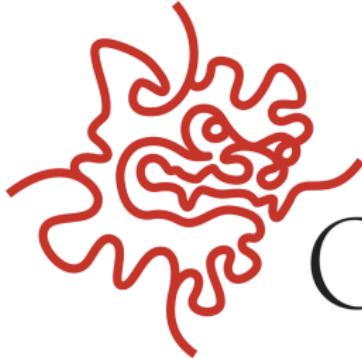


Massively parallel split-step Fourier techniques for simulating quantum systems on graphics processing units

James Schloss

Advisor: **Thomas Busch**
Quantum Systems Unit

December 9, 2019

A red graphic element consisting of several wavy, organic shapes resembling stylized neurons or a brain.

OIST

This project has created fast, GPU-accelerated software for the simulation of superfluid systems

$$\mathbb{GP}^{\hat{U}}_E$$

This project has created fast, GPU-accelerated software for the simulation of superfluid systems

- ▶ The split-step Fourier method

$$\mathbf{G}_\mathbf{P}^{\left(\hat{U}\right)_E}$$

This project has created fast, GPU-accelerated software for the simulation of superfluid systems

- ▶ The split-step Fourier method
- ▶ GPU architecture and the GPUE codebase

$$G_P \binom{\hat{U}}{E}$$

This project has created fast, GPU-accelerated software for the simulation of superfluid systems

- ▶ The split-step Fourier method
- ▶ GPU architecture and the GPUE codebase
- ▶ Chaotic vortex dynamics in a few-vortex system

$$G_P \binom{\hat{U}}{E}$$

This project has created fast, GPU-accelerated software for the simulation of superfluid systems

- ▶ The split-step Fourier method
- ▶ GPU architecture and the GPUE codebase
- ▶ Chaotic vortex dynamics in a few-vortex system
- ▶ Vortex rings with artificial magnetic fields

$$GP\binom{\hat{U}}{E}$$

This project has created fast, GPU-accelerated software for the simulation of superfluid systems

- ▶ The split-step Fourier method
- ▶ GPU architecture and the GPUE codebase
- ▶ Chaotic vortex dynamics in a few-vortex system
- ▶ Vortex rings with artificial magnetic fields
- ▶ Conclusions and future directions

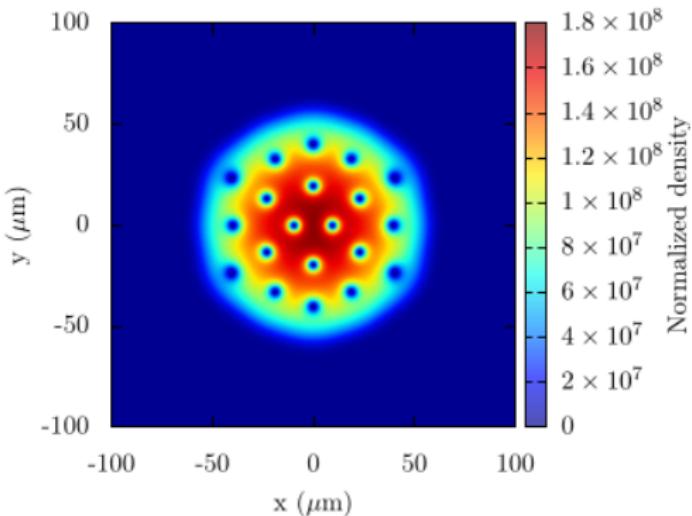
$$GP\binom{\hat{U}}{E}$$

The split-step Fourier method

algorithm-archive.org

The split-step Fourier method

algorithm-archive.org



I'll show you how to make this

The Fourier Transform

Fourier Transform:

$$F(\xi) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i t \xi} dt$$

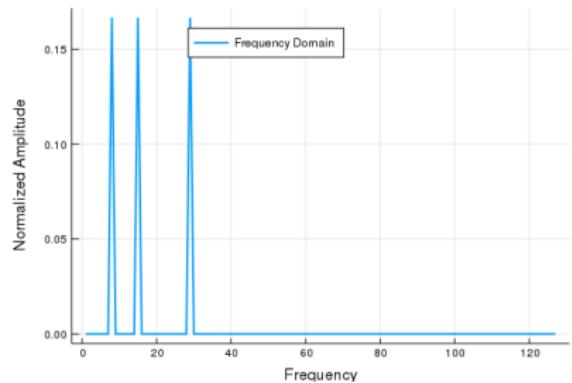
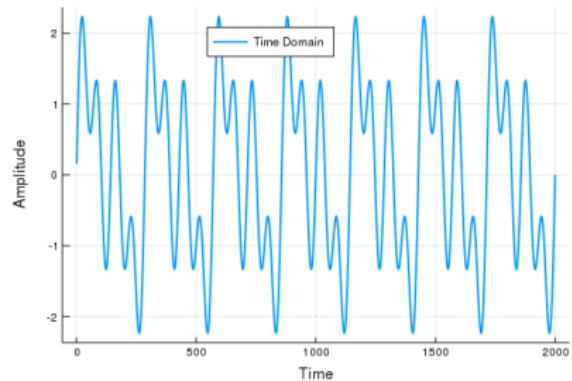
$$f(t) = \int_{-\infty}^{\infty} F(\xi) e^{2\pi i t \xi} d\xi$$

The Fourier Transform

Fourier Transform:

$$F(\xi) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i t \xi} dt$$

$$f(t) = \int_{-\infty}^{\infty} F(\xi) e^{2\pi i t \xi} d\xi$$



The Fourier Transform

Fourier Transform:

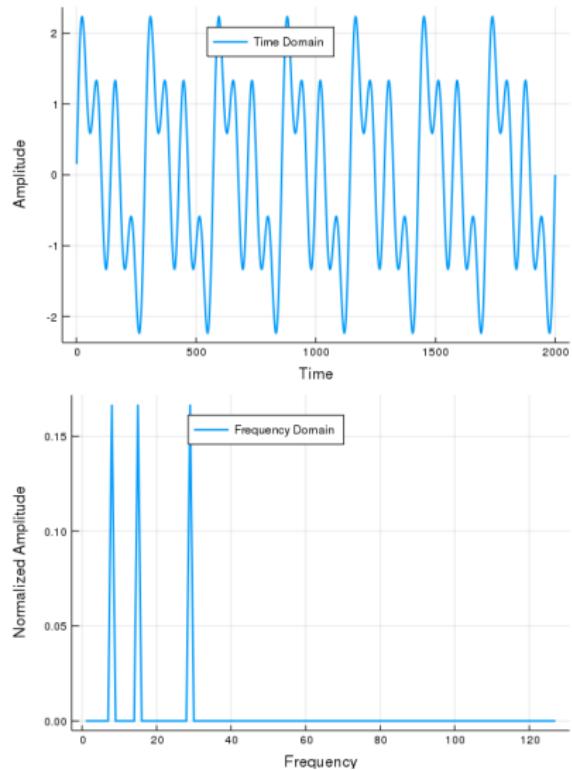
$$F(\xi) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i t \xi} dt$$

$$f(t) = \int_{-\infty}^{\infty} F(\xi) e^{2\pi i t \xi} d\xi$$

Discrete Fourier Transform:

$$X_k = \sum_{n=0}^{N-1} x_n e^{2\pi i k n / N}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{-2\pi i k n / N}$$



The Fourier Transform

Fourier Transform:

$$F(\xi) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i t \xi} dt$$

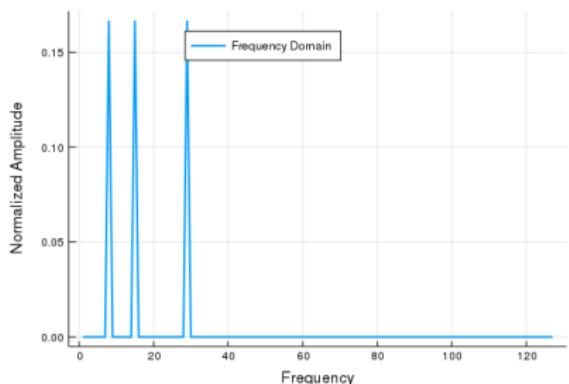
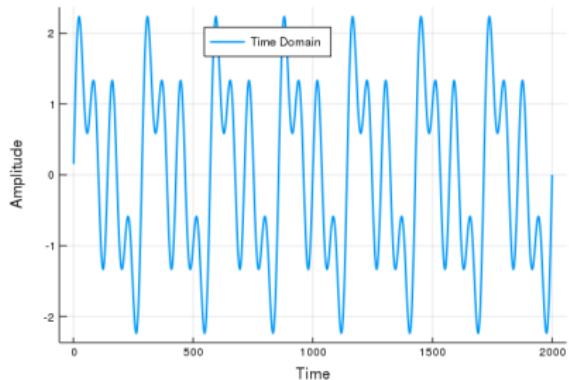
$$f(t) = \int_{-\infty}^{\infty} F(\xi) e^{2\pi i t \xi} d\xi$$

Discrete Fourier Transform:

$$X_k = \sum_{n=0}^{N-1} x_n e^{2\pi i k n / N}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{-2\pi i k n / N}$$

This is a *global* operation
requiring matrix multiplications
→ FFT



1D Schrödinger equation:

$$i\hbar \frac{\partial \Psi(\mathbf{r}, t)}{\partial t} = \left(\frac{p^2}{2m} + V_0(\mathbf{r}) \right) \Psi(\mathbf{r}, t) = \mathcal{H}\Psi(\mathbf{r}, t)$$

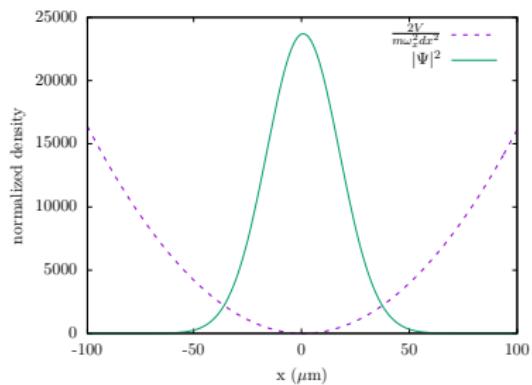
1D Schrödinger equation:

$$i\hbar \frac{\partial \Psi(\mathbf{r}, t)}{\partial t} = \left(\frac{p^2}{2m} + V_0(\mathbf{r}) \right) \Psi(\mathbf{r}, t) = \mathcal{H}\Psi(\mathbf{r}, t)$$

