

# ATK Tutorial for Molecular Devices

Calculate and analyze a molecular device configuration

---

---

# **ATK Tutorial for Molecular Devices: Calculate and analyze a molecular device configuration**

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. Setting up and running the zero-bias calculation .....	2
Input of the device configuration .....	2
Setting up the calculation with the Script Generator .....	2
3. Analysis of the zero-bias results .....	5
The transmission spectrum and density of states .....	5
The Molecular Projected Self-Consistent Hamiltonian (MPSH) .....	6
Transmission eigenvalues and eigenstates .....	9
Energy dependent LDDOS .....	14
4. I-V characteristics .....	17
Setting up the calculation .....	17
Calculating the I-V curve .....	18
Transmission spectrum at 1.6 V bias .....	18
The LDDOS at 1.6 V .....	19
Calculating the voltage drop .....	20
Bibliography .....	23

# CHAPTER 1. INTRODUCTION

---

This tutorial focuses on the calculation and analysis of a molecular device. The device consists of a dithiol benzene (DTB) molecule (sometimes also called benzene dithiol, BDT) in contact with two gold (111) surfaces. This is a classic molecular device, sometimes called the fruit fly of molecular electronics, and it has been studied in a large number of publications, see for instance [1], [2] and references therein.

Some of the calculations in this tutorial are quite time-consuming; for a quicker introduction to quantum transport studies with ATK, see the [ATK Tutorial for Device Configurations](#).

It is assumed that you are familiar with the general work flow of VNL, as described in [the basic ATK Tutorial](#).


The DFT model in Atomistix ToolKit (ATK) is the underlying calculation engine used in 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](#).

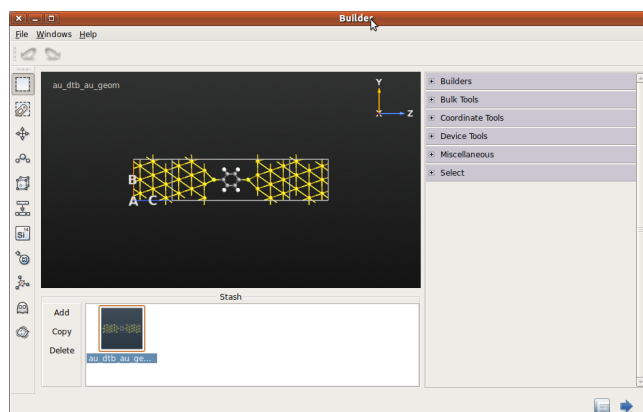
## CHAPTER 2. SETTING UP AND RUNNING THE ZERO-BIAS CALCULATION

---

### INPUT OF THE DEVICE CONFIGURATION

---

Start up VNL and open the **Builder** by left-clicking the icon  in the VNL tools bar. Select **Add** → **Add from Files** in the stash menu bar. This will open a file browser in the VNL examples directory. Select the file `au_dtb_au_geom.py` and import it into the builder.



#### **Note**


The geometry of the device can also be build from scratch by following the [builder-tutorial](#) available at the [QuantumWise](#) website.

#### **Tip**

The geometry was not relaxed. This is not critical for the concepts we will discuss in this tutorial, but it is an important step to consider for any real calculations with ATK. To learn how to relax a device geometry, see the [ATK Tutorial for Device Configurations](#).

### SETTING UP THE CALCULATION WITH THE SCRIPT GENERATOR

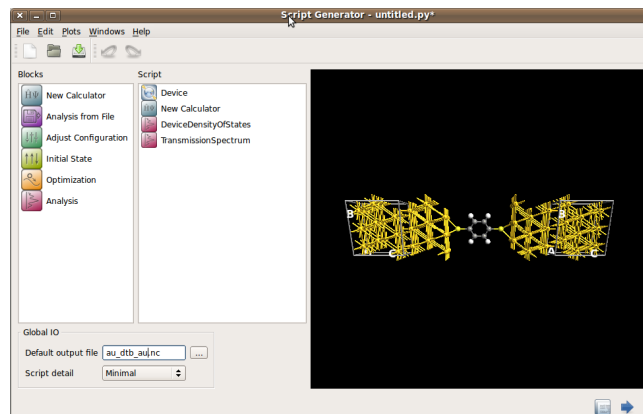
---

Now send the geometry to the **Script Generator** by pressing the “Send to” button  in the lower right-hand corner of the Builder and select **Script Generator** from the pop-up menu.

---

In the **Script Generator** add the following script elements by double-clicking them in the "Blocks" panel:

1. **New Calculator**
2. **Analysis/DeviceDensityOfStates**
3. **Analysis/TransmissionSpectrum**
4. Also, change the default filename to `au_dtb_au.nc`



Next open the **New Calculator** block by double-clicking it in the "Script" panel, and change the following settings:

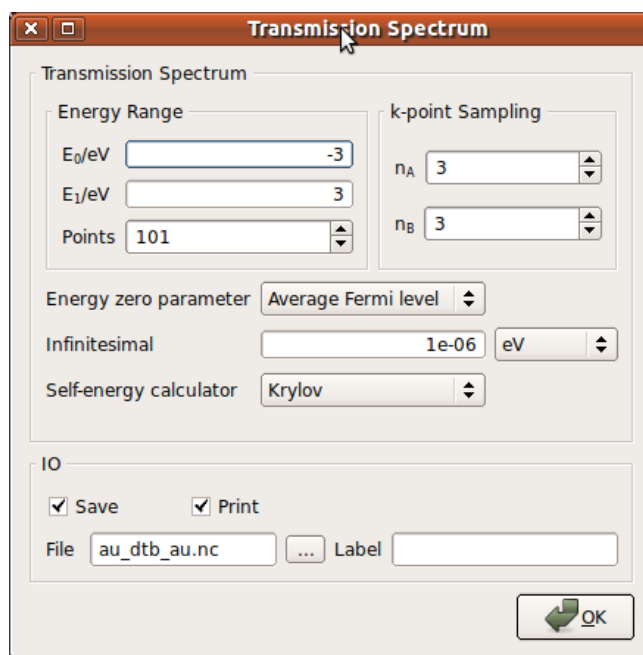
1. set the k-point sampling to  $3 \times 3 \times 50$  points;
2. change the basis set for gold to `SingleZetaPolarized`, to save some calculation time;

Open (double-click) the **DeviceDensityOfStates** block and change the following settings


1. Set the k-point sampling to  $3 \times 3$  points.
2. Set the energy range to  $(-3, 3)$  eV.
3. Set the self-energy calculator to `Krylov`.

Open the **TransmissionSpectrum** block and make the same modifications, i.e.

1. Set the k-point sampling to  $3 \times 3$  points.
2. Set the energy range to  $(-3, 3)$  eV.
3. Set the self-energy calculator to `Krylov`.



Finally, save the calculation script into the file `au_dtb_au.py`.

