SUMMARY REPORT OF SEMESTER PROJECT

ShiLin WANG shi-lin.wang@epfl.ch SCIPER: 294925
Supervisor : Dr. Luciano Abriata
Laboratory of Biomolecular Modeling

Abstract

Template-free *de novo* protein design provides a powerful tool for the engineering of epitope-mimicking immunogens with complex structural motifs. Normally to discriminate a good/bad design requires intensive lab-experiments. In this report, I aim to find a discriminator by means of MD simulation. Several simulation properties as discriminator candidate are to be proposed and discussed.

1 Introduction

It is evidenced that only a few human pathogens have broadly neutralizing antibody (nAb) epitopes, which are often surrounded by strain-specific, non-neutralizing or disease-enhancing epitopes [1]. As such, developing of immunogens, which can mimic pathogens' neutralizing epitope and so precisely trigger elicitation of specific antibody response, is particularly focused. Thanks to the progress of vaccine technology, many neutralizing antibodies in complex with their antigen have been structurally characterized. It thus provides the foundation for structure-based immunogen design.

Toward the goal to develop a structure-based epitope-mimicking immunogen, B. E. Correia et al. have applied computational method to transplant the linear helix-turn-helix motif from RSVF (respiratory syncytial virus F protein) antegenic site II onto heterologous protein scaffold [2]. It shows to elicit nAb response in non-human-primate (NHP) with boosting immunization. However, despite the success of this research, it turns out computational method was not technically preferred for a structurally complex motif.

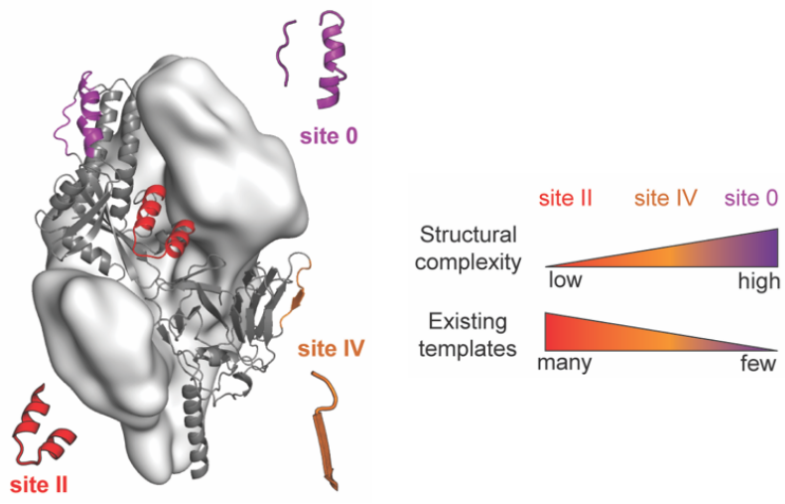


Figure 1: RSVF neuatralizaing epitope and its corresponding antigenic site (0, II, IV). Legend shows the structural complexity and number of templates in protein data bank for respective antigenic site.

As shown by Figure-1, one can roughly see that antigenic site 0 has a kinked alpha helix plus a disordered loop, and antigenic site IV has a distorted beta strand. Compared with antigenic site I, antigenic site 0 and IV have more complex structure. Normally, a motif with higher structural complexity corresponds to a lower number of existing templates.

Complex structural motif poses a barrier for structure-based immunogen design. For instance, F. Sesterhenn, C. Yang and J. Bonet et al. have applied template-based *de novo* method to design immunogens for RSVF antigenic site 0 and IV [3]. They search for structurally matching protein scaffold in protein data bank (PDB). Given a low or even zero number of matching templates, they increase the tolerance to have more samples to try on. To acquire antigenic site 0 and IV respectively, **Rosetta FunFolDes** is used to fold the design protein repeatedly. Nevertheless, it turns out a poor bonding affinities even though the designed immunogen are properly mutated after *in vitro* evolution. The overall process is summarized and shown as Figure-2.

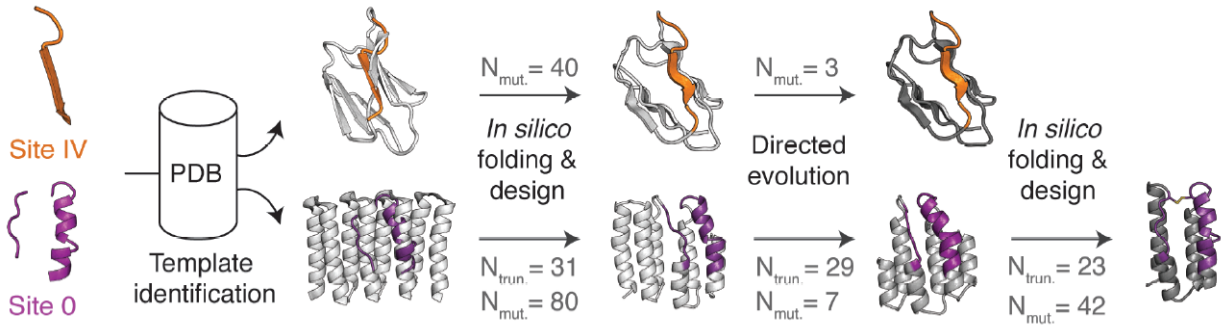


Figure 2: Schematic representation of template-based *de novo* method for RSVF antigenic site 0 and IV, where N_{mut} is the number of mutattion and N_{tru} is the number of truncation.

Based on previous discussion, it can be noticed that template-based *de novo* method suffers from two major caveats. On one hand, the overall process requires a series of optimization by *in vitro* evolution and *in silico* computation. On the other hand, the obtained immunogen design is still sub-optimal, which limits the binding affinity.

To address such problem, F. Sesterhenn, C. Yang and J. Bonet et al. come up with a novel template-free *de nove* method [3]. For an example, the immunogen design for RSVF antigenic site 0 is shown as Figure-3. As can be seen, this method consists of three steps. First, space is viewed in a layered, 2D dimension. In such 2D space, topological compatible protein motifs, secondary structure elements (SSEs) as well as possible SSEs connection are extensively enumerated. Second, obtained 2D topology is projected into 3D space. Then a structural refinement will be done such as tuning the distances and orientation between SSEs. Finally, **Rosetta FunFolDes** is applied for sequence design to stabilize the previously refined structure.

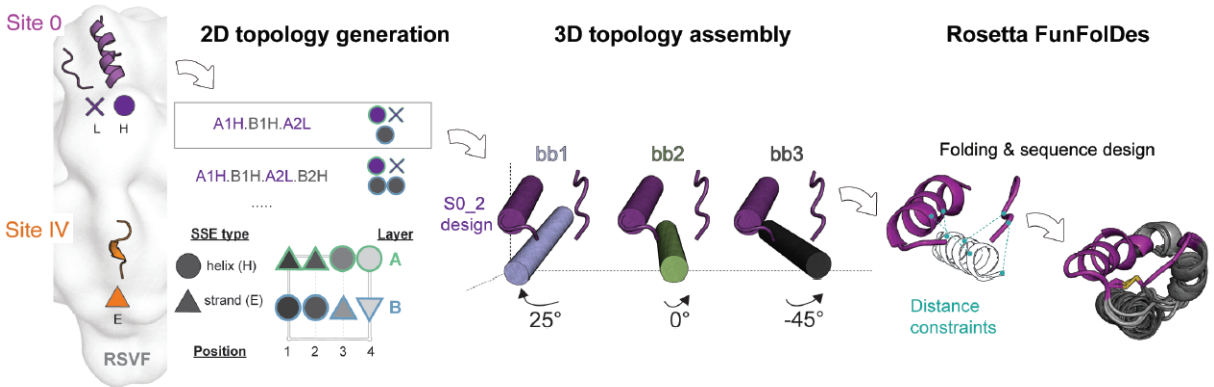


Figure 3: Schematic representation of template-free *de novo* method for RSVF antigenic site 0, where, for example, $A1H$ represents layer A, position 1 with α-helix (H) and $A2L$ represents layer A, position 2 with loop (L). Naming for three generated structural variants in step 2 (topology assembly) are arbitrary.

The screening process for designed immunogens is based on two experimental parameters, folding stability and binding affinity, using yeast display. To evaluate folding stability, deigned immunogen is subject to selective pressure of resistance to nonspecific protease chymotrypsin, which is an enzyme to digest partially unfolded proteins. Next, to evaluate binding affinity, designed immunogen is binding to RSV-nAb (101F antibody). Immunogens with high resistance to chymotrypsin and high binding affinity to 101F antibody are selected.

Finally, designed immunogens for three RSVF antigenic sites are grafted on carrier such as RSV nucleoprotein or ferritin without overlap. It turns out such "cocktail formulation" is able to induce relevant levels of serum neutralization in NHP, which shows the feasibility of computational design for structurally complex motif.

