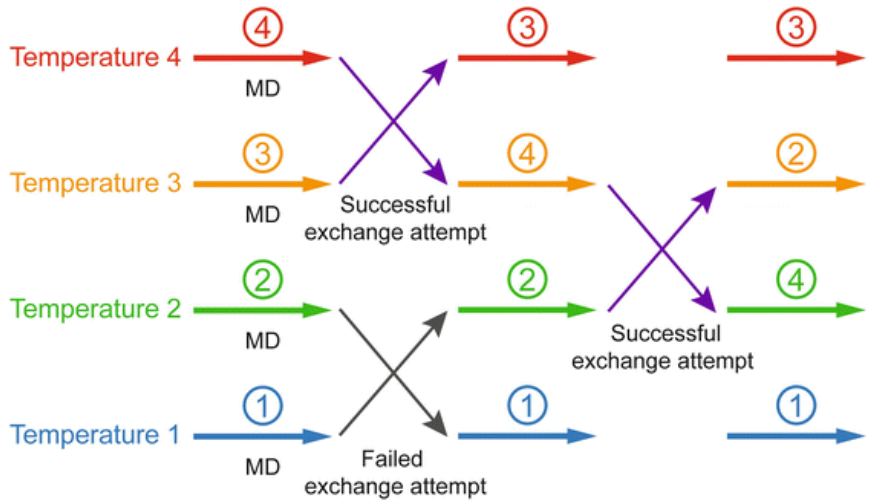
A Tutorial on Replica exchange molecular dynamics (REMD) simulations

Written by Sasthi Mandal
Edited by Dr. Atanu Acharya
May 2025

Replica exchange molecular dynamics (REMD) simulations



Source: Qi, R., Wei, G., Ma, B., Nussinov, R. (2018). Replica Exchange Molecular Dynamics: A Practical Application Protocol with Solutions to Common Problems and a Peptide Aggregation and Self-Assembly Example. In: Nilsson, B., Doran, T. (eds) Peptide Self-Assembly. Methods in Molecular Biology, vol 1777. (https://doi.org/10.1007/978-1-4939-7811-3_5).

REMD method was first introduced by Okamoto et. al. REMD method allows users to simulate several copies of the system in parallel for MD simulations at different temperatures or at the same temperature but for different Hamiltonians. Furthermore, the replicas can be swapped between different temperatures using the Metropolis algorithm. In this way, REMD can overcome free energy barrier of the system and explore the free energy surface by sampling conformational space sufficiently.

1. Running an equilibrium MD simulations

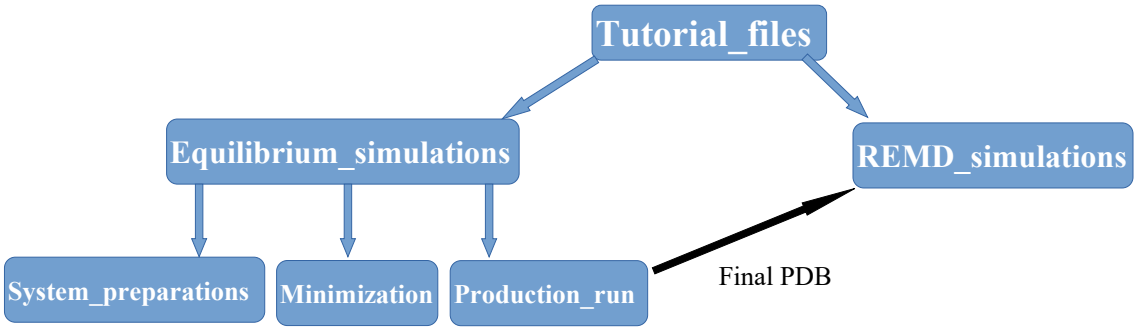
1.1 Prerequisite

(a) NAMD 2.10 or later
(b) VMD 1.9 or later

This tutorial assumes prior experience with Linux systems and basic command-line operations. We begin with standard molecular dynamics (MD) simulations before progressing to REMD.

