IMPLEMENTAÇÃO DA COMPUTAÇÃO MASSIVAMENTE PARALELA NA SIMULAÇÃO MONTE CARLO EM TRANSPORTE DE RADIAÇÃO IONIZANTE

RICARDO CESAR VASCONCELOS LOUREIRO

SALVADOR

2024

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA BAHIA
CAMPUS SALVADOR

RICARDO CESAR VASCONCELOS LOUREIRO

IMPLEMENTAÇÃO DA COMPUTAÇÃO MASSIVAMENTE PARALELA NA SIMULAÇÃO MONTE CARLO EM TRANSPORTE DE RADIAÇÃO IONIZANTE

SALVADOR
2024

RICARDO CESAR VASCONCELOS LOUREIRO

IMPLEMENTAÇÃO DA COMPUTAÇÃO MASSIVAMENTE PARALELA NA SIMULAÇÃO MONTE CARLO EM TRANSPORTE DE RADIAÇÃO IONIZANTE

Relatório do Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia da Bahia, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas e Produtos, para a obtenção do título de Mestre.

Orientador: Wilson Otto Gomes Batista

Coorientador: Antônio Carlos dos Santos Souza

SALVADOR

2024

Sistema de Bibliotecas - IFBA

L892i Loureiro, Ricardo Cesar Vasconcelos

Implementação da computação massivamente paralela na simulação monte carlo em transporte de radiação ionizante / Ricardo Cesar Vasconcelos Loureiro; orientador Wilson Otto Gomes Batista; coorientador Antônio Carlos dos Santos Souza -- Salvador, 2024.

62 p.

Relatório do Trabalho de Conclusão de Curso (Programa de Pós-Graduação em Engenharia de Sistemas e Produtos) -- Instituto Federal da Bahia, 2024.

1. Radiação ionizante. 2. Método monte carlo. 3. Computação paralela. 4. Código penelope. I. Batista, Wilson Otto Gomes, orient. II. Souza, Antônio Carlos dos Santos, coorient. III. TÍTULO.

CDU 544.543.7

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Bahia

PROGRAMA DE
PÓS-GRADUAÇÃO EM
ENGENHARIA DE SISTEMAS E PRODUTOS

INSTITUTO FEDERAL DA BAHIA

PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE SISTEMAS E PRODUTOS – PPGESP

IMPLEMENTAÇÃO DA COMPUTAÇÃO MASSIVAMENTE PARALELA NA SIMULAÇÃO MONTE CARLO EM TRANSPORTE DE RADIAÇÃO IONIZANTE

RICARDO CÉSAR VASCONCELOS LOUREIRO

Produto(s) Gerado(s): Relatório e Registro de Programa de Computador;

Orientador: Prof. Dr. Wilson Otto Gomes Batista
Coorientador: Prof. Dr. Antônio Carlos dos Santos Souza

Banca examinadora:

_____

Prof. Dr. Wilson Otto Gomes Batista
Orientador – Instituto Federal da Bahia (IFBA)

_____

Prof. Dr. - Antônio Carlos dos Santos Souza
Coorientador – Instituto Federal da Bahia (IFBA)

_____

Prof. Dr. Luiz Claudio Machado dos Santos
Membro Externo - Instituto Federal da Bahia (IFBA)

_____

Profa Dra. Maria Rosangela Soares
Membro Externo - Universidade Federal de Rondônia (UNIR)

Trabalho de Conclusão de Curso aprovado pela banca examinadora em 23/08//2024.

Documento assinado eletronicamente por **Maria Rosangela Soares**, **Usuário Externo**, em 23/08/2024, às 19:42, conforme decreto nº 8.539/2015.

Documento assinado eletronicamente por **WILSON OTTO GOMES BATISTA**, **Docente Permanente**, em 23/08/2024, às 19:42, conforme decreto nº 8.539/2015.

Documento assinado eletronicamente por **LUIZ CLAUDIO MACHADO DOS SANTOS**, **Professor do Ensino Básico, Técnico e Tecnológico do Câmpus Salvador**, em 23/08/2024, às 21:18, conforme decreto nº 8.539/2015.

Documento assinado eletronicamente por **ANTONIO CARLOS DOS SANTOS SOUZA**, **Docente Permanente**, em 24/08/2024, às 09:21, conforme decreto nº 8.539/2015.

A autenticidade do documento pode ser conferida no site http://sei.ifba.edu.br/sei/controlador_externo.php?acao=documento_conferir&acao_origem=documento_conferir&id_orgao_acesso_externo=0 informando o código verificador **3665741** e o código CRC **B2560C11**.

DEDICATÓRIA

Dedico este trabalho a todos os que me auxiliaram nessa longa jornada, em especial, a minha esposa e parceira de vida Camila Loureiro.

AGRADECIMENTOS

A todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho, em especial a Vagner da Silva e aos meus mentores Wilson Otto e Antônio Carlos.

“Não podemos resolver nossos problemas com o mesmo pensamento que usamos quando os criamos.” (Albert Einstein)

RESUMO

O Código utilizado é o PENELOPE (Penetration and ENErgy LOss of Positrons and Electrons) que é um código computacional baseado no MMC, criado com o objetivo de auxiliar nas simulações de sistemas de transporte de partículas neutras e carregadas utilizando métodos determinísticos e estocásticos. Dentre os problemas que hoje essa área enfrenta, um dos principais, é o tempo de espera até o final da simulação, que em determinado casos, podem durar dias, inviabilizando assim que o pesquisador possa utilizar o seu computador pessoal para efetuar as simulações e também a dificuldade de configuração em um arquivo de texto que exige tabulações específicas para que a simulação ocorra.

O Software de Paralelização do Penelope (SOPPE) é uma solução baseada no uso da arquitetura massivamente paralela com o uso da GPU (Unidades de Processamento Gráfico), o que resultou em incríveis ganhos de velocidade de até 145 vezes, a configuração dessas simulações também foi aprimorada devido ao desenvolvimento de uma interface amigável, especificamente projetada para esse propósito. As simulações foram realizadas utilizando apenas o computador pessoal do pesquisador, juntamente com uma placa de vídeo para paralelização, contribuindo significativamente para o avanço dos estudos nessa área.

Palavras-chave: Radiação Ionizante, Método Monte Carlo, Computação Paralela, Código PENELOPE.

ABSTRACT

The code used is PENELOPE (Penetration and ENErgy LOss of Positrons and Electrons), which is a computer code based on MMC, created with the objective of assisting in the simulations of transport systems of neutral and charged particles using deterministic and stochastic methods. Among the problems that this area faces today, one of the main ones is the waiting time until the end of the simulation, which in certain cases can last days, thus making it impossible for the researcher to use his personal computer to carry out the simulations, and also the difficulty of configuring a text file that requires specific tabs for the simulation to occur.

The Penelope Parallelization Software (SOPPE) is a solution based on the use of massively parallel architecture with the use of the GPU (Graphics Processing Units), which resulted in incredible speed gains of up to 145 times. The configuration of these simulations was also improved due to the development of a user-friendly interface, specifically designed for this purpose. The simulations were carried out using only the researcher's personal computer, together with a video card for parallelization, contributing significantly to the advancement of studies in this area.

Keywords: Ionizing Radiation, Monte Carlo Method, Parallel Computing, PENELOPE Code.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

1 INTRODUÇÃO

A necessidade da utilização e entendimento de técnicas que utilizam transporte de radiações ionizantes é seguida da carência de avaliar os valores de dose absorvida decorrentes destas técnicas. Dentre os métodos utilizados para efetuar esta avaliação, destaca-se o método de Monte Carlo (MMC), sendo esse método utilizado na maioria das aplicações científicas que envolvem simulações de transporte de radiação, como por exemplo, dosimetria interna, proteção radiológica, radiologia e radioterapia etc [1].

O MMC é uma abordagem de simulação que usa números aleatórios para modelar como a radiação se move na matéria, criando eventos aleatórios chamados de "histórias" para acompanhar a trajetória de uma partícula desde sua origem até seu fim. Essa simulação com o MMC possibilita a obtenção de resultados detalhados, confiáveis e sem a necessidade de procedimentos experimentais trabalhosos e custosos.

Contudo, essas simulações de transporte de radiação em geometrias complexas como a anatomia humana, por exemplo, implicam em um elevado custo computacional para obtenção de resultados com incerteza estatística nos níveis aceitáveis, limitando assim o seu uso. Nas geometrias complexas, as quais adicionam e/ou potencializam as interações, o custo computacional se amplia significativamente [2, 3]. Nesse contexto, a computação massivamente paralela se apresenta como alternativa viável.

Existem vários códigos abertos disponíveis para esse propósito tais como: PENELOPE (Penetration and ENErgy LOss of Positrons and Electrons); EGS (Electron Gamma Shower), Geant4(GEometry ANd Tracking) e PenRed (PenRed (Parallel Engine for Radiation Energy Deposition)) que utilizam o MMC para diversas simulações e são baseados nos modelos de espalhamento e absorção que combinam uma base numérica de dados com a seção de choque para os diferentes mecanismos de interação [4].

O código PENELOPE permite apenas a simulação de fótons, elétrons e pósitrons. Os demais códigos citados permitem ampliar o leque de partículas incluindo por exemplo prótons e nêutrons [1].

A utilização do MMC gera benefícios, permitindo, por exemplo, que a avaliação da dose absorvida em tecidos e órgãos submetidos a um exame radiológico seja simulada com alta precisão, mas no geral exige um alto custo computacional e demanda muito tempo [4].

A presente proposta visa apresentar uma solução baseada no uso da arquitetura massivamente paralela que permita a execução de simulação com o MMC aplicada em

transporte de radiação com redução de tempo de processamento, com uma interface amigável e utilização do computador pessoal, evitando assim a necessidade de máquinas de grande porte ou "clusters" com dezenas de máquinas e que possa ser aplicada no código PENELOPE [1].

2 RELATÓRIO DESCRITIVO

IMPLEMENTAÇÃO DA COMPUTAÇÃO MASSIVAMENTE PARALELA NA SIMULAÇÃO MONTE CARLO EM TRANSPORTE DE RADIAÇÃO IONIZANTE

Campo de aplicação

O Software de Paralelização do Penelope (SOPPE) é uma aplicação para a área de física médica, tem como principal campo de aplicação o uso para a pesquisa nesta área, utilizado por pesquisadores e alunos que precisam efetuar simulações em transporte de radiação ionizante, visando assim, auxiliar a configuração, acelerar o resultado e diminuir o custo computacional.

Motivação e Objetivos

O método de Monte Carlo é utilizado na maioria das aplicações científicas que envolvem simulações em transporte de radiação. A sua utilização gera benefícios, permitindo, por exemplo, que a avaliação da dose absorvida em tecidos e órgãos submetidos a um exame radiológico ou tratamentos, seja simulada com alta precisão, mas no geral exige um alto custo computacional e demanda muito tempo [1].

Estado da técnica

A simulação com o Método de Monte Carlo possibilita a obtenção de resultados detalhados, confiáveis e sem a necessidade de procedimentos experimentais trabalhosos e custosos. Contudo, essas simulações de transporte de radiação em geometrias complexas como a anatomia humana, implicam em um elevado custo computacional para obtenção de resultados com incerteza estatística nos níveis aceitáveis, limitando assim o seu uso. Nesse contexto, a computação massivamente paralela se apresenta como alternativa viável [2, 3].

Existem vários códigos abertos disponíveis para efetuar essas simulações utilizando o Método de Monte Carlos (MMC) para diversas simulações e são baseados nos modelos de espalhamento e absorção que combinam uma base numérica de dados com a seção de choque para os diferentes mecanismos de interação [4].

Alguns dos códigos mais utilizados são:

1. PENELOPE que apresenta uma geometria mais simplificada para simulações em materiais planos, esféricos, cilíndricos entre outros, utilizando o Método Monte Carlo no transporte de fótons, pósitrons e elétrons.
2. Electron Gamma Shower (EGS) que é mais de uso mais geral, para o transporte de elétrons e fótons, é um código bastante utilizado na área da física médica [5].
3. Geometry and Tracking (Geant4) que simula o transporte através da matéria, permitindo a utilização de processos físicos, eletromagnéticos e ópticos, por isso é um código utilizado nas áreas da física, médica, de partículas, nuclear entre outras [6].
4. Parallel Engine for Radiation Energy Deposition (PenRed) é baseado no PENELOPE utilizando o paralelismo no transporte de elétrons e fótons através da matéria [7].

Problemas do estado da técnica

Alto custo computacional para gerar as simulações, a alta demanda de tempo para gerar as simulações, a complexidade na configuração e a impossibilidade da utilização do computador pessoal do pesquisador devido a necessidade de uma máquina com mais poder computacional.

Vantagens da proposta

O projeto visa apresentar uma solução baseada no uso da arquitetura massivamente paralela que permita a execução da simulação aplicada em transporte de radiação com redução de tempo de processamento, interface amigável e a utilização do computador pessoal. O SOPPE utiliza como base o PENELOPE, um dos códigos de simulação Monte Carlos mais utilizados pelos pesquisadores.

Descrição e Método detalhado do Sistema/Produto/Processo Proposto

Diferente das Unidades de Processamento Central (CPUs), que foram projetadas para gerenciar tarefas de forma sequencial, as Unidades de Processamento Gráfico (GPUs) são otimizadas para executar muitas operações em paralelo, o que torna a GPU extremamente vantajosa em aplicações como processamento de imagens, aprendizado de máquina e simulações físicas.

Desenvolver para GPUs fornece um ganho muito maior do que desenvolver para CPUs, pois as GPUs oferecem duas a três ordens de magnitude mais paralelismo de *threads* em comparação com as CPUs. [8, 9]

As GPUs são projetadas com uma arquitetura altamente paralela, permitindo que milhares de *threads* sejam executadas simultaneamente. Em contraste, as CPUs possuem um número limitado de núcleos otimizados para tarefas sequenciais, enquanto as GPUs possuem centenas ou milhares de núcleos menores, otimizados para executar múltiplas operações em paralelo. Este paralelismo massivo pode resultar em melhorias significativas no desempenho e na eficiência energética, reduzindo o tempo necessário para realizar tarefas complexas. Devido a isso, o uso da GPU quando comparado com a CPU oferece inúmeros benefícios, especialmente em aplicações que demandam alto desempenho [9].

Essa arquitetura paralela das GPUs é o que permite essa execução simultânea de milhares de *threads*, cada uma pode executar a mesma instrução em diferentes conjuntos de dados, isso é particularmente útil para tarefas que envolvem grandes volumes de dados que podem ser processados de forma independente [10].

Já a Arquitetura de Dispositivo Unificado de Computação (CUDA) é um tipo de arquitetura desenvolvida pela NVIDIA, que permite que a GPU execute cálculos mais complexos paralelamente, além disso essa arquitetura disponha de uma interface mais amigável, que facilita o desenvolvimento do código permitindo que se ignore a API gráfica da GPU, podendo programar em C, C++, Fortran, entre outras linguagens para o código rodar nas GPUs [10].

Um programa CUDA é estruturado em duas partes principais: o programa do *host* e os *kernels* paralelos. O *host* consiste em uma ou mais *threads* sequenciais que são executadas na CPU, enquanto o *kernel* compreende um ou mais *kernels* paralelos que são organizados para rodar em uma GPU. O *kernel* é responsável por executar um programa sequencial em um conjunto de *threads* paralelas, permitindo a paralelização das tarefas [10].

Esse modelo de desenvolvimento possui um Programa Único Dados Múltiplos (SPMD) que é um estilo de *software* de programa único, dados múltiplos, no qual o desenvolvimento é feito para uma *thread* e depois é instanciado e executado por as outras *threads* em paralelo em várias GPUs [11].

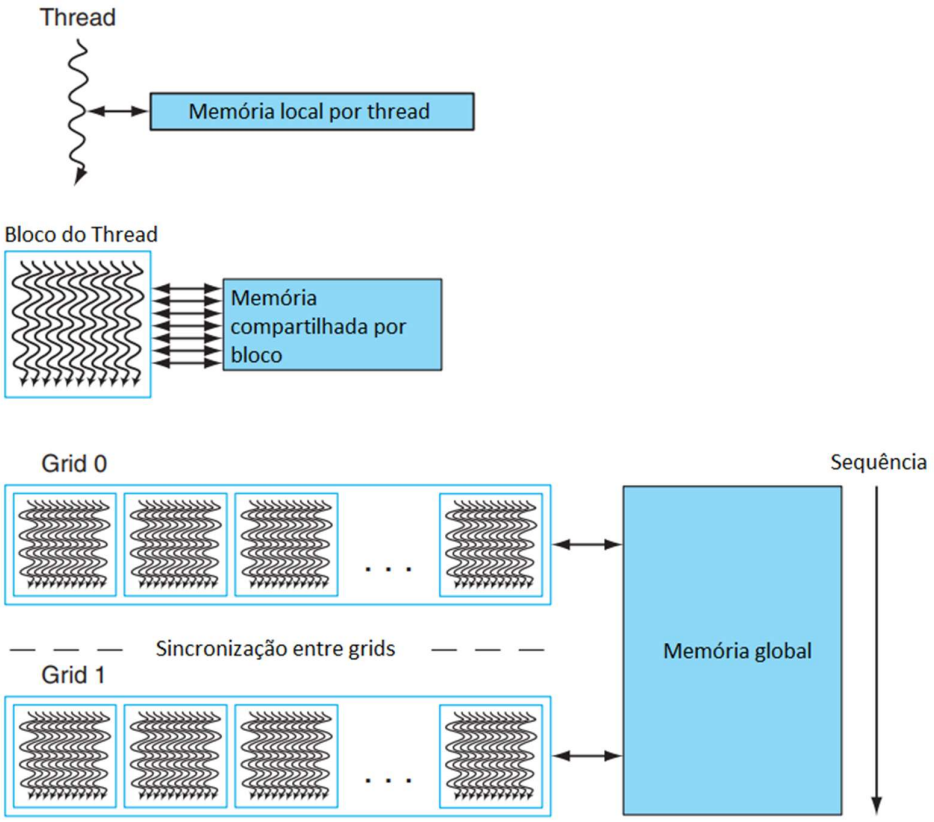
Essas *threads* são unidades básicas de execução dentro de um processo e permitem que um programa execute múltiplas operações simultaneamente de forma paralela. As *threads* de um mesmo processo compartilham o mesmo espaço de memória e recursos, o que facilita a

comunicação e sincronização entre elas. Cada *thread* pode ser executada independentemente, permitindo que diferentes partes de um programa rodem simultaneamente. Por isso criar e gerenciar *threads* é geralmente menos custoso do que criar e gerenciar processos, porque as *threads* de um mesmo processo compartilham muitos recursos [12].

