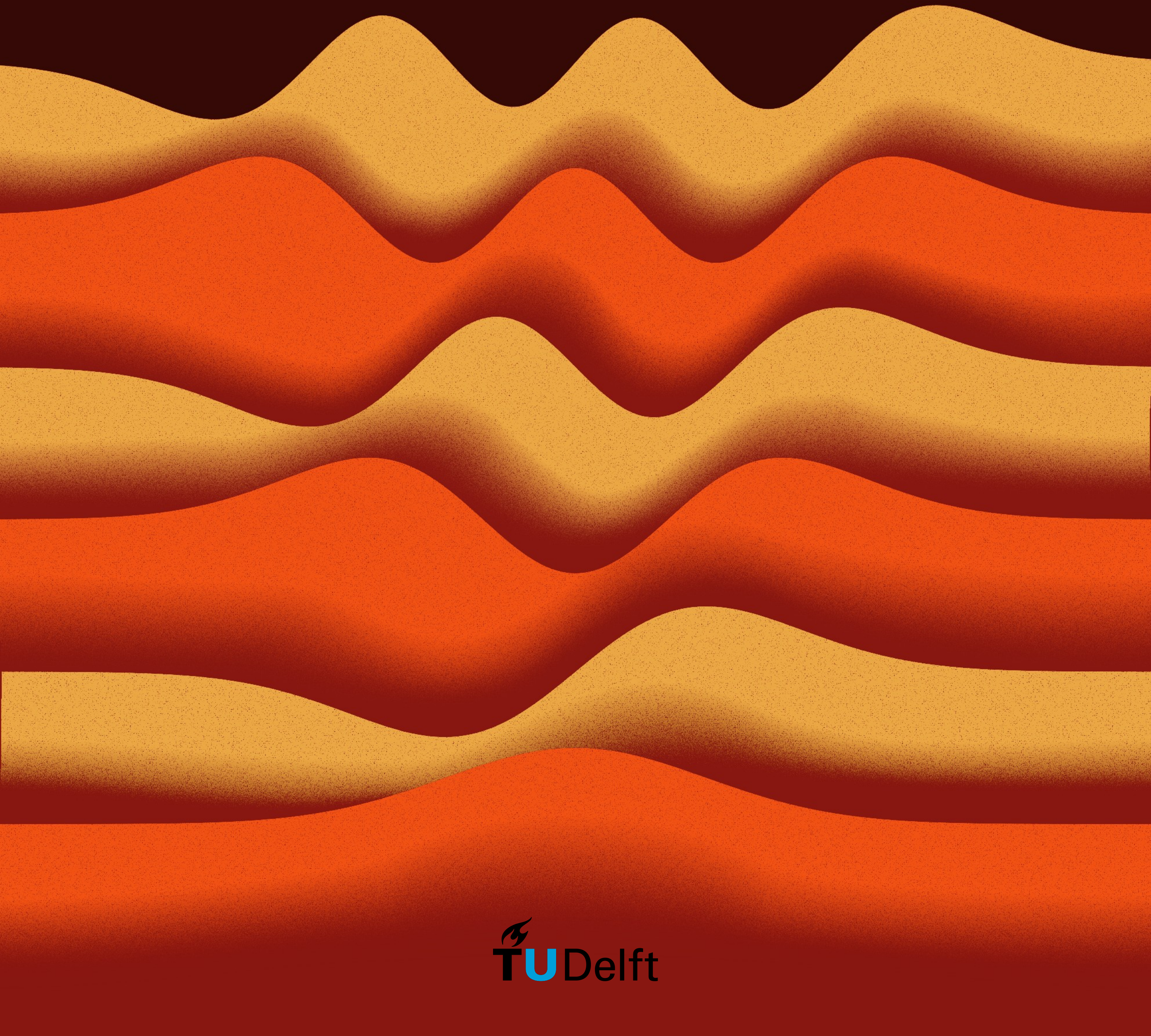
Discrete geometry optimization for quantum dot devices

Supervisors:
DR. ANTON AKHMEROV
DR. MATTHIAS MÖLLER
DR. KOSTAS VILKELIS
JUAN TORRES LUNA

Author:
JULIAN SANDERS

May 7, 2025



To Mik,

whose beautiful brain did not take to math,
but took to injustice even less

who fought so extraordinarily hard,
for themself and for others

whose battle has come to an end,
and who is finally at peace,
somewhere in the cosmos,
between the sun and the moon,
scattered in the laws of nature

I'm proud of you.

Contents

Abstract

While quantum devices have seen major advancements in recent years, there are still significant challenges to scaling up their computational power. Geometry optimization techniques pose a useful tool for tackling these challenges and improving the characteristics of quantum dot devices.

These devices consist of metal gate electrodes on a semiconductor heterostructure. Within the semiconductor heterostructure, the electron wavefunctions used as the qubits are 'trapped' by the potential induced by these metal gates.

In this work, we have modeled the potential induced by the gates by discretizing the corresponding Poisson equation using the finite-volume method. The discretized linear system is solved with factorization-based solvers, of which we make repeated calls more efficient by applying the Woodbury identity. The potential is used to solve the Schrodinger equation, of which the eigenstates are transformed to a maximally localized basis to obtain the dot wavefunctions. The gate voltages of the device are tuned so the effective Hamiltonian of the dots approaches a target Hamiltonian. We have modelled the disorder sensitivity of the devices by inducing changes to the boundary of the gate electrodes, for which the model is evaluated efficiently by utilizing perturbation theory.

Using this device model, we have implemented a discrete geometry optimization algorithm to optimize for the gate electrode shapes. This algorithm generates a range of random changes to the geometry shape and evaluates which one has the best characteristics.

We have demonstrated that this technique is effective for optimizing devices to be less sensitive to gate shape disorder, to have higher level spacing, and to have more local gate-dot interactions. We have applied it to double dot devices, triple dot devices, and double dot devices with wires. The algorithm does not converge to the global minimum of the optimization problem, as different initial conditions lead to marginally different results.

We have implemented several strategies for the sake of computational efficiency. The use of the Woodbury identity, perturbation theory for loss function gradients, and linear corrections for disordered geometries lead to an estimated speedup of more than 62 times.

Since the aim of this project was to be a proof-of-concept for geometry optimization techniques for quantum devices, we simplified some of the dynamics for computational efficiency or coding efficiency. We have not modelled the Coulomb repulsion between electrons in different dots, nor the effects of strain on the system. Additionally, the square-grid discretization of the gate electrodes has an impact on the resulting geometries.

Nonetheless, we have established that it is possible to apply discrete geometry optimization techniques to improve the characteristics of modelled quantum dot devices. Moreover, we have successfully introduced various strategies to improve the computational efficiency of the model.

Readers guide

As this thesis covers the work of a double-degree master's thesis project in Applied Physics and Applied mathematics, it incorporates components from both fields. There is significant overlap between these two aspects of the work: To motivate the mathematical techniques requires explaining the physical processes, and to describe how the physics is modelled involves giving mathematical details. It is more sensible to write about both aspects together than to separate them. However, this report will be read by Mathematicians and Physicists with different backgrounds and research interests. That is why we provide this reader's guide to suggest which sections and subsections will be especially relevant and interesting depending on the reader's background and interest.

In case you, the reader, have a background in Mathematics, we assume that you will be interested in how the device is modelled, how the optimization algorithm works and what techniques have been applied to make the algorithm more computationally efficient. Section 3 dives into the numerical methods used to model the device. Section 4 explains the discrete geometry optimization algorithm. Concerning the techniques applied for computational efficiency, they are explained in subsections 3.1, 3.3, 3.4 and 4.4. Their efficacy is shown and discussed in section 6. If you are interested in computational science, the explanation of how the algorithm is parallelized in subsection 4.4 could intrigue you. Subsection 5.4, which discusses the convergence of the algorithm can also be of particular interest.

Several novel techniques have been introduced in this work. In subsection 3.1 it is explained how we leveraged the Woodbury identity to compute solutions for the Poisson equation for various boundary conditions more efficiently. Additionally, we applied perturbation theory to obtain derivatives of the target function of a minimization procedure, see subsection 3.4. Similarly, perturbation theory is used to quickly calculate the effect of shape disorder to the boundary conditions of the Poisson equation, see subsection 3.5. The application of perturbation theory in this instance required a novel technique, namely the perturbation theory for Wannier transformations. This technique is explained in 3.3, with supplementary information in Appendix A.

If you have a background in physics, we presume you will be more interested in what our algorithm models and how effective it is. A description of the physics of the device is given in section 2. A detailed description of how the device is modelled can be found in section 3. The algorithm itself is explained in section 4. There are several possible target quantities that the algorithm can optimize for. The most important, shape disorder is explained in subsection 3.5. How the level spacing of the quantum dots and the gate-dot coupling is quantified is also explained in subsection 3.5. The efficacy of the algorithm is demonstrated for various devices and targets in section 5.

1. Introduction

The field of quantum computing has seen major advancements in the last years, most notably devices that can perform error correction below the threshold needed to scale up logical qubits [1, 2]. However, significant challenges remain. In order to build quantum computers that can outperform classical computers, more work is needed in two essential facets. Primarily, we need to scale up the number of qubits on a device. Additionally, we need to improve the coherence time of qubits and the fidelity of gate operations. With a sufficient number of good-quality qubits, quantum algorithms become possible at scales on which they outperform classical algorithms [3, 4].

Semiconductor spin qubits are a promising platform for implementing quantum computation [5]. This is due to the advantages such as scalability [6], long coherence times, fast two-qubit gates, fault-tolerant operation [7], and established R&D [8]. Moreover, it is possible to integrate semiconducting qubits on chips containing classical transistor electronics [9]. These advantages provide a promising outlook regarding the possible qubit numbers and quality of devices implemented with this technology.

On the flip side, there are challenges in manufacturing large numbers of sufficient quality semiconductor spin qubits. Although the established R&D of the semiconductor industry allows for the manufacturing of devices with many qubits, the control of those qubits is more challenging to scale up [10, 11]. Additionally, various sources of disorder cause qubit decoherence, and the fidelity of the qubit gates is limited by the quality of the materials used [12, 13, 14].

Current lithography techniques offer a unique advantage for semiconductor quantum dot devices; flexibility in the design of the shapes of the gate electrodes. However, state-of-the-art quantum dot devices are fabricated using rectangular or polygonal gate regions [7].

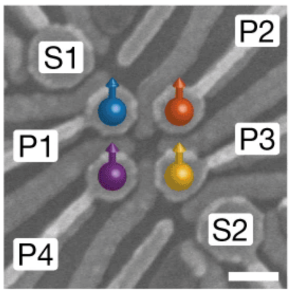
This poses an interesting question: **Could the flexibility in gate design be used to improve the quality of quantum dot devices?**

In this work, we demonstrate that it is worthwhile to implement a discrete shape optimization algorithm for the gate electrodes of semiconductor quantum dot devices. Moreover, we have shown numerically that the optimized gate geometries found by our algorithm have improved characteristics.

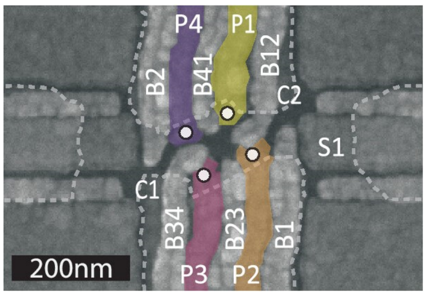
2. Physics of semiconductor quantum dot devices

By fabricating a device with two differently doped layers of semiconductor, a two-dimensional electron gas (2DEG) is created. On the top layer of this semiconductor stack, metal electrodes are deposited. Examples of these metal electrodes on devices that have been fabricated by experimentalists are shown in figures 1a, 1b and 1c. When a voltage is applied to these gate electrodes, they induce an electrostatic potential in their surrounding space. By applying a proper set of gate voltages to these electrodes, valleys in the electrostatic potential in the 2DEG are formed. These valleys serve to trap electrons within certain regions, referred to as quantum dots. The spins of these isolated particles are used to define qubits.

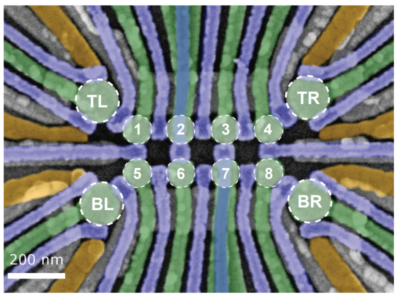
In order to perform quantum algorithms, we not only need to have qubits but also single-qubit and two-qubit gate operations. In a semiconductor spin qubit device, single-qubit gate operations are performed by applying microwave pulses to the gates. Two qubit gates are performed by varying the potential well shapes. By changing the electrostatic potential so the wavefunctions of two electrons overlap slightly, a two-qubit gate operation is implemented. Due to the Pauli exclusion principle, the interaction between the two electrons will be different depending on their spins.



(1a) A colored Scanning Electron Microscope (SEM) image of the gate electrodes of a quadruple dot device implemented with a Ge/SiGe heterostructure [15].



(1b) A SEM image of a small-footprint quandruple dot device with visible variations in gate structure. Implemented on a ^{28}Si/SiGe heterostructure [16].



(1c) An 8-dot array impmented on a Ge/SiGe quantum well heterostructure. The aim of this device was to research the shuttling of electrons and holes between dots [17].

To get the right potential landscape to have isolated or interacting qubits, the voltages of the gate electrodes of the device need to be tuned. The most simple approach to finding these tuning points is to sweep a range of gate voltages and to measure the characteristics of the device. Once a tuning point has been found that traps electrons, virtual gates are defined by measuring the effect of changing gate electrode voltages.

When a device is fabricated, its gate electrode shapes will have small defects depending on the result of the fabrication process. This can be seen to some degree in subfigures 1a and 1c, but it is most clear in figure 1b. We refer to these defects as ‘shape disorder’ in this work. Due to this shape disorder, the tuning point of the device will be slightly different from the one without shape disorder. This deviation from the theoretical tuning point makes it more difficult to tune the device. Therefore, it is advantageous to design devices to be less sensitive to this shape disorder.

With advances in material technology, other sources of disorder such as charge disorder and lattice defects are having less and less of an impact on device performance. Shape disorder is more difficult to remove, as the fabrication accuracy of lithography techniques is limited in resolution. Given sufficient funds and the right materials, the highest resolution can be that of a 3 nm process [18]. Regardless of what resolution a lab can achieve, an essential step to scaling up the number of qubits is making the device pitch smaller. Hence, the smaller devices will still have a shape disorder proportional to the total gate size.

Ideally, every gate electrode only controls its corresponding quantum dot. If that were the case, every virtual gate would simply correspond to changing the voltage of one gate electrode. However, this perfect gate locality is not possible to achieve, as changing the voltage of one gate electrode also changes the potential landscape for the dots it is not supposed to target. While perfect gate or lever arm locality is not achievable, it nonetheless is an interesting quantity to consider.

To maximize the coherence times of the qubits defined on the device, it is essential to have a large band gap to the energy levels not used as qubit states. The larger this level spacing is, the less leakage there will be from the qubit states to higher energy levels. This level spacing corresponds to the size of the potential well the electrons are trapped in. Narrower potential wells lead to larger level spacings.

3. Numerical simulation of quantum dot devices

The ultimate aim of this project is to optimize the gate geometries of quantum dot devices. While experiments are indispensable in determining the efficacy of device designs, it is not feasible to use them for geometry optimization due to the time and resources it would take. Hence, we need to numerically simulate the behaviour of the spin qubit device to evaluate various gate geometries efficiently. We are interested in quantities such as the device's sensitivity to shape disorder, the level spacing of the dot energy levels, and how the gates are coupled to the dot parameters. Hence, we require our numerical simulation to be able to calculate the values of these parameters.

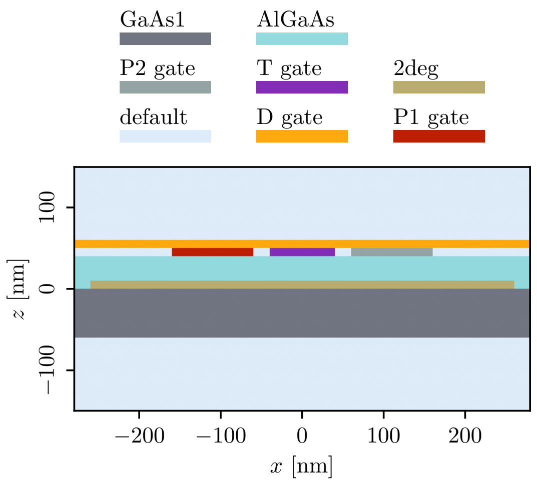
As was explained in the previous section, the quantum dots are created by trapping electrons in potential wells in a 2DEG. In order to model the device in a realistic setting, we combine electrostatic simulations with quantum mechanical simulations. Since the interaction between the qubits is an important quantity, we model the interaction between the dots using their wavefunctions.

Considering our numerical model of the device will be used for geometry optimization, which involves calling the model many times for different geometries, computational efficiency is important. Therefore, we made several assumptions for the sake of better computational performance.

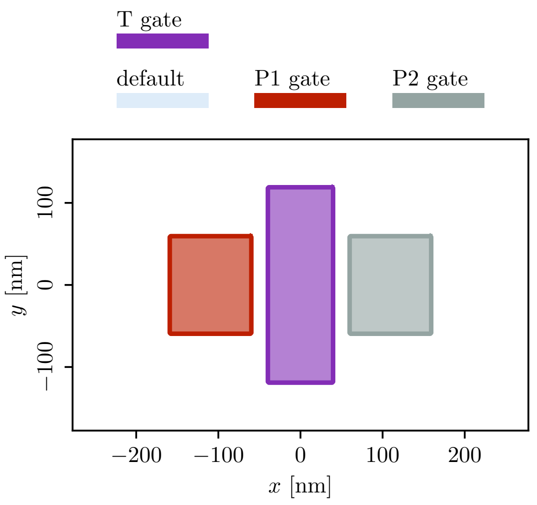
Foremost, we neglect the self-consistency of the electrostatic-Schrodinger equation. That is, we assumed that the effect of the electrons in the 2DEG on the potential was negligible. To model these effects we would need to couple the wavefunction and electrostatic models, as the solution of the wavefunction would impact the shape of the potential and vice versa. Solving this coupled electrostatic-Schrodinger equation would require more computational resources. Nonetheless, the wavefunction solutions would not be much different, as we are operating in a regime where we have few electrons trapped in the potential valleys of a device.

Secondly, we modelled the 2DEG as a two-dimensional plane. A marginally more accurate technique would be to model it in three dimensions, where the electrons are confined in the third dimension by the potential at the interface of the doped semiconductors. However, this would increase the size of the wavefunction model significantly. An essential step of our device model is tuning the device, as will be explained in section 3.4. This involves modelling the wavefunctions for many sets of gate electrode voltages. While it is not infeasible to do this with a 3D model of the 2DEG, the two-dimensional option is a better fit for the proof-of-concept character of this research project. Moreover, for many materials, the wave function is separable in the z direction. Therefore, the z-component does not need to be modelled along with the xy-plane component.

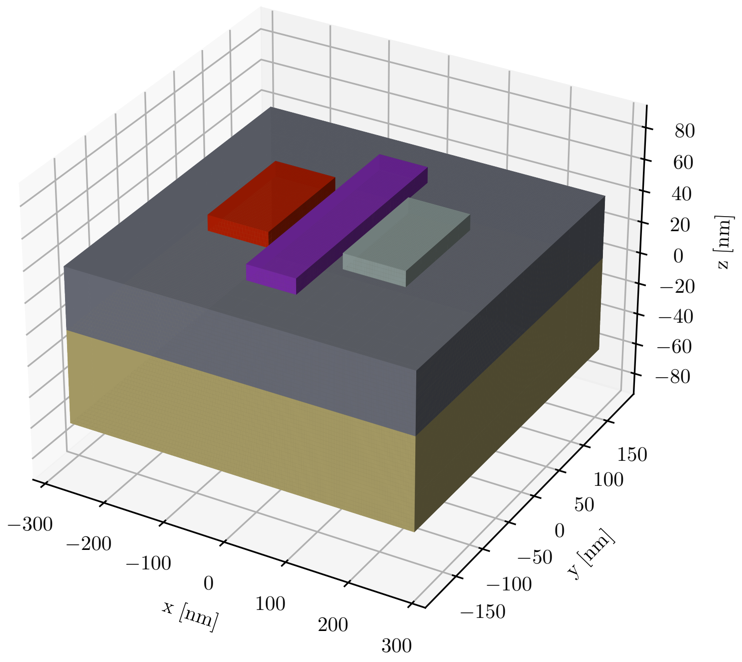
Using these assumptions, we define a model of the different semiconductor layers, the 2DEG plane and the gates. In subfigure 2a and figure 2b the cross sections of this model are plotted. A three dimensional view, without the depletion gate, is shown in figure 2c.



(2a) A side view of the heterostructure of the modelled device. Note that no gate geometry is defined yet, but that optimization regions are plotted instead.



(2b) A top view of yhe initial gate geometries of a double dot device. For the sake of model simplicity, the connecting wires have been left out.



(2c) A three dimentional view of the device heterostructure and gates, with the depletion gate removed for the sake of visibility.

