INSTITUTO DE ENGENHARIA NUCLEAR

MARCELO CARVALHO DOS SANTOS

MODELO COMPUTACIONAL PARALELO BASEADO EM GPU PARA CÁLCULO EM TEMPO REAL DA DISPERSÃO ATMOSFÉRICA DE RADIONUCLÍDEOS NAS VIZINHANÇAS DE UMA CENTRAL NUCLEAR

Rio de Janeiro

2018

MARCELO CARVALHO DOS SANTOS

MODELO COMPUTACIONAL PARALELO BASEADO EM GPU PARA CÁLCULO EM TEMPO REAL DA DISPERSÃO ATMOSFÉRICA DE RADIONUCLÍDEOS NAS VIZINHANÇAS DE UMA CENTRAL NUCLEAR

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

Orientadores: Prof. Dr. Claudio Marcio do Nascimento Abreu Pereira

Prof. Dr. Roberto Schirru

Rio de Janeiro

2018

SANTOS Carvalho, Marcelo

Modelo computacional paralelo baseado em gpu para cálculo em tempo real da dispersão atmosférica de radionuclídeos nas vizinhanças de uma central nuclear / Marcelo Carvalho dos Santos – Rio de Janeiro: CNEN/IEN, 2018.

xiii, 67f. : il.; 31 cm.

Orientadores: Claudio Marcio do Nascimento Abreu Pereira e Roberto Schirru

Dissertação (Mestrado em Ciência e Tecnologia Nucleares) – Instituto de Engenharia Nuclear, PPGIEN, 2018.

1. Dispersão Atmosférica de Radionuclídeos. 2. GPU. 3. Computação Paralela

MODELO COMPUTACIONAL PARALELO BASEADO EM GPU PARA CÁLCULO EM TEMPO REAL DA DISPERSÃO ATMOSFÉRICA DE RADIONUCLÍDEOS NAS VIZINHANÇAS DE UMA CENTRAL NUCLEAR

Marcelo Carvalho dos Santos

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA NUCLEARES DO INSTITUTO DE ENGENHARIA NUCLEAR COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA E TECNOLOGIA NUCLEARES.

Aprovada por:

_____

Prof. Cláudio Márcio do Nascimento Abreu Pereira, D.Sc.

_____

Dr. Roberto Schirru, D.Sc.

_____

Prof. Celso Marcelo Franklin Lapa, D.Sc.

_____

Dr. André Luís da Silva Pinheiro, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2018

AGRADECIMENTOS

A Deus que me guiou durante toda essa jornada.

A minha mãe, Celia Regina, que sempre esteve presente me apoiando em todos os momentos.

Ao professor Roberro Schirru que com sua sabedoria sempre apoiou a elaboração deste trabalho.

Ao professor e amigo, Cláudio Márcio, que acreditou no meu potencial e me forneceu a oportunidade de embarcar nessa jornada. O professor desde o primeiro momento, esteve sempre disponível e disposto a nos orientar e ajudar durante essa caminhada.

Ao amigo e companheiro de batalhas Filipe Santana, que me ajudou a superar os desafios durante essa caminhada.

Agradecimento a Comissão Nacional de Energia Nuclear (CNEN) pela bolsa de estudos fornecida durante o curso de mestrado

RESUMO

Uma estimativa rápida e precisa da dispersão atmosférica de radionuclídeos (DAR) é de fundamental importância para o apoio a decisão em casos de acidentes com liberação de materiais radioativos em uma central nuclear. Com o objetivo de aperfeiçoar o sistema de dispersão atmosférica de radionuclídeos (SDAR) da Central Nuclear Almirante Álvaro Alberto (CNAAA), foi proposto um refinamento nos cálculos dos modelos físicos envolvidos. No entanto, o refinamento desejado impõe um grande aumento no custo computacional, fazendo com que os computadores atuais necessitem de um tempo proibitivo para processar os cálculos, impossibilitando a execução do sistema em tempo real. Sendo assim, a fim de acelerar a execução deste sistema e permitir o seu uso efetivo na previsão de DAR em tempo real, é proposta uma abordagem utilizando computação paralela baseada em unidades de processamento gráfico (GPU). Essencialmente, o SDAR usado na CNAAA consiste em quatro módulos (programas) principais: Termo Fonte, Campo de Vento, Dispersão de Pluma e Dose, e Projeção. Este trabalho centra-se no desenvolvimento de uma versão paralela baseada em GPU do módulo Dispersão de Pluma e Dose, com foco no cálculo da dispersão. O módulo Dispersão de Pluma usa um modelo tridimensional de bufadas com trajetória lagrangeana e difusão gaussiana para realizar os cálculos do transporte e difusão de radionuclídeos na atmosfera. Devido às restrições do programa original, uma versão sequencial atualizada foi desenvolvida e utilizada como base para a implementação de um novo algoritmo paralelo baseado em GPU. O programa paralelo foi projetado usando a linguagem de programação C e o Compute Unified Device Architecture (CUDA), em conjunto com técnicas de programação paralela. Como resultado, o tempo de execução de uma simulação do modelo do transporte e difusão de radionuclídeos refinado diminuiu de 2498,59 s (executado em uma CPU Intel-Core I5 7500) para 67,91 s (rodando em uma GPU GTX-1070). Aqui, as questões mais importantes da implementação paralela, bem como os resultados comparativos são apresentados e discutidos.

Palavras-Chave: Dispersão Atmosférica de Radionuclídeos, GPU, Computação Paralela.

ABSTRACT

A fast and accurate estimate of the atmospheric dispersion of radionuclides (ADR) is of fundamental importance for support the decisions in cases of accidents involving the release of radioactive materials at a nuclear power station. Aiming to improve the atmospheric dispersion of radionuclides system (ADRS) of the Almirante Álvaro Alberto Nuclear Power Plant (CNAAA), a refinement was proposed in the calculations of the physical models involved. However, the desired refinement imposes a large increase in computational cost, making current computers need a prohibitive time to process the calculations, making it impossible to run the system in real time. Therefore, in order to accelerate the execution of this system and to allow its effective use in predicting real-time ADS, an approach using parallel computation based on GPUs is proposed. Essentially, the ADRS used in the CNAAA consists of four main calculation modules (programs): Source Term, Wind Field, Plume Dispersion and Dose, and Projection. This work focuses on the development of a parallel version based on the GPU of the Plume Dispersion and Dose module, with focus on the dispersion calculation. The Plume Dispersion and Dose module uses a three-dimensional model of lagrangian trajectory and Gaussian diffusion to perform calculations of the transport and diffusion of radionuclides into the atmosphere. Due to the constraints of the original program, an updated sequential version was developed and used as the basis for the implementation of a new GPU-based parallel algorithm. The parallel program was designed using the C programming language and the Compute Unified Device Architecture (CUDA), in conjunction with parallel programming techniques. As a result, the runtime of a refined dispersion model simulation decreased from 2498.59 s (running on an Intel-Core I5 7500 CPU) to 67.91 s (running on a GTX-1070 GPU). Here, the most important issues of parallel implementation as well as comparative results are presented and discussed.

Keywords: Atmospheric Dispersion of Radionuclides, GPU, Parallel Computing.

LISTA DE FIGURAS

Figura. 1 – Estrutura modular do SDAR usado na CNAAA 5
Figura. 2 – Localização da CNAAA e a área de atuação do seu SDAR 6
Figura. 3 – Representação da malha reticulada tridimensional com uma de suas células em destaque 18
Figura. 4 – Esquema de uma pluma gaussiana 24
Figura. 5 – Representação esquemática do modelo de bufadas, que ainda consideram uma dispersão gaussiana, mas também levam em consideração as mudanças temporais e espaciais do vento 25
Figura. 6 – Arquitetura simplificada da CPU e da GPU 30
Figura. 7 – A arquitetura do Streaming Multiprocessor (SM) de uma GPU da arquitetura NVIDIA Pascal. 32
Figura. 8 – Exemplo de uma grid com 6 blocos com cada um contendo 12 threads 34
Figura. 9 – Função sequencial em C somando 2 vetores 34
Figura. 10 – Função paralela em CUDA somando 2 vetores 34
Figura. 11 – Exemplo do padrão de indexação em CUDA 35
Figura. 12 – Etapas do processo de transferência de dados entre memórias. 36
Figura. 13 – Algoritmo demonstrando o loop sequencial 37
Figura. 14 – Algoritmo CUDA tradicional 37
Figura. 15 – Algoritmo CUDA utilizando grid-strid loop 38
Figura. 16 – Estrutura do algoritmo do módulo Dispersão de Pluma e Dose 40
Figura. 17 – Arquivo de Topografia, as células em branco representam os valores 0 que foram retirados para melhor leitura 41
Figura. 18 – Arquivo de Topografia com Cores 42
Figura. 19 – Estrutura do algoritmo da função Tradif 43
Figura. 20 – Parte do código sequencial da função Tradif que mostra a posição de invocação da função CCMR 45
Figura. 21 – Função sequencial CCMR 46
Figura. 22 – Invocação da função kernel CCMR 47
Figura. 23 – Método tradicional para definir as dimensões da grid 48
Figura. 24 – Função kernel CCMR 49
Figura. 25 – Função device CCMIP 50
Figura. 26 – Função device CCMGN 50

Figura. 27 – Dimensão das células para diferentes níveis de refinamento 54
Figura. 28 – Tempo de execução da função CCMR em relação ao nível de refinamento 57
Figura. 29 – Tempo de execução da função Tradif em relação ao nível de refinamento 58
Figura. 30 – Speedups da função CCMR para diferentes níveis de refinamento 58
Figura. 31 – Speedups da função Tradif para diferentes níveis de refinamento 59

LISTA DE TABELAS

Tabela 1 – Níveis de altura do terreno. 17
Tabela 2 – Tempo de processamento relativo das funções do módulo Dispersão de Pluma e Dose 40
Tabela 3 – Tempo de processamento relativo das funções da Tradif. 43
Tabela 4 – Valor do erro absoluto médio e do erro absoluto médio percentual para a taxa de dose acumulada geradas pelo algoritmo original (Fortran) e o sequencial (C) 53
Tabela 5 – Valor do erro absoluto médio e do erro absoluto médio percentual para a taxa de dose acumulada geradas pelo algoritmo original (Fortran) e o paralelo (CUDA) 53
Tabela 6 – Tamanho da resolução espacial para diferentes níveis de refinamento. 54
Tabela 7 – Domínio computacional para diferentes níveis de refinamento. 55
Tabela 8 – Speedup e tempos de execução do algoritmo sequencial (CPU) e paralelo (GPU) da função CCMR para diferentes níveis de refinamento. 56
Tabela 9 – Speedup e tempos de execução do algoritmo sequencial (CPU) e paralelo (GPU) da função Tradif para diferentes níveis de refinamento. 57

LISTA DE ABREVIATURAS E SIGLAS

ADR	- Atmospheric Dispersion of Radionuclides
ADRS	- Atmospheric Dispersion of Radionuclides System
CNAAA	- Central Nuclear Almirante Álvaro Alberto
CPU	- Central Processing Unit (Unidade Central de Processamento)
CUDA	- Compute Unified Device Architecture
DAR	- Dispersão Atmosférica de Radionuclídeos
FPGA	- Field Programmable Gate Array
FORTRAN	- IBM Mathematical FORmula TRANslation System
GPC	- Graphics Processing Cluster (Cluster de Processamento Gráfico)
GPGPU	- General Purpose Computing on Graphic Processing Unit
GPU	- Graphics Processor Unit (Unidade Gráfica de Processamento)
HPC	- High Performance Computing (Computação de Alto Desempenho)
IAEA	- International Atomic Energy Agency
IEA	- International Energy Agency
INES	- International Nuclear and Radiological Event Scale
MPI	- Message Passing Interface
PVM	- Parallel Virtual Machine
SIMD	- Single Instruction Multiple Data
SM	- Streaming Multiprocessors
SDAR	- Sistema de Dispersão Atmosférica de Radionuclídeos
UDM	- Urban Dispersion Model
ULA	- Unidade Lógica e Aritmética
WNA	- World Nuclear Association
ZPL	- Zonas de Planejamento de Emergência

SUMÁRIO

1. INTRODUÇÃO.....1

1.1. APRESENTAÇÃO DO PROBLEMA.....4

1.2. OBJETIVOS.....6

1.3. JUSTIFICATIVA.....8

1.4. METEDOLOGIA.....10

2. FUNDAMENTAÇÃO TEÓRICA.....12

2.1. DISPERSÃO ATMOSFÉRICA DE RADIONUCLíDEOS.....12

2.1.1. O Modelo de Dispersão Atmosférica Urbana.....15

2.2. O SISTEMA DE DISPERSÃO ATMOSFÉRICA DE RADIONUCLÍDEOS DA CENTRAL NUCLEAR ALMIRANTE ÁLVARO ALBERTO.....16

2.2.1. O Módulo Dispersão de Pluma e Dose.....20

2.2.1.1. O Modelo de Transporte e Difusão de Radionuclídeos.....21

2.3. COMPUTAÇÃO PARALELA.....26

2.3.1. Computação de Alto Desempenho.....27

2.3.2. Taxonomia de Flynn.....28

2.3.3. GPGPU - General-Purpose Computing on Graphics Processing.....29

2.3.3.1. Diferença Básica entre a CPU e a GPU.....30

2.3.3.2. Arquitetura da GPU.....31

2.3.4. CUDA - Compute Unified Device Architecture.....32

2.3.5. A Técnica de Paralelização Grid-Stride Loop.....36

3. IMPLEMENTAÇÃO DO ALGORITMO PARALELO DE TRANSPORTE E DIFUSÃO DE RADIONUCLÍDEOS 38

3.1. ESTRUTURA DO ALGORITMO DO MÓDULO DISPERSÃO DE PLUMA E DOSE.....38

3.1.1. Funções de Entrada.....39

3.1.2. Dosem.....41

3.1.3. Tradif.....41

3.1.3.1. Cálculo da Concentração Média de Radionuclídeos (CCMR).....42

3.2. DESENVOLVIMENTO DA VERSÃO SEQUENCIAL DA FUNÇÃO DO CÁLCULO DA CONCENTRAÇÃO MÉDIA DE RADIONUCLÍDEOS (CCMR).....43

3.3. DESENVOLVIMENTO DA VERSÃO PARALELA DA FUNÇÃO DO CÁLCULO DA CONCENTRAÇÃO MÉDIA DE RADIONUCLÍDEOS (CCMR).....45

4. EXPERIMENTOS E RESULTADOS.....50

4.1. VALIDAÇÃO DO ALGORITMO SEQUENCIAL E DO PARALELO.....50

4.2. REFINAMENTO DO DOMÍNIO COMPUTACIONAL.....52

4.3. ANÁLISE DOS RESULTADOS.....54

5. CONCLUSÕES E TRABALHOS FUTUROS.....58

REFERÊNCIAS.....60

1. INTRODUÇÃO

Segundo a (INTERNATIONAL ENERGY AGENCY, 2017), a energia nuclear é a segunda maior fonte de produção de energia elétrica com baixa emissão de carbono, sendo as usinas nucleares responsáveis por 10,6% da produção de energia elétrica no mundo. Além disso, a energia nuclear e as outras fontes de baixa emissão de carbono possuem papel fundamental na visão da *Internacional Energy Agency* (IEA) para o futuro do setor energético.

O Acordo de Paris de 2015, estabeleceu como um dos objetivos, assegurar que o aumento da temperatura média global seja menor que 2°C. Contudo, para que esse objetivo seja alcançado, é necessário que até 2050 o uso de energias emissoras de CO2 diminua em 85%, e essa parcela seja ocupada pelas fontes energéticas de baixa emissão de CO2. Sendo assim, como a indústria nuclear é um dos principais componentes para que este cenário seja alcançado, a *World Nuclear Association* (WNA) desenvolveu o programa Harmony, o qual estabelece como objetivo que a energia nuclear seja responsável por 25% da produção mundial de energia elétrica até 2050. A WNA afirma que um dos fatores principais para o sucesso do Harmony é o estabelecimento de um paradigma de segurança eficiente, de modo que, a indústria nuclear global assim como as usinas nucleares e todas as partes interessadas precisam assegurar o bem-estar do público, reduzindo as emissões de fontes poluentes e assegurando altos padrões de segurança nuclear.

A segurança é um aspecto prioritário no setor nuclear, não podendo ser comprometida por qualquer razão, de modo que, para que uma usina nuclear entre em operação esta deve garantir o cumprimento e manutenção dos diversos critérios de segurança, estipulados por órgãos governamentais e por organismos internacionais. Entretanto, embora o risco de acidentes em centrais nucleares seja pequeno, estas não são imunes a falhas.

Segundo a WNA, 11 acidentes graves envolvendo reatores nucleares ocorreram nos últimos 65 anos (1952-2017). Dentre estes os seguintes se destacaram:

- Three Mile Island – EUA – 1979: Este acidente alcançou nível 5 na *International Nuclear and Radiological Event Scale* (INES), e foi o maior acidente nuclear da história dos EUA até hoje. O acidente aconteceu devido a fusão parcial do núcleo do reator, ocasionado pelo mal funcionamento da sua bomba de água. Consequências graves foram evitadas devido ao trabalho dos técnicos que conseguiram recuperar o

sistema de resfriamento do reator. Entretanto, para realizar o procedimento gases radioativos precisaram ser liberados para o meio externo.

- Chernobyl – Ucrânia – 1986: O maior acidente nuclear da história da humanidade, possuindo nível 7, o máximo na INES. O acidente ocorreu devido a um conjunto de fatores, como o design falho do reator, aliado com erros de operação cometido pelos operadores do reator. Este é o único acidente envolvendo energia nuclear comercial em que houve fatalidades devido a radiação. No acidente, o reator Chernobyl 4 foi destruído, liberando uma grande quantidade de material radioativo para atmosfera, formando uma pluma radioativa que percorreu grande parte da Europa, contaminando e matando milhares de pessoas.

  Fukushima – Japão – 2011: O segundo maior acidente nuclear da história, também ganhando nível 7, o máximo na INES. O acidente aconteceu devido a um terremoto de 8.9 graus na escala Richter, acompanhado por um tsunami de 15 metros que desativou o fornecimento de energia e o resfriamento de três reatores Fukushima Daiichi, acarretando no derretimento do núcleo desses reatores. Uma grande quantidade de material nuclear foi liberado para o meio ambiente, tendo como principal via a atmosfera, incluindo produtos de fissão, como Iodo-131, Césio-134 e Césio-137. Atualmente a área de Fukushima afetada pela radiação ainda está sendo descontaminada. Contudo, a exposição à radiação na época do acidente foi baixa devido ao rápido trabalho de evacuação que foi iniciado desde o primeiro dia do acidente. No total mais de 100 mil pessoas foram evacuadas.

Os acidentes envolvendo reatores nuclear, embora sejam poucos, podem causar graves consequências para o público e o meio ambiente, principalmente quando há vazamento de material radioativo para o meio externo. Em virtude disso, embora o setor nuclear possua uma filosofia de segurança contínua buscando sempre se adaptar, melhorar e reduzir os riscos. A possibilidade de um acidente é sempre levada em consideração. De modo que, toda central nuclear possui um plano para situações de emergência.

A Central Nuclear Almirante Álvaro Alberto (CNAAA), por exemplo, segundo a (ELETRONUCLEAR, 2011) possui um plano de emergência delimitado por Zonas de Planejamento de Emergência (ZPE), estas representam o raio de distância do edifício do reator de Angra I. No plano da CNAAA, são definidas 4 ZPE: de 3 km; 5 km; 10 km; e 15 km. Em caso de acidente, existem ações pré-definidas que devem ser tomadas de acordo com o grau de impacto do acidente. Por exemplo, em caso de emergência geral, onde há

possibilidade real de liberação de material radioativo para o meio externo é definido que deve haver a ação de evacuação da população da ZPE-3 km para a ZPE-5km, e em caso de agravamento da situação, a evacuação deve ser estendida para a ZPE-10 km.

No planejamento de emergência de centrais nucleares, a evacuação é uma ação que é tomada somente em casos graves, onde por exemplo há suspeita de liberação de material radioativo para o meio externo. Uma vez que, o núcleo de um reator nuclear produz radionuclídeos, como por exemplo o Césio-137, Iodo-131 e o Xenônio-133, e dependendo da gravidade do acidente esses materiais podem ser liberados na forma: de aerossol (Césio -137 e Iodo-131) e de gás (Xenônio-133), dando origem a uma pluma radioativa que pode percorrer grandes distâncias, de acordo com as características da liberação e das condições climáticas. Esta pluma radioativa prejudica o meio ambiente, contaminando o ar, água, solo, plantas, assim como a população que está no raio de ação da pluma. De modo que, a população pode ser afetada de várias formas, podendo ser exposta a altas doses de radiação, acarretando em consequências imediatas para a saúde ou a longo prazo, como câncer de tireóide ou até mesmo problemas genéticos que podem afetar gerações posteriores.

Nesses tipos de acidente nuclear onde há formação de pluma radioativa, o trabalho de evacuação é primordial para que a população seja removida da área de atuação da pluma e assim protegida da exposição à radiação. Contudo, para que a evacuação seja realizada de maneira eficiente, a informação sobre o comportamento da pluma radioativa é um fator essencial para auxiliar os responsáveis no processo de tomada de decisão, a fim de guiar a população para longe de possíveis áreas afetadas. Isto posto, as centrais nucleares utilizam modelos de dispersão atmosférica para poder prever as consequências de uma liberação radioativa para atmosfera.

