Journal of Computational Physics 286 (2015) 49–61

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

An approximate framework for quantum transport calculation with model order reduction

Quan Chen [a,*], Jun Li [b], Chiyung Yam [c], Yu Zhang [b], Ngai Wong [a], Guanhua Chen [b]

[a] Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong
[b] Department of Chemistry, The University of Hong Kong, Hong Kong
[c] Beijing Computational Science Research Center, China

ARTICLE INFO

Article history:
Received 25 October 2014
Received in revised form 12 January 2015
Accepted 19 January 2015
Available online 28 January 2015

Keywords:
Model order reduction
Sparse matrices
NEGF
Non-equilibrium transport
Low-rank approximation

ABSTRACT

A new approximate computational framework is proposed for computing the non-equilibrium charge density in the context of the non-equilibrium Green's function (NEGF) method for quantum mechanical transport problems. The framework consists of a new formulation, called the X-formulation, for single-energy density calculation based on the solution of sparse linear systems, and a projection-based nonlinear model order reduction (MOR) approach to address the large number of energy points required for large applied biases. The advantages of the new methods are confirmed by numerical experiments.

1. Introduction

With the advent of 10 nm technology node, quantum mechanical (QM) phenomena have emerged as a central issue in modeling and simulation of nano-devices wherein numerical calculations based on first-principle QM physics have become indispensable. The non-equilibrium Green's function (NEGF) method [1,2] is the most widely used numerical tool for QM transport problems, providing an effective approach to solve the Schrödinger equation self-consistently with the Poisson equation. The most computation intensive step in NEGF is the calculation of charge density from the Green's function, which involves computing a subset of entries in the inverse of a large sparse matrix and has to be repeated many times. In this context, the recursive Green's function (RGF) method [3,4] has been the workhorse over years. With a mathematical origin in the semi-separable matrix [5], RGF allows efficient calculation of some blocks in the inverse of a block tri-diagonal matrix without forming the whole inverse. While effective for long and thin structures, RGF becomes inefficient when the target structures have a large cross-section. The complexity of RGF can be estimated as $O(N_x^3 N_y)$, where N_x is the average size of the matrix blocks treated by dense matrix algebra and is proportional to the layer size, and N_y is the number of the blocks in scale with the length. Therefore, the cubic growth of computation with the cross-sectional area renders RGF difficult to be used to simulate realistic devices wherein the cross-section of current-carrying channel easily exceeds 10×10 nm^2. To mitigate the performance bottleneck, efficient alternatives have been proposed, such as the FIND method [6,7] based on the nested dissection and the selected inversion (SelInv) [8,9] based on the hierarchical Schur complement. All of these methods

* Corresponding author. Tel.: +852 2219 4846.
E-mail address: quanchen@eee.hku.hk (Q. Chen).

http://dx.doi.org/10.1016/j.jcp.2015.01.032
0021-9991/

exhibit a lower asymptotic complexity than RGF by exploiting more advanced matrix partitioning techniques than the 1D partitioning used by RGF [10].

Another challenge facing NEGF in nano-device modeling lies in the large amount of energy points needed to evaluate the non-equilibrium (NEQ) charge density, due to the highly oscillatory integration along the real energy axis. Typical values used in practice are 400–500 points per eV [11]. Simulations involving large devices, as a consequence, are usually performed under small applied bias, e.g., 50 mV, so as to limit the number of energy points to a few dozen. However, realistic biases are in the range of $0.5 \sim 1$ V, and 10 times more energy points as well as computation are thereby demanded. On the other hand, since these energy points are "clustered" within a relatively small range, it is reasonable to expect that the solutions at these points are not truly independent of each other; they may have substantial overlap in the information they carry. Such information correlation has not been fully exploited by existing methods, in which each energy is solved individually.

Model order reduction (MOR) represents a family of mathematical techniques aiming to reduce the number of degrees of freedoms in a numerical system, while preserving important system properties such as I/O relations [12]. The rationale underlying MOR is that, with efforts spent in building a smaller system that to some extent mimics the original large system, significant speedup can be achieved by simulating this reduced-order-model (ROM), and recovering afterwards the responses of the original system from that of the ROM. Further saving is available if the ROM can be used repeatedly, e.g., in a design exploration phase. Originated from system control theory, MOR (or equivalents under different names) has numerous applications in many different scientific areas. Given the large number of repetitions of charge density evaluation in NEGF, MOR comes as a natural option to bring down the computation cost. However, most existing MOR techniques require the system of interest to be described in the state-space or descriptor representations [13]. The present formulation of NEQ charge density is not in these desired forms, and this hinders the application of MOR.

In this paper, we aim to develop an approximate computation framework for the NEQ calculation in NEGF. The method consists of two ingredients. The first ingredient is a new formulation of the single-energy density calculation, based on solving a sparse linear system with multiple right-hand-side (RHS) vectors. Different from RGF, FIND or SelInv which are all direct methods, the proposed formulation, dubbed X-formulation since it is based on solving $AX = b$, allows tradeoff between accuracy and computational cost. It also enables the use of iterative methods, a well-established field with abundant literature, to replace the direct LU (or LDLT) factorization for long-term scalability. More importantly, the X-formulation removes the obstacle to applying MOR by transforming the NEQ charge density calculation problem into a state-space representation. Hence, a MOR technique is developed to be the second ingredient of our framework. Because of the nonlinear nature of the problem, we choose a projection-based MOR method based on the parameter-dependent Krylov subspace (PDKS). The idea is to solve the full-size problem at a small number of energy points, use the solutions to construct a subspace basis, and perform the numerical quadratures with the ROM obtained by projecting the original system onto this subspace.

The rest of the paper is organized as follows. Section 2 will review the problem formulation for NEQ charge density calculation and the exact methods. Section 3 will describe the new X-formulation for single-energy calculation. Section 4 will give the details of the MOR algorithm. Section 5 will present numerical results and Section 6 will draw the conclusion.

2. Non-equilibrium charge calculation in NEGF

For simplicity, we assume a two-terminal device and do not include scattering mechanisms throughout the paper. The central quantity in NEGF is the (retarded) Green's function $G(E) \in \mathbb{C}^{N\times N}$ with N being the number of orbitals, which reads

$$G(E) = M(E)^{-1} = \left[ES - H - \Sigma_L(E) - \Sigma_R(E)\right]^{-1}, \tag{1}$$

where E is the energy (scalar), $S \in \mathbb{R}^{N\times N}$ is the overlap matrix (real symmetric) for non-orthogonal basis, $H \in \mathbb{R}^{N\times N}$ the Hamiltonian (real symmetric but indefinite) of the device under consideration and $\Sigma_{L,R}(E) \in \mathbb{C}^{N\times N}$ the self-energy matrices (complex symmetric) accounting for the influence of the left and right leads. Within the NEGF formalism $\Sigma_L(E)$ and $\Sigma_R(E)$ are nonlinear functions of E without closed-form expressions, and usually have only small $N_c \times N_c$ nonzero blocks at the top-left and the lower-right corner, respectively. Let $\mathcal{I}_L$ and $\mathcal{I}_R$ be two integer vectors containing the matrix indices corresponding to the left and right leads respectively, then the nonzero blocks of the self-energy matrices are defined by

$$\begin{aligned} \Sigma_L^{blk} &= \Sigma_L(\mathcal{I}_L, \mathcal{I}_L) = \tau_{DL} g_{LL} \tau_{DL}^\dagger \\ \Sigma_R^{blk} &= \Sigma_R(\mathcal{I}_R, \mathcal{I}_R) = \tau_{DR} g_{RR} \tau_{DR}^\dagger \end{aligned} \tag{2}$$

where $\tau_{DL} = Es_{DL} - h_{DL}$ and $\tau_{DR} = Es_{DR} - h_{DR}$ are the outer blocks coupling the leads and the device, and $g_{LL}, g_{RR} \in \mathbb{C}^{N_c\times N_c}$ are the surface Green's functions of the left and right leads, which can be regarded as a special type of Green's function "condensing" the influence of a semi-infinite lead to the layer immediately adjacent to the device. Superscript † denotes the conjugate transpose.

