So You Want to do Materials Research:
a guide to aBuild and the skills you need to use it

Lydia Harris and Eli Harris

June 27, 2019

Abstract

The python module aBuild is meant to automate the process of building an MTP model for a paticular system. The documents presented in this folder/book are meant to be a reference guide to bash, git, aBuild, etc. Disclaimer: we wrote this to help ourselves remember these things, and as a favor to Brother Nelson. We reserve the right for some of it to be incomplete or confusing at times. If you have questions, don't be afraid to ask us or Brother Nelson.

1 Getting Started

To do computational research you will need to learn what the command line is and some bash commands–see Appendix B: Bash Commands. Go learn your commands it will make your life easier!!

Now that you know the basics of the command line environment it is time to utilize those skills to log into the supercomuter. If you have not yet made an account, go to marylou.byu.edu click on request an account. Your faculty mentor (most likely Brother Nelson) will need to approve your account. The whole process may take a couple of days. Once you have an account, if you are using Mac or Linux simply open the terminal and use the ssh command (see Appendix B: Bash Commands.). If you have a Windows machine you have several options: you can use the Ubuntu app, Windows power shell, Windows terminal, etc.

1.1 Make directories

Once logged into the supercomputer you will need to build the following directories (see Appendix B: Bash Commands):

- `\home\codes\aBuild`
- `\home\bin`
- `\home\environments`
- `\home\system-species` (i.e. AgPt)

If you don't know what system you're gonna study, talk with Brother Nelson before you make your system-species folder. He'll help you find one to study.

1.2 Download aBuild

Now it is time to download aBuild. The python module aBuild is meant to automate the process of building a MTP model for a paticular system. The module is continuously undergoing improvements. To have version control we use git. If you are unfamiliar with git go read Appendix D: Basics of Github. Git is an extremely helpful tool and used across many different organizations. It's a skill you should add to your resume once you are familiar with it. Once you've learned how to, fork/download aBuild from https://github.com/lancejnelson/aBuild (see table 8 for a quick refresher).

1.3 Virtual Environment/bash_profile

A convenient way to install and run software is in a virtual environment. Because aBuild is a software package, it needs to be installed for you to be able to run it. We will install it in a python virtual environment. This is how you will create a virtual environment and install aBuild:

Table 1: Making a virtual environment and installing aBuild

Command	**Description**
cd environments	go into that directory
python −m venv name_of_env	make the environment
emacs .bash_profile	edit your .bash_profile
`function workon {`	add these lines to it:
`source ~/environments/$1/bin/activate`	$1 refers to the name of
`}`	your virtual environment
ctrl+x ctrl+c y	quit the editor
source .bash_profile	update your .bash_profile
workon name_of_env	enters your environment
python install -e ~/codes/aBuild	install aBuild in your virtual environment
ctrl+d	exits your environment

Since we used the .bash_profile to make the virtual environment work, I'll talk a little more about that now. Your .bash_profile holds the settings that are applied each time you log in to the supercomputer. You can load software modules, change settings, and a lot of other things. Some helpful additions to your .bash_profile are given here (actually I basically just copied and pasted my entire .bash_profile here), although these were the current software modules on Marylou in Dec 2018, and I reserve the right to have old versions of the modules loaded, since it is probably not Dec 2018 anymore.

- export PATH= $ PATH:~/bin
- PS1="\u:\w\$ "
- module load libfabric
- module purge
- module load compiler_intel/13.0.1
- module load gdb/7.9.1
- module load gcc/6.4
- module load python/3.6
- export MAKESTRX=~/bin/makestr.x
- export GETKPTS=~/bin/getKPoints
- export ENUMX=~/bin/enum.x
- export F90=ifort
- export HISTSIZE=100000
- set completion-ignore-case on
- function workon {
  source~/environments/$1/bin/activate
  }
- alias hh='history | grep '

1.4 Copy other software packages

You'll need enum.x (.x means it's an executable file), makestr.x, and getKPoints. There may be copies of these in the group folder that you can copy to your bin. Ask Brother Nelson for help with this if there's not.

2 Run aBuild: Training Process

Now that you have your account all set up, you are ready to start training the model to your specific system. This section will guide you through the different commands needed to train the model.

Your folder will need builder.py and a yaml file in it to run any of these commands. Copy these from `~/codes/aBuild/aBuild/templates/master.yml` and `~/codes/aBuild/aBuild/scripts/builder.py`. Name your yaml something intuitive (e.g. if you're studying the silver gold system, maybe name it "AgAu," etc..). Now you'll need to edit the yaml to make sure everything in there matches your system (such as the title, species, root, potcar directory, potcar versions, potcar setups, mindistance, concs, nconfigs, sizes, etc..). If you need help deciphering the yaml, go see the example.yml file in the aBuildReferenceGuide repository on git. It has a bunch of comments to help you understand what's going on, although you don't really need to understand what's going on to get started.

Now you can start building your model. You can see the steps in table 2 below. aBuild commands start with "`python builder.py **YML**`" and then some tag(s). This prefix is only used for the tags denoted in this table by the '-'.

Table 2: Algorithm steps and their descriptions.

