Journal of Magnetic Resonance 211 (2011) 67–73

Contents lists available at ScienceDirect

Journal of Magnetic Resonance

journal homepage: www.elsevier.com/locate/jmr

GPU accelerated Monte Carlo simulation of pulsed-field gradient NMR experiments

Christopher A. Waudby *, John Christodoulou

Institute of Structural and Molecular Biology, University College London and Birkbeck College, WC1E 6BT, UK

ARTICLE INFO

Article history:
Received 8 February 2011
Revised 7 April 2011
Available online 23 April 2011

Keywords:
CUDA
GPGPU
Parallel processing
Restricted diffusion
PGSE

ABSTRACT

The simulation of diffusion by Monte Carlo methods is often essential to describing NMR measurements of diffusion in porous media. However, simulation timescales must often span hundreds of milliseconds, with large numbers of trajectories required to ensure statistical convergence. Here we demonstrate that by parallelising code to run on graphics processing units (GPUs), these calculations may be accelerated by over three orders of magnitude, opening new frontiers in experimental design and analysis. As such cards are commonly installed on most desktop computers, we expect that this will prove useful in many cases where simple analytical descriptions are not available or appropriate, e.g. in complex geometries or where short gradient pulse approximations do not hold, or for the analysis of diffusion-weighted MRI in complex tissues such as the lungs and brain.

1. Introduction

NMR pulsed-field gradient (PFG) experiments such as the pulsed-gradient spin-echo (PGSE) are a powerful tool for the characterisation of molecular transport processes in a wide range of systems [1–3]. Porous media, such as rocks, foams, cells and more complex tissues, are of importance to many fields, from geology to biology, and PFG NMR experiments are capable not just of characterising diffusion within such media, but also of revealing geometric information about the environment, such as pore sizes and tortuosity [2,4–9]. Moreover, these methods form the basis of a variety of diffusion-weighted magnetic resonance imaging (MRI) techniques, ranging from a simple contrast mechanism to diffusion-tensor imaging and neural tractography [10–13].

The results of PFG experiments can be expressed analytically in a closed form only for a small number of relatively simple pulse sequences and confining geometries. Often simplifications such as short gradient pulse or Gaussian phase approximations are required [2,3,14], yet there are many interesting cases where these approximations do not hold [15–19]. Rigorous analytical solutions may be computed using matrix formulations of the Bloch–Torrey equation (describing the magnetisation of diffusing spins) [3,20–23], but for complex geometries this still remains a complex task. An attractive alternative therefore remains the simulation of a large number of diffusing spins, in order to calculate directly the expected experimental results [15,24–26].

The echo attenuation or relative intensity measured by a PFG experiment, $E = I/I_0$, is the ensemble average of the phase, φ, accumulated by a spin due to the magnetic field experienced along its trajectory. As virtually all experiments, including spin and stimulated echos, are designed to refocus the evolution of magnetisation due to the static field, we assume phase evolution is determined only by the pulsed-gradient field, $\mathbf{G}(t)$. To account for the effect of 180° pulses, this is conveniently combined with the coherence order, n, as an effective field $\mathbf{G}_{\text{eff}}(t) = n(t)\ \mathbf{G}(t)$ [2,3,20]. Thus, the phase accumulated by a single spin, over a time T, is:

$$\phi = \int_0^T \gamma \mathbf{G}_{\text{eff}}(t)\mathbf{r}(t)dt \tag{1}$$

where γ is the gyromagnetic ratio and $\mathbf{r}(t)$ describes the trajectory of the spin. The observed echo intensity is described by the ensemble-averaged phase:

$$E = \langle e^{i\phi} \rangle = \langle \cos\phi \rangle + i\langle \sin\phi \rangle \tag{2}$$

The above description of PFG experiments applies equally to simulations as to the physical experiment itself. In this case, N diffusive trajectories may be generated by a simple Monte Carlo algorithm, in which random displacements along each dimension during a time step δt are sampled from a Gaussian with variance $2D\delta t$. The implementation of boundary conditions, essential to the description of any confined system, is discussed further below. The average in Eq. (2) is to be taken over realisations of the diffusive trajectories of the particles, and also their initial positions and, for an isotropic sample, orientations of the surrounding geometry. In the absence of flow, the imaginary component (phase) is zero, so for many practical purposes it is only required to compute the $\cos\langle\phi\rangle$ term of Eq. (2) to calculate the echo attenuation. The statistical uncertainty is the standard error of the mean:

* Corresponding author.
E-mail address: c.waudby@ucl.ac.uk (C.A. Waudby).

1090-7807/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jmr.2011.04.004

$$\sigma_E = \sqrt{\frac{\langle \cos^2 \phi \rangle - \langle \cos \phi^2 \rangle}{N}} \quad (3)$$

A fundamental drawback with simulation approaches to the analysis of experiments is that a large number of trajectories (typically 10^5–10^6) are required to reduce statistical errors, the relative magnitude of which scale as $N^{-1/2}$. The simulation algorithm itself however, is an example of an *embarrassingly parallel* problem, due to the absence of interactions between individual trajectories. Graphics processing units (GPUs) are highly optimised for such parallel calculations, and modern hardware can process several hundreds of such problems ('threads') simultaneously. Recent years have seen an effort, loosely termed 'general-purpose computing on the GPU' (GPGPU), to harness this low-cost, high-performance hardware for scientific calculations [27], and a number of libraries such as OpenCL and Compute Unified Device Architecture (CUDA) have been developed to interface in relatively high-level languages such as C to the GPU hardware [28,29]. Here we show that by performing computation on the GPU, the Monte Carlo simulation of PFG NMR experiments may be accelerated by up to three orders of magnitude, opening up new possibilities in the design and analysis of experiments.