The NEQ Mulliken charge of each orbital is defined by the energy integral

$$Q_{neq} = -\frac{2}{\pi} \int_{E_b}^{E_t} dE\, q(E) \tag{3}$$

where E_b and E_t are respectively the bottom and top energy levels for the integral. The single-energy electron density consists of three parts

$$q(E) = diag\{G^<(E)S\} - diag\{\widetilde{G}^<_{DL}(E)s^T_{DL}\} - diag\{\widetilde{G}^<_{DR}(E)s^T_{DR}\} \tag{4}$$

where $G^<(E)$ is the lesser Green's function of the device defined by (assuming higher potential in the right lead)

$$G^<(E) = G(E)\Gamma_R(E)G(E)^\dagger \tag{5}$$

with $\Gamma_R = \mathrm{Im}(\Sigma_R)$. The last two terms in (4) involve the outer blocks of the lesser Green's function, $\widetilde{G}^<_{DL}, \widetilde{G}^<_{DR} \in \mathbb{C}^{N\times N_c}$ and the outer blocks of the overlap matrix, $s_{DL}, s_{DR} \in \mathbb{R}^{N_c\times N_c}$. Notice that $\widetilde{G}^<_{LD}$ is a tall matrix containing only one nonzero block $G^<_{DL} \in \mathbb{C}^{N_c\times N_c}$ at the top, i.e., $\widetilde{G}^<_{DL}(\mathcal{I}_L, :) = G^<_{DL}$. Likewise, $\widetilde{G}^<_{DR}(\mathcal{I}_R, :) = G^<_{DR}$ has only one nonzero block at the bottom. The outer blocks in (4) are defined as

$$\begin{aligned}
\widetilde{G}^<_{DL}(E) &= G(E)\Gamma_R(E)G(E)^\dagger\widetilde{\tau}_{DL}(E)g_{LL}(E)^\dagger \\
\widetilde{G}^<_{DR}(E) &= G(E)\Gamma_R(E)G(E)^\dagger\widetilde{\tau}_{DR}(E)g_{RR}(E)^\dagger + G(E)\widetilde{\tau}_{DR}(E)\,\mathrm{Im}\big(g_{RR}(E)\big)
\end{aligned} \tag{6}$$

Again $\widetilde{\tau}_{DL}$ and $\widetilde{\tau}_{DR}$ are tall matrices with $\widetilde{\tau}_{DL}(\mathcal{I}_L, :) = \tau_{DL}$ and $\widetilde{\tau}_{DR}(\mathcal{I}_R, :) = \tau_{DR}$.

The integral (3) is commonly evaluated by numerical integration

$$Q_{neq} = -\frac{2}{\pi}\sum_i^{N_E} q(E_i)w_i \tag{7}$$

where N_E evaluation points are selected within $[E_b, E_t]$ according to a specific quadrature rule, such as the Gauss–Legendre quadrature, and w_i are the corresponding weights (the scalar Fermi function is incorporated into w_i and omitted throughout the paper).

Computing the diagonals of the matrix products in (4) is the most time-consuming step for large problems, and several approaches have been proposed to accelerate this step. RGF is the most prevailing one wherein the computational grid is partitioned into layers along one direction (usually the transport direction), and M becomes a symmetric block tri-diagonal matrix. Then any block in G can be computed by a recursive algorithm using the tri-diagonal blocks of M. The known problems with RGF include the quasi-1D assumption and the cubically growing computation with the block size. The FIND [6,7] and SelInv [8,9] methods are more advanced alternatives to RGF, by partitioning the grid points into arbitrarily shaped clusters organized in a binary tree [10]. That way, the minimum block size is no longer limited by the layer size, and can be made much smaller for computational efficiency. Nevertheless, the methods mentioned above are all exact methods that do not allow tradeoff between accuracy and performance, and they rely on sparse block LU factorization which may have scalability issue when applied to truly large problems. In the next section, we will propose a new approximate formulation for NEQ density calculation which is based on solving sparse linear systems directly.

3. A new formulation of NEQ charge density based on solution of sparse linear systems

The new formulation begins with processing the Γ_R matrix in (5), which contains only one $N_c \times N_c$ nonzero block Γ^{blk}_R at the lower right corner. We assume real E to make τ_{DR} a real matrix and plug it in (2), then

$$\Gamma^{blk}_R = \mathrm{Im}\big(\Sigma^{blk}_R\big) = \tau_{DR}\,\mathrm{Im}(g_{RR})\tau^\dagger_{DR} \tag{8}$$

We exploit the fact that Γ^{blk}_R is real symmetric, and thus admits a symmetric Schur decomposition

$$\Gamma^{blk}_R = \widetilde{U}\widetilde{D}\widetilde{U}^\dagger$$

where $\widetilde{U}$ is a unitary matrix and $\widetilde{D}$ a diagonal matrix containing the eigenvalues[1] of Γ^{blk}_R. A truncation is then performed to drop the eigenvalues smaller than a prescribed threshold (rank tolerance ϵ_{rank}) and the corresponding columns in $\widetilde{U}$. Suppose p eigenvalues are kept, the rank-p approximation of Γ^{blk}_R is

$$\Gamma^{blk}_R(E) \approx U(E)D(E)U(E)^\dagger \tag{9}$$

where $U \in \mathbb{R}^{N_c\times p}$ has unitary columns, $U^\dagger U = I$, and the rank-p approximation of Γ_R reads

$$\Gamma_R(E) \approx Y(E)D(E)Y(E)^\dagger, \qquad Y(E) = \begin{bmatrix} \mathrm{O} \\ U(E) \end{bmatrix} \tag{10}$$

with O being an $(N - N_c) \times p$ zero matrix. The assumption here is that we can drop some less important information contained in the self-energy matrix and still obtain a good approximation to the Γ to be used in charge density calculations.

[1] The absolute values of the eigenvalues are the singular values in this case.

Algorithm 1: Charge density calculation with X-formulation.

begin
  $Q \leftarrow 0$
  for $j \leftarrow 1$ **to** N_E **do**
    Obtain $U(E_j), D(E_j)$ via low-rank approximation (9)
    $\beta_j \leftarrow U(E_j)^\dagger \tau_{DR}(E_j)\,\mathrm{Im}(g_{RR}(E_j))$
    $D_j \leftarrow w_j D(E_j)$, $\beta_j \leftarrow w_j \beta(E_j)$
    $M_j \leftarrow E_j S - H - \Sigma(E_j)$
    $X_j \leftarrow M_j^{-1} U(E_j)$
    $X_j^L \leftarrow X_j(\mathcal{I}_L, :)$ and $X_j^R \leftarrow X_j(\mathcal{I}_R, :)$
    $Q \leftarrow Q + \mathrm{sum}(X_j .* (D_j X_j^\dagger S)^T, 2)$
    $Q(\mathcal{I}_L) \leftarrow Q(\mathcal{I}_L) - \mathrm{sum}(X_j^L .* (D_j X_j^{L\dagger} \tau_{DL}(E_j) g_{LL}(E_j)^\dagger s_{DL}^T)^T, 2)$
    $Q(\mathcal{I}_R) \leftarrow Q(\mathcal{I}_R) - \mathrm{sum}(X_j^R .* ((D_j X_j^{R\dagger} \tau_{DR}(E_j) g_{RR}(E_j)^\dagger + \beta_j) s_{DR}^T)^T, 2)$
  $Q \leftarrow -\frac{2}{\pi} Q$

