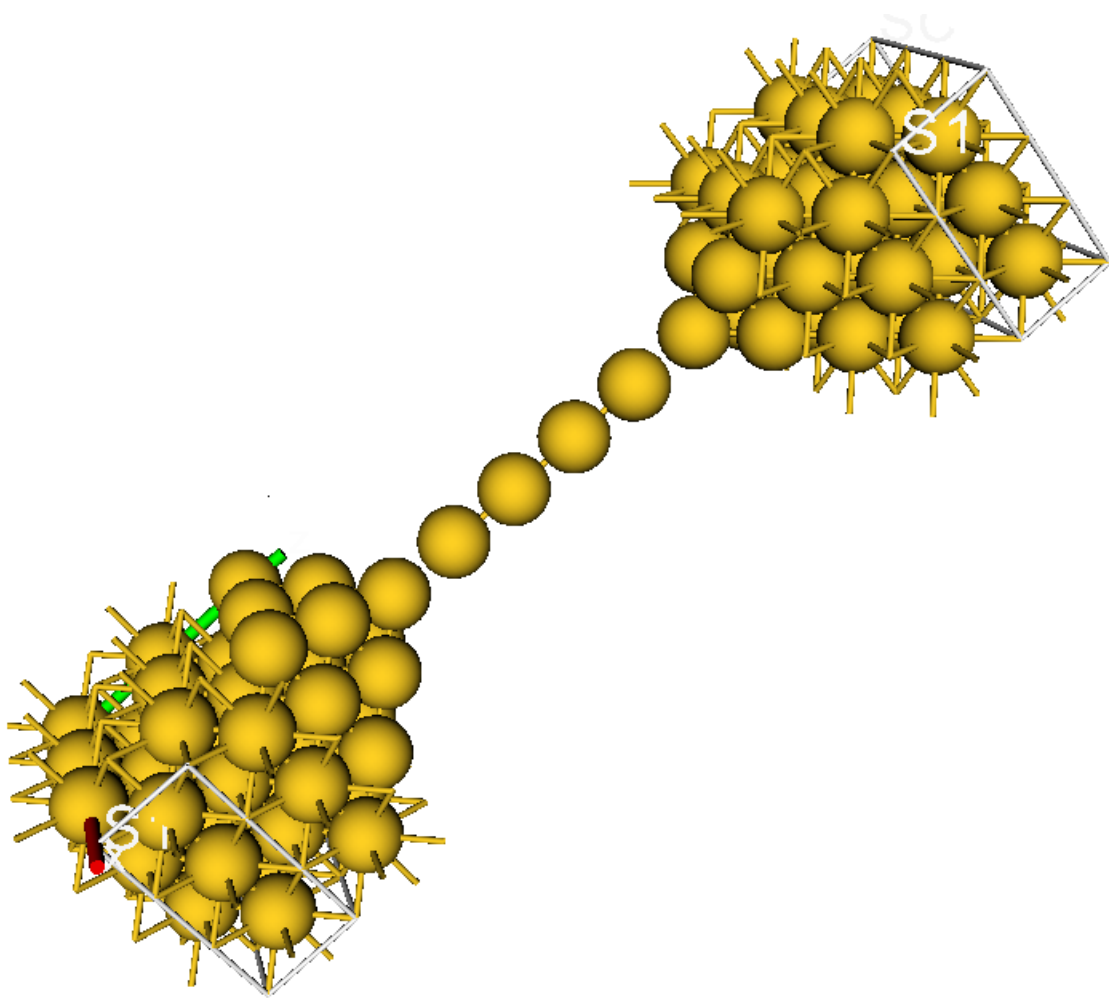


ATOMISTIX TOOLKIT & VIRTUAL NANOLAB

ADVANCED TUTORIAL



Transport Properties
of Gold Monowires

Contents

Introduction	3
Prerequisites.....	3
Building a 4-atom wire	4
Step 1: The gold surfaces.....	4
Step 2: The pyramidal tips.....	6
Step 3: Building the monowire.....	7
Step 4: Finalizing the geometry	8
Choosing the convergence parameters	11
Computing the zero-bias conductance.....	12
Band structure of an isolated atomic wire	15
Setting up N-atom chains	18
References	20

Introduction

This tutorial will demonstrate how to use with Atomistix ToolKit to compute the zero-bias conductance of a gold monowire between gold [100] surfaces. The inspiration for this tutorial is Ref. 1. A picture of the system is shown on the front page of this tutorial.

We will cover the basic steps of setting up the geometry, defining the parameters for the self-consistent calculation, and finally computing the conductance from the transmission spectrum. We will use Virtual NanoLab for as much of the work as possible, although we will resort to NanoLanguage scripting to automate the setup, to make it easy to change the number of atoms in the wire, and for certain analysis tasks.

Prerequisites

Naturally a working installation of **Virtual NanoLab** (version 2008.10 is recommended, but all should work equally well with 2008.02) will be required, including a valid license for the software. A free demo license can be requested on our web site quantumwise.com!

The systems involved in this tutorial are relatively large (albeit not very large), plus that gold contains quite many valence electrons. It is therefore strongly recommended to use a 64-bit computer to run the calculations, ideally in parallel on a cluster with 4–8 (or more) nodes. In this case one can reduce the calculations to a few hours each, while on a 32-bit laptop or workstation they might run for 10–12 hours or more.

It will be assumed that the reader is familiar with the basic operations in the software corresponding to the level of experience gained from completing the **Virtual NanoLab** tutorial. In particular, we will not in detail demonstrate how to set the parameters for the calculation. We will, however carefully specify which parameters to choose in order to obtain convergence, as this is quite sensitive in this system.

No particular proficiency in programming or **NanoLanguage** will be necessary, although we will use some simple scripts to execute certain tasks more efficiently.

