INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2013; **95**:587–607
Published online 26 June 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.4518

A nested dissection approach to modeling transport in nanodevices: Algorithms and applications

U. Hetmaniuk[1,*,†], Y. Zhao[2] and M. P. Anantram[2]

[1]*Department of Applied Mathematics, University of Washington, Box 353925, Seattle, WA 98195-3925, USA*
[2]*Department of Electrical Engineering, University of Washington, Box 352500, Seattle, WA 98195-2500, USA*

SUMMARY

Modeling nanoscale devices quantum mechanically is a computationally challenging problem where new methods to solve the underlying equations are in a dire need. In this paper, we present an approach to calculate the charge density in nanoscale devices, within the context of the nonequilibrium Green's function approach. Our approach exploits recent advances in using an established graph partitioning approach. The developed method has the capability to handle open boundary conditions that are represented by full self-energy matrices required for realistic modeling of nanoscale devices. Our method to calculate the electron density has a reduced complexity compared with the established recursive Green's function approach. As an example, we apply our algorithm to a quantum well superlattice and a carbon nanotube, which are represented by a continuum and tight binding Hamiltonian, respectively, and demonstrate significant speedup over the recursive method. Copyright © 2013 John Wiley & Sons, Ltd.

Received 17 January 2013; Accepted 2 April 2013

KEY WORDS: nanodevice; numerical simulation, device modeling; quantum transport; superlattice; nanomaterials; graphene; Green's functions; nanotechnology; nanotransistor; tunneling; design

1. INTRODUCTION

With the advent of smaller nano-electronic devices, where quantum mechanics is central to the device operation, and new nanomaterials, such as nanotubes, graphene, and nanowires, quantum mechanical simulations have become a necessity. The nonequilibrium Green's function (NEGF) method [1–3] has emerged as a powerful modeling approach for these nanodevices and nanomaterials. The NEGF method is based on the self-consistent coupling of Schrödinger and Poisson equations and is designed to capture electron scattering effects with phonons.

A typical NEGF-based simulation solves three Green's function equations,

$$\begin{cases} \mathbf{A}(E)\mathbf{G}^r(E) = \mathbf{I} \\ \mathbf{A}(E)\mathbf{G}^<(E) = \mathbf{\Sigma}^<(\mathbf{G}^r(E))^\dagger \\ \mathbf{A}(E)\mathbf{G}^>(E) = \mathbf{\Sigma}^>(\mathbf{G}^r(E))^\dagger \end{cases} \tag{1}$$

where the sparse matrix $\mathbf{A}$ is defined by

$$\mathbf{A} = E\mathbf{I} - \mathbf{H} - \mathbf{\Sigma}_L^r - \mathbf{\Sigma}_R^r - \mathbf{\Sigma}_{\text{Phonon}}^r \tag{2}$$

$\mathbf{G}^r(E)$ is called the retarded Green's function, describing local density of states and the propagation of electrons injected in the device, and $(\mathbf{G}^r(E))^\dagger$ its Hermitian conjugate. $\mathbf{G}^<(E)$, the lesser

*Correspondence to: U. Hetmaniuk, Department of Applied Mathematics, University of Washington, Box 353925, Seattle, WA 98195-3925, USA.
†E-mail: hetmaniu@uw.edu

Copyright © 2013 John Wiley & Sons, Ltd.

Green's function, represents the electron correlation function for energy level E; the diagonal elements of $\mathbf{G}^<(E)$ represent the electron density per unit energy. $\mathbf{G}^>(E)$, the greater Green's function, represents the hole correlation function for energy level E, which is proportional to the density of unoccupied states. $\mathbf{I}$ is the identity matrix, and $\mathbf{H}$ the system Hamiltonian. $\mathbf{\Sigma}_L^r$ and $\mathbf{\Sigma}_R^r$ represent the self energies due to left and right contact coupling, and $\mathbf{\Sigma}_{\text{Phonon}}^r$ corresponds to the self energy governing electron-phonon scattering. The matrix $\mathbf{\Sigma}^<$ corresponds to the lesser self energy, and the matrix $\mathbf{\Sigma}^>$ to the greater self energy. The Green functions are then incorporated in the coupling between the Schrödinger and the Poisson equations – see [1] for further details. The self-consistent solution of the Schrödinger and Poisson equations requires to solve (1) many times until consistency is achieved. It is well appreciated that the computationally intensive part of this calculation is solving (1) for the diagonal element of $\mathbf{G}^<$ (electron density) and $\mathbf{G}^>$ at all energies E. The objective of this paper is to present a new algorithm for accelerating the solution of (1).

The recursive Green's function (RGF) method [4–8] is an effective method, often used in practice, to compute $\mathbf{G}^r$, $\mathbf{G}^<$, and $\mathbf{G}^>$. For elongated devices, this approach remains the most efficient. Recently, the hierarchical Schur complement (HSC) method [9, 10] and the fast inverse using nested dissection (FIND) method [11, 12] exploit the nested dissection method [13] to exhibit a significant speedup. The key ideas behind these two algorithms are to partition the whole matrix $\mathbf{A}$ into blocks for an efficient block LU-factorization. This factorization is then re-used to fill in all diagonal blocks of the Green's function and some off-diagonal blocks in a specific order. These two algorithms are more efficient than RGF and have reduced the operation count down to a multiple of the cost for a block LU-factorization of a sparse matrix. Approximate methods that are efficient in the ballistic limit exist also. For example, the contact block reduction [14, 15] accelerates the computation by using a limited number of modes to represent the matrices. The focus of this paper is to develop an exact method that works in the presence of scattering.

For calculating the lesser Green's function $\mathbf{G}^<$, advanced algorithms for an arbitrary sparse matrix are still in their infancy.[‡] The RGF method remains an effective method, especially for elongated devices. The extension of FIND [12] for $\mathbf{G}^<$ yields a reduced asymptotic complexity, but the constant in front of the asymptotic term hinders the reduction in runtime. Li *et al.* [16] have recently proposed a modification of FIND for a significant speedup, but their partitioning of the matrix $\mathbf{A}$ requires some preprocessing. The contribution of this paper is to present an extension of the HSC method for calculating diagonal blocks for $\mathbf{G}^<$ with partitions from existing graph partitioning libraries (e.g., the package METIS [17]).

The rest of the paper is organized as follows. Section 2 will review exact methods and discuss differences between previous algorithms and proposed approach. Section 3 will give a mathematical description of the new algorithm. Finally, Section 4 describes numerical experiments to highlight its efficiency. The discussion and analysis in this paper will focus on two-dimensional problems, whereas three-dimensional problems will be illustrated in a future publication.

2. REVIEW OF EXACT METHODS

Consider a device that can be topologically broken down into layers as shown in Figure 1. For an effective mass Hamiltonian, the blue dots represent grid points of the discretized Green's function equation, whereas, in the case of a tight binding Hamiltonian, the blue dots represent orbitals on an atom.

When a five-point stencil is used for discretization, the resulting Hamiltonian is a symmetric block tridiagonal matrix as shown in Figure 2, where each diagonal block represents the Hamiltonian of a layer in Figure 1. The ith diagonal block of the Hamiltonian represents the coupling between grid points / atoms in a layer. The off-diagonal blocks to its left and right represents coupling to between layers i and $i-1$ and $i+1$, respectively. Both the diagonal and the off-diagonal blocks of the

[‡]An efficient algorithm for calculating $\mathbf{G}^<$ is also efficient for computing $\mathbf{G}^>$.

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

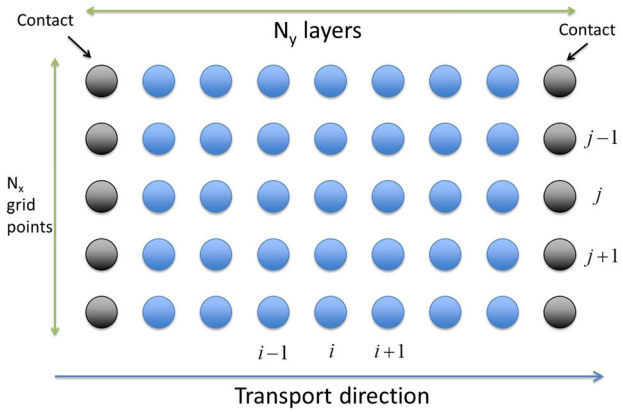


Figure 1. Nanodevice partitioned into N_y layers. Each layer contains N_x grid points.

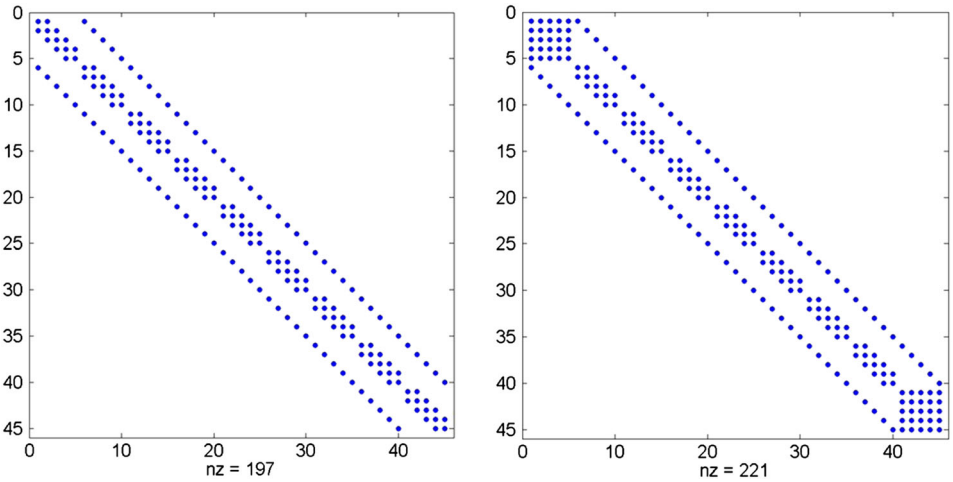


Figure 2. **H** (left) and **A** (right) matrix shape, nonzero entries are highlighted.

Hamiltonian are sparse in many examples where either an effective mass or a tight binding Hamiltonian represents the dynamics – examples of such device include silicon nanowires, nanotubes, and graphene.

The left and right coupling contacts (indicated in Figure 1) are two semi-infinite leads connected with the device, and infinite matrices represent their Hamiltonians. Their respective effect can be folded into layer 1 and layer N_y, resulting in dense blocks for the first and the N_yth diagonal blocks of the self-energy matrices $\mathbf{\Sigma}_L^r$ and $\mathbf{\Sigma}_R^r$. The resulting matrix structure of the matrix **A**, defined in (2), is shown in Figure 2. Note that the self-energy matrix $\mathbf{\Sigma}_{\text{Phonon}}^r$ is set to be diagonal at each interior grid point, which may arise because of electron–phonon interaction or any other interaction. Relaxing the requirement of diagonal $\mathbf{\Sigma}_{\text{Phonon}}^r$ to include more realistic models of scattering and solving equations (1) remains a challenge.

The most common approach to compute blocks of $\mathbf{G}^r$ and $\mathbf{G}^<$ is the RGF method [1, 18]. RGF is an algorithm composed of two passes to compute $\mathbf{G}^r$ and two passes to compute $\mathbf{G}^<$. In both cases, the passes are interpreted as follows:

(i) The first pass marches one layer at a time from *left to right* along the y-direction and, recursively, *folds* the effect of left layers into the current layer;
(ii) The second pass marches one layer at a time from *right to left* along the y-direction and, recursively, *extracts* the diagonal blocks and the nearest neighbor off-diagonal blocks for the final result.

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

Table I. Complexity of algorithms to compute diagonal blocks of $\mathbf{G}^r$.

