

MANUAL

SOPPE

Software de Paralelização
do Penelope

1. Introdução

O SOPPE (Software de Paralelização do Penelope) é um código de simulação que utiliza 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). 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 uma geometria voxelizada.

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 utilizando 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.

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.

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

Operating System	<input checked="" type="radio"/> Linux	<input type="radio"/> Windows					
Architecture	<input checked="" type="radio"/> x86_64	<input type="radio"/> ppc64le	<input type="radio"/> arm64-sbsa	<input type="radio"/> aarch64-jetson			
Distribution	<input type="radio"/> CentOS	<input type="radio"/> Debian	<input type="radio"/> Fedora	<input type="radio"/> KylinOS	<input type="radio"/> OpenSUSE	<input type="radio"/> RHEL	<input type="radio"/> Rocky
	<input type="radio"/> SLES	<input checked="" type="radio"/> Ubuntu	<input type="radio"/> WSL-Ubuntu				
Version	<input type="radio"/> 18.04	<input type="radio"/> 20.04	<input checked="" type="radio"/> 22.04				
Installer Type	<input type="radio"/> deb (local)	<input type="radio"/> deb (network)	<input checked="" type="radio"/> runfile (local)				

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

Download Installer for Linux Ubuntu 22.04 x86_64

The base installer is available for download below.

➤Base Installer

Installation Instructions:

```
$ wget https://developer.download.nvidia.com/compute/cuda/12.1.1/local_installers/cuda_12.1.1_530.30.02_linux.run
$ sudo sh cuda_12.1.1_530.30.02_linux.run
```

- 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);

```
xserver-xorg-video-nvidia-515-server/jammy-updates,jammy-security 515.105.01-0ubuntu0.22.04.1 amd64
NVIDIA binary Xorg driver

xserver-xorg-video-nvidia-525/jammy-updates,jammy-security 525.105.17-0ubuntu0.22.04.1 amd64
NVIDIA binary Xorg driver

xserver-xorg-video-nvidia-525-server/jammy-updates,jammy-security 525.105.17-0ubuntu0.22.04.1 amd64
NVIDIA binary Xorg driver

xserver-xorg-video-nvidia-530/jammy-updates,jammy-security 530.41.03-0ubuntu0.22.04.2 amd64
NVIDIA binary Xorg driver

vagner@vagner-MS-7C02:~/Documentos/tcc/MC-GPU_v1.3_RELEASE_2012-12-12$ sudo apt
install nvidia-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;

```
End User License Agreement
-----

NVIDIA Software License Agreement and CUDA Supplement to
Software License Agreement. Last updated: October 8, 2021

The CUDA Toolkit End User License Agreement applies to the
NVIDIA CUDA Toolkit, the NVIDIA CUDA Samples, the NVIDIA
Display Driver, NVIDIA Nsight tools (Visual Studio Edition),
and the associated documentation on CUDA APIs, programming
model and development tools. If you do not agree with the
terms and conditions of the license agreement, then do not
download or use the software.

Last updated: October 8, 2021.

Preface
-----

Do you accept the above EULA? (accept/decline/quit):
accept
```

- 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;

```
CUDA Installer
- [ ] Driver
  [ ] 530.30.02
+ [X] CUDA Toolkit 12.1
  [X] CUDA Demo Suite 12.1
  [X] CUDA Documentation 12.1
- [ ] Kernel Objects
  [ ] nvidia-fs
Options
Install

