This code uses the formalism of non-equilibrium Green functions to obtain transient responses of observables (current densities, conductance, spectral functions) of electronic systems interacting with electromagnetic fields. The code offers a method to find the solution of the dynamical evolution of interacting many-body systems. Various types of Green’s functions (Matsubara, left/right mixed, retarded, advanced, lesser, greater) are used to solve the corresponding equations of motion. The methods use self-energy functionals and Dyson equations. The code is aimed at the study of transient dynamics from an initial equilibrium state, induced by time-dependent electromagnetic contributions to the Hamiltonian, using a contour that includes an imaginary time (Matsubara branch) plus real time.

**Modules:** for a description of the formalism and the used numerical approximations read the “Description of the non-equilibrium Green formalism for electronic systems interacting with electromagnetic fields”

* \_\_init\_\_.py: it defines and initializes variables in the code
* Analytical\_continuation.py: it recovers the lesser, greater, retarded and advanced Green functions and define the causal and anti-causal functions, see sections 2.5 and 2.7
* Bootstrap.py: it computes the first k time-steps to calculate the Volterra integro-differential equations (VIDE). See Section 5.3
* contour\_funcs.py: defines the functions on the Keldysh-Schwinger contour. See section 2.5 and 2.8 for a definition of the contour and the functions defined on it.
* convolution.py: see section 5.2 for a description of how the convolutions are computed for the different functions on the contour.
* interpolator.py: see section 5.1 for a description of the numerical approximation to interpolations used in the code.
* lattice.py. defines a lattice model.
* linalg.py: defines the routines for operations of matrices and tensors used in the code.
* matsubara\_init.py: computes the values for Matsubara Green functions on the imaginary part of the contour.
* observables.py: defines the current density operators, see section 3.6
* printing.py:
* stepping.py: defines stepping along the real time part of the contour, see section 5.3
* time\_evolution.py: it defines how the different functions (Green functions, self-energies, screened interactions) are propagated along the real time part of the contour. See sections 2.8-2.11
* utils.py:
* vide.py: it defines the numerical approximations for the Volterra integro-differential equations (VIDE), see section 5.3.