To run the script on your local machine, send it to the **Job Manager** by pressing the “Send to” button  in lower right-hand corner of the Script Generator and select Job Manager from the pop-up menu. In the Job Manager, press **Run Queue** to launch the job. On a modern, single-node computer the script should take around an hour to execute.

The calculation speed can be increased greatly by using a parallel computer. For information about how to run jobs in parallel, see the [parallel tutorial on our website](#).

## CHAPTER 3. ANALYSIS OF THE ZERO-BIAS RESULTS

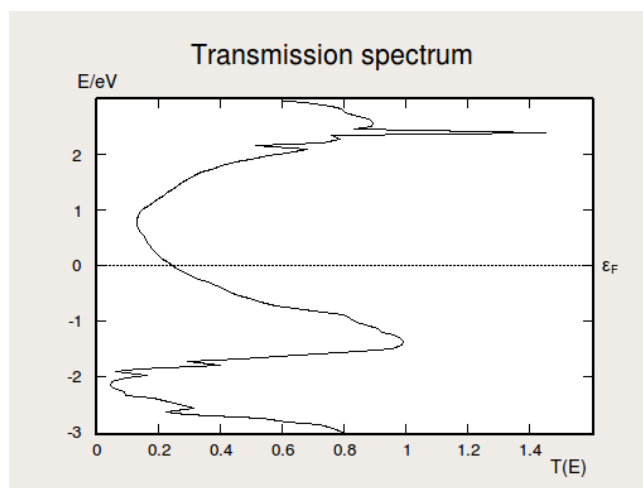
---

In this chapter you will extract data from the zero-bias calculation. You will first compare the transmission spectrum with the Partial Device Density Of States (PDDOS) of the phenylene ring. Subsequently, you will calculate the Molecular Projected Self-Consistent Hamiltonian (MPSH), and rationalize the transmission in terms of the MPSH states. To this end you will also calculate the transmission eigenstates. Finally, you will calculate the energy-dependent Local Density of States (LDOS), in order to obtain a spatial view of the energy levels.

### THE TRANSMISSION SPECTRUM AND DENSITY OF STATES

---

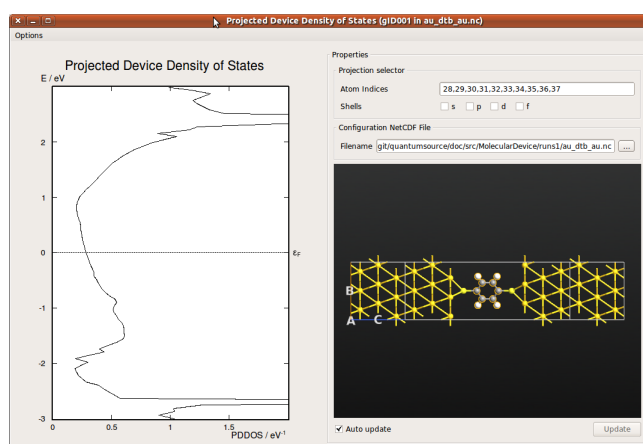
First take a look at the analysis objects calculated in the previous section. Locate the file `au_dtb_au.nc` in the VNL file browser, select the "Transmission Spectrum" object in the Result Browser (upper right panel), and finally click the Plot "Show" button in the lower panel, the Result Viewer.



**Figure 3.1:** The transmission spectrum with no applied bias.

There are three broad peaks, at around -3, -1.5 and 2.5 eV. Also note the narrow peak at 2.4 eV which rises above  $T(E) = 1$ . In the following you will investigate the origin of these peaks.


Next select the Device Density of States object in the NC file, and click the PDDOS "Show" button to open the PDDOS analyzer window. Select all carbon and hydrogen atoms by enclosing them with a rectangle using **left** mouse button (or select the individual atoms with Ctrl **left-click**). If you zoom into the figure by drawing a rectangle with the left mouse button, you should now see the following plot.



Note the similarity in the peak structure of the PDDOS and the transmission spectrum, indicating that there is a clear correspondence between the energy levels on the phenylene ring and the transmission spectrum.

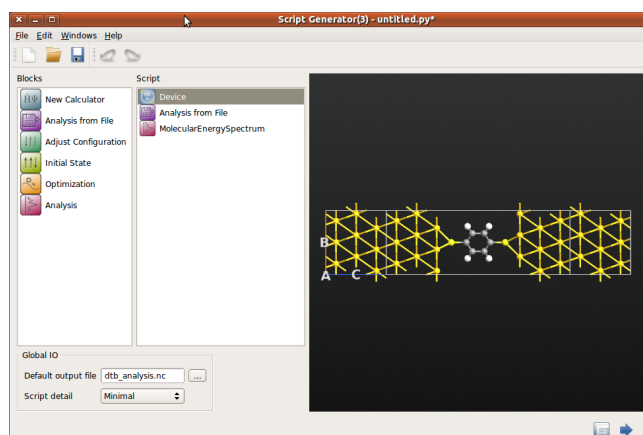
## THE MOLECULAR PROJECTED SELF-CONSISTENT HAMILTONIAN (MPSH)

To understand the energy levels of the phenylene ring in the presence of the surrounding electrodes, you will now calculate the MPSH states. The MPSH states are obtained by diagonalizing the molecular part of the full self-consistent Hamiltonian [1].

Select the `au_dtb_au.nc` in the VNL main window file browser, and drag and drop the “Device Configuration `gID000`” onto the Scripter  in the VNL tool bar.

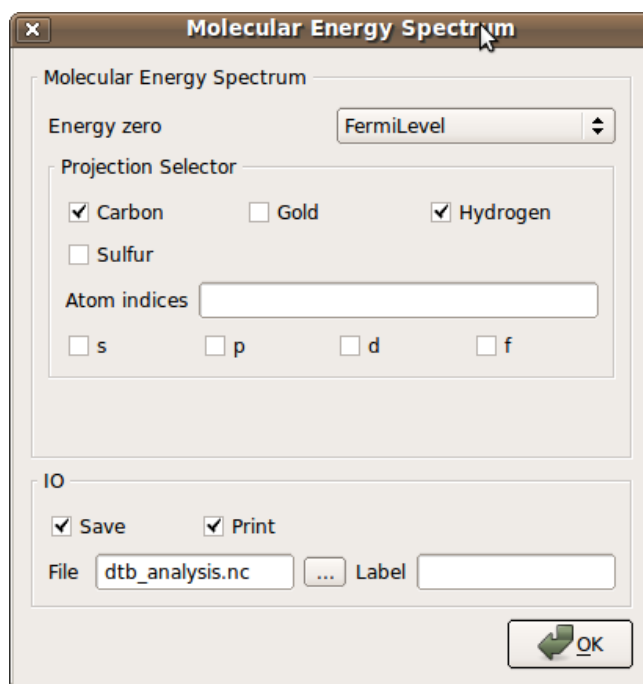
This will open the **Script Generator** tool.

