CHAPTER III
RESULTS AND DISCUSSION

3.1 Protocol for preparation of model system with pseudoatoms.

There are many steps to prepare the protein system before running NAMD-PaDSAR such as, attaching EP pseudoatoms into the protein, adding the OXY and NIC pseudoatoms, generating a protein structure file or psf file and a configuration file etc. The preparation of model system with pseudoatoms has been done through script commands executed using the program VMD. According to Figure 2.1, there are two VMD-script files called *run.tcl* and *res.tcl*. The *run.tcl* consists of two parts; user-defined and command part. The user defined part (shown in gray square box below) allows users to modify their own parameters and options for the simulation such as number of OXY and NIC, number of simulation steps, input file name, the residue to be patched EPs atoms etc. In the command part (shown in gray square box) users do not need to change anything.


```
#################      general part
set     inputPDB      "distort.pdb"
set     selection     "protein and not name POT"
set     no_Segment    4
set     seg_Name      { A B C D }
```

1: To set the name of protein.

2: To select only protein from monomer to generate the tetramer.

3: To specify number of segment. 4 for generating tetramer.

4: To set the name of each segment.

```
#################      Pseudo atoms part
set     noOXY          300
set     noNIC          300
set     psueLen        45
set     max_z_OXY      16
set     min_z_OXY      -16
set     max_z_NIC      45
```

```
set    min_z_NIC      -45
set    hole           30
```

This part is for generating the OXY and NIC initial position and all value are described in Figure 3.1 (Å unit.)

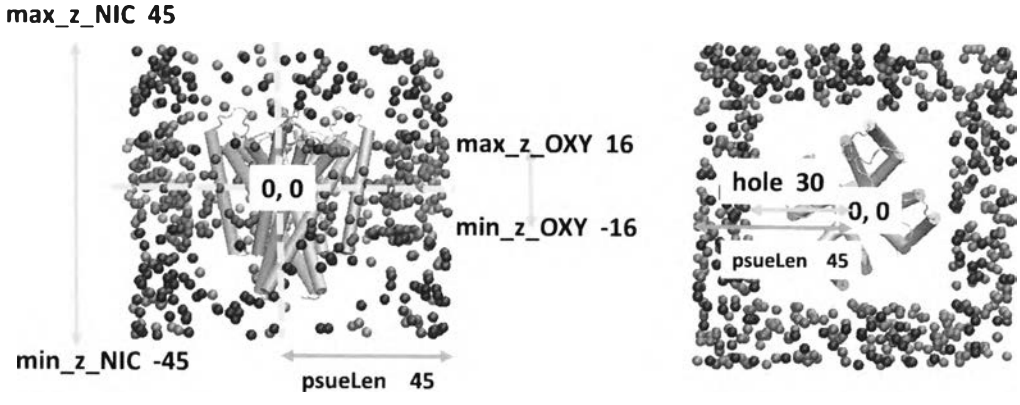

Figure 3.1 The parameters for generating the OXY and NIC initial position.


```
#################      PSF part
set    topo_file      "toptest.inp"
set    ep1_name       "EP1"
set    ep2_name       "EP2"
set    ep3_name       "EP3"
set    patchep1       { 25, 26, 29, 30, 33, 36, 40, 44, 47, 48, 86, 87, 89, 91, 92, 95, 96,
97, 98, 99, 100, 101, 102, 103,105, 106, 107, 108, 109, 110, 111, 113, 114, 115,
118, 119}
set    patchep2       { 23, 24, 27, 28, 39, 49, 50, 52, 117, 120}
set    patchep3       { 31, 32, 34, 35, 37, 38, 41, 42, 45, 46, 90, 94 }
```

1: To insert a topology file

2, 3, 4: name of EP type one, two and three respectively.

5, 6, 7: To specify which residue in protein to be patched with EP types such as EP1, EP2 and EP3 respectively.

```
#################       another part
set    fixatom          "name POT or protein and resid 60 to 80"
set    ndres            "protein and not (resid 55 to 73 or resid 107 to 109) "
```

1: To specify the fix atom.

2: To specify the residue to be applied the secondary structure restrains.

```
# command part : No need to change
source res.tcl
loadPDB $inputPDB $selection
set k 1
foreach M $seg_Name {
        gen_SEG  "inputPDB.pdb" "$M" "$k"
       incr k
}
cat_File        $no_Segment
prep_File       protein.pdb  $seg_Name
make_Pseudo         [expr    $noOXY*4]    OXY    $psueLen         $min_z_OXY
$max_z_OXY
make_Pseudo2        $noOXY OXY $hole $hole $max_z_OXY $min_z_OXY
make_Pseudo         [expr $noNIC*8] NIC $psueLen  $min_z_NIC  $max_z_NIC
make_Pseudo2        $noNIC NIC $hole $hole $max_z_OXY $min_z_OXY
gen_Psf         $ep1_name   $patchep1   $ep2_name   $patchep2     $ep3_name
$patchep3 $topo_file $seg_Name
move2origin   $fixatom
2ndrestrain   $ndres
rest_layer      buble.tcl  $noNIC
gen_conf        proteininput namd.conf
delfile         $seg_Name $no_Segment
exit
```

The *res.tcl* is a subprogram to be called after executing the *run.tcl*.

Set of command libraries for generating multi-mer protein from monomer	

proc loadPDB { pdbinput select} {
mol load pdb $pdbinput
set chain [atomselect top "$select"]
$chain writepdb inputPDB.pdb
mol delete all
}
	proc gen_SEG { pdbinput seglabel seg_name } {
	mol load pdb $pdbinput
	set chain [atomselect top "segname $seglabel"]
	$chain set chain $seglabel
	$chain set segname $seglabel
	$chain writepdb seg$seg_name.pdb
	mol delete all
	}

proc grep_get {re args out} {
set files [eval glob -types f $args]
foreach file $files {
set fp [open $file]
set oo [open $out w]
while { [gets $fp line] >= 0} {
if [regexp -- $re $line] {
if {[llength $files] > 1} {puts -nonewline $file:}
puts $line
puts $oo $line
}
}
close $fp
close $oo
}
}
	proc cat_File2 { no } {
	set new [open "protein.pdb" w]
	close $new
	for { set i 1 } { $i <= $no } { incr i } {
	set oldfile [open "monomer$i.pdb" r]
	set newfile [open "protein.pdb" a+]
	puts -nonewline $newfile [read $oldfile]
	close $oldfile
	close $newfile
	}
	}

proc cat_File { no } {
for { set i 1 } { $i <= $no } { incr i } {
grep_get "ATOM" "seg$i.pdb" "monomer$i.pdb"
}
grep_get "END" "seg1.pdb" "monomer$i.pdb"
cat_File2 $i
}
	proc prep_File { inputpdb segment} {
	mol load pdb $inputpdb
	foreach S $segment {
	set seg [atomselect top "segname $S and chain $S "]
	$seg writepdb seg$S.pdb
	$seg delete
	}
	mol delete all
	}

