



NonLinear Schrödinger Equation
multidimensional
Matlab-based
gPU-accelerated
integrators using
compact High-order Schemes



SAN DIEGO STATE
UNIVERSITY

Ronald M. Caplan, Ricardo Carretero
Nonlinear Dynamical Systems Group
Computational Science Research Center
San Diego State University



Outline

Nonlinear Schrödinger Equation

Numerical Algorithms

MATLAB and MEX

GPU CUDA MEX

Examples and Timing Results

NLSEmagic Distribution

Nonlinear Schrödinger Equation

Numerous Applications

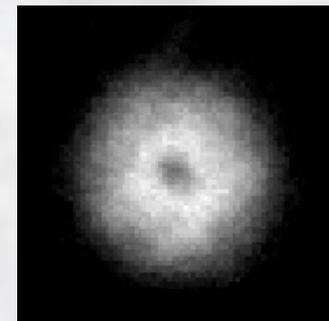


Water waves

Nonlinear optics



Bose-Einstein condensates



Ψ Wavefunction – Real and Imag

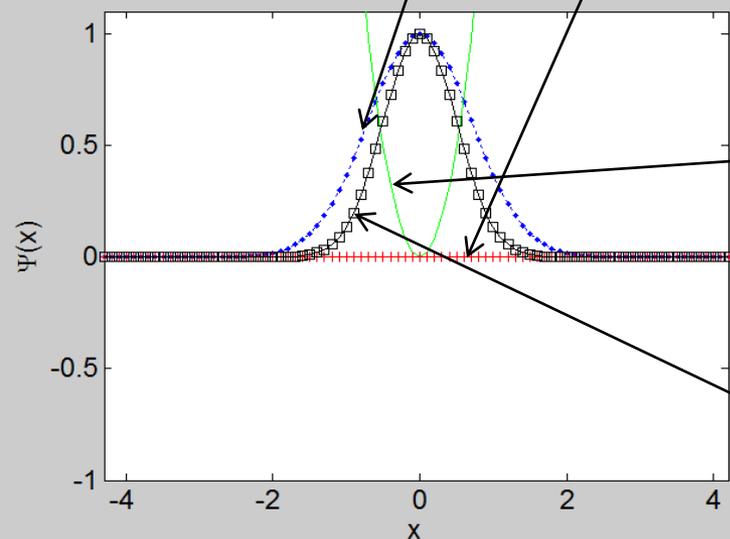
$$i \frac{\partial \Psi}{\partial t} + a \nabla^2 \Psi + (-V(\mathbf{r}) + s |\Psi|^2) \Psi = 0$$

External Potential

Parameters

Laplacian $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$

$|\Psi|^2$ Observable



Numerical Algorithms

Explicit Time Finite-Difference



$$\frac{\partial \Psi_i}{\partial t} = F(\vec{\Psi})$$

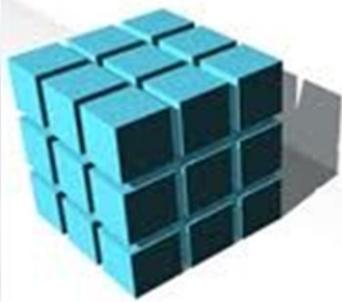
Fourth-Order
Runge-Kutta
(RK4)

$$\Psi^{n+1} = \Psi^n + \frac{\Delta t}{6} (k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = F(\Psi^n) \quad k_2 = F(\Psi^n + \frac{\Delta t}{2} k_1)$$

$$k_3 = F(\Psi^n + \frac{\Delta t}{2} k_2) \quad k_4 = F(\Psi^n + \Delta t k_3)$$

