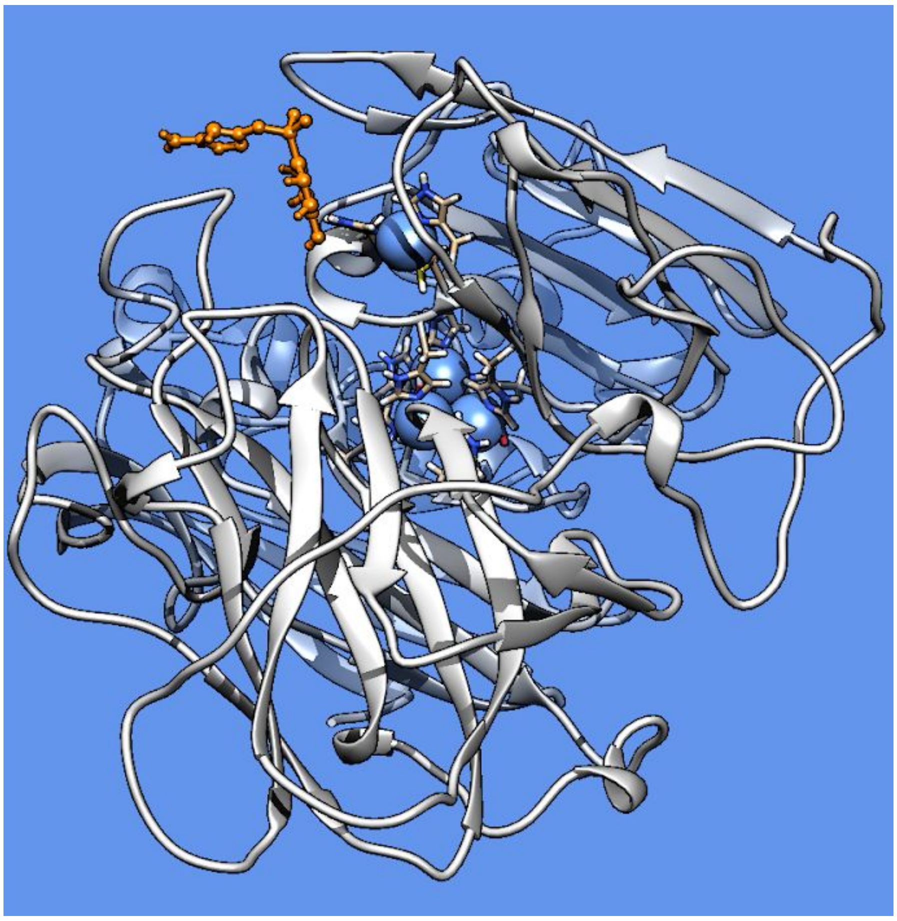
HANDBOOK IN MOLECULAR MODELING



Developed by and based on the work iGEM Stockholm 2018, this handbook serves the simple purpose of providing future iGEM teams with the possibility to perform molecular modelling in order to understand more about their experimental system using the most sophisticated methods to date.

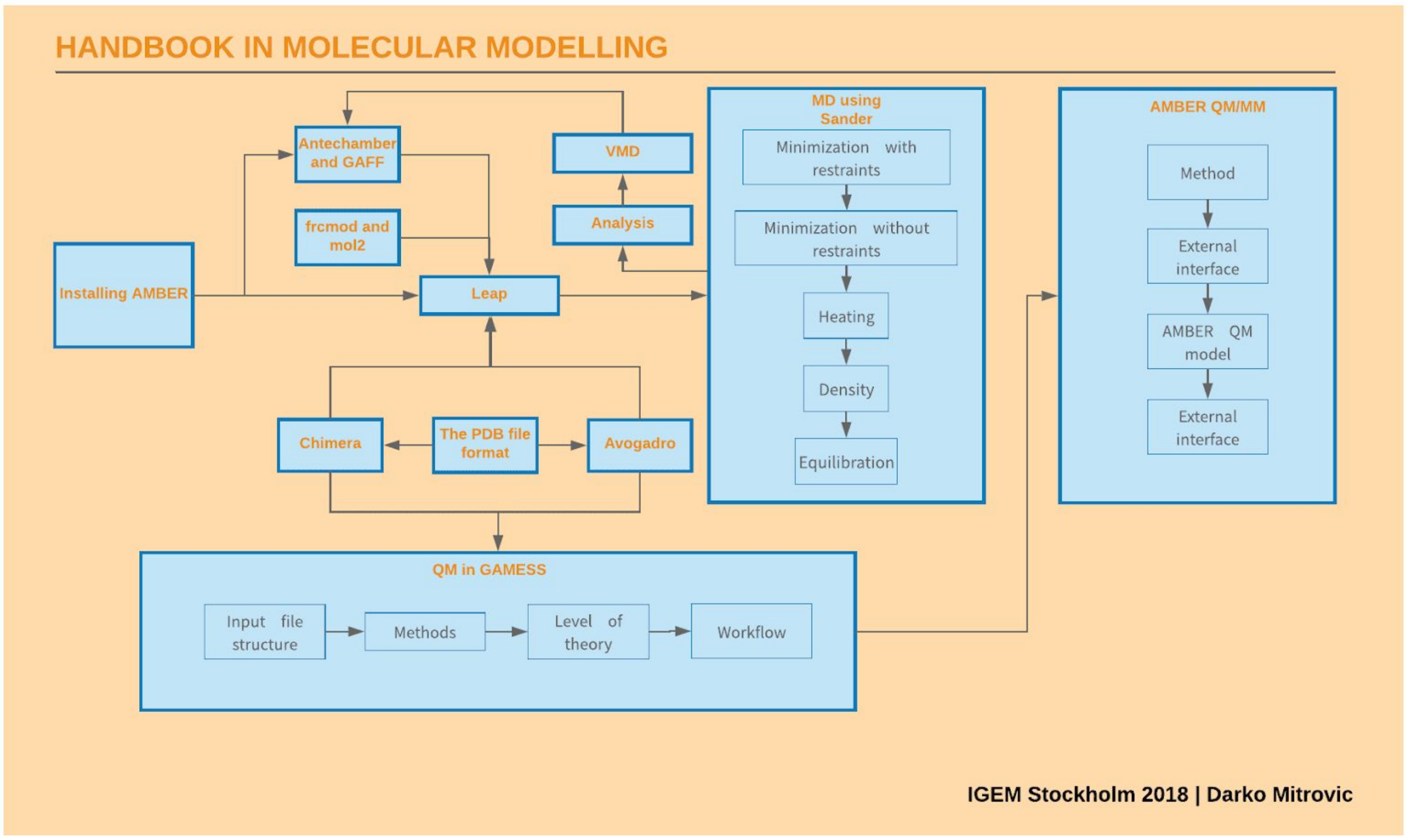
Preface

Hello and welcome to the handbook in molecular modelling! My name is Darko Mitrovic, and I am the author of this modest handbook. In the project of iGEM Stockholm 2018, the details of which you can see on our wiki, we attempted to develop an enzyme for a substrate using molecular modelling techniques. Due to the poor documentation of overall methods and why different steps are performed, we decided to make our own guide in how to do different types of molecular modelling. We hope that this handbook will provide enough of a basis for a reader of a synthetic biology background to perform their own modelling to provide a theoretical underline for the experimental work. We aim to introduce and spread these methods to be used within iGEM to fulfill the modelling goals and make modelling more accessible to teams who have never had experience in modelling.

Good luck!

Darko Mitrovic | iGEM Stockholm 2018

Contents



HANDBOOK IN MOLECULAR MODELLING

Welcome to the Handbook in molecular modelling! This is a handbook composed by iGEM Stockholm 2018, and can serve as a handy guide to anyone who is new to and want to explore the vast world of molecular modelling. Throughout this book, the flowchart above will guide you and show you how to go about doing (1) Molecular Dynamics simulations, (2) Quantum Mechanical calculations and (3) Quantum Mechanical and Molecular Mechanics simulations, to model and better understand your experimental system. There are no required prerequisite skills, other than an open mindset and a working computer!

Abstract

The role of enzymes in biological and chemical systems is to catalyze reactions, i.e. speed up a chemical reaction. The potential usage of many enzymes is hindered by low activity, specificity or other parameters that make them less efficient in industrial or biological processes. Therefore, the implications of systematically improving an enzyme for faster catalysis or less promiscuity are limitless.

Rational enzyme design is a developing and insightful method to gain knowledge and potentially very substantial results in enzyme or protein engineering approaches. It is much more cost-effective and resourceful than the many random mutagenesis approaches that are readily used by experimentalists. Within iGEM, with limited time and especially resources, it is very useful to integrate modelling with the project extensively. To do this, several different methods can be used to garner more understanding in productive binding modes, structural anomalies, steric clashes or other molecular mechanics related issues. These methods require a plethora of software and quite a bit of know-how to produce reliable and meaningful results, so attempting this should be done with caution.

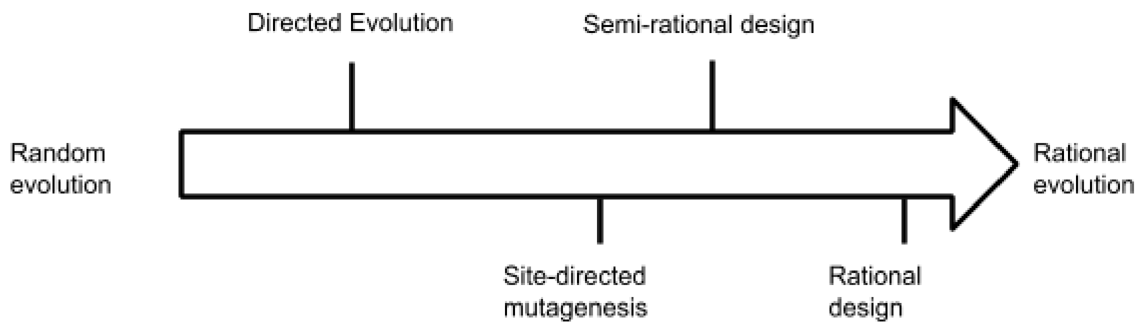
This “user manual” is directed towards overgraduate students that have some background in molecular mechanics or chemical equilibria problem. A good basic chemical intuition is highly recommended, as well as a good experience of biomolecules (structural biology) and biochemistry, especially if you are trying to model enzyme kinetics or even chemical reactions.

In short, rational enzyme design is a new, powerful tool that we think should be implemented in the iGEM competition at a bigger scale, to stress the importance of understanding your system on a molecular level, and seeing how those results correlate with the experimental results. It is a good and now easy way to elevate your iGEM project to a point where you have a complete theoretical and practical picture of your project.

How to evolve proteins

There are several ways of achieving the same goal - in this instance it is to evolve a new construct through mutations, but the methods differ in how “rational” they are, or how much our understanding of the system contributes to which mutations we in the end choose to make. As you can see, the least rational approaches of all human-made developments in protein/enzyme engineering is directed evolution (certainly actual evolution is the least rational approach (and most uncontrollable)). The more rational we become, the more we simplify the problem to an isolated, controllable system.

As we will see in a bit, even though many more parameters are taken into account in less controlled approaches, such as random mutagenesis, it can sometimes result that those other parameters than your protein of interest are developed instead. An example of this can be found in directed evolution, where if you apply a genetic drive on the system - introduce an antibiotic, for example - the host organism will find the best solution to the problem - not necessarily an improvement of your enzyme that you thought could be developed to degrade the antibiotic. It may already have a gene for an efflux pump, and evolution will drive it to that solution instead. This often happens if the original wild type protein is too far from the final, optimized mutant. Of course, you only discover this after months of expression and characterization.



Directed evolution is where one presents conditions under which wanted traits increase the survival rate of the organism producing the product, and thus, a genetic drive towards the wanted product is formed. Coupled with an induced higher mutational frequency, this is a powerful tool to drive a wanted development in a natural system.

Many semi-rational approaches focus on statistical analysis or bioinformatics to get a good guess about where to place the mutations, not why to place them on those locations, and certainly not to which residue it should be mutated. An example of this is Saturation mutagenesis, which can be done coupled with a generation analysis for great efficiency. Through generation analysis, one can look at similar enzymes within a class of organisms to, on a *sequence*, not a structure level, classify which regions of the sequence are hypervariable, highly conserved and variable throughout the ensemble. The hypervariable regions are often not of interest, and usually differ in the folding of random coils. The highly conserved regions on the other end of the spectra are highly conserved for a reason - the mutants that have mutations here have been deemed unfavorable by evolution. It is often in the variable regions that the most useful mutations can be introduced without disrupting activity, and possibly improving it. In saturation mutagenesis, you simply introduce completely or somewhat random mutations at selected (variable) sites in the sequence, and then screen or select for the most successful mutants.

Why rational enzyme design?

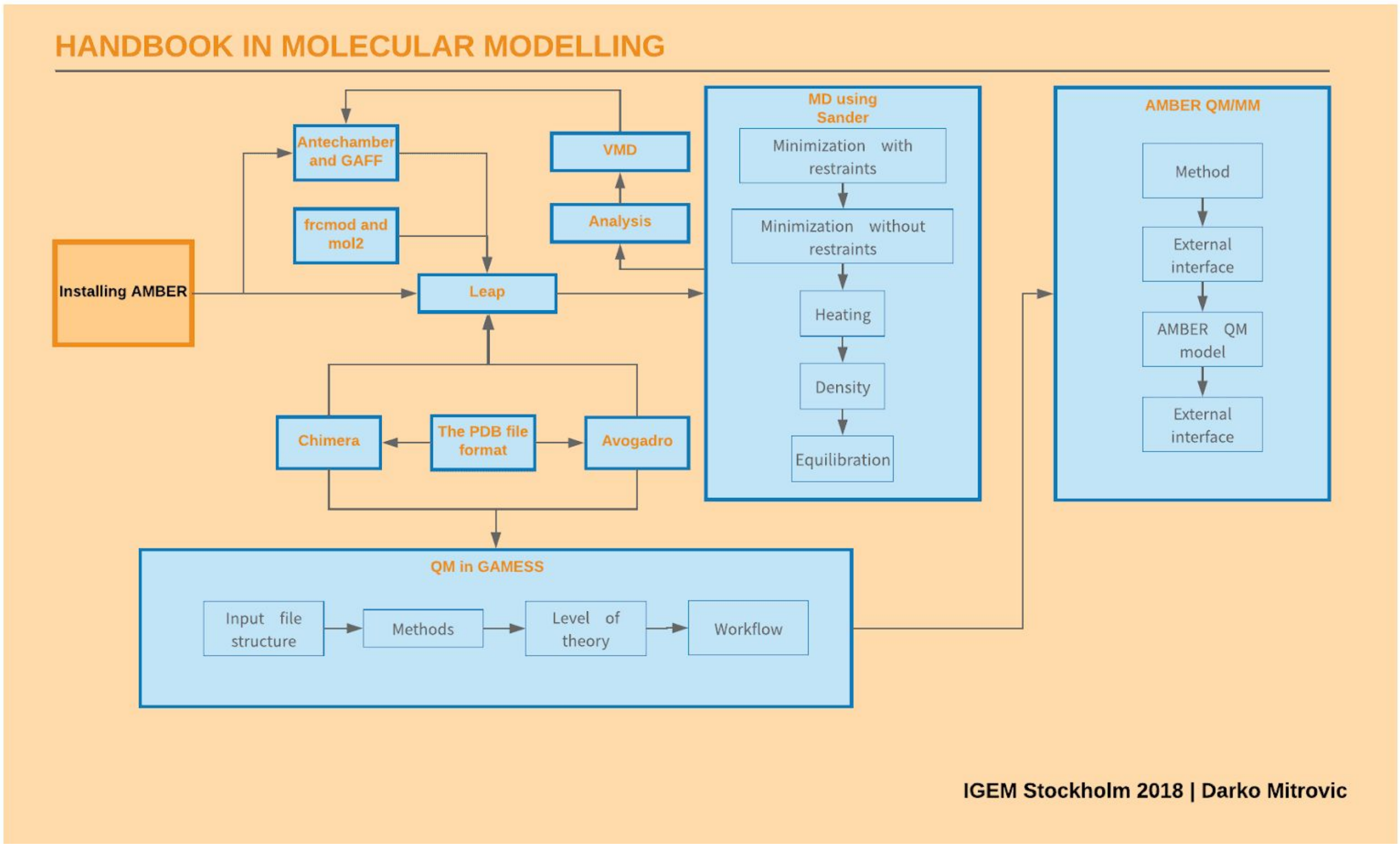
For rational enzyme design, we are often trying to understand the process which we mean to improve. It is therefore, naturally, the most logical approach to a problem. Theoretically, if we have full control and full understand. Practically, it is about approximating the system so that the error is minimized and the model system has the same parameters as the real system. This is important, since there is no fundamental flaw with rational enzyme design - it is theoretically consistent - however, the error arises from the models. If we did no approximations, then, we would have full control over the evolution of our system. Of course, we need to close off, isolate, reduce and use approximative models to get the parameters of interest. Therefore, in theory, rational enzyme design can be developed to be extremely accurate in the future, while both directed evolution and saturation mutagenesis have reached their limit of accuracy and effectiveness, and the problems presented in the previous section are inherent to the methods.

Secondly, all experimental work is very costly compared to modelling the system prior to testing. The costs and time for reagents, knowledge gathering, protocol optimization and equipment needed are very much larger than the computational costs. Of course it does take time to learn; however, the purpose of this user's manual is to alleviate future iGEM teams timewise and aid them in their modelling.

Method

The methods presented here are some of the most exact methods or approaches to rational enzyme design that exist to date. We apply all from classical to quantum mechanical approaches, including a QM/MM simulation (puts a quantum mechanical region of interest into a classically modelled environment). We aim to make the process of understanding your system as straightforward as possible. There are several depths to the modelling applied to solving a problem (1st row) using the methods (1st column), as you can see in the table:

	Optimization of Thermal stability	Optimization of Protein structure (e.g. folding)	Affinity/free energy of association	Enzyme catalysis (nuclear geometry change)	Expected Absorbance spectrum	Enzyme catalysis of electron transfer
Generation analysis	🟩	🟩	🟩	🟩	🟩	🟩
Crystal structure analysis	🟩	🟩	🟩	🟩	🟩	🟩
MD relaxation		🟩	🟩	🟩	🟩	🟩
MD simulation		🟩	🟩	🟩	🟩	🟩
QM/MM simulation			🟩	🟩	🟩	🟩
QM geometry optimization				🟩	🟩	🟩
QM reaction coordinate scan				🟩		🟩
QM saddle point calculation				🟩		🟩
QM TDDFT (excitation)					🟩	🟩
QM Electron transfer						🟩



INSTALLING AMBER

Throughout this handbook, AMBER, or ambertools, will be used as the main software for both QM/MM and MD simulations, and we will see how to use it in future chapters (different functionalities include LEaP, Antechamber and GAFF, and of course sander for the actual MD simulation). It contains the latest rendition of the Amber force fields, (ff14sb as of 2018), that is actually used in other softwares such as Gromacs. Before we can use AMBER, however, we need to install it. Despite how many programs contain automatic installation wizards, you have to do this manually for AMBER. Even though ambertools is only available for MacOS and Linux, this guide includes how to install it on Windows machines as well.

1. Introduction

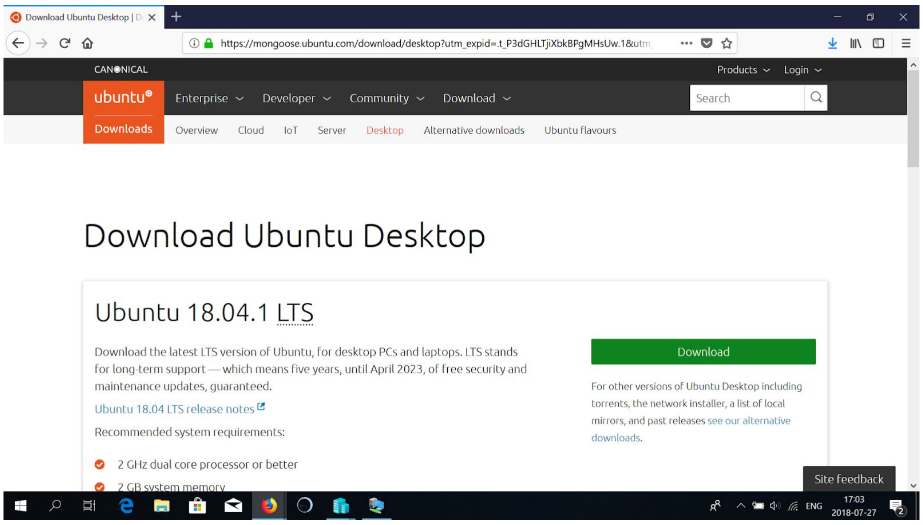
Amber is a very useful software tool for molecular mechanics related simulations and calculations. It mostly uses a classical level of theory, but is nevertheless very practical for large systems such as proteins, membranes or large polymer clusters that are simply to complex to be treated quantum mechanically. Amber is often the first step to take in modelling molecular mechanics, that paves the way for interesting result and a chance to get a very detailed view on the system one might be working on experimentally. The first step, outlined in this chapter, would be to download and install amber. Before we can do that, one should be aware of that there is no general user-friendly graphical interface for amber, but it is completely run through the terminal. Therefore, you should familiarize yourselves with the terminal. Some useful commands are:

cd <*argument*> - change directory to the path specified in <*argument*>
ls <*argument*> - lists all files and directories in the current location. Adding -s lists all sizes in kb and -a lists hidden items.
mkdir <*argument*> - creates a new directory named <*argument*>
nano <*argument*> *or* vim <*argument*> - opens a text file in the terminal, will be useful for quick editing of input files later.

2. Windows 10

There is currently no implementation of ambertools on Windows platforms; however, with Windows 10 Professional, Education or Enterprise, one can activate hyper-V, which is a virtual machine manager, where one can create an ubuntu environment ("a computer inside your computer"), and use the steps for Linux below. This is the most viable option for Windows users since many different programs that we will use in the future are in Ubuntu. The following requirements must be met:

- 64-bit Processor with Second Level Address Translation (SLAT).
- CPU support for VM Monitor Mode Extension (VT-c on Intel CPU's).
- Minimum of 4 GB memory.


https://mongoose.ubuntu.com/download/desktop?utm_expid=.t_P3dGHLTjiXbkBPgMHsUw.1&utm

Download Ubuntu Desktop

Ubuntu 18.04.1 LTS

Download the latest LTS version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years, until April 2023, of free security and maintenance updates, guaranteed.

Ubuntu 18.04 LTS release notes

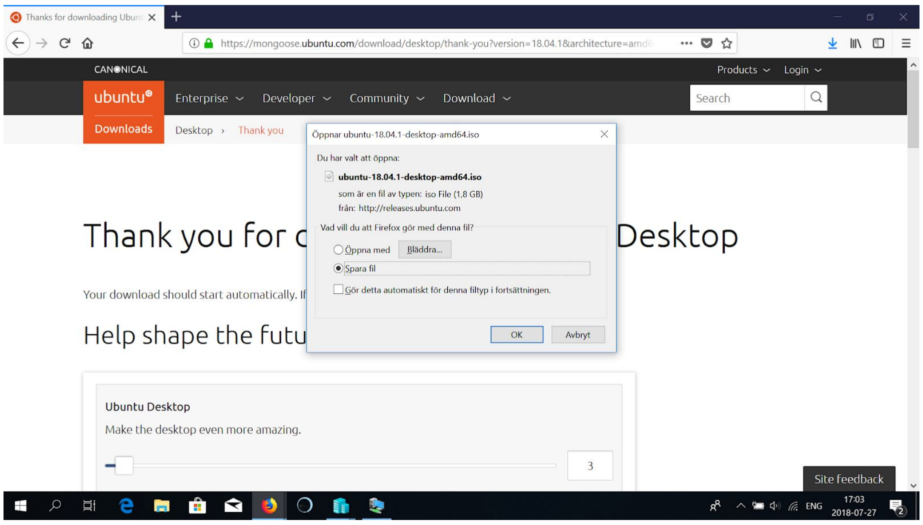
Recommended system requirements:

- 2 GHz dual core processor or better
- 2 GB system memory

Download

For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors, and past releases see our alternative downloads.

To get to this site, simply google Ubuntu download, and you should find an updated version of this file. Click Download.


https://mongoose.ubuntu.com/download/desktop/thank-you?version=18.04.1&architecture=amd

Thank you for ... Desktop

Your download should start automatically. If ...

Help shape the futu...

Öppnar ubuntu-18.04.1-desktop-amd64.iso

Du har valt att öppna:

ubuntu-18.04.1-desktop-amd64.iso

som är en fil av typen: iso File (1,8 GB)

från: http://releases.ubuntu.com

Vad vill du att Firefox gör med denna fil?

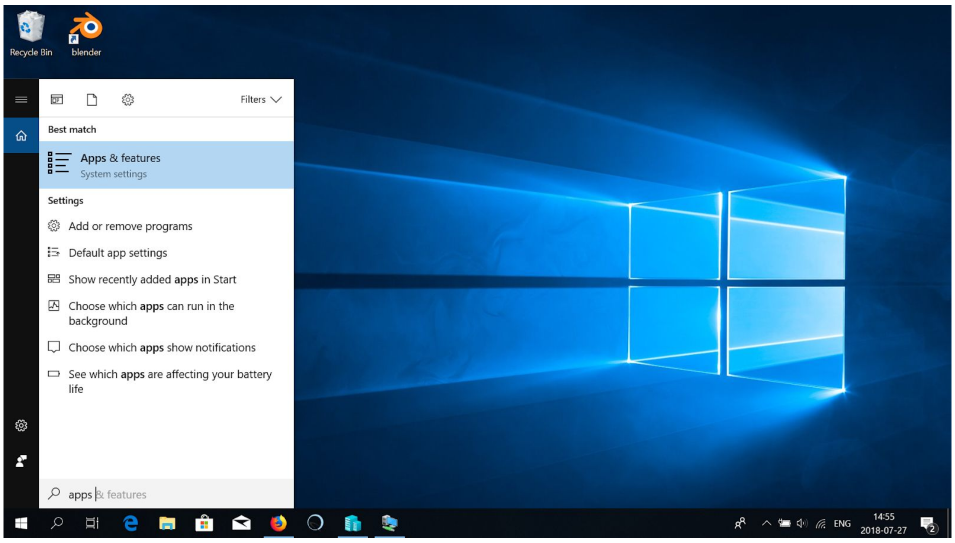
Öppna med [Bläddra...]

[✓ Spara fil]

Gör detta automatiskt för denna filtyp i fortsättningen.

OK Avbryt

Even though this is in Swedish, we can see that the file we are downloading is an iso file, or a mountable disk file. We will use this later.


Best match

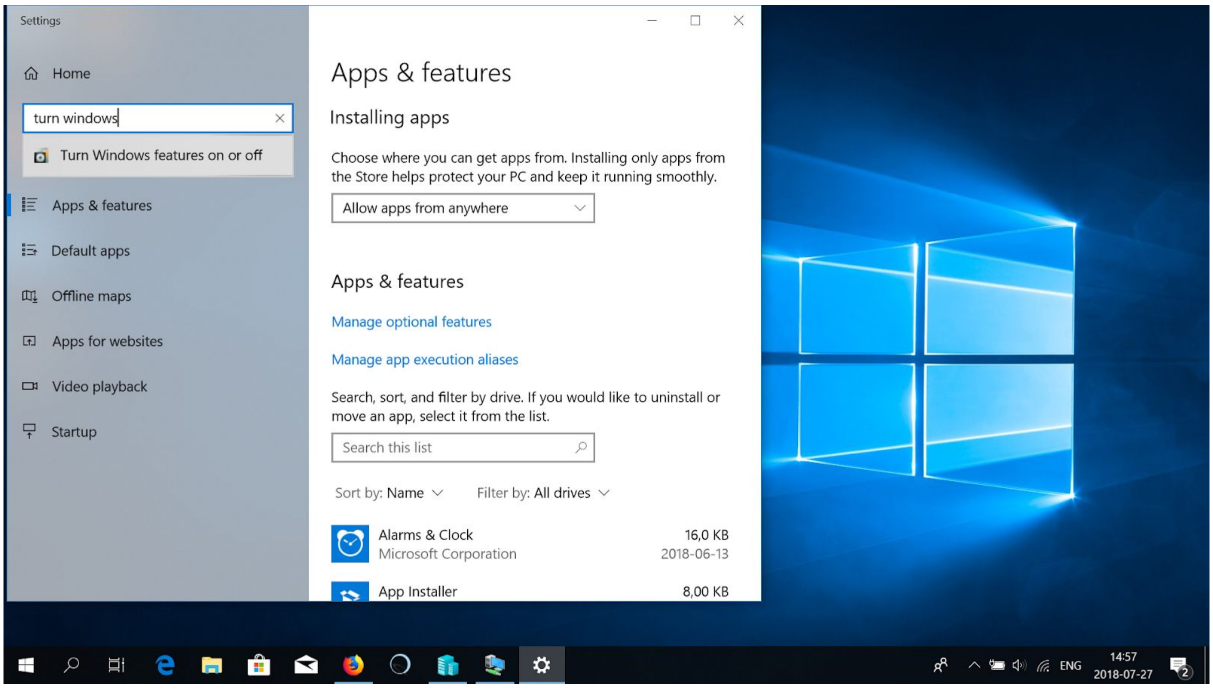
Apps & features
System settings

Settings

- Add or remove programs
- Default app settings
- Show recently added apps in Start
- Choose which apps can run in the background
- Choose which apps show notifications
- See which apps are affecting your battery life

apps & features

Go to Apps & features.



turn windows

Apps & features

Installing apps

Choose where you can get apps from. Installing only apps from the Store helps protect your PC and keep it running smoothly.

