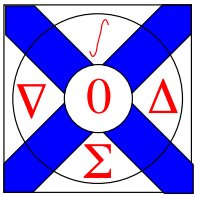

Computational Science Technical Note **CSTN-070**

Numerical Simulation of the Complex Ginzburg-Landau Equation on GPUs with CUDA

K. A. Hawick and D. P. Playne

2011

The Time Dependent Ginzburg Landau(TDGL) equation models a complex scalar field and is used to study a variety of different physical systems and exhibits phase transitional behaviours that necessitate study using numerical simulation methods. We employ fast data-parallel simulation algorithms on Graphical Processing Units (GPUs) and report on performance data and stability tradeoffs from using various implementations of both 32-bit and 64-bit complex numbers. Using NVIDIA's Compute Unified Device Architecture (CUDA) programming language running on a GTX480 GPU, we are able to simulate the TDGL with relatively large simulation system sizes of 256^3 cells and we discuss the relative computational tradeoffs between numerical accuracy and stability using different methods as well as different data precisions.

Keywords: Ginzburg-Landau equation; PDE; complex numbers; numerical precision; CUDA, GPUs

BiBTeX reference:

```
@INPROCEEDINGS{CSTN-070,
        author = {K. A. Hawick and D. P. Playne},
        title = {{Numerical Simulation of the Complex Ginzburg-Landau Equation on
                GPUs with CUDA}},
        booktitle = {Proc. IASTED International Conference on Parallel and Distributed
                Computing and Networks (PDCN)},
        year = {2011},
        number = {CSTN-070},
        pages = {39-45},
        address = {Innsbruck, Austria},
        month = {15-17 February},
        publisher = {IASTED},
        doi = {10.2316/P.2011.719-036},
        institution = {Massey University},
        keywords = {Ginzburg-Landau equation; PDE; complex numbers; numerical precision;
                CUDA, GPUs},
        owner = {kahawick},
        timestamp = {2009.02.28}
}
```

This is a early preprint of a Technical Note that may have been published elsewhere. Please cite using the information provided. Comments or quries to:

Prof Ken Hawick, Computer Science, Massey University, Albany, North Shore 102-904, Auckland, New Zealand.
Complete List available at: http://www.massey.ac.nz/~kahawick/cstn

Technical Report CSTN-070

Numerical Simulation of the Complex Ginzburg-Landau Equation on GPUs with CUDA

K.A. Hawick and D.P. Playne
Computer Science, Institute for Information and Mathematical Sciences,
Massey University, North Shore 102-904, Auckland, New Zealand
email: {k.a.hawick, d.p.playne}@massey.ac.nz
Tel: +64 9 414 0800 Fax: +64 9 441 8181

September 2010

ABSTRACT
The Time Dependent Ginzburg Landau(TDGL) equation models a complex scalar field and is used to study a variety of different physical systems and exhibits phase transitional behaviours that necessitate study using numerical simulation methods. We employ fast data-parallel simulation algorithms on Graphical Processing Units (GPUs) and report on performance data and stability tradeoffs from using various implementations of both 32-bit and 64-bit complex numbers. Using NVIDIA's Compute Unified Device Architecture (CUDA) programming language running on a GTX480 GPU, we are able to simulate the TDGL with relatively large simulation system sizes of 256^3 cells and we discuss the relative computational tradeoffs between numerical accuracy and stability using different methods as well as different data precisions.

KEY WORDS
Ginzburg-Landau equation; PDE; complex numbers; numerical precision; CUDA, GPUs.

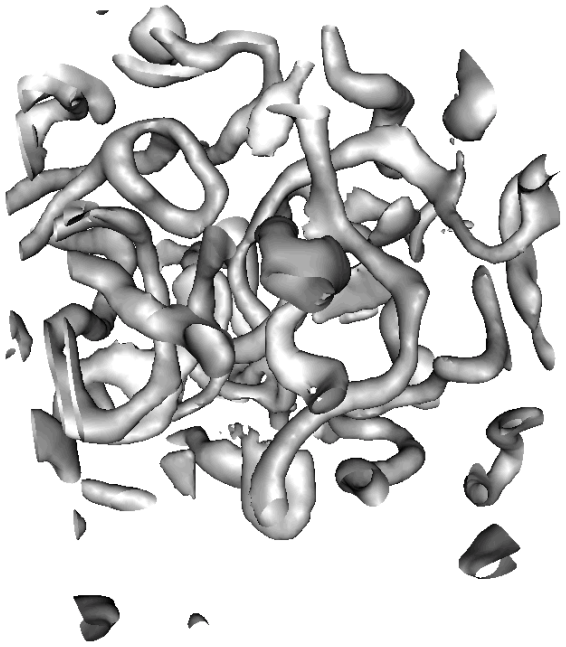

Figure 1: A iso-surface map of the magnitude of a simulated Ginzburg-Landau Field showing helical vortices and other spatial structure in the order parameter.

