

Benzene Single-Electron Transistor

Tutorial

Version 12.2.0

Benzene Single-Electron Transistor: Tutorial

Version 12.2.0

Copyright © 2008–2012 QuantumWise A/S

Atomistix ToolKit Copyright Notice

All rights reserved.

This publication may be freely redistributed in its full, unmodified form. No part of this publication may be incorporated or used in other publications without prior written permission from the publisher.

TABLE OF CONTENTS

1. Introduction	1
2. Basic theory of a molecular single-electron transistor	3
The weak and strong coupling transport regimes	3
The energy balance in the weak coupling regime	4
Charge stability diagram for a SET	5
3. Properties of an isolated benzene molecule	6
Molecular energy spectrum of a benzene molecule	6
Total energy of a charged benzene molecule	9
Charge stability diagram of benzene with gold contacts	10
4. Properties of benzene in an SET environment	14
Setting up the calculation	14
Charging energy at zero gate voltage	17
Charging energy as function of the gate voltage	18
Conclusion	26
Bibliography	27
Index	28

CHAPTER 1. INTRODUCTION

The purpose of this tutorial is to present how the Atomistix ToolKit can be used to investigate a molecular single-electron transistor (SET) . The transistor consists of a single benzene molecule weakly coupled to metal electrodes. This system was previously studied in Ref. [1], and the tutorial reproduces the results in this paper. Unlike the common type of systems studied with ATK, this device operates in the **incoherent transport** regime, and the electron transport is described by **sequential tunneling** of single electrons, rather than coherent, ballistic tunneling. The sequential transport mechanism is often also referred to as **Coulomb blockade**.

The benzene molecule is adsorbed on top of a dielectric substrate, and connected with source and drain electrodes. The transport is controlled by an electrostatic back-gate. The system is illustrated in [Figure 1.1](#).

Such a device has not been realized yet. However, the focus in this tutorial is to describe the methodology for calculating the properties of an SET, and this can then be applied to larger, more realistic devices where experiments exist.

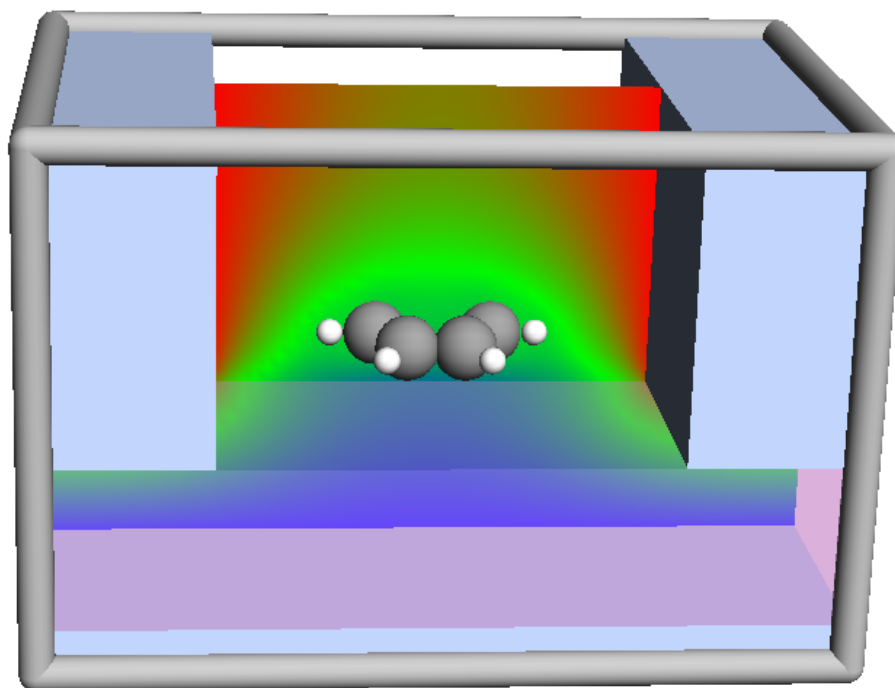


Figure 1.1: The geometry of the molecular SET considered in this tutorial consists of a single benzene molecule within an electrostatic environment which simulates metal source–drain electrodes on top of a dielectric substrate with a metallic back-gate. The contour plot shows the induced electrostatic potential for a gate voltage of 2 V and zero source–drain bias.

The objective is to calculate the **charge stability diagram**. This diagram shows the number of molecular levels inside the bias window for given values of the gate and source–drain voltages.

 **Note**

You will primarily use the graphical user interface Virtual NanoLab (VNL) for setting up and analyzing the results. To familiarize yourself with VNL, it is recommended that you go through the [VNL Tutorial](#).

Atomistix ToolKit is the underlying calculation engine calculation engine for this tutorial. A complete description of all the parameters, and in many cases a longer discussion about their physical relevance, can be found in the [ATK Reference Manual](#).

You are now ready to [begin the tutorial](#).

CHAPTER 2. BASIC THEORY OF A MOLECULAR SINGLE-ELECTRON TRANSISTOR

THE WEAK AND STRONG COUPLING TRANSPORT REGIMES

Figure 2.1 illustrates the geometry of a nanoscale transistor. Electrons are propagating from source to drain through an island. If the island is strongly coupled with the source and drain electrodes, electrons will stay a very short time on the island, and cannot localize but will move coherently through the system. This is the regime where you can use the coherent transport model in ATK for modeling the electrical properties of the system.

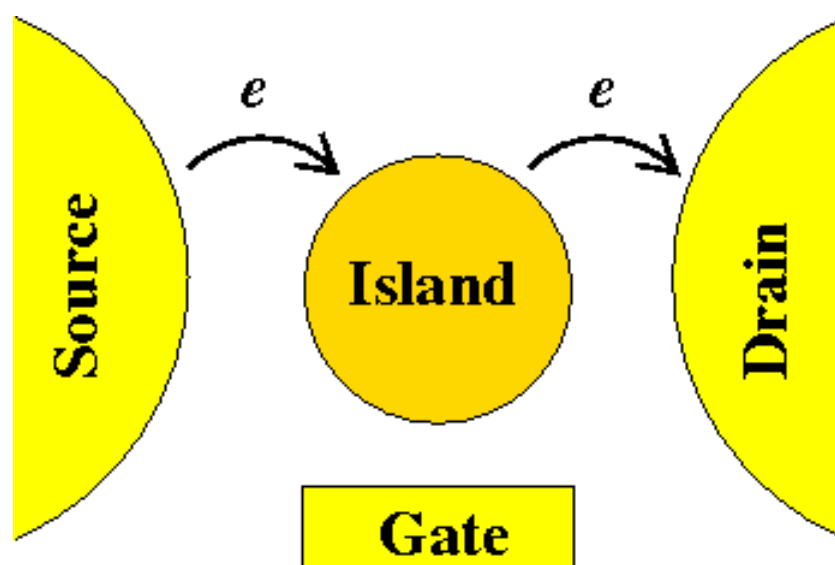


Figure 2.1: Schematic figure of a nanoscale transistor. Electrons propagate from source to drain through an island. The energy of the electronic states on the island can be controlled by an electrostatic gate.

In this tutorial, you will investigate the regime where the island instead is **weakly coupled** with the electrodes. In this regime, the electron tunnels from the source to the island and stays there for sufficiently long time to localize. The electron thereby loses all information about its original quantum state in the source electrode. Therefore, the subsequent tunneling process from the island to the drain electrode will be independent of the tunneling process into the island. This transport mechanism is referred to as sequential tunneling.