2. Implementation

The software has been implemented in C, using the CUDA library to interface with NVIDIA GPUs [28]. The code is freely available to download from the author's website (http://www.smb.ucl.ac.uk/christodoulou/software). The program flow is outlined in Fig. 1, depicting particularly the division of activities between the CPU and the GPU. For benchmarking and testing purposes, identical 'gold-standard' CPU implementations of GPU code have been written to provide a CPU-based reference. These were found particularly useful when implementing new algorithms, as it can be more complex to debug code that is running on a GPU.

There are a variety of possible implementations of boundary conditions in the simulation of restricted diffusion [30–33]. In the common case of reflecting (zero-flux) boundaries the most detailed approach would be, on detecting that a trajectory has exited the allowed region, to compute exactly the final position following (possibly multiple) specular reflection(s) from the boundary surface [30]. However, not only are these calculations highly demanding for arbitrarily defined geometries, but the detection and handling of multiple reflections gives rise to branches and loops within the GPU kernel. These are highly inefficient to execute, as the structure of the GPU multiprocessor demands that simultaneously executed 'warps' of 32 threads must follow the same program flow (although a small number of branches can be handled through predicated execution) [34]. Where this is not possible threads must be executed sequentially, and so a loop in a single thread can force the serial execution of the entire warp, destroying the expected performance gain from parallelism. We have therefore implemented an alternative 'rejection sampling' algorithm whereby the particle position is updated only if the new position is within an allowed region [31,32]. Not only is this computationally efficient, but the implementation of new geometries is reduced to defining a Boolean function that determines whether a given position is allowed or disallowed. Additionally, under steady-state conditions typical of diffusion measurements, there are no artifactual concentration gradients near reflecting boundaries [33]. Of course, it is important to ensure that the RMS displacement per time step, $\delta r = \sqrt{6D\delta t}$, is much smaller than the smallest geometric features, and so when implementing new geometries we recommend verifying that simulation results do not change when the time step is varied.

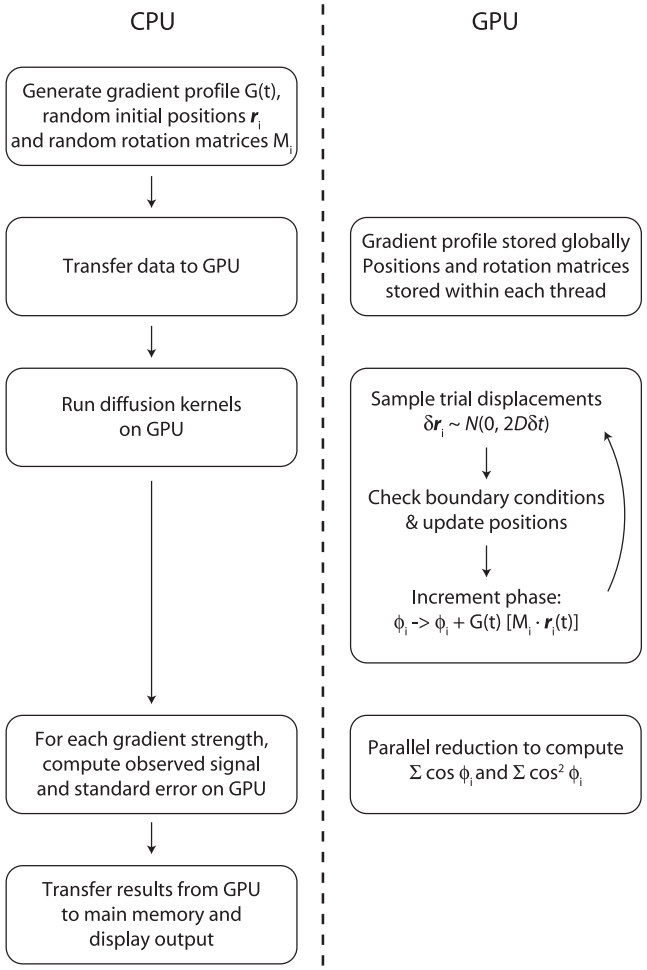

Fig. 1. Block diagram of the simulation algorithm, indicating the main program flow on the CPU and calculations performed on the GPU.

Surface relaxation, i.e. absorbing or partially absorbing boundaries, is also implementable using a rejection method for boundary conditions. A binary flag is associated with each trajectory, and upon collisions with the boundary this flag may, with a certain probability, be set to indicate that 'spin-death' has occurred and that this trajectory should not contribute to calculations of the echo attenuation. In the limit of weak surface relaxation, the killing probability is linearly proportional to the surface relaxivity [35], while in the opposite case of completely absorbing boundaries spin-death occurs with probability one. In general, the simulation of surface relaxation effects by random walk methods is computationally more intensive than similar simulations with reflecting boundaries. This is in part due to errors in the concentration gradients near surfaces, which arise as a consequence of the finite size simulation steps, irrespective of whether boundaries are implemented with a rejection algorithm or by more detailed calculation of specular reflections [33]; and also due to the increased statistical uncertainty from sampling only the smaller population of 'living' spins. Thus, both smaller time steps and a greater number of particles may need to be simulated to reach an equivalent level of accuracy to simulations with reflecting boundaries, and in these cases acceleration with the GPU may be particularly useful.

Random number generation is essential to several aspects of the simulation process. Firstly, random initial positions were generated on the CPU, with rejection sampling to ensure boundary conditions are not violated, and then transferred to the GPU memory.

