

# Spin transport in Magnetic Tunnel Junctions

## Tutorial on spin transport in Fe-MgO-Fe

---

Version 12.2.0

---

# **Spin transport in Magnetic Tunnel Junctions: Tutorial on spin transport in Fe-MgO-Fe**

Version 12.2.0

Copyright © 2008–2012 QuantumWise A/S

Atomistix ToolKit Copyright Notice

All rights reserved.

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

## TABLE OF CONTENTS

---

1. Introduction .....	1
2. Optimizing the junction geometry .....	2
Setting up the initial geometry .....	2
Setting up the initial geometry .....	3
Setting up the optimization .....	4
3. Properties of Fe-MgO-Fe with parallel spin .....	8
Relaxed device geometry .....	8
Defining and running the calculation .....	8
Analyzing the results .....	10
4. Properties of Fe-MgO-Fe with anti-parallel spin .....	15
Setting up the anti-parallel spin calculation .....	15
Analyzing the results .....	16
Bibliography .....	20

# CHAPTER 1. INTRODUCTION

---

This tutorial present how to setup, simulate and analyze a spin polarized system with ATK. The example used is a Fe-MgO-Fe magnetic tunnel junction (MTJ). This is very complex spin polarized system, initially studied in Ref. [1]. The tutorial illustrates how to build the system and setup calculations with parallel and anti-parallel spin configurations.

## CHAPTER 2. OPTIMIZING THE JUNCTION GEOMETRY

### SETTING UP THE INITIAL GEOMETRY

To set up the geometry of the Fe-MgO-Fe magnetic tunnel junction you will use a plugin in the Builder which is designed specifically for constructing this type of geometry. The first step is to download and install the plugin. To do this, start VNL and from the **Help** menu open the **AddOn Manager**.

In the AddOn Manager, select the **MTJBuilder** plugin and click **Install**. Before installing it you can see that the "Installed" version is empty, whereas the "Latest" version on the addon server is 1.0 (or some higher number).



#### Tip

Almost all functionality in the Builder is based on plugins, and if necessary, updates to these will be published via the addon server; this eliminates the need to release an entire new ATK package just to fix a small problem in a plugin. It's therefore a good idea to periodically check in the AddOn Manager if there are any updates to download. You don't have to go through the whole list, just click "Update All" to make sure you have all the latest updates.

The downloaded plugin will be installed in `vn\lib/site-packages/MTJBuilder` in the ATK installation directory (`vn\lib/python2.7/site-packages/MTJBuilder` on Linux). It is provided with complete source code, which you can inspect to get inspiration on how to write similar plugins yourself. There are many kinds of plugins in VNL - this particular one is a type which lets you add a new system in the Builder. In principle such plugins can set up any kind of geometry, but commonly they will build template systems, based on a few parameters - such as the nanotube and graphene nanoribbon builders in VNL.

---

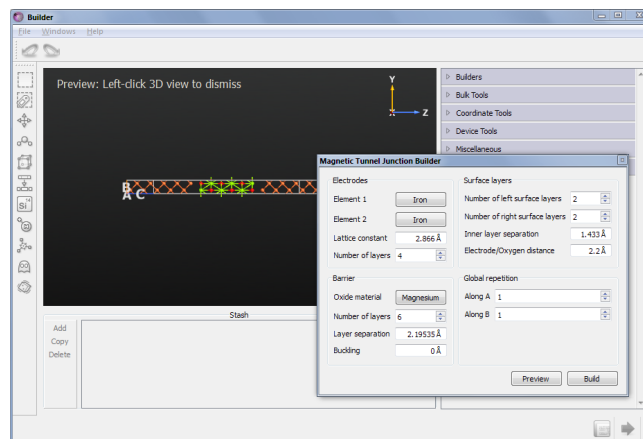
## SETTING UP THE INITIAL GEOMETRY

---

Open the **Builder** by clicking its icon  on the main VNL toolbar.

Click the **Add** button to the left of the Stash, and open the newly installed plugin "Magnetic Tunnel Junction (FeMgO-style)" from the Add Custom menu.

The MTJ builder can build a range of different magnetic tunnel junction geometries. For the present purpose, change the number of barrier layers to 6, and the number of surface layers to 2 on both the left and right side, but leave the other parameters as they are by default. Press the **Preview** button to see the geometry, then press **Build** to add it to the Stash.



**i Tip**


Hold the mouse still over each input field to get a description of the input parameters.

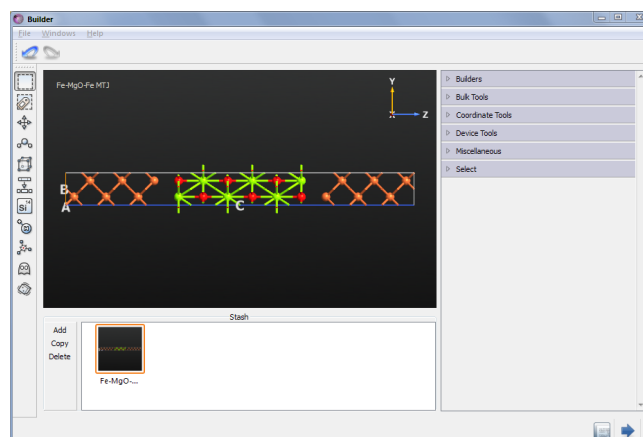
---

## CHANGING THE DEVICE GEOMETRY TO EQUIVALENT BULK

---

In order to speed up the relaxation you will perform the relaxation of the DeviceConfiguration as an equivalent bulk system. More specifically, we will optimize the central region, while keeping the electrode extensions fixed.


