Analysis of MD Simulations

Overview

- Looking at your trajectory
- Groups in analysis
- Root mean square deviations and fluctuations
- Radius of gyration and distances
- Hydrogen bonds
- Secondary structure analysis
- Free energy surfaces
- Principal component analysis:

– using Cartesian coordinates

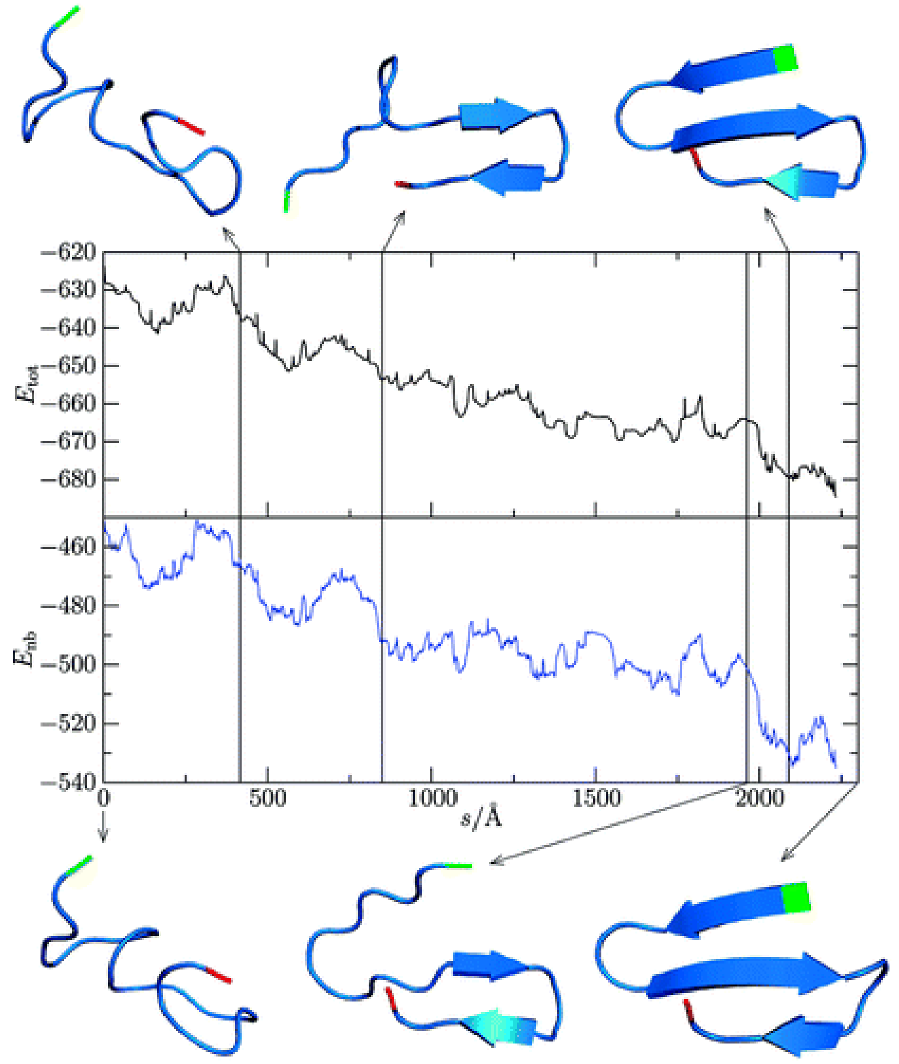
– using dihedral angles

- Clustering

Looking at your trajectory

- **Your MD simulation has finished, so what to do with it?**
- Before doing any analyzing for the results you are interested in: **look at your trajectory first** using **VMD**.
- Look at the energies and other properties, such as total pressure, pressure tensor, density, box-volume and box-sizes.
- Energies can be split into contributions, like potential, kinetic or total energy, or individual contributions, like Lennard-Jones or dihedral energies.
- It is alsways a good idea to look at the energies and pressure **already during the simulation**! If something goes wrong you can realize it before you spent all the precious computing time.

Looking at your trajectory



Example:

Total potential energy (upper) and nonbonding component (lower) – both in kcal/mol – as a function of the folding pathway for the three-Stranded β-sheet peptide Beta3s

GROMACS facilities:
Looking at your trajectory

g_energy:

Description:
g_energy extracts energy components or distance restraint data from an energy file. The user is prompted to interactively select the energy terms she wants. See all accepted options with '**g_energy -h**'

Invocation:

```
~$ g_energy -f input.edr -o output.xvg
```

You can then select the term(s) you want from the returned list

g_traj:

Description:
g_traj plots coordinates, velocities, forces and/or the box. With -com the coordinates, velocities and forces are calculated for the center of mass of each group.

Invocation:

```
~$ g_traj -f input.xtc -s input.tpr -ox output.xvg
```

Groups in analysis

- Often it is advantageous to use groups of atoms for the analysis.
- Consider a simulation of a binary mixture of components *A* and *B* for which we want to calculate the radial distribution function $g_{AB}(r)$:

$$4\pi r^2 g_{AB}(r) = V \sum_{i\in A}^{N_A} \sum_{j\in B}^{N_B} P(r)$$

where *V* is the volume and *P*(*r*) is the probability of finding a B atom at distance *r* from an *A* atom.

- By having the user define the atom numbers for groups *A* and *B* in a simple file, we can calculate this g_{AB} in the most general way
- Groups can consist of
  - a series of atom numbers
  - molecule numbers
  - a series of angles, dihedrals, bonds or vectors (in a molecule)

GROMACS facilities: Defining groups

Default groups:

System	all atoms in the system
Protein	all protein atoms
Protein-H	protein atoms excluding hydrogens
C-alpha	C_α atoms
Backbone	protein backbone atoms: N, C_α and C
MainChain	protein main chain atoms: N, C_α, C and O, including oxygens in C-terminus
MainChain+Cb	protein main chain atoms including C_β
MainChain+H	protein main chain atoms including backbone amide hydrogens and hydrogens on the N-terminus
SideChain	protein side chain atoms; that is all atoms except N, C_α, C, O, backbone amide hydrogens, oxygens in C-terminus and hydrogens on the N-terminus
SideChain-H	protein side chain atoms excluding all hydrogens

GROMACS facilities: Defining groups

Default groups:

Prot-Masses	used in virtual site constructions); only included when it protein atoms excluding dummy masses (as differs from the Protein group
Non-Protein	all non-protein atoms
DNA	all DNA atoms
RNA	all RNA atoms
Water	water molecules (names like SOL, WAT, HOH, etc.) See residuetypes.dat for a full listing
non-Water	anything not covered by the Water group
Ion	any name matching an Ion entry in residuetypes.dat
Water_and_Ions	combination of the Water and Ions groups
molecule_name	for all residues/molecules which are not recognized as protein, DNA, or RNA; one group per residue/molecule name is generated
Other	all atoms which are neither protein, DNA, nor RNA

GROMACS facilities: Defining groups

- Empty groups will not be generated.
- Groups that do not belong to the default groups can be generated using **make_ndx**:

Invocation:

```
~$ make_ndx -f input.pdb -o output.ndx
```

An internal editor then opens. Apart from the 9 listed default groups one can define additional groups. For example, say we are interested in defining a separate group for the arginine residue. Typing letter 'l' we can see that the arginine residue as a residue number '16'. To create a new group for it, type

```
r 16
```

Then press the return key to see the update groups. The new group now occupies group 10. Pressing 'q' followed by the return key saves the newly created index file and exits the editor.

Root mean square deviations in structure

- The root mean square deviation (RMSD) of certain atoms in a molecule with respect to a reference structure, $\mathbf{r}^{\text{ref}}$, is calculated as

$$\mathrm{RMSD}(t) = \left[\frac{1}{M} \sum_{i=1}^{N} m_i \left| \mathbf{r}_i(t) - \mathbf{r}_i^{\text{ref}} \right|^2 \right]^{1/2}$$

where $M=\Sigma_i\, m_i$ and $\mathbf{r}_i(t)$ is the position of atom i at time t after least square fitting the structure to the reference structure.

