DJMol

A GENERIC OPEN SOURCE MODELING PLATFORM FOR

COMPUTATIONAL MATERIALS SCIENCE AND CHEMISTRY

EMPOWERED WITH

DFTB+ SIESTA ASE OPENMD

WINDOWS VERSION 2.1 MANUAL

Dr. Krishna Mohan G P **Rahul Sunil**

Mar Baselios College of Engineering and Technology, Nalanchira, Trivandrum, Kerala-695015, India.

Dr. Kapil Gupta **Dr. Seung-Cheol Lee**

Indo-Korea Science and Technology Centre, 17/1, Bellary Rd, Yashoda Nagar, Yelahanka, Bengaluru, Karnataka-560064, India.

TABLE OF CONTENTS

1

INTRODUCTION

DJMol is an ongoing open source (licensed with GPL v3 or later) '*Modeling Platform*' project and it is architectured for performing various simulations in computational materials science / chemistry. It is a free, object-oriented software developed by IKST, (Indo-Korea Science and Technology Centre, 17/1, Bellary Rd, Yashoda Nagar, Yelahanka, Bengaluru, Karnataka 560064), Bangalore.

In its core level, DJMol is integrated with widely accepted and open source (GPL/LGPL) computational software, such as,

DFTB+ (University of Bremen, Germany), the *default calculator* of DJMol

Atomic Simulation Environment (ASE, Technical University of Denmark)

SIESTA (Institut de Ciencia de Materials de Barcelona (ICMAB), Spain)

OpenMD (University of Notre Dame., USA).

The main target of this software is the researchers (including experimentalists) and academicians and industrial researchers. In addition to this, a site, *https://sites.google.com/view/djmolplatform* is created for more technical details, updates and demonstration videos/notes etc. The code-development resources are recently included in its official site, *www.djmol.info*.

As mentioned before, the software is integrated with DFTB+ (as its default calculator), and Siesta (for higher level DFT calculations), along with ASE and OpenMD,. Hence a short review on these packages is added for a quick-reference.

Computational Tools

Here, a short overview of the four main components of the platform is described.

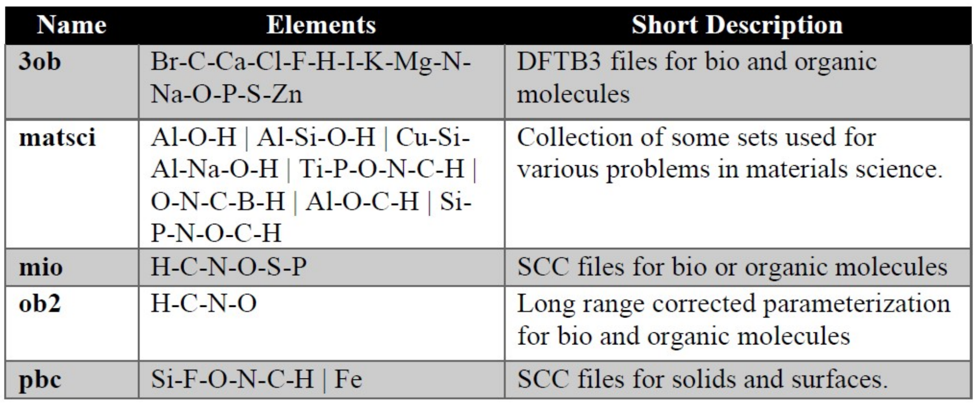
DFTB+: DFTB (Density functional tight binding) method is an electronic structure method which is specially designed to model large molecular/crystal/nano-systems. The origin of this method is routed in DFT itself, which is, arguably, the most popular *ab initio* method in computational materials science or in quantum chemistry. It is developed in 80's and it has been introduced into the computational chemistry area mainly by work of Prof. Helmut Eschrig, Prof. Gotthard Seifert, and Prof. Thomas Frauenheim et. al. Since it uses DFT parameters, DFTB is also known as *approximate* DFT method.

This method uses atomic parameters (as it is stored in the SK file set, a text file) which are generated from highly accurate atomic DFT calculations using PBE functional. It also consists of diatomic repulsive potentials from B3LYP functional. Note that DFTB+ uses minimal valence basis set (numerical Slater type AOs) and one has to explicitly specify a particular SK data file for a given calculation.

A list of SK **file set** list is given below (see*, www.dftb.org*), which is by default, added in the DJMol suite. Other SK file which are not included (such as, <3ob:freq> set which requires <3ob>) should manually incorporate into their input scripts (it is freely available); For more details, see:

https://www.dftbplus.org/ and
https://www.dftb.org/parameters/download/

Table 1: Set of common SK data set



Name	Elements	Short Description
3ob	Br-C-Ca-Cl-F-H-I-K-Mg-N-Na-O-P-S-Zn	DFTB3 files for bio and organic molecules
matsci	Al-O-H | Al-Si-O-H | Cu-Si-Al-Na-O-H | Ti-P-O-N-C-H | O-N-C-B-H | Al-O-C-H | Si-P-N-O-C-H	Collection of some sets used for various problems in materials science.
mio	H-C-N-O-S-P	SCC files for bio or organic molecules
ob2	H-C-N-O	Long range corrected parameterization for bio and organic molecules
pbc	Si-F-O-N-C-H | Fe	SCC files for solids and surfaces.

SIESTA: Spanish Initiative for Electronic Simulations with Thousands of Atoms is a fast (since it uses linear scaling DFT methods) density functional theoretical code which uses pseudopotentials. It can handle molecular as well as extended systems (metals, polymers, defects in the metals, surfaces etc.) It is a free code and for more details see: *https://departments.icmab.es/leem/siesta*. And *routinely* it can be used to predict:

• Total and partial energies.

• Atomic forces.

• Stress tensor.

• Electric dipole moment.

• Atomic, orbital and bond populations (Mulliken).

• Electron density.

• Geometry relaxation, fixed or variable cell.

• Constant-temperature molecular dynamics (Nose thermostat).

• Variable cell dynamics (Parrinello-Rahman).

• Spin polarized calculations (collinear or not).

• k-sampling of the Brillouin zone.

• Local and orbital-projected density of states.

• COOP and COHP curves for chemical bonding analysis.

• Dielectric polarization.

• Vibrations (phonons).

• Band structure.

Note that at present TranSiesta (electron transport module) is not supported by DJMol

Since Siesta uses *Numerical Atomic Orbital* (i.e. localized basis set) as its basis, some familiarity with these sets are highly appreciated. Note that a user can give 'Basis set' information to Siesta in two different ways. The most simple method is to provide a 'keyword'of the basis set. See the next table for its details:

Table 2: A set of common basis set in Siesta

Single-Zeta	Double-Zeta	Triple-Zeta	Single-Zeta Polarized	Double-Zeta Polarized	Triple-Zeta Polarized	Triple-Zeta Double-Polarized
SZ	DZ	TZ	SZP	DZP	TZP	TZDP

However, for advanced calculations a user can also give the basis set data explicitly. Refer Siesta manual for its description.

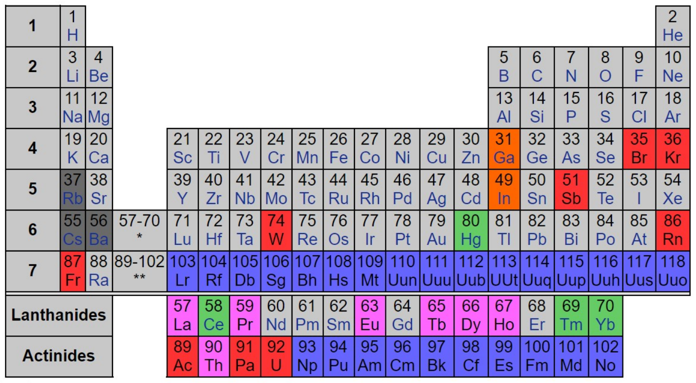
Apart from the basis set, one should also select an appropriate pseudopotential (PP). For more details, see:

[1]https://departments.icmab.es/leem/siesta/Pseudopotentials/index.html or,

[2]https://departments.icmab.es/leem/siesta/Databases/Pseudopotentials/periodictable-gga-abinit.html

The first link redirects to ATOM program which can be used to generate PP for elements within the GGA/LDA functionals. The second link gives a database of PP which was converted from the Abinit PP. Note that choosing PP is a crucial step in Siesta calculation so a user must be Very Careful at this step.

Note: ***By default, DJMol use PP from the Second link [2].***



1 | 1 H | 2 He
2 | 3 Li | 4 Be | 5 B | 6 C | 7 N | 8 O | 9 F | 10 Ne
3 | 11 Na | 12 Mg | 13 Al | 14 Si | 15 P | 16 S | 17 Cl | 18 Ar
4 | 19 K | 20 Ca | 21 Sc | 22 Ti | 23 V | 24 Cr | 25 Mn | 26 Fe | 27 Co | 28 Ni | 29 Cu | 30 Zn | 31 Ga | 32 Ge | 33 As | 34 Se | 35 Br | 36 Kr
5 | 37 Rb | 38 Sr | 39 Y | 40 Zr | 41 Nb | 42 Mo | 43 Tc | 44 Ru | 45 Rh | 46 Pd | 47 Ag | 48 Cd | 49 In | 50 Sn | 51 Sb | 52 Te | 53 I | 54 Xe
6 | 55 Cs | 56 Ba | 57-70 * | 71 Lu | 72 Hf | 73 Ta | 74 W | 75 Re | 76 Os | 77 Ir | 78 Pt | 79 Au | 80 Hg | 81 Tl | 82 Pb | 83 Bi | 84 Po | 85 At | 86 Rn
7 | 87 Fr | 88 Ra | 89-102 ** | 103 Lr | 104 Rf | 105 Db | 106 Sg | 107 Bh | 108 Hs | 109 Mt | 110 Uun | 111 Uuu | 112 Uub | 113 UUt | 114 Uuq | 115 Uup | 116 Uuh | 117 Uus | 118 Uuo
Lanthanides | 57 La | 58 Ce | 59 Pr | 60 Nd | 61 Pm | 62 Sm | 63 Eu | 64 Gd | 65 Tb | 66 Dy | 67 Ho | 68 Er | 69 Tm | 70 Yb
Actinides | 89 Ac | 90 Th | 91 Pa | 92 U | 93 Np | 94 Pu | 95 Am | 96 Cm | 97 Bk | 98 Cf | 99 Es | 100 Fm | 101 Md | 102 No

🟥 Ghost states due to different definition of local pseudopotential. SIESTA can't use ABINIT's pseudo.
🟦 SIESTA 2.0.1 internal subroutines can't handle this element (independent of the pseudo used!)
⬛ The cutoff radii of the KB projectors are very long, but SIESTA can use this pseudo.
🟩 The cutoff radius of the neutral atom potential is very long, but SIESTA can use this pseudo.
🟧 Pseudopotentials with and without core electrons promoted to the valence (semicore) available. SIESTA can use these pseudos.
🟪 No corresponding pseudo in ABINIT's database.

Figure 1.1: The pseudopotentials of Abinit which is converted for Siesta (figure courtesy: ICMAB)

A note on Functionals: In Siesta you can specify either LDA or GGA type functionals in a <.fdf> input file. For example, see below for a part of the <fdf> file, which define functionals :

Table 3: Specefication of functional in Siesta FDF file.

```
…

XC.functional       LDA      # Exchange-correlation functional ( GGA can be also used)

XC.authors          CA        # Particular parametrization of xc func.

SpinPolarized       .false.   # Spin unpolarized calculation
```

```
...
```

Table 4: Set of choice of **XC.authors** (with its type) variables.

CA (equivalent to PZ)	A LDA type
PW92	LDA
PW91	GGA
PBE	GGA
revPBE	Modified GGA-PBE
RPBE	Modified GGA-PBE
WC	Modified GGA-PBE
AM05	Modified GGA-PBE
PBEsol	Modified GGA-PBE
PBEJsJrLO	GGA-PBE
BLYP (equivalent to LYP)	GGA
DRSLL (equivalent to DF1)	van der Waals density functional (vdW-DF)
LMKLL (equivalent to DF2)	vdW-DF functional
KBM	vdW-DF functional
C09	vdW-DF functional
BH	vdW-DF functional
VV	vdW-DF functional

For a complete specification on <fdf> input format please read the Siesta Manual (v 4.0 or greater). The DJMol, is currently equipped with a SIESTA 4.1 version (with OpenMP support) and a number of its pre - as well as post-processing tools.

ASE: Atomic Simulation Environment - is a Python based LGPL library which comprises a variety of post processing tools or methods. A general outline of ASE is following (block diagram courtesy: Dr. A H Larsen)

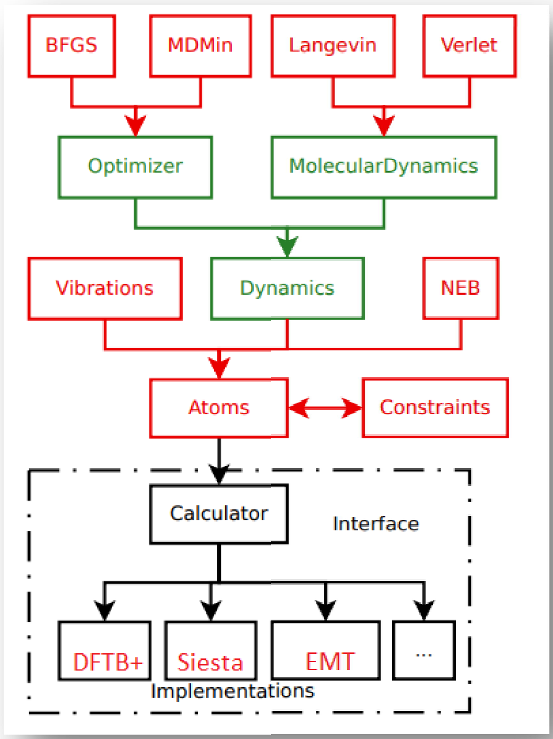


Figure 1.2: Different modules in ASE. Interface denotes the DJMol platform.

One should note that to use ASE, a minimum-level knowledge on Python programming language (version 3.x) is a mandate. See Appendix-A for a basic course on Python programming. ASE's input should written in an object-oriented style, so that the syntax of the input is highly flexible and it can be used to perform a series of jobs, by using, say, a for-loop. Another advantage is, it can be effectively coupled with other python base libraries such as NumPy, Matplotlib etc. to create different types of output files or data set.

Most useful tasks with ASE are: structure optimization (with or without constraints), molecular dynamics, and performing nudged elastic band calculations or its variants. For more details see: *https://wiki.fysik.dtu.dk/ase/*.

OpenMD: it is an MD tool based on force-field type potential and it is widely used for performing MD simulations of *open* systems (such as liquids, proteins, nanoparticles, interfaces, and other complex systems). Note that we used 2.6 version of the program. The code is written in C++ and uses Python for its data pre/post processing. In the DJMol

CygWin based C++ binaries are used and it is kept in ./OpenMD folder with other relevant parameter-files, Python scripts etc.

The OpenMD manual is accompanied with the DJMol and it is available at:

http://openmd.org/wp-content/docs/OpenMD-2.6.pdf

OpenMD uses a range of programs and Python scripts and a user must familiarize with its uses before the production calculation. Examples of OpenMD is illustrated in: *http://openmd.org/category/examples/*

Table 5: Important binaries of OpenMD and its descriptions.

openmd.exe	The Main OpenMD Executable for MD run.
atom2omd.exe	atom2omd attempts to construct .omd files from files containing only atomic coordinate information
Dump2XYZ.exe	Converts an OpenMD dump file into a file suitable for viewing in a molecular dynamics viewer.
DynamicProps.exe	Computes time correlation functions like the velocity autocorrelation function, or the mean square displacement.
elasticConstants.exe	elasticConstants computes the general elastic constants that relate stress and strain for a given input configuration.
Hydro.exe	Hydro generates hydrodynamic resistance tensor (.hydro) files which are required when using the Langevin integrator using complex rigid bodies.
icosahedralBuilder.exe	icosahedralBuilder creates single-component geometric solids that can be useful in simulating nanostructures.
nanoparticleBuilder.exe,	programs to construct nanoparticles of

nanorodBuilder.exe,nanorod_	various sizes and geometries.
omd2omd.exe	omd2omd is a utility script which helps in replicating, rotating, and translating already built OpenMD .omd, .dump, and.eor files.
randomBuilder.exe, simpleBuilder.exe	generate crystals.
recenter.exe	recenter is a utility script which moves all integrable objects in an OpenMD file so that the center of mass is at the origin.
SequentialProps.exe	Computes a time history of static properties from a dump file.
StaticProps.exe	Computes static properties like the pair distribution function.
thermalizer.exe	Thermalizer gives the atoms some initial velocities (at a given temperature) before the equilibration.

Table 4: Short Descriptions the Python scripts.

affineScale.py	OpenMD affine scaling transform Takes an OpenMD file and scales both the periodic box and the coordinates of all StuntDoubles in the system by the same amount. You can either specify a new volume scaling for isotropic scaling, or specify one (or more) of the coordinates for non-isotropic scaling.
dumpConverter.py	Dump File Converter Converts old-style OOPSE md and dump files into new OpenMD style combined files

funcflExtractor.py	Funcfl Extractor Opens an DYNAMO86 EAM funcfl file, parses the data and writes out separate files for F[rho], rho(r), and Z(r)
hbtetAnalyzer.py	It is used for doing analysis on Hydrogen Bond Tetrahedrality matrices
hydroExplainer.py	Computes predicted diffusion constants and rotational relaxation times from a hydro file. Explains the values in the hydro file in terms of properties that can be calculated from a molecular dynamics simulation.
lcorrzFit.py	A utility script to analyze of Legendre correlation functions
omdLast.py	OpenMD File Extractor: Makes omd file from the last good frame of OpenMD dump file.
omdShrink.py	OpenMD File Trimmer: Skips every n frames of an OpenMD dump file and loads it into new dump file
omd-solvator.py	OMD Solvator: Opens two omd files, one with a solute structure and one with a solvent structure. Deletes any solvent molecules that overlap with solute molecules and produces a new combined omd file. The output omd file must be edited to run properly in OpenMD. Note that the two boxes must have identical box geometries (specified on the Hmat line).
omdSplit.py	OpenMD File Splitter: Splits OpenMD dump file frames into separate omd files
pack2omd.py	Packmol RigidBody Replacer: Finds atomistic rigid bodies in a packmol-

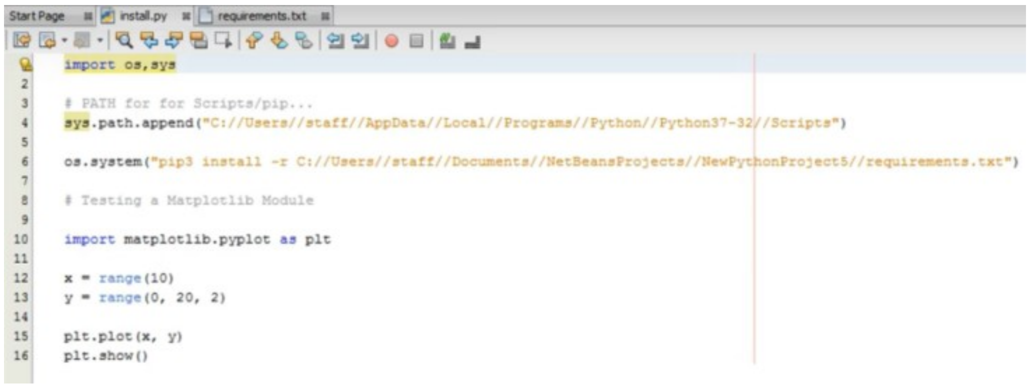
	generated xyz file and generates an OpenMD (omd) file with center of mass and orientational coordinates for rigid bodies.
principalAxisCalculator.py	It opens an XYZ file and computes the moments of inertia and principal axes for the structure in the XYZ file. Optionally rotates the structure so that the long axis (that with the smallest eigenvalue) is pointing along the z-axis.
protonSampler.py	This python script will generate proton disordered configurations of ice-Ih, given an input of the (.xyz) coordinates of the oxygen positions in the lattice.
slabBuilder.py	SlabBuilder to create starting omd files for arbitrary surface cuts specified by Miller indices (hkl) of FCC, BCC and SC materials
slipLength.py	slipLength is a built in analysis script which can compute the slip-length of a solid-liquid interface under shear. The script assumes the solid is placed in the middle of the box, with equal amounts of liquid on either side.
solLiqFricCalc.py	Used for calculating solid liquid friction coefficients
solvator.py	A script that reads in a water box and solute xyz (or pdb), merges the two systems, and deletes overlapping molecules
stat2dielectric.py	A script that computes the static dielectric constant.
stat2dipolecorr.py	A script that computes the system dipole correlation function
stat2tension.py	Used for computing surface tensions from pressure tensors in stat files

stat2thcond.py	Computes the correlation function of the heat flux vector that has been stored in a stat file. These can be used to compute the thermal conductivity.
stat2visco.py	Computes various correlation functions of the pressure and pressure tensor that have been stored in a stat file. These can be used to compute shear and bulk viscosities.
vcorr2spectrum.py	A script that processes a velocity autocorrelation function into a amplitude spectrum Post-processes a vcorr file and creates a normalized spectrum.
waterBoxer.py	builds a lattice of water molecules in a periodic box
waterReplacer.py	Finds atomistic waters in an xyz file and generates an OpenMD (omd) file with center of mass and orientational coordinates for rigid body waters.
waterRotator.py	Samples water orientations from a list of known good orientations
wcorr2spectrum.py	A script that processes a charge velocity autocorrelation function into a amplitude spectrum ;Post-processes a wcorr file and creates a normalized spectrum.

Major Components of the Software

DJMol program is, in essence, a rich client platform (RCP) GUI application that has been built on the top of the RCP framework of the Netbeans IDE. The underlying OOP paradigm permits simple and straight-forward addition and extension of extra modules or plug-ins into its RCP frame work. This application can also be viewed as a *molecular workbench* that integrates many application programs for modeling molecular systems into a single framework. The main components of the GUI are discussed below.

(1) Basic IDE components: As indicated earlier the IDE component, inherited from Netbeans RCP module, is the core part the user-interface. The IDE framework offers a standard set of tools dedicated for advanced programming or scripting tasks. IDE significantly improve productivity of its users since it automates several text based processes (code completion, version control, project management, restoring old versions of the text files etc.). Several of native IDE menus are integrated; e.g. the *File* menu selects a coordinate data by its file extension (eg. xyz or hsd) to visualize the structure and open its text to the text editor of the IDE. Note that in the current version of the program xyz or hsd files are selected as the default file extensions for the viewer. The selected file can be manipulated at different levels by using its *Edit*, *View* and *Source* menus. Other basic IDE features like refactoring of files, Git facility, visual *diff* utility (to compare the contents of two text files) etc. are also supported. Project Explorer - which is placed on the far left side - shows the packages (directories and it files) that make up Python projects; It can also be used to systematically arrange or manipulate other data. *Files* and *Favorite* explorer panels are two tools for managing files (including binaries) in general. See Fig A for the IDE text editor which displays a Python script.



```
import os,sys

# PATH for for Scripts/pip...
sys.path.append("C://Users//staff//AppData//Local//Programs//Python//Python37-32//Scripts")

os.system("pip3 install -r C://Users//staff//Documents//NetBeansProjects//NewPythonProject5//requirements.txt")

# Testing a Matplotlib Module

import matplotlib.pyplot as plt

x = range(10)
y = range(0, 20, 2)

plt.plot(x, y)
plt.show()
```

Figure 1.3: A Python script is displayed in the IDE text editor with syntax highlighting option.

(2) Top Component Window: The center panel of the software is set to display molecules and it embeds an unabridged Jmol visualizer binary and it is used in the *TopComponent* class (the basic unit of RCP display). Optionally a *scripting* tool (for Jmol scripting) can be invoked from this panel to create more sophisticated and customized molecular visualizations (See Table 5).

Table 5: Some selected one-liner commands of Jmol scripts and its descriptions.

`load =cod/1000373 {2,2,2}`	Loads 2x2x2 supercell of $NaVO_2F_2$ from COD database
`minimize`	Molecular mechanics (UFF) steepest descent minimization
`write "C:\\file.pdb"`	Save the molecule into a PDB file
`isosurface cutoff 0.01 "C:\\cube.gz"`	Visualize an isosurface from a gzipped CUBE file
`q=quaternion()`	Saving orientation in quaternion rather than Euler angles

(3) 2D and 3D Data Windows: JFreeChart library (a Java library for displaying various types of scientific data) is used to create a 2D graphs and it is usually projected onto a *JPanel* (a Swing GUI widget toolkit) to display the plots. By this, one can add more control options to the rendered graphs (See, Fig-D for a 2D plot with more control options). It is the *default* 2D data plot library of the DJMol and it has several graphic styles and it offers many configuration options to customize the rendering of graphics and it export the graphic content in PNG format. Apart from this, Matplotlib library is also used to display 2D graphics especially with Siesta add-on. The 3D graphics (scalar volumetric data such as density isosurface) windows all are embedded with Jmol application with some internal Jmol script commands. To manipulate the 3D data either Jmol scripts or Pythons scripts are used. For example, a Python script is used to convert xsf file format (which is frequently used in Siesta) into the cube format.

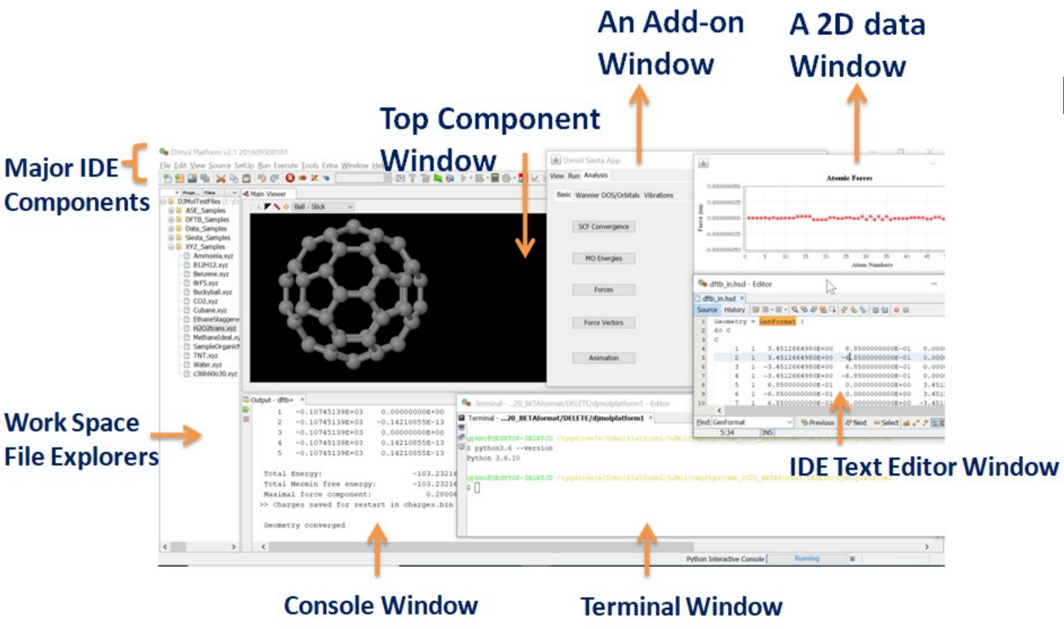
(4) Terminal Window: One can run OS commands and Python scripts with this window. Note that in the Windows version of the program the CygWin environment is automatically linked with this terminal. If necessary, WSL (a compatibility layer for

running native Linux executables under Windows 10 OS) can also be linked. See the manual for more details. The plug-in for the Python integration into the software can be switched to get Python 2.x as well as Python 3.x. It gives the terminal based (interactive mode) access to the language. As an example, by using this terminal one can install many additional packages like ASE or Matplotlib library. Using SSH commands of CygWin or WSL this terminal can be connected with other remote machines.

(5) Console Window: This window is usually placed immediately below the Top Component window and it is mainly used to display console output text (eg. to show DFTB+ output when it runs). It also contains Start and Stop buttons to execute the DFTB+ program. The log text from this window can be saved or post processed, optionally.

(6) Add-on Windows: Other major window components are from the add-on programs based on Swing API. Currently there are six add-ons are constructed and all these programs are independent of each other. And these add-ons are built from *JFrame* – the base container of the application. At present it supports all the AWT (abstract window toolkit) components. Although more recent *JavaFX* class to create desktop applications exist in Java, we always used Swing based GUIs since more libraries and tools are available for Swing API.

