Extending Nemo 5 using Python scripting (PythonSolver)

Daniel Mejia, Yaohua Tan, Yu He, Tillmann Kubis, Michael Povolotskyi, Jean Michel Sellier, Jim Fonseca, Gerhard Klimeck

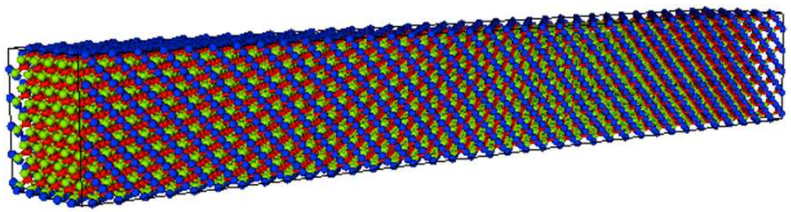
Network for Computational Nanotechnology (NCN)

Electrical and Computer Engineering

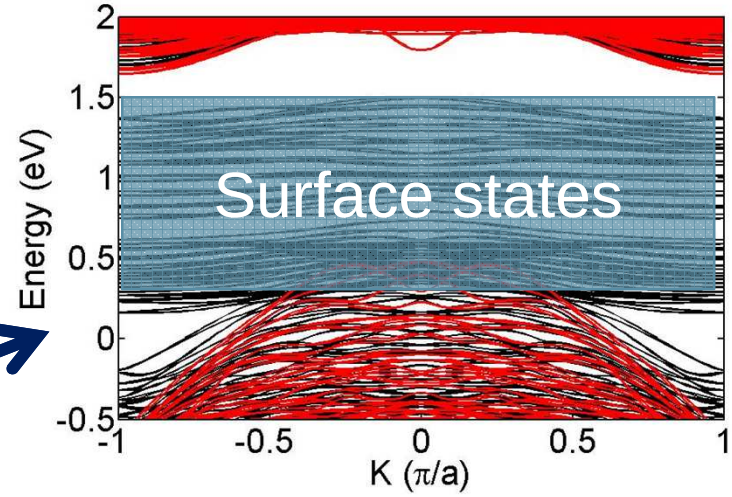
Surface Passivation

- Unpassivated surfaces contain dangling bonds and introduce undesired eigenstates within the bandgap

GaSb nanowire



Nanowire bandstructure
Red: with passivation
Black: without passivation



The goal of passivation : remove surface state from bandgap

- There is not good parameters for passivation
- How determine passivation parameters?

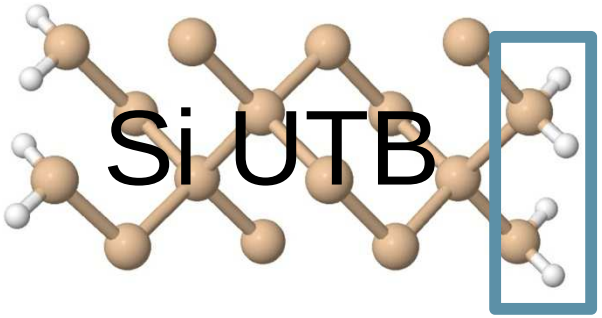
Surface Passivation

Example : optimizing Si passivation parameter.

- Determine the passivation parameter for Si-UTB to match DFT results.

Method:

Calculate E(k) using Nemo5 with different V_S_S_Sigma_H_Si and V_S_P_Sigma_H_Si.
Compare TB bands to reference bands(DFT)



Coupling parameters of H-Si bond are needed.

input deck

```
Material{
    name = Si
    tag = substrate
    crystal_structure = diamond
    regions = (1)
    Bands:TB:sp3d5sstar:param_set = param_Boykin
    Bands:TB:sp3d5sstar:param_Boykin:V_S_S_Sigma_H_Si = -4.0
    Bands:TB:sp3d5sstar:param_Boykin:V_S_P_Sigma_H_Si = 0.3
}
```

Define Material structure specifying a parameter set

```
solver {
    name = Si
    type = Schroedinger
    domain  = structure1
    active_regions = (1)
    job_list  = (…,calculate_band_structure)
    output = (energies, k-points)
    tb_basis = sp3d5sstar
    k_space_basis   = reciprocal
    k_points = [(0.5,0.0),(0,0),(0.5,0.5)]
    number_of_nodes = (10,10)
}
```

Calculate band structure by Nemo5

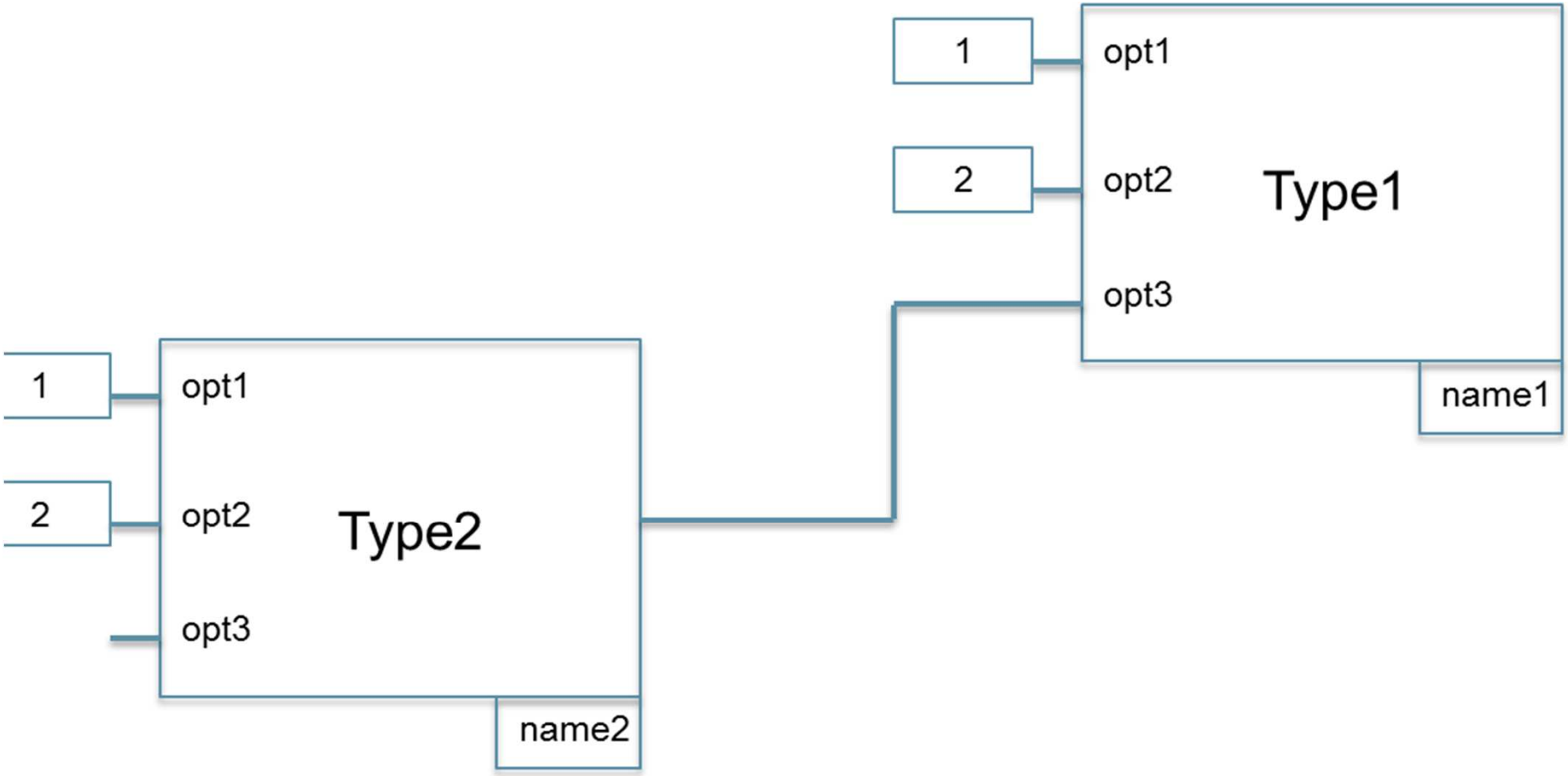
A possible Solution!!!

- Create a set of input decks with different options
- Consolidate all results
- Use an external tool (such as Matlab), post-processing files
- OR use a Python-Solver

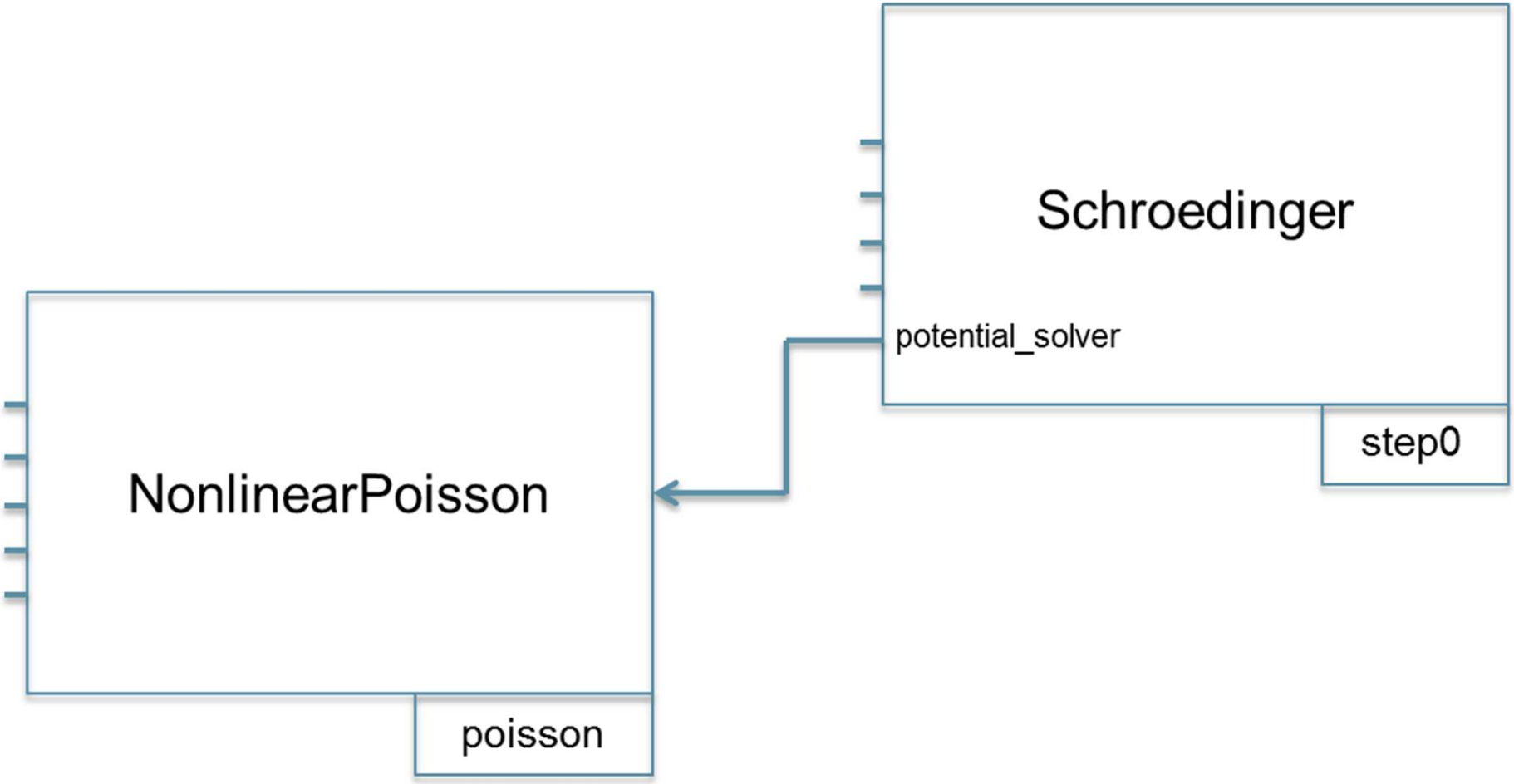
Solver

```
solver  {
        name = solver_name
        type = solver_type
        domain = solver_domain
        input_option_1  = input_1
        input_option_2  = input_2
        input_option_3  = input_3
        input_option_4  = input_4
        …
}
```