Step	Description
-enum	enumerates the crystalline structures up to sizes specified in your yaml –run interactively
-setup_relax	Builds to-relax.cfg and relax.ini; runs calc-grade –run interactively
qsub* jobscript_relax.sh	mlp relax: needs to-relax.cfg, pot.mtp, and relax.ini; generates: relaxed.cfg, unrelaxed.cfg, and candidates.cfg –job submission, parallel, 10-30 cores, 6-30 hrs
-setup_select_add	Concatenates all of the candidate.cfg_#, selection.log_#, relaxed.cfg_# and unrelaxed.cfg_# into one file each. relaxed.cfg file should get bigger and bigger with each iteration. Also builds a submission script. –run interactively
qsub* jobscript_select.sh	mlp select-add: generates: new_training.cfg; needs: train.cfg, candidate.cfg –job submission, single core, 1-4 hrs
-add	builds A folders in training set and creates jobscript –run interactively
qsub* jobscript_vasp.sh	runs vasp calculations for the selected configurations –array job, 6-30 hrs
-setup_train	Pulls data from VASP folders, builds train.cfg and pot.mtp –run interactively
qsub* jobscript_train.sh	mlp train: needs train.cfg, pot.mtp; generates: Potential.mtp –job submission, parallel, 10-20 cores, 6-12 hrs
go back to step -setup_relax	Repeat until model is fully trained, i.e. all structures relax.

*For Marylou use sbatch instead of qsub **YML** is the yaml file without the .yml extension.

The training process will begin with an empty training set. This will cause the relaxation to terminate for each structure on the first iteration. It will take several iterations before the model is able to relax all the configurations.

The mlp relax step tries to relax the structures in the to relax set. If it extrapolates too much, it stops relaxing it, and adds the structure to a preselected set.

The mlp select add step chooses from structures in candidate.cfg the structures that best fill the "missing" configuration space.

The mlp train step tries fits a "line" to the training data it is given. I say "line" and not line because it is a non linear problem, but if it helps you to think of it like a line, do that.

2.1 Other helpful MTP/aBuild commands

These might come in handy at some point

Table 3: aBuild commands and description

Command	Description
-status	prints a status report for Vasp calculations
-report	creates data report file from completed Vasp calculations
-report -file path/to/file	creates data report file from the configurations contained in the file specified
-chull -file path/to/file	creates a convex hull from the data report file specified
mlp mindist file.cfg	prints the global minimum distance of the atoms to eachother in a .cfg file. Also adds the mindist attribute to structures in the .cfg file
mlp calc-grade pot.mtp train.cfg train.cfg temp.cfg	creates state.mvs, a file needed for relaxation

3 Theory

In this section we discuss what the algorithms are doing or the theory behind the computations. You don't really need to know any of this to get started working on this project, but if you're curious, (as you should be at some point, you're a scientist, after all) here's some info for you.

The main idea is to create a MTP model from Vasp calculations. The Vasp calculations are ab initio or first principle calculations that use DFT and/or DFT+U depending on the specific system. The MTP then is trained by optimization, fitting the coefficients of the basis functions (see section 3.1:MTP for more details). After the model has been trained it attempts to relax the atoms to their happy place. If it cannot relax the atoms and still accurately predict the energy, forces, and stresses, the model then selects more configurations to add to the training set. These configurations are then evaluated using Vasp and the cycle continues until the model is able to relax all of the configurations.

The motivation of training a MTP model comes from the bottleneck caused by Vasp calculations. In searching for configurations to create the convex hull, the configuration space for varying concentrations is so vast that using DFT calculation becomes impractical. This is due to the amount of time the calculations take. By using an MTP training model we are able to explore much more of the configuration space in signicantly less time. This allows us to potentially discover new configurations of a paticular system.

DISCLAIMER: Some of this gets hard to understand. We hope that this document will be a nice introduction without too much scary stuff, but with that being said, don't go scaring yourself off by jumping into all of this too early. Each section has a statement that says "stop reading here if you're not ready." Listen to these statements.

3.1 MTP

MTP is an acronym for moment tensor potentials. It is a basis expansion, and it involves a bunch of tensors. For the purpose of doing this research, you don't really need to know a whole lot about it, except that it's a way to represent crystalline structures that is systematically improvable (meaning you can add more basis functions to get a better and better representation of the crystal) unlike classical potentials, and it is an off-lattice model (it isn't confined to some parent structure, the atoms can be located anywhere with respect to eachother), unlike cluster expansion. Basically it's a brand spanking new (published just last year), way better basis for crystal configurations than the previous methods (classical potentials and cluster expansion, mentioned earlier). Figure 1 is a good cartoon visualization of what we're trying to accomplish with this model.

