7. ANNEX I. TRAJECTORIES ANALYSIS PLOTS

7.1 rMD simulations with CHARMM27

7.1.1 RMSD

The RMSD plots from the rMD simulations with CHARMM27 at 360 K, 380 K, 400 K, 420 K, 450 K and 500 K are shown below.

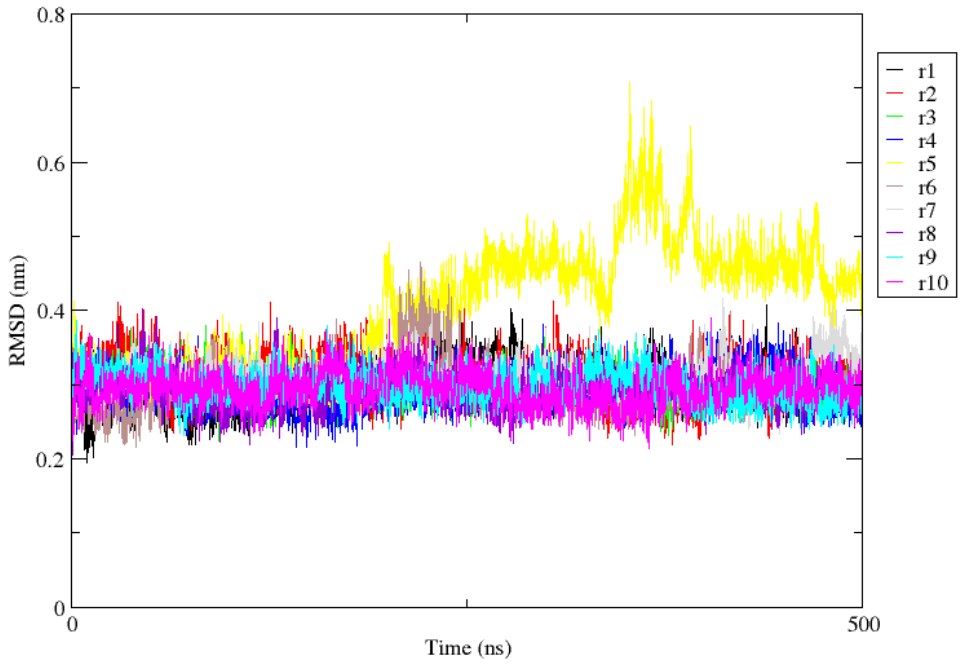

Figure 32. RMSD for the simulations at 360 K using the CHARMM27 force field.

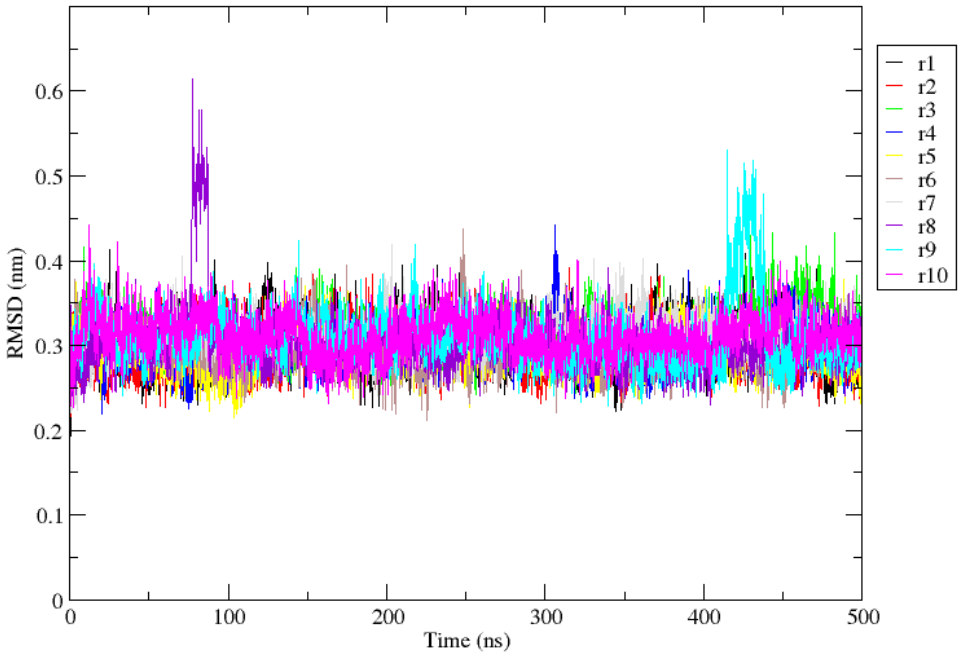

Figure 33. RMSD for the simulations at 380 K using the CHARMM27 force field.

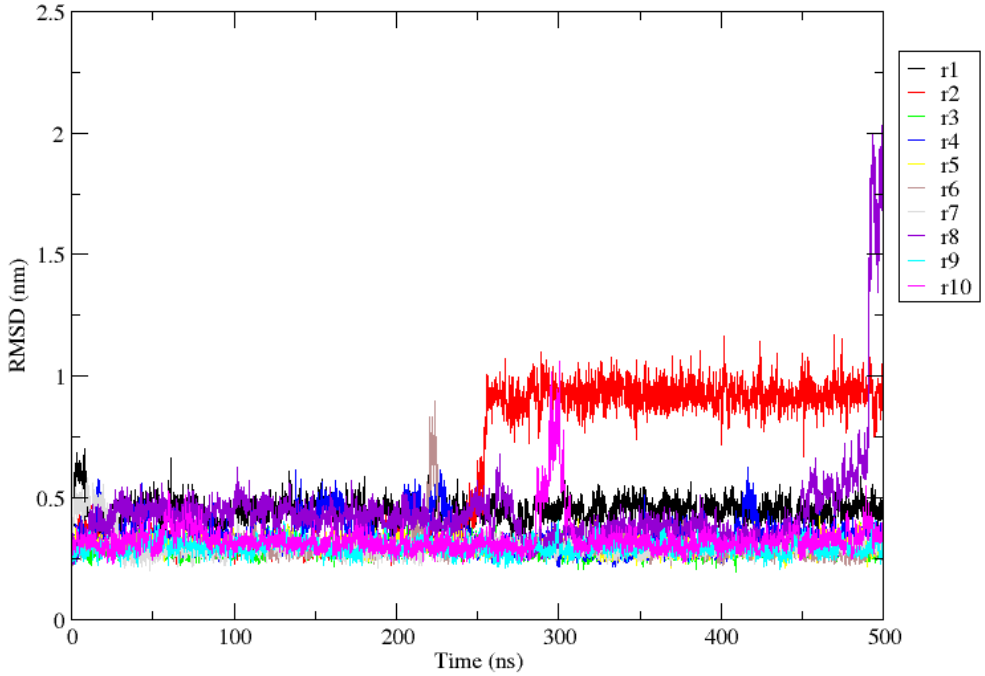

Figure 34. RMSD for the simulations at 400 K using the CHARMM27 force field.

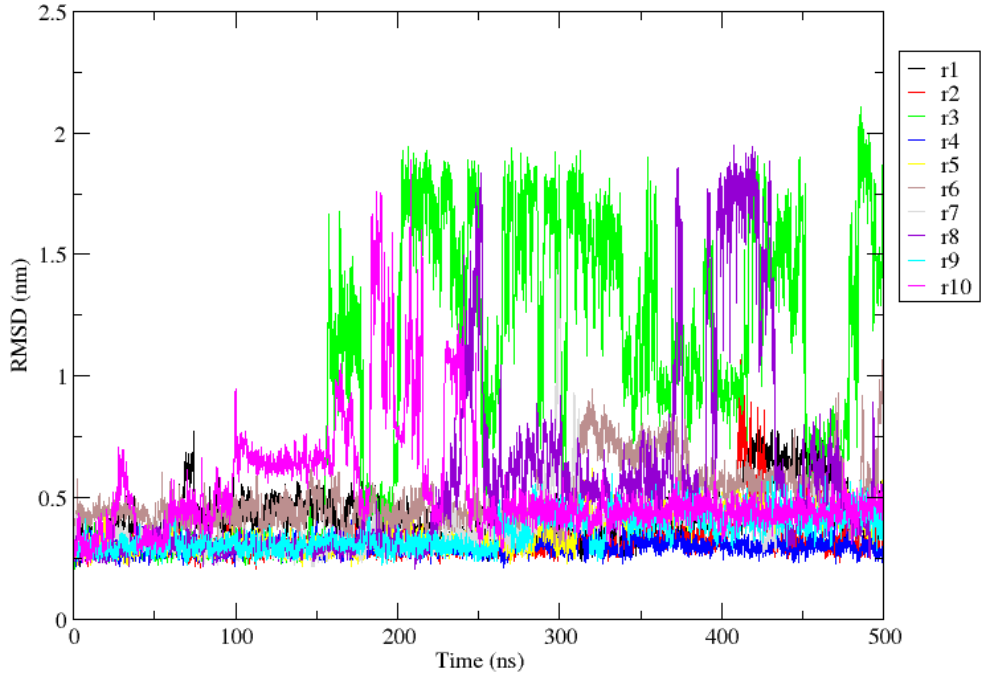

Figure 35. RMSD for the simulations at 420 K using the CHARMM27 force field.

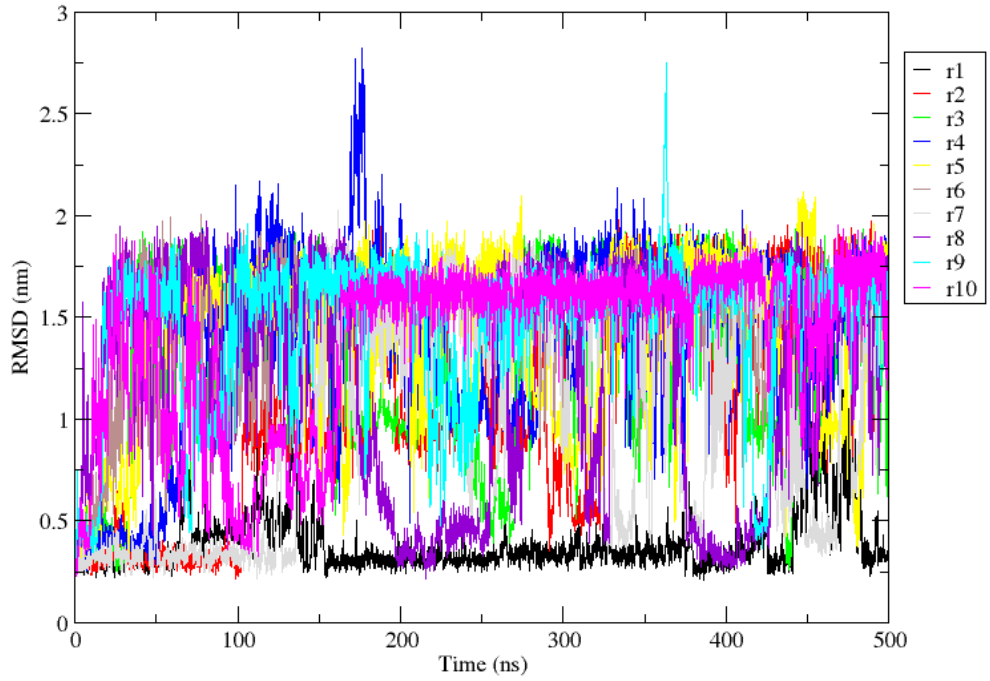

Figure 36. RMSD for the simulations at 450 K using the CHARMM27 force field.

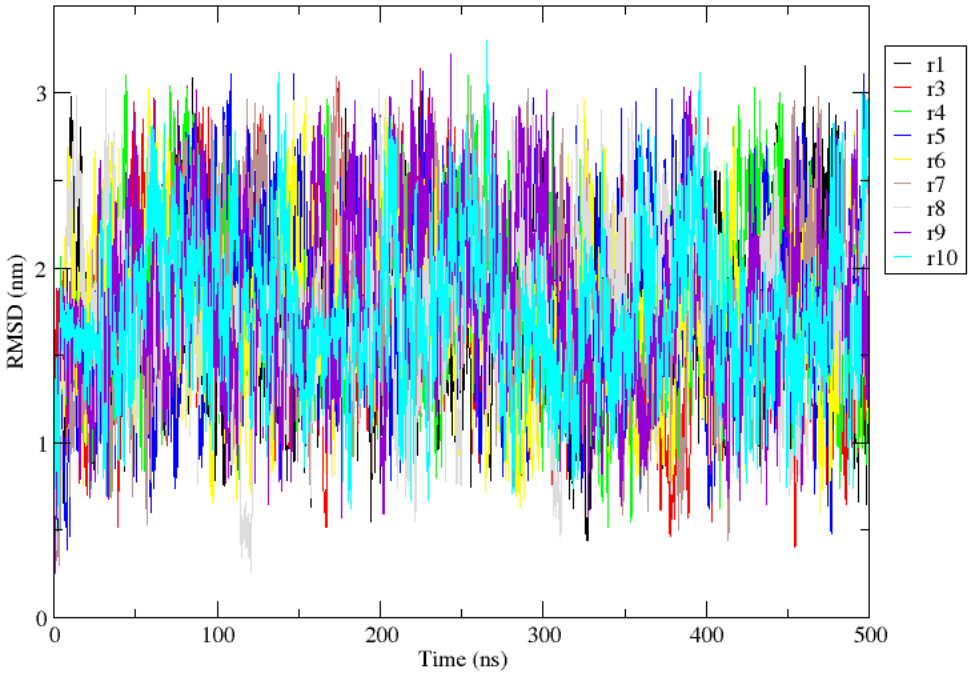

Figure 37. RMSD for the simulations at 500 K using the CHARMM27 force field.

7.1.2 Rg

The Rg plots from the rMD simulations with CHARMM27 at 360 K, 380 K, 400 K, 420 K, 450 K and 500 K are shown below.

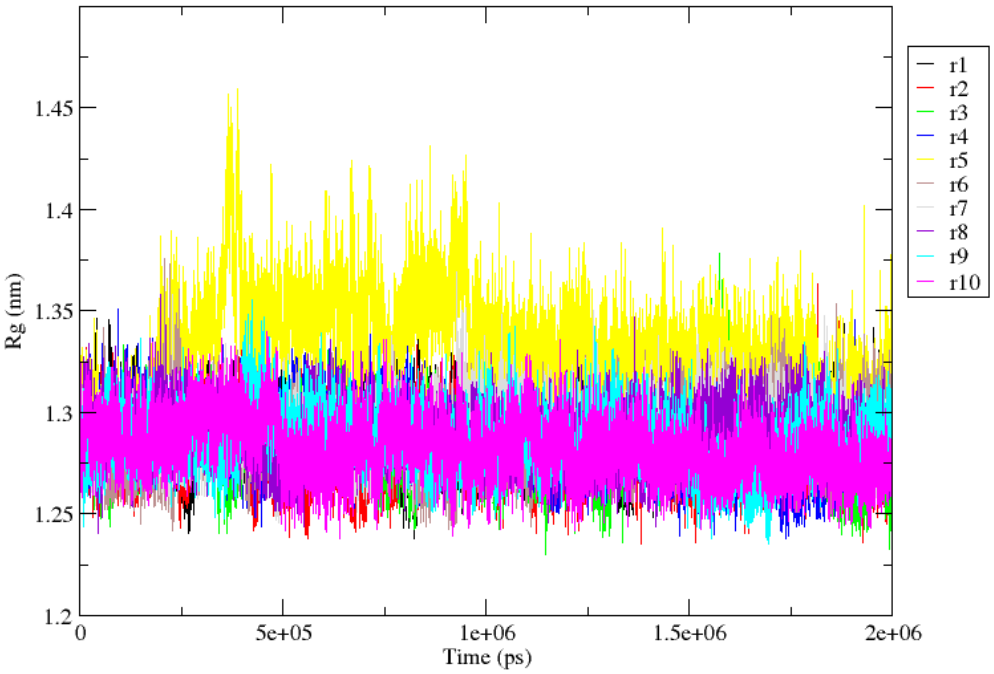

Figure 38. Rg for the simulations at 360 K using the CHARMM27 force field.

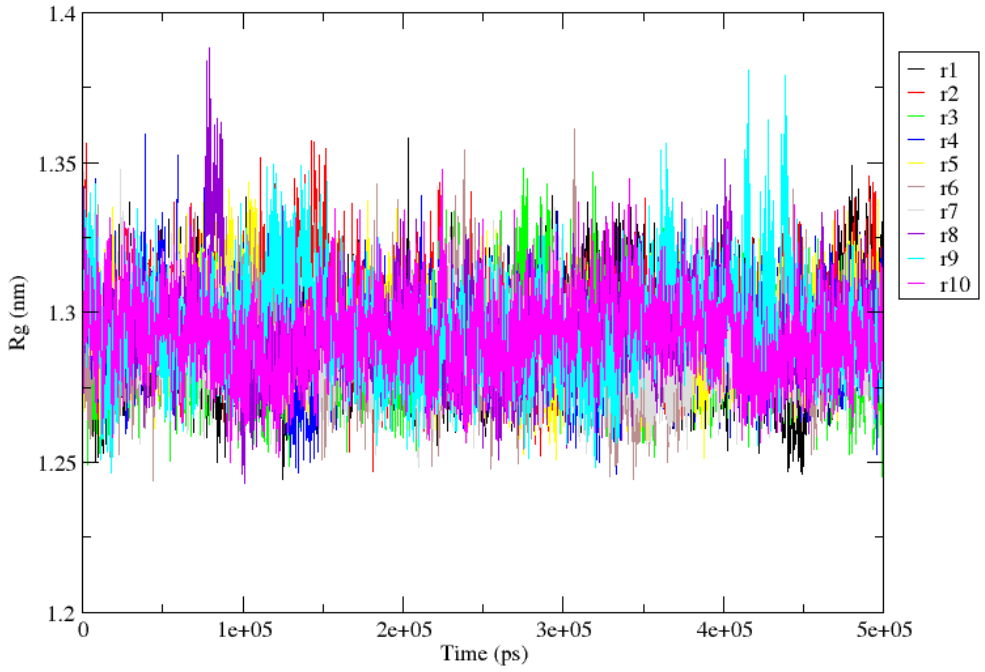

Figure 39. Rg for the simulations at 380 K using the CHARMM27 force field.

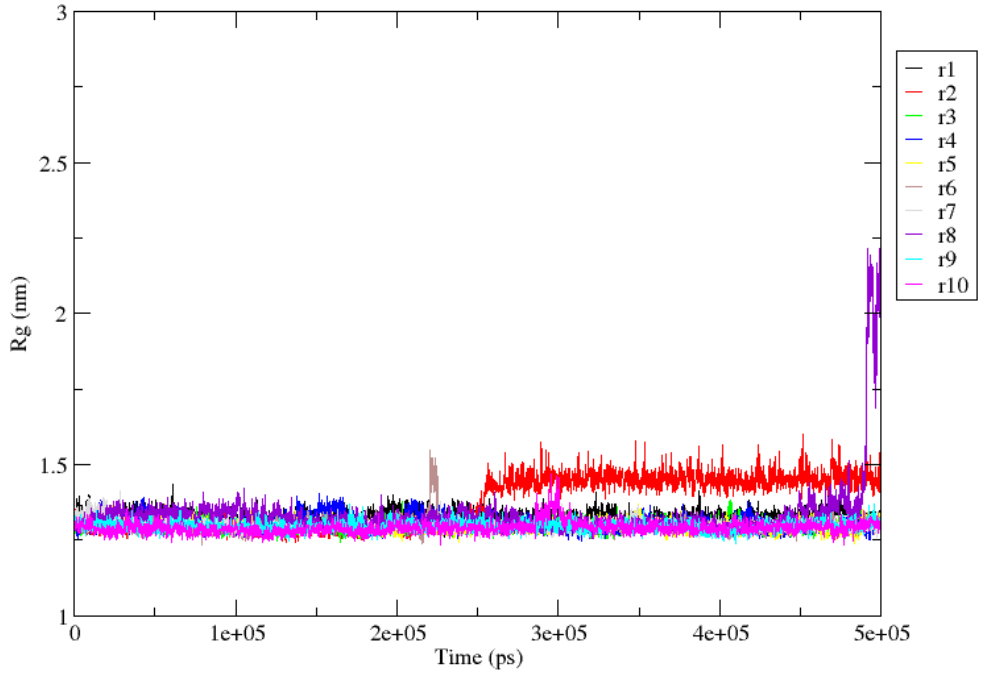

Figure 40. Rg for the simulations at 400 K using the CHARMM27 force field.

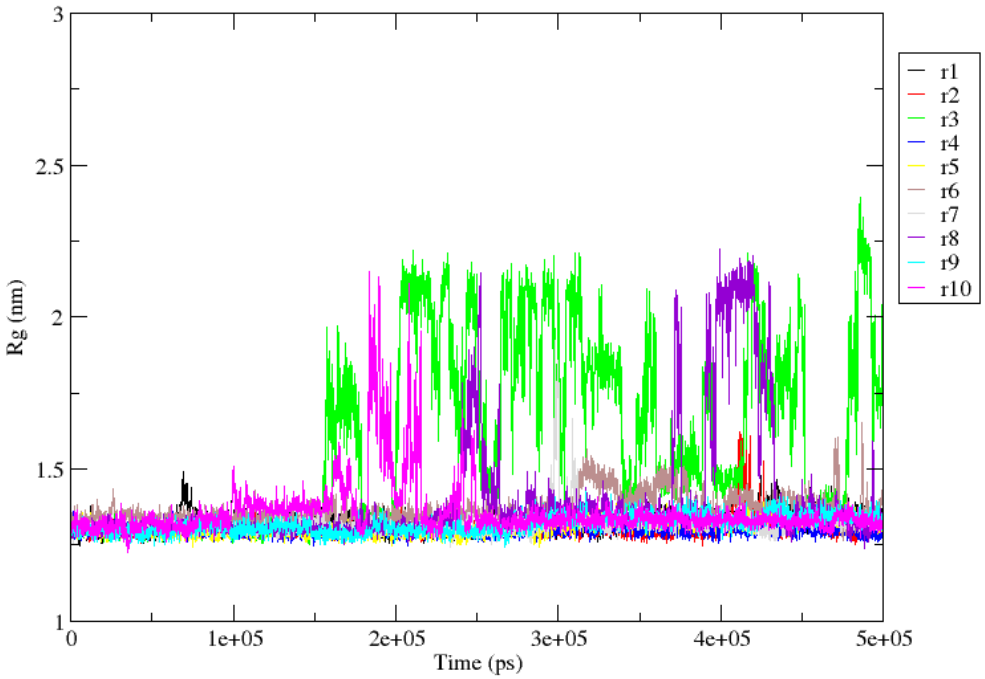

Figure 41. Rg for the simulations at 420 K using the CHARMM27 force field.

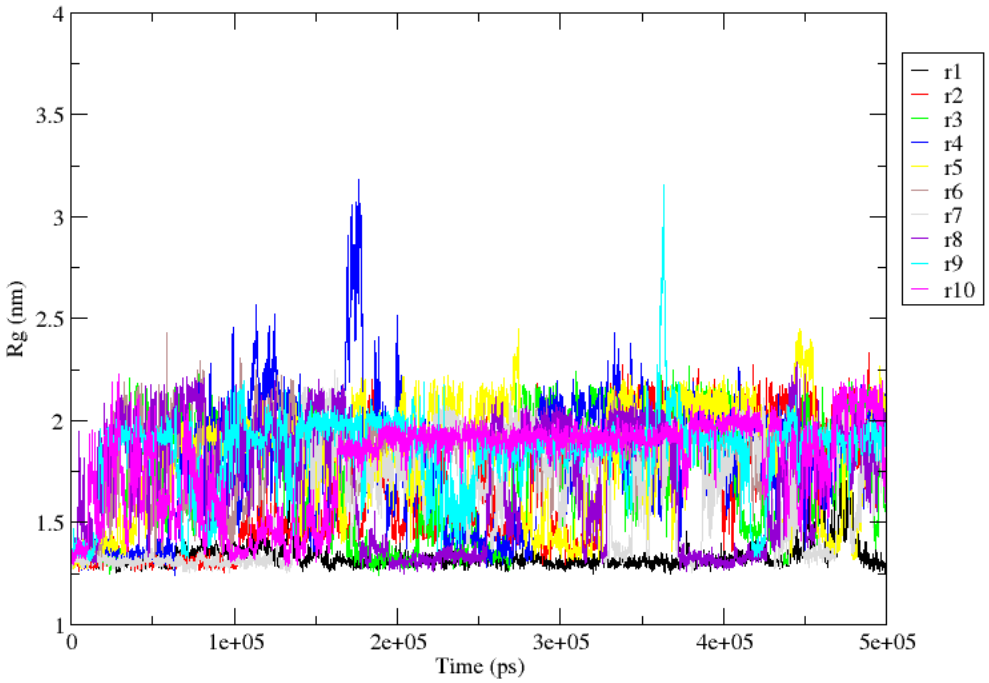

Figure 42. Rg for the simulations at 450 K using the CHARMM27 force field.

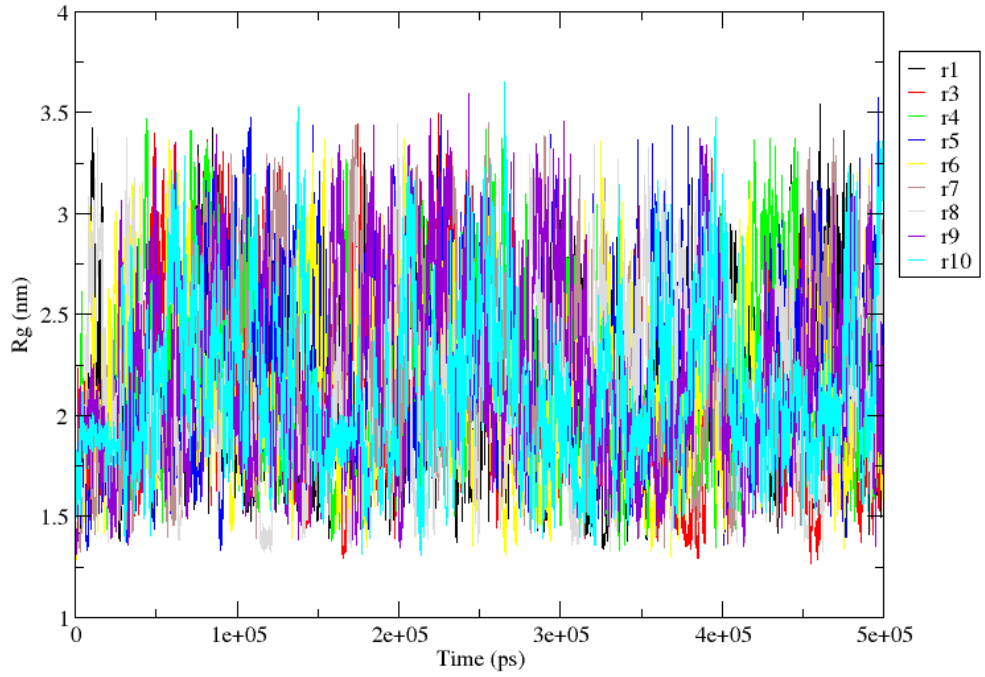

Figure 43. Rg for the simulations at 500 K using the CHARMM27 force field.

