Other Guide

version 0.1

Last generated: January 26, 2020

Table of Contents

Other Programs and Lab Skills

This guide contains information about other lab skills and programs, including stuff about reference managers, printing cloth posters, and using CAVER, Gaussian, and LICHEM. In other words, this is the random guide.

Creating Cloth Posters

Cloth research posters are more portable, durable, and substantially cheaper than traditional paper posters. Have you ever had to bring a poster tube on a plane? It's a nightmarish task.

You can use Spoonflower (page 0) to print cloth posters. The process that a few people have figured out work best for these are:

- Submit a poster with 300 DPI (dots per inch/pixels per inch) resolution
- Converting from a PDF to a PNG (a TIF, GIF, or *minimally compressed* JPG works too)
- Using the **Performance Piqué** fabric
- Using the **Yards (56" width)** size
- Use a **Center** repeat

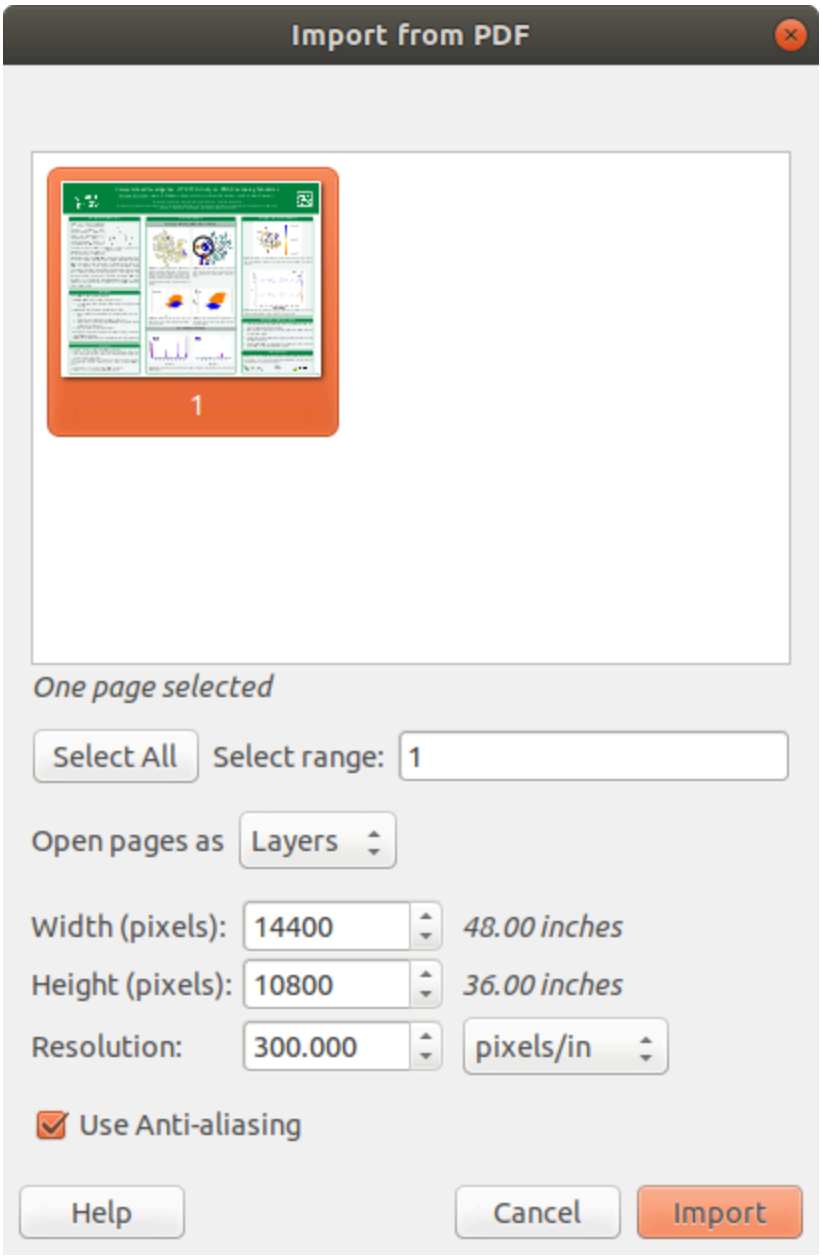
Posters generated using LaTeX are already PDFs, and posters created from PowerPoint can be exported as a PDF. From a saved PDF, these files can then be converted to a higher-resolution image file using Gimp (free) or another photo-editing software, such as Photoshop®.

Using Gimp to Convert a PDF to 300 DPI

You can open the file from the command line with:

```
gimp name_of_file.pdf
```

This will open the Import from PDF window. In this window, you can change the resolution from the default 100 pixels per inch to the wanted 300 pixels per inch. You want to do this so your poster is a higher-quality, and the text and images aren't grainy or pixellated.



The PDF resolution should be changed from the default 100 pixels/inch to 300 pixels/inch before saving in an image format.

Once the PDF is imported, you can now export it as in image format. To do this, follow `File → Export As`. I personally recommend saving as a PNG, but any of the accepted formats is okay. Make sure to explicitly change the file extension and the output file type.

Now that you've exported the image (it might be a good idea to open it up and check that everything looks right), you can upload it to Spoonflower (page 0).

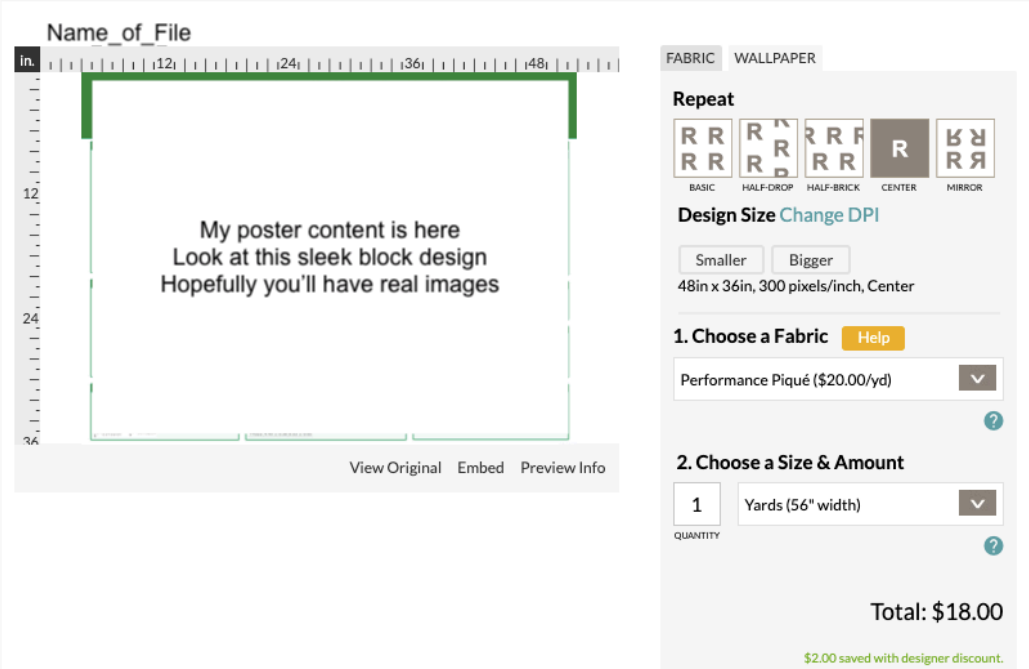
Spoonflower Uploads

It is recommended that you make an account with Spoonflower for a few reasons:

- You can sometimes get discount codes
- Your past work will be saved
- It's easier for future you

Even if you don't make an account, you can still use Spoonflower to print your cloth poster. The top left corner has an "Upload Your Design" button. Click that button! ?

From there, you upload your image.



The Spoonflower upload window.

Once it is uploaded, select your fabric and click the "smaller" button until the proper poster size (likely 48x36) is shown. Choose the "Yards (56" width)" size.

Then proceed to checkout or upload more posters! Hooray! You did it!

> ⚠ **Important:** Make sure you have your poster submitted at least 2 weeks before you need it! Shipping (even rush) takes time, and things can get lost in

the mail.

Reference Managers

A reference manager pulls the information for citations (or references) from articles, chapters, books, etc. that you read. When you read the “perfect paper” three months and 100 articles ago, they can save you time and agony. Most reference managers also allow you to save PDFs and annotate them for future use, as well as organize references into folders or tagged groups. Which reference manager you use is a personal choice, but the “Big Three” are Endnote, Mendeley, and Zotero.

Often, you’ll download a web plugin for the manager you like to use, which will autopopulate specific citation fields for articles that you select. That said, it is recommended that when you enter an article/chapter/whatever into your reference manager you take the extra time to check the autopopulated information. Sometimes they’ll do funky things, like take the title as the journal name, which can make searching for it later a hassle.

- Zotero
    - Open source
    - Lots of plugins
- Mendeley (page 0)
    - Owned by Elsevier (take that as you will)
    - Lots of plugins (but doesn’t play well with others)
- Endnote ($$$)
    - Costs money
    - Lots of plugins (works well with Word)
- Quasi-Endnote
    - Requires you to download the citation files from every article you care about and upload them to a website
    - Good for collaborating with people that use Endnote to format their work
- RefWorks
    - Schools tend to have licenses
- JabRef
    - Open source
    - Good for converting between file formats

- Paperpile ($)
    - Cheap
    - Works with Google Docs
- Qiqqa
    - 2 GB free storage
    - Not for Macs or Linux without a mirror

Finding References

Now that you know where to store references, let's talk about how to find them. There's several great places to search for literature.

The one that's probably the easiest to use, based on familiarity, is Google Scholar . Google Scholar can also search through patents and government documents in specific year ranges.

For chemistry, there's also products from the Chemical Abstracts Service (CAS). SciFinder and SciFinder^{n} are great search engines for the field.

Pre-printed articles (available before the peer-review process and full acceptance to a journal) are submitted to field-based archives.

