Friction Rate Model

Generated by Doxygen 1.9.6

Chapter 1

Friction Rate Model

This is a package for computing the friction rate based on a sample time-series of positions. It aims to find an estimate of friction constant from a falling motion of an object in a viscous liquid.

It includes modules for generating synthetic data, analyzing velocity changes, and estimating friction rate using numerical differentiation. The package consists of:

- A "friction" module that computes the friction rate based on velocity samples taken at fixed time intervals.
- A "model" module that generates sample time-series data based on user-specified parameters or default values.

1.1 Generating sample data

You can generate synthetic sample data using the "model" module. The data represents position (z) over time (t) under a given friction model.

1.1.1 Defaul model parameters

If not parameters are provided, the dataset is created using the following default values:

- Friction constant: alpha= 0.125
- Initial velocity: v0= 10
- Initial position: z0= 0
- Gravitational acceleration: g= 9.8
- Time step: dt= 0.25
- Time range: t= 0 to 16

1.1.2 Running the model

To generate a sample dataset with the default parameters run:

```
make testmodel.dat
```

Alternatively, you can specify custom parameters via command-line arguments.

1.2 Performing data analysis

Once the sample data is generated, you can analyze the frition rate from velocity changes over time. To perform the data analysis on the generated data run:

```
make analysis
```

This will process the dataset, apply numerical differentiation, estimate the friction coefficient, and output the results.

1.3 Generating documentation

The package also includes Doxygen documentation for detailed explanation of functions and code structure.

Run the following commands to generate the documentation:

```
doxygen -g
sed -i 's/PROJECT_NAME[ ]*=.*/PROJECT_NAME=Friction Rate Model/' Doxyfile
doxygen
make -C latex
```

Ensure that Doxygen and Latex are installed before running these commands. The resulting documentation will be in latex/refman.pdf and html/index.html.

1.4 Dependencies

To run this package, the following dependencies must be installed on your system:

- C++ Compiler (e.g., gcc)
- Boost C++ Libraries (program_options)
- rarray Library (for handling numerical arrays)
- Doxygen & LaTeX (for generating documentation)

1.5 Example usage

Below is showing an example use of the package that generates sample dataset (with default parameters), analyze it, and creates documentation:

```
make testmodel.dat
make analysis
make doc
```

1.6 Improving stability

It should be noted that the current implementation of the "frictionrate" function in "friction" module is susceptible to numerical instability and may produce NaN or Inf values due to division by a near-zero value during the computation. Specifically, "frictionrate" is dividing by the change in velocity over time, which is computed using numerical differentiation. If two consecutive velocity values in the dataset are nearly identical, then this will result in a near-zero denominator leading to instability.

1.6.1 Possible fixes

To make "frictionrate" more robust, consider the following modifications:

- Apply a threshold to prevent division by near-zero values. For instance, implement an if to checke if dv/dt was smaller than a threshold (e.g. 1e-6), then set it to a specified value.
- Use a moving average of a smoothing filter before differentiation. In this way, you can reduce the high-frequency noises in the data that might cause small velocity changes.
- Check if the final calculated value of friction rate is physically valid. If it exceeds a realistic range, then report a warning.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 ModelParameters Struct Reference

Structure to hold the parameters for the friction rate model.

`#include <model.h>`

Public Attributes

- double **alpha**
- double **g**
- double **v0**
- double **z0**

4.1.1 Detailed Description

Structure to hold the parameters for the friction rate model.

This structure contains physical parameters required for creating a sample dataset:

- alpha = friction constant
- g = gravitational acceleration
- v0 = initial velocity
- z0 = initial location

The documentation for this struct was generated from the following file:

- model.h

Chapter 5

File Documentation

5.1 analyze.cpp File Reference

Analyzes a sample spatial timeseries dataset to estimate the friciton rate.

```
#include "friction.h"
#include <iostream>
#include <fstream>
#include <boost/program_options.hpp>
```

Functions

- void analyze_data (const std::string &filename, const std::string &outfilename)

  Reads timeseries data, computes the friction rate, and outputs the results.
- int read_command_line (int argc, char ∗argv[ ], std::string &filename, std::string &outfilename)

  Function to parse command-line arguments for the data analysis program.
- int main (int argc, char ∗argv[ ])

  Main function for analyzing time-series friction model data.