Um modelo de dispersão atmosférica é definido pela (INTERNATIONAL ATOMIC ENERGY AGENCY, 1986) como uma relação matemática entre a quantidade (ou taxa) de efluentes e a distribuição de sua concentração na atmosfera. Além disso, na área nuclear esses modelos se tornaram ferramentas extremamente importantes, (PARK. CHOE e PARK, 2013). Uma vez que, com a sua utilização, é possível prever a dispersão de radionuclídeos na atmosfera de forma ágil e precisa, permitindo uma resposta rápida e eficiente das autoridades em caso de acidente radiológico na central nuclear. Entretanto, em muitos casos, o fator que limita a implementação dessas ferramentas é o esforço computacional demandado para realizar as simulações complexas dos modelos físicos necessários para prever a dispersão atmosférica de radionuclídeos (DAR). Os processadores atuais disponíveis no mercado ainda

não conseguem executar um modelo mais refinado e preciso, em tempo hábil, considerando que o tempo de processamento aumenta drasticamente nessas condições.

1.1. APRESENTAÇÃO DO PROBLEMA

A Central Nuclear Almirante Álvaro Alberto (CNAAA) é uma central nuclear brasileira, construída em 1971, localizada em Angra dos Reis, na costa do estado do Rio de Janeiro. Desde os anos 80, a central nuclear implementa um sistema de dispersão atmosférica de radionuclídeos (SDAR), que usa uma estrutura modular composta por quatro módulos principais: Termo Fonte, Campo de Vento, Dispersão de Pluma e Dose, e Projeção. A Figura 1 mostra a estrutura modular do SDAR usado na CNAAA.

O módulo Termo Fonte é responsável pela previsão de concentrações e taxas de liberação de material nuclear com base no inventário atual e nos status da central nuclear (incluindo, se aplicado, o acidente diagnosticado). O módulo Campo de Vento calcula a distribuição espacial do vento na região de interesse, usando um campo de velocidade tridimensional não divergente. Neste módulo, o domínio computacional é discretizado em uma malha tridimensional e, para cada célula da malha (ou nó), os campos de vento não divergentes são calculados por interpolação e extrapolação de dados, obtidos pelas torres meteorológicas, seguido por um processo de remoção da divergência.

O módulo Dispersão de Pluma e Dose utiliza o campo de vento tridimensional, gerado pelo módulo Campo de Vento, para calcular a dispersão de material radioativo na atmosfera. Além disso, o modelo aplicado no módulo, considera que a dispersão atmosférica de uma pluma, resultante de uma liberação contínua de radioatividade, pode ser simulada pela dispersão de uma sucessão de bufadas radioativas, liberadas em intervalos de tempo, contendo a mesma quantidade total de atividade. Essas bufadas radioativas são consideradas fontes pontuais em uma região de interesse, subdividida em vários volumes (células), e distribuída em uma malha tridimensional (FABRICK. SKLAREW E WILSON, 1987). Esta configuração permite que a precisão do modelo seja afetada, não apenas pela resolução espacial, mas também pelo tratamento refinado da pluma. O módulo projeção faz estimativas simplificadas de dispersões de plumas para 2h à frente.

Atualmente, o SDAR da CNAAA ainda possui várias limitações e simplificações em seu programa, pois sem esses artifícios os computadores utilizados na época em que o sistema foi desenvolvido não seriam capazes de executa-lo. Algumas simplificações importantes foram:

1. Representação em baixa resolução espacial do domínio computacional;
2. Critérios de convergência grosseiros em métodos numéricos;
3. Modelo de difusão grosseiro; e
4. Grande passo de tempo para simulação.

O SDAR da CNAAA atualmente trabalha com um domínio computacional composto por 23.048 células (67 × 43 × 8) que possuem áreas horizontais de (250 × 250) metros e dimensões verticais variáveis. Cobrindo uma área total de (17 × 11) quilômetros na vizinhança da central nuclear, como mostrado na Figura 2, na qual a posição da CNAAA, bem como as quatro estações meteorológicas (A, B, C e D) são exibidas.

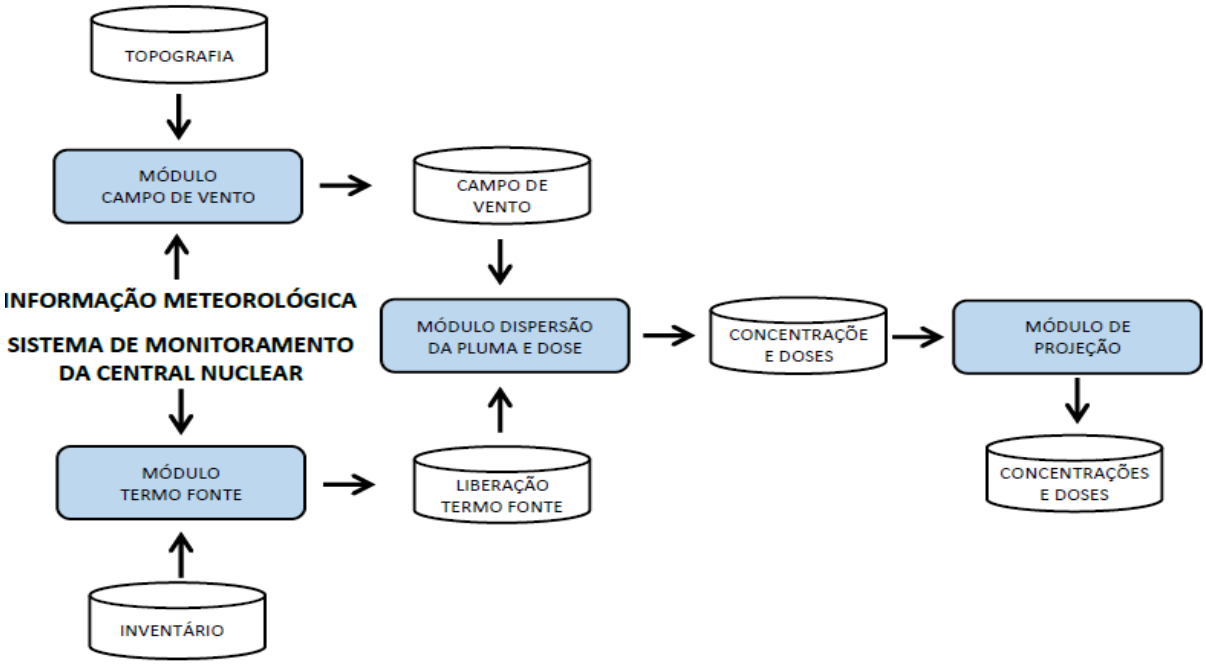

Figura. 1 – Estrutura modular do SDAR usado na CNAAA
Fonte: Pinheiro, 2017, pg 27.

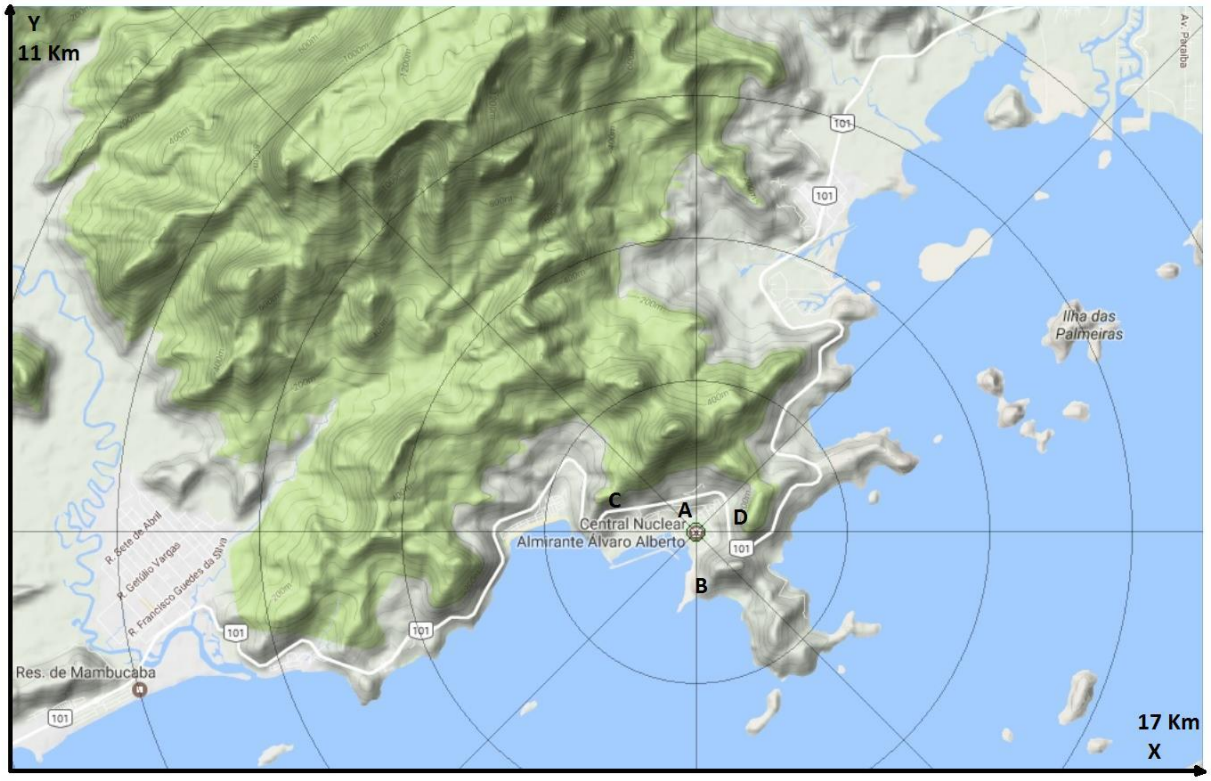

Figura. 2 – Localização da CNAAA e a área de atuação do seu SDAR

1.2. OBJETIVOS

As restrições computacionais da época em que o SDAR da CNAAA foi desenvolvido impuseram limitações que estão integradas ao sistema até hoje. Uma das principais limitações é a resolução espacial, que é equivalente às resoluções usadas nos modelos de dispersão atmosférica de longo alcance. Um exemplo de um sistema que emprega essa metodologia é o sistema ARGOS utilizado na Austrália, (AUSTRALIAN GOVERNMENT, 2008), que usa células com dimensões de 250 metros quando precisa de maiores detalhes em seu mapa. No entanto, de acordo com (PEDERSEN, LEACH e HANSEN, 2007), para regiões como Angra dos Reis, com um alto número de variações e elevações em sua topologia, uma resolução espacial de menos de 100 metros quadrados é indicada, de modo que, as pequenas alterações no terreno possam alterar os padrões de movimentação do campo de vento. Além disso, (PEDERSEN, LEACH e HANSEN, 2007) também indicam que o modelo ideal para centrais nucleares nessas regiões é o Modelo de Dispersão Urbana (UDM), (HALL, SPANTON, *et al.*, s.d.). Uma vez que, o UDM é capaz de estimar a dispersão de bufadas de poluentes, no ar, a curtas distâncias, em áreas urbanas.

Atualmente, o SDAR da CNAAA emprega uma malha tridimensional com uma resolução espacial de 62.250 m^2. Esse é um valor muito alto em comparação com o valor ideal de 100

m^2. Portanto, para que o sistema seja usado de maneira eficiente e auxilie da melhor forma possível o processo de tomada de decisão em uma emergência, é necessário ajustar a resolução espacial do sistema. No entanto, mesmo nos computadores de hoje, ao se tentar executar o SDAR com uma malha tridimensional com nível elevado de refinamento, o custo computacional cresce significativamente, impedindo a tomada de decisões em tempo real.

Desta forma, este trabalho faz parte do projeto que visa melhorar o SDAR da CNAAA, buscando desenvolver uma abordagem computacional que permita o uso de uma resolução espacial de malha mais eficiente e precisa. Portanto, o método escolhido para superar essa limitação foi o uso de técnicas de computação paralela, mais especificamente algoritmos paralelos baseados em GPU, já que como (ALMEIDA, 2009) aponta, esse tipo de algoritmo é uma solução viável para problemas que requerem alta demanda computacional. No entanto, o desenvolvimento de um algoritmo paralelo não é uma tarefa trivial, uma vez que é necessário realizar um estudo detalhado dos modelos envolvidos e seus acoplamentos, a fim de definir quais partes do sistema podem ser paralelizadas. Sendo assim, depois de analisar o programa do SDAR original, foi possível observar que a computação paralela poderia ser usada nos módulos: Campo de Vento; Dispersão de Pluma e Dose; e Projeção, que são os módulos que demandam mais tempo de processamento, e a aceleração desses módulos reduziria o tempo total para execução do sistema.

Com base no contexto descrito, este trabalho centra-se na elaboração de um algoritmo paralelo, baseado em GPU, do módulo Dispersão de Pluma e Dose, mais especificamente da parte de transporte e difusão, de forma a acelerar a execução do módulo, fazendo com que este seja processado em tempo hábil mesmo com o uso de uma malha refinada. Contudo, o algoritmo paralelo desenvolvido deve ser capaz de reproduzir o mesmo resultado de saída (ou o mais próximo possível) do o algoritmo original, já que este resultado é validado pela Eletrobras Eletronuclear para uso na CNAAA.

O projeto de paralelização do SDAR da CNAAA já foi iniciado com o tese de (PINHEIRO, 2017), onde uma versão refinada paralela do módulo Campo de Vento foi desenvolvida, alcançando uma aceleração de 84,27 vezes, em relação à versão sequencial, para um domínio computacional refinado 64 vezes, composto por 1.475.072 (536 x 344 x 8) células que possuem área de 31,5 m^2. Portanto, embora o módulo de Dispersão de Pluma e Dose não exija tanto tempo de processamento quanto o módulo Campo de Vento, este ainda leva um tempo considerável para executar uma simulação utilizando uma malha refinada. De forma que, se aplicada uma malha com resolução espacial de 79,71 m^2, valor que está dentro da faixa indicada por (PEDERSEN, LEACH e HANSEN, 2007) de resolução espacial menor que

100 m^2 para modelos de dispersão atmosféricos urbanos, o tempo de execução de uma simulação com esse nível de refinamento do módulo Dispersão de Pluma e Dose é de cerca de 40 minutos. Este tempo torna inviável a execução do SDAR da CNAAA em tempo real, onde o ideal é que o módulo de Dispersão de Pluma e Dose, que não é o módulo mais exigente em questão de demanda computacional, seja executado em alguns segundos.

Por esse motivo, neste trabalho, como no de (PINHEIRO, 2017), o domínio computacional foi refinado e um algoritmo paralelo, baseado em GPU, foi desenvolvido para o módulo Dispersão de Pluma e Dose utilizando a linguagem *Compute Unified Device Architecture* (CUDA) (BUCK, 2007), aliada com a linguagem C. Os ganhos, os resultados comparativos, bem como as limitações e questões mais relevantes do processo de paralelização e otimização serão apresentados e discutidos neste trabalho.

1.3. JUSTIFICATIVA

Os modelos de dispersão atmosférica de radionuclídeos são ferramentas importantes na engenharia nuclear, pois através deles é possível simular a dispersão atmosférica de materiais radioativos. Contudo, assim como em outros problemas da engenharia nuclear, a solução destes modelos demanda simulações computacionais de alto custo (tempo de processamento).

Neste contexto, atualmente a CNAAA utiliza um SDAR que possui algumas limitações que foram incorporadas ao sistema na época que este foi desenvolvido, de forma a viabilizar a sua execução. Uma das principais restrições limitações do sistema é o uso de uma malha grossa (ou de baixa resolução), para reduzir o custo computacional da simulação, sendo utilizada 23.048 células, com tamanho de 62.250 metros quadrados, para mapear a um espaço de 16.750 x 10.750 x 1.700 metros que representa a região de Angra dos Reis.

Todavia, a configuração de uma malha grossa não é a ideal para mapear uma região urbana como de Angra dos Reis, pois devido ao tamanho das células os detalhes do terreno, como prédios, residências e pequenas alterações de altura do terreno, que deveriam influenciar no vetor de velocidade do vento e por consequência na dispersão da pluma radioativa em caso de acidente radiológico, não são levados em consideração.

Sendo assim, para garantir uma maior precisão no processo de previsão da dispersão de radionuclídeos do SDAR da CNAAA é necessário a implementação uma malha mais refinada, com células menores, que permitam a detecção dos mínimos detalhes do terreno.

Entretanto, um dos motivos pelo qual o sistema ainda não recebeu esta atualização, é que mesmo os processadores atuais disponíveis no mercado não conseguem suprir o poder de

processamento necessário para rodar tal sistema de forma ideal. Uma vez que, refinar a malha significa diminuir o tamanho das células que a compõem, e como consequência aumentar consideravelmente o número de células necessárias para cobrir a região de interesse, e este acréscimo na quantidade de células tem um grande impacto no custo computacional, tornando a execução do sistema extremamente lenta, de modo a inviabilizar a utilização de uma malha refinada.

Contudo, analisando o algoritmo que forma o sistema é possível perceber que o aumento significante do custo computacional ao implementar uma malha fina acontece devido a arquitetura sequencial do programa, esta é estruturada de forma a utilizar apenas um processador para executar sequencialmente os diversos cálculos da simulação, que levam em conta as milhares células da malha. Neste sentido, uma maneira de solucionar este problema e viabilizar o uso de uma malha refinada no sistema, é a utilização de uma arquitetura de computação paralela, esta permitirá que os cálculos que envolvem milhares de células sejam realizados de forma paralela, reduzindo o tempo de execução do sistema.

Atualmente, diversos problemas da engenharia nuclear estão sendo solucionados com o uso da computação paralela**.** Um dos motivos da popularização da arquitetura paralela são as GPUs, estes são periféricos que podem ser adquiridos a um baixo custo e entregam um poder de processamento que antes só poderia ser encontrado em maquinas de alto desempenho.

Portanto, tendo em vista que um SDAR é uma parte essencial no planejamento de emergência de uma central nuclear, auxiliando no processo tomada de decisão, durante e após um acidente envolvendo liberação de material radioativo para meio externo. O trabalho de implementar uma arquitetura paralela para acelerar a execução do sistema e possibilitar que este seja simulado com uma malha refinada é relevante. Pois este processo aumentara a precisão das informações geradas pelo sistema e diminuirá o seu tempo de execução, permitindo que informações mais precisas cheguem mais rapidamente aos tomadores de decisão, em situação de emergência, de forma a facilitar a tomada de decisão e assim garantir a segurança da população e a mitigação das consequências do acidente.

1.4. METODOLOGIA

Primeiramente, materiais sobre modelos de dispersão atmosférica foram estudados, com o estudo centrando-se no modelo de transporte e difusão de radionuclídeos utilizado no SDAR da CNAAA. Em seguida o programa original do módulo Dispersão de Pluma e Dose foi

analisado, com objetivo de detectar os pontos no código que seriam afetados com o aumento da resolução espacial da malha.

Após esta primeira análise, tendo em vista que o código original utiliza a linguagem de programação FORTRAN, se optou por desenvolver um novo código sequencial, usando a linguagem de programação C, visando atualizar o código que foi desenvolvido na década de 80, e assim utiliza-lo como base para o desenvolvimento do algoritmo paralelo.

A elaboração do novo algoritmo sequencial possibilitou um maior entendimento sobre o funcionamento prático do módulo Dispersão de Pluma e Dose, já que este código teve como exigência que o seu resultado de saída fosse igual (ou com mínima diferença) ao do algoritmo original. De modo que, para garantir que este critério fosse atingido, foi realizada uma simulação para uma situação física e utilizando a malha original (67 x 43 x 8), onde foram gerada as informações de entrada necessárias para rodar o programa do módulo de Dispersão de Pluma e Dose, e com isso o algoritmo original (FORTRAN) e o sequencial (C) foram executados durante 20 ciclos, e o resultado destas 20 execuções, gerado pelos programas foi comparado.

Posteriormente, após o desenvolvimento e validação do algoritmo sequencial (em C) do módulo Dispersão de Pluma e Dose, diversos níveis de refinamento de malha foram utilizados para analisar o comportamento o módulo, buscando identificar quais funções eram mais afetadas pelo incremento na resolução espacial. Com este método foi possível detectar a função mais lenta do módulo.

Em seguida, tendo o conhecimento da função que mais impactava no tempo de execução do módulo, técnicas de programação paralela e programação de GPU utilizando a arquitetura CUDA foram estudadas, buscando o melhor método de paralelizar esta função. Sendo assim, após a definição da técnica a ser implementada, a linguagem de programação CUDA/C foi utilizada para desenvolver uma versão paralela do módulo Dispersão de Pluma e Dose. Após o término do desenvolvimento, está versão paralela também foi testada, foram utilizados os mesmos parâmetros da simulação usados para validar a versão sequencial.