- arXiv : physics, math, computer science, engineering, economics, etc.
- ChemRxiv : chemistry
- BioRxiv : biology, clinical trials, neuroscience, etc.
- PsyArXiv : psychological science

Some others include:

- Web of Science
- ScienceDirect
- PubMed

Also, don't neglect your local or university library. These have access to many publications, and librarians are fantastic at helping you search for technical information. Most often subscribe to their own search engines, like EBSCOhost and JSTOR.

Paywalled Articles

Published articles are not free to access. That can be very sad.

Unpaywall is a web extension that will search for freely accessible, legal copies of articles hidden behind a paywall. Several government grants require freely accessible copies *somewhere*, but that often is not on the publisher's website.

If Unpaywall can't find it, you're not out of luck. Next, search your library. Libraries have access to a large number of texts through "Inter-Library Loans" (ILL). It may take a few weeks, but you can often get the publications you need.

Authors are often allowed to distribute their work that has been published. So, you're encouraged to email the corresponding author (you may need to do some digging for their current email if it was published several years ago) asking for a copy of that publication. They're not obligated to respond, but most people are thrilled to be asked because (a) you're interested in their work (!!!) and (b) they might get a citation out of it.

```
Subject: Published Work Inquiry
Hello Professor X,

I am a [high school teacher, student, researcher, person who th
ought the title
was cool] and I stumbled upon your work "Insert Title of Work H
ere" in
[location of work in italics here].

I do not have access to this publication and was wondering if y
ou would be able
to send me a copy.

[Perhaps insert a few sentences of context about why this text
might be helpful
(e.g., I do research with these kinds of solar cells).]

Sincerely,

Person Y
```

Helpful LaTeX Packages for Chemistry

LaTeX Overview

LaTeX is a system for preparing documents, slideshows, posters (and much, much more) using high-quality typesetting. It's command-based, as opposed to layer upon layer of button pressing with common word processors. One of the advantages to using LaTeX for STEM projects is that it has a powerful and robust math rendering capabilities.

Overleaf is an online LaTeX editor that allows for collaboration. It's effectively the Google Docs version of Overleaf. The free version offers a lot, but has some limitations in terms of saving document history and compile time. Unless you have in-depth version control needs, this shouldn't be an issue. It has many tutorials for getting started with LaTeX. Another helpful resource for getting started is the LaTeX Wikibook .

If you have a specific question about a LaTeX functionality, a quick Google search will likely pull up about twenty related questions (with solutions!) on Stack Exchange . That website is incredible.

Helpful Formatting Commands

There are a few formatting commands that I never remember, but I use in almost every document.

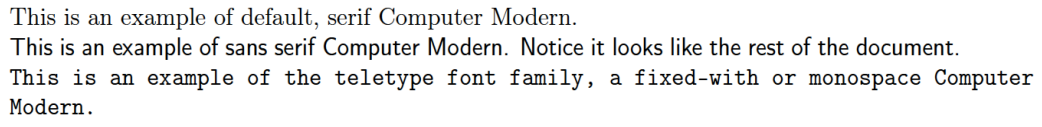
Changing font

The default font for most LaTeX classes is Computer Modern; Beamer's default is sans serif Computer Modern.

To change to sans serif Computer Modern, ad the following to your preamble:

```
%Change to sans serif font
\renewcommand{\familydefault}{\sfdefault}
```

Note: I'd recommend using this with the `sansmathfonts` package (page 17).



Font family examples.

The lines in the image were generated through:

```
\noindent{\textrm{This is an example of default, serif Compute
r Modern.}} \\
{\textsf{This is an example of sans serif Computer Modern. Noti
ce it looks
like the rest of the document.}} \\
{\texttt{This is an example of the teletype font family, a fixe
d-with or
monospace Computer Modern.}} \\
```

Font can also be changed in different ways, making them bold or italicized.

LaTeX Command	Result
`\textnormal{This is a normal example.}`	This is a normal example.
`\emph{This is an italicized example.}`	*This is an italicized example.*
`\textup{This is an upright shape example.}`	This is an upright shape example.
`\textit{This is an italicized example.}`	*This is an italicized example.*
`\textsl{This is a slanted shape example.}`	*This is a slanted shape example.*
`\textsc{This is a small capitals example.}`	THIS IS A SMALL CAPITALS EXAMPLE.
`\uppercase{This is an uppercase example.}`	THIS IS AN UPPERCASE EXAMPLE.
`\lowercase{THIS IS A LOWERCASE EXAMPLE.}`	this is a lowercase example.
`\textbf{This is a bold example.}`	**This is a bold example.**
`\textmd{This is a medium weight example.}`	This is a medium weight example.
`\textsuperscript{A superscript example.}`	A superscript example.
`\textsubscript{A subscript example.}`	A subscript example.
`\hl{Loading the \textbf{soul}package allows highlighting.}`	Loading the **soul** package allows highlighting.
`\textcolor{magenta}{A colored text example.}`	A colored text example.

Results of commands impacting fonts.

Font sizes can also be changed for sections of text. The default font size of `\normalsize` is 10pt font, but that can be redeclared through something like `\documentclass[12pt]{article}`.

LaTeX Command	Result
\tiny{Example text.}	Example text.
\scriptsize{Example text.}	Example text.
\footnotesize{Example text.}	Example text.
\small{Example text.}	Example text.
\normalsize{Example text.}	Example text.
\large{Example text.}	Example text.
\Large{Example text.}	Example text.
\huge{Example text.}	Example text.
\Huge{Example text.}	Example text.

Results of commands impacting font size.

Modifying the Table of Contents

Adding this line to your preamble will add periods after the sections, subsections, etc. in the document, provided that you haven't redefined the numbers for the Table of Contents (TOC) too.

```
\titlelabel{\thetitle.\quad}
```

The next few lines will add dots across the TOC from the entry to the page number.

```
%For the dots across TOC
\renewcommand{\cftsecleader}{\cftdotfill{\cftsecdotsep}}
%For section TOC dots
\renewcommand\cftsecdotsep{\cftdot}
%For subsection TOC dots
\renewcommand\cftsubsecdotsep{\cftdot}
%For subsubsection TOC dots
\renewcommand\cftsubsubsecdotsep{\cftdot}
```

These lines will redefine the section (etc.) numbering in both the TOC and the document to have a period.

```
% To add period after section number in TOC
\renewcommand{\thesection}{\arabic{section}.}
% Period subsection number TOC
\renewcommand{\thesubsection}{\thesection\arabic{subsection}.}
% Period subsubsection number TOC
\renewcommand{\thesubsubsection}{\thesubsection\arabic{subsubsection}.}
```

Using minipage

Minipages are used to put pages side-by-side. This is particularly useful for formatting images next to each other. The first specified minipage is placed on the left, and the second is placed on the right.

```
\begin{figure}
\begin{minipage}[t]{0.45\textwidth}
\includegraphics[width=\linewidth]{figure-one.png}
\caption{Caption for Figure 1.}
\label{fig:figure1}
\end{minipage}\hfill
\begin{minipage}[t]{0.45\textwidth}
\includegraphics[width=\linewidth]{figure-two.png}
\caption{Caption for Figure 2.}
\label{fig:figure2}
\end{
```

Packages

There are a LOT of LaTeX packages out there. Here, I've highlighted a few that I either use all the time or that are more specific to chemistry.

siunitx

`siunitx` is a package for formatting SI units.

New units can be defined for siunitx by adding lines like this to the preamble:

```
\DeclareSIUnit{\calorie}{cal}
\DeclareSIUnit{\kcal}{\kilo\calorie}
\DeclareSIUnit{\atm}{atm}
```

These new units would then be called with:

```
\SI{5}{\kcal\per\mole}
\SI[separate-uncertainty=true, multi-part-units=single]{25.3 \pm 0.5}{\atm}
\SIrange[range-phrase = --, range-units=single]{2}{64}{\calorie}
```

Resulting in:

$5\ \mathrm{kcal\ mol^{-1}}$

$25.3 \pm 0.5\ \mathrm{atm}$

$2\text{–}64\ \mathrm{cal}$

User-defined `siunitx` *units.*

Adding the following line to the preamble will change the default use of “per” (such as in `\si{\meter\per\second}`) from -1 to /.

```
\sisetup{per-mode=symbol}
```

LaTeX Command	Result
1234.2433 m/s	1234.2433 m/s
\si{\meter\per\second}	$\mathrm{m\,s^{-1}}$
\num{1234.2433}	1234.2433
\si{1234.2433}	1234 2433
\SI{1234.2433}{\meter\per\second}	$1234.2433\ \mathrm{m\,s^{-1}}$
\SI[per-mode=symbol]{1234.2433}{\meter\per\second}	$1234.2433\ \mathrm{m/s}$
\numrange{0}{255}	0 to 255
\numrange[range-phrase = --]{0}{1}	0–1
\SIrange{5}{230}{\pico\second}	5 ps to 230 ps

A list of different formats for the `siunitx` package.

The other specific features, such changing the units to be listed once with `SIrange` and `SI`, separating uncertainty, or changing the range phrase can all be added to the `sisetup` for global modification, or they can be modified locally in the command itself.

```
\sisetup{per-mode=symbol, separate-uncertainty = true, multi-pa
rt-units = si
ngle, range-phrase = --, range-units=single}
```

LaTeX Command	Symbol	LaTeX Command	Symbol
\si{\angstrom}	Å	\si{\farad}	F
\si{\ampere}	A	\si{\coulomb}	C
\si{\hertz}	Hz	\si{\mole}	mol
\si{\kelvin}	K	\si{\joule}	J
\si{\watt}	W	\si{\celsius}	°C
\si{\newton}	N	\si{\degree}	°
\si{\nano\meter}	nm	\si{\kilo\gram}	kg
\si{\ohm}	Ω	\si{\liter}	L
\si{\volt}	V	\si{\electronvolt}	eV
\si{\bar}	bar	\si{\decibel}	dB
\si{\mmHg}	mmHg	\si{\square\meter}	m^2
\si{\meter\squared}	m^2	\si{\tesla\cubed}	T^3
\si{\planckbar\tothe{5}}	$\hbar^5$	\si{\clight}	c_0
\si{\hartree}	E_h	\si{\electronmass}	m_e

A list of different options and symbols for use with the `siunitx` package.

listings

Code segments can be highlighted using the `listings` package. Set-up for `listings` includes defining the specific code environment(s) in the preamble.

