Documentation for networkDiffusion Code

E. F. Koslover

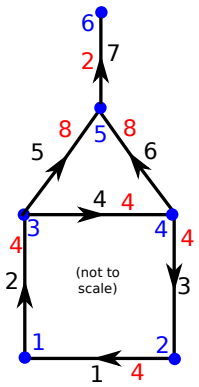
Last updated January 29, 2018

The code in this package will simulate diffusion on a network of 1D paths.

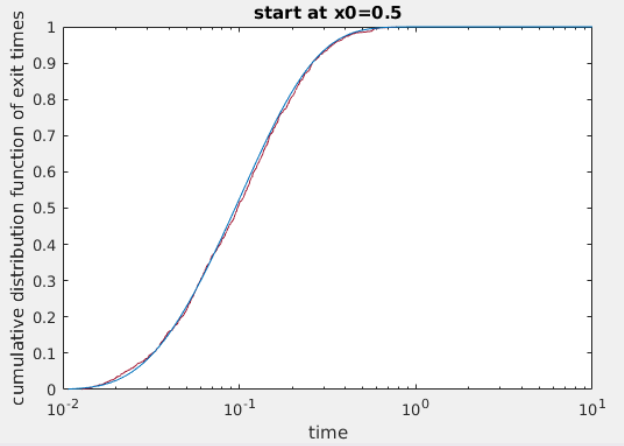
Notes for myself describing code implementation

- Assume all diffusion is on purely 1D paths, so just need to keep track of where in the network the particle goes

Example network to use in code development:
blue = node numbers
black = edge numbers
red = edge lengths



Testing Notes



Compare analytical cumulative distribution function for exit times from 2 absorbing boundary 1D interval (starting at $x_0 = 0.5$, using sine series expansion) versus numerical Brownian dynamics sims

Uses TESTSAMPLEFPT and TESTBROWNDYN subroutines.

1 Compilation Instructions

To compile and run the program, you will need the following:

- a compiler capable of handling Fortran90. The code has been tested with the gfortran and compiler. The default compiler is gfortran.
- BLAS and LAPACK libraries installed in a place where the compiler knows to look for them
- Optionally: Matlab to visualize output data

The code has been tested on Ubuntu Linux.

To compile with gfortran, go into the `source` directory. Type `make`. To compile with any other compiler that can handle Fortran90, type

```
make FC=compiler
```

substituting in the command you usually use to call the compiler.

If the compilation works properly, the executable `chainBD.exe` will appear in the main directory.

2 Usage Instructions

To run the program in the main directory, type:

```
./chainBD.exe suffix > outputfile.out
```

Here, `suffix` can be any string up to 100 characters in length. The program reads in all input information from a file named `param.suffix` where, again, `suffix` is the command-line argument. If no argument is supplied, it will look for a file named `param`. If the desired parameter file does not exist, the program will exit with an error. You can supply multiple suffixes to read in multiple parameter files.

The parameters in the input file are given in the format "*KEYWORD* value" where the possible keywords and values are described in Section 4. Each keyword goes on a separate line. Any line that starts with "#" is treated as a comment and ignored. Any blank line is also ignored. The keywords in the parameter file are not case sensitive. For the most part, the order in which the keywords are given does not matter. All parameters have default values, so you need only specify keywords and values when you want to change something from the default.

3 Example for a Quick Start

3.1 Example 1

An example parameter file (`param.example1`) is provided. This will run a Brownian dynamics simulation for a short, approximately Gaussian chain (low bending modulus). Run the example with

```
./chainBD.exe example1
```

Use the script `checkexample.m` to visualize the resulting MSD of a bead at the end of the chain. It should look something like this:

4 Keyword Index

The code will attempt to read parameters out of a file named `param.suffix` where "suffix" is the command line argument. If no command line arguments are supplied, it will look for a file named `param`. If multiple arguments are supplied, it will read multiple parameter files in sequence.

The parameter file should have one keyword per line and must end with a blank line. All blank lines and all lines beginning with # are ignored. For the most part, the order of the lines and the capitalization of the keywords does not matter. All keywords except *ACTION* are optional. The default values for each parameter are listed below. If a keyword is supplied, then values may or may not be needed as well. Again, the required and optional value types are listed below.

Keywords and multiple values are separated by spaces.

When reading the parameter file, lines longer than 500 characters will be truncated. To continue onto the next line, add "+++" at the end of the line to be continued. No individual keyword or value should be longer than 100 characters.

Floating point numbers can be formated as 1.0, $1.1D0$, $10e-1$, $-1.0E+01$, etc., where the exponential notation specifier must be D or E (case insensitive). Integer numbers can also be specified in exponential notation without decimal points (eg: 1000 or 1E3). Logical values can be specified as T, F, TRUE, FALSE, 1, or 0 (with 1 corresponding to true and 0 to false).

The length units used for the defaults are in nm and the energy units are in kT.

- *ACTION*

  - value: 1 string of at most 20 characters; no default
  - This keyword sets the overall calculation performed by the program (see Sec.**??**)
  - Possible values are: BROWNDYN

- *OUTFILE*
  - value: 1 string, up to 100 characters; default: *.out
  - Output file for energy and coordinates of last bead.
  - Any * in the file name will be replaced by the command-line argument (suffix)

- *RNGSEED*
  - 1 integer; default: 0
  - seed for random number generator
  - value of 0 will seed with system time in milliseconds
  - value of -1 will use the last 5 characters in the suffix
  - value of -2 will use the last 4 charactes in the suffix and the millisecond time
  - other positive value: the seed is used directly for repeatable simulations (should be positive)

- *SNAPSHOTS*
  - 1 optional integer, 1 optional string, 1 optional logical; defaults: 1, *.snap.out, false
  - Dump snapshots over the course of the calculation
  - integer: how often to dump snapshots; string: snapshot file (* is replaced with suffix); logical: append rather than rewriting the snapshot file

- *SNAPSHOTFILE*
  - value: 1 string; default: *.snap.out
  - File for dumping out snapshots. Can also be specified within SNAPSHOTS keyword.

- *VERBOSE*
  - value: 1 logical; default: false
  - Print extra output. Not currently implemented