QuantumWise

ATK Tutorial for Device Configurations

Set up, calculate and analyze a simple device configuration

Version 2015.2

ATK Tutorial for Device Configurations: Set up, calculate and analyze a simple device configuration

Version 2015.2

Table of Contents

Chapter 1. Introduction

This tutorial introduces you to one of the most important functionalities in Atomistix ToolKit (ATK) - transport calculations. The system you will investigate is a hydrogen molecule placed between two one-dimensional lithium wires. This system is selected for demonstration purposes, in order to make calculations very fast. However, it does encapsulate many of the relevant and interesting physical effects present in more realistic systems, and these will be discussed in the tutorial. For more interesting physical systems, you are refered to the more comprehensive tutorials available from our website.

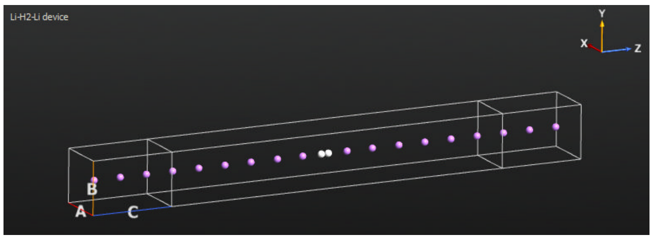


Figure 1.1:The Li-H_2-Li system to be studied in this tutorial.

The calculations will be carried out using **density-functional theory (DFT)** (for the electronic structure part) in combination with **non-equillibrium Green's function (NEGF)** techniques (for the transport part). A longer description of some more technical aspects of the parameters and models can be found in the ATK reference manual,

To set up the structures and calculations you will use the graphical interface Virtual NanoLab (VNL). It is recommended to first go through the introduction tutorial for VNL, just to become familiar with the fundamental concepts and how to operate the software.

The first step of the tutorial is to set up the geometry.

Chapter 2. Geometry for transport calculations

The fundamental structure of a device configuration

The type of atomic configuration needed for a transport calculation is fundamentally different from the other two common types of configurations used in atomic-scale simulations, viz. periodic and finite structures. In fact, a device configuration is a combination of these two: two periodic parts which define the left and right **electrodes** (source and drain, in a transistor perspective), and a finite part (called the "molecule" here) which provides the functionality of the device, and which causes the electrons and holes traveling between the two electrodes to be scattered.

Note

- The "molecule" does not have to be an actual molecule, you can e.g. have a layered material with a thin embedded film acting as the scattering center, in a metal-semiconductor-metal "sandwich", or a piece of a nanotube or graphene. It just has a finite extension in the transport direction.

- In addition to source and drain electrodes, ATK supports the inclusion of gates, for transistor simulations. This will not be covered here, but there are other tutorials on the topic.

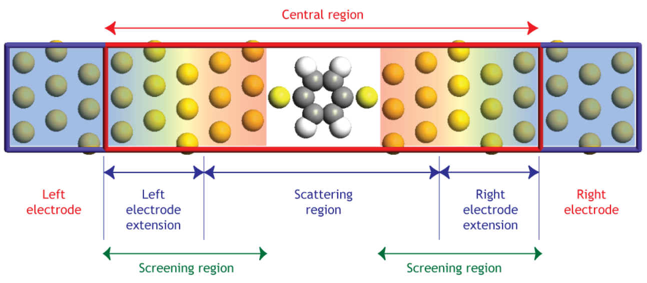
- In the example treated here the two electrodes are the same, but this is not generally necessary, which means ATK is actually able to treat a single interface between two different materials.

Technically, a device configuration in ATK is constructed by three configurations, placed next to each other (cf. Figure 1.1): the left and right electrodes, and the part between them, the so-called **central region**.

Since the electrodes are made from a bulk (or at least periodic) material, and the "molecule" is finite, there needs to be some kind of transition region between them, on both sides. These regions are rather naturally referred to as the **surfaces**. The surfaces are similar to the electrodes, in the sense that they consist of the same material, but since they are in close contact with the "molecule" they may undergo reconstruction, i.e. the ion positions may change, relative to the original periodic lattice of the electrodes.

Moreover, for numerical stability reasons, in ATK there is a region inside the central region where the atom positions are fixed to match the electrodes; these are the **electrode extensions**.

Combined, the electrode extensions and the surface regions are called the **screening region**. In these parts, which physically resemble the electrodes, the electron density and potential is computed self-consistently to provide a smooth transition from the "molecule" to the electrodes where the potential is bulk-like.



The picture above summarizes all these points.

Note

- The entire treatment of transport in this context is ballistic and elastic, i.e. there is no scattering by phonons. There is however scattering both in the electrode copies and the surfaces/molecule regions, since the self-consistent potential here is not bulk-like.
- The central region and the electrodes are equally important for transport properties - if either of them are insulating/semiconducting, no current will flow (at zero bias).

From central region to device - extracting the electrodes

From the discussion above, it should be clear that the electrodes are actually implicitly defined by the electrode extensions inside the central region, and this is also the most convenient way to construct the system, by building the central region, and then extracting the electrodes from it. You will now learn how to do this.

Defining the central region

The example system used in this tutorial is a hydrogen molecule acting as an impurity in an otherwise periodic lithium wire. The central region of this system is shown below, with the regions discussed above indicated by the colored bars .

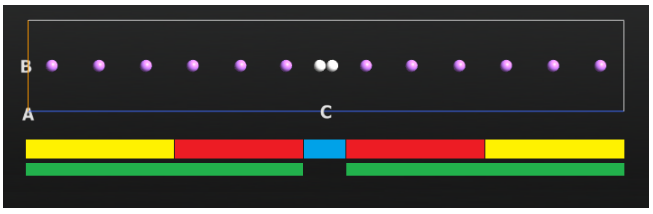


Figure 2.1:The central region, consisting of electrode extensions (yellow), surface layers (red), and molecule (blue). The screening region is also indicated (green).

The last chapter of this tutorial demonstrates how to properly construct the central region for this system. For now, you will use a pre-defined struture, and learn how to construct the device geometry from it.

The above figure is obtained using these lines of Python code:

```
# 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, 39.3702095157]*Angstrom
lattice = UnitCell(vector_a, vector_b, vector_c)

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

# Define coordinates
fractional_coordinates = [[ 0.5        ,  0.5        ,  0.03955015],
                          [ 0.5        ,  0.5        ,  0.11865069],
                          [ 0.5        ,  0.5        ,  0.19775123],
                          [ 0.5        ,  0.5        ,  0.27624721],
                          [ 0.5        ,  0.5        ,  0.35642874],
                          [ 0.5        ,  0.5        ,  0.43295913],
                          [ 0.5        ,  0.5        ,  0.48952536],
                          [ 0.5        ,  0.5        ,  0.51047676],
                          [ 0.5        ,  0.5        ,  0.56704061],
                          [ 0.5        ,  0.5        ,  0.64357304],
                          [ 0.5        ,  0.5        ,  0.72375085],
                          [ 0.5        ,  0.5        ,  0.80224875],
                          [ 0.5        ,  0.5        ,  0.88134932],
                          [ 0.5        ,  0.5        ,  0.96044984]]

# Set up configuration
bulk_configuration = BulkConfiguration(
    bravais_lattice=lattice,
    elements=elements,
    fractional_coordinates=fractional_coordinates
    )
```

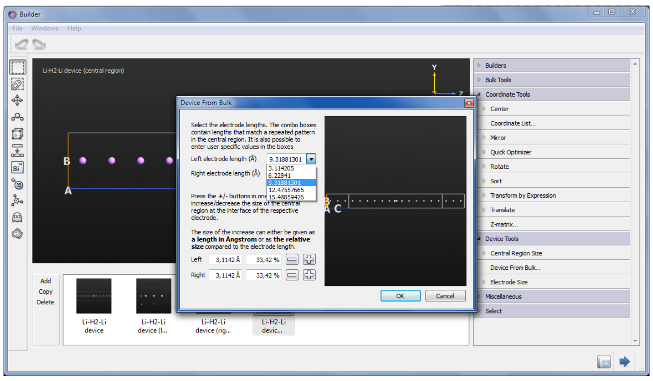
To import the structure, save the script as `central_region.py`, locate the file in the VNL main window and drag and drop it onto the Builder icon on the VNL toolbar.

Tip

You can also select the Python code above, and drag the text selection onto the Builder icon. Or, copy the selected code to the clipboard, then open the Builder and click `Add` → `From Clipboard` on the Stash menu.

Extracting the electrodes and forming the device configuration

In the Builder, open the plugin `Device Tools` → `Device from Bulk` from the right-hand side toolbar panel.


Device From Bulk

Select the electrode lengths. The combo boxes contain lengths that match a repeated pattern in the central region. It is also possible to enter user specific values in the boxes

Left electrode length (Å) 9.31881301

Right electrode length (Å)

3.114205
6.22841
9.31881301
12.47557665
15.48859426

Press the +/- buttons in one to increase/decrease the size of the central region at the interface of the respective electrode.

The size of the increase can either be given as **a length in Ångstrom** or as **the relative size** compared to the electrode length.

Left 3,1142 Å 33,42 %

Right 3,1142 Å 33,42 %

This tool will search for any periodicity in the structure on the left and right edges, and suggest possible electrodes. In this case, the central region is defined such that the first three lithium atoms have the same distance (about 3 Å) which you can see from the suggested electrode lengths in the dropdown lists.

Important

It would seem best to pick the smallest electrode possible, i.e. in this case a single Li atom. This will however not work. The reason for this is in some sense technical, and for practical purposes it is often sufficient to just recall a rule-of-thumb that the electrodes should be at least 7-8 Å long; any smaller value than that and there is a great risk that the results will be incorrect. A more complete explanation of this point can be found elsewhere.

