Spin-Tracking Simulations on Graphical Processing Units

M. Kline[1], D. Mathews[2], and L. J. Broussard[3]

[1]Transylvania University, Lexington, KY 40508, USA
[2]University of Kentucky, Lexington, KY 40506, USA
[3]Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

August 5, 2020

Abstract

The neutron's electric dipole moment (nEDM) is measured in the Spallation Neutron Source (SNS) nEDM experiment by detecting the spin-dependent capture events of polarized ^{3}He and ultracold neutrons in a measurement cell with parallel magnetic and electric fields. Simulations tracking particles' spin have been performed to better understand the systematic effects present in the experiment. However, spin-tracking on CPUs can be slow and computationally expensive. An additional constraint is maintaining high accuracy for long durations with accumulating rounding errors. GPU parallelization can be used to track many particles simultaneously and improve solver efficiency. We will present an overview of the approach and its advantages and limitations, as well as preliminary results of systematic studies from our simulation to track the spin of particles in the measurement cell.

1 INTRODUCTION

A permanent neutron electric dipole moment (nEDM) would indicate a direct violation of time reversal (T) symmetry, and by the CPT theorem, charge conjugation combined with parity (CP) symmetry. More CP violation is needed to fully understand the baryon asymmetry in the universe, so the presence of a permanent nEDM would help give insight into the origin of matter.[1] The Spallation Neutron Source (SNS) nEDM experiment[2,3] looks to measure, or place a limit on, the nEDM to a planned sensitivity of $d_n \approx 3 \times 10^{-28}$ e $\cdot$ cm, whereas the current limit is set at $|d_n| < 1.8 \times 10^{-26}$ e $\cdot$ cm.[4]

The SNS nEDM experiment consists of two measurement cells containing polarized ^{3}He and ultracold neutrons (UCNs). Both cells have a uniform magnetic field, $\vec{B}$, where one cell has an electric field, $\vec{E}$, parallel to $\vec{B}$ and the other cell anti-parallel. Cold neutrons from the SNS are downscattered into UCNs by superfluid helium in the cells. The spins are aligned with $\vec{B}$ and then an RF pulse is applied to induce a $\pi/2$ spin rotation so the spins are perpendicular to $\vec{B}$. The spins are then allowed to precess with Larmor frequency ω:

$$\omega = -\frac{2}{\hbar}(\mu_n|\vec{B}| \pm d_n|\vec{E}|), \tag{1}$$

where the positive sign represents parallel $\vec{E}$ and $\vec{B}$ fields and the negative sign anti-parallel, $\hbar$ is Planck's constant, μ_n is the magnetic dipole moment, and d_n is the electric dipole moment. For the $|\vec{B}| = 30$mG field in the experiment, the Larmor frequency is 549.7rad/s for neutrons and 610.2rad/s for ^{3}He. While the UCNs and ^{3}He precess in the measurement cell, they can undergo the spin-dependent capture reaction when the spins are anti-parallel:

$$n + {}^3\text{He} \rightarrow p + {}^3\text{H} + 764 \text{ keV}. \tag{2}$$

Thus, using SQUID sensors to measure the ^{3}He precession frequency, and measuring the scintillation light from products of the capture reactions traveling through the superfluid helium, the precession frequency of the UCNs can be determined. The difference between the UCN precession frequencies for parallel $\omega_{\uparrow\uparrow}$ and anti-parallel $\omega_{\downarrow\uparrow}$ electric and magnetic fields in the two measurement cells is used to determine the nEDM, d_n:[2]

$$d_n = \frac{\hbar(\omega_{\downarrow\uparrow} - \omega_{\uparrow\uparrow})}{4|\vec{E}|}. \tag{3}$$

Spin-tracking simulations of particles in the measurement cell are needed to better understand systematic effects such as the Bloch-Siegert induced false EDM, which is caused by the interaction of the motion-induced $\vec{E} \times \vec{v}/c^2$ field with magnetic field gradients, where $\vec{v}$ is the velocity.[3] This simulation used the Julia language and CUDA to run on Graphical Processing Units (GPUs). A simulation has previously been developed using CPU.[5,6] Since these simulations are computationally expensive, we used GPU parallelization to simulate more than 10^4 particles simultaneously per GPU.

