THE UNIVERSITY of TEXAS
HEALTH SCIENCE CENTER AT HOUSTON
SCHOOL of HEALTH INFORMATION SCIENCES

Molecular Dynamics Simulation: Practice II

For students of HI 6327 “Biomolecular Modeling”

Willy Wriggers, Ph.D.
School of Health Information Sciences

http://biomachina.org/courses/modeling/05.html

Where Does Force Field Come From?

Fitted parameters:

- ideal values (bonds, bond angles etc)
- energy constants
- non–bonded parameters (vdW, charges)

Data for the fit:

- *ab initio* quantum mechanical energies for small molecules
- *ab initio* quantum mechanical energies from distorted small molecules
- X–ray structures of small molecules
- spectroscopic data (IR, NMR) for frequencies
- transfer free energies for vdW and charges

The X-PLOR Energy Function

$$
\begin{aligned}
E_{EMPIRICAL} = & \\
\sum_{p=1}^{N} [\; & w^p_{BOND} E_{BOND} + w^p_{ANGL} E_{ANGL} + \\
& w^p_{DIHE} E_{DIHE} + w^p_{IMPR} E_{IMPR} + \\
& w^p_{VDW} E_{VDW} + w^p_{ELEC} E_{ELEC} + \\
& w^p_{PVDW} E_{PVDW} + w^p_{PELE} E_{PELE} + \\
& w^p_{HBON} E_{HBON}]. \qquad (2)
\end{aligned}
$$

Bond Stretching Energy

$$E_{BOND} = \sum_{bonds} k_b (r - r_0)^2 \tag{3}$$

- r actual bond length
- k_b energy constant
- r_0 equilibrium distance

Definition in

- Topology file: explicitely with atom names

  `bond CB HA`

- Parameter file: with atom types, k_b, r_0

  `bond HA CT 1000 1.09`

  the sequence of atom types does not matter

Angle Bending Energy

Sum of two terms:

$$E_{ANGL} = \sum_{angles} (k_\theta(\theta - \theta_0)^2 + k_{ub}(r_{13} - r_{ub})^2) \qquad (4)$$

- First term is standard angle energy.
  - θ actual value of the angle
  - θ_0 equilibrium angle
  - k_θ energy constants
- Second term is Urey-Bradley energy.
  Usually, this term is 0.
  - r_{13} actual 1–3 distance
  - r_{ub} equilibrium 1–3 distance
  - k_{ub} energy constant (default $k_{ub} = 0$)

Angle Bending Energy

- Topology file:
  - explicit definition:

    ```
    angle hb1 cb hb2
    ```

  - automatic definition:

    ```
    autogenerate angles true end
    ```

    at the beginning of topology file
    note: autogenerate does not work in "patches"

- Parameter file:
  - with atom types, k_θ, θ_0 [ub k_{ub}, r_{ub}]

    ```
    angle HA CT HA 500.0 109.0 [ub ... ...]
    ```

Dihedral Angle Energy

$$E_{DIHE} = \sum_{dihedrals} \sum_{i=1,m} k_{\phi_i}(1 + \cos(n\phi_i + \delta_i)) \qquad (5)$$

- ϕ_i actual torsion angle
- k_{ϕ_i} energy constant
- n_i periodicity
- m_i multiplicity
- δ_i *phase shift*

multiplicity: define several terms for same angle (e.g. $n_i = 2, 3$ for sugar puckers)

for $n_i = 0$, different functional form:

$$E_{DIHE} = \sum_{dihedrals} \sum_{i=1,m} k_{\phi_i}(\phi_i - \delta_i)^2 \qquad (6)$$

- ϕ_i actual torsion angle
- δ_i *equilibrium angle*

Dihedral Angle Energy

Definition in

- Topology file:
  - explicit definition:

    `dihedral ca cb cg cd`

  - automatic definition:

    `autogenerate dihedrals true end`

    at the beginning of topology file
    note: autogenerate does not work in "patches"
    note: *all* dihedral angles are generated

- Parameter file:
  - `dihedral CT CT CT CT 1.0 3 0.0`

Multiplicities have to be specified in parameter and topologies

Improper Torsion Energy

$$E_{IMPR} = \sum_{impropers} \sum_{i=1,m} k_{\phi_i}(1 + \cos(n\phi_i + \delta_i)) \qquad (7)$$

- ϕ_i actual improper torsion angle
- k_{ϕ_i} energy constant
- n_i periodicity
- m_i multiplicity
- δ_i *phase shift*

multiplicity: define several terms for same angle (e.g. $n_i = 2, 3$ for sugar puckers)

for $n_i = 0$, different functional form:

$$E_{IMPR} = \sum_{impropers} \sum_{i=1,m} k_{\phi_i}(\phi_i - \delta_i)^2 \qquad (8)$$

- ϕ_i actual improper torsion angle
- δ_i *equilibrium angle*

Improper Torsion Energy

Definition in

- Topology file:
  - *only* explicit definition:

    `improper ca n c cb`
- Parameter file:
  - `improper CA N C CT 500.0 0 35.26439`

Differences between dihedrals and improper torsions:

- Identical functional form
- For both, atoms need not be covalently linked
- Usually, $n_i = 0$ for impropers
  - define chirality
  - define planarity
- Often, impropers do not follow covalent bonds
- Usually, dihedrals around rotatable bonds

Van der Waals Function

Lennard–Jones potential

$$f_{VDW}(R) = \frac{A}{R^{12}} - \frac{B}{R^6} \tag{9}$$

$$f_{VDW}(R) = 4\varepsilon((\frac{\sigma}{R})^{12} - (\frac{\sigma}{R})^6) \tag{10}$$

- A, B Lennard–Jones parameters
- R actual distance between two atoms
- ε well–depth
- $\sigma \propto$ sum of vdW radii