Schematically all these key GUI components are shown in the figure 1.4 .



```
     1   -0.10745139E+03    0.00000000E+00
     2   -0.10745139E+03   -0.14210855E-13
     3   -0.10745139E+03    0.00000000E+00
     4   -0.10745139E+03    0.14210855E-13
     5   -0.10745139E+03    0.14210855E-13

 Total Energy:                          -103.2321
 Total Mermin free energy:              -103.2321
 Maximal force component:                 0.2800
>> Charges saved for restart in charges.bin

 Geometry converged
```

```
Geometry = GenFormat {
60 C
C
      1    1    3.4512664980E+00    6.8500000000E-01    0.000
      2    1    3.4512664980E+00   -6.8500000000E-01    0.000
      3    1   -3.4512664980E+00    6.8500000000E-01    0.000
      4    1   -3.4512664980E+00   -6.8500000000E-01    0.000
      5    1    6.8500000000E-01    0.0000000000E+00    3.451
      6    1   -6.8500000000E-01    0.0000000000E+00    3.451
      7    1    6.8500000000E-01    0.0000000000E+00   -3.451
```

```
Terminal - ...20_BETAformat/DELETE/djmolplatform1

$ python3.6 --version
Python 3.6.10

$ 
```

Figure 1.4: A schematic classification of major window components which constitute the entire GUI of DJMol program.

2

DFTB+ CALCULATIONS

Here we illustrate some standard procedures that can be performed by the DFTB+ program in conjunction with DJMol platform. At present there are three distinct file types can be opened for DFTB+ calculations, viz., [1] dftb_in.hsd, <.xyz> and <.gen> files. The first one is the default unique file type for the program and the last two file extensions can be converted into <dftb_in.hsd> file by using a script writer (see below section).

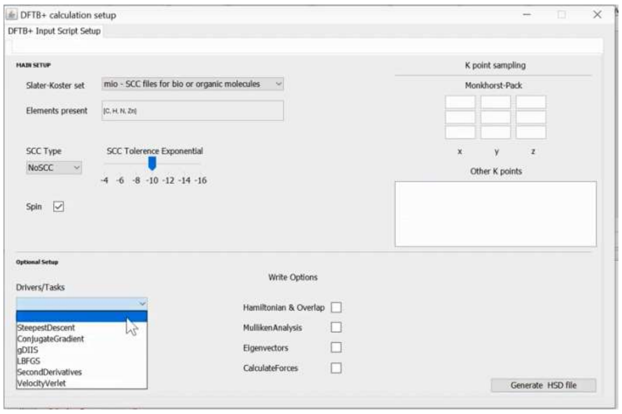
[1] DFTB+ SCRIPT WRITER

If one opens a <.xyz> or <.hsd> file, for visualizing the structure, subsequently one can also try to make a corresponding <dftb_in.hsd> file using a DFTB+ script writer program. Using this, a user can make a basic <dftb_in.hsd> script for a range of jobs, including, geometry optimization, calculating excited states or Hessian etc. It can be retrieved by:

Set Up ➢ DFTB Scripting

And the generated <dftb_in.hsd> file is kept in <**Input**> folder. Note that the atomic information including its geometry is directly taken from <gen> data of the currently loaded file.

Note that, the Script-Writer module will write <TEMPdftb_in.hsd> into the <Input> folder. It is strongly recommended that, a user should always try to inspect/modify this script before its submission, if it is necessary.


DFTB+ calculation setup
DFTB+ Input Script Setup

MAIN SETUP
Slater-Koster set: mio - SCC files for bio or organic molecules
Elements present: [C, H, N, Zn]
SCC Type: NoSCC
SCC Tolerence Exponential: -4 -6 -8 -10 -12 -14 -16
Spin ✓

K point sampling
Monkhorst-Pack
x y z
Other K points

Optional Setup
Drivers/Tasks: SteepestDescent, ConjugateGradient, gDIIS, LBFGS, SecondDerivatives, VelocityVerlet
Write Options: Hamiltonian & Overlap ☐, MullikenAnalysis ☐, Eigenvectors ☐, CalculateForces ☐
Generate HSD file

Figure 2.1: A screen shot of the script writer tool for the DFTB+ program.

[2] Executing DFTB+ Scripts

Running DFTB+ calculator is possible when you load a <dftb_in.hsd> file. Make sure that the <Input> folder should have this file before the execution and this file can be executed as:

Execute ➢ Run DFTB+

And if necessary, this Run can be killed by

Execute ➤ Stop DFTB+

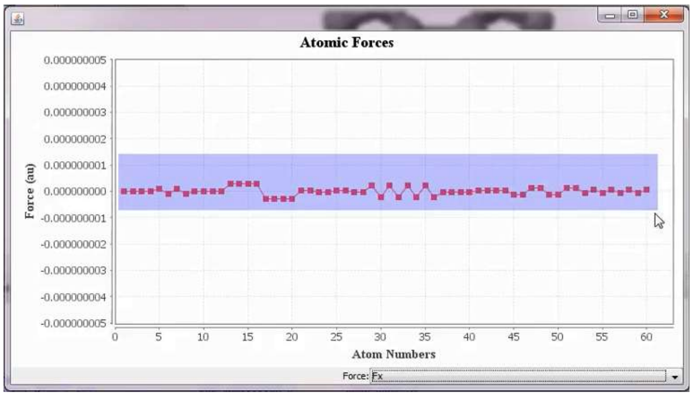
The generated/loaded files are usually placed in the` <DFTB_Scratch>filder. And these files are also sent back as a Zip file, to the directory where your current <hsd> file resides.

DFTB+ generated data can be analyzed or visualized by the program. Main data analyzers are given below.

[3] Force Components

DFTB+ out file contains information about Cartesian force components (in atomic units) and this can be visualized as a line chart as shown below. The <detailed.out> file can be opened by:

Tools ➤ ForceComponents



Force: Fx

Figure 2.2: The ***x*** components of the Cartesian forces of the Fullerene molecule after the geometry optimization. The shaded portion can be easily zoomed in for more details.

[4] Molecular Orbitals and Energy Levels

MO information (including density and density difference files) are stored in the .CUBE format. In order to get cube files one needs to ensure that dftb_in.hsd file contains relevant keywords (*WriteDetailedXML* and *WriteEigenvectors*, see example directory for a template file). After running this file it will produce 3 out files (detailed.xml, charges.bin, and eigenvec.bin) and these files should be transferred from <DFTB_Scratch> directory to <ModesBinary> folder. Then invoke:

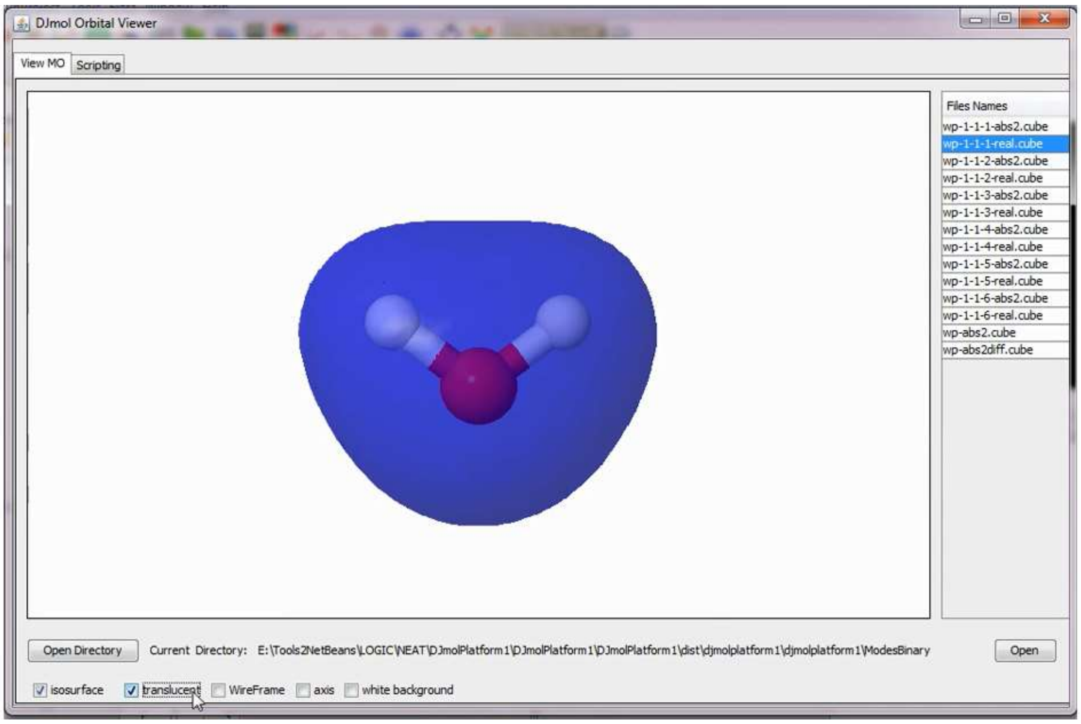
Execute ➢ Run Waveplot

From this window, select the proper SK file set (look at the combo box entitled, *Specify SK set and Generate HSD file*) and press on the button <**Generate Waveplot File**>. This will produce the needed HSD file for the Waveplot.exe binary file. View this file from the second tab and edit it if needed. Then invoke the <**Calculate**> button to produce the necessary CUBE file. Note that it might take some minutes to complete the task (depending on the size of the system).

To visualize the cubes file, first copy the Full Path of the folder, <ModesBinary> and then go to:

Tools ➢ View Cube

Then click on the button, <**Open Directory**> and insert the recently copied Full Path into it and select a CUBE file. This will list all the CUBE files into the table, and this can be visualized easily by selecting the needed file name from the table and by clicking on the <**Open**> button. Optionally one can also use *JMol scripting* facility for more advanced file visualizations or manipulations.



wp-1-1-1-abs2.cube
wp-1-1-1-real.cube
wp-1-1-2-abs2.cube
wp-1-1-2-real.cube
wp-1-1-3-abs2.cube
wp-1-1-3-real.cube
wp-1-1-4-abs2.cube
wp-1-1-4-real.cube
wp-1-1-5-abs2.cube
wp-1-1-5-real.cube
wp-1-1-6-abs2.cube
wp-1-1-6-real.cube
wp-abs2.cube
wp-abs2diff.cube

Current Directory: E:\Tools2NetBeans\LOGIC\NEAT\DJmolPlatform1\DJmolPlatform1\DJmolPlatform1\dist\djmolplatform1\djmolplatform1\ModesBinary

[✓] isosurface [✓] translucent [ ] WireFrame [ ] axis [ ] white background

Figure 2.3: Visual of one of the MOs from the list of CUBE files.

To view only MO energy levels and its degeneracies, one can use the <detailed.out> file and this can be called by:

Tools ➢ MO Levels

The green lines represent occupied and red lines represent the un occupied levels. For accurate calculations (which use tight SCC parameters) one can also look at the degeneracies levels.

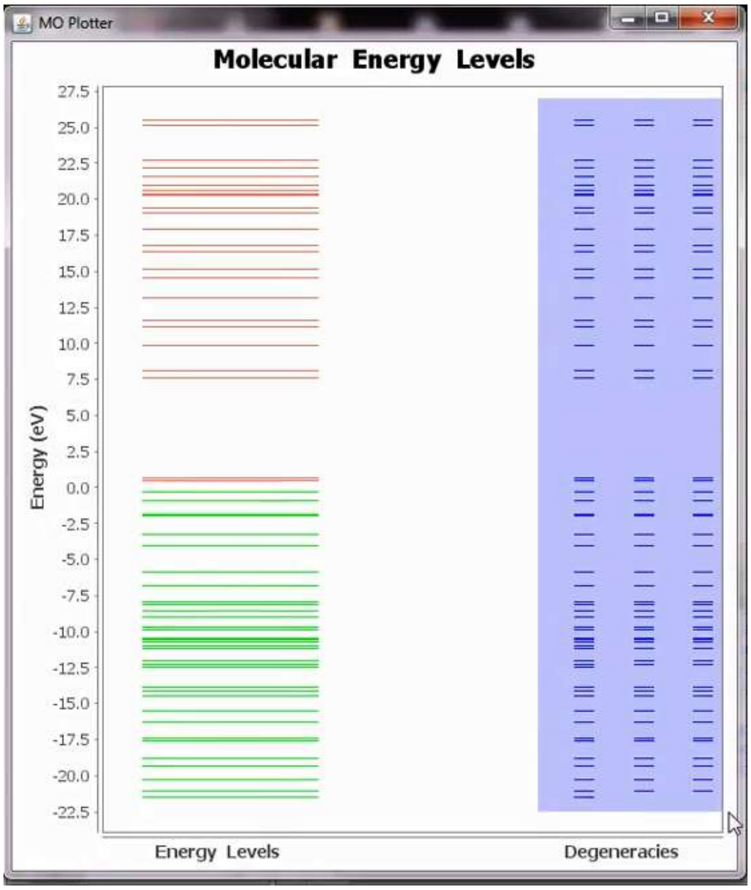


Figure 2.4: MO energy levels from the detailed.out file.

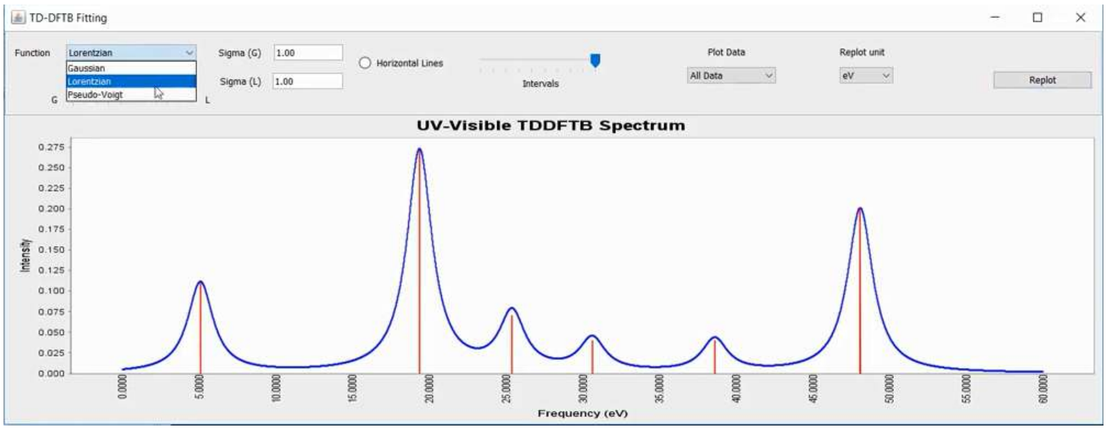
[5] UV-Visible Spectrum Convolution

DFTB+ uses TD-DFTB for calculating oscillator strength of the optical excitations and this is stored in **SPX.dat** file. An add-on is used for reading this data and this can be invoked by:

Tools ➤ UV-Vis Spectrum

Select a **SPX.dat** file to proceed. This will plot the oscillator strengths (vertical red lines) and its fitted function (broadened through convolution with a Gaussian, Lorentzian function or its linear combination, known as pseudo-Voigt functions - shown by blue curves). By default it uses *eV* however one can also obtain an equivalent spectrum using *nm* units (this creates a separate plot). Note that the oscillator strength is *dimensionless* quantity and it is described in the *y* axis.

By right clicking the plot, one can adjust several parameters (Auto Range, Zoom, Grid Lines, Axes properties etc.) of the figure and can save the image into PNG format.


TD-DFTB Fitting

Function: Lorentzian

Sigma (G): 1.00

Sigma (L): 1.00

Plot Data: All Data

Replot unit: eV

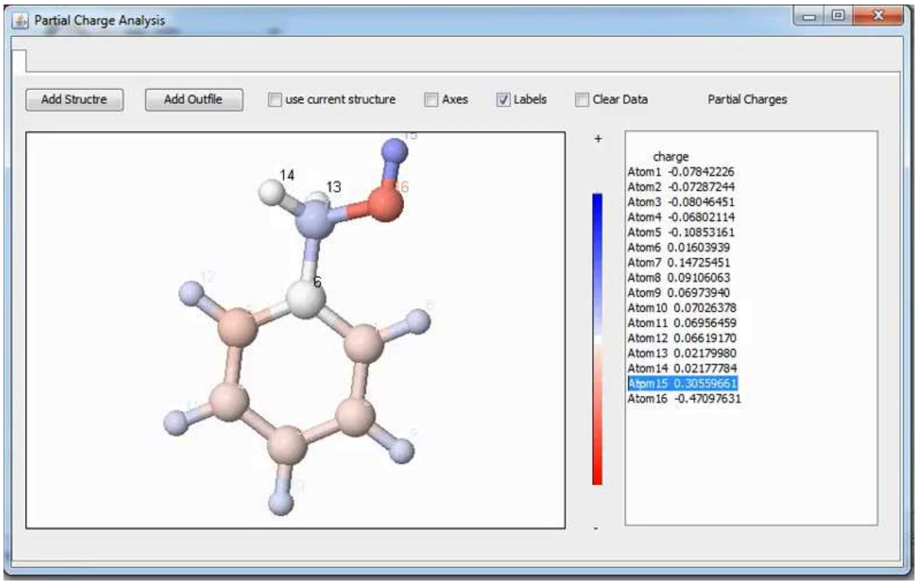
Figure 2.5: UV-Visible spectrum of an organic molecule fitted with Lorentzian convolution.

[6] Partial Charges

DFTB+ caclulates partial charges and this charges can be projected out into the individual atoms. This Utility can be invoked by:

Tools ➢ Partial Charges

This utility read both the XYZ geometry and its corresponding *detailed.out* file. A color scale will also be shown for a reference. Note that this will generate a file and this file be saved for a reference (See Appendix section for the **Saved Data**).



```
    charge
Atom1  -0.07842226
Atom2  -0.07287244
Atom3  -0.08046451
Atom4  -0.06802114
Atom5  -0.10853161
Atom6  0.01603939
Atom7  0.14725451
Atom8  0.09106063
Atom9  0.06973940
Atom10  0.07026378
Atom11  0.06956459
Atom12  0.06619170
Atom13  0.02179980
Atom14  0.02177784
Atom15  0.30559661
Atom16  -0.47097631
```

Figure 2.6: Numerical values of Mulliken's partial charges are represented on atoms

[7] Molecular Vibrations

By using Harmonic approximation DFTB+ calculates the vibrational normal modes and frequencies. This requires Hessian Matrix (calculated by the DFTB+) and a script file for modes.exe binary (which calculates modes and frequencies).

The first step is to optimize the geometry. And once the optimized geometry is ready one can do the *vibrational calculation* with the necessary keywords (it includes Driver = SecondDerivatives{} keywords; See the example directory). This will generate the needed *hessian.out* file for the *modes.exe.*

To calculate vibrational modes vectors, to visualize/animate these modes and to get the frequencies, invoke the tool as:

Execute ➢ Run Modes

Here in this tool one has to enter the optimized GEN styled geometry

```
Geometry= GenFormat {
   ...
}
```

and the hessian data as:

```
hessian = {
...
}
```

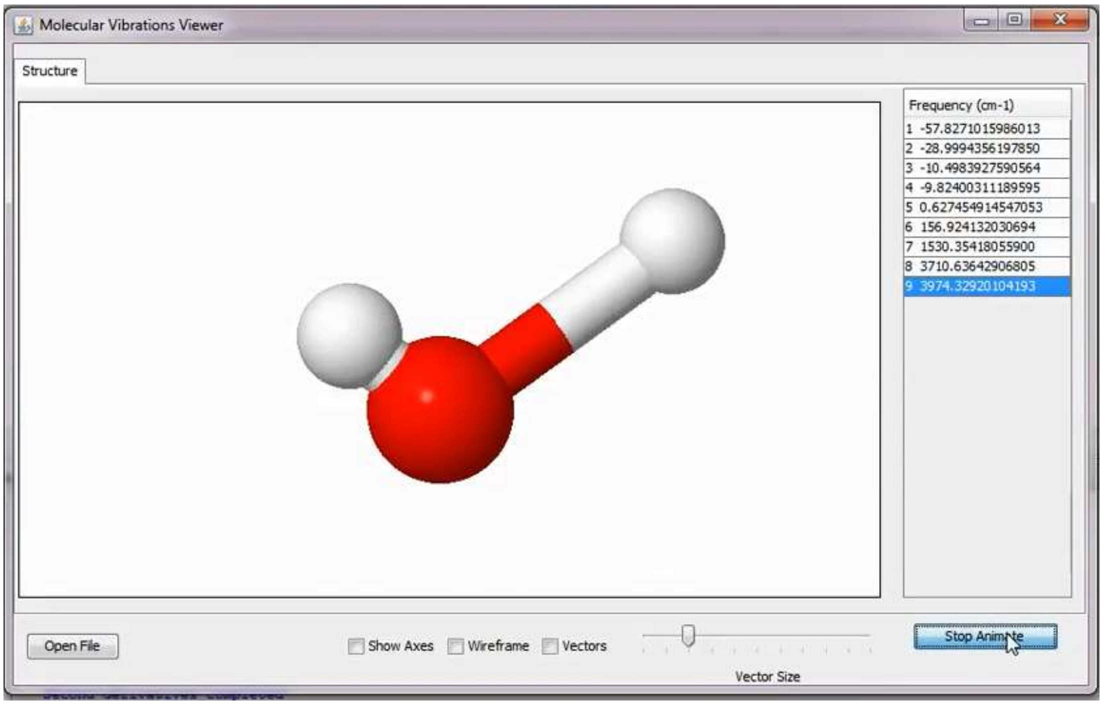
Note that the dots (...) represents the numerical value of the Hessian matrix ie. ALL the numerical value of the file , hessian.out, which is **3Nx3N** matrix data (force constant matrix).

After these two data entry select the corresponding SK file set and then click on the button: ***Generate Script.*** This will generate modes_in.hsd file in the ModesBinary folder. This file contains all the information for the modes.exe program which *diagonalizes* the hessian matrix to give modes/frequency information. The button, **Generate Modes/Frequencies** initiate this operation and produces the file, modes.xyz which contains the frequency/modes information.

To view this file another application is used this can be called by:

Tools ➤ View Modes

This will read the files modes.xyz and visualize the vectors/animations etc. Note that the Hessian data is used by the modes.exe binary *as such*. In other words it does not projects out translational and rotational motions. So that the user should be optimize the molecule well enough. If needed see *http://gaussian.com/vib/* on the projection technique of Hessian matrix (Sayvetz conditions).



	Frequency (cm-1)
1	-57.8271015986013
2	-28.9994356197850
3	-10.4983927590564
4	-9.82400311189595
5	0.627454914547053
6	156.924132030694
7	1530.35418055900
8	3710.63642906805
9	3974.32920104193

Figure 2.7: Vibrational modes of water molecules read from the file modes.xyz in the ModesBinary directory.

[8] REAL TIME MOLECULAR DYNAMICS

MD simulations can be done with or without using this tool. Normally one can submit MD job as usual (like a static calculation job). However, this MD tool is mainly used in real time so that a user can analyze Trajectory properties (Total Energy, Kinetic/Potential Energies, structure of each step) at Run Time. Moreover FFT based autocorrelation functions can be used to calculate IR spectra (from the dipole moments or from the velocity autocorrelation functions) after finishing the MD run. Note that, the MD analyzer tool should be invoked *before* the DFTB+ calculation. While running the MD calculation it will show

the progress of the data along with the current structure. Its step are:

[1] Remove ***all*** the files from ./DFTB_Scratch directory. It is Mandate.

[2] Open MD file as, File ➢ Open (hsd)

[3] Invoke the MD tool as, Tools ➢ MD Analyzer

[4] Execute MD run as, Execute ➢ Run DFTB+

[5] Wait for the MD calculation to finish; and after that use tools like, <Velocity Autocorrelation Spectra>.

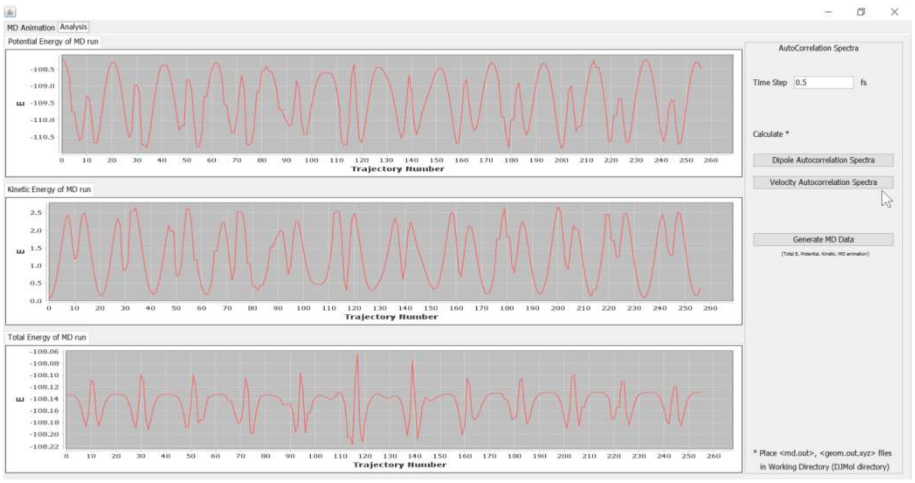

MD Animation
Analysis
Potential Energy of MD run
E
Trajectory Number
Kinetic Energy of MD run
Total Energy of MD run
AutoCorrelation Spectra
Time Step
0.5
fs
Calculate *
Dipole Autocorrelation Spectra
Velocity Autocorrelation Spectra
Generate MD Data
(Total E, Potential, Kinetic, MD animation)
* Place <md.out>, <geom.out.xyz> files in Working Directory (DJMol directory)

Figure 2.8: Variations of energies during the MD, in real time.

[9] POTENTIAL ENERGY SURFACES (2D/1D)

The DJMol creates 2D PES (potential energy surface) by performing a series of DFTB+ calculations. For example, PES of H_2O_2 (hydrogen peroxide) molecule is created by doing a batch process. To create 2D PES we need two independent internal coordinate variables and, arbitrarily, we have chosen the O-O bond length and the H-H dihedral angle (in degrees) as shown in Fig. X.

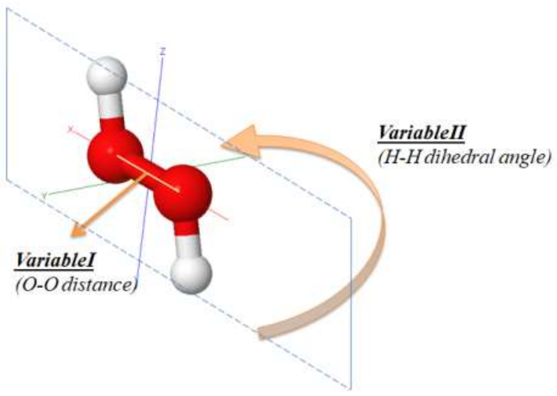
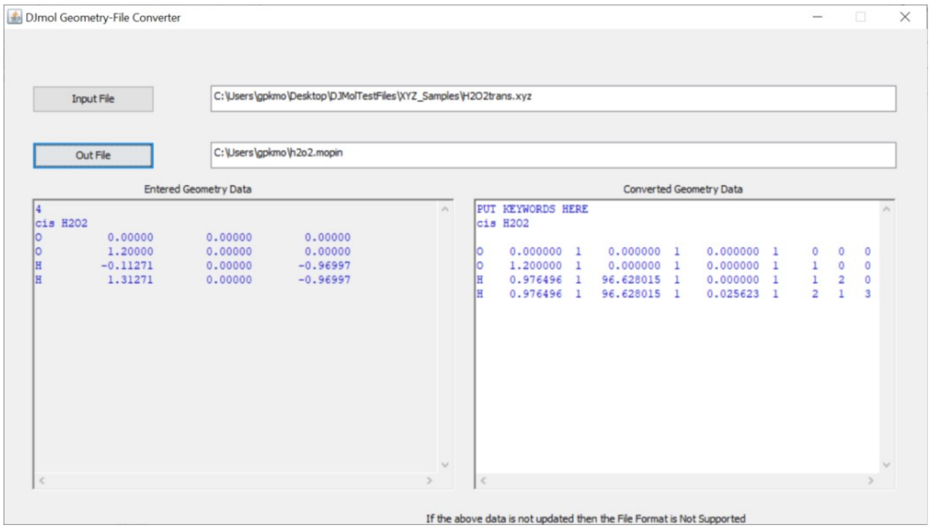


Figure 2.9: The two independent variables of H_2O_2 PES. The unit of bond length is in Angstrom and that of the dihedral angle is in degrees.