The following preamble settings include P1 for colored Python, P2 for uncolored Python, and latex for LaTeX code.

```
\lstdefinestyle{P1}{language=python,frame=tb,aboveskip=3mm,belo
wskip=3mm,
showstringspaces=false,columns=flexible,basicstyle={\small\ttfa
mily},
numbers=none,numberstyle=\tiny\color{gray},keywordstyle=\colo
r{blue},
commentstyle=\color{dkgreen},stringstyle=\color{mauve},breaklin
es=true,
breakatwhitespace=true,tabsize=3,upquote=true}
\lstdefinestyle{P2} {language=python,frame=tb,aboveskip=3mm,bel
owskip=3mm,
showstringspaces=false,columns=flexible,basicstyle={\small\ttfa
mily},
numbers=none,breaklines=true,breakatwhitespace=true,tabsize=3,u
pquote=true}
\lstdefinestyle{latex} {
language=[LaTeX]TeX,frame=tb,aboveskip=3mm,belowskip=3mm,
showstringspaces=false,columns=flexible,breaklines=true,breakat
whitespace=true,
tabsize=3,basicstyle={\normalsize\ttfamily},keywordstyle=\colo
r{blue},
identifierstyle=\color{red},upquote=true}
\lstset{language=python,frame=tb}
\lstset{language=python,frame=tb}
\lstset{language=[LaTeX]Tex,frame=tb}
```

Using one of these languages would the be evoked through something like:

```
\begin{lstlisting}[style=latex]
\usepackage{package-name}
\end{lstlisting}
```

The final portion of the `\lstdefinestyle, upquote=true`, is only available if the `textcomp` package is also loaded. Loading that package is helpful so that code with apostrophes or double quotes is able to be copied correctly.

pdfcomment

`pdfcomment` is a package that helps annotate PDFs. It can thus be used to make PDFs more accessible, such as by adding alternative text to images.

Loading in the package with the linewidth options specifies the line width in annotations.

```
\usepackage[linewidth = 0]{pdfcomment}
```

```
{\pdftooltip
{\includegraphics[width=100mm]{image.png}}{This is the alternat
ive text.}
}
```

Specific aspects of equations can be identified through the `pdfmarkupcomment` command.

sansmathfonts

`sansmathfonts` is a package extends the general LaTeX sans serif font to small caps and math.

textgreek

The `textgreek` package avoids the use of unnecessary math environments in formatting Greek letters.

Instead of `$\kappa$`, you would use `\textkappa`.

color (and xcolor)

The `color` package allows users to define colors and use them throughout documents. The LaTeX color website has the definition lines for hundreds of colors (and their associated HEX codes).

Colors can be defined by using `rgb` values (on a scale of 0–1), `RGB` values (on the traditional 0–255 scale), HEX values (known to LATEX as `HTML`), or `cmyk` values (on the 0 to 1 scale). Numbers can also be defined using `gray` from 0 to 1. These color definitions start by naming the color, specifying the color scale, and then giving the range of values.

```
\definecolor{dkgreen}{rgb}{0,0.6,0}
\definecolor{gray}{rgb}{0.5,0.5,0.5}
\definecolor{mauve}{rgb}{0.58,0,0.82}
\definecolor{classicrose}{rgb}{0.98, 0.8, 0.91}
\definecolor{gray(x11gray)}{rgb}{0.75, 0.75, 0.75}
```

The `xcolor` package also exists and allows for color mixing. Original colors are still defined using `\definecolor`, but mixed colors are defined using `\colorlet`.

To make a color called `purpling` that's 40% blue and 60% red, the definition line would be:

```
\colorlet{purpling}{blue40red}
```

mhchem

The `mhchem` package allows the formatting of chemical reactions.

The sentence "If you start with 25 g of Na_2SO_4, how many grams of SO_4^{2-} can be made?" would be properly rendered with:

```
If you start with \SI{25}{\gram} of \ce{Na2SO4}, how many gram
s of \ce{SO4^2-} can be made?
```

modiagram

The `modiagram` package generates beautiful MO diagrams for s and p orbitals.

This example shows the molecular orbital energy diagram for N_2^+.

```
\begin{center}
\begin{MOdiagram}[labels,
labels-fs=\footnotesize,
names,
names-style = {
anchor=north,
text height = 1.5ex,
text depth = .25ex,
draw = black,
rounded corners}
]
 \atom[N]{left}{
      1s, 2s, 2p = {;up,up,up}
  }
  \atom[\ce{N+}]{right}{
      1s, 2s, 2p = {;up,up}
  }
  \molecule[\ce{N2+}]{
      1sMO, 2sMO, 2pMO = {;pair,pair,up}
  }
\end{MOdiagram}
\end{center}
```

tikzorbital

`tikzorbital` is a package that uses Tikz to generate s, p, and d orbitals, as well as MO diagrams, with shading.

This example creates p_y and a p_z orbitals.

```
\begin{center}
\begin{tikzpicture}
%        \orbital[pos = {(0,3)}]{px}
%        \node[above] at (0,4) {p$_x$};
        \orbital[pos = {(2,3)}]{py}
        \node[above] at (2,4) {p$_y$};
        \orbital[pos = {(4,3)}]{pz}
        \node[above] at (4,4) {p$_z$};
\end{tikzpicture}
\end{center}
```

chemfig

`chemfig` is a package that creates chemical structures and reaction diagrams. The package can be used to make schemes, and even color specific atoms.

This example makes a small, blue benzene become a larger, black benzene.

```
\begin{center}
\setchemfig{double bond sep=4pt}
\schemestart
\footnotesize \chemfig{*6((-H)-(-H)=(-H)-(-H)=(-H)-(-H)=(-H))}
\normalsize \arrow([blue]--[black]){<=>}
\chemfig{*6((-H)-(-H)=(-H)-(-H)=(-H)-(-H)=(-H))}
\schemestop
\end{center}
```

chemstyle

The `chemstyle` package formats Latin phrases and has a symbol for standard state. Aptly, the command for that is just `\standardstate`.

xspace

The `xspace` package has a way to create the degree symbol.

Adding this:

```
\newcommand{\degree}{\ensuremath{{}^{\circ}}\xspace}
```

to the preamble will mean that using `\degree` will make the symbol.

braket

The `braket` package can be used to typeset Dirac notation and sets. The `physics` package does the same, but with substantially more features.

Caver Overview

CAVER is a software tool for analyzing tunnels in biomolecular structures.

There are several tools: (a) CAVER, a command-line tool; (b) CAVER Analyst, a GUI; and (c) a CAVER PyMOL plugin.

CAVER (Command-Line)

The command-line version requires an input file, described on the next page (page 24). Once the input file is generated, you can run it using a script, like `run-cluster.sh` below.

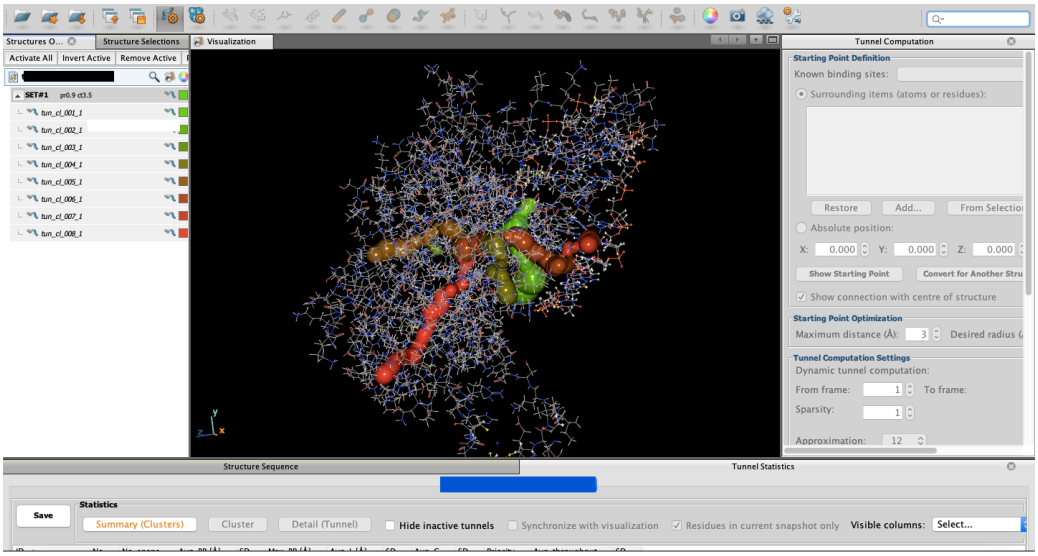
```
#!/bin/bash

## Set program location (can be in .bashrc instead)
export CAVER="/home/$USER/bin/caver_3.0/caver"

## Areas of red conservation
##
## 601 is metal center
## 603 is cofactor

listVar=(601 603)

## Backup your original config file
cp config.txt config-start.txt

for RESNUM in "${listVar[@]}"
do
        ## Use double quotes so the shell can substitute variab
les
        ## Replace the full line with "starting_point_residue
$RESNUM"
        sed "22c\starting_point_residue $RESNUM" config-start.t
xt > config.txt

        ## Run the Caver program
        java -Xmx10000m -cp $CAVER/lib -jar $CAVER/caver.jar -h
ome $CAVER -pdb ./ -conf ./config.txt -out ./caver_output_res$R
ESNUM

        ## Copy the config file used to the output folder
        cp config.txt ./caver_output_res$RESNUM/

done
```

Part of what this script does is modify the input file (`config.txt`) to account for residues that the user specifies under `listVar`. It does this by accessing the 22nd line and replacing what is there with a new residue number via `starting_point_residue $RESNUM`. After modifying the file and resaving it, the program is run. The program output files are then saved to a newly generated subfolder, named using the specified residue number.

