Virtual Gating of Quantum Dots

Zachary Parrott

August 9, 2019

Abstract

Overview of quantum dot virtual gating theory and implementation of code developed by Sukanya Kudva and Zachary Parrott, Summer 2019. Code saved to group Github account at `https://github.com/mainCSG/DotTuning`. For any questions please do not hesitate to reach out by email.

1 Theory

1.1 Constant Interaction Model

The constant interaction model to describe a quantum dot is based on being able to parameterize the Coulomb interactions among electrons in dot and those in environment through a single total capacitance C [1]. Focusing attention on the case of a double quantum dot (DQD), the total capacitance of the dot is a sum of the capacitance between the other dot, plunger gates and to the source or drain. This is given by the following sum:

$$C_{1(2)} = C_{L(R)} + C_{g1(2)d1(2)} + C_{g2(1)d1(2)} + C_M \tag{1}$$

The second assumption of this model is that the single-particle energy-level spectrum is independent of the Coulomb interactions and then the number of electrons. This leads expression for total energy of dots and the resulting electrochemical potentials.

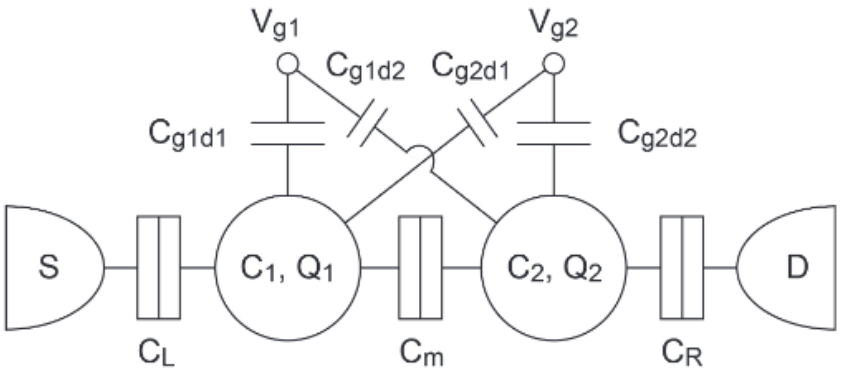

Figure 1: Capacitance diagram of a double quantum dot with cross gate capacitances included [3].

1.2 Electrochemical Potential

While the energy of the dot has a quadratic dependence on the gate voltages, the electrochemical potential has a linear dependence. The function for the electrochemical potential is found from the change in energy from adding an electron to the quantum dot. Following the derivation from van der Wiel *et al* with consideration of the cross capacitance terms expressed in Penfold-Fitch *et al* the electrochemical potential for each dot in a double quantum dot is given in the following equations [2, 3].

$$\mu_1(N_1, N_2) = (N_1 - 1/2)E_{C1} + N_2 E_{Cm} + E_1 \tag{2}$$

$$\mu_2(N_1, N_2) = (N_2 - 1/2)E_{C2} + N_1 E_{Cm} + E_2 \tag{3}$$

Where $E_{C1(2)}$ represents the charging energies of the two dots, E_{Cm} is the electrostatic coupling energy between dots, and $E_{1(2)}$ is from the single-particle energies from the plunger gates. These energies can be expressed in terms of the capacitances and gate voltages.

$$E_{C1} = e^2 \frac{C_2}{C_1 C_2 - C_M^2} \tag{4}$$

$$E_{C2} = e^2 \frac{C_1}{C_1 C_2 - C_M^2} \tag{5}$$

$$E_{Cm} = e^2 \frac{C_M}{C_1 C_2 - C_M^2} \tag{6}$$

and

$$E_1 = -\frac{V_{g1}}{|e|}(C_{g1d1}E_{C1} + C_{g1d2}E_{Cm}) - \frac{V_{g2}}{|e|}(C_{g2d2}E_{Cm} + C_{g2d1}E_{C1}) \tag{7}$$

$$E_2 = -\frac{V_{g1}}{|e|}(C_{g1d1}E_{Cm} + C_{g1d2}E_{C2}) - \frac{V_{g2}}{|e|}(C_{g2d2}E_{C2} + C_{g2d1}E_{Cm}) \tag{8}$$

Rewriting the electrochemical potentials with the expressions for E_1 and E_2:

$$\mu_1(N_1, N_2) = (N_1 - 1/2)E_{C1} + N_2 E_{Cm} - \frac{V_{g1}}{|e|}(C_{g1d1}E_{C1} + C_{g1d2}E_{Cm}) - \frac{V_{g2}}{|e|}(C_{g2d2}E_{Cm} + C_{g2d1}E_{C1}) \tag{9}$$

$$\mu_2(N_1, N_2) = (N_2 - 1/2)E_{C2} + N_1 E_{Cm} - \frac{V_{g1}}{|e|}(C_{g1d1}E_{Cm} + C_{g1d2}E_{C2}) - \frac{V_{g2}}{|e|}(C_{g2d2}E_{C2} + C_{g2d1}E_{Cm}) \tag{10}$$

Now it can be seen that the electrochemical potentials can each be seen as the sum of a function of just the electron counts and a function of the gate voltages:

$$\mu_{1(2)} = f(N_1, N_2) + g(V_{g1}, V_{g2}) \tag{11}$$

Focusing attention on the gate voltage dependence and substituting the expressions for $E_{C1(2)}$ and E_{Cm}, the voltage dependence of the electrochemical potential can be expressed as a vector. (Going forward the electrochemical potential $\mu_{1(2)}$ will refer to just the gate voltage dependence for convenience).

