PXT999 Research Project

Dissertation title: **Evaluating Monte Carlo Methods for Simulating Diffusion MRI**
Author: **Kavian Shirkoohi**
Supervisor: **Dr Leandro Beltrachini**

Date of submission: **13/5/2022**

1. Introduction

Magnetic resonance imaging (MRI) is a non-invasive neuroimaging technique for generating images of organic processes. By studying the magnetic properties, leaving the chemical properties unchanged, it is far safer than previous methods such as CT and PET scans [1]. Diffusion MRI (dMRI) is a variation of this technique, which uses nuclear magnetism to obtain images from the motion of water particles in the subject. This project investigates the Monte Carlo (MC) simulation of water diffusion to generate synthetic signal data, modelling each molecule's Brownian motion as random walk (RW). The resultant de-phasing of the applied field gradient yields the signal attenuation for the region of interest (ROI) which we can use to compute the apparent diffusion coefficient (ADC) which summarises the diffusive activity. Furthermore, the diffusion tensor (DT) can also be calculated, which gives a more in-depth view of axonal diffusion. The reverse process can be used to infer the microstructure of the ROI when unknown, which provides an opportunity for imaging beyond conventional apparatus resolution [2]. Standard MRI is sensitive to proton densities of the subject studied. T_1 and T_2 weighted images map the static structure of the tissue from fat and water. The signal will be an induced voltage from changing magnetism of the atomic configuration within the ROI. The magnitude is very small, so a contrasting agent such as Gadolinium (Gd) may be administered intravenously as a marker dye. As a result, the anatomy can be imaged in-vivo in ways which were previously inaccessible[1] [3]. Diffusion-weighting (DW) is another form of imaging, sensitive to water molecule diffusion in tissue. A distinct advantage of dMRI is that no contrasting agent is used. Since the human brain is composed of approximately 73% water [4], the use of contrast agents has little effect on the SNR and CNR ratios of DW images [5]. While standard MRI maps tissue structure, dMRI can be used to generate a DT representing the activity within it [6]. DT imaging (DTI) is useful in functional MRI (fMRI) [7] where brain activity is monitored in preference to hodology [8]. It can be used in fibre tractography (FT) to create a 3D model of neural topology [9].

The human brain is roughly 40% grey matter and 60% white matter in a healthy subject [10]. The central nervous system (CNS) contains a number of neuronal cell types [11] but all follow the same blueprint: a cell body (neurone) and a nerve fibre (axon). Bundles of axons are referred to as nerve tracts [12]. Grey matter is formed of the neurones responsible for directing and processing signals [13], whilst white matter consists of the tracts through which the signals travel [14]. Axons are insulated by a myelin sheath, unlike neurones. This myelination protects the tracts from damage and preserves the integrity of the signal [15]. The majority of signalling is carried out through a series of electrical pulses travelling through the network. Water is the primary conductor of this electrochemical activity and is essential for understanding brain function [16]. Water molecules travel between cells via aquaporins, which regulate the flow [17]. Along the direction of the tracts, there is isotropic diffusion. However, diffusion is anisotropic throughout the cross section of the fibres [18]. It is the anisotropic diffusion that we wish to study, since it can infer the microstructure of the tissue [19]. It is no surprise that white matter is the tissue of interest in dMRI. Previously overlooked, studies of white matter have led to early detections in neurological disorders where axon damage has occurred. An example is the ischaemic stroke, where electrochemical activity is significantly reduced or halted [20]. It is now understood that this is caused by inflammatory lesions causing swelling throughout axons [21]. Changes in myelination or even demyelination can result in severe signal loss [22].

Studying white matter presents many opportunities for research, from determine the age and general health of a patient [23] to the correlation of cognitive practise with myelination of relevant areas in the brain [24]. Despite state-of-the-art developments in imaging technology, there are limitations. High resolutions are important for an accurate microstructure mapping, with axon diameters in the range of μm. It is impossible to trace a single water molecule's motion, and microscopic observations are limited to the apparatus precision and practicality. The stochastic nature of Brownian motion along with the requirement of simulating many particles over a given time period makes it an ideal candidate for Monte Carlo modelling, particularly with recent advancements in parallel computing [25].

[1] Prior to advancements in radiology during the 20th century, postmortem investigations were the only way to understand organic hodology in the past. Permission had to be obtained in order to experiment with dead bodies. By the time it was accessible for study, much of the tissue had decomposed.

2. Précis of literature

2.1. Nuclear magnetic resonance

Hydrogen (H) is abundant in fat and water, and its magnetism is modelled as a dipole with an associated magnetic moment. The motion of this moment is the atomic intrinsic angular momentum, known as spin. It precesses in a conical frequency, at a periodic Lamor frequency ω_0, the natural frequency of the atom:

$$\omega_0 = \gamma B_0 \tag{1}$$

B_0 is the magnitude of the applied field, and γ the gyromagnetic ratio which is typically 2.675×10^8 $s^{-1}T^{-1}$ for a proton [26]. The state of a spin can be up or down, corresponding to higher or lower energy configurations. Nuclear magnetic resonance (NMR) is the study of this phenomena, and consists of exposing organic tissues to a strong applied field $\mathbf{B_0}$ which is perturbed by a weaker pulsed radiofrequency (RF) field $\mathbf{B_1}$ [27]. The resulting signal is characteristic of proton densities in tissue, used to determine composition. NMR spectroscopy studies the frequency spectra within the signal, whilst MRI is focused on intensity[2] [28].

Over a large spin population, moments are assumed to be randomly oriented and thus cancelling out, leaving the overall net magnetisation zero. However, when a magnetic field $\mathbf{B_0}$ is applied some spins align with its plane. The ratio of aligned to misaligned spins is the spin excess. The down spins have enough energy to preserve their orientation, whilst the up spins do not and align with the poles of the field. The spins do not precess in phase by default, so are assumed incoherent: the phases also cancel out. We define the net magnetisation vector **M** to correspond to the magnetism caused by spin excess. Our aim is to measure non-zero spin, but we cannot obtain the signal from **M** alone as the effect is minuscule[3].

By applying $\mathbf{B_1}$ perpendicular to $\mathbf{B_0}$ and pulsing it in resonance with the Lamor frequency, spin excess can be isolated. When **M** is in the plane of $\mathbf{B_1}$ there is a phase coherence in the aligned spins. After $\mathbf{B_1}$ is pulsed, spins re-align with $\mathbf{B_0}$ during a period known as relaxation time. Atomic energy configurations change and excess energy is re-emitted through photons in the RF range. This results in a magnetic flux quantified by the intensity, which induces a voltage signal. The process can be repeated several times to increase contrast as long as the subject is stationary [29]. Realignment sensitivity and relaxation time is characteristic for tissue, bone and tumours.

The cylindrical MRI machine contains a flat surface where the patient is placed. Two receiver coils are positioned at both ends for voltage induction [30, 31]. MR imaging localises these signals with additional magnetic fields of varying strengths, known as gradients [32]. Each gradient has an associated coil pair and power amplifier [33]. Setting up gradients orthogonally allows specific organs to be isolated [34]. The pulsed-gradient spin-echo (PGSE) sequence summaries their sequence of application [35]. Once the signal is acquired, the readings from each gradient are combined to generate voxels through slicing [36]. These readings are decoded through successive Fourier transforms for imaging [37].

2.2. Fick's laws of diffusion

In our study, diffusion originates from Brownian motion. For a given concentration of particles, there will be a local diffusivity coefficient *D*, derived from Einstein's ideal gas models [38]. The diffusive flux vector **J** is related to the concentration gradient φ by Fick's first law:

$$\mathbf{J} = -D\nabla\varphi \tag{2}$$

The concentration change over time is given by Fick's second law, analogous to the Heat equation [39]:

$$\frac{\partial\varphi}{\partial t} = D\Delta\varphi \tag{3}$$

With (2) and (3), the mean-squared displacement (MSD) is generalised for Brownian particles. The root MSD (RMSD) describes the distance in n-dimensional space a particle has translated after time *t*:

$$\text{RMSD} = \sqrt{2nDt} \tag{4}$$

---

[2] NMR and MRI used to be synonymous, but the former term fell out of favour with the public during the Cold War.

[3] For a 1.5T applied field, the spin excess is typically 1 in 60000 atoms. This makes it impractical to measure alone.

2.3. The Stejskal-Tanner sequence

The PGSE sequence can be modified to become diffusion-sensitive. A special variation, developed by Edward Stejskal and John Tanner in the 1960s, requires a single gradient G_D to contrast diffusing spins from stationary ones [40, 41]. The dMRI sequence (**Fig 1**) is defined as follows:

1. Applying $\mathbf{B_0}$ as usual, $\mathbf{B_1}$ is pulsed 90° causing spin excitation
2. G_D is applied for a duration δ which de-phases spins
3. $\mathbf{B_1}$ is again applied at 180° for the same duration, refocussing spins
4. G_D is applied again for δ but this time is perturbed by diffusing spins
5. A final $\mathbf{B_1}$ is pulsed 90°, re-phasing spins and resetting the system

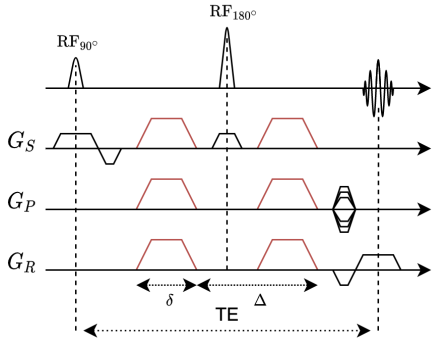


Fig 1. A schematic of the Stejskal-Tanner sequence.

The image is acquired immediately after. The key trick is that the stationary spins stay in phase with each other, so $\mathbf{B_1}$ merely inverts their orientation. The moving spins, however, lose their original positions and orientations since diffusion is memoryless [42]. If needed, another 180° application of $\mathbf{B_1}$ can be pulsed prior to acquisition to suppress eddy currents [43]. The signal attenuation quantifies dMRI and is the ratio between the diffusion-weighted and unweighted baseline signal, proportional to diffusion [44]. It can be used to calculate the apparent diffusion coefficient (ADC) for a given voxel [45].

In practise, diffusion is hindered by cell membranes, and is distinct within the intracellular space (ICS) and extracellular space (ECS). The flow between compartments is regulated by aquaporins (AQPs) [46]. AQPs exhibit a property called permeability, which can range from permitting all molecules to free pass between compartments to not allowing exchange at all [47]. The AQPs common in brain matter are AQP1 and AQP4 [48]. Selective water permeation is quantified by the coefficient κ in units μm/ms, derived from particle exchange times [49]. The attenuation caused from a diffusing particle is proportional to its de-phasing contribution [50]:

$$\phi(t) = a(t)\gamma G(t) \cdot x(t) \tag{5}$$

Where $G(t)$ is the time evolution of G_D and $x(t)$ the displacement of the spin from its start point. The function $a(t)$ is a sign shift from RF pulsing (as mentioned in step 3):

$$a(t) = \begin{cases} +1, & \text{if } t < \delta \\ -1, & \text{otherwise} \end{cases} \tag{6}$$

We can therefore express signal attenuation as a result of de-phasing from all spins, at echo time TE:

$$\frac{S}{S_0} = \left(e^{-i \int_0^{t=TE} \phi(t) dt'} \right) \tag{7}$$

We can also express this quantity in terms of the scan parameters:

$$\frac{S}{S_0} = \exp\left[-\gamma^2 G^2 \delta^2 \left(\Delta - \frac{\delta}{3} \right) D \right] \tag{8}$$

Where G is the gradient strength of G_D, δ the pulse duration and Δ the time between pulses. The ST pulses can be combined with a weaker application of PGSE gradients from standard MRI to localise this expression. We can further simplify it by introducing the b-value, a quantity characteristic of the gradient strength and timing [51]:

$$b = \gamma^2 G^2 \delta^2 (\Delta - \delta/3) \tag{9}$$

$$\frac{S}{S_0} = \exp(-b \cdot ADC) \tag{10}$$

Note D is substituted with the ADC for restricted diffusion. By imaging the same voxel with at least two different b-values, the ADC can be inferred [52]:

$$\mathrm{ADC}(x, y, z) = \ln\left[S_2(x, y, z) / S_1(x, y, z) \right] / \left(b_1 - b_2 \right) \tag{11}$$

The ADC summarises voxel diffusion, but a diffusion tensor (DT) can also be constructed from more readings [53]. The DT has the advantage of preserving directional information of the diffusion, and is well-suited for white matter tractography [54].

2.4. Computational modelling

2.4.1. Limitations of analytical methods

There are numerous means to simulate signal attenuation in the dMRI experiment [55]. The time evolution of diffusion-induced magnetism *m* is represented through the Bloch-Torrey equation, defining behaviour in curvilinear coordinates [56, 57] using quantities previously mentioned:

$$\frac{\partial m}{\partial t} = -i\omega(\boldsymbol{r}, t)m + D\nabla^2 m \tag{12}$$

The PDE in (12) can be discretised and solved iteratively using a finite-element [58] or finite-difference method [59]. The solutions are used to determine the b-values and ultimately signal attenuation [60, 61]. The most well-known application is the extensive work of Denis Grebenkov. In his 2010 paper, the restricted diffusion is solved for a two-compartment disk chamber [62] in **Fig 2**. This approach involves the use of matrices and Laplacian operators with a finite-element solver [63, 64]. The model is commonly used in MC simulations as a ground-truth framework for validation. Other approaches include formulating the Bloch-Torrey equation as an ODE, and the Kärger models [65, 66]. A recent improvement in analytical simulations includes a portable framework streamlined for finite-element discretisation which aims better performance than most MC simulations [67]. Despite the efficiency of numerical methods for dMRI, there is a crucial usage case they fall short on: simulating a complex substrate. It is not possible to apply this approach towards calculating signal attenuation for a substrate that represents several axons. Most examples work on a single axon element modelled as a primitive two-compartment disk system, with no particles placed in the ECS. Numerical methods are excellent for evaluating on a simple geometry, but their true power comes from their role in model refinement when developing an MC model.

2.4.2. Monte Carlo methods

First proposed by N. Metropolis et al in 1947, Monte Carlo methods employ stochastic modelling to solve problems which may be deterministic in nature [68]. They are useful for systems with many coupled degrees of freedom [69]. The distinctive characteristics of MC algorithms are the reliance on randomness and repeated sampling. They typically consist of a fixed number of points which traverse an n-dimensional grid which represents a relationship between variables subject to a set of rules, known as random walks (RWs). This makes them particularly suited to simulating Gaussian processes, where the results are expected to converge due to the central limit theorem [70]. They are less susceptible to the curse of dimensionality, making them scale well with increased dimensions [71]. For our usage case, we wish to model restricted Brownian motion, so we can impose boundary conditions (BCs) across the geometry of the microstructure [72]. These conditions govern the movement between compartments and ultimately permeability.

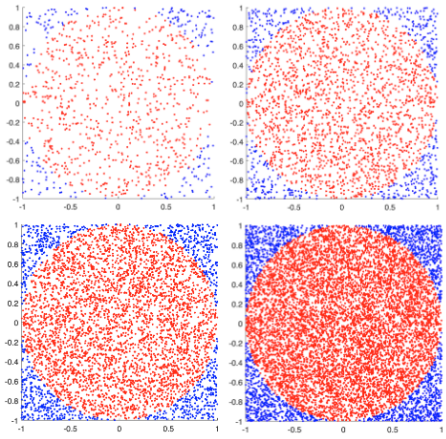


Fig 2. A Monte Carlo estimation of pi in MATALB using n = 1000, 2500, 5000 and 10,000 points. The ratio of points within a defined circular region rapidly converges to $\pi/4$ as $n \to \infty$.

Parameter choice is essential for sensible convergence, so the simulation must be validated using another method for ground-truth [73]. As previously mentioned, the simple two-compartment model used in the Grebenkov framework is a good foundation to develop the MC simulation from. Simulation parameters such as step size, time-steps, and algorithmic complexity all play a role in the quality of the MC model. The analytical model can be used to fine-tune the simulation until the error is within an acceptable range. In practise, opposite is true. MC methods are commonly employed as a ground truth for validating microstructure models in dMRI due to their robustness [74]. Initially trialled in 1990 [75], they have since enjoyed recent technological advancements. Notable models include the Camino toolkit (an open source framework developed at UCL) [76], MCSD (a MATLAB implementation) [77], RMS (3D neuroimaging) [78], DMS (diffusion microscopist simulator) [79] and NIRS (an application of validating analytical equations using an MC method) [80]. Through using a well-tuned MC simulation, the inverse problem of determining the microstructure beyond conventional resolutions can be solved heuristically [81, 82].

3. Methodology

We start by introducing key concepts in diffusion MRI and the limitations in conventional experiments. We lay out the foundations for creating a dMRI simulator in MATLAB, using a Monte Carlo loop for modelling restricted Brownian motion a random walk. We then set up a framework for evaluating the performance of a simulator in terms of accuracy, computation time and consistency of results. This includes investigating the effects of parameters and their effects on convergence, as well as finding the optimum set of parameters which gives the best tradeoff for our metrics. This framework can be used to analyse new methods which improve performance: Quadtree search structure, GAFRW and vectorisation. We suggest means of implementing these improved methods in our simulation, laying out the pseudocode and predicted performance gains in terms of big-O notation.

3.1. Simulation environment

We can discretise signal attenuation in (7) by considering N_t timesteps of size dt and N_s spins [83]:

$$\frac{S}{S_0} = \frac{1}{N_s} \sum^{N_s} e^{-i \sum_t^{Nt} \phi(t) dt} \tag{13}$$

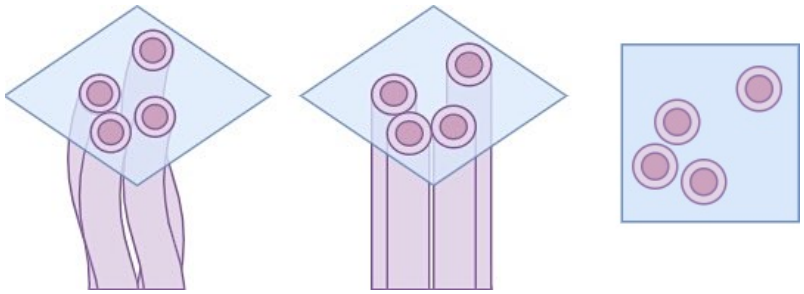


Fig. 3. The schematic for modelling axons as a cylindrical approximation of curvilinear geometry, with the 2D slice as the final result.

Since we are interested in anisotropic diffusion, we need only study the phenomena in 2D. We consider the plane bisecting axon cross sections in tissue. At any orientation, there will be tracts perpendicular to the plane. Random walks are easier to trace than transient 3D ones, and as aptly coined by the mathematician Shizuo Kakutani (author of several contributions to random walks [84]) “a drunk man will find his way home, but a drunk bird may get lost forever”. We define distance travelled dx within a single timestep dt as:

$$dx = \sqrt{4Ddt} \tag{14}$$

The expressions (13) and (14) form the basis of the Monte Carlo random walk. The microstructure can be modelled as a set of non-abutting cylinders, representing myelinated axon geometry. We can take a 2D slice of the cross-sectional plane to use as our spatial domain. Whilst this is an oversimplification of nerve hodology, it is suffice for our simulation. We want to develop a portable foundation to scale for more complex geometries, using the MATLAB suite which is well-optimised for matrix calculations [85].

We use the *diff_sim* framework as a starting point for the simulation [86]. We start by defining a grid of N_{ii}^2 pixels as the spatial domain, and construct a corresponding integer mask **I** to represent compartment geometry upon it. This rasterises the microstructure to the grid domain, and we can utilise vectorisation (linear indexing) to streamline search and update operations, similar to using binary markers [87]. This indexing process is not unlike how MATLAB stores matrix elements in memory [88], reducing the time complexity of compartment checking from $O(n^2)$ to $O(n)$, and the conversion between index and grid coordinates is handled by *maskPos* in 8.2.1. Each pixel is ‘painted’ by assigning it a value in **I** denoting the compartment it belongs to (i.e. ECS, ICS). Compartments are assigned a unique diffusivity coefficient D, and every concentric compartment pair a, b is assigned a commutative permeability coefficient $\kappa_{a,b}$. This is used in the construction of a virtual barrier restricting diffusion between compartments through change between adjacent pixels in adjacent walks. The parameters D and $\kappa_{a,b}$ are used to evaluate transmission probability P_T which ultimately decides if a particle passes through the aquaporins or is reflected.

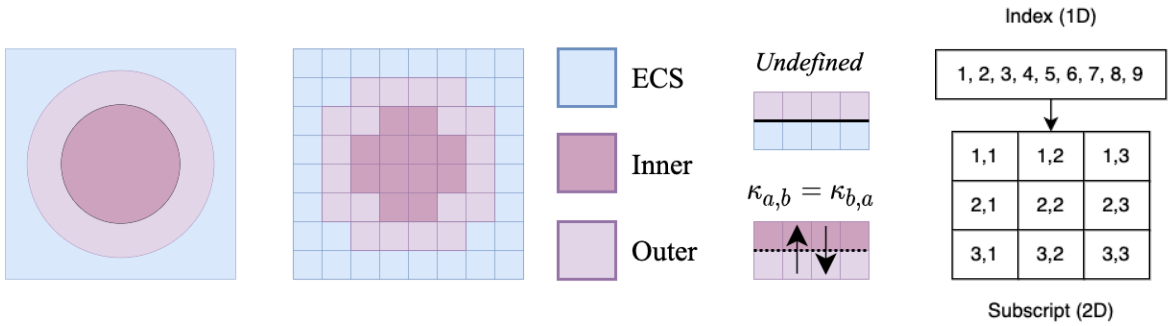


Fig 4. The process of rasterising the disk geometry onto grid (left) and the schematic of vectorisation (right).