Convert the device geometry to bulk by clicking the Bulk tool  on the left-hand side toolbar.



---

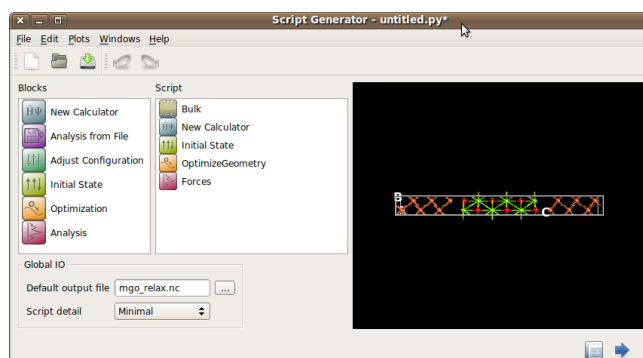
## SETTING UP THE OPTIMIZATION

---

The next step is to relax the geometry of the equivalent bulk geometry. Send the structure, using the Send To icon , to the [Script Generator](#).

In the Script Generator:

1. Change the **Default output file** to `mgo_relax.nc`; use the browse button "..." to specify the precise location of the file, if you intend to run the script from within VNL, otherwise (i.e. if you will run the script from the command line, for instance in parallel on a cluster) just write the filename;
2. Double-click **New Calculator** to insert a calculator block in the script;
3. Double-click **Initial State**, so that we can define the initial spin configuration;
4. Double-click **Optimization** and select OptimizeGeometry to request a geometry optimization.
5. Double-click **Analysis** and select **Forces**.



---

## ADJUSTING THE SCRIPT COMPONENTS

---

To set up the calculational parameters, double-click the "New Calculator" block.

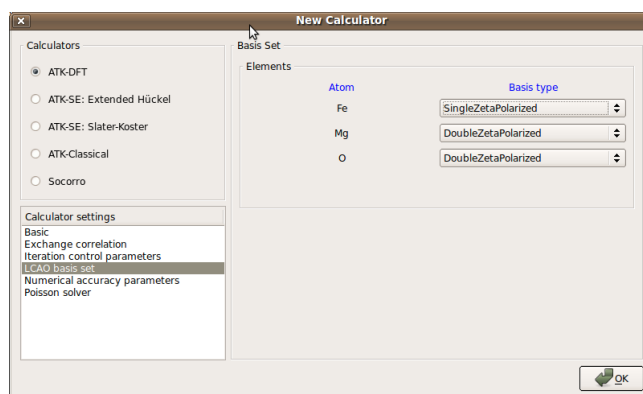
- Set the **Exchange correlation** to **SGGA** to perform a spin polarized GGA calculation with the PBE functional.
- Set the **Electron temperature** to 1200 Kelvin, to obtain faster self-consistent convergence.
- Set the k-point sampling to 6,6,2.



### Note

This choice gives a reasonable k-point sampling in the x and y directions. In the z direction you would like to model a device configuration, thus the Fe atoms at the edges must be bulk-like. Selecting 2 k-points in this direction improves the bulk like description of the Fe edge atoms.

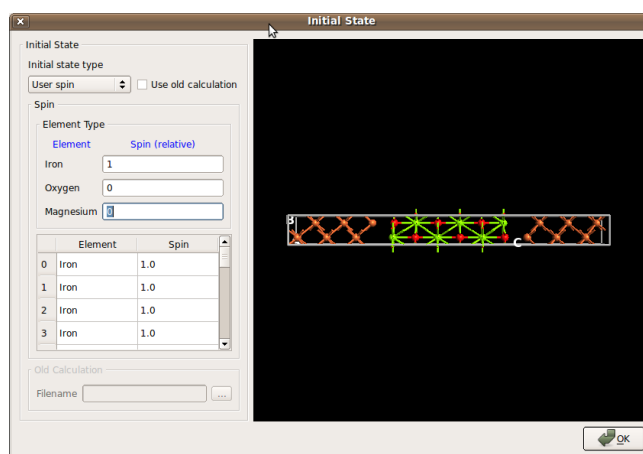
- Under **LCAO basis set**, select the **SingleZetaPolarized** basis set for Fe.



Click the OK button to close the widget.

Next open the **Initial State** block and

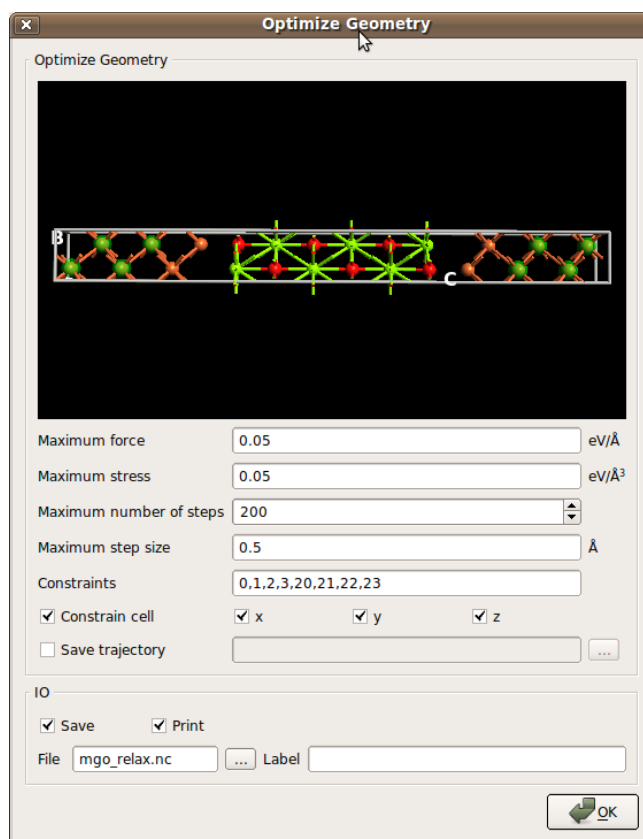
- Set **Initial State** type to "User spin".
- Set relative spin for Oxygen and Magnesium to 0.




