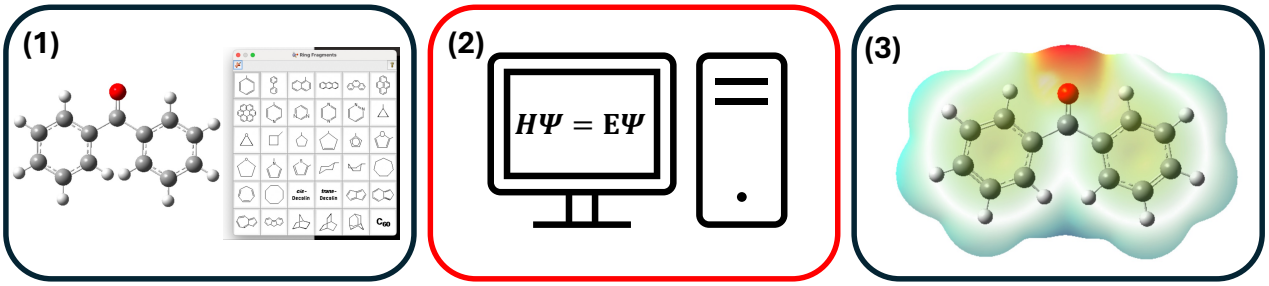
Overview



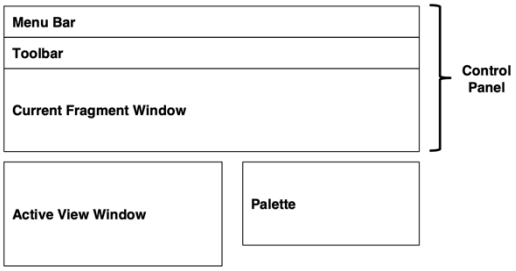
The jobs will consist of three parts: **(1)** Building, **(2)** Calculating, and **(3)** Analysis. Steps **(1)** and **(3)** can be carried out locally, while step **(2)** will be carried out on the remote computing systems. To build molecules and view results we will be using the GaussView program, which the university has a license for. GaussView, which is the GUI for Gaussian, is available on select university computers (such as the chemistry study center on the third floor of Bagley hall), as well as on hyak. Using GaussView on hyak can be done either through the hyak ondemand service, or, if you are able to, through X11 forwarding. Other building programs can be used such as avogadro, pymol, chimerax, etc. if you prefer, or if you are not on campus; though their usage is not addressed in this tutorial. Your lab may have a preferred software, check with them.

In today's tutorial we will run quantum chemical calculations using two common quantum chemistry software packages, Gaussian and Quantum Espresso. Commonly, Gaussian is used for molecular systems and small nanoparticles (<300 atoms), for systems where you are looking at bulk materials, packages such as Quantum Espresso are used. The full capabilities of neither will be fully covered today, but you can look http://www.gaussian.com, and https://www.quantum-espresso.org for additional details.

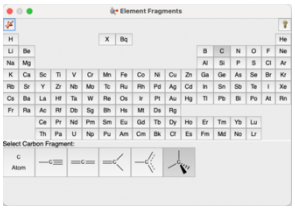
TOC

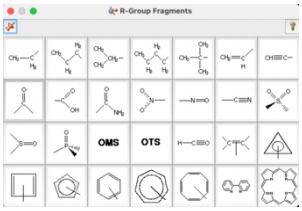
(1a) Using the GaussView Graphical User Interface

Used for setting up and saving Gaussian input files for molecular calculations.



Building Molecules





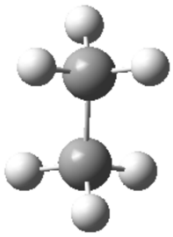
- **Fragments Palettes**
  - Elements
  - R-Groups
  - Rings
  - Biological
  - Select a fragment, and click in Active View Window to place the fragment.
- **Inquire Tool**: for quick measure of bond lengths, angles, and dihedrals
- **Clean**: adjust molecular geometry according to a set of rules designed to match chemical intuition
- **Add/remove valence** (bonds)
- **Center** (View → Center)
- **Modify Bond/Angle/Dihedral** (Sliders to fine-tune geometry)
- **View controls**

Mouse Button	Action	Key	Function
Left	Click		Select atom/place fragment
	Drag up/down		Rotate about x-axis
	Drag left/right		Rotate about y-axis
	Click	Shift	Translate molecule
Scroll	Up/down		Zoom in/out
	Click		Translate molecule
Right	Drag up/down		Zoom in/out
	Drag left/right		Rotate about z-axis
Any Above		Alt	When two or more fragments are not bonded, apply the above functions to only a single fragment.

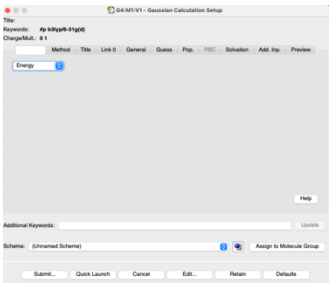
Customizing the View

- File → Preferences (*on Windows*) –OR– GaussView → Preferences (*on Mac*)
  1. Colors → Background color
  2. Display Format → Molecule
  3. Image

Let's setup a job:



- File → New Molecule Group
- Element Fragment → Carbon Tetrahedral
- Click in the Active View Window, then Click on a Hydrogen (you should now have an ethane molecule)
- Calculate → Gaussian Calculation Setup



1. Job Type: (what will be done)
   - Energy: single point energy calculation
   - Optimization: optimize the molecular geometry
   - Frequency: calculate infrared/Raman vibrational absorption spectroscopy
   - **Opt+Freq**: do both a geometry optimization and frequency calculation
   - IRC: intrinsic reaction coordinate (path) calculation
   - Scan: energetic scan over specified molecular degrees of freedom
   - Stability: ensure your wavefunction is the ground-state.
   - NMR: calculate NMR shifts
2. Method: (how the calculation will be carried out; **Ground State**, **DFT**, **B3LYP**, **6-31Gd**)
3. Title: (for your own records)
4. Link0: (computer controls, setup to use all of klone resources)
   - Memory Limit: **115 GB**
   - Shared Processors: **40**
   - Chkpoint File: **{parent file name}.chk** (stores additional job information)
5. General: (additional job controls)
   - Deselect Write Connectivity
   - Select Additional Print
6. Preview: view input file before saving.

