NLSEmagic

Nonlinear Schrödinger Equation Multidimensional Matlab-based GPU-accelerated Integrators using Compact high-order schemes

Ronald M. Caplan* **and Ricardo Carretero**
Nonlinear Dynamical Systems Group, Computational Science Research Center, and Department of Mathematics and Statistics.
San Diego State University
* Now at Predictive Science Inc. www.predsci.com

Funding provided by;
San Diego State University | CSRC | NSF
NSF-DMS-0806762

INTRODUCTION

The nonlinear Schrödinger equation (NLSE) is a universal model describing the evolution and propagation of complex field envelopes in nonlinear dispersive media. As such, it is used to describe many physical systems including the evolution of water waves, nonlinear optics, thermodynamic pulses, nonlinear waves in fluid dynamics, waves in semiconductors, and Bose-Einstein condensates (BEC).

The NLSE can be written in general form as

$$i\frac{\partial\Psi}{\partial t} + a\nabla^2\Psi - V(\mathbf{r})\Psi + s|\Psi|^2\Psi = 0$$

where Ψ is the wavefunction, a and s are parameters determined by the application, and $V(\mathbf{r})$ is an external potential function. The modulus-squared of the wavefunction $|\Psi|^2$ represents the observable in the system (such as the atomic density in a BEC or the intensity of light in nonlinear optics).

Since the NLSE is nonlinear, many of the most interesting dynamical solutions do not have a closed form and require the use of direct numerical simulations. To this end, we have developed a software package called NLSEmagic [1] which integrates the one-, two-, and three-dimensional NLSE and includes visualizations and analysis. The codes include GPU-acceleration, allowing the user to run simulations at super-computer speeds on a single GPU-enabled PC. The code is developed to be user-friendly, and is freely distributed.

MATLAB MEX FUNCTIONS

In order to make NLSEmagic as user-friendly as possible, it is written for use with MATLAB. This allows easy access to plotting and analysis functions, as well as makes the setup of problems easier.

The major drawback in using MATLAB is that MATLAB is a script language and therefore large numerical problems, even when using built-in vectorized operations, can be slow. To address this problem, MATLAB has a special compiler interface which allows one to write pure C or FORTRAN routines and call them from MATLAB. This interface is called MEX and is compatible with many compilers. NLSEmagic takes full advantage of the MEX interface by having the integration of the NLSE done by compiled MEX functions, while analysis and plotting is done by MATLAB scripts.

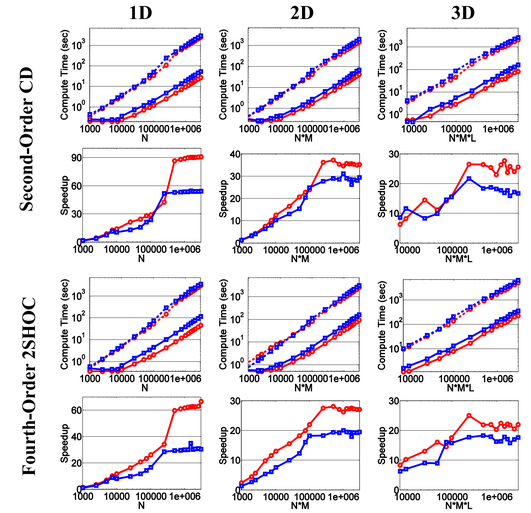
By calling pre-compiled C MEX functions, the codes run as fast as they would in C but can take advantage of the benefits of using MATLAB.
Additionally, the use of C MEX codes increases NLSEmagic's portability since the main integrator codes can easily be ported over to a stand-alone C codes.

GPU CUDA CODES with MEX

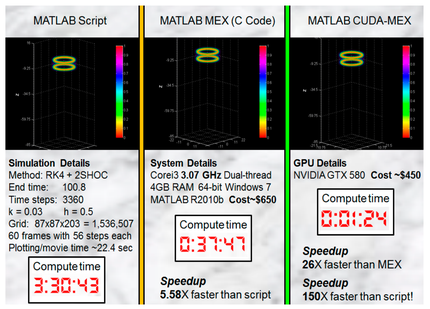
For large simulations, especially those in three dimensions, it is useful to be able to run the codes in parallel to reduce the computation times. Typically this is achieved through large parallel clusters of computers and special C code APIs called MPI and OpenMP. An alternative to this setup is the use of graphical processing units (GPU) which are stand-alone hardware cards that were originally designed for graphics processing. These cards can have hundreds of compute cores on them and are very inexpensive compared to PC clusters. An additional advantage of using GPUs is that a single user on a single PC can have the benefit of high-performance computing without the need of an account on a super-computer. Therefore, keeping in line with the goal of usability, NLSEmagic uses GPU-accelerated integrators.

