

# ATK Tutorial for Device Configurations

Setup, calculate and analyze a simple device configuration

---

# **ATK Tutorial for Device Configurations: Setup, calculate and analyze a simple 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. Input for a transport calculation .....	2
Device Configuration .....	2
The electrodes .....	4
The central region .....	5
Summary .....	6
3. Setting up a transport calculation with the Script Generator .....	7
Loading a device configuration to the Script Generator .....	7
Setting up the calculation .....	7
Running the script .....	9
Visualising the results .....	9
4. Further analysis of the results .....	11
Device density of states .....	11
Molecular Projected Self-Consistent Hamiltonian (MPSH) .....	14
Transmission eigenchannels .....	16
Several types of analysis in one script .....	18
5. I-V characteristic .....	19
Setting up the calculation .....	19
Calculating the I-V curve .....	20
Performing additional analysis using scripting .....	20
Performing an I-V scan with scripting .....	21
6. Building and optimizing the geometry .....	23
Setting up the electrode geometry .....	23
Determining the geometry of the central region .....	26
Optimization at finite bias .....	30
Index .....	33

# CHAPTER 1. INTRODUCTION

---

The ATK packages gives you access to a powerful set of modeling tools for investigating nano-scale systems, such as molecules, bulk and two-probe systems. The tools are based on the techniques

- **Density Functional Theory (DFT)**
- **Extended-Hückel Theory (EHT)**
- **Non-Equilibrium Green's Functions (NEGF)**

This tutorial introduces you to Atomistix ToolKit and the graphical interface Virtual NanoLab for its core functionality - transport calculations. The system you will investigate is a hydrogen molecule placed between two 1-dimensional lithium wires. For demonstration purposes, this system is selected in order to make calculations very light. For more interesting physical systems, you are referred to the more comprehensive tutorials available from our [website](#).

The **Atomistix ToolKit** (ATK) is the underlying calculation engine performing all calculations in **Virtual Nanolab** (VNL). 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. INPUT FOR A TRANSPORT CALCULATION

---

## DEVICE CONFIGURATION

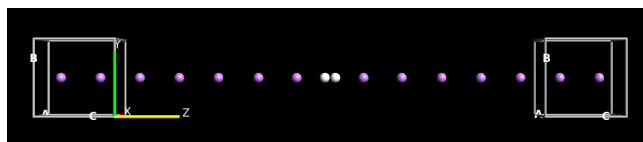
---

In order to perform a transport calculation you need to specify two things:

1. The geometry through which you wish to calculate the current. This is called the **central region**.
2. The **electrodes** that guide the current to and from the structure.

The **central region** and the **electrodes** are equally important for transport properties - if either of the **electrodes** or the **central region** are isolating then no current will flow. In ATK, a **central region** combined with **electrodes** is called a **device configuration**. The [last chapter](#), demonstrates how to build a **device configuration** using VNL, but for now you will use a pre-build device geometry, and be introduced to the concepts of the **central region** and the **electrodes**.

The example system used in this tutorial is a hydrogen molecule acting as an impurity in an otherwise periodic lithium wire. The system is shown below.



The above figure can be generated by the script

```
# Left electrode

# Set up lattice
vector_a = [6.0, 0.0, 0.0]*Angstrom
vector_b = [0.0, 6.0, 0.0]*Angstrom
vector_c = [0.0, 0.0, 6.22841]*Angstrom
left_electrode_lattice = UnitCell(vector_a, vector_b, vector_c)

# Define elements
left_electrode_elements = [Lithium, Lithium]

# Define coordinates
left_electrode_coordinates = [[ 3.          , 3.          , 1.5571025],
                              [ 3.          , 3.          , 4.6713075]]*Angstrom

# Set up configuration
left_electrode = BulkConfiguration(
```

```

    bravais_lattice=left_electrode_lattice,
    elements=left_electrode_elements,
    cartesian_coordinates=left_electrode_coordinates
)

# Right electrode

# Set up lattice
vector_a = [6.0, 0.0, 0.0]*Angstrom
vector_b = [0.0, 6.0, 0.0]*Angstrom
vector_c = [0.0, 0.0, 6.22841]*Angstrom
right_electrode_lattice = UnitCell(vector_a, vector_b, vector_c)

# Define elements
right_electrode_elements = [Lithium, Lithium]

# Define coordinates
right_electrode_coordinates = [[ 3.      , 3.      , 1.557102],
                               [ 3.      , 3.      , 4.671308]]*Angstrom

# Set up configuration
right_electrode = BulkConfiguration(
    bravais_lattice=right_electrode_lattice,
    elements=right_electrode_elements,
    cartesian_coordinates=right_electrode_coordinates
)

# Central region

# Set up lattice
vector_a = [6.0, 0.0, 0.0]*Angstrom
vector_b = [0.0, 6.0, 0.0]*Angstrom
vector_c = [0.0, 0.0, 33.1418095157]*Angstrom
central_region_lattice = UnitCell(vector_a, vector_b, vector_c)


# Define elements
central_region_elements = [Lithium, Lithium, Lithium, Lithium, Lithium, Hydrogen, Hydrogen,
                           Lithium, Lithium, Lithium, Lithium, Lithium]

# Define coordinates
central_region_coordinates = [[ 3.      , 3.      , 1.5571025 ],
                              [ 3.      , 3.      , 4.6713075 ],
                              [ 3.      , 3.      , 7.76171051],
                              [ 3.      , 3.      , 10.91847415],
                              [ 3.      , 3.      , 13.93149176],
                              [ 3.      , 3.      , 16.15851591],
                              [ 3.      , 3.      , 16.98337712],
                              [ 3.      , 3.      , 19.21030765],
                              [ 3.      , 3.      , 22.2234055 ],
                              [ 3.      , 3.      , 25.38002278],
                              [ 3.      , 3.      , 28.47050152],
                              [ 3.      , 3.      , 31.58470752]]*Angstrom

# Set up configuration
central_region = BulkConfiguration(
    bravais_lattice=central_region_lattice,
    elements=central_region_elements,
    cartesian_coordinates=central_region_coordinates
)

device_configuration = DeviceConfiguration(
    central_region,
    [left_electrode, right_electrode]
)

```

To visualize the structure, save the script, to a file, locate the file in the VNL main window and drag and drop it onto the Viewer .

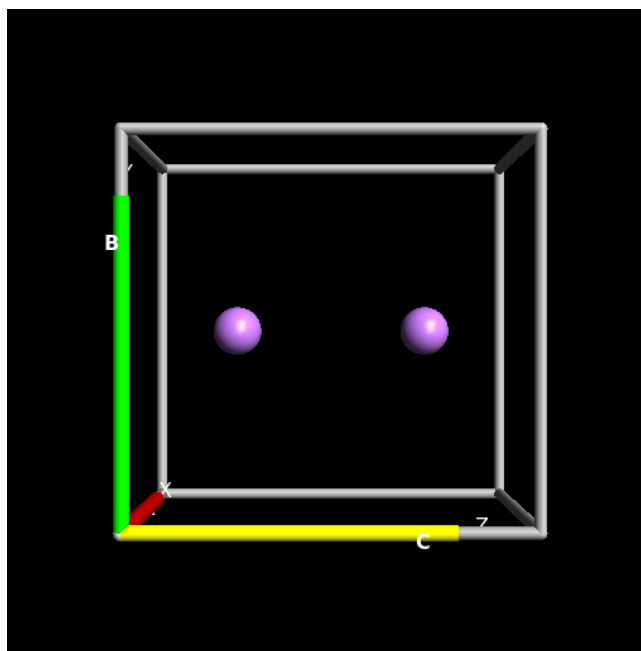
---

The script, defines the **central region** and two **electrode** regions, in the following you will learn how this is done.


## THE ELECTRODES

---

The **electrodes** are shown below



The two purple spheres represent two lithium atoms and the grey wire-frame box represents how the atoms should be repeated to create an infinite structure.

The figure can be produced by dragging the following part of the to the Viewer (select the text with the mouse and drag and drop it onto the Viewer icon ).

```
# Left electrode

# Set up lattice
vector_a = [6.0, 0.0, 0.0]*Angstrom
vector_b = [0.0, 6.0, 0.0]*Angstrom
vector_c = [0.0, 0.0, 6.22841]*Angstrom
left_electrode_lattice = UnitCell(vector_a, vector_b, vector_c)

# Define elements
left_electrode_elements = [Lithium, Lithium]

# Define coordinates
left_electrode_coordinates = [[ 3.      , 3.      , 1.5571025],
                              [ 3.      , 3.      , 4.6713075]]*Angstrom

# Set up configuration
left_electrode = BulkConfiguration(
    bravais_lattice=left_electrode_lattice,
    elements=left_electrode_elements,
    cartesian_coordinates=left_electrode_coordinates
)
```

Although it cannot be seen explicitly from the script or visualization, there is a difference between the first two dimensions (A and B in the figure) and the last one (C).

---

The structure is considered to be periodic in the first two dimensions. In the present case, where the model represents a 1-d wire, this periodicity is an artifact of the calculation. In this case, the dimensions need to be large enough to avoid interaction between the periodic images of the wire.

The last dimension is the transport direction. In this dimension, the **electrode** is also repeated infinitely, but only in one direction- the left electrode is repeated to the left (negative direction) and the right electrode to the right (positive direction).


