A visualization tool for studying the eigenmodes of a coarse-grained model of DNA

Julien Delafontaine,
section de mathématiques

Projet de semestre 2, 1e année master,
encadré par John Maddocks, Jeremy Curuksu, Daiva Petkeviciute.

Lausanne, année académique 2009 - 2010

User's guide

1. About the program

This software has been created to compare eigenfunctions of a coarse-grained piece of DNA built with respect to the model described in the following article:

> F. Lankas, O. Gonzalez, L. M. Heffler, G. Stoll, M. Moakher, J. H. Maddocks, *"On the parameterization of rigid base and basepair models of DNA from molecular dynamics sim*ulations*"*, Phys. Chem. Chem. Phys. **11** (2009) 10565-10588.

The notation was kept as often as possible, is order the source code to be easily readable.

The expression for the potential energy of the molecule in our model involves a stiffness matrix, which eigenvectors may have interesting properties. This software is supposed to help finding it out.
Secondly, the assumption that a base is only influenced by its nearest neighbours is equivalent to set a big part of the coefficients of the stiffness matrix to zero. One could wonder whether the so truncated matrix roughly has the same properties as the full one. Moreover, one finds that such truncated matrices can have negative eigenvalues, which are not consistent with the fact that the stiffnesses are the inverse of covariance matrices (positive-definite). The software was created to check the properties of such eigenmodes.

2. Basic use

Starting the program opens the main menu window.

In the field *Eigenvalue 1*, enter the index i of the i-th eigenvalue you are interested in - eigenvalues are sorted from smallest to largest - "5" means "5th smallest eigenvalue".
If set to 0, the molecule won't move and the equilibrium state will be drawn. A non-zero value will start an animation of the molecule moving in the direction of the corresponding eigenvector.
Entering a non-zero value in the field *Eigenvalue 2* will create a new instance of the molecule that will move in the direction of a second eigenvector chosen in the same way as previously.
If set to 0, nothing will happen.

For each of them one can choose either to use the stiffness matrix or a truncated version of it.
The *precision* parameter is the number of steps in the animation. It won't change the final result. Increasing it will make the animation more fluent but slower, and conversely.
Setting it to 1 will produce only the ending state, without animation.
The lambda parameter regulates the strength of the added effect (the scalar multiple of the added eigenvector).
One can choose whether to show the center line and/or the bases by checking the corresponding boxes.

Start the animation by pushing the "Draw" button.

3. How to compile the code

The program is written in Python.
http://www.python.org/

It is made of 5 files: *main.py, bases.py, draw.py, anim.py, menu.py*, and a folder containing the configuration data. The main file is main.*py*. Compile and run this last one, having the 4 other ones in the same folder. The executing function is called `run`.
For example using IPython[1], enter the folder containing the files and type

```
>>> import main
>>> main.run()
```

The numerical part of the program only uses standard libraries (numpy, scipy).

The 3D interface is built on a library called *mlab* contained in the *Mayavi 2* package. It is called using `from enthought.mayavi import mlab`. To install Mayavi 2, go to that page:
http://code.enthought.com/projects/mayavi/docs/development/html/mayavi/installation.html
All the (few) documentation is there:
http://code.enthought.com/projects/mayavi/docs/development/html/mayavi/index.html
Note that it is a part of a bigger package for scientific data analysis:
http://code.enthought.com/
For Windows users, it is much easier to install the Python(x,y) package, including everything:
http://www.pythonxy.com/

4. How to change the configuration parameters (new data)

All the input data is contained in the folder *conformation data*. Configuration parameters come from the Matlab file *parms_basejhb_XXXX.mat*. The `etav` object is intra rotations, `wav` is intra translations, `uav` is inter rotations, and `vav` is inter translations. They are contained in that order in the `w` variable. `s1b` is the stiffness matrix.
Once one has replaced or modified those files, one must also edit *sequence.txt*, which contains the sequence of bases used, e.g. GCTATATATATATATAGC (by default).

1 http://ipython.scipy.org/moin/Download

5. Functions reference

main.py

- `readParameters()`:
Reads the configuration parameters from a text or Matlab file.
Returns a vector w=[intra rotation i; intra translation i; inter rotation i, inter translation i] for i=1..number of bases, and the sequence of bases.

- `calculPoints(w)`:
Computes spatial coordinates in the lab frame from a configuration vector w as previously defined.
Returns `r`=base central points, `q`=intra base points, `d`=base frames, `g`=intra base frames, `h`=inter base frames, and the `sequence` of bases.

- `plot3d(cf, vp1, vp2, precision, m1,m2,centerline,showbases)`:
Draws the molecule of configuration `cf`=r,q,d,g,h,sequence as previously defined. Provides animation of the molecule in the direction of an eigenvector `vp1`/`vp2` of a (stiffness) matrix `m1`/`m2`.
Options: precision=number of steps in the animation; centerline=0 or 1 to hide the center line or not; showbases=0 or 1 to show the bases or not.
Uses the library *mlab* from the *Mayavi 2* package.

Possible modifications:

- Remove the '`#`' before `mlab.options.backend='envisage'` to use the full Mayavi 2 interface.
- Change the parameter `size` of the object `fig` to chenge the dimensions of the window.
- Change the `colormap` parameter of the different objects to change their colors. In particular, one can give different colors to `main_strand` and `comp_strand` to differenciate the two opposite strands. For different possible values, see http://code.enthought.com/projects/mayavi/docs/development/html/mayavi/mlab.html#adding-color-or-size-variations

anim.py

- `stiffness(m,vp_id)`:
Returns the eigenvector corrsponding to the eigenvalue vp_id of a matrix m.

- `ACP(w,vecp,mult)`:
Returns a configuration `mod` modified from `w` by adding a multiple `mult` of an eigenvector `vecp`: `wmod=w+mult*vecp`.

draw.py

- `data3d(r,d,sequence)`:
Builds vtk sources used to draw the bases and base frames.
Returns an array `points` containing the coordinates of each corner of the polygons, an array `triangles` containing the connectivity between these points, an array `centers` containing the base reference points, and an array `frames` containing the base frames moved to the corresponding base reference point.

bases.py

- `cayley(v)`:
Returns the Cayley transform (matrix) of a vector `v`.

- `rectangle(r,dir1,dir2)`:
Was only used for testing purpose. Creates rectangles to coarse-model the bases, instead of more realistic polygons.

- `adenine(r,d)`:
Returns lab frame coordinates of the corners of the polygon representing the purine (the big ones; the shapes of adenine and guanine are the same here) situated around the point `r` and oriented with respect to the base frame `d`.

- `cytosine(r,d)`:
Same for pyrimidines (the hexagonal ones).

menu.py

- `menu(action, config)`:
Creates a new frame containing the main interface.
Applies the action function to the config argument (has to be done this way because of a weakness of the *Tkinter* library).

Possible modifications:

- One can change the default values appearing in the fields by changing the `value` argument (in parenthesis) in every line containing a `variable.set(value)`.

6. Troubleshooting

Contact julien.delafontaine@epfl.ch.
Learn Python.