- $E_{min} = -\varepsilon$
- $R_{min} = \sqrt[6]{2}\sigma$
- $A = 4\sigma^{12}\varepsilon$
- $B = 4\sigma^6\varepsilon$

Van der Waals Function

Definition in

- Topology file:
  - through atom types

    ```
    atom CB  type=CT  charge=-0.30 end
    ```

- Parameter file:
  - PARAmeter NONBond statement defines ε_i and σ_i for each atom type

    ```
    nonbonded CT 0.12 3.74 0.12 3.74
    ```

    2nd ε_i, σ_i pair are for 1–4 interactions
  - ε and σ between types i and j are calculated as
    * $\varepsilon = \sqrt{\varepsilon_i \varepsilon_j}$
    * $\sigma = \frac{\sigma_i + \sigma_j}{2}$
  - PARAmeter NBFIx statement defines A, B parameters between different types

    ```
    nbfix CT N 10 1000 10 1000
    ```

    2nd A, B pair are for 1–4 interactions

Electrostatic Function

Coulomb law

$$f_{ELEC}(R) = Q_i Q_j \frac{C}{\varepsilon_o R} \tag{11}$$

Coulomb law w. "distance dependent dielectric"

$$f_{ELEC}(R) = Q_i Q_j \frac{C}{\varepsilon_o R^2} \tag{12}$$

- Q_i atomic charge
- R distance between two atoms
- ε_o dielectric constant
  e.g. for vacuum $\varepsilon_o = 1$, for water $\varepsilon_o = 80$

Electrostatic Function

Definition in

- Topology file:
  - through atomic charges

    ```
    atom CB  type=CT  charge=-0.30 end
    ```

- PARAmeter NBONds statement:
  - ε_o

    ```
    parameter nbonds epsilon=1 end end
    ```

  - straight coulomb law ("constant dielectric"):

    ```
    parameter nbonds CDIE end end
    ```

  - distance dependent dielectric:

    ```
    parameter nbonds RDIE SWITCH end end
    ```

1-4 Interactions

Some force fields have special 1–4 non–bonded parameters

- electrostatic:

```
param nbonds e14fac 0.4 end end
```

- vdW: second pair of ε, σ or A, B parameters in parameter file

```
nonbonded CT     0.12   3.74     0.12   3.74
nbfix     CT N  10   1000       10   1000
```

- set

```
param nbond NBXMOD 5 end end
```

Long-Range Switching/Shifting/Truncation

Non–bonded energies have infinite range.
Coulomb potential is modified by

- Truncation: $\Theta(R - R_{cut})$ (Step function)

  ```
  parameter nbonds TRUNcation CTNB 8.5 end end
  ```

  R_{cut} is non–bonded list cutoff

- Shift entire function such that $f = 0$ at finite R:
  $(1 - \frac{R^2}{R_{off}^2})^2$

  ```
  param nbonds SHIFt CTOFNB 7.5 end end
  ```

  note: not with RDIE

- Switch interaction gently off between R_{on} and R_{off}

  ```
  param nbonds
    SWITch CTONNB 5.5 CTOFNB 9.5
  end end
  ```

  note: not with CDIE

vdW energy is truncated or switched (with VSWItch)

Soft Repulsive Non-Bonded Energy

$$C_{rep}(max(0,(k^{rep}R_{min})^{irexp} - R^{irexp}))^{rexp} \qquad (13)$$

- no electrostatics
- no singularity at 0
- R_{min} is defined from Lennard–Jones potential
- $k^{rep} < 1$ since potential is harder than L–J around R_{min}
- switched on by specifying $k^{rep} > 0$

```
param nbonds repel= 0.8 end end
```

Non-Bonded Lists

Time–saving device:

- calculate non–bonded energy only within R_{cut}
- determine atom pairs within R_{cut} and store on list

```
param nbond CUTNB 10.5 {A} end end
```

- update frequency is determined by TOLErance

```
param nbond toler 0.5 {A} end end
```

note: CUTNB - CTOFNB $>$ 2 TOLE

Summary of Non-Bonded Options

```
parameters nbonds
  CDIEL EPS=1.0  E14FAC=0.4
  SHIFT VSWITCH  CTOFNB=7.5   CTONNB=6.5
  CUTNB=8.5   ATOM   NBXMOD=5   TOLE=0.5
  WMIN=1.5
end end
```

- NBXMOD for setup of non–bonded list:
  - NBXMOD=1 no self interactions
  - NBXMOD=2 + no 1–2 ia (bonds)
  - NBXMOD=3 + no 1–3 ia (bond angles)
  - NBXMOD=4 + no 1–4 ia
  - NBXMOD=5 like 3, but with special 1–4 parameters
- ATOM for atom–based cutoff (GROUP for group–based cutoff)
- WMIN non–bonded warning distance (no effect on energy)

Modification of Energy Function

Switch energy terms on or off:

The FLAGS statement

```
FLAGs EXCLude ELEC END
FLAGs INCLude HBON END
```

note: only BOND ANGL DIHE IMPR VDW ELEC switched on by default
all experimental terms and HBON PELE PVDW have to be turned on explicitely

Modification of Energy Function

Restrict Interactions:

Range of standard interactions are modified with CONStraints INTEraction

```
constraints
  interaction (segid A) (segid A)
  interaction (segid B) (segid B)
end
```

switches off all interactions between segids

Modification of Energy Function

Weighting Interactions:

Weights on specific interactions are modified with CONStraints INTEraction WEIGht

```
constraints
  interaction (all) (not name SG)
      weights * 1 end
  interaction (name SG) (name SG)
      weights * 0 bonds 0.01 vdw 1 elec 1 end
end
```

reduces bond energy for disulfide bridges

note: all desired interactions have to explicitely defined

Modification of Energy Function

Weighting Interactions:

with PARAmeter statement

```
parameter
   bonds (name SG) (name SG) 0.0 TOKEN
end
```

reduces the energy constant of disulfide bridges to 0, but leaves bond length alone ("TOKEN")

note: this is an example of atom based parameter definition (which will be discussed later)

Modification of Energy Function

Modify Charges:

Charges can be modified with VECTOR statement

```
vector do (charge = 0)
   (resn lys or resn arg or resn asp or resn glu)
```

sets all charges for some residues to 0.

