CHARMM WORKSHOP 1.0

Arjan van der Vaart
Department of Chemistry
University of South Florida
vandervaart@cas.usf.edu

1. Introduction

CHARMM (Chemistry at HARvard Macromolecular Mechanics) is a computer program for the simulation of biomolecular systems. Initial development of the program started in the early 1970's in the lab of Prof. Karplus at Harvard University, leading to the first molecular dynamics study of a protein (McCammon, Gelin, and Karplus, *Nature* 267, 585 (1977)). Today, more than 20 groups worldwide work on the maintenance and further development of the program. Important features of the program are:

1. It is largely self-contained
The setup, actual calculations, and analysis are all done by the same program.

2. It is fairly complete
Most simulation techniques, force fields, and analysis tools pertinent to biomolecules are part of the CHARMM program.

3. The CHARMM interface works like a programming language
Commands resemble the Fortran programming language, various mathematical manipulations on a selected set of internal and external variables are possible, and commands can include loop structures.

This workshop will provide a short introduction to the CHARMM program. After the workshop, participants should be able to setup, execute, and analyze standard simulation protocols.

2. Help

In the 1970's the working name for the CHARMM program was HARMM (HARvard Macromolecular Mechanics). This name should serve as a warning that CHARMM may be daunting and/or frustrating for users. The program has many options and features; making it a powerful and flexible tool on the one hand, and a difficult program to use on the other. Second, the program is mostly being developed by academic researchers, who prefer to write papers rather than documentation.

There are a number of important sources for help:

1. CHARMM documentation
Each CHARMM release contains a "doc" directory, with ASCII-based .doc documentation files (e.g., nbonds.doc). You can view these files on screen with commands like "less" or "more" or "cat". There is generally a .doc file for each module in CHARMM, for example a .doc file on the DYNAmics command (dynamc.doc), a .doc file on constraints (cons.doc), a .doc file on input/output (io.doc), etc. Use the name of the .doc file or grep to find out which file to read. Although this is the *only* official documentation of CHARMM, the .doc files are nearly always cryptic and sometimes incomplete.

2. CHARMM forum
Since a few years, the CHARMM forum serves as a major source of help for the CHARMM community. This online source can be accessed through the link at the www.charmm.org website. Topics are divided into several categories (click on "Main Index"), and all posts can be searched (click on "Search"; be sure to include a wide time frame for the search, by default this is set to only 1 week). You can post questions yourself after you have become a member of the forum.

3. CHARMM book
"A guide to biomolecular simulations" by O.M. Becker and M. Karplus, Springer 2006 (ISBN-10 1-4020-3586-1) is a guide book for CHARMM. It is based on a lab given at Harvard University in the 1990's and contains many exercises.

4. CHARMM tutorial
The website http://www.ch.embnet.org/MD_tutorial/ contains a small, self-contained tutorial with sample CHARMM exercises. It is a good starting point for new users.

5. CHARMM testcases
Each CHARMM release contains a "test" directory, with many sample input scripts. These scripts are mostly used by developers, to ensure that new code does not "break" existing facilities. The tests require little executation time and clearly demonstrate the syntax of certain commands.

In the future, the "CHARMM GUI" (http://www.charmm-gui.org/) may also become a good source of help; at the moment however, the site seems to be at an experimental stage.

3. Tools

Indispensible for the analysis of molecular simulations are graphical analysis tools, with which trajectories (movies of the positions of atoms) can be viewed. Although CHARMM does contain some visualization modules, it is best to use an external program for this purpose. Several good viewers exists; for example the VMD program (available for free at http://www.ks.uiuc.edu/Research/vmd/).

4. Force fields

Several force fields are available in CHARMM; for example the CHARMM, AMBER, OPLS, and MMFF force fields (see parmfile.doc and rtop.doc). Note that in some cases, a different compilation leading to a different executable should be used. Also, certain options in CHARMM are specific to the CHARMM force field (for example, fast SHAKE for water, and the atomic radii used for the GSBP routines).