3.1.1. Parameters

We will now formalise the two types of parameter used in the Monte Carlo simulation: **physical**, and **simulation**. Physical parameters involve constants and variables defining the experiment as a whole, such as the PGSE sequence or microstructure scaling in μm. These values are obtained from histological sources and are kept consistent between simulations. Scan parameters (also known as scan parameters) are unique to each simulation and are adjusted to yield results.

Type	Parameter	Typical value/range
Physical	γ — the gyromagnetic constant	2.675×10^8
	ra — Mesh grid physical scale.	$2.5 \times 10^{-6} - 1.0 \times 10^{-5}$ m
	δ — Duration of a single pulse.	50ms
	Δ — Duration between two pulses.	100ms
	G_s — Incremental gradient strength range.	$0 - 0.5T$
	κ — Permeability coefficient.	10^{-5}ms^{-1}
	D — Diffusivity coefficient.	2.0×10^{-9}m^{2}s^{-1}
	α — Radii Γ distribution shape parameter.	2.331
	β — Radii Γ distribution rate parameter (inverse scale parameter).	1.447×10^6
Simulation	N_{ii} — Number of side pixels in the grid.	$10^2 - 10^4$ (unstable beyond 10^5)
	N_{rw} — Number of random walkers.	$10^2 - 10^6$
	N_t — Number of time-steps.	$10^3 - 10^5$ (10^6 prone to scaling)
	N_{Gs} — Number of gradient data points, $\propto S$ data points	500 — Little effect except x-axis
	C_{pop} — Number of circles to attempt to populate substrate with.	$10^2 - 10^3$, dependent on N_{ii}, C_{sf}
	C_{scale} — Scale factor of circle radii. Important for scaling with N_{ii}.	$1 - 10$, or a fraction of N_{ii}
	$C_{spacing}$ — Imaginary radii drawn from circles to avoid touching.	1.1

Table 1. The parameters are divided into two sets: simulation and experiment.

The scan parameters N_{ii}, N_{rw}, and N_t, are characteristic for the Monte Carlo model used. N_{Gs} is the precision parameter and is proportional to the number of signal data points generated. It is fixed at 500, due to little improvement in our measured metrics with higher values.

3.1.2. Creating the microstructure

Our approach is based on the Kärger model, using the two-compartment disk chamber to represent diffusion in a myelinated axon [89]. We model the geometry as a pair of concentric disks each representing a diffusive compartment. We define the masking formula for this geometry in **I** as:

$$I = D_1 \leq \mathbf{R_1} + D_2 \leq \mathbf{R_2} \tag{15}$$

Here, **R** is the set of physical radii with $\mathbf{R_1} < \mathbf{R_2}$ and D_1, D_2 the disk formulae. Using groups of disks the expression in (15) can be repeated across the substrate, with slight variations in radii to represent nerve tracts. During the random walk process, we can avoid particles 'falling' off the grid after reaching the edges, by wrapping them around the grid with periodic boundary conditions. The microstructure must therefore be

periodic i.e. a unit cell of tissue composition. The precise arrangement of unit cells is more trivial than the shared characteristics between them, so we can assume tiling in the long spatial domain limit. The stochastic nature of random walks ensures each cell generates a unique value. We formalise the definition of the periodic microstructure based on the CHARMED model [90]:

- The substrate consists of N_D two-compartment concentric disks
- The disks are non-abutting when closely packed
- The placement of disks is periodic to ensure a unit cell

The process for placing disks across the grid is given by the circle packing algorithm, within the *createSubstrate* (8.2.7) function.

1. Generate N_D descending radii **R**, drawn from the gamma distribution
2. Starting with the largest in **R**, propose coordinate **C** on the grid for the centre
3. Subtend a disk of area $R^2\pi$ at **C**, wrapping any remaining area outside the grid to preserve periodicity
4. For each successive disk, check the area does not intersect with existing disks
5. Repeat steps 2-4 until the substrate is populated with approximately N_D disks[4]

The external module *bubblebath* generates the set of **R** and **C** for the above steps [91]. The function *createCirclesMask* applies (16) across **I**, mapping the radii and centres onto the domain [92]. To make disks two-layered, a second disk D_2 can be placed at C during steps 2-3, with a different radius to D_1. In axon geometry, the ratio between radii R_1, R_2 is known as the g-ratio. A value of 0.6 is typical in healthy white matter, while 1.0 represents no myelination [93]:

$$\text{g-ratio} = \frac{\mathbf{R_1}}{\mathbf{R_2}}, \text{ where } \mathbf{R_1} < \mathbf{R_2} \tag{16}$$

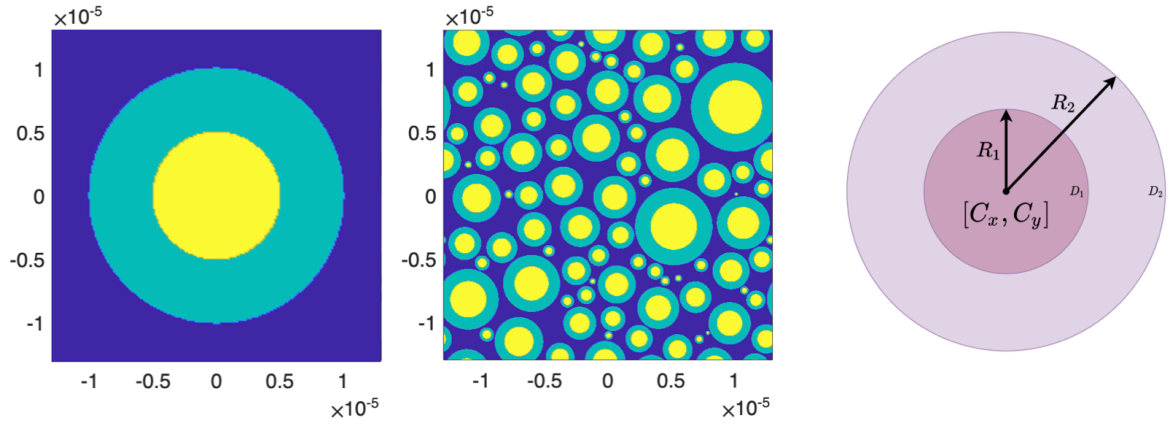


Fig 5. Sample substrates of the two-compartment disk structure (left) and periodic microstructure generated by *bubblebath* (centre), as well as an illustration of the geometric disk parameters for a single axon (right).

3.2. Monte Carlo implementation

We can use a set of Monte Carlo random walks to simulate Brownian motion, with each path corresponding to a particle's diffusive trajectory. We introduce a simple random walk using fixed step sizes *dx* and *dt*. The algorithm begins by scaling *N* points randomly across the domain, such that there are N_p remaining particles; particles placed in non-diffusive regions are deleted. An array ϕ of size N_p is initialised to keep track of accumulated dephasing from each particle. Then, the following loop is repeated for N_t timesteps:

1. Input a set of N_p positions **X** for points across **I**
2. Propose set $\mathbf{X_n}$ of N_p new positions at length dx using radius search function *randCirc* (8.2.2)
3. Make boolean mask **Ie** for particles which did not switch compartments

[4] There may be some under/overcounting due to periodic wrapping. This results in slightly more or less than N_D total disks.

4. For every particle x switching compartments, such that $a \neq b$, evaluate P_T and change $\mathbf{le}(x)$ as required

5. Use $\mathbf{le}$ as a mask to update positions such that $\forall\ \mathbf{le}(x) = 1$, $\mathbf{X} = \mathbf{X}+\mathbf{X_n}$

6. Wrap particles which 'fell' off the grid using periodic BCs, to avoid deletion

7. Input dephasing contributions per particle per timestep to ϕ

At acquisition time ($t = N_t$) the signal attenuation S is returned. This is plotted against b-value to illustrate the evolution across the substrate as the gradient is temporally varied. We can calculate the ADC by using pairs of b-values. Whilst straightforward to implement, this method scales poorly as the magnitude of parameters increases. We explore this in our experiment, with the MATLAB code for the random walk process included in the *diffSim* (8.2.6) function.

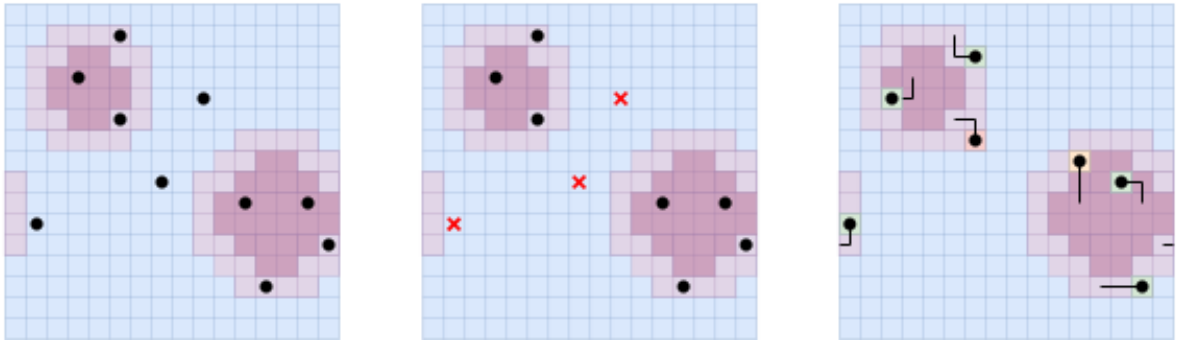


Fig 6. The evolution of the random walk algorithm. First, a substrate is rasterised on the grid. Points are placed randomly (black) with ones placed in non-diffusion regions deleted (red). The random walk process is given for two timesteps: permitted movements are green, prohibited red and conditional amber. Note periodicity at the substrate edges.

3.2.1. Collisions and permeability

Axon permeability is essential to understanding the health of the myelin sheath. Unmyelinated neurones propagate electrical signals at a lower speed than insulated ones [94, 95]. Typically, the permeability of a surface is a measure of particle flux through it, and we can express this stochastically as a transmission probability P_T for a given particle. Using indexing as a preferred approach to for-loops, we generate N_p Bernoulli random variables for each particle x during each timestep such that Px $\in$ [0,1]. We then evaluate them against P_T at each barrier encountered. At most, there are two barrier types, so a permeability mask can be used to avoid calculating P_T every time.

The derivation for P_T is adapted from the implementation of Robin BCs at the membrane boundary:

$$p = \kappa_{a,b}\sqrt{\frac{dt\pi}{D_a}} \tag{17}$$

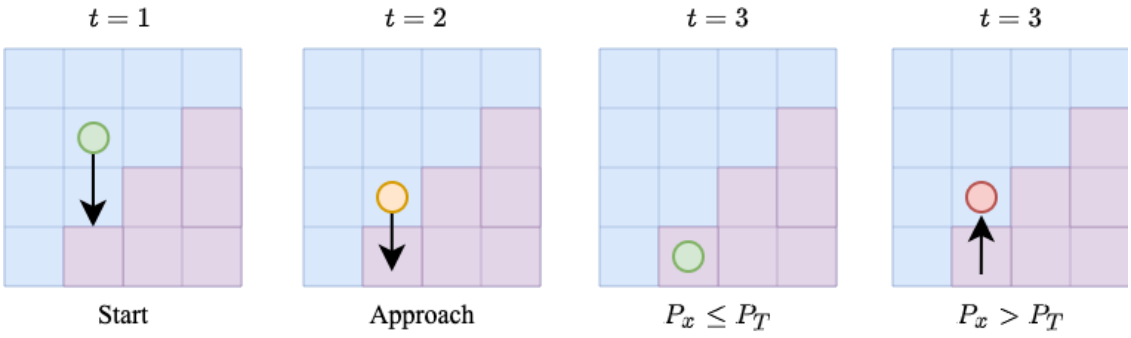


Fig 7. The diffusion process for a single particle. At $t = 1$ the particle is moving towards the barrier, which it approaches at $t = 2$. The transmission probability P_T is evaluated at $t = 3$, which results in either a transmission or a reflection.

The condition for a successful transmission of particle x into the next compartment is $P_x \leqslant P_T$, with the particle reflected for $P_x > P_T$. We need not concern about ballistics for handling reflections in the mask based approach, since the particle merely 'takes a step back' in the direction normal to the boundary. In fact, the coordinates of the reflected particle do not change at all, since the steps cancel out.

3.3. Experiment outline

The aim is to develop a robust framework for evaluating Monte Carlo methods. We begin by finding the optimum parameters through trialing a set of values, and ranking them based on three metrics: relative error, computation time, and consistency between repeat results. These parameters are used as a gold-standard for investigating the effects of simulation parameters individually and together.

We divide our experiment into two simulations. *Simulation 1* uses the two-compartment disk chamber for model validation. *Simulation 2* is performed on the periodic microstructure substrate, where there is no ground-truth answer available. Validation is carried out by comparing readings for S generated by the analytical model and that of Monte Carlo one. It is important to note that the analytical model does not generate a 'perfect' solution, but provides a reliable set of results that are adequately used for ground-truth for this purpose. In the second simulation, we evaluate the results shortlisted by validation for consistency, selecting the best answer on this basis. By trialling enough parameters, we can present a reliable and accurate prediction of dMRI signal attenuation.

3.3.1. Initial parameters and number of experiments

We use the following set of physical parameters, leaving them unchanged for the duration of this study: δ = 50ms, Δ = 100ms, $0 \leqslant G_s \leqslant 0.5$T, set $D = 2.0\times10^{-9}$m^{2}s^{-1} as the diffusivity coefficient of water at room temperature and $\kappa = 10^{-5}$ms^{-1} as the axonal water permeability. Further validation can be carried out by adjusting physical parameters in any derived future works.

We wish to evaluate as great a range of scan parameters as possible, to evaluate the method most effectively. We can take a 'head start' using insight from similar studies, introducing the following quantity:

$$U = N_t \times N_p \tag{18}$$

U is a measure of complexity in terms of the number of updates and paths, and is aptly defined by Hall and Alexander et. al. as the number of operations in a Monte Carlo loop [73]. In most cases, $N_p \approx N_{rw}$, aside deletion of points in non-diffusive regions. We use vectorisation to represent each particle to reduce algorithmic complexity such that performance is almost entirely dependent on U. The same paper mentions a recommended value of $U \geqslant 10^8$ for an effective simulation, so we consider the configurations within and just outside that range. We also trial N_{ii} for a distributed set of resolution sizes. The initial choices are summarised below:

Parameter	Choice	Justification
N_{ii}	200, 500, 1000	Variety of spatial resolutions to test N_t and N_{rw} with.
N_{rw}	10^2, 10^3, 10^4, 10^5	Logarithmic scale for a wider range.
N_t	10^2, 10^3, 10^4, 10^5	Note that above 10^5 computation becomes impractical.

Table 2. The set of scan parameters to be trialled for each simulation.

We define the total number of experiments for each simulation E as follows:

$$E = V^v \times R \tag{19}$$

This is an important quantity for eliminating unnecessary computations, where V represents the number of parameters to try (in our initial case, three), v the number of distinct values in each, and R the repeat experiments required (for *Simulation 2*). The aim in this section is to minimise E through utilising as much prior knowledge as possible. We can skip parameter combinations of N_{rw} and N_t that yield a U within our desired range as a measure of reducing E. Independent tests carried during development found the best value was $10^8 \leqslant U < 10^9$ with the results diverging below this range and taking too long to reasonably compute above it. We eliminate the following combinations:

N_{rw}	N_t	U
10^2	10^2, 10^3, 10^4, 10^5	10^4, 10^5, 10^6, 10^7
10^3	10^2, 10^3, 10^4	10^5, 10^6, 10^7
10^4	10^2, 10^3,	10^6, 10^7
10^4	10^5	10^{10}

Table 3. Scan parameters that can be omitted. A total of 10 experiments can be skipped.

This brings our initial value of $E = 48$ (with $R = 1$ for *Simulation 1*) to $E = 38$.

3.3.2. Performance metrics

To analyse our results, we use three parameter metrics. We can classify the quality of our model's results by using all three to make a holistic judgement of which results are best.

MRAE

The mean relative absolute error is a summary of the relative error between the simulation and analytical results, and is a key metric in validation. The relative error equation is trivial, but is included for completeness, and is defined for two sets of S, S_m and S_a (corresponding to simulation and analytical readings) as follows:

$$\eta = \left| \frac{S_a - S_m}{S_a} \right| \tag{20}$$

Time complexity

This is a measure of computational efficiency, through a plot of the execution time for each experiment against the parameter that is being investigated.

Initially, for finding optimal parameters, we plot the set of results with the 'best' MRAE readings as a horizontal bar chart in ascending order. This makes it clear for us to identify the best and worst performers for categorical data (parameter configurations). When investigating parameters separately (and in pairs) the time is plot as a graph to aid identifying how complex it is. Time complexity is illustrated in **Fig 8** [96].

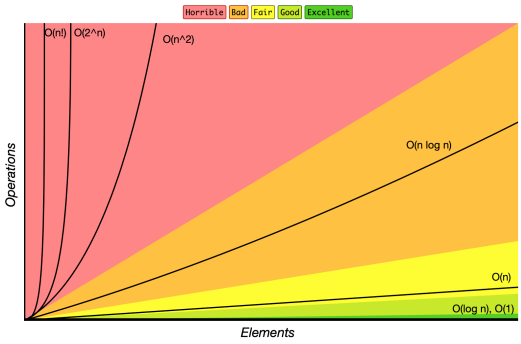


Fig 8. Time complexity as a scale of operations and elements. The lower the order of *n*, the better. Source: *bigocheatsheet.com*

Mean SD

We take the standard deviation (SD) between error results about the MRAE, for each repeated reading. We then take the mean of this value, to give us the mean SD. This is a measure of consistency between readings, given the parameters are the same, and can be used to filter lower quality results from optimal ones. It is used extensively in *Simulation 2* to assess the performance of convergence where analytical results do not exist. Since Monte Carlo methods are stochastic, repeated convergence to the same point as per the law of large numbers corresponds to an effective set of parameters [97].

3.3.3. Simulation 1: validation with an analytical model

Aims: Refine model parameters using an analytical model as ground-truth.

Objectives: Perform dMRI simulation on a two-compartment disk chamber. Measure relative error between simulated and analytical results. Refine the Monte Carlo method and its parameters until error is reduced. Investigate scalability of scan parameters and their implications on time complexity. Determine best set of parameters with a balance of accuracy and computation time.

We use the MATLAB implementation of D.S. Grebenkov's works in [62] and [63]. The ML (multi-layered) framework is an optimised solver for signal attenuation in a semi-permeable two-compartment (Kärger)

structure [98]. The physical dimensions of the inner and outer disks are $\mathbf{R_1}$ = 2.5μm and $\mathbf{R_2}$ 5.0μm respectively. This corresponds to a g-ratio of 0.5, which is not precisely 0.6 but is straightforward for calculations. It is important to note that no ECS is utilised in this experiment, as it is not considered in the analytical model, so we delete points placed in there. We plot S against b-values for both models, along with the corresponding relative error. We utilise the findings of this experiment for *Simulation 2*.

3.3.4. Simulation 2: convergence on a periodic microstructure substrate

Aims: Evaluate stability of the model on an unseen microstructure.

Objectives: Perform simulation over the periodic microstructure substrate, representative of a repeatable unit cell in white matter tissue. Repeat readings for each parameter combination R times, obtaining the mean SD as a measure of consistency. Suggest the best parameter combination, as a trade-off between speed of convergence (time) and bias (error and consistency).

We fix the number of disks in the microstructure as C_{pop} = 100, a straightforward value easy to check for 'missed' disks during substrate creation. The substrates used between readings must be kept as similar as possible to ensure a fair test. For repeat readings this is straightforward, however, the circle-packing algorithm is rasterised wrt N_{ii} and must be adequately scaled. There is no need to increase C_{pop} as our assumption of the unit cell substrate means that there are infinitely many disks for a continuous tiling, so we manually adjust C_{scale} in order generate substrates which are generally the same (8.6). We set $C_{spacing}$ = 1.1 for enough minimum space between the disks such that they are non-abutting but can also form clusters. Our physical scale is 10μm, and the parameters for Γ_R are α = 2.331 and β = 2.2 by using a similar method to Hall and Alexander [73], fitting the distribution to axon-fibres in a post-mortem study [99]. For consistency, R = 10 for each parameter set trialled and we intend to use five best-performing configurations from *Simulation 1* to study. The predicted number of experiments is E = 50.

3.3.5. Parameter investigations on performance

We use the best prediction from *Simulation 2* to provide gold-standard values for N_{ii}, N_{rw} and N_t. In turns, we run experiments for each parameter changed incrementally and investigate the same metrics as before. The reasons for doing this are twofold: we are interested in highlighting which parameters can help the simulation converge to the best result (including any missed values that could not be evaluated together due to time complexity and technological restraints) and also to identify weak points in the simulation that can be improved upon.