Examples of X-PLOR Force Fields

PARAM19:

- standard CHARMM force field
- extended atom (no explicit aliphatic hydrogens)
- typical values for energy constants

```
bond C    C      450.0  1.38
bond CH1E CH3E   225.0  1.52

angl CH1E CH2E CH3E    45.0 113.0

dihe CH1E C    N    CH1E  10.0   2  180.0
dihe X    CH1E CH2E X      1.6   3    0.0

impr CH1E X    X    CH2E  55.0   0   35.26439

nonb C   0.1200  3.7418    0.1000  3.3854
```

Examples of X-PLOR force fields

Typical CHARMM force fields we use:

param19.pro / toph19.pro / toph19.pep
(united atom model, only polar hydrogens)

parallh22x.pro / topallh22x.pro / toph22.pep
(all hydrogen, newer force field)

Download these from class web site!

There are newer versions available (v. 27, not covered here)

The X-PLOR Scripting Language

- variables (symbols)
- if–statements
- loops
- atom selection
- data structure manipulations
- many application statements
- mathematical functions
  for variable and data structure manipulations

Symbol Definitions and ‘EVALuate’

- recognized by $ sign
- symbols are defined *and* manipulated by EVALuate

```
evaluate ($count = 0)
evaluate ($filename = "dg.pdb")
```

- Symbols can be
  - real numbers
  - strings
- the type definition is implicit by usage
- type conversion by encode and decode

```
evaluate ($name = encode($count))
evaluate ($number = decode($name))
```

- $? produces list of all defined symbols

Arithmetic Operations

- Standard operations

```
+  -  *  /  **  ^  (  )
evaluate($number = (5*$count)^(3+$count))
```

- mathematical functions

```
cos  sin  ran ...
evaluate($number = sin( $count*ran() ))
```

Special Symbols

- Fundamental constants

```
$pi  $kboltz
```

- Results of certain operations (incomplete list)
  - PRINt statements define $result

```
print angle
evaluate ($rms_angle = $result)
```

  - VECTor SHOW statements define $result

```
vector show average (x) (all)
evaluate ($x_ave = $result)
```

  - ENERgy, MINImiz and DYNAmics define energy terms

```
energy end
display $ener $bond $angl
```

IF Statements

- basic structures:
  - IF (condition) THEN commands END IF
  - IF (condition) THEN commands
    ELSE commands END IF
  - "case" statement
    IF (condition) THEN commands
    ELSEIF (condition) THEN commands
    ...
    END IF
- can be nested
- note: ELSEIF is *not* ELSE IF

IF Statements

– two "end if" necessary

```
if ( $count eq 1 )
then
   coor copy end
else
   if ($count eq 2)
   then
      coor fit end
   end if
end if
```

– one "end if" necessary

```
if ( $count eq 1 )
then
   coor copy end
elseif ($count eq 2)
then
   coor fit end
end if
```

Loops

- WHILE loop:
  WHILE (condition) LOOP loop–name
  commands
  END LOOP loop–name

```
evaluate ($count = 1)
while ($count le 10) loop main
   evaluate ($count = $count + 1)
end loop main
```

- FOR loop:
  FOR variable IN (set)

```
for $filename in ( "sa_1.pdb" "sa_3.pdb" )
loop main
   coor @@$filename
end loop main
```

- FOR loop:
  FOR variable IN ID (selection)

```
for $loopid in id (all) loop main
   vector show element (x) (id $loopid)
end loop main
```

Atom Selection

- select atoms for certain operations
- selection by atom name
- wildcards and ranges
- selection by atom property
- different "queries" can be connected by AND / OR
- "queries" can be negated by NOT
- parantheses necessary for combinations of AND, OR, NOT

Selection by Atom Name

- The atom name consists of
  - SEGId, segment name defined by SEGMent
  - RESId, residue "number" (also 48b etc!)
  - RESName, residue name (ALA, VAL...)
  - NAME, atom name (N, CA...)

```
coor select (resid 5 and name hn) ... end
coor select ((resid 5 or resid 7) and name hn) ..
coor select (resid 5:7 and not name h*) ... end
```

- Atoms can also be selected by
  - CHEM (atom type defined in topology)
  - ID (internal number)

Wildcards and Ranges

- wildcards and ranges can be used for
  - SEGId
  - RESId
  - RESName
  - NAME
  - CHEM
- ranges are lexigraphical order
- indicated by ":"

```
coor sele (name ha:hg#) ... end
```

selects ha, hb1, hb2, hg1, hg2

- wildcard hierarchy
  - "*" any string (abcd, 78, 8u)
  - "#" any number (2, 43, 39987)
  - "%" any character (a, 6, j)
  - "+" any digit (0, 1, ... 9)

Selection by Atom Property

- ATTRibute selects on any atom property (coordinates, derivatives, mass, charge, ...)

```
coor sele (attribute charge > 0) ... end
```

- AROUnd, SAROund select atoms within cutoff of specified atoms

```
coor sele ((resid 1 and name ca) around 5.0) ... end
```

SAROund selects atoms also in symmetry mates

- POINt ... CUT selects atoms around point

```
coor sele (point (3.0 4.0 5.0) cut 5.0) ... end
```

Atom Selection

BYREsidue

- BYREsidue (selection)
  selects all atoms in a residue

```
coor sele= (byres( point (0 0 0) cut 5.0) )
```

STOREi and RECALLi

- Atom selections can be stored and used later

```
vector iden (store1) (name ca)

coor sele= (store1) ...
coor sele= (recall1) ...
```

‘VECTor’ Statement

The VECTor statement allows analysis and manipulation of atom properties and names.