7.1.3 RMSF

The RMSF plots from the rMD simulations with CHARMM27 at 360 K, 380 K, 400 K, 420 K, 450 K and 500 K are shown below.

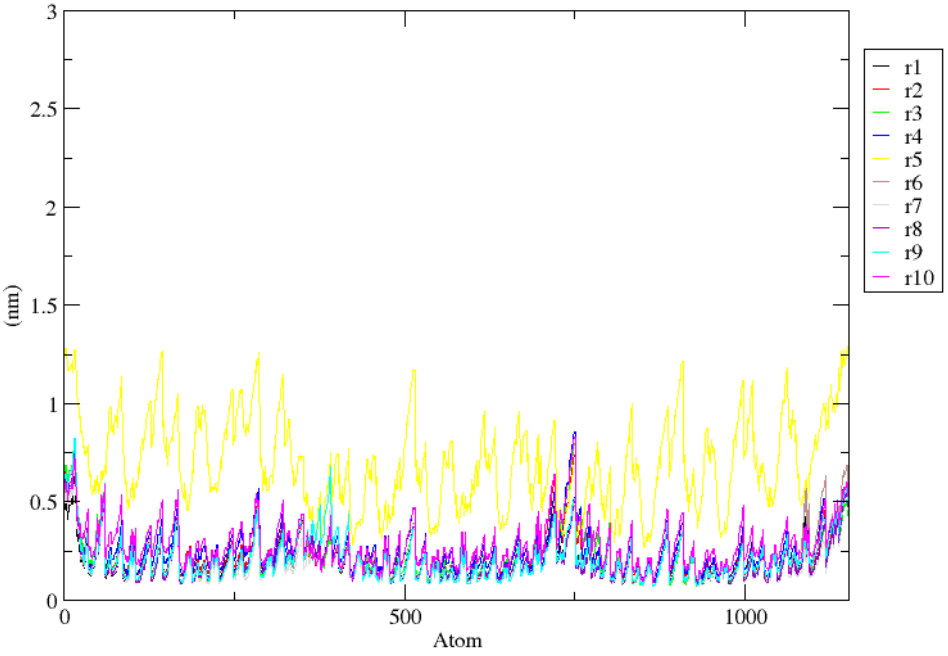

Figure 44. RMSF for the simulations at 360 K using the CHARMM27 force field.

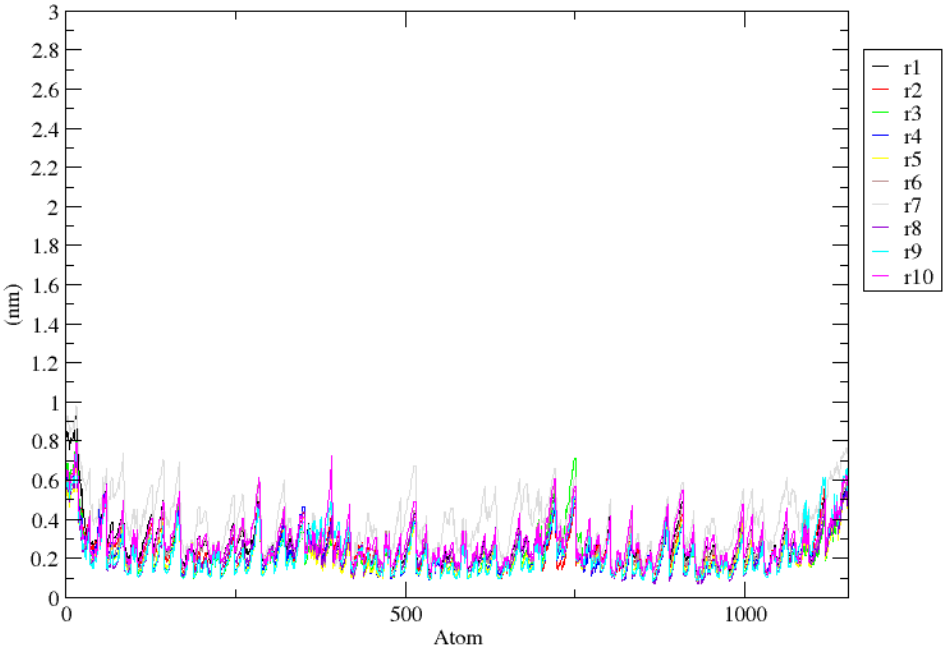

Figure 45. RMSF for the simulations at 380 K using the CHARMM27 force field.

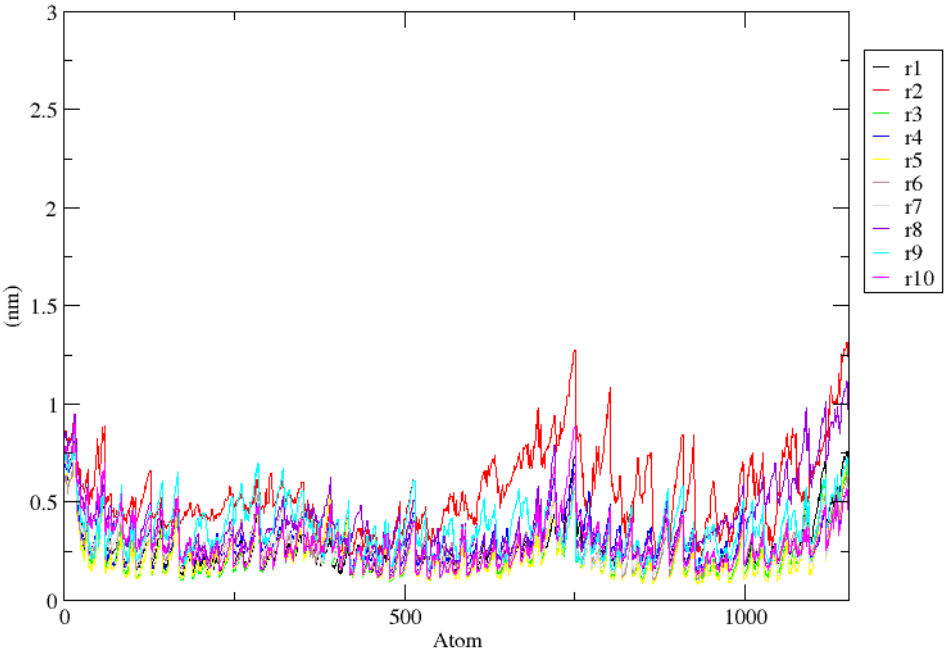

Figure 46. RMSF for the simulations at 400 K using the CHARMM27 force field.

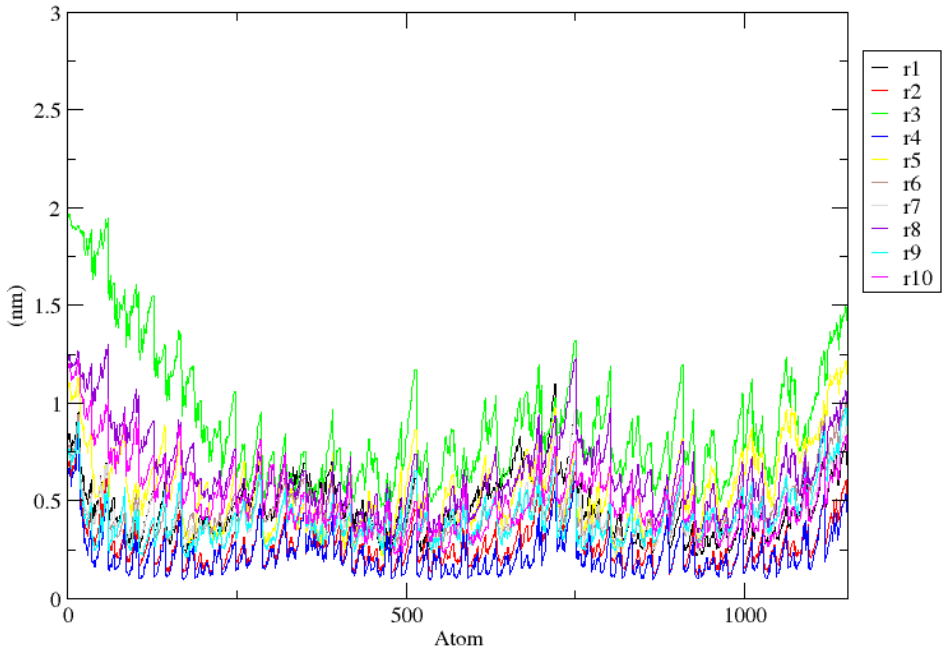

Figure 47. RMSF for the simulations at 420 K using the CHARMM27 force field.

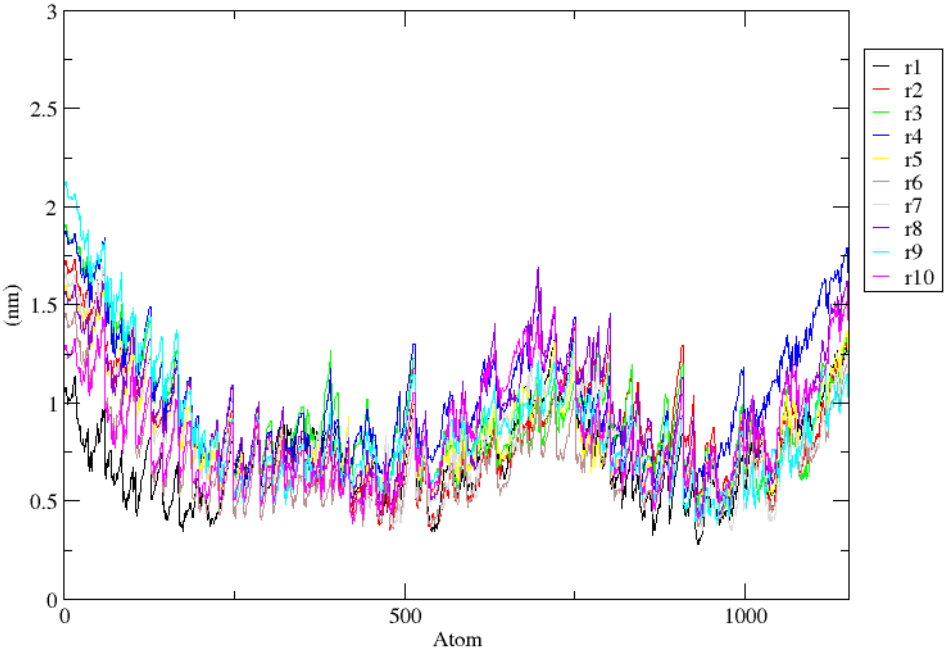

Figure 48. RMSF for the simulations at 450 K using the CHARMM27 force field.

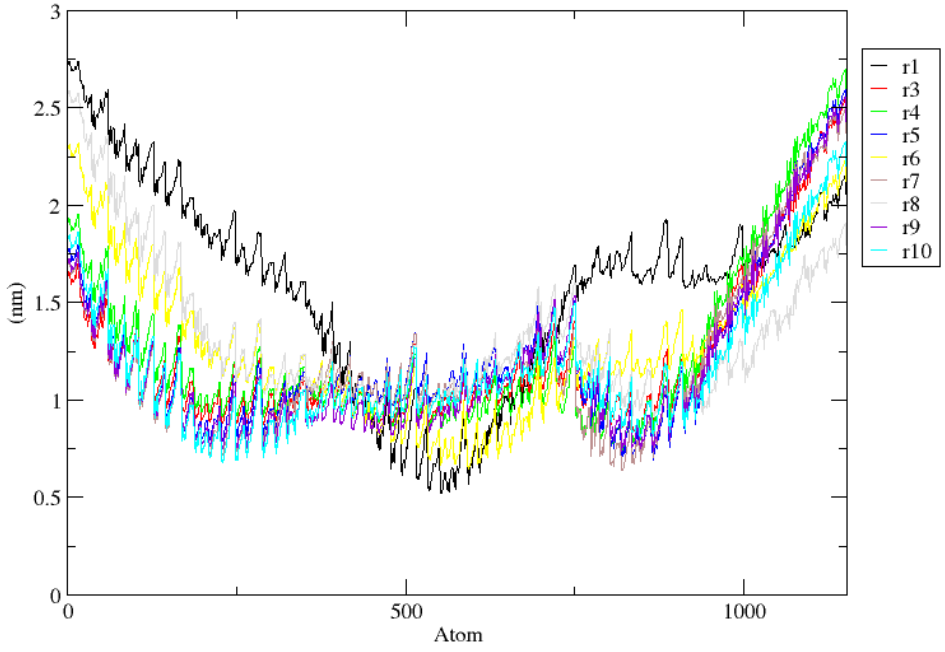

Figure 49. RMSF for the simulations at 500 K using the CHARMM27 force field.

7.1.4 H-bonds

The intra protein H-bonds plots from the rMD simulations with CHARMM27 at 360 K, 380 K, 400 K, 420 K, 450 K and 500 K are shown below.

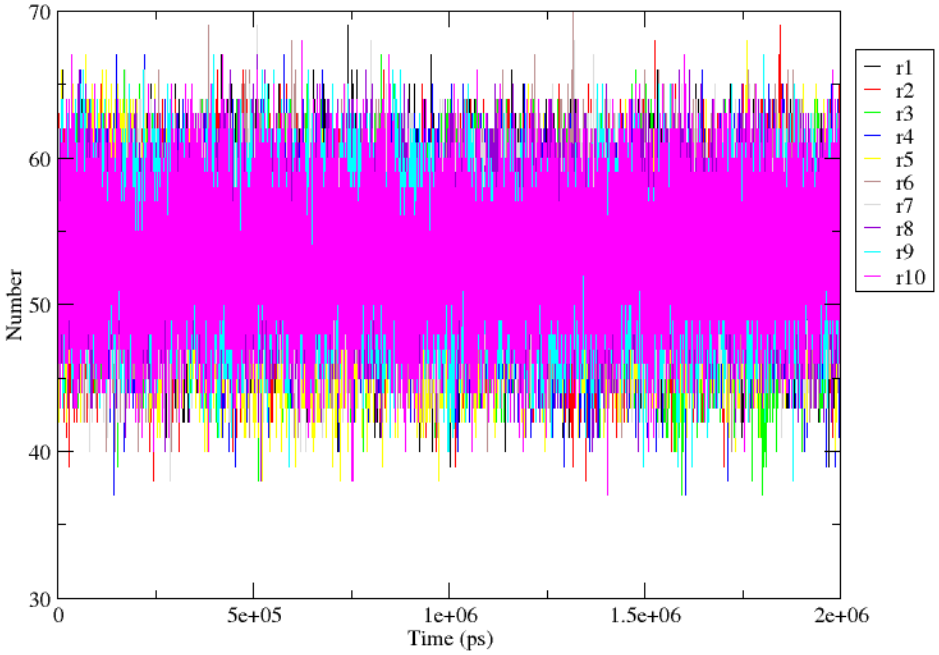

Figure 50. Intra protein H-bonds for the simulations at 360 K using the CHARMM27 force field.

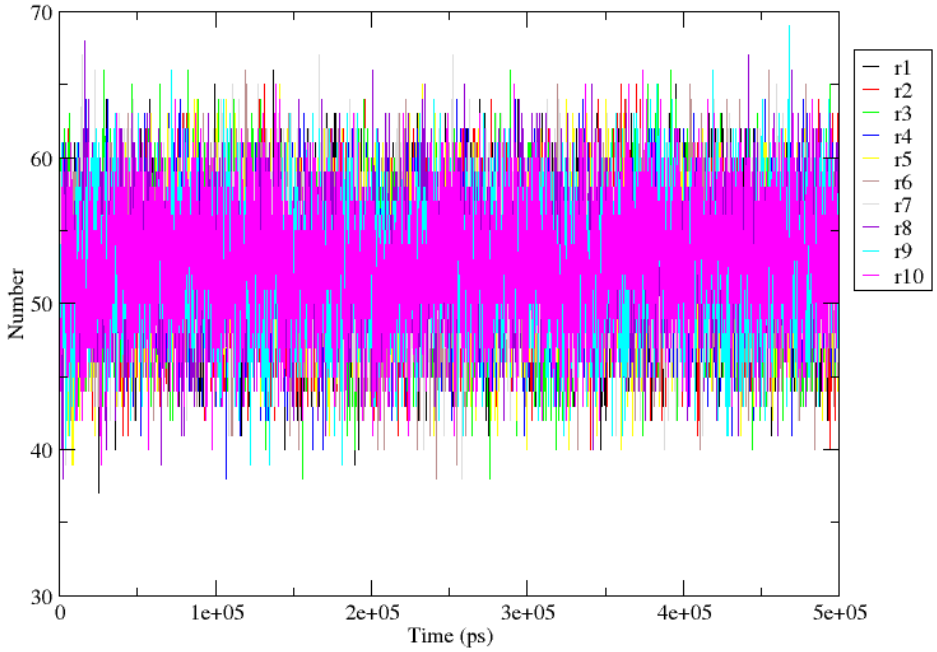

Figure 51. Intra protein H-bonds for the simulations at 380 K using the CHARMM27 force field.

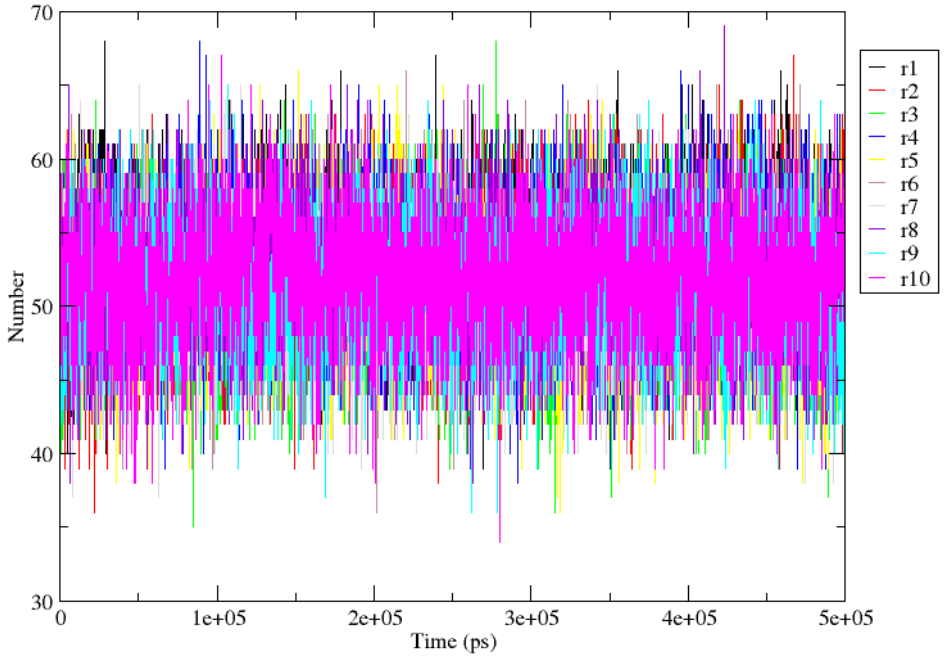

Figure 52. Intra protein H-bonds for the simulations at 400 K using the CHARMM27 force field.

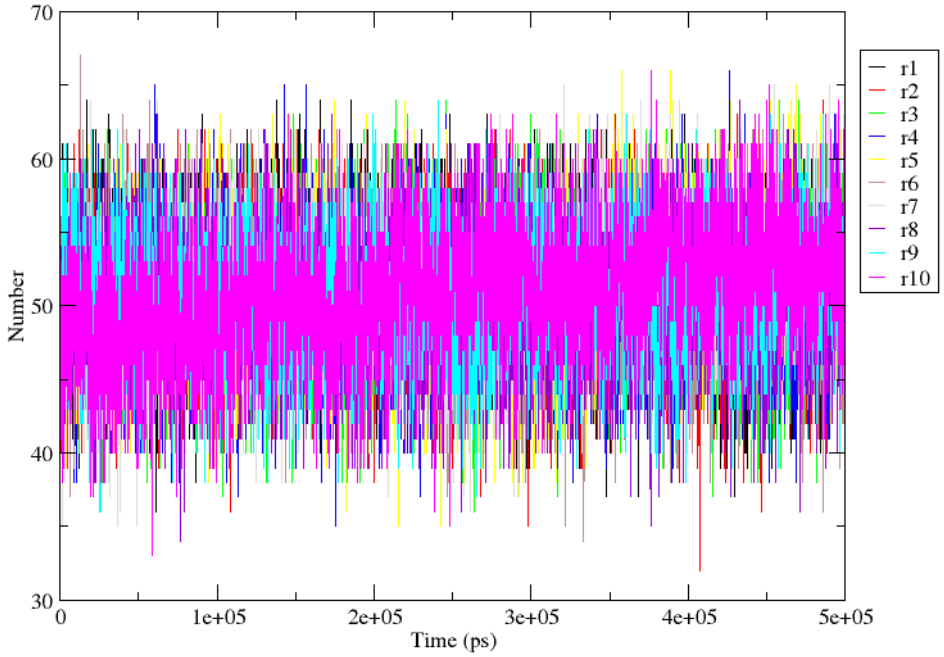

Figure 53. Intra protein H-bonds for the simulations at 420 K using the CHARMM27 force field.

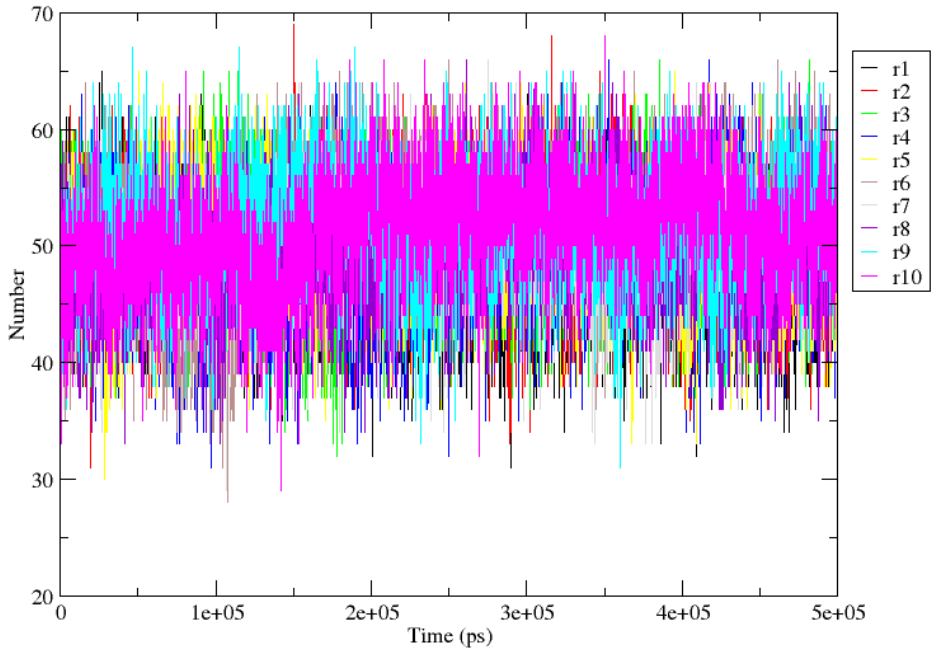

Figure 54. Intra protein H-bonds for the simulations at 450 K using the CHARMM27 force field.

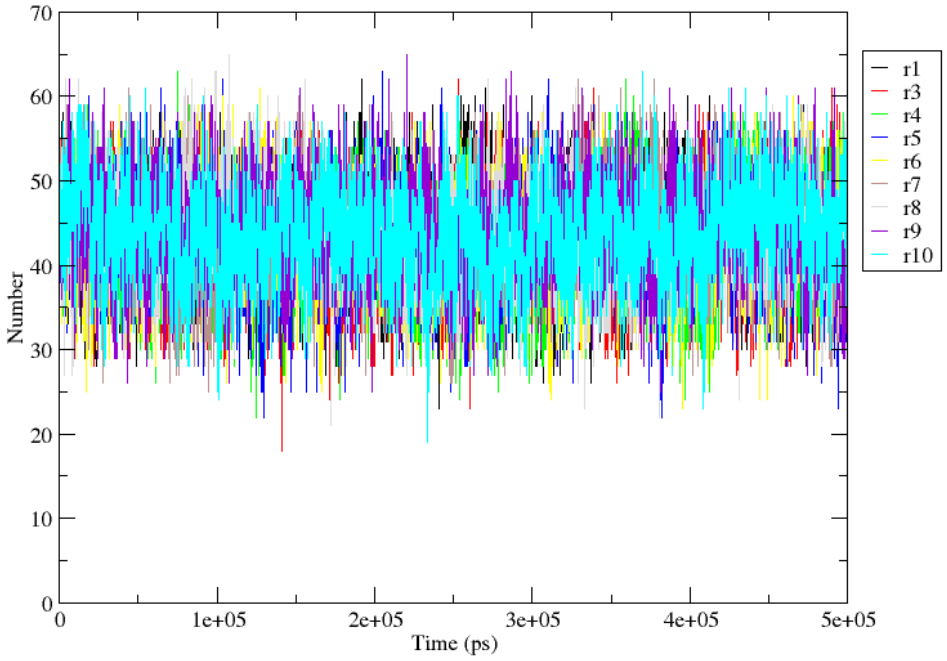

Figure 55. Intra protein H-bonds for the simulations at 500 K using the CHARMM27 force field.

7.1.5 SASA

The SASA plots from the rMD simulations with CHARMM27 at 360 K, 380 K, 400 K, 420 K, 450 K and 500 K are shown below.

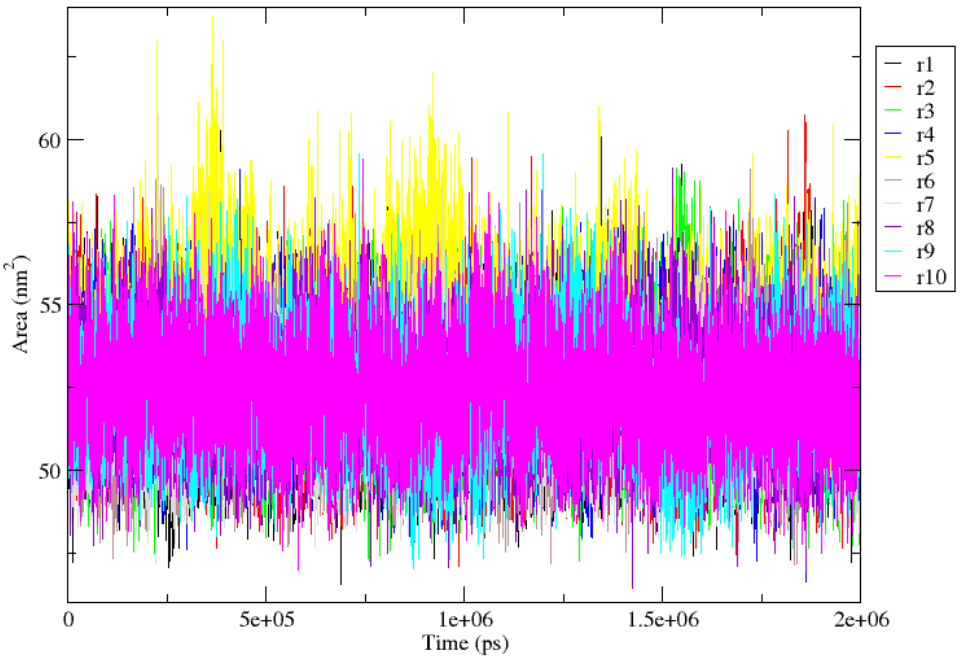

Figure 56. SASA for the simulations at 360 K using the CHARMM27 force field.