Allow apps from anywhere

Apps & features

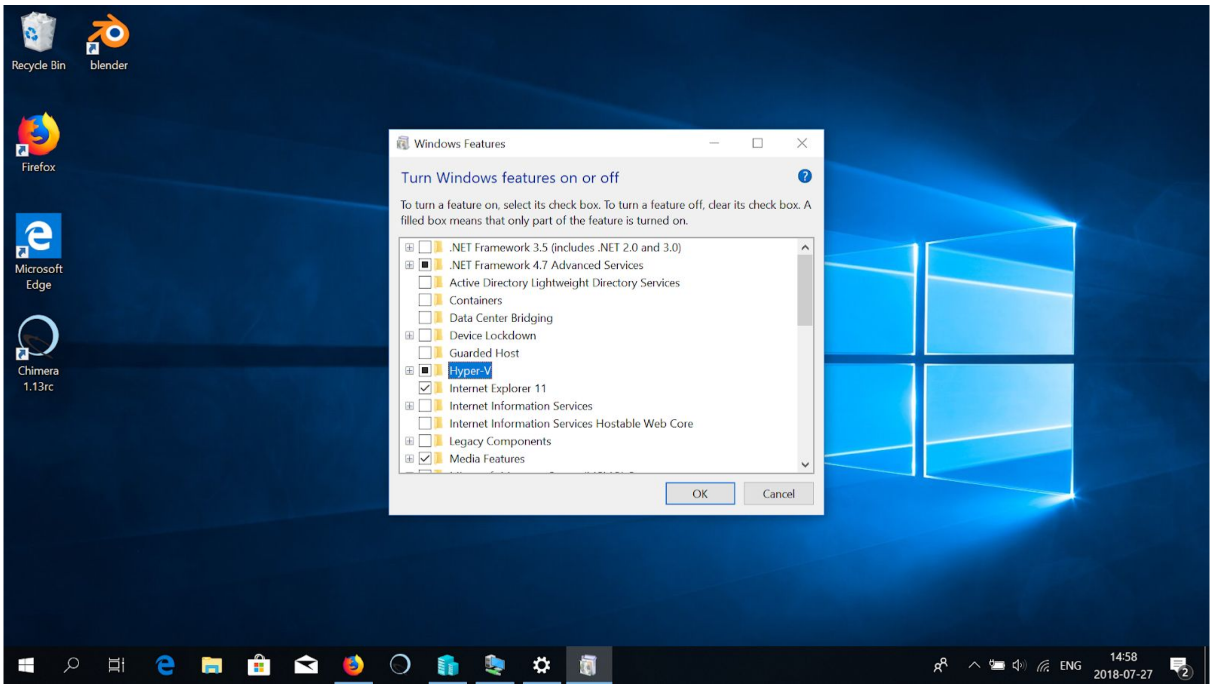
Search, sort, and filter by drive. If you would like to uninstall or move an app, select it from the list.

Search this list

Sort by: Name Filter by: All drives

Alarms & Clock	Microsoft Corporation	16,0 KB 2018-06-13
App Installer		8,00 KB

In the search field, type *Turn Windows features on or off*.



Turn Windows features on or off

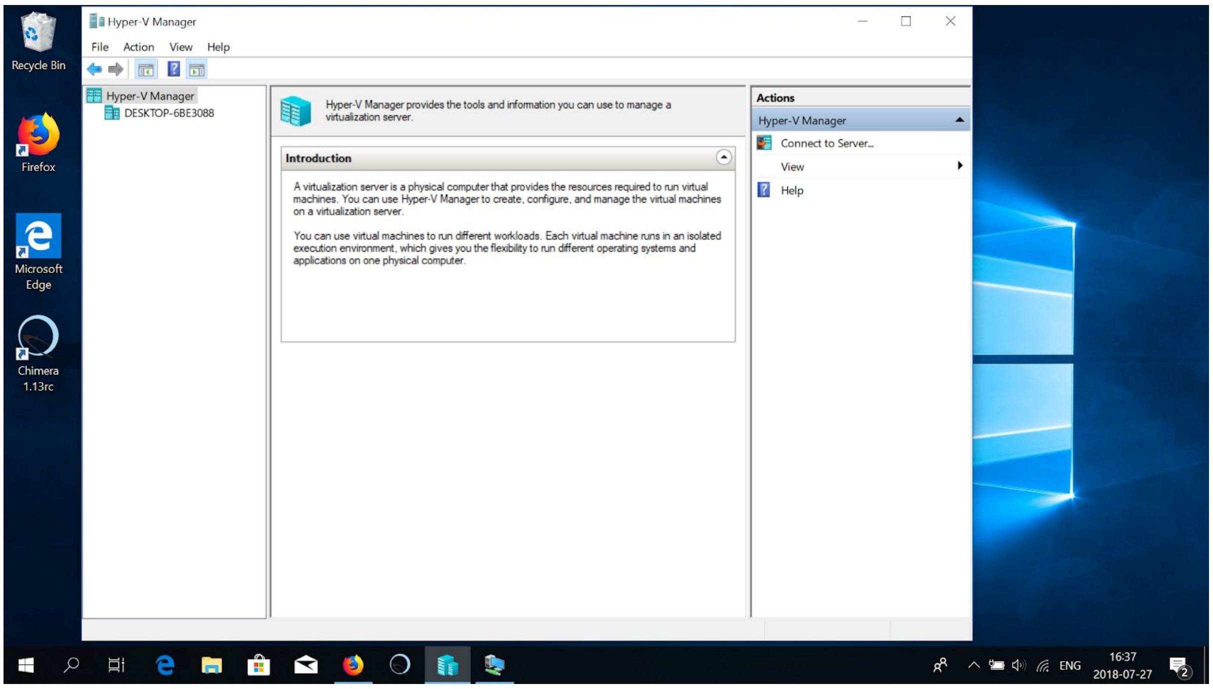
To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ☐ .NET Framework 3.5 (includes .NET 2.0 and 3.0)
- ■ .NET Framework 4.7 Advanced Services
- ☐ Active Directory Lightweight Directory Services
- ☐ Containers
- ☐ Data Center Bridging
- ☐ Device Lockdown
- ☐ Guarded Host
- ■ Hyper-V
- ✓ Internet Explorer 11
- ☐ Internet Information Services
- ☐ Internet Information Services Hostable Web Core
- ☐ Legacy Components
- ✓ Media Features

Select the Hyper-V option. It should look as it does on my screen above. Then, you need to restart your computer in order for this option to take effect and for this function to be activated.

3. Setting Up Ubuntu

The next step is to actually create a virtual machine. Open the hyper-V manager.

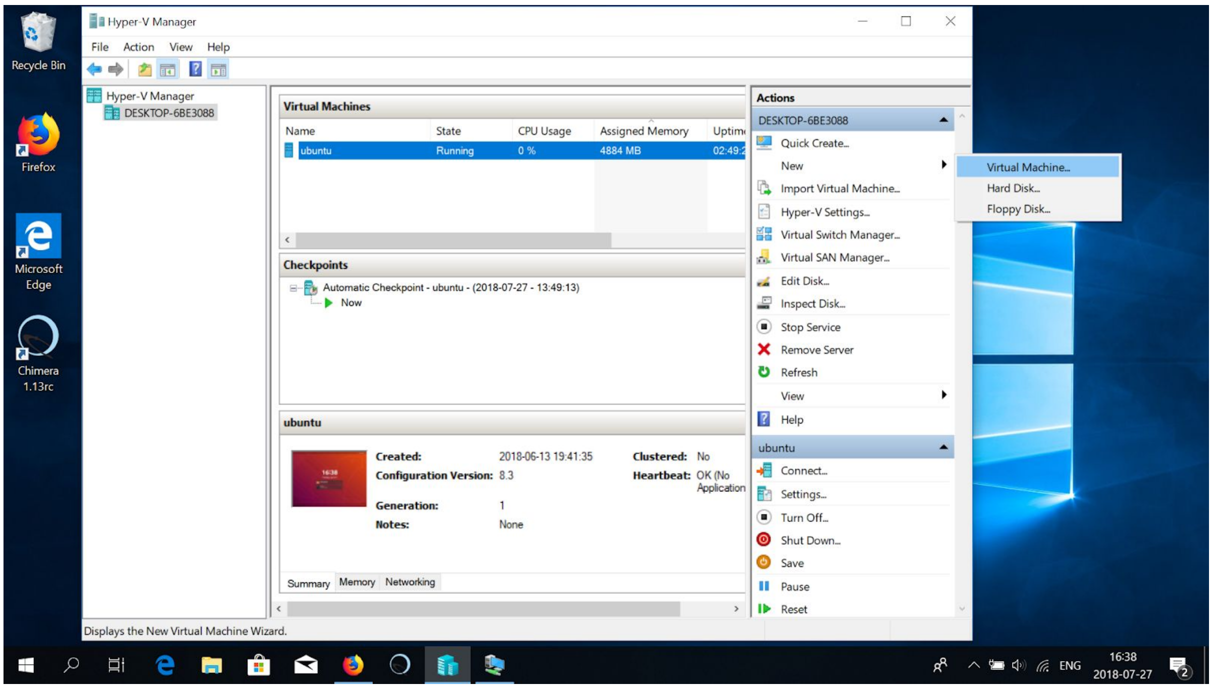


Hyper-V Manager provides the tools and information you can use to manage a virtualization server.

Introduction

A virtualization server is a physical computer that provides the resources required to run virtual machines. You can use Hyper-V Manager to create, configure, and manage the virtual machines on a virtualization server.

You can use virtual machines to run different workloads. Each virtual machine runs in an isolated execution environment, which gives you the flexibility to run different operating systems and applications on one physical computer.



Virtual Machines

Name	State	CPU Usage	Assigned Memory	Uptim
ubuntu	Running	0 %	4884 MB	02:49:

Checkpoints

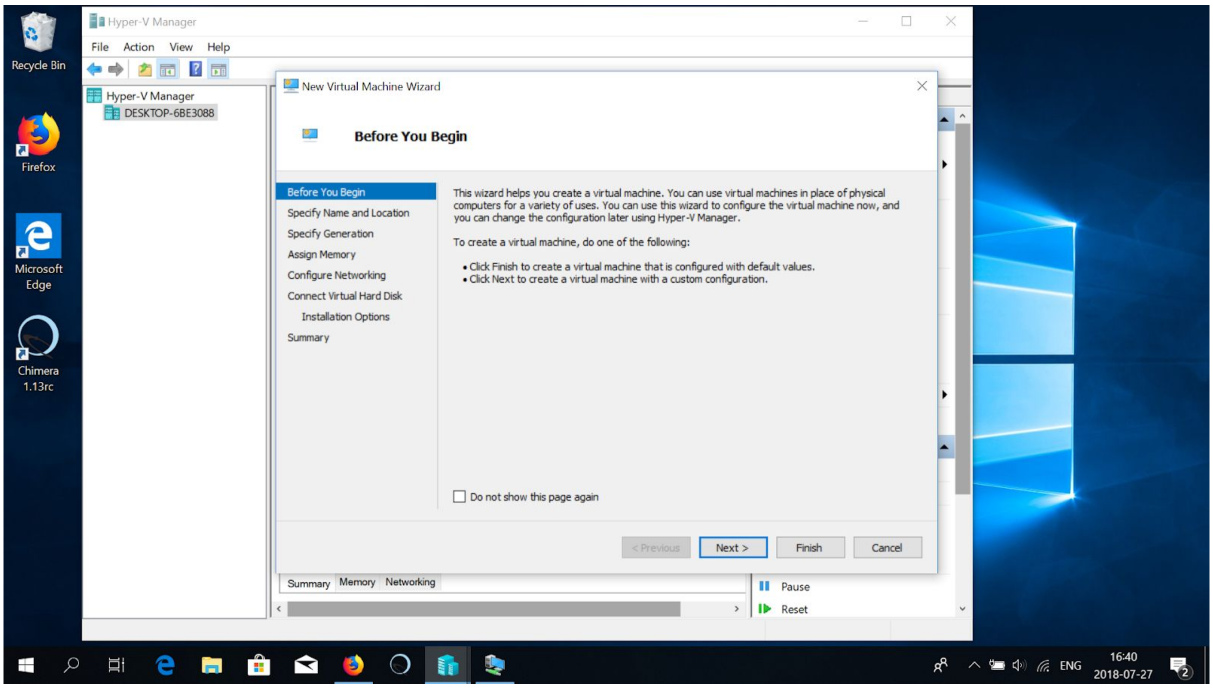
Automatic Checkpoint - ubuntu - (2018-07-27 - 13:49:13)
Now

ubuntu

Created:	2018-06-13 19:41:35	**Clustered:**	No
Configuration Version:	8.3	**Heartbeat:**	OK (No Application
Generation:	1		
Notes:	None		

Displays the New Virtual Machine Wizard.

We want to create a new Virtual Machine. Choose the highlighted option.



New Virtual Machine Wizard

Before You Begin

Before You Begin
Specify Name and Location
Specify Generation
Assign Memory
Configure Networking
Connect Virtual Hard Disk
Installation Options
Summary

This wizard helps you create a virtual machine. You can use virtual machines in place of physical computers for a variety of uses. You can use this wizard to configure the virtual machine now, and you can change the configuration later using Hyper-V Manager.

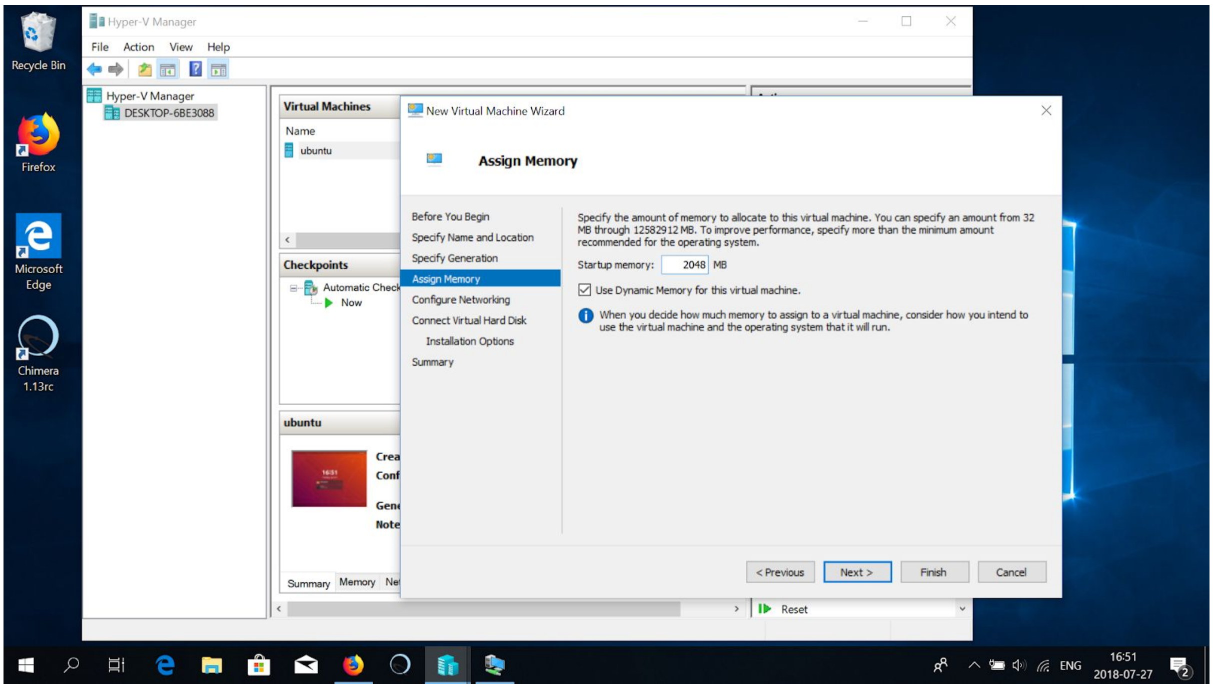
To create a virtual machine, do one of the following:

- Click Finish to create a virtual machine that is configured with default values.
- Click Next to create a virtual machine with a custom configuration.

☐ Do not show this page again

< Previous | Next > | Finish | Cancel

From here, we want to do a couple of things. Click next.



New Virtual Machine Wizard

Assign Memory

Before You Begin
Specify Name and Location
Specify Generation
Assign Memory
Configure Networking
Connect Virtual Hard Disk
Installation Options
Summary

Specify the amount of memory to allocate to this virtual machine. You can specify an amount from 32 MB through 12582912 MB. To improve performance, specify more than the minimum amount recommended for the operating system.

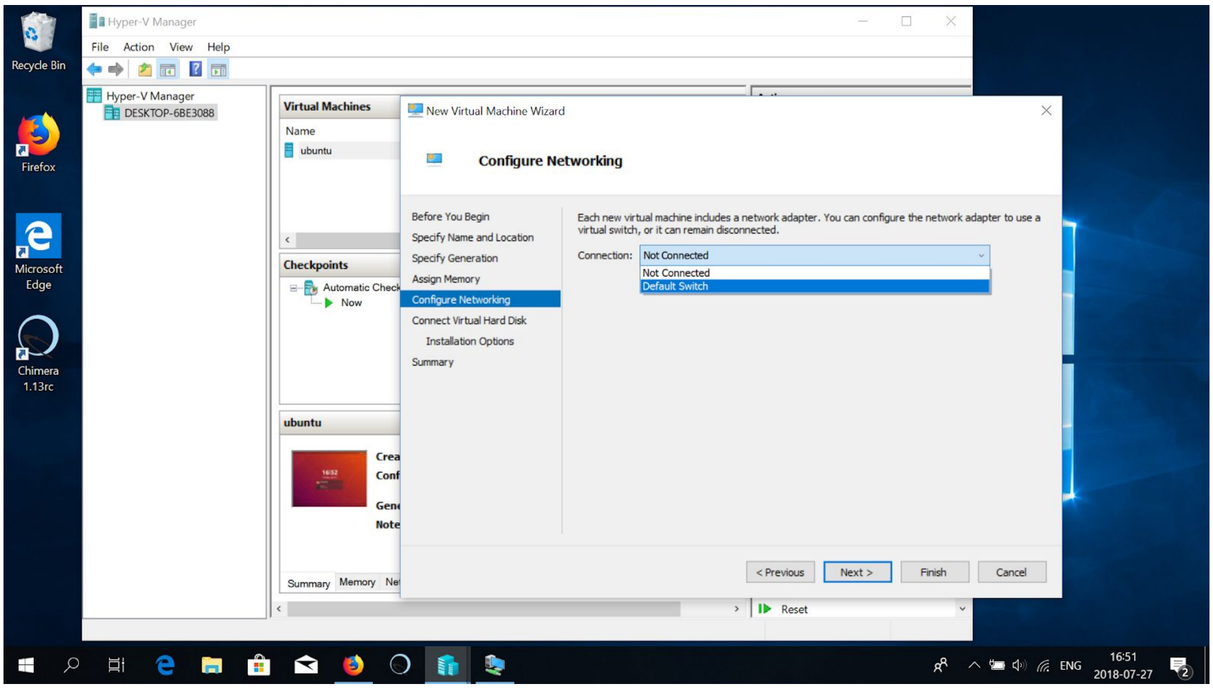
Startup memory: 2048 MB

☑ Use Dynamic Memory for this virtual machine.

When you decide how much memory to assign to a virtual machine, consider how you intend to use the virtual machine and the operating system that it will run.

< Previous | Next > | Finish | Cancel

Give the virtual machine a name, ignore the "specify generation" option, and go to assign memory. Here, you need to assign at least 2048 MB of RAM for Ubuntu to start up.

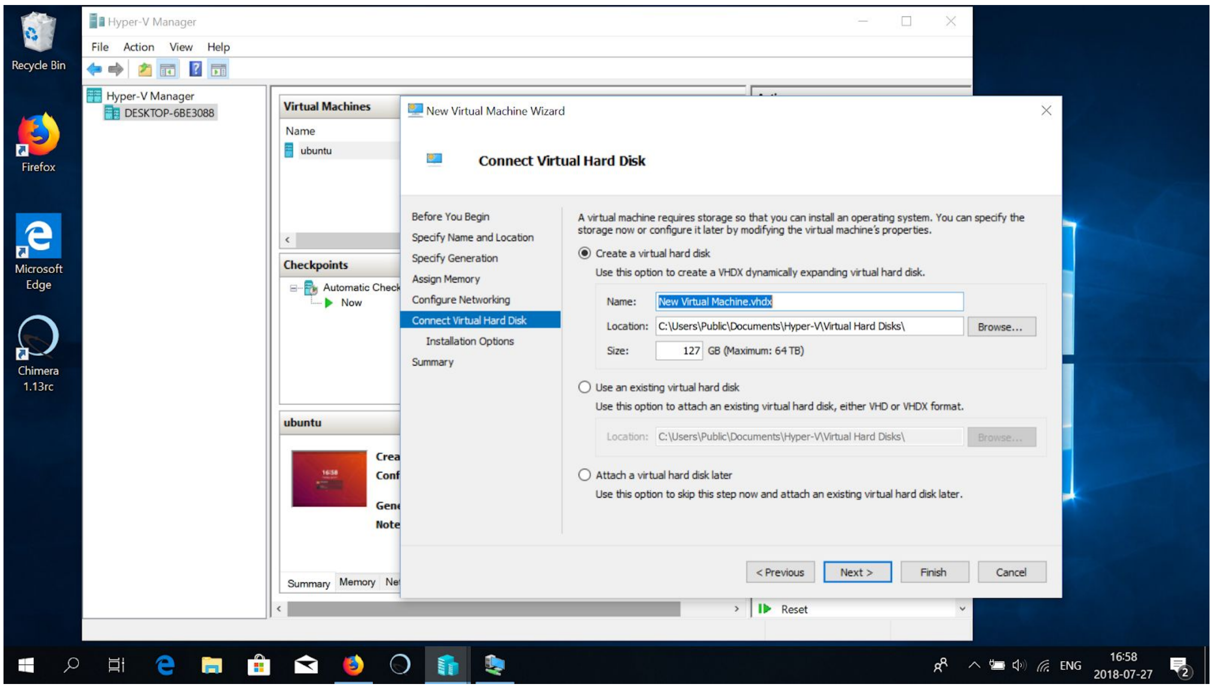


Configure Networking

Each new virtual machine includes a network adapter. You can configure the network adapter to use a virtual switch, or it can remain disconnected.

Connection: Not Connected
- Not Connected
- Default Switch

One Configure Networking, choose Default Switch that should suffice for our intents and purposes. If you are having connection problems, troubleshoot this Default Switch.



Connect Virtual Hard Disk

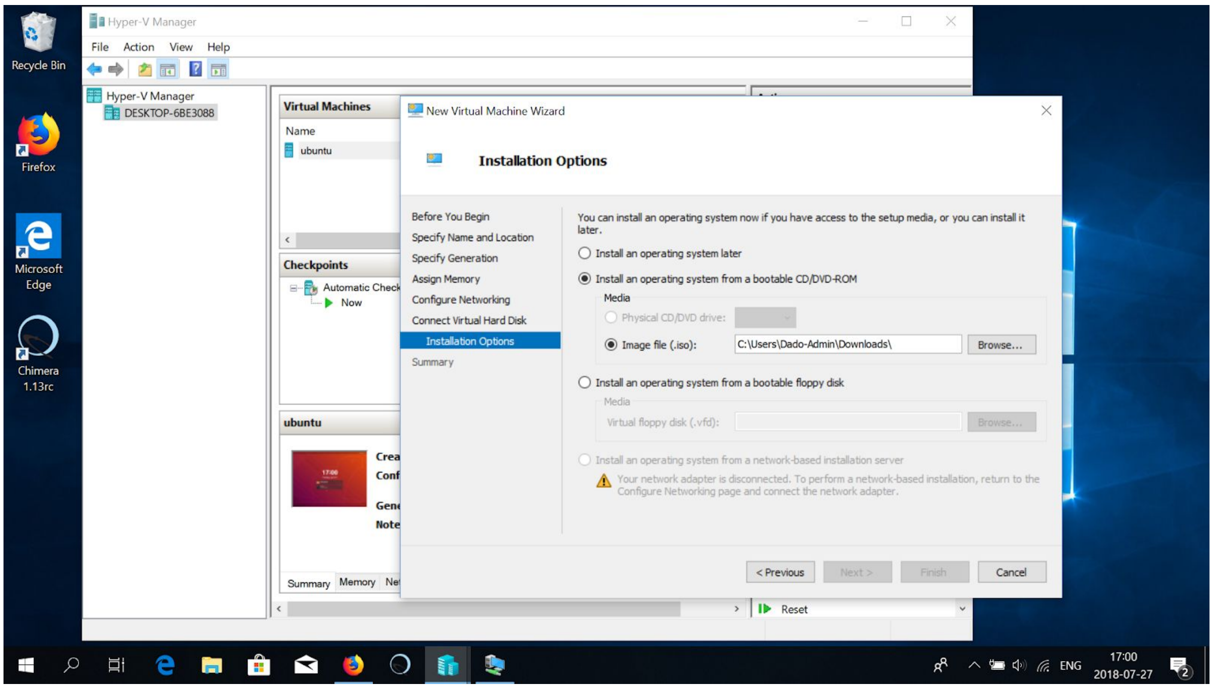
A virtual machine requires storage so that you can install an operating system. You can specify the storage now or configure it later by modifying the virtual machine's properties.

(●) Create a virtual hard disk
Use this option to create a VHDX dynamically expanding virtual hard disk.
Name: New Virtual Machine.vhdx
Location: C:\Users\Public\Documents\Hyper-V\Virtual Hard Disks\
Size: 127 GB (Maximum: 64 TB)

( ) Use an existing virtual hard disk
Use this option to attach an existing virtual hard disk, either VHD or VHDX format.
Location: C:\Users\Public\Documents\Hyper-V\Virtual Hard Disks\

( ) Attach a virtual hard disk later
Use this option to skip this step now and attach an existing virtual hard disk later.

Here, you can allocate the total hard disk size. Choose however much you can spare. I used 200GB.



New Virtual Machine Wizard

Installation Options

Before You Begin
Specify Name and Location
Specify Generation
Assign Memory
Configure Networking
Connect Virtual Hard Disk
Installation Options
Summary

You can install an operating system now if you have access to the setup media, or you can install it later.

( ) Install an operating system later
(●) Install an operating system from a bootable CD/DVD-ROM
Media
( ) Physical CD/DVD drive:
(●) Image file (.iso): C:\Users\Dado-Admin\Downloads\ Browse...
( ) Install an operating system from a bootable floppy disk
Media
Virtual floppy disk (.vfd): Browse...
( ) Install an operating system from a network-based installation server
Your network adapter is disconnected. To perform a network-based installation, return to the Configure Networking page and connect the network adapter.

< Previous | Next > | Finish | Cancel

Next, we need to install ubuntu on this virtual computer. We will use the file we downloaded earlier, .iso-file for Ubuntu. Then, you would have created the virtual machine itself. Simply double click it to start Ubuntu. From here, you can press finish and open the virtual machine by double clicking on it. Start it and follow the steps for ubuntu installation (everything is very self-explanatory there)

4. Linux (Ubuntu)

Simply googling ”amberXX download” suffices (where XX stands for the last two digits of the year). For this tutorial, we used ambertools 18, however not many crucial changes are done from update to update. There are two packages that can be downloaded - amberXX and ambertoolsXX. All the functionalities are implemented in the free version (ambertoolsXX), so that is the one we will use.

1. Upon Download, you will see that there is a *.tar.bz2* extension on the file. Unzip this file using the *tar -xvfj <argument>* command in the terminal. You have to change the working directory to the directory where the file is in your terminal using *cd*. If you are unsure of your location, use *pwd* or *ls* to navigate.
2. *Cd* into the directory called amberXX, and type:

   sudo apt-get install bc csh flex gfortran g++ xorg-dev zlib1g-dev libbz2-dev patch

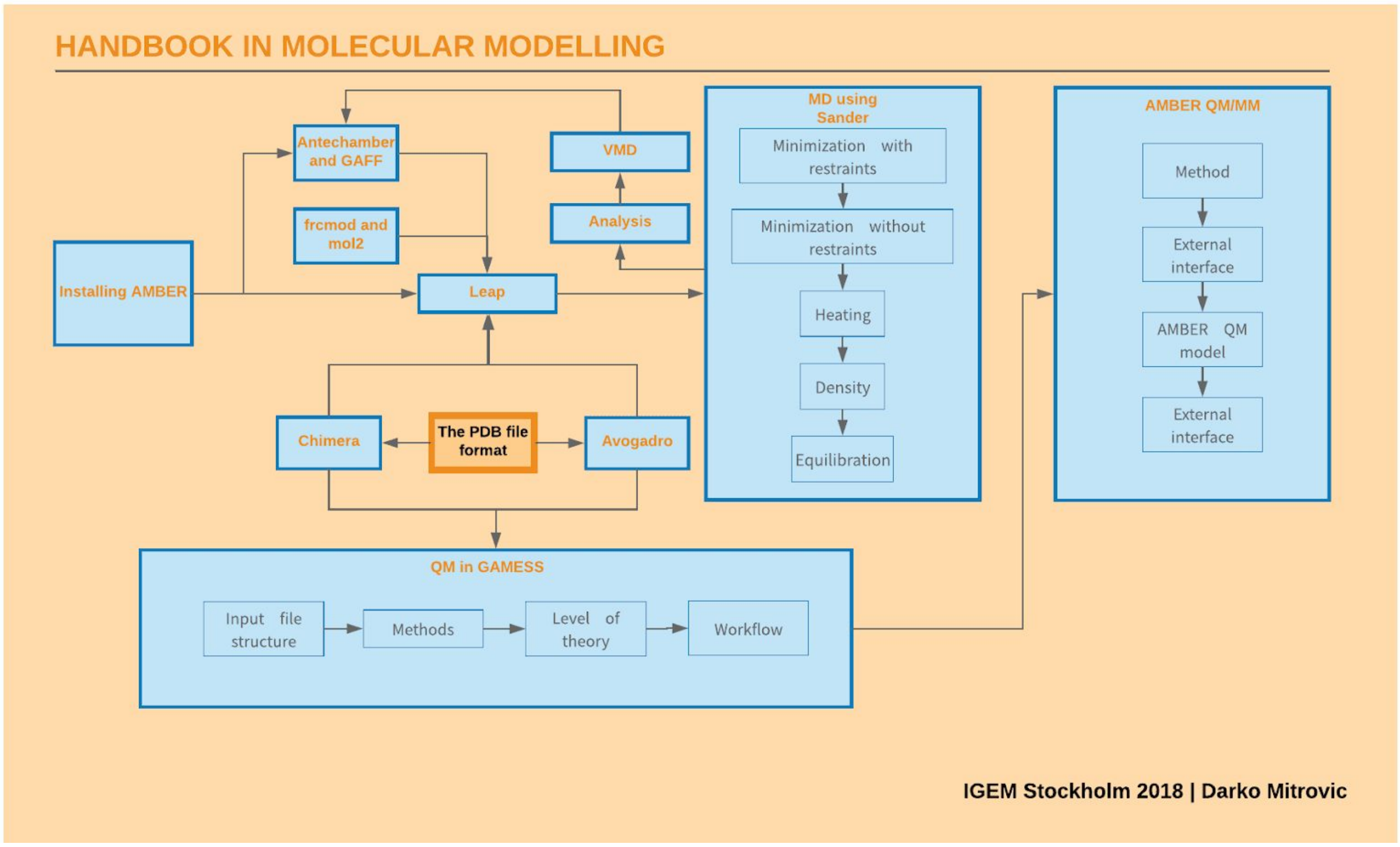
   Type your password when prompted, and answer *yes* to any questions.

3. Write *./configure gnu*, answer *yes* to any questions.
4. Write *source /home/<path_placeholder>/amberXX/amber.sh*

Note that the <path_placeholder> is to be replaced with the actual path to the amberXX directory (amberXX is also a placeholder and should be replaced with the name of your amber directory). This path can be easily found by the command *pwd*.

Sometimes, it is necessary to write this every time the terminal is opened or closed. (of course one can change the *.bashrc* file.