2 Motivation

Continued from the end of Section-1, one can see that template-free *de novo* method discriminates good designed immunogens based on high folding stability and high binding affinity. Although such method shows a powerful functionality, it in turns shows an requirement of intensive experimental labors during screening process. Therefore, one may prefer to further leverage the computational power for screening process.

It is known that such screening properties (discriminators) like folding stability and binding affinity are ensemble averaged properties. As such, a computer simulation which is practically able to calculate statistically meaningful properties, should be a good alternative to experiment. Molecular dynamics (MD) simulation is often applied for such application. Despite limited scales of time and system, it is believed to provide a certain amount of useful information. Consequently, in this report, I aim to find a suitable discriminators based on MD simulation.

3 Method

In order to search for a suitable discriminator, first, a few previously designed immunogens as samples are prepared beforehand. Screened by experimental parameters (folding stabilities and bonding affinities), each immunogen has a tag indicating good/bad. Next, a few MD simulation properties as discriminator candidates are selected. Finally, these candidates are compared to see if they can correctly distinguish a good/bad immunogen. In the following, details about samples, analysis and the simulation setup will be covered.

3.1 Samples

Tag / Type	# Good	# Bad	# Crystal
3E2H	1	1	0
4E2H	1	1	0
4H	3	3	1
4E1H	3	3	1

Table 1: Summary of selected immunogens

Samples are selected from previously designed immunogens. As table-1 shown, I categorize them into four types: *3E2H, 4E2H, 4H* and *4E1H*. Each category contains immunogens with the same epitope-mimicking region and some mutations. Some good immunogens have their experimental structure and are included with tag "Crystal".

The naming principle is straightforward. For illustration, two images are shown by Figure-4. As for α-helix, it is abbreviated as "H" and for β-strand, it is "E". For example, "3E2H" entails that the immunogen consists of three β-strand and two α-helix. Throughout this report, when referring to any specific immunogen, I will call its type first, followed by its tag, and finally its number. For an example, if refer to the second good "4E1H", the naming will simply be "4E1H-g2".

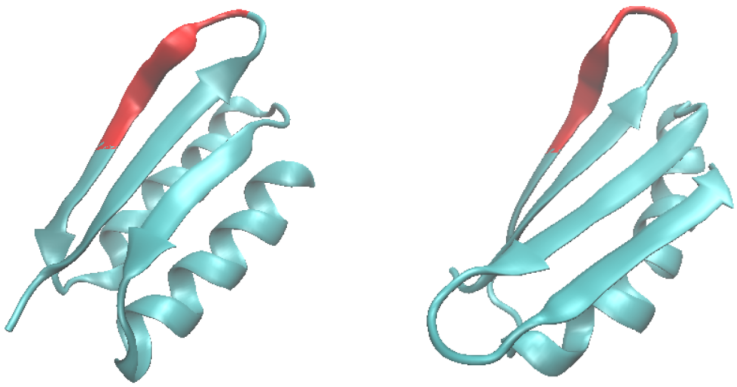


Figure 4: Structures of 3E2H-b1 (left) and 4E1H-g1 (right) where cyan color indicates the immunogen itself, red color indicates the epitope-mimicking part. The arrow motif indicates β-strand and helix motif indicates α-helix.

3.2 Analysis

In this report, starting from the folding stability of imunogens, some relevant MD simulation properties are selected as discriminator candidates. They are root mean square fluctuation (RMSF), root mean square displacement (RMSD), radius of gyration (RGYR) and native contact ratio (NC) respectively. All of them can be analyzed by software **VMD**. An automatic **BASH shell** script to launch several **VMD** analysis on **Charmm36m** trajectories is provided in Appendix-A.1. The principles for RMSF, RMSD and RGYR are very common and I will skip them. In the following, more details about NC will be elaborated.

It is known that RMSD already provides an assessment of the relative displacement for each constituent along simulation. However, the local connections can still hold even with a high RMSD. It means a group can move violently while their connection still remain. To take this exception into account, NC is therefore selected as discriminator candidate. In this report, given a defined radius (R, $R = 4.5\mathring{A}$ for this report), NC is defined as the ratio of number of "remaining" neighboring α-carbons ($N_R(t)$) to number of initial neighboring α-carbons ($N_R(0)$) as follow,

$$NC(t) \equiv \frac{N^R(t)}{N^R(0)} = \frac{\sum_i^n N_i^R(t)}{\sum_i^n N_i^R(0)} \tag{1}$$

where i indicates individual α-carbon. Appendix-A.1 only provides the code to obtain raw data containing all neighbors for each α-carbon along simulation. As for NC vs. time, Appendix-A.2 is the **Python** code for conversion.

Except from RMSF, all properties are as function of time. One can sample properties through the whole simulation given a step (0.2 ns for this report) bigger than the length of correlation time. Therefore, histograms for different properties of immunogens can be obtained. Based on the obtained histograms of certain property, a good/bad immunogen can be discriminated if their distributions are clearly separated. If so, such property as discriminator candidate will be elected. An example **Python** code calculating histogram for RGYR is given in Appendix-A.3.

3.3 Simulation setup

In this section, a rough procedure for simulation setup is shown as below. All the input files preparation can be processed by **Solution Builder** provided by **Charmm-GUI**, which is a free online platform.

1. Before setup:
   All the simulations are done by MD simulation software **Gromacs-2019** on the cluster of **FIDIS** from **EPFL**. Force-fields are **charmm36-mar2019** and **tip3p** (for salvation step). **Verlet** algorithm, leap-frog algorithm (**md**) and steepest descent method (**steep**) are selected as integrator and optimizer.

2. Salvated system setup:
   The PDB file of given immunogen is first deprotonated and then transferred to the center of a cubicbox with size roughly $6 \times 6 \times 6$ nm^3. Second, disulfide bonds are added if there is any. Third, the system is salvated by water molecule box with **Monte Carlo** method (or defining solute-box distance as 0.4 nm). To balance the charge, KCl ions are placed into the system. Next, the system is set for periodic boundary condition. Finally, minimization is run. The cutoffs for coulombic and Van der Waal interaction are both set to 1.2 nm. It will stop when the maximum force **emtol** $\leq 1000.0\ kJmol^{-1}nm^{-1}$.

3. Minimization and equilibration:
   The minimization follows the same one as salvation step. As for equilibration, the system uses the same cutoffs as before and is constrained by **Nose-Hoover** thermostat at $P = 1\ atm$ and $T = 303.15K$. Timestep is set to 0.001 ps with total 125000 steps.

4. Production:
   Production step continues from equilibration step while timestep is extended to 0.002 ps. Total simulation time is set to around 500 ns. Some more production are run if needed.

Alternatively, a simple **Bash shell** script is written for the automation of such process. However, some details for salvation step are still not verified yet. It is included in Appendix-A.4. One is kindly recommended for a view if more details about simulation is interested.

4 Result

4.1 RMSF

In this section, I first introduce the result of RMSF since it doesn't require any histogram analysis. RMSF shows the root mean square displacement for each constituents in the system throughout the simulation, which are α-carbons in this report. I include all the results in Appendix-A.5 due to their image sizes.

As can be seen in the result, there is a special region shown by different background color. Red color is used to indicate region of epitope-mimicking part. X-axis shows the index number. Below, different sequences are listed sequentially from crystal, good ones to bad ones, where alphabets represent corresponding abbreviation for different protein residues. To emphasize sites of mutation, they are colored by red. For crystal, they are all blue.

Before simulation, I was expecting different patterns for epitope-mimicking part and the rest of immunogens as former is designed for the docking with nAbs based on discussion from Section-1. Some special patterns indeed are manifested in the results. 4E2H-g epitope-mimicking part has overall lower RMSF, which entails they are locally constrained. However, opposite results are shown for 3E2H-g and 4E1H-g. Unfortunately, the overall RMSF spectrum shows

no special trends for either good or bad immunogens. To check residues one by one is time-comsuming and not effective, thus a collective behavior such as RMSD, RGYR, NC will be preferred.

4.2 RGYR

RGYR (radius of gyration) shows the relative size of immunogen along simulation and reflects whether the immunogen is under expansion or contraction. Before simulation, it is assumed that good and bad immunogens should possess different distribution in RGYR histogram. That is, a bad immunogen may have an overall higher RGYR since a high RGYR entails an expansion or even unfolding of immunogen, which could hinders the ducking. On the contrary, a good immunogen may have an overall lower RGYR so that the overall shape is smaller, which in turns decreases the steric hindrance.