Substituting (10) into (5) and (6), we obtain

$$\begin{aligned} G^<(E) &\approx \big[G(E)Y(E)\big] D(E) \big[G(E)Y(E)\big]^\dagger \\ \widetilde{G}^<_{LD}(E) &\approx \big[G(E)Y(E)\big] D(E) \big[G(E)Y(E)\big]^\dagger \widetilde{\tau}_{DL}(E) g_{LL}(E)^\dagger \\ \widetilde{G}^<_{DR}(E) &\approx \big[G(E)Y(E)\big] D(E) \big[G(E)Y(E)\big]^\dagger \widetilde{\tau}_{DR}(E) g_{RR}(E)^\dagger \\ &\quad + G(E) \widetilde{\tau}_{DR}(E)\,\mathrm{Im}\big(g_{RR}(E)\big) \end{aligned} \tag{11}$$

The key quantity to compute in (11) is the solution of the following sparse linear system

$$X(E) = G(E)Y(E) = \big[ES - H - \Sigma_L(E) - \Sigma_R(E)\big]^{-1} Y(E) \tag{12}$$

With $X^L = X(\mathcal{I}_L, :)$, $X^R = X(\mathcal{I}_R, :)$ denoting the top and bottom $N_c \times p$ blocks of X, (11) becomes

$$\begin{aligned} G^<(E) &\approx X(E) D(E) X(E)^\dagger \\ G^<_{DL}(E) &\approx X^L(E) D(E) X^L(E)^\dagger \tau_{DL}(E) g_{LL}(E)^\dagger \\ G^<_{DR}(E) &\approx X^R(E) D(E) X^R(E)^\dagger \tau_{DR}(E) g_{RR}(E)^\dagger + X^R(E) \beta(E) \end{aligned} \tag{13}$$

where $G^<_{DL}$ and $G^<_{DR}$ are the nonzero blocks of $\widetilde{G}^<_{DL}$ and $\widetilde{G}^<_{DR}$, respectively. The β in the last equation is defined by

$$\beta(E) = U(E)^\dagger \tau_{DR}(E)\,\mathrm{Im}\big(g_{RR}(E)\big) \tag{14}$$

the derivation of which is discussed in Appendix A. Eqs. (13) are the main results of the new formulation, which is named the *X-formulation* for its base on the solution X's of a series of sparse linear systems with block RHS. Note that if no low-rank approximation is applied to Γ_R and (12) is solved exactly, the X-formulation becomes an exact method.

After X is obtained, the electron density vector $q(E)$ in (4) is approximated by

$$\begin{aligned} q &\approx diag\big\{X D X^\dagger S\big\} \\ q(\mathcal{I}_L) &\approx q(\mathcal{I}_L) - diag\big\{X^L D X^{L\dagger} \tau_{DL} g_{LL}^\dagger s_{DL}^T\big\} \\ q(\mathcal{I}_R) &\approx q(\mathcal{I}_R) - diag\big\{\big(X^R D X^{R\dagger} \tau_{DR} g_{RR}^\dagger + X^R \beta\big) s_{DR}^T\big\} \end{aligned} \tag{15}$$

The main computation boils down to computing the diagonals of the outer product of two $N \times p$ matrices, in the generic form of $diag\{X_1 X_2^\dagger\}$, which can be done at a lower complexity than the brute-force $O(N^2 p)$. For instance, the first term in (15) can be computed by (under Matlab notations)

$$diag\big\{X D X^\dagger S\big\} = \mathrm{sum}\big(X .* \big(D X^\dagger S\big)^T, 2\big) \tag{16}$$

where $.*$ represents the element-wise multiplication and $\mathrm{sum}(\ldots, 2)$ denotes the summation along rows. The complexity of (16) is reduced to $O(Np)$. Non-diagonal entries of the lesser Green's function can also be readily obtained once X is available. The flow to compute the electron density Q with the X-formulation is summarized in Algorithm 1.

Several notes are important at this point:

1. The X-formulation uses the low-rank factors of Γ as the RHS when solving the linear systems. Therefore, the number of RHS vectors, i.e., the rank used in the low-rank approximation, is important for the performance, and serves as an adjustable parameter to cater for different accuracy and speed requirements in different applications. In typical

two-terminal devices without scattering, only one nonzero block in Γ needs to be factorized, and its size (the size of the contact with higher chemical potential) is generally small compared to whole structure. When scattering mechanisms are included, Γ tends to become block (or tri-block) diagonal, and the X-formulation remains applicable with a low-rank approximation applied to the whole Γ, but may need to work with an increased number of RHS vectors. In addition, the low-rank approximation is performed after the self-energy matrix is formed by an exact method in this work. It would be interesting to investigate the possibility to approximate self-energy matrices in low-rank factor form in the first place using iterative methods such as [14].

2. Both direct and iterative methods can be used to solve the sparse linear systems in (12), rendering more flexibility in the X-formulation than in RGF and its variants. When direct methods are used, the X-formulation possesses a comparable complexity with FIND and SelInv as they are all based on the LDLT factorization of complex symmetric matrices. On the other hand, state-of-the-art iterative solvers such as GMRES [15] and COCG [16] are also readily applicable with the X-formulation, which may be the only viable option for extremely large structures in term of time and memory complexity. In addition, with a block RHS and a form closed to shifted linear system (if Σ is linear in E), the solutions of the equation systems (12) are expected to have linearly dependent components among RHSs and energies, and as such the subspace needed to capture all the solutions can be significantly smaller than the lumped size of that required to solve one system with one RHS at a time. Iterative methods exploiting this property, for instance the shifted COCG method [17], can be promising candidates to be used in conjunction with the X-formulation. Parallelization may also be more straightforward with the iterative methods.
3. The X-formulation based on (12) is essentially in a simplified parametric state-space form with the time derivative term being zero. The low-rank factor of Γ serves as the "input" of the system, which is reasonable in physics since the self-energies "capsulate" the influence of outer environment. X can be viewed as the "internal states" and also the "output" needed to extract the quantities of interest, e.g., the electron density. In this regard, the X-formulation opens new avenues for introducing well-established MOR techniques into QM calculations.

4. A nonlinear MOR scheme

The most expensive step of the X-formulation is finding the solution X with (12), which has to be repeated for a prescribed set of energy points, i.e.,

$$X_j = \left[E_j S - H - \Sigma_L(E_j) - \Sigma_R(E_j)\right]^{-1} Y(E_j) = M_j^{-1} Y_j, \quad j = 1, 2, \ldots, N_E \tag{17}$$

When the applied bias is high (~ 1 V), $N_E \approx 500$ energies may be needed to approximate the oscillatory integration. Solving X_j hundreds of times poses a substantial challenge even with the state-of-the-art computing resources for large-scale problems. In this section a nonlinear MOR scheme is developed to effectively reduce the number of energies where the full-size problems are solved.

4.1. Basic projection-based MOR scheme

The NEQ integration has a feature that a dense sampling is applied to a relatively small interval (compared to the equilibrium case). As such, the solutions X's of all energies may have substantial overlap in information and can be approximated to a reasonable extent by a smaller number of sampling points. To achieve this, we select $m \ll N_E$ energy points as the interpolation points, at which X is solved exactly (or to a sufficiently high accuracy). We assume these solutions are linearly independent and thus span the subspace

$$\mathcal{K}_m = \operatorname{span}\left\{M(E_1)^{-1} Y(E_1), M(E_2)^{-1} Y(E_2), \ldots, M(E_m)^{-1} Y(E_m)\right\} \tag{18}$$

The subspace above is called the parameter-dependent Krylov subspace (PDKS) [18], which has nonlinear dependency on E both in M and Y due to the nonlinear self-energies.

