de recargas "ideais" (desde o ponto de vista das restrições imposta) foram, até agora, forçados a resolver problemas que, em graus variados, não são reais para dia-a-dia de uma usina nuclear, devido às premissas simplificadoras necessárias para facilitar a solução (DOWNAR; SESONSKE, 1988).

Qualquer tentativa de resolver o problema de otimização de recarga do núcleo PWR requer uma combinação de um método de avaliação com uma técnica de otimização. Esse método de combinação deverá possuir velocidade suficiente, para que dezenas de milhares de soluções possam ser examinadas e suficientemente preciso para que os resultados obtidos sejam significativos; e uma técnica de otimização que não requer informações derivadas, não enganada por ótimos locais.

2.2 Avaliação do padrão de recarga

Uma das atividades de gerenciamento de combustível é o padrão de recarga do núcleo. O gerenciador de combustível deve decidir qual é o "melhor" padrão de recarga dos elementos combustíveis, para que as restrições impostas a um determinado ciclo possam ser satisfeitas. As restrições de segurança são sempre as mesmas e estão presentes para todos os ciclos. Entretanto, ciclos particulares podem ter restrições especiais; como por exemplo o combustível deve atingir uma certa queima ou o ciclo deve ter uma certa duração. O projeto de um padrão de recarga envolve decisões sobre a atribuição de elementos combustíveis a locais específicos em um núcleo. Um padrão de recarga para um núcleo, com N locais de combustível envolve N decisões. Portanto, existem N variáveis de decisão para um padrão de recarga. Por exemplo um típico PWR, consistindo por 121 elementos combustíveis,o número de padrões de recarga possíveis é de cerca de $8,09 \times 10^{200}$, mas se for considerado uma simetria de 1/8 o número de padrões possíveis se reduz a 10^{25} (NICOLAU; SCHIRRU; Monteiro De Lima, 2012). A falta de informação e a alto custo computacional para avaliação de cada solução viável, agrava ainda mais a complexidade deste problema.

Pode haver vários candidatos para uma função objetivo. Pode ser, por exemplo, a minimização do enriquecimento do núcleo ou fuga de nêutrons; alternativamente, também pode-se maximizar a potência do ciclo ou k_{eff}. Pode-se notar que essas funções objetivo, são consistentes entre si. Todas elas atendem ao uso eficiente de combustível e à redução da fuga de nêutrons no reator. Suzuki e Kiyose (SUZUKI; KIYOSE, 1971) determinaram que um enriquecimento mínimo do núcleo pode ser obtido para a configuração de recarga, maximizando o fim do ciclo. Huang e Levine (H. Y. Huang, 1978) também mostraram que, se o projeto mecânico do elemento combustível permanecer fixo, o enriquecimento mínimo do núcleo para a configuração do núcleo de recarga pode ser obtido maximizando o final do ciclo. É muito claro que um valor de k_{eff} mais alto, em uma determinada queima

de núcleo, maior potência de ciclo, maior queima do combustível e melhor utilização de combustível são mutuamente consistentes. O conceito de baixa carga de fuga que visa minimizar a fuga de nêutrons é consistente com a maximização da potência do ciclo (DOWNAR; SESONSKE, 1988). De acordo com Downar, a baixa carga de fuga também fornece depreciação acelerada do combustível (DOWNAR, 1986).

O pico de potência é outro parâmetro importante, que está em função de um determinado padrão de recarga. Seu valor deve ser baixo no interesse da integridade do combustível em operação. No entanto, esse requisito é inconsistente com as funções mutuamente consistentes e seus requisitos considerados acima. Assim, temos dois requisitos mutuamente inconsistentes. Portanto, um dos dois deve ser otimizado enquanto o outro deve ser restrito. A maximização é consistente com a aplicação de um limite inferior e vice-versa. Portanto, pode-se maximizar a potência do ciclo ou limitá-lo a um valor inferior, se quiser restringi-lo. Da mesma forma, pode-se minimizar o pico de potência ou limitá-lo a um valor superior (GLASSTONE, 1994).

Dado um conjunto de valores para as variáveis de decisão, um modelo de cálculo é necessário para avaliar a função objetivo, bem como as funções a serem restringidas. No caso de um padrão de recarga, exigimos conhecimento das propriedades nucleares de cada um dos N elementos combustíveis e um modelo para calcular essas funções.

Se assumirmos que cada um dos N elementos combustíveis é diferente, então temos $N!$ possíveis padrões de recarga. Se assumirmos que todos os novos elementos de combustível são idênticos, esse número é reduzido para $N!/r!$ onde r é o número de novos elementos combustíveis. Este ainda é um número extremamente grande. É impraticável constituir e avaliar tantos padrões de recarga antes de selecionar o melhor entre eles. Resolver este problema requer uma abordagem em duas frentes, como o uso de métodos de otimização padrão, e a redução do número de variáveis de decisão.

Uma prática comum na seleção de um padrão de recarga, consistia em dividir o núcleo em regiões anulares concêntricas, onde cada uma possuía um lote diferente de combustível. O combustível com maior queima (menor enriquecimento) era colocado na região central e o novo inserido na periferia do núcleo. Os lotes com queima intermediária eram colocados nas regiões intermediárias. Nesta metodologia, a proximidade do centro do núcleo é decidida com base na quantidade de queima, quanto maior a queima, mais próximo do centro do núcleo o combustível era colocado. A Figura 1 mostra o esquema para um núcleo de três lotes. No momento da recarga, o combustível na região mais interna era descarregado e todos os outros lotes eram movidos em direção ao centro do núcleo. O combustível fresco era carregado na região mais externa (COCHRAN; TSOULFANIDIS, 1999).

A principal vantagem desse padrão era a transferência do pico de potência do centro para as regiões externas do núcleo. A média do pico de potência era reduzida.

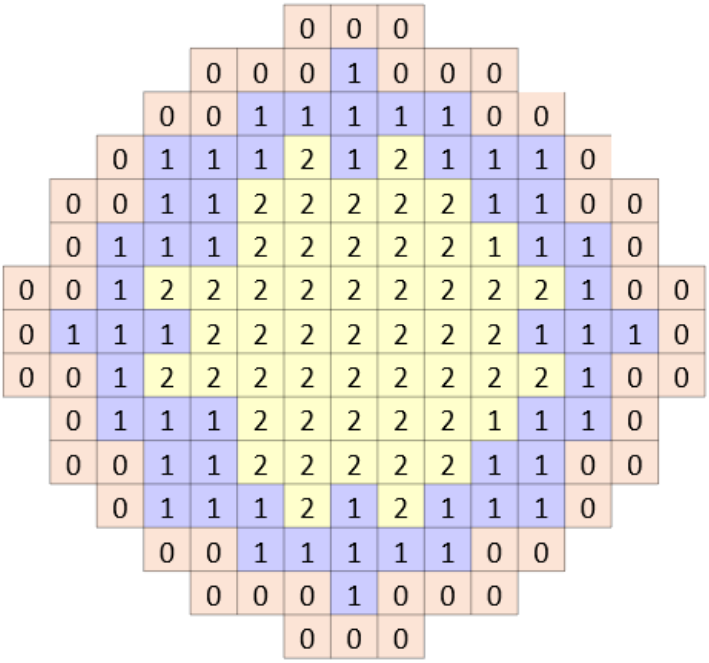


Figura 1 – Esquema de recarga. O núcleo é dividido em três regiões: 0 = combustível fresco; 1 = combustível queimado em um ciclo anterior; e 2 = combustível queimado nos dois ciclos anteriores. Adaptado de (COCHRAN; TSOULFANIDIS, 1999)

Existem duas desvantagens principais associadas a este esquema: Uma era o fato de que o combustível queimado gera, no centro do núcleo, uma potência inferior à média. O segundo era o fato de que os novos conjuntos na periferia do núcleo, ao gerar alta potência em relação à média, também produzia grande número de nêutrons de fissão rápida, uma fração relativamente grande dos quais escapa do núcleo. Esse comportamento não apenas representava uma grande desvantagem quanto a economia de nêutrons, mas também afetava o vaso de pressão, pois diminuía a sua vida útil. Por causa dessas desvantagens, especialmente a segunda, este esquema de recarga não é mais usado.

Existe outra estrategia de recarga conhecido como esquema de “tabuleiro de xadrez”, onde o núcleo do reator é dividido em pequenas regiões, de modo que cada uma consiste de quatro a seis conjuntos de lotes diferentes (Figura 2). Na hora da recarga, os conjuntos com maior queima em cada uma dessas regiões são descarregados e substituídos por novos. A principal vantagem desse esquema é que ele produz uma distribuição de energia bastante uniforme em todo o núcleo. Uma segunda vantagem é que, com a colocação adequada dos novos conjuntos, isso pode levar a um núcleo com baixa fuga de nêutrons. Em princípio, essa estratégia exigiria a movimentação apenas dos conjuntos a serem descarregados, caso em que o tempo de parada para reabastecimento seria reduzido.

Uma desavantajem consiste em que, uma fração dos nêutrons produzidos pela fissão no núcleo escapa e atinge o vaso de pressão, causando danos por radiação. Então, muitas tentativas foram feitas para reduzir o fluxo de nêutrons na posição do vaso de pressão sem ter que reduzir a potência e também sem violar quaisquer limites de segurança. Os métodos

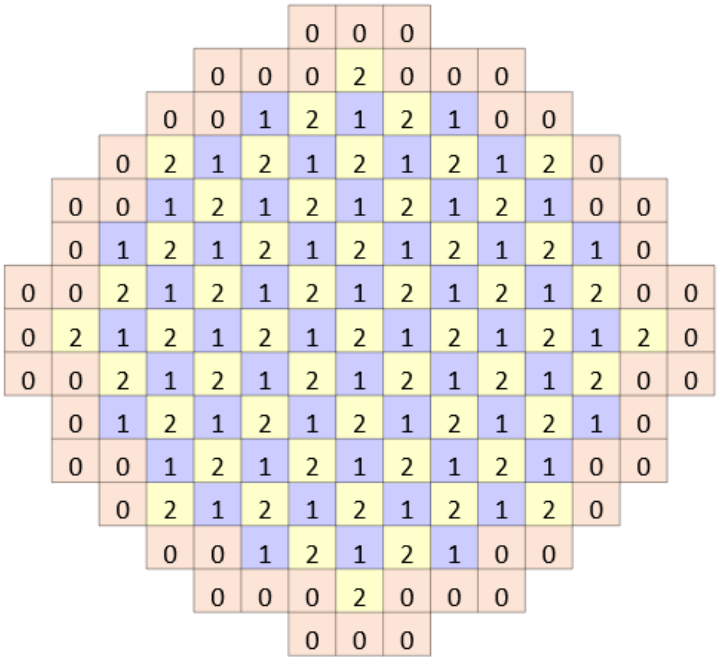


Figura 2 – Esquema de recarga (tabuleiro de xadrez). O núcleo é dividido em três regiões:0 = combustível fresco; 1 = combustível queimado em um ciclo anterior; e 2 = combustível queimado nos dois ciclos anteriores. Adaptado de (COCHRAN; TSOULFANIDIS, 1999).

que foram propostos e parcialmente implementados são baseados em procedimentos como:

- Disposição de apenas conjuntos queimados na periferia do núcleo (Figura 3);
- Realocação de alguns conjuntos situados em posições críticas para outras de menos importância;
- Substituição de alguns conjuntos periféricos por aço inoxidável;
- Disposição de materiais atenuantes, como placas de aço, entre a borda do núcleo e o vaso de pressão;
- Uso de veneno queimável com gadolínio em elementos periféricos próximos ao vaso de pressão;

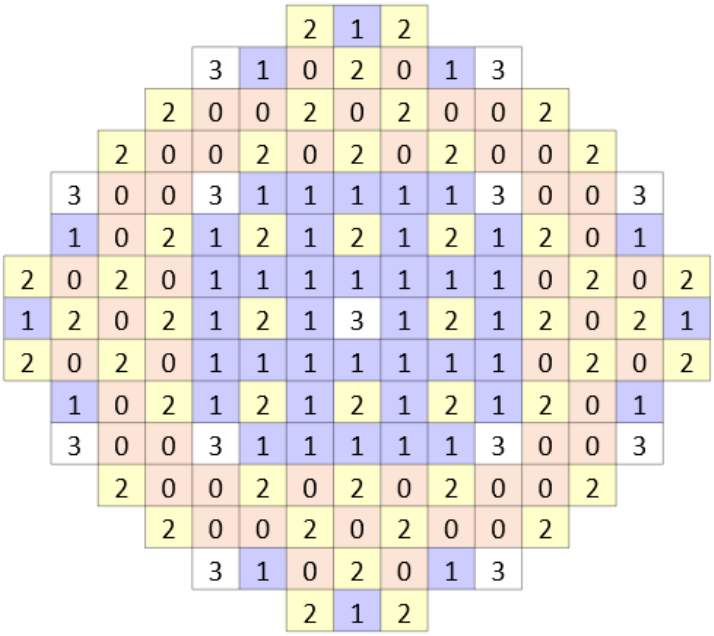


Figura 3 – Recarga do núcleo de baixa fuga: 0 = combustível fresco; 1 = combustível queimado em um ciclo anterior; 2 = combustível queimado nos dois ciclos anteriores; 3 = combustível queimado nos três ciclos anteriores. Aptado de (COCHRAN; TSOULFANIDIS, 1999).

.

Capítulo 3

Benchmark BEAVRS

Esta seção fornece uma visão geral do benchmark BEAVRS (*Benchmark for Evaluation And Validation of Reactor Simulations*) (HORELIK et al., 2013). Ele oferece dados do projeto altamente detalhados para os primeiros dois ciclos de um PWR real, bem como os dados de medição correspondentes. Esses dados básicos de referência publicamente disponíveis, permitem a validação de código sem a necessidade de acessar informações patenteadas. Isso torna o BEAVRS uma escolha ideal para validação e comparação entre diferentes códigos.

O BEAVRS foi publicado em 2013 pelo Grupo *Computational Reactor Physics* do *Massachusetts Institute of Technology* (MIT) e foi atualizado várias vezes. Seu objetivo é permitir a comparação de vários códigos de física de reatores aplicados aos cálculos de núcleo completo com dados reais. A especificação do benchmark contém uma descrição detalhada de uma usina nuclear PWR desconhecida (real) com um reator Westinghouse de 4 circuitos, localizado nos EUA. A documentação contém detalhes das medições de *Hot Zero Power* (HZP) e condições de operação do reator para o primeiro e o segundo ciclos do combustível.

Vários relatórios de pesquisa sobre soluções do BEAVRS foram publicados, com uma grande variedade de metodologias de simulação (PARK et al., 2020; WANG et al., 2018; ELSAWI; HRAIZ, 2015; LI et al., 2016; YU et al., 2020; MOSOEUNYANE, 2019; RYU et al., 2015; LIU et al., 2017; COLLINS et al., 2020). O modelo desenvolvido neste trabalho foi comparado com dados do benchmark disponíveis para o primeiro ciclo de combustível apresentados nas publicações.

3.1 Especificações gerais do BEAVRS

O benchmark BEAVRS apresenta uma planta Westinghouse de 4 loops com informação de detalhes disponíveis na literatura aberta. Seu núcleo possui simetria de um oitavo contendo 193 elementos combustíveis, onde cada conjunto possui uma configuração

Tabela 1 – Resumo dos principais parâmetros do BEAVRS.

Núcleo	
Nº de elementos combustíveis	193
Padrão de carga	w/o U235
Região 1 (ciclo 1)	1.60
Região 2 (ciclo 1)	2.40
Região 3 (ciclo 1)	3.10
Região 4A (ciclo 2)	3.20
Região 4B (ciclo 2)	3.40
Elementos combustíveis	
Configuração de rede dos elementos combustíveis	17 x17
Comprimento do combustível ativo	365,76 cm
Nº de varetas de combustível	264
Nº de separadores de grade	8
Controle	
Material da barra de controle (região superior)	B4C
Material da barra de controle (região inferior)	Ag-In-Cd
Nº de bancos de barras de controle	57
Nº de barras de venenos queimáveis	1266
Material dos venenos queimáveis	Vidrode borossilicato, 12,5 w/o B2O3
Desempenho	
Potência	3411 MWth
Pressão de operação	2250 psia

de 17×17 varetas combustíveis. A potência média é de 3411 MWth e a usina opera a uma pressão de 2.250psia (Ver Tabela 1). No ciclo 1, os elementos combustíveis têm enriquecimento de $1,6\%$, $2,4\%$ ou $3,1\%$. Um subconjunto desses conjuntos é embaralhado para o ciclo 2 e novos conjuntos de enriquecimentos de $3,2\%$ ou $3,4\%$ também são introduzidos. Os padrões de recarga para o ciclo 1 e o ciclo 2 são dados nas Figuras 5 e 6, respectivamente. Além disso, cada elemento combustível contém um número variável de varetas de venenos queimáveis (VQs). A configuração dos VQs para o ciclo 1 é dada na figura 4.

O reator apresentado no BEAVRS contém 58 elementos combustíveis com detetores, os quais estão localizados nos tubos guia centrais. A Figura 4 mostra essas posições. Quando as medições são feitas, os detectores deslocam-se verticalmente em cada um dos 58 conjuntos para garantir leituras precisas. Antes das medições, os detectores são calibrados de modo que eles passam por um mesmo elemento combustível para normalização do sinal. Os detectores são inseridos pela parte inferior através de tubos de instrumentação até atingirem o topo do conjunto. Os detectores são então puxados para trás e medem a taxa de fissão axial em 61 localizações axiais. As taxas de fissão axial podem ser normalizadas e integradas para gerar um mapa radial das medições do detector. Os detalhes exatos desse processamento de dados são apresentados na referência (HORELIK et al., 2013).

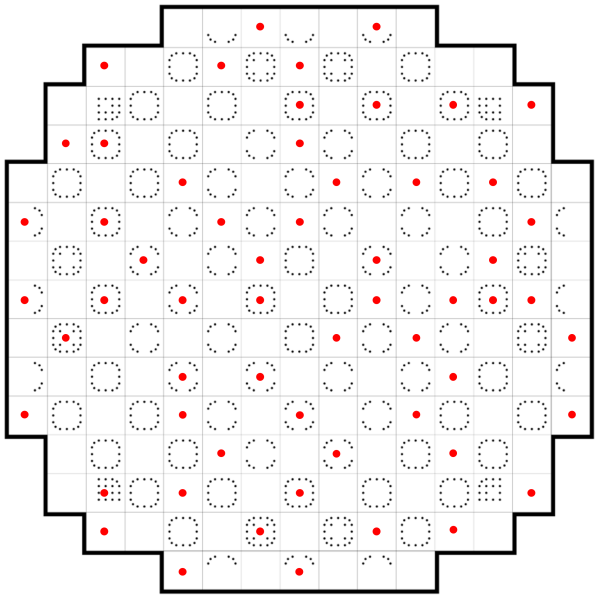


Figura 4 – Vista em escala detalhada dos venenos queimáveis no ciclo 1, mostrando as rotações adequadas (Pontos pretos). Os pontos vermelhos representam a localização dos detetores em cada elemento combustível (HORELIK et al., 2013).

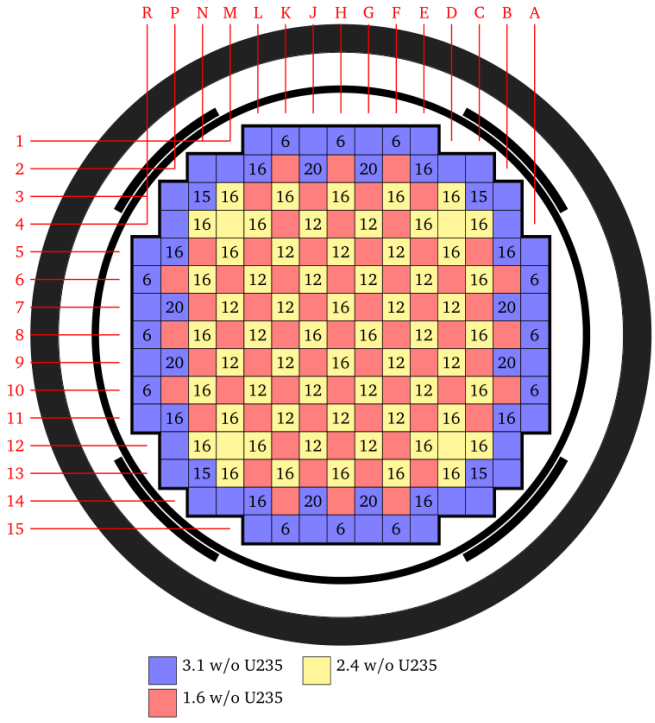


Figura 5 – Disposição dos elementos combustíveis, mostrando o padrão de recarga com os respectivos enriquecimento no ciclo 1. Os números dentro das células representam o número de venenos queimáveis (HORELIK et al., 2013).

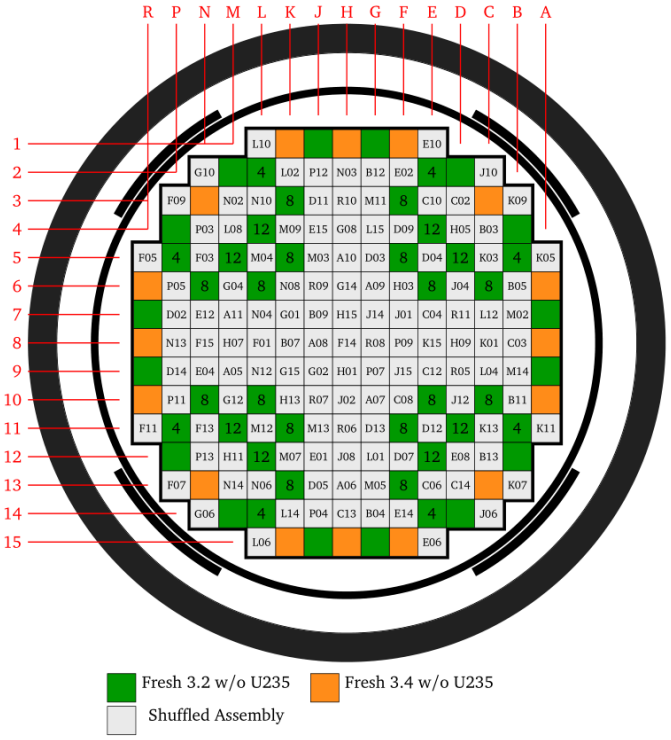


Figura 6 – Padrão de recarga do ciclo 2 (HORELIK et al., 2013).

3.2 Geometria radial

Nesta seção, os parâmetros radiais de cada tipo de vareta combustível usados nos elementos combustíveis são detalhados. Cada um deles descreve uma vareta cercada pelo refrigerante principal.

3.2.1 Varetas

O primeiro, e o nível mais básico da geometria do núcleo são as varetas que são definidos nesta seção.

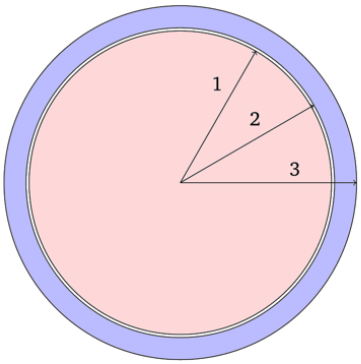
Varetas de combustível: Existem cinco tipos básicos de varetas de combustível no núcleo, dependendo do enriquecimento de urânio. No entanto, sua geometria é a mesma. A geometria da vareta de combustível é apresentada na figura 7.

Tubo guia: A parte superior e inferior do tubo guia tem geometria ligeiramente diferente. Apenas para simplificar, uma geometria única foi aplicada (ver Figura 8).

Venenos queimáveis: Os venenos queimáveis (Figura 9) são inseridas nos tubos-guia em cada elemento combustível selecionado.

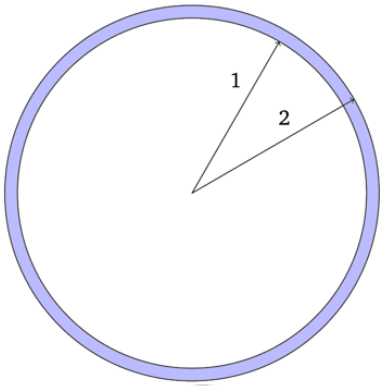
Barras de controle: Os bancos de barras de controle contêm 24 barras de Ag–In–Cd ou SS-304 como absorvedores de nêutrons, que são inseridos nos tubos guia dos ECs para o controle de reatividade ou para o desligamento do reator. As barras de controle são divididas em duas regiões; uma parte superior composta por SS304 (ver Figura 10) e

uma parte inferior composta por Ag-In-Cd (ver Figura 11).



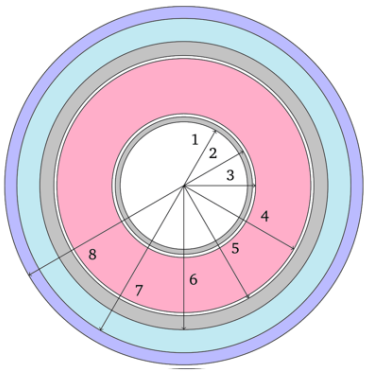
Região	Raio [cm]	Material
1	0,39218	Combustível
2	0,40005	Hélio
3	0,45720	Zircaloy

Figura 7 – Esquema da vareta de combustível UO_2 (HORELIK et al., 2013)



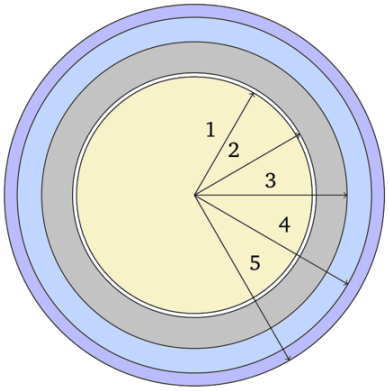
Região	Raio [cm]	Material
1	0,56134	Água
2	0,60198	Zircaloy

Figura 8 – Esquema da vareta de tubo guia (HORELIK et al., 2013).



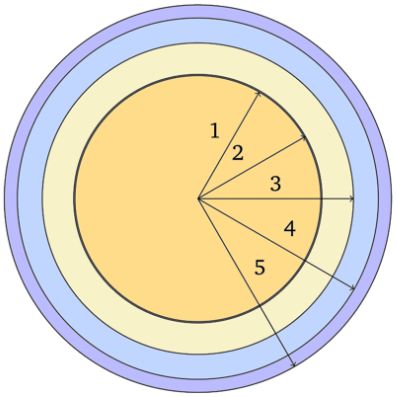
Região	Raio [cm]	Material
1	0,21400	Ar
2	0,23051	SS304
3	0,24130	Hélio
4	0,42672	Borosilicate Glass
5	0,43688	Hélio
6	0,48387	SS304
7	0,56134	Água
8	0,60198	Zircaloy

Figura 9 – Configuração do tubo guia contendo veneno queimável (HORELIK et al., 2013).



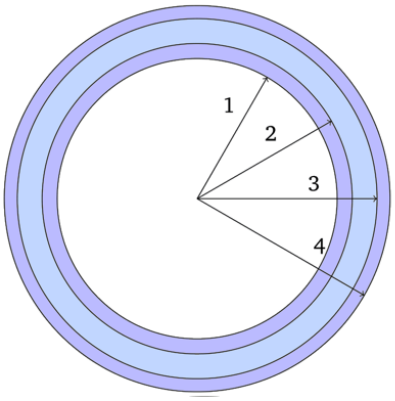
Região	Raio [cm]	Material
1	0,37338	B4C
2	0,38608	Hélio
3	0,48387	SS304
4	0,56134	Água
5	0,60198	Zircaloy

Figura 10 – Esquema da barra de controle da região superior (HORELIK et al., 2013).



Região	Raio [cm]	Material
1	0,38227	Ag-In-Cd
2	0,38608	Hélio
3	0,48387	SS304
4	0,56134	Água
5	0,60198	Zircaloy

Figura 11 – Esquema da barra de controle da região inferior (HORELIK et al., 2013).

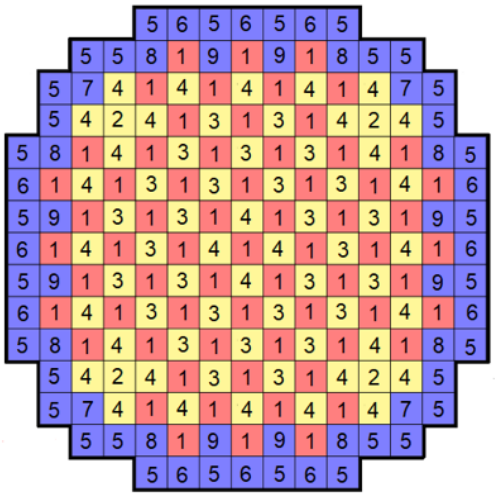


Região	Raio [cm]	Material
1	0,43688	Ar
2	0,48387	Zircaloy
3	0,56134	Água
4	0,60198	Zircaloy

Figura 12 – Esquema do tubo de instrumentação (HORELIK et al., 2013).

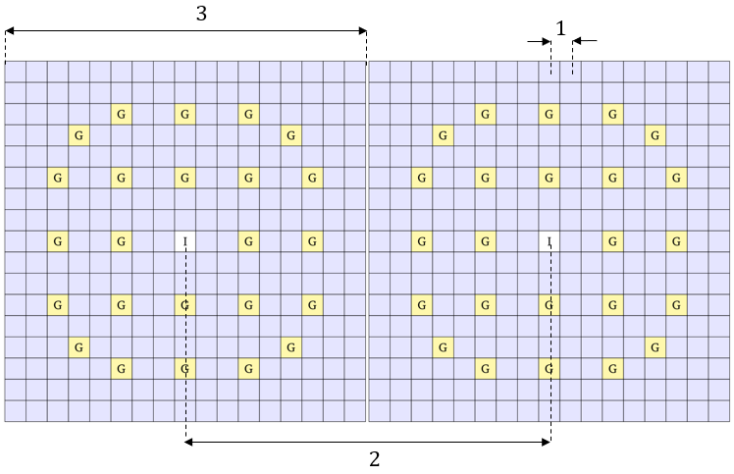
3.2.2 Elementos combustíveis (ECs)

A geometria básica dos ECs é apresentada na figura 14. Um elemento combustível é uma rede de várias combinações de varetas de combustível, barras de controle, venenos queimáveis e tubos guia. As posições dos tubos guia podem ser preenchidas com uma das várias configurações de venenos queimáveis (ver Figura 4). Para qualquer configuração, o tubo guia central pode conter um tubo de instrumentação (ver Figura 4). Para o primeiro ciclo de combustível, existem nove tipos de elementos combustíveis com uma estrutura



ID	EC	w/o U235	N° VQ
1	EC 1	1,6 %	0
2	EC 2	2,40 %	0
3	EC 3	2,40 %	12
4	EC 4	2,40 %	16
5	EC 5	3,10 %	0
6	EC 6	3,10 %	6
7	EC 7	3,10 %	15
8	EC 8	3,10 %	16
9	EC 9	3,10 %	20

Figura 13 – Nomenclatura usada neste trabalho para identificar os ECs. Onde ID são as etiquetas, w/o enriquecimento e N° VQ é o numero de venenos queimáveis no EC (Adaptado de (HORELIK et al., 2013)).



N°	Descrição	Dimensão [cm]
1	Pitch da malha dos pinos	1,259840
2	Dimensão do EC	21,50364
3	Pitch do EC	21,50364

Figura 14 – Diagrama da geometria básica do EC (Adaptado de (HORELIK et al., 2013)).

padrão 17×17 (ver Figura 15). Neste trabalho, para distinguir os elementos combustíveis (EC) uns dos outros, será usado a nomenclatura apresentada na figura 13.

O reator tem quatro bancos de barras de controle especificados pelos identificadores A, B, C e D (ver Figura 16). Cada tubo guia pode ser preenchido com a barra de controle descrita na seção 3.2.1, com exceção do tubo central.

Além dos bancos de barras de controle A, B, C e D, há 5 bancos de barras de desligamento, especificados por S_A, S_B, S_C, S_D e S_E, os quais estão posicionadas acima do núcleo do reator e não são usados em condições de operação normal da usina. A figura 16 mostra as localizações radiais dos conjuntos de barras de controle pertencentes a cada barra de controle e banco de desligamento. As especificações axiais de cada um são descritas posteriormente na seção 3.3. As posições dos bancos de barras de controle não mudam entre o ciclo 1 e o ciclo 2.