1. Delete the **New Calculator** Block (select it and press the Delete key).
2. Double-click **Analysis from File** to add it to the script.
3. Add **Analysis/MolecularEnergySpectrum**.
4. Change the output filename to `dtb_analysis.nc`.

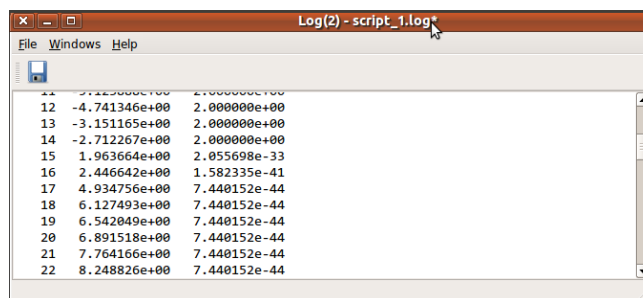


Open the **Analysis from File** script block and select `au_dtb_au.nc` from the file browser.

Open the **MolecularEnergySpectrum** block and select projection on carbon and hydrogen.



Send the script to the **Job Manager** and execute it (the execution only takes a few seconds). Inspect the MPSH energies in the log file.

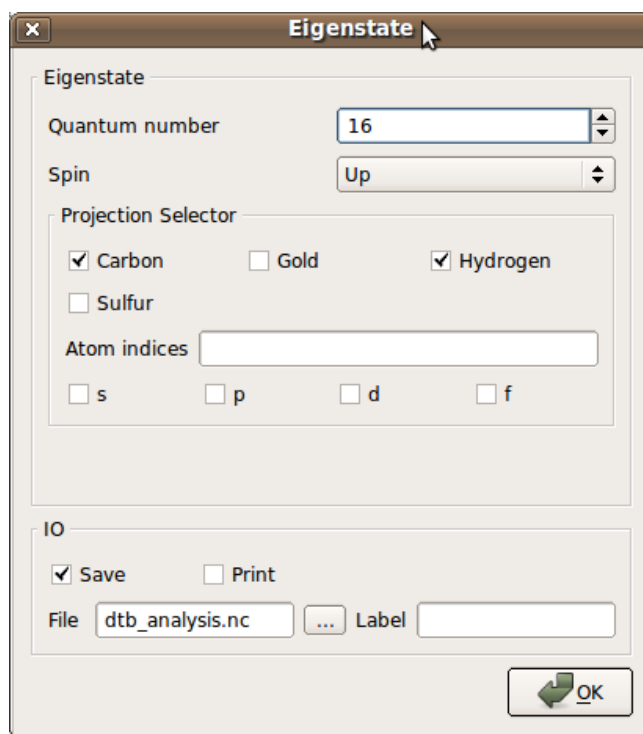


Note that state 13, 14, 15 and 16 have energies in the range relevant for the transmission spectrum. In the following you will investigate the symmetry of these states.

To calculate the eigenstates of the MPSH levels, reopen the last **Script Generator** (it is available from the windows menu at the top of each VNL window, if you didn't close it).

Add four **Analysis/EigenState** blocks.

Open each **EigenState** script block and select projection on carbon and hydrogen. Set the **Quantum number** to 13, 14, 15, 16 in each respective block.

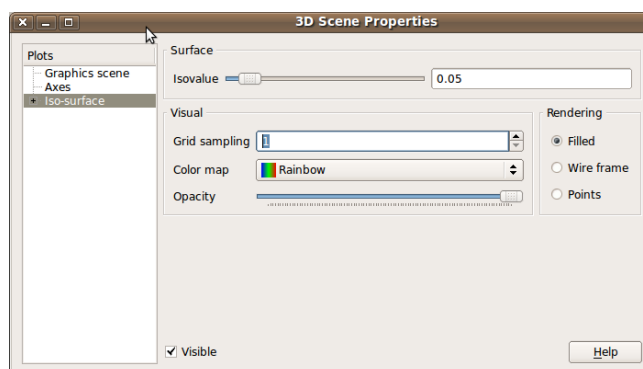


You may delete the **MolecularEnergySpectrum** block, it is no longer needed (select it and press the Delete key).

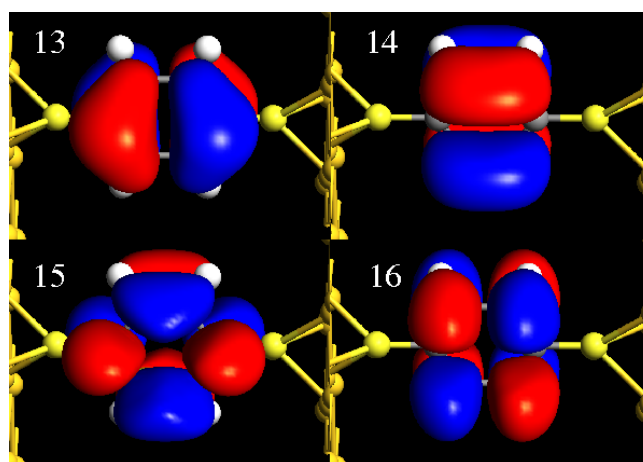
Send the script to the **Job Manager** and execute the script.

Select `dtb_analysis.nc` in the file browser, and choose the first eigenstate in the result file, and click the button **Isosurface>Show**. In the **Viewer** which opens up, open the Plot>Properties menu and select the **Isosurface** in the Plot tree.

1. Set the isovalue to 0.05.
2. Set the grid sampling to 1 for improved graphical quality.



To visualize the eigenstate together with geometry, drag and drop the file `au_dtb_au.py` onto the **Viewer**.



**Figure 3.2:** Isosurface plots of MPSH eigenstates 13, 14, 15 and 16 using an isovalue of 0.05.

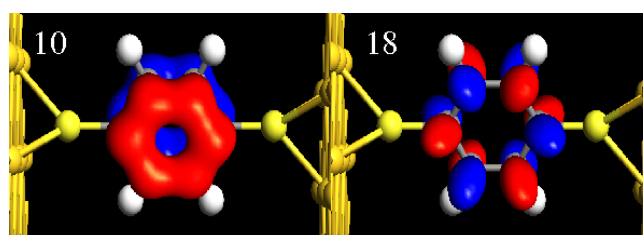
**i Tip**

To ensure the same view of the 4 eigenstates it is convenient to produce all plots in the same viewer. To do this drag and drop each eigenstate object into the same viewer. This will produce a contour plot of each eigenstate.

To generate isosurfaces instead, select from the menu **Plots** → **Show/hide** → **Eigenstate** → **Isosurface** for each eigenstate. Hide objects by removing their tick in the **Plots** → **Show/hide** menu.

