Automatic differentiable nonequilibrium Green's function formalism: An end-to-end differentiable quantum transport simulator

Zhanghao Zhouyin,[1] Xiang Chen,[2] Peng Zhang,[1,*] Jun Wang,[3,†] and Lei Wang[4]

[1]*College of Intelligence and Computing, Tianjin University, Tianjin 300354, China*
[2]*Noah's Ark Lab, Huawei, Beijing 100085, China*
[3]*University College London, London WC1E 6BT, United Kingdom*
[4]*Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China*

(Received 6 June 2023; accepted 25 October 2023; published 27 November 2023)

The state-of-the-art first-principles quantum transport theory and modeling are based on carrying out self-consistent atomistic calculations within the Keldysh nonequilibrium Green's function (NEGF) formalism. The atomistic model of the device can be at the tight-binding (TB) or the density functional theory levels, and NEGF determines the nonequilibrium carrier distribution under external bias and gate voltages. In this work, we report an end-to-end automatic differentiable NEGF simulator (AD-NEGF) within the NEGF-TB framework. AD-NEGF calculates gradient information by automatic differentiation (AD) and the implicit layer technique while guaranteeing the correctness of forward simulation. The gradient information enables accurate calculations of transport properties that depend on the derivatives of the transmission coefficient and/or charge current. More interestingly, AD-NEGF can be applied to the extremely interesting inverse design problem; namely, with a desired transport property, AD-NEGF inversely finds a possible device Hamiltonian that would produce such a property.

DOI: 10.1103/PhysRevB.108.195143

I. INTRODUCTION

Quantum transport theory provides fundamental understandings of device physics and scientific background knowledge of practical modeling tools for predicting carrier transport in electronic devices [1–3]. The state-of-the-art first-principles quantum transport theory is based on carrying out atomistic analysis within the Keldysh nonequilibrium Green's function (NEGF) formalism [4,5]. Here, the atomic model of the device can be at the tight-binding (TB) level or the density functional theory (DFT) level to capture material details of the device system. The density matrix of the device is constructed by NEGF which provides the nonequilibrium distribution of the carriers under the external bias/gate potentials for the open device structure. A self-consistent NEGF-TB or NEGF-DFT procedure solves the nonequilibrium quantum transport properties including all the atomistic details of the device. Such NEGF methods have been widely applied for device physics and are part of the larger industrial tool set of technology computer-aided design (TCAD) [6–8].

In practical applications of first-principles device modeling, after the self-consistent NEGF simulation is converged, one obtains physical quantities such as the transmission functions $T = T(E)$, where E is the carrier energy, and the electric current $I = I(V)$, where V is the externally applied bias/gate voltages, etc. With $T(E)$, $I(V)$, important physical or device parameters that depend on their derivatives can be further calculated. These include the Seebeck coefficient of thermoelectric devices [9], differential conductance of tunneling spectroscopy [10], and subthreshold swing of a metal-oxide-semiconductor field-effect transistor [11]. To calculate the parametric derivatives of $T(E)$ for instance, a dense energy mesh is usually required especially when T is a rapidly varying function of E. In industrial TCAD, one resorts to *compact models* which are analytical models of the carrier transport, in which many measured, fitted, and/or phenomenological parameters are used to achieve accuracy. For analytical models, the derivatives can be easily done. For situations where the analytical models do not exist or are difficult to establish, it will be very useful to develop an approach that directly predicts the derivatives of the transport functions without doing brute-force numerical differentiation. In addition, for device physics, being able to predict derivatives or gradients is important in high-dimensional optimization of the device models which is related to the inverse problem of property-by-design.

For this purpose, here we report an end-to-end differentiable quantum transport simulator. The automatic differentiable NEGF (hereafter called AD-NEGF) is at the level of NEGF-TB. AD-NEGF calculates gradient information efficiently by automatic differentiation (AD) and implicit layer techniques (see below) while guaranteeing the correctness of forward simulation. The gradient information enables precise calculation of differential physical quantities directly and allows model optimization at a complexity level not achievable by conventional approaches. To the best of our knowledge, an end-to-end differentiable quantum transport simulator has not been reported in the literature before.

*pzhang@tju.edu.cn
†jun.wang@cs.ucl.ac.uk

Our AD-NEGF is inspired by recent progress in artificial intelligence (AI) for quantum transport and differentiable programming. So far, machine-learning-based AI techniques have been applied to train neural networks with data generated from first-principles transport simulations. Here, the neural network serves as an efficient surrogate model to make predictions of conductance [12–14] and other transport coefficients [15]. As AI for quantum transport is still in the early stages of development, relatively simple deep-learning models such as multilayer perceptrons [16] and convolutional networks [17–19] were typically used, although more advanced and specially designed models started to appear [12]. Regarding differentiable programming, in our context, it refers to embedding physical models or numerical computation processes into the AI model to improve data efficiency, generalization capability, and interpretability. It requires an automatic differentiation framework to support implicit numerical operations such as fixed-point iterations [20], optimization [21], and initial value problems [22]. Differentiable programming has been applied to physical simulations [23,24] such as rigid-body dynamics [25,26], computational fluid dynamics [27–29], ray tracing [30], and quantum many-body simulations [31]. More specifically, in *ab initio* simulations, there have been differentiable programming in density functional theory [32,33], Hartree-Fock methods [34], coupled-cluster expansions [35], and molecular dynamics [36]. In the rest of this paper, we present a differentiable programming technique for quantum transport simulations.

We apply AD-NEGF to several situations. First, we demonstrate its ability to accurately and efficiently compute differential physical properties. Second, we demonstrate that combining AD-NEGF with gradient-based optimization can help solve the inverse problem of transport-by-design. Third, as another inverse problem, we apply AD-NEGF to find possible Slater-Koster tight-binding (SKTB) parameters of impurity dopants to reach a predetermined goal of transmission coefficient. We show that AD-NEGF gains significant advantages in these problems over conventional approaches. The rest of the paper is organized as follows. In the next section, we present details of AD-NEGF. Section III summarizes the applications of AD-NEGF. A short summary is reserved for Sec. IV.

II. AD-NEGF: THEORY AND IMPLEMENTATION

Our NEGF-TB transport method in AD-NEGF is implemented in PyTorch [37], so that we can utilize the tensor operations in PyTorch linalg and other modules, including multiplication, addition, and inversion. Our code also includes an SKTB module that generates a block-tridiagonal TB Hamiltonian [38] of the device material. The back-propagation process in AD-NEGF is improved through the use of implicit gradient techniques, the adjoint sensitivity method for partial differential equations (PDEs), and an image charge gradient method.

A. NEGF-TB transport method

Before discussing the AD process in the next section, it is helpful to briefly present our NEGF implementation on which the AD is applied. The NEGF first-principles quantum transport formalism is based on performing atomistic material-specific calculations within the NEGF framework [5]. The atomistic model can be at the level of DFT or TB as mentioned above. This work is based on using the TB Hamiltonian for the device material. The idea of the NEGF-TB or NEGF-DFT is to calculate the Hamiltonian of the device under the influence of external bias and gate voltages. In the case of NEGF-TB, the equilibrium Hamiltonian is parametrized by TB parameters and the electrostatic potential due to external electric field is calculated self-consistently. In the case of NEGF-DFT, the entire device Hamiltonian, including the effects of external electric field, is calculated self-consistently. On the other hand, the application of NEGF determines the nonequilibrium statistical information for constructing the density matrix. Typically, real-space numerical methods are used to handle transport and electrostatic boundary conditions of the open device structure. Since its first report [5], the NEGF based atomistic modeling methods have become the de facto standard approach for simulating nonequilibrium quantum transport in atomistic nanostructures. For more technical details we refer interested readers to Ref. [39] and in the rest of this section, we outline our implementation on which the AD process is developed. Some further details are summarized in Appendix A.