2 SIMULATION

2.1 Spin Precession

To track the neutrons' spins, the 5th order Runge-Kutta method was used to numerically integrate the Bloch equation

$$\dot{\sigma} = \gamma\sigma \times \vec{B} - \frac{2d_n}{\hbar}\sigma \times \vec{E}, \tag{4}$$

where σ is the normalized spin vector and γ is the gyromagnetic ratio.

An initial attempt used symplectic integration to track the spin of the neutrons. Symplectic integrators are more stable over long durations compared to a non-symplectic method.[7] Symplectic integration is a numerical solver for separable Hamiltonian systems of the form

$$\mathcal{H} = T(\vec{p}) + V(\vec{q}). \tag{5}$$

The differential equations $\frac{dx}{dt} = f(t, v)$ and $\frac{dv}{dt} = g(t, x)$ satisfy this Hamiltonian. When $\frac{\partial \vec{B}}{\partial t} = 0$, the Bloch equation (Eq. 4) can be written in the form:[5]

$$\frac{d\sigma}{dt}(t, \dot{\sigma}) = \dot{\sigma} \quad \frac{d\dot{\sigma}}{dt}(t, \sigma) = \gamma^2(\sigma \times \vec{B}) \times \vec{B}. \tag{6}$$

Then, since we don't have a Hamiltonian of the required form, from these equations we attempted to use the transformation $x = \sigma$, $v = \dot{\sigma}$ to get the differential equations in a form where symplectic integration can be used.

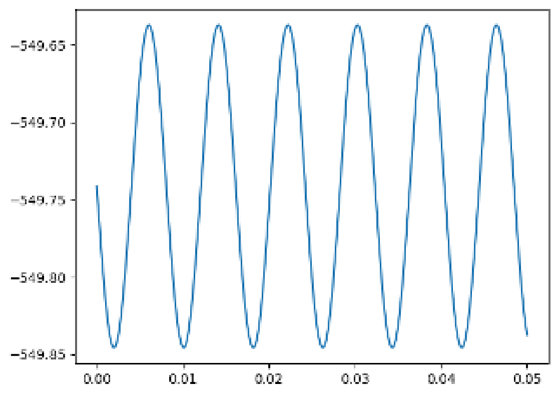

Figure 1: Neutron Larmor frequency vs propagation time in seconds for symplectic integration

The mean precession frequency using this method was very accurate to about 10^{-7}rad/s, but the precession frequency oscillated dramatically about the ideal precession frequency by about ± 0.1rad/s, as shown in Figure 1. Thus, we decided to use the 5^{th} order Runge-Kutta method to avoid the large oscillation.

Symplectic integration could still be a useful tool for tracking the position of the particles. Given a Hamiltonian $\mathcal{H} = \frac{p^2}{2m} + mgz$, the position of a particle can be tracked under the effects of gravity. However, reflections are treated as an abrupt change and don't conserve the Hamiltonian, which leads to errors with symplectic integration.

2.2 Handling wall interactions

The walls of the measurement cell have a sufficiently high Fermi potential V_F to ensure the UCNs reflect off of the walls, trapping them in the cell.[8] The neutrons do, however, still have a chance to capture on the walls. Currently, wall losses are modeled with a constant probability at each reflection, however this could be updated to include the energy-dependent probability. The probability is tuned to match an expected wall loss lifetime of 2000s, and will eventually be experimentally determined for each measurement cell.

