The present work was submitted to the High-Performance and Automatic Computing group.

DISTRIBUTED PARALLEL NON-EQUILIBRIUM GREEN'S FUNCTION APPROACH TO INELASTIC CHARGE TRANSPORT ACROSS SCHOTTKY JUNCTIONS

ALEXANDER SEBASTIAN ACHILLES

Master-Thesis

January 2017

Supervised by
Prof. Paolo Bientinesi, PhD
Dr. Edoardo Di Napoli
Dr. Urs Aeberhard

ABSTRACT

Inelastic charge transport is of paramount importance for simulating quantum transport in mesoscopic systems with the focus on novel nano-transistor concepts [7, 8] or quantum photovoltaic devices [1, 2]. The Non-Equilibirum Green's Functions (NEGF) framework is an advanced simulation approach that allows for treatment of transport phenomena and out-of-equilibrium transport. The NEGF formalism was introduced by Schwinger and co-workers [26], Kadanoff and Baym [14], and Keldysh [15] and provides a quantum-statistical mechanics picture of many-body systems far from equilibrium. The NEGF simulation is computationally very intense, due to the non-local, spectral energy and momentum dependencies of the Green's functions and the nested self-consistent loops. The model includes open boundary conditions that couple the device to an out-of-equilibrium environment where electrons move due to externally applied voltages.

The computational intensity requires a massively parallel implementation to exploit the available resources of a state-of-the-art supercomputer. Four different level of parallelism are offered within the NEGF algorithm. The first level parallelism is trivial, since no dependencies are present. The next two levels are parallelized with data distribution across the nodes employing the Message Passing Interface (MPI). The last level is parallelized on the nodes level utilizing domain decomposition. Consequently the presented parallelization approach is hybrid and emphasizes the advantages of the two different worlds: distributed and shared memory paradigm.

The presented parallelization approach is scaling on up to 458,752 cores on JUQUEEN and sets the foundation for unprecedented simulations of large systems, e.g. the simulation of a complete realistic device.

ACKNOWLEDGMENTS

First of all I want to thank Urs, Edoardo, Jonas, and my father for proofreading my thesis. This has profoundly improved the composition of this thesis.

Thanks as well to all my friends, who might not realize that they played such a central role. You helped me relaxing in stressful times.

I am grateful that the whole HPAC group warmly welcomed and integrated me into the group. Particularly I am grateful for the technical discussions with Jan, Paul and Elmar.

I also want to thank all the people at the JSC. All lot of people helped me with there expertise and experience. Particularly I want to mention my office mate David. He helped me with technical discussions and the references to other experts at the JSC. And he supported me on the Jülich days with the most essential liquid: freshly grounded and at Barista level prepared coffee mad the Jülich days particularly enjoyable and effective.

Special thanks belong to Urs for our weekly meetings. Without him the daunting task would have been a lot more difficult. I also want to thank Paolo and Edoardo. First of all for bringing me into contact with this very interesting but also challenging topic. But also for providing encouragement, guidance and expertise along the way.

Finally, I want to give my gratitude to my family, particularly my parents, without whose manifold support the completion of my thesis would be merely impossible.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The proper treatment of inelastic quantum charge transport in mesoscopic devices is one of the most challenging problems in modern nanostructures. Several models exist to compute electronic transport, cf. table 1. Large macroscopic systems are well described via the Boltzmann equation. This approach is only valid under the assumptions, that scattering processes are localized and instantaneous, only weak scattering occurs, and only slow events compared to the free mean time between collisions are of interest [10].

In contrast to macroscopic systems, the technological development over the last decades rapidly evolved towards smaller nanodevices. This allows to produce devices with characteristic dimensions that are smaller than the mean free path. One of the most well known example is the semiconductor industry: state-of-the-art lithography allows to manufacture chips with 14 nm process technology. In 2017 chips with 10 nm process technology are expected. Therefore the assumptions for the Blotzmann equation are no longer fulfilled, which makes this method invalid. Due to these small distances quantum effects like quantization or confinement become important.

The devices on this length scale are called nanodevices or mesoscopic devices. The latter term indicates that these structures are large compared to the microscopic scale but small compared to the macroscopic scale on which the Boltzmann model is valid.

Approximate	Model	Limitations	Easy, Fast
semi-classical	Boltzmann transport equations, classical Monte Carlo	only accurate up to classical limits	
	Density functional theory	Least accurate	
	Quantum Monte Carlo	classical features + quantum corrections	
quantum	Wigner Function, Density Matrix	Accurate to up to single particle description	
	Green's functions methods	Most accurate, but computationally complex	
Exact	Direct solution of the n-body Schrdinger equation	Accurate, but can be solved only for small number of particles	Difficult

Table 1: Hierarchy of Transport Models.

The Non-Equilibirum Green's Functions (NEGF) formalism is an advanced simulation approach for treatment of transport phenomena and out-of-equilibrium transport. The NEGF formalism was introduced by Schwinger and co-workers [26], Kadanoff and Baym [14], and Keldysh [15] and provides a quantum-statistical mechanics picture of many-body systems far from equilibrium. The NEGF simulation is computationally very intense, because it includes open boundary conditions that couple the device to an out-of-equilibrium environment where electrons move due to externally applied voltages.

1.1 GOALS OF THIS WORK

As described before the non-equilibrium Green's function (NEGF) method is the most accurate but also most computationally complex and costly approach to simulate nanodevices. The goal of this thesis is to develop a distributed parallelization approach for NEGF method. This will allow to reduce the wall clock time and sets the foundation for unprecedented simulation of larger system.

As an example we calculate characteristic properties of a nin-diode for a ballistic case and with inelastic scattering. These properties are the potential, the charge density, the I-V characteristics and the spectral current distribution. We will also investigate the nin-diode with a high barrier.

1.2 OUTLINE OF THE THESIS

In chapter 2 we will describe the NEGF method in general, the partitioning of the system, the Hamiltonian, and the related physical quantities. The following chapter 3 will deal with the discretization and the description of the NEGF algorithm. The different implementations and parallelization are discussed in chapter 4. This includes a first implementation in Matlab to understand the algorithm, an OpenMP parallelization, a distributed parallelization with the Message Passing Interface and a hybrid implementation which combines the Message Passing Interface (MPI) implementation with an OpenMP parallelization on the node level. Chapter 5 provides a brief description of the soft- and hardware that has been used and presents the results from the parallelization as well as physical results. The last chapter 6 gives a conclusion and highlights how this work can be used in the future as well as which further steps to improve the NEGF implementation.

1.3 RELATED WORK

The NEGF formalism has found widespread application in the fields of quantum optical devices based on high-density electron-hole plasmas such as solid state lasers [13], but also in the field of quantum transport in mesoscopic systems with the technological focus on novel nano-transistor concepts [7, 8]. More recently, the approach has been extended to optoelectronic applications at the interface between quantum optics and quantum transport, such as quantum cascade lasers [18, 22], LED's [31] and quantum photovoltaic devices [1, 2].

There are two other NEGF implementations that are also parallelized. *Omen* [25], developed by Mathieu Luiser, and *Nemo5* [32], developed at Purdue. For ballistic simulations Omen is scaling on up to 221,400 cores with 20 bias points on a Cray-XT5 machine [25]. Since the bias points can be parallelized trivially this is equivalent to scaling on 11,000 cores for 1 bias points. For

inelastic electron-phonon scattering Omen is scaling on up to 170,400 cores again for 20 bias points (= 8520 cores for 1 bias point).

2

THE NONEQUILIBRIUM GREEN'S FUNCTIONS METHOD

In this chapter the physical description of the system is specified. We introduce the Hamiltonian, the Green's function approach, and the concept of self-energies to setup the fundamental basis for the NEGF computation.

2.1 INTRODUCTION

We want to describe steady-state electronic transport in a quasi 1-dimensional system of mesoscopic dimensions under the constraint of inelastic scattering with phonons. This requires the usage of quantum transport theory that goes beyond the ballistic regime. The configuration variables of our system are the vertical coordinate z, the transverse momentum $k_{\|}$, which corresponds to the Fourier transform of (x, y) and the energy E. The unperturbed system describes free electrons in a solid, and the electron-phonon interaction is treated as a perturbation. Thus the physical system that we consider includes electrons, phonons and doping. Doping is the addition of impurities which changes the electronic properties of the semiconductor. The Born-Oppenheimer approximation, the assumption that the motion of atomic nuclei and electrons in a molecule can be separated has been used. The tight-binding model is used to describe the valence electrons.

2.2 PARTITIONING OF THE SYSTEM

Different aspects need to be considered for the theoretical description of transport phenomena. Therefore the system is partitioned into three fundamental contributions: The first part is the non-interaction system, that describes the system's kinetic energy and the potential energy associated with the interacting of the valence electrons with the ion cores, which is described by the Hamiltonian H_0 (see section 2.3). The second aspect concerns the open boundary conditions of the system, i.e. the coupling of the system to the contacts. The third part are the interactions within the system. In the following we will describe the electron-phonon interaction.

2.3 HAMILTONIAN

In quantum mechanics the physical quantities like energy, position, and momentum are observables which correspond to the eigenvalues of the corresponding operators. In the case of the total energy E of an quantum-mechanical system, the associated operator is the *Hamiltonian* operator $\hat{H}$,

and the corresponding eigenvalue problem amounts solving the associated time-independent Schrödinger equation:

$$\hat{H}|\psi\rangle = E|\psi\rangle. \tag{1}$$

In Dirac notation the state of the system is described by the "ket" vector $|\psi\rangle \in \mathcal{H}$ (Hilbert space), and by definition it is normalized against the "bra" vector (dual space to $\mathcal{H}$) $\langle\psi|\psi\rangle = 1$.

The full Hamiltonian can be partitioned into the diagonal contribution of the non-interacting system and the interaction:

$$H = H_0 + H_I \tag{2}$$

In general, the Hamiltonian will describe all relevant degrees of freedom of a system (electronic, vibrational, optical, etc.) and their mutual interactions. Here, H_0 is restricted to the electronic system, and only the electron-phonon interaction term $H_{el\text{-}ph}$ is included in H_I, because it is the only interaction being considered in this work and relevant to the physical phenomena we want to study. Using a simple single band effective mass approximation for mobile electrons in semiconductors, the Hamiltonian of non-interacting electrons in 1D can be written as follows

$$H_0 = -\frac{\hbar^2}{2}\nabla_{\mathbf{r}}\left(\frac{1}{m^\star(\mathbf{r})}\nabla_{\mathbf{r}}\right) + U(\mathbf{r}), \tag{3}$$

where $m^\star$ denotes the effective mass of the electron, the spatial vector $\mathbf{r} = (x, y, z)$, and $\Delta_{\mathbf{r}} = \nabla^2_{\mathbf{r}}$ is the Laplace operator with respect to the spatial component $\mathbf{r}$ and $U(\mathbf{r}) = U(z)$ is the effective potential of mobile electrons in the solid along z.

The matrix elements of this effective-mass Hamiltonian correspond to the matrix elements of the single-band tight-binding Hamiltonian [21].

The extension to a quasi-1D system with isotropy in the transverse plane results in a dependence on transverse quasi-momentum $\mathbf{k} = (k_x, k_y)$:

$$H_0(\mathbf{k}) = -\frac{\hbar^2}{2}\nabla_z\left(\frac{1}{m^\star(z)}\nabla_z\right) + \epsilon_k + U(z) \tag{4}$$

$$= -\frac{\hbar^2}{2}\nabla_z\left(\frac{1}{m^\star(z)}\nabla_z\right) + \frac{\hbar^2 k^2}{2m^\star} + U(z) \tag{5}$$

2.4 GREEN'S FUNCTIONS

Green's functions are an important tool in theoretical physics. In mathematics, Green's functions are used to solve inhomogeneous differential equations. In many-body theory a Green's function describes the quantum statistical ensemble average of a pair of field operators (creation and annihilation operators).

The Green's functions are useful for solution of perturbation theory problems [6, 11]. For an illustration how the Hamiltonian and the Green's function are related, let's considering the time-independent *Schrödinger equation*

$$[H_0(\mathbf{r}) + V(\mathbf{r})]\psi = E\psi. \tag{6}$$

In this case V is an perturbation of the system (not to be confused with the potential U, mentioned above). In order to solve the Schrödinger equation we define the corresponding Green's function by the differential equation

$$[E - H_0(\mathbf{R})]G_0(\mathbf{r}, \mathbf{r}', E) = \delta(\mathbf{r} - \mathbf{r}'). \tag{7}$$

where the boundary condition, $G_0(\mathbf{r}, \mathbf{r}', E) = G_0(\mathbf{r}', \mathbf{r}, E)$ is assumed. It is straight forward to identify the operator $[E - H_0(\mathbf{r})]$ as the inverse of $G_0(\mathbf{r}, \mathbf{r}', E)$. Thus we can write:

$$G_0^{-1}(\mathbf{r}, \mathbf{r}', E) = E - H_0(\mathbf{r}), \tag{8}$$

or

$$G_0^{-1}(\mathbf{r}, \mathbf{r}', E)G_0(\mathbf{r}, \mathbf{r}', E) = \delta(\mathbf{r} - \mathbf{r}'). \tag{9}$$

Inserting this equation into the Schrödinger equation yields

$$[G_0^{-1}(\mathbf{r}, \mathbf{r}', E) - V(\mathbf{r})]\psi = 0. \tag{10}$$

The integral equation

$$\psi(\mathbf{r}) = \psi^0(\mathbf{r}) + \int \mathrm{d}\mathbf{r}'\, G_0(\mathbf{r}, \mathbf{r}', E)V(\mathbf{r}')\psi(\mathbf{r}'), \tag{11}$$

solves equation 10. This can be shown by inserting eq. 11 into eq. 10

$$\begin{aligned} &G_0^{-1}(\mathbf{r}, \mathbf{r}', E)\psi^0(\mathbf{r}) \\ &+ \int \mathrm{d}\mathbf{r}', G_0^{-1}(\mathbf{r}, \mathbf{r}', E)G_0(\mathbf{r}, \mathbf{r}', E)V(\mathbf{r}')\psi(\mathbf{r}') - V(\mathbf{r})\psi(\mathbf{r}) = 0. \end{aligned} \tag{12}$$

Using eq. 9 yields to

$$\begin{aligned} &G_0^{-1}(\mathbf{r}, \mathbf{r}', E)\psi^0(\mathbf{r}) \\ &+ \int \mathrm{d}\mathbf{r}'\delta(\mathbf{r} - \mathbf{r}')V(\mathbf{r}')\psi(\mathbf{r}') - V(\mathbf{r})\psi(\mathbf{r}) = 0 \end{aligned} \tag{13}$$

$$\Leftrightarrow G_0^{-1}(\mathbf{r}, \mathbf{r}', E)\psi^0(\mathbf{r}) + V(\mathbf{r})\psi(\mathbf{r}) - V(\mathbf{r})\psi(\mathbf{r}) = 0 \tag{14}$$

$$\Leftrightarrow G_0^{-1}(\mathbf{r}, \mathbf{r}', E)\psi^0(\mathbf{r}) = 0 \tag{15}$$

which corresponds to the initial Schrödinger equation. Thus the integral equation 11 solves 10. We can now solve the integral equation 11 by iteration. Up to first order in V the solution is:

$$\psi(\mathbf{r}) = \psi^0(\mathbf{r}) + \int \mathrm{d}\mathbf{r}'\, G_0(\mathbf{r}, \mathbf{r}', E)V(\mathbf{r}')\psi^0(\mathbf{r}') + \mathcal{O}(V^2), \tag{16}$$

where ψ^0 is an eigenstate to the non-interacting Hamiltonian H_0 with eigenenergy E.