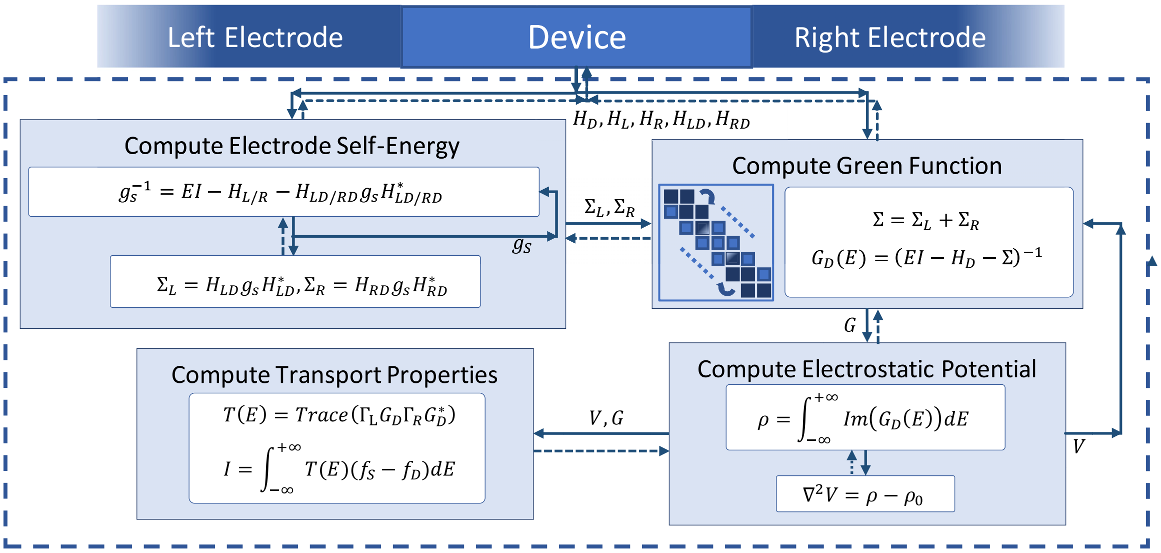
Consider a transport system made of a device scattering region and two semi-infinite electrodes that attach to the left and right sides of it, shown in Fig. 1. The Hamiltonian H of the entire system, device and electrodes, is represented with a TB model [40], which is in a block-tridiagonal form. We express the NEGF model under an orthogonal atomic basis here for brevity. The stationary Schrödinger equation of the *infinitely large* open device structure is $H\Psi = E\Psi$, where Ψ is the wave function and E the corresponding energy. The Green's function of the system is formally obtained as

$$G = [EI - H]^{-1}, \tag{1}$$

where I is the identity matrix. Note that for the open device structure, the Hamiltonian H is in fact an infinitely large matrix; thus the Green's function G cannot be directly obtained from Eq. (1). This problem is resolved by computing the Green's function only for the device scattering region G_D via its Hamiltonian H_D, and the electronic degrees of freedom in the two semi-infinite electrodes are integrated out, resulting in the self-energy Σ terms which are added to H_D. G_D is solvable since it is finite. The Green's function G_D and its conjugate construct the Keldysh NEGF $G^<$ via the Keldysh equation will give the nonequilibrium charge distribution in the device [39]. With the charge distribution, Poisson's equation for the electrostatic potential in the device scattering region V_D is solved which updates the Hamiltonian H_D. This process is self-consistently iterated to numerical convergence. Following this standard procedure [5,39], G_D is obtained by inverting a finite matrix,

$$G_D = [EI - H_D - \Sigma]^{-1}. \tag{2}$$

Since the matrix to be inverted can be cast into block-tridiagonal form due to the short-range-ness of the TB potential, we apply the efficient recursive algorithm in [41] to obtain G_D. In Eq. (2), the self-energy Σ due to the two



Left Electrode
Device
Right Electrode
$H_D, H_L, H_R, H_{LD}, H_{RD}$
Compute Electrode Self-Energy
$g_s^{-1} = EI - H_{L/R} - H_{LD/RD} g_s H^*_{LD/RD}$
g_S
$\Sigma_L = H_{LD} g_s H^*_{LD}, \Sigma_R = H_{RD} g_s H^*_{RD}$
Σ_L, Σ_R
Compute Green Function
$\Sigma = \Sigma_L + \Sigma_R$
$G_D(E) = (EI - H_D - \Sigma)^{-1}$
G
Compute Transport Properties
$T(E) = Trace(\Gamma_L G_D \Gamma_R G_D^*)$
$I = \int_{-\infty}^{+\infty} T(E)(f_S - f_D) dE$
V, G
Compute Electrostatic Potential
$\rho = \int_{-\infty}^{+\infty} Im(G_D(E)) dE$
$\nabla^2 V = \rho - \rho_0$
V

FIG. 1. Workflow of AD-NEGF. Solid lines indicate the forward simulation flow, where loops denote self-consistent iterations. Dashed lines indicate the gradient back-propagation flow.

electrodes of the device can be calculated using the surface Green's function technique [42], the details of which are summarized in Appendix A 1.

An electrostatic potential V_D in the device scattering region is established due to external bias/gate voltages applied to the device. V_D is solved self-consistently via Poisson's equation and added to the H_D,

$$\nabla \cdot \epsilon(r)\nabla[\Delta V_D(r)] = -[\rho(r;\Delta V_D) - \rho_0(r)],$$
$$\Delta V_D(r)|_{\{z_L,z_R\}} = \{V_L, V_R\}, \tag{3}$$

where V_L and V_R are boundary conditions at electrodes z_L and z_R, and $\Delta V_D = V_D - V_0$ is the difference between the real potential and the equilibrium one. The charge density ρ on the right-hand side is obtained by NEGF; the details are summarized in Appendix A 2. In our Poisson's equation solver, an efficient image charge approach based on the fast multipole method (FMM) [43,44] is used. After V_D is obtained, it is added to H_D to calculate Green's functions and the process is repeated until self-consistency.

Once the $G_D - V_D$ self-consistency reaches a small numerical tolerance, we calculate the transmission coefficient $T(E)$ by the Landauer formula and further the I-V curves $I(V)$ by integrating the $T(E)$ over the bias window. Details of the implementation are summarized in Appendix A 3.

B. Differentiable NEGF

Our differentiable NEGF model is implemented in PyTorch [37]. Automatic differentiation is achieved by implementing each operation in NEGF as a tensor operation in PyTorch. Therefore, when executing a program, PyTorch will automatically track the tensor operations to build a computational graph according to their calling orders. The corresponding gradient can be computed by running backward through the computational graph based on the chain rule. An illustration of the forward pass and backward pass can be found in Fig. 1, where the solid lines indicate the forward computation procedure and the dashed lines show the back-propagation procedure.

Moreover, implementing NEGF also includes several numerical processes that do not work straightforwardly with the automatic differentiation framework of PyTorch. Their implementation via tensor operation is either unavailable in the PyTorch framework (e.g., Poisson's equation solver) or includes iterative processes which will make the computation graph too large to track (e.g., self-consistent calculation of charge density and surface Green's function). For this purpose, we implemented several customized gradient computation methods through the torch.autograd. Function interface, which requires implementing both the forward and the backward methods. The mechanism works as follows: (1) We implement the numerical algorithm in the forward method. The operation in the forward method is not limited to tensor operation, as long as the input and output are PyTorch tensors. (2) The gradient calculation of the numerical algorithm, which is often derived analytically, needs to be implemented in the backward function. Through this, the whole numerical computation can cooperate with the PyTorch automatic differentiation system. To derive the analytical gradient of the numerical steps, we apply implicit gradient techniques for self-consistent iterations and the adjoint sensitivity method for the solvers of Poisson's equation [45]. In addition, an efficient gradient formula for the image charge method [43]—accelerated by FMM—is developed to accelerate the gradient calculation. This formulation can be regarded as a summation of point charges produced by the gradients which can also be computed with FMM. The detailed derivation and their implementation are illustrated in the rest of this section.

Regarding the implicit gradient technique, it is needed when direct automatic differentiation through the function $y = f(x)$ is unavailable or expensive to compute. Instances often arise when one wishes to calculate gradients through numerical solvers or complicated iterative algorithms. Based on the implicit function theorem [46], if there exists such constrained function $h(y, x) = 0$ where y is taken as the converged output of function f, the gradient $\frac{dy}{dx}$ is obtained as

$$\frac{dy}{dx} = -\left[\frac{\partial h(y,x)}{\partial y}\right]^{-1} \frac{\partial h(y,x)}{\partial x}. \tag{4}$$

We use the implicit gradient to derive the gradient of the surface Green's function [42] that appears in the self-energy Σ calculation (see Appendix A 1). In particular, the converged surface Green's function $g_s(\theta)$ in Appendix A 1 must satisfy the self-consistent equation (A6). Hence $h(g_s,\theta) = [A_{ll} - A_{ll-1}g_s A^{\dagger}_{l-1l}] - g_s^{-1} = 0$, where A_{ll} stands for $[ES_{ll} - H_{ll}]$, and θ denotes the input variables to compute g_s. Thus we can write down the gradient of g_s with respect to the input of the recursive algorithm θ (usually including the Hamiltonian matrix, the overlap matrix, and the energy) explicitly by

$$\frac{dg_s}{d\theta} = -\left[\frac{\partial h(g_s,\theta)}{\partial g_s}\right]^{-1} \frac{\partial h(g_s,\theta)}{\partial \theta}. \tag{5}$$