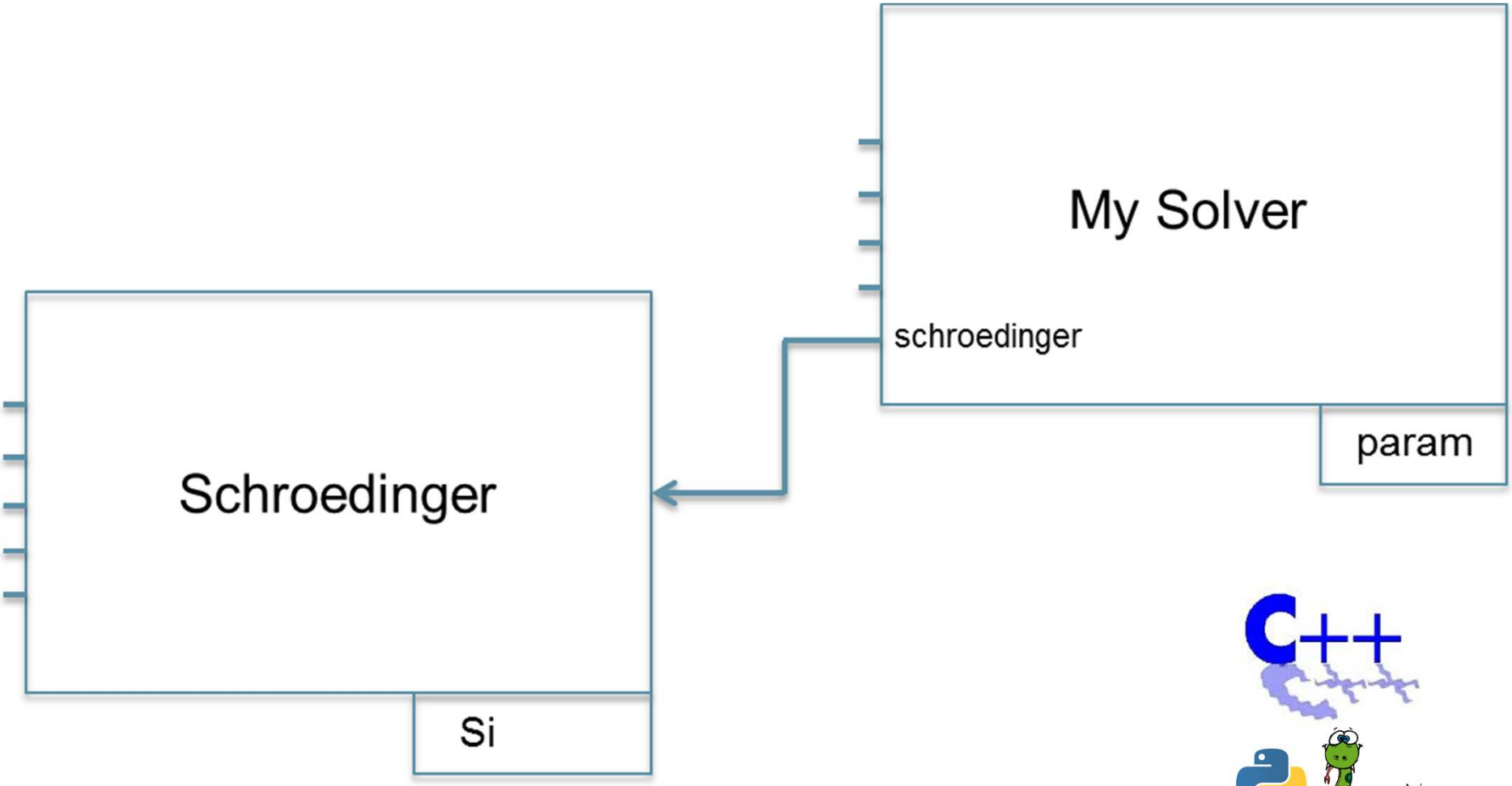
Solvers Interaction



Solvers Interaction



New Solver



Example: Nemo5 input deck

```
Material
    {
        name = Si
        tag = substrate
        crystal_structure = diamond
        regions = (1)
    }
```

Define the structure without specifying the parameter set; (parameters with be specified in the python script)

```
    Domain
    {
        dimension = (1,1,2)
        periodic = (true, true, false)
        crystal_direction1 = (1,0,0)
        crystal_direction2 = (0,1,0)
        crystal_direction3 = (0,0,1)
        space_orientation_dir1 =  (1,0,0)
        space_orientation_dir2 =  (0,1,0)
        regions = (1)
        passivate= true
        geometry_description = simple_shapes
    }
```

A silicon UTB along 001 direction is define.
The thickness is 2 unit cell, 8 Si mono-layers, with hydrogen passivated

Nemo5 input deck

```
Solvers {
    solver {
        name = Si
        type = Schroedinger
        …
    }
    solver {
        name = d1
        type = PythonSolver
        schroedinger = Si
        python_solver = ./input_python/input_python.py
    }
}
Global {
  solve = (d1)
  database = ./materials/all.mat
}
```

Specify the python solver

schroedinger: name of the solver that calculate Bandstructures.

python_solver: path to the python script

Solve the PythonSolver (d1)

Python script

./input_python/input_python.py

1) Import Libraries

↓

2) write auxiliary methods/ functions

↓

3) Overwrite Nemo5 methods

↓

4) Bind methods to Nemo5 Python Interface

Python script

./input_python/input_python.py

import numpy as np

1) Import Libraries

def do_solve (self):

3) Overwrite Nemo5 methods

```
if self.check_option( "schroedinger" ):
  solver_schroedinger_str = self.get_option( "schroedinger" )
  solver_schroedinger = self.find_simulation( solver_schroedinger_str )

  if solver_schroedinger is not None :
    Num = 10;
    Fitness = np.zeros(shape=(Num,Num))
    list_valuesi = range(Num)
    list_valuesj = range(Num)
    for i in list_valuesi :
      for j in list_valuesj :

        materials = solver_schroedinger.get_materials()

        for m in materials:
          m.set_option('Bands:TB:sp3d5s*star:param_set', 'param_Boykin')
          m.set_option('Bands:TB:sp3d5s*star:param_Boykin:V_S_S_Sigma_H_Si', str(-20+i*2))
          m.set_option('Bands:TB:sp3d5s*star:param_Boykin:V_S_P_Sigma_H_Si', str(0+j*3))

        solver_schroedinger.set_option('

        solver_schroedinger.reinit()
        solver_schroedinger.reinit_materials()
        solver_schroedinger.solve()
        solver_schroedinger.output()
        # post process
        energy = np.loadtxt('Si_' + str(i)+ '_' + str(j)+'_energies.dat','float')
        Ref_energy = [-1,-0.7,-0.5,0]
        TB_energy = energy[[25,26,27,28]]
        Dif = TB_energy-Ref_energy
        Fitness[i,j] = np.dot(Dif,Dif)
  else :
    raise NameError('solver_schroedinger doesnt exists')
  print Fitness
  np.savetxt('Fitness.dat',Fitness)
else :
  raise NameError('schroedinger parameter required')
```

spaces for indentation matter in python

do_solve
do_init
do_reinit
do_output
get_data_...

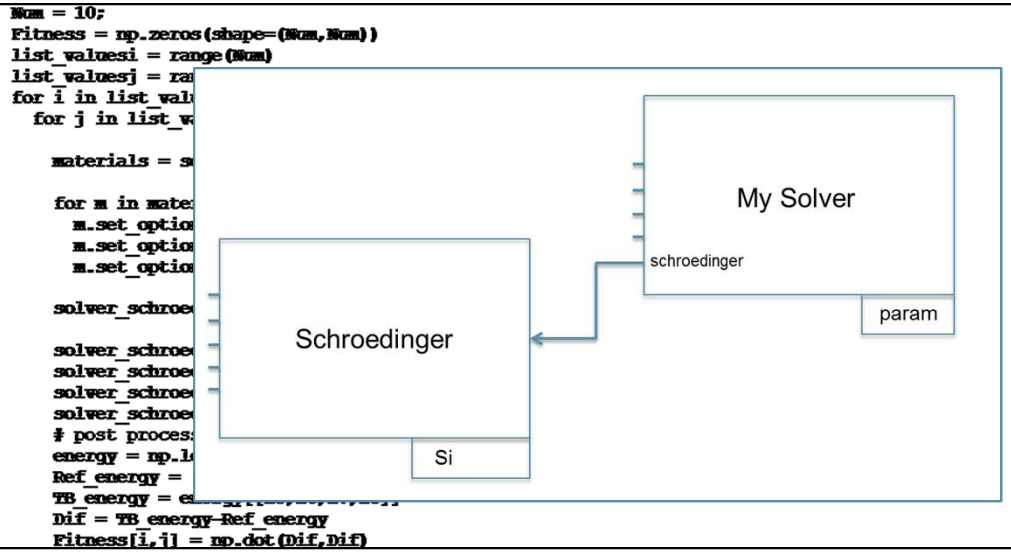
PythonSolverInterface.do_solve = do_solve

4) Bind methods to Nemo5 Python Interface

do_solve

./input_python/input_python.py

```
if self.check_option( "schroedinger" ):
    solver_schroedinger_str = self.get_option( "schroedinger" )
    solver_schroedinger = self.find_simulation( solver_schroedinger_str )
    if solver_schroedinger is not one :
```



In the nemo5 input.in :

```
solver
  {
    name = d1
    type = PythonSolver
    schroedinger = Si
    …
  }
```

```
    else :
        raise NameError('solver_schroedinger doesn't exists')
else:
    raise NameError('schroedinger parameter required')
```

do_solve

./input_python/input_python.py

```
if self.check_option( "schroedinger"
  solver_schroedinger_str = self.get_
  solver_schroedinger = self.find_si

  if solver_schroedinger is not None
```

100 samples

Reconfigure Schroedinger solver

```
for i in range(10) :
  for j in range(10) :
    materials = solver_schroedinger.get_materials()
    for m in materials:
      m.set_option('Bands:TB:sp3d5sstar:param_set', 'param_Boykin‘)
      m.set_option('Bands:TB:sp3d5sstar:param_Boykin:V_S_S_Sigma_H_Si', \
      str(-20+i*0.2))
      m.set_option('Bands:TB:sp3d5sstar:param_Boykin:V_S_P_Sigma_H_Si', \
      str(0+j*0.3))
    solver_schroedinger.set_option('output_file_suffix', '_' + str(i)+ '_' + str(j))
```

Set parameters for each material

```
        solver_schroedinger.reinit()
        solver_schroedinger.reinit_material_properties()
        solver_schroedinger.solve()
        solver_schroedinger.output()
        # post process
        energy = np.loadtxt('Si_' + str(i)+ '_' + str(j)+'_energies.dat','float')
        Ref_energy = [-1,-0.7,-0.5,0]
        TB_energy = energy[[25,26,27,28]]
        Dif = TB_energy-Ref_energy
        Fitness[i,j] = np.dot(Dif,Dif)
    else :
      raise NameError('solver_schroedinger doesnt exists')
    print Fitness
    np.savetxt('Fitness.dat',Fitness)
  else :
    raise NameError('schroedinger parameter required')
```

Specify the name of output file for each set of parameters

do_solve

./input_python/input_python.py

Re-initialize Schroedinger solver

```
if self.check_option( "schroedinger" ):
  solver_schroedinger_str = self.get_option( "schroedinger" )
  solver_schroedinger = self.find_simulation( solver_schroedinger_str )

  if solver_schroedinger is not None :
    Num = 10;
    Fitness = np.zeros(shape=(Num,Num))
    list_valuesi = range(Num)
    list_valuesj = range(Num)
    for i in list_valuesi :
      for j in list_valuesj :

        materials = solver_schroedinger.get_materials()

        for m in materials:
          m.set_option('Bands:TB:sp3d5sstar:param_set', 'param_Boykin')
          m.set_option('Bands:TB:sp3d5sstar:param_Boykin:V_S_S_Sigma_H_S
          m.set_option('Bands:TB:sp3d5sstar:param_Boykin:V_S_P_Sigma_H_S

        solver_schroedinger.set_option('output_file_suffix', '_' + str(i
```

Re-initialize the options in the solver;
Re-initialize the parameters of the material;

```
solver_schroedinger.reinit()
solver_schroedinger.reinit_material_properties()
solver_schroedinger.solve()
solver_schroedinger.output()
```

```
        energy = np.loadtxt('Si_' + str(i)+ '_' + str(j)+'_energies.dat
        Ref_energy = [-1,-0.7,-0.5,0]
        TB_energy = energy[[25,26,27,28]]
        Dif = TB_energy-Ref_energy
        Fitness[i,j] = np.dot(Dif,Dif)
  else :
    raise NameError('solver_schroedinger doesnt exists')
  print Fitness
  np.savetxt('Fitness.dat',Fitness)
else :
  raise NameError('schroedinger parameter required')
```

Generate outputs

do_solve

```
if self.check_option( "schroedinger" ):
  solver_schroedinger_str = self.get_option( "schroedinger" )
  solver_schroedinger = self.find_simulation( solver_schroedinger_str )

  if solver_schroedinger is not None :
    Num = 10;
    Fitness = np.zeros(shape=(Num,Num))
    list_valnesi = range(Num)
    list_valnesj = range(Num)
    for i in list_valnesi :
      for j in list_valnesj :

        materials = solver_schroedinger.get_materials()

        for m in materials:
          m.set_option('Bands:TB:sp3d5sstar:param_set', 'param_Boykin')
          m.set_option('Bands:TB:sp3d5sstar:param_Boykin:V_S_S_Sigma_H_Si', str(-20+i*2))
          m.set_option('Bands:TB:sp3d5sstar:param_Boykin:V_S_P_Sigma_H_Si', str(0+j*3))

        solver_schroedinger.set_option('output_file_suffix', '_' + str(i)+ '_' + str(j))

        solver_schroedinger.reinit()
        solver_schroedinger.reinit_material_properties()
        solver_schroedinger.solve()
        solver_schroedinger.output()
        # post process
```