- VECTor SHOW ELEMent (AtomArray) (selection)
  lists elements and defines $result
- VECTor SHOW AVERage (AtomArray) (selection)
  VECTor SHOW RMS (AtomArray) (selection)
  VECTor SHOW SUM (AtomArray) (selection)
  VECTor SHOW NORM (AtomArray) (selection)

```
vector show element (resid)
   (name ca and (resid 5 and name ca) around 5.0)

vector show average (x) (name ca)
```

- VECTor DO (expression) (selection)

```
vector do (b = b + x^2 + y^2 + z^2) (all)
```

- VECTor IDEN (STOREi) (selection)
  defines a STORE to be used in atom selection later

3D Vectors and Matrices

- 3D vectors can be defined explicitely, or through atom selections

```
coor translate vector= (1 0 0) end
coor translate
        vector= (head=(resid 1 and name cb)
                 tail=(resid 1 and name ca))
        distance= 5.0
```

- 3x3 matrices can be defined by

```
coor rotate
   center= (0 0 0)
   matrix= AXIS (head=(resid 1 and name cb)
                tail=(resid 1 and name ca))
           90.0
```

- or by Euler angles, Lattman angles, Quaternions, Spherical angles
- or explicitely

```
coor rotate
   center= (0 0 0)
   matrix= (1 0 0) (0 1 0) (0 0 1)
```

Output Files

- DISPLAY files
  for DISPlay statements
  open with SET DISPlay filename END
- PRINT files
  for info from PRINt statements (e.g. PRINt ANGLes)
  open with SET PRINt filename END
- coordinate, structure, parameter files
  with WRITE COOR (structure...) OUTPut= filename end
- trajectory files

Script Example I: RMS Difference

```
set display rmsd.disp end

evaluate ($maxcount = 10)

evaluate ($count = 1)
for $filename in ( @@file.list ) loop fit

   coor @@$filename
   if ($count eq 1) then
      coor copy end
   end if

   coor sele (name ca) fit end
   coor sele (name ca) rms end

   display structure $count $filename rms difference $result A

   if ($count ge $maxcount) then
      exit loop fit
   end if

   evaluate ($count = $count + 1)
end loop fit
```

Script Example I: RMS Difference

file.list contains a list of files, for example ordered by energy

```
"sa_1.pdb"
"sa_6.pdb"
...
"sa_67.pdb"
```

The display file rmsd.disp will look like this

```
structure 1 sa_1.pdb rms difference 0 A
structure 2 sa_6.pdb rms difference 1.245 A
...
structure 10 sa_67.pdb rms difference 1.87 A
```

Script Example II: B-Factors

```
set display rmsfluc.disp end

for $loopid in id (name ca) loop rms

   vector show element (resid) (id $loopid)
   evaluate ($resid = $result)
   vector show element (resn) (id $loopid)
   evaluate ($resn = $result)
   vector show norm (b) (byresidue(id $loopid) and not hydrogen)
   evaluate ($rmsfluc = $result)

   display $resn $resid $rmsfluc

end loop rms
```

Script Example III: Solvate in Water

Solvate.inp (download from class web site)

The script

- measures the extent of the structure
- overlays a water box over the structure the necessary number of times
- deletes water molecules too close to protein
- deletes water molecules too far from protein

Geometric and Energetic Analysis

Print Statement

The print statement provides a listing of pertinent information about selected bonds, angles, dihedrals, impropers, and hydrogen bonds. The listing of the print statement can be further reduced by use of the constraints interaction statement (see above).

The following example shows how to print all covalent bonds of residue 40 that deviate by more than 0.1 Å. It also provides information about the rms deviation of the selected bonds:

```
constraints interaction
        ( residue 40 ) ( residue 40 )
end
print threshold=0.1 bonds
```

(for details see online X-PLOR manual, chapter 5)

Geometric and Energetic Analysis

Pick Statement

The pick statement allows one to pick specific energy terms independent of the actual list of bonds, angles, dihedrals, and impropers in the molecular structure.

The following example picks the bond angle value among three arbitrary atoms. Note that the atoms do not have to be bonded to each other, and no parameters are required for this property. Of course, if one tried to pick a different property here, the program would come back with an error message:

```
pick angle
       ( residue 1 and name c )
       ( residue 32 and name n )
       ( residue 5 and name ca )
       geometry
```

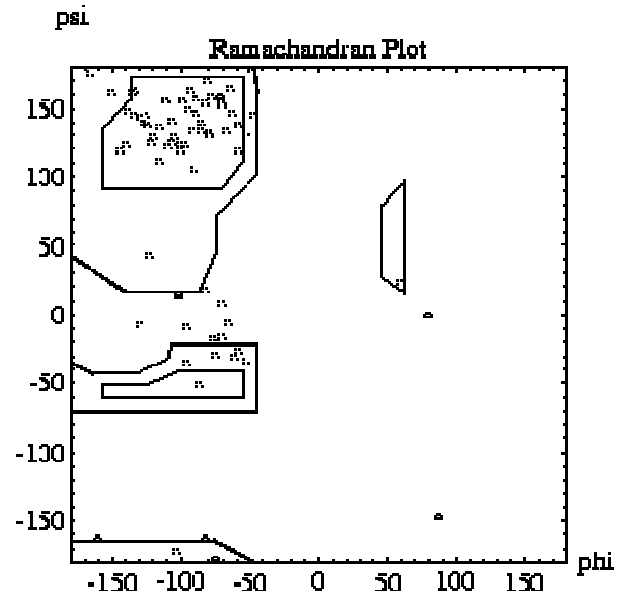
(for details see online X-PLOR manual, chapter 5)

Geometric and Energetic Analysis

Advanced Analysis

X-PLOR also provides scripts and tools for:

- Analysis of non-bonded information
- Accessible Surface Area
- Ramachandran plot



(for details see online X-PLOR manual, chapter 5)

Cartesian Coordinates

Coordinate Manipulation and Analysis

The coordinate statement is used to read and manipulate coordinates, such as least-squares fitting to a comparison coordinate set, rotation, translation, and conversion between fractional and orthogonal coordinates.