Splits into:

$$\mathcal{H}_V = V_0(\mathbf{r}) = \frac{1}{2}m\omega^2x^2$$

$$\mathcal{H}_p = \frac{p^2}{2m} = i\hbar \frac{\partial}{\partial x}$$



1D Schrödinger equation:

$$i\hbar \frac{\partial \Psi(\mathbf{r}, t)}{\partial t} = \left(\frac{p^2}{2m} + V_0(\mathbf{r}) \right) \Psi(\mathbf{r}, t) = \mathcal{H}\Psi(\mathbf{r}, t)$$

Splits into:

$$\mathcal{H}_V = V_0(\mathbf{r}) = \frac{1}{2}m\omega^2x^2$$

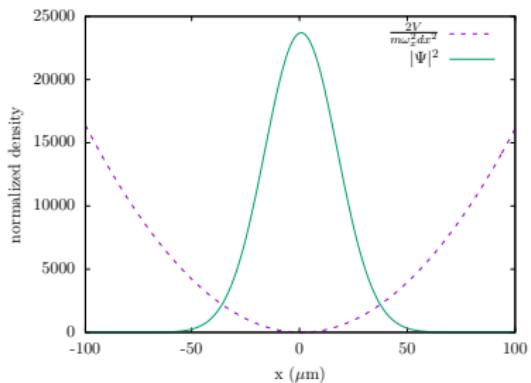
$$\mathcal{H}_p = \frac{p^2}{2m} = i\hbar \frac{\partial}{\partial x}$$

Matrices:

$$U_V = e^{-\frac{i\mathcal{H}_V dt}{\hbar}}$$

$$U_p = e^{-\frac{i\mathcal{H}_p dt}{\hbar}}$$

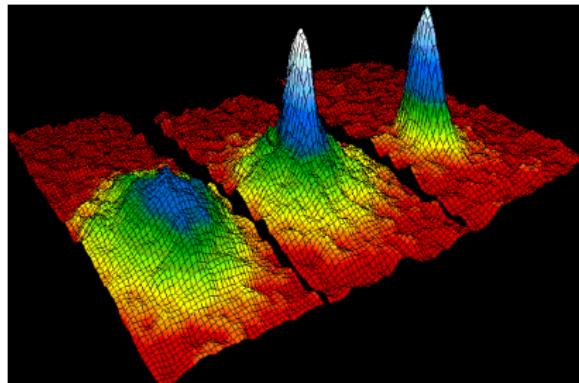
$$\Psi(\mathbf{r}, t + dt) = \mathcal{F}^{-1} [\mathcal{U}_p(dt)\mathcal{F}[\mathcal{U}_V(dt)\Psi(\mathbf{r}, t)]] + \mathcal{O}(dt^3)$$



Bose-Einstein condensation



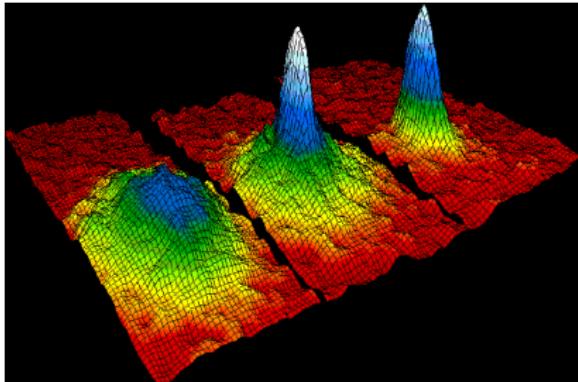
Bosons condense into a superfluid at 0 Kelvin



NIST/JILA/CU-Boulder

Bose-Einstein condensation

Bosons condense into a superfluid at 0 Kelvin



NIST/JILA/CU-Boulder

Described by the Gross–Pitaevskii equation:

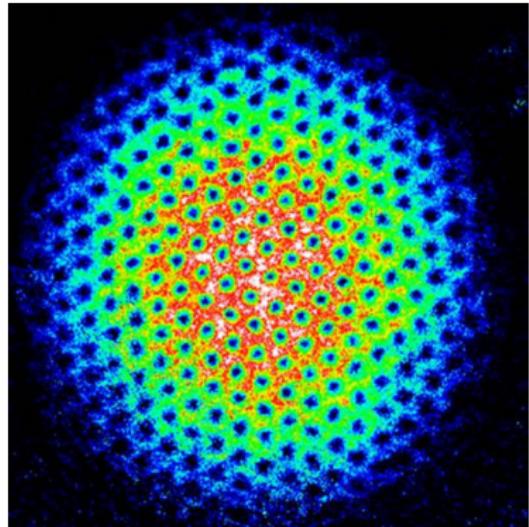
$$i\hbar \frac{\partial}{\partial t} \Psi(\mathbf{r}, t) = \left(-\frac{\hbar^2}{2m} \nabla^2 + V_0(\mathbf{r}) + g|\Psi(\mathbf{r}, t)|^2 \right) \Psi(\mathbf{r}, t)$$

Superfluid rotation

Rotation leads to many vortices along the axis of rotation



Credit: howstuffworks.com



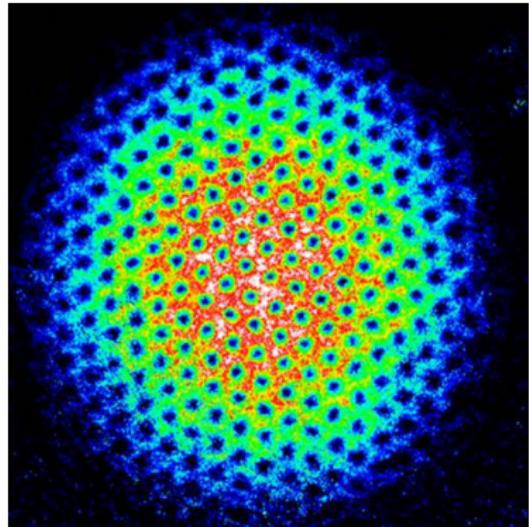
Credit: Peter Engels, JILA

Superfluid rotation

Rotation leads to many vortices along the axis of rotation



Credit: howstuffworks.com

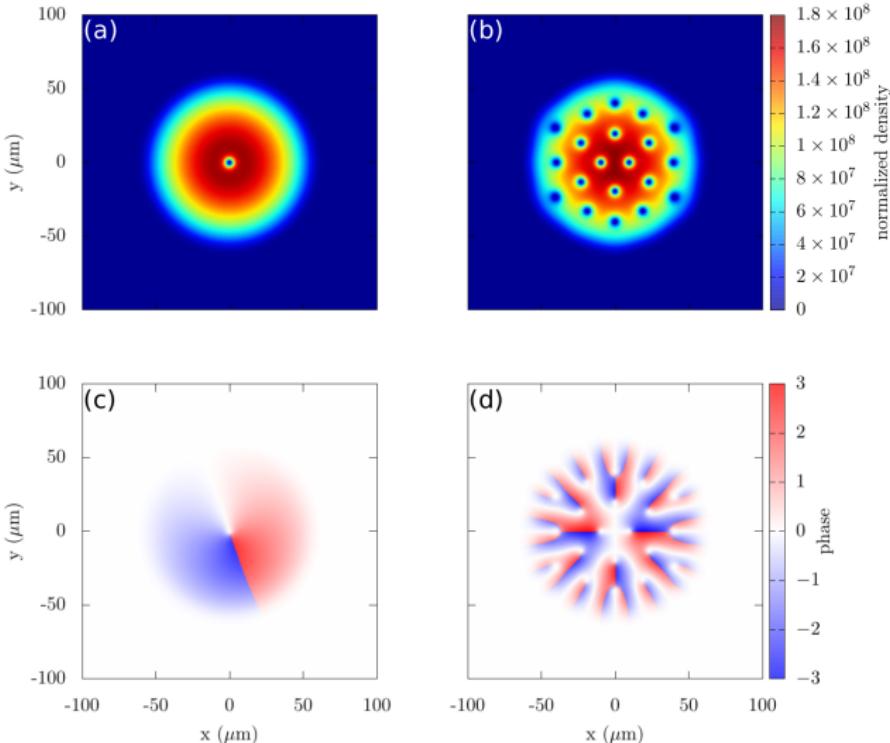


Credit: Peter Engels, JILA

$$i\hbar \frac{\partial}{\partial t} \Psi(\mathbf{r}, t) = \left(-\frac{\hbar^2}{2m} \nabla^2 + V_0(\mathbf{r}) + g|\Psi(\mathbf{r}, t)|^2 - \Omega L_z \right) \Psi(\mathbf{r}, t)$$

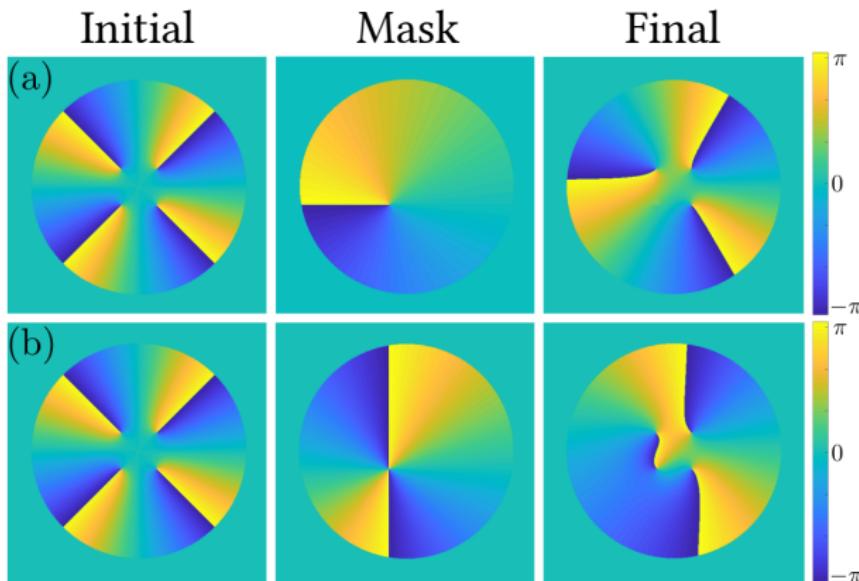
$$L_z = (xp_y - yp_x) = -i\hbar \left(x \frac{\partial}{\partial y} - y \frac{\partial}{\partial x} \right)$$