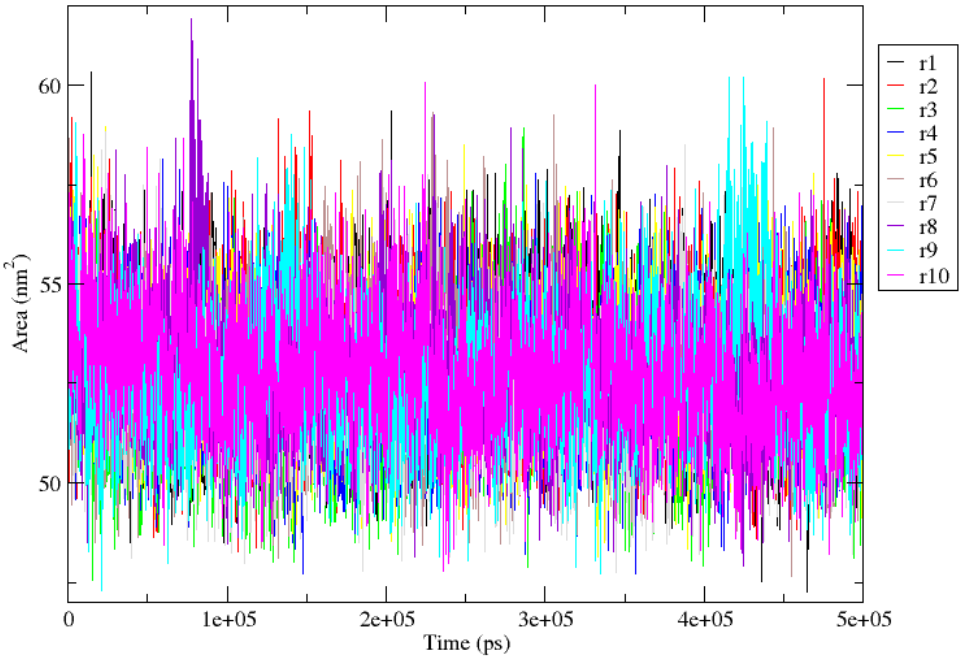

Figure 57. SASA for the simulations at 380 K using the CHARMM27 force field.

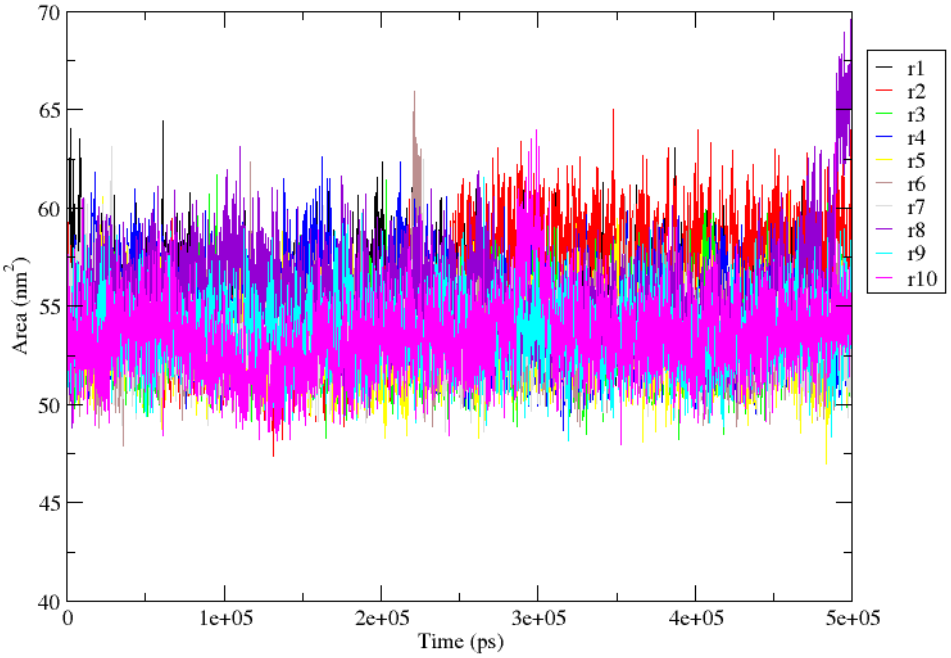

Figure 58. SASA for the simulations at 400 K using the CHARMM27 force field

.

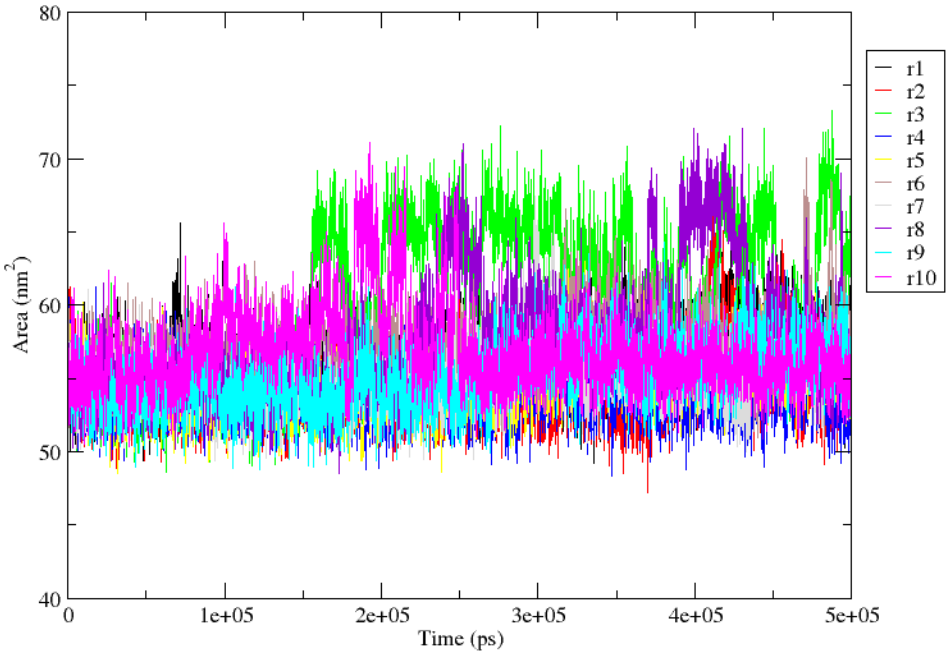

Figure 59. SASA for the simulations at 420 K using the CHARMM27 force field.

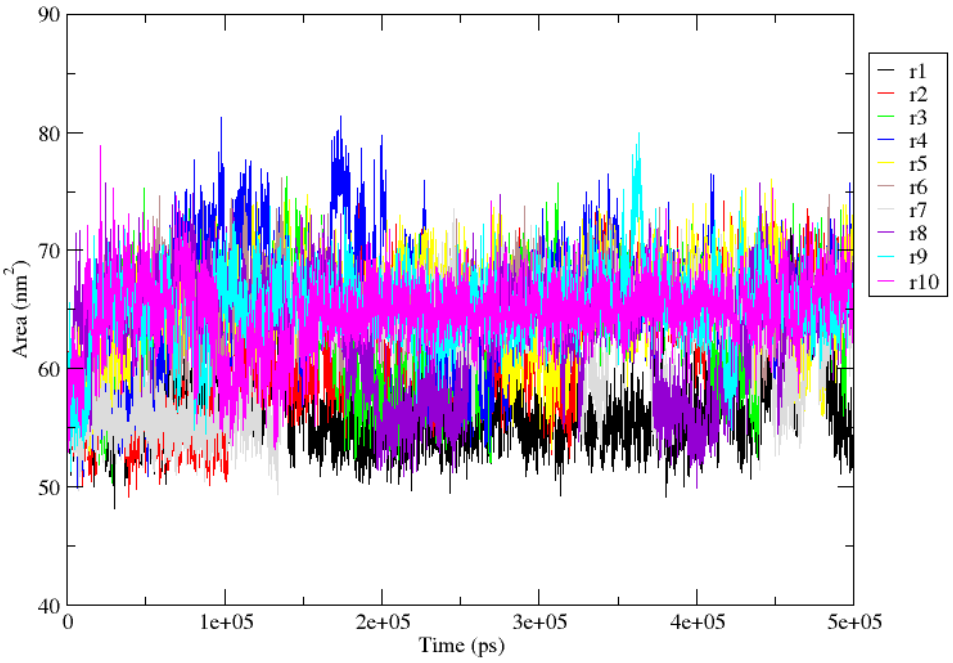

Figure 60. SASA for the simulations at 450 K using the CHARMM27 force field.

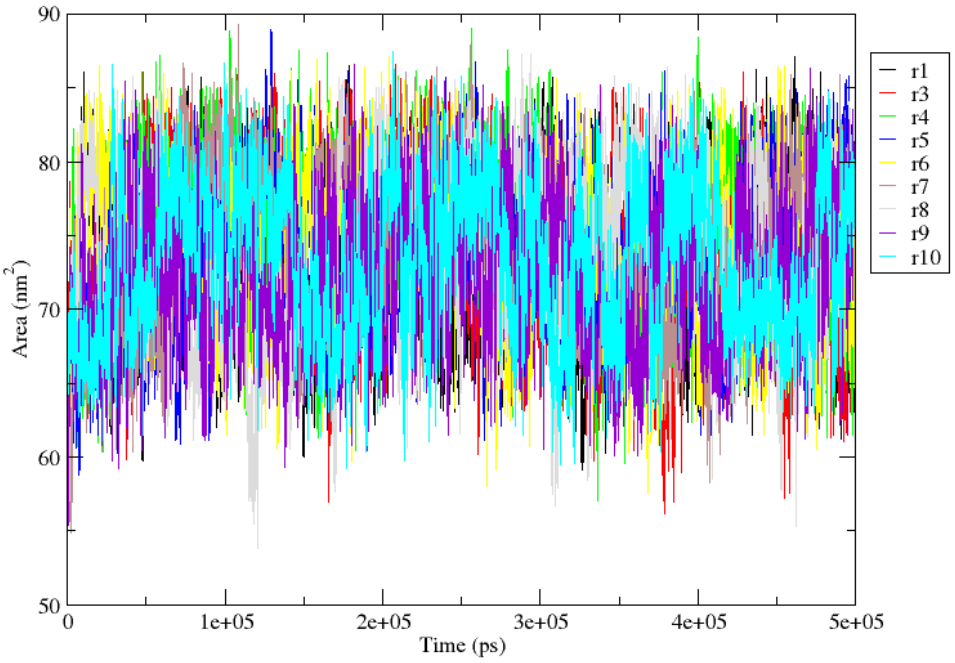

Figure 61. SASA for the simulations at 500 K using the CHARMM27 force field.

7.1.6 2D-RMSD-based clustering

The 2D-RMSD-based clustering plots from the rMD simulations with CHARMM27 at 380 K, 400 K, 420 K, 450 K and 500 K are shown below.

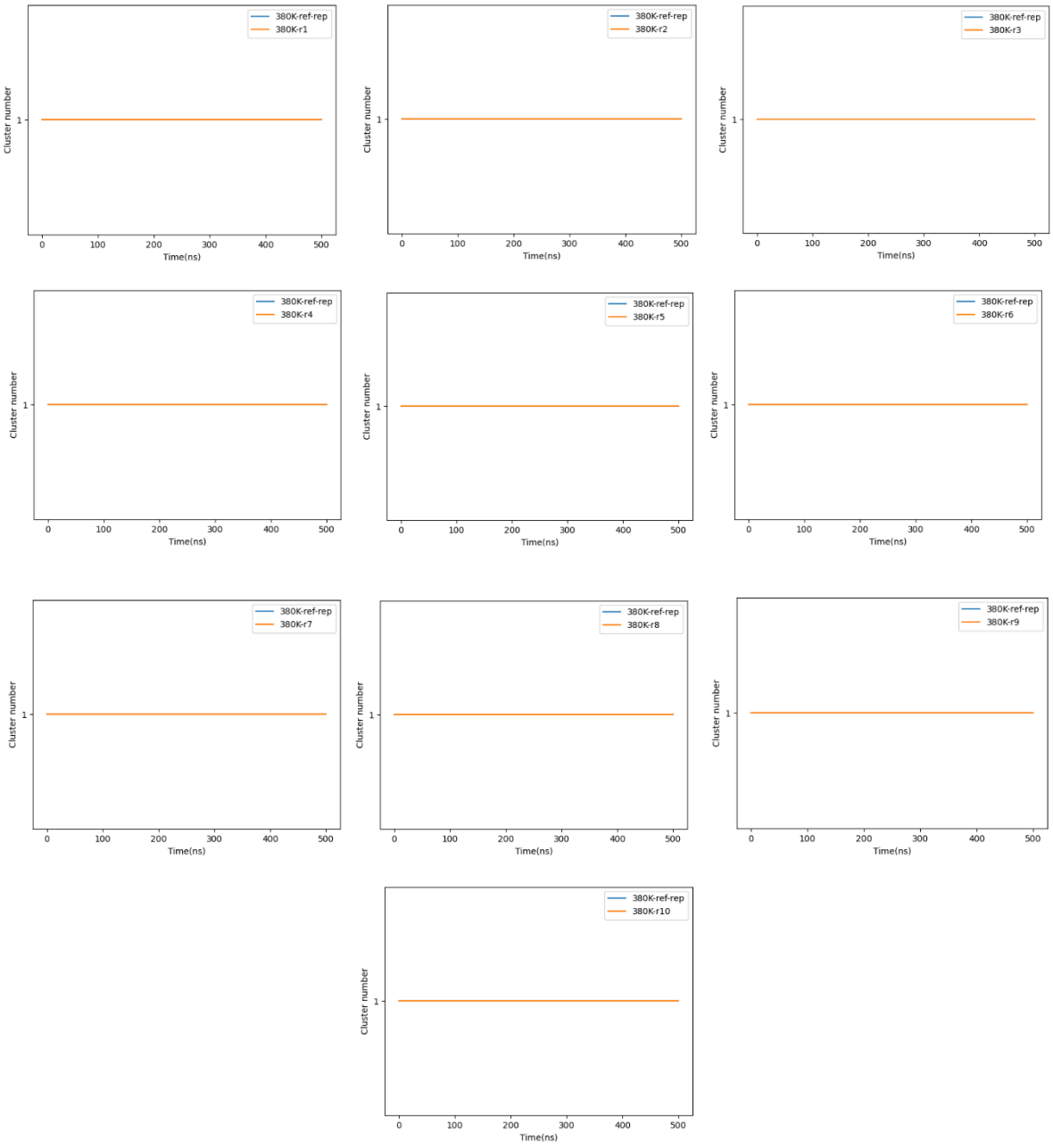

Figure 62. 2D-RMSD-based clustering for the ten replicas at 380 K using the CHARMM27 force field.

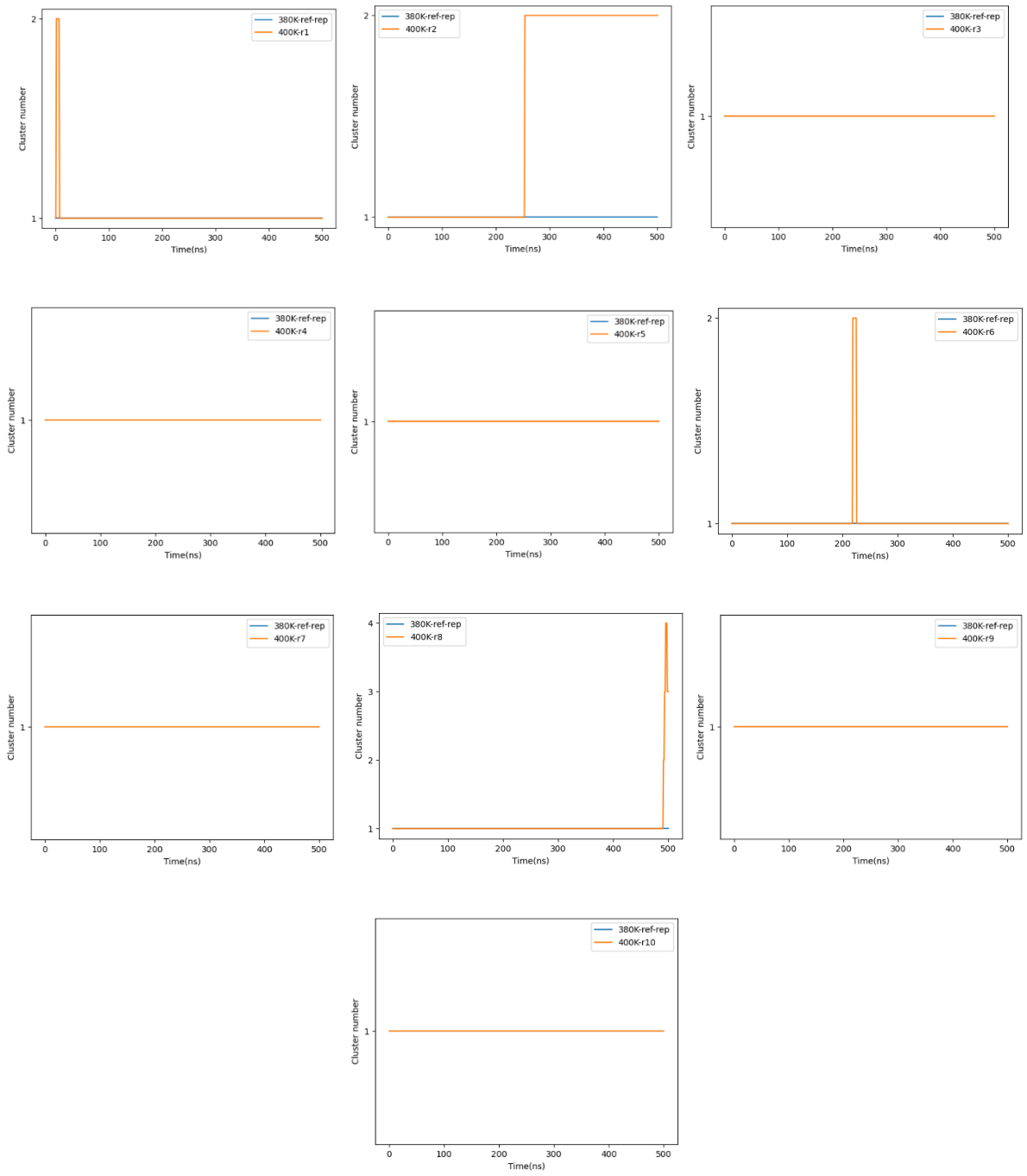

Figure 63. 2D-RMSD-based clustering for the ten replicas at 400 K using the CHARMM27 force field.

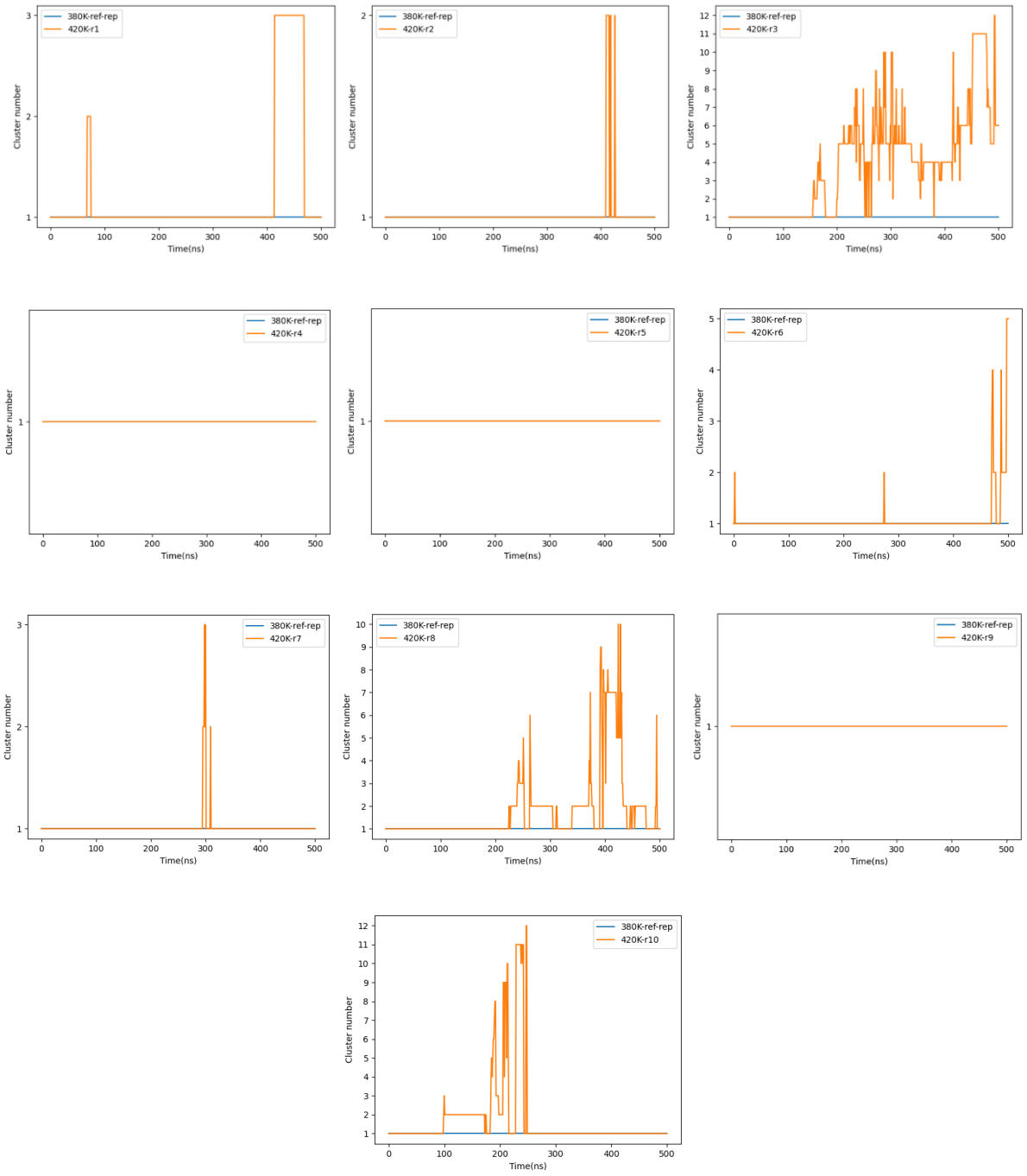

Figure 64. 2D-RMSD-based clustering for the ten replicas at 420 K using the CHARMM27 force field.

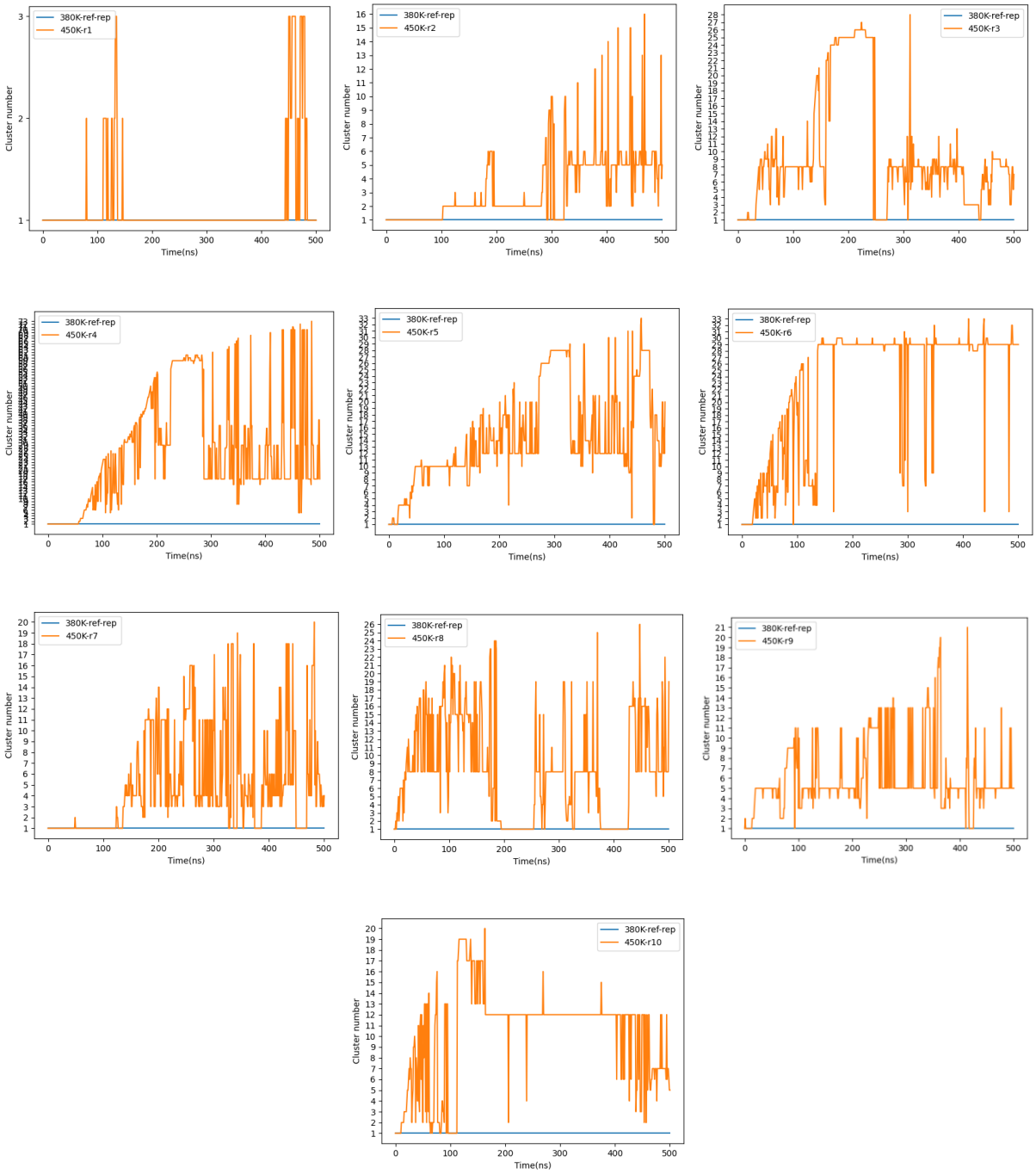

Figure 65. 2D-RMSD-based clustering for the ten replicas at 450 K using the CHARMM27 force field.