In this case, the default suggested electrode (6.2 Å) is a bit on the low side, instead you should select the one which is about 9.3 Å, corresponding to three Li atoms. Click OK to add the device configuration to the stash.

Tip

To extract the electrodes separately, you may use the "Split Device" plugin from the left-hand side toolbar. This will add three new structures to the stash, corresponding to the two electrodes and the central region. You could now perform a band structure calculation of the electrode, to compare later to the transmission spectrum. This is left as an exercise, as it is not essential for the current presentation.

Notes on the device configuration geometry

Here are a few additional things to keep in mind about the device geometry.

- The transport direction in ATK is always Z. Therefore, the electrode cells (and the central region cell) should have the **C** axis (the 3rd unit cell vector) parallel to Z.

  The two other electrode unit cell vectors (**A** and **B**) should be perpendicular to Z, i.e. they should span the XY plane.

- There is no requirement that the electrodes consist of the same material. Both electrodes and the central cell must however have the same **A** and **B** unit cell vectors.

- The required number of screening layers needs to be checked for convergence.

Periodicity and boundary conditions

By themselves, the central region and the electrodes are defined as bulk configurations, and as such they are periodic in all three directions in space. When combined into a device configuration, the entire system is still periodic in the A and B directions, but only the electrodes are periodic in the transport direction C; in fact they are semi-infinite, and are repeated periodically only to one side, outwards from the central region.

Technically, the three configurations are joined into a DeviceConfiguration class object using the construction

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

Typically you will have variables called "left" and "right" electrodes, but it is really the order in this construction which matters, in terms of determining which is the left and right electrode, not the variable names.

The fact that the electrodes must be periodic in C has some consequences for the possible cell you can choose. For instance, if you create a 111 surface from an fcc crystal, the so-formed electrode must have a multiple of 3 layers.

During the calculation of the device, the central region is not treated as periodic in the transport direction. Instead, open boundary conditions apply, and charge may flow in and out of the central region.

Due to the overall periodicity in A and B it is necessary to add some vacuum around the one-dimensional wire, to avoid interactions with the repeated images in these directions. How much vacuum is needed should be checked as a convergence parameter.

Summary

In ATK, a **device configuration** is a way of representing the atomic structure of two semi-infinite **electrodes** plus some different structure between them. The easiest way to set up a device configuration is usually to use VNL, and first define the central region, which should include a copy of the intended electrodes, and then convert it into a device. The more detailed procedure for setting up - and importantly also optimizing - the central region will be shown in the last chapter of this tutorial.

Having looked at how to define the atomic structure, you are now ready to move on to set up and run the transport calculation.

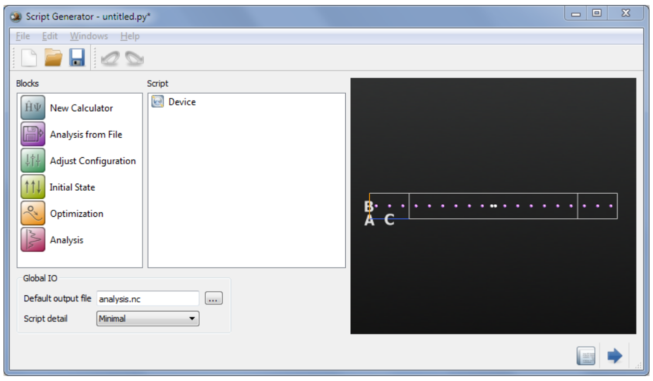
Chapter 3. Setting up a transport calculation with the Script Generator

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

Opening a device configuration in the Script Generator

Send the device geometry you just constructed in the previous chapter to the Script Generator by using the "Send to" icon in the lower right-hand corner of the Builder.

The **Script Generator** should open up and display the device geometry.



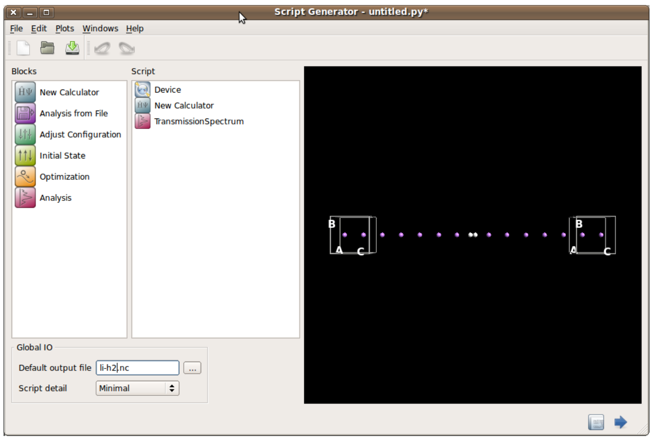
Default output file: analysis.nc

Script detail: Minimal

Setting up the parameters of the calculation

To set up the calculation of the transmission spectrum

1. double-click **New Calculator** in the "Blocks" panel to add it to the script panel,
2. double-click the **Analysis**-icon and select `TransmissionSpectrum` from the menu, and finally
3. change the output file name to `li-h2.nc`.



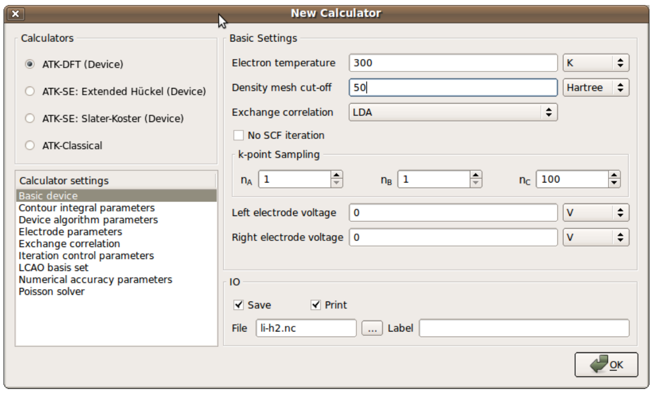
Global IO

Default output file: li-h2.nc

Script detail: Minimal

To modify the parameters of the calculator, double-click **New Calculator** in the "Script" panel.

To speed up the calculation, change the **Density mesh cut-off** to 50 Hartree.



Calculators: (●) ATK-DFT (Device) ( ) ATK-SE: Extended Hückel (Device) ( ) ATK-SE: Slater-Koster (Device) ( ) ATK-Classical

Basic Settings

Electron temperature: 300 K

Density mesh cut-off: 50 Hartree

Exchange correlation: LDA

[ ] No SCF iteration

k-point Sampling: n_A 1, n_B 1, n_C 100

Left electrode voltage: 0 V

Right electrode voltage: 0 V

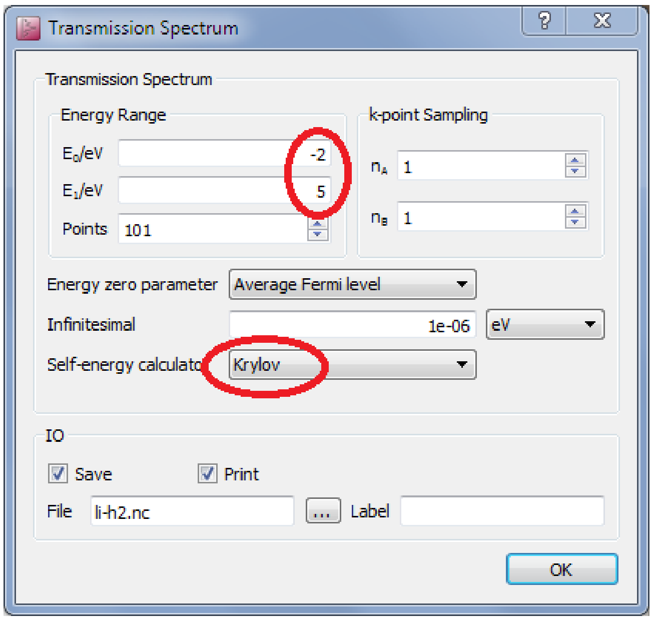
IO: [✓] Save [✓] Print

File: li-h2.nc Label:

Click **OK** to close the dialog.

Now double-click **TransmissionSpectrum** in the "Script" panel.

Adjust the energy range to [-2, 5] eV, and change the self-energy calculator to Krylov (the fastest method).



Transmission Spectrum

Energy Range

E_0/eV: -2

E_1/eV: 5

Points: 101

k-point Sampling

n_A: 1

n_B: 1

Energy zero parameter: Average Fermi level

Infinitesimal: 1e-06 eV

Self-energy calculator: Krylov

IO: [✓] Save [✓] Print

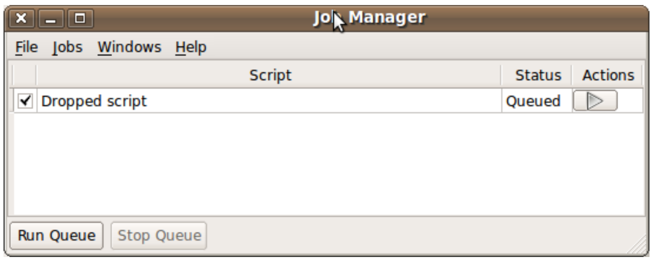
File: li-h2.nc Label:

Again, click **OK**.

The script is now ready to be run. It is however good practice to always save the input script for future reference, so first choose `File` → `Save as...` from the menu and save the generated Python script under a suitable name, like `li-h2.py`.