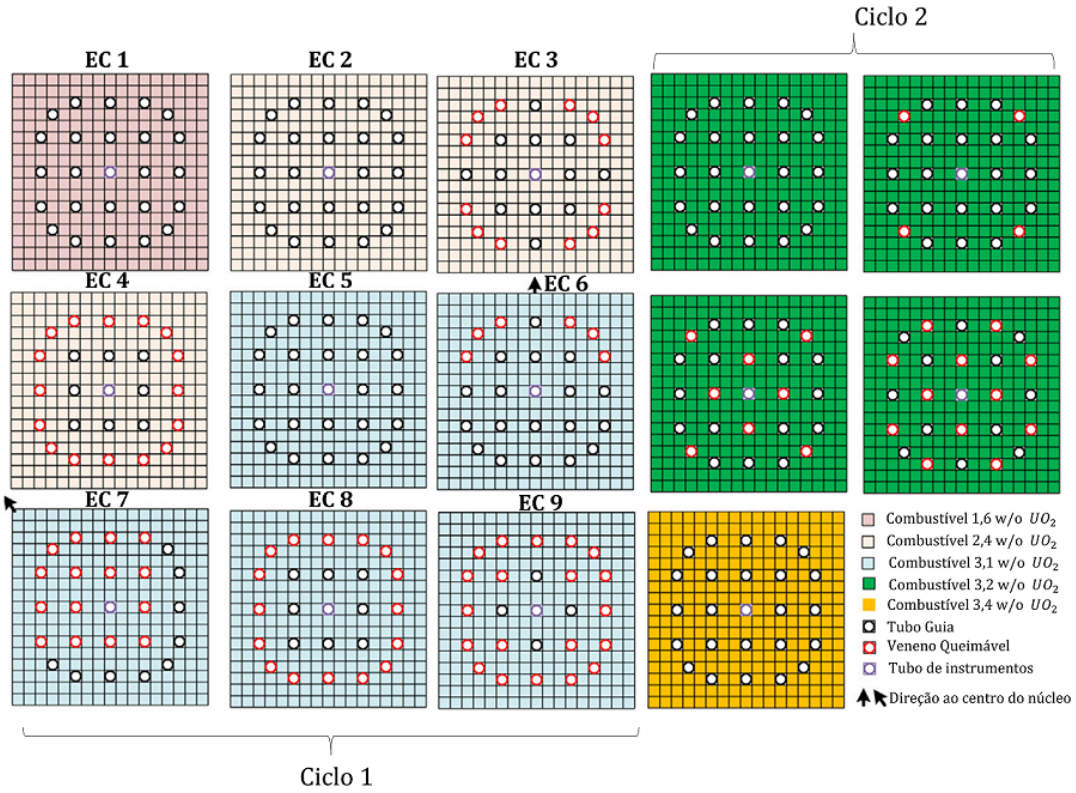


Figura 15 – Especificação dos ECs do BEAVRS para o ciclo 1 e ciclo 2 (Adaptado de (PARK et al., 2020)

3.3 Geometria axial

Os ECs e as varetas de combustível são caracterizados pela zona axial e pela distribuição diferenciada do material ao longo do comprimento (Figura 17). O conjunto de elementos combustíveis se estendem entre a parte superior (elevação 460 cm) e a parte inferior (elevação 0 cm). O comprimento da parte ativa do combustível é 365,76 cm (Figura 17). O esquema apresentado na figura 17 é um modelo simplificado da parte axial dos elementos combustíveis. Esta simplificação é feita devido a que neste trabalho modela-se a parte ativa do núcleo.

Para facilitar a modelagem, neste trabalho não se considerou a presença das grades espaçadoras; pois a sua influência não é significativa para os parâmetros avaliados.

A figura 17 mostra o layout axial da barra de controle, o qual depende do com-

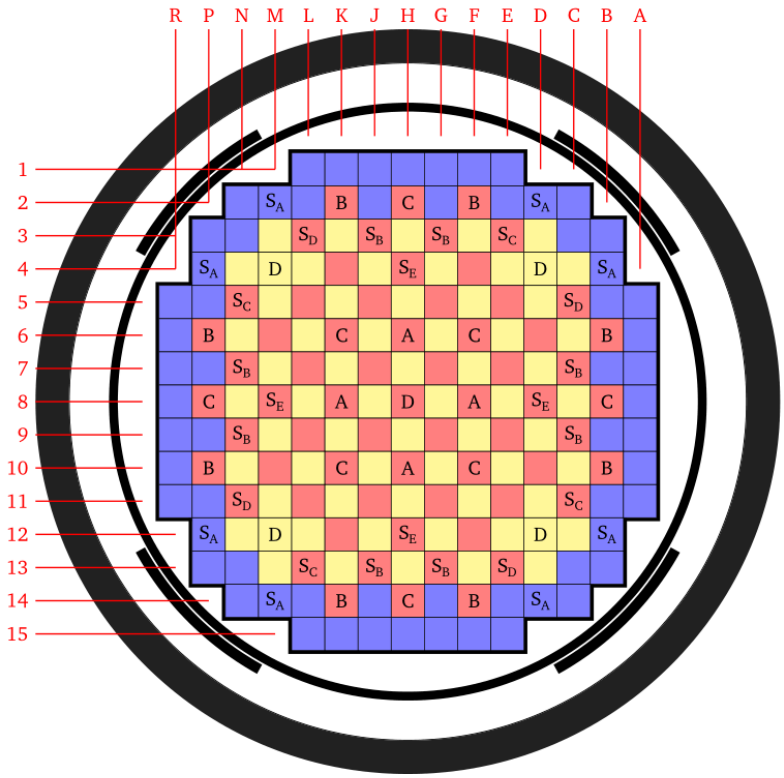


Figura 16 – Posições dos bancos de barras de controle e de desligamento (HORELIK et al., 2013).

primento de inserção dentro do tubo guia vazio, e os detalhes dependem da localização do conjunto de barras de controle. As barras de controle estão completamente extraídas em 228 “steps” até que a região ativa da barra de controle esteja completamente retirada da região ativa do combustível. Quando retirado os 228 steps, a parte inferior da região ativa da barra de controle, deve estar nivelada com a parte superior da região ativa do combustível. As barras de controle são divididas em duas regiões, uma inferior e outra superior, essas duas regiões diferem tanto no material quanto na geometria radial (Ver figura 10 e 11).A altura total das barras de controle é 360,68 cm, o que significa que cada “step” é 1,58193 cm.

Os bancos de controle e desligamento podem ter qualquer nível de inserção parcial, onde as pontas inferiores das barras podem estar em qualquer nível axial entre o step 0 e o step 228. Embora cada um desses bancos mova o conjunto de barras de controle, seu movimento é frequentemente escalonado com os outros bancos de barras de controle.

3.4 Dados operacionais

A tabela 2 apresenta a potência térmica do reator durante o teste físico inicial. Também estão incluídas as posições do banco de barras e a concentração crítica de boro. Os dados apresentados na tabela 2 correspondem ao estado *Hot Zero Power* (HZP) do

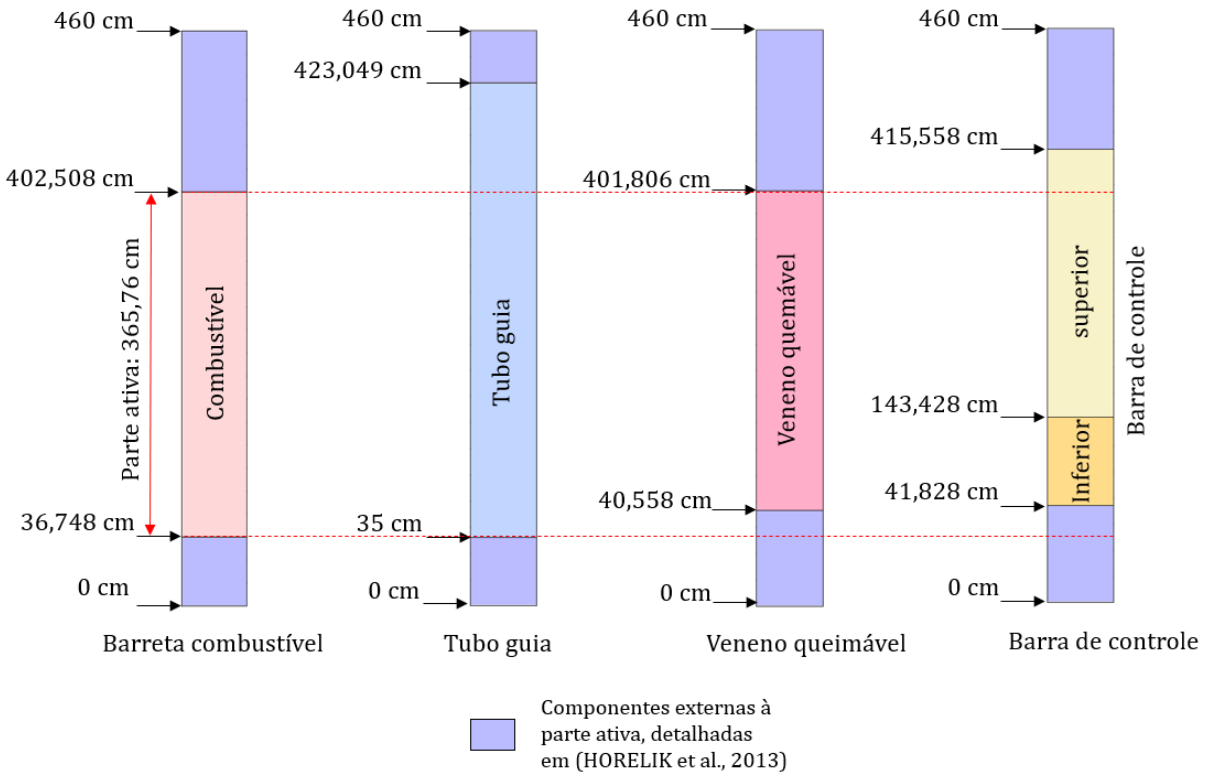


Figura 17 – Geometria axial das baretas de combustível, tubo guia, venenos queimáveis, e barras de controle (HORELIK et al., 2013).

Tabela 2 – Parâmetros do estado Hot Zero Power (HORELIK et al., 2013).

Potência do núcleo	25 MWth
Temperatura do refrigerante de entrada	560 ºF (566.483 K)
Posição das barras de controle A	Step 228
Posição das barras de controle B	Step 228
Posição das barras de controle C	Step 228
Posição das barras de controle D	Step 213
Concentração de boro	975 ppm

núcleo do reator. Esses parâmetros serão úteis para realizar a simulação nesse estado.

A figura 18 apresenta a distribuição de potência axial das medições do detector com média radial (círculos amarelos) fornecidas com as especificações de cada detetor no BEAVRS para o estado HZP.

No BEAVRS também está disponível a curva de queima do boro durante a operação do Ciclo 1 e do Ciclo 2. A figura 19 apresentam os dados do histórico de queima do boro no Ciclo 1. Finalmente, o histórico de potência do *Beginning of Cycle* (BOC) para a operação do Ciclo 1 é apresentada na figura 20.

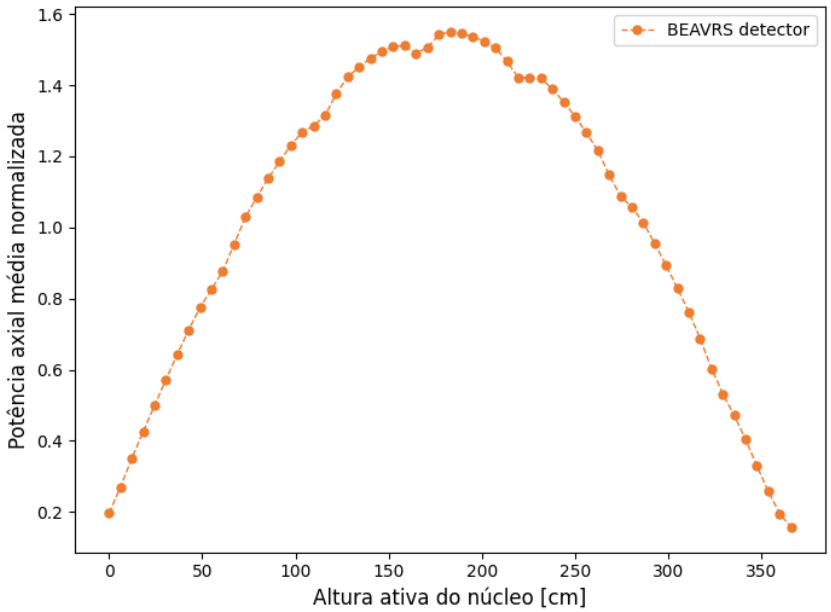
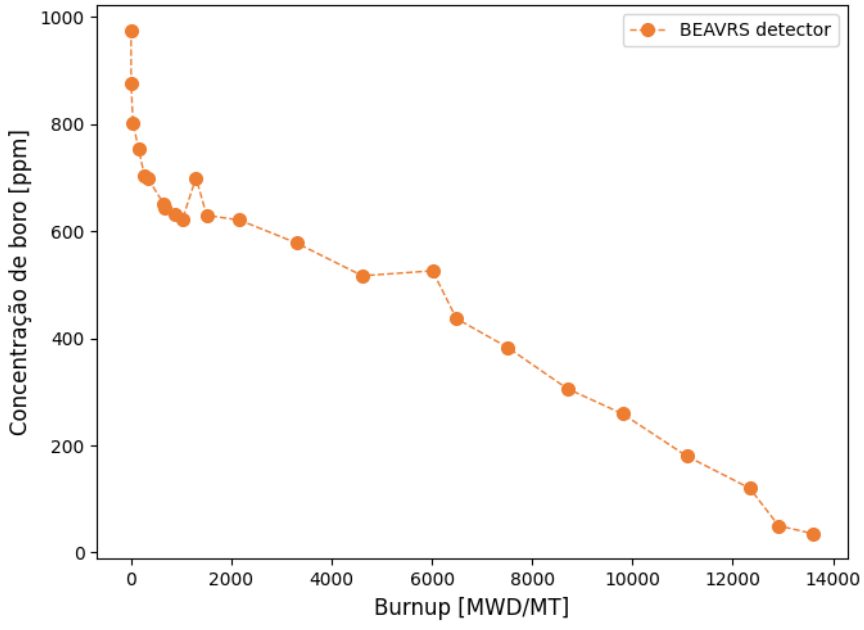


Figura 18 – Distribuição de potência axial média para HZP (HORELIK et al., 2013).



s

Figura 19 – Histórico de queima do boro no Ciclo 1 (HORELIK et al., 2013).

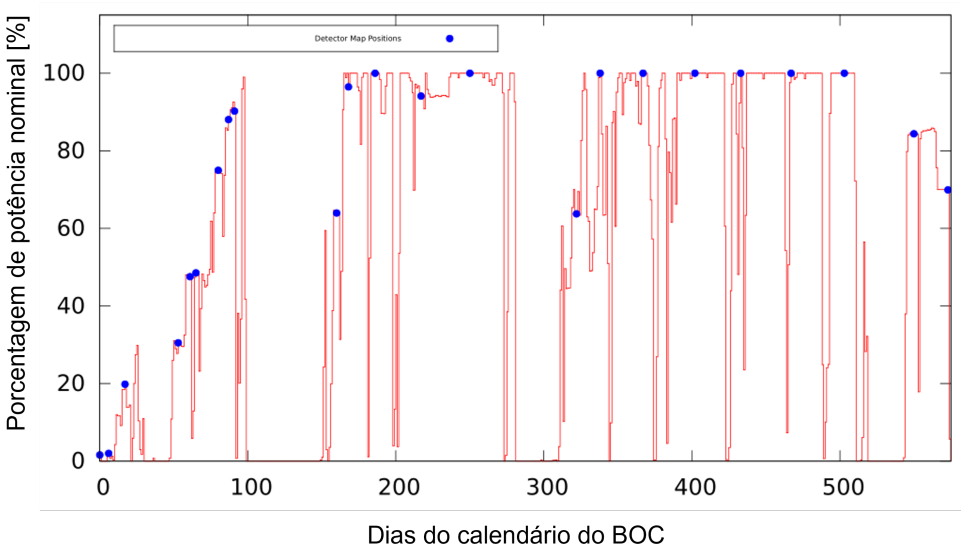


Figura 20 – Histórico de potência do Ciclo 1. (HORELIK et al., 2013).

Capítulo 4

Aprendizado de máquina

Modelos de aprendizado de máquina (*Machine Learning*) se tornaram a ferramenta mais amplamente usada para análise de dados. O termo "*Machine Learning*" tornou-se uma palavra comumente usada como um sinônimo de inteligência artificial. Este termo se refere apenas ao uso de um grande número de modelos matemáticos, como modelos de regressão estatística, regressão logística, algoritmos de agrupamento, filtros não lineares, classificadores de padrões não paramétricos, etc. (AIZENBERG, 2011). O avanço da capacidade de processamento dos computadores tem ajudado esses modelos a se tornarem muito eficientes, porque os algoritmos desenvolvidos são otimizados para o uso das GPUs (*Graphics Processing Unit*).

O aprendizado de máquina é uma técnica que determina o "modelo" a partir dos "dados". Aqui, dados significam literalmente informações como sinais, medições, documentos, áudio, imagens, etc. O modelo é o produto final do aprendizado de máquina. Na verdade, o modelo nada mais é do que o produto final que queremos alcançar. Alguns chamam o modelo de hipótese.

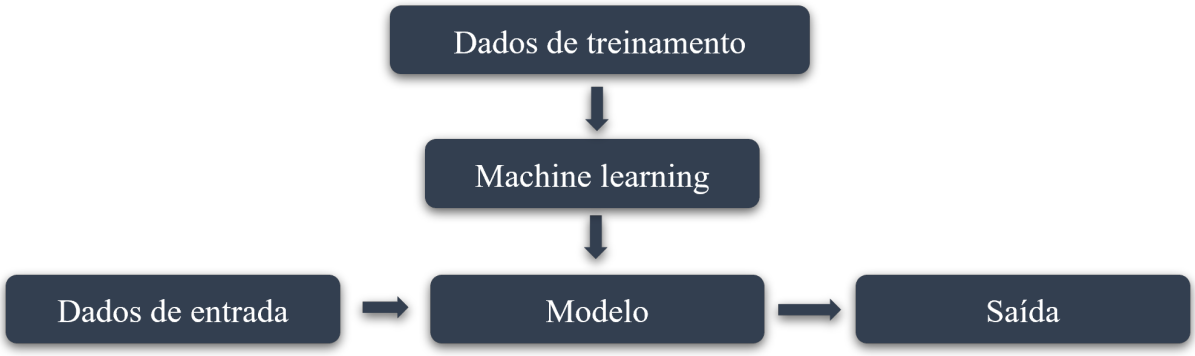


Figura 21 – Aplicação de um modelo baseado em dados de campo.

Portanto, o aprendizado de máquina é a técnica usada para encontrar (ou aprender) um modelo a partir dos dados. É adequado para problemas que envolvem inteligência, como reconhecimento de imagem, reconhecimento de fala, reconhecimento de padrões,

etc. Depois que o processo de aprendizado de máquina encontra o modelo dos dados de treinamento, aplicamos o modelo aos dados de campo reais. Esse processo é ilustrado na figura 21. O fluxo vertical na figura indica o processo de aprendizagem, e o modelo treinado é descrito como o fluxo horizontal, que é chamado de inferência.

4.1 Redes neurais artificiais (RNAs)

Os modelos de aprendizado de máquina podem ser implementados de várias maneiras. A rede neural é uma delas. Para ilustrar isso, a figura 22 mostra o mesmo esquema da figura 21, a diferença está no modelo. Uma rede neural é apenas um modelo dentro de uma grande variedade de modelos matemáticos que inclui o *Maching Learnig*.

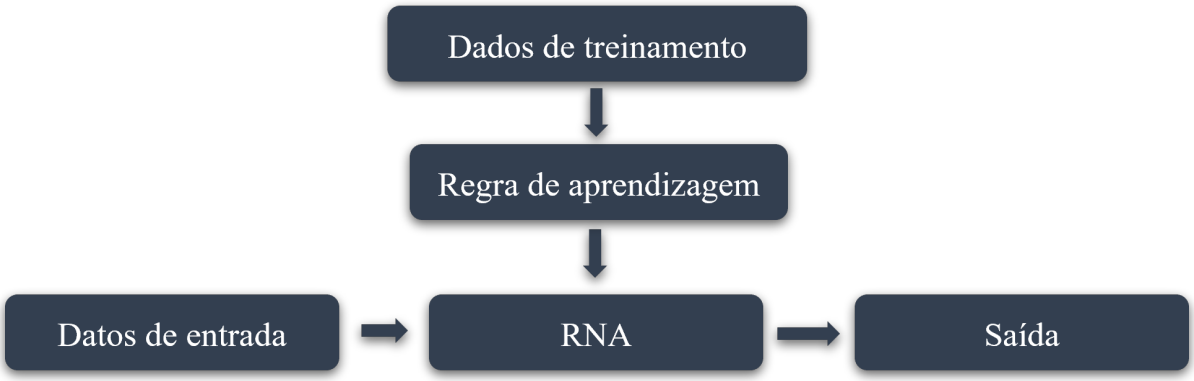


Figura 22 – Aplicação de um modelo baseado em RNAs com dados de campo.

Cada vez que aprendemos algo, nosso cérebro armazena o conhecimento. O computador usa memória para armazenar informações. Embora ambos armazenem informações, seus mecanismos são muito diferentes. O computador armazena informações em locais específicos da memória, enquanto o cérebro altera a associação de neurônios. O neurônio em si não tem capacidade de armazenamento; ele apenas transmite sinais de um neurônio para outro. O cérebro é uma rede gigantesca desses neurônios, e a associação de neurônios forma informações específicas.

A rede neural artificial (RNA) imita o mecanismo do cérebro. Como o cérebro é formado por conexões de vários neurônios, a rede neural é construída com conexões de nós, que são elementos que correspondem aos neurônios do cérebro. A rede neural imita a associação de neurônios, que é o mecanismo mais importante do cérebro, usando o valor do peso.

O primeiro modelo de neurônio artificial foi proposto por W. McCulloch e W. Pitts em 1943 (MCCULLOCH; PITTS, 1943). Eles tentaram criar um modelo matemático de processamento de informações neurais. Uma opinião comum era que um neurônio recebe alguns sinais de entrada $x_1, \ldots, x_n$ que podem ser excitatórios (“1”) ou inibitórios (“-1”). Neste processo calcula-se a soma ponderada das entradas $z = w_1x_1 + \cdots + w_nx_n$

e então, ele produz a saída excitatória (“1”) se a soma ponderada das entradas excede algum valor de limiar predeterminado, e a saída inibitória (“-1”) se isso não acontecer. Por muitos anos, é um fato conhecido que um neurônio biológico é muito mais sofisticado do ponto de vista do processamento de sinal (FAUSETT, 1994). Não é um elemento de processamento binário discreto, suas entradas e saídas são contínuas. Portanto, o modelo de McCulloch-Pitts como modelo de um neurônio biológico é muito esquemático e só se aproxima de uma ideia básica de processamento de informação neural. No entanto, é difícil superestimar a importância desse modelo. Em primeiro lugar, é historicamente o primeiro modelo de neurônio, e segundo lugar, esse modelo foi importante para compreender os mecanismos de aprendizagem que consideraremos mais tarde. Terceiro, todos os modelos neurais subsequentes são baseados na mesma abordagem do modelo McCulloch-Pitts: soma ponderada de entradas seguida pela função de transferência aplicada à soma ponderada para produzir a saída. No modelo McCulloch-Pitts o neurônio é um elemento de processamento binário. Ele recebe as entradas binárias $x_1, \ldots, x_n$ tomando seus valores do conjunto de valores de $\{-1, 1\}$ e produz a saída binária que pertence ao mesmo conjunto. Os pesos $w_1, \ldots, w_n$ podem ser números reais arbitrários e, portanto, a soma ponderada $z = w_1 x_1 + \cdots + w_n x_n$ também pode ser um número real arbitrário. A saída do neurônio $f(x_1, \ldots, x_n)$ é determinada da seguinte forma:

$$f(x_1, \ldots, x_n) = \begin{cases} 1, & \text{se } z \geq \theta \\ -1, & \text{se } z < \theta \end{cases} \tag{4.1}$$

onde θ é um limite predeterminado. A última equação pode ser transformada se o limite for incluído na soma ponderada como um “peso livre” ($w_0 = -\theta$), que muitas vezes também é chamado de viés e a soma ponderada será transformada de acordo:

$$f(x_1, \ldots, x_n) = \begin{cases} 1, & \text{se } z \geq \theta \\ -1, & \text{se } z < \theta \end{cases} \tag{4.2}$$

Este é o mesmo que

$$f(x_1, \ldots, x_n) = \text{sgn}(z) \tag{4.3}$$

onde *sgn* é uma função de sinal padrão, que é igual a 1 quando seu argumento não é negativo e -1 caso contrário (ver figura 23). Portanto, a função *sgn* em (4.3) é uma função de ativação. O neurônio McCulloch-Pitts também é frequentemente chamado de elemento limiar ou neurônio limiar (FAUSETT, 1994).

É importante mencionar que o sinal da função não é linear (HAYKIN, 2010). Portanto, o primeiro neurônio era um elemento de processamento não linear. Esta propriedade é muito importante. Todas as funções de ativação populares usadas em neurônios são

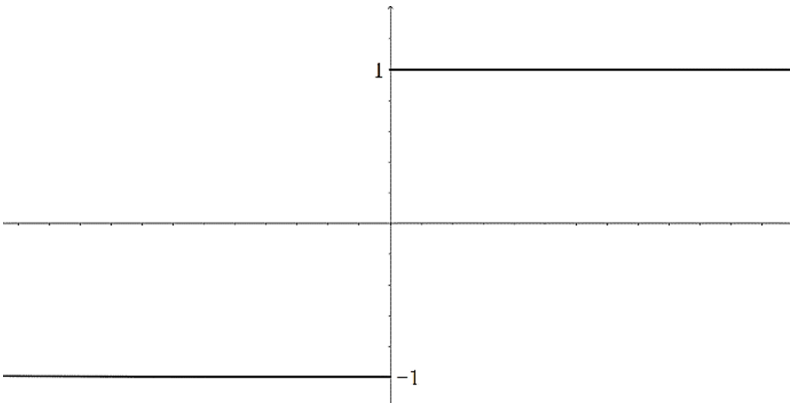


Figura 23 – Esquema que representa a função *sgn*.

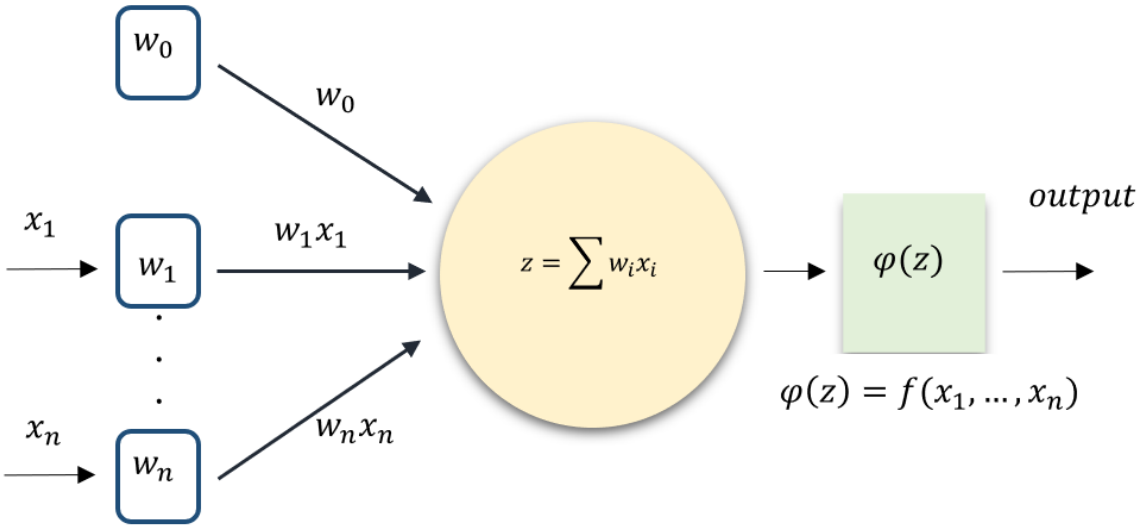


Figura 24 – Diagrama de um único neurônio.

não lineares. Não será uma superestimativa, se formos dizer que a funcionalidade de um neurônio é determinada principalmente (senão completamente) por sua função de ativação.

Vamos agora considerar o modelo mais geral de um neurônio, que comumente é usado hoje (ver figura 24). Um neurônio tem n entradas e pesos correspondentes a essas entradas. Também possui um peso livre w_0, que não corresponde a nenhuma entrada. Todos os pesos juntos formam um vetor $(w_0, w_1, \ldots, w_n)$ de ponderação dimensional $(n+1)$. Existe uma função de ativação predeterminada $\varphi(z)$ associada a um neurônio. Ele gera a saída do neurônio, limitando-o a algum intervalo razoável (permitido). O processamento neural consiste em duas etapas. A primeira etapa é o cálculo da soma ponderada das entradas dos neurônios $z = w_0 + w_1x_1 + \cdots + w_nx_n$.

A segunda etapa é o cálculo do valor da função de ativação $\varphi(z)$ para o valor z da soma ponderada. Este valor da função de ativação forma a saída do neurônio. Se alguma função descreve o mapeamento de entrada/saída $f(x_1, \ldots, x_n)$, então

$$f(x_1, \ldots, x_n) = \varphi(z) = \varphi(w_0 + w_1 x_1 + \cdots + w_n x_n) \tag{4.4}$$

4.1.1 Função de ativação ou função de transferência

A função de ativação ou transferência fornece o estado de ativação atual $a(t)$ do cálculo pós-sináptico z e o próprio estado de ativação anterior $a(t-1)$

$$a(t) = f(a(t-1), z). \tag{4.5}$$

Porém, em muitos modelos de sistemas de rede neural artificial, considera-se que o estado atual do neurônio não depende de seu estado anterior, se não apenas do estado atual.

$$a(t) = f(\mathrm{z}). \tag{4.6}$$

A função de ativação $f(z)$ é geralmente considerada determinística, e na maioria dos modelos é monotônica crescente e contínua, como é comumente observado em neurônios biológicos. A forma $y = f(z)$ das funções de ativação mais utilizadas em sistemas de rede neural artificial é mostrada na tabela 3. Em suma, designamos o potencial pós-sináptico com x, e o estado de ativação com y.

Tabela 3 – Funções de ativação comuns (algumas constantes foram omitidas)

	Função	Intervalo
Identidade	$y = x$	$]-\infty, +\infty[$
Degrau	$y = sgn(x)$	$[-1, +1]$
Linear por seções	$y = \begin{cases} -1, \text{ si } x < -1 \\ x, si + l \leq x \leq -l \\ +1, \text{ si } x > +1 \end{cases}$	$[-1, +1]$
Sigmoide	$y = \frac{1}{1+e^{-x}}$	$[0, 1]$
tangente h.	$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$[-1, +1]$
Gaussiana	$y = A \cdot e^{-Bx^2}$	$[0, +1]$
Sinusoidal	$y = A \cdot \mathrm{sen}(\omega x + \varphi)$	$[-1, +1]$

Às vezes, os algoritmos de aprendizagem requerem que a função de ativação cumpra a condição de ser diferenciável. As mais utilizadas neste sentido são as funções do tipo sigmóide, como no BP (*BackPropagation*). Outra função clássica é a Gaussiana, que é

usada em conjunto com regras de propagação que envolvem o cálculo de quadrados de distâncias (por exemplo, o Euclidiano) entre os vetores de entrada e peso. Finalmente, as funções senoidais são algumas vezes usadas, como nos casos em que é necessário expressar explicitamente uma periodicidade temporal.

4.1.2 Função de saída

Esta função fornece a saída global do neurônio e com base em seu estado de ativação atual $a(t)$. Muitas vezes, a função de saída é simplesmente a identidade $F(x) = x$, de modo que o estado de ativação do neurônio é considerado como a própria saída,

$$y(t) = F(a) = a. \tag{4.7}$$

Isso ocorre nos modelos mais comuns, como o MLP (*Multilayer Perceptron*) ou o adalina (HAYKIN, 2010). A função de saída também pode ser do tipo passo (Eq. 4.2), o que significa que o neurônio não dispara até que a ativação exceda certo limite. Em outros modelos, como a máquina de Boltzmann (HINTON; PROCESSING, 1986), é uma função estocástica de ativação, com a qual o neurônio terá um comportamento probabilístico.

4.2 Aprendizagem supervisionada

No aprendizagem supervisionado, um conjunto de treinamento é definido como $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ e cada par $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ é chamado de exemplo de treinamento; onde m é o número de exemplos de treinamento e $x_{(i)}$ é chamado de vetor de entrada. Cada $x_{(i)}$ tem o corresponde vetor saída $y_{(i)}$. Presumimos que haja uma função desconhecida $y = f(x)$ que mapeia os vetores entrada e saída. O objetivo do aprendizado supervisionado é usar o conjunto de treinamento para "aprender" ou estimar f. Chamamos essa função estimada de f'. Queremos que $f(x)$ seja mais próxima de $f^{'}(x)$, não apenas para o conjunto de treinamento, mas para todo o conjunto de dados.

