

NEMO 3-D USER GUIDE FOR QUANTUM DOT SIMULATIONS

Authors: Muhammad Usman (usman@alumni.purdue.edu)
Gerhard Klimeck (gekco@purdue.edu)
Purdue University, West Lafayette, Indiana, USA

INTRODUCTION:

Nanoelectronics is a fascinating area of research that has significant impact on our lives helping in all sort of problems related to medicine, communications, defense, etc. Due to confinement of carriers in three dimensions, quantum dots (QDs) are ideal candidates for the design of nanoscale optoelectronic devices and quantum information processing. However efficient design of QD based devices is still a huge challenge despite significant advancements in their growth and characterization techniques. This is mainly because QD morphology such as their chemical composition, size, shape etc. significantly changes during the growth of capping layer and the post-growth annealing process. Thus modeling and simulations of QDs with realistic dimensions, atomistic resolution, and essential physical effects such as strain, piezoelectricity etc is critical to understand their electronic and optical properties. This can also provide an efficient way to explore their design space to guide experiments before actual experimentation.

Most of the commercial simulators are based on continuum models and are fundamentally limited to qualitative description of the QD properties. NEMO 3-D, based on fully atomistic strain and electronic structure calculations, is a multi-scale simulator that has been compiled and tested to run on multiple cores of modern age supercomputers. Its capabilities include computation of the electronic structure within an empirical tight-binding model for quantum dots (3-D confinement), nanowires (2-D confinement), quantum wells (1-D confinement), and bulk (no confinement); strain is computed from classical valence force field method. Further details about the simulator can be found at: <https://engineering.purdue.edu/gekcogrp/software-projects/nemo3D/> and the source code can be downloaded from the NEMO3D distributions group on nanoHUB.org: https://nanohub.org/groups/nemo_3d_distribution/.

NEMO 3-D is a large and complex simulator; and understanding of its source code requires considerable knowledge of quantum mechanics, condensed matter theory, and parallel programming. Even for just running a simple QD simulation without knowing its source code, good amount of knowledge about QD modeling, XML files, parallel programming, and Linux/Unix commands is imperative. The inputs to NEMO 3-D executable are specified in an XML based input deck which contains many selectable options to set up execution and output parameters. The purpose of this user guide is to instruct a first time user about basic steps for the compilation of NEMO 3-D source code on a Linux/Unix based machine, setup of an input deck for a simple QD simulation, and understanding of various output files that are generated from a typical QD simulation. It

is advised that a user **MUST** first become fully familiar and comfortable with the simple example provided in this manual before moving on to a more complex QD device.

(A) NEMO 3-D Source Code Compilation

Below a step-by-step procedure is presented to download and compile NEMO 3-D code on Linux/Unix based server. As an example, the commands are only presented for a Purdue University machine `coates.rcac.purdue.edu`. However, this procedure can be used to compile the code at any server machine that has c/c++ language compilers, MPI libraries etc. **A user MUST select appropriate compiler in the make.inc file according to the environment where NEMO 3-D is being compiled.**

Step-1: Downloading the source code from the CVS repository.

This part is only for those who have an account on `max.ecn.purdue.edu` for having an access to the CVS repository. (For an account please contact `gekco@purdue.edu`)

(a) In your command prompt (assumed \$) type

```
$ cvs co NEMO_3D
```

After you do this, it will download few license files along with a script file `CVS_nemo3d_setup.sh`.

(b) cd to NEMO_3D folder and run this script file:

```
$ cd NEMO_3D
```

```
$ ./CVS_nemo3d_setup.sh
```

This command will download all the folders and files from the CVS repository to your local folder.

An alternative way is to download the complete software in the form of a tar file from <http://nanoHUB.org> svn repository. This can be done by requesting membership on

https://nanohub.org/groups/nemo_3d_distribution . After getting the membership, you can download the tar file containing NEMO 3-D source code. Now you can unzip this file:

```
$tar -xzvf nemo_3d_08_dec_2008.tar.gz
```

Step-2: Loading modules

(a) In your command prompt type (on `coates.rcac.purdue.edu`)

```
$ module load mpich2-intel64
```

A user must load a correct available module here. NEMO 3-D can be compiled on a bunch of different platforms using various modules. Please see the `make.inc` file for available list of compilers.

Step-3: Building/Making

```
cd to build directory in NEMO_3D
```

```
$ cd build/
```

Set an environment variable for the XERCESCROOT in NEMO_3D

```
$ setenv XERCESCROOT `pwd`/../xerces-c-src2_1_0
```

(You would also find this path at the very end of your downloading all the files n step-1/b.)

```
$ vi make.inc
```

go to the list of compiler options and use (on Coates) `BUILD_TYPE = intel9_32_mpi_intelfast`

Make sure you comment out all other options. Then type,
\$ make
(This will take some time, about 15-20 minutes, and create the executable in nemo3d/bin directory)

Step-4: Running

cd to nemo3d/bin directory.
\$ cd nemo3d/bin
You would see that an executable (nemo3d.ex) has been created in the bin directory.

(a) For serial job

Now to run a simple serial job, do the following. Go to the example folder and copy one of the example files to the bin folder. Then give the run command, for example

```
$ ./nemo3d.ex example.xml (Next section will explain how to build an XML file).
```

Outputs will be created in the binary format. To convert to ASCII format do

```
$ ln -s nemo3d-i386-(press tab key for full name) fmtdat.ex  
$ ./fmtdat.ex -a2 example_nd_Ek
```

(b) parallel job

To run a parallel MPI job you should use the pbs system. Here, how we do it on coates at Purdue:

```
$ cd test  
$ qsub Pbs_test
```

Example of Pbs_test.pbs file is given below.

```
-----  
Pbs_test.pbs  
-----  
  
#!/bin/tcsh  
#PBS -me -l nodes=1:ppn=8,walltime=80:00:00  
#PBS -q ncn  
#PBS -o nemo.out  
#PBS -e nemo.err  
# name of the executable goes on next line  
  
set executable=~/.NEMO_3D/nemo3d/bin/nemo3d.ex"  
  
##set  
set PBS_O_WORKDIR=~/.NEMO_3D/nemo3d/bin/nemo3d.ex "  
  
cd $PBS_O_WORKDIR  
  
set machinefile=`basename $PBS_NODEFILE`  
  
cp $PBS_NODEFILE machinefile.txt  
  
module load mpich2-intel64  
  
mpirun -machinefile machinefile.txt -np 8 $executable ~/.NEMO_3D/nemo3d/bin/example.xml >> live.txt
```

This pbs script uses 1 node and 8 processors per node. So the code will run on 8 cores of a single cluster node in parallel. The simulation job will be submitted to ncn queue.

Output and error log files are nemo.out and nemo.err respectively.