- Fitting does not have to use the same atoms as the calculation of the RMSD; e.g. a protein is usually fitted on the backbone atoms but the RMSD can be computed for the backbone or for the whole protein.
- Often the starting structure is taken as reference structure.
- Alternatively, for the study of protein folding the folded structure known from experiment is used as reference.

Root mean square fluctuations

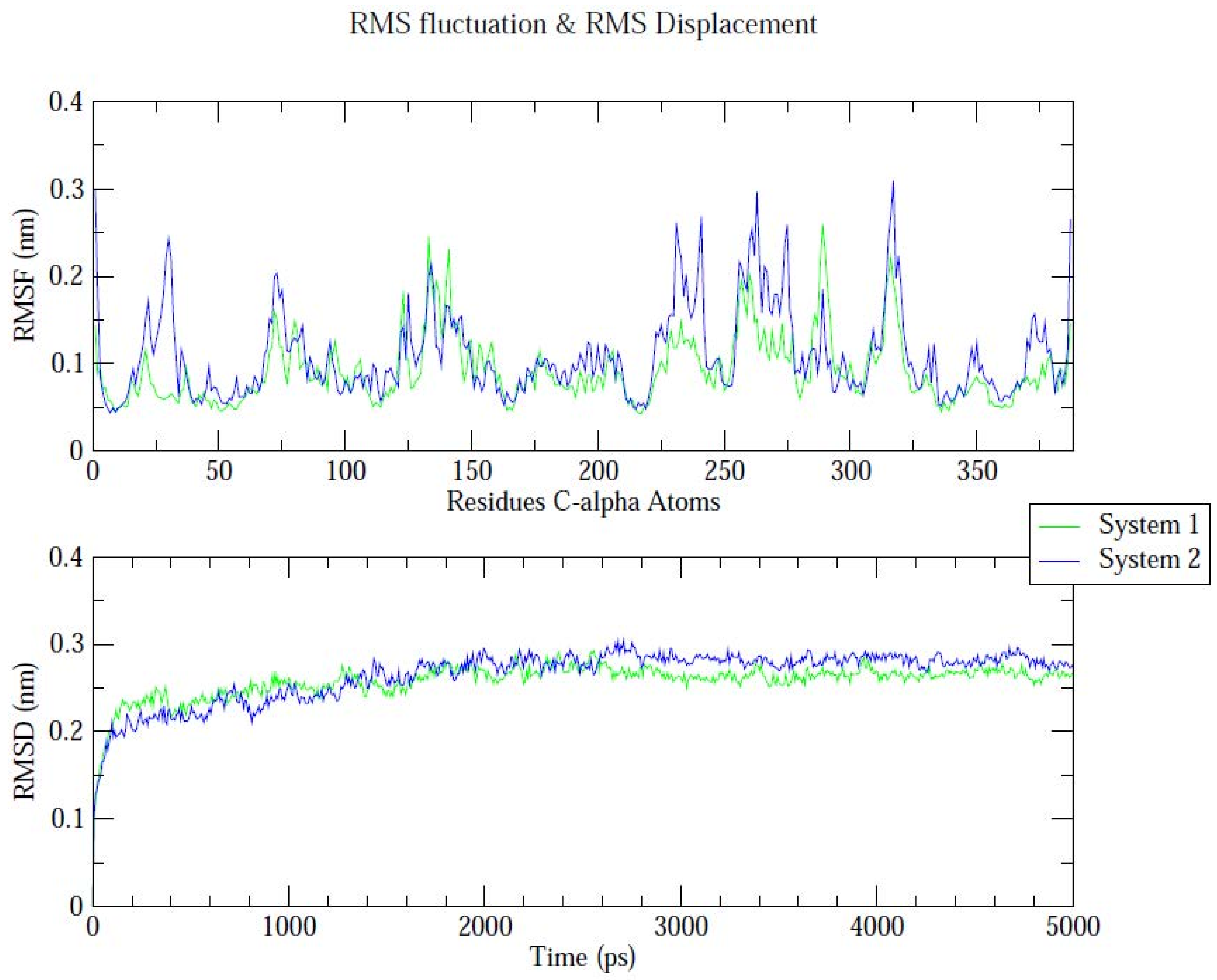
- The root mean square fluctuation (RMSF) is a measure of the deviation between the position of particle *i* and some reference position:

$$\mathrm{RMSF}_i = \left[\frac{1}{T} \sum_{t_j=1}^{T} |\mathbf{r}_i(t_j) - \mathbf{r}_i^{\mathrm{ref}}|^2 \right]^{1/2}$$

  where *T* is the time over which one wants to average and $\mathbf{r}_i^{\mathrm{ref}}$ is the reference position of particle *i*.

- Typically this reference position will be the time-averaged position of the same particle *i*, i.e., $\mathbf{r}_i^{\mathrm{ref}} = \bar{\mathbf{r}}_i$
- Difference between RMSD and RMSF: The latter is averaged over time, giving a value for each particle *i*. For the RMSD the average is taken over the particles, giving time specific values.

Example: RMSF and RMSD



GROMACS facilities: RMSD and RMSF

g_rms

Description:
g_rms compares two structures by computing the root mean square deviation with each structure in the trajectory (-f) compared with a reference in the structure file (-s). After typing the command it is necessary to select an index group for least square fitting and for RMSD calculation.

Invocation:

```
~$ g_rms -f input.xtc -s input.pdb -o rmsd.xvg
```

g_rmsf

Description:
g_rmsf computes the root mean square fluctuation (RMSF, i.e. Standard deviation) of atomic positions after (optionally) fitting to a reference frame. A group is relected for RMSF calculation after entering the comman

Invocation:

```
~$ g_rmsf -f input.xtc -s input.pdb -ox rmsf.xvg
```

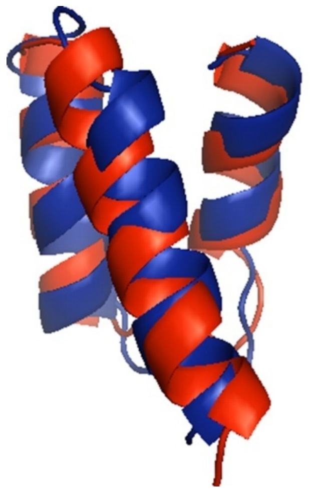
Radius of gyration

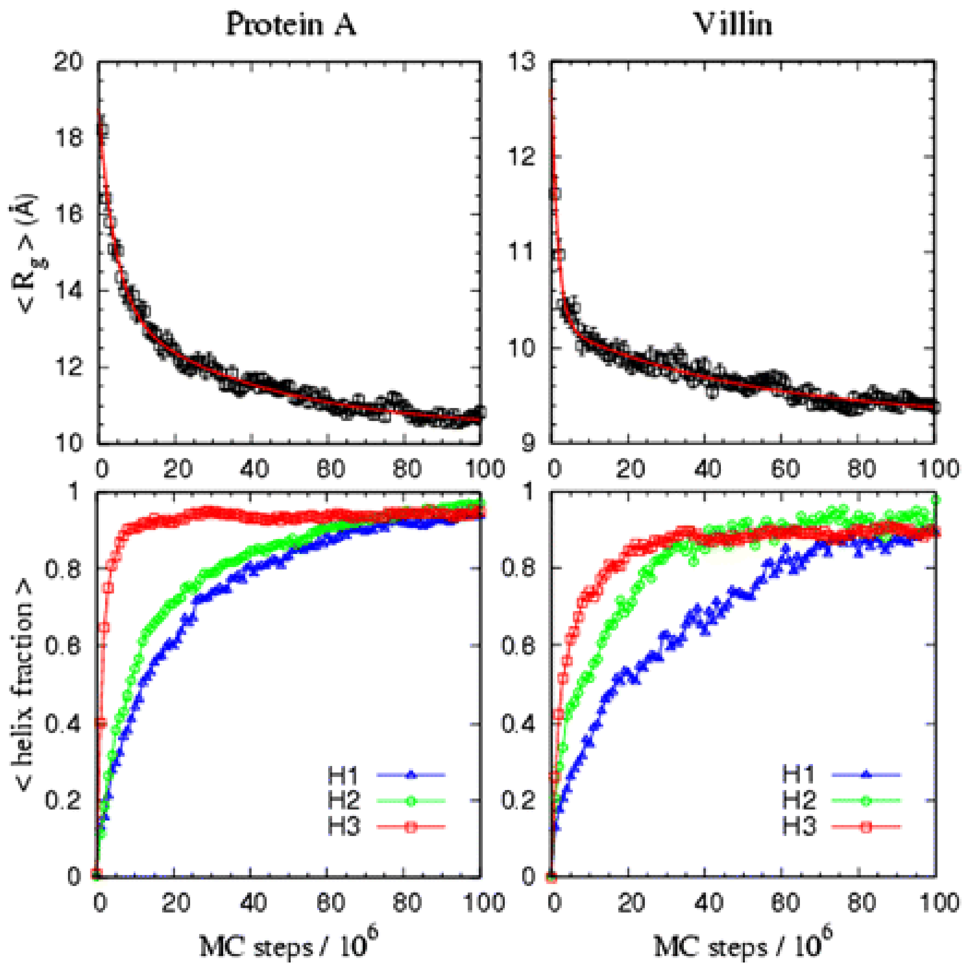
- To have a measure for the compactness of a structure, you can calculate the radius of gyration:

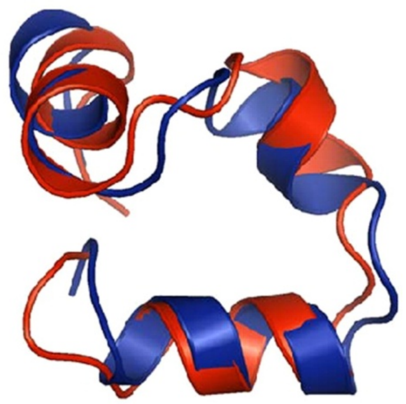
$$R_g = \left(\frac{\sum_i |\mathbf{r}_i|^2 m_i}{\sum_i m_i} \right)^2$$

where m_i is the mass of atom *i* and $\mathbf{r}_i$ the position of atom *i* with respect to the center of mass of the molecule.

Example: Radius of gyration







Example: R_g during protein folding.

Relaxation behavior of the average R_g (Upper) and average fraction helicity (Lower) of each individual helix, obtained at T ≈ 300 K Monte Carlo runs for protein A (left) and villin (right).

GROMACS facility: Radius of gyration

g_gyrate

Description:
g_gyrate computes the radius of gyration of a group of atoms and the radii of gyration about the x, y and z axes, as a function of time. The atoms are explicitly mass weighted.

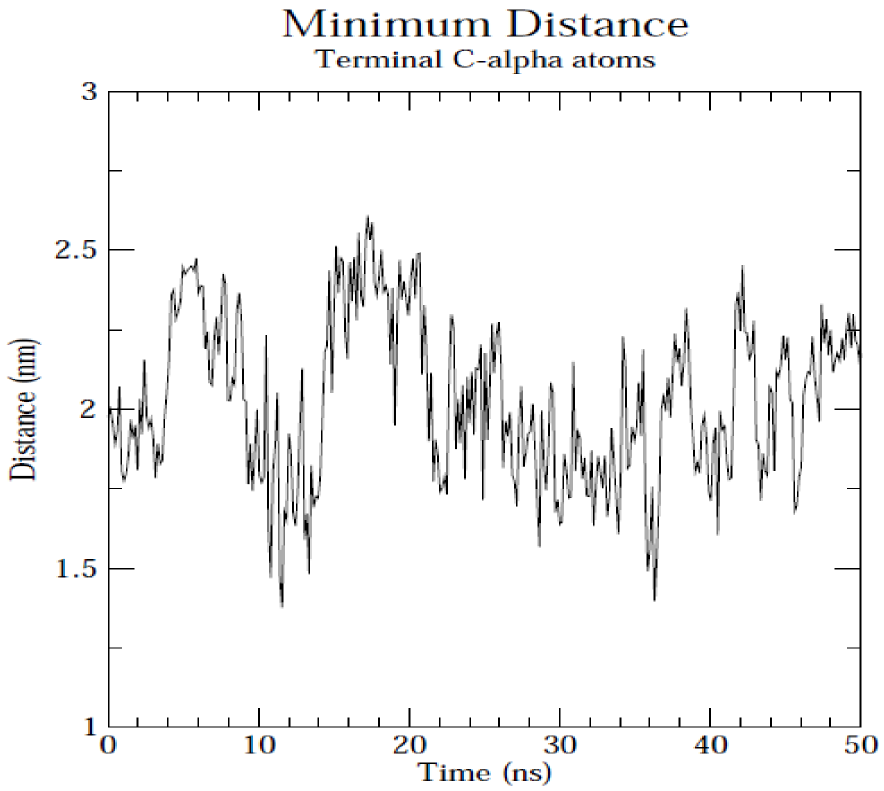
Invocation:

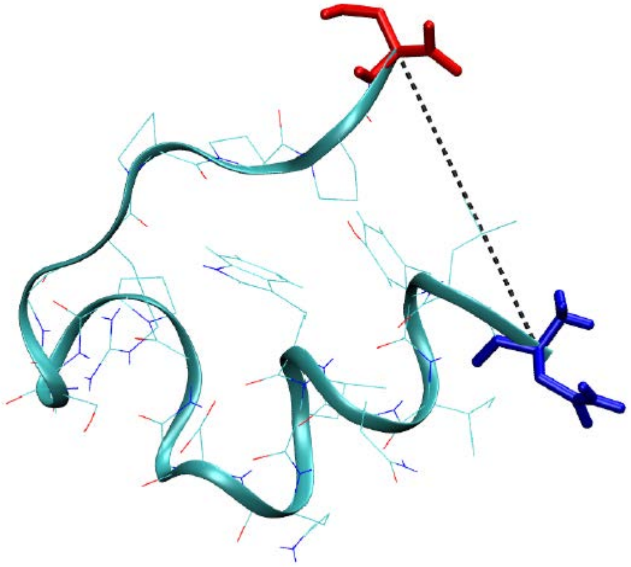
```
~$ g_gyrate -f input.xtc -s input.pdb -o gyrate.xvg
```

After invoking the command you are requested to select a group for which the calculation is to be performed.

Distances in the structure

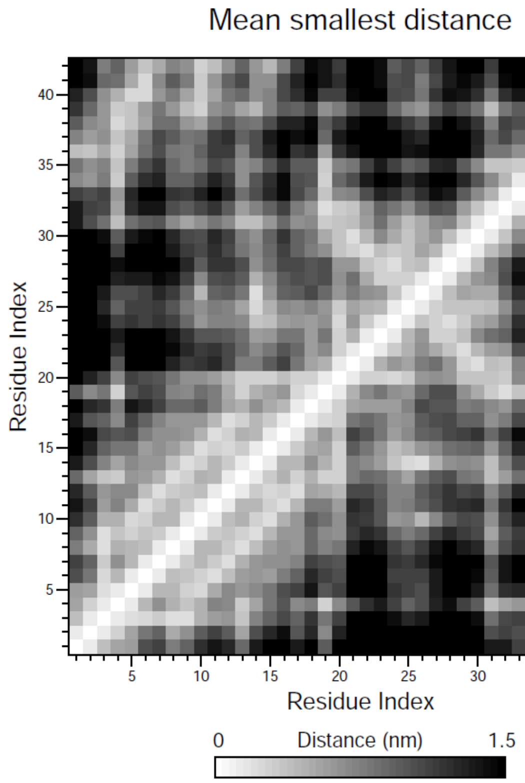
- To get information about contacts in the protein one can plot the **distances between two atoms** or the **minimum distance between two groups of atoms**, e.g., protein side-chains in a salt bridge.

