Brief documentation for GETELEC

Andreas Kyritsakis and Flyura Djurabekova
Deparment of Physics and Helsinki Institute of Physics, University of Helsinki,
PO Box 43 (Pietari Kalmin katu 2), 00014 Helsinki, Finland

GETELEC is a scientific software for calculating electron emission currents and the Nottingham effect heat. For details see the corresponding scientific paper. Updated versions of the GETELEC project can always be downloaded from https://github.com/AndKyr/GETELEC.

I. GENERAL INSTRUCTIONS FOR USING GETELEC

A. Requirements, downloading and compiling

Currently GETELEEC is developed only for Linux systems, but a version that can be compiled in other operating systems will be available in forthcoming updates. After downloading the zip file, the user needs to extract it and in the resulting folder execute "make" (GNU make is required) to compile it and build the GETELEC static and dynamic libraries. For successful building, it is required that in the system there are installed gfortran (version 5 or later) and gcc (version 5 or later) and a recent version of ar. By executing "make tests" some test programs are compiled and executed, outputting some of the plots included in the present paper. For those tests to produce correct results, existing installation of python (version 2.7 or later) with the accompanying libraries "numpy" (version 0.13 or later), "matplotlib" (version 1.3 or later) and "scipy" (version 0.13 or later) is required.

B. Usage as a FORTRAN static library

GETELEC consists mainly of a FORTRAN 2003 module ("modules/getelec.f90") which contains all the essential data types and subroutines for the calculations. After the compilation, the user can call GETELEC public data types and subroutines by including the static library "lib/libgetelec.a" and the dependency library "lib/libslatec.a". The main data type in the module that handles all the emission data and parameters is `EmissionData`. The user mainly has to define the input parameters which are members of this data type. The first member that has to be defined is the `mode` that determines how the barrier will be input. If `mode`=(0 or -1), the barrier will be given in the form of the 3-parameter model of eq. (2) of the corresponding paper and the user has to specify the three members of the data type:

- `F`: Local field in V/nm
- `R`: Local radius of curvature in nm
- `gamma` : "enhancement factor" γ of eq. (2)

Another two members must always be specified as input are:

- `W`: Work function ϕ in eV
- `kT`: k_BT (Boltzmann constant $\times$ temperature) in eV

If `mode` is set to (1,2,-10,-11,-20,-21) then the barrier has to be input (x_i, V_i) vectors, and the user must allocate and give values to the members-vectors `xr`, `Vr` correspondingly. For more details about the specific modes of calculation see the commented source file. In general, when (x_i, V_i) are available, the recommended mode of calculation is `mode=-21`. Furthermore, the logical variable member of the data type `full` must be determined. If it is true, the full calculation is carried out, while if it is false, the integration over energies is done according to the GTF approximations for CPU efficiency.

When these input parameters are set, the main subroutine `cur_dens` might be called to calculate the current density and the Nottingham heat. After successful execution, all the output members of the data type will be determined. They are:

- `Jem`: The current density J in A/nm^2

- `heat`: The Nottingham heat density P_N in W/nm^2
- `Gam`: The Gamow exponent G
- `xm`: The abcissa where the maximum of the barrier appears in nm
- `Um`: The value U_m of the maximum of the barrier in eV (with respect to the Fermi energy)
- `maxbeta`: derivative $-G'(E=0)$ in eV^{-1}
- `minbeta`: derivative $-G'(E=U_m)$ in eV^{-1}
- `regime`: character indicating the regime of calculation. `'f'` for field, `'i'` for intermediate and `'t'` for thermal
- `sharpeness`: character indicating if the approximate formula ("blunt") or numerical integration ("sharp") where used. `'b'` and `'s'` correspondingly.
- `ierr`: integer indicating errors in the calculation. If it is 0 the subroutine has been executed successfully. If not, detailed debugging information may be found on the commented code.

C. User-defined parameters

Finally, GETELEC gives the ability to the user to define some global calculation parameters. In order to do that, a text file "GetelecPar.in" must be present in the working directory. Its format must be exactly as specified in the existing example file. If the parameter file does not exist in the directory, the input parameters will take their standard default value. However, if it exists, all values must be specified in the same order. The specified parameters are the following:

- `xlim`: χ_{lim}(Described in the main text).
- `nlimfield`: n_{high} (Described in the main text).
- `nlimthermal`: n_{low} (Described in the main text).
- `nmaxlim`: Maximum limit for the usage of GTF formulas in Field regime. If $n >$`nmaxlim`, the temperature corrected version of the FN equation is used.
- `gammalim`: Maximum acceptable γ. If $\gamma >$ `gammalim` the approximate analytic expressions are used.
- `Jfitlim`: Minimum limit for interpolation with spline. If $J <$ `Jfitlim` polynomial fitting is forced.
- `varlim`: Limit of variance for the fitting approximation (has meaning for `mode`=-2). If the fitting variance is more than `varlim` the algorithm switches to spline mode.
- `epsfit`: Requested algorithm for all fittings of the input potential (x_i, V_i)
- `Nmaxpoly`: Maximum degree of the fitted to (x_i, V_i) polynomial.
- `spectra`: If true, spectroscopy data (Energy distribution of the emitted current) are written to a file "spectra.csv".

II. INTERFACE WITH C - DYNAMIC LIBRARY

GETELEC provides with an interface so that its main functions can be called from the C language. Its communication is done through a C structure that contains all the input-output parameters. The input parameters are: `F`, `W`, `R`, `Temp` = Temperature in K , `xr`, `Vr`, `mode`, `full`. The corresponding output variables are `Jem`, `heat`, `regime`, `sharp`, `ierr`. They all have exactly the same meaning and format as their corresponding for the FORTRAN type described in section I B. In order to call GETELEC from C one has to include the header file "modules/getelec.h" and link with the dynamic library "lib/libgetelec.so". Then a data structure with the above parameters has to be created and the interface C function "`cur_dens_c()`" has to be called.

III. INTERFACE WITH PYTHON - FITTING ALGORITHM

Once GETELEC can be called from the C language and built into a dynamic library, it is very easy to write an interface also for Python. Such an interface would be very useful since Python provides with extensive scientific libraries that can be used to manipulate very easily the input and output of GETELEC. It also facilitates the usage of GETELEC for the analysis of experimental $I-V$ data as the python library scipy provides with the fitting algorithm.

The python module "getelec_mod.py" provides with this interface. It contains all the necessary classes and functions to facilitate it. All the communication is done through the class `Emission`, which has exactly the same data members as the corresponding C structure. An object of the class can be easily created by calling the function `emission_create()` and giving the appropriate input. The arrays `xr,Vr` must be given as numpy arrays. The output object contains both all the input parameters and the output results from GETELEC.

Another important function contained in "getelec_mod.py" is `fitML()`. It takes as input two arrays of experimental I-V data in the form of $1/V - \log(I)$ and fits to them the GETELEC parameters that best reproduce them. See section 3.2 of the paper for more details. The other input parameters of the function are lists with 3 elements that contain the minimum value, the initial guess and the maximum value of every fitting parameter.

The simplest way to fit experimental data is putting the $V-I$ data in two columns of a .csv file (text file with comma separated values) and run the script "python/fitdata.py" giving the name of the file as command line argument. The fitting parameters bounds and initial guesses can be changed by modifying the first lines of the script.