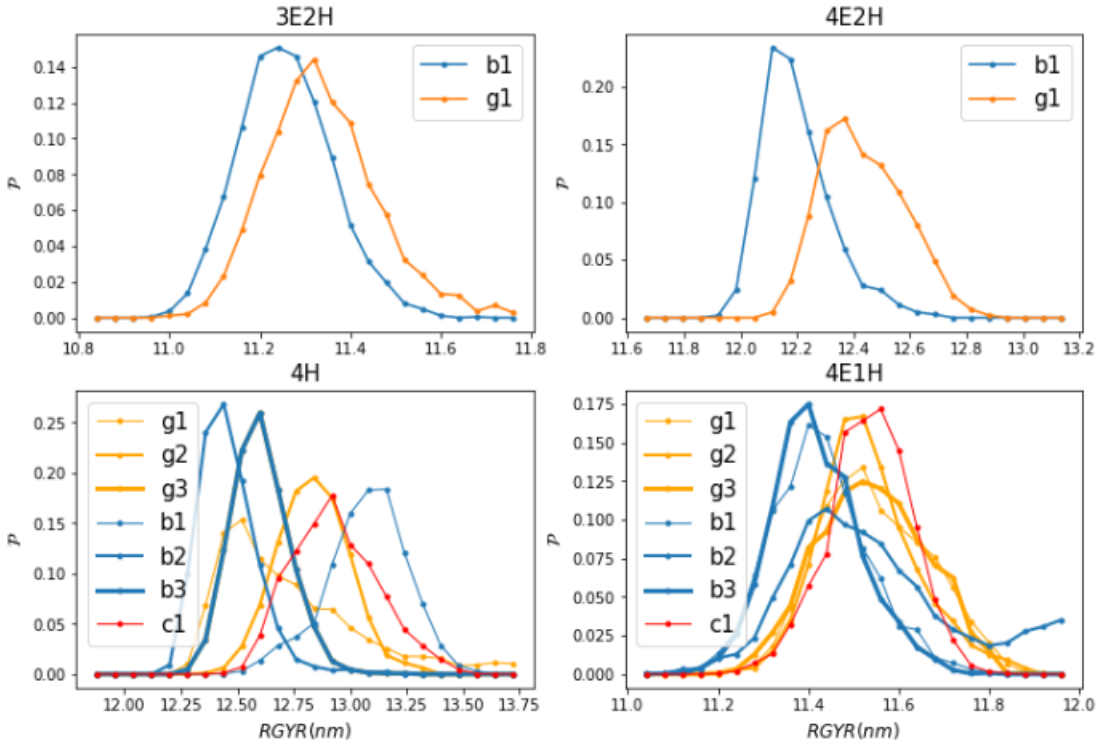


Figure 5: Histogram analysis of RGYR for four types of immunogens, where orange color represents good ones, blue color represents bad ones and red color represents crystal one. Line-width indicates different mutated immunogens. Y-axis ($\mathcal{P}$) is the relative counts for each bin.

Figure-5 shows the RGYR histogram for each immunogen. Starting from 3E2H and 4E2H, it can be seen that good immunogen has an overall higher RGYR compared with bad one. The same trend can be noticed for 4E1H, where good and crystal immunogens have relatively higher RGYR than bad ones do. As for 4H, the distribution of good and crystal immunogen seems to be "sandwiched" by bad ones.

From Figure-5, one may notice that there is no distinct separation between good and bad immunogens. Moreover, these results does not apply to my assumption that a bad immunogen should have a higher RGYR. Instead, at least three types of immunogens such as 3E2H, 4E2H and 4E1H lead to a right opposite conclusion. As such, there must be a different explanation. I stop here and move on to next discriminator candidates.

4.3 RMSD

As described in Section-3.2, RMSD is one way to assess the overall displacement of target along simulation, which reflects the flexibility of target. Such target can be any subgroup of the system. In this report, to isolate the difference of epitope-mimicking part from the rest, two targets are selected (α-carbon of whole immunogen and α-carbon of epitope-mimicking part). Both results will be discussed in the following.

4.3.1 RMSD of immunogen

Figure-6 shows the protein RMSD distribution for four types of immunogens. There is no clear distinction between good and bad immunogens. Compared with Figure-5, one may notice some similar trends. For example of 3E2H and 4E2H, good immunogens have overall higher RMSD. On the other hand, for 4H and 4E1H, good immunogens seem to be "sandwiched" by bad ones. It shows that the bad immunogens may have structures too rigid or too flexible, which leads to a confined movements (low RMSD) or to a violent movements (high RMSD).

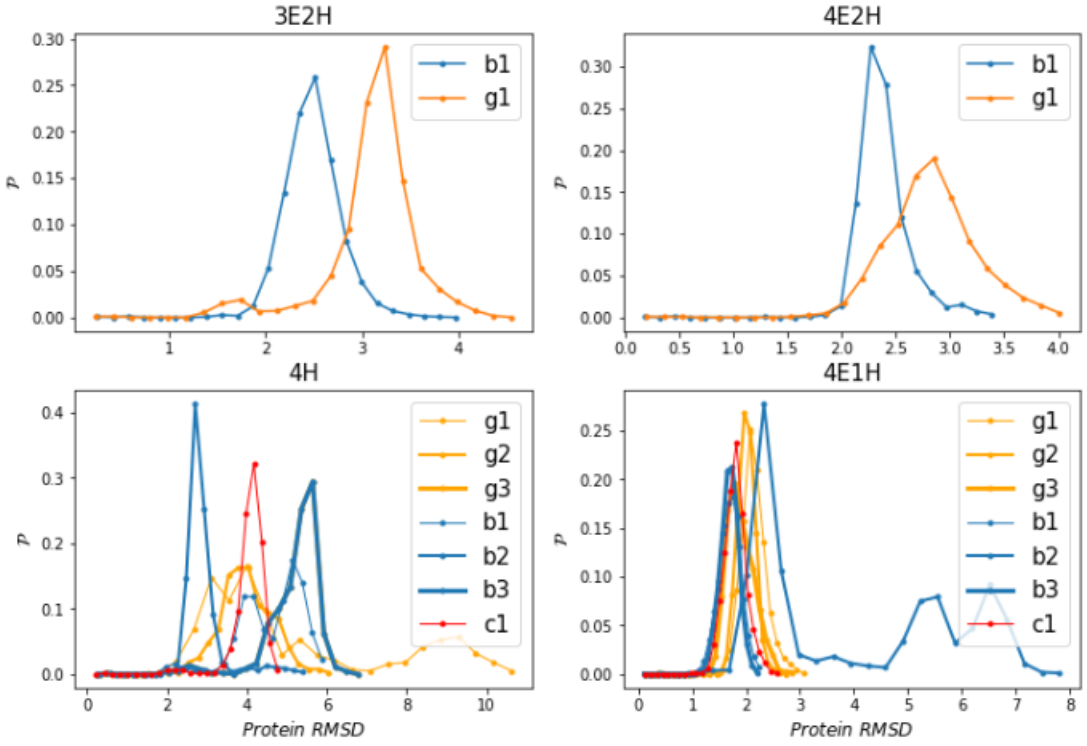


Figure 6: Histogram analysis of RMSD of whole immunogen for four types of immunogens, where orange color represents good ones, blue color represents bad ones and red color represents crystal one. Line-width indicates different mutated immunogens. Y-axis ($\mathcal{P}$) is the relative counts for each bin.

The reason behind such similarity is reasonable. As RMSD goes higher, the sturcture is more flexible. This can result in a higher RGYR and expansion of immunogen. Thus one can see a similar trend such as 3E2H. On the other hand, a flexible chain can also result in a lower RGYR and contraction of immunogen. For example, 4E1H-b2 peaks at a higher RMSD value while peaks at a lower RGYR value. The contraction or expansion of immunogen depends on many factors such as solution, pH value, neighboring interaction and etc.

To sum up, based on the observation, it seems that good immunogens, including crystal ones, may possess some degrees of freedom so that they can "move a little bit", which is helpful to position themselves during docking. On the other hand, bad immunogens have structures too rigid or too flexible. This in turns may hinder the docking process.

4.3.2 RMSD of epitope-mimicking part

From Section-4.3.1, I conclude that good immunogens require mild degree of freedom, which can be expressed in terms of RMSD. RMSD for good immunogens are bounded by RMSD of bad immunogens. It is also important to see how does it behave for epitope-mimicking part. Figure-7 shows the result of it.