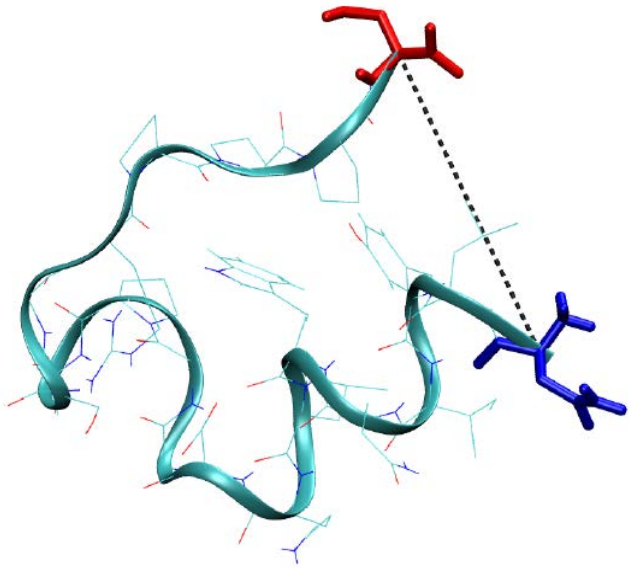




Distances in the structure

- If one plots the **distances between all residues** of the protein, one obtains a symmetrical **matrix**. Plotting these matrices for different time frames, one can analyze changes in the structure.





GROMACS facilities: Distances

g_sgangle calculates the distance between the geometrical centers of two groups.

```
~$ g_sgangle -f input.xtc -s input.tpr -oa output.xvg -n index.ndx
```

g_mindist calculates the minimum distance between two groups of atoms during time. It also calculates the number of contacts between these groups within a certain radius r_{max}.

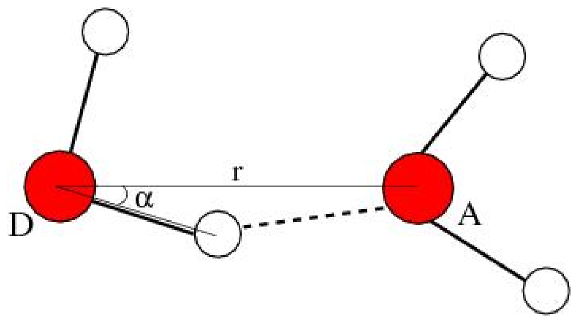
```
~$ echo X Y | g_mindist -f input.xtc -s input.pdb -od output.xvg
```

g_mdmat monitors the minimum distances between protein residues defined as the smallest distance between any pair of atoms of the two residues in question. The output is a symmetrical matrix which can be visualized with a program such as xv.

```
 ~$ g_mdmat -f input.xtc -s input.pdb -o output.xvg
```

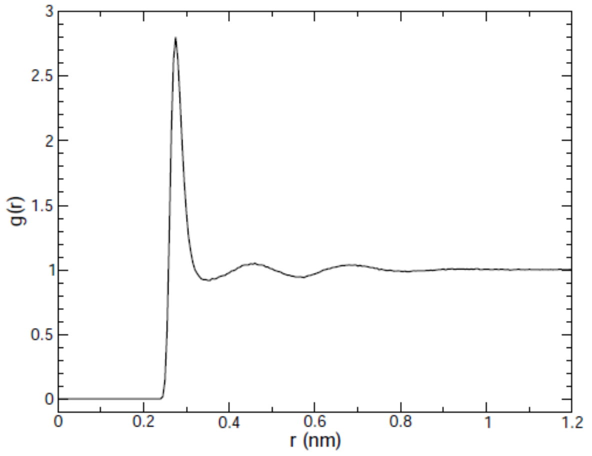
Hydrogen bonds

- To determine whether an H-bond exists between a donor D and an acceptor A, a geometrical criterion is used:



$$r \leq r_{\mathrm{HB}} = 0.35\,\mathrm{nm}$$

$$\alpha \leq \alpha_{\mathrm{HB}} = 30^\circ$$



- The lifetime of an H-bond can calculated from the autocorrelation function

$$C(\tau) = \langle s_i(t) s_i(t+\tau) \rangle$$

where $s_i(t)=\{0,1\}$ is the existence function of the H-bond *i* at time *t*. The integral of $C(\tau)$ gives an estimate of the average H-bond lifetime:

$$\tau_{\mathrm{HB}} = \int_0^\infty C(\tau) d\tau$$

GROMACS facilities: Hydrogen bonds

`g_hbond` analyses all hydrogen bonds existing between two groups of atoms (which must be either identical or non-overlapping) or in specified donor-hydrogen-acceptor triplets using the D−A distance and D−H−A angle criterions.

Invocation:

```
~$ g_hbond -f input.xtc -s input.pdb -num hbnum.xvg
```

You will be asked for choose two groups one representing acceptor and the other the H-bond donor group.

GROMACS facilities: Hydrogen bonds

Output:

- The total number of H-bonds in each time frame.

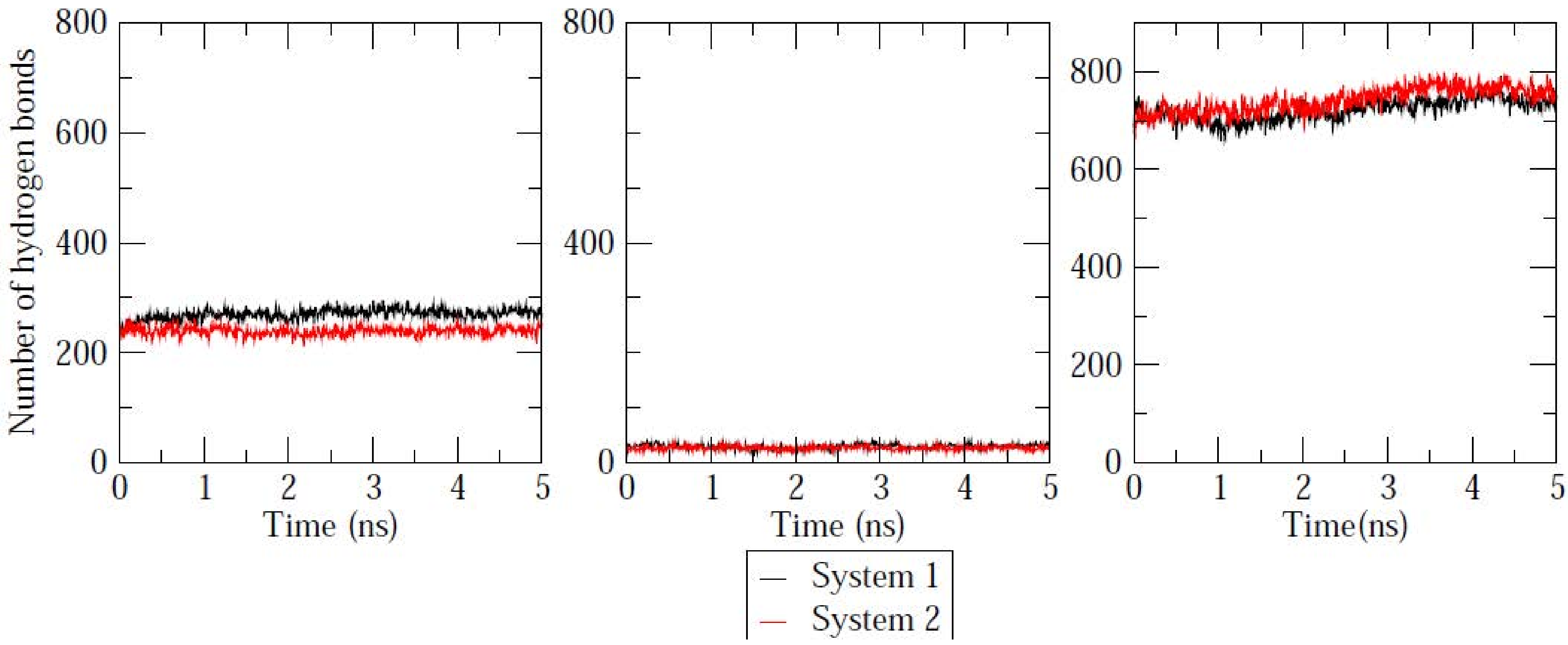
- The number of H-bonds in time between residues, divided into groups *n*-*n*+*i* where *n* and *n*+*i* stand for residue numbers and *i*=0 to 6. The group for i=6 also includes all H-bonds for i>6. This analysis provides a measure for the formation of α-helices or β-turns or strands.

- The autocorrelation function and the lifetime integral, both averaged over all hydrogen bonds, will be output allowing to analyse the H-bond kinetics.