All the required files are located in a directory named `Tutorial_files`. To access it, open the terminal and enter: *cd Tutorial_files*.
Within the *Tutorial_files* directory, you'll find two subdirectories: *Equilibrium_simulations* and *REMD_simulations*. We'll begin with the conventional MD simulations. Navigate to the appropriate folder by typing: *cd Equilibrium_simulations* in the terminal.

Here's a flowchart illustrating the directory structure:



1.2 Running a conventional MD simulation

In this tutorial, we will focus on the cell-penetrating peptide RW16, with the sequence (RRWRRWWRRWWRRWRR).

To perform molecular dynamics (MD) simulations using NAMD, the following steps will be carried out:
(I) Building the system
(II) Energy minimization
(III) Equilibration
(IV) Production run

(I) Building the system

Navigate to the system_building subdirectory within the Equilibrium_simulations directory by typing: cd System_building in the terminal. You will find all the necessary files to build the system inside the System_building subdirectory.

a) Structure file
The structure file (PDB) for the cell-penetrating peptide RW16 is available from the RCSB Protein Data Bank under the PDB ID **RW16**. Alternatively, a copy of this file is included in the tutorial materials.

b) psf file
NAMD requires a PSF (Protein Structure File) to perform MD simulations. A PSF file contains information about atom masses, charges, atom types, and the connectivity between atoms, including bonds, angles, dihedrals, and impropers.

We will create a PSF file for the system. To do so, both a structure file (PDB) and the appropriate force-field parameters are required.

To create a psf file for the system, open an editor using vi, then copy and paste the following lines. Save it as *rw16.pgn*.

```
package require psfgen

set parpath /home/scmandal/toppar         ; Path to the topology files
topology $parpath/top_all36_prot.rtf        ; Topology file for protein
topology $parpath/toppar_water_ions.str  ; Topology file for water and
ions

pdbalias residue HIS HSE
pdbalias atom ILE CD1 CD
segment U {pdb 6rqs.pdb}
coordpdb 6rqs.pdb U
guesscoord
writepdb 6rqs_processed.pdb
writepsf 6rqs_processed.psf
```

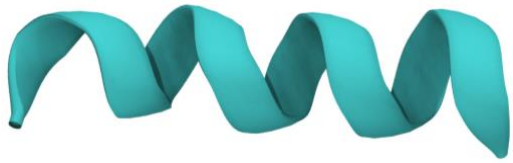


Fig. 1. Structure of RW16. (*6rqs_processed.pdb)*

Note: The following lines:
```
pdbalias residue HIS HSE
pdbalias atom ILE CD1 CD
```

are used to match residue and atom names between the PDB file and the force field definitions. The residue name `HIS` is ambiguous because it does not specify the protonation state of histidine. Depending on the protonation state, histidine can be represented as:

- `HSE` = proton on **NE2**
- `HSD` = proton on **ND1**
- `HSP` = **both** NE2 and ND1 are protonated (positively charged histidine)

Pdb files often write histidine residues as HIS without specifying its protonation state.

`pdbalias residue HIS HSE` – This renames all the HIS residues as HSE.

In the CHARMM topology, the isoleucine atom CD1 in PDBs is actually called CD.

`pdbalias atom ILE CD1 CD` – This renames the CD1 atom as CD for all the isoleucine residues

However, since our system contains no histidine or isoleucine residues, these two aliasing lines are not necessary.

Solvate the system

```
package require solvate
solvate 6rqs_processed.psf 6rqs_processed.pdb -t 30 -o 6rqs_solv
```

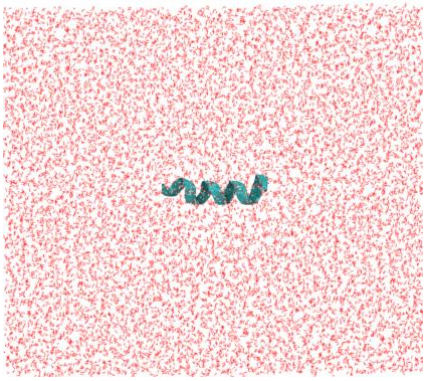


Fig. 2. RW16 in water

This will add a 30 Å thick solvent layer around all sides of the solute. Adjust the `-t 30` value as needed to control the thickness of the solvent padding.
This command will generate a PSF and PDB file named `6rqs_solv.psf` and `6rqs_solv.pdb`, respectively.