In first, make the structure of the molecule (or see Example folder for its coordinates); then using the <Convert Structure> tool (See Fig.2), convert this XYZ formatted file into the MOPIN format (which contains an equivalent z-matrix data). And copy the content of this file into <Build PES HSD> tool as shown in the Fig. 3.

By clicking the button, <Generate PES HSD Files>, this will generate 441 HSD input files for the DFTB+ program (it will take around ten minutes); copy all of these HSD files into the <SCRATCH> folder (a user can also select any other directory instead of this one) and then invoke the <Batch Processing Tool>. This will start DFTB+ program to execute all these files and create the PES in the <PESScan> folder.



C:\Users\gpkmo\Desktop\DJMolTestFiles\XYZ_Samples\H2O2trans.xyz

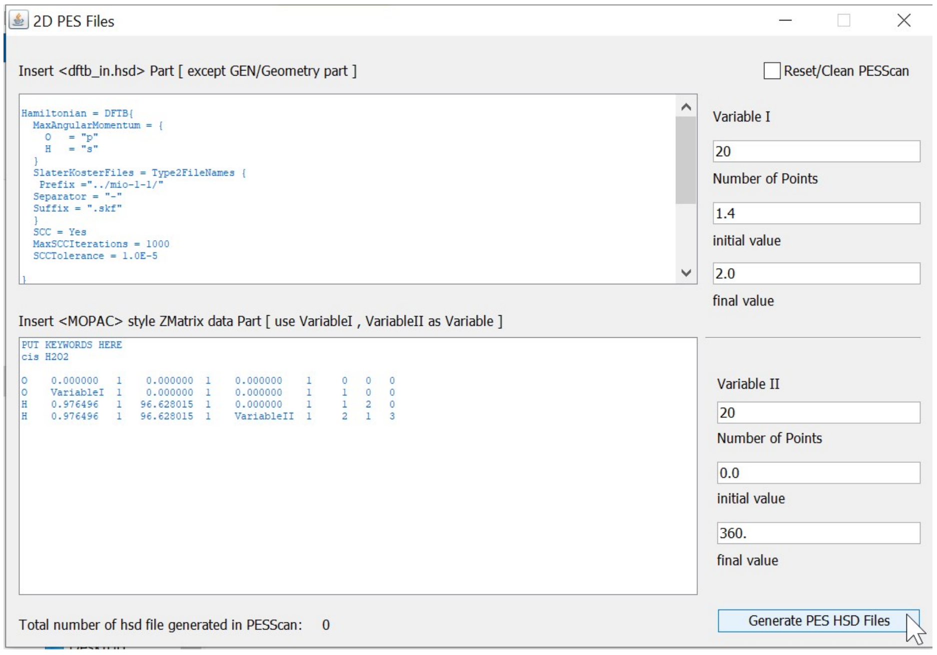
C:\Users\gpkmo\h2o2.mopin

```
4
cis H2O2
O        0.00000       0.00000       0.00000
O        1.20000       0.00000       0.00000
H       -0.11271       0.00000      -0.96997
H        1.31271       0.00000      -0.96997
```

```
PUT KEYWORDS HERE
cis H2O2

O    0.000000  1    0.000000  1    0.000000  1     0   0   0
O    1.200000  1    0.000000  1    0.000000  1     1   0   0
H    0.976496  1   96.628015  1    0.000000  1     1   2   0
H    0.976496  1   96.628015  1    0.025623  1     2   1   3
```

If the above data is not updated then the File Format is Not Supported

Figure 2.10: The Convert tool is used to transform Cartesian file into a Mopin file (which essentially contain the z-matrix in Mopac's format).



Insert <dftb_in.hsd> Part [except GEN/Geometry part]

```
Hamiltonian = DFTB{
  MaxAngularMomentum = {
    O   = "p"
    H   = "s"
  }
  SlaterKosterFiles = Type2FileNames {
   Prefix ="../mio-1-1/"
  Separator = "-"
  Suffix = ".skf"
  }
  SCC = Yes
  MaxSCCIterations = 1000
  SCCTolerance = 1.0E-5

}
```

Insert <MOPAC> style ZMatrix data Part [use VariableI , VariableII as Variable]

```
PUT KEYWORDS HERE
cis H2O2

O    0.000000   1    0.000000  1    0.000000    1     0   0   0
O    VariableI  1    0.000000  1    0.000000    1     1   0   0
H    0.976496   1   96.628015  1    0.000000    1     1   2   0
H    0.976496   1   96.628015  1    VariableII  1     2   1   3
```

Variable I
20
Number of Points
1.4
initial value
2.0
final value

Variable II
20
Number of Points
0.0
initial value
360.
final value

Total number of hsd file generated in PESScan: 0

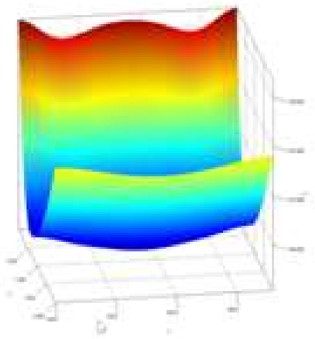
Figure 2.11: The converted MOPIN files content is inserted in the below text area and insert the variable name in the appropriate position. VariableI represents the O-O bond length (starts from 1.2Å) and VariableII, it H-H dihedral angle (starts with 0.0^o).

After finishing the batch process, <PESViewer> can be invoked to get the plot of 2D PES or its contour diagram as shown in the Fig. 4. This utility will also indicates the minimum energy data point on the PES (in other words from this point one can start geometry optimization to locate local minimum) and its corresponding geometry file.

In <PESScan> folder one will see the following data files (See Table 6).

Table 6: Datafiles of PES calculations.

View2DPES.dat	*Gives the data intervals in x and y directions and energy as E(x,y) in eV*
2DPESscan.dat	*It contains the sorted file names used in the PES creation with energy (in eV and in Hartree).*
2DPESscanmovie.xyz	*Contains the geometry of the individual molecules in xyz format. It can be visualized/animated with DJMol. This can be userd to construct PES with other ab initio programs.*



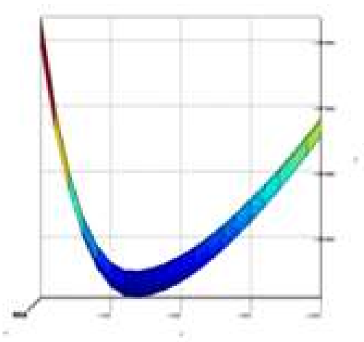


Figure 2.12: The PES of the cis-trans conversion of H_2O_2 . Note that the optimum bond length is around 1.48 Å and for shorter bond lengths the energy minimum is located near to 90.0o or 270.0^o.

3

MODELING WITH SIESTA

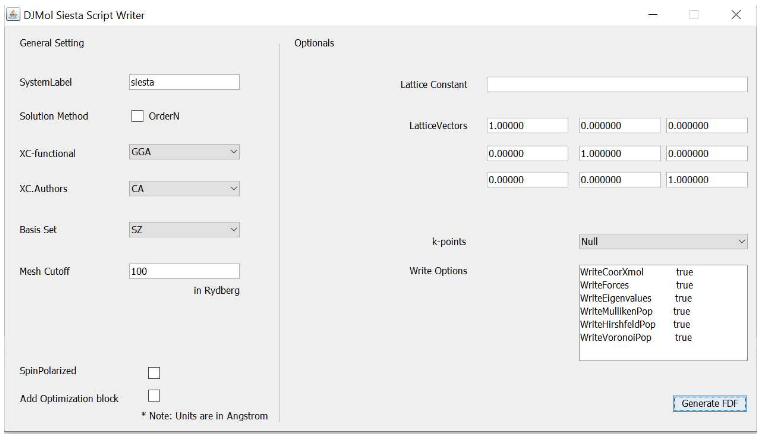
This add-on is mainly used for performing small to moderate level Siesta DFT calculations (with hundred of atoms). For larger calculations users are advised to use cluster/GCP machines with MPI parallel binaries. This add-on is equipped with OpenMP so that multi core machines can be used for its calculations, if it is found necessary. Note that Siesta 4.1 version is used in this add-on.

At the present version all molecular structural data and non Z-matrix styled crystalline FDF data is supported by this add-on.

[1] SIESTA SCRIPT WRITER

A basic Siesta script (.fdf file) write is written; it can be used to make a template FDF file for molecular as well as crystalline systems. Most commonly used key words are added and this can be selected optionally. However user is advised to refer Siesta 4.1 manual for finalizing the script. This tool can be invoked as:

Set Up ➢ Siesta Scripting


DJMol Siesta Script Writer

General Setting

SystemLabel	siesta
Solution Method	☐ OrderN
XC-functional	GGA
XC.Authors	CA
Basis Set	SZ
Mesh Cutoff	100 in Rydberg
SpinPolarized	☐
Add Optimization block	☐

* Note: Units are in Angstrom

Optionals

Lattice Constant			
LatticeVectors	1.00000	0.000000	0.000000
	0.00000	1.000000	0.000000
	0.00000	0.000000	1.000000
k-points	Null		

Write Options

WriteCoorXmol	true
WriteForces	true
WriteEigenvalues	true
WriteMullikenPop	true
WriteHirshfeldPop	true
WriteVoronoiPop	true

Figure 3.1: A script writer tool (FDF writer) in the Siesta add-on.

The newly generated file is kept in: **.\Input** directory.

If an XYZ file is loaded into the Main Viewer, FDF file will generated with that geometry.

[2] Siesta Add-On and Executing the Program

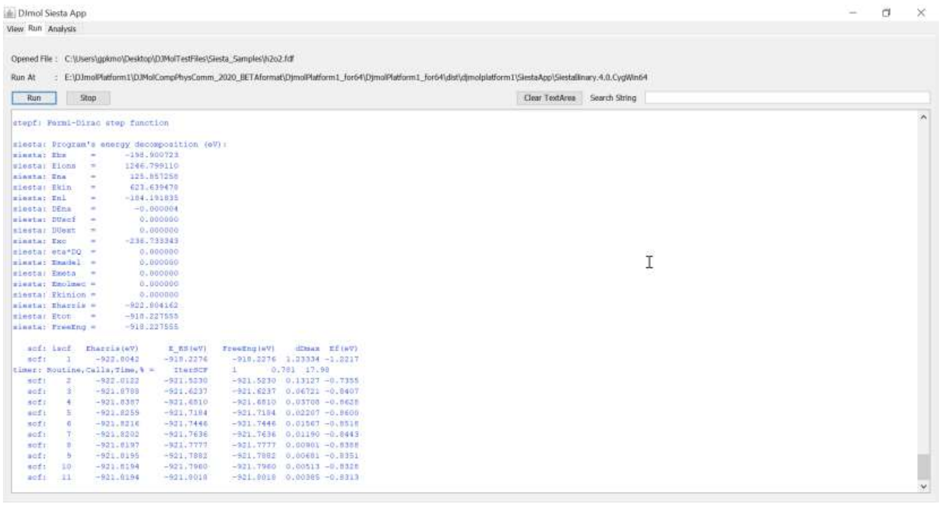
All the basic tasks for Siesta (except its script writing) is carried out in Siesta add-on and it can be called by:

Execute ➢ Siesta Calculator

It will launch the add-on as a separate program. A variety of FDF file is supported by the *Jmol* and hence this add-on. Use <Open> for calling these FDF files.

For example, h2o2.fdf can be opened and it can be submitted in this application, by simply going to the tab <Run> and clicking the Run button. The below figure shows a console output of a Siesta run. Note that the output file is stored in,

.\SiestaApp\SiestaBinary.4.0.CygWin64 directory. And its console log file is saved in **.\SiestaApp\siestaMAIN.log.**



Opened File : C:\Users\gpkmo\Desktop\DJMolTestFiles\Siesta_Samples\h2o2.fdf

Run At : E:\DJmolPlatform1\DJMolCompPhysComm_2020_BETAformat\DjmolPlatform1_for64\DjmolPlatform1_for64\dist\djmolplatform1\SiestaApp\SiestaBinary.4.0.CygWin64

```
stepf: Fermi-Dirac step function

siesta: Program's energy decomposition (eV):
siesta: Ebs     =      -198.900723
siesta: Eions   =      1246.799110
siesta: Ena     =       125.857258
siesta: Ekin    =       623.639470
siesta: Enl     =      -184.191835
siesta: DEna    =        -0.000004
siesta: DUscf   =         0.000000
siesta: DUext   =         0.000000
siesta: Exc     =      -236.733343
siesta: eta*DQ  =         0.000000
siesta: Emadel  =         0.000000
siesta: Emeta   =         0.000000
siesta: Emolmec =         0.000000
siesta: Ekinion =         0.000000
siesta: Eharris =      -922.004162
siesta: Etot    =      -918.227555
siesta: FreeEng =      -918.227555

   scf: iscf   Eharris(eV)      E_KS(eV)   FreeEng(eV)    dDmax  Ef(eV)
   scf:    1     -922.0042     -918.2276     -918.2276  1.23334 -1.2217
timer: Routine,Calls,Time,% =    IterSCF     1       0.781  17.98
   scf:    2     -922.0122     -921.5230     -921.5230  0.13127 -0.7355
   scf:    3     -921.8788     -921.6237     -921.6237  0.06721 -0.8407
   scf:    4     -921.8387     -921.6810     -921.6810  0.03708 -0.8628
   scf:    5     -921.8259     -921.7184     -921.7184  0.02207 -0.8600
   scf:    6     -921.8216     -921.7446     -921.7446  0.01567 -0.8518
   scf:    7     -921.8202     -921.7636     -921.7636  0.01190 -0.8443
   scf:    8     -921.8197     -921.7777     -921.7777  0.00901 -0.8388
   scf:    9     -921.8195     -921.7882     -921.7882  0.00681 -0.8351
   scf:   10     -921.8194     -921.7960     -921.7960  0.00513 -0.8328
   scf:   11     -921.8194     -921.8018     -921.8018  0.00385 -0.8313
```

Figure 3.2: Siesta log is shown in the console window of the add-on.

In <Analysis> some standard or routine calculations or analysis can be done, such as checking SCF convergence, Cartesian Force components and vectors (of optimized structure), Eigenvalues etc. from the relevant files in **.\SiestaApp\SiestaBinary.4.0.CygWin64.** See the below figure of h2o2.fdf out file examples.

[3] BASIC ANALAYZERS

After running the calculations, one can use many basic analyzers, for example to check SCF convergence for single point as well as geometry optimization etc.

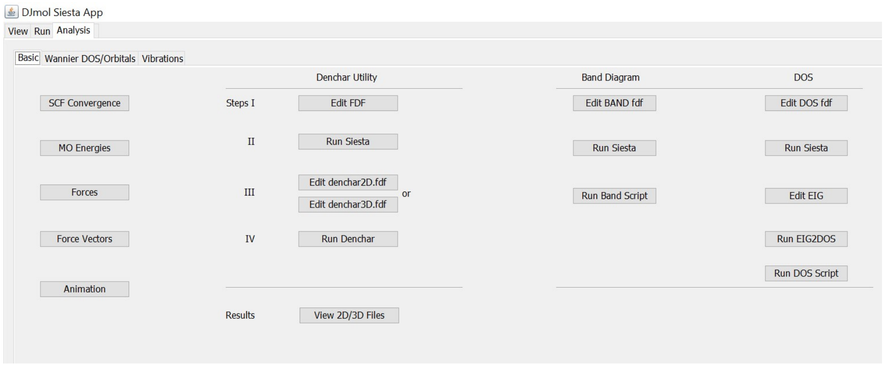


Figure 3.3: The window shows basic commands for the Siesta post processing.

Some of the analyzers results are shown below.

- ***Basic Analyzers Results:***

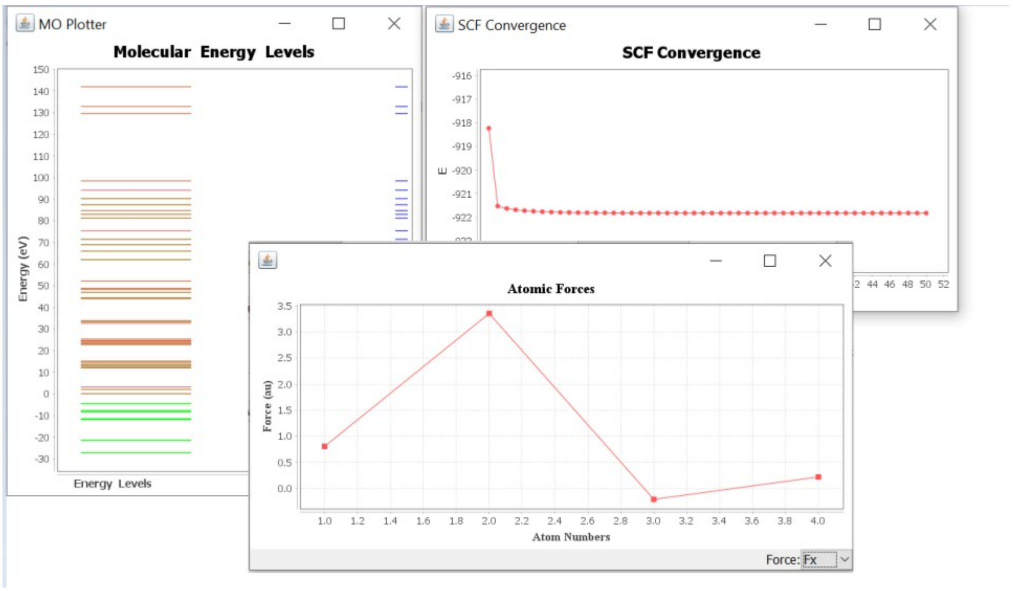


Figure 3.4: Results from the Siesta post processing is shown.

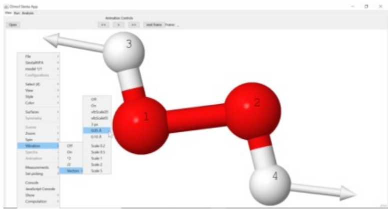
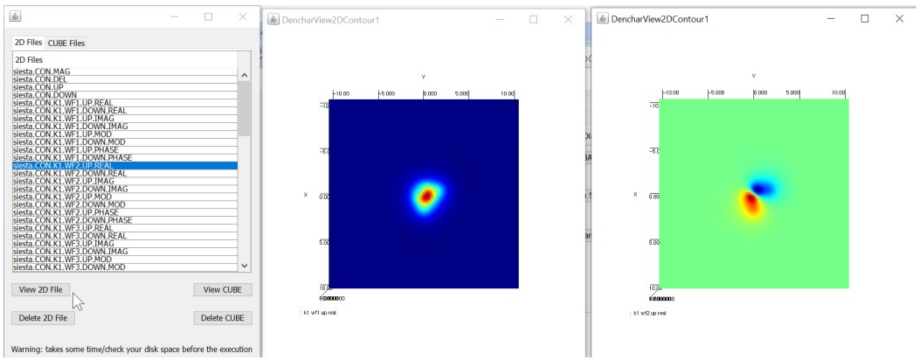


Figure 3.5: Force component vectors of each atom in H_2O_2 (non equilibrium geometry).

- ***Denchar Results***

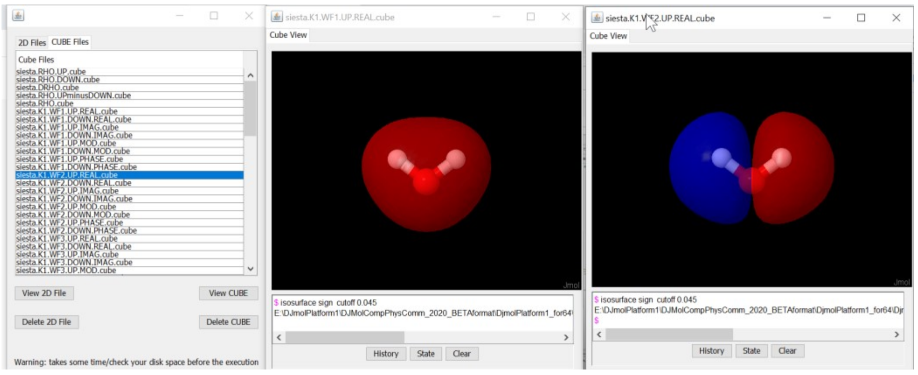
Denchar is a Siesta utility-program to plot charge densities and wave functions in real space. It can be used in 2D or in 3D. A template script for water molecule is given (but this can be modified very easily for other molecules), and its 2D/3D data can be obtained by following Steps I-IV, systematically. Note that Γ point used in this calculations.



```
2D Files
siesta.CON.MAG
siesta.CON.DEL
siesta.CON.UP
siesta.CON.DOWN
siesta.CON.K1.WF1.UP.REAL
siesta.CON.K1.WF1.DOWN.REAL
siesta.CON.K1.WF1.UP.IMAG
siesta.CON.K1.WF1.DOWN.IMAG
siesta.CON.K1.WF1.UP.MOD
siesta.CON.K1.WF1.DOWN.MOD
siesta.CON.K1.WF1.UP.PHASE
siesta.CON.K1.WF1.DOWN.PHASE
siesta.CON.K1.WF2.UP.REAL
siesta.CON.K1.WF2.DOWN.REAL
siesta.CON.K1.WF2.UP.IMAG
siesta.CON.K1.WF2.DOWN.IMAG
siesta.CON.K1.WF2.UP.MOD
siesta.CON.K1.WF2.DOWN.MOD
siesta.CON.K1.WF2.UP.PHASE
siesta.CON.K1.WF2.DOWN.PHASE
siesta.CON.K1.WF3.UP.REAL
siesta.CON.K1.WF3.DOWN.REAL
siesta.CON.K1.WF3.UP.IMAG
siesta.CON.K1.WF3.DOWN.IMAG
siesta.CON.K1.WF3.UP.MOD
siesta.CON.K1.WF3.DOWN.MOD
```

Warning: takes some time/check your disk space before the execution

(a)



```
Cube Files
siesta.RHO.UP.cube
siesta.RHO.DOWN.cube
siesta.DRHO.cube
siesta.RHO.UPminusDOWN.cube
siesta.RHO.cube
siesta.K1.WF1.UP.REAL.cube
siesta.K1.WF1.DOWN.REAL.cube
siesta.K1.WF1.UP.IMAG.cube
siesta.K1.WF1.DOWN.IMAG.cube
siesta.K1.WF1.UP.MOD.cube
siesta.K1.WF1.DOWN.MOD.cube
siesta.K1.WF1.UP.PHASE.cube
siesta.K1.WF1.DOWN.PHASE.cube
siesta.K1.WF2.UP.REAL.cube
siesta.K1.WF2.DOWN.REAL.cube
siesta.K1.WF2.UP.IMAG.cube
siesta.K1.WF2.DOWN.IMAG.cube
siesta.K1.WF2.UP.MOD.cube
siesta.K1.WF2.DOWN.MOD.cube
siesta.K1.WF2.UP.PHASE.cube
siesta.K1.WF2.DOWN.PHASE.cube
siesta.K1.WF3.UP.REAL.cube
siesta.K1.WF3.DOWN.REAL.cube
siesta.K1.WF3.UP.IMAG.cube
siesta.K1.WF3.DOWN.IMAG.cube
siesta.K1.WF3.UP.MOD.cube
```

Warning: takes some time/check your disk space before the execution

```
$ isosurface sign cutoff 0.045
E:\DJmolPlatform1\DJMolCompPhysComm_2020_BETAformat\DjmolPlatform1_for64\
```

```
$ isosurface sign cutoff 0.045
E:\DJmolPlatform1\DJMolCompPhysComm_2020_BETAformat\DjmolPlatform1_for64\Dj
$
```

(b)

Figure 3.6: (a)Shows list of 2D data files of water molecule from the Denchar utility, with two different *real* wavefunctions (the first two low lying orbitals in a contour diagram);(b) shows equivalent 3D data.

Similary, band diagram can also be generates with template scripts.

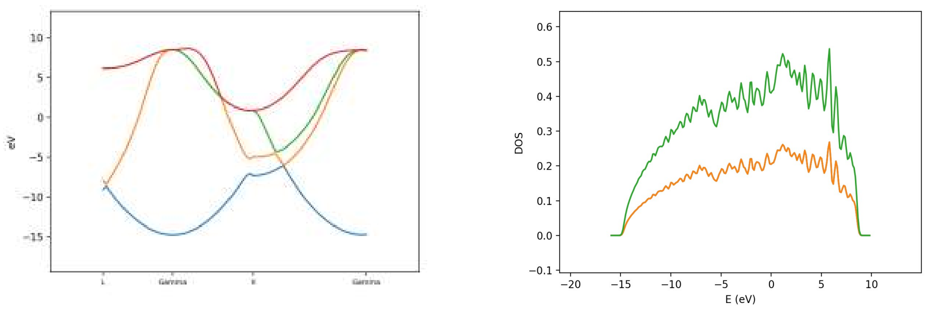
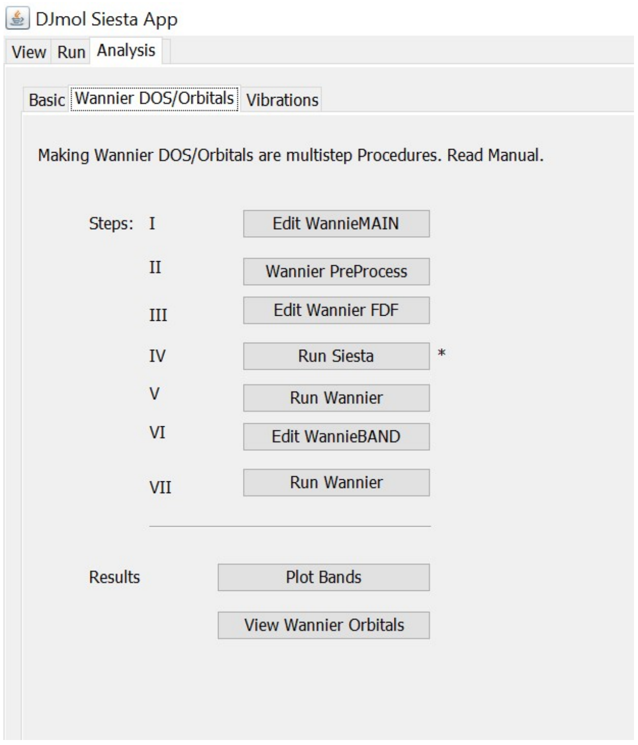


Figure 3.7: Band and DOS diagram of Aluminium (FCC). Note that in DOS plot, there are actually three lines (lower orange line is composed of two lines which corresponds to spin up and down contributions, whereas green line corresponds to the total DOS).

[4] Wannier Orbitals and Band Diagrams

To Wannier get localized molecular orbitals of crystalline or extended systems from the Bloch wavefunction, Wannier90.exe program is used in conjunction with the Siesta binaries. See *www.wannier.org* for more details.

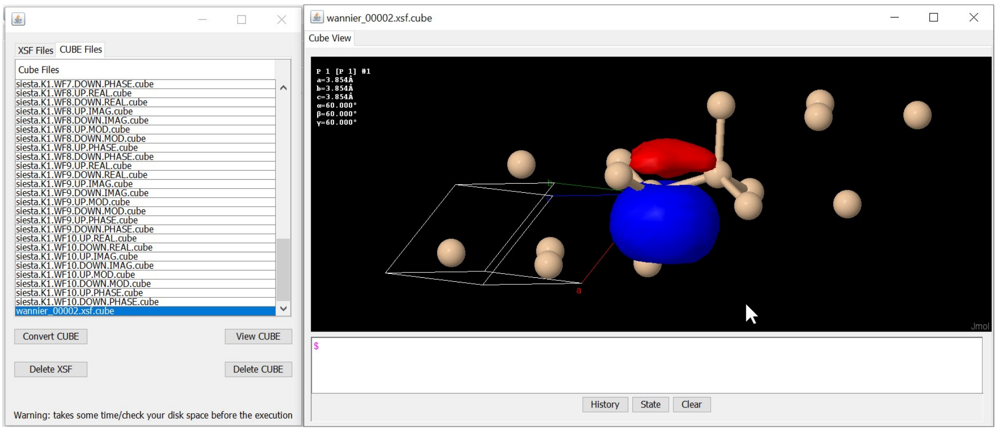
In the tab <Wannier DOS/Orbitals> this information can be obtained by following steps I to VII (as an example template, FCC bulk Si is used). Results can be viewed by using <Plot Bands> or by <View Wannier Orbitals>. This template files can be easily modified for other systems.