Set of command libraries for generating OXY and NIC PsDAs

```
proc make_Pseudo { no name bound min max} {
        set out [open $name.pdb w]
        for { set i 1 } { $i <= $no } { incr i } {
puts $out [format "%s %6d %3s %3s x%4d %3d.000 %3d.000 %3d.000 0.00 0.00 " "ATOM" $i $name $name $i [rand_range [expr -1*$bound] $bound ] [rand_range [expr -1*$bound] $bound ] [rand_range $min $max ]]
}
close $out
}
```

```
proc rand_range { min max } {
return [expr int(rand() * ($max - $min)) + $min]
}
```

```
proc catNIC { } {
        for { set i 1 } { $i <= 2 } { incr i } {
        set oldfile [open "temp$i.pdb" r]
        set newfile [open "NICtemp.pdb" a+]
        puts -nonewline $newfile [read $oldfile]
        close $oldfile
        close $newfile
        }
}
```

```
proc make_Pseudo2 { no name x1 y1 min_up max_low } {
        if {$name == "OXY" } {
                mol load pdb $name.pdb
        set a [atomselect top "name $name and abs(x) > $x1 or abs(y) > $y1 "]
                $a writepdb 2$name.pdb
                mol delete all

              mol load pdb 2$name.pdb
                set b [atomselect top "index < $no "]
            $b set chain O
            $b set segname O
                $b writepdb $name.pdb
                mol delete all
                file delete 2$name.pdb


        } elseif { $name == "NIC" } {
        mol load pdb $name.pdb
        set a [atomselect top "name $name and (abs(x) > $x1 or abs(y) > $y1) and z > [expr $min_up - 5 ] "]
                $a writepdb 3$name.pdb
                mol delete all

            mol load pdb 3$name.pdb
            set b [atomselect top "index < [expr $no/2] "]
            $b writepdb part1$name.pdb
            mol delete all



            mol load pdb $name.pdb
            set a [atomselect top "name $name and (abs(x) > $x1 or abs(y) > $y1) and z < [expr $max_low + 5 ] "]
```

```
        $a writepdb 4$name.pdb
        mol delete all

        mol load pdb 4$name.pdb
        set b [atomselect top "index < [expr $no/2] "]
        $b writepdb part2$name.pdb
        mol delete all



            grep_get "ATOM" "part1$name.pdb" "temp1.pdb"
            grep_get "ATOM" "part2$name.pdb" "temp2.pdb"

            catNIC
            mol load pdb NICtemp.pdb
        set all [atomselect top "all"]
            $all set chain N
            $all set segname N
        $all writepdb NIC.pdb
        mol delete all
            file delete NICtemp.pdb
            file delete 2$name.pdb
            file delete 3$name.pdb
            file delete 4$name.pdb
            file delete part1$name.pdb
            file delete part2$name.pdb
            file delete temp1.pdb
            file delete temp2.pdb
}
}
```

Set of command libraries for generating protein coordinate, topology files, fix atom file and secondary restrain file.

```
proc gen_Psf {ep1 patchep1 ep2 patchep2
ep3 patchep3 topo segment} {
package require psfgen
topology $topo
pdbalias residue HIS HSD
pdbalias atom ILE CD1 CD
pdbalias atom HOH O OH2
pdbalias residue HOH TIP3

        foreach S $segment {
                segment $S {
                pdb seg$S.pdb
                }
        coordpdb seg$S.pdb $S
#       patch GLUP $S:71

        foreach i $patchep1 {
                patch $ep1 $S:$i
        }

        foreach i  $patchep2 {
                patch $ep2 $S:$i
        }

        foreach i $patchep3 {
                patch $ep3 $S:$i
        }

        }

segment K {
pdb pota.pdb
}
coordpdb pota.pdb K


segment O {
pdb OXY.pdb
}
```

```
proc move2origin { fixatom    } {
mol load pdb proteininp.pdb
set all [atomselect top all]
set mov [atomselect top "not name OXY
and not name NIC and not name POT"]
set m [measure center $mov weight mass]
set g [vecinvert $m]
set fix [atomselect top "$fixatom"]
$mov moveby $g
$all set occupancy 0
$all set beta 0
$all writepdb proteininput.pdb
$fix set beta 1
$all writepdb proteininputfix.pdb
mol delete all
}
```

```
proc 2ndrestrain { selected  } {
mol new proteininput.psf
mol addfile  proteininp.pdb
package require ssrestraints
ssrestraints -psf proteininput.psf -pdb
proteininp.pdb -o extrabonds.txt -sel
$selected
mol delete all
#exit

}
```

```
proc delfile {seg noseg } {
        foreach a $seg {
        puts "seg$a.pdb"
        file delete seg$a.pdb
        }
        for { set i 1 } { $i <= $noseg  } { incr i
} {
        file delete monomer$i.pdb
        file delete seg$i.pdb
        }
        file delete tmer$i.pdb
```

```
coordpdb OXY.pdb O



segment N {
pdb NIC.pdb
}
coordpdb NIC.pdb N



guesscoord
writepdb proteininp.pdb
writepsf proteininput.psf
#exit
}
```

```
        file delete  proteininp.pdb
        file delete protein.pdb
        file delete NIC.pdb
        file delete OXY.pdb
}
```