Fitness

Read nemo5 results

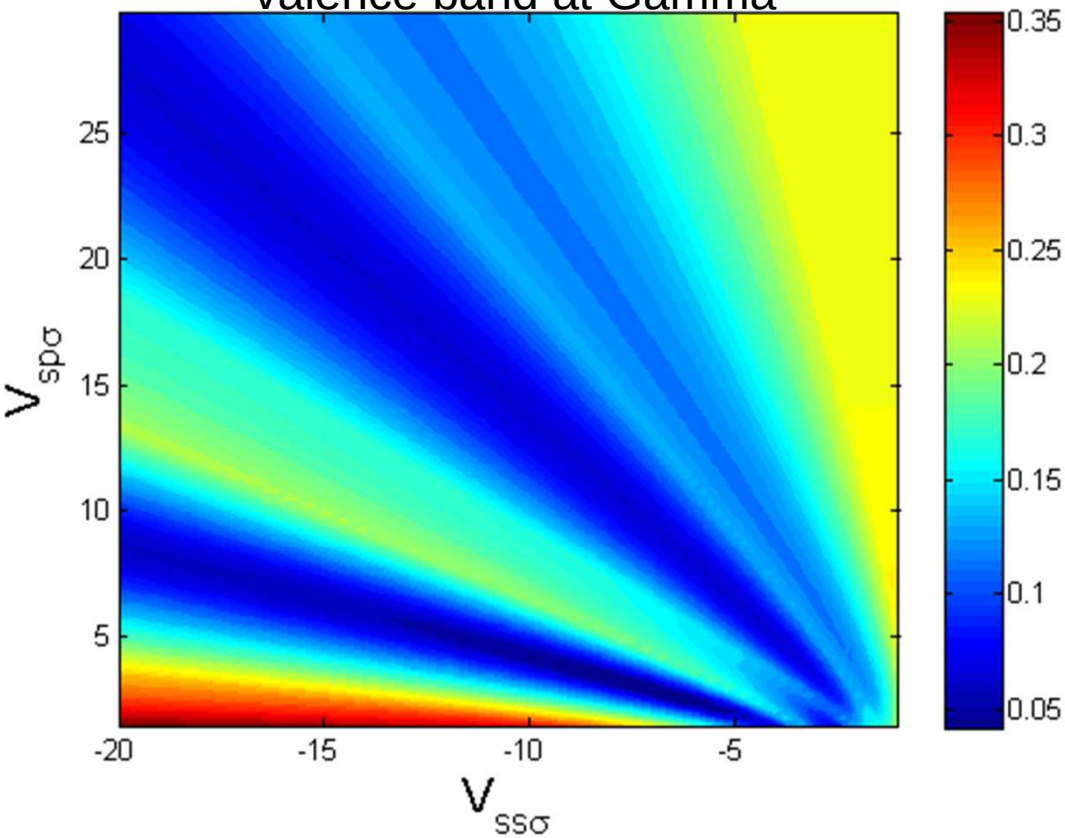
```
energy = np.loadtxt('Si_' + str(i)+ '_' + str(j)+'_energies.dat','float')
Ref_energy = [-1,-0.7,-0.5,0]
TB_energy = energy[[25,26,27,28]]
Dif = TB_energy-Ref_energy
Fitness[i,j] = np.dot(Dif,Dif)
np.savetxt('Fitness.dat',Fitness)
```

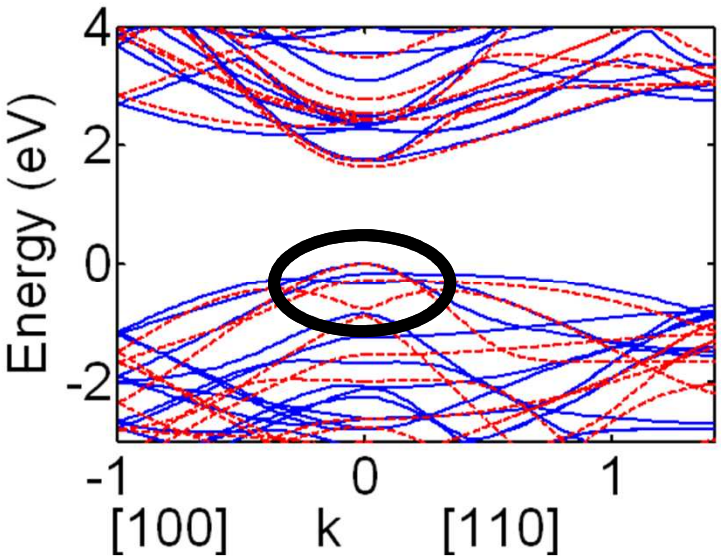
Compare selected bands with reference bands

```
  else :
    raise NameError('solver_schroedinger doesnt exists')
  print Fitness
  np.savetxt('Fitness.dat',Fitness)
else :
  raise NameError('schroedinger parameter required')
```

Results of Example

Average error of the highest 4 valence band at Gamma





Band structure of Si UTB:
Blue lines: DFT results
Red lines: nemo5 results

100 different values for Vspσ_H_Si and Vssσ_H_Si →10000 samples
DFT result is generated by ABINIT

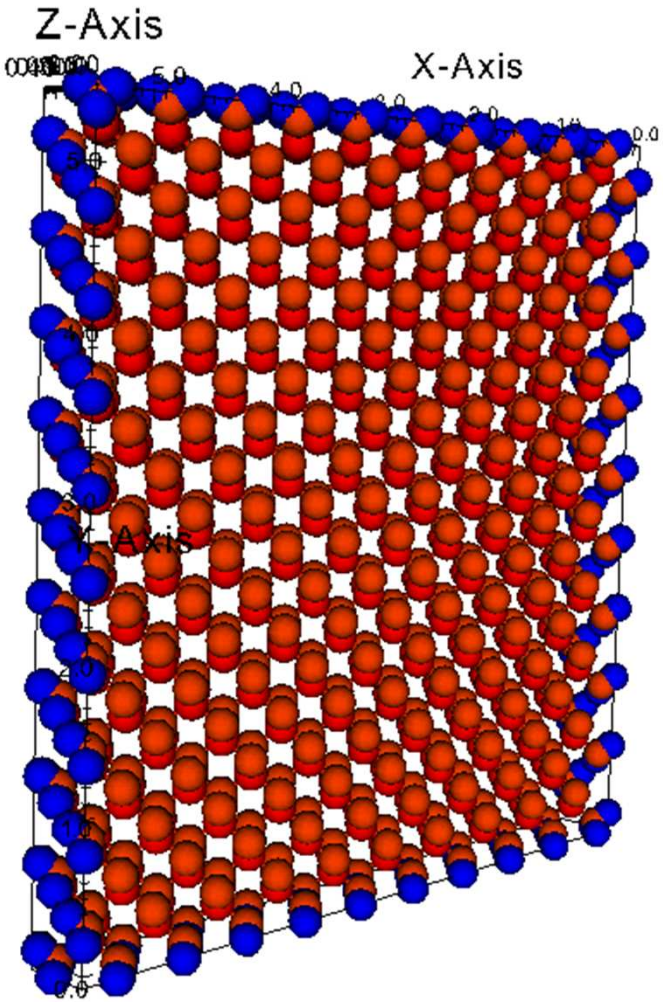
step0.in

~/public_samples/Python_solver/Python_tutorial/step0.in

- GaAs Nanowire

- Solvers:
  - NonlinearPoisson
  - Schroedinger
  - Structure

```
Region{
        shape = cuboid
        region_number = 1
        priority = 1
        min = (-1.5,-1.5,-1.5)
        max = (5.5,5.5,2.5)
}
```



step0.in

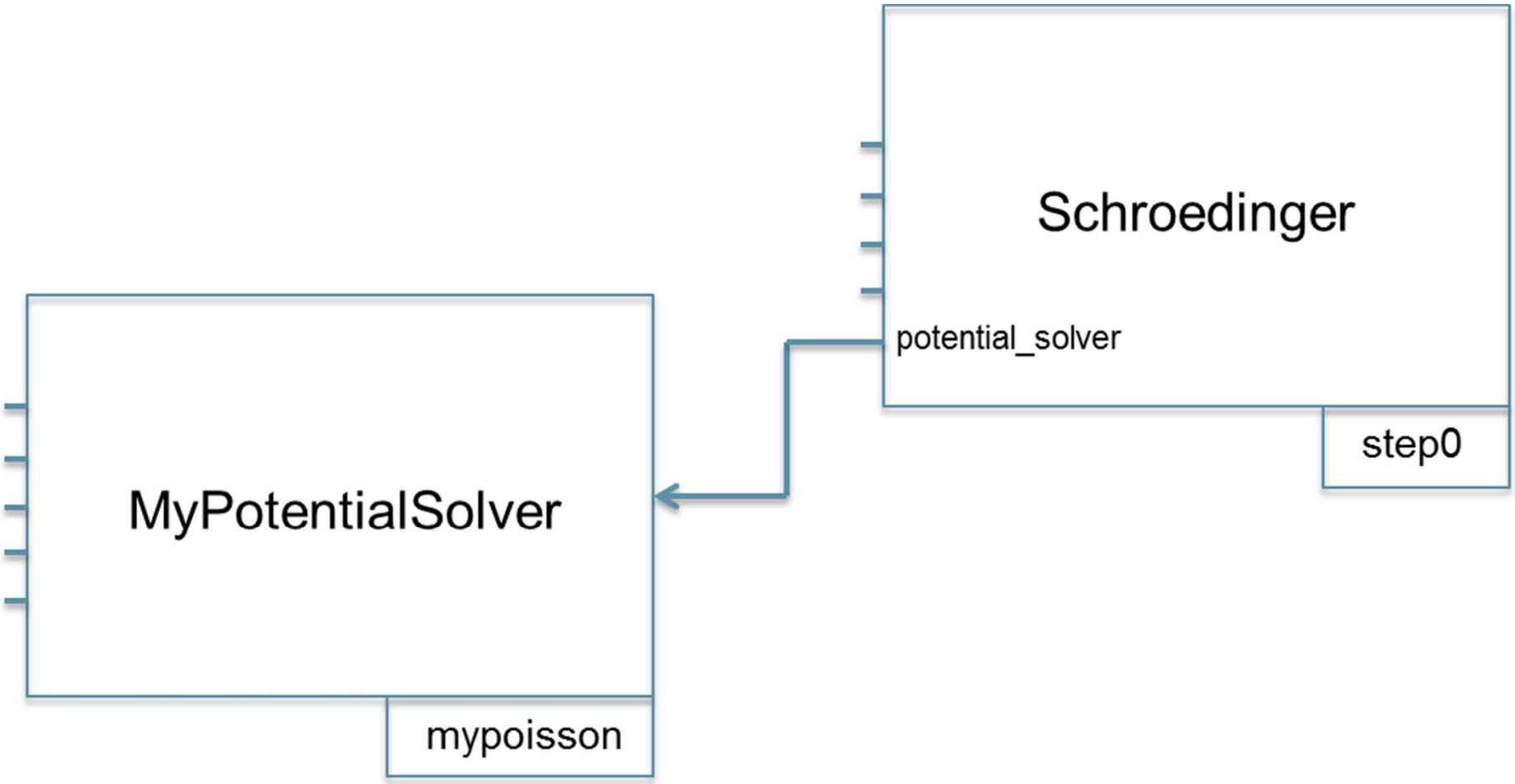
~/public_samples/Python_solver/Python_tutorial/step0.in

```
…
solver {
    name       = poisson
    type       = NonlinearPoisson
    domain     = continuum
    …
}
solver {
    name           = step0
    type           = Schroedinger
    domain         = atomic_structure
    active_regions = (1)
    job_list = (passivate_H, calculate_band_structure)
    output = (energies,eigenfunctions_Point3D, eigenfunctions_VTK)
    potential_solver = poisson
    …
}
…
```

