GPU-accelerated solutions of the nonlinear Schrödinger equation

Benjamin D. Smith, Logan W. Cooke, and Lindsay J. LeBlanc
Department of Physics, University of Alberta, Edmonton AB T6G 2E1, Canada
(Dated: October 29, 2020)

As a first approximation beyond linearity, the nonlinear Schrödinger equation reliably describes a broad class of physical systems. Though numerical solutions of this model are well-established, these methods can be computationally complex, especially when system-specific details are incorporated. In this paper, we demonstrate how numerical computations that exploit the features of a graphics processing unit (GPU) result in 40-70× reduction in the time for solutions (depending on hardware details). As a specific case study, we investigate the Gross-Pitaevskii equation, a specific version of the nonlinear Schrödinger model, as it describes a trapped, interacting two-component Bose-Einstein condensate subject to spatially dependent interspin coupling, resulting in an analog to a spin-Hall system. This computational method allows us to probe high-resolution spatial features – revealing an interaction dependent phase transition – all in a reasonable amount of time with readily available hardware.

I. INTRODUCTION

There are many problems in physics for which the only realistic approach to a solution is through numerical techniques. The nonlinear Schrödinger equation (NLSE) is one such mathematical model with widespread applications throughout physics. Notably, this equation explains superfluid and magnetic properties of dilute Bose-Einstein condensates (BECs) [1, 2], but it also successfully describes plasma Langmuir waves [3], soliton dynamics [4], the propagation of light in nonlinear media [5, 7], surface gravity water waves [8] and rogue waves [9], superconductivity [10], and even certain financial situations [11]. The connecting thread between these disparate physical phenomena is the slowly-varying evolution of a weakly nonlinear, complex wave packet in a dispersive environment [10].

A NLSE can take many forms [12], but many physical phenomena can be adequately described with a cubic nonlinearity. For a complex scalar function $\psi$, the NLSE can be written in the general form:

$$i \frac{\partial \psi}{\partial t} = -a \nabla^2 \psi + b \psi + c |\psi|^2 \psi,$$

where the parameter $a > 0$, and where $c$ (which can be positive or negative) represents the strength of the nonlinearity. It is worthy to note that as $c$ goes to zero, the NLSE reduces to the familiar Schrödinger equation.

While very few analytical solutions of the NLSE exist, the literature is replete with techniques for finding numerical solutions [12–17]. Usually these methods involve direct integration of the equation in space and time. Throughout the last decade, these techniques have been maturing and growing more accessible. This is demonstrated by the availability of open, self-contained solver packages, such as the GPELab toolbox for MATLAB® [18, 19]. Despite the wealth of numerical techniques, the NLSE’s nonlinear term makes it quite computationally intensive to solve [20]. These solutions often demand extremely small mesh spacings to accurately represent small features, such as superfluid BEC vortices [17], thus making grid sizes very large.

A computer’s central processing unit (CPU) computes grid points one at a time as a “serial processor.” These devices are optimized to remove latency between calculations and typically compute sequential operations at a rate of about a GHz. In many cases, however, a serial processor cannot reasonably meet the demands of integrating the NLSE across large grid sizes. This results in very long computational run-times [12], which, in extreme cases, can span days to weeks [21].

For particular operations and algorithms, graphics processing units (GPUs) offer a significant increase in computational power. They accomplish this by parallelizing. There are two notable types of parallel computing: (1) Task-parallelism, akin to vehicles on an assembly line, operates on distinct and independent sets of data concurrently. (2) Data-parallelism operates on all elements of a single data collection at the same time; the Hadamard product, or element-wise matrix multiplication (represented as $A \circ B$), is an example of an operation that is exceedingly data-parallel [22]. With access to anywhere from hundreds to thousands of multi-processors and shared memory, GPUs can leverage both types of parallelism to accelerate computations far beyond the capacity of a CPU. The ratio of corresponding CPU and GPU evaluation times is often known in this context as “speedup.” Researchers using GPUs to solve the NLSE over the past decade have reported speedups of tens [23, 24] to hundreds [9, 25] of times.

In this paper, we introduce a general approach for GPU-accelerating numerical computations of the NLSE. Several problem-specific third-party Python modules for GPU-accelerating the NLSE exist [22, 26], but they are not amenable to highly-specialized research problems. Using NVIDIA graphics hardware and tools from the open-source Python community, we demonstrate $39 \times$ and $68 \times$ speedup of our pseudospinor NLSE code for our particular hardware configurations. This approach required no detailed knowledge of GPU architecture, and demonstrates that a substantial computational speedup is possible using high-level programming tools like those.
found in the Python environment. This is particularly important for numerical calculations that make predictions for or comparisons to experimental results, where rapid calculations allow for timely parameter iterations and the best optimizations.

This paper is organized as follows. Section II gives a basic introduction to NVIDIA GPU operation. Section III introduces a particular example of the NSLE, and includes the theoretical description of the physical configuration and the algorithm for solving it. Section IV describes our original calculations, and outlines how we implemented GPU-acceleration, and Section V shows a performance comparison of our implementation across different CPU and GPU devices. Section VI discusses the physics behind the results from our example calculation: a simulation of a spin-dependent gauge potential that results in quantized vortices in the spin-Hall regime. Finally, Section VII discusses the broader significance or our results, before concluding with Section VIII.

II. GPU OPERATION

In this paper, we give a high-level introduction to GPUs, pointing out essential features and concepts, and leave details of their use in general-purpose computing to other excellent reviews [27, 28].

Graphics cards and GPUs were originally developed to render virtual 3D graphics in real-time, a task which is highly data- and task-parallel in nature [28]. While GPUs were exceptionally good for rendering graphics, they worked with strict fixed-function pipelines. Realizing the utility in general-purpose GPU computing, graphics card manufacturers developed API frameworks to directly program almost all their GPUs’ resources. There are two predominant APIs for programming GPUs: OpenCL (open source, maintained by Khronos Group) and CUDA (proprietaryly developed by NVIDIA Corporation). In this work, we will restrict our discussion to CUDA and NVIDIA hardware. Similar to other frameworks, CUDA is a low-level interface to the GPU and its usage requires a detailed knowledge of the GPU architecture and resources [28]. Alternatively, the Python community has developed accessible Python packages with back-end interfaces to CUDA (PyCUDA), and for fast-Fourier transforms (cuFFT). Though primarily for machine learning, these packages provide a user-friendly platform for GPU acceleration in standard scientific computing.