1 Introduction

Graphical Processing Units (GPUs) have attracted considerable interest in the parallel computing literature recently because of their capacity for high-performance data-parallel computations at commodity pricing. One area that has not developed as fast as might have been hoped however is floating point performance, and especially 64-bit precision capabilities. This is entirely understandable as originally of course GPUs were aimed at the acceleration of graphical processing and the calculations necessary for the rendering of interactive computer games. As these devices are used more and more for accelerating scientific calculations however, it is worthwhile investigating the performance and accuracy tradeoffs.

Many scientific calculations, including simulations of partial differential equations (PDEs) rely heavily on floating point calculations and often on double rather than just single precision. The work we discuss in this present paper considers the Time Dependent Ginzburg-Landau (TDGL) equation solved using finite-difference methods on a regular rectilinear mesh. This PDE uses a complex field variable ψ and can therefore also be formulated as a set of two coupled equations each with a wholly real field variable.

We explore the numerical stability limitations of a finite difference solver for the TDGL equation on GPUs using: the built in compiled language capability for pairs of 32-bit floats to make a `complex<float>`; similar built-in pairs of 64 bit double to use `complex<double>`; and finally our own hand crafted wholly real pairs of coupled equations using 32-bit and 64-bit fields.

Modern GPUs such as the GTX 480 from NVIDIA now have available on-board floating point and even double precision floating point units and we do indeed obtain excellent speed ups over that attainable on a single core of modern CPU. However as will be seen the tradeoff space is complicated and seems likely to remain so for at least the next few CPU and GPU hardware generations.

Complex arithmetic has its own complications and subtleties and quite often system supplied complex arithmetic software implementations ("`complex.h`" packages for C/C++ compilers for example) only provide naive high-school level definitions of some fundamental complex arithmetic operations. This is important for scientific simulations.

The Time Dependent Ginzburg Landau (TDGL) equation solved numerically in 3-dimensions gives rise to some very interesting phase transitional behaviour and to a set of complex spatial structures. Figure 1 shows a snapshot configuration rendered using surface reconstruction techniques. The helical vortices and other spatial structures are clearly seen and accurate sampling of the statistical behaviour of such systems requires many independently initialised numerical experiments and hence a fast solver platform.

Our present paper is organised as: a brief derivation and motivational description of the TDGL in Section 2; a description of the numerical methods we employ in Section 3; details of the parallelisation techniques used with GPUs in Section 5; and a set of performance results including a speed up analysis is given in Section 6. We discuss the implications of these ideas and results in Section 7.

2 TDGL Equation Derivation

Ginzburg-Landau theory [1] is an important theoretical framework for describing the thermodynamic and macroscopic properties of a superconductor without recourse to simulation of microscopic details such as individual atoms, spins, or Cooper pairs [2]. The theory and its uses in describing type II superconductors [3] was developed considerably by Abrikosov [4] and a historical account of the ideas and development was given in the Nobel Lectures of Ginzburg and Abrikosov in 2003 [5, 6]. The Time-Dependent Ginzburg Landau(TDGL) equation and a variation of it known as the Complex Ginzburg Landau equation (CGLE) [7] has been applied and used in a number of applications by various authors since. Ginzburg-Landau theory also arises as the scaling limit for the XY model and also displays important theoretical similarities with the Higgs mechanisms [8] in particle systems [9] which is obviously a topical area as the search continues at the Large Hadron Collider for evidence of the Higgs boson. The most important application of the theory however is for explaining the behaviour of super-conduction and super-fluidity. Important work has been done by many researchers in this area including: work on resonant tunneling effects in CuO2 layers in high temperature superconductors by Abrikosov [10]; cross-over effects in superconductivity [11]; the Hall effect in superconductors [12]; thin film solutions [13]; diffusion induced turbulence [14]; effects from hollow spheres [15]; Lyupanov functions and stability limits [16] and soliton effects and non-linearities [17] in the Ginzburg-Landau equations.