### Note

In the example script, , the left and right electrode are defined in exactly the same way. The script distinguishes between left and right electrode when the **electrodes** are combined with the **central region**. The left electrode is written first (left) and the right electrode is written last (right).

```
device_configuration = DeviceConfiguration(  
    central_region,  
    [left_electrode, right_electrode]  
)
```

The **electrodes** serve another purpose than simply representing a periodic structure. They also represent the size of a solution domain used by the transport calculation - a solution domain that must have a certain size. This is why the **electrodes** are represented by two lithium atoms instead of just one.

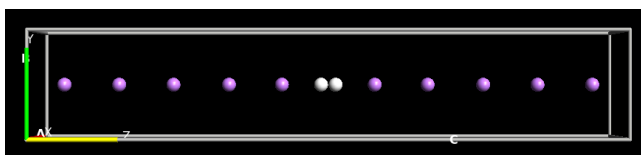
### Note

The size of the electrode cell in the c-direction is a convergence parameter and it must have an adequate size in this direction. For most calculations,  $> 7 \text{ \AA}$  is sufficient. The **electrodes** are large enough when they are larger than the maximum interaction range of the Hamiltonian. In a DFT calculation the maximum interaction range will be at least twice the maximum cut-off radius of the basis set used. The cut-off radius can be found using the **Basis Set Explorer**. The **Basis Set Explorer** can be accessed from the **Custom Analyzer**  via the [Analyzers](#) menu item.

## THE CENTRAL REGION

---

The **central region** is shown below



The script defining the **central region** has the same elements as the one defining the **electrodes**

```
# Central region  
  
# Set up lattice  
vector_a = [6.0, 0.0, 0.0]*Angstrom  
vector_b = [0.0, 6.0, 0.0]*Angstrom  
vector_c = [0.0, 0.0, 33.1418095157]*Angstrom  
central_region_lattice = UnitCell(vector_a, vector_b, vector_c)
```

```

# Define elements
central_region_elements = [Lithium, Lithium, Lithium, Lithium, Lithium, Hydrogen, Hydrogen,
                           Lithium, Lithium, Lithium, Lithium, Lithium]

# Define coordinates
central_region_coordinates = [[ 3.      , 3.      , 1.5571025 ],
                              [ 3.      , 3.      , 4.6713075 ],
                              [ 3.      , 3.      , 7.76171051],
                              [ 3.      , 3.      , 10.91847415],
                              [ 3.      , 3.      , 13.93149176],
                              [ 3.      , 3.      , 16.15851591],
                              [ 3.      , 3.      , 16.98337712],
                              [ 3.      , 3.      , 19.21030765],
                              [ 3.      , 3.      , 22.2234055 ],
                              [ 3.      , 3.      , 25.38002278],
                              [ 3.      , 3.      , 28.47050152],
                              [ 3.      , 3.      , 31.58470752]]*Angstrom

# Set up configuration
central_region = BulkConfiguration(
    bravais_lattice=central_region_lattice,
    elements=central_region_elements,
    cartesian_coordinates=central_region_coordinates
)

```

The **central region** represents what is in-between the two **electrodes**. In the transport dimension (C in the figure), the left side of the grey box represents where the left electrode starts. The right side of the grey box represents where the right electrode starts.

The atoms in the **central region** near the **electrodes** are used as boundary layers between the perfectly periodic **electrodes** and the **central region**. These boundary layers must be a perfect match of the **electrodes**.



## Note

The first atoms defined in the **central region** must obey a certain syntax. The atoms must exactly match the coordinates and elements of the left electrode. Likewise the last atoms in the **central region** must match the atoms in the right electrode. However for the last atoms, the match is slightly different in the C-direction. Here the length of the **central region** in the C-direction minus the C-coordinates must match the length of the **right electrode** minus the C-coordinates of the right electrode.

## SUMMARY

In ATK, a **device configuration** is a way of representing the atomic structure of two semi-infinite **electrodes** plus some structure in-between. In this chapter, you have learned how the **device configuration** is represented visually and how it is implemented in NanoLanguage. A **device configuration** can be set up interactively with VNL and subsequently optimized to find the equilibrium atomic geometry. You will learn how to do this in the [last chapter](#). Having looked at how to define the atomic structure, you are now ready to move on to define the type of calculation.


# CHAPTER 3. SETTING UP A TRANSPORT CALCULATION WITH THE SCRIPT GENERATOR

---

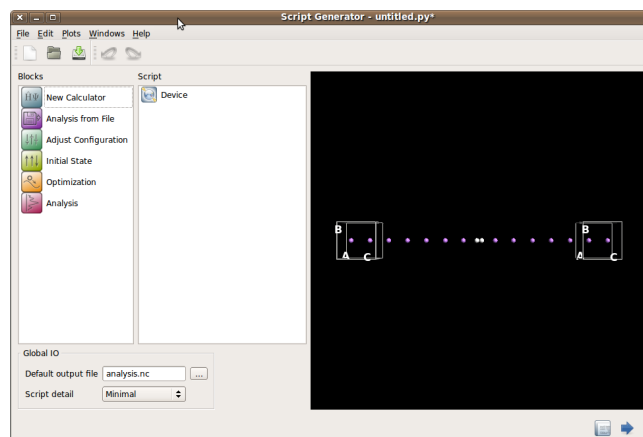
In this chapter you will learn how to perform a transport calculation once the **device configuration** has been defined.

## LOADING A DEVICE CONFIGURATION TO THE SCRIPT GENERATOR

---

In the file browser of the VNL main window, select the file constructed in the previous chapter and drag and drop the file to the **Script Generator** icon .



The **Script Generator** should now look like this



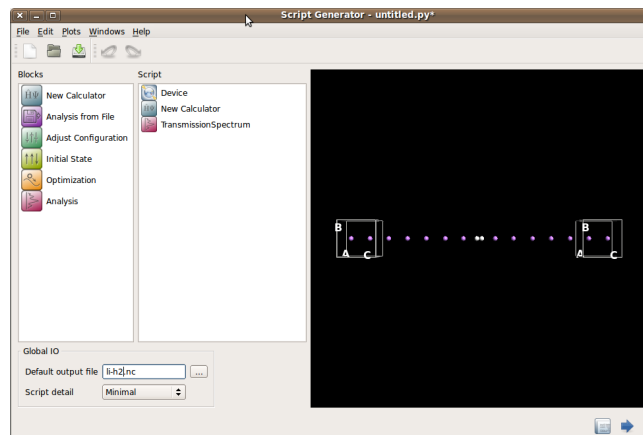
## SETTING UP THE CALCULATION

---

To set up the calculation of the transmission spectrum in the **Script Generator**:

1. **Double-click** the **New Calculator**-icon .
2. **Double-click** the **Analysis**-icon  and select TransmissionSpectrum from the menu.
3. Change the default file name to `li-h2.nc`.

The window should now have the following appearance

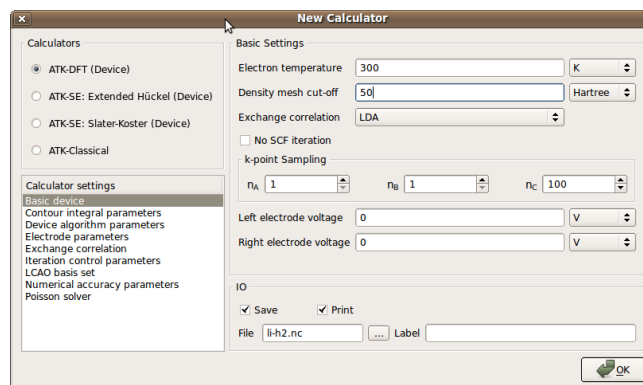


To modify the default parameters **double-click** the **New Calculator** icon that has appeared in the right panel. The **New Calculator** widget will now open.

To speed up the calculation change the following settings

- Set the **mesh cut off** to 50 Ry.

After the changes the **New Calculator** widget should look like this

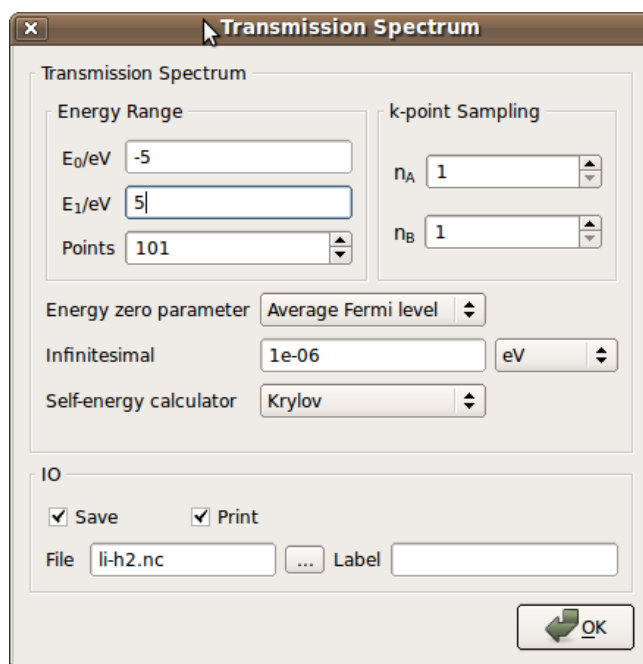


Press the **OK**-button to return to the **Script Generator**.

Now **double-click** the **TransmissionSpectrum** analysis object in the right panel.


- Adjust the range of the plot to **[-5, 5]**.

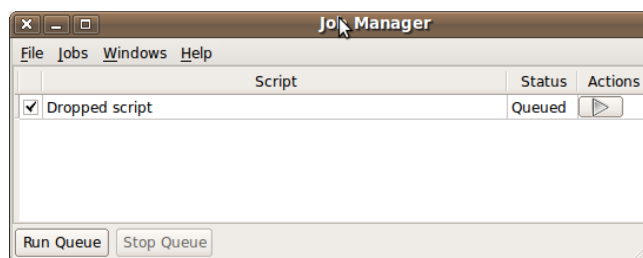
The menu should now have the following appearance



Again, press the **OK**-button to return to the **Script Generator**. The script is now ready to be run.