We can also write the results eq. 11 in terms of the full Green's function

$$\psi(\mathbf{r}) = \psi^0(\mathbf{r}) + \int \mathrm{d}\mathbf{r}' G(\mathbf{r}, \mathbf{r}', E) V(\mathbf{r}') \psi^0(\mathbf{r}'). \tag{17}$$

For completeness we will note the general definition of the Green's functions as they are usually defined in the literature [6, 11]. The retarded Green's function is defined by

$$G^R(x,t;x',t') = -i\Theta(t-t') \left\langle \left[\Psi(x,t), \Psi^\dagger(x',t') \right]_\pm \right\rangle. \tag{18}$$

Conventionally the upper sign corresponds to the fermion case, whereas the lower case is for bosons. $[A,B]_+ = \{A,B\} = AB + BA$ is the anticommutator and $[A,B]_- = [A,B] = AB - BA$ is the ordinary commutator. $\Psi^\dagger$ is the field operator that creates a particle a position x and time t. The retarded function is nonzero only for $t > t'$. We also define the advanced Green's function which is nonzero only for $t < t'$:

$$G^A(x,t;x',t') = +i\Theta(t-t') \left\langle \left[\Psi(x,t), \Psi^\dagger(x',t') \right]_\pm \right\rangle. \tag{19}$$

it is convenient to define also two more Green's functions:

$$G^>(x,t;x',t') = -i \left\langle \left[\Psi(x,t), \Psi^\dagger(x',t') \right]_\pm \right\rangle, \tag{20}$$

$$G^<(x,t;x',t') = -i(\mp) \left\langle \left[\Psi(x,t), \Psi^\dagger(x',t') \right]_\pm \right\rangle. \tag{21}$$

Thus we can rewrite the retarded and advanced Green's function:

$$G^R(x,t;x',t') = \Theta(t-t') \left[G^>(x,t;x',t') - G^<(x,t;x',t') \right], \tag{22}$$

$$G^A(x,t;x',t') = \Theta(t'-t) \left[G^<(x,t;x',t') - G^>(x,t;x',t') \right]. \tag{23}$$

The average $\langle \ldots \rangle$ in the definitions of the Green's functions is the quantum statistical ensemble average.

We are interested in the stationary state, which means only the time difference $t - t'$ is important. Thus, it is convenient to use the Green's function in energy representation rather than in time representation. To change the representation we perform a Fourier transform

$$G^R(\mathbf{r}, \mathbf{r}', E) = \int G(\mathbf{r}, \mathbf{r}', t)\, \mathrm{e}^{\mathrm{i}Et/\hbar}\, \mathrm{d}t. \tag{24}$$

Since we are using a quasi-1D Hamiltonian with isotropy in the transverse plane and a dependence on transverse quasi-momentum, the Green's functions that we are using in the following depend on the following arguments:

$$G^R(\mathbf{r}, \mathbf{r}', E) = G^R(z, z'; \mathbf{k}, E) \tag{25}$$

We can verify the following relation:

$$G^R(z, z'; \mathbf{k}, E) = G^A(z, z'; \mathbf{k}, E)^* \tag{26}$$

We can use the Keldysh equation [2, 8, 21, 27, 30] to obtain the lesser Green's function:

$$G^<(z, z'; \mathbf{k}, E) = \int \mathrm{d}z_1 \int \mathrm{d}z_2 \, G^R(z, z_1; \mathbf{k}, E) \Sigma^<(z_1, z_2; \mathbf{k}, E) G^A(z_2, z'; \mathbf{k}, E). \tag{27}$$

Because the set of Green's functions is coupled, we can calculate the greater Green's function by

$$G^>(z, z'; \mathbf{k}, E) = G^R(z, z'; \mathbf{k}, E) - G^A(z, z'; \mathbf{k}, E) + G^<(z, z'; \mathbf{k}, E). \tag{28}$$

2.5 DYSON EQUATION AND SELF-ENERGY

The Dyson equation defines the equation of motion of the electrons. It is defined by (suppressing the arguments and summations)

$$G^R = G_0^R + G_0^R \Sigma^R G^R \tag{29}$$

where Σ^R defines the retarded self-energy. G_0^R is the interaction-free retarded Green's function.

For describing transport phenomena it is important to include the exchange of particles with the environment, i.e. the contact. That requires the creation and annihilation of particles. The vacuum is not empty anymore, particles can be created and annihilated. A heuristic explanation of the self-energies is that it contains all possible interactions of virtual particles with other particles and the vacuum.

From equation 29 we get an expression for the full Green's function:

$$\Leftrightarrow (1 - G_0^R \Sigma^R) G^R = G_0^R \tag{30}$$

$$\Leftrightarrow G^R = G_0^R \left[1 - G_0^R \Sigma^R\right]^{-1} \tag{31}$$

$$= \left[(G_0^R)^{-1} - \Sigma^R\right]^{-1}. \tag{32}$$

By inserting the $(G_0^R)^{-1} = E - H_0$, we get the following expression for the Dyson equation:

$$G^R(z, z'; k, E) = \left[E - H_0(z, z'; k, E) - \Sigma^R(z, z'; k, E)\right]^{-1} \tag{33}$$

2.6 CHARGE DENSITY

For given spin, the spectral electron density as expressed by the electron GF components reads

$$n(z,E) = -\mathrm{i}\sum_{\mathbf{k}} G^{<}(z,z;\mathbf{k},E)/(2\pi). \tag{34}$$

2.7 POISSON EQUATION

Charge density and electrostatic potential are connected by the Poisson equation:

$$\partial_z\left(\epsilon(z)\partial_z\left(-\frac{U(z)}{e}\right)\right) = e(n(z) - N_d(z)) \tag{35}$$

ϵ denotes the static dielectric constant and N_d denotes the ionized dopant concentration. This equation needs to be solved self-consistently. Thus, it needs to be iterated according to the Dyson and Keldysh equations. At every step of the self-consistent iteration, the potential is then updated with the a simple Kerker mixing scheme:

$$U^{i+1} = \lambda U^{new} + (1-\lambda)U^{i}, \tag{36}$$

where U^{new} is the new potential that was calculated by the Poisson equation. Once the self-consistency loop has converged, the potential is compatible with the charge density that depends in turn on the Green's function.

2.8 SPECTRAL FUNCTION AND LOCAL DENSITY OF STATES

The spectral function is defined by

$$\mathcal{A}(\mathbf{k},z,E) = \mathrm{i}\left[G^{R}(z,z;\mathbf{k},E) - G^{A}(z,z;\mathbf{k},E)\right]. \tag{37}$$

The local density of states describes the spatially resolved density of states. It's defined by

$$\mathrm{LDOS}(z,E) = \sum_{\mathbf{k}}\mathrm{LDOS}(z,\mathbf{k},E) = -\frac{1}{\pi}\sum_{\mathbf{k}}\mathrm{Im}\,G^{R}(z,z;\mathbf{k},E), \tag{38}$$

where summation over spin was assumed in the last equality.
Since $\mathrm{Im}\,G^{R} = \frac{i}{2}\left(G^{R} - G^{A}\right)$ we can rewrite the local density of states as

$$\mathrm{LDOS}(z,E) = \frac{1}{2\pi}\sum_{\mathbf{k}}\mathcal{A}(\mathbf{k},z,E) \tag{39}$$

2.9 COUPLING TO THE CONTACT

The coupling to the left and right contact is done via a self-energy:

$$\Sigma^R_{LR}(k,E) = \begin{pmatrix} -t\exp(ik_L a) & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & -t\exp(ik_R a) \end{pmatrix} \tag{40}$$

where t is the hopping parameter, defined in eq. 58, $k_{L/R}$ is the longitudinal wave vector in the contact, and a is the lattice parameter.

2.10 ELECTRON-PHONON SCATTERING

The general form for the lesser/greater electron-phonon interaction self-energy on the level of the first self-consistent Born approximation (SCBA) and assuming free equilibrium phonons is given by [20, 21, 30]

$$\begin{aligned}\Sigma^{\gtrless}_{z,z'}(\mathbf{k},E) = \frac{1}{V}\sum_{\mathbf{q}}|U_{\mathbf{k}-\mathbf{q}}|^2\big[\, & N_{LO}G^{<}_{z,z'}(\mathbf{q},E-\hbar\omega) \\ & + (N_{LO}+1)G^{<}_{z,z'}(\mathbf{q},E+\hbar\omega)\big].\end{aligned} \tag{41}$$

For dispersionless polar optical phonons with a constant energy $\hbar\omega_{LO}$ the interaction strength is given by [21, 30]

$$U_q = \sqrt{\frac{e^2\hbar\omega_{LO}}{2V}\left(\frac{1}{\epsilon_\infty}-\frac{1}{\epsilon_0}\right)}\frac{q}{q^2+q_0^2}, \tag{42}$$

where ϵ_0 is the static dielectric constant, ϵ_∞ is the high frequency dielectric constant and q_0 is the inverse Debye screening length. N_{LO} is the phonon occupation number, which corresponds to the Bose distribution:

$$N_{LO} = \frac{1}{\mathrm{e}^{\frac{\hbar\omega_{LO}}{kT}}-1}. \tag{43}$$

Thus, we can write the electron-phonon self-energy as

$$\begin{aligned}\Sigma^{\gtrless}_{z,z'}(\mathbf{k},E) = & \frac{\mathrm{e}^2\hbar\omega_{LO}}{2V}\left(\frac{1}{\epsilon_\infty}-\frac{1}{\epsilon_0}\right)\frac{V}{(2\pi)^3}\int_0^{q_{max}} q\,\mathrm{d}q \\ & \int_{-\pi/a}^{\pi/a}\mathrm{d}q_z\int_0^{2\pi}\mathrm{d}\phi\ \mathrm{e}^{iq_z(z-z')} \\ & \times\frac{q_z^2+q^2+k^2+2kq\cos\phi}{\left(q_z^2+q^2+k^2+q_0^2+2kq\cos\phi\right)^2} \\ & \times\big[N_{LO}G^{\gtrless}(\mathbf{q},E\pm\hbar\omega_{LO}) \\ & \quad + (N_{LO}+1)G^{\lessgtr}(\mathbf{q},E\mp\hbar\omega_{LO})\big]_{z,z'}.\end{aligned} \tag{44}$$

This can be simplified to:

$$\begin{aligned}\Sigma^{\gtrless}(\mathbf{k},E) =& \frac{e^2\hbar\omega_{LO}}{4\pi^2}\int_0^{q_{\max}} \mathrm{d}q\, qF(q,\Delta_{z,z'},k,q_0)\times \\ &\times\big(N_{LO}\times G^{\gtrless}(\mathbf{q},E\pm\hbar\omega_{LO}) \\ &\quad + (N_{LO}+1)G^{\gtrless}(\mathbf{q},E\mp\hbar\omega_{LO})\big)\end{aligned} \tag{45}$$

where $\Delta_{z,z'} = |z-z'|$ and the function F is defined by

$$\begin{aligned}F(q,\Delta_{ij},k,q_0) = &\int_0^{\pi/a}\mathrm{d}q\,\cos(q_z\Delta_{z,z'}) \\ &\left(\frac{1}{\sqrt{(q_z^2+q^2+q_0^2+k^2)^2-4k^2q^2}}\right. \\ &\left. -q_0^2\frac{q_z^2+q^2+q_0^2+k^2}{\left((q_z^2+q^2+q_0^2+k^2)^2-4k^2q^2\right)}\right).\end{aligned} \tag{46}$$

The function F doesn't change during the self-consistency loop, thus we can precompute this quantity and store it in memory.

The retarded self-energy is defined by [1, 19, 20, 22, 30]:

$$\begin{aligned}\Sigma^R_{z,z'}(\mathbf{k},E) =& \frac{1}{2}\Big(\Sigma^{>}_{z,z'}(\mathbf{k},E)-\Sigma^{<}_{z,z'}(\mathbf{k},E)\Big) \\ &-\mathrm{i}\mathcal{P}\int\frac{\mathrm{d}E'}{2\pi}\,\frac{\Sigma^{<}_{z,z'}(\mathbf{k},E')-\Sigma^{>}_{z,z'}(\mathbf{k},E)}{E-E'}\end{aligned} \tag{47}$$

$$\begin{aligned}=& i\int\frac{\mathrm{d}E'}{2\pi}\sum_{\mathbf{q},\lambda}|U_{\mathbf{q},\lambda}|^2\Big(D^R_{\mathbf{q},\lambda}G^R(\mathbf{k}-\mathbf{q},E-E') \\ &+D^R_{\mathbf{q},\lambda}(E')G^{<}(\mathbf{k}-\mathbf{q},E-E') \\ &+D^{<}_{\mathbf{q},\lambda}(E')G^R(\mathbf{k}-\mathbf{q},E-E')\Big).\end{aligned} \tag{48}$$

Here $\mathcal{P}$ denotes the Cauchy principal value.

On the same level of approximation (SCBA with free equilibrium phonons), the retarded self energy can be written as [1, 22, 30]

$$\begin{aligned}\Sigma^R_{z,z'}(\mathbf{k},E) =& \frac{e^2\,\hbar\omega_{LO}}{2V}\left(\frac{1}{\epsilon_\infty}-\frac{1}{\epsilon_0}\right)\frac{V}{(2\pi)^3}\int_0^{q_{max}} q\,\mathrm{d}q \\ &\times \int_{-\pi/a}^{\pi/a}\mathrm{d}q_z\int_0^{2\pi}\mathrm{d}\phi\; e^{iq_z(z-z')} \\ &\times \frac{q_z^2+q^2+k^2+2kq\cos\phi}{\left(q_z^2+q^2+k^2+q_0^2+2kq\cos\phi\right)^2} \\ &\times \Bigg[N_{LO}G^R(\mathbf{q},E\pm\hbar\omega_{LO}) \\ &\quad + (N_{LO}+1)G^R(\mathbf{q},E\mp\hbar\omega_{LO}) \\ &\quad + \frac{1}{2}G^<(\mathbf{q},E-\hbar\omega_{LO}) - \frac{1}{2}G^<(\mathbf{q},E+\hbar\omega_{LO}) \\ &\quad + i\mathcal{P}\int\frac{\mathrm{d}E}{2\pi}\left(\frac{G^<(\mathbf{q},E-E')}{E'-\hbar\omega_{LO}} - \frac{G^<(\mathbf{q},E-E')}{E'+\hbar\omega_{LO}}\right)\Bigg]_{z,z'}\end{aligned} \tag{49}$$

Conventionally, the principle value integral is neglected, thus

$$\begin{aligned}\Sigma^R_{z,z'}(\mathbf{k},E) =& \frac{e^2\,\hbar\omega_{LO}}{2V}\left(\frac{1}{\epsilon_\infty}-\frac{1}{\epsilon_0}\right)\frac{V}{(2\pi)^3}\int_0^{q_{max}} q\,\mathrm{d}q \\ &\times \int_{-\pi/a}^{\pi/a}\mathrm{d}q_z\int_0^{2\pi}\mathrm{d}\phi\; e^{iq_z(z-z')} \\ &\times \frac{q_z^2+q^2+k^2+2kq\cos\phi}{\left(q_z^2+q^2+k^2+q_0^2+2kq\cos\phi\right)^2} \\ &\times \Big[N_{LO}G^R(\mathbf{q},E\pm\hbar\omega_{LO}) \\ &\quad + (N_{LO}+1)G^R(\mathbf{q},E\mp\hbar\omega_{LO}) \\ &\quad + \frac{1}{2}G^<(\mathbf{q},E-\hbar\omega_{LO}) - \frac{1}{2}G^<(\mathbf{q},E+\hbar\omega_{LO})\Big]_{z,z'}\end{aligned} \tag{50}$$

3

FROM PHYSICS TO MATH

In this chapter the mathematical interpretation of the physical equation is discussed and the algorithm is presented. We will start with a translation of the physical description presented in chapter 2 into a mathematical picture and later into a computer understandable representation. The algorithm for the ballistic case is presented as well as the inelastic case.