The Ginzburg-Landau formulation is based on the concept of a free energy functional for the superconductor that can be written in terms of a complex order parameter, given here as ψ – and Ginzburg modestly referred to the theory as the "ψ-Theory". This order parameter gives a measure of how deeply the system is into its superconducting phase. As discussed in the references given above, the free energy function has the typical form:

$$F = F_n + \alpha|\psi|^2 + \frac{\beta}{2}|\psi|^4 + \frac{1}{2m}\left|(-i\hbar\nabla - 2e\mathbf{A})\,\psi\right|^2 + \frac{|\mathbf{B}|^2}{2\mu_0} \quad (1)$$

where F_n is the free energy in the normal phase, α and β are phenomenological parameters, m is an effective mass, $\mathbf{A}$ is the electromagnetic vector potential, and $\mathbf{B}$ ($\equiv$ rot$\mathbf{A}$, or $\nabla \times \mathbf{A}$) is the magnetic induction. Minimizing the free energy with respect to fluctuations in the order parameter and the vector potential give rise to the Ginzburg-Landau equations:

$$\alpha\psi + \beta|\psi|^2\psi + \frac{1}{2m}\left(-i\hbar\nabla - 2e\mathbf{A}\right)^2\psi = 0 \quad (2)$$

$$\mathbf{j} = \frac{2e}{m}Re\left\{\psi^*\left(-i\hbar\nabla - 2e\mathbf{A}\right)\psi\right\} \quad (3)$$

where j is the electrical current density and Re the real part. Equation 2 specifies the order parameter ψ in terms of the applied magnetic field. This equation has a similar structure to the time-independent Schroedinger equation. Equation 3 then determines the superconducting current. We employ the usual notation so that e is the electronic charge, and μ_0 the free space permittivity.

These equations suggest that there are two characteristic (physical) lengths that occur in a superconductor. These are usually known as the coherence length ξ:

$$\xi = \sqrt{\frac{\hbar^2}{2m|\alpha|}} \quad (4)$$

which characterises thermodynamic fluctuation sizes that occur in the superconducting phase, and secondly, the penetration depth λ is given by:

$$\lambda = \sqrt{\frac{m}{4\mu_0 e^2 \psi_0^2}} \tag{5}$$

where ψ_0 is the equilibrium value of the order parameter ψ when there is no electromagnetic field. The penetration depth then characterises the physical depth to which the superconductor is penetrable by an externally applied magnetic field.

The ratio $\kappa = \lambda/\xi$ is usually known as **the** Ginzburg-Landau Parameter. Type I superconductors are defined to be those for which $\kappa < 1/\sqrt{(2)}$, and Type II superconductors are those for which $\kappa > 1/\sqrt{(2)}$. In Type I superconductors there is a first order phase transition from the normal state to the superconducting state, and for Type II the transition is second order. This is consistent with Ginzburg-Landau theory. In 1957, Abrikosov [4] showed that for the case of a Type II superconductor, a (high) magnetic field will penetrate the system in quantized flux tubes which have important spatial patterns. It is possible to visualise both the order parameter - the ψ field directly as well as looking at current flow patterns.

For our purposes in this present paper we put aside an analysis of the current properties directly and focus on solving the order equation 2 numerically. A simple form of the Ginzburg Landau partial differential equation is:

$$i\frac{\partial \psi}{\partial t} + p\frac{\partial^2 \psi}{\partial x^2} + q|\psi|^2\psi = i\gamma\psi \tag{6}$$

where the constants $p, q \in \mathbb{C}$ and $\gamma \in \mathbb{R}$.

We have experimented numerically with parameters p, q on the unit square in complex space, and $\gamma \in [-1, 1]$ and with various initialisations of the complex field u with random uniform values - or at least uniform amplitudes so that $u_x(t = 0) = \alpha + i\beta$ with α and β chosen st $\sqrt{(\alpha^2 + \beta^2)} < 1$ and is uniform and the phase angle $\phi = atan(\alpha/\beta)$ is uniform on $[0, 2\pi]$.