No contexto do CUDA, *threads* são extremamente importantes para o processamento paralelo em GPUs, uma GPU pode conter milhares de *threads*, que são organizadas em blocos de *threads*, que por sua vez, são organizados em *grids*. Com isso as *threads* permitem que o problema a ser resolvido seja dividido em tarefas menores que podem ser executadas em paralelo, aumentando assim a eficiência e reduzindo o tempo total de execução, facilitando a escalabilidade e permitindo aproveitamento máximo dos recursos de *hardware* disponíveis, ajudando por exemplo, simulações que exigem muitos cálculos numéricos, podem ser divididas em várias *threads* para acelerar o processamento utilizando os múltiplos núcleos de CPU e ou unidades de processamento de GPU [13, 14].

A figura 1 ilustra como a arquitetura CUDA organiza a execução de programas em uma hierarquia de *threads*, blocos de *threads* e *grids*. Esta organização permite a comunicação eficiente e sincronização entre *threads*, otimizando o desempenho para computação paralela em GPUs [15].

Figura 1 - Hierarquia CUDA de *threads*, blocos de *threads* e *grids* de blocos, com espaços de memória correspondentes.



Fonte: Nickolls [15]

O Software de Paralelização do Penelope (SOPPE) é um código de simulação derivado do projeto MC-GPU [16, 17], um código Monte Carlo baseado em GPU que simula o transporte de fótons em uma geometria voxelizada. Utilizando o poder computacional das GPUs e desenvolvido com o modelo de programação CUDA da NVIDIA [18] para maximizar o desempenho nas GPUs NVIDIA, o SOPPE implementa um algoritmo de simulação de Monte Carlo massivamente paralela usando o código PENELOPE [19], sendo capaz de operar tanto com GPU, utilizando a linguagem CUDA, quanto com CPU, utilizando a linguagem C para ser executado.

Resultados e Aplicação

O SOPPE é um software para a área de física médica que ajudará a trazer inovação para a área, uma vez que não será mais necessário que o pesquisador utilize maquinas de grande porte e muitas vezes de difícil acesso para gerar por exemplo, simulações de absorção

de doses de radiação em tecidos e órgãos em um exame que utilize radiação ionizante, podendo utilizar assim o seu computador pessoal e sem precisar demandar dos alto custo computacional e tempo devido ao uso do paralelismo e isso tudo através de uma interface amigável.

O Anexo I é um exemplo de como é feito a configuração sem o SOPPE, o pesquisador precisa efetuar essa configuração através de um arquivo de texto, o que acaba sendo bastante complicado, pois um dado faltando ou em um local incorreto ocasionará em não funcionamento ou erros na simulação. Com a utilização deste arquivo é difícil identificar esses erros, o que acaba demandando bastante tempo e em alguns casos necessitando reinicialização dessa configuração.

Já com o uso SOPPE temos o benefício de uma interface amigável, como podemos ver na figura 2 que é a tela de configuração básica, que o usuário pode preencher com dados sobre o número total de histórias, o espectro de energia de raios-x, a geometria voxelizada, os materiais e por fim qual o modo de execução, se será via CPU ou GPU.

Caso o usuário demande de mais configurações ele pode contar com a parte de configurações avançadas da aplicação, como verifica-se na figura 3 que é a tela de configuração avançadas que contem desde as opções da simulação, como quantidades de GPUs, semente para a geração de números aleatórios e *threads* por bloco CUDA, passando pela parte da fonte que configura-se a posição da fonte, os cosenos e a abertura do feixe, na parte do detector de imagem, existe a possibilidade de escolher o nome do arquivo de imagem de saída, o número de pixels, as dimensões da imagem e a distância da fonte.

Ainda sobre as configurações avançadas, temos a parte de tomografia computadorizada, que dá a possibilidade de alterar configurações como número e ângulo das projeções, distância da fonte e translação, na parte de dose depositada, pode-se alterar dados como contagem do material, nome do arquivo de dose e dose roi no voxel dos eixos x, y e z. A parte da configuração avançada finaliza com a aba de geometria e materiais, onde pode-se escolher a geometria e adicionar os matérias.

Durante a configuração o usuário conta com avisos, tais como o auxílio se esquecer de selecionar o modo de execução, ao remover um material da listagem e ao adentrar na parte de configurações avançadas que podem ser vistos na figura 4. Após efetuadas as configurações necessárias e escolhido o modo de execução entre CPU e GPU e executada a simulação, o SOPPE conta com a tela com o resultado da simulação e os arquivos gerados que podem ser vistos na figura 5.

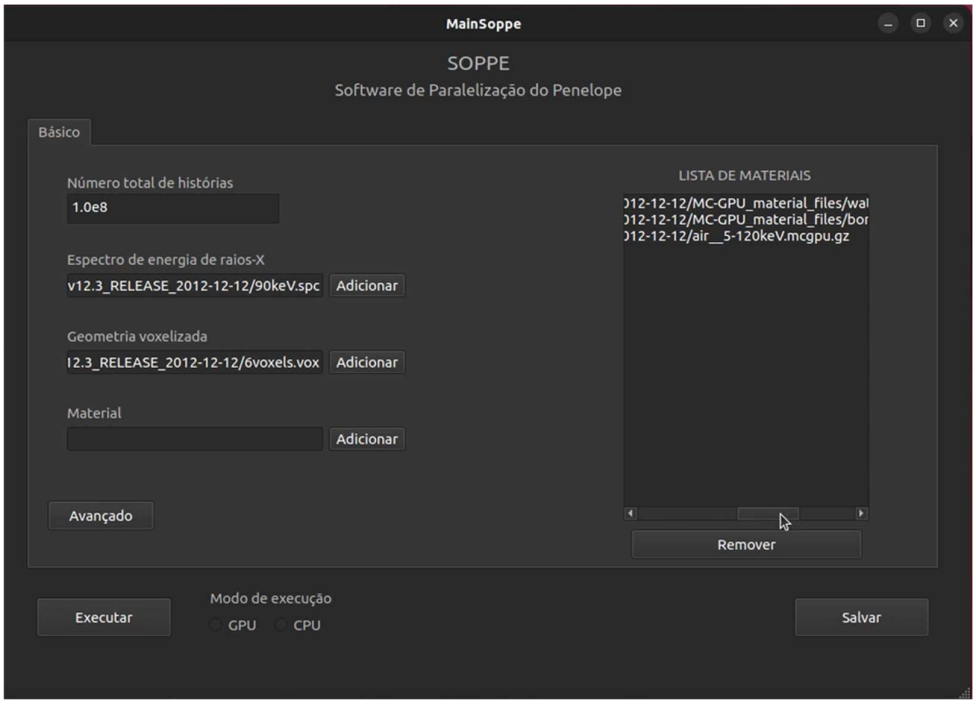
Além do ganho da interface, obteve-se ganhos na velocidade das simulações, para exemplificar isso, foram feitas diversas simulações com a configuração AMD Ryzen 9 5900x quando utilizado a CPU, que é a utilização do PENELOPE sem o paralelismo e quando utilizada a GPU, foi utilizada a placa de vídeo com a configuração NVidia RTX 4070 Ti Super 16GB, que verifica-se os ganhos do paralelismo. Em ambos os casos foram utilizados o sistema operacional o Linux Ubuntu na versão 22.04.3.

De posse dessas informações podemos ver na figura 6 que é a simulação com 6 voxels, fonte de 60kVp e os materiais foram titânio e osso, utilizando a CPU a simulação finalizou em pouco mais de 2411 segundos, já na figura 7, foi efetuada a simulação com os mesmos parâmetros utilizando a GPU e foi finalizado em um tempo de pouco menos do que 20 segundos, obtendo assim um ganho de aproximadamente 120 vezes quando comparado com a simulação feita utilizando a CPU, ou seja, sem o uso do paralelismo, que é o que o pesquisador que não tem acesso ao SOPPE utiliza.

Nas Figuras 9 a 23 verifica-se os resultados dos testes feitos sempre seguindo esse mesma ordem, simulação com a utilização da CPU, após isso a simulação com a utilização da GPU, alterando fonte, materiais e números de voxels. Após isso verifica-se nas Figuras 24 a 32 as imagens geradas através dessa simulações feitas anteriormente.

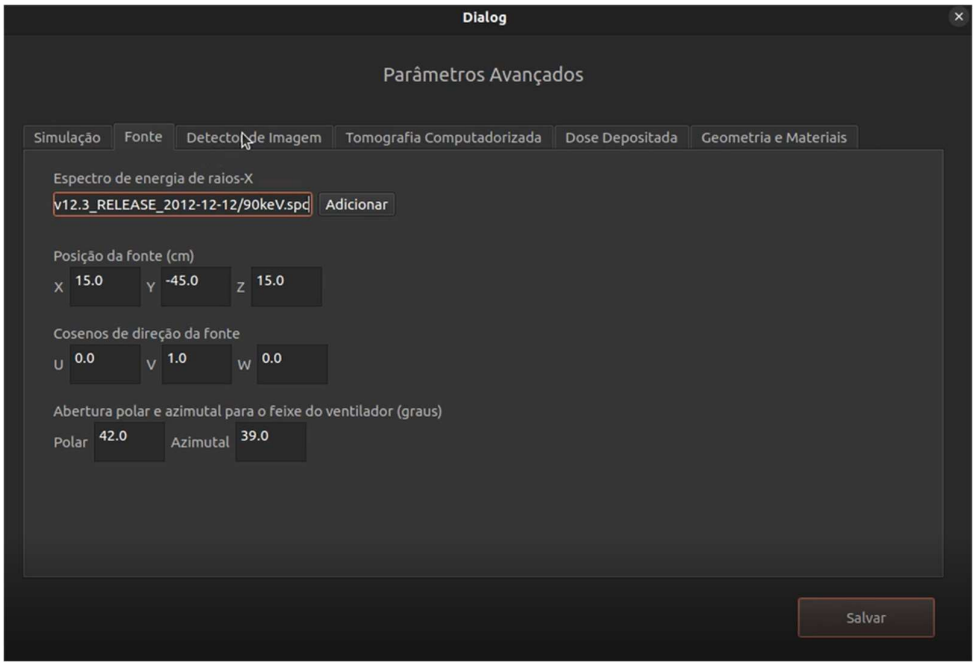
Os resultados foram excepcionais, pois as simulações utilizando a GPU foram feitas entre 111 e 145 vezes mais rápidas em relação as simulações feitas utilizando a CPU, como podem ser vistos na Tabela 1 que mostra um quadro comparativo dessas simulações efetuadas e seus respectivos ganhos.

Figura 2 – Interface de configuração básica do SOPPE.



Fonte: Elaboração Própria.

Figura 3 – Interface de configuração avançada do SOPPE.

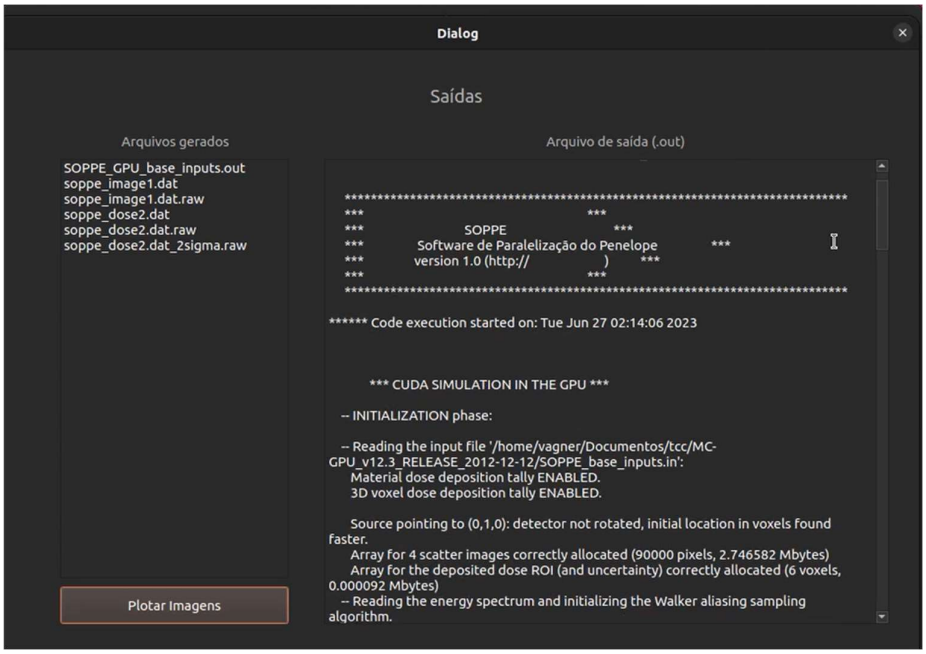


Fonte: Elaboração Própria.

Figura 4 – Mensagens de ajuda na interface de configuração do SOPPE.

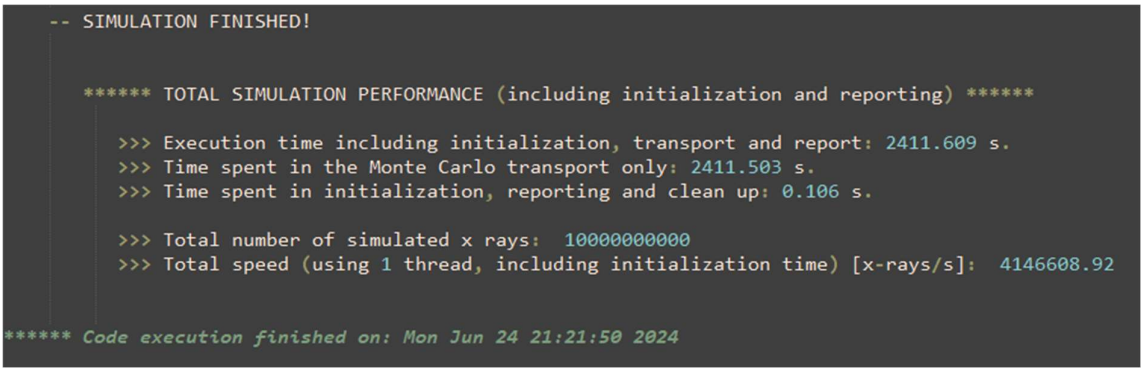
Fonte: Elaboração Própria.

Figura 5 – Retorno da simulação na interface do SOPPE.



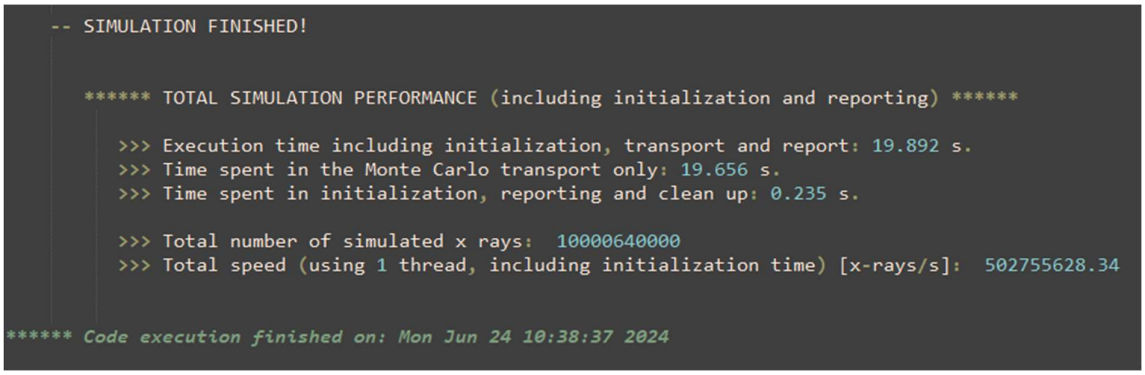
Fonte: Elaboração Própria.

Figura 6 – Resultado da simulação através da CPU com 6 voxels, fonte de 60 kVp 3,5 mm Al, utilizando os materiais titânio e osso.



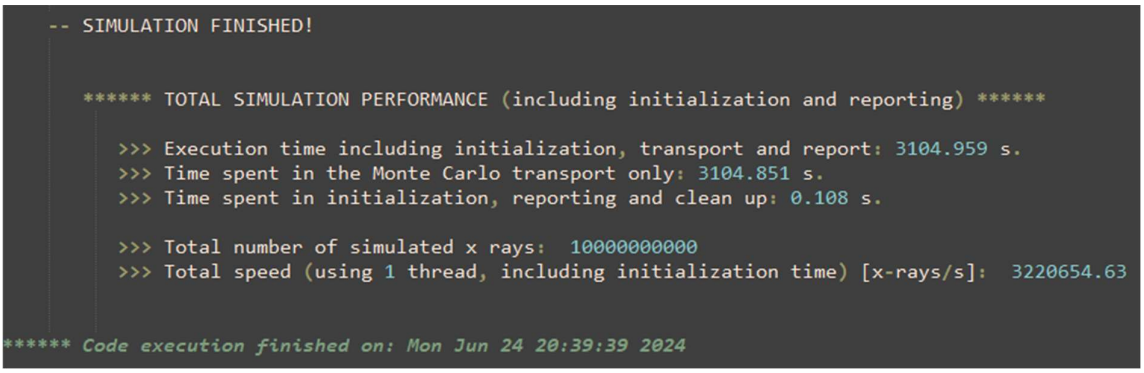
Fonte: Elaboração Própria.