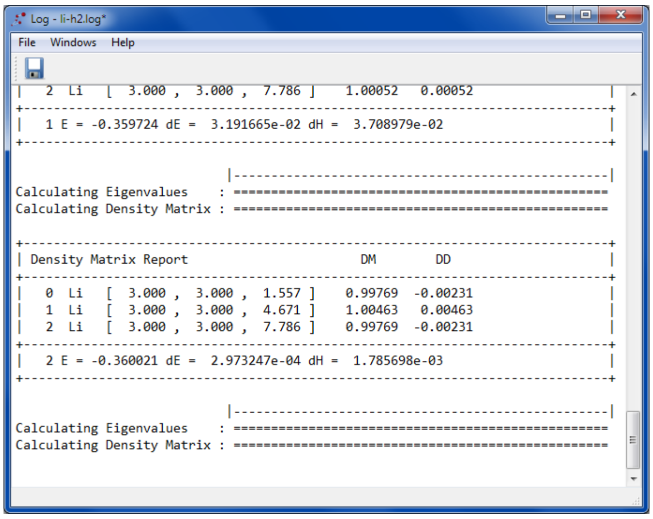
Running the script

Send the script to the **Job Manager** by pressing the "Send to" button in the lower right-hand corner of the Script Generator, and select `Job Manager`.



	Script	Status	Actions
✓	Dropped script	Queued	▷

Click **Run Queue** to start the calculation. A window appears with log messages about how the calculation procedes. The job should finish in a couple of minutes only. You can save the log output for future reference by pressing Ctrl+S in the log window.



```
|   2  Li   [  3.000 ,  3.000 ,  7.786 ]    1.00052   0.00052                |
+---------------------------------------------------------------------------+
|   1 E = -0.359724 dE =  3.191665e-02 dH =  3.708979e-02                    |
+---------------------------------------------------------------------------+

                            |-----------------------------------------------|
Calculating Eigenvalues    : ================================================
Calculating Density Matrix : ================================================

+---------------------------------------------------------------------------+
| Density Matrix Report                          DM        DD                |
+---------------------------------------------------------------------------+
|   0  Li   [  3.000 ,  3.000 ,  1.557 ]    0.99769  -0.00231                |
|   1  Li   [  3.000 ,  3.000 ,  4.671 ]    1.00463   0.00463                |
|   2  Li   [  3.000 ,  3.000 ,  7.786 ]    0.99769  -0.00231                |
+---------------------------------------------------------------------------+
|   2 E = -0.360021 dE =  2.973247e-04 dH =  1.785698e-03                    |
+---------------------------------------------------------------------------+

                            |-----------------------------------------------|
Calculating Eigenvalues    : ================================================
Calculating Density Matrix : ================================================
```

You have now performed a complete transport calculation with ATK! Granted, it was only for zero bias, and you have yet to analyze the results, but the fundamental work flow is the same for all transport calculations as for this system.

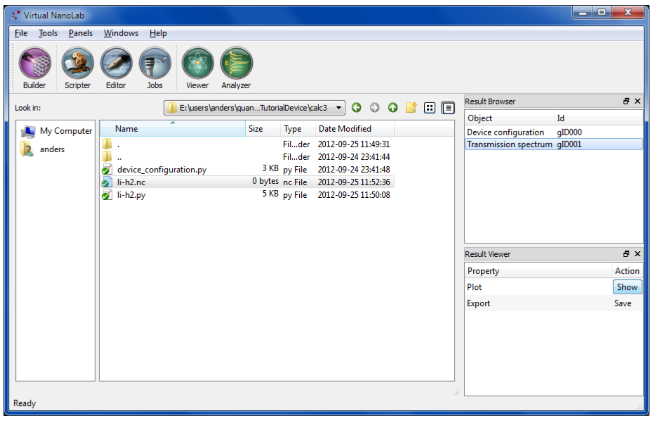
Visualizing the results

In order to visualize the results of the calculation you just performed, go through the following steps:

1. Locate and select the file `li-h2.nc` in the VNL main window.

2. Select the **Transmission spectrum** in the list of objects in the file, shown in the **Result Browser** (the upper right panel).

3. Click **Show** next to the "Plot" command in the **Result Viewer** (the lower right panel).



Name	Size	Type	Date Modified
.		Fil...der	2012-09-25 11:49:31
..		Fil...der	2012-09-24 23:41:44
device_configuration.py	3 KB	py File	2012-09-24 23:41:48
li-h2.nc	0 bytes	nc File	2012-09-25 11:52:36
li-h2.py	5 KB	py File	2012-09-25 11:50:08

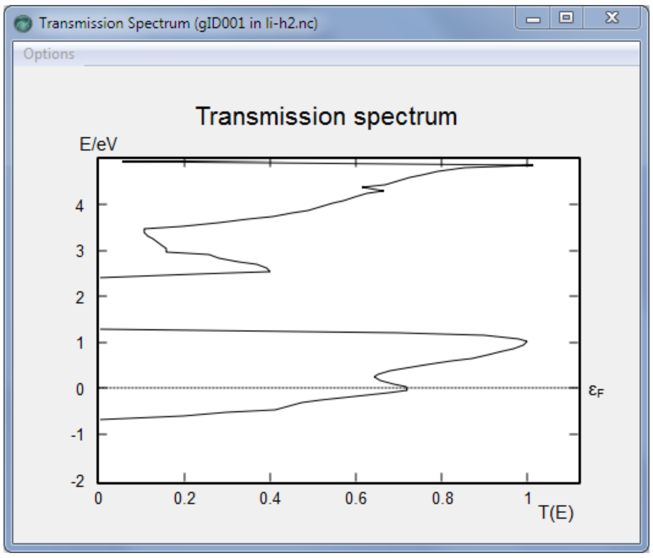
Result Browser

Object	Id
Device configuration	gID000
Transmission spectrum	gID001

Result Viewer

Property	Action
Plot	Show
Export	Save

A plot of the transmission spectrum will now appear. You can right-click the plot and choose "Anti aliasing" for improved plot quality.



By right-clicking on the window the plot can be saved by choosing `Export image` from the contex menu. The data contained in the plot can also be exported for use in a third-party plotting software by choosing `Export data`.

Do not close the Script Generator, as you will reuse it in the next section.

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.

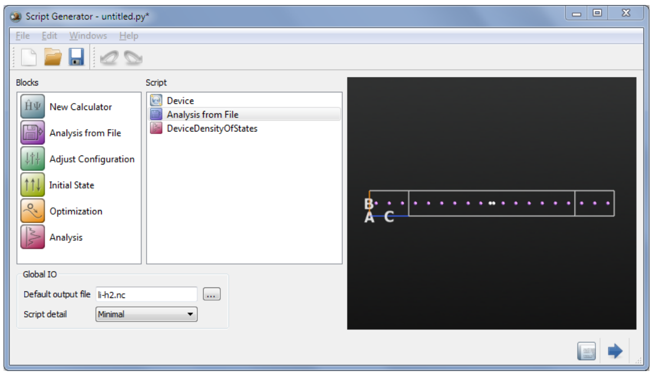
For instance, you will first want to look at the transmission spectrum in order to find out at which energies the electron transfer is strongest, and then later go in and study in more detail e.g. the local density of states at this energy. In this case, there is no need to redo the self-consistent calculation.

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.



Device
Analysis from File
DeviceDensityOfStates

Global IO
Default output file: li-h2.nc
Script detail: Minimal

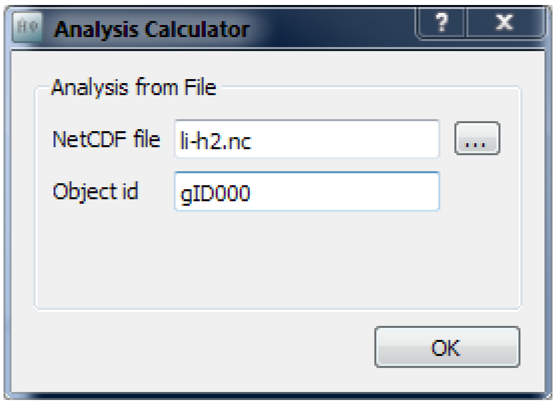
Tip

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`.

Tip

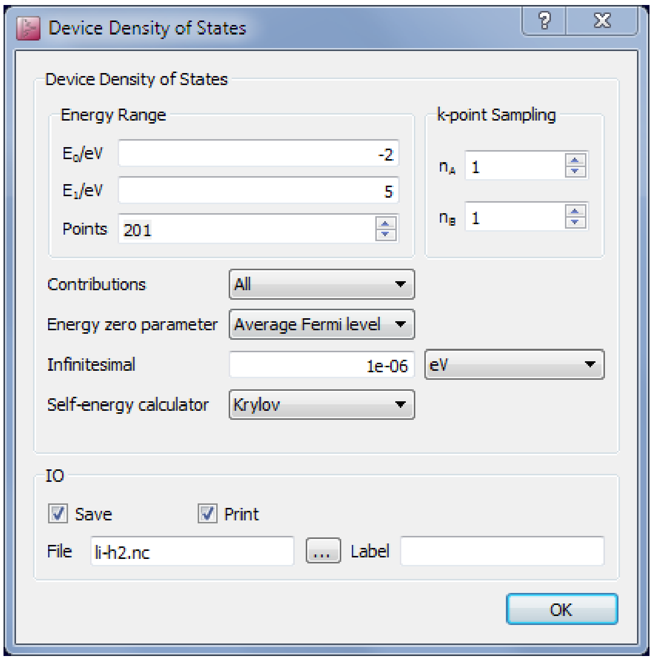
If the file contains several configurations, you can specify which one to restore by providing its **ID**, which is a unique identifier for each object in the NetCDF file. The first configuration almost always has **Id** "gID000", but in more general cases (e.g. after running a geometry optimization)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



NetCDF file: li-h2.nc
Object id: gID000

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.