Algorithm	Complexity when $N_x = N_y$	Complexity when $N_x < N_y$
RGF [18]	$\mathcal{O}\left(N_x^4\right)$	$\leqslant 10 N_x^3 N_y$
FIND [11, 19]	$\mathcal{O}\left(N_x^3\right)$	$\leqslant 143 N_x^2 N_y$
HSC [9, 19]	$\mathcal{O}\left(N_x^3\right)$	$\leqslant 87 N_x^2 N_y$

Numerically, it is essential to notice that the RGF method exploits the matrix sparsity of *only* at the block level, which means that it separates the whole problem into subproblems of *full* matrix operations. The complexity of this method is, at most, $10 N_x^3 N_y$ (when $N_x \leqslant N_y$).

To compute block entries of $\mathbf{G}^r$, two recent advances, namely FIND [11, 12] and HSC [9, 10], utilize the nested dissection method [13] to exhibit a significant speedup. These methods explicitly exploit the sparsity of $\mathbf{A}$ via a sparse block LU-factorization of the whole matrix and re-use this factorization to fill in all diagonal blocks of the Green's function and some off-diagonal blocks in a specific order. FIND and HSC have a strong mathematical component, and their physical interpretation is less obvious. The main difference between RGF and these methods is the replacement of *layers* of grid points organized along a specific direction with *arbitrarily shaped clusters* of grid points organized in a binary tree. Such choice allows to *fold* and to *extract* in any physical direction when following the vertical hierarchy of the binary tree. Further details about FIND and HSC can be found in their respective references. Table I summarizes the complexity of these three state-of-the-art algorithms when computing entries in $\mathbf{G}^r$.

Even though FIND [11] and HSC [9] have two distinct mathematical motivations, their runtime complexities exhibit the same dominating term. But the constants multiplying these asymptotic terms differ greatly. The runtime for FIND [11] contains a large constant because of the usage of thick boundaries between the clusters of grid points (or width-2 separators – a width-2 separator is a boundary between clusters that has a thickness of two grid points or two atoms). In addition, generating these clusters with thick boundaries is not compatible with most existing partitioning libraries. On the other hand, the HSC method use thin boundaries between clusters with a thickness of one grid point (or width-1 separators) and can directly exploit partitions from existing partitioning libraries. As a result, to compute diagonal blocks for $\mathbf{G}^r$, the HSC method [9, p. 758] is more efficient than FIND [11].

To compute diagonal entries for $\mathbf{G}^<$, only the RGF [1] and FIND [12] methods have been extended. Table II displays the runtime complexity for computing diagonal blocks of $\mathbf{G}^<$.

The extension of FIND [12] for $\mathbf{G}^<$ still uses thick boundaries that result in a large constant for the runtime and that are incompatible with existing partitioning libraries. Recently, Li *et al.* [16] have proposed an extension of FIND [11, 12] that enables thin boundaries (width-1 separators). This new algorithm yields a significant speedup over earlier versions of FIND [11, 12]. The mathematical motivation remains different from HSC. The clustering part needs to identify peripheral sets, an information that is not provided directly by most partitioning libraries.

The contribution of this paper is to present an extension of the HSC method [9] for calculating the diagonal blocks for $\mathbf{G}^<$. This extension uses thin boundaries (or width-1 separators) and is compatible with existing partitioning libraries – namely, the extension is combined with the graph partitioning package METIS [17]. The same extension computes efficiently diagonal blocks for $\mathbf{G}^>$ in a separate step. For the sake of conciseness, the rest of the paper is focused only on computing $\mathbf{G}^<$.

Table II. Complexity of algorithms to compute diagonal blocks of $\mathbf{G}^<$.

Algorithm	Complexity when $N_x = N_y$	Complexity when $N_x < N_y$
RGF [18]	$\leqslant 4 N_x^4$	$\mathcal{O}\left(N_x^3 N_y\right)$
FIND [12]	$\leqslant 457 N_x^3$	$\mathcal{O}\left(N_x^2 N_y\right)$

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

3. MATHEMATICAL DESCRIPTION OF THE ALGORITHM

In this section, a detailed mathematical description for the extension of HSC to compute blocks of $\mathbf{G}^<$ is given. The key ingredients are

(i) An efficient sparse block $\mathbf{LDL}^T$-factorization of $\mathbf{A}$. The block sparse factorization will gather grid points into arbitrarily shaped clusters (instead of layers, such as in RGF). Such choice allows to fold and to extract in any physical direction when eliminating entries in $\mathbf{A}$. The factorization yields formulas to calculate the diagonal blocks and off-diagonal blocks for $\mathbf{G}^r$ and $\mathbf{G}^<$. Exploiting the resulting algebraic relations results in an algorithm with a cost significantly smaller than the full inversion of matrix $\mathbf{A}$.

(ii) An appropriate order of operations. The cost of a matrix multiplication **BCD** depends on the order of operations. When $\mathbf{B}$ is $m\times p$, $\mathbf{C}$ is $p\times k$, and $\mathbf{D}$ is $k\times n$, (**BC**) **D** costs $2mk(n+p)$ operations, and **B** (**CD**) costs $2np(m+k)$. The order of operations can have a large effect when multiplying series of matrices together (which is the case for computing entries of $\mathbf{G}^r$ and $\mathbf{G}^<$). Furthermore, when working with sparse matrices, one order of operations may preserve sparsity, whereas another may not.

First, a simple description with three clusters is given. Then, the approach is extended to an arbitrary number of levels and a multilevel binary tree.

3.1. *Description for a simple case*

The basic idea is to partition the nanodevice into three disjoint regions (L, R, S) – see Figure 3.

Such a partition is easily obtained via the nested dissection, introduced by George [13]. Nested dissection divides the system into two disconnected sets and an interface, called the separator S. With this partition, the matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{LL} & \mathbf{0} & \mathbf{A}_{LS} \\ \mathbf{0} & \mathbf{A}_{RR} & \mathbf{A}_{RS} \\ \mathbf{A}_{LS}^T & \mathbf{A}_{RS}^T & \mathbf{A}_{SS} \end{bmatrix}.$$

Note that matrix $\mathbf{A}$ is typically complex symmetric. The block $\mathbf{LDL}^T$-factorization of $\mathbf{A}$ is

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{LS}^T\mathbf{A}_{LL}^{-1} & \mathbf{A}_{RS}^T\mathbf{A}_{RR}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{LL} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{RR} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \widehat{\mathbf{A}}_{SS} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{A}_{LL}^{-1}\mathbf{A}_{LS} \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_{RR}^{-1}\mathbf{A}_{RS} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$$

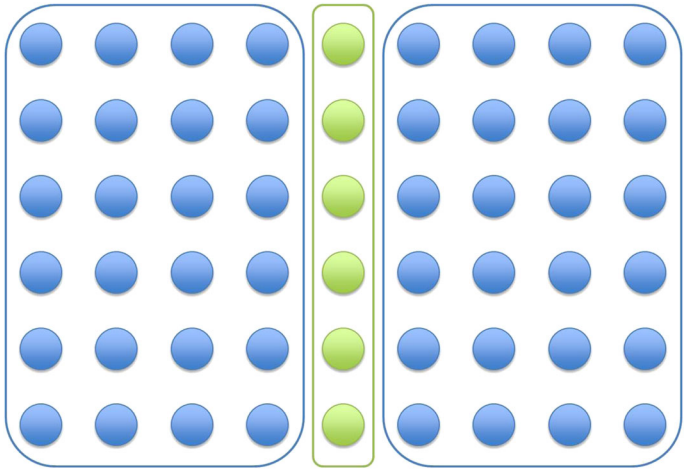


Figure 3. Nanodevice partitioned into two subregions (L, R) and a separator S (L stands for left, and R for right).

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

where $\widehat{\mathbf{A}}_{SS}$ is the Schur complement,

$$\widehat{\mathbf{A}}_{SS} = \mathbf{A}_{SS} - \mathbf{A}_{LS}^T \mathbf{A}_{LL}^{-1} \mathbf{A}_{LS} - \mathbf{A}_{RS}^T \mathbf{A}_{RR}^{-1} \mathbf{A}_{RS}.$$

The matrix $\mathbf{G}^r$ satisfies the relation

$$\mathbf{G}^r = \left(\mathbf{I} - \mathbf{L}^T\right) \mathbf{G}^r + \mathbf{D}^{-1}\mathbf{L}^{-1} \quad \text{with} \quad \mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T \tag{3}$$

(described in Takahashi *et al.* [20] and Erisman and Tinney [21]). The block notation yields

$$\mathbf{G}^r = -\begin{bmatrix} \mathbf{A}_{LL}^{-1}\mathbf{A}_{LS}\mathbf{G}_{SL}^r & \mathbf{A}_{LL}^{-1}\mathbf{A}_{LS}\mathbf{G}_{SR}^r & \mathbf{A}_{LL}^{-1}\mathbf{A}_{LS}\mathbf{G}_{SS}^r \\ \mathbf{A}_{RR}^{-1}\mathbf{A}_{RS}\mathbf{G}_{SL}^r & \mathbf{A}_{RR}^{-1}\mathbf{A}_{RS}\mathbf{G}_{SR}^r & \mathbf{A}_{RR}^{-1}\mathbf{A}_{RS}\mathbf{G}_{SS}^r \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{A}_{LL}^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{RR}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \widehat{\mathbf{A}}_{SS}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{LS}^T\mathbf{A}_{LL}^{-1} & -\mathbf{A}_{RS}^T\mathbf{A}_{RR}^{-1} & \mathbf{I} \end{bmatrix}.$$

This equation indicates

$$\begin{aligned} \mathbf{G}_{SS}^r &= \left(\widehat{\mathbf{A}}_{SS}\right)^{-1}, \\ \mathbf{G}_{LS}^r &= -\mathbf{A}_{LL}^{-1}\mathbf{A}_{LS}\mathbf{G}_{SS}^r, \\ \mathbf{G}_{RS}^r &= -\mathbf{A}_{RR}^{-1}\mathbf{A}_{RS}\mathbf{G}_{SS}^r. \end{aligned}$$

The diagonal blocks $\mathbf{G}_{LL}^r$ and $\mathbf{G}_{RR}^r$, for the regions L and R, respectively, are computed independently of each other,

$$\begin{aligned} \mathbf{G}_{LL}^r &= \mathbf{A}_{LL}^{-1} - \mathbf{A}_{LL}^{-1}\mathbf{A}_{LS}\left(\mathbf{G}_{LS}^r\right)^T = \mathbf{A}_{LL}^{-1} + \mathbf{A}_{LL}^{-1}\mathbf{A}_{LS}\mathbf{G}_{SS}^r\mathbf{A}_{LS}^T\mathbf{A}_{LL}^{-1} \\ \mathbf{G}_{RR}^r &= \mathbf{A}_{RR}^{-1} - \mathbf{A}_{RR}^{-1}\mathbf{A}_{RS}\left(\mathbf{G}_{RS}^r\right)^T = \mathbf{A}_{RR}^{-1} + \mathbf{A}_{RR}^{-1}\mathbf{A}_{RS}\mathbf{G}_{SS}^r\mathbf{A}_{RS}^T\mathbf{A}_{RR}^{-1} \end{aligned}$$

(where the symmetry of $\mathbf{G}^r$ has been exploited). For this simple case, the resulting algorithm matches exactly the HSC method [9].