The GPU codes of NLSEmagic are written in C using the NVIDIA native API called CUDA. These CUDA C codes are then linked to MATLAB through the MEX interface in the same manner as the non-GPU codes. Therefore NLSEmagic will run much faster on a GPU–equipped PC.

Speedup Results
The GPU integrators of NLSEmagic are very efficient. The following speedup results were obtained using a \$450 GPU card (NVIDIA GTX 580). The details of the simulations can be found in the documentation on **www.nlsemagic.com** and in Ref. [1].



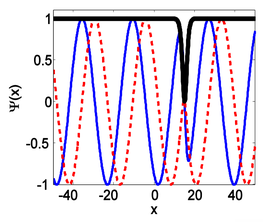
An example of the timing results of a 3D simulation of vortex rings:



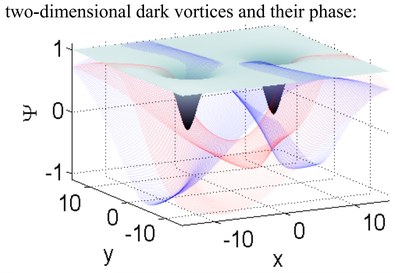
EXAMPLES

Some examples of simulations performed with NLSEmagic:

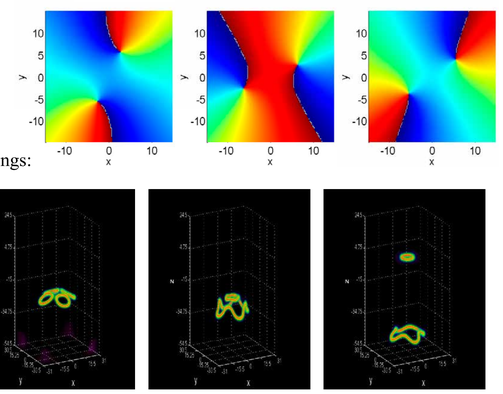
One-dimensional co-moving dark soliton:



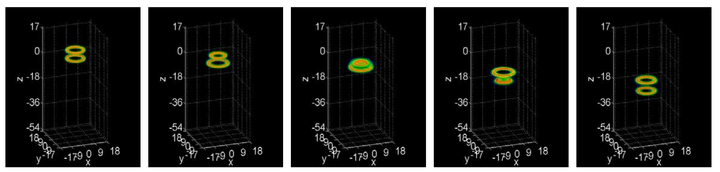
Two rotating, equally-charged two-dimensional dark vortices and their phase:



Four merging three-dimensional dark vortex rings:



Two leap-frogging three-dimensional dark vortex rings:



NUMERICAL SCHEMES

NLSEmagic uses fully explicit finite-difference schemes to integrate the NLSE. The classic fourth-order Runge-Kutta (RK4) is used for the time-stepping, while the spatial derivatives are approximated by either a standard second-order central differencing (CD) or a fourth-order two-step high-order compact (2SHOC) scheme [2]. A compact scheme is used in order to allow easier parallel implementation. The 2SHOC scheme in one-dimension is given by

$$1)\qquad D_i = \frac{1}{h^2}\left(\Psi_{i+1} - 2\Psi_i + \Psi_{i-1}\right),$$

$$2)\qquad \nabla^2\Psi_i \approx \frac{7}{6}D_i - \frac{1}{12}\left(D_{i+1} + D_{i-1}\right).$$

in two-dimensions by

$$1)\quad D_{i,j} = \delta_x^2\Psi_{i,j} + \delta_y^2\Psi_{i,j} = \frac{1}{h^2}\begin{array}{|c|c|c|}\hline & 1 & \\ \hline 1 & -4 & 1 \\ \hline & 1 & \\ \hline\end{array}\,\Psi_{i,j}$$

$$2)\quad \nabla^2\Psi_{i,j} \approx -\frac{1}{12}\begin{array}{|c|c|c|}\hline & 1 & \\ \hline 1 & -12 & 1 \\ \hline & 1 & \\ \hline\end{array}\,D_{i,j} + \frac{1}{6\,h^2}\begin{array}{|c|c|c|}\hline 1 & & 1 \\ \hline & -4 & \\ \hline 1 & & 1 \\ \hline\end{array}\,\Psi_{i,j},$$

