



# GPGPU Parallelization of Select Quantum Monte Carlo Algorithms

**Caroline Marcks**  
Tufts University  
caroline.marcks@tufts.edu

**Dillon Skeehan**  
University of Rochester  
dillon.skeehan@rochester.edu

## ABSTRACT:

Quantum Monte Carlo is the process by which the Schrödinger equation can be solved, therefore allowing scientists to describe particles on a fundamental level. Given a system of particles and a time step, one can compute an update to that set. After millions of updates, the Schrödinger equation converges and is a model for that body of particles. Because of the computational complexity of each update and the number of updates required before convergence, this is quite slow. The processing power of parallel computation on graphics processing units can therefore be harnessed to improve timing. This presentation delves into two components of the Quantum Monte Carlo problem, namely updating the Slater matrix's inverse and interparticle distances.

## UPDATING THE SLATER MATRIX INVERSE:

If the position of a particle in the system is changed, the Slater matrix and its inverse must be updated. Stable matrix inversion takes  $O(n^3)$  time where  $n$  is the dimension of the matrix. Inverse updating, however, can be completed in  $O(n^2)$  time with the Sherman Morrison Formula. Given a column vector ( $u$ ) and a row vector ( $v^T$ ), the update of an inverse can be computed as shown in figure 1:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u}$$

Figure 1. The Sherman Morrison Formula

## METHODS OF PARALLELIZATION:

**Non-Replica parallelization:** individually parallelize and optimize the computations for computing one inverse's update at a time.

Major challenge: Synchronization/communication in the completion of steps across thread blocks

**Replica Parallelization:** parallelize to run multiple inverse updates (12 found to be optimal) at once.

Major challenge: Memory accessing time and coalescence

Figure 2. CPU vs GPU Inversion and Update Timings

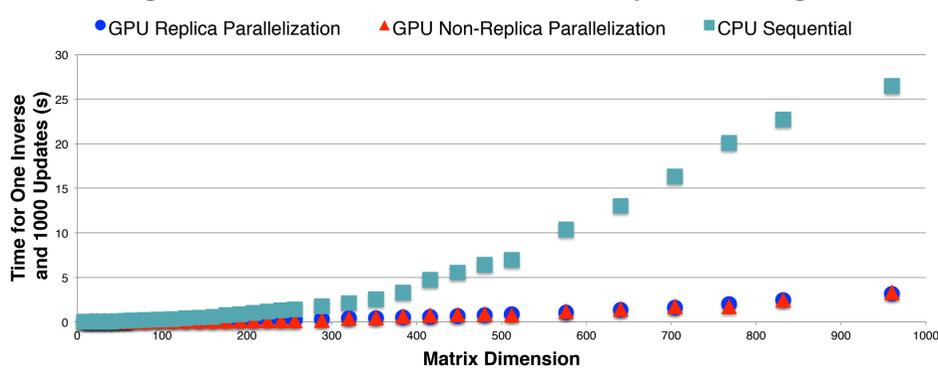
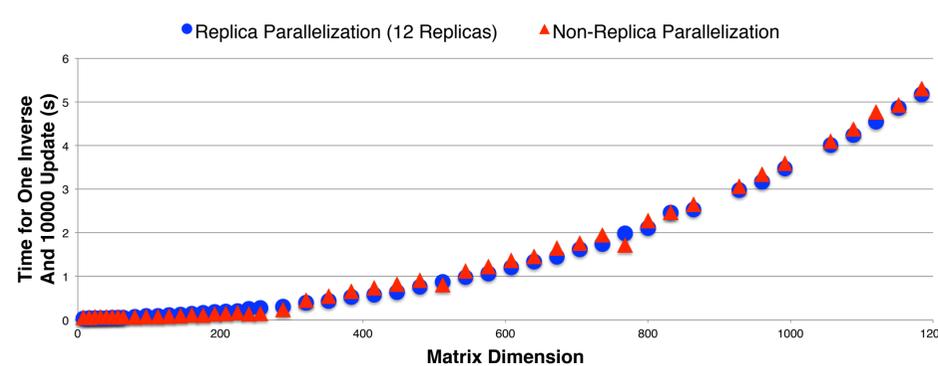


Figure 3. UP CLOSE: Replica vs Non-Replica GPU Parallelization



From these graphs, once can see that the GPU is a viable tool for speeding up sequential matrix inversion and updating. However, the distinction between replica and non replica methods of parallelization is inconclusive. For a 1000 by 1000 matrix, the GPU methods are roughly 8 times faster than the sequential CPU method.

## UPDATING THE INTERPARTICLE DISTANCES:

The system to which the particles are confined is assumed to be an arbitrary parallelepiped under periodic boundary conditions (figure 4). When the position of a particle is changed within the system, that particle's distance to its closest neighbors in the image must be recomputed.

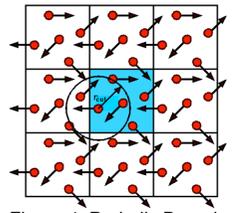


Figure 4. Periodic Boundary Conditions

For this component, we explored two different algorithms, a tree search vs. a linear search, both sequentially and in parallel.