## RUNNING THE SCRIPT

Send the script to the **Job Manager** by pressing the “send-to”-button  and select **Job Manager** from the pop-up menu. The **Job Manager** should look like this



Now press the **Process Queue**-button to start the calculation. A window should appear with log messages. The job should finish within a few seconds.

## VISUALISING THE RESULTS

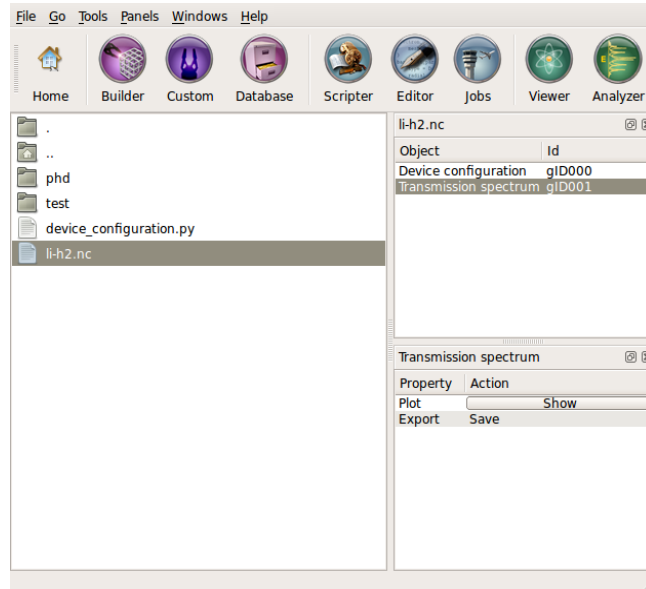
In order to visualize the results:

1. Locate the file `li-h2.nc` using the VNL main window.
2. Mark the file by **left**-clicking on it.
3. The objects that the file contains will appear in the upper right panel.
4. Mark the label **Transmission spectrum** by **left**-clicking on it.

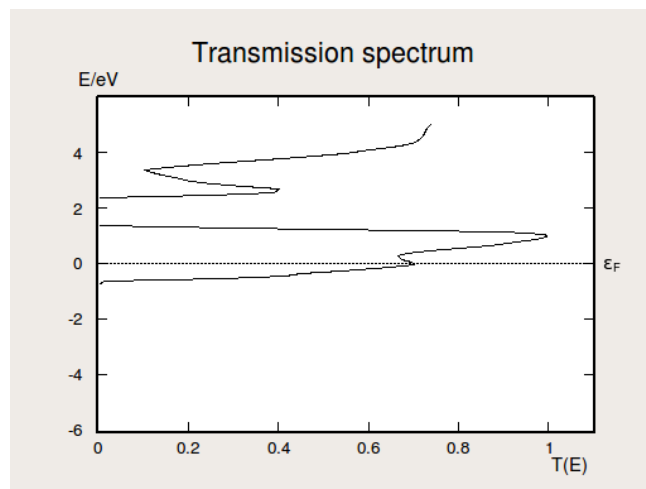
5. Possible actions that can be performed will appear in the lower right panel.

6. **Left**-click on the Plot **Show** button in the lower right panel.

After these operations the VNL main window should look like the following



The transmission plot should appear



By **right**-clicking on the **Transmission spectrum** window the plot can be saved by choosing Export image. The data contained in the plot can be exported to be used in third-party plotting software by choosing Export data.

Do not delete the file `li-h2.nc` since you will have to use it later in this tutorial.

## CHAPTER 4. FURTHER ANALYSIS OF THE RESULTS

---

In addition to the possibility of performing analysis in the same script as the self-consistent calculation, it is also possible in ATK to do the analysis later. This is convenient since the self-consistent calculation in most cases is far more time-consuming than the analysis calculation, but at the same you may not know, beforehand, all the quantities that you wish to compute.



For instance, you will first want to look at the transmission spectrum in order to find out at which energies the electron transfer is strongest, and then later go in and study in more detail e.g. the local density of states at this energy. In this case, there is no need to redo the self-consistent calculation.

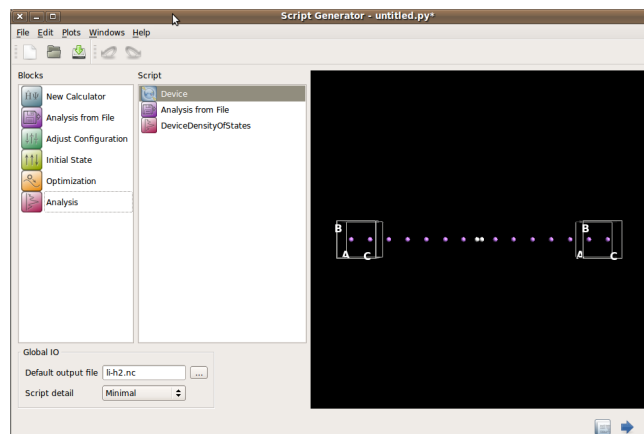
Moreover, the post-convergence analysis is often much less computationally demanding, and can usually be performed on a workstation or a laptop, while you may want to run the main calculation on a cluster or some dedicated computational server.

### DEVICE DENSITY OF STATES

---

Let us calculate the density of states of our device. Go back to the **Script Generator** and delete **New Calculator** and **TransmissionSpectrum** from the Script panel by marking them and pressing **Delete**.

Next double-click the **Analysis from File** block , and also double-click **Analysis**  and choose **DeviceDensityOfStates** from the menu.



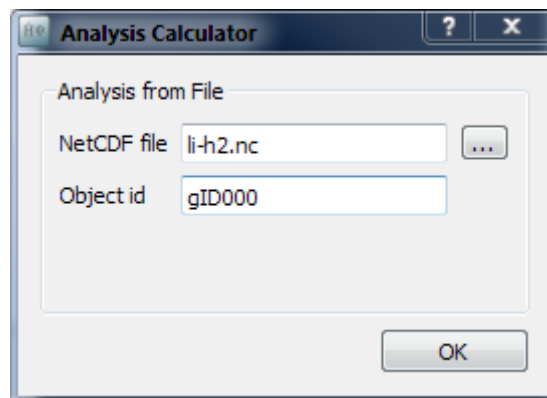
**i Tip**

Instead of modifying the open Script Generator you may open a new Script Generator. In this case the configuration is not shown, but this does not matter for the calculation. (Remember, in this case, to set the filename `li-h2.nc` where the results will be saved.)

The **Analysis from File** block is used to restore the state of a previous calculation, so that you can perform the analysis. Double-click this script block and specify the **NetCDF file** which contains our converged calculation. Click the button "... " to open a file browser, and locate the file `li-h2.nc`.

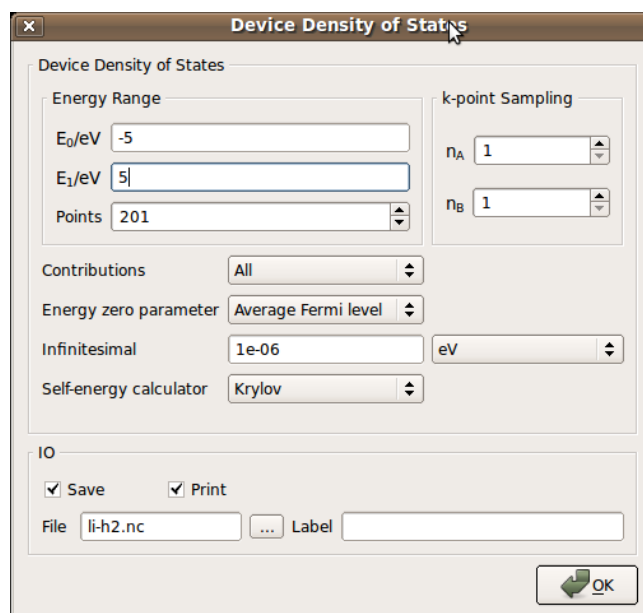
**i Tip**

If the file contains several configurations, you can specify which one to restore by providing its **ID**, which is a unique identifier for each object in the NetCDF file. The first configuration almost always has **Id** "gID000", but in more general cases (e.g. after running a geometry optimization) it is recommended to inspect the IDs in the Result Browser first.



Press **Ok** to return to the Script Generator.

Now double-click the **DeviceDensityOfStates** object in the Script panel. Set the range to -5, 5, and change the number of points to 201 for better resolution.

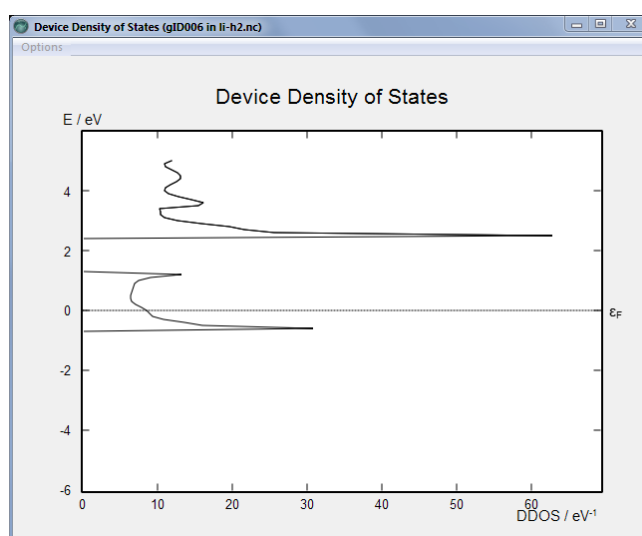


**i** **Tip**

In this example you will save the results in the same file as the self-consistent calculation itself, but it is also possible to store the results in a different file. This can be convenient especially for large data sets like 3D grids etc, in order not to bloat the NetCDF file from which you perform the analysis, and for data which you may not be interested in keeping permanently.