Secondly, an extremely large number of random displacements must be sampled by the GPU from a Gaussian distribution with variance $2D\delta t$ (where D is the diffusion coefficient and δt is the simulation time step). Random numbers uniformly distributed in the unit interval were generated within each thread by an XOR-shift algorithm [36,37] and transformed into Gaussian variates using the Box–Muller transform (which although numerically intensive avoids branches and loops, resulting in efficient performance on the GPU) [38]. Finally, while in the first instance three-dimensional trajectories are generated in a fixed (shared) reference frame, to reflect the isotropic nature of most NMR samples we have provided the option to rotate each three-dimensional trajectory by a random (isotropically distributed) angle prior to computation of the phase. This is also equivalent to applying gradients in a random direction for each spin, and simplifies the task of specifying new geometries, as only the local environment need be described. Rotations are performed using randomly generated rotation matrices [38] computed on the CPU beforehand and stored locally within each thread.

2.1. Numerical precision and stability

The CUDA library provides two options for the evaluation of mathematical functions such as square root, sine and cosine: 'library' functions which are expanded by the compiler into longer sequences of instructions with guaranteed error bounds, and 'intrinsic' hardware-optimised functions with greatly improved performance, but potentially reduced accuracy [34]. We have evaluated the use of fast intrinsic functions in the random number generator, and by comparing the results of simulations using the fast, intrinsic functions against results from the 'gold-standard' CPU implementation, any reduction in simulation accuracy is not detectable within the typical statistical uncertainty of repeated simulations. However, the more accurate library functions are used in the final calculations of phase averages (Eqs. (2) and (3)), where the calculation cost is not a critical factor.

When analysing the numerical stability of an algorithm, it is important to consider the precision with which the raw numeric data is stored and manipulated by the hardware. If calculations are not performed with sufficient precision, rouding errors can occur: e.g. when adding a small number to a large number, the large number may not have sufficient precision to accurately represent the small number. GPUs have historically operated with single-precision floating point arithmetic, where each variable is represented by 32 bits (4 bytes) with 8 bits encoding the exponent and 24 encoding the significand, giving a precision of $\log_{10} 2^{24} \approx 7$ decimal digits. By contrast, computations on the CPU are commonly performed with double-precision arithmetic, with 11 bits encoding the exponent and 53 encoding the significand, giving a precision of approximately 16 decimal digits and virtually eliminating the possibility of rounding errors. While modern generations of NVIDIA cards (compute capabilities of 1.3 or greater [28,34]) can also perform double-precision arithmetic, this can come at a substantial performance cost. For our purposes, we find that single-precision arithmetic is sufficient provided that care is taken to avoid rounding errors when evaluating repeated summations (i.e. when small numbers may be added to large numbers). These might arise in the accumulation of phase within each trajectory, and particularly in the summations required to compute averages at the end of the simulation. The former problem has been addressed by using the Kahan summation algorithm [39], whereby a compensator variable is additionally computed to correct for small errors – effectively, using two single-precision variables to approximate a double-precision calculation, at a minimal performance cost.

Loss of precision in the final summations for computing phase averages (Eqs. (2) and (3)) may also be addressed using the Kahan summation algorithm. However, a more efficient alternative is to take advantage of the inherent parallelism of the GPU using a 'parallel reduction' algorithm, an iterative method whereby data are progressively summed in pairs (i.e. computing pair-wise sums, then sums of pairs, etc.) [40,41]. This approach benefits both performance and accuracy. Large arrays of phase data no longer need to be copied to the main memory (a slow process due to the limited bandwidth available) but can be rapidly processed in situ on the GPU. The total number of summations required for the analysis of N particles is also greatly reduced, from $O(N)$ in the direct and Kahan summation algorithms, to $O(\log_2 N)$ using the parallel reduction method. Moreover, all addition operations are now between numbers of similar magnitudes, which significantly reduces the potential for rounding errors, further improving the accuracy of the final result.

2.2. Analytical references

For the purposes of validation, we have selected three test cases for which echo attenuations can be computed analytically. Firstly, we recall unrestricted diffusion measured by a pulsed-gradient spin-echo with two trapezoidal gradient pulses having strength G and length δ, ramp time τ, and diffusion period Δ (Fig. 2A). Including the correction to the Stejskal–Tanner equation for the finite ramp times [42], the echo attenuation is:

$$E = \exp\left(-\gamma^2 G^2 \left[\left(\Delta - \frac{\delta}{3}\right)\delta^2 + \frac{1}{30}\tau^3 - \frac{1}{6}\delta\tau^2\right] D\right) = \exp(-bD) \qquad (4)$$

Secondly, we consider the constant gradient spin-echo of length T and gradient strength G as shown in Fig. 2B, applied to particles diffusing between parallel planes separated by length $2a$, with reflecting (no surface relaxation) or absorbing (complete surface relaxation) boundaries, and where the gradient is applied along the normal to the planes. This provides an example of restricted diffusion where the echo attenuation may be evaluated to arbitrary precision using an expansion of the propagator in the Laplacian eigenbasis [20]. Echo attenuations can be expressed in terms of

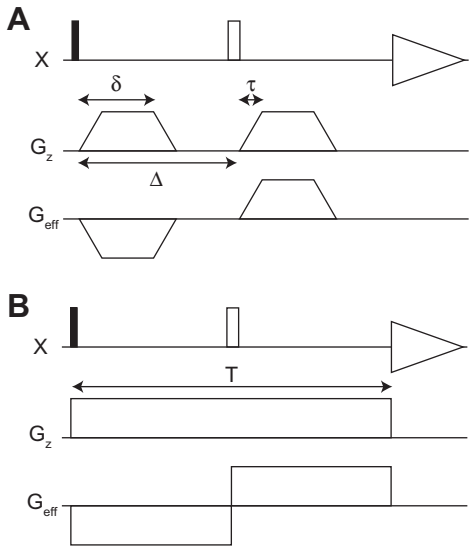

Fig. 2. Pulse sequences used for validation of the simulations, showing the effective gradient profile G_{eff} after including the effect of 180° pulses. (A) Pulsed-gradient spin-echo with trapezoidal gradients having ramp time τ and total area $G\delta$. (B) Constant gradient spin-echo with echo time T and gradient strength G.