Ionize the system

```
package require autoionize
autoionize -psf 6rqs_solv.psf -pdb 6rqs_solv.pdb -neutralize -o
6rqs_solv_neutral
quit
```

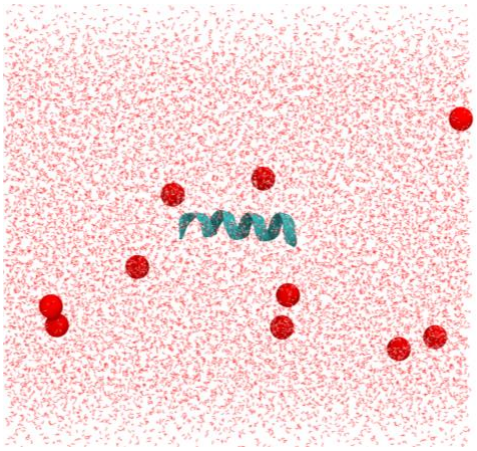


Fig. 3. RW16 in water and ions (*6rqs_solv_neutral.pdb).*

This will neutralize the system by adding the required ions, replacing solvent molecules as needed.

Copy all the commands into a file named `rw16_sys_build.pgn`, and run it in the terminal using the following command: *vmd -dispdev text -e rw16.pgn*
This will generate a PSF and PDB file named `6rqs_solv_neutral.psf` and `6rqs_solv_neutral.pdb`, respectively. These files will contain the peptide, solvent, and the added ions required to neutralize the system.

(II) Energy minimization
Perform energy minimization to remove any unfavorable atomic contacts within the system. Use the *minim.conf* configuration file provided in the tutorial to carry out this step. After minimization, plot the potential energy against the minimization steps. The graph should show convergence to a stable, negative potential energy value.

Navigate to this directory by typing `cd ../Minimization`. You will find all the necessary input and output files.

(III) Equilibration
Once minimization is complete and bad contacts are resolved, the next step is to equilibrate the system to reach the desired temperature. In this tutorial, we proceed with an NPT ensemble for equilibration (alternatively, NVT can be run prior to NPT if needed). Use the *equilibrate.conf* file included in the tutorial materials to perform this step.

Navigate to this directory by typing `cd ../Equilibration`. You will find all the necessary input and output files.

(IV) Production run
With the system equilibrated, initiate the production molecular dynamics (MD) simulation using the *production.conf* file. In this tutorial, a 10 ns production run is conducted. Output files, including the final structure and trajectory files, are generated. The final structure file will be used as the starting point for the Replica Exchange Molecular Dynamics (REMD) simulation in the next section.

Navigate to the `Production_run` directory by typing:

cd ../Production_run

Here, you will find all the input and output files related to the production MD simulations.

2. REMD Simulation Setup

REMD simulations will be initiated using the final structure obtained from the 10 ns conventional MD run. In this final structure, the solute may have drifted from the center of the simulation box, and in some cases, parts of the solute might appear to exit one side of the box and re-enter from the opposite side due to periodic boundary conditions. Therefore, it is good practice to wrap the structure before proceeding with further MD simulations.

To perform REMD, four configuration files are required: base.namd, remd.conf, replica_remd.conf, and job0.conf.

All the above files are provided in the following directory: Tutorial_files/REMD_simulations
Enter to the directory by typing cd Tutorial_files/REMD_simulations

2.1 base.namd file

The parameters specified in base.namd are largely identical to those used in the equilibrium MD simulation.

