Solving time-dependent Schrödinger equation on parallel platforms
Yaroslav Lutsyshyn, Dieter Bauer

Long-wavelength pulses in 1D

TDSE calculations with large wavelength fields are numerically demanding even in 1D. Given the intensity is kept constant, increasing \( \lambda \) requires

- longer pulse
- with larger electron velocities
- which require larger simulation box
- with finer grid
- and smaller timesteps.

Thus the number of gridpoints and the number of timesteps both grow as \( \lambda^2 \). Overall, computational demand \( \sim \lambda^6 \).

A large-grid simulation is also sensitive to the error accumulation during the propagation. The standard (and very efficient) Thomas algorithm for the reduction of a tridiagonal matrix accumulates a small amount of numerical round-off errors. For a large grid with \( N \sim 10^6 \) gridpoints, this amounts to \( \sim \sqrt{N} \sim 10^{-12} \) of numerical error accumulated per timestep. Combined with the large required number of timesteps (e.g. \( \sim 10^7 \)), the error remains small but may begin to wash out interesting features. Our block-algorithm does not drag the errors through the entire grid and appears to be at least as robust with respect to round-off accumulation as the original serial method.

Parallelizing Crank-Nicolson method

The time-dependent Schrödinger equation on a grid is most conveniently solved with the Crank-Nicolson method.

\[
\left( 1 + i \frac{\Delta t}{2} \right) \psi(t + \Delta t) = \left( 1 - i \frac{\Delta t}{2} \right) \psi(t)
\]

The implicit left-hand stage requires solving a system of linear equations, \( (1 + i \Delta t/2) \psi = \psi(t) \). For \( t \) the matrix is tridiagonal and the equation can be solved very efficiently with the Thomas algorithm (elimination by top-bottom-top pass through the matrix). Unfortunately, this algorithm is quintessentially serial.

We propose to parallelize each stage of the solver by breaking the matrix into square blocks and performing Thomas reduction on each block independently. The lines between the blocks are then made parallelizable each stage of the solver by breaking the matrix into square blocks and performing Thomas reduction on each block independently. The lines between the blocks are then made

The advantages include

- No need for additional memory.
- Leading numerical costs scale as \( 17N/P \).
- Efficient, only \( 3N \) total divisions (vs \( 2N \) for serial method).
- Stable to round-off error accumulation.
- No special requirement on matrix size.
- Blocks can differ in size.
- Same algorithm can be applied to the reduced matrix.
- For \( P \) reduction stages, cost scales as \( \mathcal{O}(N^{3/4}) \) and can be optimized for \( P \gg 1 \).
- Reverse (substitution) stage is trivially parallel, no communication is required.
- Communication cost scales as the number of parallel workers and is independent of the matrix size.

We simulated a model 1D hydrogen atom \((V = -1/\sqrt{x^2 + y^2})\) in a 3-cycle laser pulse. The intensity was fixed \((2 \times 10^{14} \text{ W/cm}^2)\) while we varied wavelength from 1 to 7 \( \mu \text{m} \).

The largest grid corresponded to \( 3 \times 7 \mu \text{m} \):
- \( \lambda = 0.00651 \times 4 \times 10^4 \text{ cm} \)
- \( \Delta x = 0.025 \text{ cm} \)
- \( \Delta t = 0.0022 \text{ s} \)
- \( N \approx 1.45 \times 10^6 \text{ timesteps} \)

Such a calculation currently takes 11 hours on Nvidia Kepler K40 GPU programmed in CUDA-C with our parallelization scheme (3 reduction stages starting with 3000 threads).

Energy spectra: Comparison of electron spectra obtained with the window operator for various pulse wavelengths. The same parallelization scheme is applicable for the window operator itself.

Test system: 1D atom in a long-wavelength pulse

In the case of atoms in intense laser fields in 3D, the wavefunction can be expanded in a relatively small number of spherical harmonics to greatly enhance the performance. For a GPU, the \( i \)-dimension is smaller than the number of parallel threads, thus one needs to parallelize the \( r \)-direction of the grid.

The same communication problems as on the \( x \times y \) grid arise on the \( r \)-grid. Additionally, the number of parallel threads can be greater than \( \lambda \) (large MPI systems or GPU). Thus the need to create the TDSE algorithm that is parallelized in the \( r \)-direction. Such approach also allows for a smaller communication overhead than the parallelization in the \( x \)-direction.

Extension to higher dimensions

Distributed-memory parallelization of 2D (and larger) grids makes a "trivial" alternating-direction parallelization impractical because of the need to transfer on each step the amounts of data comparable to the number of gridpoints.

Our method allows to keep blocks of wavefunction data fully local to parallel processes, while a minimal amount of information on the "joining line" is exchanged. Thus the numerical load per step is \( \sim N^2 \), while the communication is limited to \( \sim N \). Most of the communication can also be time-masked by the computation.

In the case of atoms in intense laser fields in 3D, the wavefunction can be expanded in a relatively small number of spherical harmonics to greatly enhance the performance. For a GPU, the \( i \)-dimension is smaller than the number of parallel threads, thus one needs to parallelize the \( r \)-direction of the grid.

The same communication problems as on the \( x \times y \) grid arise on the \( r \)-grid. Additionally, the number of parallel threads can be greater than \( \lambda \) (large MPI systems or GPU). Thus the need to create the TDSE algorithm that is parallelized in the \( r \)-direction. Such approach also allows for a smaller communication overhead than the parallelization in the \( x \)-direction.

Extending the method to two and three dimensions is an ongoing project.