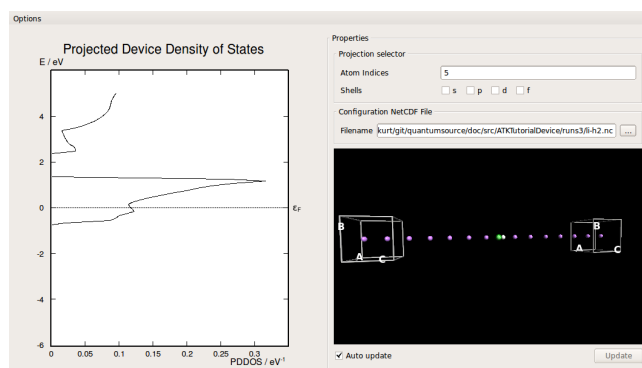
Send the calculation to the **Job Manager** and start the job in the same way as with the calculation of the transmission spectrum. It should only take seconds to run.

To plot the device density of states, select the NetCDF file in the VNL main window and choose **Device density of states** in the Result Browser, and then click the **Show** button next to **DDOS** in the Result Viewer.



The device density of states object also contains information on the partial density of states (PDDOS) which can be analyzed by instead clicking the **Show** button next to **PDDOS** in the Result Viewer.

In the window that opens up, the total DDOS is shown, but you can project the device density of states on e.g. one of the hydrogen atoms by selecting the atom in the 3D window.



Note how the partial density of states for hydrogen has the same general shape as the transmission spectrum. This indicates that the hydrogen molecule is the bottle-neck for transport in this system.

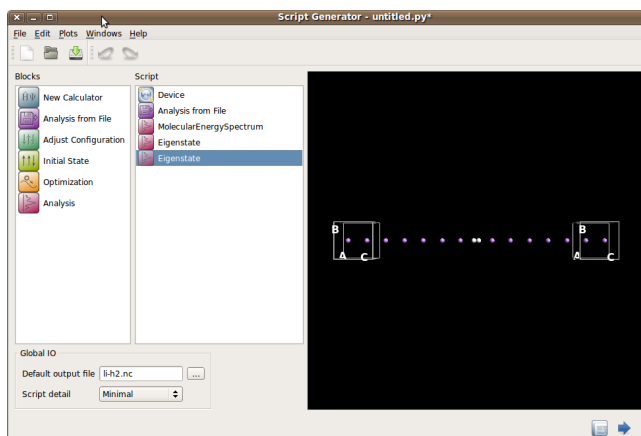
**Tip**

The PDDOS of several atoms can be displayed by selecting the atoms using Ctrl-left-click. By ticking the s,p,d,f boxes, the PDDOS will be projected onto the angular momenta of the selected atoms.

## MOLECULAR PROJECTED SELF-CONSISTENT HAMILTONIAN (MPSH)

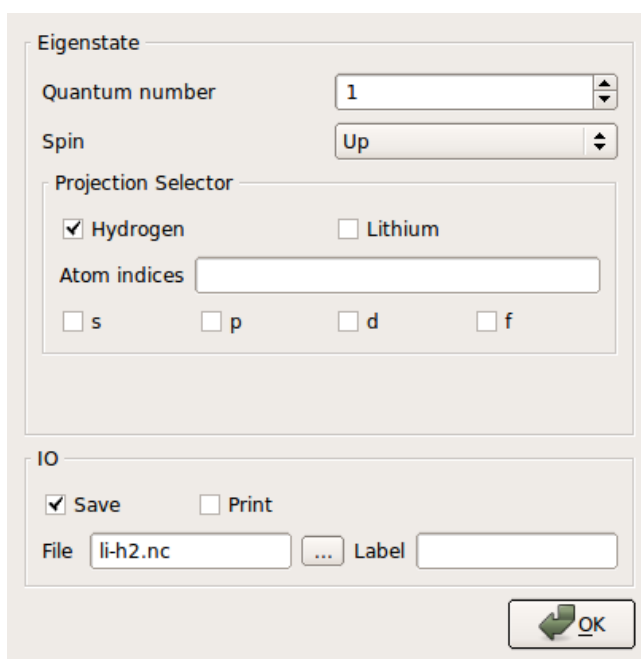
In this section you will learn how to calculate the eigenstates of the hydrogen molecule inside the lithium chain, the so-called Molecular Projected Self-Consistent Hamiltonian (MPSH) spectrum.

Return to the Script Generator window, delete the **DeviceDensityOfStates** object and instead insert a **MolecularEnergySpectrum** and two **Eigenstate** analysis objects.



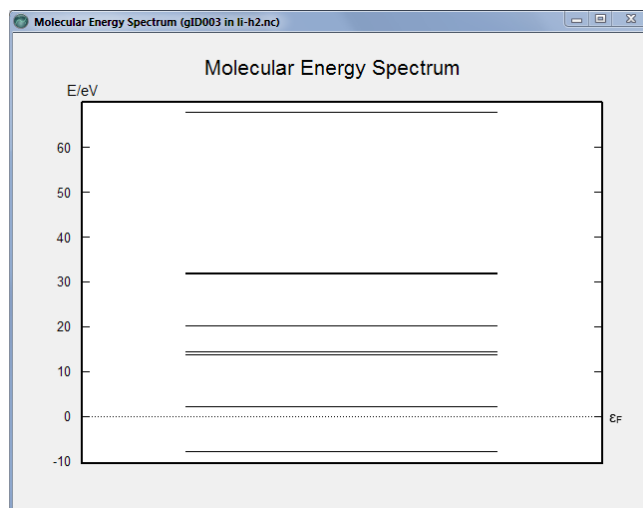
Open the **MolecularEnergySpectrum** and tick **Hydrogen** in the **Projection Selector**.

Open each of the **Eigenstate** widgets, select projection onto the hydrogen atoms, and select quantum number 0 for the first eigenstate and number 1 for the second eigenstate.



Execute the script by sending it to the Job Manager. Three new objects will now be added to the file `li-h2.nc`.

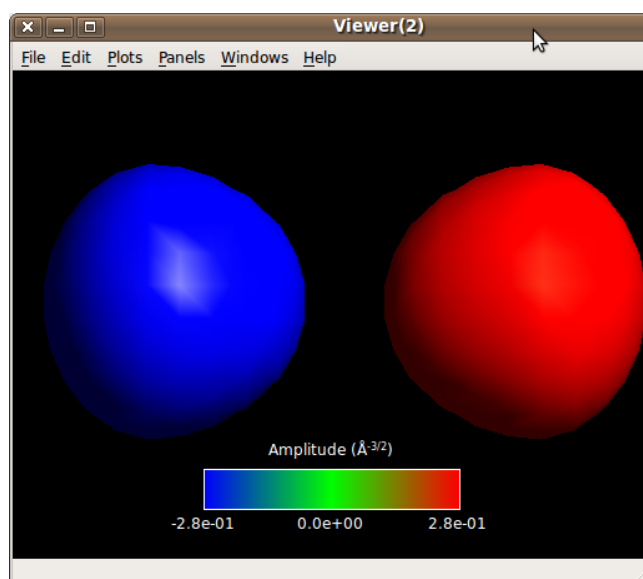
The data objects can be inspected from the Result Viewer in the VNL main window. To view the MPSH spectrum, select the **Molecular energy spectrum** object in the `li-h2.nc` file and click the Show button next to Plot in the Result Viewer.



There is one eigenstate below the Fermi level; this state corresponds to the HOMO state in the isolated hydrogen molecule, and naturally the next higher state is the LUMO (it is indeed above the Fermi level). If we compare the HOMO-LUMO gap of the  $H_2$  molecule when placed in the lithium chain to that of the isolated gas-phase molecule, as computed by ATK, we find that it is reduced from 12.1 eV to about 10.1 eV. More specifically, in the gas-phase, the LUMO lies 6 eV above the Fermi level, while when embedded in the lithium chain, this state is only 2 eV above, which explains the relatively large transmission of this system.

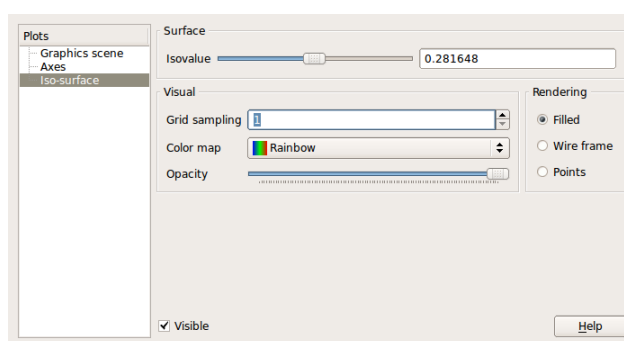
**Tip**

To view the values of the molecular levels, either look in the log produced while running the calculation, or right-click the plot and choose Export data from the popup menu, and open the exported file in e.g. the Editor in VNL.



To obtain an isosurface of eigenstate 1,

1. Select the second **Eigenstate** object in the `li-h2.nc` file.
2. Click the **Show** button next to the **Isosurface** property.
3. Right-click the Viewer window and choose **Properties** from the popup menu.
4. Set the grid sampling to 1 for the isosurface.

