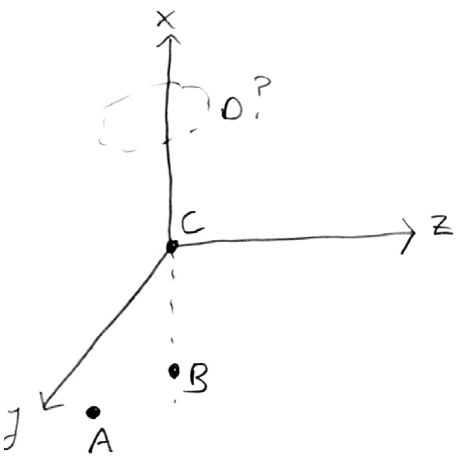


point A is N_i

point B is $C_{\alpha i}$

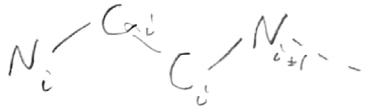
point C is C_i

point D is N_{i+1}

Morgan Nance

Hw #2

Q #1



point C: $(0, 0, 0)$ by definition

C_i

point B: $(-1.5, 0, 0)$ down x-axis by C-C bond length

$C_{\alpha i}$

point A: $(-l\cos\theta, l\sin\theta, 0)$ $\theta = 115°$

N_i $(-1.3\cdot\cos(115°), 1.3\cdot\sin(115°), 0)$

$(0.466, 1.21, 0)$

point D: $(-l\cos\theta, l\sin\theta\cos\phi, l\sin\theta\sin\phi)$ $l = 1.3$, $\theta = 115°$, $\phi = -126.1°$

N_{i+1} $(-1.3\cos(115°), 1.3\cdot\sin(115°)\cdot\cos(-126.1°), 1.3\cdot\sin(115°)\cdot\sin(-126.1°))$

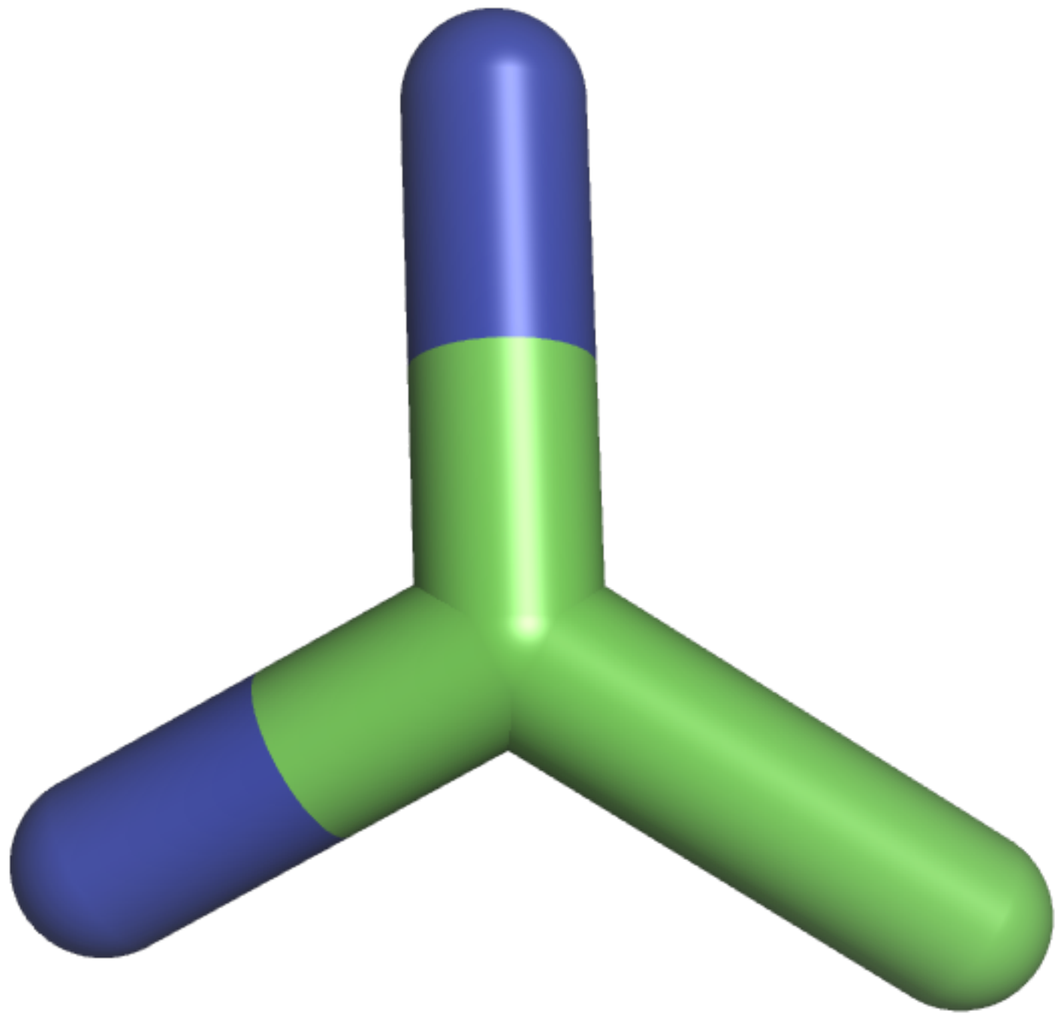
$N_{i+1} = D = (0.634, -0.800, -1.098) \times M$