(B) Setting up an Input Deck

The previous section explains how NEMO 3-D code can be downloaded and compiled to get an executable. NEMO 3-D executable runs on XML based input file, referred to as example.xml in the last section. This XML file is modified by the user to specify device geometry, tight binding material parameters, execution options and output files needed. The executable examines the contents of this file according to predefined syntactical rules. If the file is correctly edited, NEMO 3-D extracts input parameters from this file and executes the code according to the requirements specified in the XML file. The input deck XML is divided into four major parts:

1. Geometry Construction
2. Material Database
3. Execution Parameters
4. Output Parameters

Below each part of the XML file will be described in detail. To explain the construction of the input XML file, example device geometry will be used consisting of a single InAs quantum dot in GaAs buffer. The quantum dot is placed on top of a wetting layer 0.5nm thick. The details of the geometry are shown in the figure below.

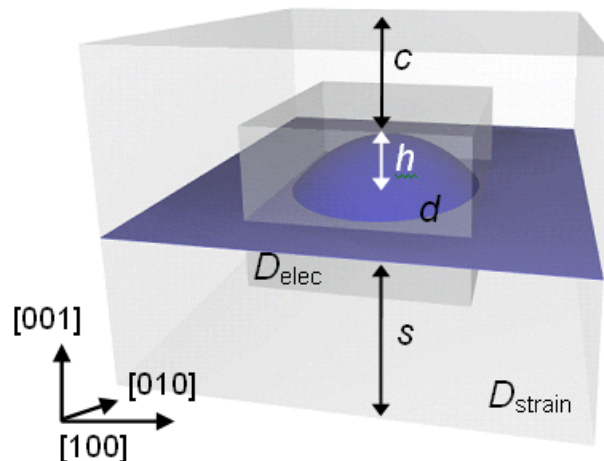


Figure D.1: An example device geometry consisting of a single InAs quantum dot. The height of the quantum dot is h (nm) and base diameter is d (nm). The strain domain is

marked as Dstrain and electronic structure domain is marked as Delec. The substrate thickness is s (nm) and cap layer is c (nm) thick.

(B.1) Geometry Construction

In NEMO 3-D input deck, geometry is constructed by dividing it into various components. For example, the device shown in figure D.1 can be divided into four components: (1) Strain domain (2) Electronic Domain (3) Wetting Layer (4) Quantum dot. Now all of these components will be represented by one shape unit inside the input deck. More important is to know that since the geometry is constructed in NEMO 3-D atomistically, so each subsequent shape replaces the atoms of the previous shape on which this is being placed. For example, let us suppose, we have a box shape of size $30 \times 30 \times 30 \text{ nm}^3$ made up of GaAs with its bottom left corner at $(0, 0, 0)$. Now we define a second shape of size $10 \times 10 \times 10 \text{ nm}^3$ starting at $(10, 10, 10)$ coordinate made up of InAs material. The second shape will replace all the 'Ga' atoms with coordinates $10 \leq x \leq 20$, $10 \leq y \leq 20$, and $10 \leq z \leq 20$ by 'In' atoms. This implies that we define the largest shape first, then the next smaller, then the next smaller and so on. Each shape replaces the atoms of the previous shapes by its material specification within the boundary of its spatial dimensions.

In order to fully understand the geometry construction for the input deck of NEMO 3-D, let us construct the geometry defined in figure D.1. The size of various shapes is:

- (1) Strain Domain: $30 \times 30 \times 30 \text{ nm}^3$, Shape: Box, Material: GaAs
- (2) Wetting Layer: $30 \times 30 \times 30 \text{ nm}^3$, Shape: Box, Material: InAs
- (3) Electronic Domain: $20 \times 20 \times 20 \text{ nm}^3$, Shape: Box, Material: GaAs
- (4) Quantum Dot: $10 \times 10 \times 4 \text{ nm}^3$, Shape: Dome, Material: InAs

In the input deck, the following shapes will be defined:

Shape 1: Box type, GaAs Material, Starting point: $(0, 0, 0)$, Length in x, y, z is: 30, 30, 30 (nm)

→ This is the strain domain of the system. Since this is the largest one, we define it as the first shape.

Shape 2: Box type, InAs Material, Starting Point: $(0, 0, 12.5)$, Length in x, y, z is: 30, 30, 0.5 (nm)

→This is the wetting layer. Since the wetting layer is grown on the whole substrate, its lateral size is 30nm. The quantum dot will be placed in the middle of the whole structure. The wetting layer z coordinate is calculated as $(30-4)/2 = 13\text{nm}-0.5\text{nm} = 12.5\text{nm}$

Shape 3: Box type, GaAs Material, Starting point: (5, 5, 5), Length in x, y, z is: 20, 20, 20 (nm)

→This is the electronic box. Since the quantum dots are highly confined systems, so we normally choose the smaller region to calculate the electronic structure. This reduces the computational time. Here Shape 1 will be selected for only strain calculations and Shape 3 will be selected for both strain and electronic structure calculations. This shape is also placed in the middle of Shape 1.

Shape 4: Box type, InAs Material, Starting Point: (5, 5, 12.5), Length in x, y, z is: 20, 20, 0.5 (nm)

→Since Shape 3 has replaced the wetting layer atoms of Shape 2, we have to redefine the wetting layer inside the electronic domain.

Shape 5: Dome type, InAs Material, Starting Point: (10, 10, 13), Length in x, y, z: 10, 10, 4 (nm)

→This is the quantum dot which is placed in the middle of whole structure.

Hence the geometry of a single quantum dot device in figure D.1 can be described in NEMO 3-D input deck with the help of five shapes. Below, I have explained the XML description of each shape component along with a brief description of each of the options. Five such components constitute the whole geometry.

```

<group type="obj">
  <!-- Shape Descriptor -->
  <name>ShapeName_1</name>
  <cTag>Shape</cTag>
  <desc>Shape descriptor</desc>
  <param type="deletebranch">
    <name>Delete This Shape</name>
    <cTag>DUMMY</cTag>
    <value>dumval</value>
    <desc>None</desc>
  </param>
  <param type="option">
    <name>Shape</name>
    <cTag>Shape</cTag>
    <desc>type of geometric object</desc>
    <value>Box</value>
    <option>Box</option>
    <option>Rhombhedron</option>
    <option>Dome</option>
    <option>Pyramid</option>
    <option>Pyramid_spds</option>
    <option>Pyramid_clipped</option>
    <option>Cylinder</option>
    <option>Cylinder_x</option>
    <option>Ellipsoid</option>
  </param>
  <param type="boolean">
    <name>Has Cell Granularity?</name>
    <cTag>has_cell_granularity</cTag>
    <desc>does this shape have cell granularity?</desc>
    <value>true</value>
  </param>
  <param type="boolean">
    <name>Strain AND Electronic Calculation?</name>
    <cTag>use_for_strain_and_electronic</cTag>
    <desc>also use this shape for electron calculation</desc>
    <value>true</value>
  </param>
  <param type="real">
    <name>x0 (nm)</name>
    <cTag>xorigin</cTag>
    <desc>minimum value in x-direction</desc>
    <value>0</value>
  </param>
  <param type="real">
    <name>y0 (nm)</name>
    <cTag>yorigin</cTag>
    <desc>minimum value in y-direction</desc>
    <value>0</value>
  </param>
  <param type="real">
    <name>z0 (nm)</name>
    <cTag>zorigin</cTag>
    <desc>minimum value in z-direction</desc>
    <value>0</value>
  </param>
  <param type="real">
    <name>L_x (nm)</name>
    <cTag>x</cTag>
    <desc>maximum length in x-direction</desc>
    <value>10.0</value>
  </param>
  <param type="real">
    <name>L_y (nm)</name>
    <cTag>y</cTag>
    <desc>maximum length in y-direction</desc>
    <value>10.0</value>
  </param>
  <param type="real">
    <name>L_z (nm)</name>
    <cTag>z</cTag>
    <desc>maximum height in z-direction</desc>
    <value>10.0</value>
  </param>
</group>

```

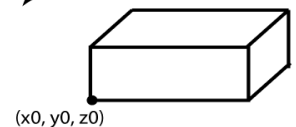
This is the header of the shape component. ShapeName_1 defines the first shape. You can specify any shape name you want. cTag field defines the identifier that is used in the source code to get the value of the field from XML file. <value> ... </value> defines the user selected value. In this case, we set it to dumval.