The `ethane.gjf` file:

```
%chk=ethane.chk
%nprocshared=40
%mem=115GB
#p opt freq=(raman) b3lyp/6-31g(d)
      --blank--
Title Card Required
      --blank--
0 1
 C                  0.29411765    0.89965397    0.00000000
 H                  0.65077207   -0.10915604    0.00000000
 H                  0.65079049    1.40405216   -0.87365150
 H                 -0.77588235    0.89966715    0.00000000
 C                  0.80745987    1.62561024    1.25740497
 H                  0.45240352    2.63498333    1.25642745
 H                  0.44919076    1.12234181    2.13105486
 H                  1.87745806    1.62390355    1.25838372
      --blank--
```

Note the lines labelled "--blank--" are required blank lines in the input file.

(2a) Running a Gaussian Job

Using computational resources offered by the University of Washington.

Upload files to the computing cluster

- We will need to create a "work directory" where we can place all of our files:
    - `` `ssh $usr@klone.hyak.uw.edu` ``
    - `` `mkdir -p /gscratch/scrubbed/$usr` ``
    - *Note* files in the "`/gscratch/scrubbed/$usr`" directory will be removed after ~24 days of inactivity. Make sure you create local backups!
- Then, we will upload the file:
    - `` `scp ethane.gjf $usr@klone.hyak.uw.edu:/gscratch/scrubbed/$usr/` ``
- The hyak computer cluster uses the slurm scheduler to run jobs, to submit to slurm a submission script needs to be generated. An example follows.
- You can view your currently running (and pending) jobs with `` `squeue -u $usr` ``, which returns some helpful output:
    - The JOBID allows you to cancel jobs and see information about specific jobs.
    - The ACCOUNT is which account the job is submitted to (these will most likely all be memc)
    - The NAME is the name you gave your job
    - The ST is the status, it will either be R (running), PD (pending), or CG (stopped)
    - The TIME is the time that the job has been running
    - NODELIST(REASON) returns either the node(s) assigned to your job, or will have printing for why the job is pending. Usually pending has three reasons:
        a. Resources: Waiting to be assigned resources.
        b. Priority: Waiting for your priority to be high enough to assign resources.
        c. QOSGrpXYZLimit: (XYZ=Cpu/Mem/Gpu/etc.) Where the account you submitted to has allocated the maximum resources of type and your job is waiting for more to become available.

```
[rbeck4@klone-login03 ~]$ squeue -u rbeck4
             JOBID   ACCOUNT     NAME     USER ST       TIME    TIME_LEFT  PRIORITY NODES   NODELIST(REASON)
          18809470       cei CsUOCl_g   rbeck4  R    1:32:20   4-22:27:40  866      3   n[3229-3230,3350]
        18786068_5       cei chir_per   rbeck4  R    3:59:46   6-12:00:14  861      2   n[3050-3051]
          18809646      chem   REU_PH   rbeck4  R    1:01:24   4-14:58:36  808      3   n[3258,3304,3380]
```

    - If you belong to multiple accounts, you can check the usage with `` `hyakalloc` `` and change the account you submit to depending on if any have unused resources.

The `ethane.sh` submission file:

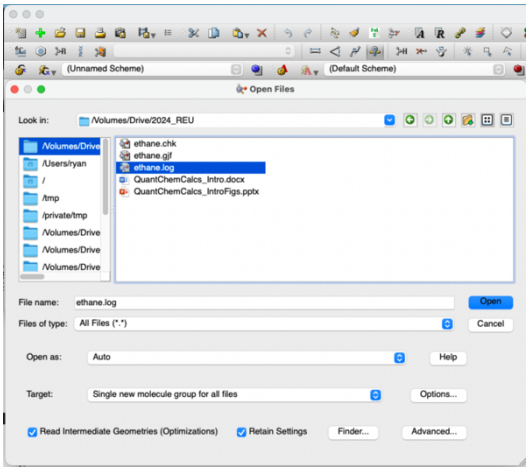
```
#!/bin/bash
#SBATCH --job-name=ethane
#SBATCH --nodes=1
#SBATCH --cpus-per-task=40
#SBATCH --time=1:00:00
#SBATCH --mem=160G
#SBATCH --mail-type=FAIL,END
#SBATCH --mail-user=$usr@uw.edu
#SBATCH --partition=compute
#SBATCH --account=memc

## load Gaussian environment
source /etc/profile
module load chem/g16/b01

export scrDir="/scr/"
mkdir -p $scrDir
export GAUSS_SCRDIR=$scrDir

## Det. number of threads
export num_threads=$(echo $SLURM_JOB_CPUS_PER_NODE| cut -f1 -d"(" )

## Det. Memory
gbmem=`expr $SLURM_MEM_PER_NODE / 1000`
gbmem=`expr $gbmem - 10`
echo "Parsed memory: $gbmem"

for inputfile in ethane.gjf
do

## Set threads and memory:
sed -i "/mem/s/.*/%mem=${gbmem}GB/" $inputfile
sed -i "/nproc/s/.*/%nprocshared=${num_threads}/" $inputfile

## Determine file root name:
if [[ "$inputfile" == *".gjf"* ]]
then
  echo ".GJF INPUT"
  fileNM="$( basename "$inputfile" ".gjf" )"
else
  echo ".COM INPUT"
  fileNM="$( basename "$inputfile" ".com" )"
fi

## run Gaussian
g16 $inputfile
formchk "${fileNM}.chk" "${fileNM}.fchk"
cubegen 4 density=scf ${fileNM}.fchk ${fileNM}_den.cube
cubegen 4 MO=HOMO,LUMO ${fileNM}.fchk ${fileNM}.cube
done
exit 0
```

- We can now submit this script with `sbatch ethane.sh`
- If you have set your uw email on line 8 it will automatically notify you once the job finishes (or if it fails).
- You can see how far into the job Gaussian is by looking at the output file with `less ethane.log`
- A successful Gaussian job will terminate with the printing “Normal termination”.

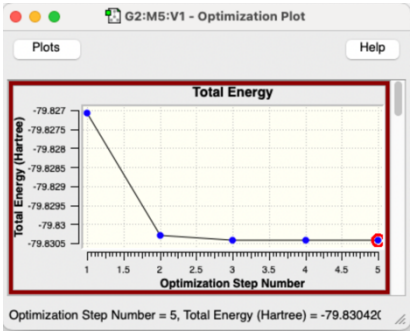
(3a) Examining Results

After the Gaussian job successfully completes, we will be presented with several files, the ones you will most likely want to keep are the `ethane.log` and `ethane.chk`. In our `ethane.sh` file we also generated two `ethane*.cube` files which contain visualizations of the wavefunction.