When working with GPUs, one should be aware of architecture: the layout design and techniques used in implementing the operations, instructions, data types, registers, memory hierarchies, control units, and processors [28] which are the key factors for performance. One metric describing a NVIDIA GPU’s architecture is its compute capability (CC), which describes the CUDA computing features available on the GPU. For example, CC $\geq 6.x$ (Pascal) can perform 64-bit addition operations, whereas CC $\leq 5.x$ (Maxwell) cannot natively do so [30].

When comparing the performance of different GPU devices, it is also important to note that, within a certain architecture, performance scales with processing core numbers, memory, and clock rates. Between architectures, however, this relationship is not so simple because of the vastly different hardware and instruction sets available to each. A device’s performance can be assessed by benchmarking, or systematically measuring and comparing the time in which a device executes a particular algorithm [12, 25].

III. MODEL AND ALGORITHM

To demonstrate GPU acceleration for the solutions of the NLSE, we consider a model that benefits from the GPU’s features: the Gross-Pitaevskii equation (GPE) is a form of the NLSE used to model weakly interacting superfluids in the mean-field regime, and it is especially well-suited to describe a dilute neutral-atom BEC [2]. The GPE is well-studied in this context, and significant work has gone into improving the path to solutions [13, 14, 16, 31, 32], and to reveal a variety of physical phenomena ranging from vortex creation and dynamics [11, 33, 40] to the many-body states of a spinor system [11, 34]. Here, we take the opportunity provided by the GPU to move beyond the standard GPE: we study the physical consequences of spin- and momentum-dependent coupling, and exploit the power of the GPU to render high-resolution solutions that would otherwise be prohibitively time-expensive. With GPU-based calculations, numerical results can be obtained for realistic experimental conditions in a reasonable amount of time, allowing for numerically informed optimizations of experimental procedures.

A. The coupled pseudospinor Gross-Pitaevskii equation

A standard approach to studying trapped neutral-atom BECs uses the GPE, where a single-component order parameter $\psi(r) = \sqrt{\rho(r)}e^{i\phi(r)}$ represents the state of the system, where $\rho(r)$ is the real-space density and $\phi(r)$ is the phase profile. For a trapped gas of atoms, the GPE describes this order parameter as

$$i\hbar \frac{\partial}{\partial t} \psi(r) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(r) + g|\psi(r)|^2 \right] \psi(r),$$

where the first term in the right-hand bracket represents the kinetic energy with atomic mass $m$; the second is the trapping potential energy; and the third term is the interaction energy, where an interaction parameter $g = 4\pi\hbar^2a_{sc}/m$ is characterized by the interactonic scattering length $a_{sc}$.
Moving beyond this single-component model, we next consider the spinor condensate: a two- (or more-) component system where a higher-dimensional order parameter describes the density of atoms in two (or more) spin states. In describing experimental systems with alkali metal atoms, these spin states are pseudospins whose real identities are defined by \( m_F \) levels in the ground state manifold. In this case of a two-spinor, the order parameter takes the form \( \psi \rightarrow \tilde{\Psi} = \{ \Psi\uparrow, \Psi\downarrow \} \). An additional consideration in spinor systems is that the interspin scattering lengths may differ from each other and from the intraspin scattering lengths, and this must be accounted for in the interaction terms in the GPE. For the two-spinor, there are three interaction strengths to consider: \( g_{\uparrow\uparrow}, g_{\downarrow\downarrow}, \) and \( g_{\uparrow\downarrow} = g_{\downarrow\uparrow} \).

Next, we add the possibility for external coupling between the pseudospins, which, when made to be spin-dependent through clever choices of the coupling fields, can result in “artificial gauge fields” \([46, 47]\) that mimic the effects of magnetic fields, electric fields, and/or spin-momentum coupling in these atomic systems. We consider here the case where two lasers with opposite propagation directions (\( \pm \hat{x} \)) effect a two-photon Raman transition that results in a spatially-periodic spin-wave in the BEC along the recoil direction \( \hat{x} \); in this work, we limit the discussion to a spin wave, and thus momentum transfer, along one dimension, but this type of interaction could be extended to additional dimensions. This spatial-periodicity can be gauged away through a unitary transformation which shifts the two bare spin dispersions opposite directions in \( \hat{k}_x \) momentum space. In this rotated picture, the effective kinetic + potential energy Hamiltonian that describes this process for two components is \([1, 48]\)

\[
\hat{H} = \left[ \frac{\hbar^2 k^2}{2m} + V(r) \right] \hat{1} - \frac{\hbar^2 k_L \hat{k}_x}{m} \hat{\sigma}_z + \frac{\hbar \Omega(r)}{2} \hat{\sigma}_x + \frac{\hbar \delta(r)}{2} \hat{\sigma}_y, \tag{3}
\]

where \( k_L \) is the magnitude of the lasers’ wavevector, \( \delta(r) \) is the two-photon Raman detuning, \( \Omega(r) \) is the two-photon Raman coupling strength, and \( \{ \hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z, \hat{1} \} \) are the Pauli and identity matrices in the spinor basis. The characteristic energy scale of this Hamiltonian is \( E_L = \hbar^2 k_L^2 / 2m \), the energy of a single-photon recoil. Experimentally, spatial dependence in the detuning and Raman coupling can readily be achieved with a spatially-dependent magnetic field (via the Zeeman effect) or a spatial light modulator device (via the ac Stark effect), respectively.