The force field is specified by two files: a topology files that describes the connectivity of the atoms (bonds, angles, dihedrals and improper dihedrals), and a parameter file that contains the parameters needed for the energy function. Please read the original literature for an overview of the functional form of the force fields. These files are listed in the CHARMM subdirectory toppar. For the CHARMM force field, the three main force fields are specified in the following files:

1. CHARMM19
toph19.inp and param19.inp
This is an extended atom force field; only polar hydrogen atoms are present, the other hydrogen atoms are captured in their bonded heavy atoms. This force field is mostly used for implicit solvent simulations.

2. CHARMM22
top_all22_prot.inp and par_all22_prot.inp
All atoms are explicitly represented. This force field is used for proteins.

3. CHARMM27
top_all27_prot_na.rtf and par_all27_prot_na.prm
All atoms are explicitly represented; this force field is used for proteins and DNA/RNA (the protein part is equivalent to CHARMM22). Similar files are present for proteins and lipids. Note that the CMAP energy terms are automatically included in these files for the later versions of CHARMM; be careful when using the files from older distributions, since the CMAP energy terms may be missing.

The other files in the toppar directory contain the topologies and parameters for various implicit solvent models (e.g. acepar19.inp for the ACE implicit solvent model), for polarizable force fields (e.g. top_all30_cheq_prot.inp), for certain chemical compounds (e.g. top_all34_ethers.rtf), etc.

The toppar/stream subdirectory contains parameters for a variety of other chemical compounds, for example for cholesterol and retinol. Read the 00readme file in this directory on how to use these various force field files!

The toppar/non_charmm directory contains the force field specification of non-CHARMM force fields. Again, read the 00readme file in this directory on how to use these files!

At this point, it is good to note a major philosophical difference between CHARMM and AMBER. AMBER has an optimistic attitude towards chemistry: since chemical compounds contain similar groups, parameters should be transferable from one chemical compound to the next. Thus, according to AMBER, you can simulate new chemical compounds that have not been explicitly parameterized by using the parameters from existing groups. CHARMM has a negative attitude: chemistry is complex and you cannot expect to accurately predict properties using non-optimized parameters. Thus, according to CHARMM, parameters are not transferable, so you should carefully parameterize any compound not listed in the force field. The truth is probably somewhere in the middle, although it is certainly true that in order to obtain calculated properties of high quality, very careful parameterizations are usually needed. In practice, one can usually take existing parameters if the chemical differences are small, but has to reparameterize (part of) the new molecule when the chemical differences with existing models are large. As always, careful comparisons should be made with experimental data.

5. Exercise 1: Minimizing a molecule

In this exercise we will build the bpti protein using the CHARMM19 force field. We will minimize the protein in vacuum, and calculate the rmsd deviation with the experimental structure.

1. The pdb file

Download the pdb file with code 1BPT from the protein data bank (http://www.rcsb.org). Note that 1 pecularity of CHARMM is, that it cannot read files with mixed large cap - small cap letters: all letters should either be capitals, or small size. Move the file to the workshop/exercise1/ directory, and name it 1bpt.pdb.

Another pecularity of CHARMM is, that it can only read 1 chain per pdb file. Since the 1bpt.pdb file also contains phosphate ions and water, we need to remove these first. Edit the file 1bpt.pdb and remove the phosphate and water molecules.

2. CHARMM input file

CHARMM can be run interactively, but usually it's easier to collect all commands in a single script file. For this exercise, all commands are collected in the file minim.inp. Some notes when you look at minim.inp:

a. All CHARMM input files start with comments. These comments have a "*" in the first column, the last comment should be a single "*".

b. All CHARMM commands are recognized by their first four letters; the remainder is ignored. That's why the first four letters of a command are usually capitalized.

c. You can set variables in CHARMM (see below). Here we specify the location of the toppar directory in the TOPPAR variable. The contents of this variable are later retrieved

by @TOPPAR. Be sure to set the location of this variable properly to your toppar directory.

d. First, you need to read the topology, then the parameter files. As you can see, the commands to open files and to read their contents resemble Fortran. See io.doc for more information on reading files.

e. Comments for the user are preceded by a "!". You can put this at any line.