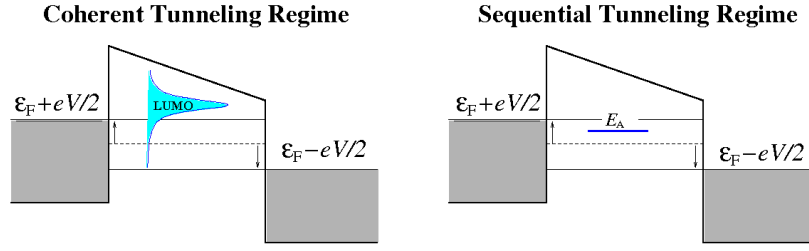


Figure 2.2: In the strong coupling regime (left), the electron propagates coherently through the device, and the electronic states of the island have a finite lifetime and are thus broadened. In the weak coupling regime (right), the electron localizes on the island and propagates by sequential tunneling. The electronic states on the island are in this case discrete.

The two transport regimes are illustrated in Figure 2.2. In the strong coupling regime, a current can flow through the system even when the island does not have any electronic states within the bias window. In this case, the electron can propagate through the “tails” of the finite lifetime broadened state. This is illustrated in the figure by the broadened LUMO (lowest unoccupied molecular orbital).

However, in the weak coupling regime the electronic states of the island are not broadened, and electron transport is only possible if the island has an electronic level within the bias window. This is illustrated by the **electron affinity** level (EA) in Figure 2.2. Notably, the position of the electron affinity level can be adjusted by an external gate potential, and by appropriate tuning, the island can be opened or closed for transport. For source–drain voltages below the charging energy of the island, there will only be one energy level within the bias window, and the system will work as a single electron transistor, as desired for our present discussion.

THE ENERGY BALANCE IN THE WEAK COUPLING REGIME

In the following, the focus is on the weak coupling regime where the transport is described by sequential tunneling. It is convenient to introduce the function $E^{\text{island}}(N)$, which gives the energy of the island as function of the number of electrons on the island. Similar energy functions are introduced for the source and drain electrodes, $E^{\text{source}}(N)$ and $E^{\text{drain}}(N)$.

For the electron to move from the source electrode onto the island, the electron must have a lower energy on the island, i.e.

$$E^{\text{source}}(M) + E^{\text{island}}(N) \geq E^{\text{source}}(M - 1) + E^{\text{island}}(N + 1),$$

where N is the initial number of electrons on the island and M is the initial number of electrons in the source electrode.

In order to move the electron from the island to the drain electrode, it must have a lower energy in the drain electrode:

$$E^{\text{drain}}(K) + E^{\text{island}}(N + 1) \geq E^{\text{drain}}(K + 1) + E^{\text{island}}(N),$$

where K is the initial number of electrons in the drain electrode.

The maximum energy of the electron in the source electrode is $-W + eV/2$, where W is the work function of the electrode and V the applied bias. Assuming that the electron with maximum energy tunnels onto the island, then

$$E^{\text{source}}(M) - E^{\text{source}}(M - 1) = -W + eV/2.$$

The above tunneling criterion, gives rise to the following condition

$$-W + eV/2 + E^{\text{island}}(N) \geq E^{\text{island}}(N + 1).$$

Similarly, $-W - eV/2$ is the minimum energy of an electron in the drain electrode, thus

$$E^{\text{island}}(N+1) \geq -W - eV/2 + E^{\text{island}}(N).$$

The requirement for a current in the device is therefore

$$e|V|/2 \geq \Delta E^{\text{island}}(N) + W \geq -e|V|/2,$$

where $\Delta E^{\text{island}}(N) = E^{\text{island}}(N+1) - E^{\text{island}}(N)$ is the **charging energy** of the island.

CHARGE STABILITY DIAGRAM FOR A SET

The charging energy of the island can be modified by an electrostatic gate. Thus, by tuning the gate voltage, the energy levels of the SET can be moved in and out of the bias window. This dependence of the SET conductance on the gate voltage and the bias potential is illustrated by the **charge stability diagram** in Figure 2.3. The plot uses a color code to show the number of energy levels inside the bias window for a given value of the source–drain bias and the gate voltage. The conductance is directly related to the number of energy levels inside the bias window.

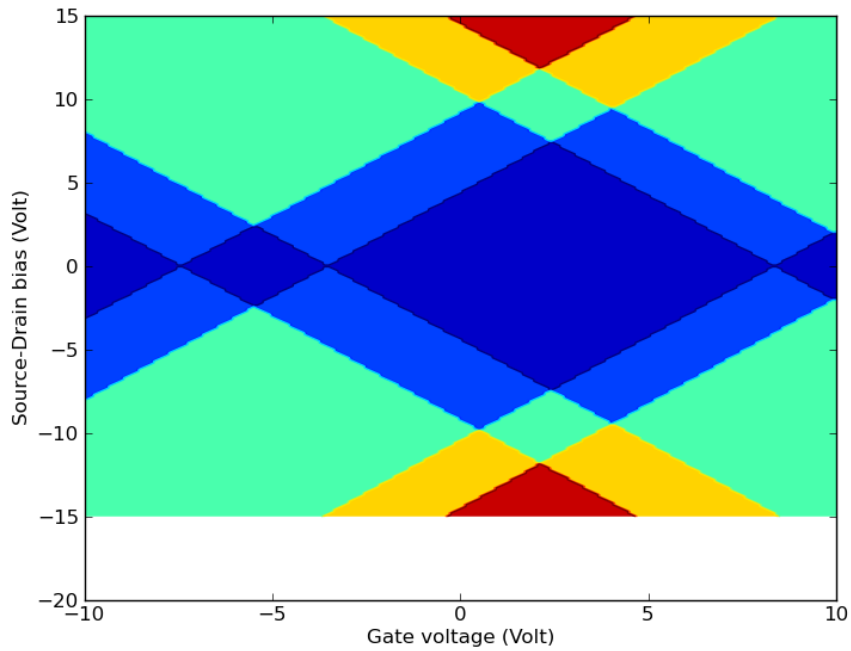


Figure 2.3: The charge stability diagram for the benzene SET, as calculated in the section called “Self-consistent charge stability diagram”. The colors show the number of charge states in the bias window for a given gate voltage. The color map is: blue (0), light blue (1), green (2), orange (3), and red (4).

CHAPTER 3. PROPERTIES OF AN ISOLATED BENZENE MOLECULE

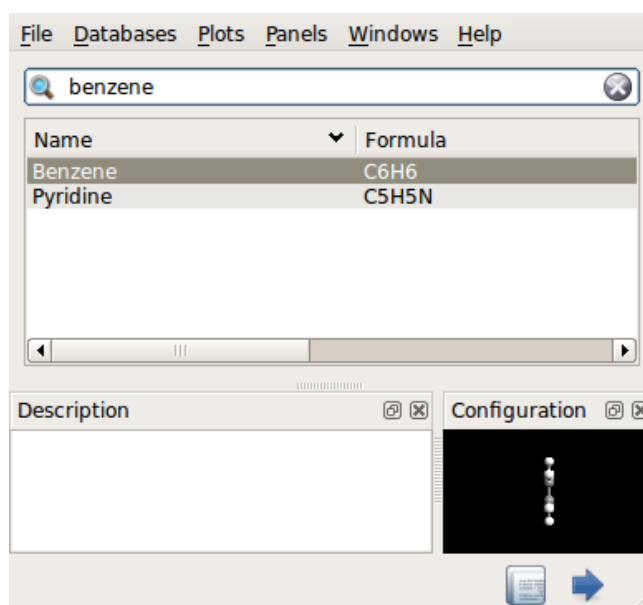
MOLECULAR ENERGY SPECTRUM OF A BENZENE MOLECULE

You will in this chapter model a benzene SET using calculated properties of an isolated benzene molecule in the gas phase. In the next chapter you will compare this model to the results obtained from calculating the properties of the benzene molecule in an electrostatic environment.

SETTING UP THE GEOMETRY

Start up VNL and launch the **Database** tool by clicking the icon  on the **Toolbar**.