Set of command libraries for applying the tclBC and generating the bubble.tcl

```
proc get_psue_id  { re args } {
      set files [eval glob -types f $args]
      foreach file $files {
      set fp [open $file]
      while { [gets $fp line] >= 0} {
            if [regexp -- $re $line] {
                  if {[llength $files] > 1} {puts -nonewline $file:}
                        return [lindex $line 1]
                        break
            }
     }
      close $fp
      }
}
```

```
proc rest_layer { file noNIC  }  {
 set NICid        [get_psue_id "NIC" proteininput.pdb]
 set OXYid        [ get_psue_id "OXY" proteininput.pdb]
 set text_file "
wrapmode cell
# Two first agruments of calcforces are automatically forwarded
# to it by NAMD. The other 4 arguments match the list of 4 values
# from command tclBCArgs.
```

```
foreach { x0 y0 z0 } \$bubbleCenter { break }

proc calcforces {step unique Rstart Rtarget Rrate K} {
  global x0 y0 z0  ;# defined in tclBCScript{ ... }

  # increase R, starting from \$Rstart, by \$Rrate at each step,
  # until it reaches \$Rtarget; then keep it constant

  set R \[expr \$Rstart + \$Rrate * \$step]
  if { \$R > \$Rtarget } { set R \$Rtarget }

  # get the components of the bubble center vector

  if { \$step % 200 == 0} { cleardrops}
  # pick atoms of the given patch one by one
  while { \[nextatom]  } {
          set atomid \[getid]
          #OXY atom
          if {  \$atomid > [expr $OXYid -1 ] && \$atomid < $NICid } {
          set rvec \[getcoord] ;# get the atom's coordinates
          foreach { x y z } \$rvec { break } ;# get components of the vector
             # find the distance between the atom and the bubble center
          # (long lines can be broken by a backslash and continued
          # on the next line)

          #set rho \[expr sqrt((\$x-$\x0)*(\$x-\$x0) + (\$y-\$y0)*(\$y-\$y0) + \
          #(\$z-\$z0)*(\$z-\$z0))]

          set rho \[expr sqrt((\$z-\$z0)*(\$z-\$z0))]

          # if the atom is inside the sphere, push it away radially
          set roxy \[expr \$R + 2 ]
           if { \$rho > \$roxy } {
                    #set forceX \[expr -1*\$K * (\$x-\$x0) / \$rho]
                    #set forceY \[expr -1*\$K * (\$y-\$y0) / \$rho]
                    set forceZ \[expr -1*\$K * (\$z-\$z0) / \$rho]
                    addforce \"0.0 0.0 \$forceZ\"
             } elseif { \$rho < \$R  } {
             dropatom
            }
   } elseif { \$atomid > [expr $NICid -1 ] && \$atomid < [expr $NICid + $noNIC ]  } {
```

```
        #nic Atom
        set rvec \[getcoord] ;# get the atom's coordinates
        foreach { x y z } \$rvec { break } ;
         set rho \[expr sqrt((\$z-\$z0)*(\$z-\$z0))]
        set rnic \[expr \$R - 2 ]
         if { \$rho < \$rnic } {
                #set forceX \[expr -1*\$K * (\$x-\$x0) / \$rho]
                #set forceY \[expr -1*\$K * (\$y-\$y0) / \$rho]
                set forceZ \[expr \$K * (\$z-\$z0) / \$rho]
                addforce \"0.0 0.0 \$forceZ\"
           } elseif { \$rho > \$R } {
          dropatom
          }
  } else  {
        dropatom ;# no longer consider this atom until \"cleardrops\"
        }
}
}
"
set fb [open $file w]
puts $fb $text_file
close $fb
}
```


Set of command libraries for generating NAMD configuration file.

```
proc gen_conf { MySystem  conf } {
set   text_file1   "
#######################################################
#######################################################
############namd configuration file ##################
structure          ./${MySystem}.psf
coordinates         ./${MySystem}.pdb
set ctonnb          9
set ctofnb          10
set pairdis         12
set temperature     310
set outputname      ${MySystem}
firsttimestep       0
#####################################################################
## SIMULATION PARAMETERS                              ##
```

```
#####################################################################
# Input
paraTypeCharmm          on
parameters              param19-epr.inp
COMmotion               no
temperature             \$temperature
# tclBC
tclBC on
tclBCScript \{
   set bubbleCenter    \"0.0 0.0 0.0 \"
   set tclBCScript      buble.tcl
   source \$tclBCScript
\}
tclBCArgs \{0. 15. .01 5.\}


# Force-Field Parameters
exclude                 scaled1-4
1-4scaling              1.0
cutoff                  \$ctofnb
nonbondedScaling        1.0
switching               on
switchdist              \$ctonnb
pairlistdist            \$pairdis
pairlistsPerCycle       2
margin                  0.0
#fixed atom
fixedAtoms on
fixedAtomsForces off
fixedAtomsFile ./${MySystem}fix.pdb
fixedAtomsCol B
#2nd stcr restraint
extraBonds on
extraBondsFile extrabonds.txt
# Integrator Parameters
timestep                2.0  ;# 2fs/step
rigidBonds              all  ;# needed for 2fs steps
nonbondedFreq           1
useSettle               on
fullElectFrequency      2
stepspercycle           20
```

```
#############################################################
## Boundary Conditions                                     ##
#############################################################
# Periodic Boundary Conditions
cellBasisVector1      100 0   0
cellBasisVector2      0   100 0
cellBasisVector3      0   0   100
cellOrigin           0.00  0.00  3.87
XSTfile              \$outputname
XSTfreq               2000
wrapAll               on
#############################################################
## EXECUTION SCRIPT                                        ##
#############################################################
# Output
outputName             \$outputname
binaryrestart         yes
restartfreq           1000     ;# 500steps = every 1ps
DCDfreq               500
outputEnergies         500
outputPressure         500
outputtiming           500
# Minimization
minimize              2000
reinitvels            310
run                  500000 ;#1  ns  310K
"
set fb  [ open $conf w ]
puts $fb $text_file1
close $fb
}
```

To run the script file, type the command:

vmd –dispdev text –e run.tcl

After executing run.tcl, a total of six output files were generated for the next run with NAMD simulation.

proteininput.pdb: a protein coordinate file or *pdb* file that serves as the initial coordinate of all atoms in the system in the Cartesian coordinate.

proteininput.psf: the protein structure file or *psf* file. This file is generally used in accompany with the *pdb* file to run the simulation on NAMD. It tells NAMD that which atoms in a *pdb* file are bonded together.

proteininputfix.pdb: almost similar to the proteininput.pdb. It differs in the beta column. The value of 1 is for fixed atoms and 0 for non-fixed atoms.

namd.conf: a NAMD configuration file used in MD simulation.

bubble.tcl: a file containing script commands for maintaining the boundary for the movement of NICs and OXYs during the simulation via tclBC (tcl boundary condition) command. This file is added in the namd.conf.

extrabond.txt: files for restraining the secondary structure of the protein via extrabond commands.