Building a 4-atom wire

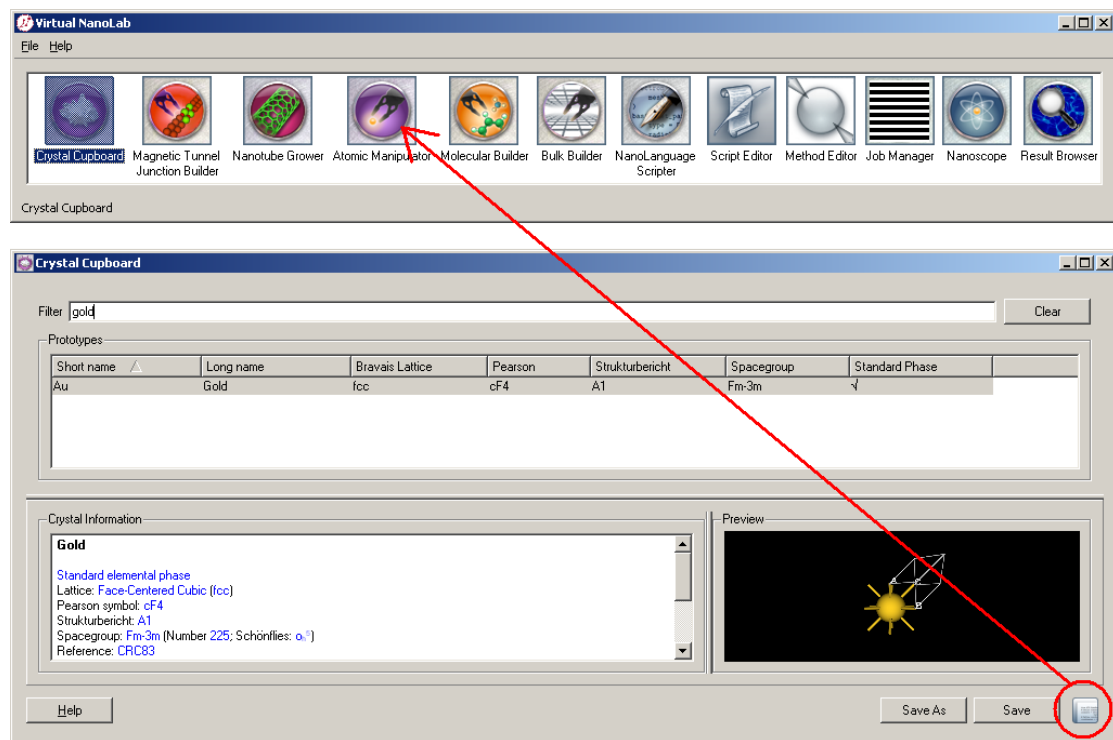
We will start by building the system with 4 atoms in the wire from scratch in VNL. Later on we will generalize the setup to allow for any number of atoms by using scripting.

Note: The wire will actually consist of 6 atoms, but the first and the last are considered to be part of the surface rather than the wire, in correspondence with Ref. 1.

The system contains 3 distinct parts: the electrodes, the pyramidal connections, and the wire in the middle. Our first step is to set up the Au [100] surfaces.

Step 1: The gold surfaces

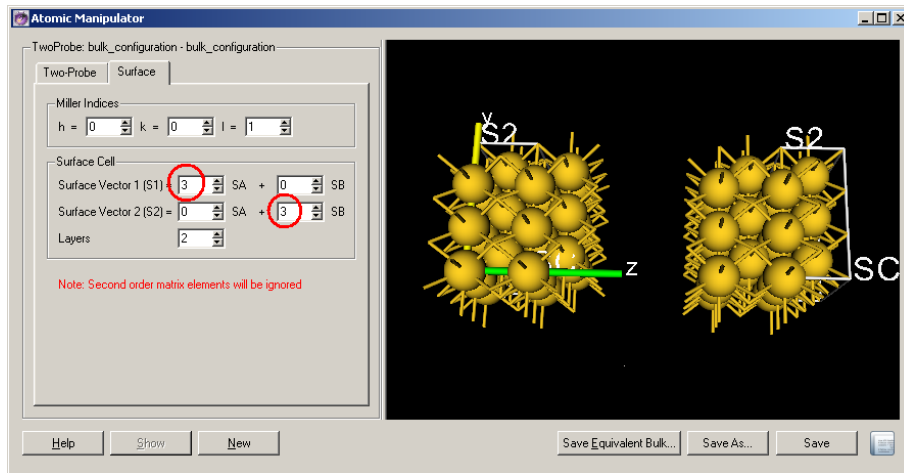
Launch **Virtual NanoLab** and double-click the **Crystal Cupboard** icon. Type “gold” in the “**Filter**” box; the search returns only one hit.



Drag the icon in the lower right corner of the **Crystal Cupboard** to the **Atomic Manipulator** icon in the main VNL window and drop it there.

Cleave the crystal by right-clicking the 3D preview and choose “**Cleave**” from the context menu. We will use the default [100] Miller indices.

Then set up a 3x3 surface cell (see the figure).

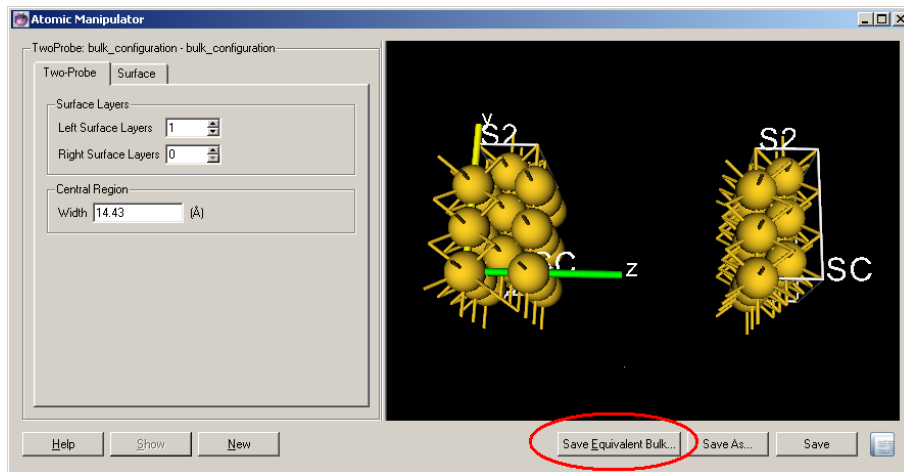


Switch to the **“Two-Probe”** tab and specify

- Left Surface Layers = 1
- Right Surface Layers = 0
- Central Region Width = 14.43 Å

The central region width is chosen as $5 \cdot r_0$ where r_0 is the Au–Au distance in the wire, which we take as 2.886 Å [1]. The number 5 is because we will insert 4 atoms in the wire.

