

**Objective:** to create interactive documentation system of Nemo5 entities

- Access available options of an entity
- Validate options on an inputdeck
- Generate an updated manual automatically

**Approach:**

[1] to develop an interface between developers and nemo5 entities options.

[2] to define a API to access information about available option and properties.

[3] to validate the correctness of and inputdeck on runtime.

**Results:**

- “./nemo --help #Solver” lists all available options for a specified solver, it shows a default value if it is not required and a description.
- “./nemo --list-solves” lists all available solvers.
- “./nemo --manual”, generate an updated manual including all solvers and options registered [not available yet].



```

carter.rcac.purdue.edu - PuTTY
-----
-bash-4.1$ ./nemo --help Poisson
Usage: nemo [INPUTDECK FILE]... [PETSC CONFIG_PARAMS]...
NanoElectronics Modeling Tool 5 (NEMO5), purdue university; gekco group
Type nemo --version to see the program version and modules
Type nemo --manual to generate pdf manual
Type nemo --list-solvers to list configured Solvers
Type nemo --help [SOLVER_TYPE] to see solver options
-----
Poisson : This simulation solves the linear Poisson equation for a fixed density
[#] (*) = required, ( ) = default value atomistic_output
-----
[1] ( ) atomistic_output ( ) : Save atom-based quantities to file. The list can contain the entries potential, charge, charge_cm-3, free_charge, free_charge_cm-3, doping, doping_cm-3, conduction_band and valence_band. For simulations without grid parallelization VTK is used as format, otherwise Silo
[2] ( ) average_over_cell (true) : Leave this one true (default).
[3] ( ) charges_model (electron_hole) : Can be either electron hole (most cases), in which case doping charges are added to the excess charges from a Schrodinger or other solver, or electron_core_model in which case there are only electrons and these are counter balanced with a fixed positive charge based on the atomic type
[4] ( ) chem_pot (0) :
[5] ( ) copy_solution_from ( ) :
[6] ( ) density_solver ( ) : A list with names of solvers where charge can be found
[7] ( ) do_nonlinearpoisson_outputs_xyz_format (If true, then output will be done in the cartesian XYZ format, which is handy for Matlab-processing.) :
[8] ( ) do_output_nonlinearpoisson_potential (false) : If true, then at the end of NonlinearPoisson, the potential is output in a separate file in FEM format. This potential can be used later to restart the simulation, or to do a fixed potential run
[9] (*) domain : The name of the domain in which the simulation is carried out. corresponding the name parameter in the Domain section
[10] ( ) fem_output ( ) :
[11] ( ) friends ( ) :
[12] ( ) laplacian ( ) :
[13] (*) name : A unique name tag / identifier for the specific instance of the solver
[14] ( ) node_potential_output (false) : If true, the potential is written to (sim-name)_nodal_potential.dat
[15] ( ) one_dim_output ( ) : Interpolates atom-based quantities onto an axis and generates 1D ASCII output compatible with 1D Matlab-plots. The list can contain the entries potential, free_charge_cm-3, doping_cm-3, conduction_band and valence_band.
[16] ( ) one_dim_output_average (true) : If true then unit cells are used for the 1D discretization along some direction and some averaging is done within the cells. If false then the orthogonal projection of the atomic position serves as the 1D discretization
[17] ( ) poisson_vtk_mesh ( ) :
[18] ( ) shift_copied_solution_by_voltage (0.0) :
[19] ( ) tic_toc_name ($name) :
[20] (*) type : The simulation type ( )
[21] ( ) use_average_density_as_a_guess (false) : Must be false (true has a bug)
-----
-bash-4.1$
  
```