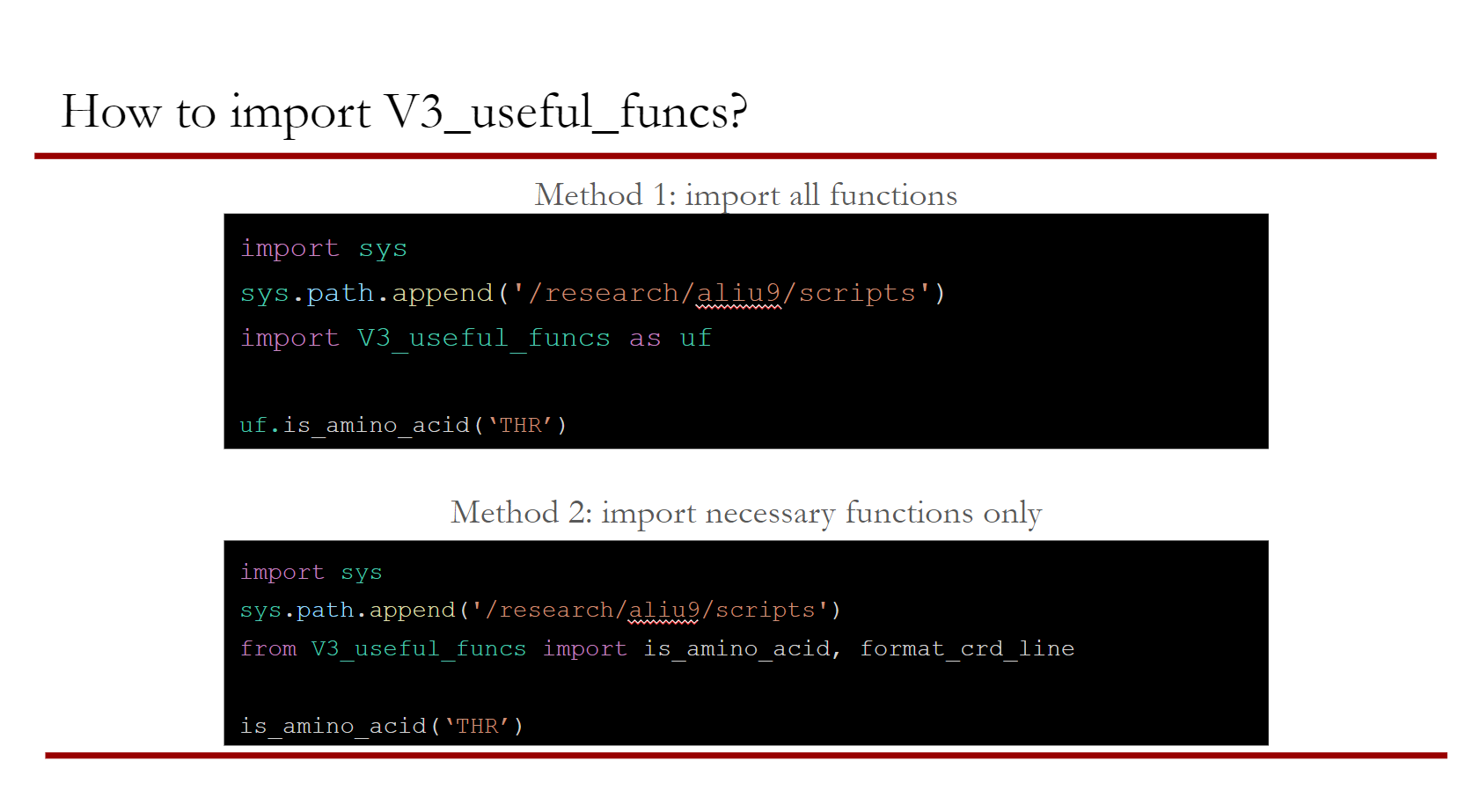
Check out the ppt that talks about the motivation for creating this Python module and some example code using it: [**https://docs.google.com/presentation/d/1MtQAstXywmzHmVjZ4GQN3G4OgczyIfvGgDwx7p6SoRE/edit?usp=sharing**](https://docs.google.com/presentation/d/1MtQAstXywmzHmVjZ4GQN3G4OgczyIfvGgDwx7p6SoRE/edit?usp=sharing)

