The Journal of Open Source Software

Disimpy: A massively parallel Monte Carlo simulator for generating diffusion-weighted MRI data in Python

Leevi Kerkelä[1], Fabio Nery[1], Matt G. Hall[2,1], and Chris A. Clark[1]

1 UCL Great Ormond Street Institute of Child Health, University College London, London, United Kingdom **2** National Physical Laboratory, Teddington, United Kingdom

DOI: 10.21105/joss.02527

Software

- Review
- Repository
- Archive

Editor: Kevin M. Moerman

Reviewers:

- @DARSakthi
- @grlee77
- @ritagnunes

Submitted: 11 July 2020
Published: 26 August 2020

Summary

Disimpy is a simulator for generating diffusion-weighted magnetic resonance imaging (dMRI) data that is useful in the development and validation of new methods for data acquisition and analysis. Diffusion of water is modelled as an ensemble of random walkers whose trajectories are generated on an Nvidia (Nvidia Corporation, Santa Clara, California, United States) CUDA-capable (Nickolls, Buck, Garland, & Skadron, 2008) graphical processing unit (GPU). The massive parallelization results in a significant performance gain, enabling simulation experiments to be performed on standard laptop and desktop computers. Disimpy is written in Python (Python Software Foundation), making its source code very approachable and easily extensible.

Statement of need

Since the diffusion of water in biological tissues is restricted by microscopic obstacles such as cell organelles, myelin, and macromolecules, dMRI enables the study of tissue microstructure *in vivo* by probing the displacements of water molecules (Behrens & Johansen-Berg, 2009). It has become a standard tool in neuroscience (Assaf, Johansen-Berg, & Thiebaut de Schotten, 2019), and a large number of data acquisition and analysis methods have been developed to tackle the difficult inverse problem of inferring microstructural properties of tissue from the dMRI signal (Novikov, Fieremans, Jespersen, & Kiselev, 2019).

Simulations have played an important role in the development of the field because they do not require the use of expensive scanner time and they provide a powerful tool for investigating the accuracy and precision of new methods, e.g., (Tournier, Calamante, & Connelly, 2007). Generally, dMRI simulations are based on modelling diffusion inside some geometry to obtain a solution to the diffusion equation, e.g., (Ianuș, Alexander, & Drobnjak, 2016; Li et al., 2019), or modelling diffusion using a more generalizable Monte Carlo approach, e.g., (Hall & Alexander, 2009; Rafael-Patino et al., 2020). The Monte Carlo method enables the use of complex and realistic tissue microstructure models, e.g., (Callaghan, Alexander, Palombo, & Zhang, 2020), and can be significantly accelerated using GPU computing, e.g., (Nguyen, Hernández-Garzón, & Valette, 2018).

Here, we present Disimpy, a GPU-accelerated dMRI simulator that enables a large amount of synthetic data to be generated on standard desktop and laptop computers without needing to access high performance computing clusters. Disimpy is written in Python, making its source code very approachable to researchers with no prior experience in GPU computing.

Features

Disimpy uses efficient numerical methods from Numpy (Van der Walt, Colbert, & Varoquaux, 2011) and Scipy (Virtanen et al., 2020). Numba (Lam, Pitrou, & Seibert, 2015) is used to compile Python code into CUDA kernels and device functions (Nickolls et al., 2008) which are executed on the GPU. The random walker steps are generated in a massively parallel fashion on individual threads of one-dimensional CUDA blocks, resulting in a performance gain of over an order of magnitude when compared to Camino (Hall & Alexander, 2009), a popular dMRI simulator written in Java (Figure 1). Given that random walker Monte Carlo dMRI simulations require at least 10^4 random walkers for sufficient convergence (Hall & Alexander, 2009), it is important that Disimpy's runtime does not linearly depend on the number of random walkers until the number of walkers is in the thousands or tens of thousands, depending on the GPU.