For calculating entries in the correlation matrix $\mathbf{G}^<$, Petersen *et al.* [19] generalized Takahashi's method by writing

$$\mathbf{L}^T\mathbf{G}^<\left(\mathbf{L}^T\right)^\dagger = \mathbf{D}^{-1}\mathbf{L}^{-1}\mathbf{\Sigma}^<\mathbf{L}^{-\dagger}\mathbf{D}^{-\dagger}. \tag{4}$$

Recurrence formulas are described in [19]. Our approach utilizes a variant of (4), namely

$$\mathbf{G}^< = \left(\mathbf{I} - \mathbf{L}^T\right)\mathbf{G}^< + \mathbf{D}^{-1}\mathbf{L}^{-1}\mathbf{\Sigma}^<\left(\mathbf{G}^r\right)^\dagger, \tag{5}$$

and exploits the symmetry of $\mathbf{G}^r$ and the skew-Hermitian property of $\mathbf{\Sigma}^<$ and $\mathbf{G}^<$,

$$\left(\mathbf{G}^<\right)^\dagger = -\mathbf{G}^< \quad \text{and} \quad \left(\mathbf{\Sigma}^<\right)^\dagger = -\mathbf{\Sigma}^<. \tag{6}$$

The block $\mathbf{L}\mathbf{D}\mathbf{L}^T$-factorization of $\mathbf{A}$ yields

$$\mathbf{G}^< = -\begin{bmatrix} \mathbf{A}_{LL}^{-1}\mathbf{A}_{LS}\mathbf{G}_{SL}^< & \mathbf{A}_{LL}^{-1}\mathbf{A}_{LS}\mathbf{G}_{SR}^< & \mathbf{A}_{LL}^{-1}\mathbf{A}_{LS}\mathbf{G}_{SS}^< \\ \mathbf{A}_{RR}^{-1}\mathbf{A}_{RS}\mathbf{G}_{SL}^< & \mathbf{A}_{RR}^{-1}\mathbf{A}_{RS}\mathbf{G}_{SR}^< & \mathbf{A}_{RR}^{-1}\mathbf{A}_{RS}\mathbf{G}_{SS}^< \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{A}_{LL}^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{RR}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \left(\widehat{\mathbf{A}}_{SS}\right)^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{LS}^T\mathbf{A}_{LL}^{-1} & -\mathbf{A}_{RS}^T\mathbf{A}_{RR}^{-1} & \mathbf{I} \end{bmatrix} \mathbf{\Sigma}^<\left(\mathbf{G}^r\right)^\dagger.$$

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

Parts of $\mathbf{G}^r$ are computed with the previous algorithm, namely $\mathbf{G}^r_{LL}$, $\mathbf{G}^r_{RR}$, $\mathbf{G}^r_{SS}$, $\mathbf{G}^r_{RS}$, and $\mathbf{G}^r_{LS}$. By assumption, $\mathbf{\Sigma}^<$ is a block-diagonal skew-Hermitian matrix with purely imaginary entries. The partial matrix multiplication gives

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\mathbf{A}^T_{LS}\mathbf{A}^{-1}_{LL} & -\mathbf{A}^T_{RS}\mathbf{A}^{-1}_{RR} & \mathbf{I} \end{bmatrix} \mathbf{\Sigma}^< \left(\mathbf{G}^r\right)^\dagger =$$

$$\begin{bmatrix} \mathbf{\Sigma}^<_{LL}\left(\mathbf{G}^r_{LL}\right)^\dagger & * & \mathbf{\Sigma}^<_{LL}\left(\mathbf{G}^r_{SL}\right)^\dagger \\ * & \mathbf{\Sigma}^<_{RR}\left(\mathbf{G}^r_{RR}\right)^\dagger & \mathbf{\Sigma}^<_{RR}\left(\mathbf{G}^r_{RS}\right)^\dagger \\ * & * & \mathbf{\Sigma}^<_{SS}\left(\mathbf{G}^r_{SS}\right)^\dagger - \mathbf{A}^T_{LS}\mathbf{A}^{-1}_{LL}\mathbf{\Sigma}^<_{LL}\left(\mathbf{G}^r_{SL}\right)^\dagger - \mathbf{A}^T_{RS}\mathbf{A}^{-1}_{RR}\mathbf{\Sigma}^<_{RR}\left(\mathbf{G}^r_{SR}\right)^\dagger \end{bmatrix}$$

(where the starred blocks are not computed). This relation indicates

$$\begin{aligned} \mathbf{G}^<_{SS} &= \mathbf{G}^r_{SS}\left(\mathbf{\Sigma}^<_{SS}\left(\mathbf{G}^r_{SS}\right)^\dagger - \mathbf{A}^T_{LS}\mathbf{A}^{-1}_{LL}\mathbf{\Sigma}^<_{LL}\left(\mathbf{G}^r_{SL}\right)^\dagger - \mathbf{A}^T_{RS}\mathbf{A}^{-1}_{RR}\mathbf{\Sigma}^<_{RR}\left(\mathbf{G}^r_{SR}\right)^\dagger\right), \\ \mathbf{G}^<_{LS} &= -\mathbf{A}^{-1}_{LL}\mathbf{A}_{LS}\mathbf{G}^<_{SS} + \mathbf{A}^{-1}_{LL}\mathbf{\Sigma}^<_{LL}\left(\mathbf{G}^r_{SL}\right)^\dagger, \\ \mathbf{G}^<_{RS} &= -\mathbf{A}^{-1}_{RR}\mathbf{A}_{RS}\mathbf{G}^<_{SS} + \mathbf{A}^{-1}_{RR}\mathbf{\Sigma}^<_{RR}\left(\mathbf{G}^r_{SR}\right)^\dagger. \end{aligned}$$

Finally, the diagonal blocks $\mathbf{G}^<_{LL}$ and $\mathbf{G}^<_{RR}$, for the regions L and R, respectively, are computed independently of each other,

$$\begin{aligned} \mathbf{G}^<_{LL} &= \mathbf{A}^{-1}_{LL}\mathbf{\Sigma}^<_{LL}\left(\mathbf{G}^r_{LL}\right)^\dagger - \mathbf{A}^{-1}_{LL}\mathbf{A}_{LS}\left(\mathbf{G}^<_{LS}\right)^\dagger \\ \mathbf{G}^<_{RR} &= \mathbf{A}^{-1}_{RR}\mathbf{\Sigma}^<_{RR}\left(\mathbf{G}^r_{RR}\right)^\dagger - \mathbf{A}^{-1}_{RR}\mathbf{A}_{RS}\left(\mathbf{G}^<_{RS}\right)^\dagger \end{aligned}$$

(where the skew-Hermitian property of $\mathbf{G}^<$ has been exploited).

This derivation for calculating the correlation matrix $\mathbf{G}^<$ differs from the FIND method [12, 16] in several aspects. It uses thin boundaries obtained directly from the nested dissection. This extension requires one sparse factorization and one back substitution, whereas FIND utilizes only sparse factorizations but applied many times with different orderings. The order of operations to obtain the diagonal blocks of $\mathbf{G}^<$ is also different from the recurrence in Petersen *et al.* [19], which uses the sequence

$$\begin{aligned} \mathbf{\Sigma}^< &\to \mathbf{L}^{-1}\mathbf{\Sigma}^<\left(\mathbf{L}^{-1}\right)^\dagger \to \mathbf{D}^{-1}\mathbf{L}^{-1}\mathbf{\Sigma}^<\left(\mathbf{L}^{-1}\right)^\dagger\left(\mathbf{D}^{-1}\right)^\dagger \\ &\to \mathbf{L}^{-T}\mathbf{D}^{-1}\mathbf{L}^{-1}\mathbf{\Sigma}^<\left(\mathbf{L}^{-1}\right)^\dagger\left(\mathbf{D}^{-1}\right)^\dagger\left(\mathbf{L}^{-T}\right)^\dagger, \end{aligned}$$

whereas our HSC extension uses

$$\mathbf{\Sigma}^< \to \mathbf{\Sigma}^<\left(\mathbf{G}^r\right)^\dagger \to \mathbf{L}^{-1}\mathbf{\Sigma}^<\left(\mathbf{G}^r\right)^\dagger \to \mathbf{D}^{-1}\mathbf{L}^{-1}\mathbf{\Sigma}^<\left(\mathbf{G}^r\right)^\dagger \to \mathbf{L}^{-T}\mathbf{D}^{-1}\mathbf{L}^{-1}\mathbf{\Sigma}^<\left(\mathbf{G}^r\right)^\dagger.$$

When working with sparse matrices, specific order of operations may result in fewer operations. In numerical experiments, the latter ordering was more efficient.

3.2. Description for a multilevel case

In this section, the description is extended to an arbitrary number of clusters.

Even though computing the diagonal for the inverse of a matrix is not equivalent to a sparse factorization, both problems benefit from matrix reordering. The multilevel nested dissection, introduced by George [13], lends itself naturally to the creation of grid points clusters. Typically, nested dissection divides the system into two disconnected sets and an interface, called the separator. Then, the process is repeated recursively on each set to create a multilevel binary tree.

On the basis of the hierarchical structure of the tree, let P_i denote the set of all cluster indices j such that cluster j is an ancestor of cluster i. For example, for Figure 4, P_5 is equal to $\{1, 2\}$, and $P_{15} = \{1, 3, 7\}$. Let C_i denote the set of all cluster indices j such that cluster j is a descendant of cluster i. For the partition on Figure 4, C_4 is equal to $\{8, 9\}$, and $C_3 = \{6, 7, 12, 13, 14, 15\}$.

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

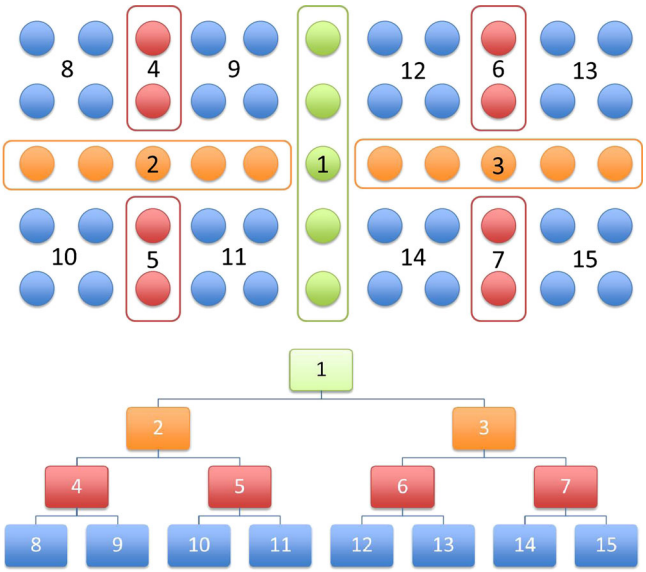


Figure 4. Example of a multilevel partition.

Note that a cluster may or may not have a direct coupling in the matrix $\mathbf{A}$ to any of its ancestors or descendants.

Once the partition is set, the algorithm may be separated into two distinct parts: computation of $\mathbf{G}^r$ and computation of $\mathbf{G}^<$.

Computation of blocks for $\mathbf{G}^r$. In the binary tree, the levels are labeled from bottom to top, where level 1 contains all clusters at the end of the tree and level L contains only the original separator. For simplicity of presentation, let $\mathbf{A}^{(l)}$ denote the matrix transformed from $\mathbf{A}$ after folding all the clusters up to level l. Note that $\mathbf{A}^{(0)}$ is set to $\mathbf{A}$ and $\mathbf{A}^{(L-1)}$ is block diagonal.

The computation of blocks for $\mathbf{G}^r$ involve three steps: folding the lower level clusters unto the higher ones, inversion of the matrix for the main separator, and extracting of the diagonal blocks for the current level from blocks on higher level.

The algorithm for the first step goes as follows:

- For $l = 1$ up to $L - 1$,
  - $\mathbf{A}^{(l)} = \mathbf{A}^{(l-1)}$
  - For all the clusters i on level l,
    - $\mathbf{\Psi}_{i,j} = -\left(\mathbf{A}^{(l)}_{i,i}\right)^{-1} \mathbf{A}^{(l)}_{i,j}$ for all j in P_i
    - $\mathbf{A}^{(l)}_{j,k} = \mathbf{A}^{(l)}_{j,k} + \mathbf{\Psi}^T_{i,j} \mathbf{A}^{(l)}_{i,k}$ for all j and k in P_i
    - $\mathbf{A}^{(l)}_{k,j} = \left(\mathbf{A}^{(l)}_{j,k}\right)^T$ for all j and k in P_i
    - $\mathbf{A}^{(l)}_{i,j} = \mathbf{0}$ and $\mathbf{A}^{(l)}_{j,i} = \mathbf{0}$ for all j in P_i
  - end
- end

The next step is written as the inversion of $\mathbf{A}^{(L-1)}$, which is symmetric and block diagonal.

- $\mathbf{G}^{(L-1)} = \left(\mathbf{A}^{(L-1)}\right)^{-1}$.

