Supporting Information
Accelerated CDOCKER with GPUs, parallel simulated annealing and fast Fourier transforms

Xinqiang Ding,[†] Yujin Wu,[‡] Yanming Wang,[‡] Jonah Z. Vilseck,[‡] and Charles L. Brooks III[*,‡,†,¶]

†*Department of Computational Medicine & Bioinformatics, University of Michigan*

‡*Department of Chemistry, University of Michigan*

¶*Biophysics Program, University of Michigan, Ann Arbor, Michigan 48109, United States*

E-mail: brookscl@umich.edu

S1 Details of calculating protein grid potentials in CDOCKER

As shown in Fig.1A in the manuscript, the binding pocket of a protein receptor is represented as a cubic grid. Both electrostatic potential and van der Waals interaction potentials of a protein are calculated on all the lattice points.

S1.1 Electrostatic grid potential

To calculate the electrostatic potential at the lattice point (l, m, n) whose coordinate is represented as $\vec{r}(l, m, n)$, a prob atom with a unit charge is placed at the lattice point (l, m, n) and the protein electrostatic potential at the lattice point (l, m, n), $V_{\text{elec}}^{\text{grid}}(l, m, n)$, is

the sum of the electrostatic interaction energy between the prob atom and all the protein atoms (excluding flexible side chain atoms if flexible receptor docking is used):

$$V_{\text{elec}}^{\text{grid}}(l,m,n) = \sum_{j \in P} \frac{1}{4\pi\epsilon} \frac{q_j}{|\vec{r}(l,m,n) - \vec{r}_j|}, \tag{S1}$$

where P is the collection of all protein atoms (excluding flexible side chain atoms if flexible receptor docking is used) and q_j is the charge of the protein atom j. When soft-core potentials are used, the protein electrostatic potential on the lattice point (l,m,n), $V_{\text{elec}}^{\text{soft-core}}(l,m,n)$, is calculated using the formula in Eq. 1, i.e.,

$$V_{\text{elec}}^{\text{soft-core}}(l,m,n) = \begin{cases} V_{\text{elec}}^{\text{grid}}(l,m,n), \text{ if } V_{\text{elec}}^{\text{grid}}(l,m,n) \leq \dfrac{E_{\max}}{2} \\ E_{\max} - a \cdot (|\vec{r}(l,m,n) - \vec{r}_j|)^b, \text{otherwise} \end{cases}, \tag{S2}$$

where $E_{\max}$ is the parameter controlling the "softness" of the electrostatic potential, which is given in Tables 1 and 2; a and b are constants such that both the potential $V_{\text{elec}}^{\text{soft-core}}(l,m,n)$ and the corresponding force are continuous with respect to the distance $|\vec{r}(l,m,n) - \vec{r}_j|$.

S1.2 van der Waals interaction grid potential

As discussed in the manuscript, in contrast to the electrostatic potential, which is represented using only one grid, the van der Waals interaction potential is represented using multiple grids. The grid $V_{\text{vdw}}^{r^{\min}}$ represents the van der Waals interaction potential of a prob atom with the specific Lennard-Jones potential radius $r^{\min}$. The value of $V_{\text{vdw}}^{r^{\min}}$ at the lattice point (l,m,n) is defined as

$$V_{\text{vdw}}^{r^{\min}}(l,m,n) = \sum_{j \in P} \sqrt{\epsilon_j} \left[\left(\frac{(r^{\min} + r_j^{\min})/2}{|\vec{r}(l,m,n) - \vec{r}_j|} \right)^{12} - 2 \left(\frac{(r^{\min} + r_j^{\min})/2}{|\vec{r}(l,m,n) - \vec{r}_j|} \right)^{6} \right], \tag{S3}$$

where P is the collection of all protein atoms (excluding flexible side chain atoms if flexible receptor docking is used); $r_j^{\min}$ is the Lennard-Jones potential radius of the protein atom j;

