Memristor mathematical model, ver. 1.0

https://github.com/eugnsp/memristor

December 25, 2018

Equations, their discretization and algorithms

Note: in what follows atomic units are used.

Heat equation

The heat equation for the lattice temperature $T(\mathbf{x})$ is

$$-\nabla \cdot [c(\mathbf{x})\nabla T(\mathbf{x})] = s(\mathbf{x}), \tag{1}$$

where $c(\mathbf{x})$ is the thermal conductivity, and $s(\mathbf{x})$ is the volume heat density of the source.

In the cylindrical coordinates this equation for $T(\mathbf{x}) = T(r, z)$ takes the form:

$$-\frac{\partial}{\partial r}\left[c(r,z)\frac{\partial T}{\partial r}\right] - \frac{c(r,z)}{r}\frac{\partial T}{\partial r} - \frac{\partial}{\partial z}\left[c(r,z)\frac{\partial T}{\partial z}\right] = s(r,z). \tag{2}$$

The structure of the heat source term is

$$s(r,z) = \frac{\theta[r_0(z) - r]}{\pi r_0(z)^2} s(z), \tag{3}$$

where $r_0(z)$ is the source radius, and $s(z)$ is its linear density. It is tempting to approximate the source with a delta-functional one

$$s(r,z) = \delta(\pi r^2)s(z) = \frac{\delta(r)}{2\pi r}s(z). \tag{4}$$

However, the solution of (2) is singular at $r = 0$: $T(r) \sim \ln(r)$. Hence, finite r_0 should be retained. For simplicity we assume $r_0 := \{\mathsf{heat\_source\_radius}\}$ to be independent of z. Due to the logarithmic divergence the solution is very sensitive to the value of r_0 for $r < r_0$.

The heat is produced due to the Joule heating:

$$s(z) = I^2 r_c(z), \tag{5}$$

where I is the total current, and $r_c(z)$ is the linear resistivity of the core.

The thermal conductivity is assumed to be uniform in the whole system:

$$c(r,z) = c := \{\mathsf{thermal\_conductivity}\}. \tag{6}$$

Boundary conditions

The uniform Dirichlet boundary condition is assumed at the outer surface:

$$T(\mathbf{x})|_\Gamma = T_0 := \{\mathsf{temperature}\}. \tag{7}$$

This condition translates into the following condition in the cylindrical coordinates:

$$T(r,z)|_\Gamma = T_0. \tag{8}$$

along with the compatibility condition at $r = 0$

$$\left.\frac{\partial T(r,z)}{\partial r}\right|_{\Gamma_0} = 0, \tag{9}$$

Discretization

Multiplying eq. (2) by r, we obtain

$$-r\nabla\cdot(c\nabla T) - c\frac{\partial T}{\partial r} = \frac{I^2}{\pi r_0^2}\theta(r_0 - r)r_c, \quad \nabla = (\partial_r, \partial_z). \tag{10}$$

After multiplication by a test function $\chi(r,z)$ and integration by parts we get

$$-\int_\Omega r\chi\nabla\cdot(c\nabla T) - \int_\Omega \chi c\frac{\partial T}{\partial r} = \int_\Omega c[\nabla(r\chi)\cdot\nabla T] - \int_\Gamma r\chi c[\nabla T\cdot\hat{\mathbf{n}}] - \int_\Omega \chi c\frac{\partial T}{\partial r} =$$
$$= \int_\Omega rc[\nabla\chi\cdot\nabla T] - \int_\Gamma r\chi c[\nabla T\cdot\hat{\mathbf{n}}] = \frac{I^2}{\pi r_0^2}\int_\Omega r\chi\theta(r_0 - r)r_c. \tag{11}$$

The space of test functions is chosen such that χ vanishs at the Diriclet boundary. Then due to the compatibility condition (9) the boundary term drops out:

$$\int_\Omega rc[\nabla\chi\cdot\nabla T] = \frac{I^2}{\pi r_0^2}\int_\Omega r\chi\theta(r_0 - r)r_c. \tag{12}$$

To account for non-zero Dirichlet boundary conditions, we make a substitution $T \to T + T_b$, where T now has zero boundary conditions, and T_b is an arbitrary function such that $T_b|_\Gamma = T_0$:

$$\int_\Omega rc[\nabla\chi\cdot\nabla T] = \frac{I^2}{\pi r_0^2}\int_\Omega r\chi\theta(r_0 - r)r_c - \int_\Omega rc[\nabla\chi\cdot\nabla T_b] \tag{13}$$