Note that all the states are anti-symmetric in the plane of the benzene ring. This means that they are the  $\pi$ -orbitals of the benzene ring, i.e. they are linear combinations of the p-orbitals on each carbon atom which are perpendicular to the plane of the molecule. There are 6 such p-orbitals which means that there are 6  $\pi$ -orbitals.

By visually inspecting the symmetry of the different eigenstates the remaining two  $\pi$ -orbitals are identified as state 10 and 18. These are visualized in [Figure 3.3](#).



**Figure 3.3:** Isosurface plots of MPSH eigenstates 10 and 18 using an isovalue of 0.15.

**🔍 Note**

There are 6 electrons in the  $\pi$ -band, corresponding to that lowest 3 states are occupied. We may therefore denote state 14 the HOMO and state 15 the LUMO.

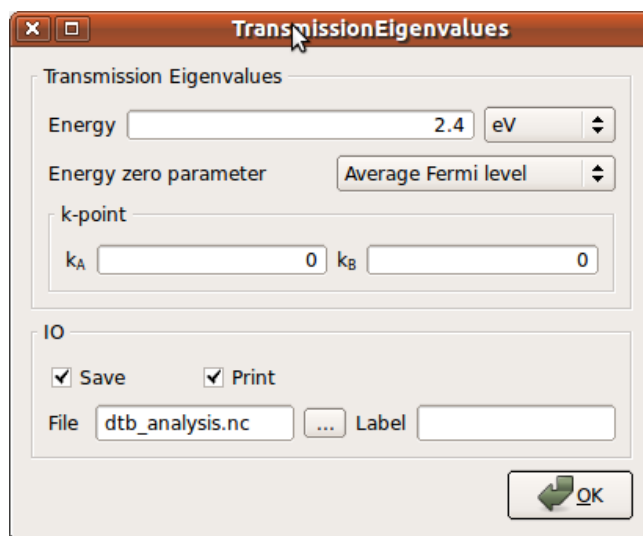
## TRANSMISSION EIGENVALUES AND EIGENSTATES

In this chapter you will analyze the transmission eigenstates at the peak positions in the transmission spectrum, i.e. at energies -3, -1.5 and 2.4 eV.

Re-open the **Script Generator** and delete the **Eigenstate** blocks.

Instead, add three **Analysis/TransmissionEigenvalues** blocks.

Set the energies to -3, -1.5 and 2.4 eV, into the respective **TransmissionEigenvalues** blocks.



Execute the script using the **Job Manager** and inspect the log file.

The transmission eigenvalues are obtained by diagonalizing the transmission matrix. The number of eigenvalues indicates the number of individual channels through the molecule, and the eigenvalue shows the strength of each channel. The eigenvalues are the true transmission probabilities, and thus lie in the interval [0,1]. If several channels are available at a particular energy, their sum - and hence the transmission coefficient at this energy - may however be larger than 1.

Most of the eigenvalues are very small and can be neglected. The two largest eigenvalues at each energy is listed below:

```
| Energy                = -3.000000e+00 eV |
| -----              |
| Number of transmission modes = 21      |
+-----+
Eigenvalues (Up):
 8.108561e-01
 1.734352e-02
```

```
| Energy                = -1.500000e+00 eV |
| -----              |
| Number of transmission modes = 7       |
+-----+
Eigenvalues (Up):
 7.993160e-01
 2.220177e-02
```

```
| Energy                = 2.400000e+00 eV |
| -----              |
| Number of transmission modes = 6      |
+-----+
Eigenvalues (Up):
 9.614759e-01
 6.840315e-01
```

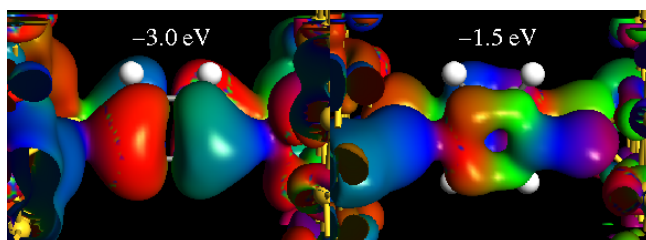
---

Note that at the first two energies there is only one dominating eigenstate, while at the last energy there are two dominating eigenvalues. (The fact that they do not sum up to the total transmission spectrum  $T(E)$  will be discussed in the [next section!](#))

The next step is to visualize the transmission eigenstates for the first two peak energies, where there is only one significant eigenvalue.

- Re-open the **Script Generator** and delete the **TransmissionEigenvalues** blocks.
- Add two **Analysis/TransmissionEigenstates** blocks.
- Enter the energies -3 and -1.5 into the **Analysis/TransmissionEigenstates** blocks. Leave the quantum number at 0, corresponding to the *highest* eigenvalue.
- Execute the script via the **Job Manager**.

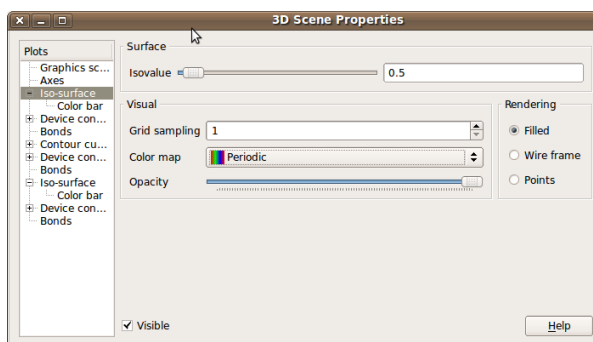
The objects will again be saved in the file `dtb_analysis.nc` and can be visualized using the same approach as for the eigenstates. Below are shown isosurfaces of the two transmission eigenstates.



**Figure 3.4:** Isosurfaces of the transmission eigenstates for the only significant transmission eigenchannel at  $k=(0,0)$  and energies -3.0 eV and -1.5 eV.

**i Tip**

The transmission eigenstate is a complex wave function. The isosurface shows the absolute value of the wave function while the color of the isosurface indicates the phase. Since the phase is periodic, it is best to use a periodic color map. The color map can be chosen from the plot **Properties**; below is shown the settings used for producing the plots above.