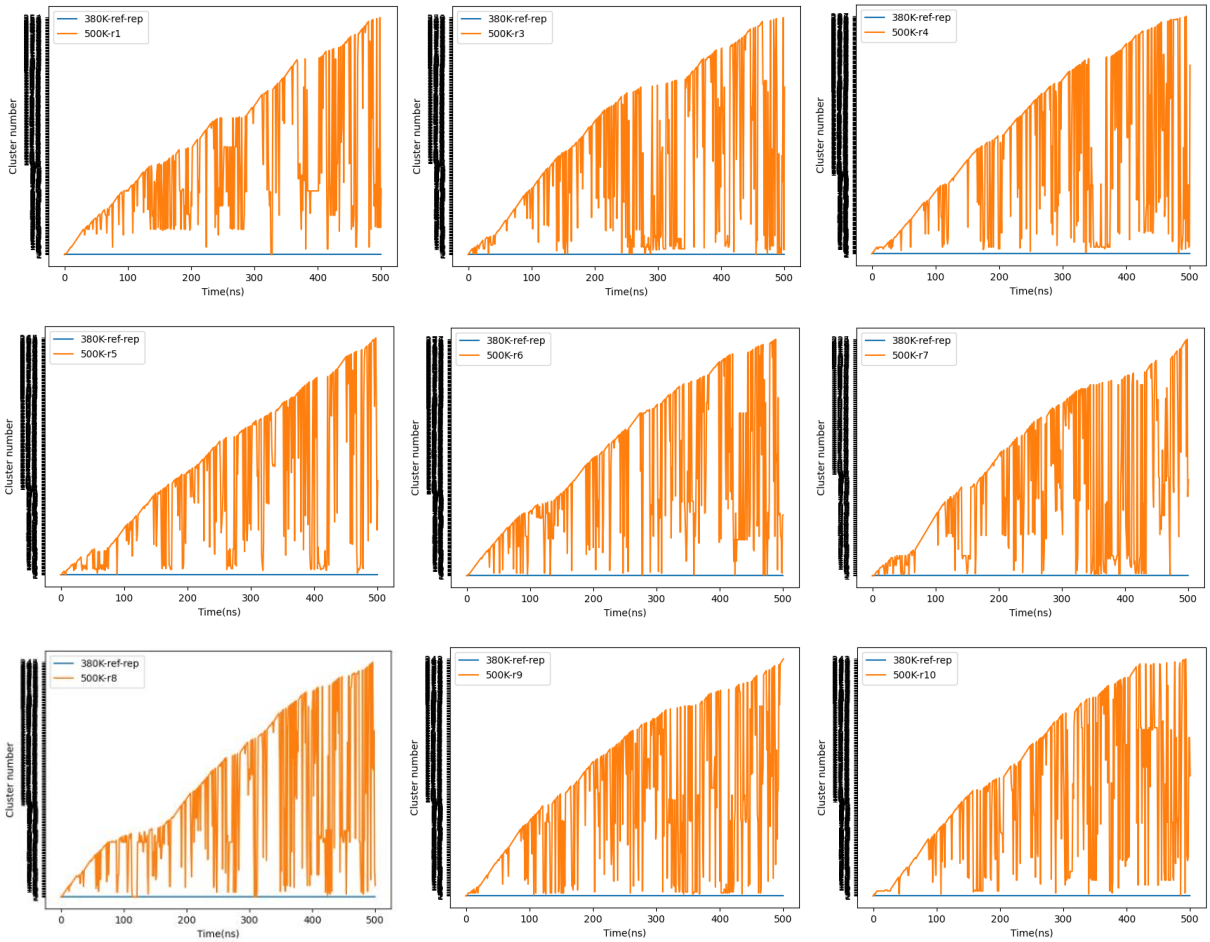

Figure 66. 2D-RMSD-based clustering for the nine replicas at 500 K using the CHARMM27 force field.

7.2 rMD simulations with AMBER99-disp

7.2.1 RMSD

The RMSD plots from the rMD simulations with AMBER99-disp at 360 K, 380 K, 450 K and 500 K are shown below.

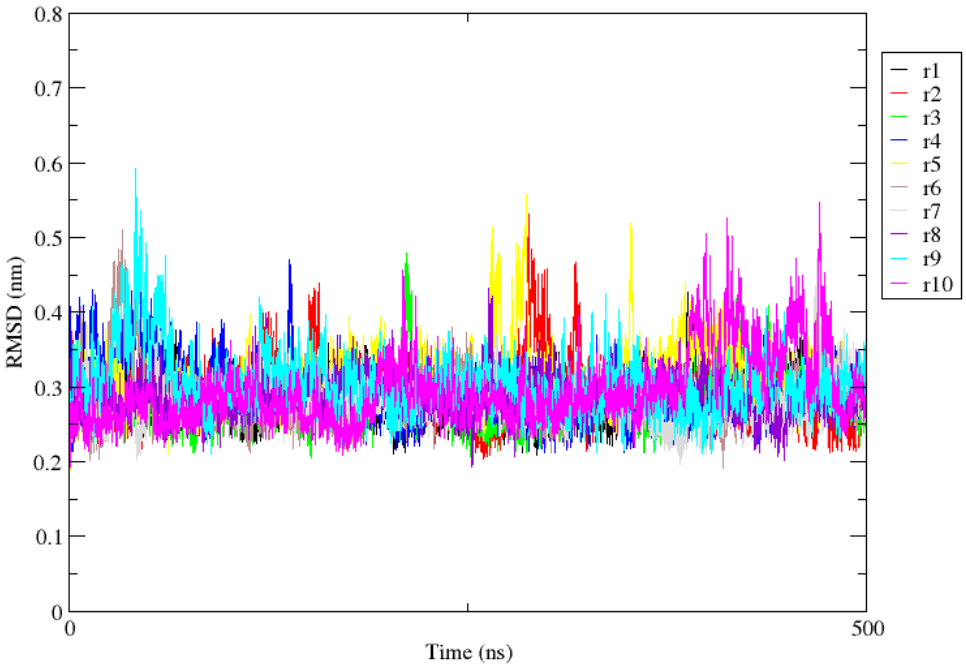

Figure 67. RMSD for the simulations at 360 K using the AMBER99SB-disp force field.

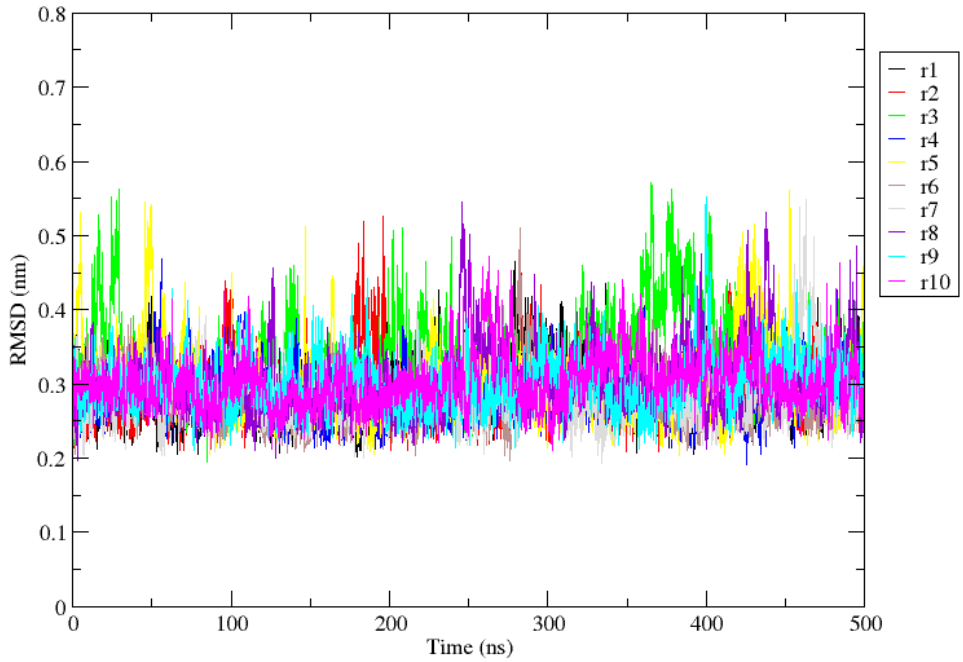

Figure 68. RMSD for the simulations at 380 K using the AMBER99SB-disp force field.

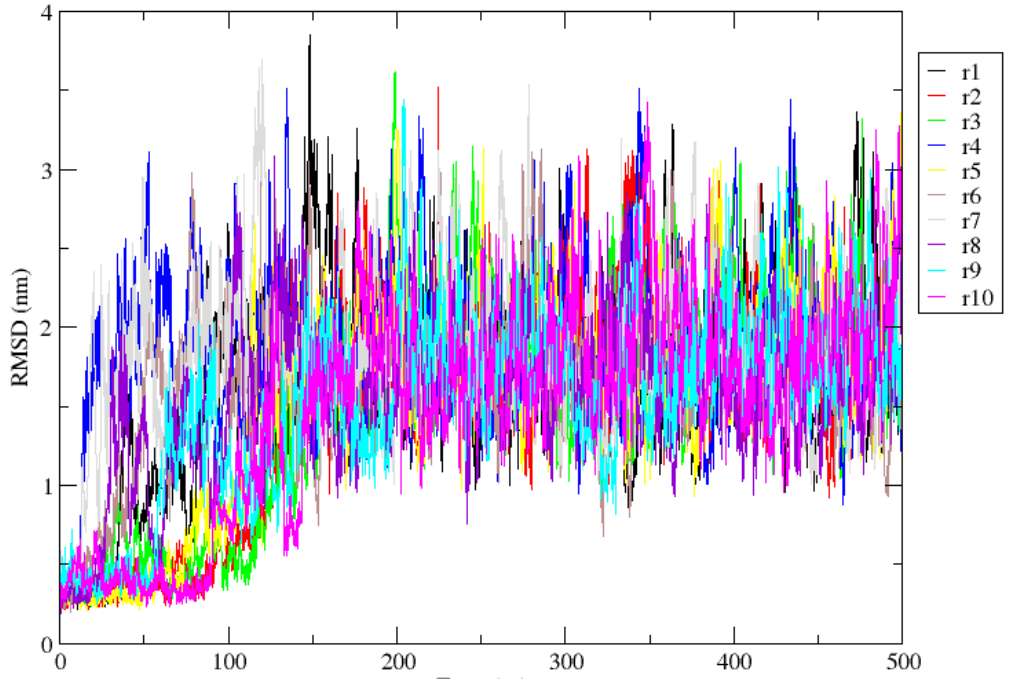

Figure 69. RMSD for the simulations at 450 K using the AMBER99SB-disp force field.

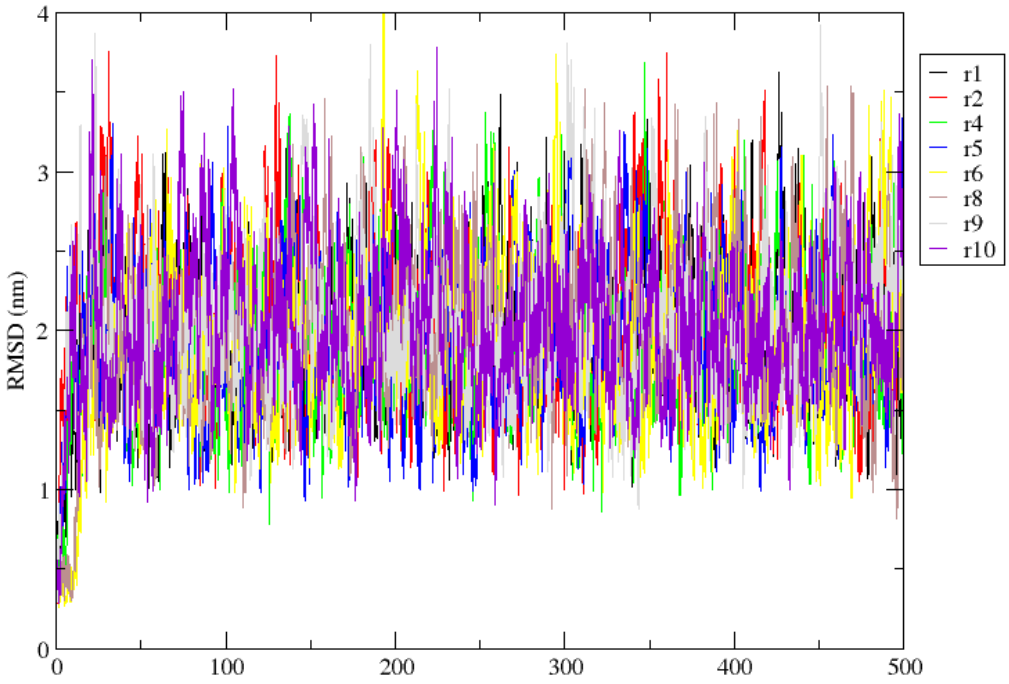

Figure 70. RMSD for the simulations at 500 K using the AMBER99SB-disp force field.

7.2.2 Rg

The Rg plots from the rMD simulations with AMBER99-disp at 360 K, 380 K, 450 K and 500 K are shown below.

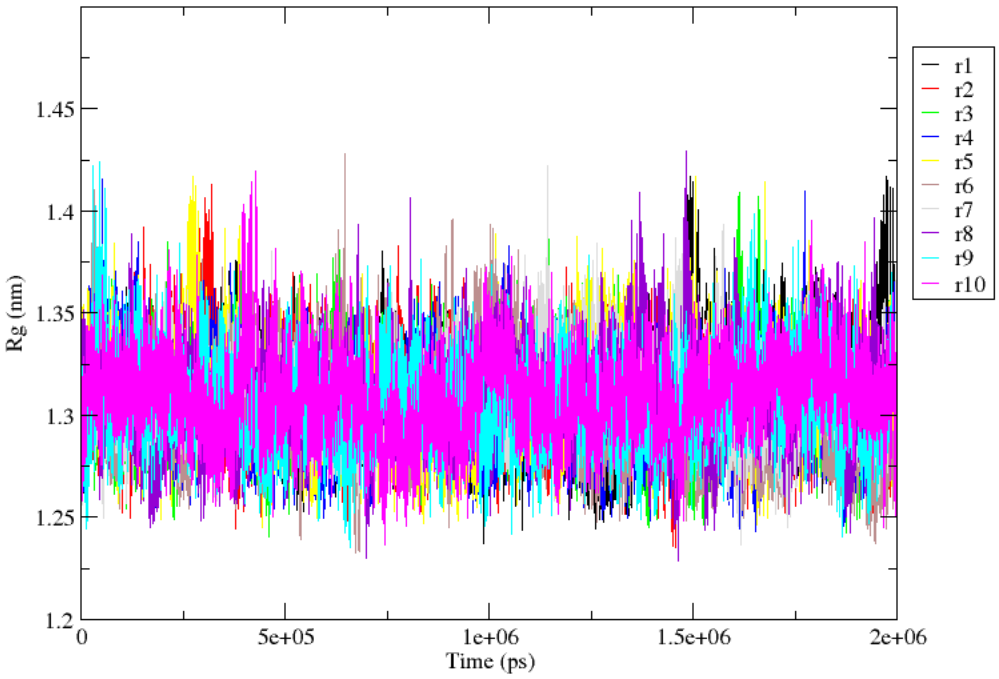

Figure 71. Rg for the simulations at 360 K using the AMBER99SB-disp force field.

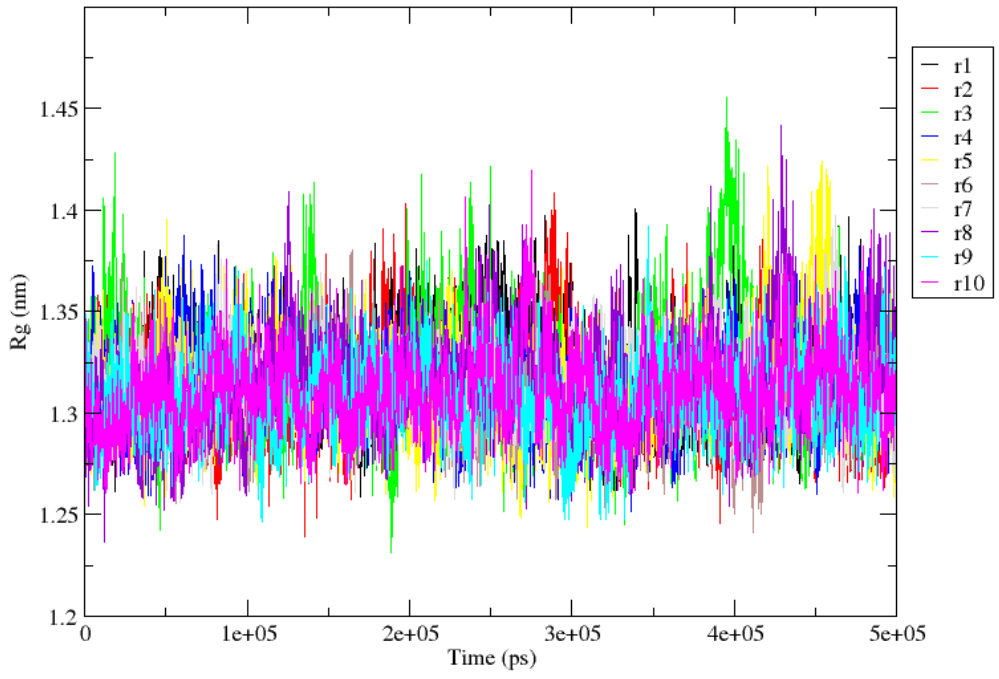

Figure 72. Rg for the simulations at 380 K using the AMBER99SB-disp force field.

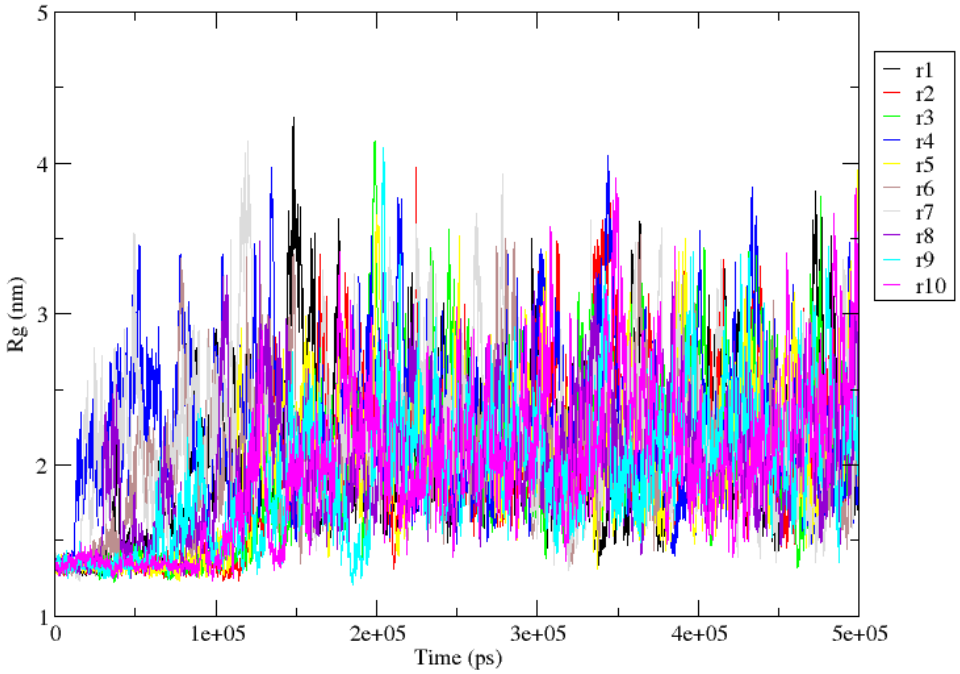

Figure 73. Rg for the simulations at 450 K using the AMBER99SB-disp force field.

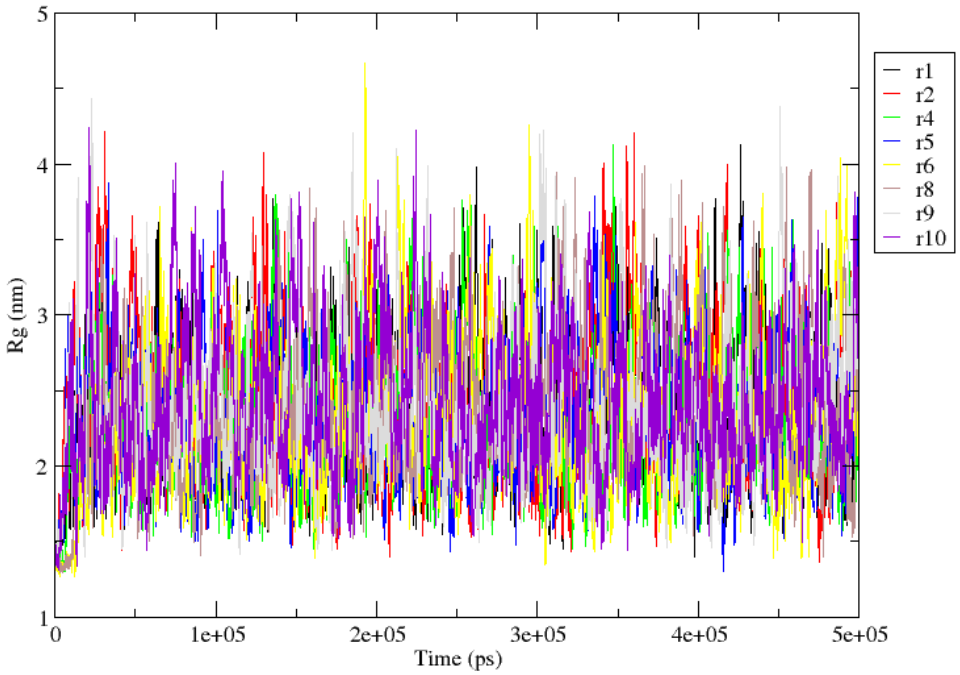

Figure 74. Rg for the simulations at 500 K using the AMBER99SB-disp force field.

7.2.3 RMSF

The RMSF plots from the rMD simulations with AMBER99-disp at 360 K, 380 K, 450 K and 500 K are shown below.

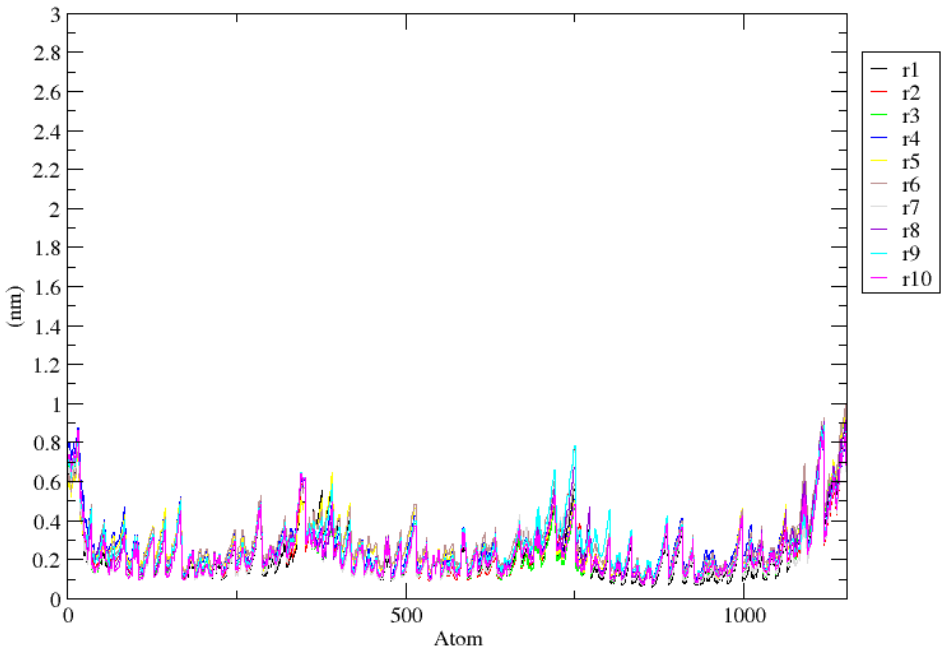

Figure 75. RMSF for the simulations at 360 K using the AMBER99SB-disp force field.

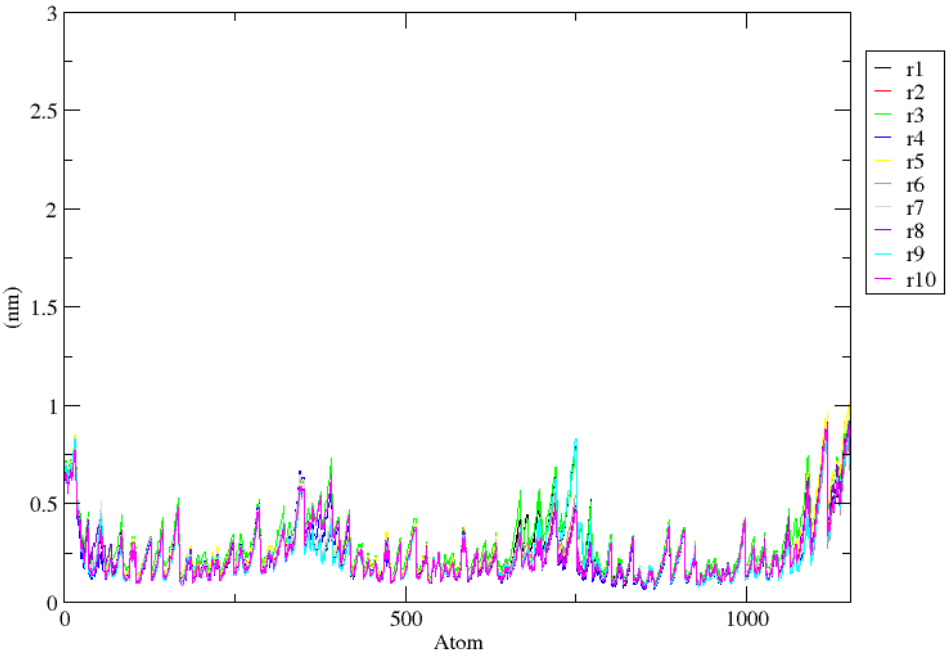

Figure 76. RMSF for the simulations at 380 K using the AMBER99SB-disp force field.

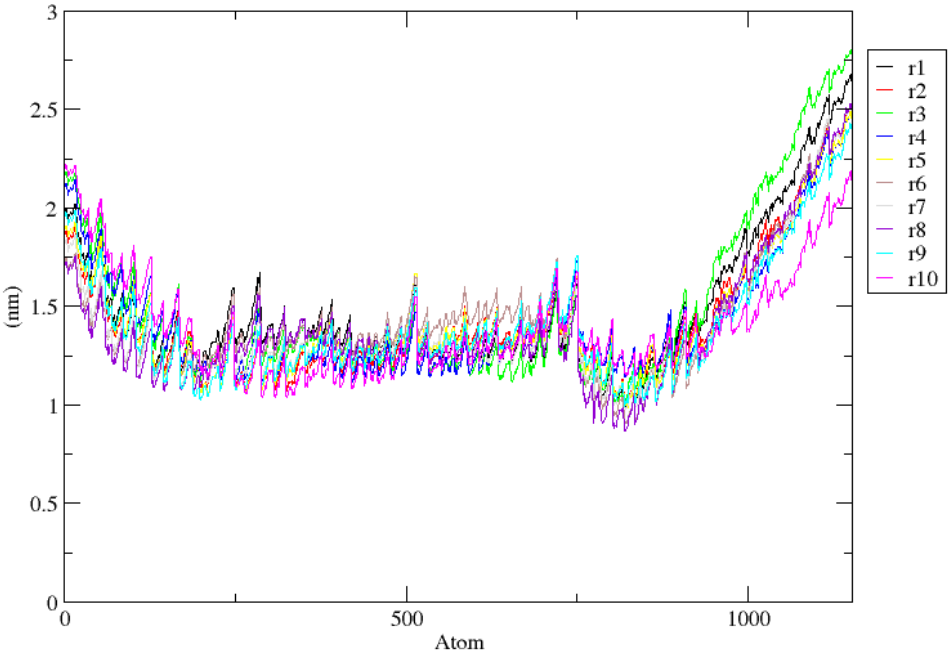

Figure 77. RMSF for the simulations at 450 K using the AMBER99SB-disp force field.