> ℹ **Note:** If you modify the input file layout, you will also have to change the `22c` portion of the sed command, based on its new line number.

CAVER Analyst

Once a structure is loaded in (`File → Open Structure`), follow `Tunnel → Computation` . From there you can enter all of the criteria for a tunnel search.



Tunnels identified with CAVER Analyst.

Under `Known binding sites` , a specific residue or atom can be given using the PDB's residue or atom number. It is suggested that you use a cofactor or inhibitor for this criterion.

PyMOL plugin

PyMOL is a molecular visualization software. To use PyMOL for academic research, you need to purchase a license. More information on that is available on their website.

Once both PyMOL and the plugin are installed, you can use CAVER interactively. The information from the command-line input file becomes fill-in-the-blank boxes.

Caver Input

The input file contains a lot of information. Each of the keywords is described in the CAVER documentation .

A particular PDB residue or atom number in the PDB can be specified with `starting_point_residue` or `starting_point_atom`. It is suggested that you use a cofactor or inhibitor for these keywords.

```
#*****************************
# CALCULATION SETUP
#*****************************
load_tunnels no
load_cluster_tree no
stop_after never

#*****************************
# INPUT DATA
#*****************************
time_sparsity 1
first_frame 1
last_frame 1
#last_frame 100000

#*****************************
# TUNNEL CALCULATION
#*****************************

## RES 601 ==  MN
#starting_point_atom 7263
starting_point_residue 603
#starting_point_coordinates

## Default probe = 0.9, shell_rad = 3, shell_depth = 4
probe_radius 0.9
shell_radius 3
shell_depth 4


#*****************************
# TUNNEL CLUSTERING
#*****************************
clustering average_link
weighting_coefficient 1
clustering_threshold 3.5

exclude_start_zone 2
exclude_end_zone 0
min_middle_zone 5
save_zones no


#*****************************
# GENERATION OF OUTPUTS
```

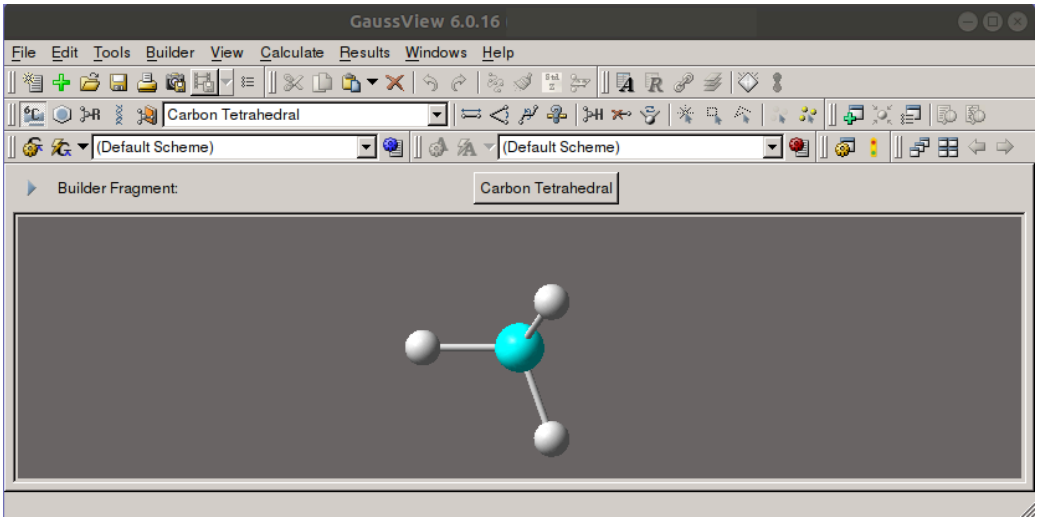
```
#******************************
one_tunnel_in_snapshot cheapest
max_output_clusters 999
save_dynamics_visualization yes
#save_dynamics_visualization no

generate_summary yes
generate_tunnel_characteristics yes
generate_tunnel_profiles yes

generate_histograms no
bottleneck_histogram 0.0 2.0 20
throughput_histogram 0 1.0 10

generate_bottleneck_heat_map no
bottleneck_heat_map_range 1.0 2.0
bottleneck_heat_map_element_size 10 10

generate_profile_heat_map yes
profile_heat_map_resolution 0.5
profile_heat_map_range 1.0 2.0
profile_heat_map_element_size 20 10

compute_tunnel_residues yes
residue_contact_distance 3.0

compute_bottleneck_residues yes
bottleneck_contact_distance 3.0


#******************************
# ADVANCED SETTINGS
#******************************

#------------------------------
# Starting point optimization
#------------------------------
#max_distance 3
#desired_radius 5
max_distance 10
desired_radius 5

#------------------------------
# Advanced tunnel calculation
#------------------------------
```

```
#number_of_approximating_balls 12
number_of_approximating_balls 20
add_central_sphere yes

max_number_of_tunnels 10000
max_limiting_radius 100

cost_function_exponent 2

automatic_shell_radius no
automatic_shell_radius_bottleneck_multiplier 2
starting_point_protection_radius 4

#-----------------------------
# Redundant tunnels removal
#-----------------------------
frame_clustering yes
frame_weighting_coefficient 1
frame_clustering_threshold 1

frame_exclude_start_zone 0
frame_exclude_end_zone 0
frame_min_middle_zone 5

#-----------------------------
# Averaging of tunnel ends
#-----------------------------
average_surface_frame yes
average_surface_global yes

average_surface_smoothness_angle 10
average_surface_point_min_angle 5
average_surface_tunnel_sampling_step 0.5

#-----------------------------
# Approximate clustering
#-----------------------------
do_approximate_clustering no
cluster_by_hierarchical_clustering 20000
max_training_clusters 15
generate_unclassified_cluster no

#-----------------------------
# Outputs
#-----------------------------
```

```
profile_tunnel_sampling_step 0.5
visualization_tunnel_sampling_step 1

visualize_tunnels_per_cluster 5000
visualization_subsampling random

compute_errors no
save_error_profiles no

#path_to_vmd "C:/Program Files/University of Illinois/VMD/vmd.exe"
path_to_vmd /usr/local/bin/vmd
generate_trajectory no

#------------------------------
# Others
#------------------------------
swap yes
seed 1    #no default, if missing, the seed is random
```

Gaussian Overview

Gaussian is a program for computational chemistry. GaussView is the graphical viewer that assists with using Gaussian. Gaussian can be used for a variety of calculations, including molecular orbitals, energies, vibrational frequencies, and structure. Most of these explanations will be about how to use GaussView (to later use Gaussian).



The GaussView 6.0 window.

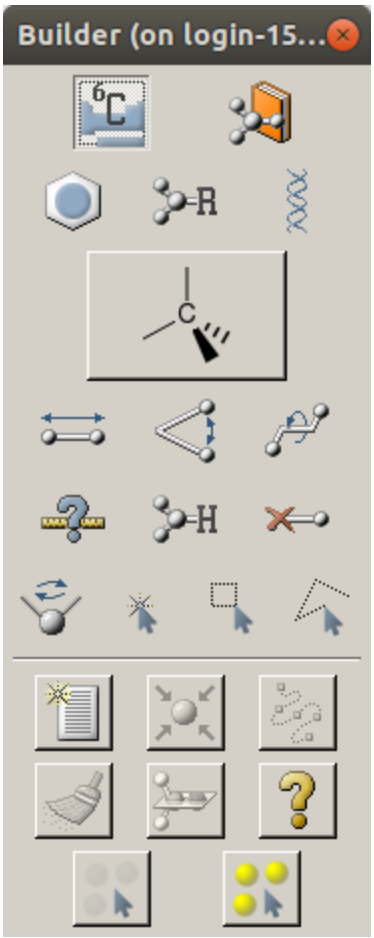
Navigation

Mouse Button	Action	Use
Left	Click	Selects or adds item
	Move left/right	Rotate y-axis
	Move up/down	Rotate x-axis
Scroll	Roll	Zoom in/out molecule
Right	Move left/right	Rotate z-axis
	Move up/down	Zoom in/out molecule

Builder

The builder has a lot of buttons that each perform a different function. These functions, spaced like the builder, are:

Element Fragment | Custom Fragment
Ring fragment | R-group fragment | Biological fragment
Fragment Choice
Distance | Angle | Dihedral
Inquire | Add hydrogen | Delete group
Rotate | Click Atoms | Box-select | Drag-select
New build window | Center window | Rebond
Clean structure | Symmetrize | Builder Help
Deselect all atoms | Select all atoms



The GaussView builder.

Gaussian Input Files

Gaussian input files have the file extension `.gjf` or `.com`. Jobs will read in data from a checkpoint (`.chk`) file, or write to new file. Logfiles (`.log`) contain all the information about the job, and whether it failed or ran successfully.

> ❗ **Warning:** A successful run does not mean correct run. Check your data! If it doesn't make chemical sense, question it!