Spatial Finite-Difference



Second-Order
Central Differencing (CD)

$$\nabla^2 \Psi_i = (\delta_x^2 + \delta_y^2 + \delta_z^2) \Psi_i$$

Fourth-Order Two-step
High-Order Scheme (2SHOC)

$$1) D_i = (\delta_x^2 + \delta_y^2 + \delta_z^2) \Psi_i$$

$$2) \nabla^2 \Psi_i = \mathbf{A} D_i + \mathbf{B} \Psi_i$$

Boundary Conditions



Dirichlet

$$\Psi_b = B$$

$$\frac{\partial \Psi_b}{\partial t} = 0$$

Laplacian-Zero

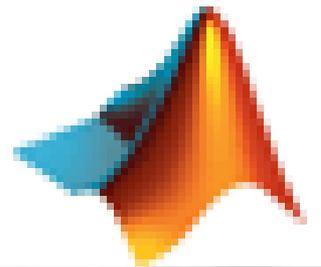
$$\nabla^2 \Psi_b = 0$$

$$\frac{\partial \Psi_b}{\partial t} = i (s |\Psi_b|^2 - V_b) \Psi_b$$

Modulus-Squared Dirichlet

$$|\Psi_b|^2 = 0$$

$$\frac{\partial \Psi_b}{\partial t} \approx i \operatorname{Im} \left[\frac{1}{\Psi_{b-1}} \frac{\partial \Psi_{b-1}}{\partial t} \right] \Psi_b$$



MathWorks® MATLAB

- Used widely as educational tool and for research and industry
- Offers easy integrated visualizations, analysis tools. built-in math functions, easy to code, etc.

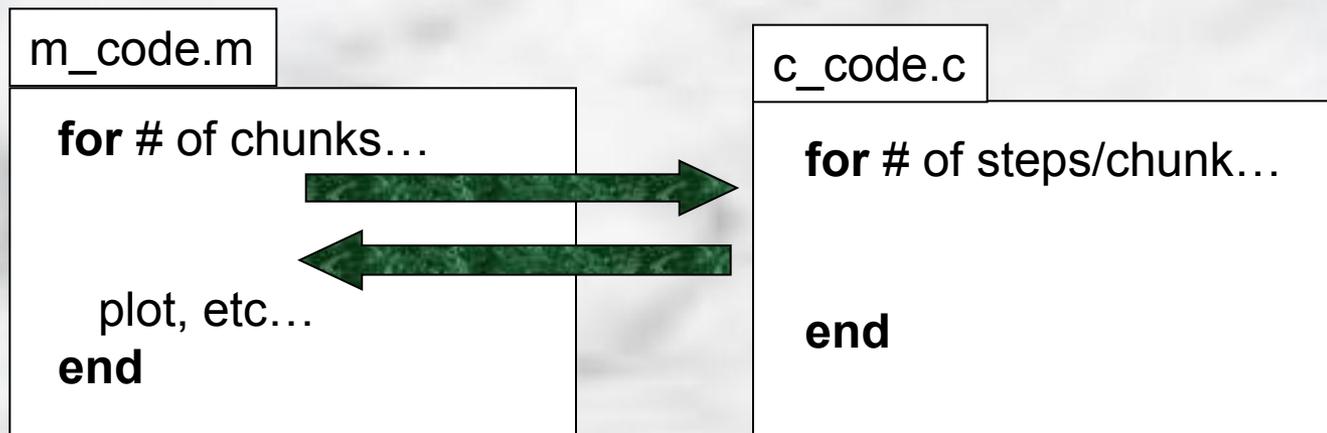
Why not use it for everything?

- Since codes are scripts (not compiled), “real” code use is SLOW

MEX

Solution

- Can write MEX files: C code with MATLAB interface.
- Easy to port if needed
- Allows compiled C code speed with the advantages of MATLAB





Graphical Processing Units (GPUs)