Figura 7 – Resultado da simulação através da GPU com 6 voxels, fonte de 60 kVp 3,5 mm Al, utilizando os materiais titânio e osso.



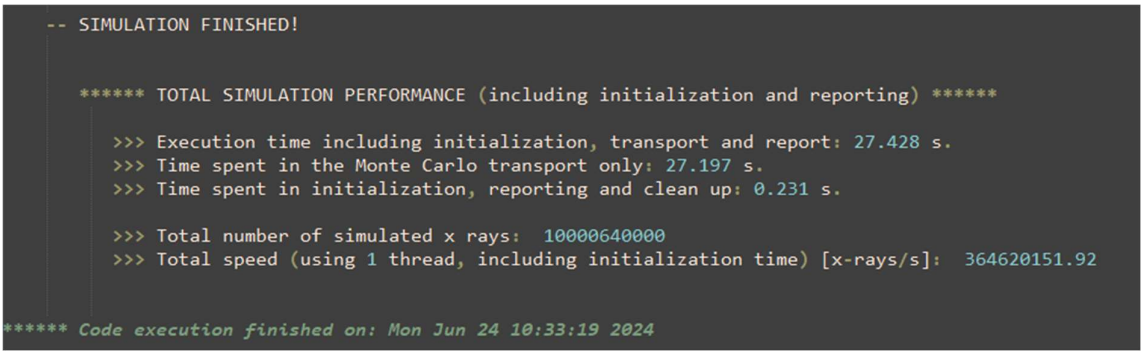
Fonte: Elaboração Própria.

Figura 8 – Resultado da simulação através da CPU com 6 voxels, fonte de 90 kVp 4 mm Al, utilizando os materiais titânio e osso.



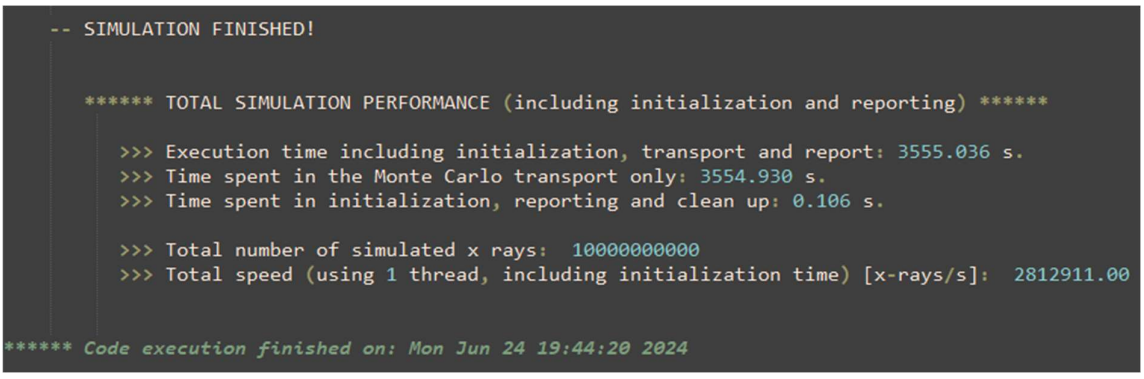
Fonte: Elaboração Própria.

Figura 9 – Resultado da simulação através da GPU com 6 voxels, fonte de 90 kVp 4 mm Al, utilizando os materiais titânio e osso.



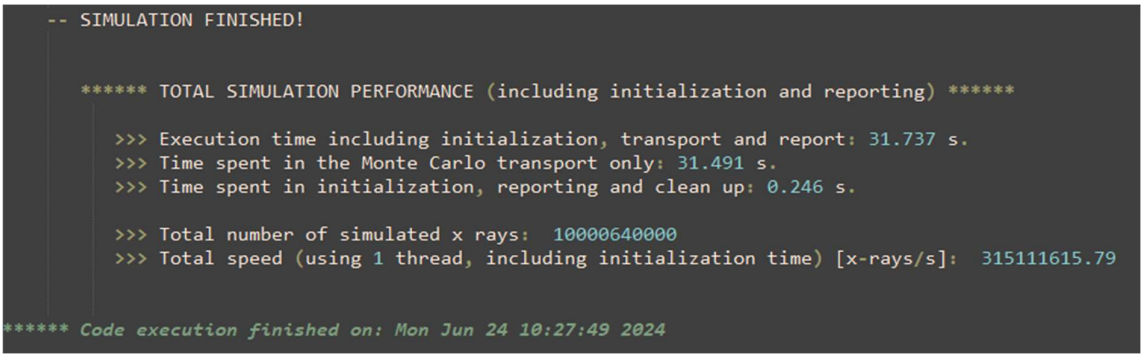
Fonte: Elaboração Própria.

Figura 10 – Resultado da simulação através da CPU com 6 voxels, fonte de 120 kVp 4,3 mm Al, utilizando os materiais titânio e osso.



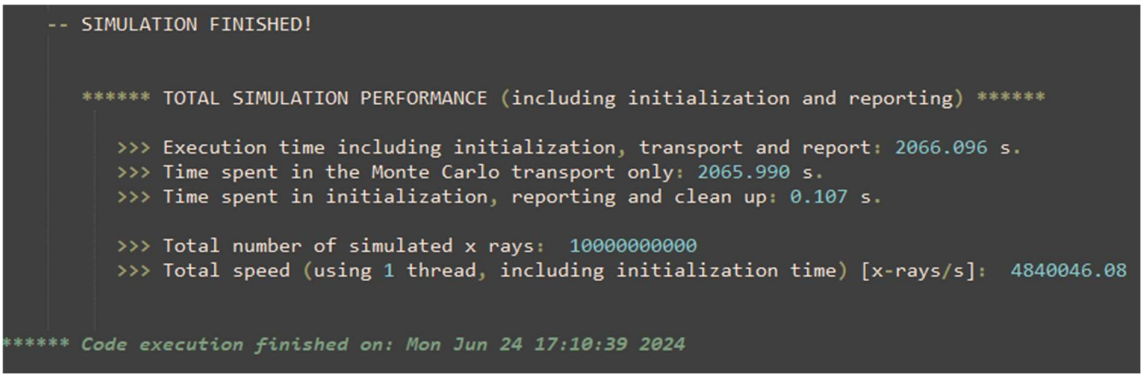
Fonte: Elaboração Própria.

Figura 11 – Resultado da simulação através da GPU com 6 voxels, fonte de 120 kVp 4,3 mm Al, utilizando os materiais titânio e osso.



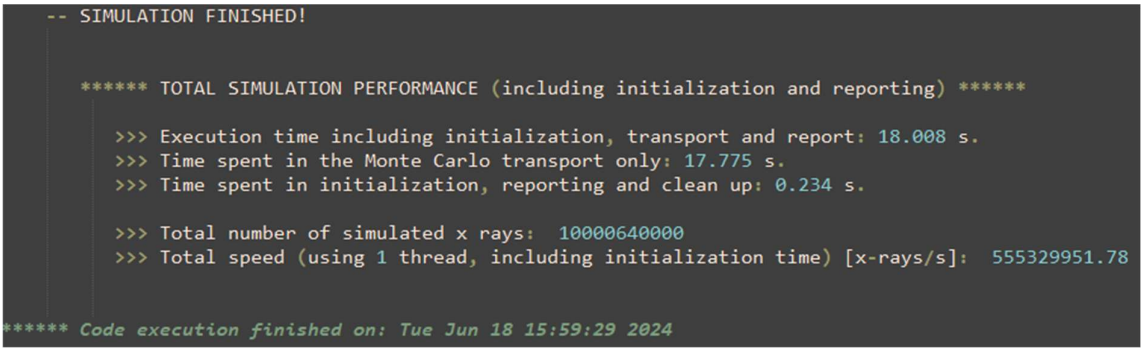
Fonte: Elaboração Própria.

Figura 12 – Resultado da simulação através da CPU com 900 voxels, fonte de 60 kVp 3,5 mm Al, utilizando os materiais titânio e osso.



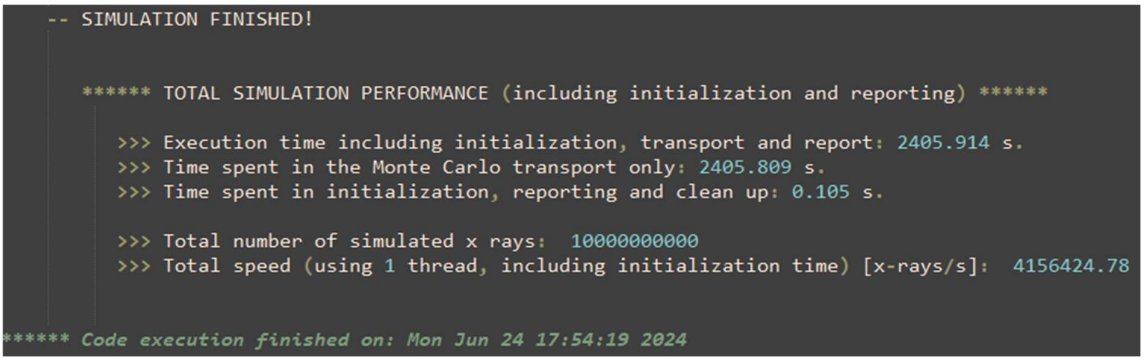
Fonte: Elaboração Própria.

Figura 13 – Resultado da simulação através da GPU com 900 voxels, fonte de 60 kVp 3,5 mm Al, utilizando os materiais titânio e osso.



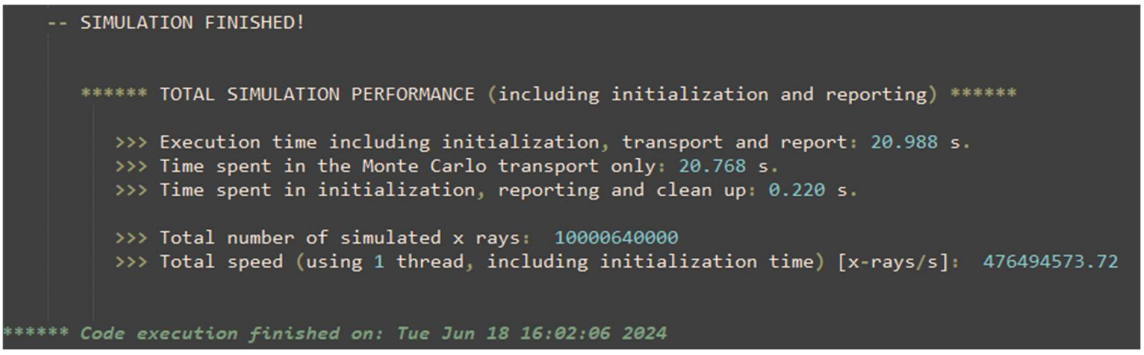
Fonte: Elaboração Própria.

Figura 14 – Resultado da simulação através da CPU com 900 voxels, fonte de 90 kVp 4 mm Al, utilizando os materiais titânio e osso.



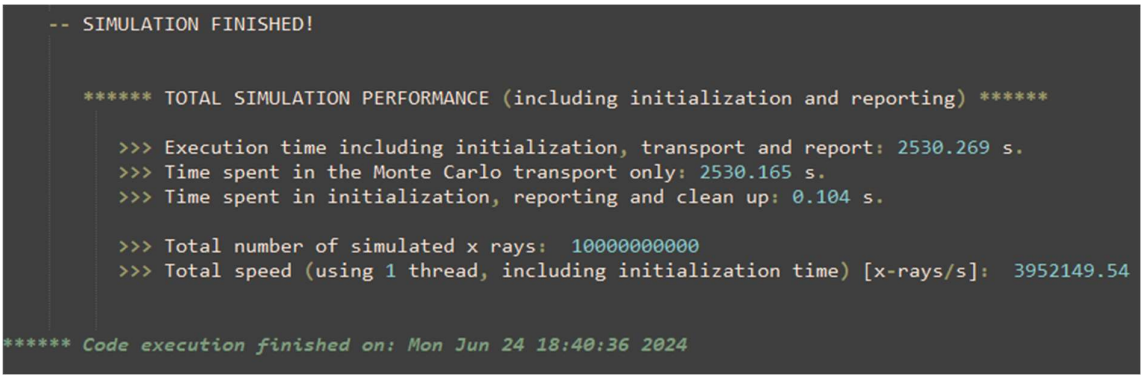
Fonte: Elaboração Própria.

Figura 15 – Resultado da simulação através da GPU com 900 voxels, fonte de 90 kVp 4 mm Al, utilizando os materiais titânio e osso.



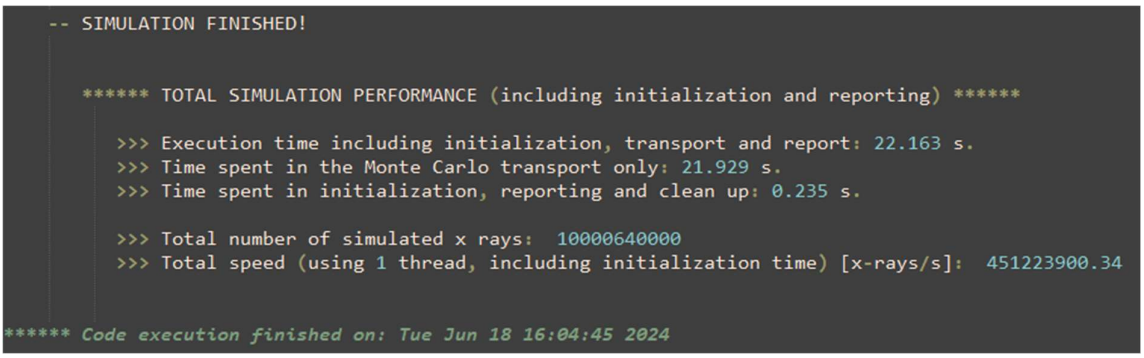
Fonte: Elaboração Própria.

Figura 16 – Resultado da simulação através da CPU com 900 voxels, fonte de 120 kVp 4,3 mm Al, utilizando os materiais titânio e osso.



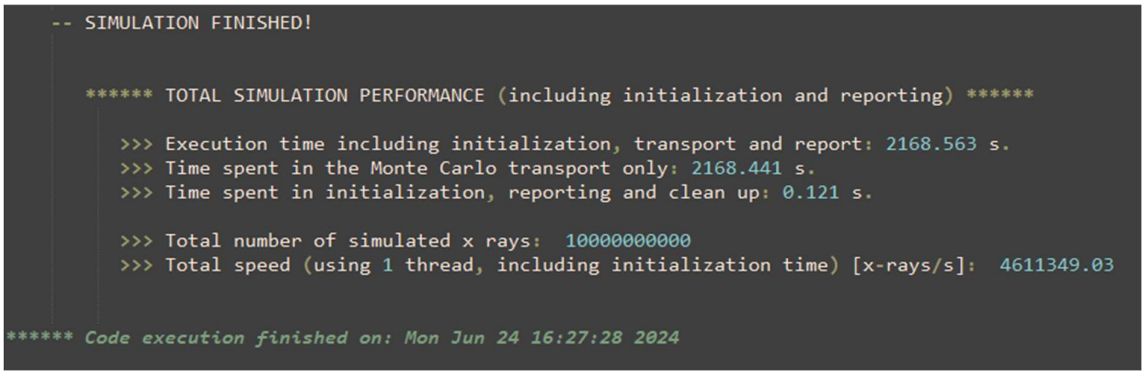
Fonte: Elaboração Própria.

Figura 17 – Resultado da simulação através da GPU com 900 voxels, fonte de 120 kVp 4,3 mm Al, utilizando os materiais titânio e osso.



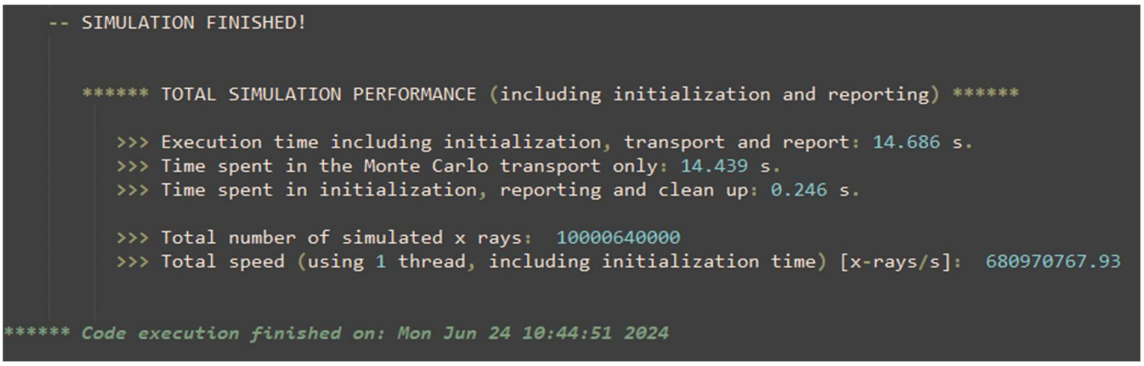
Fonte: Elaboração Própria.

Figura 18 – Resultado da simulação através da CPU com 900 voxels, fonte de 60 kVp 3,5 mm Al, utilizando os materiais titânio, osso e ar.