Up/Down: Move | Left/Right: Expand | 'Enter': Select | 'A': Advanced options
```

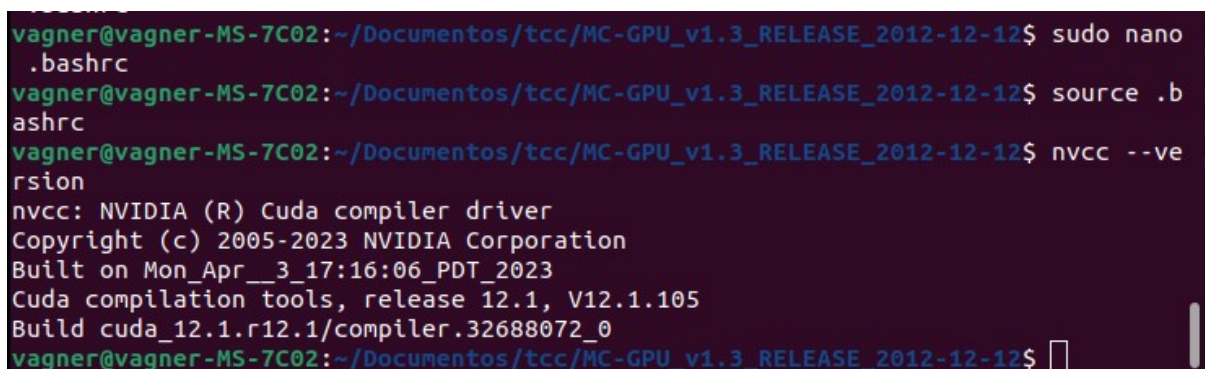
- n. Após a instalação, digite o comando **sudo nano .bashrc**;
- o. Dentro do arquivo que foi aberto no terminal, digite as seguintes linhas (substituindo o número da versão com segundo número pelo que foi baixado, no exemplo versão 12.1):

```
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
```

A screenshot of a terminal window showing the nano text editor editing the .bashrc file. The editor's title bar shows 'GNU nano 6.2' and '.bashrc'. The content of the file includes two export statements: 'export PATH=\$PATH:/usr/local/cuda-12.1/bin' and 'export LD_LIBRARY_PATH=/usr/local/cuda-12.1/lib64'. The bottom status bar of the nano editor displays various keyboard shortcuts: ^G Ajuda, ^O Gravar, ^W Onde está?, ^K Recortar, ^T Executar, ^C Local, ^X Sair, ^R Ler o arq, ^_ Substituir, ^U Colar, ^J Justificar, and ^_/ Ir p/ linha.

```
GNU nano 6.2 .bashrc
export PATH=$PATH:/usr/local/cuda-12.1/bin
export LD_LIBRARY_PATH=/usr/local/cuda-12.1/lib64
^G Ajuda ^O Gravar ^W Onde está? ^K Recortar ^T Executar ^C Local
^X Sair ^R Ler o arq ^_ Substituir ^U Colar ^J Justificar ^_/ Ir p/ linha
```

- 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.

A screenshot of a terminal window showing a series of commands and their outputs. The user runs 'sudo nano .bashrc', then 'source .bashrc', and finally 'nvcc --version'. The output of 'nvcc --version' shows the NVIDIA CUDA compiler driver version 12.1, built on April 3, 2023.

```
vagner@vagner-MS-7C02:~/Documentos/tcc/MC-GPU_v1.3_RELEASE_2012-12-12$ sudo nano
.bashrc
vagner@vagner-MS-7C02:~/Documentos/tcc/MC-GPU_v1.3_RELEASE_2012-12-12$ source .b
ashrc
vagner@vagner-MS-7C02:~/Documentos/tcc/MC-GPU_v1.3_RELEASE_2012-12-12$ nvcc --ve
rsion
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Mon_Apr__3_17:16:06_PDT_2023
Cuda compilation tools, release 12.1, V12.1.105
Build cuda_12.1.r12.1/compiler.32688072_0
vagner@vagner-MS-7C02:~/Documentos/tcc/MC-GPU_v1.3_RELEASE_2012-12-12$
```

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.