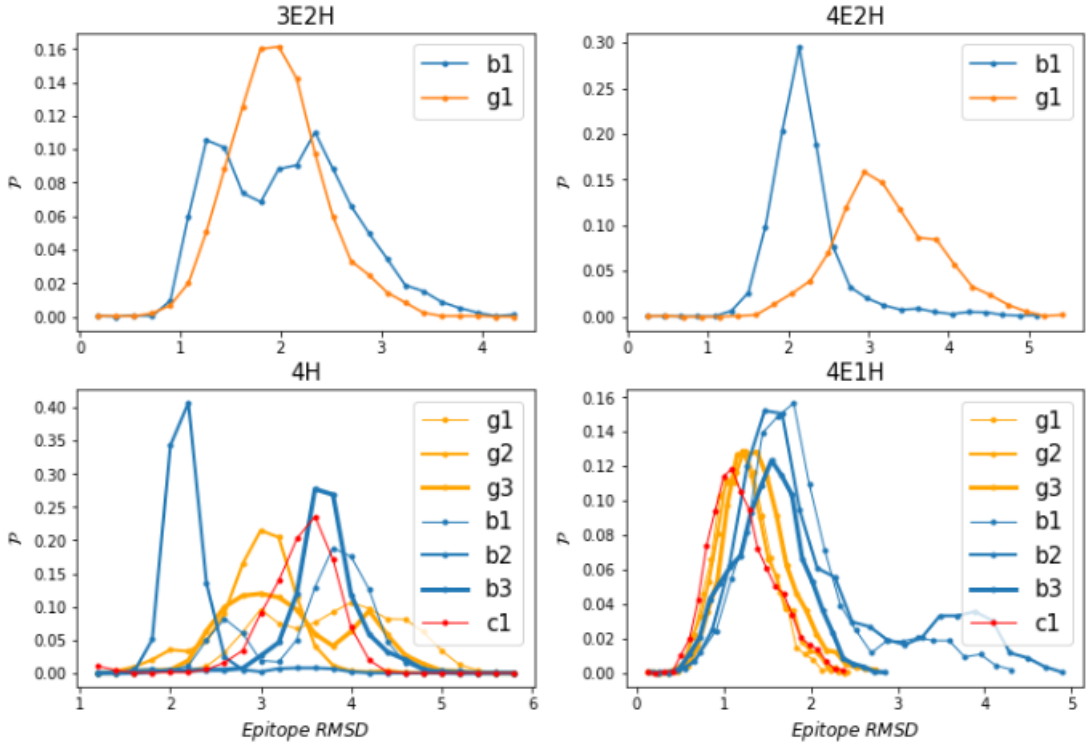


Figure 7: Histogram analysis of RMSD of epitope-mimicking part for four types of immunogens, where orange color represents good ones, blue color represents bad ones and red color represents crystal one. Line-width indicates different mutated immunogens. Y-axis ($\mathcal{P}$) is the relative counts for each bin.

From a simple observation, it can be noticed that some types of immunogens have opposite trends such as 4E2H and 4E1H. In addition, for 3E2H, it is not clear whether good or bad immunogens have higher RMSD. Finally, although it is also not clear, a rough "sandwiched" structure can be observed for 4H. The interpretation of these distribution is relatively complicated. Even such, a similar conclusion from Section-4.3.1 can also apply to epitope-mimicking part of immunogen.

4.4 NC

Aside from RMSD, which shows the flexibility of target, NC provides a way to assess the evolution of overall neighboring connections and helps to check whether the local structure is changes. Results of NC histogram analysis are shown as Figure-8.

First, there is still no clear distinction. Second, it can be noticed that these results no longer follow the same trend as RGYR and RMSD. For example of 3E2H, good immunogen has lower NC, which means the number of leaving neighbors of good immunogen is more than the one of bad immunogens. Although one can explain it arises from the expansion of immunogen, such explanation cannot apply to other types of immunogens. For example, 4E2H has an opposite trend compared with 3E2H. Moreover, 4E1H seems to be no structure at all.

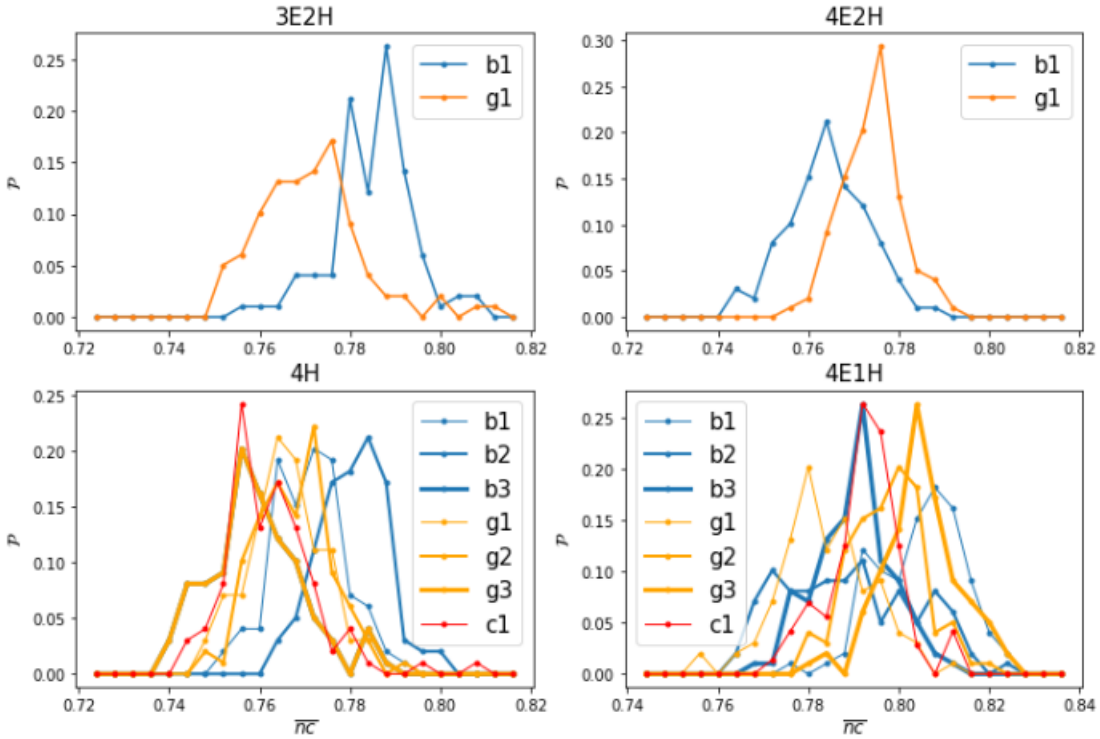


Figure 8: Histogram analysis of NC for four types of immunogens, where orange color represents good ones, blue color represents bad ones and red color represents crystal one. Line-width indicates different mutated immunogens. Y-axis ($\mathcal{P}$) is the relative counts for each bin.

4.5 More samples

For previous study, I simulate only once for each immunogen. To have more independent samples, I simply do 5 more simulation for each immunogen. Results are shown in Appendix-A.6. It shows that some trends do hold the previous trends while some don't. For example, result of RMSD of epitope-mimicking part holds roughly the same trend as previous. Though, more importantly, it should be noted that some distributions seem to unify gradually.

5 Conclusion

In this report, results show that among all four selected discriminators (RMSF, RGYR, RMSD and NC), there is no one capable to distinguish good/bad immunogens. During the analysis, a special trend is observed. It shows that values of RMSD and RGYR of good immunogens are bounded by values of bad immunogens. It is assumed that good immunogens require some degree of freedom, which cannot be too much or too less, so that they can better position themselves during ducking. After increasing sampling size, the distinctions for some cases become blurred. It entails that the parameter may be the same regardless good or bad immunogens. Finally, there could be no such single discriminator capable to distinguish immunogens but a collective behavior of different discriminators instead. Another method to discriminate immunogens based on multi-discriminators will also be preferred.

For the future work, I will first propose to conduct a longer simulation (a few ns). Second, when analyzing multiple independent simulations, it is crucial to look into the trajectories and to make sure they are not experiencing some violent conformational change such as unfolding. In this way, a more robust statistic can be obtained. Finally, I can also try some different analysis parameters and methods.

References

[1] D. Sok and D. R. Burton, "Recent progress in broadly neutralizing antibodies to hiv," *Nature immunology*, vol. 19, no. 11, pp. 1179–1188, 2018.

[2] B. E. Correia, J. T. Bates, R. J. Loomis, G. Baneyx, C. Carrico, J. G. Jardine, P. Rupert, C. Correnti, O. Kalyuzhniy, V. Vittal, *et al.*, "Proof of principle for epitope-focused vaccine design," *Nature*, vol. 507, no. 7491, pp. 201–206, 2014.

[3] F. Sesterhenn, C. Yang, J. Cramer, J. Bonet, X. Wen, L. Abriata, I. Kucharska, C.-I. Chiang, Y. Wang, G. Castoro, *et al.*, "De novo protein design enables precise induction of functional antibodies in vivo," *bioRxiv*, p. 685867, 2020.