For a detailed explanation of these parameters and their functions, please refer to the NAMD tutorial documentation.

```
set psffile system.psf
set pdbfile system.pdb
set parPATH /home/scmandal/toppar

set cell_x 155.7
set cell_y 158.7
set cell_z 241.2
set cen {0.10599816590547562 0.12088041752576828 0.2550520896911621}

structure                 $psffile
coordinates             $pdbfile

paraTypeCharmm          on
parameters              $parPATH/par_all36m_prot.prm
parameters              $parPATH/NBFIX_JC_03132019.str
parameters              $parPATH/toppar_water_ions_modified.prm
parameters
$parPATH/SCMmodified/par_hemeOx_charmm36_modified.prm
parameters              $parPATH/par_all36_carb.prm
parameters              $parPATH/par_all36_lipid.prm
parameters              $parPATH/par_all36_cgenff.prm
parameters              $parPATH/toppar_all36_lipid_bacterial.str
parameters              $parPATH/toppar_all36_lipid_lps.str
parameters              $parPATH/toppar_all36_carb_imlab.str

exclude                  scaled1-4
1-4scaling              1.0

switching             on
switchdist            10
cutoff                     12
pairlistdist          13.5
rigidbonds            all
timestep                 2.0
stepspercycle           10

nonbondedFreq           1;          # nonbonded forces every step
```

```
fullElectFrequency      2
pme                         yes
pmegridspacing           1

langevin                  on
langevinDamping     1.0
langevinHydrogen     off

LangevinPiston            on
LangevinPistonPeriod 400
LangevinPistonDecay  200
useGroupPressure      yes
useFlexibleCell       no ;    # no for water box, yes for membrane
useConstantRatio     yes ;   # keeps the ratio of the unit cell in the x-y plane constant A=B

cellBasisVector1    ${cell_x}    0.0     0.0
cellBasisVector2    0.0       ${cell_y}  0.0
cellBasisVector3    0.0       0.0     ${cell_z}
cellOrigin          $cen

wrapWater           on
wrapAll             on

# NAMD 3 specific keywords for high performance computing

CUDASOAintegrate    on
Margin               4
```

2.2 remd.conf file

The remd.conf file contains the following necessary parameters to run a REMD simulation:

```
set num_replicas 6
set min_temp 300
set max_temp 350
set steps_per_run 1000
set num_runs 1000
set runs_per_frame 1
set frames_per_restart 50
set namd_config_file "base.namd"
set output_root "output/%s/RW16"
```

The keywords listed above are explained in detail below:

- `set num_replicas 6`

This defines the total number of replicas to simulate at different temperatures. Here, we choose 6 replicas in the temperature range 300-350 K. The more replicas you choose, the more smoother your sampling would be, but requires more computational resources. You can modify the number of replicas and choose a different number according to your system and what you want to achieve.

- *set min_temp 300*

This sets the minimum temperature to one of the replicas. The typical value of setting the minimum temperature is in the range (~270 – 310 K) for proteins, and often 310 K for membrane.

- *set max_temp 350*

Sets the maximum temperature to one of the replicas. Typically, for small peptides this ranges, 500-600K, for proteins, 450-550 K. Sometimes, too high temperature makes the protein denature. Hence, look at it before setting the maximum temperature.

- *set steps_per_run 1000*

This sets the number of steps to run before the exchange happens. In this case, all replicas will run for 5000 steps and then attempt to exchange temperatures with neighboring replicas. Example: Here, 1000 steps is equivalent to 2 ps of simulation time if you are using 2 fs time step. Based on the Metropolis criteria, after every 2 ps, each replica will attempt to exchange its configuration with a neighboring replica at a different temperature.

- *set num_runs 2000*

This sets how many replica exchange attempts will be made. The total number of steps to be run per replica is *steps_per_run* x *num_runs*.
(**Note:** num_runs should be divisible by runs_per_frame x frames_per_restart.)