Diffusion can be simulated without restrictions, inside analytically defined geometries (cylinders, spheres, ellipsoids), and in arbitrary geometries defined by triangular meshes (Hall, Nedjati-Gilani, & Alexander, 2017; Panagiotaki et al., 2010) (Figure 2). Importantly, the random walk model of diffusion is able to capture time-dependent diffusion.

Disimpy supports arbitrary diffusion encoding gradient sequences, such as those used in conventional pulsed field gradient experiments (Stejskal & Tanner, 1965) as well as the recently developed q-space trajectory encoding (Eriksson, Lasic, & Topgaard, 2013; Sjölund et al., 2015). Useful helper functions for generating and manipulating gradient arrays are provided. Synthetic data from multiple gradient encoding schemes can be generated from the same simulation.

Documentation, tutorial, and contributing guidelines are provided at https://disimpy.readthedocs.io/.

Signal generation

Signal generation in Disimpy follows the framework established in (Hall & Alexander, 2009) which is briefly summarized here. Equation 1 and Equation 2 describe theory. Equation 3, Equation 4, and Equation 5 describe the numerical implementation in Disimpy.

In a dMRI experiment, the target nuclei are exposed to time-dependent magnetic field gradients which render the signal sensitive to diffusion. During the experiment, the spin of a nucleus experiences a path-dependent phase shift given by

$$\phi(t) = \gamma \int_0^t \mathbf{B}_0 + \mathbf{G}(t') \cdot \mathbf{r}(t') dt', \tag{1}$$

where γ is the gyromagnetic ratio of the nucleus, $\mathbf{B}_0$ is the static main magnetic field of the scanner, $\mathbf{G}(t)$ is the diffusion encoding gradient, and $\mathbf{r}(t)$ is the location of the nucleus. $\mathbf{G}$ and $\mathbf{B}_0$ change sign after the application of the refocusing pulse.

An imaging voxel contains an ensemble of nuclei for which the total signal is given by

$$S = S_0 \int_{-\infty}^{\infty} P(\phi) \exp{(i\phi)}\, d\phi, \tag{2}$$

where P is the spin phase distribution and S_0 is the signal without diffusion-weighting while keeping other imaging parameters unchanged.

In Disimpy, diffusion is modelled as a three-dimensional random walk over discrete time. The steps, which are randomly sampled from a uniform distribution over the surface of a sphere using the xoroshiro128+ pseudorandom number generator (Blackman & Vigna, 2018), have a fixed length

$$l = \sqrt{6 \cdot D \cdot dt}, \tag{3}$$

where D is the diffusion coefficient and dt is the duration of a time step. At every time point in the simulation, each walker accumulates phase given by

$$d\phi = \gamma \mathbf{G}(t) \cdot \mathbf{r}(t) dt. \tag{4}$$

At the end of the simulated dynamics, the normalized diffusion-weighted signal is calculated as the sum of the real parts of signals from all random walkers

$$S = \sum_{j=1}^{N} \text{Re}\left(\exp\left(i\phi_j\right)\right), \tag{5}$$

where N is the number of random walkers.

The initial positions of the random walkers are drawn from a uniform distribution across the diffusion environment. When a random walker collides with a restricting barrier, it is elastically reflected off the collision point in such a way that the random walker's total path length during dt is equal to l.