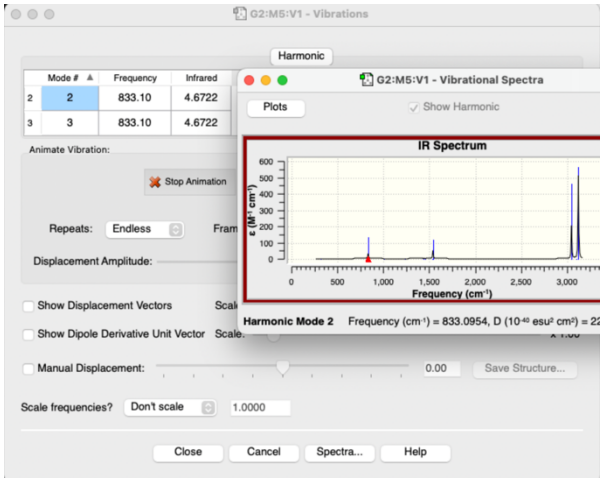
Optimization and Vibrations



- Open the file in Gaussview, make sure that the “Read Intermediate Geometries” option is checked (to view optimization results).
- The system energy as a function of the optimization step can be examined through Results → Optimization.
- Geometric properties can be plotted by selecting the Plots → Plot Molecular Property option.

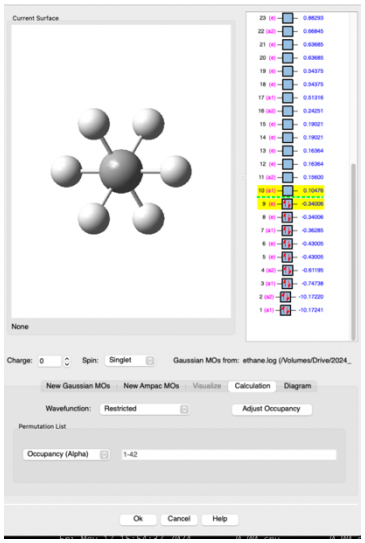


- The normal modes can be viewed by through Results → Vibrations.
  - *Note* that when you run geometry optimizations it is best practice to run a frequency calculation to ensure you have no negative (imaginary) roots and are thus in a geometric minimum.
- Specific normal modes can be visualized by clicking the “Start Animation”
- The infrared (and Raman if computed) spectrum can be visualized by selecting “Spectra”.

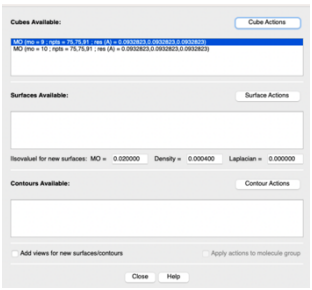


Looking at Molecular Orbitals

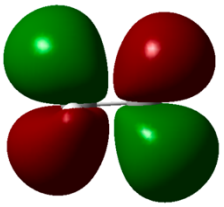
We can visualize the wavefunction, this information is kept in the .chk file (and can be visualized using the .cube files we generated at the end of the job.sh file).



1. With the ethane.log file open, go to Tools → MOs
   - This presents the energy levels of each MO.
   - The highest occupied (HO) and lowest unoccupied (LU) molecular orbital (MO) are closely related to electronic excitations (UV/Vis absorption). The type of transition can be analyzed by looking at the HOMO/LUMO.
   - If you open the ethane.chk file, specific MOs can be visualized in this window instead of through the .cube file.
2. Using GaussView open the `ethane.cube` file



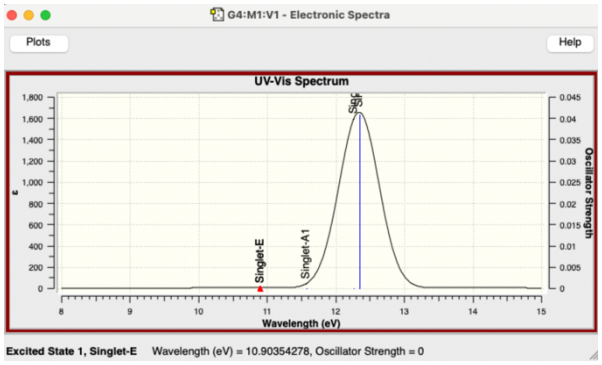
3. Go to Results → Surfaces/Contours
   - We have two available surfaces (MOs 9 and 10, corresponding to the HOMO and LUMO in the MO viewer).
   - Select Surface Actions → New Surface and you can visualize the HOMO.
   - You can hide this MO by selecting Surface Actions → Hide Surface.
   - To visualize the LUMO, select MO 10 under available cubes, then Surface Actions → New Surface.



4. Looking at the HOMO and LUMO we can see the π^* to σ^* transition.

UV/Vis

The UV/Vis absorption spectrum for a molecule can be computed using Linear Response TDDFT (the td keyword).



- Create another input file using the geometry of the optimized ethane molecule.
- Under “Job Type” we will do “Energy”, and under “Methods” we will do “TD-SCF”
- Once the job has completed we can open the .log file in gaussview.
- The predicted UV/Vis spectrum is located under Results → UV-Vis.
  - By default GaussView will plot the spectrum in wavelength, by right-clicking the spectrum and going to “Properties” you can change the units.
  - A gaussian broadening of 0.33 eV is automagically applied to the sticks, the fwhm value can be changed in “Properties” as well.

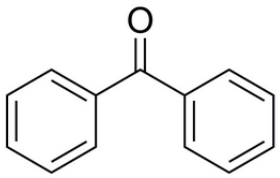
- To see the type of transition, we can look at the .log file for the "Excited State" printing. We can see in the below image the first transition is from the HOMO → LUMO (which we've already visualized) with the second transition being from the HOMO-1 → LUMO.

```
447  Excited State   1:      Singlet-E     10.9033 eV  113.71 nm  f=0.0000  <S**2>=0.000
448        9 -> 10         0.70497
449  This state for optimization and/or second-order correction.
450  Total Energy, E(TD-HF/TD-DFT) =  -79.4297322635
451  Copying the excited state density for this state as the 1-particle RhoCI density.
452
453  Excited State   2:      Singlet-E     10.9033 eV  113.71 nm  f=0.0000  <S**2>=0.000
454        8 -> 10         0.70497
```

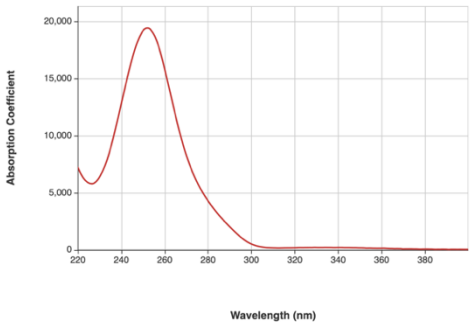
  - Some transitions may involve more than one MO pair, in these cases the 'weight' value (0.70497 in states 1 and 2 above) gives an indication of the most significant transition.
  - In cases where there are several MO transitions of near-equal weight; transforming the MOs into natural transition orbitals may be helpful (https://gaussian.com/faq4/).