5. In the amberXX directory, type *make install*, and answer yes if prompted.
6. Now it’s ready to use!



The PDB file format

The PDB file is a powerful tool for viewing protein or nucleic structures. The format is tailored to fit polypeptides, but can cunningly be used in tandem with AMBER parameter files such as frcmod and mol2 to introduce organic molecules or inorganic complexes that are ready and parameterized. In this chapter, we will go through the file format briefly.

1. Introduction

The PDB file format is very useful at first glance, but a lot of manual editing could be necessary in future steps in Amber and especially the leap environment, where recognition of standard residues and non-standard residues entered by the user is crucial. While it may be user friendly in visualisation programs, it contains many different functionalities for separation of protein chains, recognition of residues and the CONECT records to make it easier for physicists making crystal structures to communicate with the structural biologists making models based on those crystal structures. In order to tamper with these functionalities to use in Leap, it is essential to understand them. While PDB files are designed for proteins, they can be used (with limited success most of the time) on any molecular structure.

2. Structure and linearity

The basic structure of the PDB file is:

I. Title
II. Secondary Structure record
III. Atom list
IV. CONECT records

These objects are arranged in a very long list, containing all information necessary for e.g. Chimera to visualise a correct and informative structure, including bonds and etc. In short, the title is trivial, the Secondary Structure record determines how the ribbon visualisation option will look, and also has some implications if rotamer libraries are applied. The Atom list contains the coordinates of all atoms, which residues they are linked to and information on the element type (C, H, O, etc). Lastly, the CONECT records specify which atoms are bonded to which.

Additionally, an important note to make is the linearity of the atom list. I.e. you cannot reorder residues or atoms that are not inherently in order. having residue 101,102,104,103 in this order is illegal.

3. Secondary Structure record

The Secondary structure record contains all information that is necessary for a visualisation program to know where to form Helices and Sheets in the 3D-structure of the protein of interest. The first column signals the type of secondary structure, either HELIX or SHEET. The subsequent columns contain the residues that are contained within this structure.

4. CONECT record

The CONECT record is very important to keep track of. For leap, it is best to simply not include it (for complex structures with many non-standard residues incorporated in the protein backbone.) These records work in both ways, i.e. if atom number 780 is bonded to 782, then the connect record would contain two separate accounts of this bond; CONECT

780 782 and CONECT 782 780. You can manually specify additional bonds if they are important in visualisations. Glycosylations, for example, are usually not bonded in the crystal structure but it can be good to bond them in the PDB file.

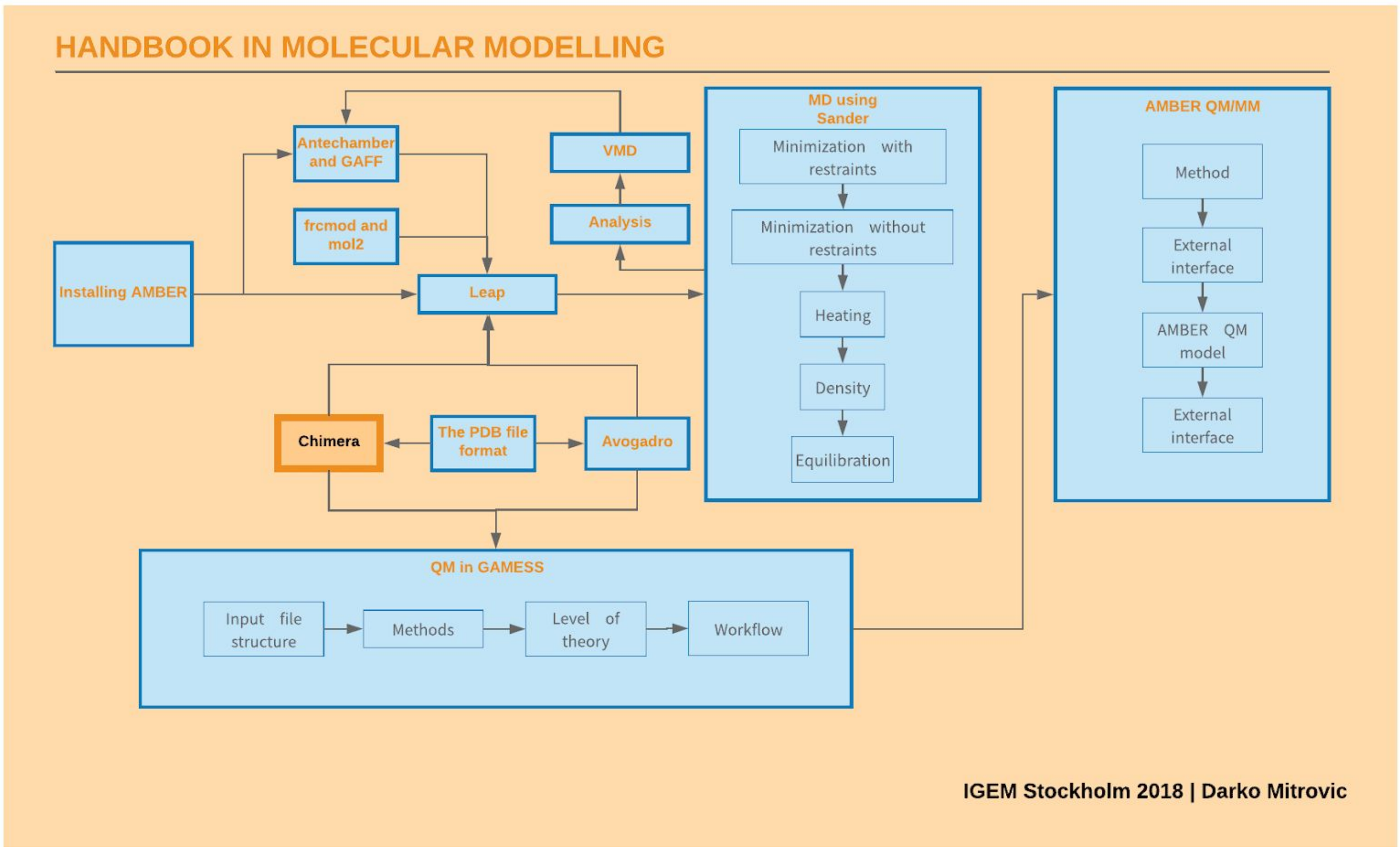
5. Chains, residues and atom numbers

The atom list is the bulk of the PDB text file, and contains all coordinates in XYZ-format of all the atoms. The columns represent:

I. HETATM or ATOM, the type of atom. HETATM is only used for non-protein residues or molecules. ATOM is the default atom type that can be expected to be in a protein.
II. Atom number. The number of the atom. Even though, residues can through errors be renumbered, atom numbers are very consistent throughout the whole procedure.
III. Atom name. While the name has no physical meaning, it is very important the same atom name does not occur twice within the same residue. Also, NEVER change the atom name of a standard residue.
IV. Then comes the residue name and number, that are fairly self-explanatory. Just remember the linearity
V. Then comes the X, Y and Z coordinate in three respective columns
VI. Then comes two columns that aren't very important. They should have the values 1.00 and 0.00.
VII. The last column signals which element the atom in question is.

6. TER cards

TER cards are a marker that signal the end of one chain, and the beginning of another. For several separated molecules, it is crucial to include a TER card for recognition of the N- and C-termini in LEaP, whereby it will add any missing atoms (often times just hydrogens, but sometimes it adds OXT, an oxygen on the carboxyl group on the C-terminal). If a TER card is missed, the backbone bonds will simply continue which will cause huge problems in further simulations. Usually, TER cards are automatically included when saving files in Chimera, or most of the time when downloading the files from RCPDB.



CHIMERA

Chimera is a great visualisation program for proteins (and exclusively proteins!), that is the leading visualisation and protein editing program thanks to the tailored routines to the PDB file format. A good understanding on the PDB file format in tandem with Chimera can make protein editing an absolute breeze. It is also a very good tool for making nice pictures and adding effects. It can also be exported in various formats (although not as many as VMD).

1. Introduction

Chimera is a must-have tool for anyone that will attempt to understand any molecular functionality of proteins. It is an easy tool for adding solvents, visualising PDB-files without complete CONECT-records, and is very good at recognizing protein structures and differentiating them from any anomalies such as prosthetic groups, glycosylations or ligands. This is driven both by the naming of residues (ALA, GLY, etc.) and the ATOM and HETATM type along with the TER card. Before you look at this, a prerequisite would be to really have a good look at the PDB-file. It is also good to understand what you gain with a PDB file, and especially what you loose with it and what information it inherently lacks.

Anyway, Chimera is very useful for viewing both your input and output structures, even though VMD or Avogadro can sometimes to it better. For visualisation, chimera is one of the best tools due to its' emphasis on display settings and the many ways you can change the appearance and colour of virtually every single unit (atoms, bonds), and even add very convincing shadows or ambient occlusion effects.

2. Mutation

Especially for rational enzyme or protein design, mutations are crucial. In Chimera, there is a tool called rotamers, where mutations on standard residues can be made to incorporate non-standard residues; however, it first requires that you select the residue to be mutated. The atom specifier tool is very useful in this. The syntax for selection of residues, atoms and models is:

# - model
: - Residue name or number
@ atom number

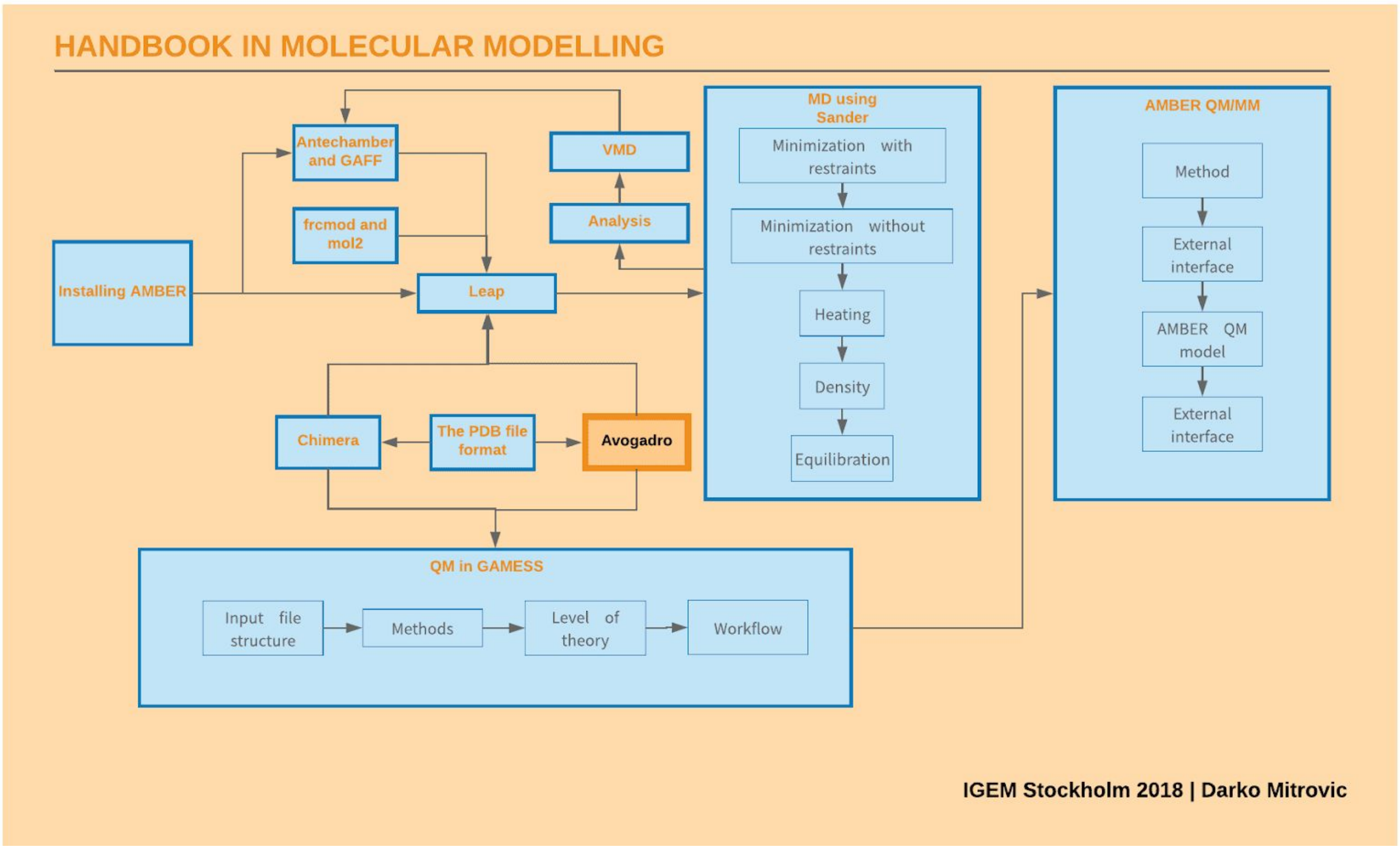
Only the residue selection syntax is relevant for mutagenesis of a single residue. Using an arbitrary rotamer library, one can see what would be the expected positioning of the new residue. Look out for any steric clashes.

3. Alignment

If you have to very similar models, you can align their structures according to different parameters, for example backbone or similar. It is useful in generational analysis where several crystal structures are available to get more useful information and intuition in how the structures differ.

In order to do this, open the model panel and scroll down until you see the align option. Select the models you would like to align over each other, and press apply. A new screen will appear where you can choose if you only want different parts of the protein to line up, for example active domains or only the binding domain or other interesting protein motifs.

Before you save a pdb-file containing different models, it is important to merge these models using the combine command on the command line, or simply select the copy/combine option on the model panel.



AVOGADRO

Avogadro, similar to Chimera or VMD, is a good visualisation program. Whereas Chimera can be used for protein editing and structural manipulation, Avogadro is used in these guides for atomic editing. It can also be used for orbital visualisation, and is the go-to program after using QM programs (in either .cube or .fchk format).

1. Introduction

Avogadro is a very useful tool for both analysis and advanced building that is much harder to do in other visualization program. In fact, Avogadro can be used, not only make structures or analyse orbital conformations, but also for input generation for a plethora of Quantum Mechanical based programs, including Gaussian, Q-chem, NWChem and GAMESS-UK. While the interface is quite self-explanatory, it could be good to clarify a few basic functions. Here, we will go through how to visualize orbitals, build structures and exporting it into pdb-format.

2. Building structures and generating input

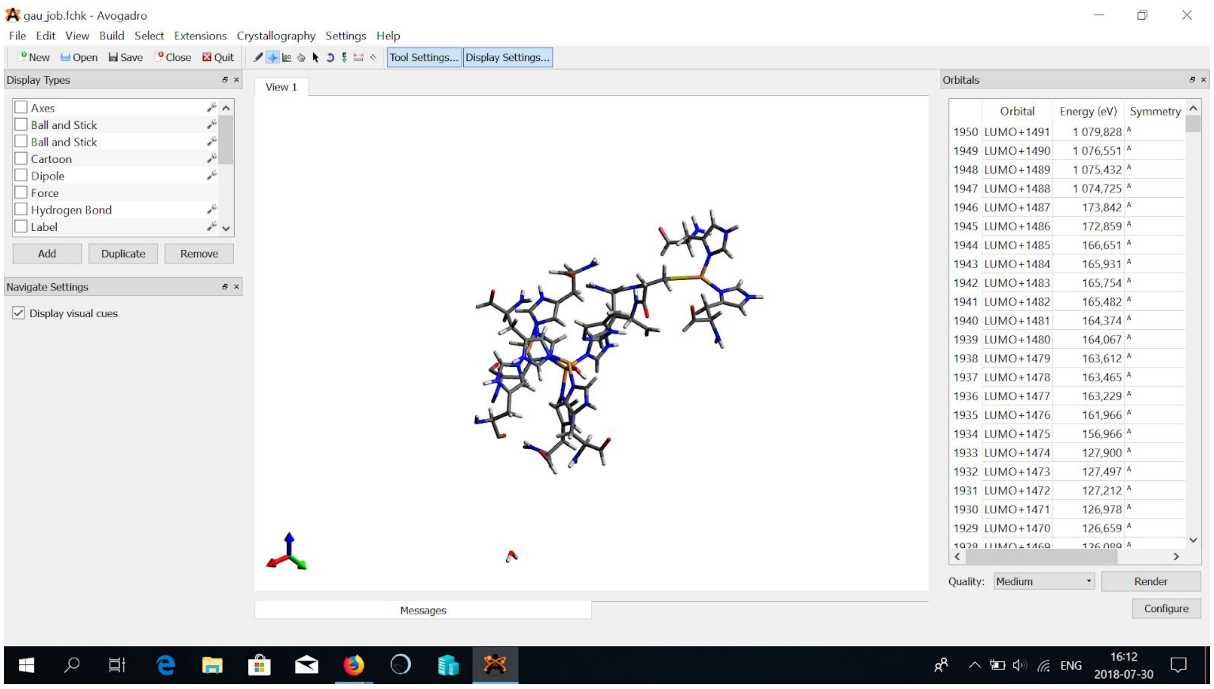
Most input structures from pdb lack hydrogens, which makes it crucial to add them before any simulation or actual calculations are performed. Avogadro does this much better than, say chimera *for small molecules*, both inorganic and organic. If you, say, want to dock a substrate in chimera, it is easier to chemically modify a molecular file in avogadro and then export it to pdb-format. The function *add hydrogens* is very useful in this instance. It is important to check chirality or other important factors such as protonation state for certain residues. Avogadro treats every chemical bond as a covalent bond, and assumes charges=0. Additionally, using the drawing tool, you can add virtually every element at an arbitrary position, define new bonds and redefine existing structures, although a good starting structure is recommended. After this, it is always useful to perform a geometric optimization in, say, GAMESS or NWChem.

In the *Extensions* section, you can choose which program to make an input file for. While there are options that you can choose for every, many functions are absent from this GUI, and manually editing the input file grants more possibilities - even logic trees in NWChem. I personally only use this function as a good way of getting the geometry of the initial structure into a XYZ-format. If you want, you can use a Z-matrix format as well (there is usually a box for that on the GUI)

3. Orbital and structural analysis

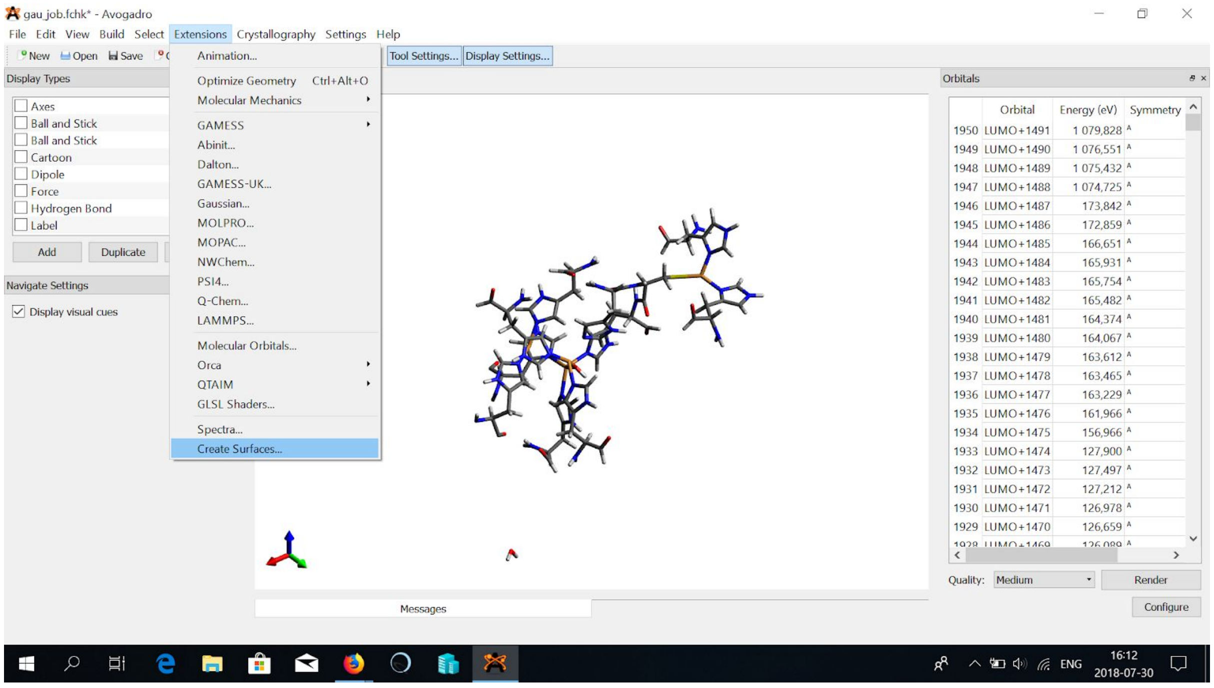
If you have a molecular file, chances are it can be opened in Avogadro. Chimera has some very weird issues with opening e.g. mol2-files, which we will use later (see LeAP-section). In short, simply find your file and open it in Avogadro. There are some graphical options that could be changed if one would be interested - under Display Settings… you can toggle Display Types, which is an additional window that will be visible on the left side of the screen, where you can change how the molecule is displayed. This is very useful for visualisation.

If you have orbital cube information stored in your file, e.g. .cube or .fchk from gaussian or NWChem, these files can also be opened by Avogadro. To visualise the Molecular Orbitals themselves, follow these steps:



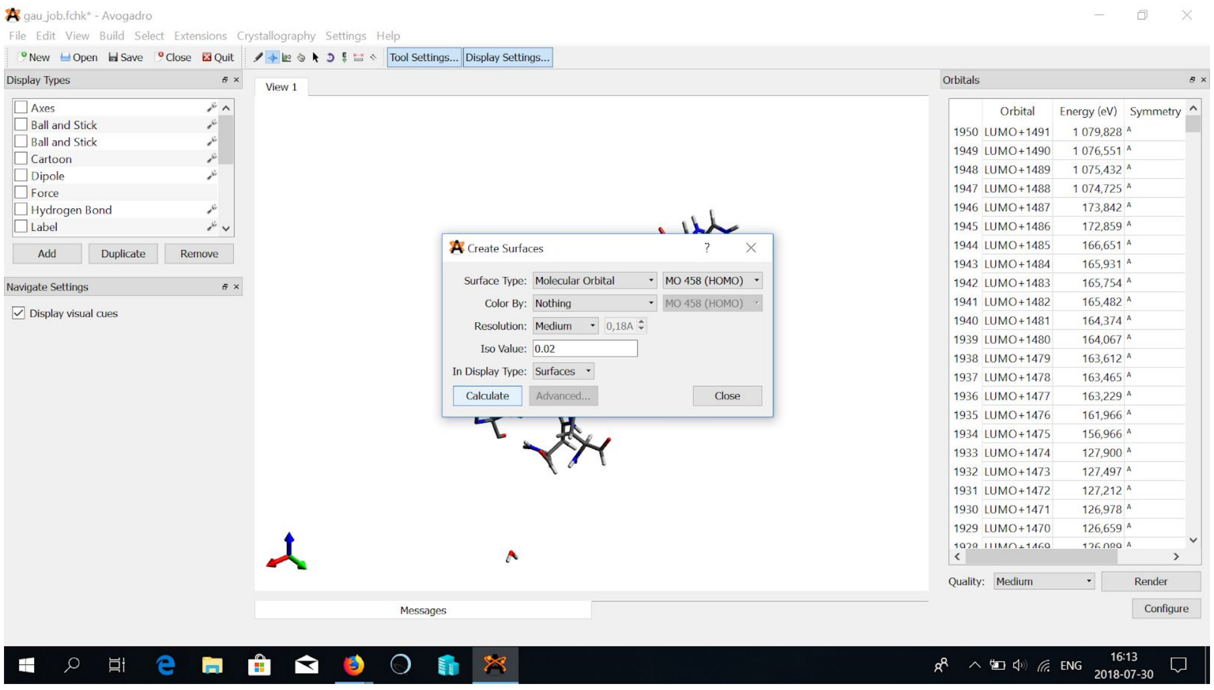
	Orbital	Energy (eV)	Symmetry
1950	LUMO+1491	1 079,828	A
1949	LUMO+1490	1 076,551	A
1948	LUMO+1489	1 075,432	A
1947	LUMO+1488	1 074,725	A
1946	LUMO+1487	173,842	A
1945	LUMO+1486	172,859	A
1944	LUMO+1485	166,651	A
1943	LUMO+1484	165,931	A
1942	LUMO+1483	165,754	A
1941	LUMO+1482	165,482	A
1940	LUMO+1481	164,374	A
1939	LUMO+1480	164,067	A
1938	LUMO+1479	163,612	A
1937	LUMO+1478	163,465	A
1936	LUMO+1477	163,229	A
1935	LUMO+1476	161,966	A
1934	LUMO+1475	156,966	A
1933	LUMO+1474	127,900	A
1932	LUMO+1473	127,497	A
1931	LUMO+1472	127,212	A
1930	LUMO+1471	126,978	A
1929	LUMO+1470	126,659	A

Open your structure file. Along with this, a list containing some HOMO's and LUMO's should also appear. Orbitals from HOMO-2 to LUMO+2 with a standard iso-value of low quality will be automatically rendered upon opening the file, so it may take a bit more time than usual.



	Orbital	Energy (eV)	Symmetry
1950	LUMO+1491	1 079,828	A
1949	LUMO+1490	1 076,551	A
1948	LUMO+1489	1 075,432	A
1947	LUMO+1488	1 074,725	A
1946	LUMO+1487	173,842	A
1945	LUMO+1486	172,859	A
1944	LUMO+1485	166,651	A
1943	LUMO+1484	165,931	A
1942	LUMO+1483	165,754	A
1941	LUMO+1482	165,482	A
1940	LUMO+1481	164,374	A
1939	LUMO+1480	164,067	A
1938	LUMO+1479	163,612	A
1937	LUMO+1478	163,465	A
1936	LUMO+1477	163,229	A
1935	LUMO+1476	161,966	A
1934	LUMO+1475	156,966	A
1933	LUMO+1474	127,900	A
1932	LUMO+1473	127,497	A
1931	LUMO+1472	127,212	A
1930	LUMO+1471	126,978	A
1929	LUMO+1470	126,659	A

Often, these automatically rendered orbitals aren't nearly good enough to visualise the full extent of the calculation, so generating custom orbitals is usually more informative. Select Create Surfaces… under Extensions.



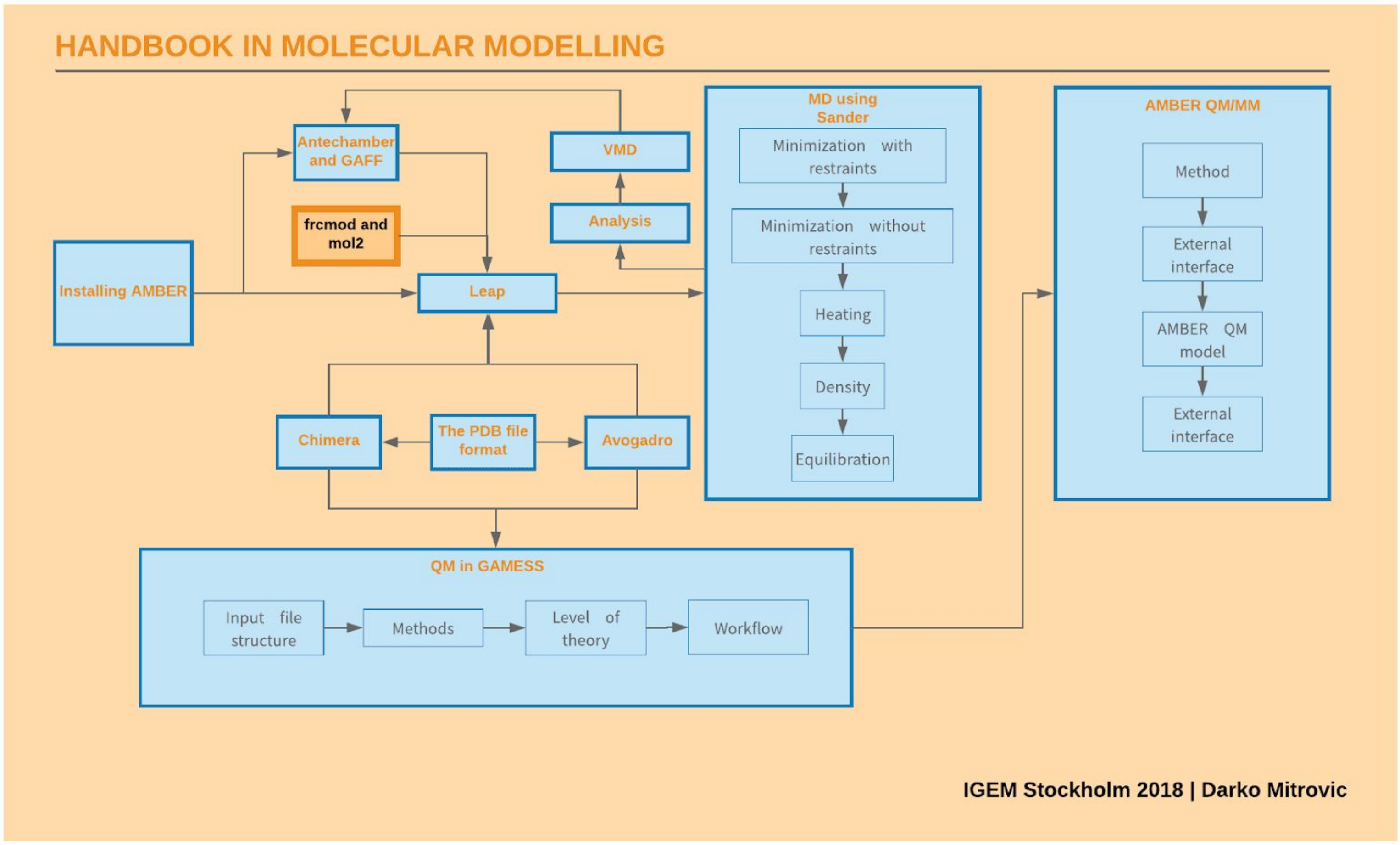
Create Surfaces

Surface Type: Molecular Orbital — MO 458 (HOMO)
Color By: Nothing — MO 458 (HOMO)
Resolution: Medium — 0,18A
Iso Value: 0.02
In Display Type: Surfaces

Calculate — Advanced... — Close

	Orbital	Energy (eV)	Symmetry
1950	LUMO+1491	1 079,828	A
1949	LUMO+1490	1 076,551	A
1948	LUMO+1489	1 075,432	A
1947	LUMO+1488	1 074,725	A
1946	LUMO+1487	173,842	A
1945	LUMO+1486	172,859	A
1944	LUMO+1485	166,651	A
1943	LUMO+1484	165,931	A
1942	LUMO+1483	165,754	A
1941	LUMO+1482	165,482	A
1940	LUMO+1481	164,374	A
1939	LUMO+1480	164,067	A
1938	LUMO+1479	163,612	A
1937	LUMO+1478	163,465	A
1936	LUMO+1477	163,229	A
1935	LUMO+1476	161,966	A
1934	LUMO+1475	156,966	A
1933	LUMO+1474	127,900	A
1932	LUMO+1473	127,497	A
1931	LUMO+1472	127,212	A
1930	LUMO+1471	126,978	A
1929	LUMO+1470	126,659	A
1928	LUMO+1469	126,089	A

Here, you can choose the orbital of choice, and the surface type to Molecular orbital. This function creates an iso-surface of the electron probability, so changing the iso-value changes the size of these surfaces.