4.3 Função de erro

Normalmente, o processo de aprendizagem requer a definição de uma função de erro L (ou função de perda) que quantifica a diferença entre a saída computada o e o verdadeiro valor y para uma entrada $\vec{x}$ sobre um conjunto de vários pares de entrada-saída $(\vec{x}, y)$. Historicamente, esta função de erro é o erro quadrático médio (EQM), definido para um conjunto de N pares de entrada-saída $X = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N)\}$, como

$$L(X) = \frac{1}{2N} \sum_{i=1}^{N} (o_i - y_i)^2 = \frac{1}{2N} \sum_{i=1}^{N} \left(g\left(\vec{w} \cdot \vec{x}_i + b\right) - y_i\right)^2 \tag{4.8}$$

onde o_i denota a saída do modelo e g a função de ativação. O fator de $\frac{1}{2}$ é incluído para simplificar o cálculo da derivada posteriormente. Assim, $E(X) = 0$ quando $o_i = y_i$ para todos os pares de entrada-saída $(\vec{x_i}, y_i)$ no X. Então um objetivo natural é tentar mudar $\vec{w}$ e b de tal modo que $E(X)$ é o mais próximo de zero.

4.4 Redes Neurais Multicamadas

Um único neurônio não pode fazer muito, mas podemos combinar muitos deles em uma estrutura composta por camadas, cada uma com um número diferente de neurônios (Ver figura 25), formando assim uma rede neural conhecida como *Multi Layer Perceptron* (MLP). O vetor de entrada $\vec{x}$ passa pela primeira camada, cujos valores de saída são conectados aos valores de entrada da próxima camada, e assim por diante, até que a rede forneça, como resultado, as saídas da última camada. Pode-se organizar a rede em várias camadas, tornando-a "profunda" e permitindo que ela seja capaz de aprender cada vez mais padrões e relacionamentos complexos presentes nos dados.

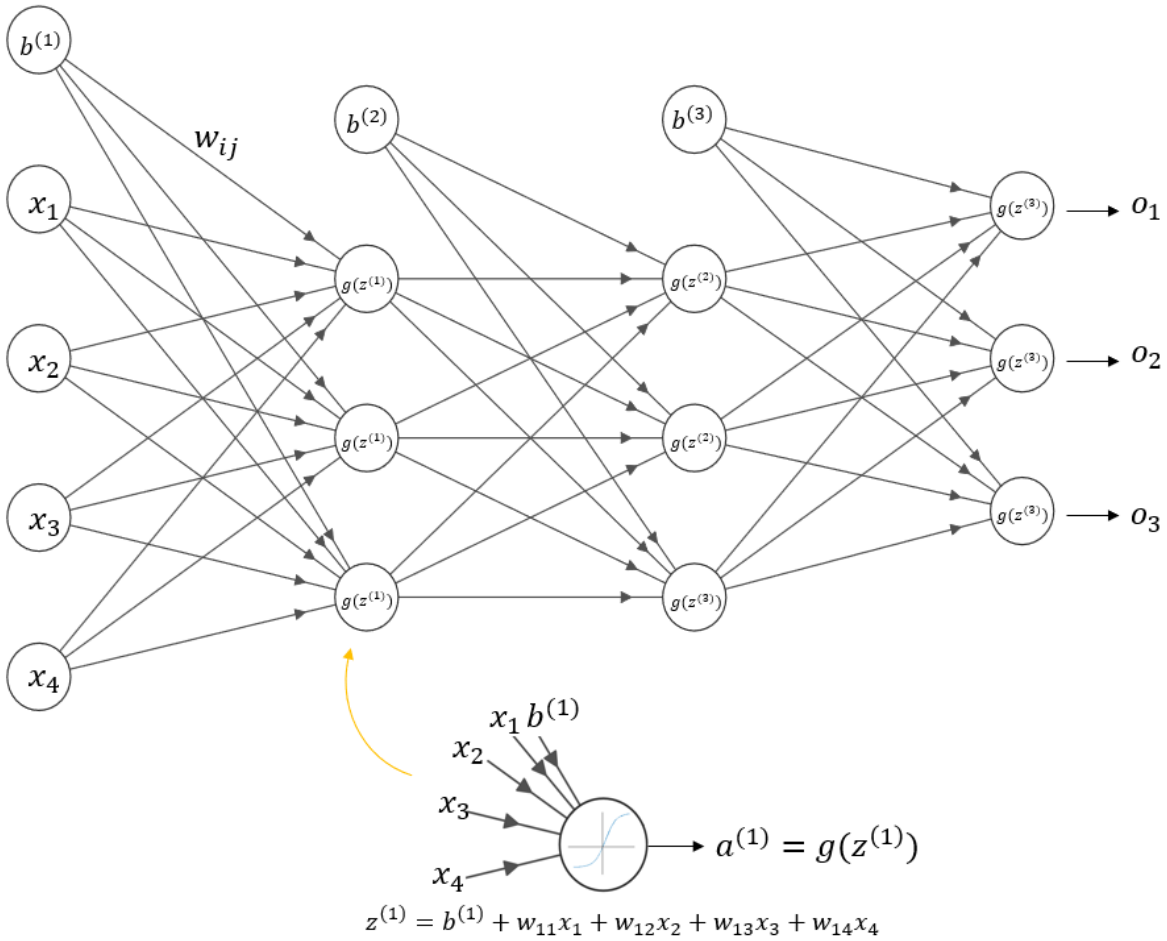


Figura 25 – Exemplo de uma RNA com uma camada de entrada, 2 camadas ocultas e uma camada de saída.

4.5 Treinando uma rede neural

Na rede neural de camada única, o processo de treinamento é relativamente simples porque o erro (ou função de perda) pode ser calculado como uma função direta dos pesos, o que permite fácil cálculo do gradiente. No caso de redes multicamadas, o problema é que a perda é uma função de composição complicada dos pesos nas camadas anteriores. O gradiente de uma função de composição é calculado usando o algoritmo de retropropagação (*Backpropagation*). Ele usa a regra da cadeia do cálculo diferencial, que calcula os gradientes de erro em termos de soma de produtos de gradiente local sobre os vários caminhos de um nó para saída, embora esse somatório tenha um número exponencial de componentes (caminhos). O algoritmo contém duas fases principais, conhecidas como fases de avanço (*Forward*) e retrocesso *Backward*. A fase de avanço é necessária para calcular os valores de saída e as derivadas locais em vários nós, e a fase de retrocesso é necessária para acumular os produtos desses valores locais em todos os caminhos do nó para a saída.

Na fase de avanço, as entradas para uma instância de treinamento são alimentadas na rede neural. Isso resulta em uma cascata direta de cálculos entre as camadas, usando o conjunto atual de pesos. A saída final prevista pode ser comparada àquela da instância de treinamento e a derivada da função de perda em relação à saída é calculada. A derivada dessa perda agora precisa ser calculada com relação aos pesos em todas as camadas na fase reversa.

O principal objetivo da fase de retrocesso é aprender o gradiente da função de perda em relação aos diferentes pesos usando a regra da cadeia do cálculo diferencial. Esses gradientes são usados para atualizar os pesos. Como esses gradientes são aprendidos na direção reversa, começando no nó de saída, esse processo de aprendizagem é conhecido como fase reversa.

Se considere uma sequência de unidades ocultas $h_1, h_2, \ldots, h_k$ seguida pela saída o, em relação à qual a função de perda L é calculada. Além disso, assumimos que o peso da conexão da unidade oculta h_r a h_{r+1} é $W(h_r, h_{r+1})$. Então, no caso de existir um único caminho de h_1 para o, pode-se derivar o gradiente da função de perda em relação a qualquer um desses pesos de aresta usando a regra da cadeia:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \frac{\partial L}{\partial o} \cdot \left[\frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right] \frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}} \forall r \in 1 \ldots k \tag{4.9}$$

A expressão acima assume que existe apenas um único caminho de h_1 a o existe na rede, enquanto um número exponencial de caminhos pode existir na realidade. Uma variante generalizada da regra da cadeia, conhecida como regra da cadeia multivariável, calcula o gradiente em um gráfico, onde mais de um caminho pode existir. Isso é obtido adicionando a composição ao longo de cada um dos caminhos de h_1 a o. Portanto, generalizando a

expressão acima para o caso em que existe um conjunto P de caminhos de h_r a o:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \underbrace{\frac{\partial L}{\partial o} \cdot \left[\sum_{[h_r,h_{r+1},\ldots h_k,o]\in\mathcal{P}} \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right]}_{\text{Cálculo do Backpropagation } \Delta(h_r,o)=\frac{\partial L}{\partial h_r}} \frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}} \tag{4.10}$$

O termo $\mathrm{d}\Delta\left(h_r, o\right) = \frac{\partial L}{\partial h_r}$ é agregado ao longo de um número exponencialmente crescente de caminhos (com relação ao comprimento do caminho), que parece ser intratável à primeira vista. Um ponto-chave é que uma rede neural não tem ciclos, e é possível calcular tal agregação de uma forma de princípio na direção reversa, primeiro computando $\Delta\left(h_k, o\right)$ para os nós h_k mais próximos de o, e então computar recursivamente esses valores para nós em camadas anteriores em termos de nós em camadas posteriores. Além disso, o valor de $\Delta(o, o)$ para cada nó de saída é inicializado da seguinte forma:

$$\Delta(o, o) = \frac{\partial L}{\partial o} \tag{4.11}$$

Este tipo de técnica de programação dinâmica é frequentemente usado para calcular com eficiência todos os tipos de funções centradas no caminho em gráficos acíclicos direcionados, que de outra forma exigiriam um número exponencial de operações. A recursão para $\Delta\left(h_r, o\right)$ pode ser derivada usando a regra da cadeia multivariável:

$$\Delta\left(h_r, o\right) = \frac{\partial L}{\partial h_r} = \sum_{h:h_r\Rightarrow h} \frac{\partial L}{\partial h} \frac{\partial h}{\partial h_r} = \sum_{h:h_r\Rightarrow h} \frac{\partial h}{\partial h_r} \Delta(h, o) \tag{4.12}$$

Uma vez que cada h está em uma camada posterior a h_r, $\Delta\left(h, o\right)$ já foi calculado durante a avaliação de $\Delta\left(h_r, o\right)$. No entanto, ainda precisamos avaliar $\frac{\partial h}{\partial h_r}$ para calcular a equação 4.12. Considere uma situação na qual a aresta que une h_r com h tem peso $w(h_r, h)$, e seja a_h o valor calculado na unidade oculta h antes de aplicar a função de ativação $\Phi(\cdot)$. Em outras palavras, temos $h = \Phi\left(a_h\right)$, onde a_h é uma combinação linear de suas entradas de unidades da camada anterior incidentes em h. Então, pela regra da cadeia univariada, a seguinte expressão para $\frac{\partial h}{\partial h_r}$ pode ser derivada:

$$\frac{\partial h}{\partial h_r} = \frac{\partial h}{\partial a_h} \cdot \frac{\partial a_h}{\partial h_r} = \frac{\partial \Phi\left(a_h\right)}{\partial a_h} \cdot w_{(h_r,h)} = \Phi'\left(a_h\right) \cdot w_{(h_r,h)} \tag{4.13}$$

Este valor de $\frac{\partial h}{\partial h_r}$ é usado na equação 4.12, que é repetida recursivamente na direção inversa, começando com o nó de saída. As atualizações correspondentes na direção inversa são as seguintes:

$$\Delta\left(h_r, o\right) = \sum_{h:h_r\Rightarrow h} \Phi'\left(a_h\right) \cdot w_{(h_r,h)} \cdot \Delta(h, o) \tag{4.14}$$

Portanto, gradientes são acumulados sucessivamente direção reversa, e cada nó é processado exatamente uma vez em uma passagem reversa. Observe que o cálculo da

equação 4.12 (que requer operações proporcionais ao número de arestas de saída) precisa ser repetido para cada aresta de entrada no nó para calcular o gradiente em relação a todos os pesos das arestas. Finalmente, a equação 4.13 requer o cálculo de $\frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}}$, que é facilmente calculado da seguinte forma:

$$\frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}} = h_{r-1} \cdot \Phi'\left(a_{h_r}\right) \tag{4.15}$$

Aqui, o gradiente principal que é retropropagado e a derivada em relação às ativações de camada e o gradiente em relação aos pesos é facilmente calculada para qualquer borda incidente na unidade correspondente.

A seguir, fornecemos uma versão alternativa da recursão de programação dinâmica, que é mais comumente vista em livros didáticos. Observe que a equação 4.13 usa as variáveis nas camadas ocultas como as variáveis de "cadeia" para a recursão de programação dinâmica. Também é possível usar os valores de pré-ativação das variáveis para a regra da cadeia. As variáveis de pré-ativação em um neurônio são obtidas após a aplicação da transformada linear como variáveis intermediárias. O valor de pré-ativação da variável oculta $h = \Phi\left(a_h\right)$ é a_h. Portanto, em vez da equação 4.13, pode-se usar a seguinte regra da cadeia:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \underbrace{\frac{\partial L}{\partial o} \cdot \Phi'\left(a_o\right) \cdot \left[\sum_{[h_r,h_{r+1},\ldots h_k,o]\in\mathcal{P}} \frac{\partial a_o}{\partial a_{h_k}} \prod_{i=r}^{k-1} \frac{\partial a_{h_{i+1}}}{\partial a_{h_i}}\right]}_{\text{Cálculo do Backpropagation } \delta(h_r,o)=\frac{\partial L}{\partial a_{h_r}}} \underbrace{\frac{\partial a_{h_r}}{\partial w_{(h_{r-1},h_r)}}}_{h_{r-1}} \tag{4.16}$$

Aqui, introduzimos a notação $\delta\left(h_r, o\right) = \frac{\partial L}{\partial a_{h_r}}$ em vez de $\Delta\left(h_r, o\right) = \frac{\partial L}{\partial h_r}$ para configurar a equação recursiva. O valor de $\delta(O, O) = \frac{\partial L}{\partial a_o}$ é inicializado da seguinte forma:

$$\delta(o, o) = \frac{\partial L}{\partial a_o} = \Phi'\left(a_o\right) \cdot \frac{\partial L}{\partial o} \tag{4.17}$$

Então, pode-se usar a regra de cadeia multivariável para configurar uma recursão semelhante:

$$\delta\left(h_r, o\right) = \frac{\partial L}{\partial a_{h_r}} = \sum_{h:h_r\Rightarrow h} \overbrace{\frac{\partial L}{\partial a_h}}^{\delta(h,o)} \underbrace{\frac{\partial a_h}{\partial a_{h_r}}}_{\Phi'(a_{h_r})w_{(h_r,h)}} = \Phi'\left(a_{h_r}\right) \sum_{h:h_r\Rightarrow h} w_{(h_r,h)} \cdot \delta(h, o) \tag{4.18}$$

Essa condição de recursão é encontrada mais comumente em livros que discutem retropropagação. A derivada parcial da perda em relação ao peso é então calculada usando $\delta\left(h_r, o\right)$ como segue:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \delta\left(h_r, o\right) \cdot h_{r-1} \tag{4.19}$$

Tal como acontece com a rede de camada única, o processo de atualização dos nós é repetido para a convergência, alternando repetidamente os dados de treinamento em épocas. Uma rede neural pode, às vezes, exigir milhares de épocas por meio dos dados de treinamento para aprender os pesos nos diferentes nós.

Parte III

Metodologia

Capítulo 5

Simulação neutrônica do BEAVRS

Com base no Benchmark BEAVRS, que contém especificações, dados operacionais e medições para um reator PWR, este capítulo apresenta a metodologia da simulação neutrônica do núcleo BEAVRS com dois objetivos: validar a modelagem usada neste trabalhos e obter dados de diferentes configurações espaciais dos ECs. Este calculo é importante, pois os dados gerados serão usados para treinar o modelo de otimização.

Atualmente, a análise completa do núcleo de um reator nuclear ainda é baseada no esquema tradicional de cálculo em duas etapas que são: (1) a homogeneização espacial e condensação de grupos de energia usando código de célula, e (2) o cálculo do núcleo completo em 3D usando poucas constantes de grupo, gerado na fase anterior. O trabalho de Samchez (SANCHEZ, 2009) fornece uma boa revisão das técnicas modernas de homogeneização. Nesse sentido a metodologia empregada nesse trabalho consiste em duas etapas. A primeira consiste na obtenção das seções de choque macroscópicas dos elementos combustíveis com o código WIM-ANL. Na segunda, as seções de choque obtidas serão utilizadas para simular o núcleo ativo do reator utilizando o código PARCS. Essas simulações são feitas para o estado *Hot Zero Power* (HZP) e para primeiro ciclo de operação.

5.1 Obtenção das seções de choque macroscópicas

A família de códigos WIMS permite o cálculo de distribuições de fluxo de nêutrons e valores de k_{inf} e k_{eff} em uma variedade de tipos de reatores. O objetivo principal do código é o cálculo de seções de choque homogeneizadas para áreas representativas do núcleo do reator, que são posteriormente utilizadas nos códigos nodais que resolvem a equação de difusão de muiltigrupos, como por exemplo, o PARCS (DEEN et al., 2004).

O código WIMS pode usar quatro modelos de geometria, que são:

- ***Pin Cell:*** representa uma rede infinita de células idênticas,
- ***Cluster:*** modela um elemento combustível ou uma macrocélula com regiões anulares,

- ***MULTICELL:*** representa um conjunto de células (Macrocélula) ou grupos conectados por probabilidades de que um nêutron escape de um dos tipos de células (ou grupos) interaja em outro tipo de células (ou grupos),
- ***SUPERCELL:*** expande o número de materiais de ressonância em um determinado caso do WIMS e melhora o tratamento das regiões de refletor e controle.

Sob a palavra "macrocélula" entende-se um fragmento repetitivo do núcleo, onde "repetitivo" significa que, por razões físicas, uma condição de contorno reflexiva pode ser aplicada em seu limite.

Neste trabalho se utiliza o método MULTICELL, ele oferece a possibilidade adicional de que, qualquer uma das células da figura 26a também possa ser um grupo de *pin cell* por si só. Qualquer uma dessas células, que por algum motivo se consideram diferentes das outras, pode ser calculada de forma independente. O cálculo é executado separadamente para cada tipo de célula, mas a condição de contorno usual é substituída por "limites intercelulares" expressos em termos das probabilidades de que um nêutron escape de um tipo de célula e interaja em outro. Isso é expresso esquematicamente na figura 26b, onde as células foram separadas para os cálculos e suas interconexões devem ser fornecidas de fora. A figura 26c ilustra que os tipos de células também podem diferir não apenas na composição, mas também nas dimensões.

Duas probabilidades básicas são apresentadas:

$P_{i\mathrm{A,jB}}$: probabilidade de que um nêutron nascido na região i da célula A tenha sua primeira colisão na região j da célula B, e

Q_{AB} : probabilidade de que um nêutron deixe o limite da célula A e tenha uma colisão na célula B.

Para essas probabilidades, podemos escrever:

$$P_{i\mathbf{A},j\mathbf{B}} = \delta_{\mathrm{AB}} p_{ij} + p_{ib}^{\mathbf{A}} \cdot Q_{\mathrm{AB}} \cdot \frac{p_{bj}^{\mathrm{B}}}{(1 - p_{bb}^{\mathrm{B}})} \tag{5.1}$$

onde:

p_{ij} : probabilidade de um nêutron nascido na região i ter sua primeira colisão na região j,

p_{ib}^{A} : probabilidade de que um nêutron nascido na região i da célula A alcance o limite da célula sem colisão,

$p_{bb}^{\mathrm{B}} = 1 - \sum_{j=1}^{J} p_{bj}$

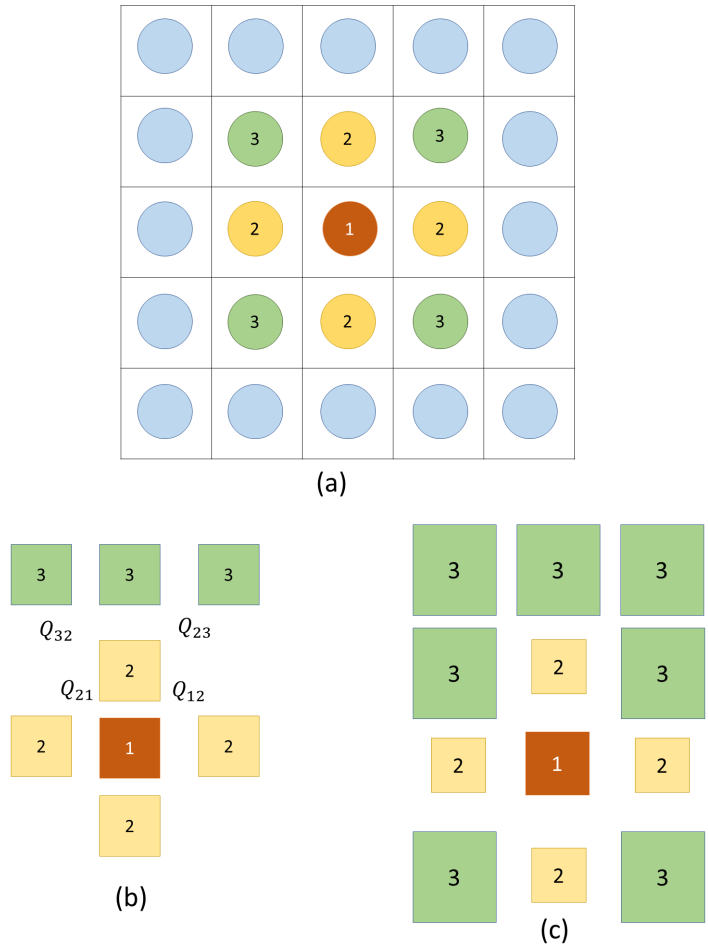


Figura 26 – Interpretação esquemática da abordagem MULTICELL (a) Macrocélula a ser calculado com os tipos de células escolhidos, (b) O cálculo do espectro de 69 grupos é realizado separadamente para cada tipo, (c) As células podem diferir em composição e geometria.

p_{bb}^{B} : probabilidade de um nêutron aparecer na fronteira da célula para ter sua primeira colisão na região j.

Para determinar Q_{AB}, podemos assumir que $Q_{AB} = S_{AB}$ é a probabilidade de que um nêutron deixando a superficie da célula A entre na superfície da célula B. Então S_{AB} é um dado puramente geométrico, isto é, seus elementos são frações das superfícies A e B, ou seja, é uma fração de superfície celular A contígua a B. O código introduz uma correção para o fluxo "não plano" com base na suposição de que o limite reflete parcialmente, ou seja, uma fração dos nêutrons que atingem o limite podem cruzá-lo.

Os cartões usados nas entradas para fazer uso do método MULTICELL, que diferem dos outros métodos, são os seguintes:

CELL i n: Em um cálculo multicélula, todos os dados de geometria referem-se ao cartão `CELL, i` é o cartão que identifica o tipo de célula, e `n` é o número de células deste

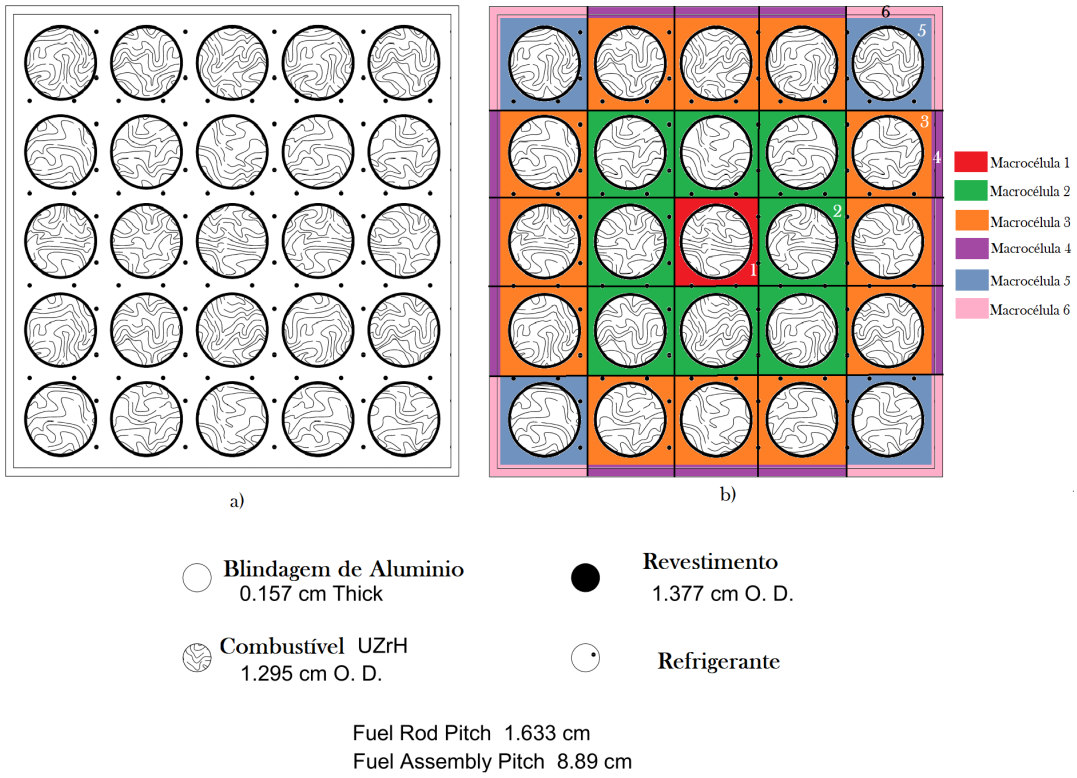


Figura 27 – : a) Esquema do elemento combustível do exemplo 9 (DEEN et al., 2004), b) Modelo MULTICELL do elemento combustível.

tipo (Default 1). Para minimizar os erros de arredondamento, pode ser aconselhável numerar os tipos de células em ordem crescente do valor **n**.

CSPECTRUM m: Se este cartão for fornecido com qualquer valor **m** diferente de 1 no cartão **CELL**, os espectros de colapsação e as seções de choque de ressonância microscópica para **CELL i** serão configurados igual ao conjunto **i-1**.

PCELL i Pij (para todo j): **i** é o tipo de célula. Os valores de **Pij** são as probabilidades de que um nêutron que sai de uma célula do tipo **i** entre imediatamente em uma célula do tipo **j** (sem necessariamente colidir com essa célula). A lista de valores de **Pij** pode ser repetida para cada grupo sucessivamente. Quando a lista terminar, o último conjunto será usado para todos os grupos restantes.

NCELL i: **i** é o número de células diferentes.

Para entender melhor o modelo MULTICELL, analisa-se o exemplo 8 do manual de usuário do WIMS-ANL (DEEN et al., 2004) que está relacionado a um *input* de um elemento (Figura 27a). O *input* apresenta os valores do cartão PCELL sem muita claridade de como foram obtidas, e nesse sentido, vale a pena desvendar como foi calculado esses valores. Uma vez compreendido ficará fácil entender o cálculo PCELL para o problema abordado neste trabalho.

Neste exemplo, o elemento combustível é divido em 6 macrocélulas (Figura 27)b. Portanto existem 6 conjuntos de probabilidades; por exemplo, a macrocélula 2 interatua com as macrocélulas 1 e 3 (P_{21} e P_{23}), e consigo mesma (P_{22}). Mas não interatua com resto das outras, então os valores serão zero (P_{24},P_{25},$P_{26} = 0$).

Então, os valores de P_{ij} são:

$$P_{ij} = \frac{l_{ij}}{L_i} \tag{5.2}$$

onde:

P_{ij} são as probabilidades de que um nêutron que sai de uma célula do tipo i entre imediatamente em uma célula do tipo j.

l_{ij} são a soma dos comprimentos de interação entre as células das macrocélulas i e j.

L_i é a soma de todos os perímetros das células da macrocélula i.

A macrocélula 1 ($i = 1$) interatua apenas com a 2. Para calcular as probabilidades, primeiro calcula-se a soma dos perímetros da macrocélula 1 (L_1), como trata-se de uma só célula, L_1 é:

$$L_1 = 4a$$

Onde a é o passo do reticulado das varetas combustíveis (1,633 cm)

Então, os valores de P_{1j} são:

$P_{11} = \frac{0}{4a} = 0$	$P_{12} = \frac{4a}{4a} = 1$	$P_{13} = \frac{0}{4a} = 0$
$P_{14} = \frac{0}{4a} = 0$	$P_{15} = \frac{0}{4a} = 0$	$P_{16} = \frac{0}{4a} = 0$

Portanto, o cartão **PCELL** da macrocélula 1 é

```
PCELL 1 (0. 1. 0. 0. 0. 0.)
```

Para a macrocélula 2 ($i = 2$), o valor de L_2 é:

$$L_2 = \sum_{r=1}^{8} 4a = 8(4a) = 32a$$

O cálculo dos valores de l_{2j}, não são muito triviais como o anterior caso. As células de tipo 2 interatuam com as células de tipo 1; então o comprimento de interação (l_{21}) é 4a.

Para o valor l_{22}, temos que identificar qual é o comprimento de interação entre essas células do mesmo tipo. Primeiro analisamos uma célula de tipo 2, vemos que interatua com outra célula também de tipo 2, num comprimento de 2a, portanto o valor de l_{22} é a

soma de todas as interações das células de tipo 2, ou seja $8 \times 2a = 16a$. Seguindo a mesma lógica para as outras células, os valores de P_{2j} são:

$P_{21} = \frac{4a}{32a} = 0,125$	$P_{22} = \frac{16a}{32a} = 0,5$	$P_{23} = \frac{12a}{32a} = 0,375$
$P_{24} = \frac{0}{32a} = 0$	$P_{25} = \frac{0}{32a} = 0$	$P_{26} = \frac{0}{32a} = 0$

Portanto, o cartão **PCELL** da macrocélula 2 é

```
PCELL 2 (0.125 0.5 0.375 0.0 0.0 0.0)
```

Para a macrocélula 3 ($i = 3$), o valor de L_3 é:

$$L_3 = 12 \cdot 4a = 48a$$

Para calcular os valores de l_{3j} segue-se a mesma lógica da macrocélula 2, então obtém-se

$P_{31} = \frac{0}{48a} = 0$	$P_{32} = \frac{12a}{48a} = 0,25$	$P_{33} = \frac{16a}{48a} = 0,333$
$P_{34} = \frac{12a}{48a} = 0,25$	$P_{35} = \frac{8a}{48a} = 0,167$	$P_{36} = \frac{0}{48a} = 0$

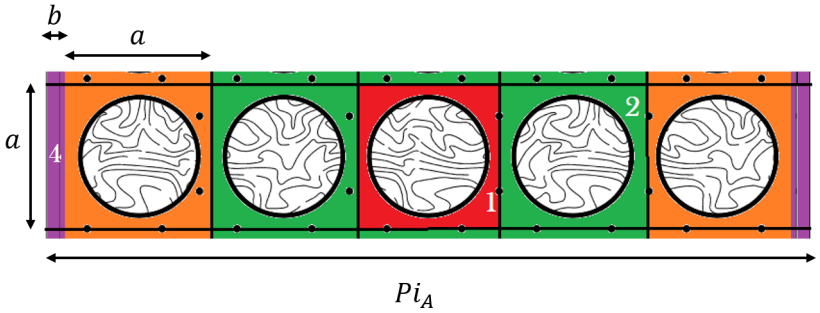
Portanto, o cartão **PCELL** da macrocélula 3 é

```
PCELL 3 (0.0 0.25 0.333 0.25 0.167 0.0)
```

Para a macrocélula 4 ($i = 4$), o cálculo de L_4 torna-se um pouco complexa, pois as células de tipo 4 não são quadrados, então L_4 é

$$L_4 = 12 \cdot 2a + 12 \cdot 2b = 24a + 24b$$

Agora calculamos o valor de b



$$b = \frac{Pi_A - 5a}{2} = \frac{8,89 \text{ cm} - 5 \cdot 1,633 \text{ cm}}{2} = 0,362 \text{ cm}$$

Então,

$$L_4 = 24(1,633 \text{ cm}) + 24(0,362 \text{ cm}) = 47,88 \text{ cm}$$

Para calcular os valores de l_{4j} segue-se a mesma lógica da macrocélula 2, exceto para o P_{44}. Então obtém-se

$P_{41} = \frac{0}{47,88} = 0$	$P_{42} = \frac{0}{47,88} = 0$	$P_{43} = \frac{12 \cdot 1,633}{47,88} = 0,4093$
$P_{44} = \frac{16b+12a}{47,88} = 0,5302$	$P_{45} = \frac{0}{47,88} = 0$	$P_{46} = \frac{8b}{47,88} = 0,0605$

Para o cálculo de P_{44}, deve-se considerar que, os nêutrons que interatuam com a cobertura de alumínio, são refletidos dentro da própria célula, e então, no comprimento de interação, deve-se somar o comprimento reflexivo $(16b+12a)$. Portanto, o cartão **PCELL** da macrocélula 4 é

```
PCELL 4 (0.0 0.0 0.4093 0.5302 0.0 0.0605)
```

Para a macrocélula 5 $(i = 5)$, o valor de L_5 é:

$$L_5 = 4 \cdot 4a = 16a$$

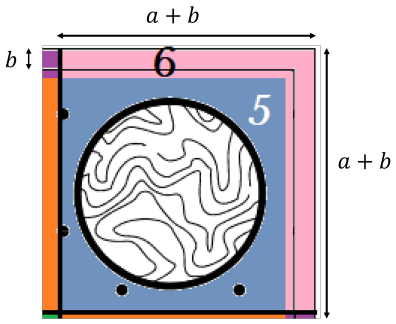
Para calcular os valores de l_{5j} segue-se a mesma lógica da macrocélula 2. Então obtém-se:

$P_{51} = \frac{0}{16a} = 0$	$P_{52} = \frac{0}{16a} = 0$	$P_{53} = \frac{8a}{16a} = 0,5$
$P_{54} = \frac{0}{16a} = 0$	$P_{55} = \frac{0}{16a} = 0$	$P_{56} = \frac{8a}{16a} = 0,5$

Portanto, o cartão **PCELL** da macrocélula 5 é

```
PCELL 5 (0.0 0.0 0.5 0.0 0.0 0.5)
```

Para a macrocélula 6 $(i = 6)$, o valor de L_6 é:



$$L_6 = 4[2(a+b) + 2a + 2b] = 16(a+b) = 31,92 \text{ cm}$$

Para calcular os valores de l_{6j} segue-se a mesma lógica da macrocélula 4. Então obtém-se:

$P_{61} = \frac{0}{31,92} = 0$	$P_{62} = \frac{0}{31,92} = 0$	$P_{63} = \frac{0}{31,92} = 0$
$P_{64} = \frac{8b}{31,92} = 0,0907$	$P_{65} = \frac{8a}{31,92} = 0,4093$	$P_{66} = \frac{4 \cdot 2(a+b)}{31,92} = 0,5$

Portanto, o cartão **PCELL** da macrocélula 6 é

```
PCELL 6 (0. 0. 0. 0.0907 0.4093 0.5)
```

O elemento combustível do exemplo 8 do manual do usuário (DEEN et al., 2004), é similar aos elementos combustíveis do BEAVRS, com a diferença de que os ECs não contém cobertura de alumínio. Nas figuras 28 ao 33 são mostrados os diferentes tipos de macrocélulas que foram considerados para modelar os elementos combustíveis no código WIMS-ANL. Os tipos de macrocélulas são numerados, também marcados com cores diferentes para identificar que cada um deles são diferentes. Os ECs que são mostrados, correspondem ao ciclo 1 do reator (BEAVRS) que são 9 elementos combustíveis, dos quais o EC 1, EC 2 e EC 5 são iguais em geometria, mas não em enriquecimento. Da mesma forma o EC 4 e EC 8 são iguais em geometria e o resto dos EC (EC 3, E 6 e EC 9) são diferentes em geometria. Deve-se entender "diferença em geometria" à diferença do um EC em relação ao outro na posição e no número das varetas de venenos queimáveis. Para os EC que são iguais em geometria, as matrizes de probabilidades também são iguais, pois a probabilidades de interação, tem a ver com a geometria e não com o enriquecimento. Portanto, com essas considerações, haverá 6 matrizes de probabilidades diferentes. As tabelas 6 ao 11, contem os valores de P_{ij} (matrizes de probabilidades) calculados com a mesma lógica do exemplo 8 apresentada anteriormente.