Let $V_m = [v_1, v_2, \ldots, v_m] \in \mathbb{R}^{N \times \widetilde{n}}$, $\widetilde{n} = \sum_j^m p_j$, be an orthonormal basis constructed from the m accurate solutions above, the solution of (12) is approximated by solving a reduced-order system obtained from a projection with V_m

$$X_m(E) = V_m M_m(E)^{-1} Y_m(E) \tag{19}$$

in which

$$\begin{aligned} M_m(E) &= \left(M(E)V_m\right)^{\dagger} M(E) V_m \\ Y_m(E) &= \left(M(E)V_m\right)^{\dagger} Y(E) \end{aligned} \tag{20}$$

Note that the Petrov–Galerkin condition is enforced in (20) to minimize the residual over V_m

$$\left\| R_m(E) \right\|_F = \left\| M(E) X_m(E) - Y(E) \right\|_F = \left\| M(E) V_m M_m(E)^{-1} Y_m(E) - Y(E) \right\|_F \tag{21}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

Algorithm 2: MOR with predetermined interpolation points.

Data: $H, S, \Sigma, \mathcal{I}_{intp}$
Result: Non-equilibrium charge density Q

1 **begin**
2 **for** $m \leftarrow 1$ **to** m_{intp} **do**
3 $j \leftarrow \mathcal{I}_{intp}(m)$
4 $M(E_j) \leftarrow E_j S - H - \Sigma(E_j)$
5 $X_m \leftarrow M(E_j)^{-1} Y(E_j)$
   // Modified Gram-Schmidt orthogonalization
6 **if** $m == 1$ **then**
7 $V_1 \leftarrow qr(X_1)$
8 **else**
9 $V_m \leftarrow orth([V_{m-1}, X_m])$
10 $W \leftarrow 0$, $W_{DL} \leftarrow 0$ and $W_{DR} \leftarrow 0$
11 **for** $j \leftarrow 1$ **to** N_E **do**
12 $\beta_j \leftarrow U(E_j)^\dagger \tau_{DR}(E_j) \mathrm{Im}(g_{RR}(E_j))$
13 $D_j \leftarrow w_j D(E_j)$, $\beta_j \leftarrow w_j \beta(E_j)$
14 Construct $M_m(E_j)$ and $Y_m(E_j)$ via (20)
15 Solve the reduced system $Z_m(E_j) \leftarrow M_m(E_j)^{-1} Y_m(E_j)$
16 $X_j^L \leftarrow V_m(\mathcal{I}_L, :) Z_m(E_j)$ and $X_j^R \leftarrow V_m(\mathcal{I}_R, :) Z_m(E_j)$
17 $W \leftarrow W + Z_m(E_j) D_j Z_m(E_j)^\dagger$
18 $W_{DL} \leftarrow W_{DL} + X_j^L (D_j X_j^{L\dagger} \tau_{DL}(E_j) g_{LL}(E_j)^\dagger)$
19 $W_{DR} \leftarrow W_{DR} + X_j^R (D_j X_j^{R\dagger} \tau_{DR}(E_j) g_{RR}(E_j)^\dagger + \beta_j)$
20 $Q \leftarrow -\frac{2}{\pi} \mathrm{sum}(V_m W .* (V_m^\dagger S)^T, 2)$
21 $Q(\mathcal{I}_L) \leftarrow \rho(\mathcal{I}_L) - (-\frac{2}{\pi}) \mathrm{sum}(W_{DL} .* s_{DL}, 2)$
22 $Q(\mathcal{I}_R) \leftarrow \rho(\mathcal{I}_R) - (-\frac{2}{\pi}) \mathrm{sum}(W_{DR} .* s_{DR}, 2)$

The basic MOR method with a predetermined set of interpolation points is given in Algorithm 2. The vector $\mathcal{I}_{intp}$ stores the m_{intp} selected interpolation points. The modified Gram-Schmidt scheme is employed to orthogonalize V_m. Note that in Line 20–22, the multiplications with the S matrices are done only *one time* after the second *for* loop, which is much more efficient than having S multiplied with X every step within the *for* loop as in the non-MOR Algorithm 1. This is because V_m is energy independent, and the E-dependent matrices $Z_m D Z_m^\dagger \in \mathbb{C}^{\widetilde{n} \times \widetilde{n}}$ are generally of small sizes and can be formed and added to W efficiently within the loop. In contrast, it is prohibitive to evaluate and store $XDX^\dagger$ of the original size during the *for* loop in Algorithm 1,[2] and thus the multiplications with the S matrices must be performed immediately before relevant information is lost.

The PDKS in (18) is generated by adding p_j basis vectors at a time, where p_j is the number of columns of $U(E_j)$. Therefore a linear system with p_j RHSs needs to be solved each time, which may still be expensive for large p_j. On the other hand, not all RHS vectors have equal contribution to the error reduction, and it is natural to ask whether we can select the most important RHS vectors based on their effectiveness in reducing error. In addition, although choosing interpolation points a priori is convenient, in practice determining them with high quality is difficult, if not impossible, for general nonlinear problems. In the following section, adaptive approaches are developed to optimize the selection of the RHS vectors and the interpolation points in the MOR algorithm.

4.2. Adaptive selection of RHS vectors and interpolation points

Firstly, to generate the RHS matrix that is cost-effective in error reduction, we extend the tangential interpolation idea applied in the rational Krylov subspace [19] to the parametric case. The tangential interpolation refers to that, instead of solving $M_j^{-1} Y_j$ with p_j RHS columns, we solve $M_j^{-1} Y_j d_j$, with $d_j \in \mathbb{C}^{p_j \times l_j}$, $l_j \leq p_j$ being a "selector matrix" to cut the number of RHSs from p_j to l_j. In other words, the interpolation is not exact, but along some tangential direction d_j, at the point j. The tangential PDKS thus becomes

$$\mathcal{T}_m = \mathrm{span}\left\{ M(E_1)^{-1} Y(E_1) d_1, M(E_2)^{-1} Y(E_2) d_2, \ldots, M(E_m)^{-1} Y(E_m) d_m \right\} \tag{22}$$

Following [19], we obtain d_j from the singular value decomposition (SVD) of the residual $R_{j-1}(E_j)$. Specifically, d_j is chosen as the right singular vectors corresponding to the l_j largest singular values to maximize $\| R_{j-1}(E_j) d_j \|$. Our implementation determines l_j according to a user-defined SVD tolerance (ϵ_{svd}), such that

$$\sigma_{l_j} < \epsilon_{svd} \sigma_1 \tag{23}$$

[2] Although it is possible to compute and store only the entries in $XDX^\dagger$ that will be needed in subsequent multiplication with S, the computational cost remains much higher than the evaluation of $Z_m D Z_m^\dagger$.

where $\sigma_1, \sigma_2, \ldots$ are the singular values of $R_{j-1}(E_j)$ in the descending order. Note that this tangential reduction is performed on top of the low-rank approximation of Γ (10) used by the X-formulation, and thus the full-size solution in the interpolation stage of MOR involves generally fewer RHSs than in the X-formulation at the same energy.

The second issue concerns the selection of the interpolation points. More interpolation points help produce high-quality ROMs to benefit the subsequent integration, but at the price of solving more systems of the original size. For more regular problems wherein $M(E)$ are exactly shifted linear systems, e.g., $M(E) = A - EB$, priori schemes exist for choosing the interpolation points (or shifts) of optimal asymptotic convergence by solving the third Zolotaryov problem [20]. However, such schemes are not applicable in this case due to the nonlinear, non-analytic, self-energy terms. An alternate scheme was proposed in [21], which chooses the interpolation points, within a given spectral interval, in a greedy fashion to minimize the residuals of the approximated systems. In this paper, we extend the greedy search approach to select the next interpolation point based on the residual estimate of the current approximation.