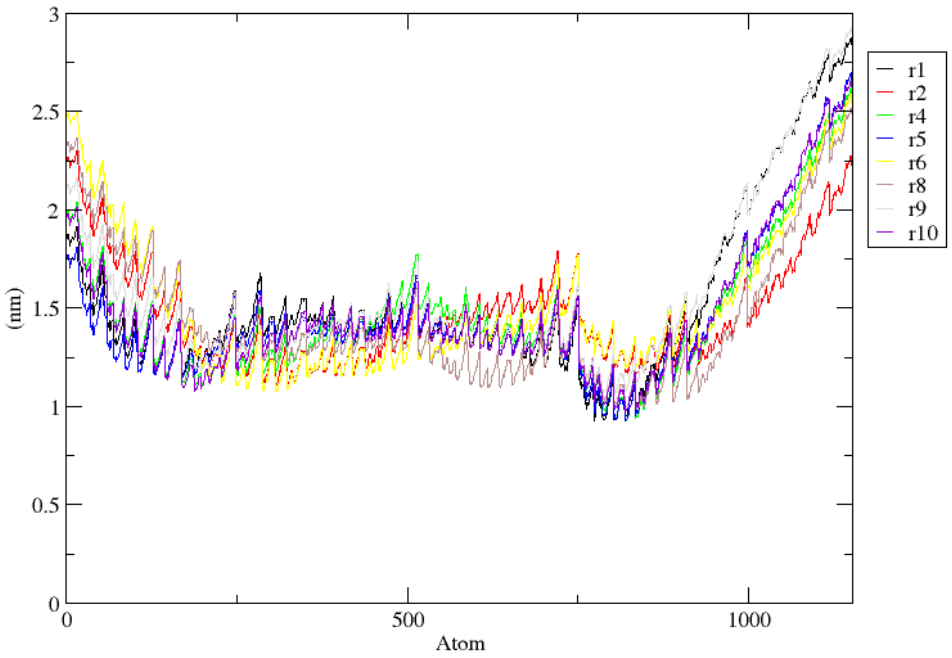

Figure 78. RMSF for the simulations at 500 K using the AMBER99SB-disp force field.

7.2.4 H-bonds

The intra protein H-bonds plots from the rMD simulations with AMBER99-disp at 360 K, 380 K, 450 K and 500 K are shown below.

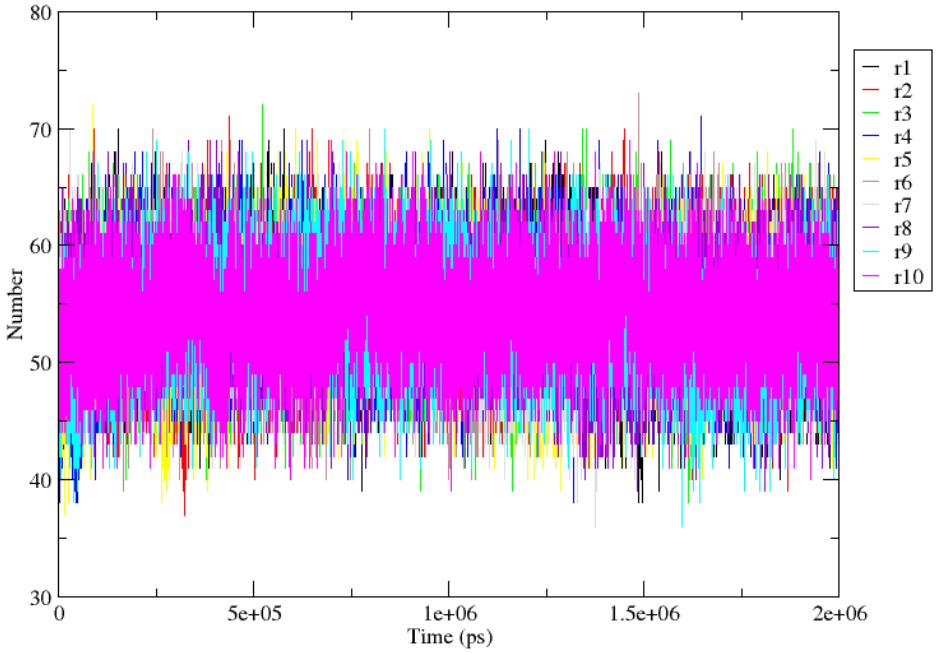

Figure 79. Intra protein H-bonds for the simulations at 360 K using the AMBER99SB-disp force field.

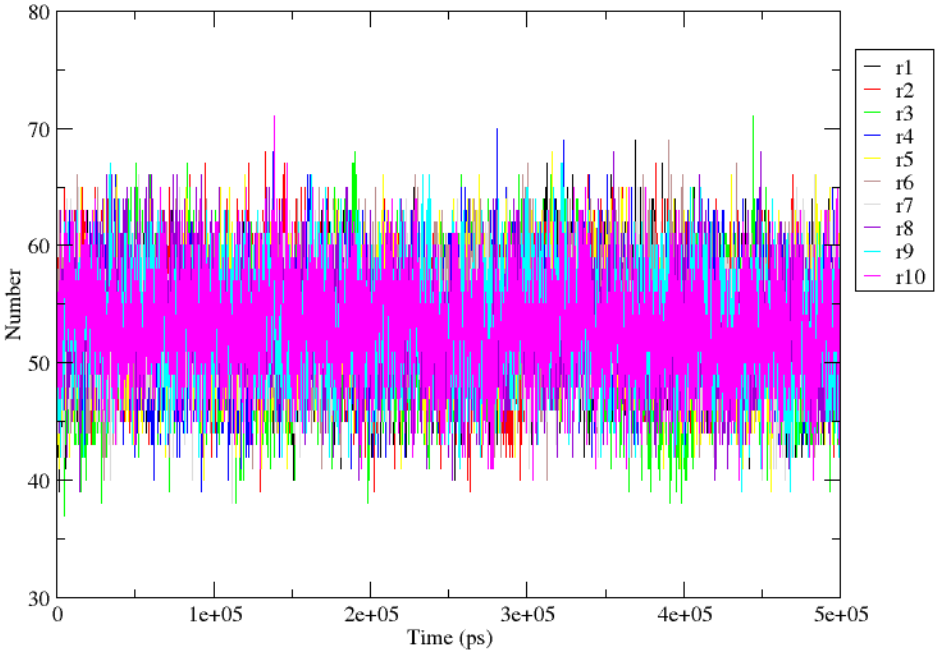

Figure 80. Intra protein H-bonds for the simulations at 380 K using the AMBER99SB-disp force field.

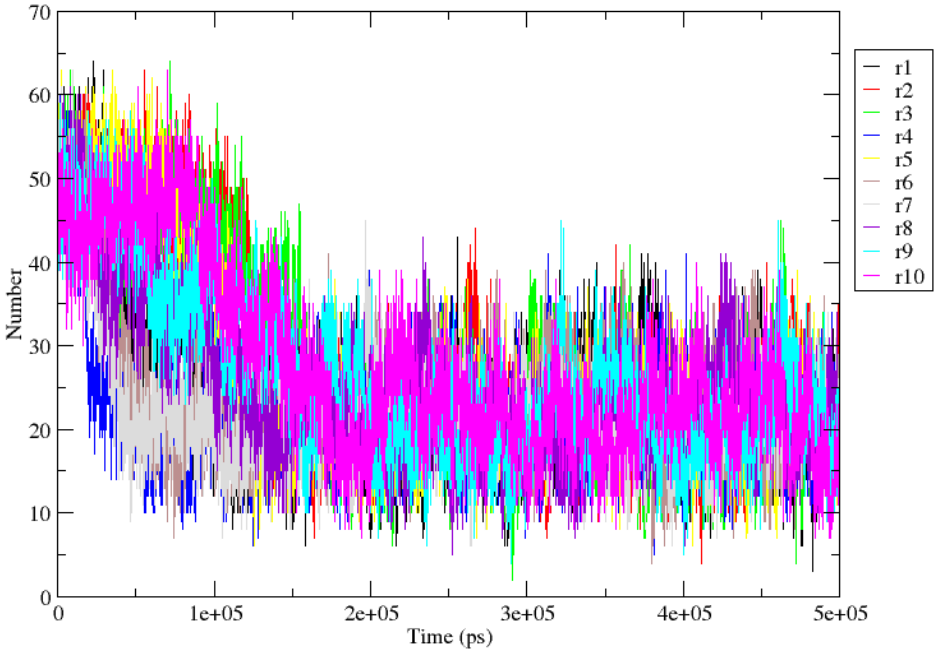

Figure 81. Intra protein H-bonds for the simulations at 450 K using the AMBER99SB-disp force field.

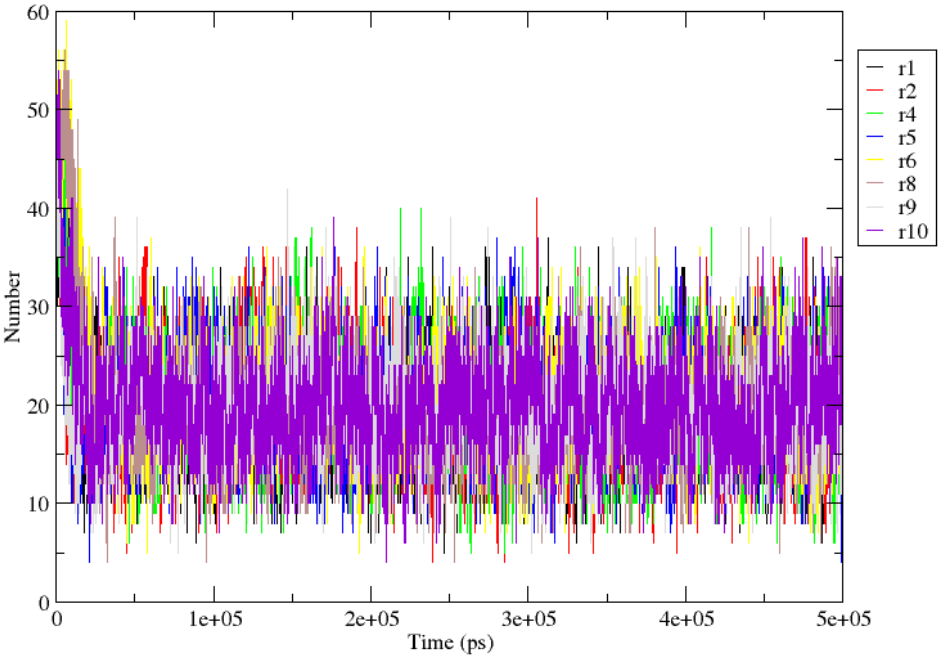

Figure 82. Intra protein H-bonds for the simulations at 500 K using the AMBER99SB-disp force field.

7.2.5 SASA

The SASA plots from the rMD simulations with AMBER99-disp at 360 K, 380 K, 450 K and 500 K are shown below.

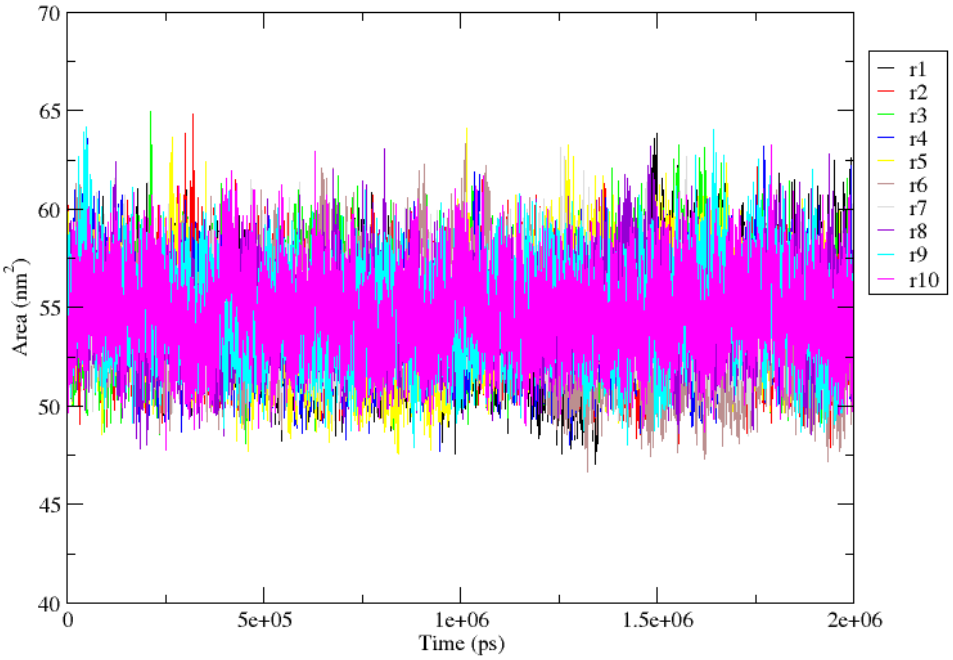

Figure 83. SASA for the simulations at 360 K using the AMBER99SB-disp force field.

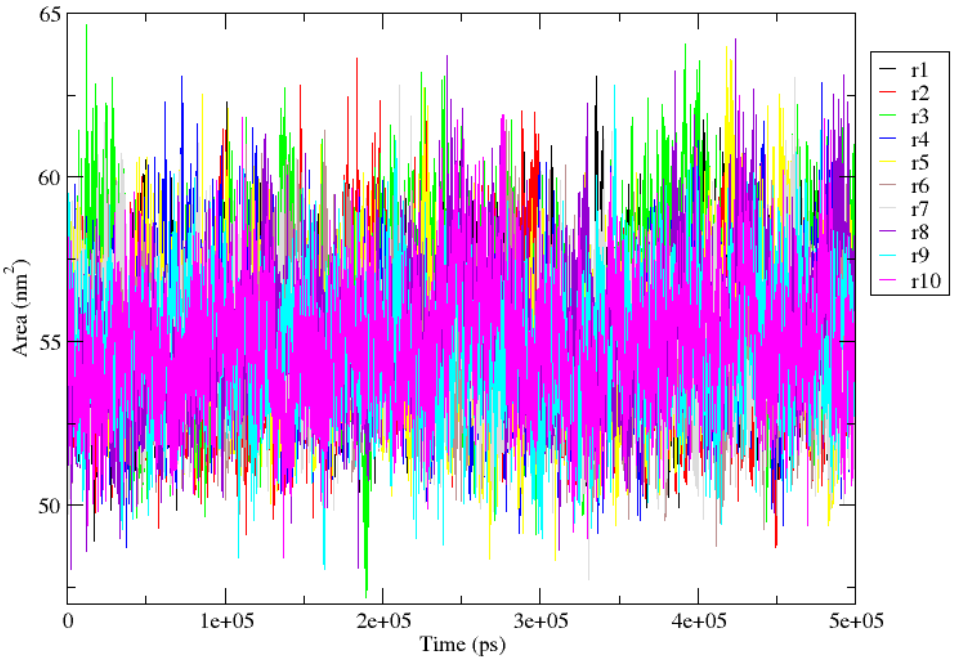

Figure 84. SASA for the simulations at 380 K using the AMBER99SB-disp force field.

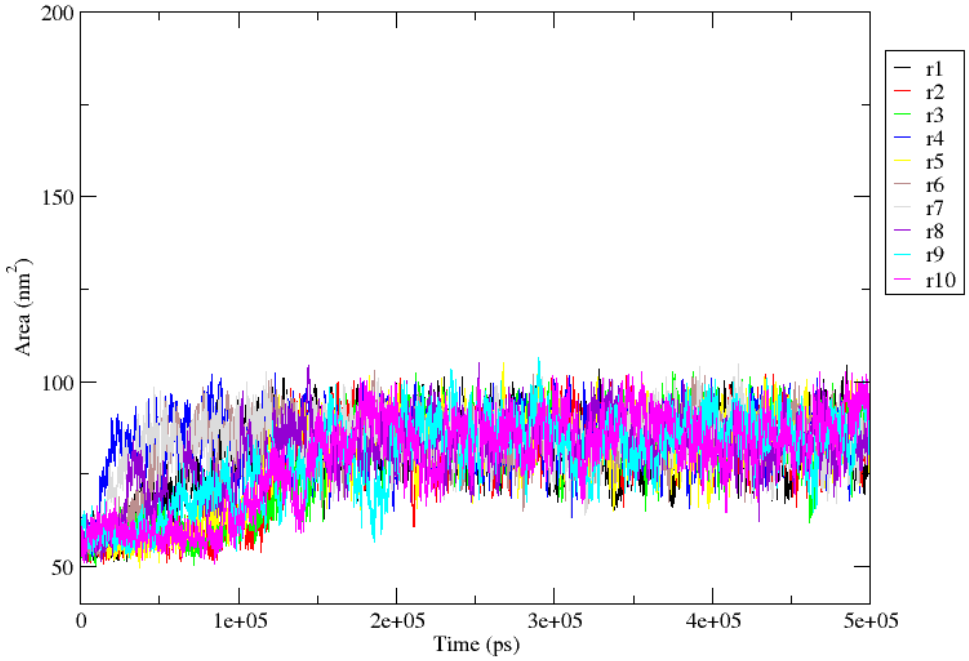

Figure 85. SASA for the simulations at 450 K using the AMBER99SB-disp force field.

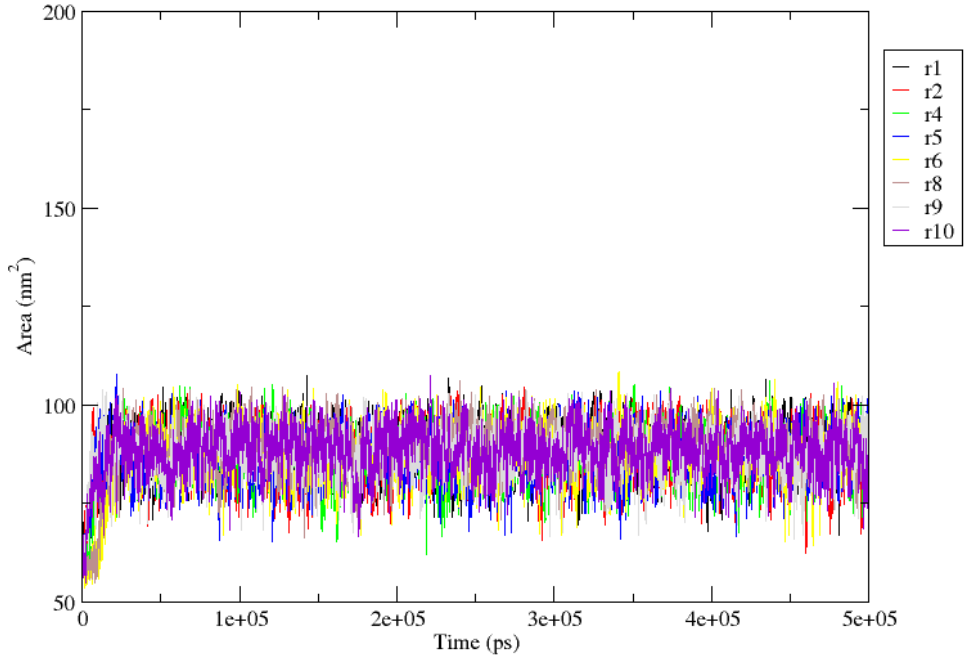

Figure 86. SASA for the simulations at 500 K using the AMBER99SB-disp force field.

7.2.6 2D-RMSD-based clustering

The 2D-RMSD-based clustering plots from the rMD simulations with AMBER99-disp at 380 K, 450 K and 500 K are shown below.

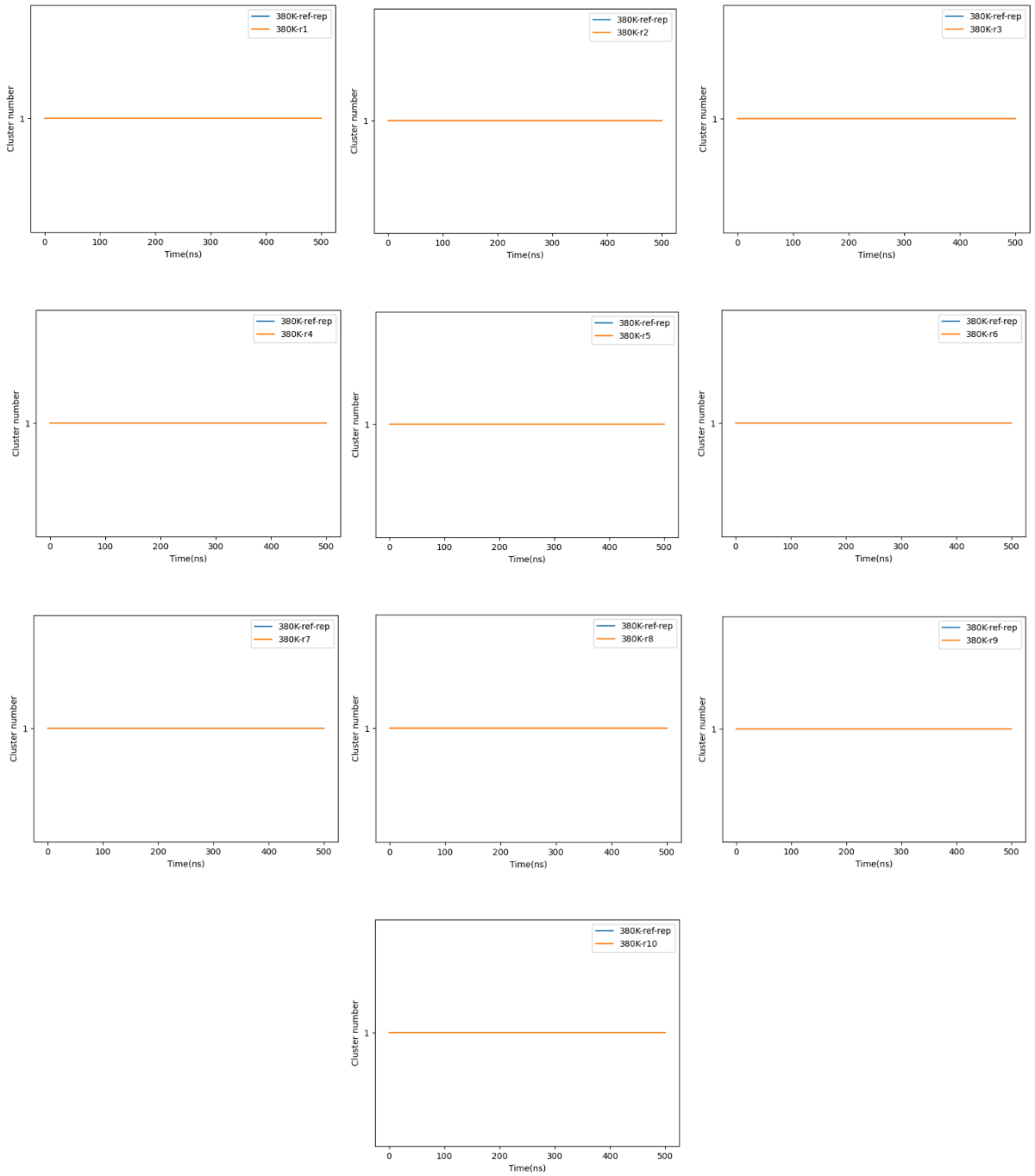

Figure 87. 2D-RMSD-based clustering for the ten replicas at 380 K using the AMBER99SB-disp force field.

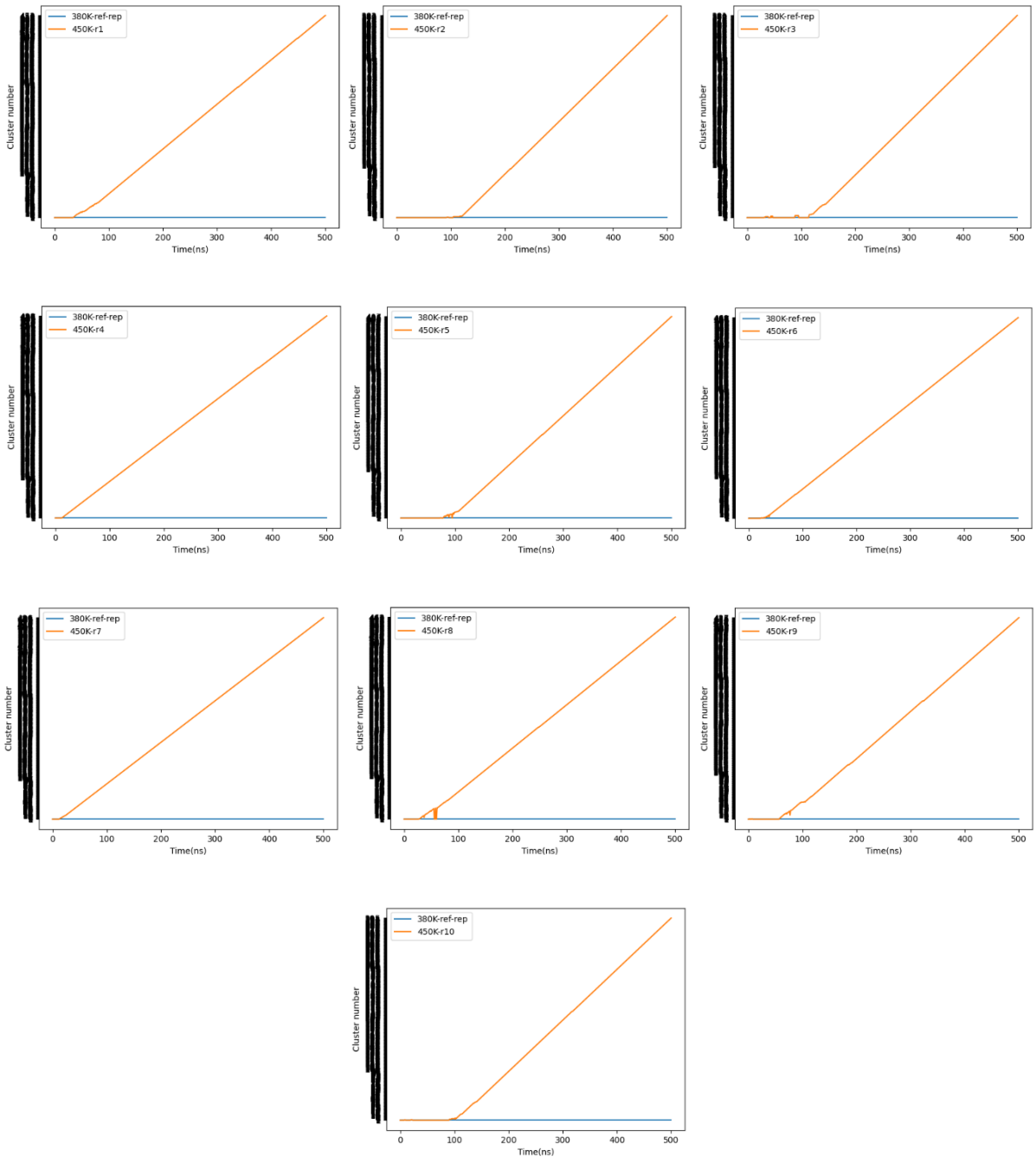

Figure 88. 2D-RMSD-based clustering for the ten replicas at 450 K using the AMBER99SB-disp force field.