Rearranging so that the time derivative stands alone on the left hand side, we obtain:

$$\frac{\partial \psi}{\partial t} = -\frac{p}{i}\frac{\partial^2 \psi}{\partial x^2} - \frac{q}{i}|\psi|^2\psi + \gamma\psi \tag{7}$$

which is similar form to previous works on solving rectilinear field equations with finite difference methods – except that the field ψ is a field of complex numbers. We have written the PDE derivation in terms of a single spatial variable x but the $\frac{d^2}{dx^2}$ extends to the Laplacian operator ∇^2 in arbitrary dimensions. We are able to construct complex arithmetic expressions for equation 7 using second order spatial operators and second or higher order time integration methods.

Valuable work has been done verifying the existence and location of global and unique solutions to the TDGL [18]. In addition to cell dynamics and growth properties [19] – also displayed by related equations such as the Cahn-Hilliard model [20] – simulations of the TDGL also exhibit vortical structure and vortex dynamics [9, 21] There is continued interest in the spatial patterns, turbulent effects and the chaotic and complex phase-field [22] structure of the TDGL equation and this work requires numerical methods and has been extensively studied in lower dimensions (1-dimension and 2-dimensions) [23, 24] Work in higher dimensions – that is in realistic 3-dimensional systems is more difficult [25] and more computationally expensive. A number of researchers report work on different numerical techniques including spectral methods [26] and finite-element methods [27] and other techniques [28]. Most work reported appears to use finite difference methods [29,30] – perhaps due to the numerical difficulties presented by the use of complex numbers, the need for simple and verifiable solver codes and the need for good computational performance. We discuss the numerical issues further in Section 3.

3 Numerical Methods

To simulate this equation numerically, we approximate continuous space of the field with hyper-cubic mesh of discrete cells. Each cell represents a length (1D), area (2D) or volume (3D) of the field. To calculate the interactions between neighbouring cells, the continuous spatial calculus operator $\frac{\partial^2}{\partial x^2}$ is replaced with a discrete approximation. This discrete operator can be formulated for any number of dimensions (like the mesh) and we show its form for one-, two- and three-dimensions in Figure 2.

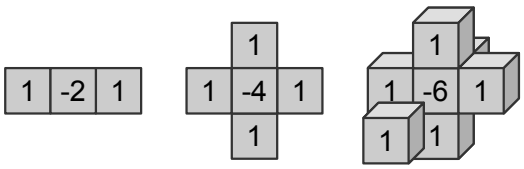

Figure 2: Laplacian operator in one-, two- and three-dimensions.

Using this approximation we can formulate a discrete equation to calculate the potential function of the Ginzburg-Landau equation. This allows us to simulate the equation on a CPU or GPU. Two possible ways to implement this calculation on the GPU are discussed in Section 5. The next step is to determine a suitable numerical integration method.

To numerically simulate the Ginzburg-Landau equation we must use a suitably stable integration method. In previous

work with the Cahn-Hilliard equation [31] we have found the Runge-Kutta 2^{nd} (Equation 8) order method to be sufficient for this sort of simulation [32].

$$y_{n+1} = y_n + hf(t_n + \frac{h}{2}, y_n + \frac{h}{2}f(t_n, y_n)) \quad (8)$$

However, with the additional complexity of the Ginzburg-Landau equation this may no longer but sufficient. We have compared the Runge-Kutta 2^{nd} and Runge-Kutta 4^{th} (Equation 9) order methods to determine whether the RK2 method is suitable, we have found the RK2 error rate to be approximately 3.7×10^{-11} per hit.

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1) \\ k_3 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2) \\ k_4 &= f(t_n + h, y_n + hk_3) \\ y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \quad (9)$$

Another consideration when computing the simulation is the accuracy of the floating point calculations. While GPUs can now process double precision floating point values, they are still more suited to using single-precision values. To determine whether single-precision floating point values are sufficient for this simulation, we have compared single-precision and double-precision RK4 implementations. We have found the error to be approximately 3.0×10^{-16} per hit. As we are measuring the statistical properties of the model as opposed to finding an exact solution this is an acceptable error rate.

4 GPU Architecture

The development of Graphical Processing Units (GPUs) has been driven largely by the games industry. To keep up with the increasing demands, these GPUs have turned into highly parallel processing architectures. With the development of GPGPU technology, these architectures have been adopted as a very popular and powerful processing platform. In sight of this, the latest generation NVIDIA Fermi GPUs have been designed with this application in mind.

The Fermi architecture GPUs contain 11-15 multiprocessors which each contain 32 scalar processors [33]. All scalar processors on a multiprocessor execute the same instruction while separate multiprocessors are free to execute instructions independently. The architecture of these GPUs can be seen in Figure 3.