3.1. Modelling the electrostatic potential

To find the electric potential in the 2DEG, we solve Maxwell's equation for the electrostatic potential. As the potential will be constant for constant gate voltages, we use a time-independent form of this equation. This leaves us with a Poisson equation;

$$\nabla^2\phi = \rho \tag{3.1}$$

$$\phi(\mathbf{r}) = V_i \qquad \text{for all } \mathbf{r} \in \delta\Omega_i \qquad \text{for } i \in \{1, ..., n_{\text{gates}}\} \tag{3.2}$$

$$\lim_{\mathbf{r}\to\infty} \phi(\mathbf{r}) = 0. \tag{3.3}$$

By substituting the charge density, ρ, we can solve for the electrostatic potential, ϕ. The boundary conditions of this Poisson equation are imposed by the metal regions. The gate shapes are described by the domains $\Omega_i \subset \mathbb{R}^3$ for $i = 1, ..., n_{\text{gates}}$, with boundaries $\delta\Omega_i$. The gate voltages are $\mathbf{V} \in \mathbb{R}^{n_{\text{gates}}}$. There also is a final boundary condition; infinitely far away from the device, the potential must vanish.

The linear nature of this partial differential equation provides us with a convenient technique: When one wants to solve the equation for a certain set of gate voltages $\mathbf{V}$, one simply takes the linear superposition of Green's functions of the potential of each gate;

$$\begin{aligned}\phi =& V_D\phi_D + V_{P,1}\phi_{P,1} + \\ & ... + V_{T,n_{\text{dots}}-1}\phi_{T,n_{\text{dots}}-1}\end{aligned} \tag{3.4}$$

Here the ϕ_D, $\phi_{P,1}$, ... and $\phi_{T,n_{\text{dots}}-1}$ terms represent the Green's potentials for the different gate electrodes, denoted by the indices (D), $(P,1)$, ..., (P,n_{dots}), $(T,1)$, ... and $(P,n_{\text{dots}}-1)$. The V terms represent the voltage of their respective electrodes.

Thus, when the potentials for multiple gate voltages need to be evaluated, one does not need to solve the Poisson equation for each set of boundary conditions. This is useful, as tuning the gate voltages involves an optimization routine that calls the potential solver for many different gate voltages, and solving the potential once is one of the most computationally intensive processes of our algorithm.

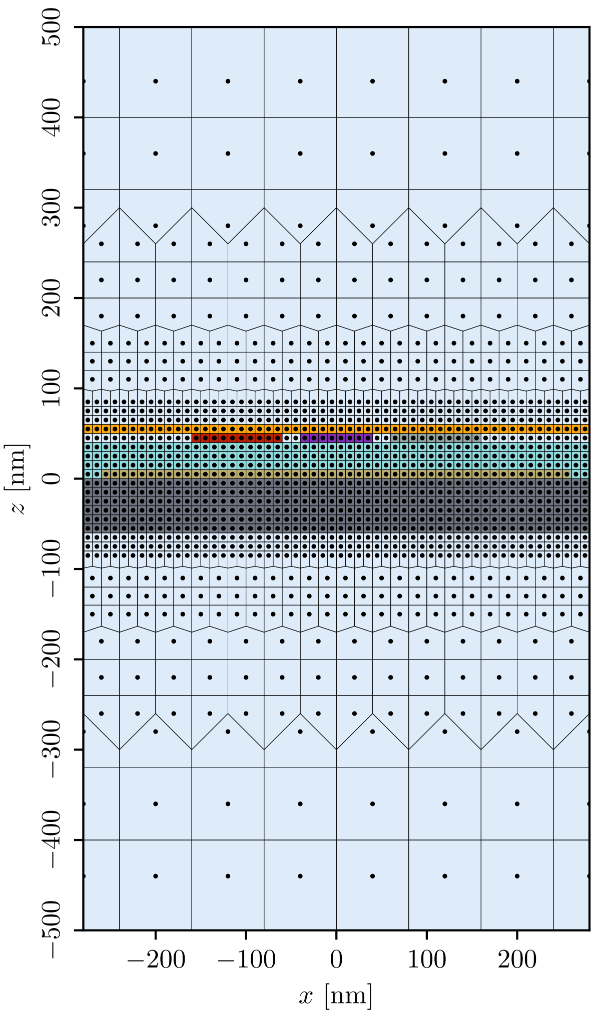


Figure 3: A cross-section of the Voronoi mesh used to discretize the simulation volume. Note that the Vorenoi center grid grows more sparse further away from the device.

Our model uses the Finite Volume Method to solve equation (3.1). We utilize the Python module Pescado for this end [19, 20]. While Finite Element and Finite Boundary methods are more common for Electromagnetic Field simulations, an advantage of the Finite Volume Method is that it ensures flux and charge conservation. The conservation of this quantity is essential for the stability of the convergence of the self-consistent electrostatic-Schrodinger equation solver implemented in Pescado. As we neglect the potential induced by the electrons in the 2DEG, this would not be an issue in our case. Nonetheless, the use of the Finite Volume Method makes it easier to extend the algorithm proposed in this work to use more precise potential solvers.

The Finite Volume Method is a method to discretize partial differential equations that relies on discretizing the domain using cells and using conservation laws derived from the differential equation to define conservation laws for the cell boundaries. In this instance, we can apply it to the strong form of the Poisson equation (3.1). When applying the divergence theorem to each cell individually, we see that

$$\int_{\Omega_i} \nabla \cdot (\epsilon \nabla \psi) dV = \int_{\Omega_i} \rho dV \qquad \text{for all } i \tag{3.5}$$

$$\sum_{j \in N(i)} \int_{S_{i,j}} \epsilon \nabla \psi \cdot \mathbf{n} dS = \int_{\Sigma_i} \rho dV \qquad \text{for all } i \tag{3.6}$$

$$\sum_{j \in N(i)} \Phi_{i,j} = Q_i \qquad \text{for all } i. \tag{3.7}$$

Here Ω_i denotes the interior of cell i, which has surfaces $S_{i,j}$ to its neighboring cells with indices $j \in N(i)$. Q_i and $\Phi_{i,j}$ are variables introduced to denote the total charge in a cell and the flux through the surface between two cells, respectively.

To simplify the flux integral, we can approximate the electrostatic permittivity $\epsilon_{i,j}$ at the boundary using $\epsilon_{i,j} = \frac{\epsilon_i \epsilon_j}{\epsilon_i + \epsilon_j}$. Moreover, we can approximate the directional derivative of the potential using the finite difference method; $\nabla \psi \cdot \mathbf{n}|_{i,j} \approx \frac{\psi_j - \psi_i}{||\mathbf{r}_j - \mathbf{r}_i||} = \frac{\psi_j - \psi_i}{d_{i,j}}$. Here we make use of the new notation $d_{i,j} = ||\mathbf{r}_j - \mathbf{r}_i||$. Note that this aprroximation for $\nabla \psi \cdot \mathbf{n}|_{i,j}$ will introduce an error of order $\mathcal{O}(d_{i,j}^2)$. This leaves us with a simplified expression for the cell boundary fluxes,

$$\Phi_{i,j} \approx \frac{\epsilon_{i,j} S_{i,j}}{d_{i,j}} (\phi_j - \phi_i). \tag{3.8}$$

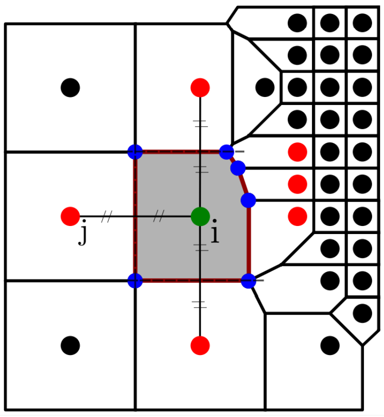


Figure 4: An example diagram of Voronoi cells. An example point i is chosen and is denoted by a green dot. The centers of the neighboring cells are shown in red [20].

We can use this, and the fact that $\sum_{j \in N(i)} \Phi_{i,j} = Q_i$ for all i to construct a linear system to solve for the unknown potentials, ψ_i. This linear system,

$$\Delta \boldsymbol{\phi} = \boldsymbol{\rho}, \tag{3.9}$$

will have elements

$$\Delta_{i,j} = \begin{cases} 0 & \text{if } j \notin N(i) \text{ nor } i \neq j \\ \frac{\epsilon_{i,j} S_{i,j}}{d_{i,j}} & \text{if } j \in N(i) \\ -\sum_{j \in N(i)} \Delta_{i,j} & \text{if } i = j \end{cases} \tag{3.10}$$

and

$$\rho_i = Q_i. \tag{3.11}$$

Since solving the Poisson equation numerically for an infinite domain would not be feasible, we approximate the vanishing boundary conditions with a box of Dirichlet boundary conditions that is much larger than the device itself. We then proceed to discretize the resulting simulation space using a Voronoi mesh. We use rectangular grids to define the center points of the Voronoi cells, see figure 3. The equidistant nature of the cell boundaries makes this meshing approach well-suited to finite-volume methods.

Because we need the potential to be accurate near the 2DEG and the gate electrodes, we chose a fine grid size within the semiconductor heterostructure and surrounding the gates. For the sake of computational efficiency, we use larger rectangular grid spacing further away from the device. Because this limits the number of Voronoi cells, the linear system that results from the discretization of the Poisson Equation will be smaller. This greatly increases the computational efficiency, while only slightly lowering the accuracy of the solution in the region of interest.

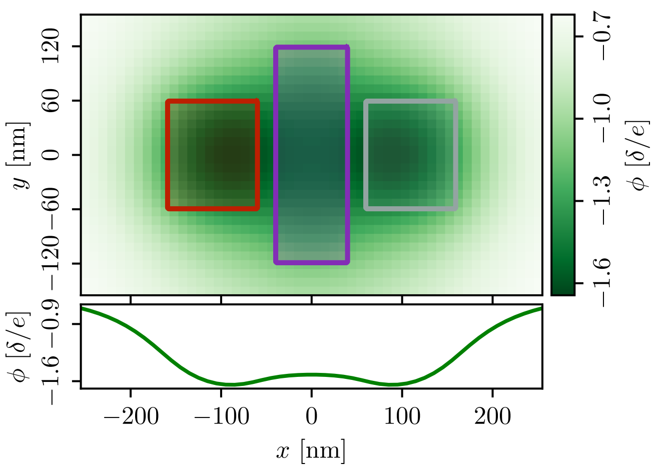


Figure 5: The electrostatic potential in the 2DEG for a tuned double dot device with the initial gate geometries. The lower subfigure is a cross-section of the above plot at $y = 0$, which serves to illustrate the double well potential shape even more clearly than the image plot.

By discretizing the Poisson equation using the Finite Volume method, we obtain a linear system. The solution of this linear system corresponds to the potentials at the Voronoi centers, which we can use to find the electrostatic potential in the 2DEG, see figure 5. To solve this linear system, we chose to make use of a multifrontal factorization algorithm implemented in the MUMPS package [21, 22]. We chose this instead of iterative solvers for two reasons: Primarily, for the linear system size of this proof-of-concept project the decomposition-based solver is faster than the iterative solver. This speedup becomes even more pronounced when considering the fact that for each gate a Green's function potential needs to be calculated. The MUMPS factorization can be re-used for each gate, but an iterative solver would have to be re-run to obtain each gate Green's potential. It is even possible to reuse the factorization of a base linear system to solve the potential when the gate geometries are slightly different.

The aim of this project is to do a device optimization considering shape disorder. Both updating a device geometry in a geometry optimization iteration, as well as modeling shape disorder involve changing the geometry of the gate electrodes slightly. Therefore, much is gained computationally by having a way to re-use the solver information from the base device potential for these slightly changed geometries. We have devised a way to do this by making use of the Woodbury identity [23].

When we change the gate geometries slightly, we change the shape of the boundary of the Poisson equation. To avoid having to generate a new mesh over the new domain, our discretization of the system discretizes both the interior space as well as the boundaries. This means that the gates, which are boundary conditions, are included in the Voronoi mesh.

The Voronoi cells of the discretization (Δ) can be separated into three categories: The interior cells (I), the Neumann boundary sites (N), and the Dirichlet boundary sites (D). The potential (ψ) is unknown at the interior and Neumann sites, but it is known at the Dirichlet boundary sites by definition. The charge (ρ) is unknown in the Neumann sites but known in the rest of the heterostructure. Considering this, we arrange the linear system such that the unknown quantities are on the left-hand side, and the known quantities are on the right-hand side [20],

$$\begin{bmatrix} \Delta_{I,I} & \Delta_{I,N} & 0_{I,D} \\ \Delta_{N,I} & \Delta_{N,N} & 0_{N,D} \\ \Delta_{D,I} & \Delta_{D,N} & I_{D,D} \end{bmatrix} \begin{bmatrix} \phi_I \\ \phi_N \\ \rho_D \end{bmatrix} = \begin{bmatrix} I_{I,I} & 0_{I,N} & \Delta_{I,D} \\ 0_{N,I} & I_{N,N} & \Delta_{N,D} \\ 0_{D,I} & 0_{D,N} & \Delta_{D,D} \end{bmatrix} \begin{bmatrix} \rho_I \\ \rho_N \\ \phi_D \end{bmatrix}. \tag{3.12}$$

When a cell is changed from an interior site to a gate electrode site, the column corresponding to the cell index from the left-hand side matrix is swapped with the one from the right-hand side matrix. Since this changes the linear system by just one column, we use the Woodbury identity to update the factorization of the right-hand side matrix [23]. Naturally, this strategy also extends to geometry changes of several Voronoi cells. However, it will get more computationally expensive the more sites are changed relative to the geometry of the base factorization.

This Woodbury updating strategy is derived by applying the Woodbury identity to these column swaps;

$$A'^{-1} = (A + WV)^{-1} = A^{-1} - A^{-1}W(I + VA^{-1}W)^{-1}VA^{-1}. \tag{3.13}$$

Here A' and A denote the left-hand side of the linear system with and without the geometry change. $W \in \mathbb{R}^{n\times k}$ and $V \in \mathbb{R}^{k\times n}$ compose the operations needed to change update A.

As one can see, finding the Green's potentials of the new device now only involves applying the base factorization, which takes just $\mathcal{O}(n^2)$ FLOPS.

3.2. Modelling the dot wavefunctions and effective Hamiltonian

Using the potential in the 2DEG that our Poisson solver provides, we solve the Schrodinger equation,

$$-\frac{\hbar^2}{2m^*}\nabla^2\psi(x,y) + \phi(x,y)\psi(x,y) = E\psi(x,y). \tag{3.14}$$

This yields wavefunctions, ψ_k, for certain energy levels E_k. To solve equation (3.14) numerically, we discretize this PDE using a finite difference scheme. The resulting tight binding model yields a Hamiltonian of which the eigenvectors and eigenvalues correspond to the eigenstates and energy levels of the quantum system. Numerically, the eigenvectors can be found quickly using the Implicitly Restarted Lanczos Method implemented in the `eigsh` function in scipy [24, 25]. If less than 1e-6 accuracy is acceptable, the Locally Optimal Block Preconditioned Conjugate Gradient Method implemented in the `scipy` function `lobpcg` can be more computationally efficient, depending on the number of eigenvectors desired and the Hamiltonian size [25, 26].

Since the qubits in a quantum dot device are defined based on occupancies of single dots, we need to take our eigenstates to a maximally localized basis. We rotate our eigenbasis to the maximally localized basis using a Wannier operator, W, which is the result of a simultaneous diagonalization of the $\hat{x}$ and $\hat{y}$ operators projected to the eigenbasis [27, 28]. A simultaneous diagonalization is a solution to the minimization problem;

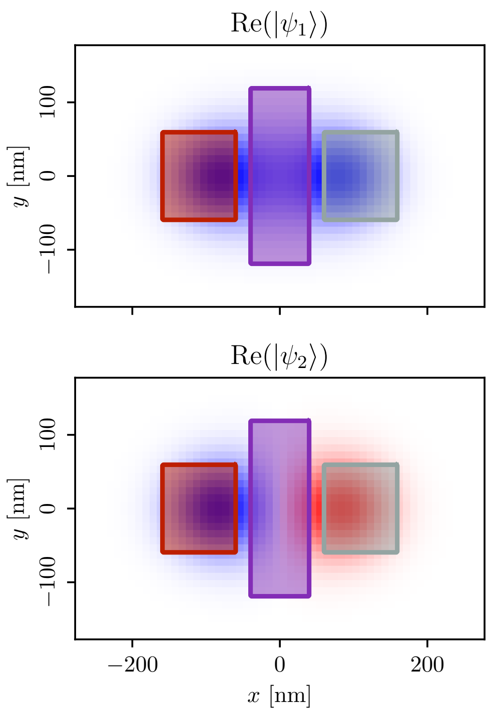
$$\min_{W \text{unitary}} \operatorname{Tr}\left\{(W^\dagger P_x W)^2 + (W^\dagger P_y W)^2\right\}. \tag{3.15}$$

The solution to this minimization problem is obtained by solving

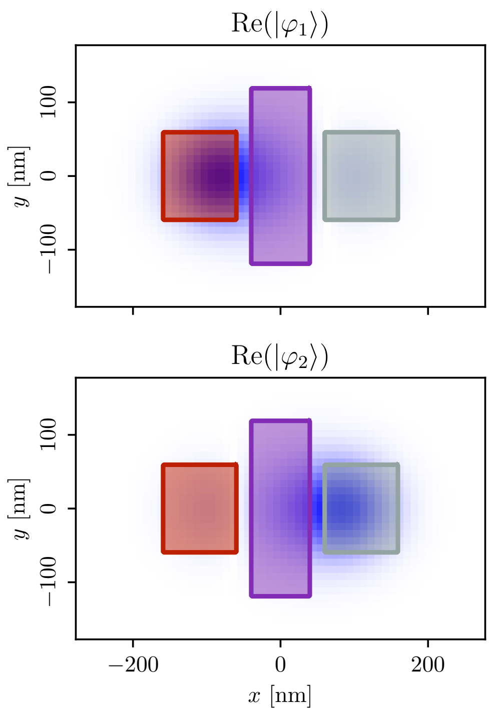
$$\left[W^\dagger P_x W, D(W^\dagger P_x W)\right] + \left[W^\dagger P_y W, D(W^\dagger P_y W)\right] = 0 \tag{3.16}$$

for W, where W must be a unitary $n_{\text{dots}} \times n_{\text{dots}}$ matrix. Here $P_x = \Psi_A^\dagger \hat{x}\Psi_A$ and $P_x = \Psi_A^\dagger \hat{y}\Psi_A$ are the projected position operators in the subspace $\Psi_A = [|\psi_1\rangle, |\psi_2\rangle, ..., |\psi_{n_{\text{dots}}}\rangle]$. The operator $D : \mathbb{R}^{n_{\text{dots}}\times n_{\text{dots}}} \to \mathbb{R}^{n_{\text{dots}}\times n_{\text{dots}}}$ is an operator that refers to constructing a sparse output matrix with just the values of the main diagonal of the input matrix.

When the dot centers are located in a line along the x-axis, as is the case for the simple double dot and triple dot device layouts, the term $[W^\dagger P_y W, D(W^\dagger P_y W)]$ in equation (3.16) is zero. That is due to the shape of the eigenstates, which all have the same y-expectation value. Hence, regardless of how the basis is rotated by operator W, $D(W^\dagger P_y W)$ will be a diagonal matrix with the same value along its diagonal. It therefore commutes with $W^\dagger P_y W$. As the equation to find the Wannier operator is now reduced to $[W^\dagger P_x W, D(W^\dagger P_x W)] = 0$, a solution for W is the orthonormalized eigenvectors of P_x, as when $P_x = W\Lambda W^\dagger$, $[W^\dagger P_x W, D(W^\dagger P_x W)] = [\Lambda, D(\Lambda)] = [\Lambda, \Lambda] = 0$. This diagonalization of P_x is a less computationally complex procedure than a simultaneous diagonalization of P_x and P_y, which is why we implemented it for the simple double and triple dot cases where the dots are located along the x-axis.



(6a) The first two eigenfunctions of the initial double dot device



(6b) The maximally localized wavefunctions of the initial double dot device.

Figure 6: The eigenfunctions and maximally localized wavefunctions for the initial geometry (outlined shape plot) of a double dot device tuned to $t = 0.005\ delta$ and $\mu_1 = \mu_2 = -1.5\ \delta$. The wavefunctions are shown as blue-red image plots overlaid on the geometries, which are shown as outlined shape plots.

Using this Wannier operator W, we are able to transform our eigenstate basis to a maximally localized basis; $\Phi = \Psi W$. An example for a simple double-dot device is provided in figure 6.

