Table of Contents

Electron transport calculations with electron-phonon coupling included via the special thermal displacement method - STD-Landauer

Version: 2017.1

Downloads & Links

PDF
all-files.zip

Silicon device
Inelastic current in a silicon p-n junction
Introduction to QuantumATK
Transport calculations with QuantumATK
SpecialThermalDisplacement
ATK Reference Manual

Phonons have a large influence on the electronic transport properties of semiconductors, but most methods for modeling them are computationally very expensive. In a recent paper; *First-Principles Electron Transport with Phonon Coupling: Large-Scale at Low Cost* [1], Gunst et al. proposed a new method, which is computationally equivalent to traditional 0 K calculations. We call this method **STD-Landauer**. In this case study, we will go through the calculations required to produce one of the figures in the paper, explaining how QuantumATK was used to create this research.

Specifically, we will re-create figure 2c in the paper [1]. This figure shows the IV characteristics of the short silicon p-n junction, comparing the current with no electron-phonon coupling included, and two different methods for including it. It shows how the electron-phonon coupling effects are very important to include for a correct description of the reverse current.

We will compare the new **STD-Landauer** method to the more well-established Lowest Order Expansion (**LOE**) method, and show that the STD-Landauer can achieve comparable accuracy at much lower computational cost in both time and memory.

Note

QuantumWise case studies are primarily directed at experienced users of QuantumATK. Instructions are deliberately concise in order to focus mostly on the scientific content.

For more basic details on how to use QuantumATK, please refer to our Tutorials, with Introduction to

QuantumATK and Transport calculations with QuantumATK being good places to start.

Building the device

We will here use a simple silicon p-n junction, and the workflow for building it is already explained in detail in a tutorial: Silicon device. If you do not know how to build such a device, you may go through the steps in the tutorial, applying the modifications mentioned below.

Finding the lattice constant of silicon

As is done in the paper, we will model the structural properties, including phonons, with a classical force field. This means that we first need to find the lattice constant for silicon with the *'StillingerWeber_Si_1985* force field. Do a geometry optimization with the following criteria for the forces and stress:

```
max_forces=0.001*eV/Ang,
max_stress=0.01*GPa,
```

This will result in a lattice constant of 5.430952 Å, very close to the database value of 5.4306 Å. Use this optimized bulk configuration as the basis for the following.

Building the pristine device

In this case we go with a device which is actually too short, compared to the screening length at the chosen doping level. However, this choice was made in the paper to allow running of the heavy LOE calculations later on. You should therefore choose 24 layers instead of 52, as specified in the tutorial. You can use the same approach for doping, or alternately you can use the doping widget as described in this tutorial: NiSi2–Si interface. However, note that the doping should be n-type on the left and p-type on the right.

Tip

For more details on the implementation of doping in QuantumATK, please consult this note: Doping methods available in QuantumATK

Applying the special thermal displacement to the atomic positions

Having built the doped pristine device, we now need to apply the thermal displacements to build the device for calculations with the atomic positions at 300 K. This means that we first need to compute the dynamical matrix, which is independent of the temperature and is used to compute the relevant phonon modes. Afterwards, they are weighted using to the procedure of Zacharias and Giustino [2] and the atoms are displaced accordingly. Use the pristine device, again with the *'StillingerWeber_Si_1985* forcefield, and calculate the dynamical matrix. You may follow the procedure in this tutorial: Inelastic current in a silicon p-n junction, except for these changes:

- Insert an optimize geometry block with a force tolerance of *0.0001 eV/Ang*
- For the Dynamical matrix, change repetitions to *custom* and *3x3*.