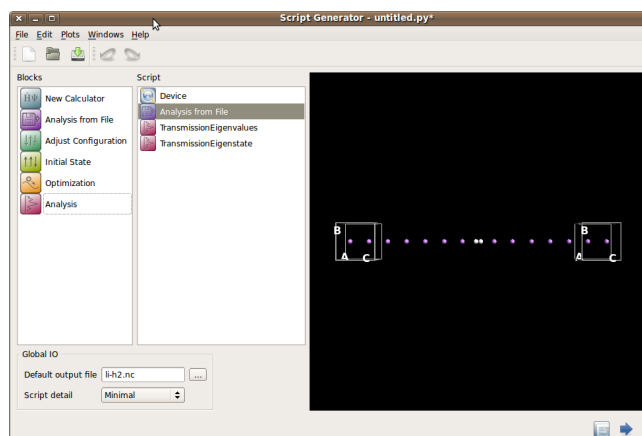


This state clearly resembles the LUMO state of the isolated  $H_2$  molecule.

## TRANSMISSION EIGENCHANNELS

Another method by which we gain more insight into which orbitals that are responsible for the transport, is to perform a transmission eigenchannel analysis. In this case the transmission matrix is diagonalized to find the electron paths which contribute most to the transmission coefficient for a particular energy and **k**-point.

Return to the Script Generator window, delete the **MolecularEnergySpectrum** and the two **Eigenstate** objects. Instead, insert a **TransmissionEigenvalues** block and a **TransmissionEigenstate** block (from **Analysis**).



We will stay with the default parameters, which means to perform the transmission eigenchannel analysis at the Fermi energy, at the  $\Gamma$  point.

Execute the script by sending it to the Job Manager, and two new objects will be added to the `li-h2.nc` data file.

The transmission eigenvalues are printed to the log window; if you closed it, it can be opened again from the Job Manager by clicking the **Show** button in the Log column. Alternatively, you can

view the eigenvalues by selecting the Transmission eigenvalues object in `Li-h2.nc` in the Result Browser, and then click the **Print>Show** button in the Result Viewer.

We see that the transmission at the Fermi energy is due to a single channel with transmission 0.7.

```
Transmission Eigenvalues Report
-----
Left electrode Fermi level = -2.826189e+00
Right electrode Fermi level = -2.826189e+00
Energy zero = -2.826189e+00
Energy = 0.000000e+00
Unit = eV
-----
Number of transmission modes = 1
7.039164e-01
-----
```

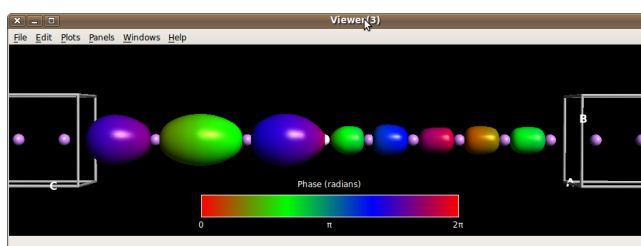
The transmission eigenchannel can be visualized as an isosurface, by selecting the Transmission Eigenstate object in `Li-h2.nc` in the Result Browser, and clicking **Isosurface>Show** in the Result Viewer. The transmission eigenstate is a 3D complex function, and the isosurface is given by the absolute value of the complex number, while the color indicates the phase.

To visualize the Device Configuration on top of the Transmission Eigenstate drag and drop the Device Configuration object in `Li-h2.nc` in the Result Browser onto the viewer window.

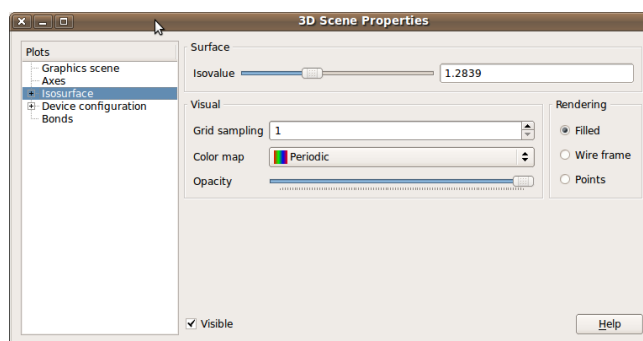
### Note

We knew beforehand that there would only be one eigenchannel. Normally, you would first compute the eigenvalues to find out the number of channels, and then compute the corresponding number of eigenstates.

Note how the transmission eigenstate has lower amplitude on the right, due to scattering by the hydrogen molecule. The wave function has the same wavelength on both sides (it takes 4 atom units for the color to repeat). However, there is a 180 degrees phase shift across the hydrogen molecule, showing that the electron propagates through the anti-bonding LUMO state of the hydrogen molecule.



The above picture was obtained by making the following settings in the isosurface property widget.

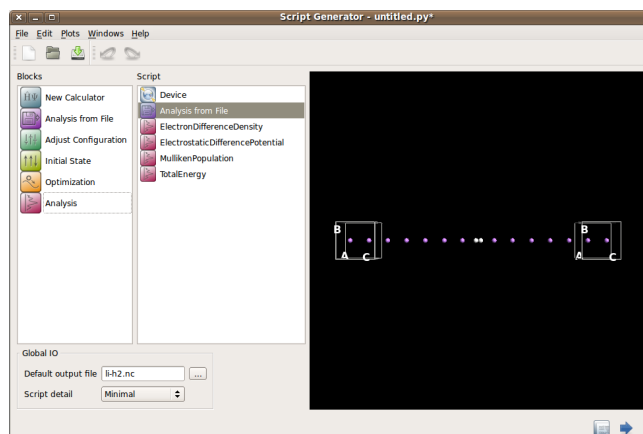


## SEVERAL TYPES OF ANALYSIS IN ONE SCRIPT

If you would like to perform several types of analysis, there is no reason to split them into several jobs. In the following you will do a large number of analysis in the same script.

First delete the previous analysis objects in the script panel of the **Script Generator** (but keep the **Analysis from File** block). Then add the following analysis objects, one after the other:

1. **ElectronDifferenceDensity**
2. **ElectrostaticDifferencePotential**
3. **MullikenPopulation**
4. **TotalEnergy**



Send the job to the Job Manager and start it. Once the calculation has finished all the information is available in the file `li-h2.nc`.

## CHAPTER 5. I-V CHARACTERISTIC


---

In this chapter you will learn how to calculate the I-V characteristic of a device - the current as a function of bias.

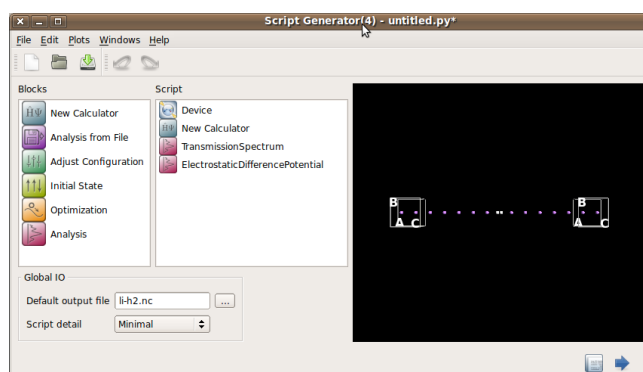
### SETTING UP THE CALCULATION

---

VNL has a **Custom Analyzer** that takes an NetCDF-file containing transmission spectra taken at different bias and calculates the I-V characteristic. You will use this tool.

As the first thing you must set up the calculation that generates the transmission spectra. In the VNL main window, drag the file `li-h2.nc` onto the **Script Generator** icon  and change the output file name to `li-h2.nc`.

Add a **New Calculator** and a **TransmissionSpectrum** analysis item to the right panel. Also add a **ElectrostaticDifferencePotential** analysis item, since you will need this result later in the tutorial.



Now **double-click** the **New Calculator** item to open its menu.

- Set the **Right electrode voltage** to 1 V.

Return to the **Script Generator**.


Also **double-click** the **TransmissionSpectrum** item. Change the range to `[-5, 5]` to match the already calculated transmission spectrum.

Next send the job to the **Job Manager** and start it. The script should save a new **Transmission spectrum** to the file `li-h2.nc`.

---

## CALCULATING THE I-V CURVE

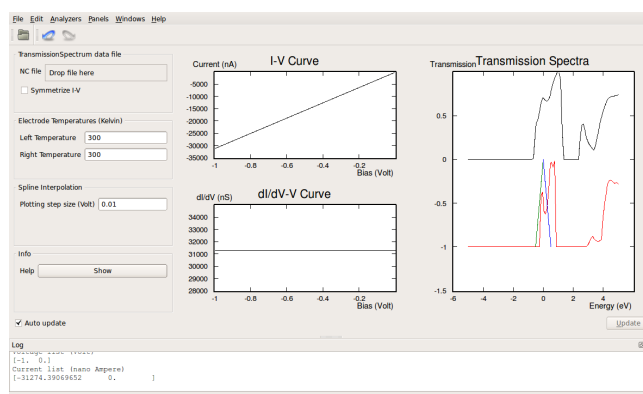
---

Launch the **Custom Analyzer** tool by left-clicking the **Analyzer** icon  on the Toolbar in the main VNL window.

The **Custom Analyzer** tool comes with a few build-in analysis tools. Besides these tools, it is possible to write scripts that enables the tool to perform special analysis functions.