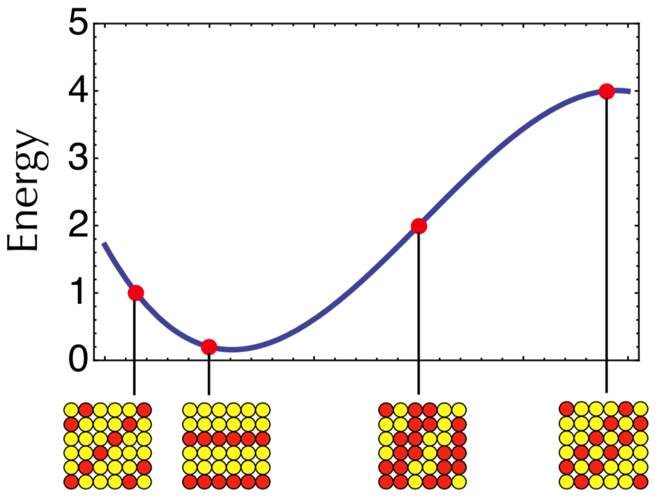

Figure 1: Simple 2-dimensional visualization of configuration space vs. energy with a best fit "line". Each configuration is on the x-axis with it's corresponding energy on the y-axis. In reality this graph would be N+1 dimensional, where N is the number of basis functions, $B_\alpha(\mathfrak{n})$.

This is your quitting point, but in case the previous paragraph isn't enough for you, here's a quick summary of the MTP basis, with a little math, but not very much explanation of the math. Sorry about that. In this github repository we will also include Bro. Nelson's report that has a really good explanation of the MTP basis, including a sample problem that can help you understand it if you want to.

The moment tensor potential (MTP) basis is a set of orthogonal basis functions given by:

$$V\left(\mathfrak{n}\right) = \sum_{\alpha} \xi_{\alpha} B_{\alpha}\left(\mathfrak{n}\right) \tag{1}$$

The MTP basis can be used to represent crystalline configuration space. A crystalline structure can be evaluated on this basis to an arbitrary precision. A simple analogy to this kind of evaluation is the Fourier transform, where higher and higher frequencies can be added to make a better fit to any function.

The basis functions $B_{\alpha}\left(\mathfrak{n}\right)$ of the MTP basis depend on the set of moment tensor descriptors:

$$M_{\mu,\nu}\left(\mathfrak{n}_i\right) = \sum_{j} f_{\mu}\left(\left|\mathbf{r}_{ij}\right|, z_i, z_j\right) \underbrace{\mathbf{r}_{ij} \otimes ... \otimes \mathbf{r}_{ij}}_{\nu times} \tag{2}$$

These descriptors are dependent on the immediate neighborhood, $\mathfrak{n}_i$, of the ith atom, within some R_{cut}, as shown in Figure 2. Each atom in the neighborhood of the ith atom introduces four degrees of freedom to the energy contribution, V_i. The total energy, E, depends on each energy contribution, V_i. The four degrees of freedom are the three coordinates in Euclidean space of the separation between atoms i and j, r_{ij}, and a discrete variable, z_j, that represents the species of the neighboring atom.

The $f_{\mu}\left(\left|\mathbf{r}_{ij}\right|, z_i, z_j\right)$ term in Equation 2 is given by:

$$f_{\mu}\left(\rho, z_i, z_j\right) = \sum_{k} c^{(k)}_{\mu,z_i,z_j} Q^{(k)}\left(\rho\right) \tag{3}$$

where

$$Q^{(k)}\left(\rho\right) = T_k\left(\rho\right)\left(R_{cut} - \rho\right)^2 \tag{4}$$

In Equation 4, $T_k\left(\rho\right)$ are the Chebyshev polynomials on the interval $[R_{min}, R_{cut}]$.

The $\mathbf{r}_{ij} \otimes ... \otimes \mathbf{r}_{ij}$ terms in Equation 2 contain angular information about the neighborhood $\mathfrak{n}_i$ and are tensors of rank ν. The basis functions $B_{\alpha}\left(\mathfrak{n}\right)$ are made up all of possible contractions of any number of $M_{\mu,\nu}\left(\mathfrak{n}_i\right)$ that result in a scalar. The maximum depth of these calculations is chosen, with more levels providing more accuracy. This attribute makes the basis a systematically improvable functional form, similar to including more frequencies in a Fourier transformation.

The ξ_{α} and $c^{(k)}_{\mu,z_i,z_j}$ terms in Equations 1 and 3 are parameters that must be optimized. A quasi-Newton optimization technique is used to fit these to the data provided by the training set. The configuration space created by these basis functions is N dimensional, where N is the number of basis functions the crystal was evaluated on. The optimization can be visualized in 2 dimensions,

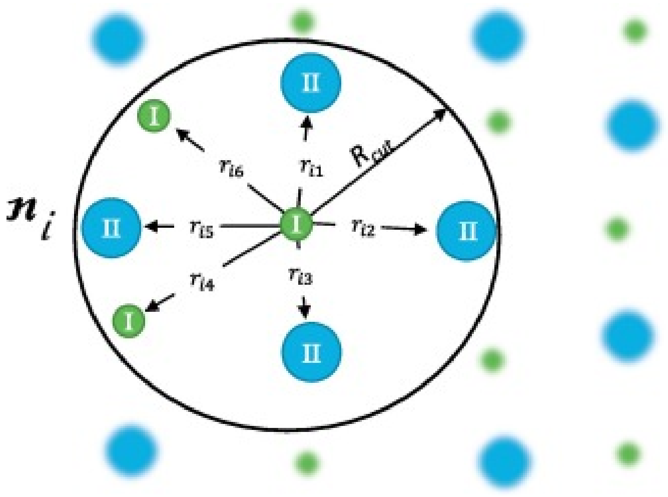

Figure 2: The ith atom's neighborhood is made up of each atom within some R_{cut} of itself. The total energy E is made up of the contributions from individual neighborhoods. The energy contribution, V_i, of neighborhood $\mathfrak{n}_i$ depends on the separation between atoms i and j, r_{ij}, and a discrete variable, z_j, that represents the species of the atom in the neighborhood (I or II in this illustration) [1].