- Index groups are output containing the analyzed groups: all donor-hydrogen atom pairs and acceptor atoms in these groups, donor-hydrogen-acceptor triplets involved in hydrogen bonds between the analyzed groups, and all solvent atoms involved in insertion.

Example: Hydrogen bonds

MainChain - MainChain MainChain - SideChain Protein - Water



Secondary structure analysis

- The DSSP (Dictionary of Secondary Structure for Proteins) program allows the analysis of secondary structure.
- DSSP uses a pattern-recognition process of hydrogen-bonded and geometrical features:

− elementary **H-bonding patterns**: ***turn*** and ***bridge***

− repeating turns = ***helices***, repeating bridges = ***ladders***

− connected ladders = ***sheets***

− **geometric structure** defined in terms of **torsion** and **curvature** of differential geometry

− torsion → ***handedness*** of helices and twisted β-sheets

− curvature → curved pieces = ***bends***

− **solvent exposure** = number of possible H_2O molecules in contact with a residue

GROMACS facilities:DSSP

`do_dssp` invokes DSSP which is interfaced to GROMACS.

The DSSP output assigns each residue a letter according to its secondary structure:

H = alpha helix
B = residue in isolated beta-bridge
E = extended strand, participates in beta ladder
G = 3-helix (3/10 helix)
I = 5 helix (pi helix)
T = hydrogen bonded turn
S = bend
A blank stands for loop or irregular, which is often called (random) coil.

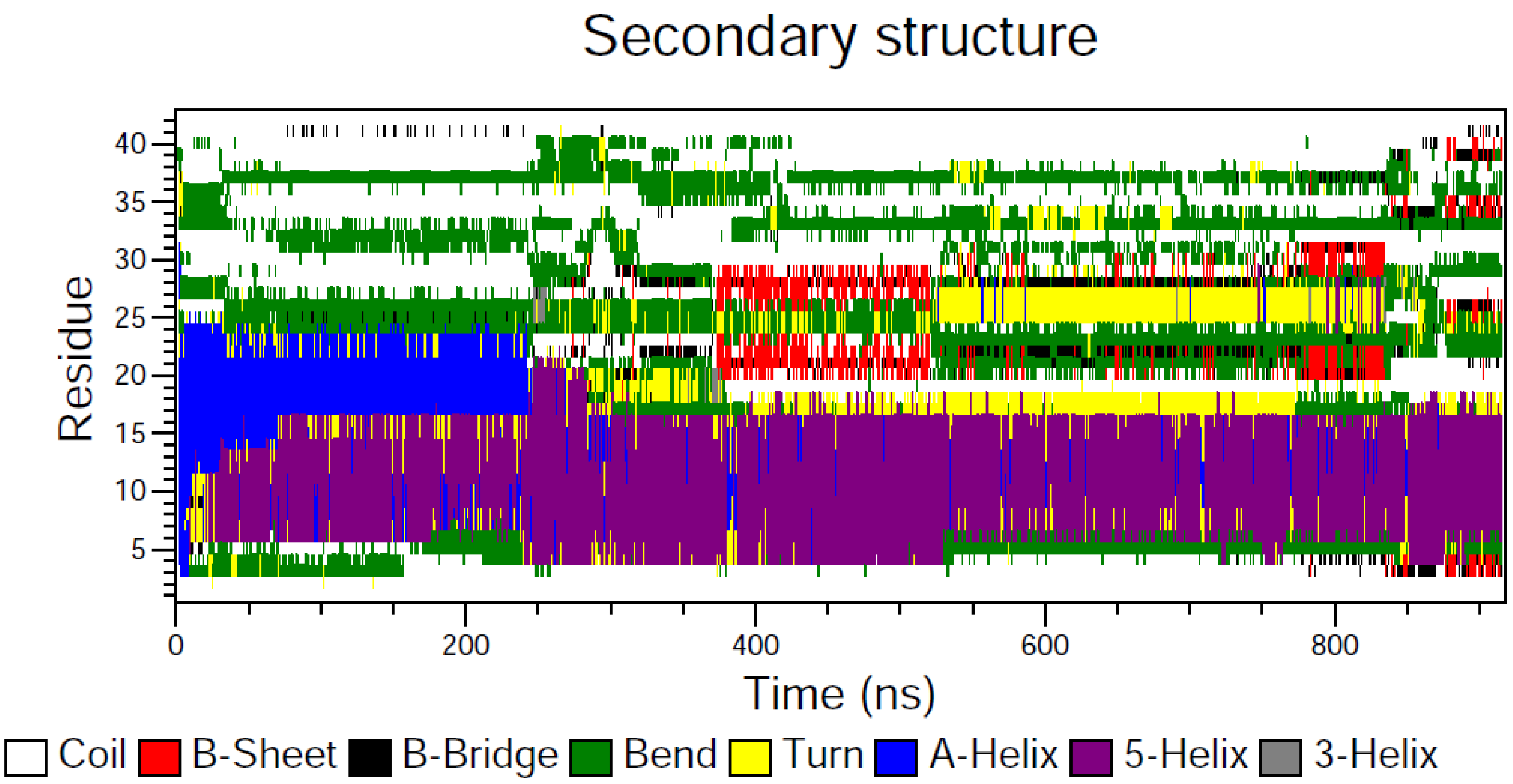
do_dssp produces an output plot which can be viewed with ***xmgrace***.

GROMACS facilities: DSSP

Invocation:

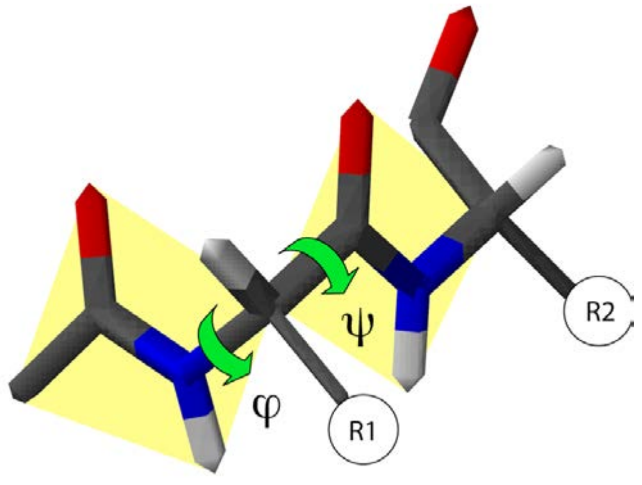
```
~$ do_dssp -f input.xtc -s input.pdb -o output.xpm
```

The only group acceptable for the calculation is the Mainchain (group 5) in the default index. Other outputs like the **SASA plot** can also be generated. Secondary structural transitions with time can be monitored in the plot.

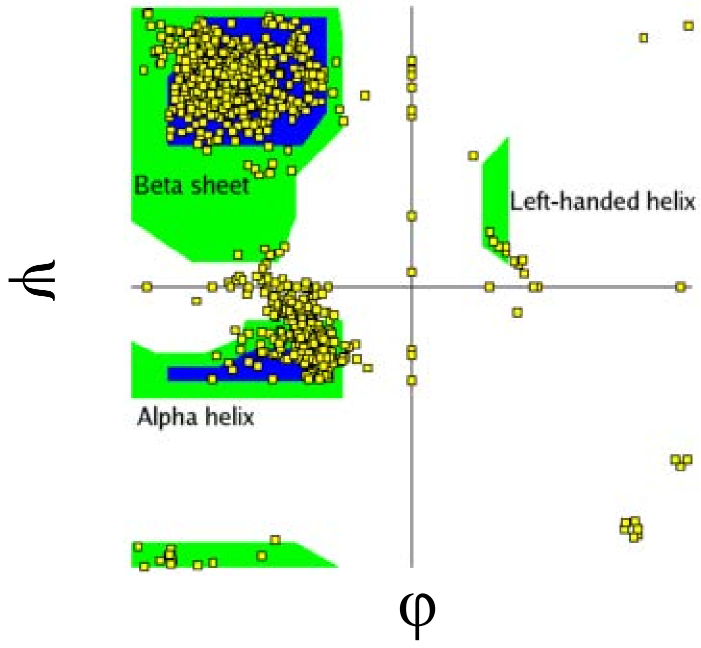


Secondary structure: Ramachandran plot

- A Ramachandran plot is the projection of the structure between two residues on the two dihedral angles φ and ψ of the protein backbone:



φ : C–N–CA–C
ψ : N–CA–C–N



- `g_rama` generates a Ramachandran plot in GROMACS.
- `xrama` generates the animation of the Ramachandran plot in time.