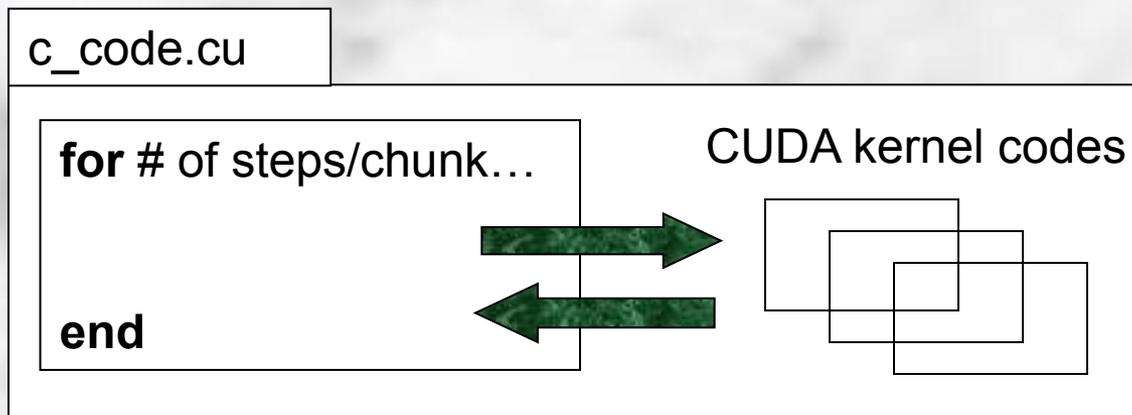
- **Cheap:** ~\$1 per core for desktop card, ~\$5 per core for compute-only workstation card.
- **Fast:** Up to **1024 cores** at ~1.5GHz
Top card boasts peak ~ **2 Tflops** FP, ~ **1 Tflops** DP
- **Medium-large problem capacity:** RAM 1.5GB -> 6.0GB

Why doesn't everyone use it?

- **Not for all problems**
- **Learning Curve:** Need special C/FORTRAN code using CUDA API
- **Preexisting Conditions:** Translation of standard parallel code into GPU "difficult".

