# Tips and Tricks for Minimizing RNA structures with GROMACS

Magdalena A. Jonikas

August 7th, 2009

**Abstract**

This document is meant to be a simple guide to running gromacs minimizations of full atomic RNA structures using the amber force field. It is written from the point of view of a non-MD-expert who learned to use the minimization tools that are pract of gromacs. This document contains the steps we followed as the last part of our Coarse to Atomic (C2A) project (to remove gaps and collisions from reconstructed full atomic RNA molecules). The information contained in this document may also be useful for users interested in minimizing RNA structures for other purposes. There are many website, documents and papers that contain information about running gromacs and using the amber force field, we have merely combined the information from these many sources for the particular application of running energy minimization of RNA molecules. This document includes tips and tricks we have found useful as we experimented with running gromacs and we hope they will be helpful for you as well. The code described in this document can also be found on the C2A project website at www.simtk.org/home/c2a.

# Contents

# 1 Tips for installing GROMACS

We will assume you are using a unix platform.

## 1.1 Downloading and installing GROMACS

Download the latest gromacs binaries from the gromacs website to your machine. Decompress the tar-ball, and enter the gromacs directory:

```
tar xzf gromacs-0.0.0.tar.gz
```

To configure and install with single precision (you will need sudo privileges to instal globally):

```
./configure
make
sudo make install
```

If the configuration step crashes because of an X11 error, try:

```
./configure --without-x
```

To make the first make command run faster, try:

```
make -j8
```

To configure and install a double precision version of gromacs, try:

```
./configure --without-x --enable-doube --program-suffix=
    _d
make -j8
sudo make install
```

Following the "make install" command with the recommended:

```
make links
```

To link the executables to /usr/local/bin.
If you find that you need to re-run any make commands, first run:

```
make clean
```

We recommend you make both single and double precision installations of gromacs. If you followed our instructions, double precision commands with have the suffix "_d", so you will be able to specify which one you want to use. Gromacs will install in the /usr/local/gromacs directory that it creates.

### 1.2 Download and install the XDR file library

Download the latest version of the xdr file library from the gromacs website. Decompress, configure and install:

```
tar xzf xdrfile-x.x.tar.gz
cd xdrfile-x.x
./configure
sudo make install
```

## 2 Extra files you need to use the amber force field

Download the latest amber port files from http://chemistry.csulb.edu/ffamber/ and make sure you copy the following files to your /usr/local/gromacs/share/-gromacs/top/ directory:

- ffamber03.atp
- ffamber03bon.itp
- ffamber03-c.tdb
- ffamber03.hdb
- ffamber03.itp
- ffamber03nb.itp
- ffamber03-n.tdb
- ffamber03.rtp

You will also need to create an spc-amber.itp file in that same top directory containing the following text:

```
[ moleculetype ]
; molname    nrexcl
SOL        2

[ atoms ]
;   nr    type   resnr  residue   atom    cgnr      charge
          mass
     1   amber99_54    1   SOL      OW       1       -0.82
         15.99940
     2   amber99_55    1   SOL     HW1       1        0.41
         1.00800
     3   amber99_55    1   SOL     HW2       1        0.41
         1.00800
```

```
#ifdef FLEXIBLE
[ bonds ]
; i    j  funct  length    force.c.
1   2   1   0.1    345000    0.1      345000
1   3   1   0.1    345000    0.1      345000

[ angles ]
; i    j   k  funct  angle  force.c.
2   1   3   1   109.47    383    109.47    383
#else
[ settles ]
; OW  funct  doh    dhh
1   1   0.1    0.16333

[ exclusions ]
1   2   3
2   1   3
3   1   2
#endif
```

One last thing, you will need to add the following to the beginning of your ions.itp file:

```
[ moleculetype ]
; Name              nrexcl
Na                  1

[ atoms ]
;    nr    type   resnr  residue  atom    cgnr      charge
          mass
1 amber99_31  1  Na Na  1  1  22.99


[ moleculetype ]
; Name              nrexcl
Cl                 1

[ atoms ]
1 amber99_30  1  Cl  Cl  1  -1  35.45
```