In the following example, the first set is least-squares fitted to the second (comparison) set using C-alpha atoms only. Note that all atoms are translated and rotated. Then the rms difference between the two sets is computed for backbone atoms and stored in the symbol $1. Finally, the individual rms differences are printed for all backbone atoms that show rms differences greater than 1Å and the fitted coordinates (C-alpha only) are written to a file.

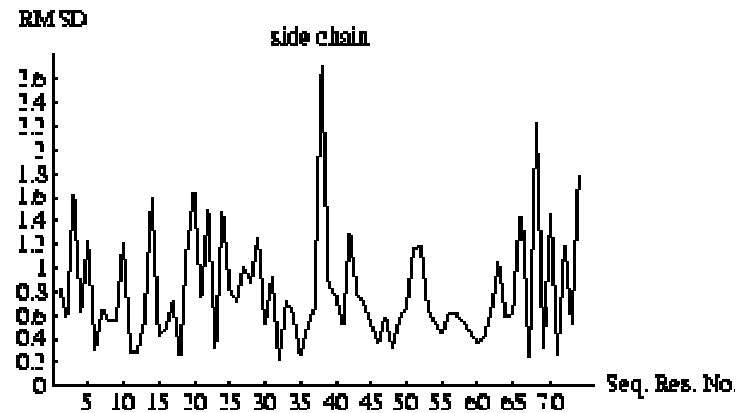
```
coordinates fit selection=( name ca ) end
coordinates rms selection=( name ca or name n or name c ) end
evaluate ($1=$result)
vector show ( b )
   ( attribute b > 1.0 and ( name ca or name n or name c ))
write coordinates output=outfile.pdb selection =(name ca) end
```

(for details see online X-PLOR manual, chapter 6)

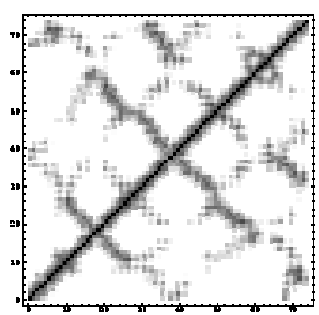
Cartesian Coordinates

Advanced Coordinate Manipulation

X-PLOR also provides scripts and tools for:



- Analysis of rmsd vs. residue number
- Auto-orienting and centering
- Any kind of translating / rotating coordinates
- Computing the radius of gyration $R_{gyr} = \sqrt{< (r_i - < r_i >)^2 >}$
- Distance matrix calculations



(for details see online X-PLOR manual, chapter 6)

Constraints & Restraints: What's the Difference?

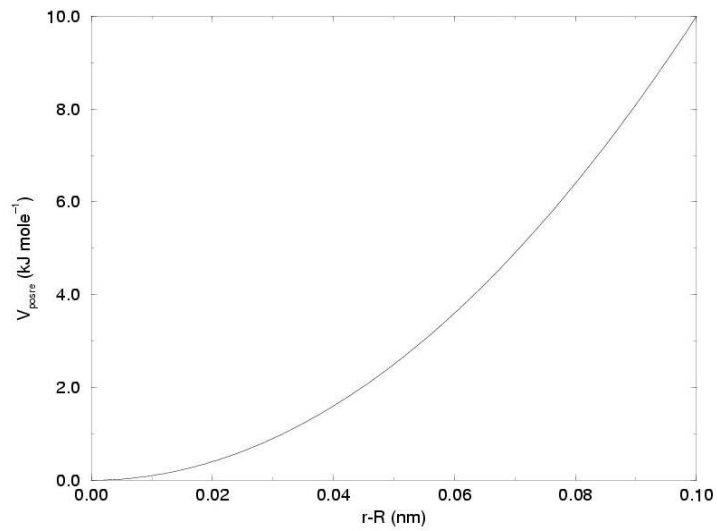
- A constraint is a requirement that the system is forced to satisfy
- With a restrain the system is able to deviate from the desired value

Position Restraints

To a fixed reference position $\boldsymbol{R}_i$: used during equilibration to avoid dramatic rearrangements of some parts of the system

e.g. position restraints to protein after insertion in bilayer to re-equilibrate the lipids

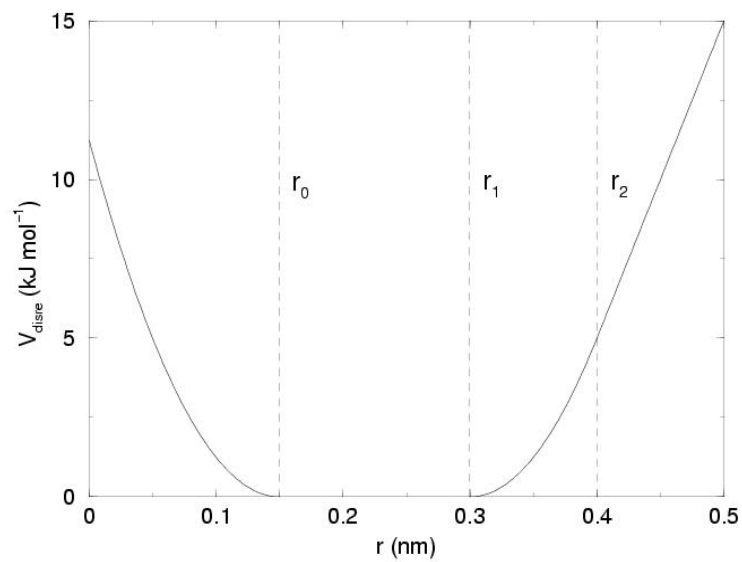
$$V_{pr}(\boldsymbol{r}_i) = \frac{1}{2} k_{pr} |\boldsymbol{r}_i - \boldsymbol{R}_i|^2$$