Run the script in a folder called `dyn-mat` to calculate the dynamical matrix; it should take no more than a few minutes. You may also download it here: dyn-mat.py. Finally, create the device configuration at 300 K using a simple QuantumATK Python script: build_std_conf.py. Note that the QuantumATK method is called `SpecialThermalDisplacement` and you may find the manual page here: SpecialThermalDisplacement.

```
# -*- coding: utf-8 -*-
filename = 'device-pristine.hdf5'
filename_dynmat = 'ff-device-si.hdf5'
filesave = 'device-std-300K.hdf5'
device_configuration = nlread(filename, DeviceConfiguration)[-1]
dynamical_matrix = nlread(filename_dynmat,DynamicalMatrix)[-1]
T = 300

device_configuration_STD = SpecialThermalDisplacement(device_configuration, dynamical_matrix=dynamical_
device_configuration_STD.update()
nlsave(filesave,device_configuration_STD)
```

Run it, and you will have both the pristine and thermally displaced configurations ready for the actual calculations. You may also download the pristine configuration as a script: device_pristine.py or an .hdf5 file: device_pristine.hdf5. You may also download the thermally displaced configuration here as a script: device_std.py or an .hdf5 file: device_std.hdf5.

Tip

As you can see here, applying the STD method is very simple, and for the actual calculations, we will not not do it in a separate script.

Calculations

There are several ways in which one may approach these calculations, but here we first do the calculations for zero temperature and bias, and then we use those results as basis for the phonon-including calculations at 300 K, using the special thermal displacement (STD) method, and for finite bias in both cases.

Note

- Be aware that most of the calculations are impractical to perform on a personal computer and requires some sort of dedicated computational server/supercomputer.
- The calculations will generate a lot of files, so we recommend that each type of calculation is done in a separate sub-folder. We provide the names we used and which are assumed in the scripts, but you may of course choose other names and then modify the subsequent scripts accordingly.

Calculations at 0 K and zero bias

First we calculate the PLDOS for the np-junction at zero temperature and bias. Do these calculations in a folder named `0K-zero-bias`. Take the relaxed structure from before, and attach a DeviceLCAOCalculator to it, with the following parameters:

- *k-point sampling:* 9 x 9 x 101.
- *Broadening:* 300 K.
- *Exchange-correlation functional:* LDA.PW.
- *Basis set:* SingleZetaPolarized.
- *Density mesh cutoff:* 75 Hartree
- *Poisson solver:* [Parallel] Conjugate Gradient.
- *Iteration control parameters:*
  - Maximum steps: 1000

- Tolerance: 0.0000001
- Number of history steps: 10

Add a **ProjectedLocalDensityOfStates** analysis object, with these parameters.

- *Energy interval:* -1 to 1 eV.
- *Method:* Device DOS.
- *k-point sampling:* 21 x 21.

Finally, you need to make a few manual changes to the *device_iteration_control_parameters* of the full device calculation:

- Remove *tolerance=1e-07,*
- Add *damping_factor=0.01,*

You can also download the script here: 0K-zero-bias.py Run the script, which should take about 20 minutes on a modern laptop.

Calculations at 300 K and zero bias

For the calculations with the STD configuration, we can do things a little simpler, to ensure that we use the same parameters for the calculator. Do these calculations in a folder named `300K-zero-bias`. Make a copy of the previous script and delete everything related to the structure and calculator, leaving only the PLDOS block. Replace the deleted part with this:

```
# -*- coding: utf-8 -*-
# -------------------------------------------------------------
# Two-probe Configuration
# -------------------------------------------------------------
filename = '../0K-zero-bias.py/0K-zero-bias.py.hdf5'
filename_dynmat = '../dyn-mat/ff-device-si.hdf5'
filesave = '300K-zero-bias.py.hdf5'
device_configuration = nlread(filename, DeviceConfiguration)[-1]
dynamical_matrix = nlread(filename_dynmat,DynamicalMatrix)[-1]
T = 300

device_configuration_STD = SpecialThermalDisplacement(device_configuration, dynamical_matrix=dynamical_
device_configuration_STD.update()
nlsave(filesave,device_configuration_STD)
```

This will read in the previously calculated Dynamical matrix as well as the configuration, including calculator, from the 0 K and zero bias calculation, and create a new configuration with the thermally displaced atoms and the same calculator as for 0 K. You may download the full script here: 300K-zero-bias.py

Calculations at finite bias

In this case study, we will first do the self-consistent part of the finite bias calculations, and then afterwards calculate the transmissions. We do this because convergence for the 300 K structure can be a little tricky, and it is easier to fine-tune the convergence for difficult bias-points when doing it in this way.