FRCMOD AND MOL2

The frcmod and mol2 file formats are very useful in the process in getting from an initial prepared PDB file from Chimera or Avogadro, to a processed and ready file for Leap. This chapter is essential for anyone who is in need of parametrization of nonstandard organic residues, but also valuable for any QM-parametrization that you would like to do yourselves.

1. Introduction

The frcmod and mol2 files are very special files. They are more than the PDB-file associated with the same structure, in the way that they are the connection between the coordinates and the parameters for the structure, i.e. atom types (mol2) bond types (mol2) and bond parameters (frcmod) as well as nonbonded parameters (frcmod). They will be used readily in the MD simulations and QM/MM simulations in Amber, and are used in tandem with the existing gaff.dat data files in the parameter libraries pre-existing in amber.

2. mol2

mol2 is a more interesting file format than PDB. Instead of only containing coordinates and secondary structure information, the mol2 file also includes atom types (not only elements, but also how they are connected (bond order) and how they are hybridized (as in organic chemistry, the hybridization of atomic orbitals in this way is nonsensical in quantum mechanics). nr, for example, is a Nitrogen atom in an aromatic structure. These types determines parameters mulliken charges and van der waals radii, i.e. nonbonded parameters, but are also used to define and categorise different types of bonds. A n-h and a nr-h bond have very different parameters, for example).

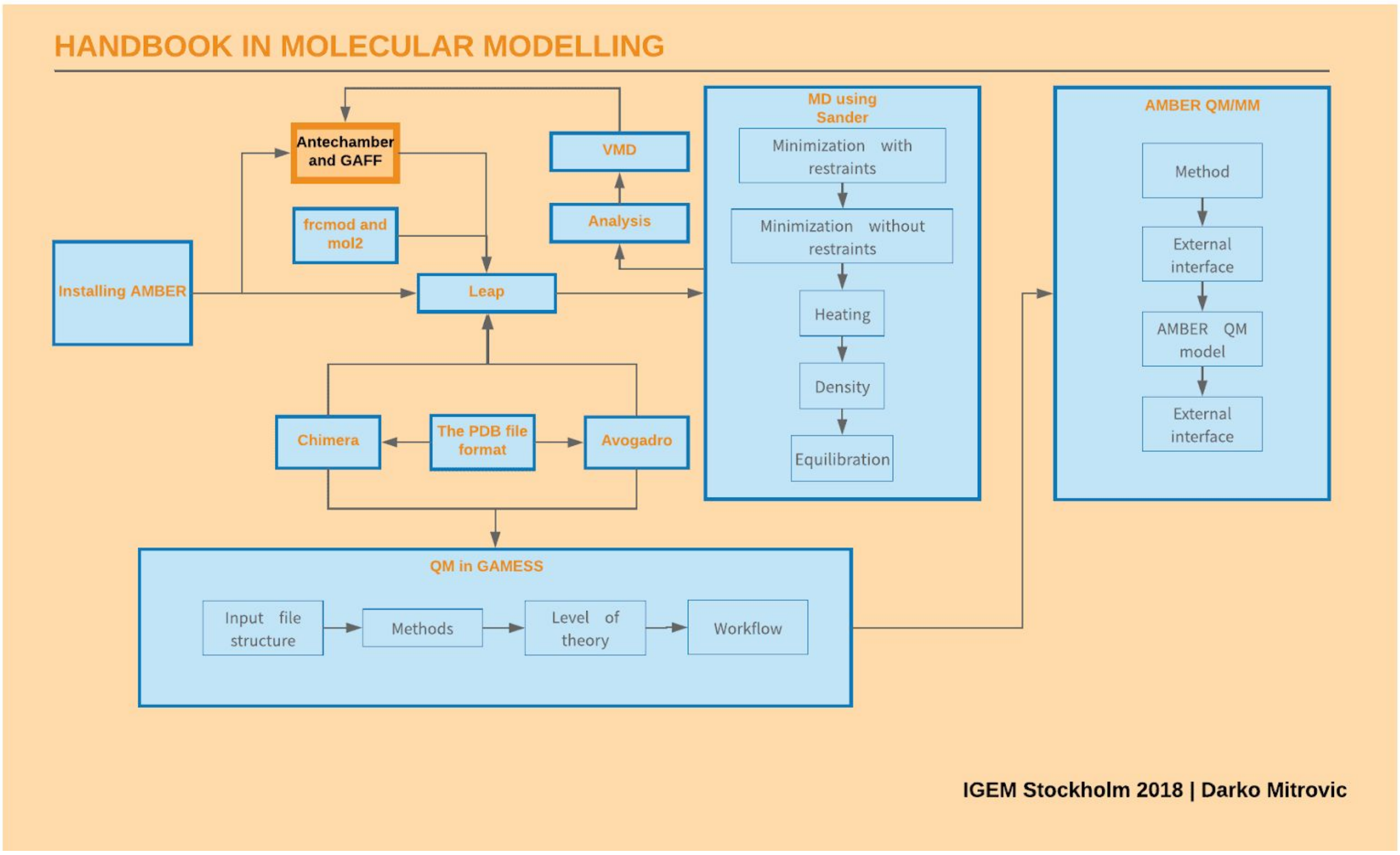
3. frcmod

The frcmod file contains these parameters that I have mentioned so extensively. There are a few different parameters that describe certain constants in the Amber Force Fields equations. They are individual for every single bond, atom and residue, and are:

K_r- bonding force constant
r_0- equilibrium bonding distance
K_θ- bond angle force constant
θ_0- equilibrium bonding angle
V_n- dihedral torsion constant
ϕ,γ- dihedral (spherical) angles
A_{ij}- Van der Waals repulsion constant
C_{ij}- Van der Waals attraction constant
q_i,q_j- partial charges of residue i and j.

$$E_{total} = \sum_{bonds} K_r(r-r_0)^2 + \sum_{angles} K_\theta(\theta-\theta_0)^2 + \sum_{torsions} \frac{V_n}{2}[1 \pm \cos(n\phi-\gamma)] + \sum_{non-bonded} \left[\frac{A_{ij}}{r_{ij}^{12}} - \frac{C_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{r_{ij}}\right] \qquad (1)$$

The bonded parameters are defined in the frcmod file, i bonds, angles and dihedrals (torsions). These can be determined ab initio in QM calculations, but since libraries exist for ordinary, covalently bonded atoms, this is not needed in most cases. In the case of heme groups or complexed metallic ions, you have to make the parameters ab initio. This is very complicated and is really only recommended if you absolutely need it, since nothing in QM is straightforward. frcmod files are essential for input generation of non-standard residues, as they will provide LEaP with these parameters, and mol2 will provide LEaP with the geometric structure in which you can find these bonds. To compile these from the gaff.dat library, you would do best to use antechamber, which incidentally is the next chapter.



ANTECHAMBER AND GAFF

Antechamber is a routine for compilation of frcmod and mol2 files from pdb files. We will use this as the link between the initial crystal structure and LEaP. GAFF is a library that contains a lot of empirical and QM data on an array of bonds, angles and dihedrals, basically all the parameters that you would need in your frcmod files. Antechamber takes this information, and packs the PDB file with this information by matching the atom type in the mol2 file with the parameter information from GAFF. Again, reading the frcmod and mol2 section is crucial for deeper understanding of this chapter.

1. Introduction

To manually implement all parameters and nonstandard residue geometries into useful frcmod and mol2 files is very difficult for large structures, so that is why we use Antechamber for this. Gaff is simply the library from which Amber takes all existent parameters for organic molecules.

2. Antechamber usage

Antechamber is just what it sounds like, for non-standard residues at least, and is used to prepare the files for LEaP. Antechamber takes the information from a pdb file, and analyses the structure in the way that seems logical to the program. No triplets, broken bonds, or incomplete valences are accepted here, so make sure that every bond is defined in the CONECT records, and that every hydrogen is added in suitable positions. (Please see the sections on the PDB-files and frcmod and mol2 for further explanation). Furthermore, it is important to open the file and control that every bond has been interpreted properly, and that the recognition of atom types matches well with the type presented in the pdb file.

To initiate generation of the mol2 file, run antechamber after sourcing the amber.sh directory (see the installing amber section).

```
antechamber -i <pdb_file> -fi pdb -o <output_file_name> -fo mol2
```

Furthermore, now that we have identified the atom types, we can continue with coupling those atom types and bonds that are implicitly stated in the bond record of the mol2 file to the parameters that are stored in the gaff library. parmchk2 is the routine that takes care of this, but it simply compiles what already exists in the library, so if your bond types aren't covered by these files, it will simply fail. Be sure to check the output of parmchk2 since it does not always give error messages in the terminal. Also, it is useful to check the frcmod file generated by parmchk2, to see how exact the parameters used were. For dihedrals especially, patterns like x - n - o -x or similar indicates low accuracy, because any interactions between the substituents at positions x are ignored. See the Gaff library section for more information. Anyway, let's run the parmchk2 command like this:

```
parmchk2 -i <mol2_file> -f mol2 -o <frcmod_file_name>
```

The Amber developers have a history of changing the name of this routine for some reason, so if you can't find parmchk2 on your version of amber, cd into the bin folder and type ls. There, there should be an executable file with a name in the same fashion as parmchk2. The difference between Amber16 and Amber18 is simply parmchk and parmchk2, respectively. Thus, a parmchk3 would not be unlikely to pop up in the future.

3. Gaff library

The gaff.dat library is where all the standard parameters for organic molecules can be found. If you have more information about specific dihedrals for examples, look into this file and add them with the suitable syntax and they will be automatically implemented. The more refined parameters you have, the more exactly you can model your system. Dihedrals are in many cases very unspecific, and can be updated through literature search or a QM-calculation. Simply opening this file with a text editor can prove to be very useful for understanding what kind of parameters you are dealing with. The bonded parameters are listed below. This can be found in the frcmod or gaff.dat files in the *amberXX/dat/leap/parm/* path.

```
CA-cd  418.3   1.4290   gaff cd-cd
cc-cd  504.0   1.3710   gaff cc-cd
cc-nc  431.6   1.3760   gaff cc-nc
cc-c   377.4   1.4620   gaff c -cc
cd-nc  494.6   1.3350   gaff cd-nc
```

The first column represents between what atom types the bond is considered. The atom types should be specified in your mol2 file. The second column gives the force constant of the bond, and the third column gives the equilbrium bond length, when there is no (virtual) potential energy stored in the bond. For nonbonded parameters, they are most often described in the source (include *source leaprc.gaff* in LEaP) , but sometimes parmchk2 chooses to include them in the frcmod file. They usually look like this:

```
nd       1.8240  0.1700          gaff nd
nc       1.8240  0.1700          gaff nc
ne       1.8240  0.1700          gaff ne
```

The two columns are the A_{ij} and C_{ij} parameters that are used in the Amber Force Field ff14SB, i.e. repulsive and attractive Van der Waals constants. Further, angular parameters look like this:

```
H1-CX-nd   50.110     108.570   gaff h1-c3-nd
C -CX-nd   66.810     111.370   gaff c -c3-na
H1-CX-cd   47.030     111.620   gaff cd-c3-h1; considered parm10 HC-CT-CA  (50.0 109.5)
but cd prefered over CA
3C-CX-cd   63.580     111.890   gaff c3-c3-cd; considered parm10 CT-CT-CA  (63.0 114.00)
but cd prefered over CA
```

The text is simply notes on approximations or sources left by the developers or providers of the source material. The first column gives the angular force constant, and the equilbrium angle is given in the second column. The angle is of course considered over the planar three point structure by looking at the angle formed by two line segments from the peripheral atoms connecting at the central one. The angle formed is then the angle considered. This is important to think about if you are doing this manually, it can be confusing sometimes. For dihedrals, things are pretty self- explanatory:

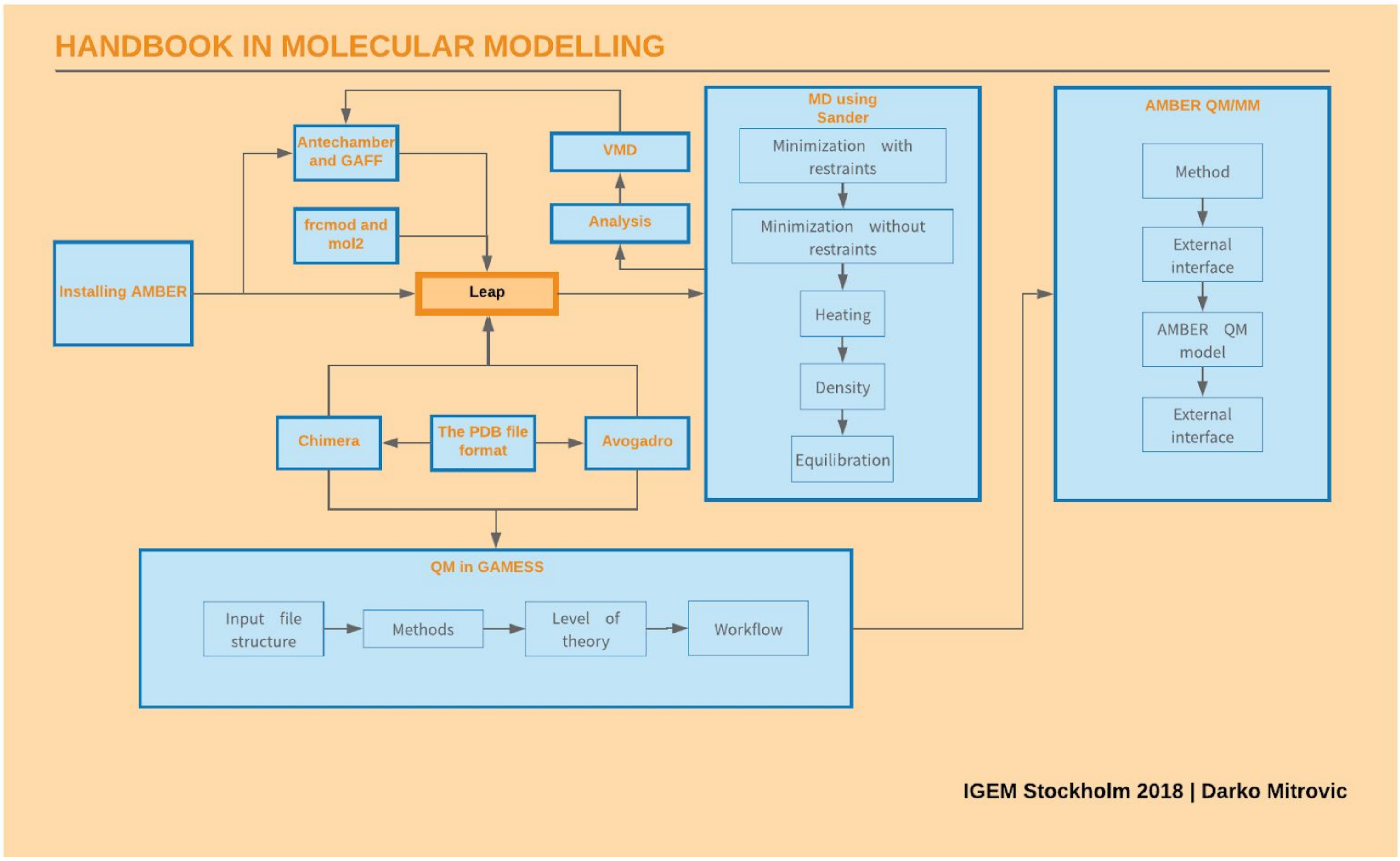
```
X -nd-os-X     1        4.800             180.000          2.00
```

```
X -nd-ss-X      1       4.800           180.000         2.00
X -nf-pe-X      1       5.400           180.000         2.00
c3-c -sh-hs     1       2.250           180.000         -2.000
c3-c -sh-hs     1       1.300           180.000         1.000
c2-c2-ss-c3     1       1.100           180.000         -2.000
c2-c2-ss-c3     1       0.700           180.000         3.000
```

These are some dihedral parameters that I found in one of the gaff.dat files. Here, we can see that if you are looking at the nf-pe dihedrals, any substituents to these atoms will be the treated the same, even if they can have very different properties. If you see too many of these X-es in your frcmod file, maybe take a look if you can find parameters manually. On the other hand, you have very nice dihedral information for the c-sh bonds. Sometimes, you can approximate your parameters by finding the ones that have the most chemically similar species. The first column is the source (not relevant), the second column is the torsion force constant, the third is the equilibrium dihedral angle (most are 180º due to steric clashes/interactions). You don't need to worry about the last one either (if you do, you already know it). Lastly, we have mass parameters, that are very easy to get a hold of (just google the atomic weight of the element you are considering):

```
nd 14.01        0.530   gaff nd; Sp2 N in non-pure aromatic systems, identical to nc
nc 14.01        0.530   gaff nc;
ne 14.01        0.530   gaff ne/nf; Inner Sp2 N in conjugated systems
cc 12.01        0.360   gaff cc/cd; Sp2 carbons in non-pure aromatic systems, identical to cd
cd 12.01        0.360   gaff cc/cd  Sp2 carbons in non-pure aromatic systems, identical to cc
cf 12.01        0.360   gaff cf; Inner Sp2 carbons in conjugated systems
```

As you can see, the parameters should be constant for a single element (all cc, cd and cf atom types correspond to carbon with different electronic conjugation). This can be tweaked if you are doing experiments with weird isotopes. This can change your simulation quite a bit, so be careful.

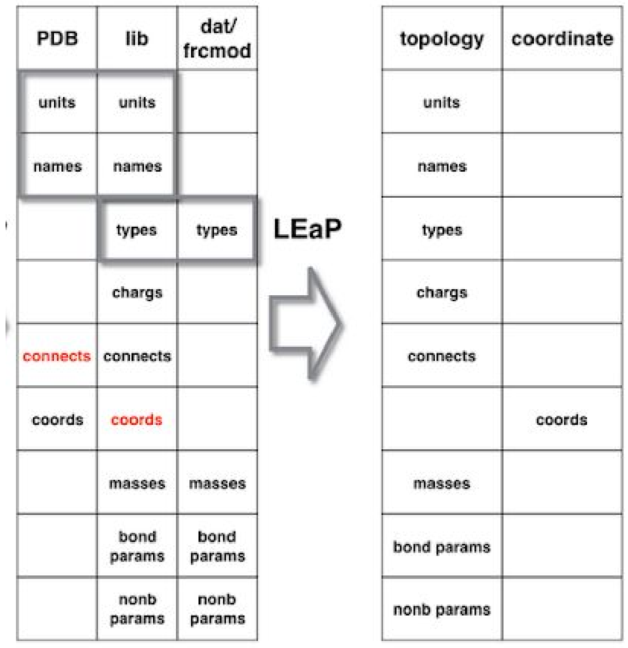


LEAP

Leap is sometimes what I want to do after a long day of difficult parametrization of organic componds, but it is incidentally the name of the program that I do it in. Leap is a great program for preparing files in different formats for the actual MD simulation in Sander, but is very unintuitive at first glance. To understand this better, I would highly suggest looking at the section about frcmod and mol2. Also, a quick refresh on the PDB file format and what information is contained within could be very useful. It is a tool provided in the ambertools package (both tleap - terminal based - and xleap - graphical user interface -)

1. Introduction

Before we can start with the Molecular Dynamics simulation, we need to prepare the appropriate files. For the Amber Force Field to work we need three things; (1) parameters (what properties the atoms and bonds involved have) (2) topology (how the atoms are connected), and (3) coordinates (where the atoms are). The first two things are stored in the parameter/topology file (.prmtop) and the third thing is stored in the input coordinates file (.inpcrd). Since the only thing we want to change during a Molecular Dynamics simulation are the coordinates/geometry of the structure, it is very, contrary to what it might appear on the picture, very convenient to separate .prmtop and .inpcrd. This enables us to perform subsequent simulations by simply exchanging the .inpcrd file. Even the format can be changed to .mdcrd, .rst7 or .rst, and latter programs can still handle it.



PDB	lib	dat/ frcmod
units	units	
names	names	
	types	types
	chargs	
connects	connects	
coords	coords	
	masses	masses
	bond params	bond params
	nonb params	nonb params

topology	coordinate
units	
names	
types	
chargs	
connects	
	coords
masses	
bond params	
nonb params	

Everything we have done so far has been to prepare the initial structure for leap. Leap compiles all the PDB records with what is found in the mol2 files or existing libraries. In the PDB file, we have records of names and coordinates of every atom from the crystal structure. As you can see, the CONECT records in the in the PDB file are not too important within leap, since we will see how it recognises already existing residues depending on how we load them into the LEaP routines, and replaces the CONECTs in the PDB files with bonds in the parameter and topology file. The coordinates in the library files are updated to match at the correct places in the PDB file. (for example, if our nonstandard residue is a modified Alanine called ALAX, Leap will simply add the missing atoms according to what is stored in the library files at the appropriate location so that the coordinates from a mol2 file are aligned with those of the PDB file). That is why you can remove all the CONECTs from a PDB file and still find that the topology is correct. Especially with standard amino acids, this is very easy, since the TER cards determine when the N- and C- termini come on the chain, and it will bind the backbone of everything inbetween. Also, hydrogens will be added automatically on all residues if you have loaded the correct source material.

2. tleap and xleap

There are two implementations of LEaP in the ambertools package, tleap and xleap. xleap is, as the name suggests, a visualisation tool with a separate GUI that will be initiated when prompted with the command *xleap* in the terminal. It sucks for large structures however, so it is easier to simply use a script in tleap. tleap is the terminal based implementation of LEaP in the ambertools package. The only difference is that when typing *edit <molecule>* in xleap, you can visually see the structure. It is only useful for small molecules however, because of that the bonding has to be done manually by dragging the mouse from one atom to another, which is immensely tedious. Let us therefore continue with tleap.

Before we load in the pdb file that contains the appropriate geometry, we need to load all files that we need to process it so that the topology is correct and the missing atoms are added so that they match the records. Firstly, we load the general libraries for standard protein residues, organic molecules, and water residues. In this case, I used the TIP3P model for solvation. In future renditions, you should use updated versions of the standard Amber force field. In this case, I used the latest one - ff14SB.

source leaprc.protein.ff14SB
source leaprc.gaff
source leaprc.water.tip3p

If we have any non-standard residues apart from water, we need to load BOTH the frcmod and mol2 files for this file (for information on how to generate these files, please have a look at the antechamber and gaff section). THIS IS VERY IMPORTANT: You have to name the object into which you are loading the mol2 file the same as the residue in the template pdb file is called.

<object_residue_name> = loadmol2 <mol2_file>

loadamberparams <frcmod_file>

Then, we can finally assign the structure of the pdb file into an object in LEaP:

<object> = loadpdb <template_pdb_file>

Now, it should have been properly loaded. Be sure to watch out for any messages warning about doublets (they will be deleted, so if you have misnumbered the residues, the atom count will be incorrect), or if any unexpected heavy atoms were added. Before we continue, we have to process the file so that the topology is normal. This is only needed if you have non-standard residues. If you have a non-standard amino acid in your backbone or a non-standard residue that needs to be bonded to another residue, the bond command must be used:

bond <object>.<resnumber>.<atomname> <object>.<resnumber2>.<atomname2>

The syntax here can be confusing at first glance, but it is really easy. In the PDB file, you can find the residue number of the residues that you want to connect. Within each residue, the atom name can be found (for example N, C, CA, HE2, OD1 etc.). This ties in with the reason why each atom name has to be unique and why LEaP deletes any doublets - to make sure that you only bond the atoms that you meant to connect. If you get an error message here, go to the PDB file and remove the guilty CONECT lines to proceed successfully. Note that the atom numbers don’t match if LEaP added missing atoms when loading the pdb file (this can get really complicated, so don’t even bother thinking about how the atom numbers change due to LEaP weirdness). Below is an example of coupling the N-terminal of amino acid number 65 to the C-terminal of number 64, and I loaded the pdb file into an object that I named mol (*mol = loadpdb <template_pdb_file>*):

bond mol.65.N mol.64.C

To check the file for misplaced atoms, it is always useful to save a pdb file. Note that this PDB file will NOT necessarily have the correct topology, and will never show the results of the bond commands. It will be evident if you have bonded the molecules correctly if your results look fine and you have no residues flying away. Anyway, save off a pdb file by referring to the object that you loaded the pdb file into:

savepdb <object> <output_filename>

It is usually really useful to solvate your protein or structure in a box of water. (you can use other geometric objects than cubes (box), but this is easiest as a starting point). But before I do this step, I usually save parameters and input coordinates (prmtop and inpcrd files) for the the dry protein only.

saveamberparm <object> <filename.prmtop> <filename.inpcrd>

solvatebox <object> <solvent_model> <solvent_radius>

Or, if you, like I am here, using the solvent force field TIP3P in a box format, replace *<solvent_model>* with TIP3PBOX. I usually set *<solvent_radius>* to about 10.0. This value is in Ångströms. What is also good to do to simulate realistic conditions is to add explicit ions counteracting the charges on the protein. The last value, that I have set to 0 can be changed if you want another overall charge.

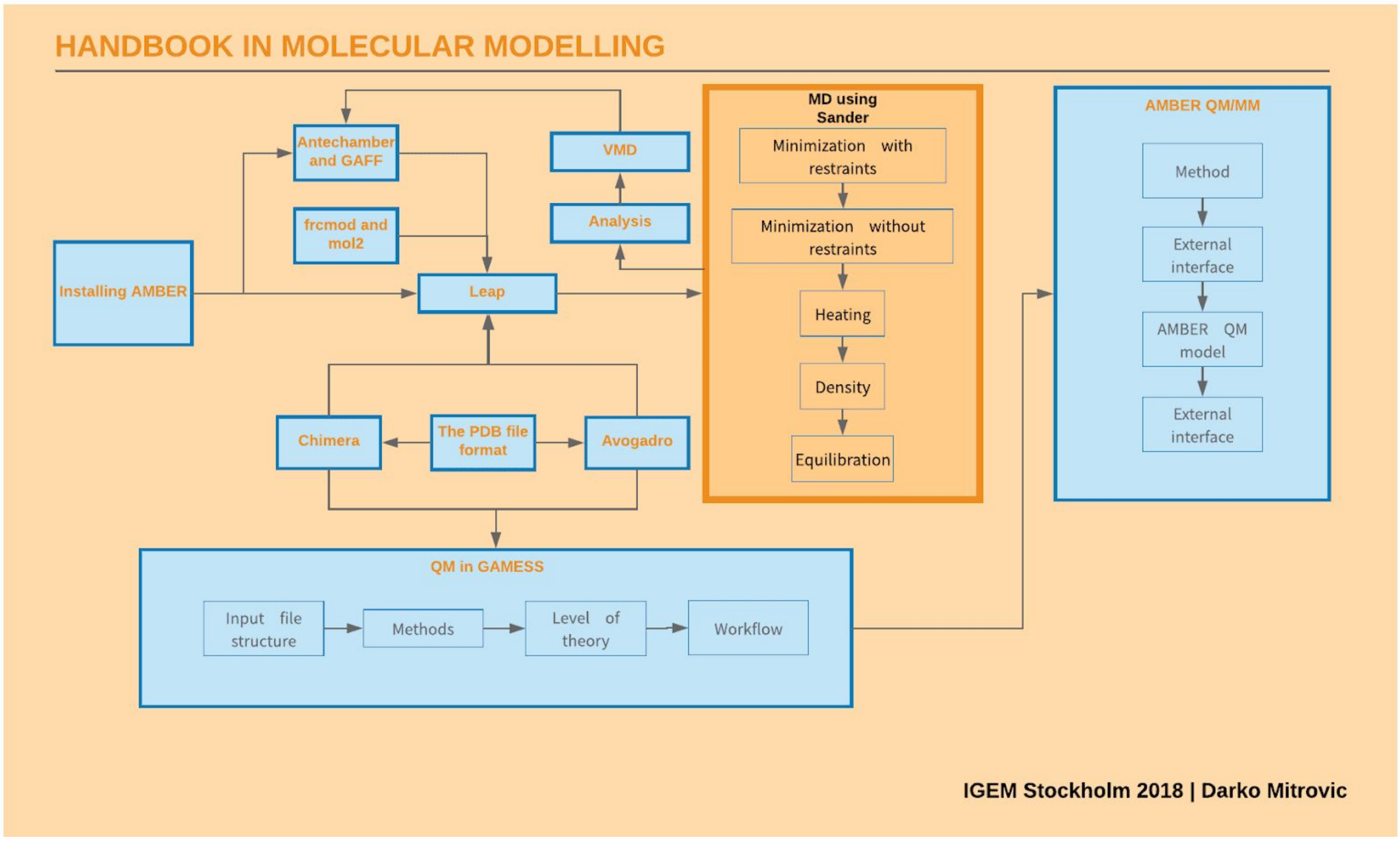
addions <object> Na+ 0
addions <object> Cl- 0

You can add other ions, but be sure to check the frcmod files for ions in the *amberXX/dat/leap/parm/ location* to see which ones are available. Then, we need to save the final parameters. We do that in the same way as before.

saveamberparm <object> <filename.prmtop> <filename.inpcrd>

With tleap, you can write all of this into a txt file named <filename.txt>, and execute it using:

tleap -f <filename.txt> , and the output should be given directly in the terminal. Good luck; if you succeed with this, the rest will (should) be a breeze.



MD USING SANDER

Sander is a subroutine in AMBER that is used for Molecular Dynamics simulation. To perform a viable, equilibrated simulation, we need to do a number of things first. This guide is aimed at first-time users of sander, but also contains tips and tricks if you are having difficulties with your simulation. While a full input generation guide is available in the AMBER reference manual, this is only a brief introduction to a few of the basic functionalities in sander. For QM/MM in Sander, please see the AMBER QM/MM section.

1. Introduction

In Amber, there are two different implementations - one faster (pmemd) and one slower (sander). The slower one has more functionalities (we can even use it for QM/MM calculations and simulations) and the faster one works on easier terms and can often not be trusted as much, but is a good tool nevertheless. This introduction will cover most of the important functionalities for simple Molecular Dynamics simulations (MD) in sander.