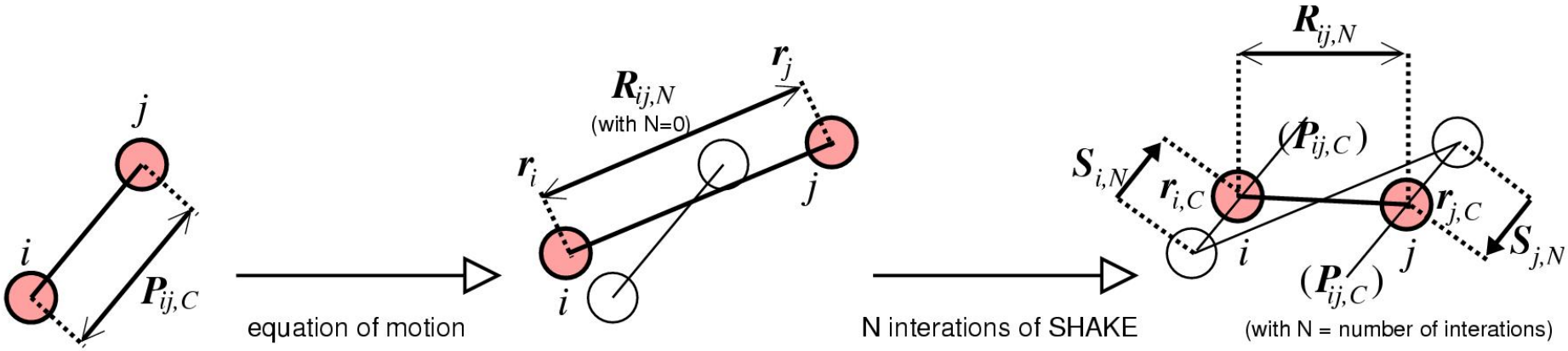
Distance Restraints

They add a penalty to the potential when the distance between specified pairs of atoms exceeds a threshold value

$$V_{dr}(r_{ij}) = \begin{cases} \frac{1}{2}k_{dr}(r_{ij}-r_0)^2 & \text{for} \quad r_{ij} < r_0 \\ 0 & \text{for} \quad r_0 \leq r_{ij} < r_1 \\ \frac{1}{2}k_{dr}(r_{ij}-r_1)^2 & \text{for} \quad r_1 \leq r_{ij} < r_2 \\ \frac{1}{2}k_{dr}(r_2-r_1)(2r_{ij}-r_2-r_1) & \text{for} \quad r_2 \leq r_{ij} \end{cases}$$



Distance Constraints: SHAKE Algorithm



Coordinate *Re*straints

Advanced Coordinate Manipulation

X-PLOR provides scripts and tools for:

- Point restraints (e.g. attaching atoms to points by a harmonic spring)
- Planarity restraints
- Dihedral Angle Restraints

(for details see online X-PLOR manual, chapter 7)

Coordinate Constraints

Advanced Coordinate Manipulation

X-PLOR provides scripts and tools for:

- Fixing atomic positions

The following example fixes all C-alpha atoms:

```
coordinates fix=( name ca ) end
```

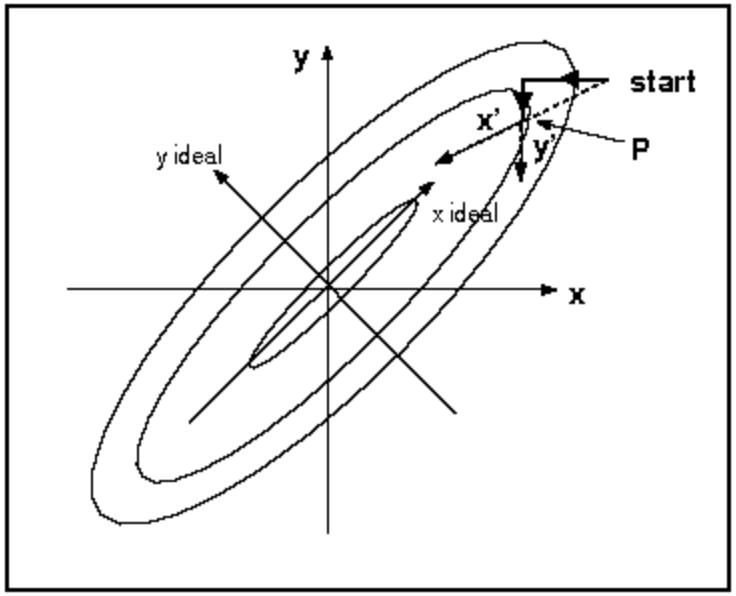
- Fixing distances between atoms (SHAKE)

(for details see online X-PLOR manual, chapter 8)

Energy Minimization

- Minimization:
  Powell's conjugate gradient minimization
- Molecular dynamics:
  (numerical solution of Newton's equations of motion)
  with temperature variation (simulated annealing)
- Rigid body dynamics
- Rigid body minimization
  (with Powell's method)
- Grid search
  through command language
- Monte Carlo simulated annealing
  through command language

Powell's Conjugate Gradient Method



- uses gradient information
- a "complete" minimization is a series of one–dimensional minimizations (one for each degree of freedom)

Powell's Conjugate Gradient Method

- started by MINImize POWEll
- Minimization is performed until one convergence criterion is met.
- NSTEp: maximum number of steps
- TOLGradient: target norm of gradient
- Other parameters:
- DROP: expected initial drop in energy (default 0.001, optimal value 10...100)
- NPRInt: Information is printed every NPRINT steps

Notes:

- Minimization defines variables $ener, $grad, $bond... of energy terms that are turned on with FLAG statement
- Minimization often terminates with "Line search abandoned".