A Appendix

A.1 BASH shell script for MD analysis of RMSF, RMSD, RGYR and NC

```
#Make sure files have the same storage hierarchy
LIST_PROTEIN='4e2h 3e2h 4h 4e1h'
LIST_MODIFICATION='g b'
LIST_NUM_EACH='1 2 3'
LIST_REPLICA='r1 r2 r3 r4 r5'
VMD_EXE='PATH_TO_VMD'

for p in $LIST_PROTEIN; do
    for m in $LIST_MODIFICATION; do
        for n in $LIST_NUM_EACH; do
            for r in $LIST_REPLICA; do

FIL_DIR="PATH_TO_DATA/${p}/${m}${n}/${r}"
OUT_DIR="PATH_TO_OUTPUT"
OUTPUT="${OUT_DIR}/log"
INPUT="script_${p}_${m}${n}_${r}"

#Do analysis and skip for exception (see log)
ls $FIL_DIR > /dev/null 2>&1
if [ $? -eq 0 ]; then

#Set up epitope sites
    if [ "$p" = "4e1h" ]; then
        epitope='11 to 16'
    elif [ "$p" = "4e1h" ]; then
        epitope='12 to 17'
    elif [ "$p" = "4e2h" ]; then
        epitope='45 to 50'
    elif [ "$p" = "3e2h" ]; then
        epitope='11 to 39 57 to 71'
    fi

#Create input for VMD tcl script
cat > $OUT_DIR/$INPUT << EOF
#read trj files
mol new $FIL_DIR/step3_charmm2gmx.pdb type pdb\
    first 0 last -1 step 1 filebonds 1 autobonds 1 waitfor all
mol addfile $FIL_DIR/step3_charmm2gmx.psf type psf\
    first 0 last -1 step 1 filebonds 1 autobonds 1 waitfor all
mol addfile $FIL_DIR/step4.1_equilibration-noPBC.xtc type xtc\
    first 0 last -1 step 20 filebonds 1 autobonds 1 waitfor all
mol addfile $FIL_DIR/step5_1-noPBC.xtc type xtc\
    first 0 last -1 step 1 filebonds 1 autobonds 1 waitfor all
mol addfile $FIL_DIR/step5_1.part0002-noPBC.xtc type xtc\
    first 0 last -1 step 1 filebonds 1 autobonds 1 waitfor all
```

```
#set frame up to 2500-
animate delete beg 2500 end 99999 skip -1

#align alpha carbon "CA" w.r.t frame 0
set ref [atomselect top "protein and name CA" frame 0]
set sel [atomselect top "protein and name CA"]
set all [atomselect top "all"]
set n [molinfo top get numframes]
for {set i 1} {\$i < \$n} {incr i} {
    \$sel frame \$i
    \$all frame \$i
    \$all move [measure fit \$sel \$ref]
    }

#calc & out protein CA rmsd
set outfile [open "$OUT_DIR/${p}_${m}${n}_protein_rmsd.txt" w]
set ref [atomselect top "protein and name CA" frame 0]
set sel [atomselect top "protein and name CA"]
set n [molinfo top get numframes]
for {set i 1} {\$i < \$n} {incr i} {
  \$sel frame \$i
  set rmsd [measure rmsd \$sel \$ref]
  puts \$outfile "\$i \$rmsd"
  }
close \$outfile

#calc & out epitope CA rmsd
set outfile [open "$OUT_DIR/${p}_${m}${n}_epitope_rmsd.txt" w]
set ref [atomselect top "protein and name CA and (resid $epitope)" frame 0]
set sel [atomselect top "protein and name CA and (resid $epitope)"]
for {set i 0} {\$i < \$n} {incr i} {
  \$sel frame \$i
  set rmsd [measure rmsd \$sel \$ref]
  puts \$outfile "\$i \$rmsd"
}
close \$outfile

#calc & out CA rmsf
set outfile [open "$OUT_DIR/${p}_${m}${n}_rmsf.txt" w]
set sel [atomselect top "protein and name CA"]
set mol [\$sel molindex]
for {set i 0} {\$i < [\$sel num]} {incr i} {
    set rmsf [measure rmsf \$sel]
    puts \$outfile "[expr {\$i+1}] [lindex \$rmsf \$i]"
    }
close \$outfile

#calc & out rgyr
set outfile [open "$OUT_DIR/${p}_${m}${n}_rgyr.txt" w]
set mol [molinfo top]
set sel [atomselect top "protein"]
set frames [molinfo \$mol get numframes]
for {set i 0} {\$i < \$frames} {incr i} {
    \$sel frame \$i
    \$sel update
    set a [measure rgyr \$sel]
    puts \$outfile "\$i \$a"
    }
close \$outfile

#calc & out cn
set outfile [open "$OUT_DIR/${p}_${m}${n}_cn.txt" w]
set nf [molinfo top get numframes]
set ca [atomselect top "name CA"]
```

```
set nca [\$ca num]
puts \$outfile "\$nca"
for {set f 0} {\$f < \$nf} {incr f 25} {
    foreach index [\$ca get index] {
        set a [atomselect top "(all within 4.5 of index \$index)\
                               and ({name \"C.*\"} or {name \"N.*\"}\
                               or {name \"O.*\"} and not {name \"OH.*\"})"]
        \$a frame \$f
        \$a update
        set cn [\$a get index]
        puts \$outfile "\$f \$index \$cn"
    }
}
close \$outfile
exit
EOF

#Do VMD analysis
  eval "$VMD_EXE -e $INPUT"
  rm $INPUT

#Write exception to log
else
  echo "PROTEIN ${p}_${m}${n} CANNOT FOUND" >> $OUTPUT
fi
            done
        done
    done
done
```

A.2 Python code to convert raw NC data

```
import numpy as np
#Make sure files have the same storage hierarchy and naming style
for k in ['3e2h','4e2h','4e1h','4h']:
    for l in ['b','g','c']:
        for t in range(1,4,1):
            for j in ['r1','r2','r3','r4','r5']:
            #Do NC conversion and skip the exception
                try :
                    #Read file
                    f = open(str(k) + "_" + str(l) + str(t) + "_" + j + "_cn.txt")
                    data=[]
                    data = f.readlines()[0:]
                    n_ca = int(data[0])
                    lines = data[1:]
                    nf = len(lines[0::n_ca])
                    ic=0
                    for i in range(n_ca):
                        ic+=len(lines[0:n_ca][i].split())-2
                    nc = [ic]
                    for frame in range(0,nf):
                        nc.append(ic)
                        for index in range(n_ca):
                            for i in lines[index::n_ca][0].split():
                                if i in lines[index::n_ca][frame].split():
                                    pass
                                else: nc[frame]-=1
                    native_contact=[]
                    for i in range(nf):
                        native_contact.append([i*25, float(nc[i])/float(nc[0])])
                    native_contact=np.array(native_contact)
                    #Output file
                    with open(str(k) + "_" + str(l) + str(t) + "_" + j + "-cn.txt", 'w') as f:
                        for i in range(len(native_contact)):
```

```
                        f.write("%s %s \n"   %(native_contact[i][0], native_contact[i][1]))
                except:
                    pass
```

A.3 Python code for histogram analysis (RGYR)

```
import numpy as np
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2,2, figsize=(15, 10), facecolor='w', edgecolor='k')
fig.subplots_adjust(hspace = 0.35, wspace= 0.25)
axs = axs.ravel()
proteins=['3e2h','4e2h','4h','4e1h']
colors={'b':'tab:blue','g':'orange','c':'red'}

#Make sure the files have the same storage hierarchy and naming style.
for i in range(len(proteins)):
    for j in ['b','g','c']:
        for k in range(1,4,1):
            try:
                #Read files
                #This code is for RGYR, while one is welcome to modify for different properties
                #such as RMSD, CN etc. For example, change file name below to '_protein_rmsd'
                f = open('../pipeline/' + proteins[i] + '_' + j + str(k) + '_rgyr.txt')
                lines = f.readlines()
                tmp = []
                for line in lines:
                    try:
                        tmp.append([float(line.split()[0]), float(line.split()[1])])
                    except:
                        pass

                #Calculate histogram
                tmp=np.array(tmp)
                mi=np.min(tmp[:,1])-0.1
                mx=np.max(tmp[:,1])+0.1
                step=(mx-mi+0.2)/25
                test=np.histogram(tmp[:,1], bins=np.arange(mi,mx,step), density=True)

                #Plot
                axs[i].plot(test[1][1:], test[0]/np.sum(test[0]), label=j+str(k), marker='o',\
                            markersize=3, color=colors[j], lw=k)
                axs[i].axvline(test[1][np.argmax(test[0])+1], color=colors[j], linestyle='--',\
                               lw=1)
                axs[i].legend(fontsize=15)
                axs[i].set_ylabel("$\mathcal{P}$", fontsize=12)
                axs[i].set_xlabel("$RGYR$", fontsize=12)
                axs[i].set_title(proteins[i], fontsize=15)
            except: pass
plt.savefig('./rgyr.png')
```