"configuration" vs energy, shown in Figure 1, where the appropriate terms are chosen to minimize the error of a best fit "line" to the training data.

If you would like to know more, see ref [1]. There are several other less understandable (for undergraduates) papers that talk about the basis, in refs [2, 3, 4].

3.2 DFT

In quantum mechanics, the Schrödinger equation must be solved to find the energy of a system. Because of the size of many body problems, it is impossible to solve the Schrödinger equation exactly for the system. This leads scientists to an approximation called density functional theory (DFT). It is an ab-initio calculation, or a first-principles calculation, if you've ever heard those terms before. Density functional theory is based on the assertion that the ground state energy of a system is a unique functional (function of a function) of the electron density (which is a function) [5].

This is your quitting point. If you'd like to know more, go ahead and keep on reading!

In this energy functional mentioned earlier, called the Kohn-Sham equation [6], every term can be known exactly except the exchange-correlation (XC) functional.

The XC functional can be approximated in many different ways, including the Local Density Approximation (LDA), which assumes the electronic density

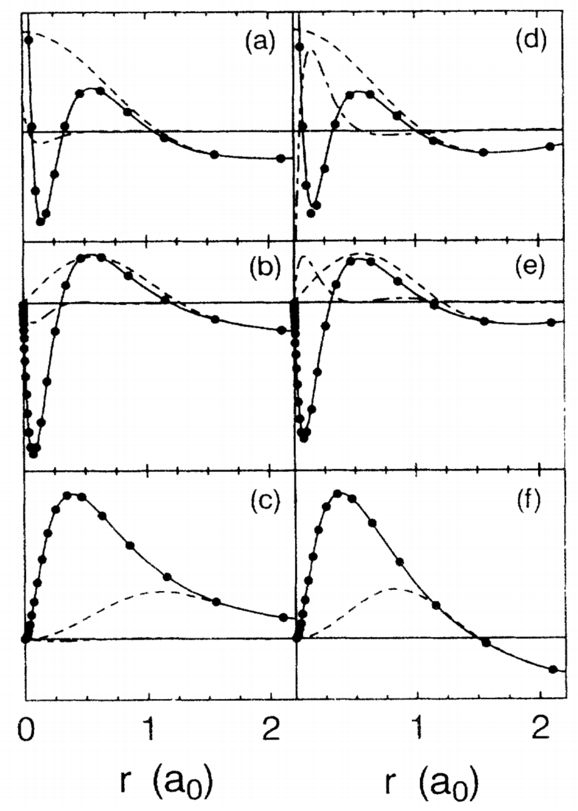

Figure 3: Comparison of atomic wave functions of Mn using the PAW method (solid line) with the exact result (bullets) for a given energy and angular momentum. Shown also are their differences magnified by a factor of 10 (dash-dotted line), and their pseudo wave functions (dashed line) [7].

behaves locally like a homogenous electron gas. Another approach is the Generalized Gradient Approximation (GGA), which is similar to LDA but includes the local gradient, and other derivative methods of these two basic methods.

Because the Kohn-Sham equation depends on the electronic density and is also used to find the density, an iterative approach that converges to be self consistent is used to solve for the energy of the system. DFT calculations are done in k-space, and sample points in the first Brillouin zone called k-points are chosen and appropriately weighted to replace the integrals in the Kohn-Sham equation. These are the KPOINTS files in your Vasp folders.

The method of pseudopotentials is often used to reduce the computational load of DFT calculations. This method approximates the inner electrons as a "frozen core" that place the electrons surrounding them in an effective potential. The two common methods of this approximation are the projector augmented-wave (PAW) method and the UltraSoft PseudoPotential (USPP) method. The PAW method allows all-electrons orbitals to be reconstructed from the pseudo-orbitals, as shown in Figure 3.

Vasp stands for the Vienna Ab initio Simulation Package [8]. This is the package we use to do DFT. Vasp requires the user to choose a method for approximating the XC functional. Vasp also prefers the use of the PAW method, but a specific PAW potential must be chosen. These are the POTCARS you specified in the yaml.

This is pretty much all I understand about DFT. If you'd like to know more, there's a pretty good lecture series on Youtube: https://youtu.be/vJkNv095Aj8, and a good book you could read [9].

3.2.1 DFT+U

With modeling large atoms (such as uranium), there can be some difficulties with getting the DFT calculations to converge to a "correct" total energy. You probably won't use the method I'm about to talk about, so you probably don't need to read this section unless you've talked to Brother Nelson about modeling large atoms, or you're morbidly curious. Read on, if you want.

This effect happens because the traditional treatment of electrons in DFT calculations allows the Coulomb repulsion to scatter the electrons, when in large atoms, the d and f electrons are strongly correlated and localized. Because the energy of the system is dependent upon the electron density, an incorrect density will often predict an energy that is too high.