Superfluid vortex phase



Each vortex has a 2π complex phase winding

Phase imprinting



Phase masks can induce dynamical vortices

Artificial magnetic fields



Magnetic fields cause rotation in *charged* particles

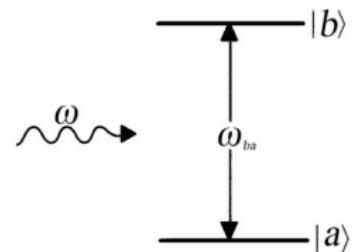
Magnetic fields cause rotation in *charged* particles

- ▶ If a two-level atom moves slowly in a tuned light field, *Berry's connection* is

$$\mathbf{A} = i\hbar \langle \psi_I | \nabla \psi_I \rangle$$

- ▶ The magnetic field is

$$\mathbf{B} = \nabla \times \mathbf{A}$$



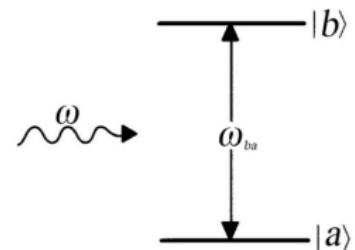
Magnetic fields cause rotation in *charged* particles

- ▶ If a two-level atom moves slowly in a tuned light field, *Berry's connection* is

$$\mathbf{A} = i\hbar \langle \psi_I | \nabla \psi_I \rangle$$

- ▶ The magnetic field is

$$\mathbf{B} = \nabla \times \mathbf{A}$$



- ▶ Vortices follow the magnetic field lines

Modifications to the GPE



With gauge fields, the GPE becomes

$$\mathcal{H} = \frac{(p - m\mathbf{A})^2}{2m} + V_0 + g|\Psi(\mathbf{r}, t)|^2$$

Modifications to the GPE

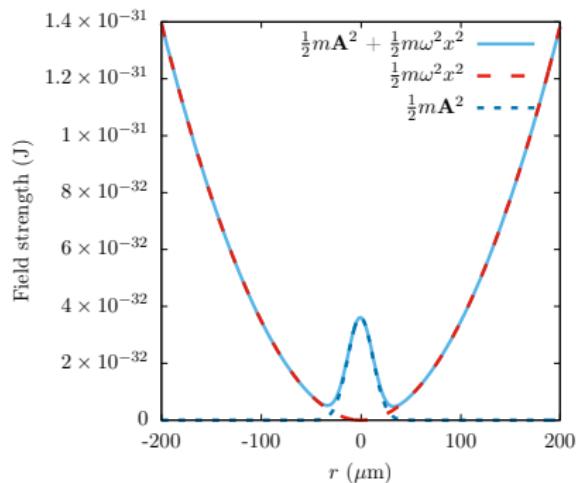
With gauge fields, the GPE becomes

$$\mathcal{H} = \frac{(p - m\mathbf{A})^2}{2m} + V_0 + g|\psi(\mathbf{r}, t)|^2$$

Which creates

- ▶ A position-space component that couples with the trap

$$\frac{m\mathbf{A}^2}{2}$$



Modifications to the GPE

With gauge fields, the GPE becomes

$$\mathcal{H} = \frac{(p - m\mathbf{A})^2}{2m} + V_0 + g|\psi(\mathbf{r}, t)|^2$$

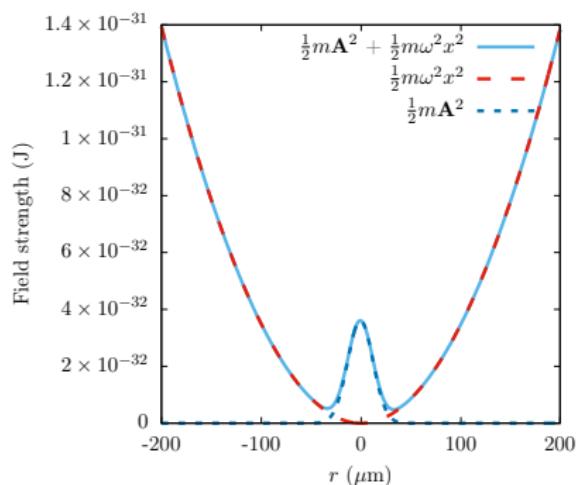
Which creates

- ▶ A position-space component that couples with the trap

$$\frac{m\mathbf{A}^2}{2}$$

- ▶ Components in position and momentum-space, that require 1D FFT's

$$-\left(\frac{p\mathbf{A} + \mathbf{A}p}{2} \right)$$

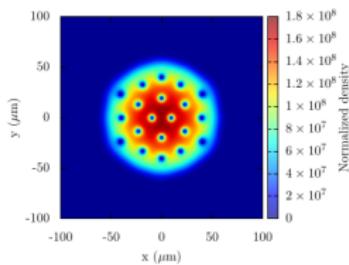


Physics

- ▶ Quantized vortices can be formed with rotation, phase imprinting, and artificial magnetic fields
- ▶ Quantum state engineering can be used
(New Journal of Physics 18 (3), 035012)

Computer Science

- ▶ The SSFM requires a large number of FFT's
- ▶ This system is a great for testing spectral methods



GPU computing and the GPU^E codebase

GPU^E: Graphics Processing Unit Gross–Pitaevskii Equation solver
J Schloss, LJ O'Riordan

Journal of Open Source Software 3 (32), 1037

What is a GPU?

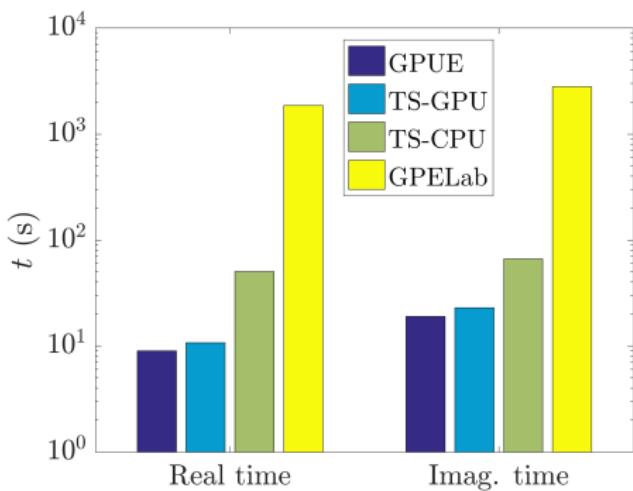
- ▶ Graphics processing units (GPUs) are massively parallel computing devices



What is a GPU?



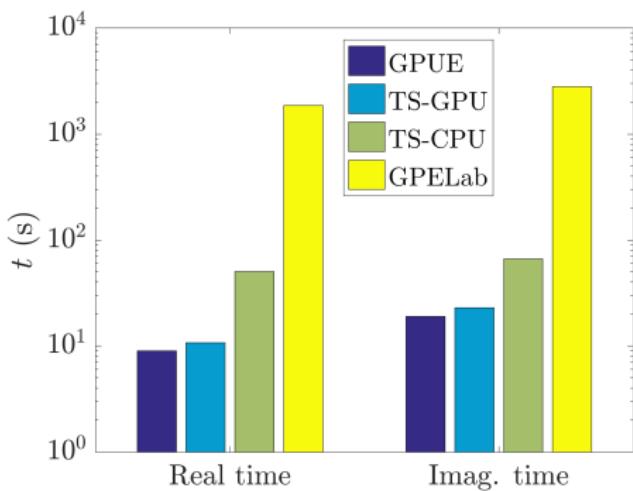
- ▶ Graphics processing units (GPUs) are massively parallel computing devices
- ▶ GPUs are fast for parallel tasks



What is a GPU?



- ▶ Graphics processing units (GPUs) are massively parallel computing devices
- ▶ GPUs are fast for parallel tasks
- ▶ Summit uses GPUs



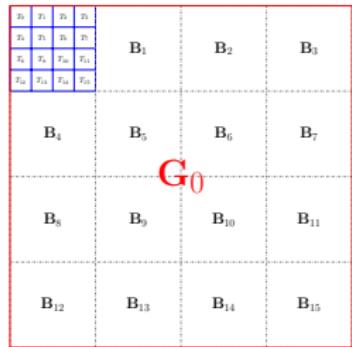
GPU memory hierarchy



- ▶ GPU memory is weird...