Free energy surfaces

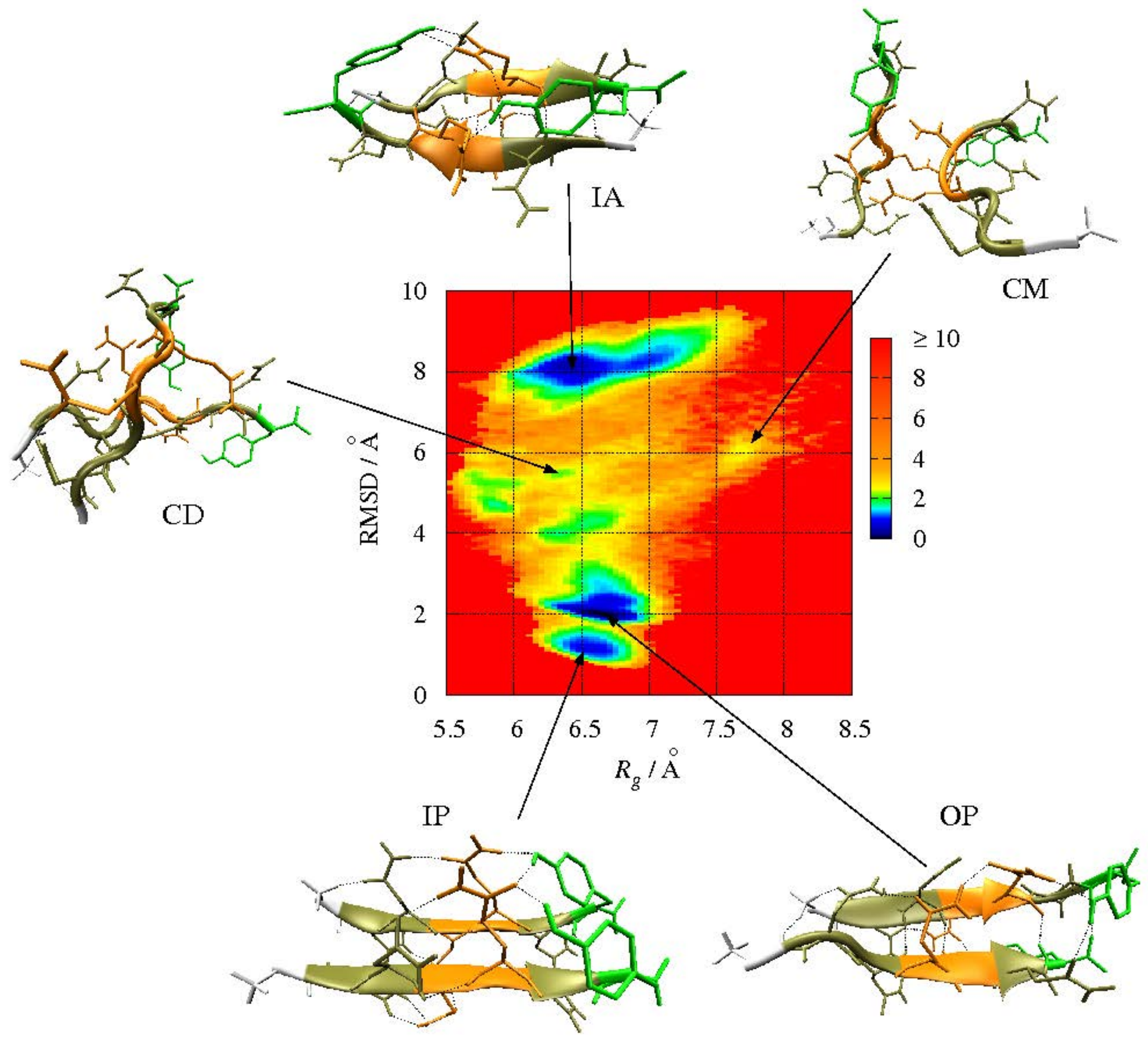
- Biomolecular processes, such as folding or aggregation, can be described in terms of the molecule's **free energy**:

$$\Delta G(R) = -k_\mathrm{B}T\left[\ln P(R) - \ln P_\mathrm{max}\right]$$

  where k_B is the Boltzmann constant, P is the probability distribution of the molecular system along some coordinate R, and P_{max} denotes its maximum, which is substracted to ensure ΔG = 0 for the lowest free energy minimum.

- Popular choices for *R* (so-called **order parameters**): R_g, rmsd, number of hydrogen bonds or native contacts, …
- Typically the free energy is plotted along two such order parameters, giving rise to a (reduced) **free energy surface (FES).**

Free energy surfaces



Example:
FES for the aggregation of the GNNQQNY peptide forming a dimer in terms of Rg and the RMSD from a perfect parallel β-sheet

Generating FESs with generateFES.py

```
#!/usr/bin/python

import sys,string
import numpy as np
import math


try:
        infilename = sys.argv[1]; outfilename = sys.argv[9]
        _minv1 = sys.argv[2]; _maxv1 = sys.argv[3]
        _minv2 = sys.argv[4]; _maxv2 = sys.argv[5]
        _i1 = sys.argv[6]; _i2 = sys.argv[7]
        _temp = sys.argv[8]
except:
     print "Usage:",sys.argv[0], "infile minv1 maxv1 minv2 maxv2 devisions1 devisions2 temperature outfile"; sys.exit(1)

##### Variable Initializations ##########

ifile = open(infilename,'r')     # open file for reading
ofile = open(outfilename,'w')    # open file for writing

i1 = int(_i1)
i2 = int(_i2)

minv1 = float(_minv1)
maxv1 = float(_maxv1)
minv2 = float(_minv2)
maxv2 = float(_maxv2)

V = np.zeros((i1,i2))
DG = np.zeros((i1,i2))


I1 = maxv1 - minv1
I2 = maxv2 - minv2

kB = 3.2976268E-24
An = 6.02214179E23
T = float(_temp)

##########################################
```

Generating FESs with generateFES.py

```
for line in ifile:
     v1 = float(line.split()[0])
     for x in range(i1):
                if v1 <= minv1+(x+1)*I1/i1 and v1 > minv1+x*I1/i1:
                        v2 = float(line.split()[1])
                        for y in range(i2):
                                if v2 <= minv2+(y+1)*I2/i2 and v2 > minv2+y*I2/i2:
                                        V[x][y] = V[x][y] +1
                                        break
                        break

##### Finding the maximum
P = list()
for x in range(i1):
        for y in range(i2):
                P.append(V[x][y])

Pmax = max(P)
#####

LnPmax = math.log(Pmax)

for x in range(i1):
  for y in range(i2):
        if V[x][y] == 0:
                DG[x][y] = 10
                continue
        else:
                DG[x][y] = -0.001*An*kB*T*(math.log(V[x][y])-LnPmax)

for x in range(i1):
        for y in range(i2):
                ofile.write(str((2*minv1+(2*x+1)*I1/i1)/2) + "\t" + str((2*minv2+(2*y+1)*I2/i2)/2) + "\t" + str(DG[x][y])
+"\n")

        ofile.write("\n")

ofile.close()
ifile.close()
```

Generating FESs with `generateFES.py`

Invocation:

```
~$ python generateFES.py infile minv1 maxv1 minv2 maxv2
devisions1 devisions2 temperature outfile
```

infile: rows represent the time-ordered conformations, the two columns correspond to the values of two order parameters, R_1 and R_2, for each of the conformations, e.g.

# R_g	rmsd
12.1	13.2
11.5	13.2

minv1, maxv1, minv2, maxv2: minimum and maximum values of the two order parameters for which the FES shall be produced

devisions1, devisions2: number of bins for the two order parameters

temperature: temperature for which the FES shall be produced

outfile: output file for the FES containing three columns with $R1$, $R2$ and ΔG.