É importante notar que alguns dos ECs tem a mesma matriz de probabilidades, isto é devido a que as probabilidades de interação de uma macrocélula com outra dependem da geometria, e não da composição. Uma vez que as matrizes de probabilidades para cada EC foram obtidas, elas serão usadas para escrever as entradas para cada EC separadamente. O primeiro passo para validar a metodologia será simular o núcleo em condições HZP. Os parâmetros para a simulação nestas condições são mostrados na tabela 4.

Tabela 4 – Parâmetros usados nos inputs do WIMS-ANL para HZP.

Temperatura do combustível	566 K
Temperatura do refrigerante	566 K
Concentração de boro	975 ppm

Para as condições de queima do núcleo, também foi utilizado o modelo MULTICELL. Não foram feitas grandes modificações na estrutura das entradas no estado HZP, apenas foi adicionada o cartão relacionado ao comando de queima do combustível. Os parâmetros usados são mostrados na tabela 5.

Tabela 5 – Parâmetros usados para a queima de combustível.

Temperatura do combustível	900 K
Temperatura do refrigerante	560 K
Concentração inicial de boro	975 ppm
Dias de operação (Ciclo 1)	575

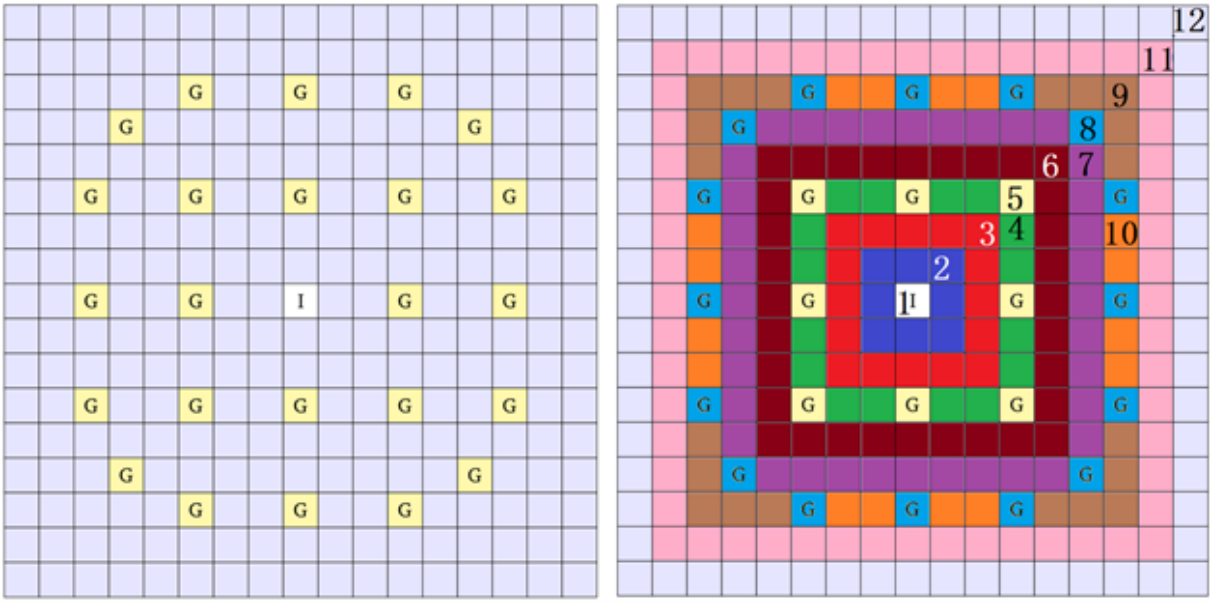


Figura 28 – Esquema do modelo MULTICELL para o EC 1, EC 2 e EC 5.

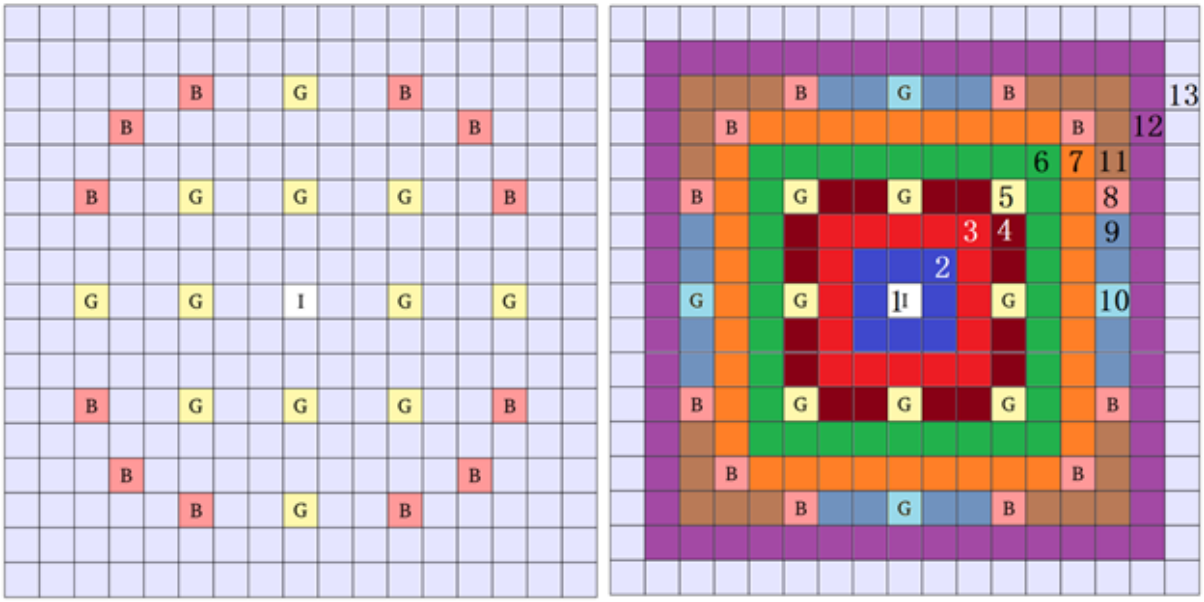


Figura 29 – Esquema do modelo MULTICELL para o EC 3.

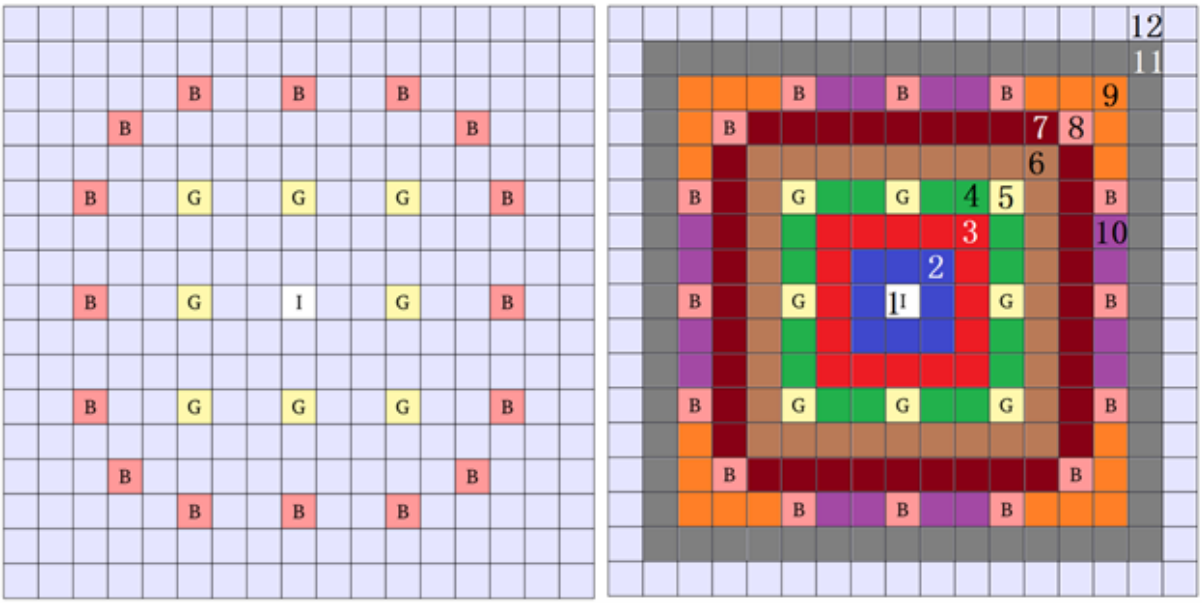


Figura 30 – Esquema do modelo MULTICELL para o EC 4 e EC 8.

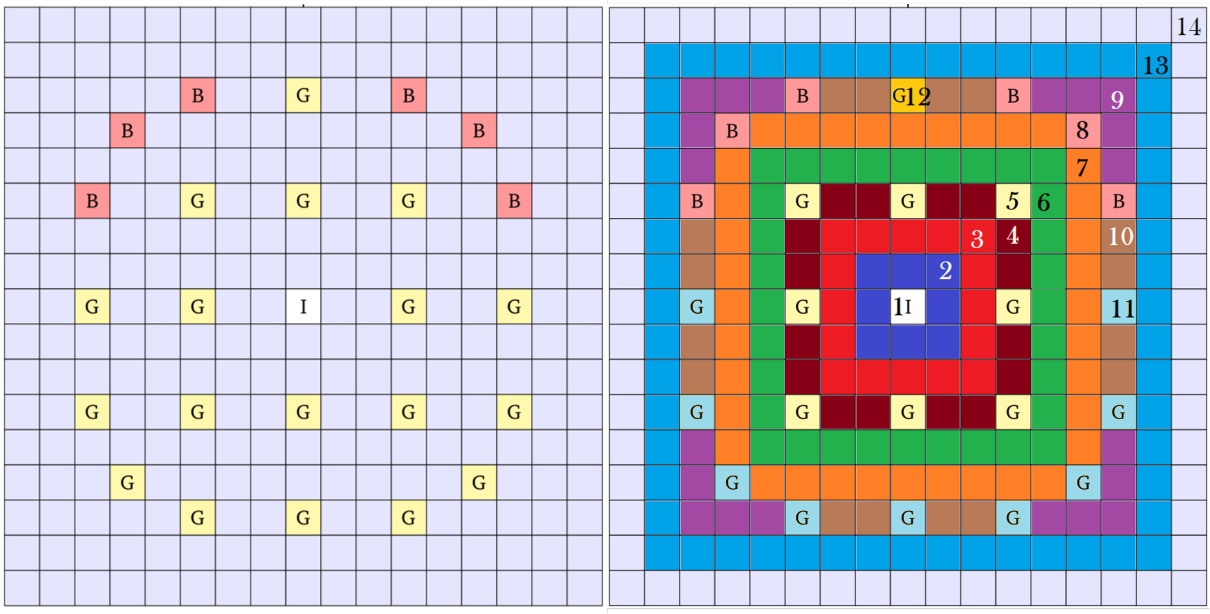


Figura 31 – Esquema do modelo MULTICELL para o EC 6.

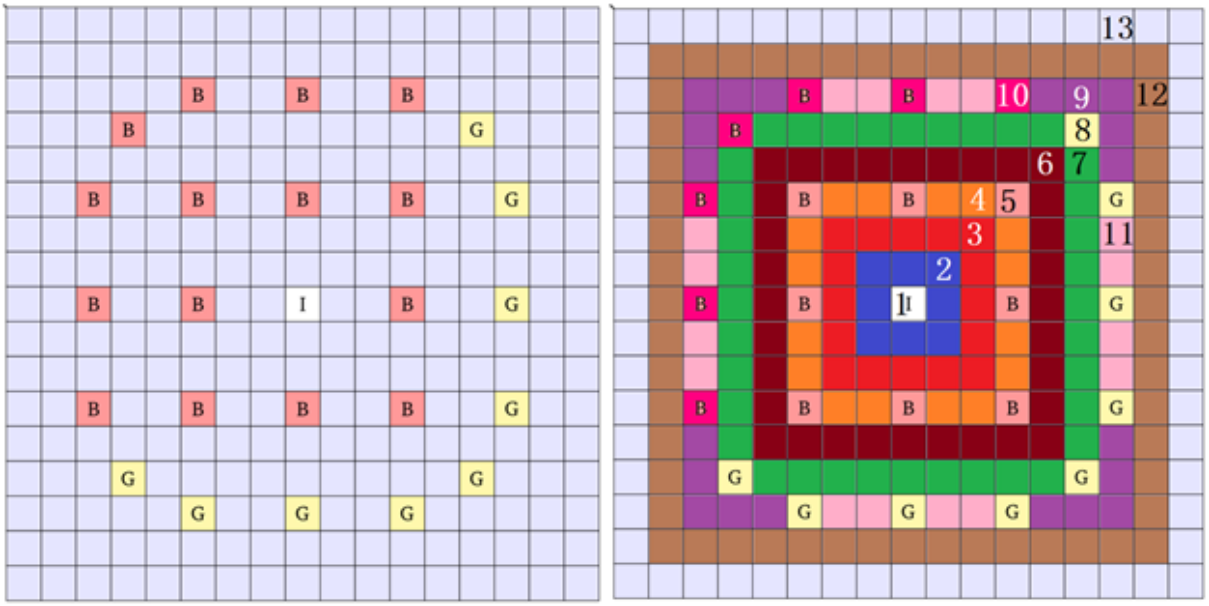


Figura 32 – Esquema do modelo MULTICELL para o EC 7.

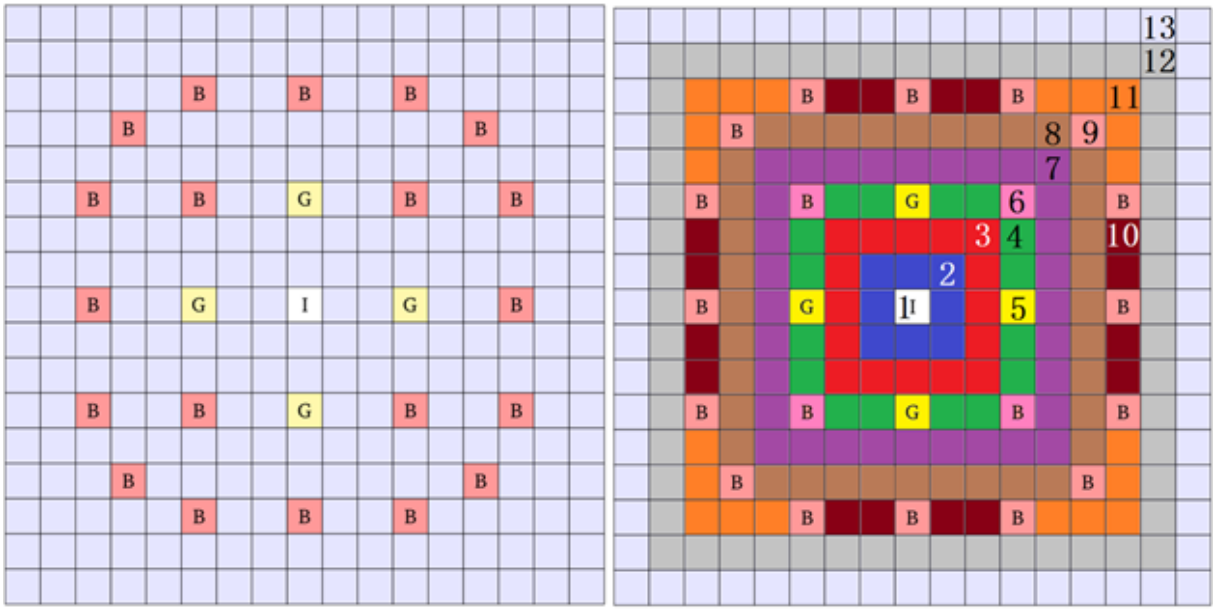


Figura 33 – Esquema do modelo MULTICELL para o EC 9.

Tabela 6 – Valores de P_{ij} para EC 1, EC 2 e EC 5.

i j	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1.000	0	0	0	0	0	0	0	0	0	0
2	0.125	0.500	0.375	0	0	0	0	0	0	0	0	0
3	0	0.188	0.500	0.250	0.063	0	0	0	0	0	0	0
4	0	0	0.250	0.250	0.250	0.250	0	0	0	0	0	0
5	0	0	0.125	0.500	0	0.375	0	0	0	0	0	0
6	0	0	0	0.125	0.094	0.500	0.281	0	0	0	0	0
7	0	0	0	0	0	0.250	0.444	0.139	0.056	0.111	0	0
8	0	0	0	0	0	0	0.313	0	0.250	0.250	0.188	0
9	0	0	0	0	0	0	0.100	0.200	0.400	0	0.300	0
10	0	0	0	0	0	0	0.250	0.250	0	0.250	0.250	0
11	0	0	0	0	0	0	0	0.054	0.107	0.071	0.500	0.268
12	0	0	0	0	0	0	0	0	0	0	0.234	0.766

Tabela 7 – Valores de P_{ij} para EC 3.

ij	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1.000	0	0	0	0	0	0	0	0	0	0	0
2	0.125	0.500	0.375	0	0	0	0	0	0	0	0	0	0
3	0	0.188	0.500	0.250	0.063	0	0	0	0	0	0	0	0
4	0	0	0.250	0.250	0.250	0.250	0	0	0	0	0	0	0
5	0	0	0.125	0.500	0	0.375	0	0	0	0	0	0	0
6	0	0	0	0.125	0.094	0.500	0.281	0	0	0	0	0	0
7	0	0	0	0	0	0.250	0.444	0.111	0.111	0.028	0.056	0	0
8	0	0	0	0	0	0	0.333	0	0.167	0	0.333	0.167	0
9	0	0	0	0	0	0	0.250	0.125	0.250	0.125	0	0.250	0
10	0	0	0	0	0	0	0.250	0	0.500	0	0	0.250	0
11	0	0	0	0	0	0	0.100	0.200	0	0	0.400	0.300	0
12	0	0	0	0	0	0	0	0.036	0.071	0.018	0.107	0.500	0.268
13	0	0	0	0	0	0	0	0	0	0	0	0.234	0.766

Tabela 8 – Valores de P_{ij} para EC 4 e EC 8.

i j	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1.000	0	0	0	0	0	0	0	0	0	0
2	0.125	0.500	0.375	0	0	0	0	0	0	0	0	0
3	0	0.188	0.500	0.250	0.063	0	0	0	0	0	0	0
4	0	0	0.250	0.250	0.250	0.250	0	0	0	0	0	0
5	0	0	0.125	0.500	0	0.375	0	0	0	0	0	0
6	0	0	0	0.125	0.094	0.500	0.281	0	0	0	0	0
7	0	0	0	0	0	0.250	0.444	0.139	0.056	0.111	0	0
8	0	0	0	0	0	0	0.313	0	0.250	0.250	0.188	0
9	0	0	0	0	0	0	0.100	0.200	0.400	0	0.300	0
10	0	0	0	0	0	0	0.250	0.250	0	0.250	0.250	0
11	0	0	0	0	0	0	0	0.054	0.107	0.071	0.500	0.268
12	0	0	0	0	0	0	0	0	0	0	0.234	0.766

Tabela 9 – Valores de P_{ij} para EC 6.

ij	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0.125	0.5	0.375	0	0	0	0	0	0	0	0	0	0	0
3	0	0.1875	0.5	0.25	0.063	0	0	0	0	0	0	0	0	0
4	0	0	0.25	0.25	0.25	0.25	0	0	0	0	0	0	0	0
5	0	0	0.125	0.5	0	0.375	0	0	0	0	0	0	0	0
6	0	0	0	0.125	0.094	0.5	0.281	0	0	0	0	0	0	0
7	0	0	0	0	0	0.25	0.444	0.056	0.056	0.111	0.076	0.007	0	0
8	0	0	0	0	0	0	0.333	0	0.333	0.167	0	0	0.167	0
9	0	0	0	0	0	0	0.1	0.1	0.4	0	0.1	0	0.3	0
10	0	0	0	0	0	0	0.25	0.063	0	0.25	0.156	0.031	0.25	0
11	0	0	0	0	0	0	0.306	0	0.222	0.278	0	0	0.194	0
12	0	0	0	0	0	0	0.25	0	0	0.5	0	0	0.25	0
13	0	0	0	0	0	0	0	0.018	0.107	0.071	0.031	0.004	0.5	0.268
14	0	0	0	0	0	0	0	0	0	0	0	0	0.234	0.766

Tabela 10 – Valores de P_{ij} para EC 7.

ij	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1.000	0	0	0	0	0	0	0	0	0	0	0
2	0.125	0.500	0.375	0	0	0	0	0	0	0	0	0	0
3	0	0.188	0.500	0.250	0.063	0	0	0	0	0	0	0	0
4	0	0	0.250	0.250	0.250	0.250	0	0	0	0	0	0	0
5	0	0	0.125	0.500	0	0.375	0	0	0	0	0	0	0
6	0	0	0	0.125	0.094	0.500	0.281	0	0	0	0	0	0
7	0	0	0	0	0	0.250	0.444	0.083	0.056	0.056	0.111	0	0
8	0	0	0	0	0	0	0.333	0	0.278	0	0.222	0.167	0
9	0	0	0	0	0	0	0.100	0.125	0.400	0.075	0	0.300	0
10	0	0	0	0	0	0	0.286	0	0.214	0	0.286	0.214	0
11	0	0	0	0	0	0	0.250	0.125	0	0.125	0.250	0.250	0
12	0	0	0	0	0	0	0	0.027	0.107	0.027	0.071	0.500	0.268
13	0	0	0	0	0	0	0	0	0	0	0	0.234	0.766

Tabela 11 – Valores de P_{ij} para EC 9.

i j	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0.125	0.500	0.375	0	0	0	0	0	0	0	0	0	0
3	0	0.188	0.500	0.250	0.063	0	0	0	0	0	0	0	0
4	0	0	0.250	0.250	0.125	0.125	0.250	0	0	0	0	0	0
5	0	0	0.250	0.500	0	0	0.250	0	0	0	0	0	0
6	0	0	0	0.500	0	0	0.500	0	0	0	0	0	0
7	0	0	0	0.125	0.031	0.063	0.500	0.281	0	0	0	0	0
8	0	0	0	0	0	0	0.250	0.444	0.139	0.111	0.056	0	0
9	0	0	0	0	0	0	0	0.313	0	0.250	0.250	0.188	0
10	0	0	0	0	0	0	0	0.250	0.250	0.250	0	0.250	0
11	0	0	0	0	0	0	0	0.100	0.200	0	0.400	0.300	0
12	0	0	0	0	0	0	0	0	0.054	0.071	0.107	0.500	0.268
13	0	0	0	0	0	0	0	0	0	0	0	0.234	0.766

No arquivo de saída do WIMS, podemos encontrar as seções de choque para cada elemento combustível, com as quais podemos simular o núcleo completo. Esses resultados não incluem as condições de contorno do núcleo do reator, ou seja, o refletor do núcleo não é levado em consideração. Para ter um modelo completo é necessário obter as seções de choque do refletor. Devido à necessidade dessa inclusão, adotamos a metodologia sugerida no manual do usuário WIMS-ANL (DEEN et al., 2004), que consiste em semi-homogeneizar os elementos combustíveis vizinhos ao refletor (Ver figura 34). Esta semi-homogeneização permite que o WIMS use o espectro de energia dos nêutrons que provêm do elemento combustível, para obter apenas as seções de choque macroscópicas homogeneizadas do refletor.

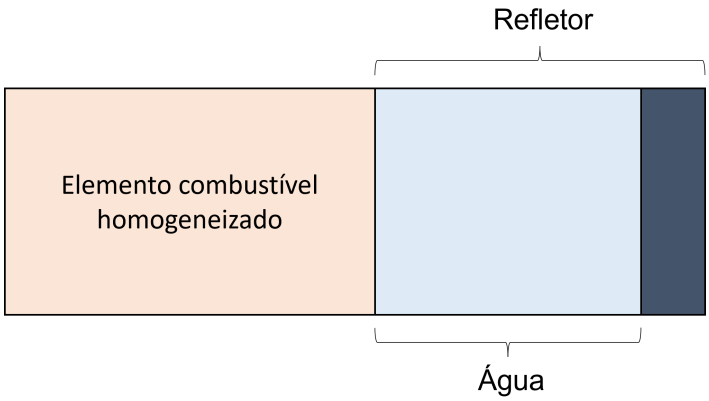


Figura 34 – Seção transversal do sistema modelado para obter as seções de choque macroscópicas do refletor.

A lógica da semi-homogeneização é similar para todos os elementos combustíveis vizinhos ao refletor. Esta metodologia está esquematizada na figura 35a, que ilustra a semi-homogeneização do elemento combustível 9 (EC 9), onde cada região de diferente cor (figura 35b) corresponde uma subdivisão de diferentes materiais.

5.2 Simulação do núcleo com o código PARCS

O modelo do núcleo com PARCS, terá a mesma nodalização radial que a recarga padrão do núcleo, um nó para um EC e um total de 257 nós (Figura 36a). A nodalização axial consiste em 52 níveis axiais da parte ativa do núcleo de igual comprimento com dois nós refletores, superior e inferior (ver figura 36b).

Outra metodologia de modelagem foi usada para os refletores radiais e axiais, onde foram utilizadas as sugestões do manual do usuário do WIMS-ANL."(ver seção 5.1). Para simular o refletor radial e axial, o EC em cinza (ver figura 36), foi aplicado como uma região quadrada adicional, tendo o mesmo tamanho que o outro EC é preenchido com 2 regiões: água e aço (Stainless Steel). Além disso, por uma questão de simplicidade e para reduzir o esforço computacional, os modelos de grade espaçadora não foram considerados.

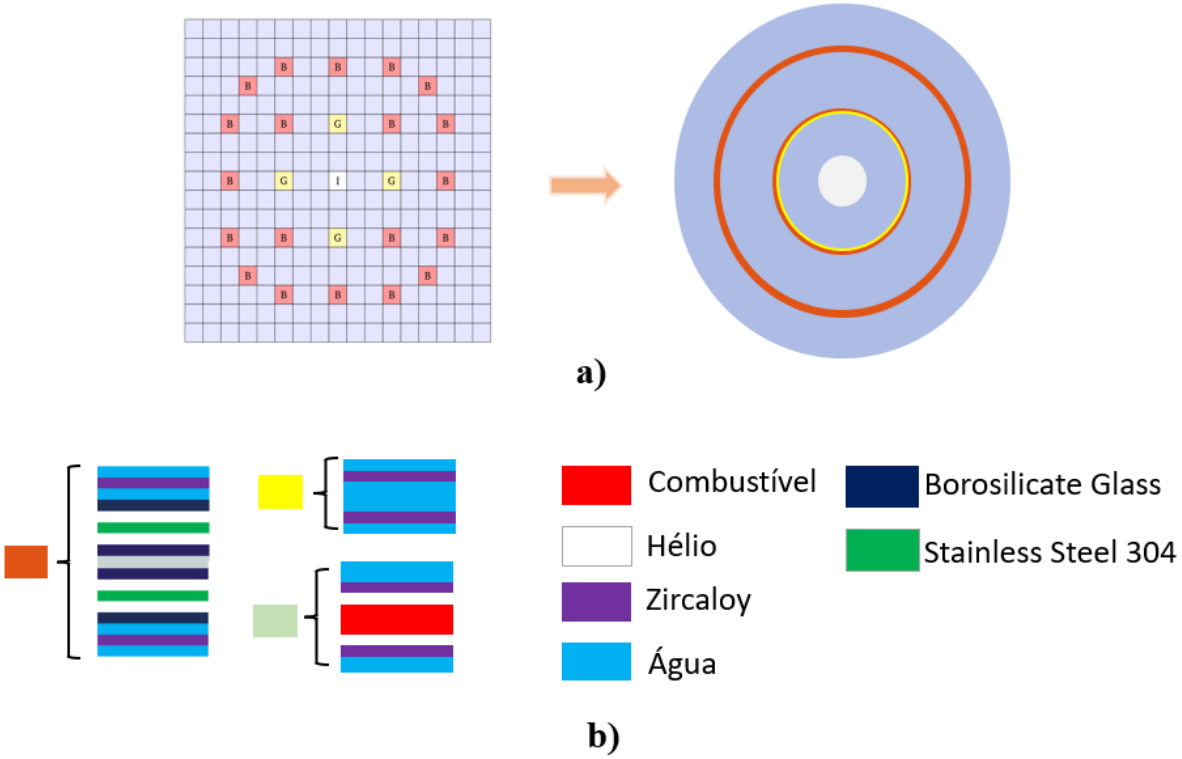


Figura 35 – A figura a) mostra a semi-homogeneização do EC 8, a figura b) mostra as subdivisões das regiões da homogeneização da figura a).