To run a simulation, type the following command

namd2 +p8 namd.conf > namd.out

proteininput.dcd: an output file containing MD trajectory.


```
>Main< (runfile) 54 % ls
.:
bubble.tcl                  extrabonds.txt              namd.conf                   namd.out
namd                        param19-epr.inp             proteininput                proteininput.BAK
proteininput.coor           proteininput.coor.BAK       proteininput.dcd            proteininput.dcd.BAK
proteininput.pdb            proteininput.psf            proteininput.restart.coor   proteininput.restart.coor.old
proteininput.restart.vel    proteininput.restart.vel.old proteininput.restart.xsc   proteininput.restart.xsc.old
proteininput.vel            proteininput.vel.BAK        proteininput.xsc            proteininput.xsc.BAK
proteininputfix.pdb         rmsd_all.dat                rmsdall.tcl
```

Figure 3.2 Output files from NAMD

3.2 Amount of CPU time

3.2.1 The effect of tclBC and the modification of bubble.tcl

For a typical calculation of pair interactions, an amount of CPU time in the MD simulation depends on the number of atoms. This calculation took a lot of time because the *bubble.tcl* script has to verify whether NICs and OXYs positions are in the boundary for every step of simulation.

To speed up the computational performance, the modification of the *bubble.tcl* by incorporating *dropatom* and *cleardrops* script commands was introduced[12]. This very differs from the original *bubble.tcl* because all atoms in pdb file will be picked up at the first time and if the current atom is not OXY or NIC that

atom will not be taken into consideration and assigned as "dropped atom". If the OXY or NIC are in their layer it will be assigned as dropped atom. All dropped atoms will not be picked up to the calculation loop (in tclBC) until the number of step reach the number that specified in the *cleardrops* script. The *cleardrops* is always used in accompany with the condition loop. For example "*if { $step % 200 == 0} { cleardrops}* ", the *bubble.tcl* executes all atoms in every 200 steps and tclBC can omit these dropped atoms during these steps (Figure 3.3). The concept introducing the dropped atoms is similar to the neighbor list technique used in MD simulation.

The comparison of the simulation time between the original and the modified *bubble.tcl* is shown in Table 3.1. The simulations were carried out for 50,000 steps. The result shows that in case of a total of 300 OXY and NIC atoms the total CPU time in modified *bubble.tcl* is reduced by 39.2 % with respect to the original one. And the % CPU time reduction increases as the number of atoms increases. The computational efficiency is greater for the system with a larger size because there are more omitted atoms in the system. Note that even the % CPU time reduction increases, the time of CPU usage in the larger scale is still longer than the smaller scale.

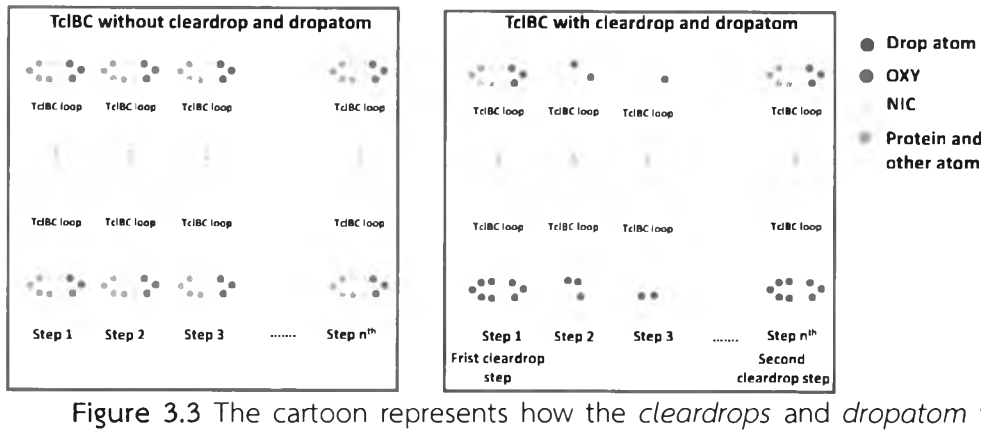

Figure 3.3 The cartoon represents how the *cleardrops* and *dropatom* work. The tclBC A. without *cleardrops* and *dropatom* commands. All atom in system will be considered in tclBC loop every step. B. with the *cleardrops* and *dropatom* command. All atoms will be considered only the step that set as the *cleardrops* step and the atom labelled *dropatom* will not be considered for the other.

Table 3.1 The comparison of the simulation time spending.

data	tclBC (bubble.tcl)	Number of OXY and NIC PsDAs	Time (min)	% CPU time reduction
1	original	300	4.36	-
2	modified	300	2.65	39.2
3	original	600	6.44	-
4	modified	600	3.21	50.2
5	original	1200	11.29	-
6	modified	1200	3.90	65.5

3.2.2 Number of OXY and NIC PsDAs.

In a PaDSAR simulation, there are three groups of atoms in the system. The first group is protein atoms. The second is EPs pseudoatoms. And the last group is OXY and NIC pseudoatoms. Number of atoms of the first two groups depends on the protein size that is a total number of residues used in the calculation. It, therefore, remained constant during simulation. Number of OXY and NIC can be varied. Four NAMD-PaDSAR simulations were performed by setting the number of OXY and NIC to 300, 600, 1200 and 2400. This corresponds to the ratio of OXY:NIC: residue of 0.74:0.74:1, 1.47:1.47:1, 2.94:2.94:1 and 5.88:5.88:1, respectively (total number of residues of KcsA = 102 x 4 = 408). Each model were regenerated and re-run for ten times.

The averaged RMSD calculated from the last 10 steps for 4 x 10 samples is shown in Figure 3.4. The RMSD from the results, all runs gave structure with RMSD with respect to the native structure lower than 3Å. This suggests that NAMD-PaDSAR does not significantly distort the structure. Comparing overall performance (CPU and RMSD results) among these dataset, the simulation with 300 OXY and NIC atoms gave the best choice. Therefore 300 atoms will be used in the subsequent test.

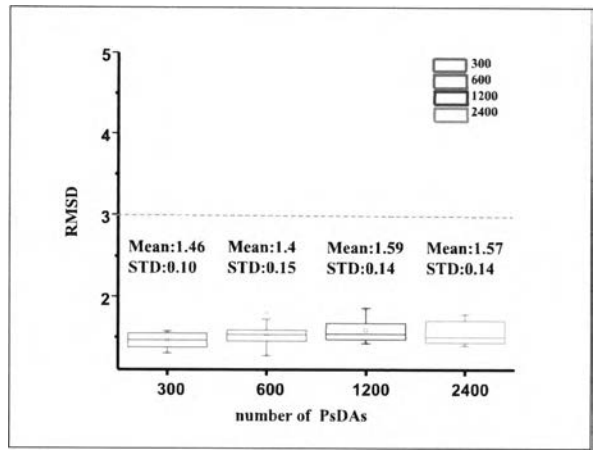

Figure 3.4 The box chart shows the average RMSD of 4 models.