3.1 MATHEMATICAL DESCRIPTION

The physical description needs some mathematical extension to form a complete picture. One of these objectives is the discretization that was briefly mentioned in chapter 2. Other important issues are the integration on a discretized grid, the inversion of the retarded Green's function and the treatment of the boundary conditions for the energy shift in the electron-phonon self-energy.

3.1.1 *Discretization*

The physical equations given in chapter 2 assume an infinite continuous space. However computers usually work with finite-precision arithmetic. Thus we have to transform the continuous space into a finite discrete grid. Therefore we need to discretize and limit the spatial domain, the energy and the momentum domain. The spatial domain is discretized by an equidistant discrete lattice (cf. equation 52). In the input file the lattice parameter a and the number of points N_p are specified. By these two values the total size of the system is specified by:

$$z_{max} = a \times N. \tag{51}$$

Thus, the grid points on the equidistant grid are located at

$$z = i_z \times a \tag{52}$$

where $i_z \in \mathbb{N}_0$. One can approximate the derivative in the Hamiltonian by finite differences:

$$\nabla_z \psi(z)|_{z=(i_z+\frac{1}{2})\times a} = \frac{\psi_{i_z+1} - \psi_{i_z}}{a}. \tag{53}$$

By repeating this step we can rewrite the Laplace operator as

$$\Delta_z \psi(z)|_{z=i_z \times a} = \Delta_z^2 \psi(z)\big|_{z=i_z \times a} \tag{54}$$

$$= \frac{\nabla_z \psi(z)|_{z=(i_z+\frac{1}{2})\times a} - \nabla_z \psi(z)|_{z=(i_z-\frac{1}{2})\times a}}{a} \tag{55}$$

$$= \frac{\psi_{i_z+1} - 2\psi_{i_z} + \psi_{i_z-1}}{a^2} \tag{56}$$

With these results, we can write the discretized Hamiltonian for a homogeneous system as a matrix:

$$H_0 = \begin{pmatrix} 2t & -t & 0 & 0 & 0 \\ -t & 2t & -t & 0 & 0 \\ 0 & -t & 2t & -t & 0 \\ 0 & 0 & -t & 2t & -t \\ 0 & 0 & 0 & -t & 2t \end{pmatrix} \tag{57}$$

where the hopping parameter t is defined as

$$t := \frac{\hbar^2}{2m^\star a^2}. \tag{58}$$

In general the hopping parameter can also depend on the spatial coordinate.

The infinite continuous energy domain is also represented as an equally spaced finite discrete lattice. Similar to the spatial domain, the energy grid is defined by the spacing between two energy points and minimal and maximal energy. The energy window has to be chosen large enough manually to fit the potential and the scattering. This is done by running the simulation and then adjusting the energy grid. The mentioned action could also be automated by using a mesh refinement or by using adaptive integration in the energy space. This feature needs to be tackled in future work.

The momentum space is also discretized by another equally spaced finite discrete lattice. The minimum of the momentum is always zero, thus the momentum grid is specified by the maximum momentum and a spacing factor x_k. The difference of two momentum values is then defined by

$$\mathrm{d}K = \frac{x_K \pi}{a} \tag{59}$$

where a is the spacing in the spatial domain. The equal spacing in the momentum domain corresponds to a quadratic spacing in the contribution to the Hamiltonian:

$$\epsilon_K = \frac{K^2 \hbar^2}{2m^\star}. \tag{60}$$

This choice is reasonable since the contribution for smaller momentum is higher and the contribution is decaying. The maximal K_{max} value has to be chosen carefully. A numerical experiment (see figure 21) has shown that the

maximal momentum value needs to be at least as large as the corresponding energy maximum:

$$\frac{K_{max}^2 \hbar^2}{2m} \geq E_{max} \tag{61}$$

The figure of the spectral density profile for each momentum distribution plotted against the energy is presented in chapter 5 (see figure 21).

3.1.2 *The explicit inversion*

Explicit inversion is usually avoided. Normally it is replaced by solving a linear system. In the NEGF framework this is not possible. The retarded Green's function is the fundamental quantity that is used for the calculation of several other quantities. Thus, it is required to explicitly compute the inverse of the matrix. The retarded Green's function fulfills the following properties: It is a complex symmetric matrix. To invert this matrix the factorization of this matrix is first calculated using the Bunch-Kaufman diagonal pivoting method. The form of this factorization is

$$A = L \times D \times L^T. \tag{62}$$

This function is implemented in LAPACK and it is called ZSYTRF [4]. This implementation is the blocked version of the algorithm and it is calling Level 3 BLAS functions. This factorization can then be used to compute the inverse. For this step the LAPACK function ZSYTRI is used. By the utilization of these two LAPACK functions the explicit inverse of the matrix is calculated.

3.1.3 *Momentum and Energy Integration*

The physical definitions stated in Chapter 2 contain integrations. E.g. the integration of the charge density:

$$n(z_i) = \int n(z_i, E) \mathrm{d}E. \tag{63}$$

This integration needs to be formulated on the discretized lattice. Different methods [28] exist for the integration on a equally spaced grid with $x_i = x_0 + ih$ where h denotes the spacing parameter. The easiest but also most inaccurate method is a simple Riemann sum

$$\int_{x_0}^{x_1} f(x) \mathrm{d}x = h f(x_0) + \mathcal{O}(h^2 f'). \tag{64}$$

More advanced methods are the Trapezoidal rule,

$$\int_{x_0}^{x_1} f(x) \mathrm{d}x = h \left[\frac{1}{2} f(x_0) + \frac{1}{2} f(x_1) \right] + \mathcal{O}(h^3 f'') \tag{65}$$

or the Simpson's rule,

$$\int_{x_0}^{x_2} f(x)\mathrm{d}x = h\left[\frac{1}{3}f(x_0) + \frac{4}{3}f(x_1) + \frac{1}{3}f(x_2)\right] + \mathcal{O}(h^5 f^{(4)}). \quad (66)$$

There are also higher order rules, the five-point formula is called Bode's rule. The even higher order rules are not named after famous persons.

During the development of the distributed implementation the simple Riemann sum has been used, mainly because of easier debugging. Due to the fact that the difference between the methods is only a change in the prefactors the others methods can also be used by a simple change in the source code. This is highly recommended whenever physical results should be calculated.

3.1.4 *The boundary conditions of the self-energy*

Within the integration of the self-energy we access the energy shifted by $\hbar\omega$:

$$\begin{aligned}\Sigma^{\gtrless}(k,E) =& \frac{e^2\hbar\omega_{LO}}{4\pi^2}\int_0^{q_{\max}} \mathrm{d}q\, qF(q,\Delta_{ij},k,q_0)\times \quad (67)\\ &\Big(N_{LO}\times G^{\gtrless}(q,E\pm\hbar\omega_{LO}) + (N_{LO}+1)G^{\gtrless}(q,E\mp\hbar\omega_{LO})\Big)\end{aligned}$$

However the energy grid that we are using is finite. That means that at the boundaries we would need to use Green's functions for energies that we haven't calculated. Therefore we chose the energy values that are the closest to the actual energy value:

$$E_{shift} = E \pm \hbar\omega_{LO} \quad (68)$$

$$E_{shift} = \begin{cases} 0, & \text{if } E \leq 0 \\ N_E - 1, & \text{if } E \geq N_E \\ E_{shift}, & \text{otherwise} \end{cases} \quad (69)$$

3.2 THE ALGORITHM

3.2.1 *Ballistic*

The NEGF framework is based on Green's functions. A Green's function is the operator that inverts the Hamilton operator. In the discretized description the Hamiltonian corresponds to a $Np \times Np$ matrix, where Np defines the number of points in the spatial dimension. For every momentum point the Hamiltonian varies due to a shift in energy. Thus the *Dyson equation* corresponds to an explicit inversion of a matrix.

For each bias point, the shift in the external potential, the potential is calculated by a self-consistent iteration. In the beginning the potential is initialized by a guess, e.g.

$$U[z] = 0 \quad \forall z. \tag{70}$$

This loop iterates until the potential is converged, e.g. the convergence criterion ϵ_U is reached. This criterion is compared to the infinity norm of the potential:

$$\|U\|_\infty < \epsilon_U. \tag{71}$$

The Green's function is calculated for every momentum and energy point. The boundary self-energies Σ_1 and Σ_2 and the depending scattering rate are initialized in the beginning of every energy and momentum step. The retarded Green's function can then be calculated by inverting the matrix:

$$\mathbf{G}^R = \left(E\mathbb{1} - \mathbf{H}(K) - \mathbf{\Sigma}^R\right)^{-1}. \tag{72}$$

The spatially momentum and energy resolved charge density can be calculated by using the exploiting of the Green's function, the scattering rate and the Fermi function. By integrating over the momentum and energy domain, we get the spatially dependent charge density:

$$\begin{aligned} n(z,k,E) = \mathrm{Re}(\mathrm{diag}(G^R \times\ & (f_1(k,E) \times \Gamma_1 \\ & + f_2(k,E) \times \Gamma_2) \times G^R)) \end{aligned} \tag{73}$$

$$n(k,z) = \int \mathrm{d}E\, n(z,k,E) \tag{74}$$

$$n(z) = \int \mathrm{d}k \frac{k}{2\pi} n(z,k) \tag{75}$$

With the charge density the *Poisson equation*

$$\partial_z\left(\epsilon(z)\partial_z\left(-\frac{U(z)}{e}\right)\right) = q(n(z) - N_d(z)) \tag{76}$$

can be solved which then defines the new potential. With the new potential the potential can be updated by using simple mixing (also known as Kerker mixing):

$$U^{i+1} = \lambda U^{new} + (1-\lambda)U^i. \tag{77}$$

The complete ballistic NEGF algorithm is summarized in algorithm 1.

3.2.2 *Recursive Green's Functions*

For the ballistic case we do not have to calculate all the matrix elements of the retarded Green's function. For the calculation of the charge density n only the diagonal and the first column of G^R are needed (due to symmetry also

Algorithm 1 NEGF algorithm for the ballistic case

procedure NEGF
  for BIAS V **do**
    guess U
    while $dU < \epsilon_U$ **do**
      for k **do**
        for E **do**
          $\Sigma_1(1,1) = -t_0 \exp\left(i \times \mathrm{acos}(1 - (E - U(1))/(2t_0))\right)$
          $\Sigma_2(n,n) = -t_0 \exp\left(i \times \mathrm{acos}(1 - (E - U(n))/(2t_0))\right)$
          $\Gamma_1 = i \times (\Sigma_1 - \Sigma_1^\dagger)$
          $\Gamma_2 = i \times (\Sigma_2 - \Sigma_2^\dagger)$
          $\Sigma^R = \Sigma_1 + \Sigma_2$
          $G^R(k,E) = \left[E\mathbb{1} - H(k) - \Sigma^R\right]^{-1}$
          $G^A(k,E) = (G^R)^\dagger$
          $A = \mathrm{diag}(i \times (G^R - G^A))$
          $n(z,k,E) = \mathrm{Re}(\mathrm{diag}(G^R \times (f_1(k,E) \times \Gamma_1 + f_2(k,E) \times \Gamma_2) \times G^R))$
          $n(k,z) = \int \mathrm{d}E\, n(z,k,E)$
        end for
        $n(z) = \int \mathrm{d}k \frac{k}{2\pi} n(z,k)$
      end for
      Solve Poisson: $\partial_z(\epsilon(z)\partial_z\phi(z)) = e(n(z) - N_d(z))$
      Update Potential: $U^{i+1} = \lambda U^{new} + (1-\lambda)U^i$
      Stop iteration if $\|U\|_\infty < \epsilon_U$ is fulfilled
    end while
  end for
end procedure

the right column is known). Where G denotes the exact Green's function the lower case g's are needed the left- and right-injected density of states g^l and g^r. We can thus recursively build the matrix [21]. In the first step the right-injected density of states g^r is calculated. We can start with the last element and can then compute the related elements. In the second step the diagonal can be calculated and the first column can be calculated. The equations are given by:

$$g^r_{N,N} = (E - H_{N,N} - \Sigma_{N,N})^{-1} \tag{78}$$

$$g^r_{n,n} = \left(E - H_{n,n} - H_{n,n} - H_{n,n+1} g^r_{n+1,n+1} H_{n+1,n}\right)^{-1} \qquad 2 \leq n \leq N \tag{79}$$

$$G_{1,1} = \left[E - H_{1,1} - \Sigma_{1,1} - H_{1,2} g^r_{2,2} H_{2,1}\right]^{-1} \tag{80}$$

$$G_{n,n} = g^r_{n,n} + g^R_{n,n} H_{n,n-1} G_{n-1,n-1} H_{n-1,n} g^r_{n,n} \qquad 2 \leq n \leq N \tag{81}$$

$$G_{n,1} = -g^R_{n,n} H_{n,n-1} G_{n-1,1} \qquad 2 \leq n \leq N \tag{82}$$

The whole process can be visualized by

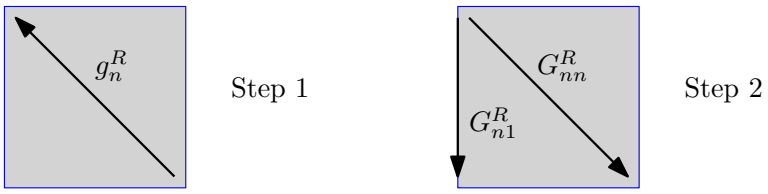


Figure 1: An illustration of the recursive Green's functions method.

By this approach the inversion of the matrix can be performed in $\mathcal{O}(n)$ instead of $\mathcal{O}(n^3)$, where n is the number of matrix elements in one dimension, which is equivalent to the number of spatial points Np.
The charge density can directly be calculated similar as in 3.2.1.

3.2.3 *Inelastic*

In inelastic case we also have electron-phonon scattering. This case is different from the ballistic case. Here another self-consistent loop is inside the potential loop. The self-energy also has to be solved self-consistently. Thus we have two nested self-consistent loops. The outer loop is the potential loop. For each potential step the self-energy is solved self-consistently. We start similar to the ballistic case by setting up the boundary self-energies Σ_1 and Σ_2 and the corresponding scattering rate. Again the retarded Green's function is solved by a matrix inversion. Then the lesser Green's function can be computed via the *Keldysh equation* whereas the greater Green function can be calculated by the completeness relation. The charge density can be computed by the relation given in section 2.6, similar to the ballistic case.