The FES can be plotted using gnuplot.

Plotting the FES using gnuplot

Invocation:

```
~$ gnuplot < plotFES.gnu
```

- The gnuplot input file plotFES.gnu is:

```
set term postscript eps color enhanced
set output "FES.eps"
set pm3d map
set view map
splot 'outfile' t ''
set out
```

- The resulting graphics can be viewed with gv:

```
~$ gv FES.eps
```

Principal component analysis (PCA)

- Principal component analysis is also called **covariance analysis** or **essential dynamics**.
- It uses the covariance matrix σ of the atomic coordinates:

$$\sigma_{ij} = \langle (q_i - \langle q_i \rangle)(q_j - \langle q_j \rangle) \rangle$$

where $q_1,\ldots,q_{3N}$ are the mass-weighted Cartesian coordinates and ‹...› denotes the average over all sampled conformations.

- By diagonalizing σ, we obtain 3N eigenvectors $\mathbf{v}^{(i)}$ and eigenvalues λ_i with

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{3N}$$

Principal component analysis

- The eigenvectors and eigenvalues of σ yield the modes of collective motion and their amplitudes.
- The principal components (PCs)

$$V_i = \mathbf{v}^{(i)} \cdot \mathbf{q} = v_1^{(i)} q_1 + v_2^{(i)} q_2 + v_3^{(i)} q_3 + \dots + v_{3N-2}^{(i)} q_{3N-2} + v_{3N-1}^{(i)} q_{3N-1} + v_{3N}^{(i)} q_{3N}$$

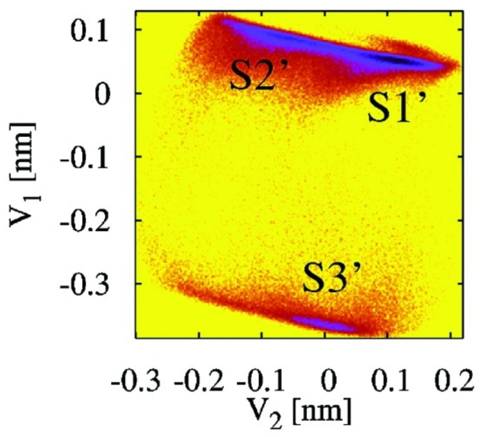
can then be used, for example, to represent the free energy surface.

- From the above equation we see that, e.g., the first three components $v_1^{(i)}$, $v_2^{(i)}$ and $v_3^{(i)}$ of the eigenvector $\mathbf{v}^{(i)}$ reflect the influence of the *x*, *y* and *z* coordinates of the first atom on the ith PC.
- A suitable measure of this influence is:

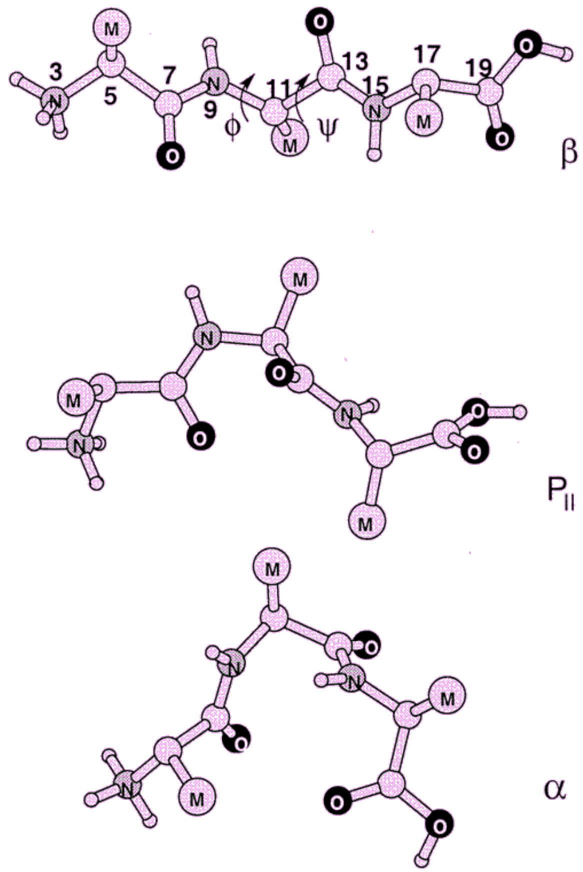
$$\Delta_1^{(i)} = (v_1^{(i)})^2 + (v_2^{(i)})^2 + (v_3^{(i)})^2$$

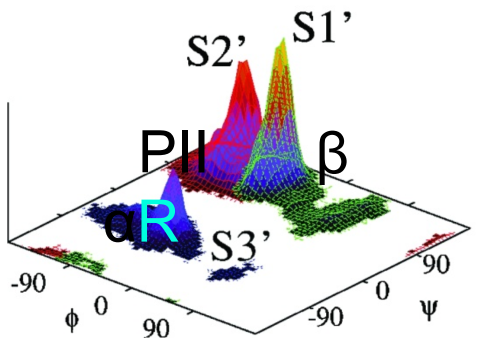
Principal component analysis

Example:



Free energy surface of Ala_3 in water as obtained from a 100 ns MD simulation using PCA.





The (φ,ψ) distribution pertaining to the labeled energy minima.

GROMACS facilities: PCA

`g_covar` builds and diagonalizes the covariance matrix.

Invocation:

```
~$ echo 1 1 | g_covar -f in.xtc -s in.tpr -o eigcv.trr
```

`g_anaeig` analyzes and plots the principal components and their overlap with the coordinates.

Invocation:

```
~$ g_anaeig -v eigenvec.trr -f nj.xtc -eig eigenval.xvg
-noxvgr -s average.pdb -first 1 -last 2 -2d 2dproj_1_2.xvg
```

Geometric clustering

- Geometric clustering is performed to identify similar structures sampled during the MD simulation.
- Various clustering methods exist.
- One of the most often used **clustering algorithms** applied to MD trajectories is the one developed by X. **Daura** (Angew. Chem. Int. Ed.1999, 38, pp 236-240).:

− It is based on the mutual RMSD between all conformations sampled during the MD simulation.

− A RMSD cut-off is chosen to determine cluster membership.

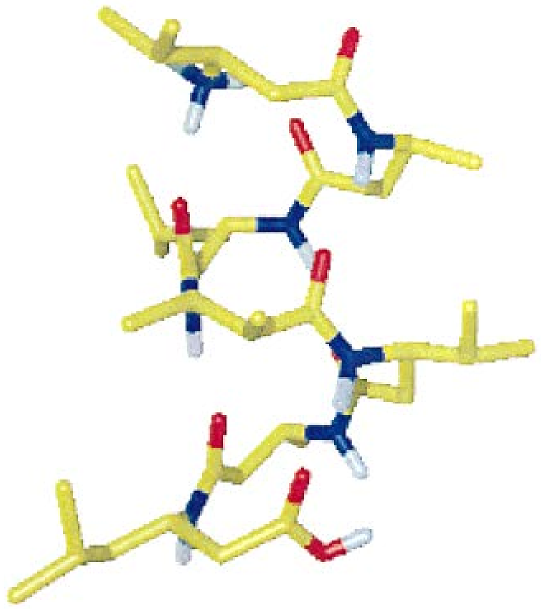
− Count number of neighbors using cut-off.

− Take structure with largest number of neighbors with all its neighbors as cluster and eleminate it from the pool of clusters.

− Repeat for remaining structures in pool until all structures are assigned to a cluster.