ϵ_j is the Lennard-Jones potential depth well of the protein atom j. Similarly, when soft-core potential is used, the van der Waals interaction potential on the lattice point (l, m, n), $V_{\text{vdw}}^{\text{soft-core-r}^{\min}}(l, m, n)$, is calculated using the formula in Eq. 1, i.e.,

$$V_{\text{vdw}}^{\text{soft-core-r}^{\min}}(l, m, n) = \begin{cases} V_{\text{vdw}}^{r^{\min}}(l, m, n), \text{ if } V_{\text{vdw}}^{r^{\min}}(l, m, n) \leq \dfrac{E_{\max}}{2} \\ E_{\max} - a \cdot (|\vec{r}(l, m, n) - \vec{r}_j|)^b, \text{otherwise} \end{cases}, \tag{S4}$$

where $E_{\max}$ is the parameter controlling the "softness" of the van der Waals interaction potential, which is given in Table 1 and Table 2; a and b are constants such that both the potential $V_{\text{vdw}}^{\text{soft-core-r}^{\min}}(l, m, n)$ and the corresponding force are continuous with respect to the distance $|\vec{r}(l, m, n) - \vec{r}_j|$.

The Lennard-Jones potential radius $r^{\min}$ of an atom depends on the atom's atom type. Because there are only finite number of atom types in the CHARMM and CGENFF force field, the number of possible values of $r^{\min}$ is also finite. In addition, the values of $r^{\min}$ can be very similar or even the same for different atom types. Therefore, a small set of $r^{\min}$ values are selected such that the $r^{\min}$ values of all atom types in both CHARMM and CGENFF force field can be well approximated using one value from the set. Based on the CHARMM c36 protein force field and CGENFF force field, the selected set of $r^{\min}$ is { 0.450, 0.900, 1.400, 1.800, 2.200, 2.520, 2.600, 2.680, 2.936, 3.126, 3.200, 3.300, 3.400, 3.500, 3.530, 3.600, 3.700, 3.800, 4.000, 4.100, 4.180, 4.300, 4.400, 4.480, 4.550, 4.600 }.

S2 Accelerated calculation of soft-core grid potentials using graphical processing units (GPUs)

The CUDA programming model was used in this project. Two GPU kernels were created and wrapped as external C functions to generate the protein grid and the ligand grid. The protein grid kernel computes the electrostatics grid for $V_{\text{elec}}^{\text{grid}}$ in Eq. S2 and V_{vdw} in Eq. S4.

The protein grid kernel uses a single CUDA thread to update a given grid point by looping through all protein atoms and performing the necessary calculations based on the properties of protein atoms. The ligand grid kernel is used to compute the complementary electrostatics grid for the charge term Q^{grid} in Eq. 4 of the main text and the complementary van der Waals grid for the well depth term $\sqrt{\epsilon_i}$ in Eq. 7 of the main text. The ligand grid kernel uses a single CUDA thread to calculate the grid of a given ligand conformation by looping through all ligand atoms and mapping the charge and van der Waals well depth term of each atom onto the corresponding grid points using trilinear interpolation. The ligand conformations were generated in CPU for simplicity with negligible additional computational cost.

S3 Computational details for rigid receptor docking using CDOCKER, DOCK, Autodock and Autodock Vina

S3.1 Preparation of ligand initial conformations

The rigid receptor docking was conducted with two ways of preparing ligand initial conformations. One way is to use a ligand's native conformation, which is the ligand's conformation when it binds with the receptor, as the initial conformation. The other way is to random conformations. Starting with a ligand's native conformation, its random conformations were obtained by randomly rotating the ligand's rotatable bonds (dihedral angles) using the `obrotamer` command from the `Open Babel` software. When the rigid receptor docking involves multiple trials, each trial uses a different ligand random conformation, sampled independently, as the initial conformation.

S3.2 Definition of docking grid boxes

All protein-ligand docking methods benchmarked in this study require a pre-specified grid box. The grid box defines the binding pocket inside which a ligand binds with the receptor.

The grid box also defines the search space for the searching algorithm identifying the ligand pose with the best score. Therefore, the definition of grid boxes can significantly affect a docking method's performance. To make a fair comparison between different docking methods, we used the same grid box across all the docking methods presented in this study.