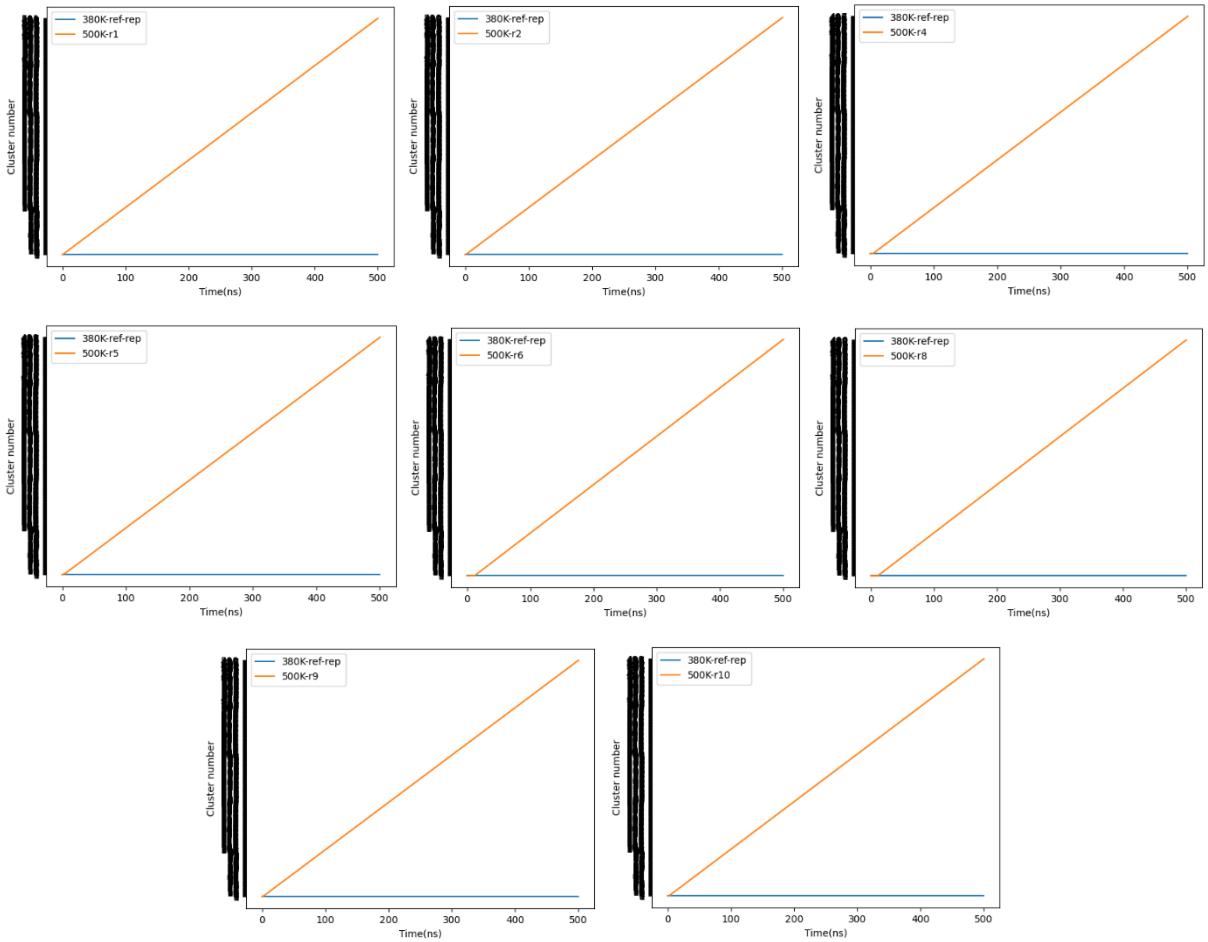

Figure 89. 2D-RMSD-based clustering for the eight replicas at 500 K using the AMBER99SB-disp force field.

7.3 Ladder-based temperature scanning MD simulations with CHARMM27

The RMSD, Rg, RMSF, intra protein H-bonds, SASA and 2D-RMSD-based clustering plots for the ladder-based temperature scanning MD simulations with CHARMM27 are shown below.

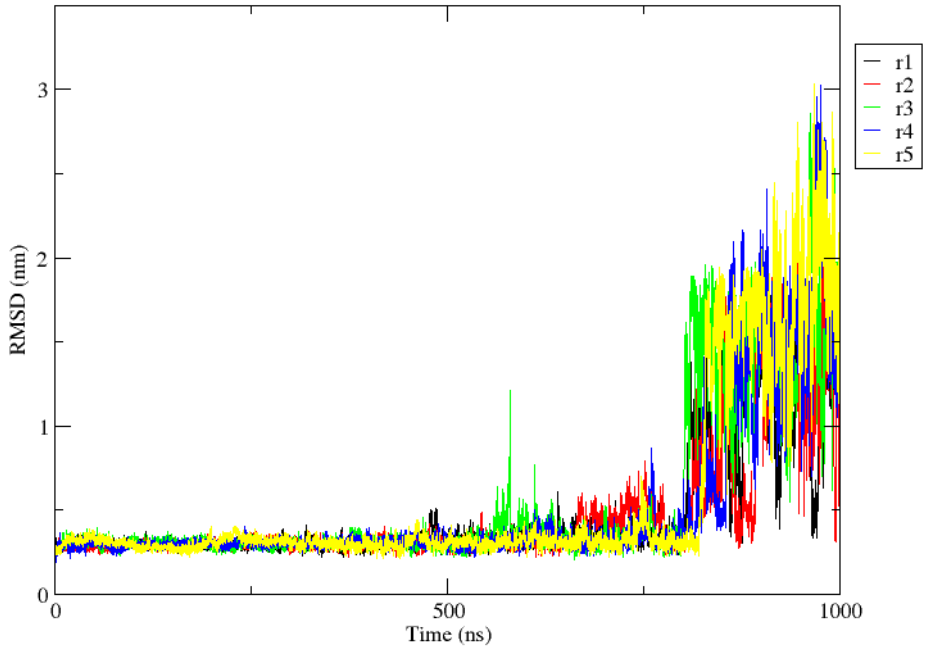

Figure 90. RMSD of the ladder-based temperature scanning MD simulations with CHARMM27.

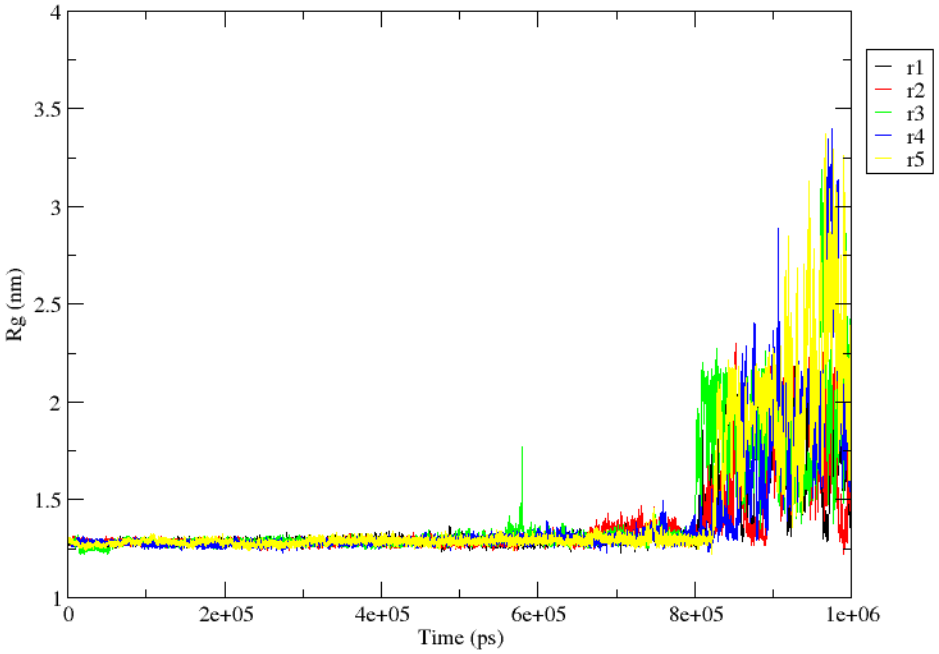

Figure 91. Rg of the ladder-based temperature scanning MD simulations with CHARMM27.

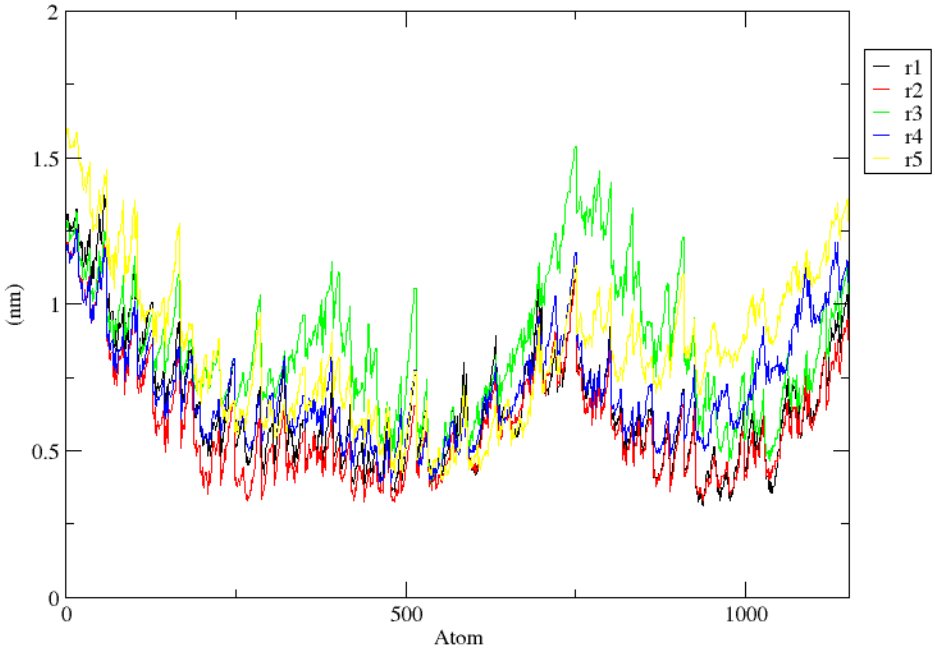

Figure 92. RMSF of the ladder-based temperature scanning MD simulations with CHARMM27.

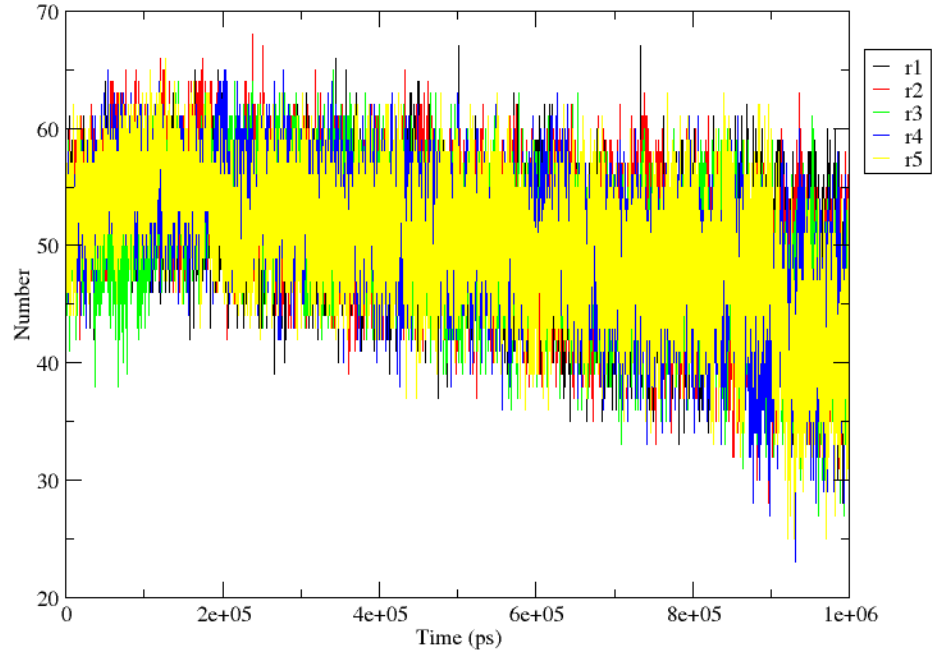

Figure 93. Intra protein H-bonds of the ladder-based temperature scanning MD simulations with CHARMM27.

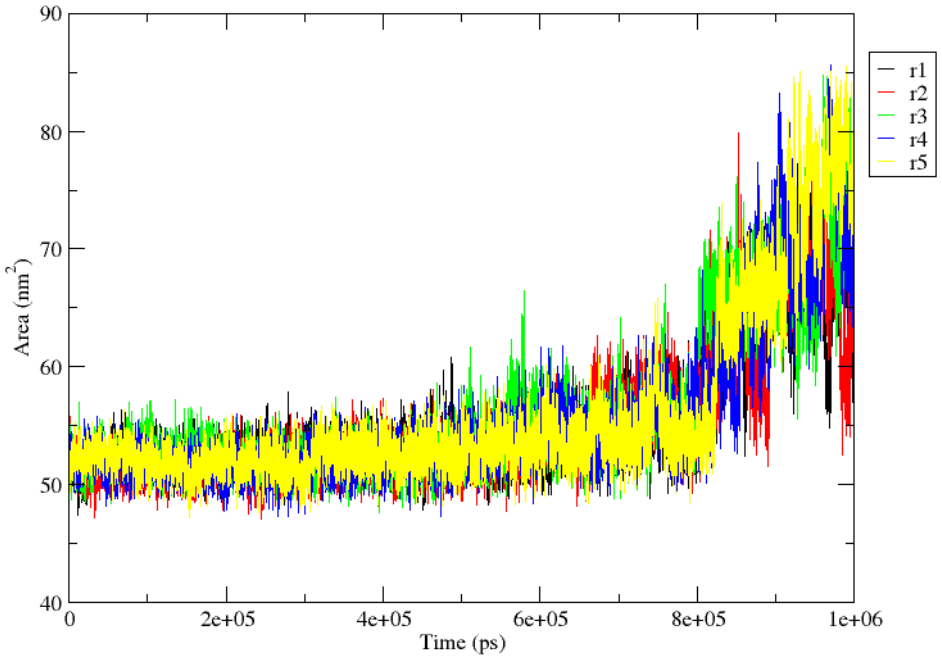

Figure 94. SASA of the ladder-based temperature scanning MD simulations with CHARMM27.

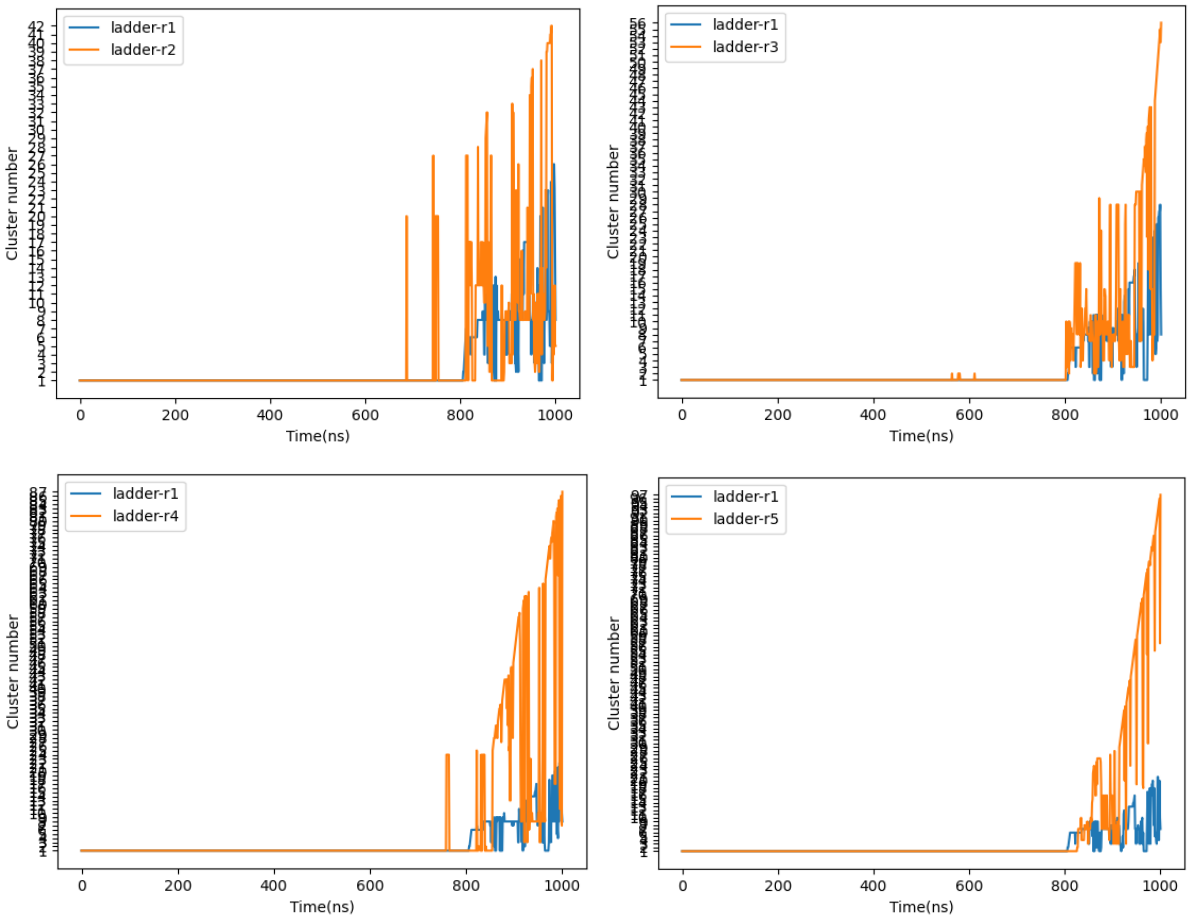

Figure 95. 2D-RMSD-based clustering for the five replicas of the ladder-based temperature scanning MD simulations with CHARMM27.

7.4 Ladder-based temperature scanning MD simulations with AMBER99-disp

The RMSD, Rg, RMSF, intra protein H-bonds, SASA and 2D-RMSD-based clustering plots for the ladder-based temperature scanning MD simulations with AMBER99-disp are shown below.

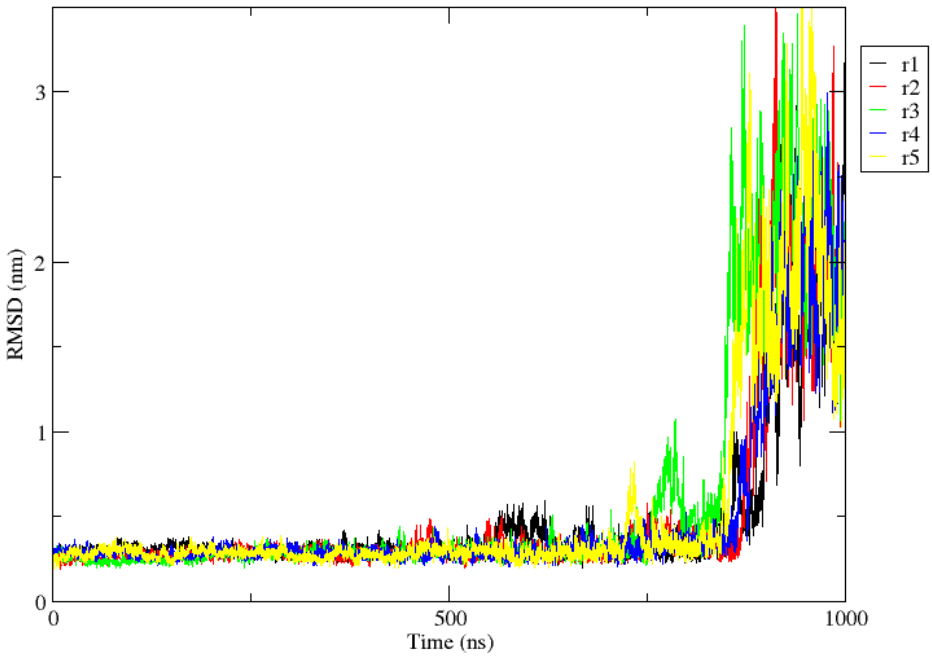

Figure 96. RMSD of the ladder-based temperature scanning MD simulations with AMBER99-disp.

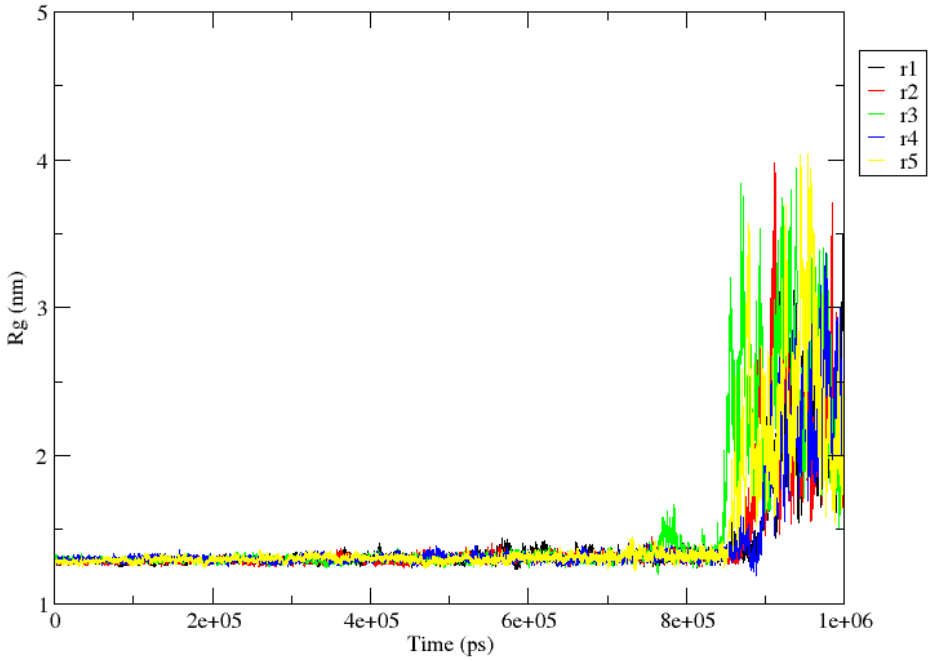

Figure 97. Rg of the ladder-based temperature scanning MD simulations with AMBER99-disp.

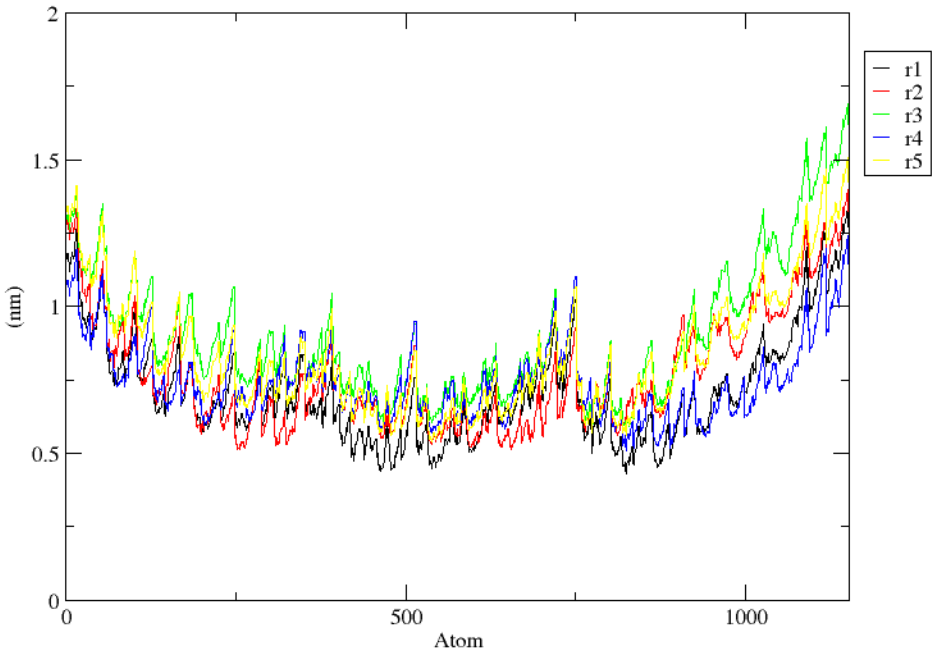

Figure 98. RMSF of the ladder-based temperature scanning MD simulations with AMBER99-disp.

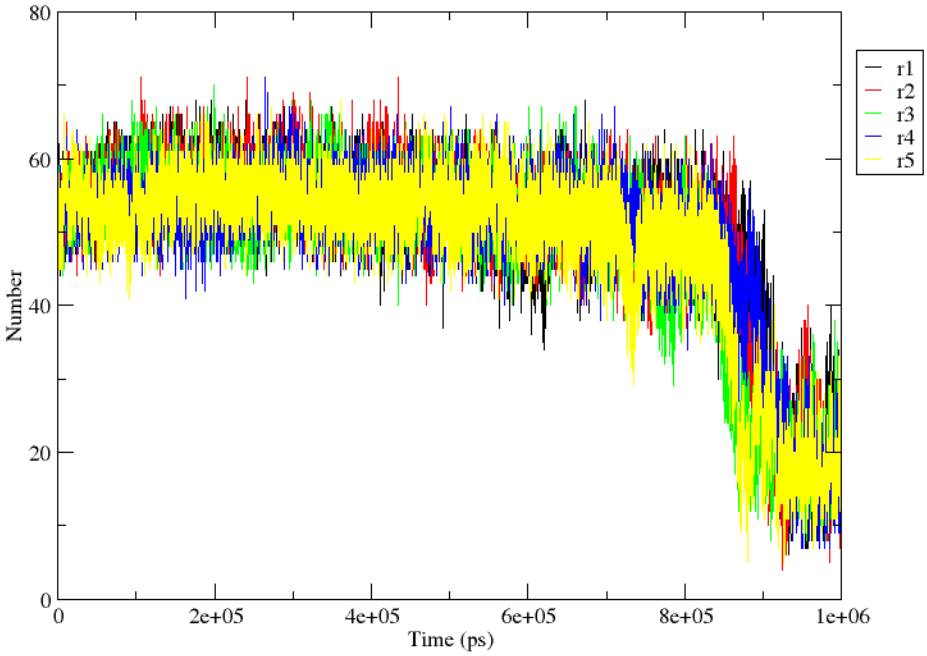

Figure 99. Intra protein H-bonds of the ladder-based temperature scanning MD simulations with AMBER99-disp.