To remedy the traditional treatment of electrons in large atoms, DFT+U should be used. DFT+U is also known as LDAU. Without the addition of the U parameter to DFT calculations, the calculation may converge, but it will likely converge to a non-physical solution.

The recommended method to ensure DFT+U converges to the "correct" total energy is to follow a ramping scheme discussed in ref [10] that begins with a U parameter of 0 and ramps up to 4.5 (in steps of 1, e.g. $0 \Rightarrow 1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 4.5$), using the charge density calculated in the previous iteration. This helps ensure convergence to a "true" total energy.

These are the settings in the INCAR that must be used to employ DFT+U:

- ENCUT = 550
- LWAVE = False
- LDAU = True
- LDAUTYPE = 1
- LDAUL = 3 -1 -1
- LDAUU = # 0 0 (# changes with ramping scheme)
- ICHARG = 1 (after the first iteration of ramping scheme)
- LDAUJ = 0.51 0 0
- ISMEAR = -5
- LMAXMIX = 6
- ISIF = 2
- NSW = 0

3.3 Optimization Routines

Optimization has to do with finding the best choice of some possible options. That's a really broad concept, but with relation to what we are doing, we want to minimize the error of a best fit line. We create a best fit "line" of energy vs. structure (remember figure 1?) This is a really simplified explanation of what the algorithm is actually doing, because a structure is decomposed into a set of N basis functions, and each basis function has a rank n tensor (where n is the order of the system, e.g. binary, ternary) that also has to be optimized with it, so it's not a linear system.

The optimization scheme used to train the model is called BFGS. It stands for Broyden-Fletcher-Goldfarb-Shanno. And now we're at your quitting point. Feel free to keep reading if you're very curious (good for you):

The BFGS algorithm is called a Quasi-Newton method, and is based on the Newton method of optimization, which uses the Taylor expansion of a function out to two derivatives, takes the partial with respect to Δx and iteratively searches for the value of x that makes the resulting function equal 0. This is the same as finding where the derivative of a function equals 0. For the Newton method, we need to find the second derivative to make this equation. For a multivariable function, the second derivative is the Hessian matrix. This matrix can be expensive to calculate, and thus Quasi-Newton methods were developed, which approximate the Hessian in one way or another. The BFGS algorithm requires an initial guess for the Hessian (usually an identity matrix of appropriate dimension) and iteratively approximates a new one as it solves for the x that minimizes the function. The equation for the Hessian matrix that the BFGS algorithm uses is as follows:

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k^t}{s_k^T B_k s_k} \tag{5}$$

with $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ and $s_k = x_{k+1} - x_k$.

The algorithm essentially finds a step direction (direction of steepest descent), finds an acceptable step size (backtracking line-search algorithm), takes the step, and then solves for the approximate Hessian matrix until the new x value and the old one are close enough to each other, according to some epsilon value.

The backtracking line-search algorithm start with a maximum step size, and makes it smaller by some factor $\tau \in (0, 1)$ until it satisfies what is called the Armijo-Goldstein condition, as follows:

$$f(x + \alpha p) \leq \alpha c m \tag{6}$$

where $m = p^T \nabla f(x)$ and $c \in (0, 1)$ is some control parameter.

Are you happy you read to the end of this section?

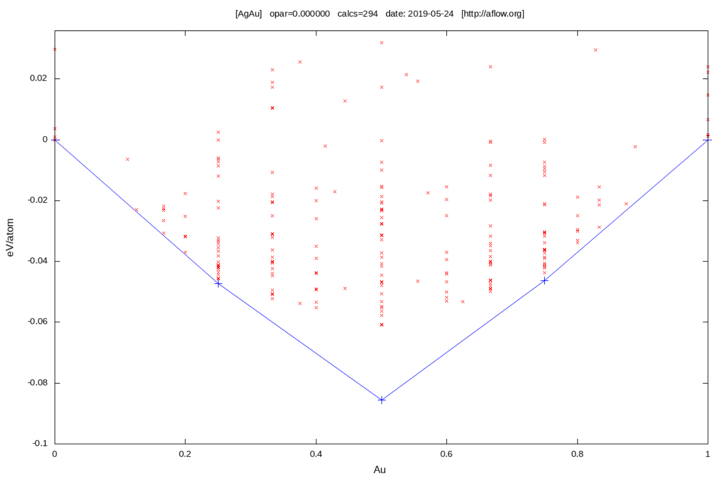

Figure 4: Convex hull of Ag-Au system as determined by 294 high-throughput ab initio calculations [11].

3.4 Convex Hull

This is actually an important section. Go ahead and read the whole thing if you get this far.

A convex hull is a cool mathematical tool, but it also has physical importance. See figure 4 for a visualization. The convex hull has some hints to what it is in it's name (for once a useful name for something). It is constructed of the lowest energy structures of a system that can be connected with a line that is convex. It looks like the hull of a boat. The breaking points of the convex hull represent the ground state structures of a system. (If you didn't know, you can't actually make a crystalline structure with any concentration of materials. For example, there is such thing as UO_2 and U_3O_8, but there is no such thing as U_3O_2).

I lied to you. This is your quitting point, don't bother reading this unless you're doing a ternary system (which you very well may be doing...). A three body system's convex hull is the same thing mathematically, but is maybe a bit harder to visualize because to see all of it you need 3 dimensions. But you can represent it in 2 dimensions, like in figure 5.