In practice, the operation requires only the inversion of the block for the top separator. All the other blocks have already been inverted during the folding steps.

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

Finally, all the diagonal blocks of $\mathbf{G}^r$ are extracted one level at a time. The algorithm goes as follows:

- For $l = L-2$ down to 0,
  - $\mathbf{G}^{(l)} = \mathbf{G}^{(l+1)}$
  - For all the clusters i on level l,
    - $\mathbf{G}^{(l)}_{i,j} = \mathbf{G}^{(l)}_{i,j} + \sum_{k\in P_i} \mathbf{\Psi}_{i,k}\mathbf{G}^{(l)}_{k,j}$ for all cluster indices j in P_i
    - $\mathbf{G}^{(l)}_{j,i} = \left(\mathbf{G}^{(l)}_{i,j}\right)^T$ for all cluster indices j in P_i
    - $\mathbf{G}^{(l)}_{i,i} = \mathbf{G}^{(l)}_{i,i} + \sum_{j\in P_i} \mathbf{\Psi}_{i,j}\mathbf{G}^{(l)}_{j,i}$
  - end
- end

The resulting algorithm to compute block entries in $\mathbf{G}^r$ matches exactly the HSC method [9].

Note that matrix $\mathbf{G}^{(0)}$ is not equal to $\mathbf{G}^r$ because $\mathbf{G}^{(0)}$ is incomplete (see an example in the Appendix). However, all the entries in $\mathbf{G}^{(0)}$, in particular, the diagonal entries, match the corresponding entries in $\mathbf{G}^r$.

Computation of blocks for $\mathbf{G}^<$. The algorithm consists of four steps. The first step uses the matrix $\mathbf{G}^{(0)}$ computed previously.

- $\mathbf{N} = \mathbf{\Sigma}^< \left(\mathbf{G}^{(0)}\right)^\dagger$.

All the entries in $\mathbf{G}^{(0)}$ match the corresponding entries in $\mathbf{G}^r$ and are sufficient to compute diagonal blocks of $\mathbf{G}^<$. The matrix $\Sigma^<$ is typically block diagonal. The matrix $\mathbf{N}$ will have the same structure and shape as $\mathbf{G}^{(0)}$. The matrix multiplication is carried out block by block.

Next, the lower level clusters are folded into the higher ones. This step is critical and the most time consuming. Let $\mathbf{N}^{(l)}$ denote the matrix transformed from $\mathbf{N}$ after folding all the clusters up to level l. $\mathbf{N}^{(0)}$ is set to $\mathbf{N}$.

- For $l = 1$ up to $L-1$,
  - $\mathbf{N}^{(l)} = \mathbf{N}^{(l-1)}$
  - For all the clusters i on level l,
    - $\mathbf{N}^{(l)}_{j,k} = \mathbf{N}^{(l)}_{j,k} + \mathbf{\Psi}^T_{i,j}\mathbf{N}^{(l)}_{i,k}$ for all j and k in P_i
  - end
- end

Similar to Step 1, Step 3 is a block diagonal multiplication.

- $\mathbf{P}^{(L-1)} = \mathbf{G}^{(L-1)}\mathbf{N}^{(L-1)}$.

Finally, Step 4 extracts all the diagonal blocks one level at a time. This step is similar to the extraction in the $\mathbf{G}^r$ algorithm. The operations are the following:

- For $l = L-2$ down to 0,
  - $\mathbf{P}^{(l)} = \mathbf{P}^{(l+1)}$
  - For all the clusters i on level l,
    - $\mathbf{P}^{(l)}_{i,j} = \mathbf{P}^{(l)}_{i,j} + \sum_{k\in P_i} \mathbf{\Psi}_{i,k}\mathbf{P}^{(l)}_{k,j}$ for all cluster indices j in P_i
    - $\mathbf{P}^{(l)}_{j,i} = -\left(\mathbf{P}^{(l)}_{i,j}\right)^\dagger$ for all cluster indices j in P_i
    - $\mathbf{P}^{(l)}_{i,i} = \mathbf{P}^{(l)}_{i,i} + \sum_{j\in P_i} \mathbf{\Psi}_{i,j}\mathbf{P}^{(l)}_{j,i}$
  - end
- end

At the end, matrix $\mathbf{P}^{(0)}$ is not equal to $\mathbf{G}^<$ because $\mathbf{P}^{(0)}$ is incomplete (see an example in the Appendix). However, all the entries in $\mathbf{P}^{(0)}$, in particular, the diagonal entries, match the corresponding entries in $\mathbf{G}^<$.

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

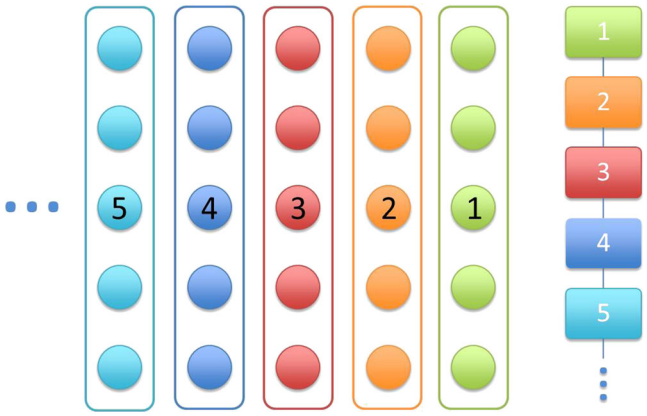


Figure 5. Partition generating the recursive Green's function algorithm.

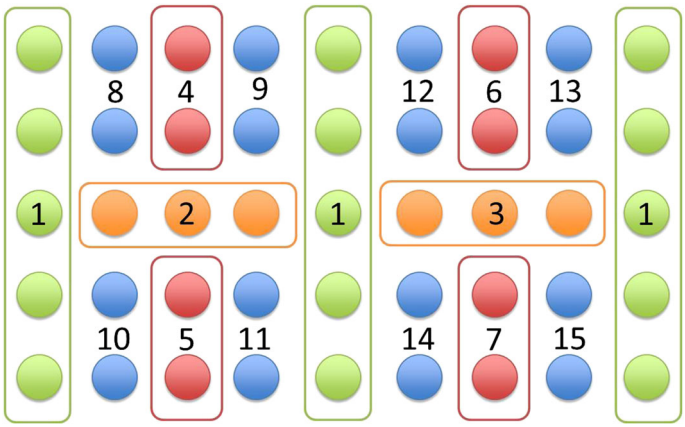


Figure 6. First method to partition system with two dense layers and two ends.

Comments on the system partition. The partitioning of the system (or the clustering of points) is the key step for the efficiency of this algorithm. The partition should follow two rules:

(i) For clusters within the same level on the binary tree, no interaction is allowed. Operations on blocks at the same level are performed independently.
(ii) The partition should minimize the size of separators and reduce the clusters down to a size manageable for an inversion of the corresponding block matrix.

The multilevel nested dissection generates a partition that satisfies those rules. It is worthwhile to note that, as long as the rules stated previously are followed, systems with nonuniform distribution of points or with a different stencil could be treated correctly.

The RGF algorithm is included in the previous description with a very specific partition. Such a partition is illustrated in Figure 5.

In many cases, self-energy functions add two $N_x \times N_x$ dense blocks into the input sparse matrix **A** in first and last block (Figure 2). One possible partition combines the two contacts together with the middle separator 1, as shown in Figure 6. The weakness of such clustering is the size of the first separator (or root region). A larger separator increases the computational cost spent on this level. More descendant blocks are coupled with this separator, and the total number of operations will increase dramatically. Another partition that satisfies the previous two rules is plotted in Figure 7.

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

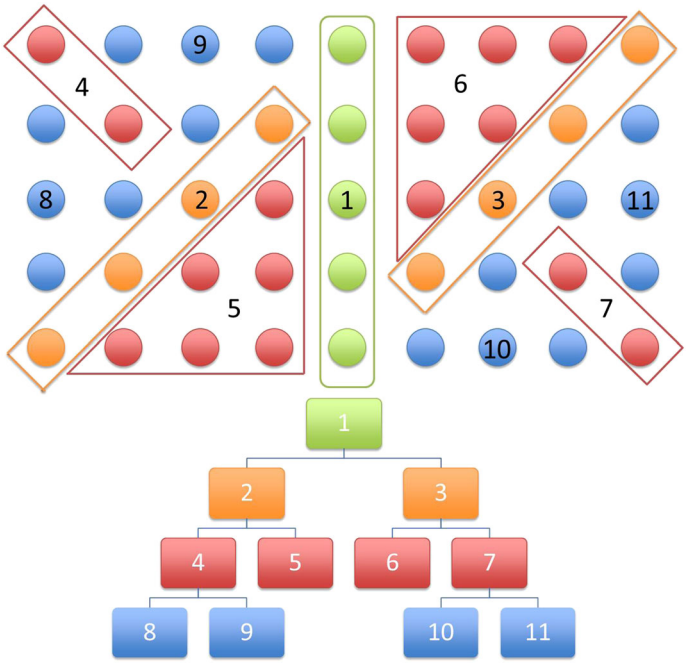


Figure 7. Partition generated by METIS for system including dense layers at two ends.

4. NUMERICAL EXPERIMENTS

This section describes numerical experiments on two simple models: a superlattice structure and a graphene nanotube. The algorithm is implemented in a C code that is interfaced with METIS [17].

4.1. Cost analysis

First, the complexity of the HSC extension is compared numerically to the complexity of RGF. A model device is considered where the system Hamiltonian is discretized with a five-point stencil. The left and right contact self energies are neglected for this section only. A typical partition is plotted in Figure 4.

The numerical estimate tracks the operation counts step by step for all the matrix multiplications and matrix inversions throughout the code. For a multiplication of two matrices with dimensions

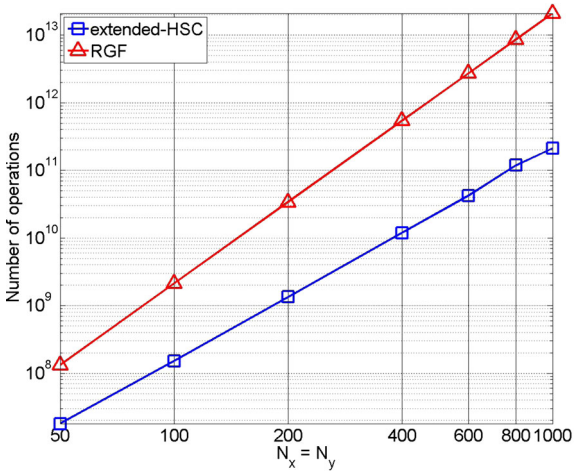


Figure 8. Numerical count comparison for our algorithm (blue) and recursive Green's function (RGF; red).

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

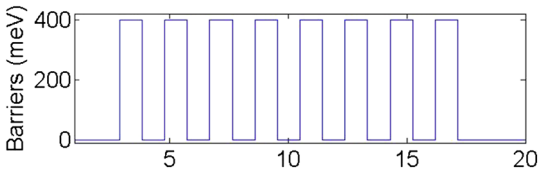


Figure 9. Barrier structure for a model superlattice device.

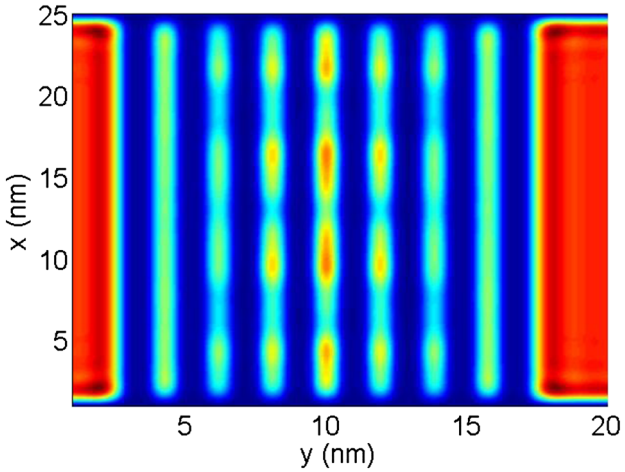


Figure 10. Electron density for a model superlattice device.