If the neutron doesn't capture on the wall, it can undergo either a specular or diffuse reflection. To model specular reflections, periodic boundary conditions were added. Not only does this make specular reflections easier to handle, it also preserves accuracy, whereas handling specular reflections by changing the velocity vector would require a variable step size to maintain accuracy, which can add significant computation time. One problem with doing this, however, is that the top and bottom walls do not behave properly, as gravity breaks the symmetry. Instead of reflecting back and begin decelerating after reflecting off the bottom wall, the neutron will keep accelerating in the same direction. To solve for this, the cell was duplicated into two, where the top cell has normal gravity and the bottom cell has antigravity. This presents another problem, however: When crossing one of these boundaries, the neutron will be accelerating in the wrong direction for a part of the step, meaning that the resultant speed will be wrong.

To test this, the model was compared to an analytic solution for the particle oscillating between the two regions. This small increase in speed at each reflection caused an exponential accumulation

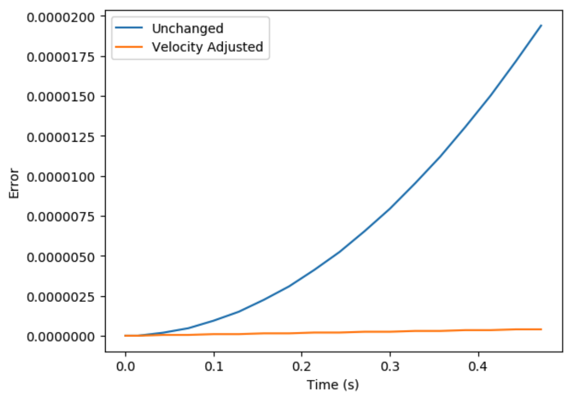

Figure 2: Accumulated error vs propagation time of the particle. The blue line is incorrectly handling vertical wall reflections, the orange corrects for the speed after a vertical reflection.

of error, as shown in Figure 2. This wrong speed can be corrected by rescaling the velocity vector to the same speed as the previous step. This reduces the exponential accumulation of error to linear. The small error in position isn't important, as the smoothness of the measurement cell wall surfaces aren't precise to that level. What is important is that the error in the distribution of UCN velocities remains small. While not correcting the velocities will lead to them to continually increase, rescaling them keeps this error small.

The probability of a particle undergoing a diffuse reflection depends on the material properties of the wall, as well as the incident angle of the neutron. The probability of a diffuse reflection is proportional to $\cos\theta_i$, where θ_i is the incident angle between the neutron's velocity and the surface's normal vector. The probability is left as a simulation variable, with a typical value being around 20% for $\theta_i = 0$. If a diffuse reflection does take place, the outgoing velocity must preserve, on average, the density of the phase space of the particles. Thus, the outgoing direction can't be chosen from a simple uniform half-spherical distribution, but must be weighted so that for a uniformly distributed random number $y \in [0,1]$, the outgoing angle $\theta = \cos^{-1}\left[(1-y)^{1/3}\right]$. The azimuthal angle, ϕ, can be picked from a uniform distribution in 2π.[5]

2.3 Decay and capture events

After each time step in the simulation, the neutron has a probability of either undergoing β decay or reacting with the ^{3}He present in the cell. Based upon the mean lifetime[9] of $\tau_\beta = 881.5$s and time step h, the probability of a neutron decaying in any given step is $\Gamma_\beta h$ where $\Gamma_\beta = 1/\tau_\beta$.[10]

Based on a n-^{3}He absorption lifetime of $\tau_3 = 500$s, the capture rate is $\Gamma_3 = 1/\tau_3 = 2 \times 10^{-3}$Hz. The capture cross-section is strongly spin-dependent, where the probability is maximal when the spins are anti-aligned and approximately zero when aligned, giving an effective capture rate $\Gamma_3'(\theta) = \Gamma_3(1-\cos\theta)$.[5] Thus, the probability of a neutron capture in any given step is given as $h\Gamma_3(1-\cos\theta)$. ^{3}He atoms are not tracked in the simulation, their spins are solved analytically. At this time, the simulation does not include any ambient background events, which would add an additional uncertainty when trying to determine the precession frequency.

