User guide for MiTMoJCo

Dmitry R. Gulevich

d.r.gulevich@metalab.ifmo.ru

ITMO University, St. Petersburg, Russia

10 July 2017

Contents

1   Introduction

MiTMoJCo (**Mi**croscopic **T**unneling **Mo**del for **J**osephson **Co**ntacts) is C code designed for modeling superconducting Josephson junctions within the formalism of microscopic tunneling theory. The purpose of the code is to offer implementation of a computationally demanding part of this calculation which is evaluation of the superconducting pair and quasiparticle tunnel currents. The tunnel currents calculated by MiTMoJCo are offered to user's disposal to be employed in a specialized ODE/PDE solver or within a finite difference or finite element scheme in a custom C code. The source code contains examples of modeling some common cases of Josephson contacts.

MiTMoJCo is written by Dmitry R. Gulevich (ITMO University, St Petersburg, Russia) and is made publicly available under the GNU General Public License. This implies that you may freely copy, distribute, or modify the sources, but the copyright for the original code remains with the author and the ITMO University. The code can be downloaded from the github online repository https://github.com/drgulevich/mitmojco. If you find the code useful in your research, we kindly ask you to include a reference to the original paper [1] in studies that use MiTMoJCo.

2   Basic Usage

2.1   Theory behind MiTMoJCo

The goal of MiTMoJCo is to aid evaluation of the tunnel current density in a Josephson tunnel junction,

$$j(\mathbf{r}, t) = \alpha_N \frac{\partial \varphi}{\partial t} + \bar{j}(\mathbf{r}, t) \tag{1}$$

where

$$\begin{aligned}
\bar{j}(\mathbf{r}, t) = \frac{k}{\operatorname{Re} \tilde{j}_p(0)} \int_0^\infty \Bigg\{ & j_p(kt') \sin\left[\frac{\varphi(\mathbf{r}, t) + \varphi(\mathbf{r}, t - t')}{2} \right] \\
& + \bar{j}_{qp}(kt') \sin\left[\frac{\varphi(\mathbf{r}, t) - \varphi(\mathbf{r}, t - t')}{2} \right] \Bigg\} \, dt',
\end{aligned} \tag{2}$$

is the *reduced* tunnel current density which depends on the history of evolution of the superconducting phase difference $\varphi(\mathbf{r}, t)$ through the convolution with pair and quasiparticle time-domain kernels $j_p(\tau)$ and $\bar{j}_{qp}(\tau)$. The tunnel current density (1) enters the integro-differential equation describing dynamics of superconducting phase difference in a Josephson tunnel junction,

$$\frac{\partial^2 \varphi}{\partial t^2} - \left(1 + \beta \frac{\partial}{\partial t}\right) \nabla^2 \varphi + \alpha_N \frac{\partial \varphi}{\partial t} + \bar{j}(\mathbf{r}, t) = 0 \tag{3}$$

$$\mathbf{n} \cdot \left(1 + \beta \frac{\partial}{\partial t}\right) \nabla \varphi = \mathbf{e}_z \cdot [\mathbf{n} \times \mathbf{h}]$$

where $\mathbf{n}$ is the in-plane outward normal, $\mathbf{h}$ is the normalized magnetic field in units $j_c \lambda_J$.

The bar over the reduced tunnel current density $\bar{j}(\mathbf{r}, t)$ and a reduced quasiparticle time-domain kernel $\bar{j}_{qp}(\tau)$ signifies that the normal resistance contribution has been subtracted: it enters to the full tunnel current (1) and the equation (3) explicitly as a damping term $\alpha_N \partial \varphi / \partial t$. This is done for computational reasons to avoid the singularity at $\tau = 0$ and obtain a regular behaviour of the quasiparticle time-domain kernel as a function of time. Furthermore, this allows constructing convenient semi-implicit numerical schemes where part of the tunnel current (the term $\alpha_N \partial \varphi / \partial t$) is computed implicitly. The time-domain kernels are defined by Fourier transforms of the *tunnel current amplitudes* which can be calculated theoretically from the Bardeen–Cooper–Schrieffer (BCS) theory, or, found experimentally.