Por fim, com as duas versões (paralela e sequencial) do módulo Dispersão de Pluma e Dose desenvolvidas e validadas, foram realizados experimentos para analisar e comparar o tempo de processamento de ambas as versões do módulo, em vários níveis de refinamento de malha. Os experimentos, foram realizados com uma base real de dados, de modo que os dados do campo de vento de entrada, utilizado nos cálculos de dispersão, foram obtidos das 4 estações meteorológicas (A, B, C e D), que podem ser visualizadas na Figura 2.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. DISPERSÃO ATMOSFÉRICA DE RADIONUCLÍDEOS

Na visão de (EISENBUD, 1973 apud COELHO & RIBEIRO, 2008), para avaliar as consequências e grau de risco de uma liberação de radionuclídeos para atmosfera, é necessário prever o destino dos efluentes no espaço e no tempo, ou seja, é preciso saber como os efluentes são diluídos e transportados na atmosfera, e os meios pelos quais estes são depositados na superfície. Nesse sentido, (MELO, 2011), reforça que ao serem lançados na atmosfera, os poluentes são governados e influenciados por um vasto número de processos de alta complexidade. Em virtude disso, para que seja possível avaliar e estudar, de forma precisa a dispersão dos poluentes na atmosfera, os seguintes métodos podem ser utilizados:

a) Experimento em Campo: Esse método possibilita que os reais dados sobre o fenômeno sejam obtidos. Em contrapartida, este também possui um alto custo, e além disso, o trabalho de controlar as variáveis de interesse durante o experimento, é complexo e muitas vezes até impossível.
b) Experimento de Túnel de Vento: Este procedimento, ao contrário do anterior, tem a vantagem de ser executado em ambientes e condições controladas, permitindo que a análise dos resultados seja feita de forma mais descomplicada. Contudo, estes experimentos são realizados a partir de modelos físicos, que envolvem um custo elevado para implementação.
c) Modelagem Matemática: Neste método, a modelagem matemática é utilizada para calcular a dispersão atmosférica de poluentes, utilizando como base a solução das equações essenciais de transporte que não apresentam solução analítica, de modo que, para que a solução seja encontrada são utilizados métodos numéricos. Os modelos matemáticos algébricos, elaborados com base na simplificação das equações de transporte, vêm sendo vastamente empregados, devido ao seu baixo custo e agilidade na obtenção da previsão do impacto ambiental.

Em relação a modelagem matemática, (VELLOSO, 2007), destaca, que vários fatores influenciam na escolha de um modelo matemático, sendo indispensável o conhecimento sobre a geografia e a meteorologia da área em que o modelo atuara. Neste contexto, os modelos podem ser descritos a partir: do seu tipo de fonte (pontual, de volume, área, de linha); pelo

terreno (plano, complexo, plano e complexo); pela escala espacial (local, urbano, regional, global); e pela escala temporal (inferior à uma hora, entre uma e 24 horas, superior a 24 horas). Sendo assim, levando as características citadas em consideração, (SOARES, 2012), separa em duas classes principais os modelos matemáticos usados na dispersão atmosférica de poluentes:

a) Modelos Eurelianos: A modelagem eureliana teve princípio com trabalhos como: o de Reynold em 1973, sobre ozônio em áreas urbanizadas; o de Shir e Shieh em 1974, para SO_2 em áreas urbanas; e também os de Egan em 1976 e o de Carmichael em 1979, sobre a escala local de enxofre (DALY & ZANNETTI, 2007). A principal característica dos modelos eurelianos é o seu sistema referência fixo, em que o processo de dispersão é retratado a partir de um observador fixo em relação a terra (SOARES, 2012). Na modelagem eureliana, a área de interesse é dividida em uma malha que se expande horizontalmente e verticalmente, onde uma equação diferencial (E.D.A) é resolvida explicitamente em pontos desta malha. Nesse contexto, os modelos eurelianos, operam de forma determinística, pois procuram predizer a concentração de poluentes em um volume (LABORATÓRIO DE METEROLOGIA E QUALIDADE DO AR, s.d.).
b) Modelos Lagrangeanos: As pesquisas com modelagem lagrangeana se iniciaram com os trabalhos de Rohde em 1972 e 1974, Eliassen em 1975 e Fisher também em 1975, estes estudos foram voltados à discrição do transporte de longo alcance de partículas de enxofre na atmosfera (DALY & ZANNETTI, 2007). Os modelos lagrangeanos, diferentemente dos modelos eurelianos, trabalham com um referencial móvel, a trajetória de cada partícula do poluente é descrita a partir de um referencial que se move em conjunto com a pluma de poluentes (SOARES, 2012). A modelagem lagrangeana funciona de forma probabilística, onde a predição consiste na probabilidade de certa partícula estar em uma determinada posição especifica (LABORATÓRIO DE METEROLOGIA E QUALIDADE DO AR, s.d.).

Além dos modelos eurelianos e lagrangeanos citados anteriormente, outro tipo de modelo relevante que precisa ser mencionado, é o modelo gaussiano. Este tipo de modelo pode ser considerado uma subclasse dos modelos eurelianos e lagrangeanos, devido a sua capacidade de ser utilizado com os dois métodos de discrição de movimento (eureliano ou lagrangeano).

Contudo, para que isso seja possível, é necessário que a distribuição vertical e lateral da concentração de poluentes apresentem distribuição gaussiana (MELO, 2011).

Os modelos gaussianos são vastamente empregados e estudados, sendo geralmente os mais usados em centrais nucleares. Atualmente as centrais nucleares brasileiras fazem uso deste tipo de modelo.

Dentre os diversos modelos gaussianos na literatura, (DALY & ZANNETTY, 2007) e (PINHEIRO, 2017) destacam dois modelos relevantes que são amplamente utilizados na comunidade cientifica, estes são o AERMOD (AERmic MODel) um modelo de dispersão eureliano gaussiano, que trabalha com uma pluma de estado estacionário, e utiliza um único campo de vento derivado de informações meteorológicas da superfície, do ar superior, e do local, para transportar as espécies emitidas. Além disso, o AERMOD também combina os dados meteorológicos com dados geofísicos, como o tipo e as elevações do terreno, para assim derivar os parâmetros da camada limite da atmosfera. O outro modelo gaussiano em destaque, é o CALPUFF (*California Puff Model*), este é um modelo lagrangeano gaussiano não estacionário, neste, a pluma de poluentes é descrita como uma série de bufadas (*puffs*), contendo o material contaminante que são utilizadas na simulação de como as condições meteorológicas variáveis vão influenciar: no tempo e espaço de transporte, na transformação e remoção dos poluentes na atmosfera.

Um dos motivos da ampla utilização dos modelos gaussianos é devido as suas simplificações e capacidade de manipular estas simplificações. Por exemplo, originalmente, as seguintes simplificações eram implementas nestes modelos: vento com direção constante, taxa de emissão constante, difusão turbulenta desprezível na direção do escoamento, turbulência homogênea, relevo uniforme, inexistência de obstáculos no escoamento. Entretanto, com o tempo algumas destas simplificações começaram a ser modificadas para melhor adequação a certos problemas. Sendo que uma das adaptações mais utilizadas, principalmente para centrais nucleares, é a representação da presença de obstáculos na modelagem matemática, esta modificação acarreta no surgimento de áreas com intensa recirculação e movimentos turbulentos fortemente tridimensionais e anisotrópicos (MELO, 2011).

Neste contexto, (PINHEIRO, 2017), destaca que o modelo de dispersão atmosférica urbana (Urban Dispersion Model – UDM) vem ganhando bastante atenção pelos pesquisadores na área nuclear, por causa da centrais nucleares próximas a regiões urbanas, como a CNAAA.

2.1.1 O Modelo de Dispersão Atmosférica Urbana

A modelagem da dispersão atmosférica de poluentes em ambientes urbanos é um processo complexo, pois diversos fatores precisam ser levados em consideração, alguns dos elementos relevantes na modelagem urbana são os obstáculos, como: os edifícios que alteram os campos de fluxo e desviam o vento, causando correntes ascendentes e descendentes, canalizadas entre edifícios; áreas de ventos calmos adjacentes a ventos fortes; além de árvores, veículos em movimento, saídas de exaustão, entre outros elementos que complicam a modelagem. Entretanto, embora a modelagem da dispersão urbana seja complicada, existe um grande interesse nos modelos de dispersão urbanos, devido à necessidade de haver ferramentas que possam ser usadas para responder, planejar e avaliar as consequências de uma liberação aérea de materiais tóxicos em ambientes urbanos (BROWN, s.d.).

Ademais, Brown também destaca que embora não seja um fenômeno cotidiano, liberações de gases perigosos e aerossóis também ocorrem em ambientes urbanos populosos e estas liberações são potencialmente ameaçadoras para a vida humana. Além disso, a emissão de material tóxico em áreas urbanas pode ter diversas origens, como: acidente durante o transporte de produtos químicos perigosos; liberação premeditada devido a ataques terroristas com um agente químico, biológico ou radiológico; ou acidentes em industrias que lidam com material tóxico, como as usinas nucleares.

Posto isto, em relação a modelagem atmosférica urbana na área nuclear (HALL. SPANTON, *et al.*, s.d.) apresentam o modelo de dispersão urbana (ou em *inglês urban dispersion model* (UDM)), este foi projetado para estimar a dispersão de liberações instantâneas (bufadas ou *puff*s) de contaminantes aéreos, em intervalos curtos de distância em áreas urbanas, onde as obstruções da superfície do terreno, principalmente edifícios e elevações no terreno alteram os padrões da dispersão. O UDM foi elaborado para trabalhar com múltiplas fontes no solo a distâncias entre cerca de 10 m e 10 km, uma vez que, à distancias superiores a 10 km, os dispersantes tendem a preencher toda a camada limite e a natureza da superfície torna-se menos importante para a dispersão.

Além disso, (PEDERSEN, LEACH e HANSEN, 2007), reforçam que um dos principais desafios de um modelo de dispersão urbana é o processo de modelar a deslocação de cada bufada, de modo que os edifícios influenciem na trajetória do transporte, com bufadas sendo dispersas acima e ao redor das edificações, e também sendo canalizadas nas ruas. Sendo assim, devido quantidade de obstáculos que devem ser levados em consideração em um modelo de dispersão urbano, (PEDERSEN, LEACH e HANSEN, 2007), indica que o

tamanho das células, que formam a malha que representa a área urbana de interesse, devem ser menores que, preferencialmente, 100 m^2.

2.2. O SISTEMA DE DISPERSÃO ATMOSFÉRICA DE RADIONUCLÍDEOS DA CENTRAL NUCLEAR ALMIRANTE ÁLVARO ALBERTO

O sistema de dispersão atmosférica de radionuclídeos (SDAR), empregado na Central Nuclear Almirante Álvaro Alberto (CNAAA), tem como principal função, ser uma ferramenta capaz de auxiliar no processo de tomada de decisões, caso haja necessidade de realizar evacuação, devido à acidentes na central nuclear que acarretem em liberação de radionuclídeos para a meio ambiente.

O SDAR é um componente de grande relevância no planejamento de emergências da CNAAA. Uma vez que, em situações de emergência, o tempo de resposta é um fator crucial, de modo que, as decisões tomadas nestas situações devem ser feitas de forma rápida, mas também precisam ser precisas. Neste contexto, o SDAR entra em ação, fornecendo aos tomadores de decisão informações importantes relacionadas a situação do acidente, em termos das consequências para a população dentro da área delimitada pelo sistema.

Contudo, este um sistema que só deve ser ativado essencialmente em situações de emergência ou para simulações de treinamento destas situações. Neste sentido, o sistema só pode ser ativado de duas maneiras: de forma automática e de forma manual.

Na forma automática, o sistema é ativado no modo de operação em emergência sempre que uma das seguintes situações acontecem:

- Sinal de injeção de segurança;
- Alarme de alto nível de radiação em qualquer um dos monitores de radiação;
- Alarme de alta atividade nas salas dos compressores do sistema de tratamento de rejeitos gasosos;
- Baixa pressão nos tanques do sistema de tratamento de rejeitos gasosos; e
- Alta pressão no tanque de alivio do pressurizador (pressão de ruptura do disco).

Enquanto que na forma manual, o sistema pode ser ativado tanto no modo de operação de emergência, no caso de um acidente real ou no modo de operação de simulação, para treinamento e/ou análise.

No modo de operação em emergência, é considerado a existência de um acidente real e só pode ser desativado pelo pessoal da Proteção Radiológica, com ordem expressa da Gerência de Situações de Emergência (FURNAS CENTRAIS ELÉTRICAS S.A., 1987). Já no caso do modo de operação em simulação, o sistema simula um acidente, mas pode ser desativado a qualquer tempo. O objetivo do SDAR da CNAAA, consiste na avaliação da dose radiológica no meio ambiente e o cálculo da dose e da previsão de dose recebida por um indivíduo em um determinado ponto afastado da fonte geradora de radiação, que no caso de um acidente nuclear, seria a própria usina nuclear.

Para que a dose radiológica neste ponto seja devidamente calculada, é necessário determinar como a concentração dos radionuclídeos liberados no acidente variam espacialmente e temporalmente na atmosfera. Por sua vez, para computo da distribuição espacial e da evolução temporal do processo de transporte dos radionuclídeos emitidos, é necessário que seja calculada a quantidade de material radioativo liberado durante o acidente e como foi o transporte e difusão deste material desde a sua origem até o seu destino.

Como o transporte e difusão ocorrem em meio atmosférico é necessário levarmos em consideração as variações com o tempo das condições meteorológicas como a velocidade e a estabilidade do vento. Atualmente o sistema leva em consideração um intervalo de tempo de 15 minutos denominado de ciclo. Em cada ciclo, as condições meteorológicas são mantidas constantes para que sejam efetuados os cálculos necessários para a avaliação da dose radiológica em um determinado ponto.

A cada ciclo do sistema os seguintes cálculos são realizados:

1. Avaliação da taxa de liberação de radionuclídeos para a atmosfera;
2. Determinação do campo de velocidade e estabilidade do vento;
3. Cálculo do transporte e difusão do material radioativo na atmosfera;
4. Avaliação da distribuição espacial da dose; e
5. Cálculo da dose decorrente de projeções do ciclo atual.

A avaliação da taxa de liberação dos radionuclídeos para a atmosfera, depende fundamentalmente do tipo de acidente (real ou simulado), cada acidente em particular possui suas próprias características que interferem na quantidade dos radionuclídeos liberados. A atividade inicial de cada radionuclídeo disponível para liberação depende do acidente ocorrido

e do tipo de radionuclídeo considerado, identificando-se os caminhos pelos quais essa liberação pode ocorrer.

Por conta destas particularidades, o operador da usina ao diagnosticar/simular o acidente ocorrido, deve informar ao sistema através de uma linha de comando qual o acidente em questão para que a atividade total liberada no ciclo possa ser calculada pelo modelo de termo fonte, fazendo um balanço da atividade através de cada caminho possível compatível com o acidente, de acordo com o *status* dos diversos componentes dos caminhos.

A taxa de liberação média do ciclo é calculada, dividindo-se a atividade total liberada no ciclo, pela quantidade de bufadas liberadas no mesmo ciclo. O valor da taxa de liberação média é tomado como a atividade liberada por uma bufada no ciclo. Uma vez calculada a taxa de liberação média por ciclo, o sistema deve então calcular a distribuição espacial do campo de velocidade do vento baseado nos valores médios do ciclo obtidos nas torres meteorológicas, da velocidade do vento (magnitude e direção).

Para tanto, a topologia da região de interesse é representada por uma malha reticulada tridimensional de tamanho fixo 67 x 43 x 8, Figura 3, composta por células retangulares com dimensões $\Delta xn = \Delta yn = 250$ m e Δzk = variável. Além disso, neste tipo de representação de topografia do terreno, a dimensão Δzk da célula é variável devido ao fato que os obstáculos são levados em consideração na representação, de forma que, as células sofrem uma elevação na coordenada Z para corresponder com o nível de elevação do terreno. A Tabela 1 mostra os níveis de elevação do terreno.

Tabela 1 – Níveis de altura do terreno.

Z_8	1700 m	ΔZ_8	450m
Z_7	1250 m	ΔZ_7	450 m
Z_6	800 m	ΔZ_6	350 m
Z_5	450 m	ΔZ_5	200 m
Z_4	250 m	ΔZ_4	100 m
Z_3	150 m	ΔZ_3	50 m
Z_2	100 m	ΔZ_2	50 m
Z_1	50 m	ΔZ_1	50 m
Z_0	0 m	-	-

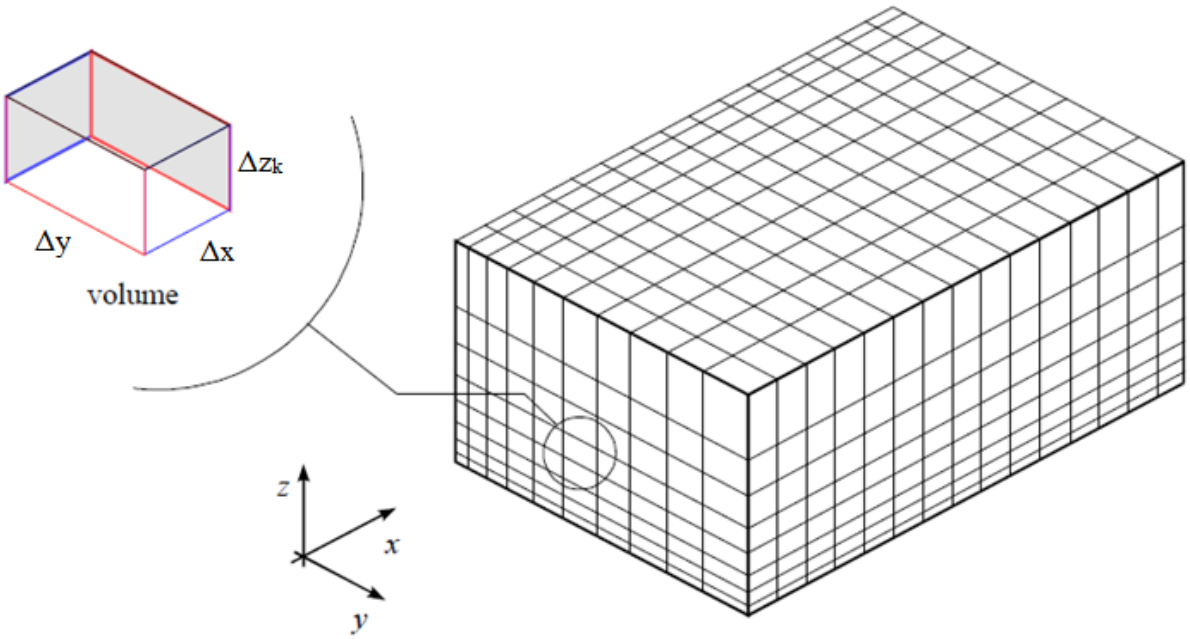

Figura. 3 – Representação da malha reticulada tridimensional com uma de suas células em destaque
Fonte: Pinheiro, 2017, pg 35.

O campo de vento é tornado consistente em massa após um processo de extrapolação e interpolação a partir dos valores fornecidos, através de um processo interativo que elimina as divergências do campo interpolado. Com o campo de vento devidamente conhecido pelo processo acima descrito e distribuído na malha tridimensional, o cálculo do transporte e da difusão dos radionuclídeos na atmosfera é então iniciado e possui um modelo de bufadas tridimensional com trajetória lagrangeana variável e difusão gaussiana.

Cada bufada possui uma taxa de liberação média que foi calculada pelo modulo termo fonte e é liberada e transportada durante um intervalo de advecção, cujo valor é definido a cada ciclo, com a velocidade local do campo de vento no ponto que corresponde ao início do intervalo de advecção. A contribuição de uma bufada para a concentração em cada ponto da malha é determinada pelo modelo gaussiano ao longo da trajetória percorrida pela bufada durante o ciclo e a concentração em cada ponto da malha é o somatório das contribuições médias das bufadas naquele determinado ponto, durante o ciclo que estiver sendo processado.

Os seguintes efeitos físicos são considerados no modelo de transporte e difusão (COPPE/UFRJ - NUCLEAR, LABORATÓRIO DE ANÁLISE E SEGURANÇA, 1987).

- Efeito de esteira (“*building wake*”) causado pelos edifícios da usina;

- Elevação da pluma (empuxo térmico e/ou quantidade de movimento devido à velocidade de saída do material liberado);
- Depleção seca (deposição de particulados e iodos no solo);
- Depleção molhada (deposição de particulados e iodos em caso de chuva durante o transporte atmosférico);
- Decaimento radioativo;
- Reflexão no solo; e
- Variação do coeficiente de dispersão quando a trajetória da bufada atravessa regiões com diferentes classes de estabilidade.

Ao final de cada ciclo, são geradas tabelas que possuem a concentração média de iodos e particulados a nível do solo e para gases nobres a nível do solo, 50m, 125m e 275m. Com base nas tabelas geradas, são calculadas a taxa de dose média no ciclo e as doses acumuladas desde o início do acidente para a tireoide e pulmão, devido à inalação de iodos, particulados e gases nobres e para o corpo inteiro devido à imersão na nuvem radioativa onde somente existe a contribuição dos gases nobres usando-se o modelo de pluma finita.
Assumindo a persistência das condições do ciclo atual, tais como condições meteorológicas e taxa de liberação dos radionuclídeos para a atmosfera, o sistema fornece projeções para uma e duas horas, para as taxas de dose média e doses acumuladas, contados a partir do final do ciclo atual.

2.2.1 O Módulo Dispersão de Pluma e Dose

O módulo Dispersão de Pluma e Dose, utilizado no SDAR da CNAAA é composto por duas partes:

a) Transporte e Difusão: Nesta parte do módulo é realizado o cálculo do transporte e difusão da concentração de radionuclídeos liberados para a atmosfera durante o acidente. De modo a determinar a concentração de radionuclídeos em cada ponto da malha;
b) Calculo de Dose: Já esta parte do módulo, utiliza os valores das concentrações de radionuclídeos, para calcular a distribuição espacial das doses na malha e assim prever a as taxas médias de dose que uma pessoa receberia se esta estivesse em determinado ponto da malha.