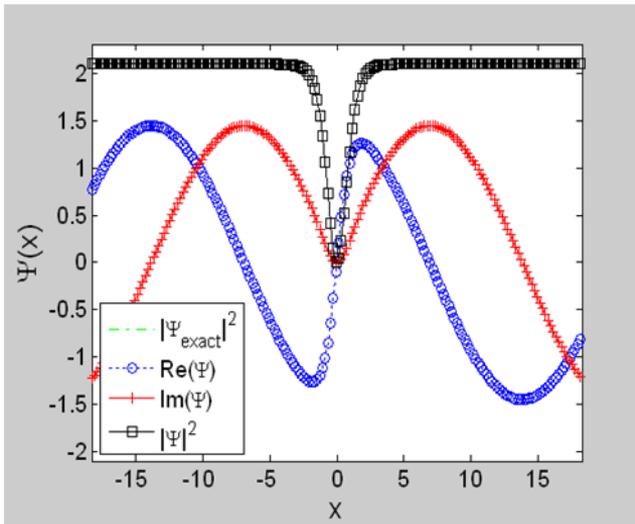
nvMEX

Set of files that allows compiling CUDA GPU codes in a MEX file

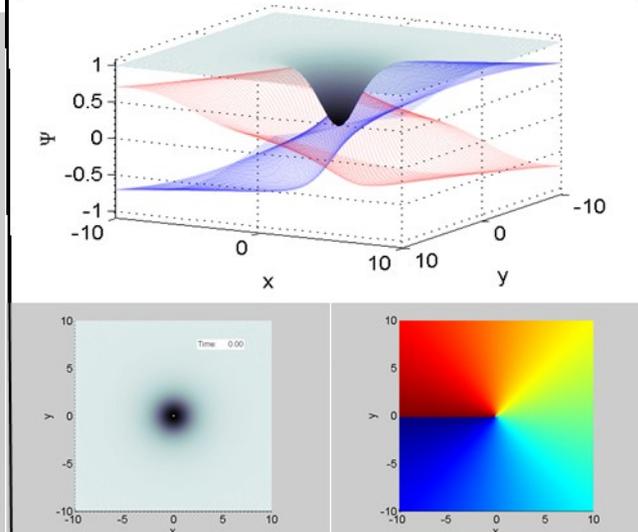


Examples and Timing Results

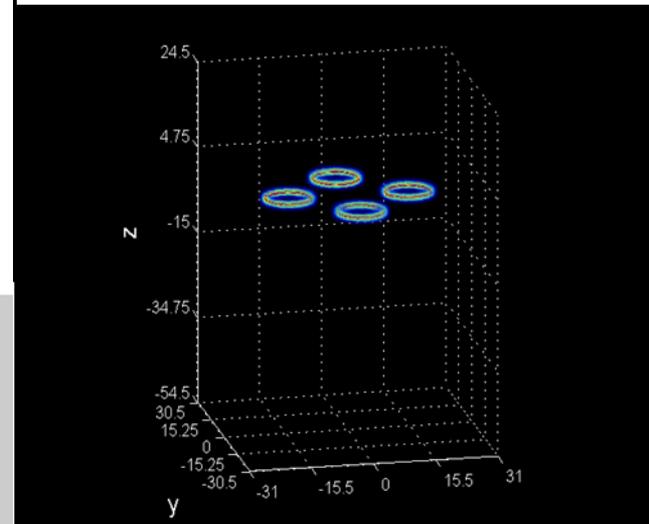
1D Co-moving Dark Soliton



2D Steady-State Vortex

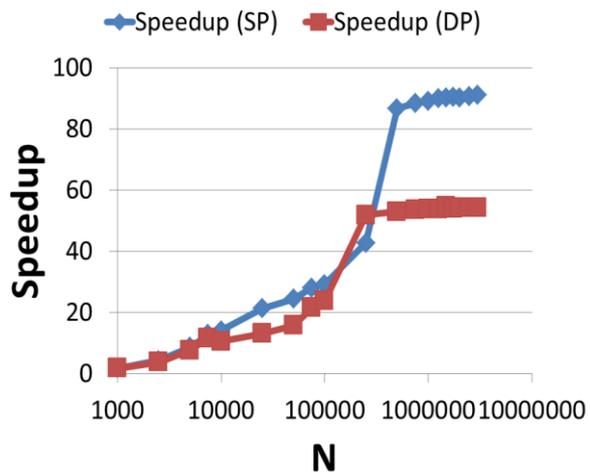


3D Dark Vortex Rings

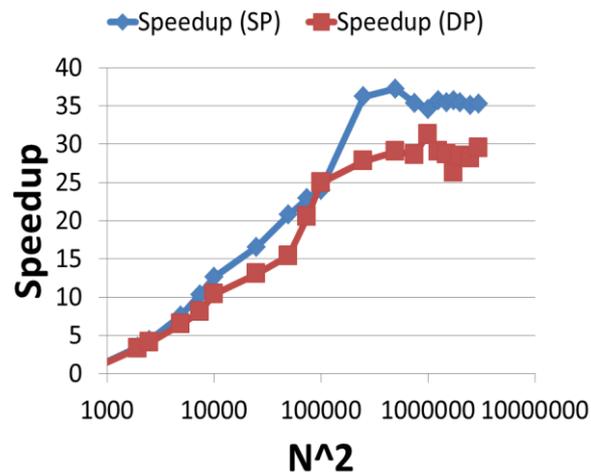


Speedup results (for similar examples):