Next select **Databases** → **Molecules** from the menu. In the search field, type benzene. You should now see the following



 **Tip**

To get a front view of the molecule, rotate the view of the molecule with the **left**-mouse button.

SETTING UP THE CALCULATOR

Click the **Send To** icon  in the lower right-hand corner and select Script Generator to transfer the benzene geometry to the Script Generator tool.

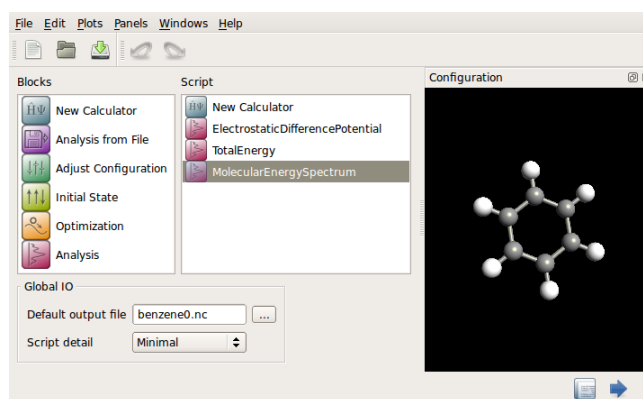
In the **Script Generator**:

1. Double-click **New Calculator** in the left column to insert a method definition block in the script (the DFT method will be selected by default).
2. Double-click **Analysis** and insert **ElectrostaticDifferencePotential**, **MolecularEnergySpectrum**, and **TotalEnergy**
3. Set the **Default output file** to benzene0.nc.

Tip

It is a good idea to create an empty, dedicated directory for storing the scripts and results of the tutorial. When you specify the name of the NetCDF file, make sure to include the full path to this directory, to ensure that it is easily located after the calculations have finished.

The Script Generator window should now look like this.



Next **double**-click the **MolecularEnergySpectrum** icon that has appeared in the right panel in order to change the default parameters. Set the energy zero to **Absolute Energy** such that energies are reported relative to the Vacuum level.

Select File → Save and save the simulation script as `benzene0.py`.

MOLECULAR ENERGY SPECTRUM OF BENZENE

Now drop the file `benzene0.py` on the **Job Manager**  and start processing the queue.

After a few minutes the calculation has converged, and you can inspect the log file to determine the HOMO and the LUMO levels of the molecule. In the print-out of the molecular energy levels you should find

14	-5.907277e+00	2.000000e+00
15	-7.292105e-01	6.413524e-44

The occupations show that level 14 is the Highest Occupied Molecular Orbital (HOMO), while level 15 is the Lowest Unoccupied Molecular Orbital (LUMO). Thus,

$$E_{\text{HOMO}} = -5.91 \text{ eV}$$

$$E_{\text{LUMO}} = -0.73 \text{ eV}$$

It is instructive to compare these values with the ionization and affinity energy of benzene. The **ionization energy**, E^I , is the cost of removing a single electron from the molecule and should be equal to $-E_{\text{HOMO}}$. The **affinity energy**, E^A is the energy gain of adding an additional electron, and should be equal to $-E_{\text{LUMO}}$.

The experimental value for the ionization energy is $E^I = 9.25 \text{ eV}$ [2], and the E_{HOMO} value is thus 3.35 eV off – a rather poor approximation.

The experimental affinity energy of benzene is negative, i.e. benzene cannot bind an additional electron and there is therefore no experimental data for the affinity energy. The experimental optical excitation energy of benzene is 10.54 eV [3] which gives an upper bound for the affinity energy of $E^A \leq -1.29 \text{ eV}$. The E_{LUMO} value is thus at least 2 eV off, again a very poor result.

All this is however to be expected: the HOMO and LUMO energies give a poor approximation for the ionization and affinity energies of benzene, since when the molecule becomes charged, there will be an additional charging energy which cannot be described by a calculation of the neutral molecule. In order to properly describe the charging energy, you need to perform self-consistent calculations for a charged molecule, and this is the topic of the next section.

TOTAL ENERGY OF A CHARGED BENZENE MOLECULE

To improve the calculation of the ionization and affinity energies of benzene, you will now make an explicit calculation of the total energy of a charged system. For this purpose define the total energy, E^q of a system with charge q . Thus,

$$E^I = E(N) - E(N - 1) = E^0 - E^{+1}$$

where E^0 is the energy of the neutral system with N electrons and E^{+1} is the energy of the positive ion with $N - 1$ electrons.

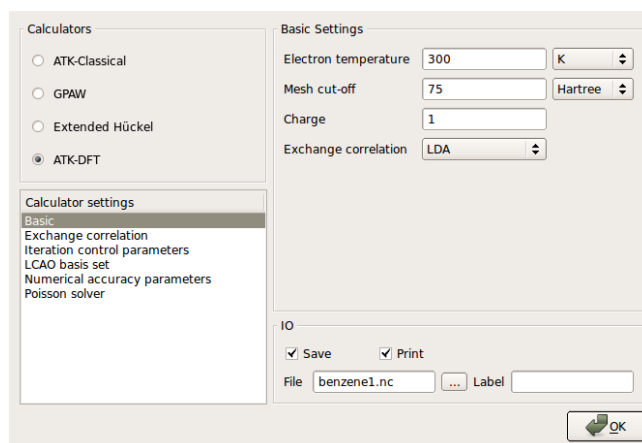
Similarly, the affinity energy is given by

$$E^A = E(N) - E(N + 1) = E^0 - E^{-1}$$

CALCULATIONS FOR THE POSITIVE ION

Re-open the script generator tool, and change the default output file to `benzene1.nc`.

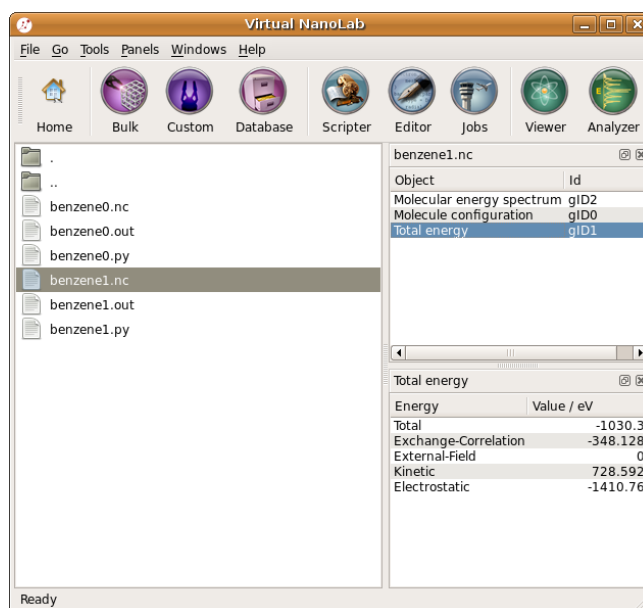
Next **Double-left-click** the New Calculator block and set the charge to 1



Now execute the script using the Job manager.

INSPECTING THE RESULTS

In the VNL main window, select `benzene1.nc` and **left-click** the “Total energy” and you should see the following



Similarly, inspect `benzene0.nc` and find the total energies

$$E^0 = -1039.45 \text{ eV},$$

$$E^{+1} = -1030.3 \text{ eV}.$$

From these values, we can calculate the ionization energy

$$E^I = -9.15 \text{ eV}.$$

which is in excellent agreement with the experimental value.

THE BENZENE AFFINITY ENERGY

Now perform a similar calculation for the negative ion, by reopening the Script Generator and

- change the charge parameter to -1 in the New Calculator block.
- change the default filename to `benzene-1.nc`

Execute the script and find that

$$E^{-1} = -1037.11 \text{ eV}.$$

Consequently, the affinity energy becomes

$$E^A = -2.34 \text{ eV},$$

again in excellent agreement with the experimental value. You see that ATK-DFT can provide very accurate results for the charging energy of molecular systems, if the calculations are performed in the correct way!