f. The GENErate command makes an internal representation of the molecule for CHARMM. It extracts all bond, angle, dihedral terms etc for the energy function, and sets the proper parameters. GENErate makes a new segment, the name of the segment is the word directly after the GENErate statement (up to the first four letters). By default, the zwitterionic state of the molecule is built.

g. You can use the PRINt COOR command to print the coordinates; here we use the SELEct command to print just a subset of coordinates (see select.doc). Since uninitialized coordinates have a value of 9999 by default, we can select the atoms for which the x, y, or z coordinate are 9999 to see which atoms we are missing. Note that the SELEction statement is ended by END.

h. A dash "-" at the end of the line means that the command is continued on the next line.

i. The STOP command terminates CHARMM.

3. Running CHARMM

Run this CHARMM script to find out which atomic coordinates are missing:

```
charmm < minim.inp > minim.out
```

where we assume that the path of the charmm executable is known. Examine the output file, and answer the following questions:

Questions: What atoms are missing? What heavy atoms are missing? Were these atoms present in the pdb file?

4. Editing the pdb file

For historical reasons, atom CD1 of isoleucine is called CD in CHARMM. So, to read the coordinates of this atom properly, we have to edit the pdb file and rename isoleucine CD1 to CD. Perform this edit, and rerun CHARMM. Are there still any missing heavy atoms that were present in the pdb file?

5. Building missing atoms

The IC PARAm and IC BUILd commands build the missing atoms using the information from the parameter files (see ic.doc). While hydrogen atoms can also be build with this facility, its generally better to use the HBUIld facility, which uses an energy function to properly place the hydrogens (optimizing for hydrogen bonding). This command should generally be used twice. Remove the first STOP in charmm.inp and build the missing atoms; check to see that all atoms are initialized.

6. Disulfide bonds

Examine the structure of bpti with VMD. Are there any disulfide bonds? We can make these disulfide bonds in CHARMM using patches. For example, the statement

```
PATCH DISU A 12 B 23
```

makes a disulfide bond between residue 12 of segment A and residue 23 of segment B. Insert the proper PATCH statements to make the disulfide bonds of bpti. Rerun CHARMM with the second STOP removed.

7. Minimization

After removing the third STOP, the input file will perform a short minimization. Some notes:

a. First we need to setup the cutoffs for the electrostatic and Van Der Waals interactions. There are many possible ways of doing this, which will severely impact the performance and the accuracy of the calculation. It is pertinent that you read the recommendations in nbonds.doc before doing any calculation! Also, some advanced algorithms are available that may significantly decrease the time needed to perform the calculations (BYCU, BYCC, etc); however, these may not work for all situations.

Questions: Read nbonds.doc with special attention to the recommended options for generating the nonbonded list. What options should we use here (vacuum calculation)?

b. Before doing a minimization, energy needs to be called.

c. Several minimization routines are available in CHARMM: see minimiz.doc. Here we will first use a steepest descent minimization, followed by an adapted base Newton-Raphson minimization. Note that we can add the NBONds options to the MINImize or ENERgy statements.

d. The final coordinates are written as pdb file and as CHARMM coordinate files. The latter have more accuracy; moreover, in this format multiple segments (chains) can be read into CHARMM. View the minimized coordinates with VMD and compare these to the initial structure.

8. Analysis
After removing the fourth stop, we can perform some simple analysis of the minimized structure using the COOR command (see corman.doc). This powerful set of commands manipulates the coordinates. CHARMM has two spaces for the coordinates: the normal set and a comparison set. We fill the comparison set with the experimental coordinates, and then use the COOR ORIE RMSD command to calculate the rmsd.

6. Atom selection

As illustrated in Exercise 1, CHARMM has powerful commands to select a subset of the system. Moreover, these selections can be stored and recalled by name through the DEFIne command. For example:

```
define water select resname tip3 end                    ! water molecules (TIP3P)
define bb select type n .or. type ca .or. type c end    ! protein backbone
print coor select water end          ! print water coordinates
print coor select bb end             ! print backbone coordinates
```

Definitions can also be used in the DEFIne command, for example:

```
define water select resname tip3 end
define centerprot select (.not. water) .and. (point 0 0 0 cut 3) end
```

The file select.doc gives a full overview of the select statement.