The transmission eigenstate at -3.0 eV has a clear resemblance with MPSH state 13, i.e. the HOMO-1 state. The transmission eigenstate at -1.5 eV is also a  $\pi$ -state,

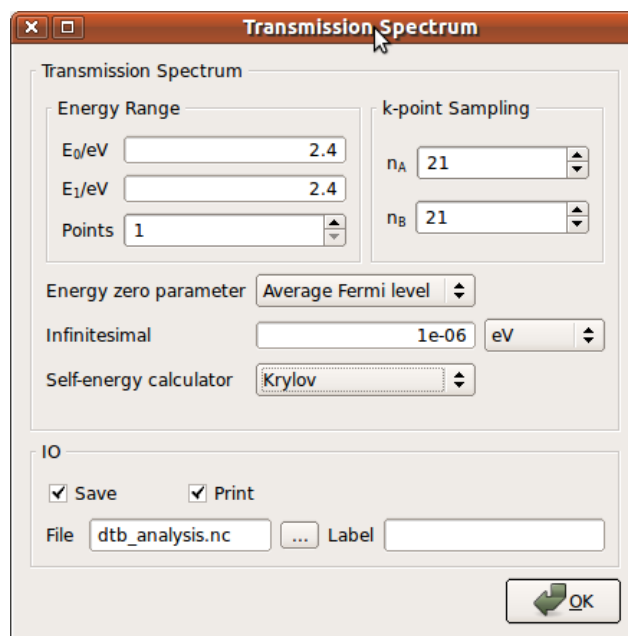
---

## K-DEPENDENT TRANSMISSION

You will now analyze the transmission at 2.4 eV. Previously, you found that there are two important transmission eigenstates at this energy with transmission probabilities 0.961 and

0.684. The sum of these eigenvalues give a total transmission of 1.645, is different from the value 1.45 obtained from inspecting Figure 3.1. This is because Figure 3.1 shows the k-point averaged transmission. This discrepancy suggests that there must be a strong k-dependence of the transmission at this energy. You will now calculate the k-dependent transmission coefficients at this energy.

- Re-open the **Script Generator** and delete the **TransmissionEigenstates** blocks.
- Add an **Analysis/TransmissionSpectrum** block.
- Open the **TransmissionSpectrum** block and set
  1. energy start and end equally to 2.4 eV,
  2. number of energy points to 1,
  3. k-point sampling to 21 x 21 points,
  4. Self-energy calculator to Krylov.



- Execute the script using the **Job Manager**.

When the calculation has finished you can visualize the k-dependent transmission with the script below

```
#read the transmission spectrum
transmission = nload('dtb_analysis.nc',TransmissionSpectrum)[-1]
#get the transmission spectrum
trans_coeff = transmission.transmission()
#select spin up component
T_uu = trans_coeff[0]
# shape of the components are [energy_points, k_points]
(n_e,n_k) = T_uu.shape
# assume equal many k_points in A and B directions
n_A=n_B=numpy.sqrt(n_k)
#reshape the arrays for plotting, assume only one energy point
T_uu = T_uu.reshape(n_A,n_B)
#get the k-points, x component
```

```

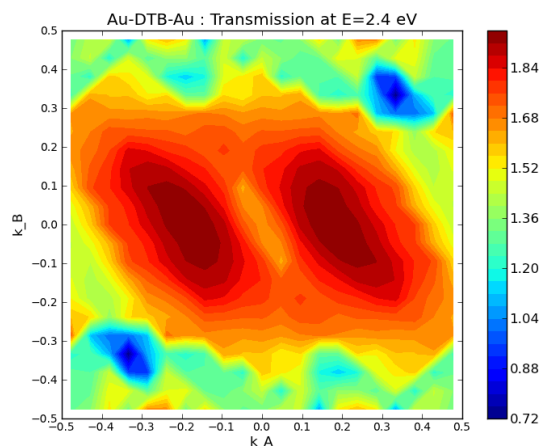
K_A = transmission.kpoints()[:,0]
K_A = K_A.reshape(n_A,n_B)
#get the k-points, y component
K_B = transmission.kpoints()[:,1]
K_B = K_B.reshape(n_A,n_B)

import pylab
#plot the transmission
pylab.figure()
pylab.xlabel('k_A',fontsize=12,family='sans-serif')
pylab.ylabel('k_B',fontsize=12,family='sans-serif')
pylab.contourf(K_A,K_B,T_uu,40)
pylab.colorbar()
pylab.axis([-0.5,0.5,-0.5,0.5])
pylab.yticks(numpy.arange(-0.5,0.51,0.1),fontsize=10)
pylab.xticks(numpy.arange(-0.5,0.51,0.1),fontsize=10)
pylab.title('Au-DTB-Au : Transmission at E=2.4 eV')
pylab.savefig('kdependent_transmission.png',dpi=100)

pylab.show()

```

Save the file on your computer and execute it by dropping the file on the **Job Manager**. (If ATK cannot find the data file, `dtb_analysis.nc`, you might need to set the absolute path for the file in the script.) You should now see the following plot



**Figure 3.5:** k-dependent transmission at E=2.4 eV.

The contour plot of the k-dependent transmission above has a pronounced peak at  $(k_A, k_B) = (0.18, 0.0)$ , with a transmission coefficient of  $\sim 2$ . In the computation of the transmission spectrum earlier, because we did not have a dense enough k-point sampling, and hence the transmission was underestimated, i.e. we obtained 1.45 while the 21x21 mesh gives 1.58.



### Note

This shows the importance of carefully checking the convergence of the results in the k-point sampling of the transmission spectrum. There is no reason to assume that the same number of points used to get a correct electron density (i.e. the k-points used in the self-consistent calculation) also give an accurate transmission.

---

## K-DEPENDENT TRANSMISSION EIGENCHANNELS

---

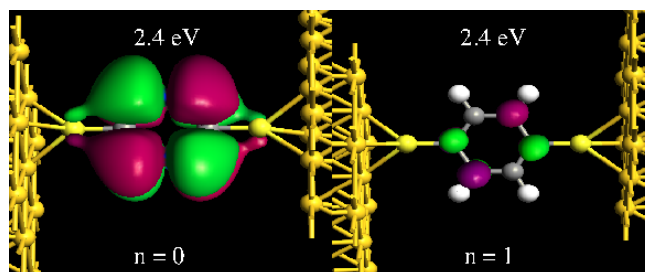
You will now calculate the transmission eigenstates at the k-point with the strongest transmission. We can choose either point  $(-0.18,0)$  or  $(0.18,0)$ ; they are equivalent by time-reversal symmetry.

- Re-open the **Script Generator** and delete the **TransmissionSpectrum** block.
- Add an **Analysis/TransmissionEigenvalues** block.
- Add two **Analysis/TransmissionEigenstates** blocks.
  1. Open the **TransmissionEigenvalues** block, set the energy to 2.4 eV, and the k-point to  $(0.18,0)$
  2. Open the first **TransmissionEigenstates** block, set the energy to 2.4 eV, and set the k-point to  $(0.18,0)$
  3. Open the second **TransmissionEigenstates** block, set the energy to 2.4 eV, set the k-point to  $(0.18,0)$  and the quantum number to 1.



- Execute the script using the **Job Manager**.

Inspect the log file to find the transmission eigenvalues, 0.995 and 0.955 and plot the corresponding eigenstates as shown below.



**Figure 3.6:** Isosurfaces of value 0.5 of the two highest transmission eigenstates at energy 2.4 eV.

By comparing to the MPSH eigenstates, we can find that the  $n=0$  eigenstate arises from transmission through MPSH state 16. Since the state has a nodal plane it couples very weakly with the electrodes, and gives rise to a narrow peak in the transmission spectrum.