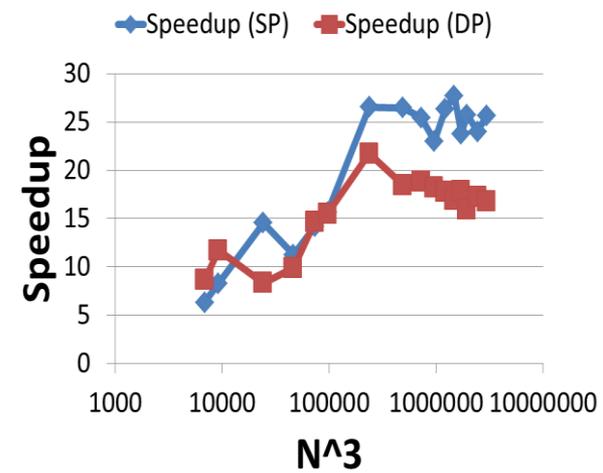
1D



2D

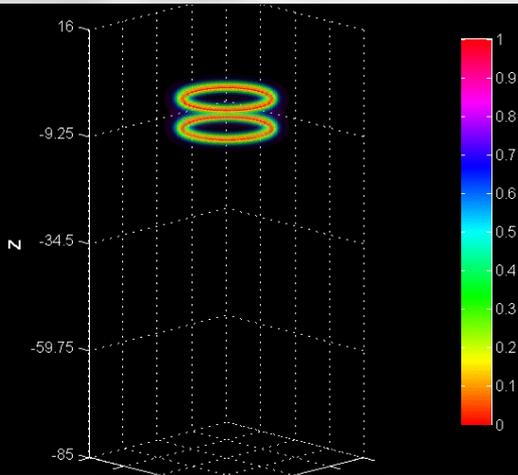


3D



Examples and Timing Results

MATLAB Script



Simulation Details

Method: RK4 + 2SHOC

End time: 100.8

Time steps: 3360

$k = 0.03$ $h = 0.5$

Grid: $87 \times 87 \times 203 = 1,536,507$

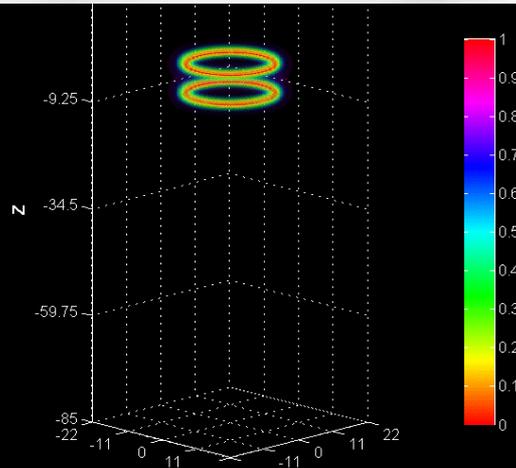
60 frames with 56 steps each

Plotting/movie time ~ 22.4 sec

Compute time

3:30:43

MATLAB MEX (C Code)



System Details

Corei3 **3.07 GHz** Dual-thread

4GB RAM 64-bit Windows 7

MATLAB R2010b **Cost~\$650**

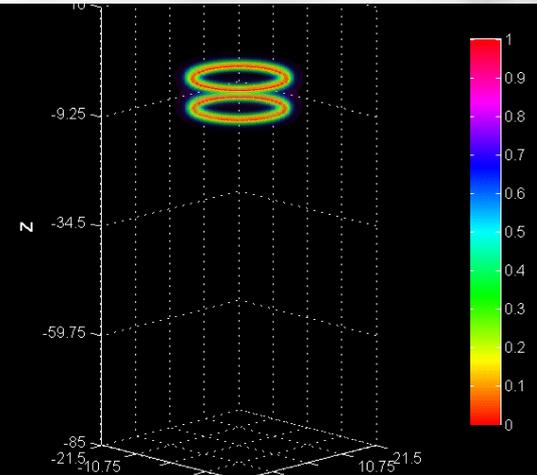
Compute time

0:37:47

Speedup

5.58X faster than script

MATLAB CUDA-MEX



GPU Details

NVIDIA GTX 580 **Cost ~\$450**

Compute time

0:01:24

Speedup

26X faster than MEX

Speedup

150X faster than script!



<http://www.nlsemagic.com>

Home Download Documentation Contact Examples

NLSEmagic

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

Please donate to support NLSEmagic:

[Donate](#)

Like 2

NLSEmagic is a package of C and MATLAB script codes which simulate the nonlinear Schrödinger equation in one, two, and three dimensions. The code includes MEX integrators in C, as well as NVIDIA CUDA-enabled GPU-accelerated MEX files in C. The MATLAB script files call the compiled MEX codes forming an easy-to-use highly efficient program. The codes utilize a fourth-order (in time) Runge-Kutta scheme combined with the choice of standard second-order (in space) finite differencing, or a compact two-step fourth-order (in space) finite differencing.

The code is being developed as part of my Ph.D. thesis project, and will include two versions. One is a streamlined easy-to-follow script code which is meant as an example of how to use the MEX codes, while the other version is a full-research code which can reproduce my research results.

NLSEmagic is freely distributed for use and modification. However, acknowledgment of authorship is appreciated.

Update 10/28/11: All codes and driver scripts have been updated. The new 2D and 3D CUDA codes now run almost twice as fast. Windows 64 compiled binaries have also been added.

Site last updated 10/28/11
© Ronald Caplan 2011

- Full code package available as a free download.
- Setup guide documentation.
- Full research scripts and additional documentation to come soon...

NLSEmagic 

<http://www.nlsemagic.com>

facebook <http://www.facebook.com/nlsemagic>



Questions?