Exploring QR Factorization on GPU for Quantum Monte Carlo Simulation

Tyler McDaniel (UNC Asheville), Ming Wong (University of South Carolina)

Mentors: Ed D'Azevedo, Ying Wai Li, Kwai Wong

Abstract

QMCPACK is open-source scientific software designed to perform Quantum Monte Carlo simulations, which are first principles methods for determining the properties of the electronic structure of atoms, molecules, and solids. One major objective is to find the ground state for a physical system. Our task is to investigate possible alternatives to the existing method in QMCPACK for evaluating single-particle updates to a system's electron configuration by improving the computational efficiency and numerical stability of the algorithms.

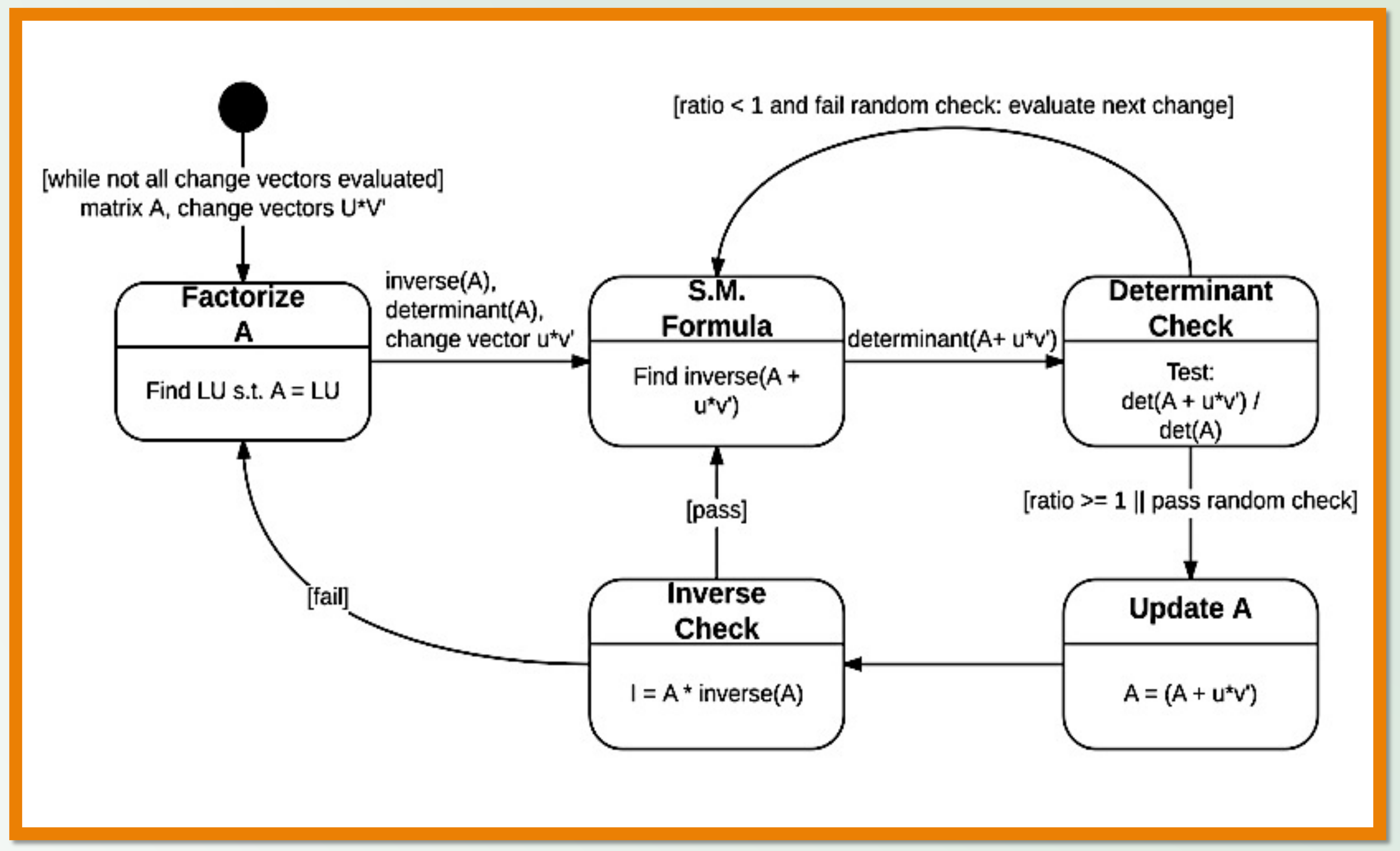
Background

Slater Determinant

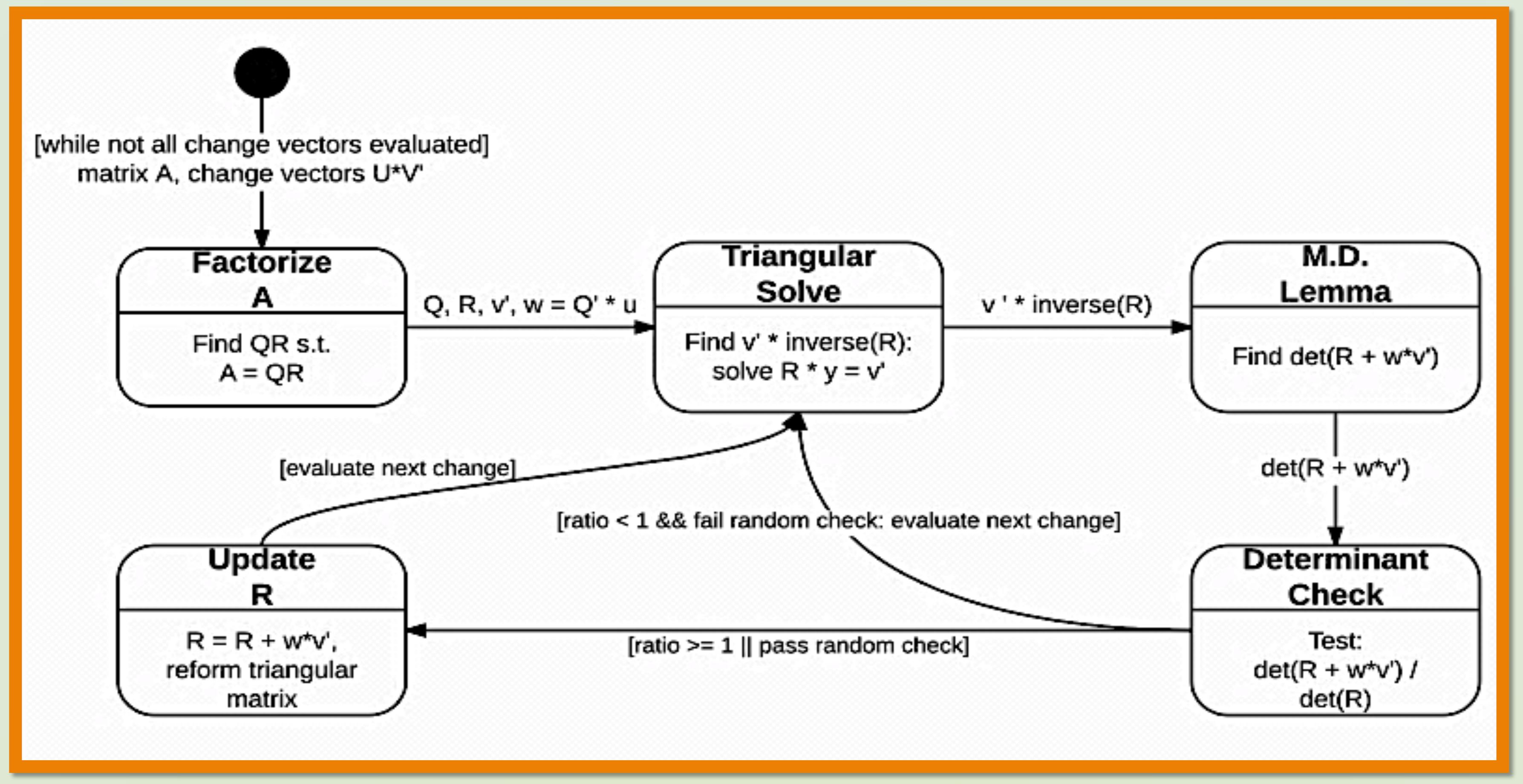
$$\Psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \chi_1(\mathbf{x}_1) & \chi_2(\mathbf{x}_1) & \cdots & \chi_N(\mathbf{x}_1) \\ \chi_1(\mathbf{x}_2) & \chi_2(\mathbf{x}_2) & \cdots & \chi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_1(\mathbf{x}_N) & \chi_2(\mathbf{x}_N) & \cdots & \chi_N(\mathbf{x}_N) \end{vmatrix}$$

