Parallelization of Non-equilibrium Green's Function (NEGF) Simulations

Yang Dan

Department of Materials Science and Engineering,
University of Illinois at Urbana-Champaign
yangdan2@illinois.edu

Abstract

In physics and nanoscale technology, non-equilibrium Green's function (NEGF) is a widely used tool for quantum transport simulation, but it is usually expensive for several reasons, among which a huge amount of computational resource is consumed by the recursive Green's function (RGF) algorithm that makes up the central part of the NEGF method. The RGF algorithm focuses on solving a subset of the entries in the inverse of large sparse Hermitian matrices repeatedly. In this research a way of parallelizing the RGF algorithm is implemented, tested and analyzed, in the hope of improving the applicability of NEGF method to more realistic engineering problems as a whole.

1. Motivation and Background

- NEGF makes excellent predictions for important physical quantities related to quantum transport, such as the electron density of states (DOS) and the transmission spectra of an Field Effect Transistor(FET). But the simulation is usually expensive in time and memory.
- Simulation based on the NEGF equation derived from Schrödinger's equation

$$\mathbf{A} \cdot \mathbf{G}(E) = \mathbf{I} \tag{1}$$

where

$$\mathbf{A} = E\mathbf{I} - \mathbf{H} - \mathbf{\Sigma_L}(E) - \mathbf{\Sigma_R}(E) \tag{2}$$

$\mathbf{G}(E)$: the Green's function;
$\mathbf{H}$: the Hamiltonian matrix, **block tridiagonal** under tight-bounding approximation;
$\mathbf{\Sigma_L}(E)$: self energy matrix of the left reservoir, **only nonzero at the upper left corner**;
$\mathbf{\Sigma_R}(E)$: self energy matrix of the right reservoir, **only nonzero at the lower right corner**;
E: energy; I: identity matrix.

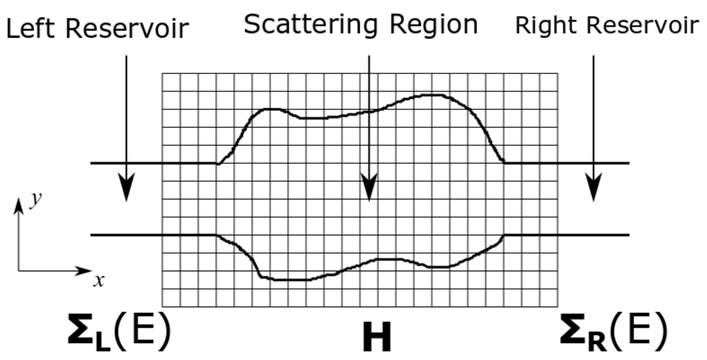

Figure 1: *A typical model of a 2-D transport process where particles scatter in a scattering region between two reservoirs. The simulation is done on a 2-D uniform grid.*

- N slices along the x direction and M slices along the y direction, yielding **block tridiagonal matrix** $\mathbf{A}$ of size $N \times N$ in blocks, with each block $\mathbf{A_{ij}}$ of size $M \times M$ and being dense

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & 0 & \cdots & 0 & 0 & 0 \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} & \cdots & 0 & 0 & 0 \\ 0 & \mathbf{A}_{32} & \mathbf{A}_{33} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \mathbf{A}_{N-2,N-2} & \mathbf{A}_{N-2,N-1} & 0 \\ 0 & 0 & 0 & \cdots & \mathbf{A}_{N-1,N-2} & \mathbf{A}_{N-1,N-1} & \mathbf{A}_{N-1,N} \\ 0 & 0 & 0 & \cdots & 0 & \mathbf{A}_{N,N-1} & \mathbf{A}_{N,N} \end{bmatrix} \tag{3}$$

- Matrix $\mathbf{A}$ is often **very large and sparse, but Hermitian**. The problem is reduced to calculating $\mathbf{A}^{-1}$. But fortunately it is not necessary to inverse the whole matrix to get the desired physical quantities. For example for the transmission spectra calculation

$$T(E) = \text{Trace}[\gamma_R(E)\mathbf{A}^{-1}_{1,N}\gamma_L(E)\mathbf{A}^{\dagger}_{1,N}] \tag{4}$$

- $\gamma_R(E)$ and $\gamma_L(E)$ can be easily obtained from the self-energy matrices.
- The key problem lies in calculating $\mathbf{A}^{-1}_{1,N}$

2. Theory and Algorithm

2.1 Domain Decomposition, Reordering, and Calculating Schur's Complement

- Different sequences of two manipulations, reordering or inversion of matrix $\mathbf{A}$, gives the same result.