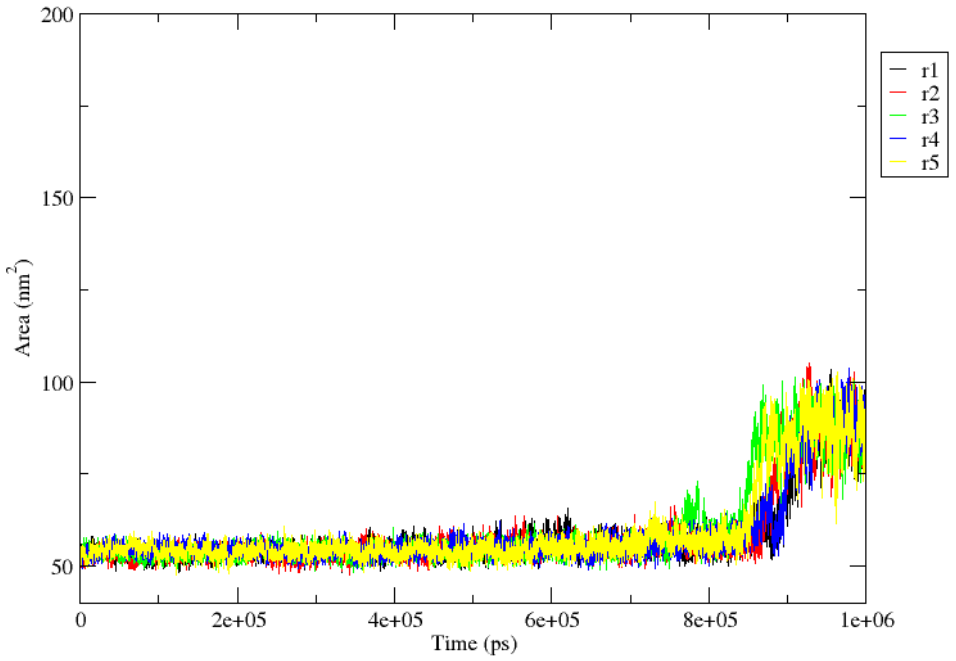

Figure 100. SASA of the ladder-based temperature scanning MD simulations with AMBER99-disp.

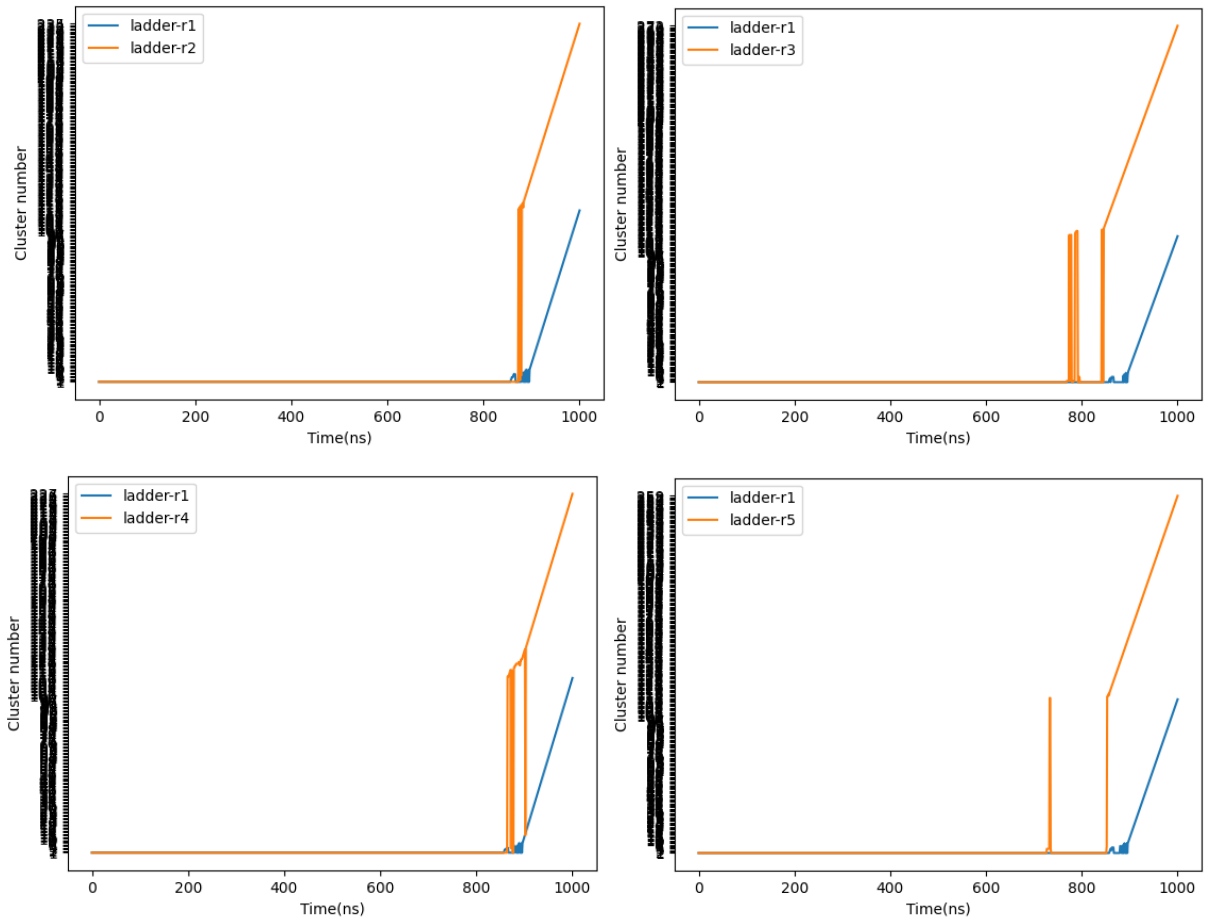

Figure 101. 2D-RMSD-based clustering for the five replicas of the ladder-based temperature scanning MD simulations with AMBER99-disp.

7.5 Ramp-based temperature scanning MD simulations with CHARMM27

The RMSD, Rg, RMSF, intra protein H-bonds, SASA and 2D-RMSD-based clustering plots for the ramp-based temperature scanning MD simulations with CHARMM27 are shown below.

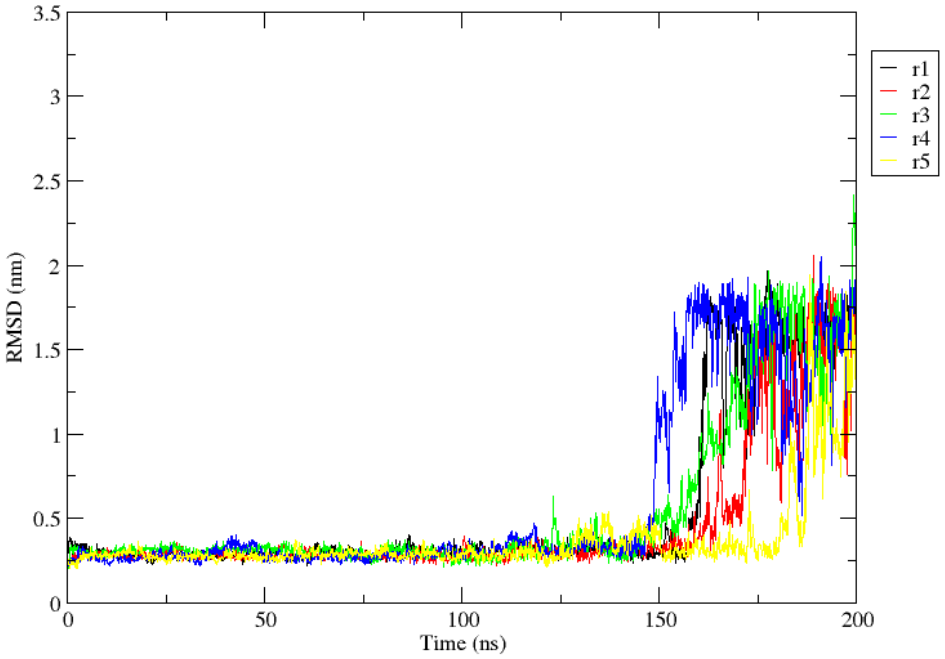

Figure 102. RMSD of the ramp-based temperature scanning MD simulations with CHARMM27.

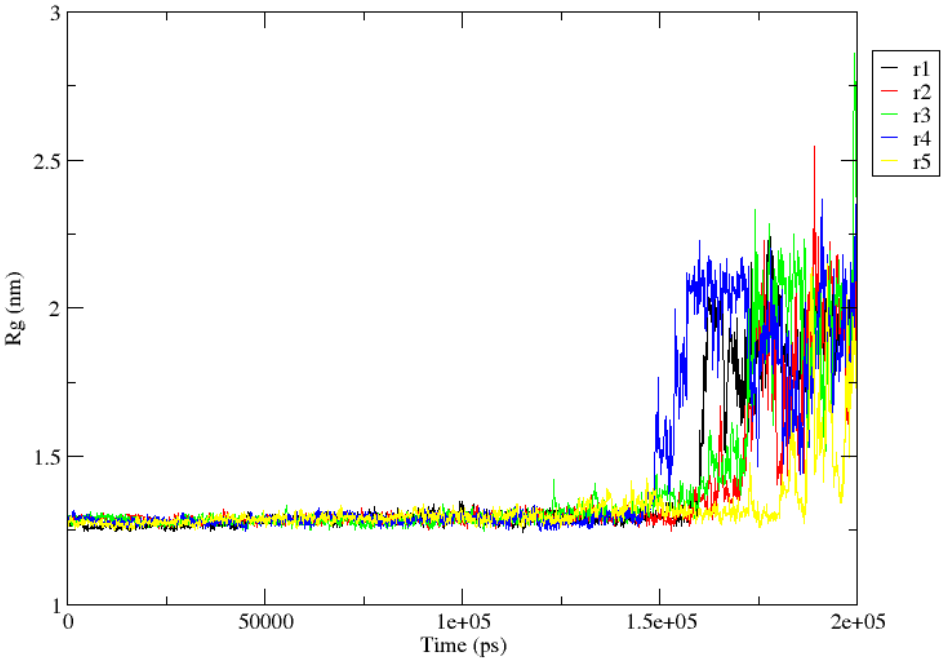

Figure 103. Rg of the ramp-based temperature scanning MD simulations with CHARMM27.

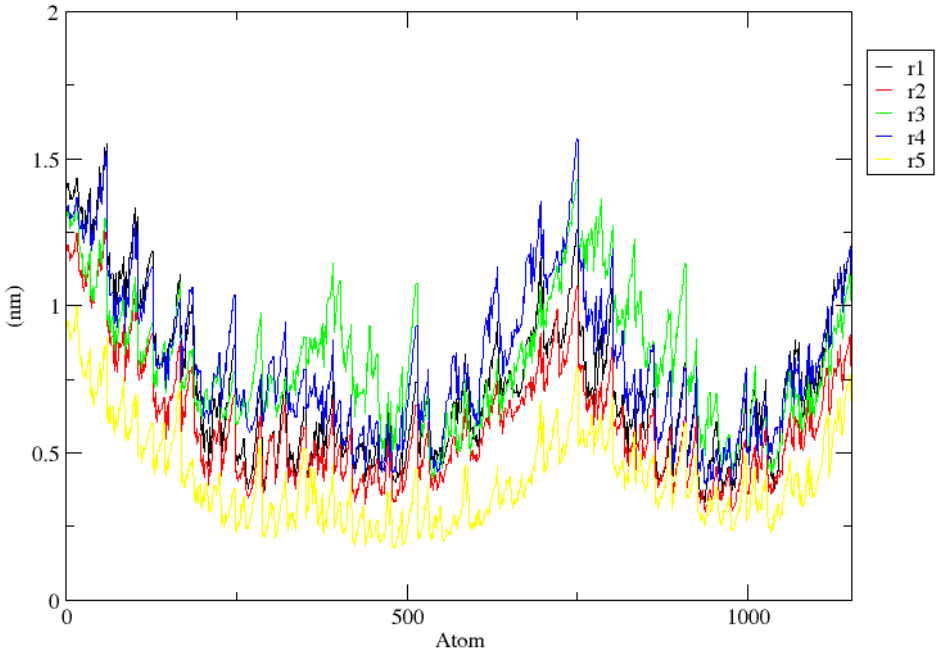

Figure 104. RMSF of the ramp-based temperature scanning MD simulations with CHARMM27.

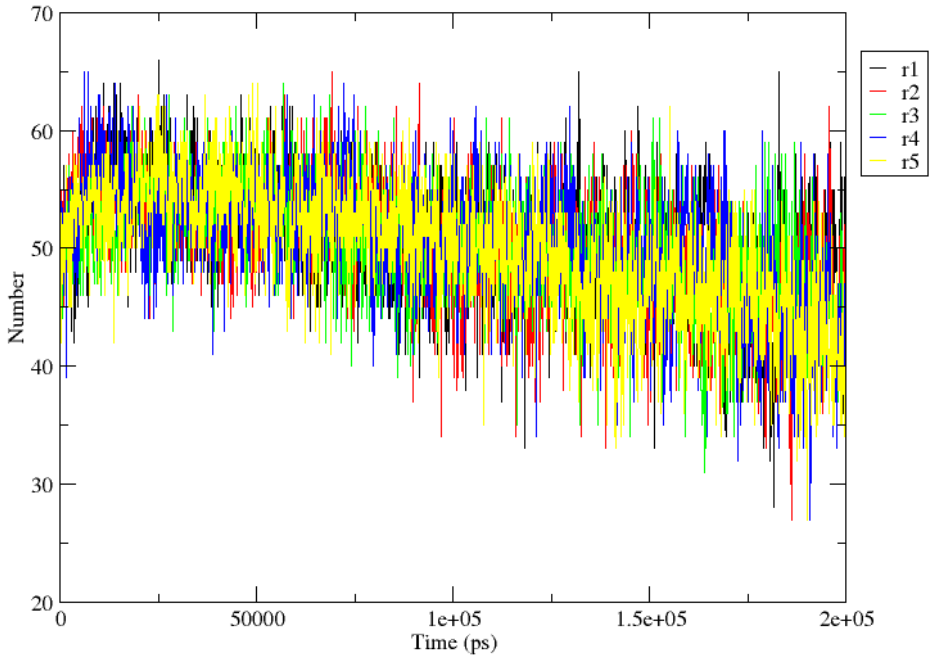

Figure 105. Intra protein H-bonds of the ramp-based temperature scanning MD simulations with CHARMM27.

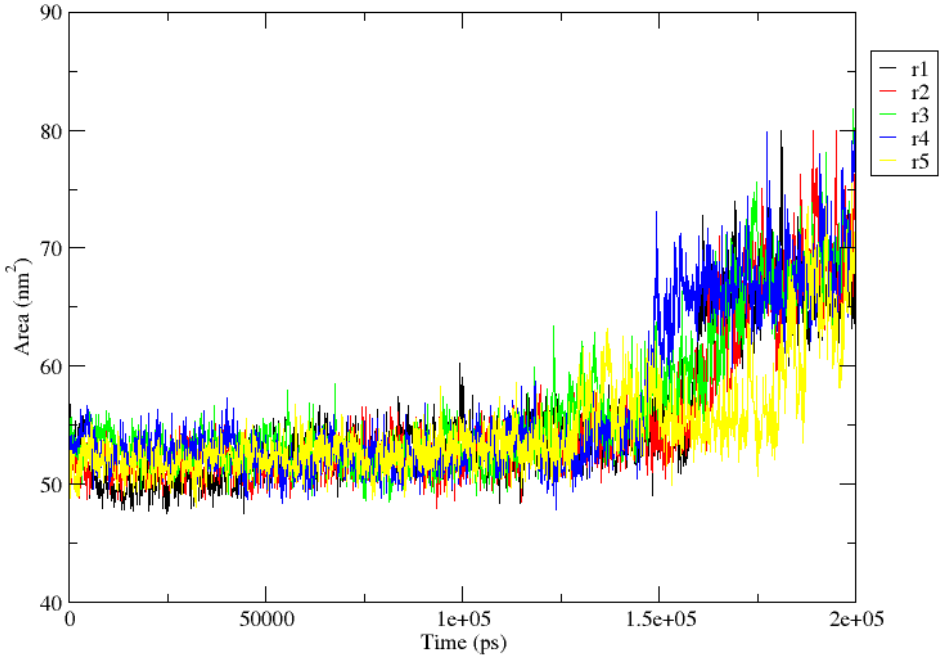

Figure 106. SASA of the ramp-based temperature scanning MD simulations with CHARMM27.

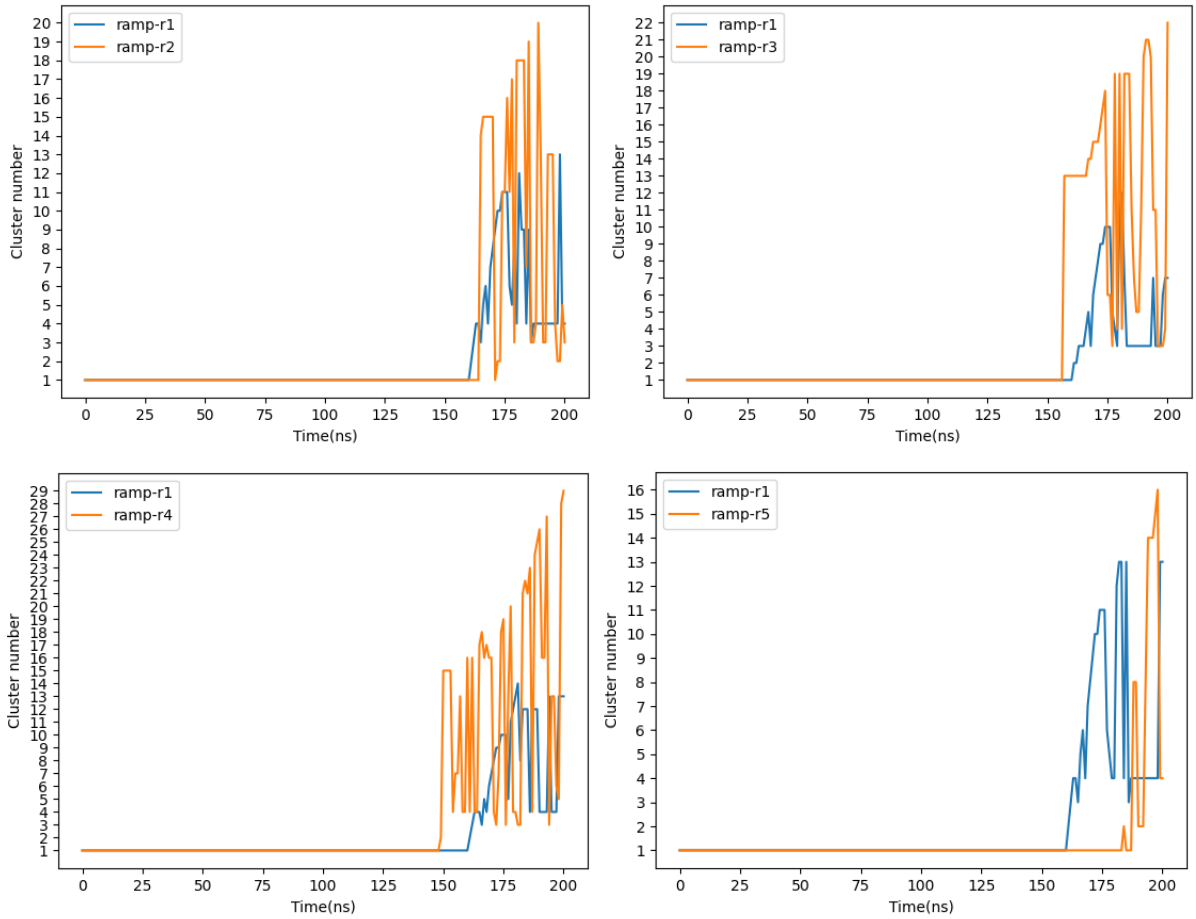

Figure 107. 2D-RMSD-based clustering for the five replicas of the ramp-based temperature scanning MD simulations with CHARMM27.

7.6 Ramp-based temperature scanning MD simulations with AMBER99-disp

The RMSD, Rg, RMSF, intra protein H-bonds, SASA and 2D-RMSD-based clustering plots for the ramp-based temperature scanning MD simulations with AMBER99-disp are shown below.

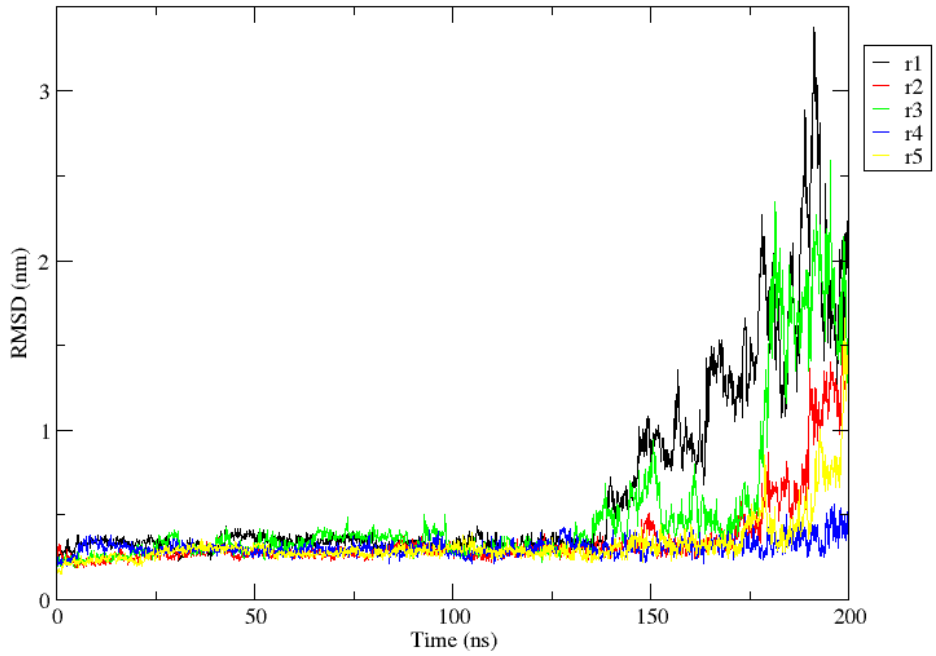

Figure 108. RMSD of the ramp-based temperature scanning MD simulations with AMBER99-disp.

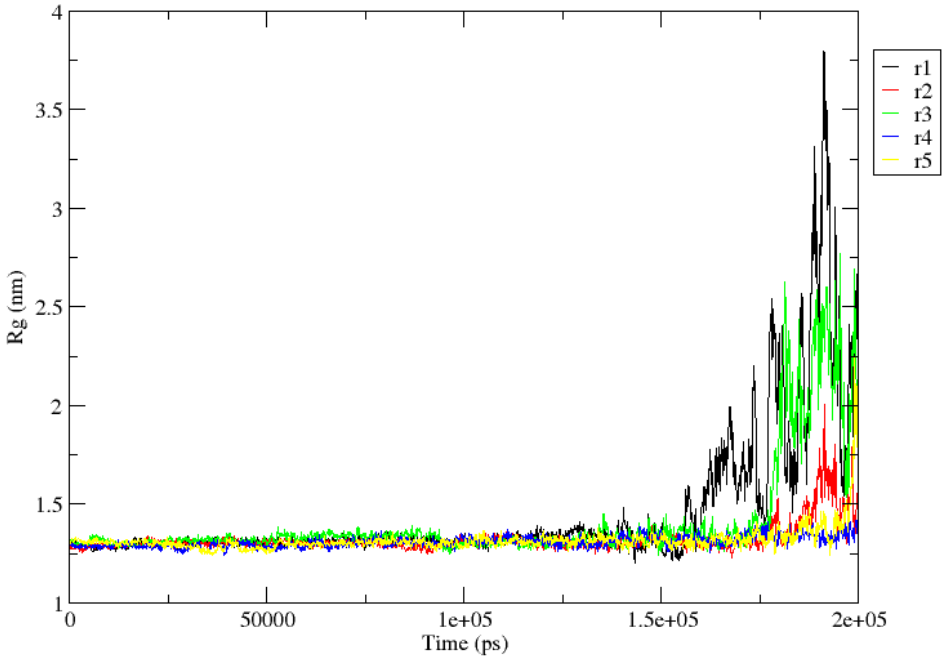

Figure 109. Rg of the ramp-based temperature scanning MD simulations with AMBER99-disp.

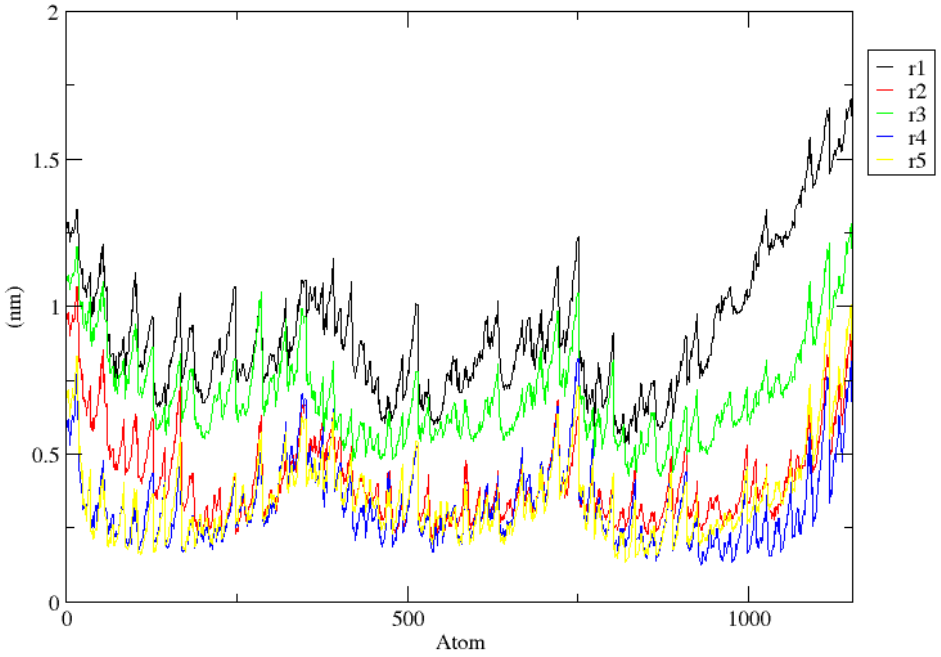

Figure 110. RMSF of the ramp-based temperature scanning MD simulations with AMBER99-disp.