Here we select the shape of the geometry component. Various options are Box, Rhombhedron, Dome, Pyramid, Pyramid_spds, Pyramid_clipped, Cylinder, Cylinder_x, and Ellipsoid. The value selected in this case is Box.

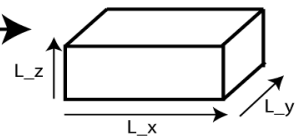
True : The structure will have cell granularity. This means that the shapes will be constructed on a unit cell layer-by-layer basis.
 False: The structure will have atomic granularity. This means that the shapes will be constructed on an atomic layer-by-layer basis.

True : This component will be used in strain and electronic structure calculations.
 False: This component will be used only for strain calculations.

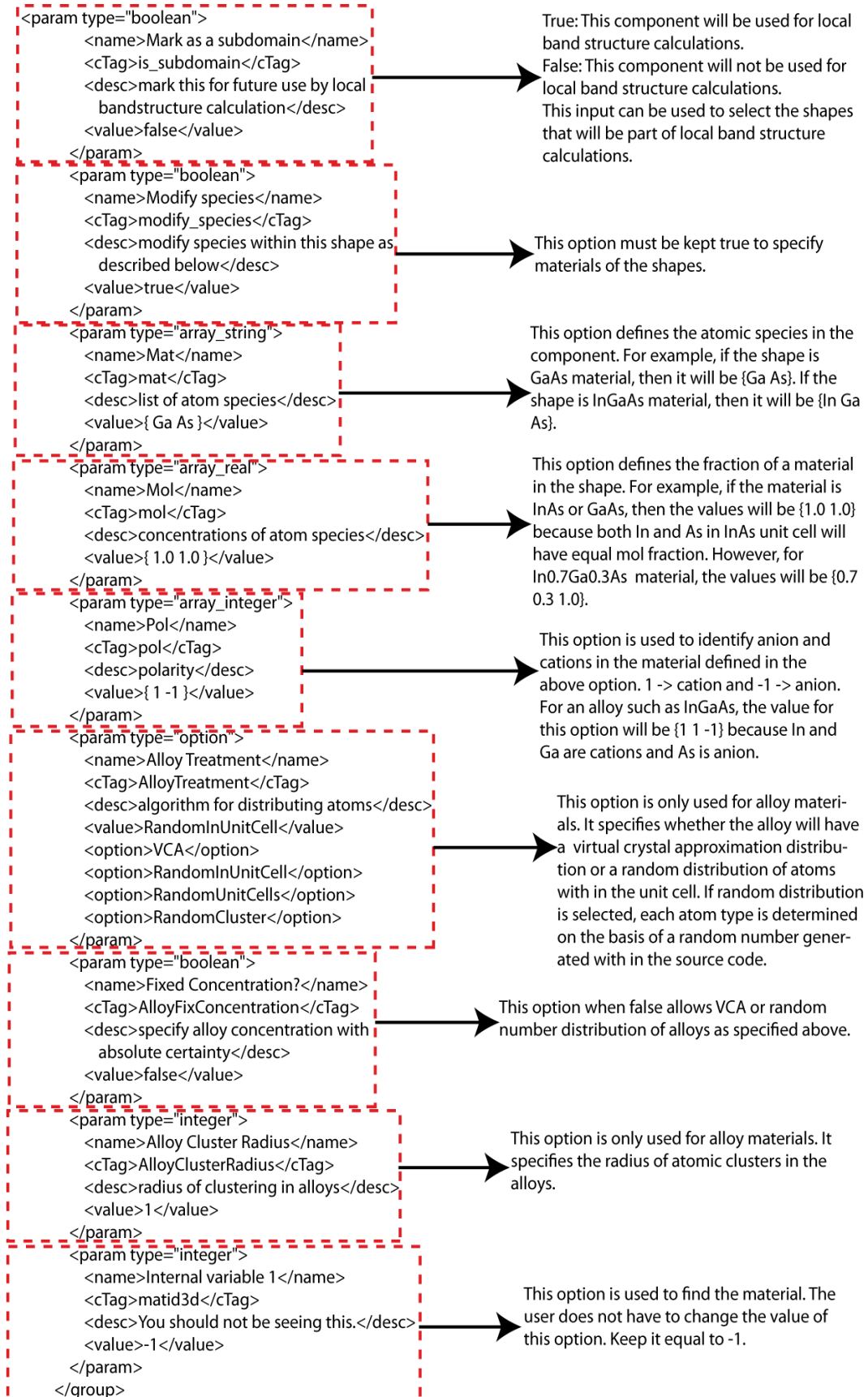
These define the bottom left corner coordinates of the component in the device geometry. For a box:



These define the length of the components in cartesian x, y and z directions within the device geometry. For a box:



For other shapes such as dome, L_x, L_y, and L_z are still the size of the box that contains the dome shape.



Besides describing the geometry of the device, we have to select some parameters related to entire simulation domain. These parameters define the crystal structure to be constructed, the tight binding parameters to be used etc. Brief description of these options is presented below:

```

<param type="option">
  <name>Crystal structure</name>
  <cTag>CrystalStruct</cTag>
  <desc>crystal structure</desc>
  <value>Zincblende</value>
  <option>Cubic</option>
  <option>Diamond</option>
  <option>Zincblende</option>
  <option>Diamond_FCC</option>
  <option>Zincblende_FCC</option>
  <option>Wurtzite</option>
</param>

```

→ This option is used to select the crystal structure. For InAs, GaAs quantum dot, the value should be Zincblende.

```

<param type="option">
  <name>Band Model</name>
  <cTag>band_model</cTag>
  <desc>basis for electronic calculation</desc>
  <value>Bands_20_sp3d5s_spin</value>
  <option>Bands_1_s_nospin</option>
  <option>Bands_10_sp3s_spin</option>
  <option>Bands_10_sp3d5s_nospin</option>
  <option>Bands_20_sp3d5s_spin</option>
</param>

```

→ This option is used to select the band model for the calculation of the electronic structure. Various options are:
 (1) 20 band sp3d5s* model including spin orbital coupling
 (2) 10 band sp3d5s* with out spin
 (3) 10 band sp3d5s* with spin
 (4) Single band model without spin.

```

<param type="real">
  <name>a_x (nm)</name>
  <cTag>a_lattice_x</cTag>
  <desc>lattice constant in x-dir</desc>
  <value>0.56532</value>
</param>
<param type="real">
  <name>a_y (nm)</name>
  <cTag>a_lattice_y</cTag>
  <desc>lattice constant in y-dir</desc>
  <value>0.56532</value>
</param>
<param type="real">
  <name>a_z (nm)</name>
  <cTag>a_lattice_z</cTag>
  <desc>lattice constant in z-dir</desc>
  <value>0.56532</value>
</param>

```

→ Here we specify the lattice constant of the material crystal structure in the x, y, z cartesian coordinate system.