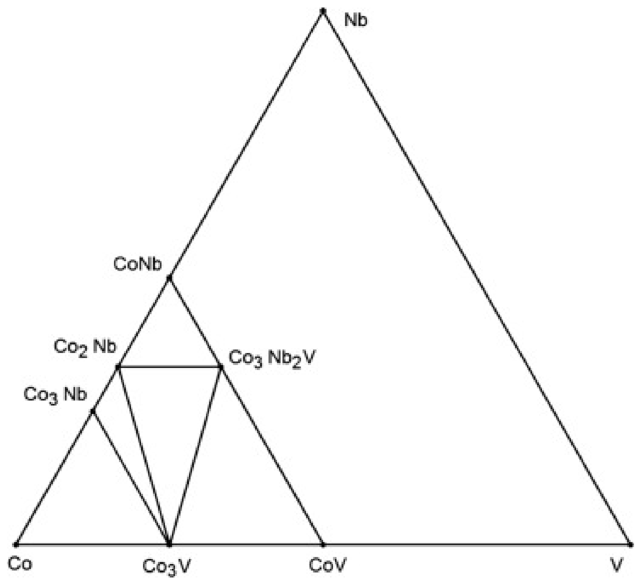

Figure 5: Convex hull of Convex hull of the Co-Nb-V system constructed by MTP in the Co-rich region [1].

Appendix A: VASP files and what they do

Pretty self explanatory name. If you want to learn a bit more about VASP, this is a good section to read.

POSCAR

This is the "POSition" car. Not really sure who came up with the naming convetion, but whatever `\_(-.-)_/`. In this file there is title. There is also the lattice parameter, which is what you multiply all the lattice vectors by to get the cartesian coordinates for the lattice vectos, which are the three lines that follow. Then there are the atom counts for each species (in reverse alphabetical order), and then the coordinate system. This can be "D" for direct, meaning that you multiply the first number of the basis vector by the first lattice vector, the second by the second, and the third by the third, then add up the x-coordinates, the y-coordinates, and the z-coordinates to find the cartesian coordinate of each atom within the unit cell. The "C" stands for cartesian coordinates, so each number is just an x, y, or z component of the basis atom's position. Next come the basis vectors, in either direct or cartesian coordinates.

If you didn't already know, the lattice vectors are the repeating unit of a crystalline structure. If you slide the origin to the end of one lattice vector, you will end up on the same position (as in, the same position but in a neighboring unit cell). The basis vectors are the positions of all the atoms in the unit cell.

Usually there is one basis vector with a value of (0,0,0), meaning that the lattice vectors lie in the middle of one of the atoms.

INCAR

This is is “INput” car. This file has all the settings for the vasp calculations. In Appendix E: Troubleshooting VASP I talk about some of the settings we’ve used in past works. They’re probably safe to use, but you might want to consult Brother Nelson about this.

PRECALC

This is the input file for k point generation. The only thing you need to touch in here is mindistance, unless you find yourself in some very dire circumstances. See Appendix E: Troubleshooting VASP for an explanation of such dire circumstances...

POTCAR

This is the “POTential” car. It has the psuedopotentials in it. You can `grep TITEL POTCAR` to make sure that the correct atom types are in here.

KPOINTS

This is generated by the getKPoints script. You probably don’t ever really have to worry about it unless you forget to generate it.

OUTCAR

This is the “OUTput” car. It has some of the output in it. You can `grep TOTEN OUTCAR` to see the total energy. If the energy has converged, you will see “`free  energy`” with two spaces.

CONTCAR

CONTCAR

If you let the atoms relax, this is the new POSCAR. Kinda. It just specifies the new positions of the atoms.

CHGCAR

OSZICAR

Appendix B: Bash Commands

To do any of this work, you need to learn to navigate your command line. Macs and Linux have a built in command line (terminal). On Windows, there are several options:

1. Windows Powershell. See: https://docs.microsoft.com/en-us/windows- server/administration/windows-commands/powershell for instructions on how to configure your pc to use it.
2. The Ubuntu app–simply search for it in your windows store.
3. Windows recently came out with Windows Terminal as a central place for all the command line userfaces.
4. If you know any other apps, feel free to use them

Your command line allows you to communicate with your machine by typing. The language you use is called bash, and if you want to make a script to execute bash commands, you call it a shell script. See Table 4 for a list of helpful commands.

Table 4: Bash commands and what they mean

Command	**What it does**
`ls`	list contents of current directory
`ls -a`	show hidden files too
`ls -altr`	see the last changes made to the files in a directory
`mkdir directory`	make a new directory
`cd directory`	change directory
`cd ..`	go back a directory
`cd ../..`	go back two directories
`cd ~ or cd`	go to your home directory
`pwd`	print working directory
`~/`	means your root
`.`	means the current directory
`cp file/to/copy where/newName`	copy a file
`cp file/to/copy .`	copy a file to current directory without changing the name
`cp directory/* .`	copy all the files in a directory
	to the current directory
`cp -r directory new/directory`	copy a directory recursively