Finally, we incorporate the last three terms of Eq. (3) into the GPE pseudospinor Hamiltonian Eq. (2), taking into account the appropriate signs of the detuning and the momentum shift; we interpret the spinor components \( \{ \Psi\uparrow, \Psi\downarrow \} \) as bare spins that have undergone a spin-dependent momentum shift \( \{ | \uparrow, -k_L \rangle, | \downarrow, +k_L \rangle \} \). After converting all quantities to dimensionless ones (denoted by tildes), the equations for each \( \tilde{\Psi}_{\uparrow(\downarrow)} \) component are

\[
-i \frac{\partial}{\partial t} \tilde{\Psi}_{\uparrow(\downarrow)} = \left[ \hat{H}^{(1)}_{\uparrow(\downarrow)} + \hat{H}^{(2)}_{\uparrow(\downarrow)} \right] \tilde{\Psi}_{\uparrow(\downarrow)} + \hat{H}^{(3)} \tilde{\Psi}_{\downarrow(\uparrow)}, \tag{4}
\]

where

\[
\hat{H}^{(1)}_{\uparrow(\downarrow)} = -\frac{1}{2} \hat{L}_z \pm i \hat{k}_L \hat{L}_x, \tag{5}
\]

\[
\hat{H}^{(2)}_{\uparrow(\downarrow)} = V(r) \pm \frac{1}{2} \frac{\hbar}{\Delta \Omega} [\tilde{\Psi}_{\uparrow(\downarrow)}]^2 + \tilde{g}_{\uparrow(\downarrow)} |\tilde{\Psi}_{\downarrow(\uparrow)}|^2, \tag{6}
\]

\[
\hat{H}^{(3)} = \frac{1}{2} \hat{\Omega}(r), \tag{7}
\]

represent the kinetic energy [Eq. (5)], the potential and interaction energies [Eq. (6)], and the Raman coupling energy [Eq. (7)]; the upper (lower) signs in these equations refers to the \( \uparrow (\downarrow) \) components. This pseudospinor GPE describes the emergence of both superfluid effects (e.g. quantized vortices [1]) as well as magnetic structures (e.g. stripes, spin domains [44]).

### B. Algorithm

In this section, we describe the basic elements of our algorithm for integrating solutions to the GPE of Eq. (4). We begin by assuming that the BEC is confined to the \( x - y \) plane by a strong harmonic trapping potential in the transverse direction with frequency \( \omega_z \). Out-of-plane excitations are suppressed, and the dimensionless order

![FIG. 1. A flow diagram of the time-splitting spectral algorithm. Starting from some initial order parameter \( \tilde{\Psi}(r) \), we propagate in time by applying evolution operators \( U_i \) [see Eqs. (1)-(3)] and fast Fourier transforms \( \mathcal{F} \). The operations applied sequentially in the loop represent those for a single propagation time step \( \Delta t \); this loop is repeated \( M \) times, whereupon the final order parameter is available.](image-url)
parameter along this dimension takes on a Gaussian profile with unit norm. If this strong-confinement condition is fulfilled, we can approximate our system as quasi-2D and represent the order parameters and operators on 2D grids.

Our method for integrating solutions to the pseudospinor GPE relies on the well-established time splitting spectral (TSSP) method. Time is discretized so that a single step forward in time is computed by applying the evolution operators corresponding to each term in the Hamiltonian to the previous step’s order parameter. The evolution operators are given by

$$U^{(i)}(\Delta \tau) = \exp(\mathcal{H}^{(i)} \Delta \tau),$$

where $\Delta \tau$ is a unitless time step smaller than any relevant time scales of the system. One of the key features of the TSSP method involves transforming to reciprocal space where $\mathcal{H}^{(1)}$ is diagonal, since $\nabla^2 \rightarrow -k^2$. In this way we perform many FFTs to avoid the more computationally-expensive finite-difference Laplacian in $U^{(1)}$.

Starting from an initial spinor order parameter, the GPE is propagated in time with the evolution operators. This takes place in a loop over $M$ discrete time steps of length $\Delta \tau$. As shown in Figure 1, the $U^{(1/3)}$ operators are applied with the familiar Strang splitting for stability and to reduce errors induced by the various non-commuting $U^{(i)}$ operators. Propagation in real-time yields dynamics of the spinor system, while propagation in imaginary-time ($\Delta \tau \rightarrow i\Delta \tau$) asymptotically approaches ground state solutions.

While the operations are highly data-parallel they must be applied sequentially. This informed our choice for this acceleration technique and hardware, since there are many ways to accelerate algorithms using multiple CPUs or even multiple GPUs. In this case, it is advantageous to maintain data on a single device better suited to performing the required operations, rather than dealing with the additional transfer times between various devices (even if those devices are also well-suited for the tasks). This way, the data does not leave the device until the entire simulation is complete.

IV. IMPLEMENTATIONS

A. CPU-Based

Our original (non-GPU) implementation exclusively employed the NumPy scientific computing library. In this version of our code, the order parameter was represented by a complex-valued 2D NumPy array. The energy terms making up the Hamiltonians $\mathcal{H}^{(1)}$ and $\mathcal{H}^{(2/3)}$ were represented in reciprocal and real space, respectively, on real-valued NumPy arrays. We pre-computed and stored the potential $V(r)$, kinetic $\mathcal{H}^{(1)}$, Raman coupling $\Omega(r)$, and Raman detuning $\delta(r)$ terms since they were constant throughout the propagation loop; the nonlinear mean-field terms $\bar{g}|\Psi|^2$ depended on the densities and therefore were calculated at each time step. From these energy terms, we then computed the corresponding evolution operators [Eq. (3)]. “Applying” an evolution operator amounted to Hadamard multiplication of the complex operator and the order parameter array. Throughout a single time-step loop, four 2D FFTs and $\gtrsim 20$ Hadamard products were performed. Since probability density is not conserved in imaginary-time propagation, we normalized the order parameters to the total atom number at each step.