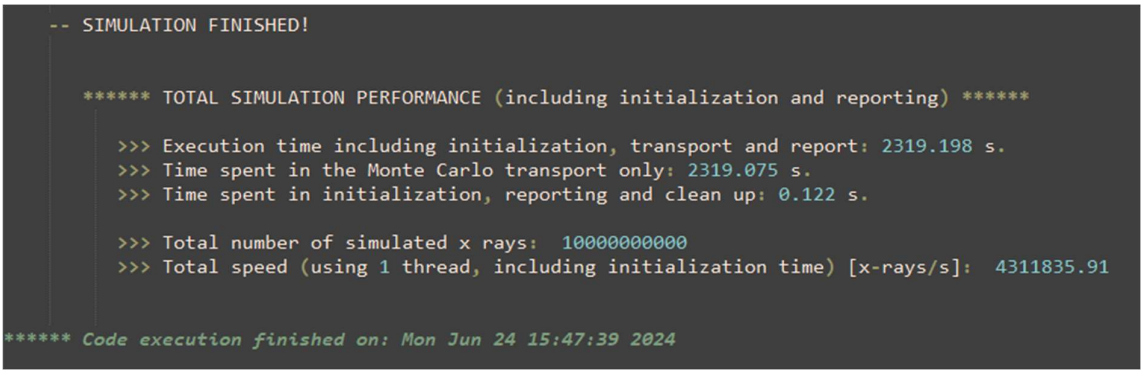
Fonte: Elaboração Própria.

Figura 19 – Resultado da simulação através da GPU com 900 voxels, fonte de 60 kVp 3,5 mm Al, utilizando os materiais titânio, osso e ar.



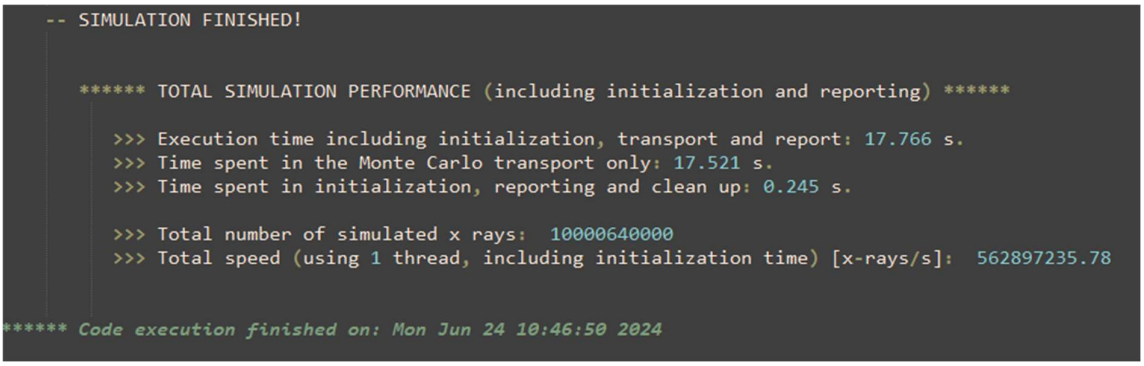
Fonte: Elaboração Própria.

Figura 20 – Resultado da simulação através da CPU com 900 voxels, fonte de 90 kVp 4 mm Al, utilizando os materiais titânio, osso e ar.



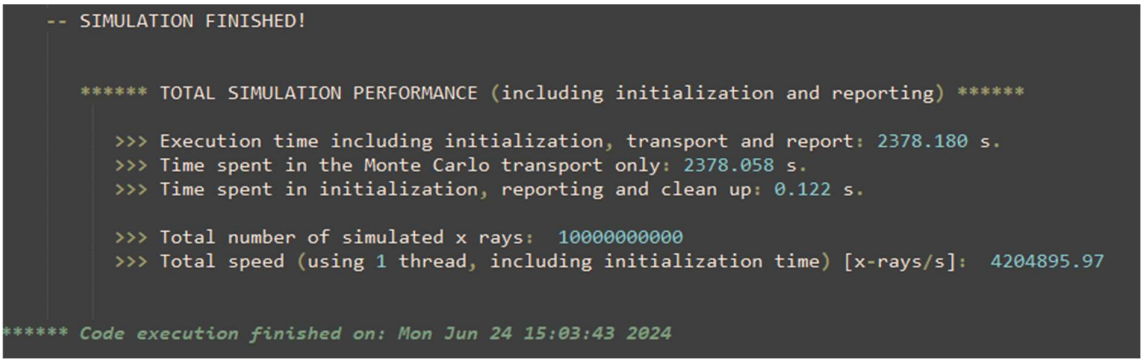
Fonte: Elaboração Própria.

Figura 21 – Resultado da simulação através da GPU com 900 voxels, fonte de 90 kVp 4 mm Al, utilizando os materiais titânio, osso e ar.



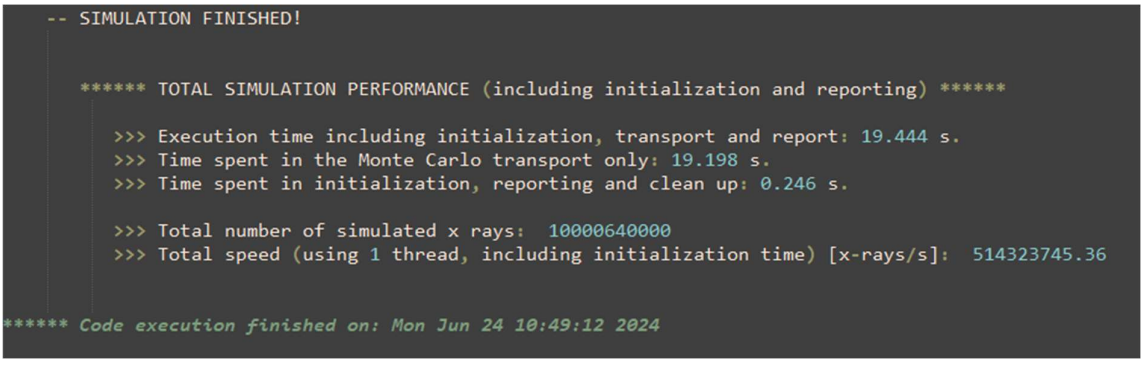
Fonte: Elaboração Própria.

Figura 22 – Resultado da simulação através da CPU com 900 voxels, fonte de 120 kVp 4,3 mm Al, utilizando os materiais titânio, osso e ar.



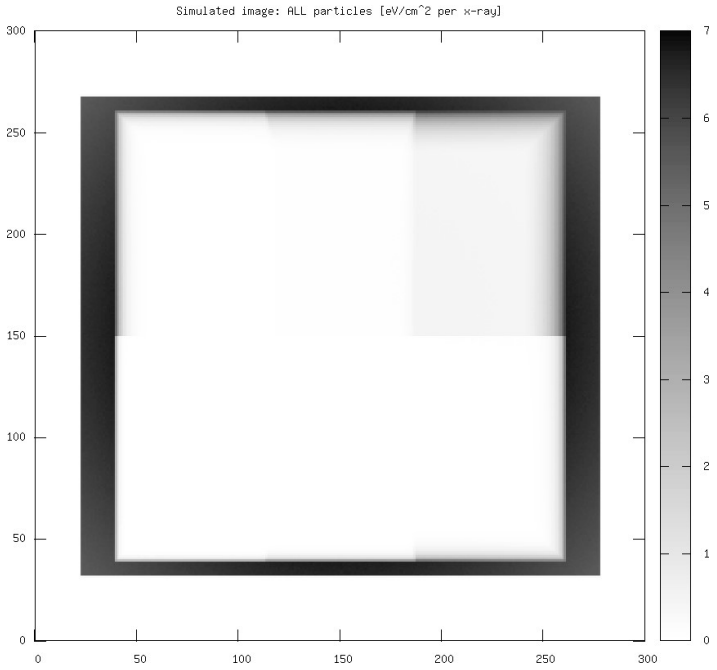
Fonte: Elaboração Própria.

Figura 23 – Resultado da simulação através da CPU com 900 voxels, fonte de 120 kVp 4,3 mm Al, utilizando os materiais titânio, osso e ar.



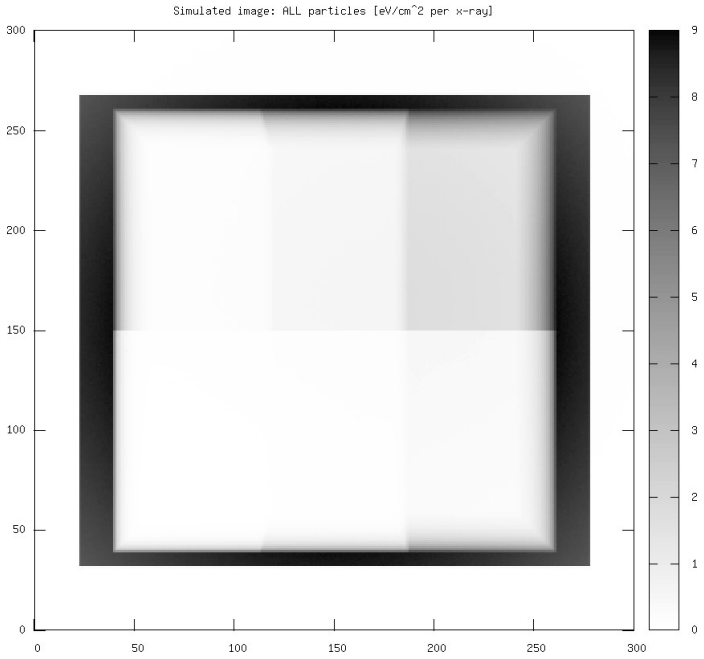
Fonte: Elaboração Própria.

Figura 24 – Imagem gerada da simulação com 6 voxels, com fonte de 60 kVp 3,5 mm Al, utilizando os materiais titânio e osso.



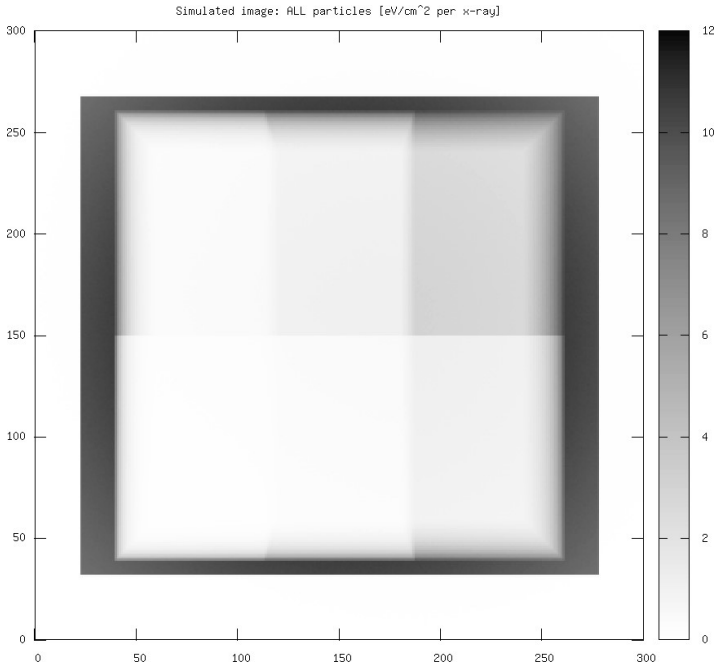
Fonte: Elaboração Própria.

Figura 25 – Imagem gerada da simulação com 6 voxels, com fonte de 90 kVp 4 mm Al, utilizando os materiais titânio e osso.



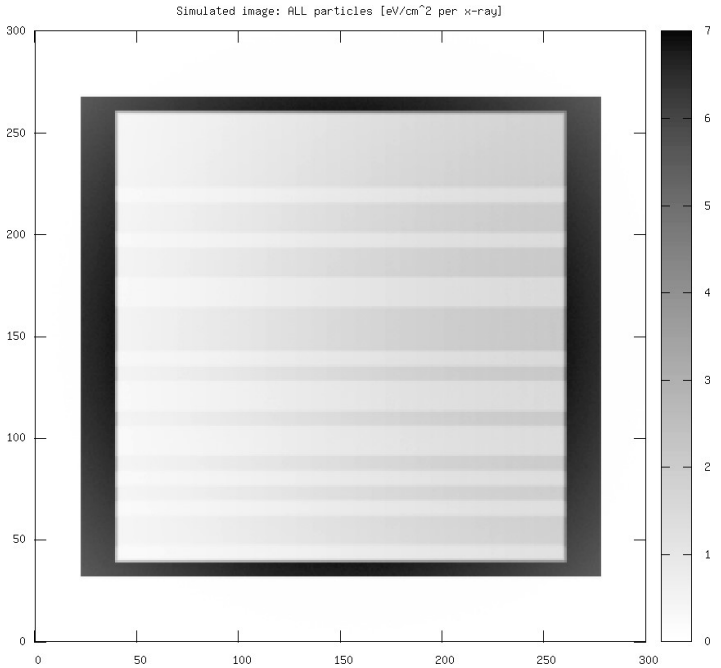
Fonte: Elaboração Própria.

Figura 26 – Imagem gerada da simulação com 6 voxels, com fonte de 120kVp 4,3 mm Al, utilizando os materiais titânio e osso.



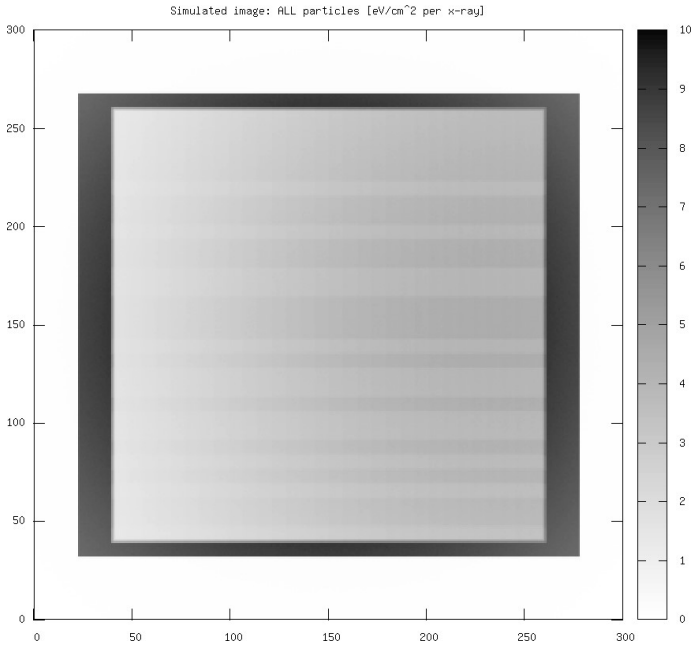
Fonte: Elaboração Própria.

Figura 27 – Imagem gerada da simulação com 900 voxels, com fonte de 60 kVp 3,5 mm Al, utilizando os materiais titânio e osso.



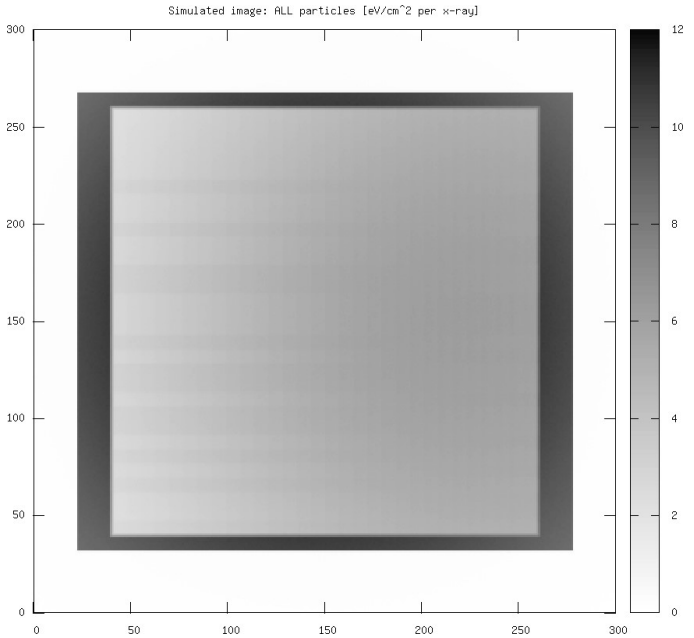
Fonte: Elaboração Própria.

Figura 28 – Imagem gerada da simulação com 900 voxels, com fonte de 90 kVp 4 mm Al, utilizando os materiais titânio e osso.



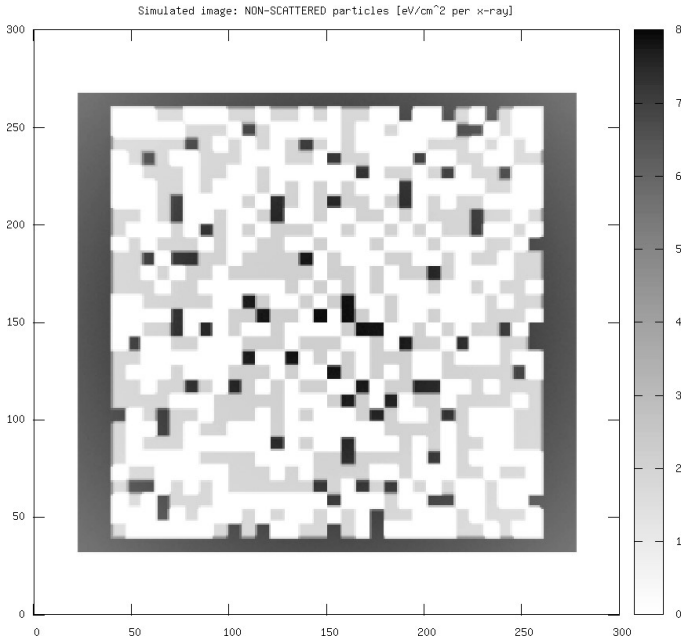
Fonte: Elaboração Própria.