Click the OK button to close the widget.

Next open the **Optimization** block, in order to specify that the first and last four atoms, corresponding to the electrode atoms, should be kept fixed during the relaxation. This is very important - if these atoms are moved, you no longer have a periodic structure at the edges of the central cell to convert back into electrodes.

Select the atoms indicated in the picture below by holding down the Ctrl key and drawing a rectangle around first those on the left, then those on the right.



Click the OK button to close the widget.

You do not need to modify the **Forces** block, thus click the “Save file” icon , and save the script into the file `mgo_relax.py`.

## RUNNING THE OPTIMIZATION CALCULATION

You are now ready to perform the relaxation. This calculation will take a 1-2 hours on a PC, but can be sped up significantly when executed on a parallel computer.

To run it in serial on your computer, just drop the script on the Job Manager, or type

```
atkpython mgo_relax.py > mgo_relax.log
```

on the command line.

For information on how to run the script on a parallel computer, see our guide on [how to perform parallel calculations with ATK](#).

## INSPECTING THE RESULTS

To inspect the forces of the relaxed geometry, select the file `mg_relax.nc` in the result browser of the VNL main window. **Left**-click the Forces object and press the Show button in action widget. You will now see the forces on all atoms


Atom	F <sub>x</sub> (eV/Ang)	F <sub>y</sub> (eV/Ang)	F <sub>z</sub> (eV/Ang)	Unit
0	9.13775672e-13	1.00306043e-12	-8.58080998e-02	eV/Ang
1	2.09379046e-12	2.29398267e-12	2.49580507e-02	eV/Ang
2	8.13308278e-13	1.10507659e-12	-9.86754520e-02	eV/Ang
3	2.91447862e-12	2.88742001e-12	-1.25889442e-01	eV/Ang
4	1.76637943e-12	1.70235384e-12	3.70597741e-02	eV/Ang
5	3.59134848e-12	2.12721142e-12	4.30367104e-02	eV/Ang
6	7.65804204e-14	4.82960410e-13	1.99610706e-02	eV/Ang
7	7.72252167e-13	5.80177782e-13	-2.95766528e-02	eV/Ang
8	6.90040132e-13	3.25920330e-14	3.46622793e-02	eV/Ang
9	7.04359810e-13	8.60068279e-13	-2.25856336e-02	eV/Ang
10	4.31007749e-13	1.66055048e-13	9.28244806e-04	eV/Ang
11	5.37797894e-13	6.41311611e-13	6.22117802e-03	eV/Ang
12	7.44029046e-13	2.54656652e-13	-6.25927480e-03	eV/Ang
13	2.69196904e-13	7.21698129e-13	-8.92753052e-04	eV/Ang
14	-1.03298650e-13	2.85549293e-13	2.26131559e-02	eV/Ang
15	4.01046946e-13	3.92778721e-13	-3.46933157e-02	eV/Ang
16	4.40836552e-13	1.28400812e-12	2.95754998e-02	eV/Ang
17	9.80336669e-13	5.30054111e-14	-1.99555485e-02	eV/Ang
18	1.45310210e-12	2.35084362e-12	-4.30635927e-02	eV/Ang
19	2.81911961e-12	1.65460751e-12	-3.70374150e-02	eV/Ang
20	1.63336666e-12	1.78746093e-12	1.25889702e-01	eV/Ang
21	2.98014568e-12	2.44649889e-12	9.86766209e-02	eV/Ang
22	1.34034982e-12	1.05499762e-12	-2.49556203e-02	eV/Ang
23	2.08714313e-12	2.36245572e-12	8.58102453e-02	eV/Ang

Note that all forces in the x-y direction are for all practical purposes zero. For the atoms which have been relaxed, the forces are less than the optimization criterion of 0.05 eV/Ang. For the electrode atoms the forces are slightly larger. The forces are pointing out of the cell, indicating that the cell is under compressive strain. To lower this force you can add surface layers to increase the cell length in the z-direction, however the forces are rather small and this will have a rather minimal effect on the results. You will now use the optimized structure for [calculations of transport properties of the device](#).