B. GPU-Acceleration

Our algorithm relies heavily on the FFT and Hadamard product, which both have the potential to be highly data-parallel operations. We approached this problem from both software and hardware sides. At the time we began, we found several established CUDA-compatible Python packages, such as Tensorflow, providing wrappers of the needed cuFFT library. However, a newer package, PyTorch, stood out to us because it has a “native Python” interface and is intentionally designed to have similar, if not identical, syntax to NumPy, implying a short learning curve and minimal changes to our original code. As with other machine learning packages, PyTorch code can execute on either a CPU or a CUDA-enabled GPU; this made it convenient to develop and test our code on a CPU before scaling it up to run on a GPU workstation. We also found that PyTorch made it easy to specify processor devices.

While adapting the syntax was straightforward, PyTorch does not support a native complex data type for their Tensor array objects. Since complex operations are essential to our algorithm, we converted our 2D complex NumPy arrays to PyTorch Tensors, where an extra third dimension holds the real and imaginary parts. This is a commonly-employed workaround for PyTorch users to represent complex numbers, as it is the natural input/output data structure to the PyTorch FFT functions. We can visualize this structure as two stacked $N_x$ by $N_y$ arrays where the top layer is the real part of the order parameter and the bottom layer is the imaginary part. We have two such complex Tensors in a Python list representing the two spinor components. This data structure worked well for the cuFFT PyTorch wrapper functions; however, to treat the complex parts properly, we had to implement new functions extending the native PyTorch Hadamard product, the exponential function, and other trigonometric functions. These functions and data structures by-and-large provided drop-in replacements to those from our previous code.

On the hardware side, we constructed two different computer workstations with NVIDIA graphics cards. Our first workstation contains a GeForce 980 Ti (Maxwell arch., C.C. 5.2), a common commercial gaming graphics card. Our second workstation contains a Titan V (Volta arch., C.C. 7.0). In addition to the two workstations,
TABLE I. Our PyTorch implementation can execute on any of our CUDA-enabled NVIDIA graphics cards (top) or our CPUs (bottom). The corresponding GPU/CPU hardware pairs are installed on a commercial laptop and two custom-built workstations. Key specifications of these devices are given.

| GPUs      | GeForce MX150 | GeForce 980 Ti | TITAN V |
|-----------|---------------|----------------|---------|
| Architecture | Pascal       | Maxwell        | Volta   |
| # CUDA Cores | 6.1          | 5.2            | 7.0     |
| Clock (Boost) [GHz] | 1.47 (1.53) | 1.0 (1.07) | 1.2 (1.45) |
| VRAM Mem. [GB] | 64           | 384            | 3072    |
| Mem. bus width [bits] | 384          | 2816           | 5120    |
| Mem. bus width [bits] | 2.0          | 6.0            | 12.0    |
| Mem. bus width [bits] | 48.06       | 336.6          | 651.3   |

| CPUs      | Intel i5-7200U | AMD FX-6300 | Intel i9-9900K |
|-----------|----------------|-------------|---------------|
| Clock (Boost) [GHz] | 2.5 (3.1)  | 3.5 (4.1)  | 3.7 (5.0)    |
| Assoc. RAM [GB] | 8             | 16          | 32           |

we also had a commercial Acer Aspire laptop with an integrated NVIDIA GeForce MX150 graphics card (Pascal arch, C.C. 6.1). Specifications for these three devices, along with their corresponding CPUs, are summarized in Table I.

V. PERFORMANCE BENCHMARKING

In this section, we show and analyze the benchmark results of our propagation stepping function, compare the performance of our three GPUs and three CPUs, and compare the respective speedup. As mentioned previously, with PyTorch we can readily configure which of our six devices to compute with. Our benchmarks are computed on only one device at a time; we are not performing what is sometimes called “heterogeneous” or distributed computing with multiple processor devices simultaneously. These benchmarks also only compare the performance of our revised PyTorch code.

To make a fair comparison between GPU and CPU performance, we transferred the order parameter and energy grids to the GPU’s memory before running the benchmarks on those devices, thereby avoiding the relatively slow data transfer rate between CPU RAM and GPU memory. We exclusively employ double-precision floats (float64) in our simulations for accuracy, and therefore use this same precision in these initial benchmarks.

We measured the propagation function evaluation times using the Python timeit module. We separately timed (with 100 ns resolution) many different evaluations of our propagation stepping function, and repeated this process on each device for different 2D grid sizes.

We fit a power-law function of the form $f(N) = aN^\nu + c$ to the CPU evaluation time data. We found that the three CPU evaluation times scale very similarly, and on average $\sim N^{1.06(4)}$. With the GPUs, however, we observed two different scaling behaviors. At small grid sizes, the evaluation times are completely independent of grid size; in fact, fitting a straight line to these data revealed that the average slope does not differ significantly from zero [$2(4) \times 10^{-8}$ sec/pts]. This implies that we lose absolutely nothing in runtime with a higher grid resolutions. At large grid sizes, however, the GPU times scale exactly like those of the CPUs, $\sim N^{1.01(2)}$. For each GPU, an abrupt corner appeared at the transition between small and large grid sizes.

In addition to timing the entire propagation step, we also timed the individual FFT and Hadamard functions on our fastest CPU and GPU, for both single- and double-precision floats (Table II). Except for the computational overhead imposed by our data structure, we would expect an even greater speedup of the Hadamard product than what we measured here.

As mentioned above, it is generally difficult to inter-
TABLE II. Time per call for the FFT (forward transform) and Hadamard functions, using a grid size of 1024 × 1024, for single-precision (float32) and double-precision (float64) values.

| Grid size   | τ_{FFT} float32 | τ_{FFT} float64 | τ_{Hadamard} float32 | τ_{Hadamard} float64 |
|-------------|-----------------|-----------------|-----------------------|-----------------------|
| i9 CPU [ms] | 16.8(4)         | 35.1(6)         | 7.5(1)                | 15.2(5)               |
| Titan V GPU [ms] | 0.67(2)         | 0.85(8)         | 0.456(9)              | 0.54(2)               |

VI. SIMULATION EXAMPLE: SPIN HALL SYSTEM

In this section, we demonstrate our accelerated GPU method by simulating the ground states of a spin Hall system. In such a system, the spin-up and spin-down constituents experience effective magnetic fields of equal magnitude but opposite direction. The spin Hall effect has been investigated theoretically [53, 54] and experimentally [55] using Raman-induced spin-orbit coupling in ultracold atoms; the presence of a spatial gradient in the Raman coupling and an effective “electric” force (a role played by gravity) generate transverse spin Hall currents. Other work showed that interspin interactions can greatly alter the properties of spin Hall states [56, 57]. In the example simulations that follow, we investigate the mean-field ground states of a two-component BEC subject to a spatially varying spin-dependent gauge potential. Similar to the proposal given in [55], these states are generated in situ in the absence of any effective electric force and reside in the classical spin Hall regime (ν = N/N_y ≫ 1) [50].