- The Slater Determinant is a way of expressing the many-particle wave function for a system of electrons (or other fermions) with anti-symmetry.
- Each $\chi_{1,2\cdots n}$ represents the wave function for a single particle.
- When two particles' positions are exchanged, two rows are switched in the above matrix, which changes the sign of the determinant and satisfies the anti-symmetric property.

QMCPACK Existing Implementation



Proposed Implementation



Utilizing QR Factorization

- Provides more numerical stability
- Less sensitive to ill-conditioned matrices than LU factorization
- Obviates the need to periodically recalculate the inverse of matrix A from scratch

$$A = QR, \quad Q = \begin{pmatrix} q_1 & \ldots & q_n \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & & 0 & r_{nn} \end{pmatrix}$$

Note that Q is orthonormal and R is upper triangular.

Rank-k Update

- Contiguous accepted change columns can be stored in a sub-matrix, delaying the need to explicitly update matrix R.
- The estimated determinant from this continuous sequence of changes can be calculated using the matrix determinant lemma.

Matrix Determinant Lemma Rank-1 Case

Suppose A is an invertible square matrix and u, v are column vectors.

$$\det(\mathbf{A} + \mathbf{u}\mathbf{v}^{\mathrm{T}}) = (1 + \mathbf{v}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{u}) \det(\mathbf{A}).$$

Matrix Determinant Lemma Rank-k Case

Suppose A is an invertible n-by-n matrix and U, V are n-by-m matrices.

$$\det(\mathbf{A} + \mathbf{U}\mathbf{V}^{\mathrm{T}}) = \det(\mathbf{I_m} + \mathbf{V}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{U}) \det(\mathbf{A}).$$

Update QR Techniques

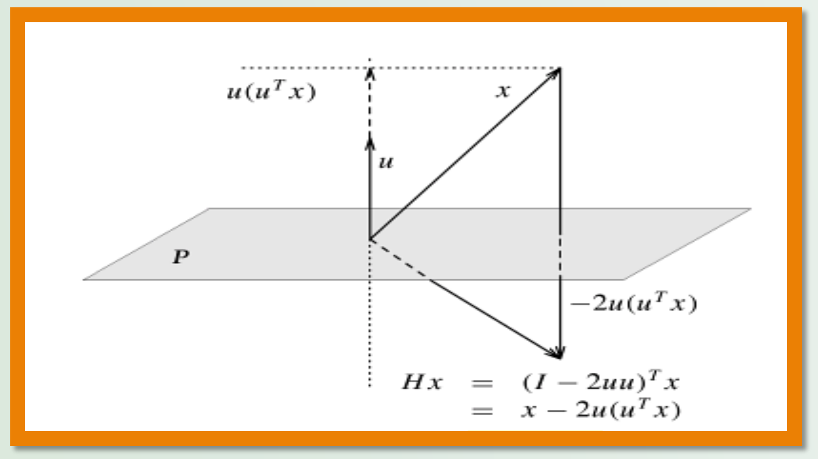
Given's Rotations

A Given's Rotation matrix, where c is cos(θ) and s is sin(θ), rotates the x_i and x_j of the element in a matrix by θ.