Tip

In QuantumATK we have implemented an object called IVCurve, which automatically calculates the transmission at specified bias values. This is very useful for relatively simple systems and not too high bias values, where there are no problems with the convergence of the SCF loop. In this case, the calculations with the thermally displaced device are more difficult to converge, so we will not use the

IVCurve object. For more information on how to use it, see Transport calculations with QuantumATK.

The script itself is actually fairly simple, as it simply reads in the zero bias calculation to use as a starting point, and then keeps re-using the previous bias-point as an initial state for the next one. Here we show the script for the un-displaced configuration:

```
# Set bias
# Positive: Reverse and Negative: Forward

pos_bias_list = [0.10, 0.20, 0.30, 0.40, 0.50, 0.55, 0.60, 0.70]*Volt
neg_bias_list = [-0.10, -0.20, -0.30, -0.40, -0.50, -0.60]*Volt

# Read DeviceConfiguration
zero_bias_file = '../0K-zero-bias/0K-zero-bias.py.hdf5'
device_configuration = nlread(zero_bias_file, DeviceConfiguration)[-1]

for bias in pos_bias_list:
    if processIsMaster():
        print "Bias is now: ", bias
    # Get the calculator
    calculator = device_configuration.calculator()

    # Set the bias voltage
    calculator=calculator(electrode_voltages=(bias/2, -bias/2))

    # Attach the calculator and use the old initial state
    device_configuration.setCalculator(
      calculator(),
      initial_state=device_configuration)
    device_configuration.update()
    nlsave('device_bias_%.2f.hdf5' % bias.inUnitsOf(Volt), device_configuration)

# Read DeviceConfiguration
device_configuration = nlread(zero_bias_file, DeviceConfiguration)[-1]

for bias in neg_bias_list:
    if processIsMaster():
        print "Bias is now: ", bias
    # Get the calculator
    calculator = device_configuration.calculator()

    # Set the bias voltage
    calculator=calculator(electrode_voltages=(bias/2, -bias/2))

    # Attach the calculator and use the old initial state
    device_configuration.setCalculator(
      calculator(),
      initial_state=device_configuration)
    device_configuration.update()
    nlsave('device_bias_%.2f.hdf5' % bias.inUnitsOf(Volt), device_configuration)
```

Run these 2 scripts, in folders named `0K-iv` and `300K-iv` respectively. The titles should be self-explanatory:

- 0K-iv-scf.py
- 300K-iv-scf.py

They should each take a couple of hours on a 16-core node. Afterwards, we want to calculate the transmissions for each of the configurations and the PLDOS for the most extreme bias points. The first is essentially a loop over the previously calculated files, using the same settings for the Transmission Spectrum as before. This should take a couple of hours on a 16-core node. The PLDOS scripts read in the converged calculation and calculates the PLDOS. They should each take 10-20 minutes using 4 MPI processes on a laptop.

- 0K-iv-transmission.py
- 300K-iv-transmission.py
- pldos-0K-forward.py
- pldos-0K-backward.py
- pldos-300K-forward.py
- pldos-300K-backward.py

Lowest Order Expansion (LOE) Calculations

Finally, we also need to make the LOE calculations, which takes the inelastic contributions into account through an approximation of the full electron-phonon coupling matrix. This approximation is valid when the Density of States (DOS) varies slowly around the Fermi level. For this, we need to calculate the Hamiltonian Derivatives, which is a fairly heavy calculation. We therefore sample the electronic Brillouin zone in the
Γ point only. This is a good approximation in this case, as tunneling is the dominant effect and is in turn dominated by transmission at
Γ. Note that this reduces the computational time by a factor of 9 compared to using the same k-point density as in the STD calculations.

Afterwards, we calculate the actual electron-phonon couplings, a calculation which can in general be very time-consuming, due to the many possible combinations of k- and q-points.