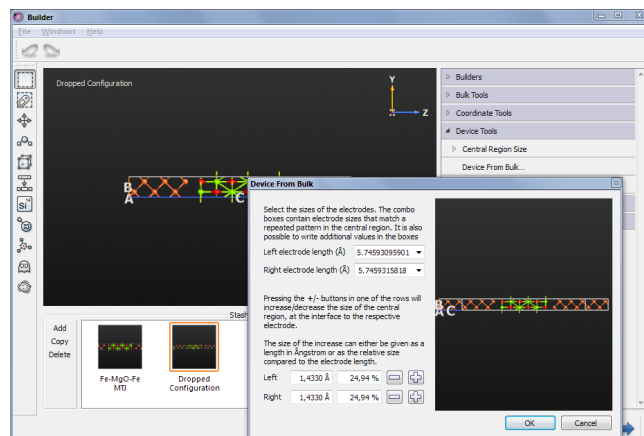
# CHAPTER 3. PROPERTIES OF Fe-MgO-Fe WITH PARALLEL SPIN

In this chapter you will transform the relaxed geometry back into a device configuration, perform a self-consistent calculation for the case of parallel spin polarization of the electrodes. You will also calculate the transmission spectrum as function of the k-vector parallel to the transport direction.


## RELAXED DEVICE GEOMETRY

In the VNL Result Browser, select the second "Bulk configuration" (glD001) in the result file `ngo_relax.nc` and drag and drop it onto the Builder icon .

The relaxed structure represents the central region of the desired device configuration. In order to transform the structure into a device configuration, click **Device Tools** in the right-hand side plugin panel, and click **Device From Bulk...** Click OK to accept the suggest electrode length, which is the same as the original structure built with the MTJ Builder plugin had, and which corresponds to the four atoms on each side that were constrained in the optimization.



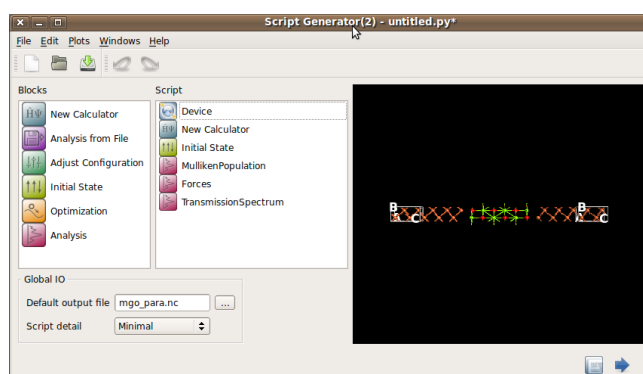
## DEFINING AND RUNNING THE CALCULATION

Send the structure to the **Script Generator** using the "Send To" icon .

In the Script Generator

1. change the **Default output file** to `mgo_para.nc`
2. double-click **New Calculator** to insert a calculator block in the script;
3. double-click **Initial State**, so that you can define the initial spin configuration;
4. double-click **Analysis** and select **MullikenPopulation**.
5. double-click **Analysis** and select **Forces**.
6. double-click **Analysis** and select **TransmissionSpectrum**.

The Script Generator should now have the following settings:



## ADJUSTING THE SCRIPT COMPONENTS

Double-click the "New Calculator" block.

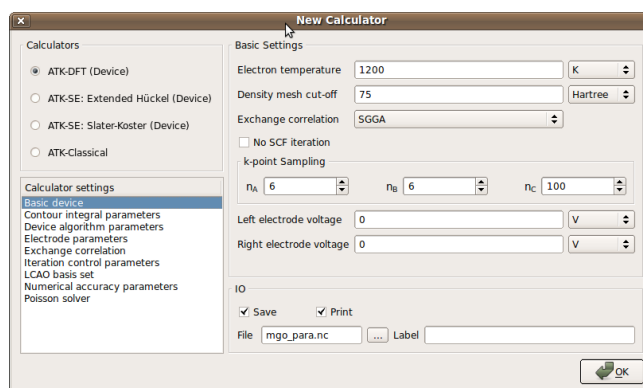
- Set the **Exchange correlation** to **SGGA** to perform a spin polarized GGA calculation with the PBE functional.
- Set the **Electron temperature** to 1200 Kelvin, to improve the convergence.
- Set the **k-point** sampling to (6,6,100).



### Note

For a device, the k-points in the C direction are only used for the electrode calculation.

- Under **LCAO basis set**, select the **SingleZetaPoLarized** basis set for Fe.



Click the OK button to close the widget.

---

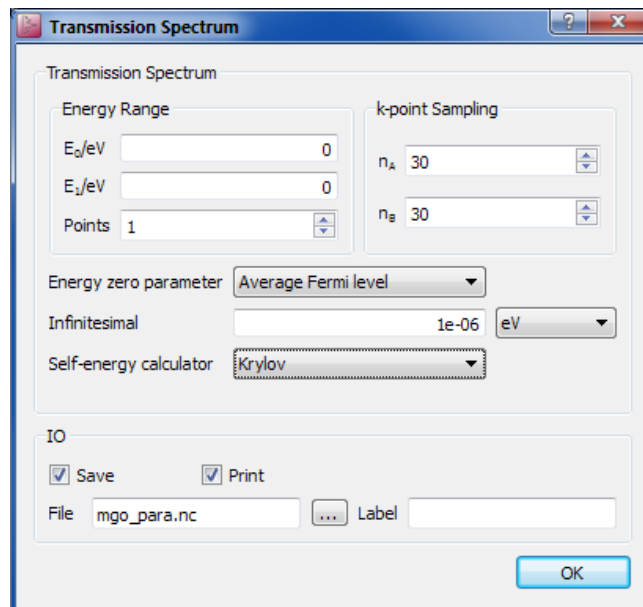
Next open the **Initial State** block and

- select **User spin**,
- set relative spin for Oxygen and Magnesium to 0.