The effective Hamiltonian of the localized states gives us the chemical potential of their respective dots ($\boldsymbol{\mu}$), as well as the hopping energies between the dots ($\boldsymbol{t}$). This effective Hamiltonian is calculated using the Wannier operator (W) and the eigenvalues;

$$H_{\text{eff}} = W^{\dagger}\text{diag}\{E_1, E_2, \dots, E_n\}W = \begin{bmatrix} \mu_1 & t_{1,2} & t_{1,3} & \cdots & t_{1,n} \\ t_{1,2} & \mu_2 & t_{2,3} & & \vdots \\ t_{1,3} & t_{2,3} & \mu_3 & \ddots & \\ \vdots & & \ddots & \ddots & t_{n-1,n} \\ t_{1,n} & \cdots & t_{n-2,n} & t_{n-1,n} & \mu_n \end{bmatrix}. \tag{3.17}$$

Throughout this work, we have used an energy scale of δ for energy and δ/e for voltage and potential. This particular value was chosen such that the energy, voltage, and potential values can easily be compared to the lattice hopping in the 2DEG, which is 1 in δ units. Shifting the energy values to this scale has the added benefit that these values will not be very small, which could pose issues considering truncation errors of their `double` datatype. The energy unit δ is defined according to the smallest lattice spacing in the 2DEG, so

$$\delta = \frac{\hbar^2}{2m_{\text{eff}}a^2}, \tag{3.18}$$

where m_{eff} is the effective mass of the electrons in the 2DEG and a is the minimal lattice spacing. As we used a lattice spacing of 10 nm, the energy unit $\delta = 0.00568654\ e\text{V} = 9.11084 \cdot 10^{-22}$J.

This implies a voltage and potential unit of $\delta/e = 0.00568654$V. The voltage unit is the energy unit divided by the electron charge since the particles for which we are solving the Schrodinger equation are electrons.

3.3. Perturbations to the Hamiltonian

In subsections 3.4 and 3.5 we will apply perturbations to the model to calculate effective Hamiltonian gradients and to approximate the outcome for slightly different potentials. An overview of the mathematics behind this perturbation theory is provided here.

For simplicity, we will just consider one perturbation to the 2DEG potential, that is $\phi = \phi_0 + \epsilon\phi_1$. Here ϕ is the perturbed potential in the 2DEG, ϕ_0 is the unperturbed potential and ϕ_1 is the perturbation to the potential with coefficient ϵ. The same concepts also work for different perturbations to the potential, $\phi = \phi_0 + \epsilon_a\phi_{1,a} + \epsilon_b\phi_{1,b} + ...$ As the potential in the 2DEG is perturbed, the discretization Hamiltonian, also needs to be perturbed;

$$H = H_0 + \epsilon H_1. \tag{3.19}$$

Here H_0 is the unperturbed Hamiltonian, and H_1 is the perturbation to the Hamiltonian. As this perturbation follows from a perturbation to the potential, $H_1 = \text{diag}\{\phi_1\}$.

The lowest energy eigenvectors of H_0 can be found using numerical methods. These are represented by $\Phi_A = [\phi_1, ..., \phi_{n_\text{dots}}]$. We proceed to use these as the subspace in which the perturbative series H is block diagonalized. That is, we find unitary rotations $U \in \mathbb{R}^{n\times n}$, such that $\tilde{H} = U^\dagger H U$ yields a matrix in which the off-diagonal blocks are zero:

$$\tilde{H} = \begin{bmatrix} \tilde{H}^{AA} & \emptyset \\ \emptyset & \tilde{H}^{BB} \end{bmatrix} \tag{3.20}$$

The algorithm implemented in the Pymablock `block_diagonalize` function in implicit mode provides these matrices as series [29];

$$\tilde{H} = \tilde{H}_0 + \epsilon\tilde{H}_1 + \epsilon^2\tilde{H}_2 + ... \text{ and} \tag{3.21}$$

$$U = U_0 + \epsilon U_1 + \epsilon^2 U_2 + ... \,. \tag{3.22}$$

These series can be queried up to a desired order. However, the higher-order terms progressively take more computational time.

The series for matrix $\tilde{H}^{AA}$ gives 'corrections' to the energies in the interaction matrix $\Psi_A^\dagger(H_0 + \epsilon H_1)\Psi_A$ for different orders. However, we are interested in the corrections to the effective Hamiltonian, $W^\dagger\Psi_A^\dagger(H_0 + \epsilon H_1)\Psi_A W$. In section 3.2 it was outlined how the Wannier matrix W_0 can be found for an unperturbed Hamiltonian. To obtain a series of corrections to the effective Hamiltonian akin to $\tilde{H}^{AA}$, we derived a mathematical way to get a series of corrections for W. For the simple case where the dots are in a line along the x-axis, we drop the P_y term in equation (3.16). Accordingly, we set up our projected position operator in the x direction;

$$P_x = \begin{bmatrix} \Psi_A^\dagger\hat{x}\Psi_A & \Psi_A^\dagger\hat{x}(\mathbb{I} - \Psi_A\Psi_A^\dagger) \\ (\mathbb{I} - \Psi_A\Psi_A^\dagger)^\dagger\hat{x}\Psi_A & (\mathbb{I} - \Psi_A\Psi_A^\dagger)^\dagger\hat{x}(\mathbb{I} - \Psi_A\Psi_A^\dagger) \end{bmatrix} . \tag{3.23}$$

We then rotate this operator to the perturbed eigenvector basis using U from the block diagonalization. This is done as we are interested in the projected position operator corrected for the Hamiltonian perturbation. This yields the expansion;

$$\tilde{P}_x = U^\dagger P_x U = \tilde{P}_{x,0} + \epsilon\tilde{P}_{x,1} + \epsilon^2\tilde{P}_{x,2} + ... \,. \tag{3.24}$$

The procedure differs when the dots are not in a line, the mathematical steps for this are provided in appendix A.

Once we have the expansion for the rotated projected position operator, we can use it to find the series for the Wannier operator, W. As we are working in the 'dots in a line' regime, the simultaneous diagonalization of equation (3.15) reduces to a diagonalization of the $\tilde{P}_x$ expansion. Thus, we are looking for an expansion $W = W_0 + \epsilon W_1 + ...$ that diagonalizes $\tilde{P}_x$;

$$\tilde{P}_x^{AA} = W\Lambda W^\dagger. \tag{3.25}$$

Pymablock can also be used to this end. We first diagonalize matrix $\tilde{P}_{x,0}^{AA}$ to obtain the unitary matrix W_0. Thus, we solve $\tilde{P}_{x,0}^{AA} = W_0\Lambda_0 W_0^\dagger$ using `eigsh`. This W_0 is then used to define series $W_0^\dagger \tilde{P}_x^{AA} W_0$ of which the first term is a diagonal matrix of size n_{dots}. By chaining $n_{\text{dots}} - 1$ block diagonalizations, this series can be fully diagonalized. That is, we have a series of diagonal matrices Λ and a series of matrixes S such that;

$$W_0^\dagger \tilde{P}_x^{AA} W_0 = S\Lambda S^\dagger \ . \tag{3.26}$$

Using the series $S = I + \epsilon S_1 + \epsilon^2 S_2 + ...$ and matrix W_0 the series for W is calculated;

$$W = W_0 S = W_0 + \epsilon W_1 + \epsilon^2 W_2 + ... \tag{3.27}$$

The series for the effective Hamiltonian is obtained by multiplying the series for the Wannier operator, W, and the series for the subspace energies, $\tilde{H}^{AA}$;

$$\tilde{H}_{\text{eff}} = W^\dagger \tilde{H}^{AA} W = \tilde{H}_{\text{eff},0} + \epsilon \tilde{H}_{\text{eff},1} + \epsilon^2 \tilde{H}_{\text{eff},2} + ... \ . \tag{3.28}$$

From this series of the effective Hamiltonian series for $\mu_1, ..., \mu_{n_{\text{dots}}}$ and $t_1, ..., t_{n_{\text{dots}}-1}$ are obtained by selecting the relevant element in the matrix.

Using this series for $\tilde{H}_{\text{eff}}$ we can obtain gradients with respect to the coefficient of the perturbation;

$$\frac{\partial \tilde{H}_{\text{eff}}}{\partial \epsilon} = \tilde{H}_{\text{eff},1}. \tag{3.29}$$

Additionally, we can approximate the effective Hamiltonian for the potential with the perturbation, $\tilde{H}_{\text{eff}}$, up to a certain order of ϵ;

$$\tilde{H}_{\text{eff}} = \tilde{H}_{\text{eff},0} + \epsilon \tilde{H}_{\text{eff},1} + \mathcal{O}(\epsilon^2). \tag{3.30}$$

These two tools will be applied in the next sections.

3.4. Tuning the gate voltages of a device

We need to be able to control the effective Hamiltonian of the trapped particles in the device to perform two-qubit gate operations. Therefore, we need to use our model of how the wavefunctions and their effective Hamiltonian relate to the gate voltages to tune the matrix elements of the effective Hamiltonian independently.

Because we would like to tune the device's effective Hamiltonian to a certain target, we need to define a loss function in terms of the gate voltages that measure how close the device is to the target effective Hamiltonian. In a typical device, the chemical potentials and hopping energy values are of different orders of magnitude, and they respond differently to changes in gate voltages (linear versus exponential). Therefore, it makes sense to treat them differently in the tuning loss function. Thus, we define a tuning loss function consisting of two components;

$$L_{\text{tun}}(\mathbf{V}) = C_t L_t(\mathbf{t}(\mathbf{V})) + C_\mu L_\mu(\boldsymbol{\mu}(\mathbf{V})), \tag{3.31}$$

$$L_t(\mathbf{t}; t_0) = \sum_{i=1}^{n-1} \ln\left(\frac{t_i}{t_0} + \epsilon\right)^2, \tag{3.32}$$

$$L_\mu(\boldsymbol{\mu}; \mu_0) = \sum_{i=1}^{n} \frac{(\mu_i - \mu_0)^2}{t_0^2} \ [30]. \tag{3.33}$$

Here t_0 and μ_0 are the target effective Hamiltonian values. $\epsilon = 10^{-8}$ is a term to limit the unboundedness of the $\ln$ function for values approaching zero. For simplicity, we defined equation (3.32) for n dots in an array. So the only desired hoppings are those between neighboring dots, referred to as $t_1, t_2, \ldots, t_{n_{\text{dots}}-1}$. Naturally, the same concept of a logarithmic scale can be extended to work for devices with more complex dot connectivity arrangements. Note that the loss function components can be combined for different coefficients, which will impact the shape and convexity of the loss landscape. In figure 7 the tuning loss of a double dot device with a fixed depletion gate voltage (0 V) is shown using an image plot for $C_t = 10000$ and $C_\mu = 1$.

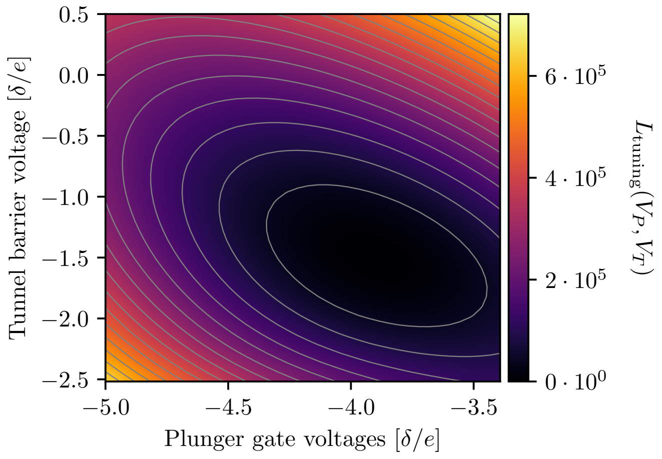


Figure 7: The 'landscape' of the tuning loss function defined by equation (3.31)

As optimization problems with fewer parameters are solved faster, we have found a way to halve the number of free variables in the tuning problem. Primarily, since shifting all gate voltages (in V) up or down by the same amount will yield the same eigenstates but with the eigenenergies shifted by the same amount (in eV), we can fix one gate voltage. The natural choice for this is the depletion gate, as this one would be manipulated the least in an experimental setup. If we would not do so, the optimum of the tuning losses would become degenerate, as we would have more free voltage variables than tuning loss terms for effective Hamiltonian values.

Moreover, we chose to constrain our geometries to be symmetric; meaning that the last plunger gate is a mirror image of the first plunger gate, the first tunnel barrier is a mirror image of the last tunnel barrier, and so on. When such a symmetric device is tuned to a target effective Hamiltonian with symmetric dot $\mu_1 = \mu_{n_{\text{dots}}}, \mu_2 = \mu_{n_{\text{dots}}-1}\ldots$ and $t_1 = t_{n_{\text{dots}}-1}, \ldots$, we can reduce the number of free variables in the optimization problem further by coupling the gate voltages of gates and their symmetric counterpart. Accordingly, we reduce the $2n_{\text{dots}}$ parameter space $\boldsymbol{V} = (V_D, V_{P,1}, ..., V_{P,n_{\text{dots}}}, V_{T,1}, ..., V_{T,n_{\text{dots}}-1})^T$ to $\boldsymbol{V} = (C, V_{P,1}, ..., V_{P,1}, V_{T,1}, ..., V_{T,1})^T$, which only has n_{dots} free variables. This does rely on the assumption that there are no asymmetric imperfections, which would not be the case for disordered devices.

The gradients of the effective Hamiltonian values with respect to the gate voltages are found using perturbation theory. Using the potential Green's functions ($\boldsymbol{\phi}_I$), we can define a perturbation to the Hamiltonian for each gate of the device;

$$H' = H_0 + \epsilon_D \text{diag}\{\boldsymbol{\phi}_D\} + \epsilon_{P,1}\text{diag}\{\boldsymbol{\phi}_{P,1}\} + ... + \epsilon_{T,n_{\text{dots}}-1}\text{diag}\{\boldsymbol{\phi}_{T,n_{\text{dots}}-1}\}. \tag{3.34}$$

We are interested in how these perturbations affect the $\boldsymbol{\mu}$ and $\boldsymbol{t}$ values, as these are the parameters used in the tuning loss functions (3.31). To obtain series that correct the μ and t values for the voltage perturbations, we make use of the theory described in section 3.3.

The Jacobian of the energy values ($\boldsymbol{t}$, $\boldsymbol{\mu}$) is then obtained from their first-order terms of the effective Hamiltonian series;

$$\frac{\partial\boldsymbol{\mu}}{\partial\mathbf{V}} = [D_0(H_{\text{eff}}^{1,D}), D_0(H_{\text{eff}}^{1,P,1}), ..., D_0(H_{\text{eff}}^{1,T,n_{\text{dots}}-1})] \tag{3.35}$$

$$\frac{\partial\boldsymbol{t}}{\partial\mathbf{V}} = [D_1(H_{\text{eff}}^{1,D}), D_1(H_{\text{eff}}^{1,P,1}), ..., D_1(H_{\text{eff}}^{1,T,n_{\text{dots}}-1})]. \tag{3.36}$$

Here $D_k : \mathbb{R}^{n\times n} \to \mathbb{R}^{n-k}$ is an operator that refers to constructing a vector from the elements of the k-th diagonal of the matrix. As this process only involves diagonalizing the Hamiltonian once, it is much faster than finding gradients using a finite difference approach. Proceedingly, the gradient of the effective Hamiltonian values can be combined with the derivatives of the loss functions (3.31), (3.32), and (3.33) to obtain a gradient for the total tuning loss function [31].

Using the Jacobian of the tuning loss function, we apply a gradient-based optimizer to tune the gate voltages of the device. An example of how this optimizer steps towards the tuning point can be seen in figure 8. In our testing, we found the Limited Memory BFGS method to be the fastest [32]. Most notably, for double and triple dot devices it outperforms the fastest gradient-free method, Nelder-Mead [33], and a Hessian-based method, Newton Conjugate Gradient [34].

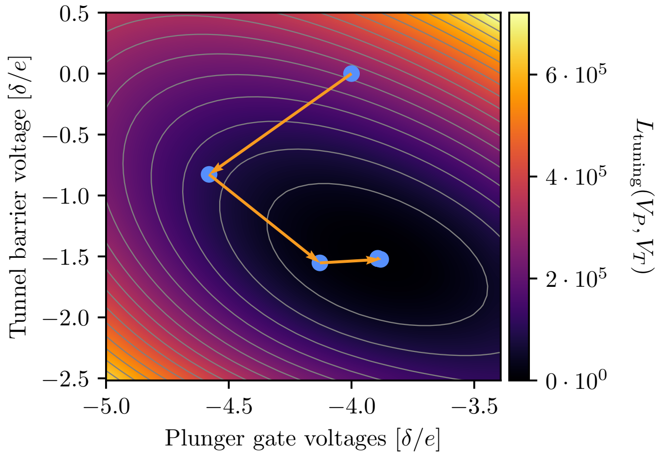


Figure 8: The steps of the Limited Memory BFGS optimizer from the initial gate voltages to the tuning point.

3.5. Modelling shape disorder to the gate geometries

When a device is fabricated, there is a certain manufacturing error that introduces disorder in the shape of the gate electrodes. This disorder causes the device's properties to be slightly different from the undisordered device. For example, in figure 1b small defects in the gate electrode shapes are visible.

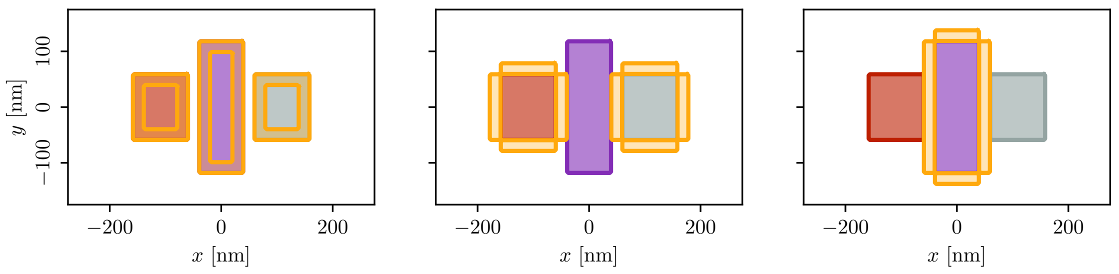


Figure 9: The inside and outside boundaries of the gates where the disorder pixels are added or subtracted. The first plot shows the inside boundary of all the gates, the second plot the outside boundary of the plunger gates, and the third plot shows the outside boundary of the tunnel barrier.

Since we aim to optimize the gate geometries so the device is least sensitive to this shape disorder, we need a way to model it. As the gate electrodes are extrusions of two-dimensional images, we describe their geometries using two-dimensional arrays of pixels. We now model the disorder by adding or removing voxels from the boundaries of the gate geometries. These boundary regions are plotted in figure 9 for the initial geometry. To obtain a measure of how sensitive a certain device is to shape disorder, we loop over all the possible voxels at the gate boundary that can be added or removed. This process is illustrated in figure 10.

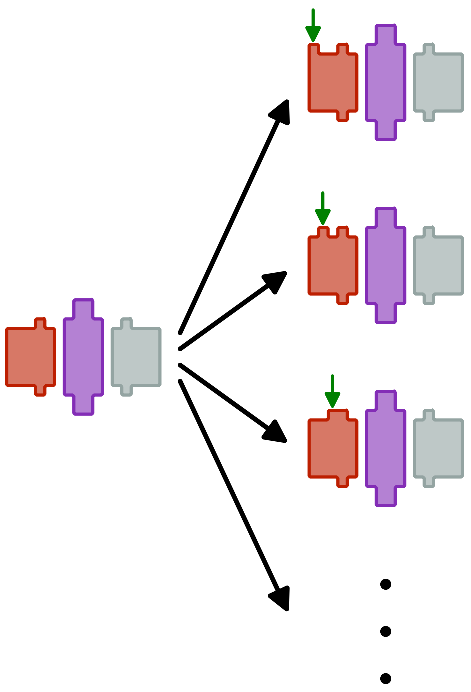


Figure 10: A schematic view of how all one-pixel geometries are generated from some trial device geometry. The green arrows illustrate how the algorithm iterates over all one-pixel changes to the geometries. Note that we loop over all the possible one-pixel boundary modifications, as suggested by the ellipsis.

The tuning points of all these gate geometries with one disorder pixel added or removed (s_i) are then compared to the tuning point of the initial geometry (s_0) using the following formula;

$$L(\mathbf{s}_0) = \sum_{i=1}^{n_{\text{disorder}}} ||\mathbf{V}(\mathbf{s}_i) - \mathbf{V}(\mathbf{s}_0)||^2. \quad (3.37)$$

Here $\mathbf{V}(s)$ refers to the tuning point of a certain geometry (s). Note that there is no normalization term ($\frac{1}{n_{\text{disorder}}}$) in front of the sum since a geometry with a larger boundary is more subject to disorder. We contrived this loss function ourselves, as we needed a metric that combines the voltage deviation without normalizing them for the number of disorders. This would be the case for something like the variance in the voltage deviations.

Because it would take a lot of computational resources to evaluate the whole device model for each disordered geometry, we devised a way to do it more efficiently. As mentioned in section 3.1, we use a Woodbury updating strategy to find the potential Green's functions quickly by reusing the matrix factorization of the non-disordered device.