Water Example

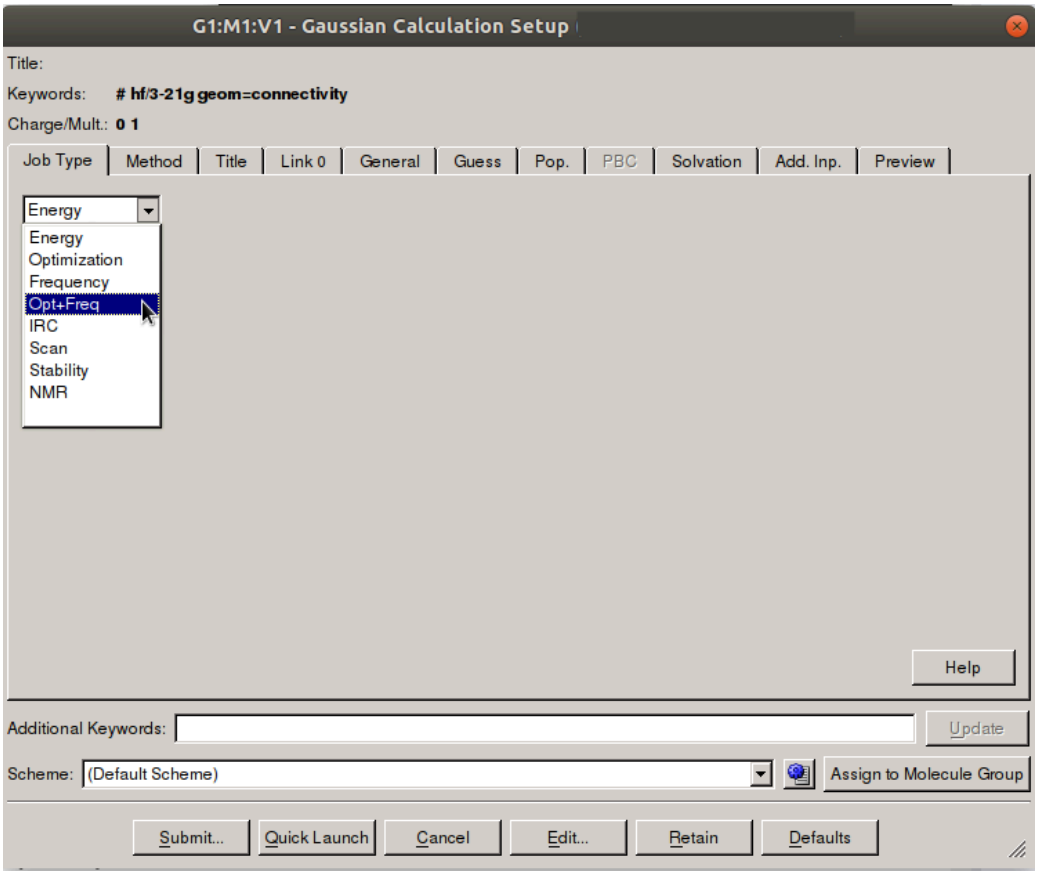
Here is what a basic input for water would look like.

```
%mem=32MB
% chk=water.chk
# opt freq geom b3lyp/6-31+g(d,p) geom=connectivity integral=ul
trafine scf=maxcyc=1024

water

0 1
O 1.80172602 0.06746038 0.00000000
H 2.76172599 0.06770148 0.00000000
H 1.48149872 0.97247667 0.00000000
```

This input, with explanatory comments marked with `###` is:

```
%mem=32MB ### system resources (both % words or %words OK)
% chk=water.chk ### specifies the checkpoint file to write
# opt freq geom b3lyp/6-31+g(d,p) geom=connectivity integral=ul
trafine scf=maxcyc=1024 ### Job specifics—NO ENTER KEY
### I am a required blank line
water ### Titlecard: this line is required
### I am a required blank line
0 1 ### Charge, spin multiplicity
O 1.80172602 0.06746038 0.00000000
H 2.76172599 0.06770148 0.00000000
H 1.48149872 0.97247667 0.00000000
### I am a required blank line
```

`Calculate → Gaussian Calculation Set-Up`



Different Gaussian job types.

Job Type

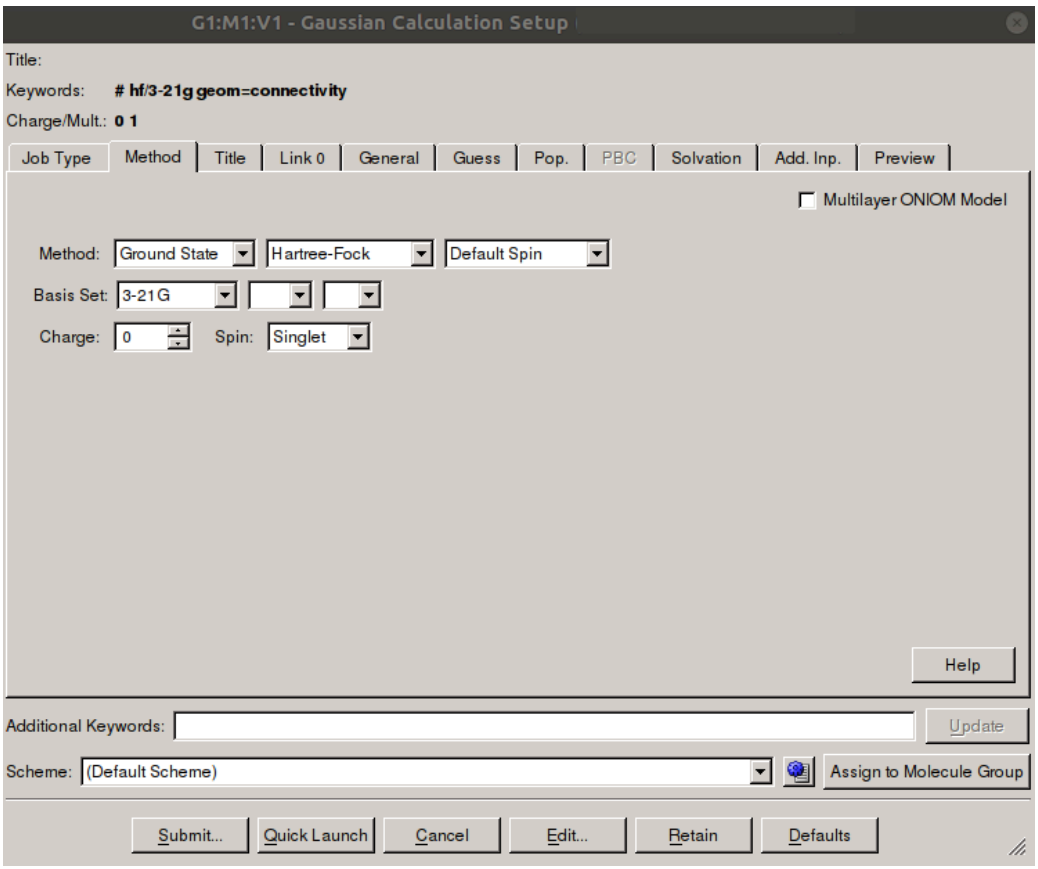
1. Energy: Calculates the energy and wavefunction at a single, fixed geometry
2. Optimization: Attempts to find the structure's minimum energy configuration
3. Frequency: Gives thermochemical properties of the structure
4. Opt + Freq: Performs both optimization and frequency
5. IRC: Asks to follow a reaction path by integrating the intrinsic reaction coordinate
6. Scan: Scans the potential energy surface by performing single-point energy calculations
7. Stability: Checks stability by determining if imaginary (negative)

frequencies exist (Yes? That's bad, unless a transition state.)

8. NMR: Predicts NMR shielding tensors

Method

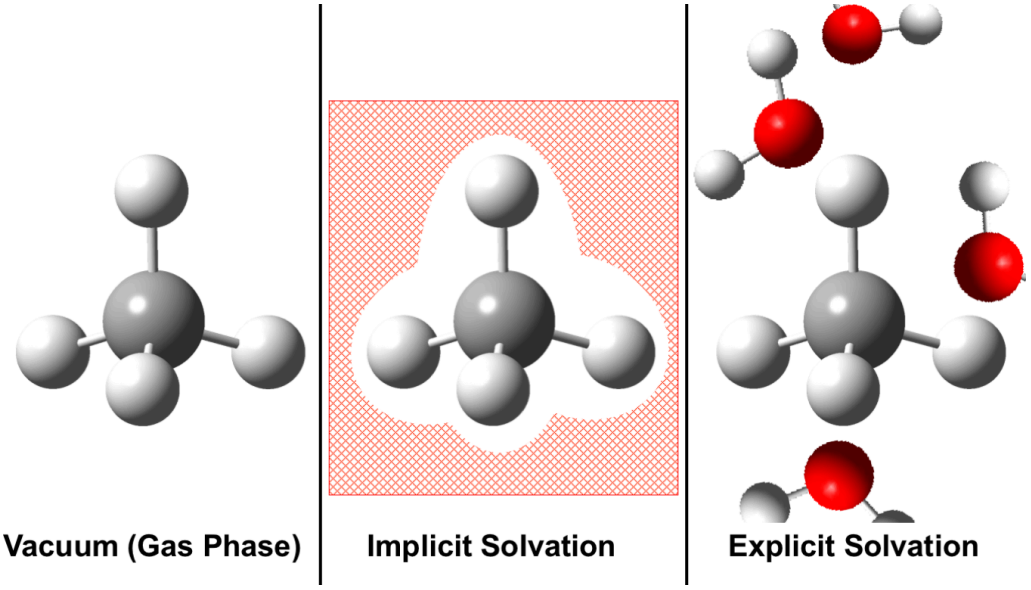
- Method: the "school of thought" for the wavefunction
- Basis set: contains all the functions that represent the wavefunction
    - Higher-order basis sets are often more accurate, but at a higher computational/time cost
    - Diffuse functions (ex: +, ++) describe very electronegative atoms
    - Polarization functions (ex: d, p) allow for angular flexibility (i.e., reduces the strain from lots of electrons)
- Charge: overall charge of the structure
- Spin: pairing of the electrons
    - Singlet: All electrons paired
    - Multiplicity = Number of unpaired electrons + 1



The method tab.

Solvation

- Implicit: Solvent is a polarizable continuum with dielectric constant, ε
    - Not terribly costly
    - Cannot model specific interactions, like hydrogen bonds
    - Magician waving a wand there's some magic happening, but you can't see it
- Explicit: Want a solvent? Build it in. Put it there.
    - Expensive…
    - Can get stuff like hydrogen bonds
    - As magician's apprentice, you see all the things going into the "magic"



Different solvation models.

