EVERYTHING NEEDED TO RUN ROTATIONAL DIFFUSION SIMULATIONS

January 4, 2015

1 Rotational Diffusion

Code for running simulations of temporally heterogeneous rotational diffusion is contained in the file `rotation_.c`, which can be compiled by running

```
gcc rotation_.c -lm -o rotate.
```

The resulting executable file (here named `rotate`), may then be run from the command line, and will output a number of trajectories named `rot.`n`.dat` (n is the index of the run) listing the x, y, z positions of a point on the surface of a unit sphere as a function of time.

Simulation parameters are set by calling the program with a number of flags, decsribed below. For example, running

```
rotate -r 50 -t 1000000 -tf 2.0 -xm 3.0
```

will create 50 runs, each 10^6 steps long, with a FWHM of the $\log \tau_r$ distribution of 2.0 and a median $\log \tau_x$ of 3.0. All τ values are in terms of the rank-2 rotational correlation functions, i.e. setting $\log \tau_r$ to 2.0 will guarantee that the exponential rank-2 RCF decays to $1/e$ after 100 steps.

flag	default	description
-r	1	number of runs
-s	0	index of first run
-h	0	prints header at top of output file
-p	0	outputs separate file of tau values
-t	10000	trajectory length
-tm	2.0	median $\log \tau_r$
-tf	1.0	fwhm $\log \tau_r$
-tc	0.0	τ_r correlation (ρ_r)
-xm	2.0	median $\log \tau_x$
-xf	0.0	fwhm $\log \tau_x$
-xc	0.0	τ_x correlation (ρ_x)

2 Linear Dichroism

The resulting trajectories may be turned into a (noisy) linear dichroism signal. The file linear_dichroism_.c can be compiled with

```
gcc linear_dichroism_.c -lm -o dichroism .
```

The resulting executable file (here named `dichroism`), may be run from the command line. It needs one argument, the name of the simulated trajectory. Running

```
dichroism rot.0.dat
```

will create the file `rot.0.dat.LD`, containing the computed dichroism signal. The details of this procedure are given in comments in the source file, and parameters such as noise intensity, etc. may be changed by editing the appropriately named variables in the source code.

3 Correlation Functions

The files `frame_.c`, `frame_.h`, `frame_corr_.h`, and `frame_rcor_.h` contain code for calulating auto-correlation functions from either of the above trajectory formats.

As given, the code will compute the ACF for a standard x, y, z trajectory file. To instead calculate the ACF of a single variable (as output from the Linear Dichroism program above), one can edit frame_.h changing the value of `FORMAT` from `XYZ` to `DAT`, and the value of `ANALYSIS` from `RCOR` to `CORR`.

In the latter case, since the program only runs through the input file once, it computes the actual ACF of the variable as given, wheras one generally wants to compute the ACF of the variable less its average. In this case, one needs to compute the average separately, and feed it to the program with the `-off` flag (see below).

The program is compiled with

```
c99 frame_.c -lm -w -o frame ,
```

making sure all `***.h` files are also in the present directory. The resulting executable file (here named `frame`), may then be run from the command line, with a number of flags setting the analysis parameters (see table below).

At least one flag (`-f`) must be set, specifying the input file. Running

```
frame -f input.dat
```

will output `input.dat.rcor.avg`, or `input.dat.corr.avg`, depending on the format of the input file. In both cases, the output is formatted as tab-separated values. For the former, the columns corrsepond to time, rank-1 RCF, rank-2 RCF, LD ACF, and number of points used. For the latter the columns are time, value, and number of points used. The output file begins with a number of header lines, each beginning with a `#` character (`xmgrace` reads these as comments), specifying the parameters used for the analysis. This header can be turned off with the `-h` flag.

ACF's are calculated at base-2 exponential time points, up to time 2^{T_c}, then linearly in increments of 2^{T_c} out to time $N_c \cdot 2^{T_c}$. For example, with $T_c = 3$ and $N_c = 4$, the ACF is computed at times $t = 1, 2, 4, 8, 16, 24, 32$. This scheme is useful for equal sampling of a signal expected to decay exponentially, hence for rotational diffusion it should be ok to set $N_\mathrm{c} = 1$ and find an appropriate value of T_c. In practice, I usually set $N_\mathrm{c} \sim 150$, and then find an appropriate T_c, making sure the ACF decays to zero well before the final time. These parameters are set with the flags `-tc` and `-nc` (T_c must be less than 32).

The program will compute the ACF's using N initial frames, equally spaced throughout the trajectory, set with the flag `-n`. I generally used $N = 400\,000$ for rotation data. The program can also chop trajectories to a specified length, say 1000 frames, with the flag `-chop 1000`.

The various flags, which may come in any order, are summarized here:

flag	argument?	description
-f	yes	input filename
-tc	yes	T_c
-nc	yes	N_c
-n	yes	N
-chop	yes	length to chop at
-off	yes	offset for ACF calc
-h	no	do not print output file head
-q	no	do not print to screen

So, for example,

```
frame -f input.dat -tc 3 -nc 4 -n 500000 -chop 1000 -q
```

will compute the ACF from the file input.dat, with $T_\mathrm{c} = 3$, $N_\mathrm{c} = 4$, $N = 500\,000$, chopping the trajectory to 1000 frames, and not printing updates to the screen.

4 ACF Fitting

The file `rotation_analysis_.nb` is a Mathematica 10 notebook with example routines for importing and fitting the computed ACF's to stretched exponentials, and for fitting the resulting parameter distributions to histograms, using the built-in functions `NonlinearModelFit` and `Histogram`.

There are a few notes inside for the functions I wrote to do this, but wouldn't be much use trying to describe it all in detail. The notebook should serve as a good starting point, but one will likely need to read the documentation for these functions, as well as for importing data and plotting, to get a feel for how it all works if they're going to use this seriously... or choose a more familiar method for performing the fits.