In Eqs. (3) and (2) time is measured in units of the inverse angular Josephson frequency ω_J^{-1}, the spatial coordinates are expressed in units of the Josephson penetration length λ_J, the reduced current density $\bar{j}(\mathbf{r}, t)$ is normalized to $V_g / A R_N$, where V_g is the gap voltage, A is the total area of the junction and R_N is the normal resistance.

2.2   Compilation requirements

The source code can be compiled with available C compilers. Its only dependency is the OpenMP library used to perform parallel computation of the tunnel currents on a shared memory machine.

The source code contains:

`mitmojco.c` and the header `mitmojco.h`: core MiTMoJCo routines,
`opt_filter.c` and the header `opt_filter.h`: optimum filtration routine,
directory `amplitudes/` containing fits of the tunnel current amplitudes,
directory `python/` containing supplementary Python routines,
`doc/mitmojco-user-guide.pdf`: this user guide,
`Makefile` to compile the code.

Before compiling the code user should be modify the `Makefile` to suit his needs. By default GCC compiler

```
CC = gcc
```

with optimization flag

```
OPTIMIZE  = -Ofast -march=native
```

are enabled. Change this line if working in the debug mode or a different level of optimization is required.

2.3   Tunnel Current Amplitudes

The current amplitude files have extension `.fit`. User is supplied with a set of tunnel current amplitudes which are pre-calculated for a symmetric junction (Nb-AlO$_x$-Nb)

with $\Delta_1 = \Delta_2 = 1.4$ meV at temperature T=4.2 K and different degrees of smoothing, see table 2 In the calculations of the current voltage characteristics (IVC) of Josephson flux flow oscillator in Ref. [1] value $\delta = 0.008$ was used as it fitted best the experimental IVC of a small Nb/AlO$_\mathrm{x}$/Nb junction. Note, that the tunnel current amplitudes are supplied without the account of the pair suppression α_supp which is controlled separately within the function call (see the subsection 2.4 below).

For historical reasons we also provide fits given in the Refs. [2] and [3]. However, due to their extremely bad performance in the subgap region, we discourage from using them for other purposes than debugging your code and/or reproducing results of Refs. [2] and [3, 4].

Filename	δ	N	Rel. tolerance	Abs. tolerance
BCS42_001.fit	0.001	10	5.0×10^{-3}	1.0×10^{-3}
BCS42_002.fit	0.002	9	5.0×10^{-3}	1.0×10^{-3}
BCS42_004.fit	0.004	9	4.0×10^{-3}	8.0×10^{-4}
BCS42_008.fit	0.008	8	5.0×10^{-3}	1.0×10^{-3}
BCS42_016.fit	0.016	8	5.0×10^{-3}	1.0×10^{-3}
BCS42_032.fit	0.032	8	4.0×10^{-3}	8.0×10^{-4}
BCS42_064.fit	0.064	8	5.0×10^{-3}	1.0×10^{-3}

Table 1: Library of pre-calculated fits of BCS tunnel current amplitudes provided by MiTMoJCo. Tunnel current amplitudes are calculated from the BCS theory at a fixed temperature $T = 4.2$ K for a symmetric junction made by two identical superconductors with gap energy 1.4 meV and are smoothed using different values of the phenomenological smoothing parameter δ. Number of the fitting exponentials N, relative and absolute tolerances of the fit in the frequency region $|\omega/\omega_g| \leq 2$ are also shown.

The fits are found by calculating parameters which minimize the cost function

$$\sum_X \int_0^2 D(X^\mathrm{fit}, X^\mathrm{exact})^2 \, d\xi, \tag{4}$$

where

$$D(X^\mathrm{fit}, X^\mathrm{exact}) \equiv \frac{|X^\mathrm{fit} - X^\mathrm{exact}|}{\max(\tau_a/\tau_r, |X^\mathrm{exact}|)}, \tag{5}$$

is the relative difference between the fitted and exact functions $X = \mathrm{Re}\,\tilde{j}_p(\xi)$, $\mathrm{Im}\,\tilde{j}_p(\xi)$, $\mathrm{Re}\,\tilde{j}_{qp}(\xi)$, $\mathrm{Im}\,\tilde{j}_{qp}(\xi)$, and $\tau_{a,r}$ are absolute and relative tolerances, respectively.

Tunnel current amplitudes for asymmetric junctions (e.g. Nb-AlN-NbN and similar) will also be provided and are coming soon.