We consider a harmonically confined, two-spinor BEC of 10^4 atoms, where the trapping frequencies ω_x ≫ ω_y ≡ ω⊥. The harmonic oscillator length a_0 = \sqrt{\hbar/mω⊥} and energy E = \hbarω⊥ set the characteristic length and energy scales for our simulations. The intraspin interaction parameters are g_↑ = g_∥ ≡ g.

The single-particle physics of this problem can be analyzed by diagonalizing the Hamiltonian Eq. (3). For weak coupling (ℏΩ < 4E_L), the lower-energy band takes a double-well shape [48], where the two minima of the dispersion relationship sit at ±k_L in the limit Ω → 0, and where the eigenstates in this band vary across k, with |↓⟩ dominating the state near +k_L, and |↑⟩ dominating the state near −k_L (even for nonzero Ω). If the atoms are confined to this lowest energy band, an effective Hamiltonian for the case of one-dimensional Raman coupling applies:

\[ \hat{H}_{eff,x} = \frac{\hbar^2}{2m} (k_x + A_\phi \hat{\sigma}_z)^2, \] (9)

where the magnitude of the artificial gauge potential A_\phi multiplies the Pauli matrix in the dressed-spin basis, and scales as A_\phi = k_L [1 − (ℏΩ/4E_L)^2]^{1/2} for ℏΩ ≤ 4E_L and δ = 0 [48]. Since the dressed spins are nearly equivalent to the bare spins near the minima of the dispersion, a spatially varying Ω(y) results in a y-dependence of A_\phi(y), and a magnetic field for each spin B_{1(z)} that is equal in magnitude, but opposite in direction.

We imposed a spatially-varying Raman coupling of the form

\[ \frac{\hbar\Omega(y)}{E_L} = \sqrt{8y - y^2} \] (10)
to linearize $\mathcal{A}_x(y)$. As shown in Figure 4(b), this created two spin-dependent degenerate wells in $k$-space that moved inward from $k_y = \pm 1 \rightarrow 0$ as $y$ increases, amounting to a spin-dependent Abelian gauge potential. In the simulations shown in Figure 4, we chose a Raman coupling profile that yielded a uniform synthetic spin-dependent magnetic field with a magnitude of $|B_{\tau(t)}|^2 = 0.369a_0^{-2}$ in the region of the BEC. Before each simulation trial, we randomly seeded each order-parameter component with 50 uniformly distributed vortices; the seeded vortices had opposite windings for each component since we expected the components to acquire opposite angular momentum. To quickly converge to the ground state in imaginary time propagation, we periodically “annealed” the system with Gaussian noise.

We simulated and characterized ground state solutions for different values of $g_{12}$. From the real space densities $\rho_{\tau(t)} = |\Psi_{\tau(t)}|^2$, we calculated the system-averaged phase separation parameter

$$s = \sum_{(x,y)} \left[ 1 - \frac{\langle \rho_{\tau}(r)\rho_{t}(r) \rangle}{\sqrt{\langle \rho_{\tau}^2(r) \rangle \langle \rho_{t}^2(r) \rangle}} \right],$$

where the sum ran over all points $r = (x, y)$ in the 2D region. For small interspin interactions, stable vortex configurations arose with high vortex eccentricity along the $y$-direction. From the phase profile $\phi_{\tau(t)}(r)$ of each order parameter component, we calculated the total angular momentum, or circulation, of the BEC,

$$\langle \hat{L}_z \rangle_{\tau(t)} = \oint_C \nabla \phi_{\tau(t)}(r) \cdot d\ell = \frac{2\pi\hbar}{m} n_{\tau(t)},$$

where $C$ is a closed, counter-clockwise contour enclosing 99% of the total atom population [the thick black line in Figure 4(d)]. Because each order-parameter component is single-valued, the total number of $2\pi$-phase windings $n_{\tau(t)}$ takes on integer values. In both of our characterizations, We see a clear phase transition in the ground state at $g_{12}/g \approx 0.5$ [Figure 4(e-f)].

Some obvious continuations of this work would simulate negative $g_{12}$ values, as well as various synthetic magnetic field strengths. These magnetic field strengths are limited to a maximum value of $|B_{\tau(t)}|^2 \approx 0.700a_0^{-2}$ by the possible gauge potentials (i.e. $|\mathcal{A}^\tau| \in [0,1]$) and by the physical size of the BEC. It would also be interesting to search for edge effects in a 2D uniform BEC. We see a clear phase transition in the ground state at $g_{12}/g \approx 0.5$ [Figure 4(e-f)].