$$\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \frac{-1}{|e|} \frac{e^2}{C_1 C_2 - C_m^2} \begin{pmatrix} C_{g1d1}C_2 + C_{g1d2}C_M & C_{g2d2}C_M + C_{g2d1}C_1 \\ C_{g1d1}C_M + C_{g1d2}C_1 & C_{g2d2}C_1 + C_{g2d1}C_M \end{pmatrix} \begin{pmatrix} V_{g1} \\ V_{g2} \end{pmatrix} \tag{12}$$

1.3 Virtual Gate Voltage Space

As an aside, van der Wiel *et al* present a general capacitance matrix such that the charges on all nodes in a general electrostatic network is given by $Q = CV$, a linear function of the potentials of the nodes [2]. Diagonal elements of the matrix, $C_j j$, are the total capacitance of node j. An off-diagonal term is negative of the capacitance between node j and node k, $C_{jk} = C_{kj} = -c_{jk}$. This can be subdivided into the charges nodes, quantum dots in our case, and the voltage sources:

$$\begin{pmatrix} Q_c \\ Q_v \end{pmatrix} = \begin{pmatrix} C_{cc} & C_{cv} \\ C_{vc} & C_{vv} \end{pmatrix} = \begin{pmatrix} V_c \\ V_v \end{pmatrix} \tag{13}$$

For our case of a double quantum dot $C_c c$ and $C_c v$ are:

$$C_{cc} = \begin{pmatrix} C_1 & -C_M \\ -C_M & C_2 \end{pmatrix} \tag{14}$$

$$C_{cv} = \begin{pmatrix} -C_{g1d1} & -C_{g2d1} & -C_L & -C_{Rd1} \\ -C_{g1d2} & -C_{g2d2} & -C_{Ld2} & -C_R \end{pmatrix} \tag{15}$$

Where the nodes of the source and drain have been included. If they are not included then $C_c v$ is just the first two columns. From here it can quickly be shown that the electrochemical potentials in equation 12 can expressed in terms of $C_c c$ and $C_c v$.

$$\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \frac{-1}{|e|} \frac{e^2}{C_1 C_2 - C_m^2} \begin{pmatrix} C_2 & C_M \\ C_M & C_1 \end{pmatrix} \begin{pmatrix} -C_{g1d1} & -C_{g2d1} \\ -C_{g1d2} & -C_{g2d2} \end{pmatrix} \begin{pmatrix} V_{g1} \\ V_{g2} \end{pmatrix} = \frac{e^2}{|e|} C_{cc}^{-1} C_{cv} \begin{pmatrix} V_{g1} \\ V_{g2} \end{pmatrix} \tag{16}$$

Following the notation of A.R. Mills *et al* the matrix $G = C_{cc}^{-1} C_{cv}$ provides a conversion from gate voltage to electrochemical potential with: $\vec{\mu} = |e| G \vec{V_g}$ [4]. Ultimately for virtual gating we desire electrochemical potential to be just a function of virtual gates: $\vec{\mu} = (cons)\vec{u}$, where $\vec{u}$ is the virtual gate voltages. Virtual gates will be in terms of the physical plunger gate voltages $\vec{u} = G\vec{V_g}$. Then to conduct an experiment in the virtual gate voltage space, the physical plunger gate voltages will be set to be a linear combination of the virtual gates in the following equation.

$$\vec{V_g} = G^{-1}\vec{u} = C_{cv}^{-1} C_{cc} \vec{u} \tag{17}$$

$$\begin{pmatrix} V_{g1} \\ V_{g2} \end{pmatrix} = \begin{pmatrix} -C_{g1d1} & -C_{g2d1} \\ -C_{g1d2} & -C_{g2d2} \end{pmatrix}^{-1} \begin{pmatrix} C_1 & -C_M \\ -C_M & C_2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{18}$$

$$= \frac{1}{C_{g1d1} C_{g2d2} - C_{g1d2} C_{g2d1}} \begin{pmatrix} -C_1 C_{g2d2} - C_M C_{g2d1} & C_M C_{g2d2} + C_2 C_{g2d1} \\ C_1 C_{g1d2} + C_M C_{g1d1} & -C_2 C_{g1d1} - C_M C_{g1d2} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{19}$$

$$\vec{u} = G\vec{V_g} = C_{cc}^{-1} C_{cv} \vec{V_g} \tag{20}$$

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} C_1 & -C_M \\ -C_M & C_2 \end{pmatrix}^{-1} \begin{pmatrix} -C_{g1d1} & -C_{g2d1} \\ -C_{g1d2} & -C_{g2d2} \end{pmatrix} \begin{pmatrix} V_{g1} \\ V_{g2} \end{pmatrix} \tag{21}$$

$$= \frac{-1}{C_1 C_2 - C_M^2} \begin{pmatrix} C_2 C_{g1d1} + C_M C_{g1d2} & C_M C_{g2d2} + C_2 C_{g2d1} \\ C_1 C_{g1d2} + C_M C_{g1d1} & C_1 C_{g2d2} + C_M C_{g2d1} \end{pmatrix} \begin{pmatrix} V_{g1} \\ V_{g2} \end{pmatrix} \tag{22}$$

Following the method of A.R. Mills *et al* the matrix G should put $\vec{u}$ in units of comparable magnitude of the voltage gate space. This can be done by setting the first entry $G_{11} = 1$. Thus the task of creating virtual gating which seeks to have tunable parameter that only affects the electrochemical potential of one dot, reduce the cross and mutual capacitance, is a matter of finding all the capacitances in the system.