Isto posto, esse trabalho foca na parte transporte e difusão de radionuclídeos, que para realizar os cálculos, considera um dado campo de vento livre de divergência tridimensional, gerado pelo módulo Campo de Vento, e a taxa de liberação média de radionuclídeos estimada no módulo Termo Fonte.

No módulo Dispersão de Pluma e Dose, o transporte e difusão de radionuclídeos na atmosfera é determinado através de um modelo de bufadas (*puff model*) tridimensional com trajetória lagrangeana variável e difusão gaussiana. No capítulo 2.2.1.1 será abordado a modelagem matemática envolvida no transporte e difusão de radionuclídeos.

2.2.1.1 O Modelo de Transporte e Difusão de Radionuclídeos

A modelagem matemática e as equações que descrevem o transporte e difusão de poluentes na atmosfera são explicados no trabalho de (GELYBÓ, LAGZI, LEELŐSSY e MÉSZÁROS, 2013). De modo que, os autores primeiramente definem a equação de transporte, explicando que em um volume selecionado do fluido (*V1*), a conservação de massa do componente descrito com concentração *c* pode ser expressa através da equação 1:

$$\frac{d}{dt}\iiint_{v_1} c dV = -\oiint_{\partial v_1} c\bar{v} d\bar{A} + \iiint_{v_1} S_c dV - \oiint_{v_1} (D_c \nabla \mathrm{c}) \tag{1}$$

Onde: $\vec{v}$ é o vetor do vento; S_c é termo fonte; e D_c é o coeficiente de difusão.

Na equação 1, a alteração da massa total do material no volume V1, é representada como a soma do fluxo advectivo através das bordas do volume, dos termos da fonte dentro do volume e do fluxo difusivo. Como o campo de concentração é o único elemento desconhecido na equação 1, esta pode ser transformada em uma forma diferencial, como mostra a equação 2, usando a fórmula de Gauss e generalizando as integrações para qualquer volume V1:

$$\frac{\partial \mathrm{c}}{\partial \mathrm{t}} = -\nabla \cdot (c\vec{v}) + S_c + D_c \nabla \mathrm{c} \tag{2}$$

A equação 2 descreve processos de advecção, fonte e difusão molecular. A deposição seca e úmida, a decadência química ou radioativa faz parte do termo S_c, enquanto a sedimentação gravitacional pode ser adicionada como um componente de advecção extra. Contudo, a difusão turbulenta, não é representada na equação 2. De modo que, é através da teoria de Reynold, que a turbulência é levada em consideração, ao dividir o vento e o campo de concentração em tempo médio e valores de perturbação turbulenta:

$$\vec{v} = \bar{\vec{v}}\,\vec{v}' \tag{3}$$

$$\mathrm{c} = \overline{c}\,\mathrm{c}' \tag{4}$$

A partir das equações 2, 3 e 4, a equação 5 de dispersão que considera o tempo médio e os componentes perturbação é formada:

$$\frac{\partial \overline{c}}{\partial \mathrm{t}} + \frac{\partial \mathrm{c}'}{\partial \mathrm{t}} = -\nabla\left(\bar{\vec{v}}\overline{c}\right) - \nabla\left(\bar{\vec{v}}\mathrm{c}'\right) - \nabla(\vec{v}'\overline{c}) - \nabla(\vec{v}'\mathrm{c}') + S_c + \nabla(D_c\nabla\overline{c}) + \nabla(D_c\nabla\mathrm{c}') \tag{5}$$

(GELYBÓ, LAGZI, LEELŐSSY e MÉSZÁROS, 2013), explicam que na equação 5, todos os componentes que contenham um único termo de perturbação serão eliminados, pois o modelo de Reynolds baseia-se no pressuposto que a média de tempo das perturbações turbulentas é zero. No entanto, na realidade, nem todas as perturbações desaparecerão, já que a média do tempo da covariância não é necessariamente 0. Esta situação é representada na equação 6:

$$\frac{\partial \bar{c}}{\partial \mathrm{t}} = -\nabla\left(\underline{\overline{v}}\,\overline{c}\right) - \overline{\nabla\left(\underline{v}'\mathrm{c}'\right)} + S_c + \nabla(D_c\nabla\overline{c}) \tag{6}$$

Ademais, Gelybó, Lagzi, Leelőssy e Mészáros também explicam que a equação de dispersão para fluxos turbulentos, equação 5, pode ser apresentada da mesma forma que a equação 2, com a adição dos três termos de Covariância de Vórtices Turbulentos, ou *Eddy Covariance* (EC). Neste sentido, escrevendo explicitamente os componentes turbulentos e assumindo uma

difusão molecular isotrópica, a equação 6 pode ser reescrita, como mostra a equação 7, e essa uma forma amplamente aplicada na modelagem da dispersão atmosférica:

$$\frac{\partial \bar{c}}{\partial \mathrm{t}} = -\nabla\left(\bar{\vec{v}}\,\bar{c}\right) + S_c + D_c\nabla^2\bar{c} - \overline{\frac{\partial(\mathrm{u}'\mathrm{c}')}{\partial \mathrm{x}}} - \overline{\frac{\partial(\mathrm{v}'\mathrm{c}')}{\partial \mathrm{y}}} - \overline{\frac{\partial(\mathrm{w}'\mathrm{c}')}{\partial \mathrm{z}}} \tag{7}$$

O lado direito da equação 7 descreve a advecção, os termos fonte, a difusão molecular e os fluxos turbulentos horizontais e verticais. Além disso, na equação 7, quatro novas variáveis são introduzidas na equação.

Para fechar a equação de dispersão turbulenta, (GELYBÓ, LAGZI, LEELŐSSY e MÉSZÁROS, 2013), relatam que existem duas maneiras, sendo a primeira a construção de novas equações de transporte para os fluxos turbulentos e a segunda é usando a parametrização para expressar os fluxos turbulentos com a concentração média do tempo e os valores do vento. A primeira abordagem leva ao modelo Reynold (Reynolds Stress Models (RSM)), enquanto o segundo método é o mais utilizado sendo a teoria do transporte gradiente, ou *K-theory*. A teoria do transporte gradiente, funciona de forma análoga lei de Fick para difusão molecular, baseando-se na ideia que o fluxo turbulento direcional x é proporcional ao primeiro componente do gradiente do campo de concentração, como mostra a equação 8. A partir dessa abordagem, a equação 7 é representada, de acordo com a equação 9, onde os fluxos turbulentos são expressos como um termo de difusão adicional.

$$\overline{u^2c^2} = K_x\frac{\partial \bar{c}}{\partial \mathrm{x}} \tag{8}$$

$$\frac{\partial \bar{c}}{\partial \mathrm{t}} = -\nabla\left(\bar{\vec{v}}\,\bar{c}\right) + S_c + D_c\nabla^2\bar{c} + \nabla(\underline{\underline{K}}\nabla\bar{c}) \tag{9}$$

Na equação 9, K é uma matriz diagonal das difusividades turbulentas Kx, Ky, Kz. Devido aos diferentes processos turbulentos atmosféricos na direção horizontal e vertical, K não pode ser assumido como isotrópico. Além disso, enquanto D_c é uma propriedade das espécies químicas, K é uma propriedade do fluxo, portanto, varia tanto no espaço quanto no tempo. Com base nisso, assumindo um fluido incompressível e turbulência horizontal isotrópica e

negligenciando a difusão molecular, a equação de dispersão pode ser escrita de acordo com equação 10:

$$\frac{\partial \bar{c}}{\partial \mathrm{t}} = -\underline{\overline{v}}\nabla\bar{c} + S_c + \nabla_h\left((K_h\nabla_h\bar{c}) + \frac{\partial}{\partial \mathrm{z}}K_z\frac{\partial \bar{c}}{\partial \mathrm{t}}\right) \qquad (10)$$

Onde: ∇_h é o operador de divergência horizontal; K_h é a difusividade turbulenta horizontal; e K_z é difusividade turbulenta vertical. Estes dois parâmetros precisam ser estimados em cada ponto da malha através de várias parametrizações.

A equação de difusão turbulenta, equação 10, que pode ser resolvida através de diversos métodos: numericamente com discretização espacial de variáveis em uma malha, sendo esta a abordagem eureliana; analiticamente fornecendo uma distribuição Gaussiana, utilizada em modelos de dispersão Gaussiana; ou de forma estocástica, onde em vez de resolver a equação diferencial parcial (EDP), equação 10, o campo de concentração é dado como uma superposição de um grande número de partículas à deriva, sendo esta a abordagem lagrangeana.

Na abordagem gaussiana, assume-se um fluxo homogêneo e em estado estacionário e uma fonte pontual estável, de forma que a equação 10 pode ser analiticamente integrada, resultando na equação 11 que é a conhecida distribuição Gaussiana de plumas:

$$\mathrm{c}(x,y,z) = \frac{Q}{2\sigma\pi_y\sigma_z u}exp\left(\frac{-y^2}{2\sigma_y^2}\right)\left(exp\left(\frac{-(z-h)^2}{2\sigma_z^2}\right) + exp\left(\frac{-(z+h)^2}{2\sigma_z^2}\right)\right) \qquad (11)$$

Onde: c é uma concentração em uma determinada posição; Q é o termo fonte; x é o vento; y é o vento cruzado; z é a direção vertical; e u é a velocidade do vento na altura h da liberação. Enquanto, os desvios σ_y, σ_z descrevem o vento transversal e a mistura vertical do poluente, estes são construídos a partir dos valores K_h, K_{hz} e K_z da equação 10. A Equação 11 descreve um processo de mistura que resulta em uma distribuição de concentração Gaussiana tanto no vento transversal quanto na direção vertical, centrada na linha a favor do vento a partir da fonte, a Figura 4 ilustra esse processo. O último termo da equação expressa a reflexão total do solo, portanto, esta fórmula não conta com deposição seca e úmida. Além disso, adicionando um terceiro componente vertical à equação, a reflexão total de uma camada de inversão

também pode ser calculada. A decantação gravitacional e o decaimento químico ou radioativa são negligenciados.

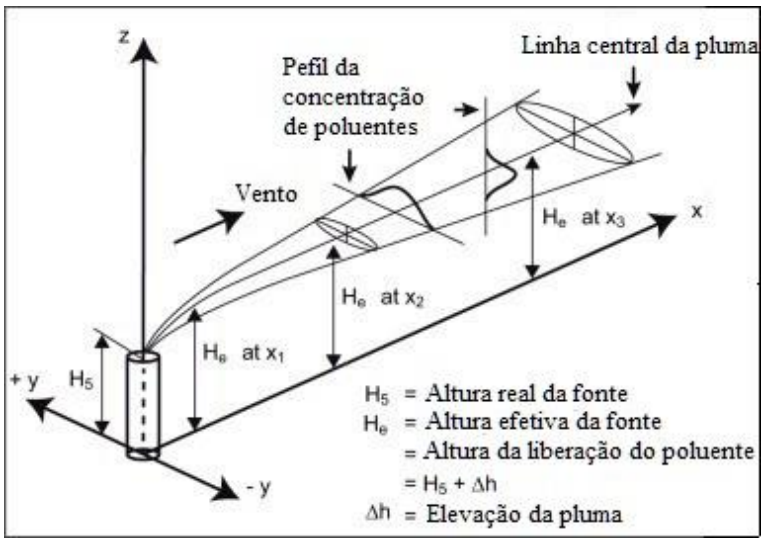

Figura. 4 – Esquema de uma pluma gaussiana
Fonte: Atmospheric Chemistry, Capítulo 10.3: Gaussian dispersion models

Neste contexto, a difusão gaussiana é um fator importante do modelo de transporte e difusão do módulo Dispersão de Pluma e Dose. Uma vez que, este módulo faz uso do modelo de bufadas (*puff model*), este pode ser categorizado como uma intersecção entre o modelo de difusão gaussiano e o lagrangeano. Pois, este mantem a teoria que que o padrão de concentração pode ser satisfatoriamente descrito com uma distribuição Gaussiana. No entanto, a linha central de uma pluma não é uma direção direta do vento, mas uma trajetória lagrangeana, como mostra a Figura 5. Dessa maneira, as mudanças temporais e espaciais do vento são levadas em consideração durante o processo. Ademais, no modelo de bufadas, a concentração de radionuclídeos liberada é fracionada em pequenas unidades, chamadas bufadas (ou *puffs* em inglês), e em seguida as trajetórias de todas as bufadas são calculadas de maneira lagrangeana (de acordo com a equação 11), contudo, o padrão de concentração gaussiano é mantido dentro de cada bufada. De modo que, o campo de concentração final é dado como uma superposição de todas as distribuições de concentração de bufadas, como mostra a equação 12 (GELYBÓ, LAGZI, LEELOSSY e MÉSZÁROS, 2013).

$$\mathrm{c}(x,y,z) = \frac{Q\Delta t}{(2\pi)^{\frac{3}{2}}} \sum_{k=1}^{N} \frac{1}{\sigma_{xk}\sigma_{yk}\sigma_{zk}} exp\left(-\frac{(x_k - x)^2}{2\sigma_{xk^2}} - \frac{(y_k - y)^2}{2\sigma_{yk^2}} - \frac{(z_k - z)^2}{2\sigma_{zk^2}}\right) \qquad (12)$$

Onde: $Q\Delta t$ é o termo fonte, N é o número de bufadas; (xk, yk, zk) representam a posição da k^{n} bufada; e σ_{xk} é o desvio direcional x da distribuição gaussiana dentro da k-ésima bufada. A equação 12 é semelhante à equação 11 com a diferença que no modelo de bufadas é necessário realizar uma soma de muitos componentes. Ademais, no modelo de bufadas o efeito da turbulência é calculado de duas maneiras diferentes: com uma abordagem estocástica de passeio aleatório (*random-walk*) nas trajetórias das bufadas; e através do desvio de uma distribuição normal dentro de cada bufada.

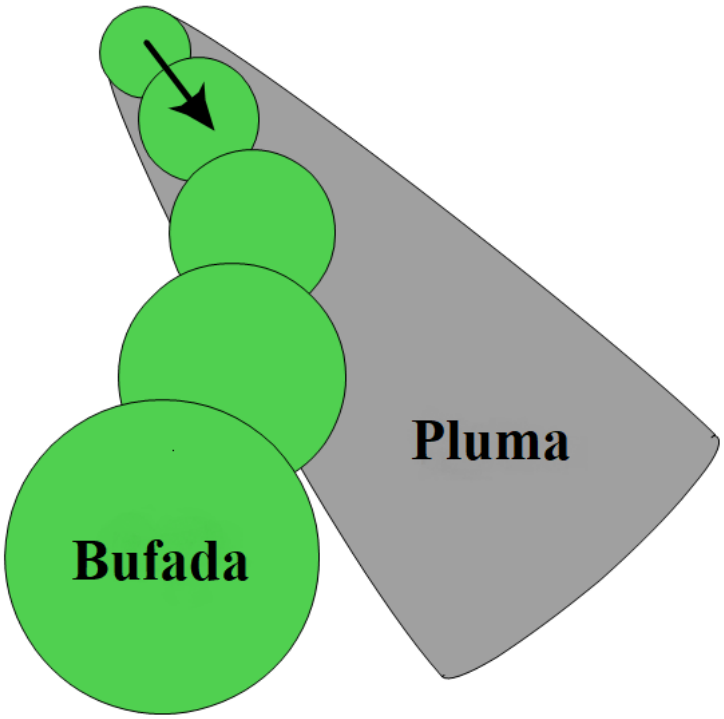

Figura. 5 – Representação esquemática do modelo de bufadas, que ainda consideram uma dispersão gaussiana, mas também levam em consideração as mudanças temporais e espaciais do vento
Fonte: Atmospheric Chemistry, Capítulo 10.4: Lagrangian models

2.3. COMPUTAÇÃO PARALELA

A computação paralela demonstrou ser uma área de extrema relevância no campo da ciência da computação, desempenhando um papel fundamental em pesquisas e resolução de problemas que exigem alto desempenho computacional. Sabe-se que os algoritmos concebidos para execução sequencial (em um único processador) podem não ser eficientes para resolver determinados tipos de problemas complexos e demorados, em um tempo razoável (NAVARRO & KAHLER & MATEU, 2014). Além disso, se o problema aumentar, este pode atingir um nível que nem mesmo uma CPU com arquitetura de múltiplos núcleos é

capaz de alcançar uma solução eficiente. Portanto, às vezes, apenas com o uso de processadores massivamente paralelos, é possível alcançar a eficiência desejada.

Embora no passado, o poder de processamento massivamente paralelo para computação de alto desempenho, ou High Performance Computer (HPC), tenha sido limitado aos supercomputadores, na última década, a evolução da indústria dos jogos e da realidade virtual impulsionaram o desenvolvimento de hardwares com um alto poder de processamento gráfico capazes de executar as suas aplicações. Em virtude disso, as unidades de processamento gráfico, ou *graphics processing units* (GPUs), precisaram se tornar mais sofisticadas para suprir essa demanda, de modo que, diversas melhorias estruturais foram sendo implementadas nas GPUs ao longo do tempo, sendo as principais: o aumento no número e na capacidade de processamento dos núcleos da GPU, e o uso de VRAMs (Video RAMs) com maior capacidade de armazenamento e capazes de alcançar altas taxas de transferência de dados. Hoje as GPUs são consideradas supercomputadores de pequena escala, possuindo um enorme poder de processamento paralelo. Atualmente, as GPUs são também são utilizadas como plataformas de uso geral de baixo custo para aplicações de computação paralela. Nos últimos anos, pesquisadores e profissionais do campo da engenharia nuclear estão investigando o uso da GPU para resolver problemas complexos que demandam um longo tempo de processamento, obtendo ótimos resultados.

Um exemplo, é o trabalho desenvolvido por (HEIMLICH & MOL & PEREIRA, 2011), que alcançou uma aceleração de 125 vezes em relação à versão sequencial, empregando uma GPU para implementar uma solução baseada em diferenças finitas da equação de calor, usando uma abordagem Red-Black Gauss Seidel. Posteriormente, (PEREIRA., *et al*, 2013) mostrou que é possível atingir uma velocidade de 2000 vezes empregando um sistema Multi-GPU, usando 8 GPUs GTX 480, para executar um problema de simulação de transporte de nêutrons. Mais recentemente, (HEIMLICH & SILVA & MARTINEZ, 2016) investigou a implementação de um algoritmo paralelo baseado em GPU do processo de queima do reator PWR, neste experimento pode-se observar uma aceleração de cerca de 100 vezes. Além disso, foi desenvolvida uma versão baseada em GPU do módulo Campo de Vento do SDAR da CNAAA (PINHEIRO), em que uma aceleração de cerca de 84,27 vezes foi alcançada usando uma GTX-680.

2.3.1 Computação de Alto Desempenho

A computação de alto desempenho, ou em inglês High Performance Computer (HPC), refere-se à utilização de um sistema computacional dotado de alto poder processamento, para trabalhar com tarefas que possuem grande volume de dados e necessitam de intensivo poder computacional para serem executadas.

Normalmente, a HPC é aplicada na ciência e na engenharia para solucionar problemas que usando sistemas computacionais comuns se tornariam inviáveis, pois, ou demandariam um longo tempo para serem processadas (devido ao número de operações), ou seriam impossíveis de serem resolvidos devido as limitações de recursos para suportar o grande volume de dados. Neste Contexto, a HPC é a abordagem que para superar essas limitações faz uso de: dispositivos especializados; hardwares de ponta; ou acumula poder computacional através de várias unidades de processamento implementados em conjunto. De modo que, estas técnicas de distribuição de dados e operações em várias unidades baseia-se no conceito de paralelização, (MULTIPHYSICS CYCLOPEDIA, s.d.).

A HPC tem uma forte relação com a computação paralela, onde problemas maiores são divididos em problemas menores que são resolvidos de forma paralela. De modo que, os tipos de arquiteturas mais amplamente utilizadas na HPC são: a de multiprocessadores, onde múltiplas unidades de processamento, geralmente compartilhando a mesma memória, são usadas para executar tarefas de maneira concorrente; e a de multicomputadores, onde uma rede de computadores é formada, com cada máquina possuindo a sua própria memória e unidades de controle e processamento, de maneira de distribuir a demanda de processamento entre os computadores da rede, permitindo que o sistema como um todo seja capaz de lidar rapidamente com grandes quantidades de dados e realizar cálculos complexos.