**To import V3\_useful\_funcs:**



**crd\_line\_contains\_atom(line)**

If crd line is a header line, return false. If it contains information about atom, return true. A line is considered to be a header if it starts with \* or if it only contains the number of atoms in the crd file

**get\_pdb\_box\_size(line)**

Take in pdb line containing the size of simulation box as a string, and return the box dimension as tuple of floats (x,y,z)

**change\_atom\_name(atom\_name)**

Take in atom\_name as str and moves the first character of of the atom\_name to the last position until the first character is no longer a number eg. given 1HG return HG1, 12HG return HG12. This function is needed because there's a bug in our LPBE solver that can't have atom\_names starting with a number, so during snapshot preparation, atom\_names are changed

**is\_amino\_acid(res\_type)**

Given 3 letter string (letter case insensitive), return true if residue is amino acid and false if res\_type is not amino acid (sol, lipid). Raise exception if res\_type is anything other than str.

**only\_alphabet(string)**

Given a string, return only the characters that are alphabets. This function was created because the res column in a gro file has both res num and res type eg. 12ARG. Given 12ARG, the function will return ARG.

**only\_digit(string)**

Given a string, return only the characters that are numbers, eg only\_digit(file13.crd) will return 13. Note, only\_digit(1file3.crd) will return 13 too

**read\_itp\_atom\_lines(line)**

Given line from [ atom ] section of itp file return None if line starts with ;

If line doesn't start with ; return dictionary with keys atom\_type, res\_id, res\_type, atom\_name, charge

**itp\_atom\_lines(f):**

Take in f as the path to itp file (this function handles opening the file), store the [ atoms ] section of itp file as a list of strings by looking for lines that occur between [ atoms ] and [ bonds ] headers to create a list called itp\_lines

Using itp\_lines list, this function returns a dictionary with atom\_name : charge as key val pairs. This function was written for adding charges into crd file

**itp\_total\_charge(atom\_charges)**

given a dictionary with atom\_name and charges as key value pairs, (output of itp\_atom\_lines) return net charge of molecule by summing up all charges in atom\_charges dictionary

**return\_itp\_molecule\_type(itp)**

given path of itp file, open and read it and return molecule type eg (POPE, POPG, Protein)

**create\_gams\_table(header, set\_i, set\_j, matrix, col\_spacing=15)**

take in header as string: 'd(i,j) distance in thousands of miles'

take in set\_i as array of strings: ['seattle', 'san-diego']

take in set\_j as array of strings: ['new-york', 'chicago', 'topeka']

take in matrix as 2d array of strings : [['2.5', '1.7', '1.8'], ['2.5', '1.8', '1.4']]

col\_spacing is an optional argument. The default value is 15. Adjust to bigger number for wider spacing and smaller number for tighter spacing

return str:

Table

d(i,j) distance in thousands of miles

new-york chicago topeka

seattle 2.5 1.7 1.8

san-diego 2.5 1.8 1.4

assumes set\_i is the rows, set\_j is col

Note if numpy array is fed in instead of python array, the column spacing gets messed up

**create\_table(matrix, col\_spacing=15)**

take in matrix output a table as string, col\_spacing optional argument, default 15, adjust bigger number for wider spacing, smaller number for tighter spacing and return a table.