7. Variables

In CHARMM, variables can be set and manipulated. For example, in Exercise 1, we introduced a variable TOPPAR to point to the directory where the topology and parameter files are stored. The value of this variable was retrieved by @TOPPAR. In addition to user-defined variables, several internal CHARMM variables can also be retrieved. For example, after performing a RMSD overlay (as in Exercise 1), the value of the rmsd is stored in the variable ?RMS. Note that internal variables are recalled by ?, external (or user-defined variables) by @. Most internal variables are listed in subst.doc.

The variables can be manipulated in several ways. For example, we can perform algebraic manipulations by the CALC command (see miscom.doc):

```
set x 0.0                    ! define variable x, set it to zero
calc y = @x + 3              ! define y = x + 3
calc z  = sin ( @y + ?pi )   ! π is known as the internal variable ?pi
incr x by 1                  ! x=x+1
ener                         ! calculate the energy
```

```
calc avener = ?ENER / ?NATOM    ! average energy per atom
```

Note that the "=" is optional in the CALC statement.

Variables can be initialized in the CHARMM script, or be given as option to the CHARMM command:

```
charmm X=3 < test.inp > test.out
charmm X=4 TOPPAR=/home/me/myself/i/toppar < newtest.inp > newtest.out
```

In addition to these variables, CHARMM also provides access to vector variables. The length of these vectors equals the number of atoms in the system. For some weird reason, access to the vector variables is given through the SCALar command (see scalar.doc). Several of the vectors are predefined, for example "x" for the x coordinates, "dx" for the force in the x direction, "mass" for the atomic masses, etc. In addition to these vectors, the user can define up to 9 additional vectors. These user-defined vectors are called "sca1", "sca2", etc. (up to "sca9"). For example:

```
scalar sca1 = x          ! set sca1 equal to the x-coordinate
scalar sca2 = mass       ! set sca2 equal to the atomic mass
scalar sca2 divi         ! take the inverse of sca2: sca2 = 1/sca2
scalar sca1 prod sca2    ! multiply sca1 by sca2: sca1 = sca1 * sca2 = x/mass
scalar sca1 mult 2       ! multiply sca1 by the number 2
```

The scalar commands are extremely handy for analysis or setup.

8. Program flow

Program flow in CHARMM can be controlled by IF ... ELSE ... ENDIF statements and by GOTO/LABEL statements. Coupled with the use of variables, this enables a programmable interface to CHARMM. For example, "subroutines" can be written in the following way:

```
set return here              ! set exit label
goto subroutine              ! jump to the "label subroutine" statement
label here                   ! exit label
...
stop                         ! stop the program

label subroutine             ! start of "subroutine"
...
goto @return                 ! exit
```

Labels can also contain variables, for example

```
goto subroutine
label here-1                    ! first option
...
stop
label here-2                    ! second option
...
stop

label subroutine
...
calc i @x + 1
goto here-@i                    ! programmable label
```

In a similar way, program flow can be controlled by IF statements:

```
IF @x .lt. 3 THEN
  ...
ELSE
  IF @x .gt. 4 THEN
    ...
  ELSE
    ...
  ENDIF
ENDIF
```

Note that an "ELSEIF" statement does not exist in CHARMM; instead, you can embed multiple IF statements. Since CHARMM can only handle one logical statement at the time, you can also use embedding to resolve the evaluation of multiple logical statements:

```
! IF @a .eq. 1   .and.  @b .eq. 1 THEN       ! cannot be handled, use instead:
IF @a .eq. 1 THEN
  IF @b .eq. 1 THEN
  ...
  ENDIF
ENDIF

! IF @a .eq. 1    .or.   @b .eq. 1 THEN       ! cannot be handled, use instead:
If @a .eq. 1 THEN
  ...
ELSE
  IF @b .eq. 1 THEN
  ...
  ENDIF
ENDIF
```