Figura 29 – Imagem gerada da simulação com 900 voxels, com fonte de 120 kVp 4,3 mm Al, utilizando os materiais titânio e osso.



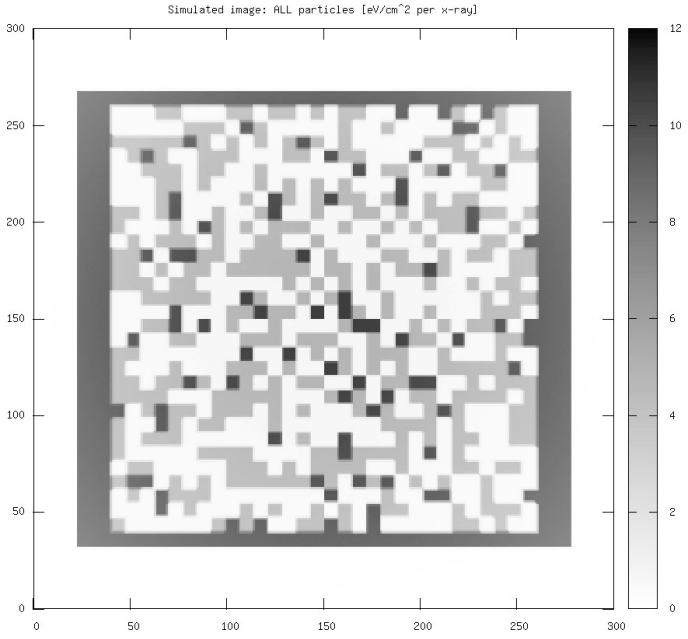
Fonte: Elaboração Própria.

Figura 30 – Imagem gerada da simulação com 900 voxels, com fonte de 60 kVp 3,5 mm Al, utilizando os materiais titânio, osso e ar.



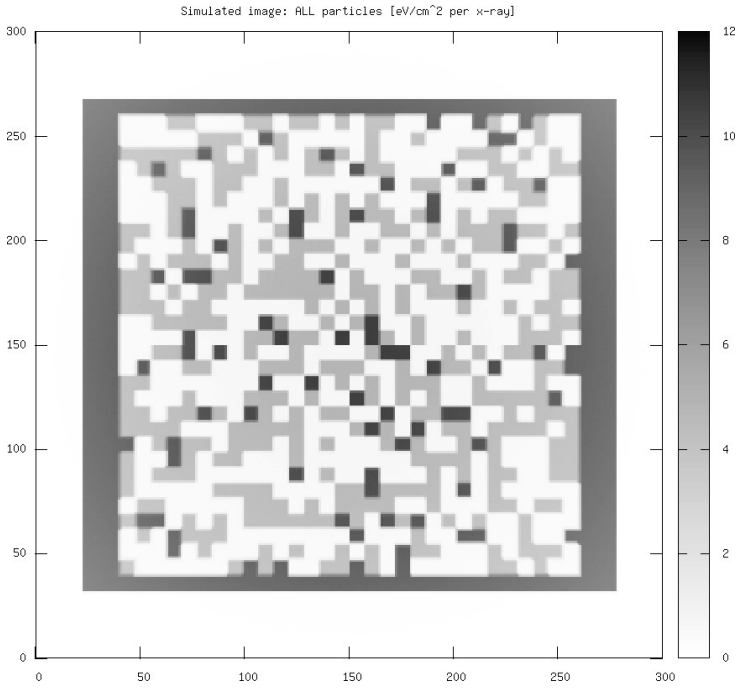
Fonte: Elaboração Própria.

Figura 31 – Imagem gerada da simulação com 900 voxels, com fonte de 90 kVp 4 mm Al, utilizando os materiais titânio, osso e ar.



Fonte: Elaboração Própria.

Figura 32 – Imagem gerada da simulação com 900 voxels, com fonte de 120 kVp 4,3 mm Al, utilizando os materiais titânio e osso.



Fonte: Elaboração Própria.

Tabela 1 – Comparação dos tempos obtidos com simulações utilizando a CPU e a GPU.

Fonte*1	Voxels	Material	eV/g/história	Desvio Padrão	CPU	GPU	Ganho
60 kVp_3.5	6	T / O	3,41 / 2,98	9E-05 / 8E-05	2.412	20	12060%
90 kVp_4.0	6	T / O	4,27 / 3,28	1,10E-04 / 9E-05	3.105	27	11500%
120 kVp_4.3	6	T / O	4,76 / 3,40	1,30E-04 / 1E-04	3.555	32	11109%
60 kVp_3.5	900	T / O	25,49 / 8,61	7,90E-04 / 2,20E-04	2.066	18	11478%
90 kVp_4.0	900	T / O	25,33 / 7,95	9,10E-04 / 2,50E-04	2.406	21	11457%
120 kVp_4.3	900	T / O	23,96 / 7,40	9,60E-04 / 2,60E-04	2.530	22	11500%
60 kVp_3.5	900	T / O / Ar	8,29 / 10,68 / 3,24	1,90E-04 / 4,10E-04 / 1,950E-02	2.169	15	14460%
90 kVp_4.0	900	T / O / Ar	10,16 / 9,03 / 2,47	2,40E-04 / 4,30E-04 / 1,750E-02	2.319	18	12883%
120 kVp_4.3	900	T / O / Ar	10,85 / 8,06 / 2,27	2,70E-04 / 4,30E-04 / 1,690E-02	2.378	19	12516%

Fonte: Elaboração Própria.

Legenda:

Fonte = Espetro (Fótons)

T = Titânio

O = Osso

CPU / GPU = Tempo em segundos

Ganho = Tempo CPU / Tempo GPU

O relatório apresentado sobre o SOPPE destaca seus aspectos mais relevantes, evidenciando o potencial promissor da aplicação. Embora ainda sejam necessários testes adicionais com variações de configurações para cobrir uma gama mais ampla de cenários, o software demonstra-se altamente eficaz.

Em particular, comprova-se que pesquisadores e professores podem utilizar suas máquinas pessoais para realizar simulações e gerar imagens a partir dessas simulações, contribuindo significativamente tanto para o avanço da área da física médica quanto para a exemplificação de conceitos para estudantes.

3 CONCLUSÕES / TRABALHOS FUTUROS

Concluímos que objetivo principal foi alcançado, pois com o uso do SOPPE houve ganho de desempenho que foi alcançado por meio da aceleração proporcionada pelo uso da GPU, resultando em uma melhoria de desempenho superior a 100 vezes em comparação com as simulações realizadas com a utilização da CPU, obtendo também o registro de software, que assegura a autoria da ideia que está documentado no Anexo II.

Além disso, outros benefícios do uso do SOPPE poderão ser obtidos em trabalhos futuros, que trarão melhorias como: a possibilidade de o usuário acessar o histórico das simulações realizadas, a compatibilidade com o sistema operacional Windows e outras placas de vídeo, a opção de agendamento das simulações com desligamento automático da máquina após a sua conclusão, e a paralelização de outros códigos.

REFERÊNCIAS

[1] NEA, “PENELOPE-2018: A Code System for Monte Carlo Simulation of Electron and Photon Transport. Workshop Proceedings Barcelona,” Spain 28 January – 1 February 2019. Disponível em: <https://www.oecd-nea.org/jcms/pl_46441/penelope-2018-a-code-system-for-monte-carlo-simulation-of-electron-and-photon-transport>. Acesso em: 16 mai. 2022.

[2] Koger B, Kirkby C. Optimization of photon beam energies in gold nanoparticle enhanced arc radiation therapy using Monte Carlo methods. Phys Med Biol. 2016 Dec 21;61(24):8839-8853. doi: 10.1088/1361-6560/61/24/8839. Epub 2016 Dec 2. PMID: 27910829.

[3] SOBOL, I. M. A Primer for the Monte Carlo Method. London: CRC Press, 1994.

[4] Shi M, et al. GPU-accelerated Monte Carlo simulation of MV-CBCT. Phys Med Biol. 2020 Dec 2;65(23):235042. doi: 10.1088/1361-6560/abaeba. PMID: 33263311.

[5] KAWRAKOW, I. Accurate condensed history Monte Carlo simulation of electron transport. I. EGSnrc, the new EGS4 version. Medical Physics, College Park, v. 27, n. 3, p. 485-498, 2000.

[6] Agostinelli, S., Allison, J., Amako, K., Apostolakis, J., Araujo, H., et al. (2003). Geant4—a simulation toolkit. Nuclear Instruments and Methods in Physics Research, 506(3), 250-303.

[7] V. Giménez-Alventosa, V. Giménez Gómez, and S. Oliver. Penred: Anextensible and parallel monte-carlo framework for radiation transportbased on penelope. Computer Physics Communications, page 108065,2021.

[8] Owens, John & Houston, Mike & Luebke, David & Green, Simon & Stone, John & Phillips, James. (2008). GPU computing. Proceedings of the IEEE. 96. 879-899. 10.1109/JPROC.2008.917757.

[9] Michalakes, John & Vachharajani, Manish. (2008). GPU acceleration of numerical weather prediction. Parallel Processing Letters. 18. 1 - 7. 10.1109/IPDPS.2008.4536351.

[10] DALLY, William J.; KECKLER, Stephen W.; KIRK, David B. Evolution of the graphics processing unit (GPU). IEEE Micro, v. 41, n. 6, p. 42-51, 2021.

[11] Raina, Rajat & Madhavan, Anand & Ng, Andrew. (2009). Large-scale deep unsupervised learning using graphics processors. Proceedings of the 26th International Conference On Machine Learning, ICML 2009. 382. 110. 10.1145/1553374.1553486.

[12] Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). CUDA: An Architecture for High-Performance Computing. IEEE Micro, 28(2), 39-55.

[13] Hennessy, J. L., & Patterson, D. A. (2017). Computer Architecture: A Quantitative Approach (6th ed.). Morgan Kaufmann.

[14] Kirk, D. B., & Hwu, W. W. (2010). Programming Massively Parallel Processors: A Hands-on Approach. Morgan Kaufmann.

[15] Nickolls, J.R., & Dally, W.J. (2010). The GPU Computing Era. IEEE Micro, 30.

[16] Badal A, Badano A. Accelerating Monte Carlo simulations of photon transport in a voxelized geometry using a massively parallel graphics processing unit. Med Phys. 2009 Nov;36(11):4878-80. doi: 10.1118/1.3231824. PMID: 19994495.

[17] MC-GPU, “Código de simulação de Monte Carlo”, acesso em: 5 jun 2023.
[Online]. Disponível em <https://github.com/DIDSR/MCGPU>.

[18] NVIDIA Corporation, NVIDIA CUDA(TM) Develop, Optimize and Deploy GPU-Accelerated Apps. Disponível em <https://developer.nvidia.com/cuda-toolkit> Acesso em: 26 mai. 2023.

[19] RODRÍGUEZ, E.A.V.; ALCÓN, E.P.Q.; RODRIGUEZ, M.L.; et al: Dosimetric parameters estimation using PENELOPE Monte-Carlo simulation code: Model 6711 a 125I brachytherapy seed, Applied Radiation and Isotopes, vol.63, pp. 41– 48, 2005

ANEXO I – Exemplo de configuração de arquivo antes da utilização da interface do SOPPE

6voxels.vox - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
# >>>> PenEasy 2008 VOXEL GEOMETRY FILE >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
#
#   Simple geometry with 9 voxels with 2 materials and 3 different densities.
#   The format of this voxel geometry is compatible with penEasy 2008 and
#   with MC-GPU v.1.1.
#   The voxels are listed with the X index changing first, then the Y index and
#   the Z index last. Blank lines can be optionally inserted to separate the
#   each X column for different Y, and double blanks to separate XY blocks.
#
[SECTION VOXELS HEADER v.2008-04-13]
  3     1     2     No. OF VOXELS IN X,Y,Z
 10.0  10.0 15.0   VOXEL SIZE (cm) ALONG X,Y,Z
 1                  COLUMN NUMBER WHERE MATERIAL ID IS LOCATED
 2                  COLUMN NUMBER WHERE THE MASS DENSITY IS LOCATED
 1                  BLANK LINES AT END OF X,Y-CYCLES (1=YES,0=NO)
[END OF VXH SECTION]
1 1.50
1 1.00
1 0.50