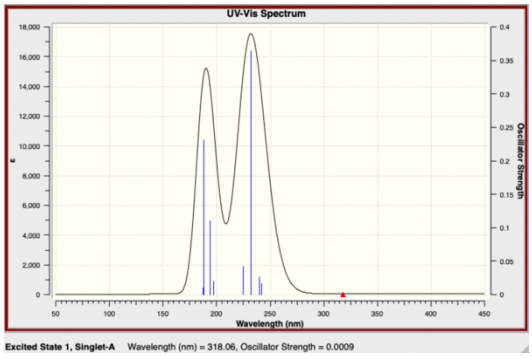
Another Example

The photochemCAD database (https://www.photochemcad.com) contains several small, organic molecules with their UV/Vis absorption information. Let's examine A36, benzophenone (https://www.photochemcad.com/databases/common-compounds/aromatic-hydrocarbons/benzophenone):



At a cam-b3lyp/6-311G(D) level of theory, the first optimization fails (negative frequencies). Rerunning with "opt=readfc", we can observe the first UV-Vis response at 318 nm, comparing against the experimental spectrum there is not a strong response in this region. Looking at the value for the oscillator strength we see that Gaussian also predicts that we will not see an absorption peak here. The first response we expect to see a response for is at 242 nm; which matches decently with the experimental database showing a peak at 252 nm. Allowing the peaks to broaden we see that we are similar to the experimental line shape.

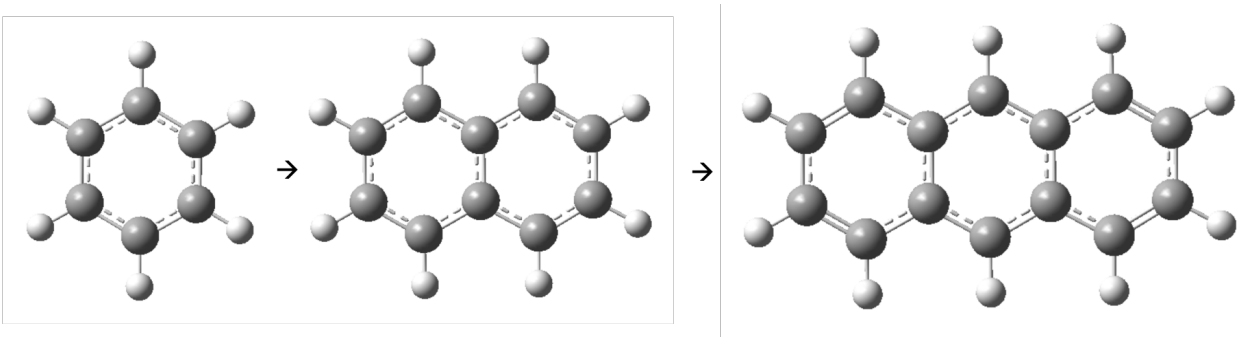




Note choice of solvent, basis set, and functional can all influence peak position; along with any additional possible atomic configurations.

Size Dependence

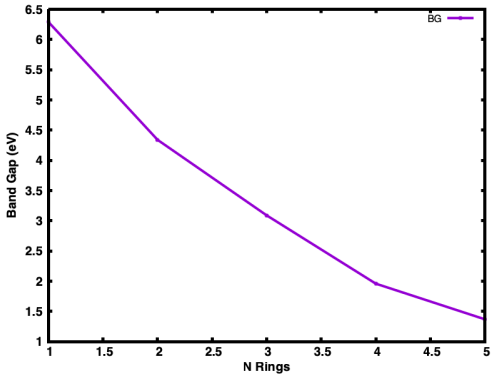
The position of the absorption peak changes inversely with respect to system size. We can look at this though the N-cene series starting with benzene and increasing the number of rings. Given the close relationship between the first excitation energy and the band gap (LUMO-HOMO), we can save time by just looking at band gaps:



Plotting the band gap as a function of the number of rings at a hseh1pbe (known in literature as HSE06)/6-311G(D) level of theory for rings NRing = 1 →5 we get:

```
..$ grep -m1 -A1 "virt" *cene.log
1cene.log- Alpha  occ. eigenvalues --   -0.24749
1cene.log: Alpha virt. eigenvalues --   -0.01632  -0.01039   0.04353   0.07816   0.07963
--
2cene.log- Alpha  occ. eigenvalues --   -0.32824  -0.28724  -0.24565  -0.21247
2cene.log: Alpha virt. eigenvalues --   -0.05328  -0.01999   0.01440   0.04665   0.06041
--
3cene.log- Alpha  occ. eigenvalues --   -0.24422  -0.19086
3cene.log: Alpha virt. eigenvalues --   -0.07715  -0.02180  -0.00604   0.01980   0.04847
--
4cene.log- Alpha  occ. eigenvalues --   -0.28864  -0.27666  -0.24545  -0.23041  -0.17203
4cene.log: Alpha virt. eigenvalues --   -0.09997  -0.03680  -0.02411   0.00499   0.01845
--
5cene.log- Alpha  occ. eigenvalues --   -0.24795  -0.21255  -0.16289
5cene.log: Alpha virt. eigenvalues --   -0.11246  -0.05786  -0.02490  -0.01318   0.01156
```

N Rings	Band Gap (eV)*
1	6.29
2	4.34
3	3.09
4	1.96
5	1.37

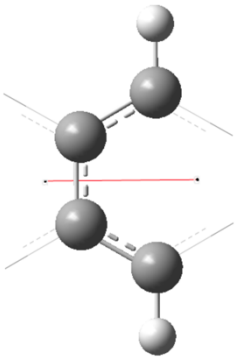


Note conversion between the reported Hartree atomic units and electron Volts is 27.211 eV/Har.

What if we want to go to N>5?

Translation Vector

Given the translational symmetry of the NCene series, we can represent a continuous series by specifying a unit cell. A one-dimensional (line) system can be seen to the right, where the repetition is along one axis (vector shown in red). To examine and modify these translation vectors GaussView offers several options under Tools → PBC.