Moreover, instead of returning the device for each disorder voxel, we approximate the tuning point of the new geometry by making use of perturbation theory. To this end, we perturb the tight-binding Hamiltonian of the base device with the difference of the potential of the new device (ϕ^*_{disorder}) with respect to the old one at the old tuning point (ϕ_{base}). Additionally, we perturb it with the potential Green's functions of the new device;

$$\begin{aligned} H' =& H_0 + \epsilon_{\text{potential}}\text{diag}\{\phi^*_{\text{disorder}} - \phi_{\text{base}}\} \\ &+ \epsilon_D \text{diag}\{\phi^*_D\} + \epsilon_{P,1}\text{diag}\{\phi^*_{P,1}\} + \ldots + \epsilon_{T,n_{\text{dots}}-1}\text{diag}\{\phi^*_{T,n_{\text{dots}}-1}\}. \end{aligned} \quad (3.38)$$

In section 3.3 it is outlined how a series that corrects $\boldsymbol{\mu}$ and $\boldsymbol{t}$ for these perturbations is obtained. Here, we make use of both the approximation and the gradient finding 'tricks' that are possible using these series. By summing over the disorder potential perturbation terms ($\epsilon_{\text{potential}}$ and finding the Jacobian using the Green's function perturbations ($\epsilon_{P,1}, \ldots, \epsilon_{T,n_{\text{dots}}-1}$) we obtain Jacobians of the effective Hamiltonian values of the new device at the old tuning point. These Jacobians are used to correct the gate voltages so the chemical potentials and hopping energies are once again tuned to their target values;

$$\begin{aligned} \frac{\partial \mathbf{t}}{\partial \mathbf{V}}(\mathbf{V} - \mathbf{V}_{\text{tuned}}) &= \mathbf{t} - \mathbf{t}_0 \\ \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{V}}(\mathbf{V} - \mathbf{V}_{\text{tuned}}) &= \boldsymbol{\mu} - \boldsymbol{\mu}_0. \end{aligned} \quad (3.39)$$

In this system of equations $\frac{\partial \mathbf{t}}{\partial \mathbf{V}}$ and $\frac{\partial \mu}{\partial \mathbf{V}}$ refer to the Jacobians of the chemical potential ($\boldsymbol{\mu}$) and hopping energy ($\boldsymbol{t}$) values. These Jacobians are found most efficiently by defining perturbations to the Hamiltonian using the potential Green's functions, see section 3.1 and 3.3. $\mathbf{V}_{\text{tuned}}$ refers to the tuned gate voltages of the non-disordered device. $\boldsymbol{\mu}_0$ and $\boldsymbol{t}_0$ are the target effective Hamiltonian values, while $\boldsymbol{\mu}$ and $\boldsymbol{t}$ are the effective Hamiltonian values of the disordered device at the old tuning point $\mathbf{V}_{\text{tuned}}$. The system is solved for $\mathbf{V}$, the updated tuning point of the disordered device. With this perturbative correction approach to updating the tuning point, we do not need to tune the gate voltages of each disordered geometry. This greatly improves computational efficiency, with only a slight accuracy sacrifice. The difference in these tuning points for the initial geometry with three possible one-pixel disorders can be found in figure 12. A more tuning point comparison for all possible one-pixel disorders is provided in figure 11 in the form of two histograms.

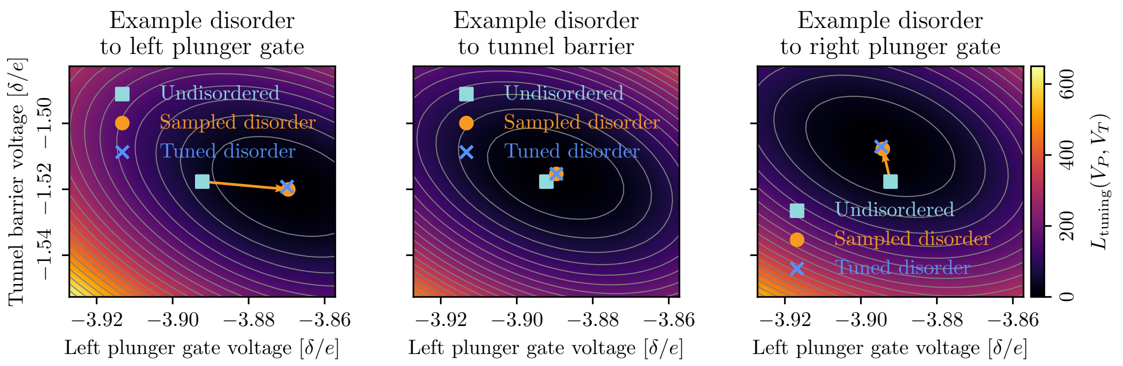


Figure 12: A comparison between the non-disordered tuning point, the perturbatively corrected tuning point, and the re-tuned tuning point for three different one-pixel disorders applied to the initial double dot device geometry

Other relevant device quantities include the gate locality and the level spacing. Optimizing a device geometry for a high gate locality leads to a device for which the virtual gates are much closer to single gate manipulations. What makes this an interesting quantity to optimize for is the fact that the process of finding virtual gates takes a lot of experimental time and data, hence making the virtual gates closer to single gate manipulations, and therefore easier to find is advantageous.

The level spacing is the energy gap between the qubit energy levels and the closest unused wavefunction, ψ_{n+1}. When this gap is too small, it enables decoherence of the qubit states. Thus, it is advantageous to design devices for which this gap is large.

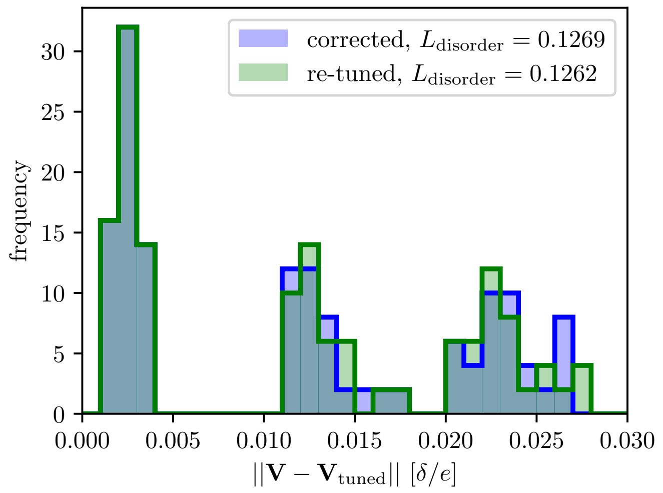


Figure 11: A comparison of the tuning points resulting from perturbatively correcting the disordered device tuning points and retuning the disordered devices. One can see that for this geometry the initial geometry and the disorder sensitivity metrics are very similar. The similarity of these two approaches justifies our use of the perturbative corrections for the sake of computational efficiency.

We formulated loss terms for

these quantities;

$$L_{\frac{\partial\mu}{\partial V}}(\mathbf{V}) = \sum_{j=1}^{n_{\text{dots}}} \frac{\sum_{i=1}^{n_{\text{gates}}-1} \frac{\partial\mu_j}{\partial V_{T,i}}^{2} + \sum_{i=1}^{n_{\text{gates}}} \frac{\partial\mu_j}{\partial V_{P,i}}^{2}}{\frac{\partial\mu_j}{\partial V_{P,j}}^{2}} \tag{3.40}$$

$$L_{\frac{\partial t}{\partial V}}(\mathbf{V}) = \sum_{j=1}^{n_{\text{dots}}-1} \frac{\sum_{i=1}^{n_{\text{gates}}} \frac{\partial t_j}{\partial V_{P,i}}^{2} + \sum_{i=1}^{n_{\text{gates}}-1} \frac{\partial t_j}{\partial V_{T,i}}^{2}}{\frac{\partial t_j}{\partial V_{T,j}}^{2}} \tag{3.41}$$

$$L_{dE}(\mathbf{V}) = \frac{t_0}{E_{n_{\text{dots}}+1} - \frac{1}{n_{\text{dots}}}\sum_{i=1}^{n_{\text{dots}}} E_i}. \tag{3.42}$$

Here V_D, V_P, and V_T refer to the depletion gate voltage, the plunger gate voltages, and the tunnel barrier voltages respectively. In the locality loss terms, $L_{\frac{\partial\mu}{\partial V}}(\mathbf{V})$ and $L_{\frac{\partial t}{\partial V}}(\mathbf{V}$, the interactions between gates their corresponding parameter are in the denominator. The effects of all gates on that parameter are summed in the numerator. In the ideal case where every parameter is only affected by its corresponding plunger or tunnel gate, we would see that $L_{\frac{\partial\mu}{\partial V}}(\mathbf{V}) = n_{\text{dots}}$ and $L_{\frac{\partial t}{\partial V}}(\mathbf{V}) = n_{\text{dots}} - 1$.

4. The geometry optimization algorithm

4.1. Algorithm overview

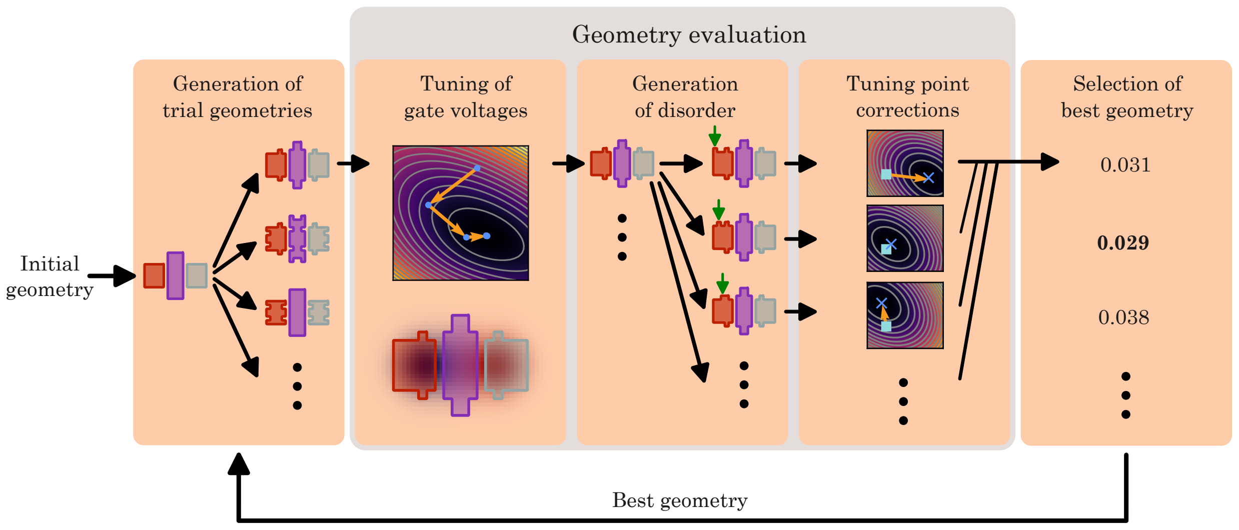


Figure 13: A diagram showing the geometry generation and geometry evalution steps. The substeps of the geometry evaluation for one device are shown. Note that for each trial geometry the evaluation is repeated.

Our geometry optimization algorithm, which is largely based on greedy optimization, consists of two essential parts: In the first place, new geometries need to be generated from the current geometry. Proceedingly, these geometries need to be evaluated in order to find the 'best' one. This 'best' geometry is then used as the initial geometry in the next iteration of the algorithm. An overview of the steps of the algorithm is given in figure 13.

This generation-evaluation cycle is repeated until a certain number of iterations is reached. Alternatively, one could implement a stopping criterion based on the evaluation loss value.

Our geometry optimization approach matches the experimental constraints. When one would optimize a device design experimentally, one would first choose a device geometry to test. One then manufactures it, and then one tunes the gate voltages to get the device to a certain operational regime. Our algorithm is based on the same principles but can test many more geometries due to the efficient nature of the device model.

4.2. Generation of new gate geometries

The first step of each iteration of the algorithm is to generate k connected gate geometries from the initial geometry. We do this by identifying the inside and outside boundary of each gate (see figure 9), and randomly selecting pixels in the outside boundary to become part of the gate, and in the inside boundary to be part of the vacuum. This way, the new geometries will be similar to the initial geometry, albeit with slight boundary changes. An example can be found in figure 14. An important hyperparameter is the probability of this Bernoulli distribution, which can be set higher or lower to generate new geometries that are less or more similar to the initial geometry.

This approach mimics a gradient descent algorithm, where one changes the target parameter in the direction which provides the largest improvement. However, it would be too computationally expensive to investigate all the possible changes that can be made to the geometry boundary. Therefore, this space of boundary changes is sampled with the stochastic approach outlined above. This still provides information on which 'direction' of geometry change improves the device.

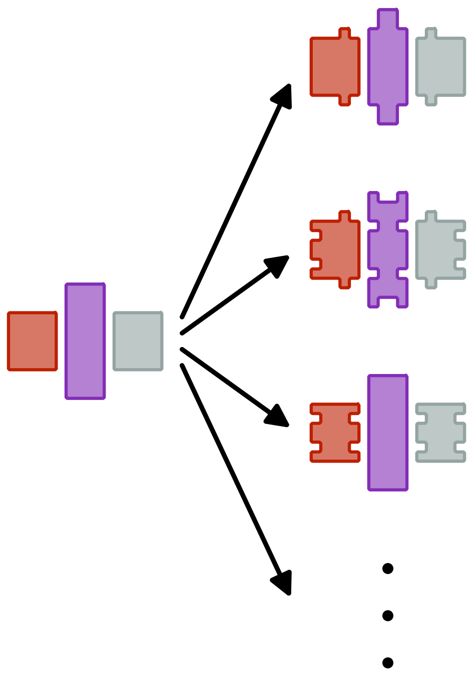


Figure 14: A schematic view of how multiple trial geometries are generated from some initial geometry. Note how pixels are added and removed from the gate boundaries while keeping the entire gate shape connected.

It is important to keep the gate electrodes connected since the voltage of an unconnected gate cannot be controlled. Therefore, we implement an additional constraint in the generation of new geometries; after modifying the boundary of each gate, a connectivity check is done. If the geometry is not connected the gate modification is discarded and reattempted. If the gate is connected the next geometry will be modified. While this reattempting strategy is not the most efficient, the computation time to generate new geometries is negligible compared to the potential solving and voltage tuning steps of the device model. Hence further optimization is not strictly needed.

There is another option included in the algorithm code; whether to have the gate connect to a wire. As devices in the lab will need wires connecting their gates to control electronics, one can imagine that it is informative to be able to know the optimal device layout including wire connectivity constraints. To ensure wire connectivity, we set one voxel on the edge of the optimization region to permanently be part of a certain gate. As long as the initial geometries are connected to this boundary pixel, the connectivity constraint will ensure that the geometries remain connected to their boundary wire voxel. Aside from defining pixels on the boundary of the optimization region to be part of the wire, we add additional leads connecting to the optimization region in the device model. An example of a simple device with such wires is plotted in figure 15

4.3. Evaluation of new geometries

To tune the gate voltages of the device with a certain geometry, we use a gradient-based minimizer on the gate voltages to get the device's effective Hamiltonian close to target values. This full procedure is outlined in subsection 3.4. The gate voltage tuning is done for all the geometries that were generated from the initial geometry. The resulting tuned gate voltages are stored, as they will be used in the disorder sensitivity and gate locality evaluation. Moreover, the gate voltages of the 'best' device will be used as the initial point for the voltage tuning procedure of all devices in the next iteration of the algorithm.

Once we have tuned devices for each geometry, we iterate over all the possible one-pixel disorders for each geometry, as outlined in subsection 3.5. The tuning points of every one-pixel disorder are aggregated into a measure of disorder sensitivity using equation (3.37). This disorder evaluation is done for each geometry generated in this iteration, so we can compare the disorder sensitivity of different device geometries.

Aside from optimizing for disorder sensitivity, it is relevant to be able to maximize the level spacing of a device and to maximize the gate locality. These quantities are calculated using the tuning point of the non-disordered geometry. The gate locality and level spacing loss are then calculated using (3.40), (3.41) and (3.42).

The disorder sensitivity loss and any other loss terms for optimization targets are then summed. Whichever geometry has the lowest total loss value will be selected to be the initial geometry in the next iteration of the algorithm.

4.4. Parallelization of the algorithm

The evaluation of the geometries is relatively the most time-consuming step of the algorithm. That is the case since the evaluation requires tuning and disorder sampling k different geometries for each iteration. Conveniently, these k different evaluation processes lend themself well to being parallelized.

To enable the parallel evaluation of different geometries, each compute node used will need a factorization of the matrix used to solve for the potential, see equation (3.12). This matrix will have to be based on a certain base geometry but can be used with Woodbury updates (equation (3.13)) to solve for the potential of other geometries. Since this factorization is a large object, it is not feasible to send it to all the nodes. It is therefore calculated separately on each compute node. It can either just be calculated in the initial iteration for the initial geometry, or updated after some number of iterations to match the current best geometry. We have found that resetting the geometry after each iteration is the fastest, as the Woodbury updates are more efficient when the geometry they solve for does not differ much from the geometry the factorization was based on.

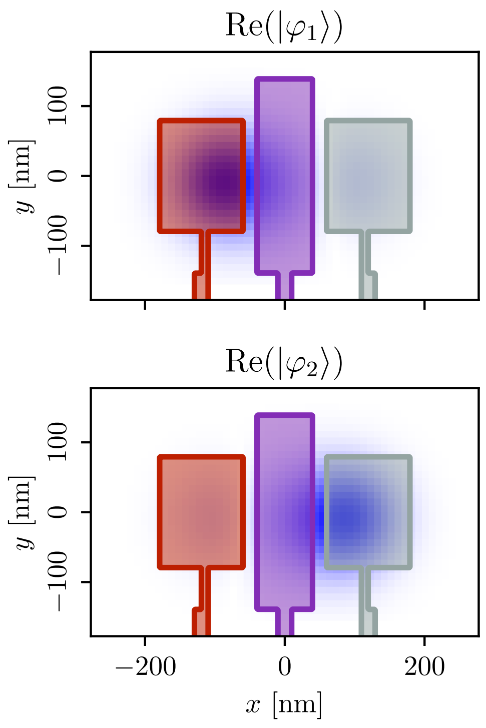


Figure 15: Plots of the initial device when the wire connectivity contraints are applied. The gate geometries are shown as the oulined shape plots. The tuned wavefunctions for $\mu_1 = \mu_2 = -1.5\ \delta$ and $t = 0.005\ \delta$ are shown as the blue image plots.

The k different trial geometries are scattered to different computational nodes, ensuring an even distribution of geometries per worker node. In our case, we chose to use k different cores of the hpc05 cluster, giving each core one geometry to evaluate. The cores proceed to calculate the potential, tune the device, and calculate the disorder sensitivity, as outlined in section 3. Afterward, the resulting loss values are sent back to the main computational node to be compared, and the best geometry is selected.

5. Example applications of the algorithm

5.1. The characteristics of the ‘naive’ initial double dot device geometry

For the geometry optimization, we use a ‘naive’ device geometry design as the initial geometry. For the double dot device, this initial geometry consists of two rectangular gate electrodes for the plunger gates, and a more narrow but taller rectangular tunnel barrier, as can be seen in figure 16. The idea behind this initial design is that this is akin to a simple double dot device used in experimental settings.

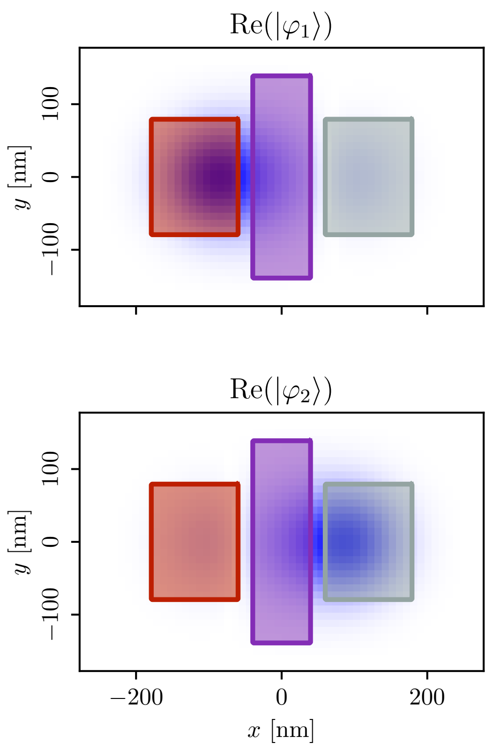


Figure 16: The geometry of the ‘naive’ gate geometry design. When the gate voltages of this device are tuned for $t = 0.005\ \delta$ and $\mu_1 = \mu_2 = -1.5\ \delta$, it has the above wavefunctions as its maximally localized basis.

We tuned the initial geometry to the effective Hamiltonian values of $t_0 = 0.005\ \delta$ and $\mu_1 = \mu_2 = -1.5\ \delta$. This corresponds to 28.4 μeV and −8.53 meV. The t_0 value was chosen to be less than 1 δ since a coupling of 1 times the energy scale would correspond to the hopping between two neighboring sites in the tight-binding model. Consequently, the model would produce results that are only 1 lattice spacing apart, at which point the model would not be accurate. We found that for μ values that are too small ($0 > \mu > 1$) the device would simply tune to produce a shallow potential well that does not confine the dots. For very large μ values ($\mu << 1$) the dots become very small since the potential well needs to be very sharp to house wavefunctions with such low eigen energies. Hence we chose $\mu - 1.5\ \delta$. The maximally localized wavefunctions of the initial device tuned to $t = 0.005\ \delta$ and $\mu_1 = \mu_2 = -1.5\ \delta$ are provided in figure 16.