For a protein-ligand complex, specifying a grid box requires defining the position of its center and its size along the x, y and z dimensions. The center of its grid box is defined as the center of the ligand's native pose, i.e., the coordinate of the center is equal to the mean of the coordinates of all ligand atoms in the ligand's native pose. We used cubic grid box for all docking methods, so the sizes of the grid box along x, y, and z directions are the same. To define the size of the grid box, we first computed the sizes of the ligand along x, y, and z directions in the ligand's native pose. Then the size of the cubic grid box is defined to be the largest size of the ligand along x, y, and z directions plus 5Å. In addition, if the size of the cubic grid box calculated above is less then 21Å, the size of the cubic grid box is set to 21Å. Therefore, the size of the cubic grid box is always greater than or equal to 21Å.

S3.3 Final docking pose and calculation of docking accuracy

In CDOCKER, multiple trials of MD-based simulated annealing were used to search for docking poses. Each trial generates a candidate docking pose, which is the ligand pose after MD-based simulated annealing and energy minimization. Candidate docking poses from all trials are ranked based on the energy which is the sum of the intra interaction energy of a ligand and the ligand's interaction energy with the receptor. The candidate pose with the lowest energy is used as the final docking pose. In DOCK, Autodock, and Autodock Vina, 10 candidate ligand poses are generated and scored. Similarly, the candidate ligand pose with the best score is used as the final docking pose.

For all protein-ligand docking methods presented in this study, the docking accuracy is calculated based on the final docking pose. The docking accuracy is calculated as the percentage of protein-ligand complexes in benchmark datasets for which the root mean square

deviation (RMSD) of the final docking pose is less than 2.0 Å with respect to the native ligand pose.

S4 Computational details for flexible receptor docking using CDOCKER

S4.1 Preparation of ligand initial conformations

Ligand random conformations were used as initial conformations in flexible receptor docking. The same procedure as used in rigid receptor was used to generate ligand random conformations.

S4.2 Definition of docking grid boxes

A similar criteria as used in rigid receptor docking was used to define the position and the size of docking grid boxes in flexible receptor docking. The only difference is that the size of the cubic grid box is defined to be the largest size of the ligand along x, y, and z directions plus 10 Å. Similarly, if the size of the cubic grid box is less than 21 Å, it is set to 21 Å.

S4.3 Grid generation for SEQ17 dataset

The apo-protein uses the same definition of the grid boxes and flexible side chains with the corresponding holo-protein. When generating the potential energy grids, these flexible side chains and the corresponding backbones are deleted. The rest of the protein is used to generate the potential energy grid.

S5 Examples of CHARMM script using the four new functionalities introduced in this study

S5.1 Generate protein grid potentials using GPUs

Calculating protein grid potentials using GPUs is invoked by the following two CHARMM commands `FFTG RADI` and `FFTG PGEN`:

```
FFTG RADI UNIT <integer>
  UNIT    Unit from which van der Waals radii of probe atoms are read.
          The unit number should point to a formatted file.
          An example file can be found at ./test/data/
          fftdock_c36prot_cgenff_probes.txt

  This command reads the values of van der Waals radii of probe atoms.
  These values are used for calculating protein grid potentials.
  For each radii, a corresponding grid potential will be calculated
  in the following command.

FFTG PGEN XMAX <float> YMAX <float> ZMAX <float> -
          XCEN <float> YCEN <float> ZCEN <float> -
          DGRI <float> -
          EMAX <float> MINE <float> MAXE <float> -
          SELE <atom selection> END -
          <RDIE|CDIE> EPSILON <float>

  XMAX,YMAX,ZMAX    length of the grid along X, Y, Z direction
  XCEN,YCEN,ZCEN    X, Y, Z coordinates of the grid center
```

```
    DGRI              grid spacing
    EMAX              the maximum value for the soft-core
                      van der Waals energy
    MINE              the minimum value for the soft-core (negative)
                      electrostatic energy
    MAXE              the maximum value for the soft-core (negative)
                      electrostatic energy
    <atom selection>  selection of atoms based on which potential
                      grids are calculated
    RDIE|CDIE         distance dependent dielectric | constant dielectric
    EPSILON           dielectric constant

    This command calculates the protein grid potentials on GPUs
    based on the above parameters specifying the dimension of
    grids and the softness of soft-core potentials.
```