Also the integration of the momentum energy-resolved spatial charge density is performed in the same manner. Then the electron-phonon self energy has to be computed. This function depends for one energy and momentum point on all other Green's functions in momentum space and on two other energy points due to the shift $\pm\hbar\omega_{LO}$. Thus this equation is coupled to the other Green's functions. The electron-phonon self-energy is defined by an integration over all momentum points. Once the self-energy is calculated the infinity norm of the difference between the old and the new self-energy is computed. When the result is smaller than the convergence criterion the self-energy is considered to be self-consistent and the iteration is stopped. Finally the Possion equation is solved and the potential is updated, this part is equivalent to the ballistic computation.

The algorithm is visualized in algorithm 2 (see page 23).

3.2.4 *Flowchart*

The flowchart of the complete NEGF simulation is visualized in figure 2.

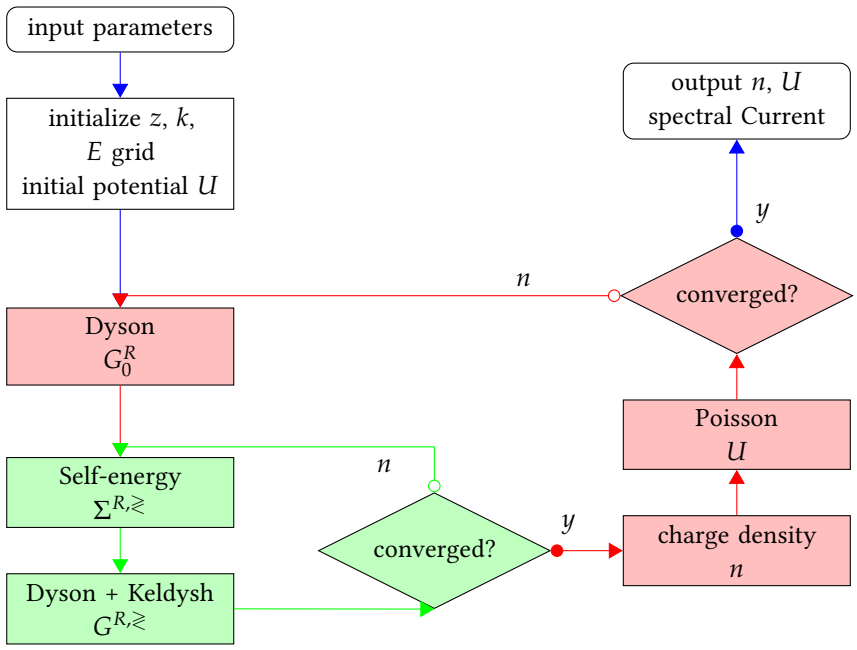


Figure 2: Flowchart for the NEGF simulation. The inner self-consistency loop, visualized in green, connects the Geen's functions and the self energies, while the outer loop, visualized in red, provides the update of the potential from the solution of the Possion equation.

Algorithm 2 NEGF algorithm for the Inelastic case

procedure NEGF
  for BIAS V **do**
    guess U
    while $dU < \epsilon_U$ **do**
      while $\mathrm{d}\Sigma < \epsilon_\Sigma$ **do**
        for k **do**
          for E **do**
            $\Sigma_1(1,1) = -t_0 \exp\left(i \times \mathrm{acos}(1 - (E - U(1))/(2t_0))\right)$
            $\Sigma_2(n,n) = -t_0 \exp\left(i \times \mathrm{acos}(1 - (E - U(n))/(2t_0))\right)$
            $\Gamma_1 = i \times (\Sigma_1 - \Sigma_1^\dagger)$
            $\Gamma_2 = i \times (\Sigma_2 - \Sigma_2^\dagger)$
            $\Sigma_{in,1} = f_1(k,E) \times \mathrm{diag}(\Gamma_1)$
            $\Sigma_{in,2} = f_2(k,E) \times \mathrm{diag}(\Gamma_2)$
            $\Sigma_{out,1} = (1 - f_1(k,E)) \times \mathrm{diag}(\Gamma_1)$
            $\Sigma_{out,2} = (1 - f_2(k,E)) \times \mathrm{diag}(\Gamma_2)$
            $\Gamma_p = \Sigma_{in} + \Sigma_{out} = \Sigma_{in,1} + \Sigma_{in,2} + \Sigma_{out,1} + \Sigma_{out,2}$
            $\Sigma^R = \Sigma_1 + \Sigma_2 + \Sigma_{\mathrm{phonon}}(k,E) + i/2 \times \mathrm{diag}(\Gamma_p)$
            $G^R(k,E) = \left[E\mathbb{1} - H(k) - \Sigma^R\right]^{-1}$
            $G^A(k,E) = (G^R)^\dagger$
            $G^<(k,E) = G^R(\Sigma_B^< + \Sigma_{\mathrm{ph}}^<) \times G^A$
            $G^>(k,E) = G^R - G^A - G^>$
            $A = \mathrm{diag}(i \times (G^R - G^A))$
            $n(z,k,E) = \mathrm{Re}(\mathrm{diag}(G^R \times (f_1(k,E) \times \Gamma_1 + f_2(k,E) \times \Gamma_2) \times G^R))$
          end for
          $n(k,z) = \int \mathrm{d}E\, n(z,k,E)$
        end for
        $n(z) = \int \mathrm{d}k \frac{k}{2\pi} n(z,k)$
        for k **do**
          for E **do**
            $\Sigma^{>}{}^{<}(k,E) = \frac{e^2 \hbar \omega_{LO}}{4\pi^2} \int_0^{q_{\max}} \mathrm{d}q\, qF(q,\Delta_{ij},k,q_0) \times \left[N_{LO} \times G^{>/<}(q, E \pm \hbar\omega_{LO} + (N_{LO}+1)G^{>/<}(q, E \mp \hbar\omega_{LO})\right]$
          end for
        end for
        Stop iteration if $\|\Sigma_{el\text{-}ph}^{i+1} - \Sigma_{el\text{-}ph}^{i}\|_\infty < \epsilon_\Sigma$
      end while
      Solve Poisson: $\partial_z(\epsilon(z)\partial_z\phi(z)) = e(n(z) - N_d(z))$
      Update Potential: $U^{i+1} = \lambda U^{new} + (1-\lambda)U^i$
      Stop iteration if $\|U\|_\infty < \epsilon_U$ is fulfilled
      Update Potential
    end while
  end for
end procedure

4

IMPLEMENTATION

This chapter presents the implementation and the parallelization. A `Matlab` reference implementation has been realized to test the algorithm and perform computations for comparison. For realistic simulations this implementation is particularly slow, due to the limitations in `Matlab`. The simulation was translated into C and linked to efficient BLAS and LAPACK calls and self-developed kernels for those operations where no efficient implementation existed. For testing issues this version was parallelized with `OpenMP`. The data distribution leads to a distributed implementation employing Message Passing Interface (`MPI`). In the last step the distributed version was parallelized on node-level side with `OpenMP` which yields to an efficient hybrid implementation for the NEGF framework.

4.1 THE MATLAB VERSION

In a first step a a Matlab reference version was implemented. The purpose of this version was to understand and analyze the algorithm. However this implementation was particular slow, since the NEGF algorithm contains huge nested loops and the way Matlab works does not favour loops in general. Especially the inelastic non-equilibrium case is the most compute intensive computation in the work. The Matlab version took nearly a whole day to compute this case on a very performed workstation. Therefore we take a closer look on the parallelization approach in the following sections.

4.2 LEVELS OF PARALLELISM

In this part the levels of parallelism within the non-equilibrium Green's function algorithm are discussed. Four different levels can be identified. The outermost level is the distribution of the bias points. This level is especially easy, because no subpart depends on the outer calculation, i.e. the computations at different bias voltages are completely independent. The parallelization of this level can therefore be done easily, e.g. by starting multiple jobs on a supercomputer or by one job with multiple instances.

The next level is the momentum level. This level is not trivial because integration over the k-points, which is required for the evaluation of physical quantities such as the charge density and of the self-energies for inelastic scattering, does depend on all other k-values, making communication necessary. The computation of the Green's functions, on the other hand, is completely independed of the components at other k-values.

The third level is the energy level. The dependencies between the different energies has its origin in the integration over the energy domain for the evaluation of physical quantities as well as in the energy shift in the self-energy computation.

The fourth and last level is the spatial domain decomposition. The data for each momentum and energy point is a $Np \times Np$ matrix. The operation on this level can be parallelized by using multithreaded BLAS or LAPACK functions or by employing OpenMP directives.

The levels of parallelism are visualized in figure 3.

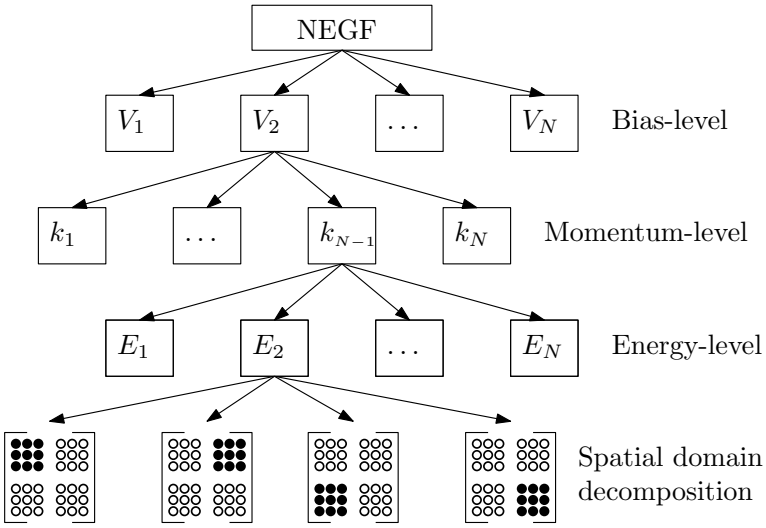


Figure 3: The four levels of parallelism that can be applied to the NEGF formalism are shown. These levels are i) bias points, ii) momentum points, iii) energy points and iv) the spatial domain decomposition. Illustration adapted from [25].

4.3 PARALLELIZATION WITH OPENMP

In the parallelization with OpenMP the momentum points are distributed across all available threads. The calculation of the Green's functions has no dependencies, thus it can be parallelized with a simple OpenMP statement:

```
#pragma omp parallel for
for( iK=0; iK<NK; iK++ )
{
  ...
}
```

The number of momentum points N_K is split up between all thread, such that each thread is working on $N_{K,local}$. The parallelization works optimally when the number of momentum points is an integer multiple of the number of threads. The parallelization with OpenMP is shown in figure 4.

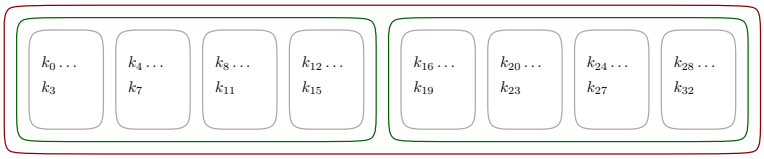


Figure 4: The distribution of 32 momentum points is shown on one node (red line) with 2 sockets (green lines) and 4 threads (grey lines) each. Each thread is working on $N_{K,local} = 4$ local k-points. In this scenario the distribution is optimal, since all momentum points can be distributed evenly.

4.4 PARALLELIZATION WITH MPI

The NEGF simulation can be so computationally intensive that the calculation on one node takes a too long or the NEGF program requires more memory than what is available on one single node. In both cases, simulation software is needed which can utilize more than one node. This requires distributed memory programming. The most convenient solution for this is called Message Passing Interface (MPI). This is a standard that describes how to exchange messages with other computers over Ethernet or Infiniband.

This explicit communication requires that the programmer explicitly defines the send and receive statements whenever data is needed which is not stored locally. This additional communication can be seen as overhead compared to the serial/threaded version. This overhead should be avoided as much as possible because this is a drawback regarding the scaling of the distributed implementation on multiple nodes. The obvious solution would be to avoid any unnecessary communication, e.g. by reusing the received data. The other approach is to overlap computation and communication, in such a way the additional cost of communication can be hidden. However this is only possible when the computation is compute bound. The CPU and the network card are sharing the same memory bandwidth. So when the computation is memory-bound and the full memory bandwidth is in use, the overlap wouldn't work that well.

4.4.1 *Distribution of the momentum and energy points*

The NEGF architecture needs to be mapped to the architecture of a modern supercomputer. Most of the current high end supercomputers consist of several nodes connected with a fast network. The network is most likely arranged in a Fat tree or torus topology. These architectures prefer communication between nodes that are close to each other, such that the hop count gets minimized.

From the levels of parallelism presented in section 4.2 the momentum and energy levels are parallelized with MPI. These levels form a hierarchy in which each node is performing the calculation of a subset of energy and

a subset of momentum points. Integration over the energy or momentum domain will require collective MPI communication.

The MPI standard provides a function to map a topology inside the program to the node topology. These are called virtual topologies. One type of virtual topologies are the Cartesian topologies. Each process is connected to its neighbors in a virtual grid. The boundaries can be cyclic, which specifies whether the ends are connected or not. Each process can be identified by Cartesian coordinates. A reorder option allows the MPI implementation to optimize the mapping from the virtual to the physical topology for better performance.

In the NEGF framework we are using a 2D Cartesian communicator. The momentum points are distributed along the first dimension on the 2D grid. The energy points are distributed along the second dimension. This distribution is visualized in figure 5.

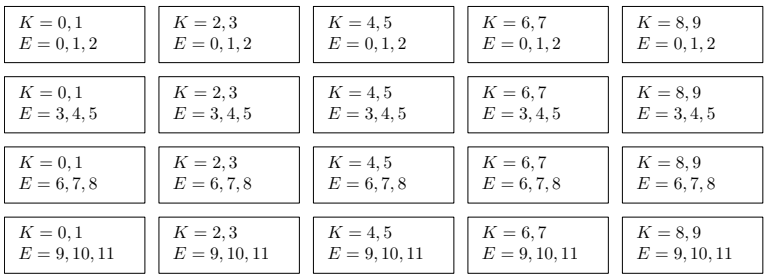


Figure 5: Exemplary momentum and energy distribution of $N_K = 10$ and $N_E = 12$ on 20 nodes (black squares). In this example every node stores $N_{K,local} = 2$ local momentum data points and $N_{E,local} = 3$ local energy points.

Each node gets a subset of the number of momentum points N_K and the number of energy points N_E. For illustrative purpose let us have a look at an examples:

$$N_K = 32 \quad N_E = 128 \tag{83}$$

$$N_{nodes} = 32 \tag{84}$$

$$\Rightarrow D_0 = 8 \quad D_1 = 4 \tag{85}$$

$$\Rightarrow N_{K,local} = 4 \quad N_{E,local} = 32 \tag{86}$$

The creation of the dimension D_0 and D_1 is done by the MPI library with the function `MPI_Dims_create`. This function helps the user to select a balanced distribution. For the 2-dimensional case one dimension can be provided by the user. Obviously the global number of points needs to be integer dividable by the dimension size to ensure evenly distributed workload. A check was implemented to find a balanced distribution as well as fulfill the mentioned constrain.