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

Next drag and drop the file `li-h2.nc` onto the NC file drop zone and you should see the following plot



The left most plots show the I-V and dI/dV-V curves. The curves were obtained by importing each Transmission Spectrum in the nc file and integrating the transmission coefficient over the bias windows. 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 embrace the part of the Transmission spectra which are inside the bias window.

---

## PERFORMING ADDITIONAL ANALYSIS USING SCRIPTING

---

It is possible to determine the current from a previously calculated transmission spectrum analysis using a python script (In fact this is how the I-V Curve Custom Analyzer works). An example script `current.py` is given below

```
<xi:include></xi:include>
```

Dropping this script on the **Job Manager** will produce the following output

```

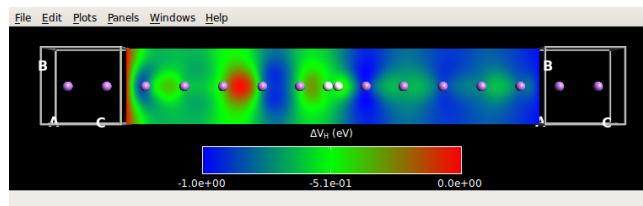
File Windows Help
-----
NanoLanguageScript execution started
-----
Current Report
-----
Applied bias      = -1.000 Volt
Bias window      = [ 0.000 Volt, 1.000 Volt ]
Bias window covered = 100.00 %
Calculated current = -3.12743906965e-05 A
Spin             = All available
-----
current = -3.12743906965e-05 A
NanoLanguageScript finished successfully
-----
NanoLanguageScript execution finished

```

Another example of analysis carried out via scripting is that of calculating the voltage drop across a device. Assuming that you have performed the earlier calculations and that the NetCDF file `li-h2.nc` is available, you can carry out the analysis by dropping the following script named `voltage_drop.py` on the **Job Manager**.

```
<xi:include></xi:include>
```

The result of this script is to save a new electrostatic difference potential object into the `li-h2.nc` file. A convenient ID tag named **voltagedrop** is attached to the object for easy identification when browsing the NetCDF file's contents. This new potential visualizes the potential drop across the device. From the file `li-h2.nc`, open the **Electrostatic difference potential** marked 'voltagedrop'. Then drag the **Device configuration** onto the open Viewer window. The resulting image should look like this



The potential in the right part of the device is -1 eV, corresponding to the applied right potential of  $V_R = 1$  Volt, which gives rise to an electro-static potential of  $-eV_R = -1$  eV.

 **Note**

The relative high current running through the device at 1 Volt, has the effect that the atoms in the central region never reach an equilibrium electron distribution. Part of the assumption in the computational model is that atoms close to the electrodes have an equilibrium distribution, thus, results from 1-d systems with a high current must be treated and analyzed with care. In the present case it has the effect that the potential in the left part of the device never reach 0 eV.

## PERFORMING AN I-V SCAN WITH SCRIPTING

To perform calculations at several voltages it is convenient to use scripting. Below is illustrated a script which performs self-consistent calculations at 0.25, 0.5 and 0.75 Volt.

```

#read in the old configuration
device_configuration = nload("li-h2.nc",DeviceConfiguration)[0]
calculator = device_configuration.calculator()

# Define bias voltages
voltage_list=[0.25, 0.5, 0.75]*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*Volt, voltage)),
    initial_state=device_configuration)

# Calculate the transmission spectrum
transmission_spectrum = TransmissionSpectrum(
    configuration=device_configuration,
    energies=numpy.linspace(-5,5,100)*eV,
    )

nlsave('li-h2.nc', transmission_spectrum)
```

# CHAPTER 6. BUILDING AND OPTIMIZING THE GEOMETRY

---

In the previous part of the tutorial ignored an important part of a solid-state calculation - calculating the position of the atoms. In this chapter you will learn how to set up an initial guess for the structure using VNL and subsequently finding the structure where none of the atoms experience any forces or stress - the optimized geometry. Potentially, the optimized geometry can have completely different properties than the initial guess, thus the optimization step is always important.


## SETTING UP THE ELECTRODE GEOMETRY



---

The first task is to determine the electrode geometry. To find an initial guess for the geometry you will use the VNL **Builder**. In the second step we use ATK-DFT to optimize the geometry.

## BUILDING AN INITIAL GUESS FOR THE ELECTRODE GEOMETRY

---

You will now construct a one-dimensional lithium wire with the **Builder**. First launch the **Builder** tool by **left**-clicking the icon  on the Toolbar in the main VNL window.

The **Builder** is started in molecule mode . In the left panel switch to bulk mode by **left**-clicking the bulk icon .

The initial configuration consists of a single hydrogen atom. Transform the hydrogen atom into a lithium atom by **double**-clicking the hydrogen **cell** in the **Basis** table and then selecting lithium from the list of elements.

The next step is to change the dimensions of the unit cell to define a one-dimensional chain. To do this, first select **Unit cell** from the lattice list under the **Type** combo-box found in the **Lattice Parameters** section of the **Lattice** panel. This will enable you to customize the unit cell by modifying the primitive vectors of the unit cell (A,B and C) directly in the corresponding table.

Modify the unit cell vectors such that the unit cell vectors A, B, and C point in the directions (6,0,0), (0,6,0), and (0,0,2.2) respectively. The 2.2 Ångstrom is an initial guess of the inter-atomic distance between successive lithium atoms in the **electrode** and this distance will be optimized with ATK-DFT. The large size in the other two directions is to ensure reasonable separation between the one dimensional wires in the repeated structure.

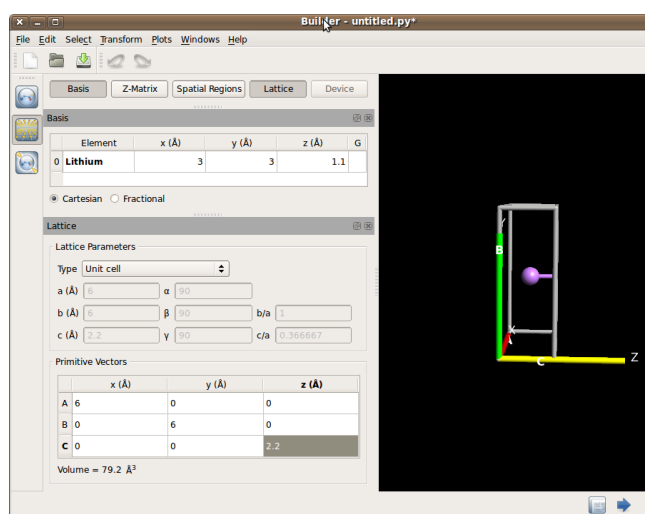
## Note

To remove all interactions the box should be larger than the basis set interaction range which is 8 Ångstrom, however, for improved performance it is made a little smaller.


Finally, move the lithium atom from the origin of the system into the center of the newly modified unit cell, by using the menu command **Transform** → **Center**, which is available from the **Builder** menu bar.

Thus, you have setup a lithium chain with an inter-atomic distance of 2.2 Ångstrom. Since the cell is also periodic in the x and y directions, you have an infinite array of chains, and the chains are separated by a distance of 6 Ångstrom in the x and y directions.

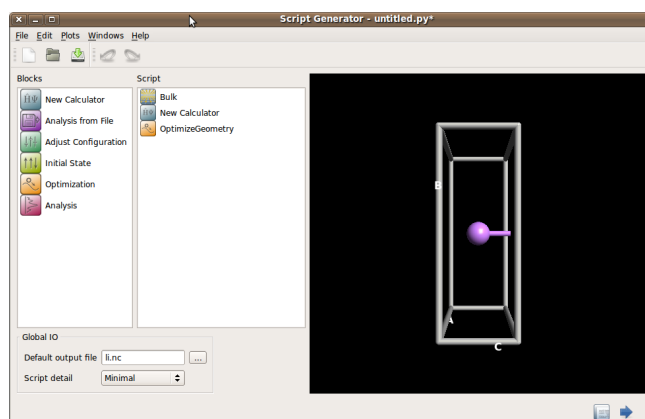
The window should now look like the following



## OPTIMIZING THE ELECTRODE GEOMETRY

You will now use ATK-DFT to optimized the electrode geometry. Press the send to button  in the lower right corner of the bulk builder and select **Script Generator**.

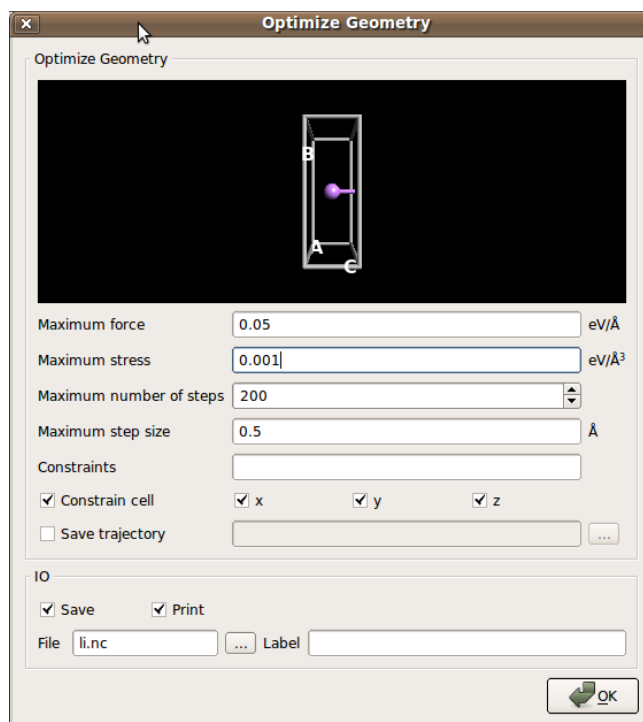
In the script generator add a **New Calculator** and a **Optimize Geometry** object to the right panel. Next change the output file name to `Li.nc`.