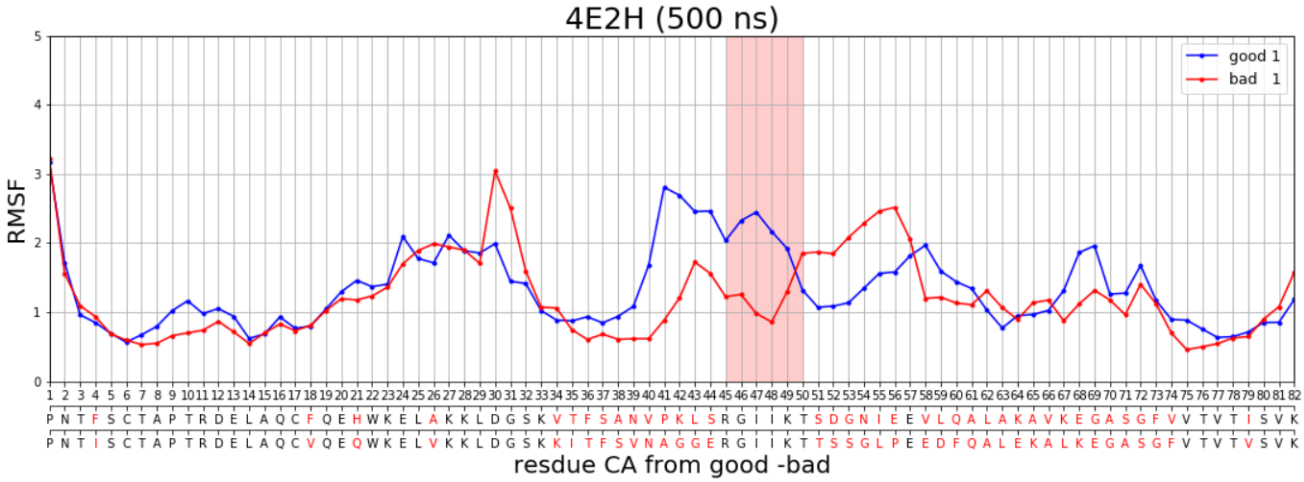
A.4 BASH shell script for single immunogen (3E2H_g1) input preparation

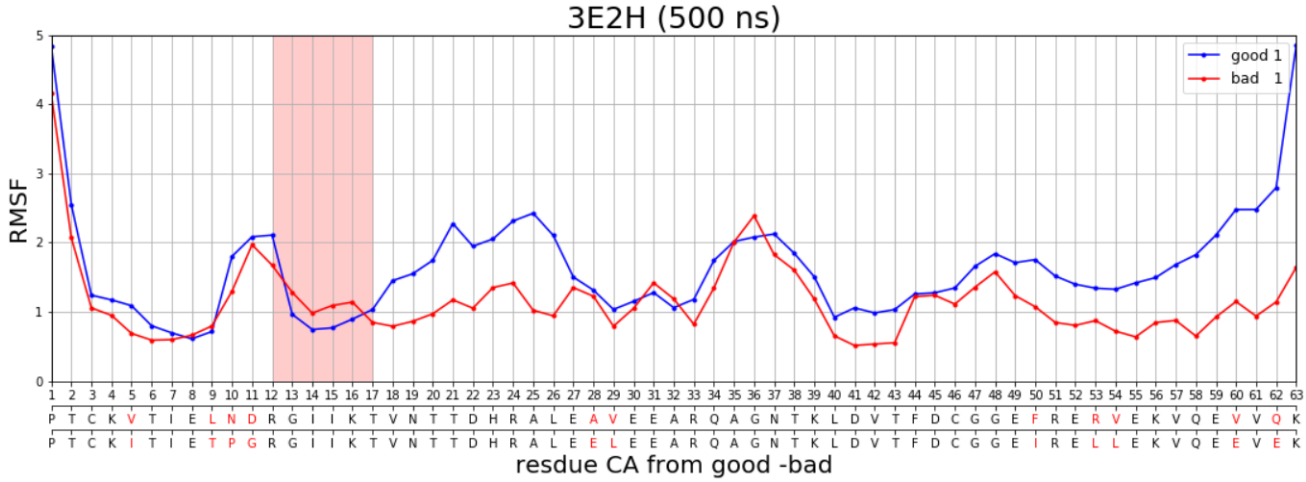
```
#Make sure input file is presence in the same folder of script
#Load required module
module purge
module load gcc mvapich2 gromacs/2019.2-mpi

#Set up input and output name ; only support PDB file
#The input file should be already deprotonated
INPUT='dH_3E2H_enrich_best' #No need to add .pdb
OUTPUT='3E2H_enrich_best'   #No need to add .pdb

#Get the force-field from internet
wget http://mackerell.umaryland.edu/download.php?filename=CHARMM_ff_params_files/charmm36-mar2019.
```

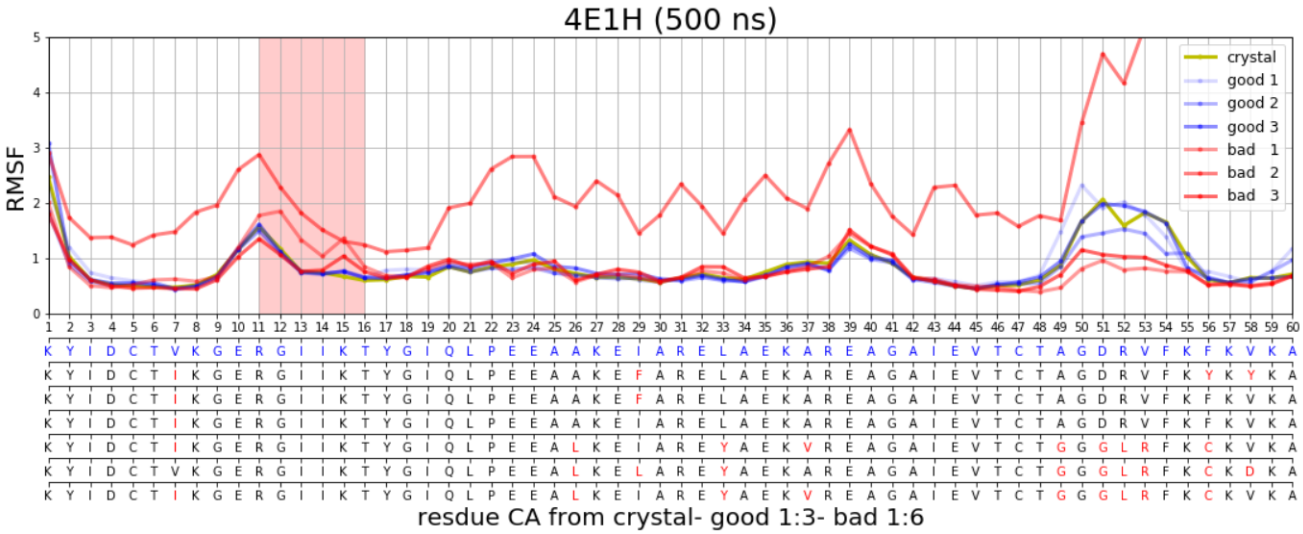
```
mv download.php?filename=CHARMM_ff_params_files%2Fcharmm36-mar2019.ff.tgz charmm36-mar2019.ff.tgz
tar -xvf charmm36-mar2019.ff.tgz
rm charmm36-mar2019.ff.tgz

#Create input file for salvation step
cat > ions.mdp << EOF
integrator  = steep
emtol       = 1000.0
emstep      = 0.01
nsteps      = 50000
nstlist         = 1
cutoff-scheme   = Verlet
ns_type         = grid
coulombtype     = cutoff
rcoulomb        = 1.2
rvdw            = 1.2
pbc             = xyz
EOF


#Start processing
#Solvate system
echo "yes" | gmx_mpi pdb2gmx -f ${INPUT}.pdb -o ${OUTPUT}.gro -ff charmm36-mar2019\
                            -water tip3p -ss yes
gmx_mpi editconf -f ${OUTPUT}.gro -o ${OUTPUT}.gro -c -d 0.5 -bt cubic
gmx_mpi solvate -cp ${OUTPUT}.gro -cs spc216.gro -o ${OUTPUT}.gro -p topol.top
gmx_mpi grompp -f ions.mdp -c ${OUTPUT}.gro -p topol.top -o ions.tpr
echo "13" | gmx_mpi genion -s ions.tpr -o ${OUTPUT}.gro -p topol.top -pname K -nname CL -neutral
gmx_mpi editconf -f ${OUTPUT}.gro -o step3_charmm2gmx.pdb
echo "q" | gmx_mpi make_ndx -f ${OUTPUT}.gro -o index.ndx
rm \#* mdout.mdp posre.itp ions.*


#Create input file for minimization, equilibration and production steps
cat > step4.0_minimization.mdp << EOF
define                  = -DREST_ON -DSTEP4_0
integrator              = steep
emtol                   = 1000.0
nsteps                  = 5000
nstlist                 = 10
cutoff-scheme           = Verlet
rlist                   = 1.2
vdwtype                 = Cut-off
vdw-modifier            = Force-switch
rvdw_switch             = 1.0
rvdw                    = 1.2
coulombtype             = pme
rcoulomb                = 1.2
constraints             = h-bonds
constraint_algorithm    = LINCS
EOF

cat > step4.1_equilibration.mdp << EOF
define                  = -DREST_ON -DSTEP4_1
integrator              = md
dt                      = 0.001
nsteps                  = 125000
nstxtcout               = 5000
nstvout                 = 5000
nstfout                 = 5000
nstcalcenergy           = 100
nstenergy               = 1000
nstlog                  = 1000
cutoff-scheme           = Verlet
nstlist                 = 20
```