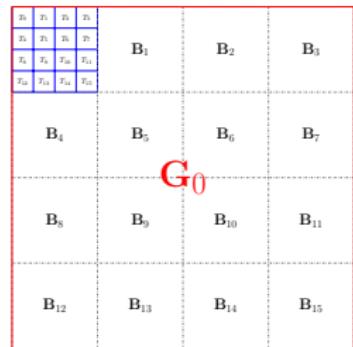
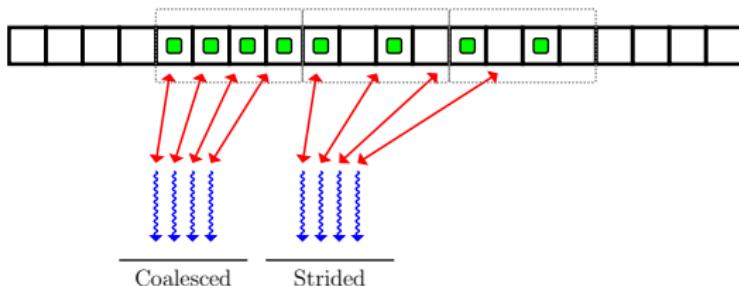
GPU memory hierarchy

- ▶ GPU memory is weird...
- ▶ Computing threads in blocks, blocks in grids



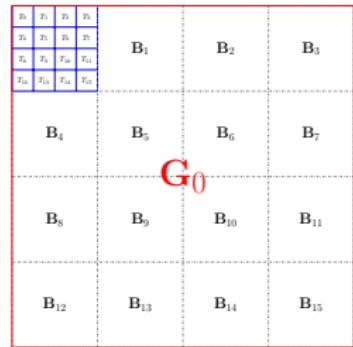
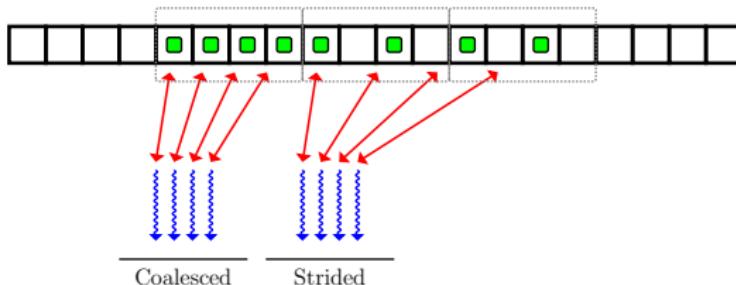
GPU memory hierarchy

- ▶ GPU memory is weird...
- ▶ Computing threads in blocks, blocks in grids
- ▶ Memory coalescence



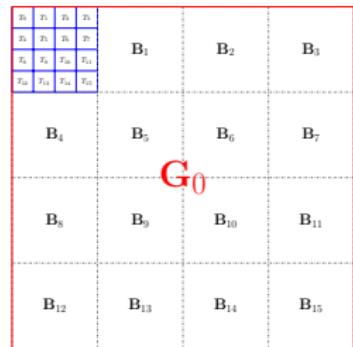
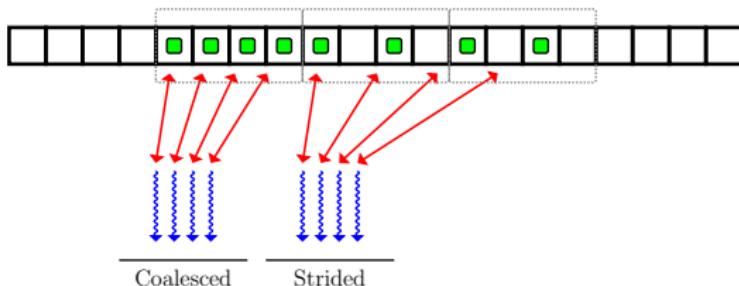
GPU memory hierarchy

- ▶ GPU memory is weird...
- ▶ Computing threads in blocks, blocks in grids
- ▶ Memory coalescence
- ▶ Data transfer is slow



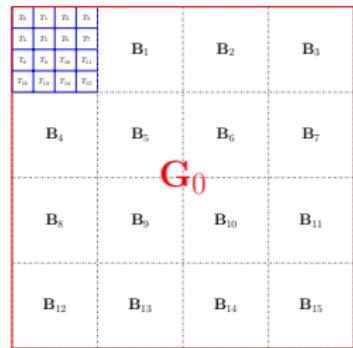
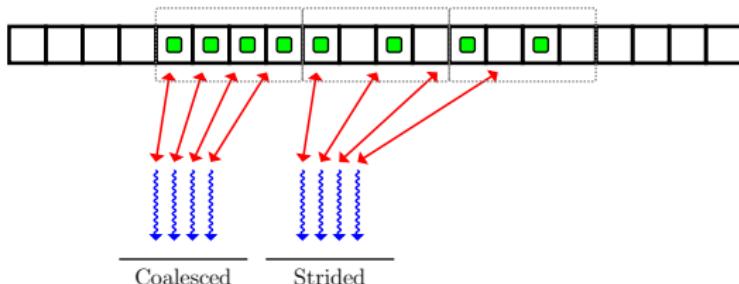
GPU memory hierarchy

- ▶ GPU memory is weird...
- ▶ Computing threads in blocks, blocks in grids
- ▶ Memory coalescence
- ▶ Data transfer is slow
- ▶ Recursion and iteration is slow

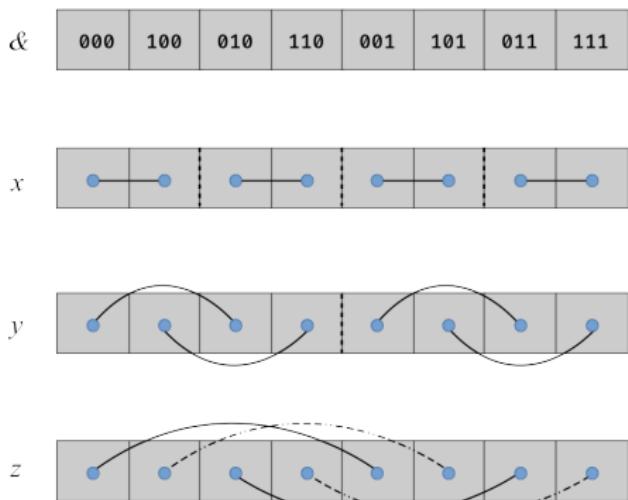
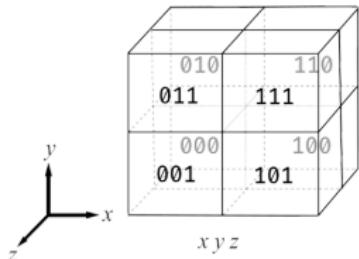


GPU memory hierarchy

- ▶ GPU memory is weird...
- ▶ Computing threads in blocks, blocks in grids
- ▶ Memory coalescence
- ▶ Data transfer is slow
- ▶ Recursion and iteration is slow
- ▶ Limited memory



- ▶ Global operations
- ▶ 1D FFT's are not supported by CuFFT
- ▶ Transposes are necessary



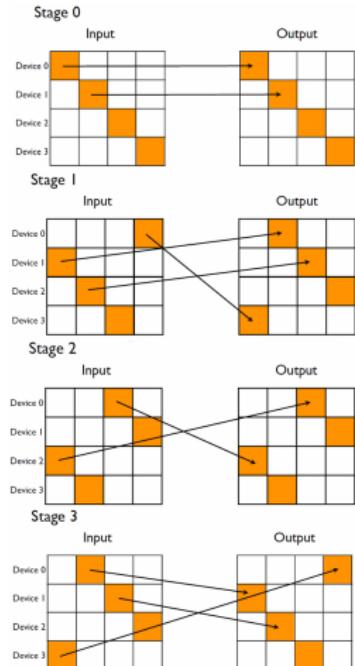
GPU transpose

- ▶ 2D out-of-place transpose \approx copy
(shared memory, coalescence, bank conflicts)
- ▶ In-place transposes are inefficient
- ▶ 3D permutations of arrays have 70% efficiency

Distributed transpose

GPU transpose

- ▶ 2D out-of-place transpose \approx copy
(shared memory, coalescence, bank conflicts)
- ▶ In-place transposes are inefficient
- ▶ 3D permutations of arrays have 70% efficiency
- ▶ 2D distributed example



Ruetsch, G. and Fatica, M. 2013

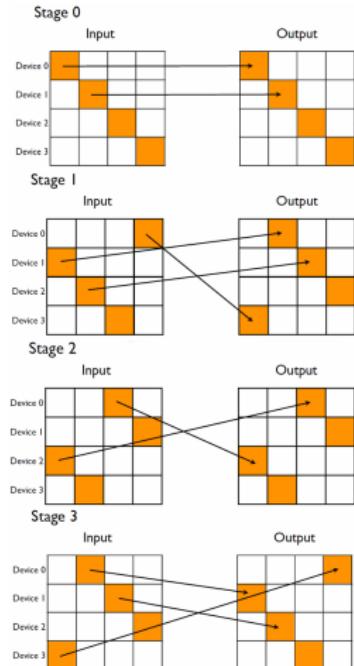
Distributed transpose



GPU transpose

- ▶ 2D out-of-place transpose \approx copy
(shared memory, coalescence, bank conflicts)
- ▶ In-place transposes are inefficient
- ▶ 3D permutations of arrays have 70% efficiency
- ▶ 2D distributed example

Application in GPUE and CLIMA



Ruetsch, G. and Fatica, M. 2013

Two important features

$$GP\left(\hat{U} \atop E\right)$$

- ▶ Novel features useful for understanding examples
- ▶ No discussion of implementation details

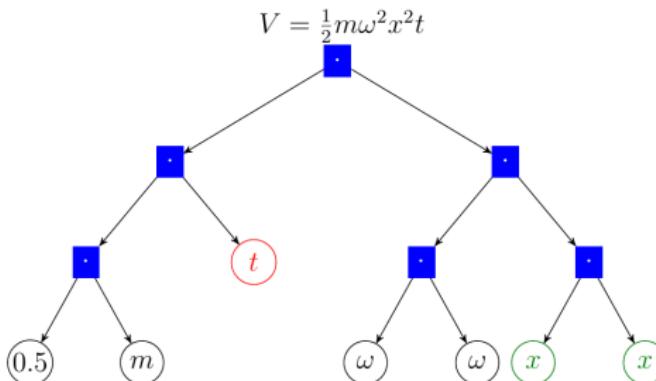
Reminders:

- ▶ CUDA code is hard to write
- ▶ Users want time-dependent fields for state engineering
- ▶ GPUs have limited memory

Expression trees

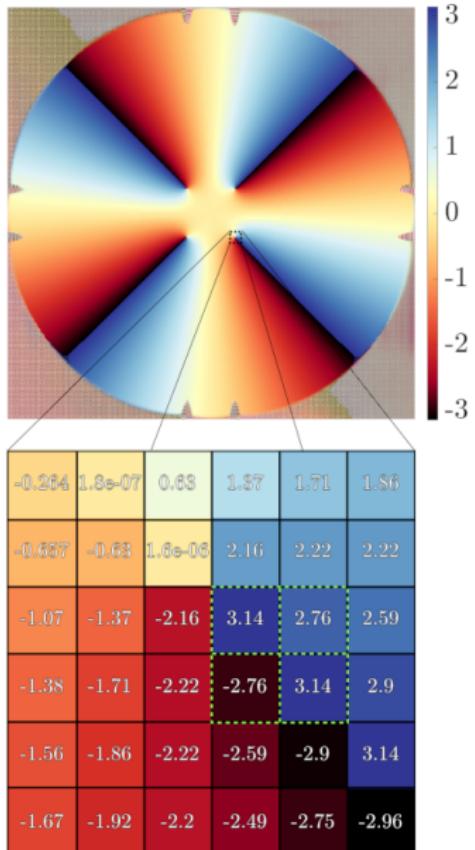
Reminders:

- ▶ CUDA code is hard to write
- ▶ Users want time-dependent fields for state engineering
- ▶ GPUs have limited memory



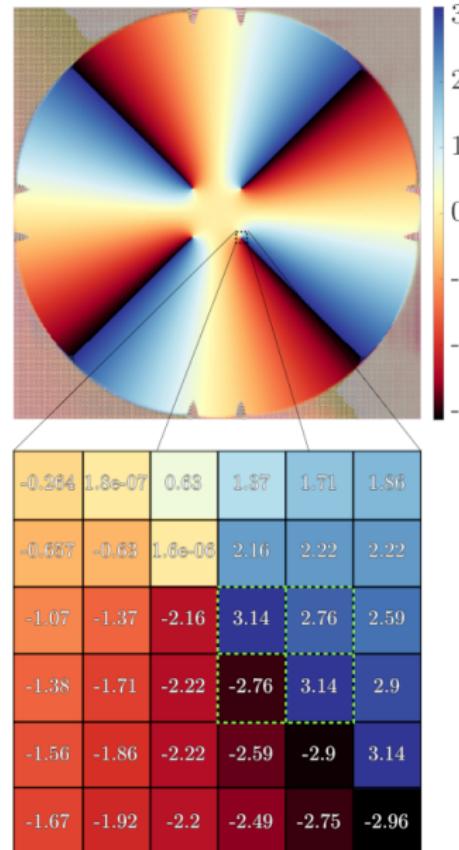
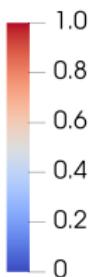
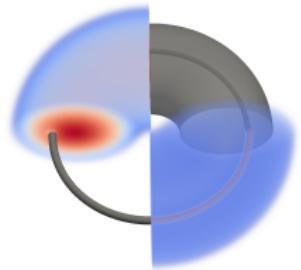
Vortex tracking and highlighting

- Phase plaquettes in 2D



Vortex tracking and highlighting

- ▶ Phase plaquettes in 2D
- ▶ Vortex highlighting in 3D



Physics

- ▶ GPU is fast, can do dynamic simulations, and vortex detection
- ▶ Multi-component simulations and HDF5 have also been implemented

Computer Science

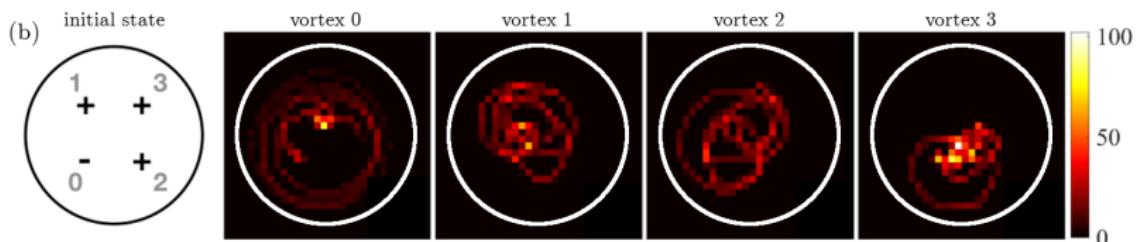
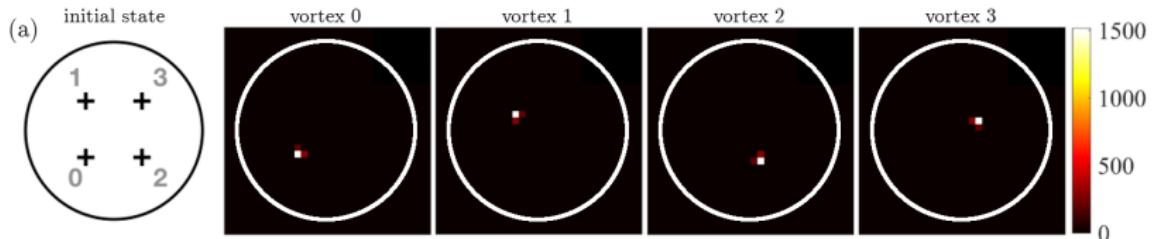
- ▶ The distributed transpose is hard
- ▶ Expression trees save a lot of GPU memory when usable

Chaotic vortex dynamics in 2D BEC simulations

Chaotic few-body vortex dynamics in rotating Bose-Einstein condensates
T Zhang, J Schloss, A Thomasen, LJ O'Riordan, T Busch, A White

Physical Review Fluids 4 (5), 054701

3 vortex, 1 anti-vortex



Dynamics

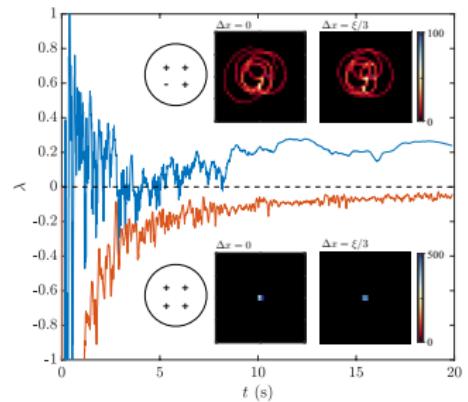
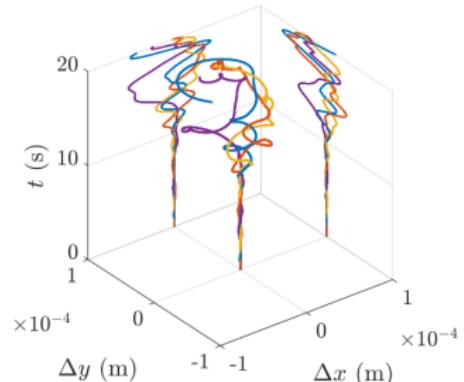


Divergence in trajectories

Divergence in trajectory when Lyapunov exp (λ) becomes positive

$$|\delta \mathbf{P}(t)| \approx e^{\lambda t} |\delta \mathbf{P}_0|$$

$$\lambda = \lim_{t \rightarrow \infty} \frac{1}{t} \ln \frac{||\delta \mathbf{P}(t)||}{||\delta \mathbf{P}(0)||}$$



Physics

- ▶ Chaotic dynamics of few-vortex systems is accelerated by collisional events

Computer Science

- ▶ 50 TB of data
- ▶ Reliant on post-processing metrics (Lyapunov exp, Vortex tracking)
- ▶ 1 hour per run, infeasible with other software

3D vortex ring generation in toroidal BEC systems

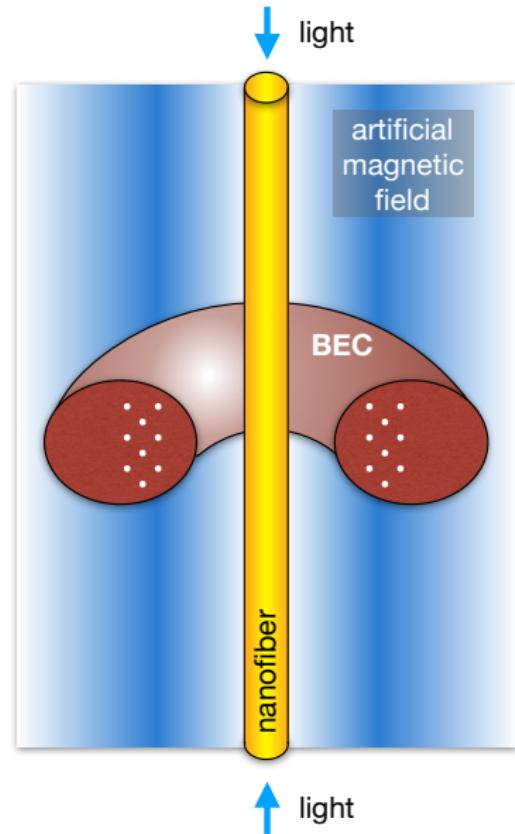
Controlled creation of three-dimensional vortex structures in
Bose–Einstein condensates using artificial magnetic fields

J Schloss, P Barnett, R Sachdeva, T Busch

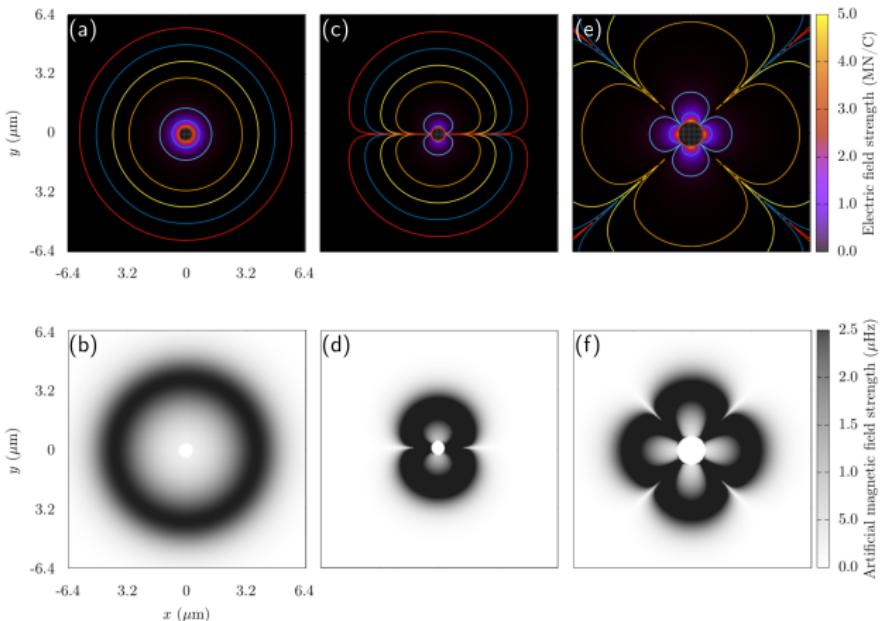
arXiv preprint arXiv:1910.02364

The system

- ▶ BEC toroidally trapped around nanofiber
- ▶ Nanofiber generates **A**
- ▶ Vortices follow $\mathbf{B} = \nabla \times \mathbf{A}$