3.2.3 The number of simulation step.

For the refolding simulation, NAMD-PaDSAR was performed with eight distorted KcsA structures. First four datasets were set to 1,000,000 steps (2ns) and the last four datasets were performed for 50,000 steps. The RMSD results suggest that the protein systems reach equilibrium and are well-behaved. (Figure 3.5 A). It appears that the number of step can be set up at 50,000 steps for the simulation to reach the equilibrium (Figure 3.5 B).

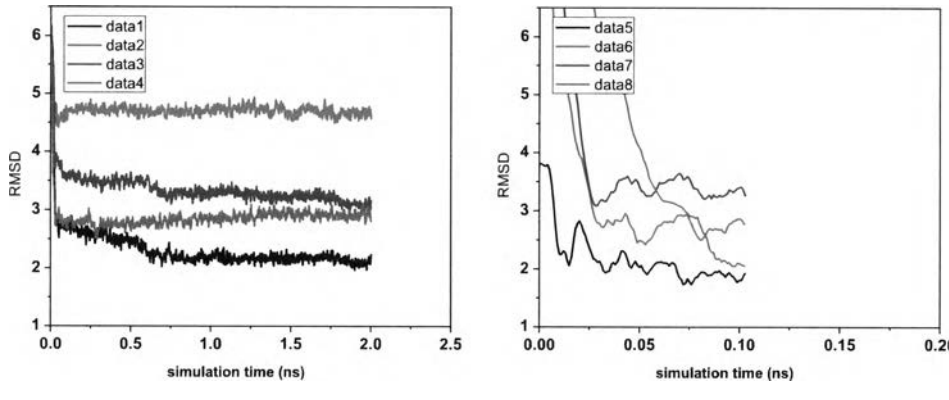

Figure 3.5 The RMSD for 8 data sets A. 1,000,000 steps B. 50,000 steps.

3.3 The radial distribution function of pseudoatoms.

An analysis of radial distribution function (RDF) is a good measure of how well the restrained potentials are utilized. In CHARMM-PaDSAR, three types of Lennard-Jones (LJ) like interactions were previously introduced. However, due to the limitation of NAMD-PaDSAR, only two types of LJ like potentials that are type I and type III can be implemented in this study. The type I refers to the favor interaction as the EP-type being in the favor environment whereas the type III indicates unflavored interaction. The matching pair of pseudoatoms of type I includes EP2-NIC and EP3-OXY.For type III, the repulsive interaction potential is computed due to mismatching pairs which include the following pair: EP1-OXY, EP2-OXY, EP1-NIC, EP1-OXY, EP1-PVP, EP2-PVP and EP3-PVP. It should also be noted that the EP1-PVP interaction which was the type II potential in the previous study was computed using the type III potential.

The radial distribution function plots of type I show the single sharp peak in EP2-NIC and EP3-OXY RDF at roughly 2 Å (black line), indicating that EP2 and EP3 particles position in the correct surrounding environments (water and lipid respectively). In type III, the RDF plots of EP1-NIC, EP1-OXY, EP2-OXY and EP3-NIC revealed a similar pattern (Figure 3.6). No apparent peak found for the RDF of EP1-NIC and EP1-OXY suggested that EP1 particles are not accessible by NIC and OXY, implying that they are inside the protein. An interpretation for the EP2-OXY and EP3-NIC RDF is that no EP2 particles are positioned in the lipid membrane region whereas none of EP3 is outside the membrane. In addition, the evidence of the EP2-PVP and EP3-PVP RDF plots revealed that EP2 and EP3 particles are located on the solvent exposed surface. A peak at 6 Å in the RDF plot of EP1-PVP, EP2-PVP and EP3-PVP is due to the pseudo-bond with a bond-length of 6Å defined the force field. Based on the RDF results, one can conclude that the simulation with NAMD-PaDSAR are capable of maintaining membrane and non-membrane parts of protein structure with the appropriate orientation in both water and membrane environment.

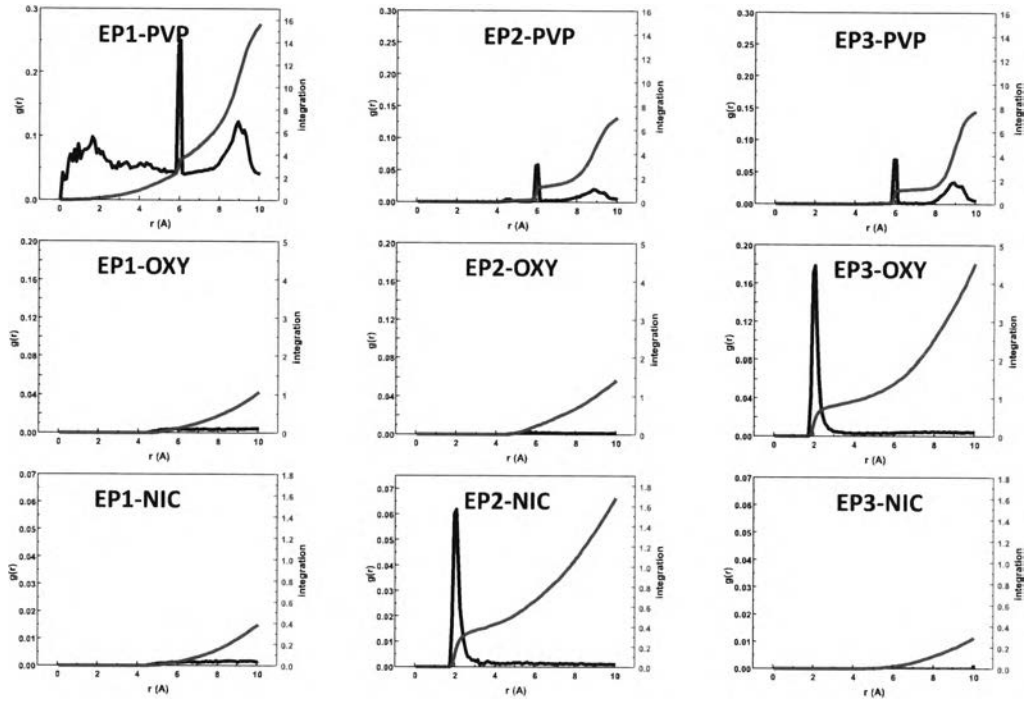

Figure 3.6 The radial distribution function plot, g(r) black line, and its integration, blue line, of nine pairs of pseudoatoms.