2. Usage

This program actually calculates the needed trajectories, so be careful with running nanosecond long jobs on your laptop. For my calculations, I used supercomputers in Umeå and Uppsala (hpc2n and Uppmax, respectively). Because of that, there are potentially a lot of arguments that can be entered. It is important that you treat different classes of files properly within the language of the routine. It is run using an input file (.txt), parameter/topology file (.prmtop), input coordinates (.mdcrd, .inpcrd, .rst or .rst7), and a reference file (often times .inpcrd or .rst). Make sure that all of the files you will use are in your working directory, which is also where multiple output files while end up. The following options to sander are available:

-O overwrites previous files.
-i input file (.in). The input tells sander what operations to perform and with what settings to do them. This will become clearer further below.
-p parameter/topology file (.prmtop). The prmtop file generated by LEaP
-c input coordinate file (.inpcrd, .mdcrd, .rst, .rst7). The inpcrd file generated by LEaP, or a binary restart file generated by a previous sander session. Can also be a mdcrd file, which can be generated by sander upon request by the user. This is in most cases absolutely redundant, and obsolete even at the time of writing this. So is the rst7 format.
-r name of the restart file (.rst) generated by sander. It is a binary restart file that can be used within sander synonymously with inpcrd, as explained above. It contains information about velocities and other interesting quantities.
-o name of the output file (.out) generated by sander. A text file containing condensed information about the MD, as energy for every iteration (depending on what your ntpr value is), as well as estimated time left.
-ref a reference file (.inpcrd, .rst, .mdcrd). Only applicable if you are using restraints (ntr=1 in the .in file). It is this file that will be the template for the restraints you put on your system, and a deviation from this structure in absolute space (no gyration or translocation , i.e. relative positional change, is considered).
-inf an information file similar to the output file, but with information only about the last iteration.
-x NetCDF file for trajectory analysis (can be ASCII too, but I would not use this format). Only available if the ntx option is enabled (=1). This is useful for visualisation of your trajectory in VMD, see the section about VMD for more information. Never run an MD without printing the trajectory with ntx=1.

Sander is run by simply typing the following in the terminal:

sander -O -i <input> -p <prmtop> -c <inpcrd> -o <output> -r <rst> -ref <refc> -inf <mdinfo> -x <nc>

Typically you do not run only one of these, but several sequential runs that have different input settings. The general workflow is minimization with restraints, relaxed minimization, density simulation, heating simulation, MD simulation with equilibration. The following sections will explain what differences you would typically see in the different input files. The following settings are available for these input files. (taken from the Amber18 reference manual). If you are not interested in why we choose different options for different types of runs, skip this list (it is quite long), and go to section 3. Minimization with restraints.

imin= 0 (default) Run molecular dynamics without any minimization.
imin= 1 Perform an energy minimization.
imin= 5 Read in a trajectory for analysis. Although sander will write energy information in the output files (using ntpr), it is often desirable to calculate the energies of a set of structures at a later point. In particular, one may wish to post-process a set of structures using a different energy function than was used to generate the structures.
nmropt= 0 (default) No nmr-type analysis will be done.
nmropt= 1 NMR restraints and weight changes will be read.
nmropt= 2 NMR restraints, weight changes, NOESY volumes, chemical shifts and residual dipolar restraints will be read.

ntx= 1 (default) Coordinates, but no velocities, will be read; either formatted (ASCII) files or NetCDF files can be used, as the input file type will be auto-detected.
ntx= 5 Coordinates and velocities will be read from either a NetCDF or a formatted (ASCII) coordinate file.
ntb>0. The velocity information will only be used if irest= 1 (see below).

irest= 0 (default) Do not restart the simulation; instead, run as a new simulation. Velocities in the inputcoordinate file, if any, will be ignored, and the time step count will be set to 0 (unless overridden by t; see below).
irest= 1 Restart the simulation, reading coordinates and velocities from a previously saved restart file. The velocity information is necessary when restarting, so ntx (see above) must be 4 or higher if irest= 1.

ntxo = 1 Formatted (ASCII). Format of the final coordinates, velocities, and box size (if constant volume or pressure run) written to file "restrt".
ntxo = 2 (default) NetCDF file (recommended, unless you have a workflow that requires the formatted form.)

ntpr Every ntpr steps, energy information will be printed in human-readable form to files "mdout" and "mdinfo". "mdinfo" is closed and reopened each time, so it always contains the most recent energy and temperature. Default 50.

ntave Every ntave steps of dynamics, running averages of average energies and fluctuations over the last ntave steps will be printed out. A value of 0 disables this printout. Setting ntave to a value ½ or ¼ of

nstlim provides a simple way to look at convergence during the simulation. Default = 0 (disabled).
ntwr Every ntwr steps during dynamics, the “restrt” file will be written, ensuring that recovery from a crash will not be so painful. No matter what the value of ntwr, a restrt file will be written at the end of the run, i.e., after nstlim steps (for dynamics) or maxcyc steps (for minimization). If ntwr <0, a unique copy of the file, “restrt_<nstep>”, is written every abs(ntwr) steps. This option is useful if for example one wants to run free energy perturbations from multiple starting points or save a series of restrt files for minimization. Default = nstlim.

ntwx: Every ntwx steps, the coordinates will be written to the mdcrd file. If ntwx
= 0, no coordinate trajectory file will be written. Default = 0.

ntwv: Every ntwv steps, the velocities will be written to the mdvel file. If ntwv
= 0, no velocity trajectory file will be written. If ntwv= -1, velocities will be written to mdcrd, which then becomes a combined coordinate/velocity trajectory file, at the interval defined by ntwx . This option is available only for binary NetCDF output (ioutfm= 1). Most users will have no need for a velocity trajectory file and so can safely leave ntwv at the default. Default = 0. Note that dumping velocities frequently, like forces or coordinates, will introduce potentially significant I/O and communication overhead, hurting both
performance and parallel scaling.

ntwf: Only use set this to -1 if you want force output in mdcrd. I would highly recommend you not to touch this option.

ntwprt= 0 (default) Include all atoms of the system when writing trajectories.
ntwprt> 0 Include only atoms 1 to ntwprt when writing trajectories.

idecomp Perform energy decomposition according to a chosen scheme. In former distributions this option was only really useful in conjunction with mm_pbsa, where it is turned on automatically if required. Also, idecomp is really useful in qm/mm.
= 0 (default) Do not decompose energies.
= 1 Decompose energies on a per-residue basis; 1-4 EEL + 1-4 VDW are added to internal (bond, angle, dihedral) energies.
= 2 Decompose energies on a per-residue basis; 1-4 EEL + 1-4 VDW are added to EEL and VDW.
= 3 Decompose energies on a pairwise per-residue basis; otherwise equivalent to idecomp
= 1. Not available in TI simulations.
= 4 Decompose energies on a pairwise per-residue basis; otherwise equivalent to idecomp
= 2. Not available in TI simulations.If energy decomposition is requested, residues may be chosen by the RRES and/or LRES card. The REScard is used to select the residues about

which information is written out. See chapters 21.1 or 33 for more information. Use of idecomp >0 is incompatible with ntr>0 or ibelly>0.

ibelly
Flag for belly type dynamics. If set to 1, a subset of the atoms in the system will be allowed to move, and the coordinates of the rest will be frozen. The moving atoms are specified with bellymask. This option is not available when igb>0. When belly type dynamics is in use, bonded energy terms, vdW interactions, and direct space electrostatic interactions are not calculated for pairs of frozen atoms. Note that this does not provide any significant speed advantage. Freezing atoms can be useful for some applications but is maintained primarily for backwards compatibility with older versions of Amber. Most applications should use the ntr variable instead to restrain parts of the system to stay close to some initial configuration.

ntr Flag for restraining specified atoms in Cartesian space using a harmonic potential, if ntr > 0. The re- strained atoms are determined by the restraintmask string. The force constant is given by restraint_wt. The coordinates are read in "restrt" format from the "refc" file. Default= 0.

maxcyc The maximum number of cycles of minimization. Default = 1.

ncyc If NTMIN is 1 then the method of minimization will be switched from steepest descent to conjugate gradient after NCYC cycles. Default 10.

ntmin Flag for the method of minimization.
= 0 Full conjugate gradient minimization. The first 4 cycles are steepest descent at the start of the run and after every nonbonded pairlist update.
= 1 For NCYC cycles the steepest descent method is used then conjugate gradient is switched on (default).
= 2 Only the steepest descent method is used.
= 3 The XMIN method is used.
= 4 The LMOD method is used.

dx0: The initial step length. If the initial step length is too big then will give a huge energy; however the minimizer is smart enough to adjust itself. Default 0.01.

nstlim Number of MD-steps to be performed. Default 1.

t The time at the start (psec) this is for your own reference and is not critical. Start time is taken from the coordinate input file if IREST=1. Default 0.0.

dtThe time step (psec). Recommended MAXIMUM is .002 if SHAKE is used, or .001 if it isn’t.

ntt
= 0 Constant total energy classical dynamics (assuming that ntb <2, as should probably always be the case when ntt=0).

= 1 Constant temperature, using the weak-coupling algorithm.[353] A single scaling factor is used for all atoms. Note that this algorithm just ensures that the total kinetic energy is appropriate for the desired temperature; it does nothing to ensure that the temperature is even over all parts of the molecule. Atomic collisions will tend to ensure an even temperature distribution, but this is not guaranteed, and there are many subtle problems that can arise with weak temperature coupling.
= 2 Andersen-like temperature coupling scheme,[355] in which imaginary "collisions" randomize the velocities to a distribution corresponding to temp0 every vrand
steps. Note that in between these "massive collisions", the dynamics is Newtonian. Hence, time correlation functions (etc.) can be computed in these sections, and the results averaged over an initial canonical distribution. Note also that too high a collision rate (too small a value of vrand) will slow down the speed at which the molecules explore configuration space, whereas too low a rate means that the canonical distribution of energies will be sampled slowly.
= 3 Use Langevin dynamics with the collision frequency

temp0 Reference temperature at which the system is to be kept, if ntt > 0. Note that for temperatures above 300K, the step size should be reduced since increased distance traveled between evaluations can lead to SHAKE and other problems. Default 300.

ig The seed for the pseudo-random number generator. Default is 1000.

vlimit If not equal to 0.0, then any component of the velocity that is greater than abs(VLIMIT) will be reduced to VLIMIT (preserving the sign).

nkija For use with ntt=9 and ntt=10., For ntt=9, this the number of substeps of dt when integrating the thermostat equations of motion, for greater accuracy. For ntt=10, this specifies the number of additional auxiliary velocity variables v1 and v2, which will total nkija×v1+nkija×v2 [360].Default is 1 for both integrators.

idistr For the isokinetic integrator (ntt=9), the frequency at which the thermostat velocity distribution functions are accumulated.

ntp Flag for constant pressure dynamics. This option should be set to 1 or 2 when Constant Pressure periodic boundary conditions are used.
= 0 No pressure scaling (Default)
= 1 md with isotropic position scaling
= 2 md with anisotropic (x-,y-,z-) pressure scaling: this should only be used with orthogonal boxes (i.e. with all angles set to 90 degrees).
= 3 md with semiisotropic pressure scaling: this is only available with constant surface tension (csurften > 0) and orthogonal boxes. This links the pressure coupling in the two directions tangential to the interface.

barostat Flag used to control which barostat to use in order to control the pressure.
= 1 Berendsen (Default)
= 2 Monte Carlo barostat

mcbarint Number of steps between volume change attempts performed as part of the Monte Carlo barostat. Default is 100.

pres0 Reference pressure (in units of bars, where 1 bar≈0.987 atm) at which the system is maintained

taup Pressure relaxation time (in ps), when NTP>0. The recommended value is between 1.0 and 5.0 psec. Default value is 1.0, but larger values may sometimes be necessary

3. Minimization with restraints

This is usually the first input file that is used in a md simulation. The goal of this run is simply to relax the structure (the initial guess sometimes contains steric clashes or alike that need to be taken care of), so what we want to do is to make the system relax, but not fall apart. If you have as much as one small steric clash between the substrate molecule and the rest of the protein, they will simply dissociate and your substrate will be pushes out. If you are unlucky, the whole protein could unfold during this step. That is why we add restraints. The restraints make it so that we keep the system from straying to much from the initial structure. The key to this is applying a large force constant, and running a short minimization, so that we can restart it in the subsequent minimization without the velocity information. The key to this is to set ntr=1. NOTE: the restraintmask only applies to the residues that were in my template protein it is very IMPORTANT that you change this to suit your system.

```
initial minimisation
 &cntrl
          imin   = 1, ! Minimization turned on.
          maxcyc = 10000, ! Total number of cycles set to 10000.
          ncyc   = 5000, ! Steepest decent minimization for the first 5000 cycles.
          ntb      = 1, cut   = 15 ! NVT ( cut = interactions only happen within a 15 Å radius)
          ntr      = 1, ! Turn on cartesian restraints
          restraintmask = ':1-505', ! Only restrain residue (:=residue,@=atom) 1 through 505.
          restraint_wt = 500 ! force constant for the atoms only in the restraintmask.
 /
```

4. Minimization without restraints

This is pretty self-explanatory, if you have read the previous section (Minimization with restraints). We do another minimization, but we remove the restraints on the protein, and hope that any energy extremes have been eliminated (steric clashes etc.). If you experience difficulty with getting good structures here, use the input file from the previous section and reduce the restraintmask to only cover the atoms that are diverging geometrically. Also, check your parameters, it is often in this stage wrong parameters will be visible.

```
initial minimisation
 &cntrl
          imin   = 1,
          maxcyc = 10000,
          ncyc   = 5000,
          ntb      = 1,
          ntr      = 0,
          cut      = 15
/
```

5. Heating

As you know, the crystal structure is at approximately 0K, so we need to slowly increase the temperature without making the protein or solvent explode. To do this, we use a Langevin Thermostat (ntt=3), and make an isovolumetric heating. Later, we shall perform one isobaric calculation to make sure that the density of the solvent is correct.

```
Relaxation 1
 &cntrl
          imin=0, ntx=1, ! Run MD [imin=0]
          ntb=1,cut=12.0, ! NVT, 8 A cutoff
          ntp=0, ! No barostat
          ntc=2, ntf=2 ! Shake on
          ntt=3, gamma_ln=1.0, ! Langevin Thermostat, 1.0ps-1
          tempi=0.0, ! Initial Temperature of 0K
          nstlim=50000, dt=0.001,! 25K steps x 2fs = 50ps
          iwrap=1, ! Wrap coordinates to central box
          ioutfm=1, ! Write binary mdcrd
          ntpr=5000, ntwx=5000, ! Write to mdout and mdcrd every 5,000 steps
          ntwr=50000, ! Write restart file every 50,000 steps
          ntr=1, restraint_wt=50.0, ! Restrain backbone atoms with 4.0 KCal/Mol/A
          restraintmask='@C,CA,N',
          ig=-1, ! Use 'random' random number seed.
          nmropt=1, ! Used to ramp temperature slowly
 /
&wt type='TEMP0', istep1=0, istep2=50000,value1=0.0, value2=10.0, /
&wt type='END'  /
```

As you can see, we added another namelist than only the &cntrl one. here, we are specifying weights for the nmropt=1 option. We are simply telling it to heat up from the first iteration to the end of the simulation. The restraintmask only applies to the backbone atoms now. You usually do not expect the structure to perturb too much during this step.

6. Density

As mentioned before, the next step is to try and condensate the solvent - when doing heating with minimal box restraints (only iwrap), you do not take into consideration that we actually have a huge solution, even if our explicit solvent model only stretches about 10-30Å away from the protein or solute. This means that we have to account for the external pressure that is being applied on this very small region of interest. We do this by enabling pres0=1 , which means that we have 1 atm of pressure in the solution. As per usual, we are restraining the backbone of the protein, but in this case, please compare the restraint_wt in the examples I have listed. In this simulation, the restraint_wt is 5.0, but in the initial minimization, it was 500. That means that we are loosening up and relaxing the system as we think that we approach an equilibrated system.

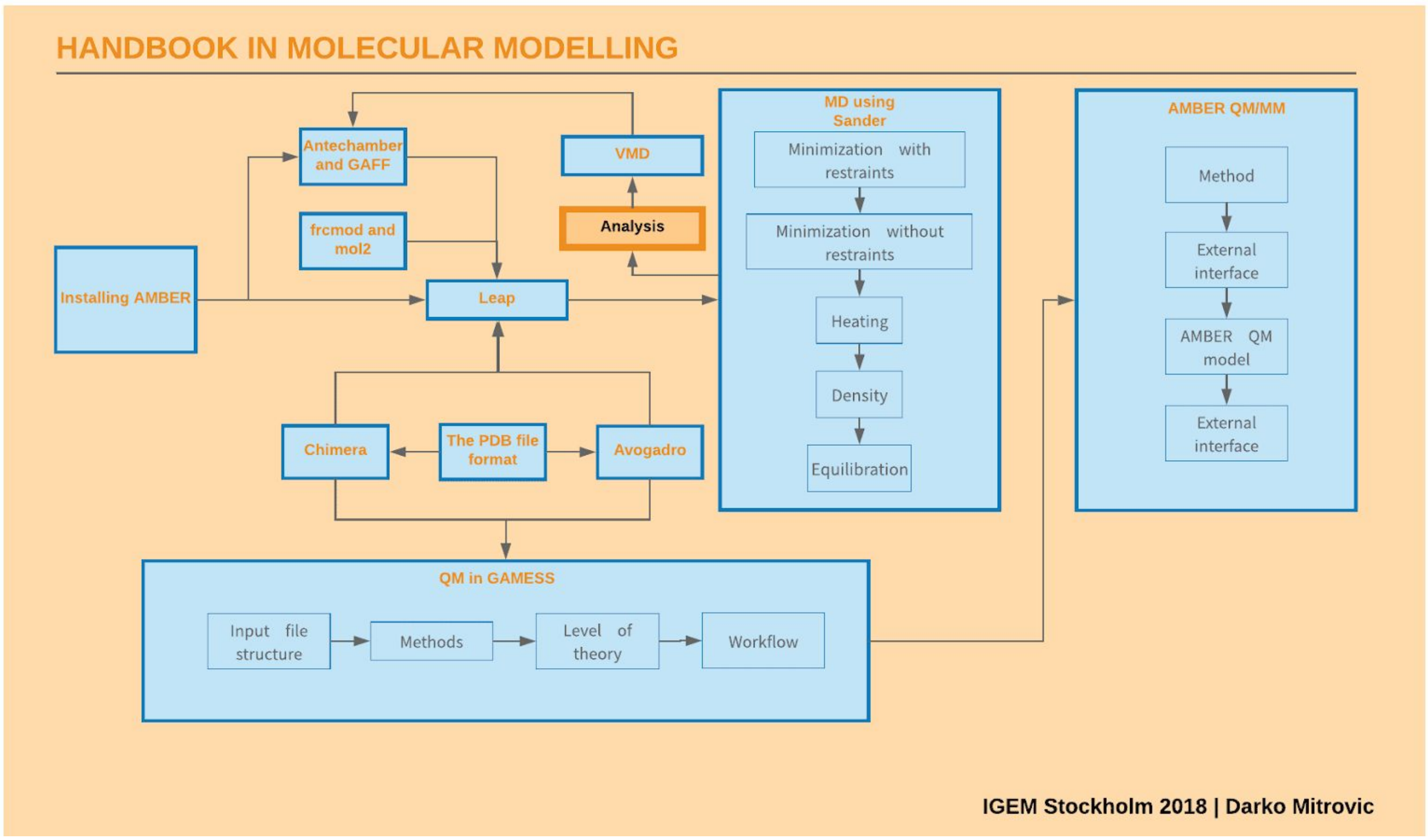
```
Relaxation 2
&cntrl
  imin=0, ntx=5, irest=1, ! Read box and velocity info from inpcrd
  ntb=2,cut=8.0, ! NPT
  ntp=1, pres0=1.0, ! Anisotropic pressure scaling at 1 atm
  ntc=2,
  ntf=2,
  ntt=3, temp0=300.0, gamma_ln=1.0,
  nstlim=500000, dt=0.002,
  iwrap=1, ioutfm=1,
  ntpr=5000, ntwr=50000, ntwx=5000,
  ntr=1, restraint_wt=5.0, restraintmask=':1-505', ! Keep weak restraints on backbone
  ig=-1,
  nmropt=1,
/
&wt type='TEMP0', istep1=0, istep2=125000,value1=50.0, value2=300.0 /
&wt type='TEMP0', istep1=125001, istep2=500000,value1=300.0, value2=300.0 /
&wt type='END'  /
```

Here, we are using a bit more complicated restraining model with nmropt=1. 25% of the simulation is bringing the solvent from 50 K to 300 K, and then, for 75%, we are simply keeping the temperature at 300 K, which is the desired temperature in this simulation. After this, you have successfully gone from a harsh, crude crystal structure *in vacuo* to a solvated system, hopefully equilibrated, and natural system.

7. MD simulation at desired temperature

Finally, after all parametization, input generation, refinement of structure, and initial equlibration simulations, we are finally ready for our production simulation as we call it within AMBER, or Molecular Dynamics simulation, as outsiders would refer to it. These commands should be pretty self-explanatory, as we are not doing anything new during this simulation.

```
Production simulation
 &cntrl
  imin = 0, irest = 1, ntx = 7,
 ntb = 2, pres0 = 1.0, ntp = 1,
 taup = 2.0,
 cut = 10, ntr = 0,
 ntc = 2, ntf = 2,
 temp0 = 300.0,
 ntt = 3, gamma_ln = 2.0,
 nstlim = 100000000, dt = 0.002,ioutfm=1,
 ntpr = 10000, ntwx = 10000, ntwr = 10000, ntxo = 2
/
```



ANALYSIS

Upon completion of the Molecular dynamics simulation, we need to make sense of the millions of velocities, coordinates and forces throughout every frame of the simulation, that can sometimes stretch for as long as 10^6 frames. Instead of qualitatively viewing the structure evolving in VMD throughout every frame (while somewhat informative, incredibly tedious and hides a lot of important statistics), we can use the cpptraj subroutine to generate plots, graphics and data in csv format to visualise or use further. A lot about MD has to do with analysis of large amounts of data, and finding ways of expressing how your data correlates to physical significant phenomena is essential to a successful and impactful result from MD simulation.

1. Introduction

Generally, if no major structural difference occurs during the simulation (with major, I refer to domain translocation, or motor proteins in flagella and microtubules), no interesting information/data can be taken just by looking on the trajectory analysis in VMD. Instead, one can look at a few parameters that investigate how distances, forces, energies, and tertiary structure change and form during the entire simulation. This is particularly interesting for subjects such as electron transfer, or investigation of productive binding modes, where geometries and distances have to be analysed for every frame of simulation.

To generate the general information, the out file is used, which contains energies for the every iteration, even if you have set ntx to print every 1000th step. For finer analysis of distances between individual atoms or residues, you have to use your .prmtop file along with your trajectory file (preferably in .nc, or NetCDF format, but ASCII should work. ASCII is not binary though, so the files are bigger). This means that you can only analyse the steps that you have printed in ntx.

2. Parameters for investigation

There are many different parameters that can be investigated through the parm and traj commands in cpptraj (see paragraph 3 about cpptraj - commands and usage). Using different commands and codes in cpptraj, you can extract virtually any information available for the trajectory that you want to analyse. However, there are 5 parameters that should always be investigated to verify the validity of your MD simulation.

Temperature analysis. When you set your production simulation to 300K, you make the solvent shake so that the kinetic energy of all the solvent particles amounts to 300K. That doesn’t happen immediately, however, since the solute needs time to equilibrate according to the force field parameters. Therefore, if you see an MD simulations temperature curve, and it is still rising by the end of the simulation, it has not reached equilibrium, and the results cannot be trusted. This applies to every parameter since when equilibrium is reached, we are able to simulate the system under realistic conditions.

Energies. If kinetic energies, potential energies or total energies for the system are oscillating, rising, or jumping between states (should look like stairs or platforms), the system probably undergoes major conformational shifts. A more likely option is that the protein of interest has unfolded during simulation, or, if you have two different proteins, that they associate/dissociate with each other.

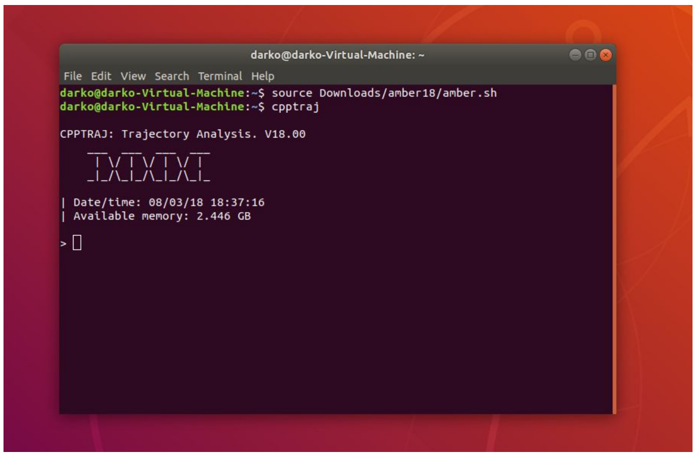
RMSD. This is the root mean square of the distance from the original structure. This is not, however, in cartesian coordinates, but the structures are aligned by minimizing the RMSD. A change in RMSD means that the protein structure has not reached equilibrium, and probably is in the process of relaxation. If you are unlucky, your enzyme is unfolding.

RMSF. This is the root mean square of the forces within the system. This is a good measure for if the system has relaxed, or if it is strained. You should see a downwards trend if you are starting from a crystal structure, or an upwards trend if you are heating your system.

Radius of gyration. This is only applicable for globular proteins. The radius of gyration measures the approximate radius that the object has during simulation. If this radius changes significantly, it means that the protein is expanding or shrinking. If it expands, it is unfolding. If it shrinks, it is folding more.

3. cpptraj - commands and usage

You can generate most of this data using the cpptraj subroutine in ambertools. Initiating cpptraj can be done by simply typing cpptraj after sourcing amber.sh (see the section about installing amber). This should be what you see. To generate the necessary files, we will need to load in the parameter/topology and trajectory data. To load in the parameter/topology file, use the command


darko@darko-Virtual-Machine: ~
File Edit View Search Terminal Help

```
darko@darko-Virtual-Machine:~$ source Downloads/amber18/amber.sh
darko@darko-Virtual-Machine:~$ cpptraj

CPPTRAJ: Trajectory Analysis. V18.00
    ___  ___  ___  ___
     | \/ | \/ | \/ |
    _|_/\_|_/\_|_/\_|_

| Date/time: 08/03/18 18:37:16
| Available memory: 2.446 GB

>
```

parm <path_to_.prmtop_file>

The parameter/topology has to be loaded first. Be sure to load the parameters associated with your trajectory file. Then, we can load it:

trajin <path_to_.nc_file>

Next, we can use a plethora of different commands to generate the necessary files. Here is a list of commands, their usage, and what output you can get from it. Before we can do that however, we need to install xmgrace or similar programs that we can use to visualise the .agr files we will generate. Simply google xmgrace and there is usually really nice installation guide. To start xmgrace, use *xmgrace <input_file.agr>* in the terminal, and you will see an interactive graph.

Sometimes, you only want information about a specific set of atoms. We need to select these using the internal cpptraj masking syntax. As per usual within amber, the symbols are : for residues, @ for atoms and # for models. @ and : are applicable both to numbers as well as names for residues and atoms. For example, for a residue called ALA with the resid 2, :2 will refer to that particular residue, while :ALA refers to all residues called ALA. To get information about atoms and residues loaded through the parm and trajin commands, you can simply type *resinfo* or *atominfo*. If you only want information about a certain selection, use the masking syntax after the *resinfo* or *atominfo* command.

Also, in the masking syntax, there are operators that function the same as the logical operators AND and OR, where AND is exclusive and OR is exclusive. The symbols are, as per usual & for AND and | for OR. For example, if we mean to select only the CA atoms within the first 100 residues, the syntax would be:

:1-100|CA

Lastly, and perhaps most interestingly for users with enough imagination, you can use distance restraints, i.e. only masking atoms that are within a certain radius from an atom or every atom of a residue. For example, the following line means to select every atom within a radius of 5.0 Å of atom number 300:

@300<@5.0

Or, if you want every atom within every residue in which at least one atom is closer than 5 Å from atom number 300:

@300<:5.0

Now, we have all the tools to use the commands. The usage should now be straightforward, so whenever <*mask*> appears, use to syntax provided above. To find the distance between two masks, type:

distance <name> <mask1> <mask2> out <filename>

Where, if you would like to open this output file (generated by *out <filename>*) in xmgrace, add *.agr* as an extension to your filename, and cpptraj will automatically create the file in the correct format. Next, we can find rmsd's and rmsf's by using:

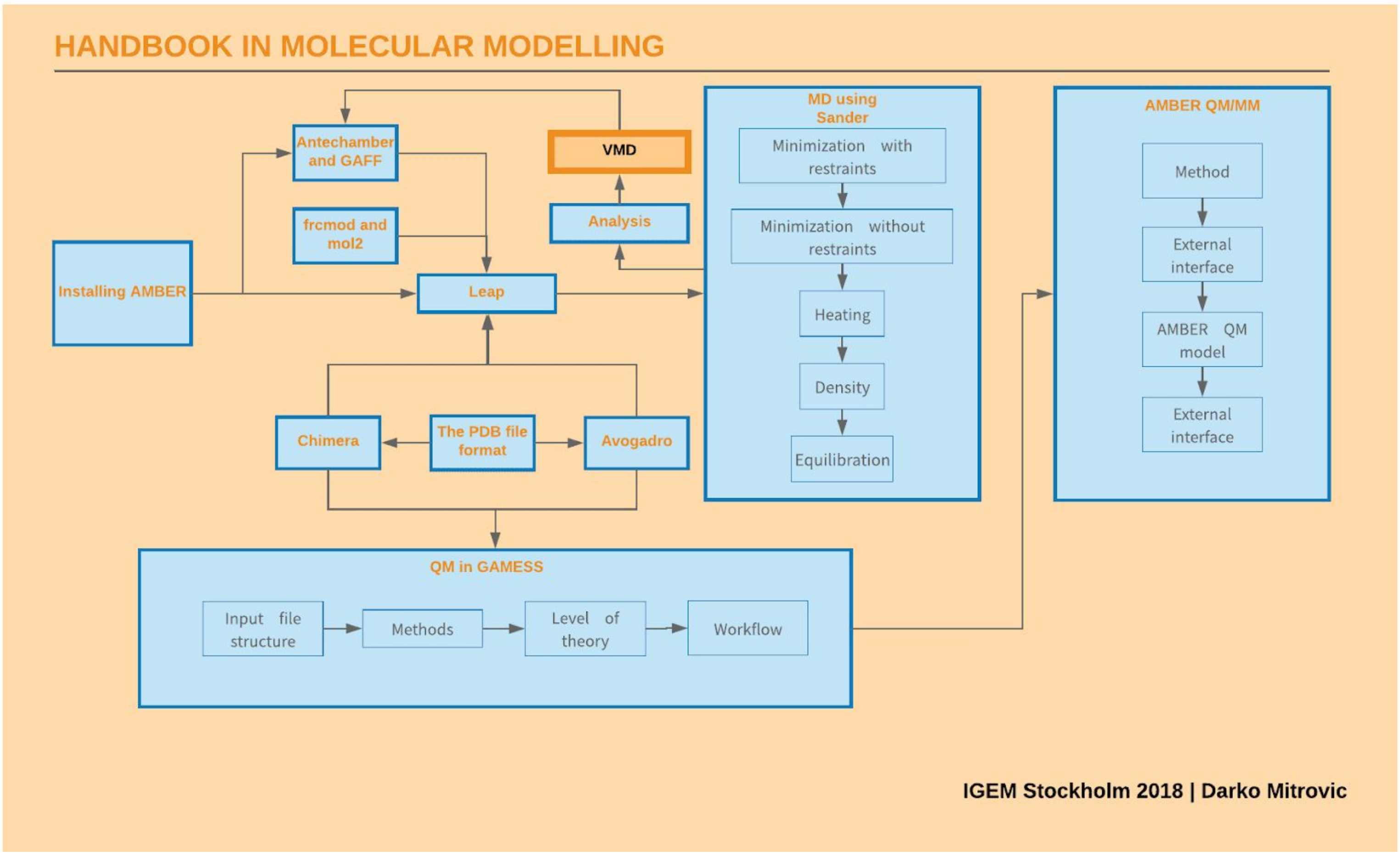
rms <name> <mask1> first out <filename>

atomicfluct out <name> <mask1> <filename>

adding nofit or mass will take cartesian coordinates RMS's and mass-weighted RMS's, respectively. The first one above is used for RMSD, and the second one is for RMSF. Finally, radius of gyration calculations can be done in an analogous fashion, by the simple command:

radgyr <name> <mask1> out <filename>

Again, adding .agr as an extension should be recognisable to cpptraj and readable for xmgrace (and a lot of other plotting routines/programs).