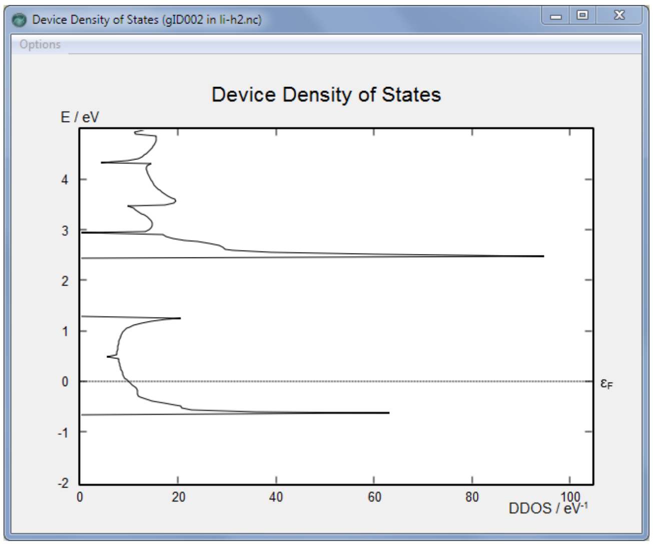
E_0/eV: -2
E_1/eV: 5
Points: 201
n_A: 1
n_B: 1
Contributions: All
Energy zero parameter: Average Fermi level
Infinitesimal: 1e-06 eV
Self-energy calculator: Krylov
☑ Save ☑ Print
File: li-h2.nc
Label:

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.

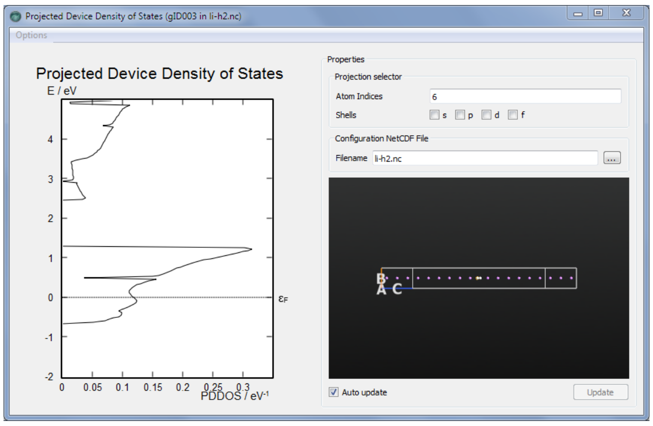
Send the calculation to the **Job Manager** and start the job in the same way as with the calculation of the transmission spectrum. It should only take seconds to run.

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.



Projected Device Density of States (gID003 in li-h2.nc)
Options
Projected Device Density of States
E / eV
4
3
2
1
0
-1
-2
0
0.05
0.1
0.15
0.2
0.25
0.3
PDDOS / eV^{-1}
ε_F
B
A C
Update

Properties

Projection selector

Atom Indices 6

Shells ☐ s ☐ p ☐ d ☐ f

Configuration NetCDF File

Filename li-h2.nc

☑ Auto update

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.

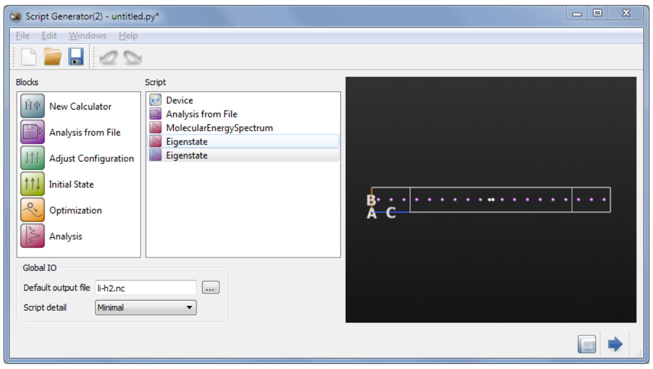
Tip

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.


Script Generator(2) - untitled.py*

Blocks: New Calculator, Analysis from File, Adjust Configuration, Initial State, Optimization, Analysis

Script:
- Device
- Analysis from File
- MolecularEnergySpectrum
- Eigenstate
- Eigenstate

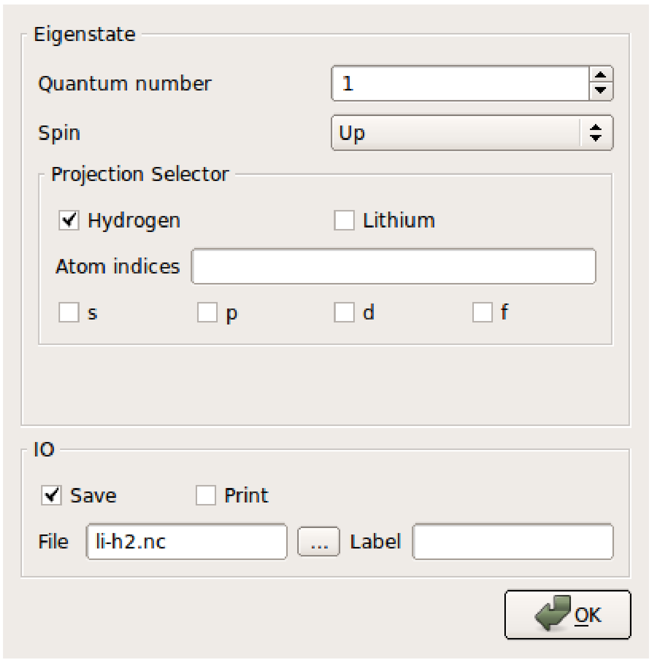
Global IO

Default output file: li-h2.nc

Script detail: Minimal

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.


Eigenstate

Quantum number: 1

Spin: Up

Projection Selector

[✓] Hydrogen   [ ] Lithium

Atom indices:

[ ] s   [ ] p   [ ] d   [ ] f

IO

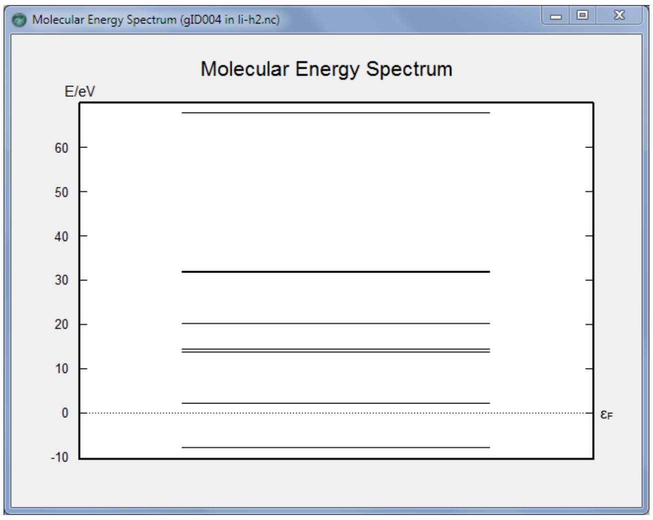
[✓] Save   [ ] Print

File: li-h2.nc   ...   Label:

OK

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

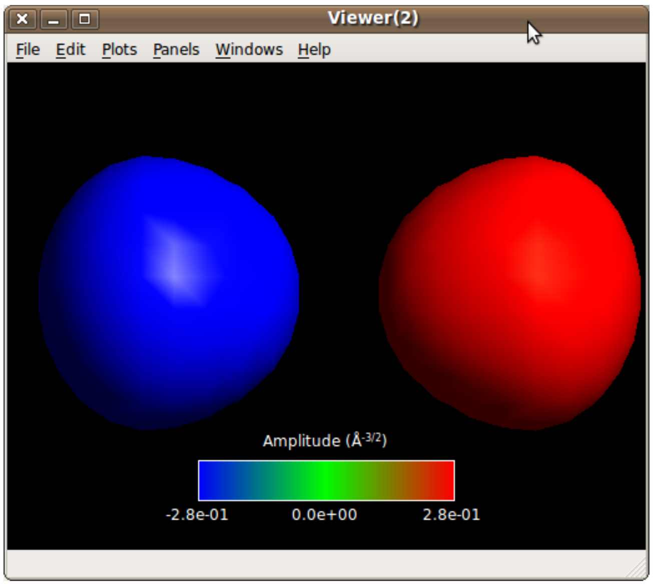
The data objects can be inspected from the Result Viewer in the VNL main window. To view the MPSH spectrum, select the **Molecular energy spectrum** object in the `li-h2.nc` file and click the Show button next to Plot in the Result Viewer.



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.

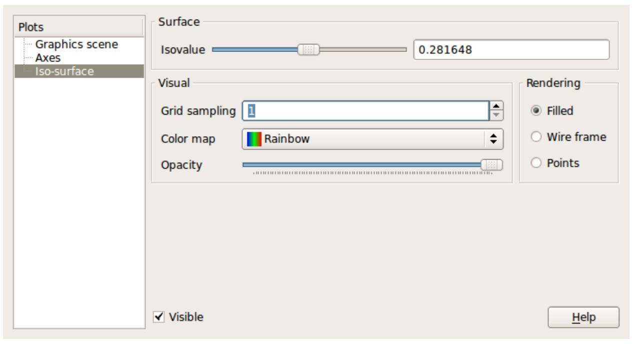
Tip

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` form the popup menu.
4. Set the grid sampling to 1 for the isosurface.


Plots
Graphics scene
Axes
Iso-surface

Surface
Isovalue: 0.281648

Visual
Grid sampling: 1
Color map: Rainbow
Opacity

Rendering
[✓ Filled]
Wire frame
Points

✓ Visible

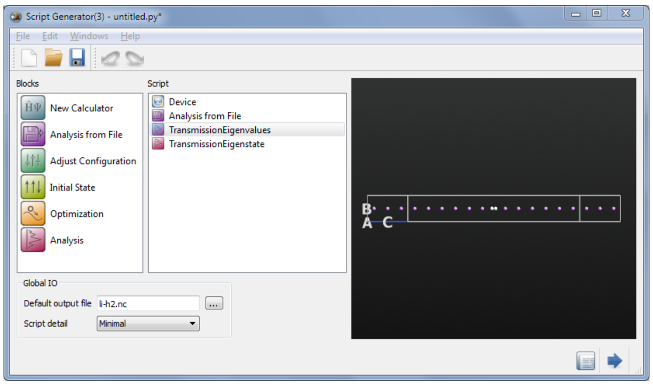
Help

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**).