2.3.2 Taxonomia de Flynn

Atualmente, a taxonomia de Flynn é método mais aceito para categorizar arquiteturas computacionais, esta, com base nas suas entradas e saídas, classifica as arquiteturas através de dois fatores: sequência de instruções e sequência de dados que são transmitidos para o processador. A partir disso, (FLYNN, 1972), entende a sequência de instruções ou fluxo de instruções como sendo um conjunto de instruções a serem executadas, e a sequência de dados, ou fluxo de dados, como sendo conjunto de dados. Com base neste conceito, a taxonomia de Flynn divide as arquiteturas computacionais em 4 categorias:

A. SISD – Single Instruction, Single Data: Esta é a classificação mais clássica, que considera que um único fluxo de instruções de entrada e um único fluxo de dados de saída. Nesta taxonomia, estão representados a maioria dos computadores atuais que são baseados no modelo de Von Neumann, uma vez que fazem uso de processadores sequenciais onde uma instrução completa é executada por vez, e cada uma das instruções manipula um dado especifico ou dado da operação.
B. SIMD – Single Instruction, Multiple Data: Nesta taxonomia, enquadram-se os sistemas em que múltiplos fluxos de dados são executados, simultaneamente, em um único ciclo de instrução. Esta arquitetura utiliza uma organização do tipo mestre/escravo, onde N elementos de processamento (escravos) são supervisionados por uma única unidade de controle (mestre). Todos os processadores recebem a mesma instrução para executar, mas recebem diferentes faixas de dados para processar. Nesta categoria enquadram-se as GPUs, FPGAs e os grandes computadores vetoriais.
C. MISD – Multiple Instruction, Single Data: Este tipo de arquitetura trabalha com múltiplos fluxos de entrada e apenas um único fluxo de saída, ou seja, nestes tipos de sistemas existem N elementos de processamento, onde cada um deles possui a sua unidade de controle e de processamento, contudo, dividem o mesmo espaço de memória e executam instruções distintas sobre um mesmo conjunto de dados. Esta é uma estrutura pouco aplicada e muitos autores chegam a nem a considera-las, entretanto, esta arquitetura é geralmente implementada em sistemas que precisam ter um certo nível de redundância a falhas. Exemplos desta arquitetura são os processadores vetoriais e em alguns casos, (NAKAMURA, 2005), cita que os "*macropipelines*", também se enquadram nesta categoria, já que a saída de uma unidade de processamento serve de entrada para outra unidade de processamento.
D. MIMD – Multiple Instruction, Multiple Data: Esta classe pode ser considerada a mais avançada tecnologicamente, sendo que a maioria dos sistemas multiprocessados (multiprocessadores e os multicomputadores) estão dentro desta categoria. Os sistemas MIMD operam com múltiplos fluxos de instruções de entrada e múltiplos fluxos de dados de saída. De modo que, cada unidade de processamento é controlada pela sua própria unidade de controle, processando instruções independentes sobre fluxos de dados diferentes. Nesta categoria, se enquadram as tecnologias de processamento distribuído, como Message Passing Interface (MPI) e Parallel Virtual Machine (PVM).

2.3.3 GPGPU - General-Purpose Computing on Graphics Processing

A evolução da área computação está fortemente entrelaçada com o aumento do poder de processamento. De tal forma, que até os anos 2000, os microprocessadores baseados em uma única central de processamento, principalmente os da família Intel Pentium e AMD Opteron, impulsionaram o progresso no campo computacional, ao viabilizar que grande poder de processamento fosse alcançado a um baixo custo. Contudo, desde 2003, a evolução deste tipo de arquitetura de microprocessador vem diminuindo, uma vez que, o método tradicional da indústria de simplesmente aumentar o número de transistores no processador para assim incrementar a sua velocidade de processamento se mostrou ineficiente, devido ao alto consumo de energia e problemas de dissipação de calor, estes fatores limitam o aumento da frequência do *clock* e o número de atividades que podem ser executadas em cada período de *clock*, dentro de uma única CPU.

Em virtude disso, praticamente todas as desenvolvedoras de processadores começaram a migrar para arquiteturas de microprocessadores com múltiplas unidades de processamento, isto na verdade significou o uso múltiplos núcleos de processamento em um único processador, para assim aumentar o poder de processamento (HWU & KIRK, 2012).

Entretanto, também em 2003, outro tipo de arquitetura de processadores que ganhou força foi a de muitos núcleos (ou *manycores*), como as GPUs que são projetadas com núcleos pequenos, mas em grande quantidade, voltados para executar threads em paralelo (*multithread*). Nos últimos anos, a arquitetura de processadores com muitos núcleos se consolidou, com as GPUs, assumindo papel principal quando o assunto é poder de processamento computacional. Pois, as CPUs são projetadas para maximizar a velocidade de execução dos programas sequenciais, possuindo poucos núcleos e manipulando poucos threads. Enquanto que as GPUs, em contrapartida, com sua arquitetura de muitos núcleos são projetadas com milhares de núcleos e conseguem trabalhar com milhares de threads. Por exemplo, um processador de alta performance Intel Core I9 Extreme de última geração (*Skylake-X*), possui 18 núcleos e trabalha com 36 threads. Enquanto, que uma GPU NVIDIA GTX 1080 TI de última geração (Pascal) tem 3584 núcleos podendo suportar 54344 threads rodando de forma simultânea (cada núcleo consegue executar até 16 threads concorrentemente).

Devido a diferença conceitual de arquitetura, a performance das GPUs passou superar, por uma grande margem, o desempenho de processamento das CPUs, de forma que os

pesquisadores e programadores passaram a usar as GPUs para executar aplicações de propósito geral que antes só eram executadas pela CPU, (PINHEIRO, 2017).

2.3.3.1 Diferença Básica entre a CPU e a GPU

Atualmente, as CPUs também usam uma arquitetura mais orientada para o processamento paralelo, buscando aumentar seu desempenho. No entanto, a CPU não consegue atingir o alto poder de processamento das GPUs, pois existem diferenças fundamentais entre suas arquiteturas.

A GPU é um hardware projetado para alcançar um alto desempenho através de um paralelismo maciço, de modo que, o layout de sua arquitetura, como mostrado na Figura 6, é feito de forma que, a maior parte da sua superfície é composta por Unidades Lógicas Aritméticas (ULA), ou em inglês *Arithmetic Logic Units* (ALU), e somente uma pequena região é dedicada ao controle e cache. Em contraste, a maior parte da arquitetura da CPU é reservada para as unidades de controle e cache, deixando apenas uma pequena área para as unidades de processamento. A diferença de arquiteturas e por consequência de desempenhos deve-se ao propósito distinto dos dois dispositivos. A CPU é projetada para ser mais flexível, capaz de lidar com várias tarefas distintas, tentando manter um equilíbrio entre o poder de processamento e a funcionalidade de propósito geral, enquanto a GPU sacrifica a flexibilidade em troca de um alto poder de processamento, tornando-se mais restrita, mas também capaz de executar uma enorme quantidade de cálculos paralelos.

Figura. 6 – Arquitetura simplificada da CPU e da GPU
Fonte: Programming Massively Parallel Processors, Second Edition: A Hands-on Approach, Pg4

2.3.3.2 Arquitetura da GPU

A GPU utilizada no desenvolvimento deste trabalho foi a GeForce GTX 1070, esta GPU implementa a arquitetura Pascal, que é a sexta geração de dispositivos NVIDIA capazes de usar a tecnologia CUDA. A GTX-1070 é dividida em 3 Graphics Processing Cluster (GPCs), onde cada GPC, por sua vez, possui 5 Streaming Multiprocessors (SMs) com 128 processadores CUDA em cada SM. A nível de hardware, o SM é o principal componente da GPU, porque é nele que o trabalho é realmente realizado através dos processadores CUDA que executam as operações matemáticas para os threads. Além disso, para controlar a execução dos processadores CUDA, os SMs têm o *Warp Schedulers* responsável por organizar o processo de execução em grupos de 32 threads, denominados *Warps*. Cada *Warp* é alocado na *Dispatcher Unit* onde as instruções são executadas. A Figura 7 mostra o esquema das SMs das GPU que utilizam a arquitetura Pascal, onde a SM é dividida em 4 partes, cada uma contendo um agendador de warps responsável por distribuir os threads, em grupos de 32 threads, chamados warps, para suas unidades de execução.

Figura. 7 – A arquitetura do Streaming Multiprocessor (SM) de uma GPU da arquitetura NVIDIA Pascal.
Fonte: GeForce GTX 1080 Whitepaper, Pg18

2.3.4 CUDA - Compute Unified Device Architecture

As GPUs foram originalmente desenvolvidas para funcionar apenas com imagens, mapeando *arrays* contendendo grandes conjuntos de dados (pixels e vértices da imagem) em matrizes a serem executadas por threads em paralelo. No entanto, ao longo do tempo, percebeu-se que as aplicações que processam um conjunto maciço de dados podem se beneficiar do paradigma de

programação paralela usado nas GPUs, com o requisito que os dados da aplicação precisam ser mapeados para a forma de matriz.

Além disso, o uso de GPUs vem mudando a maneira como as aplicações de alto desempenho são estruturadas. Uma vez que, os núcleos de uma GPU são projetados para funcionar usando o modelo de programação SIMD (Single Instruction Multiple Data), onde uma instrução é empregada em vários dados em paralelo. Portanto, ao desenvolver um aplicativo que será executado na GPU, o algoritmo deve ser estruturado com base em programação paralela. Por esse motivo, este trabalho usa o CUDA, que é uma plataforma e modelo de programação paralela, desenvolvido pela NVIDIA, como base principal para o projeto. A plataforma CUDA foi escolhida principalmente por sua flexibilidade, permitindo o uso de linguagens de uso geral, como Fortran, C ou Python. Nesse sentido, embora o programa original do módulo de Dispersão de Pluma e Dose utilize a linguagem Fortran, no trabalho não foi utilizado o CUDA Fortran pois seria preciso comprar o compilador CUDA Fortran PGI. Sendo assim, nesse trabalho se optou por usar o CUDA C no desenvolvimento.

O CUDA C funciona como uma extensão de linguagens C, adicionando recursos que facilitam o desenvolvimento paralelo. Um desses recursos é a capacidade de executar funções em C, chamadas de funções kernel, que quando invocados são executados simultaneamente N vezes por N processadores CUDA. Nesta perspectiva, o modelo de programação empregado no CUDA é fortemente integrado ao hardware das GPUs NVIDIA. Uma vez que, quando uma kernel é iniciada, uma informação solicitando a criação de uma grid de threads é enviada da CPU para a GPU, dentro desta grid os threads são agrupadas em blocos, como mostrado na Figura 8. Ao receber a solicitação, a GPU usa a sua *Work Distributed Unit* para distribuir os blocos de threads para os Streaming Multiprocessors (SMs), procurando aqueles que dispõem no momento dos recursos necessários para executar a tarefa. Este processo visa intensificar o paralelismo na GPU, distribuindo uniformemente o trabalho entre as SMs.

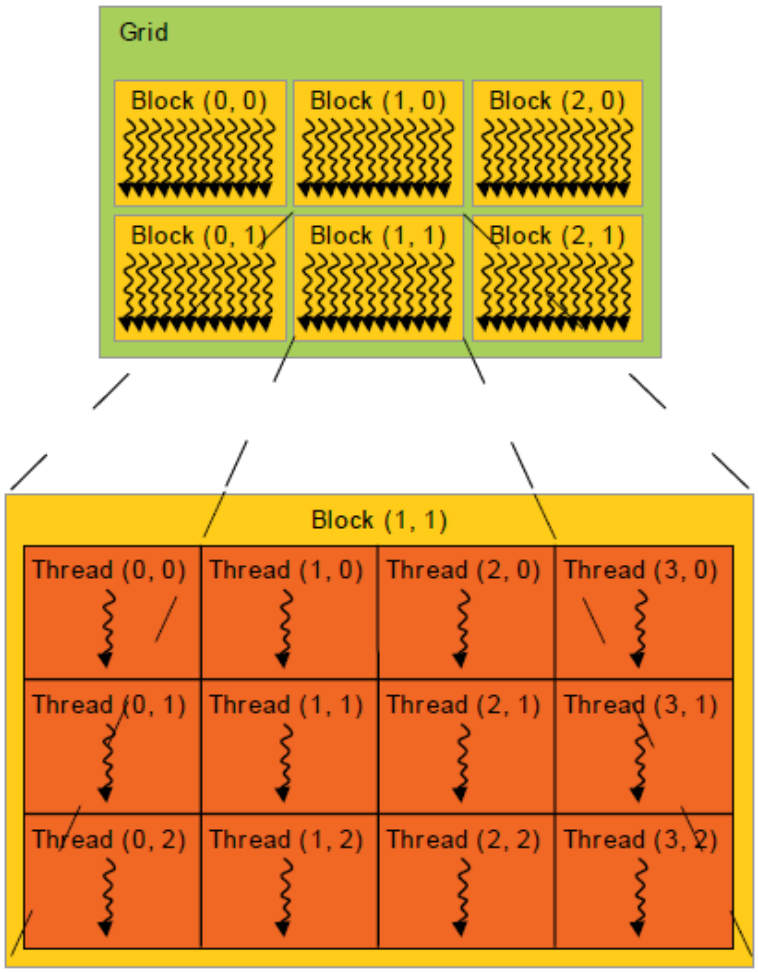

Figura. 8 – Exemplo de uma grid com 6 blocos com cada um contendo 12 threads
Fonte: CUDA C Programming Guide, Pg11

Para ilustrar a estrutura básica de um programa em CUDA C, a Figuras 9 e a Figura 10 apresentam uma função que soma 2 vetores, uma versão em C e outra em CUDA C, respectivamente:

```
void soma_Vetores_CPU(int size, int *vec_A, int *vec_B, int *vec_C) {
    for (int i = 0; i < size; i++) {
        vec_C[i] = vec_A[i] + vec_B[i];
    }
}
```

Figura. 9 – Função sequencial em C somando 2 vetores

```
__global__ void soma_Vetores_GPU(int *vec_A, int *vec_B, int *vec_C) {
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    vec_C[i] = vec_A[i] + vec_B[i];
}
```

Figura. 10 – Função paralela em CUDA somando 2 vetores

Observa-se que em vez de um loop *for* (Figura 9) para iterar os vetores e assim somar os seus elementos, a função CUDA (Figura 10) utiliza os threads para realizar as operações de soma, especificando a posição global do thread, representado pela variável *i*, responsável por cada posição especifica do vetor. Ademais, uma instrução *if* é geralmente adicionada ao código CUDA para garantir que não há acessos fora dos limites dos vetores. O processo de soma da Figura 10 ocorre porque, em CUDA, os processadores CUDA operam em paralelo, com cada da operação de soma, sendo executada de forma independente, para cada posição *i* dos vetores. Contudo, para que isso seja possível, para que cada thread possa processar um elemento vetorial resultante, é preciso definir com antecedência a posição global do thread. Sendo assim, a Figura 11 exemplifica o processo de indexação (identificação do index global do thread) em CUDA, onde: gridDim.x (é o número de blocos), blockDim.x (é o número de threads em cada bloco), blockIdx.x (é o índice do bloco atual dentro a grid) e threadIdx. X (é o índice do segmento atual dentro do bloco).

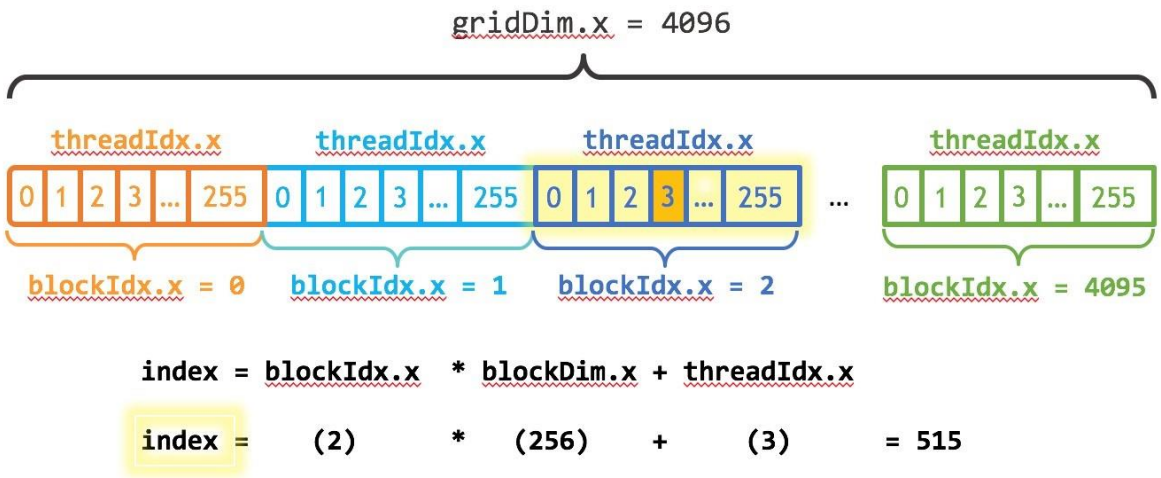

Figura. 11 – Exemplo do padrão de indexação em CUDA
Fonte: An Even Easier Introduction to CUDA, Disponível em: < https://devblogs.nvidia.com/even-easier-introduction-cuda/>. Acessado em: 02/03/2018

Além disso, em CUDA, a CPU e a GPU possuem espaços de memória distintos, de modo que é necessário realizar transferência de dados entre as duas memórias antes e depois da invocação da kernel. O algoritmo na Figura 12 mostra as etapas básicas para executar uma função kernel em CUDA, as etapas são as seguintes: 1 - alocação de memória GPU, 2 - Dados de entrada transferidos da memória da CPU para a memória GPU, 3 inicializações do kernel e 4 - Dados de saída transferidos para a memória da CPU

```
// Passo 1: Alocar Espaço de Memória na GPU
cudaMalloc((void**)&d_c, size*sizeof(int));
cudaMalloc((void**)&d_a, size*sizeof(int));
cudaMalloc((void**)&d_b, size*sizeof(int));

// Passo 2: Copiar os Dados de Entrada da Memória do Host
para a Memória do Device
cudaMemcpy(d_a, a, size*sizeof(int), cudaMemcpyHostToDevice);
cudaMemcpy(d_b, b, size*sizeof(int), cudaMemcpyHostToDevice);

// Passo 3: Invocar a Função Kernel
soma_Vetores_GPU << <blocks, threadsPerBlock >> >(d_c, d_a,
d_b);

// Passo 4: Copiar os Dados de Saída da Memória do Device
para a Memória do Host
cudaMemcpy(a, d_a, size*sizeof(int), cudaMemcpyDeviceToHost);
cudaMemcpy(b, d_b, size*sizeof(int), cudaMemcpyDeviceToHost);
```

Figura. 12 – Etapas do processo de transferência de dados entre memórias.

É importante enfatizar que o processo de transferência entre memórias pode ser uma desvantagem na aceleração de uma aplicação. Já que, quanto maior a quantidade de dados para transferir, maior é o tempo de transferência e caso o tempo de processamento da aplicação for pequeno, em comparação ao tempo de transferência de dados, os ganhos com paralelismo podem ser reduzidos.

2.3.5 A Técnica de Paralelização Grid Stride Loop

A programação em CUDA pode ser utilizada em conjunto com diversas técnicas de paralelização. Principalmente se algoritmo original contiver loops, uma vez que existem técnicas paralelas de reestruturação de loops sequenciais. Sendo assim, neste capitulo será aborda técnica de paralelização grid stride loop que foi usada no desenvolvimento desse trabalho.

Deste modo, para compreender como funciona a técnica grid stride loop é necessário entender como um loop geralmente é tratado no CUDA, para isso, o exemplo da Figura 13 que mostra uma implementação básica de um loop será usado.

```
// Algoritmo Original
void Grid_Stride_Loop(int n, float a, float *x, float *y) {
    for (int i = 0; i < n; ++i)
        y[i] = a * x[i] + y[i];
}
```

Figura. 13 – Algoritmo demonstrando o loop sequencial

Segundo (HARRIS, 2013), a função da Figura 13 seria reescrita em CUDA, de acordo com a Figura 14. Essa abordagem é chamada de kernel monolítico, pois, considera que uma única *grid* possui threads suficientes para processar cada um dos elementos de um vetor, ou seja, presume-se que ao iniciar uma kernel o número de threads enviados é suficiente para cobrir o vetor em sua totalidade, de modo que, cada *thread* seja responsável por uma posição vetorial. No entanto, a limitação desta técnica é quando o vetor possui um elevado número de elementos e não é possível alocar os threads necessárias para cobrir todo o vetor em uma única chamada. Neste tipo de situação, a técnica grid-stride loop pode ser aplicada, pois esta técnica não pressupõe que uma única *grid* de threads é capaz de iterar todo o vetor, e em vez disso, usa um *loop* para fazer iterações no vetor com um tamanho de *grid* de cada vez.

```
// Algoritmo CUDA
__global__ void Grid_Stride_Loop (int n, float a, float *x, float *y) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < n) y[i] = a * x[i] + y[i];
}
```

Figura. 14 – Algoritmo CUDA tradicional

A Figura 15 mostra a codificação utilizando a técnica grid stride_loop, é possível notar que o passo do *loop* é blockDim.x * gridDim.x (número de threads no bloco * número de blocos na grid) que é o total de threads da *grid*. Supondo que haja 2048 threads na *grid*, o segmento 0 calculará os elementos 0, 2048, 4096, etc. Assim, usando um loop com passo igual ao tamanho da grid, é assegurado que todo o endereçamento dentro dos *warps* seja uma unidade-passo, obtendo o uso máximo de threads, assim como na versão monolítica. Além disso, em uma situação em que o grid-stride loop é usado, mas, a grid é grande o suficiente para cobrir todo o vetor, o custo computacional é essencialmente igual ao da instrução *if* usada na abordagem monolítica, uma vez que, o incremento do loop só será avaliado quando a condição do loop for verdadeira.

```
// Algoritmo Grid-Stride Loop
__global__ void Grid_Stride_Loop (int n, float a, float *x, float *y) {
    for (int i = blockIdx.x * blockDim.x + threadIdx.x;
    i < n; i += blockDim.x * gridDim.x)
        y[i] = a * x[i] + y[i];
}
```

Figura. 15 – Algoritmo CUDA utilizando grid-strid loop

3. IMPLEMENTAÇÃO DO ALGORITMO PARALELO DE TRANSPORTE E DIFUSÃO DE RADIONUCLÍDEOS

O trabalho descrito neste documento, tem como principal objetivo o desenvolvimento de uma versão paralela, baseada em GPU, do módulo Dispersão de Pluma e Dose, mais especificamente da parte de transporte e difusão do módulo, do sistema de dispersão atmosférica de radionuclídeos (SDAR) da CNAAA. De forma a permitir que seja possível executar o módulo em tempo real utilizando uma malha com um nível mais elevado de refinamento. Sendo assim, neste capitulo, será abordado o processo de desenvolvimento e implementação do algoritmo paralelo, baseado em GPU, do transporte e difusão de radionuclídeos.