3.4 Varying the Improper dihedral force constant (K_{imp}) of EPs atom.

According to the concept of PaDSAR, the covalent bond between a EP atom and PVP (which is overlaid on top of the α-carbon atom of amino acid residue) is defined as a rigid bond. A strong interaction from the LJ potential between OXY or NIC and EPs atoms can result in local structure distortion of the protein. The improper dihedral angle of EP atoms (Cα-N-CO-EPs) are defined at 0 degree in order to keep the EP atom on the N-Cα-CO plane. Apparently, this is not always the case for the nitroxide spin as the flexibility of the spin labeled sidechain which consists of 5 rotatable bonds (χ_1, χ_2, χ_3, χ_4, and χ_5). Thus, the force constant of this improper dihedral angle, K_{imp} has been modified by taking into account the flexibility of the nitroxide sidechain. To define the appropriate value of K_{imp} , five runs of NAMD-PaDSAR were performed with different K_{imp} values corresponding to 10, 25, 55 and 70 kcal mol^{-1} rad^{-2} respectively (Figure 3.7). It appears that the case of K_{imp} = 10 kcal mol^{-1} rad^{-2} gives the lowest RMSD in almost all cases even in case of distorted protein cannot be refolded back to the native structure (RMSD more than 3.0 Å).

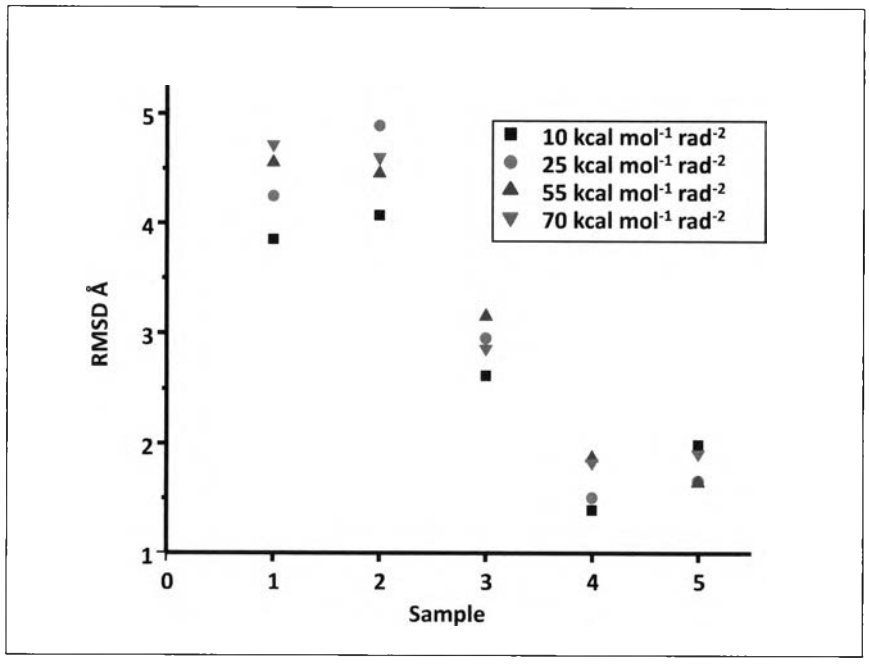

Figure 3.7 The RMSD of five samples with varying the improper dihedral force constant

To demonstrate the fluctuation of improper dihedral angles caused by varying K_{imp} of the EP atoms in the different environment, amino acid residues of KcsA in sample number 4 were classified into water-soluble (WAT), membrane (MEM) and water-membrane interfacial (INT) residues (Figure 3.8 and Table 3.2). WAT residues are residues that exposed to solution either extracellular or intracellular side of the lipid bilayer. With a common representation of membrane protein orientation, WAT residues are defined by the z-position in which every atoms of the WAT residue are $z > 15$ or $z < -15$ from the origin in Å unit. MEM residues are embedded in the membrane region where every atoms of MEM residues position $-15 > z > 15$ Å. INT residue is defined in a way that some of residue atoms are located in either water or membrane region.

For more deeply details, amino acid residues were distinguished by the solvent accessibility (SA) into three groups (Table 3.3), high, medium and low. The high group represents the residues that highly exposed to the solvent (SA > 0.51 -1.0) and expected that it is the main part of interaction between EP atoms and NIC and OXY PsDAs. The medium is the moderately exposed group and the low group is almost buried in the structure.

Table 3.2 Classification of amino acid residues by the position in environments.

Environment	Amino acid residue				
Solution	SER22	ALA23	LEU24	HSD25	GLU51
	ARG52	GLY53	ALA54	PRO55	GLY56
	ALA57	GLN58	ILE60	THR61	VAL84
	ALA111	THR112	TRP113	PHE114	VAL115
	GLY116	ARG117	GLU118	GLN119	GLU120
	ARG121	ARG122	GLY123	HSD124	
Membrane	ALA29	GLY30	ALA31	ALA32	THR33
	VAL34	LEU35	LEU36	VAL37	ILE38
	VAL39	LEU40	LEU41	ALA42	GLY43
	SER44	TYR45	LEU46	ALA47	ALA65
	LEU66	TRP67	TRP68	SER69	VAL70
	GLU71	THR72	ALA73	THR74	THR75
	VAL76	GLY77	TYR78	TRP87	GLY88
	CYS90	VAL91	ALA92	VAL93	VAL94
	VAL95	MET96	VAL97	ALA98	GLY99
	ILE100	THR101	SER102	PHE103	GLY104
	LEU105	VAL106	THR107		
Interface	TRP26	ARG27	ALA28	VAL48	LEU49
	ALA50	LEU59	TYR62	PRO63	ARG64
	GLY79	ASP80	LEU81	TYR82	PRO83
	THR85	LEU86	ARG89	ALA108	ALA109
	LEU110				

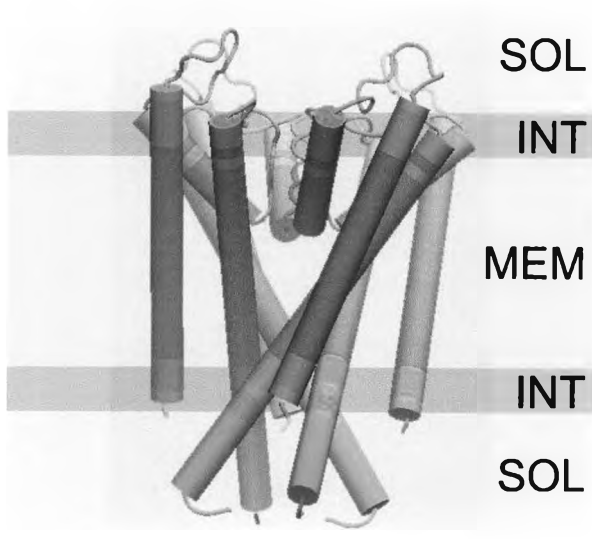

Figure 3.8 The cartoon shows classification of amino acid residues into three types related with environments and defined by the distance in z direction (Å) form the origin coordinate. SOL: solution ($z > 15$ or $z < -15$), MEM: membrane ($-15 < z < 15$) and INT: interface (residue that overlaps both SOL and MEM).