Making Wannier DOS/Orbitals are multistep Procedures. Read Manual.

Figure 3.8: Steps of Wannier data calculations.

<Plot Bands> give Wannier band diagram and <View Wannier Orbitals> list XSF files and that can be converted into the CUBE file format to display Wannier functions.



Cube Files

```
siesta.K1.WF7.DOWN.PHASE.cube
siesta.K1.WF8.UP.REAL.cube
siesta.K1.WF8.DOWN.REAL.cube
siesta.K1.WF8.UP.IMAG.cube
siesta.K1.WF8.DOWN.IMAG.cube
siesta.K1.WF8.UP.MOD.cube
siesta.K1.WF8.DOWN.MOD.cube
siesta.K1.WF8.UP.PHASE.cube
siesta.K1.WF8.DOWN.PHASE.cube
siesta.K1.WF9.UP.REAL.cube
siesta.K1.WF9.DOWN.REAL.cube
siesta.K1.WF9.UP.IMAG.cube
siesta.K1.WF9.DOWN.IMAG.cube
siesta.K1.WF9.UP.MOD.cube
siesta.K1.WF9.DOWN.MOD.cube
siesta.K1.WF9.UP.PHASE.cube
siesta.K1.WF9.DOWN.PHASE.cube
siesta.K1.WF10.UP.REAL.cube
siesta.K1.WF10.DOWN.REAL.cube
siesta.K1.WF10.UP.IMAG.cube
siesta.K1.WF10.DOWN.IMAG.cube
siesta.K1.WF10.UP.MOD.cube
siesta.K1.WF10.DOWN.MOD.cube
siesta.K1.WF10.UP.PHASE.cube
siesta.K1.WF10.DOWN.PHASE.cube
wannier_00002.xsf.cube
```

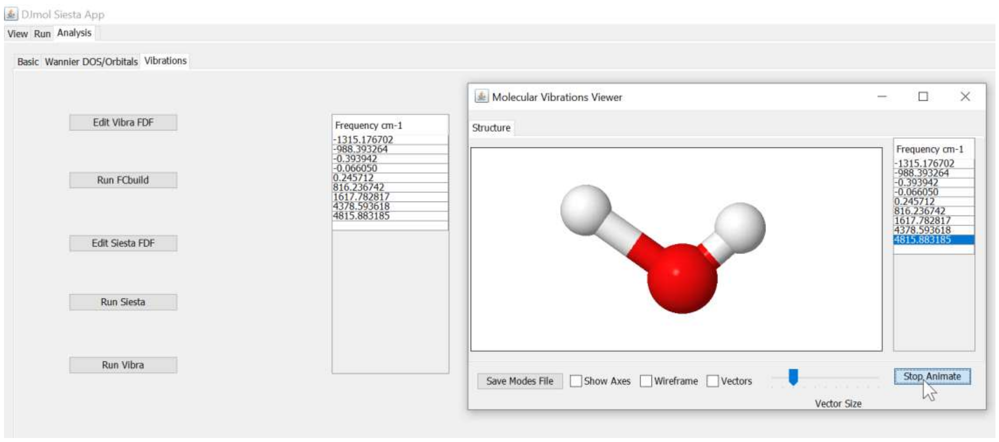
Warning: takes some time/check your disk space before the execution

```
P 1 [P 1] #1
a=3.854Å
b=3.854Å
c=3.854Å
α=60.000°
β=60.000°
γ=60.000°
```

Figure 3.9: A Wannier orbital of bulk Si generated from a XSF file.

[5] Molecular Vibrations

Molecular vibrations (using ⍰ point) can be easily calculated with the application. In the tab, <Vibrations>, execute the steps I-V, one after another (note that example script is made for water molecule, but it can be readily modified for other molecules.). The last command, <Run Vibra> calculates eigenvalues/vectors and it gives an another window in which different modes can be animated. The essential molecules vibrational data is saved in .\SiestaApp\SiestaBinary.4.0.CygWin64\SiestaVibModes.xyz.



```
Frequency cm-1
-1315.176702
-988.393264
-0.393942
-0.066050
0.245712
816.236742
1617.782817
4378.593618
4815.883185
```

Figure 3.10: A vibrational frequency calculator module and its mode-displayer.

4

OPENMD SIMULATIONS

OpenMD add-on is aimed to perform MD calculations with OpenMD binary/scripts. This tool has four parts, (1), Viewer (2), Editor (3), Terminal and, (4), Utilities.

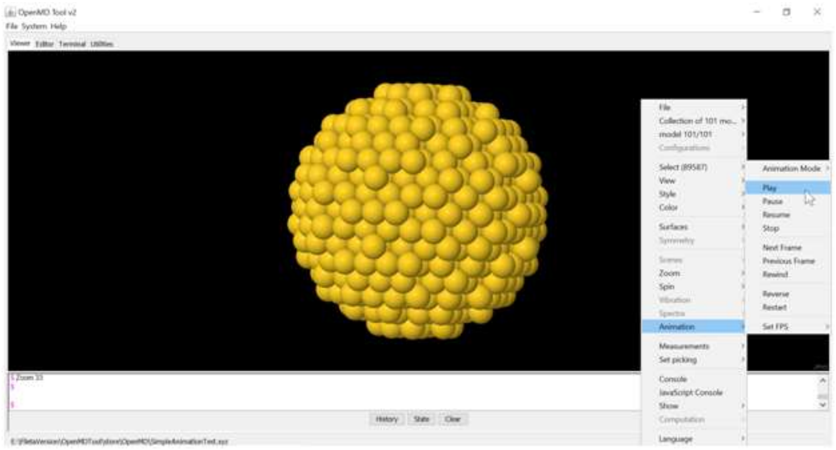


Figure 4.1: *Front end of the OpenMD add-on tool displays Ag nanoparticle obtained using OpenMD.*

In viewer a normal XYZ file can be viewed (usually it is obtained from OMD *dump* file by using a converter, *Dump2XYZ*). All other standard Jmol view facilities are available here.

Note that XYZ file will keep in the <OpenMD/Tempview> directory whereas the OMD file will be kept in the <./OpenMD> folder.

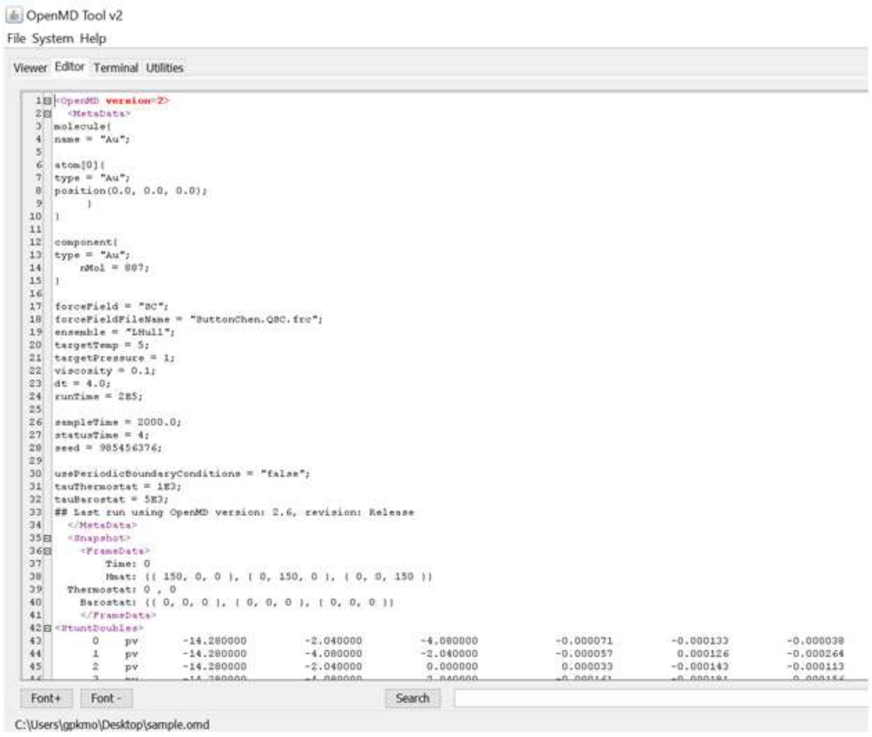
The Editor tool will open an OMD file. Since OMD file is a kind of XML file, a special editor is used to *fold* and *un-fold* many sections of this file as shown in the following file.

In Terminal-Tab, simple DOS commands can be executed. Note that one should add:

cmd.exe /c

before the Python scripts (eg. cmd.exe /c python.exe slabBuilder.py). The commands are stored in a stack-array and it can be easily called back (See Stored Commands).

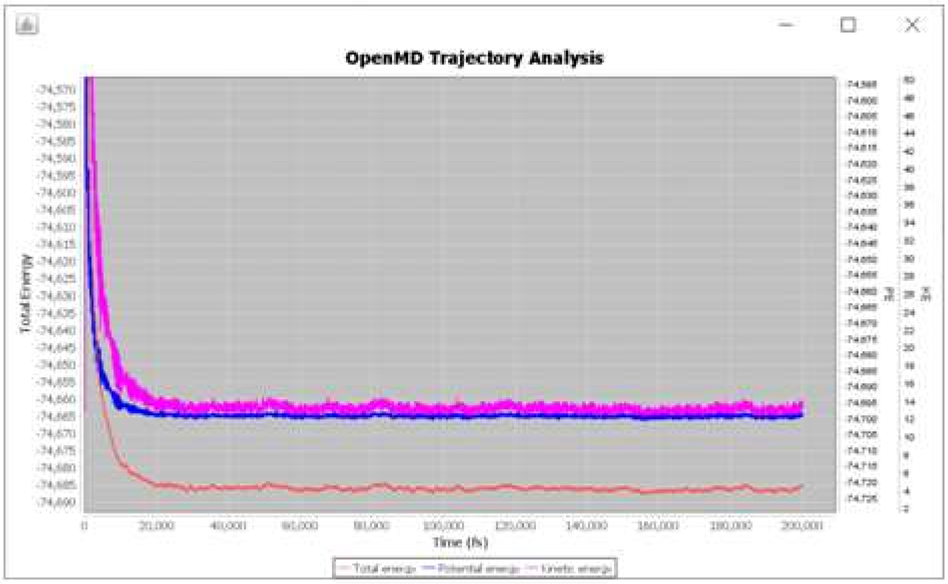
In the Utility-Tab several tools are added (eg. <Clean OpenMD> will delete the MD out files and <Open Trajectory> will open a separate viewer window to animate trajectories etc.). The <Plot Energies> will open <Stat> file and plot total, kinetic and potential energies. Similarly, <T P V> plot variation of temperature, pressure and volume during the MD run as it is stored in the <Stat> file.



```
<OpenMD version=2>
  <MetaData>
molecule{
name = "Au";

atom[0]{
type = "Au";
position(0.0, 0.0, 0.0);
      }
}

component{
type = "Au";
    nMol = 887;
}

forceField = "SC";
forceFieldFileName = "SuttonChen.QSC.frc";
ensemble = "LHull";
targetTemp = 5;
targetPressure = 1;
viscosity = 0.1;
dt = 4.0;
runTime = 2E5;

sampleTime = 2000.0;
statusTime = 4;
seed = 985456376;

usePeriodicBoundaryConditions = "false";
tauThermostat = 1E3;
tauBarostat = 5E3;
## Last run using OpenMD version: 2.6, revision: Release
  </MetaData>
  <Snapshot>
    <FrameData>
        Time: 0
        Hmat: {{ 150, 0, 0 }, { 0, 150, 0 }, { 0, 0, 150 }}
  Thermostat: 0 , 0
    Barostat: {{ 0, 0, 0 }, { 0, 0, 0 }, { 0, 0, 0 }}
    </FrameData>
<StuntDoubles>
      0     pv      -14.280000      -2.040000      -4.080000      -0.000071      -0.000133      -0.000038
      1     pv      -14.280000      -4.080000      -2.040000      -0.000057       0.000126      -0.000264
      2     pv      -14.280000      -2.040000       0.000000       0.000033      -0.000143      -0.000113
```

C:\Users\gpkmo\Desktop\sample.omd

Figure 4.2: OpenMD Tool’s XML styled editor for the OMD scripts.



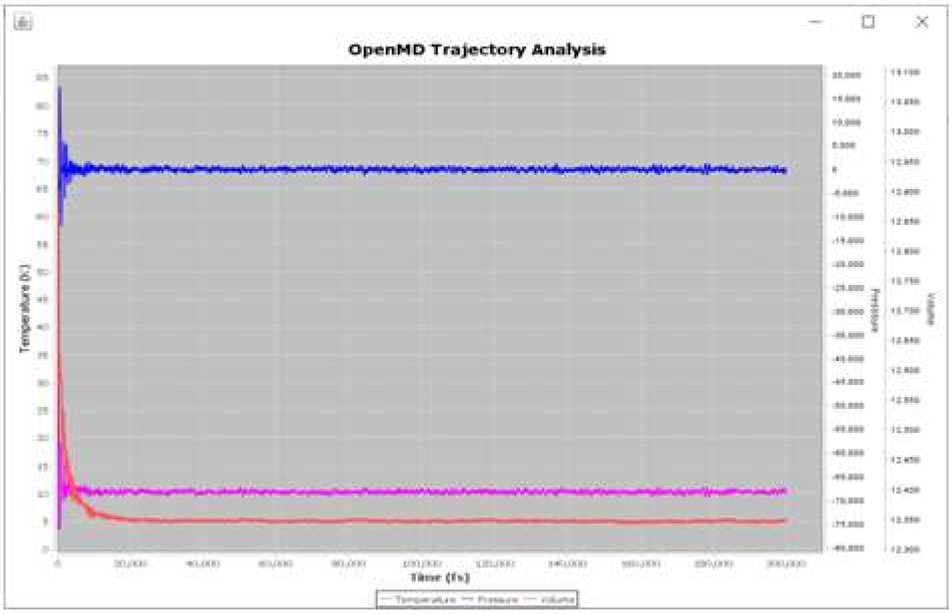


Figure 4.3: OpenMD Analyze variation of different energies during an MD simulation.

Since analyzing bond length gives an indication of reaction progress, <Calculate Bond Length> tool can be used to extract Bond length between two selected atoms ("Atom Number 1" and "Atom Number 2"). This will read XYZ file which *doesn't* use Periodic Boundary Conditions or PBC (as an example see, SampleAnimationTest.xyz file).

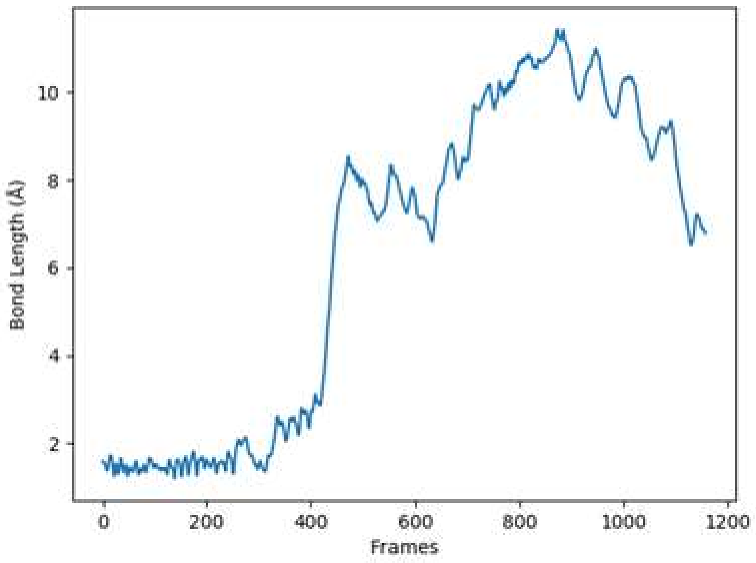


Figure 4.4: An out file from the MDAnalysis script. It describes that the bond is break near 400'th frame (in a periodic cubic box cell).

However, if PBC is applied more advanced script is needed to extract this bond length information. For example, if the system is equipped with Python and ***MDAnalysis*** package, a user can:

1. Edit <MDAnalysis> script (**MDanalysisv1.py**) if needed, eg. to change atom numbers

2. Use <Add MDAnalysis in Terminal>, which will add cmd**.exe /c python.exe MDanalysisv1.py** in the Terminal window

3. Go to ***Terminal*** and execute the above command.

The sample MDAnalysis script is:

```
import MDAnalysis as mda
from MDAnalysis.analysis.distances import dist
import matplotlib.pyplot as plt
import numpy as np

u = mda.Universe('mdPBCTrajectoryTest.xyz')

mybox=np.array([16., 16., 16., 90., 90., 90.], dtype='f')

# 1-2'th BL atoms in XYZ

distances = []

for ts in u.trajectory:
        distances.append(dist(mda.AtomGroup([u.atoms[0]]),mda.AtomGroup([u.atom
s[1]]),box=[16., 16., 16., 90, 90, 90])[2][0] )

plt.switch_backend('agg')
plt.plot(distances)
plt.xlabel('Frames')
plt.ylabel('Bond Length (Å)')
plt.show()
plt.savefig('1-2.png')
```

We strongly recommend to use MDAnalysis tools to post process the OpenMD data.

The details of MDAnalysis package is available at: *https://www.mdanalysis.org/*

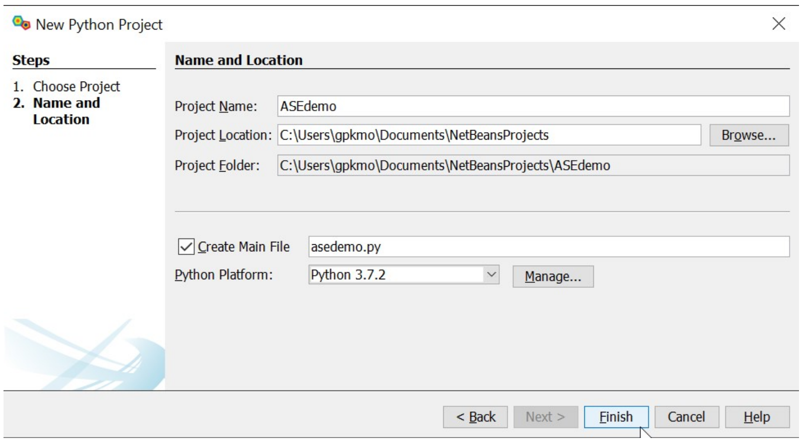
5

PYTHON SCRIPTING WITH ASE

To demonstrate ASE scripting within the platform, we must create a Python project:

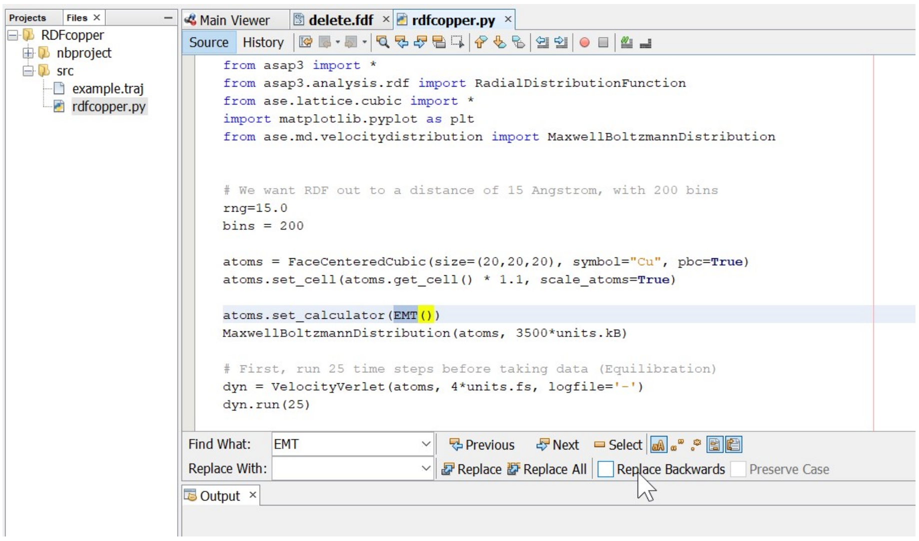
File ➢ New Project ➢ Python ➢ Python Project – Ant

and save this in a suitable directory. We name the main class as <ASEdemo>.



Project Name: ASEdemo
Project Location: C:\Users\gpkmo\Documents\NetBeansProjects
Project Folder: C:\Users\gpkmo\Documents\NetBeansProjects\ASEdemo
☑ Create Main File: asedemo.py
Python Platform: Python 3.7.2

Figure 5.1: Setting up a Python ASE project in the DJMol application.



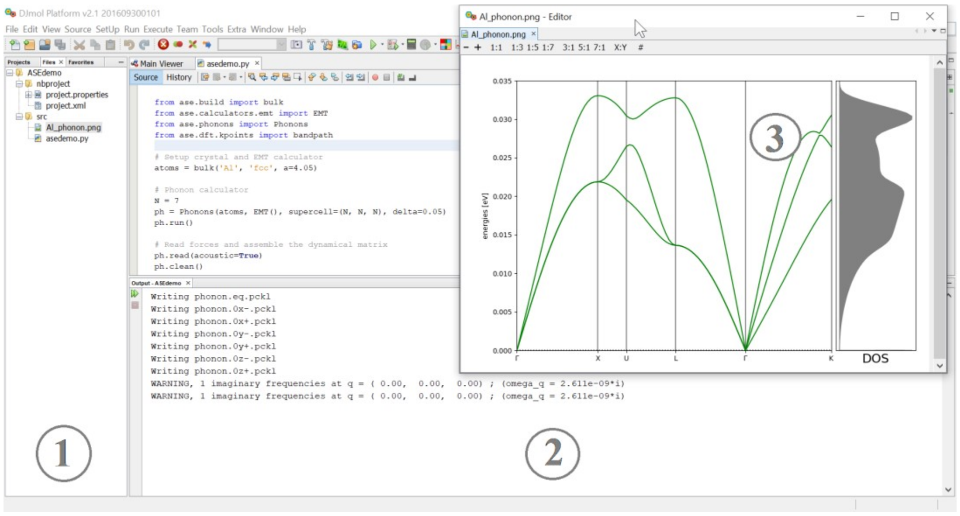
```
from asap3 import *
from asap3.analysis.rdf import RadialDistributionFunction
from ase.lattice.cubic import *
import matplotlib.pyplot as plt
from ase.md.velocitydistribution import MaxwellBoltzmannDistribution


# We want RDF out to a distance of 15 Angstrom, with 200 bins
rng=15.0
bins = 200

atoms = FaceCenteredCubic(size=(20,20,20), symbol="Cu", pbc=True)
atoms.set_cell(atoms.get_cell() * 1.1, scale_atoms=True)

atoms.set_calculator(EMT())
MaxwellBoltzmannDistribution(atoms, 3500*units.kB)

# First, run 25 time steps before taking data (Equilibration)
dyn = VelocityVerlet(atoms, 4*units.fs, logfile='-')
dyn.run(25)
```

Figure 5.2: A script window displays an ASE script.

The purpose of this ASE calculation is to obtain the *phonon dispersion for bulk aluminum* using a 7×7×7 supercell within effective medium theory (EMT as it is implemented in ASE). For the source, see, Ex2_phononbulkAl_DOS.py file from the example directory. After running it, this script will create the band and the DOS diagram of the system.

The script is provided in the Example directory. Once the script is ready, use <Run Project> command from the <Run> menu to execute the script. It will initiate ASE engine and produce the band and DOS diagrams. Optionally, users can develop their own python scripts in this project directory for other tasks (eg. to save the animation of a particular mode).

Run ➢ Run Project

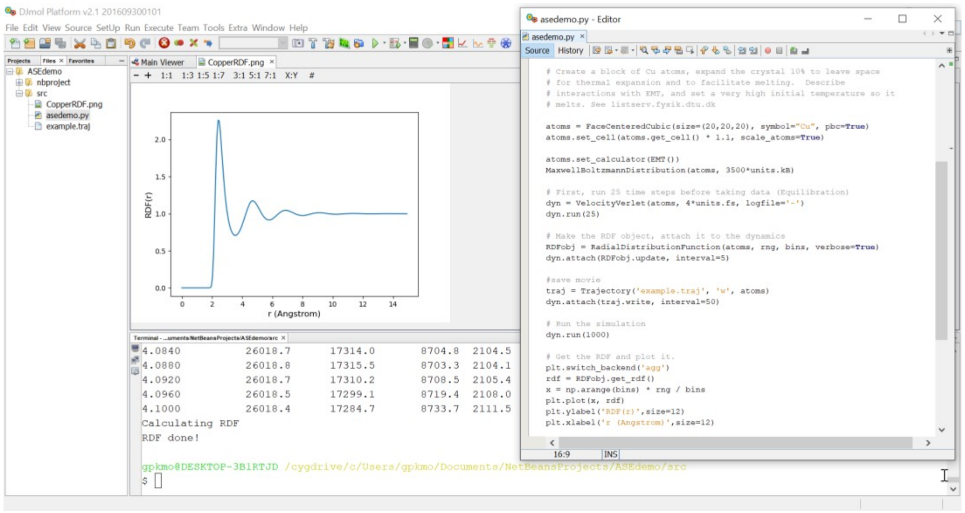


```
from ase.build import bulk
from ase.calculators.emt import EMT
from ase.phonons import Phonons
from ase.dft.kpoints import bandpath

# Setup crystal and EMT calculator
atoms = bulk('Al', 'fcc', a=4.05)

# Phonon calculator
N = 7
ph = Phonons(atoms, EMT(), supercell=(N, N, N), delta=0.05)
ph.run()

# Read forces and assemble the dynamical matrix
ph.read(acoustic=True)
ph.clean()
```

```
Writing phonon.eq.pckl
Writing phonon.0x-.pckl
Writing phonon.0x+.pckl
Writing phonon.0y-.pckl
Writing phonon.0y+.pckl
Writing phonon.0z-.pckl
Writing phonon.0z+.pckl
WARNING, 1 imaginary frequencies at q = ( 0.00,  0.00,  0.00) ; (omega_q = 2.611e-09*i)
WARNING, 1 imaginary frequencies at q = ( 0.00,  0.00,  0.00) ; (omega_q = 2.611e-09*i)
```

Figure 5.3: (1) is the *Workplace* and (2) is the console output from the Python run and (3) is the resultant Band and DOS figures that is created in the *Workplace*.

In the next advanced example (see example directory of Python) we will show that RDF (radial distribution function) of melting copper in a cubic box. Here, EMT is also used but RDF function is taking from ASAP calculator. ASAP is a calculator for doing large-scale classical molecular dynamics within the ASE package (its libraries can be compiled in CygWin or WSL). Below figure shows the RDF from ASAP calculator and uses systems terminal.



```
# Create a block of Cu atoms, expand the crystal 10% to leave space
# for thermal expansion and to facilitate melting.  Describe
# interactions with EMT, and set a very high initial temperature so it
# melts. See listserv.fysik.dtu.dk

atoms = FaceCenteredCubic(size=(20,20,20), symbol="Cu", pbc=True)
atoms.set_cell(atoms.get_cell() * 1.1, scale_atoms=True)

atoms.set_calculator(EMT())
MaxwellBoltzmannDistribution(atoms, 3500*units.kB)

# First, run 25 time steps before taking data (Equilibration)
dyn = VelocityVerlet(atoms, 4*units.fs, logfile='-')
dyn.run(25)

# Make the RDF object, attach it to the dynamics
RDFobj = RadialDistributionFunction(atoms, rng, bins, verbose=True)
dyn.attach(RDFobj.update, interval=5)

#save movie
traj = Trajectory('example.traj', 'w', atoms)
dyn.attach(traj.write, interval=50)

# Run the simulation
dyn.run(1000)

# Get the RDF and plot it.
plt.switch_backend('agg')
rdf = RDFobj.get_rdf()
x = np.arange(bins) * rng / bins
plt.plot(x, rdf)
plt.ylabel('RDF(r)',size=12)
plt.xlabel('r (Angstrom)',size=12)
```

```
4.0840         26018.7      17314.0       8704.8  2104.5
4.0880         26018.8      17315.5       8703.3  2104.1
4.0920         26018.7      17310.2       8708.5  2105.4
4.0960         26018.5      17299.1       8719.4  2108.0
4.1000         26018.4      17284.7       8733.7  2111.5
Calculating RDF
RDF done!

gpkmo@DESKTOP-3B1RTJD /cygdrive/c/Users/gpkmo/Documents/NetBeansProjects/ASEdemo/src
$ 
```

Figure 5.4: Radial distribution function of melting copper (FCC) at 2500K from an ASAP calculator.