When other chemical potential values are desired, one can simply tune all the gates up or down by the desired level. This includes changing the voltage of the depletion gate, which was fixed at 0 V for the sake of limiting the number of free variables in the tuning minimization problem. For instance, if one would like the same wavefunctions as seen in figure 16 but for $\mu = 0\ \delta$, one can simply increase all gate voltages by 1.5 times the voltage scale defined by the lattice spacing.

When tuned, the initial geometry produces a device with an effective Hamiltonian that matches the target. More importantly, however, we can obtain its disorder sensitivity with the procedure described in subsection 3.5. This allows us to see if our algorithm can produce device geometries for which this characteristic is lower. The disorder sensitivity, along with the level spacing and gate locality of this initial device is provided in table 1.

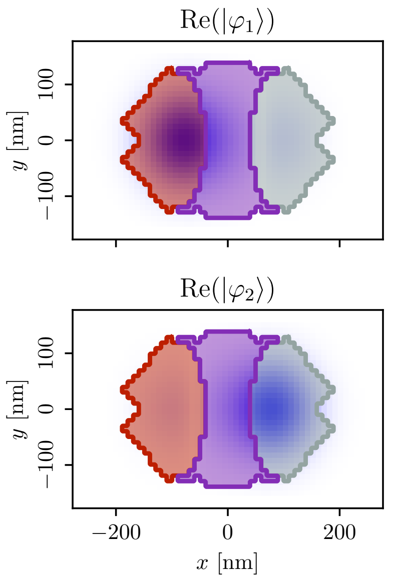
Table 1: The characteristic values of a device with the intial gate geometries, as seen in figure 16. This device has been tuned to $\mu_1 = \mu_2 = -1.5\ \delta$ and $t = 0.005\ \delta$.

Variable	**Value**
$t\ [\delta]$	0.0050
$\mu_1, \mu_2\ [\delta]$	-1.5000
$dE\ [\delta]$	0.0702
L_{dE}	0.0713
L_{disorder}	0.0307
$\frac{\partial t}{\partial \boldsymbol{V}}\ [e]$	$\begin{pmatrix} 0.0044 & 0.0062 & 0.0062 & -0.0168 \end{pmatrix}$
$\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{V}}\ [e]$	$\begin{pmatrix} 0.3742 & 0.3274 & 0.0756 & 0.2227 \\ 0.3742 & 0.0756 & 0.3274 & 0.2227 \end{pmatrix}$
$\boldsymbol{V} = \begin{pmatrix} V_D & V_{P,1} & V_{P,2} & V_T \end{pmatrix}^T\ [\delta/e]$	$\begin{pmatrix} 0.000 & -2.847 & -2.847 & -1.818 \end{pmatrix}^T$

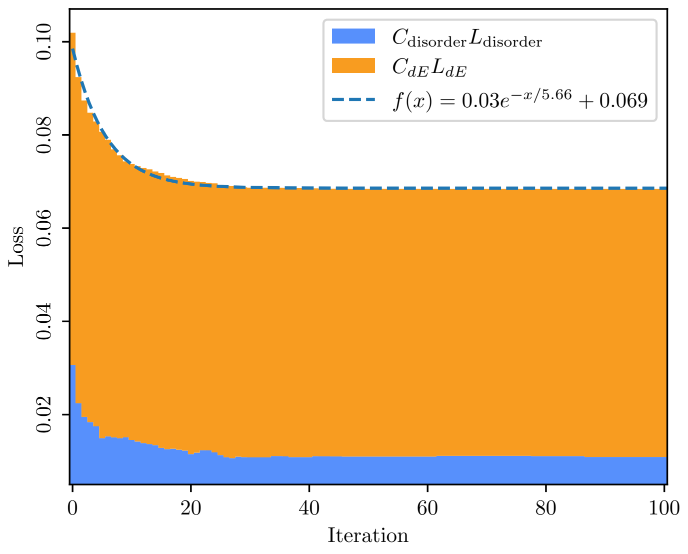
5.2. Geometry optimization results for double dot devices

5.2.1. Optimization for disorder sensitivity without connecting wires

We have applied our algorithm to the double dot geometry for 100 iterations, using the weights $C_{dE} = 1$ and $C_{\text{disorder}} = 1$. This yields the geometry with maximally localized wavefunctions as seen in figure 17a. The convergence of the loss values per iteration is plotted in figure 17b.



(17a) The resulting device geometry after 100 iterations of the geometry optimization algorithm for $k = 100$.



(17b) The convergence of the loss components $C_{dE}L_{dE}$ and $C_{\text{disorder}}L_{\text{disorder}}$ during the optimization algorithm. The exact initial and final values are provided in table 2.

In figure 17a one can observe that the plunger gates no longer assume square shapes, but instead roughly resemble cardioids. Moreover, the final plunger gate geometries have become larger than the initial plunger gates. The tunnel barrier in the middle of the final device has a shape that is narrow in the middle and grows wider further from the center. It is larger compared to its initial shape.

From the loss convergence data presented in figure 17b, it becomes evident that the disorder loss decreases 3 fold during the 100 iterations of the algorithm. In the meantime, the level spacing loss remains roughly constant. The largest improvement in disorder sensitivity happens in the first iteration, suggesting that the pixels changed within that iteration have a large impact on the disorder sensitivity. This first iteration geometry is plotted in figure 16. The exact loss values are provided in table 2.

Table 2: The initial, first iteration and final loss component values of the geometry optimization given in figure 17b. Note that the disorder sensitivity of the final geometry is three times less than the intial geometry.

	Initial Geometry	First iteration Geometry	Final Geometry
$C_{\text{disorder}} L_{\text{disorder}}$	0.0307	0.0224	0.0109
$C_{dE} L_{dE}$	0.0713	0.0699	0.0575

As we aim to demonstrate that the disorder sensitivity has been improved by our geometry optimization process, we compared the voltage deviations from the tuning point for the final device to the initial device. This comparison is illustrated in the histogram in figure 18. The histogram for the final geometry is closer to zero. This demonstrates the fact that the tuning point of the final device is less sensitive to shape disorder.

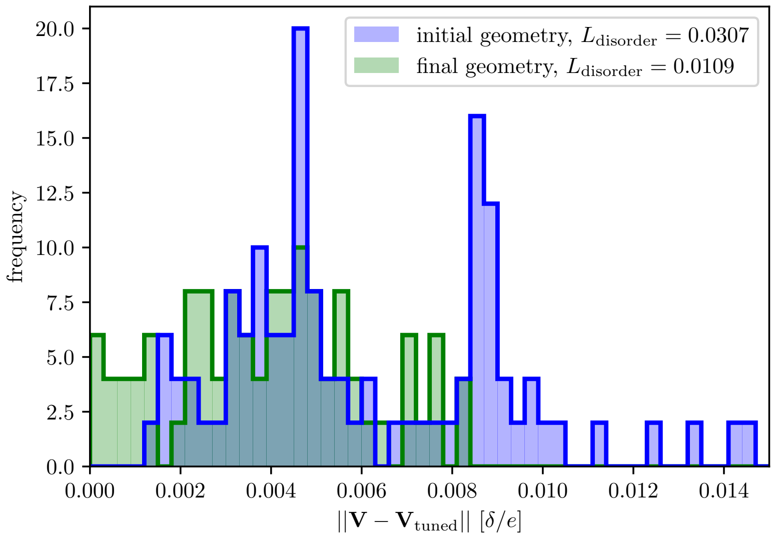


Figure 18: A comparison of how disorder affects the tuning point of the initial geometry and the final geometry. The histogram shows that the spread of tuning point deviations for the final geometry is smaller than that of the initial geometry.

In figure 19 and table 2 it is clear that the level spacing loss term ($C_{dE} L_{dE}$) does not decrease much compared to the disorder sensitivity. This fact matches our intuition; devices with a high level spacing require small dots. Small dot wavefunctions require narrow potential wells. To generate a narrow potential well, one needs small gate electrodes. Unfortunately, small gate electrodes are more sensitive to disorder. This is due to the fact that the change in tuning point caused by a disordered pixel is more pronounced when the total gate area is smaller.

Hence, since we are optimizing for disorder sensitivity and level spacing simultaneously, the algorithm approaches a balance between the two. From that point, it is unable to reduce the level spacing loss term without sacrificing disorder sensitivity.

To demonstrate that our algorithm works for different target tradeoffs between disorder sensitivity and level spacing, we have run the same model for $C_{\text{disorder}} = 1$, and $C_{dE} = 0.1, 1, 10$, and 100. The resulting geometries and their maximally localized wavefunctions are plotted in figure 19.

This plot illustrates the fact that when a higher weight is given to level spacing during the optimization, the wavefunctions become smaller. The gates that induce the potential wells that house these wavefunctions are therefore also smaller. In these results, it can also be seen that an optimization without level spacing terms produces a geometry that fills nearly the entire region allocated to have its pixels changed during the optimization process.

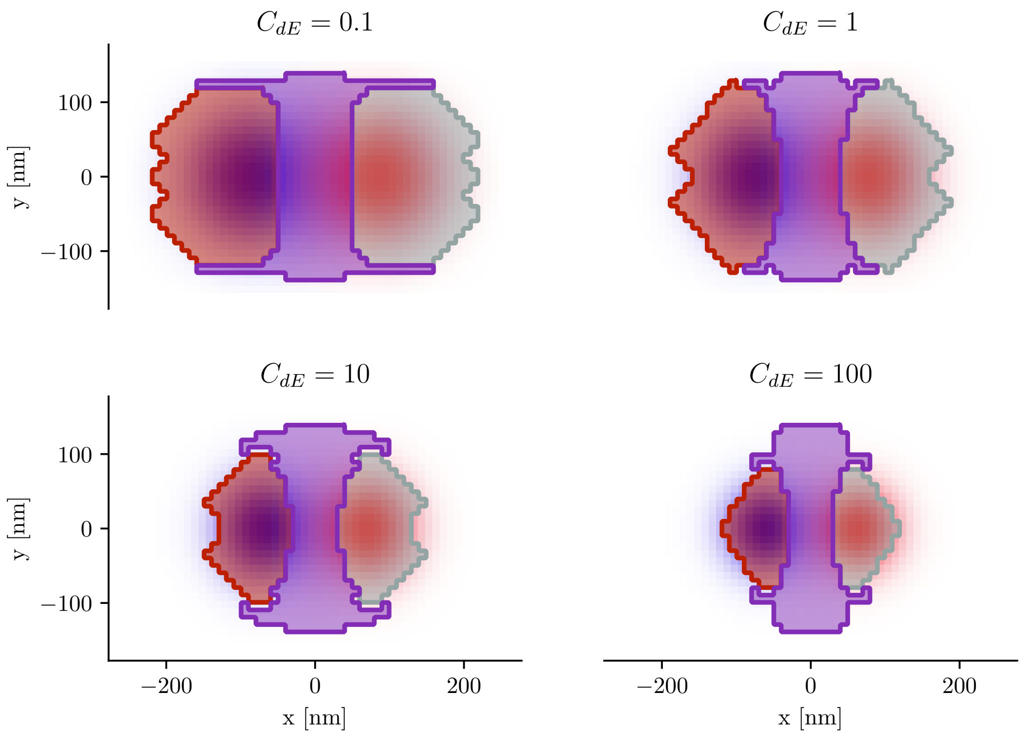
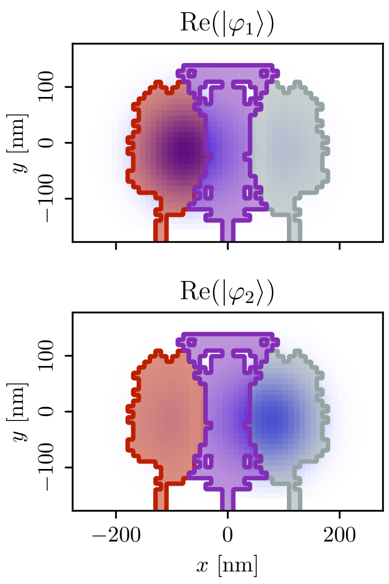


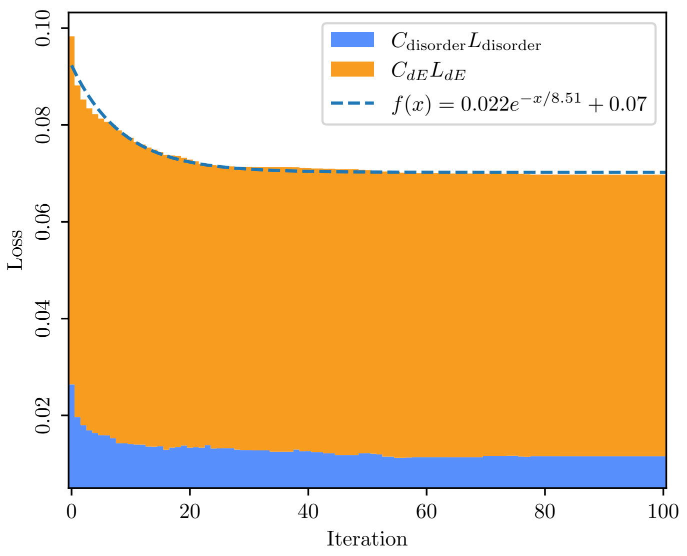
Figure 19: The results of 100 iterations of the geometry optimization algorithm for various C_{dE} weights and $C_{\text{disorder}} = 1$. The geometries (outlined shapes) and their maximally localized wavefunctions at their tuning point (red and blue image plots) are shown. Note that assigning a larger weight to the level spacing causes the geometry optimization to converge to devices with a smaller pitch.

5.2.2. Optimization for disorder sensitivity with connecting wires

As discussed in section 4.2, we are able to constrain our geometries to have connecting wires. This will give more experimentally relevant results, as experimental devices need wires attached to the gate electrodes to control the voltage levels. Hence, we ran our geometry optimization algorithm with this constraint. This was repeated for tuning variables $\mu_1 = \mu_2 = -1.5\ \delta$, $t = 0.005\ \delta$and for loss component weights $C_{dE} = 1$ and $C_{\text{disorder}} = 1$. The resulting geometry after 100 iterations is shown in figure 20a, and the loss convergence during the optimization is plotted in figure 20b.



(20a) The resulting device geometry after 100 iterations of the geometry optimization algorithm for $k = 100$ with wire connectivity contraints. Notice that the geometries have a similar size and general shape to those without the wire connectivity contrains, as seen in figure 17a



(20b) The convergence of the loss components $C_{dE}L_{dE}$ and $C_{\mathrm{disorder}}L_{\mathrm{disorder}}$ during the optimization algorithm with wire connectivity constraints. Table 3 has the values of these loss terms for the intial and final device, and compares them to the case without the wire constraints.

From figure 20a and figure 17a it becomes clear that the resulting geometry with wires does not differ much from the one without wires. This validates our intuition that running the algorithm without the wires suffices when exploring geometry parameter spaces. Once a rough layout has been found, a more detailed analysis with wire constraints can be performed.

When regarding figure 20b and figure 17b, one can notice that the convergence of the optimization with wire constraints is slower than the one without. When fitting a function of the form $f(x) = ae^{-\frac{x}{\tau}} + b$ to the loss convergence plots, we see that the optimization without the wire constraints has a decay of $\tau = 5.663$ iterations, versus a decay of $\tau = 8.512$ iterations for the optimization with wires. The time to convergence is more than 1.5 times slower. This is due to the fact that adding the wires to the gates breaks the symmetry of the device in the y direction. Before we were able to use the symmetry of the device without wires to reduce the parameter space of the geometry optimization problem. Now we must allow for geometries that are not symmetric in the y direction. Due to this, we must let all 672 pixels in the geometry parameter space vary freely, instead of only varying the top 336 pixels and mirroring this shape to the bottom half. This larger parameter space naturally causes the convergence to be slower.

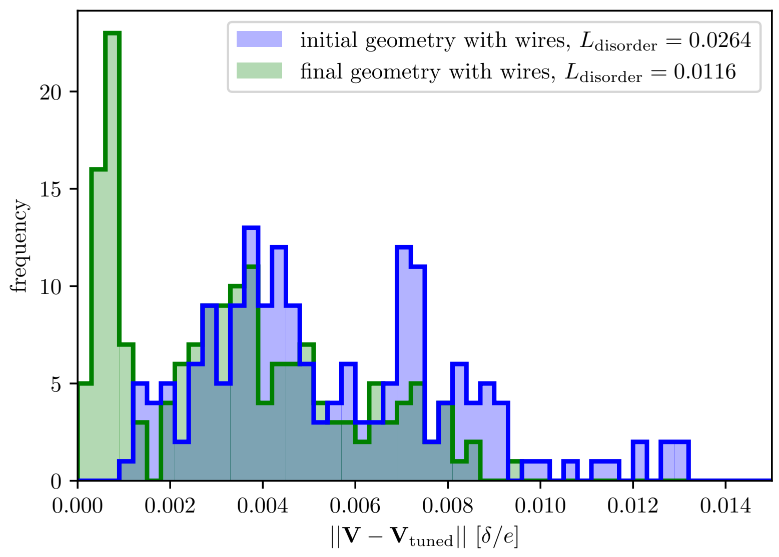


Figure 21: Histograms of the tuning point voltage deviations caused by disorder. These show that the final device of the optimization with wire contraints is less than a third as sensitive to gate shape disorder as the intial device.

In table 2 one can see that the device resulting from the optimization of a device with wires is slightly more susceptible to disorder than the one without wires. The same goes for the level spacing, which also has a slightly higher loss for the device with the wire attached. This makes sense, given that the extra constraint of having an additional wire limits which solutions are possible. Thus, the minima of devices with wire constraints will be worse compared to those of devices without wires.

Table 3: The loss component values of the intial and final devices for the geometry optimization with and without wire contrains. The disorder sensitivity for the device with wires is slightly higher than the one without these contraints.

	Initial geometry without wires	Initial geometry with wires	Final geometry without wires	Final geometry with wires
$C_{\text{disorder}}L_{\text{disorder}}$	0.0307	0.0264	0.0109	0.0116
$C_{dE}L_{dE}$	0.0713	0.0714	0.0575	0.0582

In table 2 it stands out that the initial geometry with wires is less sensitive to disorder than the one without wires. This is remarkable since we would expect that the added wire makes it even more sensitive to disorder, as it introduces more possible sites for disorder. A possible explanation can be found by considering the gate voltage tuning points of these initial devices. The development of the gate voltage tuning points is plotted in figure 22. Note that the initial plunger gate voltage for the device with wires is more than 0.1 higher than for the one without. We hypothesize that since the device with wires is already tuned to produce a dot even though there is an appendage to the plunger gate, adding additional pixels to the plunger gate will not affect the tuning point as much as when the plunger gate is a simple rectangle. The effect on the tunnel barrier is less, as the added wire to the tunnel barrier gate is further away from the dot wavefunctions.

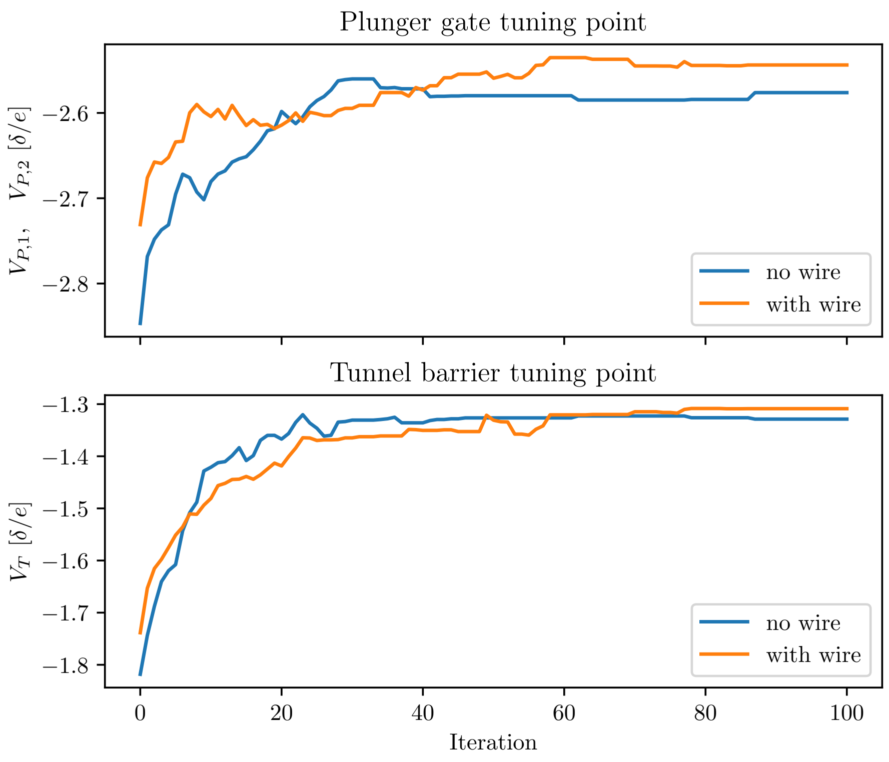


Figure 22: Plots showing the evolution of the volage components of the tuning point as the geometry optimization converges. The plunger gate voltages of the device with wire contrains are intially much closer to the values they assume for the converged device. This possibly explains why the inital device with wire contraints is less sensitive to disorder, as seen in table 3

.