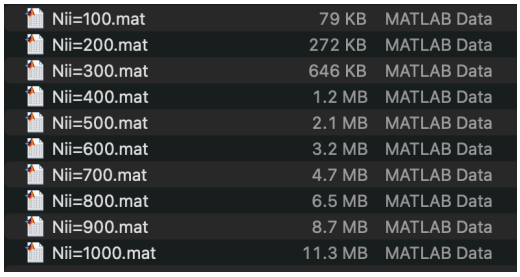


Fig 9. A snapshot of the file sizes for experiments with increasing sizes of N_{ii}. There is an O(n^2) relationship, since N_{ii} is squared for the total grid size.

Independent analysis

Keeping other parameters fixed, we investigate the effects N_{ii}, N_{rw} and N_t adjusted incrementally for given ranges. We use $N_{ii} \in [10^1,10^3]$, $N_{rw} \in [10^2,10^5]$ and $N_t \in [10^2,10^4]$. Logarithmic scales are utilised for all three to give a greater range of analysis. Graphs of MRAE and time complexity will be plotted per parameter, and we will utilise these to investigate the effects of increasing each parameter. We must take care when considering the range of N_{ii} since the file sizes scale tremendously (**Fig 9**).

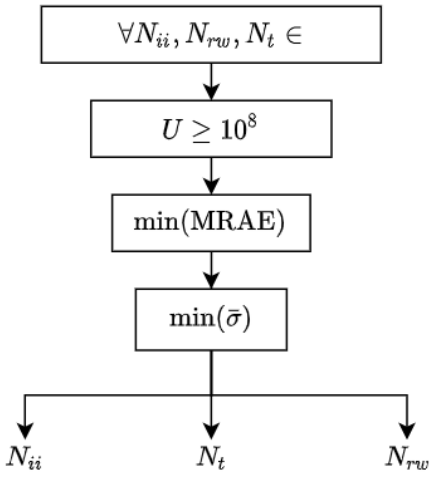


Fig 10. The schematic for selecting the optimum configurations. Each process narrows our search space until we have the best performing values for all three parameters.

Bivariate analysis

We can compare the performance of the program from adjusting parameters in pairs, by plotting their combined results as a *scatter* plot on the same axis and observing the differences between the lines of best fit. We trial N_{ii} and N_{rw}, N_{rw} and N_t, N_{ii} and N_t, to gain a picture of the interrelationship between these parameters. It is important to use the same range for both parameters when they are being investigated, and thus we restrict our evaluation domain to the following: N_{ii}, $N_{rw} \in [10^2,10^3]$, N_{rw}, $N_t \in [10^2,10^4]$, N_{ii}, $N_t \in [10^2,10^3]$.

4. Results and discussion

4.1. Finding optimal parameters

4.1.1. Simulation 1: error and time complexity

For the simple two-compartment structure the ML was initially tested, with the findings being consistent with the figures in [63]. We used this structure to evaluate S_a and S_m, found the five configurations with the lowest MRAE and plotted the relative error for N_{Gs} points. The most important results are given below, with the remaining figures in (8.5.1).

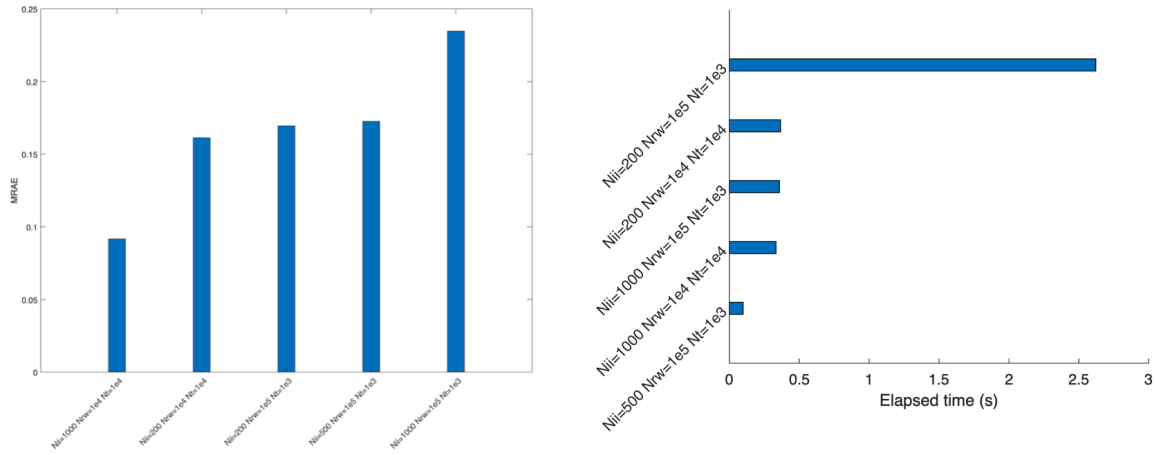


Fig 11. The parameter combinations with minimum MRAE, and the corresponding runtimes.

The *best* performing configuration out of these was $N_{ii} = 10^3$, $N_{rw} = 10^4$ and $N_t = 10^4$ (MRAE = 0.092, t = 0.4s). We chose this value from a holistic analysis of MRAE and runtime: this result had the lowest MRAE in the entire experiment, and although it was not the best-performing result, the differences between quicker configurations were marginal (as observed in **Fig 11**).

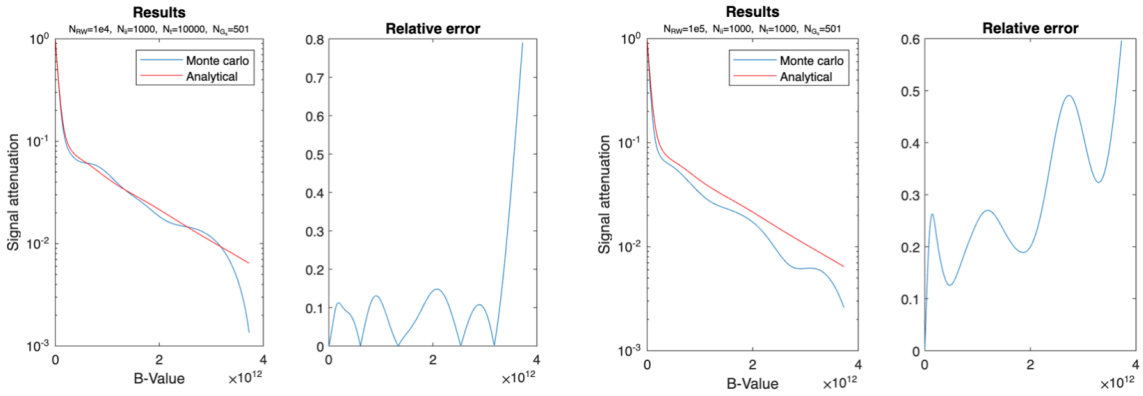


Fig 12. Plots of the best (left) and worst (right) performing optimal configurations.

4.1.2. Simulation 2: consistency

We also evaluate these configurations on the periodic microstructure substrate, to ascertain which converges the fastest on a 'realistic' substrate, as well as performing the most consistently between repeat readings. The most consistent result was found for $N_{ii} = 200$, $N_{rw} = 10^5$ and $N_t = 10^3$ but this did not necessarily mean it was the best answer, as it was also the worst in its class for elapsed runtime (26.6s). We must bear in mind the substrate now takes water molecules in *both* ICS and ECS to reflect realistic modelling in a slice. Lower values of N_{ii} with higher N_t updates reduce the effects of over-stepping (lower SD) but do not imply the result will be more accurate.

We instead chose N_{ii} = 10^3, N_{rw} = 10^5 and N_t = 10^3 for the optimal configuration. This set performed moderately well (17.8s) whilst yielding a far lower SD than its faster neighbours ($\bar{\sigma}$ = 0.122 to 3 s.f.). The result performed similarly in *Simulation 1* with 0.4s (the median of elapsed times), however it is important to note that this result also yielded the highest error (MRAE = 0.244). Consistency suggests the error stabilises at this value, so we will consider this result still valid as the optimum value set for the purposes of our evaluation.

N_{ii}	N_{rw}	N_t	Mean SD $\bar{\sigma}$ (5 s.f.)
200	10^5	10^3	0.12133
1000	10^5	10^3	0.12206
500	10^5	10^3	0.12515
1000	10^4	10^4	0.15156
200	10^4	10^4	0.15904

Table 4. The mean SD of the best-performing candidates, in ascending order.

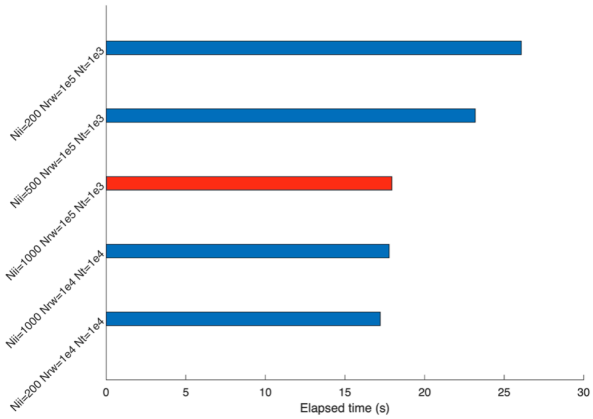


Fig 13. The plot of elapsed execution time per configuration. The visual approach quickly shows optimal results across metrics.

The sample plots of S with R = 10 for each of the five configurations are given in (8.5.2).

4.2. Independent parameter analysis

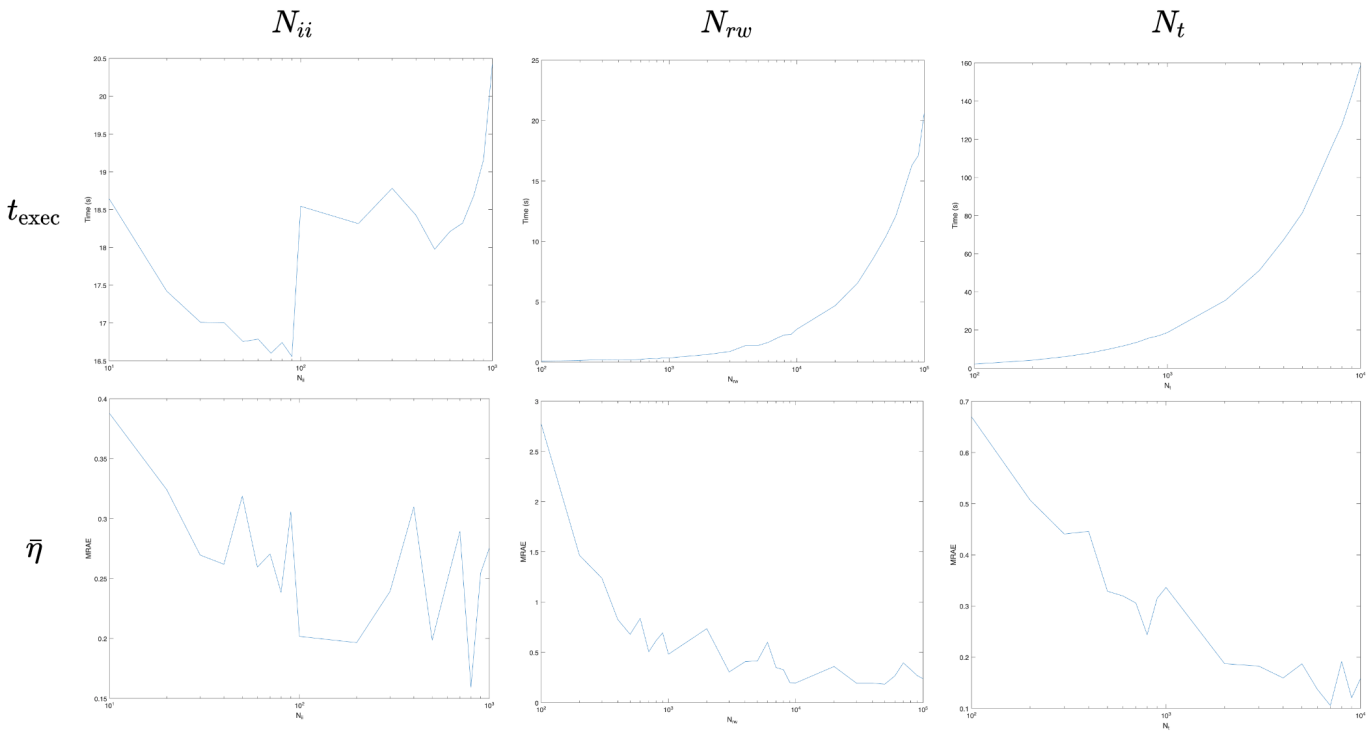


Fig 14. The resultant plots of error and computation time through incremental values. All plots are on a $\log_{10}$ scale, for a larger range to analyse.

All three quantities showed a gradual decrease in error with higher input parameters (whether this is inversely quadratic or linear cannot be directly determined). Time complexity scaled quadratically, as is expected of the basic O(n^2) method. The reason for the 'bump' in N_{ii} execution time is likely due to the increased order of magnitude and its implications on N_{ii}^2. All parameters displayed a behaviour as expected. Ideally, we would wish to increase the rate of MRAE decrease, whilst reduce the increase of t_c.

Variable and range	Fixed	Number of data points used
$N_{ii} \in [10^1,10^3]$	$N_{rw} = 10^5$, $N_t = 10^3$	19
$N_{rw} \in [10^2,10^5]$	$N_{ii} = 10^3$, $N_t = 10^3$	28
$N_t \in [10^2,10^4]$	$N_{ii} = 10^3$, $N_{rw} = 10^5$	19

Table 5. A summary of the parameters used in the independent analysis.

4.3. Bivariate parameter analysis

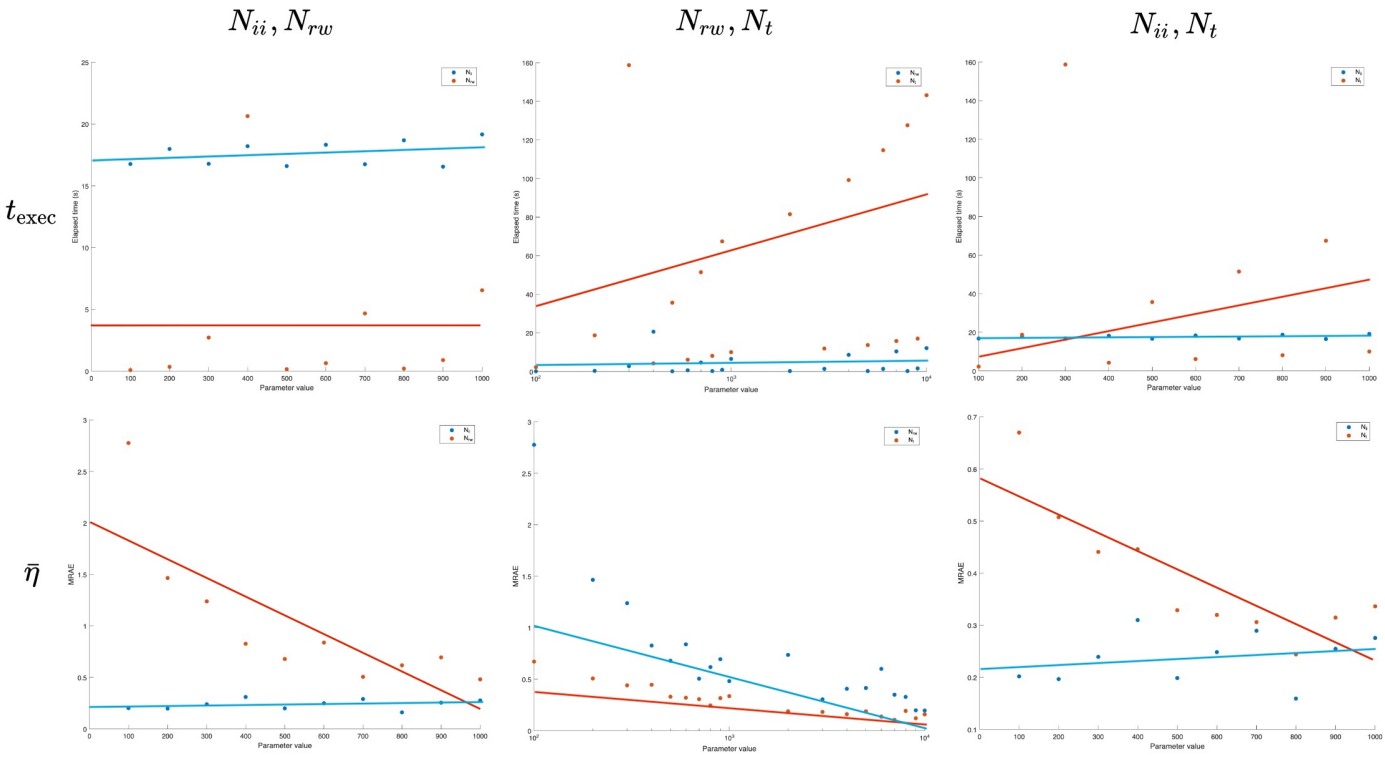


Fig 15. Plotting the correlation between computation time (top) and MRAE (bottom) for all three combinations. Plots are on a linear scale except for N_{rw}, N_t which uses a $\log_{10}$ scale. A least-squares line of best fit is drawn to approximate the correlation.

Both N_{ii}, N_{rw} had a linear impact on computation time; albeit negligibly. MRAE decreased with more paths, whilst showing a gentle increase with spatial resolution. In both cases, substrate resolution appeared to have a negative effect on time complexity and error. The most prominent consequence of N_{ii} appears to be the file size of the substrate and simulation variables that were used (**Fig 9**). We conclude N_{ii} has little influence over the simulation performance except when needing to scale with N_{rw}.

The combination of N_{rw}, N_t, is ultimately the simulation complexity U and we were able to investigate a higher value range due to almost negligible effects on storage space in comparison. The reduction in error is almost this as much from increasing N_{rw} compared to N_t indicating an important finding: the amount of particles evaluated affects the results far greater than the number of updates where they are evaluated.

Variable and range	Fixed	Number of data points used
N_{ii}, $N_{rw} \in [10^2,10^3]$	$N_t = 10^3$	10
N_{rw}, $N_t \in [10^2,10^4]$	$N_{ii} = 10^3$	19
N_{ii}, $N_t \in [10^2,10^3]$	$N_{rw} = 10^5$	10

Table 6. A summary of the parameters used in the bivariate analysis.

Finally, we investigated N_{ii} and N_t which led to trivial results. Increasing N_{ii} with N_t had a similar effect on error, however both were inversely proportional to each other. For time complexity, again, the number of updates had a far greater influence than the substrate resolution.

5. Conclusion and further work

5.1. Interpretation of the results

5.1.1. Independent analysis

Time complexity was generally found to be $O(n^2)$, with a quadratic increase in the time taken for the simulation to produce results while parameters were increased. Error was reduced quadratically with an increase of N_{rw} and somewhat linearly elsewhere. More intermediate values for N_{ii} and N_t could clarify this, and a higher range would clearly expose the true nature of the relationship.

5.1.2. Bivariate analysis

For the time complexity there was a correlation in the same direction between all parameters, however the effects of N_{rw} were much more prominent than that of N_t. Increasing both highlighted how more effective tuning N_{rw} would be, with a proportionally lower MRAE when compared to reducing N_t by the same amount in elapsed time.

5.1.3. Weak points identified

The main disadvantage to the conventional Monte Carlo method was the *time stepping error*. This is due to the fixed-timestep discretisation of the *S* expression, and is inversely proportional to the size of *dt*. In the worst case, we consider an experiment with $N_t = 1$, and *dx* the size of the substrate. The gradient coefficients would cancel out (and successively dephasing and *S* readings), since the PGSE sequence is a step function $\in$ [-1, +1]. Ultimately, a higher N_t would reduce dt, but as shown an $O(n^2)$ algorithm is less than ideal for scaling with larger numbers. Using a high value of Nt with a relatively lower N_{rw} and N_{ii} can introduce the phenomena of *overstepping*, however. Overstepping is caused when there are too many timesteps with not enough particles to be sensibly simulated by them; an extreme case would be a grid with four pixels and using two particles: for $N_t \rightarrow \infty$ the random walks become less ‘random’ as repeated steps are replicated infinitely. A low number of permutations on the grid would be caused from lower spatial resolution and path sampling.

5.1.4. Conclusions and questions raised

Selecting optimum parameters was less about finding the ‘best’ values and moreso of scaling them appropriately. With N_{ii}, the quantities N_{rw} (and successively N_t) would need to be increased in order for the results to remain sensible. This means that a more accurate simulation would become computationally expensive to an exponential degree, since combining the time complexities for all three parameters (assuming they are uniformly scaled) would result in $O(n^2)^3 \approx O(n!)$, the worst possible performance for any algorithm. Thus careful consideration must be taken in *which* proportion each parameter would need to be increased. We have identified N_{rw} as the parameter with the highest impact, so we need only adjust N_t as a fraction (an order of magnitude or two lower) of this. N_{ii} was purely down to the substrate requirements, i.e. survival of the geometry of smaller disks during rasterisation.