Another way that the implicit gradient is applied is to compute gradients through the self-consistent Poisson's equation under external electrostatic boundary conditions. Note that Poisson's equation (3) depends on charge density ρ, while the charge density is given by the density matrix via NEGF in Eq. (A10), also shown in the self-consistent loop of Fig. 1. To perform back-propagation through Poisson's equation solver, the adjoint sensitivity method [45,47] for PDE-constrained optimization is adopted, which is a technique for constrained optimization in inverse problems. Here, the forward process of the numerical PDE solver is unaltered which is often denoted as the state equation that links the controlled parameter and the state of the constrained system. Meanwhile, an adjoint state equation that connects the perturbation of variables and states is solved by using the same numerical solver. The gradients can then be evaluated with the adjoint state, and join in the gradient chain of backward propagation. Since the adjoint state equation is often independent of the number of controlled variables, the total complexity is proportional to the forward process which makes it suitable for control problems with scalar output and high-dimensional inputs. Recently, the adjoint method has been applied in constructing subtle neural networks containing dynamic physical processes including neural ODE [22] and the deep-equilibrium model [20], which can be considered as examples of cooperations of automatic differentiation and adjoint methods. For our problem here, since in TB models the electrostatic potential is established by point charges, $\Delta\rho(r) = \sum_i \Delta q_i \delta(r - r_i)$, we developed a method to evaluate the gradients of such situations. By linearity of Poisson's equation, the original form is decomposed into a Laplace's equation with the Dirichlet boundary condition and a Poisson's equation with zero Dirichlet boundary condition,

$$\begin{aligned} -\nabla^2[\Delta V_1(r)] &= 0, \\ \Delta V_1(r)|_{\{z_L,z_R\}} &= \{V_L, V_R\}, \end{aligned} \tag{6}$$

$$\begin{aligned} -\nabla^2[\Delta V_2(r)] &= \frac{1}{\epsilon}\Delta\rho(r), \\ \Delta V_2(r)|_{\Sigma} &= 0. \end{aligned} \tag{7}$$

Laplace's equation can be easily solved. The second equation can be solved with image charges [43,48], and the second potential can be written as

$$\begin{aligned} V_2(r_i) = &\sum_{j\in N, j\neq i} \frac{q_j}{4\pi\epsilon} \frac{1}{\sqrt{t_{ij}^2 + (z_i - z_j)^2}} \\ &+ \sum_{j\in N} \frac{q_j}{4\pi\epsilon} \sum_{n=1}^{\infty} \left[\frac{1}{\sqrt{t_{ij}^2 + \Delta_1^2}} - \frac{1}{\sqrt{t_{ij}^2 + \Delta_2^2}} \right. \\ &\left. + \frac{1}{\sqrt{t_{ij}^2 + \Delta_3^2}} - \frac{1}{\sqrt{t_{ij}^2 + \Delta_4^2}} \right], \end{aligned} \tag{8}$$

where $t_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2$, and Δ_i stands for the distance between charges in the central region and all the induced images in the electrodes, which equals

$$\begin{aligned} \Delta_1 &= z_i + (2n+1)z_j + 2(n+1)(d - z_j), \\ \Delta_2 &= z_i + (2n+1)z_j + 2n(d - z_j), \\ \Delta_3 &= (d - z_i) + (2n+1)(d - z_j) + 2(n+1)z_j, \\ \Delta_4 &= (d - z_) + (2n+1)(d - z_j) + 2nz_j, \end{aligned}$$

where the integer n gives the repetitive image charges to be summed over. Practically, the summation is computed until achieving high accuracy, which usually requires hundreds of sites. To speed up this calculation, we apply the FMM [49] to reduce the computational complexity from $O(N^3)$ to $O(N^{4/3})$, where N is the number of sites. To perform backward propagation through the fast multipole layer, the gradient of the output potential to the charges is required. By taking the derivative of a target objective $L : C^d \rightarrow R$, the derivative of L with respect to charge q_j can be expanded as the image summation form of accumulated gradients from the last layer, which is

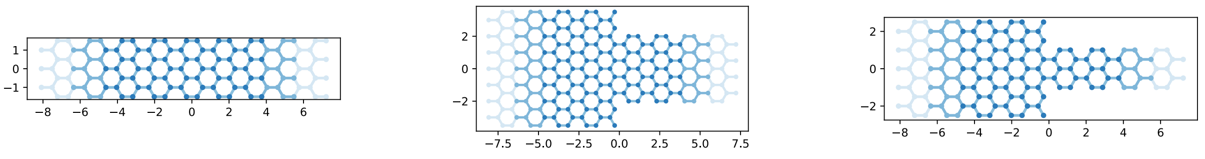
$$\begin{aligned} \frac{\partial L(V)}{\partial q_j} &= \sum_i \frac{\partial L}{\partial V_i} \frac{\partial V_i}{\partial q_j} \\ &= \sum_{i\in N, i\neq j} \frac{\partial L/\partial V_i}{4\pi\epsilon} \frac{1}{\sqrt{t_{ij}^2 + (z_j - z_i)^2}} \\ &\quad + \sum_{i\in N} \frac{\partial L/\partial V_i}{4\pi\epsilon} \sum_{n=1}^{\infty} \left[\frac{1}{\sqrt{t_{ij}^2 + \Delta_1^2}} - \frac{1}{\sqrt{t_{ij}^2 + \Delta_2^2}} \right. \\ &\quad \left. + \frac{1}{\sqrt{t_{ij}^2 + \Delta_3^2}} - \frac{1}{\sqrt{t_{ij}^2 + \Delta_4^2}} \right]. \end{aligned} \tag{9}$$

Similarly, computing gradients of this form can be accelerated by FMM with a complexity of $O(N^{4/3})$, which is significantly faster than iteratively solving the adjoint Poisson's equation.

In summary, AD-NEGF is realized by the following steps:

(1) The entire calculation is implemented in PyTorch so that the explicit numerical expressions are automatically differentiated by PyTorch.

(2) For the implicit equations such as the self-consistent iterations for the surface Green's function and the



(a) Structure of an AGNR with width 7 and length 5.

(b) Structure of a 7-4 graphene nano-junction.

(c) Structure of a 5-2 graphene nano-junction.

FIG. 2. Device structures used in the experiments.

nonequilibrium charge densities, as well as Poisson's equation, we implement the corresponding numerical solvers in the PyTorch autograd forward functions, and implement the gradient computation methods in the corresponding PyTorch autograd backward functions [i.e., implicit gradient of Eq. (4) for iterative solvers and charge gradient of Eq. (9) for Poisson's equation solver]. Therefore, in such cases, the gradients through the numerical solvers are computed by our customized algorithms instead of automatic differentiation.

(3) By implementing the above steps, the gradient of the entire NEGF-TB process can be computed end-to-end simply by back-propagation.

III. EXAMPLES OF AD-NEGF

We have applied AD-NEGF to two-probe transport junctions made of armchair graphene nanoribbon (AGNR), shown in Fig. 2. In all examples, the Hamiltonians are represented by a tight-binding model, with the *pz* orbital for each carbon atom. The parameters we use include hopping integrals of $V_{pp\pi}$, which are -2.97, -0.073, and -0.33 (in units of eV) for the first three nearest neighbors, and the corresponding overlaps $O_{pp\pi}$ as 0.073, 0.018, and 0.026. More details of the calculation parameters can be found in Appendix D.

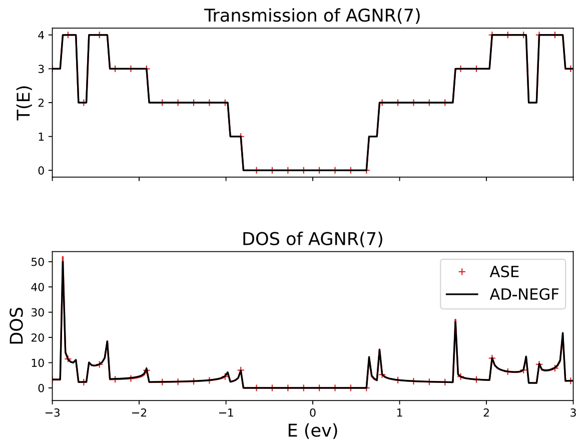
A. Differential transmission

Differential transmission is needed when calculating physical quantities such as the Seebeck coefficient and differential conductance. The Seebeck coefficient, also known as thermoelectric power, measures the induced voltage across a transport junction in response to a temperature gradient. Theoretically, the Seebeck coefficient is related to the derivative of the transmission function $T(E)$ versus the energy E, evaluated at the chemical potential of the system [50]:

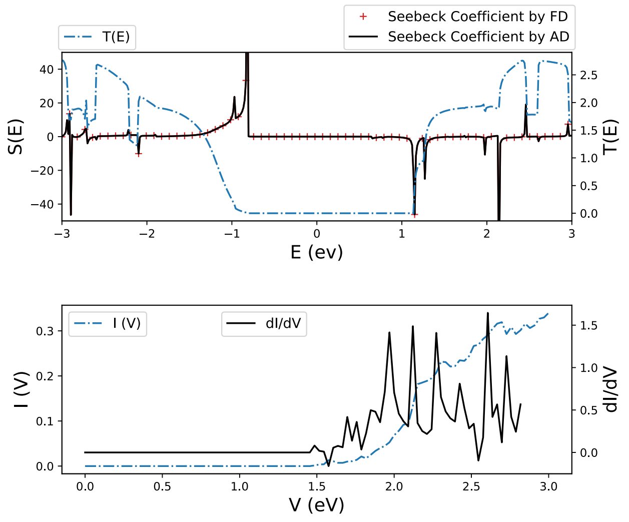
$$S = -\frac{\pi^2 k_B^2 \Upsilon}{3e}\frac{\partial \ln[T(E)]}{\partial E}, \tag{10}$$

where Υ is the temperature and k_B the Boltzmann constant. The differential conductance is another very useful quantity that is related to differential transmission. It is commonly used to analyze nonlinear current-voltage characteristics in tunneling spectroscopy, and devices with negative differential conductance are used in electronic oscillators and amplifiers. Theoretically, the differential conductance is obtained by the gradient of electric current to applied bias voltage: $G = \frac{dI}{dV}$.