FIG. 4. (a) A pair of counter-propagating Raman lasers couple two atomic levels of a harmonically-confined BEC. A spatial light modulator device (not shown) tailors the laser intensity to vary proportional to $\Omega(y)$ [Eq. (10)] along the dashed line. (b) The Raman momentum-energy dispersion as a function of $y$-position due to the spatially-tailored Raman coupling profile shown in on the left 3D wall. This coupling is assumed to be uniform along the $x$-direction. The solid red and blue curves indicate the double-well minima, or the gauge potentials $\mathcal{A}_x^\tau(y)\hat{\sigma}_z$. (c) Enlarged detail of the calculated real-space (top, 2.6× mag.) and $k$-space (bottom, 9.8× mag.) ground state densities on a 1024 x 1024 grid for $g_{12} = 0.1$. The trapping frequencies are $\omega_\perp, \omega_\parallel/2\pi = (50, 2000)$ Hz. The smooth central regions of each real-space density correspond to momentum components near $k_y = 0$. (d) The spatial phase profiles $\phi_{\tau(t)}(x, y)$ of the solutions from (c), showing opposite vortex windings in each component. (e) The phase separation of the two components as a function of the interspin interaction strength $g_{12}/g$. Error bars indicate the standard deviation of several trials. (f) The absolute value of $\langle \hat{L}_z \rangle_{\tau(t)}$ averaged for both components, as a function of $g_{12}/g$. We expected that the circulation, and hence the magnitude of the $B_{\tau(t)}$, experienced by each spin component would be the same, however, interactions and the initial random seeding generally tended to imbalance the respective angular momenta for a given simulation trial. For values of $g_{12}/g$ larger than $\sim 0.5$, the spins were completely phase-mixed with no angular momentum present in either component.
Throughout all this, the speedup of our GPU method is evident: a single typical trial executed in 120 minutes on the GPU versus an estimated > 3 days on the i9 CPU. Moreover, all the data presented in Figure 4 would have taken almost 6 months of continuous computation on the i9. We note that extending this procedure to 3D could remain quite challenging: on a cubic 3D mesh with a size of $1024^3$, even a single-component order parameter would require over 17 GB of memory to store it. This is larger than the VRAM capacity of the best graphics cards on the market today. Without sacrificing resolution or precision, more sophisticated computational techniques would be required.

VII. DISCUSSION

We have described a GPU-based approach to solving the GPE that provided a significant speed-up, which let us investigate details of a system that would have been otherwise inaccessible, due to the long times needed to calculate and optimize the system. In our case, the specifics of the GPE made it particularly difficult to access other methods: while some of the terms involved are best calculated in real space, others are better suited to momentum space. With the addition of direct coupling between spinor components and interactions, the approach required sequential calculations involving FFTs between real and momentum space representations. The GPU architecture and its excellent handling of FFTs is well-suited to this algorithm, while other approaches such as multiple-processor parallelization [61] cannot offer the same straightforward advantages.

With the availability and specifications of GPU hardware continuing to improve, we anticipate the approach taken here becoming widespread throughout the physics and scientific communities: without needing to know or manipulate details of the hardware architecture, one can access the advantages GPUs have to offer while working with high-level programmatic tools and relatively inexpensive hardware. Though we worked within the Python environment here, this approach is broadly applicable to other similarly high-level environments. Within the open-source Python environment, we found the many tools available through packages like Numpy and PyTorch worked well for us, and we anticipate that recent updates to other packages like Numba and CuPy will offer advantages for future work in this area, including CuPy’s complex-number support and Numba’s accessible libraries.

VIII. CONCLUSION

In this paper, we demonstrated a straightforward method to accelerate Python code solving the 2D pseudospinor nonlinear Schrödinger/Gross-Pitaevskii equation, achieving speedups of $39 \times$ and $68 \times$. We accomplished this with NVIDIA hardware upgrades on custom workstations, and with relatively minimal changes to our previous code, migrating from NumPy arrays to PyTorch Tensors. With these upgrades, we demonstrated their performance by simulating a spin Hall system with a spatially-varying Raman coupling. This illustrates the simplicity and accessibility of high-performance GPU computing for solving computationally expensive, nonlinear differential equations, and the accessibility of these methods for “everyday” scientific computing.

ACKNOWLEDGMENTS

We thank Zaheen Farraz Ahmad for many insightful discussions on GPU computing, benchmarking, and the various Python libraries available. We also gratefully acknowledge the support of NVIDIA Corporation and their grant of the Titan V GPU used in this work.

BDS and LWC contributed equally to this work.

REFERENCES

* lindsay.leblanc@ualberta.ca

[1] J. Radič, T. A. Sedrakyan, I. B. Spielman, and V. Galitski, Vortices in spin-orbit-coupled Bose-Einstein condensates, Physical Review A 84, 063604 (2011)
[2] F. Dalfovo, S. Giorgini, L. P. Pitaevskii, and S. Stringari, Theory of Bose-Einstein condensation in trapped gases, Reviews of Modern Physics 71, 463 (1999)
[3] W. Bao, The Nonlinear Schrödinger Equation and Applications in Bose-Einstein Condensation and Plasma Physics, in Dynamics in Models of Coarsening, Coagulation, Condensation and Quantization, Vol. 9 {SCIENTIFIC}, 2007 pp. 141-239.
[4] R. Caplan, NLSemagic: Nonlinear Schrödinger equation multi-dimensional Matlab-based GPU-accelerated integrators using compact high-order schemes. Computer Physics Communications 184, 1250 (2013)
[5] G. Kuracz, L. R. Kiperman, F. Reyna, and P. I. Fierens, Simulation of pulse propagation in nonlinear optical fibers using GPUs, in 2016 IEEE Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI) IEEE (IEEE, 2016) pp. 1–5.
[6] M. Brehler, M. Schirwon, D. Goddeke, and P. M. Krummrich, A GPU-Accelerated Fourth-Order Runge-Kutta in the Interaction Picture Method for the Simulation of Nonlinear Signal Propagation in Multimode Fibers, Journal of Lightwave Technology 35, 3622 (2017)
[7] G. Siyu and Z. Jianyong, GPU-accelerated performance on numerically solving multimode Schrödinger equation, in Sixth Symposium on Novel Optoelectronic Detection Technology and Applications, Vol. 11455, edited by H. Jiang and J. Chu, International Society for Optics and Photonics (SPIE, 2020) p. 133.
[8] D. H. Peregrine, Water waves, nonlinear Schrödinger equations and their solutions, The Journal of the Australian Mathematical Society. Series B. Applied Mathematics 25, 16 (1983)
TABLE III. Comparison of research works accelerating solutions to nonlinear Schrödinger equations using GPU parallelism. Almost all the works cited here were conducted in CUDA, while ours was done in Python. The reported speedups are broken down by single- and double-precision; single precision calculations are generally faster than double-precision ones on currently-available GPUs.