Tip

For more information on how to calculate the full electron-phonon coupling, see f.ex. the tutorial: Phonon-limited mobility in graphene using the Boltzmann transport equation.

We need to set up a calculator similar to before, with only the k-point sampling changed, followed by a calculation of the Hamiltonian derivatives. Do these calculations in a folder named `loe`. You can use a script similar to before, where you read in the calculator and modify it, or set everything up from scratch in a new script. Here we have chosen to read in the configuration from an existing file, but setting up the calculator anew. The Hamiltonian derivatives should look like this:

```
# -------------------------------------------------------------
# Hamiltonian Derivatives
# -------------------------------------------------------------
hamiltonian_derivatives = HamiltonianDerivatives(
    configuration=device_configuration,
    repetitions=(3, 3, 1),
    atomic_displacement=0.01*Angstrom,
    constraints=[0,1,2,3,44,45,46,47],
    use_equivalent_bulk=False,
    )
nlsave(filesave, hamiltonian_derivatives)
```

You may download the script here: ham-der.py. Ensure it points at the right files and run it. In our case, it took about 5 hours on a total of 48 cores with 1 TB of total memory.

Afterwards, we need to run the actual LOE calculations. We use only 1 q-point, in accordance with the paper, which is a good approximation in this case, as the tunneling is also dominated by the
Γ-point in the phonon Brillouin zone. This also gives a more direct comparison with the STD method. If the goal is to get as accurate LOE calculations as possible, the number of q-points sampled, should always be converged for the system of interest. You may download the script here: loe.py. Make sure it points at the right files, and run it. This should take no more than a few minutes on a laptop.

Tip

For more information on the different parameters of the inelastic transmission spectrum or further discussion of the k- and q-dependence of the transmission in this system, you may consult this tutorial: Inelastic current in a silicon p-n junction.

Computational timings

The main advantage of the STD method over the LOE method is computational efficiency, and we will therefore present the timings in a more systematic way. In the table, we show the timings for the full IV curve for each method, normalized to the total number of core-hours on equivalent machines.

Table 9 Computational timings in core-hours¶

	Noninteracting	STD	LOE
Time spent on IV Curve:	28 h	129 h	468 h

We see here that the STD method takes longer than the noninteracting calculation, which is mostly due to convergence being more difficult. In this case study, we have chosen a conservative approach which improves convergence, but at the expense of longer calculations. It is probably possible to find other settings which also converge, and use less time. With the parameters we have chosen for the LOE calculation, almost all the time is spent on calculating the Hamiltonian derivatives, and very little is spent on computing the actual electron-phonon interactions afterwards. However, this is in general not true, and the LOE part itself increases very quickly with the number of k- and q-points included. Additionally, as noted above, we have reduced the calculation time for the hamiltonian derivatives by a factor of 9 by reducing the k-point sampling to just the
Γ-point.