You can also choose any other number of atoms to use for the wire (3, 5, 9, whatever), but after this step you are locked into that choice. This is why the scripting solution, to be discussed later, will be more flexible.



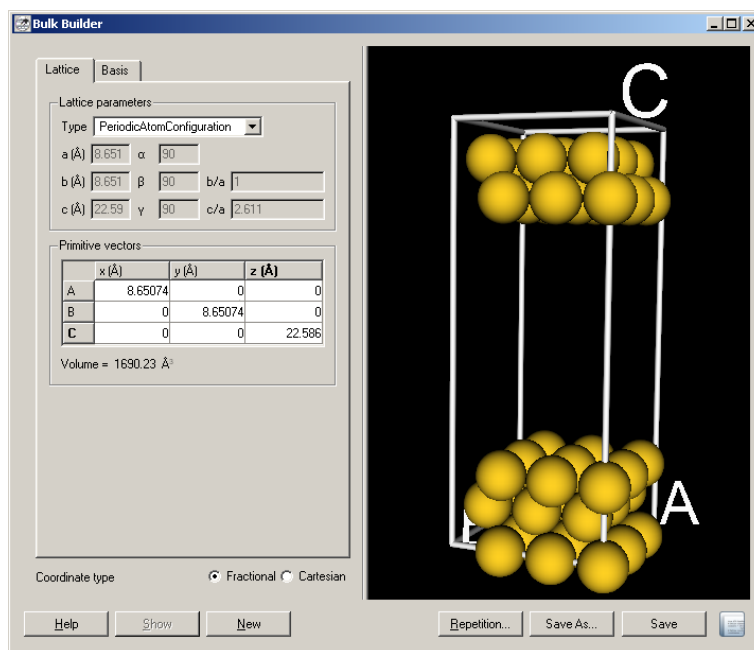
We will use the generated gold layers to create the pyramidal tips that connect the monowire to the surface. For this, we need to convert the system from a two-probe to a periodic configuration. This is achieved by clicking the button **“Save Equivalent Bulk”**. Save the system as a Python file in a convenient location.

Leave the **Atomic Manipulator** window open; we will return to it shortly.

Step 2: The pyramidal tips

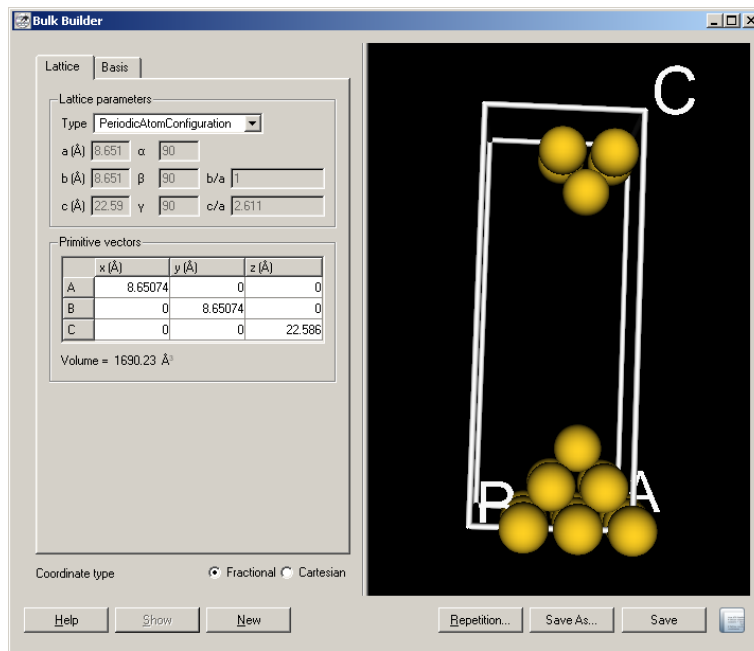
Drop the created equivalent bulk script on the **Bulk Builder** icon on the main VNL window.

To make it easier to select the relevant atoms, right-click the 3D preview window and bring up the **"Properties"** dialog. Select the **"Atoms"** plot (under **"Bulk Structure"**) and switch to **"User covalent radii"** for **"Atom sizes"**. Click **OK**.



Remove the relevant atoms to make the pyramidal tip, by selecting atom by atom and pressing the **Delete** button on the keyboard. You should delete 8 atoms in the two innermost surface layers, and 5 atoms in the layer below that, in both surfaces. Make sure that the tips are exactly opposite to each other!

The result should look like the figure below; the two tip atoms should have X and Y coordinates = 2.88358 Å.



Now drop the system back onto the open **Atomic Manipulator** window. Nothing appears to happen, but that is because the system is hidden under the electrodes; we will soon make it appear in its proper position!

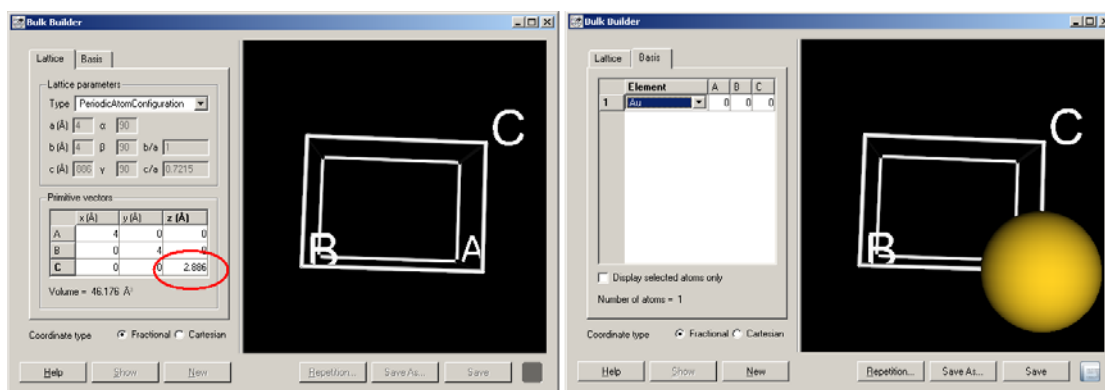
Before we do that, however, let us return to the **Bulk Builder** to build the gold monowire.