For the AGNR system of width 7 and length 5, as shown in Fig. 2(a), the transmission function $T(E)$ and the density of states (DOS) are calculated by our AD-NEGF, shown in Fig. 3(a). The results are in perfect agreement with those obtained by the Atomic Simulation Environment (ASE) [51]. AD-NEGF is then deployed to obtain the Seebeck coefficient by Eq. (10) and the differential conductance; results are shown in Fig. 3(b). The steplike transmission function $T(E)$ leads to singular behavior in its derivative, giving rise to peaks in the Seebeck coefficient curve. While a direct brute-force calculation of the differentiation can be done [FD; red pluses



(a) Transmission and DOS calculated by AD-NEGF and confirmed with ASE.



(b) Seebeck coefficient and differential conductance calculated by AD-NEGF.

FIG. 3. Transmission quantity computation with AD-NEGF.

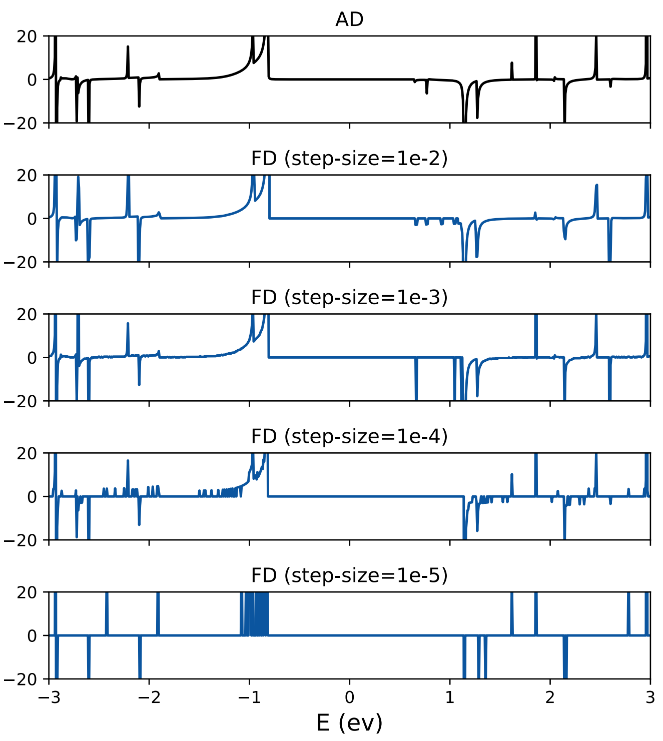


FIG. 4. Comparison of automatic differentiation and numerical differentiation to compute the Seebeck coefficient with different step sizes.

in Fig. 3(b)], such calculation is highly sensitive to the fine energy mesh. In comparison, AD-NEGF gives precise values of the differentiation at any energy point (black curve). In particular, for direct numerical differentiation, the trade-off between the truncation error and the round-off error is observed by selecting different energy mesh sizes from 10^{-2} to 10^{-5} eV. With a coarse mesh, peaks in the Seebeck coefficient may be missing or mistakenly generated due to the truncation error (see Fig. 4). With a very fine mesh, lacking machine precision causes significant noise which may lead to meaningless results. In addition to accuracy, evaluating the Seebeck coefficient with AD-NEGF is also faster than numerical differentiation by roughly 30%, due to our particular back-propagation procedure in AD-NEGF. We compare the end-to-end wall time of computing derivatives for one-dimensional variables (here the Seebeck coefficient) in Table I and high-dimensional variables (here the Hamiltonian) in Table II. In Table I, it is shown clearly that the AD and FD

TABLE I. Wall time of computing Seebeck coefficients.

Number of atoms	Time (FD)	Time (AD)
18	0.17	0.12
36	1.17	0.54
54	3.95	1.78
72	16.17	6.86
90	34.70	14.03
108	68.89	45.28
126	127.52	60.11

TABLE II. Wall time of computing high-dimensional derivatives.

Number of parameters	Time (FD)	Time (AD)
324	5.26	0.02
1296	20.14	0.02
2916	34.71	0.04
5184	55.35	0.04
8100	106.48	0.05

implementation share a similar trend, while AD is still faster by roughly 30%. On the other hand, in the high-dimensional derivative computation task, the scale of the FD method is linear with the number of the variable dimension, while the time cost of AD still follows a constant order, as shown in Table II.

To summarize, the correctness and effectiveness of AD-NEGF are validated in comparison with direct brute-force numerical differentiation. In AD-NEGF, differential transport quantities are calculated by simply calling a single backward step. Moreover, the process of computing derivatives is itself differentiable, permitting the computation of higher-order derivatives such as the nonlinear conductance coefficients [52].

B. Transmission by design

In this section, we show that AD-NEGF can be potentially useful to give insight to the problem of transport-by-design. Namely, if one wishes to obtain a desired transport property, can one design a Hamiltonian that does produce it? Such an inverse problem is very difficult, if not impossible, to solve; here we show AD-NEGF may lead to a possible route. In general, the inverse problem is about inferring input parameters reversely from the objective. One approach is using black-box optimization methods to sample a large number of input combinations, but the computational cost grows exponentially with the number of parameters, making it intractable for this task. On the other hand, based on AD-NEGF, the gradient of the transport property with respect to the Hamiltonian elements can be computed by simply calling the forward and backward computation each for one time, the computational complexity of which is irrelevant of the number of parameters to be determined in the Hamiltonian. Such characteristics of AD-NEGF allow for conducting gradient-based optimization on the Hamiltonian elements to fit the desired properties. Through such paradigms, AD-NEGF holds the potential to solve transport-by-design problems in material science.

Let us consider a 7-4 graphene nanojunction, consisting of 7 graphene rings on the left and 4 rings on the right. The transmission coefficient $T_{74}(E)$ of this system is calculated by NEGF-TB, shown as the blue dash-dotted curve in the lower panel of Fig. 5, which serves as the desired result. In this toy exercise of the inverse problem, we wish to find a Hamiltonian that will produce this $T_{74}(E)$. To this end, we may start from any physically sound Hamiltonian as an initial guess, for example, the Hamiltonian of a 5-2 graphene nanojunction, H_{52}, which produces a totally different transmission $T_{52}(E)$ as depicted by the green dashed curve in Fig. 5. With the $T_{74}(E)$

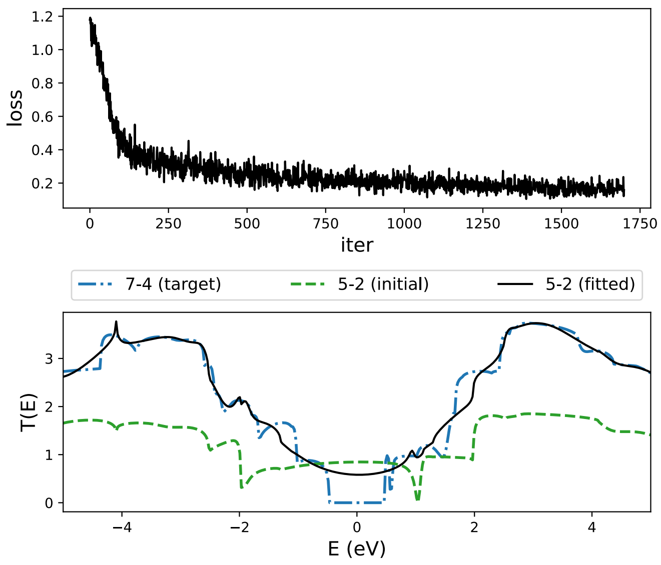


FIG. 5. The fitting loss and the fitted transmission curve of a 5-2 graphene nanojunction.

as the goal and using the gradient-based optimization in AD-NEGF, it is possible to automatically vary the parameters in H_{52} such that it generates the desired result T_{74}. The fitting parameters are the elements of H_{52} including the device, leads, and the corresponding couplings. For this exercise, the dimension of the optimizing variables is at the level of 10^4. The transmission curve, as shown in Fig. 5, consists of 2000 energy points sampled from (−5 eV, 5 eV). The loss function is defined as the l^2 norm of the target and fitted transmission at given energy points, which is loss $= ||T_{74}(E) - \hat{T}_{52}(E)||$. Since directly computing the gradients of all 2000 points is inefficient, we apply the stochastic gradient descent algorithm to conduct minibatch optimization which has shown supremacy of efficiency and performance in high-dimensional optimization problems. The fitting parameters are optimized with the Adam optimizer [53] built in PyTorch, making the procedure highly similar to training a neural network.