If you only use 1 statement, you may skip the THEN:

```
IF @a .eq. 1 THEN
   SET b 1
ENDIF

! is equivalent to
IF @a .eq. 1 SET b 1
```

9. System calls

Charmm can call external programs by the SYSTem command. For example:

```
system "echo I am here > logfile"
system "echo ?ENER >> logfile"
```

System commands can be combined with STREAM statements for very flexible program flow:

```
system "echo * charmm stream > tmp; echo * >> tmp"
system "awk '/REACHED/{print "SET X ",$NF;exit;}' charmm.out >> tmp"
system "echo return >> tmp"
stream tmp
calc newx @X +3
```

Running this input file as

```
charmm < charmm.inp > charmm.out
```

will result in the generation of a stream file tmp, which depends on the output of the current job. The stream file sets a variable x, which is read by CHARMM through the STREAM command.

Feedback flow like this is handy in many instances.

One note on different architectures: notably IBM machines are notorious for not correctly managing system commands and/or goto commands. Sometimes you can work around this by putting these commands in seperate stream files; in general, to avoid unpleasant surprises, you should test your job script before starting a production run.

10. Exercise 2: A protein in water

In this exercise we will perform a short molecular dynamics simulation of insulin in water. We will use periodic boundary conditions and the Particle Mesh Ewald method for the long-range electrostatics.

1. Setup
Download the pdb file with access code 4INS from the protein data bank. We will only consider chain A and B; make a file 4insa.pdb that contains chain A and 4insb.pdb that contains chain B. Be sure to rename the isoleucine CD1 atoms (see Exercise 1). Remember that CHARMM is picky about mixing upper and lower case letters. Then execute CHARMM using

```
charmm STAGE=1 < insulin.inp > insulin.out
```

What does the STAGE variable do? Do you understand the program flow? Did CHARMM finish correctly?

2. Histidines
CHARMM needs to know the protonation site of the histidines. There are several options possible; for example protonation of NE, protonation of ND, or double protonation. These residues have different names: HSE, HSD, and HSP, respectively. Normally, one would use experimental data or perform continuum electrostatics or quantum calculations to establish the proper protonation states. Here, we will use the ND protonation state. Edit the appropriate pdb file and change HIS to HSD.

View the molecule with VMD; are there any disulfide bonds to take care of? Edit the file insulin.inp (before the first stop statement) to add missing atoms and to patch the disulfide bonds. Execute CHARMM using

```
charmm STAGE=2 < insulin.inp > insulin.out
```

3. Minimization
It's a good idea to perform a slight minization in vacuum to remove some possible bad steric contacts. Add the minimization commands and execute CHARMM using

```
charmm STAGE=3 < insulin.inp > insulin.out
```

4. Solvation
CHARMM comes with very few pre-equilibrated waterboxes: since each simulated system will be different, the user has to construct such a box by himself. Here we will construct a large cubic box from 8 smaller cubic boxes. We can use the special "read sequ tips" command, to read in a bunch of water molecules. In the input file, the first layer of water boxes is generated: you should edit the input file to add the second layer of water boxes. In the exercise, we leave a small layer of water around the protein, in real life this might be bigger. After deleting waters that overlap with the protein or extend too far from

the protein, we need to add ions such that the total system is neutral (since we will use the particle mesh Ewald method later on). Where did CHARMM print the total charge of the system? Edit the file to add the proper number and kinds of ions for charge neutrality. We continue by saving a psf file (protein structure file): this file contains all the connectivity data needed by CHARMM. After all edits run CHARMM by

```
charmm STAGE=4 < insulin.inp > insulin.out
```

Be sure to visualize the resulting coordinate file with VMD.

5. Molecular dynamics

CHARMM can handle periodic boundary conditions of various box shapes (cubic, orthorhombic, rectangular, etc.); here we will use cubic periodic boundary conditions. These are setup by the CRYStal and IMAGe commands. After this, the nonbonded options are set; in this case we will use the particle mesh Ewald method for the long-range electrostatic interactions. After a short (too short) minimization, we jump right to a molecular dynamics run in the NPT ensemble, using the Nosé-Hoover algorithm. Note that normally, we would perform long minimizations using restraints, slow heat-up and equilibration using restraints in the NVT ensemble, and after equilibration move to NPT. Execute the commands by

```
charmm STAGE=5 < insulin.inp > insulin.out
```

Be sure to visualize the trajectory with VMD and to examine the CHARMM output file.