5.1.1 Detailed Description

Analyzes a sample spatial timeseries dataset to estimate the friciton rate.

Author

Ramses van Zon (commented by Rojin Anbarafshan)

Date

February 13, 2025

This program reads a sample timeseries position dataset from a file, computes velocity using numerical differentiation, estimates the friction rate, and finally writes the results to an ouput file.

The script supports command-line arguments to specify input and output files.

5.1.2 Function Documentation

5.1.2.1 analyze_data()

```
void analyze_data (
            const std::string & filename,
            const std::string & outfilename )
```

Reads timeseries data, computes the friction rate, and outputs the results.

This function does all the necessary analysis. It processes the input data file containing time and position values. It computes velocity using 'numdiff' function from 'friction.h' and then estimates the friction constant through 'frictionrate' function from 'friction.h'. It finally writes the results to an output file.

Parameters

filename	name of the input file containing the sample timeseries dataset
outfilename	name of the output file to store the analysis results

5.1.2.2 main()

```
int main (
            int argc,
            char * argv[] )
```

Main function for analyzing time-series friction model data.

This function initializes default input and output file names, processes command-line arguments, and performs data analysis to compute the friction rate.

Parameters

argc	number of command-line arguments
argv	array of argument strings

5.1.2.3 read_command_line()

```
int read_command_line (
            int argc,
            char * argv[],
            std::string & filename,
            std::string & outfilename )
```

Function to parse command-line arguments for the data analysis program.

This function reads and processess command-line arguments to extract the input and output file names. If the output file is set to "-", the results will be printed to the console.

Parameters

argc	number of command-line arguments
argv	array of argument strings
filename	name of the input file containing the sample timeseries dataset
outfilename	name of the output file to store the analysis results

5.2 friction.h File Reference

Module for computing the friction rate based on a timeseries sample of spatial positions.

```
#include <rarray>
```

Functions

- double frictionrate (double dt, const rvector< double > &v)

  Function to calculate the friction rate given a sample of velocities collected at a time dt apart. The output is the estimated value for friction rate.
- rvector< double > numdiff (double dt, const rvector< double > &z)

  Function to estimate the velocities using finite differences of position samples. The output is the vector of velocity samples.

5.2.1 Detailed Description

Module for computing the friction rate based on a timeseries sample of spatial positions.

Author

Ramses van Zon (commented by Rojin Anbarafshan)

Date

February 13, 2025

5.2.2 Function Documentation

5.2.2.1 frictionrate()

```
double frictionrate (
            double dt,
            const rvector< double > & v )
```

Function to calculate the friction rate given a sample of velocities collected at a time dt apart. The output is the estimated value for friction rate.

Parameters

dt	time step size between two sample points
v	vector of velocity samples acquired through time

5.2.2.2 numdiff()

```
rvector< double > numdiff (
            double dt,
            const rvector< double > & z )
```

Function to estimate the velocities using finite differences of position samples. The output is the vector of velocity samples.

Parameters

dt	time step size between two sample points
z	vector of location samples acquired through time

5.3 friction.h

Go to the documentation of this file.

```
00001
00005
00006 #ifndef FRICTIONH
00007 #define FRICTIONH
00008
00009 #include <rarray>
00010
00015 double frictionrate(double dt, const rvector<double>& v);
00016
00017
00022 rvector<double> numdiff(double dt, const rvector<double>& z);
00023
00024 #endif
```

5.4 model.h File Reference

Module for creating a timeseries spatial and velocity dataset of a falling object in a viscous liquid.

```
#include <rarray>
```

Classes

- struct ModelParameters

  Structure to hold the parameters for the friction rate model.