The results are displayed in Fig. 5. The upper panel shows the loss function versus the optimization iteration, where the loss is reduced to a considerably low level after a few hundred iterations, which means the converged H_{52} parameters of the 5-2 nanojunction could approximately produce $T_{74}(E)$ of the 7-4 nanojunction. Indeed, the black solid curve in the lower panel, obtained by the converged H_{52}, is akin to a smoothed T_{74} of the 7-4 junction. This is consistent with the intuition since a graphene junction of 5-2 has fewer degrees of freedom than that of a 7-4 nano-junction. We mention in passing that we have also tried traditional black-box optimization methods including Bayesian optimization, genetic algorithms, and gradient-based optimization with numerical differentiation, but none works for this problem because of the curse of dimensionality.

Finally, we wish to mention that solving the problem of transport-by-design can be potentially very useful in applications where a particular transport property is desired. As we have shown here, AD-NEGF can inversely determine a Hamiltonian that would approximately generate the desired property. Since the Hamiltonian matrix elements are made of atomic potentials, it would provide tremendous intuitions on the material and external manipulation (i.e., stress, doping, impurity, external fields, etc.) to produce the desired transport.

C. On-site doping

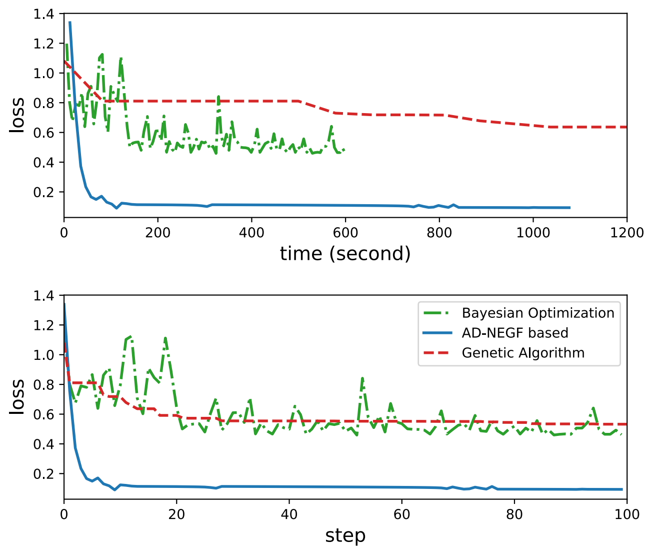
Modern device engineering is capable of manipulating material properties at the atomic level. By stress and impurity doping, etc., electronic structure and material parameters can be controlled and modified for better device performance. Here we further explore the possibility to solve practical inverse problems with AD-NEGF through an end-to-end doping optimization. The doped structure is illustrated in Fig. 6, where some of the carbon atoms (in gray) are replaced with the impurities (in Navajo white). To this end, as the desired goal we wish to reduce the average transmission of the AGNR in a specified energy range of (−1 eV, 1 eV), by doping impurity atoms into the scattering region. Doping can be modeled as an effective change in the site and the hopping parameters in the TB Hamiltonian, i.e., the diagonal and off-diagonal elements of the Hamiltonian matrix. In contrast to the inverse problem presented in the last section, here we are only allowed to vary the SKTB parameters associated with the dopant atoms, leaving parameters of the host material not touched. The parameters for optimization include orbital energy and two-center integrals. We also use the *pz* orbital as the TB basis for the impurity atoms. With three nearest-neighbor hopping and overlap integrals, the total number of optimization variables is 13 for one kind of impurity atom. The loss function of the optimization is defined as loss $= ||T(E)||$ for $E \in (-1\text{ eV}, 1\text{ eV})$.

For comparison, we also apply conventional optimization methods of genetic algorithm and Bayesian optimization. The results are displayed in Fig. 7. In the loss diagram in the left panels, the gradient-based method in AD-NEGF converges significantly faster and much better than the conventional approaches in terms of computational time as well as the total iteration steps. We also found that the conventional optimizations are sensitive to the preset hyperparameters, and were not able to reach the loss level of AD-NEGF (see left panels of Fig. 7). Corresponding to the loss curves, the AD-NEGF has essentially reached our design goal of reducing transmission in the energy range of (−1 eV, 1 eV), shown in the right panels of Fig. 7 (blue curve). In comparison, the conventional optimization was not able to reach the design goal (green and red curves).

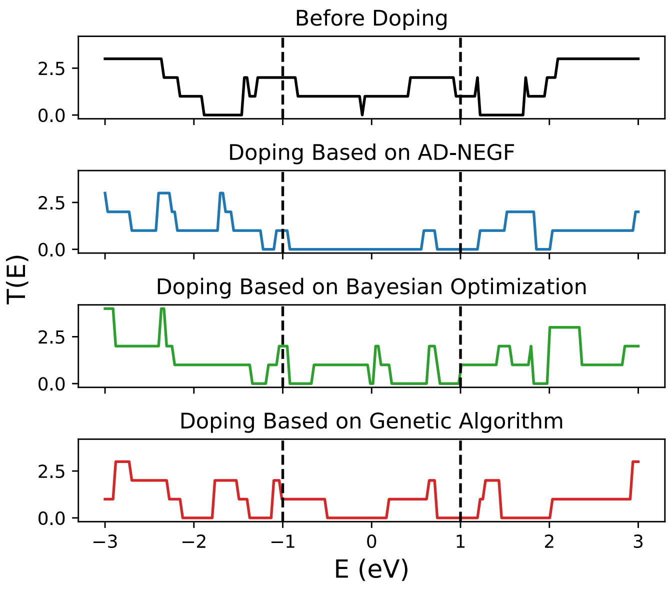
These results validate the effectiveness of the AD-NEGF method in conducting practical atomic-level inverse design to optimize transport properties by cooperating with material models.

IV. SUMMARY

Motivated by recent advances in AI for quantum transport and differentiable programming, we have developed an automatic differentiation capability into the atomistic NEGF quantum transport simulator. The end-to-end automatic differentiable NEGF simulator, AD-NEGF, calculates gradient information by AD while guaranteeing the correctness



(a) Loss against running time and iteration steps respectively.



(b) Original and optimized transmission curves.

FIG. 6. Comparison between AD-NEGF and conventional black-box optimization methods in the doping optimization task.

of forward simulation. The gradient information enables accurate predictions of transport properties that depend on the derivatives of the transmission coefficient and/or charge current, such as the Seebeck coefficients in the thermoelectric phenomenon and differential conductance in nonlinear carrier transport. For ballistic transport in confined nanostructures, transmission functions often vary rapidly as a function of carrier energy due to quantum interference, causing its derivative to be singular and thus hard to accurately determine. AD-NEGF solves such problems very accurately as we demonstrated in this work. In the experimental technique of inelastic tunneling spectroscopy (IETS) [54], the opening of phonon-assisted transmission channels leads to slight changes of the measured current at a certain bias voltage, and the signal is picked up by measuring the differential conductance. In NEGF simulations of IETS, AD-NEGF can be used to directly compare with the measured signal. AD-NEGF can also be very useful for accurate and efficient simulations of other transport coefficients related to nonlinear expansions of current and/or charge versus external voltages.

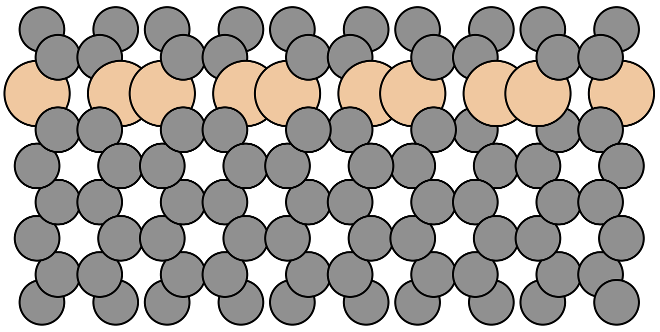


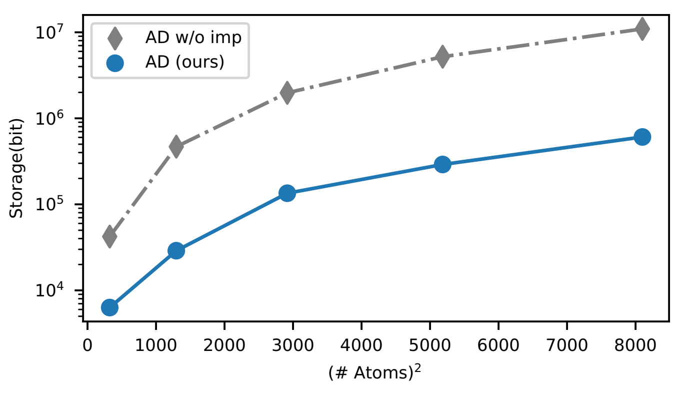
FIG. 7. The structure of the doped AGNR system, where the atoms depicted in Navajo white are replaced with impurities.

