11th International Workshop on Computational Electronics
TU Wien, 25-27 May 2006

Fast Inverse using Nested Dissection for the Non Equilibrium Green's Function

S. Li and E. Darve
Institute for Computational and Mathematical Engineering, Stanford University
496 Lomita Mall, Durand building, Room 265, Stanford, CA 94305-4040, U.S.A.
e-mail: darve@stanford.edu

INTRODUCTION

In recent years, nanoscale MOS transistors as well as nanowires and molecular electronic devices have been actively studied [1]. It is now possible to manufacture transistors with channel lengths as small as 10 nm and below. At these scales, quantum effects such as tunneling are significant and have to be included in the simulation model. Despite the fact that transport issues for nano-transistors, nanowires and molecular electronic devices are very different from one another, they can be treated with the common formalism provided by the Non Equilibrium Green's Function. This approach is based on the coupled solution of the Schrödinger and Poisson equations. Our algorithm focuses on the computationally most expensive part which is the solution of the Schrödinger equation for the electron density, which is then used in the Poisson equation. After certain key simplifications [4], the problem is reduced to computing the diagonal elements of the matrix $\boldsymbol{G} = [E\boldsymbol{I} - \boldsymbol{H} - \boldsymbol{\Sigma}]^{-1}$ (retarded Green's function) and $\boldsymbol{G}^< = \boldsymbol{G}\boldsymbol{\Sigma}^<\boldsymbol{G}^\dagger$ (less-than Green's function), where the energy level E, Hamiltonian matrix $\boldsymbol{H}$, and self energies $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}^<$ (see Svizhenko [2] for those notations) are in this work considered to be given. ($\dagger$ denotes the transpose conjugate of a matrix.) We now describe how to compute the diagonal of $\boldsymbol{G}$ and $\boldsymbol{G}^<$ efficiently. Our algorithm can be derived for devices of arbitrary geometry and for arbitrary boundary conditions; however for simplicity we will focus in this paper on 2D rectangular devices (a typical geometry used for modeling MOSFETs [2]).

DESCRIPTION OF THE ALGORITHM

Let's first consider $\boldsymbol{G}$ (the extension to $\boldsymbol{G}^<$ is presented below). The basic idea of our algorithm is to perform many LU factorizations on the given matrix to compute the diagonal elements of its inverse. By taking advantage of the sparsity of the given matrix, LU factorizations can be performed very efficiently. Once the LU factorization is complete, we can easily compute the last entry on the diagonal of the inverse: for the $n \times n$ matrix $\boldsymbol{G}^{-1} = \boldsymbol{L}\boldsymbol{U}$, we have $\boldsymbol{G}_{nn} = 1/\boldsymbol{U}_{nn}$. Although we can only compute $\boldsymbol{G}_{nn}$ in this way, we can choose any node and reorder $\boldsymbol{G}^{-1}$ to make the node correspond to the (n, n) entry of the reordered matrix and thus compute all the diagonal elements of $\boldsymbol{G}$.

If we have to perform a full LU factorization for each of the n reordered matrices, the algorithm will not be computationally efficient even though each LU factorization is very fast. However, if we reorder those matrices properly, many partial factorizations for different reordered matrices turn out to be identical. We can store the results of those partial factorizations in a binary tree (its structure follows the nested dissection procedure of George [5]) and reuse them many times thereby reducing considerably the computational cost. As a result, the total cost of performing the LU factorizations on all the n reordered matrices is of the same order as performing the LU factorization on one matrix and thus computing all the diagonal elements of $\boldsymbol{G}$ is very efficient. This is the main merit of our algorithm.

For a square mesh of size $N \times N$, the computational cost dominates at the top level of the binary tree. The cost of performing partial factorization at the top level is $O(N^3)$ and we showed that the total cost turns out to be also $O(N^3)$. For rectangle meshes of size $N_x \times N_y$ with $N_x < N_y$, the cost is $O(N_x^2 N_y)$.

We can also extend the above idea to computing

$G^<$. Define $R \overset{def}{=} L^{-1}\Sigma^< L^{-\dagger} = UG^<U^\dagger$, then we have $G^<_{nn} = R_{nn}/|U_{nn}|^2$. In principle, R is a dense matrix which is very expensive to calculate. However, by taking advantage of the sparsity of $\Sigma^<$, we created an algorithm to compute R_{nn} whose cost is of the same order as computing U_{nn}. By using a re-ordering strategy and storage of intermediate steps similar to the ones described above, we derived an algorithm to calculate the diagonal of $G^<$ in $O(N_x^2 N_y)$ steps.