This function is different from create\_gams\_table in that it doesn't take in headers

**create\_gams\_list(header, set\_i, vector)**

take in header as string: 'a(i) capacity of plant i in cases'

take in set\_ i as an array of strings: ['seattle', 'san-diego']

take in vector as an array of strings: ['350', '600']

return str:

Parameters

a(i) capacity of plant i in cases

/ seattle 350

san-diego 600 /;

**create\_gams\_scalar(name\_description, val)**

take in name\_description as string: 'f freight in dollars per case per thousand miles

take in val as string or float or int: 90

return str: Scalar f freight in dollars per case per thousand miles /90/;

**create\_gams\_sets(sets)**

take in sets as dictionary, the key is name and description of the set

the value is the members of the set

{'i canning plants' : ['seattle', 'san-diego'],

'j markets' : ['new-york', 'chicago', 'topeka'],

}

return str:

Sets

i canning plants /seattle, san-diego/

j markets /new-york, chicago, topeka/;

**extract\_from\_gams(gams\_file, var, cons=True, float\_out=False)**

given path of gams\_file and name of variable to look for in gams file (output of gams with extension lst)

if cons is True (default value) return a float eg. deltaG

else return an array eg. optimal qs

float\_out set to False by default, will return as strs or list of strs

This function makes the assumption that the output lines is unbroken:

---- VAR qs charges on side chain

LOWER LEVEL UPPER MARGINAL

q1 -1.0000 -1.0000 1.0000 2.0401

q2 -1.0000 0.3008 1.0000 .

q3 -1.0000 1.0000 1.0000 -0.0255

q4 -1.0000 0.3353 1.0000 EPS

q5 -1.0000 -0.6835 1.0000 -1.254636E-9

q6 -1.0000 1.0000 1.0000 -3.0361

q7 -1.0000 -0.6557 1.0000 EPS

q8 -1.0000 0.7477 1.0000 EPS

It wouldn’t work if page break occurs:

---- VAR qs charges on side chain

LOWER LEVEL UPPER MARGINAL

q1 -1.0000 -1.0000 1.0000 2.0401

q2 -1.0000 0.3008 1.0000 .

q3 -1.0000 1.0000 1.0000 -0.0255

q4 -1.0000 0.3353 1.0000 EPS

q5 -1.0000 -0.6835 1.0000 -1.254636E-9

----------------page break

q6 -1.0000 1.0000 1.0000 -3.0361

q7 -1.0000 -0.6557 1.0000 EPS

q8 -1.0000 0.7477 1.0000 EPS

**generate\_run\_file\_output(atoms\_charged, name=0)**

given the atoms being charged, and the name of the grp return a string of 3 lines:

generate\_run\_file\_output('0', '0')

will return:

mark=output

atoms\_charged=(atomno eq "0")

name="0"

name is an optional argument, if no argument given, name is assumed to be the same as atoms\_charged

**generate\_run\_file\_bound\_unbound\_state(mark, state, atoms\_charged, atoms\_shape, atoms\_center)**

given 4 strs return a group of 4 lines in the run file, user need to provide parentheses and quotation marks. This function could be improved if the user doesn't have to put in the () or the "" marks in the argument

**create\_run\_file\_partial\_charge\_opt(s\_atoms, current\_atom)**

s\_atoms is a list of side chain atoms return entirety of run file

**read\_diff\_table\_w\_4\_vals(diff\_table\_path)**

Given path of diff table (for deltaG calculation with 4 values not for charge optimization), open and read the file. Return a dictionary with keys RDP, LDP, and INT, and corresponding values as floats

**extract\_matrix\_vec\_cons(res\_dir, vec1='Cqr', vec2='Lstqt', cons1='Ltt', cons2='Cqrt', cons\_dir='constants')**

given the path to res\_dir as string, assemble Lmatirx, Cqr vector, and Lstqt vector and return as dictionary {Lmatrix : [], Cqr : [], Lstqt : []...}

Numbers in arrays are of type string

Takes in optional parameters for name of output grp, default is Cqr and Lstqt vectors

Assumes atom dirs are numbered, and looks for atoms in difference.table file as numbers