More interestingly, AD-NEGF can be applied to the inverse design problem; namely, with a desired transport property, AD-NEGF inversely finds a possible device Hamiltonian that would produce such a property. While property-by-design is a dream goal of materials and device physics, it is an extremely difficult problem to solve. In particular, due to the high dimensionality of Hamiltonian matrices in NEGF simulations, conventional optimization techniques are essentially powerless for such inverse problems. To this end, we showed that AD-NEGF with gradient-based optimization has great potential. Here we showed that starting from a predefined transport property (transmission function), AD-NEGF inversely determines a possible Hamiltonian that would approximately produce it. Though the examples are relatively simple, the idea is very clear. In real practical applications, once the model Hamiltonian is inversely determined, one may investigate its on-site and hopping parameters which would generate deep insights into how to realize such a model with real materials.

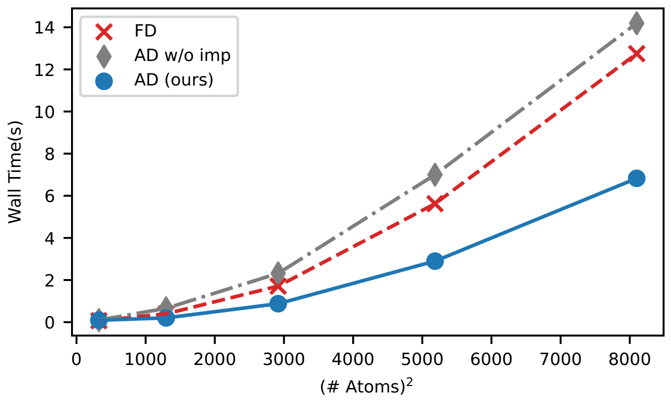
Our code is available on GitHub [55] for cross-checking. The code will be maintained as an open-source repository in the future.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Hong Guo for useful discussions on quantum transport theory and for his critical reading of the manuscript before submission. This work is supported in part by the Natural Science Foundation of China (Grant No. 62276188). L.W. is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDB30000000 and the National Natural Science Foundation of China under Grant No. T2121001.



(a) Memory consumption



(b) Wall time

FIG. 8. Comparison of memory and time costs for the surface Green's function.

APPENDIX A: ADDITIONAL DETAILS ON THE NEGF METHOD

1. Self-energy

The self-energy of electrodes is computed from the surface Green's function g_s of the electrode layer coupled with devices. Here we assume that the system is made up of a device and two semi-infinite contacts on the side. Equation (1) can be expanded in the following form:

$$\begin{bmatrix} A_L & A_{LD} & 0 \\ A_{DL} & A_D & A_{DR} \\ 0 & A_{RD} & A_R \end{bmatrix}\begin{bmatrix} G_L & G_{LD} & G_{LR} \\ G_{DL} & G_D & G_{DR} \\ G_{RL} & G_{RD} & G_R \end{bmatrix} = I, \qquad \text{(A1)}$$

where $A = [EI - H]$, and the subscripts are used to distinguish the matrix elements corresponding to the left lead (L), the device (D), the right lead (R), and their interactions. Thanks to its block-tridiagonal form, the device Green's function G_D satisfies

$$\big[A_D - A_{DL}A_L^{-1}A_{LD} - A_{DR}A_R^{-1}A_{RD}\big]G_D = I. \qquad \text{(A2)}$$

Since $A_D = [EI - H_D]$, compared with Eq. (2), we have

$$\Sigma^L = A_{DL}A_L^{-1}A_{LD}, \qquad \text{(A3)}$$

$$\Sigma^R = A_{DR}A_R^{-1}A_{RD}, \qquad \text{(A4)}$$

$$\Sigma = \Sigma^L + \Sigma^R. \qquad \text{(A5)}$$

To avoid using the full leads Hamiltonian, it is assumed that only the neighboring layers have interactions with each other. We denote the left lead layer connected to the device by l. Then the left self-energy can be simplified as $\Sigma^L = A_{Dl}A_l^{-1}A_{lD}$. The coupling matrix A_{lD} is given as input of NEGF. What remains unclear is A_l^{-1}, the bottom-right block of A_L^{-1}. This is known as the surface Green's function, denoted as g_s. By utilizing the ideal lead assumption that removing one layer of the lead will not change g_s, we obtain a self-consistent form as

$$g_s^{-1} = [A_l - A_{l,l-1}g_sA_{l-1,l}^{\dagger}], \qquad \text{(A6)}$$

where $A_{l,l-1}$ is the block in $[EI - H]$ for the coupling between layer l and layer $l-1$. We implemented the Lopez-Sancho algorithm [42], as illustrated in Algorithm, to accelerate the convergence speed. We have also implemented a modern method based on the generalized eigenvalue problem [56] as an alternative.

2. Computation of the self-consistent electrostatic potential

Denote the charge densities in the equilibrium and nonequilibrium states as ρ_0 and ρ, and the potential fields from the original neutral and redistributed charges as V_0 and V. The equilibrium and non-equilibrium Hamiltonian can be expressed as $H_0 = T + V_0$, $H_{\text{neq}} = T + V$, where T is the kinetic energy. Poisson's equation relates potentials to the corresponding charge densities:

$$\begin{aligned} \nabla\cdot\epsilon(r)\nabla V(r) &= -\rho(r), \\ \nabla\cdot\epsilon(r)\nabla V_0(r) &= -\rho_0(r). \end{aligned} \qquad \text{(A7)}$$

Therefore we have $\nabla\cdot\epsilon(r)\nabla[\Delta V(r)] = -[\rho(r) - \rho_0(r)]$, where $\Delta V = V - V_0$ is used to correct the Hamiltonian by $H_{\text{neq}} = H_0 + \Delta V$. The updated H_{neq} will again be used to update ΔV. Hence a self-consistent iteration is constructed:

$$\begin{aligned} \nabla\cdot\epsilon(r)\nabla[\Delta V(r)] &= -[\rho(r;\Delta V) - \rho_0(r)], \\ \Delta V(r)|_{\{z_L,z_R\}} &= \{V_L, V_R\}. \end{aligned} \qquad \text{(A8)}$$

Charge densities are necessary inputs for the above equation. Denote potentials in the left and right electrodes as u_l and u_r (assume $u_l < u_r$); then the charge density $\rho(r) = -\frac{i}{2\pi}\int_{-\infty}^{+\infty} dE\, G(E)$, which can be decomposed into equilibrium and nonequilibrium terms:

$$\begin{aligned} \rho(r) &= \rho_{\text{eq}}(r) + \rho_{\text{neq}}(r) & \text{(A9)} \\ &= \frac{1}{\pi}\text{Im}\bigg[\int_{-\infty}^{u_l} dE\, G_D(E)\bigg] + \frac{1}{2\pi}\int_{u_l}^{u_r} dE\, G_D^{<}(E). & \text{(A10)} \end{aligned}$$

The first integration up to infinity can be computed efficiently using contour integration with the residue theorem. It is achieved by expanding the Fermi-Dirac function [57,58]. On the other hand, the nonequilibrium charge density ρ_{neq} is computed directly by numerical integration. The density of neutral charges ρ_0 can be computed by setting $u_l = u_r = 0$.

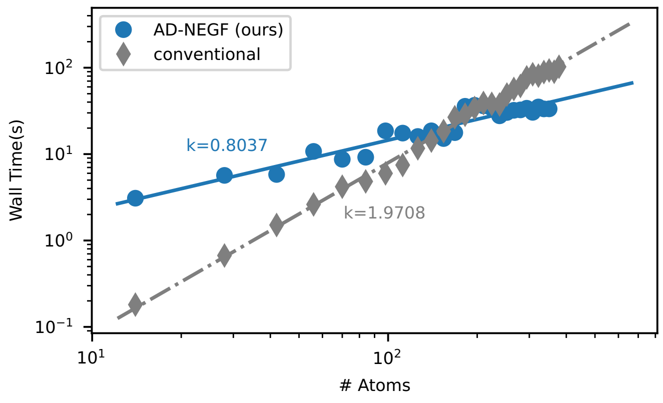


FIG. 9. Gradient computation cost of Poisson's equation.