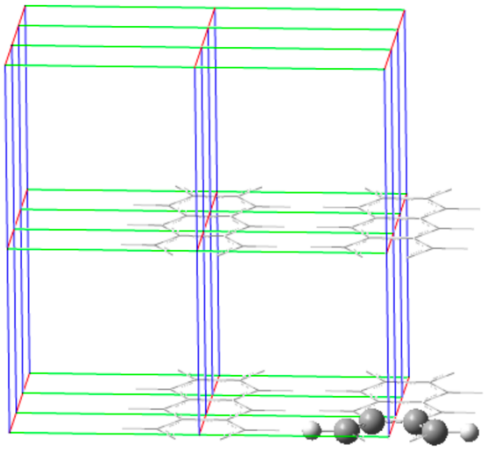
The `NCene.gjf` file:

```
%chk=NCene.chk
%nprocshared=40
%mem=50
#p pbepbe 6-311G(D) auto

Title Card Required

0 1
 C                  1.86810380    0.69746953    0.00101870
 C                  0.65979215    1.39493233    0.00120027
 C                  0.66002631   -1.39501201   -0.00183300
 C                  1.86820170   -0.69735888   -0.00034086
 H                  0.66005394    2.49458605    0.00304949
 H                  0.66041222   -2.49477192   -0.00332019
 Tv                 2.43516999    0.00000000    0.00000000
```

Note the single "Tv" line above, this is the 1D translation vector. We can increase the dimensionality by adding two extra Tv vectors to the end of the geometry block:

```
 Tv                 0.00000000   12.00000000    0.00000000
 Tv                 0.00000000    0.00000000   12.00000000
```

This creates the 3D unit cell shown on the right, where each of the 'balls' are defined atoms, and all of the 'lines' are the periodic images.



While Gaussian is able to do some Periodic Boundary Condition jobs, different software packages are commonly used. Two of the more common ones are the Vienna Ab initio Simulation Package (VASP) and Quantum Espresso. We will do a QE example to demonstrate running PBC jobs on hyak.

(1b) Setting Up Quantum Espresso:

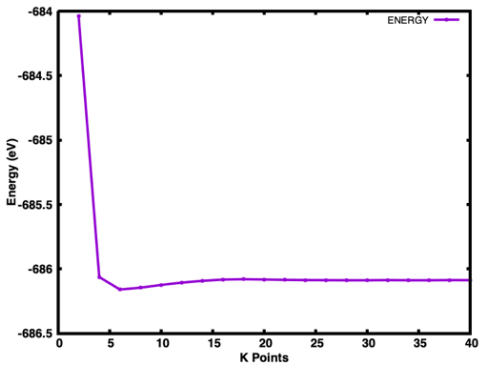
Quantum Espresso and Gaussian work in different basis representations. While we have been working in a GTO (gaussian type orbital) basis, many PBC calculations are carried out in a plain wave basis. A full discussion and comparison between the two is outside the scope of this document, however this means there are different steps we need to take when setting up QE jobs. QE does have a GUI, which is capable of viewing and modifying input files (XCrysDen), however the manipulation of this is out of scope.

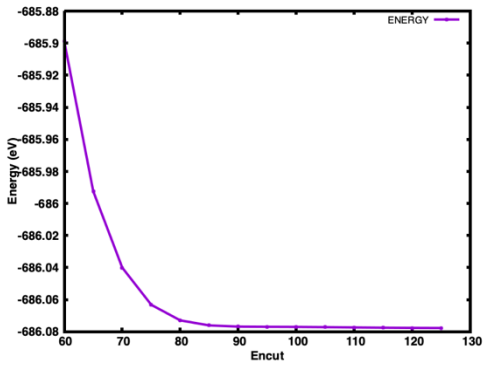
- Unlike in the Gaussian job, QE only represents the valence electrons explicitly, as such a pseudo potential must be used for the core electrons. QE is able to read potential files in the .upf format. I recommend (http://www.pseudo-dojo.org/) for already-generated potential files.
- Where the Gaussian job had the 'route' section (the `#p` area of the input file), QE has several keywords and values to define. For all options and usage visit: https://www.quantum-espresso.org/Doc/INPUT_PW.html

The `01.SCF.in` file:

```
&control
  calculation = 'scf'
  prefix = 'ncene'
  pseudo_dir = './'
  verbosity = 'high'
/
&system
  ibrav = 0
  nat = 6
  ntyp = 2
  ecutwfc = 125
  nbnd = 18
  occupations = 'smearing'
  smearing = 'gaussian'
  degauss = 0.01
/
&electrons
/
ATOMIC_SPECIES
  C 12.01 C.upf
  H 1.008 H.upf
CELL_PARAMETERS {angstrom}
    2.43516999    0.00000000    0.00000000
    0.00000000   12.00000000    0.00000000
    0.00000000    0.00000000   12.00000000
ATOMIC_POSITIONS {angstrom}
 C  1.86810380    0.69746953    0.00101870
 C  0.65979215    1.39493233    0.00120027
 C  0.66002631   -1.39501201   -0.00183300
 C  1.86820170   -0.69735888   -0.00034086
 H  0.66005394    2.49458605    0.00304949
 H  0.66041222   -2.49477192   -0.00332019
K_POINTS {automatic}
  10 5 5 0 0 0
```

Note by default the QE will only calculate information for the occupied bands, if you desire information about the virtual space/band gap then you will want to increase the nbnd value to nbnd > (# e^- / 2)
Note for PBC jobs the k-point grid, and ecutwfc values should be checked to ensure they are large enough to describe your system:





- The scf job type will perform a single-point calculation (as we did for the N-Cene series above). Two other common job types are:
  - bands: after doing an scf calculation compute the information to get the band structure for a system.
  - vc-relax: optimize both the atomic positions, and the unit cell vectors (relax will only do ion positions)
- The pseudo potentials are defined under "ATOMIC SPECIES". You can have multiple similar atoms (*e.g.* C1, C2) if you have magnetic centers, just be sure to update the ntyp value.

(2b) Running Quantum Espresso:

In the above ncene.in file, the pseudo_dir variable tells quantum espresso where to locate the pseudopotential files. Above we have it set to the current directory, as such we will need to have:

- `ncene.in`
- `run.sh`
- `C.upf` and `H.upf`

All in the directory where we are submitting the job. The pseudopotential `*.upf` files must be either self-generated, or previously generated potentials can be obtained through sites such as pseudo-dojo.org.

