Group6_BENG280A_Project_2

Contributor: Yang Han (yah015@ucsd.edu), Shadee Hemaidan (shemaidan@ucsd.edu), Tsung-Han Lu (tsl012@ucsd.edu)

Diffusion MRI (dMRI) simulation code used for Project 2 in BENG 280A (FA25). The repository contains 1D Monte-Carlo diffusion simulations under PGSE (Pulsed Gradient Spin Echo) sequences, with CPU, GPU and notebook implementations and optional restricted diffusion (reflecting boundaries) and parallelization over gradient amplitudes.

Link to our Github Repo

Project Presentation

Environment Setup

The core dependencies used across the scripts/notebooks are:

- `python` (3.9+ recommended)
- `numpy`
- `matplotlib`
- `joblib` (for parallelization in the notebook)
- `cupy` (for GPU-accelerated simulations in `1D_GPU.py`, requires CUDA-capable GPU)

You can install them either via Conda (recommended) or via pip.

Conda environment

```
conda create -n beng280a_dMRI python=3.10 numpy matplotlib joblib
conda activate beng280a_dMRI

# Install CuPy matching your CUDA version, for example (CUDA 12):
conda install -c conda-forge cupy
```

If you do not have a compatible GPU or do not plan to use the GPU script, you can skip installing `cupy`.

pip installation

From inside your virtual environment:

```
pip install numpy matplotlib joblib

# Optional: GPU support with CuPy (choose wheel matching your CUDA):
pip install cupy-cuda12x  # example for CUDA 12.x
```

Files and Their Purpose

- `1D_dMRI.ipynb`
  Jupyter notebook that walks through a 1D Monte-Carlo diffusion simulation under a gradient. It:
  - Defines the PGSE gradient waveform and simulates Brownian motion of many spins (particles) in 1D.
  - Assumes uniform diffusivity across the whole space.
  - Computes the dMRI signal from the sum of accumulated phase and compares it to the Stejskal–Tanner theoretical signal calculation with pure diffusivity.
  - Visualizes position histograms, mean-squared displacement over time, gradient waveforms, and signal vs. b-value.
  - Extends the model to restricted diffusion (reflecting boundaries) to mimic cellular confinement. Explores how restricted diffusion affect the signal.
  - Uses `joblib` to parallelize simulations with higher particle counts across different gradient amplitudes for faster exploration.

- `1D_CPU.py`
  Stand-alone CPU implementation for large-scale 1D dMRI simulations:
  - Builds a PGSE gradient waveform (`make_pgse_gradient`).
  - Simulates Brownian motion of a large number of spins in batches (`simulate_batch_cpu`), with optional reflecting boundaries.
  - Accumulates phase over time and computes the average complex signal magnitude.
  - Computes the corresponding Stejskal–Tanner theoretical signal for comparison and prints summary statistics and relative error.
  - Intended for large simulations on CPU only (no GPU required).

- `1D_GPU.py`
  GPU-accelerated version of the 1D Monte-Carlo dMRI simulation using CuPy:
  - Constructs the PGSE gradient on CPU and transfers it to GPU.
  - Simulates diffusion and phase accumulation on the GPU in batches (`simulate_batch_gpu`).
  - Supports optional reflecting boundaries to model restricted diffusion.
  - Aggregates results across batches in `run_massive_simulation`, allowing up to ~10^{10} spins (depending on VRAM and runtime).
  - Prints simulated vs. theoretical signal and b-value summary.

- `1D_Diff_Gradient_v3.ipynb`
  - Constructed 1D space with different diffusivity for each bin. Center bin has the highest diffusivity. Bins farther from center have lower diffusivity. All particles still start at the center of bin.
  - Simulates particle diffusion and phase accumulationin similar fashion compared to `1D_dMRI.ipynb`
  - Creates animation to visualize the diffusion and phase accumulation process under uniform and changing diffusivity of a few particles.

- `2D_dMRI_v1.ipynb`
  - Initial scale up of Monte-Carlo diffusion to 2D. The simulation space has anisotropy that particles diffuses faster along x-axis but slower along y-axis

    - Same gradient design is applied, despite at different angles (tunable parameter) to explore how apparent diffusivity and signal changes.
    - Offers the option to simulate restricted diffusion in a boxed x and y space. Compares signal to ST equation calculation.

- `2D_dMRI_v2.py`

    - Runs a 2D anisotropic diffusion simulation where diffusion along x is faster than along y, under a rotating PGSE gradient. The core diffusion idea is the same as previously.
    - Computes the apparent/effective diffusivity as a function of gradient angle and compares Monte-Carlo estimates with the Stejskal–Tanner prediction.
    - Produces a two-panel figure: a schematic of the anisotropic diffusion tensor and gradient direction, and a plot of effective diffusivity vs. gradient angle.

How to Run

- To explore and reproduce figures interactively, open the notebooks (e.g. `1D_dMRI.ipynb`) in Jupyter Lab / VS Code and run cells from top to bottom.

- To run the CPU script from the command line:

```
python 1D_CPU.py
```

- To run the GPU script (requires configured CUDA + CuPy):

```
python 1D_GPU.py
```

Adjust parameters (e.g., number of spins, batch size, gradient amplitude, presence of boundaries) inside each script/notebook to match your experiment or hardware limitations.