Note

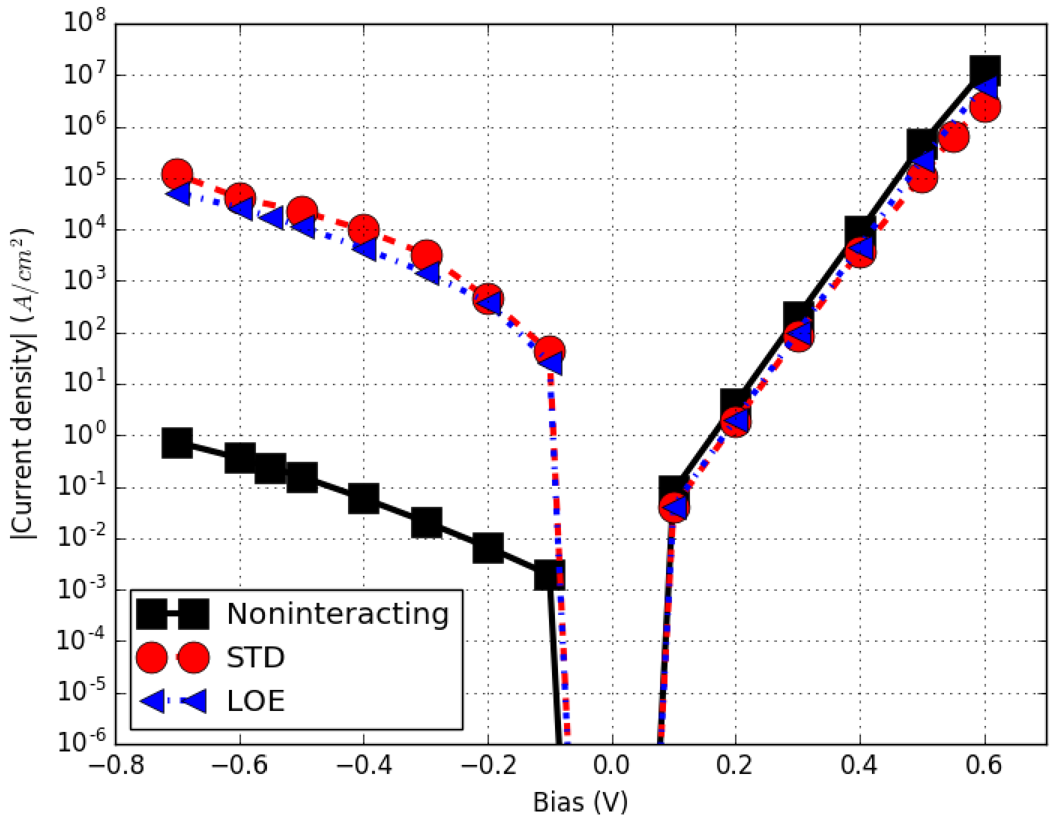
Note that the Hamiltonian derivatives were computed on special nodes with extra memory, which is an added complication of such demanding calculations.

Analysis and discussion

We have now calculated the current as a function of bias using three methods:

- **Noninteracting**: uses a pristine configuration and does not incorporate phonons in any way.
- **SpecialThermalDisplacement (STD)**: Uses a single thermally displaced configuration to account for phonons. This is a new method introduced in the paper this case study is based on.
- *Perturbation Theory (PT)* using the **Lowest Order Expansion (LOE)**: Approximate approach to include the full electron-phonon contribution to the current.

The main point of these calculations is to show that the STD method is as accurate as the LOE method, in this case, but at a much lower computational cost, as there is no need to compute the very expensive dH/dR matrix. We will now plot our data and see how they compare. Download this plotting script, make sure it points at the right files and run it: plot-iv.py. You should get a plot looking like this:



We see how inclusion of the electron-phonon coupling (EPC) drastically increases the reverse current. We also see that both the STD and LOE approaches show this increase, and are in clear agreement about the effect and importance of the electron-phonon coupling. Note that for the longer device, which is also treated in the paper [1], the results are also in good agreement with experiments. The increase in current reduces the $I_{\mathrm{ON}}/I_{\mathrm{OFF}}$ figure-of-merit to be on the order of 1 when going below a bias of ± 0.5 V. This means that the current will be essentially the same for both forwards and backwards bias, and the device no longer acts as a diode at low bias. See the table for details:

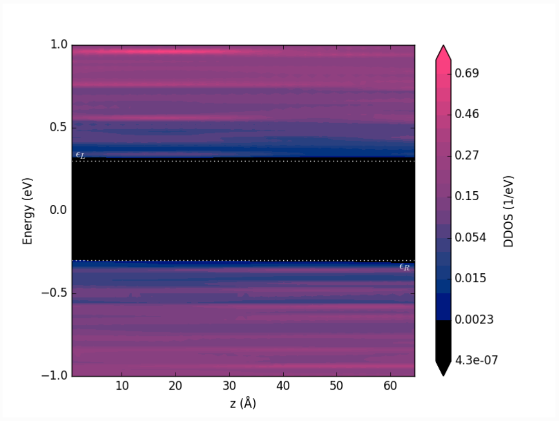
Table 10
$I_{\mathrm{ON}}/I_{\mathrm{OFF}}$ *figure-of-merit.¶*