2.4 Simulated signal

From this simulation, we can generate a plot of the simulated signal vs time, shown in Figures 3-4. The large-scale plot shows the decay of $2^{21} (\approx 2 \times 10^6)$ neutrons throughout the simulation. Zooming in shows that the capture events oscillate. The frequency of the capture events can be determined by fitting the data using the maximum likelihood method. The uncertainty in the fit parameters can be determined using a χ^2 goodness of fit test. A more detailed explanation of this process is described in Appendix A.

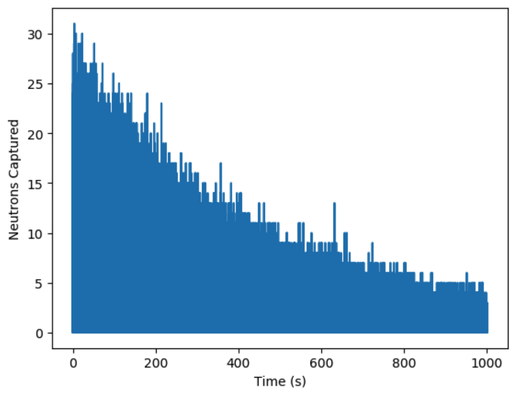

Figure 3: Simulated signal vs time

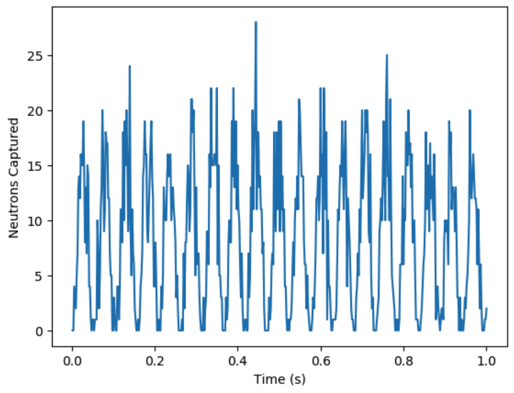

Figure 4: Zoomed in simulated signal vs time

Given the optimal ω from the maximum likelihood fit, the nEDM can be calculated by the equation

$$d_n = \frac{\hbar}{2|\vec{E}|}[\omega - (\gamma_3 - \gamma_n)\frac{\omega_3}{\gamma_3}]. \tag{7}$$

The uncertainty in d_n is taken directly from the uncertainty in ω, which is calculated using Eq. 16. From this, given an input $d_n = 3.3 \times 10^{-27} \mathrm{e} \cdot \mathrm{cm}$, the fit calculated $d_n = (7.2 \times 10^{-23} \pm 6.5 \times 10^{-22})$ e $\cdot$ cm. While the error here is very large, it falls within 1σ, and additional runs will reduce this uncertainty.

3 GPU OPTIMIZATION

Spin-tracking simulations are very computationally expensive. Simulating many particles opens up the possibility of parallelization, so that all particles can be computed at the same time. Thus, running simulations on GPUs can be an efficient way to run these simulations. We used the CUDA.jl package[11] to run on Nvidia GPUs. The current graphics card being used is the GeForce GTX 1080 Ti.

Our initial approach to optimize the code for GPU was to convert each step into a matrix form. Matrix multiplication is very fast on GPUs since it can easily be computed in parallel. To accomplish this, all differential equations - equations of motion and the Bloch equation - had to be converted into matrix form. Then, the Runge-Kutta method had to be converted into a matrix equation, as well. While this approach did turn out to be efficient, it lacked the flexibility needed