S5.2 FFT-based docking

The following CHARMM script shows how to first generate protein grid potentials using GPUs and then apply FFT-based docking.

```
! read the topology and parameter files
read rtf card name @0/top_all36_prot.rtf
read rtf card name @0/top_all36_cgenff.rtf append
read param card name @0/par_all36_prot.prm
read param card name @0/par_all36_cgenff.prm append

! read protein
read sequ pdb name @0/t4l.pdb
```

```
gene PRO setup
auto angle dihe

read coor pdb name @0/t4l.pdb resid
ic fill
ic param
ic build
hbuild sele all end
PRINt COORdinate SELEct .NOT. INITialized END

! specify parameters for protein grid potentials
set xcen = 26.912
set ycen = 6.126
set zcen = 4.179
set emax = 2
set mine = -20
set maxe = 40
set space = 0.5
set xmax = 8
set ymax = 8
set zmax = 8

! read van der Waals radii
open unit 10 read form name @0/fftdock_c36prot_cgenff_probes.txt
fftg radii unit 10

! generate protein potential grid
```

```
fftg pgen xmax @xmax ymax @xmax zmax @xmax -
    xcen @xcen ycen @ycen zcen @zcen -
    dgrid @space emax @EMAX mine @MINE maxe @MAXE -
    sele segid pro end -
    rdie epsilon 3


! read ligand
stream @0/benzene.stream

! real ligand pdb
read sequ pdb name @0/benzene.pdb
generate LIGA setup
read coor pdb name @0benzene.pdb resid
print coor sele .not. init end

! copy the native coordinate to comparison coordinates
print coor selec segid LIGA end
coor copy comp

! translate ligand by a random amount
calc xoff = 10*?RAND
calc yoff = 10*?RAND
calc zoff = 10*?RAND
coor tran xdir @xoff ydir @yoff zdir @zoff selec segid LIGA end
print coor selec segid LIGA end
```

```
! ! read grid potentials
! open unit 30 read form name @0/fft_dock_grid.txt
! FFTG read UNIT 30 form

! FFT docking
FFTG LCON NCON 10 ICON 1 NROK 2 NQUA 1 IQUA sele segi LIGA end

! open unit 40 read form name @0/fftdock_rotation_1.qua
! FFTG LCON NCON 10 ICON 1 NROK 2 QUAU 40 sele segi LIGA end
! stop

! copy coordinate of docked ligand to main coordinate set
FFTG COOR ICON 1 IROT 1
print coor selec segid LIGA end

! calculated rmsd between docked structure and native structure
coor rms selec segid LIGA end comp

! the value of rms should be less than 0.5. It is about 0.373899
show ?rms
@testcheck ?rms 0.3739 0.01 FFT_DOCK
```