4

Filename	N	Source
OSZ_Table_1.fit	4	Table 1 in Ref. [2]
OSZ_Table_2.fit	5	Table 2 in Ref. [2]
GJHS_Table_1.fit	4	Table 1 in Ref. [3]
GJHS_Table_2.fit	5	Table 2 in Ref. [3]

Table 2: Fits of tunnel current amplitudes given in Refs. [2] and [3].

2.4   MiTMoJCo interface

The interface for MiTMoJCo is implemented in the spirit of object-oriented programming. To use MiTMoJCo in your C code include the header file,

```
#include "mitmojko.h"
```

and create a tunnel current object (`TunnelCurrentType` pointer) by calling the constructor declared in the header file as

```
TunnelCurrentType* mitmojco_create(
    char *filename, double a_supp, double kgap, double dt,
    int Ntotal, double *phi, int Nskip, int *skipinds );
```

Arguments of the constructor are:

> `char *filename`: tunnel current amplitudes file,
> `double a_supp`: pair current suppression parameter α_{supp},
> `double kgap`: normalized gap frequency k,
> `double dt`: integration time step,
> `int Ntotal`: size of `phi` array,
> `double *phi`: pointer to the superconducting difference array,
> `int Nskip`: number of shadow nodes to skip,
> `int *skipinds`: pointer to the array storing indices of the shadow nodes.

Often in numerical schemes one introduces *shadow nodes* used for treating the boundary conditions. However, the tunnel current only needs to be evaluated for the *physical nodes* and, therefore, its evaluation at the shadow nodes can be skipped. The last two arguments accepted by the constructor are the number of the shadow nodes and the pointer to the array with their indices in the `phi` array.

The type `TunnelCurrentType` is defined in MiTMoJCo header file as a structure,
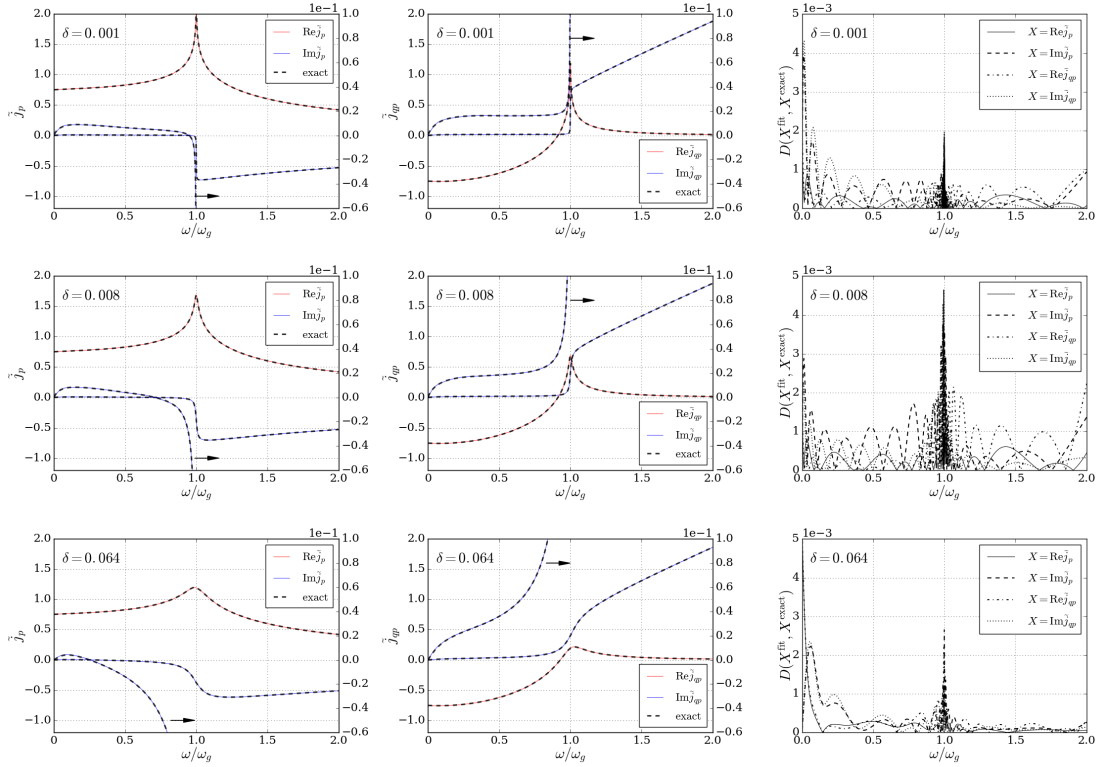
```
typedef struct {
```