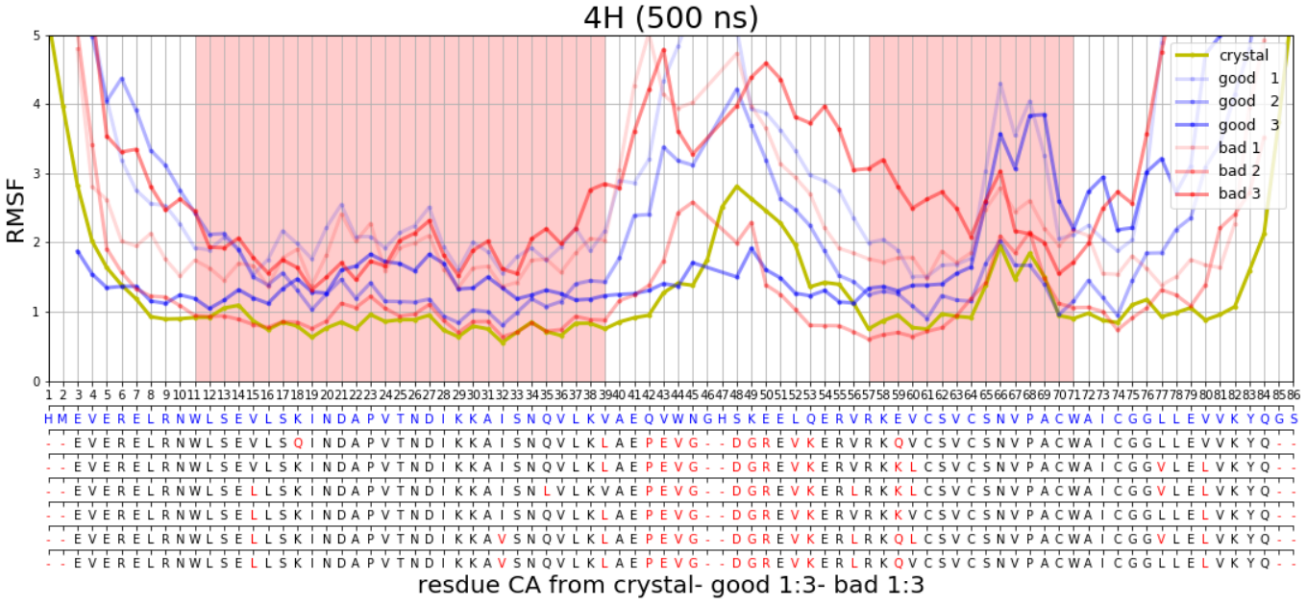
```
rlist                    = 1.2
coulombtype              = pme
rcoulomb                 = 1.2
vdwtype                  = Cut-off
vdw-modifier             = Force-switch
rvdw_switch              = 1.0
rvdw                     = 1.2
tcoupl                   = Nose-Hoover
tc_grps                  = Protein Water_and_ions
tau_t                    = 1.0     1.0
ref_t                    = 303.15    303.15
constraints              = h-bonds
constraint_algorithm     = LINCS
nstcomm                  = 100
comm_mode                = linear
comm_grps                = Protein Water_and_ions
gen-vel                  = yes
gen-temp                 = 303.15
gen-seed                 = -1
refcoord_scaling         = com
EOF

cat > step5_production.mdp << EOF
integrator               = md
dt                       = 0.002
nsteps                   = 5000000000
nstxout                  = 0         ; suppress bulky .trr file by specifying
nstvout                  = 0         ; 0 for output frequency of nstxout,
nstfout                  = 0         ; nstvout, and nstfout
nstenergy                = 100000      ; save energies every 10.0 ps
nstlog                   = 100000      ; update log file every 10.0 ps
nstxout-compressed       = 100000      ; save compressed coordinates every 200 ps
cutoff-scheme            = Verlet
nstlist                  = 20
rlist                    = 1.0
coulombtype              = pme
rcoulomb                 = 1.2
vdwtype                  = Cut-off
vdw-modifier             = Force-switch
rvdw_switch              = 0.8
rvdw                     = 1.2
tcoupl                   = Nose-Hoover
tc_grps                  = Protein Water_and_ions
tau_t                    = 1.0     1.0
ref_t                    = 303.15 303.15
pcoupl                   = Parrinello-Rahman
pcoupltype               = isotropic
tau_p                    = 5.0
compressibility          = 4.5e-5
ref_p                    = 1.0
constraints              = h-bonds
constraint_algorithm     = LINCS
continuation             = yes
nstcomm                  = 100
comm_mode                = linear
comm_grps                = Protein Water_and_ions
refcoord_scaling         = com
EOF
```