1.4 Charge Stability Diagrams

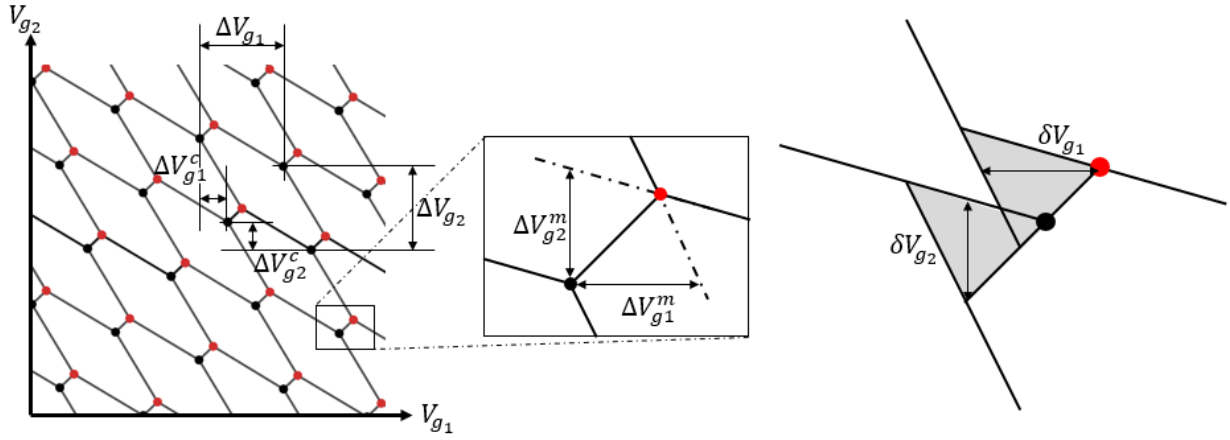

Figure 2: Typical charge stability diagram with dimensions highlighted and in case of an applied source-drain bias.

In the characterization of a DQD, the charge stability diagram provides a visualization of the Coulomb blockade. As a 2D sweep of both plunger gate voltages this plot provides dimensions that relate to specific changes in the electrochemical potentials where the count of electrons on either dot changes. This can then be used to give value to the capacitance terms. From the equations presented in Penfold-Fitch *et al*:

$$C_{g1(2)d1(2)} = \frac{\Delta V_{g2(1)}}{\Delta V_{g1}\Delta V_{g2} - \Delta V^c_{g1}\Delta V^c_{g2}} \tag{23}$$

$$C_{g1(2)d2(1)} = \frac{\Delta V^c_{g2(1)}}{\Delta V_{g1}\Delta V_{g2} - \Delta V^c_{g1}\Delta V^c_{g2}} \tag{24}$$

$$\frac{C_{1(2)}}{C_M} = \frac{|e|}{C_{g2(1)d2(1)}\Delta V^m_{g2(1)}} - \frac{C_{g2(1)d1(2)}}{C_{g2(1)d2(1)}} \tag{25}$$

And with a source-drain bias voltage V_{bias}:

$$|e|\delta V_{g1(2)} = \frac{C_1C_2 - C_M^2}{C_{g1(2)d1(2)}C_{2(1)} + C_{g1(2)d2(1)}C_M}|eV_{bias}| \tag{26}$$

These equations for the capacitances present three different regimes for which how much can be extracted. In the first case, either due to a broad sweep or operating at higher

temperature, only rough current blobs can be seen and no triple point pair separation. At this level the gate capacitances and cross gate capacitances can be extracted. The second regime is where there is visible separation of the triple points, $V^m_{g2(1)}$, leading to additionally extract the ratios $C_{1(2)}/C_M$. The final case is when a source-drain bias is applied and bias triangles are visible. In practice for the charge stability diagrams previously made for our devices, at zero bias there still is bias triangles so a baseline has to be established.

1.5 Charge Stability Diagram in Virtual Gate Space

When the full capacitance matrix is determined and a dual sweep of the virtual gates is run a charge stability diagram will look the second plot in figure 3. Of note is that around a particular triple point pair the boundaries between adjacent cells are only horizontal or vertical. This demonstrates the intent of virtual gating to allow one to tune to a configuration of electrons with only adjusting one virtual gate.