5.2. Improved methods

We now suggest some methods for reducing the number of steps required which have little compromise on accuracy. The complexity is significantly reduced by omission of irrelevant steps and approximations over isotropic regions.

5.2.1. The case for simplifying steps: Quadtree structure

The quadtree structure is a means of drastically reducing the size of a 2D grid by merging isotropic components together. This way, the time and spatial steps during a Monte Carlo loop can be incremented as such to traverse entire sections of the grid in one iteration.

Quadtrees are derived from the 3D Octree structure and can rapidly decompose a large grid into a set of subdivisions based on characteristic properties encountered. Commonly used in *n*-body problems in physics (known as Barnes-Hut simulations) [100], quadtrees are a very efficient way of reducing the number of areas on a 2D space in which calculations are carried out. The grid is partitioned into quadrant areas of different sizes. Each quadrant can be split into a further set of quadrants, and so on. The decision for splitting a quadrant is made by ascertaining the level of detail within its area. Denser regions with more particles (or in our case, substrate geometry) will be represented by more quadrants to reflect that. If a given quadrant cannot fit an entire particle in one of its sub-quadrants, the splitting ends. This process vastly reduces the complexity of the 2D domain, since regions are given a level of detail proportional to their relevance.

An application of this structure in Monte Carlo dMRI modelling is in [79] where a 3D cube-splitting application (known as an octree) is employed to increase performance. To implement the quadtree structure into our environment, we use *qtdecomp* to decompose the substrate masking grid **I** dependent on the variation of the compartments per pixel. The result is a sparse matrix representing the reduced form of the grid in terms of quadrant corner vertices (bottom-leftmost per block). The decision not to develop a standalone function for the random walk was made to avoid local variables (which are harder to trace during debugging).

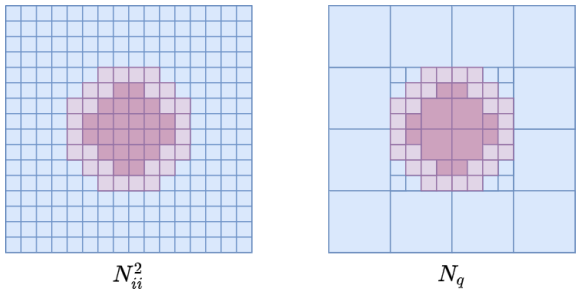


Fig 16. The quadtree decomposition applied to a two-compartment chamber.

=The most efficient representation of the reduced substrate is as a network. Each node represents the centre point of a block, with the connections between the nodes indicating the ‘available’ moves between adjacent blocks. A random walker placed on any given coordinate cannot translate diagonally between pixels, so it is important that only translations between adjacent blocks are permitted. An adjacency matrix is used to represent these relationships. An intuitive step towards using the matrix is that the edges can be ‘weighted’ dependent on their permeability. Each node is assigned the compartment of the majority of pixels within its block. It is important to keep track of the original dimensions for each block (a varying *dx* in powers of 2), so we summarise that information for each node. The calculation of dephasing becomes non-trivial as steps are skipped, so it is important to keep track of *dx* and *dt* for an effective accumulation.The random walk algorithm to traverse a network is given by *randomGraphMove* which is an adaptation of MATLAB code used to simulate random walks on the London Underground network [101].

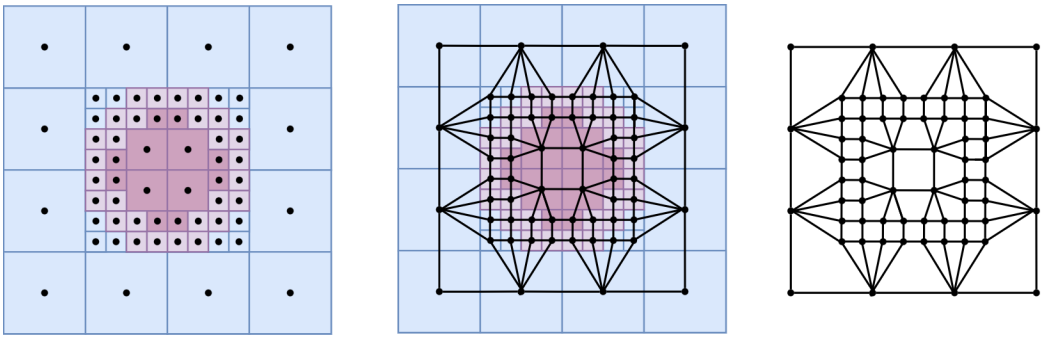


Fig 17. The process of generating a quadtree network: the centre points of each block represent a node, and the edges the available translations between them. The network can be further simplified by removing nodes in non-diffusion regions and the edges connecting them. A set of quadtree decompositions for different substrates for *Simulations 1 and 2* is included (8.3).

The obvious advantage of this approach is the reduction in the number of coordinates required to traverse. The quadtree structure is represented in memory as a *sparse matrix* (one which removes unnecessary zero-values from the data) which is much faster to translate O(Log*N*) versus the full grid O(N^2) with *N* being the distance between values. Whilst some extra computation will be used to convert the reduced grid into an adjacency matrix, the trade-off for convergence speeds and access to more accurate parameters is justifiable.

It is important to note that the quadtree network does not serve as a substrate for the original mesh grid. It is merely a lookup-table (or ‘cheat sheet’) containing the relevant positions that can be skipped where no interesting results would be sacrificed. The values of *dt* and *dx* would be updated accordingly to correspond with the spatial space skipping, in order to keep the dephasing calculation intact. The algorithm for creating

a quadtree structure and translating it can be found in 8.2.10. We also provide some initial results for simple usage cases in 8.7.

5.2.2. Skipping unnecessary steps: 'fast' random walks

Another form of improving the efficiency of the Monte Carlo loop is by introducing 'fast' random walks, namely the Geometrically Adaptive Fast Random Walk (GAFRW) algorithm devised by Grebenkov in 2011 [102]. This approach reduces the amount of calculations by, similar to the Quadtree structure, skipping unnecessary calculations during random walk steps albeit in a different way. The GAFRW considers a diffusing particle within a given compartment, and subtends a circle from that particle's position with a radius which is equal to the distance to the nearest 'obstacle' (another compartment location). Then, a point on that circle is selected at random, and the particle position is updated accordingly. This process is recursive, and is repeated for all particles.

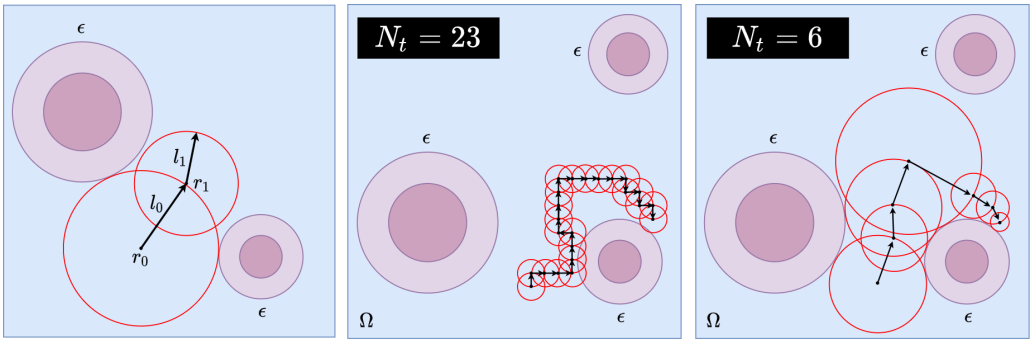


Fig 18. Left to right: the parameters of the GAFRW for the first two steps, the path of a diffusing particle and the steps required using the default random walk algorithm, and the equivalent process and steps using the fast 'random' walk. The same *randCirc* function is used, but the radius varies in GAFRW.

The dephasing becomes non-trivial as variable step sizes *dx* and *dt* sizes are used, and so the time a particle spends within a 'circle' (with isotropic Brownian motion assumed inside of it) must be estimated. Exit times are generated which estimate how long a particle will remain within a compartment, based on 'survival probability' for brownian motion at the centre point of the disk. To avoid complex numerical calculations, the exit times are generated using pseudo-random numbers. We can generate a set of 'exit times' for Brownian particles per circle, which correspond to the estimated duration it would spend before reaching the perimeter and represents *dt* at this stage accordingly. This time, the timestep loop is nested within the particle evaluation; by evaluating for every particle *then* for every timestep, the dephasing from various dt values can be summarised for the particle at the end of its diffusive time evolution. We can save this data to evaluate *S* at the end of the simulation. The untested code to implement into our Monte Carlo model is given in 8.2.11.

5.3. Future work

5.3.1. 3D implementation and DTI

Extending the work into 3D is simply a matter of multi-slice imaging. We can reconstruct the polygonal representation of nerve tracts by capturing segments of their geometry within equidistant planes bisecting the cross section. The results for each slice can then be combined. To gain a more accurate picture, we can repeat this process for several orientations, capturing many nerve fibres within the ROI and using our 3D model as a means to check our orientations are constructing a sensible image [6]. This is useful for Diffusion Tensor Imaging (DTI), where the 3D representation of nerve tracts can be used to create a *tractogram* [103]. DTI utilises a symmetric tensor which holds the coefficients of diffusion in each orientation. DTI can generate a time-sensitive map of diffusion, which is useful in functional MRI (fMRI) [7]. A key advantage of DTI is that it indicates the *direction* of diffusion, not just magnitude.

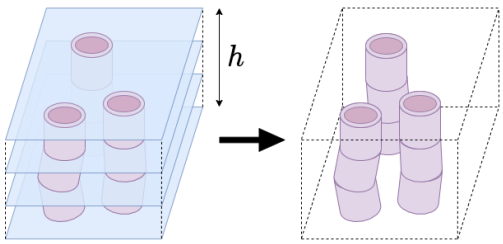


Fig 19. The process for constructing 3D axon models for DTI using multi-slicing.

5.3.2. *Real application: Ischaemic stroke detection*

The swelling induced from lesions in white matter is important for studying neurological disorders. A typical example of a common disorder involving lesions is the ischaemic stroke. The ischaemic stroke is the most common type of stroke — 87% of all strokes are reported as ischaemic strokes [104] — and is a result of blood clots restricting flow of blood and oxygen throughout the brain. Axons may undergo swelling from lesions caused by plaques as a result of atherosclerosis, or are clogged from broken-apart blood clots from atrial fibrillation. The ischaemic stroke is known to influence the ADC in a predictable manner, with it initially decreasing and then increasing over a short period after [105]. Empirical for this suggestions include additional obstruction of water molecules from cell swelling.

We can model this phenomena by considering a real-time microstructure deformation, dynamically increasing the radii as the random walk loop evolves. The particle positions must be consistent between each update of the substrate. It is important to note that the *Quadtree structure* in 5.2.1 is not suitable for modelling in this case. A practical application of scenario is given by Hall and Alexander, which considers deforming cylinders and handles the case for overlapping geometry through chord intersections (**Fig 20**, [73]). This paper uses the same Γ distribution of radii as in 3.1.2 so is straightforward to implement in our simulation. It is important to note that in our case myelinated sheaths will deform along with the rest of the axon, however keep their thickness (closing the gap on the inner part of the axon). Thus on **I**, component marked '1' are susceptible to deformation where those marked '2' are not.

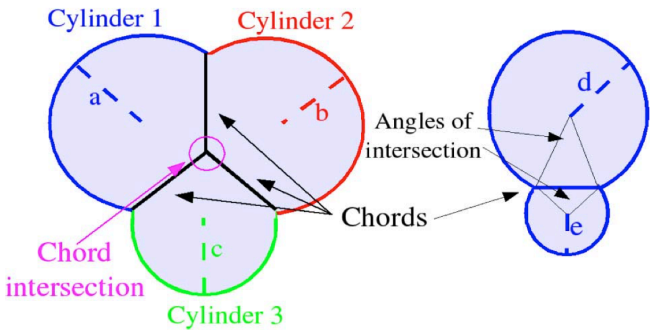


Fig 20. The chord intersection process for the deformed radii. Note that this case only considers a single compartment axon model; we would need to adapt for myelinated axons in ours. Source: *Hall and Alexander.*

5.3.3. *Parallelisation and cluster computing*

A huge advantage of Monte Carlo models for technology advancements in recent years, is that they are 'embarrassingly parallel' — that is, little to no effort is required to separate the problem into parallel tasks [106]. This is especially true for the case of our approach towards particle diffusion, since spin-spin interactions are not considered, so each diffusive path can be treated independently [107]. This yields tremendous performance increases for scaling N_{rw} and N_{ii}, eliminating the restrictions that must be considered for computing higher values of N_t. Using parallel computing, time dependence is solely based on N_t alone as each particle trajectory is computed separately. The study in [108] outlines a means of implementing Monte Carlo code for parallelisation, which is easily transferrable to our dMRI framework.

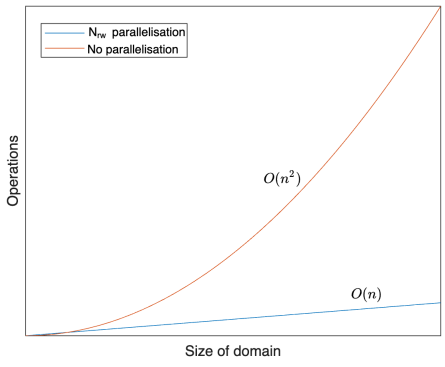


Fig 21. The projected improvement in time complexity with parallelisation as a function of the number of operations to the size of domain.

Originally, the plan was to evaluate the simulations on the CUBRIC computer cluster, a high-specification processing network for implementing parallel computing [109] using the Linux operating system. The code would be adapted for parallelisation by assigning a virtual thread for each particle in N_{rw} as well as taking advantage of the increased storage space for larger N_{ii} (data is stored *temporarily* on the cluster for the duration of the experiment before being deleted, permitting larger values to be stored for results generation, ideal for our usage case). Amdahl's Law predicts that the performance gains are proportional to the fractional improvements in the methods used [110], and thus by limiting time dependence in N_{rw} (and therefore N_{ii}) the overall problem for the standalone Monte Carlo method can be reduced from an $O(n^2)$ time dependence on N_{rw}, N_t to a single $O(n)$ dependence on N_t.

6. Critique of the project

6.1. Project versus proposal

The final project differed significantly from the original proposal. Due to a number of factors including an interruption of study (IOS) and different methodologies being trialled and used, we arrived to a different resolve. Instead of simulating the *ischaemic stroke* (as was originally outlined) we focussed on evaluating the efficiency and accuracy of the Monte Carlo methods instead. The measures included in the experiment outline are a framework to observe the performance increases for improved methods such as Quadtree and GAFRW, and were intended to contrast these to the original method. We present the original workflow outlined in the proposal below, along with the adjusted workflow across the timespan normalised for the same period.

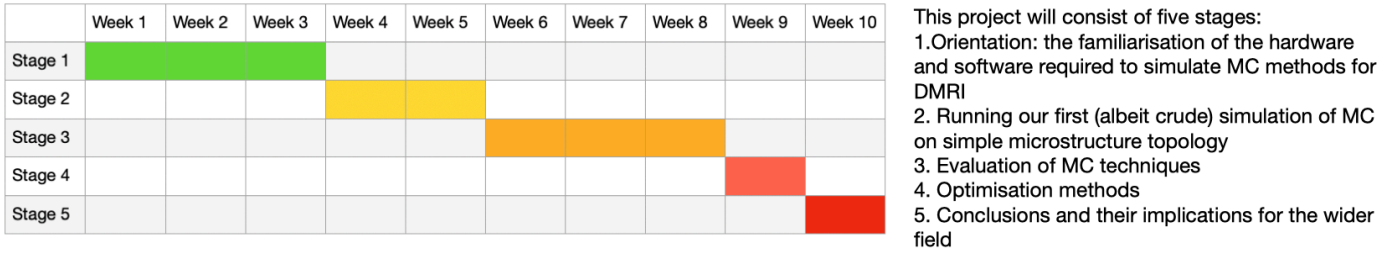


Fig 22. The original project trajectory. Steps 3-4 ended up becoming the bulk of the study, with parameter analysis as the focus.

Trialling ‘candidates’ for the Monte Carlo methods was a naive approach that promised more than what could be reasonably delivered in the timeframe. Not only was studying the wider subject area of dMRI and Monte Carlo methods an unexplored area of knowledge for the study, but practise with MATLAB and becoming familiar with the differences to *Pythonic* programming was another factor in slowing progress.

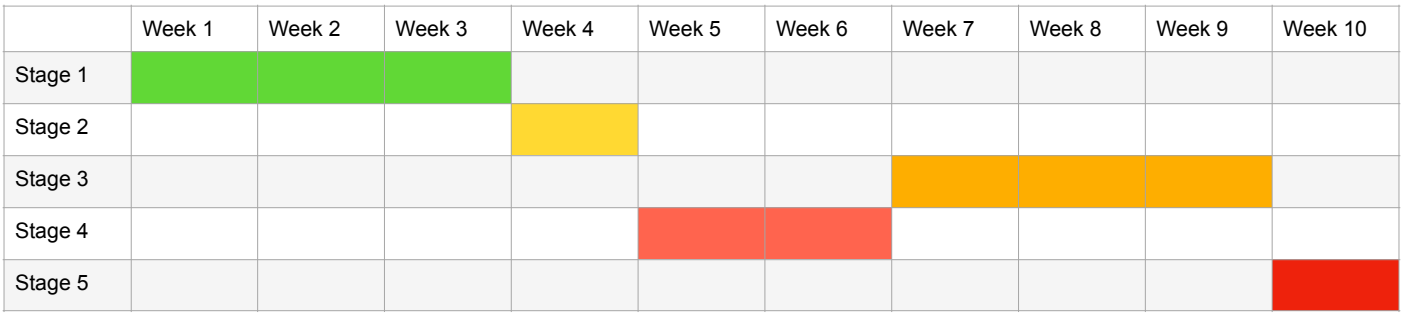


Fig 23. The *actual* trajectory of the project, normalised to the completion period of 10 weeks. Intermittent gaps were present between weeks due to the IOS and external circumstances, however the actual workload differed due to more focus on optimisation and evaluation of improved methods.

Optimisation methods were in fact, the improved methods mentioned such as GAFRW and Quadtree. They were implemented first before the evaluation stage of methods, because we had to consider the quantities to emphasise which highlighted the improvements such as the ranges for evaluating scan parameters.

6.1.1. Compromises and justifications

Due to time constraints, we were not able to implement the improved methods in the same manner as the original one was. We managed to generate some results for the network-based Quadtree random walk algorithm, however this was limited to a degree. However, we detail the instructions on implementing it in the code and within the improved methods mentioned in this document. Workload was split up due to the IOS across several months of intermittent progress, but a similar amount of time was utilised within the new setting. Due to the unconventional timeframe of the project, we were unable to gain access for the CUBRIC computing cluster towards the experimental stage however we did make extensive use of the MATLAB Online cloud computing unit. Thus the range of N_{ii} was affected by the maximum storage space on MATLAB Drive (5GB) but our results were sufficient for the purposes of this study. The main bottleneck in timing was the conversion of well-known functions to MATLAB and inaccessibility of cluster computing. As outlined in 5.3.3, parameters could be rapidly trialled and scaled if need be as a ‘brute force’ approach towards evaluation. We outlined the instructions and provided references towards implementing this in our code, however.

6.2. *Scientific merit*

6.2.1. *As foundation for further study*

Whilst unable to fully implement the improved methods in time, we were able to create a robust foundation for doing so (as well as successive improvements). This study provides the framework for three important concepts, previously evaluated separately in literature: permeability (two-compartment structures), circle packing (periodic microstructure substrate) and the ability to implement lesions for modelling neurological disorders in real time. The simplicity of the mask-based approach makes it straightforward to add and remove components, and utilising MRAE and Mean SD is a versatile form of evaluation with the latter applicable to cases without ground-truth data. We suggest the following workflow for utilising *diffSim* for future studies:

6.2.2. *Real world application: the inverse problem*

The modelling process of the dMRI phenomena we have discussed so far is known as a forward-problem, however its applications in real-world research would involve an inverse-problem variation. Known parameters can be adjusted that manipulate the generation of the substrate and the simulation within it, to infer unknown parameters that would otherwise be impossible to determine [111]. This study is merely one of the many potent applications of computational physics that is rapidly conquering a multitude of research areas which were once thought of as unreachable [112-114]. **Fig. 25** gives a visual overview of the inverse problem modelling cycle [115] whilst **Fig 26.** is an application of this modelling process to dMRI.