and in three dimensions by

$$1)\quad D_{i,j,k} = \frac{1}{h^2}\left(\begin{array}{|c|c|c|}\hline & & \\ \hline & 1 & \\ \hline & & \\ \hline\end{array}\,\Psi_{i,j+1,k} + \begin{array}{|c|c|c|}\hline & 1 & \\ \hline 1 & -6 & 1 \\ \hline & 1 & \\ \hline\end{array}\,\Psi_{i,j,k} + \begin{array}{|c|c|c|}\hline & & \\ \hline & 1 & \\ \hline & & \\ \hline\end{array}\,\Psi_{i,j-1,k}\right),$$

$$2)\quad \nabla^2\Psi_{i,j,k} \approx -\frac{1}{12}\left(\begin{array}{|c|c|c|}\hline & & \\ \hline & 1 & \\ \hline & & \\ \hline\end{array}\,D_{i,j+1,k} + \begin{array}{|c|c|c|}\hline & 1 & \\ \hline 1 & -10 & 1 \\ \hline & 1 & \\ \hline\end{array}\,D_{i,j,k} + \begin{array}{|c|c|c|}\hline & & \\ \hline & 1 & \\ \hline & & \\ \hline\end{array}\,D_{i,j-1,k}\right)$$
$$+\frac{1}{6\,h^2}\left(\begin{array}{|c|c|c|}\hline & & \\ \hline 1 & & 1 \\ \hline & & \\ \hline\end{array}\,\Psi_{i,j+1,k} + \begin{array}{|c|c|c|}\hline 1 & & 1 \\ \hline & -12 & \\ \hline 1 & & 1 \\ \hline\end{array}\,\Psi_{i,j,k} + \begin{array}{|c|c|c|}\hline & & \\ \hline 1 & & 1 \\ \hline & & \\ \hline\end{array}\,\Psi_{i,j-1,k}\right).$$

Boundary Conditions
NLSEmagic has three options for computing boundary conditions. These are:
1) Dirichlet defined as $\Psi_b = \text{constant}$

2) Laplacian-zero defined as $\nabla^2\Psi = 0$

3) A modulus-squared Dirichlet (MSD) condition [3] defined as
$|\Psi|^2 = \text{constant}$
and is formulated in terms of the temporal derivative at the boundary as

$$\Psi_{t,b} \approx i\,\mathrm{Im}\left[\frac{\Psi_{t,b-1}}{\Psi_{b-1}}\right]\Psi_b.$$

and in terms of the Laplacian at the boundary as

$$\nabla^2\Psi_b \approx \left[\mathrm{Im}\left(i\,\frac{\nabla^2\Psi_{b-1}}{\Psi_{b-1}}\right) + \frac{1}{a}\left(N_{b-1} - N_b\right)\right]\Psi_b,$$

where $N_b = s\,|\Psi_b|^2 - V_b, \qquad N_{b-1} = s\,|\Psi_{b-1}|^2 - V_{b-1}.$

Numerical Stability Bounds
The stability restrictions on the time-step can be estimated as [4]

$$k_{\mathrm{CD}} < 0.8 \times \frac{h^2}{d\sqrt{2}\,a},$$

for the RK4+CD scheme and

$$k_{\mathrm{2SHOC}} < 0.8 \times \left(\frac{3}{4}\right)\frac{h^2}{d\sqrt{2}\,a},$$

for the RK4+2SHOC scheme where h is the spatial step-size and d is the dimensionality.

References

[1] R. M. Caplan. NLSEmagic: Nonlinear Schrödinger equation multi-dimensional Matlab-based GPU-accelerated integrators using compact high-order schemes, Comput. Phys. Commun. (2013) doi: 10.1016/j.cpc.2012.12.010.
[2] R. M. Caplan and R. Carretero. A two-step high order compact scheme for the Laplacian operator and its implementation in a fully explicit method for integrating the nonlinear Schrödinger equation. Submitted to Appl. Math and Comp., (2012) arXiv:1109.1027.
[3] R. M. Caplan and R. Carretero. A modulus-squared Dirichlet boundary condition for time-dependent complex partial differential equations and its application to the nonlinear Schrödinger equation. In preparation, arXiv:1110.0569.
[4] R. M. Caplan and R. Carretero. Numerical stability of explicit Runge-Kutta finite-difference schemes for the nonlinear Schrödinger equation. Submitted to App. Num. Math., (2012) arXiv:1107.4810.

www.nlsemagic.com FREE DOWNLOAD