Step 3: Building the monowire

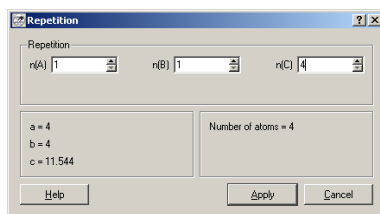
In the Bulk Builder, click **New**. Then select **"PeriodicAtomConfiguration"** from the drop-down list **"Type"**.

Set the Au–Au distance (2.886 Å) as the Z component of the C vector.

Switch to the **"Basis"** tab and insert an atom by right-clicking the white box to the left, and choose **"Insert atom"** from the context menu. Change the element to **Au**.



Finally, click the button “**Repetition**” and apply a repetition of 4 times along the C direction. If you chose another number of atoms for the wire earlier, when setting the distance between the tips, adjust the repetition factor accordingly here!



This completes the setup of the wire. Drop it on the open **Atomic Manipulator** window to add it to the two-probe system (again, it is hidden under the electrodes at first).

Step 4: Finalizing the geometry

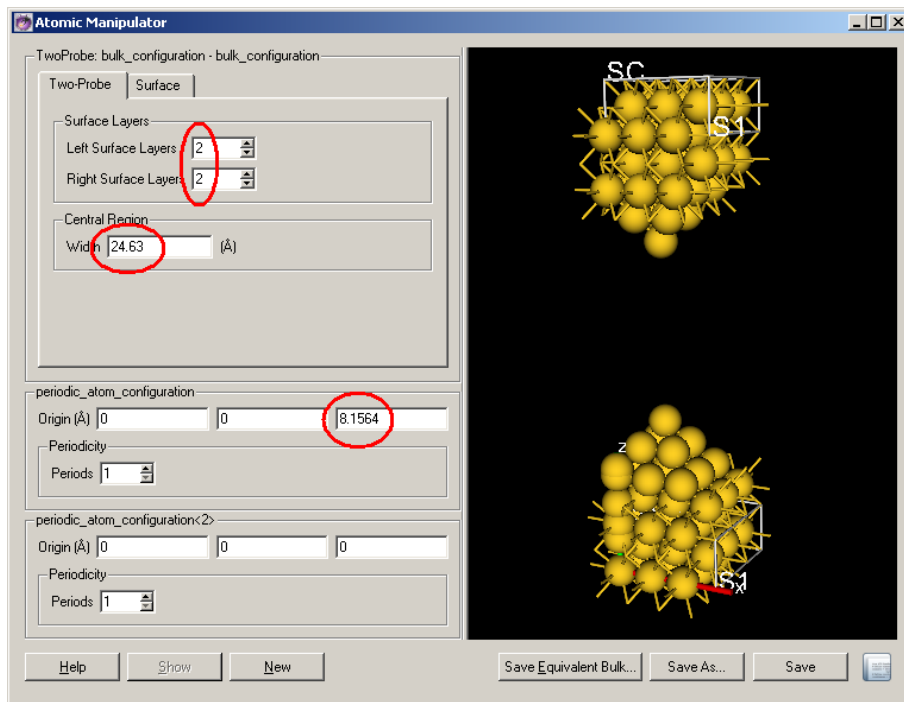
The final step in the geometry setup is to position the tips and the wire properly in the central region.

First we must add some surface layers to match the stacking sequence. Therefore, enter 2 for both left and right surface layers (or any other even number).

Then, we must increase the central region width by 5 layer separations; the layers are half a lattice constant (which is 4.0782 \AA) apart on the $[100]$ surface, thus the new central region width becomes $14.43 + 5 * 2.0391 \text{ \AA} = 24.6255 \text{ \AA}$.

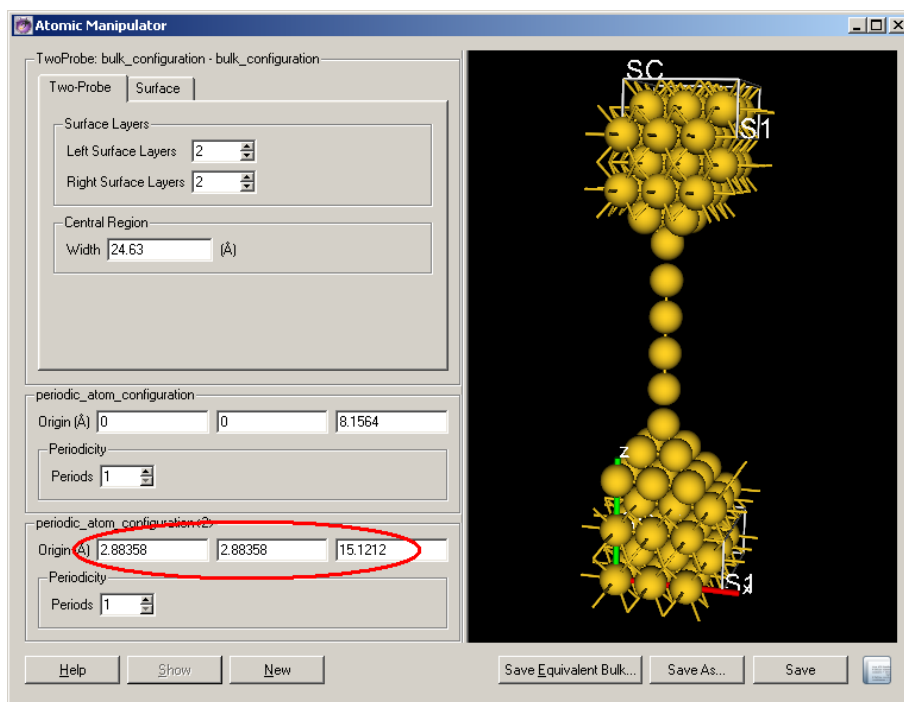
The tips need to be shifted in the Z direction. The leftmost layer of the left tip should be equivalent to the 5th surface layer of the left electrode, that is its Z position should be $4 * 2.0391 \text{ \AA} = 8.1564 \text{ \AA}$.

If you chose a different number of surface layers, you must adjust the two numbers above (and some below) accordingly.