CHARGE STABILITY DIAGRAM OF BENZENE WITH GOLD CONTACTS

In this section, you will compute and plot the charge stability diagram for benzene, assuming that it retains its gas phase properties even when it is part of the complete SET geometry. As will

be shown in the next chapter, this is in fact a rather poor approximation, since the molecular charge states are renormalized in the electrostatic environment of the SET. However, for comparison with the results in the next chapter, the analysis below is instructive and will highlight the effect of the electrostatic surroundings on the benzene SET.

To complete the plot, the energies of the doubly charged systems are needed. Using the same approach as for the singly charged systems, find

$$E^{+2} = -1014.57 \text{ eV},$$

$$E^{-2} = -1028.72 \text{ eV}.$$

From this, you can obtain the following addition energies

Table 3.1: Charging energy $E^{n-1} - E^n$ of different charge states of the isolated benzene molecule.

State	Charging energy (eV)
+2	-15.73
+1	-9.15
0	2.34
-1	8.39

GENERATING THE CHARGE STABILITY DIAGRAM

Based on the calculated addition energies of benzene in the gas phase, you will now model how the molecule will function in a SET setup, where it is connected with gold electrodes. To use the formalism in [the section called "The energy balance in the weak coupling regime"](#) [5], you will also need the work function of gold [4],

$$W = 5.28 \text{ eV}.$$

Furthermore, you will need the dependence of the addition energies on the gate voltage, V_G . To first order there is a linear dependence and the linear coefficient is called the gate coupling constant α .

$$\Delta E(N, V_G) = \Delta E(N) + \alpha V_G$$

At this point the value of α is unknown and it is in the following analysis set to 1 for simplicity. [In the next chapter](#) you will determine the gate coupling constant from a more detailed analysis of the fully self-consistent charge stability diagram. As it turns out, for this system α is indeed close to 1, and the linear approximation is quite accurate.

The condition for transmission through a given molecular charge states is then given by

$$-|V_{SD}|/2 \leq \Delta E(N, V_G) + W \leq |V_{SD}|/2.$$

For a given value of V_{SD} and V_G , you can now calculate the number of charge states in the bias window corresponding to the charge stability diagram. Below is a Python script, which uses the addition energies of the isolated benzene molecule (stored in the variable called `delta_e`, defined on line 7) to calculate the charge stability diagram.

```
import numpy
import pylab

#----- Parameter definitons -----#

# Addition energies, already computed
delta_e = numpy.array([-15.73,-9.15,2.34,8.39])
```

```

# Workfunction for the assumed electrodes
w = 5.28

# Gate coupling constant
gate_coupling = 1.0

# Gate bias interval
v_g_interval = numpy.linspace(-15,15,151)

# Source-drain bias interval
v_sd_interval = numpy.linspace(-30,30,301)

#----- End of parameter definitons -----#
# test test test
# Calculate the number of charge states in the bias window
def conductionChannels(v_g,v_sd):
    return numpy.sum( abs( delta_e+w+gate_coupling*v_g) <= abs(v_sd/2) )

# Generate the mesh points of the contour plot
X, Y = numpy.meshgrid(v_g_interval,v_sd_interval)

# Evaluate the number of charge states for each mesh point
Z = [ conductionChannels(X[i,j],Y[i,j])
      for i in range(numpy.shape(X)[0])
      for j in range(numpy.shape(X)[1])]

Z = numpy.array(Z).reshape(numpy.shape(X))

# Make the plot
pylab.contour(X,Y,Z)
pylab.contourf(X,Y,Z)
pylab.xlabel("Gate voltage (Volt)")
pylab.ylabel("Source-Drain bias (Volt)")
pylab.show()

```

The so-obtained diagram is shown in [Figure 3.1](#) (note the plot has been zoomed). As will be shown in the next chapter, the properties of the benzene molecule change, however, when it is placed in the SET environment, and the charge stability will be modified accordingly.

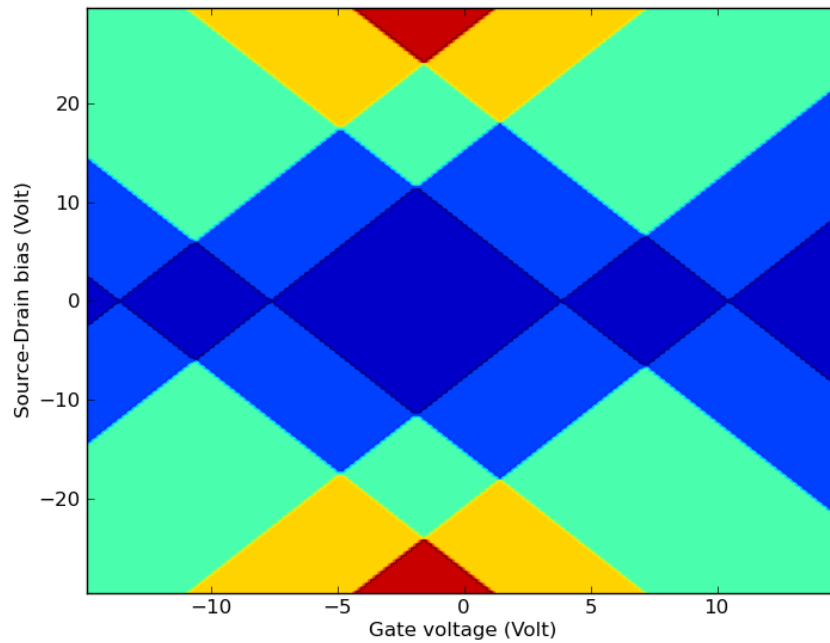


Figure 3.1: The charge stability diagram for a benzene SET, calculated from the properties of the benzene molecule in the gas phase. The colors show the number of charge states in the bias window for a given gate voltage, and exhibit the typical diamond shapes also seen in experimental Coulomb blockade measurements. The color map is: blue (0), light blue (1), green (2), orange (3), and red (4).

In the [next chapter](#) you will model the benzene molecule in a more realistic electrostatic environment.

CHAPTER 4. PROPERTIES OF BENZENE IN AN SET ENVIRONMENT

SETTING UP THE CALCULATION



In this chapter, you will position the benzene molecule in a single-electron transistor environment, consisting of source and drain electrodes, and a gate electrode. Since the molecule is only weakly coupled with the electrodes, the main interaction with the electrodes will be electrostatic, and this interaction will be modelled using a continuum model of the electrodes and the dielectric substrate, rather than an atomistic one.

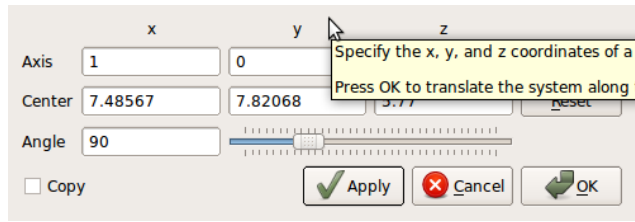
Also note that the system will be modelled as an isolated system, and not as a **DeviceConfiguration** as we otherwise do in ATK for transport properties. This is because, as already mentioned, the transport mechanism is sequential tunneling, and not coherent ballistic tunneling.

BENZENE IN A SUPERCELL

You will now use VNL to set up the simulation of a benzene molecule on top of a 3.7 Angstrom thick dielectric slab, with a metal back-gate, and surrounded by source and drain metal electrodes.