Quantum espresso is able to easily take advantage of mpi (inter-node parallelization). In addition, intra-node parallelization can be used by setting the environment variable OMP_NUM_THREADS. The following file will run on one node, 10 instances, each using 4 openmp threads.

The `QEsub.sh` file:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=40
#SBATCH --time=100:00:00
#SBATCH --job-name=REU_QE
#SBATCH --mem=0
#SBATCH --mail-type=FAIL,END
#SBATCH --mail-user=$usr@uw.edu

#SBATCH --partition=compute
#SBATCH --account=memc

source /etc/profile
module load chem/qe

export OMP_NUM_THREADS=4
procs=10

#JOB
mpirun -np $procs pw.x -i 01.SCF.in >01.SCF.out

exit 0
```

(3b) Examining Results

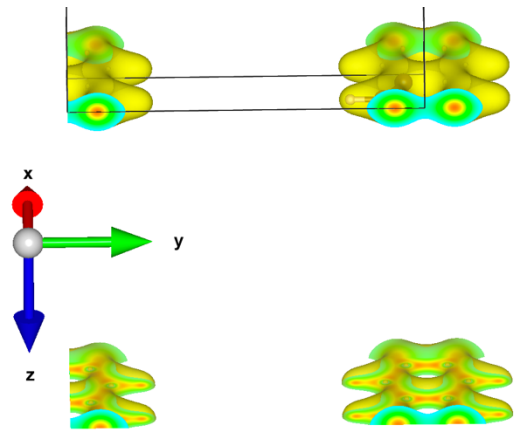
From the SCF job we can obtain the energy of the system (helpful for comparing configurations), and even the band gap or fermi level (since we have defined additional bands) depending if we have a conductor or insulator.

```
6887      the Fermi energy is    -2.7292 ev
6888
6889 !    total energy              =     -50.49958868 Ry
6890      estimated scf accuracy    <       0.00000011 Ry
6891      smearing contrib. (-TS)   =      -0.00000258 Ry
6892      internal energy E=F+TS    =     -50.49958610 Ry
6893
6894      The total energy is F=E-TS. E is the sum of the following terms:
6895      one-electron contribution =    -234.77474156 Ry
6896      hartree contribution      =     120.13884999 Ry
6897      xc contribution           =     -18.56852006 Ry
6898      ewald contribution        =      82.70482553 Ry
6899
6900      convergence has been achieved in   8 iterations
```

- The final, converged SCF energy is printed with a leading "!"
- The fermi energy is given (top of highest occupied band), depending on the system it may also return "highest occupied, lowest unoccupied" printing.
- Total charge (and spin, if applicable) density can be plotted using the pp.x program (`pp.x <charge_pp.in>charge_pp.out`)

The `charge_pp.in` file:

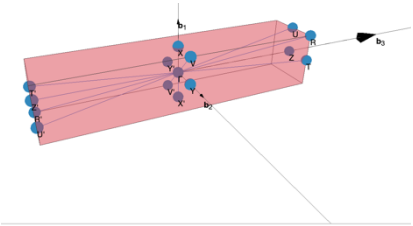
```
&INPUTPP
  prefix = 'ncene'
  filplot = 'ncene.chg'
  plot_num = 0
/
&PLOT
  filepp(1) = 'ncene.chg'
  iflag = 3
  output_format = 6
  fileout = 'ncene_charge.cube'
/
```

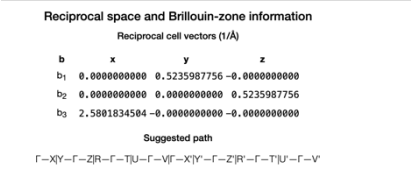


Note The returned ncene_charge.cube file will be periodic, unlike in the earlier case with the MOs, as such a program such as VESTA (above image) or VMD (or others) will be required to visualize the surface.

Electronic Structure

In the above job, we assigned a k-mesh and allowed QE to distribute the points as it wished. This allowed us to view the overall band gap, and obtain the charge density, however if we want to obtain a deeper understanding of the electronic structure (similar to examining the MOs previously) we will need to specify specific paths through our Brillouin zone.





- Since we have already obtained a converged wavefunction (01.SCF.out) we do not need to calculate one again. The jobtype we will do is 'bands':
- We still have to pick a path, however. A helpful tool to do so is the MaterialsCloud's seekpath tool (tools.materialscloud.org/seekpath/)
    - Upload the 01.SCF.in file, select "QE" under the file format, then click "Calculate My Structure"
    - This will return an interactive representation of the Brillouin zone, as well as a recommended path through it.
    - At the bottom of the page there is the ability to copy and paste the recommended path (QE pw.x input). You will have to fill in the prefix information, but the ATOMIC POSITIONS, K_POINTS, and CELL_PARAMETERS can all be directly coped out.

The `02.NSCF.in` file (prefix):

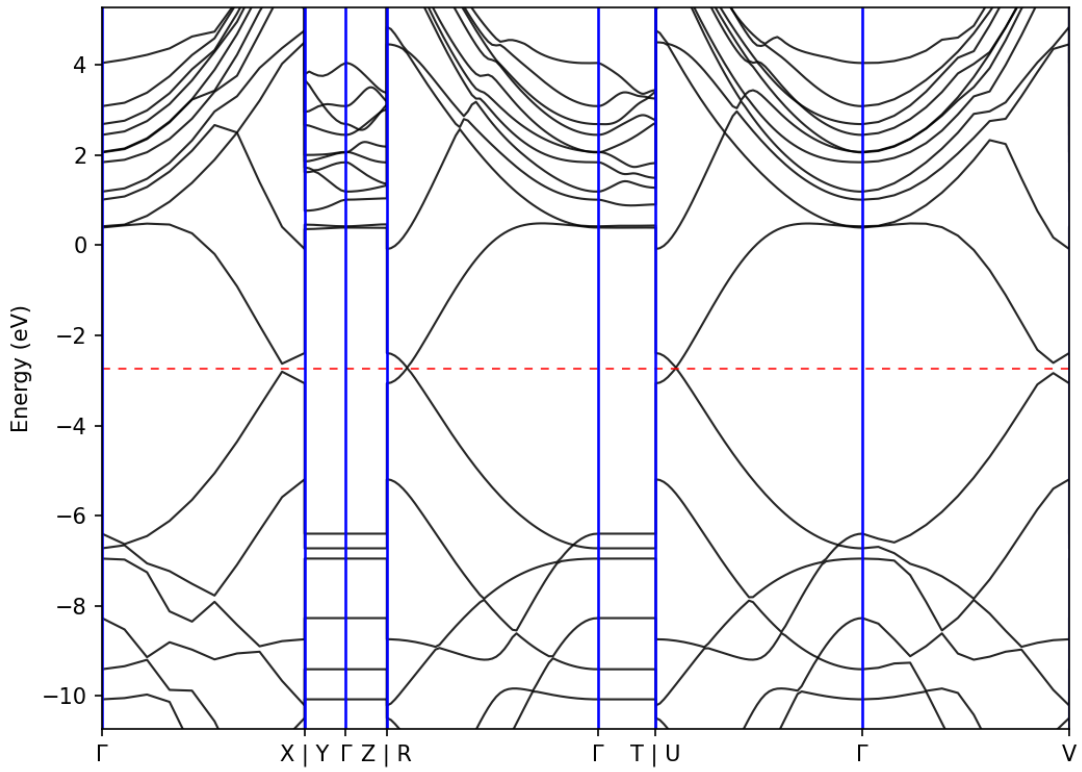
```
&control
  calculation = 'bands'
  prefix = 'ncene'
  pseudo_dir = './'
  verbosity = 'high'
  occupations = 'smearing'
  smearing = 'gaussian'
  degauss = 0.01
/
&system
  ibrav = 0
  nat = 6
  ntyp = 2
  ecutwfc = 125
  nbnd = 20
/
&electrons
/
ATOMIC_SPECIES
[…]
```

- After running the 'bands' calculation we need to process the information, QE does this using the bands.x program.

The `03.BANDS.in` file:

```
&BANDS
  prefix = 'ncene'
  outdir = './'
  filband = 'ncene_bands.dat'
/
```

- This can be run with `` `bands.x <03.BANDS.in>03.BANDS.out` ``
- The formatted output ncene_bands.dat.gnu can be read in to plot a band diagram:

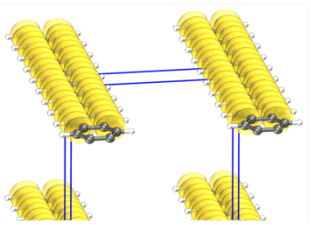


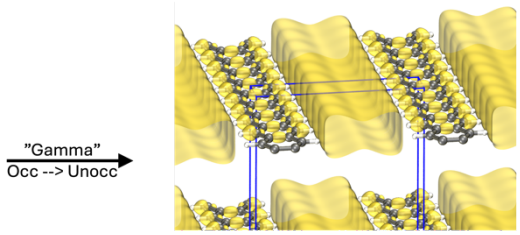
The `plotBands.py` file:

```
import matplotlib.pyplot as plt
from matplotlib import rcParamsDefault
import numpy as np

plt.rcParams["figure.dpi"]=150
plt.rcParams["figure.facecolor"]="white"
plt.rcParams["figure.figsize"]=(8, 6)

# load data
data = np.loadtxt('./ncene_bands.dat.gnu')
numk = int(len(data[:,0]) / np.sum(data[0,0] == data[:,0]))
k = data[:numk,0]
bands = np.reshape(data[:, 1], (-1, numk))

for band in range(len(bands)):
    plt.plot(k, bands[band, :], linewidth=1, alpha=0.85, color='k')

# Fermi energy (01.SCF.out)
fermi = -2.7292
plt.axhline(fermi, linestyle=(0, (5, 5)), linewidth=0.75, color='r')

# High symmetry k-points (check 03.BANDS.out)
# Note that we have to remove repeat values, not interested in time
reversal
xpos = [0.0000, 0.5000, 0.6015, 0.7029, 1.2231, 1.3666, 1.8768, 2.3870]

# High symmetry lables (materialscloud.org/seekpath/)
lbls = [ "$\Gamma$", "X | Y", "$\Gamma$", "Z | R", "$\Gamma$", "T | U", \
         "$\Gamma$", "V" ]

# Mark High Symm points (03.BANDS.out)
for point in xpos:
  plt.axvline(point, linewidth=1.25, color='b')

# Label High-symmetry points (seekpath)
plt.xticks(ticks=xpos, labels=lbls)

# Formatting
plt.xlim(min(xpos), max(xpos)) #Show range we are interested in
plt.ylim([fermi-8, fermi+8]) #Interested in valence space
plt.ylabel("Energy (eV)")

plt.savefig("plotBand.png")
```

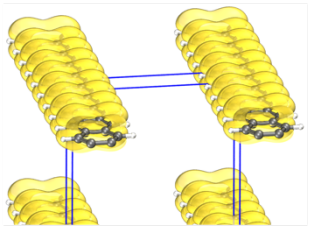
- The number of symmetry points that we actually need are much less than we have above since we are in only one dimension.
- Notable in the above band plot is the presence of a Dirac cone, which we expect to see from a graphene-like system.
- Similar to how we plotted out individual MOs earlier, we can plot the partial charge contributions for select bands at specific k-points using pp.x.

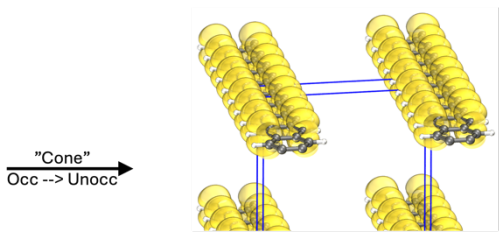
- I am interested in looking at the gamma (1) point and near one of the cones (75). Our system has 18 electrons so bands 9 and 10 correspond to the highest and lowest occupied, respectively.






```
2583           k = 0.0000 0.0000 0.0000 ( 55889 PWs)   bands (ev):
2584
2585    -21.4506 -18.6989 -14.2139 -10.0727  -9.4056  -8.2713  -6.9524  -6.7237
2586     -6.3997   0.3889   0.4169   1.0078   1.1860   1.8382   2.0544   2.0671
2587      2.4446   2.6832   3.0812   4.0392
```






```
3033           k = 0.4519 0.0917 0.0917 ( 55906 PWs)   bands (ev):
3034
3035    -17.9300 -16.1725 -15.3245 -13.7364 -10.9295  -9.7515  -8.7720  -5.5021
3036     -2.7262  -2.7162   0.2946   4.2575   4.3045   5.7310   5.8106   6.3294
3037      6.4682   6.5072   7.2351   7.7517
```

The `bands_pp.sh` file:

```
#!/bin/bash

for i in 1 75
do
  for j in 9 10
  do

    pp.x<<IN
&INPUTPP
  prefix = 'ncene'
  filplot = 'ncene.chg'
  plot_num = 7
  kpoint(1)=$i
  kband(1)=$j
/
&PLOT
  filepp(1) = 'ncene.chg'
  iflag = 3
  output_format = 6
  fileout = "ncene_band_k${i}_b${j}.cube"
/
IN

  done
done
```

Note in the above case we are iterating over 4 values, pp.x does support specifying kpoint(i) i=1,2; however, this will return a range of bands and kpoints from the one specified at 1 to 2.

Geometry Optimization

Oftentimes the utilized unit- and super-cells must be allowed to relax. A simple relaxation job input follows:

The `04_OPT.in` file:

```
&control
  calculation = 'vc-relax'
  prefix = 'ncene'
  pseudo_dir = './'
/
&system
  ibrav = 0
  nat = 6
  ntyp = 2
  ecutwfc = 125
  occupations = 'smearing'
  smearing = 'gaussian'
  degauss = 0.01
/
&electrons
/
&IONS
  ion_dynamics = 'bfgs'
/
&CELL
  cell_dynamics = 'bfgs'
/
ATOMIC_SPECIES
  C 12.01 C.upf
  H 1.008 H.upf
CELL_PARAMETERS {angstrom}
    2.43516999    0.00000000    0.00000000
    0.00000000   12.00000000    0.00000000
    0.00000000    0.00000000   12.00000000
ATOMIC_POSITIONS {angstrom}
 C  1.86810380    0.69746953    0.00101870
 C  0.65979215    1.39493233    0.00120027
 C  0.66002631   -1.39501201   -0.00183300
 C  1.86820170   -0.69735888   -0.00034086
 H  0.66005394    2.49458605    0.00304949
 H  0.66041222   -2.49477192   -0.00332019
K_POINTS {automatic}
  10 10 10 0 0 0
```

Note The 'vc-relax' jobtype will relax both atomic positions and the cell (and needs both IONS and CELL sections), to only optimize atomic positions the 'relax' jobtype can be used (and only IONS section is required).

- To run this job, the same `QE.sub` file can be used, just replacing the `01.SCF.in` (and out) file with `04.OPT.in` (and out).

To examine the optimization results of the QE job, we can either open the `04.OPT.out` file in xcrysden, or translate the structure to a version readable by GaussView:

Note another common program to look at .cif and generate files, especially for VASP is VESTA

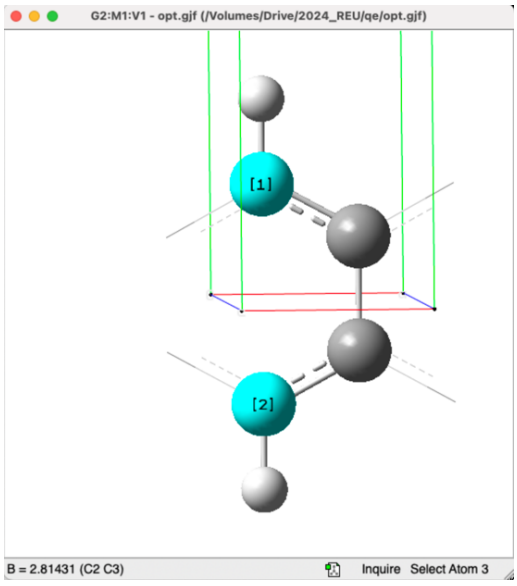
```
1532 Begin final coordinates
1533      new unit-cell volume =   2397.80960 a.u.^3 (   355.31872 Ang^3 )
1534      density =      0.23393 g/cm^3
1535
1536 CELL_PARAMETERS (angstrom)
1537    2.461842968   0.000051570   0.000001077
1538    0.000253664  12.027233738   0.000023217
1539    0.000005270   0.000023222  12.000297010
1540
1541 ATOMIC_POSITIONS (angstrom)
1542 C            1.8949310562        0.7314837301        0.0009243423
1543 C            0.6640007637        1.4071421781        0.0017113060
1544 C            0.6640552609       -1.4071667353       -0.0016883952
1545 C            1.8949558802       -0.7314562700       -0.0008370459
1546 H            0.6641897828        2.4988423509        0.0028783859
1547 H            0.6643016350       -2.4988654687       -0.0032113700
1548 End final coordinates
```

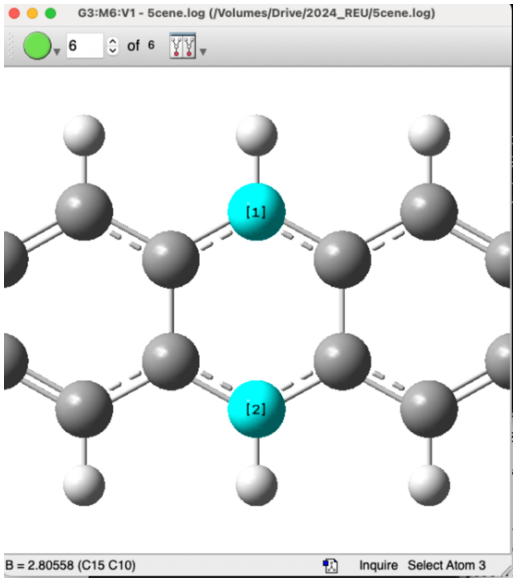
To do so, we can take the finalized geometry from the QE job and create a .gjf file:

- The final cell printing (both ionic positions and cell coordinates) is located after the "Begin final coordinates" printing in the .out file
- The Atomic positions can be directly copied in, however the before each of the vectors in CELL_PARAMETERS we will need to specify 'Tv' to tell GaussView it is a PBC file.

```
1 #p hf/sto-3g
2
3 Title
4
5 0 1
6  C            1.8949310562        0.7314837301        0.0009243423
7  C            0.6640007637        1.4071421781        0.0017113060
8  C            0.6640552609       -1.4071667353       -0.0016883952
9  C            1.8949558802       -0.7314562700       -0.0008370459
10 H            0.6641897828        2.4988423509        0.0028783859
11 H            0.6643016350       -2.4988654687       -0.0032113700
12 Tv  2.461842968   0.000051570   0.000001077
13 Tv  0.000253664  12.027233738   0.000023217
14 Tv  0.000005270   0.000023222  12.000297010
15
```

- Comparing the before and after .gjf files we can see that the rings have expanded slightly (bond C2-C3 2.79 Å → 2.81 Å)
- If we optimize the `*cene.gjf` structures from before we see a similar expansion of the rings.





While it is possible to calculate the vibrational absorption spectra for periodic systems, this is non-trivial and outside the scope of this tutorial.