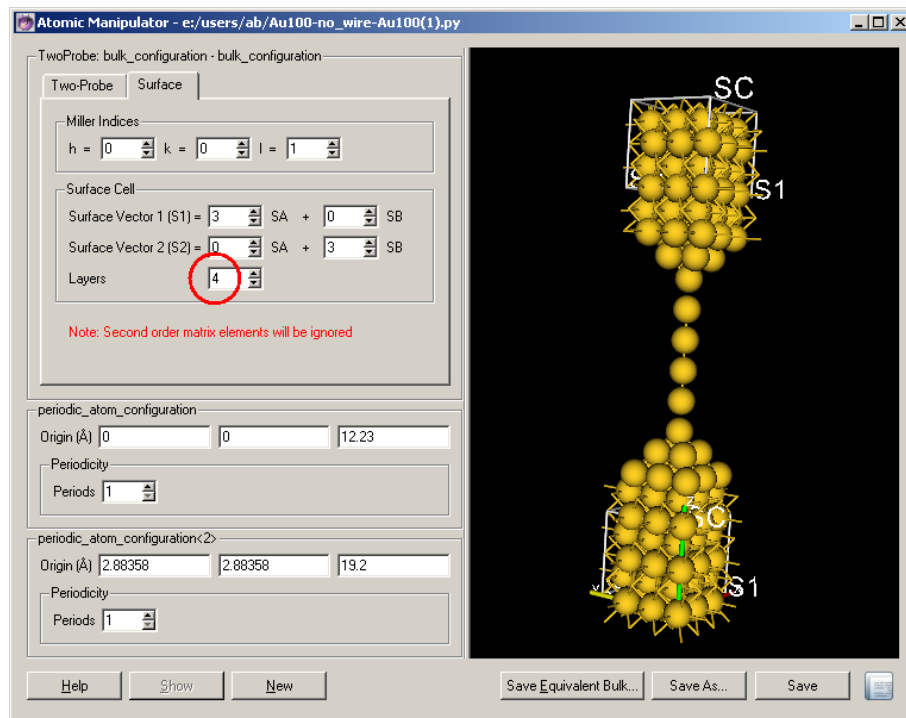
The wire should also be shifted. Remembering that the tip atoms have $X = Y = 2.88358 \text{ \AA}$, we use these values for the origin position in X and Y for the wire.

In the Z direction, the leftmost wire atom should be 2.8868 \AA to the right of the left pyramid tip atom, that is at $12.2344 + 2.8868 \text{ \AA} = 15.1212 \text{ \AA}$.



The absolutely final step is to increase the number of layers in the electrode. The default suggested by VNL, 2 layers, is too short and this may cause problems for the

convergence. We therefore return to the **“Surface”** tab and increase the number of layers to 4.



Note how this does not influence the other parts of the setup, the central region is just shifted to match the new electrodes.

This completes the geometry setup! Save the geometry as a Python file (**“Save”** button).

Choosing the convergence parameters

We now proceed to compute the conductance.

Drop the system from the **Atomic Manipulator** on the **NanoLanguage Scripter**. For the most part we will use default parameters, except:

- Basis set parameters: **Type = SingleZetaPolarized** (saves time and memory, without compromising the accuracy substantially for gold)
- Brillouin zone integration parameters: **Number of k-points = 5 x 5 x 100**
- Electron density parameters: **Mesh cut-off = 200 Ry**
- Eigenstate occupation parameters: **Electron temperature = 1000 K**
- Iteration mixing parameters: **Diagonal mixing parameter = 0.05**

These settings in part determine the level of desired accuracy, but more importantly they are carefully chosen to obtain convergence readily.

Alternatively, and especially attractive if you run this calculation on a **single** machine, not in a parallel, you can

- increase the electron temperature to **2000 K**, and
- reduce the k-point sampling to **2x2x100**.

It is generally the case that the lower the temperature, the more k-points are needed. The results for 2000 K will be slightly less accurate, however.

The second set of parameters will save a large amount of time, but does not really compromise the convergence. For example, with these parameters the calculation takes the same time on 2 nodes as the “full” calculation does on 4 nodes. The only difference in the transmission spectrum is that the higher temperature one is, as expected, more “averaged” and has less sharp features.

Make sure to specify a checkpoint file name on the **“Self-Consistent Calculation”** tab. Since we will not run this calculation via the Job Manager, it is better *not* to specify any path, but only the file name. In this way, the checkpoint file will be created in the same directory where the script is run from.

Save the script!

As mentioned, running the calculation is demanding, and should really be done on a cluster or 64-bit Linux machine. On a cluster with 2–4 nodes, it takes about 2–3 hours to converge the self-consistent calculation.

Computing the zero-bias conductance

Once the self-consistent loop has converged, it is time to determine the conductance of the monowire. At zero bias, the conductance is directly given by the transmission at the Fermi level, so this is what we need to compute.

In a system like this, the transmission coefficients depend on the \mathbf{k} -vector in the 2D Brillouin zone in the plane perpendicular to the transport direction. This is not the case for 1D systems like nanotubes, but in our case it means that we need to apply k-point sampling also in the evaluation of the transmission spectrum. We do not know, however, *a priori* how many k-points that are necessary, so we will need to perform a series of calculations to determine this. This is why we did not include the transmission spectrum calculation as part of the script above.

Our approach will therefore be to restore the converged calculation from the checkpoint, and compute the Fermi level transmission for an increasing number of k-points, and see when the result converges. This task can be accomplished with a very simple script, so there is no need to involve VNL; we would anyway have to modify the script by hand (or make 10 individual scripts).

Therefore, open the **Script Editor** or an external editor of your choice, and create a script with the following lines (the script is also provided with this tutorial, `converge_numk_transmission.py`, so you can just drag that file onto the editor):

```
from ATK.TwoProbe import *
from ATK.MPI import processIsMaster

checkpoint_file = "Au100-Au4-Au100.nc"
scf = restoreSelfConsistentCalculation(checkpoint_file)

for N in range(1,10):

    kpoints = brillouinZoneIntegrationParameters((N, N))

    transmission_spectrum = calculateTransmissionSpectrum(
        self_consistent_calculation = scf,
        energies = (0.0,)*electronVolt,
        brillouin_zone_integration_parameters = kpoints,
    )
    if processIsMaster():
        print N, transmission_spectrum.coefficients()[0]
```

Change the name of the checkpoint file to match what you used for the self-consistent calculation.

Naturally it is highly desirable to run also this calculation on the cluster, but it is also manageable on a laptop or workstation; it takes about an hour to complete (but would take much shorter on the cluster!).