To utilise the GPU, the computational problem must be split into thousands or millions of threads. The GPU hardware is capable of managing, scheduling and executing these threads (also known as kernels). These kernels are managed in groups known as blocks. While the programmer has no direct control over how the kernels are executed, they can control the size of each thread block and thus control which threads are executed together. Each block can have a maximum size of 1024 threads for a Fermi architecture GPU.

One of the major performance considerations when designing a GPU program is how to access memory. As GPUs are designed to have a high computational throughput, they do not contain the same levels of memory cache hierarchy commonly seen in CPUs. The main area of memory on the GPU is called **global** memory, this is the only type of memory that can be accessed by the CPU host. All data being passed to the GPU program or returned to the host must be stored in **global** memory. One major advantage of the Fermi architecture GPUs is that access to the **global** memory is cached by L2 cache (shared between all multiprocessors) and L1 cache (individual to each multiprocessor). In this work we use cached **global** memory access for all data access and we have previously found it is the optimal memory type for this access pattern [34].

5 CUDA Implementations

Most discussions of GPU finite-differencing implementations focus on the performance benefits of various memory configurations [32, 34]. However, in this work we wish to focus more on the possible ways to implement complex number calculations on the GPU. Thus in all simulations we use **global** memory for accessing the field. On the Fermi architecture GPUs, we have found that cached global memory access provides the best performance for this type of memory access patterns [34].

The challenge we address in this work is how complex number calculations can best be implemented on the GPU. With CUDA we have the option to implement our own complex number class and overload the operators as device functions which allows us to write the Ginzburg-Landau equation in its simple form. The other option we have explored is to store the complex field as two separate arrays (representing the real and imaginary values) and perform the calculations of the real and imaginary values manually.

5.1 Method A - Complex Data Type

Method A processes and stores the field values as complex numbers. The field is of type complex and the calculations are done using complex number arithmetic. To allow for this we have created our own **complex** number class and overloaded the operators required by the Ginzburg-Landau equation. As the computation is performed on the GPU,

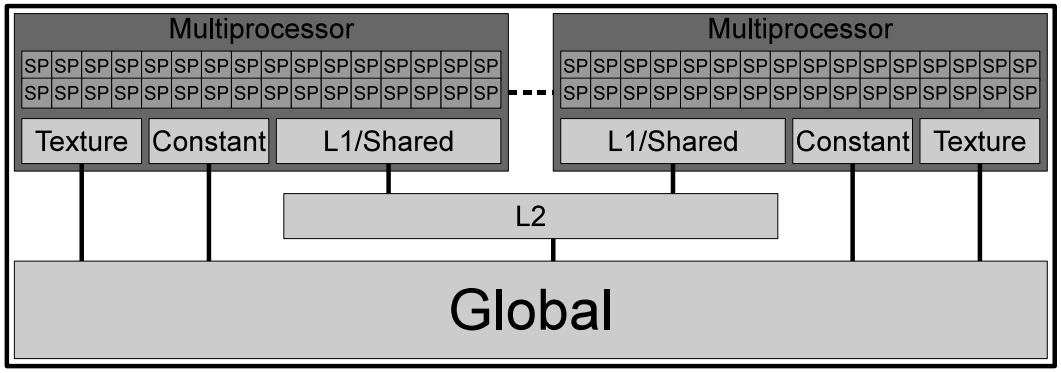

Figure 3: The architecture of NVIDIA Fermi GPUs.

we cannot use the normal operators. Instead we use **device** functions to overload the complex class operators so they can be called from a CUDA kernel. A partial implementation of this class is shown in Listing 1.

Listing 1: Partial implementation of a complex number class with device overloaded operators.

```
class complex {
  public:
    float re;
    float im;
};

__device__ complex operator+(const complex &a,
                             const complex &b){
    return complex(a.re + b.re, a.im + b.im);
}

__device__ complex operator-(const complex &a,
                             const complex &b){
    return complex(a.re - b.re, a.im - b.im);
}
```

The major advantage of implementing this class is that it allows us to write the equation in it's standard mathematical form (as shown in Equation 7). It also allows memory to be allocated and read as complex numbers. This makes the code easy to read and understand, however it does require the use of a complex class. There is another way in which this simulation can be computed.