# 3 PDB format tricks

You will need to modify your PDB file before you can use it with gromacs. In particular, all residue names need to be preceded with the letter R (e.g. U

needs to be RU). You cannot have the "*" symbol that is used in some old PDB formats for base carbon atoms, it needs to be replaced with the "" symbol. Finally, the O2 oxygen needs to be replaced with the simply O letter. The following commands will fix your $NAME.pdb file, you may want to create you own script that runs these commands automatically on your PDB file:

```
sed -i 's/ C /RC /g' $NAME.pdb
sed -i 's/ G /RG /g' $NAME.pdb
sed -i 's/ A /RA /g' $NAME.pdb
sed -i 's/ U /RU /g' $NAME.pdb
sed -i "s/*/'/g" $NAME.pdb
sed -i 's/O2   RC/O   RC/g' $NAME.pdb
sed -i 's/O2   RU/O   RU/g' $NAME.pdb
```

# 4 Parameter file

You will need an em.mdp file in the folder in which you perform your minimization. This is a parameter file for the minimization process. In this example we use steepest descent minimization. Here is the file we used in our C2A paper:

```
;
;   User jonikas
;   12/2/08
;   Input file
;
cpp                 =  /usr/bin/cpp
define              =  -DFLEX_SPC
constraints         =  none
integrator          =  steep
nsteps              =  1000
;
;   Energy minimizing stuff
;
emtol               =  1000
emstep              =  0.01

nstcomm             =  1
ns_type             =  grid
rlist               =  1
rcoulomb            =  1.0
rvdw                =  1.0
Tcoupl              =  no
Pcoupl              =  no
gen_vel             =  no
```

# 5 Minimizing, finally

Here are the steps need to minimize:

1. Make a topology file from your PDB file using the pdb2gmx command
2. Make a box and add waters to the box
3. Perform an initial energy minimization
4. Add ions
5. Make and index file
6. Perform a final energy minimization

Let's walk through the process using our example (modified as described above) PDB file $NAME.pdb. At the very end of this document, we give the code for a bash script that performs all these steps automatically.

Before we start anything though, we need to set some paths using the command:

```
source /usr/local/gromacs/bin/GMXRC
```

## 5.1 Making a topology file

Run the following command:

```
pdb2gmx -f $NAME.pdb -p $NAME.top -o $NAME.gro -ff
    amber03
```

Check that the output file $NAME.gro was created as output.

## 5.2 Making a box and adding waters

Make the box:

```
editconf -f $NAME -o -d 2.0
genbox -cp out -cs -p $NAME -o $NAME-b4em
```

Check for the output file $NAME-b4em.gro

## 5.3 Performing the initial energy minimization

First you will need to edit the .top file reference to spc.itp to spc-amber.itp, try:

```
sed -i 's/spc.itp/spc-amber.itp/g' $NAME.top
```

The energy minimization process has two steps. You can read the details in the gromacs manual. Here's how you run them:

```
grompp -v -f em -c $NAME_b4em -o $NAME_em -p $NAME
mdrun -v -s $NAME_em -o $NAME_em -c $NAME_after_em -g
    $NAME_emlog
```

To use your double precision installation use the commands grompp_d and mdrun_d. After the grompp command, check for the output file $NAME-em.tpr. After the mdrun command, check for the output file $NAME-em.trr.

## 5.4 Adding ions

You need to add the same number of Na ions as you have residues in your system to balance out the charges in the system. For example, tRNA has 76 residues, so you would add 76 Na ions. Let's set the variable $NP to the number of ions we want to use.

```
genion -s $NAME_em -o $NAME_after_ion -p $NAME -np $NP
```

When prompted to select a group, type the value 5. This will let the program know that you want to replace 76 solvent molecules with Na and it will attempt to do so. Check for the output file $NAME-after_ion.gro.

## 5.5 Making an index file

You now need to make an index file, which requires your interaction:

```
make_ndx -f $NAME_after_ion
```

When prompted, type the following (those are vertical bars, not the letter el) to make a group consisting of all four residue types:

```
1|2|3|4
```

When prompter again, type the following to name the group:

```
name 7 RNA
q
```

The "q" command quits out of the program. Check to make sure the file index.ndx was created.

## 5.6 Performing the final energy minimization

You will now perform the final energy minimization, again with two steps:

```
grompp -v -f em -o $NAME_em2 -c $NAME_after_ion -p $NAME
    -n
mdrun -v -s $NAME_em2 -o $NAME_em2 -c $NAME_after_em2 -g
    $NAME_em2log
```

Once again, you can use the suffix _d to use your double precision installation (e.g. grompp_d). After the grompp command, check the for file $NAME-em2.tpr. After the mdrun command, check for the file $NAME-em2.trr.