When the distribution of the dimension is determined, the communicator for the MPI communication can be created. A 2D Cartesian Communicator

is created without periodicity neither in momentum nor in energy direction with `MPI_Cart_create`. Later a similar communicator with periodicity in k-dimensions is created with the same assignment of corresponding Cartesian coordinates. This is needed for the self-energy integration. The reorder option is enabled to optimize the mapping.

Each rank stores a subset of momentum $N_{K,local}$ and energy points $N_{E,local}$. The rank has a local view $i_E \in [0, N_{E,local}]$ and $i_K \in [0, N_{K,local}]$. The global momentum/energy value can be calculated with the help of the MPI library. As mentioned above each rank in the Cartesian topology can be identified by Cartesian coordinates with the function `MPI_Cart_coords`. The global momentum/energy values can then be determined via

$$i_{K,global} = i_{K,local} + C_0 * N_{K,local} \tag{87}$$

$$i_{E,global} = i_{E,local} + C_1 * N_{E,local} \tag{88}$$

where $C_{0/1}$ denotes the Cartesian coordinate of the rank in dimension 0 or 1, respectively.

In some cases only horizontal (across the momentum points) or vertical (across the energy points) communication is needed, e.g. for the integration of the charge density. If we consider the integration to be two steps, we have one step for momentum integration $n(z, E) = \int \mathrm{d}k\, n(z, k, E)$ and thus only horizontal communication. The second step in obtaining the charge carrier density, we integrate over the energy domain: $n(z) = \int \mathrm{d}E\, n(z, E)$, here only vertical communication appears. For one energy point the momentum points are forming a subset of the whole set. Within the MPI framework we can divide the Cartesian Communicator into communicators which contain only a subset of nodes. This can be done with the `MPI_Cart_sub` function. With the `remain` argument can be specified which dimension should remain in the derived communicator. In the NEGF implementation we have two subcommunicators, one for the momentum and one for the energy. Each MPI rank has this two additional communicators, but they differ on each MPI rank. A illustration is shown in figure 6.

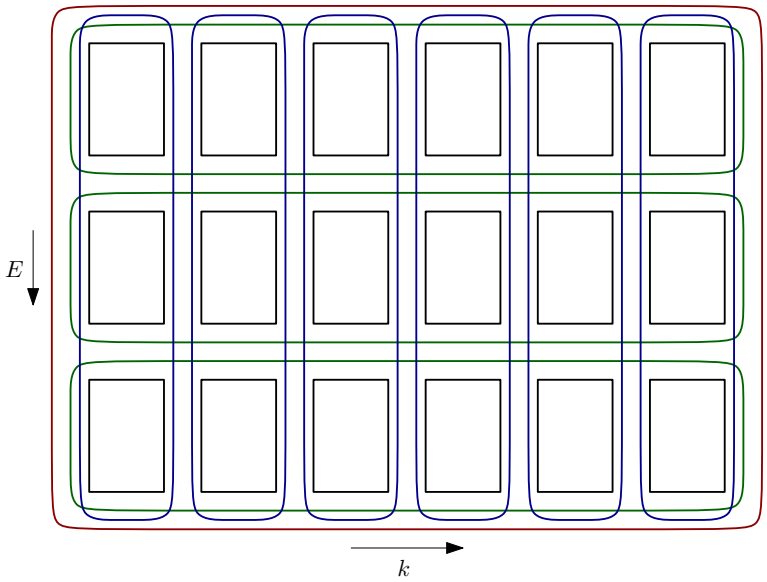


Figure 6: The 2D Cartesian topology. 18 nodes are shown, illustrated by the black squares. The 2-dimensional communicator is shown in red. The communicators that contain all k-points for a certain energy point are visualized in green. The communicator for all E-points for a specific momentum point is shown in blue. These three different communicators are used in the program.

4.4.2 *Distributed self-energy integration*

The polar optical phonon self-energy is defined by:

$$\Sigma_{ij}^{\gtrless}(k,E) = \frac{e^2\hbar\omega_{LO}}{4\pi^2}\sum_{0}^{q_{\max}}\Delta q\, qF(q,\Delta_{ij},k,q_0)\times \left(N_{LO}\times G_{ij}^{\gtrless}(q,E\pm\hbar\omega_{LO}) + (N_{LO}+1)G_{ij}^{\gtrless}(q,E\mp\hbar\omega_{LO})\right) \tag{89}$$

where we have written the integration as a Riemann summation where Δq denotes the spacing in the momentum grid for illustration.

Distributing the calculation induces the problem that the data that are needed for the computation are not stored locally. This implies that communication is needed in order to perform the calculation. In this case the computation of $\Sigma^{\gtrless}(k_i,E_j)$ for one specific momentum points k_i and one specific energy point E_j depends on the Green's functions for all other momentum points and two other energy points:

$$\Sigma^{\gtrless}(k_i,E_j) = f\Big(G^{\gtrless}(k_0,E_j\pm\hbar\omega), G^{\gtrless}(k_1,E_j\pm\hbar\omega),\ldots, G^{\gtrless}(k_{N_K-1},E_j\pm\hbar\omega)\Big) \tag{90}$$

Some of these values are local but most of them are not local.

The computation for the self-energy is done in two steps: First the communication for the energy is performed, then the communication for the momentum is executed.

For the communication it is obviously required to know which ranks communicate with each other. Whenever the needed energy points are outside the local view, i.e. $i_{E.local} \pm \hbar\omega \notin [0, N_{E,local}]$ data has to be communicated. The node difference n_{diff}, which is defined by the Cartesian difference of the two nodes that need to exchange data, can be calculated by the given values. The corresponding rank number can be obtained by the function `MPI_Cart_shift` (see [12]). This function returns a source and destination for the shift. With the help of these results it can be determined whether the actual rank needs to send and/or receive data. The data is sent via an nonblocking sending command `MPI_Isend`. This allows to overlap communication and computation. The receive is also done nonblocking. This allows to perform the local update simultaneously to the communication.

For each momentum and energy point the local energy picture is created:

$$H^{\gtrless}(q, E) = N_{LO} \times G^{\gtrless}(q, E \pm \hbar\omega_{LO}) + (N_{LO} + 1)G^{\gtrless}(q, E \mp \hbar\omega_{LO}). \tag{91}$$

This value is needed on all other nodes that take care of the same energy point but a different momentum point. Or to speak in the MPI picture: this value $H(q, E)$ needs to be communicated among the communicator in the momentum direction (cf. fig 6). For each different momentum value H contributes to the self-energy with a different prefactor:

$$\Sigma_{ij}^{\gtrless}(k, E) = \frac{e^2\hbar\omega_{LO}}{4\pi^2} \sum_{0}^{q_{max}} \Delta q \, qF(q, \Delta_{ij}, k, q_0) \times H_{ij}^{\gtrless}(q, E). \tag{92}$$

For the communication across the momentum dimension, every node in this dimension has to communicate with all the other nodes. This is realized by a loop over all nodes in this dimension. The sender and receiver are determined with `MPI_Cart_shift`. With this function we get a `source` and `destination` for the communication. Each node sends the matrix to the destination and receives the matrix from source. Since the global value of K and Q has to be known in the prefactor in equation 92, we have to calculate them. With the `source` value we can get the Cartesian coordinate with the function `MPI_Cart_coords`. We can then calculate both global values via

$$i_{Q,global} = i_{Q,local} + C_{0,H} * N_{K,local} \tag{93}$$

$$i_{K,2,global} = i_{K,local} + C_0 * N_{K,local} \tag{94}$$

where $C_{0,H}$ is the first coordinate of the source from the communication and C_0 is the local first coordinate of the local point. Then the local update of the self-energy with the global values is performed.

The integration is schematically depicted in figure 7.

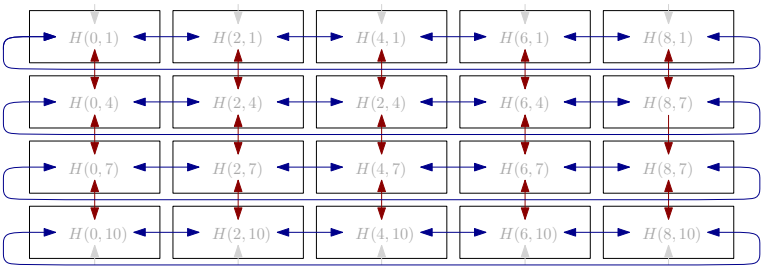


Figure 7: A snapshot of the self-energy integration. The illustration shows the global indices. The same distribution as in figure 5 is assumed. Each node is at the local step $K_{local} = 0$ $E_{local} = 1$. In the first step communication across the energy dimension (shown in red) is performed to build the local view of $H(K, E)$. Due to the boundary condition there is no communication at the border. In this case the local values are taken, this is visualized with the gray arrows. After the local view $H(K, E)$ is built, this matrix is communicated to all other nodes in the momentum level. This all-to-all like communication is visualized in blue. This snapshot of the communication is the same for every local energy and momentum point.

The pseudo code for the self-energy integration is shown in algorithm 3.

Algorithm 3 Distributed Integration of the self-energy

procedure INTEGRATION-SELF-ENERGY
  for iK=0; iK< NKloc; iK++ **do**
    for iE=0; iE<NEloc; iE++ **do**
      Compute node ndiff difference for communication
      Get source and destination
      Communicate for $G(iK, iE + \hbar\omega)$ and $G(iK, iE - \hbar\omega)$
      $H(iK, iE) = \left[N_{LO} \times G(iK, iE - \hbar\omega) + (N_{LO} + 1) \times G(iK, iE + \hbar\omega)\right]$
      Update local:
      for iQ=0; iQ< NKloc; iQ++ **do**
        $\Sigma(iK, iE) += \alpha F(iQ, \Delta_{z,z'}, iK, q_0) H(iK, iE)$
      end for
      for k Dimension **do**
        Send H and receive remote H
        Update with remote data:
        for iQ=0; iQ< NKloc; iQ++ **do**
          $\Sigma(iK, iE) += \alpha F(iQ, \Delta_{z,z'}, iK, q_0) H(iK, iE)$
        end for
      end for
    end for
  end for
end procedure

4.5 HYBRID PARALLELIZATION WITH MPI AND OPENMP

Previously we have shown how we can parallelize the NEGF algorithms in two different ways: a shared memory approach with OpenMP which worked perfectly on one node was presented. The second approach is a distributed memory implementation that is using MPI for explicit communication between ranks. This explicit communication constitutes an overhead within the execution. The generic goal in the development of efficient programs is to avoid overhead as much as possible, since it decelerates the runtime advantage from running the program on multiple resources, i.e. multiple nodes. A state-of-the-art supercomputer consists usually of different levels: On the node level we have one or multiple sockets that can communicate with a very fast interconnect and basically share the same address space. Thus the threaded approach works excellent on one full node, assuming that we have enough workload to employ all available tasks. This threaded approach implies indirect communication, since the different threads can read and write to shared memory areas. On this level we would like to avoid explicit communication. However, when we want to utilize more than one node we necessarily need explicit communication. This findings lead naturally to the intention to combine the strengths of both approaches: a combination of explicit and implicit communication, a combination of distributed and shared memory, or in other words a combination of MPI and OpenMP.

5 RESULTS

The hardware and software will be described in the beginning of this chapter. I have used the RWTH Compute cluster, the JURECA and JUQUEEN. Then I will present timing and scaling results for different machines used. Subsequently I will present the physical results of the NIN diode for the ballistic case and also for the inelastic case.

5.1 EXPERIMENTAL ENVIRONMENT

Whenever we are talking about timing, speedup or performance it is essential to specify the used hardware. This is important to make the results comparable and to allow reproducibility.

5.1.1 *Hardware*

An important part of the environment description is the used hardware. In the following we discuss technical details of the different machines.

5.1.1.1 *RWTH Computer Cluster*

The RWTH Computer Cluster is an inhomogeneous cluster with a lot of different hardware. I have only used the part with the Intel Haswell processors. The detailed hardware specifications are listed in table 2. An illustration of the configuration is shown in figure 8. The theoretical per node peak performance is given by:

$$\begin{aligned} \textit{node performance in GFlops} &= \textit{cores} \times \textit{freq} \times \textit{flops/cycle} && (95)\\ &= 24 \times 2.5\,\mathrm{GHz} \times 16\,\frac{\mathrm{flops}}{\mathrm{cycle}} && (96)\\ &= 960\,\mathrm{GFlops}. && (97) \end{aligned}$$

During the development of the distributed implementation a performance problem was detected on these machines. Randomly the computers show only half the expected performance in computer bound and memory bound problems. This problem has not been fixed yet. Therefore no performance results are presented for these machines.

5.1.1.2 *JURECA*

JURECA is quite similar to the AICES Haswell nodes. It is equipped with the same processor and also has 960 Gflops theoretical peak performance per

property	value
CPU	2x Haswell E5-2680 v3
RAM	64 GB
CPU frequency	2,500 MHz
RAM frequency	2,133 MHz
number of cores	2
number of cores	6
number of threads	12
flops/cycle	16
Level 1 cache size	12 x 64 KB Instruction
	12 x 64 KB Data
Level 2 cache size	12 x 256 KB
Level 3 cache size	30 MB

Table 2: Specifications of the AICES Haswell nodes in the RWTH Computer Cluster.

Figure 8: A graphical representation of the hardware specifications of the AICES Haswell nodes.

node. The standard node has 128 GB, but there are also a few nodes with up to 1,024 GB. The important difference is the faster network regarding bandwidth and latency. An EDR InfiniBand is used in a non-blocking fat-tree topology. A tabulated summary of the specifications of JURECA is presented in tabular 3. A graphical illustration of the node configuration is shown in figure 9.

5.1.1.3 *JUQUEEN*

The third machine used is JUQUEEN. It is one of the fastest supercomputer in the world with a total theoretical peak performance of 5.9 Pflops [33]. The per node performance is given by:

$$\begin{aligned} \textit{node performance in GFlops} &= \textit{cores} \times \textit{freq} \times \textit{flops/cycle} && (98)\\ &= 16 \times 1.6\,\text{GHz} \times 8\,\frac{\text{flops}}{\text{cycle}} && (99)\\ &= 204.8\,\text{GFlops} && (100) \end{aligned}$$

The big benefit of JUQUEEN is network: the 5D-torus topology.

property	value
CPU	2x Haswell E5-2680 v3
RAM	128 GB
CPU frequency	2,500 MHz
RAM frequency	2,133 MHz
number of cores	2
number of cores	6
number of threads	12
flops/cycle	16
Level 1 cache size	12 x 64 KB Instruction
	12 x 64 KB Data
Level 2 cache size	12 x 256 KB
Level 3 cache size	30 MB
Interconnect	InfiniBand EDR
Network topology	non-blocking fat tree

Table 3: Specifications of the JURECA.

Figure 9: A graphical representation of the hardware specifications of the JURECA nodes.

Figure 10: A graphical representation of the hardware specifications of the JUQUEEN nodes.

property	value
CPU	IBM PowerPC A2
RAM	16 GB
CPU frequency	1,600 MHz
RAM frequency	1,333 MHz
number of cores	16
number of threads	64
flops/cycle	8
Level 1 cache size	16 x 16 KB Instruction
	16 x 16 KB Data
Level 2 cache size	32 MB
Interconnect	custom
Network topology	5D Torus

Table 4: Specifications of JUQUEEN.