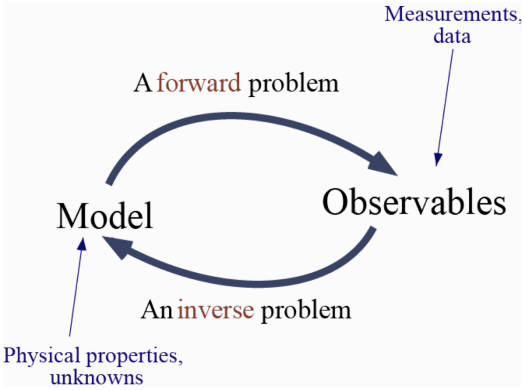


Fig 24. The modelling process of forward and inverse problems as a repeatable cycle. Source: *Australian National University.*

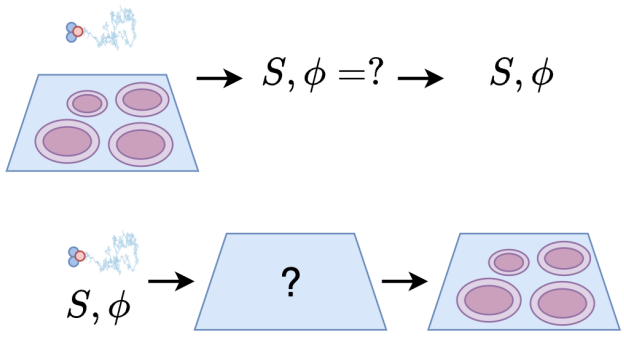


Fig 26. The forward problem (top) determines the dephasing per particle ϕ and resultant signal attenuation S by adjusting the substrate and random walk parameters until S converges to a known value (i.e. an analytical result). The inverse problem (bottom) is the opposite process: the signal values and random walks are adjusted to infer the microstructure, which in real experiments, is generally unknown.

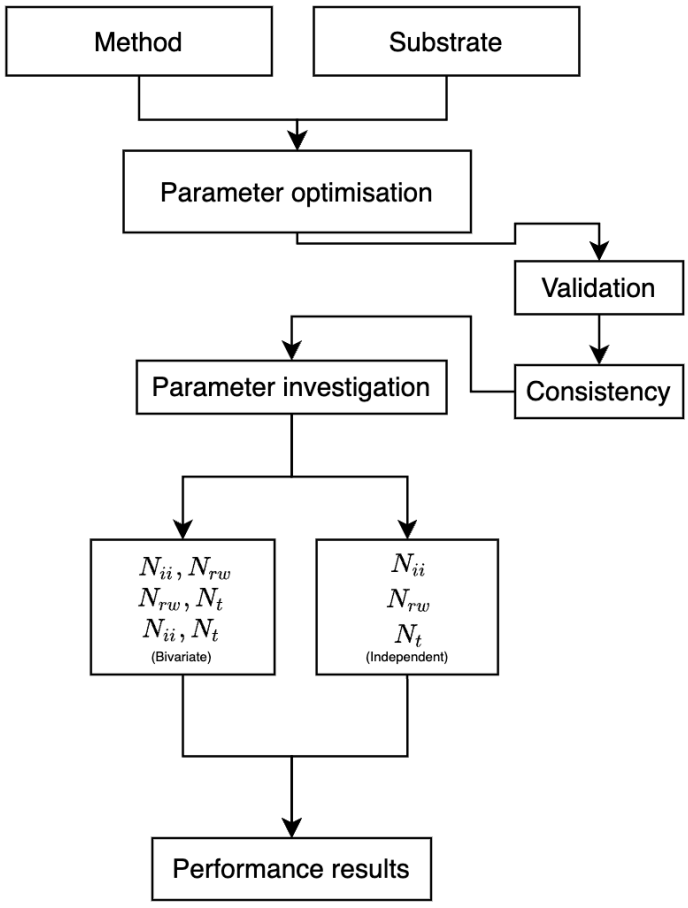


Fig 25. The process of evaluating a method, given the method and substrate inputs, to produce the performance results for MRAE, Mean SD and t_c.

6.3. Reflective statement

6.3.1. Thinking dimensionally and other skills acquired

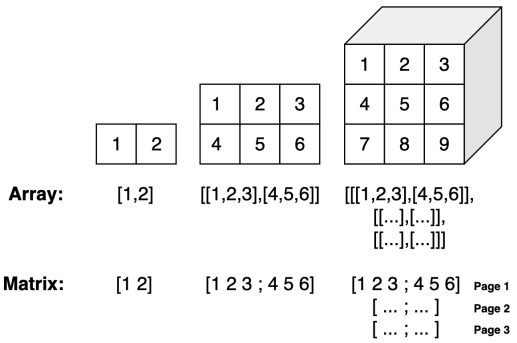


Fig 27. The same data for both array and matrix data structures. The limitations of the matrix lie where dimensions > 2 become difficult to access; a workaround (albeit impractical) is to group the 3D matrix into a set of 2D arrays, referenced by matrix name. This requires a range of lookup functions in order to reference the correct element; it can be performed far easier with a conventional array.

Prior to this project, the MATLAB suite was unfamiliar to me. Thinking dimensionally (in terms of matrices) was not an alien concept, having dealt with arrays and different data structures throughout my degree. However, explicitly considering each variable as a matrix of which the dimensions must conform for arithmetical operations and iterations was an interesting thought exercise. Looking back, too much time was spent on dimensionalising formulae and data structures within the code. Having a background in Python, I was used to multi-dimensional arrays and considered matrices to be a 2D subset of these. Thus, it was difficult to 'rack my head around' the limitations of 2D data structures where iterations through arrays required *always* specifying the rows and columns. N-D tensors in MATLAB were represented as 'pages' for any degrees higher than two, but are somewhat impractical to work with. Nevertheless, MATLAB provided a means of accessing data (when provided in an efficient 2D matrix) and performing operations at a much faster speed than conventional arrays, with mathematical procedures such as finding the inverse and element-wise operations being widely integrated. The concept of indexing was relatively new to me, but was proven to be a valuable alternative to for-looping for multidimensional data structures. Boolean and integer masks, along with rasterisation, introduced me to basic 2D imaging techniques which were consolidated by my pre-university background in Computer Science. Algorithmic complexity was a quantity I was able to apply in depth for the first time, and provided a useful means of evaluating algorithms based on the element-operation scaling influences on time.

Both top-down and bottom-up approaches to breaking down a concept like dMRI simulation helped reinforce my ability to communicate abstract concepts from a variety of angles. The top-down approach was used to describe the dMRI phenomena scientifically, working through the explanation of the signal attenuation and how it ultimately is derived from the movement of excited spins within tissue. Approaching the same problem from a bottom-up perspective, we started modelling every individual particle as a brownian random walk, and then discretising formulae that impose restrictions on its movement and specify the signal attenuation contributions. This gave me the opportunity to display through knowledge about a scientific process from both sides: the empirical top-down approach (where we change parameters to observe the effects on dependent variables, gathering data readings) and the computational bottom-up approach (re-creating the phenomena in a discrete setting to solve the problem by generating data readings).

6.3.2. Advice for starting over

If I had to start this project again, I would clarify on etymology of Monte Carlo methods before continuing further research. Monte Carlo methods operate essentially the *same*, and the plural use of methods simply refers to different ways of implementing the stochastic sampling process for different features. For example, the Quadtree structure was a different *method* to that of the GAFRW one. It utilised Monte Carlo random walks to traverse a *network* of shortcut coordinates, whereas the GAFRW was a random walk technique that *skipped* unnecessary steps. Using the MATLAB Drive from the start would have helped with transferring files between workstations, and albeit, make modelling between operating systems much easier (i.e. using the Linux-based CUBRIC cluster). Pre-generating results for each parameter combination and creating separate functions to analysis them would help with splitting the workload over different time periods, as simulation settings and results can be saved for further analysis. The original simulation settings can then be deleted to save space (as would be the case for experiments with high N_{ii} substrates).

There are two final tips I would give towards anyone who wishes to make performance improvements, that cannot be stressed enough: utilisation of parallelisation for N_{rw}, and indexing for virtually any other quantity. Naturally, time-steps require a for-loop or while-loop since the modelling is an iterative process, but for all other loops and array accessing routines, I would thoroughly recommend using as much linear indexing as possible. The performance gains are tremendous and would allow greater access to a higher order of parameters, allowing for a more thorough and in-depth evaluation [116].

7. Bibliography

[1] S. D. Shorvon, "A history of neuroimaging in epilepsy 1909–2009," *Epilepsia,* vol. 50, pp. 39-49, 2009.

[2] G. Schmidt, D. Dinter, M. F. Reiser, and S. O. Schoenberg, "The uses and limitations of whole-body magnetic resonance imaging," (in eng), *Dtsch Arztebl Int,* vol. 107, no. 22, pp. 383-389, 2010, doi: 10.3238/arztebl.2010.0383.

[3] O. Guy-Evans, "The History of Neuroimaging Techniques," *Simply Psychology,* 2021. [Online]. Available: https://www.simplypsychology.org/neuroimaging.html. (accessed 8 Dec, 2021).

[4] H. Mitchell, T. Hamilton, F. Steggerda, and H. Bean, "The chemical composition of the adult human body and its bearing on the biochemistry of growth," *Journal of Biological Chemistry,* vol. 158, no. 3, pp. 625-637, 1945.

[5] F.-Y. Chiu *et al.*, "Effect of intravenous gadolinium-DTPA on diffusion-weighted magnetic resonance images for evaluation of focal hepatic lesions," *Journal of computer assisted tomography,* vol. 29, no. 2, pp. 176-180, 2005.

[6] P. J. Basser, J. Mattiello, and D. LeBihan, "MR diffusion tensor spectroscopy and imaging," (in eng), *Biophys J,* vol. 66, no. 1, pp. 259-67, Jan 1994, doi: 10.1016/s0006-3495(94)80775-1.

[7] E. Scaccianoce *et al.*, "Combined DTI-fMRI Analysis for a Quantitative Assessment of Connections Between WM Bundles and Their Peripheral Cortical Fields in Verbal Fluency," (in eng), *Brain Topogr,* vol. 29, no. 6, pp. 814-823, Nov 2016, doi: 10.1007/s10548-016-0516-0.

[8] L. Vavassori, S. Sarubbo, and L. Petit, "Hodology of the superior longitudinal system of the human brain: a historical perspective, the current controversies, and a proposal," (in eng), *Brain Struct Funct,* vol. 226, no. 5, pp. 1363-1384, Jun 2021, doi: 10.1007/s00429-021-02265-0.

[9] D. Kuhnt *et al.*, "Fiber tractography based on diffusion tensor imaging compared with high-angular-resolution diffusion imaging with compressed sensing: initial experience," (in eng), *Neurosurgery,* vol. 72 Suppl 1, no. 0 1, pp. 165-75, Jan 2013, doi: 10.1227/NEU.0b013e318270d9fb.

[10] S. P. Choi *et al.*, "The density ratio of grey to white matter on computed tomography as an early predictor of vegetative state or death after cardiac arrest," (in eng), *Emerg Med J,* vol. 25, no. 10, pp. 666-9, Oct 2008, doi: 10.1136/emj.2007.053306.

[11] R. H. Masland, "Neuronal cell types," *Current Biology,* vol. 14, no. 13, pp. R497-R500, 2004.

[12] E. Luders, P. M. Thompson, and A. W. Toga, "The development of the corpus callosum in the healthy human brain," *Journal of Neuroscience,* vol. 30, no. 33, pp. 10985-10990, 2010.

[13] A. A. Mercadante and P. Tadi, "Neuroanatomy, Gray Matter," 2020.

[14] I. M. McDonough and J. T. Siegel, "The Relation Between White Matter Microstructure and Network Complexity: Implications for Processing Efficiency," (in English), *Frontiers in Integrative Neuroscience,* Original Research vol. 12, 2018-September-24 2018, doi: 10.3389/fnint.2018.00043.

[15] D. Suminaite, D. A. Lyons, and M. R. Livesey, "Myelinated axon physiology and regulation of neural circuit function," (in eng), *Glia,* vol. 67, no. 11, pp. 2050-2062, 2019, doi: 10.1002/glia.23665.

[16] M. Amiry-Moghaddam and O. P. Ottersen, "The molecular basis of water transport in the brain," *Nature Reviews Neuroscience,* vol. 4, no. 12, pp. 991-1001, 2003/12/01 2003, doi: 10.1038/nrn1252.

[17] D. A. Rubenstein, W. Yin, and M. D. Frame, "Chapter 11 - Intraocular fluid flow," in *Biofluid Mechanics (Third Edition)*, D. A. Rubenstein, W. Yin, and M. D. Frame Eds.: Academic Press, 2022, pp. 423-442.

[18] C. Beaulieu, "The basis of anisotropic water diffusion in the nervous system–a technical review," NMR in Biomedicine: An International Journal Devoted to the Development and Application of Magnetic Resonance In Vivo, vol. 15, no. 7-8, pp. 435-455, 2002.

[19] R. E. Roberts, E. J. Anderson, and M. Husain, "White Matter Microstructure and Cognitive Function," *The Neuroscientist,* vol. 19, no. 1, pp. 8-15, 2013, doi: 10.1177/1073858411421218.

[20] R. G. González, "Clinical MRI of acute ischemic stroke," (in eng), *J Magn Reson Imaging,* vol. 36, no. 2, pp. 259-271, 2012, doi: 10.1002/jmri.23595.

[21] S. K. Daniels and A. L. Foundas, "Lesion localization in acute stroke," *Journal of Neuroimaging,* vol. 9, no. 2, pp. 91-98, 1999.

[22] P. Maggi *et al.*, "The formation of inflammatory demyelinated lesions in cerebral white matter," (in eng), *Ann Neurol,* vol. 76, no. 4, pp. 594-608, 2014, doi: 10.1002/ana.24242.

[23] I. S. Buyanova and M. Arsalidou, "Cerebral White Matter Myelination and Relations to Age, Gender, and Cognition: A Selective Review," (in English), *Frontiers in Human Neuroscience,* Review vol. 15, 2021-July-06 2021, doi: 10.3389/fnhum.2021.662031.

[24] C. Sampaio-Baptista *et al.*, "Motor skill learning induces changes in white matter microstructure and myelination," *Journal of Neuroscience,* vol. 33, no. 50, pp. 19499-19503, 2013.
[25] J. S. Rosenthal, "Parallel computing and Monte Carlo algorithms," *Far east journal of theoretical statistics,* vol. 4, no. 2, pp. 207-236, 2000.
[26] R. Driscoll and P. Bender, "Proton gyromagnetic ratio," *Physical Review Letters,* vol. 1, no. 11, p. 413, 1958.
[27] D. I. Hoult and B. Bhakar, "NMR signal reception: Virtual photons and coherent spontaneous emission," *Concepts in Magnetic Resonance: An Educational Journal,* vol. 9, no. 5, pp. 277-297, 1997.
[28] L. Axel, "Revised glossary of MR terms," *Radiology,* vol. 162, no. 3, pp. 874-874, 1987.
[29] M. L. Lipton, Totally accessible MRI: a user's guide to principles, technology, and applications. Springer Science & Business Media, 2010.
[30] R. Turner, "Gradient coil design: a review of methods," *Magnetic resonance imaging,* vol. 11, no. 7, pp. 903-920, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/0730725X9390209V?via%3Dihub. (accessed 2 April, 2022).
[31] S. S. Hidalgo-Tobon, "Theory of gradient coil design methods for magnetic resonance imaging," *Concepts in Magnetic Resonance Part A,* vol. 36, no. 4, pp. 223-242, 2010.
[32] B. Aldefeld and P. Börnert, "Effects of gradient anisotropy in MRI," *Magnetic resonance in medicine,* vol. 39, no. 4, pp. 606-614, 1998. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.1910390414?sid=nlm%3Apubmed. (accessed 8 March, 2022).
[33] P. Glover, "Interaction of MRI field gradients with the human body," *Physics in Medicine & Biology,* vol. 54, no. 21, p. R99, 2009.
[34] A. Walker, G. Liney, P. Metcalfe, and L. Holloway, "MRI distortion: considerations for MRI based radiotherapy treatment planning," *Australasian physical & engineering sciences in medicine,* vol. 37, no. 1, pp. 103-113, 2014.
[35] E. F. Jackson, L. E. Ginsberg, D. F. Schomer, and N. E. Leeds, "A review of MRI pulse sequences and techniques in neuroimaging," *Surgical Neurology,* vol. 47, no. 2, pp. 185-199, 1997.
[36] G. Katti, S. A. Ara, and A. Shireen, "Magnetic resonance imaging (MRI)–A review," *International journal of dental clinics,* vol. 3, no. 1, pp. 65-70, 2011.
[37] B. Rosen and L. Wald, "MR Image Encoding."
[38] J. Philibert, "One and a half century of diffusion: Fick, Einstein before and beyond," 2006.
[39] S. Guenneau and T. M. Puvirajesinghe, "Fick's second law transformed: one path to cloaking in mass diffusion," *Journal of The Royal Society Interface,* vol. 10, no. 83, p. 20130106, 2013, doi: doi:10.1098/rsif.2013.0106.
[40] E. O. Stejskal and J. E. Tanner, "Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient," *The journal of chemical physics,* vol. 42, no. 1, pp. 288-292, 1965.
[41] J. E. Tanner, Use of a pulsed magnetic-field gradient for measurements of self-diffusion by spin-echo nuclear magnetic resonance with applications to restricted diffusion in several tissues and emulsions. The University of Wisconsin-Madison, 1966.
[42] M. Dobrzynski, R. Gauges, U. Kummer, J. Pahle, and P. Willy, "When do diffusion-limited trajectories become memoryless?," in *Proceedings of the 5th Workshop on Computation of Biochemical Pathways and Genetic Networks*, 2008.
[43] J. J. Valsamis, P. I. Dubovan, and C. A. Baron, "Characterization and correction of time-varying eddy currents for diffusion MRI," *Magnetic Resonance in Medicine,* vol. 87, no. 5, pp. 2209-2223, 2022.
[44] A. Rokem *et al.*, "Evaluating the accuracy of diffusion MRI models in white matter," *PLoS One,* vol. 10, no. 4, p. e0123272, 2015.
[45] M. Niknejad, Bell, D., "Apparent diffusion coefficient," *Radiology Reference Article,* 27 Oct 2021, doi: 10.53347/rID-21759.
[46] M. J. Tait, S. Saadoun, B. A. Bell, and M. C. Papadopoulos, "Water movements in the brain: role of aquaporins," *Trends in neurosciences,* vol. 31, no. 1, pp. 37-43, 2008.
[47] K. Yue, J. Jiang, P. Zhang, and L. Kai, "Functional Analysis of Aquaporin Water Permeability Using an Escherichia coli-Based Cell-Free Protein Synthesis System," (in English), *Frontiers in Bioengineering and Biotechnology,* Methods vol. 8, 2020-August-19 2020, doi: 10.3389/fbioe.2020.01000.
[48] A. S. Filippidis, R. B. Carozza, and H. L. Rekate, "Aquaporins in Brain Edema and Neuropathological Conditions," (in eng), *Int J Mol Sci,* vol. 18, no. 1, p. 55, 2016, doi: 10.3390/ijms18010055.