5.2 Method B - No Complex Type

This method does not use any complex number type, instead it stores and computes the real and imaginary parts separately. The complex arithmetic is thus composed manually by the programmer. The equation can be split into two separate calculations of the real and imaginary parts. We use the notation r and i to denote the real and imaginary fields and $_r$ and $_i$ to denote the real and imaginary parts of parameters. These two separate equations are as follows:

$$\begin{aligned} \frac{\partial r}{\partial t} &= -p_i \nabla^2 r - p_r \nabla^2 i \\ &+ -q_i(r^3 - i^3) - q_r(ri^2 + r^2 i) \\ &+ yr \end{aligned} \tag{10}$$

$$\begin{aligned} \frac{\partial i}{\partial t} &= p_r \nabla^2 r - p_i \nabla^2 i \\ &+ q_r(r^3 - i^3) - q_i(ri^2 + r^2 i) \\ &+ yi \end{aligned} \tag{11}$$

Correctly converting these equations is a significantly more complicated and error-prone process than for the equation i complex number form. Values must be loaded from both the real and imaginary fields, the Laplacian operator must be applied to both of these fields and the final code is significantly harder to read and debug (especially when using the Runge-Kutta 4^{th} order method). However, there are substantial performance benefits to this method which are presented in Section 6.

6 Performance Results

To compare the performance of the two CUDA implementations of the Ginzburg-Landau equation, we have measure the performance across a number of field lengths $N = 32^3 ... 256^3$. We present data for both methods using both the Runge-Kutta 4^{th} and 2^{nd} order integration methods. These simulations have been computed on a NVIDIA GTX480 GPU hosted on a 3.33 Ghz Intel Core i7 Extreme 980X with 12GB of DDR3 - 2000 RAM running Ubuntu 9.10 and CUDA 3.1.

The results for the two CUDA implementations are shown in Figure 4. Both these methods use **global** memory to

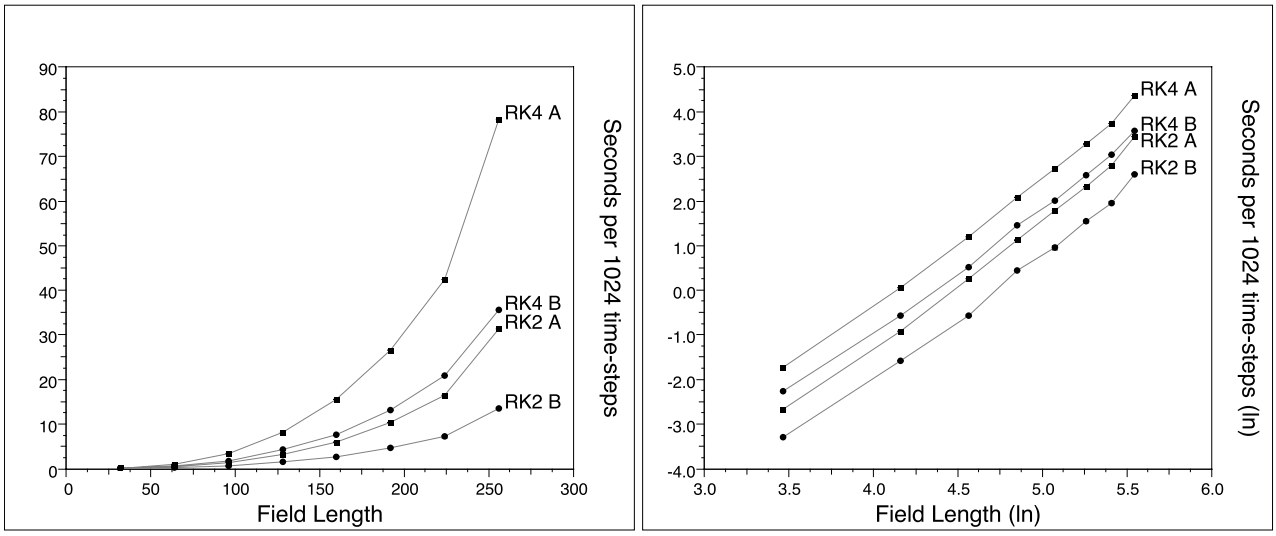

Figure 4: A comparison of the CUDA implementations A&B across field lengths $N = \{32^3...256^3\}$. Note that the error in the timing is not shown as the error bars are smaller than the symbol size of the plots. Results are shown in normal scale (left) and ln-ln scale (right).

store the fields (and intermediate steps) as it provides the best access times with the Fermi cache structure. As expected the RK2 implementations perform the fastest as there are less steps and memory transactions required but this method does not provide the same level of accuracy as RK4.