The  $n=1$  eigenstate also has 2 nodal planes, however, does not have a direct resemblance with neither MPSH 15 or 16, which are the MPSH states with 2 nodal planes. Thus, the  $n=1$  state must therefore be a linear combination of MPSH 15 and 16. This state has a stronger coupling with the electrodes, and gives rise to the broad peak in the transmission spectrum around 2.4 eV.

---

## ENERGY DEPENDENT LDDOS

---

In this section you will calculate the energy-dependent Local Device Density of States (LDDOS). It is related to the PDDOS calculated above, however, instead of projecting the DDOS onto orbitals, the DDOS is projected in real space.

- Re-open the **Script Generator** and delete the previous analysis blocks.
- Add an **Analysis/LocalDeviceDensityOfStates** block.
- Open the **LocalDeviceDensityOfStates** block and set the k-point sampling to 3 x 3 and the self-energy calculator to Krylov.
- Change the output filename to `lddos.nc`

The aim is to calculate the LDDOS in the energy range -3,3 eV with steps of 0.2 eV. This gives 31 spectra, and to set this up in the Script Generator will require that you add 30 more LocalDeviceDensityOfStates blocks. Instead of this tedious task, you will use scripting.

Use "Send To"  to transfer the script to the **Editor**.

In the Editor, add the following lines before the **LocalDeviceDensityOfStates** block.

```
energies = numpy.linspace(-3,3,31)
for e in energies:
```

Indent the rest of the script by selecting the lines and pressing the TAB key. Finally, change the energy keyword into

```
energy=e*eV,
```

The script should now look like the following

```
device_configuration = nload('au_dtb_au.nc', object_id='gID000')[0]
# make list of energies to be used for ldos
energies = numpy.linspace(-3,3,31)
# calculate ldos for each energy in the list
for e in energies:
    local_device_density_of_states = LocalDeviceDensityOfStates(
        configuration=device_configuration,
        energy=e*eV,
        kpoints=MonkhorstPackGrid(3,3),
        contributions=All,
        energy_zero_parameter=AverageFermiLevel,
        infinitesimal=1e-06*eV,
        self_energy_calculator=KrylovSelfEnergy(),
        spin=Spin.Sum,
    )
    nsave('lddos.nc', local_device_density_of_states)
```

The script will take around 30 min to execute on a serial machine. You may execute it using the **Job Manager**.

To plot the data use the script below

```
# import list with lddos
lddos_list = nload('lddos.nc', LocalDeviceDensityOfStates)
#Find the z-spacing
dX, dY, dZ = lddos_list[0].volumeElement().convertTo(Ang)
dz = dZ.norm()
shape = lddos_list[0].shape()
z = dz * numpy.arange(shape[2])
```

```

# calculate average lddos along z for each spectrum
energies = []
lddos_z = []
for lddos in lddos_list:
    energies = energies + [lddos.energy().inUnitsOf(eV)]
    avg_z = numpy.apply_over_axes(numpy.mean,lddos[:, :, :], [0,1]).flatten()
    lddos_z = lddos_z + [avg_z]

# plot as contour plot
# make variables
energies = numpy.array(energies)
X, Y = numpy.meshgrid(z, energies)
Z = numpy.array(lddos_z).reshape(numpy.shape(X))

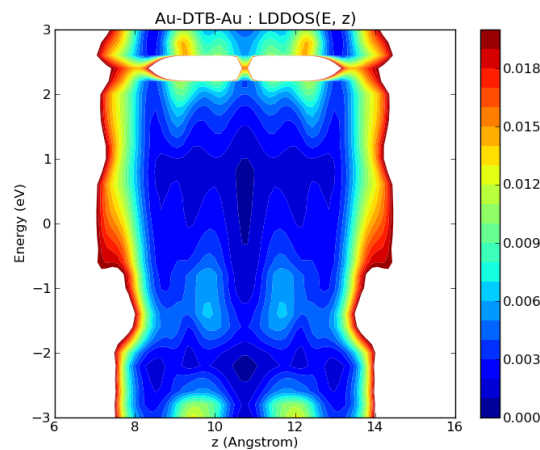
import pylab
#plot the LDDOS(E, z)
pylab.xlabel('z (Angstrom)', fontsize=12, family='sans-serif')
pylab.ylabel('Energy (eV)', fontsize=12, family='sans-serif')
contour_values = numpy.linspace(0,0.02,21)
pylab.contourf(X, Y, Z, contour_values)
pylab.colorbar()
pylab.axis([6,16,-3, 3])

pylab.title('Au-DTB-Au : LDDOS(E, z)')
pylab.savefig('lddos.png', dpi=100)

pylab.show()

```

Execute the script, and you should see the following plot.



**Figure 3.7:** Local Device Density of States (LDDOS) averaged over  $x$  and  $y$ , plotted along  $z$  as a function of energy. In the white regions the LDDOS is very high, corresponding to metallic regions.

The LDDOS shows a high density in the metallic gold electrodes (white regions) and the central colored region corresponds to the vacuum gap. In the central region there are three peaks as a function of energy, positioned at -3, -1.4, 2.4 eV, in agreement with the peaks in the transmission spectrum.

If you add more LDDOS objects (i.e. more points in energy, the script above uses 31) you can enhance the energy resolution.

## CHAPTER 4. I-V CHARACTERISTICS

---

In this chapter you will calculate the I-V characteristics of the Au-DTB-Au molecular device, and analyze the device for an applied bias of 1.6 V.

### SETTING UP THE CALCULATION

---

To calculate an I-V curve, a self-consistent calculation for each voltage is required. The following script perform such a calculations for voltages, 0.2, 0.4, ..., 3.0 V.

```
#read in the 0 V configuration
device_configuration = nload("au_dtb_au.nc",DeviceConfiguration)[0]
calculator = device_configuration.calculator()

#make a fast krylov self energy calculator
device_algorithm_parameters = DeviceAlgorithmParameters(
    self_energy_calculator_real=KrylovSelfEnergy(save_self_energies=True,
                                                  lambda_min=0.1))

# Define bias voltages
voltage_list= 0.2 *numpy.arange(1,16)*Volt

#make loop
for voltage in voltage_list:
    # Set new calculator with modified electrode voltages on the configuration
    # use the self consistent state of the old calculation as starting input.
    device_configuration.setCalculator(
        calculator(electrode_voltages=(-0.5*voltage, 0.5*voltage),
                  device_algorithm_parameters=device_algorithm_parameters),
        initial_state=device_configuration)

    # Calculate the transmission spectrum
    transmission_spectrum = TransmissionSpectrum(
        configuration=device_configuration,
        energies=numpy.linspace(-3,3,101)*eV,
        kpoints=MonkhorstPackGrid(3,3),
    )

#save the results
nlsave('au_dtb_au.nc', device_configuration)
nlsave('au_dtb_au.nc', transmission_spectrum)
```

The script loops over a list of bias voltages and performs a self-consistent calculation for each bias. The self-consistent state of the previous calculation is used as initial guess for the next higher bias. The full calculation of the I-V curve will take about 20 times longer than the zero-bias calculation. If available, it is recommended to use a parallel computer. If you do not have access to a parallel computer, you may start the script on your local machine using the **Job Manager** and run it over-night.