Viewing Gaussian Results

To open a checkpoint file, follow:

1. `File`
2. `Open`
3. Select the `.chk` file

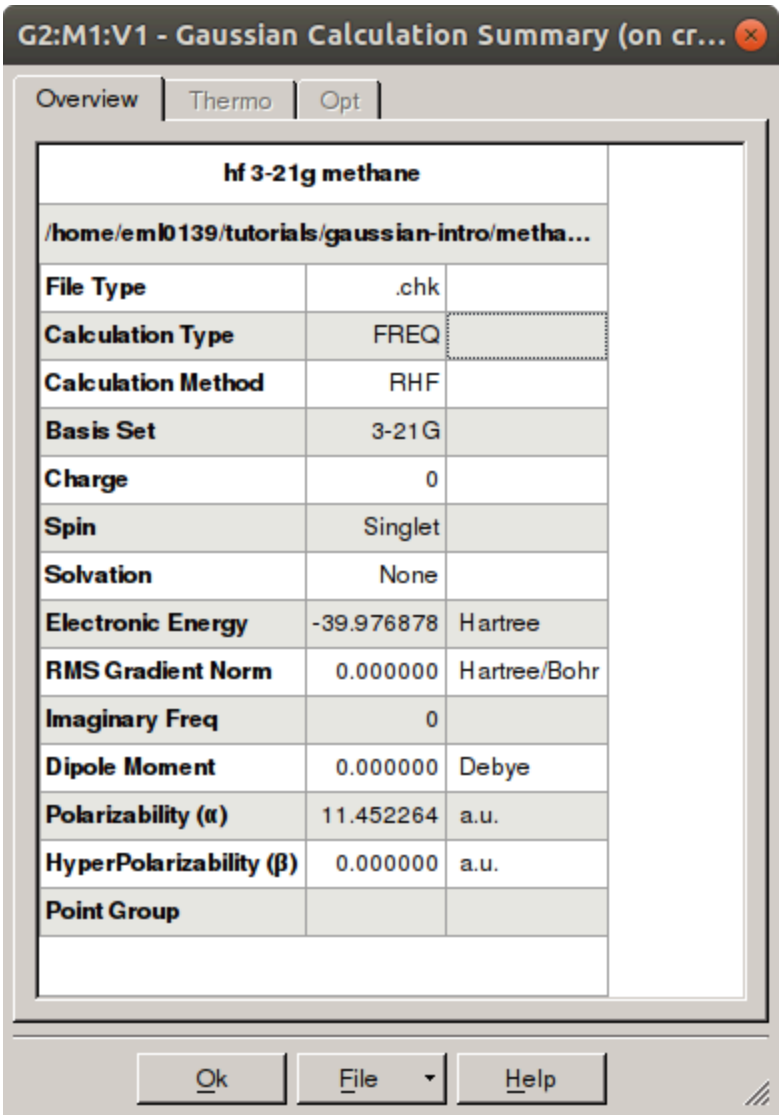
Gaussian Logfile

The logfile (`.log`) contains all of the job information. You should always check the logfile for successful job completion! The “it did something” way is to check the end for an end quote. The “real” way is to check for convergence. That said, not everything that has a Gaussian logfile will have convergence information, but it will have information indicating that what you were doing worked.

```
IF OTHER PEOPLE ARE GOING TO TALK, CONVERSATION IS SIMPLY
IMPOSSIBLE.
-- WHISTLER'S
PRINCIPLE
Job cpu time: 0 days 0 hours 0 minutes 3.0 seconds.
Elapsed time: 0 days 0 hours 0 minutes 2.8 seconds.
File lengths (MBytes): RWF= 6 Int= 0 D2E= 0 Chk= 1 Scr= 1
Normal termination of Gaussian 16 at Fri Jun 7 10:02:03 2019.
```

Summary

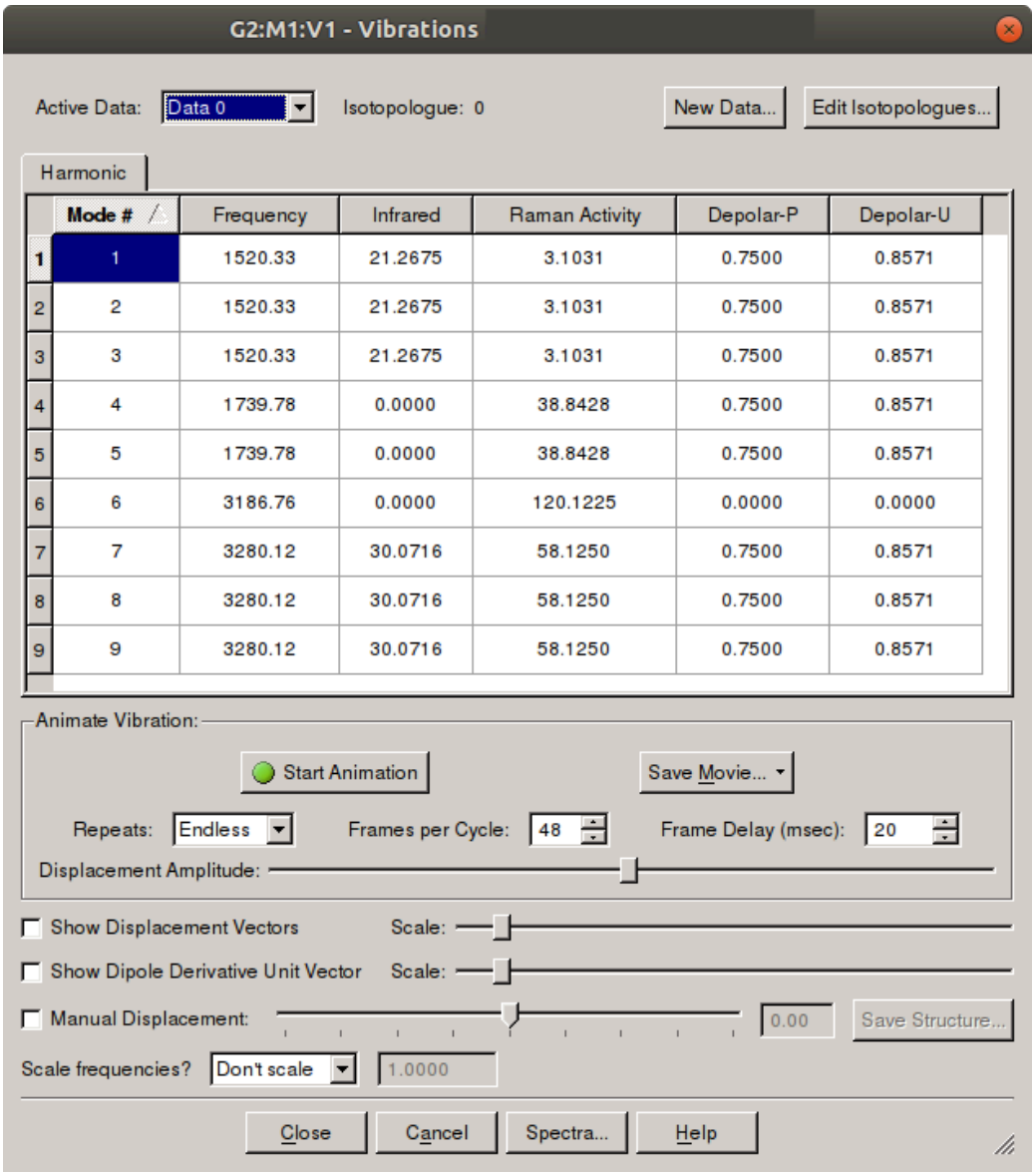
The summary window contains job information and some results. To access the results summary, follow: `Results → Summary`. One E_h (hartree) is equivalent to 627.509474 kcal mol^{-1}. There’s a great energy converter page from Colby College that you can use to make this more meaningful to you.



The Gaussian summary window.

Vibrations

The vibrations window contains frequency and Raman information. From this window, you can watch animations of stretching and vibrations. To access the window, follow: `Results → Vibrations`.

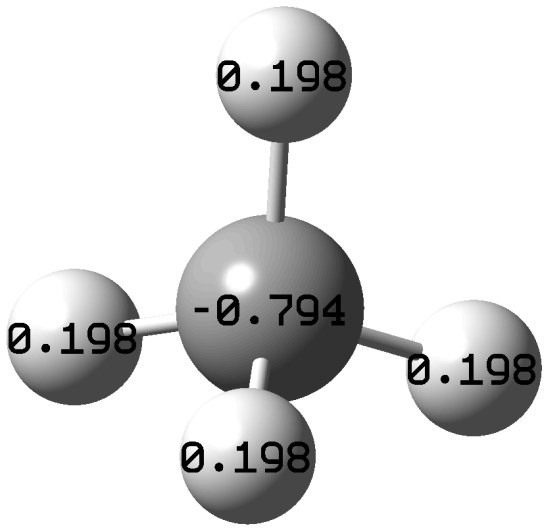


	Mode #	Frequency	Infrared	Raman Activity	Depolar-P	Depolar-U
1	1	1520.33	21.2675	3.1031	0.7500	0.8571
2	2	1520.33	21.2675	3.1031	0.7500	0.8571
3	3	1520.33	21.2675	3.1031	0.7500	0.8571
4	4	1739.78	0.0000	38.8428	0.7500	0.8571
5	5	1739.78	0.0000	38.8428	0.7500	0.8571
6	6	3186.76	0.0000	120.1225	0.0000	0.0000
7	7	3280.12	30.0716	58.1250	0.7500	0.8571
8	8	3280.12	30.0716	58.1250	0.7500	0.8571
9	9	3280.12	30.0716	58.1250	0.7500	0.8571

The Gaussian vibrations window.