Posto isto, um dos principais critérios levado em consideração durante o desenvolvimento, foi que o algoritmo paralelo deveria ser elaborado com modificações mínimas, em relação ao modelo de transporte e difusão utilizado no módulo, de forma que, o resultado de saída do algoritmo paralelo teria que reproduzir os mesmos resultados do algoritmo original. Uma vez que, o algoritmo original, embora tenha sido desenvolvido na década de 80, ainda é utilizado na CNAAA e os resultados gerados por ele são validados pela Eletrobrás Eletronuclear.

Neste sentido, para que fosse possível utilizar o CUDA/C para o desenvolvimento da versão paralela do programa, foi necessário elaborar uma nova versão sequencial completa do módulo Dispersão de Pluma e Dose, utilizando a linguagem C. Aproximadamente 2500 linhas de código foram feitas.

3.1 ESTRUTURA DO ALGORITMO DO MÓDULO DISPERSÃO DE PLUMA E DOSE

A estrutura básica do programa do módulo Dispersão de Pluma e Dose é exibida na Figura 16. Enquanto que a Tabela 2 mostra o tempo relativo das funções principais que compões o módulo.

```
Função Dispersao_Pluma_Calculo_Dose
        Leitura_Terreno;
        Leitura_Campo_Vento;
        Leitura_Taxa_Media_Liberacao_Bufadas;
        Tradif;
        Dosem;
```

Figura. 16 – Estrutura do algoritmo do módulo Dispersão de Pluma e Dose

Tabela 2 – Tempo de processamento relativo das funções do módulo Dispersão de Pluma e Dose

Nome da Função	Tempo de Processamento Relativo
Leitura_Terreno	0,29%
Leitura_Campo_Vento	3,56%
Leitura_Taxa_Media_Liberacao_Bufadas	0,06%
Dosem	1,05%
Tradif	95,05%

3.1.1 Funções de Entrada

O programa Dispersão_Pluma_Calculo_Dose possui três funções de entrada de dados. Sendo a primeira, a função Leitura_Taxa_Media_Liberacao_Bufadas, esta ocupa 0,06% do tempo de processamento do programa, sendo responsável por ler o arquivo gerado pelo módulo Termo Fonte, que possui as informações sobre as características da liberação, mais especificamente o arquivo contêm os seguintes dados: os valores da atividade dos radionuclídeos em cada bufada liberada no ciclo; a velocidade de saída das bufadas liberadas; a altura da liberação das bufadas; e a temperatura de liberação das bufadas. Já a segunda função de entrada é a Leitura_Campo_Vento, consumindo 3,56% do tempo de processamento do módulo, esta função é encarregada de ler os dados contidos no arquivo gerado pelo módulo Campo de Vento. Este arquivo possui as seguintes informações sobre o campo de velocidade de vento tridimensional: os componentes de velocidade do vento na direção X, Y e Z; o campo de classes de estabilidade; a taxa de arraste da chuva; e a temperatura do ar. A última função de entrada é a Leitura_Terreno, está função ocupa 0,29% do tempo de execução do módulo, e cabe a ela realizar o trabalho de leitura do arquivo que contém a topografia do terreno. O arquivo em questão, possui o formato de uma matriz de tamanho 67 x 43, contendo 2881 células. Sendo que à cada uma destas células é atribuído um valor que varia de 0 a 7, esta escala de valores representa os 8 níveis de altura variáveis, mostrados na Tabela 1, no capitulo 2.2. A Figura 17 apresenta a configuração do arquivo de topografia utilizado no programa. Já

a Figura 18 exibe este mesmo arquivo, só que empregando cores diferentes para cada nível de altura, para facilitar a visualização do relevo da região.

Figura. 17 – Arquivo de Topografia, as células em branco representam os valores 0 que foram retirados para melhor leitura
Fonte: Modelo Computacional Paralelo Baseado em GPU para Cálculo do Transporte e Difusão de Radionuclídeos, Pg75.

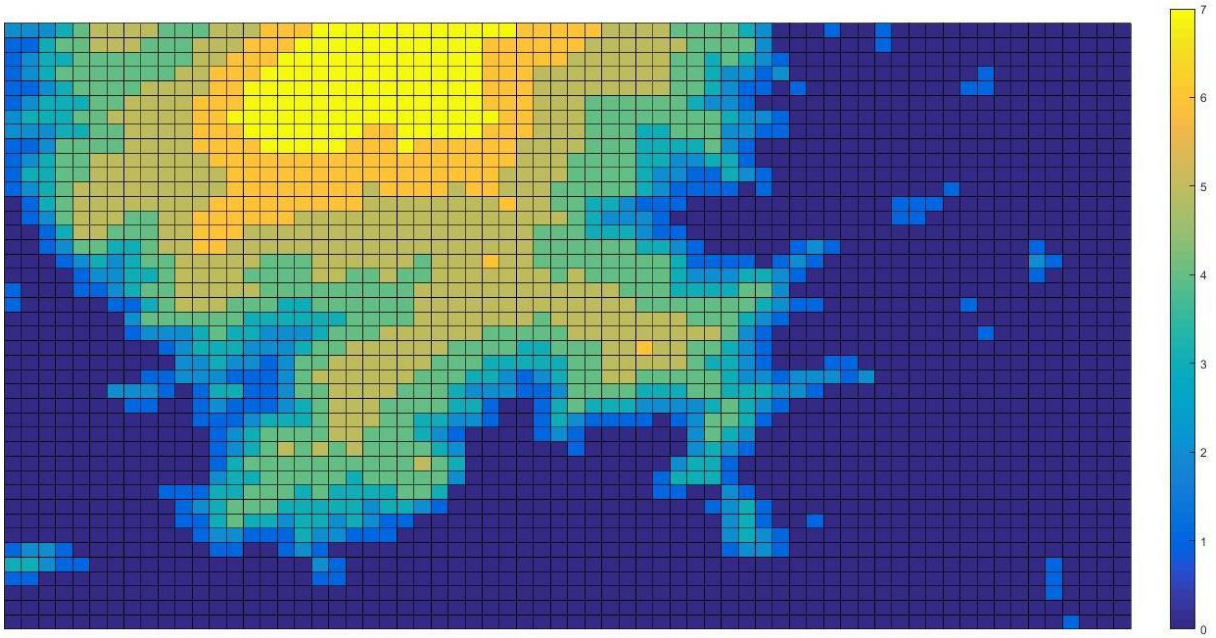

Figura. 18 – Arquivo de Topografia com Cores
Fonte: Modelo Computacional Paralelo Baseado em GPU para Cálculo do Transporte e Difusão de Radionuclídeos, Pg76

3.1.2 Dosem

A função Dosem, embora ocupe apenas 1,05% do tempo de processamento do programa, é uma das principais funções do módulo. Sendo responsável por calcular a distribuição espacial da dose radiológica no terreno, a partir das tabelas geradas na função Tradif contendo a concentração média de iodos e particulados a nível do solo e para gases nobres a nível do solo, 50m, 125m e 275m. Na função Dosem são calculadas a taxa de dose média no ciclo e as doses acumuladas desde o início do acidente para a tireoide e pulmão, devido à inalação de iodos, particulados e gases nobres e para o corpo inteiro devido à imersão na nuvem onde somente existe a contribuição dos gases nobres usando-se o modelo de pluma finita.

3.1.3 Tradif

A Tradif é a principal função do programa, sendo a função que realiza os cálculos de transporte e difusão de radionuclídeos na atmosfera, em virtude disso, esta função é a que mais impacta no tempo de execução do módulo, ocupando 95,05% do tempo de processamento total do programa. A função Tradif é composta por várias subfunções mostradas na Figura 19. Além disso, a Tabela 3 mostra o tempo de processamento relativo de cada uma dessas funções. Entre elas, a função Calculo_Concentracao_Media_Radionuclídeos

(CCMR) representa 99,71% do tempo de execução total da Tradif, sendo a função mais lenta do módulo. Em virtude disso, esta função tornou-se o principal alvo para a paralelização.

Função **Tradif**

```
Iniciacao_Parametros_Bufadas;
Calculo_Liberacao_Bufadas;
Calculo_Nivel_Bufadas;
Interpolacao_Velocidade_Vento;
Verificao_Desvio_Bufadas;
Calculo_Difusao_Bufadas;
Calculo_Preliminar_Concentracao_Media_Radionuclideos;
Calculo_Concentracao_Media_Radionuclideos;
Registro_Parametros_Bufadas;
Calculo_Deplecao_Radionuclideos_Bufadas;
Invercao_Indexes_Bufadas;
```

Figura. 19 – Estrutura do algoritmo da função Tradif

Tabela 3 – Tempo de processamento relativo das funções da Tradif.

Nome da Função	Tempo de Processamento Relativo
Iniciacao_Parametros_Bufadas	0,07%
Calculo_Liberacao_Bufadas	0,008%
Calculo_Nível_Bufadas	0,0003%
Interpolacao_Velocidade_Vento	0,0002%
Verificao_Desvio_Bufadas	0%
Calculo_Difusao_Bufadas	0,00008%
Calculo_Preliminar_Concentracao_Media_Radionuclideos	0,15%
Calculo_Concentracao_Media_Radionuclideos	99,77%
Registro_Parametros_Bufadas	0,00004%
Calculo_Deplecao_Radionuclideos_Bufadas	0,0002%
Invercao_Indexes_Bufadas	0%

3.1.3.1 Cálculo da Concentração Média de Radionuclídeos

A função Calculo_Concentracao_Media_Radionuclideos (CCMR) é responsável por calcular a contribuição das bufadas, com base na trajetória percorrida por cada bufada durante o ciclo, para a concentração média de radionuclídeos (gases nobre, iodos e particulados a nível do solo), em cada ponto da malha. De modo que, a concentração em cada ponto da malha consiste no somatório das contribuições médias de todas as bufadas naquele determinado ponto.

Neste trabalho, foram desenvolvidas duas versões da função CCMR, uma versão sequencial usando a linguagem de programação C que serviu como base para a elaboração da versão paralela utilizando CUDA.

3.2 DESENVOLVIMENTO DA VERSÃO SEQUENCIAL DA FUNÇÃO DO CÁLCULO DA CONCENTRAÇÃO MÉDIA DE RADIONUCLÍDEOS (CCMR)

A fim de facilitar a paralelização e procurando obter um maior entendimento da estruturação do algoritmo, uma versão sequencial do programa do módulo Dispersão de Pluma e Dose foi desenvolvida utilizando a linguagem de programação C.

A função sequencial CCMR foi investigada a fim de detectar os pontos do código que deveriam receber maior atenção durante o trabalho de paralelização. Portanto, o primeiro ponto observado foi o local de invocação da função CCMR, dentro da função Tradif. Conforme mostrado na Figura 20, a função CCMR é invocada dentro de um loop aninhado (*nested loop*) composto por três loops. Sendo que os dois loops internos são executados com valores constantes, Exec_Namos = 4 e Exec_Nadv = 15. Enquanto que o número de iterações do loop externo (NPUFS) é variável, podendo ser alterado de acordo com o número de bufadas na malha, sendo que a cada ciclo de execução do programa o número de bufadas na malha pode ser modificado. Neste contexto, em uma simulação configurada para executar o módulo Dispersão de Pluma e Dose durante 20 ciclos e usando o nível original de refinamento da malha (67 x 43 x 8), a função CCMR é invocada 34.320 vezes.

Além disso, o outro ponto observado foi a análise da organização interna da própria função CCMR, para identificar os pontos potenciais para a paralelização. A organização do algoritmo da função CCMR é mostrado na Figura 21. A função CCMR é controlada por cinco loops, os dois loops externos (I2. J2), representam as coordenadas X e Y no plano cartesiano e são responsáveis pela determinação da distância entre um ponto de concentração e o centro da pluma radioativa. O loop seguinte, (NUIP), é usado para calcular a concentração média de iodo e particulados em um ponto (X, Y) no plano a nível do solo. Os últimos dois loops internos (KCT e NUGN), são, respectivamente, um para adicionar a coordenada Z ao plano cartesiano e o outro para controlar o cálculo da concentração média de gases nobres em um ponto (X, Y, Z) do plano. O número de iterações de cada um destes loops, considerando o nível de refinamento original da malha, é I2 (Coordenada X) = 67, J2 Coordenada Y = 43, NUIP = 32, KCT (Coordenada Z) = 4 e NUGN = 14.

```
void Tradif() {
    ...
    // Loop de designação da bufada
    for (NP = 1; NP <= NPUFS; NP++){
        ...
        // Designação dos looops de advecção
       for (M = 1; M <= Exec_Nadv; M++){
           ...
           // Transporte e disseminação em intervalos de amostragem
           for (N = 1; N <= Exec_Namos; N++){
               ...
               Calculo_Concentracao_Media_Radionuclideos(<parâmetros>);
           } // N
           ...
       } // M
       ...
    } // NP
    ...
}
```

Figura. 20 – Parte do código sequencial da função Tradif que mostra a posição de invocação da função CCMR

```
void Calculo_Concentracao_Media_Radionuclideos (<parâmetros>){
    // Declarações e Inicializações
    ...
    // Calculo das Concentrações: // Determinação da distância do ponto da
malha ao centro de massa da bufada
    for (IC = I1; IC <= I2; IC++){
        X = IC * DX - DX * 0.5 - XP;
        XX = X * X;
        for (JC = J1; JC <= J2; JC++){
           Y = JC * DY - DY * 0.5 - YP; YY = Y * Y;
           AUX = (XX + YY) / DSIGH2; T1 = Esp(<parâmetros>);
           // Calculo de Altura para Reflexão no Solo
           KT = Iht[IC][JC]; ZT = ALTURA[KT + 1];
           H = ZP - ZT;
           H2 = H * H;
           T2 = Esp(<parâmetros>) * FATOR;
           // Concentração média de iodos e particulados no solo (ci / m3)
           for (NU = 1; NU <= NUIP; NU++){
               T3 = QPUF[NU][NP] / CONST1; CONCIP[NU][JC][IC] =
               CONCIP[NU][JC][IC] + T2 * T3;
          } // NU
          // Concentração média de de gases nobres (ci / m3)
          ZZ = 0.0;
          for (KC = 1; KC <= KCT; KC++){
             ZR = ZT + ZZ - ZP;
             ZR2 = ZR * ZR; AUX = powf((ZR + 2.0*H),2);
             T4 = Esp(<parâmetros>);
             T5 = Esp(<parâmetros>);
             T6 = (T4 + T5) * T1 * FATOR;
             for (NU = 1; NU <= NUGN; NU++){
                T7 = QPUF[NU + NUIP][NP] / CONST2;
                CONCGN[NU][KC][JC][IC] = CONCGN[NU][KC][JC][IC] + T7*T6;
            } // NU
            ZZ = ZZ + Hnz[KC];
         } // KC
       } // JC
    } // IC
    ...
}
```

Figura. 21 – Função sequencial CCMR

3.3 DESENVOLVIMENTO DA VERSÃO PARALELA DA FUNÇÃO DO CÁLCULO DA CONCENTRAÇÃO MÉDIA DE RADIONUCLÍDEOS (CCMR)

O algoritmo paralelo da função CCMR foi elaborado com o uso da linguagem de programação CUDA / C, explicada no capitulo 2.3.5. No entanto, como mostrado na Figura 21, a versão sequencial da função CCMR possui vários loops que exigem um alto custo computacional, e embora o número de loops no código indique um grande potencial de paralelização, esta não é uma tarefa trivial.

Em virtude disso, a técnica de paralelização grid_stride loop, abordada no capitulo 2.3.6, foi utilizada na elaboração do algoritmo paralelo, pois está técnica incorpora uma maior

flexibilidade ao código desenvolvido, já que os seus *loops stride* permitem que o dispositivo CUDA seja capaz de trabalhar com problemas de qualquer tamanho, mesmo que este exceda o tamanho máximo da grid que o dispositivo CUDA suporta.

Neste sentido, a técnica grid-stride loop se encaixa na situação proposta por esse trabalho, que é fazer com que o módulo Dispersão de Pluma e Dose seja capaz de operar com domínios computacionais refinados, pois aumentar o nível de refinamento significa ampliar o tamanho do problema, já que a quantidade de iterações dos loops e número de células que precisaram ser processadas crescem significantemente, de acordo com o nível refinamento.

A partir disso, a técnica grid-stride loop foi levada em consideração mesmo na configuração dos parâmetros de execução da função kernel CCMR. Considerando que, (HARRIS, 2013), instrui que os parâmetros de execução da função kernel devem ser configurados de maneira a limitar a quantidade de blocos na grid, para ajustar o desempenho da kernel. Uma vez que, ao se utilizar a técnica grid-stride loop na função kernel é possível reutilizar o mesmo thread para diversos cálculos, e isso amortiza o custo computacional de criar e destruir threads.

Neste sentido, a Figura 22 mostra a invocação da função kernel CCMR e os seus parâmetros de configuração de execução, definidos por: DimBlock (dimensões do bloco de threads) e DimGrid (dimensões da grid de blocos). Além disso, ainda na Figura 22, é possível observar que DimBlock possui duas dimensões com dimBlock.x = 32 e dimBlock.y = 16, ou seja um bloco com 512 threads. Estes valores para as dimensões do bloco foram obtidos de forma empírica, contudo, ainda sim, a recomendação de (HWU e Kirk, 2013), que em geral as dimensões dos blocos de threads devem ser múltiplos de 32 devido a razões de eficiência de hardware, foi seguida.

```
void Tradif() {
    ...
    // Definindo a quantidade Threads por Bloco e a quantidade de Blocos na Grade
    int n = 64;
    dim3 DimGrid(ceil(NX/n), ceil(NY/n));
    dim3 DimBlock(32, 16);
// Invocando a Kernel
   for (short int index = 0; index < COUNT; index++) {
       Calculo_Concentracao_Media_Radionuclideos << <DimGrid, DimBlock >> > (<parâmetros>);
   } // index
   ...
}
```

Figura. 22 – Invocação da função kernel CCMR

Por outro lado, para definir DimGrid, pode-se usar o método tradicional exibido na Figura 23, onde: n é o número de elementos de um array; blockSize é o número de threads no bloco; e ceil é uma função C para arredondamento para cima. O uso deste método garante que o número de blocos de threads na grid será suficiente para cobrir todos os elementos do array.

Entretanto, na função kernel CCMR o DimGrid foi configurado seguindo a recomendação de (HARRIS, 2013), explicada anteriormente, de modo que, para diminuir o número de blocos de threads na grid, foi utilizada uma constante denominada N, com valor de 64 (valor obtido a partir de testes empíricos), no lugar do blockSize, para assim alcançar um número menor de blocos na grid.

```
dim3 DimGrid (ceil(n/blockSize))
```

Figura. 23 – Método tradicional para definir as dimensões da grid

Ainda em relação a invocação da função kernel CCMR (Figura 22), o ponto de chamada da função foi modificado. Já que no algoritmo sequencial, a função CCMR era invocada dentro de um loop aninhado, que além da invocação, também era responsável por iterar diversos cálculos. Portanto, para evitar problemas de desempenho, uma vez que a cada invocação da função kernel a GPU precisaria aguardar o processamento dos outros cálculos sequenciais contidos nos loops, para a partir disso executar novamente a função. O ponto de invocação da função kernel CCMR foi movida para fora dos loops aninhados e foi colocada dentro de um único loop, responsável apenas por chamar a função kernel a cada iteração. Além disso, um problema detectado durante a realocação do ponto de invocação da função kernel CCMR, foi que alguns dos parâmetros da função CCMR eram variáveis, e estas eram calculadas dentro dos loops aninhados e modificadas a cada iteração desses loops. Sendo assim, para solucionar este problema, foram criados vetores para armazenar os valores dessas variáveis a cada iteração dos loops. Ademais, ao final da última iteração dos loops aninhados, os valores desses vetores são transferidos para a memória da GPU, sendo usados como parâmetros na chamada da função kernel CCMR.

A Figura 24 mostra a função kernel CCMR usando a técnica gride stride loop. Se comparada com a Figura 21, é possível observar que os dois loops externos foram substituídos por dois *loops stride*. Além disso, a parte do código responsável por realizar os cálculos da concentração média no algoritmo sequencial, foi substituída por duas funções device (funções que só podem ser invocadas pela kernel e executadas pela GPU) no algoritmo paralelo.

A função device Concentracao_Media_Iodo_Particulados (CMIP), mostrada na Figura 25, calcula a concentração média de iodo e particulados. Enquanto que a função device Concentracao_Media_Gases_Nobres (CMGN), mostrada na Figura 26, estima a concentração média de gases nobres. Em ambas as funções, os loops tradicionais que controlavam os cálculos de concentração média foram preservados, apenas com uma ligeira alteração no

processo de indexação. Nas funções device, os *loops_stride* não foram usados, porque estas não são funções kernels, cujos threads são alocados durante a invocação.