VMD

VMD is a visualisation software mainly for Linux and MacOS machines. We will use it to visualise molecules, orbitals and even trajectories (movies!) from our MD simulations. It is also useful since we can export our file to a number of file formats for work in 3D animation software, 3D printing, or simply for creating nice-looking images. This guide also includes a small section on installing the software in question.

1. Introduction

While I personally have some unfinished business with VMD, it is generally viewed as a good visualisation tool, and that I have to give it. Never use it, however, for input generation - Avogadro, Chimera and even cpptraj and LEaP are better with this. (The last two are commands within AMBER). VMD can be used to visualise trajectories (i.e. actual movies, your simulation frame for up to a second to second time scale), molecular orbitals and has more options for shading, lights and surface options. In short, VMD is usually what you use to render images for scientific articles, posters or your Wiki! Another great things is that you can export them in virtually any format to be used in animation programs such as blender.

2. VMD Download and installation

The VMD download and installation is very similar to the installation of AMBER. I would recommend to install AMBER first, and then install this, since I am unsure whether additional packages are needed, especially in Windows-Hyper-V environments. VMD is available both for MacOS and Ubuntu, but not Windows. Please see the Installing Amber section for an explanation on how to set up the ubuntu environment in a virtual machine in windows. (NOTE: You can do an X11-Powershell environment as well but I really recommend the virtual machine option). Anyway, google something along the lines of “download vmd”, and you will find a link to the download site. Please download it either for Linux 64x or MacOS. Cd into the Downloads directory from your home folder.

cd Downloads

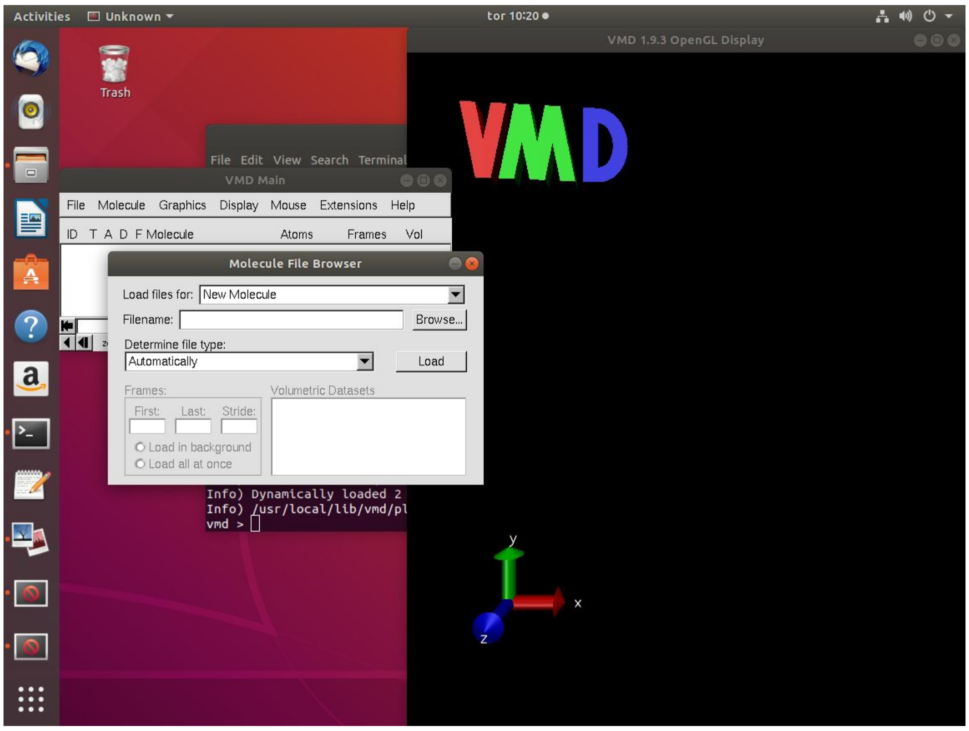
Then, we need to unzip the downloaded file. If the extension is tar.bz2 or tar.gunzip, use the *tar* command with different flags. Google it if you’re having problems here, it should be straightforward but they keep changing the zip method.

tar -xvfj <vmd_directory>

cd into the unzipped vmd directory, and type *./configure*. When configured, you can go ahead and install vmd by *make install*. Then, answer yes to any questions. Then, it should have been added to your .bashrc, so simply typing *vmd* in the terminal will suffice.

3. Usage with MD

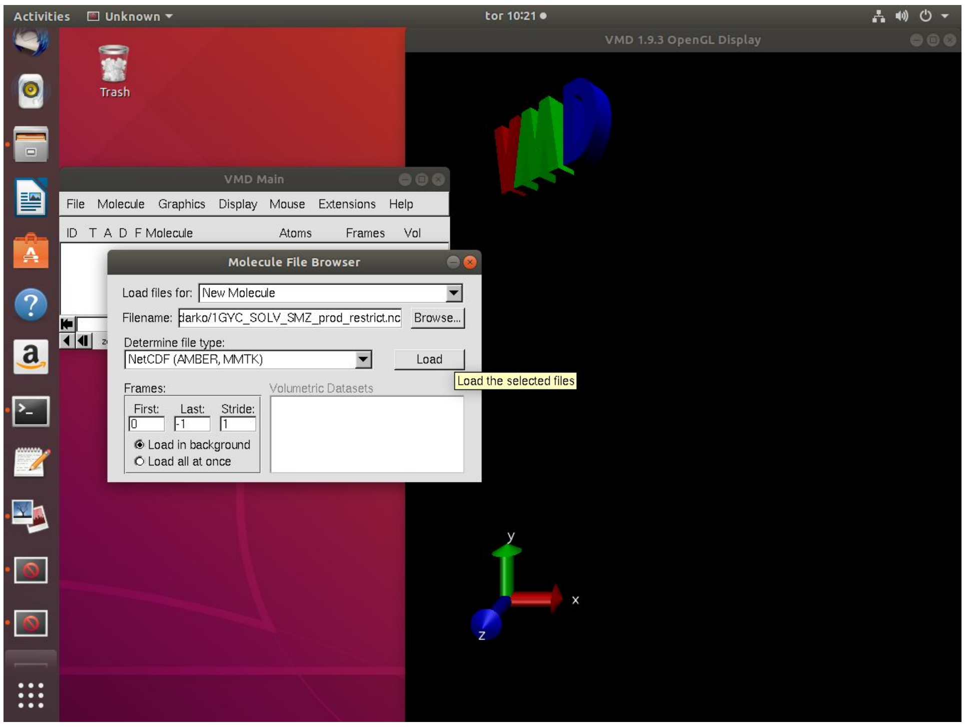
In MD, the hardest thing is to know which files to import into VMD. No conversion from amber is needed. Choose the Amber trajectory file format for import of a .nc file, and the prmtop file and import it in the Amber parameter/topology file format. Then, you should be able to see a movie-like trajectory with every frame representing the coordinates taken every ntx:th iteration (see the section about the sander/pmemd routine for more information on what this means). This was done a ubuntu virtual machine, MacOS looks a bit different of course but has the same core functionalities. After you have clicked New Molecule… under File, you should be greeted with this:



Load files for: New Molecule
Filename:
Determine file type: Automatically
Frames:
First: Last: Stride:
Load in background
Load all at once

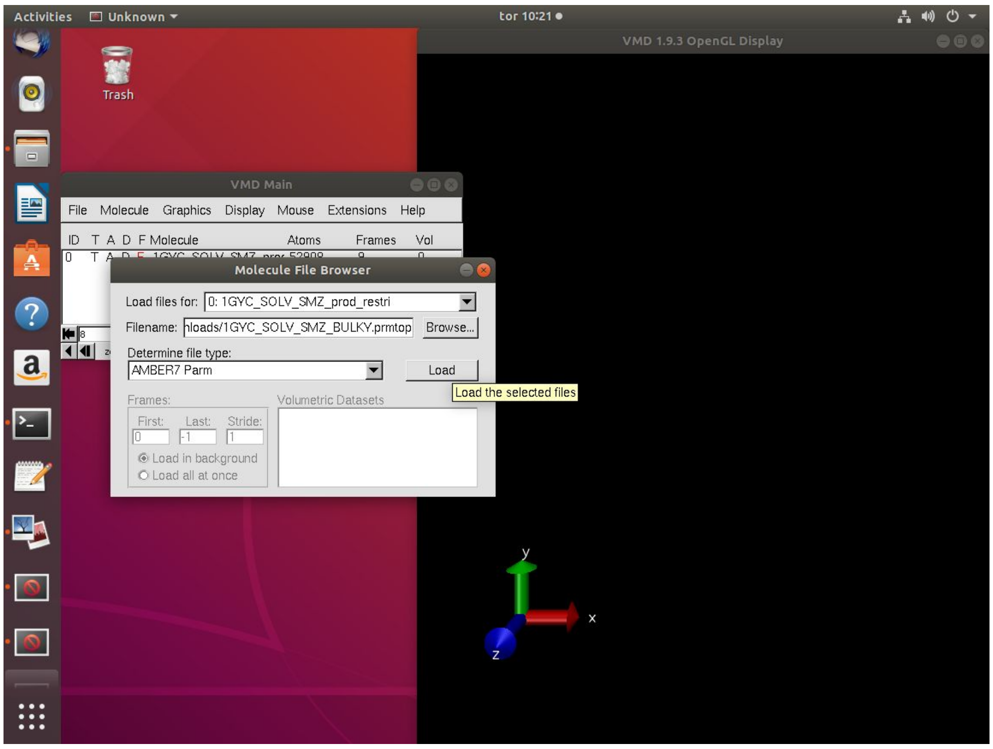
```
Info) Dynamically loaded 2
Info) /usr/local/lib/vmd/pl
vmd >
```

When selecting a file in .nc format (netCDF), vmd should determine the file format automatically.



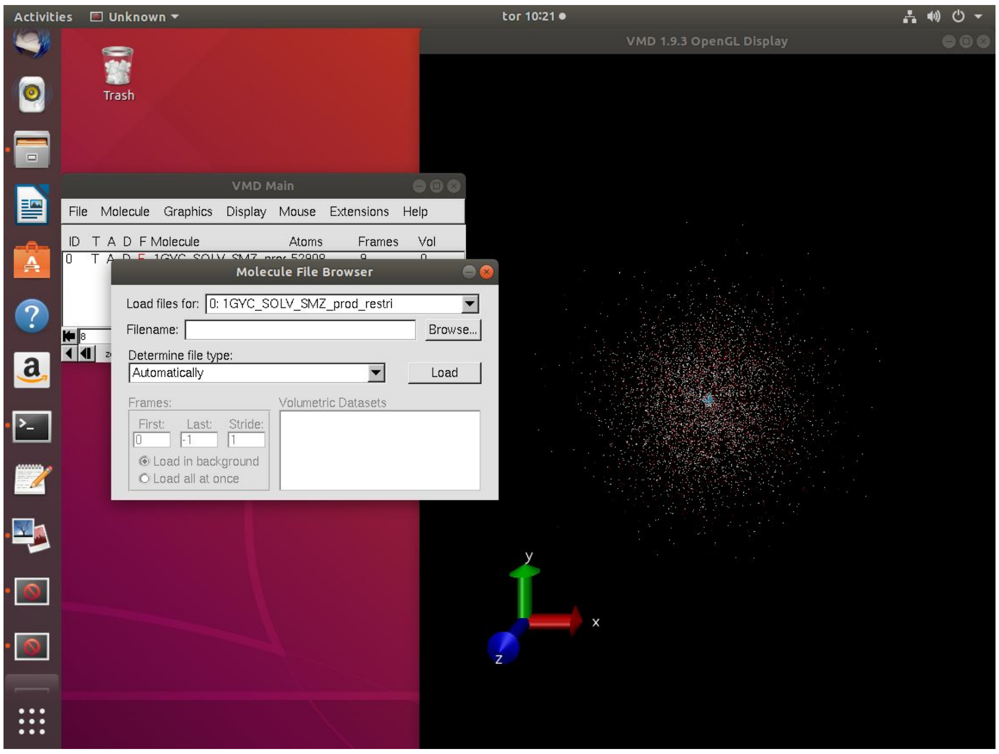
Load files for: New Molecule
Filename: darko/1GYC_SOLV_SMZ_prod_restrict.nc
Determine file type: NetCDF (AMBER, MMTK)
Frames:
First: 0 Last: -1 Stride: 1
(●) Load in background
( ) Load all at once

I chose one 8-frame simulation I did a while ago, named 1GYC_SOLV_SMZ_prod_restrict.nc. Thanks to the binary nc formatting, the file type could be determined by vmd. Sometimes, you need to this manually. The file type should always be NetCDF (AMBER, MMTK).



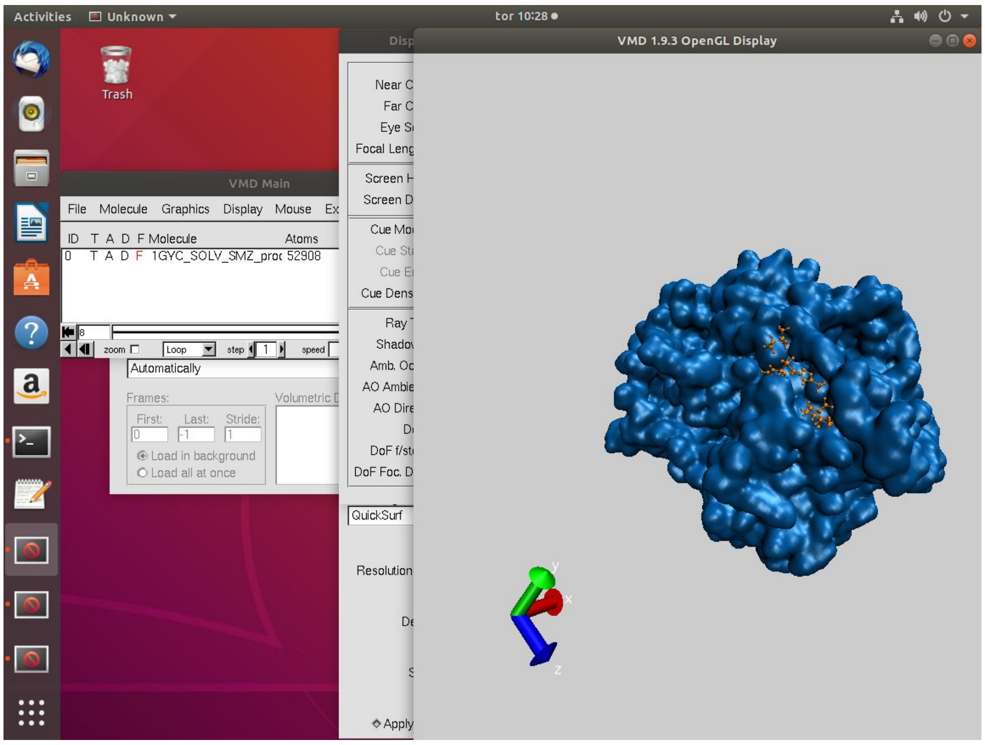
Load files for: 0: 1GYC_SOLV_SMZ_prod_restri
Filename: loads/1GYC_SOLV_SMZ_BULKY.prmtop
Determine file type: AMBER7 Parm
First: 0 Last: -1 Stride: 1

When you load the .nc file, nothing should happen. You are loading a trajectory with new coordinates, but no information for VMD to relate to any interesting quantities, such as atom types and bonds (topology). So for that, we need to load the .prmtop file that we started the simulation with (again, the section about sander explains how this is utilized). It should recognize it as an amber parameter file.



Load files for: 0: 1GYC_SOLV_SMZ_prod_restri
Filename:
Determine file type: Automatically
First: 0 Last: -1 Stride: 1

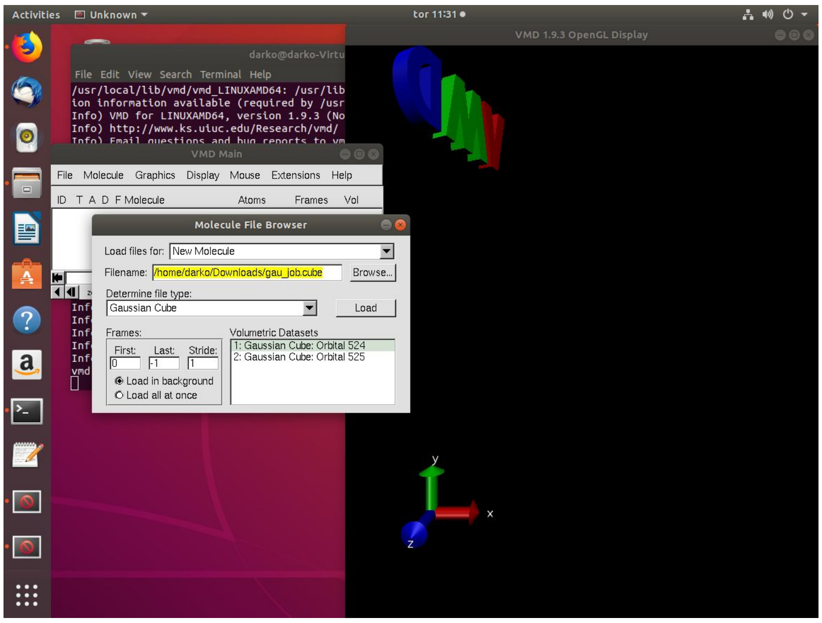
Now, you should be able to see your system. Mine consists of a solvated protein. Now, you can really play around with the Graphics and Display options. I got this nice 3D surface out of it:



```
ID  T A D F Molecule           Atoms
0   T A D F 1GYC_SOLV_SMZ_proc 52908
```

4. Usage with QM

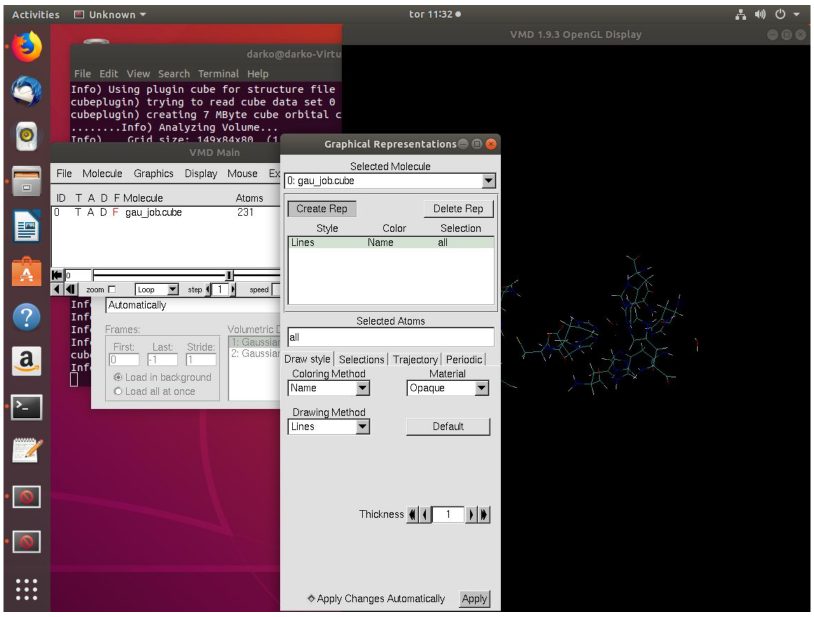
Repeat the process for usage in MD. You should see the same molecule as if you loaded a MD output file, however, when running the QM, you can use a couple of commands in e.g. gaussian to get cube files from a checkpoint (.chk file). These cube files contain volume data interpreted as isosurfaces in VMD, along with the molecule information, which is separate from the molecular coordinates. This is perfect for visualising orbitals.


Activities
Unknown
tor 11:31
VMD 1.9.3 OpenGL Display
darko@darko-Virtu
File Edit View Search Terminal Help
VMD Main
File Molecule Graphics Display Mouse Extensions Help
ID T A D F Molecule Atoms Frames Vol
Molecule File Browser
Browse...
Load

```
/usr/local/lib/vmd/vmd_LINUXAMD64: /usr/lib
ion information available (required by /usr
Info) VMD for LINUXAMD64, version 1.9.3 (No
Info) http://www.ks.uiuc.edu/Research/vmd/
Info) Email questions and bug reports to vm
```

Load files for: New Molecule
Filename: /home/darko/Downloads/gau_job.cube
Determine file type: Gaussian Cube
Frames: First: 0 Last: -1 Stride: 1
[✓ Load in background] Load all at once
Volumetric Datasets:
1: Gaussian Cube: Orbital 524
2: Gaussian Cube: Orbital 525

When you open the VMD visual interface, follow the same steps as in MD to load a molecule. Load your .cube file into VMD, and it should be determined in the correct format (gaussian cube). Into this particular cube file, I have loaded two molecular orbitals (by the *MO=HOMO,LUMO* option in the *cubegen* command), and I load them separately into VMD.



```
Info) Using plugin cube for structure file
cubeplugin) trying to read cube data set 0
cubeplugin) creating 7 MByte cube orbital c
........Info) Analyzing Volume...
Info)     Grid size: 149x84x80  (1
```

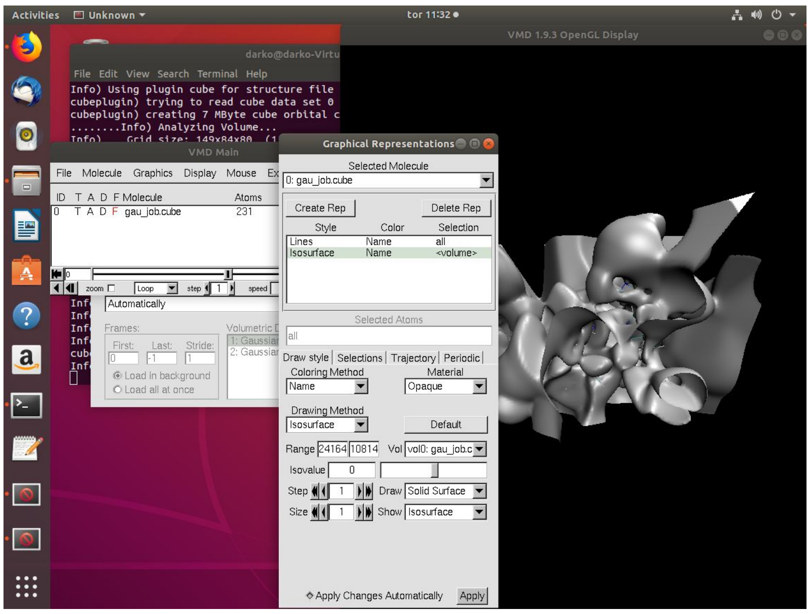
Graphical Representations — Selected Molecule: 0: gau_job.cube

Style	Color	Selection
Lines	Name	all

Selected Atoms: all

Coloring Method: Name; Material: Opaque; Drawing Method: Lines; Thickness: 1

You should see only the molecular structure in lines or CPK. To create the isosurfaces that correlate to the orbitals. To do this, go to graphics and choose graphical representations. The following window should appear.



Graphical Representations — Selected Molecule: 0: gau_job.cube

Style	Color	Selection
Lines	Name	all
Isosurface	Name	<volume>

Selected Atoms: all

Coloring Method: Name; Material: Opaque; Drawing Method: Isosurface

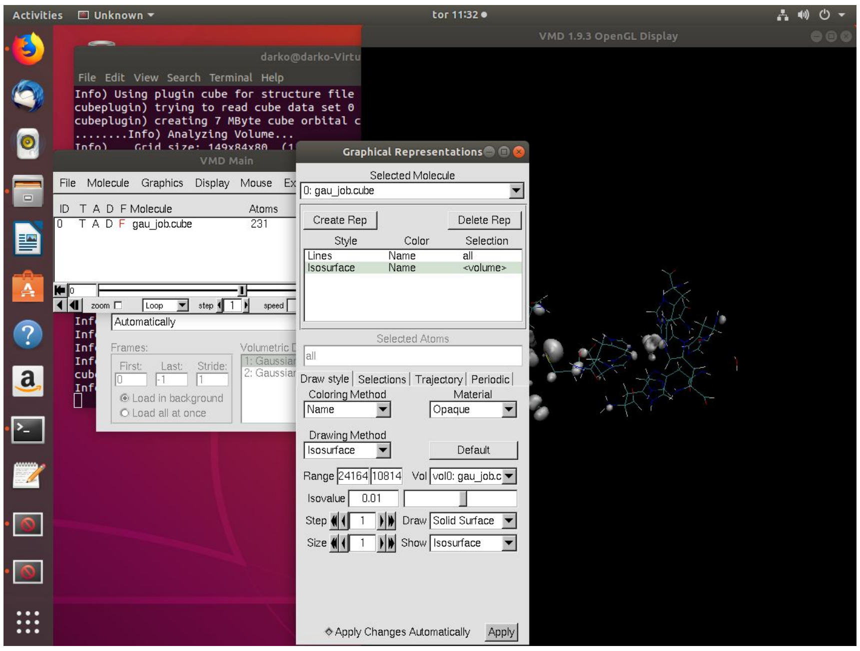
Range 24164 10814; Vol vol0: gau_job.c

Isovalue 0

Step 1; Draw Solid Surface

Size 1; Show Isosurface

Press “create Rep.”, and choose “Isosurface” under the dropdown list. Selection should be set to <volume>. A plethora of options should appear. Your setup should look like mine. Pay special attention to the options “Draw” and “Show”. The isovalue is set to zero. Now we will change it.


Activities
Unknown
tor 11:32
VMD 1.9.3 OpenGL Display
darko@darko-Virtu
File Edit View Search Terminal Help
VMD Main
File Molecule Graphics Display Mouse
Graphical Representations
Create Rep
Delete Rep
Draw style Selections Trajectory Periodic
Default
Apply Changes Automatically
Apply

```
Info) Using plugin cube for structure file
cubeplugin) trying to read cube data set 0
cubeplugin) creating 7 MByte cube orbital c
........Info) Analyzing Volume...
Info)     Grid size: 149x84x80  (1
```

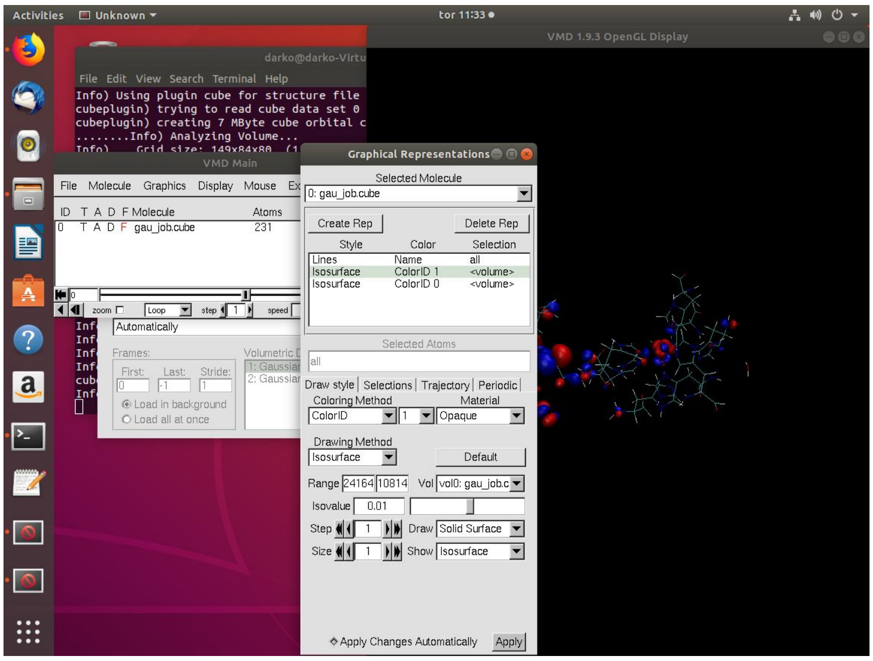
ID	T	A	D	F	Molecule	Atoms
0	T	A	D	F	gau_job.cube	231

Selected Molecule: 0: gau_job.cube

Style	Color	Selection
Lines	Name	all
Isosurface	Name	<volume>

Selected Atoms: all
Coloring Method: Name — Material: Opaque
Drawing Method: Isosurface
Range 24164 10814 — Vol vol0: gau_job.c
Isovalue 0.01
Step 1 — Draw Solid Surface
Size 1 — Show Isosurface

The orbitals that we want to visualise are electron iso-probability curves, which means that the electron in question has a probability that is larger than the iso-value to be inside of the isosurface. (iso means the same) Thus, the larger the iso-value the smaller the orbitals. Usually an iso-value between 0.01 and 0.1 suffices for a good picture of the electronic density.