In first add Terminal into the system by:

Window ➢ IDE Tools ➢ Terminal

This will get CygWin ot WSL, depends upon the settings. After this, type the code in the terminal:

```
python3.6m asedemo.py
```

It will launch an ASAP calculation, and calculates RDF of the melting copper atoms. Note that its RDF is very similar to that of water and it confirms that *liquid state* of Cu atoms at high temperature is well represented by the ASAP model.

6

TOOLS OF DJMOL

Tools are generally calculator independent (add-on) modules which can assist some steps in the modeling.

Some of the important Tools are described below.

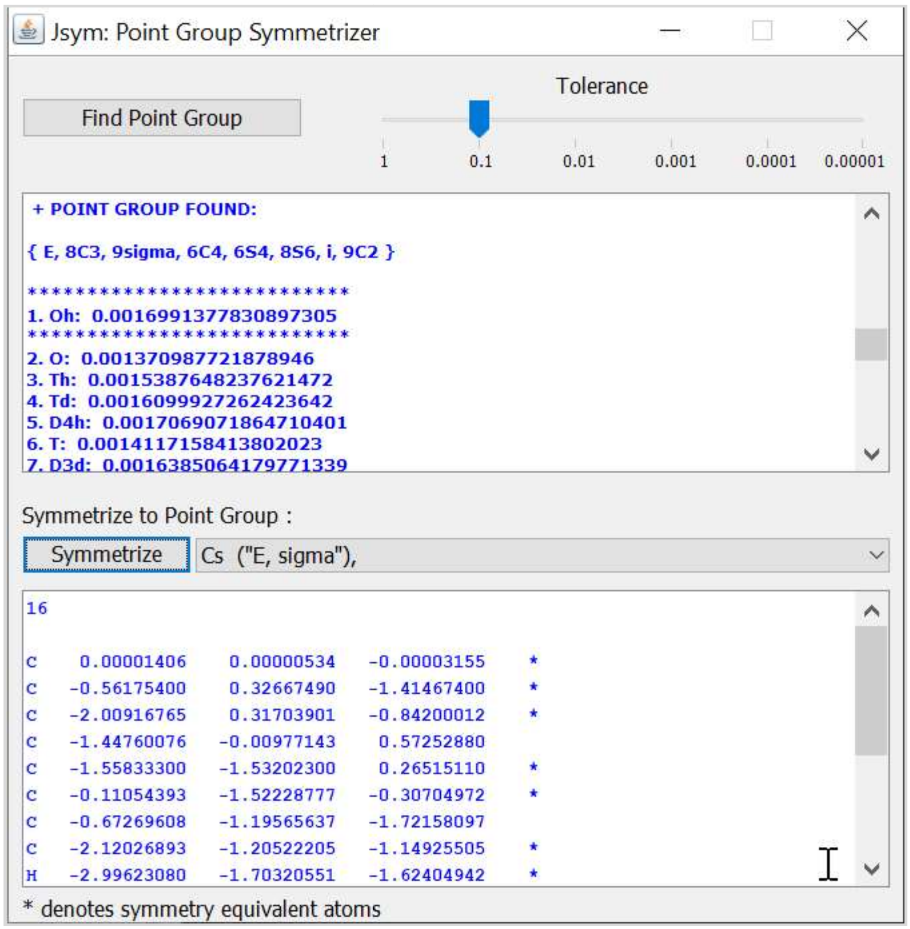
http://

[1] Analysing Point Group

DJMol use a *point group analyzer* and from this one can easily find the point group of a molecule and its subgroups. It can also tell the next possible point group within a specified error-bar. This module can be invoked by:

Tools ➢ PointGroup Analyzer

This tool can also be used to deform the geometry of molecules so that one can modify a molecule’s PG into another one (in other words, we can constrain a PG into its sub group).



```
+ POINT GROUP FOUND:

{ E, 8C3, 9sigma, 6C4, 6S4, 8S6, i, 9C2 }

***************************
1. Oh:  0.0016991377830897305
***************************
2. O:  0.001370987721878946
3. Th:  0.0015387648237621472
4. Td:  0.0016099927262423642
5. D4h:  0.0017069071864710401
6. T:  0.0014117158413802023
7. D3d:  0.0016385064179771339
```

```
16

C     0.00001406    0.00000534   -0.00003155    *
C    -0.56175400    0.32667490   -1.41467400    *
C    -2.00916765    0.31703901   -0.84200012    *
C    -1.44760076   -0.00977143    0.57252880
C    -1.55833300   -1.53202300    0.26515110    *
C    -0.11054393   -1.52228777   -0.30704972    *
C    -0.67269608   -1.19565637   -1.72158097
C    -2.12026893   -1.20522205   -1.14925505    *
H    -2.99623080   -1.70320551   -1.62404942    *
```

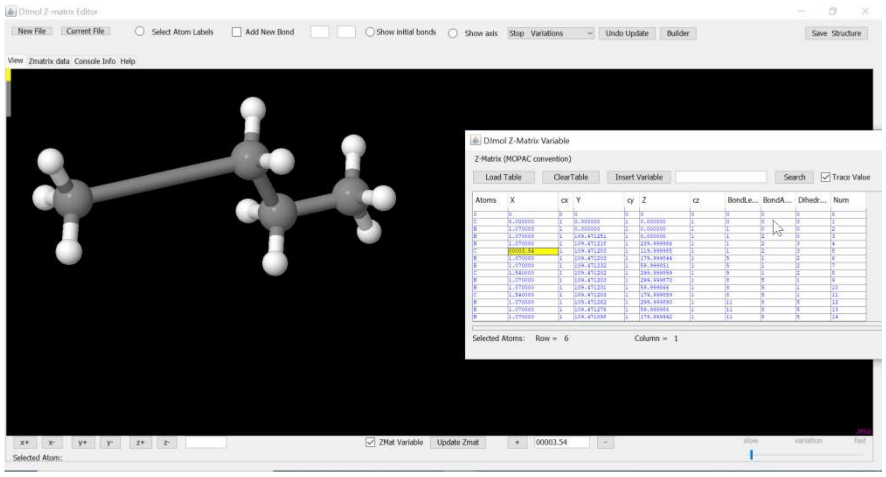
* denotes symmetry equivalent atoms

[2] Z-Matrix Editor

Z-Matrix or Z-mat tool can be called by:

Extra ➢ Z-Matrix Tool

This Z-Mat editor is made for the XYZ files; It can be either used as an independent program (use File button to load a required XYZ file) or as an auxiliary program to manipulate the loaded XYZ file. Use either <New File> button to vary Z-matrix a of a new Cartesian file or <Current File> button to fetch the currently loaded geometry from the main panel of DJmol.



Atoms	X	cx	Y	cy	Z	cz	BondLe...	BondA...	Dihedr...	Num
0	0	0	0	0	0	0	0	0	0	0
C	0.000000	1	0.000000	1	0.000000	1	0	0	0	1
H	1.070000	1	0.000000	1	0.000000	1	1	0	0	2
H	1.070000	1	109.471251	1	0.000000	1	1	2	0	3
H	1.070000	1	109.471218	1	239.999986	1	1	2	3	4
C	00003.54	1	109.471203	1	119.999985	1	1	2	3	5
H	1.070000	1	109.471203	1	179.999844	1	5	1	2	6
H	1.070000	1	109.471232	1	59.999851	1	5	1	2	7
C	1.540000	1	109.471202	1	299.999859	1	5	1	2	8
H	1.070000	1	109.471203	1	299.999873	1	8	5	1	9
H	1.070000	1	109.471281	1	59.999866	1	8	5	1	10
C	1.540000	1	109.471203	1	179.999859	1	8	5	1	11
H	1.070000	1	109.471262	1	299.999890	1	11	8	5	12
H	1.070000	1	109.471276	1	59.999988	1	11	8	5	13
H	1.070000	1	109.471098	1	179.999942	1	11	8	5	14

Selected Atoms: Row = 6 Column = 1

Figure 6.1: A butane molecule (exaggerated) and its Z-matrix data.

To change Z-Mat Data:

After displaying the geometry, select <Select a Z-mat variable> radio button to show the corresponding Z-matrix data (MOPAC style, generated from the OpenBabel application) of the displayed molecule in this application.

Also select any single cell from either <X>, or <Y> or <Z> column, which represents bond-length, bond angles, or dihedral angles, respectively. After that select <Insert Variable> button to make the cell element as a Z-mat variable.

And then use the <Save> button and close the window. After that, immediately select the <Start Zmatrix Variation> menu from the top of the window, and by using [+] and [-] buttons, start the variation.

One can either save the data (<Update Zmat> button) or discard the variation and to move the molecule into its original state by using <Undo Update> button.

It is also important to note that to stop and undo the current variation,first to select <Stop Variations> and press <Undo Update> button>.

If you want to move only X or Y or Z positions of a single atom use the menu (<Move selected Atom>), and select an atom (use a click on the desired atom) , and press any[x+][x-]...buttons near the bottom left corner.

Important: *To save the updated Geometry press <Update Zmat>* ***AND then*** *<Save Structure> button.*

Or to discard the variation and to move the molecule into its original state by using <Undo Update> button.

Also don't forget to use <Save Structure> button to save the modified Cartesian file. It will be saved as **CurrentFinalXYZ**.**xyz** file in the folder <**ZMATfolder**> in the program directory.

If necessary use:

[1] <Select Atoms Labels> to see individual atom labels, which may be useful when one manipulates Z-Mat.

[2] <Show initial bonds> for keeping the original bonding pattern, irrespective of the current geometric state. This can be useful when some bonds are too far away from the initial bonded state.

[3] <Show axis> to show cartesian axis

[4] <Add New Bond>[n1] [n2] - to show a new bond between the atomlabel, n1 to atomlabel2, n2 atoms.

[5] <Builder> option can be used if necessary.

Note that a user can also enter numerical values (in float) to the Text Fields. Also Move the slide bar to maximum when you vary the Angles/Dihedral angles and to a minimum if you want to manipulate the distance.

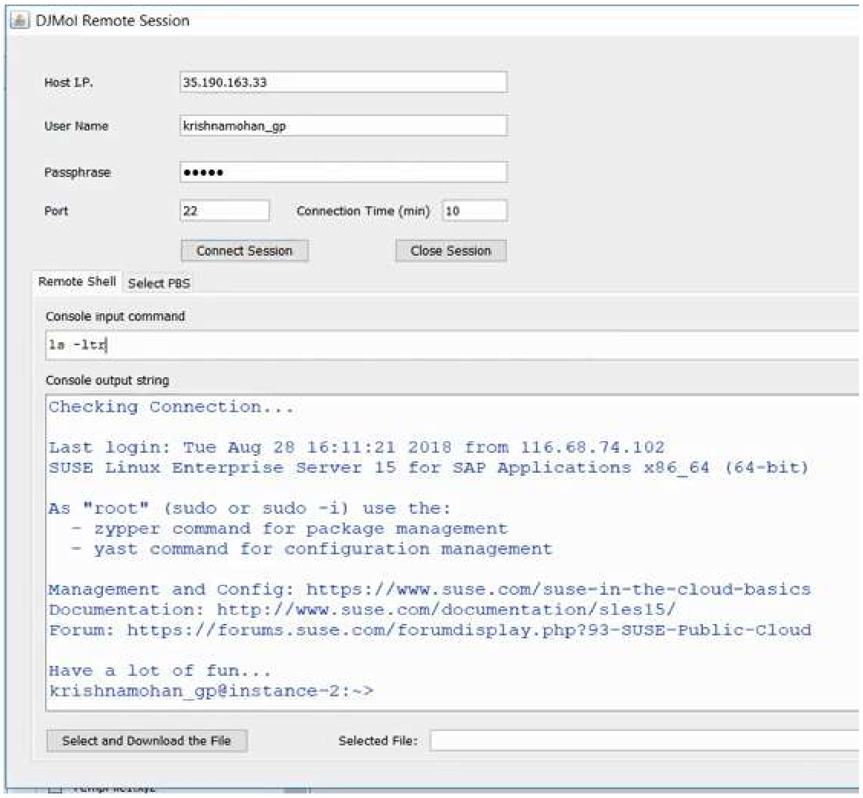
[3] Remote Submission of Scripts

For time consuming job, such as MD calculations, it is a better idea to use a dedicated machine like Linux cluster/Google Cloud etc. instead of a PC. In this case, one need to generate SSH keys (ie. one private and one public key), preferably, by using PUTTYGEN application; see https://www.puttygen.com/ to download this mandate application.

Note that we need <RSA> based private key (which is stored in your PC) and a public key (stored in the Remote machine). Note that you must enter a <Passphrase> for the <Private key> and its format should be in the **OpenSSH** (ie. not in the PPK format, which is the PuttyGens' default). PuttyGEN can convert this private key in PPK format to OpenSSH format. Finally the Private Key should be kept in the <ssh> folder with a file name, <id_rsa>. Make sure that the file <known_hosts> is existed in <ssh> format. After this one can invoke the <Remote Submission Tool> as:

RunProject ➢ Remote Submission

This tool can be used instead of PUTTY, and it is equipped with tools for file upload/download purposes. The downloaded files are kept in <SCP> folder, by default.



```
ls -ltr
```

```
Checking Connection...

Last login: Tue Aug 28 16:11:21 2018 from 116.68.74.102
SUSE Linux Enterprise Server 15 for SAP Applications x86_64 (64-bit)

As "root" (sudo or sudo -i) use the:
  - zypper command for package management
  - yast command for configuration management

Management and Config: https://www.suse.com/suse-in-the-cloud-basics
Documentation: http://www.suse.com/documentation/sles15/
Forum: https://forums.suse.com/forumdisplay.php?93-SUSE-Public-Cloud

Have a lot of fun...
krishnamohan_gp@instance-2:~>
```

Figure 6.2: A sample SSH session (for Google Cloud Platform remote connection and file transactions).

[4] Version Control using GIT

Version control (VC) systems are a category of software tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members. **Git** is an example of VCS.

Why the VC is used in the DJMol? Since the text based input scripts are playing an important role in the modeling (inputs of Siesta/DFTB/ASE, all are text files). Using VC, these data can be re-edited remotely and at the same time it keeps different version of these files in a systematic way (for example, later, it can be used by collaborators/public).

Some Basic Definitions to be familiarized are:

1. **FETCH:** The git fetch command downloads commits, files, and refs from a remote repository into your local repo. Fetching is what you do when you want to see what everybody else has been working on.

2. **PULL:** The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content.

3. **PUSH:** The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo. It is the counterparts to git fetch.

How to use Git to push and pull in DJMol?

1. Initializing Version Control.

    a. Set up a Git Remote Repository in GitHub and copy the Repository URL.

    b. Open the project in which you need to use version Control.

    c. Initializing the Local Repository.

        Team ➢ Git ➢ Initialize ➢ (OK)

    d. Linking the Local Repository and the online Repository

        Team ➢ Remote ➢ Pull ➢ Fill ➢ Check Master ➢ Finish

        Specify ➢ PateTheRepository ➢ UserName/PassWord (if needed) ➢ Next

2. Push

a. After making necessary changes in the Project.

b. From the Project sidebar, add the needed files to the repository by right-clicking on the file and Clicking on **Add**.

c. Commit the files to the Repository

Team ➢ Commit ➢ AddACommitMessage ➢ Commit

d. Push to remote repository.

Team ➢ Remote ➢ PushToUpstream ➢ Yes

3. Pull

a. Fetch

Team ➢ Remote ➢ Fetch ➢ Next ➢ Finish

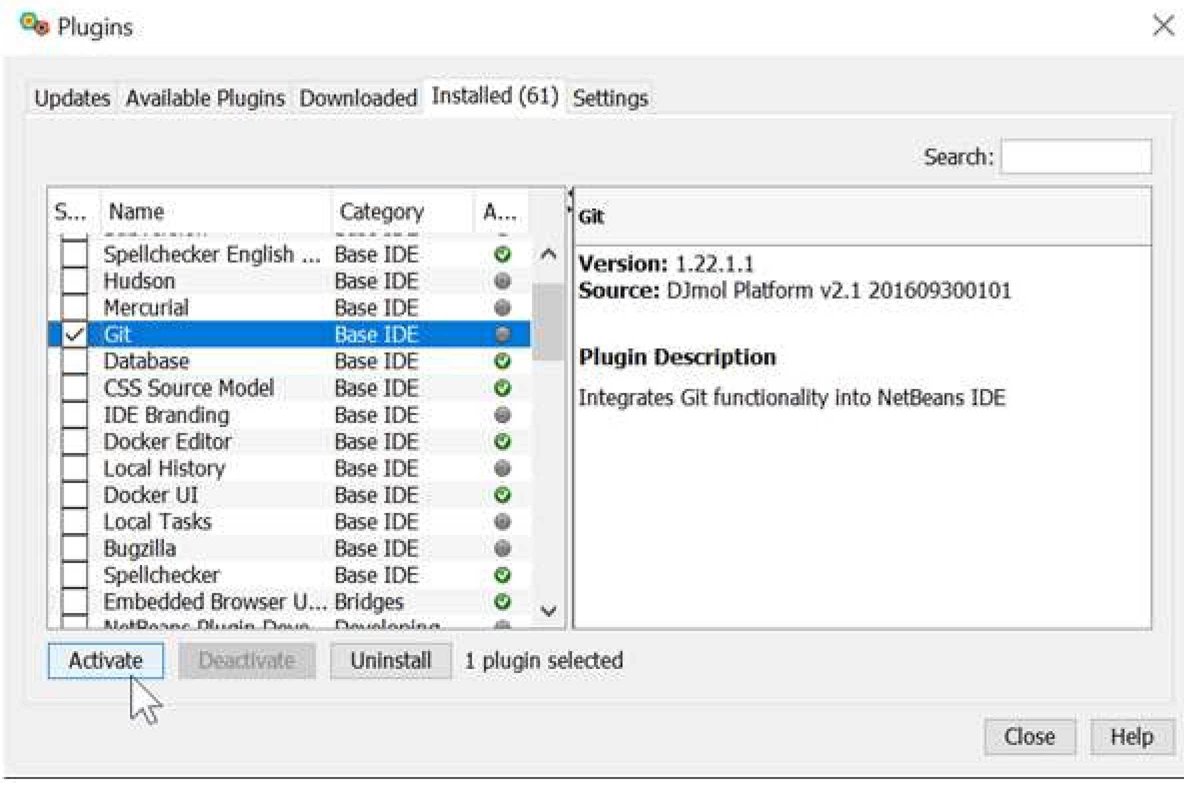
b. Pull

Team ➢ Remote ➢ Pull ➢ Next ➢ CheckMaster ➢ Finish

Installation of Git

Adding Github Plugins

After installing the application, search <Git> in the plug-ins search field. Select the <Git> from the <Installed Packages> and click on <Activate> as shown below figure.



Plugins

Updates | Available Plugins | Downloaded | Installed (61) | Settings

Search:

S...	Name	Category	A...
☐	Spellchecker English ...	Base IDE	✓
☐	Hudson	Base IDE	●
☐	Mercurial	Base IDE	●
☑	Git	Base IDE	●
☐	Database	Base IDE	✓
☐	CSS Source Model	Base IDE	✓
☐	IDE Branding	Base IDE	●
☐	Docker Editor	Base IDE	✓
☐	Local History	Base IDE	●
☐	Docker UI	Base IDE	✓
☐	Local Tasks	Base IDE	●
☐	Bugzilla	Base IDE	●
☐	Spellchecker	Base IDE	✓
☐	Embedded Browser U...	Bridges	✓

Git

Version: 1.22.1.1
Source: DJmol Platform v2.1 201609300101

Plugin Description

Integrates Git functionality into NetBeans IDE

Activate | Deactivate | Uninstall | 1 plugin selected

Close | Help

Figure 6.3: Installation of Github plugin in DJMol.

[5] Database File Retrieving

To fetch different structural file form various open file repositories, a Database tool is implemented. It is essentially uses Jmol's DB module.

It can be launched by:

Extra ➤ Database Tool

Open URL: It can be used to open XYZ or PDB file or any other Jmol recognizable file from the web server, say from Github.

Open MOL: It is mainly for, SMILES, InChI, or CAS from either a PubChem database or from NCI/NIH database.

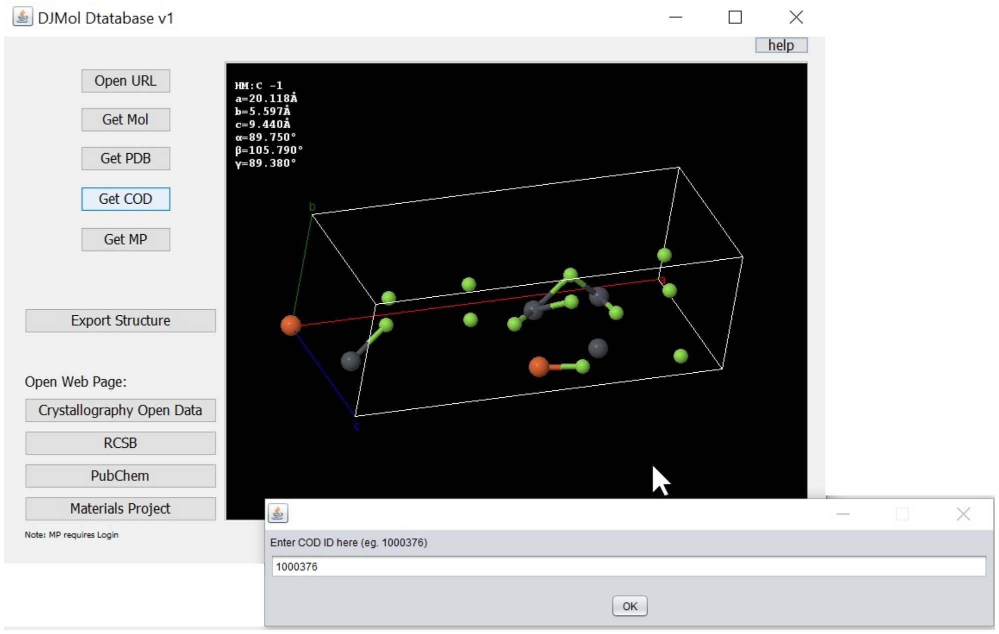
Open PDB: It use RCSB web (please prefer RCSB) to load 4-character PDB ID (eg. 1crn) or 3 -character ligand (eg. 60C).

Open COD: It opens a specific COD ID from *http://www.crystallography.net/cod/*

Open Materials Project: It opens a specific Materials Project ID number:

All opened file is saved in ./Database folder from

Export To: Only MOL, XYZ and PDB formats are supported for export its images.



```
HM:C -1
a=20.118Å
b=5.597Å
c=9.440Å
α=89.750°
β=105.790°
γ=89.380°
```

Enter COD ID here (eg. 1000376)

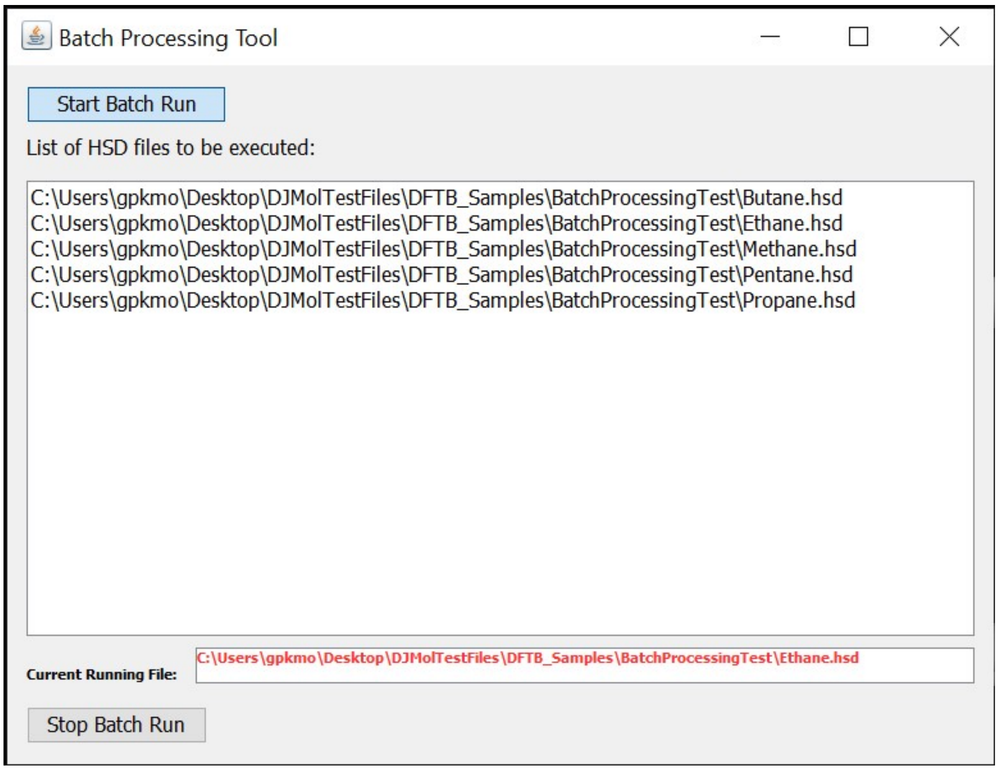
1000376

Figure 6.4: Showing a Database window with a retrieved COD file from COD databse.

[6] MISCELLANEOUS TOOLS

Some of the miscellaneous tools are shown below:

Start Batch Process: For DFTB+ calculation, a number of HSD files in a particular folder can be called one after another (known as *batch processing*). This will be useful for building 1D PES or to analyze energies of particular set of molecules. The resultant TGZ files contain all the out files including the submitted HSD file, and it will be transformed to the folder at the end of the each calculation.



List of HSD files to be executed:

```
C:\Users\gpkmo\Desktop\DJMolTestFiles\DFTB_Samples\BatchProcessingTest\Butane.hsd
C:\Users\gpkmo\Desktop\DJMolTestFiles\DFTB_Samples\BatchProcessingTest\Ethane.hsd
C:\Users\gpkmo\Desktop\DJMolTestFiles\DFTB_Samples\BatchProcessingTest\Methane.hsd
C:\Users\gpkmo\Desktop\DJMolTestFiles\DFTB_Samples\BatchProcessingTest\Pentane.hsd
C:\Users\gpkmo\Desktop\DJMolTestFiles\DFTB_Samples\BatchProcessingTest\Propane.hsd
```

Current Running File: C:\Users\gpkmo\Desktop\DJMolTestFiles\DFTB_Samples\BatchProcessingTest\Ethane.hsd

Calculator: System's default calculator can be invoked.

GIF Utility: GIF animation files can be viewed or generated using this utility (eg. phonon modes). Time between two frames can be adjusted and the resultant file is stored in **./Scratch_images.**

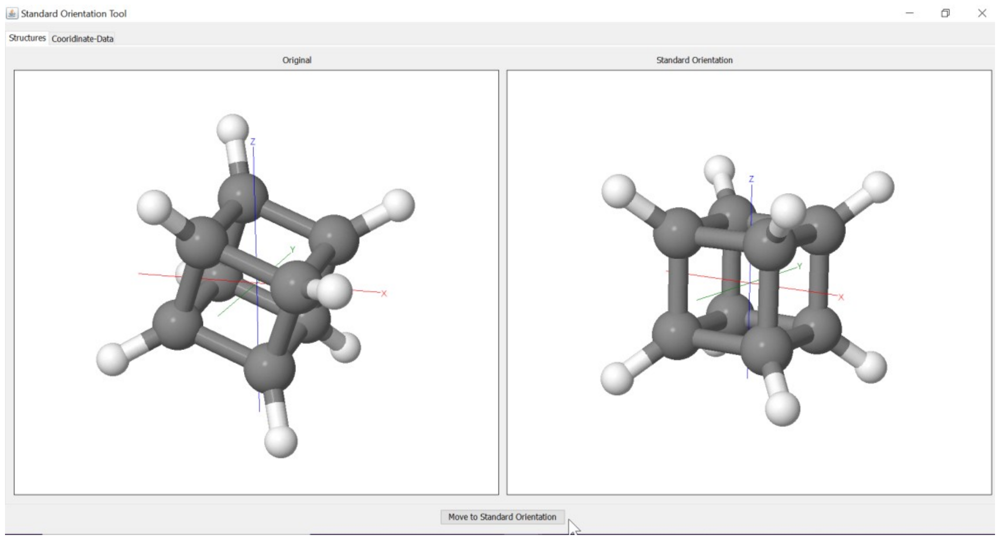
Process Status: It is used to monitor current resources of the computer system (used/availble RAM, HDD space etc.). This can be used before starting a calculation. For example, if the PC memory is low, allocate more RAM by stopping other less important process by using Window's **Task Manager** utility.