**Linear Search:** Simplest approach. Loops through entire list of images and selects the closest image of each particle

Pros: No setup time, trivial to parallelize

Cons: Inefficient use of resources, fixed (long) running time

**Tree Search:** Complex approach. Using binary space partitioning (specifically an octree), each image is assigned to a cell. The cells are then mapped to a linear ordering and a tree is constructed from multiple linear orderings (figure 5).

Pros: on average faster search time

Cons: requires setup of data structure, not trivial to parallelize

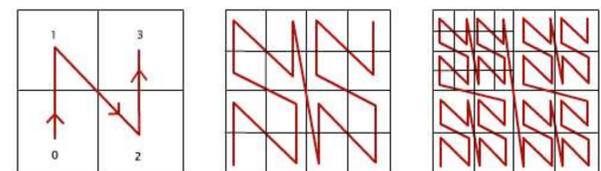
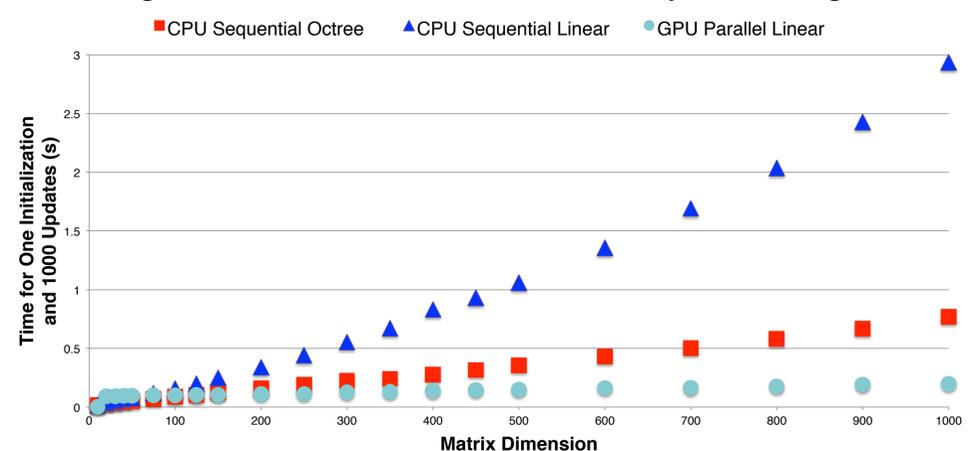


Figure 5. Linear Orderings from Recursive Morton Curve

Figure 6. CPU vs GPU Initialization and Update Timings



	CPU Linear vs CPU Octree	CPU Linear vs GPU Linear	CPU Octree vs GPU Linear
Crossover Points (# elements)	20	75	150
Speed Ups	4x	15x	4x

Figure 7. Implementation Comparisons

The GPU Parallel octree implementation has not yet been completed, however it is already clear that the GPU parallelization provides large performance gains as seen in figure 7. It is expected that the parallel octree will provide similar gains.

As one can see, the GPU is an appropriate tool for parallelizing certain portions of the Quantum Monte Carlo algorithms. It's strengths lie in tasks that are data parallel and involve limited communication. The Sherman Morrison formula falls easily into this category, and the interparticle distance calculations appear to as well. However, future work is still needed in both areas. In the matrix inverse update algorithm, more work needs to be done on memory coalescence in replica parallelization, as even larger gains can be expected from such work. On the parallel interparticle distance calculations, the implementation of the parallel octree must be completed along with a parallel sort.

## CITATIONS:

- Bartlett, M. S. (1951). An inverse matrix adjustment arising in discriminant analysis. *The Annals of Mathematical Statistics*, 22(1), 107-111. doi:10.1214/aoms/117729698
- Prekshu Ajmera, Rhushabh Goradia, Sharat Chandran, Srinivas Aluru. (2008). "Fast, Parallel, GPU-based Space Filling Curves and Octrees" Department of Computer Science & Engineering, IIT Bombay. Symposium on Interactive 3D Graphics and Games.
- (n.d.). Retrieved from <http://www.compsoc.man.ac.uk/~lucky/Democritus/Theory/psc-mi.html>
- (n.d.). peano curve. [Web Graphic]. Retrieved from [http://www.gitta.info/SpatPartitio/en/html/RegDecomp\\_learningObject3.html](http://www.gitta.info/SpatPartitio/en/html/RegDecomp_learningObject3.html)

## ACKNOWLEDGEMENTS:

- Dr. Craig Zilles
- Jill Peckham
- Dr. Lucas Wagner
- NSF Grant DMS-1024936

## HARDWARE:

- 32 GB RAM
- 2 Intel Xeon CPU
- E5-2640 @ 2.5 GHz
- GeForce GTX 670

## COMPUTING LIBRARIES:

- BLAS
- CUBLAS
- CURAND
- BOOST