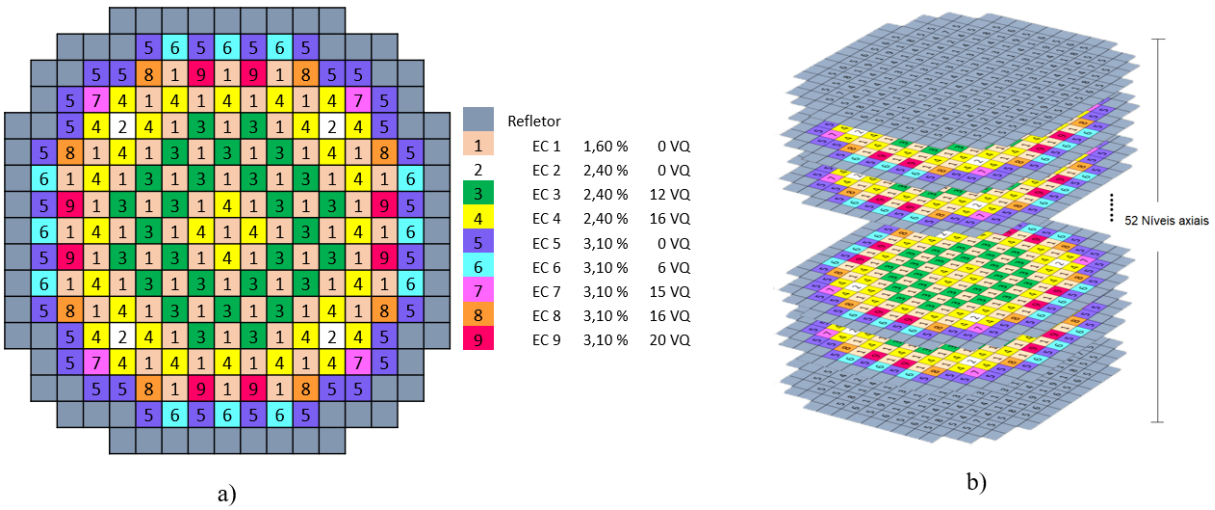


Figura 36 – a) Perfil radial modelado no PARCS do BEAVR, b) Perfil axial do núcleo modelado no PARCS.

Para o estado HZP se usaram os parâmetros apresentados na tabela 2 onde o banco de barras de controle D estão parcialmente inseridas.

Para as condições de queima do combustível, se considerou que todas as barras de controle estão fora do núcleo. Os parâmetros usados para a simulação, apresenta-se na tabela 5.

Para a condição de operação do reator (queima do combustível), como já men-

cionado, este trabalho segue a metodologia de simulação neutrônica de duas etapas. A primeira consiste em obter dados de seção de choque para todos os ECs do núcleo durante cada período de queima (intervalo de dias de operação), usando o código WIMS-ANL. Neste trabalho foram utilizados 23 intervalos de tempo, em cada um deles o reator está a uma determinada potência operacional, os quais são detalhados no BEAVRS (HORELIK et al., 2013).

A segunda etapa consiste no uso das seções de choque como entrada para simular o núcleo com o código PARCS 2.4. Para visualizar esta simulação de forma simples, a figura 37 esquematiza o processo que foi utilizado combinando os códigos WIMS-ANL e PARCS 2.4.

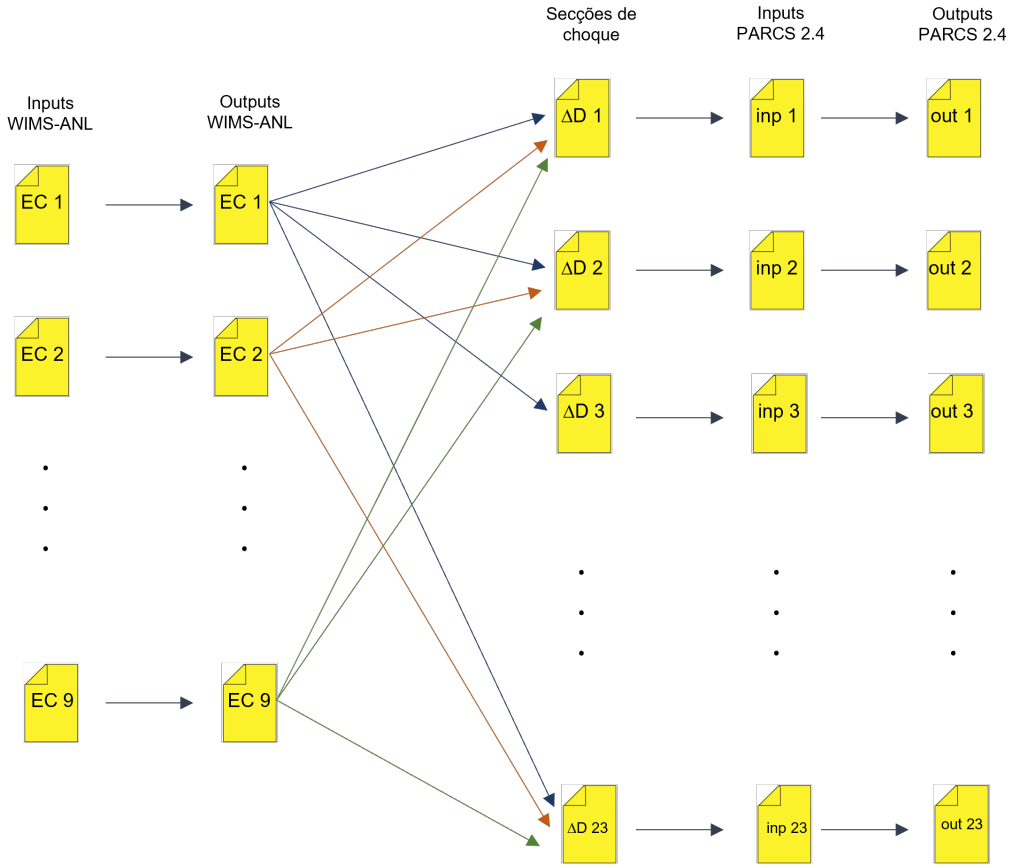


Figura 37 – Esquematização do processo de simulação do núcleo com WIMS-ANL e PARCS 2.4. para a queima do combustível Onde $\Delta D1, 2, ...$ se refere a o numero de intervalos de dias que foram utilizadas.

Na figura 37 nota-se que os valores de seção de choque para cada intervalo de tempo ($\Delta D1$, $\Delta D2, ...,$ $\Delta D23$) foram extraídos a partir de cada *output* do WIMS-ANL. Devido à grande quantidade de dados envolvidos no processo mostrado na figura 37, foi desenvolvido um script que facilita todos os cálculos e a formatação das entradas. A função *xs_fluel_assembly* (ANEXO A) lê os valores das seções de choque calculadas pelo código WIMS-ANL (saídas EC). A função *xs_fluel_assembly* retorna como saída os valores das seções de choque formatados conforme exigido no código PARCS; este processo é repetido

para cada intervalo de tempo com a função *main_XS_parcs* (ANEXO A). E finalmente os arquivos criados pela função *main_XS_parcs* são simulados com o código PARCS.

Capítulo 6

Otimização do padrão de recarga

Na seção 2.1, foi visto que as estratégias seguidas no processo de recarga de combustível dependem de muitos fatores. Neste trabalho, o objetivo é abordar o estudo do padrão de recarga, apenas do ponto de vista da configuração espacial dos elementos combustíveis, ou seja, pretende-se encontrar a configuração espacial apropriada para os parâmetros a serem maximizados ou minimizados.

Seguindo essa linha de raciocínio, abaixo está uma lista bastante simplificada das etapas usadas para atingir a meta:

I. O modelo neutrônico do reator PWR (BEAVRS) descrito no capítulo 5 é desenvolvido. Neste modelo as seguintes etapas são consideradas,

   a) As seções de choque macroscópicas são calculadas com os modelos desenvolvidos segundo o elemento combustível e suas características usando o modelo MULTICELL disponível no código WIMS-ANL;

   b) As seções de choque macroscópicas geradas são utilizadas no modelo do núcleo desenvolvido para o código PARCS, com o qual são obtidas a distribuição de potência e k_{eff}, a fim de validar o modelo e utilizá-lo para treinar o modelo.

II. O modelo do núcleo do item anterior (I.b) é usado para obter a distribuição de potência e k_{eff} de 40.000 diferentes configurações espaciais dos ECs. Com ajuda de um script feito em Python, as posições dos ECs foram geradas de forma aleatória.

III. Os resultados do item II são utilizados para treinar o modelo baseado na arquitetura "*transformers*".

O item I foi apresentado em detalhes no capítulo 5. O item II é uma etapa simples que consiste em gerar 40.000 posições aleatórias dos ECs e, em seguida, simular cada uma delas com o código PARCS seguindo a metodologia do item I. Essas simulações nos

oferecem 40.000 dados de distribuições de potência e k_{eff} de cada padrão de recarga, as quais serão o conjunto de dados para treinar o modelo de otimização.

Este capítulo apresentará a metodologia utilizada para a otimização da recarga de acordo com os itens II e III. Esses itens serão detalhados nas próximas seções.

6.1 Geração de dados de treinamento

Todo modelo supervisionado de aprendizado de máquina necessita de dados de treinamento e nesse sentido, esta seção apresenta a metodologia utilizada para gerar esses dados. A ideia principal da metodologia é simular diferentes configurações espaciais dos ECs, que foram escolhidas aleatoriamente. Obviamente, há um grande número de configurações possíveis e para reduzir esse número, o modelo será simplificado empregando a simetria de 1/8 do núcleo (Ver figura 38).

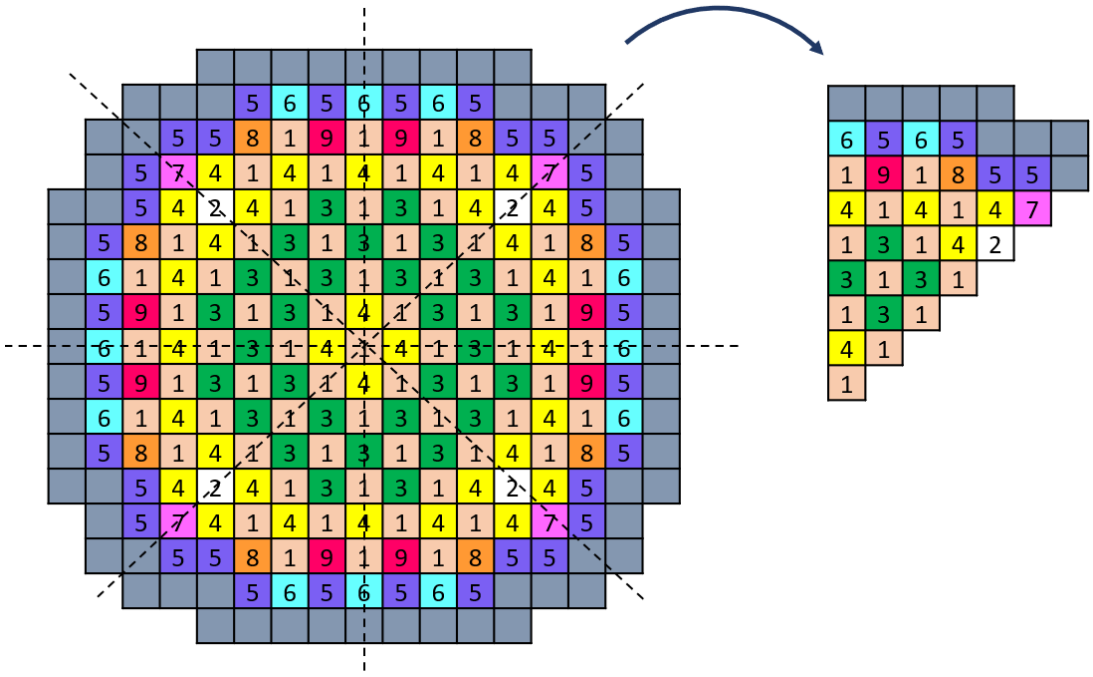


Figura 38 – Esquematização da simetria de 1/8 de nucelo.

Nesta fase, 40.000 configurações foram obtidas aleatoriamente com o *script nucleo_aleatorio* (ANEXO A); onde cada uma deles é simulado com o código PARCS. Os parâmetros da simulação são os mesmos utilizados na queima do combustível (Ver tabela 5), mas deve-se destacar que essa etapa não contempla o cálculo da evolução do combustível, apenas foram utilizados os parâmetros dessa condição, ou seja , a temperatura do combustível, a temperatura do refrigerante e a concentração de boro são as mesmas (Ver tabela 5).

Após o término dessas simulações, haverá 40.000 outputs com informações sobre a distribuição de potência radial média e os k_{eff} de cada uma das configurações. Como

estamos trabalhando com uma simetria de 1/8, apenas os dados correspondentes a essa simetria serão usados (ver figura 39). Os dados da distribuição de potência radial média serão reorganizados em uma matriz de dimensão 40000 × 31, o qual será o conjunto de dados de treinamento. Cada fila da matriz contém a informação correspondente a uma configuração dos ECs, a figura 39 esquematiza este processo.

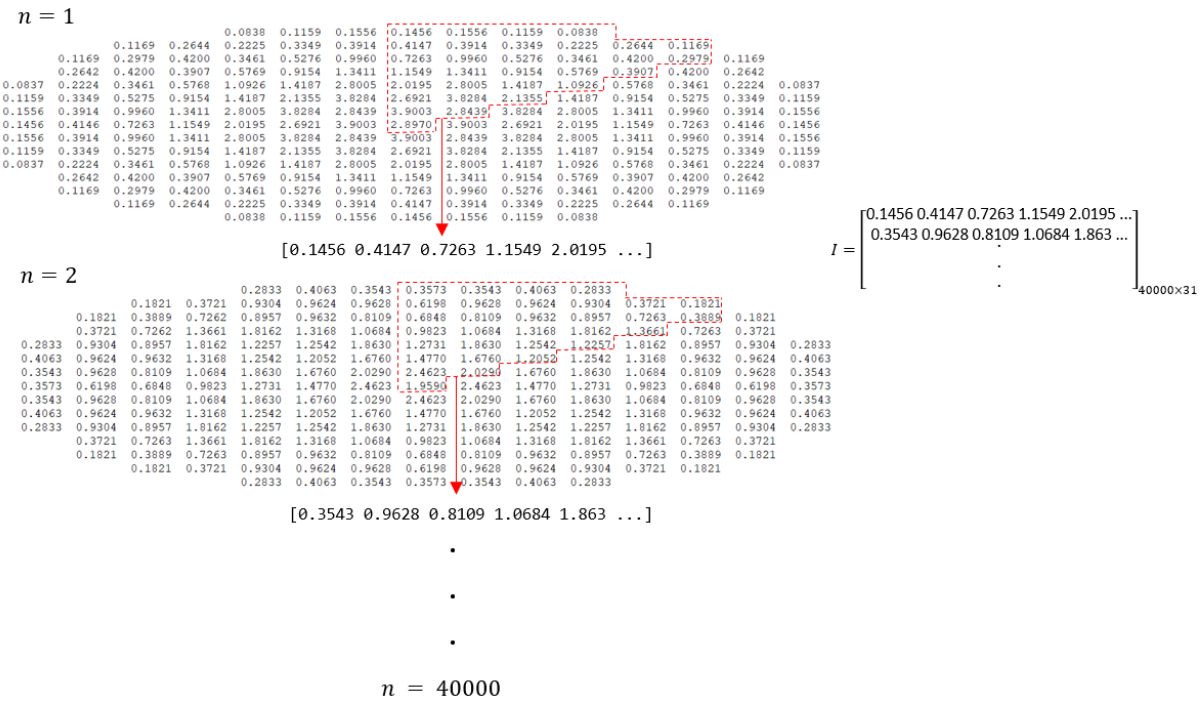


Figura 39 – Esquema de organização dos outputs (distribuição de potência radial média) do código PARCS, os quais formam uma matriz I.

A matriz I de dimensão 40000 × 31 (ver figura 39) é formada pelos *outputs* do código PARCS, isto é, cada 1/8 de distribuição de potência radial é reorganizada em um vetor de dimensão 1×31; no total são 40.000 *outputs*, pelo qual uma matriz I de dimensão 40000 × 31 será obtida. A matriz I será o *input* de treinamento do modelo.

A matriz O de dimensão 40000 × 40 (ver figura 40) é formada pelas posições dos ECs (input do código PARCS), isto é, cada 1/8 das posições dos ECs é reorganizada em um vetor de dimensão 1×31. No total são 40.000 vetores com informação das posições aleatórias dos ECs, o qual forma uma matriz de dimensão 40000 × 40. A matriz O será usada como o output de treinamento do modelo.

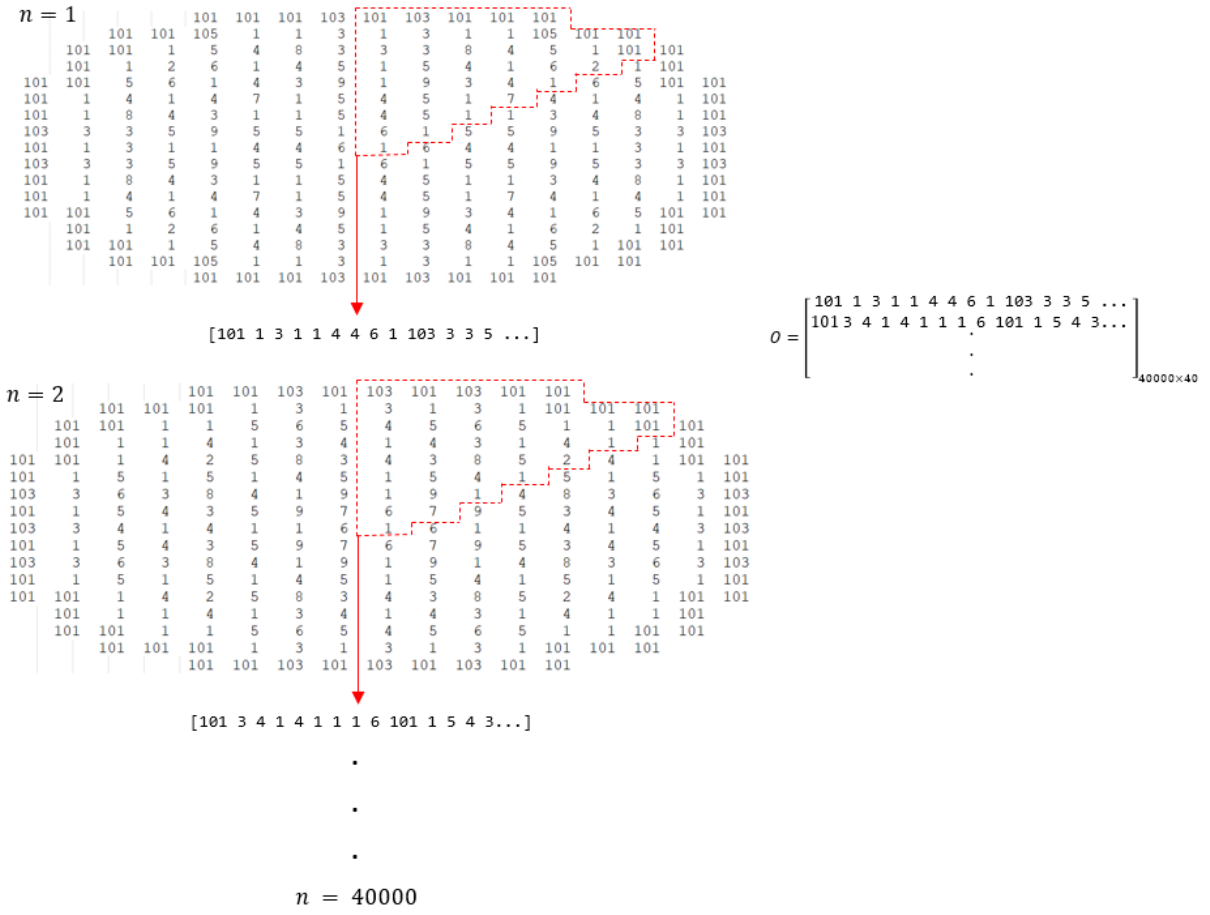


Figura 40 – Esquema de organização dos inputs (posições dos ECs) do código PARCS, os quais formam uma matriz O, onde os números de 1 a até 9 são os diferentes ECs e os maiores a 100 são os elementos refletores.

6.2 Metodologia de otimização

Esta seção apresenta os detalhes do item III, mencionado no início deste capítulo. Conforme explicado no capítulo 2, o número de configurações possíveis dos ECs é bastante elevado, e por isso, neste trabalho, 1/8 do núcleo é utilizado, o que facilitará a simulação. A simulação do núcleo do reator fornece informações sobre o comportamento da potência radial. Cada um dos núcleos simulados é diferenciado pela localização dos ECs, o que se reflete em uma variedade de distribuições de potência para cada um dos núcleos. Essas informações ajudarão a propor um modelo que nos permita encontrar uma configuração dos ECs adequada aos critérios a serem considerados.

Neste trabalho, a busca da distribuição de potencia radial que atendem as restrições impostas, serão consideradas apenas do ponto de vista da posição dos ECs; nesse sentido, a principal variável de análise são as posições dos ECs e sua correspondente distribuição de potência, pois essas duas variáveis estão extremamente ligadas.

No Capítulo 2 (secção 2.2), vimos que a distribuição de potência na parte central do núcleo deve ser o mais plana possível, e deve diminuir à medida que se aproxima da

periferia do núcleo. A figura 41 mostra uma possível distribuição de potência "ideal" com base nas características desejadas. Seguindo esse raciocínio, este trabalho busca encontrar um modelo que retorne as posições dos ECs que atendam às restrições mencionadas.

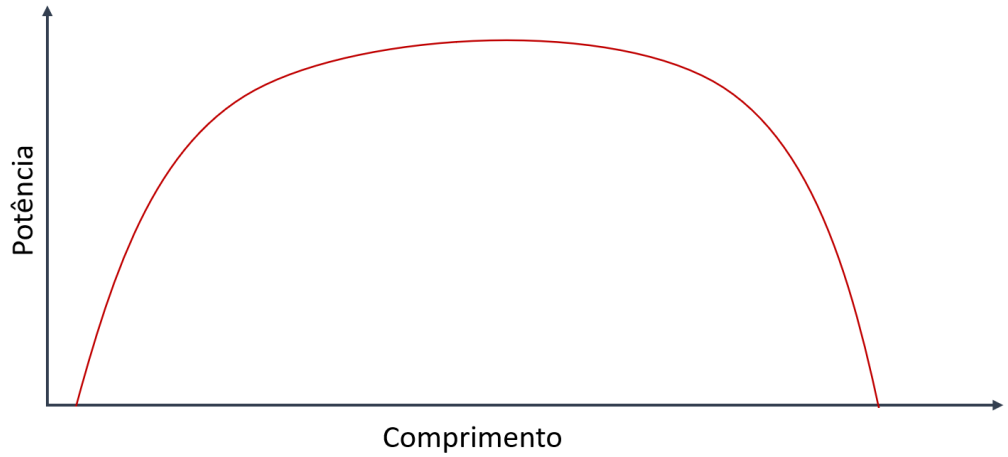


Figura 41 – Possível distribuição de potência radial média de acordo com as restrições usadas.

Então, o objetivo da metodologia é encontrar um modelo que, a partir de um *input* com informações sobre a distribuição de potência, as posições dos ECs sejam obtidas como *output* (para a distribuição de potência correspondente). Por tanto, o modelo deve ser capaz de identificar que a ordem dos ECs é uma variável importante, além do fato de haver ECs mais relevantes do que outros, ou seja, ECs de maior enriquecimento têm maior efeito em uma determinada região. Um aspecto importante é que o modelo deve compreender os rótulos dos ECs como representativos, e o número com o qual são identificados são apenas "rótulos"; ou seja, o elemento combustível 1 pode ser qualquer outro número ou mesmo uma letra.

Um modelo de aprendizado de máquina que pode ser apropriado para este tipo de dados são os "*transformers*". Este modelo é típico de processamento de linguagem natural, mas existem muitos trabalhos que utilizam este modelo para outros fins, entre eles, o mais proeminente é o trabalho de DeepMind alphafold (JUMPER et al., 2021), que desenvolveu um modelo baseado em *transformer* para predizer estruturas de proteínas biológicas.

Neste trabalho o modelo dos *transformers* será adaptado para predizer a distribuição espacial dos ECs, dada uma distribuição de potência radial média como input. Este modelo é explicado com mais detalhes na próxima seção.

6.2.1 Modelo *trasformer*

A operação fundamental de qualquer arquitetura de *transformer* é a operação de auto atenção. A auto atenção é uma operação de sequência a sequência. Vamos chamar os

vetores de entrada $x_1, x_2, \ldots x_t$ e os vetores de saída correspondentes $y_1, y_2, \ldots, y_t$. Todos eles possuem dimensão k.

Para produzir o vetor de saída y_1, a operação de auto atenção simplesmente leva uma média ponderada de todos os vetores de entrada;

$$y_i = \sum_j w_{ij} x_j \tag{6.1}$$

onde i indexa sobre toda a sequência e os pesos somam sobre j. O peso w_{ij} não é um parâmetro, como em uma rede neural normal, mas é derivado de uma função sobre x_i e x_j. A opção mais simples para esta função é o produto escalar:

$$w'_{ij} = x_i^T x_j \tag{6.2}$$

onde x_i é o vetor de entrada na mesma posição que o vetor de saída atual y_i. Para o próximo vetor de saída, obtemos uma série inteiramente nova de produtos escalares e uma soma ponderada diferente. O produto escalar dá um valor em qualquer lugar entre o infinito negativo e positivo. Então aplicamos um $softmax$ para mapear os valores para $[0, 1]$ e para garantir que eles somam 1 em toda a sequência:

$$w_{ij} = \frac{e^{w'_{ij}}}{\sum_j e^{w'_{ij}}} \tag{6.3}$$

essa é a operação básica de auto atenção (VASWANI et al., 2017).

Alguns outros ingredientes são necessários para um *transformer* completo, que discutiremos mais tarde, mas esta é a operação fundamental. Esta é a única operação em toda a arquitetura que propaga informações entre vetores. Todas as outras operações no *transformer* são aplicadas a cada vetor na sequência de entrada sem interações entre os vetores.

Apesar de sua simplicidade, não é imediatamente óbvio entender por que a auto-atenção funciona. Para entender de forma intuitiva, podemos analisar o seguinte exemplo. Digamos que estamos diante de uma sequência de palavras. Para aplicar a auto atenção, simplesmente atribuímos a cada palavra t em nosso vocabulário um *embedding* vetor v_t (cujos valores aprenderemos). Isso é conhecido como uma camada de incorporação na modelagem de sequência. Ele transforma a sequência de palavras *o, gato, caminha, na, rua* na sequência vetorial

$$v_o, v_{\text{gato}}, v_{\text{caminha}}, v_{na}, v_{\text{rua}}$$

Se alimentarmos essa sequência em uma camada de autoatenção, a saída será outra sequência de vetores

$$y_o, y_{\text{gato}}, y_{\text{caminha}}, y_{\text{na}}, y_{\text{rua}}$$

onde y_{gato} é uma soma ponderada de todos os vetores *embedding* na primeira sequência, ponderada por seu produto escalar (normalizado) com v_{gato} .

Sabe-se que, a relação de duas palavras em uma oração simples, está inteiramente determinado pelo verbo; na maioria dos casos, o artigo definido não é muito relevante para a interpretação das outras palavras da frase; portanto, provavelmente acabaremos com um *embedding* v_0 que tem um produto escalar baixo ou negativo com todas as outras palavras. Por outro lado, para interpretar o que significa *caminha* nesta frase, é muito útil descobrir quem está fazendo a ação. Provavelmente, isso é expresso por um substantivo, portanto, para substantivos como gato e verbos como caminhar, provavelmente aprenderemos os *embedding* y_{gato} e $y_{caminha}$ que têm um produto escalar positivo e alto juntos.

Esta é a intuição básica por trás da auto atenção. O produto escalar expressa como estão relacionados dois vetores na sequência de entrada, e os vetores de saída são somas ponderadas sobre toda a sequência de entrada, com os pesos determinados por esses produtos escalares.

Antes de prosseguirmos, vale a pena observar as seguintes propriedades, que são incomuns para uma operação sequência a sequência:

- Não há parâmetros (ainda). O que a auto atenção básica realmente faz é inteiramente determinado por qualquer mecanismo que crie a sequência de entrada. Os mecanismos *upstream*, como uma camada de incorporação, conduzem a auto atenção ao aprender representações com produtos de ponto específicos.
- A auto atenção vê sua entrada como um conjunto, não uma sequência. Se permutamos a sequência de entrada, a sequência de saída será exatamente a mesma, ou seja, a auto atenção é equivariada em permutação. Vamos mitigar isso um pouco quando construirmos o transformador completo, mas a auto atenção por si só na verdade ignora a natureza sequencial da entrada.

A auto atenção real usada nos *transformers* modernos depende de três passos adicionais.

6.2.1.1 Consultas, chaves e valores

Cada vetor de entrada x_i é usado de três maneiras diferentes na operação de auto-atenção:

- É comparado a todos os outros vetores para estabelecer os pesos para sua própria saída y_i,
- É comparado a todos os outros vetores para estabelecer os pesos para a saída do j-ésimo vetor y_i,

- É usado como parte da soma ponderada para calcular cada vetor de saída, uma vez que os pesos tenham sido estabelecidos.

Essas funções costumam ser chamadas de consulta, chave e valor. Na auto-atenção básica que vimos até agora, cada vetor de entrada deve desempenhar todas as três funções. Facilitamos os cálculos derivando novos vetores para cada função, aplicando uma transformação linear ao vetor de entrada original. Em outras palavras, adicionamos três matrizes $k \times k$ de peso W_q, W_k, W_v e calculamos três transformações lineares de cada x_i, para as três partes diferentes da auto atenção:

$$q_i = W_q x_i \quad k_i = W_k x_i \quad v_i = W_v x_i$$

$$w'_{ij} = q_i^T k_j$$

$$w_{ij} = \text{softmax}\left(w'_{ij}\right)$$

$$y_i = \sum_j w_{ij} v_j \tag{6.4}$$

Isso fornece à camada de auto-atenção alguns parâmetros controláveis e permite que ela modifique os vetores de entrada para se adequar às três funções que devem desempenhar.

6.2.1.2 Dimensionamento do produto escalar

A função $softmax$ pode ser sensível a valores de entrada muito grandes. Eles eliminam o gradiente e tornam o aprendizado mais lento ou fazem com que ele pare completamente. Uma vez que o valor médio do produto escalar cresce com a dimensão de incorporação k, ajuda a dimensionar o produto escalar um pouco para trás para impedir que as entradas para a função $softmax$ cresçam muito:

$$w'_{ij} = \frac{q_i^T k_j}{\sqrt{k}} \tag{6.5}$$

Para entender porque do $\sqrt{k}$, imagine um vetor em $\mathbb{R}^k$ com valores todos c. Seu comprimento euclidiano é $\sqrt{k}c$. Portanto, estamos dividindo o valor pelo qual o aumento na dimensão aumenta o comprimento dos vetores médios.

6.2.1.3 Atenção multi-cabeça

Podemos dar à auto atenção um maior poder de discriminação, combinando vários mecanismos de auto-atenção (que indexaremos com r), cada um com diferentes matrizes W_{rq}, W_{rk}, W_{rv}. Estes são chamados de cabeças de atenção.

A maneira mais simples de entender a auto-atenção com várias cabeças é vê-la como um pequeno número de cópias do mecanismo de auto atenção aplicado em paralelo,

cada uma com sua própria chave, valor e transformação de consulta. Isso funciona bem, mas para cabeças de número R, a operação de autoatenção é R vezes mais lenta.

Há uma maneira de implementar a auto-atenção com várias cabeças de modo que seja quase tão rápido quanto a versão com uma cabeça, mas ainda tem-se o benefício de ter diferentes matrizes de atenção em paralelo. Para fazer isso, divide-se cada vetor de entrada em vetores de menor tamanho: se o vetor de entrada tem 256 dimensões e temos 8 cabeças de atenção, nós o cortamos em 8 vetores menores de 32 dimensões. Para cada bloco, geramos chaves, valores e consultas de 32 dimensões cada. Isso significa que as matrizes W_{rq}, W_{rk}, W_{rv} são todas 32×32.

6.2.2 Arquitetura do *transformer*

Um *transformer* não é apenas uma camada de auto atenção, é uma arquitetura. Não está muito claro o que se qualifica ou não como um *transformer*, mas aqui usaremos a seguinte definição:

Qualquer arquitetura projetada para processar um conjunto conectado de unidades, como os "tokens" em uma sequência, em que a única interação entre as unidades é por meio da auto atenção.

Tal como acontece com outros mecanismos, como convoluções, uma abordagem padrão surgiu para construir camadas de auto-atenção em uma rede maior. O primeiro passo é envolver a auto atenção em um bloco que podemos repetir.

Existem algumas variações sobre como construir um bloco do *transformer* básico, mas a maioria delas é estruturada aproximadamente é como se mostra na figura 42.

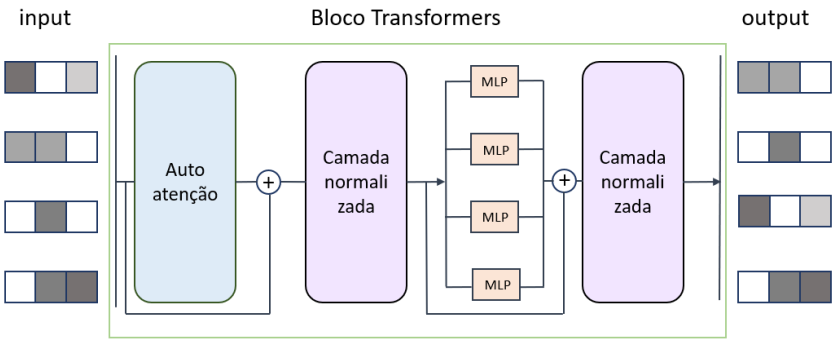


Figura 42 – Esquema da arquitetura geral de um modelo *transformer*.