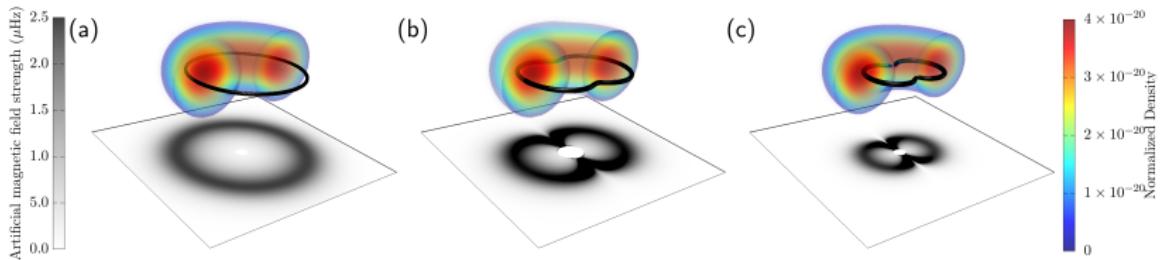


Optical nanofiber

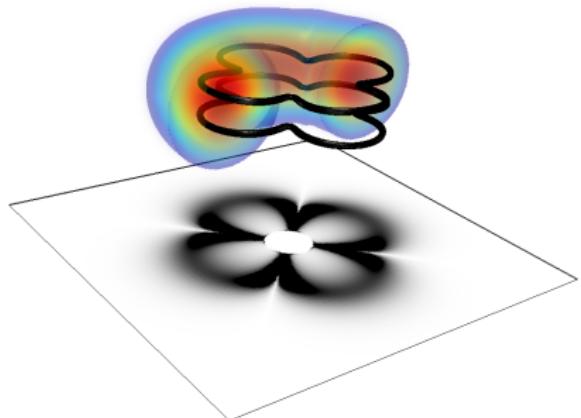


$$\mathbf{A} = z\hbar\kappa_0(n_1 + 1)\tilde{s} \left[\frac{|d_r E_r + d_\phi E_\phi + d_z E_z|^2}{1 + \tilde{s}^2 |d_r E_r + d_\phi E_\phi + d_z E_z|^2} \right]$$

Vortex structures

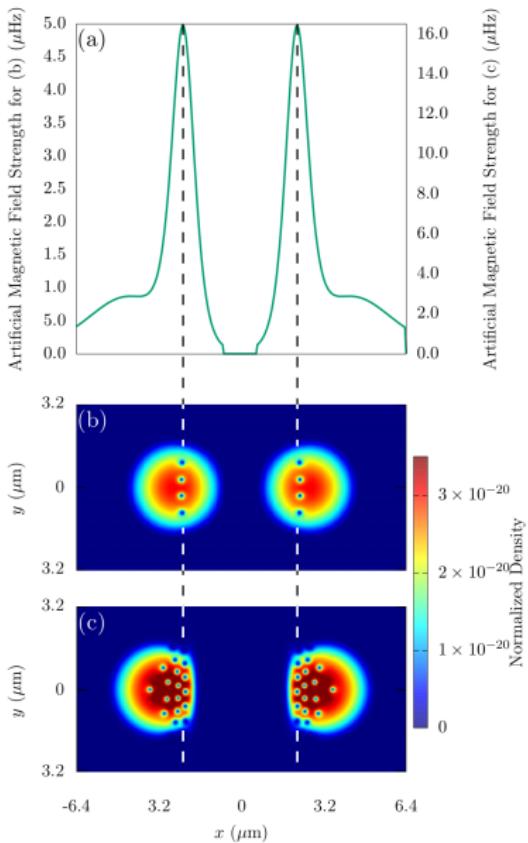


- ▶ Transition with linear polarization
- ▶ Vortex structures align with **B**



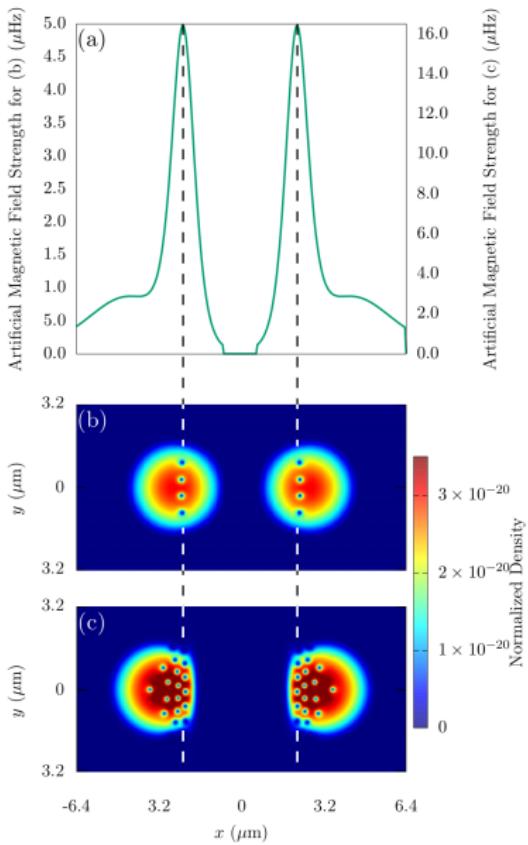
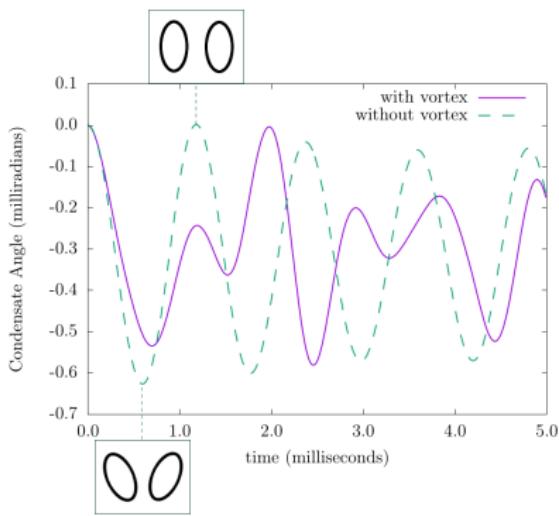
Interesting notes

- ▶ A vortex ring lattice can be generated with high **B**



Interesting notes

- ▶ A vortex ring lattice can be generated with high **B**
- ▶ Scissors modes can be used to detect vortices



Physics

- ▶ We can generate, control, and detect vortex structures in a BEC around a nanofiber
- ▶ Dynamic simulations are underway!

Computer Science

- ▶ Dynamic simulations require a large amount of fileIO
- ▶ Scaling to a larger grid requires multiple GPU's

Overall conclusions

Overall conclusions



- ▶ The SSFM is nice
- ▶ GPU computing is fast
- ▶ We can create chaos
- ▶ We can create not-chaos

- ▶ Distributed Transpose
- ▶ General-purpose Hamiltonian solver
- ▶ Full vortex tracking
- ▶ Octree-based gridding for SSFM
- ▶ GPUE.jl

- ▶ Collaborators: Lee, Rashi, Angela, TT, Andreas, Peter, Albert, Jeremie, Ben, Irina, **Thomas**
- ▶ Sponsors: JSPS, OIST, SCDA
- ▶ Other: AAA

Conclusions



Quantum state engineering

Quantum optimal control



Overall goal: optimize cost function by twiddling control parameters

Overall goal: optimize cost function by twiddling control parameters

- ▶ Many known methods, such as **gradient descent**, **genetic algorithms**, and **machine learning**

Overall goal: optimize cost function by twiddling control parameters

- ▶ Many known methods, such as **gradient descent**, **genetic algorithms**, and **machine learning**
- ▶ For quantum systems, cost function is often the fidelity:

$$\mathcal{F} = |\langle \psi | \phi \rangle|^2$$

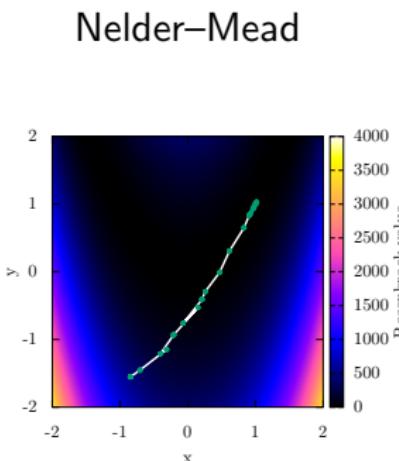
which required re-simulation every time a control parameter is changed

Overall goal: optimize cost function by twiddling control parameters

- ▶ Many known methods, such as **gradient descent**, **genetic algorithms**, and **machine learning**
- ▶ For quantum systems, cost function is often the fidelity:

$$\mathcal{F} = |\langle \psi | \phi \rangle|^2$$

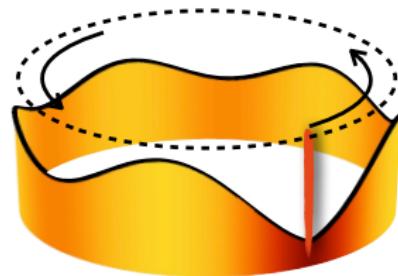
which required re-simulation every time a control parameter is changed