[49] H. D. Landahl, "A note on the units of membrane permeability to water," *The bulletin of mathematical biophysics,* vol. 10, no. 3, pp. 187-190, 1948/09/01 1948, doi: 10.1007/BF02477492.
[50] W. S. Price, "Pulsed-field gradient nuclear magnetic resonance as a tool for studying translational diffusion: Part 1. Basic theory," *Concepts in Magnetic Resonance: An Educational Journal,* vol. 9, no. 5, pp. 299-336, 1997.
[51] D. Le Bihan and E. Breton, "Imagerie de diffusion in-vivo par résonance magnétique nucléaire," *Comptes-Rendus de l'Académie des Sciences,* vol. 93, no. 5, pp. 27-34, 1985.
[52] R. N. Sener, "Diffusion MRI: apparent diffusion coefficient (ADC) values in the normal brain and a classification of brain disorders based on ADC values," (in eng), *Comput Med Imaging Graph,* vol. 25, no. 4, pp. 299-326, Jul-Aug 2001, doi: 10.1016/s0895-6111(00)00083-5.
[53] T. A. G. M. Huisman, "Diffusion-weighted and diffusion tensor imaging of the brain, made easy," (in eng), *Cancer Imaging,* vol. 10 Spec no A, no. 1A, pp. S163-S171, 2010, doi: 10.1102/1470-7330.2010.9023.
[54] O. Cakir *et al.*, "Comparison of the diagnostic performances of diffusion parameters in diffusion weighted imaging and diffusion tensor imaging of breast lesions," (in eng), *Eur J Radiol,* vol. 82, no. 12, pp. e801-6, Dec 2013, doi: 10.1016/j.ejrad.2013.09.001.
[55] D. C. Alexander, "Workshop on Computational DMRI." [Online]. Available: http://picsl.upenn.edu/cdmri08. (accessed 19 Jan, 2022).
[56] D. Rohmer and G. T. Gullberg, "A bloch-torrey equation for diffusion in a deforming media," 2006.
[57] M. G. Hall, "Continuity, the Bloch-Torrey equation, and Diffusion MRI," *arXiv preprint arXiv:1608.02859,* 2016.
[58] V.-D. Nguyen, "A finite elements method to solve the Bloch-Torrey equation applied to diffusion magnetic resonance imaging of biological tissues," 2014.
[59] S. Hwang, C.-L. Chin, F. Wehrli, and D. Hackney, "An image-based Finite Difference Model for Simulating Restricted Diffusion," Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine, vol. 50, pp. 373-82, 08/01 2003, doi: 10.1002/mrm.10536.
[60] J. A. McNab, "The Bloch-Torrey Equation & Diffusion Imaging (DWI, DTI, q-Space Imaging)."
[61] H. C. Torrey, "Bloch equations with diffusion terms," *Physical review,* vol. 104, no. 3, p. 563, 1956.
[62] D. S. Grebenkov, "Pulsed-gradient spin-echo monitoring of restricted diffusion in multilayered structures," *J Magn Reson,* vol. 205, no. 2, pp. 181-95, Aug 2010, doi: 10.1016/j.jmr.2010.04.017.
[63] D. S. Grebenkov, "NMR survey of reflected Brownian motion," *Review of Modern Physics,* 2007.
[64] D. S. Grebenkov and B.-T. Nguyen, "Geometrical structure of Laplacian eigenfunctions," *siam REVIEW,* vol. 55, no. 4, pp. 601-667, 2013.
[65] J. Kärger, "Zur Bestimmung der Diffusion in einem Zweibereichsystem mit Hilfe von gepulsten Feldgradienten," *Annalen der Physik,* vol. 479, no. 1-2, pp. 1-4, 1969.
[66] H. T. Nguyen, "Numerical investigations of some mathematical models of the diffusion MRI signal," Université Paris Sud-Paris XI, 2014.
[67] V.-D. Nguyen *et al.*, "Portable simulation framework for diffusion MRI," *Journal of Magnetic Resonance,* vol. 309, p. 106611, 2019.
[68] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev, "Why the Monte Carlo method is so important today," *Wiley Interdisciplinary Reviews: Computational Statistics,* vol. 6, no. 6, pp. 386-392, 2014.
[69] H. Kahn, "Applications of monte carlo," RAND Corp., Santa Monica, Calif., 1954.
[70] N. Chopin, "Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference," *The Annals of Statistics,* vol. 32, no. 6, pp. 2385-2411, 2004.
[71] F. Y. Kuo and I. H. Sloan, "Lifting the curse of dimensionality," *Notices of the AMS,* vol. 52, no. 11, pp. 1320-1328, 2005.
[72] D. S. Novikov, E. Fieremans, J. H. Jensen, and J. A. Helpern, "Random walks with barriers," *Nature physics,* vol. 7, no. 6, pp. 508-514, 2011.
[73] M. G. Hall and D. C. Alexander, "Convergence and parameter choice for Monte-Carlo simulations of diffusion MRI," *IEEE Trans Med Imaging,* vol. 28, no. 9, pp. 1354-64, Sep 2009, doi: 10.1109/TMI.2009.2015756.
[74] J. Rafael-Patino, D. Romascano, A. Ramirez-Manzanares, E. J. Canales-Rodríguez, G. Girard, and J.-P. Thiran, "Robust Monte-Carlo Simulations in Diffusion-MRI: Effect of the Substrate Complexity and Parameter Choice on the Reproducibility of Results," *Frontiers in Neuroinformatics,* 10.3389/fninf.2020.00008 vol. 14, p. 8, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fninf.2020.00008

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7076166/pdf/fninf-14-00008.pdf. (accessed 15 April, 2022).
[75] H.-G. Lipinski, "Monte Carlo simulation of extracellular diffusion in brain tissues," *Physics in Medicine & Biology,* vol. 35, no. 3, p. 441, 1990.
[76] P. A. Cook, Y. Bai, M. G. Hall, S. Nedjati-Gilani, K. K. Seunarine, and D. C. Alexander, "Camino: Diffusion MRI reconstruction and processing," 2005.
[77] D. Sousa and H. Ferreira, "MCSD: A MATLAB Tool for Monte-Carlo Simulations of Diffusion in biological Tissues," *Journal of Open Source Software,* vol. 3, no. 30, 26 October 2018, doi: 10.21105/joss.00966.
[78] H.-H. Lee, E. Fieremans, and D. S. Novikov, "Realistic Microstructure Simulator (RMS): Monte Carlo simulations of diffusion in three-dimensional cell segmentations of microscopy images," *Journal of Neuroscience Methods,* vol. 350, p. 109018, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0165027020304416?via%3Dihub
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8026575/pdf/nihms-1651786.pdf. (accessed 5 April, 2022).
[79] C.-H. Yeh, B. Schmitt, D. Le Bihan, J.-R. Li-Schlittgen, C.-P. Lin, and C. Poupon, "Diffusion microscopist simulator: a general Monte Carlo simulation system for diffusion magnetic resonance imaging," *PLoS One,* vol. 8, no. 10, p. e76626, 2013. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3794953/pdf/pone.0076626.pdf. (accessed 5 April, 2022).
[80] Y. Tachibana, T. Duval, and T. Obata, "Monte Carlo Simulator for Diffusion-weighted Imaging Sequences," *Magn Reson Med Sci,* vol. 20, no. 2, pp. 222-226, Jun 1 2021, doi: 10.2463/mrms.bc.2020-0013.
[81] M. Nilsson, J. Lätt, F. Ståhlberg, D. van Westen, and H. Hagslätt, "The importance of axonal undulation in diffusion MR measurements: a Monte Carlo simulation study," *NMR in Biomedicine,* vol. 25, no. 5, pp. 795-805, 2012.
[82] G. Rensonnet *et al.*, "Towards microstructure fingerprinting: estimation of tissue properties from a dictionary of Monte Carlo diffusion MRI simulations," *NeuroImage,* vol. 184, pp. 964-980, 2019.
[83] A. Szafer, J. Zhong, and J. C. Gore, "Theoretical model for water diffusion in tissues," *Magnetic resonance in medicine,* vol. 33, no. 5, pp. 697-712, 1995.
[84] S. Kakutani, "RANDOM WALK AND THE TYPE PROBLEM OP RIEMANN SURFACES," *Contributions to the Theory of Riemann Surfaces.(AM-30), Volume 30,* vol. 30, p. 95, 1953.
[85] A. Kharab and R. Guenther, An introduction to numerical methods: a MATLAB® approach. CRC press, 2018.
[86] L. Beltrachini, "Diffusion Simulator," GitHub, 2021.
[87] K. V. Nguyen, E. Hernandez-Garzon, and J. Valette, "Efficient GPU-based Monte-Carlo simulation of diffusion in real astrocytes reconstructed from confocal microscopy," *J Magn Reson,* vol. 296, pp. 188-199, Nov 2018, doi: 10.1016/j.jmr.2018.09.013.
[88] S. Eddins and L. Shure, "Matrix indexing in Matlab," Mathworks Newsletter, https://www. mathworks. com/company/newsletters/articles/matrix-indexing-in-matlab. html, 2001 (accessed 17 April, 2022).
[89] J. Kärger, "NMR self-diffusion studies in heterogeneous systems," *Advances in Colloid and Interface Science,* vol. 23, pp. 129-148, 1985.
[90] Dk, S. S., Y. Assaf, C. J. Evans, and Jones, "Improved precision in CHARMED assessment of white matter through sampling scheme optimization and model parsimony testing," *Magnetic resonance in medicine,* vol. 71, no. 2, 2014 Feb 2014, doi: 10.1002/mrm.24717.
[91] A. Danz, "Bubblebath.m," MATLAB Central File Exchange, 2020 (accessed 9 October, 2021).
[92] B. Shoelson, "createCirclesMask.m," MATLAB Central File Exchange, 2014. (accessed 17 April, 2022).
[93] K. L. West, N. D. Kelm, R. P. Carson, and M. D. Does, "A revised model for estimating g-ratio from MRI," (in eng), *NeuroImage,* vol. 125, pp. 1155-1158, 2016, doi: 10.1016/j.neuroimage.2015.08.017.
[94] F. Sanders and D. Whitteridge, "Conduction velocity and myelin thickness in regenerating nerve fibres," *The Journal of physiology,* vol. 105, no. 2, p. 152, 1946.
[95] R. S. Smith and Z. J. Koles, "Myelinated nerve fibers: computed effect of myelin thickness on conduction velocity," *American Journal of Physiology-Legacy Content,* vol. 219, no. 5, pp. 1256-1258, 1970.
[96] E. Rowell. "Big-O Cheat Sheet." https://www.bigocheatsheet.com (accessed 3 March, 2022).
[97] F. Kraaikamp and H. Meester, "A modern introduction to probability and statistics," ed: Springer London, 2005.
[98] D. S. Grebenkov, "ML (multi-layered) dMRI framework for MATLAB," 2012.

[99] F. Aboitiz, A. B. Scheibel, R. S. Fisher, and E. Zaidel, "Fiber composition of the human corpus callosum," *Brain Research,* vol. 598, no. 1, pp. 143-153, 1992/12/11/ 1992, doi: https://doi.org/10.1016/0006-8993(92)90178-C.
[100] S. Pfalzner and P. Gibbon, "Many-Body Tree Methods in Physics," *Optics & Photonics News,* vol. 9, no. 10, p. 47, 1998.
[101] M. Hoyle. "Random Walks in MATLAB." https://www.mathworks.com/matlabcentral/fileexchange/22003-random-walks-in-matlab (accessed 28 Feb, 2022).
[102] D. S. Grebenkov, "A fast random walk algorithm for computing the pulsed-gradient spin-echo signal in multiscale porous media," *J Magn Reson,* vol. 208, no. 2, pp. 243-55, Feb 2011, doi: 10.1016/j.jmr.2010.11.009.
[103] D. Le Bihan et al., "Diffusion tensor imaging: concepts and applications," Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine, vol. 13, no. 4, pp. 534-546, 2001.
[104] E. J. Benjamin *et al.*, "Heart disease and stroke statistics—2017 update: a report from the American Heart Association," *circulation,* vol. 135, no. 10, pp. e146-e603, 2017.
[105] J. Kucharczyk, J. Mintorovitch, H. S. Asgari, and M. Moseley, "Diffusion/perfusion MR imaging of acute cerebral ischemia," *Magnetic resonance in medicine,* vol. 19, no. 2, pp. 311-315, 1991.
[106] M. Herlihy and N. Shavit, "Mutual Exclusion," *The Art of Multiprocessor Programming,* p. 24, 2012.
[107] I. Foster, Designing and building parallel programs: concepts and tools for parallel software engineering. Addison-Wesley, 2020.
[108] C.-m. Ma, "Implementation of a Monte Carlo code on a parallel computer system," *Parallel Computing,* vol. 20, no. 7, pp. 991-1005, 1994.
[109] CUBRIC, "Cardiff University Brain Research Imaging Centre (CUBRIC)," 2021. [Online]. Available: https://www.cardiff.ac.uk/research-equipment/facilities/view/cubric. (accessed 2 Nov, 2021).
[110] D. P. Rodgers, "Improvements in multiprocessor system design," *ACM SIGARCH Computer Architecture News,* vol. 13, no. 3, pp. 225-231, 1985.
[111] W. L. Rodi and R. L. Mackie, "The inverse problem," *The Magnetotelluric Method: Theory and Practice,* pp. 347-414, 2012.
[112] A. K. Liu, J. W. Belliveau, and A. M. Dale, "Spatiotemporal imaging of human brain activity using functional MRI constrained magnetoencephalography data: Monte Carlo simulations," *Proceedings of the National Academy of Sciences,* vol. 95, no. 15, pp. 8945-8950, 1998.
[113] D. M. Schmidt, J. S. George, and C. C. Wood, "Bayesian inference applied to the electromagnetic inverse problem," *Human brain mapping,* vol. 7, no. 3, pp. 195-212, 1999.
[114] P. B. Weichman, D. R. Lun, M. H. Ritzwoller, and E. M. Lavely, "Study of surface nuclear magnetic resonance inverse problems," *Journal of Applied Geophysics,* vol. 50, no. 1-2, pp. 129-147, 2002.
[115] M. Sambridge. "Inverse Problems and Seismic tomography." http://rses.anu.edu.au/~hrvoje/PHYS3070/Malcolm_Sambridge_lecture.pdf (accessed 3 April, 2022).
[116] M. Strauss. "Speeding up Python Code: Fast Filtering and Slow Loops." https://towardsdatascience.com/speeding-up-python-code-fast-filtering-and-slow-loops-8e11a09a9c2f (accessed 24 April, 2022).
[117] M. Mao, "Monte Carlo Simulation of Diffusion Magnetic Resonance Imaging," *University of Waterloo,* 2019.

8. Appendices

8.1. Special plots

8.1.1. Trial runs for the two-compartment structure

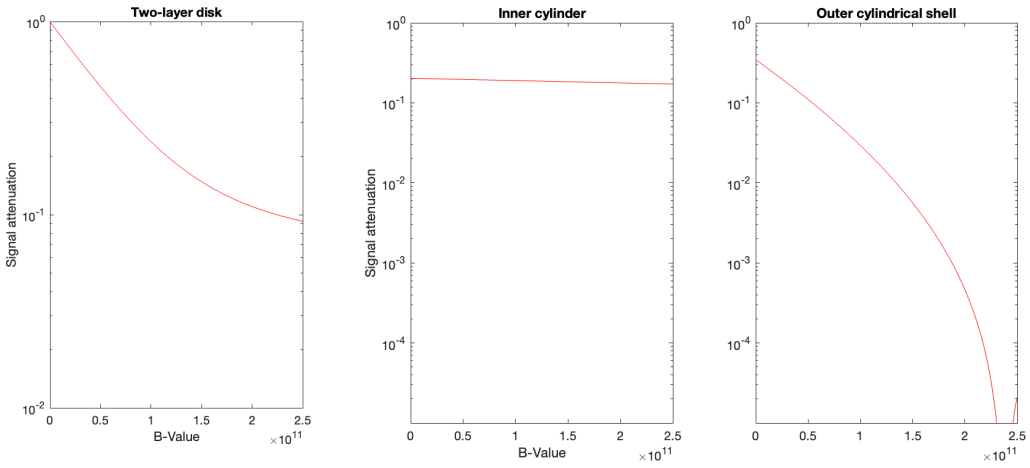


Fig 28. The first run (left) considers the two-compartment structure ensemble, whilst the second (right) evaluates *S* for each compartment separately.

8.1.2. Extremes cases for values of N_t in Simulation 2

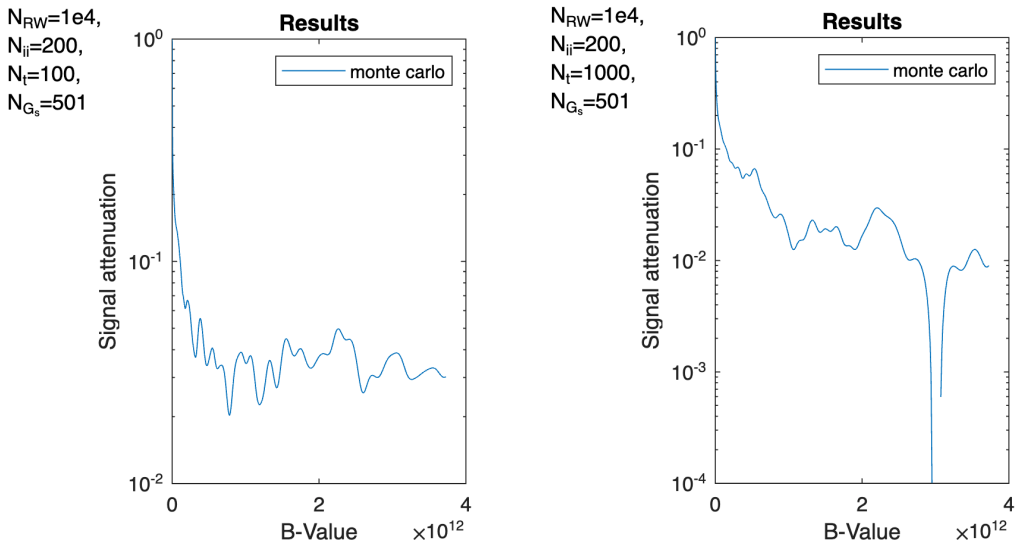


Fig 29. In extreme cases of lower N_t, the results suffer Gaussian noise (left). However, if N_t is high enough, over-stepping can cause a new problem with divisions by zero in some parts of the signal (right). This is particularly true for results with $U < 10^8$.

8.2. MATLAB

8.2.1. maskPos.m

```
%% Compute voxel in I where each particle in X belongs to
function ind_s=maskPos(X,N_i,r)
    X=ceil(prodsum(X,1./r));
    X(X<1)=1;
    X(X(:,1)>N_i(1),1)=N_i(1);
    X(X(:,2)>N_i(2),2)=N_i(2);
    if length(N_i)==2
        ind_s=sub2ind(N_i,X(:,1),X(:,2));
    elseif length(N_i)==3
        X(X(:,3)>N_i(3),3)=N_i(3);
        ind_s=sub2ind(N_i,X(:,1),X(:,2),X(:,3));
    end
end
```

8.2.2. randCirc.m

```
%% Random circular direction
function s=randCirc(N,dim)
    if dim==2
        U=rand(N,1);
        s=[cos(2*pi*U),sin(2*pi*U)];
    elseif dim==3
        fi=2*pi*rand(N,1);
        ti=acos(1-2*rand(N,1));
        s=[sin(ti).*cos(fi),sin(ti).*sin(fi),cos(ti)];
    end
end
```

8.2.3. prodsum.m

```
%% Dot product
function A=prodsum(A,B)
    dim=size(A,2);
    if size(B,2)==dim
        for kk=1:dim,A(:,kk)=A(:,kk).*B(:,kk);end
    else
        for kk=1:dim,A(:,kk)=A(:,kk).*B;end
    end
end
```

8.2.4. getNodePos.m

```
function nodePos = getNodePos(node,X,Q)
    % Get position of node in grid coordinates
    Y = X';
    nodePos = [X(Q.ind(node)),Y(Q.ind(node))];
end
```

8.2.5. randomGraphMove.m

```
function out = randomGraphMove(X,S)
    % Produce a random edge on a graph defined by S
    % X = current node, S = adjacency matrix S(i,j)
    V = S(:,X); % Node this one is connected with
    Idx = find(V);
    v = V(Idx)/sum(V);
    s = cumsum(v);
    r = rand;
    ind = find(s > r);
    out = Idx(ind); % adapted code from RandomGraphMove
end
```

8.2.6. *diffSim.m*

```
function S=diffSim(I,seq,simu,gam)
    t=seq.t; G=seq.G; D=simu.D; N=simu.N; r=simu.r;
    N_i = size(I);
    dim = numel(N_i);
    N_c = sum(unique(I(:))>0);
    dt  = t(2)-t(1);
    dx  = sqrt(2*dim*dt*D');
    if N_c>1, P = simu.P else P = NaN end
    %% setup points
    N       = round(N*numel(I)/sum(I(:)>0));
    X       = prodsum(rand(N,dim),r.*N_i);
    ind_s   = maskPos(X,N_i,r); % sub2ind
    % delete points in non-diffusion regions
    X(I(ind_s)==0,:)    = [];
    ind_s(I(ind_s)==0)  = [];
    % total particles remaining
    N_p     = size(X,1);
    phase   = zeros(N_p,1);
    %% random walk
    for tt=1:length(t)
        Xn = prodsum(randCirc(N_p,dim),dx(I(ind_s)+1));
        ind_s = maskPos(X,N_i,r); % prev
        ind_p = maskPos(mod(X+Xn,N_i(1)*r),N_i,r); % next
        Ie = I(ind_s)==I(ind_p); % particles that moved pixels
        % check re permeability
        if ~isnan(P(1))
            Pp = rand(N_p,1);
            for pp=1:length(Ie) % caution! O(n^2)
                if Ie(pp) == 0 % get particles that didn't move
                    a = I(ind_s(pp)); % prev
                    b = I(ind_p(pp)); % next
                    if a>0 && b>0 % exclude ECS
                        Pt = P(a,b)*sqrt((dt*pi)/D(a)); %2*P(a,b)*sqrt((dt*pi)/D(a));
                        if Pp(pp) <= Pt
                            Ie(pp) = 1;
                        end
                    end
                end
            end
        end
        % move points
        X(Ie,:) = X(Ie,:) + Xn(Ie,:);
        phase   = phase + sum(prodsum(X,G(tt,:)),2); % eqn
        % periodic BC
        M1=rem(X(Ie,:),N_i(1)*r);
        M2=mod(X(Ie,:),N_i(1)*r);
        ind_bc=(M2~=X(Ie,:)).*sign(M1);
        phase(Ie)=phase(Ie)+sum(prodsum(-ind_bc*N_i(1)*r,sum(G(1:tt,:))),2);
        X(Ie,:)=M2;
    end
    %% compute signal
    if ~isfield(seq,'G_s'),seq.G_s=1;end
    S=zeros(length(seq.G_s),1);
    for gg=1:length(S),S(gg)=mean(exp(1j*gam*dt*phase*seq.G_s(gg)));end

end
```

8.2.7. *createSubstrate.m*

```
function [I, disks] = createSubstrate(ra,N_ii,C,layers)

    pop = C.pop;
    sf = C.scale;
    spacing = C.spacing;
    alpha = C.alpha;
    beta = C.beta;

    % Gamma distribution
    gamma = gamrnd(alpha,beta,pop,1);
    radii = sort(gamma','descend')*sf;

    % Bubblebath parameters
    Bb.frameSize=[N_ii,N_ii];
    Bb.overlapType='relative';
    Bb.overlap=spacing-1;
    Bb.density=(pop/(N_ii));
    Bb.circSize=radii;
    Bb.nSizes=NaN;
    Bb.edgeType=3;
    Bb.maxCircsPerRad=1;
    Bb.supressWarning=true;
    [BbData, BbHandles, Bb] = bubblebath(Bb);

    % Output
    disks.centers = BbData(:,1:2);
    disks.radii = BbData(:,3);
    disks.count = length(disks.radii);
    difference = (disks.count-pop);
    if difference<0
        disks.surplus = 0;
        disks.missed = norm(difference);
    else
        disks.surplus = norm(difference);
        disks.missed = 0;
    end

    % Create mask from output
    I = uint8(createCirclesMask([N_ii N_ii],disks.centers+0.5*N_ii,disks.radii));
    if layers > 1
        % Create mask from output
        I = I + uint8(createCirclesMask([N_ii
N_ii],disks.centers+0.5*N_ii,disks.radii*0.6));
        % 0.6 = g ratio factor
    end
end
```