When you run it, the script will print the following output:

```
1 0.992829440749
2 0.959447462443
3 0.963655792865
4 0.965489047645
5 0.966639117579
6 0.967083845635
7 0.967385069609
8 0.96783419824
9 0.967611030166
```

It is thus sufficient to use 6x6 k-points to get an accurate value of the zero-bias conductance. We now also have our first data point, the conductance for the chain with 4 atoms.

It is also interesting to plot the actual transmission spectrum. Now we know how many points we need, let us simply reuse the script above for the task, and just add a few lines.

Let us compute the spectrum from -3 to 3 eV using 100 points (this script is also provided with the tutorial, `calculate_transmission_spectrum.py`):

```
from ATK.TwoProbe import *
from ATK.MPI import processIsMaster
import numpy

checkpoint_file = "Au100-Au4-Au100.nc"
scf = restoreSelfConsistentCalculation(checkpoint_file)

kpoints = brillouinZoneIntegrationParameters((6, 6))
energies = numpy.linspace(-3,3,100)*eV

transmission_spectrum = calculateTransmissionSpectrum(
    self_consistent_calculation = scf,
    energies = energies,
    brillouin_zone_integration_parameters = kpoints,
)

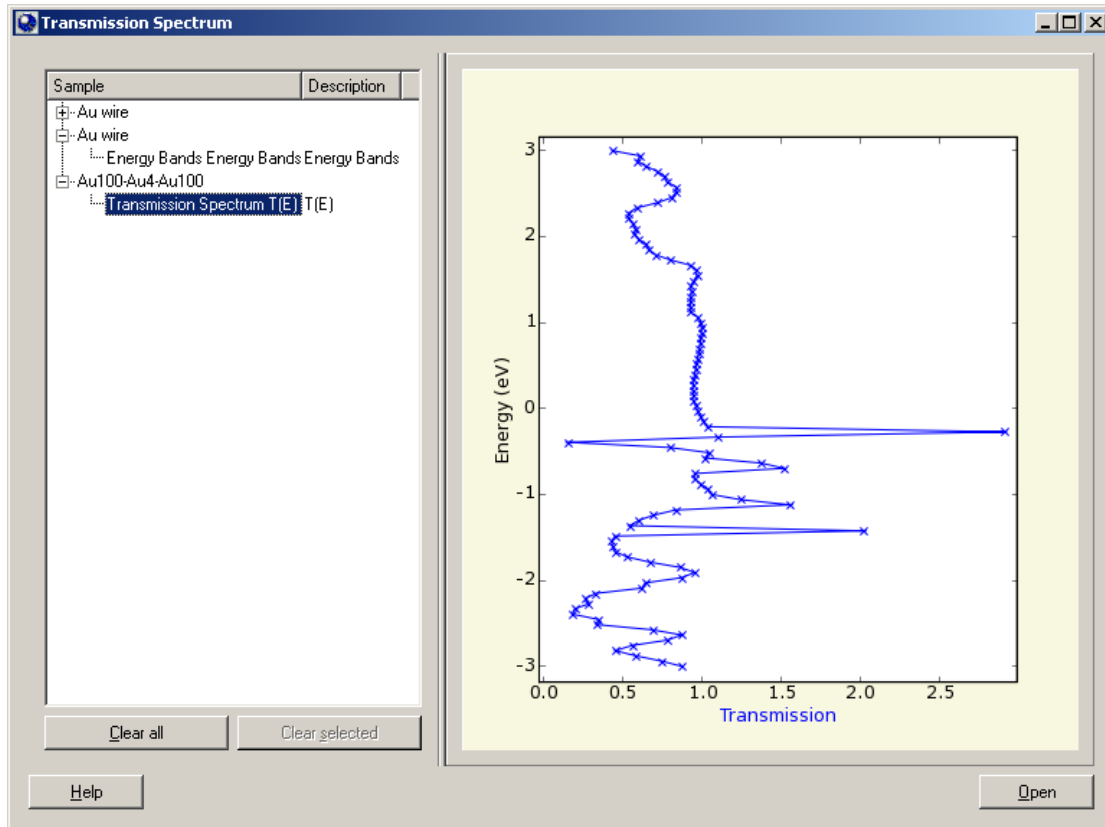
if processIsMaster():
    file = VNLFile("Au100-Au4-Au100_Transmission.vnl")
    file.addToSample(transmission_spectrum, "Au100-Au4-Au100", "T(E)")
```

This task could just as well have been set up using the [NanoLanguage Scripter](#).

This calculation is expensive, since there are so many energy points, and should really be run on in parallel too (it will scale linearly with the number of MPI nodes too, so use as many as you can afford).

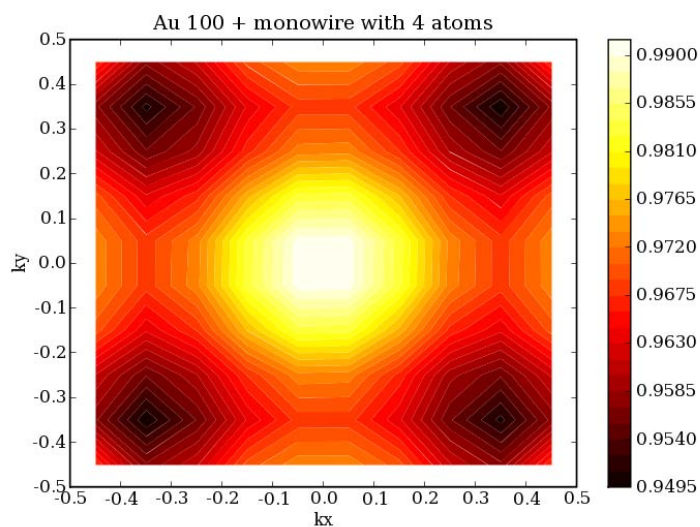
Once it finishes, import the VNL file to the [Result Browser](#) and open the transmission spectrum. There is a strong singularity in the spectrum about 0.3 eV below the Fermi level, and above the Fermi level the transmission is close to unity in a broad interval.

Later on we will compare the band structure of the gold monowire to the transmission.



Another interesting thing to plot is the k-point dependent transmission coefficients at the Fermi level. Again, a script (`plot_k_transmission.py`) to produce the plot is attached with this tutorial; it relies on an external module for generating a Monkhorst–Pack grid of k-points, and runs in reasonable time on a workstation/laptop (5–10 minutes for 10x10 k-points).