Script Generator(3) - untitled.py*

File Edit Windows Help

Blocks
New Calculator
Analysis from File
Adjust Configuration
Initial State
Optimization
Analysis

Script
Device
Analysis from File
TransmissionEigenvalues
TransmissionEigenstate

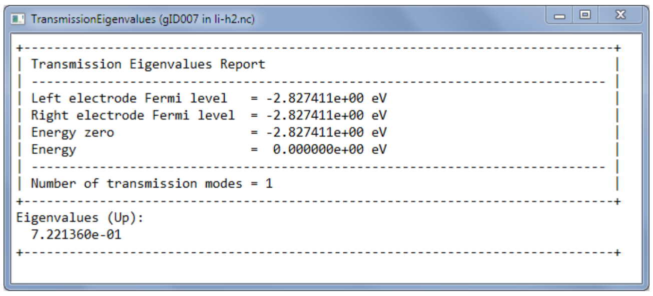
Global IO
Default output file: li-h2.nc
Script detail: Minimal

We will stay with the default parameters, which means to perform the transmission eigenchannel analysis at the Fermi energy, at the Γ 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 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.



```
TransmissionEigenvalues (gID007 in li-h2.nc)
+------------------------------------------------------------------------------+
| Transmission Eigenvalues Report                                              |
| ---------------------------------------------------------------------------- |
| Left electrode Fermi level   = -2.827411e+00 eV                              |
| Right electrode Fermi level  = -2.827411e+00 eV                              |
| Energy zero                  = -2.827411e+00 eV                              |
| Energy                       =  0.000000e+00 eV                              |
| ---------------------------------------------------------------------------- |
| Number of transmission modes = 1                                             |
+------------------------------------------------------------------------------+
Eigenvalues (Up):
  7.221360e-01
+------------------------------------------------------------------------------+
```

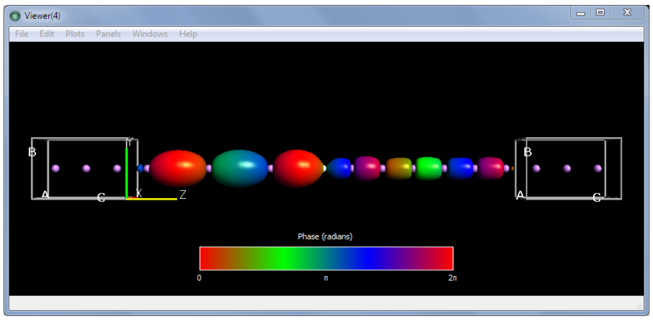
The transmission eigenchannel can be visualized as an isosurface, by selecting the Transmission Eigenstate object in `li-h2.nc` in the Result Browser, and clicking **Isosurface>Show** in the Result Viewer. The transmission eigenstate is a 3D complex function, and the isosurface is given by the absolute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

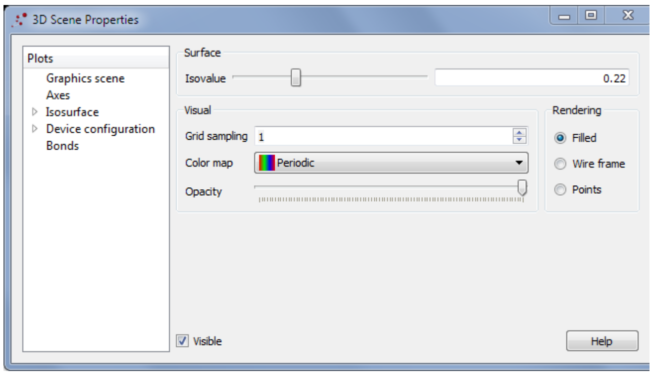
Note

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.



Surface
Isovalue 0.22

Visual
Grid sampling 1
Color map Periodic
Opacity

Rendering
[✓ Filled] Wire frame Points

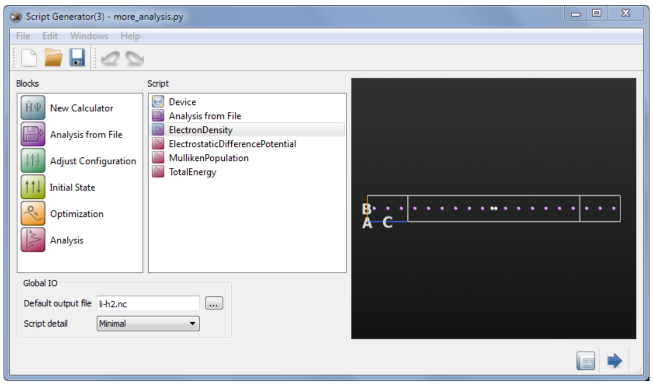
[✓ Visible]

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**



Script
- Device
- Analysis from File
- ElectronDensity
- ElectrostaticDifferencePotential
- MullikenPopulation
- TotalEnergy

Global IO
Default output file li-h2.nc
Script detail Minimal

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 CURVE

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

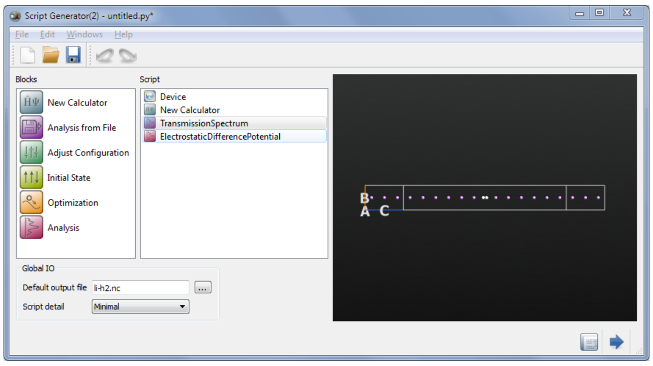
VNL has a convenient tool that takes a NetCDF-file containing transmission spectra taken at different bias and calculates the I-V curve. To use this tool, you just need to set up the calculations that generate the transmission spectra at the desired bias points. In this chapter you will first do it manually for a single value of the bias, and then learn how to set up a loop to generate a whole sequence of finite bias calculations.

SELF-CONSISTENT CALCULATION AT FINITE BIAS - A SINGLE POINT

SETTING UP THE CALCULATOR

In the VNL main window, select the file `li-h2.nc` and drag the Device Configuration from the Result Browser onto the **Script Generator** icon. Change the output file name to `li-h2.nc` - the finite bias calculation will be saved in the same file as the zero-bias results, so that the I-V curve can easily be plotted.

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.


Script Generator(2) - untitled.py*

Blocks: New Calculator, Analysis from File, Adjust Configuration, Initial State, Optimization, Analysis

Script: Device, New Calculator, TransmissionSpectrum, ElectrostaticDifferencePotential

Global IO

Default output file: li-h2.nc

Script detail: Minimal

Now double-click the **New Calculator** in the "Script" panel and set the **Right electrode voltage** to 1 V.

Also open the **TransmissionSpectrum** block and change the range to **[-2, 5] eV** and select the **Krylov** method to match the already calculated transmission spectrum for zero bias.

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

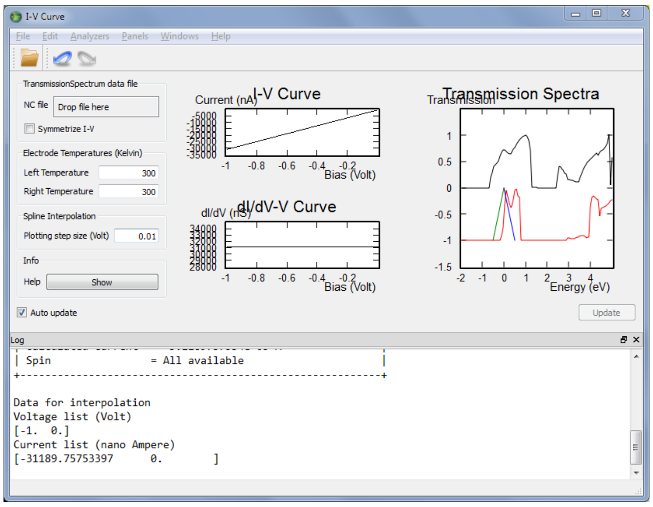
Calculating the I-V curve

Launch the **Custom Analyzer** tool by clicking the **Analyzer** icon 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 built-in I-V curve analyzer by selecting `Analyzers` → `I-V Curve` from the menu.

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



```
| Spin                    = All available                      |
+--------------------------------------------------------------+

Data for interpolation
Voltage list (Volt)
[-1.  0.]
Current list (nano Ampere)
[-31189.75753397     0.        ]
```

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 I-V curve tool 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 the different bias voltages; the spectra have been displaced vertically for clarity. The green/blue wedge embraces the part of the transmission which is inside the bias window.