(B.2) Material Database

This part of the input deck XML file is the material database. Here, tight binding and strain parameters of various materials are specified. Each material starts with:

```
<group type="obj">
```

```
<name>material_1</name> → This is the material number in the data base
```

```
<cTag>Material</cTag> → This is the tag for the source code to read this option.
```

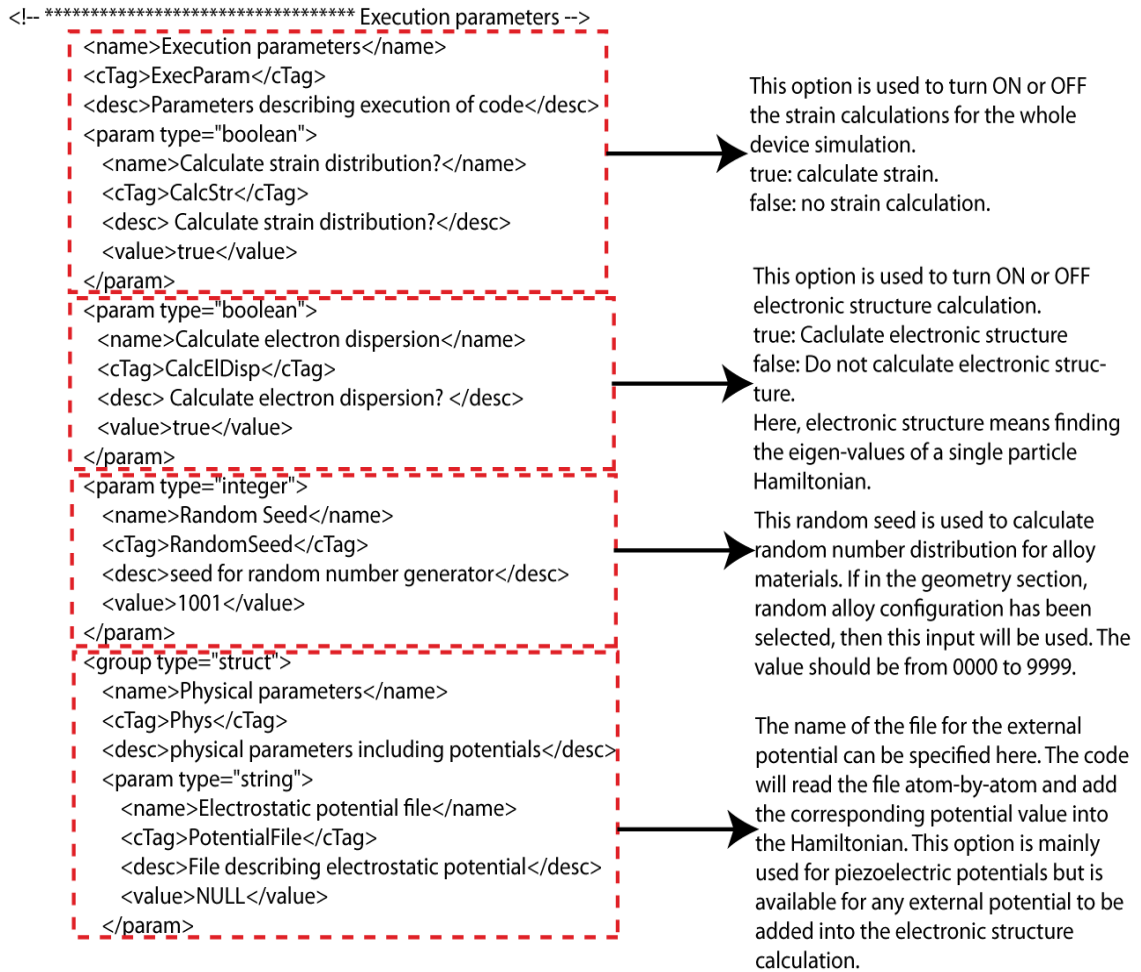
```
<desc>GaAs bulk properties (spds*)</desc> → The name of the material and parameter type.
```

There are currently nineteen (19) materials described in the database. The user does not need to change anything in this section unless new materials need to be specified. A new material can be added by duplicating a material group with resetting material parameters.

(B.3) Execution Parameters

After specifying the geometry and adding the required materials into the material data base, the next step is to select execution options from the execution parameter set. NEMO 3-D is a huge software package and it allows computation of many different parameters under different settings. For example, we can turn off or on the strain calculation. We can turn on or off the electronic structure calculations. We can turn on or off optical matrix element calculations etc. The flexibility to turn ON/OFF various parameters means that we can reduce the computation time by selecting only those parameters that are needed in a particular study.

The execution parameters that are needed to be selected are described below. There are other parameters that have not been described below but are accessible. These parameters are not important for quantum dot simulations and hence should be left with their default values.



NEMO 3-D allows the application of external electrical and magnetic fields for the study of quantum confined stark shifts and g-factor engineering. Below, the execution parameters related to the application of external electrical or magnetic fields are described:

```

<param type="boolean">
  <name>Enable constant magnetic field?</name>
  <cTag>MagneticFieldOn</cTag>
  <desc>is magnetic field on?</desc>
  <value>>false</value>
</param>
<param type="real">
  <name>Bx</name>
  <cTag>Bx</cTag>
  <desc>x component of magnetic field (Tesla)</desc>
  <value>0</value>
</param>
<param type="real">
  <name>By</name>
  <cTag>By</cTag>
  <desc>y component of magnetic field (Tesla)</desc>
  <value>0</value>
</param>
<param type="real">
  <name>Bz</name>
  <cTag>Bz</cTag>
  <desc>z component of magnetic field (Tesla)</desc>
  <value>1</value>
</param>