Unit Conversion: A basic unit converter for length, time, energy etc. it also includes atomic unit.

Matrix Viewer: 2D matrix data viewer of the program. It is useful to analyze the overall shape and symmetry of the matrix data, such as Hessian or Hamiltonian. See example directory for its sample file.

Convert Structure: *Open Babel* is used to convert from one structure data into another data. See Figure 2.10.

Standard Orientation: Using symmetry a disoriented molecule can be oriented with respect to a symmetry axis. This tool is useful for making a more systematic Z-matrix or Cartesian file*. It is strongly recommended that the standard oriented geometry data should be used in the Z-matrix tool to re-adjust its coordinates.*



Open DJMol Directory: It will open the parent directory.

Send Message: User can mail the forum using this utility. Please academic E-mail ID (avoid .com E-mail IDs – it won't support). If needed images should be linked as an URL link (say, by drive.google.com/… link).

7

ARCHITECTURE OF THE PROGRAM

Software Architecture

In practice, computational materials science research encompasses three distinctive steps, viz. (1), constructing the geometrical data or structure of the system (2), performing computations and, (3), obtaining results by the post processing of calculated data. All these steps can be effectively coupled by means of modeling platform such that a user can create, visualize, and share various input data files and run it with appropriate programs to obtain output data to analyze the results.

By integrating molecular (or crystal) visualizer, scripting tools (in this case, Python and Jmol scripting) and other standard features of an IDE (integrated development environment), a user can interactively build or manipulate structures of materials or molecules, and its atomistic properties can be calculated with appropriate ab initio programs from either a local or a remote machine. A number of built-in tools, scripts are provided for the analysis purpose. To demonstrate the platform we have used DFTB+ as well as Siesta electronic structure code along with ASE (atomic simulation environment) and OpenMD molecular dynamics package.

The programming language, Java (version 8) was chosen for this project since it is free, strongly object-oriented and shows its neutrality to the various operating systems. The object oriented programming is arguably the most popular software development technique which effectively manages the complexity in code development [9]. We preferred Java, since (1) it does not use pointers, and (2), it supports threads implicitly. Although pointers provide direct access to the computer memory, careless use of the pointers will easily leads to segmentation fault and other vulnerables. It is our personal opinion that the bugs emerged from the improper application of pointers are difficult to trace or debug, in addition to this, code with pointers is difficult to translate into other computer languages without the pointer features. Concerning threads, in this application, we used a number of threads (i.e. user threads), apart from the JVM generated daemon threads. Since the threads are inherently supported by the language one can readily create a thread by extending a Thread class. Exception handling of Java is equipped with two types, namely, unchecked and checked exception handling. And in most of the cases we used checked exceptions, as it is less tedious to implement.

The core of the DJMol application consists of (Apache-) Netbeans Platform and it can be regarded as the engine behind the Netbeans IDE. In other words, many of the technical features of DJMol program is *inherited* from Netbeans IDE which is initially designed for developing Java/Javascript and C/C++ applications and consists a number of utilities to increase the productivity of a user (for example, it has an advanced sourcecode editor with code completion utilities, tools for refactoring, version control systems, Git based collaboration tools etc.) and these utilities can also be used efficiently for various modeling or scripting tasks. Moreover, this platform consists of a set of independent modular software components called modules (e.g., an SSH module to communicate with an external cloud platform). Apart from this, the program supports plug-ins so that one can selectively add or remove new features into it (e.g. Python interpreter) without being re-compiled the code. Optionally, users of the code can make their own plug-ins, for example, to incorporate another *ab initio* package. See the below for the schematic structure of the program. To the best of our knowledge, DJMol is the first opensource modeling platform which is built from a programming IDE.

To display the molecular, crystal, nanostructures a Java library, Jmol, has been embedded into the program. The Jmol program is an open source, cross-platform and a highly independent (i.e. it does not depend on any third party libraries like Java3D or OpenGL) 3D visualization application. Apart from this, there are two distinct features associated with Jmol: (1) Jmol can be programmatically embedded into any Java code which uses Swing application programming interface (2), It supports an internal command scripting, so that a user can control or manipulate the display or send and retrieve parameters, data, commands etc. For example, to manipulate Gaussian cube files a user can effectively apply this internal scripting ability of Jmol. Unlike most of the visualizers, Jmol is capable of perceiving the molecular structure in three dimensions by applying stereographic projections. A viewer, with appropriate anaglyph spectacles, 3D perspective images of the molecules can be interactively visualized. Apart from this, Jmol consists of an UFF (uniform force field) method and it can be used to pre-optimize a variety of organic as well as inorganic molecules. To create 2D data plots we have used either Matplotlib based scripts or Jfreechart libraries. In the case of 3D data plots, (for example, the potential energy surfaces or orbital contour diagrams), a Java based opensource library *jzy3d* has been used.

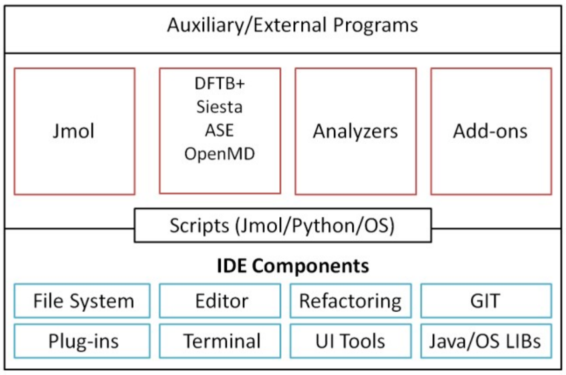


Figure 7.1: A schematic diagram of DJMol software architecture. The core of the program is an IDE which is connected to other modules or programs.

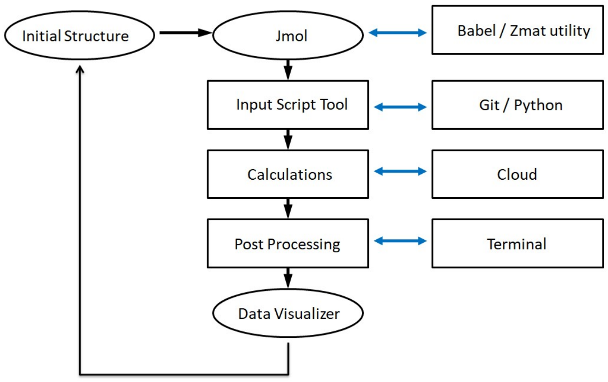
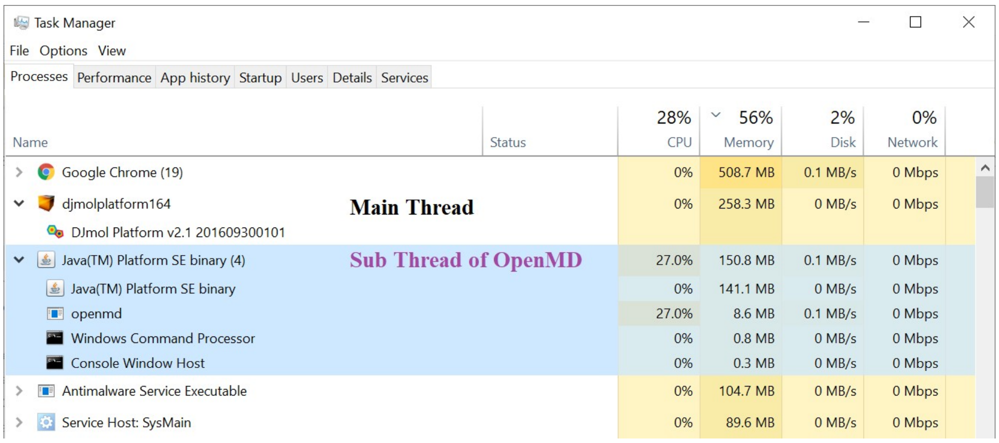


Figure 7.2: A standard workflow for a modeling task under the DJMol program (mandate workflows are indicated by black arrows).

Java's native thread is used to initiate add-ons so that its console can be controlled independently, if needed. Below figure shows Windos' **Task managers** snapshot of OpenMD add-on, while it is under running (and it indicate two independent processes).


Task Manager

File Options View

Processes | Performance | App history | Startup | Users | Details | Services

Name	Status	28% CPU	56% Memory	2% Disk	0% Network
Google Chrome (19)		0%	508.7 MB	0.1 MB/s	0 Mbps
djmolplatform164	**Main Thread**	0%	258.3 MB	0 MB/s	0 Mbps
DJmol Platform v2.1 201609300101					
Java(TM) Platform SE binary (4)	**Sub Thread of OpenMD**	27.0%	150.8 MB	0.1 MB/s	0 Mbps
Java(TM) Platform SE binary		0%	141.1 MB	0 MB/s	0 Mbps
openmd		27.0%	8.6 MB	0.1 MB/s	0 Mbps
Windows Command Processor		0%	0.8 MB	0 MB/s	0 Mbps
Console Window Host		0%	0.3 MB	0 MB/s	0 Mbps
Antimalware Service Executable		0%	104.7 MB	0 MB/s	0 Mbps
Service Host: SysMain		0%	89.6 MB	0 MB/s	0 Mbps

8

SOFTWARE APPLICATIONS

Demonstration of DJMol - 1

UV-Visible Spectrum of Silver Nanocluster using DFTB+

To demonstrate the basic functionality of the software, here we describe how an UV-Visible spectrum of a silver nanocluster can be calculated using DFTB+ code in conjunction with the DJMol software. There are three steps to calculate the spectrum, *viz.* (1) Obtain the structure of Ag nanocluster, (2) Optimize the structure and create the corresponding HSD file for the TD-DFTB calculation and, (3) Plot the convoluted Spectrum data. Note that we arbitrarily choose Ag_{177} nanocluster (its diameter is around 1.5 nm) for the TD-DFTB calculation. And the geometry of this cluster is generated by OpenMD package (in CygWin environment) by using experimental lattice constant of silver (4.08 A) and FCC lattice structure. The following commands are used in the *DJMol terminal* to obtain the Cartesian coordinate data of the system:

```
cmd.exe /c  nanoparticleBuilder --latticeConstant=4.08 --radius=15 Ag.omd -o
```

```
NP15.omd

cmd.exe /c  Dump2XYZ -i NP15.omd
```

The resultant coordinate data (XYZ file), can be visualized by DJMol program to generate a *dftb_in.hsd* file using the *DFTB+ Script Writer* tool with appropriate settings, for example, by adding a conjugate gradient method. See *SetUp* menu for this utility. This HSD file can be then used to generate an optimized Ag_{177} cluster by executing a DFTB+ calculation. It can be seen that this optimized cluster (See *geo_end.xyz* file for the data) is approximately spherical in symmetry. The optimized structure was again visualized and used in the *Script Writer* tool to make another HSD file to include instructions for Casida method (a TD-DFTB algorithm). The essential part of this script is:

```
...
ExcitedState = Casida {
    NrOfExcitations=100
    StateOfInterest=0
    Symmetry=singlet
    WriteTransitions=no
    WriteSPTransitions=yes
    WriteMulliken=no
    WriteCoefficients=no
    WriteEigenvectors=no
    WriteTransitionDipole=no
    OscillatorWindow=0.01
}
...
```

In all the DFTB calculation, we used *hyb-0-2* parameter set is used for the silver atoms. This newly generated HSD file is submitted to a remote Linux machine using the SSH utility (See *Extra* menu in the tool bar) of the DJMol and after finishing the calculation a SPX output file was retrieved in the local machine, which contains the oscillator strengths of different electronic excitations. It was then plotted with *UV-Vis spectrum* from the *Tool* menu to obtain the spectrum of the molecule. A Lorentzian convolution (with 0.035 eV of full width at half maximum) is used to obtain the spectrum as shown in the Fig.X. Note that the *absorption maximum* wavelength (λ_{max}) is located around 440 nm *i.*e. around the indigo-blue region. By plotting molecular orbital energy levels (using *MOLevels* tool and with the *detailed.out* file) one can seen that the Fermi level is near to 3.9 eV and band gap of this cluster is negligibly small (See the inset figure in Fig. X). And it qualitatively indicates that Ag_{177} behaves like a semi-metal.

Different sized nanoparticles can be constructed by applying the above said method to study *red shift* property of the system. See the Fig.XX. Experimentally this is a known fact for the Ag_n system (By comparing λ_{max} values of clusters, we can seen that, λ_{max} shifts towards UV region if we increase the size of the nano clusters.). This can be qualitatively justified by *particle in a box* analogy. Note that the diameter of Ag_{177} is longer than that of the Ag_{13} so that its electron can move longer along its diameter (the *box length*). This will reduce its ΔE values significantly (by increasing the size of the diameter of a silver cluster, a photon from the *red* region - a wavelength that corresponds to a lower energy - can cause an electronic excitation). For experimental UV-Vis spectra, see: https://www.sigmaaldrich.com/technical-documents/articles/materials-science/nanomaterials/silver-nanoparticles.html.

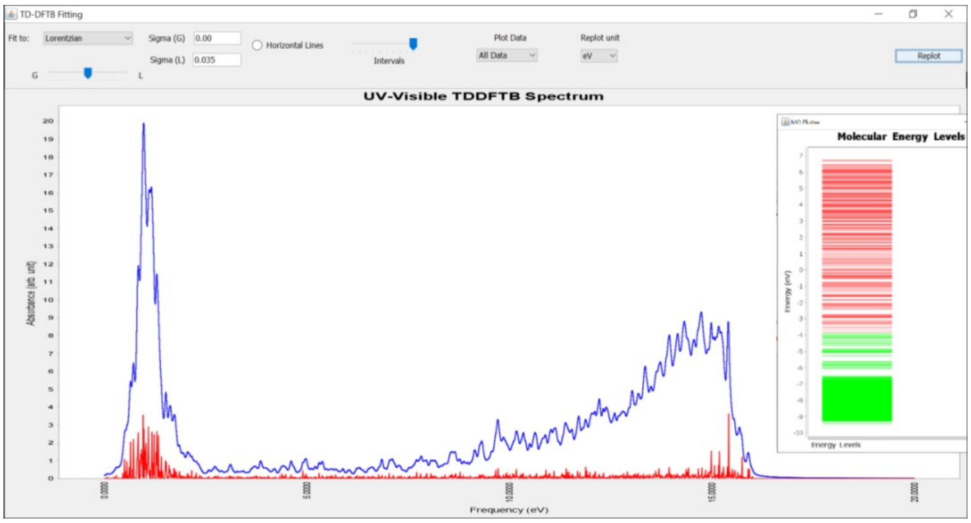

Fit to: Lorentzian
Sigma (G) 0.00
Sigma (L) 0.035
Plot Data: All Data
Replot unit: eV

Figure 8.1: The UV-Visible spectrum of Ag_{177} nanocluster as it is plotted with the *UV-Vis Spectrum* tool. The vertical red line shows the oscillator strengths of electronic transitions. The inset figure shows HOMOs (green lines) and LUMOs (red lines) of the system.

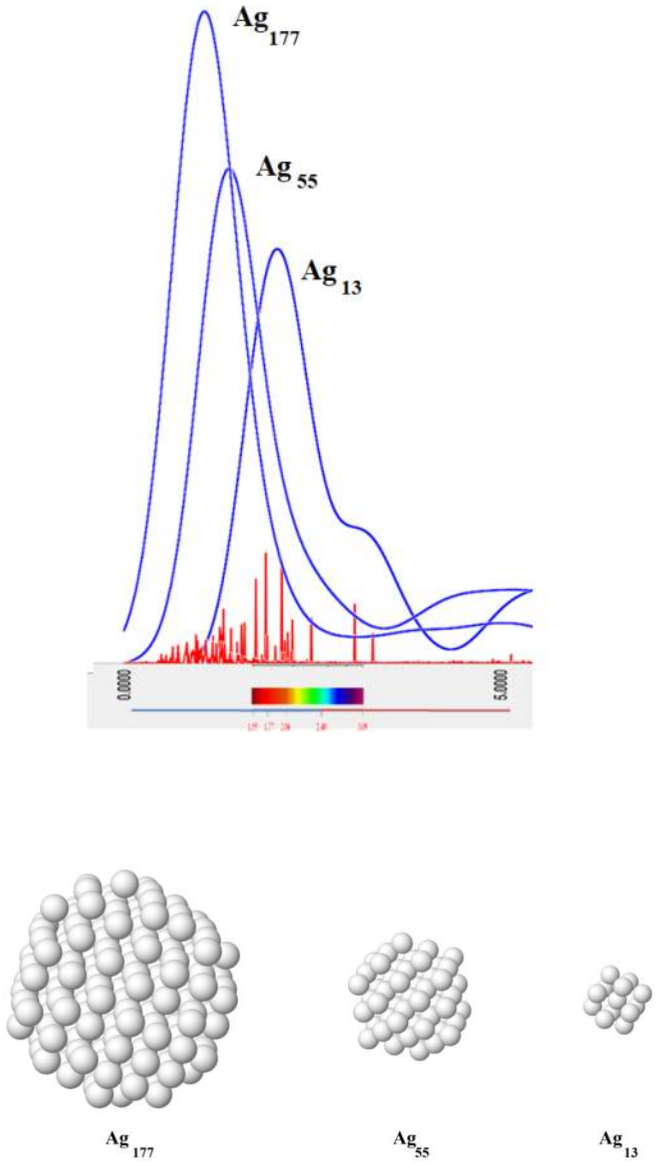


Figure 8.2: Sizes of different AG-nanoparticle and its UV-Vis spectrum. Note the *red shift* of

Demonstration of DJMol - 2

Frontier Orbitals of C_{60} Calculated with Siesta

Volumetric MO data can be made by using executing steps described in **Denchar Utility**, systematically using the Siesta add-on. The Siesta add-on can be invoked by:

Execute ➤ Siesta Calculator

In this section we shall obtain the *frontier orbitals* of Buckminster fullerene. See the example file *Fullerene.fdf* under the Siesta_Samples folder for its coordinate (note that it is an un-optimized structure; its optimization procedure is left to the user as an additional exercise). Since we used a molecular system in a unit cell, the default k-point (*Γ*-point) is used to represent electronic wavefunction coefficients. Note that HOMO and LUMO coefficients are at 120 and 121'th ***k***-points. See the Siesta Manual for more details.

And change the main setting of the fdf file as (ie. except the coordinate as it is inserted between, *AtomicCoordinatesAndAtomSpecies* block):

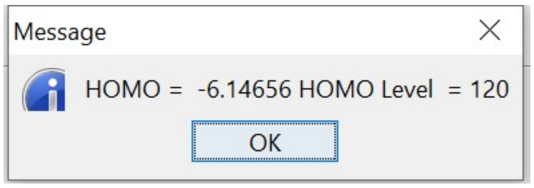
```
#-------------------------------------------------

#   IMPORTANT: DO NOT CHANGE SystemLabel

#-------------------------------------------------




SystemName          Fullerene molecule        # Descriptive name of the system

SystemLabel         siesta                    # Short name for naming files


NumberOfAtoms       60

NumberOfSpecies     1


SpinPolarized               T


WriteDenchar                T

WriteWaveFunctions          T

COOP.Write                  T
```

```
%block WaveFuncKpoints
  0.000 0.000 0.000 from 120 to 121           # Gamma wavefuncs Here, 120 = HOMO
%endblock WaveFuncKPoints


%block ChemicalSpeciesLabel
 1  6  C                                      # SpeciesIndex,AtomicNumber,SpeciesLabel
%endblock ChemicalSpeciesLabel


LatticeConstant       25.0 Ang
%block LatticeVectors
 1.0    0.0    0.0
 0.0    1.0    0.0
 0.0    0.0    1.0
%endblock LatticeVectors


AtomicCoordinatesFormat  Ang                  # Format for coordinates


%block AtomicCoordinatesAndAtomicSpecies
. . .
%endblock AtomicCoordinatesAndAtomicSpecies


WriteEigenvalues      T


#######################################################
# For the following parameters, default value is ok.
MeshCutoff               100. Ry       # Mesh cutoff. real space mesh (Ry)
PAO.BasisType            split         # Type of PAO basis set
PAO.EnergyShift          50 meV
PAO.BasisSize            DZ            # with polarization


# SCF options
MaxSCFIterations         60            # Maximum number of SCF iter
```

```
DM.MixingWeight          0.1          # New DM amount for next SCF cycle

DM.Tolerance             1.d-4        # Tolerance in maximum difference

                                      # between input and output DM

DM.NumberPulay           3

DM.UseSaveDM             F            # to use continuation files

MD.USeSaveXV             F


NeglNonOverlapInt        false        # Neglect non-overlap interactions


SolutionMethod           diagon       # OrderN or Diagon

ElectronicTemperature    6 K          # Temp. for Fermi smearing (Ry)


XC.Functional            GGA

XC.Authors               PBE
```

From this we can seen that, a double zeta type basis is used along with the PBE functional for the DFT calculation. A cubic box of 25 Å lengths is used to place the molecule.

Run the calculation as usual and after finishing the run, plot MO energy levels, using **MO Energies**, which also indicate the level of HOMO (from the first occupied level, which is level 1):

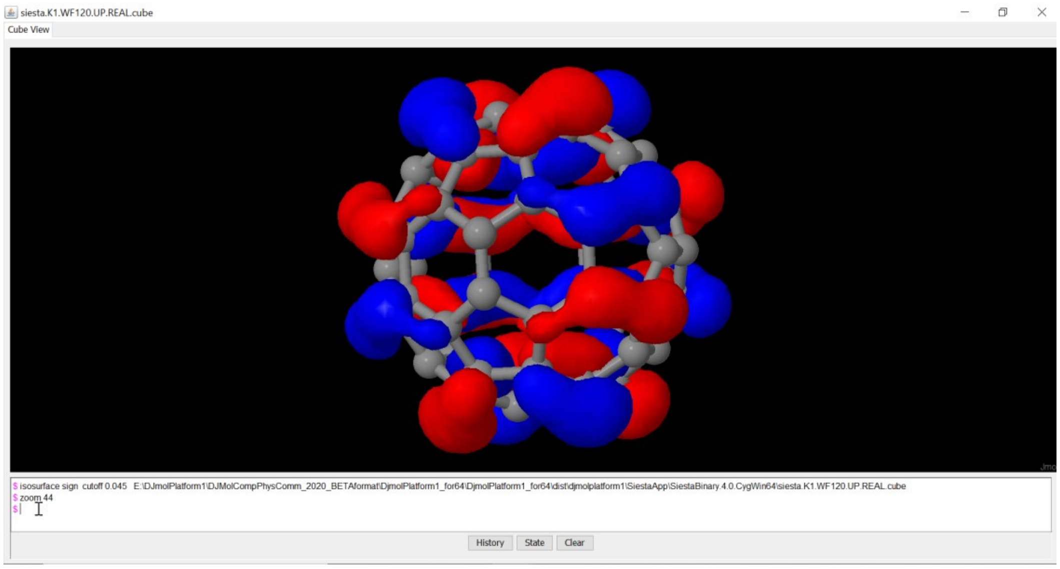

Message

HOMO = -6.14656 HOMO Level = 120

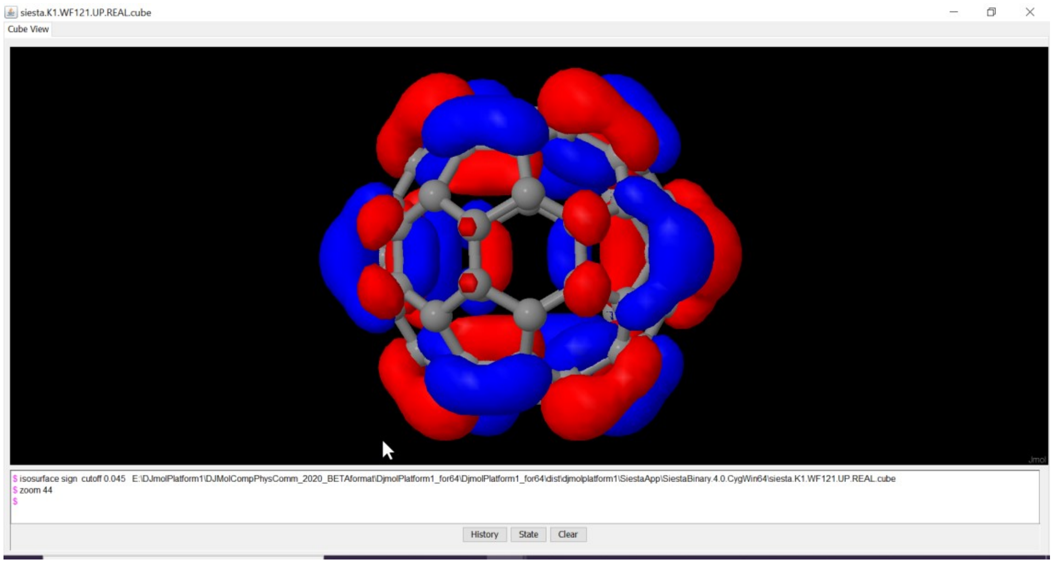
Then using, **Edit denchar3D.fdf** option insert the below text therein, and execute its next step, which **is Run Denchar**. After finishing **View 2D/3D Files** buttons can be clicked to view orbitals. Please see the next figures for the MOs of HOMO and LUMO (MO level, 120 and 121, respectively).

```
#-------------------------------------------------
#   Note: It is a Sample Script only
#         You have to Modify this according
#         to your system/requirement
#     See the Manual for more details
#-------------------------------------------------

Denchar.TypeOfRun                   3D

Denchar.PlotCharge                  T     # If .true. SystemLabel.DM   must be present
Denchar.PlotWaveFunctions           T     # If .true. SystemLabel.WFSX must be present

Denchar.CoorUnits       Ang
Denchar.DensityUnits    Ele/Ang**3

# set the mesh for wavefunction plot
Denchar.NumberPointsX 100
Denchar.NumberPointsY 100
Denchar.NumberPointsZ 100

Denchar.MinX          -12.5 Ang
Denchar.MaxX           12.5 Ang
Denchar.MinY          -12.5 Ang
Denchar.MaxY           12.5 Ang
Denchar.MinZ          -12.5 Ang
Denchar.MaxZ           12.5 Ang
```

```
%block ChemicalSpeciesLabel

 1   6   C       # Species index, atomic number, species label

%endblock ChemicalSpeciesLabel
```



```
$ isosurface sign  cutoff 0.045   E:\DJmolPlatform1\DJMolCompPhysComm_2020_BETAformat\DjmolPlatform1_for64\DjmolPlatform1_for64\dist\djmolplatform1\SiestaApp\SiestaBinary.4.0.CygWin64\siesta.K1.WF120.UP.REAL.cube
$ zoom 44
$
```



```
$ isosurface sign  cutoff 0.045   E:\DJmolPlatform1\DJMolCompPhysComm_2020_BETAformat\DjmolPlatform1_for64\DjmolPlatform1_for64\dist\djmolplatform1\SiestaApp\SiestaBinary.4.0.CygWin64\siesta.K1.WF121.UP.REAL.cube
$ zoom 44
$
```

Figure 8.3: Using *Γ*-point Siesta's SCF calculations yields MO coefficients. Here 120'th wavefunction at the *Γ*-point (alias MO-120) is HOMO (upper figure) and MO-121 is LUMO. The qualities can be adjusted with Denchar parameter settings.

Demonstration of DJMol - 3

OpenMD MD simulation of Gold-nanoparticle

In this demonstration, we would obtain the gold-gold atom pair distribution function from the OpenMD molecular dynamics trajectory. One can open OpenMD tool by:

Execute ➢ OpenMD Tool

In first we have to construct a Au-nanoparticle (with *icosahedron* symmetry). A basic starting script for the calculation is given below (its file name is gold.omd, and note HTML styled tags in it)

```
<OpenMD>

  <MetaData>


molecule{

name = "Au";


atom[0]{

type = "Au";

position(0.0, 0.0, 0.0);

     }

}


component{

type = "Au";

nMol = 1;

}


forceField = "SC";

forceFieldFileName = "SuttonChen.QSC.frc";
```

```
  </MetaData>

</OpenMD>
```

Using the command (in the OpenMD tool's command shell window) save the file, gold.omd ,with the above data.