```
vagner@vagner-MS-7C02:~/Documentos/tcc/MC-GPU_v1.3_RELEASE_2012-12-12$ gcc --version
gcc (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

vagner@vagner-MS-7C02:~/Documentos/tcc/MC-GPU_v1.3_RELEASE_2012-12-12$
```

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.

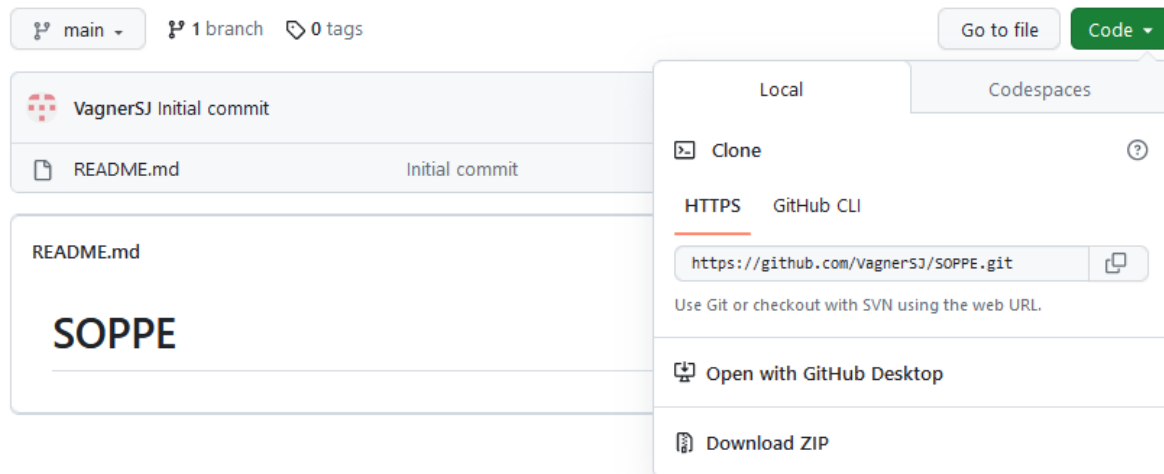
```
vagner@vagner-MS-7C02:~/Documentos/tcc/MC-GPU_v1.3_RELEASE_2012-12-12$ mpicc --version
gcc (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