Geometric clustering

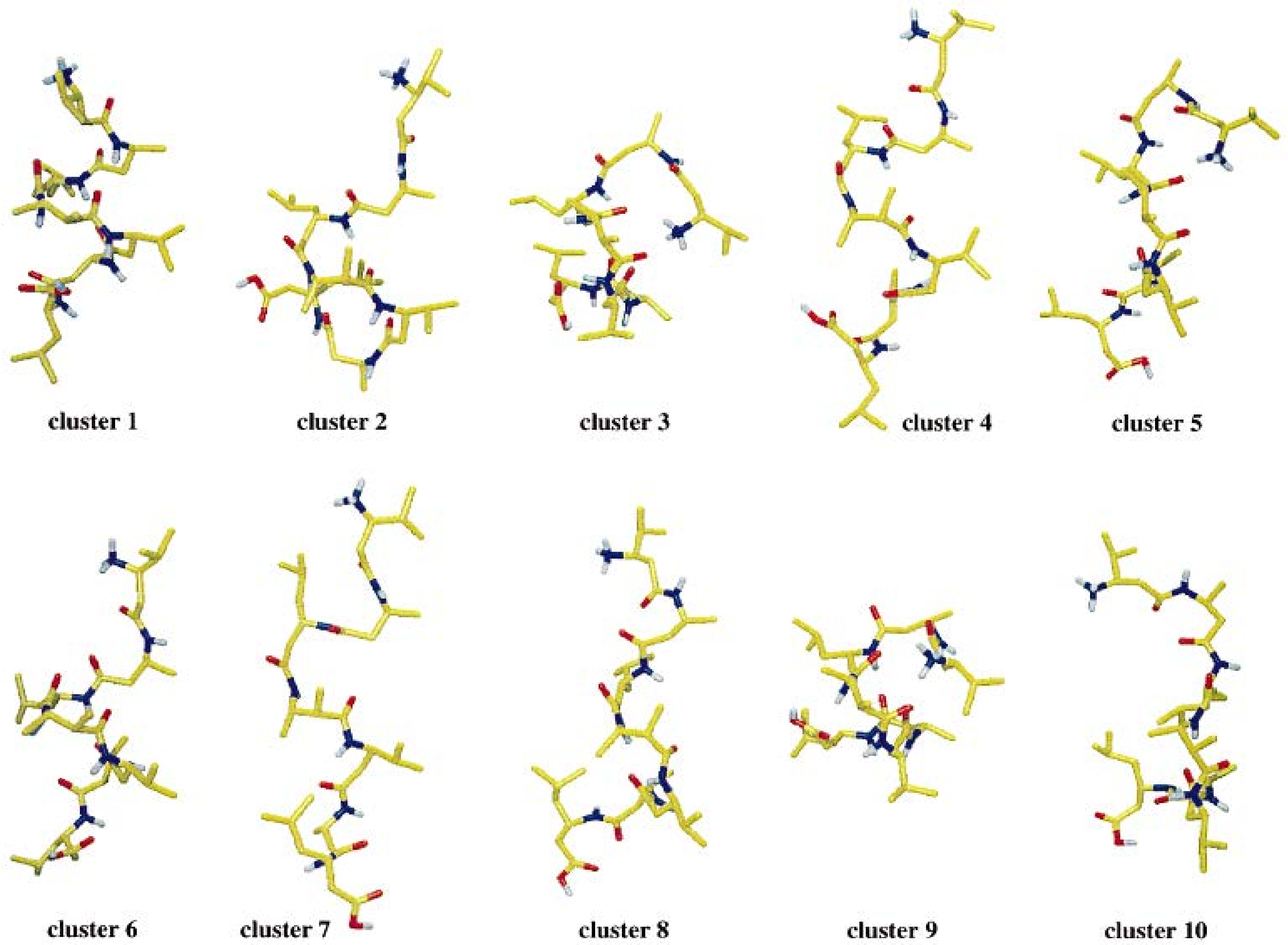
- All clusters are mutually exclusive, so a structure can only be a member of a single cluster.
- The structure with the largest number of neighbors in each of the clusters is taken as the cluster representative, i.e., the cluster centre.
- **Example:** Conformations sampled during a 50-ns MD simulation of a β-heptapeptide in methanol at 340 K; clustering performed with a 1 Å cut-off



The native state:
a left-handed 3_1-hlix at room temperature

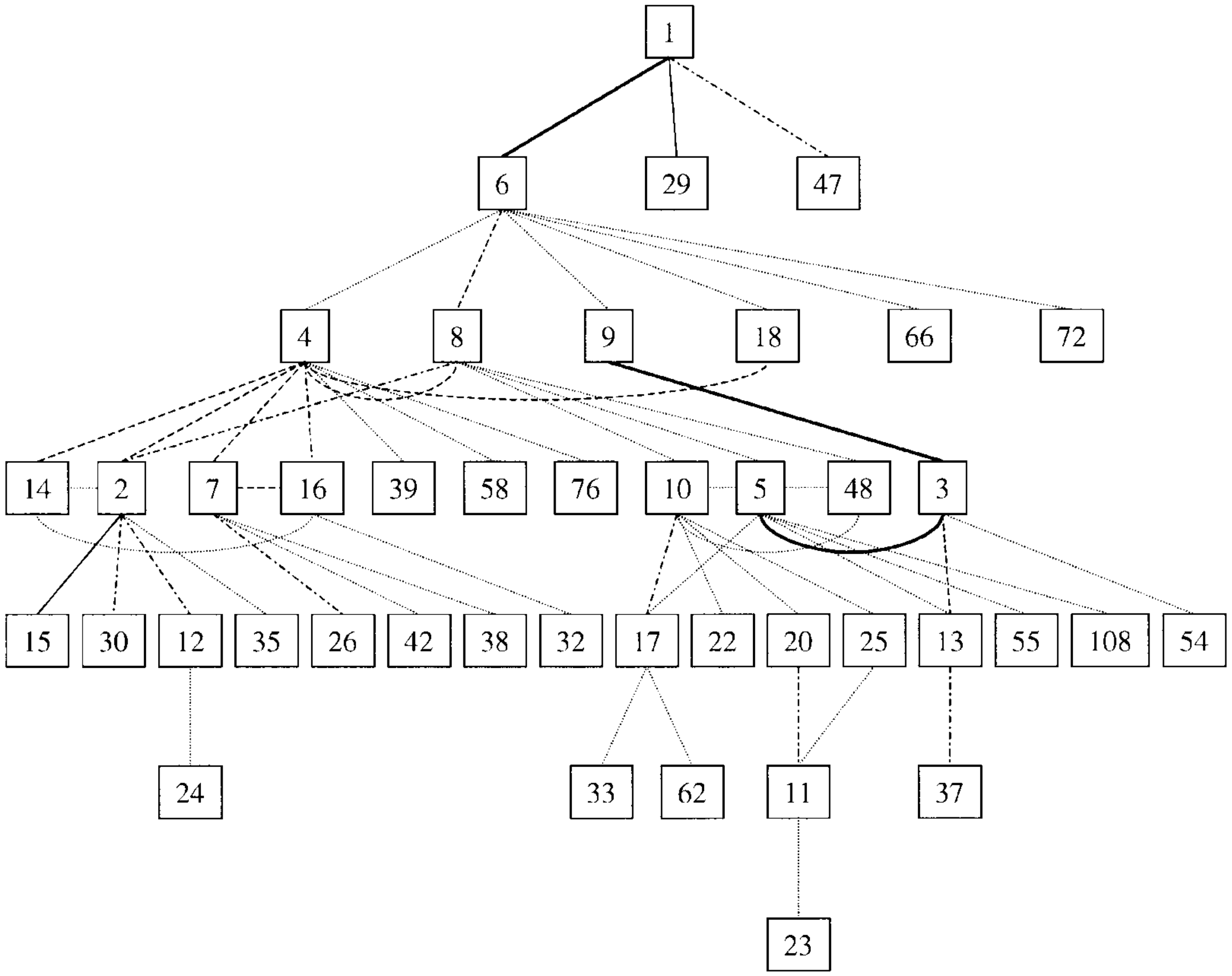
Geometric clustering

- **Example:** The 10 most populated clusters at 340 K



Geometric clustering

- **Example:** Pathways between clusters of structures at 340 K



GROMACS facilities: Clustering

`g_cluster` can cluster structures with several different methods.

Distances between structures can be determined from a trajectory or read from an XPM matrix file with the -dm option. RMS deviation after fitting or RMS deviationof atom-pair distances can be used to define the distance between structures.

The method option **gromos** uses the algorithm as described by Daura et al.

Invocation:

```
~$ echo 1 1 | g_cluster -f input.xtc -s input.tpr -method gromos
     – cl out.pdb -g out.log
```

The midpoint of each cluster is written with the -cl option while the clusters are written in the log file. Use **`g_cluster -h`** to see all other available options.