The Forces and MullikenPopulation blocks do not take any parameters.

You will only compute the transmission at the Fermi level in this example, to save time. Therefore, open the TransmissionSpectrum block, and set

- $E_0=E_1=0$  eV and Points=1,
- k-point grid sampling = 30x30 points.
- Self-energy calculator: Krylov



Finally save the script as `mgo_para.py` and run the calculation. The calculation will take a similar time to run as the optimization performed in the previous section, and it is possible to obtain a very good speedup on a parallel computer.

### ! Important

Do not close the Script Generator! You will use it again to set up the anti-parallel calculation in the [next chapter](#).

---

## ANALYZING THE RESULTS

---

### INSPECTING THE FORCES

To inspect the forces, locate the result file `mgo_para.nc` in the vnl browser and **left-click** the file, and highlight the forces object. With a **left-click** on the show button in the Forces action window you obtain the following report:

```

Forces (gID001 in mgo_para.nc)
-----
Forces Report
-----
 0 [ 0. 0. 0. ] eV/Ang
 1 [ 0. 0. 0. ] eV/Ang
 2 [ 0. 0. 0. ] eV/Ang
 3 [ 0. 0. 0. ] eV/Ang
 4 [ 1.84841188e-12 8.11283163e-13 4.20385031e-02 ] eV/Ang
 5 [ 1.79378350e-12 2.39293007e-12 4.29159339e-02 ] eV/Ang
 6 [ 9.46832835e-13 6.85226340e-13 2.45975069e-02 ] eV/Ang
 7 [ -1.77007233e-13 -1.60601331e-13 -2.81300259e-02 ] eV/Ang
 8 [ -4.11346922e-13 -4.91587460e-13 3.46885388e-02 ] eV/Ang
 9 [ 1.52918389e-12 1.61810967e-12 -2.23460854e-02 ] eV/Ang
10 [ 9.02786551e-13 7.50269337e-13 1.01990754e-03 ] eV/Ang
11 [ -4.87939207e-14 -2.46984928e-13 6.36074225e-03 ] eV/Ang
12 [ -6.26428628e-13 -3.80360747e-13 -6.44637691e-03 ] eV/Ang
13 [ 1.70889921e-12 1.57761022e-12 -9.35823621e-04 ] eV/Ang
14 [ 7.58335159e-13 9.39637615e-13 2.24019631e-02 ] eV/Ang
15 [ -3.50954417e-13 -2.57099948e-13 -3.47575190e-02 ] eV/Ang
16 [ -3.13295127e-13 -4.21179237e-13 2.80435336e-02 ] eV/Ang
17 [ 1.32870986e-12 1.31836337e-12 -2.46759454e-02 ] eV/Ang
18 [ 1.60628467e-12 1.04334782e-12 -4.29353422e-02 ] eV/Ang
19 [ 1.14640366e-12 2.24281446e-12 -4.19700166e-02 ] eV/Ang
20 [ 0. 0. 0. ] eV/Ang
21 [ 0. 0. 0. ] eV/Ang
22 [ 0. 0. 0. ] eV/Ang
23 [ 0. 0. 0. ] eV/Ang

```

The forces on the equivalent electrodes atoms inside the central cell are not calculated, and these are therefore set to zero. For the other atoms, the forces are small, and it can be concluded that relaxing the geometry for the equivalent bulk structure is a good strategy for this system.

## INSPECTING THE MULLIKEN POPULATION

Similarly, you can inspect the Mulliken population and obtain the following report:

```

MullikenPopulation (gID001 in mgo_para.nc)
-----
 0.470 0.470
 0.500 0.500
 0.296 0.099 0.098 0.099
 0.387 0.127 0.132 0.127
-----
 4 Fe 5.175 4.417 0.866 0.850 0.930 0.850 0.921
      2.844 1.952 0.489 0.497 0.231 0.497 0.239
      s
      0.462 0.462
      0.506 0.506
      y z x
      0.296 0.098 0.101 0.098
      0.386 0.127 0.132 0.127
-----
 5 Fe 5.528 4.672 0.914 0.939 0.949 0.930 0.931
      2.480 1.633 0.386 0.350 0.241 0.350 0.307
      s
      0.488 0.488
      0.482 0.482
      y z x
      0.368 0.135 0.099 0.135
      0.365 0.130 0.104 0.130
-----
 6 Mg 0.470 -0.039 -0.039
      0.533 -0.048 -0.048
      s
      0.200 0.200
      0.228 0.228
      y z x
      0.310 0.092 0.127 0.092
      0.353 0.092 0.168 0.092
-----
 7 O 3.501 0.077 0.077
      3.466 0.080 0.080
      y z x
      0.498 0.167 0.165 0.167
      0.507 0.167 0.173 0.167
      s
      0.815 0.815
      0.809 0.809
      y z x

```

In the leftmost column is shown the total Mulliken population per spin channel for each atom. It is important to inspect this result in order to check that each atom is correctly spin polarized. Note that all Fe atoms are polarized with their majority spin pointing up and the O atoms are unpolarized. The Mg atoms next to the interfaces (on both sides) couple anti-parallel with the Fe atoms, while the Mg atoms away from the interfaces are unpolarized.

The report also shows the population for each shell and resolves it on individual orbitals.