S5.3 Parallel simulated annealing on GPUs

The following CHARMM script shows how to use the parallel simulated annealing on GPUs for docking.

```
! read the topology and parameter files
read rtf card name "../charmm/toppar/top_all36_prot.rtf"
read rtf card name "../charmm/toppar/top_all36_na.rtf" append
read rtf card name "../charmm/toppar/top_all36_carb.rtf" append
read rtf card name "../charmm/toppar/top_all36_cgenff.rtf" append
read param card  name "../charmm/toppar/par_all36_prot.prm"
read param card  name "../charmm/toppar/par_all36_na.prm" append
read param card  name "../charmm/toppar/par_all36_carb.prm" append
read param card name "../charmm/toppar/par_all36_cgenff.prm" append
bomlev -2
stream ./ligand.stream
bomlev 0

! read protein
read sequ pdb name "protein_with_h.pdb"
generate PROT setup
read coor pdb name "protein_with_h.pdb" resid

! read ligand and native coordinates
bomlev -1
read sequ pdb name ./ligand.pdb
generate LIGA setup
bomlev 0
read coor pdb name ./ligand.pdb resi

! copy native coordinates to comparsion set
coor copy comp
```

```
update atom switch vswitch cutnb 22 cutof 20 cuton 18 rdie epsilon 3

!!!! FFT DOCK  !!!!
! read grid for fft dock
open unit 30 read unform name ./grid-emax-100-mine--100-maxe-100.bin
FFTG read UNIT 30
close unit 30

! FFT dock for each conformer
open unit 33 read form name "../charmm/test/data/fftdock_rotation_1.qua"
set numrot = 100
set i = 1
! fft dock
FFTG LCON NCON 1 ICON 1 NROK @numrot SIZB 100 QUAU 33 sele segi LIGA end
close unit 33

! Free the GPU device from FFT dock, such that
! the same GPU device can be used in the following OpenMM_DOCK
FFTG CLEA

! this will fill the ICB array for bond parameters,
! which are used in OpenMM_DOCK
update nbxmod 5 ctonnb 18 ctofnb 20 cutnb 22 vatom vswitch switch rdie epsilon 3

! read grid potential for OpenMM dock
open unit 31 read unform name ./grid-emax-0.6-mine--0.4-maxe-8.0.bin
open unit 32 read unform name ./grid-emax-3-mine--20-maxe-40.bin
```

```
OMMD grid UNIS 31 UNIH 32
close unit 31
close unit 32

! define the fixed and flexible parts of the system
define backbone -
   sele ( -
   segid prot .and. -
   (type n .or. type ca .or. type c .or. type o .or. type oct* -
   .or. type ha* .or. type hn .or. type ht*) -
   ) show end

define flex sele ( segid LIGA ) .or. -
       (.byres. (((segid LIGA .and. ( .not. hydrogen )) .around. 5) -
        .and. ( .not. hydrogen))) .and. ( .not. backbone )  show end


!define fix sele (.byres. flex) .and. (type ca) show end
define fix sele (.byres. flex) .and. backbone show end

! create OpenMM system and context
calc NumCopy = @numrot
OMMD build sele fix end sele flex end ncopy @NumCopy

! set initial coordinates for each copy of ligand
open unit 40 write card name ./dock_result.txt
echu 40
```

```
set idxrot = 1
label setcoor
    ! copy initial ligand coordinates from FFT dock results
    ! into the main coordinates
    FFTG COOR ICON 1 IROT @idxrot

    ! copy initial ligand coordinates from the main coordinates
    ! into OpenMM dock
    OMMD setc idxc @idxrot

    incr idxrot by 1
    if @idxrot .le. @numrot then
        goto setcoor
        endif

! run parallel simulated annealing
OMMD CGRS SOFT 1 HARD 0 emax 1.5 mine -10.0 maxe 20.0 eps 3
OMMD SIAN NSTE 3000 FIRST 300 FINAT 700 NHRQ 50 INCT 1

OMMD CGRS SOFT 1 HARD 0 emax 0.6 mine -0.4 maxe 8.0 eps 3
OMMD SIAN NSTE 14000 FIRST 700 FINAT 300 NHRQ 50 INCT -1

OMMD CGRS SOFT 0 HARD 1 emax 3 mine -20 maxe 40.0 eps 3
OMMD SIAN NSTE 7000 FIRST 500 FINAT 300 NHRQ 50 INCT -1

OMMD CGRS SOFT 0 HARD 1 emax 30 mine -200 maxe 400.0 eps 3
```

```
OMMD SIAN NSTE 3000 FIRST 400 FINAT 50 NHRQ 50 INCT -1

! hard grid potential
grid clear
open unit 30 read unform name ./grid-emax-100-mine--100-maxe-100.bin
grid read unit 30 select segid LIGA end
close unit 30
grid on

set idxcopy = 1
label LoopEnergy
    ! copy coordinate from OpenMM_dock results into the main coordinates
    OMMD COOR IDXC @idxcopy

    ! mini sd nstep 100
    ! mini conj nstep 200

    ! ener
    ! set dockener = ?ener
    ! set dockvdw = ?vdw
    ! set dockelec = ?elec

    ! read coor comp select segid LIGA end pdb name ./ligand.pdb
    ! coor rms select segid LIGA .and. ( .not. type *H* ) end
    ! echo 1gkc @dockener ?rms

    write coor pdb name ./output/protein_ligand_@idxcopy.pdb
```

```
    incr idxcopy by 1
    if @idxcopy .le. @numcopy then
        goto LoopEnergy
    endif
stop
```