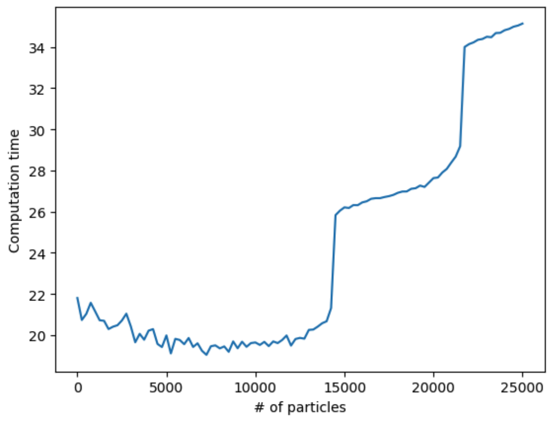

Figure 5: Computation time vs number of particles simulated. The jumps indicate the calling of another set of streaming multiprocessors.

for this simulation, such as being able to include a time-varying magnetic field. See Appendix B for a derivation of this approach and a more detailed explanation of its benefits and limitations.

A more flexible approach was to write GPU kernels that would handle the different physics. The first kernel handles the movement of the neutron, including moving the neutron to the next step, implementing specular and diffuse reflections, as well as wall losses. Then, another kernel implements the Runge-Kutta algorithm to solve the Bloch equation and calculate the spin at the next step. Another kernel handles n-^{3}He capture events and β decay, and a final kernel stores the event type and the state of the neutrons that either got absorbed in the wall, experienced a capture reaction, or underwent β decay. Those neutrons then get reinitialized at t=0 to not waste computation time on neutrons that have already experienced an event. This means that not all particles are necessarily in the same time step. These kernels are called sequentially in a loop for the desired number of iterations.

This method still is able to utilize GPU parallelization while also allowing additional features that don't work in the matrix approach. This method is also very fast, and the computation time doesn't start to increase until there are about 15,000 particles being simulated. At that point, the task requires another set of streaming multiprocessors (SMs), causing a jump in computation time. Then, the computation time will remain approximately constant until another set of SMs are required. This behavior is shown in Figure 5.

4 CONCLUSIONS

We successfully built a framework based on a simplified model to track UCNs in the measurement cell that can produce simulated signals and run efficiently in parallel on GPUs. Future work to expand on this work will include adding physics to better understand systematic effects leading to a false EDM measurement, like the Bloch-Siegert induced false EDM. For the nEDM experiment to meet its sensitivity goal of $3 \times 10^{-28}\mathrm{e} \cdot \mathrm{cm}$, more computational power is needed to simulate the 10^{11} events required. Thus, this type of GPU-based or a hybrid CPU/GPU-based software would be a good application for the Summit supercomputer.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internship program, and by the U.S. Department of Energy, Office of Nuclear Physics under contract number DE-AC05-00OR2272.
We are grateful to C. Swank for helpful discussions and for providing CPU-based spin-tracking codes.

A SIGNAL PROCESSING

A.1 Maximum Likelihood Fit

The events are sorted into 7ms bins and the number of events per bin is plotted vs time. The rate of events, $\Gamma(t)$, is described by the equation

$$\Gamma(t) = N(t) \times [\Gamma_\beta + \Gamma_3(1 - cos[\omega t + \phi_0])] + \Gamma_B(t), \tag{8}$$

where the rate for a single neutron is multiplied by $N(t)$, the number of remaining neutrons in the measurement cell at time t, described by the equation

$$N(t) = N_0\, e^{-t/\tau_{\text{eff}}}. \tag{9}$$

This equation describes the decay of the number of neutrons in the measurement cell seen in Fig. 3. The effective decay parameter τ_{eff}, to first order, is related to the signal rates by

$$\frac{1}{\tau_{\text{eff}}} \simeq \Gamma_\beta + \Gamma_3 + \frac{1}{\tau_{\text{loss}}}. \tag{10}$$

A background rate Γ_B is included in the signal rate but is set to zero in the simulation, as it has not been added yet. There is also a polarization term for both the neutrons and ^{3}He that has been excluded as it hasn't been added to the simulation yet.