8.2.8. Sim1.m

```
% Two-layer structure

%% params
% physical
ra    = [2.5,5.0]*1e-6;
gam   = 2.675e8;
gMax  = 0.5;
delta = 50e-3;
Delta = 100e-3;
tf    = 2*Delta;
diffu = [2.0,2.0,2.0]*1e-9;
perma = [1.0,0]*1e-5;
relax = [Inf Inf];

% scan
N_ii  = 200;
N_rw  = 1e5;
N_t   = 1e3;
U     = N_rw * N_t;
t     = linspace(0,tf,N_t);

%% substrate
[X,Y] = meshgrid(linspace(-ra(2)*1.3,ra(2)*1.3,N_ii));
d = sqrt(X.^2+Y.^2);
I = uint8(d<=ra(2)) + uint8(d<ra(1));

%% sequence
gRes = 0.001; gSteps = gMax/gRes;
G=0*t; G(t<=delta)=1; G(t>=Delta&t<=Delta+delta)=-1;
dir=[1,0]; G=kron(dir,G');
seq.G = G; seq.t = t; seq.G_s = 0:gRes:gMax;
bVal = (gam^2 * seq.G_s.*seq.G_s * delta^2 * (Delta - delta/3));

%% analytical answer
if ~exist ('Sa','var') % debugging - save time
    ML.d = 2; ML.r = ra; ML.D = [diffu(1), diffu(2)]; ML.W = [perma(1),perma(2)]; ML.T
= relax;
    Sa = ML_compute(seq.G_s,delta,Delta,ML);
end

%% simulation
tic
simu.r = X(1,2)-X(1,1);
simu.N = N_rw;
simu.D = diffu;
simu.P = flip([ML.W; flip(ML.W)]);
S = diffSim(I,seq,simu,gam);
elapsedTime = toc;
save(sprintf('Results/Sim1/Nii=%d Nrw=1e%d Nt=1e%d',N_ii,log10(N_rw),log10(N_t)));
```

8.2.9. Sim2.m

```
% Multi-axonal substrate

%% params
% physical
ra    = 1e-5;
gam   = 2.675e8;
gMax  = 0.5;
delta = 50e-3;
Delta = 100e-3;
tf    = 2*Delta;
diffu = [2.0,2.0,2.0]*1e-9;
perma = [1.0,0]*1e-5;
relax = [Inf Inf];

% scan
N_ii  = 300;
N_rw  = 1e4;
N_t   = 1e4;
U     = N_rw * N_t;
t     = linspace(0,tf,N_t);

%% substrate
[X,Y] = meshgrid(linspace(-ra*1.3,ra*1.3,N_ii));
C.pop     = 100; % scale with N_ii
C.scale   = 2.1; % visually adjust this as required
C.spacing = 1.1;
C.alpha   = 2.331;
C.beta    = 2.2;
[I,disks] = createSubstrate(ra,N_ii,C,2);
save(sprintf("Results/Substr/Nii=%d.mat",N_ii),'I','disks','N_ii','ra');
%disks

%draw substrate
figure; [X,Y] = meshgrid(linspace(-ra*1.3,ra*1.3,N_ii));
subplot(1,3,1),pcolor(X,Y,I),axis image,shading interp,

%% sequence
gRes = 0.001; gSteps = gMax/gRes;
G=0*t; G(t<=delta)=1; G(t>=Delta&t<=Delta+delta)=-1;
dir=[1,0]; G=kron(dir,G');
seq.G = G; seq.t = t; seq.G_s = 0:gRes:gMax;
bVal = (gam^2 * seq.G_s.*seq.G_s * delta^2 * (Delta - delta/3));

%% simulation
tic
simu.r = X(1,2)-X(1,1);
simu.N = N_rw;
simu.D = diffu;
simu.P = flip([perma; flip(perma)]);
S = diffSim(I,seq,simu,gam);
elapsedTime = toc;
save(sprintf('Results/Sim2/Nii=%d Nrw=1e%d Nt=1e%d',N_ii,log10(N_rw),log10(N_t)));
```

8.2.10. Quadtree.m

```
% Two-layer structure

%% params
ra    = [2.5,5.0]*1e-6;
gam   = 2.675e8;
gMax  = 0.5;
delta = 50e-3;
Delta = 100e-3;
tf    = 2*Delta;
diffu = [2.0,2.0,2.0]*1e-9;
perma = [1.0,0]*1e-5;
relax = [Inf Inf];
N_ii  = 2^5; % must be a power of 2
N_rw  = 1e3;
N_t   = 1e3;
t     = linspace(0,tf,N_t);

%% substrate
[X,Y] = meshgrid(linspace(-ra(2)*1.3,ra(2)*1.3,N_ii));
d = sqrt(X.^2+Y.^2);
I = uint8(d<=ra(2)) + uint8(d<ra(1));

%% sequence
gSteps = 1e3;
gRes = gMax/gSteps;
G=0*t; G(t<=delta)=1; G(t>=Delta&t<=Delta+delta)=-1;
dir=[1,0]; G=kron(dir,G');
seq.G = G; seq.t = t; seq.G_s = 0:gRes:gMax;
bVal = (gam^2 * seq.G_s.*seq.G_s * delta^2 * (Delta - delta/3));

%% analytical answer
if ~exist ('Sa','var') % debugging - save time
    ML.d = 2; ML.r = ra; ML.D = [diffu(1), diffu(2)]; ML.W = [perma(1),perma(2)]; ML.T
= relax;
    Sa = ML_compute(seq.G_s,delta,Delta,ML);
end

%% quadtree - provides shortcuts to take during a random walk
% simulation parameters
dim = 2;
r   = X(1,2)-X(1,1); % parent resolution
dt  = t(2)-t(1);
Pt  = perma(1)* sqrt((dt*pi)/diffu(2)); % Pt
D   = diffu;
P   = [-1 -1 -1; -1 +1 Pt; -1 Pt +1]; % 11 permitted if ECS used
N_i = size(I);
N_p = N_rw;

% quadtree and properties
Q.quadtree = qtdecomp(I,0,2); % indexed by pos in I
[x,y,d_ii] = find(Q.quadtree); blocks = [x+(d_ii/2),y+(d_ii/2),d_ii];
Q.topLeft  = [x,y];
Q.count    = length(blocks);
Q.centre   = [blocks(:,1) blocks(:,2)];
Q.dx       = d_ii;
Q.ind      = maskPos(Q.centre*r,N_i,r);
Q.mask     = I(Q.ind);

%% simulation
% model as a network - create adjacency matrix
Q.adj = zeros(Q.count); % -1 = not connected (0 reserved for permeability)
for a=1:Q.count
    x0 = Q.topLeft(a,1);x1 = Q.topLeft(a,1)+Q.dx(a);
    y0 = Q.topLeft(a,2);y1 = Q.topLeft(a,2)+Q.dx(a);
    boundingBox = [[x0 y0];[x1 y0];[x0 y1];[x1 y1]];
    for b=1:Q.count % get all nodes b connected to a
```

```
        if a~=b
            if      all(Q.topLeft(b,:)==boundingBox(1,:)) || ...
                    all(Q.topLeft(b,:)==boundingBox(2,:)) || ...
                    all(Q.topLeft(b,:)==boundingBox(3,:)) || ...
                    all(Q.topLeft(b,:)==boundingBox(4,:))
                Q.adj(a,b) = P(Q.mask(b)+1,Q.mask(a)+1); % weight edges w/permeability
                Q.adj(b,a) = P(Q.mask(a)+1,Q.mask(b)+1);
            end
        end
    end
end

% radius search algorithm with variable search size
%T = graph(Q.quadtree); plot(T);
%A = adjacency(T,'weighted'); [a1,a2,a3] = find(A);

% throw in particles at permitted positions
availPos = Q.centre(Q.mask~=1,:);

% select from availPos, N_p times
ind = zeros(N_p,2);
pos = zeros(N_p,2);
for ii=1:N_p
    seed = randi(length(availPos),1);
    ind(ii,:) = [availPos(seed,1),availPos(seed,2)];
    pos(ii,1) = X(1,ind(ii,1));
    pos(ii,2) = X(2,ind(ii,2));
end
startPos = pos;
phase = zeros(N_p,1);
moves = zeros(N_p,2);
% the random walk
% skip corresponding dt and dx based on box size
for n=1:N_p
    ii = 1;
    tt = 1;
    while tt < length(t) - max(Q.dx)
        oldPos = pos(n,:);
        oldInd = ind(n,:);
        oldNode = find(Q.centre(:,1)==oldInd(1) & Q.centre(:,2)==oldInd(2));
        jumpSize = Q.dx(oldNode);

        neighbours = findNeighbours(oldNode,Q.adj);
        newNode = neighbours(randi(length(neighbours),1));

        newInd = Q.ind(newNode);
        newPos(1) = X(newInd); newPos(2) = Y(newInd);

        Pt = Q.adj(oldNode,newNode);
        prob = rand(1,1);
        if prob<=Pt
            pos(n,1) = newPos(1);
            pos(n,2) = newPos(2);
        end

        %moves(n,1) = moves(n,1) + pos(n,1);
        %moves(n,2) = moves(n,2) + pos(n,2);

        tt = tt + jumpSize;
        ii = ii + 1; % number of total steps taken

        displ = [(startPos(n,1) - pos(n,1)),(startPos(n,2) - pos(n,2))];

        phase(n) = phase(n) + jumpSize*sum(prodsum(displ,G(tt,:)),2);
    end
end
```

```
%% compute signal
if ~isfield(seq,'G_s'),seq.G_s=1;end
S=zeros(length(seq.G_s),1);
for gg=1:length(S),S(gg)=mean(exp(1j*gam*dt*phase*seq.G_s(gg)));end

%% plot results
figure('Position', [200 200 2000 800]),
%subplot(1,3,1),pcolor(X,Y,I),axis image,shading interp,
subplot(1,3,2),semilogy(bVal,real(S)),hold on,semilogy(bVal,Sa,'r-'), %axis([0,
2.5e11, 1e-2,1])
legend('monte carlo','grebenkov'),title('Results')
ylabel('Signal attenuation'),xlabel('B-Value')
subplot(1,3,3),plot(bVal,abs((Sa-real(S'))./Sa)),title('relative error'), %axis([0,
2.5e11, 1e-2,1])
annotation('textbox',[.0 1.0 .0 .0], ...
    'String',sprintf('N_{RW}=%d, N_{ii}=%d, N_t=%d, N_{G_s}
=%d',N_rw,N_ii,length(t),length(seq.G_s)),'EdgeColor','none')
```

8.2.11. GAFRW.m

Side note: Attribution for this code is partially derived from the works of a similar study from the University of Waterloo [117]. Here, the author details functions for calculating the exit times and a fast random walk, albeit using a slightly different framework to the method in our study. We were unable to tune and test the algorithm during study, but have included the framework for doing so below.

```
% Two-layer structure

%% params
ra    = [2.5,5.0]*1e-6;
gam   = 2.675e8;
gMax  = 0.5;
delta = 50e-3;
Delta = 100e-3;
tf    = 2*Delta;
diffu = [2.0,2.0,2.0]*1e-9;
perma = [1.0,0]*1e-5;
relax = [Inf Inf];

N_ii  = 2^5;
N_rw  = 1e2;
N_t   = 1e2;
t     = linspace(0,tf,N_t);

const = Tables_FRW;

%% substrate
[X,Y] = meshgrid(linspace(-ra(2)*1.3,ra(2)*1.3,N_ii));
d = sqrt(X.^2+Y.^2);
I = uint8(d<=ra(2)) + uint8(d<ra(1));

%% sequence
gSteps = 1e3;
gRes = gMax/gSteps;
G=0*t; G(t<=delta)=1; G(t>=Delta&t<=Delta+delta)=-1;
dir=[1,0]; G=kron(dir,G');
seq.G = G; seq.t = t; seq.G_s = 0:gRes:gMax;
bVal = (gam^2 * seq.G_s.*seq.G_s * delta^2 * (Delta - delta/3));

%% analytical answer
if ~exist ('Sa','var') % debugging - save time
    ML.d = 2; ML.r = ra; ML.D = [diffu(1), diffu(2)]; ML.W = [perma(1),perma(2)]; ML.T
= relax;
    Sa = ML_compute(seq.G_s,delta,Delta,ML);
end

%% simulation with GAFRW
% setup
N_i = size(I);
dim = numel(N_i);
N_c = sum(unique(I(:))>0);
r   = X(1,2)-X(1,1); % for I, not quadtree
dt  = t(2)-t(1);
D   = diffu;
dx  = sqrt(2*dim*dt*D');
Pt  = 1.0e-5* sqrt((dt*pi)/diffu(2)); % Pt
if N_c>1
        P = [0 0 0; 0 +1 Pt; 0 Pt +1]; % 11 permitted if ECS used
    else
        P = NaN;
end
% throw in particles
N       = round(N_rw*numel(I)/sum(I(:)>0));
pos     = prodsum(rand(N,dim),r.*N_i);
ind_s   = maskPos(pos,N_i,r);
% delete points in non-diffusion regions
pos(I(ind_s)==0,:)    = [];
```

```
ind_s(I(ind_s)==0)  = [];
% total particles remaining
N_p     = size(pos,1);
phase   = zeros(N_p,1);

% fast random walk
Tm = max(t);
%for tt=1:N_t
for n=1:N_p % do for each particle separately
    Tr = Tm;
    while Tr > 0
        curPos = pos(n,:);
        curComp = I(maskPos(curPos,N_i,r));
        [k,dist] = dsearchn(curPos,[X(I~=curComp),Y(I~=curComp)]);
        dist = min(dist);
        tau = gen_tau(const);
        t_unit = dist^2/D(1);
        dt = t_unit * tau;
        u = randCirc(1,dim);

        if dt < Tr
            Tr = Tr - dt;
        else
            dt = Tr;
            Tr = 0;
            tau = Tr / t_unit;
        end
        % subtend search radius
        posStep = prodsum(randCirc(1,dim),dist);
        ind_s = maskPos(curPos,N_i,r);
        ind_p = maskPos(mod(curPos+posStep,N_i(1)*r),N_i,r);
        Ie = I(ind_s)==I(ind_p);
        if ~isnan(P(1))
            if Ie ~= 1
                a = I(ind_s);
                b = I(ind_p);
                if a>0 && b>0
                    Pt = P(a+1,b+1);
                    prob = rand(1,1);
                    if prob <= Pt
                        Ie = 1;
                    end
                end
            end
        end
        newPos = curPos + posStep;
        %phase(n) = phase(n) + sum(prodsum(newPos,G(tt,:)),2);
        dph_nl = (evalPhi1(const, tau)* dist * t_unit * u);
        phase(n)=phase(n)+sum(prodsum(dt * newPos, dph_nl));

%         M1=rem(newPos,N_i(1)*r);
%         M2=mod(newPos,N_i(1)*r);
%         ind_bc=(M2~=newPos).*sign(M1);
%         % (!) NPA read up
%         dph_nl = (evalPhi1(const, tau)* dist * t_unit);% * u);
%         phase(n)=phase(n)+(dt * newPos + dph_nl);
%         %sum(prodsum(-ind_bc*N_i(1)*r,sum(G(1:tt,:))),2);
%         newPos=M2;

    end
end
%end

% compute signal
if ~isfield(seq,'G_s'),seq.G_s=1;end
S=zeros(length(seq.G_s),1);
for gg=1:length(S),S(gg)=mean(exp(1j*gam*dt*phase*seq.G_s(gg)));end
```

```
%% plot
figure('Position', [200 200 2000 800]),
%subplot(1,3,1),pcolor(X,Y,I),axis image,shading interp,
subplot(1,3,2),semilogy(bVal,real(S)),hold on,semilogy(bVal,Sa,'r-'), %axis([0,
2.5e11, 1e-2,1])
legend('monte carlo','grebenkov'),title('Results')
ylabel('Signal attenuation'),xlabel('B-Value')
subplot(1,3,3),plot(bVal,abs((Sa-real(S'))./Sa)),title('relative error'), %axis([0,
2.5e11, 1e-2,1])
annotation('textbox',[.0 1.0 .0 .0], ...
    'String',sprintf('N_{RW}=%d, N_{ii}=%d, N_t=%d, N_{G_s}
=%d',N_rw,N_ii,length(t),length(seq.G_s)),'EdgeColor','none')

%% functions (makeshift)
% Attribution: Mao Ming, University of Waterloo, 2019
function tau = gen_tau(const)
    U = rand;
    while U == 1, U = rand, end
    if U < 1e-5
        A = 2*log(U*sqrt(pi)/2);
        t = 1; % initial guess;
        f = t*log(t)+A*t+1/2;
        err = abs(f);
        df = log(t)+1+A;
        while err > eps
            t = t - f/df;
            f = t*log(t)+A*t+1/2;
            err = abs(f);
            df = log(t)+1+A;
        end
        tau = t;
    elseif U < 0.99
        tau = interp1(const.F_tau(:,2),const.F_tau(:,1),U);
    else
        tau = log(2/(1-U))/(pi^2);
    end
end
function const = Tables_FRW()
    load('PhysicalConstants.mat','D','gamma')
    load('Ftau.mat','newFtau')
    load('Phi1.mat','Phi1');
    load('Phi2.mat','Phi2');
    load('R(t).mat','Rtable');
    const.D = D;
    const.gam = gamma;
    const.F_tau = newFtau(2:135,:);
    const.Phi_1 = Phi1(1:800,:);
    const.Phi_2 = Phi2;
    const.R_rms = Rtable(1:200,:);
end
function output = evalPhi1(const, tau)
    if tau < 0.05
        output = tau*(1-2*tau)/2;
    elseif tau < 0.8
        output = interp1(const.Phi_1(:,1),const.Phi_1(:,2),tau);
    else
        output = 0.75/pi^2;
    end
end
function output = evalPhi2(const, tau)
    if tau < 1e-3
        output = sqrt(2*tau^3/3);
    else
        output = interp1(const.Phi_2(:,1),const.Phi_2(:,2),tau);
    end
end
```

8.3. *Quadtree plots*

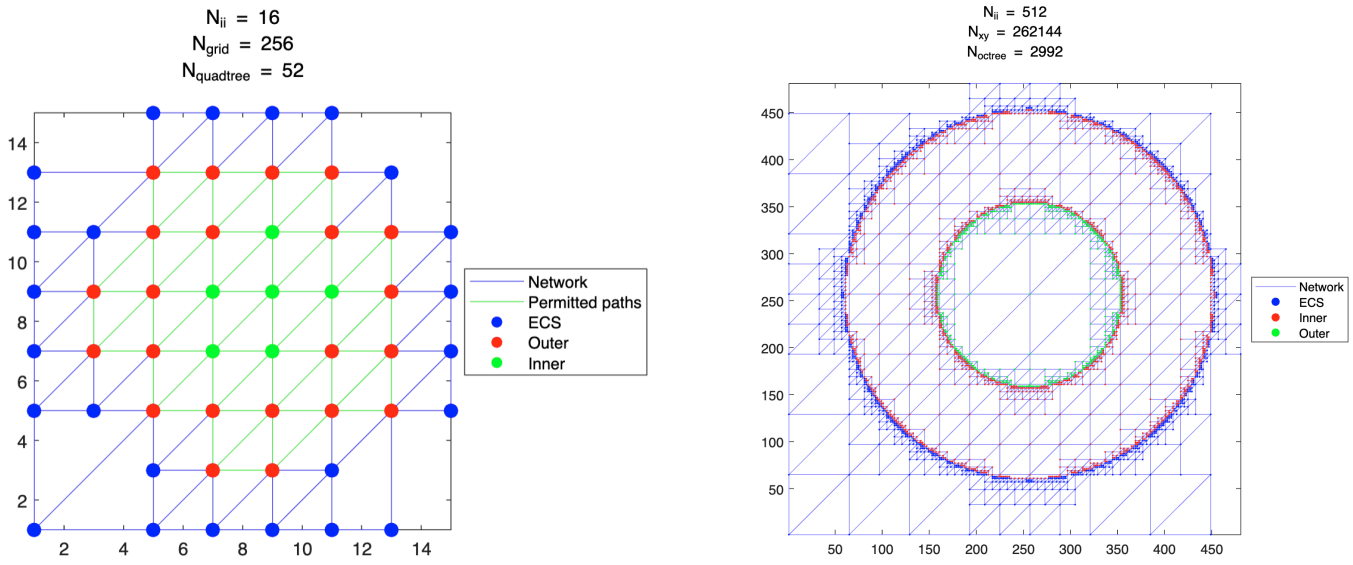


Fig 30. Quadtree plots for Simulation 1 for N_{ii} = 16 (left) and N_{ii} = 512 (right). Notice how the number of quadtree centre coordinates is a drastic reduction of the total coordinates on the grid.