the dimensionless parameters qa and p, where $q = \gamma GT/4\pi$ is the wavenumber of the magnetisation grating (i.e. $\delta = T/2$), and $p = DT/a^2$ is a dimensionless diffusion coefficient, expressing the echo time relative to the expected time to diffuse across distance a, from the centre of the system to the boundary. Calculations were carried out in Mathematica 8 (Wolfram Research, Champaign IL), truncating the expansion after 20 terms in the case of reflecting boundary conditions. Indistinguishable results were obtained when the expansion was further truncated to 10 terms, indicating that the calculation had converged satisfactorily. However, in the case of absorbing boundary conditions (i.e. surface relaxation) 2000 terms were required to attain the same degree of convergence. The slow convergence in such cases has been noted previously [20], and provides further impetus for rapid Monte Carlo simulations.

Finally, we consider unrestricted diffusion in the presence of uniform flow, where the echo signal acquires a non-zero phase term, ϕ, which in the short gradient pulse limit is related to the flow velocity, v:

$$\phi = \gamma\delta G\Delta v \tag{5}$$

In this case the imaginary component of Eq. (2) cannot be neglected, but must be calculated in order to compute the phase:

$$\langle\phi\rangle = \tan^{-1}\left(\frac{\langle\sin\phi\rangle}{\langle\cos\phi\rangle}\right) \tag{6}$$

The echo amplitude is independent of flow (in our case but not generally, e.g. in the presence of convection [43]) and is determined by diffusion alone, according to Eq. (4), although the amplitude must now be calculated using both real and imaginary components of the echo signal (Eq. (2)).

3. Results

3.1. Free diffusion

Observations of free diffusion ($D = 10^{-10}$ m^2 s^{-1}) were simulated using a 25 ms pulsed-gradient spin echo, with trapezoidal gradients having a 0.1 ms ramp time, a 5 ms pulse length and a maximum strength of 1 T m^{-1}, under which conditions ($b_{max}D = 4$, Eq. (4)) a large echo attenuation is expected. Simulations were performed on 2^{20} = 1,048,576 particles with a 10 μs time step, including rotation into randomly distributed reference frames to validate the rotation algorithm. Simulated echo attenuations are plotted in Fig. 3A and show excellent agreement with the theoretical expectation from Eq. (4).

Residuals between simulation and theory are plotted in Fig. 3B, also showing 68% confidence intervals (the standard error of the mean). Within the statistical uncertainty the agreement is nearly exact. However, when a naive direct summation is used instead of parallel reduction to compute the phase averages, we observe systematic errors in the simulation results due to rounding errors in the single-precision arithmetic (Fig. 3B, red[1] curve). Although small (a maximum error of 0.3%) such errors have the potential to accumulate unpredictably, particularly should even larger numbers of trajectories be simulated.

3.2. Diffusion between parallel planes

Simulation conditions were chosen to explore diffusion between two parallel planes, with reflecting boundaries, across a range of diffusional regimes from free diffusion and localisation to motional narrowing, as described by Grebenkov [20]. qa, the dimensionless gradient strength, was varied from 0 to 2.1, to explore a range of length scales both greater and smaller than the inter-plane separation. Three simulations were performed with values of the dimensionless diffusion coefficient, p, varied from 0.64 (free diffusion and localisation) to 64 (motional narrowing). These choices correspond to the physical parameters $D = 10^{-9}$ m^2 s^{-1}, $2a = 5$ μm, $T = 4$, 40 and 400 ms, and $G_{max} = 10$, 1 and 0.1 T m^{-1}.

Fig. 3C plots simulated and theoretical attenuations for the three echo times described above. As for the unrestricted case, simulations were performed on 2^{20} = 1,048,576 particles with a 4 μs time step, during which the RMS displacement along one dimension is $\sqrt{2DT} = 0.09$ μm, much smaller than the 5 μm distance between the planes. Simulation results did not vary with changes in the step size, indicating that the boundary conditions were adequately implemented. Again, the match between simulation and theory was excellent, further indicated by the residual plots in Fig. 3D which show the agreement to be nearly exact within the statistical uncertainty.

A fourth simulation was also performed to demonstrate and test the implementation of absorbing boundary conditions (i.e. surface relaxation), using similar parameters as above, with $p = 0.64$ (Fig. 3C, orange). A smaller time step of 4 ns, and a greater number of particles, $N = 2^{22} = 4.2$ million, were required in order for simulations to converge with a similar level of agreement as for reflecting boundaries, and the total simulation time therefore increased proportionally (although more detailed theoretical calculations were also required). The reported uncertainty in the echo attenuation (Eq. (3)) was extended to account for the additional uncertainty, $N_{alive}^{1/2}$, in the number of 'living' spins, N_{alive}, and again the agreement thus achieved with theoretical results was nearly exact within the statistical uncertainty.

3.3. Free diffusion with uniform flow

Pulsed-field gradient spin-echo experiments were simulated for free diffusion ($D = 10^{-9}$ m^2 s^{-1}) with uniform flow of velocity, v, parallel to the applied gradient and varied from 0 to 100 μm s^{-1}. These parameters were selected such that, under the applied experimental conditions ($\Delta = 0.1$ s, $\delta = 2$ ms, $\tau = 0.01$ ms, $G_{max} = 0.4$ T m^{-1}) substantial echo attenuations are expected ($b_{max}D = 4.5$, Eq. (4)). 2^{20} particles were simulated with a time step of 10 μm, and echo attenuations and echo phases thus obtained are plotted in Fig. 3E and F alongside theoretical calculations (Eqs. (4) and (5)). Due to the singularity in Eq. (6) when the phase approaches $\pm\pi/2$, or when the total amplitude is small, errors were propagated by standard Monte Carlo methods. Again, excellent agreement is observed in all cases, with no large deviations beyond the statistical uncertainty.