Questions: How can you check in your output file that the periodic boundary conditions were setup properly? How can you check that the image list was setup properly? What properties does CHARMM print out during the dynamics?

11. Exercise 3: Correl

The correl module (see correl.doc) contains powerful trajectory (including coordinate and velocity trajectories) analysis tools. Options include: the calculation of time-correlation functions, autocorrelation functions, and time series. These functions can be calculated for a large set of user-defined objects, including dipoles, vectors, angles, etc. The functions can also be manipulated by a range of mathematical operations. You should at least browse the correl.doc document to get a sense of the different possibilities. Since the correl module is quite efficient, you should try to rephrase the property that you want to calculate into an object that can be handled by this module.

In this short exercise, we will use correl to calculate some elementary time series from the insulin trajectory generated in Exercise 2.

1. Correl.inp
Carefully read correl.inp. What are we calculating? How many entries do you expect in each of the .dat files (unit 14-16)? Execute this CHARMM script. Check some of the calculated angles with VMD.

2. Rmsd and radius of gyration
Modify the correl.inp such that the backbone rmsd with the experimental structure and the radius of gyration are calculated for each frame of the trajectory.

Question: What should MAXA be for the rmsd/radius of gyration calculation?

12. Bomb levels and print levels

CHARMM normally prints a lot of useful output. The amount of output can be controlled by the PRNLevel command (see miscom.doc):

```
prnlevel 5      ! default print level
prnlevel 0      ! print less than default
prnlevel 7      ! print more than default
```

The maximum print level is believed to be 15, the minimum -15. For parallel jobs, the printlevel can be controlled at the node level (with 0 being the master node):

```
prnlev -15              ! set print level to lowest for all nodes
prnlev 5 node 0         ! then set print level to normal for master node
```

In a similar way, termination of CHARMM due to errors is controlled by the BOMBlevel command:

```
bomb 0                  ! default bomb level
bomb -1                 ! continue when minor errors occur
```

The default BOMBlevel is 0; decreasing the BOMBlevel with cause CHARMM to continue after errors are found. Be very careful in using this option: while some errors are minor, many indicate significant problems with the simulation.

13. Exercise 4: Biased molecular dynamics

CHARMM contains several modules for biasing simulations: methods that force trajectory along a reaction coordinate. Some of these reaction coordinates are predefined geometrical objects, some are energetical; with some the free energy of the transition can be calculated, with some this is much more difficult. Here we will consider the biased molecular dynamics method (BMD; see hqbm.doc). In BMD a half-quadratic biasing potential is added to the energy function, which pushes the trajectory in a forward direction along a pre-defined reaction coordinate. Several variants of the BMD method have been implemented: some of these can also be used for Φ analysis, calculation of protection factors, etc. The user is referred to the original literature (some of which is cited in hqbm.doc) for more information.

We will simulate the opening motion of the small protein adenylate kinase. This protein catalyzes the transfer of a phosphoryl group from ATP to AMP; the mechanism involves large hinge motions of the two binding pockets.

1. Setup
Download 4AKE and 1AKE pdb files from the protein data bank. Which conformer corresponds to the open state, which to the closed state? Edit the SET OPENPDB and SET CLOSEDPDB statements in adk.inp accordingly. Prepare pdb files for chains A; be sure to check the Ile and His residues (you can take the HSD state); name the pdb files 1akea.pdb and 4akea.pdb. Note that to make life simple, we will not consider any of the ligands: the opening motion is known to also occur in the absence of ligands (see Henzler-Wildman *et. al*, *Nature* 450, 913 (2007)). Do the proteins have the same sequence? If not, what should we do?