A arquitetura consiste em uma cada de auto atenção, uma camada normalizada, uma cada de RNA *feed-forward* (um MLP é aplicado independentemente a cada vetor) e uma ultima camada normalizada. As conexões residuais são adicionadas em torno de ambos, antes da normalização. A ordem dos vários componentes não é imutável; o importante é combinar auto atenção com *feed-forward* local e adicionar normalização e conexões residuais.

Neste trabalho usou-se a arquitetura aprestado no trabalho original de Vaswani (VASWANI et al., 2017), o modelo é apresentado na figura 43.

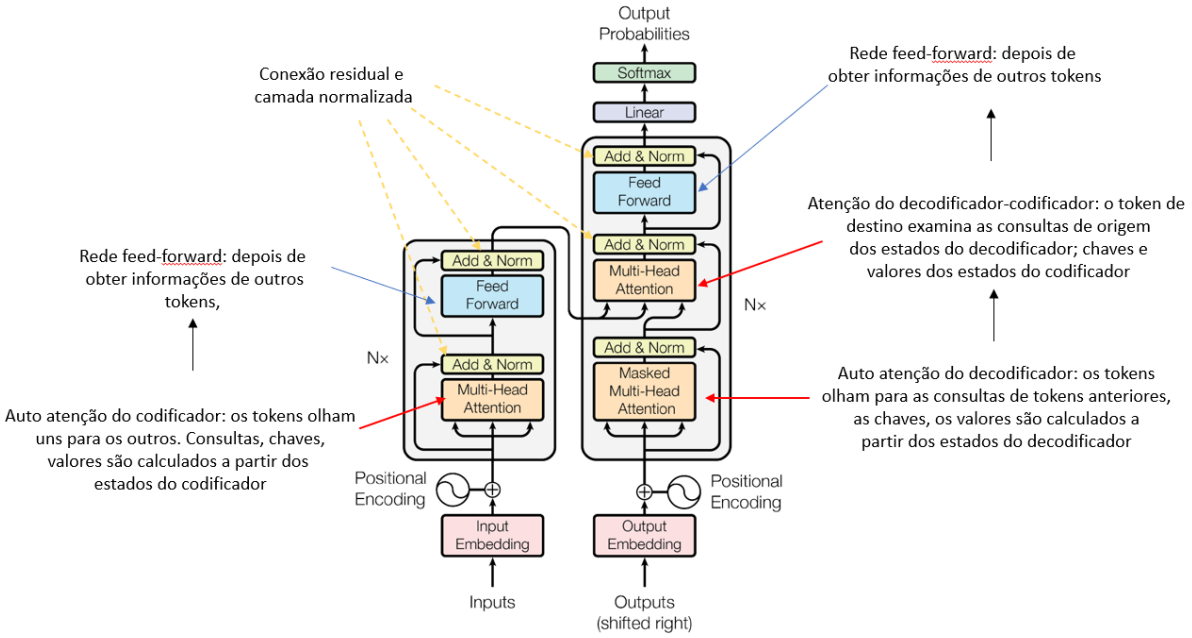


Figura 43 – Esquema da arquitetura original *transformer* apresentada no trabalho de Vaswani (VASWANI et al., 2017).

A arquitetura *transformer* (VASWANI et al., 2017) será usada para obter a configuração espacial dos ECs, dado um input com a distribuição de potencia radial. Serão usados 40000 configurações dos ECs (output) e suas correspondestes distribuições de potência (input) como dados de treinamento. O código desenvolvido neste trabalho foi completamente adaptado do *scrpit* publicado por *TensorFflow*, disponibilizado em (TENSORFLOW, 2021).

6.2.3 Uso do modelo *trasformer*

A arquitetura *trasformer* é amplamente utilizada na área de processamento de linguagem natural. As ferramentas de tradução de texto fazem uso dessa arquitetura, pois esse modelo é eficiente para encontrar relacionamentos sequenciais. Neste trabalho o modelo é utilizado da mesma forma que é utilizado em um tradutor de texto. No nosso caso, o “texto de entrada” (o que se pretende traduzir) será a distribuição da potência radial média e o "texto de saída"(texto traduzido) será a distribuição espacial dos elementos combustíveis.

A figura 44 descreve o processo de simulação. A figura mostra como os dados de distribuição de potência radial média (Matriz I 6.1, veja seção) são usados como entrada para o modelo, e os padrões de recarga de EC são usados como saída (Matriz O, veja

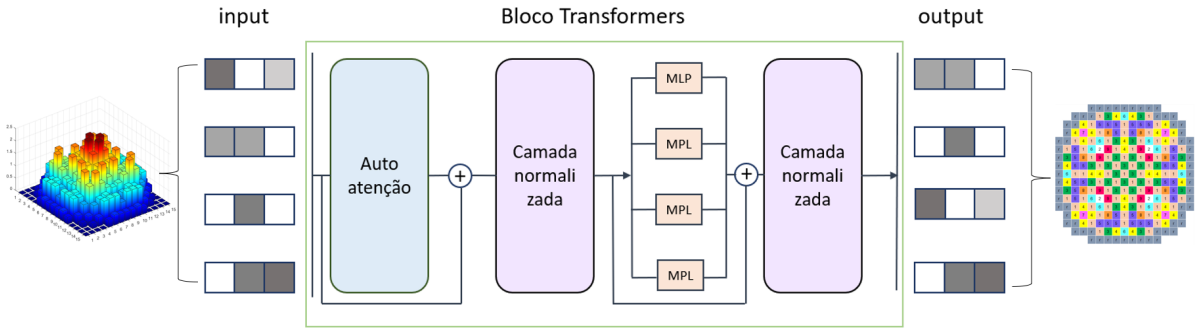


Figura 44 – Esquema do modelo *transformer* usado neste trabalho.

seção 6.1). Após treinar o modelo, ele será capaz de identificar qual é o padrão de recarga para uma determinada distribuição de potência radial média.

O modelo é capaz de fazer isso porque, da mesma forma que em uma frase o verbo é o componente com maior peso, e faz a frase ter sentido; em um padrão de recarga existem elementos combustíveis que têm um efeito maior na distribuição da potência radial. E a posição destes é importante para que um padrão de recarga se comporte de uma certa maneira. Nesse sentido, de forma genérica, podemos dizer que o modelo desenvolvido neste trabalho é um "tradutor", que nos leva uma potência radial média ao seu padrão de recarga correspondente.

6.3 Ferramentas usadas

Este trabalho consiste na utilização de técnicas de *Machine Learning*, conforme já mencionado no Capítulo 1, as quais precisam de dados para obter modelos do sistema estudado. No caso do estudo de otimização da recarga para um reator nuclear, os dados foram obtidos a partir da simulação de diferentes distribuições do padrão de recarga, utilizando os códigos WIMS-ANL e PARCS.

6.3.1 PARCS

PARCS é um código de cálculo neutrônico que resolve problemas de estado estacionário no espaço tridimensional e problemas com dependência temporal. Ele resolve equações de difusão de multigrupo em geometria cartesiana, cilíndrica ou hexagonal. Existe uma versão (v3.0) acoplada diretamente ao código de cálculo termo-hidráulico TRACE para cálculo de transientes (T. Downar, Y. Xu, 2010), podendo também ser acoplada ao código RELAP5 por meio de uma máquina virtual paralela (PVM) (T. Downar, Y. Xu, 2006).

A resolução de problemas cinéticos espaciais, aborda problemas físicos como o transporte de nêutrons. O problema do reator crítico associado a equação de transporte

dependente do tempo é resolvida. O primeiro passo para solucionar esses problemas é discretizar o espaço e, no caso da dependência do tempo, também o tempo. Para discretizar o tempo, PARCS usa o método *alpha2* além de uma integração analítica de precursores de segunda ordem. Se o comportamento exponencial for previsto, uma transformação exponencial será aplicada. A discretização de tempo permite grandes intervalos de tempo, mesmo para transientes envolvendo inserções de reatividade. Para discretização espacial, o código PARCS usa uma nodalização na qual o método CMFD (*Coarse Mesh Finite Diffusion*) é resolvido e o método de dois nós é usado para encontrar o coeficiente de correção para correntes internodais.

6.3.2 WIMS

O WIMS usa a teoria do transporte para calcular o fluxo de nêutrons em função da energia e das coordenadas espaciais em uma célula unidimensional. Duas das opções de transporte mais comumente usadas são DSN e PERSEUS (probabilidades de colisão). A solução de transporte pode ser executada com qualquer grupo de energia especificada pelo usuário até o número máximo de grupos de bibliotecas disponíveis. Esta solução de transporte principal é precedida por um ou mais cálculos de espectro de energia do fluxo SPECTROX. As soluções desta etapa de cálculo são usadas para calcular os dados de seção de choque no número de grupos definidos pelo usuário (FEWGROUPS), as quais são usadas na sequência da solução de transporte principal, de modo que WIMS produz um autovalor estimado de SPECTROX, assim como o autovalor de transporte na seqüência principal. A versão WIMS-ANL também tem uma solução homogênea de fluxo médio infinito usando o método de transporte DSN ou PERSEUS (DEEN et al., 2004). Vale ressaltar que existe uma grande variedade de versões, cada uma modificada a partir da versão original do WIMS.

O WIMS inclui algoritmos de probabilidade de colisão para modelar células de barras de combustível em uma matriz regular (x, y) ou grupos de barras em uma matriz (r, θ). Os algoritmos não fornecem uma solução de fluxo bidimensional precisa, mas fornecem uma boa aproximação. Este método de solução está incluído no WIMS-ANL.

No WIMS-ANL, após a conclusão da solução de transporte, as seções de choque do grupo são calculadas em grupos amplos (< 20 grupos) e podem ser escritas no formato ISOTXS com dados de seção de choque microscópicas ou macroscópicas que podem ser usados em outros códigos de transporte ou difusão. Neste trabalho utiliza-se a versão WIMS-ANL, pois o arquivo de saída está otimizado para facilitar ao usuário encontrar as variáveis de interesse. Outro motivo é a possibilidade de usar uma biblioteca de seção de choque microscópica conforme o interesse do usuário, neste caso ENDF/B-VI.

6.3.3 PYTHON

Python é uma linguagem de programação de alto nível que consiste em uma sintaxe muito fácil de entender. Foi criado no início dos anos 90 na Holanda e é considerado um projeto relativamente jovem em comparação com outras linguagens, como Pascal, Fortran e Java etc. Por outro lado, Python se caracteriza por possuir algumas características de linguagens que o precedem.

Vantagens

As principais vantagens desta linguagem são as seguintes

- **Estilo flexível**: oferece muitas ferramentas para criar código de maneira flexível. Por exemplo, para usar uma lista de vários tipos de dados, não precisa declarar cada um deles (isso é feito para determinar a classe dos dados). Por outro lado, a sintaxe é compreensível e em algumas funções se assemelham a estruturas de outras linguagens.
- **Ordenado e limpo**: linguagem legível e compreensível para qualquer programador que queira trabalhar em uma estrutura já estabelecida e isso se deve à organização de seus módulos.
- **Comunidade ativa**: Ao contrário de outros tipos menos populares de linguagens, Python tem uma comunidade ativa de usuários comprometidos em ajudar com as atualizações.
- ***Open Source***: esta linguagem de programação é um software livre.
- **Simplificado e rápido**: possui vários padrões orientados para a ação. E por se tratar de uma linguagem interpretada, sua execução é rápida, pois não precisa ser compilada. Com isso economiza-se muito tempo para programar e criar projetos.
- **Multiplataforma**: Este linguajem pode ser usado em vários sistemas operacionais como Linux, Windows ou Mac OS. Por outro lado, inclui as bibliotecas mais populares dentro do intérprete, para que não se perca tempo instalando-as como ocorre com outras linguagens.

Desvantagens

Embora tenha muitas vantagens, é importante que conhecer suas desvantagens:

- **Lentidão ao executar vários *threads*:** embora não precise ser compilado, se houver necessidade de executar vários *threads* de programação, pode não aproveitar todo o potencial do hardware. Por exemplo, pode-se obter erros do interpretador ou simplesmente ter problemas para usar todos os núcleos do processador.

- **Documentação:** o Python não possui documentação, então pode ter-se problemas para entender algumas bibliotecas e certas estruturas comparado com outras linguagens de programação como Java ou C++.
- **Identificadores protegidos:** ao contrário de outros tipos de linguagens, Python não possui identificadores protegidos, portanto os métodos usados são públicos.
- **Simulações:** para simulações físicas a linguagem Python pode ser complexa, pois não funciona com matrizes por padrão, como acontece com outras linguagens como o Matlab. Em suma, esta linguagem é útil desde que não dependa de uma matriz ou tenha que trabalhar com um vetor complexo, caso contrário deve-se importar bibliotecas.

Neste trabalho Python é utilizado para automatizar a montagem dos arquivos de entrada do WIMS-ANL e PARCS. No texto de referência de BEAVRS (HORELIK et al., 2013), podemos ver que a quantidade de dados fornecidos é bastante grande. Então, a organização deles manualmente torna-se difícil, pois tratamos com um número grande de materiais que dependem de etiquetas, densidades, concentrações, geometrias, etc. Além disso, WIMS-ANL e PARCS usam-se de forma conjunta em processo interativo, ou seja, os dados de um processo dependem da anterior. Essas dificuldades tornam a Python uma excelente alternativa para automatizar o processo.

6.3.4 PYTORCH

O foco principal deste trabalho é o uso de modelos *Machine Learning* para abordar o estudo da optimização de recarga. O campo do *Machine Learning* reúne diversos modelos matemáticos, e programar todas elas seriam uma tarefa muito demorada. Afortunadamente existem pacotes em Python que facilitam o uso desses modelos. Existem várias opções em termos de tecnologias e bibliotecas, sendo *Tensorflow*, desenvolvido pelo Google, a mais difundida atualmente. No entanto, neste trabalho se usará *PyTorch*, uma alternativa emergente que está ganhando força rapidamente graças à sua facilidade de uso e outras vantagens, como sua capacidade nativa de rodar na GPU, que permite acelerar processos tradicionalmente lentos, como formação de modelos.

Seus elementos fundamentais são os tensores, que podem ser igualados a vetores de uma ou mais dimensões. *PyTorch* tem uma interface muito simples para criar redes neurais, apesar de trabalhar diretamente com tensores sem a necessidade de uma biblioteca de nível superior, como *Keras* para *Theano* ou *Tensorflow*. O *PyTorch* usa internamente CUDA para rodar com a GPU, uma API que conecta a CPU à GPU que foi desenvolvida pela NVIDIA.

Parte IV

Resultados

Capítulo 7

Resultados das simuações neutrônicas

Este capítulo apresenta os resultados das simulações nuetrônicas do PWR, fazendo uso dos dados que são apresentados no BEAVRS, as quais posteriormente serão utilizadas para o treinamento do algoritmo baseado em *transformer*. Os resultados aqui apresentados visam a validação da simulação neutrônica do WIMS-ANL e do PARCS.

7.1 Seções de choque macroscópicas

De acordo com a metodologia apresentada na seção 5.1, diferentes tipos de células foram escolhidas para cada elemento combustível (ver figura 28 a figura 33). Essas células interagem entre si pela probabilidade de interação, ou seja, a probabilidade de que um nêutron que deixe o tipo de célula i entre imediatamente no tipo de célula j (P_{ij}). O cálculo dessas probabilidades é puramente geométrico.

Os valores apresentados nas tabelas 6 a 11, mostram como algumas das células não interatuam com outras, e os valores correspondentes a eles sãos zeros. É evidente que estes valores mudaram conforme à escolha das células, e em nosso caso, as seleções foram feitas como mostram nas figuras 28. a 33.

Esta primeira etapa consistiu no cálculo das seções de choque macroscópicas e na validação da metodologia empregada para gerá-las, com objetivo de utilizá-las. na simulação do núcleo ativo do reator com o código PARCS. Os cálculos foram feitos para as condições de HZP (*Hot Zero Power*), onde a temperatura do combustível e do refrigerante é 566 K. Devido à grande quantidade de dados das seções de choque, nesta seção apresentaremos apenas os valores de k_{eff}, isso para mostrar que os valores são consistentes, os quais foram comparados com outros trabalhos. Nesta primeira etapa, foi simulada a configuração do ciclo 1 (HORELIK et al., 2013), que consiste em 9 elementos combustíveis com diferentes enriquecimentos e diferentes números de venenos queimáveis em cada elemento combustível, conforme mostrado na figura 5.

Na literatura, podemos encontrar alguns trabalhos que simulam o BEAVRS. Na

maioria deles, simulam a queima de combustível, mas o trabalho de (RYU et al., 2015) nos ajuda a validar a metodologia que foi utilizada. Ryu apresenta os valores de k_{eff} dos ECs obtidos com o código nTRANCER e McCARD. A tabela 12 apresenta os valores de k_{eff} calculados pelos códigos nTRACER e McCARD, nos trabalhos de referência, e os resultados calculados pelo código WIMS dessa presente pesquisa.

Tabela 12 – Resultados dos k-eff para os diferentes elementos combustíveis, obtidos com os códigos WIMS-ANL, nTRANCER e McCARD ((RYU et al., 2015)).

EC	w/o U235	VQ	WIMS-ANL	nTRACER	McCARD	Dev.Std
EC 1	1,6	0	0,998173	0,99340	0,99483	0,0024496
EC 2	2,4	0	1,138959	1,13696	1,13751	0,0010326
EC 3	2,4	12	1,013309	1,01318	1,01432	0,0006243
EC 4	2,4	16	0,979513	0,97454	0,97597	0,0025602
EC 5	3,1	0	1,219676	1,21971	1,21971	1,963E − 05
EC 6	3,1	6	1,154867	1,16273	1,16313	0,0046595
EC 7	3,1	15	1,072441	1,07835	1,07895	0,0035973
EC 8	3,1	16	1,066354	1,06340	1,06416	0,0015339
EC 9	3,1	20	1,028134	1,02769	1,02886	0,0005906

Na tabela 12 pode-se observar que os k_{eff} obtidos com WIMS-ANL, são muito semelhantes comparados com os resultados obtidos com os códigos nTRANCER e McCARD. Isso se reflete nos valores do desvio padrão. Apesar de utilizarem diferentes bibliotecas de seções de choque microscópicas, os resultados são bastante similares. No caso do trabalho de Ryu et al. a biblioteca usada é ENDF-B / VII e neste trabalho se usou ENDF / B-VI. Os resultados apresentados neste trabalho, podem-se considerar válidos para o objetivo proposto.

No arquivo de saída do WIMS, podemos encontrar as seções de choque para cada elemento combustível, com as quais podemos simular o núcleo ativo do reator. Esses resultados não incluem as condições de contorno, ou seja, o refletor do núcleo não é levado em consideração. Para ter um modelo completo é necessário obter as seções de choque do refletor. Devido à necessidade de inclusão do refletor, adotamos a metodologia sugerida no manual do usuário WIMS-ANL (DEEN et al., 2004), que consiste em semi-homogeneizar os elementos combustíveis vizinhos ao refletor (Ver figura 34). Esta semi-homogeneização permite que o WIMS use o espectro de energia dos nêutrons que provêm do elemento combustível, para obter apenas as seções de choque macroscópicas homogeneizadas do refletor.

O processo de homogeneização (seguindo parâmetros geométricos) também é uma alternativa para calcular as seções de choque de cada EC. É importante ressaltar que o processo de homogeneização é feito em qualquer um dos casos do WIMS-ANL. A diferença é que no modelo MULTICELL não é realizada homogeneização prévia, pois espera-se que o código escolha a melhor estratégia. No caso da semi-homogeneização mostrada na figura

35, ela é feita seguindo a estratégia geométrica na descrição apresentada nessa figura 35.

Neste trabalho, o modelo MULTICELL foi utilizado para calcular as seções de choque de cada EC, e a estratégia de semi-homogeneização prévia foi utilizada para calcular as seções de choque dos elementos refletores. Nesse processo, foi possível calcular as seções de choque de cada EC. Embora tais dados sejam utilizados apenas para o cálculo das seções de choque do refletor, é interessante comparar os resultados do k_{eff} para as duas diferentes geometrias consideradas para a simulação do elemento combustível. A tabela 13 mostra os resultados dos *k-eff's* dos ECs com o método de semi-homogeneização prévia.

Tabela 13 – Resultados de k_{eff} para os elementos combustíveis homogeneizados, também se mostra os resultados de Ryu (RYU et al., 2015).

EC	w/o U235	VQ	WIMS[1]	WIMS[2]	nTRACER	McCARD	Dev.Std
EC 1	1,6	0	0,998531	0,998173	0,99340	0,99483	0,00265
EC 2	2,4	0	1,139025	1,138959	1,13696	1,13751	0,00107
EC 3	2,4	12	0,950450	1,013309	1,01318	1,01432	0,03655
EC 4	2,4	16	0,920691	0,979513	0,97454	0,97597	0,03151
EC 5	3,1	0	1,211479	1,219676	1,21971	1,21971	0,00475
EC 6	3,1	6	1,098588	1,154867	1,16273	1,16313	0,03715
EC 7	3,1	15	0,979611	1,072441	1,07835	1,07895	0,05718
EC 8	3,1	16	1,005246	1,066354	1,06340	1,06416	0,03380
EC 9	3,1	20	0,936386	1,028134	1,02769	1,02886	0,05305

WIMS[1]: k_{eff}'s obtido com a homogeneização.
WIMS[2]: k_{eff}'s obtido com o modelo MULTICELL.

A homogeneização oferece excelentes resultados quando os elementos combustíveis não contêm venenos quemáveis. Mas não se obtém bons resultados quando se consideram os venenos quemáveis, que pode ser observada nos valores dos $k - eff's$ (Tabela 12 e 13). Apesar dessa diferença, sabemos que o elemento combustível predominante no núcleo, são aqueles que não contêm venenos quemáveis, e por esse motivo espera-se que essas diferenças não influenciem muito na simulação de todo o núcleo com o código PARCS.

Os resultados das seções de choque não estão incluídos neste texto, devido à grande quantidade de dados. As seções de choque do refletor serão incluídas no arquivo de entrada do PARCS, sendo que esses elementos possuem dimensões iguais aos elementos combustíveis.

7.2 Modelo do núcleo com PARCS

O modelo do núcleo com PARCS, terá a mesma nodalização radial que o padrão de recarga do núcleo, um nó para cada EC e um total de 257 nós (Figura 36). A nodalização axial consiste em 52 níveis axiais da parte ativa do núcleo de igual comprimento e dois nós refletores, superior e inferior (ver figura 34).

7.2.1 Resultados do *Hot Zero Power*

A figura 45 apresenta a distribuição axial das medições dos detectores (círculos laranja) fornecidas com a especificação BEAVRS (estado *Hot Zero Power*), as quais são comparadas com a distribuição axial de potência calculado com PARCS. Os resultados apresentam uma concordância aceitável com uma ligeira subestimação da potência na parte inferior do núcleo. O motivo é a modelagem do refletor inferior, visto que o mesmo modelo foi usado para o refletor superior e inferior; já que existe qdiferença entre as dimensões do refletor superior e inferior no BEAVRS. Também é possível observar a ausência de pequenas depressões de fluxo devido à inexistência das grades espaçadoras no modelo PARCS.

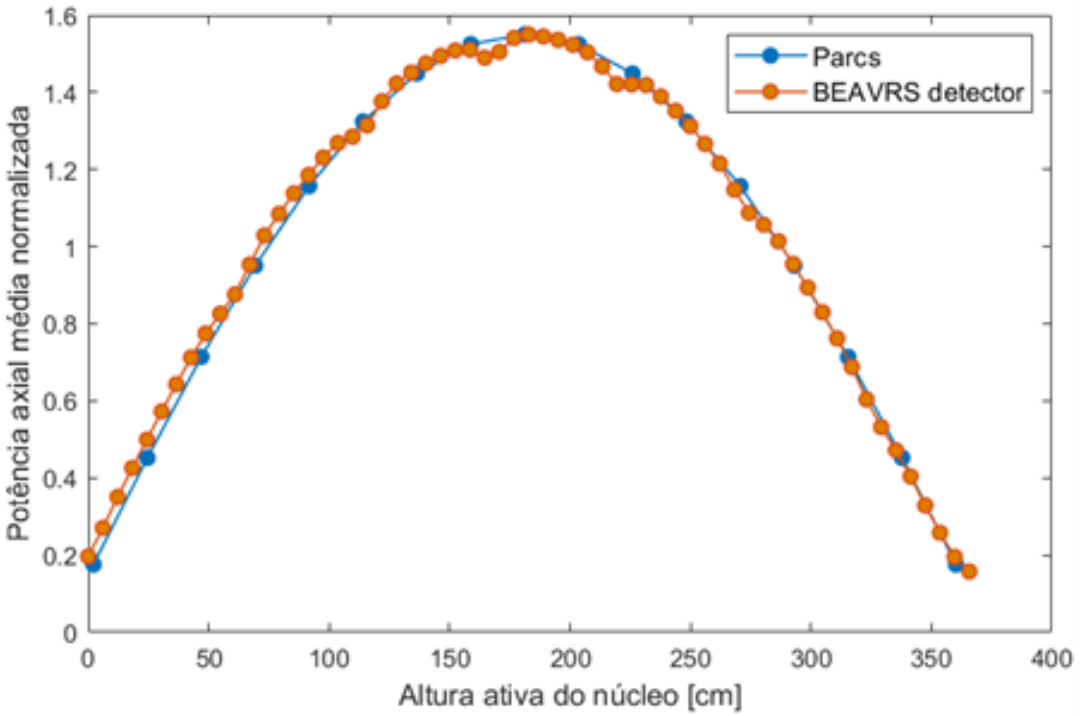


Figura 45 – Comparação das medições de potência axial de BEAVRS (pontos laranjas) com a potência calculada por PARCS (azul).

A figura 46a mostra a distribuição de potência radial média obtida por Zhuo Li (LI et al., 2016) para o estado *Hot Zero Power*, e na figura 46b mostra a distribuição de potência obtida com PARCS. Tais figuras apresentam um comportamento similar da distribuição de potência. O ideal seria comparar os duas figuras com dados explícitos, mas os dados não foram publicadas na literatura.

Os resultados da distribuição axial e a distribuição radial, nos permite validar o modelo usado neste trabalho. A validação do modelo ajudará a obter resultados confiáveis durante o treinamento do modelo *transformer*. Este modelo será usado para gerar novas distribuições espaciais dos elementos combustíveis, os quais serão simulados novamente no código PARCS.

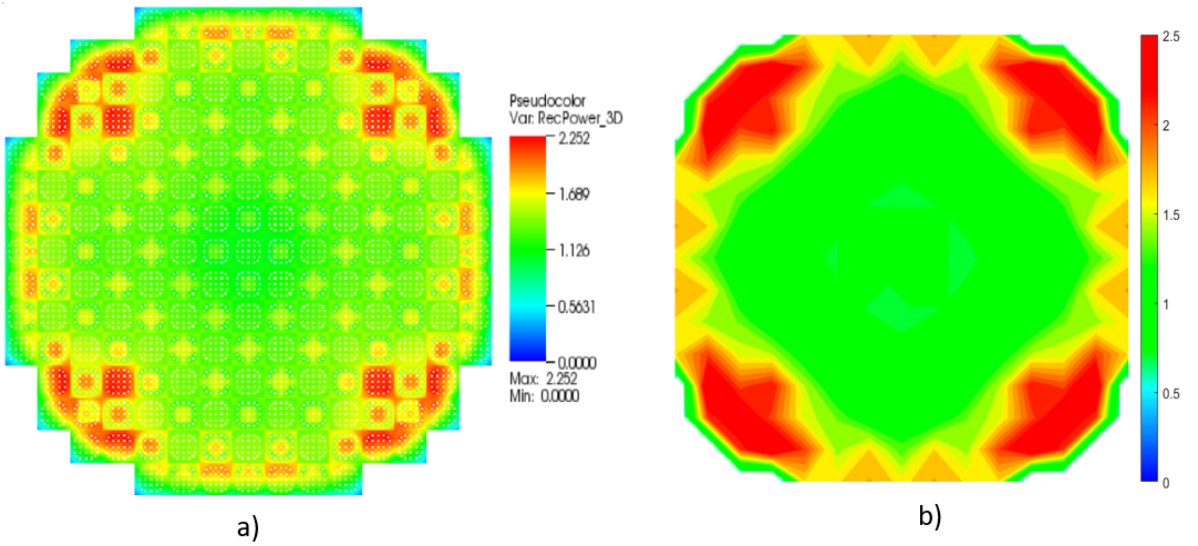


Figura 46 – a) Distribuição de potência radial obtida por Li et al. (LI et al., 2016). b) Distribuição de potência radial média obtida como PARCS

Outro parâmetro de interesse para este trabalho, é o valor de k_{eff} o qual foi calculado pelo PARCS. A tabela 14 apresenta os valores de k_{eff} calculados pelo PARCS e por outros códigos usados em outros trabalhos e apresentados por (WANG et al., 2018).O desvio padrão dos valores de k_{eff} é de 0,00197, esse valor mostra que o resultado obtido com o modelo utilizado neste trabalho, está muito próximo de outros valores encontrados na literatura.

Tabela 14 – Valores de k-eff do nucelo obtidos com PARCS e outros códigos apresentados no trabalho de Wang (WANG et al., 2018)

	PARCS	SuperMC	OpenMC	JMCT	VERA-CS	nTRACER	Dev. Std.
k-eff	1, 005531	1, 00204	1, 0008	1, 00217	0, 99972	1, 00147	0, 00197

7.2.2 Resultados do primeiro ciclo de operação

Esta seção apresenta os resultados da queima de combustível do primeiro ciclo de operação. O cálculo da evolução do combustível consistiu em 23 etapas de tempo, que são compostas por 575 dias de operação, sendo que 326 dias o núcleo estava a 100% da potência. As especificações principais para cada intervalo de tempo são descritas na tabela 15. Ela apresenta a potência média para cada intervalo de tempo, a temperatura dos refrigerante e a posição dos bancos de barras de controle.

A simulação consistiu em duas etapas. Na primeira calculou-se as seções de choque para cada intervalo de tempo e para cada EC com o código WIMS-ANL. Na segunda fase, as seções de choque geradas pelo WIMS-ANL foram usadas como *input* para o código PARCS. Esses *inputs* se separaram para cada intervalo de tempo, donde especificou-se as correspondentes posições das barras de controle (ver figura 37).

Tabela 15 – Parâmetros usados na queima do combustível para o primeiro ciclo (HORELIK et al., 2013).

Dias	Potência MWt	Potência %	Temp. Refrig. [F]	Boro [ppm]	A	B	C	D
0	24,07	0,706	558,875	975,25	228	228	228	207
7	119,61	3,507	558,125	876,79	229	176	69	215
18	692,69	20,31	554,275	801,36	229	228	228	165
54	1105,96	32,42	557,975	754,69	228	228	229	162
62	1660,93	48,69	557,35	703,12	228	228	229	228
66	1668,06	48,90	557,425	700,00	228	228	228	204
81	2525,37	74,04	560,125	652,15	228	228	228	192
82	2495,80	73,17	558,775	643,54	228	228	228	158
88	3051,70	89,47	560	632,38	228	228	228	217
92	3365,70	98,67	557,1	622,62	228	228	228	207
161	2205,55	64,66	556,4	700,08	228	228	228	187
169	3403,53	99,78	561,3	629,86	228	228	228	198
187	3410,19	99,98	561,375	621,00	228	228	228	192
218	3198,76	93,78	561,5	578,23	228	228	228	195
251	3397,52	99,60	560,925	516,92	228	228	228	194
323	2171,18	63,65	556,55	526,18	228	228	228	179
339	3400,67	99,70	560,975	436,92	228	228	228	199
368	3387,21	99,30	560,475	383,21	228	228	228	215
403	3406,24	99,86	560,6	306,09	228	228	228	222
434	3394,29	99,51	561,125	259	228	228	228	20
468	3407,78	99,91	560,825	180	228	228	228	218
504	3403,71	99,79	560,575	120	228	228	228	215
551	2881,59	84,48	558,95	50	228	228	228	216
573	2382,84	69,86	557,175	35	228	228	228	208

Os resultados do k_{eff} para cada intervalo de tempo são apresentados na tabela 16. Os valores obtidos mostram resultados consistes, pois todos são próximos de 1.

A figura 47 mostra a distribuição de potência radial quando o núcleo está a 100 % de potência. Este resultado vai ajudar a comparar os resultados obtidos com o modelo *transformer*.