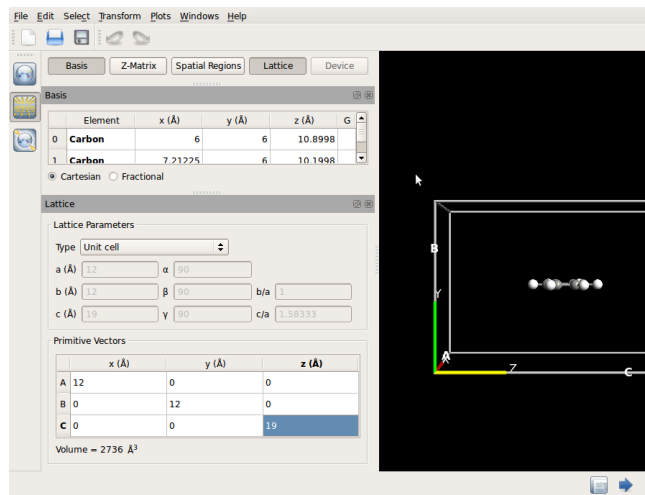
First place the benzene molecule inside a large simulation “supercell” through the following steps

1. Drop the file `benzene0.py` onto the **Builder** 
2. In the Builder, convert the molecule to a periodic configuration by clicking the icon  on the left-hand side of the coordinate list.
3. Change the lattice vectors to:
A: 12.0 0.0 0.0
B: 0.0 12.0 0.0
C: 0.0 0.0 19.0
4. Select all atoms, using Select → All
5. Rotate the molecule 90 degrees about the X axis using Transform → Rotate



6. Center the molecule in the cell (in all directions) using Transform → Center.

The Builder should now look as below.



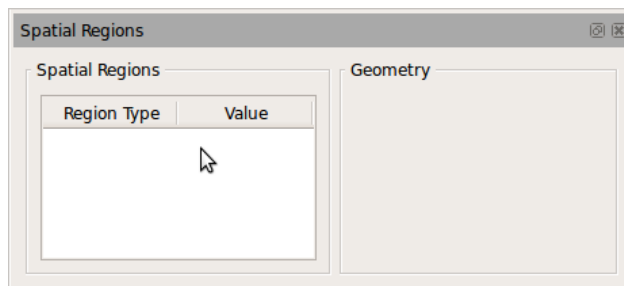
SETTING UP THE CONTINUUM REGIONS

Next step is to add continuum electrodes and a dielectric substrate inside the supercell.

To do this, you need to enable the **Spatial Region** panel by selecting Panels → Spatial Regions. To have enough space, detach the panel by double-clicking the title bar "Spatial Regions", and enlarge it a bit.

First, define the metallic back-gate:

1. **Right**-click inside the spatial regions panel and select Insert region.



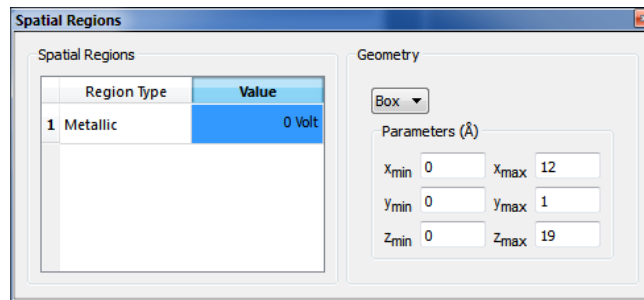
2. Change the value to 0 Volt

3. Set the geometry parameters to:

$$x_{\min} = 0, x_{\max} = 12$$

$$y_{\min} = 0, y_{\max} = 1$$

$$z_{\min} = 0, z_{\max} = 19$$



Now follow the same steps to add a dielectric substrate and source–drain electrodes

1. For the substrate, double-click the **Region Type** field and change the type to **Dielectric**. Then, change the value to $10 \epsilon_0$ and define the geometry as follows:

$$x_{\min} = 0, x_{\max} = 12$$

$$y_{\min} = 1, y_{\max} = 4.7$$

$$z_{\min} = 0, z_{\max} = 19$$

2. Add a metallic region with the value 0 Volt and geometry parameters

$$x_{\min} = 0, x_{\max} = 12$$

$$y_{\min} = 4.7, y_{\max} = 12.$$

$$z_{\min} = 0, z_{\max} = 4$$

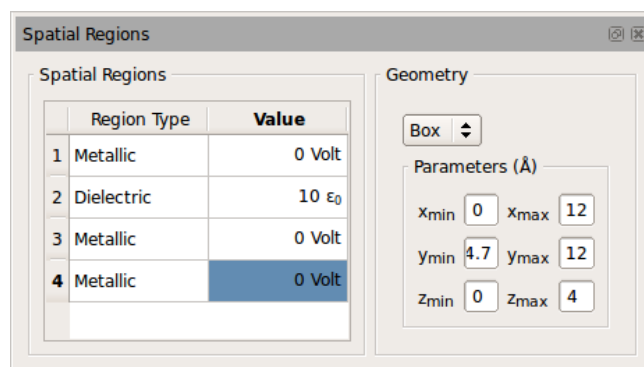
3. Add a metallic region, with value 0 Volt and geometry parameters

$$x_{\min} = 0, x_{\max} = 12$$

$$y_{\min} = 4.7, y_{\max} = 12.$$

$$z_{\min} = 15, z_{\max} = 19$$

This completes the geometry setup. Close the Spatial Regions panel, and the Builder should now look like below:



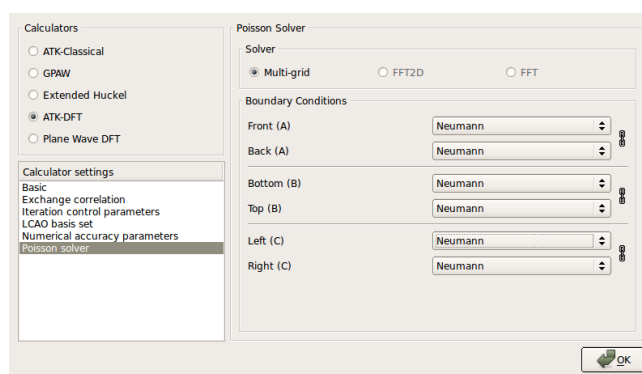
Before you continue, save the geometry as a script, `benzene_set.py`.

SETTING UP THE CALCULATION METHOD

Next, send the benzene geometry to the **Script Generator** using the **Send To** button .

In the **Script Generator**


1. Double-click **New Calculator**.
2. Double-click **Analysis** and insert **ElectrostaticDifferencePotential**, **MolecularEnergySpectrum**, and **TotalEnergy**
3. Set the energy zero of the **MolecularEnergySpectrum** to **Absolute Energy**
4. Set the **Default output file** to `benzene_set0.nc` (again, with proper path).
5. Finally, double-click the **New calculator** block in the right-hand column to modify the method parameters. Under **PoissonSolver**, select **MultiGridSolver** with **Neumann** boundary conditions. In this way, the perpendicular component of the electric field will be zero at the boundaries.



Save the script as `benzene_set0.py`.

CHARGING ENERGY AT ZERO GATE VOLTAGE

MOLECULAR ENERGY SPECTRUM OF NEUTRAL BENZENE IN THE SET GEOMETRY

Drop the file `benzene_set0.py` on the Job Manager  and start the calculation.

This calculation has a larger grid than the previous calculation for benzene in the gas phase, and will take around 6 times longer to finish.

Inspecting the log file, we find that the HOMO and the LUMO levels of the molecule are only modified very slightly compared to the gas phase.

CALCULATING THE TOTAL ENERGY OF A CHARGED BENZENE MOLECULE IN THE SET ENVIRONMENT

In the same way as in the previous chapter, you will now have to compute the total energy of the charged systems in order to improve the value of the addition energy.