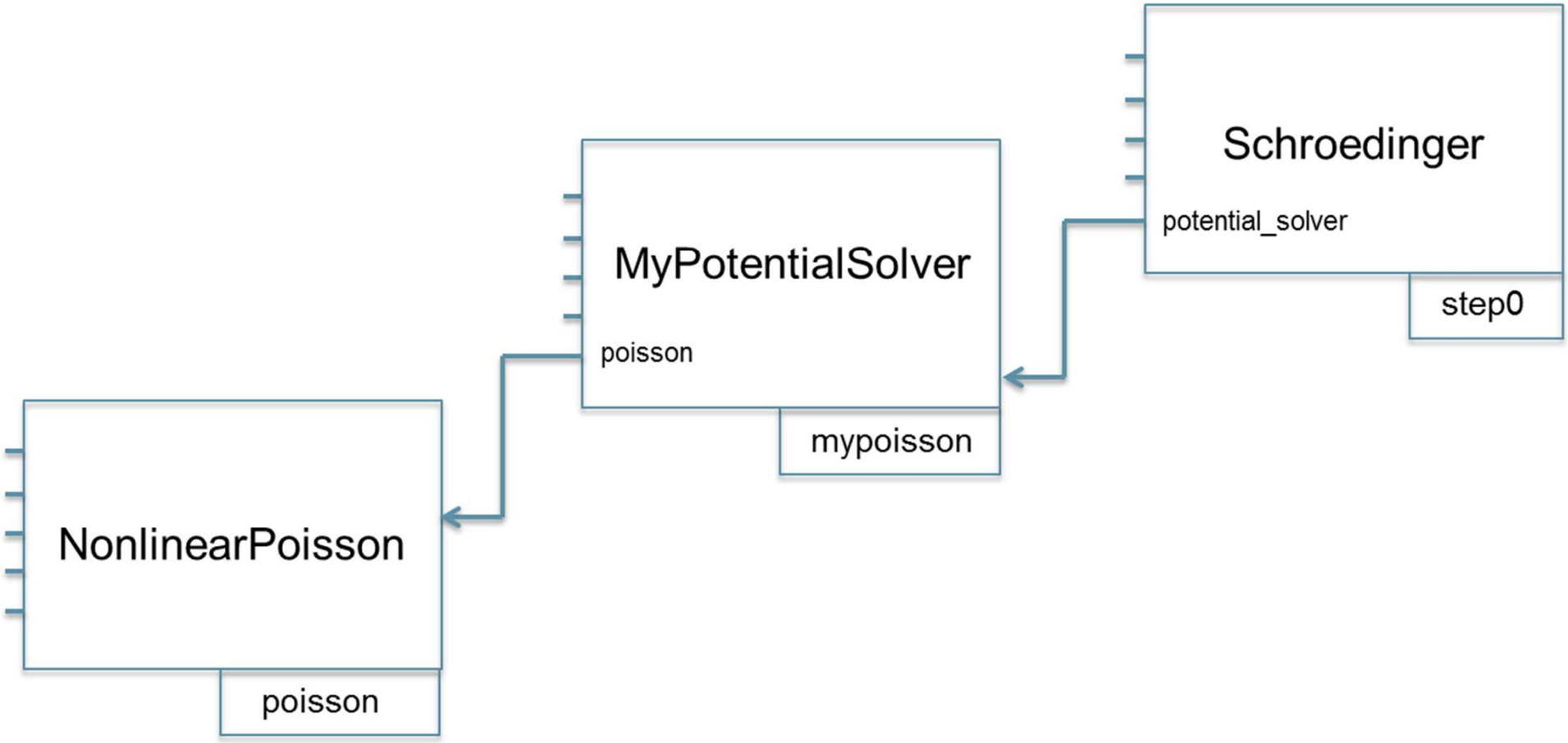
VTK Format

3D Format

Solvers Interaction



Solvers Interaction



step1.in

~/public_samples/Python_solver/Python_tutorial/step1.in

```
solver {
  name = poisson
  type = NonlinearPoisson
  domain = continuum
  …
}
```

```
solver {
  name = pythonpoisson
  type = PythonSolver
  domain = atomic_structure
  poisson = poisson
  python_solver = ./step1.py
}
```

```
solver {
  name = step1
  type = Schroedinger
  domain = atomic_structure
  potential_solver = pythonpoisson
  …
}
```

PythonSolvers Interaction

Data Flow

Call methods:
get_data(…)
get_multidata(…)

step1.py

Rewrite method:
get_..._data_...(…)

~/public_samples/Python_solver/Python_tutorial/step1.py

1) Import Libraries

```
import math
import random
```

- import numpy
- import scipy
- import vtk
- from math import *

- import sys
- sys.path.append('…/my_path/')

~/public_samples/Python_solver/Python_tutorial/step1.py

2) write auxiliary methods/ functions

```
#returns a perturbation [-0.05, 0.05]
def random_perturbation ():
  return (random.random()-0.5)/10

#pot = Atom Id
#ori_pot = Original Potential
#new_pot = Perturbated Potential
def print_potential (pot, ori_pot, new_pot):
  txt = ""
  txt += "Potential[" + str(pot) + "]:" + str(ori_pot)
  txt += " - " + str(new_pot)
  print txt
```

- Functions

```
def name_function(…):
  …
```

- Methods

```
def name_method( self, …):
  …
```

~/public_samples/Python_solver/Python_tutorial/step1.py

3) Overwrite Nemo5 methods

```
def do_init(self):
  self.potentials = {}
  if self.check_option( "poisson" ):
    poisson_str = self.get_option( "poisson" )
    self.poisson = self.find_simulation( poisson_str )
  else:
    self.poisson = None
```

- do_init(self):
- do_reinit(self):
- do_solve(self):
- do_output(self):

- get_..._data_... (self, …)

bool check_option(string opt) -> was ‘opt’ defined in the input deck?
string get_option(string opt) -> returns input option ‘opt’
N5_Solver find_simulation (string solver_name) -> returns a ‘pointer’ to the instance of solver_name

~/public_samples/Python_solver/Python_tutorial/step1.py

3) Overwrite Nemo5 methods

```
void Simulation::get_data(const string& variable,
                const vector<double>& in_data, std::vector<double>&  out_data)
```

```
def get_list_uint_data_list_double ( self, variable, in_data ):
 out_data = []
  if self.poisson is not None:
    self.poisson.get_multidata(params, in_data, result, "list<uint>:list<double>")
    for i, v in enumerate(in_data[0]):
      original_potential = out_data[0][i]
      if v in self.potentials:
        out_data[0][i] = self.potentials[v]
      else :
        out_data[0][i] += random_perturbation()
        self.potentials[v] = result[0][i]
      print_potential(v, original_potential, result[0][i])
  return out_data
```

Given a list of unsigned integer, this returns a list of doubles

step1.py

~/public_samples/Python_solver/Python_tutorial/step1.py

4) Bind methods to Nemo5 Python Interface

```
import math
import random
def do_init(self):
  self.potentials = {}
  if self.check_option( "poisson" ):
    poisson_str = self.get_option( "poisson" )
    self.poisson = self.find_simulation( poisson_str )
  else:
    self.poisson = None

def random_perturbation ():
  return (random.random()-0.5)/100

#pot = Atom Id
#ori_pot = Original Potential
#new_pot = Perturbated Potential
def print_potential (pot, ori_pot, new_pot):
  txt = ""
  txt += "Potential[" + str(pot) + "]:" + str(ori_pot)
```

```
PythonSolverInterface.do_init = do_init
PythonSolverInterface.get_list_uint_data_list_double = get_list_uint_data_list_double
```

```
    self.poisson.get_multidata(params, variables, result, "list<uint>:list<double>")
    for i, v in enumerate(variables[0]):
      original_potential = result[0][i]
      if v in self.potentials:
        result[0][i] = self.potentials[v]
      else :
        result[0][i] += random_perturbation()
        self.potentials[v] = result[0][i]
      print_potential(v, original_potential, result[0][i])
  return result

PythonSolverInterface.do_init = do_init
PythonSolverInterface.get_list_uint_data_list_double = get_list_uint_data_list_double
```

Run step1_small.py

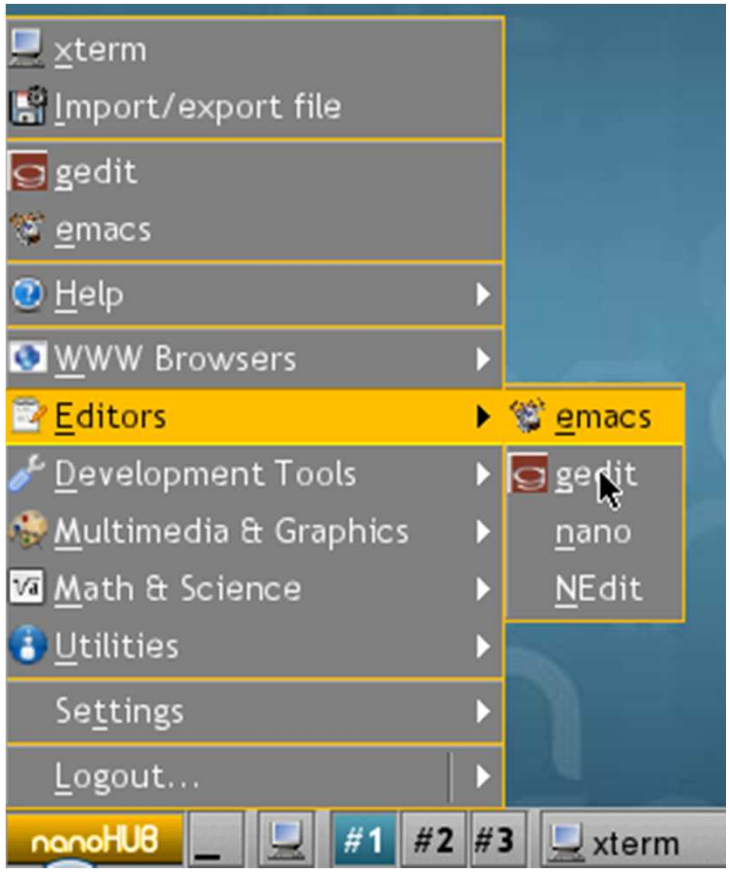
```
submit -v coates \
    -i /apps/share64/nemo/examples/current/materials/all.mat \
    -i ./step1_small.in \
    -i ./step1.py \
    nemo-r7962 step1_small.in
```

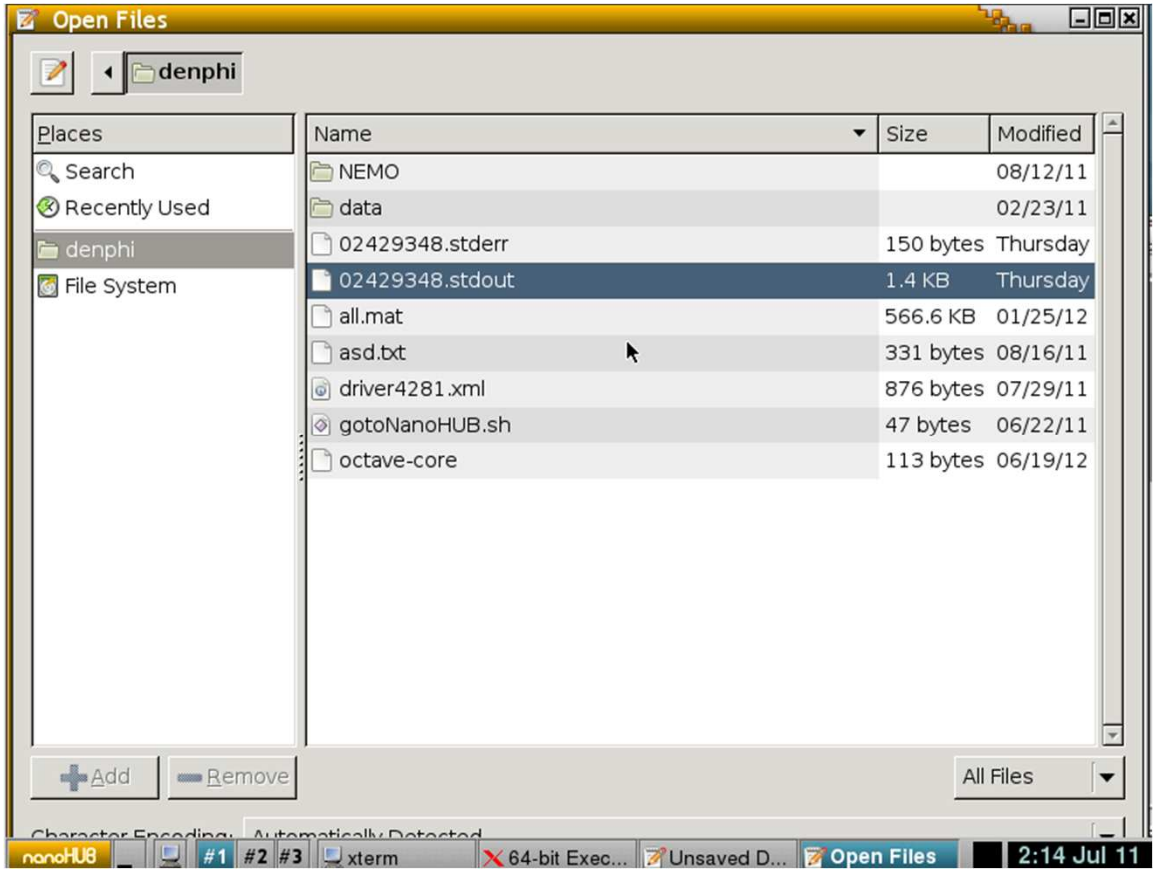
Open file XXXXXX.stdout in a text editor (emacs, vi, gedit, …)

Search line (~320):
“[Schroedinger] band dispersion loop at: 0%, k=(0,0,0)”

step1_small.py

Search line (~320): “[Schroedinger] band dispersion loop at: 0%, k=(0,0,0)”