A comparação dos resultados do PARCS com as medições de concentração de boro de BEAVRS é apresentada na figura 48. A diferença mais significativa foi observada para um ponto operacional próximo ao final da primeira interrupção, com um erro relativo percentual de 21,51%. Considerando todos os pontos, obteve-se um erro quadrático médio de 26,9.

Tabela 16 – Resultados do k-eff para cada intervalo de tempo (HFP).

Δt [dias]	MW/T	k-eff
7	0,000	1,005531
11	2,150	1,009294
36	2,767	1,002709
8	4,326	1,003765
4	4,947	1,001411
15	7,960	1,000718
1	8,162	0,99931
6	9,985	1,000349
4	11,123	1,000276
69	8,047	1,000482
8	8,916	1,020774
18	11,564	1,021989
31	15,124	1,025285
33	18,380	1,033684
72	18,614	1,042859
16	19,146	1,057975
29	20,402	1,060041
35	21,591	1,062162
31	22,587	1,063209
34	23,684	1,063118
36	24,487	1,061748
47	23,440	1,059258
22	23,741	1,055652

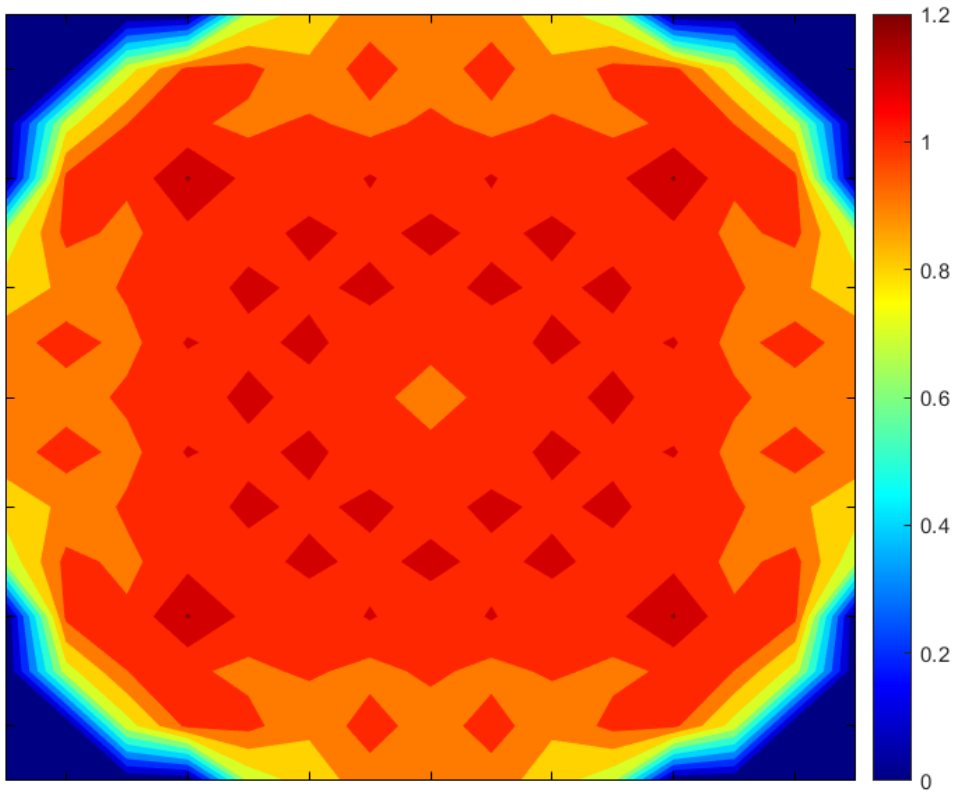


Figura 47 – Distribuição de potência radial média obtida com PARCS para 100% de potência.

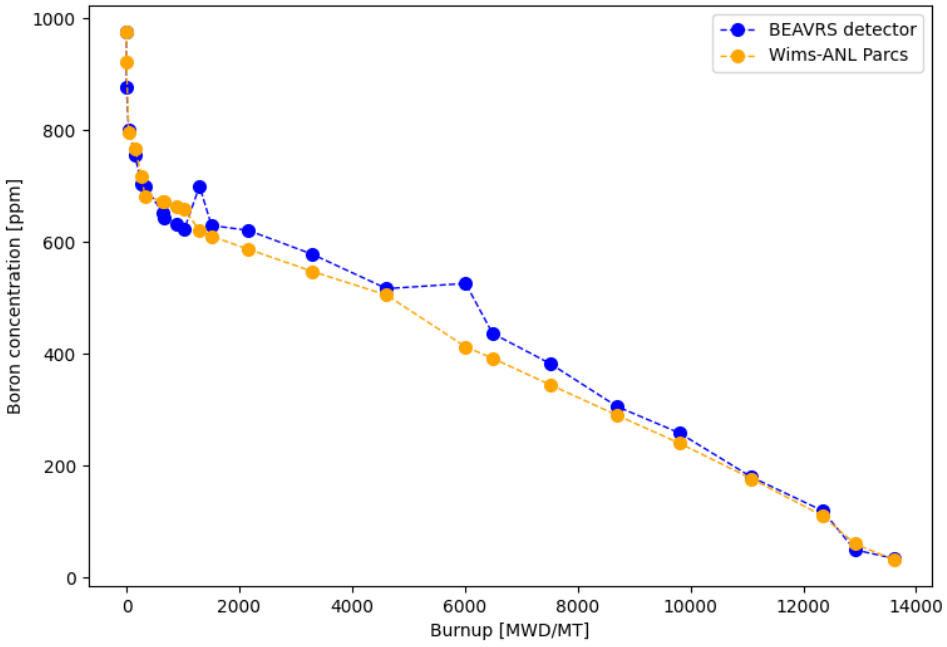


Figura 48 – Comparação da concentração de borro medidos pelos detetores (HORELIK et al., 2013) com os valores obtidos por PARCS/WIMS-ANL.

Capítulo 8

Resultados da otimização do padrão de recarga

Neste capítulo se apresenta os resultados do modelo *transformes*, descrito no capítulo 6. Os dados utilizados para o treinamento do modelo foram gerados conforme descrito na seção 6.1. O processo de treinamento e os resultados são temas abordados nas próximas seções.

8.0.1 Treinamento do modelo *transformer*

O treinamento foi feito com 40000 conjuntos de dados. A matriz *input* I (ver figura 39) contem informação da distribuição de potência radial média de 40000 padrões de recarga. A matriz *output* O (ver figura 40) contêm informações das posições de cada elemento combustível. Essas matrizes foram usadas para treinar o modelo *transformer*. O treinamento foi feito na plataforma *GoogleColab*. Para usar o modelo *transformer*, requer-se de parâmetros como, o tipo de função de perda, número de cabeças, etc. A tabela 17 mostra os parâmetros que foram usadas no modelo. Esses valores são recomendados no trabalho original de Wang (WANG et al., 2018). Na figura 49 mostra-se a função de perda em relação ao tempo de treinamento.

Tabela 17 – Parâmetros usados para o modelo *transformer* (WANG et al., 2018).

Função de perda	Entropia cruzada
Número de cabeças	8
Número de codificadores	4
Número de decodificadores	4

Após treinar o modelo, foram feitos vários testes para comprovar a acurácia dos resultados. Com esse objetivo, foram gerados novos padrões de recarga de forma

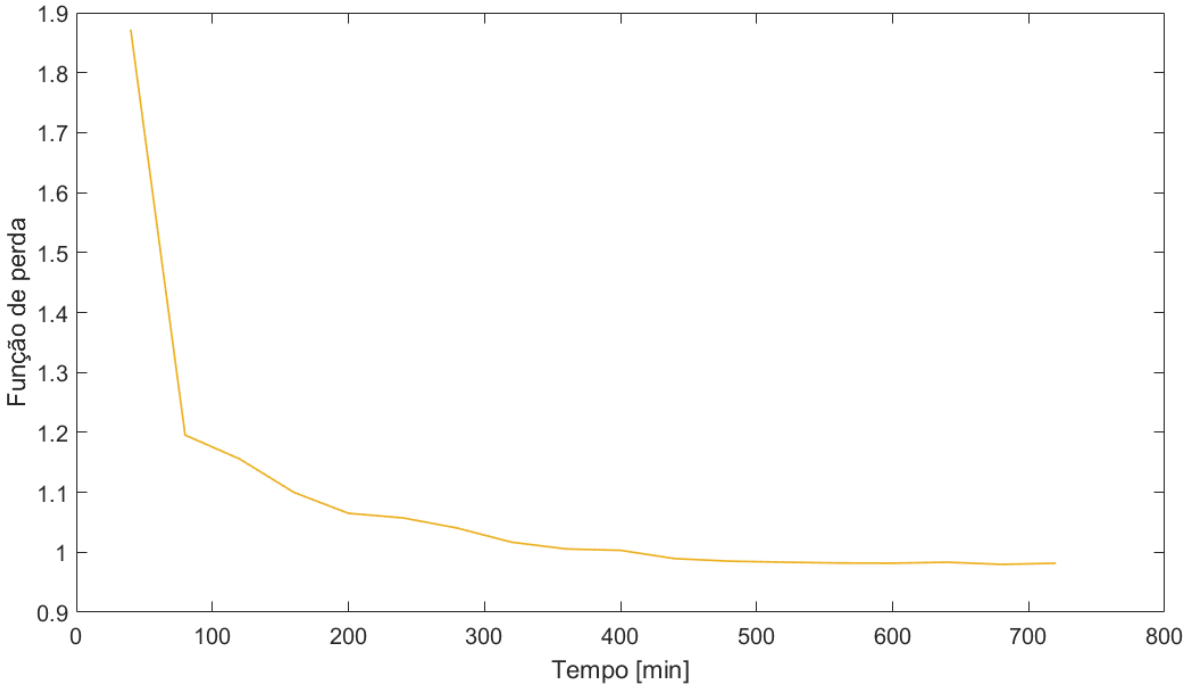


Figura 49 – Função de perda do treinamento do modelo vs tempo.

aleatória com o *script nucleo_aleatorio* (ver Anexo A), e estes são usados como *input* no código PARCS para obter as distribuições de potência radial média. Posteriormente, as distribuições de potência radial média (aleatória) foram usadas como input no modelo *transformer* para obter o padrão de recarga correspondente. Os resultados de três exemplos se mostram nas figuras 50, 51 e 52. As figuras 50a, 51a, 52a ilustram exemplos de padrão de recarga geradas aleatoriamente e suas correspondentes distribuições de potência radial calculadas por PARCS. A figura 50b, 51b, 52b ilustram os padrões de recarga previstas pelo modelo *transformer*.

As figuras 53, 54 e 55 apresentam os erros relativos porcentuais entre as distribuições de potência prevista pelo modelo *transformer* com relação à distribuição de potência usada como *input* no modelo.

Embora as distribuições de potência previstas pelo modelo sejam visualmente semelhantes, o erro relativo mostra que existe um erro máximo de 60% para algumas posições dos ECs (ver figuras 53, 54 e 55). As figuras apresentadas mostram o núcleo completo apesar de que não é necessário, pois escolheu-se a simetria de 1/8. O modelo faz uma boa aproximação no comportamento da distribuição de potência radial, mas não faz uma boa aproximação na precisão dos valores. Apesar desse problema, o modelo pode ser usado para obter o padrão de recarga dos ECs mais próximo para uma distribuição de potência desejada.

O modelo pode ser aplicado em situações em que se busca um determinado comportamento da distribuição de potência, pois ele recebe como entrada uma distribuição de

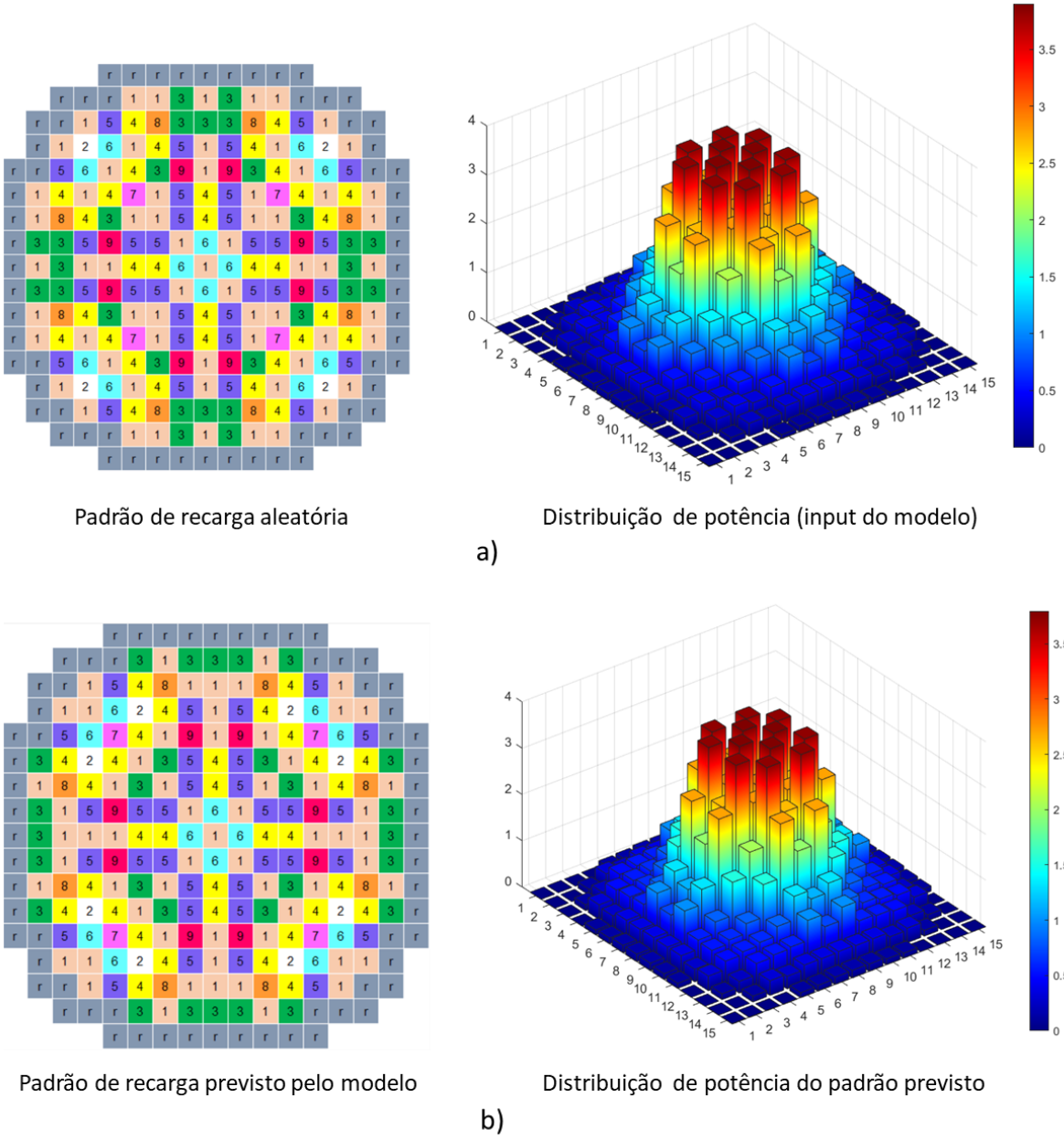


Figura 50 – Exemplo 1 de teste do modelo *transformer*. a) Padrão de recarga gerada de forma aleatória e sua distribuição de potência, b) Padrão de recarga prevista pelo modelo *transformer* e sua distribuição de potência.

potência radial média e ele retorna o padrão de recarga mais próximo para essa distribuição. Na maioria dos casos, uma usina nuclear visa não apenas otimizar os parâmetros físicos desejados, mas também precisa atender restrições como demanda de energia, economia, queima do combustível, etc. Essas restrições devem ser atendidas e não necessariamente do ponto de vista da otimização. Nesse sentido, o modelo poderia ser usado para encontrar a melhor forma de ajustar certos parâmetros para atender às demandas exigidas.

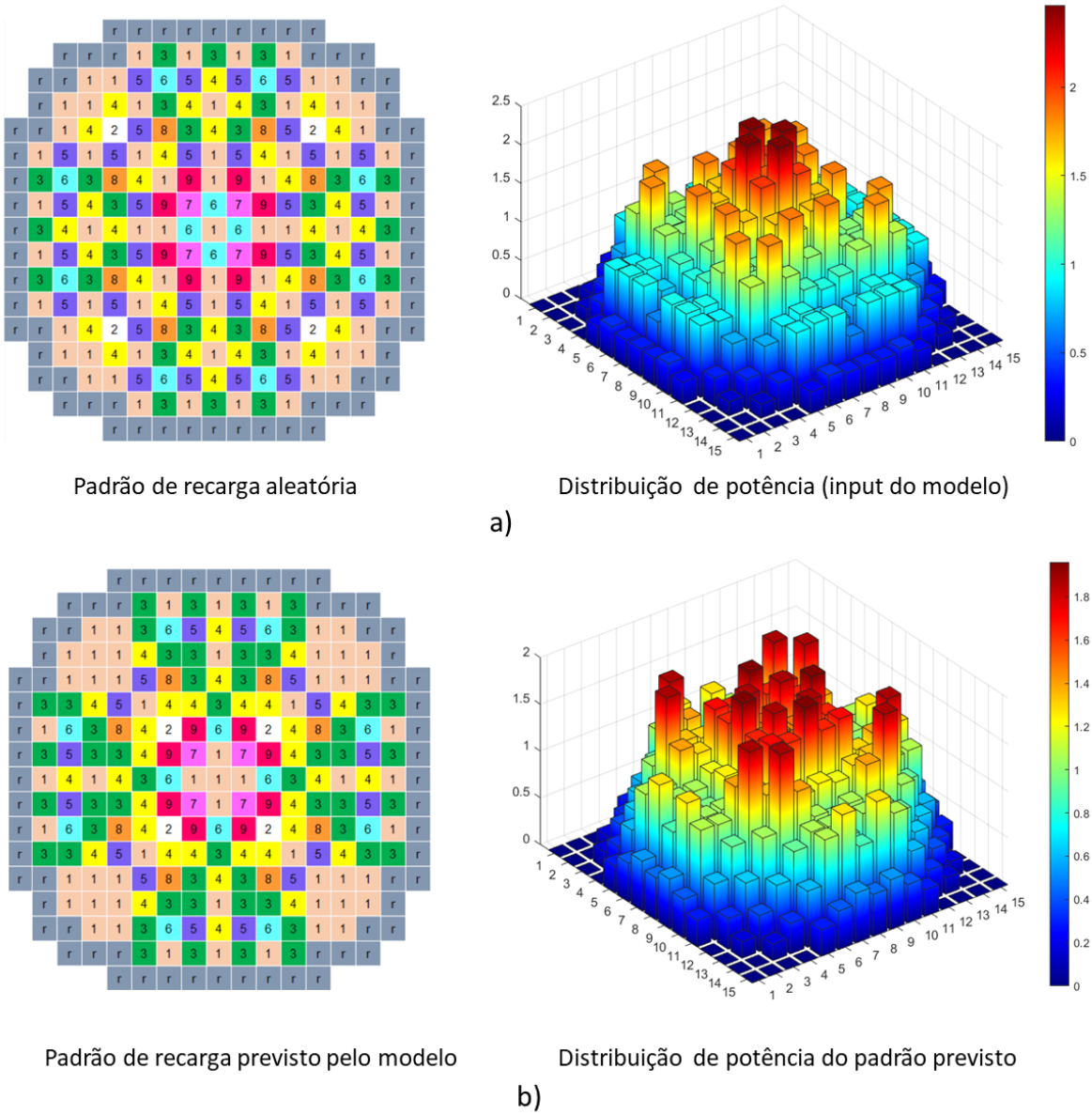
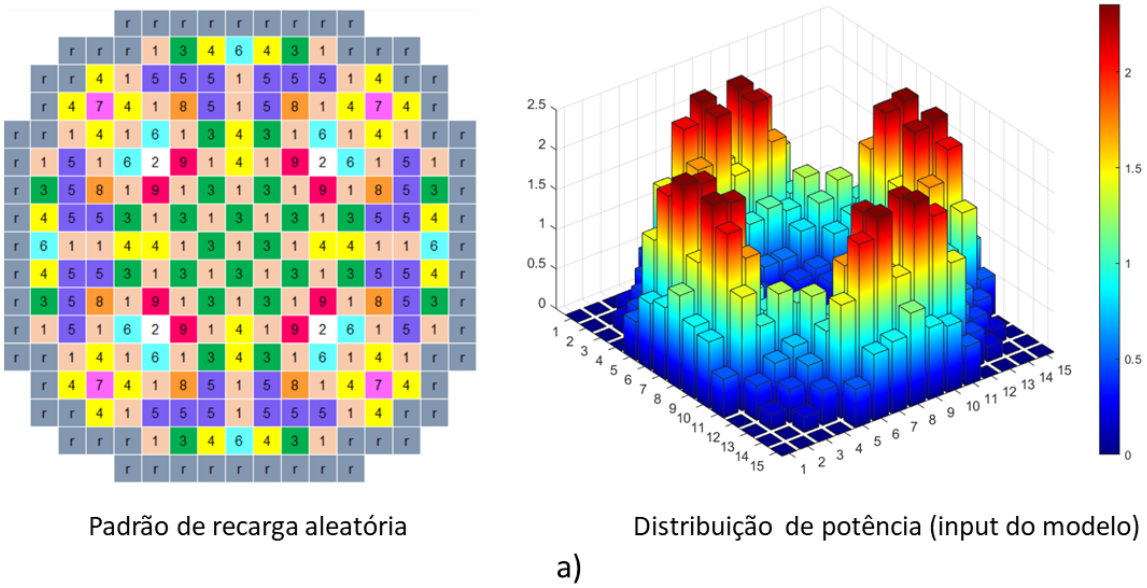


Figura 51 – Exemplo 2 de teste do modelo *transformer*. a) Padrão de recarga gerada de forma aleatória e sua distribuição de potência, b) Padrão de recarga prevista pelo modelo *transformer* e sua distribuição de potência.



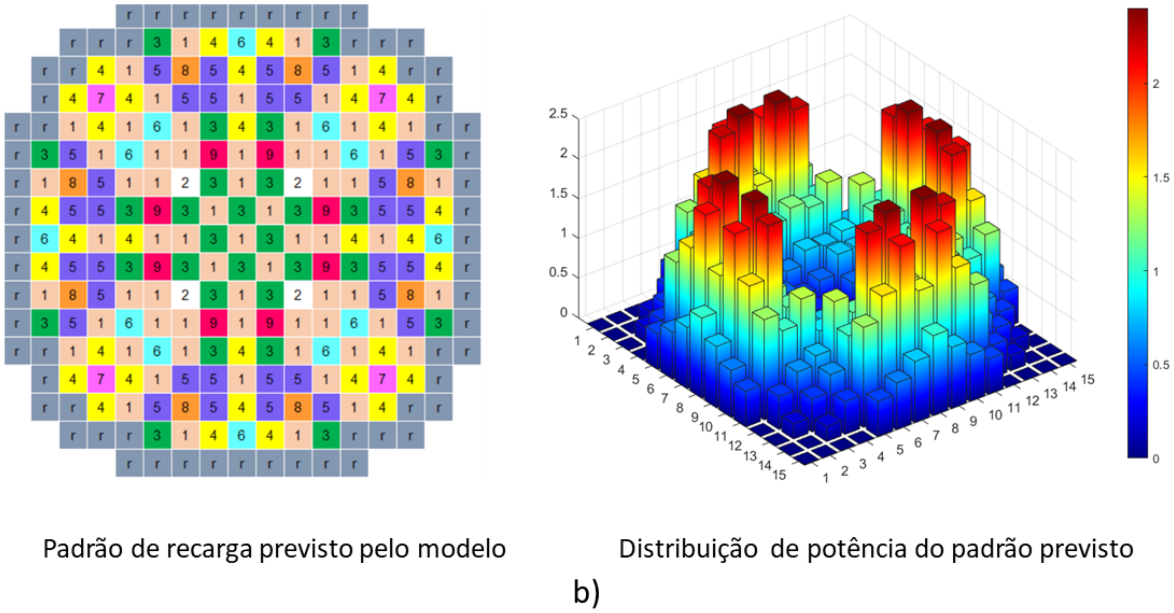


Figura 52 – Exemplo 3 de teste do modelo *transformer*. a) Padrão de recarga gerada de forma aleatória e sua distribuição de potência, b) Padrão de recarga prevista pelo modelo *transformer* e sua distribuição de potência.

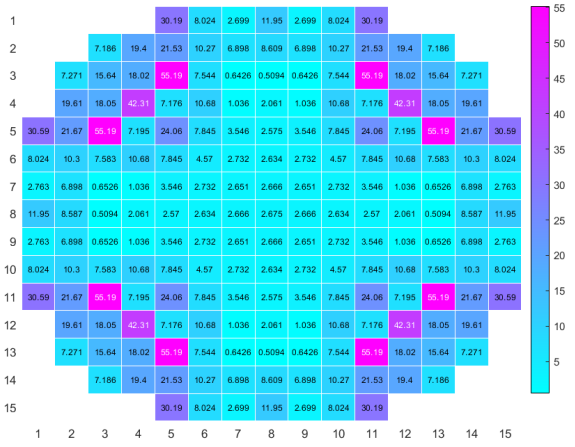


Figura 53 – Erro relativo percentual da distribuição de potência prevista pelo modelo para o exemplo 1.

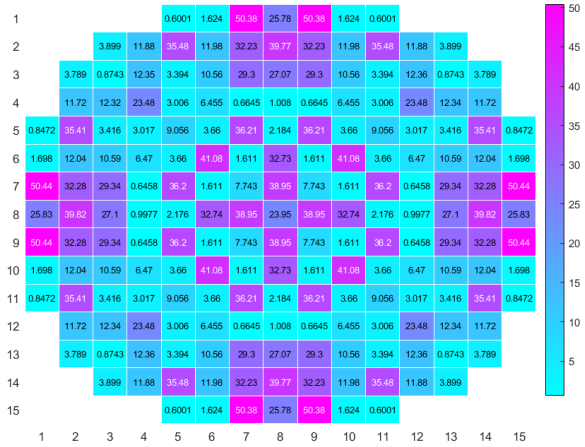


Figura 54 – Erro relativo percentual da distribuição de potência prevista pelo modelo para o exemplo 2.

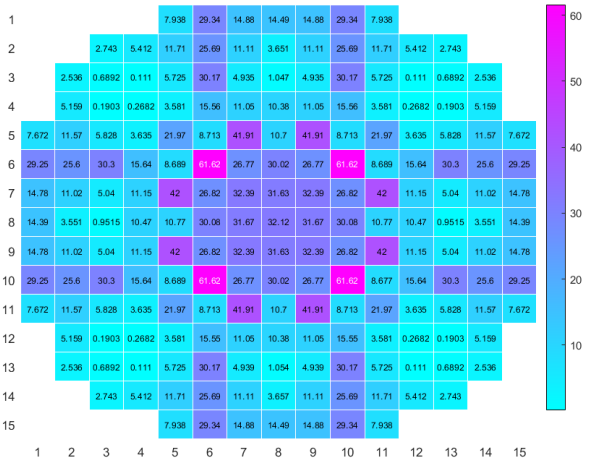


Figura 55 – Erro relativo percentual da distribuição de potência prevista pelo modelo para o exemplo 3.

8.0.2 Proposta de distribuição de potência

O objetivo principal deste trabalho é encontrar a configuração espacial "ideal" dos elementos combustíveis dada uma distribuição de potência radial média. Para atingir esse objetivo, no capítulo 6, vimos que, para que um padrão de recarrega seja ideal, há uma variedade de restrições. Neste trabalho, é proposta uma distribuição de potência que atende às restrições apresentadas na seção 6.2. O perfil de distribuição de potência proposto é ilustrado na figura 56, o valor será normalizada com um valor máximo de 2.1.

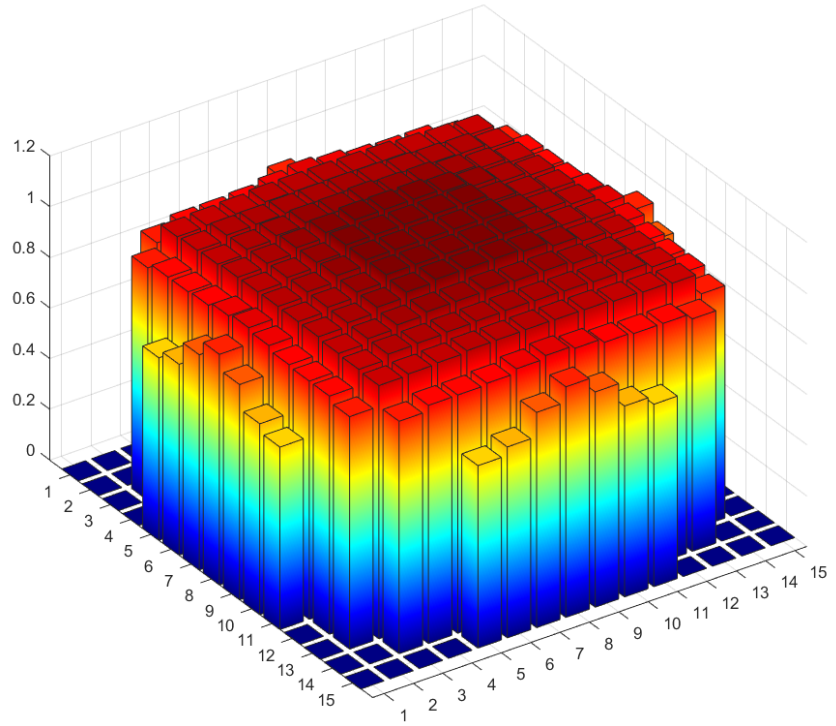


Figura 56 – Proposta da distribuição de potência radial de acordo com as reostricções usadas.

A distribuição de potência proposta é utilizada como entrada no modelo *transformer*, e o modelo retorna as posições dos elementos combustíveis mais próximos dessa distribuição. Este resultado é apresentado na figura 57.

A figura 57a ilustra o padrão de recarga que o modelo *transformer* retorna dada a distribuição de potência radial proposta. A figura 57b apresenta a distribuição de potência radial média calculada por PARCS para o padrão de recarga previsto pelo modelo *transformer*. Percebe-se que a distribuição de potência proposta é diferente da esperada. A distribuição de potência proposta decai muito lentamente a partir da parte central, o que a torna o mais plana possível. Apesar da diferença, podemos perceber que, na distribuição de potência prevista, os valores reduzem à medida que se afasta da parte central, e o valor máximo é próximo de 2,1, que foi imposto ao modelo. A figura 58 mostra uma perspectiva 2D da distribuição de potência, pode-se perceber que o comportamento é parecido ao que

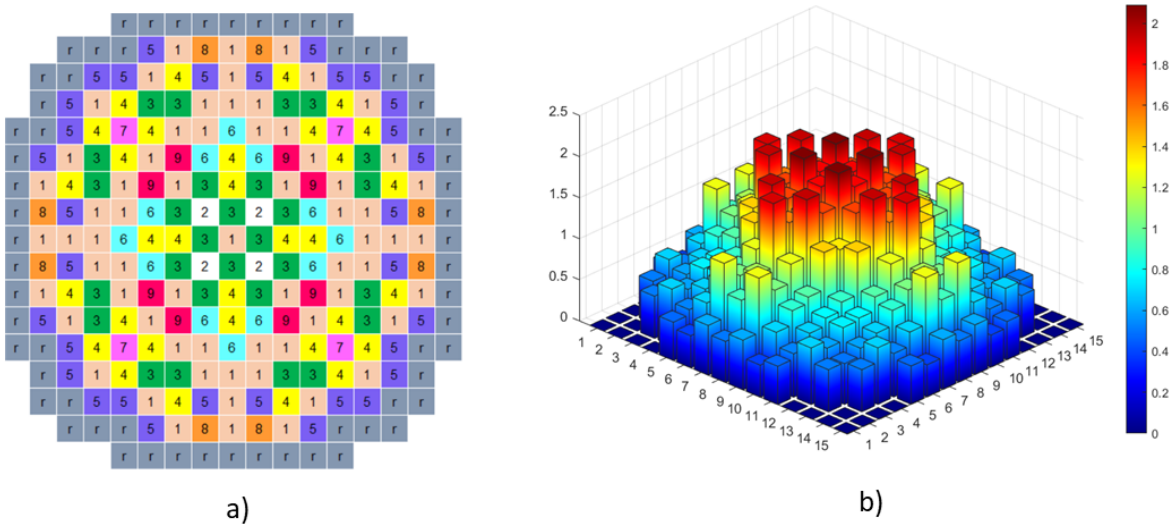


Figura 57 – a) Padrão de recarga prevista pelo modelo. b) Distribuição de potência radial do padrão previsto.

se buscava (ver figura 41), mas exitem muitos pontos donde a potencia difere do proposto. O valor de k_{eff} está dentro do valor esperado ($k_{eff} = 1,019523$), ou seja o valor não é menor que 1 nem muito maior que 1.

