

GROMACS example for Blacklight

This example is part of the gmxdemo file given at www.gromacs.org, and simulates Molecular Dynamics (MD) using the GROMACS software package. The objective of this example is to demonstrate the use of different GROMACS modules. The complete steps for a simulation can be found at www.gromacs.org.

The demo uses several GROMACS modules to simulate a small peptide in water. The only input files needed are a pdb file of a small peptide and an mdp file containing run parameters for the energy minimization step.

If you have any questions about this demonstration, please email PSC User Services, or check the resources on the website: <http://www.gromacs.org>.

The mdrun executables are compiled specifically for the compute nodes. **mdrun must be launched with a mpirun command.**

. All other GROMACS executables are serial and can be run on your own workstation or on blacklight's front-end nodes.

This document is organized as follows. First a summary of the commands needed to perform the MD are given. Links from each command lead to an explanation of that step. The first three steps set up the run time environment, and the remainder perform the simulation.

Summary of commands (4 processor run)

1. [qsub -l -q debug -l ncpus=16 -l walltime=30:00](#)
2. [module load gromacs](#)
3. [cd *working-directory*](#)
4. [pdb2gmx -f cpeptide.pdb -o cpeptide.gro -p cpeptide.top](#)
5. [editconf -f cpeptide.gro -o cpeptide.gro -d 0.5 & output. genbox](#)

6. [genbox -cp cpeptide.gro -cs -o cpeptide b4em.gro -p cpeptide .top &&& output.genbox](#)
7. [grompp -f em -c cpeptide b4em -p cpeptide -o cpeptide_em && output.grompp_em](#)
8. [mpirun -np \\$PBS_NCPUS mdrun -s cpeptide_em -o cpeptide_em -c cpeptide b4pr -v && ! output.mdrun_em](#)

Please note, as stated above, the mdrun executable runs on only blacklight's compute nodes. It can only be run in batch mode, with the mpirun command. You must submit a [batch job to blacklight's queues](#)

Explanation of Demo

Setting up the run environment

1. Obtain an interactive session

To begin the demo, obtain an [interactive debugging session](#) . Ask for 16 processors and a thirty minute wall time.

```
qsub -l -q debug -l ncpus=16 -l walltime=30:00
```

2. Set up the gromacs run time environment

The module package allows a user's environment to be dynamically altered. For example, directories can be added to the default path. This will just make subsequent commands a little shorter to type.

Load the gromacs module to define the directories where the gromacs executables are stored.

```
module load gromacs
```

3. Move to the submission directory

Move to the directory where the qsub command was executed from. Otherwise, you are in your home directory by default.

cd working-directory **Running the simulation**

4. Generate a topology file

Before you can start any simulation, you need a molecular topology file. This topology file (.top extension) is generated by the program **pdb2gmx**. The peptide pdb file (.pdb extension) is the input for **pdb2gmx**.

Because most pdb files do not contain all hydrogen atoms, the **pdb2gmx** program will also add them to the peptide. The output file containing the structure of the peptide when hydrogen atoms are added is a gromos structure file (.gro extension).

```
pdb2gmx -f cpeptide.pdb -o cpeptide.gro -p cpeptide.top
```

You will be prompted to choose a force field and a water model for the simulation. For this example, choosing the first options (force field: AMBER03 protein, nucleic AMBER94, water model: TIP3P) is fine.

Input files: [cpeptide.pdb](#)

Output files: [cpeptide.gro](#) , [cpeptide.top](#) , [posre.itp](#)

5. Define the water box size

Because a simulation of a peptide in vacua is a bit unrealistic, you have to solvate the peptide in a box of water. Use **genbox** to do this.

First, use the program **editconf** to define the right box size for the system.

```
editconf -f cpeptide.gro -o cpeptide.gro -d 2.0 >& ! output.genbox
```

Input files: [cpeptide.gro](#) from step 4

Output files: [cpeptide.gro](#) , [output.genbox](#)

6. Solvate the peptide

The **genbox** program reads the peptide structure file and the input file containing the size of the desired water box. The output of **genbox** is a gromos structure file of a peptide solvated in a box of water. **genbox** also changes the topology file (.top extension) to include water.

```
genbox -cp cpeptide.gro -cs -o cpeptide_b4em.gro -p cpeptide.top >>& output.genbox
```

Input files: [cpeptide.gro](#) from step 5

Output files: [cpeptide_b4em.gro](#) , [cpeptide.top](#) , [output.genbox](#)

7. Energy minimization, step 1: preprocess the input files

In principle, you can start a Molecular Dynamics simulation now. However, it is not very wise to do so, because the system is full of close contacts. These close contacts are mainly a result of the **genbox** program. The added solvent might have some close contacts with the peptide resulting in very high repulsive energies. If you start a Molecular Dynamics (MD) simulation without energy minimization, the system would be unstable because of these high energies.

The standard procedure to remove these close contacts is Energy Minimization (EM). Energy minimization slightly changes the coordinates of the system to remove high energies from it.

Before you can start the Energy Minimization, you have to preprocess all the input files with the GROMACS preprocessor **grompp**. **grompp** preprocesses the topology file (.top), the structure file (.gro) and a parameter file (.mdp) resulting in a binary topology file (.tpr). This binary topology file contains all information for a simulation (in this case, an energy minimization).

```
grompp -f em -c cpeptide_b4em -p cpeptide -o cpeptide_em >& output.grompp_em
```

Input files: [cpeptide.top](#) from step 6, [cpeptide_b4em.gro](#) from step 6, and [em.mdp](#).

Output files: [cpeptide_em.tpr](#), [mdout.mdp](#), and [output.grompp_em](#)

8. Energy minimization, step 2

Now that the binary topology file is generated, you can start the energy minimization (EM). The program which performs the EM is called **mdrun**. In fact, all simulations are performed by the same **mdrun** program.

Create a batch job script that will run the EM, and submit it with the [qsub command](#). After it finishes, check the output of the program. The first number (from left to right) is the number of the iteration step. The second number is the step size, which gives an indication of the change in the system. The third number is the potential energy of the system. This number starts at a high value, rapidly drops and converges to a large negative value.

NOTE: the variable -np must specify the same number of processors with which you intend to run **mdrun**. Here it is set to the variable \$PBS_NCPUS so this command does not change even when a different number of processors is used.

Include this command in your script to run the EM:

```
mpirun -np $PBS_NCPUS mdrun -s cpeptide_em -o cpeptide_em -c cpeptide_b4pr -v >&  
output.mdrun_em
```

Input files: [cpeptide_em.tpr](#) from step 7

Output files: [cpeptide_em.trr](#) , [cpeptide_b4pr.gro](#) , [output.mdrun_em](#) , [ener.edr](#) , [md.log](#) , [md](#)
[1.log](#)

,

[md2.log](#)

,

[md3.log](#)

.