### 5.7 Running all gromacs steps automatically

At the end of this document, you will find a code for a bash script that will run all the steps above automatically, so you don't need to interact with the gromacs code and can go for a bike ride instead.

## 6 Getting information from the output

There are probably two pieces of information you want after completing the minimization process: a PDB format file of the final minimized structure and information about the energy of the minimization process.

### 6.1 Getting the energies of the minimization

Use the g_energy command to extract energy information from the latest minimization step by simple typing:

```
g_energy
```

You will be prompted for the type of information you want to extract. You probably want the potential energy, but you might also be interested in bond, angle and dihedral energies. For all four of these, you would type:

```
1 2 3 9
```

Note that you need to follow the input with a blank line to exit the program. Now look at the energy.xvg file that was just created. You will find values for each of the variables you requested over the course the of simulation. Most interesting are the last numbers, because those are the values for the final conformation.

### 6.2 Getting a PDB file of the final conformation

To get a PDB format file of the last frame from the last minimization, use the following command:

```
trjconv -f $NAME-em2.trr -s $NAME-em2.tpr -n -o $NAME-em2
    .pdb
```

You will be prompted to select a group. We previously created group 7 for RNA, so type 7 and return. Check that the file $NAME-em2.pdb was created. Now you can view the minimized structure in with your favorite graphics program (VMD, pymol, etc.).

# 7 Useful documents and websites

Gromacs home page:
`http://www.gromacs.org/`

Gromacs manual:
`http://www.gromacs.org/@api/deki/files/82/=gromacs4_manual.pdf`

Gromacs introductory tutorial by John E. Kerrigan
`http://eugen.leitl.org/chem/kerrigje/pdf_files/fwspidr_tutor.pdf`

Parallel Molecular Dynamics: Gromacs By Eric Lindahl:
`http://www.scribd.com/doc/2958291/Parallel-Molecular-Dynamics-Gromacs`

ffamber for gromacs website:
`http://chemistry.csulb.edu/ffamber/`

Details on spc topology file from Yang Ye:
`http://lists.gromacs.org/pipermail/gmx-users/2007-November/030540.html`


# Acknowledgements

Thanks to Peter Kasson for tips on installing gromacs.

**APPENDIX**

# A Bash script to run all steps in an automated fashion

This script takes two command line inputs: The root name of your PDB file and the number of residues in the molecule (used to add the right number of ions). For example if your PDB file is called "mypdbfile.pdb" and it is an RNA molecule containing 76 residues, you would run this script like this:

```
sh minimize.sh mypdbfile 76
```

The script is designed to exit with error code 1 and and error message if one of the steps fails to produce the needed output file for the next step. It also saves all the verbose gromacs output to out-x.txt where x refers to the file root name and the specific step. You can check these file for hints if a particular step crashes. In addition to needing an em.mdp file in the same folder were you are running the minimization, you also need two expect scripts for which we give the code in other appendices (these interact with gromacs while you go on a bike ride).