Given V_m the projection basis generated at the mth step, an optimal implementation is to compute the residual estimates via (21) for all the unselected points, among which the one with the largest residual is chosen as the interpolation point at the $m+1$th step. However, each residual evaluation with (21) involves forming the dense tall matrix MV_m, computing the inner product to generate M_m (20) and solving a dense matrix equation $M_m^{-1}Y_m$. When the dimension of M_m is large, checking residuals for hundreds of points may become very expensive. To speed up the point selection procedure, following strategies are adopted in our implementation:

1. The interpolation process starts with a set of $m_{init} > 1$ initial interpolation points $\mathcal{I}_{init}$, chosen equidistantly in the indices. After each residual sweeping, $m_{new} > 1$ points $\mathcal{I}_{new}$ with the largest residuals are selected and appended to the interpolation point vector $\mathcal{I}_{intp}$. A new check of residuals is employed only after these m_{new} points are solved, and thus the *number of times to perform residual checking* is largely reduced. However, this is a suboptimal scheme. To see this, suppose $\mathcal{I}_{new} = [j_1, j_2, \ldots, j_{m_{new}}]$ with descending residuals are picked, then adding new basis vectors from the solution at E_{j_1} will affect the residuals at all the remaining E's, in particular it tends to suppress most the residuals in the vicinity of E_{j_1}. Thus there is no guarantee that $E_{j_2}, \ldots, E_{j_{m_{new}}}$ remain meaningful choices for the subsequent steps, if they are closed to E_{j_1} and already have their residuals reduced by the solution at E_{j_1}. In other words, we hope that $E_{j_2}, \ldots, E_{j_{m_{new}}}$ would still be, or at least close to, the ones that would be selected in the one-point-at-a-time strategy in later stage. To this end, we force the candidate points in $\mathcal{I}_{new}$ to be separated by a minimal distance d_{min}. This way, the interpolation points are more likely to be allocated to the needed regions.
2. The second strategy is to reduce the *number of points participating in a residual checking*. Remind that in (20) the Petrov–Galerkin projection is applied, which is chosen deliberately over the standard Galkerkin projection to ensure the residuals drop monotonically with the expansion of projection basis. Consequently, one can safely exclude the points that have residuals already smaller than a given residual tolerance (ϵ_{res}) in the residual sweep as the residuals can only become smaller later on, and fewer points need to be checked at later stage of the algorithm when the residual evaluation becomes more expensive.
3. Finally, the *individual residual evaluation* in (21) can be sped up by an incremental update of R_m reusing the results from previous steps. In Appendix B, we present an efficient update scheme for R_m based on incremental LDLT factorization, which dramatically reduces the computation by only working on the newly generated data in each step.

The adaptive algorithm for choosing RHS and interpolation points is given in Algorithm 3.

There are several tolerances serving for different purposes in the X-formulation + MOR framework. The rank tolerance ϵ_{rank} controls the number of RHSs in solving linear systems in the X-formulation: higher ϵ_{rank} means fewer RHSs but lower accuracy in approximating the impact of self-energies. The number of RHSs also affects the time needed to perform matrix–matrix multiplications in the second *for* loop in Algorithm 2. The SVD tolerance ϵ_{svd} determines the number of RHSs solved in the interpolation phase of MOR, generally smaller than the one selected by ϵ_{rank} in the X-formulation. Smaller ϵ_{svd} tends to add more vectors to the projection basis V at a time and is likely to reduce the number of interpolation points needed to achieve the same accuracy, yet requires more computation time and may result in larger ROMs. The residual tolerance ϵ_{res} directly controls the balance between accuracy and efficiency of the MOR method.

Finally, the MOR method presented above can be viewed as one particular way to reuse the information shared among the solutions at different energies, as discussed in Section 3, but by no means is the only way or the most efficient one. Investigations for more effective strategies, e.g., smarter choices of interpolation points, are highly desired to design faster algorithms for QM calculations. In addition, the current MOR method works with a given set of quadrature points determined a priori by a quadrature rule. It would be interesting to incorporate the MOR scheme into the adaptive integration technique such as [22] to gain further saving.

5. Numerical experiments

In this section we test the proposed computation framework using three 3D silicon nanowires (SiNW) of different sizes. Table 1 details the specifications of the three SiNWs and Fig. 1 plots the 3D structure of SiNw2. The coding is done in Matlab and the tests were performed on a machine with 2.7 GHz CPU and 32 GB memory. To simplify the discussion, in this paper we use the direct solver PARDISO [23] to solve all the linear systems, yet iterative solvers are readily applicable

Algorithm 3: Adaptive selection of RHS vectors and interpolation points.

Data: $H, S, \Sigma, m_{max}, \mathcal{I}_{init}$
Result: Projection basis V_m
begin
  $\mathcal{I}_{intp} \leftarrow \mathcal{I}_{init}$, $m_{intp} \leftarrow m_{init}$
  $\mathcal{I}_{remain} \leftarrow \mathcal{I}_{all} \backslash \mathcal{I}_{intp}$, $m_{remain} \leftarrow N_E - m_{intp}$
  for $m \leftarrow 1$ **to** m_{max} **do**
    if $m \leq m_{intp}$ **then**
      $j \leftarrow \mathcal{I}_{intp}(m)$
    else
      for $k \in \mathcal{I}_{remain}$ **do** `// Estimate residuals`
        Compute $RES(k) = \|R_m(E_k)\|_F$ via (21) using V_{m-1}
      if $\max(RES) \leq \epsilon_{res}$ **then**
        Exit
      else
        Select m_{new} interpolation points $\mathcal{I}_{new}$ based on certain strategy
        $\mathcal{I}_{intp} \leftarrow [\mathcal{I}_{intp}, \mathcal{I}_{new}]$ `// update` $\mathcal{I}_{intp}$
        $m_{intp} \leftarrow m_{intp} + m_{new}$
        $j \leftarrow \mathcal{I}_{intp}(m)$
        Find $\mathcal{I}_{small}$ where $RES(\mathcal{I}_{small}) < \epsilon_{res}$
        $RES(\mathcal{I}_{small}) \leftarrow 0$
        $\mathcal{I}_{remain} \leftarrow \mathcal{I}_{remain} \backslash [\mathcal{I}_{new}, \mathcal{I}_{small}]$ `// No need to check residuals at these points`
    `// Point` j `will be the next interpolation point`
    $M(E_j) \leftarrow E_j S - H - \Sigma(E_j)$
    Compute the leading right singular vectors of $R_m(E_j)$ to define d_j
    $Y_m = [0, U(E_j)d_j]^T$
    $X_m \leftarrow M(E_j)^{-1} Y_m$
    if $m == 1$ **then**
      $V_1 \leftarrow qr(X_1)$
    else
      $V_m \leftarrow orth([V_{m-1}, X_m])$

Table 1
Sizes of silicon nanowires.

Case	Cross sec. (nm^2)	Length (nm)	# of atoms	Matrix size	Contact size N_c
SiNW1	1.5×1.5	50	3264	15,252	166
SiNW2	5×2.5	25	12,540	53,268	1244
SiNW3	8×8	20	54,800	203,000	4060

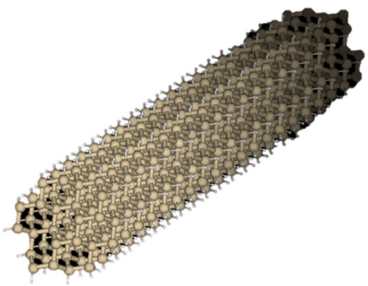

Fig. 1. 3D picture of SiNW2.

if better performance can be achieved. A small imaginary part of 10^{-5} is added to all the energies to avoid the possible singularity of $M(E)$.

5.1. Performance of X-formulation

We start with comparing the performance of the X-formulation against RGF for single-energy calculation with the three examples in Table 2. The absolute rank tolerance ϵ_{rank} for approximating Γ_R is set to be 10^{-6} and the error is measured by $\|q - q_{RGF}\|_F / \|q_{RGF}\|_F$. All the solutions are obtained at $E = -2.1529$ eV. As expected, RGF exhibits a low efficiency when the cross-section of a structure is large, with approximately a cubic increase in computation and a quadratic increase in

Table 2
Performance of X-formulation ($\epsilon_{rank} = 10^{-6}$) compared with RGF.