Charge Distribution

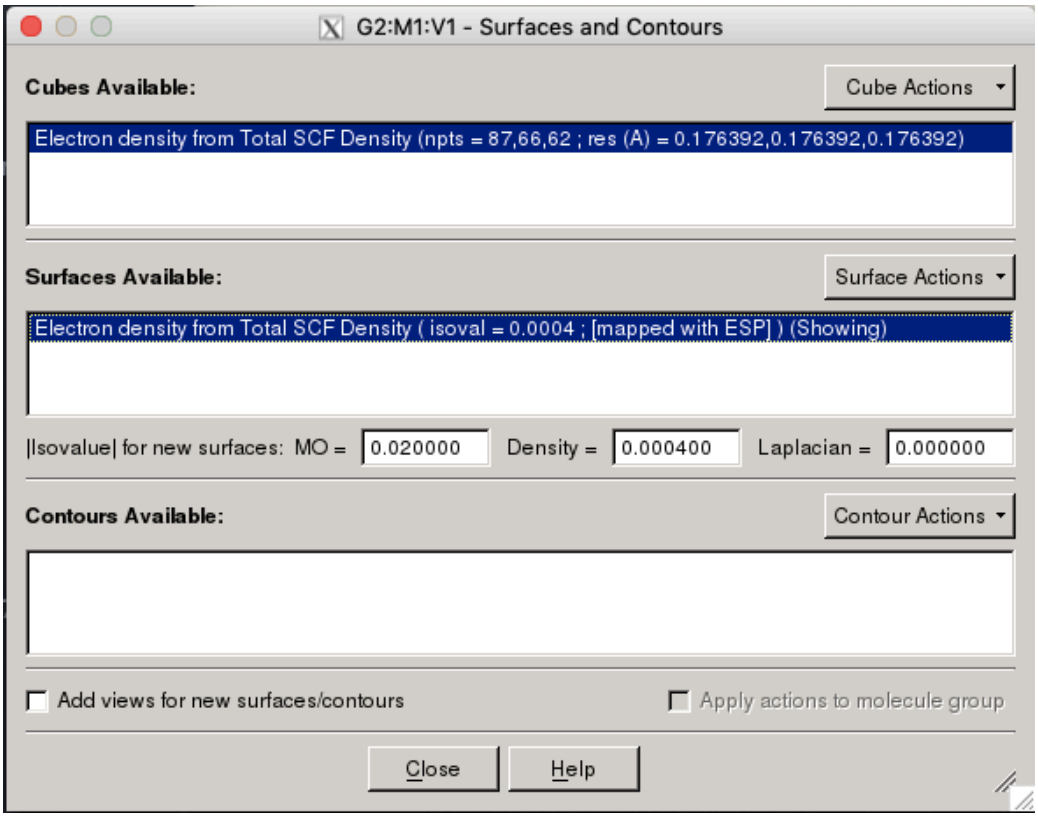
The charge distribution can be plotted on the structure. To access the distribution, follow: `Results → Charge Distribution`.



Charges mapped onto methane.

Cube Data

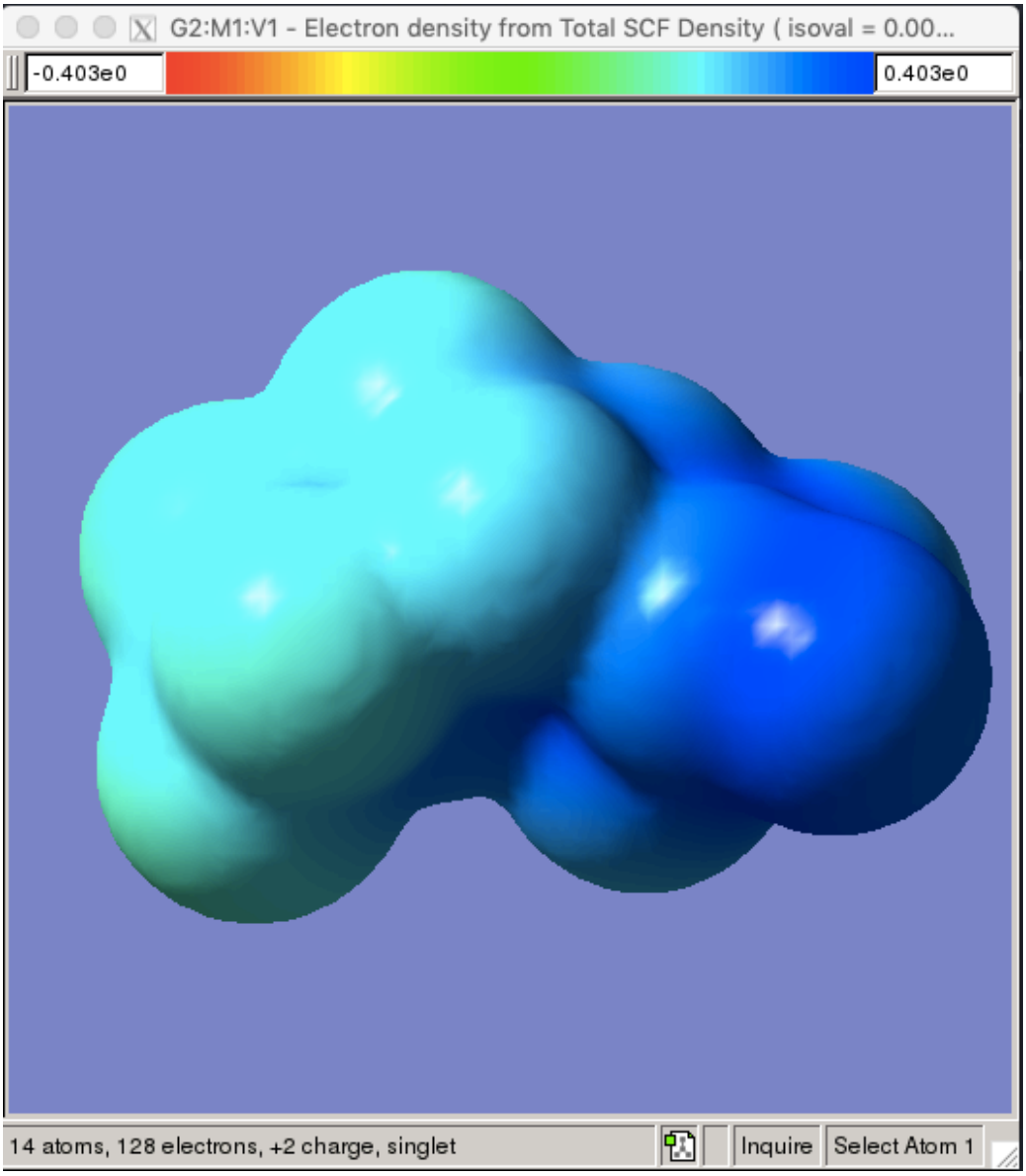
Different data can be plotted onto the structure. This can be accomplished by following `Results → Surfaces/Contours`, which brings up the surfaces and contours window.



The Surfaces and Contours window. Here, the electron density cube was selected and and ESP surface was mapped onto it.

First, pick a type of information to show on the structure through `Cube Actions → New Cube`. You can plot several different properties. For the example below, I picked the total density as the content for the new cube. The grid options specify how smooth the surface looks. `Coarse` is fine for quick images, and `Medium` should be alright for presentations or posters.

You can also use those cubes to plot different information. To do so, follow `Surfaces Available → New Mapped Surfaces` in the second block. I picked electrostatic potential (ESP) for the example.



The electrostatic potential mapped onto the structure.

Cubes can be saved, but surfaces will be recalculated each time.

Making Images

While a white background is preferred for scientific images, GaussView uses purple by default.

> ℹ **Note:** Try to avoid red/green and blue/yellow color combinations, as they are common forms of colorblindness.

Change the background color

1. Right-click on the image
2. `View`
3. `Display Format`
4. `General`

Save image (allows you to reset the background color)

1. Right-click on the image
2. `File`
3. `Save Image`
4. `Advanced`
5. `File Save Options`

Transparent background

1. Set the alpha channel value to zero (0)
2. Save as `.png` or `.tif`

LICHEM Overview

LICHEM interfaces QM and MM software for QM/MM calculations. It can be downloaded from GitHub

As of right now, it has wrappers for Gaussian, NWChem, PSI4, and TINKER.

Using PDBXYZ

LICHEM converts a TINKER XYZ file to a standard XYZ file. That means that a PDB file would first need to be converted to a TINKER XYZ, which can be done using `pdbxyz`.

`pdbxyz` is a program in the TINKER software package. Its usage is:

```
$ pdbxyz name_of_pdb.pdb -key tinker.key
```

Running this command with a complete PDB often causes issues. One way to avoid these issues is to break up the PDB into several smaller PDBs and convert them individually. This is particularly critical for double-stranded nucleic acids, where each strand needs to be in its own PDB file to convert correctly. It is recommended that every location that a TER would be starts a new file to convert, as well as adding in other new files. So, one system could be broken into:

1. Protein
2. Non-standard residue
3. Substrate nucleic acid strand
4. Complement nucleic acid strand
5. Metal
6. Counterions
7. Water

After using `pdbxyz` on each of these individual files, all the files can be combined into a single TINKER XYZ file using TINKER's `xyzedit`.

```
$ xyzedit name_of_protein.xyz -key tinker.key
```

This will bring up a list of options, one of which is `Append a second XYZ file to current one`. Enter the number corresponding to this option (likely `18`) and then enter in the next XYZ file in the order you'd like them to be appended. Once you've finished choosing this option and file addition process, hit enter without entering a new option. The new, combined file will be named `name_of_protein.xyz_2`.

Set-Up Using Tinker and Gaussian

> **Warning:** This page is under development and the information contained herein may not be a complete representation of the set-up process.

First, install Tinker7 on your machine from the Tinker website . While Tinker8 has been released, the use of the analyze command is different than in Tinker7, and will make generating the frozen atoms list more difficult.

> **Tip:** If you have multiple versions of Tinker installed, export a global variable to the bin of each edition name. Bash shell example: `export Tinker7=/home/$USER/bin/tinker7/bin` and then call the program with `$Tinker7/xyzedit`.