2 1.50
2 1.00
2 0.50
```

ANEXO II – Certificado de Registro de Programa de Computador

REPÚBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DO DESENVOLVIMENTO, INDÚSTRIA, COMÉRCIO E SERVIÇOS
INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL
DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS

Certificado de Registro de Programa de Computador

Processo Nº: **BR512024000360-6**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 10/05/2023, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: Software de Paralelização do código Penelope - PenEasy (SOPPE)

Data de criação: 10/05/2023

Titular(es): INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA BAHIA - IFBA

Autor(es): WILSON OTTO GOMES BATISTA; ANTÔNIO CARLOS DOS SANTOS SOUZA; RICARDO CÉSAR VASCONCELOS LOUREIRO; VAGNER DA SILVA DE JESUS

Linguagem: C++

Campo de aplicação: ED-01; FQ-11

Tipo de programa: AP-01; CD-06; FA-01; GI-06; LG-01; SO-02

Algoritmo hash: SHA-512

Resumo digital hash:
737348cde408124dd75ced77283a29845c770c87d2cc8493d5f00157b9cac8659acad4ace526a118c265912484830 66d0dfc2ad96526ef064cb3086a8b743558

Expedido em: 20/02/2024

Aprovado por:
Joelson Gomes Pequeno
Chefe Substituto da DIPTO - PORTARIA/INPI/DIRPA Nº 02, DE 10 DE FEVEREIRO DE 2021

APENDICE I – Características de absorção e espelhamento dos materiais.

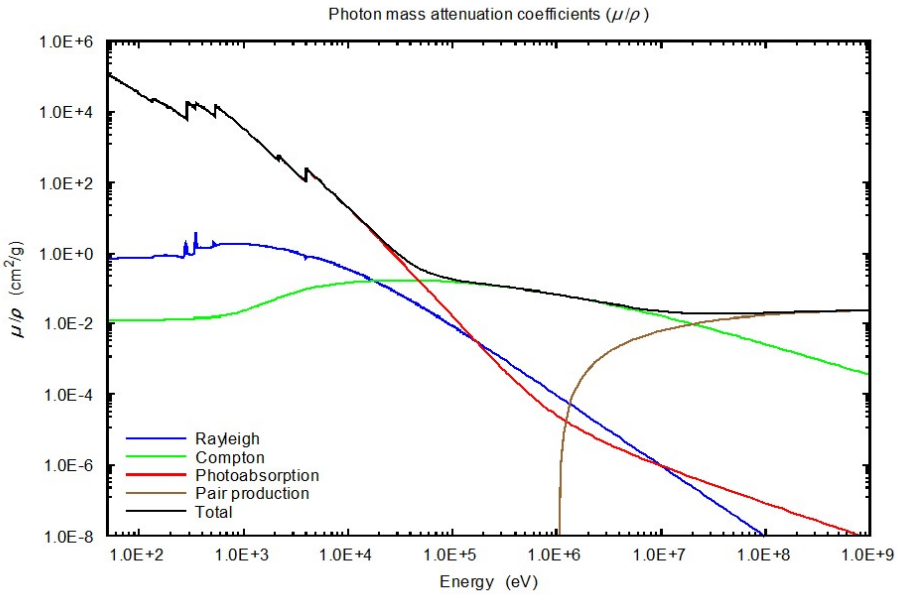


Figura 33 – Coeficientes de atenuação para o titânio: espalhamento Rayleigh; efeito fotoelétrico; espalhamento Compton; produção de pares e total.

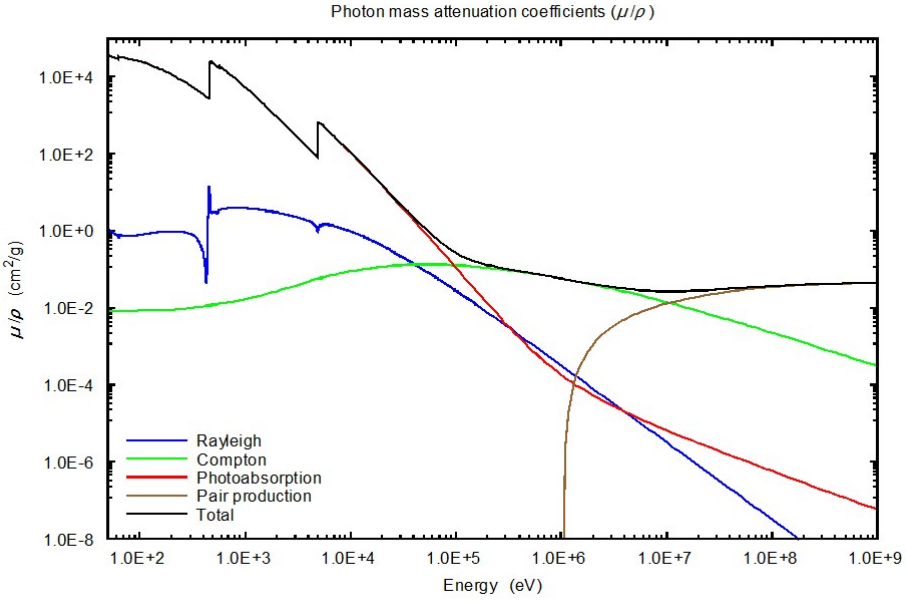


Figura 34 – Coeficientes de atenuação para o osso: espalhamento Rayleigh; efeito fotoelétrico; espalhamento Compton; produção de pares e total.

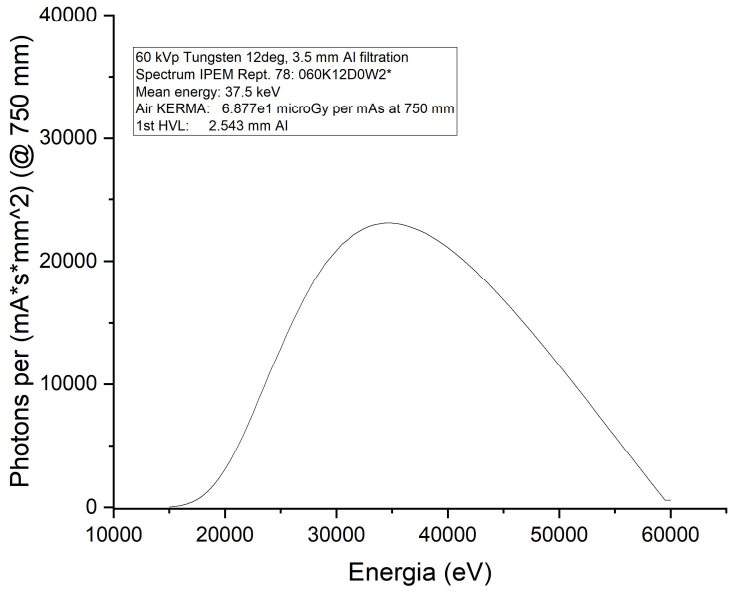


Figura 35 – Espectro de distribuição de energia para 60 kVp; 3,5 mm de Al obtido através do relatório IPEM#78 utilizado como fonte nas simulações.

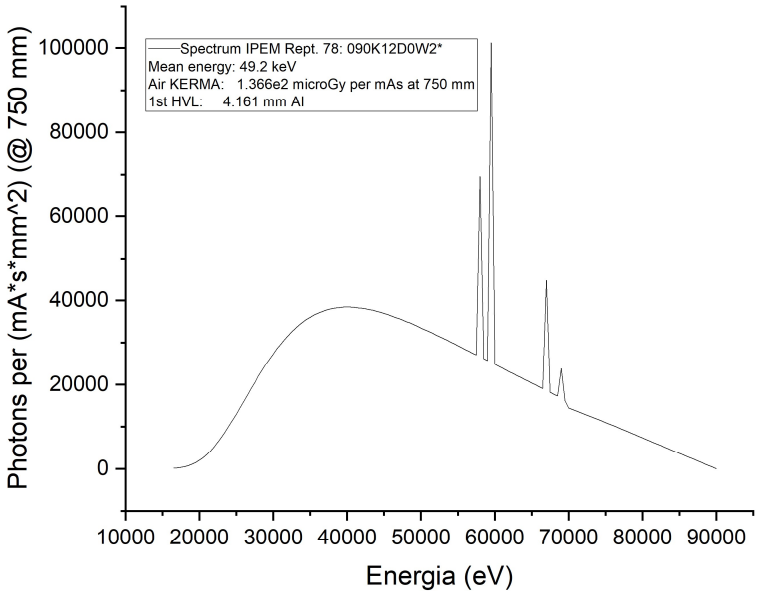


Figura 36 – Espectro de distribuição de energia para 90 kVp; 4 mm de Al obtido através do relatório IPEM#78 utilizado como fonte nas simulações.

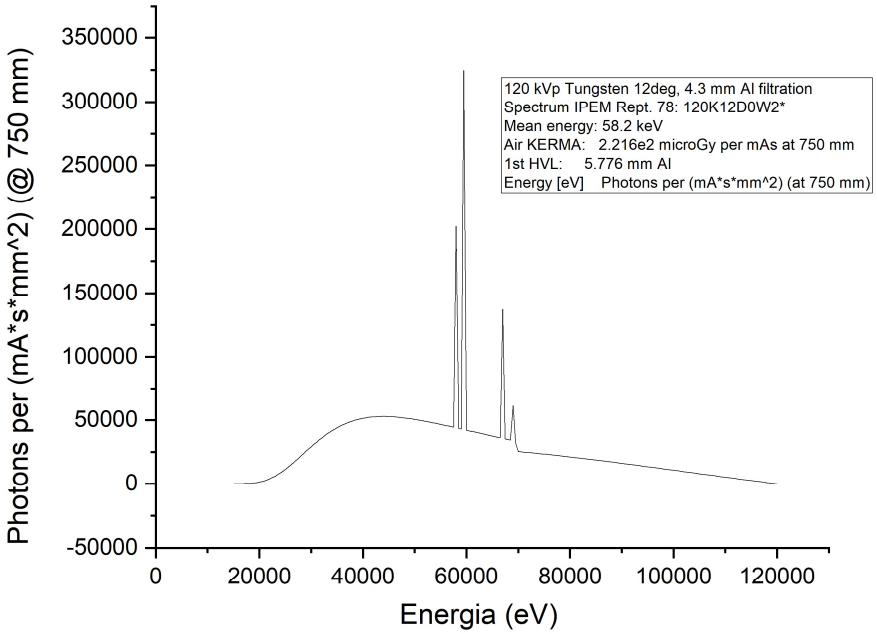


Figura 37 – Espectro de distribuição de energia para 120 kVp; 4,3 mm de Al obtido através do relatório IPEM#78 utilizado como fonte nas simulações.

APENDICE II – Manual do SOPPE

MANUAL

SOPPE

Software de Paralelização do Código Penelope

1. Introdução

O SOPPE (Software de Paralelização do Código Penelope / PenEasy) é um código de simulação que utilizado o Método de Monte Carlo para gerar imagens sintéticas radiográficas e tomografias computadorizadas utilizando o poder computacional de placas de unidade de processamento gráfico (GPU)[1]. Para isso, ele utiliza modelos realistas da anatomia humana e realiza simulações com a implementação da simulação de Monte Carlo de forma paralela para realizar o transporte de raios-X em modelos computacionais[2].

O SOPPE utiliza como base o MC-GPU, um software de código livre, com última release em 2012, que realiza o transporte de raios-X de forma paralela utilizado o poder computacional presente nas GPUs NVIDIA usando o modelo de programação CUDA, no qual seus modelos de interação e propriedades dos materiais foram adaptados do PENELOPE 2006, sendo este também um software de código livre[3,4].

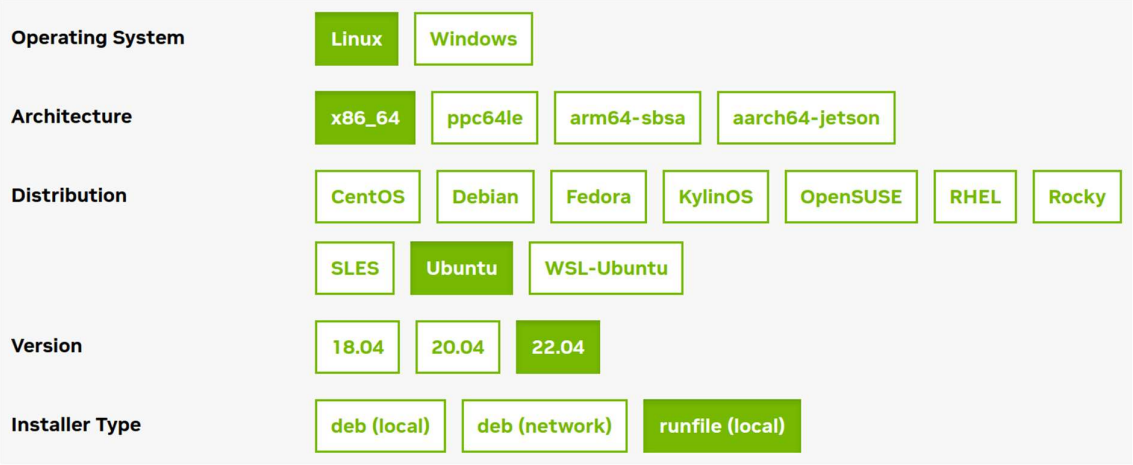
Para utilização do software é necessário estar utilizando um sistema Linux além da instalação e configuração de algumas ferramentas como as bibliotecas CUDA, o compilador GNU GCC e a biblioteca openMPI.

2. Instalação

2.1. CUDA Toolkit

Caso seja solicitado durante a instalação, digite a senha que utiliza para acessar o sistema operacional para continuar.

a. Acesse o site https://developer.nvidia.com/cuda-downloads.
b. Escolha as opções para o sistema Linux que estiver utilizando (no exemplo utilizamos o Ubuntu) com a última opção sendo **runfile (local)**;

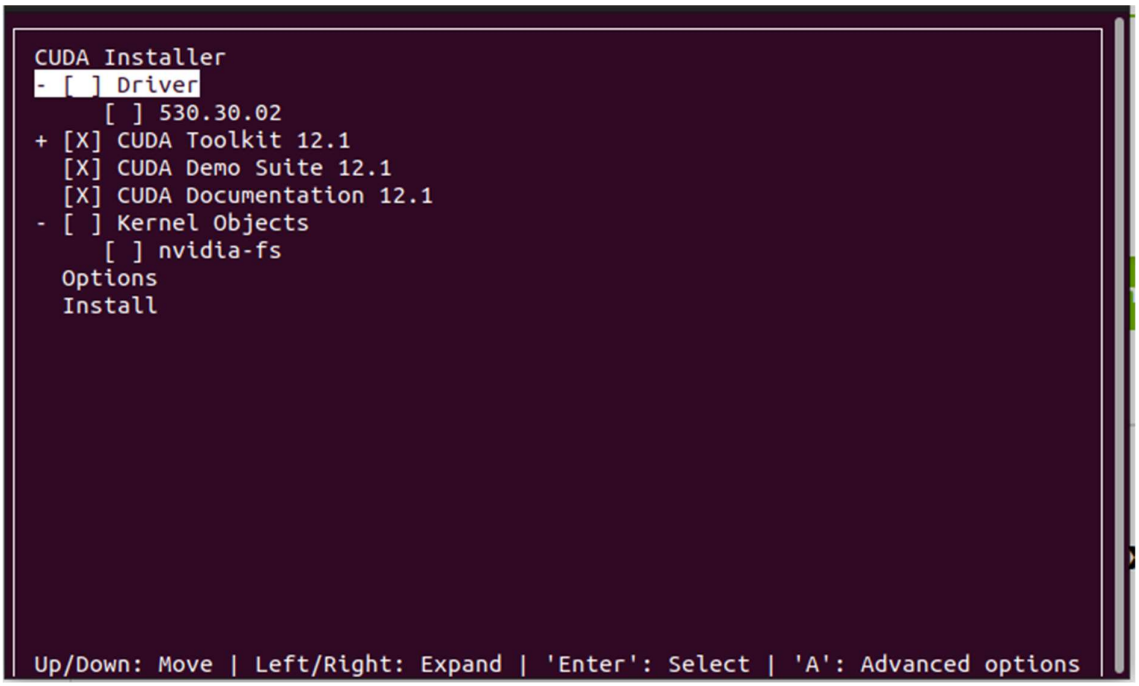


c. Após escolher a última opção, será apresentado os comandos para continuar a instalação;

d. Abra o terminal;
e. Digite **sudo apt update;**
f. Digite **sudo apt upgrade-y;**
g. Digite **apt search nvidia-driver**;
h. Digite **sudo apt install nvidia-driver-numero da versao**, substituindo o número da versão pelo número do último driver sem o sufixo **-server** que apareceu no item anterior (no exemplo driver 530);

i. Após a instalação, repita os passo **e** e **f** e reinicie a máquina;
j. No terminal, execute a primeira instrução que apareceu na tela do passo **c** (no exemplo a versão 12.1.1);
k. Após o término do download, execute a segunda instrução;
l. Aparecerá os termos de uso. Aceitos digitando **accept** e apertando Enter;

m. Nesse tela, não instalaremos o driver novamente. Para desmarcar a opção de driver, mantenha na seleção em driver e aperte a tecla de Espaço, sumindo assim o X da marcação. Em seguida, desça até a opção **Install** e aperte Enter;

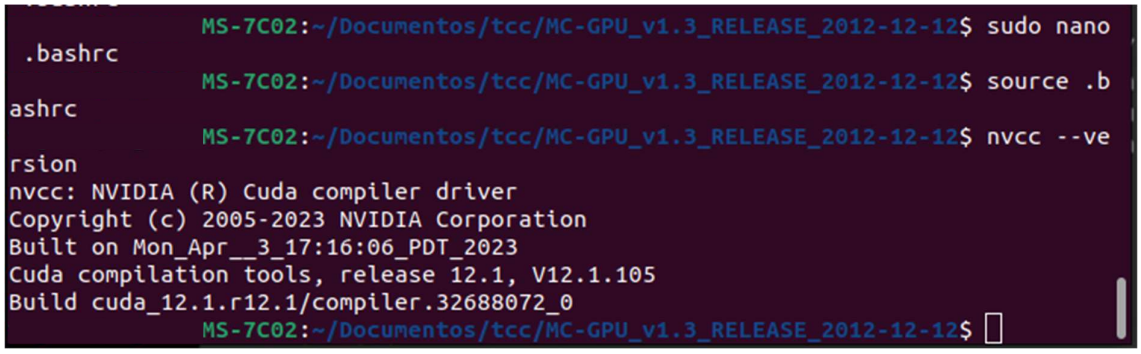


n. Após a instalação, digite o comando **sudo nano .bashrc;**

o. Dentro do arquivo que foi aberto no terminal, digite as seguintes linhas (substituindo o número da versão com segundo número pelo que foi baixado, no exemplo versão 12.1):

export PATH=$PATH:/usr/local/cuda-numero da versao com segundo numero/bin
export LD_LIBRARY_PATH=/usr/local/cuda-numero da versao com segundo numero/lib64

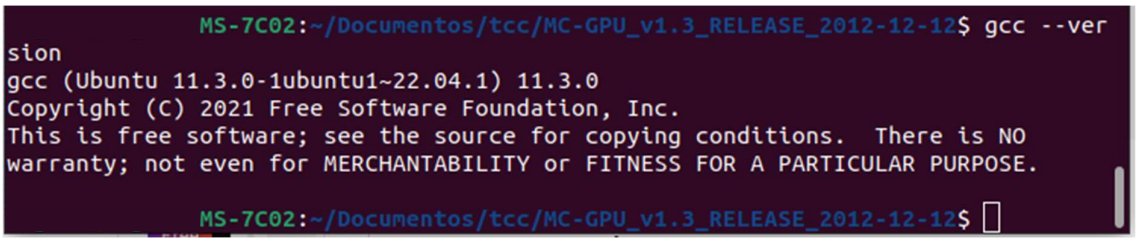
p. Aperte **Ctrl+O** para salvar o arquivo. Após salvar, aperte **Ctrl+X** para fechar o arquivo;
q. Digite **source .bashrc** e aperte Enter;
r. Digite **nvcc --version** para verificar se foi instalado com sucesso.



2.2. GNU GCC

Caso seja solicitado durante a instalação, digite a senha que utiliza para acessar o sistema operacional para continuar.

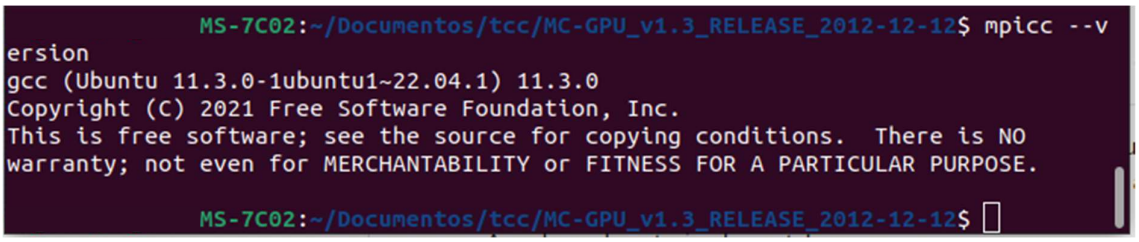
a. Abra o terminal;
b. Digite **sudo apt-get update;**
c. Digite **sudo apt-get install gcc;**
d. Após a conclusão, digite **gcc --version** para verificar se foi instalado com sucesso.



2.3. MPI

Caso seja solicitado durante a instalação, digite a senha que utiliza para acessar o sistema operacional para continuar.

a. Abra o terminal;
b. Digite **sudo apt install mpich;**
c. Após a conclusão, digite **mpicc --version** para verificar se foi instalado com sucesso.

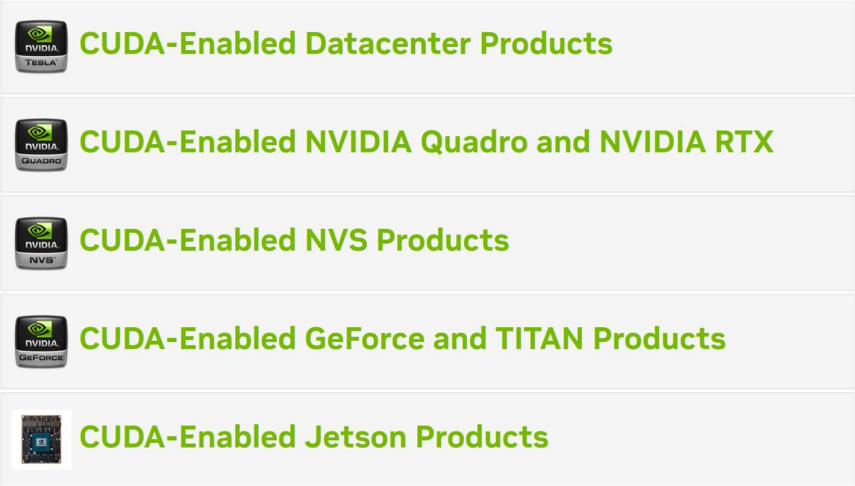


3. Configurações

3.1. Compilação da versão GPU

Obs.: Caso ocorra algum erro de reconhecimento da NVidia CUDA, utilize o comando **sudo source .bashrc** para carregar as pastas da biblioteca nas variáveis de ambiente.

a. Acesse a página https://developer.nvidia.com/cuda-gpus;
b. Clique na seção que corresponde a GPU que será utilizada (no exemplo do passo ‘c’ GTX/GeForce);



c. Na seção expandida, encontre o poder computacional da GPU que será utilizada (no exemplo GTX 1060);

GeForce GTX 1080 Ti	6.1
GeForce GTX 1080	6.1
GeForce GTX 1070 Ti	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1
GeForce GTX 1050	6.1

d. Na pasta do código fonte do SOPPE, clique com o botão direito e abra o terminal;
e. Execute o comando **sudo source .bashrc**;
f. Execute o comando abaixo para compilar o programa, substituindo nos locais indicados pelos poder computacional adquirido no passo ‘c’, sem os pontos separadores (exemplo 6.1 -> 61);

nvcc -DUSING_CUDA -DUSING_MPI SOPPE.cu -o SOPPE.x -O3 -use_fast_math -L/usr/lib/ -l. -I/usr/local/cuda/include -I/usr/local/cuda/samples/common/inc -I/usr/local/cuda/samples/shared/inc/ -I/usr/lib/x86_64-linux-gnu/openmpi/include -lmpi -lz --ptxas-options=-v -gencode=arch=compute_61,code=sm_61 -gencode=arch=compute_61,code=sm_61

g. Será gerado o arquivo **SOPPE.x** que será utilizado para realizar as simulações utilizando a GPU.

3.2. Compilação da versão CPU

a. Na pasta do código fonte do SOPPE, clique com o botão direito e abra o terminal;
b. Execute o comando abaixo para compilar o programa;

gcc -x c -O3 SOPPE.cu -o SOPPE_CPU.x -I./ -lm -lz

c. Será gerado o arquivo **SOPPE_CPU.x** que será utilizado para realizar as simulações utilizando a CPU.

4. Instruções de uso

No arquivo de entrada, existem diversos parâmetros e configurações para manipular uma simulação. No passo a passo vamos nos ater aos parâmetros necessários para executar uma simulação.

4.1. Configuração das entradas

a. Na primeira linha da seção [SECTION SIMULATION CONFIG] podemos determinar o número de histórias que serão realizadas na simulação. Caso o número seja abaixo de 100.000, ele será considerado como tempo total de simulação;