In figure 23 the geometries resulting from optimization with wire constraints for a range of C_{disorder} and C_{dE} weights are plotted. The shapes and sizes of these geometries are similar to those without the wire constraints, see figure 19. This validates our approach of neglecting the wire constraints when the algorithm is used for more exploratory purposes, with the aim of finding the rough shapes and separations of gates. However, when the aim is to apply the algorithm to optimize finer details, a more accurate optimization with the wire constraint is needed. This is illustrated by the fact that the finer details of the geometries do differ for the two cases. Namely, in the wire case, the horizontal lines of the tunnel barriers seen in the first two geometries of figure 19 are not present. In the meantime, there are other details such as 'holes' in the tunnel barrier gate in the optimal geometries with wire constraints that are not present in the case without the wires.

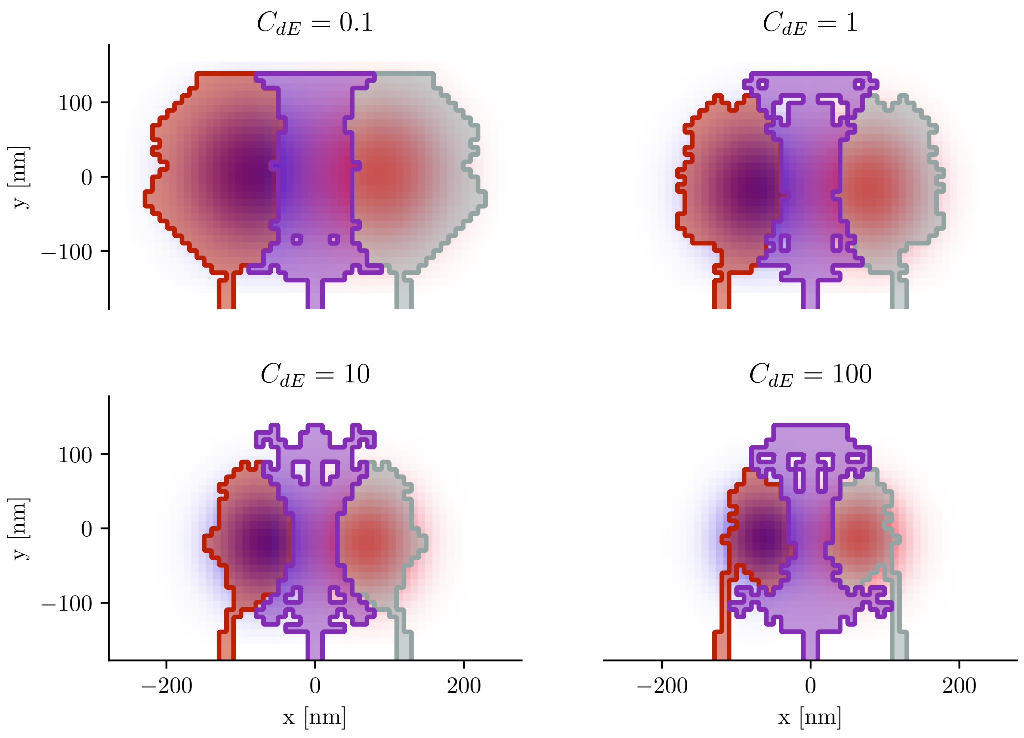


Figure 23: The converged geometries after 100 iterations of the geometry optimization algorithm with wire contraints for different C_{dE} weight settings and $C_{\text{disorder}} = 1$. Similar to the case without wire contraints, figure 23, for higher disorder sensitivity weights the device gates end up smaller and closer together.

5.2.3. Optimization for gate locality without disorder sensitivity

As mentioned in subsection 3.5, we can choose a maximal gate locality as the optimization target. This way the interaction between a gate and its corresponding dot will be maximized, while the effects of the other gates on that dot will be minimized. There are two components to this; maximizing the plunger gate and chemical potential (μ) interaction, defined by equation (3.40), and maximizing the tunnel barriers and the hopping energy (t) interaction, given in equation 3.41. A different balance in the weights of these two terms will prioritize different aspects of the geometries. Therefore, we have run the geometry optimization for a range of these weights, while setting the weight of the disorder sensitivity and level spacing losses to zero; $C_{\text{disorder}} = 0$, $C_{dE} = 0$.

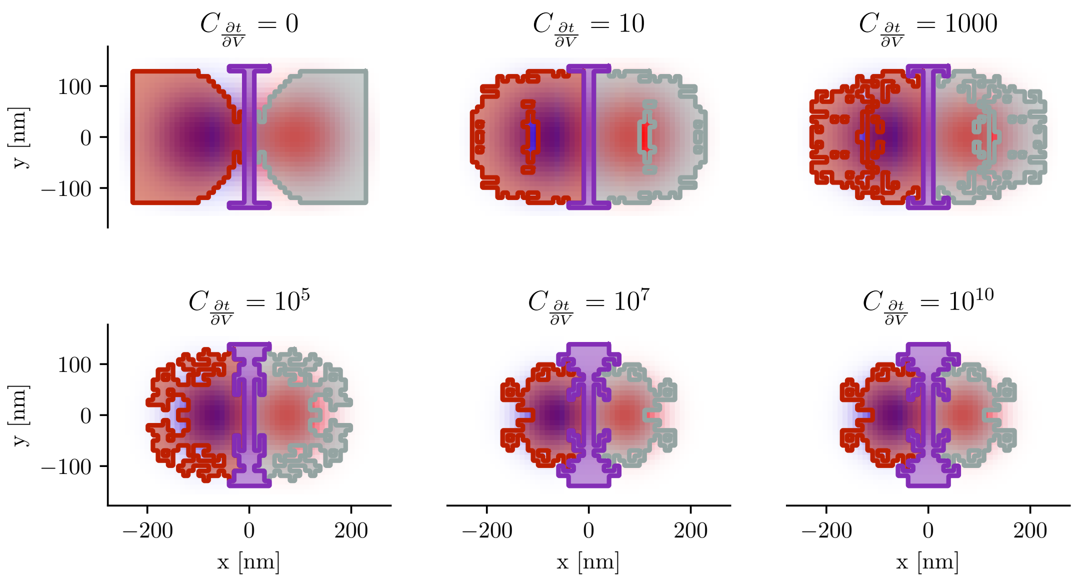


Figure 24: The converged geometries for weights $C_{\text{disorder}} = 0$, $C_{\frac{\partial \mu}{\partial V}} = 1$ and $C_{\frac{\partial t}{\partial V}} = 0, 10, 1000, 10^5, 10^7$ and 10^{10}. Notice the jagged, coral like shapes of the $C_{\frac{\partial t}{\partial V}} = 1000$ and 10^5 geometries. We see this jagged nature reflected in the results because we have set the disorder sensitivity weight to zero.

In figure 24 the resulting geometries for different balances in the weights of the two gate locality loss functions are shown. The chosen weights are $C_{\frac{\partial \mu}{\partial V}} = 1$ and $C_{\frac{\partial t}{\partial V}} = 0, 10, 1000, 10^5, 10^7$ and 10^{10}. Note that on the left side of the figure, where the weight of the tunnel barrier gate locality is small, the tunnel barrier is as narrow as it can be while still being connected to its leads on the top and bottom. Further to the right, the interaction of the tunnel barrier with the hopping energy between the dots is prioritized compared to the interaction of the plunger gates and the chemical potential. Consequently, the devices to the right have tunnel gates that are wider at the top and bottom. The opposite is true for the plunger gates. The geometries that are optimized with a higher weight for the plunger gate locality have larger plunger gates, which spread out towards the edges, away from the dot that they should affect minimally.

From the jagged geometries in figure 24, it becomes clear that not including disorder sensitivity as an optimization target ($C_{\text{disorder}} = 0$) allows the algorithm to converge to devices that do not have smooth boundaries. Naturally, these geometries are very sensitive to shape disorder. This highlights the need for optimizing for shape disorder sensitivity when performing such a geometry optimization.

5.2.4. Optimization for combinations of disorder sensitivity, gate locality, and level spacing

In previous paragraphs, we have shown the results of applying the algorithm to optimize disorder sensitivity with level spacing and gate locality with level spacing. These targets can also be combined. The results for various weights $C_{\text{disorder}} = 1$, $C_{\frac{\partial t}{\partial V}} = 0$, $C_{\frac{\partial \mu}{\partial V}} = 0.0001, 0.001, 0.01, 0.1$ and $C_{dE} = 0.1, 1, 10$ are plotted in figure 25.

From these geometries, it becomes evident that increasing the level spacing weight in the optimization makes for smaller devices, as was discussed in section 5.2.1. Something we see in this comparison is that increasing the weight of the plunger gate locality ($C_{\frac{\partial \mu}{\partial V}}$) causes the optimal plunger gates to be larger and more spread out near the left and right edges. Moreover, it decreases the size of the tunnel barrier gate. This aligns with our expectations: A device with larger plunger gates and a smaller tunnel barrier will have dots that are more affected by their plunger gates than the tunnel barrier. As the gate locality loss assigns a cost to the effect of the left plunger gate to the right dot and vice versa, the growth of the plunger gates will be mostly towards the edges. Away from the other dots, which the plunger should not have a large effect on. Hence, we see that the larger bulk of these plunger gates is located more to the left and right edges.

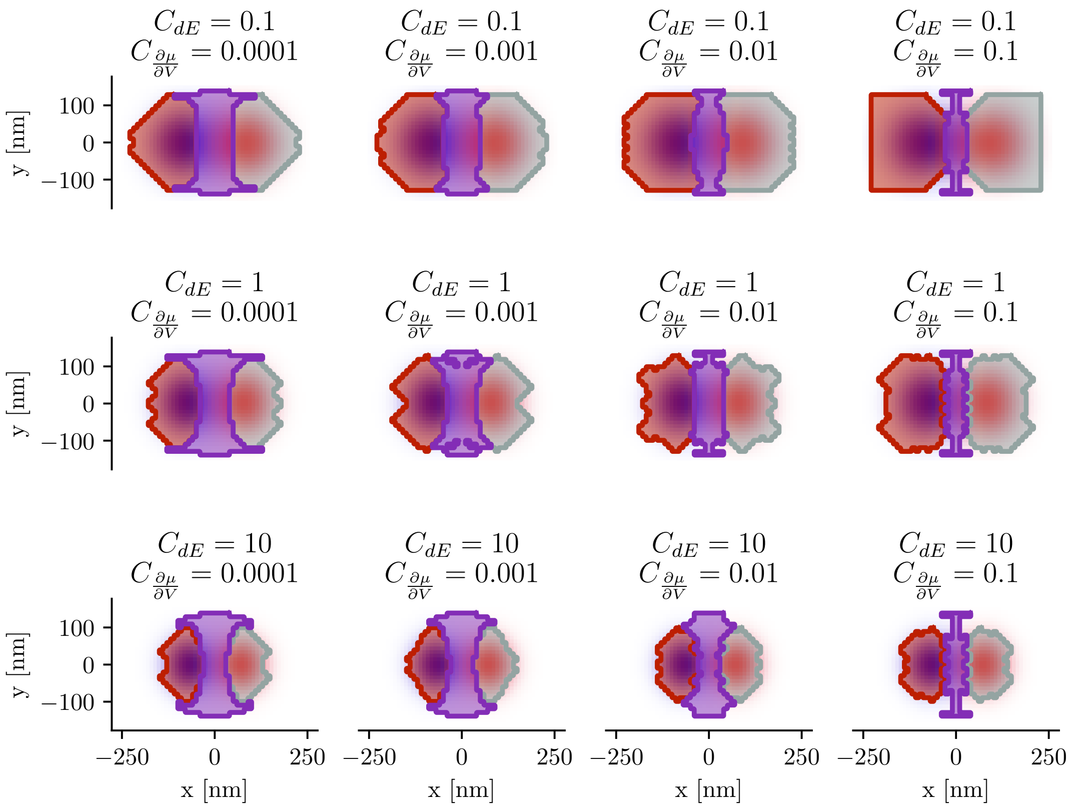


Figure 25: The converged geometries for geometry optimizations for the weights $C_{\text{disorder}} = 1, C_{\frac{\partial t}{\partial V}} = 0, C_{\frac{\partial\mu}{\partial V}} = 0.0001, 0.001, 0.01, 0.1$ and $C_{dE} = 0.1, 1, 10$.

5.3. Geometry optimization results for triple dot arrays

The results of a geometry optimization for triple dot devices for disorder sensitivity with various level spacing weights are plotted in figure 26. The convergence of their loss terms is plotted in figure 27 Just like in the double dot case, we see that when we include no level spacing constraint the geometries will fill the entire space provided. When we increase the level spacing weight, the gates and wavefunctions will become smaller. Moreover, we see that the tunnel barriers and plunger gates are moved closer to the center, meaning that the pitch of the device is smaller.

A pixel size of 20 nm was used to generate these results, for the sake of computational efficiency. Note that this makes the disorder sensitivity higher by default, as the effect of an extra 20 nm pixel is more than a 10 nm pixel. While there will be fewer sites for disorder, as the boundary consists of fewer pixels than it would for a 10 nm resolution, the fact that this device is larger, and has 5 tunable gates instead of 3 makes the disorder sensitivity higher by default. Therefore, the level spacing terms were also chosen to be higher. We see about the same size gates and disorder loss component balance for $C_{dE} = 1000$ in the three-dot case as we saw for $C_{dE} = 10$ in the two-dot case (compare figure 26 to figure 19).

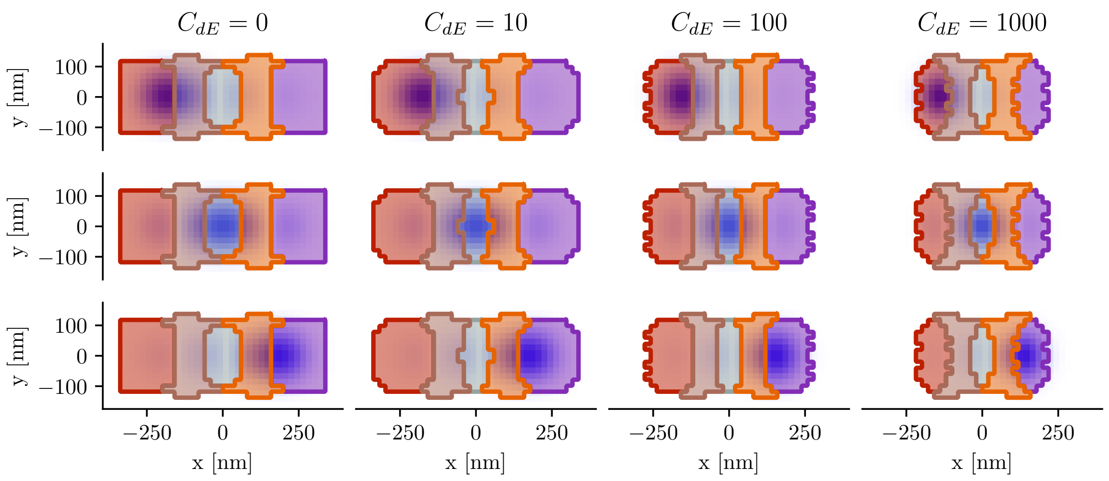


Figure 26: A comparison of the converged geometries of a triple dot device for the level spacing weights $C_{dE} = 0, 10, 100, 1000$ and $C_{\text{disorder}} = 1$. The resultion for the triple dot device voxels was 20 nm. Once more the results confirm that higher level spacing weights lead to devices with a smaller pitch and smaller dot wavefunctions.

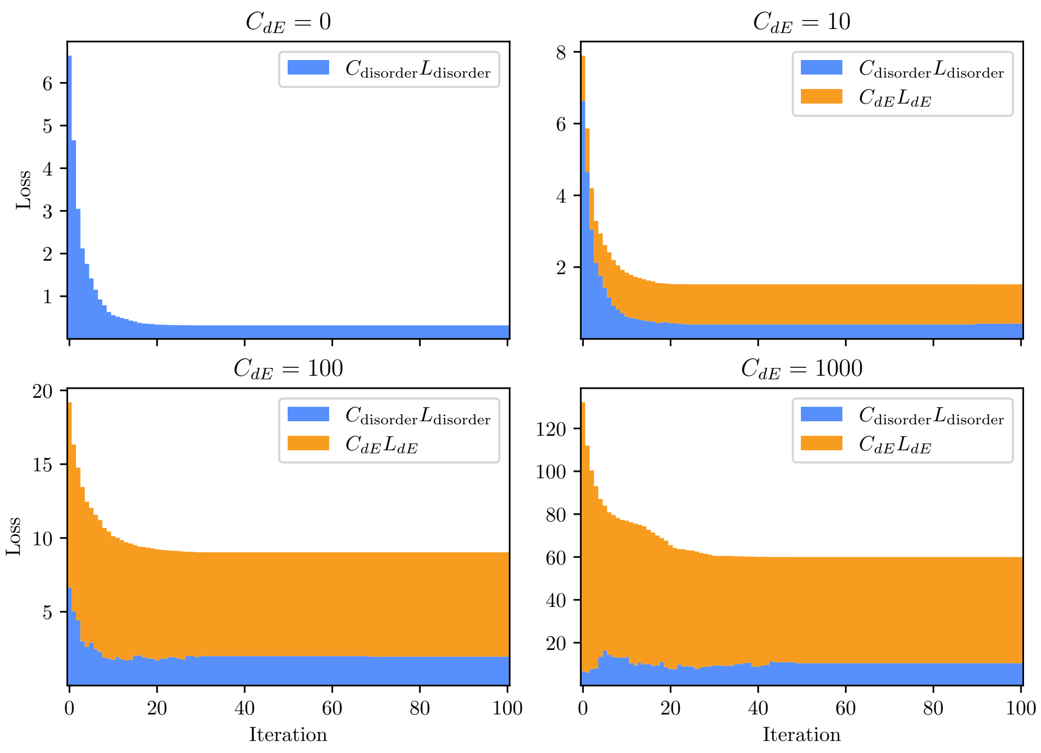


Figure 27: The convergence of the loss terms during the geometry optimization to the triple dot devices shown in figure 26. The disorder sensitivity weights for all runs was $C_{\text{disorder}} = 1$, and the level spacing weight were $C_{dE} = 0, 10, 100$ and 1000. These different weights clearly have an impact on the compostion of the loss of the final device; larger level spacing weights lead to larger weighted loss components.

5.4. Stability analysis of optimal solutions

In previous sections, we have shown that our geometry optimization algorithm improves upon the initial device geometry. However, we would like to know whether it finds local or global minima to the optimization problem. Unfortunately, as this is a discrete optimization problem, and not a continuous one we cannot take the gradient of the loss function to see if it is zero. Therefore, we have come up with an ad hoc method to investigate this convergence stability. We generated different initial geometry shapes and applied the geometry optimization algorithm to compare the final geometries the algorithm converges to.

These random initial conditions were generated using Perlin noise. In our application of this random generation, we ensured that the tunnel barrier has a higher probability of having pixels in the center of the device. Moreover, we ensured that all gates were connected by removing the smallest ‘islands’ from the shapes. Five of these random initial geometries are plotted in the top row of figure 28.