$i \times j$ and $j \times k$, a total of ijk operations is added. For inversion of a matrix of dimension $i \times i$, i^3 operations are counted.

Figure 8 shows the cost comparison between the HSC extension and the RGF for two-dimensional square systems with the same number of grid points (or atoms) per direction, that is, $N_x = N_y = N$. The plot in logarithmic scale indicates that RGF exhibits a complexity of $\mathcal{O}(N^4)$ whereas the HSC extension shows a $\mathcal{O}(N^3)$ complexity.

4.2. Results

4.2.1. Superlattice device. A superlattice device is typically a multilayered energy barriers system. The device is composed of repeating junctions of energy barriers and wells. To verify the simulation results, a two-dimensional system of lengths $l_x = 25$ nm and $l_y = 20$ nm is considered and plotted in Figure 9. Here, the structure has eight barriers, each of width 1 nm and of height 400 meV. The wells have a width of 1 nm. The length of the left flat band region is 2 nm, and the right flat band region is 3 nm long.

A simulation with a five-point stencil discretization on a grid with spacing $\mathrm{d}x = \mathrm{d}y = 0.1$ nm is made for the Fermi energy $E_f = 140$ meV and 500 energy points uniformly distributed between 0 meV and 500 meV. The density of states, electron density, and current are calculated by the RGF method and the extended HSC method. The output electron density is plotted in Figure 10. Linear electron density in the y-direction is illustrated in Figure 11. The figures indicate that the charge distribution in the barrier-well multilayer junctions are symmetric, as expected.

Next, devices of lengths $l_x = N_x \times 0.1$ nm and $l_y = N_y \times 0.1$ nm are used to compare the two algorithms. The number of barriers is kept at 8. The lengths for the two-sides flat region are adjusted according to the lengths of device l_x and l_y. The other parameters remain unchanged.

In Figure 12(a), diagonal self-energy matrices are used for the left and right contacts. Calculation times are compared for square systems – that is, $N_x = N_y = N$ – and plotted in Figure 12(a). As expected, the HSC extension exhibits smaller CPU times and a complexity of $\mathcal{O}(N^3)$, whereas RGF's complexity is $\mathcal{O}(N^4)$.

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

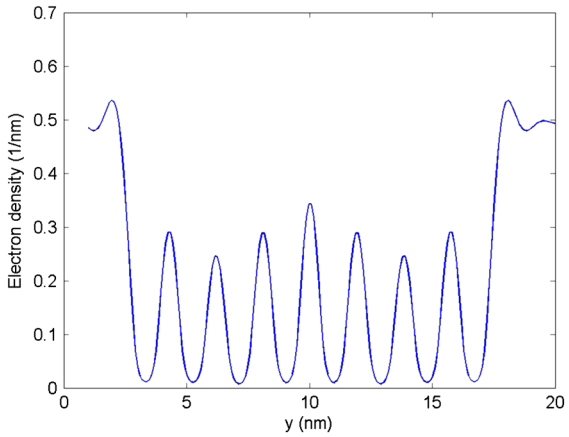


Figure 11. Electron density in y-direction of the example superlattice structure.

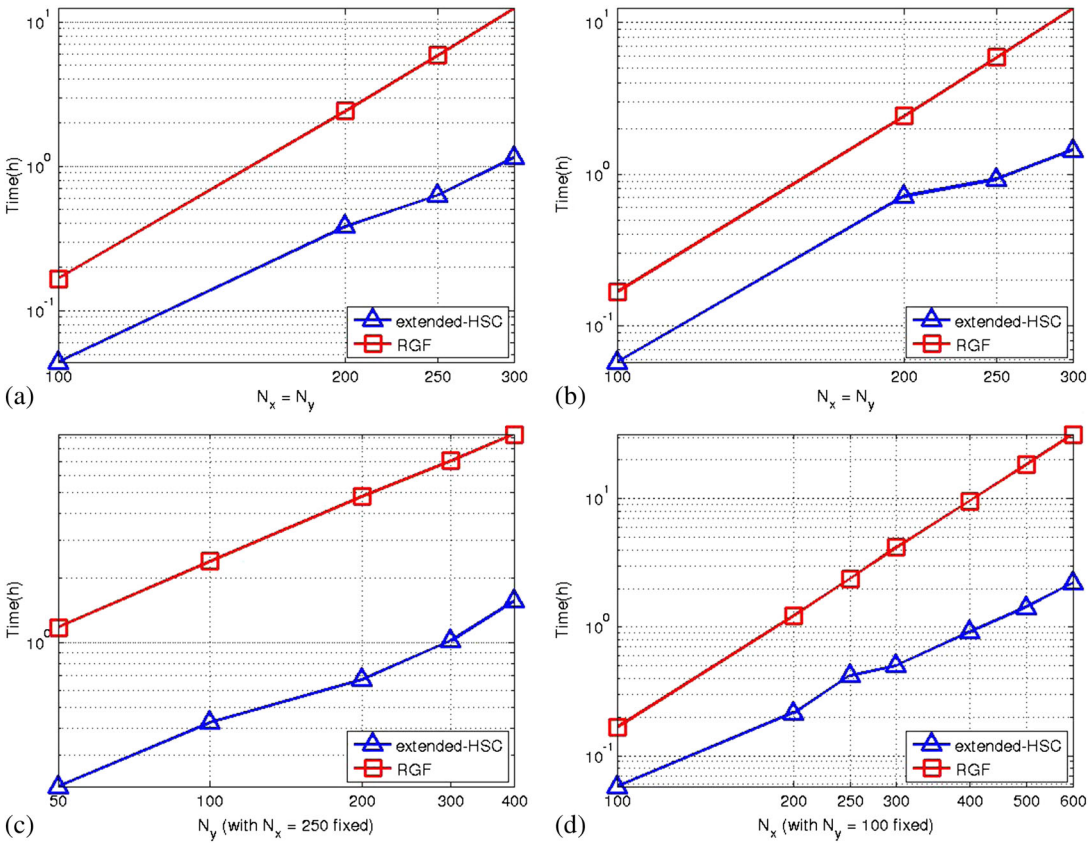


Figure 12. Superlattice device nonequilibrium Green's function simulation computation time comparison for recursive Green's function (RGF) and our methods, all systems grid spacing is 0.1 nm. (a) Square system of with diagonal self-energy matrix; (b) square system of with dense self-energy matrix; (c) for systems in this plot, the length in the x-direction is fixed at 25 nm, whereas the length in the y-direction is increased. (d) For systems in this plot, the length in the y-direction is fixed at 10 nm, whereas the length in the x-direction is increased. Dense self-energy matrices are used in (c) and (d) devices.

In Figure 12(b), CPU times for square systems with dense self-energy matrices for both contacts are plotted. Here again, the HSC extension exhibits smaller CPU times. A complexity $\mathcal{O}(N^3)$ for HSC extension compared with $\mathcal{O}(N^4)$ for RGF can be seen.

Figure 12(c) plots CPU times for rectangular devices where $l_x = 25$ nm (or $N_x = 250$) and the length in the y-direction is varied. Dense self-energy blocks for the left and right contacts are

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

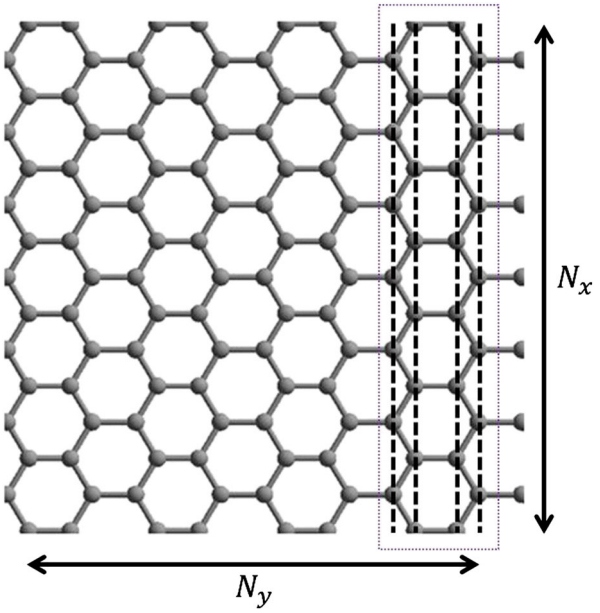


Figure 13. Graphene hexagonal structure decomposed by tight biding method. Dashed rectangular illustrates one repeating hexagon layer. Dashed lines represent inner four atom layers, showing the atoms ordering in tight binding Hamiltonian construction.

employed. The implementation of RGF is biased towards the x-direction so that its complexity is $\mathcal{O}\left(N_y\right)$. The linear trend is clearly present in the plot. For the HSC extension, similar to the cost of a sparse $\mathbf{LDL}^T$ factorization, the computational cost is $\mathcal{O}\left(N_y\right)$. Clearly, the constant for RGF is larger for this device.

To illustrate the dependence of this constant with respect to N_x, Figure 12(d) plots the CPU times when N_y is fixed and N_x is varied. The recorded CPU times illustrate that the RGF method has an asymptotic complexity $\mathcal{O}\left(N_x^3\right)$ whereas the HSC extension exhibits a complexity $\mathcal{O}\left(N_x^2\right)$. So, for rectangular devices, the RGF method has a complexity $\mathcal{O}\left(N_x^3 N_y\right)$, and the HSC extension a complexity $\mathcal{O}\left(N_x^2 N_y\right)$.

4.2.2. Graphene. Graphene is one of the most promising next generation materials. Its remarkable electric properties, such as high carrier mobility and zero band gaps, generate a rapidly increasing interest in the electronic device community. Since 2007, many advances in graphene-based transistor development have been reported [22].

The NEGF simulation of graphene transport is based on tight binding method, which yields a four-point-stencil Hamiltonian because of system decomposition of carbon atoms coupling (Figure 13). In the numerical experiments, armchair planar graphene nanoribbon structures are simulated. The on-site energy for each carbon atom is 0, and the hopping parameter between two nearest carbon atoms is -3.1 eV. The Fermi energy is set to 0. The simulation is run for only one energy point $E = 0.5$ eV.

Simulation timings are plotted in Figure 14 for graphene structures of different sizes. To minimize the dimension of blocks to invert in RGF, one layer of hexagonal structure is divided into four layers (see the dashed lines in Figure 13). Conclusions on the asymptotic complexity remain unchanged. Namely, the HSC extension is more efficient than the RGF method for square and rectangular structures. The complexity of RGF for four-point stencil behaves as $\mathcal{O}\left(N_x^3 N_y\right)$ with the same constant as five-point stencil, which is expected because of the layered system partition. In Figure 14(a and b), for a four-point stencil system, our HSC extension exhibits a complexity of $\mathcal{O}(N^3)$ for square system. Figure 14(c and d) also demonstrate a complexity growing linearly with N_y when N_x is fixed and quadratically with N_x when N_y is fixed. The final result is a complexity $\mathcal{O}\left(N_x^2 N_y\right)$. The