The results in Figure 3.9 show that the degree of solvent accessibility and the position of EP atoms in different environment region do not affect to the fluctuation of an improper dihedral angle during the simulation when compare the same value of K in each data set and no significantly different between EP atoms located in low, medium and high solvent accessibility as same as the EP atoms located in different environment region. Contrast with two factors above the variation of K is directly effect to fluctuation of EP atoms, 70 (blue), 55 (red), 25(green) and 10 (black) kcal $mol^{-1}rad^{-2}$. A low value of K allows a more fluctuation of the EP atoms. It appears that refolding of decoy proteins to the native structure is related with the fluctuation of the EP atoms. Because of the flexibility of the spin-labeled side chain allowing EP atoms to fluctuate might be more capable to explain the real system.

Table 3.3 Classification of amino acid residues by the solvent accessibility.

Solvent accessibility	Amino acid residue				
Low (0.00-0.20)	ALA23	LEU24	HSD25	ALA28	ALA29
	ALA32	LEU36	VAL39	LEU40	GLY43
	SER44	ALA47	VAL48	ALA50	GLU51
	ALA54	ALA57	LEU59	ALA65	TRP67
	TRP68	SER69	VAL70	GLU71	THR72
	ALA73	THR74	THR75	VAL76	GLY77
	TYR78	GLY79	ASP80	LEU81	PRO83
	GLY88	ARG89	VAL91	ALA92	VAL95
	MET96	ALA98	GLY99	ILE100	SER102
	PHE103	GLY104	LEU105	VAL106	THR107
	ALA108	ALA109	LEU110	ALA111	THR112
	PHE114	VAL115	GLY116	GLU118	GLN119
Medium (0.21 – 0.50)	TRP26	GLY30	THR33	VAL37	ALA42
	LEU46	ILE60	THR61	TYR62	ARG64
	LEU66	TYR82	VAL84	THR85	CYS90
	VAL93	VAL97	THR101	TRP113	ARG117
	ARG122	HSD124			
High (0.51-1.00)	ARG27	ALA31	VAL34	LEU35	ILE38
	LEU41	TYR45	LEU49	ARG52	GLY53
	PRO55	GLY56	GLN58	PRO63	LEU86
	TRP87	VAL94	GLU120	ARG121	GLY123

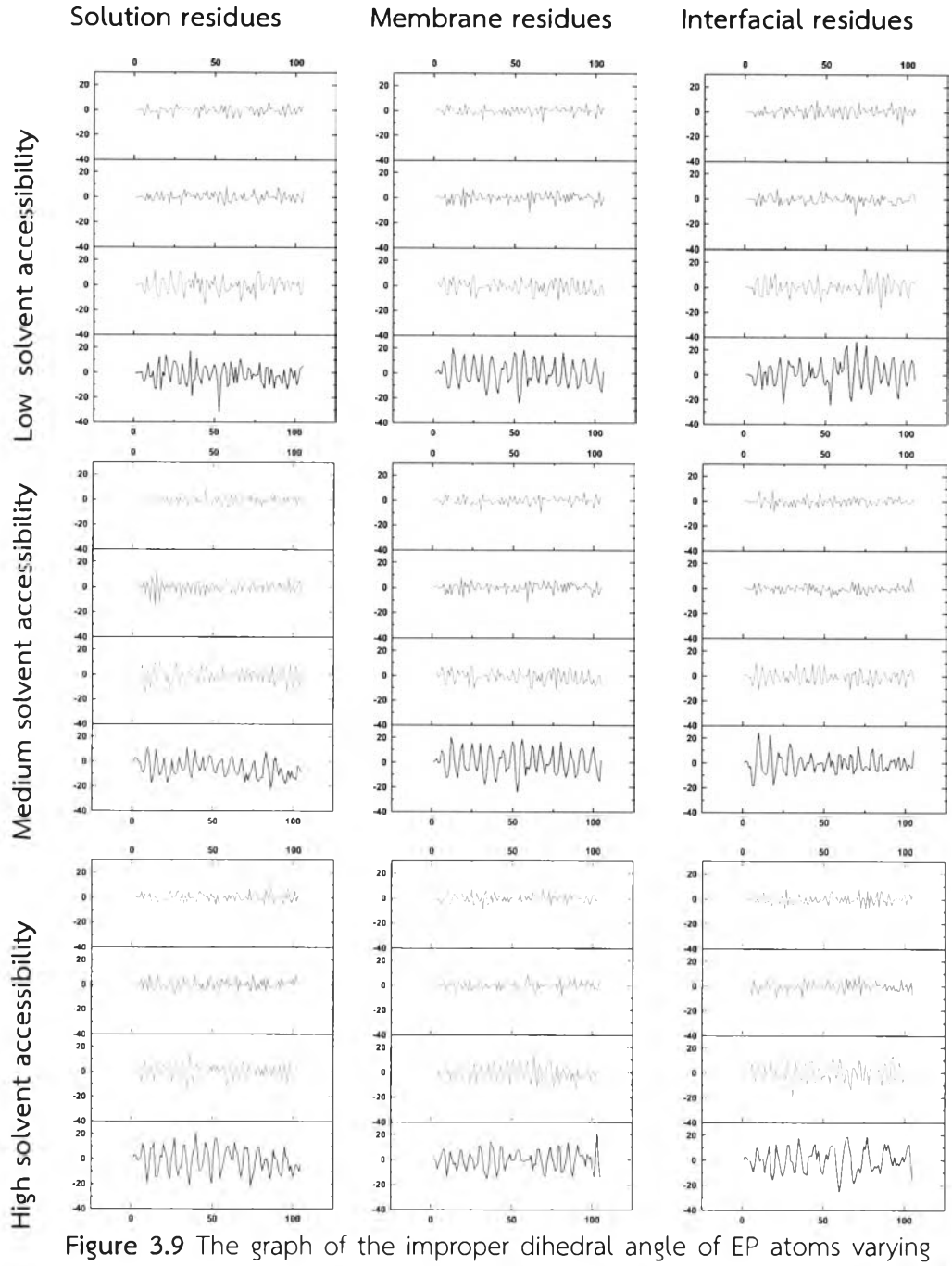

Figure 3.9 The graph of the improper dihedral angle of EP atoms varying K values during the simulation, 70 (blue), 55 (red), 25(green) and 10 (black) kcal mol^{-1} rad^{-2} in different positions related with the environment (column) and solvent accessibility (row). y-axis is the improper dihedral angle and x-axis is the simulation frame.