Case	Method	N_x in RGF	# of RHS	Runtime (s)	Memory (Mb)	Error
SiNW1	RGF	166	–	2.6	325	–
	X-formula	–	23	1.1	200	2.0E−07
SiNW2	RGF	1244	–	294	4900	–
	X-formula	–	201	32	1350	4.8E−07
SiNW3	RGF	4060	–	17,334	51,000	–
	X-formula	–	1038	1071	7200	4.2E−07

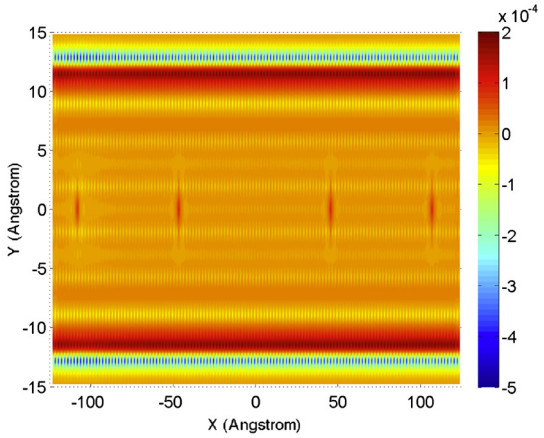

Fig. 2. Electron density of the SiNW2 example.

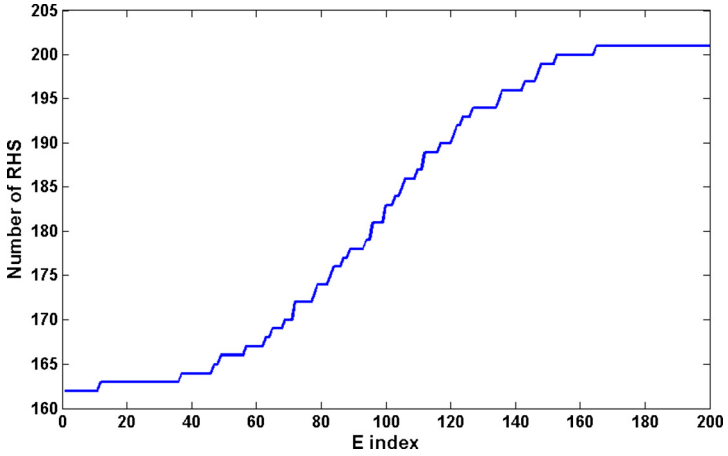

Fig. 3. Number of RHSs at different energy points (SiNW2, $\epsilon_{rank} = 10^{-6}$).

memory with respect to N_x. The X-formulation, on the other hand, shows better time and memory scalability, with an over 16× speedup for the largest case. The accuracy is reasonable with the errors all below ϵ_{rank}.

Next we compute the electron density of SiNW2 by an energy integration with 0.5 eV bias in $[-2.652904, -2.152904]$ eV, and 200 energy points chosen by the Gauss–Legendre quadrature. The calculated electron density on the middle x–y plane is plotted in Fig. 2. In Fig. 3, we show the numbers of RHSs at the 200 points for the same $\epsilon_{rank} = 10^{-6}$. As can be seen, more RHSs are kept for the energies near the right end of the spectrum, since the right lead has higher contribution to the charge density distribution in the device. Because of the low-rank compression, the RHS number is much smaller than the size of the nonzero block (4060) in the self-energy matrix, which is one main contributor of the numerical advantages of the X-formulation.

Table 3 shows the performance of X-formulation for different rank tolerances with the same set of 200 points. The number of RHSs is averaged over the 200 points, and the total runtime includes the time of the PARDISO solver and the time of the subsequent matrix–matrix multiplications. Overall, the X-formulation is found to be nearly one-order faster than RGF, and allows tradeoff between accuracy and efficiency in different applications by adjusting ϵ_{rank}. Larger ϵ_{rank} results in

Table 3
X-formulation with different rank tolerances for SiNW2 with 200 points.

Method	ϵ_{rank}	# of RHS (avg.)	PARDISO time (s)	Total time (s)	Error
RGF	–	–	–	58,845	–
X-Form	10^{-4}	4	3667	3793	1.00E−04
X-Form	10^{-5}	102	4856	5110	3.60E−06
X-Form	10^{-6}	182	5817	6180	9.70E−07

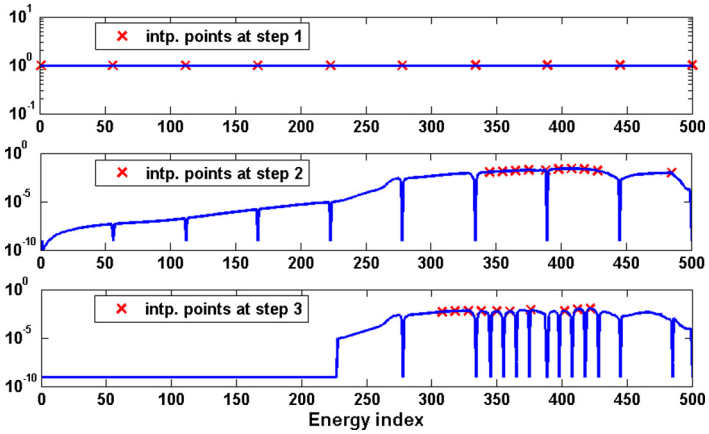

Fig. 4. Interpolation point selection in the first three steps of MOR (SiNW1). For better visualization, the residuals at the points selected in the previous steps, together with those already smaller than ϵ_{res} (all included in $\mathcal{I}_{small}$ in Algorithm 3), are all set to be 10^{-9}.

fewer RHS vectors but less accurate solution, with an error generally below the chosen ϵ_{rank}. It is also observed that the performance of direct solver is not very sensitive to the number of RHSs, i.e., the runtime of PARDISO is increased by only 1.3× when the number of RHSs increases from 4 to 102, which can be explained by the fact that the main cost of direct solvers is on the matrix factorization and the number of RHSs only affects the runtime in the back substitution step of a lower complexity.

5.2. Performance of MOR

We first visualize in Fig. 4 the point selection scheme presented in Section 4.2 using SiNW1. The integration is performed in $[-2.652904, -1.652904]$ eV with 1 eV bias and 500 quadrature points. After each residual sweeping $m_{new} = 10$ points are selected with the minimal separation $d_{min} = 10$. Fig. 4 shows the residuals of all energy points at the first three steps, where the interpolation points selected are marked by red crosses. In the 1st step, the points are distributed equidistantly over the whole range. In the 2nd step, the residuals at the points selected in the 1st round have been reduced small, and 10 new points are put to the places where the residuals are the largest, subject to the constraint that each point must be separated from others by at least d_{min} points. Similar selection is done in the 3rd step. Note that, after step 2, some points at the left end already have residuals smaller than ϵ_{rank}, therefore these points are in the "safe zone" and will not participate in the residual search of step 3.

The performance of MOR is recorded in Table 4 with SiNW1 and SiNW2 using the integration setting as mentioned above. T_{slc}, T_{PAR} and T_{int} refer to the time spent on the point selection, the PARDISO solutions and the numerical integration using the ROM, respectively. The error is measured against the reference solution obtained by solving the X-formulation directly at all the energy points. A 5× saving is achieved for SiNW1, wherein only 77 full-size systems are solved instead of 500, and the integration is performed with a ROM of size 408 in contrast to the original size of 15,252. The reduction in the number of full-size solutions is less significant for SiNW2, where 88 out of 200 energy points are required. However the runtime saving in solving linear systems, 1669 s vs. 6180 s, is higher than the 2.5× saving in the number of energies, which can be explained by the fact that fewer RHSs are involved in each full-size solution in the interpolation phase of MOR, thanks to the adaptive selection of RHSs described in Section 4.2. Fig. 5 compares the RHS numbers needed in the non-adaptive and the adaptive schemes. Except at the first energy the two schemes solve the same number of RHSs, at the remaining points selected for interpolation, adaptive scheme uses a much smaller number of RHSs because the singular values of residuals drop rapidly and only a few of them deserve to be kept. This results in faster interpolation and smaller ROMs.

