Introduction to AFM Line Scan Simulator

Samuel Ng

1 Marcus Theory

Marcus theory models the rate of thermally assisted adiabatic electron transfers. The rate equation for an electron tunneling from an initial site to a final site is given by:

$$\nu_{if} = \frac{|t_{if}|^2}{\hbar}\sqrt{\frac{\pi}{\lambda k_B T}}\exp\left(-\frac{(\Delta G_{if}+\lambda)^2}{4\lambda k_B T}\right) \tag{1}$$

Where t_{if} is the tunneling term (overlap of wavefunctions); λ is the self-trapping energy; ΔG_{if} is the energy difference between the configurations before and after electron exchange.

2 AFM Line Scan Simulation

2.1 Implementation

A tool for simulating AFM line scans has been implemented. Marcus theory has been used to approximate electron tunneling rates, where ΔG_{if} is treated as the difference in Coulombic potential energy between the initial and final electron configurations. The simulator takes the position of dangling bonds, scan type (read or write) and other parameters to produce simulated line scan plots.

The simulation engine consists of two classes:

- AFMLine: This is the class that users interact with. It contains the user configuration for the AFM tip and dangling bonds, and facilitates the line scan simulation. To approximate an AFM's behavior, a measurement is made at each DB location to see whether it contains an electron. Plots are also generated by this class.
- MarcusModel: This class is called by *AFMLine* to generate the timing for electron tunneling events. Marcus theory is used to determine electron hopping rates between two sites, which is used to generate randomized timings for electron tunneling events from an exponential distribution. This class is also responsible for keeping track of electron count and position.

Currently, two items are still pending proper implementation: 1) the current model of the AFM tip is a result of trial-and-error parameterization, Lucian's model should be able to fill this gap; 2) the electron population count is determined by hand, an automated implementation is needed.

2.2 Usage

Example simulation setup options can be found at *afm-sim/src/python/demo.py*, executing it yields a line scan plot. The contents of *demo.py* have also been included below for your convenience:

```
#!/usr/bin/env python

from afm import AFMLine
```

```
4
5  # set the corrdinates of the dangling bonds in lattice units
6  #X = [1, 12, 15, 22, 25]          # 1-2-2
7  X = [1, 8, 10, 15, 17, 24]      # 1-2-2-1
8
9
10 afm = AFMLine(X)              # create AFM handler for the given DB arrangement
11
12 # set the scan type
13 afm.setScanType(0)            # read-read, set by default
14 #afm.setScanType(1, .01)       # write(right)-read(left) with a strength of 10meV
15 #afm.setScanType(-1, .01)      # write(left)-read(right) with a strength of 10meV
16
17 # set the bias
18 #afm.setBias(.00012)           # 120 ueV/angstrom bias gradient
19
20 # run the AFM and produce the AFM image, image will be saved to tempfig.pdf
21 # Nel:       number of electrons in the system, half filled by default
22 # nscans:    number of line scans, 2 per sweep with read-read else 1
23 # pad:       number of lattice sites included on the [left, right] edge of the scan
24 # srate:     tip speed in angstroms/second
25 afm.run(Nel=3, nscans=200, pad=[3,3], srate=82)
```

If you wish to alter the behavior of the simulation beyond what's shown above, you may tweak the 'magic numbers' in *afm-sim/src/python/marcus.py* within the section *# somewhat magic numbers*.