---

## ANALYZING THE K-PARALLEL TRANSMISSION

---

To plot the k-dependent transmission, use the following script

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

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

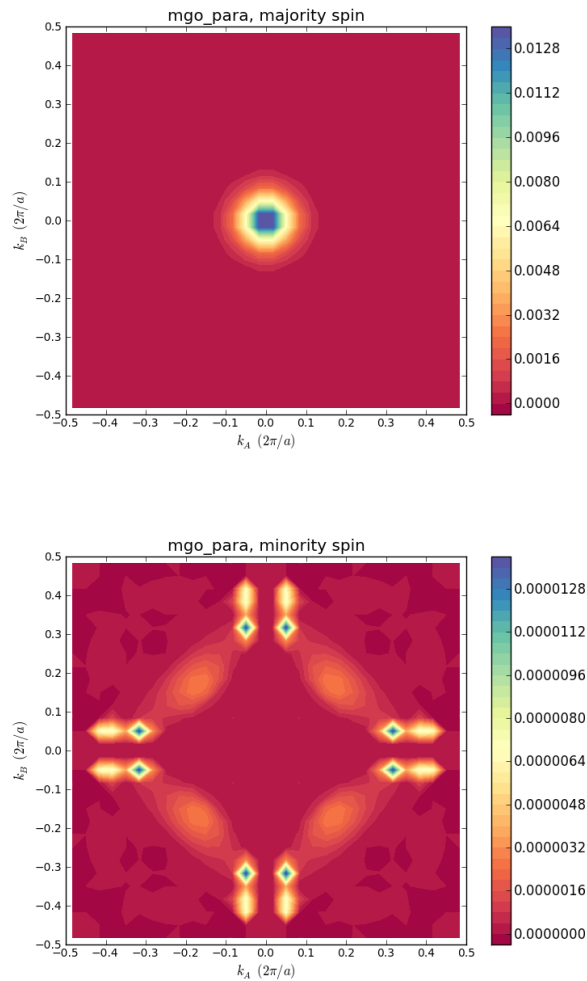
import pylab
#plot the transmission majority spin
pylab.figure()
pylab.xlabel("$k_A \ (2\pi/a)$",fontsize=12, family='sans-serif')
pylab.ylabel("$k_B \ (2\pi/a)$",fontsize=12, family='sans-serif')
pylab.contourf(K_A,K_B,T_uu,40, cmap=pylab.cm.Spectral)
pylab.colorbar()
pylab.axis([- .5, .5, - .5, .5])
pylab.yticks(numpy.arange(-0.5,0.51,0.1),fontsize=10)
pylab.xticks(numpy.arange(-0.5,0.51,0.1),fontsize=10)
pylab.title('mgo_para, majority spin')
pylab.savefig('mgo_para_majority.png',dpi=100)

#plot the transmission minority spin
pylab.figure()
pylab.xlabel("$k_A \ (2\pi/a)$",fontsize=12, family='sans-serif')
pylab.ylabel("$k_B \ (2\pi/a)$",fontsize=12, family='sans-serif')
pylab.contourf(K_A,K_B,T_dd,40, cmap=pylab.cm.Spectral)
pylab.colorbar()
pylab.axis([- .5, .5, - .5, .5])
pylab.yticks(numpy.arange(-0.5,0.51,0.1),fontsize=10)
pylab.xticks(numpy.arange(-0.5,0.51,0.1),fontsize=10)
pylab.title('mgo_para, minority spin')
pylab.savefig('mgo_para_minority.png',dpi=100)
pylab.show()
```

Save the script to you computer and execute it by dropping it onto the job manager. You should now see the following

### Tip

Instead of saving the script you can also highlight the script text in the document and drag it onto the job manager.



### EXPLAINING THE SCRIPT FOR PLOTTING THE K-DEPENDENT TRANSMISSION COEFFICIENTS

The script uses NanoLanguage components to load the k-dependent transmission spectrum from the file `mgo_para.nc` and then the `matplotlib` package for plotting. In the following some of the details in the script will be explained.

The first two lines in the script read a list with all Transmission spectrum objects from the file `mgo_para.nc` and assigns the first element in the list to the variable `transmission`.

```
#read the transmission spectrum
transmission = nload('mgo_para.nc',TransmissionSpectrum)[0]
```

The next line assigns the data in the transmission spectrum object into the variable `trans_coeff`. The data is a nested list with transmission coefficient per spin, energy and k-point.

```
#get the transmission spectrum
trans_coeff = transmission.transmission()
```

The two spin components are extracted from the data.

```
#select spin up component
T_uu = trans_coeff[0]
```

```
#select spin down component
T_dd = trans_coeff[1]
```

Each spin component has the shape energy points times k-point.

```
# shape of the components are [energy_points, k_points]
(n_e,n_k) = T_uu.shape
```

For plotting with matplotlib the array's must be reshaped into: number of k-points in the A direction, number of k-points in the B direction