Figure 1: Fits of tunnel current amplitudes: `BCS42_001.fit` (top), `BCS42_008.fit` (middle) and `BCS42_064.fit` (bottom). Solid red and blue lines represent fit to the real and imaginary parts of the pair and quasiparticle currents in the form of a sum of exponents. The exact theoretical tunnel current amplitudes based on which the fitting was done, are shown by dashed lines for comparison. To illustrate the behaviour of the tunnel current amplitudes in the subgap region, 20x zoom of the imaginary parts of the tunnel current amplitudes is shown. Relative difference of the fitted and exact amplitudes defined by Eq. (5) is shown on the left.

```
    bool error;
    double Rejptilde0;
    double alphaN;
    double *jbar;
    void *self;
} TunnelCurrentType;
```

which has member variables:

bool error: handler to check that no errors occurred during the object creation (**false** if no errors occurred an **true** is error occurred such as e.g. missing amplitudes file or incorrect file format).

double Rejptilde0: normalized critical current $\mathrm{Re}\,\tilde{j}_p(0)$.

double alphaN: damping due to the normal resistance α_N.

double *jbar: pointer to the reduced current density array $\bar{j}(\mathbf{r}, t)$.

void *self is used internally to access private member variables by the MiTMoJCo methods and is not intended for use by a regular user.

Object member variables are accessed via the arrow operator **->**. For example, to access value of the error flag use **tunnel_current->error**.

To illustrate a particular example, a code implementing 1D model long Josephson junction discretized into 100 nodes (98 physical and 2 shadow nodes to treat the boundary conditions), may start with

```
// Superconducting phase difference array (100 nodes)
double *phi = malloc( 100 * sizeof(double) );

// Indices of the shadow nodes
int *skipinds={0,99};

// Create tunnel current object
TunnelCurrentType *tunnel_current = mitmojco_create(
    "amplitudes/BCS42_008.fit", 0.7, 3.3, 0.01,
    100, phi, 2, skipinds );

// Check that no errors occurred during the object creation
if( tunnel_current->error )
    return;
```

The first two calls define superconducting phase difference array **phi** and array of shadow nodes indices **skipinds**. Then, the tunnel current object is created with parameters: tunnel current amplitudes from file **BCS_T42_smooth.fit** will be used, pair current suppression $\alpha_{\mathrm{supp}} = 0.7$, normalized gap frequency $k = 3.3$ and time step 0.01. Two shadow nodes with indices 0 and 99 will be skipped in the evaluation of the tunnel current.

Upon creating the tunnel current object the three methods available to the user are, as declared in the header file,

```
extern void mitmojco_init( TunnelCurrentType *object );
extern void mitmojco_update( TunnelCurrentType *object );
extern void mitmojco_free( TunnelCurrentType *object );
```

Below we will discuss each of these methods in more detail.

Method `mitmojco_init` initializes the state of the memory integral, assuming no dynamics in the past (that is, the supplied state is assumed to be stationary in which the system existed for an infinite time). Use

```
mitmojco_init( tunnel_current );
```

to initialize the tunnel current object based on the initial value of `phi`. Note that that `mitmojco_init` should be called after the array `phi` is initialized as its values will be accessed by the address supplied to the object constructor.

Method `mitmojco_update` updates values of the tunnel current at the physical nodes, based on the updated values of the superconducting phase difference `phi` obtained within the numerical scheme (recall that the addresses of the physical nodes in `phi` are already known to MiTMoJCo at the constructor call during the object creation). Call

```
mitmojco_update( tunnel_current );
```

to calculate values of the reduced tunnel current density $\bar{j}(\mathbf{r}, t)$. These can be later accessed via the object pointer, `tunnel_current->jbar`.