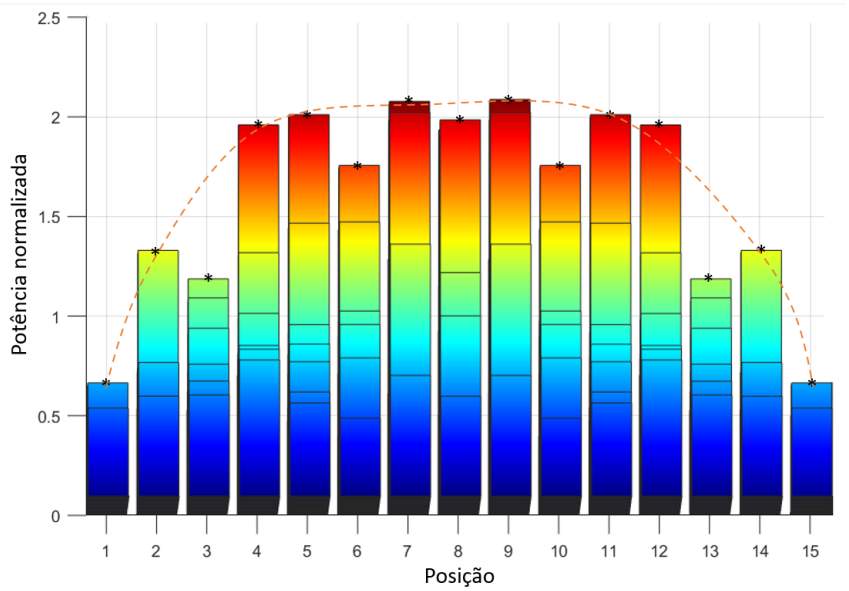


Figura 58 – Perspectiva 2D da distribuição prevista.

A partir dos resultados podemos inferir que talvez não exista uma distribuição de potência como a que foi proposta, e o modelo obteve a distribuição mais próxima. Provavelmente se o modelo for implementado com mais parâmetros, ele poderá encontrar exatamente a distribuição de potência proposta. Esses parâmetros extras podem ser a possibilidade de variar o enriquecimento do combustível, temperatura do refrigerante, etc.

Parte V

Conclusões

Capítulo 9

Conclusões

Neste trabalho foi abordado o problema de patrão de recarga de um reator PWR, com o objetivo de encontrar uma distribuição "ótima" da configuração espacial dos elementos combustíveis. Para atingir este objetivo, foi considerado o *benchmark* BEAVRS, onde os detalhes de operação, geometria, composição, etc. de um reator PWR da Westinghouse são especificados. Taís dados foram utilizados para simular o núcleo do reator em duas etapas. A primeira esteve ligada ao cálculo das seções de choque macroscópicas através do código WIMS-ANL. A segunda etapa consistiu em utilizar as seções de choque no arquivo de entrada do PARCS, com a finalidade de obter a distribuição de potência radial.

Para obter as seções de choque com o código WIMS-ANL, a homogeneização dos elementos combustíveis é comumente realizada. Neste trabalho foi adotada uma metodologia diferente, onde foi utilizado o método MULTICELL. Esta metodologia foi usada por oferecer melhores respostas para elementos combustíveis que continham venenos queimáveis. Os valores do k_{eff} calculados pelo WIMS-ANL, para cada tipo de elemento combustível, foram comparados com trabalhos de referência e os resultados foram bastante satisfatórios em comparação com outros trabalhos, com desvio padrão máximo de 0,00466. Também se calculou o k_{eff} dos ECs com o método de homogeneização, e para esse caso, o valor máximo do desvio padrão foi 0,053. Verificou-se que que o método de homogeneização funciona bem para ECs que não contêm venenos queimáveis.

As seções de choque macroscópicas calculadas pelo WIMS-ANL foram utilizadas no código PARCS para obter a distribuição de potência radial e o k_{eff} do núcleo. Os cálculos foram feitos para o estado *Hot Zero Power* e para a queima do combustível. Para o estado HZP, foi obtida a distribuição de potência radial e axial, cujos valores estão muito próximos aos medidos pelos detectores. Para a queima do combustível, pôde-se verificar que a curva do histórico de queima do boro é consistente com os dados oferecidos no *benchmark*. Esta primeira etapa do trabalho foi realizada com duas finalidades: a primeira foi validar a metodologia utilizada na simulação neutrônica. Ela permitiu verificar que o modelo é consistente com dados reais e possibilitou a obtenção de outras simulações com

dados confiáveis. A segunda finalidade foi de gerar um conjunto de dados de treinamento para o modelo basado na arquitetura *transfomers* usado neste trabalho. Com a simulação neutrônica foram obtidos 40000 conjuntos de dados de padrões de recarga e distribuições de potência radial média, que foram utilizados no treinamento do modelo.

A segunda parte deste trabalho consistiu em abordar o problema do padrão de recarga. A metodologia utilizada foi propor uma distribuição de potência considerada ideal com os requisitos da teoria e, em seguida, utilizá-la como entrada para um modelo de aprendizado de máquina com objetivo de que ele retornasse o padrão de recarga com a distribuição de potência proposta. A ideia principal foi desenvolver um modelo baseado na arquitetura *transformer*, que recebe uma distribuição de potência como entrada e retorna o padrão de recarga como saída. O modelo foi treinado com 40000 conjuntos de dados, o treinamento foi feito na plataforma GoogleColab que levou aproximadamente 13 horas para finalizar.

Três exemplos foram analisados para verificar a eficiência do modelo *transformer*. As distribuições de potência foram calculadas a partir de três padrões de recarga gerados aleatoriamente. Em todos os três casos, as distribuições de potência dos padrões previstos são semelhantes em comportamento às reais. Apesar da semelhança, o erro relativo para alguns dos ECs foram próximos a 60%.

Percebeu-se que a distribuição de potência proposta é diferente da prevista pelo modelo. Esse fato pode ser devido à falta de mais dados de treinamento ou ausência de parâmetros que melhor se ajustem ao modelo, por exemplo poderia-se considerar os valores das seções de choque macroscópicas de cada EC dentro do modelo *transformer*. Apesar da evidente diferença numérica na distribuição de potência prevista em relação à real, a figura 58 mostra um comportamento de decaimento da parte central para as periferias do núcleo, com 2 pontos onde esta tendência não é seguida, e o valor máximo é próximo a 2,1, valor imposto ao modelo. Na perspectiva 2D da distribuição de potência, nota-se que o comportamento cumpre com alguma das exigências impostas ao modelo, além disso o valor de k_{eff} esta dentro do valor esperado (1,019523).

A partir dos resultados podemos inferir que talvez não exista uma distribuição de potência como a que foi proposta, e o modelo obteve a distribuição mais próxima. Provavelmente se o modelo for implementado com mais parâmetros, ele poderá encontrar exatamente a distribuição de potência proposta. Esses parâmetros extras podem ser a possibilidade de variar o enriquecimento do combustível, temperatura do refrigerante, etc.

A versão atual do modelo desenvolvido pode ser aplicado em situações em que se busca um determinado comportamento da distribuição de potência, pois ele recebe como entrada uma distribuição de potência e ele retorna o padrão de recarga mais próximo a essa distribuição de potência.

Sabe-se que a distribuição de potência radial não permanece constante ao longo da queima do combustível, neste sentido o modelo pode ser utilizado considerando os dados de queima do combustível, e assim este trabalho teria uma aplicação prática. O próximo passo para a continuação deste trabalho, pode consistir no uso de mais parâmetros na obtenção dos dados de treinamento, esses dados poderiam ser as seções de choque macroscópicas, informações sobre as posições das barras de controle e os diferentes enriquecimentos das barras de combustível. Quanto mais parâmetros forem usados no modelo do *transformer*, menor será o seu erro.

Referências

AIZENBERG, I. N. *Complex-valued neural networks with multi-valued neurons*. [S.l.]: Springer, 2011. 262 p. ISBN 3642203523. Citado na página 40.

ALIM, F.; IVANOV, K.; LEVINE, S. H. New genetic algorithms (ga) to optimize pwr reactors: Part i: Loading pattern and burnable poison placement optimization techniques for pwrs. *Annals of Nuclear Energy*, Elsevier, v. 35, n. 1, p. 93–112, 2008. Citado na página 18.

BABAZADEH, D.; BOROUSHAKI, M.; LUCAS, C. Optimization of fuel core loading pattern design in a vver nuclear power reactors using particle swarm optimization (pso). *Annals of nuclear energy*, Elsevier, v. 36, n. 7, p. 923–930, 2009. Citado na página 18.

CASTILLO, A. et al. Fuel loading and control rod patterns optimization in a bwr using tabu search. *Annals of Nuclear Energy*, Elsevier, v. 34, n. 3, p. 207–212, 2007. Citado na página 18.

CHAO, Y.-A.; HU, C.-W.; SUO, C.-A. A theory of fuel management via backward diffusion calculation. *Nuclear Science and Engineering*, Taylor & Francis, v. 93, n. 1, p. 78–87, 1986. Citado na página 18.

CHEN, H. et al. Pre-trained image processing transformer. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2021. p. 12299–12310. Citado na página 19.

COCHRAN, R.; TSOULFANIDIS, N. *The Nuclear Fuel Cycle: Analysis and Management*. [S.l.]: American Nuclear Society, 1999. ISBN 9780894484513. Citado 5 vezes nas páginas 8, 24, 25, 26 e 27.

COLLINS, B. et al. Simulation of the BEAVRS benchmark using VERA. *Annals of Nuclear Energy*, Elsevier Ltd, v. 145, sep 2020. Citado na página 28.

DEEN, J. R. et al. *WIMS-ANL USER MANUAL REV. 6*. ARGONNE, ILLINOIS, 2004. 170 p. Citado 8 vezes nas páginas 9, 18, 52, 55, 59, 65, 80 e 85.

DO, B. Q.; NGUYEN, L. P. Application of a genetic algorithm to the fuel reload optimization for a research reactor. *Applied Mathematics and Computation*, Elsevier, v. 187, n. 2, p. 977–988, 2007. Citado na página 18.

DOWNAR, T. J. Accelerated fuel depreciation as an economic incentive for low-leakage fuel management. *Annals of Nuclear Energy*, Pergamon, v. 13, n. 10, p. 545–548, jan 1986. ISSN 03064549. Citado na página 24.

DOWNAR, T. J.; SESONSKE, A. Light Water Reactor Fuel Cycle Optimization: Theory Versus Practice. In: . Springer, Boston, MA, 1988. p. 71–126. Disponível em: <https://link.springer.com/chapter/10.1007/978-1-4613-9925-4_2>. Citado 2 vezes nas páginas 23 e 24.

ELSAWI, M. A.; HRAIZ, A. S. Benchmarking of the WIMS9/PARCS/TRACE code system for neutronic calculations of the Westinghouse AP1000™ reactor. *Nuclear Engineering and Design*, 2015. ISSN 00295493. Citado na página 28.

FARIA, E. F.; PEREIRA, C. Nuclear fuel loading pattern optimisation using a neural network. *Annals of Nuclear Energy*, Pergamon, v. 30, n. 5, p. 603–613, mar 2003. ISSN 03064549. Citado na página 18.

FAUSETT, L. V. *Fundamentals of neural networks : architectures, algorithms, and applications*. [S.l.]: Prentice-Hall, 1994. 461 p. ISBN 9780133341867. Citado na página 42.

GLASSTONE, S. *Nuclear Reactor Engineering: Reactor systems engineering*. 4. ed. New York: Springer Science & Business Media, 1994. Citado 2 vezes nas páginas 17 e 24.

H. Y. Huang, S. H. L. A New Method for Optimizing Core Reloads. *Transactions of American Nuclear Society*, v. 30, p. 339–341, 1978. Citado na página 23.

HAYKIN, S. *Neural networks and learning machines(Thrid Edition)*. 3. ed. [S.l.: s.n.], 2010. 936 p. ISBN 9783642025341. Citado 2 vezes nas páginas 42 e 45.

HINTON, G.; PROCESSING, T. S. P. distributed. Learning and relearning in Boltzmann machines. University of Toronto, p. 282–317, 1986. Disponível em: <http://www.cs.utoronto.ca/$\sim$hinton/absps/pdp7.pdf>. Citado na página 45.

HORELIK, N. et al. Benchmark for evaluation and validation of reactor simulations (BEAVRS). In: *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, M and C 2013*. [S.l.: s.n.], 2013. v. 4, p. 2986–2999. ISBN 9781627486439. Citado 21 vezes nas páginas 8, 9, 10, 12, 18, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 67, 82, 84, 89 e 91.

HU, Y. et al. Improvement of characteristic statistic algorithm and its application on equilibrium cycle reloading optimization. 2006. Citado na página 18.

IAEA, T.-. *In-Core Fuel Management: Reloading Techniques*. Vienna: INTERNATIONAL ATOMIC ENERGY AGENCY, 1992. (TECDOC Series, 816). Disponível em: <https://www.iaea.org/publications/5462/in-core-fuel-management-reloading-techniques>. Citado na página 21.

IAEA, T.-. *Reload Design and Core Management in Operating Nuclear Power Plants*. Vienna: INTERNATIONAL ATOMIC ENERGY AGENCY, 2020. (TECDOC Series, 1898). ISBN 978-92-0-101720-8. Disponível em: <https://www.iaea.org/publications/13585/reload-design-and-core-management-in-operating-nuclear-power-plants>. Citado na página 21.

JIANG, S. et al. Estimation of distribution algorithms for nuclear reactor fuel management optimisation. *Annals of Nuclear Energy*, Elsevier, v. 33, n. 11-12, p. 1039–1057, 2006. Citado na página 18.

JUMPER, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature 2021 596:7873*, Nature Publishing Group, v. 596, n. 7873, p. 583–589, jul 2021. ISSN 1476-4687. Disponível em: <https://www.nature.com/articles/s41586-021-03819-2>. Citado na página 73.

KALIVAS, J. H. *Adaption of simulated annealing to chemical optimization problems*. [S.l.]: Elsevier, 1995. Citado na página 17.

KASHI, S. et al. Bat algorithm for the fuel arrangement optimization of reactor core. *Annals of Nuclear Energy*, Elsevier, v. 64, p. 144–151, 2014. Citado na página 18.

KLERK, E. de et al. Optimization of nuclear reactor reloading patterns. *Annals of Operations Research*, Springer, v. 69, p. 65–84, 1997. Citado 2 vezes nas páginas 17 e 18.

LI, Z. et al. On-line monitoring analysis of BEAVRS benchmark using NECP-ONION. In: *Physics of Reactors 2016, PHYSOR 2016: Unifying Theory and Experiments in the 21st Century*. [S.l.: s.n.], 2016. ISBN 9781510825734. Citado 4 vezes nas páginas 10, 28, 87 e 88.

LIU, S. et al. BEAVRS full core burnup calculation in hot full power condition by RMC code. *Annals of Nuclear Energy*, Elsevier Ltd, v. 101, p. 434–446, mar 2017. Citado na página 28.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, Kluwer Academic Publishers, v. 5, n. 4, p. 115–133, dec 1943. ISSN 00074985. Citado na página 41.

MOSOEUNYANE, S. *Neutronic analysis of selected fuel assemblies of the MIT BEAVRS benchmark using HEADE collision probability code*. Tese (Doutorado) — North-West University (South Africa), 2019. Citado na página 28.

NICOLAU, A. D. S.; SCHIRRU, R.; Monteiro De Lima, A. M. Nuclear reactor reload using Quantum Inspired Algorithm. *Progress in Nuclear Energy*, Pergamon, v. 55, p. 40–48, mar 2012. ISSN 0149-1970. Citado 2 vezes nas páginas 18 e 23.

ORTIZ, J. J.; REQUENA, I. Using a multi-state recurrent neural network to optimize loading patterns in BWRs. *Annals of Nuclear Energy*, Pergamon, v. 31, n. 7, p. 789–803, may 2004. ISSN 03064549. Citado na página 18.

PARK, H. J. et al. BEAVRS benchmark analyses by DeCART stand-alone calculations and comparison with DeCART/MATRA multi-physics coupling calculations. *Nuclear Engineering and Technology*, 2020. ISSN 2234358X. Citado 3 vezes nas páginas 9, 28 e 35.

PARK, T. K. et al. Multiobjective loading pattern optimization by simulated annealing employing discontinuous penalty function and screening technique. *Nuclear Science and Engineering*, Taylor & Francis, v. 162, n. 2, p. 134–147, 2009. Citado na página 18.

POURSALEHI, N.; ZOLFAGHARI, A.; MINUCHEHR, A. Performance comparison of zeroth order nodal expansion methods in 3d rectangular geometry. *Nuclear engineering and design*, Elsevier, v. 252, p. 248–266, 2012. Citado na página 18.

POURSALEHI, N.; ZOLFAGHARI, A.; MINUCHEHR, A. Development of a high order and multi-dimensional nodal code, acnec3d, for reactor core analysis. *Annals of Nuclear Energy*, Elsevier, v. 55, p. 211–224, 2013. Citado na página 18.

POURSALEHI, N.; ZOLFAGHARI, A.; MINUCHEHR, A. Differential harmony search algorithm to optimize pwrs loading pattern. *Nuclear Engineering and Design*, Elsevier, v. 257, p. 161–174, 2013. Citado na página 18.

POURSALEHI, N.; ZOLFAGHARI, A.; MINUCHEHR, A. Pwr loading pattern optimization using harmony search algorithm. *Annals of Nuclear Energy*, Elsevier, v. 53, p. 288–298, 2013. Citado na página 18.

RYU, M. et al. Solution of the BEAVRS benchmark using the nTRACER direct whole core calculation code. *Journal of Nuclear Science and Technology*, 2015. ISSN 00223131. Citado 4 vezes nas páginas 12, 28, 85 e 86.

SANCHEZ, R. *Assembly homogenization techniques for core calculations*. 2009. Citado na página 52.

SECKER, J. R. et al. Optimum discharge burnup and cycle length for PWRs. *Nuclear Technology*, v. 151, n. 2, 2005. ISSN 00295450. Citado na página 17.

SI, S. Integer permutation programming and the loading pattern optimization code superlpos used at snerdi. In: *Proceedings of the PHYSOR 2002 International Conference, Seoul, Available on CD-ROM*. [S.l.: s.n.], 2002. Citado na página 18.

ŠMUC, T.; PEVEC, D.; PETROVIĆ, B. Annealing strategies for loading pattern optimization. *Annals of nuclear energy*, Elsevier, v. 21, n. 6, p. 325–336, 1994. Citado na página 18.

SUZUKI, A.; KIYOSE, R. Maximizing the Average Fuel Burnup Over Entire Core: A Poison Management Optimization Problem for Multizone Light-Water Reactor Cores. *Nuclear Science and Engineering*, Informa UK Limited, v. 44, n. 2, p. 121–134, may 1971. ISSN 0029-5639. Disponível em: <https://www.tandfonline.com/doi/abs/10.13182/NSE71-A19662>. Citado na página 23.

T. Downar, Y. Xu, T. K. *PARCS v2.7 USER MANUALU.S. NRC Core Neutronics Simulator*. W. Lafayette, Indiana, 2006. Citado 2 vezes nas páginas 18 e 79.

T. Downar, Y. Xu, V. S. *PARCS v3.0, U.S. NRC Core Neutronics Simulator, USER MANUAL*. Rockville, Maryland, 2010. Disponível em: <https://www.nrc.gov/docs/ML1016/ML101610098.pdf>. Citado na página 79.

TENSORFLOW. *Modelo transformador para la comprensión del lenguaje | Text*. 2021. Disponível em: <https://www.tensorflow.org/text/tutorials/transformer>. Citado na página 78.

VASWANI, A. et al. Attention Is All You Need. *Advances in Neural Information Processing Systems*, Neural information processing systems foundation, v. 2017-December, p. 5999–6009, jun 2017. Disponível em: <https://arxiv.org/abs/1706.03762v5>. Citado 4 vezes nas páginas 10, 19, 74 e 78.

WANG, Z. et al. Validation of SuperMC with BEAVRS benchmark at hot zero power condition. *Annals of Nuclear Energy*, Elsevier Ltd, v. 111, p. 709–714, jan 2018. ISSN 18732100. Citado 4 vezes nas páginas 12, 28, 88 e 92.

WU, N. et al. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020. Citado na página 19.

YAMAMOTO, A. A quantitative comparison of loading pattern optimization methods for in-core fuel management of pwr. *Journal of nuclear science and technology*, Taylor & Francis, v. 34, n. 4, p. 339–347, 1997. Citado na página 18.

YU, J. et al. Fuel performance analysis of BEAVRS benchmark Cycle 1 depletion with MCS/FRAPCON coupled system. *Annals of Nuclear Energy*, 2020. ISSN 18732100. Citado na página 28.

Anexos

ANEXO A

Scripts em Python.

Script que ajuda a calcular e formatar as secções de choque para serem usadas no código PARCS 2.4

```
from os import scandir


def xs_fluel_assembly(burn_step, file_output_xs):
    f = open(file_output_xs, 'r')
    texto = f.readlines()
    f.close()

    #burn_step = 4
    bus = '                    SCRAMBLE BROAD-GROUP EDIT,  Burn Step  '
    if burn_step >= 10:
        bus = '                    SCRAMBLE BROAD-GROUP EDIT,  Burn Step '

    step = bus + str(burn_step) + '\n'
    posXS = texto.index(step)

    gr1 = texto[posXS + 7]
    gr2 = texto[posXS + 8]
    # ---------------------- XS Absorption ----------------------------------
    absp = [gr1[13:13+12].replace(" ", ""), gr2[13:13+12].replace(" ", "")]
    # ---------------------- XS Capture -------------------------------------
    capt = [gr1[26:26+13].replace(" ", ""), gr2[26:26+13].replace(" ", "")]
    # ---------------------- XS Nu*Fission ----------------------------------
    nFis = [gr1[40:40+13].replace(" ", ""), gr2[40:40+13].replace(" ", "")]
    # ---------------------- XS Fission -------------------------------------
    fis =  [gr1[53:54+12].replace(" ", ""), gr2[53:54+12].replace(" ", "")]
    # ---------------------- XS trasporte -----------------------------------
    gr1t = texto[posXS + 13]
    gr2t = texto[posXS + 14]
    trast = [gr1t[13:13+12].replace(" ", ""), gr2t[13:13+12].replace(" ", "")]
    # ---------------------- XS Scattering ----------------------------------
    gr1s = texto[posXS + 20]
    gr2s = texto[posXS + 21]
    scat = [gr1s[13:13+12].replace(" ", ""), gr2s[13:13+12].replace(" ", "")]
    # ====================== Energy per fission ==============================
    en_f = 3.100000
    xs_f1 = fis[0][0:7]
    expo = int(fis[0][8:len(fis[0])]) -11
    kappaFis1 = str(round(en_f * float(xs_f1), 5)) + 'E' + str(expo)
```

```
    xs_f2 = fis[1][0:7]
    expo2 = int(fis[1][8:len(fis[1])]) -11
    kappaFis2 = str(round(en_f * float(xs_f2), 5)) + 'E' + str(expo2)
    kappaFis = [kappaFis1, kappaFis2]
    # ========================================PARCAS Format ========================================
    dsc = '!␣␣␣␣␣␣␣␣␣␣␣␣␣X-Sec_tr␣␣␣␣␣X-Sec_abs␣␣␣␣nuX-Sec_fiss␣kappaX-Sec_f␣␣␣␣X-Sec_s\n'
    Line1 = '␣base_macro␣' + trast[0] + '␣␣' + absp[0] + '␣␣' + nFis[0] + '␣␣' +
     kappaFis[0] + '␣␣' + scat[1]+'\n'
    Line2 = '␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣' + trast[1] + '␣␣' + absp[1] + '␣␣' + nFis[1] + '␣␣'
    + kappaFis[1] + '\n'

    macroXSf = [dsc, Line1 , Line2]
    return macroXSf

def macroXsRf(wimsXsout_Reflc):
    #archivo = '14-23'
    archivo = [arch.name for arch in scandir(wimsXsout_Reflc) if arch.is_file()]
    XStodos = []
    for i in range(len(archivo)):

        file = wimsXsout_Reflc + archivo[i] #+ '.out'
        f = open(file, 'r')
        texto = f.readlines()
        f.close()
        posXS = []
        for j in range(len(texto)):
            if texto[j] == 'CROSS-SECTIONS:\n':
                posXS.append(j)
        # ---------------- XS Water ----------------------------------------
        posXSw = posXS[0]
        gr1 = texto[posXSw + 3]
        gr2 = texto[posXSw + 4]
        # ---------------------- XS Absorption ------------------------------------
        absp = [gr1[13:13+12].replace("␣", ""), gr2[13:13+12].replace("␣", "")]
        # ---------------------- XS Capture ---------------------------------------
        capt = [gr1[26:26+13].replace("␣", ""), gr2[26:26+13].replace("␣", "")]
        # ---------------------- XS Nu*Fission ------------------------------------
        nFis = [gr1[40:40+13].replace("␣", ""), gr2[40:40+13].replace("␣", "")]
        # ---------------------- XS Fission ---------------------------------------
        fis =  [gr1[53:54+12].replace("␣", ""), gr2[53:54+12].replace("␣", "")]
        # ---------------------- XS trasporte -------------------------------------
        gr1t = texto[posXSw + 9]
        gr2t = texto[posXSw + 10]
        trast = [gr1t[13:13+12].replace("␣", ""), gr2t[13:13+12].replace("␣", "")]
        # ---------------------- XS Scattering ------------------------------------
        gr1s = texto[posXSw + 16]
        gr2s = texto[posXSw + 17]
        scat = [gr1s[13:13+12].replace("␣", ""), gr2s[13:13+12].replace("␣", "")]
        # ========================================PARCAS Format ========================================
        comp_numW = '␣comp_num␣␣␣␣' + archivo[i][:-4] + '\n'

        dsc = '!␣␣␣␣␣␣␣␣␣␣␣␣␣␣X-Sec_tr␣␣␣␣␣X-Sec_abs␣␣␣␣nuX-Sec_fiss␣kappaX-Sec_f␣␣␣␣X-Sec_s\n'
        Line1 = '␣base_macro␣␣' + trast[0] + '␣␣' + absp[0] + '␣␣' + nFis[0] + '␣␣'
        + fis[0] + '␣␣' + scat[1]+'\n'
        Line2 = '␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣' + trast[1] + '␣␣' + absp[1] + '␣␣' + nFis[1] + '␣␣'
        + fis[1] + '\n'
        macroXSw = [comp_numW, dsc, Line1 , Line2]
        XStodos = XStodos + macroXSw
    return XStodos
```

```
def main_XS_parcs():
    wimsXsout_Reflc = '//Inputs␣WIMS-ANL␣reflector//output//'
    r = macroXsRf(wimsXsout_Reflc)
    file_output_xs = '//output'
    fileOut = '//Parcs␣input//'
    archivo_nombres = [arch.name for arch in scandir(file_output_xs) if arch.is_file()]

    estep_end = 30

    for i in range(2,estep_end+1):
        lineas = []
        for j in range(len(archivo_nombres)):
            lineas.append('␣comp_num␣␣␣␣␣' + archivo_nombres[j][:-4] + '\n')
            lineas = lineas + xs_fluel_assembly(i, file_output_xs + '\\'+ archivo_nombres[j])

        lineas_final = lineas + r
        fileXS = fileOut + 'XS_' + str(i)
        #print(fileXS)
        text_file = open(fileXS, "w")
        for k in range(len(lineas_final)):
            text_file.write(lineas_final[k])
        text_file.close()
```

Scripts para a obtenção de padrões de recarga aleatórias.

```
import numpy as np
def import_txt(file):
    texto = open(file, "r")
    lines = texto.readlines()
    texto.close()
    c = -1
    for r in lines:
        c = c + 1
        lines[c] = r.rstrip('\n')

    delimeters = [',', ';','\t', '␣']
    for i0 in delimeters:
        ndelt = lines[0].count(i0)
        if ndelt != 0:
           tipDel = i0
    M = []
    c = -1
    if tipDel != '␣':
        for i in lines:
            c = c + 1
            M.append(i.split(tipDel))
        M = np.asarray(M)
        return M
    else:
        for j in lines:
            lista = j.split(tipDel)
            while ('' in lista) == True:
                lista.remove('')
            M.append(lista)
        M = np.asarray(M)
        return M
```

Arquivo que utiliza a função *nucleo_aleatorio* //Layout radial//1 octavo.txt (ver figura 38):

```
r   r   r   r   r   nul nul
6   5   6   5   r   r   r
1   9   1   8   5   5   r
4   1   4   1   4   7   0
1   3   1   4   2   0   0
3   1   3   1   0   0   0
1   3   1   0   0   0   0
4   1   0   0   0   0   0
1   0   0   0   0   0   0
```

```
from import_txt import import_txt
import random
import copy
import numpy as np

#file_1octavo = '//Layout radial//1 octavo.txt'

def nucleo_aleatorio(file_1octavo):
    nucelo_1oct = import_txt(file_1octavo)

    y_elementos = nucelo_1oct[1:-1, 0]

    d_elementos = []

    d_f, d_c  = len(nucelo_1oct[:, 1]) - 2, 1
    d_f, d_c  = d_f, d_c
    while str(nucelo_1oct[d_f, d_c]).isdigit() == True:
        d_elementos.append(str(nucelo_1oct[d_f, d_c]))
        d_f, d_c = d_f - 1, d_c + 1

    # Parte central do nucleo  1/8
    elementos_medio = []
    for k in range(3, 9):
        d_f1, d_c1  = len(nucelo_1oct[:, 1]) - k, 1
        d_f1, d_c1  = d_f1, d_c1
        while str(nucelo_1oct[d_f1, d_c1]).isdigit() == True:
            elementos_medio.append(str(nucelo_1oct[d_f1, d_c1]))
            d_f1, d_c1 = d_f1 - 1, d_c1 + 1


    #listas de forma aleatoria
    random.shuffle(y_elementos)
    random.shuffle(d_elementos)
    random.shuffle(elementos_medio)


    nucleo_ale_1oct = copy.copy(nucelo_1oct)

    for i in range(len(y_elementos)):
        nucleo_ale_1oct[i+1, 0] = y_elementos[i]


    # Parte central  1/8
    cont = 0
    for r in range(3, 9):
        d_f2, d_c2  = len(nucelo_1oct[:, 1]) - r, 1
        d_f2, d_c2  = d_f2, d_c2
        while str(nucelo_1oct[d_f2, d_c2]).isdigit() == True:
            nucleo_ale_1oct[d_f2, d_c2] = elementos_medio[cont]
            cont = cont + 1
            d_f2, d_c2 = d_f2 - 1, d_c2 + 1
```

```
# Parte diagonal do nucleo de 1/8

d_f3, d_c3  = len(nucelo_1oct[:, 1]) - 2, 1
d_f3, d_c3  = d_f3, d_c3
c = 0
while str(nucelo_1oct[d_f3, d_c3]).isdigit() == True:
    nucleo_ale_1oct[d_f3, d_c3] = d_elementos[c]
    c = c + 1
    d_f3, d_c3 = d_f3 - 1, d_c3 + 1



#matriz_aleatoria_tras = np.rot90(np.rot90(matriz_aleatoria))
matriz_aleatoria = nucleo_ale_1oct
mat = copy.copy(nucleo_ale_1oct)
matriz_aleatoria_q = np.rot90(mat[::-1])
f1, c1 = matriz_aleatoria_q.shape[0], matriz_aleatoria_q.shape[1]
for n in range(f1):
    for m in range(c1):
        if matriz_aleatoria_q[n, m] != '0':
            matriz_aleatoria_q[n, m] = '0'
            break

l_matris = max([f1, c1])

# 1/4 do nucleo
m_1_q = np.empty([l_matris, l_matris], dtype=object)

for i1 in range(matriz_aleatoria.shape[0]):
    for j1 in range(matriz_aleatoria.shape[1]):
        if matriz_aleatoria[i1, j1] != '0':
            m_1_q[i1, j1] = str(matriz_aleatoria[i1, j1])

for i2 in range(matriz_aleatoria_q.shape[0]):
    for j2 in range(matriz_aleatoria_q.shape[1]):
        if matriz_aleatoria_q[i2, j2] != '0':
            m_1_q[i2 + 2, j2] = str(matriz_aleatoria_q[i2, j2])

for i3 in range(m_1_q.shape[0]):
    for j3 in range(m_1_q.shape[1]):
        if (m_1_q[i3, j3] == 'nul') or (m_1_q[i3, j3] == None):
            m_1_q[i3, j3] = 0

m1_4 = copy.copy(m_1_q)
mq = m1_4[::-1]
mq_2 = mq[1:, :]

mat_medio = np.concatenate((m_1_q, mq_2), axis=0)
mat_r = copy.copy(mat_medio)
mat_medio_refl = np.rot90(np.rot90(mat_r[::-1]))
mat_medio_refl = mat_medio_refl[:, 0:-1]

nucleo = np.concatenate((mat_medio_refl, mat_medio), axis=1)
return nucleo
```