Method B shows significant performance benefits over Method A with both the RK2 and RK4 implementations computing the simulation approximately twice as fast as their counterparts. This result shows that while computing the real and imaginary parts of the equation separately does involve considerably more effort, if intensive simulations are required, the performance gains can be worth the additional programmer effort.

7 Conclusion

We have shown how the Time-Dependent Ginzburg-Landau equation can be simulated numerically using finite-difference methods on 3-dimensional regular meshes of relatively large system sizes of up to 256^3 cells. We have discussed how complex number calculations can be implemented on Graphical Processing Units. We have discussed accuracy/performance tradeoffs for Runge-Kutta 2^{nd} and 4^{th} order integration methods and single- and double-precision floating point calculations.

We have shown how complex number calculations can be implemented on the GPU with either a complex number class and overloaded **device** operators or by computing the real and imaginary calculations separately. We have presented data for both of these implementations on the latest NVIDIA Fermi architecture GPUs and have shown that computing the real and imaginary parts separately is more complicated but provides approximately twice the computational performance.

The TDGL order parameter field exhibits regions of high curvature around the helical structures seen in Figure 1. These are likely the main reason that the fourth order time-integration method is preferable, in comparison to the second order that is adequate for the (spatially smoother) Cahn-Hilliard equation.

The computational power of even a single GPU device makes semi-interactive simulation and visualisation of the TDGL computationally feasible and therefore will allow a more systematic exploration of its rich spatial structure and various computational experiments. We have been able to simulate systems sizes that are adequate for interactive simulations of the TDGL but we are also exploring how multiple GPU devices can be used together to support the simulation of even larger system sizes for more detailed statistical investigations in "batch mode."

References

[1] Ginzburg, V.L., Landau, L.D.: (Published in English in Collected papers of L.D.Landau, Oxford Press, 1965, pp138-167). Zh. Eksp. Teor. Fiz. **20** (1950) 1064 Edited I.D. ter Haar.

[2] Bardeen, J., Cooper, L.N., Schrieffer, J.R.: Theory of superconductivity. Phys. Rev. **108** (1957) 1175–1204

[3] van Saarloos, W.: The Complex Ginzburg-Landau Equation for Beginners. In Cladis, P., Palffy-Muhoray, P., eds.: Spatiotemporal Patterns in Nonequilibrium Systems. Addison-Wesley (1994)

[4] Abrikosov, A.A.: On the magnetic properties of superconductors of the second group. Sov. Phys. JETP **5** (1957) 1174

[5] Ginzburg, V.L.: On superconductivity and superfluidity. In Frangsmyr, T., ed.: Les Prix Nobel - The Nobel Prizes 2003. Nobel Foundation (2004) 96–127

[6] Abrikosov, A.A.: Type II Superconductors and the Vortex Lattice. In Frangsmyr, T., ed.: Les Prix Nobel - The Nobel Prizes 2003. Nobel Foundation (2004) 59–67

[7] Aranson, I., Kramer, L.: The world of the complex Ginzburg-Landau equation. Rev. Mod. Phys. **74** (2002) 99–143

[8] Higgs, P.: Broken symmetries, masles particles and gauge fields. Physics Letters **12** (1964) 132–133

[9] Kleinert, H.: Vortex origin of tricritical point in Ginzburg-Landau Theory. Europhys. Lett. **74** (2006) 889

[10] Abrikosov, A.A.: Ginzburg-Landau equations for the extended saddle-point model. Phys. Rev. B **56** (1997) 446–452

[11] de Melo, C.R.S., Randeria, M., Engelbrecht, J.R.: Crossover from BCS to Bose Superconductivity: Transition Temperature and Time-Dependent Ginzburg-Landau Theory. Phys. Rev. Lett. **71** (1993) 3202–3205

[12] Yang, T.J., Dai, M.C.: Time Dependent Ginzberg-Landau Equation for dx2-y2 Wave Superconductors: Hall Effect in the Low-Field Regome. Journal of Low Temperature Physics **117** (1999) 569–573

[13] Machida, M., Kaburaki, H.: Direct Simulation of the Time-Dependent Ginzburg-Landau Equation for Type-II Supercomducting Thin Film: Vortex Dynamics and V-i Characteristics. Phys. Rev. Lett. **71** (1993) 3206–3209