```

This option is used to apply an external magnetic field to the device.
 true: field is applied
 false: field is not applied
 The value of the external magnetic field components in x, y, z cartesian coordinates is specified. The unit is Tesla.

```

<param type="string">
  <name>Gauge choice for Bfield</name>
  <cTag>Bgauge</cTag>
  <desc>Gauge for the magnetic field, 'Symmetric' gives  $A = (-By/2, Bx/2, 0)$ , 'Asymmetric_x' gives  $A = (0, Bx, 0)$  useful with Efield along x, 'Asymmetric_y' gives  $A = (-By, 0, 0)$ , useful with Efield along y</desc>
  <value>NONE</value>
  <option>NONE</option>
  <option>Symmetric</option>
  <option>Asymmetric_x</option>
  <option>Asymmetric_y</option>
</param>

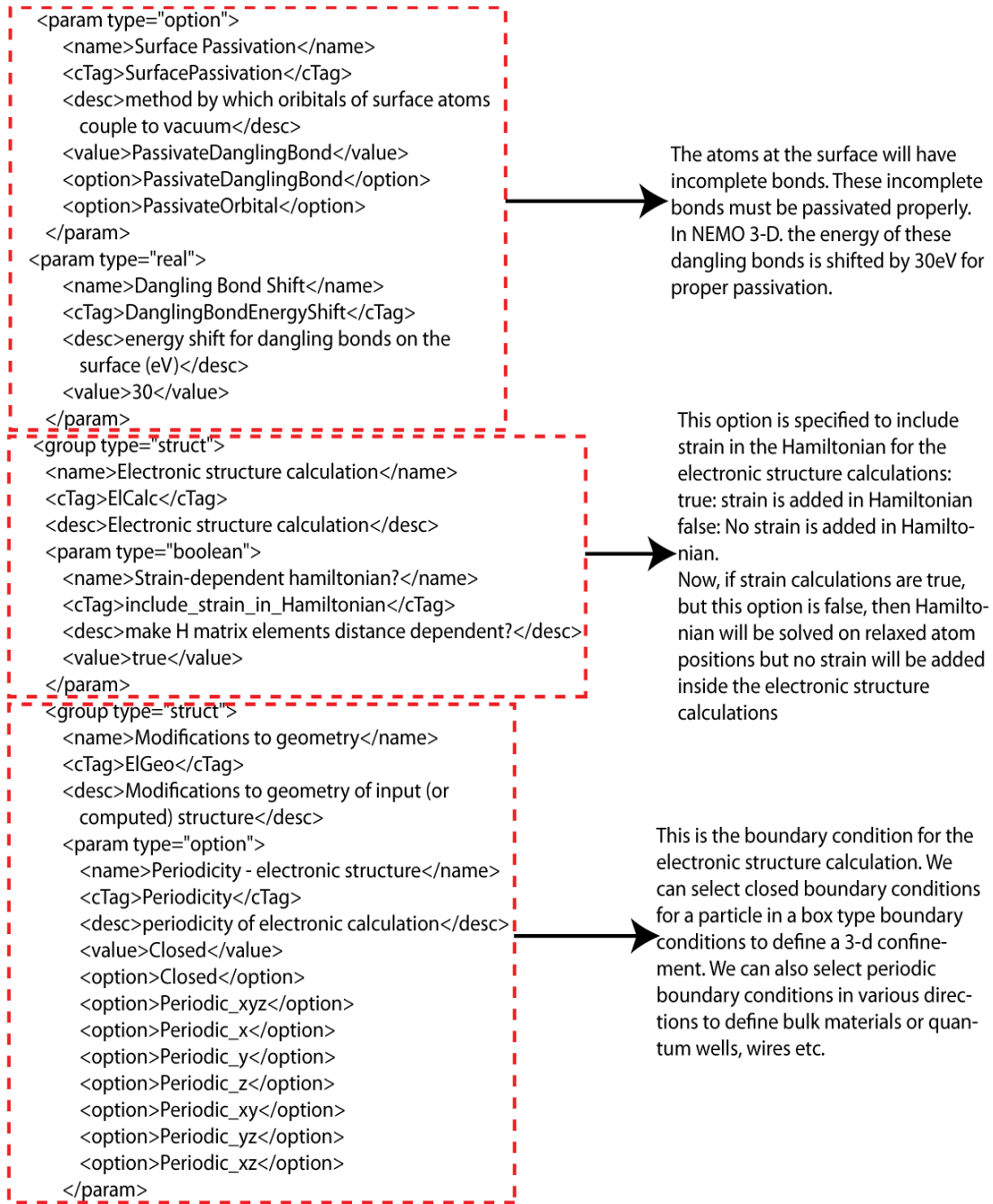
```

```

<param type="boolean">
  <name>Enable constant electric field?</name>
  <cTag>EfieldON</cTag>
  <desc>EfieldON</desc>
  <value>>false</value>
</param>
<param type="real">
  <name>Ex</name>
  <cTag>Ex</cTag>
  <desc>x component of electric field (eV)</desc>
  <value>0</value>
</param>
<param type="real">
  <name>Ey</name>
  <cTag>Ey</cTag>
  <desc>y component of electric field (eV)</desc>
  <value>0</value>
</param>
<param type="real">
  <name>Ez</name>
  <cTag>Ez</cTag>
  <desc>z component of electric field (eV)</desc>
  <value>0</value>
</param>

```

This option is used to apply an external electrical field to the device.
 true: field is applied
 false: field is not applied
 The value of the external electric field components in x, y, z cartesian coordinates is specified. The electrical field is converted to potential energy in eV.



Typical quantum dot simulation contains millions of atoms. The corresponding single particle Hamiltonians are also very large. The calculation of Eigen-values and Eigen-vectors of these large Hamiltonians poses a very computationally expensive problem. NEMO 3-D have various Eigen-solvers that are optimized to run on many cores

to quickly find the Eigen-values and Eigen-vectors to these multi-million atom Hamiltonians. Below I have described the settings of only one Eigen-solver “Lanczos” which is more commonly used for quantum dot simulations. The settings of other Eigen-solvers are more or less similar and can be easily adjusted.

```

<group type="struct">
  <name>Algorithms</name>
  <cTag>ElAlg</cTag>
  <desc>parameters relating to particular algorithms</desc>
  <param type="option">
    <name>Electronic eigensolver</name>
    <cTag>ResFind</cTag>
    <desc>method to find resonances</desc>
    <value>Lancz</value>
    <option>Lancz</option>
    <option>Rayleigh</option>
    <option>LanczAndRayleigh</option>
    <option>Direct</option>
    <option>Readstates</option>
    <option>ParpackSolver</option>
    <option>PQTraceminSolver</option>
    <option>PCTraceminSolver</option>
    <option>BlockLanczos</option>
  </param>
  <param type="integer">
    <name>Number of BZ points</name>
    <cTag>Nk</cTag>
    <desc>Number of k space points</desc>
    <value>1</value>
  </param>
  <param type="array_real">
    <name>Initial k-vector</name>
    <cTag>k0</cTag>
    <desc>(kx, ky, kz)</desc>
    <value>{ 0 0 0 }</value>
  </param>
  <param type="array_real">
    <name>Final k-vector</name>
    <cTag>kf</cTag>
    <desc>(kx, ky, kz)</desc>
    <value>{ 0 0 0 }</value>
  </param>
</group>

```

Here we can select the eigen-solver to calculate eigen-values of the single particle Hamiltonian. Various eigen-solvers are part of the NEMO 3-D package and can be selected for accuracy and speed requirements. For new users, it is recommended to use the Direct solver for bulk band structure calculations and the Lanczos (Lancz) solver for quantum dot energy spectrum calculation.

Here we specify the k-space parameters to calculate the energy spectrum. Number of Bz points specify how many points will be calculated within one Brillouin zone. Initial and final k-space points describe the direction in the k-space to calculate the energy spectrum. For example to calculate the bulk band structure of GaAs material in the [111] direction, one can specify the following:
 Number of BZ points: 60
 initial k-vector: {0 0 0}
 final k-vector: {1 1 1}
 The default value is for an InAs quantum dot system, where the material is a direct band gap material so we calculate eigen-values at the (0 0 0) point of k-space.