A abordagem paralela desenvolvida para a função CCMR permite que a contribuição das bufadas para a concentração de radionuclídeos na malha seja calculada de forma paralela. Uma vez que, na versão sequencial as concentrações gases nobres, iodos e particulados são calculadas para um ponto da malha tridimensional de cada vez, o que demanda um elevado tempo de processamento. Contudo, na versão paralela o cálculo das concentrações de radionuclídeos são realizados para diversos pontos da malha ao mesmo tempo, ou seja, as funções device CMIP e CMGN são executadas simultaneamente em múltiplos pontos da malha tridimensional ao mesmo tempo.

```
__global__ void Calculo_Concentracao_Media_Radionuclideos (<parâmetros>){
    // Declarações e Inicializações
    ...
    // Calculo das Concentrações:
    // Determinação da distância do ponto da malha ao centro de massa da
bufada
    for (IC = blockIdx.x * blockDim.x + threadIdx.x + I1[index + 1];
    IC <= I2[index + 1];
    IC += blockDim.x * gridDim.x) {
        X = IC * DX - DX * 0.5 - XP; XX = X * X;
        for (JC = blockIdx.y * blockDim.y + threadIdx.y + J1[index + 1];
        JC <= J2[index + 1];
        JC += blockDim.y * gridDim.y) {
           if (J1[index + 1] > NY) continue;
           if (J2[index + 1] < 1) continue;
           Y = JC * DY - DY * 0.5 - YP[index + 1];
           YY = Y * Y; AUX = (XX + YY) / DSIGH2[index + 1];
           T1 = Esp_Device(-AUX, A, B, C);
           // Calculo de Altura para Reflexão no Solo
           index_IHT = (JC)+(IC)*(NY + 1);
           KT = IHT[index_IHT];
           if ((KT + 1 < 0) || (KT + 1 > 10)) {
               KT = 0;
           }
           ZT = Altura[KT + 1];
           H = ZP[index + 1] - ZT;
           H2 = H * H;
           T2 = Esp_Device(-AUX - H2 / DSIGZ2[index + 1], A, B, C) * FATOR;
           // Concentração média de iodos e particulados no solo (ci / m3)
           Calculo_Concentracao_Media_Iodo_Particulados (<parâmetros>);
           // Concentração média de de gases nobres (ci / m3)
           Calculo_Concentracao_Media_Gases_Nobres (<parâmetros>];
        } // JC
    } // IC
    ...
}
```

Figura. 24 – Função kernel CCMR

```
__device__ void Calculo_Concentracao_Media_Iodo_Particulados (<parâmetros>) {
    // Declarações e Inicializações
    ...
    // Concentracao média de iodo e particulados no solo (ci / m3)
   for (NU = 0; NU < NUIP; NU++) {
       T3 = QPUF_ACIP[(NU + 1) + (NUIP*index)] / CONST1[index + 1];
       index_CONCIP = ((JC)+((NU + 1))*(NY + 1)) * (NX + 1) + (IC);
       CONCIP[index_CONCIP] = CONCIP[index_CONCIP] + T2 * T3;
   }
}
```

Figura. 25 – Função device CCMIP

```
__device__ void Calculo_Concentracao_Media_Gases_Nobres (<parâmetros>) {
    // Declarações e Inicializações
    ...
    // Concentração média de gases nobres (ci / m3)
    for (KC = 1; KC <= KCT; KC++) {
        ZR = ZT + ZZ - ZP[index + 1];
        ZR2 = ZR * ZR; AUX = powf((ZR + 2.0*H), 2);
        T4 = Esp_Device(<parâmetros>);
        T5 = Esp_Device(<parâmetros>);
        T6 = (T4 + T5) * T1 * FATOR;
        for (NU = 0; NU < NUGN; NU++) {
            T7 = QPUF_ANGCC[(NU + 1) + (NUGN*index)] / CONST2[index + 1];
            CONCGN[NUu][KC][JC][IC] = CONCGN[NUu][KC][JC][IC] + T7*T6;
            NUu = NUu + 1;
        } // NU
        NUu = 1; ZZ = ZZ + Hnz[KC];
    } // KC
}
```

Figura. 26 – Função device CCMGN

4. EXPERIMENTOS E RESULTADOS

Neste capitulo serão abordados os experimentos realizados na versão sequencial e paralela baseada em GPU do módulo Dispersão de Pluma e Dose, e os resultados gerados por estes experimentos também serão analisados.

Em relação aos experimentos, visando que os resultados obtidos fossem os mais próximos dos reais, as simulações foram realizadas usando as informações topográficas e meteorológicas da região nas vizinhanças da CNAAA, em Angra dos Reis. A Figura 2 mostra de forma mais precisa a região abrangida pelas simulações. Contudo, é importante salientar que as simulações que serão mostradas nesse capitulo foram realizas com base em apenas uma situação física, ou seja, para um caso de liberação e para um campo de vento. Outro aspecto relevante dos experimentos, é que as simulações foram configuradas para que as versões do módulo Dispersão de Pluma e Dose fossem executadas durante 20 ciclos.

Os seguintes tópicos são abordados nesta seção: validação do algoritmo sequencial e do paralelo; refinamento do domínio computacional; e tempos de execução e *speedups* dos algoritmos sequencial e paralelo.

4.1 VALIDAÇÃO DO ALGORITMO SEQUENCIAL E DO PARALELO

O programa original do sistema dispersão atmosférica de radionuclídeos (DAR) da CNAAA foi desenvolvido na década de 80 usando a linguagem FORTRAN, e algumas limitações foram integradas ao sistema para permitir sua execução nos computadores da época. Contudo, apesar dessas limitações, o sistema é utilizado até hoje na CNAAA e os resultados gerados por ele são validados pela Eletronuclear. Em virtude disso, um dos requisitos deste trabalho é que o algoritmo paralelo desenvolvido não seja apenas mais rápido, mas também seja capaz de alcançar resultados de saída semelhantes ao do programa original.

O resultado de saída do módulo Dispersão de Pluma e Dose é gerado na função Dosem, e este consiste em um arquivo contendo a distribuição espacial das taxas de doses na malha, calculada com base nos valores das concentrações médias de iodo, particulados e gases nobres estimados na função CCMR. O módulo produz a distribuição espacial para 3 tipos de doses: a taxa de dose para o pulmão e para a tireoide consideram a inalação de iodo, particulados e gases nobres para a estimativa; e taxa de dose para o corpo inteiro, que para estimativa considera uma imersão completa na nuvem radioativa, onde existe apenas a contribuição da concentração de gases nobres, usando o modelo de pluma finita. Outro ponto relevante, é que

para os 3 tipos de doses citadas, a função Dosem calcula a taxa média de dose para o ciclo atual do programa e também calcula a taxa média acumulada de doses, desde o primeiro ciclo do programa até o último.

Portanto, para validar os algoritmos desenvolvidos no trabalho, foram feitas simulações, nas quais a versão original (Fortran), a sequencial (C) e a paralela (CUDA) dos algoritmos do módulo Dispersão de Pluma e Dose foram executadas durante 20 ciclos e os resultados de saída foram comparados. Ao final das simulações foram gerados 3 arquivos de saída, com cada um contendo os valores das taxas de doses (pulmão, tireoide e corpo inteiro), estimados por cada versão do algoritmo. Sendo assim, para comparar os resultados foram utilizados dois métodos, o Erro Absoluto Médio (ou em inglês *Mean Absolute Error* (MAE)) e o Erro Absoluto Médio Percentual (*Mean Absolute Percentage Error* (MAPE)). Além disso, para comparar as saídas foi utilizado os arquivos que contêm os valores da taxa de dose acumulada dos 20 ciclos de execução, sendo o MAE e MAPE calculados para cada tipo de dose (corpo inteiro, pulmão e tireoide). As equações 13 e 14 mostram respectivamente as fórmulas do MAE e do MAPE.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |O_i - N_i| \tag{13}$$

$$\mathrm{MAPE} = \frac{100}{n}\sum_{i=1}^{n} \left|\frac{O_i - N_i}{O_i}\right| \tag{14}$$

Onde: n é o número total de doses na saída; O_ié o valor de dose original; e N_i é o novo valor de dose.

A Tabela 4 mostra o erro absoluto médio e o erro absoluto médio percentual para as taxas de doses acumuladas de corpo inteiro, pulmão e tireoide, geradas pelo algoritmo original (Fortran) e o sequencial (C). Da mesma forma, a Tabela 5 exibe os resultados MAE e MAPE das taxas de doses acumuladas produzidas pelo algoritmo original (Fortran) e pelo paralelo (CUDA).

Tabela 4 – Valor do erro absoluto médio e do erro absoluto médio percentual para a taxa de dose acumulada geradas pelo algoritmo original (Fortran) e o sequencial (C)

Tipo de Dose	**Erro Absoluto Médio (MAE)**	**Erro Absoluto Médio Percentual (MAPE)**
Corpo Inteiro	2,7620442E-07	0,0103797%
Pulmão	5,4667377E-05	0,0099493%
Tireoide	7,8711710E-03	0,0099486%

Tabela 5 – Valor do erro absoluto médio e do erro absoluto médio percentual para a taxa de dose acumulada geradas pelo algoritmo original (Fortran) e o paralelo (CUDA)

Tipo de Dose	**Erro Absoluto Médio (MAE)**	**Erro Absoluto Médio Percentual (MAPE)**
Corpo Inteiro	2,7620289E-07	0,0103798%
Pulmão	5,4667865E-05	0,0099496%
Tireóide	7,8707549E-03	0,0099483%

Levando em consideração os resultados mostrados na Tabela 4 e na Tabela 5, estes revelam que a diferença entre a distribuição espacial das taxas de doses (pulmão, tireoide e corpo inteiro), geradas pelo programa original e as produzidas pelos algoritmos sequencial e paralelo são consideradas pequenas e não afetam a tomada de decisão. De modo que, os algoritmos desenvolvidos neste trabalho foram aceitos em termos de reprodução dos cálculos originais.

4.2 REFINAMENTO DO DOMÍNIO COMPUTACIONAL

Como explicado no Capitulo 2, para que os sistemas DAR de centrais nucleares situadas em regiões urbanas com grandes variações em sua topologia, como é o caso da CNAAA, em Angra dos Reis, sejam capazes de prever a dispersão de radionuclídeos na área de interesse de forma precisa, é necessário que a resolução espacial das células na malha que representa a região seja menor que 100 m^2 para que as pequenas elevações no terreno sejam levadas em consideração pelo modelo e assim influenciem no padrão de dispersão dos radionuclídeos. O SDAR da CNAAA, possui atualmente uma resolução espacial 62.500 m^2, uma resolução muito distante da ideal.

Neste sentido, com objetivo de avaliar, com resoluções menores de malha, o comportamento dos algoritmos desenvolvidos e validados neste trabalho, foram realizadas simulações utilizando sucessivos níveis de refinamento do domínio computacional. Neste contexto, o processo de refinar significa reduzir, de acordo com o nível de refinamento, o tamanho das

áreas horizontais da célula original, contudo mantendo as áreas verticais. A Figura 27 mostra como essa redução foi feita, com a célula original sendo sucessivamente subdividida horizontalmente a cada aumento no nível de refinamento. A Tabela 6 também exibe o tamanho da célula, assim como a resolução espacial para cada nível de refinamento. Enquanto que a Tabela 7, apresenta o a configuração do domínio computacional para cada nível de refinamento, é possível perceber que com a redução do tamanho das células há um grande aumento no número total de células que precisam ser processadas pelo sistema.

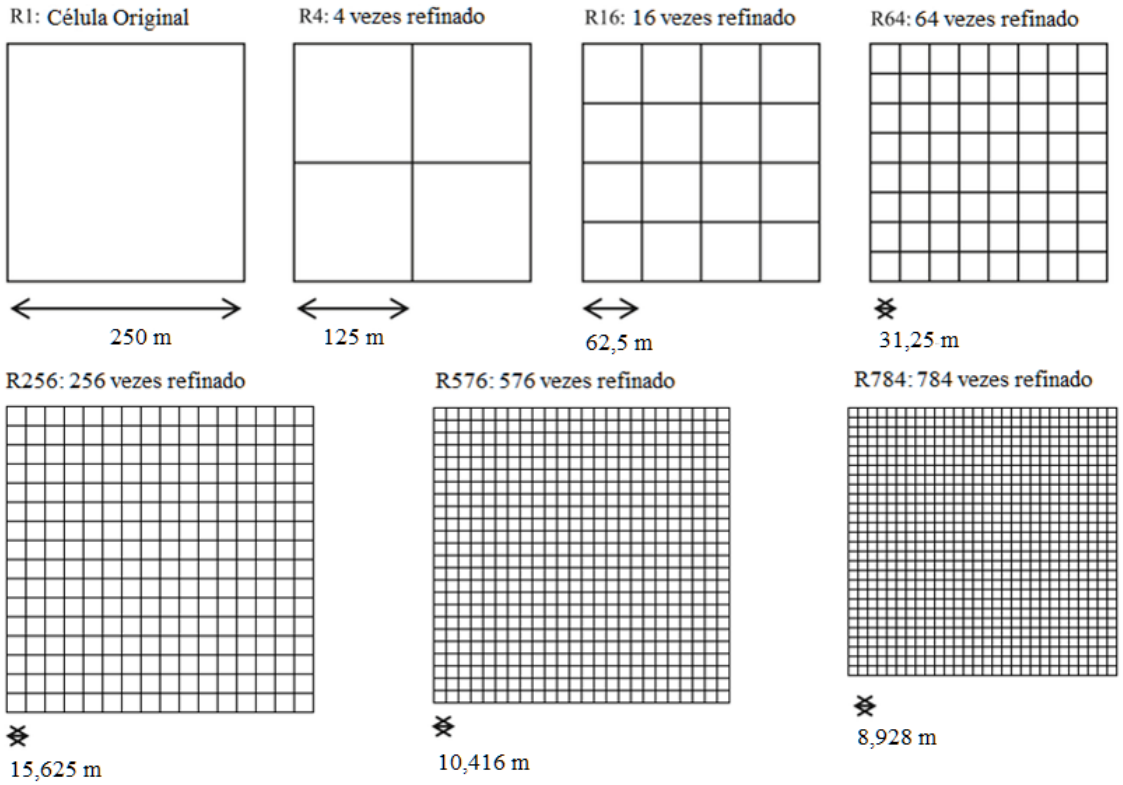

Figura. 27 – Dimensão das células para diferentes níveis de refinamento

Tabela 6 – Tamanho da resolução espacial para diferentes níveis de refinamento.

Nível de Refinamento	**Dimensões da Célula (metros)**	**Resolução Espacial (metros^2)**
R1	250 x 250	62.500
R4	125 x 125	15.625
R16	62,5 x 62,5	3.906,25
R64	31,25 x 31,25	976,56
R256	15,625 x 15,625	244,14
R576	10,416 x 10,416	108,49
R784	8,928 x 8,928	79,71

Tabela 7 – Domínio computacional para diferentes níveis de refinamento.

	Dimensão			
Nível de Refinamento	**X**	**Y**	**Z**	**N.º Células**
R1	67	43	8	23.048
R4	134	86	8	92.192
R16	268	172	8	368.768
R64	536	344	8	1.475.072
R256	1072	688	8	5.900.288
R576	1608	1032	8	13.275.648
R784	1876	1204	8	18.069.632

Nesta etapa, visando testar o comportamento dos algoritmos desenvolvidos, assim como contribuir para análise do SDAR como um todo, além dos níveis de refinamento (R1, R4, R16 e R64) investigados no módulo Campo de Vento no trabalho de (PINHEIRO, 2017), neste trabalho também foram analisados os níveis (R256, R576 e R784), com o objetivo de testar o comportamento do módulo Dispersão de Pluma e Dose com resoluções espaciais próximas ou inferiores aos 100 m^2, e servir como para parâmetro para testes em outros módulos. Entretanto, como também é citado em (PINHEIRO, 2017), o melhor nível de refinamento só será definido quando o SDAR da CNAAA for analisado por completo, utilizando as informações: deste trabalho do módulo Dispersão de Pluma e Dose, o do módulo Campo Vento e o do módulo Projeção que ainda vai ser desenvolvido futuramente.

4.3 ANÁLISE DOS RESULTADOS

Nesta seção, serão analisados os resultados dos experimentos realizados no algoritmo sequencial e no paralelo. Como explicado no capítulo anterior, ambos os algoritmos foram submetidos a diversas simulações, utilizando vários níveis de refinamento (R1, R4, R64 R256 R576, e R784), isto foi feito com intuito de avaliar a performance da versão paralela, afim de estimar e avaliar os ganhos obtidos com implementação do programa paralelo baseado em GPU.

Sendo assim, primeiramente a Tabela 8 mostra os resultados comparativos entre os tempos de execução (média de 5 execuções) para o algoritmo sequencial (usando CPU Intel-I5 7500 @ 3,40 GHz - 3,80 GHz) e o algoritmo paralelo (usando GPU GTX-1070) da função CCMR. Enquanto que, utilizando a mesma configuração de *hardware*, a Tabela 9 mostra os resultados comparativos entre os tempos de execução (média de 5 execuções) para o algoritmo sequencial e o algoritmo paralelo da função Tradif.

Analisado as tabelas 8 e 9, é possível observar que o tempo de execução aumenta significativamente de acordo com o nível de refinamento do domínio computacional, contudo, isso já era esperado, pois como foi mostrado na Tabela 7, aumentar o nível de refinamento implica em um aumento substancial no número de células que precisam ser executadas no programa.

No entanto, o aumento no tempo de processamento dos algoritmos é significativamente distinto. Enquanto nos níveis mais elevados de refinamento (R576 e R784) o tempo de execução do algoritmo sequencial chega a superar 30 minutos, no algoritmo paralelo o tempo de processamento fica na faixa do 1 minuto. A Figura 28 e a Figura 29 ilustram o comportamento de ambos os algoritmos, em relação ao crescimento no tempo de execução de acordo com o nível de refinamento.

Os valores de *speedup*, quantificam os ganhos da versão paralela em relação a versão sequencial. Contudo, analisando a Fígura 30 e a Figura 31 é possível perceber que em níveis mais elevados de refinamento (R16, R64, R256, R576 e R784) os valores de *speedup* crescem substancialmente. E a explicação para esse comportamento é o fato que em domínios computacionais mais altos, o tempo de processamento na GPU é maior. Enquanto em domínios menores, como o tempo que a GPU passa realizando os cálculos matemáticas é relativamente curto, as operações como: gestão de threads; transferência de dados entre CPU e GPU; e alocação de memória na GPU, possuem um impacto maior no tempo total de execução.

Tabela 8 – *Speedup* e tempos de execução do algoritmo sequencial (CPU) e paralelo (GPU) da função CCMR para diferentes níveis de refinamento.

Nível de Refinamento	**TCPU(s)**	***TGPU(s)**	**Speedup**
R1	1,52	1,70	0,89
R4	5,90	1,78	3,31
R16	23,01	2,34	9,83
R64	107,77	5,61	19,21
R256	687,62	20,99	32,75
R576	1759,43	43,30	40,63
R784	2492,54	59,53	41,87

*Considerando: função kernel GPU + alocação de memória na GPU + transferência de dados para a GPU.

Tabela 9 – *Speedup* e tempos de execução do algoritmo sequencial (CPU) e paralelo (GPU) da função Tradif para diferentes níveis de refinamento.