The result is plotted below, and shows almost unit transmission throughout the entire Brillouin zone; the dark areas have only slightly lower transmission than the central peak.



Band structure of an isolated atomic wire

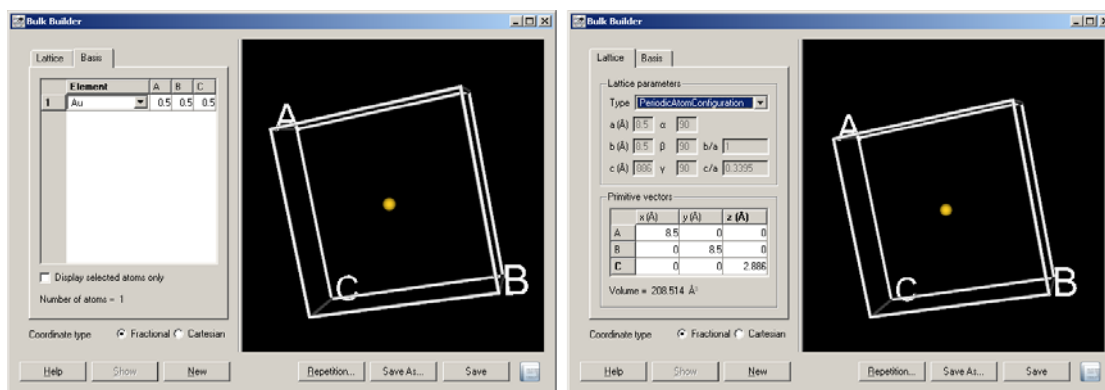
For comparison, and in order to explain some of the features in the transmission spectrum, we can also compute and plot the band structure of an isolated gold monowire.

This is quite straightforward. Open the **Bulk Builder**, and click **“New”** to be on the safe side.

Define a **“PeriodicAtomConfiguration”** from the drop-down **“Type”** under **“Lattice Parameters”**.

Set the unit cell components to 8.5 Å for A/B along X/Y, respectively, and the Au–Au distance 2.886 Å for C along Z. The value 8.5 Å matches closely the unit cell for the 3x3 surface used above. Using this it will also allow us to see if any degeneracies are split due to residual electrostatic interactions between neighboring cells since the structure is periodic in X and Y.

Switch to the **“Basis”** tab and insert an atom by right-clicking the white box to the left and choose **“Insert atom”**. Change the element to gold (Au) and set the position of the atom to the middle of the cell (for nicer plotting), i.e. place it at (0.5, 0.5, 0.5) in fractional coordinates.



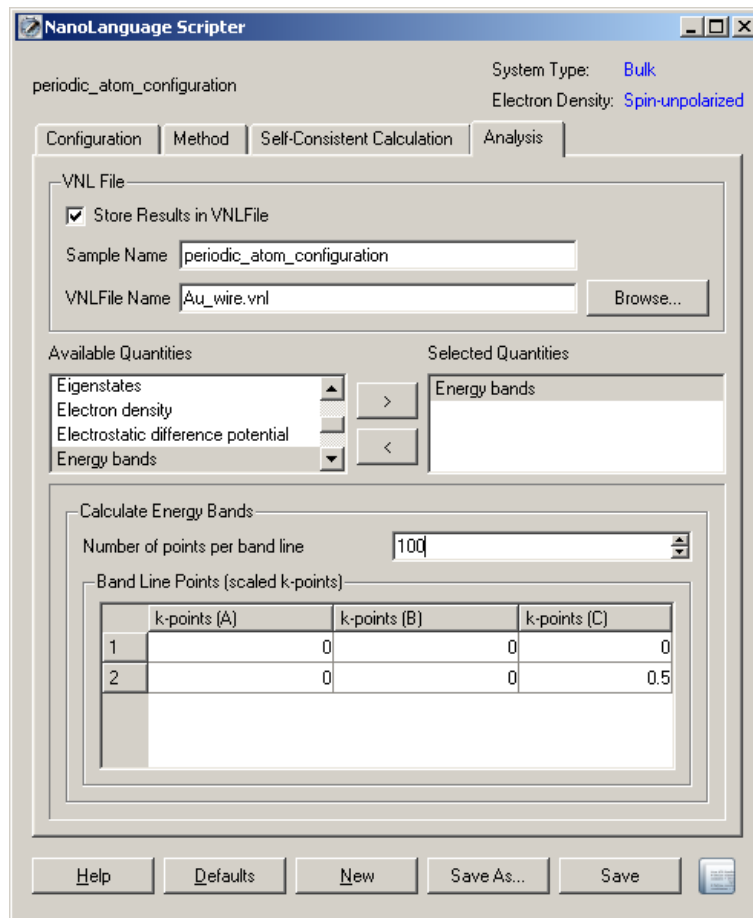
Save the system in a VNL file, say `Au_wire.vnl`, by clicking the **“Save As...”** button; don’t forget to change **“Save as type”** to **“VNL file”**!

Next, drop the structure on the **NanoLanguage Scripter**. We will use similar parameters as in the two-probe case to make it converge, that is:

- Basis set parameters: **Type = SingleZetaPolarized**
- Brillouin zone integration parameters: **Number of k-points = 1 x 1 x 100** (we don’t need any k-point sampling in the XY plane for a one-dimensional system)
- Electron density parameters: **Mesh cut-off = 200 Ry**
- Eigenstate occupation parameters: **Electron temperature = 1000 K**
- Iteration mixing parameters: **Diagonal mixing parameter = 0.05**

On the **Analysis** tab we enter the same VNL file (with full path; the best option is to use the **Browse** button to locate it!) as the file where we saved the geometry. Do not change

the sample name. Finally, add **“Energy Bands”** to the selected analysis quantities. The k-points will already be correctly set up, but increase the number of points to 100 for smoother plotting.