Activities
Unknown
tor 11:33
VMD 1.9.3 OpenGL Display
darko@darko-Virtu
File Edit View Search Terminal Help
VMD Main
File Molecule Graphics Display Mouse
Graphical Representations
Create Rep
Delete Rep
Draw style Selections Trajectory Periodic
Default
Apply Changes Automatically
Apply

```
Info) Using plugin cube for structure file
cubeplugin) trying to read cube data set 0
cubeplugin) creating 7 MByte cube orbital c
........Info) Analyzing Volume...
Info)     Grid size: 149x84x80  (1
```

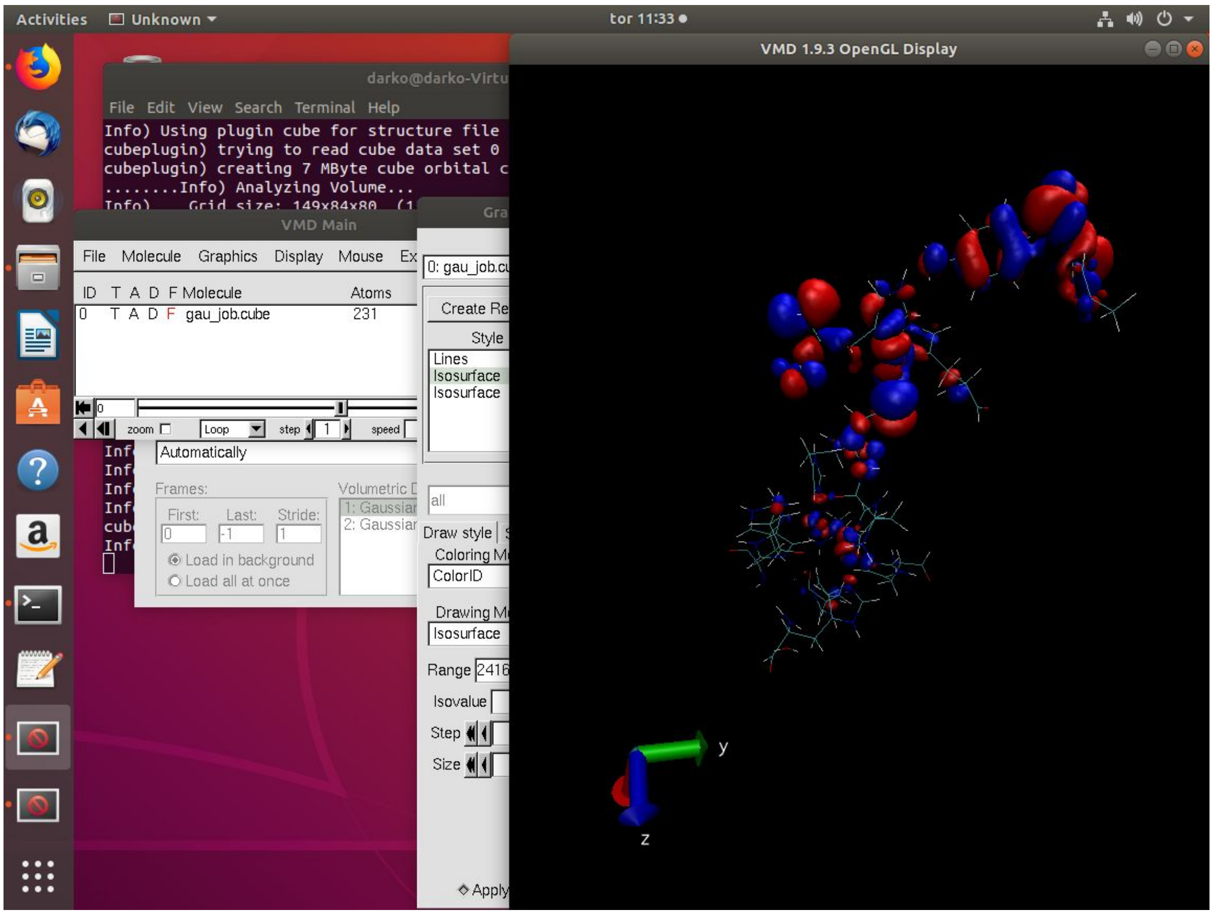
ID	T	A	D	F	Molecule	Atoms
0	T	A	D	F	gau_job.cube	231

Selected Molecule: 0: gau_job.cube

Style	Color	Selection
Lines	Name	all
Isosurface	ColorID 1	<volume>
Isosurface	ColorID 0	<volume>

Selected Atoms: all
Coloring Method: ColorID 1 — Material: Opaque
Drawing Method: Isosurface
Range 24164 10814 — Vol vol0: gau_job.c
Isovalue 0.01
Step 1 — Draw Solid Surface
Size 1 — Show Isosurface

If we set the Coloring Method to ColorID, we can change the colour of the orbitals. Creating a new representation and setting the iso-value to -0.01 will give you the other phase of the orbitals, and you will then have a full picture of the electron distribution.



```
Info) Using plugin cube for structure file
cubeplugin) trying to read cube data set 0
cubeplugin) creating 7 MByte cube orbital c
........Info) Analyzing Volume...
Info)    Grid size: 149x84x80  (1
```

ID	T	A	D	F	Molecule	Atoms
0	T	A	D	F	gau_job.cube	231

zoom Loop step 1 speed

Automatically

Frames: First: 0 Last: -1 Stride: 1

Load in background / Load all at once

Volumetric D: 1: Gaussiar 2: Gaussiar

0: gau_job.cu

Create Re

Style: Lines, Isosurface, Isosurface

all

Draw style

Coloring M: ColorID

Drawing M: Isosurface

Range 2416

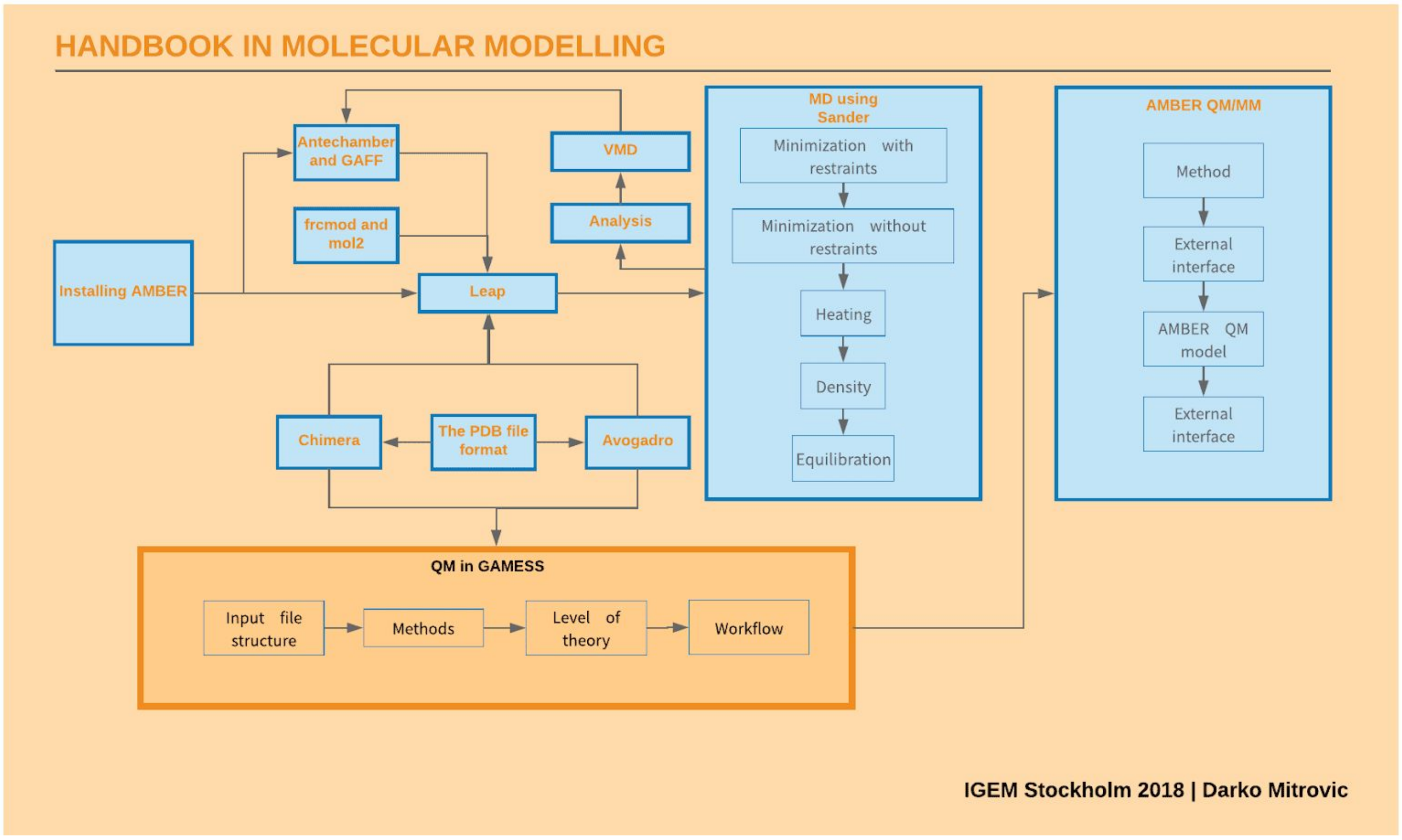
Isovalue

Step

Size

Apply

This was my final picture, without any other display settings.



QM in GAMESS

GAMESS (*General Atomic and Molecular Electronic Structure System)* is a software for Quantum mechanical (QM) calculations, and is free for academic use. This section will clear out the syntax, logic structure, and input format in GAMESS. If you are not familiar with computational quantum chemistry, either read up on the subject or only read section 6 in this chapter.

1. Introduction

Welcome to the Quantum Mechanics (QM) chapter. Beware, from this point, things are much less trivial than what they have been in the Molecular Dynamics and classical simulations chapters. Using the tools provided by GAMESS, Gaussian16 and NWChem, you can calculate everything from free energies of association and dissociation (affinity), saddle point and frequency calculations (activation energy and reaction coordinate), perform 3D-scans for potential energy surfaces (for reaction characterization), geometry optimizations (for preparation of files as well as electron density and orbital analysis), integration in the time domain (for excitation energies or UV/VIS absorbance spectra), as well as optimized frequency calculations for parametrizations for Molecular Dynamics simulations (see the previous chapter).

As mentioned, the QM part of your project can cost a lot of time, but gives a whole new picture of the actual biochemistry of the enzyme. Since this is the deepest theories available, some of them physically exact (but with computational approximations), which means that development in these areas can lead to virtually exact calculations even of large systems. But with such a powerful tools comes a lot of problems.

Firstly, understanding the methods and choosing methods require at least a background in organic chemistry, and a good understanding of orbitals and basic quantum chemistry, i.e. what the Schrödinger equation is, what the electronic Hamiltonian is and why calculations for any other system than a one-electron system never can be exact, with the current iterative approach.

Secondly, using the methods can sometimes be unintuitive, which is why making a guide for this is very challenging. Sometimes, to solve convergence issues or calculation errors, some creativity is due, for example trying a completely different approach. Here, it is good to point out that you should never be afraid of trying new things. Sometimes, if your calculations are taking very long and still aren’t converging, using more advanced basis sets and augmented functionals can do the trick. In the end, the more you understand about computational quantum chemistry, the easier solving problems will be.

Thirdly, analysis of results are crucial - never trust the output blindly, even if you have a good energy value - the density may be completely wrong, and for spin unrestricted runs, spin contamination often occurs. This means that choosing your model well and having control over what parameters you want to find is crucial to how impactful your results can be to your overall project.

Before we begin, there are a few technical prerequisites. You have to have Avogadro installed (see the section on Avogadro) for input generation, and be familiar with Chimera for the pre-processing of your protein-sized system.

Additionally, a good point when working with a quantum chemical approach is that you try to mimic nature, so never forget that your modelled system has basis in a natural system. The more you know how your natural system behaves, the more you know how your system *should* behave. Analysing differences between your model and nature can often give a lot of information of in what direction you should develop your model (and sometimes even give fundamental information or clues about the natural system).

2. Background

If you are completely new to QM, it would be a good idea to read up on a few concepts, or just trust that I know what I'm doing and follow the workflows below. The final flowchart in this chapter is a sort of "too long, didn't read" summary of the how to use different methods in different situations. If that doesn't work, going back to each section to troubleshoot is an easier way to go about QM than trying to understand everything from the start. In many ways, the proper way to approach this part of your project is with an open mind and a will to learn.

As you can see, even the handbook required a short user instruction when it comes to QM, so it is quite complicated. There are many programs that can perform quantum mechanical calculations. Among these, GAMESS is a good, non-commercial (free!) program that has many functionalities implemented, e.g. tools for excited state geometry and energy calculation, many functionals for DFT, Time-dependant DFT as well as an array of state-of-the-art basis sets. Note that GAMESS is for academic use only, so do not use it if you want to make a profit out of your project in the future. For that, I would recommend Gaussian16 or, my personal favourite, NWChem. Also, amber contains quantum mechanical routines that must be run in GAMESS binaries, which means that for QM/MM, GAMESS is absolutely necessary to install. In Sander, it is activated using the *ifqnt=1* keyword in the *&cntrl* list, and a separate *&qmmm* list, then if *qm_theory='EXTERN'*, you can add a third namelist called *&gms*, and add your options for the qm calculation. The standard is usually HF STO-3G SCF calculation, so it is very important to change it. Sander will then generate a GAMESS input file with the specified options in the &gms namelist.

After installing it on any device, rungms can be used in the terminal with the input file as an argument, for standalone quantum mechanical calculations. For integration in the time domain, for example real time time dependant DFT (RT-TDDFT), I would recommend that you buy NWChem, that is much more straightforward than this.

```
rungms <input_file_name>
```

Most supercomputers also have this program, and it usually runs on both cpu's and gpu's. For supercomputers though, I would recommend to use *ssh -x11* or similar so that you can access the visual interface gaussview, where you can generate the

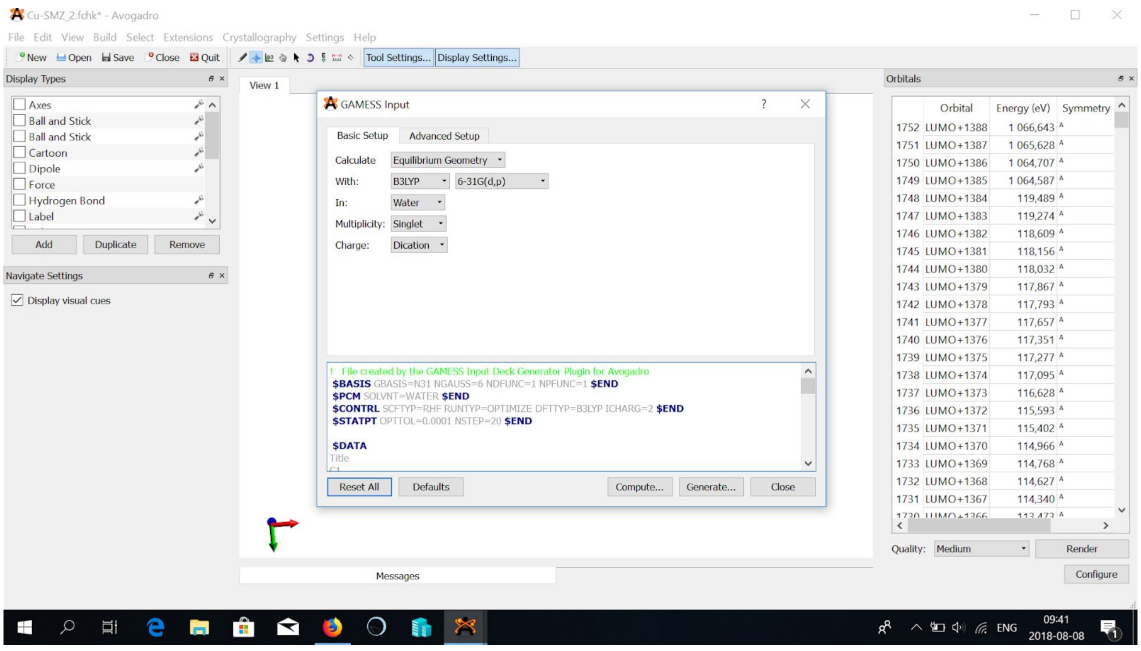
GAMESS is a software that does Quantum Mechanical calculations. This is crucial because it is only with quantum mechanical or EVB based models that we can model the forming or breaking of bonds. (EVB takes 1 month to learn and 2 months to calibrate for a protein-sized system, so it is not recommended if you don't have an expert on EVB within your iGEM team). This guide will only cover how to use the different models and methods and what kind of information you can get from this. Remember that there virtually is no limit to what chemical reaction or interaction you would like to model; however, beware of electron transfer or quantum mechanical tunneling phenomena, since these require a very good understanding of the chemical and physical properties of your system, to condition the correct environment, and diffuse corrected cluster functionals. Also, for more intricate kinetics, you need to do several computationally costly calculations, so please research everything carefully before you decide on what system you would like to set up to solve your issue.

It should also be mentioned, to whomever hasn't read the first chapters about classical modelling, that a major downside with QM is that the calculations take a long time to complete. Some of my calculations took 2 weeks on a supercomputer with GPU-volta nodes and 2x5120(CUDA) and 2x640(Tensor) processors, so be careful with starting too lengthy and heavy runs(Yet again, the system I investigated contained 4 unique organometallic complexes with a couple of diffuse orbital that stretch across 25 Å with over 700 electrons, so no wonder it took so long). While most things converge with enough computing power, it is much more effective to condition your system to work in an environment that is sufficiently simple, but accurate enough.

3. Input file structure

First of all, we need to find how our system looks empirically. For longer projects, I would recommend to first run the calculation as exactly as you can, and then try to reproduce those results with much simpler settings. This is vital if you want to get good results in higher throughput.

Before we begin with the individual methods, it could be good to go through the input file structure, and the format to present it in. If you don't want to copy the XYZ-coordinates from your .pdb file I suggest you use Avogadro for input generation. It also gets a lot of the basic structure of the input file set for you to edit it, instead of building it from scratch. Through the extensions… option you can choose Gamess. While the same functions and levels of theory are included in both GAMESS-UK and Gamess, the input file structure differs a lot, and GAMESS-UK is much more similar to NWChem. The following structure is easier to follow because it has less density and more straightforward namelists. Avogadro will generate a .in file for you depending on what options on the GUI you choose. Your screen should look something like this:



Calculate: Equilibrium Geometry
With: B3LYP 6-31G(d,p)
In: Water
Multiplicity: Singlet
Charge: Dication

```
!  File created by the GAMESS Input Deck Generator Plugin for Avogadro
 $BASIS GBASIS=N31 NGAUSS=6 NDFUNC=1 NPFUNC=1 $END
 $PCM SOLVNT=WATER $END
 $CONTRL SCFTYP=RHF RUNTYP=OPTIMIZE DFTTYP=B3LYP ICHARG=2 $END
 $STATPT OPTTOL=0.0001 NSTEP=20 $END

 $DATA
Title
```

	Orbital	Energy (eV)	Symmetry
1752	LUMO+1388	1 066,643	A
1751	LUMO+1387	1 065,628	A
1750	LUMO+1386	1 064,707	A
1749	LUMO+1385	1 064,587	A
1748	LUMO+1384	119,489	A
1747	LUMO+1383	119,274	A
1746	LUMO+1382	118,609	A
1745	LUMO+1381	118,156	A
1744	LUMO+1380	118,032	A
1743	LUMO+1379	117,867	A
1742	LUMO+1378	117,793	A
1741	LUMO+1377	117,657	A
1740	LUMO+1376	117,351	A
1739	LUMO+1375	117,277	A
1738	LUMO+1374	117,095	A
1737	LUMO+1373	116,628	A
1736	LUMO+1372	115,593	A
1735	LUMO+1371	115,402	A
1734	LUMO+1370	114,966	A
1733	LUMO+1369	114,768	A
1732	LUMO+1368	114,627	A
1731	LUMO+1367	114,340	A
1730	LUMO+1366	113,473	A

Quality: Medium

There are two symbols that signal the type of information that will follow - ! and $. ! is used for commenting wherever you like in the input deck, and $ signals a namelist. All namelists must be closed using the $END card. Avogadro automatically makes the following comment:

! File created by the GAMESS Input Deck Generator Plugin for Avogadro

Next, we need to define the basis set completely using the $BASIS namelist. Below, the classical 6-31G(2d,p) has been defined by its' composition. Instead of having one keyword for that, in gamess, you can build your functions yourselves. Thus, a 7-31G(2d,p) could just as easily be created. Below, we have a split-valence basis set, as developed by John Pople. The first number represents the number of gaussian core orbital basis functions (NGAUSS=6), while the second and third numbers represents how many basis functions are used for the valence orbitals respectively (constructed with double-zeta valence). For triple-zeta, you simply add another number. This is done using the GBASIS=N31 keyword. For triple valence, use the GBASIS=TZV keyword. Lastly, to add d or p polarization functions, NDFUNC=2 and NPFUNC=1 are used in this case. DIFFSP=.TRUE. adds diffuse p-functions. All of these keywords can get complicated really fast, but the following basis set should work in most cases:

$BASIS GBASIS=N31 NGAUSS=6 NDFUNC=2 NPFUNC=1 $END

Then, you can optionally add a Polarizable continuum model, where the solvent can be set to behave in ways similar to, for example water. This is added by the $PCM namelist. Do not forget to close the namelist with the $END card.

$PCM SOLVNT=WATER $END

Next is the $CONTRL list. This is the most important namelist, since it defines your calculation in terms of what you want to do (RUNTYP), what level of theory you would like to use, and overall system options, for example charge. Below, I have used SCFTYP=RHF, i.e. with a restricted SCF calculation with regards to spin; however, despite the RHF option, you can still use the DFTTYP to specify a DFT functional, i.e. you are still performing a DFT calculation. Here, I used a B3LYP functional. The last keyword can be good to use, ICHARG=2, which essentially specifies the total number of electrons. It is easier to do this in relation to the number of protons (which elements you have used in your system), than to define the absolute number of electrons. ICHARG is therefore simply the total charge of the system. For other systems than singlets, you need to specify your multiplicity with the MULT=N, where N is an integer. Again, you have to include the $END card.

$CONTRL SCFTYP=RHF RUNTYP=OPTIMIZE DFTTYP=B3LYP ICHARG=2 $END

The statpt controls the iterative options, for example step length and number of steps. Often times, if your geometry doesn't change and the energy has converged, you have an adequate number of steps. I wouldn't recommend straying much from these options.

```
$STATPT OPTTOL=0.0001 NSTEP=20 $END
```

Now comes the part that Avogadro does the best. The $DATA list specifies the geometry of your system, i.e. all of your coordinates for all of your atoms. The *Title* and *C1* options should not be changed since they are trivial. Then, there's a 5-column table. The columns contain information about the (1) Atom type (determines the number of protons in your system, and indirectly through the ICHARG option, the number of electrons), (2) atom mass (can be changed for isotopes) (3) X-coordinate, (4) Y-coordinate and lastly, (5) Z-coordinate. Again, $END is needed.

```
$DATA
Title
C1
O 8.0 14.70163 -5.22234 -3.01771
S 16.0 14.72202 -3.89024 -1.94780
O 8.0 16.08787 -3.00082 -2.46899
C 6.0 7.29504 2.78779 2.98611
C 6.0 8.38202 2.18983 3.56762
N 7.0 7.61563 3.04200 1.68572
C 6.0 8.87564 2.64499 1.43180
N 7.0 9.33579 2.10700 2.57469
Cu 29.0 6.41195 3.81391 0.23443
O 8.0 -10.37893 1.07192 -2.73613
H 1.0 15.36987 -5.63802 -0.13679
H 1.0 16.04435 -5.26948 2.74964
H 1.0 16.27901 -1.87660 5.21148
H 1.0 17.30639 -3.28855 5.05026
H 1.0 15.61337 -3.47926 5.47351
$END
```

4. Methods

To solve the schrödinger equation for the wavefunction, you need to find the Hamiltonian, whose eigenvalues are the energies that composes our system. First of all, we use the Born-Oppenheimer approximation, which assumes that, due to the large mass difference between electrons and protons, the wavefunction of the proton will be essentially static for the protons in relation to the electrons. Thus, we assume that we can freeze the position of the protons and treat them as point charges in relation to the electrons. Also, some calculations utilize the Tamm-Dancoff approximation, for example time dependant DFT frequency calculations.

In this case of the frozen core approximation, the form of our Hamiltonian for every electron has nuclear-electron attraction energy and kinetic energy. It also has a correlation exchange term. If you have ever had a lecture in the electronic Hamiltonian matrix, then you know that the electronic correlation is a problem. Since we don't know the exact electronic geometry (the electrons are delocalized in their orbitals), we cannot know at which distances the electrons interact, and thereby we cannot know the forces on each electron, and thus the potential energy of each electron in correlation to each other. Therefore, approximations to gauge the exchange correlation energy have to be in place. A common misconception is that the exchange correlation energy is the biggest contributing factor to the total energy of the system, but in fact, it is often less than a few percent of the total energy. For precise calculations and systems where you have steric clashes, electron-electron interactions have to be taken into consideration.

4.1. HF

HF, or Hartree-Fock, is a so called level of theory. Level of theory in a quantum chemical sense refers to how accurate we are choosing to describe our system. These are the general approximations or methods that we use to find the wavefunction. Since we know that the eigenvectors of the Hamiltonian form a non-interacting set of basis vectors (attributes of a hermitian matrix), we can diagonalise the matrix in order to find its eigenvectors. The eigenvalues of the Hamiltonian are of course the energies of the corresponding eigenvectors (incidentally, these eigenvectors correspond to the wavefunction). In Hartree-Fock, we choose this set as a set of non-interacting orbitals, and we can therefore ignore the electronic exchange contribution, i.e. our electronic matrix is diagonal. (The study of the off-diagonal elements - called the coupling elements - plays a huge role in charge transfer reactions, and can be seen as the geometric overlap of the donor and acceptor spin-orbitals).

The Hartree-Fock method, as mentioned, uses the slater determinant, which is a mathematical construct that is based on the constraint of antisymmetry in electronic systems. If the slater determinant is too close to zero, the program usually warns for degeneracy. Since electron degeneracy is a phenomenon that almost exclusively happens in very dense stars and systems under high pressure and very tight density, this is usually an error with the specified system. Often times, it points to a wrong geometry, so tweaking the initial coordinates will often lead to a better conditioned, solvable system.

To summarize, Hartree-Fock is a method where the exchange correlation term is zero. This is electron-electron repulsion energy, so excluding it yields a guess that is most of the time lower than the expected (Sometimes, you can have very diffuse occupied-virtual orbital interactions which lowers the energy of this term significantly). Underestimating the energy of a system by not accounting for electron-electron repulsion energy can be very inaccurate, especially for systems under 100 atoms, and should never be used.

4.2. DFT

Density Functional Theory (DFT) is the leading method in quantum chemistry. The main point of this approach is that you can, theoretically, model your system to infinite accuracy, i.e. the Hamiltonian is exact for a multiple-electron system, however the methods used to solve the resulting differential equations come with their own approximations. Solving these equations exactly will give you an exact model. The exchange correlation term in this case is a functional of the electron density within a certain region (see the LDA and GGA approximations to this theory), that takes the electron repulsion from the true electron density.

These equations are called the Kohn-Sham equations, and build on the idea of having a fictitious system with a series of fictitious electrons that are non-interacting, but placed so that the charge density results in the exact electron density. However, computationally, these methods can become very costly depending on the functional that we use. Therefore, we can used local density approximations (LDA) or generalized gradient approximations (GGA).

4.2.1. LDA

The local density approximation treats the local electron density as a uniform electron gas, which affects a point charge in it. This approximation does not consider the gradient of the charge distribution; however, empirically, we can see that it holds well for slowly varying gradients. This is thus not good for polarized systems, or multi-molecular systems where long range interactions such as hydrogen bonds, ionic or dipole interactions. Exchange correlation functionals Ceperly-Alder 1980 Quantum Monte Carlo simulation, or Chachiyo 2016. For a list of all functionals, please visit the GAMESS homepage.

4.2.2. GGA

Generalized Gradient Approximation, on the other hand, also takes the gradient of the electronic density into consideration, i.e. the Exchange correlation is a function of both the position and electronic charge gradient. This means that we can take not only the point value of the electronic density at every position, but also take into consideration the immediate local, and global electronic density gradient. This eliminates most of the problems with simple LDA. Common problems include overestimation of the gradient contribution, but can be normalised by a linear combination of other exchange correlation terms. This is where we enter the realm of very good Hybrid Functionals. Again, a quick visit to the GAMESS manual or homepage for the DFT functionals sections will suffice.

4.2.3. Hybrid Functionals

There are also hybrid functionals, that, as you would guess, are a hybrid between different functionals. The most widely used functionals are of course found under this category, since they can encompass any wanted property, at the cost of computational time. CAM-B3LYP or PBE0 for long range, and regular B3LYP for ordinary systems where orbitals are localised within the same molecule, are very good functionals that should work in all cases.

4.3. SCF

The very concept of the schrödinger equation involves the solving of a differential equation. For this, we employ iterative numerical methods to solve the system. When it is solved, we say that it is *converged*. A converged system means that it is self-consistent, i.e. that the recursive functional does not change any of its properties or values. This means that, in the case of DFT, the density is a function of the density. When the difference in the value for each iteration has decreased under the convergence threshold, the system is considered to be solved.

There are several convergence schemes, and they do not always work. The take home message is that if your initial guess is good enough, it will most likely converge. The steepest-descent and Newton-Raphson quadratic schemes or Direct Inversion of Iterative

Subspace (DIIS) usually don't differ too much in whether they succeed to converge the system or not, however sometimes they differ greatly in speed. I would recommend trying SCF=QC, which is a quadratic convergence scheme that works for systems with large initial energy gradients, i.e. if you have steric clashes or positioned the system so that there are very strained bonds or anti-binding Molecular orbitals present in the initial Huckel guess. With that said, we can smoothly venture into the land of initial guesses for molecular orbitals.

4.4. Initial MO Guess

The initial guess of any quantum mechanical is essential for how fast and if the calculation will converge. If the initial guess is too far away from the SCF, it is crucial to condition the solution matrix to fit your system better. In geometric terms, it means that, while the coordinates of the atomic cores are static (Born-Oppenheimer approximation), the electronic density differs. Frankly, a good initial guess of the electron density leads to a system that is easier to solve. It is from this electron density that the system will be evolved, instead of trying a new electron density ab initio for every iteration. The energy gradient of the system will then be evaluated, and you can then gauge how to evolve the system to a lower energy (for energy optimization), or evolve the system in the time domain (excitation and electron transfer theory). The standard initial guess in GAMESS is the Huckel guess, which is an electronic density constructed by the individual atomic orbitals of every atom. This is of course a wide approximation, but usually suffices for most calculations.

The problem arises when you have several different subsequent runs, and you would like to carry over information about Molecular orbitals from run to run. Then, you have to import the molecular orbitals from a previous run, given by the .vec file. It contains the vector information about the molecular orbitals. This is done by importing the vector file using the GUESS=MOREAD keyword and the separate $VEC name list (see GAMESS manual, the structure of this list changes quite a bit with every rendition).