A Simple Energy Minimization

To minimize, we need

- PSF file
- energy parameters
- starting coordinates (X–PLOR PDB format)

```
structure @@diala.psf end

parameter @@TOPPAR:parallhdg.pro end

coor @@diala.pdb end

mini powell nstep= 50 end

REMARK after 50 steps powell
write coor output=diala_min.pdb end

stop
```

Example: Building Hydrogens

```
flags exclude vdw elec end   ! For quick hydrogen building

hbuild
        selection = (hydrogen)
        phistep=20
end

flags include vdw elec end

constraints fix = (not(hydrogen))
end

minimize powell
        drop = 20
        nstep = 1000
end

write coordinate output = withhydrogens.pdb end
```

(for details see online X-PLOR manual, chapter 6)

Demo

minimization demo: download from class web site

1. run the following and wait until it exits (this will create the PSF): *xplor < prepare.inp > prepare.log &*
2. read and understand minimize.inp
3. run the following: *xplor < minimize.inp*
4. become familiar with and understand the output
5. you could also do the following to create a log file and look at it as it is being created:

xplor < minimize.inp > minimize.log & ; tail -f minimize.log

6. If you're adventurous the prepare.inp script explains how the PSF is generated:

- first we did not discuss sugars (polysaccharides) in class, you can read more here:
http://www.accessexcellence.org/RC/VL/GG/garland_PDFs/Panel_2.03b.pdf
- we're dealing with sugars and protein here so suger parameters/topolgy files need to be read also
- segments CHA, CHB, CHC, CHD are 4 protein chains with sequence read from start.pdb
- segment CHE contains sugars in linear sequence starting at residue number 5
- an explicit sequence is specified with fake residues 1-4 that are deleted later before reading the coordinates
- then disulfide bond patches (cystein cross-links) are applied to the protein chains
- sugars are highly branched so we need to add also a considerable number of sugar cross-links also
- finally four sugar puckers have to be patched to the alpha from the standard beta form
- hydrogen building
- quick minimization
- writing psf and pdb

7. If you want look at the structures with VMD...

Molecular Dynamics

Newton equations

Molecular dynamics is the numerical solution of Newton's equations of motion

$$m_i \frac{d^2 x_i}{dt^2}(t) = -\nabla_{x_i} E_{TOTAL} \tag{1}$$

- second order differential equation
- masses m_i are defined in topology file
- E_{TOTAL} is the X–PLOR energy function (force field and experimental terms)

The Verlet Algorithm

- derived from a linear approximation

$$x_i(t+h) = 2x_i(t) - x_i(t-h) + F_i(t)\frac{h^2}{m_i} \qquad (2)$$

- velocities are calculated by

$$v_i(t) = \frac{1}{2h}(x_i(t+h) - x_i(t-h)) \qquad (3)$$

- very simple, very stable
- started by

```
DYNAmics LANGevin ... END
DYNAmics VERLet   ... END
```

Example: Newtonian Dynamics

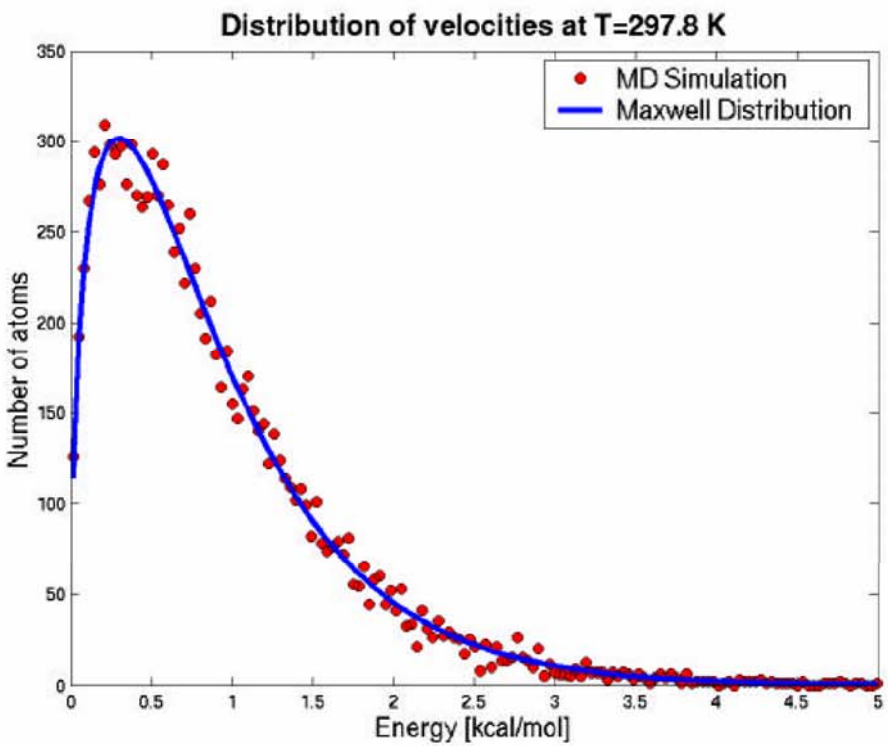
```
! initialize velocities
dynamics verlet
        nstep=1 ! nr. integration steps
        timestep=0.0000001 ! integration step, ps
        iasvel=maxwell ! initial velocity assignment
        first = 300 ! temperature
end

! production run
dynamics verlet
        nstep=1000 ! nr. integration steps
        nprint=10 ! energy values printed every nprint steps
        timestep=0.001 ! integration step, ps
        iasvel=current ! current temperature
        iprfrq=50        ! energy statistics printed every iprfrq steps
        ntrfrq=0 ! Frequency of removal of COM motion
end
```

NOTE: Normally one would not initiate temperature directly at 300K but perform slow heating first.