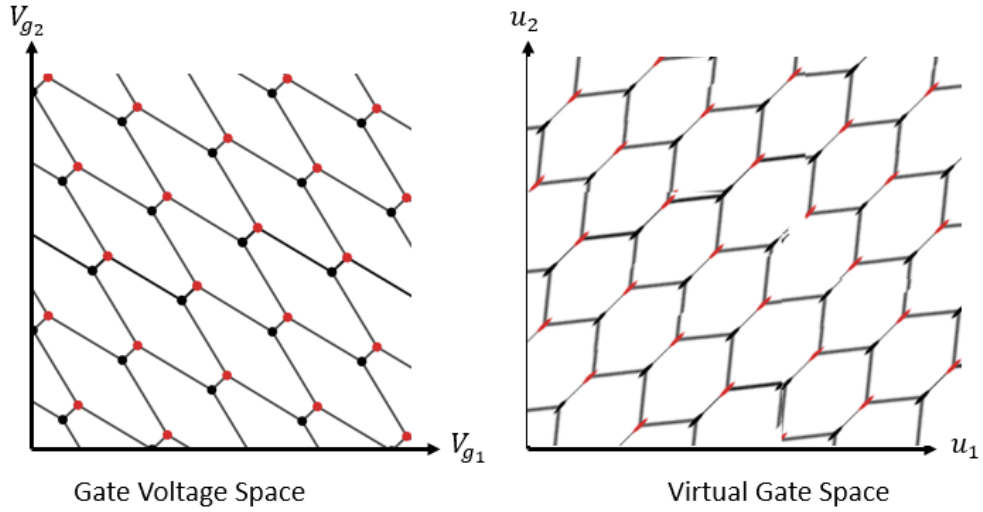

Figure 3: Typical charge stability diagram and result after solving for virtual gates.

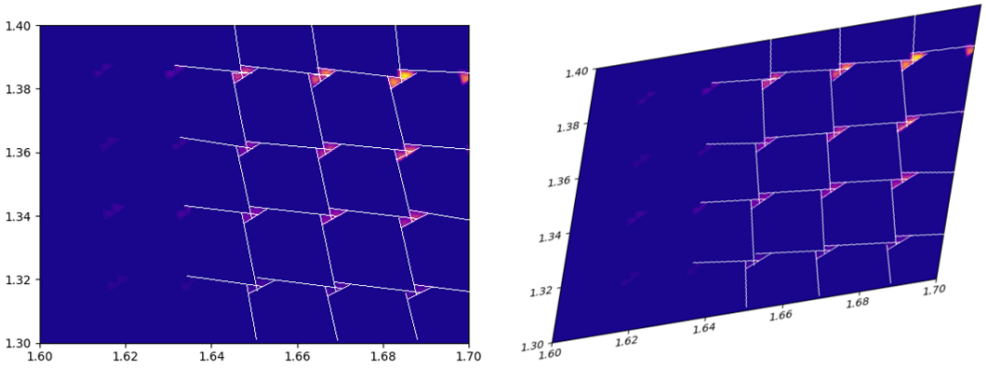

Figure 4: Computed capacitance matrix applied as an affine transform on image as demonstration of virtual gating.

1.6 Gate Capacitance Correction

When only the gate capacitances are determined there is still room for a partial correction in a virtual space. This method is not founded around the electrochemical potential basis presented, but rather only reduces cross gate capacitance effects while still retaining effects from mutual capacitance between dots. This is accomplished by using the large lattice dimensions to get the slope between nearest electron triple points to get a slope to offset. This is given by the following matrix:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 1 & \frac{C_{g1d2}}{C_{g2d2}} \\ \frac{C_{g2d1}}{C_{g1d1}} & 1 \end{pmatrix} \begin{pmatrix} V_{g1} \\ V_{g2} \end{pmatrix} \tag{27}$$

$$\begin{pmatrix} V_{g1} \\ V_{g2} \end{pmatrix} = \frac{C_{g1d1}C_{g2d2}}{C_{g1d1}C_{g2d2} - C_{g2d1}C_{g1d2}} \begin{pmatrix} 1 & -\frac{C_{g1d2}}{C_{g2d2}} \\ -\frac{C_{g2d1}}{C_{g1d1}} & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{28}$$

A dual sweep of the virtual gates is run a charge stability diagram will look the second plot in figure 5. The result lacks orthogonal boundaries, but still could allow for finer control of devices in cases of strong cross gate capacitances. Now the triple point pairs are all grouped on a rectangular lattice as opposed to the skewed quadrilateral of before.

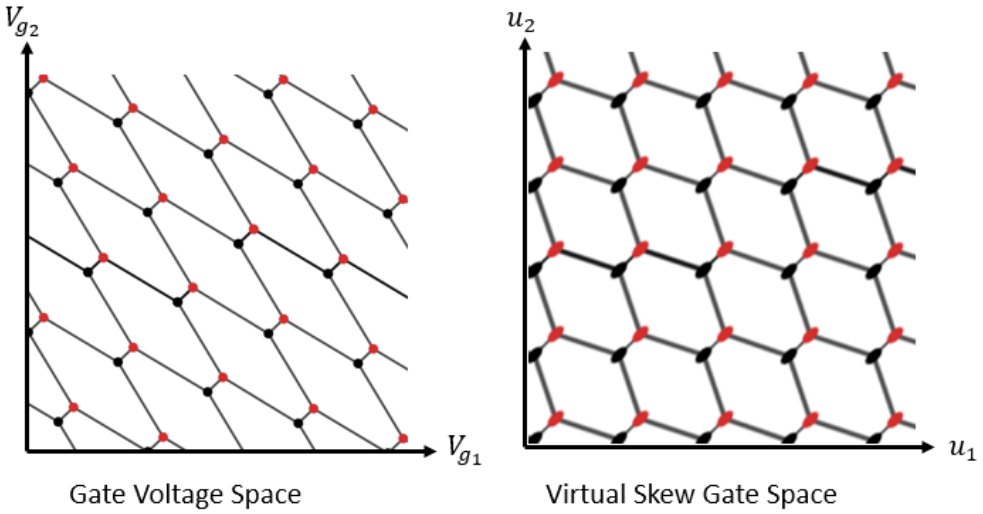

Figure 5: Typical charge stability diagram and result after cross capacitance reduction is applied with virtual gating.