3. Expressions of transport properties

With the NEGF theory, electronic transport properties can be derived, such as the transmission probability [$T(E)$], the density of states (DOS), the electronic current (I), and the equilibrium and nonequilibrium electronic densities (ρ_{eq} and ρ_{neq}). Here we list some of the expressions:

$$T(E) = \mathrm{Trace}[\Gamma_L(E)G_D(E)\Gamma_R(E)G_D^\dagger(E)], \quad \text{(A11)}$$

$$\mathrm{DOS}(E) = -\frac{1}{\pi}\mathrm{Trace}\{\mathrm{Im}[G_D(E)]\}, \quad \text{(A12)}$$

$$I = \frac{2e}{\hbar}\int_{-\infty}^{+\infty}\frac{dE}{2\pi}T(E)[f(E-u_l)-f(E-u_r)], \quad \text{(A13)}$$

$$\rho(r) = \frac{1}{\pi}\mathrm{Im}\left[\int_{-\infty}^{u_l} dE\, G_D(E)\right] + \frac{1}{2\pi}\int_{u_l}^{u_r} dE\, G_D^<(E). \quad \text{(A14)}$$

For Eq. (A13), the integral range of the current is decided by the subtraction of the Fermi-Dirac function, which is a little wider than (u_l, u_r).

APPENDIX B: COMPUTATIONAL COST ANALYSIS OF CUSTOMIZED GRADIENT COMPUTATION

In this Appendix, we analyze the space and time cost of the customized gradient computation method adopted in

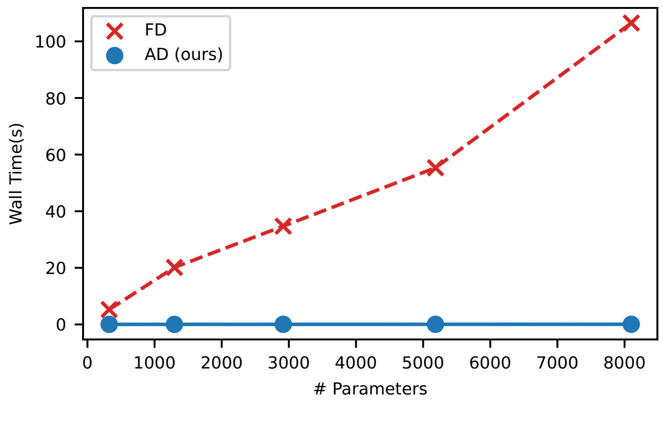


FIG. 10. Wall time of Hamiltonian gradient computation against the dimension of the input variable.

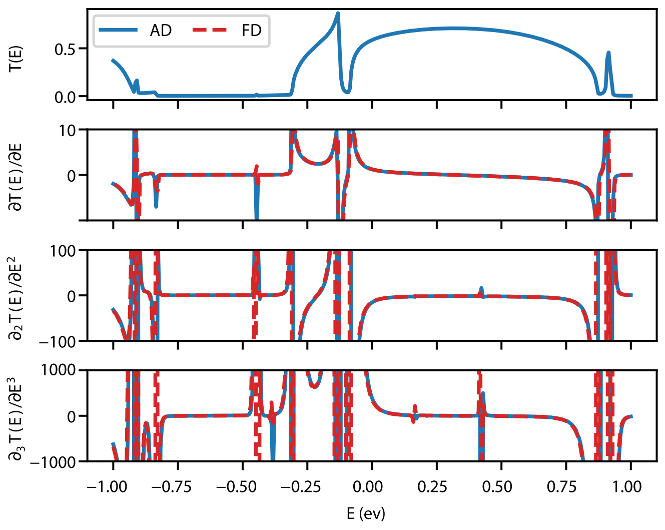


FIG. 11. Computation of high-order derivatives via AD-NEGF and FD.

AD-NEGF, including the implicit gradient method for computing gradients through the self-consistent iteration, and the image charge method used in solving Poisson's equation. For the implicit gradient method, we compared the space and time required to perform back-propagation through the surface Green's function algorithm with and without adopting the implicit gradient method. We also illustrate the wall time of computing the gradient of Poisson's equation. We consider a nanoribbon as the test system, where we fix the width as 7 atoms, and vary the length of the system size (number of atoms).

1. Implicit gradient method for the surface Green's function

The computational analysis of the implicit gradient method is provided in Fig. 8. The computational time and storage here are collected when computing the Seebeck coefficient, which is a 1D derivative as explained in the main text. From Fig. 8(a) we can clearly observe that the direct back-propagation costs 10 times the memory of our implementation. This aligns with our proposal that tracking and expanding the computation graph of the recursive algorithm is very memory-demanding. In Fig. 8(b), we also demonstrate the effectiveness of the implicit gradient method in accelerating the gradient computation, which is faster than the result of FD and AD without using the implicit gradient method.

2. Gradient of Poisson's equation solution

In Fig. 9, we compare the computational time cost for Poisson's equation. We can see that when compared with the conventional point charge method, our implementation is much faster. The computational complexity of our algorithm is sublinear, while for the conventional method it is roughly quadratic. This matches with the theoretical analysis that the FMM method reduces the computational complexity from $O(N^3)$ to $O(N^{4/3})$, where the extra speedup is due to the built-in parallelization from PyTorch tensor operations.

TABLE III. The hyperparameters of the genetic algorithm.

Parameter	Value
max_num_iteration	None
population_size	20
mutation_probability	0.1
elit_ratio	0.01
crossover_probability	0.5
parents_portion	0.3
crossover_type	uniform
max_iteration_without_improv	None

3. Scaling of end-to-end Hamiltonian gradient computation

We analyze the time cost of computing derivatives using AD-NEGF and the conventional FD method, as displayed in Fig. 10. More specifically speaking, we compute the derivative of the transmission with respect to the input Hamiltonian elementwise. The figure clearly shows the advantage of AD to compute derivatives with high-dimensional input variables. On the other hand, the computation time of the FD method, as expected, scales linearly with the input dimension, which costs far more than the AD method.

APPENDIX C: DEMONSTRATION OF COMPUTING HIGH-ORDER DERIVATIVES

AD-NEGF can naturally generalize to computing high-order derivatives by directly applying the AD on the custom first-order derivative. In Fig. 11, we show an example of computing derivatives of the transmission coefficient with respect to energy, with first, second, and third order. The result given by AD-NEGF is validated by comparing with the numerical finite difference method.

TABLE IV. The hyperparameters of the Bayesian optimization algorithm. ucb: Upper confidence bound.

Parameter	Value
random_state	3
verbose	2
kind	ucb
kappa	2.5
xi	0.0

APPENDIX D: ADDITIONAL DETAILS ON COMPUTATIONAL SETUP

The simulations are run on an Intel Xeon CPU E5-2650 version 4 at 2.20 GHz CPU, and an NVIDIA Tesla P40 GPU. We implemented our method in PyTorch 1.9.1. We validated the correctness of our simulation results by comparing with ASE of version 3.22.0.

In the simulations, we set the learning rate of the Adam optimizer as 0.001, and the batch size as 64. Bayesian optimization is implemented based on Ref. [59], and the genetic algorithm is implemented based on Ref. [60]. The bounds of the optimization variables for the black-box optimizers are $(\theta_0 - 0.3, \theta_0 + 0.3)$, where θ_0 is the initial value, namely the original 5-2 nanojunction TB Hamiltonian for the transmission curve fitting experiment, and undoped SKTB parameters for the device doping optimization experiment. The hyperparameters of the genetic algorithm are listed in Table III, and the hyperparameters of the Bayesian optimization algorithm are listed in Table IV.

[1] Y. V. Nazarov and Y. M. Blanter, *Quantum Transport: Introduction to Nanoscience* (Cambridge University Press, Cambridge, UK, 2009).

[2] D. Ryndyk, *Theory of Quantum Transport at Nanoscale: An Introduction* (Springer, Berlin, 2015).

[3] M. Wimmer, Quantum transport in nanostructures: From computational concepts to spintronics in graphene and magnetic tunnel junctions, Ph.D. thesis, Universität Regensburg, 2009.

[4] C. Jacoboni, *Theory of Electron Transport in Semiconductors: A Pathway from Elementary Physics to Nonequilibrium Green Functions* (Springer Science & Business Media, New York, 2010).

[5] J. Taylor, H. Guo, and J. Wang, *Ab initio* modeling of quantum transport properties of molecular electronic devices, Phys. Rev. B **63**, 245407 (2001).

[6] C. Medina-Bailon, T. Dutta, A. Rezaei, D. Nagy, F. Adamu-Lema, V. P. Georgiev, and A. Asenov, Simulation and modeling of novel electronic device architectures with NESS (Nano-Electronic Simulation Software): A modular nano TCAD simulation framework, Micromachines **12**, 680 (2021).

[7] L. Silvestri, M. Palsgaard, R. Rhyner, M. Frey, J. Wellendorff, S. Smidstrup, R. Gull, and K. El Sayed, Hierarchical modeling for TCAD simulation of short-channel 2D material-based FETs, Solid-State Electron. **200**, 108533 (2023).

[8] S. Smidstrup, T. Markussen, P. Vancraeyveld, J. Wellendorff, J. Schneider, T. Gunst, B. Verstichel, D. Stradi, P. A. Khomyakov, U. G. Vej-Hansen *et al.*, QuantumATK: An integrated platform of electronic and atomic-scale modelling tools, J. Phys.: Condens. Matter **32**, 015901 (2020).