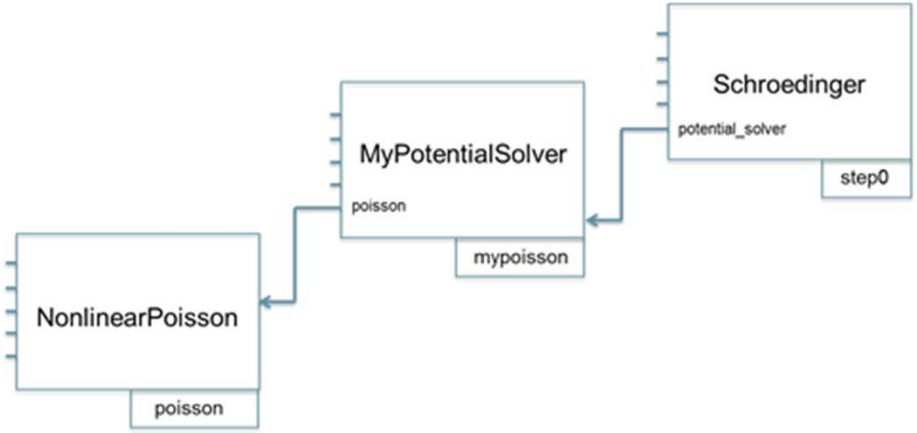
Run step1.py

submit -v coates -i ./all.mat -i ./step1.in -i ./step1.py nemo-rXXXX step1.in

Atom ID

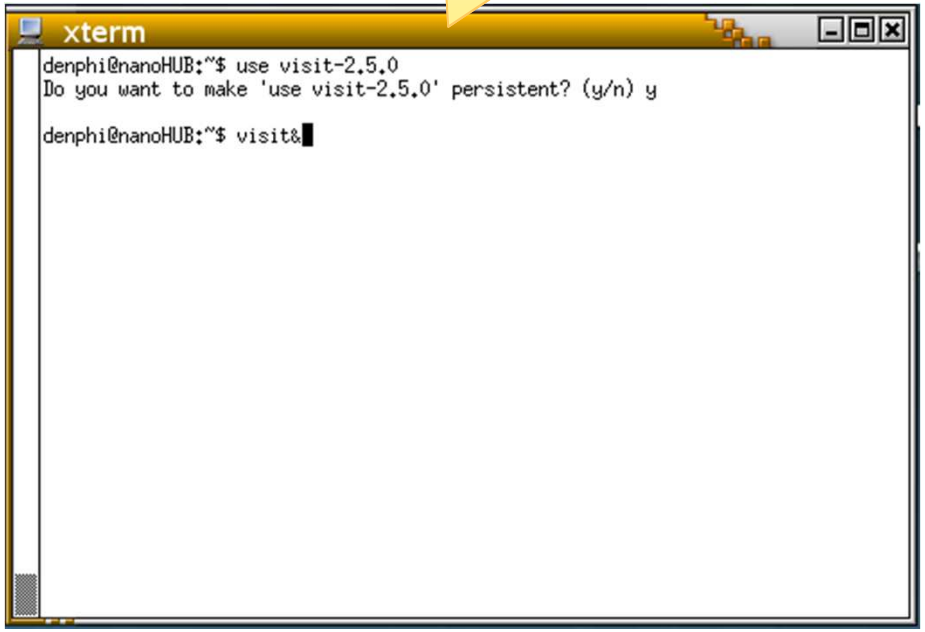
```
Potential[123]:-0.711241071429 - -0.71468656041
Potential[124]:-0.711241071429 - -0.708356163895
Potential[125]:-0.711241071429 - -0.707456438156
Potential[126]:-0.711241071429 - -0.713384328581
Potential[127]:-0.711241071429 - -0.7121059371
Potential[128]:-0.711241071429 - -0.710035799555
Potential[129]:-0.711241071429 - -0.711474255723
Potential[130]:-0.711241071429 - -0.711047785595
Potential[131]:-0.711241071429 - -0.712618402222
Potential[132]:-0.711241071429 - -0.711746789194
Potential[133]:-0.711241071429 - -0.715607662224
Potential[134]:-0.711241071429 - -0.707339627721
Potential[135]:-0.711241071429 - -0.71571539359
Potential[136]:-0.711241071429 - -0.715568649722
Potential[137]:-0.711241071429 - -0.716199576789
```

After Before

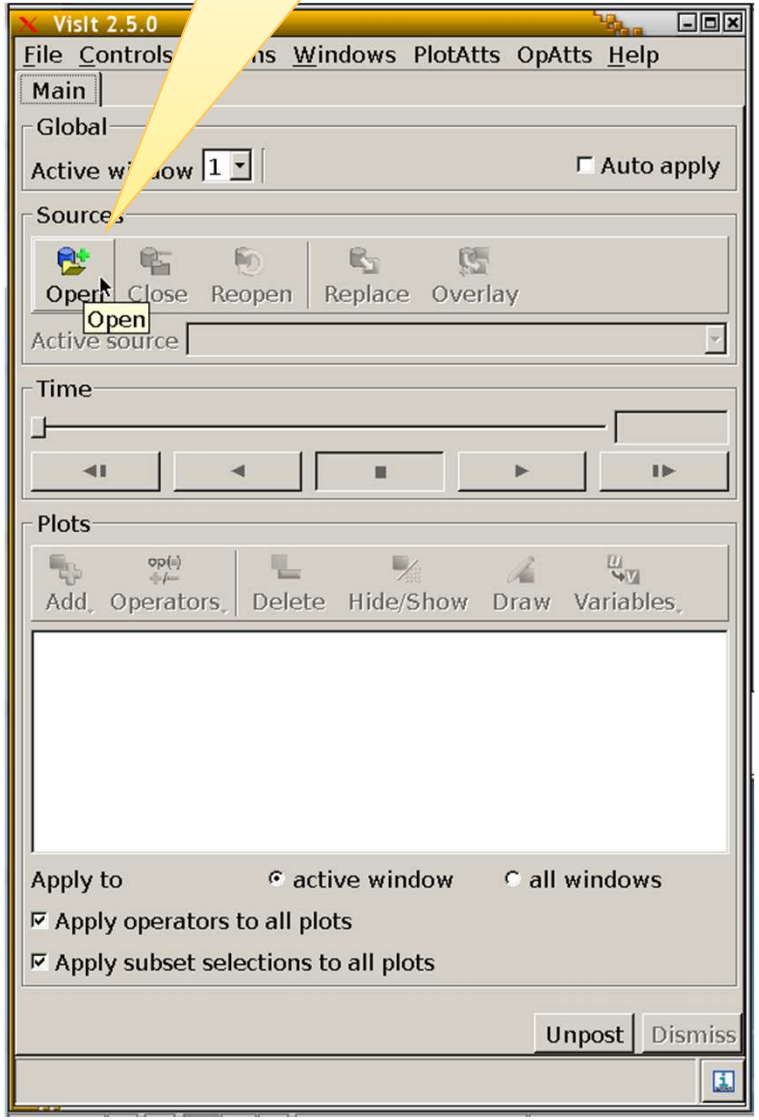



```
def print_potential (pot, ori_pot, new_pot):
"Potential["+str(pot)+"]:"+str(ori_pot)+"-"+str(new_pot)
```

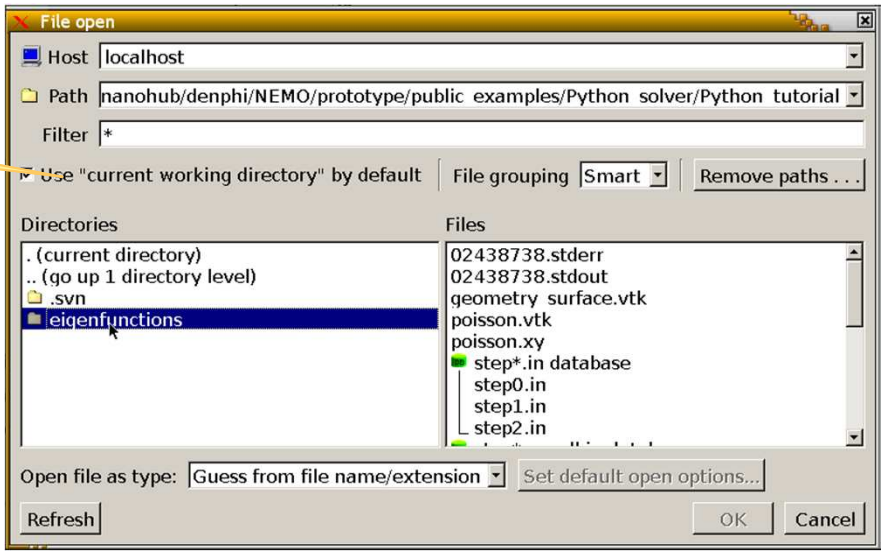
- Load VisIt:
  - use visit-2.5.0 (y)
  - visit &