3.4. Performance

Simulation performance was investigated using an Intel Core 2 Quad (2.4 GHz) processor equipped with an NVIDIA GTX 280 graphics processing card. This GPU, a model introduced in June 2008, comprises 240 cores operating at 1.3 GHz, with a theoretical peak performance of 933 GFLOPS (floating point operations per second). The CPU has a nominal peak performance of 38.4 GFLOPS, assuming full utilisation of processing cores and of the single-instruction multiple-data architecture (SIMD, which allows the calculation of 4 single-precision operations simultaneously). However, due to the complexity of writing efficient parallel code for CPUs (discussed further below) the CPU benchmarks have not been optimised in these regards, resulting in an effective peak performance of only 2.4 GFLOPS, 388 times less than the GPU. In

[1] For interpretation of color in Fig. 3, the reader is referred to the web version of this article.

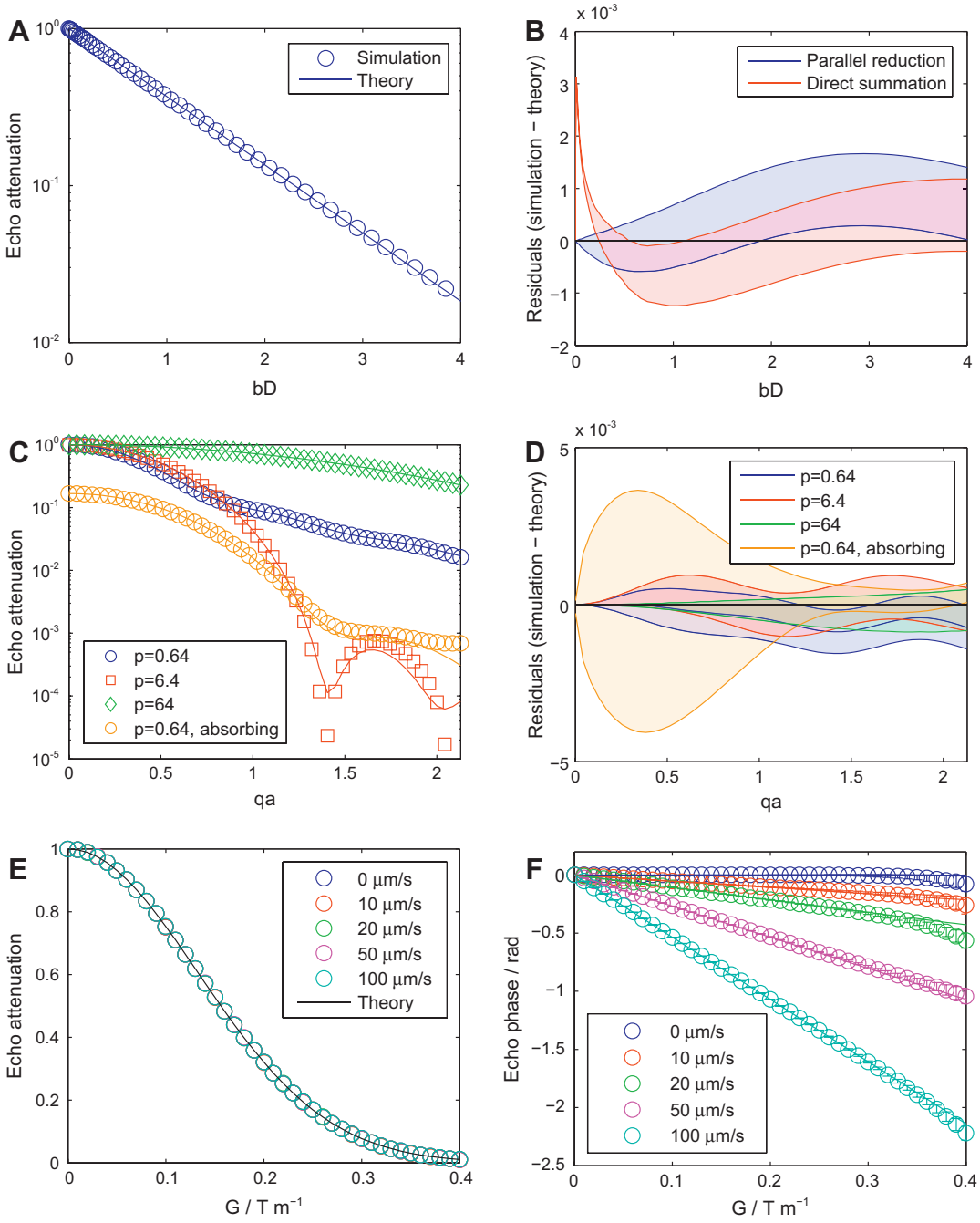

Fig. 3. Validation of the simulation algorithm. (A) Simulated (symbols) and theoretical (line, Eq. (4)) pulsed-gradient echo attenuations for unrestricted diffusion. (B) Residual plots of the difference between simulation results and theoretical expectations plotted in (A), showing the effect of computing phase averages by parallel reduction and by direct summation. 68% confidence intervals (standard error of the mean) are shown as shaded regions. (C) Simulated (symbols) and theoretical (line) gradient echo attenuations for diffusion between parallel planes, with reflecting boundaries except where specified. (D) Residual plots of the difference between simulation results and theoretical expectations plotted in (C). 68% confidence intervals (standard error of the mean) are shown as shaded regions. (E) Simulated (symbols) and theoretical (line, Eq. (4)) pulsed-gradient echo attenuations for unrestricted diffusion ($D = 10^{-9}$ m^2 s^{-1}) with uniform flow, with velocities as indicated. (F) Echo phases calculated for the same simulations (symbols) with theoretical phases (lines) calculated from Eq. (5). Error bars indicate 68% confidence intervals.

principle, with appropriate optimisations, the performance improvements observed here for the GPU could be reduced up to 16-fold. Nevertheless we believe that the values we report here will reflect real-life improvements for most users.