<pre> <name>Lanczos</name> <cTag>Lanczos</cTag> <desc>lanczos parameters</desc> <param type="real"> <name>Convergence Tolerance</name> <cTag>ConvTolerance</cTag> <desc>NONE</desc> <value>1e-07</value> </param> </pre>	<p>This option is used to select the convergence criteria of the Lanczos eigen-solver. The higher value means more computational time and more accuracy. Typically for quantum dot systems, a value of 1e-06 to 1e-07 is good enough for proper convergence of eigen-values</p>
<pre> <param type="integer"> <name>Maximum Iteration Number</name> <cTag>MaxIter</cTag> <desc>NONE</desc> <value>2400</value> </param> </pre>	<p>This value specifies the number of iterations required to find the requested number or eigen-values. A higher value should be specified here because once Lanczos finds the specified number of eigen-values, it stops automatically. However a lower value means it might not be able to find the requested eigen-values. A typical value for a quantum dot system is between 10000 to 15000.</p>
<pre> <param type="integer"> <name>conv_method</name> <cTag>conv_method</cTag> <desc>1 for eigenvalues and 2 for eigenvectors</desc> <value>1</value> </param> </pre>	<p>This value should be kept at 1 to find eigen-values of the system. In case, where only eigenvectors are needed, it can be set to 2.</p>
<pre> <param type="integer"> <name>Number of requested VB eigenvalues</name> <cTag>NumEigReq_vb</cTag> <desc>NONE</desc> <value>3</value> </param> <param type="integer"> <name>Number of requested CB eigenvalues</name> <cTag>NumEigReq_cb</cTag> <desc>NONE</desc> <value>3</value> </param> </pre>	<p>Here we can specify the number of conduction bands and the number of valence band eigen-values required from the simulation. For a value of 3 and 3, the simulation will return the lowest three conduction band energy states and highest three valence band energy states.</p>
<pre> <param type="array_real"> <name>Valence band energy range</name> <cTag>Erange_vb</cTag> <desc>Search range for valence band eigenvalues</desc> <value>{-0.6 0.2}</value> </param> <param type="array_real"> <name>Conduction band energy range</name> <cTag>Erange_cb</cTag> <desc>Search range for conduction band eigenvalues</desc> <value>{ 1.4 2.0}</value> </param> </pre>	<p>This parameter is very important because it specifies the range where Lanczos will try to find the energy values. If we specify the wrong range, it will never find a converged eigen-value, or it will give only those eigen-values that are in the range specified. For example for InAs/GaAs quantum dots, the lowest conduction band energy state is 1.2-1.3eV. Now if we specify the range {1.5-1.8}, we will never get the lowest conduction band energy state. It is the same case for the valence band. For InAs/GaAs quantum dots, the conduction band energy range should be {1.2 2.0} and the valence band energy range should be {-0.2 0.4}</p>

Below the selection of strain parameters is described. Strain is calculated in NEMO 3-D by relaxing the atoms according to the valence force field method by minimizing the total energy of the system. The selection of the correct strain model and the boundary conditions is critical for proper strain calculations:


```

<group type="struct">
  <name>Strain calculation</name>
  <cTag>Strain</cTag>
  <desc>strain parameters</desc>
  <param type="option">
    <name>Strain Model</name>
    <cTag>StrainModel</cTag>
    <desc>strain model</desc>
    <value>VFF_keating_strained</value>
    <option>None</option>
    <option>VFF_keating</option>
    <option>VFF_keating_strained</option>
    <option>Read</option>
  </param>

```

This option is used to select the strain model for the VFF strain calculation.

VFF_keating: This model relaxes the atoms using the standard VFF Keating strain model with keating potentials.

VFF_keating_strained: This model includes the anharmonic corrections in it. For details see Olga et al. APL 2006.

```

<group type="struct">
  <name>Keating</name>
  <cTag>Keating</cTag>
  <desc>keating parameters</desc>
  <param type="option">
    <name>Strain constraint</name>
    <cTag>BoundCond</cTag>
    <desc>impose constraint on configuration</desc>
    <value>FixZmin</value>
    <option>AllFree</option>
    <option>FixOne</option>
    <option>FixSurfaceAll</option>
    <option>FixInterior</option>
    <option>FixZmin</option>
  </param>

```

This option specifies boundary conditions for the strain domain.

FixZmin: The bottom of the strain domain will be fixed to the buffer lattice constant.

AllFree: All the sides of the strain domain will be free to relax.

FixSurfaceAll: This will fix all the surfaces of the strain domain to the unstrained lattice constant. This is equivalent setting closed boundary conditions.

```

<param type="option">
  <name>Periodicity (strain)</name>
  <cTag>Periodicity_strain</cTag>
  <desc>periodicity for strain calculation</desc>
  <value>Periodic_xy</value>
  <option>SameAsGlobal</option>
  <option>Closed</option>
  <option>Periodic_xyz</option>
  <option>Periodic_x</option>
  <option>Periodic_y</option>
  <option>Periodic_z</option>
  <option>Periodic_xy</option>
  <option>Periodic_yz</option>
  <option>Periodic_xz</option>
</param>

```

This option can be used to further apply the boundary conditions to the strain domain. The options are the closed boundary conditions and the periodic boundary conditions along the various axis.

A typical quantum dot simulation requires bottom fixed, sides periodic and top free. This can be achieved by selecting FixZmin from the above option and Periodic_xy from this option. The top will automatically be free because no boundary conditions are imposed on that.


```

<param type="option">
  <name>Minimize wrt lattice const</name>
  <cTag>MinWrtLatt</cTag>
  <desc>allow the periodicity to relax</desc>
  <value>none</value>
  <option>none</option>
  <option>x</option>
  <option>y</option>
  <option>z</option>
  <option>xy</option>
  <option>xz</option>
  <option>yz</option>
  <option>xyz</option>
</param>

```

For a quantum dot simulation, this option must be set to none. This option can be used to allow the periodicity of the structure to relax in the specified direction.

```

<param type="boolean">
  <name>Update lattice constants before strain calculation?</name>
  <cTag>UpdateLatt</cTag>
  <desc>update lattice constants before strain calculation?</desc>
  <value>false</value>
</param>
<param type="array_real">
  <name>If UpdateLatt is true, what are the new lattice constants?</name>
  <cTag>UpdateLattConst</cTag>
  <desc>If UpdateLatt is true, what are the new lattice constants?</desc>
  <value>{ 0.56532 0.56532 0.56532 }</value>
</param>

```

Normally the lattice constants of the material are not whole numbers but rather they are in the form of fractions. For example, GaAs has a lattice constant of 0.56532. To make the geometry construction easy, we can use 0.5 when specifying lattice constant in the x, y, z directions. Here we can make this input true and specify the real lattice constant. That means the geometry will be constructed using the 0.5 lattice constant, and before strain relaxation it will be slightly modified to follow real lattice constant. This option is only for the user's convenience.

```

<param type="real">
  <name>What is the scaling factor for dE/da?</name>
  <cTag>SCALE_FCTR</cTag>
  <desc>What is the scaling factor (S) for dE/da?
    Use number between 1e-2 to 1e-5 depending
    on the number of atoms (N) in strain calculation.
    An ideal S is about 1.0/sqrt(N).
  </desc>
  <value>1e-3</value>
</param>