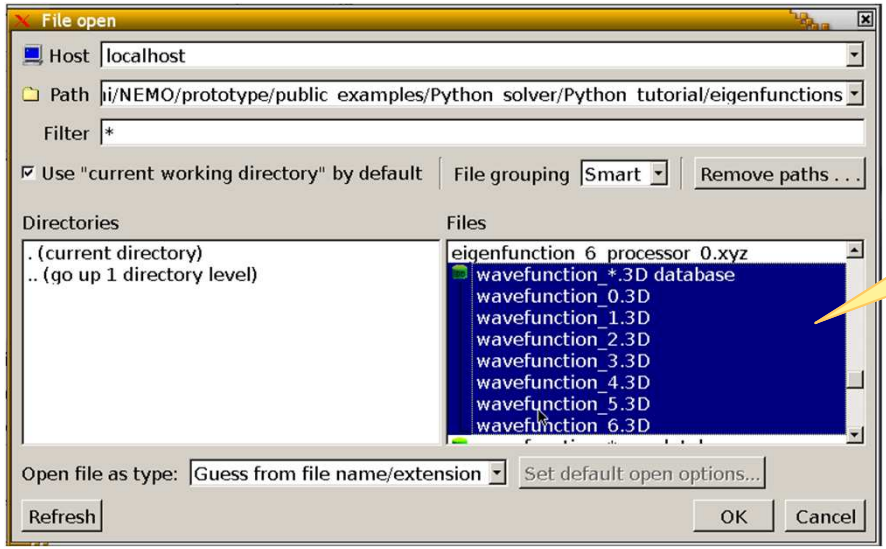


- Open Database



Locate eigenfunctions folder

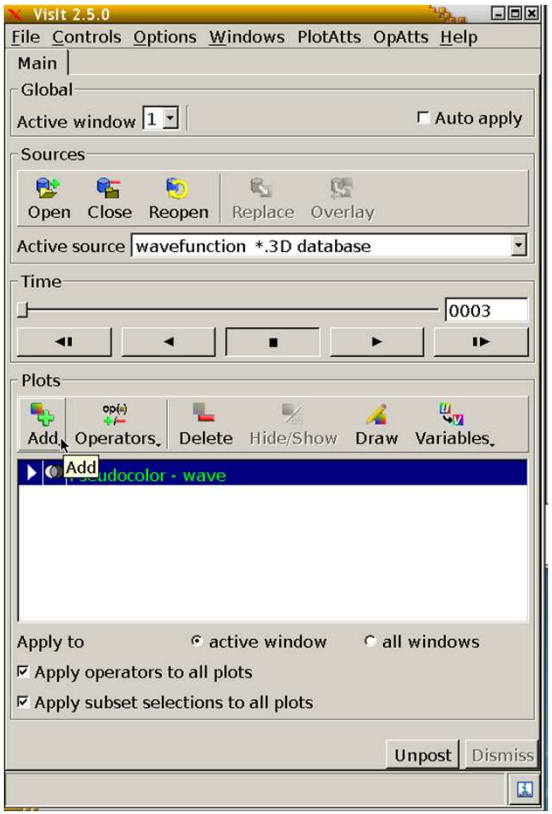


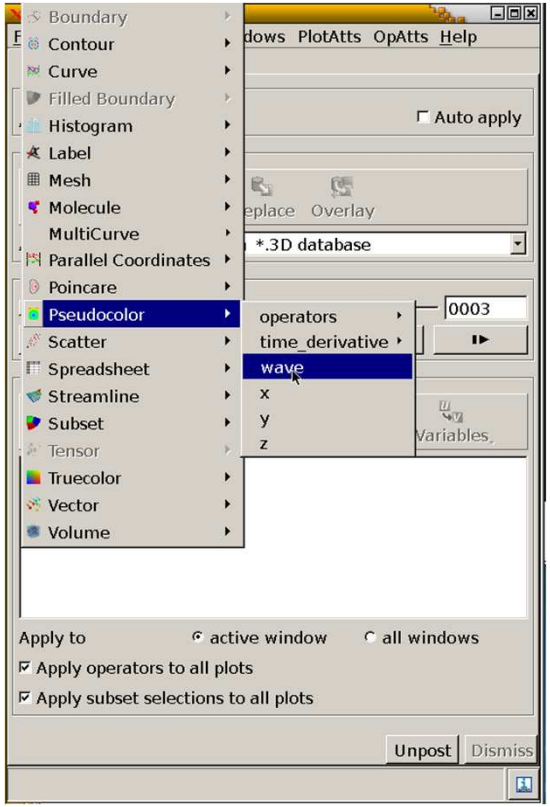


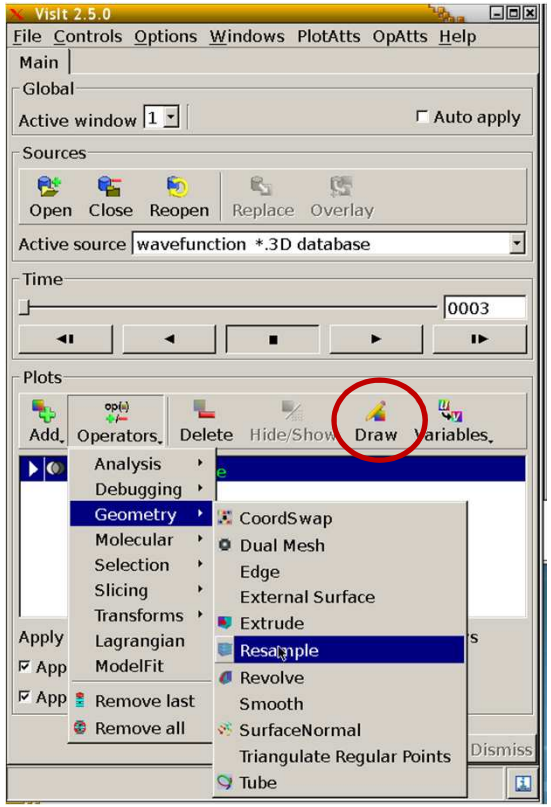
Load wavefunction DB

Visit

1) Add / PseudoColor / Wave
2) Operators / Geometry / Resample
3) Draw

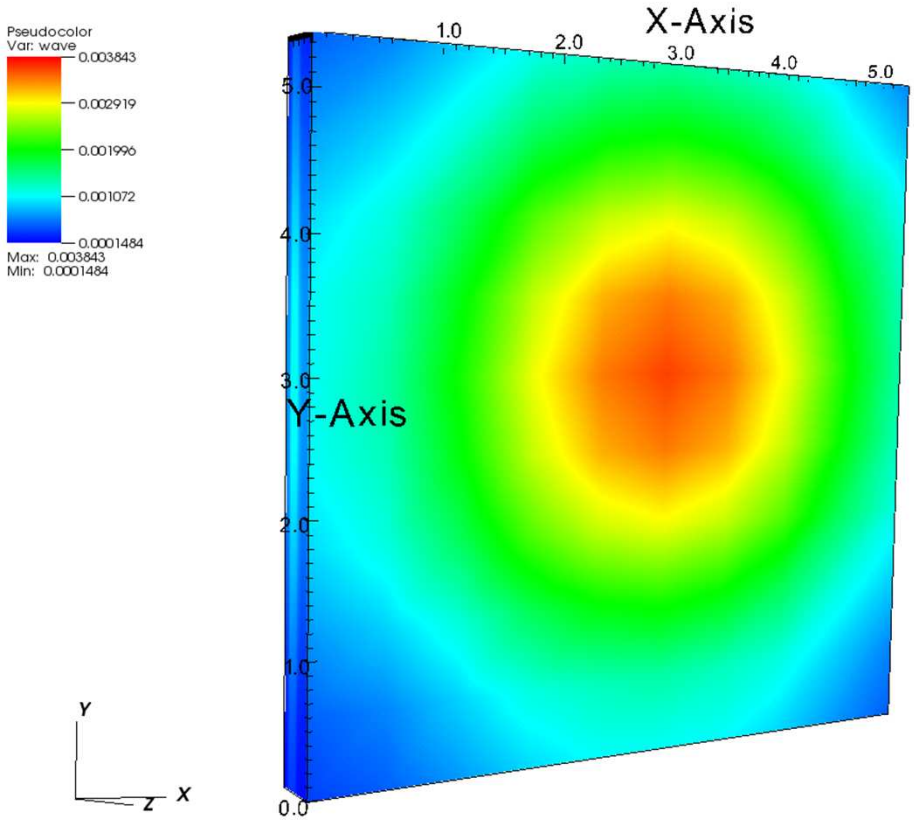




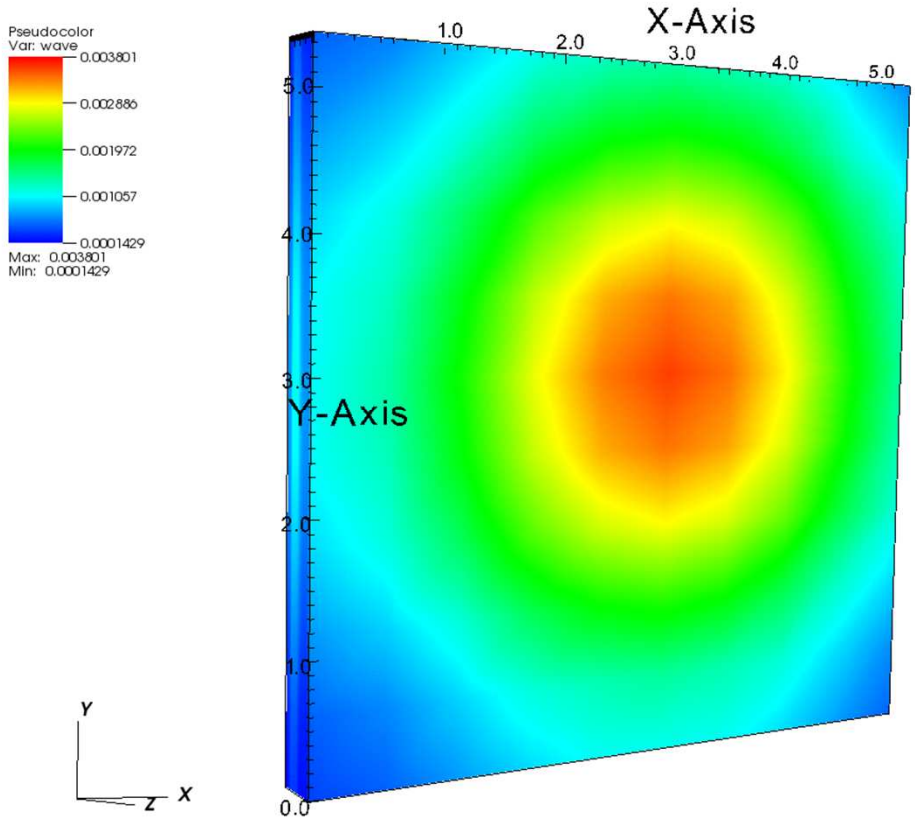


step0 vs step1

eigenfunctions/wavefunction_0.xyz

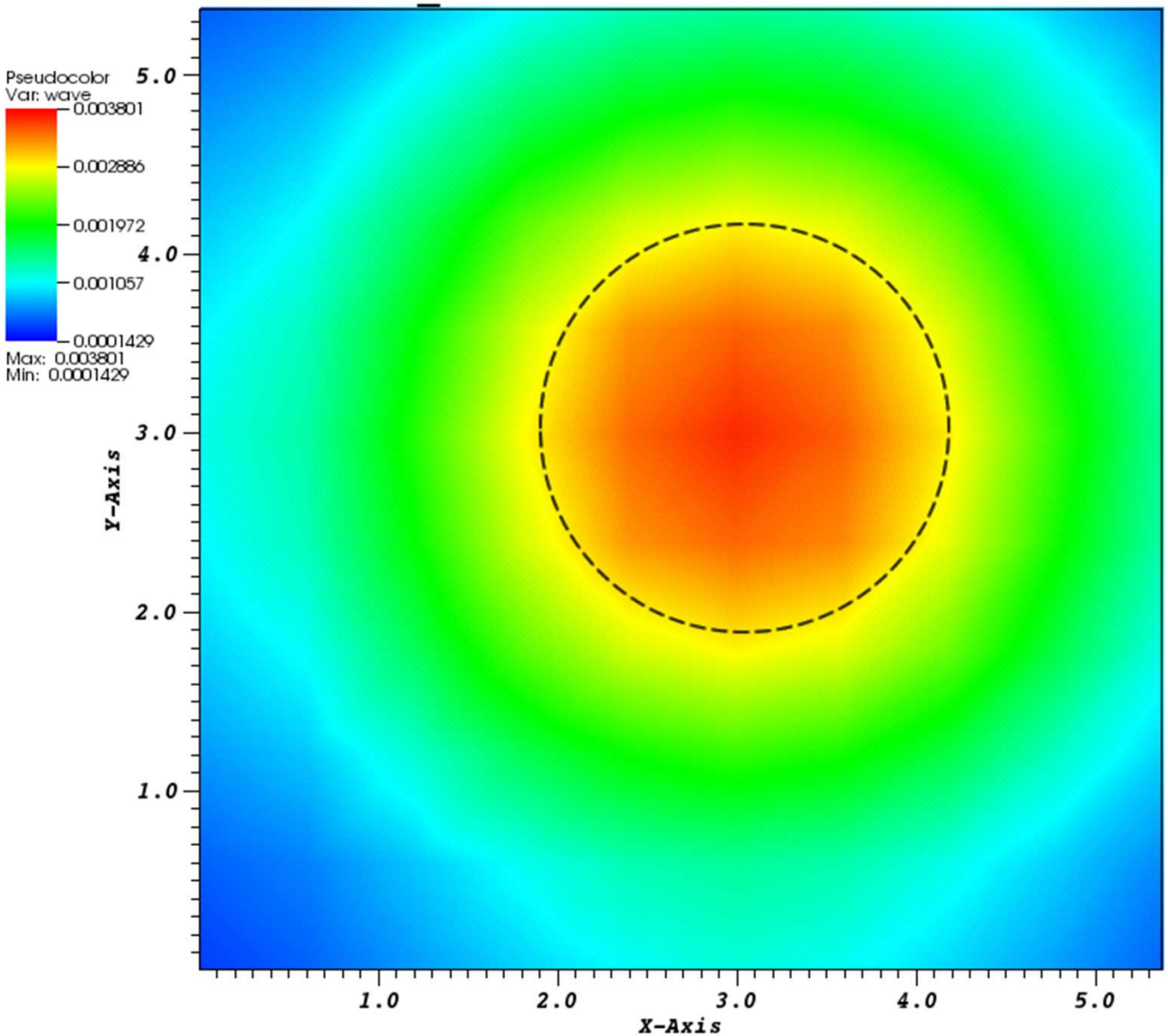


step0.py



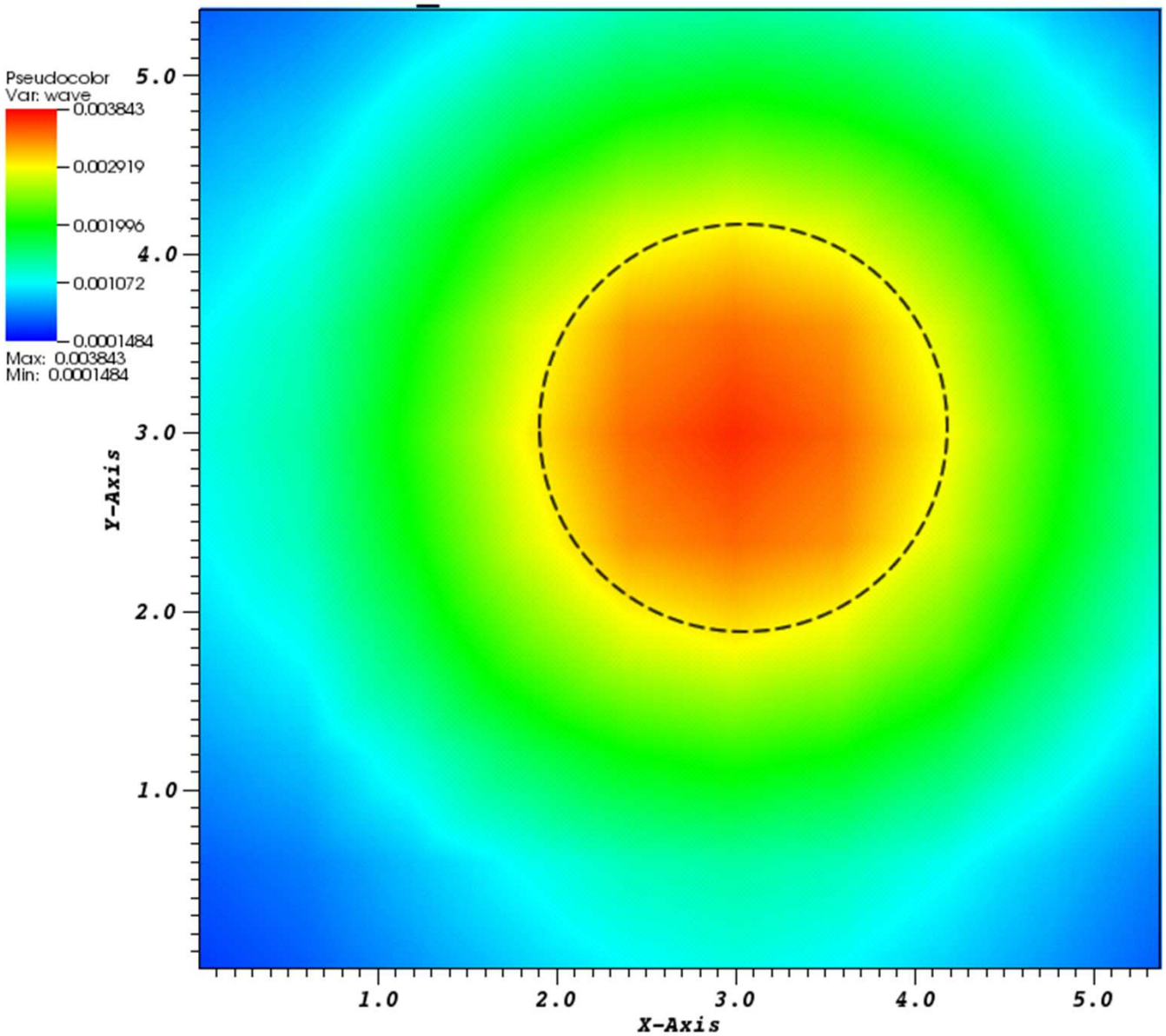
step1.py

step0 vs step1



PURDUE UNIVERSITY

step0 vs step1



PURDUE UNIVERSITY

Exercise

- Edit random_perturbation function. File (step1.py)

- Examples:
  - return (random.random()-0.5)
  - return random.random()
  - return (random.random()-0.5)*2

step1.py (~/NEMO/prototype/public_examples/Python_solver/Pyt...