Simulations were run of diffusion ($D = 10^{-9}$ m^2 s^{-1}) in a randomly oriented $5 \times 200 \times 200$ µm box with a 4 µs time step over 40 ms (10,000 steps) and $2^{20} = 1{,}048{,}576$ particles (unless otherwise varied). For assessment of performance, manual inspection of the code estimated that each iteration of the diffusion kernel comprises 67 floating point operations. Operations such as log and cosine which require multiple operations per invocation have been accounted for as described by [37], while integer operations and binary manipulations (bitwise XOR and shift operators required by the random number generator) have not been included.

Fig. 4A compares the run time for simulations performed on the GPU and on the CPU. A 325-fold acceleration is observed when the

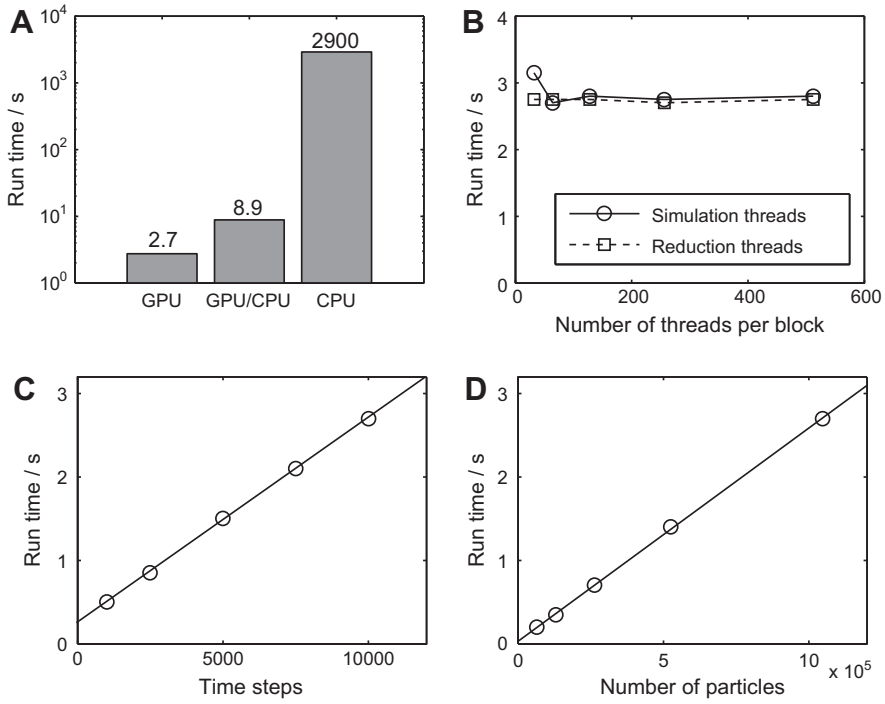

Fig. 4. Performance of the simulation algorithm, based on the time required to simulate 1,048,576 particles over 10,000 time steps. (A) Performance of CPU and GPU compared. GPU/CPU indicates simulations run on the GPU, with phase averages computed on the CPU. (B) Optimisation of the number of threads per block, for both simulation kernels and parallel reduction kernels. (C) Scaling of simulation run time with the number of time steps. (D) Scaling of simulation run time with the number of particles.

simulation kernel is executed on the GPU. At this point, performance is limited instead by the averaging of phases at the end of the simulation; when this is also performed on the GPU by parallel reduction, the final acceleration is by a remarkable three orders of magnitude.

Performance has little dependence on the number of threads per block for either simulation or parallel reduction kernels (Fig. 4B), apart from an increased run time for simulations run with 32 threads per block. At this point, access to gradient profiles stored in global memory is probably limiting, as not enough threads are available for the scheduler to hide the latency of global memory access. Equally, there is no advantage to having more than 64 threads per block, which implies either that peak memory bandwidth has been reached, or that we have fully hidden the memory access latency, such that the processor is always doing useful calculations rather than memory operations. Code profiling revealed that the utilised memory bandwidth was approximately 8 GB/s, which is substantially less than the hardware limit of 140 GB/s, implying that memory access latency is indeed fully hidden by the scheduler.

The run time scales linearly with the number of time steps in the simulation (Fig. 4C) and with the number of particles (Fig. 4D). The small offset in Fig. 4C may be attributed to a combination of initial memory transfers and the final parallel reductions, both of which are independent of the simulation length, but not the number of particles as the zero intercept in Fig. 4D indicates. A calculation time per step of 245 μs may be derived from Fig. 4C, for the simulation of 2^{20} particles. Given the earlier estimate of 67 floating point operations per particle per step, this indicates a net performance of 290 GFLOPS, or 31% of the nominal peak performance – without including integer and logical operations. This is highly satisfactory, and is a strong indicator that the simulation is limited by the GPU performance itself rather than memory bandwidth, and that the implementation of boundary conditions has successfully avoided thread serialisation due to branch divergences.

4. Discussion

In this work, we have demonstrated the implementation of Monte Carlo simulations of PFG experiments optimised for execution on GPUs. The simulation algorithms have been validated against analytical results for free and restricted diffusion, flow, and surface relaxation, and benchmarking results showed a 1000-fold acceleration when compared to identical code run on a CPU.