`cmd.exe /c notepad.exe gold.omd`

Then run the following commands one after another:

```
cmd.exe /c .\icosahedralBuilder.exe  -o file.omd --shell=8 --
latticeConstant=4.08 --ico gold.omd
```

```
cmd.exe /c Dump2XYZ -i file.omd
```

```
cmd.exe /c more file.xyz
```

If everything is correct you will get a gold-nanoparticle structure as shown in the below figure-a. The above procedure construct a file called, file.omd which contains all the XYZ data of the nanoparticle.

However in order to start an MD run, we need initial velocities. For the purpose please run the command:

```
cmd.exe /c .\thermalizer -t 5 -o file-5K.omd file.omd
```

It will create a new file, file-5K.omd with new initial velovities and 5 Kelvin as its target temperature. Its main parameters are:

```
forceField = "SC";

forceFieldFileName = "SuttonChen.QSC.frc";


ensemble = "LHull";

targetTemp = 5;

targetPressure = 1;

viscosity = 0.1;
```

```
dt = 2.0;

runTime = 2E5;

sampleTime = 500.0;

statusTime = 4;

seed = 985456376;

usePeriodicBoundaryConditions = "false";

tauThermostat = 1E3;

tauBarostat = 5E3;
```

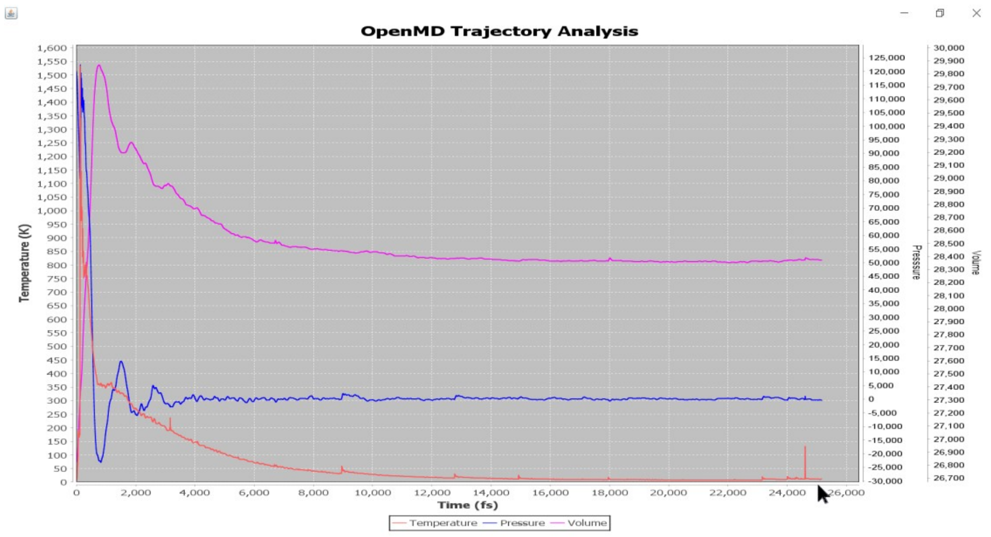
Then run the actual OpenMD molecular dynamics run as,

```
cmd.exe /c openmd file-5K.omd
```

This may take a while (2 hour), and its trajectory information will stored in the dump file. From this MD animation can be retrived as:

```
cmd.exe /c Dump2XYZ -b  -i file-5K.dump
```

Moreover, from the below figure one can see that the 5K thermalization is achieved in the MD run (note its convergence to 5K)



After that, a script utility to retrieve, $g_{Au\text{-}Au}(r)$, as:

```
cmd.exe /c StaticProps.exe -i file-5K.dump --gofr --sele1="select Au*" --sele2="select Au*"
```

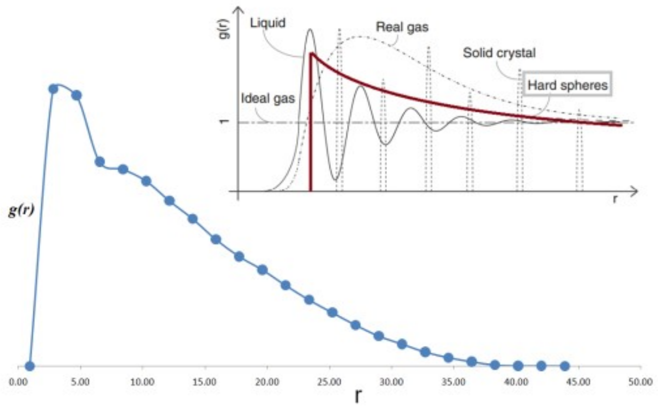


Figure 8.4: Pair correlation function (*g(r)*) obtained from the MD trajectory data which use only test-level parameters. Note the similarity with *g(r)* obtained by a hard-sphere model (inset figure courtesy, *J. Nanopart. Res, 13, 4277 (2011).*)

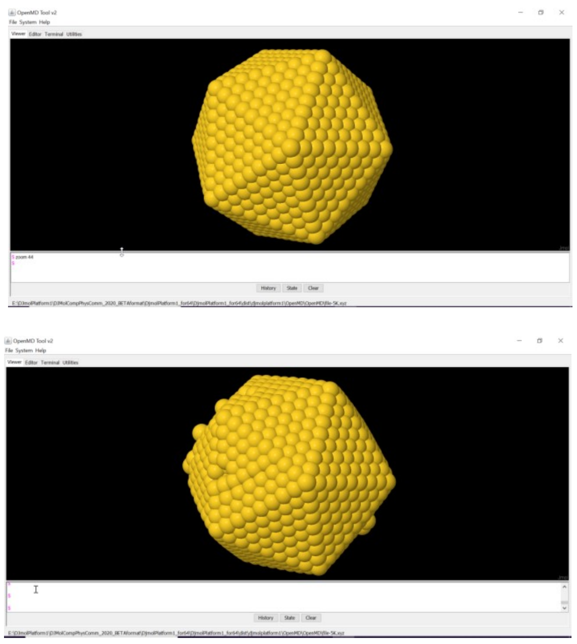


Figure 8.5: The initial (top) and final (bottom) snapshot of the MD run. Note that the system's initial temperature was 0K and it was equilibrated to 5K during the MD run.

Demonstration of DJMol - 4

ASE simulation of an adatom diffusion

Here we use ASE to obtain the energy profile of a physical process - *diffusion of an adatom* or adsorbed atom (Au) on the top of a metallic surface of Al atoms. Nudged elastic band (NEB) method is used for the calculation, in which an initial structure (where an adatom i.e. gold atom is placed in the left side) and a final structure (adatom is placed in the right side) of the process is used as its input data. For more details please visit ASE tutorial web page. Note that we used EMT calculators (it is a really fast albeit highly approximated potential function) for this calculation however one can use Siesta within ASE to obtain more realistic diffusion barrier.

Refer Chapter-5 for creating a **Python Project**. Then add an **Empty module** (InitialFinalStructure.py fie) into the **src** directory. Then add the following Python source into it:

```
from ase.build import fcc100, add_adsorbate
from ase.constraints import FixAtoms
from ase.calculators.emt import EMT
from ase.optimize import QuasiNewton

# 3x3-Al(001) surface with 3 layers and an
# Au atom adsorbed in a hollow site:
slab = fcc100('Al', size=(3, 3, 3))
add_adsorbate(slab, 'Au', 1.7, 'hollow')
slab.center(axis=2, vacuum=4.0)

# Fix second and third layers:
mask = [atom.tag > 1 for atom in slab]
slab.set_constraint(FixAtoms(mask=mask))

# Use EMT potential:
slab.calc = EMT()

# Initial state:
qn = QuasiNewton(slab, trajectory='initial.traj')
qn.run(fmax=0.05)

# Final state:
slab[-1].x += slab.get_cell()[0, 0] / 3
qn = QuasiNewton(slab, trajectory='final.traj')
qn.run(fmax=0.05)
```

Then Run the file (Run ► Run File). This will create, initial.traj and final.traj files which contains the initial and final structure respectively.

After this, the NEB calculation can be started (don’t forget to create a new Python file for this) and its Python source is given below:

```
# To change this license header, choose License Headers in Project
Properties.
# To change this template file, choose Tools | Templates
# and open the template in the editor.
from ase.io import read
from ase.constraints import FixAtoms
from ase.calculators.emt import EMT
from ase.neb import NEB
from ase.optimize import BFGS

initial = read('initial.traj')
final = read('final.traj')

constraint = FixAtoms(mask=[atom.tag > 1 for atom in initial])

images = [initial]
for i in range(9):
    image = initial.copy()
    image.calc = EMT()
    image.set_constraint(constraint)
    images.append(image)

images.append(final)

neb = NEB(images)
neb.interpolate()
qn = BFGS(neb, trajectory='neb.traj')
qn.run(fmax=0.05)
```

The energy profile can be saved in a PNG image by using the following script:

```
import matplotlib.pyplot as plt
from ase.neb import NEBTools
from ase.io import read

images = read('neb.traj@-11:')

nebtools = NEBTools(images)

# Get the calculated barrier and the energy change of the reaction.
Ef, dE = nebtools.get_barrier()

# Get the barrier without any interpolation between highest images.
Ef, dE = nebtools.get_barrier(fit=False)

# Get the actual maximum force at this point in the simulation.
max_force = nebtools.get_fmax()

# Create a figure like that coming from ASE-GUI.
fig = nebtools.plot_band()
fig.savefig('diffusion-barrier.png')

# Create a figure with custom parameters.
fig = plt.figure(figsize=(5.5, 4.0))
ax = fig.add_axes((0.15, 0.15, 0.8, 0.75))
nebtools.plot_band(ax)
fig.savefig('diffusion-barrier.png')
```

Finally, the neb.traj file, which contains the entire geometry of the process in binary format, is then used to obtain XYZ format by the following script:

```
import matplotlib.pyplot as plt
from ase.calculators.emt import EMT
from ase.neb import NEB
from ase.optimize import BFGS
from ase.io import read, write
import os

# read the last structures (of 5 images used in NEB)
images = read('neb.traj@-11:')

for i in range(0,  len(images)):
    atoms = images[i]
    #print(atoms.get_positions())
    write('delete.xyz', atoms)
    inFile = open(os.path.join( 'delete.xyz'), 'r')
    fileStr = inFile.read()
    outFile = open("neb_movie.xyz", 'a')
    outFile.write(fileStr)

# @type outFile
outFile.close()
inFile.close()
```

Additionally, using DJMol's xyz File loader (File▶Open Structure) the geometries of the process as it is saved in "neb_movie.xyz" can be displayed or animated (See the below figure).

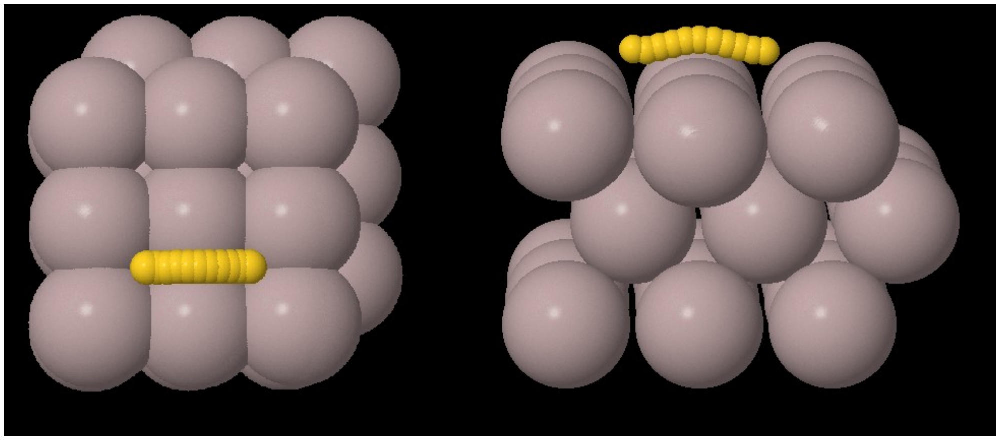


Figure 8.6: Diffusion of gold atom over the aluminium surface (FCC) from one hollow site to another, in different perspectives; Nine NEB images were used for this NEB profile.

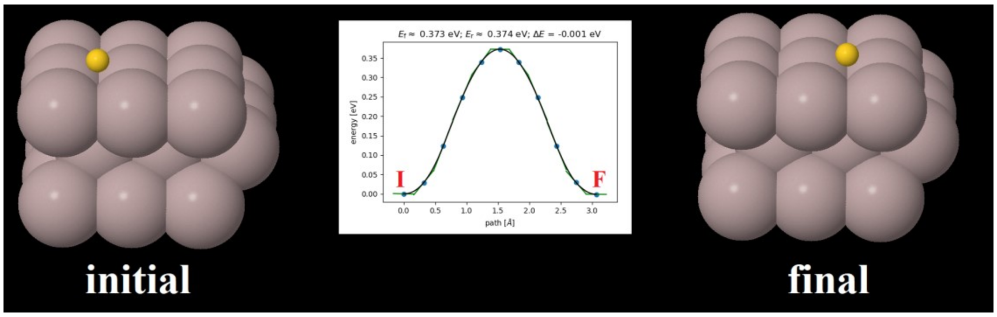


Figure 8.7: Starting (I) and ending (F) geometries of the Diffusion of gold atom; The diffusion barrier (around 0.35 eV) is shown in the inset.

APPENDICES

PYTHON SCRIPTING FOR DJMOL APPLICATIONS

Since DJMol uses Python for its scripting (and in ASE), a basic knowledge of Python (version 3) is highly appreciated. This appendix is based on the book, ***Think Python*** 2nd Edition by Allen B. Downey (Creative Commons Attribution-NonCommercial 3.0 Unported License). Some familiarity with programming is assumed for this appendix.

How to use Python within the DJMol?

To use DJMol, Python is a mandate tool and it should be installed. For this, you have to install the Python (windows version 3 or later) immediately after the installation of the DJMol program into your system. See Appendix - 3 for its details (including the details of running a python in the DJMol)

Basics of python scripting

Python is designed as an interpreter language and it is widely used in scripting purposes. It supports both procedural and object-oriented paradigm.

Variable assignment and Operators

In Python variable assignment is performed by, **=** operator. It also provides operators, which are special symbols that represent operations like addition or multiplication. The operators **+**, **-**, **/**, and ***** perform addition, subtraction, division, and multiplication, respectively. And the operator ****** performs exponentiation; that is, it raises a number to a power. An illustration (hopefully self-explanatory!) of these commands are following (here, >>> string represents a command prompt; for the comment one can use #).

```
>>> x=5

>>> x

5

>>> x+2 # a comment

7

>>> x-2

3

>>> x*2

10

>>> x**2

25

>>> x%2

1

>>> x/2

2.5
```

Note that in numerical computation one need to use integers and decimals. Though it doesn't require explicit declaration of type of the variables, there may be times when you want to specify the type of a variable. This can be done with casting operation.

Casting in python is therefore done using constructor functions:

- int() - constructs an integer number from an integer literal, a float literal (by rounding down to the previous whole number), or a string literal (providing the string represents a whole number)
- float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)

- str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

```
>>> a=1


>>> type(a)

<class 'int'>


>>> b=3.1415


>>> type(b)

<class 'float'>


>>> a+b

4.141500000000001


>>> b**b

36.45491472872008


>>> y = int(2.8)

>>> y

2


>>> x = float(1)

>>> x

1.0
```

```
>>> z = float("3")

>>> z

3.0
```

String, List, Methods etc.

Apart from ***int***, ***string*** type is also used in python. Its initiation is simple (you can use either ‘ or “ to make string):

```
>>> my_string = 'thisStringI'

>>> my_string

'thisString'


>>> my_string = "thisString"

>>> my_string

'thisString'
```

Like a string, a list is a sequence of values (perhaps it may be the Pythons most important built-in data type; unlike C or Fortran python does not have an ***Array*** data type). In a string, the values are characters; *in a **list,** they can be **any type***. The values in a list are called ***elements*** or sometimes ***items***. There are several ways to create a new list; the simplest is to enclose the elements in square brackets:

```
>>> a = 'is'

>>> b = 'nice'

>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]


>>> my_list[0]

'my'
```

Note that a list can be used as an Array (and its index starts from 0 and not from 1, like a C array).

List Manipulation Rules

`>>> my_list[0]`
`'my'`
	Select the first item

`>>> my_list[1:3]`
`['list', 'is']`
	Select items at index 1 and 2

`>>> my_list[1:]`
`['list', 'is', 'nice']`
	Select items after index 0

`>>> my_list[:3]`
`['my', 'list', 'is']`
	Select items before index 3

`>>> my_list[:]`
`['my', 'list', 'is', 'nice']`
	Copy my_list

List Operations

```
>>> my_list + my_list

['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']



>>> my_list * 2

['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
```

Methods for the Lists

The Methods are equivalent to functions/subroutines; Since the ***List*** is an object a method for that list is called by a **DOT (.)** operator immediately after the Object name, as it is illustrated below:

`>>> my_list.index(a)`
 	
`2`
	Get the index of an item

`>>> my_list.count(a)`
 	
`1`
	Count an item

`>>> my_list.append('NewOne')`	Append an item at a time

`>>> my_list`
 	
`['my', 'list', 'is', 'nice', 'NewOne']`
	It shows the appended List

`>>> my_list.remove('is')`	

`>>> my_list`
 	
`['my', 'list', 'nice', 'NewOne']`
	Remove an Item from the list

Libraries

In Python a good collection of libraries are available say, for doing math, or to do string/array manipulation, Numerical computations, graphics etc.

The math library of python can be imported like:

```
>>> import math


>>> math.sqrt(9.99)

3.1606961258558215
```

For ASE scripting, most of the times you need to use NumPy and Scipy libraries (and it is not shipped with standard Python, so one needs to install it, see the next appendix). If you installed NumPy you can call it like:

`>>>import numpy`
 	
`Or`
 	
`>>>import numpy as np`
	Importing NumPy Lib.
	
	Constructing an Array

`>>> Array=np.array([[1, 2], [3, 4]])`	

`>>> Array`
`array([[1, 2],`
`       [3, 4]])`
	Print that Array

`>>> np.transpose(Array)`
`array([[1, 3],`
`       [2, 4]])`
	Transpose of the Matrix

`>>> Array.dot(Array)`
`array([[ 7, 10],`
`       [15, 22]])`
	Matrix Multiplication

<u>Basics of ASE scripting</u>

Hopefully one can be now understood ASE scripting. A sample ASE script is given below with some explanations. The aim is to calculate N2 molecular energy with EMT calculator (Effective Medium Potential, a crude empirical model generally used for testing purposes or for very large scale MD). Note that the following code has six lines (didn't use line-breaker)

`from ase import Atoms`	Import Atoms object
`from ase.calculators.emt import EMT`	Import EMT

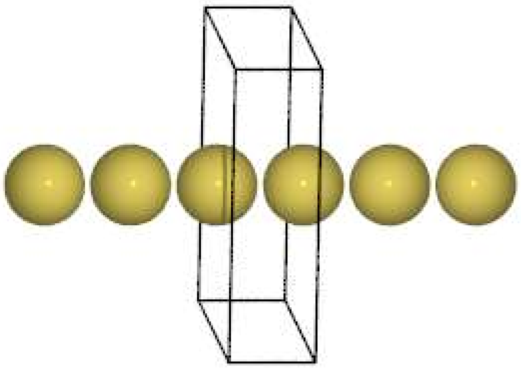
`d = 1.1 # Bond Length in Angstrom`
 	
`molecule = Atoms('2N', [(0., 0., 0.), (0., 0., d)])`
	Define a **molecule** with Atoms Object

`molecule.set_calculator(EMT())`	That molecule is **attached** with EMT
`e_molecule = molecule.get_potential_energy()`	**Calculate PE** (this Really invoke the EMT calculations)
`print ('Nitrogen molecule energy: %5.2f eV' % e_molecule)`	Printing the PE

Note that the important step is to create the **Atoms** object - which is a collection of atoms. Everything else is based on this **Atoms object (**for example DFT **calculators** can be attached to this object to get DFT total energy**).**

A little more advanced example is following (an infinite gold wire with BL= 2.9 A).

```
from ase import Atoms

d = 2.9

L = 10.0

wire = Atoms('Au',

             positions=[[0, L
/ 2, L / 2]],

             cell=[d, L, L],

             pbc=[1, 0, 0])
```



Courtesy: ASE Camd

A list of ***generally used*** GET/SET methods for the Atoms Object is:

GET Methods	SET Methods
get_atomic_numbers()	set_atomic_numbers()
get_initial_charges()	set_initial_charges()
get_charges()	set_chemical_symbols()
get_chemical_symbols()	set_initial_magnetic_moments()
get_initial_magnetic_moments()	set_masses()
get_magnetic_moments()	set_momenta()
get_masses()	set_positions()
get_momenta()	set_scaled_positions()

get_forces()	set_tags()
get_positions()	set_velocities()
get_potential_energies()	
get_scaled_positions()	
get_stresses()	
get_tags()	
get_velocities()	

In essence, the **Set methods** are used to supply (the necessary) information for the **Atoms** object, whereas **Get methods** are applied to get useful information (usually after setting a calculator)

INSTALLATION OF PYTHON AND NUMPY

As it is said before, one need to install Python (3.x, 64 bit) scripting language and NumPy on the system ***before*** the DJMol installation.

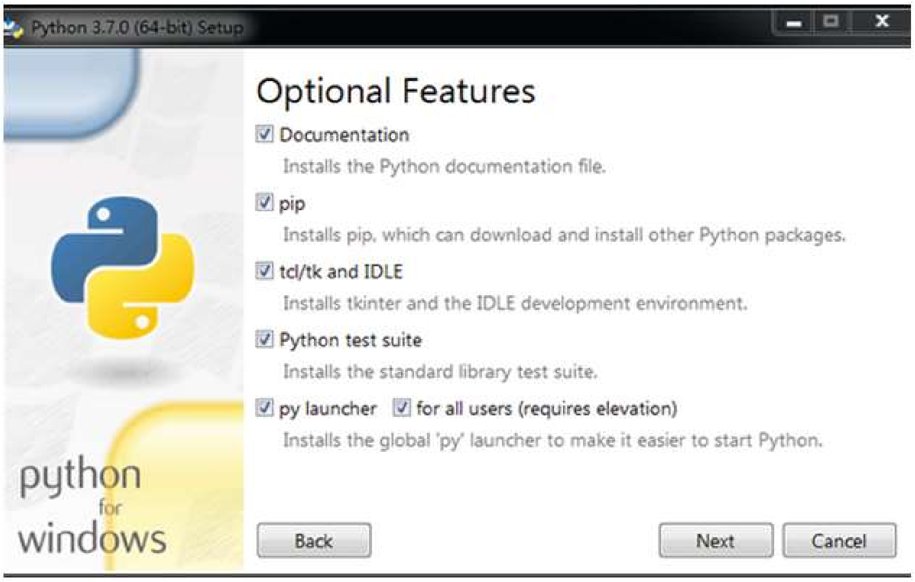
Using this tutorial, one is expected to get sufficient information on how to install a **64 bit** version of **python 3**and **NumPy** on a **Windows**System.

I. Python Installation

1) First, download a 64 bit version of any 3.7.x from python.org. Clicking on the following link automatically downloads the required python

https://www.python.org/ftp/python/3.7.2/python-3.7.2.exe

2) Install the above file with <**pip**> support, and for that select custom installation and check all the boxes in the Optional Features window.



Python 3.7.0 (64-bit) Setup

Optional Features

☑ Documentation
Installs the Python documentation file.

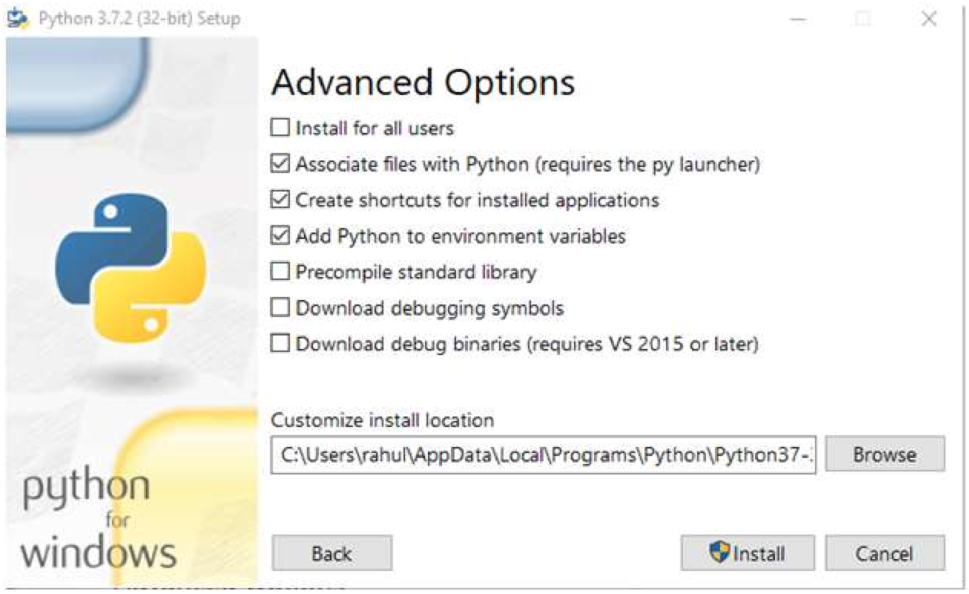
☑ pip
Installs pip, which can download and install other Python packages.

☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

☑ Python test suite
Installs the standard library test suite.

☑ py launcher ☑ for all users (requires elevation)
Installs the global 'py' launcher to make it easier to start Python.

Note: don't forget to check (See next figure) "***Add Python to environment variables***" for accessing python using the command prompt.



Advanced Options

☐ Install for all users
☑ Associate files with Python (requires the py launcher)
☑ Create shortcuts for installed applications
☑ Add Python to environment variables
☐ Precompile standard library
☐ Download debugging symbols
☐ Download debug binaries (requires VS 2015 or later)

Customize install location

C:\Users\rahul\AppData\Local\Programs\Python\Python37-

Python Modules Installations

After installing 3.X Python in Windows, you can simply open a command window (using Administrative privilege) and type:

```
python -m pip install numpy
python -m pip install scipy
python -m pip install ase==3.17.0
python -m pip install matplotlib
```

To install **numpy, scipy, ase** and **matplotlib** libraries (note: it will install latest versions except for ASE).

(For a specific installation you can also do:
	python -m pip install nameOfPackage==x.y.z,
where x,y,z are version numbers)

II. <u>NumPy Installation</u>

If the above command: **`python -m pip install numpy`** fails, this method can be used to install the NumPy in Windows OS.

Disclaimer: Since NumPy doesn't have an official build for windows, we'll be downloading an **unofficial version** from *https://www.lfd.uci.edu/* which is maintained by Christoph Gohlke, Laboratory for Fluorescence Dynamics, University of California, Irvine.