6. Conclusion

In this paper we present a new approximate framework for NEGF transport calculation. The two key ingredients are the X-formulation for single-energy charge density calculation based on solving sparse linear systems with block RHSs, and the

Table 4
Performance of MOR. The same set of parameters are used: $\epsilon_{rank} = 10^{-6}$, $\epsilon_{res} = 10^{-5}$, $\epsilon_{svd} = 10^{-2}$, $m_{init} = m_{new} = 10$ and $d_{min} = 10$.

Case	# of E	non-MOR time (s)	# of intp. points	ROM size	MOR time (s) ($T_{slc} + T_{PAR} + T_{int}$)	Error
SiNW1	500	660	77	408	123 (27 + 80 + 16)	2.7E−6
SiNW2	200	6180	88	1337	2375 (374 + 1669 + 332)	1.8E−5

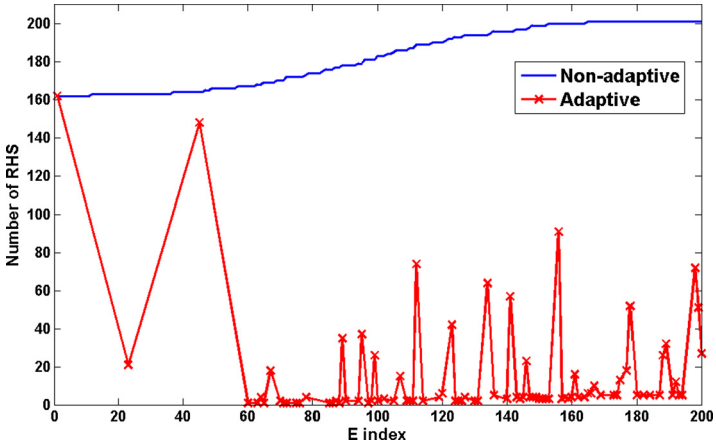

Fig. 5. Number of RHSs without and with adaptive selection (SiNW2, $\epsilon_{rank} = 10^{-6}$).

MOR method to address the difficulty arising from many energy points required for large biases. The former exploits matrix sparsity directly, enabling tradeoff between accuracy and performance, and transforms the problem into a state-space form that allows MOR to be applied. The MOR method then aims to reuse the information shared among individual solutions by building a small ROM from a small fraction of energy points. A series of acceleration techniques, including the adaptive selection of RHS vectors and interpolation points and the incremental update of residual estimates, is developed to further enhance the performance of MOR. The numerical experiments confirm the advantages of the proposed framework.

Acknowledgements

The authors would like to acknowledge the support from Hong Kong University Grant Council (AoE/P-04/08) and Hong Kong General Research Fund (GRF) project HKU710913E. The authors would also like to thank Prof. Michael Ng in the Department of Mathematics, The Hong Kong Baptist University, for the valuable discussions and suggestions.

Appendix A. Formation of β

Hereafter the dependency on E is dropped when it is clear from the context. The motivation to introduce β in (13) is to reuse the same solution X^R, generated using U as the RHS, to compute the second term for $\widetilde{G}^<_{DR}$ in (11). To this end, we approximate $\tau_{DR}\,\mathrm{Im}(g_{RR})$ by its orthogonal projection onto U

$$\tau_{DR}\,\mathrm{Im}(g_{RR}) = UU^\dagger \tau_{DR}\,\mathrm{Im}(g_{RR}) \tag{A.1}$$

whereby β is derived as the "coordinate" matrix under the basis U

$$\beta = U^\dagger \tau_{DR}\,\mathrm{Im}(g_{RR})$$

It is observed in our experiments that (A.1) is generally a good approximation provided that $UDU^\dagger$ is a good approximation of Γ_R. However, unlike the well-known error bound for SVD-based rank-k approximation

$$\left\| \Gamma_R - UDU^\dagger \right\|_F = \sum_{i=k+1}^{N_c} \sigma_i, \tag{A.2}$$

where σ_i are the singular values sorted in the descending order, the error bound for (A.1) is less obvious. Hence we ensure the accuracy of (A.1) by a numerical means, wherein we measure the projection error by

$$\left\| U_N U_N^\dagger \tau_{DR}\,\mathrm{Im}(g_{RR}) \right\|_F \tag{A.3}$$

where U_N denotes the orthogonal complement of U and is available in the Schur decomposition (8). If the error is large, we move some vectors from U_N to U to enlarge the projection subspace to suppress the error. In our experiments fewer than 50 vectors need to be added to U in the worst case, and only the result of $Q(\mathcal{I}_R)$ will be slightly affected. One may also avoid this complication by applying another low-rank approximation to $\tau_{DR}\,\mathrm{Im}(g_{RR})$ directly, and solving the additional RHS vectors thereby generated. As demonstrated in Table 3, the increase in cost is expected to be mild since matrix factors in direct methods or preconditioners in iterative methods can be reused.

Appendix B. Incremental calculation of residuals

Since only the bottom block of the RHS matrix Y is nonzero, we also evaluate the residual at the same block for faster computation. This is justified in that the residual estimates need not to be exact, and the saving from a faster residual evaluation can easily outweigh the cost of solving slightly more points due to a less optimal selection scheme. Therefore the residual used in the residual sweeping reads

$$R_m^{blk} = M(\mathcal{I}_R, :)V_m M_m^{-1} Y_m - U \tag{B.1}$$

We focus on accelerating the two most expensive operations, namely, forming M_m and solving $M_m^{-1}Y_m$. For the first part, we aim to build M_m in an incremental manner by reusing the result from previous steps. Recall that

$$M_m = (MV_m)^\dagger MV_m = \big(M[V_{m-1}, v_m]\big)^\dagger M[V_{m-1}, v_m] = \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^\dagger & M_{22} \end{bmatrix} \tag{B.2}$$

where V_{m-1} is the aggregated basis vectors from the first $m-1$ steps, and v_m contains the new vectors generated at the mth step. The blocks in M_m are given by

$$\begin{aligned} M_{11} &= M_{m-1} = (MV_{m-1})^\dagger MV_{m-1} \\ M_{12} &= (MV_{m-1})^\dagger Mv_m \\ M_{22} &= (Mv_m)^\dagger Mv_m \end{aligned} \tag{B.3}$$

in which M_{m-1} is the old matrix from the last step, and M_{12} and M_{22} are the new blocks we need to compute. In addition, M_{12} and M_{22} can be obtained efficiently by minimizing the amount of computation involving E-dependent self-energy matrices. Take M_{12} for instance

$$\begin{aligned} &(MV_{m-1})^\dagger Mv_m = (ESV_{m-1} - HV_{m-1} - \Sigma V_{m-1})^\dagger (ESv_m - Hv_m - \Sigma v_m) = A_1 - A_2 \\ &A_1 = |E|^2 (SV_{m-1})^\dagger Sv_m - \bar{E}(SV_{m-1})^\dagger Hv_m - E(Sv_m)^\dagger HV_{m-1} + (HV_{m-1})^\dagger Hv_m \\ &A_2 = \left(ESV_{m-1} - HV_{m-1} - \frac{1}{2}\Sigma V_{m-1}\right)^\dagger (\Sigma v_m) + (\Sigma V_{m-1})^\dagger \left(ESv_m - Hv_m - \frac{1}{2}\Sigma v_m\right) \end{aligned} \tag{B.4}$$

The A_1 term does not involve Σ and is linear in E, thus one can pre-compute the E-independent matrices priori to the residual checking and perform only scalar-matrix multiplications within the loop. Meanwhile SV_m, HV_m can be expanded incrementally in the same fashion as in (B.2). The nonlinear A_2 term has to be evaluated for each E. However, both ΣV_{m-1} and Σv_m have only two nonzero blocks (the top and bottom blocks), so it suffices to compute only the corresponding blocks in the other two terms, which reduces the whole computation to four block matrix–matrix multiplications of relatively small sizes. Again, one can grow ΣV_m incrementally by storing the two nonzero blocks. The efficient computation of M_{22} follows analogously.