```

This option is useful for fast convergence of the strain. The value specified here must be calculated using the formula:
 $1/(\text{sqrt}(\text{number of atoms}))$

```

<param type="integer">
  <name>Maximum iteration number</name>
  <cTag>MaxIter</cTag>
  <desc>maximum number of iterations</desc>
  <value>30000</value>
</param>

```

These options specify the maximum number of iteration and convergence criteria for strain calculations. The typical values for a quantum dot simulation are 30000-40000 and 1e-6 to 1e-7.

```

<param type="real">
  <name>Tolerance</name>
  <cTag>tol</cTag>
  <desc>tolerance</desc>
  <value>1e-7</value>
</param>

```

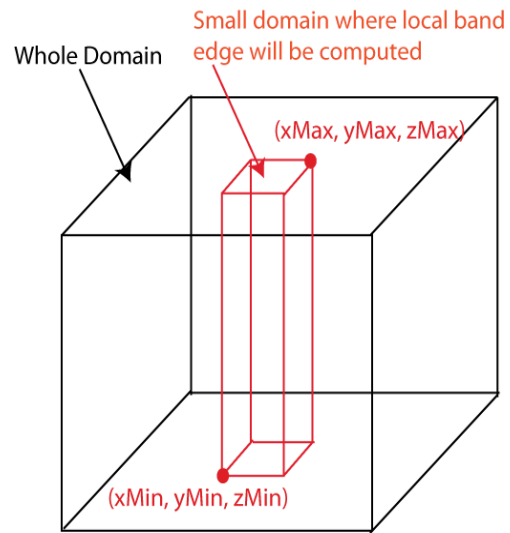
```

<group type="struct">
  <!-- Subdomain -->
  <name>Subdomain</name>
  <cTag>Subdomain</cTag>
  <desc>subdomain.</desc>
  <param type="real">
    <name>xMin</name>
    <cTag>xMin</cTag>
    <desc>xMin</desc>
    <value>0</value>
  </param>
  <param type="real">
    <name>yMin</name>
    <cTag>yMin</cTag>
    <desc>yMin</desc>
    <value>3.5</value>
  </param>
  <param type="real">
    <name>zMin</name>
    <cTag>zMin</cTag>
    <desc>zMin</desc>
    <value>3</value>
  </param>
  <param type="real">
    <name>xMax</name>
    <cTag>xMax</cTag>
    <desc>xMax</desc>
    <value>8</value>
  </param>
  <param type="real">
    <name>yMax</name>
    <cTag>yMax</cTag>
    <desc>yMax</desc>
    <value>3.7</value>
  </param>
  <param type="real">
    <name>zMax</name>
    <cTag>zMax</cTag>
    <desc>zMax</desc>
    <value>4.3</value>
  </param>
</group>

```

If we want to calculate local band edges, we have to specify the spatial region inside the geometry for which it calculate the local band edge data. The shapes that lie in the subdomain defined here must have the local band edge option true. The region is specified as follows:

This defines the lower left corner of the region where the local band edge is to be computed



This defines the upper right corner of the region where the local band edge is to be computed












(B.4) Output Parameters

The final section of the NEMO 3-D input deck is the output parameter section. After selecting the right parameters in the execution parameter section, the NEMO 3-D simulation will compute the various physical quantities for a quantum dot simulation. However, in general, all of the computed quantities are not needed. For example, one might need the electronic structure including the strain and external electrical field, but

not the strain data or electrical field data. The solution will be to turn ON the strain, external field and electronic structure calculations in the execution parameters section and turn OFF the strain and external field in the output parameter section. The electronic structure will be turned ON in the output parameter section. These settings will generate output files which contain the electronic structure data including the strain and the external fields. Since the quantum dot simulations are multi-million atom simulations, so the output files will be very large. By turning OFF the unnecessary output files in the output parameter section, much of the disk space can be saved.

Below, the various options available in the output parameter section of the NEMO 3-D input deck are described:

<pre> <!-- Output parameters. --> <name>Output</name> <cTag>ElOut</cTag> <desc>output parameters</desc> <param type="boolean"> <name>EnergyDispersion</name> <cTag>EnergyDispersion</cTag> <desc>EnergyDispersion</desc> <value>true</value> </param> </pre>	<p>→ This option when true will create a file *Ek that will contain the eigen-values of the single particle Hamiltonian as a function of kx, ky, kz</p>
<pre> <param type="boolean"> <name>Eigenvectors</name> <cTag>Eigvect</cTag> <desc>write electronic wavefunctions to file?</desc> <value>false</value> </param> </pre>	<p>→ This option when true will dump out *evec_* files that will contain eigen-vectors of the system representing single particle energy states i.e. Ψ</p>
<pre> <param type="boolean"> <name> Psi(r) ^2</name> <cTag>PsiSqr</cTag> <desc>write modulus squared of the wavefunctions to file</desc> <value>true</value> </param> </pre>	<p>→ This option when true will generate output files *wf_* that contain data for $\Psi ^2$</p>
<pre> <param type="boolean"> <name> Psi(cell) ^2</name> <cTag>PsiSqr_cell</cTag> <desc>write modulus squared of the wavefunctions to file (but averaged over a unit cell)</desc> <value>false</value> </param> </pre>	<p>→ This option when true will generate output files *wf_* that contain data for $\Psi ^2$ but it will be averaged over a unit cell</p>
<pre> <param type="boolean"> <name>Visualization_3D</name> <cTag>Visualization_3D</cTag> <desc>write output files of dx-format for 3D visualization</desc> <value>true</value> </param> </pre>	<p>→ This option when true will generate extra files for $\Psi ^2$ for the visualization purpose. These files will have dx format and can be used with visualizer on nanoHUB.org to plot the wave functions.</p>
<pre> <param type="boolean"> <name>Local Bandstructure</name> <cTag>LocalBands</cTag> <desc>compute local band structure?</desc> <value>false</value> </param> </pre>	<p>→ This option when true will calculate the local band structure in the real space at each unit cell. The spatial domain where local band edge data will be generated is specified using subdomains.</p>
<pre> <param type="boolean"> <name>Trace of eigenvalues</name> <cTag>TraceEigval</cTag> <desc>Save a log of the computed Ritz values generated by the Lanczos algorithm</desc> <value>false</value> </param> </pre>	<p>→ This option when true will generate the trace of eigen-values as a function of Lanczos iteration. The output file generated is *eval. This file can be used to check the convergence of the eigen-values for troubleshooting purpose.</p>