5.1.2 *Software*

The NEGF simulation software is written in C. Since the different architectures do not allow to use the same compilers and libraries, the used software is shortly summarized: On Intel architecture the Intel C Compiler is used, on the Blue Gene Q architecture the IBM XL compiler is used. The optimized Blas and Lapack library for each architecture was used. On Intel this is the Intel MKL library. On the Blue Gene Q the ESSL library is used.

5.2 PARALLELIZATION COMPARISON

5.2.1 *OpenMP scaling*

As described in section 4.3 the NEGF software was parallelized with OpenMP. A scaling plot is shown in figure 11. In this scenario the number of momentum points N_K can be evenly divided by the number of used cores. This is the best case scenario, because the work distribution is most even. In this case we reach a speedup of 19.70 on 24 cores without Hyperthreading. With Hyperthreading we get better timings (see table 5), but only a speedup of 16.29 on 24 cores.

cores	time without HP	time with HP
1	10,274.19 s	7,612.34 s
3	3,442.18 s	2,653.47 s
6	1,702.69 s	1,309.87 s
12	985.77 s	801.32 s
24	521.54 s	467.21 s

Table 5: Runtimes on JURECA for the OpenMP implementation without and with Hyperthreading. The system size is $N_k = 48$, $N_E = 301$, and $N_p = 100$.

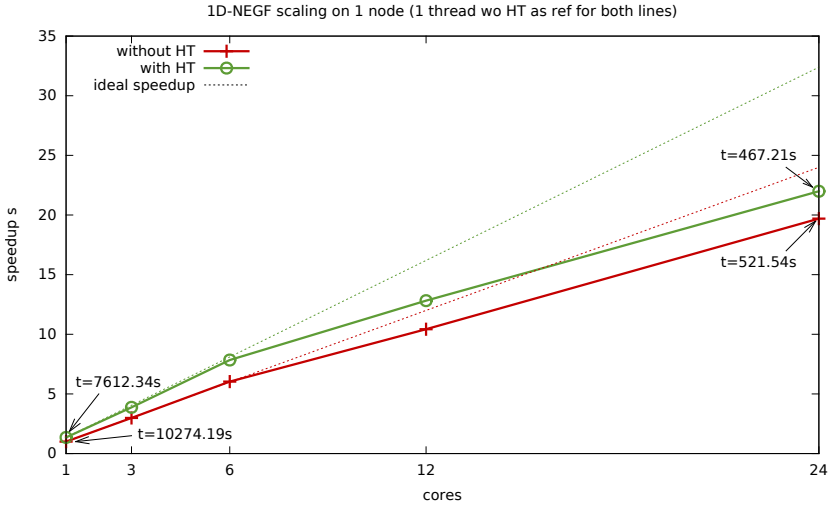


Figure 11: Speedup plot for the OpenMP implementation on one Jureca node. The system size is defined by $N_K = 48$, $N_E = 301$, and $N_p = 100$. Results without Hyperthreading are shown in red, whereas green shows the results with Hyperthreading enabled. The ideal speedup for both situations is visualized by the dotted lines. Both lines are normalized to the execution using 1 cores without Hyperthreading. Representative timings are given.

The worst case would be when then number of k-points can not evenly distributed across all cores. E.g., to distributed $N_K = 49$ momentum points on a 24-cores machine. This experiment was also performed and presented in figure 12. We can see that the runtime on 1-core increases only slightly, whereas the runtime on all 24-cores is much higher compared to the previous case. Thus the speedup on 24-cores decreased to 14.34 on 24 cores without Hyperthreading and 12.20 with Hyperthreading.

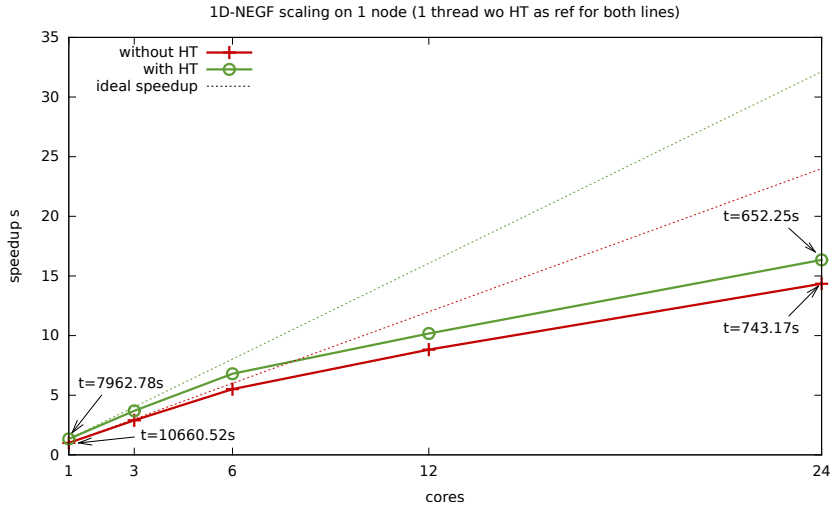
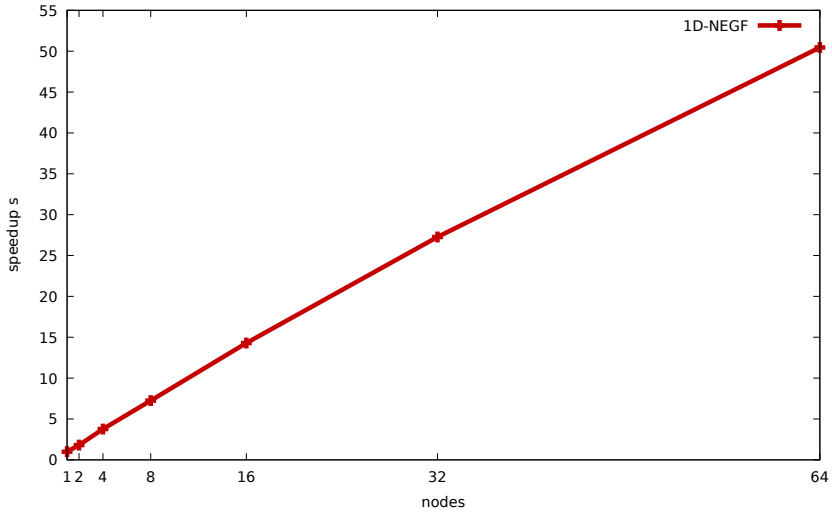


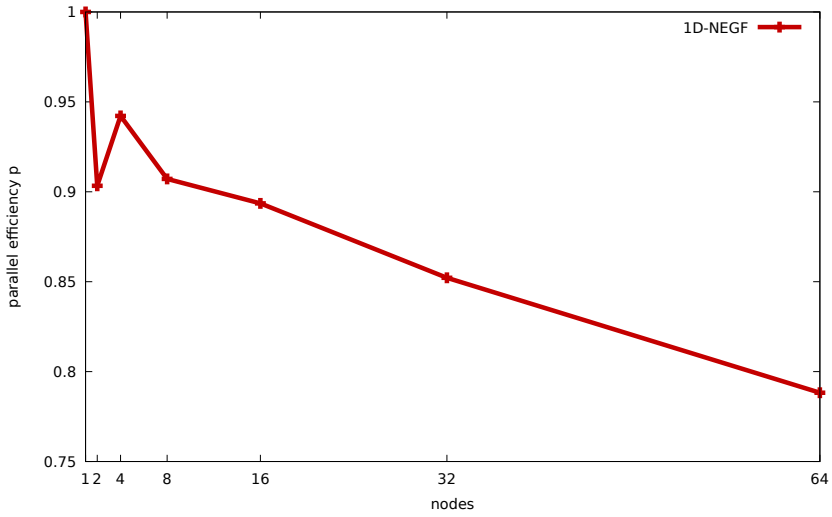
Figure 12: Similar setup as shown in figure 11, but this time with $N_K = 49$. Thus the number of momentum points can not be evenly distributed on 24 cores. This corresponds to the worst case scenario for the OpenMP implementation.

5.2.2 *MPI scaling*

In the NEGF simulation the most computationally intensive part is the calculation and integration of the inelastic self-energy. This part is also the complicated part regarding the distributed implementation, as described in section 4.4. Therefore this part is measured separately. The speedup of the MPI implementation is shown in figure 13.



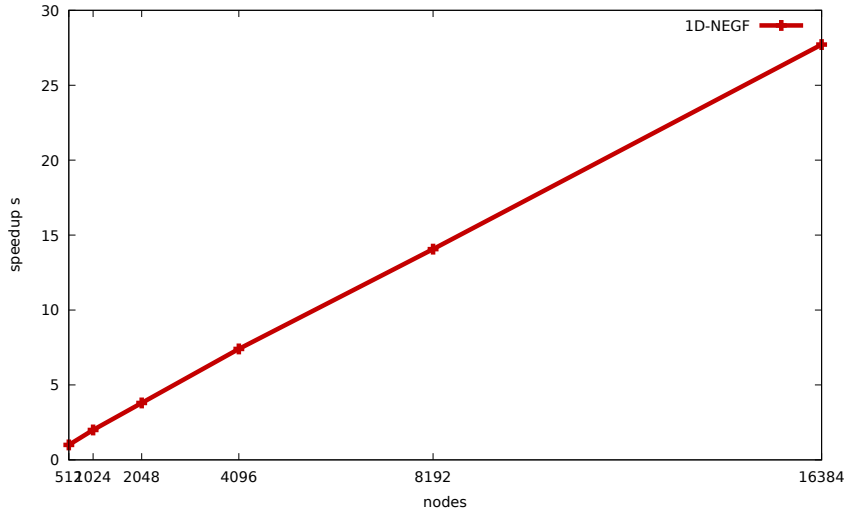
(a) Speedup



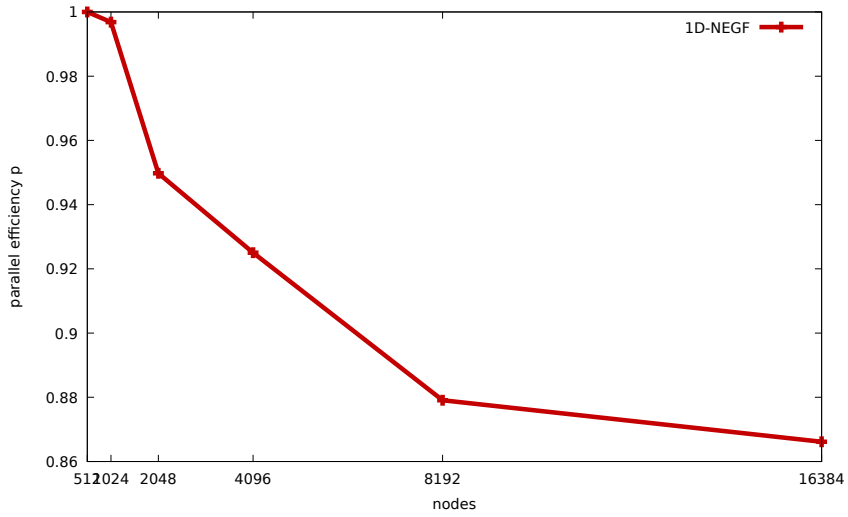
(b) efficiency

Figure 13: Scaling of the self-energy integration on JURECA. In this case only one thread per node was used to visualize only the MPI characteristics. The system sizes are $N_K = 256$, $N_E = 256$, and $N_P = 100$.

The same scaling experiment was performed on JUQUEEN with a bigger input problem on more nodes, see figure 14. The self-energy integration is also scaling on more cores.



(a) Speedup

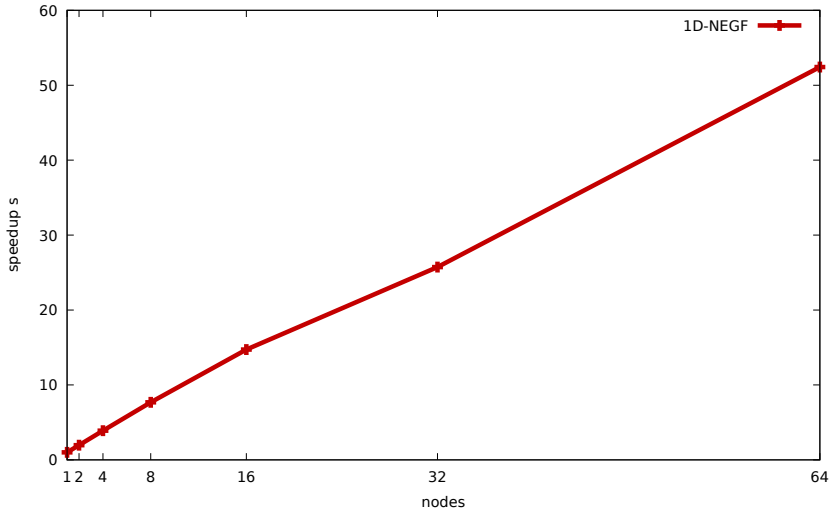


(b) efficiency

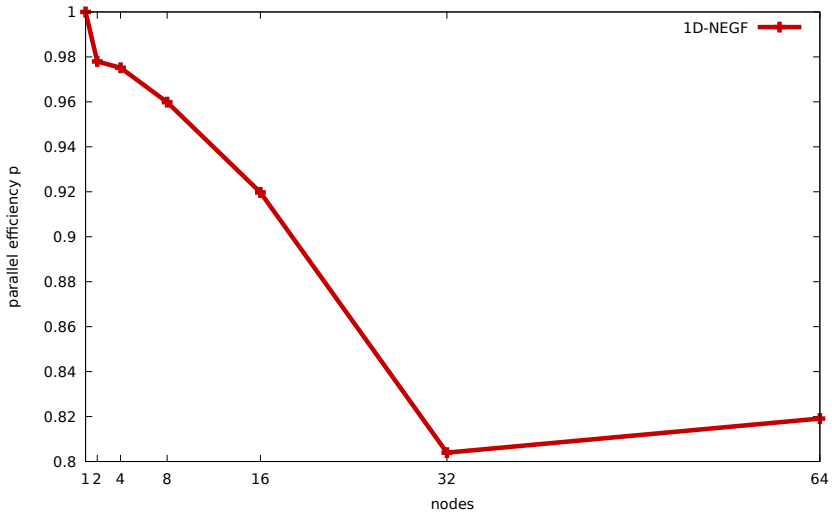
Figure 14: Scaling of the self-energy integration on JUQUEEN. In this case also only one thread per node was used to visualize only the MPI characteristics. The system sizes are $N_K = 1024$, $N_E = 1024$, and $N_P = 100$.

5.2.3 *Hybrid scaling*

The hybrid implementation as described in section 4.5 combines the MPI and the OpenMP parallelization. In the scaling experiment one MPI rank per node was used. On the node level OpenMP was used to exploit parallelism.