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

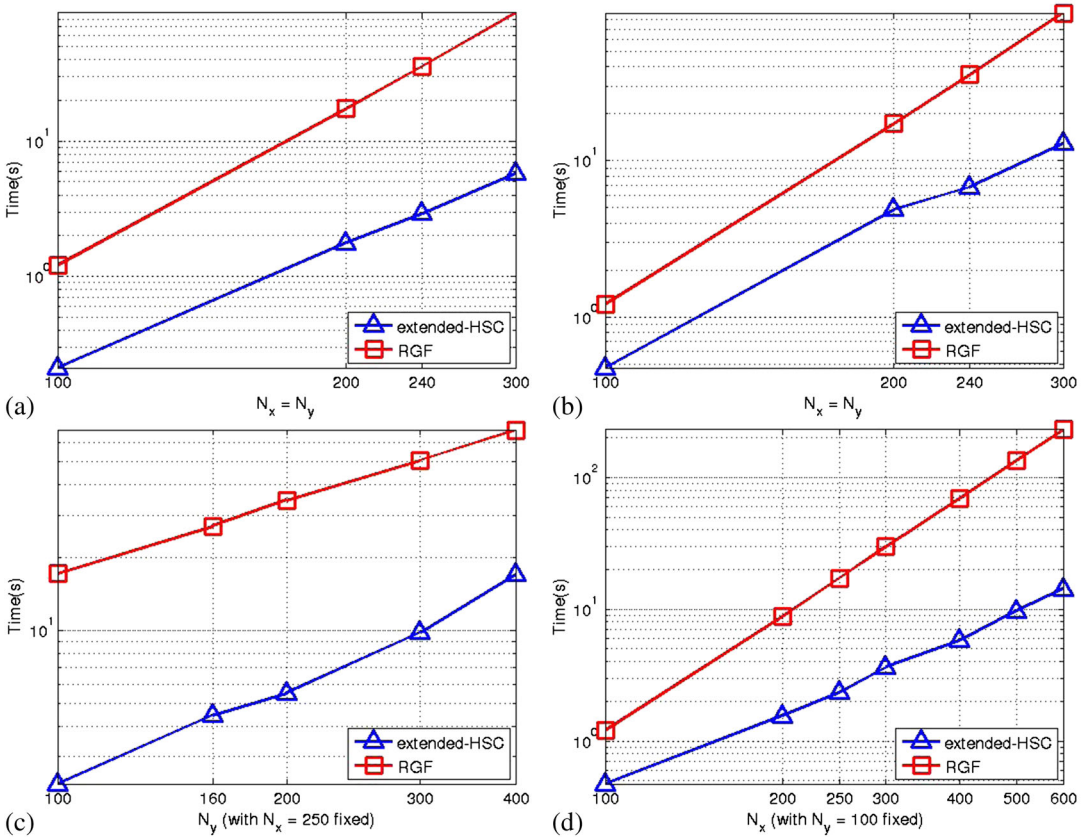


Figure 14. Graphene device nonequilibrium Green's function simulation computation time comparison for recursive Green's function (RGF) and the hierarchical Schur complement (HSC) extension, based on tight binding theory. (a) Square system of with diagonal self-energy matrix. (b) Square system of with dense self-energy matrix. (c) For systems in this plot, the number of atoms in the x-direction is set to $N_x = 250$, whereas the number of atoms in the y-direction is increased. (d) For systems in this plot, the number of atoms in the y-direction is set to $N_y = 100$, whereas the number of atoms in the x-direction is increased. Dense self-energy matrices are used in (c) and (d) devices.

constant in front of $N_x^2 N_y$ for the extended HSC method is smaller for the graphene structures than for the superlattice devices. This reduction is explained by a more efficient partitioning of four-point stencil systems by METIS .

5. CONCLUSION

In this paper, an approach to calculate the charge density in nanoscale devices, within the context of the NEGF approach, is presented. This work exploits recent advances to use an established graph partitioning method, namely the nested dissection method. This contribution does not require any processing of the partition, and it can handle open boundary conditions, represented by full self-energy matrices. The key ingredients are an efficient sparse block $\mathbf{LDL}^T$-factorization and an appropriate order of operations to preserve the sparsity as much as possible. The resulting algorithm was illustrated on a quantum well superlattice and a carbon nanotube, which are represented by a continuum and tight binding Hamiltonian, respectively, and demonstrated a significant speedup over the recursive method RGF.

APPENDIX A: DESCRIPTION OF THE ALGORITHM FOR A THREE-LEVEL TREE

To make the extension more comprehensive, a description of the HSC extension is given for a three-level system (Figure A.1).

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

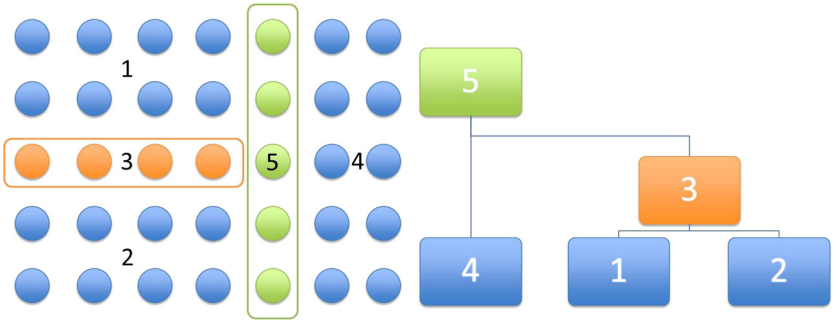


Figure A.1. Partition for a three-level system.

The size of the partition is chosen to make the description as relevant as possible without becoming overcomplicated. The first level contains regions 1, 2, and 4. The second level refers to region 3, and the top level is the root or region 5. To illustrate the algorithm, steps from Section 3.2 are described for this particular device.

When a five-point stencil is used for discretization, the structure of the matrix $\mathbf{A}$, after reordering, is

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{13} & \mathbf{0} & \mathbf{A}_{15} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} & \mathbf{0} & \mathbf{A}_{25} \\ \mathbf{A}_{13}^T & \mathbf{A}_{23}^T & \mathbf{A}_{33} & \mathbf{0} & \mathbf{A}_{35} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{44} & \mathbf{A}_{45} \\ \mathbf{A}_{15}^T & \mathbf{A}_{25}^T & \mathbf{A}_{35}^T & \mathbf{A}_{45}^T & \mathbf{A}_{55} \end{bmatrix}$$

The blocks $\mathbf{A}_{11}$ and $\mathbf{A}_{44}$ are dense for representing the contacts with the semi-infinite leads.

Recall that $\mathbf{A}^{(0)}$ is equal to $\mathbf{A}$. Then, inner points in regions 1, 2, and 4 are eliminated by block Gaussian elimination – the effects of the inner points in regions 1, 2, and 4 are folded over their boundary. This first step yields the matrix $\mathbf{A}^{(1)}$

$$\mathbf{A}^{(1)} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{33}^{(1)} & \mathbf{0} & \mathbf{A}_{35}^{(1)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{44} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \left(\mathbf{A}_{35}^{(1)}\right)^T & \mathbf{0} & \mathbf{A}_{55}^{(1)} \end{bmatrix}$$

where the updated matrices are

$$\begin{aligned} \mathbf{A}_{33}^{(1)} &= \mathbf{A}_{33} - \mathbf{A}_{13}^T \mathbf{A}_{11}^{-1} \mathbf{A}_{13} - \mathbf{A}_{23}^T \mathbf{A}_{22}^{-1} \mathbf{A}_{23}, \\ \mathbf{A}_{35}^{(1)} &= \mathbf{A}_{35} - \mathbf{A}_{13}^T \mathbf{A}_{11}^{-1} \mathbf{A}_{15} - \mathbf{A}_{23}^T \mathbf{A}_{22}^{-1} \mathbf{A}_{25}, \\ \mathbf{A}_{55}^{(1)} &= \mathbf{A}_{55} - \mathbf{A}_{15}^T \mathbf{A}_{11}^{-1} \mathbf{A}_{15} - \mathbf{A}_{25}^T \mathbf{A}_{22}^{-1} \mathbf{A}_{25} - \mathbf{A}_{45}^T \mathbf{A}_{44}^{-1} \mathbf{A}_{45}. \end{aligned}$$

After folding the effects of regions 1 and 2, block $\mathbf{A}_{35}^{(1)}$ is now a dense block. Figure A.2 illustrates the change of sparsity between $\mathbf{A}$ and $\mathbf{A}^{(1)}$.

In the next step, the remaining off-diagonal blocks are eliminated to obtain the matrix $\mathbf{A}^{(2)}$,

$$\mathbf{A}^{(2)} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{33}^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{44} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{55}^{(2)} \end{bmatrix}$$

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

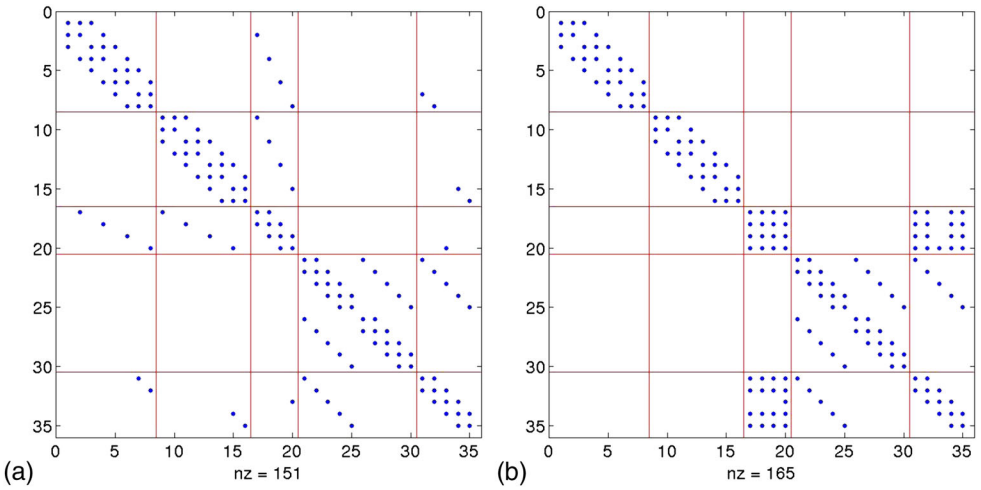


Figure A.2. Sparsity of matrix $\mathbf{A}$ and matrix $\mathbf{A}^{(1)}$. (a) $nz = 151$, and (b) $nz = 165$.

where the block $\mathbf{A}_{55}^{(2)}$ is

$$\mathbf{A}_{55}^{(2)} = \mathbf{A}_{55}^{(1)} - \left(\mathbf{A}_{35}^{(1)}\right)^T \left(\mathbf{A}_{33}^{(1)}\right)^{-1} \mathbf{A}_{35}^{(1)}.$$

Note that several blocks are unchanged, such as $\mathbf{A}_{11}$, $\mathbf{A}_{22}$, $\mathbf{A}_{33}^{(1)}$, and $\mathbf{A}_{44}$. The next step is written as the inversion of $\mathbf{A}^{(2)}$, which is symmetric and block diagonal,

$$\mathbf{G}^{(2)} = \begin{bmatrix} \mathbf{A}_{11}^{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \left(\mathbf{A}_{33}^{(1)}\right)^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{44}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \left(\mathbf{A}_{55}^{(2)}\right)^{-1} \end{bmatrix}.$$

This operation requires only the inversion of the block $\mathbf{A}_{55}^{(2)}$. All the other blocks have been inverted during the folding steps.