For $M_m^{-1}Y_m$, we apply the block LU-update [24] to generate the block factorization of M_m, which reduces to block LDLT-update in this case since M_m is Hermitian. Let M_{11} in (B.2) have the LDLT factorization $ldl(M_{11}) = L_{11}D_{11}L_{11}^\dagger$,[3] then the block factorization of the augmented matrix can be updated as

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{12}^\dagger & M_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} D_{11} & \\ & D_{22} \end{bmatrix} \begin{bmatrix} L_{11} & \\ L_{21} & L_{22} \end{bmatrix}^\dagger \tag{B.5}$$

where

$$L_{21}^\dagger = D_{11}^{-1} L_{11}^{-1} M_{12}, \qquad L_{22} = ldl\big(M_{22} - L_{21}D_{11}L_{21}^\dagger\big) \tag{B.6}$$

In other words, if L_{11}, D_{11} are stored from the previous step, the new L, D factors can be obtained easily by one back substitution and one LDLT of a small matrix with the newly generated M_{12} and M_{22}.

Despite being fast, the incremental residual update has a drawback that it needs to store the intermediate matrices, such as ΣV_m, for all the energies remaining in the residual checklist. Strategies to lower memory usage, e.g., the low-rank tensor approximation [25] given the 3-dimensional data structure, will be investigated in the future.

[3] In practice a permutation matrix P is usually needed to make L truly low-triangular, i.e., $M_{11} = P_{11}L_{11}D_{11}(P_{11}L_{11})^\dagger$, but its incorporation is straightforward and can be updated in the same way as D.

References

[1] S. Datta, Nanoscale device modeling: the Green's function method, Superlattices Microstruct. 28 (4) (2000) 253–278, http://dx.doi.org/10.1006/spmi.2000.0920.

[2] S. Datta, The non-equilibrium Green's function (NEGF) formalism: an elementary introduction, in: International Electron Devices Meeting, 2002, pp. 703–706.

[3] R. Haydock, V. Heine, M.J. Kelly, Electronic structure based on the local atomic environment for tight-binding bands, J. Phys. C, Solid State Phys. 5 (20) (1972) 2845, http://dx.doi.org/10.1088/0022-3719/5/20/004.

[4] F. Sols, M. Macucci, U. Ravaioli, K. Hess, Theory for a quantum modulated transistor, J. Appl. Phys. 66 (8) (1989) 3892–3906, http://dx.doi.org/10.1063/1.344032.

[5] J. Jain, H. Li, S. Cauley, C.-K. Koh, V. Balakrishnan, Numerically stable algorithms for inversion of block tridiagonal and banded matrices, Tech. rep., Purdue, 2007.

[6] S. Li, S. Ahmed, G. Klimeck, E. Darve, Computing entries of the inverse of a sparse matrix using the FIND algorithm, J. Comput. Phys. 227 (22) (2008) 9408–9427, http://dx.doi.org/10.1016/j.jcp.2008.06.033.

[7] S. Li, E. Darve, Extension and optimization of the FIND algorithm: computing Green's and less-than Green's functions, J. Comput. Phys. 231 (4) (2012) 1121–1139, http://dx.doi.org/10.1016/j.jcp.2011.05.027.

[8] L. Lin, J. Lu, L. Ying, R. Car, E. Weinan, Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems, Commun. Math. Sci. 7 (3) (2009) 755–777.

[9] L. Lin, C. Yang, J.C. Meza, J. Lu, L. Ying, W.N. E, SelInv—an algorithm for selected inversion of a sparse symmetric matrix, ACM Trans. Math. Softw. 37 (4) (2011) 40:1–40:19, http://dx.doi.org/10.1145/1916461.1916464.

[10] U. Hetmaniuk, Y. Zhao, M. Anantram, A nested dissection approach to modeling transport in nanodevices: algorithms and applications, Int. J. Numer. Methods Eng. 95 (7) (2013) 587–607, http://dx.doi.org/10.1002/nme.4518.

[11] J. Maassen, M. Harb, V. Michaud-Rioux, Y. Zhu, H. Guo, Quantum transport modeling from first principles, Proc. IEEE 101 (2) (2013) 518–530, http://dx.doi.org/10.1109/JPROC.2012.2197810.

[12] W.H. Schilders, H.A. Van der Vorst, J. Rommes, Model Order Reduction: Theory, Research Aspects and Applications, vol. 13, Springer, 2008.

[13] S. Tan, L. He, Advanced Model Order Reduction Techniques in VLSI Design, Cambridge University Press, 2007.

[14] H.H.B. Sørensen, P.C. Hansen, D.E. Petersen, S. Skelboe, K. Stokbro, Krylov subspace method for evaluating the self-energy matrices in electron transport calculations, Phys. Rev. B 77 (15) (2008) 155301, http://dx.doi.org/10.1103/PhysRevB.77.155301.

[15] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869, http://dx.doi.org/10.1137/0907058.

[16] H. Van der Vorst, J.B.M. Melissen, A Petrov–Galerkin type method for solving $Axk = b$, where A is symmetric complex, IEEE Trans. Magn. 26 (2) (1990) 706–708, http://dx.doi.org/10.1109/20.106415.

[17] R. Takayama, T. Hoshi, T. Sogabe, S.-L. Zhang, T. Fujiwara, Linear algebraic calculation of the Green's function for large-scale electronic structure theory, Phys. Rev. B 73 (16) (2006) 165108, http://dx.doi.org/10.1103/PhysRevB.73.165108.

[18] M. Zaslavsky, V. Druskin, Solution of time-convolutionary Maxwell's equations using parameter-dependent Krylov subspace reduction, J. Comput. Phys. 229 (12) (2010) 4831–4839, http://dx.doi.org/10.1016/j.jcp.2010.03.019.

[19] V. Druskin, V. Simoncini, M. Zaslavsky, Adaptive tangential interpolation in rational Krylov subspaces for MIMO dynamical systems, SIAM J. Matrix Anal. Appl. 35 (2) (2014) 476–498, http://dx.doi.org/10.1137/120898784.

[20] V. Druskin, L. Knizhnerman, M. Zaslavsky, Solution of large scale evolutionary problems using rational Krylov subspaces with optimized shifts, SIAM J. Sci. Comput. 31 (5) (2009) 3760–3780, http://dx.doi.org/10.1137/080742403.

[21] V. Druskin, C. Lieberman, M. Zaslavsky, On adaptive choice of shifts in rational Krylov subspace reduction of evolutionary problems, SIAM J. Sci. Comput. 32 (5) (2010) 2485–2496, http://dx.doi.org/10.1137/090774082.

[22] O. Baumgartner, M. Karner, S. Holzer, M. Pourfath, T. Grasser, H. Kosina, Adaptive energy integration of non-equilibrium Green's functions, in: Proceedings of the 2007 NSTI Nanotechnology Conference, vol. 3, 2007, pp. 145–148.

[23] O. Schenk, et al., PARDISO 5.0.0 Solver Project, http://www.pardiso-project.org/, 2014.

[24] H.M. Huynh, A large-scale quadratic programming solver based on block-LU updates of the KKT system, Ph.D. thesis, Stanford University, 2008.

[25] T. Kolda, B. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500, http://dx.doi.org/10.1137/07070111X.