File Edit View Search Tools Documents Help

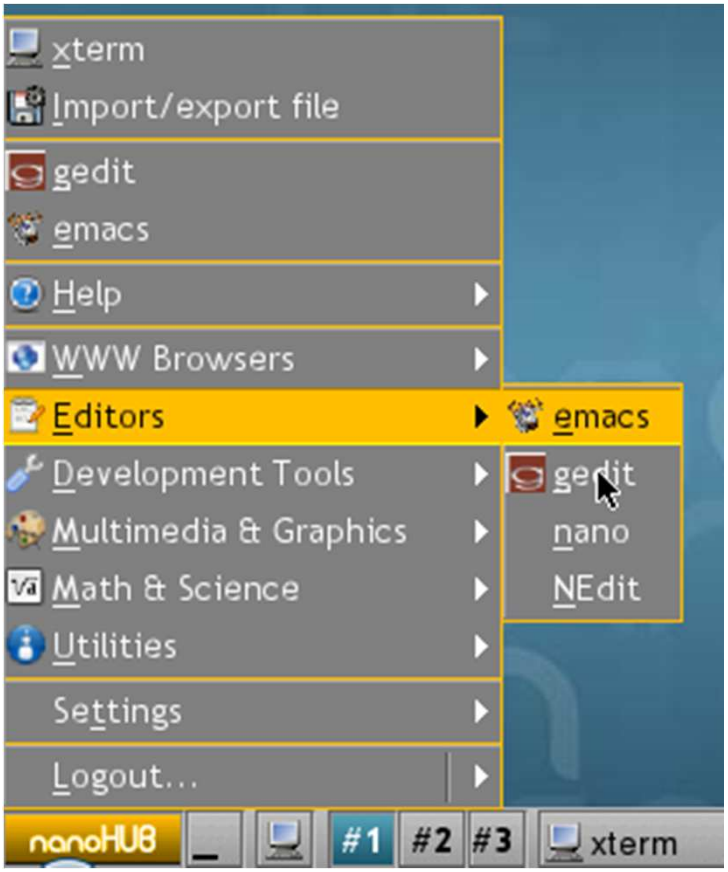
New Open Save Print Undo Redo Cut Copy Paste Find Replace

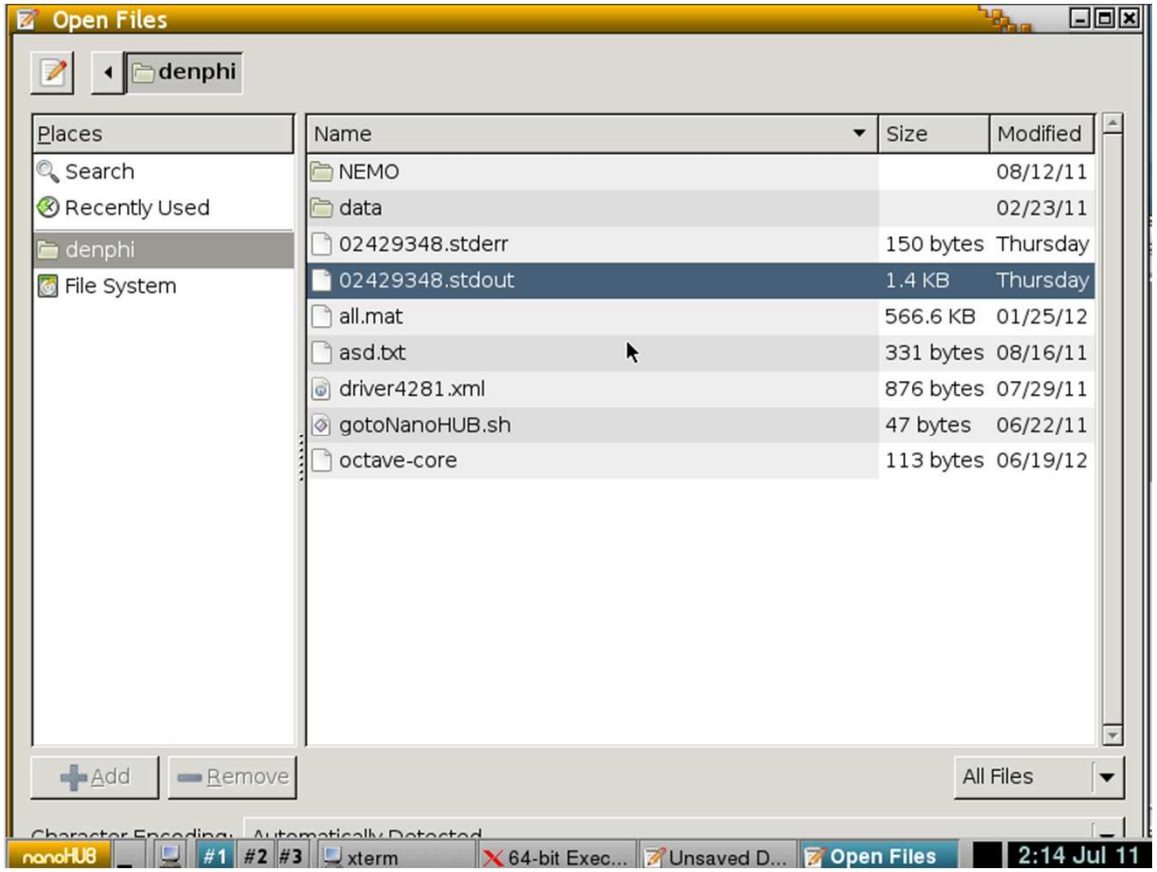
step1.py

```python
import math
import random
def do_init(self):
  self.potentials = {}
  if self.check_option( "poisson" ):
    poisson_str = self.get_option( "poisson" )
    self.poisson = self.find_simulation( poisson_str )
  else:
    self.poisson = None

def random_perturbation ():
  return (random.random()-0.5)/10

#pot = Atom Id
#ori_pot = Original Potential
#new_pot = Perturbated Potential
def print_potential (pot, ori_pot, new_pot):
  txt = ""
  txt += "Potential[" + str(pot) + "]:" + str(ori_pot)
  txt += " - " + str(new_pot)
  print txt

def get_list_uint_data_list_double ( self, params, variables ):
  result = []
  if self.poisson is not None:
    self.poisson.get_multidata(params, variables, result,
"list<uint>:list<double>")
    for i, v in enumerate(variables[0]):
      original_potential = result[0][i]
      if v in self.potentials:
        result[0][i] = self.potentials[v]
      else :
        result[0][i] += random_perturbation()
        self.potentials[v] = result[0][i]
      print_potential(v, original_potential, result[0][i])
  return result
```

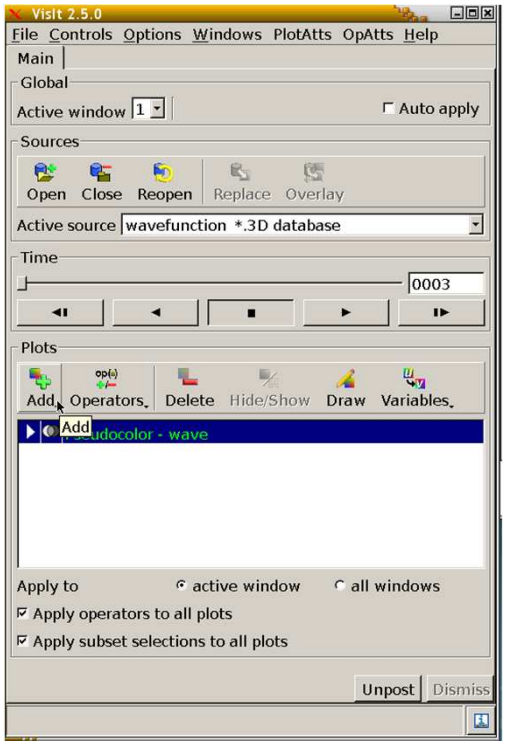
Python | Tab Width: 8 | Ln 22, Col 3 | INS

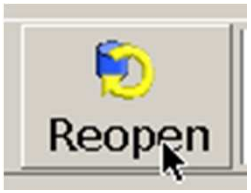
step1_small.py





```
submit -v coates \
    -i /apps/share64/nemo/examples/current/materials/all.mat \
    -i ./step1_small.in \
    -i ./step1.py \
    nemo-r7962 step1_small.in
```