[14] Battogtokh, D., Mikhailov, A.: Controlling turbulence in the Complex Ginzburg-Landau Equation. Physica D **90** (1996) 84–95

[15] Du, Q., Ju, L.: Approximations of a Ginzburg-Landau Model for Superconducting Hollow Spheres based on Spherical Centroidal Voronoi Tessellations. Mathematics of Computation **74** (2004) 1257–1280

[16] Tang, Q., Wang, S.: Time Dependent Ginzburg-Landau Equations of Superconductivity. Physica D **88** (1995) 139–166

[17] Biswas, A.: Temporal 1-Soliton Solution of the Complex Ginzburg-Landau Equation with Power Law Nonlinearity. Progress in Electromagnetics Research **96** (2009) 1–7

[18] Du, Q.: Global Existence and Uniqueness of Solutions of the Time-Dependent Ginzburg-Landau Model for Superconductivity. Applicable Analysis **53** (1994) 1–17

[19] Iwamatsu, M., Nakamura, M.: Cell Dynamics Simulation of KolmogorovJohnsonMehlAvrami Kinetics of Phase Transformation. Jpn. J. Appl. Phys. **44** (2005) 6688–6694

[20] Cahn, J., Hilliard, J.: Free energy of a non-uniform system III. Nucleation in a two point compressible fluid. J.Chem.Phys. **31** (1959) 688–699

[21] Gropp, W.D., Kaper, H.G., Leaf, G.K., Levine, D.M., Palumbo, M., Vinokur, V.M.: Numerical Simulation of Vortex Dynamics in Type-II Superconductors. Journal of Computational Physics **123** (1996) 254–266

[22] Anderson, D.M., McFadden, G.B., Wheeler, A.A.: A Phase-field model with convection: Numerical Simulations. Interfaces for the Twenty-First Century (1999)

[23] Doering, C.R., Gibbon, J.D., Holm, D.D., Nicolaenko, B.: Low-dimensional behaviour in the complex ginzburg-landau equation. Nonlinearity **1** (1988) 279–309

[24] Manneville, P., Chate, H.: Phase Turbulence in the Two-Dimensional Complex Ginzburg-Landau Equation. Physica D **96** (1996) 30–46

[25] Lin, F., Rivire, T.: Complex Ginzburg-Landau equations in high dimensions and codimension two area minimizing currents. Journal of the European Mathematical Society **1** (1999) 237–311 10.1007/s100970050008.

[26] Chen, L.Q., Shen, J.: Applications of semi-implicit Fourier-spectral method to phase field equations. Computer Physics Communications **108** (1998) 147–158

[27] Larsen, T.S., peter Sorensen, M., Pedersen, N.F., Madsen, S.: The Ginzburg-Landau Equation Solved by the Finite Element Method. In: Proc. 2006 Nordic COMSOL Conference. (2006)

[28] Zhang, Y., Bao, W., Du, Q.: Numerical Simulation of Vortex Dynamics in Ginzburg-Landau-Schrodinger Equation. Euro. Jnl. of Applied Mathematics **18** (2007) 607–630

[29] Wang, T., Guo, B.: Analysis of Some Finite Difference Schemes for Two-Dimensional Ginzburg-Landau Equation. Wiley (2010)

[30] Ardelea, A., Pardhanani, A.L., Carey, G.F., Richardson, W.B.: Numerical Discretization and Simulation of Ginzburg Landau Models for Superconductivity. Technical report, Texas Institute for Computational and Applied Mathematics (2008)

[31] Hawick, K.A., Playne, D.P.: Modelling, Simulating and Visualizing the Cahn-Hilliard-Cook Field Equation. International Journal of Computer Aided Engineering and Technology (IJCAET) **2** (2010) 78–93

[32] Leist, A., Playne, D., Hawick, K.: Exploiting Graphical Processing Units for Data-Parallel Scientific Applications. Concurrency and Computation: Practice and Experience **21** (2009) 2400–2437 CSTN-065.

[33] NVIDIA® Corporation: CUDA™ 3.1 Programming Guide. (2010) Last accessed September 2010.

[34] Playne, D.P., Hawick, K.A.: Comparison of GPU Architectures for Asynchronous Communication with Finite-Differencing Applications. Technical Report CSTN-111, Massey University (2010) submitted to: Concurrency and Computation: Practice and Experience.