Next, diagonal blocks of $\mathbf{G}^r$ are extracted one level at a time. Starting from the main root (or separator), blocks at level 2 are updated to obtain

$$\mathbf{G}^{(1)} = \begin{bmatrix} \mathbf{G}_{11}^{(2)} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{22}^{(2)} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_{33}^{(1)} & \mathbf{0} & \mathbf{G}_{35}^{(1)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{G}_{44}^{(2)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \left(\mathbf{G}_{35}^{(1)}\right)^T & \mathbf{0} & \mathbf{G}_{55}^{(2)} \end{bmatrix}$$

with

$$\mathbf{G}_{35}^{(1)} = -\left(\mathbf{A}_{33}^{(1)}\right)^{-1} \mathbf{A}_{35}^{(1)} \mathbf{G}_{55}^{(2)} = \mathbf{\Psi}_{35} \mathbf{G}_{55}^{(2)},$$
$$\mathbf{G}_{33}^{(1)} = \mathbf{G}_{33}^{(2)} - \left(\mathbf{A}_{33}^{(1)}\right)^{-1} \mathbf{A}_{35}^{(1)} \left(\mathbf{G}_{35}^{(1)}\right)^T = \mathbf{G}_{33}^{(2)} + \mathbf{\Psi}_{35} \left(\mathbf{G}_{35}^{(1)}\right)^T.$$

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

Finally, blocks for regions 1, 2, and 4 are updated, yielding the matrix $\mathbf{G}^{(0)}$,

$$\mathbf{G}^{(0)} = \begin{bmatrix} \mathbf{G}_{11}^{(0)} & \mathbf{0} & \mathbf{G}_{13}^{(0)} & \mathbf{0} & \mathbf{G}_{15}^{(0)} \\ \mathbf{0} & \mathbf{G}_{22}^{(0)} & \mathbf{G}_{23}^{(0)} & \mathbf{0} & \mathbf{G}_{25}^{(0)} \\ \left(\mathbf{G}_{13}^{(0)}\right)^T & \left(\mathbf{G}_{23}^{(0)}\right)^T & \mathbf{G}_{33}^{(1)} & \mathbf{0} & \mathbf{G}_{35}^{(1)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{G}_{44}^{(0)} & \mathbf{G}_{45}^{(0)} \\ \left(\mathbf{G}_{15}^{(0)}\right)^T & \left(\mathbf{G}_{25}^{(0)}\right)^T & \left(\mathbf{G}_{35}^{(1)}\right)^T & \left(\mathbf{G}_{45}^{(0)}\right)^T & \mathbf{G}_{55}^{(2)} \end{bmatrix}.$$

Blocks for region 4 are satisfying

$$\begin{aligned} \mathbf{G}_{45}^{(0)} &= -\left(\mathbf{A}_{44}^{(0)}\right)^{-1} \mathbf{A}_{45}^{(0)} \mathbf{G}_{55}^{(2)} = \mathbf{\Psi}_{45} \mathbf{G}_{55}^{(2)} \\ \mathbf{G}_{44}^{(0)} &= \mathbf{G}_{44}^{(2)} - \left(\mathbf{A}_{44}^{(0)}\right)^{-1} \mathbf{A}_{45}^{(0)} \left(\mathbf{G}_{45}^{(0)}\right)^T. \end{aligned}$$

Blocks for region 1 are defined by

$$\begin{aligned} \mathbf{G}_{15}^{(0)} &= -\left(\mathbf{A}_{11}^{(0)}\right)^{-1} \mathbf{A}_{15}^{(0)} \mathbf{G}_{55}^{(2)} - \left(\mathbf{A}_{11}^{(0)}\right)^{-1} \mathbf{A}_{13}^{(0)} \mathbf{G}_{35}^{(1)} \\ \mathbf{G}_{13}^{(0)} &= -\left(\mathbf{A}_{11}^{(0)}\right)^{-1} \mathbf{A}_{13}^{(0)} \mathbf{G}_{33}^{(1)} - \left(\mathbf{A}_{11}^{(0)}\right)^{-1} \mathbf{A}_{15}^{(0)} \left(\mathbf{G}_{35}^{(1)}\right)^T \\ \mathbf{G}_{11}^{(0)} &= \left(\mathbf{A}_{11}^{(0)}\right)^{-1} - \left(\mathbf{A}_{11}^{(0)}\right)^{-1} \mathbf{A}_{13}^{(0)} \left(\mathbf{G}_{13}^{(0)}\right)^T - \left(\mathbf{A}_{11}^{(0)}\right)^{-1} \mathbf{A}_{15}^{(0)} \left(\mathbf{G}_{15}^{(0)}\right)^T, \end{aligned}$$

and blocks for region 2

$$\begin{aligned} \mathbf{G}_{25}^{(0)} &= -\left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{25}^{(0)} \mathbf{G}_{55}^{(2)} - \left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{23}^{(0)} \mathbf{G}_{35}^{(1)} \\ \mathbf{G}_{23}^{(0)} &= -\left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{23}^{(0)} \mathbf{G}_{33}^{(1)} - \left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{25}^{(0)} \left(\mathbf{G}_{35}^{(1)}\right)^T \\ \mathbf{G}_{22}^{(0)} &= \left(\mathbf{A}_{22}^{(0)}\right)^{-1} - \left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{23}^{(0)} \left(\mathbf{G}_{23}^{(0)}\right)^T - \left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{25}^{(0)} \left(\mathbf{G}_{25}^{(0)}\right)^T. \end{aligned}$$

Figure A.3 displays the sparsity of the resulting matrix $\mathbf{G}^{(0)}$. All the entries in $\mathbf{G}^{(0)}$ are equal to their corresponding entries in $\mathbf{G}^r$.

The computation of diagonal blocks in $\mathbf{G}^<$ are described for the same device. First, the matrix $\mathbf{N}$ is computed,

$$\mathbf{N} = \mathbf{\Sigma}^< \left(\mathbf{G}^<\right)^\dagger = \begin{bmatrix} \mathbf{\Sigma}_{11}^< \overline{\mathbf{G}_{11}^{(0)}} & \mathbf{0} & \mathbf{\Sigma}_{11}^< \overline{\mathbf{G}_{13}^{(0)}} & \mathbf{0} & \mathbf{\Sigma}_{11}^< \overline{\mathbf{G}_{15}^{(0)}} \\ \mathbf{0} & \mathbf{\Sigma}_{22}^< \overline{\mathbf{G}_{22}^{(0)}} & \mathbf{\Sigma}_{22}^< \overline{\mathbf{G}_{23}^{(0)}} & \mathbf{0} & \mathbf{\Sigma}_{22}^< \overline{\mathbf{G}_{25}^{(0)}} \\ \mathbf{\Sigma}_{33}^< \left(\mathbf{G}_{13}^{(0)}\right)^\dagger & \mathbf{\Sigma}_{33}^< \left(\mathbf{G}_{23}^{(0)}\right)^\dagger & \mathbf{\Sigma}_{33}^< \overline{\mathbf{G}_{33}^{(0)}} & \mathbf{0} & \mathbf{\Sigma}_{33}^< \overline{\mathbf{G}_{35}^{(0)}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}_{44}^< \overline{\mathbf{G}_{44}^{(0)}} & \mathbf{\Sigma}_{44}^< \overline{\mathbf{G}_{45}^{(0)}} \\ \mathbf{\Sigma}_{55}^< \left(\mathbf{G}_{15}^{(0)}\right)^\dagger & \mathbf{\Sigma}_{55}^< \left(\mathbf{G}_{25}^{(0)}\right)^\dagger & \mathbf{\Sigma}_{55}^< \left(\mathbf{G}_{35}^{(0)}\right)^\dagger & \mathbf{\Sigma}_{55}^< \left(\mathbf{G}_{45}^{(0)}\right)^\dagger & \mathbf{\Sigma}_{55}^< \overline{\mathbf{G}_{55}^{(0)}} \end{bmatrix}.$$

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

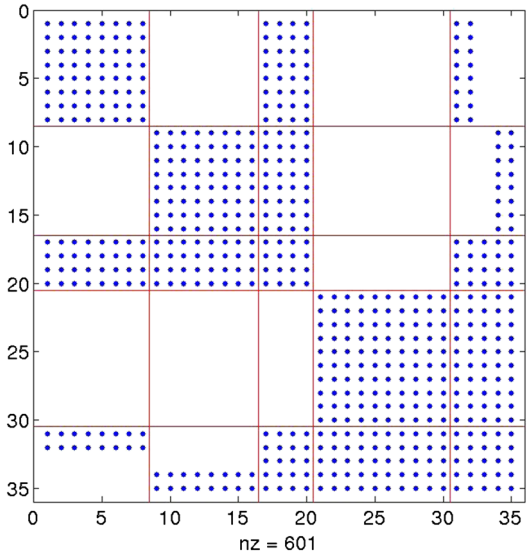


Figure A.3. Sparsity of matrix $\mathbf{G}^{(0)}$.

Set $\mathbf{N}^{(0)} = \mathbf{N}$. Next, the lower level clusters are folded into the higher ones to obtain the matrix $\mathbf{N}^{(1)}$

$$\mathbf{N}^{(1)} = \begin{bmatrix} \mathbf{N}_{11}^{(0)} & \mathbf{0} & \mathbf{N}_{13}^{(0)} & \mathbf{0} & \mathbf{N}_{15}^{(0)} \\ \mathbf{0} & \mathbf{N}_{22}^{(0)} & \mathbf{N}_{23}^{(0)} & \mathbf{0} & \mathbf{N}_{25}^{(0)} \\ \mathbf{N}_{31}^{(0)} & \mathbf{N}_{32}^{(0)} & \mathbf{N}_{33}^{(1)} & \mathbf{0} & \mathbf{N}_{35}^{(1)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{N}_{44}^{(0)} & \mathbf{N}_{45}^{(0)} \\ \mathbf{N}_{51}^{(0)} & \mathbf{N}_{52}^{(0)} & \mathbf{N}_{53}^{(1)} & \mathbf{N}_{54}^{(0)} & \mathbf{N}_{55}^{(1)} \end{bmatrix}$$

where the updated blocks are

$$\begin{aligned} \mathbf{N}_{33}^{(1)} &= \mathbf{N}_{33}^{(0)} - (\mathbf{A}_{13})^T \mathbf{A}_{11}^{-1} \mathbf{N}_{13}^{(0)} - (\mathbf{A}_{23})^T \mathbf{A}_{22}^{-1} \mathbf{N}_{23}^{(0)} \\ \mathbf{N}_{55}^{(1)} &= \mathbf{N}_{55}^{(0)} - (\mathbf{A}_{15})^T \mathbf{A}_{11}^{-1} \mathbf{N}_{15}^{(0)} - (\mathbf{A}_{25})^T \mathbf{A}_{22}^{-1} \mathbf{N}_{25}^{(0)} - (\mathbf{A}_{45})^T \mathbf{A}_{44}^{-1} \mathbf{N}_{45}^{(0)} \\ \mathbf{N}_{35}^{(1)} &= \mathbf{N}_{35}^{(0)} - (\mathbf{A}_{13})^T \mathbf{A}_{11}^{-1} \mathbf{N}_{15}^{(0)} - (\mathbf{A}_{23})^T \mathbf{A}_{22}^{-1} \mathbf{N}_{25}^{(0)} \\ \mathbf{N}_{53}^{(1)} &= \mathbf{N}_{53}^{(0)} - (\mathbf{A}_{15})^T \mathbf{A}_{11}^{-1} \mathbf{N}_{13}^{(0)} - (\mathbf{A}_{25})^T \mathbf{A}_{22}^{-1} \mathbf{N}_{23}^{(0)} \end{aligned}$$

For the top level, the block for region 5 is updated

$$\mathbf{N}^{(2)} = \begin{bmatrix} \mathbf{N}_{11}^{(0)} & \mathbf{0} & \mathbf{N}_{13}^{(0)} & \mathbf{0} & \mathbf{N}_{15}^{(0)} \\ \mathbf{0} & \mathbf{N}_{22}^{(0)} & \mathbf{N}_{23}^{(0)} & \mathbf{0} & \mathbf{N}_{25}^{(0)} \\ \mathbf{N}_{31}^{(0)} & \mathbf{N}_{32}^{(0)} & \mathbf{N}_{33}^{(1)} & \mathbf{0} & \mathbf{N}_{35}^{(1)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{N}_{44}^{(0)} & \mathbf{N}_{45}^{(0)} \\ \mathbf{N}_{51}^{(0)} & \mathbf{N}_{52}^{(0)} & \mathbf{N}_{53}^{(1)} & \mathbf{N}_{54}^{(0)} & \mathbf{N}_{55}^{(2)} \end{bmatrix}$$

with

$$\mathbf{N}_{55}^{(2)} = \mathbf{N}_{55}^{(1)} - \left(\mathbf{A}_{35}^{(1)}\right)^T \left(\mathbf{A}_{33}^{(1)}\right)^{-1} \mathbf{N}_{35}^{(1)}.$$