4. Parallel simulated annealing on GPUs

The following CHARMM script shows how to use the parallel simulated annealing on GPUs for flexible docking.

```
* TITLE
* Created by Yujin Wu
* Flexible docking (Seq17) - Docking in grids - HOLO
* Maximum translation distance as 2 A
* Parallel SA with harder grid potential
* 2019/06/17
*

!-----------------------------------------------------------------------
! Prepare protein and ligand
!-----------------------------------------------------------------------

!! Prepare protein and define flexible side chain
stream "flexchain.inp"

!! Read the topology and parameter files for ligand
read rtf card name "../../Toppar/top_all36_carb.rtf" append
```

```
read rtf card name "../../Toppar/top_all36_na.rtf" append
read rtf card name "../../Toppar/top_all36_cgenff.rtf" append
read param flex card name "../../Toppar/par_all36_carb.prm" append
read param flex card name "../../Toppar/par_all36_na.prm" append
read param flex card name "../../Toppar/par_all36_cgenff.prm" append

stream "../ligandrtf"

!! Build ligand
read sequ pdb name "./ligand.pdb"
generate LIGA setup
auto angle dihe
read coor pdb name "./ligand.pdb" resid
write coor card select segid LIGA end name "ligand.crd"

!! Ligand energy
ener
inte select segid LIGA end ! Ligand energy
set ligaener = ?ener

!------------------------------------------------------------------------
! Modify protein and ligand
!------------------------------------------------------------------------

!! Dimensional analysis
coor stats select segid LIGA end
set xcen ?XAVE
```

```
set ycen ?YAVE
set zcen ?ZAVE

!! Modify
bomlev -1
delete atom sele .not. ( flexchain .or. segid LIGA ) end
bomlev 0
write coor card select segid PROT end name "flexchain.crd"

!! Define backbone
define backbone sele -
             type n .or. -
             type ca .or. -
             type c .or. -
             type o .or. -
             type oct* .or. -
             type ha* .or. -
             type hn .or. -
             type ht* .or. -
             None end
define fixsc sele segid PROT .and. backbone end
cons fix sele fixsc end

!! Define flexible side chain and ligand
define flexsc select all .and. (.not. fixsc) end
print coor select flexsc end
print coor select fixsc end
```

```
!---------------------------------------------------------------------
! Prepare OpenMM system
! Ligand # = # of ligand conformers * # of copy for each conformer
!---------------------------------------------------------------------

!! ICB array for bond parameters
!update nbxmod 5 ctonnb 8 ctofnb 10 cutnb 12 vatom vswitch swith rdie epsilon 3
update atom switch vswitch cutnb 12 ctofnb 10 ctonnb 8 vdwe elec rdie epsilon 3

!! Read grid potential for OpenMM docking
open unit 31 read unform name grid-emax-3-mine--20-maxe-40.bin
open unit 32 read unform name grid-emax-15-mine--120-maxe--2.bin
OMMD grid UNIS 31 UNIH 32
close unit 31
close unit 32

!! Set up numbers for loop
set idxnum = 1 ! index of loop num
set idxcopy = 1  ! index of ligand copy
calc numcopy = @num * @copy
OMMD build sele fixsc end sele flexsc end ncopy @numcopy

!---------------------------------------------------------------------
! Generate random ligand conformer
!---------------------------------------------------------------------
```

```
!! Generate random rotamer
label ligandrotamer
read coor select segid PROT end card name "flexchain.crd"
system " obrotamer ./ligand.pdb | convpdb.pl -segnames | sed s/PROO/LIGA/g > \
             ligand_rotamer.pdb "
read coor pdb name "ligand_rotamer.pdb" resid

!! Brief minimization of ligand and flexible side chain
mini sd nstep 50 tolenr 0.01
mini abnr nstep 50 tolenr 0.005

!! Ligand rotamer energy
inte select segid LIGA end ! Ligand energy
set tmpener = ?ener
calc refener = @ligaener + 1000