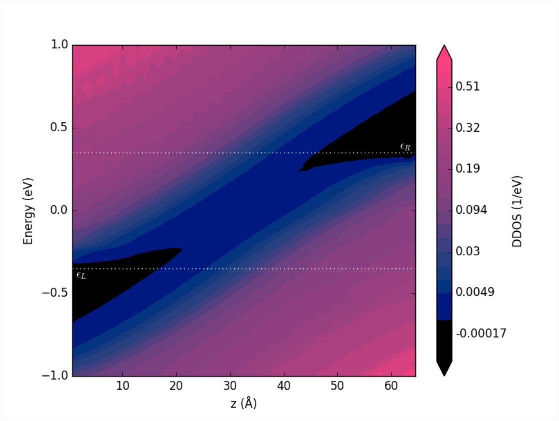
Bias [V]	Noninteracting	STD	LOE
± 0.5	$8 \cdot 10^6$	5	20
± 0.6	$7 \cdot 10^7$	58	200

Note

The $I_{\mathrm{ON}}/I_{\mathrm{OFF}}$ figure-of-merit is a basic property of diodes and describes how effective the diode is at suppressing current at reverse bias. Specifically, it is the current under forward bias divided by the absolute value of the current under a reverse bias of equal magnitude. So if it is 10 at 0.5 V, it means that the current at 0.5 V forward bias is 10 times larger than the reverse current under a reverse bias of -0.5 V.

In order to understand why phonons are very important under reverse bias, and negligible under forward bias, we need to consider the two cases in more detail. Specifically, the mechanism of transport differs between forward and reverse bias. Under forward bias, the bands are essentially flat and carriers can pass unhindered from one electrode to the other. Under reverse bias, there are no states around the Fermi level in the central region and carriers can only travel between the electrodes by tunneling. This can also be seen from a comparison of the PLDOS at forward and reverse bias, see figures below. We show the PLDOS without electron-phonon coupling included, to make the features more clear. The left/right PLDOS is at forward/reverse bias.





To a first approximation, the band-to-band tunneling probability decreases when $|k|$ increases, and increases when the carrier energy increases. Therefore, the tunneling probability will be higher if the carrier can increase its energy without increasing the momentum. This leads us to the role of phonons. This explains why we can use just the Γ-point for both k- and q-points, as the interaction gives maximum tunneling probability there. The fact that the transmission peaks at the Γ-point, is shown in more detail in the tutorial: Inelastic current in a silicon p-n junction.

The STD method compared to the LOE method

In this case study, we have used two quite different methods to account for the electron-phonon interaction. The LOE method uses perturbation theory and Green's functions to calculate the transitions between states with varying k- and q-values. The STD method simply displaces all the atoms according to a weighted average of the phonons, so that the correct thermal average of the current is obtained. It does not model the electron-phonon interactions explicitly. Beforehand, there was no guarantee that the agreement would be as good as shown here, and this will not be true in all cases. Especially if the relevant processes require momentum transfer, the STD method should not be expected to give as good results as the LOE method. However, as the two methods rely on different approximations and assumptions, it is also no guarantee that the LOE method will always give better results than the STD method. Additionally, the STD method is much faster and less memory-demanding.

References

[1] (1,2,3)
Tue Gunst, Troels Markussen, Mattias L. N. Palsgaard, Kurt Stokbro, and Mads Brandbyge. First-principles electron transport with phonon coupling: Large scale at low cost. *Phys. Rev. B*, 96(16):161404, oct 2017. arXiv:1706.09290, doi:10.1103/PhysRevB.96.161404.

[2]
Marios Zacharias and Feliciano Giustino. One-shot calculation of temperature-dependent optical spectra and phonon-induced band-gap renormalization. *Phys. Rev. B*, 94:075125, Aug 2016. doi:10.1103/PhysRevB.94.075125.

Previous Next