1.7 Scaling Past N=2

While this discussion has focused on the case a system of two quantum dots, the process to get virtual gating for larger linear array of quantum dots is largely the same. The definition of the matrices C_{cc} and C_{cv} remains unchanged, but only the first off diagonals are used. This is for two coupled reasons. Both the mutual capacitance between non-neighboring dots, i.e. C_m13, will be quite smaller than the neighboring dots, C_{m12} and C_{m23}. Likewise the cross gate capacitance C_{g3d1} will be quite smaller than C_{g2d1} and especially smaller than

C_{g1d1}. In general the effect from non-neighboring dots and their gate will have very negligible effect, Secondly, because the effect is so small, in a charge stability diagram between two nonadjacent gates, i.e. V_{g1} vs V_{g3}, the pattern will be quite faint and likely more complicated than the honeycomb due to the dot in between.

Thus, for tuning a linear array of quantum dots into virtual space one just needs to tune neighboring pairs of dots in the same method described here for double dots. See A.R. Mills *et al* Supplementary Materials[4].

2 Our Code

2.1 Overview

As described in section 1.4, the capacitances of the system can be extracted by measuring dimension of the honeycomb lattice and the bias triangles on a charge stability diagram. The fitting is broken into two sections, gate (lattice dimensions) and bias triangles, and each described in detail below. The program takes as input a CSV file of the same format that Spanish Acquisition writes during an experiment. The program is run from a command line and has options for user input along the way to change various parameters.

2.2 Gate Fitting

The gate fitting portion of the code seeks to find the capacitances related the plunger gates: $C_{g1(2)d1(2)}$ and $C_{g2(1)d1(2)}$. In the command line prompts this is referred to as the low resolution fit as it can be done on coarse data sets that only of current 'blobs' over the triple point pairings. As is seen in equations 23 and 24 this is done by measuring the dimensions of the quadrilateral drawn between a group of four nearest pairs. This gives the dimensions $\Delta V_{g1(2)}$ and $\Delta V^c_{g1(2)}$.

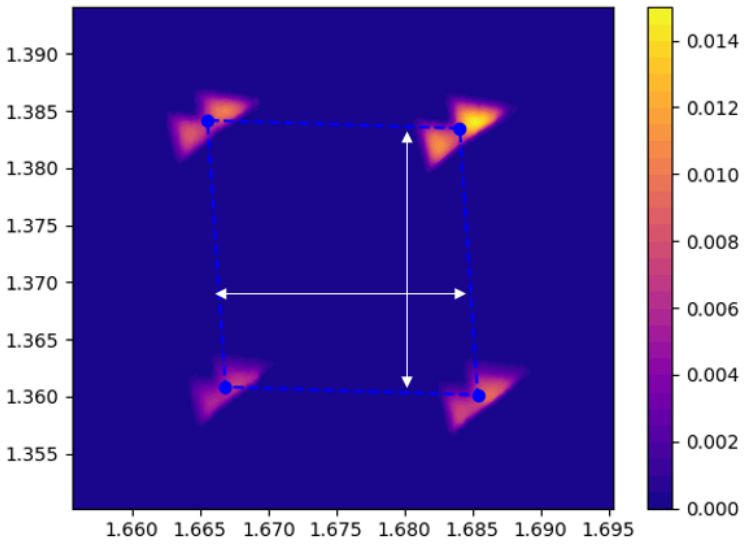

Figure 6: Result of DBSCAN cluster fitting for one honeycomb group. White arrows overlayed to show extracted dimensions ΔV_{g1} and ΔV_{g2}.

In the code this is accomplished by first applying a threshold factor that sets the background current to zero. After this only data points corresponding to the 'blobs' or bias triangles should be non zero. At a high level the next portion finds groupings of these nonzero points and clusters them. Within each cluster a central point is determined that minimizes the distance to all other points in the cluster. This central point called a centroid in the code is found for each of the four groups and lines are drawn in between. **Note:** if the data is not cropped on one group of four this portion can take some time to execute, especially on lab computers, as it will evaluate each cluster in the whole scan and find the nearest four with the greatest number of points associated with.

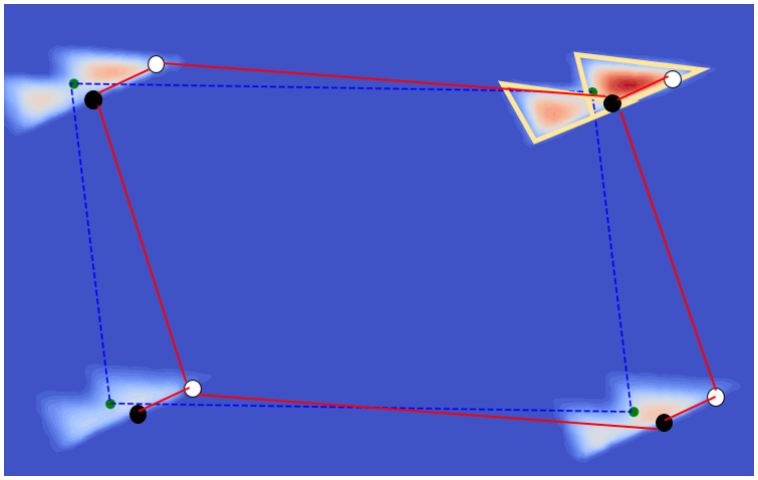

Figure 7: Honeycomb lattice overlayed over gate fitting quadrilateral. While the centroid do not align with the electron-like triple points it is merely a translation and fit is still valid.