Unlike the computing cost that dominates at the top level, the memory cost is about the same ($\propto N_x\, N_y$) at each level of the tree. We have $\log(N_xN_y)$ levels and thus the total memory cost is $O(N_xN_y\log(N_xN_y))$. This is asymptotically better than the algorithm given by Svizhenko *et al.* [2] since the memory cost of their algorithm is $O(N_x^2N_y)$.

Results

We made comparisons between our algorithms and the algorithm given by Svizhenko [2] (see Lake [3] for an earlier version in 1D). Their algorithm is based on computing all the diagonal blocks of G and $G^<$ using a forward and backward recurrence along the y axis of the device. The cost of their algorithm is $O(N_x^3N_y)$ since the inverse of N_y matrices of size N_x needs to be computed. Fig. 1 shows a comparison of running time between the two algorithms. We can see that our algorithm uses much less time when the mesh size exceeds 100×100 and scales much better overall. Fig. 2 shows the comparison of memory cost between the two algorithms. We can see that the memory cost of the two algorithms is about the same (because of the larger constant factor in our algorithm) but the memory cost of our algorithm increases slower than the other algorithm so it is asymptotically better.

Discussion

The constant factors in the running time and the memory cost of our algorithm are expected to improve by about 50%. This will be achieved by exploiting more efficiently the sparsity of G^{-1}.

The algorithm described here can be extended to arbitrary geometries. It is applicable to a large class of devices including nanotubes and nanowires.

References

[1] Y. Xue, S. Datta, and M. A. Ratner, “First-principles based matrix Green’s function approach to molecular electronic devices: general formalism,” *Chemical Physics* **281**(2/3), pp. 151-70, 2002

[2] A. Svizhenko, M. P. Anantram, T.R. Govindan, B. Biegel and R. Venugopal, “Two-dimensional quantum mechanical modeling of nanotransistors,” *Journal of Applied Physics* **91**(4), pp. 2343-2354, 2002

[3] R. Lake, G. Klimeck, R.C. Bowen, and D. Jovanovic, “Single and multiband modeling of quantum electron transport through layered semiconductor devices,” *Journal of Applied Physics* **81**(12), pp. 7845-69, 1997

[4] S. Datta, “Nanoscale device modeling: the Green’s function method,” *Superlattices and microstructures* **28**(4), pp. 253-278, 2000

[5] A. George, “Nested dissection of a regular finite element mesh,” *SIAM Journal on Numerical Analysis* **10**(2), pp. 345-63, 1973

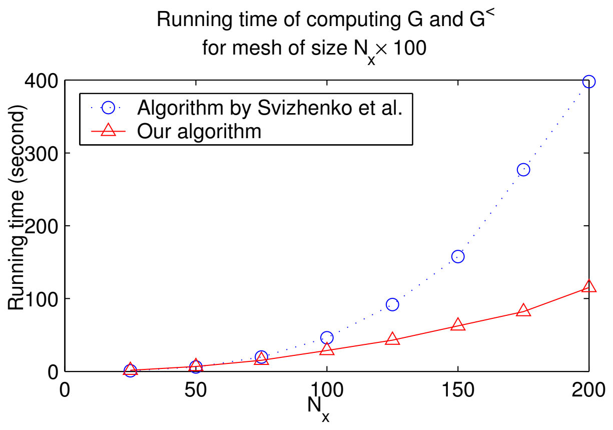

Fig. 1. Running time of our algorithm and the algorithm proposed by Svizhenko [2].

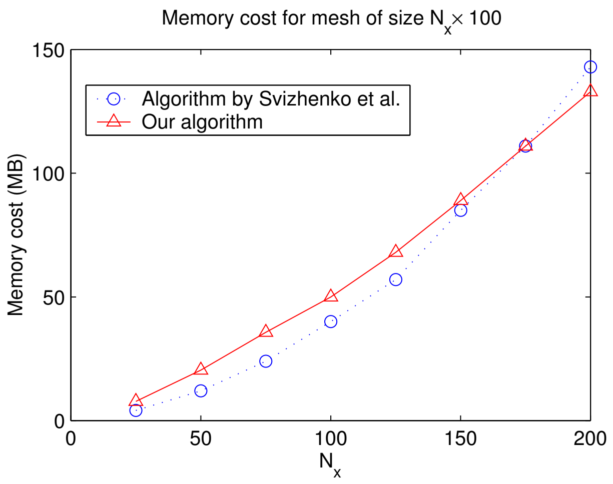

Fig. 2. Memory cost of our algorithm and the algorithm proposed by Svizhenko [2].