This large acceleration in part reflects how well suited these ‘embarrassingly parallel’ Monte Carlo simulations are to computation on the GPU, i.e. memory transfers are limited, and no interprocess communication is necessary. Not all simulation algorithms would be expected to show comparable improvements: for example, should it be required to compute multiple specular reflections at boundaries, faster performance could potentially be achieved on CPUs due to their improved ability to handle loops and branches. As noted earlier, careful optimisation of the CPU reference code could in principle improve performance up to 16-fold, but even if this were achieved GPU-based simulations would still have a ca. 20-fold performance advantage. The difference in raw CPU and GPU performance, i.e. GFLOPS, is indisputable, and continues to widen at the present time.

The dramatic increase in speed achieved with GPU-based computation opens up new opportunities in the design and analysis of experiments: taking only a few seconds rather than hours to run, simulations can be run interactively at the desktop. This may assist selection of experimental parameters, such as diffusion delays, pulse lengths and gradient strengths, before experiments are performed. It also becomes practical to couple the simulations to

least-squares or Markov Chain Monte Carlo algorithms, to fit physical parameters such as diffusion coefficients and pore sizes to experimental data directly.

A new approach to accelerating Monte Carlo PFG simulations has recently been described, using a fast random walk algorithm which continually adapts the length scale to the local geometry [44]. In that work, it is noted that the algorithm could be parallelised, and therefore could itself be a candidate for GPU-based computation, resulting in even more significant performance gains. However, the simulation algorithm employed in this work remains attractive due to its simplicity: the implementation is transparent, and the microscopic picture of individual spins could be readily adapted to include additional effects, such as multiple or non-linear gradients and anisotropic diffusion. Such flexibility will be important if the approach here is to assist and accelerate the design, analysis and interpretation of diffusion-weighted MRI data.

Finally, we point out that GPU-accelerated computing remains a new field – the CUDA library was only introduced at the end of 2006 – and that the number of cores, performance, and programmability of GPUs are all increasing rapidly with every new generation of hardware [27]. Of relevance to the practising NMR spectroscopist, density functional theory methods have already been implemented on GPUs [45,46], as have efficient algorithms for the fast Fourier transform [47], but perhaps the computationally intensive algorithms required for the processing of non-linearly sampled data [48–50] may also benefit in the future. As the technology develops further, the order-of-magnitude performance gains, as demonstrated in this work, will surely become too great to ignore.

Acknowledgments

We thank David Fallaize for assistance in setting up the GPU calculations, and John Kirkpatrick for valuable discussions. CAW and JC acknowledge support from the BBSRC (9015651/JC) and from an HFSP Young Investigators Award (RGY67/2007).