To save time, we will perform the simulations with the EEF1 implicit solvent model. CHARMM contains many implicit solvent models (for example, GENBORN, ACE, SASA, EEF1, GBSW, GBMV, FACTS, etc); some of which are good for small proteins, some of which are good for bigger proteins, some of which can be used for membranes, some for DNA, etc. Please read the original literature (see the appropriate .doc-s) on these models. Note that for some unknown reason, the topology and parameter files for EEF1 are stored in support/aspara. Edit the adk.inp file to add missing atoms (both for the closed and the open state; see adk.inp), and to add the rmsd overlay command. Run charmm using

```
charmm WHAT=1 < adk.inp > adk.out
```

Questions: What does the COOR INIT command do? Why did we execute this before reading in the open state? Why did we not use a READ SEQU and GENErate command for the open state?

2. BMD simulation

Next, we will perform the actual BMD simulation. We setup the EEF1 solvent model (see eef1.doc) and do a short minimization of the closed state (add the proper commands in adk.inp). We will use the RC1 reaction coordinate for BMD, with a very large force constant, and simulate for 5000 steps. Run CHARMM with

```
charmm WHAT=2 < adk.inp > adk.out
```

View the resulting trajectory with VMD. What motion do you observe?

Questions: What does the RC1 reaction coordinate do (see hqbm.doc)? On what atoms do we apply this? What does the bmd.dat file report? Why did we use Langevin dynamics, and what does this do? Did the trajectory reach the target?

3. Restart

Obviously, we would like to continue this simulation. We can restart the simulation using the RESTart subcommand in DYNamics and the restart file generated in step 2. Of course, we need to also re-setup the BMD. Edit adk.inp to perform a restart of another 5000 steps; be sure to not overwrite any previously generated files. Use dynamc.doc for the proper commands. Execute CHARMM with

```
charmm WHAT=3 < adk.inp > adk1.out
```

View the trajectory. How far did we get?

4. Merging the trajectories

To facilitate the visual analysis, we can merge the two trajectories with the MERGe command (see dynamc.doc). Read the appropriate commands in adk.inp and execute CHARMM with

```
charmm WHAT=4 < adk.inp > adk2.out
```

View the merged trajectory with VMD.

Questions: What does the SKIP option in MERGe do? Can we use the MERGe command for dcd files with a different number of frames? (Try it: for example by generating a dcd file with 1 or 2 frames). How can we merge trajectories of unequal length? (Hint: see the TRAJectory command in dynamc.doc).

14. Exercise 5: Targeted molecular dynamics

In this short exercise, we will look at another method to bias the trajectory along a reaction coordinate. The TMD method (see tmd.doc) uses a holonomic constraint that reduces the rmsd with the target with a predefined value at each time step. Several variants of the TMD method have been implemented (see tmd.doc); here we will use the original method for the closed to open transition of adenylate kinase.

1. TMD simulation
We will use the psf file and the open.cor and closed.cor coordinate files from Exercise 4. First we will minimize the closed structure a bit; add the appropriate commands to tmd.inp. We will use a very large (too large) rmsd step size for TMD to limit the computation time. What is the rmsd step size (see tmd.inp)? The TMD method has only been implemented for the leap frog algorithm. Second, the method has only been implemented using the Berendsen thermostat. Unfortunately, this thermostat is not very good; it has been shown to lead to "cold" and "hot" regions in the system. Better temperature control can be obtained by the Langevin integrator, which couples the system to a stochastic heat bath. How can we run two temperature controls at the same time? Edit the options for the Berendsen thermostat in such a way that the Berendsen is effectively turned off, and only the Langevin thermostat is active. Run the simulation using

```
charmm < tmd.inp > tmd.out
```

View the resulting trajectory with VMD.

Questions: What is printed in the tmd.dat file (unit 88)? Plot the rmsd with the target versus time. Repeat this for the BMD simulation (merged trajectory) of Exercise 4, and compare the results. Which method leads to more physical pathways?

15. Exercise 6: Normal mode analysis

CHARMM has multiple options for harmonic analysis. It can calculate normal modes and quasi-harmonic (or essential) modes; it can also use the elastic network model or block normal mode algorithms for large systems. All this is done by the vibran module, see vibran.doc. Here we will perform normal mode analysis of a small helix.