To fit the simulated signal, we need an expected number of events in each bin. From the rate given in Eq. 8, the expected number of events in a given interval Δt is

$$\lambda = \int_{\Delta t} \Gamma(t)\, dt. \tag{11}$$

Then for each bin, given parameter vector $\boldsymbol{\theta}$, you can calculate the number of expected events.

The maximum likelihood method was used to optimize the fit parameters $\boldsymbol{\theta}$ for the generating rate $\Gamma(t; \boldsymbol{\theta})$. The logarithmic likelihood is given as

$$\log \mathcal{L} = \sum_{i=1}^{N_{\text{bins}}} [k_i \log(\lambda_i) - \lambda_i - \log(k_i!)], \tag{12}$$

where k_i is the observed number of events in bin i and λ_i is the expected number of events. Using the Nelder-Mead optimization algorithm, this equation was maximized to find $\hat{\boldsymbol{\theta}}$, the parameters that maximize the likelihood. The obtained fit is plotted in Figure 6 with the data and the expected number of events from the input parameters.

A.2 Chi-Squared Parameter Scan

To determine the statistical significance of the fit, a χ^2 goodness of fit test was performed. The goodness of fit can be constructed by

$$\chi^2 \approx -2 \log \Lambda, \tag{13}$$

where $\Lambda = \mathcal{L}/\hat{\mathcal{L}}$ is the maximum likelihood ratio, defined to be

$$\log \Lambda = \sum_{i=1}^{N_{\text{bins}}} \left[k_i \log\left(\frac{\lambda_i}{k_i}\right) - (\lambda_i - k_i)\right]. \tag{14}$$

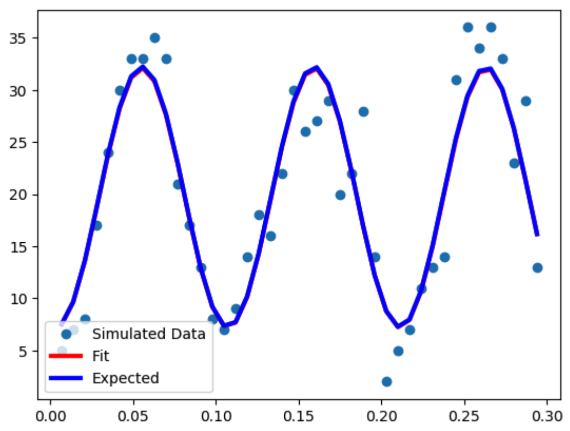

Figure 6: Maximum likelihood fit of simulated signal.

The estimated correlation matrix Σ comes from the inverse Hessian matrix of Eq. 13 with optimal fit parameters $\hat{\boldsymbol{\theta}}$. The uncertainty, then, in each parameter comes from the correlation matrix by

$$\sigma_i^2 = e_i^{\mathrm{T}} \Sigma e_i, \tag{15}$$

where e_i is the unit vector in the direction of interest. So, the error in the parameters can be calculated by

$$\sigma^2 = \mathrm{diag}^{-1}(\mathbf{H}^{-1}). \tag{16}$$

This was used to determine the uncertainty in the parameters.

Additionally, varying each parameter by the uncertainty given by the previous method, a plot was generated for the χ^2 goodness of fit, shown in Figure 7. The minimum χ^2 value in this plot is 1.07 and the maximum is 2.24, with the minimum in the center corresponding to $\hat{\boldsymbol{\theta}}$. Since the plot is an oval that doesn't appear to be rotated, the two parameters are uncorrelated.