| Year | Source | Problem | Language/Interface | GPU Speedup |
|------|--------|---------|-------------------|-------------|
| 2010 | [20]   | 1D dark solitons | CUDA | float32 |
| 2011 | [25]   | BEC in exiton semiconductor | CUDA | float32, float64 |
| 2013 | [4]    | 1D dark solitons | MATLAB® CUDA MEX | float32, float64 |
| 2013 | [60]   | Non-linear optical Bloch equations | CUDA | float32, float64 |
| 2014 | [4]    | Rogue waves | CUDA | float32, float64 |
| 2015 | [24]   | Dipolar solitons in driven BEC | - | float32, float64 |
| 2017 | [23]   | Dipolar BEC | CUDA | float32, float64 |
| 2016 | [5]    | Optical pulse propagation in fibers | CUDA | float32, float64 |
| 2020 | Present work | Pseudospinor BEC | Python/PyTorch | float32, float64 |

† Precision was not specified.

[9] C. Chabalko, A. Moitra, and B. Balachandran, Rogue waves: New forms enabled by GPU computing, Physics Letters A 378, 2377 (2014).
[10] N. Karjanto, The nonlinear Schrödinger equation: A mathematical model with its wide-ranging applications, Understanding the Schrödinger Equation: Some Non-Linear Perspectives (2019) arXiv:1912.10683.
[11] M. Wróblewski, Nonlinear Schrödinger approach to European option pricing, Open Physics 15, 280 (2017).
[12] N. M. A. Silva, GASE: a high performance solver for the Generalized Nonlinear Schrödinger equation based on heterogeneous computing, Ph.D. thesis, Universidade do Porto (2013).
[13] W. Bao, D. Jaksch, and P. A. Markowich, Numerical solution of the Gross-Pitaevskii equation for Bose-Einstein condensation, Journal of Computational Physics 187, 318 (2003) arXiv:0303239v1 [cond-mat].
[14] H. Wang, A time-splitting spectral method for computing dynamics of spinor F=1 Bose-Einstein condensates, International Journal of Computer Mathematics 84, 925 (2007).
[15] W. Bao and Y. Cai, Mathematical theory and numerical methods for Bose-Einstein condensation, Kinetic & Related Models 6, 1 (2013) arXiv:1212.5341.
[16] L. M. Symes and P. B. Blakie, Solving the spin-2 Gross-Pitaevskii equation using exact nonlinear dynamics and symplectic composition, Physical Review E 95, 013311 (2017) arXiv:1610.01269.
[17] V. Lončar, Hybrid parallel algorithms for solving nonlinear Schrödinger equation, Ph.D. thesis, Novi Sad (2017).
[18] X. Antoine and R. Duboscq, GPELab, a Matlab toolbox to solve Gross-Pitaevskii equations I: Computation of stationary solutions, Computer Physics Communications 185, 2969 (2014).
[19] X. Antoine and R. Duboscq, GPELab, a Matlab toolbox to solve Gross-Pitaevskii equations II: Dynamics and stochastic simulations, Computer Physics Communications 193, 95 (2015).
[20] R. M. Caplan and R. Carretero, Simulating the nonlinear Schrödinger equation using the computational capability of NVIDIA graphics cards, in ACSESS Proceedings (2010).
[21] R. Zamora-Zamora, G. A. Domínguez-Castro, C. Trallero-Giner, R. Paredes, and V. Romero-Rochín, Validity of Gross-Pitaevskii solutions of harmonically confined BEC gases in reduced dimensions, Journal of Physics Communications 3, 085003 (2019).
[22] J. Schloss, Massively parallel split-step Fourier techniques for simulating quantum systems on graphics processing units, Ph.D. thesis, Okinawa Institute of Science and Technology Graduate University (2019).
[23] V. Lončar, A. Baláz, A. Bogojević, S. Skrbić, P. Muruganandam, and S. K. Adhikari, CUDA programs for solving the time-dependent dipolar Gross-Pitaevskii equation in an anisotropic trap, Computer Physics Communications 200, 406 (2016).
[24] O. L. Berman, R. Y. Kezerashvili, G. V. Kolmakov, and L. M. Pomirchi, Spontaneous formation and nonequilibrium dynamics of a soliton-shaped Bose-Einstein condensate in a trap, Physical Review E 91, 062901 (2015).
[25] A. Gothandaraman, S. Sadatian, M. Faryniarz, O. L. Berman, and G. V. Kolmakov, Application of Graphics Processing Units (GPUs) to the Study of Non-linear Dynamics of the Exciton Bose-Einstein Condensate in a Semiconductor Quantum Well, in 2011 Symposium on Application Accelerators in High-Performance Computing (IEEE, 2011) pp. 68–71.
[26] P. Wittek and F. M. Cucchietti, A second-order distributed Trotter-Suzuki solver with a hybrid CPU-GPU kernel, Computer Physics Communications 184, 1165 (2013) For the Python wrapper, see the documentation at: https://github.com/trotter-suzuki-mpi/trotter-suzuki-mpi/tree/master/src/Python.
[27] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, A survey of general-purpose computation on graphics hardware, Computer Graphics Forum 26, 80 (2007).
[28] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Phillips, GPU Computing, Proceedings of the IEEE 96, 879 (2008).
When a GPU manufacturer refers to a graphics card’s “architecture”, this is what they mean, i.e. a particular version of the hardware. Specifically, their definition refers to a device’s microarchitecture. J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach, 5th ed. (Morgan Kaufmann/Elsevier, 2012).

CUDA C++ Programming Guide (2020).

P. A. Ruprecht, M. J. Holland, K. Burnett, and M. Edwards, Time-dependent solution of the nonlinear Schrödinger equation for Bose-condensed trapped neutral atoms, Physical Review A 51, 4704 (1995).