The geometries in the second row of figure 28 are the results of performing 100 geometry optimization iterations on the above initial geometries. This optimization was done using the same settings as were used to generate the geometry in figure 17a; $C_{\text{disorder}} = 1$, $C_{dE} = 1$, $\mu_0 = -1.5$ δ and $t = 0.005\ \delta$

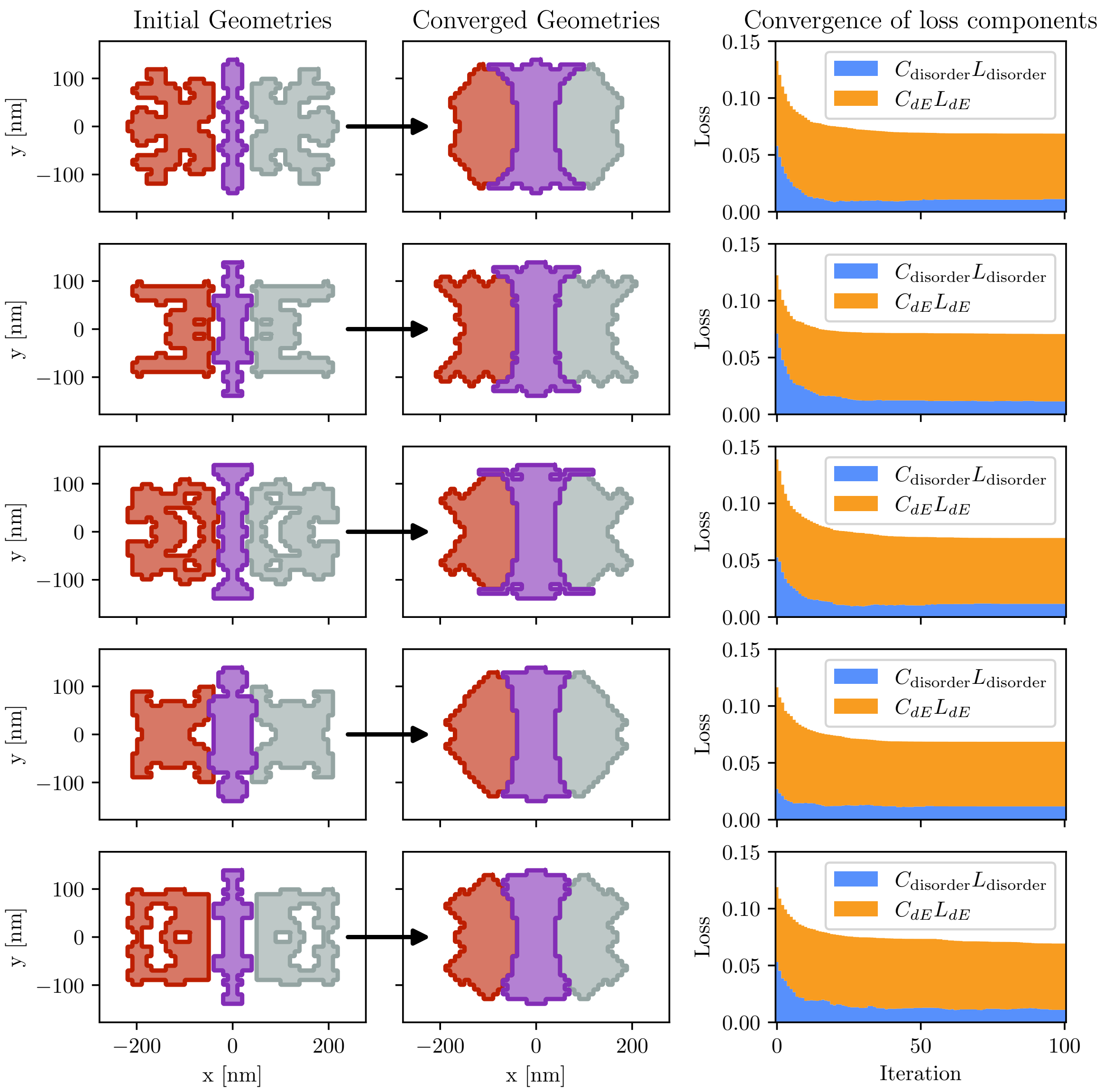


Figure 28: The randomly generated intial geometry shapes and the corresponding geometries that the optimzation algorithm converges to. Notice the similarities between the geometries near the center, while the outer edges do assume different shapes. On the right, the loss convergence for each geometry optimization have been plotted. The exact final loss values are given in table 4

From the optimized geometries in figure 28, it becomes clear that the algorithm converges consistently towards some geometrical aspects, while others differ per initial condition. Especially the ‘hourglass’ shape of the tunnel barrier is reproduced. In the center of each geometry, the boundary between the plunger gates and the tunnel barrier is nearly the same for each geometry. However, the left and right edges of the plunger gates are different. The reason for this is that the dot properties are most dependent on the gate pixels closest to the wavefunction of the dot. Therefore, the boundary between the tunnel barrier and the plunger gates in the center counts more toward the total loss than the top and bottom edges of the tunnel barrier. It also counts more than the left and right edges of the plunger gates. This explains why we see such a vast difference in the geometries at these edge regions, as opposed to the observed similarity of the center regions.

From the fact that the geometry optimization algorithm does not converge to the same geometry for each initial condition, we can conclude that it does not find a global minimum of the optimization problem after 100 iterations. This can either be due to the fact that there are many local minima near the optimal geometry, or because the stochastic way we generate new trial geometries does not exhaust the space of possible changes enough. The first option would mean that the geometries the optimizer finds are indeed optima, but that better geometries exist. For example, the second geometry in figure 28 seems to have converged, judging by the flattening of its loss convergence plot. However, its properties are worse compared to the first optimized geometry in the figure, as can be seen in table 4.

Since we do not know if the geometry the algorithm converged to is optimal from the fact that the convergence plot flattens alone, an alternative explanation could be that the geometry can be improved through changing its boundary, but that by chance that particular change has not been sampled.

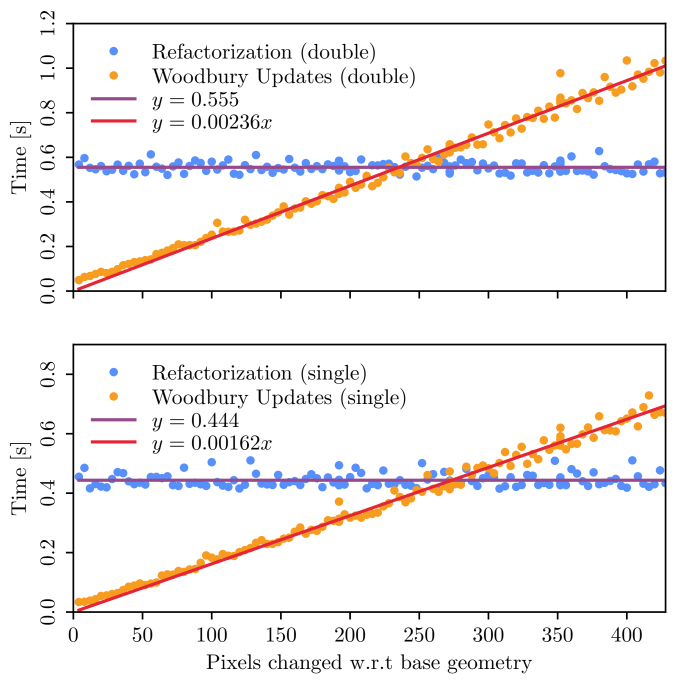
Table 4: The weighted loss component values of the converged geometries for different intial geometries, as shown in figure 28

	Geometry 1	Geometry 2	Geometry 3	Geometry 4	Geometry 5
L_{total}	0.0687	0.0707	0.0695	0.0686	0.0693
$C_{\text{disorder}}L_{\text{disorder}}$	0.0111	0.0114	0.0116	0.0117	0.0110
$C_{dE}L_{dE}$	0.0576	0.0593	0.0579	0.0569	0.0583

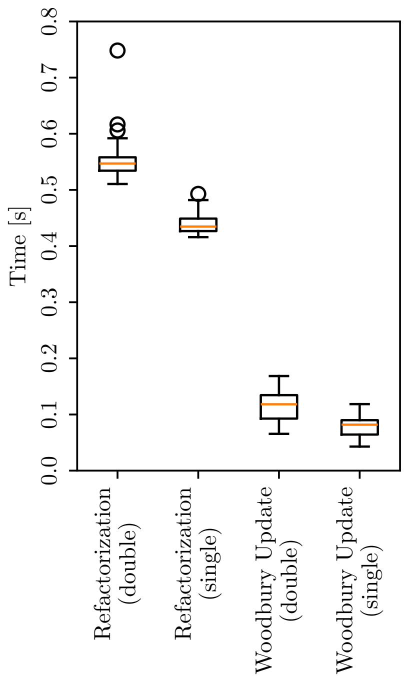
6. Performance evaluation of the algorithm

In section 3 we introduced several strategies for better computational efficiency, the most important of which are the Woodbury updating technique and the use of perturbation theory. In this section, their efficacy is demonstrated. The subproblems in which their computational times are compared are chosen to be similar to the full geometry optimization in section 5; using double dot devices with the same Voronoi cell discretization, tuning parameters, and geometry permutation parameters. This allows us to estimate how much computational efficiency we have gained in total without implementing computationally inefficient geometry optimization for the sake of comparison.

6.1. Efficacy of Woodbury updates



(29a) A demonstration of how the execution time of the Woodbury update strategy scales with the number of pixels changes w.r.t the geometry of the base factorization. Notice that after 275 ± 2 (single-point precision) or 235 ± 2 (double-point precision) factorizing the linear system for the new geometry becomes more economical than the Woodbury updating strategy.



(29b) Box plots of the computational times for solving for the potential of geometries. This is done either by refactoring the matrix of the linear system (equation (3.12)), or by applying the Woodbury Update strategy to re-use a factorization of the linear system for a base geometry (equation (3.13)). The comparison was run in both single-point and double-point precision. The mean values are 0.55 ± 0.03 s and 0.45 ± 0.03 s for the refactorizations, and 0.12 ± 0.4 s and 0.08 ± 0.02 s for the Woodbury updates.

As the Woodbury updating method relies on re-using a factorization of a base geometry to solve for the potential of different geometries, it gets more computationally intensive depending on how many pixels of the new geometry differ from the base geometry. We analyzed this for a double dot device with the same system size as used in section 5. The results are plotted in figure 29a. Here a further comparison between solving for the potential in single or double point precision is made.

From the intersection of the lines of best fit in figure 29a it becomes clear that when the new geometry differs too much from the base geometry, it is no longer more efficient to use the Woodbury updating strategy. This is the case for 261 ± 2 for single-point precision, and 223 ± 2 pixels for double-point precision.

Typically, new geometries during the first iteration of the geometry optimization differ about 33 ± 10 pixels from the initial double dot geometry (see figure 16). While this number is dependent on the shape of the base geometry and will be different for further iterations of the geometry optimization, the comparison of the execution times of both potential solvers for the first iteration is still indicative of the overall efficiency gains. In figure 29b this first iteration comparison between the compute times of refactorizing the discretization matrix and using Woodbury updates to re-use the factorization of a base geometry is shown. Here, 100 different geometries were generated from the initial geometry, and their potential was solved using a refactorization of the linear system or a Woodbury update using the factorization of the linear system of the base geometry.

6.2. Comparison of tuning methods

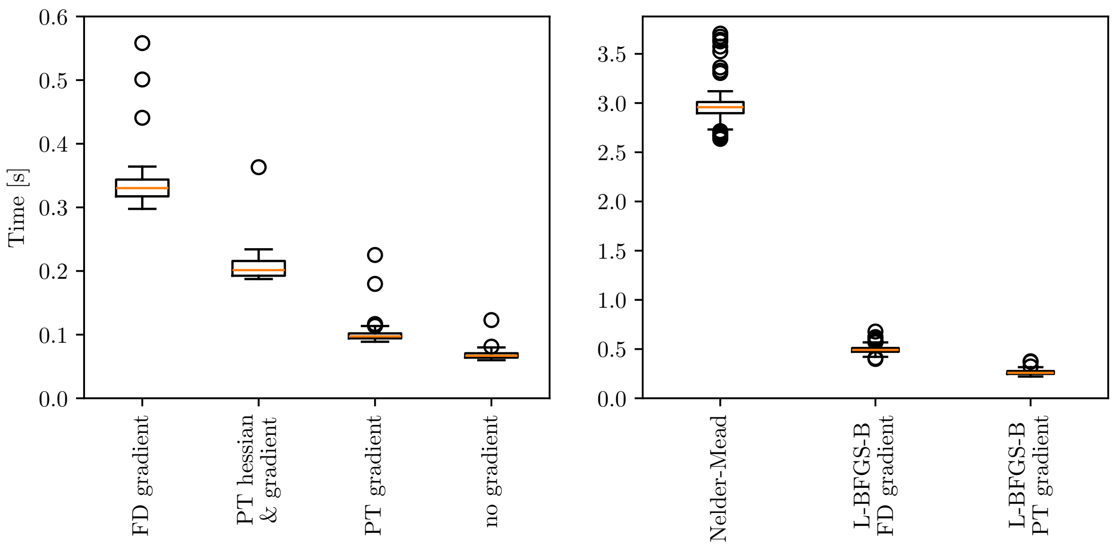


Figure 30: The right plot shows box plots of the function call times for the tuning loss function, the tuning loss function with a gradient calculated using the midpoint finite difference method (FD), and with a gradient obtained by applying Perturbation Theory (PT). The left plot shows the computational time needed to tune a device using gradient-free and gradient-based tuning methods. For the gradient-based tuning method, Limited Memory Broyden – Fletcher – Goldfarb – Shanno (L-BFGS-B) with either a midpoint finite difference gradient (FD) or a perturbation theory (PT) gradient was used. Even though the loss function call time with a perturbative gradient is longer than a gradient-free function call, the gradient-free optimization method Nelder-Mead performs almost 10 times worse than the gradient-based method using the perturbative gradient.

In subsections 3.4 and 3.5 we showed how we use perturbation theory to obtain gradients of the tuning loss function. This is much faster than using a finite difference method to obtain the gradients, as is shown in the left subfigure of figure 30. The average function call time for the loss function and its perturbation theory gradient is only 1.8 times the function call time without the gradient. If we were to use the midpoint method to obtain the gradient, this would take 4.7 times the time of a single call of the tuning function.

The right subfigure of figure 30 shows that the gradient-based Limited Memory Broyden – Fletcher – Goldfarb – Shanno algorithm (L-BFGS-B) is much faster than the gradient-free Nelder-Mead method when applied to the tuning optimization problem. Using the Perturbation Theory (PT) gradient, it only takes 0.26 ± 0.03 seconds, compared to the 0.5 ± 0.05 seconds it takes with a midpoint finite difference (FD) gradient. The gradient-free Nelder-Mead algorithm tuned the device in 3 ± 0.2 seconds.

The comparisons in figure 30 were made by generating 100 new devices from the initial double dot geometry. There new geometries were tuned to their optimum using the initial geometry tuning point as their starting voltages. The speed gain of using perturbation theory to obtain gradients would be even more pronounced for devices with more than two dots. This adds more free variables to the tuning function, which increases how many function calls are needed to tune it. Moreover, the midpoint finite difference gradient calls the loss function twice for each free variable. Due to these two effects, the performance gain of using perturbation theory gradients would be even more significant for larger devices than it already is for this double dot device example.

6.3. Speedup due to perturbative sampling of disorder

When investigating the shape disorder sensitivity of a device, we need to find the tuning points of many disordered variations of a base device. The naive approach to this is to re-tune the gate voltages for each disordered geometry. However, this tuning process is still intensive, even with the performance gains from using gradient-based optimization algorithms. The computational efficiency of this step in the algorithm has an especially large impact on the total computation time since it is repeated for each disordered geometry for each trial geometry. Using the estimates from section 6.2, when we run the algorithm for 100 trial geometries that each has about 33 possible disorder pixels, there will be about 3300 calls to these tuning point corrections. Since it is essential to speed up this process, we have implemented the first-order correction approach outlined in subsection 3.5. In this section, it is demonstrated how effective this strategy is for improving the computation time of sampling the disorder sensitivity of double dot devices.

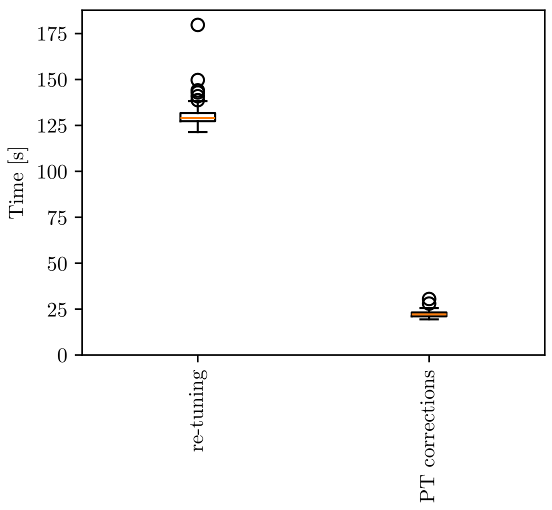


Figure 31: A comparison of the computation times associated with re-tuning to obtain the tuning points of all the one-pixel disorders, or by approximating these using first-order corrections using the gradients obtained with Perturbation Theory (PT corrections). Note that the execution time of the perturbative corrections is 23 ± 2 seconds, more than 5 times faster than the 131 ± 7 seconds it takes to re-tune all the disordered geometries.

Figure 31 shows that there is a speedup of more than 5 times when using the perturbative corrections for the gate voltages instead of re-tuning the gate voltages of the disordered geometries. The comparison was done for 100 trial geometries generated from the initial double dot geometry. For each trial geometry, all one-pixel geometries were generated, and their tuning points were found using either first-order corrections or re-tuning.

6.4. Overall computational efficiency gains

In table 5 the findings from previous subsections are summarized. The comparison in the table shows how the potentials solving steps are improved by the use of a single-point precision Woodbury Update, as opposed to re-factorizing for each geometry in double-point precision. Furthermore, the tuning of gate voltages is done more efficiently by using a gradient-based tuner and by calculating the gradients using perturbation theory, as opposed to using a gradient-free tuner. The disorder sensitivity is found by sampling the new tuning points of the roughly 33 disordered geometries of each trial geometry. This is done efficiently by using linear corrections, leveraging perturbation theory to obtain the gradients. The naive case would be to re-tune for every disordered device. In figure 31 this was done using the gradient-based tuner, but the real naive case would be to use a gradient-free tuner. Therefore, to estimate the computational time of this re-tuning for disorders with a gradient-free tuner, we multiplied the results of this comparison by how much slower gradient-free tuning is to get the value in the stared cell (*).

Using these comparisons for the algorithm substeps, we estimate that the total speedup of the optimization algorithm is about 62 times the ’naive’ unoptimized version. As the mentioned processes are the major time-consuming steps of each geometry optimization iteration, we are able to combine them to obtain an estimate of the total iteration time. We did not run the full unoptimized version of the algorithm, as it would take a lot of computational resources to reach the same conclusion as we reached now; The optimization strategies are effective. Moreover, these strategies will be even more effective the more dots and gates are simulated, as the potential solving, device tuning, and disorder sampling gains all scale with the number of gates and the system size.

Table 5: An overview of how the major steps of the geometry optimization algorithm have been sped up. An estimate of the overall efficiency gains is provided in the bottom row of the table. Note the overall speedup of 62 times, which highlights the need for such computational strategies to perform geometry optimization within a reasonable time. The starred cell (*) is a multiple of the timing results from the comparison in figure 31 with the speedup factor of perturbation gradient-based tuning. This was done to obtain an estimate of the disorder sampling time without gradient-free tuning methods.

	Calls per iteration	**Optimized call time**	**Unoptimized call time**	**Speedup factor**
Solving for the potential of trial geometries	100	0.08 ± 0.02	0.55 ± 0.03	7
Solving for the potential of disordered geometries	≈ 3300	0.04 ± 0.01	0.557 ± 0.003	14
Tuning of gate voltages	100	0.26 ± 0.03	3.0 ± 0.2	12
Finding the disorder sensitivitiy	100	22 ± 2	1510 ± 80 *	66
Estimated iteration time	1	2470 ± 20	153200 ± 800	62

Other strategies for improving computational efficiency have been implied. We implemented a ‘memory’ system so the eigenvector solver uses the last known eigenvectors as starting vectors in the diagonalization algorithm. Moreover, we made use of the fact that the device is symmetric to make use of the fact that a disordered pixel to the left plunger gate will have the same effect as the mirror image of that pixel to the right plunger gate. As these optimizations are more trivial than the ones we investigate in this section, we did not include them in the comparison. However, if these would be included the total speedup would be several times higher than 62.

7. Discussion of the used techniques and recommendations for further reseach

As the aim of this project was to demonstrate the utility of geometry optimization algorithms for quantum dot devices, several simplifications have been made. These simplifications were included because the alternatives would either be too computationally expensive for a proof of concept project, or they would take too many hours of effort to implement. Further research could explore geometry optimization algorithms that do include non-simplified models of the effects explained in this section.

An important simplification of the numerical model of the device is that we did not include electron-electron interactions. That is, the potential generated by the electrons in the 2DEG is not included in the total potential. This means that the Coulomb repulsion between the electrons in the different dots is not modeled, as well as the effect of the electrons in the dots on the charge distribution in the gate electrodes. However, as we are modeling a device with a large depletion gate covering the 2DEG, there will not be many electrons in the 2DEG. This would be different for a device that has a basin of electrons next to the dots. Since there are few electrons, the effects of their potential will not be pronounced compared to the potential induced by the gates.

In case one would like to include these effects in the model, the package we used for solving for the potential, PESCADO, includes an option to solve for the coupled electrostatic-Schrodinger equation [20]. However, when these effects are included, the total potential would no longer be a linear superposition of the Green's potentials of each gate. This concept, which was explained in section 3.1, is essential to the computational efficiency of the tuning procedure of the device. In order to tune a device without using this Green's potential trick, the potential solver would have to be run for every set of gate voltages in the tuning procedure. Moreover, the gradient of the tuning point loss would not be able to be obtained using the same perturbation theory approach. While this does not make tuning a device impossible, the fact that solving for the potential is a computationally expensive procedure would make the tuning of a single device geometry take a lot longer.

Another simplification of our model is that we did not model the effects of strain on the system. As the gate electrodes are not made of the same material as the semiconductor below them, and these devices are operated at temperatures below 4.3 K, the differences in the shrinking of the materials cause strain. This induces an extra potential in the 2DEG, as the semiconductor lattice is strained. Recent research has shown that the effects of strain are significant contributors to the dynamics of a device [35]. While it would be interesting to model these effects, we did not include them in this project.

While the use of direct solvers for the potential does not affect the accuracy of the model, it limits how much the resolution of the model can be scaled up. The matrix factorization approach we use to solve the linear system requires us to construct the matrix for the linear system in the first place. Even though this matrix is sparse, its factorization will be dense. This factorized matrix takes a lot of memory storage. If we would use iterative methods to solve the linear system, we would not need to store a factorized matrix. We could even use a matrix-free approach in conjunction with an iterative solver to solve for the potential. While this would free memory, making higher resolution simulation possible, the Woodbury update technique 3.13 would no longer work, as there is no factorization or other inverse stored. Thus, it would make larger simulations possible, but these would also take significantly more computation time.