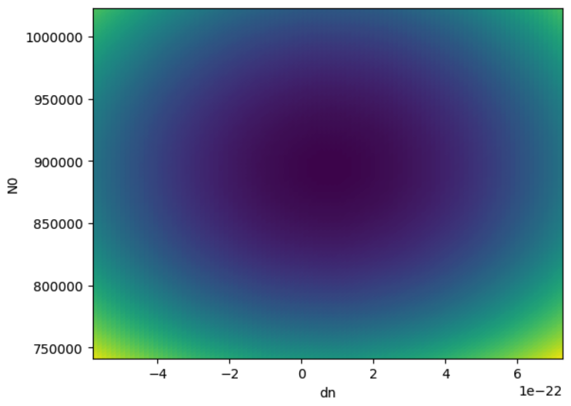

Figure 7: χ^2 parameter scan. N_0 and d_n are shown to be uncorrelated.

B RUNGE-KUTTA METHOD IN MATRIX FORM

B.1 Derivation

We want each Runge-Kutta step to be a matrix equation of the form

$$y_{n+1} = \mathrm{A}_{\mathrm{RK}}\ y_n + \mathrm{b}_{\mathrm{RK}} \tag{17}$$

so that the next state is a linear combination of the current state. This is only possible if dy/dt is also a matrix equation:

$$\frac{dy}{dt} = \mathrm{A}y + \mathrm{b}. \tag{18}$$

For the Bloch equation (Eq. 4), this differential equation becomes

$$\mathrm{A} = \begin{bmatrix} 0 & \gamma B_z - \frac{2d_n}{\hbar}E_z & -\gamma B_y + \frac{2d_n}{\hbar}E_y \\ -\gamma B_z + \frac{2d_n}{\hbar}E_z & 0 & \gamma B_x - \frac{2d_n}{\hbar}E_x \\ \gamma B_y - \frac{2d_n}{\hbar}E_y & -\gamma B_x + \frac{2d_n}{\hbar}E_x & 0 \end{bmatrix} \quad \mathrm{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Then, to find A_{RK} and b_{RK}, we must go through the 4th order Runge-Kutta method:[12,1]

$$\begin{aligned} k_1 &= \frac{dy}{dt}(y_n) \\ &= \mathrm{A}y_n + \mathrm{b} \end{aligned}$$

$$\begin{aligned} k_2 &= \frac{dy}{dt}(y_n + h\frac{k_1}{2}) \\ &= \mathrm{A}[y_n + \frac{h}{2}(\mathrm{A}y_n + \mathrm{b})] + \mathrm{b} \\ &= (\mathrm{A} + \frac{h}{2}\mathrm{A}^2)y_n + (\mathrm{b} + \frac{h}{2}\mathrm{A}\mathrm{b}) \end{aligned}$$

Continuing to follow this method, we get

$$\begin{aligned} k_3 &= \frac{dy}{dt}(y_n + h\frac{k_2}{2}) \\ &= (\mathrm{A} + \frac{h}{2}\mathrm{A}^2 + \frac{h^2}{4}\mathrm{A}^3)y_n + (\mathrm{b} + \frac{h}{2}\mathrm{A}\mathrm{b} + \frac{h^2}{4}\mathrm{A}^2\mathrm{b}) \end{aligned}$$

$$\begin{aligned} k_4 &= \frac{dy}{dt}(y_n + hk_3) \\ &= (\mathrm{A} + h\mathrm{A}^2 + \frac{h^2}{2}\mathrm{A}^3 + \frac{h^3}{4}\mathrm{A}^4)y_n + (\mathrm{b} + h\mathrm{A}\mathrm{b} + \frac{h^2}{2}\mathrm{A}^2\mathrm{b} + \frac{h^3}{4}\mathrm{A}^3\mathrm{b}) \end{aligned}$$

Then, the next step can be calculated by

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \\ &= (h\mathrm{A} + \frac{h^2}{2}\mathrm{A}^2 + \frac{h^3}{6}\mathrm{A}^3 + \frac{h^4}{24}\mathrm{A}^4)y_n + (h\mathrm{b} + \frac{h^2}{2}\mathrm{A}\mathrm{b} + \frac{h^3}{6}\mathrm{A}^2\mathrm{b} + \frac{h^4}{24}\mathrm{A}^3\mathrm{b}) \end{aligned}$$

[1]While I derive the 4th order Runge-Kutta method for simplicity, higher order RK methods follow a similar derivation.