Finally, `mitmojco_free` is used to empty the memory allocated for the tunnel current object,

```
mitmojco_free( tunnel_current );
```

It is possible to deal with several tunnel current objects simultaneously, e.g. FFO and a SIS, or, an array of Josephson junctions by creating independent object. As an example,

```
double phi_sis;

TunnelCurrentType *sis_tunnel_current = mitmojco_create(
    "amplitudes/BCS42_008.fit", 0.7, 4.1, dt,
    1, &phi_sis, 0, NULL );

double *phi_ffo = malloc( 1000 * sizeof(double) );
int *skipinds_ffo={0,999};

TunnelCurrentType *ffo_tunnel_current = mitmojco_create(
    "amplitudes/BCS42_016.fit", 0.8, 3.3, dt,
    1000, phi_ffo, 2, skipinds_ffo );
```

creates independent objects for a flux flow oscillator and SIS junction, each with its own parameters and different tunnel current amplitudes.

8

2.5   Optimum filtration of a sinusoidal signal

Optimum filtration routine for efficient calculation of a constant component of a sinusoidal signal is implemented in a supplementary code `opt_filter.c` and `opt_filter.h`. The code implements the algorithm outlined in Ref. [2]. To access the optimum filtration routine include the header file,

```
#include "opt_filter.h"
```

and begin with declaring the filter object, for example,

```
OptFilterType *voltage_filter = opt_filter_create(5);
```

where the argument of the constructor controls shape of the filtering window function: parameter n in Ref. [2]. n is integer and, in practice, takes values between 1 and 5, where 1 correspond to the arithmetic mean of the recorded values, see Ref. [2] for details. The type `OptFilterType` which emulates the class is defined in `opt_filter.h` as a structure,

```
typedef struct {
    int n;
    double a;
    double *y;
    void *self; // pointer to private struct
} OptFilterType;
```

The four methods accessible by the user are

```
extern void opt_filter_init( OptFilterType *object );
extern void opt_filter_update( OptFilterType *object, double signal );
extern double opt_filter_result( const OptFilterType *object );
extern void opt_filter_free( OptFilterType *object );
```

Method `opt_filter_init` initializes the filter object,

```
opt_filter_init(voltage_filter);
```

Method `opt_filter_update` makes a record of the signal value,

```
opt_filter_update(voltage_filter, voltage );
```

Method `opt_filter_result` returns value of the calculated DC component once the calculation is finished,

```
Vdc = opt_filter_result(voltage_filter);
```

Finally, `opt_filter_free` is used to clear the memory allocated to the filter object,

```
opt_filter_free(voltage_filter);
```

3 Examples

Examples of how MiTMoJCo can be used in modeling several common cases of Josephson junctions are provided.

To compile a particular example, type

```
$ make example-#
```

where `#` is the example number, or, type

```
$ make all
```

or, simply,

```
$ make
```

to compile all the provided examples.

3.1 Example 1: Current-biased SIS junction

Current-biased Josephson junction is described by

$$\ddot{\varphi} + \alpha_N \dot{\varphi} + \bar{j}(t) - \gamma = 0,$$

$$\bar{j}(t) = \frac{k}{\operatorname{Re} \tilde{j}_p(0)} \int_0^\infty \left\{ j_p(kt') \sin\left[\frac{\varphi(t) + \varphi(t - t')}{2}\right] + \bar{j}_{qp}(kt') \sin\left[\frac{\varphi(t) - \varphi(t - t')}{2}\right] \right\} dt',$$

where γ is the applied biasing current. To compile the first example, type in the terminal

```
$ make example-1
```

which produces executable `example-1` in the current directory. Executing

```
$ ./example-1
```

without the command line arguments produces an output

```
#=================================================
#---- Example 1: Current-biased SIS Junction -----
#=================================================
# Incorrect number of arguments.
# Please, supply 1 or 3 arguments in the following order:
# 1. Value of bias current (gamma_start)
# or
# 1. Starting value of bias current (gamma_start)
# 2. Final value of bias current (gamma_finish)
# 3. Bias current step (gamma_step)
```

Run

```
$ ./example-1 1.1
```

to calculate the normalized DC voltage at a specified bias current (e.g., $\gamma = 1.1$ in this example), or,

```
$ ./example-1 1.1 0.0 0.01
```

for a range of values. In this case, γ takes 1.1 as a starting value and decreases down to 0.0 with step 0.01.