References

[1] E. Stejskal, J. Tanner, Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient, J. Chem. Phys. 42 (1965) 288–292.
[2] P. Callaghan, Principles of Nuclear Magnetic Resonance Microscopy, first ed., Oxford University Press, USA, 1994.
[3] D.S. Grebenkov, NMR survey of reflected Brownian motion, Rev. Mod. Phys. 79 (2007) 1077.
[4] P.N. Sen, Time-dependent diffusion coefficient as a probe of geometry, Concepts Magn. Reson. 23A (2004) 1–21.
[5] Y.-Q. Song, S. Ryu, P.N. Sen, Determining multiple length scales in rocks, Nature 406 (2000) 178–181.
[6] J. Pfeuffer, U. Flögel, W. Dreher, D. Leibfritz, Restricted diffusion and exchange of intracellular water: theoretical modelling and diffusion time dependence of 1H NMR measurements on perfused glial cells, NMR Biomed. 11 (1998) 19–31.
[7] E.E. Sigmund, H. Cho, P. Chen, S. Byrnes, Y.-Q. Song, X.E. Guo, et al., Diffusion-based MR methods for bone structure and evolution, Magn. Reson. Med. 59 (2008) 28–39.
[8] P. Stevenson, A.J. Sederman, M.D. Mantle, X. Li, L.F. Gladden, Measurement of bubble size distribution in a gas–liquid foam using pulsed-field gradient nuclear magnetic resonance, J. Colloid Interf. Sci. 352 (2010) 114–120.
[9] P.P. Mitra, P.N. Sen, Effects of microgeometry and surface relaxation on NMR pulsed-field-gradient experiments: simple pore geometries, Phys. Rev. B 45 (1992) 143.
[10] D. Le Bihan, Looking into the functional architecture of the brain with diffusion MRI, Nat. Rev. Neurosci. 4 (2003) 469–480.
[11] R. Bammer, S.J. Holdsworth, W.B. Veldhuis, S.T. Skare, New methods in diffusion-weighted and diffusion tensor imaging, Magn. Reson. Imaging Clin. N. Am. 17 (2009) 175–204.
[12] J.A. McNab, K.L. Miller, Steady-state diffusion-weighted imaging: theory, acquisition and analysis, NMR Biomed. 23 (2010) 781–793.
[13] J.C. Gore, J. Xu, D.C. Colvin, T.E. Yankeelov, E.C. Parsons, M.D. Does, Characterization of tissue structure at varying length scales using temporal diffusion spectroscopy, NMR Biomed. 23 (2010) 745–756.
[14] J. Kärger, W. Heink, The propagator representation of molecular transport in microporous crystallites, J. Magn. Reson. (1969) 51 (1983) 1–7.
[15] P. Linse, O. Soderman, The Validity of the short-gradient-pulse approximation in NMR studies of restricted diffusion. simulations of molecules diffusing between planes, in cylinders and spheres, J. Magn. Reson., Ser. A 116 (1995) 77–86.
[16] W.S. Price, P. Stilbs, O. Söderman, Determination of pore space shape and size in porous systems using NMR diffusometry. Beyond the short gradient pulse approximation, J. Magn. Reson. 160 (2003) 139–143.
[17] C. Malmborg, D. Topgaard, O. Söderman, NMR diffusometry and the short gradient pulse limit approximation, J. Magn. Reson. 169 (2004) 85–91.
[18] M.D. Hurlimann, K.G. Helmer, T.M. Deswiet, P.N. Sen, Spin echoes in a constant gradient and in the presence of simple restriction, J. Magn. Reson., Ser. A 113 (1995) 260–264.
[19] J. Stepisnik, Validity limits of Gaussian approximation in cumulant expansion for diffusion attenuation of spin echo, Physica B 270 (1999) 110–117.
[20] D.S. Grebenkov, Laplacian eigenfunctions in NMR. I. A numerical tool, Concepts Magn. Reson. Part A 32A (2008) 277–301.
[21] A. Caprihan, L.Z. Wang, E. Fukushima, A multiple-narrow-pulse approximation for restricted diffusion in a time-varying field gradient, J. Magn. Reson., Ser. A 118 (1996) 94–102.
[22] P.T. Callaghan, A simple matrix formalism for spin echo analysis of restricted diffusion under generalized gradient waveforms, J. Magn. Reson. 129 (1997) 74–84.
[23] D.S. Grebenkov, Laplacian eigenfunctions in NMR. II. Theoretical advances, Concepts Magn. Reson. Part A 34A (2009) 264–296.
[24] B. Balinov, B. Jonsson, P. Linse, O. Soderman, The NMR self-diffusion method applied to restricted diffusion. simulation of echo attenuation from molecules in spheres and between planes, J. Magn. Reson., Ser. A 104 (1993) 17–25.
[25] A. Duh, A. Mohorič, J. Stepišnik, Computer simulation of the spin-echo spatial distribution in the case of restricted self-diffusion, J. Magn. Reson. 148 (2001) 257–266.
[26] R.M.E. Valckenborg, H.P. Huinink, J.J. vd Sande, K. Kopinga, Random-walk simulations of NMR dephasing effects due to uniform magnetic-field gradients in a pore, Phys. Rev. E 65 (2002) 021306.
[27] J.D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A.E. Lefohn, et al., A survey of general-purpose computation on graphics hardware, Comput. Graphics Forum 26 (2007) 80–113.
[28] NVIDIA CUDA webpage, <http://www.nvidia.com/object/cuda_home.html>, 2011.
[29] OpenCL webpage, <http://www.khronos.org/opencl/>, 2011.
[30] A.F. Ghoniem, F.S. Sherman, Grid-free simulation of diffusion using random walk methods, J. Comput. Phys. 61 (1985) 1–37.
[31] G. Drazer, J. Koplik, Tracer dispersion in two-dimensional rough fractures, Phys. Rev. E Stat. Nonlin. Soft Matter Phys. 63 (2001) 056104.
[32] P. Kurowski, I. Ippolito, J.P. Hulin, J. Koplik, E.J. Hinch, Anomalous dispersion in a dipole flow geometry, Phys. Fluids 6 (1994) 108.
[33] P. Szymczak, A.J.C. Ladd, Boundary conditions for stochastic solutions of the convection-diffusion equation, Phys. Rev. E 68 (2003) 036704.
[34] NVIDIA, CUDA C Programming Guide v3.2, 2010.
[35] D.J. Bergman, K.-J. Dunn, L.M. Schwartz, P.P. Mitra, Self-diffusion in a periodic porous medium: a comparison of different approaches, Phys. Rev. E 51 (1995) 3393.
[36] G. Marsaglia, Xorshift RNGs, J. Stat. Softw. 8 (2003) 1–6.
[37] M. Januszewski, M. Kostur, Accelerating numerical solution of stochastic differential equations with CUDA, Comput. Phys. Commun. 181 (2010) 183–188.
[38] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes, third ed. The Art of Scientific Computing, Cambridge University Press, 2007.
[39] W. Kahan, Further remarks on reducing truncation errors, Commun. ACM 8 (1965) 40.
[40] W.D. Hillis, J. Steele, Data parallel algorithms, Commun. ACM 29 (1986) 1170–1183.
[41] G. Blelloch, Vector Models for Data-Parallel Computing, MIT Press, Cambridge, MA, 1990.
[42] J. Mattiello, P.J. Basser, D. Lebihan, Analytical expressions for the b matrix in NMR diffusion imaging and spectroscopy, J. Magn. Reson., Ser. A 108 (1994) 131–141.
[43] N. Hedin, T.Y. Yu, I. Furó, Growth of C12E8 micelles with increasing temperature. a convection-compensated pgse nmr study, Langmuir 16 (2000) 7548–7550.
[44] D.S. Grebenkov, A fast random walk algorithm for computing the pulsed-gradient spin-echo signal in multiscale porous media, J. Magn. Reson. (2010).
[45] I.S. Ufimtsev, T.J. Martinez, Graphical processing units for quantum chemistry, Computing Sci. Eng. 10 (2008) 26–34.
[46] K. Yasuda, Accelerating density functional calculations with graphics processing unit, J. Chem. Theory Comput. 4 (2008) 1230–1236.
[47] K. Moreland, E. Angel, The FFT on a GPU, SIGGRAPH/Eurographics Workshop on Graphics Hardware 2003 Proceedings (2003) 112–119.
[48] K. Kazimierczuk, J. Stanek, A. Zawadzka-Kazimierczuk, W. Kozminski, Random sampling in multidimensional NMR spectroscopy, Prog. Nucl. Magn. Reson. Spectrosc. 57 (2010) 420–434.
[49] V. Jaravine, I. Ibraghimov, V.Y. Orekhov, Removal of a time barrier for high-resolution multidimensional NMR spectroscopy, Nat. Methods 3 (2006) 605–607.
[50] V.A. Jaravine, A.V. Zhuravleva, P. Permi, I. Ibraghimov, V.Y. Orekhov, Hyperdimensional NMR spectroscopy with nonlinear sampling, J. Am. Chem. Soc. 130 (2008) 3927–3936.