3.5 Refolding the decoy to the native conformation

This test used a set of distorted structures of KcsA as the starting decoy. To generate decoy structures, the TM1 and TM2 segments of the KcsA structure were forced to distort from its native conformation with RMSD in range of 3 – 13 Å using steered MD, giving rise to a total of 524 decoys. For refolding simulation of each decoy, NAMD-PaDSAR was performed using the best condition obtained from previous section. For each decoy, the refolding simulation was performed for 0.10 ns (50,000 steps) after 2000 steps of minimization.

Figure 3.10 shows the initial RMSD (x-axis) versus final RMSD (y-axis) for 524 decoys. Two shade of background represents the acceptable (orange, RMSD < = 3 Å) and unacceptable (blue, RMSD > 3 Å) areas. 433 out of 524 samples or about 82% can be refolded back to the native conformation using NAMD-PaDSAR.

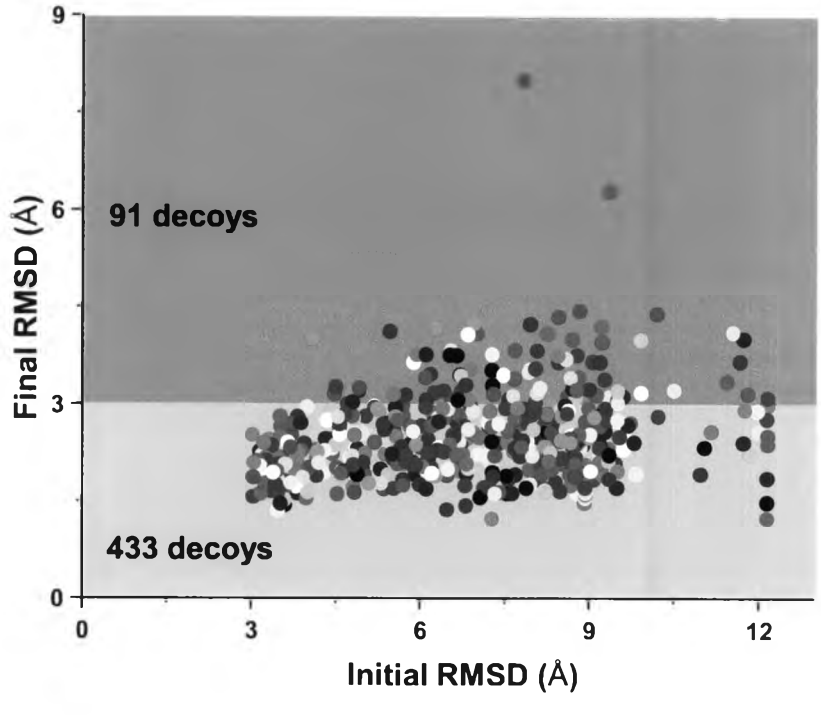

Figure 3.10 The graph shows the initial and final RMSD of 524 samples and the orange shade is the acceptable area for the final RMSD.

The percent refolding of an individual range were calculated and shown in Figure 3.11. In case of initial RMSD 3 ≤X < 4 Å, all 68 decoys or 100% can be refolded. 4 ≤ X < 5 Å, 54 out of 62 decoys or 87%. 5 ≤ X < 6 Å, 54 out of 59 decoys or 92%. 6 ≤ X < 7 Å, 56 out of 77 decoys or 72%. 7 ≤ X < 8 Å, 53 out of 67 decoys or 79%, 8 ≤ X < 9 Å, 94 out of 113 decoys or 83%. 9 ≤ X < 10 Å, 38 out of 53 decoys or 72%. 10 ≤ X < 11 Å, 2 out of 4 decoys or 50%, 11 ≤X < 12 Å, 6 out of 11 decoys or 55% and 12 ≤ X < 13 Å, 8 out of 10 or 80%. It appears that NAMD-PaDSAR is very effective for refolding the decoys that the initial RMSD in range 3 – 6 Å and also good for 6 – 10 Å. Although 10 -13 Å is quite good. More data samples are needed.

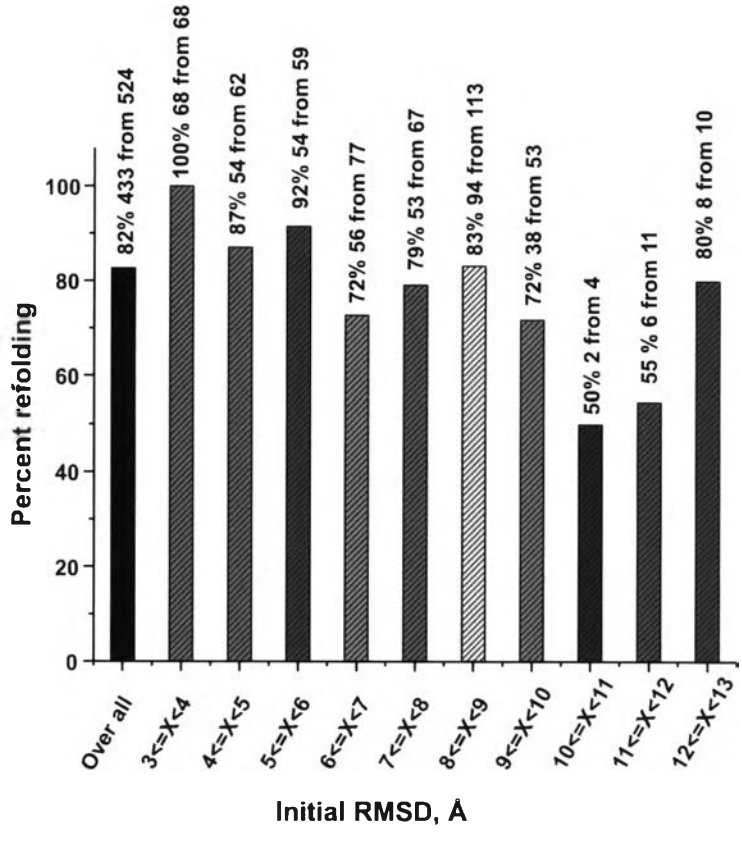

Figure 3.11 The bar graph shows the percent refolding. The first column is the overall decoys and the others are the percent refolding in an individual range.