1. Setup
Download the entry 1o06 from the protein data bank. Prepare the pdb file for CHARMM (taking care of ILE/HIS, only keeping one chain), name this file 1o06.pdb. Edit nm.inp to read the sequence, generate the protein, read in the coordinates, and to build the missing coordinates. To obtain normal modes, we first have to perform a deep minimization of the structure. To keep the protein as close to the experimental structure as possible, we

will use harmonic restraints (see cons.doc) that penalize deviations form the coordinates in the comparison set: you should add the commands to copy the experimental structure to the comparison set. We will loop over restraints, using very stiff force constants at first, and slowly releasing these. At the end, no restraints are left, and a final minimization with respect to the gradient is performed. Note that we use a distance-dependent dielectric constant (see nbonds.doc), to model the effect of water in a poor-men's-way. Read the script carefully, and be sure that you understand each step. Execute CHARMM by

```
charmm WHAT=1 < nm.inp > nm1.out
```

Did the minimization converge? Visualize an overlay of the minimized structure with the experimental structure. Are there differences? How are these quantified? Where are they located? Are they acceptable?

Questions: Repeat the minimization using a constant dielectric (be sure to use the proper NBONDs options). Visualize overlays of the resulting structure with the experimental and the previously minimized structure. What do you see? Explain the result.

2. Normal mode analysis

The VIBRan module (see vibran.doc) performs the normal mode analysis. It has also facilities to manipulate the calculated modes, or to write them to trajectory files. Here we will write various modes to file. Execute the script with

```
charmm WHAT=2 < nm.inp > nm2.out
```

Carefully check the output. Are the first 6 frequencies zero? Visualize some of the modes using VMD. Why did we use a temperature of 2000K for most modes?

16. Exercise 7: Umbrella sampling

CHARMM has various modules for free energy simulations, for example PERT for single topology thermodynamic integration (see pert.doc), BLOCk for dual topology thermodynamic integration (see block.doc), TSM for thermodynamic integration (see tsm.doc), and UMBR for umbrella sampling (see umbrel.doc). In this short exercise we will introduce the ADUM module for adaptive umbrella sampling (see adumb.doc). We will calculate the two-dimensional free energy surface of the alanine-dipeptide as a function of its φ and ψ dihedral angles.

1. Setup

We will use the ACE I implicit solvent model (see ace.doc). Since no experimental structure of the peptide is available, we need to generate an initial structure ourselves. This can be done with the IC module (see intcor.doc): add the appropriate commands in

the umb.inp input file. After specifying the nonbonded list, we need to setup the biasing potentials. These potentials use interpolating functions to connect the various bins. Let's use a bin width of 5 degrees, and 12 trigonometric functions and first order polynomial for the interpolating functions. After adding the appropriate commands (note that all commands from the ADUMb module start with UMBR, not with ADUM; this is a mistake in adumb.doc), and reading the rest of the input file, run CHARMM by

```
charmm < umb.inp > umb.out
```

2. Analysis

If all went well, an extremely long output file was generated. Where can we find the free energy surface? What is the format of the output? How many times was a free energy surface calculated? Write a shell script (or other tool) to extract the various surfaces, and visualize them using Mathematica, xmgrace, gnuplot, or another plotting tool. How do these surfaces change from the one simulation to the next? Where are the free energy basins, where are the barriers? How many pathways can be constructed that connect the various basins?

Questions: The trajectory file is a concatenation of all adaptive umbrella sampling simulations. Extract the sampled φ and ψ dihedral angles from each simulation from the trajectory. Project these onto the free energy surface of the previous run (since the biasing potential equals the inverse of the previously generated free energy surface). What do you observe for the first few runs? What happens in the last few runs? Explain the difference. What do you expect to see when the free energy surface is fully converged?

17. Conclusion

This concludes the CHARMM workshop. You are now ready to simulate your own system! There are many functions of CHARMM that we did not discuss, for example continuum electrostatic calculations, mixed quantum mechanical - classical simulations, Monte Carlo simulations, transition path sampling, etc. etc. I hope you will explore these on your own!