Modify the Script Generator to produce the result files `benzene_set-2.py`, `benzene_set-1.py`, `benzene_set1.py`, `benzene_set2.py`; Corresponding to charges -2, -1, 1, 2, respectively.

Run the calculations and you should obtain the addition energies summarized in [Table 4.1](#).

Table 4.1: Charging energy, $E^{n-1} - E^n$, of different charge states of the benzene molecule in the SET environment, at zero gate voltage.

State	Charging energy (eV)
+2	-10.17
+1	-7.5
0	0.1
-1	2.35

Comparing [Table 3.1](#) and [Table 4.1](#), note that the charging energy is reduced in the SET environment. The reduction is due to the stabilization of the charge on the benzene molecule by the electrostatic surrounding. Since the benzene ring is lying flat on the substrate, the charge on the molecule is mainly stabilized by the dielectric substrate.

CHARGING ENERGY AS FUNCTION OF THE GATE VOLTAGE

INDUCED ELECTROSTATIC POTENTIAL

In this section you will calculate the fully self-consistent total energy as a function of the gate voltage. For this purpose, you will use the following script.

```
#read in the old configuration
filename = 'benzene_set0.nc'
bulk_configuration = nload(filename,BulkConfiguration)[0]

# Define gate voltages
gate_voltage_list=[2, 4, 6, 8, -2, -4, -6, -8]*Volt

metallic_regions = bulk_configuration.metallicRegions()
metallic_region0 = metallic_regions[0]

for gate_voltage in gate_voltage_list:
    print 'Gate Voltage = ', gate_voltage

    bulk_configuration.setMetallicRegions(
        [metallic_region0(value = gate_voltage),
         metallic_regions[1],
         metallic_regions[2] ] )

    calculator = bulk_configuration.calculator()
    # Set the calculator on the configuration
    # and use the old calculation as initial state for the scf loop
    bulk_configuration.setCalculator(calculator(),
                                    initial_state=bulk_configuration)

#Analysis
electrostatic_potential = ElectrostaticDifferencePotential(bulk_configuration)
nlsave(filename, electrostatic_potential, object_id='pot'+str(gate_voltage))

molecular_energy_spectrum = MolecularEnergySpectrum(
    configuration=bulk_configuration,
    energy_zero_parameter=FermiLevel,
    projection_list=ProjectionList(All)
)
```

```

nlsave(filename, molecular_energy_spectrum, object_id='spec'+str(gate_voltage))
nlprint(molecular_energy_spectrum)

total_energy = TotalEnergy(bulk_configuration)
nlprint(total_energy)
nlsave(filename, total_energy, object_id='energy'+str(gate_voltage))

```

Start the calculation by dropping the script on the Job Manager, or run it from the command line with the command

```
atkpython gatescan0.py > gatescan0.out
```

Since in this case the script specifies 12 different self-consistent calculations, the execution will approximately take 12 times as long time as each of the previous calculations.

While it is running, you may study some of the details in the script.

As starting point for the calculation, the script reads in the model used for neutral benzene in the SET environment (calculated at zero gate voltage), which was saved in the file `benzene_set0.nc`

```

#read in the old configuration
filename = 'benzene_set0.nc'
bulk_configuration = nlread(filename,BulkConfiguration)[0]

```

Then, it defines the gate voltages which will be used.

```

# Define gate voltages
gate_voltage_list=[2, 4, 6, 8, -2, -4, -6, -8]*Volt

```

For each gate voltage, the gate electrode corresponding to the metallic region number 0 is cloned, updated with the new gate voltage, and then re-attached to the configuration.

```

metallic_regions = bulk_configuration.metallicRegions()
metallic_region0 = metallic_regions[0]

for gate_voltage in gate_voltage_list:
    print 'Gate Voltage = ', gate_voltage

    bulk_configuration.setMetallicRegions(
        [metallic_region0(value = gate_voltage),
         metallic_regions[1],
         metallic_regions[2] ] )

```

Next step is to execute a self-consistent calculation with the new geometry. This is done by attaching a new calculator to the configuration and specifying that the initial state for the self-consistent loop should be the last converged self-consistent state.

```

calculator = bulk_configuration.calculator()
# Set the calculator on the configuration
# and use the old calculation as initial state for the scf loop
bulk_configuration.setCalculator(calculator(),
                                initial_state=bulk_configuration)

```

 **Note**

The parenthesis after the variable **calculator** in the call to **setCalculator** has the effect of cloning the variable, i.e. making a new calculator which is a copy of the old calculator with the same settings.

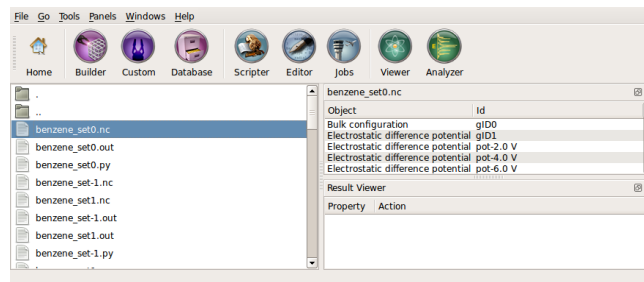
The actual self-consistent loop is initiated when the computation of a physical quantity is requested, in this case the electrostatic potential. Also note how the **object_id** of the potential is set when it is saved to the NC file. This will make it easy to identify which potential grid belongs to which gate voltage when they are all stored in the same file.

```
#Analysis
electrostatic_potential = ElectrostaticDifferencePotential(bulk_configuration)
nlsave(filename, electrostatic_potential, object_id='pot'+str(gate_voltage))
```

VISUALIZING THE GATE POTENTIAL

The electro-static potential induced by the gate voltage, is obtained by subtracting the potential at zero gate voltage from the electrostatic potential at a given gate voltage.

The electrostatic potentials are saved in the file **benzene_set0.nc**. To read the data, you need the **object_id** of each data value. These can be obtained by inspecting the file in VNL.



Having obtained the object ids, it is simple to write a small script that calculates the induced potential.

```
#filename of the data file
filename = 'benzene_set0.nc'

# read electro-static potential with gate potential 0 Volt
potential0 = nlsave(filename, object_id = 'gid1')[0]

#read electro-static potential with gate potential 2 Volt
potential1 = nlsave(filename, object_id = 'pot2.0 V')[0]

#calculate the induced potential and save it to a file
induced_potential = potential1-potential0
nlsave('InducedPotential_2V.nc',induced_potential)
```

Executing the script will generate the file **InducedPotential_2V.nc**, which can be opened in VNL to plot the induced potential. To visualize the SET geometry together with the potential, drag and drop the file **benzene_set0.py** onto the Viewer.

The resulting potential was illustrated in [Figure 1.1](#) in the introductory chapter.

TOTAL ENERGY AS FUNCTION OF THE GATE POTENTIAL

Previously you ran the zero gate voltage calculation for each charge state to produce the files `benzene_set-2.nc`, `benzene_set-1.nc`, `benzene_set0.nc`, `benzene_set1.nc`, and `benzene_set2.nc`. By using these as input, and by modifying the variable `filename` in the script `gatescan0.py` accordingly, it is now possible to perform gate voltage scans for each charge state of benzene in the SET environment.

Tip

Running all these calculations is best set up as a batch job, perhaps overnight. If you create separate scripts for each charge state, like `gatescan-2.py`, `gatescan-1.py`, etc, you can (on Linux, or under Cygwin on Windows) make a bash shell script

```
#!/bin/bash
for i in -2 -1 1 2 ; do
    echo "Running gatescan$i.py"
    time atkpython gatescan$i.py > gatescan$i.log
done
```

The script uses the command `time` to measure how long each run takes. Save the script as `gatescan.sh` in the same directory as the scripts, and give it executable permissions

```
chmod u+x gatescan.sh
```

and execute it

```
./gatescan.sh &
```

On Windows, you can make a file `gatescan.bat`, again in the same directory as the scripts, with the following contents:

```
for %i in (-2 -1 1 2) do atkpython gatescan%i.py > gatescan%i.log
```

Execute it by double-clicking the file.