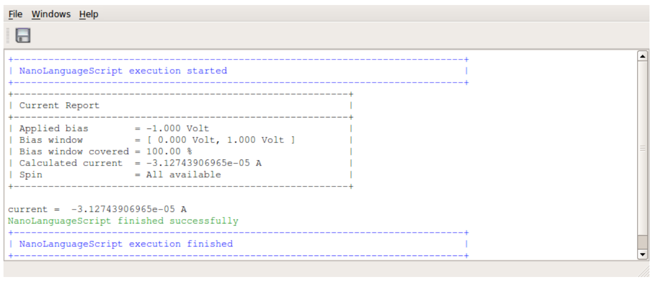
Visualizing the voltage drop

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

```
# Read in the transmission spectra from the NetCDF file
transmission_spectra = nlread("li-h2.nc",TransmissionSpectrum)

# Compute the current for all of them
```

```
for t in transmission_spectra:
    t.current()
```

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

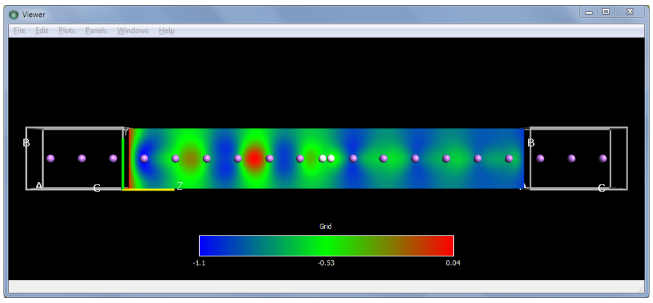


```
+------------------------------------------------------------------------------+
| NanoLanguageScript execution started                                         |
+------------------------------------------------------------------------------+
+------------------------------------------------------------+
| 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 to compute the voltage drop across the device. Assuming that you have performed the earlier calculations and that the NetCDF file `li-h2.nc` is available, you can calculate the voltage drop by dropping the following script named `voltage_drop.py` on the **Job Manager**.

```
# Read the electrostatic difference potentials
potential = nlread('li-h2.nc', ElectrostaticDifferencePotential)

# Calculate the voltage drop
voltage_drop = potential[1]-potential[0]

# Save the voltage drop, properly labeled
nlsave('li-h2.nc', voltage_drop, object_id='voltagedrop')
```

This will add a new electrostatic difference potential object to `li-h2.nc`. A convenient ID "voltagedrop" is attached to the object for easy identification in the Result Browser. Select the object there and plot it as a contour plot. Then drag the **Device configuration** onto the open Viewer window.



The potential in the right part of the device is -1 eV, corresponding to the applied bias. (The colorbar extends a bit further, but the potential is exactly -1 eV on the right-hand side edge of the central region, since this is the boundary condition set in the Poisson equation solver.)

Note

The relatively high current running through the device at 1 Volt has the effect that the atoms in the central region cannot reach an equilibrium electron distribution. Part of the assumption in the computational model is that atoms close to the electrodes have an equilibrium distribution, but results from one-dimensional 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 reaches 0 eV until the very edge (where it is constrained to attain this value, the sharp red area).

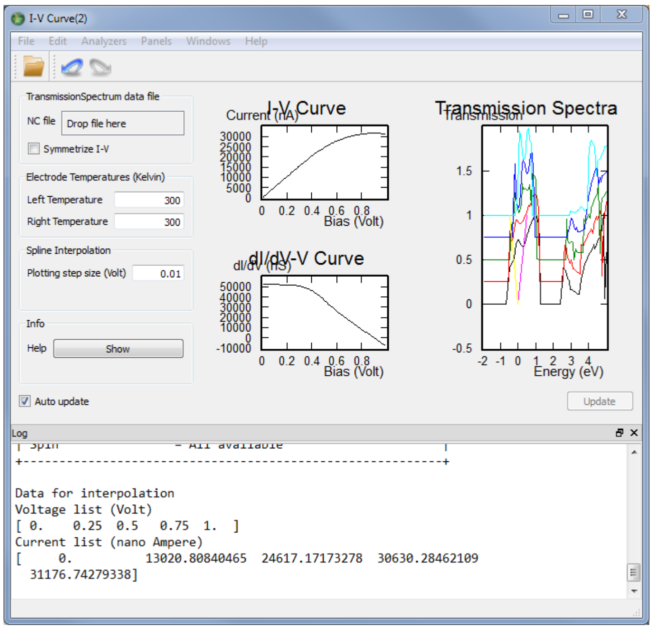
Performing an I-V scan with scripting

The I-V curve computed above only contained two points - normally you will want at least 5-10 points on the curve. In that case it is much more convenient to use Python scripting, rather than manually setting up a new script for each bias. Below is a script which performs self-consistent calculations at 0.25, 0.5, 0.75 and 1 Volt.

```
def transmission(configuration):
    # Calculate the transmission spectrum and save it in a file
    transmission_spectrum = TransmissionSpectrum(
        configuration=device_configuration,
        energies=numpy.linspace(-2,5,101)*eV,
    )
    nlsave('li-h2_iv.nc', transmission_spectrum)

# Read in the converged zero-bias calculation
device_configuration = nlread("li-h2.nc",DeviceConfiguration)[0]
calculator = device_configuration.calculator()

# Calculate and save the transmission spectrum for zero bias
transmission(device_configuration)

# Define the bias voltages for the I-V curve
voltage_list=[0.25, 0.5, 0.75, 1.0]*Volt

# Loop over the bias voltages
for voltage in voltage_list:
    # Set electrode voltages and use the self-consistent state
    # of the previous calculation as starting guess
    device_configuration.setCalculator(
          calculator(electrode_voltages=(voltage/2, -voltage/2)),
          initial_state=device_configuration)

    # Calculate and save the transmission spectrum for each bias
    transmission(device_configuration)
```

The script is very generic can be used for almost any system, where you have a converged zero-bias calculation already. The script also has an additional advantage compared to the manual approach: the calculation for the next higher bias point uses the converged calculation for the lower bias as starting guess. This reduces the calculation time considerably, and can in fact in larger systems mean the difference between successfully converging the finite-bias calculation and not.

When the calculation is finished, you can again use the Custom Analyzer (drop the NetCDF file `li-h2_iv.nc` on the dropzone) to get a more detailed I-V curve.



```
| Spin                  - All available                |
+------------------------------------------------------+