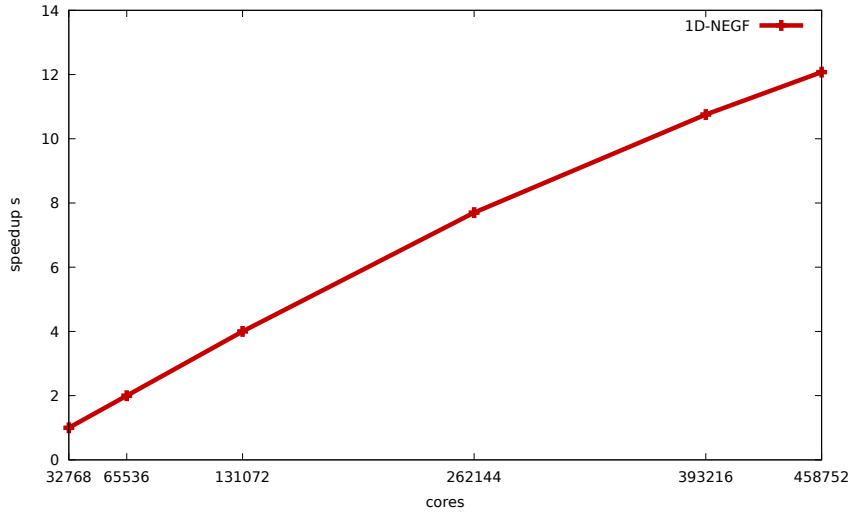
(a) Speedup



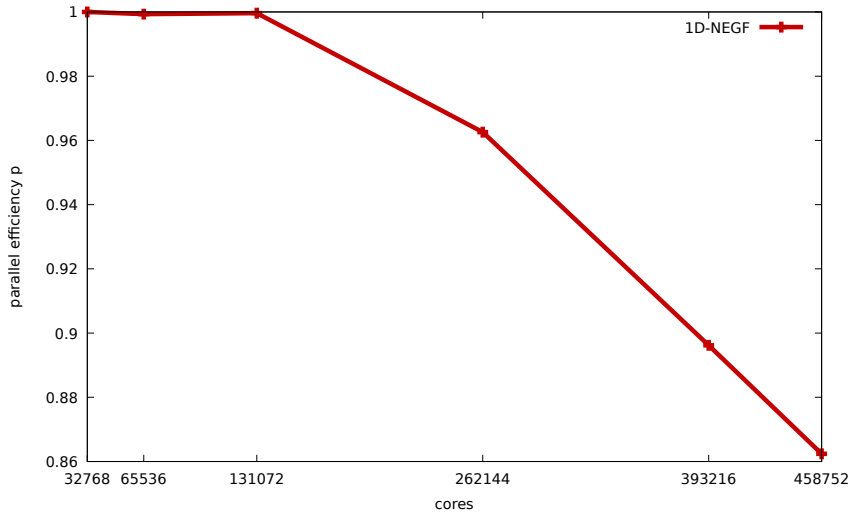
(b) efficiency

Figure 15: Hybrid scaling of the self-energy integration on JURECA. The system sizes are $N_K = 256$, $N_E = 256$, and $N_P = 100$.

Figure 16 shows the speedup plot for this hybrid implementation on JUQUEEN. We can see that the self-energy intergration is scaling on the full JUQUEEN installation, that means on all 28,672 nodes, which equals 458,752 cores. For the experiment runs 2-way SMT was used, thus 32 threads have been used on one node. The parallel efficiency on all 28,672 nodes is 0.86 related to 2,048 nodes. This is a very good result.



(a) Speedup

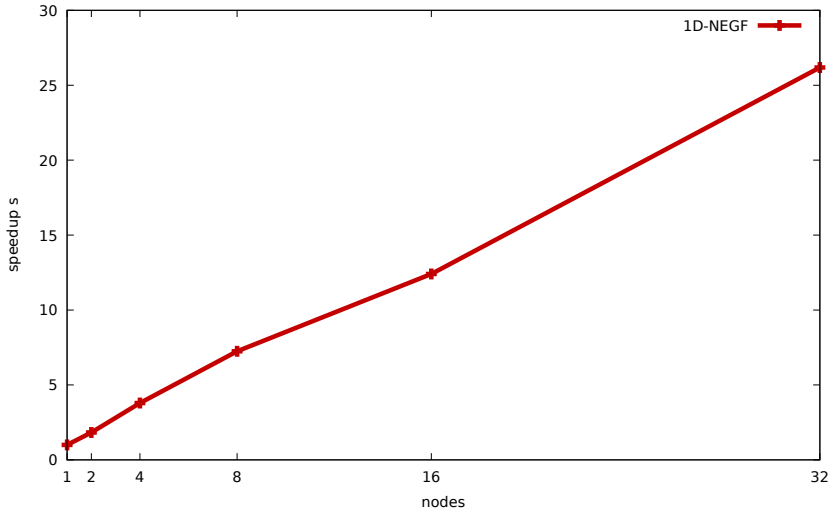


(b) efficiency

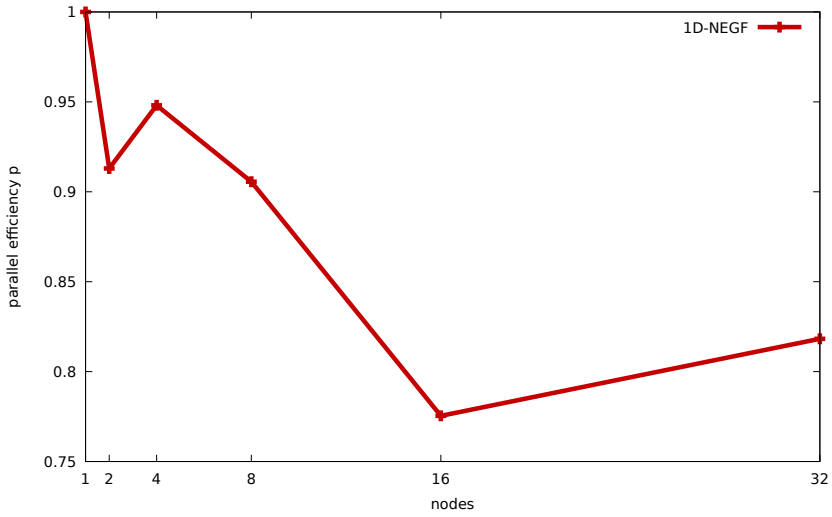
Figure 16: Hybrid scaling of the self-energy integration on JUQUEEN. The system sizes are $N_K = 2048$, $N_E = 5376$, and $N_P = 100$.

5.2.4 *Scaling of one complete self-consistent iteration*

The results before have only shown the scaling for the integration of the self-energy. This is clearly the most crucial part in the distributed implementation, but it is also important to prove that the other parts are also scaling as good as expected. In figure 17 the speedup and efficiency of one complete iteration of the NEGF implementation is shown. Comparing the results, we can see that the scaling of one complete iteration scales as good as only the integration of the self-energy.



(a) Speedup



(b) efficiency

Figure 17: One complete NEGF iteration on JURECA. The speedup and efficiency of the hybrid implementation is shown. The system sizes are $N_K = 256$, $N_E = 256$, and $N_P = 100$.

5.3 NIN-DIODE

A schematic drawing of the structure of the diode is presented in figure 18. The following material parameters have been used:

$$m = 0.25 m_0 \tag{101}$$
$$\epsilon_0 = 11.92 \epsilon \tag{102}$$
$$\epsilon_\infty = 9.93 \epsilon \tag{103}$$
$$q_0 = 5 \times 10^{-5} \tag{104}$$
$$\hbar\omega_{LO} = 37.5\text{meV} \tag{105}$$

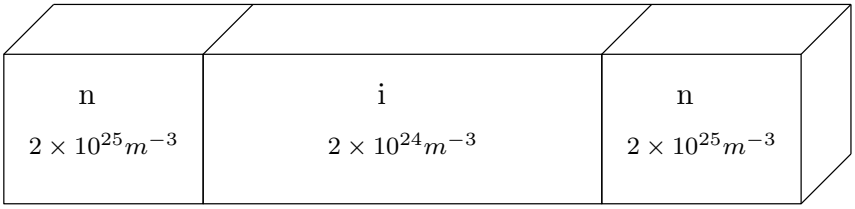


Figure 18: Schematic of diode with the doping N_d.

5.3.1 *Ballistic results*

In the following section the physical results from the ballistic NIN diode are presented.

5.3.1.1 *Charge density*

In figure 19 the charge density of the diode is shown.

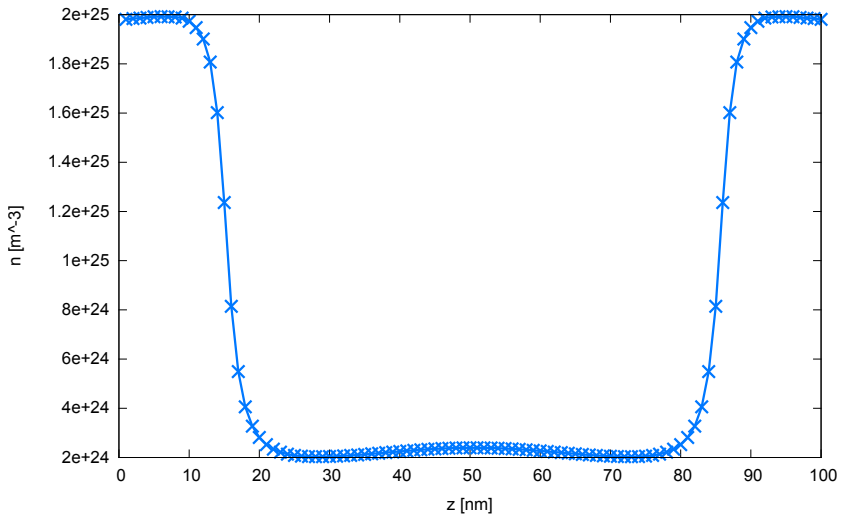


Figure 19: Charge density $n(z)$ of the ballistic nin-diode plotted as a function of the position z.

5.3.1.2 *Potenial*

The spectral current for the same situation is shown in figure 20.

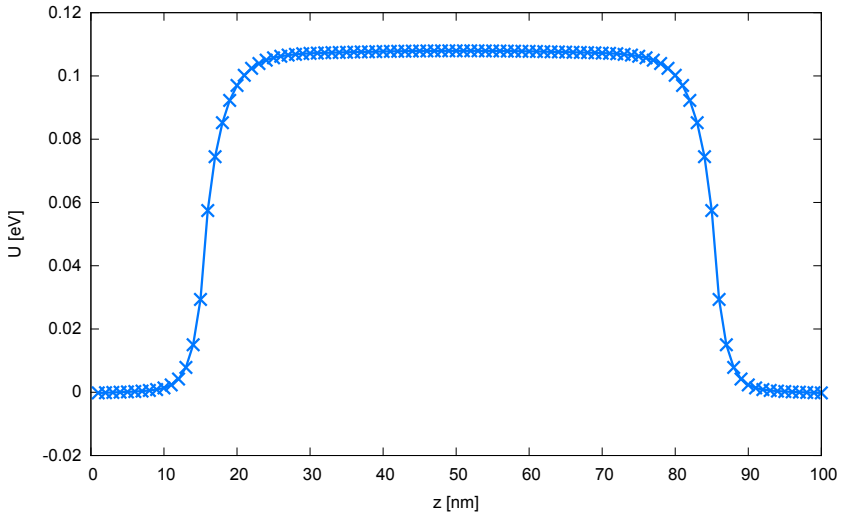


Figure 20: Potential $U(z)$ of the ballistic nin-diode plotted as a function of the position z.

5.3.1.3 *Spectral charge density distribution*

Figure 21 shows the spectral charge density $n(z = 50\text{nm}, E)$ for a fixed position in the middle of the device at $z = 50\text{nm}$ as a function of the energy E. Each momentum contribution $n(z, k, E)$ is plotted in a different color. In this case equation 61 is fulfilled:

$$0.53\text{eV} = \frac{K_{max}^2 \hbar^2}{2m} \geq E_{max} = 0.5\text{eV}. \tag{106}$$

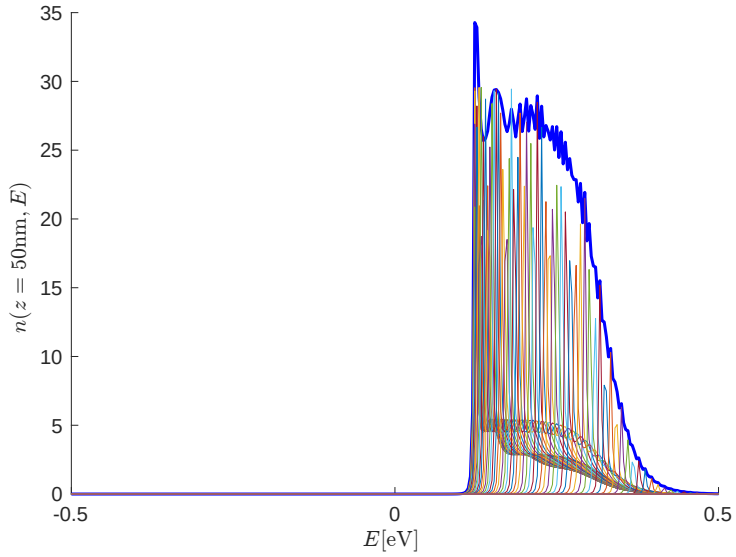


Figure 21: Spectral charge density $n(z = 50\text{nm}, E)$ of the nin-diode plotted as a function of the energy E. Each momentum contribution $n(z, k, E)$ is plotted in a different color.

5.3.2 *Inelastic results*

In the following the results of the inelastic scattering are presented. The system size is the same as before, except that now an external voltage is applied.

5.3.2.1 *Charge density*

The charge density for an applied voltage of 0.2 V is shown in figure 22.

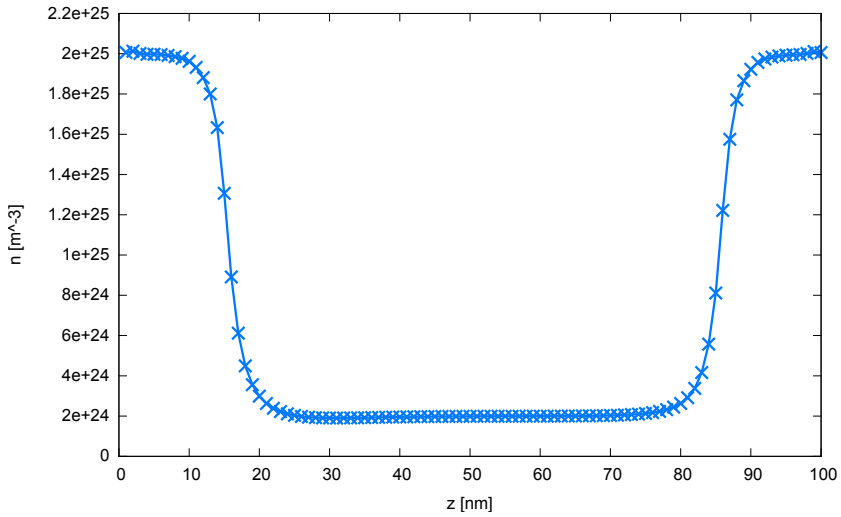


Figure 22: Charge density $n(z)$ of the nin-diode with inelastic scattering plotted as a function of the position z.

5.3.2.2 *Spectral current*

The spectral current is shown in figure 26.