Table 4 – *Continued on next page*

Table 4 – *Continued from previous page*

Command	**What it does**
`rm file/to/remove`	remove a file
`rmdir directory`	remove a directory
`rm -rf directory`	blow away a directory permanently
`mv file/to/move where/newName`	moves or renames a file
`man command`	show the manual for a command
`cat file/one file/two > new_file`	concatonate two or more files into a new file
`history`	shows a history of your commands
`less file/to/see`	shows one page of a file
	space turns the page q quits
`head file/to/see`	see the first page of a file
`head -n 8 file/to/see`	see the first 8 lines of a file
`tail file/to/see`	see the last page of a file
`tail -n 10 file/to/see`	see the last 10 lines of a file
`grep keyword file/to/search`	search a file for a keyword and print all the lines with that keyword to the screen
`history | grep keyword`	search your history for a keyword
`grep keyword file/to/search | wc -l`	count the occurences of lines with a keyword
`command | less`	pipe the output of a command to less
`command >> file`	append the output of a command to a file
`command > file`	writes the output of the command to a file
`!command`	executes the most recent command that starts with the letters you typed
`echo something`	print something to the screen

Bash Loops

Here's a table of basic bash loops and logic, and a basic example in Table 6 that relates to what we do.

Table 5: Loops in bash

Command	**What it does**
`for i in {1..100}`	for 100 iterations
`do`	
`command $i`	do this thing (`$i`references the index)
`done`	
`for i in ‘ls -d */‘`	for every directory in this directory
`do`	
`cd $i`	cd into every directory
`...`	
`if [ condition ]`	check the condition
`then`	if it’s true
`command`	do this
`else`	if it’s not
`command`	do this
`fi`	
`if [ -e file ]`	check if a file exists

Table 6: Example of a Bash loop

Command	**What it does**
`for i in {1..100}`	for 100 times
`do`	
`cd E.$i`	enter the directory named `E.#`
`if [! -e KPOINTS ]`	if KPOINTS doesn’t exist
`echo $i`	print the directory number
`getKPoints`	run the getKPoints script
`fi cd ..`	go back one directory
`done`	

Appendix C: Emacs

Emacs is a text editor that has a bunch of cool shortcuts you can learn to make editing documents super easy. On a Mac you can download Aquamacs, which uses the same commands as Emacs but you can click with your mouse. 7 has a chart of basic Emacs commands.

Table 7: Emacs commands and what they mean

Command	**What it does**
`emacs path/to/file`	enter emacs editor for existing file or creates new file with that name
ctrl+x ctrl+c y	save and quit a file
ctrl+x ctrl+c n	quit without saving
ctrl+w	cut a line
ctrl+y	paste a line
ctrl+k	kills the contents of a line
ctrl+k ctrl+k	kills a whole line
ctrl+shift+-	undo
ctrl+u `3 command`	executes the command 3 times
ctrl+x ctrl+f	find and open a file (at the bottom of the screen)

Appendix D: Basics of Github

This section is by no means comprehensive. In fact, there's probably a lot of things missing. If you'd like to actually get good at Github, you have a lot more work to do. But here's some basics.

The basic idea of Github is that several people can work on the same code at the same time. It also has version control: if you don't like the recent changes to your code, you can get rid of them by going back to an old version of the code. You can have a copy of the code on your machine to work on, and you can push your changes to the main copy of the code. Most people, however, don't have their repositories open to the public to edit. They will have to clear any suggested changes that you push. But as they make edits, you can pull their copy down to your machine. Github will warn you if there are any changes you have made to the code that will be overwritten by copying their changes. It's a powerful code sharing tool, but takes a little getting used to.

First you need to go make a Github account. Then you can fork (make a copy of) whatever repository of code that you want to copy and edit. If you don't want to make edits to the code yourself, you don't have to make a fork, you can just copy directly from the original repository each time. To give your computer (or a remote computer) access to your repositories on Github, add your computer's public SSH key to Github. How to do this:

On your command line execute the following command to copy your public ssh key:

```
pbcopy < ~/.ssh/id_rsa.pub
```

Note: If this didn't work, it means you don't have a public key and will need to generate one. See the subsection below this for instructions on how to do that.

Now you will be able to paste this key into your account on Github. To do this, go to Settings⇒SSH and GPG keys⇒New SSH key. Make sure you name it something intuitive (e.g. My Mac, Marylou, etc.).

Now you can copy code from Github to your machine. It's probably a good idea to make a directory to hold all the stuff you're going to copy with a name you'll recognize later (e.g. `aBuild/` for aBuild, etc.). Usually these directories are in `~/codes/`. Now there are several scenarios involving the code you're copying from Github. Here's a couple and what to do about them:

1. If you just want to make a copy of the code and don't expect any changes to be made to it (like ever), use the following command:

```
git clone git@github.com:user/repository.git
```

You can do this over and over again, it will just overwrite the code you copied to your machine.

2. a. If you expect to make your own changes to the code, start by making a fork of the original code on Github. Then add a master copy of your fork to your machine using the following command:

```
git remote add master git@github.com:user/repository.git
```

2. b. If you have your own code that you would like to edit and put on Github for others to see, you will have to make your own repository on Github, then use the same command as above.