Expanding $T = \sum_j T_j\chi_j$ over basis functions χ_i and using the Galerkin's method, we get the discrete system for the T_j coefficients:

$$\sum_j S_{ij}T_j = b_i, \tag{14}$$

where

$$S_{ij} = \int_\Omega rc[\nabla\chi_i\cdot\nabla\chi_j], \quad b_i = \frac{I^2}{\pi r_0^2}\int_\Omega r\chi_i\theta(r_0 - r)r_c - \int_\Omega rc[\nabla\chi_i\cdot\nabla T_b]. \tag{15}$$

Poisson's equation

The Poisson's (Laplace's) equation for the electrostatic potential $\phi(\mathbf{x})$ with zero charge density is

$$\nabla[\epsilon(\mathbf{x})\nabla\phi(\mathbf{x})] = 0. \tag{16}$$

Boundary conditions

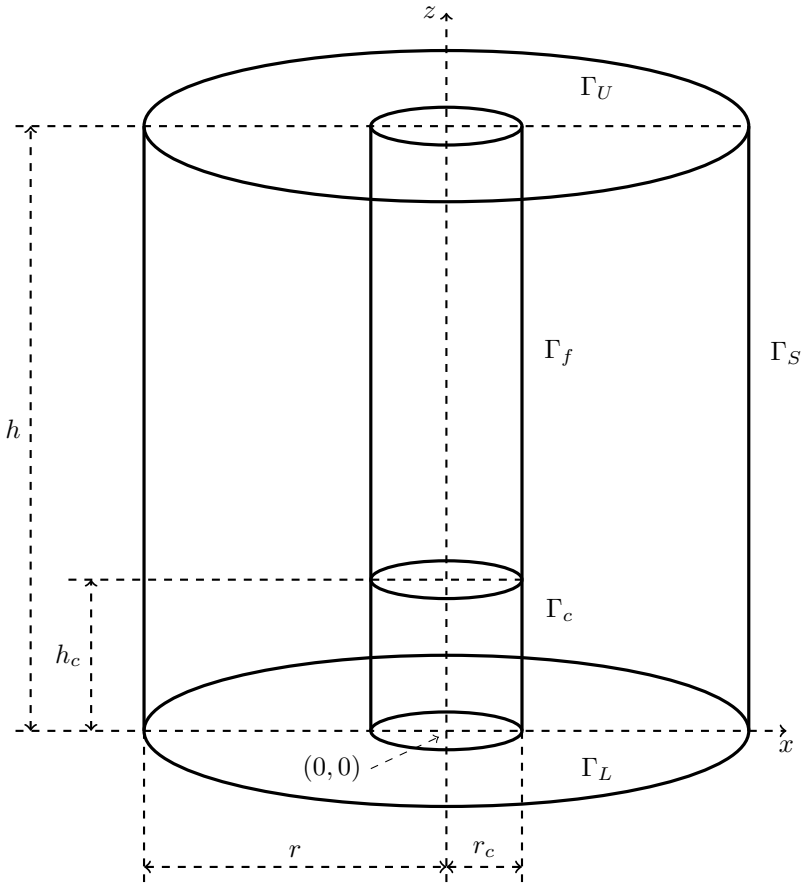

Figure 1: Boundary conditions for the Poisson equation.

Dirichlet boundary conditions are used everywhere:

$$\phi(\mathbf{x})|_{\Gamma_L \cup \Gamma_c} = 1, \quad \phi(\mathbf{x})|_{\Gamma_R} = 0, \quad \phi(\mathbf{x})|_{\Gamma_S} = 1 - \frac{z}{h}, \quad \phi(\mathbf{x})|_{\Gamma_f} = 1 - \frac{R(h_c, z)}{R(h_c, h)}, \tag{17}$$

where $R(z_1, z_2)$ is the resistance of the filament between points z_1 and z_2.

Kinetic Monte-Carlo method

The distribution of vacancies is described by the occupation numbers $v_{\mathbf{i}}$, which can only be 0 (no vacancy) or 1 (single vacancy), with indices $\mathbf{i}$ defined on the discrete 3D uniform grid $\mathcal{G}$,

$$\mathcal{G} = \{\mathbf{i} = (i, j, k) \mid \mathbf{x}_{\mathbf{i}} \in \overline{\Omega}_D\}, \tag{18}$$

where $\mathbf{x}_{\mathbf{i}}$ are the coordinates of the site $\mathbf{i}$. The grid spacing equals $\delta := \{\mathsf{grid\_delta}\}$.