Converting the TINKER XYZ with LICHEM

Once you have a Tinker XYZ file, create a `param.key` file that contains only the line with the requisite Tinker force field information. It should look something like:

```
parameters amber99.prm
```

> **Note:** You can generate one key file named `tinker.key` and update the information therein, but for the purposes of this explanation, every key file with new information will be explicitly named.

Now, use LICHEM to convert these to the necessary format.

```
lichem -convert -t my_tinker_system.xyz -k param.key > conversi
on-1.log
```

This conversion created several new files: `regions.inp`, `connect.inp`, and `xyzfile.xyz`.

The original regions file looks like this:

```
QM_atoms: 0
Pseudobond_atoms: 0
Boundary_atoms: 0
Frozen_atoms: 0
```

In this file, list the `QM_atoms`, `Pseudobond_atoms`, and `Frozen_atoms`. This definition includes the keyword, then the number of total atoms in that category, followed by each individual index number of the atoms in that category. This indexing starts with the first atom being `0`–the same as VMD's indexing, but **not** the same as the Tinker XYZ file.

For example, a system with 5 QM atoms would look like:

```
QM_atoms: 5
101 107 110 591 3822
```

☑ **Tip:** Try to list atoms in rows of 10. It will make it easier to determine which atoms need unique basis sets when they are renumbered in later steps.

Once defined, save the `regions.inp` and create a backup (e.g., `qm_list.txt`) so that you do not lose this information in the future.

Generate List of "Active" Atoms

Once the QM, boundary, and pseudobond atoms have been defined, the list of frozen atoms needs to be generated. This is done by specifying a sphere around a specific atom. Tinker7 can generate this list, but it needs some specific information in the key file.

The `get-active.key`:

```
parameters amber99sb.prm
neutral-groups
debug
sphere 3822 15
```

The `get-active.key` file starts with the specific parameter file that corresponds to the XYZ. Charge-charge interaction distances are addressed through `neutral-groups`, and `debug` prints the detailed information that contains the active atoms. Finally, the `sphere` line specifies the atom number in the Tinker XYZ (i.e., the atom in the XYZ file here we want is 3822) and the size of the sphere in angstroms.

While we're generating keys, copy the `param.key` file as `active_tinker.key`. This will be used after the analyze step.

```
$ cp param.key active_tinker.key
```

Once those keys are written, run `analyze` with Tinker7 and the `get-active.key` to get the list.

```
analyze my_tinker_system.xyz -key get-active.key
> active atoms 1835
```

You can exit the program after it prints the first set of information. All you need is the list of active atoms that is printed initially (`List of Active Atoms for Energy Calculations :`) Copy that list into the previously created `active_tinker.key` file after the parameter line.

Each of these lines needs to be specified as `active` for TINKER. This can be accomplished by using `sed` to edit the file. These lines with remove the tabs and then start each line with `active`.

```
$ sed -i -e 's/[ \t]*//' active_tinker.key
$ sed -i 's/^/active /' active_tinker.key
```

After using `sed`, edit the `active_tinker.key` file to remove the `active` on the `parameters` line.

Now, use this to add the frozen atoms to the regions file.

> ❗ **Warning:** This WILL delete the QM, boundary, and pseudobond atoms in the `regions.inp` file, so make sure those are saved in a backup file!!!

```
lichem -convert -t my_tinker_system.xyz -k active_tinker.key >
conversion-2.log
```

After the frozen atoms have been added to the regions file, re-add the list of QM, boundary, and pseudobond atoms. The boundary atoms also **need** to be repeated in the `Frozen_atoms` section. So, if the new `regions.inp` file contains 100 frozen atoms, and you specify 2 boundary atoms, you need to change it to 102 frozen atoms and add those 2 boundary atoms to the long list.

Add Keywords to the Regions File

It is now time to add the LICHEM-specific keywords. An example of some are listed below:

```
Potential_type: QMMM
QM_type: Gaussian
QM_method: B3LYP
QM_basis: GEN
QM_memory: 80 GB
QM_charge: -5
QM_spin: 2
MM_type: TINKER
Electrostatics: CHARGES
Calculation_type: DFP
Opt_stepsize: 1.00
Max_stepsize: 0.10
QM_opt_tolerance: 1e-3
MM_opt_tolerance: 1e-1
Max_opt_steps: 50
Init_path_chk: No
PBC: Yes
Box_size: 81.865 102.353 92.1
Use_LREC: Yes
LREC_cut: 25.0
Use_Ewald: Yes
Keep_files: Yes
```

The `Calculation_type` of `DFP` performs a Davidon-Fletcher-Powell optimization. It is recommended that the system first be tested using `SP` (for single-point) and ensuring that the energy is negative before continuing with `DFP`. Doing this first can help to debug problems that would arise in the longer calculation.

Using `GEN` for `QM_basis` means that the basis set information will be described in a separate file (named `BASIS` and described later).

The `box_size` information is system-specific. If you do not know your box size, and it was converted from an AMBER PDB, you can find that using Python as shown in the following code.

```
>>> import parmed as pmd
>>> pdb = pmd.load_file('my_original_system.pdb')
>>> pdb.get_box()
array([[81.865, 102.353, 92.1, 90., 90., 90. ]])
```

For this, the first three numbers, BoxX, BoxY, and BoxZ are used (**NOT** alpha, beta, and gamma).

Generate BASIS File

As mentioned, a `BASIS` file contains all of the basis set information to be used with Gaussian. LICHEM can generate this file with:

```
lichem -convert -b regions.inp
```

For the `BASIS` file, the numbers are based on the numerical order of what is listed in QM and pseudobonds sections of `regions.inp`. This means that if you have numbers `1123 1433 1353` listed in different places in the file, 1123 = 1, 1353 = 2, and 1433 = 3. As long as the referenced number is correct, you can mix and match each line (ex: `1 3 5` for `6-31G` & `2 4 6 7 8` for `6-31+G**`).

The format for the file looks like:

```
#1 #2 #3__0
GEN
****
```

There are 2 spaces between the last number and the zero.

Some examples of basis sets to put in the place of `GEN` include:

- STO-3G
    - STO-3G*

- 3-21G
- 3-21+G
    - 3-21+G*
    - 3-21+G**
- 6-31G
    - 6-31G*
    - 6-31G**
- 6-31++G
    - 6-31++G*
    - 6-31++G**
- 6-311G
- 6-311+G
- 6-311++G

You'll likely have to benchmark these for your system, as well as read the literature for the things that tend to work well for your system.

The pseudopotential information (for the pseudobonds) is listed at the end. The following uses fluorine for atoms 1, 8, 15, and 25.

```
1 8 15 25 0 STO-2G
SP 2 1.00
0.9034 1.00 1.00
0.21310 1.90904 0.57864
****

1 8 15 25 0
try1 1 2
S Component
1
1 7.75 16.49
P1
1 1.0 0.0
```

Running LICHEM

After the `BASIS` file has been written, LICHEM can now be run with Gaussian and TINKER. The following command will use 20 processors to do so.

```
## -n number of processors
## -x input xyz
## -c connectivity file
## -r regions file
## -o output xyz file
## -l output log file
lichem -n 20 -x xyzfile.xyz -c connect.inp -r regions.inp -o
system_out.xyz -l system_out.log
```

> ℹ **Note:** The `xyzfile.xyz` is the one that LICHEM generated with `lichem-convert`. A regular, non-Tinker XYZ can also be used if you have previously generated the `connect.inp` file and the numbering is the same between them.

Despite only listing `xyzfile.xyz`, `connect.inp`, and `regions.inp`, LICHEM will look for other program-specific files including `BASIS` and `tinker.key`. These files must be named accordingly.

Troubleshooting

There are a number of log files that are saved with a LICHEM job. An error could be documented in one of them, but not all of them, so it is important to look through them thoroughly. To put this another way, if it fails in one program, it will try to continue in another program, and errors may be in one log, but not the other. In a perfect world, you'll get convergence in the QM, normal termination in the MM, and a complete optimization for QM/MM.

Bad Box Information

When using LICHEM with periodic boundary conditions, the box size is given in the `regions.inp` file. However, if that box information is incorrect, the QM region may not converge because of it literally being spread across the box. The final XYZ generated by LICHEM will visibly look wrong when visualized.

Sometimes, incorrect box sizes will be accompanied by `Annihilation of the first spin contaminate:` in the `LICHM_GaussForce_0.log` file.

However, there is a chance that the physical size of the box is correct, but for some reason the system is not centered within the box. This can be addressed by using TINKER's `xyzedit` on the original TINKER xyz and selecting the `Translate Center of Mass to the Origin` option. Then, reconvert the TINKER xyz to the LICHEM xyz.

Connection Through Boundary

If you're trying to use the side chains of two residues that are next to each other, then you may get the following message:

```
Error: Two pseudo-bonds are connected through boundary atom
s!!! The connections prevent LICHEM from correctly updating th
e charges.
```

Like the message says, LICHEM complains about two pseudobonds being connected by boundary atoms. So, consider including the entirety of those two residues and cutting the backbone halfway on each side.

BASIS Issue

The BASIS file that is read by Gaussian is numbered using the `QM_atoms` and `Pseudobond_atoms` starting with `1`. If you have used an number in both the QM and pseudobond atom lists in the `regions.inp` file, or you have written a basis set incorrectly, you will likely get this error in your Gaussian output:

```
The center is too long.
```

Using a VMD-generated XYZ

If you've used VMD to save a starting XYZ for LICHEM (a true XYZ, not a Tinker XYZ), then there will be a default line written under the number of atoms.

```
8148
 generated by VMD
 N         2.990000       39.410000       20.542000
```

Having that line (even if it's blank) will result in an error with `Name of the center is too long` being printed to the Gaussian output file.

If the error persists, remove the extra whitespace at the beginning of each line using `sed` (page 0).

```
$ sed 's/^ *//' xyzfile_from_vmd.xyz > xyzfile_for_lichem.xyz
```