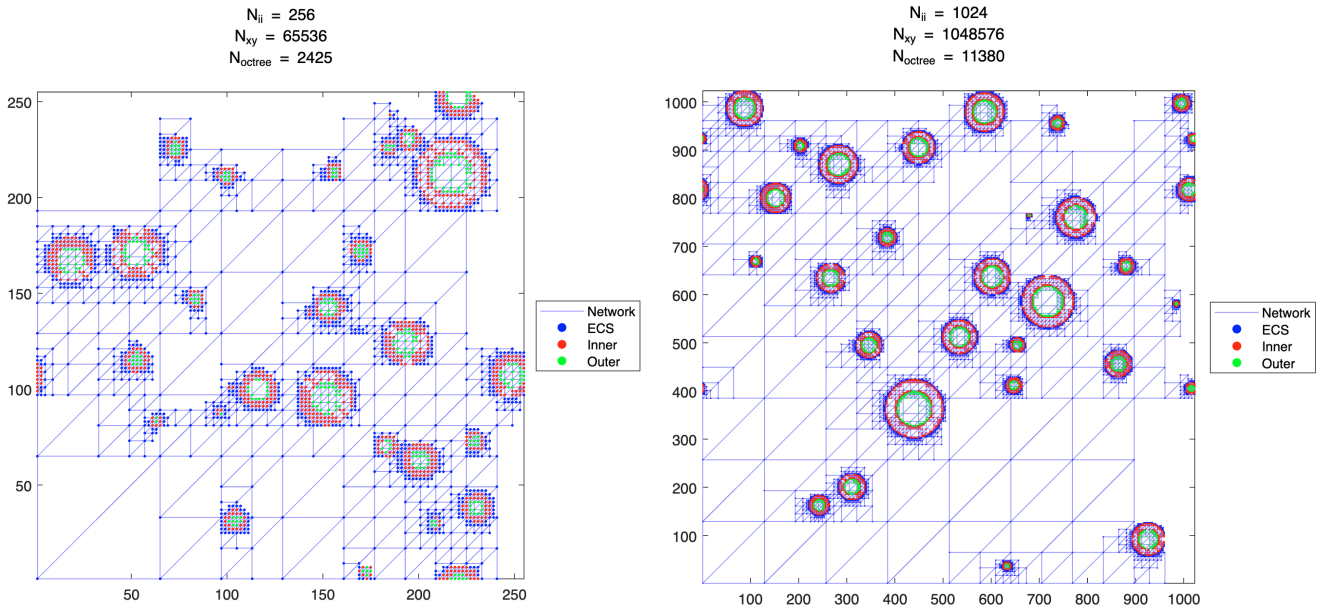


Fig 31. Quadtree plots for Simulation 2 for N_{ii} = 256 (left) and N_{ii} = 1024 (right). Increased resolution will increase the likelihood of capturing the geometry of smaller two-compartment disks. The effects of resolution are even more important as less points are used to represent different compartments.

8.4. Quadtree creation diagram

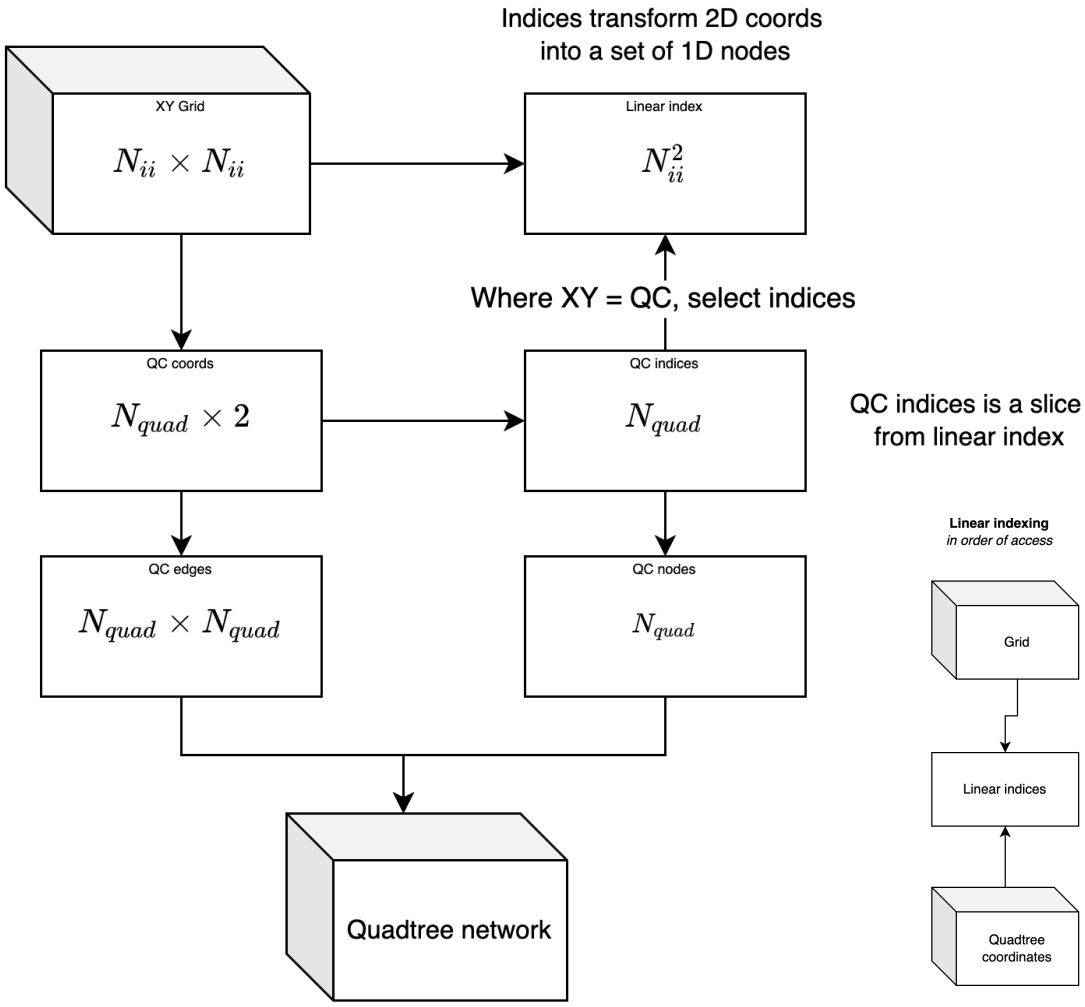


Fig 32. The data flow diagram for quadtree decomposition of a substrate, supplemented by the linear indexing diagram. Note that QC is the domain of quadtree block centre points.

8.5. *Full simulation plots*

8.5.1. *Simulation 1*

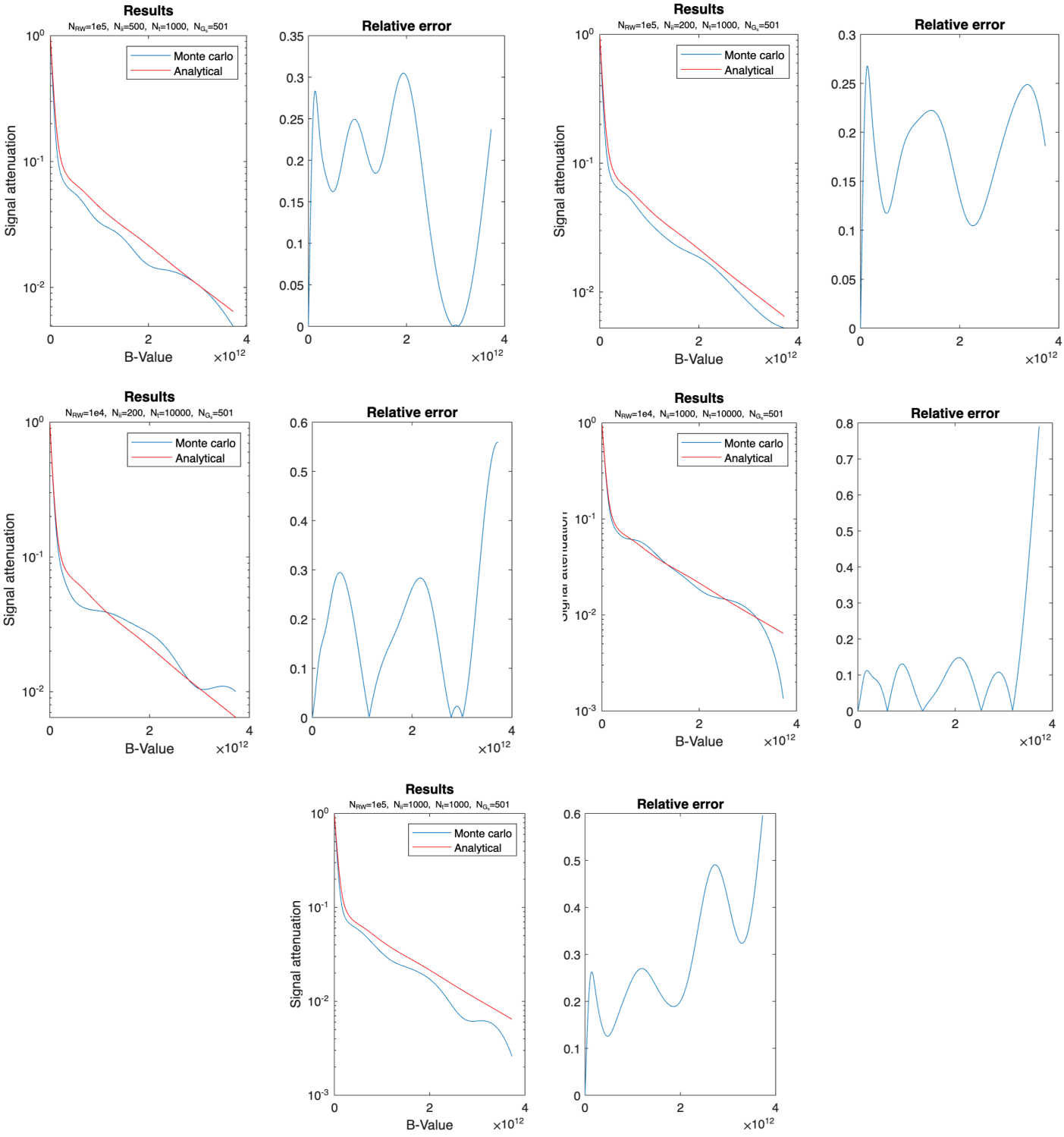


Fig 33. The results and relative error plots for all five experiments which minimised MRAE in *Simulation 1*.

8.5.2. *Simulation 2*

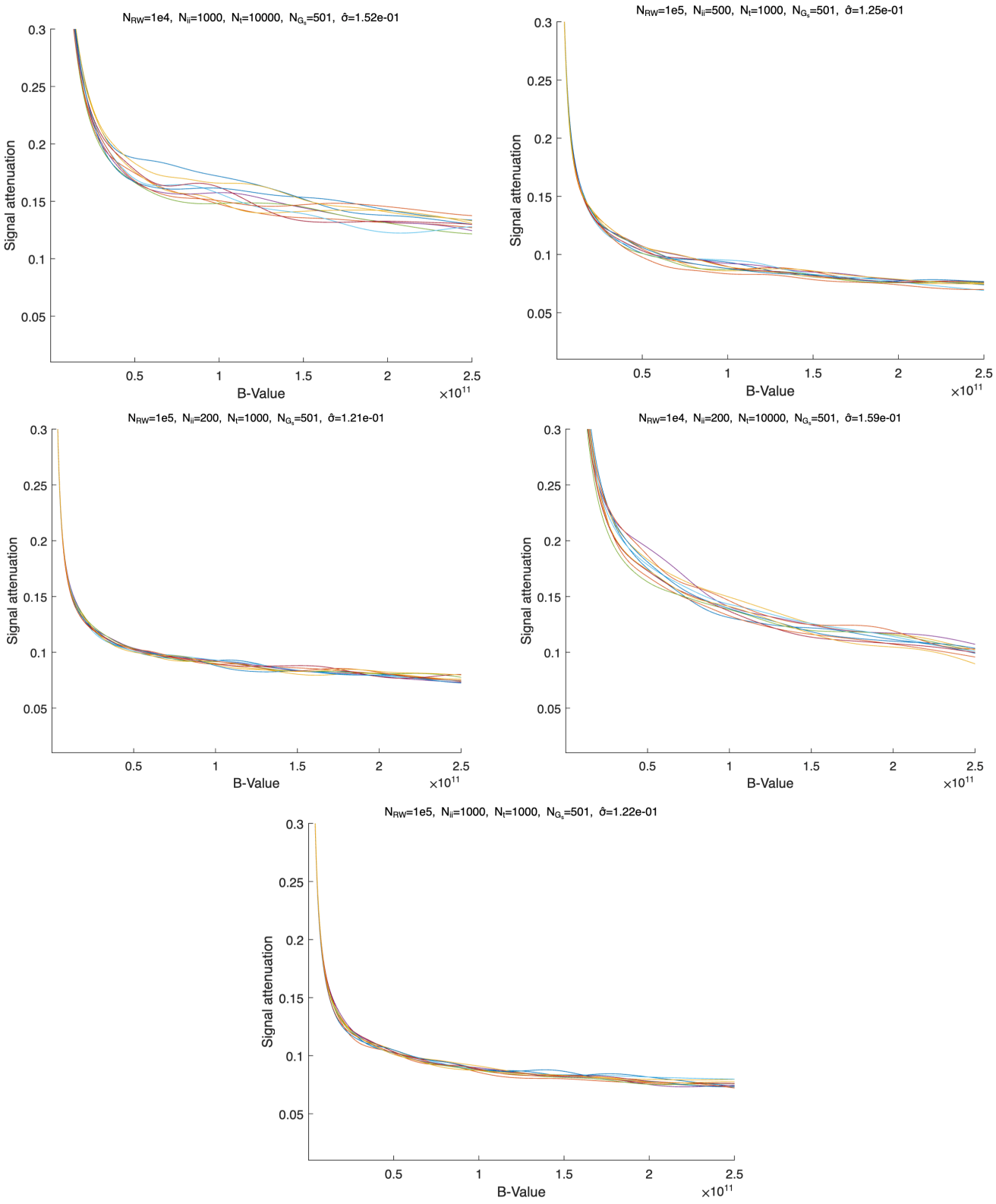
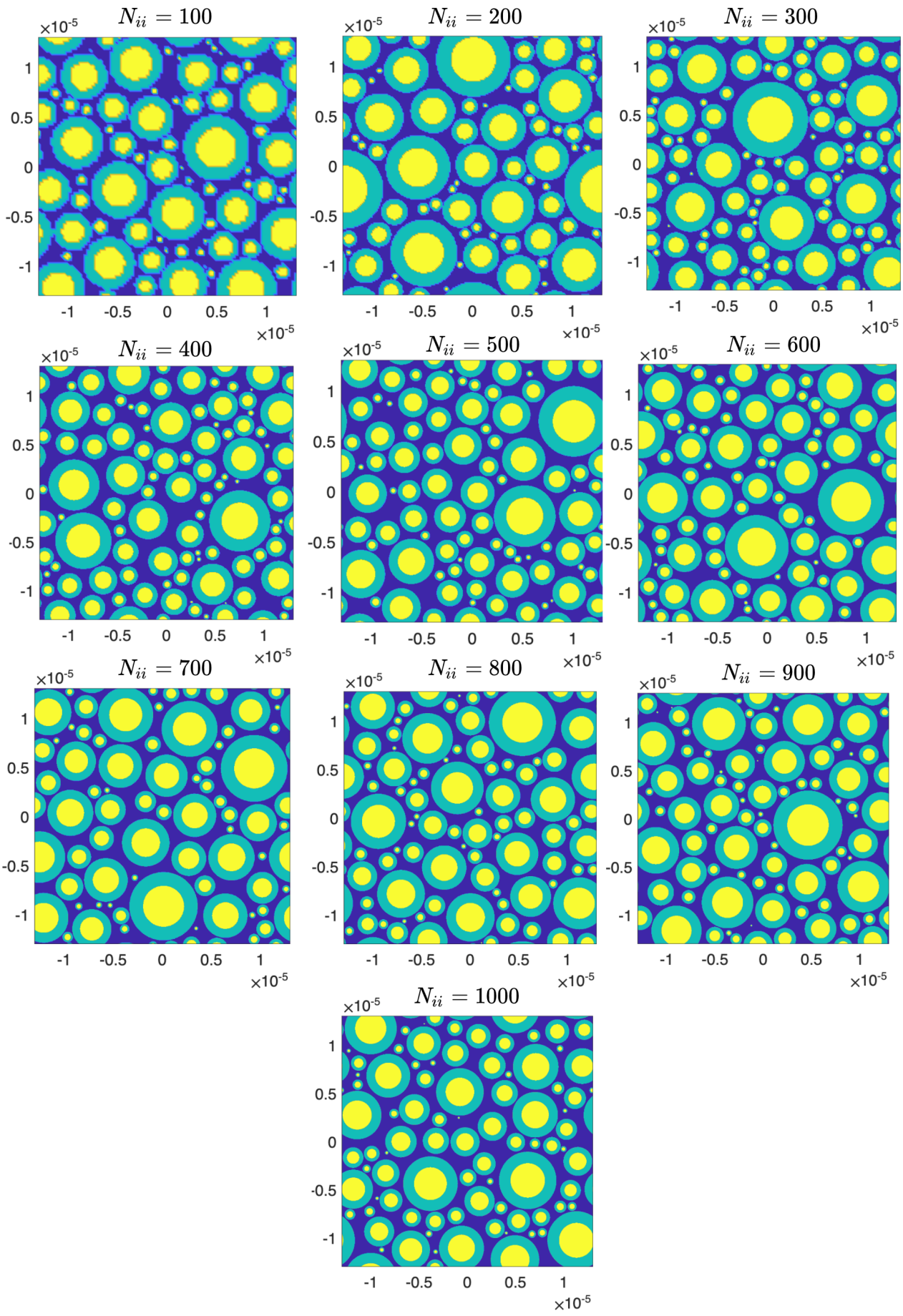


Fig 34. The results from *Simulation 2* for the same values, with repeat experiments being differentiated by colour. The lower the SD, the tighter the curve plots will be.

8.6. *Substrate plots used in Simulation 2*



8.7. Initial run for Quadtree method

Side note: This method was not *fully* evaluated but we were able to implement it and obtain some results.

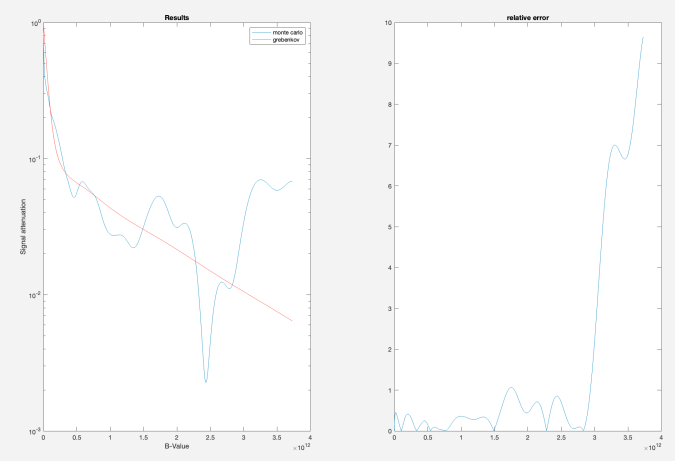


Fig 35. N_{rw} = 10^3, N_{ii} = 32, N_t = 10^2, N_{Gs} = 1001

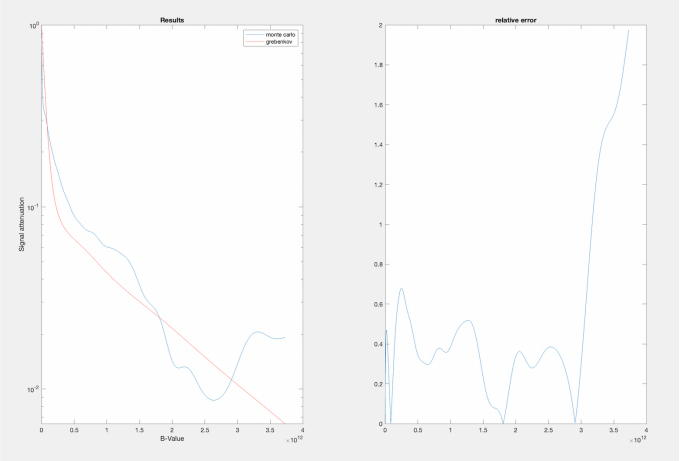


Fig 36. N_{rw} = 10^5, N_{ii} = 32, N_t = 10^2, N_{Gs} = 1001

8.8. Size efficiency gains from using Quadtree structure

```
>> whos B
  Name        Size              Bytes  Class     Attributes

  B          52x52             21632  double

>> whos A
  Name        Size              Bytes  Class     Attributes

  A          52x52              4424  double    sparse
```

Fig 37. By removing the zero-values from the matrix, the sparse matrix format can dramatically reduce the size required in memory whilst retaining dimensions.