```
#[SECTION SIMULATION CONFIG]
1.0e9                          # TOTAL NUMBER OF HISTORIES, OR SIMULATION TIME IN SECONDS IF VALUE < 100000
271828182                      # RANDOM SEED (ranecu PRNG)
0                              # GPU NUMBER TO USE WHEN MPI IS NOT USED, OR TO BE AVOIDED IN MPI RUNS
128                            # GPU THREADS PER CUDA BLOCK (multiple of 32)
100                            # SIMULATED HISTORIES PER GPU THREAD
```

b. Na primeira linha da seção [SECTION SOURCE] podemos determinar o arquivo de fonte de energia da simulação;

```
#[SECTION SOURCE]
90keV.spc                      # X-RAY ENERGY SPECTRUM FILE
15.0 -45.0  15.0               # SOURCE POSITION: X Y Z [cm]
0.0    1.0   0.0               # SOURCE DIRECTION COSINES: U V W
42.0  39.0                     # POLAR AND AZIMUTHAL APERTURES FOR THE FAN BEAM [degrees] (input negative to automatically cover the whole detector)
```

c. Nas seções [SECTION VOXELIZED GEOMETRY FILE] e [SECTION MATERIAL FILE LIST] determinamos o arquivo de geometria e os arquivos de materiais que serão utilizados nas simulações. A aplicação também aceita esses arquivos no formato do PENELOPE. Em relação aos materiais, o número máximo que se pode colocar são 15;

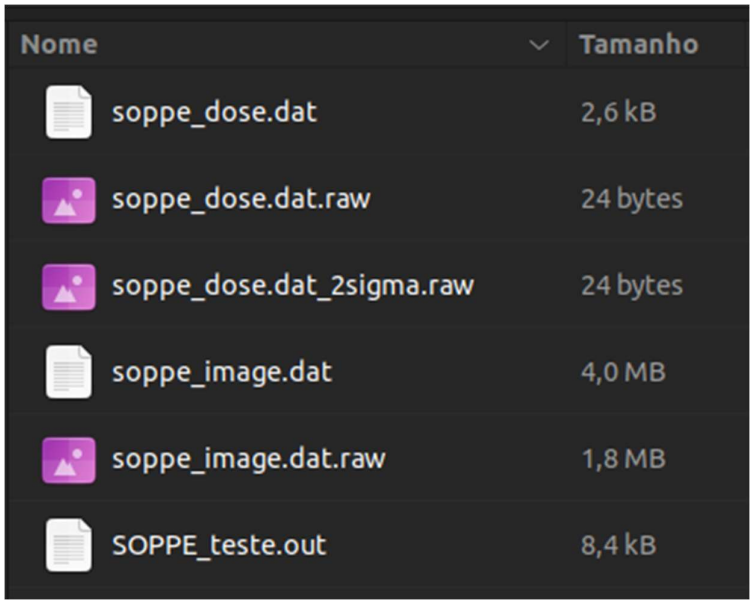
```
#[SECTION VOXELIZED GEOMETRY FILE]
6voxels.vox                    # VOXEL GEOMETRY FILE (penEasy 2008 format; .gz accepted)

#[SECTION MATERIAL FILE LIST]
./material_files/water__5-120keV.mcgpu.gz          #  1st MATERIAL FILE (.gz accepted)
./material_files/bone_ICRP110__5-120keV.mcgpu.gz   #  2nd MATERIAL FILE
```

4.2. Executando uma simulação

a. Na pasta do código fonte do SOPPE, clique com o botão direito e abra o terminal;
b. Execute o comando abaixo para iniciar uma simulação, onde:
    i. **SOPPE.x** é o arquivo compilado, podendo ser tanto o para CPU (3.2) quanto GPU (3.1);
    ii. **SOPPE.in** é o arquivo de entrada com as configurações importantes para simulação (4.1);
    iii. **SOPPE.out** é o arquivo de saída com os resultados.

./SOPPE.x SOPPE.in | tee SOPPE.out

c. A simulação será iniciada, com seu tempo variando de acordo com as configurações que foram fornecidas;
d. Ao término da simulação, serão gerados o arquivo de saída (*.out*) e os arquivos *.dat* e *.raw* contendo as imagens geradas e suas doses;

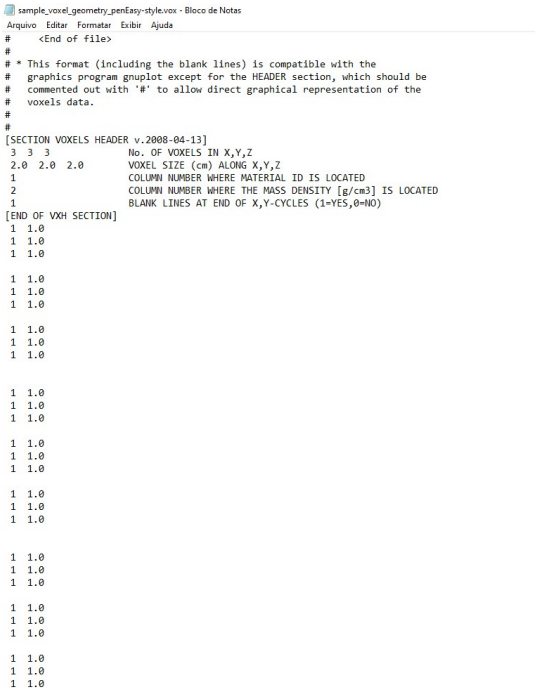


e. No arquivo de saída, são descritas várias informações sobre a simulação. No final do arquivo é informado o tempo total que levou a simulação;

```
-- SIMULATION FINISHED!


   ****** TOTAL SIMULATION PERFORMANCE (including initialization and reporting) ******

      >>> Execution time including initialization, transport and report: 8346.136 s.
      >>> Time spent in the Monte Carlo transport only: 8344.685 s.
      >>> Time spent in initialization, reporting and clean up: 1.450 s.

      >>> Total number of simulated x rays:  100000000000
      >>> Total speed (using 1 thread, including initialization time) [x-rays/s]:  11981593.10
```

5. Interface gráfica

Atualmente os pesquisadores utilizam o bloco de notas para configurar as simulações, conforme figura abaixo:



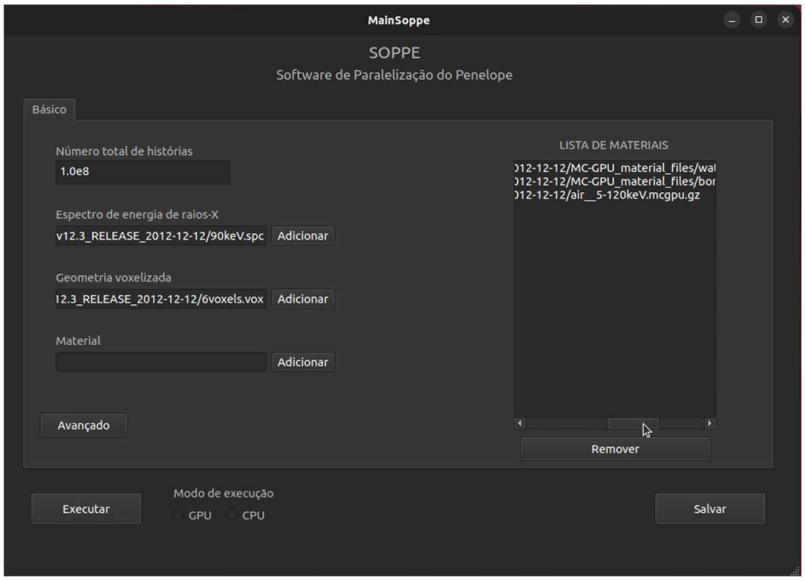
sample_voxel_geometry_penEasy-style.vox - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

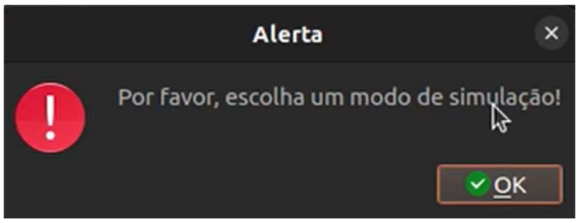
```
#     <End of file>
#
# * This format (including the blank lines) is compatible with the
#   graphics program gnuplot except for the HEADER section, which should be
#   commented out with '#' to allow direct graphical representation of the
#   voxels data.
#
#
[SECTION VOXELS HEADER v.2008-04-13]
 3  3  3              No. OF VOXELS IN X,Y,Z
 2.0  2.0  2.0        VOXEL SIZE (cm) ALONG X,Y,Z
 1                    COLUMN NUMBER WHERE MATERIAL ID IS LOCATED
 2                    COLUMN NUMBER WHERE THE MASS DENSITY [g/cm3] IS LOCATED
 1                    BLANK LINES AT END OF X,Y-CYCLES (1=YES,0=NO)
[END OF VXH SECTION]
 1  1.0
 1  1.0
 1  1.0