S. K. Adhikari, Numerical solution of the two-dimensional Gross-Pitaevskii equation for trapped interacting atoms, Physics Letters A 265, 91 (2000) arXiv:0001361 [cond-mat]

R. Dum, J. I. Cirac, M. Lewenstein, and P. Zoller, Creation of dark solitons and vortices in bose-einstein condensates, Physical Review Letters 80, 2072 (1998).

B. Jackson, J. F. McCann, and C. S. Adams, Vortex Formation in Dilute Inhomogeneous Bose-Einstein Condensates, Physical Review Letters 80, 3903 (1998).

D. L. Feder, C. W. Clark, and B. I. Schneider, Vortex Stability of Interacting Bose-Einstein Condensates Confined in Anisotropic Harmonic Traps, Physical Review Letters 82, 4956 (1999).

A. L. Fetter, Rotating trapped Bose-Einstein condensates, Reviews of Modern Physics 81, 647 (2009) arXiv:0801.2952v1.

X. F. Zhang, Z. J. Du, R. B. Tan, R. F. Dong, H. Chang, and S. G. Zhang, Vortices in a rotating two-component Bose-Einstein condensate with tunable interactions and harmonic potential, Annals of Physics 346, 154 (2014).

S. Eckel, J. G. Lee, F. Jendrzejewski, N. Murray, C. W. Clark, C. J. Lobb, W. D. Phillips, M. Edwards, and G. K. Campbell, Hysteresis in a quantized superfluid “atomtronic” circuit, Nature 506, 200 (2014).

S. W. Seo, W. J. Kwon, S. Kang, and Y. Shin, Collisonal Dynamics of Half-Quantum Vortices in a Spinor Bose-Einstein Condensate, Physical Review Letters 116, 185301 (2016) arXiv:1512.07696.

Y.-K. Liu, H.-X. Yue, L.-L. Xu, and S.-J. Yang, Vortex-pair states in spin-orbit-coupled Bose-Einstein condensates with coherent coupling, Frontiers of Physics 13, 130316 (2018).

Z. Dutton and C. W. Clark, Effective one-component description of two-component Bose-Einstein condensate dynamics, Physical Review A 71, 063618 (2005).

H. Saito and M. Ueda, Spontaneous magnetization and structure formation in a spin-1 ferromagnetic Bose-Einstein condensate, Physical Review A 72, 023610 (2005).

W. Zhang, D. L. Zhou, M.-S. Chang, M. S. Chapman, and L. You, Dynamical Instability and Domain Formation in a Spin-1 Bose-Einstein Condensate, Physical Review Letters 95, 180403 (2005).

S. De, D. L. Campbell, R. M. Price, A. Putra, B. M. Anderson, and I. B. Spielman, Quenched binary Bose-Einstein condensates: Spin-domain formation and coarsening, Physical Review A 89, 033631 (2014) arXiv:1211.3127.

E. Yukawa and M. Ueda, Morphological Superfluid in a Nonmagnetic Spin-2 Bose-Einstein Condensate, Physical Review Letters 124, 105301 (2020) arXiv:1905.07217.

I. B. Spielman, Raman processes and effective gauge potentials, Physical Review A 79, 1 (2009) arXiv:0905.2436.

J. Dalibard, F. Gerbier, G. Juzeliūnas, and P. Ōhberg, Colloquium: Artificial gauge potentials for neutral atoms, Reviews of Modern Physics 83, 1523 (2011) arXiv:1008.5378.

Y. J. Lin, K. Jiménez-García, and I. B. Spielman, Spin-orbit-coupled Bose-Einstein condensates, Nature 471, 83 (2011) arXiv:1103.3522.

W. Bao, S. Jin, and P. A. Markowich, On Time-Splitting Spectral Approximations for the Schrödinger Equation in the Semiclassical Regime, Journal of Computational Physics 175, 487 (2002).

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, E. Cournapeau, T. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, Array programming with NumPy, Nature 585, 55 (2020).

T. Chen, Q. Guo, O. Temam, Y. Wu, Y. Bao, Z. Xu, and Y. Chen, Statistical Performance Comparisons of Computers, IEEE Transactions on Computers 64, 1442 (2015).

T. Hoefler and R. Belli, Scientific benchmarking of parallel computing systems, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC ’15 (ACM Press, New York, New York, USA, 2015) pp. 1–12.

S.-L. Zhu, H. Fu, C.-J. Wu, S.-C. Zhang, and L.-M. Duan, Spin Hall Effects for Cold Atoms in a Light-Induced Gauge Potential, Physical Review Letters 97, 240401 (2006).

X.-J. Liu, X. Liu, L. C. Kwek, and C. H. Oh, Optically Induced Spin-Hall Effect in Atoms, Physical Review Letters 98, 026602 (2007).

M. C. Beeler, R. A. Williams, K. Jiménez-García, L. J. LeBlanc, A. R. Perry, and I. B. Spielman, The spin Hall effect in a quantum gas, Nature 498, 201 (2013).

S. Furukawa and M. Ueda, Quantum Hall phase diagram of two-component Bose gases: Intercomponent entanglement and pseudopotentials, Physical Review A 96, 053626 (2017).

S. Furukawa and M. Ueda, Global phase diagram of two-component Bose gases in antiparallel magnetic fields, Physical Review A 90, 033602 (2014).

H. Takeuchi, Quantum elliptic vortex in a nematic-spin Bose-Einstein condensate (2020), arXiv:2009.03556.

A. L. Gaunt, T. F. Schmidtutz, I. Gottliebovich, R. P. Smith, and Z. Hadzibabic, Bose-Einstein Condensation of Atoms in a Uniform Potential, Physical Review Letters 110, 200406 (2013) arXiv:1212.4453.

G. Demeter, Solving the Maxwell-Bloch equations for resonant nonlinear optics using GPUs, Computer Physics Communications 184, 1203 (2013).

R. Ravisankar, D. Vudragovic, P. Muruganandam, A. Balaz, and S. K. Adhikari, Spin-1 spin-orbit- and Rabi-coupled Bose-Einstein condensate solver (2020), arXiv:2009.13507.