Figures

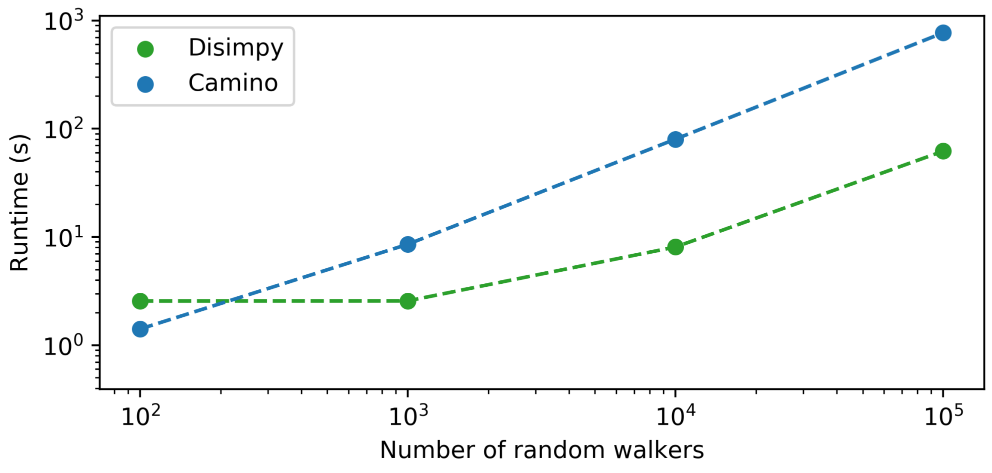

Figure 1: Performance comparison between Disimpy and Camino, a popular dMRI simulator that runs single-threaded on the CPU. The comparison was performed on a desktop computer with an Intel Xeon E5-1620 v3 3.50 GHz x 8 CPU and an Nvidia Quadro K620 GPU. The simulations were performed using a mesh consisting of 10^4 triangles, shown in Figure 2.

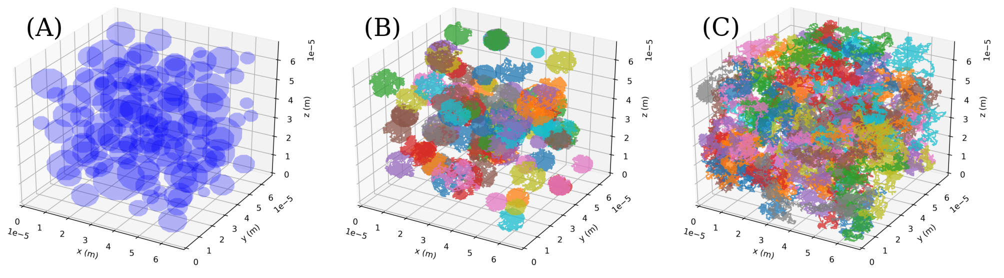

Figure 2: Example of diffusion in an environment defined by a triangular mesh. (A) Example mesh of 10^4 triangles defining the synthetic voxel consisting of 100 spheres with gamma distributed radii. Mesh kindly provided by Gyori (Gyori, Hall, Clark, Alexander, & Kaden, 2020). (B) Example trajectories of 100 random walkers whose initial positions were randomly positioned inside the spheres. Some spheres contain more than one walker. (C) Example trajectories of 100 random walkers outside the spheres.

Acknowledgements

This work was funded by the National Institute for Health Research Great Ormond Street Hospital Biomedical Research Centre (NIHR GOSH BRC).

References

Assaf, Y., Johansen-Berg, H., & Thiebaut de Schotten, M. (2019). The role of diffusion mri in neuroscience. *NMR in Biomedicine*, *32*(4), e3762. doi:10.1101/140459

Behrens, T. E., & Johansen-Berg, H. (2009). *Diffusion mri: From quantitative measurement to in-vivo neuroanatomy*. Academic Press.

Blackman, D., & Vigna, S. (2018). Scrambled linear pseudorandom number generators. *arXiv preprint arXiv:1805.01407*.

Callaghan, R., Alexander, D. C., Palombo, M., & Zhang, H. (2020). ConFiG: Contextual fibre growth to generate realistic axonal packing for diffusion mri simulation. *Neuroimage*, *220*, 117107. doi:10.1016/j.neuroimage.2020.117107

Eriksson, S., Lasic, S., & Topgaard, D. (2013). Isotropic diffusion weighting in pgse nmr by magic-angle spinning of the q-vector. *Journal of Magnetic Resonance*, *226*, 13–18. doi:10.1016/j.jmr.2012.10.015

Gyori, N., Hall, M. G., Clark, C. A., Alexander, D. C., & Kaden, E. (2020). Discrepancy between in-vivo measurements and monte-carlo simulations with spherical structures in b-tensor encoding. In *International society for magnetic resonance in medicine and society for mr radiographers & technologists virtual conference & exhibition*.