**Double-click** the **New Calculator** item in the right panel and

- Set the **mesh cut off** to 50 Ry.
  - Change the the k-point sampling in the C direction to  $nc=50$ .
- Double-click** the **Optimize Geometry** item in the right panel and
- remove the tick from constraining the cell in the z-direction.
  - Set the stress tolerance to  $0.001 \text{ eV/\AA}^3$ .


The widget should have the following settings

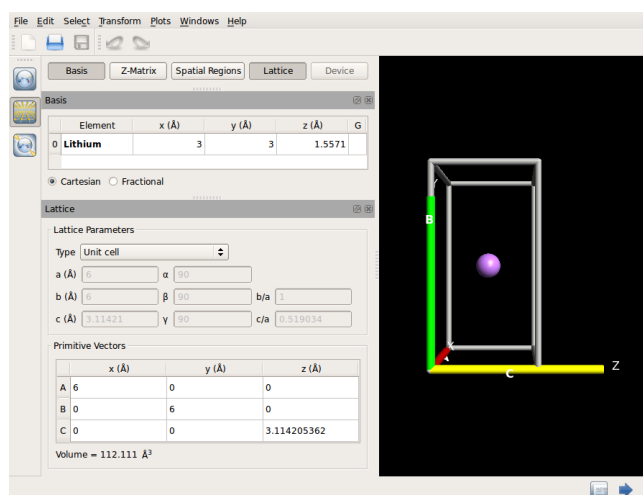


Press the **Ok**-button to return to the **Script Generator**.

Now execute the job - this should only take a few minutes.

When the optimization has finished **left-click** the `li.nc` in the result browser. The file has 2 bulk configurations.

Drag and drop the second bulk configuration to the builder  and you should see the following



Note that the distance between the lithium atoms in the chain has increased from 2.2 Å to 3.11 Å

## DETERMINING THE GEOMETRY OF THE CENTRAL REGION

The next step is to find the geometry of the central region. The steps are the same as for the electrode, i.e. first you will setup an initial guess for the geometry using the **Builder** and in the second step use ATK-DFT to optimize the geometry.

### BUILDING AN INITIAL GUESS FOR THE CENTRAL REGION GEOMETRY

You will now modify the ideal lithium chain into a wire of lithium atoms interrupted by a hydrogen molecule. This configuration will represent the central region of the device.

Start by selecting **Transform** → **Repetition** from the **Bulk Builder** menu bar. When the Bulk Repetition dialog appears, select a repetition index of 1, 1, 12 for the direction vectors A, B, and C, respectively. This will effectively repeat the system into a wire with 12 lithium atoms in a row:

#### **i** Tip

You can reset the configuration view in the **Builder** by pressing **Ctrl+R** while using the **Builder**. This will bring the entire configuration into easy view.

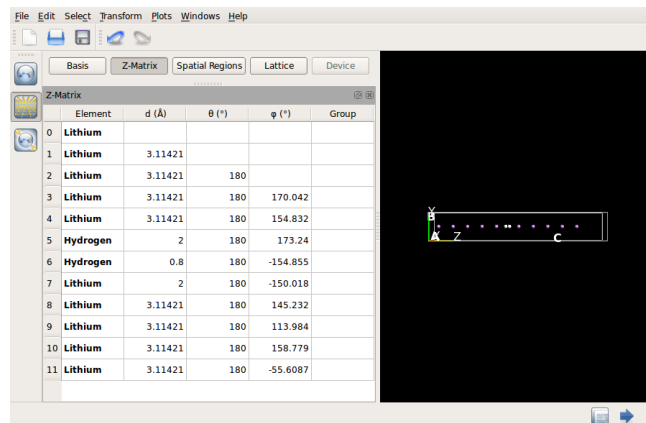
In the **Basis** table, scroll down such that atoms number 5 and 6 are visible. Change these to hydrogen atoms, as done earlier when converting the hydrogen atom to a lithium atom.


Next step is to manipulate the z-matrix of the wire atoms. **left-click** the **Lattice**-button and the **Basis**-button to remove these widgets, and **left-click** the **Z-Matrix** button to be able to change the z-matrix of selected atoms.

Next select all atoms using **Select** → **All**

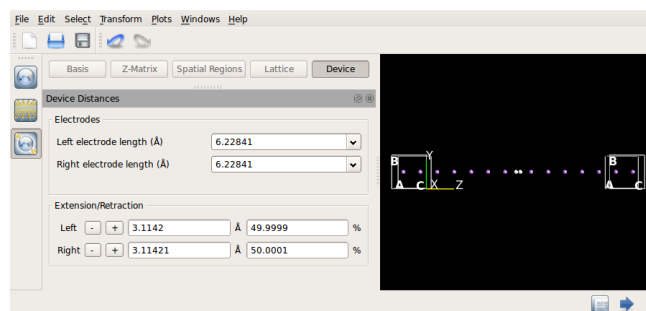
- Change bond distance of atom 5 to 2 Å
- Change bond distance of atom 6 to 0.8 Å
- Change bond distance of atom 7 to 2 Å

The builder should now have the following settings



In the left panel switch to device mode by **left**-clicking the device icon  .


The builder automatically detects the repetitions in the central region and sets up the device configuration.



## OPTIMIZING THE CENTRAL REGION GEOMETRY

The optimization of the central region geometry can be very time consuming. You will in this section learn two different approaches, an approximate method treating the device system as a bulk system, and a full open boundary condition optimization.

### OPTIMIZING THE CENTRAL REGION GEOMETRY AS EQUIVALENT BULK

In this section you will learn how to optimize the central region geometry as a bulk system. First send the device geometry to the **Script Generator** using the send to button  .

In the script generator

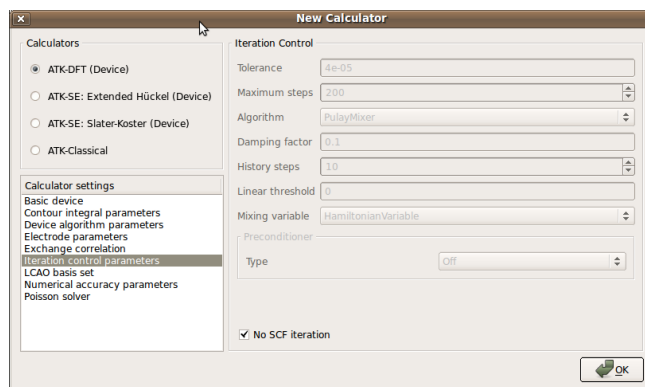
- Add a **New Calculator** object.
- Add an **Optimize Geometry** object.
- Change the output file name to `li-h2-opt.nc`.

Open the **New Calculator** object and

- Set the **mesh cut off** to 50 Ry.

- Select the **iteration control parameters** widget and tick the **No SCF iteration** box.

For a Device Configuration No SCF iteration, means no NEGF open boundary SCF loop. However, there still is an equivalent bulk SCF loop in order to make an initial guess for the electronic structure, and it is this SCF state that is used for the optimization.

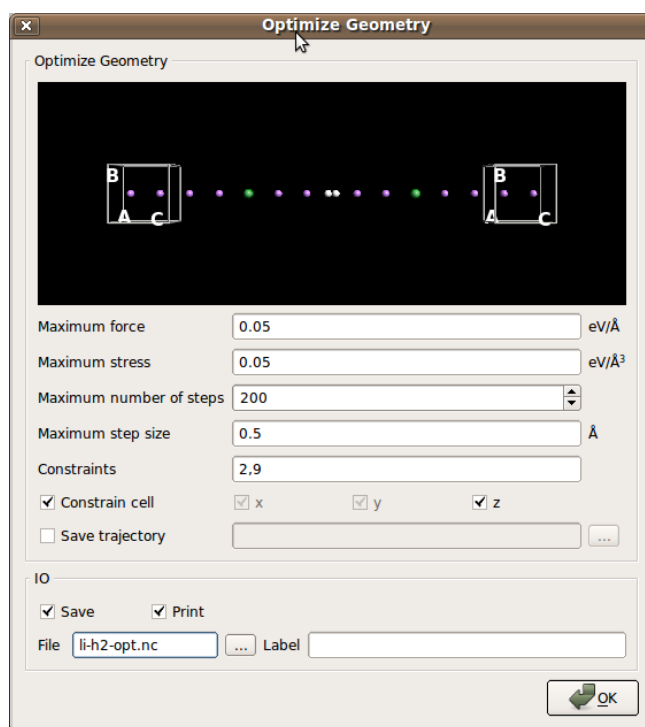


The equivalent electrode atoms in the central region are automatically fixed during a device optimization. In the following you will fix a few additional atoms, since the non self-consistent method is approximate and may give wrong forces for atoms close to the electrodes.

Open the **Optimization** object. Select atom 2 and 9 by

- **left**-click on the third atom from the left in the central region.
- Hold down the Ctrl key and **left**-click on the third atom from the right in the central region.

The widget should now have the following settings (check that you have constrained atom 2,9).




## Note

The cell vectors are not optimized, since usually the non self-consistent method is not sufficient accurate for such optimizations.

Next execute the job - this will take some time, typically 5-10 minutes.