When all these calculations have completed, you will need a convenient way to analyze the results. Rather than extracting all numbers by hand, the following script implements an utility function that reads in the total energy from the data files. The script uses the function `ninspect` to identify the total energy objects in the file, and determines the gate potential from the `object_id`.

```
from NL.IO.NLSaveUtilities import ninspect

#define function for reading in total energy and gate voltage data from a file
def readTotalEnergy(filename):
    """
    function for reading a list of total energies and gate voltages from a file
    @param filename          : filename of the netcdf file
    @return voltages, energies : list of gate voltages and energies
                                found in filename
    """
    if filename == None:
        return
```

```

#get list of data in file
file_data = nlinspect(filename)
if file_data == None:
    raise ValueError, "Wrong file format"

total_energy_list = numpy.array([])
gate_voltage_list = numpy.array([])

for id in file_data:
    if (id[0] == 'TotalEnergy'):
        total_energy=nlread(filename, object_id = id[1])[0]
        total_energy_list = numpy.append(total_energy_list,
                                         total_energy.evaluate().inUnitsOf(eV))
        #extract the gate voltage from the id of the data
        if id[1][0:3] == 'gID':
            gate_voltage = 0.0
        else:
            gate_voltage = float(id[1].strip(
                "abcdefghijklmnopqrstuvxyzABCDEFGHIJKLMNPOQRSTUVWXYZ"))
        gate_voltage_list = numpy.append(gate_voltage_list, gate_voltage)

if len(total_energy_list) == 0:
    raise ValueError, "No total energy in NC file"

#sort the data
index_list = numpy.argsort(gate_voltage_list)

return gate_voltage_list[index_list], total_energy_list[index_list]

```

The scripts below requires the function **readTotalEnergy** defined in this script. Therefore, save **readTotalEnergy.py** in the same directory as all the other files used in this tutorial.

The following script reads in the total energy from the nc data files and plots the energy as function of the gate voltage. To compare the total energies of different charge states, it is important to add the energy of the additional electron. Assuming that the gold electrodes function as electron reservoirs, the energy of an additional electron is given by the work function of the gold electrode.

```

from readTotalEnergy import readTotalEnergy

# define the work function of gold
w = 5.28

#read in the total energy data from the files: benzene_set[-2,-1,0,1,2].nc
voltage_list= []
energy_list = []

#loop over the charge states
charge_states = [-2,-1,0,1,2]
for q in charge_states:
    voltage,energy = readTotalEnergy('benzene_set'+str(q)+'.nc')
    voltage_list = voltage_list + [voltage]
    # add energy of additional/missing electrons on benzene
    energy = energy -q*w
    energy_list = energy_list + [energy]

#plot the total energies
import pylab
pylab.figure()
for i in range(len(voltage_list)):
    pylab.plot(voltage_list[i],energy_list[i])

pylab.xlabel("Gate voltage (Volt)")
pylab.ylabel("Total Energy (eV)")
pylab.show()

```

The result of running the script is shown in [Figure 4.1](#).

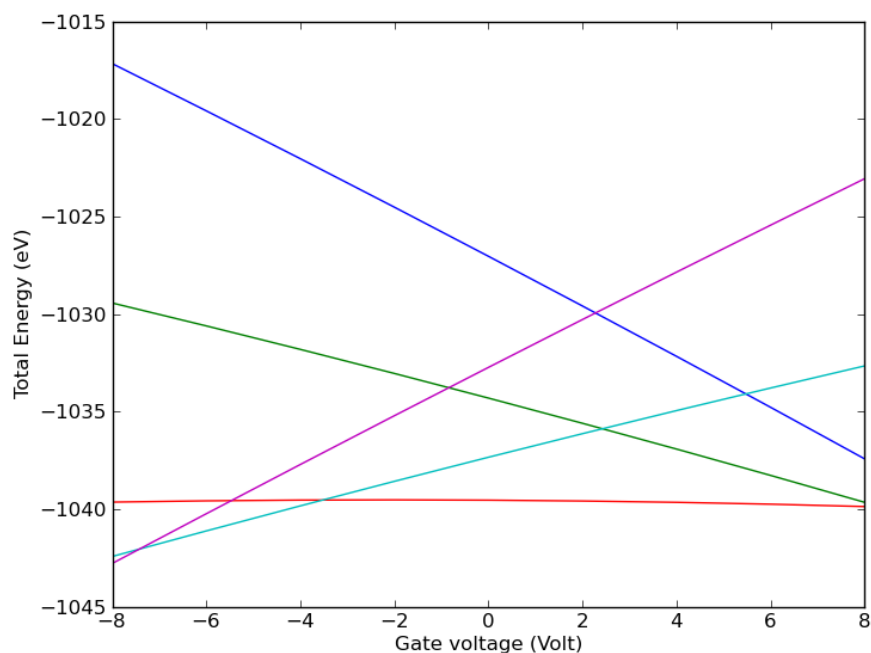


Figure 4.1: The total energy as function of the gate voltage for a benzene SET. Different curves are for different charge states of the benzene molecule, blue (-2), green (-1), red (0), turquoise (1), and violet (2).

The state with the lowest energy is the stable charge state of the benzene molecule for a given gate voltage. Negative gate voltages stabilize positive ions, while positive gate voltages stabilize negative ions. In the region around zero gate voltage, the neutral state is the lowest energy state.

The gate dependence is close to linear, and the slope is related to the charge state of molecule.

To investigate this dependence further, you may fit a linear function to the data. This is done with the following script

```
from readTotalEnergy import readTotalEnergy

#define the charge states
charge_states = numpy.array([-2,-1,0,1,2])
gate_coupling = []

#loop over the charge states
for q in charge_states:
    voltage,energy = readTotalEnergy('benzene_set'+str(q)+'.nc')

    #make linear fit of the charge state
    A = numpy.vstack([voltage,numpy.ones(len(voltage))]).T
    a,b = numpy.linalg.lstsq(A,energy)[0]

    #linear coefficient is the gate coupling
    print 'Charge state %2.0f, gate coupling %7.4f' %(q,a)
    gate_coupling = gate_coupling +[a]

#convert list to numpy.array
gate_coupling = numpy.array(gate_coupling)

#make linear fit to find gate coupling as function of charge
A = numpy.vstack([charge_states,numpy.ones(len(charge_states))]).T
a,b = numpy.linalg.lstsq(A,gate_coupling)[0]
```

```
print
print 'Gate coupling divided by charge state %6.4f' % a
```

The script produces the following output

```
Charge state -2, gate coupling -1.2660
Charge state -1, gate coupling -0.6384
Charge state 0, gate coupling -0.0143
Charge state 1, gate coupling 0.6104
Charge state 2, gate coupling 1.2323

Gate coupling divided by charge state 0.6245
```

The analysis shows that a linear relation

$$E(q, V_G) = \alpha q V_G,$$

is a good approximation for the total energy as a function of gate voltage and charge. You have also hereby calculated the gate coupling constant $\alpha = 0.62$, which in the previous chapter was assumed to have the value 1.