vagner@vagner-MS-7C02:~/Documentos/tcc/MC-GPU_v1.3_RELEASE_2012-12-12$
```

2.4. Arquivos da aplicação

Após instalar os pré-requisitos, podemos baixar os arquivos da aplicação. Eles podem ser encontrados no repositório no GitHub da aplicação.

- a. Acesse a página <https://github.com/VagnerSJ/SOPPE>;
- b. Clique no botão **Code** e em seguida baixe o arquivo ZIP;



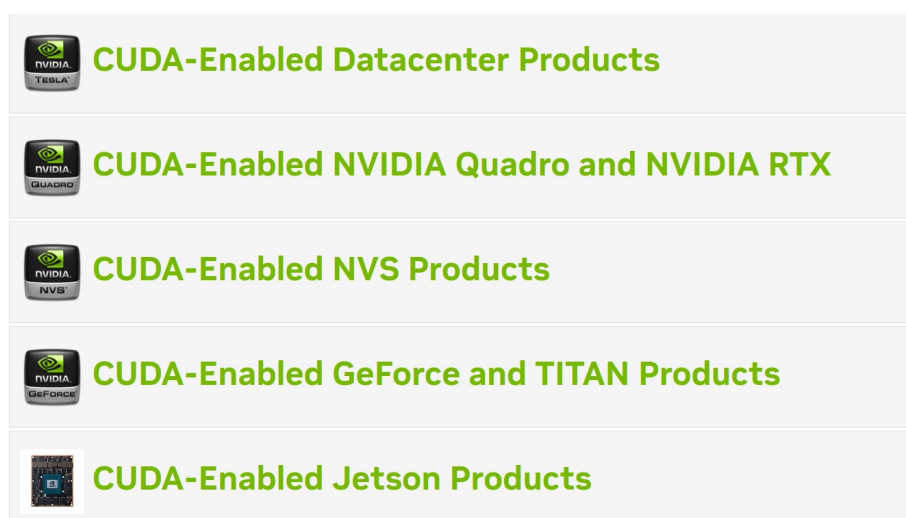
- c. Descompacte o arquivo ZIP baixado. Nele estão todos os arquivos necessários para o funcionamento do SOPPE, incluindo alguns exemplos de demonstração.

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/ -I. -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 -  
Impi -lz --ptxas-options=-v -generate=arch=compute_61,code=sm_61 -  
generate=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;

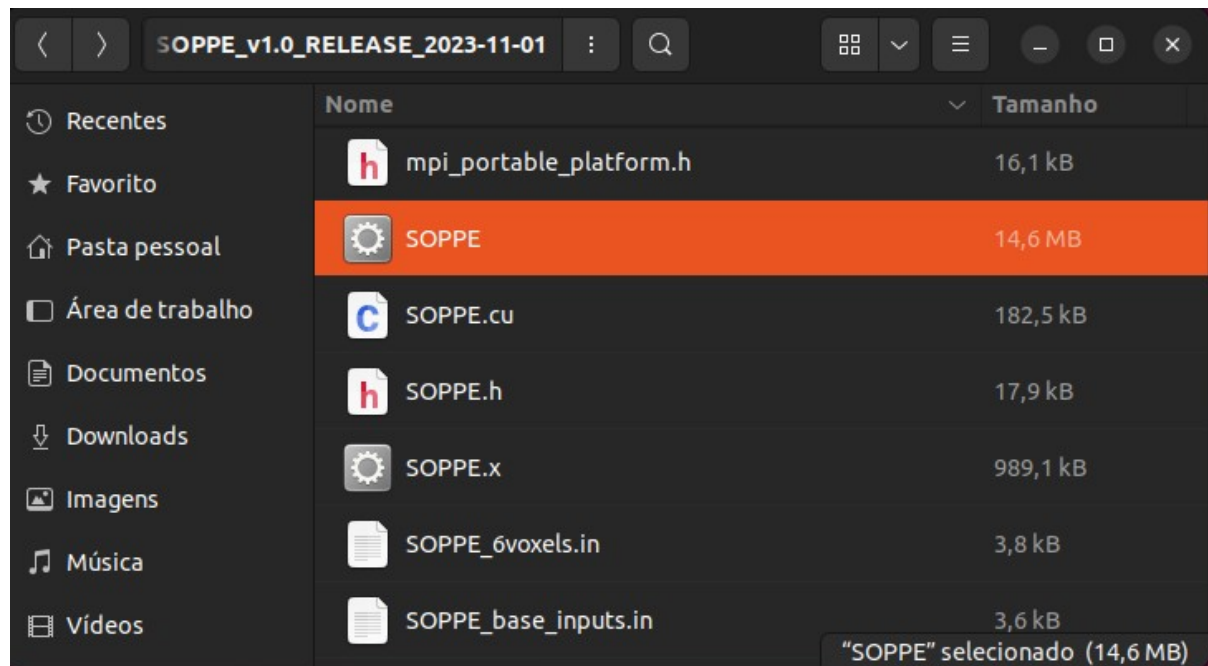
```
gcc -x c -O3 SOPPE.cu -o SOPPE_CPU.x -I./ -lm -lz
```

- 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.

Obs.: caso tenha realizado alguma modificação no código fonte, copie o novo executável gerado para a pasta de execução do SOPPE.



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;

Nome	Tamanho
 soppe_dose.dat	2,6 kB
 soppe_dose.dat.raw	24 bytes
 soppe_dose.dat_2sigma.raw	24 bytes
 soppe_image.dat	4,0 MB
 soppe_image.dat.raw	1,8 MB
 SOPPE_teste.out	8,4 kB

- 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
```