```
# assume equal many k_points in A and B directions
n_A=n_B=numpy.sqrt(n_k)
#reshape the arrays for plotting, assume only one energy point
T_uu = T_uu.reshape(n_A,n_B)
T_dd = T_dd.reshape(n_A,n_B)
```

Similarly, the A and B components of the k-vector must be obtained

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



## Note

For the calculation of the transmission spectrum ATK only uses a fraction of the k-points, since the work can be reduced by symmetry. However, the object returns the full zone in order to ease analysis and plotting.

The rest of the script is concerned with using matplotlib for plotting, to understand the details of this part see the [matplotlib webpage](#).

As for the results of the calculation, it is clear from the plots that the minority transmission is strongly suppressed in magnitude, and exhibits spikes at certain points in the Brillouin zone corresponding to resonant tunneling, whereas the majority transmission indicates regular barrier tunneling. This is all in agreement with the accepted picture of the tunneling mechanism in this system [1].

# CHAPTER 4. PROPERTIES OF Fe-MgO-Fe WITH ANTI-PARALLEL SPIN

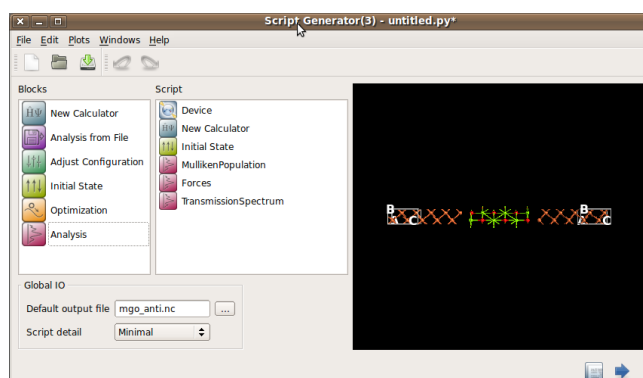
---

## SETTING UP THE ANTI-PARALLEL SPIN CALCULATION

---

You will now set up an anti-parallel spin calculation. The self-consistent state of the parallel configuration will be used as an initial state, to improve the convergence. To this end, return to the Script Generator where you set up the parallel calculation in the [previous chapter](#).

Change the **Default output file** to `mgo_anti.nc`.



## ADJUSTING THE SCRIPT COMPONENTS

---

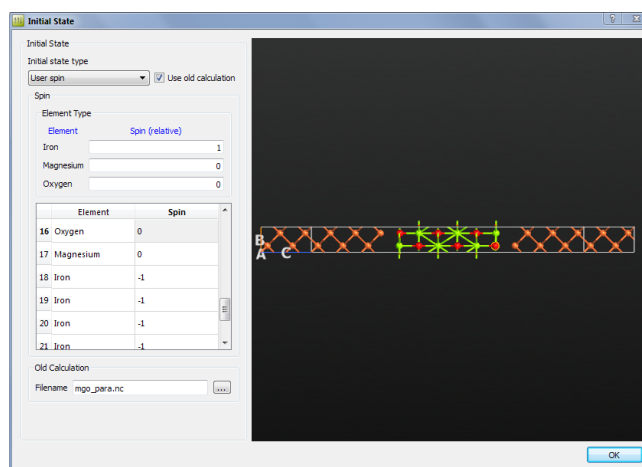
Next, double-click the **Initial State** block and modify the following parameters:

1. Activate **User spin**.
2. Check **Use old calculation**.
3. Click the "..." browse button and locate the file `mgo_para.nc` containing the converged calculation for the parallel spin configuration.

### **Note**

If you intend to run the script from the command line, just type in the filename `mgo_para.nc` without the path, and run the new script in the directory where this file resides.

4. Set the (scaled) initial spin on atoms 18 to 23 to **-1**. The Density Matrix entries on these atoms will now be flipped, and thereby give an anti-parallel initial density matrix.



Finally, modify the TransmissionSpectrum analysis object, to add more k-points to improve the resolution. Set the k-point sampling grid to 100x100 points.

Save the script as `mgo_anti.py` and run the calculation. The calculation should be quicker than the parallel spin calculation, since now you use the parallel spin calculation as a good initial guess for the anti-parallel calculation.

## ANALYZING THE RESULTS

Similarly to the previous section you can analyze the results in the data file `mgo_anti.nc`. Check that the Forces on each atom are still negligible and check the Mulliken population to ensure that each atom has a correct spin polarization.

Below is given a script for plotting the k-dependent transmission. It is a slightly modified version of the script from the previous section such that it now reads the data from `mgo_anti.nc`

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

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

import pylab
#plot the transmission majority spin
```