So,

$$\mathrm{A}_{\mathrm{RK}} = h\mathrm{A} + \frac{h^2}{2}\mathrm{A}^2 + \frac{h^3}{6}\mathrm{A}^3 + \frac{h^4}{24}\mathrm{A}^4 \tag{19}$$

$$\mathrm{b}_{\mathrm{RK}} = h\mathrm{b} + \frac{h^2}{2}\mathrm{Ab} + \frac{h^3}{6}\mathrm{A}^2\mathrm{b} + \frac{h^4}{24}\mathrm{A}^3\mathrm{b} \tag{20}$$

B.2 Benefits and Limitations

Since GPUs are very efficient at computing matrix multiplication, this method is incredibly fast. Additionally, if you don't need to add any additional physics (reflections, capture events, etc.), then for n iterations, you can very quickly solve for every iteration by doing $y_n = \mathrm{A}^n y_0 + (\mathbf{1} + \mathrm{A} + \mathrm{A}^2 + ... + \mathrm{A}^{n-1})\mathrm{b}$. This makes it a great candidate for some applications, but as you need to add more simulation features, this method becomes slower.

To add more physics, you can just run those operations in between Runge-Kutta steps. However, there are more significant limitations. This method assumes that: (1) the step size is constant and (2) the differential equation is time independent. If either of these isn't satisfied, both A_{RK} and b_{RK} will change, meaning the matrix will have to be recalculated each step, increasing computation time significantly.

Because of these limitations, we decided to not use this method and to instead put the Runge-Kutta step into a GPU kernel. This allows for a time-varying magnetic field. It also removes the requirement to calculate two matrices for the top cell, with normal gravity, and the bottom cell, with anti-gravity, and sorting the matrices to determine which particles correspond to which matrix.

References

[1]T. E. Chupp, P. Fierlinger, M. J. Ramsey-Musolf, and J. T. Singh, "Electric dipole moments of atoms, molecules, nuclei, and particles", Reviews of Modern Physics **91**, 015001 (2019).

[2]M. W. Ahmed et al., "A new cryogenic apparatus to search for the neutron electric dipole moment", Journal of Instrumentation **14**, P11017 (2019).

[3]K. K. H. Leung et al., "The neutron electric dipole moment experiment at the spallation neutron source", EPJ Web of Conferences **219**, 02005 (2019).

[4]C. Abel et al., "Measurement of the permanent electric dipole moment of the neutron", Physical Review Letters **124**, 081803 (2020).

[5]R. Schmid, "New search for the neutron electric dipole moment using ultracold neutrons at the spallation neutron source", PhD thesis (California Institute of Technology, 2014).

[6]C. Swank, `https://github.com/cmswank/spin-sim-schmid`.

[7]G. Zsigmond, *Neutronic simulations overview*, (July 2018) `http://neutron.physics.ncsu.edu/NeutronSummerSchool/talks/Zsigmond_neutronSummerSchool2018.pdf`.

[8]R. Golub, D. Richardson, and S. K. Lamoreaux, *Ultra-cold neutrons* (1991).

[9]M. Tanabashi, K. Hagiwara, K. Hikasa, K. Nakamura, Y. Sumino, F. Takahashi, J. Tanaka, K. Agashe, G. Aielli, C. Amsler, et al., "Review of particle physics", Physical Review D **98**, 030001 (2018).

[10]*Nuclear decay*, `http://electron6.phys.utk.edu/phys250/modules/module%205/nuclear_decay.htm`.

[11]`https://github.com/JuliaGPU/CUDA.jl`.

[12]W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in c*, 2nd ed. (1992), pp. 710–722.