[9] R. Kim and M. S. Lundstrom, Computational study of the Seebeck coefficient of one-dimensional composite nano-structures, J. Appl. Phys. **110**, 034511 (2011).

[10] L. Britnell, R. V. Gorbachev, A. K. Geim, L. A. Ponomarenko, A. Mishchenko, M. T. Greenaway, T. M. Fromhold, K. S. Novoselov, and L. Eaves, Resonant tunnelling and negative differential conductance in graphene transistors, Nat. Commun. **4**, 1794 (2013).

[11] R. J. Prentki, Theory and simulation of novel low-power nanotransistors, Ph.D. thesis, McGill University, 2021.

[12] M. Bürkle, U. Perera, F. Gimbert, H. Nakamura, M. Kawata, and Y. Asai, Deep-learning approach to first-principles transport simulations, Phys. Rev. Lett. **126**, 177701 (2021).

[13] K. Li, J. Lu, and F. Zhai, Neural networks for modeling electron transport properties of mesoscopic systems, Phys. Rev. B **102**, 064205 (2020).

[14] A. K. Pimachev and S. Neogi, First-principles prediction of electronic transport in fabricated semiconductor heterostructures via physics-aware machine learning, npj Comput. Mater. **7**, 93 (2021).

[15] A. Lopez-Bezanilla and O. A. von Lilienfeld, Modeling electronic quantum transport with machine learning, Phys. Rev. B **89**, 235411 (2014).

[16] T. Županĉić, I. Stresec, and M. Poljak, Predicting the transport properties of silicene nanoribbons using a neural network, in *Proceedings of the 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)* (IEEE, Piscataway, NJ, 2020), pp. 44–48.

[17] S.-C. Han, J. Choi, and S.-M. Hong, Acceleration of three-dimensional device simulation with the 3D convolutional neural network, in *Proceedings of the 2021 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)* (IEEE, Piscataway, NJ, 2021), pp. 52–55.

[18] S. Souma and M. Ogawa, Acceleration of nonequilibrium Green's function simulation for nanoscale FETs by applying convolutional neural network model, IEICE Electron. Exp. **17**, 20190739 (2020).

[19] S. Souma and M. Ogawa, Neural network model for implementation of electron-phonon scattering in nanoscale device simulations based on NEGF method, in *Proceedings of the 2021 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)* (IEEE, Piscataway, NJ, 2021), pp. 56–59.

[20] S. Bai, J. Z. Kolter, and V. Koltun, Deep equilibrium models, in *Advances in Neural Information Processing Systems*, Vol. 32 (NeurIPS, 2019).

[21] B. Amos and J. Z. Kolter, OptNet: Differentiable optimization as a layer in neural networks, in *International Conference on Machine Learning* (PMLR, 2017), pp. 136–145.

[22] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, Neural ordinary differential equations, in *Advances in Neural Information Processing Systems*, Vol. 31 (NeurIPS, 2018).

[23] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, DiffTaichi: Differentiable programming for physical simulation, arXiv:1910.00935.

[24] M. Innes, A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V. B. Shah, and W. Tebbutt, A differentiable programming system to bridge machine learning and scientific computing, arXiv:1907.07587.

[25] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, End-to-end differentiable physics for learning and control, in *Advances in Neural Information Processing Systems*, Vol. 31 (NeurIPS, 2018), pp. 71–78.

[26] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, Brax: A differentiable physics engine for large scale rigid body simulation, arXiv:2106.13281.

[27] P. Holl, N. Thuerey, and V. Koltun, Learning to control PDEs with differentiable physics, in *International Conference on Learning Representations* (PMLR, 2019).

[28] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, Machine learning: Accelerated computational fluid dynamics, Proc. Natl. Acad. Sci. USA **118**, e2101784118 (2021).

[29] C. Schenck and D. Fox, SPNets: Differentiable fluid dynamics for deep neural networks, in *Conference on Robot Learning* (PMLR, 2018), pp. 317–335.

[30] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, Differentiable Monte Carlo ray tracing through edge sampling, ACM Trans. Graph. **37**, 1 (2018).

[31] J. B. Rigo and A. K. Mitchell, Automatic differentiable numerical renormalization group, Phys. Rev. Res. **4**, 013227 (2022).

[32] M. F. Kasim and S. M. Vinko, Learning the exchange-correlation functional from nature with fully differentiable density functional theory, Phys. Rev. Lett. **127**, 126403 (2021).

[33] L. Li, S. Hoyer, R. Pederson, R. Sun, E. D. Cubuk, P. Riley, and K. Burke, Kohn-Sham equations as regularizer: Building prior knowledge into machine-learned physics, Phys. Rev. Lett. **126**, 036401 (2021).

[34] T. Tamayo-Mendoza, C. Kreisbeck, R. Lindh, and A. Aspuru-Guzik, Automatic differentiation in quantum chemistry with applications to fully variational Hartree-Fock, ACS Cent. Sci. **4**, 559 (2018).

[35] F. Pavošević and S. Hammes-Schiffer, Automatic differentiation for coupled cluster methods, arXiv:2011.11690.

[36] S. S. Schoenholz and E. D. Cubuk, JAX, M.D.: A framework for differentiable physics, in *Advances in Neural Information Processing Systems*, Vol. 33 (NeurIPS, 2020).

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai *et al.*, PyTorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems*, Vol. 32 (NeurIPS, 2019).

[38] M. V. Klymenko, J. A. Vaitkus, J. S. Smith, and J. H. Cole, NanoNET: An extendable Python framework for semi-empirical tight-binding models, Comput. Phys. Commun. **259**, 107676 (2021).

[39] J. Maassen, M. Harb, V. Michaud-Rioux, Y. Zhu, and H. Guo, Quantum transport modeling from first principles, Proc. IEEE **101**, 518 (2012).

[40] J. C. Slater and G. F. Koster, Simplified LCAO method for the periodic potential problem, Phys. Rev. **94**, 1498 (1954).

[41] M. P. Anantram, M. S. Lundstrom, and D. E. Nikonov, Modeling of nanoscale devices, Proc. IEEE **96**, 1511 (2008).

[42] M. P. Lopez Sancho, J. M. Lopez Sancho, J. M. L. Sancho, and J. Rubio, Highly convergent schemes for the calculation of bulk and surface Green functions, J. Phys. F **15**, 851 (1985).

[43] A. Svizhenko and M. P. Anantram, Effect of scattering and contacts on current and electrostatics in carbon nanotubes, Phys. Rev. B **72**, 085430 (2005).

[44] M. Zahn, Point charge between two parallel grounded planes, Am. J. Phys. **44**, 1132 (1976).

[45] L. S. Pontryagin, *Mathematical Theory of Optimal Processes* (CRC Press, Boca Raton, FL, 1987).

[46] S. G. Krantz and H. R. Parks, *The Implicit Function Theorem: History, Theory, and Applications* (Springer Science & Business Media, New York, 2002).

[47] R.-E. Plessix, A review of the adjoint-state method for computing the gradient of a functional with geophysical applications, Geophys. J. Int. **167**, 495 (2006).

[48] M. A. Harb, Scattering effects in atomistic quantum transport simulations, Ph.D. thesis, McGill University, 2019.

[49] N. Engheta, W. D. Murphy, V. Rokhlin, and M. S. Vassiliou, The fast multipole method (FMM) for electromagnetic scattering problems, IEEE Trans. Antennas Propag. **40**, 634 (1992).

[50] P. Reddy, S.-Y. Jang, R. A. Segalman, and Arun Majumdar, Thermoelectricity in molecular junctions, Science **315**, 1568 (2007).

[51] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus *et al.*, The atomic simulation environment: A Python library for working with atoms, J. Phys.: Condens. Matter **29**, 273002 (2017).

[52] Z. S. Ma, J. Wang, and H. Guo, Weakly nonlinear ac response: Theory and application, Phys. Rev. B **59**, 7575 (1999).

[53] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[54] M. A. Reed, Inelastic electron tunneling spectroscopy, Mater. Today **11**, 46 (2008).

[55] See https://github.com/floatingCatty/ADNEGF.

[56] J.-S. Wang, J. Wang, and J. T. Lü, Quantum thermal transport in nanostructures, Eur. Phys. J. B **62**, 381 (2008).

[57] D. A. Areshkin and B. K. Nikolić, Electron density and transport in top-gated graphene nanoribbon devices: First-principles Green function algorithms for systems containing a large number of atoms, Phys. Rev. B **81**, 155450 (2010).

[58] T. Ozaki, Continued fraction representation of the Fermi-Dirac function for large-scale electronic structure calculations, Phys. Rev. B **75**, 035123 (2007).

[59] F. Nogueira, Bayesian optimization: Open source constrained global optimization tool for Python, https://github.com/fmfn/BayesianOptimization.

[60] R. M. Solgi, Genetic algorithm package for Python, https://github.com/rmsolgi/geneticalgorithm.