5. Level of theory

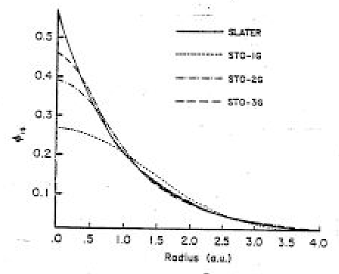
As mentioned previously, the level of theory does not always describe how accurately a problem is solved, but rather how fast/slow the method can reach a SCF (self-consistent field, see previous section for more information). This means that it is important to make the theory *fit* your system economically, instead of using the highest possible level of theory. Not only does it decrease your computational time, but you will model your system with higher accuracy. Methods of a high level of theory sometimes make approximations not very suitable in your specific case, however "lighter theory" models maybe don't. Therefore, having good control over what you are doing and how you attempt to solve your problem is key modelling accurately.

5.1. Basis Sets

Basis sets are a way of describing the Molecular Orbitals in gaussian function. A gaussian function contains several different constants that can be fitted to the behaviour of the molecular spin orbital, in wavefunction form (i.e. before we take the norm and square the result). The form of this gaussian function (a gaussian type function, for any mathematically illiterate, is a function $f(r)$ on the form Ce^{-kr^2} , where C is independent of r). The following equation gives all the parameters that can be fitted:

$G = x^l y m z^n e^{-r^2}$

Since it is not shifted on the r-axis, it is symmetrical along it. This means the peak of the wavefunction is at the centre, commonly only associated with the classical 1s orbital. Evidently, we need a more sophisticated basis. Therefore, we can add several gaussian functions with multiple different peaks, and thus, we can represent our molecular orbitals more accurately. This approach is represented by the G suffix, for Gaussian functions. Most commonly, 6-31G is used, i.e. 6 G-functions for atomic orbitals, and double zeta valence with 1 and 3 G-functions respectively. However, on this form, you can simply decrease this peak, however it will obviously still be somewhat symmetric. You need polarization function to correct for those effect. For polarization, I would recommend the correlation-consistent polarized functions basis sets called cc-pVNZ, where N=D,T,4,5,6, and represents the number of zeta valence functions (D is double, T is triple). Below is a list of usable basis functions (they are not that many).



STO-3G - Basically just three gaussian function linearly combined. Never use this since its' to crude to describe molecular orbitals. Works fairly well for atomic systems.

6-31G - This used to be the state of the art basic basis set, before correlation consistent functions came into the picture. It still works reasonably well for most systems, and is a good low-cost basis set. I personally always start with this basis set and see how well I can reproduce observed variables with this set.

6-311G - This basis set is of course a triple zeta valence basis set, and can be useful for heavier atoms, however, if you have heavier atoms such as metals or even metals complexed with histidine or other heme-like groups, you need polarization functions, and the additional zeta valence function will not help you significantly.

6-31G++ - The plus signs on the regular 6-31G basis set represents two polarization function, one for d and one for p orbitals. This is very useful for large molecules with polarized bonds, and breaks the geometric symmetry present in STO-3G basis sets.

cc-pVDZ - This state of the art (as of 2018) class of basis sets are very important for the development of basis sets in quantum chemistry. They are often referred to as producing true orbital geometries and energies, as opposed to 6-31G, that often gets the energy right. This is not very surprising considering that the exchange correlation term that depends on the density in DFT does not affect the energy that much, however the SCF while be driven to the wrong solution with 6-31G.

aug-cc-pVTZ - This basis set exemplifies what you can do with the cc-pXXX basis sets. We can add augmented functions to increase the importance of long-range interactions in, for example very polarized, hydrogen bonding, or intermolecular systems. There are a lot of other basis sets, but I would recommend to have a closer look at these ones if you want to indulge in other, less mainstream basis sets.

5.2. Functionals

As I mentioned before, DFT functionals are essential in solving complex systems accurately, where an accurate description of the electron density functional is due. In density functional theory (DFT), the Exchange correlation term is a function of the electron density functional. This garners a complex recursive function, where the electron density (via the norm of the wavefunction squared) depends on the electron density (that results from the functional). The form of this functional determines how well the system can be solved. An all-encompassing functional (e.g. infinitely expanded taylor polynomial) does not exist, so we have to fit the functional with the system we are using.

There are a number of different functionals (exclusively hybrid functionals) that work with varied accuracy on different systems. Below I have listed a couple of functionals and a short description about when to use them. They are described by the *DFTTYP=<dft_functional>* keyword

B3LYP - the B3LYP functional is one of the most used and most successful density functionals for DFT. Basically, it is a very good all-round functional, and should be used initially in any case.

CAM-B3LYP - Basically, this is a functional with long range correction (CAM). This should only be used when values from PBE0 don't give good orbitals, but B3LYP also fails to capture the essential long range interaction. For charge transfer, I would highly recommend this or the PBE0 functional.

PBE0 - Finally, PBE0 is very successful in gauging long distance behaviour, and can be used in tandem with Polarizable continuum models (PCM, see the section on input structure for more information). However, not all interactions are captured good here either (e.g. for a system with important pi-cation interactions, for which you'll need M06 class of functionals or those with D3 correction)

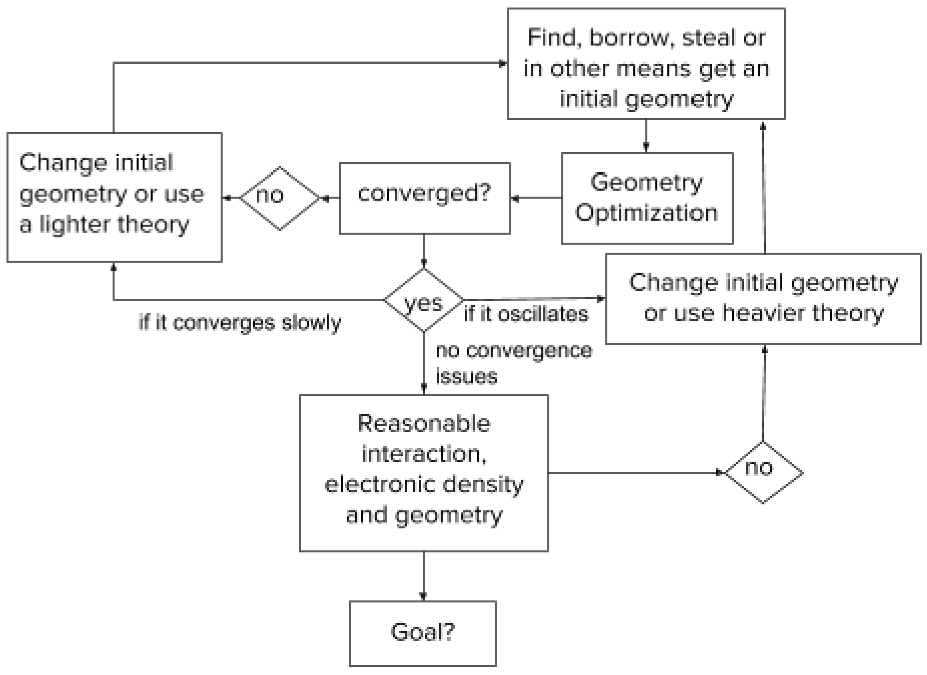
B97 - A less costly version and precursor of B3LYP, and usually gives good results.

N12 - If you don't want to use hybridized functionals to explore how GGA applies to your system, you can use N12. This is however, only recommended for program analysis and if you want to compare how GGA holds up to hybridized (nearly exact) functionals in different cases.

6. Workflow

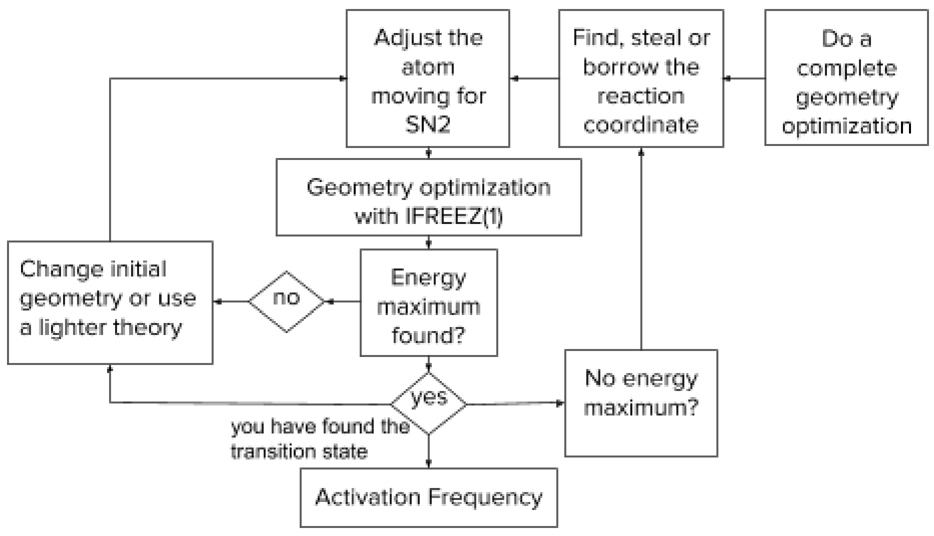
6.1. Geometry optimization

The first step of any QM run is a geometry optimization, to relax the system from local energy maxima. At every iteration, we need to converge to a complete SCF, and calculate the energy gradient with respect to the core geometry. Then, we rearrange the atoms, to create a new "initial" geometry guess and see how the energy changes. This is done automatically by the geometric optimization routine, specified in the RUNTYP keyword. The following workflow is typically used for any run, only that the geometry is very important. Often times, it is in the geometry change that you can see if the system has been modelled correctly - if you begin close to the true geometry (say, started from NMR data or crystal structure), the geometry should not change significantly. For example, when you have delocalised, planar π-systems, and have not used the appropriate model, the planar initial guess will condense and crumble into an sp3-state. In this way, you can usually see what part of the modeled system you need to change by viewing toward what solution the calculation tends to, in a similar fashion to the previous example. I have attached a picture where you can see the general workflow for this step. An important note is that if your geometry changes drastically in the beginning of the simulation, freezing some central atoms that you don't want to move too much can prove very useful. For more information on how to do this, see the section about IFREEZ(1) in the next section. Just be careful that you do not constrain the system so much so that movement is completely disabled (for example the peripheral atoms of an organometallic-complex). It should be noted that this is rarely the first solution to convergence or faulty geometry problems.


Find, borrow, steal or in other means get an initial geometry
Geometry Optimization
converged?
no
Change initial geometry or use a lighter theory
yes
if it converges slowly
if it oscillates
Change initial geometry or use heavier theory
no convergence issues
Reasonable interaction, electronic density and geometry
no
Goal?

6.2. Activation energy

For activation energy, we need two basic steps - first a scan for a saddle point (we can do this in GAMESS manually using ifreeze for a number of geometry optimization), and then a calculation of the activational frequency for this complex, which can then be correlated


Do a complete geometry optimization
Find, steal or borrow the reaction coordinate
Adjust the atom moving for SN2
Geometry optimization with IFREEZ(1)
Energy maximum found?
no
Change initial geometry or use a lighter theory
yes
you have found the transition state
No energy maximum?
Activation Frequency

to the transition state complex and the total activation energy of the complex itself. In the attached picture there is a complete flowchart for the initial procedure (finding the transition state geometry, i.e. the saddle point along the reaction coordinate). For every step of geometry optimization the complete workflow of the previous section needs to be done.

There are two very important input files that you have to construct. One is for the geometric optimization with a freezed constraint, and one is where the activation frequency is calculated.

The saddle point calculation is done by manually changing the position of the exchanged atom or molecule (for example in diels-alder reactions), and freezing the moved atom or complex at a fixed distance or cartesian position. If you would like to freeze a distance (i.e. a bond (even though there may not be one in the visualisation program)) switch from cartesian input coordinates to ZMAT format. This produces a z-matrix that contains the connecting information between atoms, without any absolute movement in cartesian space. Below, I have included the complete input information provided by GAMESS on the matter.

```
IFREEZ = array of coordinates to freeze.  These may be
         internal or Cartesian coordinates.  For example,
         IFREEZ(1)=1,3 freezes the two bond lengths in
         the $ZMAT example, which was for a triatomic
           $CONTRL NZVAR=3 $END
           $ZMAT IZMAT(1)=1,1,2,  2,1,2,3,  1,2,3  $END
         while optimizing the angle.

         If NZVAR=0, so that this value applies to the
         Cartesian coordinates instead, the input of
         IFREEZ(1)=4,8 means to freeze the x coordinate
         of the 2nd and y coordinate of the 3rd atom.

         See also IFZMAT and FVALUE in $ZMAT, and IFCART
         below, as IFREEZ does not apply to DLC internals.

         In a numerical Hessian run, IFREEZ specifies
         Cartesian displacements to be skipped for a
         Partial Hessian Analysis.  IFREEZ can pertain to
         EFP particles, but only during RUNTYP=HESSIAN,
         where the 6 translational and rotational degrees
         of freedom of each EFP come AFTER the QM atom
         coordinates.  For more information:
         J.D.Head, Int.J.Quantum Chem. 65, 827, 1997
         H.Li, J.H.Jensen
            Theoret. Chem. Acc. 107, 211-219(2002)
```

The activation frequency can be calculated a bit easier, by simply changing the RUNTYP keyword to RUNTYP=IRC, which gives the infrared frequencies. Opening this in visualisation program will give you a visual representation of about 100 different frequencies, depending on how many bonds your system contains.

6.3. Electron transfer using TDDFT

I would not recommend that you perform this step. If, however, an electron transfer reaction is crucial to your modelling, I will not stop you. Even if you have a “simple” HOMO-LUMO one electron excitation or transfer, the dynamic properties of the orbitals when occupied or virtual changes the energy of the system. This means that inferring excitation energy information from the HOMO-LUMO energy gap straight from a geometric optimization run will not give you the appropriate accuracy and will sometimes differ by 2eV even though you have a completely accurate electron density. (With the correct geometry but wrong orbital conformation, I got an error of 20eV between the HOMO-LUMO when the correct value with cc-pV5Z TDDFT was 0.1eV. That’s an error of 20000%.) I would also not use GAMESS for this, as mulliken charges and other interesting properties (for example RT-TDDFT) can be calculated using the NWChem package. Either way, if you have more than 10 atoms, consider finding a supercomputer to do this for you.

To perform a TDDFT excitation calculation, the TDDFT variable has to be specified to EXCITE (i.e. TDDFT=EXCITE) in the $CONTRL name list. Then, you can include a separate $TDDFT name list for further specifications. The following input is usually what you will need:

```
$TDDFT IROOT=1 NSTATE=10 MULT=3 $END
```

The IROOT=1 option refers to the root state, i.e. the state that should be geometrically optimized. This can sometimes be used to find the reorganizational energy of the reaction, according to Marcus theory for electron transfer.

The NSTATE=10 option gives the number of excited states to be investigated, i.e. which excitation frequencies to be calculated. They are, of course, ordered by energy. Again, don’t forget to include the $END card.

The MULT keyword is only available if a closed shell Hartree-fock method has been specified by SCFTYP in the $CONTRL name list. The multiplicity of a singly excited state often is either 1 or 3. (always 3 for one electron charge separation, i.e. 2 unpaired electrons, one in the donor and one in the acceptor orbital, if the ground state has MULT=1).

6.4. QM parametrization

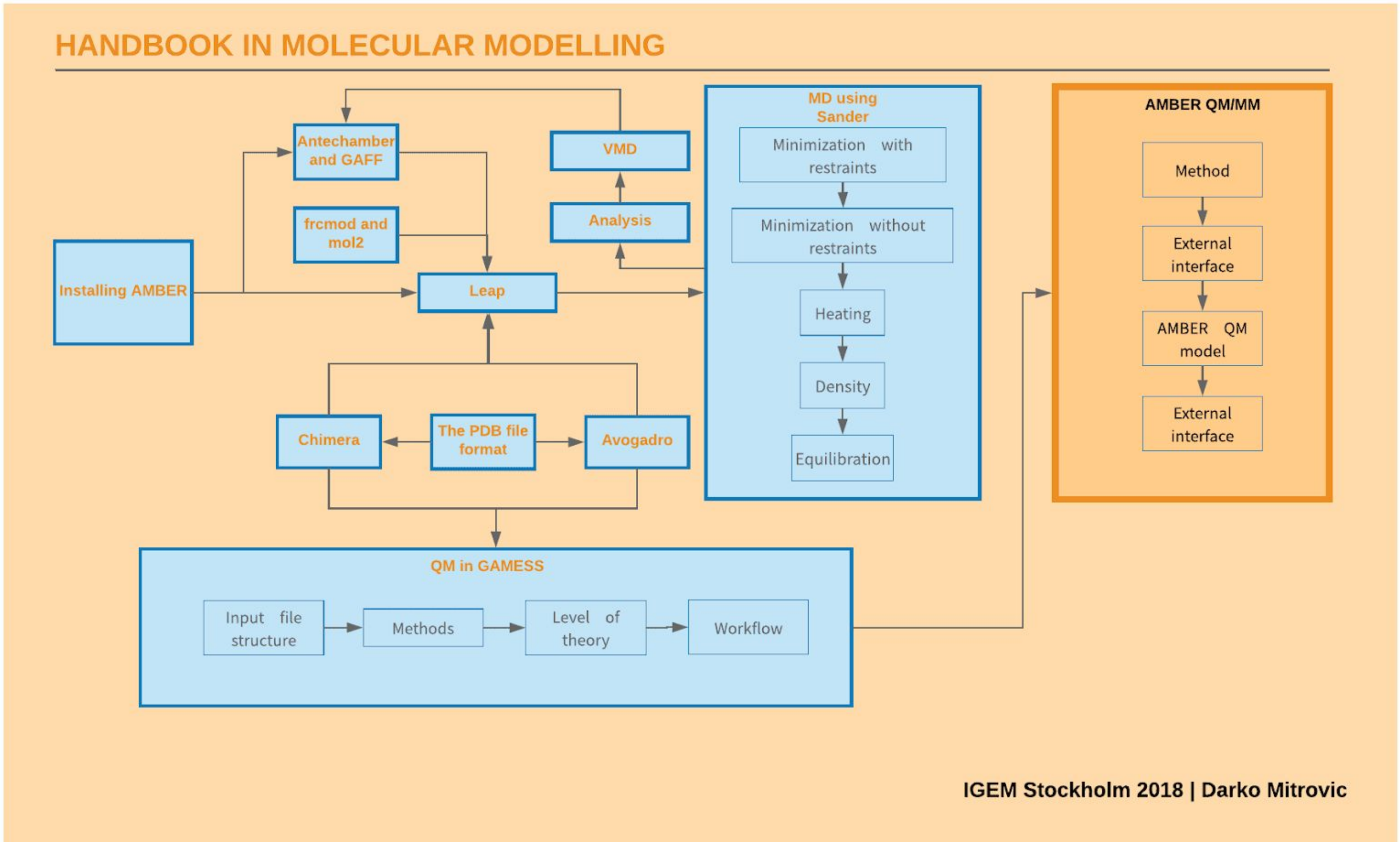
Sometimes, an organic molecule or a non standard residue is included by the user or in the initial crystal structure. We have to perform a so-called manual parametrization of this molecule in order for the Molecular Dynamics simulation to run properly. This is quite a time-consuming step, and it can sometimes be very difficult to extract information after running the simulation, since we are most of the time doing this completely manually.

There is routine called MCPB.py that one can use, however, sometimes, weird naming issues can come up (no atom can have an atom name over 3 letters and all atoms within a parametrization have to be unique, as does the individual residue names). Solving naming problems is definitely not as trivial as it sounds, and it can take days to find what has gone wrong. Also, it only works with certain file formats, so they have to be generated from the GAMESS, or preferably Gaussian output files. For more information about this, I would refer you to the Seminario method (by Joe Seminario) and the MCPB.py routine in ambertools.

In principal, a QM parametrization is very easy to conduct, and for experienced GAMESS or Gaussian users can be done in two consecutive steps, however, we still need our runs to converge, and the results to be reasonable. For larger structures that are strained, and can involve bond breaking, this approach can come with inaccuracies. If you have closed off your system reasonably well, this should not be a problem. If your system is too large to cut off in this fashion, consider using QM/MM (see the last section for more information on this)

The first step is to perform a geometry optimization. In GAMESS, all the data on every bond will be printed in the output files. From this, you can infer important equilibrium data (See the section on frcmod and mol2). Essentially, dihedral angles, bond angles and bond lengths are all calculated here. If you are having trouble navigating the huge output file, visualising this data in avogadro or MOLDEN could be easier, and then measuring these parameters. Then, you can enter them into the frcmod file as described in the frcmod and mol2 section. . Also, van der Waals parameters can be inferred from the optimized geometry, (both attractive and repulsive).

For force constants, we need to perform a frequency calculation. See the paragrah above for more information of how to do that step. From the output file, we can get all the force constants for every bond with reasonable energy. For smaller molecules, every bond that should be treated as such should have its own frequency in both the direction of the bond, the angle of the bond and the dihedral angle (spinning single bonds). It is vital to not forget that, while a bond may not be covalent, even a long distance interaction counts as a bond in the frcmod file (only that the equilibrium bond distance is larger and the force constant smaller, but it is still significant data).



AMBER QM/MM

The QM/MM module in ambertools is a very useful tool if you need to simulate a large system over time or a simple energy minimization, however need to account for some quantum mechanical effects. In this case, we can specify a small region for QM calculation, and then treat the rest of the atoms using MM-methods. This is much more effective than using PCM or solvent surface models since we can include the charge induced effect from every atom within the cutoff, and therefore, there is no need to tailor a PCM to suit the inner protein environment. For this chapter, it is required to have read the *QM in GAMESS* and *MD using Sander* chapters.

1. Introduction

Quantum mechanical/molecular mechanical simulations (QM/MM simulations) are often done where a large biomolecule (a protein, enzyme or nucleic acid) is of interest, however, while all interactions are important, there are a few interactions that cannot be modelled classically, and we have to employ quantum mechanical models instead. Of course, one could simply take the small QM region and perform a QM calculation on it. While it captures the essentials within the region of interest, it fails to capture interaction between this and the rest of the biomolecule, that tends to be quite large. Often times, polarization effects and dipole-dipole interactions can be captured quite well thanks to QM/MM. A parallel between the interface between the QM and MM regions and the Polarizable Continuum model (PCM, see GAMESS chapter for more information) can be drawn. In the same way that the solvent or environment of a molecule can be modelled implicitly (i.e. without the atoms), we model it here explicitly, but with simplifications to a classical level. Often times, the orbital configuration can change drastically in QM/MM. Also, integration twice of the forces in the time domain can give us the time-dependant evolution of a system, similar to the evolution of a classically modelled system in MD (Molecular Dynamics).

QM/MM simulations are great for characterization of reaction, and only including the chemically or photophysically active site (for enzymes or affinity proteins chemically active sites, and for fluorophores, photophysical activity) Since we change the positioning of the QM cores along the simulation, we can actually simulate a whole reaction, if it happens to satisfy the energetic evolution of a system (starting in the transition state geometry should probably make the system relax into the products or the reactants). Since we can change the electronic configuration using the QM functionality, we can take bond breakage and electronic excitation into consideration easier. A good initial guess is crucial here, and simply hoping for the reaction to occur will not suffice.

2. Method

The QM/MM method revolves around the merging of the two most powerful molecular modelling methods - one for its scope and one for its accuracy in modelling electronic configuration - and as a result, we get the best of both worlds. On paper, this sounds terrific; however in reality, there are a few issues that are related to how we actually perform these calculations.

First of all, we have the issue of that once we have selected our QM-region, how we will treat it in the QM-program, since if one QM and one MM atom are connected covalently, the multiplicity of the QM atom would change since we are breaking a bond. Thus, we need to “cap” these bonds using virtual hydrogens. It is therefore of high importance that we never break a bond that is highly polarized or contributes to the electronic configuration significantly in any other way than a bond to an H would. The capping can be followed through the log files of the external QM program, since it lists all the atoms, and marks the virtual hydrogens with an asterisk (*). The QM/MM interface isn’t a hard surface, which means that sometimes, the electronic orbitals can stretch further out than the atomic QM-region and onto MM-atoms (that are not considered as nuclei in the QM calculation) and therefore, peripheral orbitals can often not be trusted because of steric clashes and other electronic interactions. If this is the case, include a larger amount of atoms in the QM treated region. As I mentioned before, the choice for the QM region is absolutely crucial for the success of the simulation. One should also remember that for many features of conformational analysis, a good MM force field may be better than a semiempirical or DFTB quantum description. In choosing the QM/MM boundary, it is better to cut non-polar bonds (such as C-C single bonds) than to cut unsaturated or polar bonds.

Secondly, the total energy contribution from the QM region can be added onto the total MM energy, however, the nonbonded contributions to the Hamiltonian of the QM region from MM atoms has to be considered as well. This is where the “solvent” or PCM effect comes into place. By defining nonbonded van der waals and charge parameters for the MM atoms, we can create a highly accurate electrostatic surface for the QM region. This is much more accurate than overall solvent models, since they encapsulate non-globular structures poorly and can of course not take all minute differences in charge and van der waals radii that are present within a protein. This effect alone changed my HOMO-LUMO gap in one instance from 13eV to 0.1. The reason for such a huge difference is because of that the program arrived at a completely different state altogether. These nonbonded terms also affect the environment, as the MM atoms will be affected by van der waals parameters calculated internally in amber (similar to the MCPB.py routine, see the section on QM in GAMESS for more information). Also, the energy gradient of the system also tells us how strained the QM region is, and affects the forces and total energy gradient of the QM/MM system. This affects how the system is iterated. In short, while we do not include the full atoms of the MM region in the QM region, we need to include all of the nonbonded interactions, which is where we essentially get our high accuracy from.

Lastly, we need to connect how the QM program iterates to form a SCF to how that is handled by the MM part, and how we can actually simulate a system through time. In a normal MD simulation, the energy gradient, or forces, are calculated for the whole system, and then integrated over dt, which is specified in the AMBER .in file. This means that we have to calculate the energy gradient of the QM region to perform an iteration in the “MD” (production) simulation. We cannot, however perform a geometry optimization, as I have so frequently preached about in the previous chapter, because if we begin to move the nuclei of the QM region based on the QM calculations only, they tend to drift outwards due to the electron-nucleic attractions from the van der waals parameters. Therefore, we fix our atomic positions during the QM calculation, i.e. we only perform one SCF iteration (we find the solution to the schrödinger equation to find the current electronic density). The energy gradient is redundant within the QM calculation, however is needed later, because then we perform our MD iteration with the usual settings. This means that, compared to MD, this takes a lot longer because of the need of doing a complete SCF for every step, so beware of what level of theory you are using since the calculations can get quite lengthy.

3. QM/MM interface

As mentioned, QM/MM can be performed using AMBER. For this, we use the regular sander .in files. For every step of a standard MD simulation, (minimization, heating, density, equilibration and production), we can, in addition to the regular keywords in the &cntrl namelist, add a ifqnt= keyword. Below, you can see an example of the &cntrl namelist with the added ifqnt=1 keyword. In this case, this is a production simulation, which is the final step of the MD simulation, where the following happens;

Firstly, we are not performing a minimization, so imin=0, and there are no need for restraints, so ntr=0. These two options are generally needed for minimization runs. Further, we have a few pressure conditions (1 atm, specified by pres0, and taup=2.0) we are cutting interactions over 7.0 Å. This only applies to the MM parameters, not to the QM region, which is completely separate from this (of course internally, because the cutoff from the MM nonbonded terms still apply to the nonbonded terms in the Hamiltonian for the QM system). For bigger systems with many charges, use cut=20. We are doing this at about 28 ^{o}C, or exactly 300K, so tempi=300.0 and temp0 is also 300.0. nstlim and dt=0.002 define the length of the simulation, and the ntpr, ntwx, ntwr and ntxo simply define printing options of the simulation.

```
Production simulation
&cntrl
 imin = 0, irest = 1, ntx = 7,
 ntb = 2, pres0 = 1.0, ntp = 1,
 taup = 2.0,
 cut = 7.0, ntr = 0,
 ntc = 2, ntf = 2,
 tempi=300.0, temp0 = 300.0,
 ntt = 3, gamma_ln = 2.0,
 nstlim = 1000, dt = 0.002,ioutfm=1,
 ntpr = 1, ntwx = 1, ntwr = 1, ntxo = 2, ifqnt=1,
/
```

Lastly, we have the ifqnt keyword. I requires an additional namelist, specifically the *&qmmm* namelist, it contains information about both the overall information of the qm region, first specifying the actual qm region with the regular AMBER syntax (see the analysis chapter for more information on this). The keyword for this is qmmask=’’. In the example below, I have selected a number of residues ranging from 63 to 506. It is important to visually inspect your structure so that the qm region is consistent and self interacting (i.e. I would advise against having two separate qm regions in the same calculation, because they may interact nonetheless), and also to select the correct residues. Usually, it is much easier to select residues, however a more accurate cutoff for the qm region can be achieved using the selection for atoms.

Another important keyword is what defines our level of theory (qm_theory=’’). Most of the internal quantum mechanical Hamiltonians within AMBER is semi-empirical, which means that they are conditioned to work best under protein like conditions and similar environments. Involving complicated organometallic complexes gravely complicates this picture. The available hamiltonians are, as stated in the amber18 reference manual; Options are AM1, RM1, MNDO, PM3-PDDG, MNDO-PDDG, PM3-CARB1, MNDO/d (same as MNDOD), AM1/d (same as AM1D), PM6, DFTB2 (same as DFTB), and DFTB3. The dispersion correction can be switched on for AM1 and PM6 by choosing AM1-D* and PM6-D, respectively. The dispersion and hydrogen bond correction will be applied for AM1-DH+ and PM6-DH+. I would recommend PM6-DH+ if you absolutely do not want to perform an advanced qmmm calculation in an external program. For simple calculations, the standard PM3 suffices. When an internal hamiltonian is chosen, you can specify a couple of more options, such as the qmshake=1, qm_ewald=1 and qm_pme=1, which is thermodynamic properties of the system and particle mesh ewald options.

```
&qmmm
 qmmask=':63 :66 :109 :111 :162 :206 :265 :332 :334 :337 :338 :395 :398 :400 :452 :453
:454 :458 :500 :501 :503 :504 :505 :506',
 qmcharge=6,
 qm_theory='EXTERN',
                            ! qmshake=1,
                           ! qm_ewald=1
                          ! qm_pme=1
/
```

Above was an example of where I have set the qm_theory to ‘EXTERN’, which means that an external program conducts the qm part of the calculation. Then, we cannot have any of the input options that I have marked as comments (using the ! sign). Upon inclusion of the qm_theory=‘EXTERN’ option, we need to include an additional namelist - &gau, which in the case below is for any version of Gaussian. We want to do this with &gms, which is for GAMESS. Here, we need to define the level of theory according to the internal syntax within GAMESS, or whatever qm program you are using. I have left this challenge to the reader, as a final conclusion of the most interesting hybrid method of molecular modelling.

```
&gau
 basis=6-31G**(d,p)
 method=B3LYP
 mem=40GB
/
```