$$(\mathbf{PAP^T})^{-1} = (\mathbf{P^T})^{-1}\mathbf{A}^{-1}\mathbf{P}^{-1} = \mathbf{PA}^{-1}\mathbf{P^T} \tag{5}$$

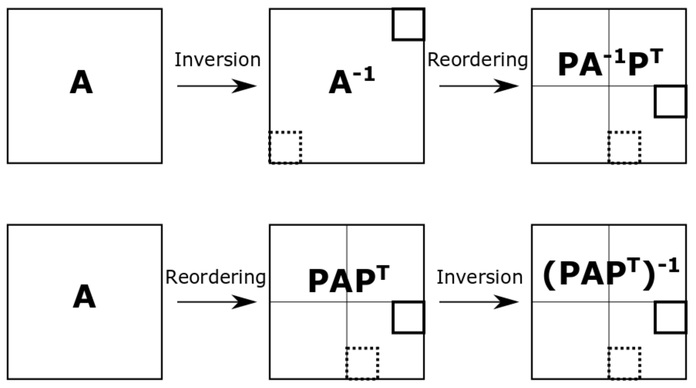

Figure 2: *Different sequences of reordering and inversion of matrix $\mathbf{A}$ keeps the position of the desire block*

- Reorder the rows and columns of matrix $\mathbf{A}$ so that it has a 2×2 block structure

$$\mathbf{PAP^T} = \begin{bmatrix} \mathbf{A}^{\alpha\alpha} & \mathbf{A}^{\alpha\beta} \\ \mathbf{A}^{\beta\alpha} & \mathbf{A}^{\beta\beta} \end{bmatrix} \tag{6}$$

and the desired block appears in the upper right block of $\mathbf{A}^{\beta\beta}$

- The inversion of Equation (6) yields

$$(\mathbf{PAP^T})^{-1} = \begin{bmatrix} \times & \times \\ \times & \mathbf{S}^{-1} \end{bmatrix} \tag{7}$$

where $\mathbf{S}$ is the Schur's complement and the upper right block of $\mathbf{S}^{-1}$ is the desired $\mathbf{A}^{-1}_{1,N}$

$$\mathbf{S} = \mathbf{A}^{\beta\beta} - \mathbf{A}^{\beta\alpha}(\mathbf{A}^{\alpha\alpha})^{-1}\mathbf{A}^{\alpha\beta} \tag{8}$$

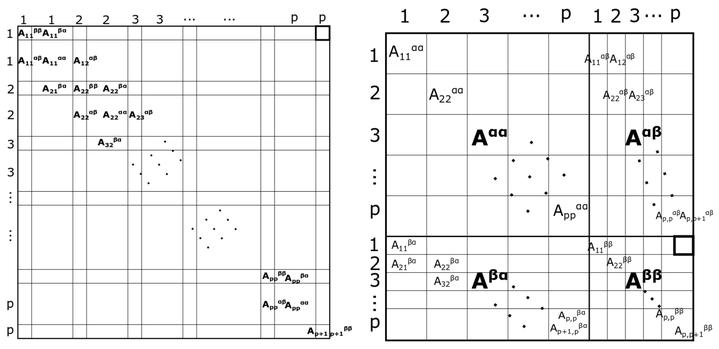

Figure 3: *The structure of matrix $\mathbf{A}$ (a) before and (b) after reordering via row and column permutations. Numbers marked by the matrices are indices of processors that own the corresponding parts of the matrices. Nonzero blocks are shown in the matrices with the subscript indicating the original position and superscript indicating the position after permutations. The desired block, whose position changes after reordering, is marked in bold.*

- After reordering, the Schur's complement $\mathbf{S}$ is calculated using block Gaussian elimination that eliminates $\mathbf{A}^{\beta\alpha}$ with $\mathbf{A}^{\alpha\alpha}$. Each processor updates the blocks it owns with communications of the boundary slices with its neighbors. After completion, $\mathbf{S}$ is block tridiagonal and Hermitian