Example Tonks–Girardeau gas system

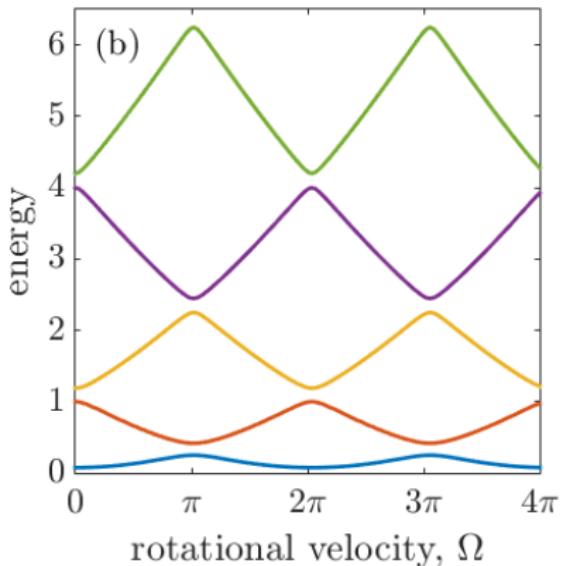
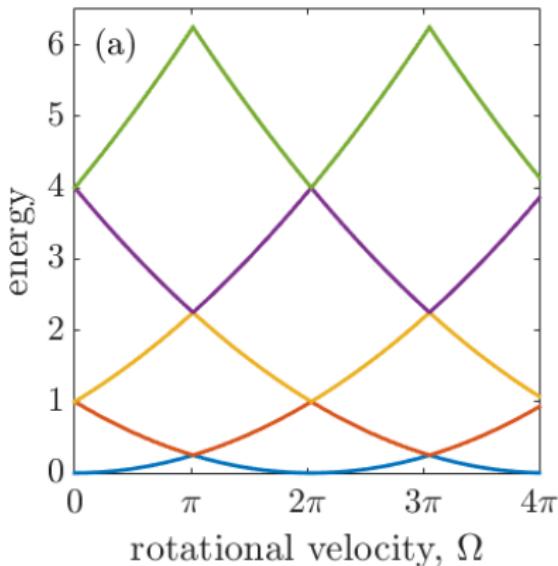
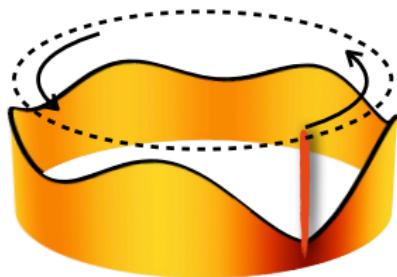


- ▶ NOON state: $|N, 0\rangle + |0, N\rangle$
- ▶ Tonks–Girardeau Gas:
 $g \rightarrow \infty$



Example Tonks–Girardeau gas system

- ▶ NOON state: $|N, 0\rangle + |0, N\rangle$
- ▶ Tonks–Girardeau Gas:
 $g \rightarrow \infty$



An example protocol is the Chopped RAndom Basis (CRAB) optimal control method where...

- ▶ A control parameter is modified with

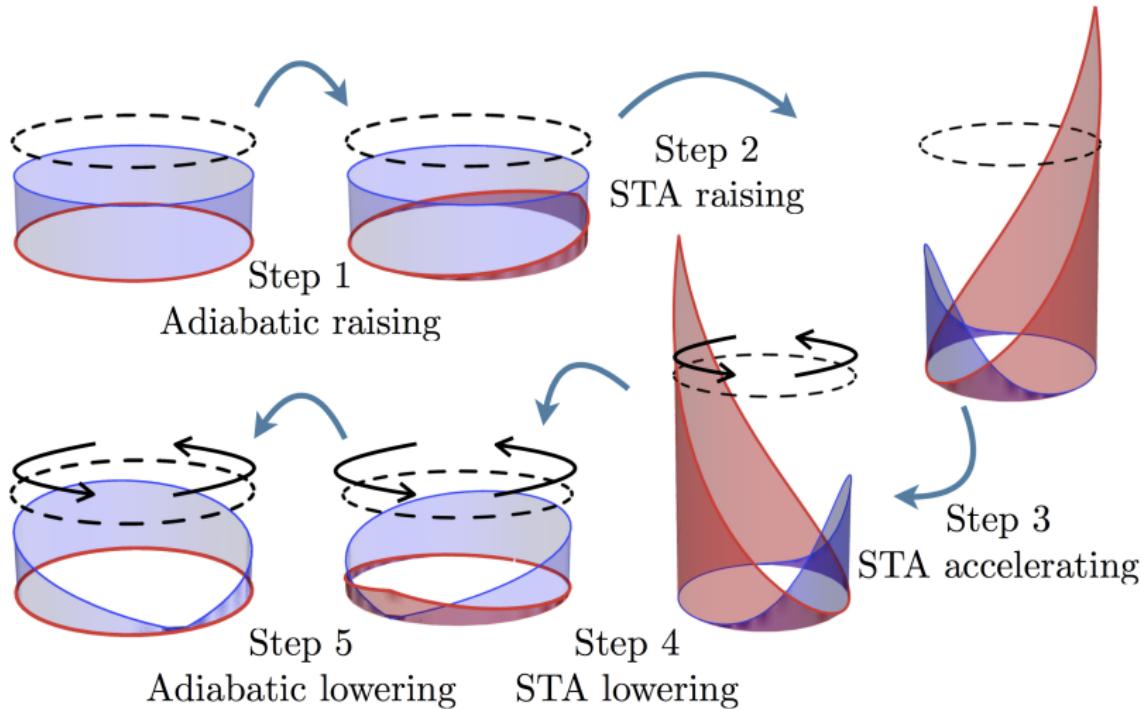
$$\Gamma^{\text{CRAB}}(t) = \Gamma^0(t)\gamma(t)$$

where

$$\gamma(t) = 1 + \frac{1}{\lambda(t)} \sum_{j=1}^J (A_j \sin(\nu_j t) + B_j \cos(\nu_j t))$$

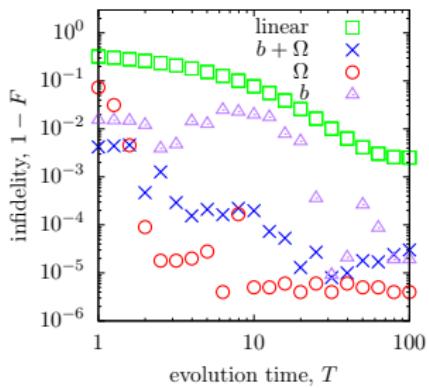
- ▶ Works if $\lim_{t \rightarrow 0} \lambda(t) = \lim_{t \rightarrow T} \lambda(t) = \infty$
- ▶ Creates a $3J$ -dimensional space to optimize (A, B, ν)

STA protocol

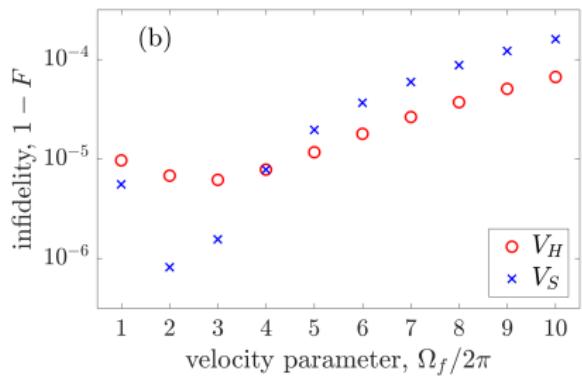


NOON optimization

Optimal control



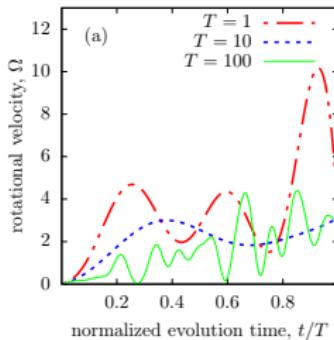
STA



Please ask questions at the end!

Fidelities with optimal control

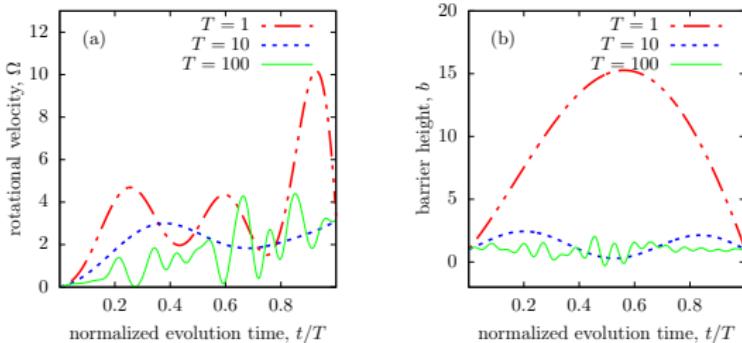
► Rotation ($\Omega(t)$)



Fidelities with optimal control

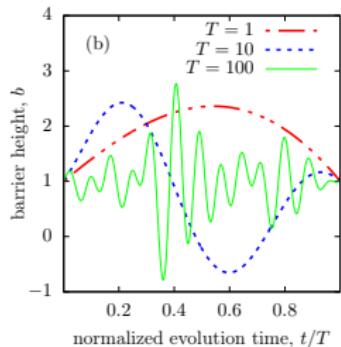
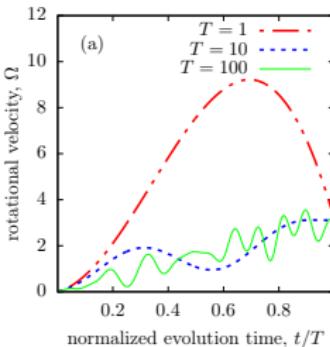
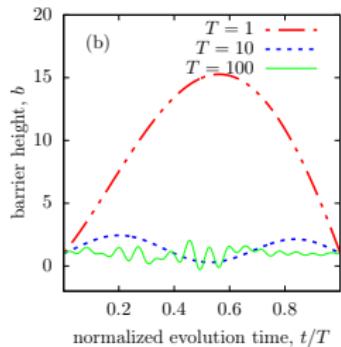
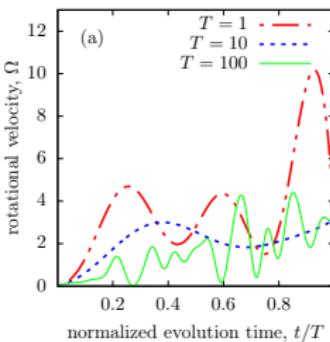