Data for interpolation
Voltage list (Volt)
[ 0.    0.25  0.5   0.75  1.  ]
Current list (nano Ampere)
[     0.          13020.80840465  24617.17173278  30630.28462109
  31176.74279338]
```

Chapter 6. Building and optimizing the geometry

In the previous part of the tutorial an important part of a solid-state calculation was tacitly ignored - calculating the positions of the atoms corresponding to the lowest total energy. In this chapter you will learn how to set up an initial guess for the structure using VNL and subsequently finding the structure where the atoms experience no forces or stress - the optimized geometry. Potentially, the optimized geometry can have completely different properties than the initial guess, thus the optimization procedure (or relaxation, as it is sometimes referred to as) is always an important step in the setup of any system.

Constructing an optimized device structure is somewhat more complicated than the more conventional case of relaxing a molecule or a periodic structure, not least because a device configuration actually consists of three separate configurations (the two electrodes and the central region), which are combined to form the device. Also, the rule discussed in ... that the first/last atoms in the central region must perfectly match the electrodes means special care has to be taken to constrain these atoms from moving during an optimization.

This tutorial will introduce a straightforward strategy which works well in most cases:

1. Make a reasonable starting guess for the electrodes. This is often simple, as the electrodes are typically made of some known periodic structure like a cleaved metal surface, a nanotube, or as in our case a chain of atoms.

2. Optimize the electrodes, both the positions of the atoms and the size of the cell, in all directions. For this step it is possible to use the minimal electrode configuration, to save time.

3. Determine the required electrode lengths by ensuring that the electrodes are long enough that there are no (or negligible) interactions with the second-nearest neighbor cell in the transport direction.

4. Use this information to construct a starting guess for the central cell. A convenient approach is:

    - Start with a system corresponding to two repetitions of each electrode (with the proper length).

    - Place the desired central configuration between these extended electrodes. Here, the big unknown is the distance between the surfaces and the central material, which affects not only the positions but also the size of the central cell, i.e. the distance between the electrodes. This is why it is so important to optimize the central region!

    Note

    These first two steps should be considered conceptually rather than literally. In many cases you would rather first add the central region to the extended left electrode, and then add the extended right electrode to that. Or, you would somehow modify a longer repetition of the electrode structure to form the central region, e.g. by removing or adding some atoms.

- Then, use the function in VNL to extract the electrodes from the central region to form the device configuration.
- Remove or add more surface layers as needed for proper screening and to allow surface reconstruction during the optimization. There is a convenient function in VNL for this as well.

5. Optimize the atomic positions and length in Z of the central region, using the following trick: by running a device calculation non-self-consistently in ATK, the central region is still run self-consistently, and can hence be optimized, under periodic boundary conditions.

   Note

   The reason it is convenient to do it this way, rather than just optimize the central region, is that the electrode extensions are automatically constrained in a device configuration.

   This step can be skipped, but without it, the next step, the final optimization, can be hard to converge.

6. Use the result of the previous step as a good starting guess for the final optimization of the central region under open boundary conditions (at zero bias).
7. One can, in principle, furthermore optimize the device for each value of the electrode voltages, before computing the transmission spectrum for each bias.

Notes

- It may be sufficient to just use the central region, optimized under periodic boundary conditions, for the transport calculations, but this should be checked from case to case. One way to do this is to compute the forces on the atoms, computed under open boundary conditions for the configuration obtained from the optimization with periodic boundary conditions, are still smaller than the force criterion (or at least small enough).
- The impact of the bias on the atomic positions can often be neglected, and the small changes in positions induced by the bias typically have a much smaller effect on quantities like the current or populations, compared to the electrostatic effect of the bias itself. Thus re-optimizing the structure for each bias is typically only needed if one is specifically interested in this effect.

You will now apply this methodology to build and optimize the system used in this tutorial - a hydrogen molecule embedded in a linear chain of lithium atoms.

Electrode geometry

The first task is to determine the electrode geometry, by constructing a starting guess and optimizing it.

Initial guess for the electrode geometry

You will now construct an infinite one-dimensional lithium wire using the **Builder** in VNL. Start the Builder by clicking the icon on the Toolbar in the main VNL window.

To create a new structure, click the button **Add** in the menu to the left of the stash and select `New configuration` from the menu that pops up. This will insert a new, trivial configuration: a hydrogen atom in empty space.

The first thing to do is inscribe this atom in a unit cell, since you need a periodic structure. Expand the plugin group "Edit Directly" and open the "Edit Lattice" plugin. As you see, there is currently no cell for this structure, but if you choose "Unit cell" at the end of the drop-down list, you can edit the size of the cell.

Set the length of the A and B vectors to 9 Å (pointing in the X and Y directions, as they are already) to make a large enough cell in the transverse direction to eliminiate interactions with the repeated copies of the chain. Then, set the length of the C vector (along Z) to 2.2 Å. This is just a guess; you will optimize the value in a short moment.

Close the "Edit Lattice" plugin.

Tip

You can reset the configuration view in the **Builder** by pressing **Ctrl+R** to bring the entire configuration into easy view.

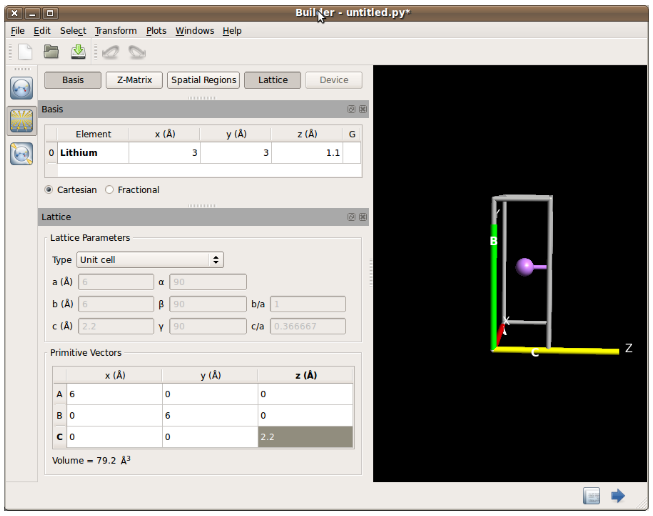
To change the atom to lithium, select the hydrogen atom, click the "Change Element" button in the left-hand side toolbar, and choose Li in the periodic table that appears.

Finally, center the structure in the cell in all directions by using the plugin "Center", available in the plugin group "Tranform Coordinates" (click Apply).

Note

To remove all interactions in X and Y the box should be larger than the basis set interaction range, which is about 8 Ångström. There is, in principle, also electrostatic interactions which can have quite a long range, and so it is necessary to check that the transverse unit cell size is large enough in each case. One way is to look for broken degeneracies in the band structure of the electrode, as such can only occur as a result of a violation of the translational invariance, not from insufficient k-point sampling or a too small basis set; another is to compute the band structure and check that the bands are completely flat in the XY plane.