An interesting further extension of this research would be to see if there would be a way to re-use the information generated by an iterative solver in a way akin to how we re-use the factorization with the Woodbury update technique in this work. Perhaps a way can be found to reuse the Hessenberg matrix generated by the GMRES method for one geometry to speed up the GMRES iterations for a slightly different geometry.

For simplicity, we have not imposed a constraint that forces there to be at least a one-pixel space between different gates. For experimental devices, this would be essential of course, as touching gates form a closed circuit, which would mean that they cannot have different voltage levels and essentially act as one big gate. Moreover, including a ‘spacing between gates’ constraint might lead to different results, as it would allow for more possible disorder pixel sites. We would recommend looking into this in case one wishes to use the algorithm to aid in the design of experimental devices.

The optimal geometries shown in this work are limited by the way we chose to discretize the geometries with 10 nm or 20 nm pixels. Consider, for instance, a gate boundary that is 100 nm long. If this boundary is aligned horizontally and discretized using 10 nm pixels, there will be 20 possible disorder pixels. However, if this same boundary were aligned at a 45-degree angle with the discretization grid, it would have roughly 15 possible disorder pixels. This makes it seem as if a gate that has boundaries that align with the discretization grid is more sensitive to disorder than that same gate but with diagonal boundaries. A possible way to mitigate this effect would be to use hexagonal pixels to discretize the 2DEG shape. Hexagonal pixels would allow the discretization of a shape to follow nonhorizontal gate boundaries a bit more closely. Another way to mitigate this issue would be to model the disorder differently than the ‘pixels at the gate boundary’ approach used in this work. While this caveat does not invalidate our conclusion about the potential of geometry optimization algorithms, it highlights the effect that the model choices have on the ‘optimal’ geometry that will be produced by the algorithm.

In this work, we show that the implemented geometry optimization method is useful when it comes to designing geometries optimized for certain characteristics. Nonetheless, we do not compare it to other optimization methods. Future research could compare our Greedy optimization approach to other discrete geometry optimization techniques, such as PSO or Genetic algorithms. It could also be compared to optimization techniques that assign a density to each cell value. This continuous parametrification of the geometry enables continuous and gradient-based techniques to be applied to the geometry optimization problem.

8. Conclusion

In conclusion, we have implemented a discrete geometry optimization algorithm that successfully improves the geometry of the gate electrodes of quantum dot devices based on various target characteristics. While it does not find globally optimal solutions for the optimization problem, the geometries it converges to have better characteristics than ‘naive’ geometry designs with rectangular gates.

We have applied the algorithm to simultaneously minimize the disorder sensitivity while maximizing the level spacing of different devices. We have shown it converges to different geometries for different balances of weights for disorder sensitivity and level spacing. Moreover, we have demonstrated that the technique works for double dot devices and triple dot devices. For the double dot devices, we have included wire connectivity constraints, which produce similar results to the geometries without these constraints.

In addition to optimizing for disorder sensitivity and level spacing, we have also applied our algorithm to optimize for gate locality. When the optimization is performed for solely gate locality terms, the geometries become jagged. This highlights the need for including disorder sensitivity in the optimization, as jagged devices are too sensitive to errors in the manufacturing process. When we simply optimize for disorder sensitivity without any of the other terms, the device gates fill nearly the entire optimization space and the dot wavefunctions become large. As this would cause leaking to higher energy levels, smaller quantum dots are desired. This can be ensured by including the level spacing in the optimization target function. Therefore, the ideal target function consists of disorder sensitivity loss and level spacing loss, with optional gate locality terms. The tradeoff between these weights should be chosen with the application of the device in mind.

The strategies we have implemented for the sake of computational efficiency have proven effective. The use of the Woodbury identity, perturbation theory for loss function gradients, and linear corrections for disordered geometries speed up their respective processes by 14, 12 and 66 times. This amounts to an overall reduction of 62 times the run time of the geometry optimization algorithm.

As this project was proof-of-concept in nature, we simplified some of the physical dynamics in our model of the device. While we recommend looking into including these dynamics in future research, this does not diminish the conclusion that our geometry optimization algorithm is effective in improving device characteristics.

All in all, we have demonstrated the potential of discrete geometry optimization algorithms for improving the design of quantum dot devices. Furthermore, we have successfully introduced the Woodbury Updating and Wannier Perturbation theory strategies to improve the computational efficiency of our model.

References

[1] R. Acharya, D. A. Abanin, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, J. Atalaya, R. Babbush *et al.*, *Quantum error correction below the surface code threshold*, Nature (2024), doi:10.1038/s41586-024-08449-y.

[2] R. Versluis, S. Poletto, N. Khammassi, B. Tarasinski, N. Haider, D. J. Michalak, A. Bruno, K. Bertels and L. DiCarlo, *Scalable quantum circuit and control for a superconducting surface code*, Phys. Rev. Appl. **8**, 034021 (2017), doi:10.1103/PhysRevApplied.8.034021.

[3] P. V. Klimov, A. Bengtsson, C. Quintana, A. Bourassa, S. Hong, A. Dunsworth, K. J. Satzinger, W. P. Livingston, V. Sivak, M. Y. Niu, T. I. Andersen, Y. Zhang *et al.*, *Optimizing quantum gates towards the scale of logical qubits*, Nature Communications **15**(1), 2442 (2024), doi:10.1038/s41467-024-46623-y.

[4] D. Wecker, B. Bauer, B. K. Clark, M. B. Hastings and M. Troyer, *Gate-count estimates for performing quantum chemistry on small quantum computers*, Phys. Rev. A **90**, 022305 (2014), doi:10.1103/PhysRevA.90.022305.

[5] D. Loss and D. P. DiVincenzo, *Quantum computation with quantum dots*, Phys. Rev. A **57**, 120 (1998), doi:10.1103/PhysRevA.57.120.

[6] R. Li, L. Petit, D. P. Franke, J. P. Dehollain, J. Helsen, M. Steudtner, N. K. Thomas, Z. R. Yoscovits, K. J. Singh, S. Wehner, L. M. K. Vandersypen, J. S. Clarke *et al.*, *A crossbar network for silicon quantum dot qubits*, Science Advances **4**(7), eaar3960 (2018), doi:10.1126/sciadv.aar3960, `https://www.science.org/doi/pdf/10.1126/sciadv.aar3960`.

[7] A. Chatterjee, P. Stevenson, S. De Franceschi, A. Morello, N. P. de Leon and F. Kuemmeth, *Semiconductor qubits in practice*, Nature Reviews Physics **3**(3), 157 (2021), doi:10.1038/s42254-021-00283-9.

[8] F. A. Zwanenburg, A. S. Dzurak, A. Morello, M. Y. Simmons, L. C. L. Hollenberg, G. Klimeck, S. Rogge, S. N. Coppersmith and M. A. Eriksson, *Silicon quantum electronics*, Rev. Mod. Phys. **85**, 961 (2013), doi:10.1103/RevModPhys.85.961.

[9] R. Maurand, X. Jehl, D. Kotekar-Patil, A. Corna, H. Bohuslavskyi, R. Laviéville, L. Hutin, S. Barraud, M. Vinet, M. Sanquer and S. De Franceschi, *A cmos silicon spin qubit*, Nature Communications **7**(1), 13575 (2016), doi:10.1038/ncomms13575.

[10] L. M. K. Vandersypen, H. Bluhm, J. S. Clarke, A. S. Dzurak, R. Ishihara, A. Morello, D. J. Reilly, L. R. Schreiber and M. Veldhorst, *Interfacing spin qubits in quantum dots and donors—hot, dense, and coherent*, npj Quantum Information **3**(1), 34 (2017), doi:10.1038/s41534-017-0038-y.

[11] F. Borsoi, N. W. Hendrickx, V. John, M. Meyer, S. Motz, F. van Riggelen, A. Sammak, S. L. de Snoo, G. Scappucci and M. Veldhorst, *Shared control of a 16 semiconductor quantum dot crossbar array*, Nature Nanotechnology **19**(1), 21 (2024), doi:10.1038/s41565-023-01491-3.

[12] N. Piot, B. Brun, V. Schmitt, S. Zihlmann, V. P. Michal, A. Apra, J. C. Abadillo-Uriel, X. Jehl, B. Bertrand, H. Niebojewski, L. Hutin, M. Vinet *et al.*, *A single hole spin with enhanced coherence in natural silicon*, Nature Nanotechnology **17**(10), 1072 (2022), doi:10.1038/s41565-022-01196-z.

[13] V. Lordi and J. M. Nichol, *Advances and opportunities in materials science for scalable quantum computing*, MRS Bulletin **46**(7), 589 (2021), doi:10.1557/s43577-021-00133-0.

[14] N. P. de Leon, K. M. Itoh, D. Kim, K. K. Mehta, T. E. Northup, H. Paik, B. S. Palmer, N. Samarth, S. Sangtawesin and D. W. Steuerman, *Materials challenges and opportunities for quantum computing hardware*, Science **372**(6539), eabb2823 (2021), doi:10.1126/science.abb2823, https://www.science.org/doi/pdf/10.1126/science.abb2823.

[15] N. W. Hendrickx, W. I. L. Lawrie, M. Russ, F. van Riggelen, S. L. de Snoo, R. N. Schouten, A. Sammak, G. Scappucci and M. Veldhorst, *A four-qubit germanium quantum processor*, Nature **591**(7851), 580 (2021), doi:10.1038/s41586-021-03332-6.

[16] M. Meyer, C. Déprez, I. N. Meijer, F. K. Unseld, S. Karwal, A. Sammak, G. Scappucci, L. M. K. Vandersypen and M. Veldhorst, *Single-electron occupation in quantum dot arrays at selectable plunger gate voltage*, Nano Letters **23**(24), 11593 (2023), doi:10.1021/acs.nanolett.3c03349.

[17] T.-K. Hsiao, P. Cova Fariña, S. D. Oosterhout, D. Jirovec, X. Zhang, C. J. van Diepen, W. I. L. Lawrie, C.-A. Wang, A. Sammak, G. Scappucci, M. Veldhorst, E. Demler *et al.*, *Exciton transport in a germanium quantum dot ladder*, Phys. Rev. X **14**, 011048 (2024), doi:10.1103/PhysRevX.14.011048.

[18] G. O'Sullivan, B. Li, R. D'Arcy, P. Dunne, P. Hayden, D. Kilbane, T. McCormack, H. Ohashi, F. O'Reilly, P. Sheridan, E. Sokell, C. Suzuki *et al.*, *Spectroscopy of highly charged ions and its relevance to euv and soft x-ray source development*, Journal of Physics B: Atomic, Molecular and Optical Physics **48**(14), 144025 (2015), doi:10.1088/0953-4075/48/14/144025.

[19] P. Armagnat, A. Lacerda-Santos, B. Rossignol, C. Groth and X. Waintal, *The self-consistent quantum-electrostatic problem in strongly non-linear regime*, SciPost Phys. **7**, 031 (2019), doi:10.21468/SciPostPhys.7.3.031.

[20] C. G. A. Lacerda-Santos and X. Waintal, *Electrostatics in semiconducting devices III : The PESCADO open source library (in preparation)* (2026).

[21] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent and J. Koster, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM Journal on Matrix Analysis and Applications **23**(1), 15 (2001), doi:10.1137/S0895479899358194, https://doi.org/10.1137/S0895479899358194.

[22] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent and S. Pralet, *Hybrid scheduling for the parallel solution of linear systems*, Parallel Computing **32**(2), 136 (2006), doi:https://doi.org/10.1016/j.parco.2005.07.004, Parallel Matrix Algorithms and Applications (PMAA'04).

[23] M. A. Woodbury, *Inverting modified matrices*, Princeton University, Princeton, NJ, Statistical Research Group, Memo. Rep. no. 42, (1950).

[24] D. Calvetti, L. Reichel and A. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electronic Trans. Numer. Anal. **2**, 1 (1994).

[25] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, *SciPy 1.0: fundamental algorithms for scientific computing in Python*, Nature methods **17**(3), 261 (2020), doi:10.1038/s41592-019-0686-2.

[26] A. V. Knyazev, *Convergence rate estimates for iterative methods for a mesh symmetric eigenvalue problem* (1987).

[27] N. Marzari, I. Souza and D. Vanderbilt, *An introduction to maximally-localized Wannier functions*, Psi-K newsletter **57**, 129 (2003).

[28] G. H. Wannier, *The Structure of Electronic Excitation Levels in Insulating Crystals*, Phys. Rev. **52**, 191 (1937), doi:10.1103/PhysRev.52.191.

[29] I. Araya Day, S. Miles, H. K. Kerstens, D. Varjas and A. R. Akhmerov, *Pymablock: an algorithm and a package for quasi-degenerate perturbation theory*, doi:10.48550/arXiv.2404.03728 (2024), `2404.03728`.

[30] S. R. Kuppuswamy, H. Kerstens, C.-X. Liu, L. Wang and A. Akhmerov, *Impact of disorder on the distribution of gate coupling strengths in a spin qubit device* (2022), `2208.02190`.

[31] M. Skorski, *Chain rules for hessian and higher derivatives made easy by tensor calculus (preprint)* (2019), doi:10.48550/arXiv.1911.13292.

[32] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Mathematical Programming **45**(1), 503 (1989), doi:10.1007/BF01589116.

[33] J. A. Nelder and R. Mead, *A simplex method for function minimization*, Comput. J. **7**, 308 (1965).

[34] T. A. Straeter, *On the Extension of the Davidon-Broyden Class of Rank One, Quasi-Newton Minimization Methods to an Infinite Dimensional Hilbert Space with Applications to Optimal Control Problems*, Dissertation, North Carolina State University (1971).

[35] J. C. Abadillo-Uriel, E. A. Rodríguez-Mena, B. Martinez and Y.-M. Niquet, *Hole-spin driving by strain-induced spin-orbit interactions*, Physical Review Letters **131**(9) (2023), doi:10.1103/physrevlett.131.097002.

Acknowledgements

8.1. Project acknowledgements

First and foremost, I am grateful for the support of my supervisors, Anton, Juan, and Matthias during this project. I appreciate that they took the time out of their busy schedules for our meetings and that they put in an effort to answer my questions. I am glad that they humored my own ideas, as well as that they provided suggestions and feedback based on their experience.

During this project, it was useful to be able to rely on Anton and Juan's input in the Mattermost channel. Their engagement with the project exceeded my expectations of thesis supervision. More specifically; I'd like to thank Juan for our discussions about modeling devices during the start of the project. These discussions provided the context that shaped what the project would become. While Kostas finished his PhD halfway through the project, it was nice to experience his contagious enthusiasm while brainstorming ways to tackle the Woodbury Updates and Wannier Perturbation Theory.

Matthias' calm outlook on the project was helpful. I enjoyed updating him about the project and discussing things more abstractly and top-down. Even though supervising joint thesis projects is different from what they are used to, I am glad Matthias and Anton were mindful of the fact that there were more project stakeholders than usual.

I want to thank all the members of the Quantum Tinkerer group for their input. It was a privilege to be part of their research community for a while. In particular, I am grateful to Isodora for taking the time to introduce me to Wannier transformation matrices. Moreover, Hugo's templates for matters such as the project notes webpage and the Gitlab CI were very useful. Moreover, while I was too busy bouldering with other people for most of the time, it was fun to join the group bouldering sessions on several occasions.

This project builds on some great packages, and I'd like to thank their developers for their work. The most significant of which is PESCADO [20]. Antonio Lacerda and the rest of the Grenoble team built a great package, and I hope to see it flourish in future research. The potential solving was done using the MUMPS package, which Anton wrapped to make the python library `python-mumps` [21]. It was a huge time save that he added support for `single` and `double` datatypes instead of only `complex128`. It was also a great advantage to be able to use Pymablock for the perturbation theory and to be able to ask questions to the developers directly [29].

Last but not least, I would like to thank the lecturers and teachers I had over the years. Thanks to them, I had the information necessary to comprehend the concepts in this project.

8.2. Personal acknowledgements

On a more personal note, I am grateful for the support my parents have given me during this project and the rest of my studies. It is great that they've supported my various interests since childhood. I also appreciate my friends and other connections for helping me see the colors in life.

I am immensely grateful to all those who supported me in the months around Mik's passing. Especially Linde, who helped me make some good memories in a difficult time. As well as my sister Andrea, who came to support me during the toughest day of my life. I am happy to know Pien, who I can always call when something is up. I am grateful for Helma, whose calls made me feel like my experience is understood. I would like to thank my mother, who made sure to check in with me, and my father, who was always prepared to help me get things in order. I also enjoyed the philosophical chats with Tom, and the chill calls with Alex. Moreover, sharing the grief over a loved one with Liv, Brecht, Didi and Mik's family made it a bit lighter.

I am also fortunate to be able to access therapists and other mental health professionals during this time. I appreciate them for doing such complex work.

Thanks all!
- Julian

Appendices

A. Perturbation theory for maximally localized wavefunctions

i. Relevance

This appendix serves as supplementary material for the section on perturbation theory, subsection 3.3. It explains how a perturbative series for the Wannier transformation matrix is obtained from multiple projected position operators.

ii. Basics of the Wannier transformation matrix W

The purpose of the Wannier transformation matrix is to take a basis of wavefunctions to a maximally localized basis. The Wannier transformation matrix is the solution to a minimization problem, the problem corresponds to a simultaneous diagonalization of non-commuting matrixes. In the event that there is only one projected position operator, the solution can be found by diagonalizing this operator. However, when there are projected position operators in multiple directions one needs to diagonalize these non-commuting matrices simultaneously, which is a more complex problem that will be explained below.

For multiple projected position operators, the solution of the minimization problem (3.15) can be found by setting the gradient of the cost function to zero.

Given projected position operators P_i (for $i = x, y, z$ for example) the transformation matrix to a maximally localized wavefunction basis can be found by solving

$$G(W) = \sum_i \Big[R_i, D(R_i)\Big] = 0$$

Where $R_i = W^\dagger P_i W$. Here W must be unitary, so subject to the constraint $W^\dagger W = WW^\dagger = I$. The D operator takes the diagonal values of its input matrix and returns a diagonal matrix with just these values.

iii. The perturbative Wannier operator for multiple projected position operators

The unperturbed Wannier operator is found from the simultaneous diagonalization. Proceeding, the perturbative terms are found using the approach outlined below;

Introduction of Notation:

$$\langle A, B\rangle = [A, D(B)]$$

We wish to find unitary matrix W^* such that $G^*(W^*) = 0$ Here G^* is based on the projected position operators:

$$P_i^* = P_{i,0} + \epsilon P_{i,1} + \epsilon^2 P_{i,2} + \dots$$

We can drop the i index for convenience, so:

$$P^* = P_0 + \epsilon P_1 + \epsilon^2 P_2 + \dots$$

We can expand our new solution, W^* as:

$$W^* = W_0(I + \epsilon W_1 + \epsilon^2 W_2 + \dots)$$

Where W_0 is the unperturbed solution. So $G(W_0) = 0$ We can define:

$$R^* = (W^*)^\dagger P W^*$$

$$R^* = (I + \epsilon W_1 + \epsilon^2 W_2 + \dots)^\dagger W_0^\dagger (P_0 + \epsilon P_1 + \epsilon^2 P_2 + \dots) W_0 (I + \epsilon W_1 + \epsilon^2 W_2 + \dots)$$
$$R^* = (I + \epsilon W_1 + \epsilon^2 W_2 + \dots)^\dagger (R_0 + \epsilon R_1 + \epsilon^2 R_2 + \dots)(I + \epsilon W_1 + \epsilon^2 W_2 + \dots)$$
$$R^* = S_0 + \epsilon S_1 + \epsilon S_2 + \dots$$

We can use this to expand $G^*(W^*)$:

$$G^*(W^*) = \langle R, R \rangle$$

$$G^*(W^*) = \langle S_0, S_0 \rangle + \epsilon\big(\langle S_1, S_0\rangle + \langle S_0, S_1 \rangle\big) + \epsilon^2\big(\langle S_2, S_0\rangle + \langle S_1, S_1\rangle + \langle S_0, S_2\rangle\big) + \dots$$

From this, we can conclude that in order to solve the k-th order perturbation in $G^*(W^*) = 0$ we see that

$$\sum_{l=0}^{k} \langle S_l, S_{k-l} \rangle = 0$$

When one is solving for the k-th order, all lower-order perturbations are already known. Therefore, we can split the known from the unknown terms in the above equation:

$$\langle S_k, S_0 \rangle + \langle S_0, S_k \rangle = -\sum_{l=1}^{k-1} \langle S_l, S_{k-l} \rangle$$

The same can be done for S_k:

$$S_k = \sum_{i=0}^{k} \sum_{j=0}^{k-i} W_i^\dagger R_j W_{k-i-j}$$

$$W_k^\dagger R_0 W_0 + W_0^\dagger R_0 W_k = S_k - \sum_{i=0}^{k} \sum_{j=1}^{k-i} W_i^\dagger R_j W_{k-i-j} - \sum_{i=1}^{k-1} W_i^\dagger R_0 W_{k-i}$$