---

## CALCULATING THE I-V CURVE

---

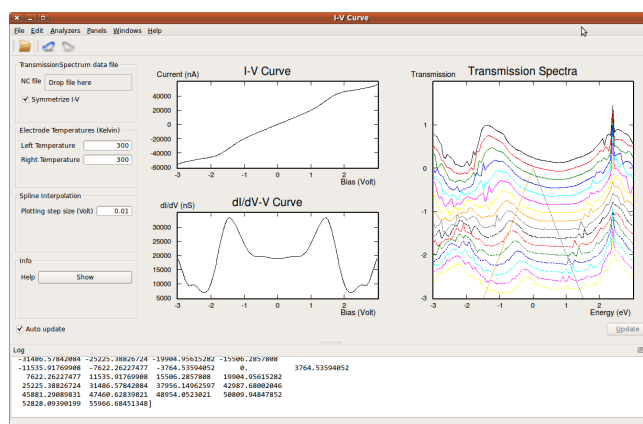
In this chapter you will calculate the I-V curve from the transmission spectrum at each bias.

Launch the **Custom Analyzer** tool  from the toolbar in the main VNL window.

Start the built-in I-V curve analyzer by selecting **Analyzers** → **I-V Curve** from the **Custom Analyzer** top menu bar.

Next, drag and drop the file `au_dtb_au.nc` onto the NC file drop zone.

Tick off **symmetrize I-V** and you should see the following plot



The left-most plots show the I-V and dI/dV-V curves. The curves are obtained by importing each transmission spectrum in the nc file and integrating the transmission coefficient over the bias window. In the input field to the left it is possible to change the electron temperature in the electrodes, and thereby the Fermi distribution used, when performing the integration over the bias windows.

The right-most plot shows the transmission spectra for different bias voltages, the spectra have been displaced vertically for clarity. The wedge signifies the part of the transmission spectra which are inside the bias window.



### Note

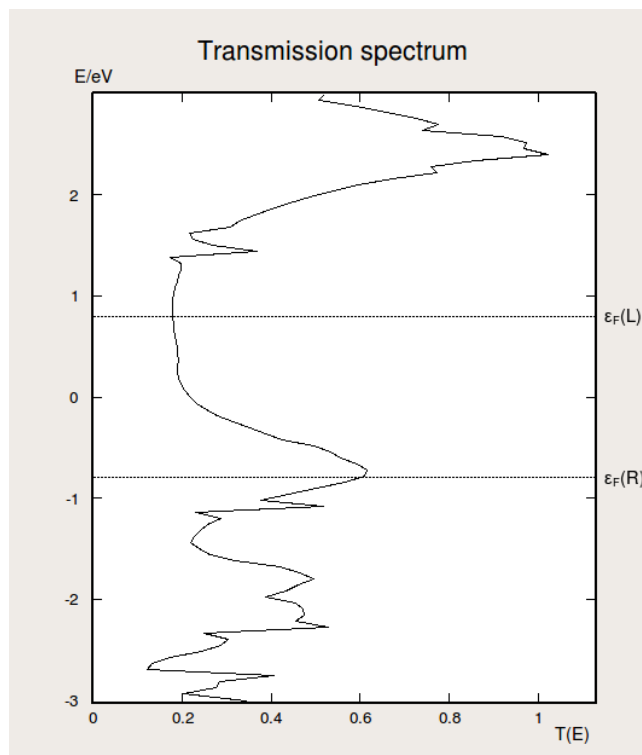
The peak in the dI/dV at 1.6 V arises from a resonance in the energy transmission spectrum entering the bias window at this voltage.

---

## TRANSMISSION SPECTRUM AT 1.6 V BIAS

---

In the following you will analyze the transport properties at 1.6 V in a little more detail. To visualize the transmission spectrum, locate the file `au_dtb_au.nc` in the result browser, and open the transmission spectrum with ID `gID018`.



Note that the zero-bias peak at -1.5 eV appears to have split into two peaks at -0.8 and -2 eV. To further investigate the electronic structure for an applied bias of 1.6 V you will now calculate the LDDOS.

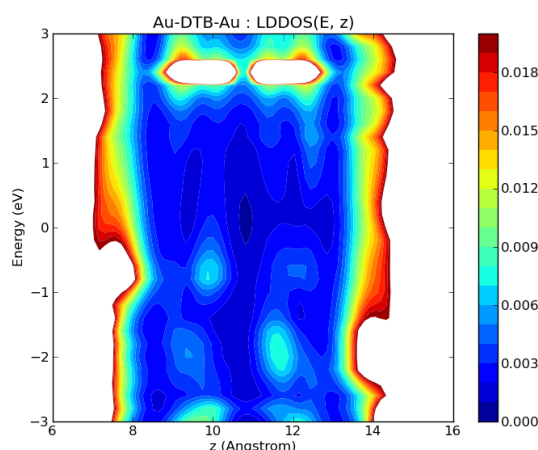
## THE LDDOS AT 1.6 V

Using the same approach as in [the section called "Energy dependent LDDOS"](#), calculate LDDOS( $E, z$ ) of the system at 1.6 V. Below is shown the script for calculating the LDDOS.

```
configuration = nload('au_dtb_au.nc', DeviceConfiguration)[8]

# make list of energies to be used for ldos
energies = numpy.linspace(-3,3,31)

# calculate ldos for each energy in the list
for e in energies:
    local_device_density_of_states = LocalDeviceDensityOfStates(
        configuration=configuration,
        energy=e*eV,
        kpoints=MonkhorstPackGrid(3,3),
        contributions=All,
        energy_zero_parameter=AverageFermiLevel,
        infinitesimal=1e-06*eV,
        self_energy_calculator=KrylovSelfEnergy(),
        spin=Spin.Sum,
    )
    nlsave(r'lddos16.nc', local_device_density_of_states)
```



**Figure 4.1:** Local Device Density of States (LDDOS) for an applied bias of 1.6 V. The LDDOS is averaged over  $x$  and  $y$  and plotted along  $z$  as a function of energy.

Compared with [Figure 3.7](#) the resonance at -1.5 eV has indeed split into two resonances, at -0.8 and -2 eV, as we saw already in the transmission spectrum above. From the LDDOS plot we see that the resonance at -0.8 is located towards the left electrode, while the resonance at -2 eV is located towards the right electrode. The splitting arises from the drop in the electrostatic potential through the molecule, as will be investigated in the next section.