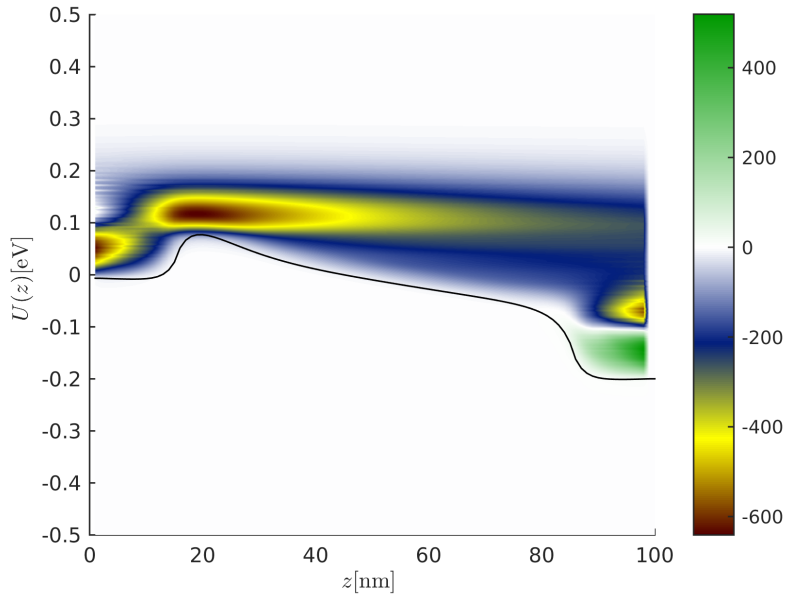


Figure 23: Spectral current and potential $U(z)$ of the nin-diode with inelastic scattering plotted as a function of the position z.

5.3.2.3 *I-V characteristics*

The I-V characteristics is given in 24.

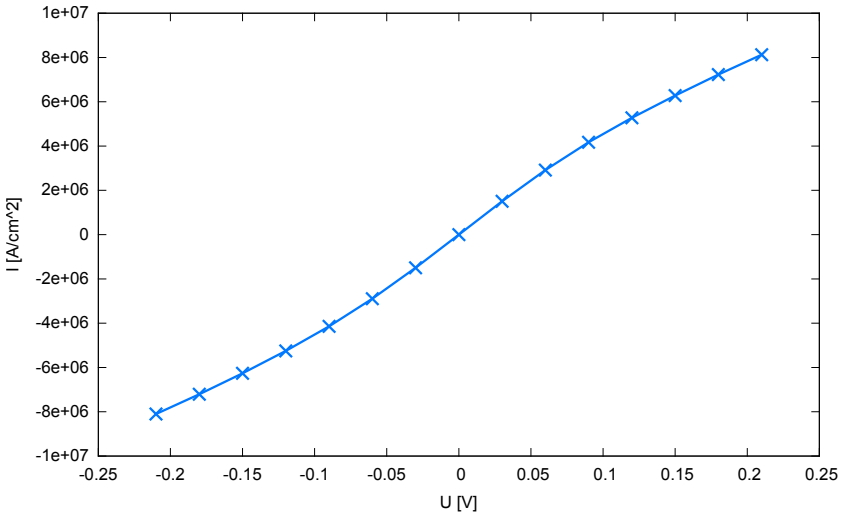


Figure 24: Current I plotted as a function of the applied bias U.

5.4 HIGH BARRIER

In the figure 26 the spectral current is shown for a case with a barrier. The device consist of two different materials which leads to an effective barrier due to the different band offset. This leads to a step-function like term in the

Hamiltonian, which can be also seen in the potential. This case represents a unipolar device with a barrier comparable to a Schottky junction.

A schematic drawing of the structure of the diode for the high barrier case is presented in figure 25. The following material parameters have been used for material 1 and 2, respectively:

$$m_1 = 0.067 m_0 \tag{107}$$
$$\epsilon_{0,1} = 13.18\epsilon \tag{108}$$
$$\epsilon_{\infty,1} = 10.89\epsilon \tag{109}$$
$$m_2 = 0.092 m_0 \tag{110}$$
$$\epsilon_{0,2} = 11.80\epsilon \tag{111}$$
$$\epsilon_{\infty,2} = 9.36\epsilon \tag{112}$$
$$\Delta E_{band\ offset} = 0.1\text{eV} \tag{113}$$
$$q_0 = 5 \times 10^{-5} \tag{114}$$
$$\hbar\omega_{LO} = 37.5\text{meV} \tag{115}$$

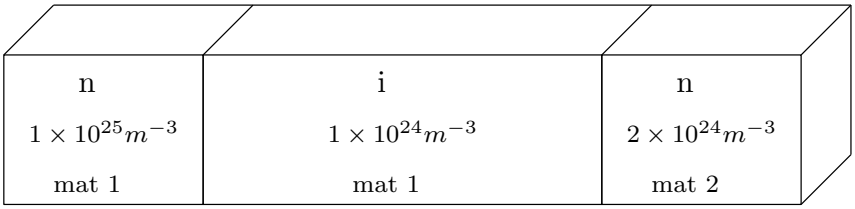


Figure 25: Schematic of diode for the high barrier case with the doping N_d.

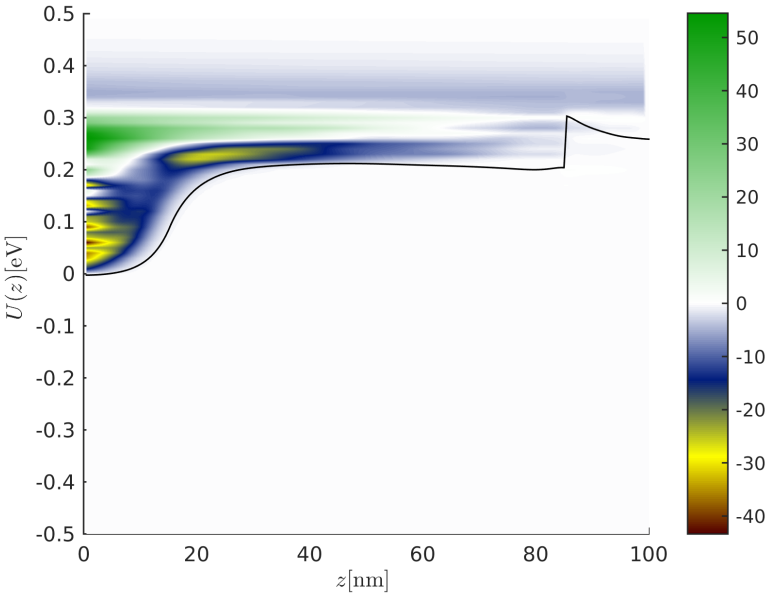


Figure 26: Spectral current and potential $U(z)$ of the nin-diode with a barrier plotted as a function of the position z.

6 CONCLUSION AND OUTLOOK

6.1 CONCLUSION

Within the scope of a this thesis, an efficient parallelization of the NEGF method for nanodevices has been implemented. In chapter 4, a detailed exposition of the strategy has been provided. Specifically, the integration of the polar optical phonon self-energy for the inelastic scattering has been derived and the difficulties which occur when the integration is across multiple nodes have been pointed out.

It has been shown that the different parallelization approaches work perfectly and show good scaling. The OpenMP approach has a speedup of 19.70 on 24 cores. Even for non ideal (regarding the parallelization) input values the code is still scaling very good, with a speedup of 14.34 on 24 cores.

With the MPI version the NEGF simulation can run on multiple nodes. The parallelization is performed such that momentum and energy points are distributed across the nodes. This allows to run on a large number of nodes. The integration across the nodes is here the most crucial point. It has been shown that the integration of the charge density as well as the integration of the self-energy in the inelastic part can be parallelized efficiently. The MPI parallelization is scaling very good on JURECA and also on JUQUEEN.

The MPI implementation has been extended to a hybrid implementation. The work on the node level, the spatial points, are parallelized with OpenMP threads. This hybrid parallelization allows to exploit the architecture of modern supercomputer most efficiently. Also this implementation scales on JURECA and JUQUEEN. The integration of the self-energy is scaling on up to 458,752 cores on JUQUEEN.

The NEGF algorithm that has been implemented can be used to simulate a nin-diode including inelastic phonon scattering as well as a case with a high barrier. The code can calculate the charge density, the potential profile, the spectral charge distribution, the spectral current distribution, as well as the I-V characteristic line, which is are very important quantities for the simulation of nanodevices.

6.2 OUTLOOK

The parallelization presented in this work does apply for the general NEGF equations and the polar optical phonon scattering in particular. This allows us to simulate simple diodes. More complex devices require additional scattering mechanisms or other kind of scattering, e.g. scattering with photons. The developed parallelization approach can be extended to address also these

features. This is possible, since the general structure, e.g. the distribution of the momentum, energy and spatial points remain the same. Thus, the approach presented in this thesis can be transfered with minor changes.

In this work a simple mixing, also known as Kerker mixing, scheme was used. This is the most basic mixing scheme for a *self-consistent field (SCF) iteration*. These SCF iterations also occur within the Density function theory (DFT) framework [17]. It has been shown that the mixing factor α depends on the material and on the size of the system [23]. For sufficiently small α the simple mixing procedure will always lead to convergence, however it might take a large amount of iterations. The DFT community has developed techniques for accelerating the self consistent field iteration. Two main methods for accelerating have been developed: advanced mixing schemes and preconditioning [9]. The important mixing schemes are Anderson's method [3], Broyden's method [5] and Pulay's method [29], which is based on Direct Inversion of Iterative Subspaces (DIIS). The second approach uses an effective preconditioner to minimize the number of iteration. Two examples for preconditioner used in the DFT community are the Kerker preconditioner [16] and an elliptic preconditioner [24]. Usually a combination of mixer and preconditioner is used for accelerating the SCF iteration, e.g. Anderson mixing + Kerker preconditioner. While the number of SCF cycles are reduced by this choice, the construction of the preconditioner comes with an additonal computational cost. The goal is to optimize this issue to achieve an overall increase in accuracy and decrease in the overall computational cost. In the future work these innovative techniques could be adapted to the NEGF framework.

The accuracy of the program depends on the granularity of the used meshes. In the present implementation an equidistant grid was used. This might not be ideal. Some parts in the energy mesh require higher accuracy whereas other parts don't benefit from a higher accuracy. E.g. localized states correspond to a sharp energy peaks. Also quantum effects like confinement lead to sharp energy resonance. Due to inelastic scattering states that haven't been occupied before can be occupied. This would require an energy refinement. In order to resolve these issues an adaptive integration mesh needs to be implemented. Another constraint which makes this more challenging is the compatibility with the conservation laws. This adaptive integration mesh will lead to better accuracy and also reduce the number of self-consistent cycles, and thus effectively shorten the simulation time.

BIBLIOGRAPHY

[1] U Aeberhard and RH Morf. "Microscopic nonequilibrium theory of quantum well solar cells." In: *Physical Review B* 77.12 (2008), p. 125343.

[2] Urs Aeberhard. "Theory and simulation of quantum photovoltaic devices based on the non-equilibrium Green's function formalism." In: *Journal of computational electronics* 10.4 (2011), pp. 394–413.

[3] Donald G Anderson. "Iterative procedures for nonlinear integral equations." In: *Journal of the ACM (JACM)* 12.4 (1965), pp. 547–560.

[4] E. Anderson et al. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-447-8 (paperback).

[5] Charles G Broyden. "A class of methods for solving nonlinear simultaneous equations." In: *Mathematics of computation* 19.92 (1965), pp. 577–593.

[6] Henrik Bruus and Karsten Flensberg. *Many-body quantum theory in condensed matter physics: an introduction*. Oxford University Press, 2004.

[7] Supriyo Datta. *Electronic transport in mesoscopic systems*. Cambridge university press, 1997.

[8] Supriyo Datta. *Quantum transport: atom to transistor*. Cambridge University Press, 2005.

[9] PH Dederichs and R Zeller. "Self-consistency iterations in electronic-structure calculations." In: *Physical Review B* 28.10 (1983), p. 5462.

[10] David K Ferry, Stephen M Goodnick, and Jonathan Bird. *Transport in nanostructures*. Cambridge University Press, 2009.

[11] Alexander L Fetter and John Dirk Walecka. *Quantum theory of many-particle systems*. Courier Corporation, 2003.

[12] Message Passing Interface Forum. *MPI: A Message-passing Interface Standard, Version 3.1 ; June 4, 2015*. June 2015.

[13] Hartmut Haug and Antti-Pekka Jauho. "Quantum kinetics and optics of semiconductors." In: (1996).

[14] Leo P Kadanoff and Gordon A Baym. *Quantum statistical mechanics*. Benjamin, 1962.

[15] Leonid V Keldysh. "Diagram technique for nonequilibrium processes." In: *Sov. Phys. JETP* 20.4 (1965), pp. 1018–1026.

[16] GP Kerker. "Efficient iteration scheme for self-consistent pseudopotential calculations." In: *Physical Review B* 23.6 (1981), p. 3082.

[17] Walter Kohn and Lu Jeu Sham. "Self-consistent equations including exchange and correlation effects." In: *Physical review* 140.4A (1965), A1133.

[18] T Kubis, C Yeh, P Vogl, A Benz, G Fasching, and C Deutsch. "Theory of nonequilibrium quantum transport and energy dissipation in terahertz quantum cascade lasers." In: *Physical Review B* 79.19 (2009), p. 195323.

[19] Tillman Kubis. "Report." In: (2011).

[20] Tillmann Kubis. "Quantum transport in semiconductor nanostructures." PhD thesis. Technische Universität München, 2009.

[21] Roger Lake, Gerhard Klimeck, R Chris Bowen, and Dejan Jovanovic. "Single and multiband modeling of quantum electron transport through layered semiconductor devices." In: *Journal of Applied Physics* 81.12 (1997), pp. 7845–7869.

[22] S-C Lee and Andreas Wacker. "Nonequilibrium Green's function theory for transport and gain properties of quantum cascade structures." In: *Physical Review B* 66.24 (2002), p. 245314.

[23] Marjana Lezaic. *Lecture notes in Density Functional Theory and Electronic structure*. 2015.

[24] Lin Lin and Chao Yang. "Elliptic preconditioner for accelerating the self-consistent field iteration in Kohn–Sham density functional theory." In: *SIAM Journal on Scientific Computing* 35.5 (2013), S277–S298.

[25] Mathieu Luisier. "Atomistic simulation of transport phenomena in nanoelectronic devices." In: *Chemical Society Reviews* 43.13 (2014), pp. 4357–4367.

[26] Paul C Martin and Julian Schwinger. "Theory of many-particle systems. I." In: *Physical Review* 115.6 (1959), pp. 1342–1373.

[27] Mahdi Pourfath. *The Non-Equilibrium Green's Function Method for Nanoscale Device Simulation*. Springer, 2014.

[28] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[29] Peter Pulay. "Convergence acceleration of iterative sequences. The case of SCF iteration." In: *Chemical Physics Letters* 73.2 (1980), pp. 393–398.

[30] Sebastian Steiger. "Modelling Nano-LEDs." PhD thesis. SWISS FEDERAL INSTITUTE OF TECHNOLOGY, 2009.

[31] Sebastian Steiger, Ratko G Veprek, and Bernd Witzigmann. "Electroluminescence from a quantum-well LED using NEGF." In: (2009), pp. 1–4.

[32] Sebastian Steiger, Michael Povolotskyi, Hong-Hyun Park, Tillmann Kubis, and Gerhard Klimeck. "NEMO5: A parallel multiscale nanoelectronics modeling tool." In: (2011).

[33] *Top 500 November 2016*. https://www.top500.org/lists/2016/11/. Accessed: 2016-12-01.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of January 23, 2017 (`classicthesis` version 4.2).

DECLARATION

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, January 2017

Alexander Sebastian Achilles