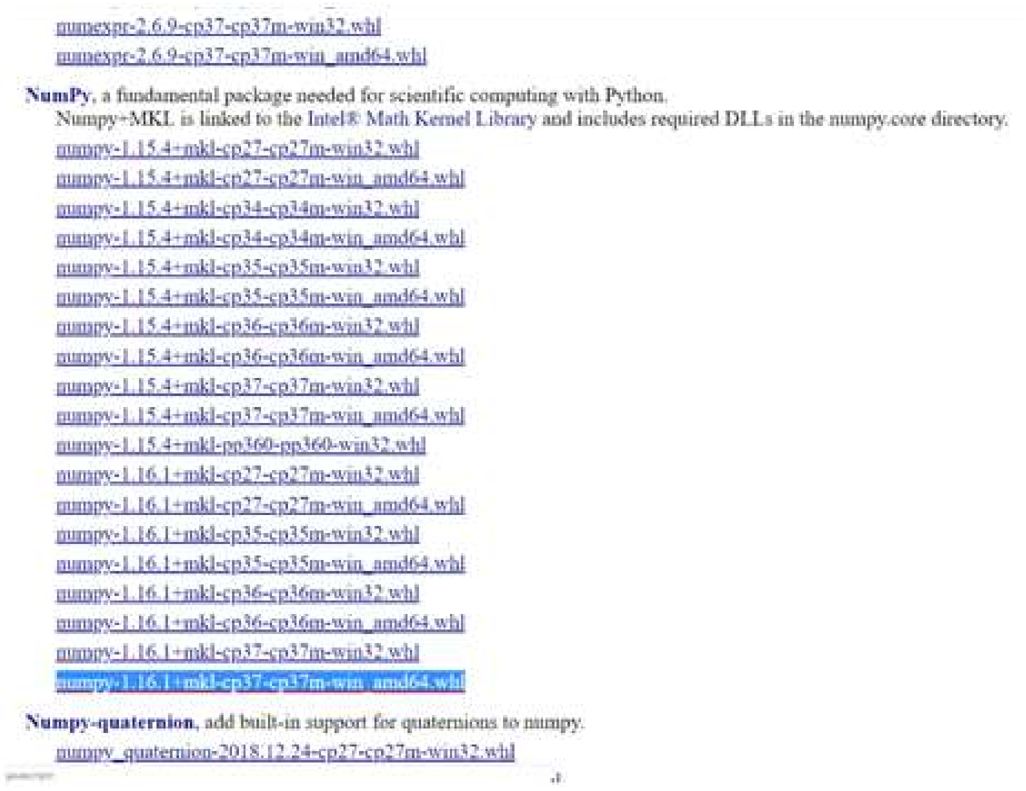
1) Download the .whl file of NumPy using the link to your Desktop. Clicking on the following link automatically downloads the required python *https://download.lfd.uci.edu/pythonlibs/r5uhg2lo/numpy-1.16.1+mkl-cp37-cp37m-win_amd64.whl*

If the file is not downloaded to the Desktop, Copy the <.whl> file to Desktop.

Note: If the above link is not working, goto

https://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy

and download the 1.16.1 version of **NumPy** (Not Numpy-quaternion).

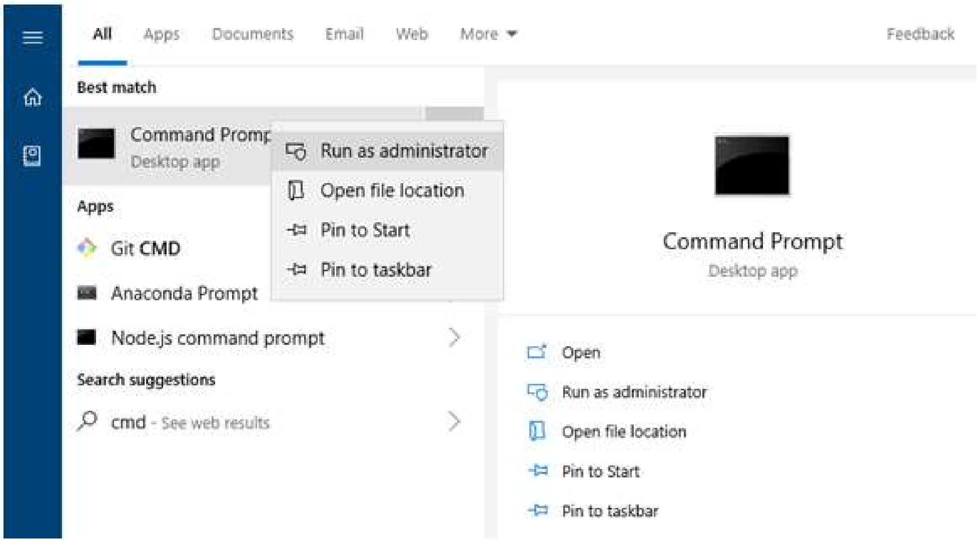


numexpr-2.6.9-cp37-cp37m-win32.whl
numexpr-2.6.9-cp37-cp37m-win_amd64.whl

NumPy, a fundamental package needed for scientific computing with Python.
Numpy+MKL is linked to the Intel® Math Kernel Library and includes required DLLs in the numpy.core directory.
numpy-1.15.4+mkl-cp27-cp27m-win32.whl
numpy-1.15.4+mkl-cp27-cp27m-win_amd64.whl
numpy-1.15.4+mkl-cp34-cp34m-win32.whl
numpy-1.15.4+mkl-cp34-cp34m-win_amd64.whl
numpy-1.15.4+mkl-cp35-cp35m-win32.whl
numpy-1.15.4+mkl-cp35-cp35m-win_amd64.whl
numpy-1.15.4+mkl-cp36-cp36m-win32.whl
numpy-1.15.4+mkl-cp36-cp36m-win_amd64.whl
numpy-1.15.4+mkl-cp37-cp37m-win32.whl
numpy-1.15.4+mkl-cp37-cp37m-win_amd64.whl
numpy-1.15.4+mkl-pp360-pp360-win32.whl
numpy-1.16.1+mkl-cp27-cp27m-win32.whl
numpy-1.16.1+mkl-cp27-cp27m-win_amd64.whl
numpy-1.16.1+mkl-cp35-cp35m-win32.whl
numpy-1.16.1+mkl-cp35-cp35m-win_amd64.whl
numpy-1.16.1+mkl-cp36-cp36m-win32.whl
numpy-1.16.1+mkl-cp36-cp36m-win_amd64.whl
numpy-1.16.1+mkl-cp37-cp37m-win32.whl
numpy-1.16.1+mkl-cp37-cp37m-win_amd64.whl

Numpy-quaternion, add built-in support for quaternions to numpy.
numpy_quaternion-2018.12.24-cp27-cp27m-win32.whl

2) Then run command prompt as administrator: Type <cmd> in start menu and right click on command prompt and click on "**Run as administrator**" (See the next figure).



3) Navigate to your Desktop where the <.whl> file is saved. Type the following command in your command prompt and press Enter.

cd %systemdrive%\users\%username%\Desktop

Or manually find your Desktop path and enter like:

cd C:\Users\staff\Desktop

4) Install NumPy using the <.whl> file; Type this command in your command prompt and press Enter.

pip install "numpy-1.16.0+mkl-cp37-cp37m-win_amd64.whl"

WINDOWS SUBSYSTEM LINUX IN DJMOL

For more advanced scripting/calculations one may want to use a real Linux operating system (instead of Cygwin emulators) in conjunction with DJMol. For this one can use WSL in the Windows 10 OS. It will give a real Linux OS and hence it guarantees a cent percentage of portability of C/C++ or Fortran codes into the Windows system.

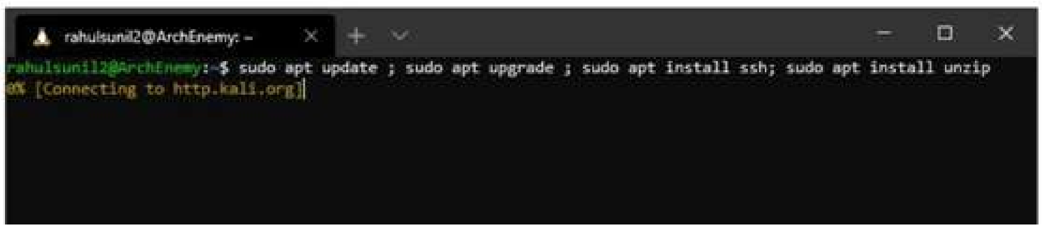
Herein we give a short description of invoking of WSL in the DJMol platform (by assuming that you already have a WSL in your PC; otherwise it can be freely downloaded from Microsoft store, visit: *https://docs.microsoft.com/en-us/windows/wsl/install-win10*

(a) Setting up the WSL

1. Update the Linux packages being used by WSL with the following commands in

your WSL Ubuntu command window:

```
sudo apt update

sudo apt upgrade

sudo apt install unzip

sudo apt install ssh
```



```
rahulsunil2@ArchEnemy:~$ sudo apt update ; sudo apt upgrade ; sudo apt install ssh; sudo apt install unzip
0% [Connecting to http.kali.org]
```

2. Your Windows installation is probably already using port 22 for its SSH server, we need to change WSL's SSH server to listen to a different port.

```
sudo nano /etc/ssh/sshd_config
```

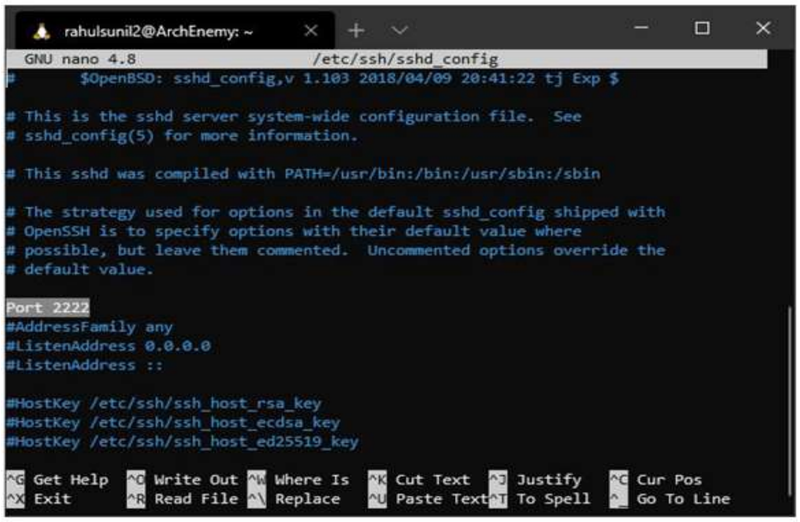
Change the lines:

# What ports, IPs and protocols we listen for

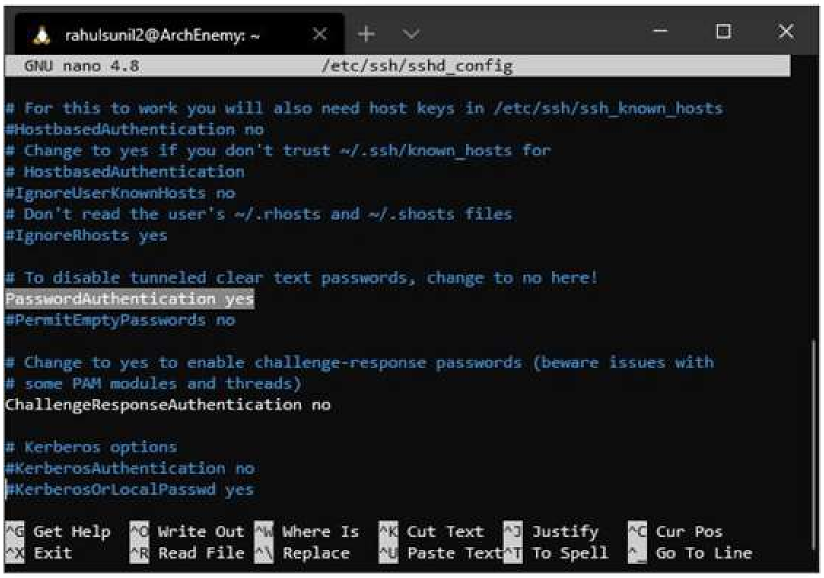
```
Port 22
```

to

```
Port 2222
```



```
  GNU nano 4.8                 /etc/ssh/sshd_config
#       $OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Port 2222
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```



```
  GNU nano 4.8                    /etc/ssh/sshd_config

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

The WSL SSH server is initially set up to use key files for authentication.

To allow authentication with passwords, change:

# Change to no to disable tunnelled clear text passwords

```
PasswordAuthentication no
```

to

```
PasswordAuthentication yes
```

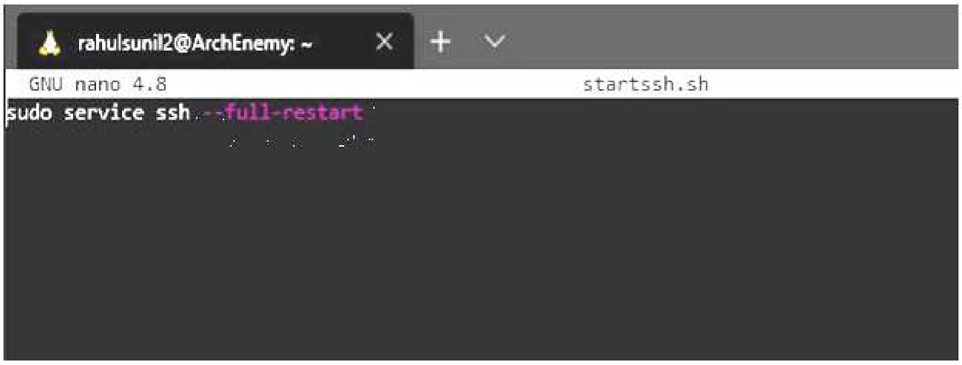
Use **Ctrl-o** to save the file. Use **Ctrl-x** to exit the nano text editor.

3. Before using the WSL SSH server you must stop and restart the SSH server. You will perform this step every time that you use WSL as the remote host from NetBeans. I suggest placing the command in a script file that you can execute.

```
nano startssh.sh
```

Insert the following text into the script file:

```
sudo service ssh --full-restart
```



```
rahulsunil2@ArchEnemy: ~
  GNU nano 4.8                                   startssh.sh
sudo service ssh --full-restart
```

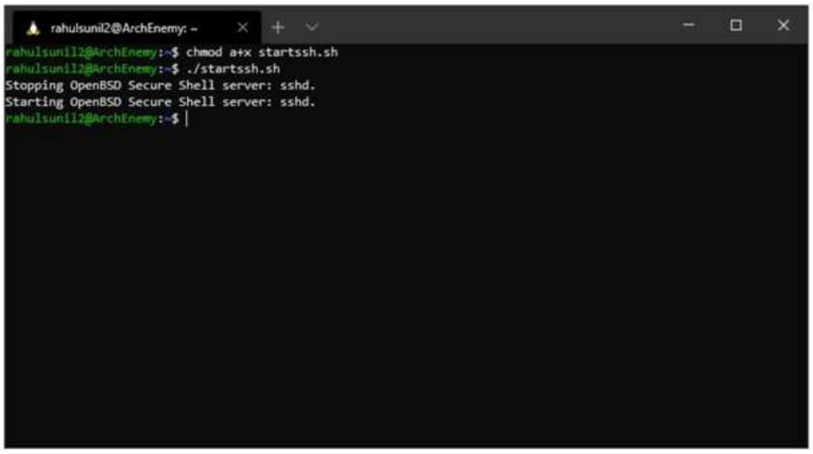
Use **Ctrl-o** to save the file. Use **Ctrl-x** to exit the nano text editor.

Back at the command line, use the following command to mark the script file as executable:

```
chmod a+x startssh.sh
```

To restart the SSH server, use the following command:

```
./startssh.sh
```

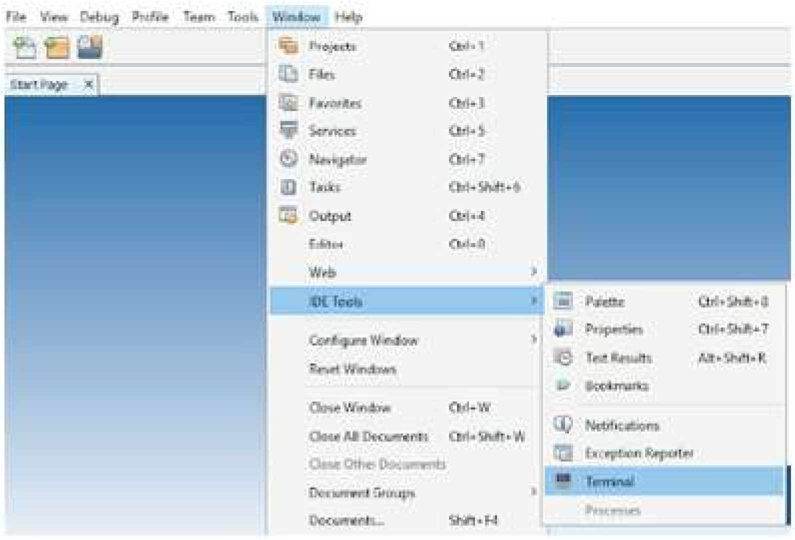


```
rahulsunil2@ArchEnemy:~$ chmod a+x startssh.sh
rahulsunil2@ArchEnemy:~$ ./startssh.sh
Stopping OpenBSD Secure Shell server: sshd.
Starting OpenBSD Secure Shell server: sshd.
rahulsunil2@ArchEnemy:~$ |
```

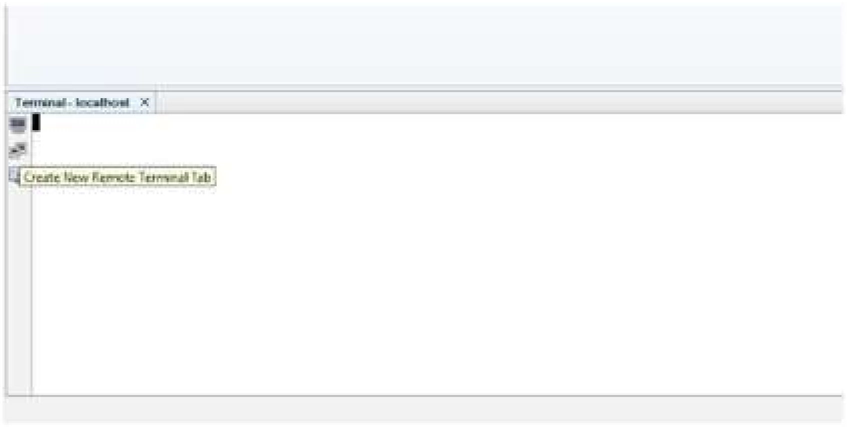
(b) Setting up the Netbeans IDE

1. Open the Terminal and invoke the Terminal as:

    Window → IDE Tools → Terminal



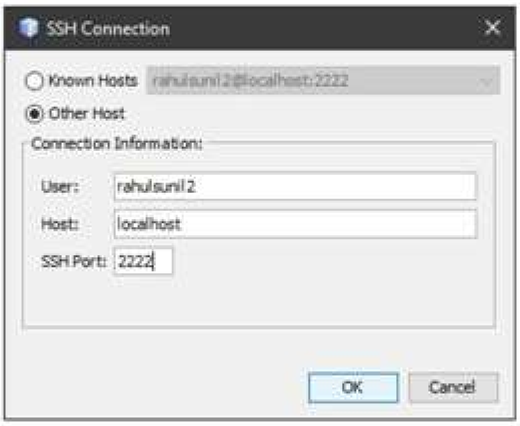
2. Create New Remote Terminal Tab.



3. Enter your credentials.

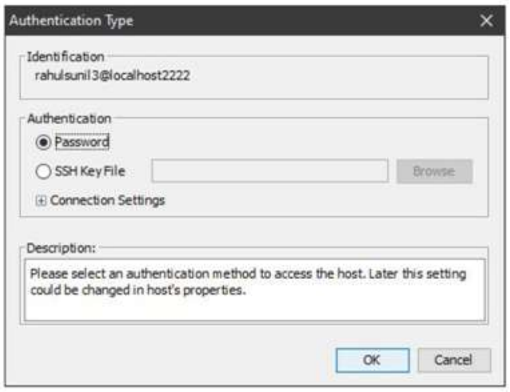
```
User : <your-username>

Host : localhost
```

```
SSH Port : 2222
```

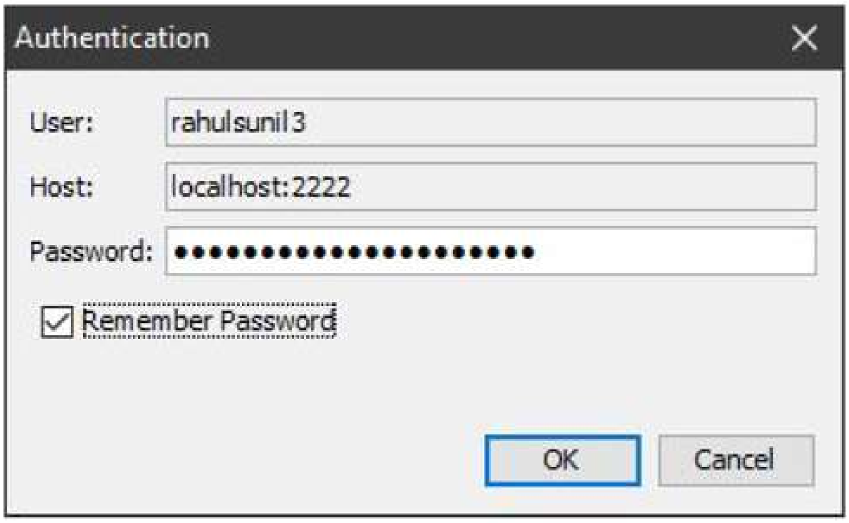


Click OK.

4. Choose Password.


Authentication Type
Identification
rahulsunil3@localhost2222
Authentication
Password
SSH Key File
Browse
Connection Settings
Description:
Please select an authentication method to access the host. Later this setting could be changed in host's properties.
OK
Cancel

5. Enter your Linux Password.


Authentication

User: rahulsunil3

Host: localhost:2222

Password: ●●●●●●●●●●●●●●●●●●●●●●

☑ Remember Password

Click OK.

#Recommended: Check Remember Password box.

The process is finished; Now one can use WSL in DJMol.

```
rahulsunil2@ArchEnemy:~$
```

SAVED DATA

DJMol generate a number of text based *temporary* data, and these are nothing but a set of post-processing data from various visualizers or converters were temporarily stored in different directories and it can be used for a numerical comparison (for example, how MO levels of DFTB+ are different from that of Siesta for a given molecule). A list of such selected post processed files with its description is shown in the Table 1.

PESScan\2DPESscanmovie.xyz	Geometries of molecules used for the PES
PESScan\2DPESscan.dat	Energies and geometries of the PES
PESScan\View2DPES.dat	Coordinates (initial, final step size) and PES energy
SCRATCH\MODFTB.dat	MO energies in DFTB+
SCRATCH\MOSiesta.dat	MO energies in Siesta
SCRATCH\ForceDFTB.dat	Cartesian Force components in DFTB+
SCRATCH\ForceSiesta.dat	Cartesian Force components in Siesta
SCRATCH\SCFSiesta.txt	SCF convergence in Siesta
SCRATCH\Uvdata_2.csv	Fitted UV-Vis data
SCRATCH\Uvdata_1.csv	UV-Vis oscillator strength for the spectrum
SCRATCH\MDDFTBmovie.xyz	Trajectory of MD in XYZ animation format
SCRATCH\TotalMD.dat	Total energies of MD run
SCRATCH\KineticMD.dat	Kinetic energies of MD run
SCRATCH\PotentialMD.dat	Potential energies of MD run
SCRATCH\VelocityAC.out	IR spectrum from MD using velocity autocorrelation
SCRATCH\DipoleAC.out	IR spectrum from MD using dipole moment autocorrelation
SCRATCH\StdOrientation.xyz	Standard orientation of the given molecule

SCRATCH\MullikenDFTB.dat	Mulliken charges from DFTB+
Input\dftb_in.hsd	DFTB+ script writer out file
Input\siestaTEMP.fdf	Siesta script writer out file
ModesBinary\modes_in.hsd	DFTB+ file used to construct vibrational data
ModesBinary\waveplot_in.hsd	DFTB+ file used to construct MO and its density data
ModesBinary\modes.xyz	DFTB vibrational modes and frequencies
ModesBinary*.cube	DFTB Cube files for MOs and Densities
SiestaApp\SiestaBinary.4.0CygWin64\*.cube	Siesta Cube files
SiestaApp\SiestaBinary.4.0.CygWin64\SiestaVibModes.xyz	Siesta vibrational modes and frequencies
SiestaApp\SiestaBinary.4.0.CygWin64\BAND.bands	Siesta (DFT) bands
SiestaApp\SiestaBinary.4.0.CygWin64\wannier.bands	Siesta Wannier bands
SiestaApp\SiestaBinary.4.0.CygWin64\wannier_*.xsf	Siesta Wannier orbitals
OpenMD\.	All OpenMD in and out files
Database\.	All downloaded Structural files from online repositories

DEMONSTRATION VIDEOS

Please visit: *https://www.youtube.com/channel/UCNczegqwli6gnuo6eqNJSPg* for Demo videos. The list is (as of October 2020):

- Windows OpenMP settings | Executing Stand-alone DFTB+
- DJMol Demo on Point Group Symmetry Detection
- Running a DFTB Calculation from DJMol
- Phonon Density of States (DOS) of Aluminium Bulk: DJMol + ASE Python Scripting
- PES demo in DJMol (Linux)
- Demo on Molecular Vibration of Water Molecule
- Demo on Partial Charges And File Conversions
- Generating/Displaying Molecular Orbitals in DJMol
- DFTB+ Input script Writer in DJMol
- DJmol/Netbeans : How to install ASE, Matplotlib with PIP
- Adding Python Plugin in Netbeans or in DJMol
- Siesta's Charge density and/or electronic wave functions: 2D Contours/Surfaces
- Siesta's Charge density and/or electronic wave functions: 3D cube files
- Remote Submission using SSH tool in DJMol (64v)
- Band diagram and DOS (density of state) plot of Aluminum FCC
- Molecular Dynamics Analysis
- UV-visible spectrum in DJMol
- Z-Matrix Structure Editor of DJMol
- Wannier AddOn in DJMol (with Siesta)
- DJMol Win64 Installation from its ZIP distribution File
- DJMol Version Control with Github
- Terminal use in DJMol
- Building of DJMol with Netbeans 8.2 (Win64OS)
- Radial Distribution Function from ASAP MD Calculation in the DJMol System
- Burning of iso-octane fuel at 2500 K (molecular dynamics with DFTB+).

COMPILING/INSTALLATION OF DJMOL AND ADD-ONS

General Information

A familiarity with programming using Netbeans IDE is assumed. All of the DJMol Program was compiled by using *Netbeans IDE, v 8.2 (64bit)* in Windows 64 OS. This program and all other mandate packages for compiling the software are available at free of cost and to download these packages please see: *http://www.djmol.info/download.html*

The needed Packages are:

JAVA 1.8 (jdk-8u201-windows-x64.exe)
Netbeans 8.2 (netbeans-8.2-javaee-windows.exe)
Python plug-in (2017-08-29-nbpython-nbms.zip)
Python 3.7.2 (python-3.7.2-amd64.exe)
Pip script (get-pip.py)
Numpy (numpy-1.16.0+mkl-cp37-cp37m-win_amd64.whl)

We also used (64 bit) the following Python packages with version numbers:

SciPy (scipy 1.2.1)
ASE (ASE 3.17.0)
Matplotlib (Matplotlib 3.0.3)

And these programs can be downloaded by using pip (use pip3) using a command window.

[A] Compiling Main Program, <DJMolplatform>

[1] Open the Project Directory, <DJmol Platform v2.1> and **Compile** and **Build** DJMol program. See the YouTube channel for its demonstration.

[2] Then make a Zip Distribution (by Selecting project <DJmol Platform v2.1> followed by selecting <**Package as**>, <**Zip Distribution**>). A new <dist> folder will be created and it holds the Zip file.

[B] Installing Program, <DJMolplatform>

[3] Go to <**dist**> folder and unzip the file and Copy **all** the files/folders from the <**Auxiliary**> folder into <**dist\djmolplatform1**>. This is shown in the demonstration.

[4] Go to <**dist\djmolplatform1\etc**> and replace the following line in the <**djmolplatform1.CONF**> file:

`default_options="--branding djmolplatform1 -J-Xms24m -J-Xmx64m"` to

```
default_options="--branding djmolplatform1 -J-Xms240m -J-Xmx640m"
```

[5] To adjust the **Resolution** of the program Right click on <**djmolplatform164.exe**> then move to <**Compatibility**> tab and click on <**Change high DPI settings**> and Select the option "**Override high DPI scaling behavior**".

[6] Follow the instruction in the **PathVariablesSetting.pdf** file.

DJMol program can be now simply executed by double clicking <**djmolplatform164.exe**> which is located in <**bin**> folder.

Note that for the First Time of the execution, you may want to select, <**Disable Modules and Continue**> option. To fix this, you can try to clean your user-directory as it is mentioned in http://wiki.netbeans.org/FaqWhatIsUserdir

And once the application starts do the following (not mandate but it is useful):

```
Close <Start Page>

Close <Services>

Add <Windows -> Favorites>

Add <Windows -> Output>
```

[B] Compiling Add-On Programs

See <AddOns_compilation.txt> in Add-On directory.