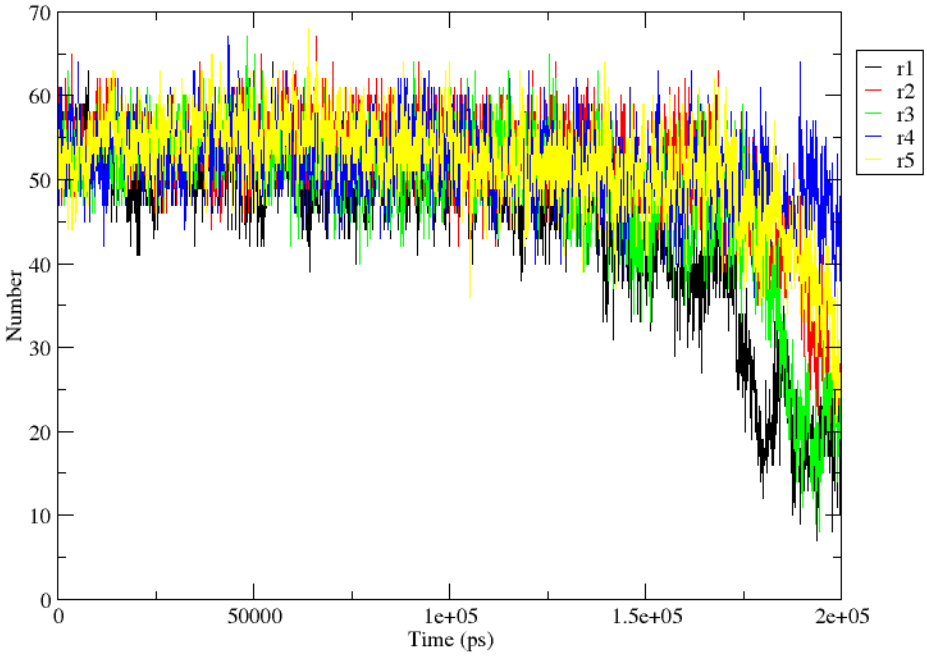

Figure 111. Intra protein H-bonds of the ramp-based temperature scanning MD simulations with AMBER99-disp.

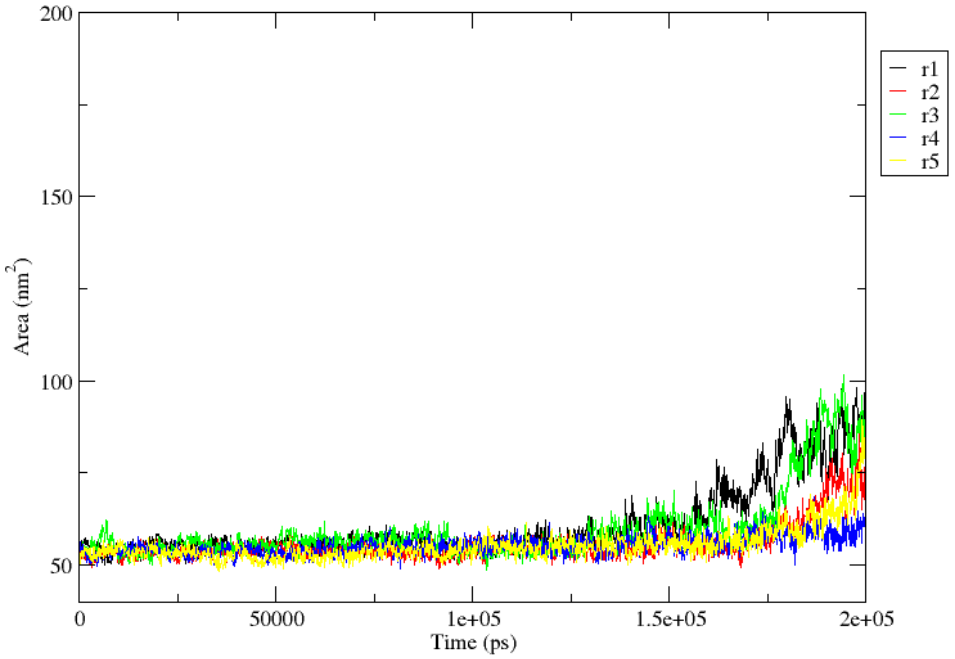

Figure 112. SASA of the ramp-based temperature scanning MD simulations with AMBER99-disp.

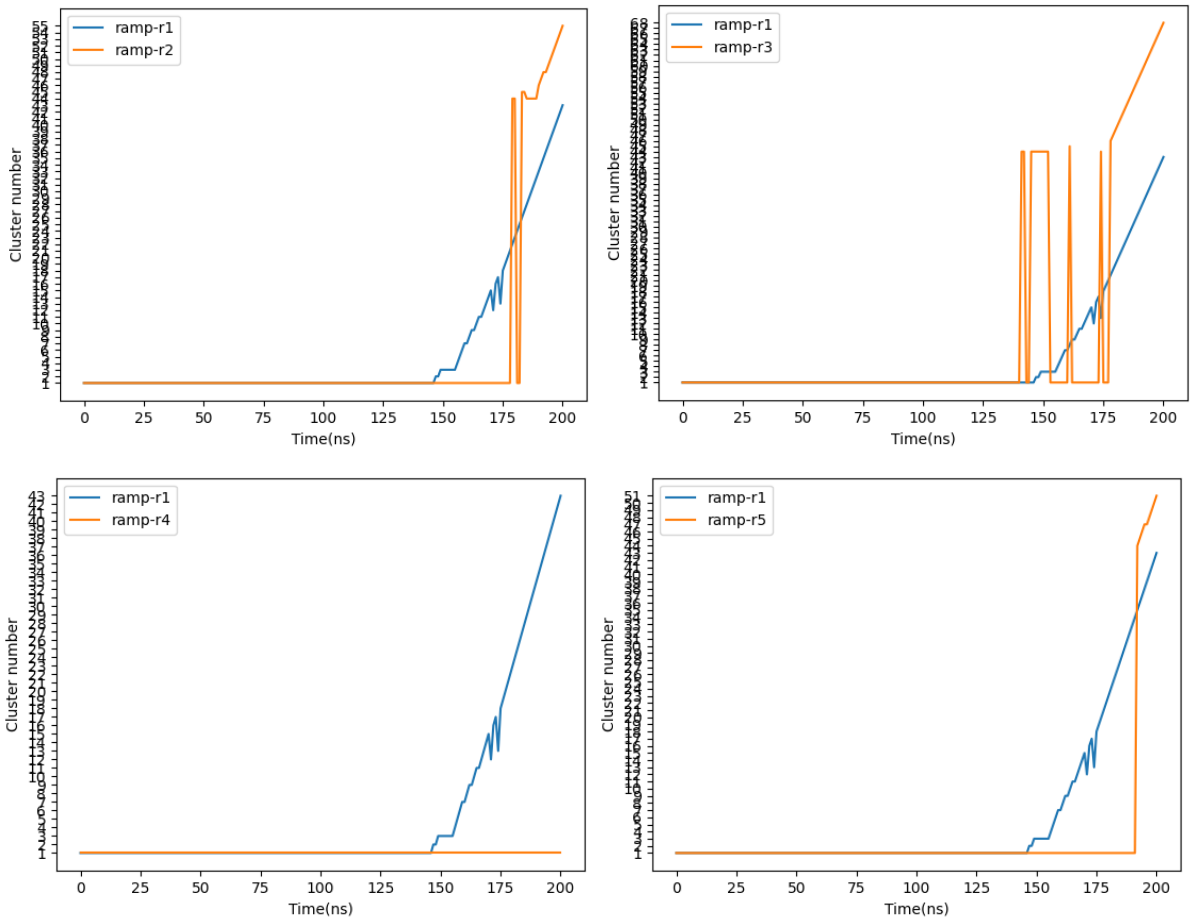

Figure 113. 2D-RMSD-based clustering for the five replicas of the ramp-based temperature scanning MD simulations with AMBER99-disp.

8. ANNEX II. SCRIPT FOR INTEGRATED ANALYSIS

```
from itertools import cycle
import matplotlib.pyplot as plt
import os
import sys
import shutil
import subprocess
from sklearn.preprocessing import StandardScaler
import glob
import pathlib

sys.path.insert(0,
"/home/dlopez/TFM/AMBER99disp_ff/2A3D/unfolding_analysis/")

from Analysis.energy import calculate_energy, read_energy
from Analysis.gyration import calculate_gyration, read_gyration
from Analysis.h_bonds_global_inter import calculate_h_bonds_global_inter,
read_h_bonds_global_inter
from Analysis.h_bonds_global_intra import calculate_h_bonds_global_intra,
read_h_bonds_global_intra
from Analysis.h_bonds_residue_inter import calculate_h_bonds_resid_inter,
read_h_bonds_resid_inter
from Analysis.h_bonds_residue_intra import calculate_h_bonds_resid_intra,
read_h_bonds_resid_intra
from Analysis.native_contacts_global import
calculate_native_contacts_global, read_native_contacts_global
from Analysis.native_contacts_local import calculate_native_contacts_local,
read_native_contacts_local
from Analysis.pressure import calculate_pressure, read_pressure
from Analysis.rmsd import calculate_rmsd, read_rmsd
from Analysis.rmsdist import calculate_rmsdist
from Analysis.rmsf import calculate_rmsf
from Analysis.sasa import calculate_sasa, read_sasa, read_sasa_local
from Analysis.ss import calculate_ss, read_ss, read_ss_coil
from Analysis.TM_score import calculate_TM_score, read_TM_score
from Plotting.colors import colors_replicas, colors_replicas_samples
from Plotting.plotting import plot_single_global, plot_single_local,
plot_compared_global, plot_compared_local



file_calc = {"energy.xvg": calculate_energy,
             "gyration.xvg": calculate_gyration,
             "h_bonds_global_inter.xvg": calculate_h_bonds_global_inter,
             "h_bonds_global_intra.xvg": calculate_h_bonds_global_intra,
             "native_contacts_global.txt":
calculate_native_contacts_global,
             "pressure.xvg": calculate_pressure,
             "rmsf.xvg": calculate_rmsf,
             "sasa.txt": calculate_sasa,
             "TMscore.xvg": calculate_TM_score}

local_file_calc = {"h_bonds_resid_inter.xvg":
calculate_h_bonds_resid_inter,
                   "h_bonds_resid_intra.xvg":
calculate_h_bonds_resid_intra,
```

```
                    "native_contacts_local.txt":
calculate_native_contacts_local}

file_read = [read_rmsd, read_energy, read_gyration,
read_h_bonds_global_inter, read_h_bonds_global_intra,
              read_native_contacts_global, read_pressure, read_sasa,
read_ss, read_ss_coil, read_TM_score]

local_file_read = [read_sasa_local, read_h_bonds_resid_inter,
read_h_bonds_resid_intra, read_native_contacts_local]


def calculate_all(base_dir, save_dir, ref_dir=None, replicas=None,
resid=None, kind=None, ref_replicas=None, ordered_output=False,
                  init_ns=0, time_step=1, final_ns=500):
    output_dir = base_dir + "/output/raw_analysis"
    trj_dir = base_dir + "/output/trajectories"
    final_dir = []
    final_out_dir = []
    final_trj_dir = []
    final_ref_dir = []
    if replicas:
        for replica in replicas:
            final_dir.append(base_dir + "/" + replica)
            final_out_dir.append(output_dir + "/" + replica)
            final_trj_dir.append(output_dir + "/" + replica)
            edr = glob.glob(base_dir + "/%s/*.edr" % replica)
            xtc = glob.glob(base_dir + "/%s/*.xtc" % replica)
            tpr = glob.glob(base_dir + "/%s/*.tpr" % replica)
            input_dir = base_dir + "/input/%s" % replica
            if not os.path.isdir(input_dir):
                os.makedirs(input_dir)
            for filetype in [edr, xtc, tpr]:
                for item in filetype:
                    shutil.copy2(item, input_dir)
        new_replicas = replicas
    else:
        final_dir.append(base_dir)
        final_out_dir.append(output_dir)
        final_trj_dir.append(trj_dir)
        new_replicas = ["data"]
        edr = glob.glob(base_dir + "/*.edr")
        xtc = glob.glob(base_dir + "/*.xtc")
        tpr = glob.glob(base_dir + "/*.tpr")
        input_dir = base_dir + "/input"
        if not os.path.isdir(input_dir):
            os.makedirs(input_dir)
        for filetype in [edr, xtc, tpr]:
            for item in filetype:
                shutil.copy2(item, input_dir)
    if not ref_dir:
        ref_dir = base_dir
        final_ref_dir = final_dir
    elif ref_replicas:
        for replica in ref_replicas:
            final_ref_dir.append(ref_dir + "/" + replica)
    else:
        final_ref_dir.append(ref_dir)
    i = 0
    for anal_dir in final_dir:
```

```
        subprocess.call("echo 'System' | gmx_mpi trjconv -f %s/*.xtc -o
%s/reference.gro -s %s/*.tpr -dump %i -pbc mol -ur compact" %
                        (anal_dir, anal_dir, anal_dir, int(init_ns*1000)),
shell=True)
    for anal_dir in final_dir:
        print(anal_dir)
        if not os.path.isdir(save_dir + "/" + new_replicas[i]):
            pathlib.Path(save_dir + "/" +
new_replicas[i]).mkdir(parents=True, exist_ok=True)
        for file in file_calc:
            print(file)
            if not os.path.isfile(anal_dir + "/" + file):
                if file == "TMscore.xvg":
                    file_calc[file](base_dir, base_ref_dir=ref_dir,
replicas=replicas, ordered_output=ordered_output,
                                    ref_replicas=ref_replicas,
init_ns=init_ns, time_step=time_step, final_ns=final_ns)
                else:
                    file_calc[file](base_dir, base_ref_dir=ref_dir,
replicas=replicas, ordered_output=ordered_output,
                                    ref_replicas=ref_replicas)
            if ordered_output:
                print(anal_dir, file, save_dir, new_replicas[i])
                shutil.copy2(anal_dir + "/" + file, save_dir + "/" +
new_replicas[i] + "/")
        if not os.path.isfile(anal_dir + "/rmsd.xvg"):
            print("rmsd")
            calculate_rmsd(base_dir, ref_dir=ref_dir, replicas=replicas,
ordered_output=ordered_output, ref_replicas=ref_replicas)
            if ordered_output:
                shutil.copy2(anal_dir + "/rmsd.xvg", save_dir + "/" +
new_replicas[i] + "/")
        if not os.path.isfile(anal_dir + "/rmsdist.xvg"):
            print("rmsdist")
            calculate_rmsdist(base_dir, ref_dir=ref_dir, replicas=replicas,
ordered_output=ordered_output, ref_replicas=ref_replicas)
            if ordered_output:
                shutil.copy2(anal_dir + "/rmsdist.xvg", save_dir + "/" +
new_replicas[i] + "/")
        if not os.path.isfile(ref_dir + "/scount.xvg"):
            print("ss")
            calculate_ss(base_dir, kind=kind, replicas=replicas,
ordered_output=ordered_output)
            if ordered_output:
                shutil.copy2(anal_dir + "/scount.xvg", save_dir + "/" +
new_replicas[i] + "/")
        i += 1

def read_all(base_dir, resid=None, replicas=False, ordered_output=False):
    output_dir = base_dir + "/output/raw_analysis"
    final_dir = []
    final_out_dir = []
    results = []
    if replicas:
        n_replicas = len(replicas)
        for replica in replicas:
            final_dir.append(base_dir + "/" + replica)
            final_out_dir.append(output_dir + "/" + replica)
    else:
        n_replicas = 1
        final_dir.append(base_dir)
```

```
        final_out_dir.append(output_dir)
    data = []
    for read_function in file_read:
        data.append(read_function(base_dir, replicas=replicas))
    if resid:
        for read_function in local_file_read:
            data.append(read_function(base_dir, replicas=replicas,
resid=resid))
    for j in range(n_replicas):
        time_series = []
        for i in range(len(data)):
            for item in data[i][j][0]:
                if round(item, 1) not in time_series:
                    time_series.append(round(item, 1))
        ordered_data = []
        for k in range(len(time_series)):
            ordered_data.append([time_series[k]])
            for i in range(len(data)):
                value = "NA"
                for n in range(len(data[i][j][0])):
                    if time_series[k] == round(data[i][j][0][n], 1):
                        value = data[i][j][1][n]
                        break
                ordered_data[k].append(value)
        file = open(final_dir[j], "w")
        file.write("Time RMSD Energy Gyration H_bonds_global_inter
H_bonds_global_intra Native_contacts_global"
                   " Pressure SASA SS SS_coil")
        if resid:
            file.write(" SASA_local H_bonds_resid_inter H_bonds_resid_intra
Native_contacts_local\n")
        else:
            file.write("\n")
        for element in ordered_data:
            file.write("%s\n" % " ".join(str(x) for x in element))
        file.close()
        if ordered_output:
            if not os.path.exists(final_out_dir[j]):
                os.makedirs(final_out_dir[j])
            shutil.copy(final_dir[j], final_out_dir[j])
        results.append(ordered_data)
    return results



rlist = ["r5"]
temp_1 = "450K"
root_dir = "/home/dlopez/TFM/AMBER99disp_ff/2A3D/INTEGRATED_ANALYSIS"
mut_resid = None
kind = None

base_dir_1 = root_dir + "/" + temp_1 + "/data_in"
save_dir_1 = root_dir + "/" + temp_1 + "/data_out"
ref_dir_1 = root_dir + "/" + temp_1 + "/reference_in"

base_dirs = [base_dir_1]
save_dirs = [save_dir_1]
ref_dirs = [ref_dir_1]

for i in range(len(base_dirs)):
```

```
    calculate_all(base_dirs[i], save_dirs[i], ref_dir=ref_dirs[i],
replicas=rlist, ref_replicas=rlist, resid=None, kind=None)
    plot_single_global(base_dirs[i], replicas=rlist, kind=kind)
```

9. ANNEX III. SCRIPT FOR 2D-RMSD-BASED CLUSTERING ANALYSIS

```
### Program to perform clustering analysis based on RMSD-2D obtained from
MD trajectories. The program execute the clustering calculations and
compare two trajectories in order to detect structural behavioural
differences and where (simulation time) unfolding events take place along
the trajectories

from sklearn.cluster import DBSCAN, KMeans, AgglomerativeClustering
from matplotlib import pyplot as plt
import subprocess
import numpy as np
import csv
import os
from os import path
from gromacs.fileformats.xvg import XVG


def read_xpm(file_dir):
    in_file = open(file_dir)
    line = in_file.readline()
    while line[0] != "\"":
        line = in_file.readline()
    line = in_file.readline()
    code_dict = {}
    while line[0] == "\"":
        line = line[1:].split()
        code_dict[line[0]] = float(line[5][1:-1])
        line = in_file.readline()
    result = []
    for line in in_file.readlines():
        if line[0] == "\"":
            line_results = []
            number = -2
            if line[-2] == ",":
                number -= 1
            for c in line[1:number]:
                line_results.append(code_dict[c])
            result.append(line_results)
    return result


def get_clustering(distance_matrix, threshold):
    clustering = AgglomerativeClustering(n_clusters=None,
affinity="precomputed", linkage="average",
distance_threshold=threshold).fit_predict(distance_matrix)
    return clustering


def sele_backbone(resid, file_dir, save_index):
    subprocess.call("gmx_mpi select -f %s_processed_backbone.pdb -s
%s_processed_backbone.pdb -on %sindex_back_resid%s.ndx -resnr number -
select "
                    "\"(group \"Backbone\" and not resid %i) and same
residue as within 1 of com of resid %i\""
```

```
                    % (file_dir,  file_dir, save_index, resid, resid,
resid), shell=True)


def plot_multiple_clustering_global(file_dir_1, file_dir_2,
save_global_dir, time_step_1, time_step_2, type_1, type_2, rep_2,
rmsd_threshold):
    if not os.path.exists("%s_processed_backbone.xtc" % (file_dir_1)):
        subprocess.call("echo 4 | gmx_mpi trjconv -f %s_processed.xtc -s
%s.tpr  -o %s_processed_backbone.xtc -skip 10" %
                    (file_dir_1, file_dir_1, file_dir_1), shell=True)
    if not os.path.exists("%s_processed_backbone.pdb" % (file_dir_1)):
        subprocess.call("echo 4 | gmx_mpi trjconv -f %s_processed.xtc -s
%s.tpr  -o %s_processed_backbone.pdb -dump 0" %
                    (file_dir_1, file_dir_1, file_dir_1), shell=True)
    if not os.path.exists("%s_processed_backbone.xtc" % (file_dir_2)):
        subprocess.call("echo 4 | gmx_mpi trjconv -f %s_processed.xtc -s
%s.tpr -o %s_processed_backbone.xtc -skip 10" %
                    (file_dir_2, file_dir_2, file_dir_2), shell=True)
    if not os.path.exists("%s_processed_backbone.pdb" % (file_dir_2)):
        subprocess.call("echo 4 | gmx_mpi trjconv -f %s_processed.xtc -s
%s.tpr -o %s_processed_backbone.pdb -dump 0" %
                    (file_dir_2, file_dir_2, file_dir_2), shell=True)
    if not os.path.exists("%srmsd_%s_ref_global.xpm" % (save_global_dir,
type_1)):
        subprocess.call("echo \'4 4\' | gmx_mpi rms -f
%s_processed_backbone.xtc -s %s_processed_backbone.pdb -o
%srmsd_%s_ref_global.xvg -m %srmsd_%s_ref_global.xpm" %
                    (file_dir_1, file_dir_1, save_global_dir, type_1,
save_global_dir, type_1), shell=True)
    if not os.path.exists("%srmsd_%s_%s_global.xpm" % (save_global_dir,
type_2, rep_2)):
        subprocess.call("echo \'4 4\' | gmx_mpi rms -f
%s_processed_backbone.xtc -s %s_processed_backbone.pdb -o
%srmsd_%s_%s_global.xvg -m %srmsd_%s_%s_global.xpm" %
                    (file_dir_2, file_dir_2, save_global_dir, type_2,
rep_2, save_global_dir, type_2, rep_2), shell=True)
    if not os.path.exists("%srmsd_%s_ref_%s_%s_global.xpm" %
(save_global_dir, type_1, type_2, rep_2)):
        subprocess.call("echo \'4 4\' | gmx_mpi rms -f
%s_processed_backbone.xtc -f2 %s_processed_backbone.xtc "
                    "-s %s_processed_backbone.pdb -o
%srmsd_%s_ref_%s_%s_global.xvg -m %srmsd_%s_ref_%s_%s_global.xpm" %
                    (file_dir_1, file_dir_2, file_dir_1, save_global_dir,
type_1, type_2, rep_2, save_global_dir, type_1, type_2, rep_2), shell=True)
    matrix_1 = read_xpm("%srmsd_%s_ref_global.xpm" % (save_global_dir,
type_1))
    matrix_2 = read_xpm("%srmsd_%s_%s_global.xpm" % (save_global_dir,
type_2, rep_2))
    matrix_1_2 = read_xpm("%srmsd_%s_ref_%s_%s_global.xpm" %
(save_global_dir, type_1, type_2, rep_2))
    matrix_2_1 = np.array(matrix_1_2[::-1]).T.tolist()[::-1]
    matrix_a = []
    for i in range(len(matrix_2)):
        result = matrix_1_2[i] + matrix_2[i]
        matrix_a.append(result)
    n_2 = len(matrix_a)
    matrix_b = []
    for i in range(len(matrix_1)):
        result = matrix_1[i] + matrix_2_1[i]
        matrix_b.append(result)
```

```
    n_1 = len(matrix_b)
    matrix = matrix_a + matrix_b
    matrix = matrix[::-1]
    result = get_clustering(matrix, rmsd_threshold)
    numbering = 1
    numbering_dict = {}
    y_1 = []
    x_1 = []
    time_1 = []
    for i in range(n_1):
        time_1.append(i * time_step_1 * 1000)
        x_1.append(i * time_step_1)
        if result[i] not in numbering_dict:
            numbering_dict[result[i]] = numbering
            numbering += 1
        y_1.append(numbering_dict[result[i]])
    a = XVG(array=[time_1, y_1])
    a.write("%sclusters_%s_reference.xvg" % (save_global_dir, type_1))


    y_2 = []
    x_2 = []
    time_2 = []
    for i in range(n_2):
        x_2.append(i * time_step_2)
        time_2.append(i * time_step_2 * 1000)
        if result[i + n_1] not in numbering_dict:
            numbering_dict[result[i + n_1]] = numbering
            numbering += 1
        y_2.append(numbering_dict[result[i + n_1]])
    a = XVG(array=[time_2, y_2])
    a.write("%sclusters_%s_%s.xvg" % (save_global_dir, type_2, rep_2))


    plt.plot(x_1, y_1, label=type_1 + "-ref-rep")
    plt.plot(x_2, y_2, label=type_2 + "-" + rep_2)
    ymax = (max([max(y_1), max(y_2)]))
    ylabels = range(1, ymax + 1)
    new_yticks = ylabels
    plt.yticks(new_yticks, ylabels)
    plt.xlabel("Time(ns)")
    plt.ylabel("Cluster number")
    plt.legend(loc="best")
    plt.savefig("%sglobal_clustering_%s_ref_vs_%s_%s.png" %
(save_global_dir, type_1, type_2, rep_2))
    plt.close()


type_1 = "380K"
type_2 = "500K"
rep_1 = "r6"
rep_2 = "r10"
file_dir_1 = "/home/dlopez/TFM/AMBER99disp_ff/2A3D/%s/%s/8-production/2a3d-
prod" % (type_1, rep_1)
file_dir_2 = "/home/dlopez/TFM/AMBER99disp_ff/2A3D/%s/%s/8-production/2a3d-
prod" % (type_2, rep_2)
rmsd_global_threshold = 0.6
rmsd_local_threshold = 0.3
save_global_dir = "/home/dlopez/TFM/AMBER99disp_ff/2A3D/CLUSTERING-
FILES/global/%s-vs-%s/reference-vs-%s_%snm/" % (type_1, type_2, rep_2,
rmsd_global_threshold)
```

```
if not os.path.exists("%s" % (save_global_dir)):
    os.makedirs("%s" % (save_global_dir))
save_local_dir = "/home/dlopez/TFM/AMBER99disp_ff/2A3D/CLUSTERING-
FILES/local/%s-vs-%s/reference-vs-%s_%snm/" % (type_1, type_2, rep_2,
rmsd_local_threshold)
if not os.path.exists("%s" % (save_local_dir)):
    os.makedirs("%s" % (save_local_dir))
time_step_1 = 1
time_step_2 = 1
save_index_1 = "/home/dlopez/TFM/AMBER99disp_ff/2A3D/%s/%s/8-production/" %
(type_1, rep_1)
save_index_2 = "/home/dlopez/TFM/AMBER99disp_ff/2A3D/%s/%s/8-production/" %
(type_2, rep_2)
resid = 116
n_groups = 9

plot_multiple_clustering_global(file_dir_1, file_dir_2, save_global_dir,
time_step_1, time_step_2, type_1, type_2, rep_2, rmsd_global_threshold)
```