if @tmpener .ge. @refener then
        goto ligandrotamer
endif

!! Output
write coor pdb select segid LIGA end name "./conformer/@IDXNUM.pdb"
write coor card select segid LIGA end name "ligand_rotamer.crd"
write coor card select segid PROT end name "tmpprotein.crd"

!-----------------------------------------------------------------------
! Generate random ligand position copy
!-----------------------------------------------------------------------
```

```
!! Energy cut off in docking
inte select segid PROT end ! Protein energy
set protener = ?ener
inte select segid LIGA end ! Ligand energy
set ligaener = ?ener
calc totalener = @protener + @ligaener + 500 ! Total energy

!! Randomlize its postion and orietation
set i = 1  ! Count number
label initialPose
      read coor select segid LIGA end card name "ligand_rotamer.crd"
      read coor select segid PROT end card name "tmpprotein.crd"
      ! rotation
      calc theta ( ?RANDOM - 0.5 ) * ( ?PI )
      calc phi ( ?RANDOM ) * ( ?PI ) * 2
      calc XDIR cos(@theta) * cos(@phi)
      calc YDIR cos(@theta) * sin(@phi)
      calc ZDIR sin(@theta)
      calc rphi ( ?RANDOM ) * 360
      coor rotate xdir @XDIR ydir @YDIR zdir @ZDIR xcen @xcen ycen @ycen zcen -
                        @zcen phi @rphi select segid LIGA end

      ! translation
      coor stats select segid LIGA end
      calc xtran ( ?RANDOM - 0.5 ) * 2 * 2
      calc ytran ( ?RANDOM - 0.5 ) * 2 * 2
```

```
calc ztran ( ?RANDOM - 0.5 ) * 2 * 2
coor trans xdir @xtran ydir @ytran zdir @ztran select segid LIGA end

! read hard grid potential
open unit 30 read unform name "grid-emax-3-mine--20-maxe-40.bin"
grid read unit 30 select flexsc end
close unit 30

! minimize with hard potential
mini sd nstep 100
mini abnr nstep 500 tolenr 0.01
grid off
grid clear

! read native potential
open unit 30 read unform name "grid-emax-10000-mine--10000-maxe-10000.bin"
grid read unit 30 select flexsc end
close unit 30

! minimize with native potential
mini sd nstep 50
mini conj nstep 100 tolenr 0.01
grid off
grid clear

! energy cutoff
enery
```

```
      if ?ener .ge. @totalener then
         goto initialPose
      endif

      ! loops
      OMMD setc idxc @idxcopy
      incr idxcopy by 1
      incr i by 1
      if @i .gt. @copy then
         incr idxnum by 1
         if @idxnum .le. @num then
            goto ligandrotamer
         endif
      endif
      if @idxcopy .le. @numcopy then
         goto initialPose
      endif
grid clear

!-----------------------------------------------------------------------
! OpenMM docking -- Parallel SA
!-----------------------------------------------------------------------

OMMD CGRS SOFT 0 HARD 1 emax 15 mine -120.0 maxe -2.0 eps 3
OMMD SIAN NSTE 3000 FIRST 300 FINAT 700 NHRQ 50 INCT 1

OMMD CGRS SOFT 0 HARD 1 emax 15 mine -120.0 maxe -2.0 eps 3
```

```
OMMD SIAN NSTE 14000 FIRST 700 FINAT 300 NHRQ 50 INCT -1

OMMD CGRS SOFT 1 HARD 0 emax 3 mine -20 maxe 40.0 eps 3
OMMD SIAN NSTE 7000 FIRST 500 FINAT 300 NHRQ 50 INCT -1

OMMD CGRS SOFT 1 HARD 0 emax 30 mine -200 maxe 400.0 eps 3
OMMD SIAN NSTE 3000 FIRST 400 FINAT 50 NHRQ 50 INCT -1

!---------------------------------------------------------------------
! Final result
!---------------------------------------------------------------------

!! Output
set idxcopy = 1
set idxnum = 1
set i = 1
label LoopOutput
      ! copy coordinate from OpenMM_dock results into the main coordinates
      OMMD COOR IDXC @idxcopy

      write coor pdb select segid LIGA end name "./ligand/@IDXNUM_@I.pdb"
      write coor pdb select segid PROT end name "./protein/@IDXNUM_@I.pdb"

      incr idxcopy by 1
      incr i by 1
      if @i .gt. @copy then
         incr idxnum by 1
```

```
      set i = 1
   endif

   if @idxcopy .le. @numcopy then
      goto LoopOutput
   endif

stop
```