Hall, M. G., & Alexander, D. C. (2009). Convergence and parameter choice for monte-carlo simulations of diffusion mri. *IEEE transactions on medical imaging*, *28*(9), 1354–1364. doi:10.1109/tmi.2009.2015756

Hall, M. G., Nedjati-Gilani, G., & Alexander, D. C. (2017). Realistic voxel sizes and reduced signal variation in monte-carlo simulation for diffusion mr data synthesis. *arXiv preprint arXiv:1701.03634*.

Ianuș, A., Alexander, D. C., & Drobnjak, I. (2016). Microstructure imaging sequence simulation toolbox. In *International workshop on simulation and synthesis in medical imaging* (pp. 34–44). Springer. doi:10.1007/978-3-319-46630-9_4

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A llvm-based python jit compiler. In *Proceedings of the second workshop on the llvm compiler infrastructure in hpc* (pp. 1–6). doi:10.1145/2833157.2833162

Li, J.-R., Nguyen, V.-D., Tran, T. N., Valdman, J., Trang, C.-B., Van Nguyen, K., Vu, D. T. S., et al. (2019). SpinDoctor: A matlab toolbox for diffusion mri simulation. *NeuroImage*, *202*, 116120. doi:10.1016/j.neuroimage.2019.116120

Nguyen, K.-V., Hernández-Garzón, E., & Valette, J. (2018). Efficient gpu-based monte-carlo simulation of diffusion in real astrocytes reconstructed from confocal microscopy. *Journal of Magnetic Resonance*, *296*, 188–199. doi:10.1016/j.jmr.2018.09.013

Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with cuda. *Queue*, *6*(2), 40–53. doi:10.1109/hotchips.2008.7476516

Novikov, D. S., Fieremans, E., Jespersen, S. N., & Kiselev, V. G. (2019). Quantifying brain microstructure with diffusion mri: Theory and parameter estimation. *NMR in Biomedicine*, *32*(4), e3998. doi:10.1002/nbm.3998

Panagiotaki, E., Hall, M. G., Zhang, H., Siow, B., Lythgoe, M. F., & Alexander, D. C. (2010). High-fidelity meshes from tissue samples for diffusion mri simulations. In *International conference on medical image computing and computer-assisted intervention* (pp. 404–411). Springer. doi:10.1007/978-3-642-15745-5_50

Rafael-Patino, J., Romascano, D., Ramirez-Manzanares, A., Canales-Rodríguez, E. J., Girard, G., & Thiran, J.-P. (2020). Robust monte-carlo simulations in diffusion-mri: Effect of the substrate complexity and parameter choice on the reproducibility of results. *Frontiers in Neuroinformatics*, *14*, 8. doi:10.3389/fninf.2020.00008

Sjölund, J., Szczepankiewicz, F., Nilsson, M., Topgaard, D., Westin, C.-F., & Knutsson, H. (2015). Constrained optimization of gradient waveforms for generalized diffusion encoding. *Journal of magnetic resonance*, *261*, 157–168. doi:10.1016/j.jmr.2015.10.012

Stejskal, E. O., & Tanner, J. E. (1965). Spin diffusion measurements: Spin echoes in the presence of a time-dependent field gradient. *The journal of chemical physics*, *42*(1), 288–292. doi:10.1063/1.1695690

Tournier, J.-D., Calamante, F., & Connelly, A. (2007). Robust determination of the fibre orientation distribution in diffusion mri: Non-negativity constrained super-resolved spherical deconvolution. *Neuroimage*, *35*(4), 1459–1472. doi:10.1016/j.neuroimage.2007.02.016

Van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in science & engineering*, *13*(2), 22–30. doi:10.1109/mcse.2011.37

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature methods*, *17*(3), 261–272. doi:10.1038/s41592-019-0686-2