3. If you want to be able to add someone else's updates to the code to your machine, the convention is to call their repository `upstream`. You can add read only access to their repository to your machine with the following command:

```
git remote add upstream git@github.com:user/repository.git
```

Now that you have a copy of the code, here's how to make changes to the code on your machine, and on Github from your machine. Here's a list of some basic Git commands. Remember that this is not a comprehensive list.

Table 8: Basic Github commands

Command	**What it does**
`git checkout file/to/update`	make a copy of upstream's version of the file on your machine
`git pull upstream master`	update your master copy with upstream's version
`git status`	tells you the status of each file on your machine with Github's copy
`git add file/to/add`	add your version of the file to the commit

Table 8 – *Continued on next page*

Table 8 – *Continued from previous page*

Command	**What it does**
`git commit`	commit the changes you added. Will open an editor for you to leave a comment
`git commit -m "comment"`	commit the changes you added and leave a comment
`git push`	pushes your latest commit to your fork on Git
`git push master upstream`	push your request to add your changes to upstream
`git log`	See history of most recent commits

Note: Atlassian has a good Git tutorial and reference guide.

SSH key generation

See https://help.github.com/en/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent#generating-a-new-ssh-key

Appendix E: Troubleshooting VASP

If you ever want to know more about the settings you can use in Vasp, you can go to: https://cms.mpi.univie.ac.at/wiki/index.php/The_VASP_Manual.

Default settings used for all calculations:
- PREC = a
- LWAVE = False
- LREAL = auto
- ISMEAR = 1
- SIGMA = 0.1

DFT+U settings used:
- ENCUT = 550
- LWAVE = False
- LDAU = True
- LDAUTYPE = 1
- LDAUL = 3 -1 -1
- LDAUU = # 0 0 (# changes with ramping scheme explained in Section **??**)
- ICHARG = 1 (after the first iteration of ramping scheme explained in Section **??**)
- LDAUJ = 0.51 0 0

- ISMEAR = -5
- LMAXMIX = 6
- ISIF = 2
- NSW = 0

Settings used to assign magnetic moments:
- ISPIN = 2
- MAGMOM = +/- 2 for U, 0 for O

In the case of non-convergence, ionic steps were allowed at a setting of:
- NSW = 3

and this additional setting was added:
- IBRION = 2

In the case of the following error:
- `VERY BAD NEWS! internal error in sub-routine SGRCON`

The following setting was changed:
- INCLUDEGAMMA = False

and KPOINTS was regenerated. If it still did not work, the following setting was changed from the default:
- SYMPREC = 10^{-4}

In the case of the following errors:
- `POSMAP internal error: symmetry equi-valent atom not found, you might try decreasing or  increasing SYMPREC by an order of magnitude.`
- `VERY BAD NEWS! internal error in sub- routine PRICEL (probably precision problem, try to change SYMPREC in INCAR ?): Sorry, number of cells and number of vectors did not agree.`
- `RHOSYG internal error: stars are not distinct, try to increase SYMPREC to e.g. 1E-4`

The following setting was changed from the default:
- SYMPREC = 10^{-4}

To run calculations in parallel, the following settings were added:
- LPLANE = True
- NCORE = (number of cores)
- LSCALU = False
- NSIM = 4

References

[1] K. Gubaev, E. V. Podryabinkin, G. L. Hart, and A. V. Shapeev, "Accelerating high-throughput searches for new alloys with active learning of interatomic potentials," *Computational Materials Science*, vol. 156, pp. 148–156, 2019.

[2] E. V. Podryabinkin and A. V. Shapeev, "Active learning of linearly parametrized interatomic potentials," *Computational Materials Science*, vol. 140, pp. 171–180, 2017.

[3] K. Gubaev, E. V. Podryabinkin, and A. V. Shapeev, "Machine learning of molecular properties: Locality and active learning," *The Journal of chemical physics*, vol. 148, no. 24, p. 241727, 2018.

[4] A. V. Shapeev, "Moment tensor potentials: A class of systematically improvable interatomic potentials," *Multiscale Modeling & Simulation*, vol. 14, no. 3, pp. 1153–1173, 2016.

[5] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," *Phys. Rev.*, vol. 136, pp. B864–B871, Nov 1964.

[6] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects," *Physical review*, vol. 140, no. 4A, p. A1133, 1965.

[7] P. E. Blöchl, "Projector augmented-wave method," *Phys. Rev. B*, vol. 50, pp. 17953–17979, Dec 1994.

[8] G. Kresse and J. Furthmüller, "Software VASP, vienna (1999)," *Phys. Rev. B*, vol. 54, no. 11, p. 169, 1996.

[9] J. Kitchin, "Modeling materials using density functional theory," *Boston, Free Software Foundation*, 2008.

[10] B. Meredig, A. Thompson, H. Hansen, C. Wolverton, and A. Van de Walle, "Method for locating low-energy solutions within DFT+ U," *Physical Review B*, vol. 82, no. 19, p. 195128, 2010.

[11] S. Curtarolo, W. Setyawan, S. Wang, J. Xue, K. Yang, R. H. Taylor, L. J. Nelson, G. L. Hart, S. Sanvito, M. Buongiorno-Nardelli, *et al.*, "AFLOWLIB. ORG: A distributed materials properties repository from high-throughput ab initio calculations," *Computational Materials Science*, vol. 58, pp. 227–235, 2012.