Example:

If steps_per_run is set to 1000 and num_runs is 2000, each replica will run for a total of:

1000 steps/run×2000 runs=2,000,000

If you use 2 fs timestep, this corresponds to:
2,000,000 steps×2 fs=4,000,000 fs=4 ns

- *set runs_per_frame 1*

This specifies how many runs (each consisting of steps_per_run MD steps) are executed before writing one frame to the trajectory output.
If runs_per_frame is set to 1, then **one trajectory frame will be saved after every run**, i.e., after every steps_per_run steps.

Example:
If steps_per_run = 1000 and runs_per_frame = 1, a trajectory frame will be written every 1000 steps (or every 2 ps if your timestep is 2 fs). Here, exchange attempts and frame writing both happen every 1000 steps.
If runs_per_frame = 10, then a frame would be written every 10,000 steps (20 ps at 2 fs timestep).

If the num_runs = 2000 and runs_per_frame = 1, then:

You will get **2000 frames per replica**, since a frame is written after every run.

- *set frames_per_restart 50*

Number of trajectory frames written before restart files are generated.

- *set namd_config_file "base.namd"*

This specifies a NAMD configuration file which contains the basic MD parameters to run a simulation.

- *set output_root "output/%s/RW16"*

Path to the output files. %s automatically replaced with replica index (e.g., 0, 1, ..., 5).

2.3 job0.conf file

```
source remd.conf

margin 2

if { ! [catch numPes] } { source replica_remd.namd }
```

NAMD first reads the remd.conf file that contains the basic REMD settings. Next, NAMD reads the replica_remd.conf containing NAMD code to run the REMD simulations.

2.4 replica_remd.conf

This file contains REMD code to run the REMD simulation.

2.5 Running a REMD simulation

Navigate to your working directory:

cd <working_directory> (Here, cd Tutorial_files/REMD_simulations)

Inside your working directory, create an output directory:

mkdir output

Go to the output directory:

cd output

Next, create a directory for each replica. For example, if you're working with 5 replicas, create 5 directories with the following command:

mkdir {0, 1, 2, 3, 4, 5}

Alternatively, you can use the bash script (`generate_replicas.sh`) provided in the Tutorial_files/REMD_simulations directory to generate the output and the replicas directories.

To run this script type in your terminal: `./generate_replicas.sh <output_directory_name> <number_of_replicas>`

(or `bash generate_replicas.sh <output_directory_name> <number_of_replicas>`)

Here,

`<output_directory_name>` = output

`<number_of_replicas>` = 6

Return to your working directory (cd Tutorial_files/REMD_simulations):

Running the REMD job using *charmrun*

```
$namd3bin/charmrun ++local ++p 30 ++ppn 5 $namd3bin/namd3 job0.conf +replicas 6 +stdout output/%d/job0.%d.log >> output/job0.log 2> output/job0.out
```

++p – Total number of processors or processor elements (PE)
++ppn – Number of PEs per node
The total number of PEs must have to be a multiple of the total number of replicas (i.e. the value after *++p* must have to be the number of PEs per node x total number of replicas).

The command *+stdout output/%d/job0.%d.log* directs the session log for the **i**th replica to the file *output/i/job0.i.log*.

Alternatively, one can use mpirun to run the REMD simulations. The command is as follows:

```
mpirun -np 30 namd +replicas 6 job0.conf +stdout output/%d/job0.%d.log
```

The total number of ranks (-np 30) must always be divisible by the total number of replicas.

2.7 Restart a REMD simulation

To restart a simulation, you need the following parameters in your conf file (job1.conf)

```
source remd.conf

source [format output/RW16.job0.restartXXX.tcl ""]

if { ! [catch numPes] } { source replica_remd.namd }
```

The `RW16.job0.restartXXX.tcl` file is generated when the current run either finishes or is interrupted after reaching the point where restart files are written. In this case, the file is named `RW16.job0.restart50.tcl`. Therefore, when restarting the simulation, you need to replace `RW16.job0.restartXXX.tcl` with `RW16.job0.restart50.tcl` in the above file (job1.conf).