You now have an infinite, periodic lithium chain with an interatomic distance of 2.2 Ångstrom. Since the cell is also periodic in the X and Y directions, in reality it is actually an infinite array of chains, separated by a distance of 9 Ångström.



Basis

	Element	x (Å)	y (Å)	z (Å)	G
0	**Lithium**	3	3	1.1	

[✓ Cartesian] Fractional

Lattice

Lattice Parameters

Type: Unit cell

a (Å) 6 α 90

b (Å) 6 β 90 b/a 1

c (Å) 2.2 γ 90 c/a 0.366667

Primitive Vectors

	x (Å)	y (Å)	z (Å)
A	6	0	0
B	0	6	0
C	0	0	2.2

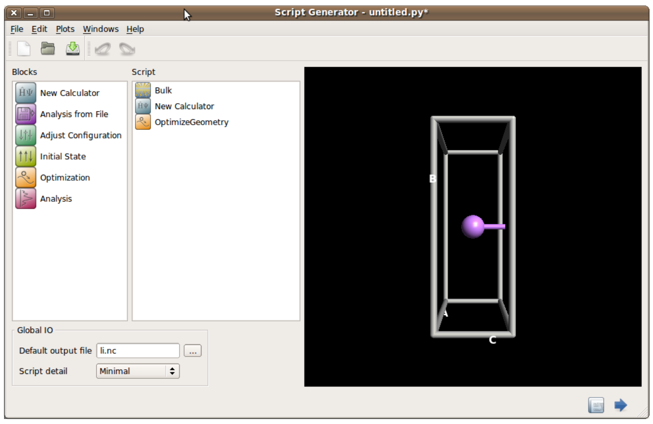
Volume = 79.2 Å^3

Optimizing the electrode geometry

You will now use ATK-DFT to optimize the electrode geometry. Press the "Send To" button in the lower right corner of the Builder and select **Script Generator**.

In the Script Generator, add a **New Calculator** and a **Optimize Geometry** block by double-clicking them in the left panel.

Change the output file name to `li.nc`.



Global IO

Default output file: li.nc

Script detail: Minimal

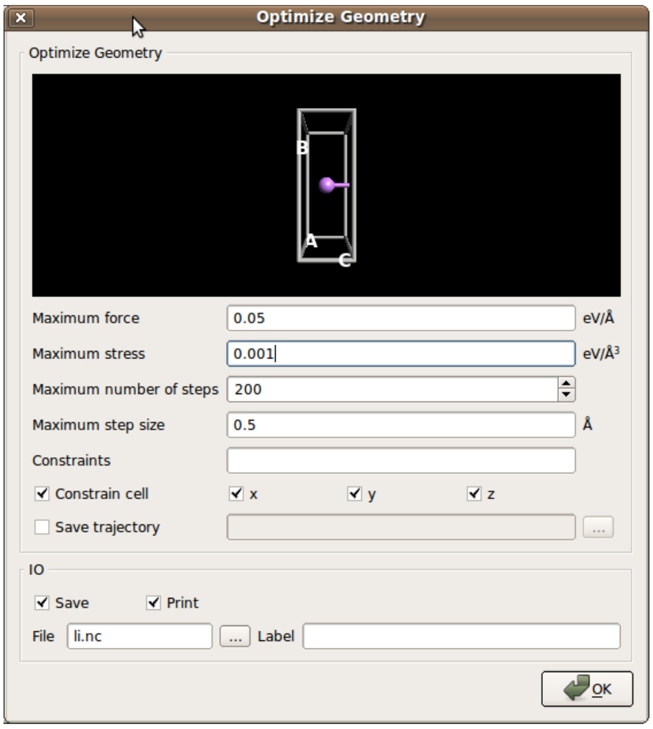
Double-click the **New Calculator** item in the Script panel to open it, and

- reduce the **mesh cut-off** to 50 Hartree to save some time;
- set the number of **k**-points in the C direction to 50 - a high value, but the calculation will be fast, and this is the most important parameter for good accuracy for this calculation.

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

Open the **Optimize Geometry** block in the Script panel and

- remove the tick from the cell constraint in the Z direction (the cell does not need to be optimized in the X and Y directions)
- Set the stress tolerance to 0.001 eV/Å3.



Maximum force 0.05 eV/Å
Maximum stress 0.001 eV/Å³
Maximum number of steps 200
Maximum step size 0.5 Å
Constraints
✓ Constrain cell ✓ x ✓ y ✓ z
Save trajectory
IO
✓ Save ✓ Print
File li.nc Label

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

Send the script to the Job Manager and run it; it should only take about a minute.

When the optimization has finished, select the file `li.nc` in the main VNL window. In the Result Browser you can see that the file contains two bulk configurations - the original geometry (gID000) and the optimized one (gID001).

Drag and drop the optimized configuration on the Builder icon on the VNL toolbar. In the Builder, change the title of the optimized configuration by selecting it in the stash, then press F2 and set the name to for instance "Optimized Li chain", so we can easily disinguish it from the initial guess.

If you open the "Edit Lattice" plugin again, you will notice that the chain lattice constant has increased to 3.03 Å. (You could also see this in the log file at the end of the optimization.) Thus our initial guess was not a very good one, but at least for this simple system it was easy enough to find the optimal value quickly anyway.

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 set up an initial guess in the the **Builder** and then the cell length in the Z direction will be optimized with ATK-DFT.

Initial guess for the central region geometry

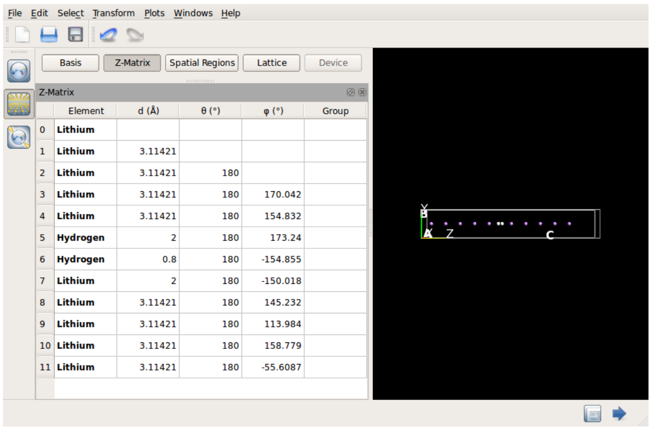
Start by repeating the optimized lithium chain 14 times in the C direction, using the "Repeat" plugin in the "Change Size" plugin group (press **Ctrl+R** to bring the system into comfortable view).

Select the two middle atoms (hold down Ctrl while left-clicking to select multiple atoms, or use e.g. the rectangle select tool from the toolbar to draw a box around the two atoms), and change them to hydrogen, using the "Change Element" button in the left-hand side toolbar.

The H-H and Li-H bond lengths are not expected to be as large as the Li-Li distance of 3.03 Å. between the hydrogen atoms, and between Ha nd Li,the hydrogen atoms and the litNext 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

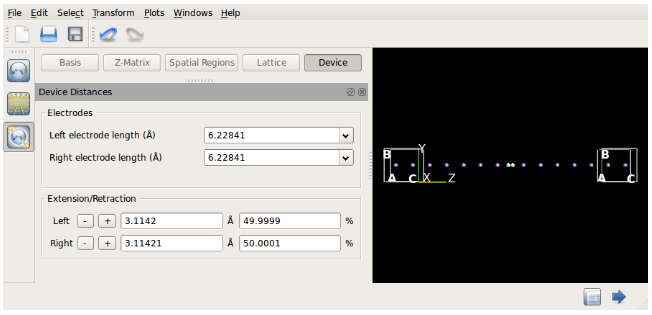


Z-Matrix

	Element	d (Å)	θ (°)	φ (°)	Group
0	Lithium				
1	Lithium	3.11421			
2	Lithium	3.11421	180		
3	Lithium	3.11421	180	170.042	
4	Lithium	3.11421	180	154.832	
5	Hydrogen	2	180	173.24	
6	Hydrogen	0.8	180	-154.855	
7	Lithium	2	180	-150.018	
8	Lithium	3.11421	180	145.232	
9	Lithium	3.11421	180	113.984	
10	Lithium	3.11421	180	158.779	
11	Lithium	3.11421	180	-55.6087	

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.



Device Distances

Electrodes

Left electrode length (Å) 6.22841

Right electrode length (Å) 6.22841

Extension/Retraction

Left - + 3.1142 Å 49.9999 %

Right - + 3.11421 Å 50.0001 %

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 .

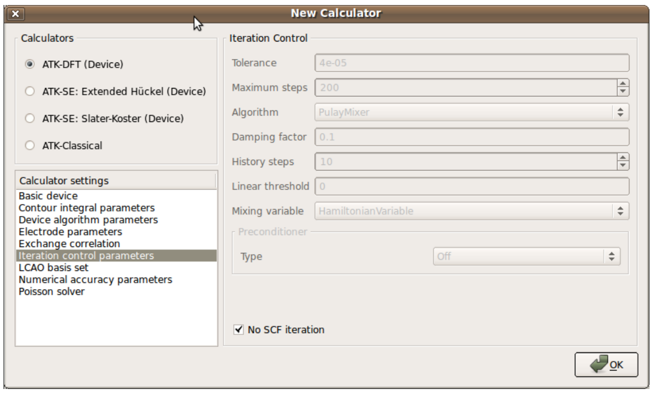
In the script generator

- 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.


New Calculator

Calculators
(● ATK-DFT (Device)) ATK-SE: Extended Hückel (Device) ATK-SE: Slater-Koster (Device) ATK-Classical

Calculator settings: Basic device, Contour integral parameters, Device algorithm parameters, Electrode parameters, Exchange correlation, **Iteration control parameters** (selected), LCAO basis set, Numerical accuracy parameters, Poisson solver

Iteration Control

Tolerance	4e-05
Maximum steps	200
Algorithm	PulayMixer
Damping factor	0.1
History steps	10
Linear threshold	0
Mixing variable	HamiltonianVariable
Preconditioner Type	Off

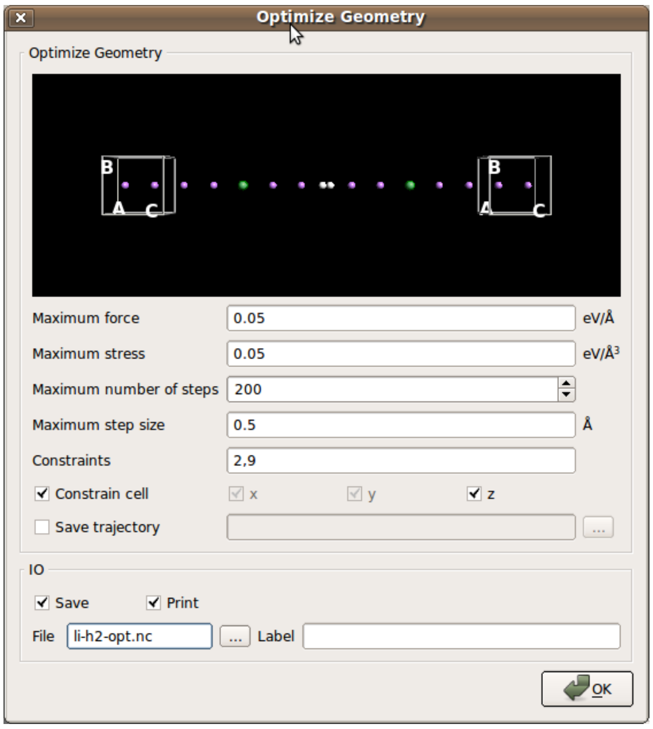
✓ No SCF iteration

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).


Optimize Geometry

Optimize Geometry

Maximum force	0.05	eV/Å
Maximum stress	0.05	eV/Å³
Maximum number of steps	200	
Maximum step size	0.5	Å
Constraints	2,9	

[✓] Constrain cell [✓] x [✓] y [✓] z

[ ] Save trajectory

IO

[✓] Save [✓] Print

File li-h2-opt.nc Label

Note

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

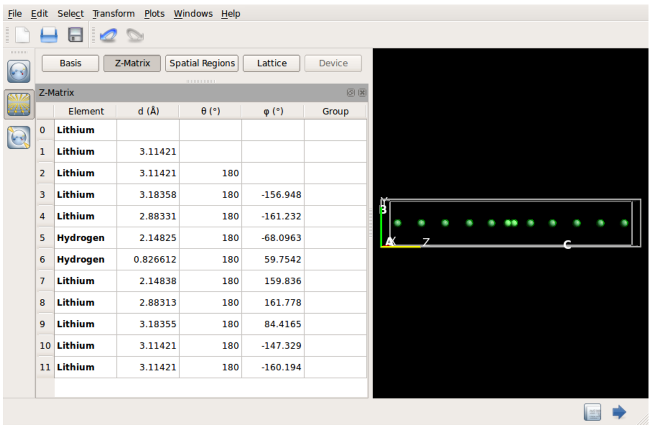
Next execute the job - this will take some time, typically 5-10 minutes.

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 hide the Basis and Lattice widgets.
- Select all atoms.

You should now see the following



Z-Matrix

	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 eV/Å3.
- Set the stress tolerance to 0.001 eV/Å3.

Note

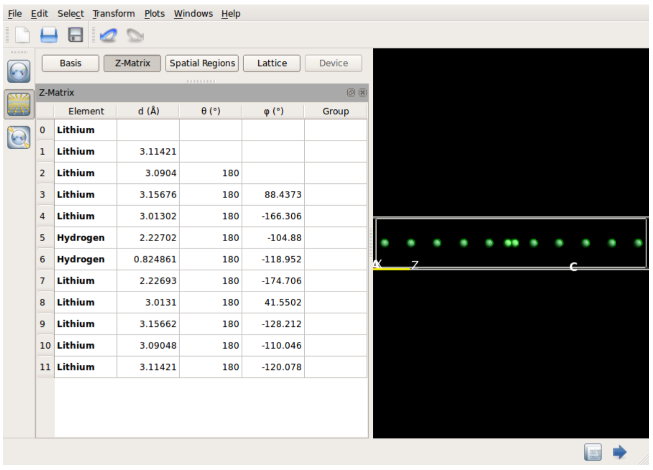
The cell size in the x and y directions cannot be optimized, since they are fixed by the electrode cell size.

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 Generator** 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)

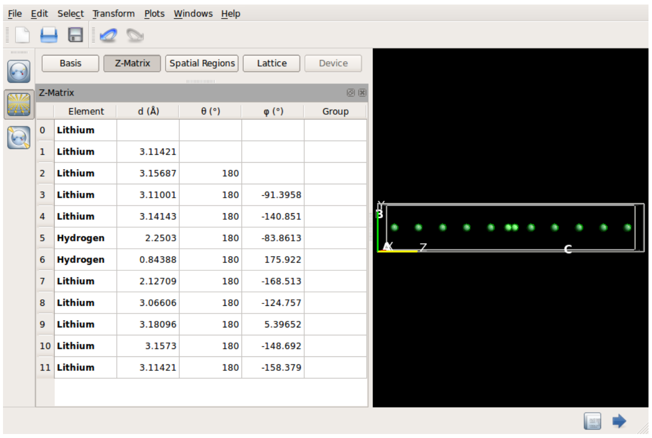
- Remove the tick from constraining the cell in the z-direction.
- Set the force tolerance to 0.01 eV/Å3.
- Set the stress tolerance to 0.001 eV/Å3.

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

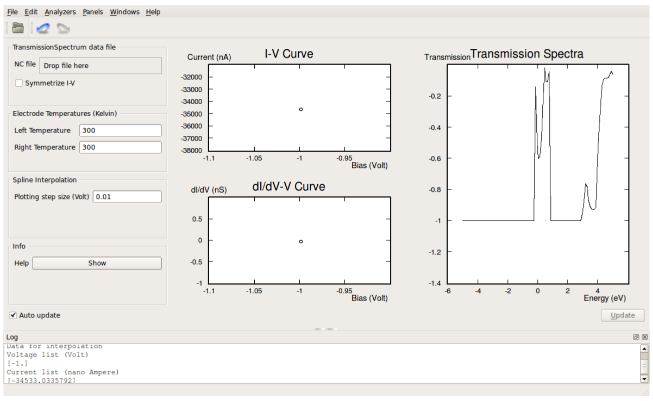


	Element	d (Å)	θ (°)	φ (°)	Group
0	Lithium				
1	Lithium	3.11421			
2	Lithium	3.15687	180		
3	Lithium	3.11001	180	-91.3958	
4	Lithium	3.14143	180	-140.851	
5	Hydrogen	2.2503	180	-83.8613	
6	Hydrogen	0.84388	180	175.922	
7	Lithium	2.12709	180	-168.513	
8	Lithium	3.06606	180	-124.757	
9	Lithium	3.18096	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.



```
Data for interpolation
Voltage list (Volt)
[-1.]
Current list (nano Ampere)
[-34533.0335792]
```

Comparing with the non-optimized geometry, it is found that the optimization has increased the current by 10 percent.