- ▶ Rotation ($\Omega(t)$)
- ▶ Barrier height ($b(t)$)



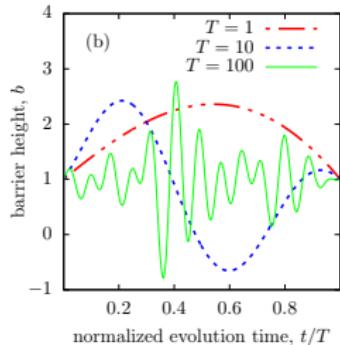
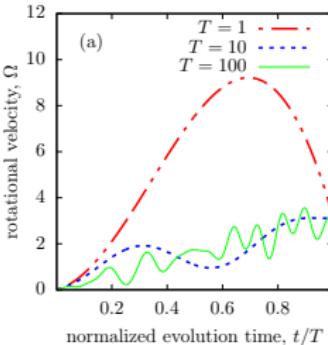
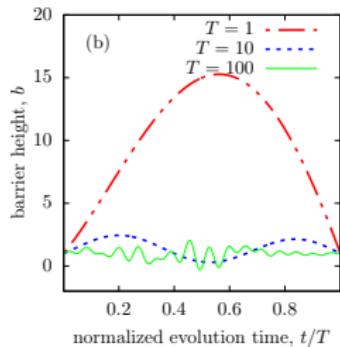
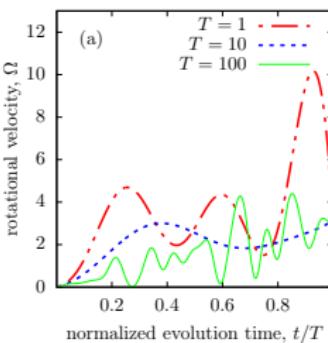
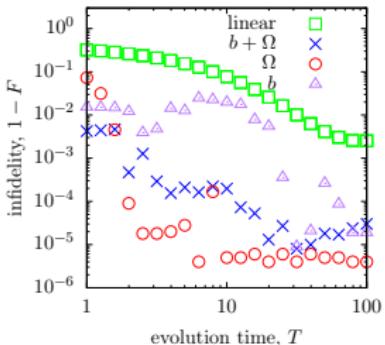
Fidelities with optimal control

- ▶ Rotation ($\Omega(t)$)
- ▶ Barrier height ($b(t)$)
- ▶ Both



Fidelities with optimal control

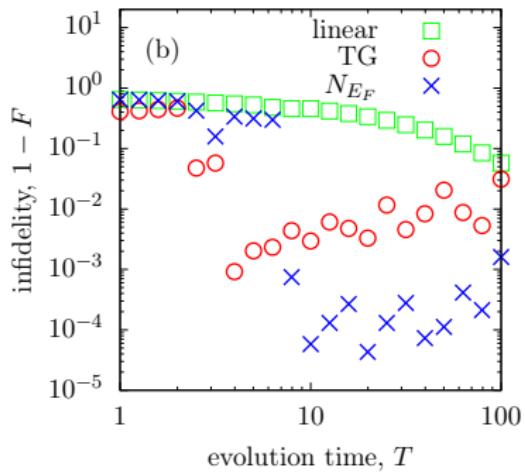
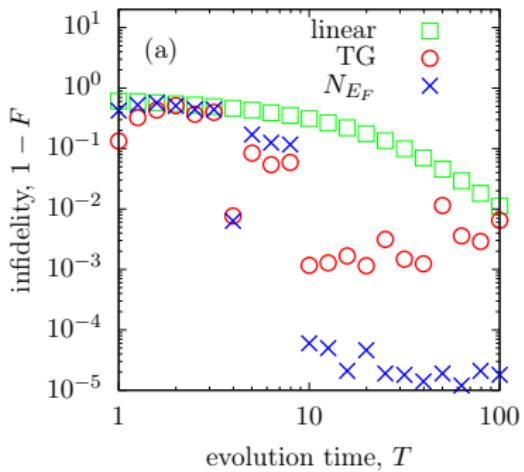
- ▶ Rotation ($\Omega(t)$)
- ▶ Barrier height ($b(t)$)
- ▶ Both



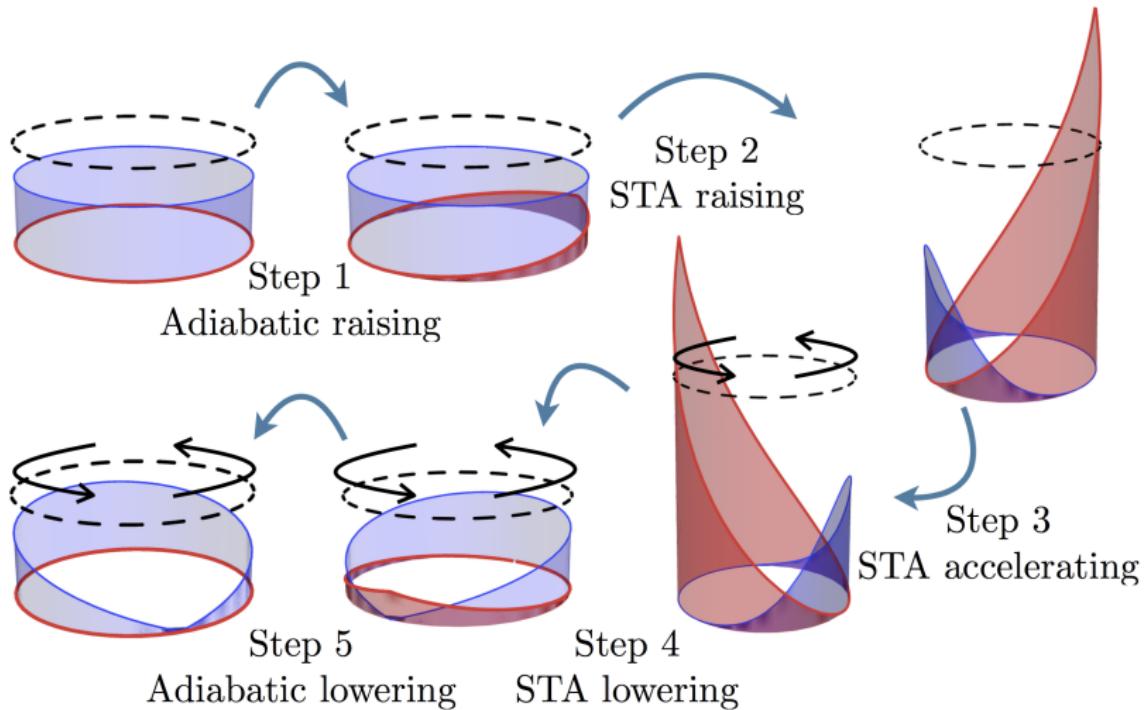
NOON Optimization



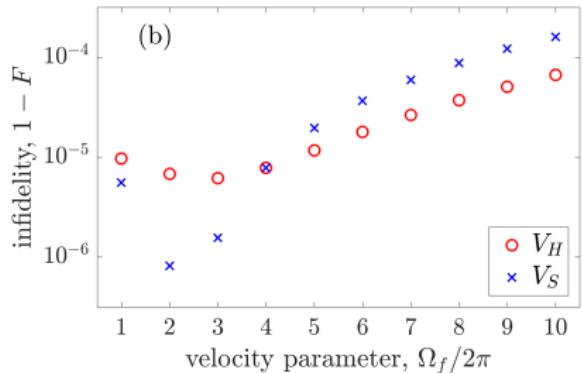
Optimizations of NOON state generation with 3 and 5 particles



STA protocol

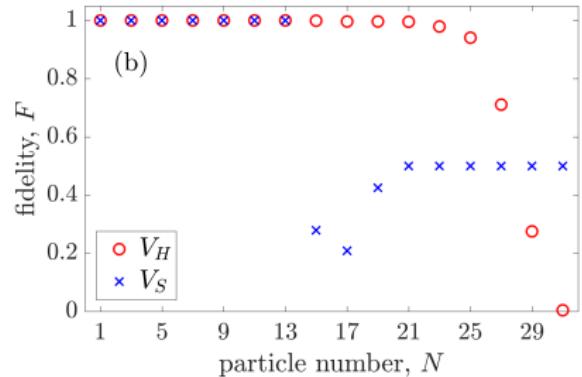
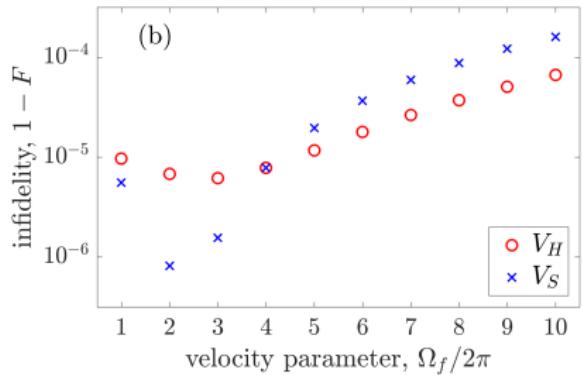
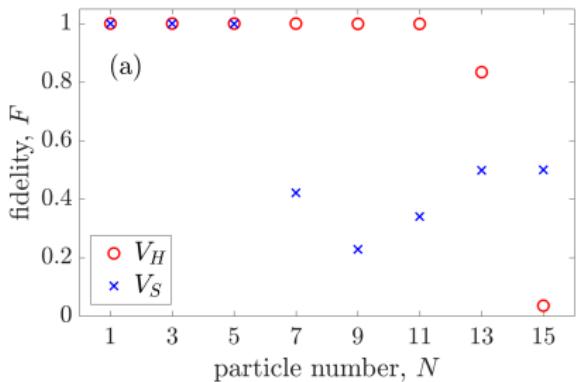


► Fidelities with rotation



STA fidelities

- ▶ Fidelities with rotation
- ▶ Fidelities with rotation of 100, 200 and higher particle number



- ▶ Recursively subdivides DFT into simple sums with twiddle factors
- ▶ Many known libraries, like FFTW, and CuFFT
- ▶ Hard to parallelize (note for later)