The *RW16.job0.restart50.tcl* file contains the following keywords:

```
set i_job 1
set i_run 50
set i_step 50000
```

set restart_root output/%s/RW16.job0.restart50

- *set i_job 1*

This sets the job index. If this is the first restart file, i_job will be 1; for the second restart file, it will be 2, and so on.

- *set i_run 50*

This indicates the total number of completed runs, calculated by dividing the total number of simulation steps by the number of steps per run (as specified in the remd.conf file). The value is always a multiple of 5 and should be rounded up to the nearest multiple of 5.

Example: If the simulation stopped at 43 ns, this value would be set to 50.

- *set i_step 50000*

This specifies the number of simulation steps completed in the previous run. It corresponds to the actual time evolution steps the system has undergone so far.

- *set restart_root output/%s/RW16.job0.restart50*

This defines the path to the restart files to be used for resuming the simulation. The %s placeholder is typically replaced by the replica ID or temperature index during runtime.

The restart files include:

A. RW16.job0.restart50.coor – coordinates

B. RW16.job0.restart50.vel – velocities

C. RW16.job0.restart50.xsc – extended system info

These files are essential for correctly resuming the simulation from the point it was interrupted.

Note: The first three lines (`i_job`, `i_run`, and `i_step`) are *not* used directly in the restart simulation but provide a reference to the state of the previous run, aiding in tracking and managing multiple restarts.

At each restart, create a new job*.conf file and include the appropriate restart.tcl file from the previous run. Make sure to replace the old restart.tcl file with the newly generated one before starting the next simulation. This ensures that the simulation resumes from the correct point.

To run this, type: *$namd3bin/charmrun ++local ++p 30 ++ppn 5 $namd3bin/namd3 job1.conf +replicas 6 +stdout output/%d/job0.%d.log >> output/job0.log 2> output/job0.out*

2.6 Output

The output files for each replica are organized within the output directory. Inside this directory, there are six subdirectories named 0 through 5, each corresponding to a specific replica. Within each replica's directory, you'll find the respective output files, including trajectory files, generated during the simulation.

To explore the output files generated from the Replica Exchange Molecular Dynamics (REMD) simulations, follow these steps in your terminal:

(a) Navigate to the Output Directory

```
cd REMD_simulations/output
```

(b) To view the available replica directories, use the ls command:
You should see directories named 0, 1, 2, 3, 4, and 5, each corresponding to a different replica.

(c) Access a Specific Replica's Output

Enter one of the replica directories, for example, replica 0:

```
cd 0
```

(d) Then, list the output files within that replica's directory:

```
ls
```

This will display the output files generated from the REMD simulation for that specific replica, including trajectory files and other relevant data.

3. Analysis

Sorting replicas

During the REMD simulations, after each exchange attempt, the replicas might get switched within different replicas at different temperatures. A system that was running at 300 K temperature, might switched to 330 K, and vice versa. This is normal for REMD, but it would be difficult to analysis the trajectory if you want to analyse properties of a system at a particular temperature. Therefore, instead of following a replica that switches between temperatures, sorting replicas involves rearranging the simulation outputs so that each trajectory corresponds to a consistent temperature throughout the simulation. In this way, without keeping track of which replica was at a certain temperature at any given moment, you can examine the time evolution of your system at that temperature (for example, 300 K).

To perform replica sorting in NAMD, use the sortreplicas executable provided in the NAMD installation directory.
For post-simulation analysis, this tool reads the exchange history and reorders the trajectories so that each output corresponds to a given temperature throughout the simulation.
To execute sortreplicas, use the following command from within the REMD_simulations directory:

```
sortreplica <job_output_root> <num_replicas> <runs_per_frame> [final_step]
```

In this tutorial, the parameters are defined as follows:
`<job_output_root>` = output
`<num_replicas>` = 6
`<runs_per_frame>` = 1
`[final_step]` = 2000000 (Last simulation step (*2000X1000 = 2000000*))