3.2   Example 2: Voltage-biased SIS junction under AC drive

Current through a small voltage-biased Josephson junction is given by the Eq. (1). In this case, one does not need to solve the differential equation, rather, the superconducting phase difference $\varphi(t)$ can be easily found from the fundamental Josephson relation if the time-dependence of the applied voltage is known.

All one needs to do in MiTMoJCo is to update φ and call

```
mitmojco_update( sis_tunnel_current );
```

at each time step to get the tunnel current.

In this example we assume a harmonic AC drive

$$V(t) = V_{dc} + V_{ac}\cos(\omega t)$$

and use the optimum filtration routine to get the resulting DC current component. Compile the code with

```
$ make example-2
```

and run

```
$ ./example-2
```

to see the command line arguments needed to run the simulation,

```
#=================================================
#---- Example 2: Voltage-biased SIS Junction -----
#=================================================
# Incorrect number of arguments.
# Please, supply 5 arguments in the following order:
# 1. Vac
# 2. omega
# 3. Vdc_start
# 4. Vdc_finish
# 5. Vdc_step
```

3.3   Example 3: Sine-Gordon breather in long Josephson junction

In this example we consider a 1D model of a long Josephson junction,

$$\varphi_{tt} - \left(1 + \beta\frac{\partial}{\partial t}\right)\varphi_{xx} + \alpha_N\varphi_t + \bar{j}(x,t) = 0$$

with open boundary conditions,

$$\varphi_x(\pm L/2, t) = 0.$$

Running the executable

```
$ ./example-3
```

produces the file `breather.dat` where dynamics of the superconducting phase difference is written during the time evolution of the breather. Use the supplied Python routine `breather.py` to animate. For this, copy the `breather.dat` to the `python` folder and start the Python 3 interactive mode,

```
$ ipython
```

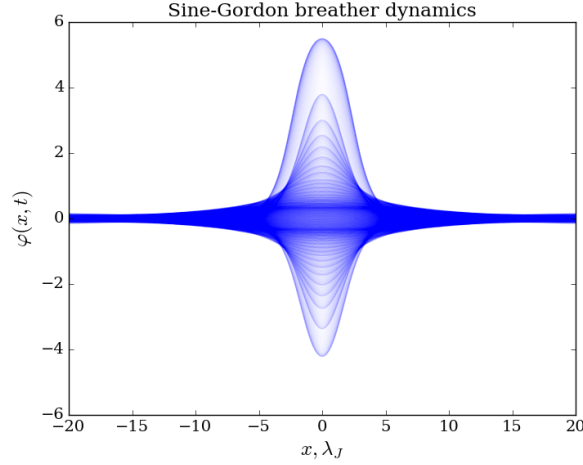Within the Python interactive mode execute

```
In [1]: run breather.py
```

Figure 2:   Time evolution of sine-Gordon breather. Many time frames super-imposed to illustrate dynamics of the superconducting phase difference. The parameters used in the calculations are: Josephson junction normalized length $L = 40$, tunnel current amplitudes file `BCS42_008.fit`, pair current suppression $\alpha_{\mathrm{supp}} = 0.7$, normalized gap frequency $k = 3.3$.

3.4   Example 4: Fluxon in an Annular Josephson junction

(in preparation)

3.5   Example 5: Flux Flow Oscillator

(in preparation)

4   Disclaimer

MiTMoJCo comes without any warranty nor guarantees to produce correct results. The author accepts no responsibility whatsoever for source code bugs, failures, crashes, expected or unexpected behavior.

References

[1] D. R. Gulevich, V. P. Koshelets, and F. V. Kusmartsev, "Josephson Flux Flow Oscillator: the Microscopic Tunneling Approach", Phys. Rev. B, to be published; arXiv:1704.03045.

[2] A. A. Odintsov, V. K. Semenov and A. B. Zorin, IEEE Trans. Magn. 23, 763 (1987).

[3] N. Grønbech-Jensen, S. A. Hattel and M. R. Samuelsen, Phys. Rev. B 45, 12457 (1992).

[4] S. A. Hattel, N. Grønbech-Jensen and M. R. Samuelsen, Phys. Lett. A 178, 150 (1993).

[5] J. R. Tucker and M. J. Feldman, Rev. Mod. Phys. 57, 1055 (1985).