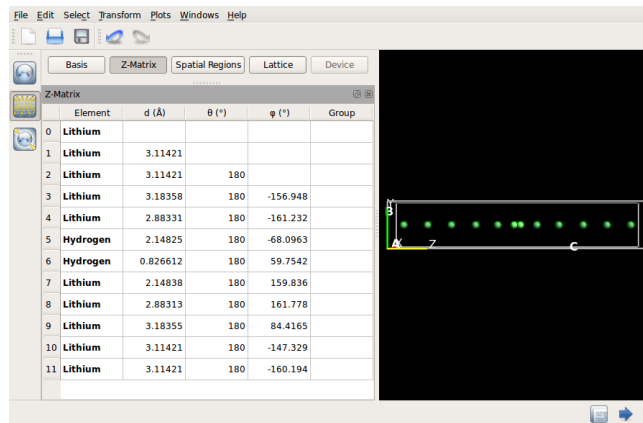
When the optimization has finished **left**-click the `li-h2-opt.nc` in the result browser. The file has 2 device configurations.

Drag and drop the second device configuration to the builder , switch to bulk mode



- Activate the Z-Matrix widget, and de-activate the Basis and Lattice widgets.
- Select all atoms.



You should now see the following



Element	d (Å)	θ (°)	φ (°)	Group
0 Lithium				
1 Lithium	3.11421			
2 Lithium	3.11421	180		
3 Lithium	3.18358	180	-156.948	
4 Lithium	2.88331	180	-161.232	
5 Hydrogen	2.14825	180	-68.0963	
6 Hydrogen	0.826612	180	59.7542	
7 Lithium	2.14838	180	159.836	
8 Lithium	2.88313	180	161.778	
9 Lithium	3.18355	180	84.4165	
10 Lithium	3.11421	180	-147.329	
11 Lithium	3.11421	180	-160.194	

## OPTIMIZING THE CENTRAL REGION GEOMETRY AS A DEVICE

In this section you will learn how to perform a full device configuration optimization of the central region. As initial geometry, you will use the configuration from the previous section which was optimized using equivalent bulk.

In the **Builder** switch back to device mode  and send the optimized device geometry to the **Script Generator** using the send to button .

In the script generator

- Add a **New Calculator** object.
- Add an **Optimize Geometry** object.
- Change the output file name to `li-h2-opt.nc`.

Open the **New Calculator** object and

- Set the **mesh cut off** to 50 Ry.

- Remove the tick from the **IO save** check box, to avoid saving the non-optimized configuration.

The optimize geometry will optimize the position of all non-equivalent electrode atoms in the central cell. In order to optimize the cell size in the z-direction

Open the **Optimization** object and



- Remove the tick from constraining the cell in the z-direction.
- Set the force tolerance to  $0.01 \text{ eV/\text{Å}^3}$ .
- Set the stress tolerance to  $0.001 \text{ eV/\text{Å}^3}$ .

### **Note**

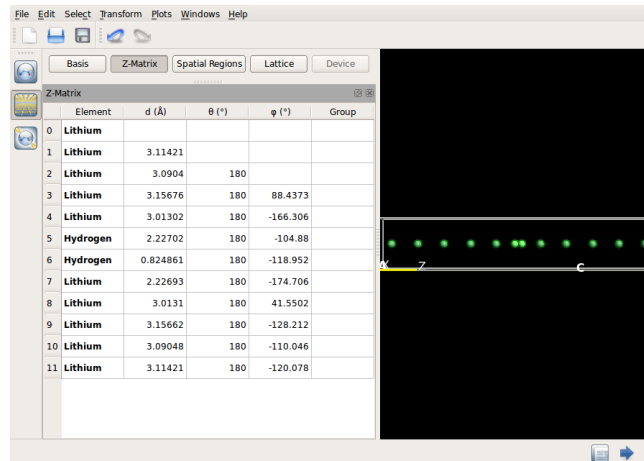
The cell size in the x and y directions cannot be optimized, since they are fixed by the electrode cell size.

Now execute the job - this will take approximately four times the previous optimization.

When the optimization has finished **left**-click the `li-h2-opt.nc` in the result browser. The file should now have 3 device configurations.

Drag and drop the third device configuration to the builder  , switch to bulk mode  .

Select all atoms and only activate the z-matrix widget and you should see the following



Element	d (Å)	θ (°)	φ (°)	Group
0 Lithium				
1 Lithium	3.11421			
2 Lithium	3.0904	180		
3 Lithium	3.15676	180	88.4373	
4 Lithium	3.01302	180	-166.306	
5 Hydrogen	2.22702	180	-104.88	
6 Hydrogen	0.824861	180	-118.952	
7 Lithium	2.22693	180	-174.706	
8 Lithium	3.0131	180	41.5502	
9 Lithium	3.15662	180	-128.212	
10 Lithium	3.09048	180	-110.046	
11 Lithium	3.11421	180	-120.078	

## OPTIMIZATION AT FINITE BIAS

Applying a finite bias will cause the position of atoms to shift slightly, so the geometry should in principle also be optimized under bias. However, the optimization under bias can be very time consuming and usually only has little effect on the transmission and is thus often omitted.

You will now use the zero bias optimized geometry as initial guess for a finite bias geometry optimization. Drag the last device configuration in the file `li-h2-opt.nc` onto the **Script Gen-**

**erator** icon 

This will open the script generator with a **New Calculator** with the settings of the calculator on the dropped configuration. Add an **Optimization** object and change the **Default output file** to `li-h2-opt-1V.nc`.

Open the **New Calculator** object and



- Set the **Right electrode voltage** to 1 Volt.
- Remove the tick from the **IO save** check box, to avoid saving the non-optimized configuration.

Open the **Optimization** object and make the following changes (the same as previously for the zero bias optimization)

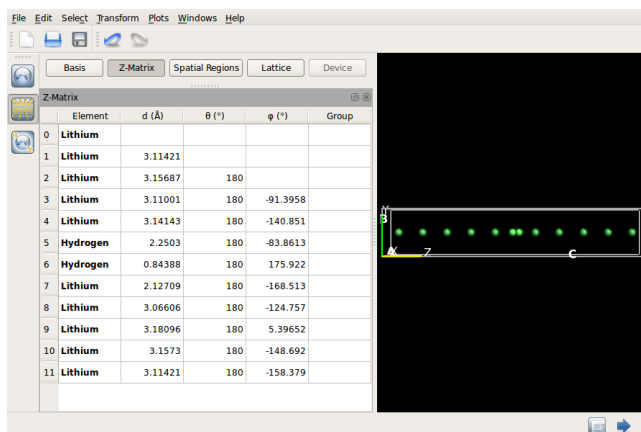
- Remove the tick from constraining the cell in the z-direction.
- Set the force tolerance to  $0.01 \text{ eV/\AA}^3$ .
- Set the stress tolerance to  $0.001 \text{ eV/\AA}^3$ .

Now execute the script by sending it to the **Job Manager**, it will take approximately double as long time as the zero bias calculation.

Finally, compare the difference in geometry of the optimization with and without bias. **Left-click** the `li-h2-opt-1V.nc` in the result browser. The file should only have the optimized device configuration.

Drag and drop the device configuration to the builder , switch to bulk mode .


Select all atoms and only activate the z-matrix widget and you should see the following

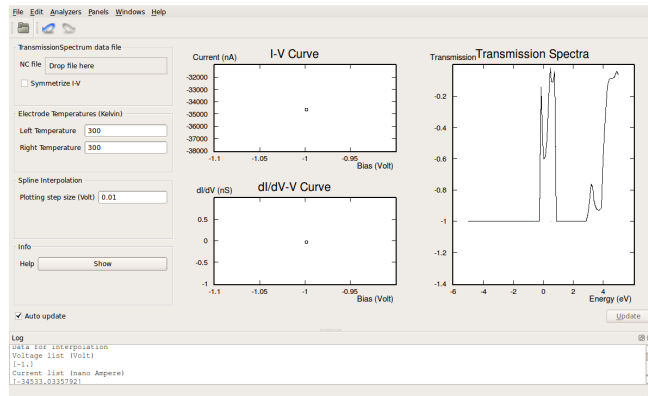


	Element	d (Å)	$\theta$ (°)	$\phi$ (°)	Group
0	Lithium				
1	Lithium	3.11421			
2	Lithium	3.15687	180		
3	Lithium	3.11001	180	-91.3958	
4	Lithium	3.14143	180	-140.851	
5	Hydrogen	2.2503	180	-83.8613	
6	Hydrogen	0.84388	180	175.922	
7	Lithium	2.12709	180	-168.513	
8	Lithium	3.06606	180	-124.757	
9	Lithium	3.18096	180	5.39652	
10	Lithium	3.1573	180	-148.692	
11	Lithium	3.11421	180	-158.379	

The most striking feature of the optimized geometry is that, while the the zero-bias geometry is fully symmetric, a slight asymmetry is introduced at finite bias.

Using analysis from file you may also calculate the transmission spectrum and save it into the file `li-h2-opt-1V.nc`.

When done open the Custom Analyzer  and select the I-V Curve tool from the Analyzers menu. Drop the file `li-h2-opt-1V.nc` onto the NC file drop zone, and the current is now displayed in the Log window.



Comparing with the non-optimized geometry we find that the optimization has increased the current by 10 percent.

# INDEX

---

## A

ATK, 1

Atomistix ToolKit (ATK), 1

## B

Builder, 23, 26

## C

calculation engine, 1

central region, 2

Custom Analyzer, 20

## D

Density Functional Theory, 1

Device Configuration, 2

device density of states, 11

## E

electrodes, 2

Extended-Hückel theory, 1

## J

Job Manager, 9

## N

Non-Equilibrium Green's Functions, 1

## P

plotting, 9

## S

Script Generator, 7

## T

transmission spectrum, 7, 19

## V

voltage drop, 20, 30