GNM Tutorial.

Introduction

To study the normal modes of a protein, a library called Pynoramix developed by this group will be used. This library has been written in python language. Along this document you will find the set of commands needed to make the analysis.

Although this set of commands can be written in a text file (script) to be executed at once, we will use an interactive python interpreter to learn how to use Pynoramix.

First of all open the interpreter in a terminal:

```
ipython
```

And load the pynoramix modules:

```
from pyn_cl_set import *
from pyn_cl_unit import *
from pyn_cl_gnm import *
from pyn_cl_anm import *
```

Loading a pdb

Now we are ready to use the objects and functions defined in pynoramix. Lets load a protein from its pdb. If the pdb (FOO.pdb along this document) is already in your directory:

```
syst_foo=cl_set('FOO.pdb')
```

If the pdb must be downloaded from the Protein Data Bank:

```
syst_foo=cl_set(download='FOO.pdb')
```

The string syst_foo can be replaced by whatever you prefer.

Now Pynoramix knows the topology of the protein: number and name of residues, atoms, ... everything but the coordinates.

The list of variables stored about the protein can be checked with:

```
dir(syst_foo)
```

For instance, try to check things as:

```
syst_foo.num_atoms
syst_foo.num_residues
```

The list of data related with the atoms can be checked with:

```
dir(syst_foo.atom[0])
```

This way, the properties of the atom N can be shown:

```
syst_foo.atom[N-1].name
syst_foo.atom[N-1].pdb_index
syst_foo.atom[N-1].resid_name
syst_foo.atom[N-1].resid_pdb_index
syst_foo.atom[N-1].bfactor
```

And the same can be done with the residues:

```
dir(syst_foo.residue[0])
```

Lets check some of the properties of the N^{th} residue:

```
syst_foo.residue[N-1].name
syst_foo.residue[N-1].pdb_index
syst_foo.residue[N-1].list_atoms
```

Lastly the coordinates of the atoms are implemented in the object syst_foo:

```
syst_foo.load_coors(syst_foo.file)
```

To obtain the coordinates of the N^{th} atom:

```
syst_foo.coors[0].xyz[N-1]
```

Gaussian Network Model

The Gaussian Network Model will be built with the coordinates of the alpha carbons. First of all a new system only with these atoms must be generated:

```
CAs=make_selection(syst_foo,'atom_name (CA)')
```

The string CAs can be replaced by whatever you prefer.

Check the number of atoms of this system, it must be equal to the number of residues in the protein described in FOO.pdb.

```
CAs.num_atoms
```

It is time to make the GNM analysis. With this purpose type:

```
gnm_CAs_foo=gnm_classic(CAs,10.0)
```

The string gnm_CAs_foo can be replaced by whatever you prefer.

Where 10.0 is the cutoff to build the contact map, and consequently the network. Change this parameter and check how this contact map changes with the command:

```
gnm_CAs_foo.plot_contact_map()
```

You can also see how the fitness of the comparison between the bfactors computed with the mean square fluctuations (msf) of the CAs and the experimental bfactors is.

```
gnm_CAs_foo.plot_bfacts()
```

A possible way to choose the proper cutoff could be looking at the best fit between these two magnitudes. The dispersion of the points (msf, bfactors) can help to decide:

```
gnm_CAs_foo.plot_dispersion_bfacts()
```

To run a systematic analysis of the mean square deviation (R^2) of this fit when the cutoff is changed type the next command:

```
gnm_CAs_foo.plot_best_cutoff()
```

From now on CO will be the value of the cutoff you want to use. Therefore, again:

```
gnm_CAs_foo=gnm_classic(CAs,CO)
```

Lets see other magnitudes obtained by this analysis:

- The eigenvalues are stored in the array:

  ```
  gnm_CAs_foo.eigenvalues[]
  ```

  Check that the first eigenvalue is equal to zero.

  ```
  gnm_CAs_foo.eigenvalues[0]
  ```

  Remember, we have as many eigenvalues as alpha carbon atoms.

- The eigenvectors are stored in the matrix:

  ```
  gnm_CAs_foo.eigenvectors[][]
  ```

  To print the firts mode (eigenvector):

  ```
  gnm_CAs_foo.eigenvectors[0][:]
  ```

  And to print the j^{th} element of the i^{th} mode:

  ```
  gnm_CAs_foo.eigenvectors[i-1][j-1]
  ```

- To get the frequencies:

  ```
  gnm_CAs_foo.freqs
  ```

  To make a plot with these values, you can use:

  ```
  pylab.xlabel('index of mode')
  pylab.ylabel('frequency')
  pylab.plot(gnm_CAs_foo.freqs,color='red')
  ```

Or

```
pylab.xlabel('index of CA')
pylab.ylabel('Contribution to the eigenvector')
pylab.plot(gnm_CAs_foo.freqs,color='red')
pylab.plot(gnm_CAs_foo.eigenvects[1][:],color='red',label='Title')
```

Check now with these tools what the meaning of the first eigenvector is.

To conclude this section, plot the correlation matrix with:

```
gnm_CAs_foo.plot_correl()
```

Or the normalized correlation matrix:

```
gnm_CAs_foo.plot_correl_norm()
```

Using PyMol to a better understanding of the modes.

Before using PyMol we need to write some files:

```
gnm_CAs_foo.write()
```

With this last command two new files have been created: contact_map.oup and gnm_vects.oup. You will find them in your folder.

Now open a new terminal, or close your current terminal with:

```
quit()
```

And type:

```
pymol FOO.pdb
```

You will find a new menu in the main pymol window called GNM. Choose the option GNM-Contact Map and open the file contact_map.oup. The network is represented linking the alpha carbons in contact (with a distance lower than CO). This network is defined in PyMol as a new object call ElasticNet, thereby we can delete it with the menu A found in the window PyMol Viewer.

Lets have a look to the modes (eigenvectors). First of all represent the secondary structure of the protein with the option cartoon.
Now, go to the menu GNM-Load Modes and select the file gnm_vects.oup. Pymol at this point has in memory the eigenvectors. We can already show them in color code (red<0<blue). With this aim, go to the menu GNM-Choose Mode and insert the index of the mode you want to plot (Remember: the index 0 belongs to the first mode).

Anisotropic Network Model

To continue with this section ipython must be opened with the protein loaded. The tools to study the anisotropic model are quite similar to those explained in the previous section.

The system containing only the alpha carbons must be created:

```
CAs=make_selection(syst_foo,'atom_name (CA)')
```

To make this analysis type this time:

```
anm_CAs_foo=anm_classic(CAs,12.0)
```

As it was made before, you can for instance check how the contact map is:

```
anm_CAs_foo.plot_contact_map()
```

To visualize the ANM modes with pymol

Check the webpage: *http://www.pymolwiki.org/index.php/Modevectors.*