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

The next step is a block-diagonal multiplication

$$\mathbf{P}^{(2)} = \begin{bmatrix} \mathbf{A}_{11}^{-1}\mathbf{N}_{11}^{(0)} & \mathbf{0} & \mathbf{A}_{11}^{-1}\mathbf{N}_{13}^{(0)} & \mathbf{0} & \mathbf{A}_{11}^{-1}\mathbf{N}_{15}^{(0)} \\ \mathbf{0} & \mathbf{A}_{22}^{-1}\mathbf{N}_{22}^{(0)} & \mathbf{A}_{22}^{-1}\mathbf{N}_{23}^{(0)} & \mathbf{0} & \mathbf{A}_{22}^{-1}\mathbf{N}_{25}^{(0)} \\ \left(\mathbf{A}_{33}^{(1)}\right)^{-1}\mathbf{N}_{31}^{(0)} & \left(\mathbf{A}_{33}^{(1)}\right)^{-1}\mathbf{N}_{32}^{(0)} & \left(\mathbf{A}_{33}^{(1)}\right)^{-1}\mathbf{N}_{33}^{(1)} & \mathbf{0} & \left(\mathbf{A}_{33}^{(1)}\right)^{-1}\mathbf{N}_{35}^{(1)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{44}^{-1}\mathbf{N}_{44}^{(0)} & \mathbf{A}_{44}^{-1}\mathbf{N}_{45}^{(0)} \\ \left(\mathbf{A}_{55}^{(2)}\right)^{-1}\mathbf{N}_{51}^{(0)} & \left(\mathbf{A}_{55}^{(2)}\right)^{-1}\mathbf{N}_{52}^{(0)} & \left(\mathbf{A}_{55}^{(2)}\right)^{-1}\mathbf{N}_{53}^{(1)} & \left(\mathbf{A}_{55}^{(2)}\right)^{-1}\mathbf{N}_{54}^{(0)} & \left(\mathbf{A}_{55}^{(2)}\right)^{-1}\mathbf{N}_{55}^{(2)} \end{bmatrix}.$$

Finally, Step 4 extracts blocks one level at a time, defining first

$$\mathbf{P}^{(1)} = \begin{bmatrix} \mathbf{P}_{11}^{(2)} & \mathbf{0} & \mathbf{P}_{13}^{(2)} & \mathbf{0} & \mathbf{P}_{15}^{(2)} \\ \mathbf{0} & \mathbf{P}_{22}^{(2)} & \mathbf{P}_{23}^{(2)} & \mathbf{0} & \mathbf{P}_{25}^{(2)} \\ \mathbf{P}_{31}^{(2)} & \mathbf{P}_{32}^{(2)} & \mathbf{P}_{33}^{(1)} & \mathbf{0} & \mathbf{P}_{35}^{(1)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{44}^{(2)} & \mathbf{P}_{45}^{(2)} \\ \mathbf{P}_{51}^{(2)} & \mathbf{P}_{52}^{(2)} & \mathbf{P}_{53}^{(1)} & \mathbf{P}_{54}^{(2)} & \mathbf{P}_{55}^{(2)} \end{bmatrix}$$

where the updated blocks are

$$\begin{aligned} \mathbf{P}_{35}^{(1)} &= \mathbf{P}_{35}^{(2)} - \left(\mathbf{A}_{33}^{(1)}\right)^{-1}\mathbf{A}_{35}^{(1)}\mathbf{P}_{55}^{(2)} \\ \mathbf{P}_{53}^{(1)} &= -\left(\mathbf{P}_{35}^{(1)}\right)^{\dagger} \\ \mathbf{P}_{33}^{(1)} &= \mathbf{P}_{33}^{(2)} - \left(\mathbf{A}_{33}^{(1)}\right)^{-1}\mathbf{A}_{35}^{(1)}\mathbf{P}_{53}^{(1)}. \end{aligned}$$

Note that $\mathbf{P}^{(1)}$ is not skew Hermitian. However, finalized blocks, such as $\mathbf{P}_{33}^{(1)}$, $\mathbf{P}_{35}^{(1)}$, $\mathbf{P}_{53}^{(1)}$, and $\mathbf{P}_{55}^{(1)}$, satisfy the skew-Hermitian property. Finally, blocks for regions 1, 2, and 4 are updated, yielding the matrix $\mathbf{P}^{(0)}$,

$$\mathbf{P}^{(0)} = \begin{bmatrix} \mathbf{P}_{11}^{(0)} & \mathbf{0} & \mathbf{P}_{13}^{(0)} & \mathbf{0} & \mathbf{P}_{15}^{(0)} \\ \mathbf{0} & \mathbf{P}_{22}^{(0)} & \mathbf{P}_{23}^{(0)} & \mathbf{0} & \mathbf{P}_{25}^{(0)} \\ -\left(\mathbf{P}_{13}^{(0)}\right)^{\dagger} & -\left(\mathbf{P}_{23}^{(0)}\right)^{\dagger} & \mathbf{P}_{33}^{(1)} & \mathbf{0} & \mathbf{P}_{35}^{(1)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{44}^{(0)} & \mathbf{P}_{45}^{(0)} \\ -\left(\mathbf{P}_{15}^{(0)}\right)^{\dagger} & -\left(\mathbf{P}_{25}^{(0)}\right)^{\dagger} & -\left(\mathbf{P}_{35}^{(1)}\right)^{\dagger} & -\left(\mathbf{P}_{45}^{(0)}\right)^{\dagger} & \mathbf{P}_{55}^{(2)} \end{bmatrix}.$$

Blocks for region 4 are satisfying

$$\begin{aligned} \mathbf{P}_{45}^{(0)} &= \mathbf{P}_{45}^{(1)} - \left(\mathbf{A}_{44}^{(0)}\right)^{-1}\mathbf{A}_{45}^{(0)}\mathbf{P}_{55}^{(2)} \\ \mathbf{P}_{44}^{(0)} &= \mathbf{P}_{44}^{(1)} + \left(\mathbf{A}_{44}^{(0)}\right)^{-1}\mathbf{A}_{45}^{(0)}\left(\mathbf{P}_{45}^{(0)}\right)^{\dagger}. \end{aligned}$$

Blocks for region 1 are defined by

$$\begin{aligned} \mathbf{P}_{15}^{(0)} &= \mathbf{P}_{15}^{(1)} - \left(\mathbf{A}_{11}^{(0)}\right)^{-1}\mathbf{A}_{15}^{(0)}\mathbf{P}_{55}^{(2)} - \left(\mathbf{A}_{11}^{(0)}\right)^{-1}\mathbf{A}_{13}^{(0)}\mathbf{P}_{35}^{(1)} \\ \mathbf{P}_{13}^{(0)} &= \mathbf{P}_{13}^{(0)} - \left(\mathbf{A}_{11}^{(0)}\right)^{-1}\mathbf{A}_{13}^{(0)}\mathbf{P}_{33}^{(1)} + \left(\mathbf{A}_{11}^{(0)}\right)^{-1}\mathbf{A}_{15}^{(0)}\left(\mathbf{P}_{35}^{(1)}\right)^{\dagger} \\ \mathbf{P}_{11}^{(0)} &= \mathbf{P}_{11}^{(0)} + \left(\mathbf{A}_{11}^{(0)}\right)^{-1}\mathbf{A}_{13}^{(0)}\left(\mathbf{P}_{13}^{(0)}\right)^{\dagger} + \left(\mathbf{A}_{11}^{(0)}\right)^{-1}\mathbf{A}_{15}^{(0)}\left(\mathbf{P}_{15}^{(0)}\right)^{\dagger}, \end{aligned}$$

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

and blocks for region 2

$$\mathbf{P}_{25}^{(0)} = \mathbf{P}_{25}^{(1)} - \left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{25}^{(0)} \mathbf{P}_{55}^{(2)} - \left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{23}^{(0)} \mathbf{P}_{35}^{(1)}$$

$$\mathbf{P}_{23}^{(0)} = \mathbf{P}_{23}^{(0)} - \left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{23}^{(0)} \mathbf{P}_{33}^{(1)} + \left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{25}^{(0)} \left(\mathbf{P}_{35}^{(1)}\right)^{\dagger}$$

$$\mathbf{P}_{22}^{(0)} = \mathbf{P}_{22}^{(0)} + \left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{23}^{(0)} \left(\mathbf{P}_{23}^{(0)}\right)^{\dagger} + \left(\mathbf{A}_{22}^{(0)}\right)^{-1} \mathbf{A}_{25}^{(0)} \left(\mathbf{P}_{25}^{(0)}\right)^{\dagger}.$$

All the entries in $\mathbf{P}^{(0)}$ are equal to their corresponding entries in $\mathbf{G}^{<}$.

ACKNOWLEDGEMENTS

The authors acknowledge the support by the National Science Foundation under Grant ECCS-1231927. The authors thank the anonymous referees for comments that led to improvements of the manuscript.

REFERENCES

1. Anantram M, Lundstrom M, Nikonov D. Modeling of nanoscale devices. *Proceedings of the IEEE* 2008; **96**(9):1511–1550.
2. Datta S. Nanoscale device modeling: the Green's function method. *Superlattices and Microstructures* 2000; **28**:253–278.
3. Datta S. The non-equilibrium Green's function (NEGF) formalism: an elementary introduction. *International Electron Devices Meeting, 2002. IEDM'02*, IEEE, 2002; 703–706.
4. Haydock R, Heine V, Kelly M. Electronic structure based on the local atomic environment for tight-binding bands. *Journal of Physics C-Solid State Physics* 1972; **5**(20):2845.
5. Haydock R. *Solid State Physics*, Vol. 35. Academic Press: New York, NY, 1980.
6. Haydock R, Nex C. A general terminator for the recursion method. *Journal of Physics C-Solid State Physics* 1985; **18**(11):2235.
7. Lake R, Klimeck G, Bowen R, Jovanovic D. Single and multi-band modeling of quantum electron transport through layered semiconductor devices. *Journal of Applied Physics* 1997; **81**:7845–7869.
8. Sols F, Macucci M, Ravaioli U, Hess K. Theory for a quantum modulated transistor. *Journal of Applied Physics* 1989; **66**(8):3892–3906.
9. Lin L, Lu J, Ying L, Car R, E W. Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems. *Communications in Mathematical Sciences* 2009; **7**(3):755–777.
10. Lin L, Yang C, Meza J, Lu J, Ying L, E W. SelInv – an algorithm for selected inversion of a sparse symmetric matrix. *ACM Transactions on Mathematical Software* 2011; **37**(40):1–19.
11. Li S, Ahmed S, Klimeck G, Darve E. Computing entries of the inverse of a sparse matrix using the FIND algorithm. *Journal of Computational Physics* 2008; **227**:9408–9427.
12. Li S, Darve E. Extension and optimization of the FIND algorithm: computing Green's and less-than Green's functions. *Journal of Computational Physics* 2011; **231**:1121-1139.
13. George A. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis* 1973; **10**(2):345–363.
14. Mamaluy D, Sabathil M, Vogl P. Efficient method for the calculation of ballistic quantum transport. *Journal of Applied Physics* 2003; **93**(8):4628–4633.
15. Mamaluy D, Vasileska D, Sabathil M, Zibold T, Vogl P. Contact block reduction method for ballistic transport and carrier densities of open nanostructures. *Physical Review B* 2005; **71**(24):245321.
16. Li S, Wu W, Darve E. A fast algorithm for sparse matrix computations related to inversion. *Journal of Computational Physics* 2013; **242**:915–945.
17. Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing* 1998; **20**(1):359–392.
18. Svizhenko A, Anantram M, Govindam T, Biegel B, Venugopal R. Two-dimensional quantum mechanical modeling of nanotransistors. *Journal of Applied Physics* 2002; **91**:2343.
19. Petersen D, Li S, Stokbro K, Sorensen H, Hansen P, Skelboe S, Darve E. A hybrid method for the parallel computation of Green's functions. *Journal of Computational Physics* 2009; **228**:5020–5039.
20. Takahashi K, Fagan J, Chin MS. Formation of a sparse bus impedance matrix and its application to short circuit study. *Eighth PICA Conference*, Minneapolis, MN, 1973; 63–69.
21. Erisman AM, Tinney WF. On computing certain elements of the inverse of a sparse matrix. *Communications of the ACM* 1975; **18**(3):177–179.
22. Schwierz F. Graphene transistors. *Nature Nanotechnology* 2010; **5**:487–496.

Copyright © 2013 John Wiley & Sons, Ltd.

DOI: 10.1002/nme