Nível de Refinamento	**TCPU(s)**	***TGPU(s)**	**Speedup**
R1	1,79	3,69	0,48
R4	6,37	3,59	1,77
R16	23,83	4,35	5,47
R64	108,93	8,44	12,90
R256	690,62	25,51	27,07
R576	1763,93	49,39	35,71
R784	2498,59	67,91	36,79

*Considerando: função *kernel* GPU + alocação de memória na GPU + transferência de dados para a GPU

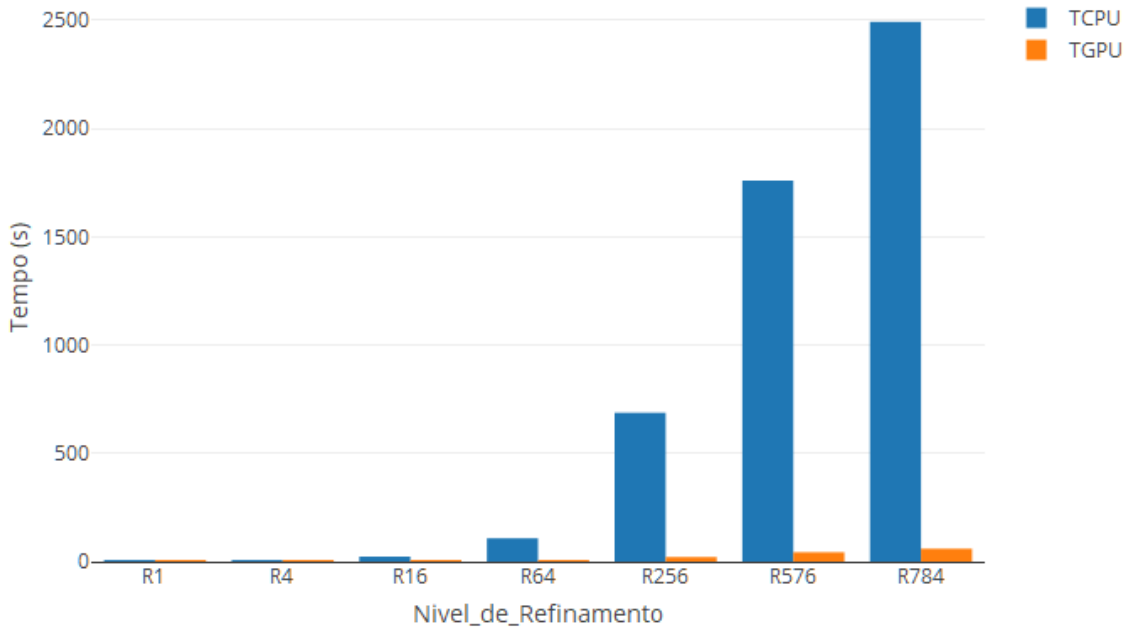

Figura. 28 – Tempo de execução da função CCMR em relação ao nível de refinamento

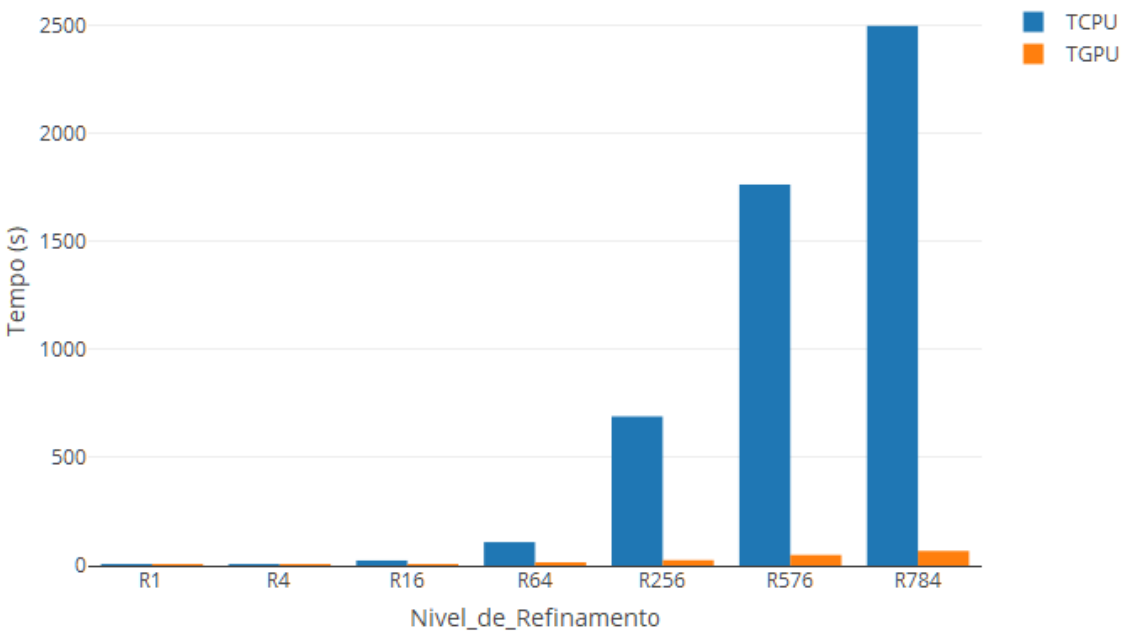

Figura. 29 – Tempo de execução da função Tradif em relação ao nível de refinamento

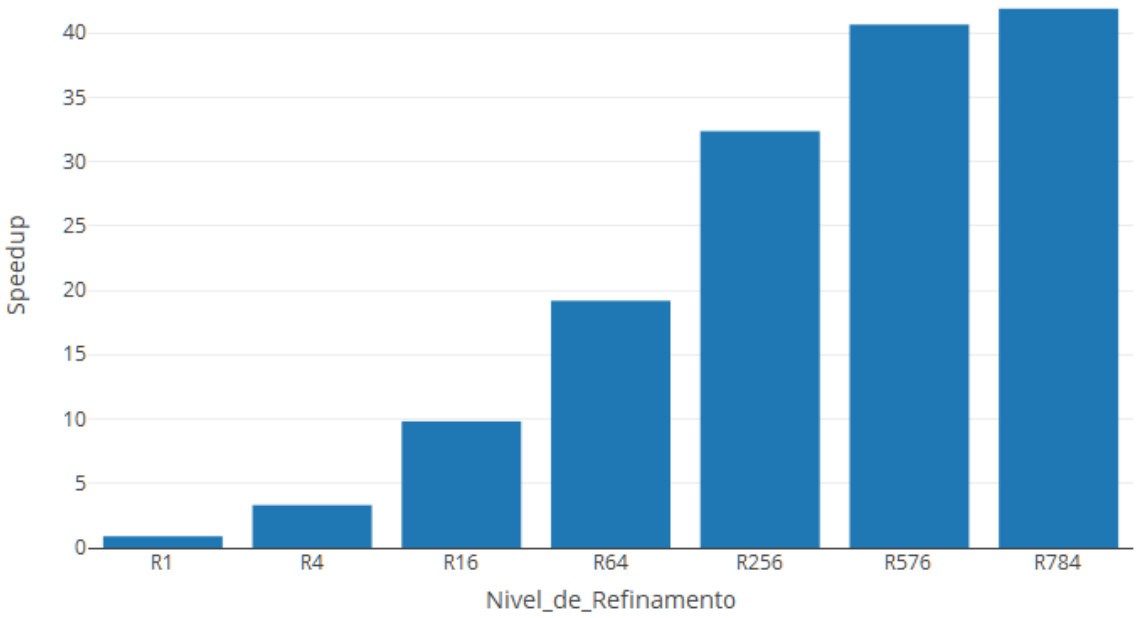

Figura. 30 – *Speedups* da função CCMR para diferentes níveis de refinamento

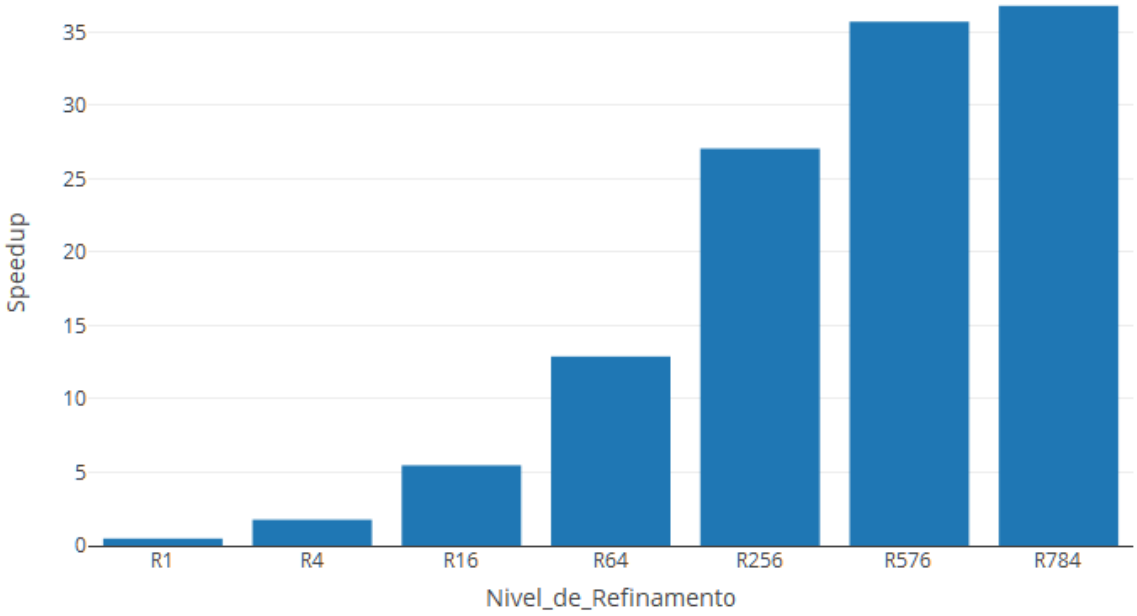

Figura. 31 – *Speedups* da função Tradif para diferentes níveis de refinamento

5. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi proposta a utilização da computação paralela, baseada em GPU, para desenvolver uma versão do módulo Dispersão de Pluma e Dose do sistema de dispersão atmosférica de radionuclídeos (SDAR) da CNAAA, capaz de trabalhar com um domínio computacional refinado. Uma vez que, a versão original do módulo, assim como o SDAR da CNAAA como um todo, não é capaz de operar com o nível de refinamento desejável, pois o tempo de processamento aumenta de tal forma que o sistema não consegue ser executado em tempo hábil.

Posto isto, tendo em vista que o módulo Dispersão de Pluma e Dose é composto por duas partes principais, uma para calcular o transporte e difusão atmosférica dos radionuclídeos e outra para estimar a taxa média de dose. Neste trabalho, se optou por focar no aspecto de transporte e difusão, já que este representa 95,05% do tempo total de execução do módulo.

Durante as investigações, antes do trabalho de paralelização ser propriamente iniciado, foi elaborada uma versão sequencial completa do algoritmo do módulo Dispersão de Pluma e Dose, utilizando a linguagem C, já que o algoritmo original foi desenvolvido na década de 80, usando a linguagem FORTRAN, e possuía características que dificultariam o trabalho de paralelização. A versão paralela do módulo, foi elaborada utilizando a linguagem CUDA/C em conjunto com a técnica de paralelização grid-stride loop. As duas versões desenvolvidas no trabalho foram validadas, gerando resultados de saída (que é uma tabela contendo a distribuição espacial das doses) muito próximas das geradas pelo algoritmo original.

Após ser verificada e testada a consistência dos algoritmos desenvolvidos, foram realizadas simulações computacionais, utilizando 7 níveis de refinamento (R1, R4, R16, R64, R256, R576, R784). Sendo que para cada uma destas simulações, ambas as versões (sequencial C e paralela CUDA/C) do módulo foram executadas durante 20 ciclos.

Avaliando os resultados das simulações foi possível detectar, assim como era esperado, que o tempo de processamento aumenta significativamente de acordo com o nível de refinamento do domínio computacional. Já que elevar o nível de refinamento também significa aumentar de forma substancial o número de células que são processadas no módulo. Contudo, observou-se que o tempo de execução dos algoritmos (sequencial e paralelo) crescem de maneira distinta, acontecendo um aumento no *speedup* à medida que o domínio computacional aumenta. De modo que, no nível de refinamento original (R1), o *speedup* é mínimo, mas quando o domínio computacional cresce, a aceleração atinge valores mais

expressivos. Sendo que o maior *speedup* alcançado foi com o nível de refinamento R784. Nesse nível de refinamento, o tempo de execução da função CCMR caiu de 2492,54s (versão sequencial, executada em um Intel-Core I5 7500 PC) para 59,53s (versão paralela, executada em uma GTX-1070), atingindo uma aceleração de 41,86 vezes. Enquanto, que para a função Tradif, usando a mesma configuração de *hardware*, o tempo de processamento foi reduzido de 2498,59s (na versão sequencial) para 67,91s (na versão paralela), alcançando o *speedup* de 36,79.

Os resultados obtidos demonstram que a utilização computação paralela baseada em GPU permite que o módulo Dispersão de Pluma e Dose opere com níveis de refinamento computacional mais elevados. Principalmente, o resultado alcançado utilizando o refinamento R784, que mostrou que é possível trabalhar com uma malha com resolução espacial de 79,71 m^2, valor que está dentro da faixa indicada por (PEDERSEN, LEACH e HANSEN, 2007), que relata que para regiões como Angra dos Reis, onde a CNAAA está localizada, com um alto número de variações em sua topologia, é recomendado a utilização de uma resolução espacial menor que 100 m^2, para que as pequenas alterações do terreno sejam levadas em consideração pelo modelo na previsão da movimentação da pluma radioativa.

Sendo assim, espera-se que os programas desenvolvidos usando computação paralela baseada em GPU, do módulo Dispersão de Pluma de Dose deste trabalho e o do módulo Campo de Vento de (PINHEIRO, 2017), sejam implementados futuramente no SDAR da CNAAA, de forma a permitir que o sistema trabalhe um domínio computacional refinado e seja capaz de funcionar em tempo real. A fim de tornar o sistema mais preciso, refletindo melhor o cenário real e dando apoio a tomada de decisões.

Em relação a trabalhos futuros, são sugeridos os seguintes tópicos:

- A utilização do algoritmo de otimização por enxame de partículas – PSO para otimizar o processo de alocação de threads;
- O uso de técnicas de otimização de acesso a memória da GPU para reduzir a sobrecarga de transferência de dados;
- Integrar os módulos paralelos de dispersão e campo de vento no sistema de dispersão de radionuclídeos da CNAAA e avaliar o sistema como um todo;
- Usar um cluster de GPUs para executar o módulo de dispersão, assim como os demais módulos já paralelizados do SDAR da CNAAA.

REFERÊNCIAS

AUSTRALIAN GOVERNMENT. Evaluation of ARGOS for use in Australia. In: GRZECHNIK, M.; TINKER, R.; SOLOMON, S. **Technical Report No. 150**. [S.l.]: [s.n.], 2008.

ALMEIDA, A. A. H. **Desenvolvimento de Algoritmos Paralelos Baseados em GPU para Solução de Problemas na Área Nuclear**. Rio de Janeiro: Dissertação (Mestrado em Engenharia de Reatores) – IEN, 2009.

BROWN, M. J. **Urban Dispersion – Challenges for Fast Response Modeling**. [s.d.]. Los Alamos National Laboratory. Los Alamos, New Mexico.

BUCK, I. **GPU computing with nVIDIA CUDA**. In: International. Conference on Computer Graphics and Interactive Techniques. ACM New York, NY, USA: [s.n.]. 2007.

DALY, A.; ZANNETTI, P. **Air Pollution Modeling – An Overview**. Ambient Air Pollution. Published by The Arab School for Science and Technology (ASST) and The EnviroComp Institute. p.18. 2007. Disponível em: <http://home.iitk.ac.in/~anubha/Modeling.pdf >.

EISENBUD, M. **Environmental Radioactivity An Interdisciplinary Monograph**, 2nd, Academic Press, N. Y.:, 1973.

ELETRONUCLEAR. **Informações sobre o plano de emergência das usinas nucleares de Angra dos Reis**. 2011. Disponível em: <http://www.eletronuclear.gov.br/Not%C3%ADcias/NoticiaDetalhes/tabid/191/NoticiaID/322/Default.aspx>. Acessado em: 02/03/2018.

FLYNN, M. J. **Some computer organizations and their effectiveness**. [S.l.]: IEEE Transactions on Computers, v. 24, n. 9, 1972.

FURNAS CENTRAIS ELÉTRICAS S.A. **Especificação Funcional do Sistema de Controle Ambiental (SCA)**. Rio de Janeiro: [s.n.], 1987.

CALIFORNIA AIR RESOURCES BOARD AND THE CALIFORNIA ENERGY RESOURCES CONSERVATION AND DEVELOPMENT COMMISSION. Point Source Model Evaluation and Development Study – The Grid Model IMPACT (Integrated Model for Plumes and Atmospherics in Complex Terrain). FABRICK, A.; SKLAREW, R.; WILSON, J. **Technical report (Contract A5-058-87)**. [S.l.]: [s.n.], 1987.

COELHO, M. J.; RIBEIRO, L. A. A. Dispersão atmosférica em sítios de usinas nucleares. **Revista Militar de Ciência e Tecnologia, Vol. XXV**, p. 45-51. 2008.

COPPE/UFRJ - NUCLEAR, LABORATÓRIO DE ANÁLISE E SEGURANÇA. **SCA-MOD - Sistema de Controle Ambiental - Modelagem**. Rio de Janeiro: [s.n.], v. 3, 1987.

HALL, D. J. SPANTON, A. M. et al. The UDM: A Puff Model for Estimating Dispersion in Urban Areas. **7th Int. Conf. on Harmonisation within Atmospheric Dispersion in Urban Areas**, p. 256-260.

HARRIS, M. **An Even Easier Introduction to CUDA**. NVIDIA Developer Blog, 2017. Disponível em: < https://devblogs.nvidia.com/even-easier-introduction-cuda/>. Acessado em: 02/03/2018.

HARRIS, M. **CUDA Pro Tip: Write Flexible Kernels with Grid-Stride Loops**. NVIDIA Developer Blog, 2013. Disponível em: < https://devblogs.nvidia.com/cuda-pro-tip-write-flexible-kernels-grid-stride-loops/>. Acessado em: 02/03/2018.

HEIMLICH, A.; MOL, A. C. A.; PEREIRA, C. M. N. A. GPU-based Monte Carlo simulation in neutron transport and finite differences heat equation evaluation. **Progress in Nuclear Energy, 53 (2)**, p. 229-239. 2011.

HEIMLICH, A.; SILVA, F. C.; MARTINEZ, A. S. Parallel GPU implementation of PWR reactor burnup. **Annals of Nuclear Energy 91 (Supplement C)**, p. 135-141. 2016.

HWU, W. -M. W.; KIRK, D. B. **Programming Massively Parallel Processors: A Hands-on Approach**. 2. ed. Elsevier, 2012.

INTERNATIONAL ATOMIC ENERGY AGENCY. **Eletricity Information: Overview (2017 edition)**. p. 3. 2017.

INTERNATIONAL ATOMIC ENERGY AGENCY. **Atmospheric Dispersion Models for Application in Relation to Radionuclide Releases**. Vienna. p. 7. 1986. (Tech doc-379).

LABORATÓRIO DE METEROLOGIA E QUALIDADE DO AR. **Modelos de Dispersão de Contaminantes na CLP**. UFGS. [s.d.]. Disponível em: <http://www.ufrgs.br/lmqa/arquivos/uploads/aula_modelos_dispersao%20[Modo%20de%20Compatibilidade].pdf>. Acessado em: 02/03/2018.

MELAZO, C. D. **Emissão de C^{14} pelas Unidades I e II da Central Nuclear Almirante Álvaro Alberto (CNAAA) e seu Efeito Local nos Níveis Ambientais**. Brasília: Tese (Doutorado em Geociências) - Universidade de Brasília, 2006.

MELO, A. M. V. **Avaliação de Desempenho dos Modelos AERMOD e CALPUFF Associados ao Modelo PRIME**. Vitória: Dissertação (Mestrado em Engenharia Ambiental) - Universidade Federal do Espírito Santo, 2011.

GELYBÓ, G.; LAGZI, I.; LEELŐSSY, A.; MÉSZÁROS, R. **Atmospheric Chemistry**. Eötvös Loránd University, 2013. Disponível em: <http://elte.prompt.hu/sites/default/files/tananyagok/AtmosphericChemistry/index.html>. Acessado em: 02/03/2018.

MULTIPHYSICS CYCLOPEDIA. **High-Performance Computing (HPC)**. [s.d.]. Disponível em: <https://br.comsol.com/multiphysics/high-performance-computing>. Acessado em: 02/03/2018.

NAKAMURA, R. H. **Desenvolvimento de Uma Aplicação Distribuída em Um Ambiente Corporativo Utilizando PVM**. Brasília: Monografia (Graduação em Engenharia da Computação) – Centro Universitário de Brasília, 2005.

NAVARRO, C. A.; -KAHLER, N. H; MATEU, L. A survey on parallel computing and its applications in data-parallel problems using GPU architectures. **Communications in Computational Physics, 15 (2),** p. 285-329. 2014.

NVIDIA. **GeForce GTX 1080 Whitepaper**. 2016.

PARK, S.-U.; CHOE, A.; PARK, M.-S. Atmospheric Dispersion and Deposition of Radionuclide (137Cs and 131I) Released from Fukushima Dai-ichi Nuclear Power Plant. **Computational Water, Energy and Environmental Engineering, 2**, p. 61-68. 2013.

PEDERSEN, U.; LEACH, S.; HANSEN, J.-E. S. **Biological Incident Response: Assessment of Airborne Dispersion**. [S.l.]. 2007.

PEREIRA, C. M. N. A.; HEIMLICH, A.; MOL, A. C. A.; MORAES, S. R. S.; RESENDE. P. Development and performance analysis of a parallel Monte Carlo neutron transport simulation program for GPU cluster using MPI and CUDA technologies. **Progress in Nuclear Energy, 65 (Supplement C)**, p. 88-94. 2013

PINHEIRO, A. L. S. **Modelo Computacional Paralelo Baseado em GPU para Cálculo do Campo de Vento de um Sistema de Dispersão Atmosférica de Radionuclídeo**s. Rio de Janeiro: Tese (Doutorado em Engenharia Nuclear) - Universidade Federal do Rio de Janeiro, 2017.

PINHEIRO, A.; DESTERRO, F.; SANTOS, M. C.; PEREIRA, C. M. N. A.; SCHIRRU, R. GPU-based implementation of a diagnostic wind field model used in real-time prediction of atmospheric dispersion of radionuclides. **Progress in Nuclear Energy, 100 (Supplement C)**, p. 146-163, 2017.

SOARES, M. S. **Estratégias de Simulação da Dispersão de Poluentes com os Modelos WRF/CALMET/CALPUFF para a Região Metropolitana do Rio De Janeiro**. Rio de Janeiro: Dissertação (Mestrado em Engenharia Mecânica) - Universidade Federal do Rio de Janeiro, 2012.

VELLOSO, M. F. A. **Avaliação de Modelos Gaussianos para Fins Regulatórios – Um Estudo para a Bacia Aérea III da Região Metropolitana do Rio De Janeiro**. Rio de Janeiro: Dissertação (Mestrado em Engenharia Mecânica) - Universidade Federal do Rio de Janeiro, 2007.