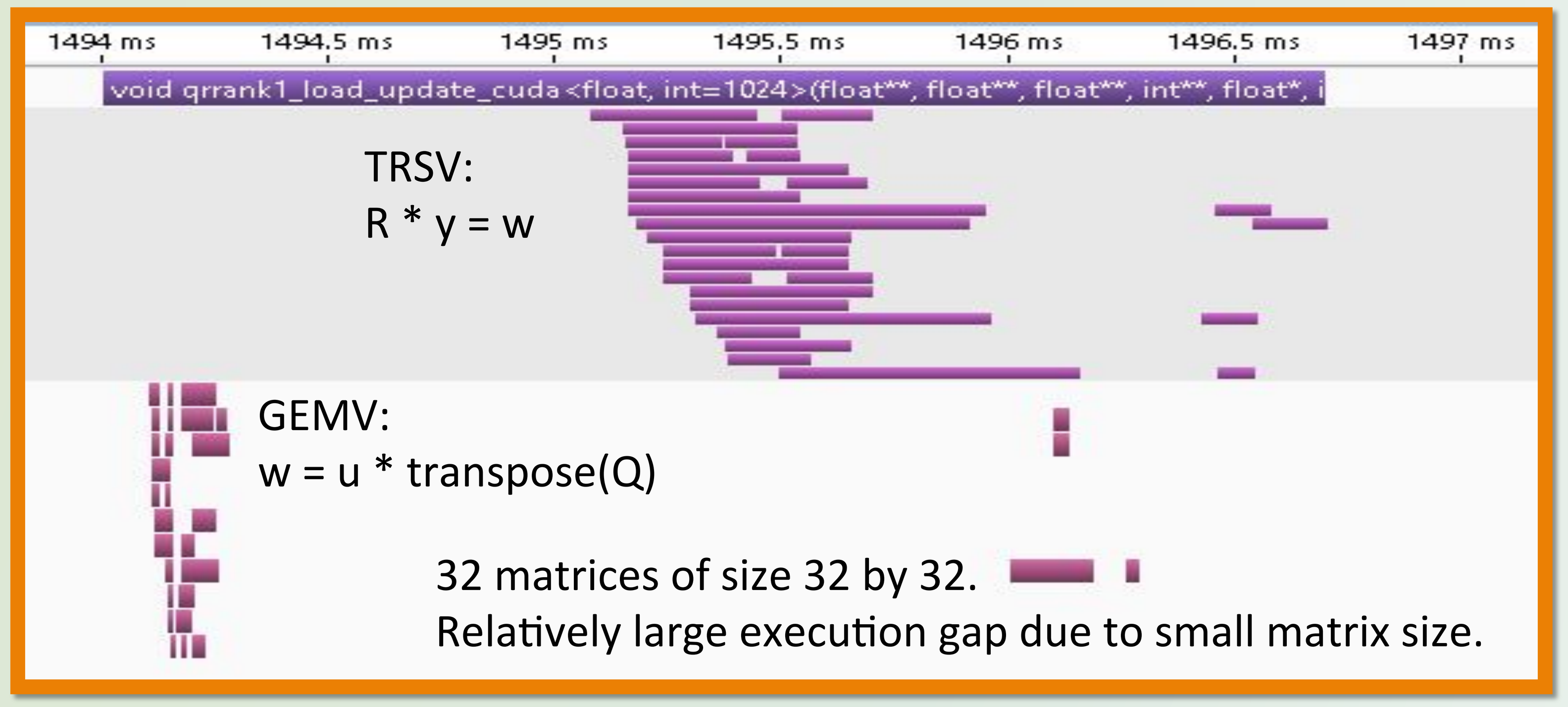
$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}^T \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}, \quad r = \sqrt{a^2 + b^2}.$$