2.3 Bias Triangle Fitting

If the 2D sweep loaded includes a source drain bias and of appropriate resolution the bias triangles can be fit to further determine $\delta V_{g1(2)}$ and get the full capacitance matrix. This fit is done by honing in on one cluster and then determining what points correspond with the boundary. Once the boundary points are identified a window for user prompt opens similar to figure 8. At this point the user selects roughly what points form the vertices of the polygon.

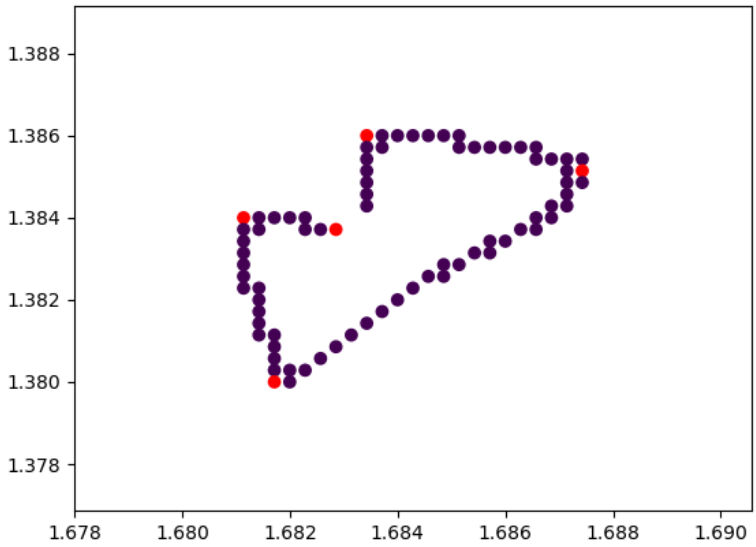

Figure 8: Screenshot of the menu to get user initial guess of the five vertices defining a pair of overlapping bias triangles.

From the five vertices entered the program seeks to a fit to find the five lines that best fit the boundary points. In the orientation shown for a positive bias the bottom line is independent while the other four are forced to be the same slope within corresponding pairs. Resulting fit to data is shown in figure 9 where the extracted dimensions have been drawn over.

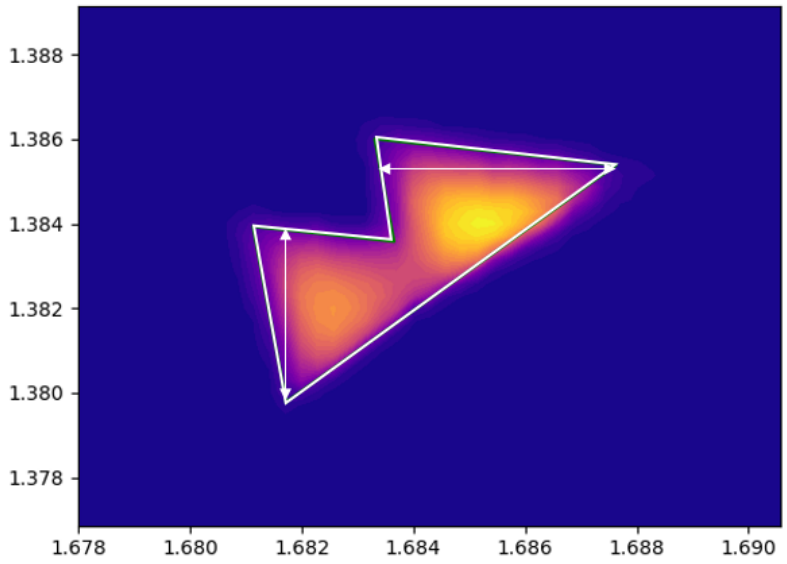

Figure 9: Bias triangle fitting with white arrows added to show extracted dimensions δV_{g1} and δV_{g2}.

2.4 Virtual Sweep Configuration

At the end of the fitting code, `main_prompt.py`, the user has the option to save the computed capacitances values to A CSV file. Then to serve as a helper to setup the virtual sweep in Spanish Acquisition the program `userinput.py` would be run. With this program through the command line the previously outputted CSV file is retrieved and the appropriate capacitance matrix is generated for both the full correction and skew correction depending on which is selected. Additionally it will tell what range in the virtual space to sweep in order to sweep over the equivalent region in the voltage gate space. Output includes what dependent function expressions to copy over to the 'Virtual Setup' program for the gate voltages.

2.5 Install and Spanish Acquisition Interface

Due to the nature of Spanish Acquisition being written on Python 2 and using many packages that have been significantly updated or no longer available any additional scripts added to the lab computers must be done in a careful matter. For the install and use of virtual gating files a virtual environment is needed. This is because the packages `pandas`, `scikit-image`, and `matplotlib`, are all dependent on newer versions of other packages that will brake Spanish Acquisition.

```
%Install virtualenv:

pip install --user virtualenv

%Make virtual environment inheriting global site-packages from home directory:
```

```
virtualenv --system-site-packages dot_tuning

%activate new environment:

source dot_tuning/bin/activate

%install additional packages to dot_tuning only:

pip install pandas --ignore-installed

pip install scikit-image --ignore-installed

%update matplotlib:

pip install matplotlib -U --ignore-installed

%For some reason CentOS python does not have tkinter:

sudo yum -y install tkinter
```

Then to run the fitting code one would navigate to the home directory from the command line and activate the environment `source dot_tuning/bin/activate`, proceed to run the code, and then Spanish Acquisition can be opened as normal from the desktop icon. When done it is good habit to close the virtual environment.