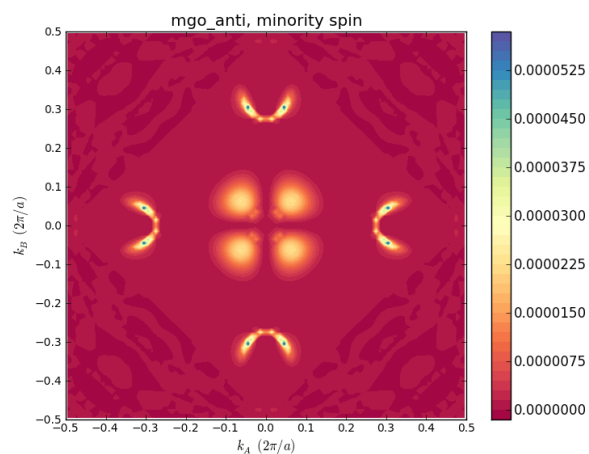
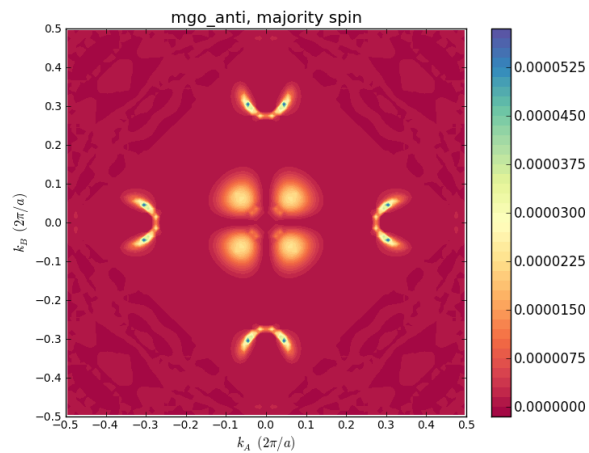
```

pylab.figure()
pylab.xlabel("$k_A \ (2\pi/a)$", fontsize=12, family='sans-serif')
pylab.ylabel("$k_B \ (2\pi/a)$", fontsize=12, family='sans-serif')
pylab.contourf(K_A, K_B, T_uu, 40, cmap=pylab.cm.Spectral)
pylab.colorbar()
pylab.axis([- .5, .5, - .5, .5])
pylab.yticks(numpy.arange(-0.5, 0.51, 0.1), fontsize=10)
pylab.xticks(numpy.arange(-0.5, 0.51, 0.1), fontsize=10)
pylab.title('mgo_anti, majority spin')
pylab.savefig('mgo_anti_majority.png', dpi=100)

#plot the transmission minority spin
pylab.figure()
pylab.xlabel("$k_A \ (2\pi/a)$", fontsize=12, family='sans-serif')
pylab.ylabel("$k_B \ (2\pi/a)$", fontsize=12, family='sans-serif')
pylab.contourf(K_A, K_B, T_dd, 40, cmap=pylab.cm.Spectral)
pylab.colorbar()
pylab.axis([- .5, .5, - .5, .5])
pylab.yticks(numpy.arange(-0.5, 0.51, 0.1), fontsize=10)
pylab.xticks(numpy.arange(-0.5, 0.51, 0.1), fontsize=10)
pylab.title('mgo_anti, minority spin')
pylab.savefig('mgo_anti_minority.png', dpi=100)
pylab.show()

```

Execute the script and you should obtain the plot below



Note that the two spin components in this case give the same transmission spectrum, due to the mirror symmetry of the system.

To understand this, first note that due to time inversion symmetry propagation from left to right is always identical to propagation from right to left. The up component of the transmission spectrum corresponds to up electrons from the left electrode propagation into the right electrode. The down component of the transmission spectrum corresponds to down electrons in the right electrode propagating into the left electrode.

Due to the anti-symmetric spin, the up electrons in the left electrode and the down electrons in the right electrode are both majority channels. Because of the mirror symmetry of the system, the propagation of the left majority channels into the right minority channels (the first transmission coefficient spin component) and the propagation of the right majority channels into the left minority channels (the second transmission coefficient spin component), are the same.

Thus, for symmetric structures the equivalence of the two spin channels is an important check for anti-symmetric calculations at zero bias.

## CALCULATING THE TUNNELING MAGNETO-RESISTANCE (TMR)

The tunneling magneto-resistance (TMR) is defined as

$$\text{TMR} = \frac{G_P - G_{AP}}{G_{AP}},$$

where  $G_P$  is the conductance in the parallel spin configuration and  $G_{AP}$  the conductance in the anti-parallel spin configuration.

The conductances can be calculated from their respective Transmission spectra. The script below performs this calculation

```
#calculate conductance for parallel spin
transmission_para = nload('mgo_para.nc',TransmissionSpectrum)[0]
conductance_para_uu = transmission_para.conductance(spin=Spin.Up)
conductance_para_dd = transmission_para.conductance(spin=Spin.Down)
conductance_para = conductance_para_uu + conductance_para_dd

#calculate conductance for anti-parallel spin
transmission_anti = nload('mgo_anti.nc',TransmissionSpectrum)[0]
conductance_anti_uu = transmission_anti.conductance(spin=Spin.Up)
conductance_anti_dd = transmission_anti.conductance(spin=Spin.Down)
conductance_anti = conductance_anti_uu + conductance_anti_dd

print 'Conductance Parallel Spin (Siemens)'
print 'Up=%8.2e, Down=%8.2e' %(conductance_para_uu.inUnitsOf(Siemens), conductance_para_dd.inUnitsOf(Siemens))
print 'Total = %8.2e' %( conductance_para.inUnitsOf(Siemens))
print

print 'Conductance Anti-Parallel Spin (Siemens)'
print 'Up=%8.2e, Down=%8.2e' %(conductance_anti_uu.inUnitsOf(Siemens), conductance_anti_dd.inUnitsOf(Siemens))
print 'Total = %8.2e' %( conductance_anti.inUnitsOf(Siemens))
print

print 'TMR (optimistic) = %8.2f percent' % \
(100.*(conductance_para-conductance_anti)/conductance_anti)
print 'TMR (pessimistic) = %8.2f percent' % \
(100.*(conductance_para-conductance_anti)/(conductance_para+conductance_anti))
```

Execute the script, and find the TMR

```
Conductance Parallel Spin (Siemens)
Up=6.97e-09, Down=2.04e-11
Total = 6.99e-09

Conductance Anti-Parallel Spin (Siemens)
```

---

Up=3.66e-11, Down=3.65e-11  
Total = 7.30e-11

TMR (optimistic) = 9469.88 percent  
TMR (pessimistic) = 97.93 percent

# BIBLIOGRAPHY

---

- [1] W.H. Butler, X.-G. Zhang, T.C. Schulthess, and J.M. MacLaren. *Phys. Rev. B*, **63**, 54416, 2001 ([link](#)).