 1  1.0
 1  1.0
 1  1.0

 1  1.0
 1  1.0
 1  1.0


 1  1.0
 1  1.0
 1  1.0

 1  1.0
 1  1.0
 1  1.0

 1  1.0
 1  1.0
 1  1.0


 1  1.0
 1  1.0
 1  1.0

 1  1.0
 1  1.0
 1  1.0

 1  1.0
 1  1.0
 1  1.0
```

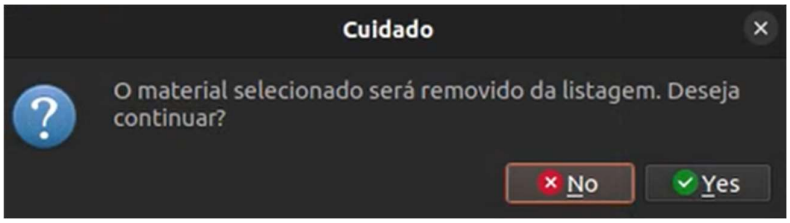
Essa falta de uma interface amigável, dificulta a configuração e também o rápido diagnóstico quando apresentado algum problema.

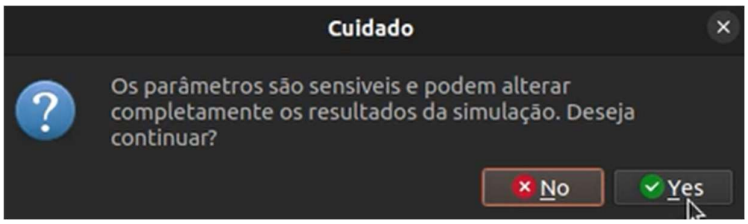
A interface do SOPPE foi criada justamente para ajudar o pesquisador nessa árdua tarefa, seja de configurar ou identificar erros, no modo básico o pesquisador pode escolher desde o número de histórias, passando pelo o espectro de energia até a lista de materiais, conforme figura abaixo:



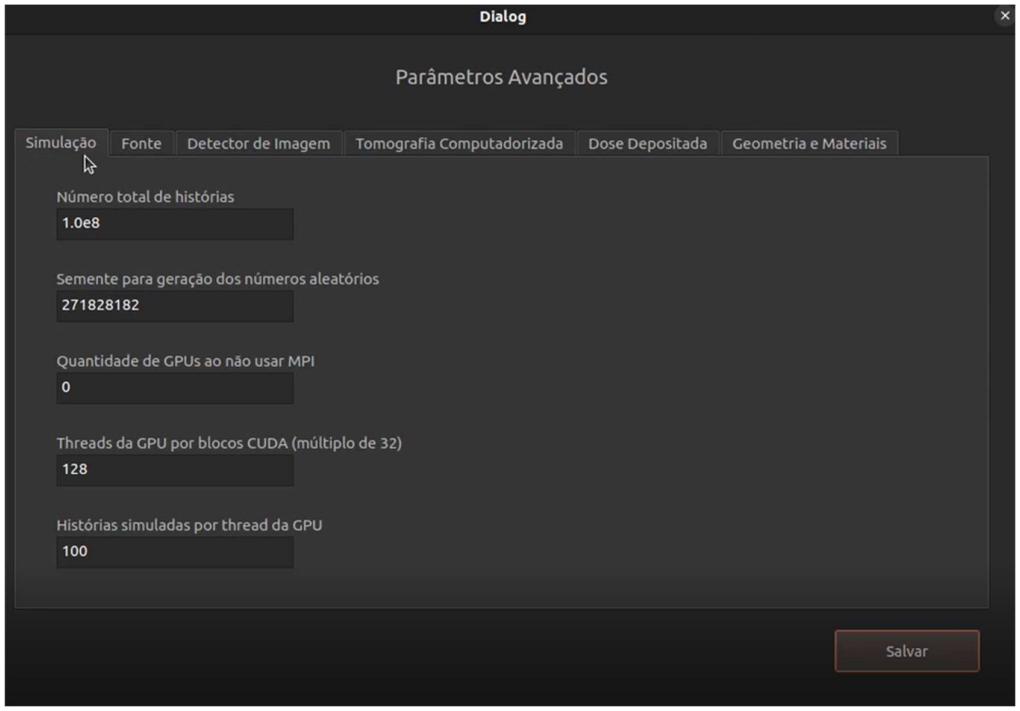
A interface do SOPPE ainda possui alertas para auxiliar o pesquisador, conforme as figuras abaixo:

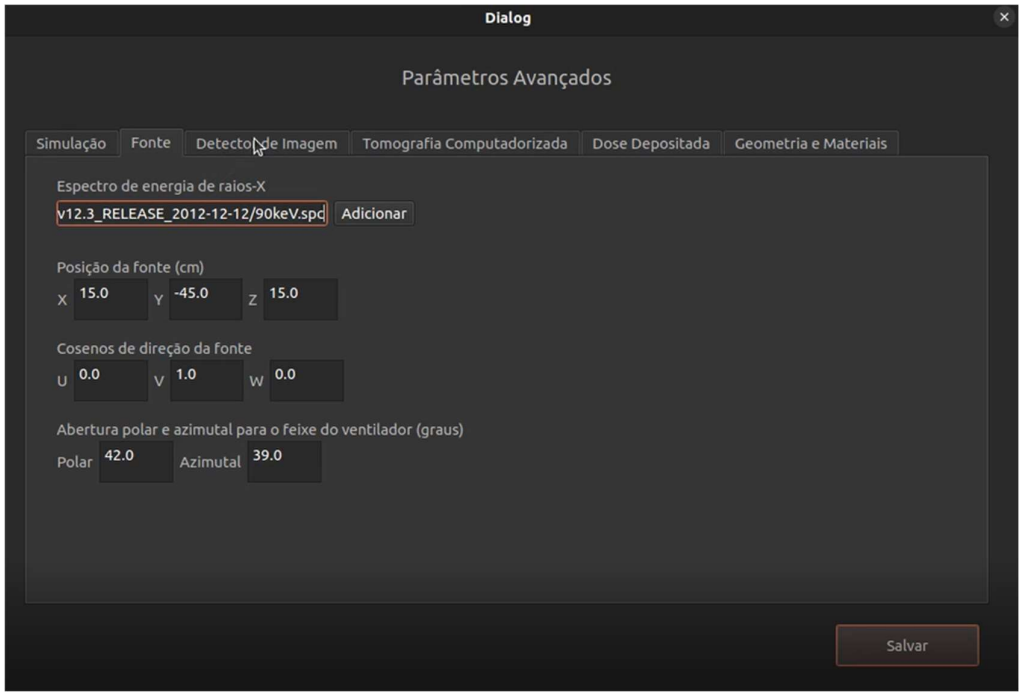


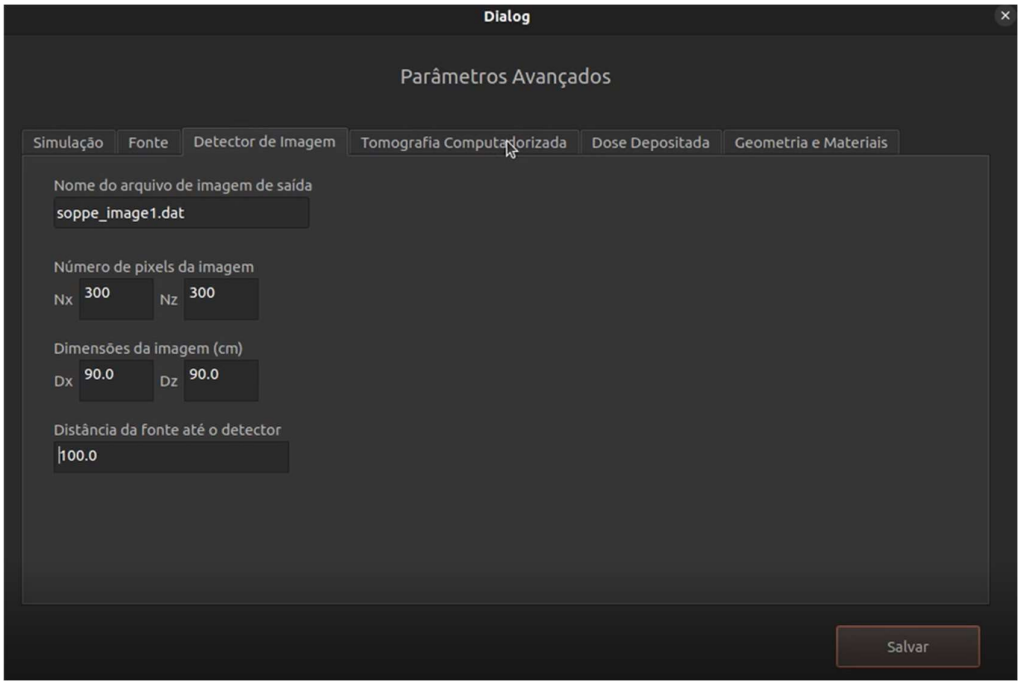


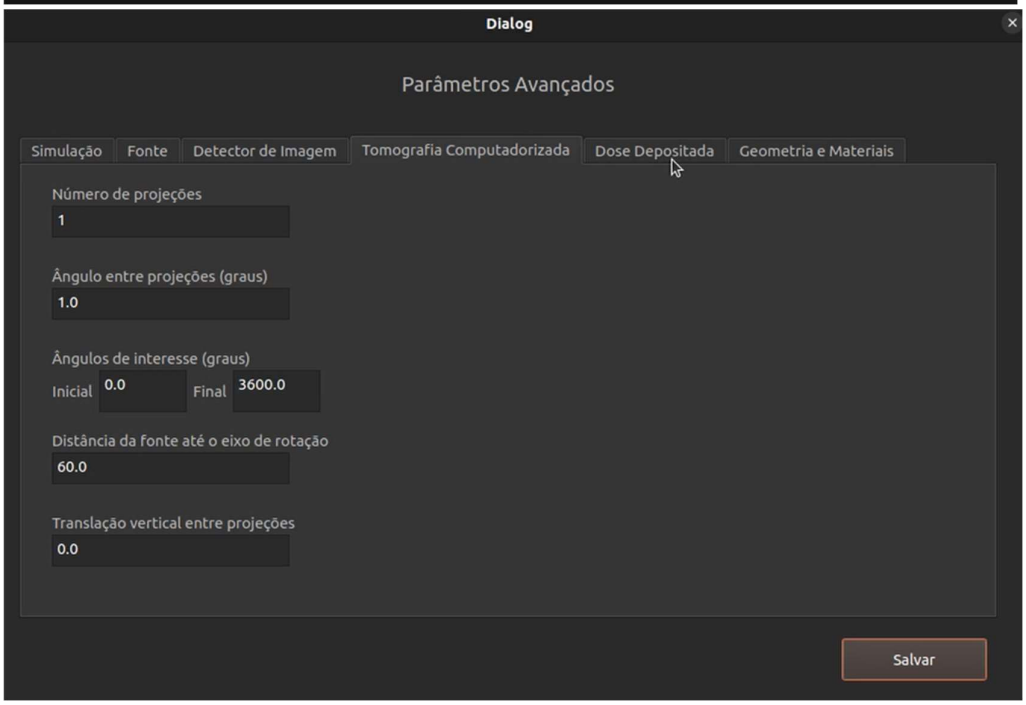


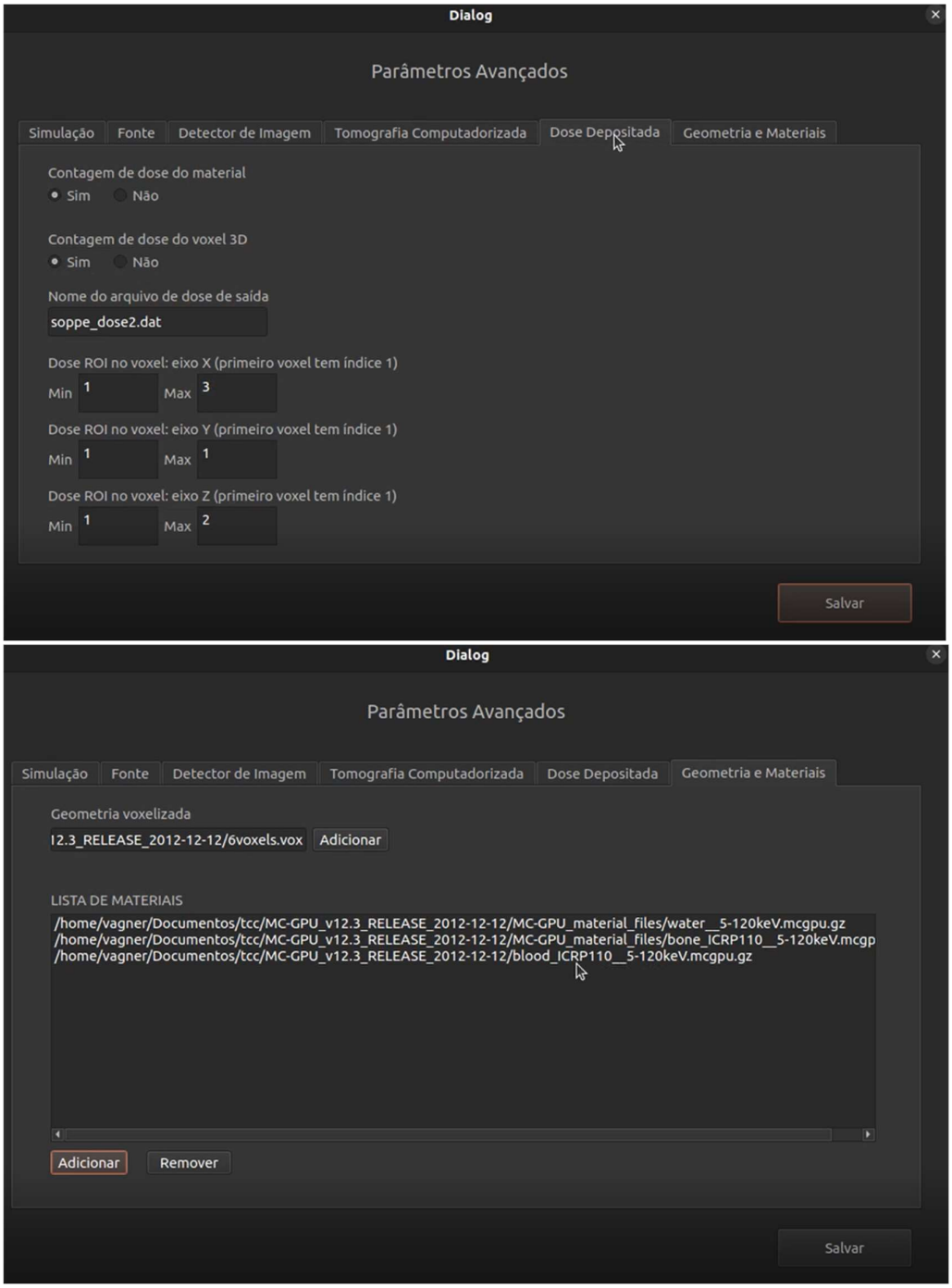
Já no modo avançado, é possível utilizar uma alta gama de configurações, dividida em abas, conforme as telas a seguir:


Dialog
Parâmetros Avançados
Simulação
Fonte
Detector de Imagem
Tomografia Computadorizada
Dose Depositada
Geometria e Materiais
Número total de histórias
1.0e8
Semente para geração dos números aleatórios
271828182
Quantidade de GPUs ao não usar MPI
0
Threads da GPU por blocos CUDA (múltiplo de 32)
128
Histórias simuladas por thread da GPU
100
Salvar

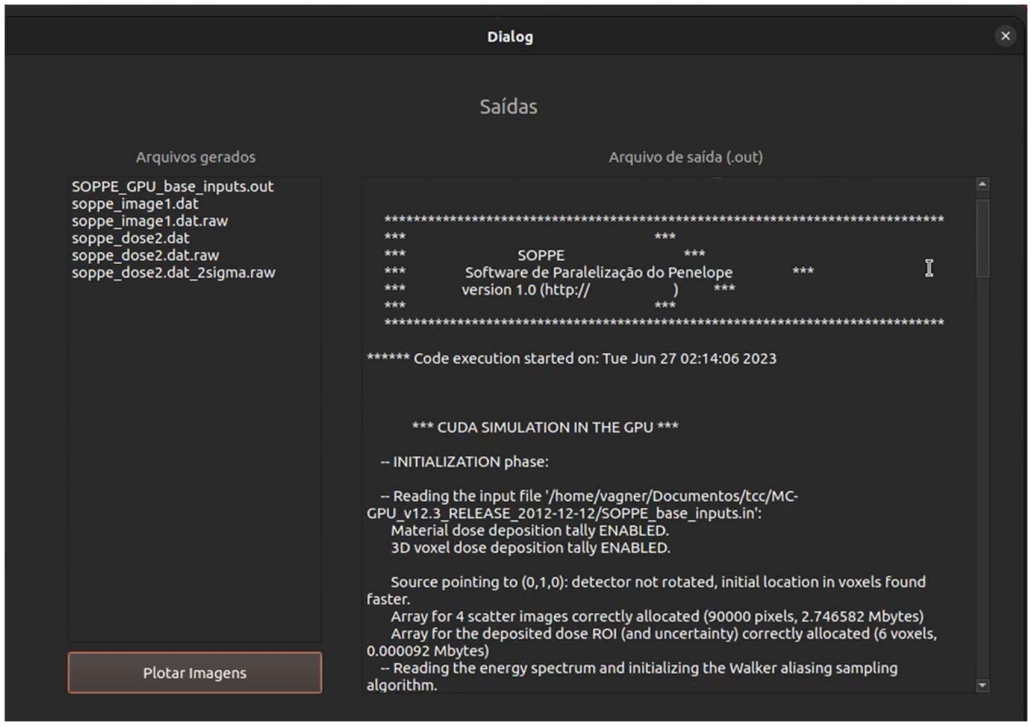

Dialog
Parâmetros Avançados
Simulação
Fonte
Detector de Imagem
Tomografia Computadorizada
Dose Depositada
Geometria e Materiais
Espectro de energia de raios-X
v12.3_RELEASE_2012-12-12/90keV.spc
Adicionar
Posição da fonte (cm)
X 15.0
Y -45.0
Z 15.0
Cosenos de direção da fonte
U 0.0
V 1.0
W 0.0
Abertura polar e azimutal para o feixe do ventilador (graus)
Polar 42.0
Azimutal 39.0
Salvar

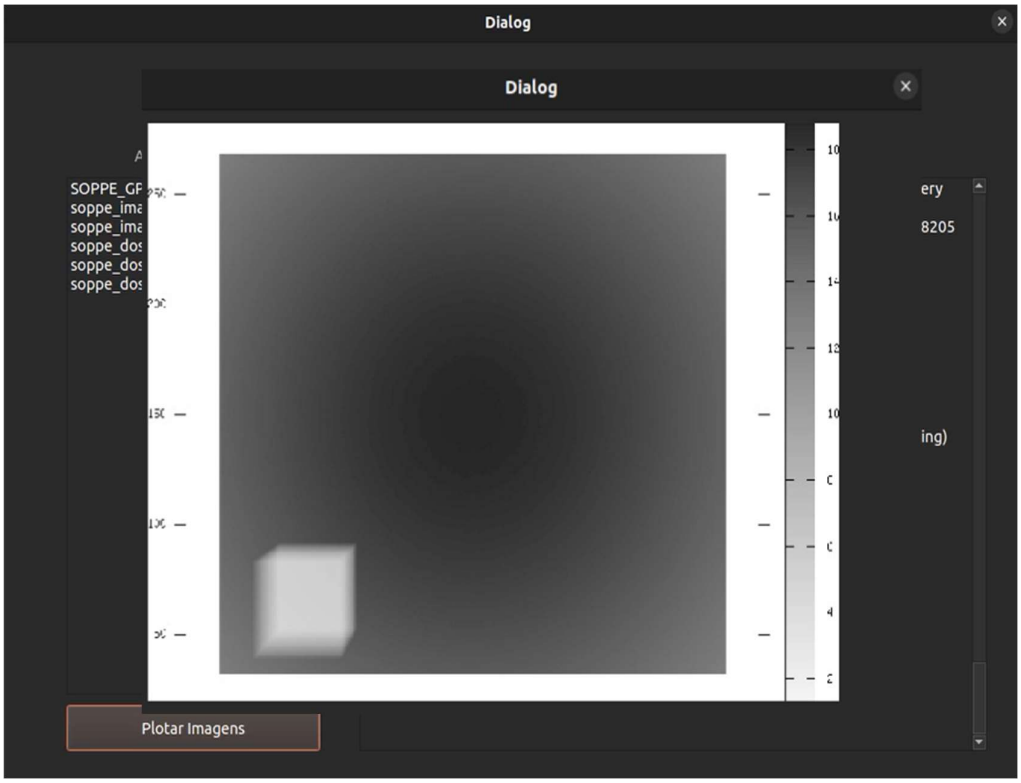

Dialog
Parâmetros Avançados
Simulação
Fonte
Detector de Imagem
Tomografia Computadorizada
Dose Depositada
Geometria e Materiais
Nome do arquivo de imagem de saída
soppe_image1.dat
Número de pixels da imagem
Nx 300
Nz 300
Dimensões da imagem (cm)
Dx 90.0
Dz 90.0
Distância da fonte até o detector
100.0
Salvar


Dialog
Parâmetros Avançados
Simulação
Fonte
Detector de Imagem
Tomografia Computadorizada
Dose Depositada
Geometria e Materiais
Número de projeções
1
Ângulo entre projeções (graus)
1.0
Ângulos de interesse (graus)
Inicial 0.0
Final 3600.0
Distância da fonte até o eixo de rotação
60.0
Translação vertical entre projeções
0.0
Salvar



Após a configuração básica ou avançada, o pesquisador executa simulação selecionando se deseja executar no modo GPU, utilizando a placa gráfica ou CPU, dispensando o uso da paralelização, e então recebe o resultado as simulação, por fim, tendo a possibilidade de plotar a imagem, conforme telas a seguir:


Dialog
Saídas
Arquivos gerados
SOPPE_GPU_base_inputs.out
soppe_image1.dat
soppe_image1.dat.raw
soppe_dose2.dat
soppe_dose2.dat.raw
soppe_dose2.dat_2sigma.raw
Plotar Imagens
Arquivo de saída (.out)
****
*** ***
*** SOPPE ***
*** Software de Paralelização do Penelope *** I
*** version 1.0 (http://) ***
*** ***
****
****** Code execution started on: Tue Jun 27 02:14:06 2023
*** CUDA SIMULATION IN THE GPU ***
-- INITIALIZATION phase:
-- Reading the input file '/home/vagner/Documentos/tcc/MC-GPU_v12.3_RELEASE_2012-12-12/SOPPE_base_inputs.in':
Material dose deposition tally ENABLED.
3D voxel dose deposition tally ENABLED.
Source pointing to (0,1,0): detector not rotated, initial location in voxels found faster.
Array for 4 scatter images correctly allocated (90000 pixels, 2.746582 Mbytes)
Array for the deposited dose ROI (and uncertainty) correctly allocated (6 voxels, 0.000092 Mbytes)
-- Reading the energy spectrum and initializing the Walker aliasing sampling algorithm.


Dialog
Dialog
Plotar Imagens

Referências Bibliográficas

[1]Shi M, et al. GPU-accelerated Monte Carlo simulation of MV-CBCT. Phys Med Biol. 2020 Dec 2;65(23):235042. doi: 10.1088/1361-6560/abaeba. PMID: 33263311.

[2]Miras H, Jiménez R, Miras C, Gomà C. CloudMC: a cloud computing application for Monte Carlo simulation. Phys Med Biol. 2013 Apr 21;58(8):N125-33. doi: 10.1088/0031-9155/58/8/N125. Epub 2013 Mar 21. PMID: 23514937.

[3]NEA (2019), PENELOPE 2018: A code system for Monte Carlo simulation of electron and photon transport: Workshop Proceedings, Barcelona, Spain, 28 January – 1 February 2019, OECD Publishing, Paris, https://doi.org/10.1787/32da5043-en.

[4]Koger B, Kirkby C. Optimization of photon beam energies in gold nanoparticle enhanced arc radiation therapy using Monte Carlo methods. Phys Med Biol. 2016 Dec 21;61(24):8839-8853. doi: 10.1088/1361-6560/61/24/8839. Epub 2016 Dec 2. PMID: 27910829.