2.6 Spanish Acquisition Update

In order to enable virtual gating experiments within the framework of the group's existing data acquisition software, **Spanish Acquisition**, the following updates were introduced. In the former version users had two options for generating values for output variables, linear and arbitrary. The linear menu creates an evenly spaced array of values between a initial and final numbers in defined number of steps. The arbitrary menu gives the option to type in any values separated by commas. For virtual gating, while the virtual gates may be swept linearly, the plunger gates which are actually being written to resources, will be a some function of multiple gates. The update adds a third means of giving values to a variable under a menu titled 'From CSV'. This menu allows the user to open a CSV file and select a column of the table to be the values to be swept over. This allows for uses outside of virtual gating experiments.

An additional program outside of 'Acquisition' and 'Data Explorer' called 'Virtual Setup' was added to aide in generating the CSV files need in the specific use case of virtual gating. This program allows the user to setup up linear sweeps in the virtual space and enter expressions for how the real gate voltages are a function of the virtual space. After setup the program will generate and save the appropriate CSV file to be used for output variable setup in 'Acquisition'. As will be discussed further in the example section, in order to have

the intended sweep in the virtual space, all dependent output variables will be made to be of the same order in 'Acquisition'. This is because the 'Virtual Setup' program takes care of creating the proper setup of loops that is normally handled by Acquisition'.

2.7 Work Flow

1. Complete 2D voltage sweep of gate voltages.
2. From command line open `main_prompt.py` and follow prompts.
   (a) Find output CSV file of sweep.
   (b) Select gate voltages and current variable names and any set values for additional outer loop sweeps.
   (c) Select whether low resolution (gate fitting) or high resolution bias triangle data is present.
   (d) Follow prompts for cropping, change to threshold value, boundary factor.
   (e) If bias triangle present, follow instructions to indicate five vertices on boundary.
   (f) Select where to save output CSV file of capacitance values to.
3. Open `virtGateHelper.py` and follow prompts.
   (a) Load CSV file exported from `main_prompt.py`.
   (b) Select whether file has only gate capacitances for skew fit or all capacitances for full virtual gating.
   (c) Enter intended sweep range in voltage gate space.
   (d) Program will return what range to sweep in the virtual space and the capacitance matrix G and G^{-1} and equations to copy over to 'Virtual Setup'.
4. With instructions given by `main_prompt.py`, open **Virtual Setup** program of Spanish Acquisition.
   (a) Select how many linear swept variables in virtual space and enter initial, final, steps and order.
   (b) Type or copy and paste equations for the dependent variables that will actually output to the plunger gates.
   (c) When done press `Evaluate/Write CSV` and save the generated CSV.
5. Return to **Spanish Acquisition**.
   (a) Add output variables for each of the gate voltages as well as the virtual variables (in order to save to the output file).
   (b) Write resources to the nonvirtual variables as normal and set all dependent variables and virtual variables to the **same order number**. In 'Virtual Setup' the order was already considered.

(c) For each variable double click the 'Values' cell and use the 'From CSV' option. Load the appropriate CSV generated by 'Virtual Setup' and select the corresponding variable name.

(d) if all of same order, Acquisition should now correctly step the dependent variables together to give the intended linear sweep in the virtual space.

6. Run the sweep and plot '2D: Colormapped..' in 'Data Explorer' with the virtual variables as X and Y.

2.8 Overview of Each Script

This section has more thorough descriptions of every Python script that is utilized in the fitting.

- **main_prompt.py**

  The main script file. Reads, loads and crops(if instructed) the data file. Run other scripts for low or high bias regime through this file.

  These scripts are for both low and high resolution regimes. (For high resolution data, you would have to run the same scripts as for low resolution data and some extra scripts)

- **crop.py**

  You can choose to uncomment this is in main script if no cropping is required. It takes 4 arbitrary points (inputs given should be adjacent vertices). All data points within this arbitrary quadrilateral are kept, the rest are set to zero.

- **curr_thresh_filter.py**

  Calculates a threshold for the current and filters out current values above the threshold. Current threshold factor(curr_thresh_factor) can be set in main.py to vary this threshold. It simply multiplies the factor to the threshold value calculated.

- **DBSCAN.py**

  (borrowed from somebody's github repo) It identifies regions around the triple points separately as 'clusters' and numbers them.

- **assign_clst_qual.py**

  Calculates 'quality' for each region(cluster). This quality is defined as sum of current values measured in the region. Better current signal means better quality.

- **assign_clst_centroid.py**

  Calculates centroids for each region(cluster).

- **find_clusters.py**

  Takes all current regions(clusters) and finds a group of 4 neighbouring clusters that have the best signal.

- **find_Cgs.py**

  Takes centroids of final 4 current regions and fits a parallelogram. Gate and cross gate capacitances i.e Cg1d1, Cg2d2, Cg1d2, Cg2d1 are calculated.