Here is the code for minimize.sh:

```
#!/bin/bash

# Created by Magdalena A. Jonikas
# 5/11/09
# Files needed in same folder em.mdp $NAME.pdb makeIndex.
    exp getEnergy.exp

NAME=$1 # Name of pdb file (without the .pdb part)
NP=$2 # Number of ions to add
echo Running $NAME

echo Checking for needed files

if [ ! -e em.mdp ]
then
  echo Missing em.mdp
  exit 1
fi

if [ ! -e $NAME.pdb ]
then
  echo Missing $NAME.pdb
  exit 1
fi
```

```
if [ ! -e makeIndex.exp ]
then
  echo Missing makeIndex.exp
  exit 1
fi

if [ ! -e getEnergy.exp ]
then
  echo Missing getEnergy.exp
  exit 1
fi

echo Fixing the pdb file syntax
sed -i 's/ C /RC /g' $NAME.pdb
sed -i 's/ G /RG /g' $NAME.pdb
sed -i 's/ A /RA /g' $NAME.pdb
sed -i 's/ U /RU /g' $NAME.pdb
sed -i "s/*/'/g" $NAME.pdb
sed -i 's/O2    RC/O     RC/g' $NAME.pdb
sed -i 's/O2    RU/O     RU/g' $NAME.pdb

echo Make topology file from pdb file
pdb2gmx -f $NAME.pdb -p $NAME.top -o $NAME.gro -ff
    amber03 >& out-pdb2gmx-$NAME.txt

if [ ! -e $NAME.gro ]
then
  echo FAIL: pdb2gmx
  exit 1
fi

echo Make box and add waters
editconf -f $NAME -o -d 2.0 >& out-editcon-$NAME.txt
genbox -cp out -cs -p $NAME -o $NAME-b4em >& out-genbox-
    $NAME.txt

if [ ! -e $NAME-b4em.gro ]
then
  echo FAIL: editconf or genbox
  exit 1
fi

# edit .top file spc.itp to spc-amber.itp
sed -i 's/spc.itp/spc-amber.itp/g' $NAME.top
```

```
echo Perform an energy minimization
grompp_d -v -f em -c $NAME_b4em -o $NAME_em -p $NAME >&
    out-grompp-em-$NAME.txt
if [ ! -e $NAME_em.tpr ]; then
  echo FAIL: grompp em
  exit 1
fi
mdrun_d -v -s $NAME_em -o $NAME_em -c $NAME_after_em -g
    $NAME_emlog >& out-mdrun-em-$NAME.txt
if [ ! -e $NAME_em.trr ]; then
  echo FAIL: mdrun em
  exit 1
fi

# Add ions (76 default for tRNA, need to change NP at top
     of page for others)
echo Adding ions
echo 5 | genion -s $NAME_em -o $NAME_after_ion -p $NAME -
    np $NP >& out-genion-$NAME.txt
if [ ! -e $NAME_after_ion.gro ]; then
  echo FAIL: genion
  exit 1
fi

echo Make index file
expect makeIndex.exp $NAME_after_ion >& out-makeindex.txt
if [ ! -e index.ndx ]; then
  echo FAIL: makeIndex
  exit 1
fi

echo Perform final energy minimization
grompp_d -v -f em -o $NAME_em2 -c $NAME_after_ion -p
    $NAME -n >& out-grompp-em2-$NAME.txt
if [ ! -e $NAME_em2.tpr ]; then
  echo FAIL: grompp em2
  exit 1
fi
mdrun_d -v -s $NAME_em2 -o $NAME_em2 -c $NAME_after_em2 -
    g $NAME_em2log >& out-mdrun-em2-$NAME.txt
if [ ! -e $NAME_em2.trr ]; then
  echo FAIL: mdrun em2
  exit 1
fi
```

```
echo Convert trajectory to pdb
echo 7 | trjconv -f $NAME-em2.trr -s $NAME-em2.tpr -n -o
    $NAME-em2.pdb >& out-convert-$NAME.txt
if [ ! -e $NAME-em2.pdb ]; then
  echo FAIL: em convert
  exit 1
fi


echo Getting energies
expect getEnergy.exp >& out-energy-$NAME.txt
awk '{ field = $NF }; END{ print field}' energy.xvg >
    $NAME-E.txt

sh fixgromacsforpdb.sh $NAME-em2.pdb
cp $NAME-em2.pdb ..
```

# B Expect script for making index file

This expect script, makeIndex.exp, interacts with the make_ndx program so you don't have to.

```
#!/usr/bin/expect -f

set force_conservative 0  ;# set to 1 to force
    conservative mode even if
              ;# script wasn't run conservatively
                  originally
if {$force_conservative} {
    set send_slow {1 .1}
    proc send {ignore arg} {
        sleep .1
        exp_send -s -- $arg
    }
}

set timeout -1
spawn make_ndx -f [lindex $argv 0]
match_max 100000
expect -exact "> "
send -- "1|2|3|4\r"
expect -exact "\r
> "
send -- "name 7 RNA\r"
expect -exact "> "
send -- "q\r"
expect eof
```

# C Expect script for extracting energy information

This expect script, getEnergy.exp, interacts with the g_energy program so you don't have to.

```
#!/usr/bin/expect -f

set force_conservative 0  ;# set to 1 to force
   conservative mode even if
              ;# script wasn't run conservatively
                  originally
if {$force_conservative} {
    set send_slow {1 .1}
    proc send {ignore arg} {
        sleep .1
        exp_send -s -- $arg
    }
}


set timeout -1
spawn g_energy
match_max 100000
expect -exact "\r"
send -- "9 10 11\r\r"
expect eof
```