<pre><param type="boolean"> <name>OptMatrixElements</name> <cTag>OptMatrixEl</cTag> <desc> OptMatrixElements </desc> <value>true</value> </param></pre>		<p>This input when true will generate data files for optical transition strengths between energy levels of conduction and valence bands. The files generated will be *TransX, *TransY, *TransZ.</p>
<pre><param type="boolean"> <name>Neighborhood</name> <cTag>Neighborhood</cTag> <desc> Neighborhood </desc> <value>>false</value> </param></pre>		<p>This option when true will generate an output file containing information about each atoms neighbors. The file generated will be *nbr.</p>
<pre><param type="boolean"> <name>Bondlength info</name> <cTag>Bondlength</cTag> <desc> Bondlength </desc> <value>true</value> </param></pre>		<p>This option when true will generate the average bond length of each atom in the structure. This information is important to see the strain induced bond deformations.</p>
<pre><param type="boolean"> <name>output strain from the electronic domain only?</name> <cTag>StrainElOnly</cTag> <desc>StrainElOnly</desc> <value>>false</value> </param></pre>		<p>Since the strain domain is normally much larger than the electronic domain, so this option can be made true to get strain data only from the electronic domain.</p>
<pre><param type="boolean"> <name>output file for tension</name> <cTag>TensionOutput</cTag> <desc>write output files of six tension components for visualization</desc> <value>true</value> </param></pre>		<p>This option when true will generate strain files. one for each strain component exx, eyy, ezz, exz, eyz. exy. These files can be used to visualize strain profiles on nanoHUB.org.</p>
<pre><param type="boolean"> <name>Displacement data for restart</name> <cTag>DisplacementData</cTag> <desc>NONE</desc> <value>>false</value> </param></pre>		<p>This option when true generates a data file containing displacement of atoms after the strain calculation. This data can be used to feed in to the electronic structure for subsequent simulations without re-calculating the strain.</p>
<pre><param type="boolean"> <name>AtomInfo</name> <cTag>AtomInfo</cTag> <desc>Print Atom-Type Information</desc> <value>true</value> </param></pre>		<p>This option when true generates a file *aType which contains the atom type of all the atoms in the structure. The atom information is stored in the same order as the atom positions in the *rAtom file. Each atom type has been assigned a number to identify it.</p>
<pre><param type="boolean"> <name>NbrIndx</name> <cTag>NbrIndx</cTag> <desc>Print Neighbor Index</desc> <value>true</value> </param></pre>		<p>This option when true will generate a file which contains for each atom the index of its 4 nearest neighbors. The indexing is done in the same order as the atom positions stored in the rAtom file.</p>
<pre><param type="boolean"> <name>Dump atomic positions before strain?</name> <cTag>AtomPosBeforeStrain</cTag> <desc>AtomPosBeforeStrain</desc> <value>true</value> </param></pre>		<p>This option when true will generate *rAtom_0 file which contain all the atom coordinates in x,y,z cartesian coordinate format. The atom positions are on a regular crystal grid without any displacements induced by the strain.</p>
<pre><param type="boolean"> <name>Dump atomic positions after strain?</name> <cTag>AtomPosAfterStrain</cTag> <desc>AtomPosAfterStrain</desc> <value>true</value> </param></pre>		<p>After the strain calculations, the relaxed atomic positions are slightly displaced from their original regular crystal grid positions. This option when true will generate a file *rAtom_1 that contains atom positions after strain relaxation.</p>
<pre><param type="boolean"> <name>ShapeInfo</name> <cTag>ShapeInfo</cTag> <desc>Print Shape Information</desc> <value>>false</value> </param></pre>		<p>The device structure is constructed in the form of various shapes, lying on the top of each other. This input when true will generate a file *shapeS that contains information about the shapes present in the structure. This file is important to generate visualization plots on nanoHUB.org.</p>

(C) Format of Output Files Generated by NEMO 3-D

After compiling NEMO 3-D source code to obtain an executable and setting up the input deck XML file for the device geometry construction, the simulation can be run serially on a single processor or as a parallel job on various processors using a pbs script as defined in section A. If the job completes successfully, the working directory will contain the output files that were selected in the output parameter section. Below, the name and format of various output files are presented for the first time user to understand the output generated by NEMO 3-D simulations. In general, the output files generated by NEMO 3-D have the name of XML file as their prefix. For example, if the name of input deck is example.xml, then all of the output files will have name example.nd_*

(1) Energy Dispersion: The name of this file is *Ek. This file contains E versus k values. The format is:

<kx ky kz Energy>

(2) Eigen-vectors: For each Eigen-value (energy level), one Eigen-vector file is computed. For example, if *Ek file contains 10 energy levels, then there will be 10 Eigen-vector files corresponding to each energy level. The name of the files will be *evec_*i*, where *i* is the index number of the energy level to which this Eigen-vector belong. The format of the file is:

<real complex> → real and complex parts of Eigen-vector.

(3) Wave functions: For each Eigen-value (energy level), one wave function file is computed. For example, if *Ek file contains 10 energy levels, then there will be 10 wave function files corresponding to each energy level. The name of the files will be *wf_*i*, where *i* is the index number of the energy level to which this wave function belong. The format of the file is:

<| ψ |²>

(4) Local Band Structure: Local band structure file contains local band edge data in the sub domain specified in the execution parameter section. One file is generated for each processor. The name of file is: *Ek_proc_*pn*_kx0.000_ky_0.000_kz_0.000_r where *pn* is the processor number. The format of the file is:

<kx ky kz x y z energy atomType>

(5) Trace of Eigen-values: The name of file is *eval. The file contains Eigen-value versus iteration number of the Lanczos Eigen-solver. The format of the file is:

<Iteration number Eigen-value>

(6) Optical Matrix Element: The output files generated are optical transition rates for incident light polarization along the X, Y and Z Cartesian coordinates. The file names are *TransX, *TransY, and *TransZ. The format of the files is:

<energy_state_i energy_state_f energy_gap transition_rate>

(7) Atom Position Files: Atom position file before the strain contains atom coordinates in the Cartesian real space. The name of the file is *rAtom_0. The atom positions file after the strain contains coordinates in the Cartesian real space after strain relaxation. The name of this file is *rAtom_1. The format of both files is:

<x y z>

(8) Atom Type File: This file contains the type of each atom in the form of a unique ASCII number assigned to the atom. For example, In atom is assigned a number 14, As atom is assigned a number 12, Ga atom is assigned a number 10 etc. The name of the file is *aType and format of the file is:

<atom_type>

(9) Neighbor Index File: The name of the file is *nbrIndx. This file contains information of 4 nearest neighbors for each atom in the geometry. The ordering is the same as the ordering of the atom positions in the rAtom file. The format of this file is:

<neighbor-1 neighbor_2 neighbor_3 neighbor_4>

Note: This document is created to serve as a guide for first time users of NEMO 3-D software who intend to use it for quantum dot simulations. The manual is created with great care to provide accurate information. However, the authors do not take any responsibility of any kind for the contents presented. The entire risk as to the quality, the performance, and the fitness of the manual for any particular purpose lies with the party using it. In no event will the authors or any party who distributed the manual be liable for damages or for any claim(s) by any third party, including but not limited to, any lost profits, lost monies, lost data or data rendered inaccurate, losses sustained by third parties, or any other special, incidental or consequential damages arising out of the use or inability to use the program, even if the possibility of such damages has been advised against.