It is assumed that vacancies hop only between nearest-neighbour sites. Each site (except for boundary ones) has six nearest neighbours. The rate (probability per unit time) of hopping is

$$\Gamma_{\mathbf{i}\to\mathbf{i}'} = v_{\mathbf{i}}(1 - v_{\mathbf{i}'})w_0 \exp\left\{-\frac{E_{\mathrm{ac}} + [U(\mathbf{x}_{\mathbf{i}'}) - U(\mathbf{x}_{\mathbf{i}})]}{T[(\mathbf{x}_{\mathbf{i}} + \mathbf{x}_{\mathbf{i}'})/2]}\right\}, \quad U(\mathbf{x}) = q\phi(\mathbf{x}), \tag{19}$$

where $w_0 := \{\mathsf{debye\_frequency}\}$ is the Debye frequency, $T(\mathbf{x})$ is the temperature, $\phi(\mathbf{x})$ is the electrostatic potential, and $q = e > 0$ is the O-vacancy charge. The pre-factor $v_{\mathbf{i}}(1 - v_{\mathbf{i}'})$ takes into account that the hopping is possible only if the source site is occupied ($v_{\mathbf{i}} = 1$) and the destination one is empty ($v_{\mathbf{i}'} = 0$).

Initial conditions

Initially all vacancies are distributed uniformly randomly over $\mathcal{G}$ with the filling factor $\rho := \{\mathsf{initial\_filling}\}$, $0 < \rho < 1$, such that the total number of vacancies is $\lfloor \rho N_{\mathcal{G}} \rfloor$, where $N_{\mathcal{G}}$ is the total number of sites in $\mathcal{G}$.

Boundary conditions

TODO. If the source or final cite lies outside the domain, the hopping probabilities are determined by the boundary conditions.

Algorithm

Variable step size method is used for Monte-Carlo simulation. The following algorithm is used.

1. Identify all possible events and compute their rates $\Gamma_k = \Gamma_{\mathbf{i}\to\mathbf{i}'}$.
2. Compute probabilities of all events: $P_k = \Gamma_k/\Gamma$, where $\Gamma = \sum_k \Gamma_k$ is the total rate. Select the next event randomly according to this probability distribution.
3. Compute the time step: $\Delta t_i = -\ln u/\Gamma$, where where u is a uniform random number in the range $(0, 1)$.
4. Check if the total simulation duration $\Delta t = \sum_i \Delta t_i$ and/or the number of steps exceed the given limits. Abort, if they do.
5. Go to step 1.

Resistance and potential distribution

Interpolation between meshes

Device simulation algorithm

The device simulation proceeds according to the following algorithm:

1. **Initialization.** Read the mesh for the heat equations from the external file; create the mesh for the Poisson equation; initialize the finite-element solvers; initialize the Monte-Carlo solver; generate the initial distribution of O-vacancies.

2. Set bias voltage to zero: $V \leftarrow 0$; set bias voltage sweep direction to "forward": $\xi \leftarrow +1$.

3. **Main loop.** Compute the filament shape.

4. Compute the linear resistance $r_c(z)$; compute the total current I; compute the linear heat source density $I^2 r_c(z)$; compute the Poisson equation boundary condition $\phi(0, z)$.

5. Solve the heat equation for $T(r, z)$; solve the Poisson equation for $\phi(r, z)$; interpolate to the Monte-Carlo grid to obtain $T(\mathbf{x}_i)$ and $\phi(\mathbf{x}_i)$.

6. Estimate the duration $\langle \Delta t \rangle$ of a single Monte-Carlo step. If it is larger than the minimum Monte-Carlo step duration $\langle \Delta t \rangle_{\min} := \{\mathsf{min\_step\_duration}\}$, set time step: $\Delta t \leftarrow \overline{\Delta t}_{\min}$, and go to step 8 (skip Monte-Carlo simulation).

7. Run the Monte-Carlo simulation for $n := \{\mathsf{steps\_per\_round}\}$ steps; compute the elapsed time Δt.

8. Update bias voltage: $V \leftarrow V + \xi s \Delta t$, where $s := \{\mathsf{bias\_sweep\_rate}\}$ is the bias voltage sweep rate.

9. Go to step 3.