## CALCULATING THE VOLTAGE DROP

You will now calculate the voltage drop and induced density for an applied bias of 1.6 V. Carry out the analysis by dropping the following script named `voltage_drop.py` on the **Job Manager**.

```
# Read the configurations
configuration_list = nload('au_dtb_au.nc', DeviceConfiguration)

#calculate the electrostatic potentials
potential_0V = ElectrostaticDifferencePotential(configuration_list[0])
potential_16V =ElectrostaticDifferencePotential(configuration_list[8])

# Calculate the voltage drop.
voltage_drop = potential_16V-potential_0V

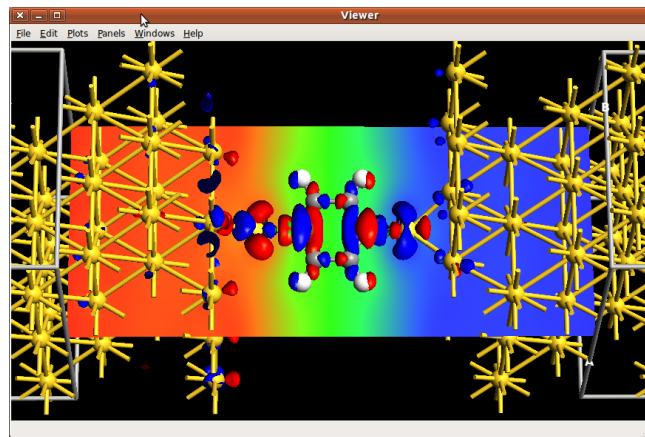
# Save the voltage drop to voltage_drop.nc
nlsave('voltage_drop.nc', voltage_drop, object_id='drop16V')

#calculate the difference densities
density_0V = ElectronDifferenceDensity(configuration_list[0])
density_16V =ElectronDifferenceDensity(configuration_list[8])

# Calculate the difference
density_diff = density_16V-density_0V

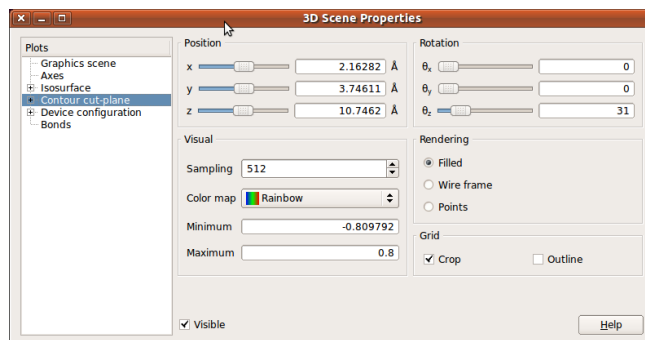
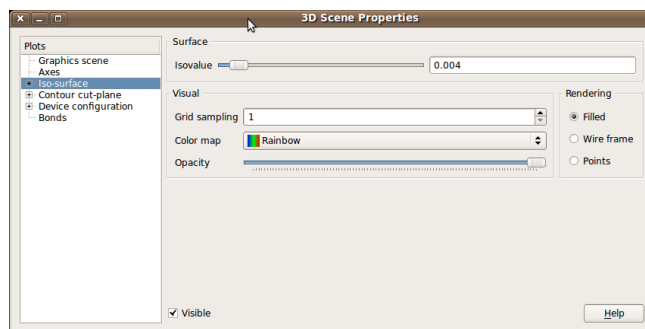
# Save the density difference to voltage_drop.nc
nlsave('voltage_drop.nc', density_diff, object_id='diff16V')
```

The result of this script is to save an `ElectrostaticDifferencePotential` object and an `ElectronDifferenceDensity` object into `voltage_drop.nc`. This potential corresponds to the potential drop across the device at 1.6 V, and the density is the induced density due to the applied bias. Visualizing the voltage drop as a contour plot and the induced density as an isosurface, the following plot can be obtained



**Figure 4.2:** Electrostatic potential drop at a bias of 1.6 V.

The plot was obtained by setting the following properties for the widgets



## THE VOLTAGE DROP ALONG THE MOLECULAR AXIS

It is useful to make a 1-d plot of the voltage drop. All 3D grids can be evaluated at a Cartesian coordinate, and in this way it is easy to plot the grids along different directions. The following script plots the potential drop along the molecular axis.

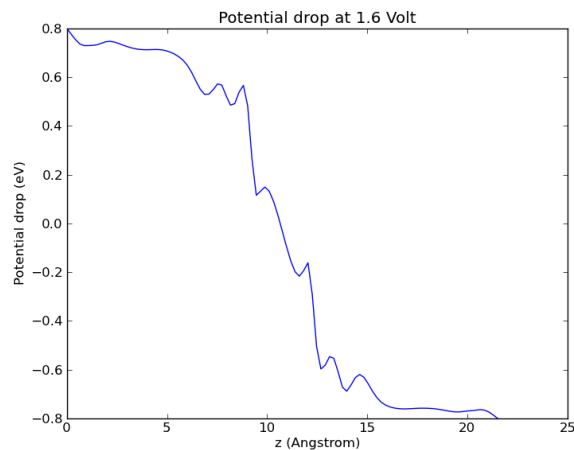
```
#read the potential drop
pot_drop = nload("voltage_drop.nc", object_id='drop16V')[0]

#define the line plot
x0 = 2.16295788*Ang
y0 = 3.74587078*Ang
z = numpy.linspace(0,21.4923367139, 101)*Ang

pot_z = numpy.array([ pot_drop.evaluate(x0,y0,z0).inUnitsOf(eV) for z0 in z])*eV
```

```
import pylab
pylab.plot(z.inUnitsOf(Ang), pot_z.inUnitsOf(eV))
pylab.xlabel('z (Angstrom)', fontsize=12, family='sans-serif')
pylab.ylabel('Potential drop (eV)', fontsize=12, family='sans-serif')
pylab.title('Potential drop at 1.6 Volt')
pylab.savefig('plot_drop16.png', dpi=100)
pylab.show()
```

Executing the script produces the following plot



**Figure 4.3:** Voltage drop along a line through the molecular axis.



### Note

The LDDOS maxima in [Figure 4.1](#) are located around 10 and 12 Angstrom. Inspection of [Figure 4.3](#) shows that there is a voltage drop of  $\sim 0.7$  eV between these points. Almost half of the total voltage drop of 1.6 V (the applied bias) takes place in this narrow space of 2 Angstrom. A drop in voltage corresponds to a quantum resistance, and this explains the observed splitting of the zero-bias resonance.

## BIBLIOGRAPHY

---

- [1] K. Stokbro, J. Taylor, M. Brandbyge, J.-L. Mozos, and P. Ordejón. *Comp. Mat. Sci.*, **27**, 151, 2003
- [2] K. S. Thygesen, and K. W. Jacobsen. *Chemical Physics*, **319**, 111, 2005