A.5 Results of RMSF

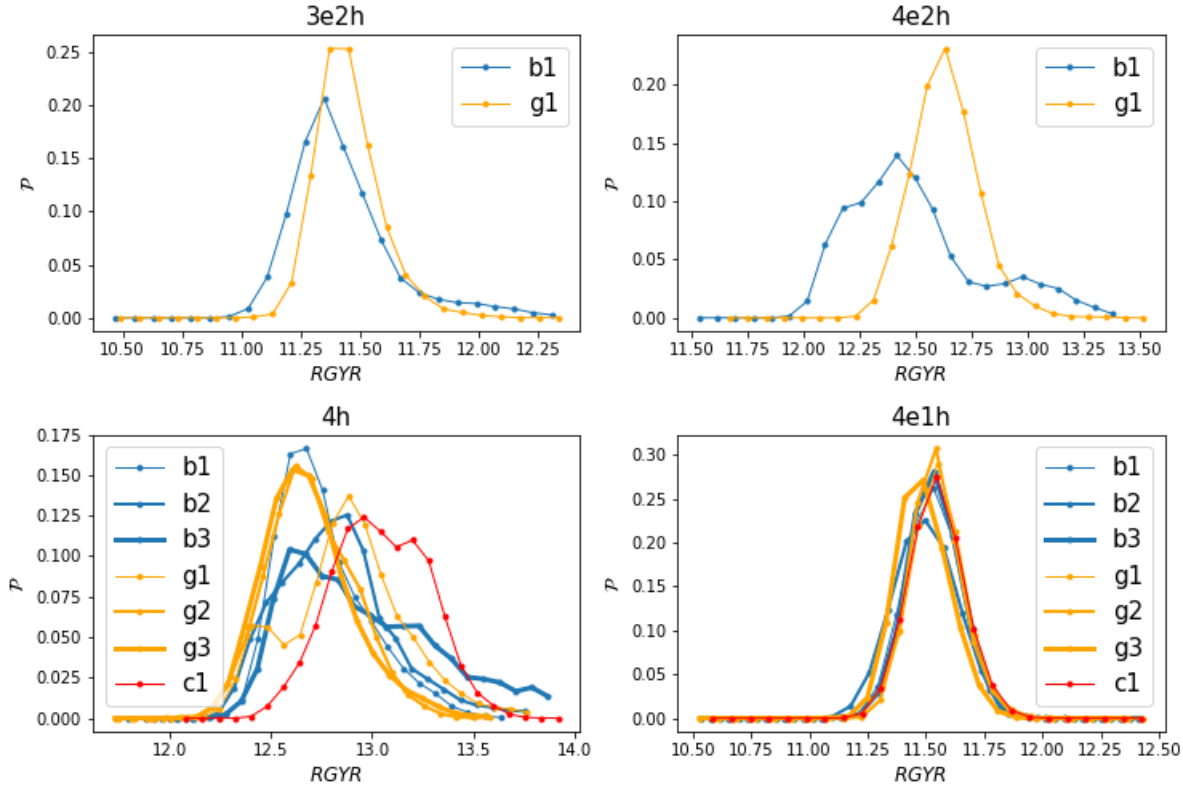


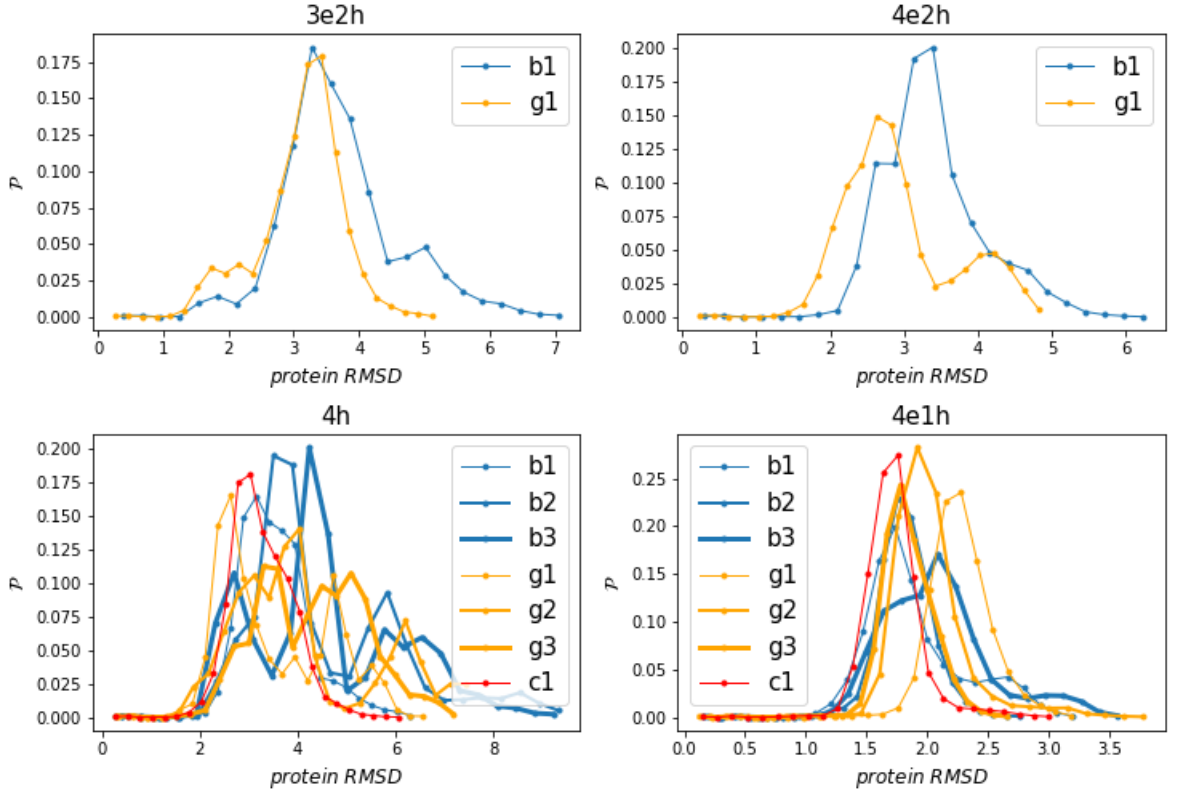


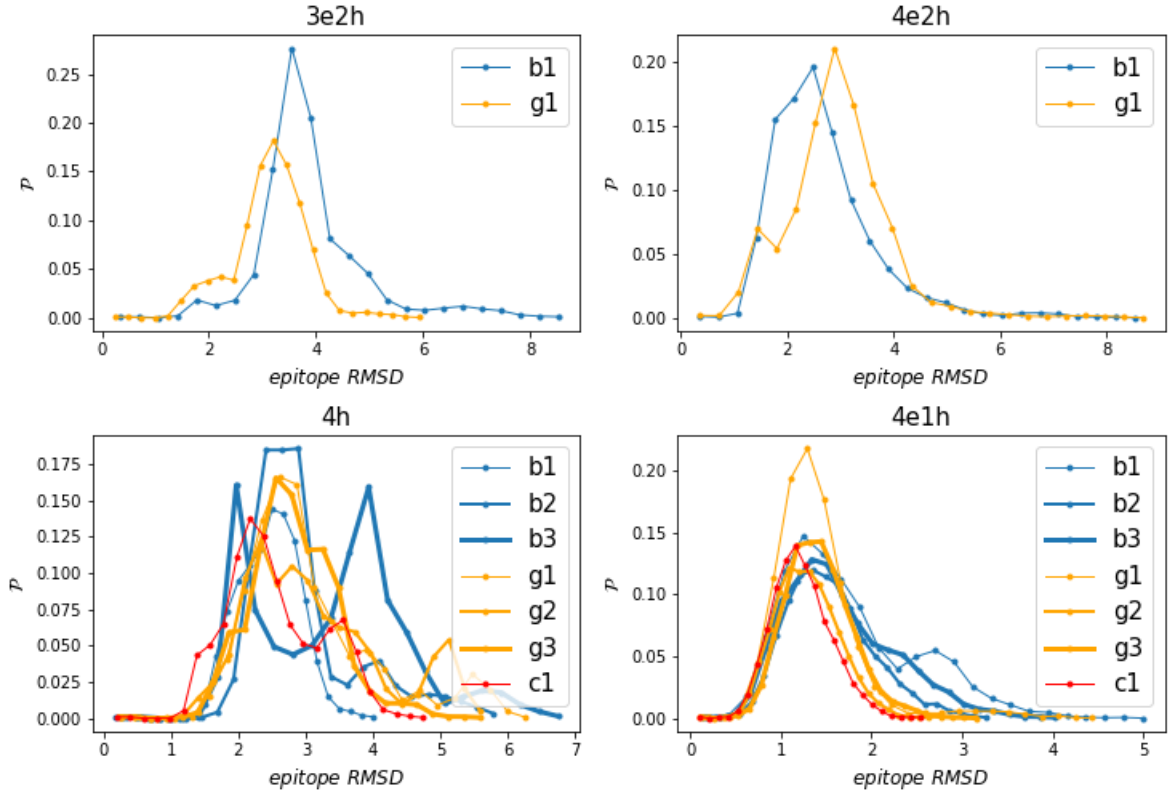


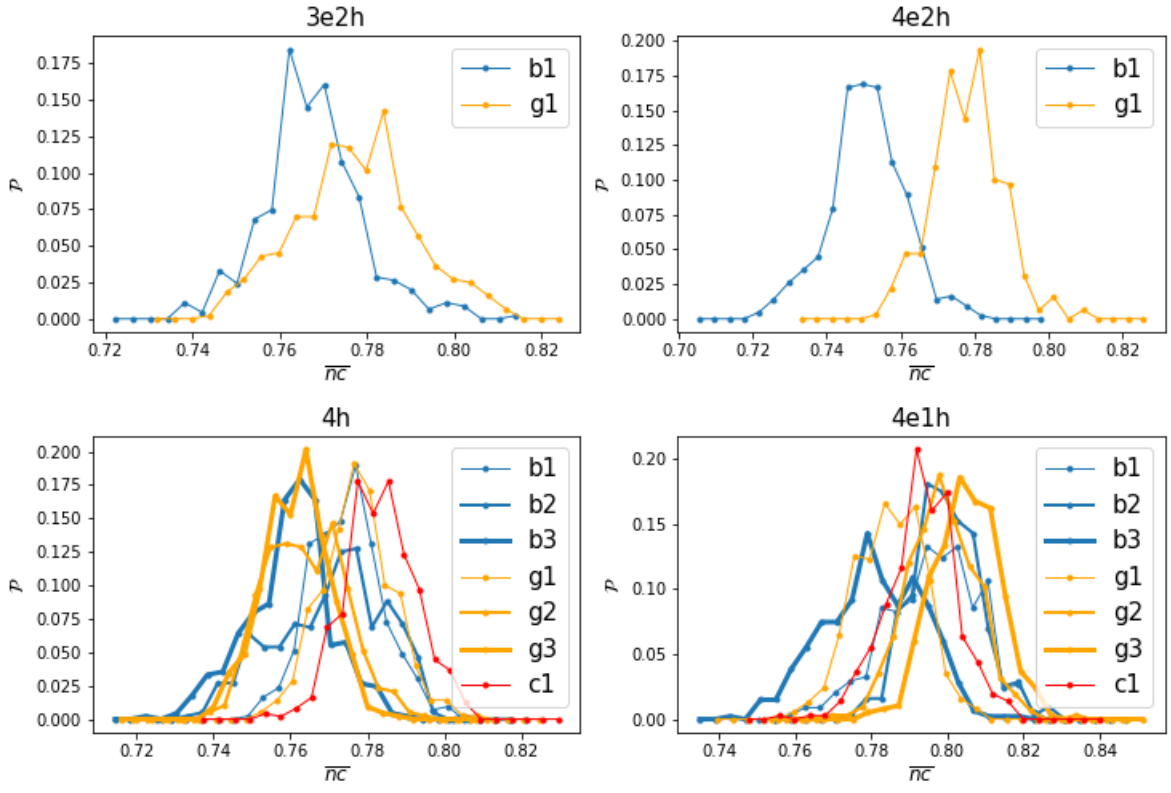


A.6 Results with more samples (5) included


3e2h
4e2h
4h
4e1h
b1
b2
b3
g1
g2
g3
c1
RGYR


3e2h
4e2h
4h
4e1h
b1
b2
b3
g1
g2
g3
c1
protein RMSD


3e2h
4e2h
4h
4e1h
b1
b2
b3
g1
g2
g3
c1
epitope RMSD


3e2h
4e2h
4h
4e1h
b1
b2
b3
g1
g2
g3
c1
nc