$$\mathbf{S} = \begin{bmatrix} \mathbf{A}^{\beta\beta}_{\mathbf{11}} & \mathbf{A}^{\beta\beta}_{\mathbf{12}} & 0 & \cdots & 0 & 0 & 0 \\ \mathbf{A}^{\beta\beta\dagger}_{\mathbf{12}} & \mathbf{A}^{\beta\beta}_{\mathbf{22}} & \mathbf{A}^{\beta\beta}_{\mathbf{23}} & \cdots & 0 & 0 & 0 \\ 0 & \mathbf{A}^{\beta\beta\dagger}_{\mathbf{23}} & \mathbf{A}^{\beta\beta}_{\mathbf{33}} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & \mathbf{A}^{\beta\beta}_{\mathbf{p-1,p-1}} & \mathbf{A}^{\beta\beta}_{\mathbf{p-1,p}} & 0 \\ 0 & 0 & 0 & \cdots & \mathbf{A}^{\beta\beta\dagger}_{\mathbf{p-1,p}} & \mathbf{A}^{\beta\beta}_{\mathbf{p,p}} & \mathbf{A}^{\beta\beta}_{\mathbf{p,p+1}} \\ 0 & 0 & 0 & \cdots & 0 & \mathbf{A}^{\beta\beta\dagger}_{\mathbf{p,p+1}} & \mathbf{A}^{\beta\beta}_{\mathbf{p+1,p+1}} \end{bmatrix} \tag{9}$$

2.2 Reducing Schur's Complement with Cyclic Reduction Algorithm

- The size of the Schur's complement $\mathbf{S}$ can be reduced to a 3×3 block matrix which is small enough to inverse directly to get the desired block. The reduction is done using cyclic reduction algorithm which reduces the size of $\mathbf{S}$ by half at each step.

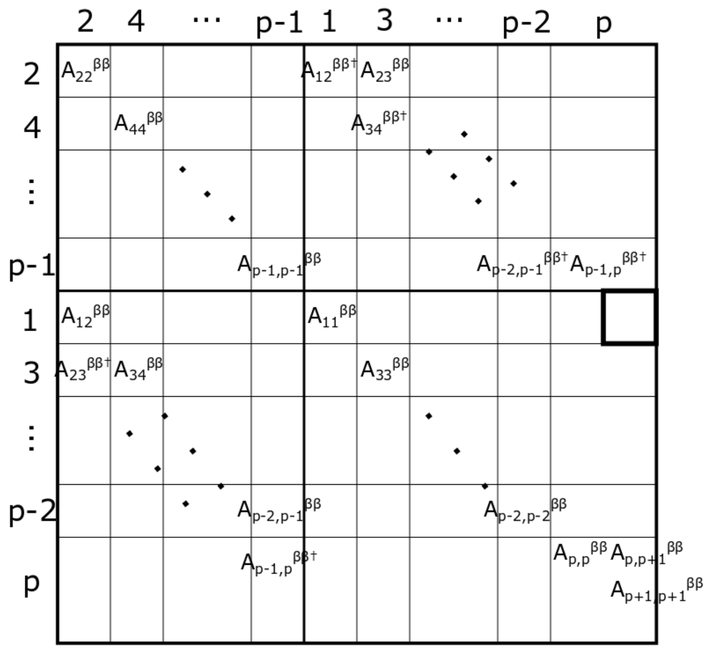

Figure 4: *Cyclic reduction of a Schur's complement matrix of block size $(p+1) \times (p+1)$. The size of Schur's complement $\mathbf{S}$ is reduced by half at every step. The desired block stays at the upper right corner of $\mathbf{S}$.*

3. Preliminary Results (by Dec. 13)

- Strong scaling experiments have been performed on UIUC campus cluster, with $N = 200$ and $M = 100$.

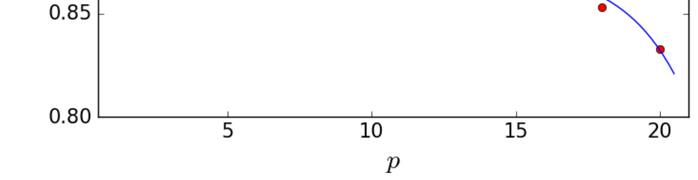

Figure 5: *Efficiency E_p as a function of the number of processors p. The dots indicate data acquired from the experiment and the line is fitted from the experimental data.*

References

[1] P.S. Drouvelis *et al*, Parallel implementation of the recursive Greens function method, *Journal of Computational Physics*, Vol 215, Issue 2, 2006, 741-756.

[2] Kuzmin A. *et al*, Fast Methods for Computing Selected Elements of the Greens Function in Massively Parallel Nanoelectronic Device Simulations. *Lecture Notes in Computer Science*, vol 8097. Springer, Berlin, Heidelberg

[3] Lin, L., Lu, L., Ying, J.: Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems. Comm. Math. Sci. 7, 755777 (2009)

[4] Lin, L., Yang, C.: Selinv - an algorithm for selected inversion of a sparse symmetric matrix. ACM Trans. on Math. Software 37 (2011)