- Extra scripts for high resolution regime:

- **fit_lines_using_initialpts.py**

  Takes one pair of bias triangles. Rough vertices should be given as input in a particular order. The triangle edges are fit as separate lines and vertices calculated. Note-If the dataset has fine sweep of only one pair of bias triangles, run this script directly instead of main.py.

- **fit_lines_4triangles.py**

  Can run this as an alternate to fit_lines_using_initialpts.py. This fits 4 pairs of bias triangles together instead of just one. Parallel set of lines of the 4 triangles are fit together. Again, rough vertices for one pair of triangles should be given as input in a particular order.

- **find_Vgms.py**

  Calculates Vgm1 and Vgm2 from the vertices and lines of the triangle(s) fit.

- **find_Cratios.py**

  Calculates C1/Cm and C2/Cm using Vgm1, Vgm2 and the gate and cross gate capacitances.

- **find_Ecs.py**

  Takes lever arm alpha_1 or alpha_2, gate and cross gate capacitances, capacitance ratios C1/Cm and C2/Cm and calculates charging energies Ec1 and Ec2 and electrostatic coupling energy Ecm.

- **find_dVgs.py**

  Calculates dVg1 and dVg2 from the vertices and lines of the fit triangle(s). These values calculated for 2 different data sets at different Vbias(source-drain bias) can be used to calculate lever arms.

- Other scripts:

- **calc_leverarms.py**

  Uses dVg1 and dVg2 from high resolution fit of the triangles for 2 different datasets measured at different V_bias to find lever arms

- **main_curr_thresh_sweep.py**

  main.py file modified for low resolution data. Run this script through curr_thresh_sweep.py

- **curr_thresh_sweep.py**

  Iterates main_curr_thresh_sweep.py for different current threshold values. Set the range of current threshold factor (that decides range of current threshold values) and the number of values to be iterated over in this range through the 'sweep' variable. Different values of gate and cross gate capacitances obtained are plotted vs current threshold factor. Mean and standard deviation of these capacitances is calculated.

- **Accumulation_gate_sweep.py**

  Iterates main_accumulation_gate_sweep.py over the different pairs of accumulation gate values swept over. A fixed current threshold factor (and hence current threshold value) is set. The data is analyzed for low resolution regime. Values of gate and cross gate capacitances calculated is plotted as a function of accumulation gate voltages on a heat map. If an iteration throws an error because the fixed current threshold set is inappropriate, capacitance values for that iteration are set to zero.

- **main_accumulation_gate_sweep.py**

  main.py file modified for low resolution data. Run this script through Accumulation_gate_sweep.py

3 Next Steps

3.1 Validation, Questions and Concerns

1. In the conversion from the real voltage gate space to virtual space the off diagonal terms on the matrix are positive in equation 22. Yet if you apply this matrix as a affine transform or a picture of the charge stability diagram you would need to use the inverse to get the intended virtual picture. I trust the derivation from chemical potential and it should work, but is a still a lingering question from my attempts to validate the code without a ready device to run a virtual space sweep on.

2. In its present form the bias triangle fitting relies on the there be enough V_{bias} such that the two triangle overlap and there is five vertices. If devices are operated in a differently than an update or modification to the fit would be needed.

3.2 Roadmap to Further Automation

In its present form the dot tuning code requires some level of user input for ensuring a proper fit. This mostly is for the bias triangle fit as it proved to be a hard problem when initially tackled from a minimization approach. If the actual capacitance values are not needed but rather just the final capacitance matrix G, then other compute vision methods could be utilized. As presented in preprint followup, the Petta group uses the Hough transform to get the slope and position of the edges of the honeycomb on the charge stability diagram [5]. Given these slopes they can get a matrix that will skew the virtual space to have horizontal and vertical charge transition lines similarly to how we do the partial skew fit. One difference

that allows their data to be more appropriate fitted with this technique is they have done their stability diagram with charge sensing which makes the charge transition lines more apparent.

There are a variety of interesting papers giving automation methods to tune quantum dots into single electron regime and for tuning double dot pairs into a double dot as opposed to a single dot (strong mutual capacitance). Likely to carry out some of these methods would require moving device control away from Spanish Acquisition in order to reduce user input.

References

[1] Hanson, R., Kouwenhoven, L.P., Petta, J.R., Tarucha, S., and Vandersypen, L.M.K., Spins in few-electron quantum dots. *Rev. Mod. Phys.*, 79, 1217-1265 (2007).

[2] van der Wiel, W.G., De Franceschi, S., Elzerman, J.M., Fujisawa, T., Tarucha, S., and Kouwenhoven, L.P., Electron transport through double quantum dots. *Rev. Mod. Phys.*, 75, 1-22 (2003).

[3] Penfold-Fitch, Z.V., Sfigakis, F., and Buitelaar, M.R., Microwave Spectroscopy of a Carbon Nanotube Charge Qubit. *Phys. Rev. Applied*, 7, 054017 (2017).

[4] Mills, A.R., Zajac, D.M., Gullans, M.J., Schupp, F.J., Hazard, T.M., and Petta, J.R., Shuttling a single charge across a one-dimensional array of silicon quantum dots. *Nat. Comm.*, 10, 1063 (2019).

[5] Mills, A.R., Feldman, M.M., Monical, C., Lewis, P.J., Larson, K.W., Mounce, A.M., and Petta, J.R., Computer-automated tuning procedures for semiconductor quantum dot arrays. *arXiv*: 1907.10775v1 (2019).