For the neutral state, $q = 0$, the relation corresponds to a horizontal line. The zoom in [Figure 4.2](#) shows that this is a reasonable good approximation

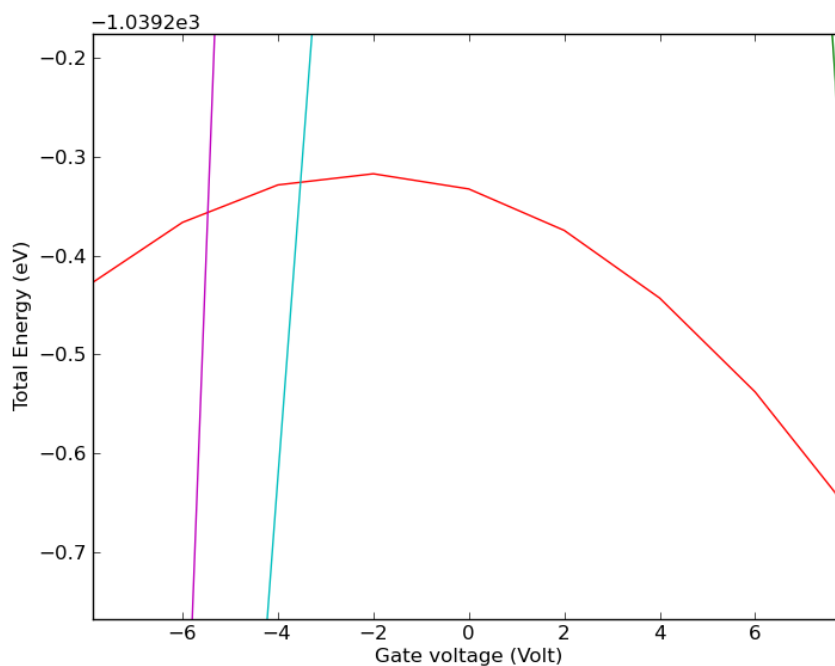


Figure 4.2: [Figure 4.1](#) zoomed to magnify the total energy dependence on gate voltage for the neutral state of benzene.

SELF-CONSISTENT CHARGE STABILITY DIAGRAM

You will now generate the charge stability diagram from the calculated gate dependence of the benzene addition energies.

The same method as above is used to read in the data from the NetCDF files, and then the data are interpolated using the **SplineInterpolation1D** class. The script below is a combination of the scripts used in the previous sections

```

from readTotalEnergy import readTotalEnergy
import pylab

# define the work function of gold
w = 5.28

#gate bias interval
v_g_interval = numpy.linspace(-10,10,201)

#source-drain bias interval
v_sd_interval = numpy.linspace(-15,15,301)

#-----finished define parameters -----#

#read in the total energy data from the files: benzene_set[-2,-1,0,1,2].nc
voltage_list= []
energy_list = []

#loop over the charge states
charge_states = [-2,-1,0,1,2]
for q in charge_states:
    voltage,energy = readTotalEnergy('benzene_set'+str(q)+'.nc')
    voltage_list = voltage_list + [voltage]
    # add energy of additional/missing electrons on benzene
    energy = energy -q*w
    energy_list = energy_list + [energy]

# make numpy arrays instead of lists
voltage_list = numpy.array(voltage_list)
energy_list = numpy.array(energy_list)

#make function that can return the charging energies
charging_energy = []
for i in range(len(charge_states)-1):
    f = SplineInterpolation1D(voltage_list[i],energy_list[i+1]-energy_list[i])
    charging_energy = charging_energy + [f]

#calculate number of charge states in the bias window
def conductionChannels(v_g,v_sd):
    channels = 0
    for f in charging_energy:
        channels = channels + (abs(f(v_g)) <= abs(v_sd/2) )
    return channels

#Generate the mesh points of the contour plot
X, Y = numpy.meshgrid(v_g_interval,v_sd_interval)

#evaluate number of charge states for each mesh point
Z = [ conductionChannels(X[i,j],Y[i,j])
      for i in range(numpy.shape(X)[0])
      for j in range(numpy.shape(X)[1])]

Z = numpy.array(Z).reshape(numpy.shape(X))

#make the plot
pylab.contour(X,Y,Z)
pylab.contourf(X,Y,Z)
pylab.xlabel("Gate voltage (Volt)")
pylab.ylabel("Source-Drain bias (Volt)")
pylab.show()

```

The result of running this script is shown in [Figure 4.3](#).

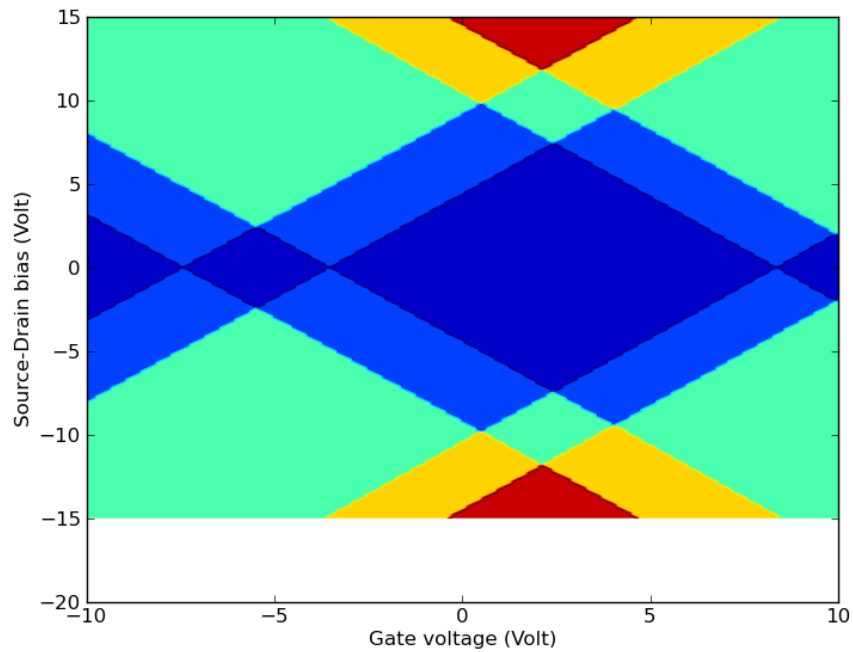


Figure 4.3: The full self-consistently calculated charge stability diagram for the benzene SET. The colors show the number of charge states in the bias window for a given gate voltage. The color map is: blue (0), light blue (1), green (2), orange (3), red (4).

CONCLUSION

This tutorial has illustrated how single-electron transistor systems can be studied with Atomistix ToolKit, and demonstrated how scripting, combined with the functionality of the graphical user interface, provides a powerful combination for setting up advanced geometries, automating the calculations, and analyzing and presenting the results.

BIBLIOGRAPHY

- [1] K. Stokbro, *J. Phys. Chem. C*, **114**, 20461, 2010. ([link](#))
- [2] S. G. Lias, J. E. Bartmess, J. E. Liebman, J. L. Holmes, R. D. Levin, W. G. Mallard, *J. Phys. Chem. Ref. Data*, **17 (suppl 1)**, 1988.
- [3] S. L. Murov, I. Carmichael, G. L Hug, *Handbook of Photochemistry*, 1993.
- [4] J. C. Rivière, *Appl. Phys. Lett.*, **8**, 172, 1966.

INDEX

A

affinity energy, 8
affinity energy, experimental, 8
ATK Reference Manual, 2

C

charge stability diagram, 5, 10, 11, 25
charge stabilization, 18
charging energy, 5, 10
clone object, 20
coherent transport, 3
Coulomb blockade, 1, 13

D

Database tool, 6
dielectric substrate, 16

E

electron affinity, 4
electrostatic potential, 20

G

gate coupling constant, 11, 24

H

HOMO, 8

I

incoherent transport, 1
induced potential, 20
ionization energy, 8
ionization energy, experimental, 8

L

LUMO, 4, 8

M

metallic region, 16
molecular energy spectrum, 17

N

Neumann boundary conditions, 17

S

Script Generator, 7
sequential tunneling, 3
single-electron transistor, 1
spatial regions, 15

T

tunneling criterion, 4

W

weak coupling regime, 4
work function gold, 11