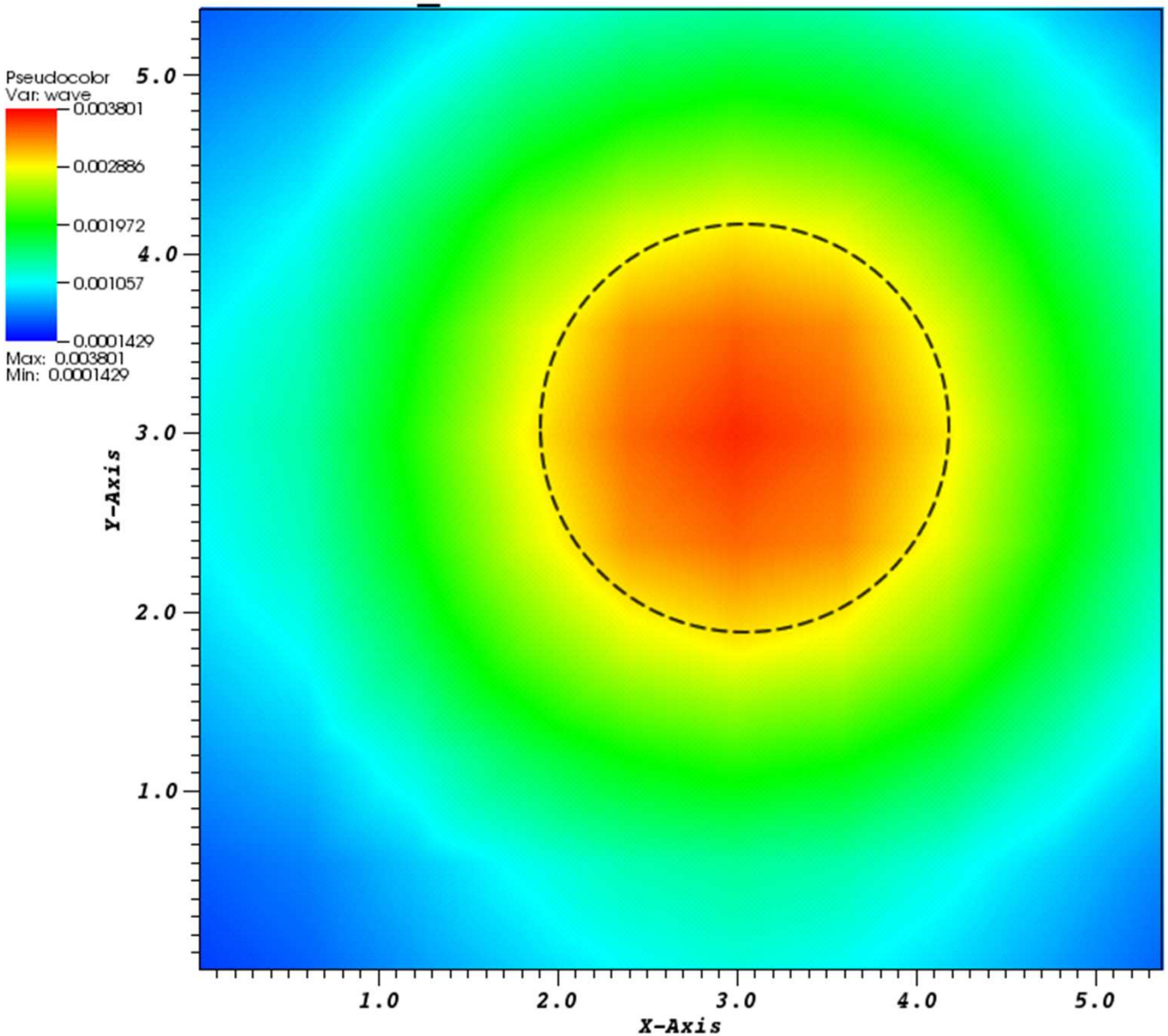




Reload database

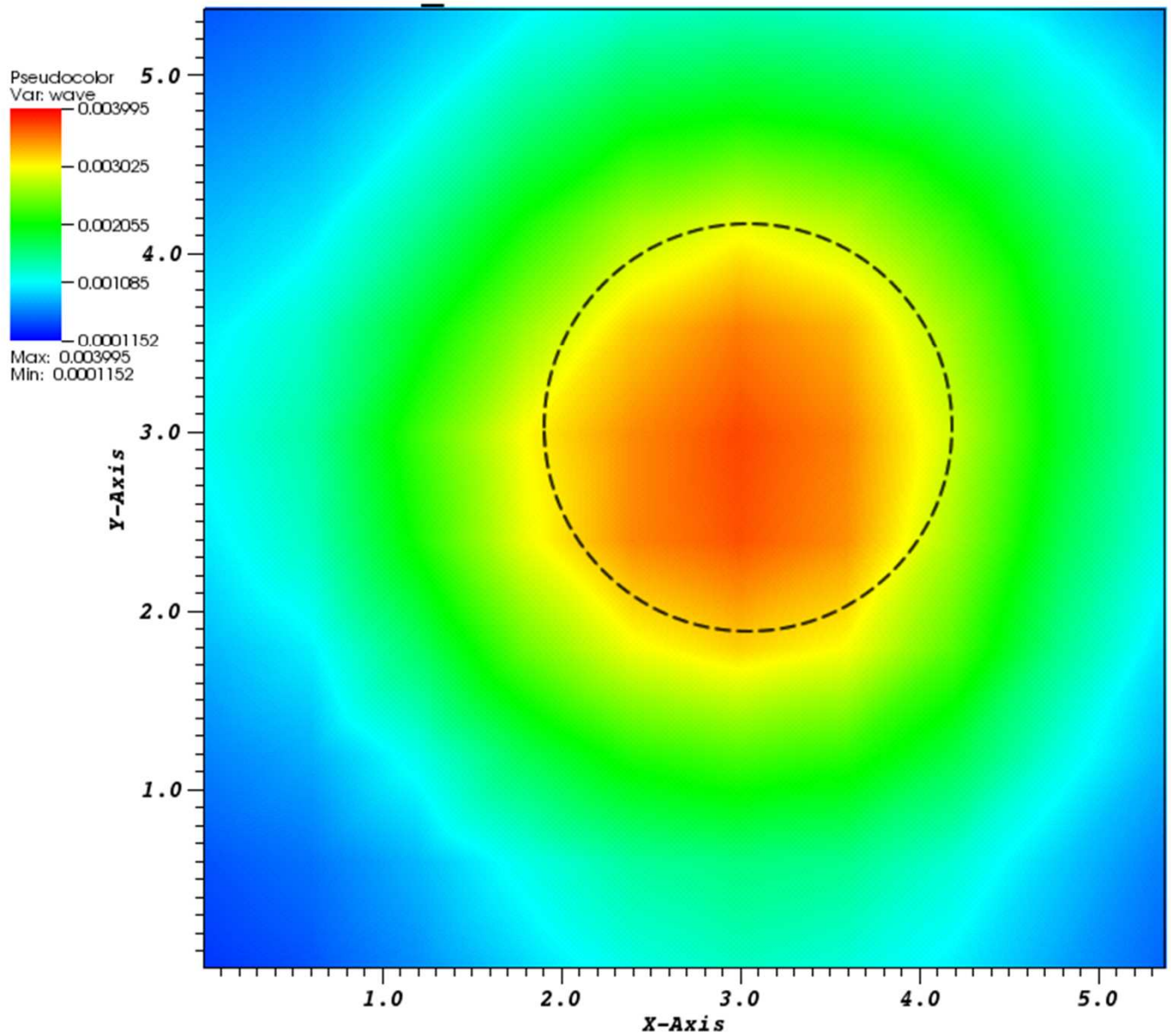
Visualize next Wavefunction

step0 vs step1



PURDUE UNIVERSITY

step0 vs step1



PURDUE UNIVERSITY

~100 Nemo5 features

PythonSolverInterface() //N5_Solver
- str get_name()
- str get_type()
- list get_options()
- bool check_option(str)
- str get_option (str)

- N5_VecDouble/N5_VecComplex()
  - » int get_size()
  - » int get_local_size()
  - » void get_values(list, list)
  - » void set_values(list, list)
  - d_values(list, list)
  - (double/complex)
  - ale(double)
  - _product(N5_Vec*, N5
  - ble_Sequential

- **N5_MatDouble/Complex_Parallel()**
  - » void set_num_nonzeros (int, int, int)
  - » void allocate_memory()
  - » bool is_ready()
  - » void add(int, int, double/complex)
  - » v
  - » c
  - » i
  - » i
  - » v

- N5_Output()
  - » void add_dataset(str name, list values
  - » void remove_dataset(str)
  - » void write_to_file(str)

- N5_Utils (static)
  - » **void db_set_parameter(str, str, s**
  - » **string db_get_parameter(str,str)**

- N5_P(object) // pointer
  - » setValue(object)
  - » .value // object

- N5_MPI (static)
  - » void N5_MPI_Comm_rank(N5_P, int)
  - » void N5_MPI_Comm_size(N5_P, int)
  - » **void N5_MPI_Bcast(object, int**
  - » void N5_MPI_Barrier()
  - » void N5_MPI_COMM_WORLD()

- **N5_Atom()**
  - » str get_atom_type ()
  - » str get_atom_tight_binding()
  - » **.position // list[3]**
  - » .ideal_position // list[3]
  - » .id // int
  - » .proc_id //int
  - » .region // int
  - » .active // bool

N5_Output create_output( )
void init_material_properties()
void reinit_material_properties()

- N5_Material()
  - » str get_name()
  - » list get_options()
  - » bool check_option(str)
  - » str get_option(str)
  - » void set_option(str, str)

- N5_Domain()
  - » list get_domain_options ()
  - » str get_name()
  - » bool if_atomistic()
  - » bool if_intialized()

- N5_AtomisticDomain (N5_Domain)
  - » int num_atoms()
  - » list get_atoms_info()
  - » list get_atoms_info_CPU(int)
  - » int get_size_CPU()

Thanks!

step2.py(add potential due an impurity)

~/public_samples/Python_solver/Python_tutorial/step2.py

Initialization (do_init)

```
def do_init(self):
  self.potentials = {}
  self.done = False
  self.charged_id_atom = 1
  self.charge_atom = -1.60217646e-19  #boron charge
  self.charged_pos_atom = [0,0,0]
  if self.check_option( "poisson" ):
    poisson_str = self.get_option( "poisson" )
    self.poisson = self.find_simulation( poisson_str )
  else:
    self.poisson = None
```

step2.py

Solution (do_solve)

$$V_{\mathbf{E}} = \frac{1}{4\pi\varepsilon_0}\frac{Q}{r}$$

Electric potential due to a point charge

Searching "impurity"

Calculating potential

```
def do_solve (self):
  if self.done is False :
    K = 8.988e9 # Nm2/C2
    domain = self.get_domain()
    atomistic_domain = N5_AtomisticDomain(domain)
    atoms = atomistic_domain.get_atoms_info();
    for atom in atoms:
      if atom.id == self.charged_id_atom:
        self.charged_pos_atom[0] = atom.position[0]
        self.charged_pos_atom[1] = atom.position[1]
        self.charged_pos_atom[2] = atom.position[2]
        self.potentials[atom.id] = 0
        break;
    #endfor
    for atom in atoms:
      if not atom.id == self.charged_id_atom:
        X = atom.position[0]-self.charged_pos_atom[0]
        Y = atom.position[1]-self.charged_pos_atom[1]
        Z = atom.position[2]-self.charged_pos_atom[2]
        dist = math.sqrt(X*X + Y*Y + Z*Z) * 1e-9 # nm to m
        Pi = ( K * (self.charge_atom) ) / (dist)
        self.potentials[atom.id] = Pi
  self.done = True
```

Communication
(get_data)

```
def get_list_uint_data_list_double ( self, params, variables ):
  result = []
  if self.poisson is not None:
    self.poisson.get_multidata(params, variables, result, "list<uint>:list<double>")
    for i, v in enumerate(variables[0]):
        result[0][i] += self.potentials[v]
  return result
```

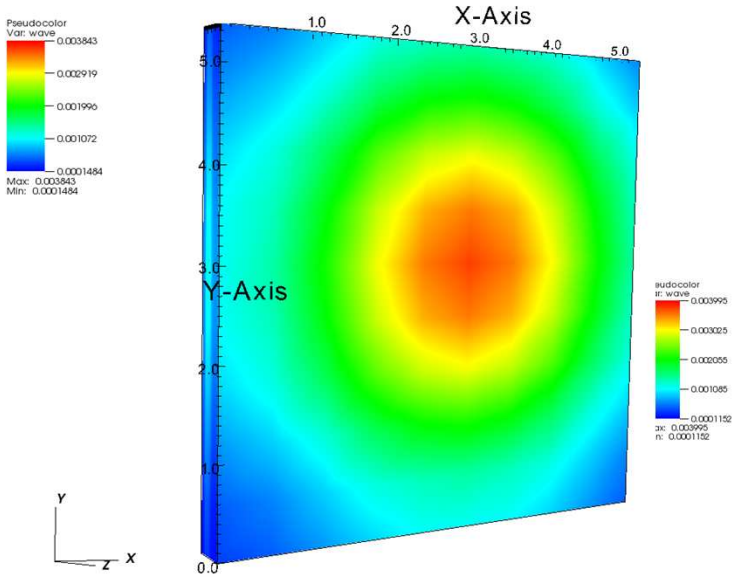
Run step2.py

```
submit -v coates \
    -i /apps/share64/nemo/examples/current/materials/all.mat \
    -i ./step2_small.in \
    -i ./step2.py \
    nemo-r7962 step2_small.in
```

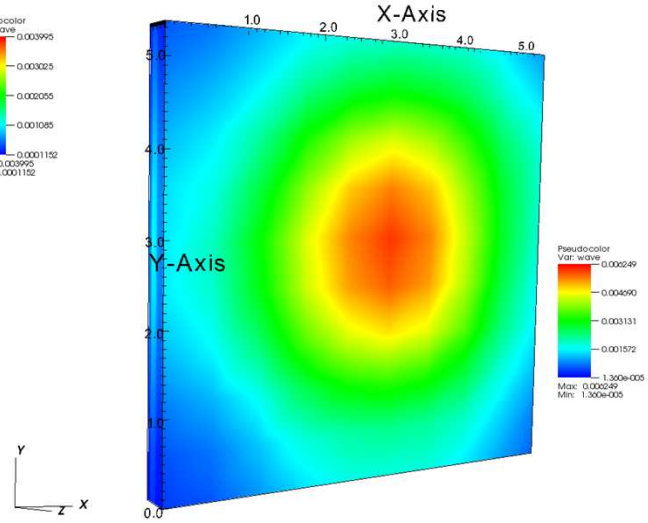
Run step2.py

submit -v coates -i ./all.mat -i ./step2.in -i ./step2.py nemo-rXXXX step2.in

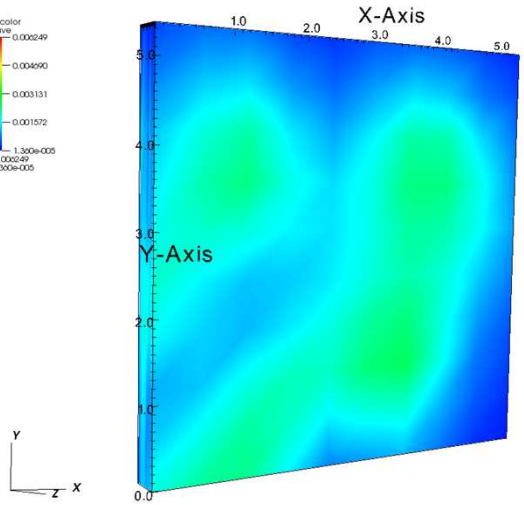
eigenfunctions/wavefunction_0.xyz



step0.py

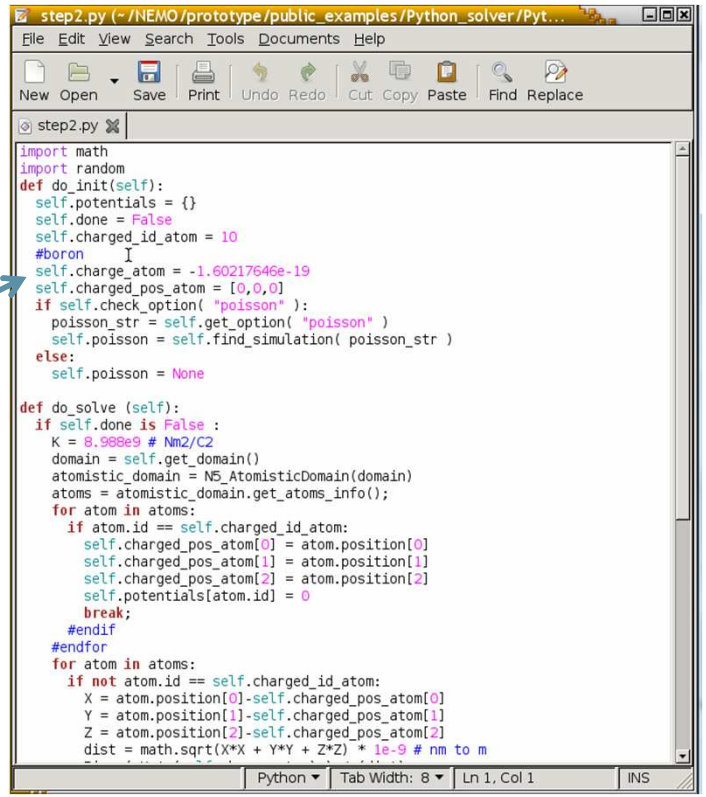


step1.py



step2.py

- File (step2.py)
  - Edit charge_atom value.
  - Edit charge_id_atom value.




```
submit -v coates \
    -i /apps/share64/nemo/examples/current/materials/all.mat \
    -i ./step2_small.in \
    -i ./step2.py \
    nemo-r7962 step2_small.in
```