but couldn't figure out M rotation matrix

Q1.b. I would translate the xyz plane to the next carbonyl carbon and recalculate the xyz coordinates for the next N atom. This would repeat using the previous residue's (ℓ, θ, χ) coordinates to answer the next.

Q 1.d. Without the M rotation matrix, the N_{i+1} is in the wrong spot. I also had a problem with the C and C_α atoms being in the wrong spot.

```
ATOM      1  N   ALA A   1       0.466   1.210   0.000  1.00 39.26
N
ATOM      2  CA  ALA A   1      -1.500   0.000   0.000  1.00 37.65
C
ATOM      3  C   ALA A   1       0.000   0.000   0.000  1.00 40.25
C
ATOM      4  N   ALA A   2       0.634  -0.800  -1.098  1.00 41.55
N
END
```



```
#!/usr/bin/python
__author__ = "morganlnance"

"""
HW2 Question 3 (Workshop #2 Exercise 2)

This program makes an alpha helix of a specified length
out of alanines. Phi and psi values are idealized.
This program dumps the resulting pose as 'helix.pdb'

Usage:   ./make_helix.py <number of helix residues>
Example: ./make_helix.py 20
"""


###########
# IMPORTS #
###########
import sys
from pyrosetta import init, \
    pose_from_sequence, PyMOLMover


#############
# ARGUMENTS #
#############
try:
    n_helix_residues = int(sys.argv[1])
except:
    print "\nGive me an integer for the number of helix residues.\n"
    sys.exit()


########
# MAIN #
########
# initialize pyrosetta and load the pose
init()
pose = pose_from_sequence("A"*n_helix_residues, "fa_standard")

# PyMOLMover for visualization
pmm = PyMOLMover()
pmm.keep_history(True)

# ideal phi and psi values
phi = -60
psi = -45

# set phi and psi
pmm.apply(pose)
```

```
for ii in range(1, pose.size()+1):
    pose.set_phi(ii, phi)
    pose.set_psi(ii, psi)
    pmm.apply(pose)
pose.dump_pdb("helix.pdb")
```

Homework 2 Question 3 Short Answer

You can ensure your structure is a proper alpha helix by checking:

1. If there is an i → i+4 backbone hydrogen bonding pattern
2. If there are 3.6 residues per turn of the helix
3. If the helix spirals in a right-handed fashion

```
#!/usr/bin/python
__author__ = "morganlnance"

"""
HW2 Question 4 (Workshop #2 Exercise 4)

This program calculates the propensities of the 20
amino acids in either a helix, sheet, or loop. Phi
and psi values defining these secondary structures
are set values taken from an NMR website. The program
prints the propensities to the screen.

Usage:   ./ss_propensities.py <.pdb file>
Example: ./ss_propensities.py 1m40.pdb
"""


############
# IMPORTS #
############
import sys
from pyrosetta import init, \
    pose_from_file


#############
# ARGUMENTS #
#############
try:
    user_input = sys.argv[1]
except IndexError:
    print "\nPlease give me a .pdb file.\n"
    sys.exit()


###############
# AMINO ACIDS #
###############
# one-letter name to three-letter name
aa_dict = { "A": "ALA", "R": "ARG", "N": "ASN", "D": "ASP", "C":
"CYS",
            "Q": "GLN", "E": "GLU", "G": "GLY", "H": "HIS", "I":
"ILE",
            "L": "LEU", "K": "LYS", "M": "MET", "F": "PHE", "P":
"PRO",
            "S": "SER", "T": "THR", "W": "TRP", "Y": "TYR", "V":
"VAL" }

# ugly dictionary instantiation
# residue name to count of helix/sheet/loop in pose
```

```
aa_helix_propensities = { "ALA": 0, "ARG": 0, "ASN": 0, "ASP": 0,
"CYS": 0,
                          "GLN": 0, "GLU": 0, "GLY": 0, "HIS": 0,
"ILE": 0,
                          "LEU": 0, "LYS": 0, "MET": 0, "PHE": 0,
"PRO": 0,
                          "SER": 0, "THR": 0, "TRP": 0, "TYR": 0,
"VAL": 0 }
aa_sheet_propensities = { "ALA": 0, "ARG": 0, "ASN": 0, "ASP": 0,
"CYS": 0,
                          "GLN": 0, "GLU": 0, "GLY": 0, "HIS": 0,
"ILE": 0,
                          "LEU": 0, "LYS": 0, "MET": 0, "PHE": 0,
"PRO": 0,
                          "SER": 0, "THR": 0, "TRP": 0, "TYR": 0,
"VAL": 0 }
aa_loop_propensities = { "ALA": 0, "ARG": 0, "ASN": 0, "ASP": 0,
"CYS": 0,
                          "GLN": 0, "GLU": 0, "GLY": 0, "HIS": 0,
"ILE": 0,
                          "LEU": 0, "LYS": 0, "MET": 0, "PHE": 0,
"PRO": 0,
                          "SER": 0, "THR": 0, "TRP": 0, "TYR": 0,
"VAL": 0 }


###############
# PHI AND PSI #
###############
# http://nmr.chem.uu.nl/~adrien/course/molmod/analysis2.html
# phi and psi for helix
helix_phi_min = -140
helix_phi_max = -30
helix_psi_min = -80
helix_psi_max = 50

# phi and psi for sheet
sheet_phi_min = -180
sheet_phi_max = -40
sheet_psi_min = 50
sheet_psi_max = 180


########
# MAIN #
########
# initialize pyrosetta and the pose
init("-mute all -ignore_unrecognized_res")
pose = pose_from_file(user_input)
```

```
############################
# GET SECONDARY STRUCTURE #
############################
sec_struct = ""
for ii in range(1, pose.size()+1):
    # if it's an amino acid
    if pose.residue(ii).is_protein():
        # HELIX
        # phi and psi range for helices
        if pose.phi(ii) < helix_phi_max \
                and pose.phi(ii) > helix_phi_min \
                and pose.psi(ii) < helix_psi_max \
                and pose.psi(ii) > helix_psi_min:
            sec_struct += "H"
            # increase counter in helix dictionary
            aa_helix_propensities[aa_dict[
                    pose.residue(ii).name1()]] += 1
        # phi and psi range for sheets
        elif pose.phi(ii) < sheet_phi_max \
                and pose.phi(ii) > sheet_phi_min \
                and pose.psi(ii) < sheet_psi_max \
                and pose.psi(ii) > sheet_psi_min:
            sec_struct += "E"
            # increase counter in sheet dictionary
            aa_sheet_propensities[aa_dict[
                    pose.residue(ii).name1()]] += 1
        # if not a helix or sheet, it's a loop
        else:
            sec_struct += "L"
            # increase counter in loop dictionary
            aa_loop_propensities[aa_dict[
                    pose.residue(ii).name1()]] += 1


################
# PROPENSITIES #
################
# number of residues that are in a helix, sheet, or loop
num_helix = float(sum(aa_helix_propensities.values()))
num_sheet = float(sum(aa_sheet_propensities.values()))
num_loop = float(sum(aa_loop_propensities.values()))
print "     HELIX\tSHEET\tLOOP"
for aa in aa_dict.itervalues():
    # helix
    helix_propensity = round(float(aa_helix_propensities[aa]) /
num_helix, 3)
    # sheet
    sheet_propensity = round(float(aa_sheet_propensities[aa]) /
num_sheet, 3)
```

```
    # loop
    loop_propensity = round(float(aa_loop_propensities[aa]) /
num_loop, 3)
    # print to screen
    print "%s: %s\t%s\t%s" %(aa, helix_propensity,
                             sheet_propensity, loop_propensity)
```

```
Found rosetta database at: /Library/Python/2.7/site-packages/
pyrosetta-4.0-py2.7-macosx-10.12-intel.egg/pyrosetta/database; using
it....
PyRosetta-4 2016 [Rosetta 2016
unknown:a19ad2ea80d9e9e4c5e99a617b8c2941a5b5e5ba 2017-09-19 09:49:00
-0700] retrieved from: git@github.com:RosettaCommons/main.git
(C) Copyright Rosetta Commons Member Institutions.
Created in JHU by Sergey Lyskov and PyRosetta Team.

(Ran using 1m40.pdb as the input)

     HELIX       SHEET   LOOP
ALA: 0.105       0.096   0.036
CYS: 0.013       0.0     0.0
GLU: 0.086       0.084   0.0
ASP: 0.053       0.072   0.071
GLY: 0.026       0.036   0.5
PHE: 0.013       0.036   0.0
ILE: 0.039       0.108   0.0
HIS: 0.026       0.012   0.036
LYS: 0.053       0.036   0.0
MET: 0.039       0.012   0.036
LEU: 0.158       0.072   0.036
ASN: 0.046       0.0     0.036
GLN: 0.046       0.012   0.0
PRO: 0.046       0.06    0.0
SER: 0.053       0.06    0.071
ARG: 0.046       0.108   0.071
THR: 0.079       0.096   0.036
TRP: 0.013       0.024   0.0
VAL: 0.059       0.048   0.0
TYR: 0.0 0.024   0.071
```