Householder Transformations

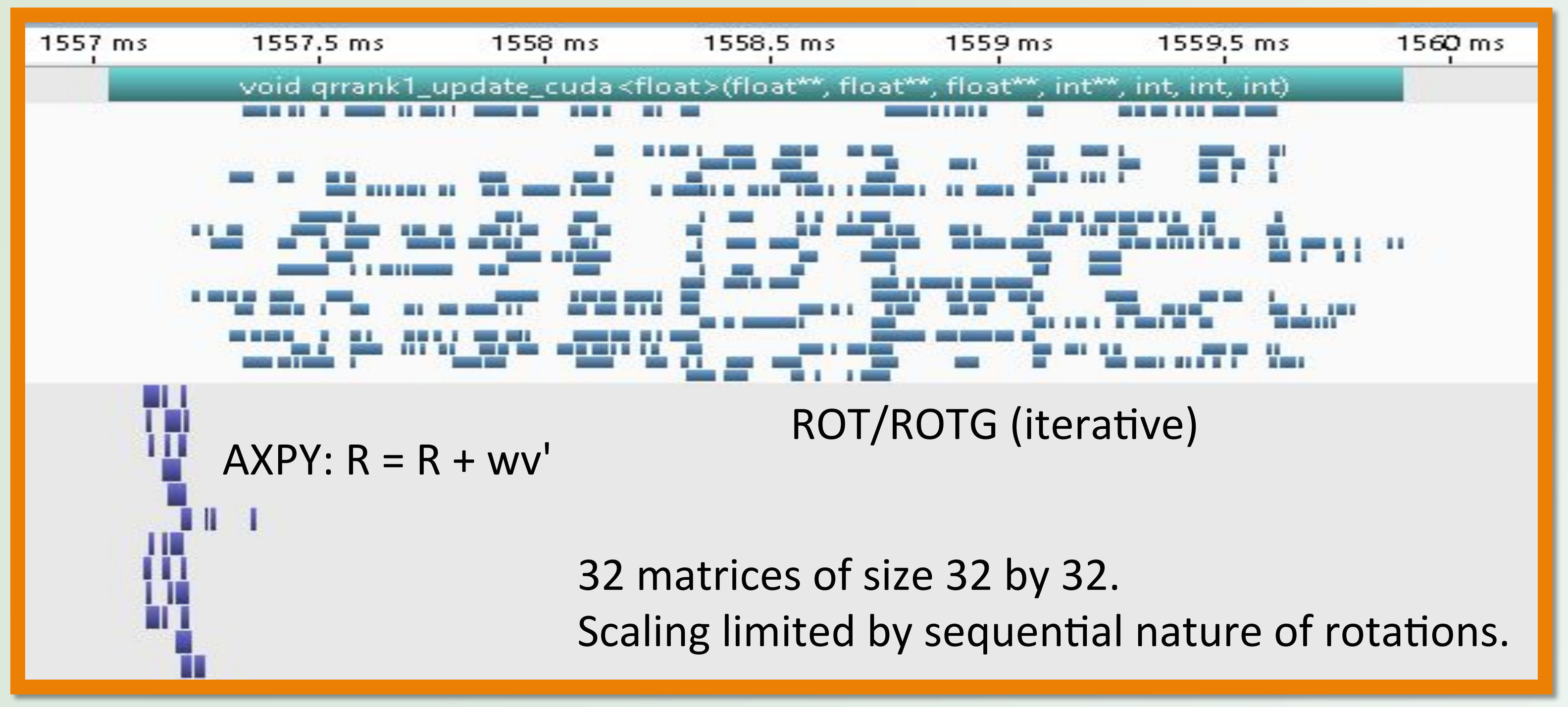
A Householder Transformation is also known as an elementary reflector.



GPU Implementation (Rank-1)



Evaluate Kernel Result: det(R + w * v') = (y[k] + 1) * det(R)



Update Kernel Result: Upper triangular R, A = Q * R preserved

Future Work

- Addressing issues in scalability and parallelization
- GPU implementation of rank-k algorithm

Acknowledgements: Funded by a National Science Foundation REU program, CSURE 2015 at UTK/ORNL.