Maxwell-Boltzmann Velocity Distribution

An NVT ensemble simulation at T=300K was terminated and continued as an NVE ensemble simulation. After an equilibration phase, the distribution of velocities from all atoms was determined (red) and fitted to the Maxwell velocity distribution (blue); the best fit corresponded to a temperature T=297.8 K.



Maxwell distribution for kinetic energies

$$f(\epsilon_k) = \frac{2}{\sqrt{\pi}} \frac{1}{(k_B T)^{\frac{3}{2}}} \sqrt{\epsilon_k} \exp\left(-\frac{\epsilon_k}{k_B T}\right)$$

Terminology:

NVT: constant number of particles, volume, temperature (temp. coupling)

NVE: constant number of particles, volume, energy (Newtonian dynamics)

Temperature Control I: Velocity Scaling

Prerequisites:

Need to set the following in *dynamics verlet* statement:

ieqfrq =<integer>
frequency of velocity rescaling

first=<real>
temperature

iscvel=<integer>
velocity rescale mode (default 0)

The simplest method is periodic explicit rescaling

- calculate actual temperature from velocities
- rescale with
- if ISCVel=0, overall scaling

$$v_i^{new} = v_i^{old}\sqrt{3nk_BT_o/ < \sum_i mv_i^{old}(t)^2 >} \quad (5)$$

- if ISCVel=1, individual scaling

$$v_i^{new} = v_i^{old}\sqrt{3k_BT_o/ < mv_i^{old}(t)^2 >} \quad (6)$$

- where
  n is the number of free atoms in the system
  k_B is Boltzmann's constant
  T_o is the target temperature
  $V_i^{old}(t)$ is the velocity of atom i

Temperature Control II: Langevin Dynamics

- X–PLOR can calculate Langevin dynamics

$$m_i \frac{d^2 x_i}{dt^2}(t) = -grad_{x_i} E + f_i(t) - m_i b_i \frac{dx_i}{dt}(t) \quad (4)$$

- in addition to Newton equation, friction terms:
- f_i is a random force on atom i
- $m_i b_i \frac{dx_i}{dt}(t)$ is a velocity dependent friction term with friction constant b_i
- the b_i are defined in the FBETA atom property

```
VECTor DO (FBETA = 10.0) (ALL)
```

- Langevin dynamics is turned on by choosing
  RBUF (everything outside RBUF)
  ILBFRQ > 0
- can be used as temperature control

Example: Langevin Dynamics

```
! initialize

vector do (fbeta=50) (not(hydrogen))
vector do (fbeta=0) (hydrogen)

dynamics verlet
         iasvel = maxwell
         first = 10
         nstep = 1
         timestep = 0.000001
end

evaluate ($step = 1) ! stepping variable used, 1ps run
evaluate ($tlan = 10) ! temperature variable used
```

Example: Langevin Dynamics

```
while ($step < 31) loop main ! 30 ps heating to 300K
  dynamics langevin
         ntrfrq = 100
         nstep = 1000
         ilbfrq = 1000
         timestep = 0.001
         iasvel = current
         iprfrq = 50
         nprint = 10
         tbath = $tlan
         rbuf = 0
  end
  ! write coordinates every ps
  evaluate ($outfile = "langevin_"+ encode($step) + ".pdb")
  write coordinates
         output = $outfile
  end
  evaluate ($step = $step + 1)
  evaluate ($tlan = $tlan + 10)
end loop main
```

Temperature Control III: Berendsen Coupling

- Berendsen's method:
  Langevin dynamics
  with adjustable friction coefficient
  and zero random force

$$b_i = b_i^I (T_o/T - 1) \tag{7}$$

- if $T > T_o, b_i$ is positive (cooling)
- if $T < T_o, b_i$ is negative (heating)
- switch on by

```
vector do (fbeta = 10) (not hydro)
dynamics verlet
   ...
   TCOUpling = TRUE
   ...
end
```

Temperature Control III: Berendsen Coupling

Some important parameters:

- TCOUPling : switch on Berendsen's method
- TBATh : target temperature for Langevin and Tcoupling
- FIRSttemp : initial temperature
- IASVelocity : initial velocity distribution
- NPRInt : output frequency
- NSTEp : number of steps
- TIMEstep : intergration step
- NTRFrq : removal of COM motion

Example: Slow Berendsen Cooling

```
evaluate ($bath = 1000)
while ($bath > 50) loop cool
  evaluate ($bath = $bath - $tempstep)

  dynamics  verlet
     nstep=1000 time=0.005
     iasvel=current
     tcoup=true tbath=$bath
     nprint=$nstep iprfrq=0
     ntrfrq=9999
  end

end loop cool
```

assumes initial velocities have been set

Resources and Further Reading

WWW:
http://xplor.csb.yale.edu

Books:
Chapters 1-11 in:
A.T. Brunger, X-PLOR Version 3.1 (Yale U Press, 1992)
online free at http://alpha2.bmc.uu.se/local_html/xplor_mirror.html

Acknowledgement:
Michael Nilges Primer
http://www.pasteur.fr/recherche/unites/Binfs/xplor/primer/
Lectures 2-4