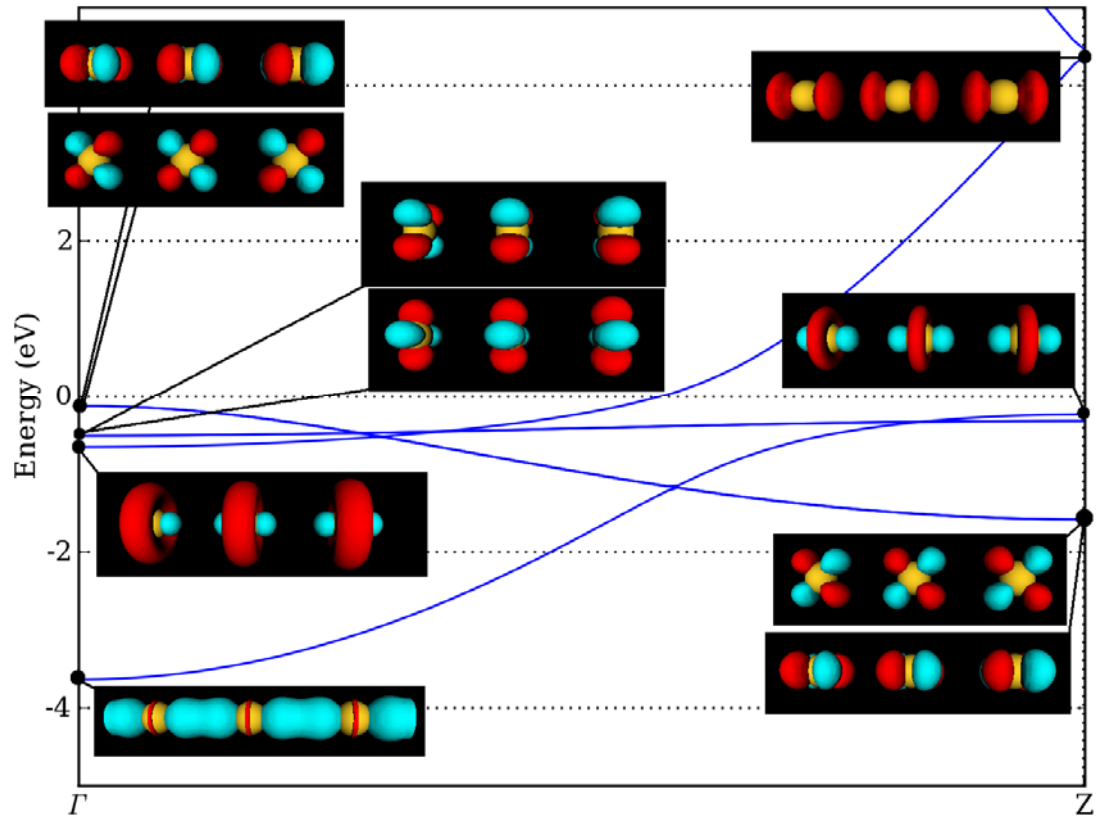


Save the script and then run it by dropping it on the **Job Manager**.

Once the calculation finishes, import the VNL file into the **Result Browser** to visualize the band structure. In the figure below we show a slightly different plot of the same data, and we have also overlaid some Bloch functions for the valence bands at the points Γ ($k_z = 0$) and Z ($k_z = \pi/a$, where a the Au–Au distance).

The wire is metallic, and we clearly see the almost flat band just below the Fermi level, which is responsible for the large jump at -0.3 eV in the transmission spectrum.

Band structure



Setting up N-atom chains

If we now would like to reproduce the results of Ref. 1 further, we would have to back to scratch, essentially, and go through all the steps of the geometry setup for 5, 6, 7, ... atoms in the wire. Although this is not a major task, it would be much more efficient to use a NanoLanguage script – especially if we wish to extend the study to involve other metals as well, and if we decide to change some parameter (the lattice constant would be different for other metals). With a script, we could just generate all geometries by changing a few parameters, and even just loop over the number of atoms.

To write such a script is a relatively complex task. Therefore, it is provided with this tutorial (`fcc100-wire.py`), and we will not go through how it is constructed, only how to use it.

At the top of the script you will find a number of lines defining the system. The user only needs to modify these to create any fcc 100 surface with a pyramidal contact to a monowire. The parameters are:

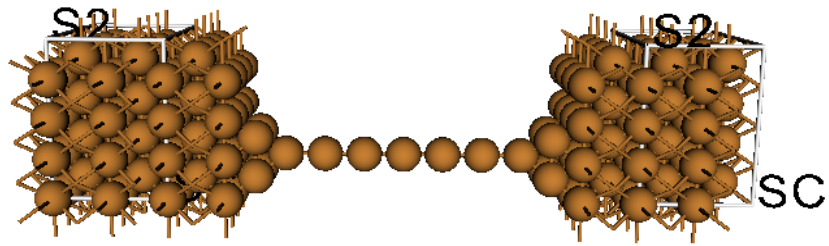
- **element** *The chemical element (e.g. Gold) of the surface and the monowire.*
- **lattice_constant** *The lattice constant of the bulk crystal; should be given as a PhysicalQuantity, e.g. $4.078 \cdot \text{Ang}$.*
- **electrode_layers** [4] *The number of electrode layers. Must be even, but 2 is not recommended (may be hard to converge).*
- **left_surface_layers** [3] *The number of left surface layers. Must be odd and > 2 to match the stacking sequence.*
- **right_surface_layers** [2] *The number of right surface layers. Must be even and > 1 to match the stacking sequence.*
- **Nsurface** [(3,3)] *A tuple containing the surface repetition numbers. Both must be > 1 .*
- **Nchain** [4] *The number of atoms in the monowire. Must be > 0 .*

NOTE: The script uses a difference definition of the number of atoms in the wire compared to the above hand-made setup and Ref. 1, in order to allow a single atom in the wire (a point contact). To set up the 4-atom wire as defined above, you therefore need to specify $N_{\text{chain}}=6$ instead.

The system is set up such that the distance between the atoms in the monowire corresponds to the nearest-neighbor distance in the bulk crystal, i.e. the lattice constant divided by $\sqrt{2}$. The layer separation along [100] is half the lattice constant.

Once you have defined all parameters as desired, simply drop the script on the [Nanoscope](#) for a preview, and then on the [NanoLanguage Scripter](#) to add the numerical parameters. That part can of course also be scripted, but that is beyond the scope of this tutorial.

An example, with a Cu fcc [100] 4x4 surface with 7 wire atoms, is shown below.



References

1. Y. J. Lee, M. Brandbyge, M. J. Puska, J. Taylor, K. Stokbro, and R. M. Nieminen, "Electron transport through monovalent atomic wires", *Physical Review B* **69**, 125409 (2004).