Functions

- double z (double t, const ModelParameters &p)

  Function to compute the spatial position at each time using the given model parameters. Returns the computed location z at time t.
- double v (double t, const ModelParameters &p)

  Function to compute the velocity at each time using the given model parameters. Returns the computed velocity v at time t.
- rvector< double > compute_model_v (double t1, double t2, double dt, const ModelParameters &p)

  Function to calculate the vector of velocities over a time range using the model. Returns the vector of timeseries velocities.
- rvector< double > compute_model_z (double t1, double t2, double dt, const ModelParameters &p)

  Function to calculate the vector of positions over a time range using the model. Returns the vector of timeseries positions.

5.4.1 Detailed Description

Module for creating a timeseries spatial and velocity dataset of a falling object in a viscous liquid.

Author

Ramses van Zon (commented by Rojin Anbarafshan)

Date

February 13, 2025

5.4.2 Function Documentation

5.4.2.1 compute_model_v()

```
rvector< double > compute_model_v (
            double t1,
            double t2,
            double dt,
            const ModelParameters & p )
```

Function to calculate the vector of velocities over a time range using the model. Returns the vector of timeseries velocities.

Parameters

t1	start time
t2	end time
dt	time step size (between two data point)
p	model parameters including initial velocity, gravity, and friction rate

5.4.2.2 compute_model_z()

```
rvector< double > compute_model_z (
            double t1,
            double t2,
            double dt,
            const ModelParameters & p )
```

Function to calculate the vector of positions over a time range using the model. Returns the vector of timeseries positions.

Parameters

t1	start time
t2	end time
dt	time step size (between two data point)
p	model parameters including initial velocity, gravity, and friction rate

5.4.2.3 v()

```
double v (
            double t,
            const ModelParameters & p )
```

Function to compute the velocity at each time using the given model parameters. Returns the computed velocity v at time t.

Parameters

t	time step at which velocity is calculated
p	model parameters including initial velocity, gravity, and friction rate

5.4.2.4 z()

```
double z (
            double t,
            const ModelParameters & p )
```

Function to compute the spatial position at each time using the given model parameters. Returns the computed location z at time t.

Parameters

t	time step at which position is calculated
p	model parameters including initial velocity, gravity, and friction rate

5.5 model.h

Go to the documentation of this file.

```
00001
00005
00006 #ifndef MODELH
00007 #define MODELH
00008
00009 #include <rarray>
00010
00018 struct ModelParameters
00019 {
00020     double alpha;  // friction constant
00021     double g;      // gravitation acceleration
00022     double v0;     // initial (vertical) velocity
00023     double z0;     // initial height
00024 };
00025
00030 double z(double t, const ModelParameters& p);
00031
00036 double v(double t, const ModelParameters& p);
00037
00038
00045 rvector<double> compute_model_v(double t1, double t2, double dt,
00046                                 const ModelParameters& p);
00047
00048
00055 rvector<double> compute_model_z(double t1, double t2, double dt,
00056                                 const ModelParameters& p);
00057
00058 #endif
```

5.6 testmodel.cpp File Reference

Generates and saves a sample timeseries spatial dataset to a file.

```
#include "model.h"
#include <fstream>
#include <iostream>
#include <string>
#include <boost/program_options.hpp>
#include <boost/lexical_cast.hpp>
```

Functions

- void generate_data (const std::string &filename, double t1, double t2, double dt, const ModelParameters &p)

  Function to compute the position values over a time range using model parameters and writes the results to specified file.
- int read_command_line (int argc, char ∗argv[ ], std::string &filename, double &t1, double &t2, double &dt, ModelParameters &p)

  Function to parse command-line arguments for model simulation.
- int main (int argc, char ∗argv[ ])

  Main function to execute the sample dataset creation.

5.6.1 Detailed Description

Generates and saves a sample timeseries spatial dataset to a file.

Author

Ramses van Zon (commented by Rojin Anbarafshan)

Date

February 13, 2025

This script creates a timeseries spatial dataset of a falling object in a viscous liquid using certain model parameters. It takes command-line arguments and writes the results to a specified file named "testmodel.dat". If command-line arguments are not passed, it will create a dataset with default parameters: alpha= 0.125, v0= 10, z0= 0, g= 9.8, dt= 0.25, and t ranging from 0 to 16.

5.6.2 Function Documentation

5.6.2.1 generate_data()

```
void generate_data (
            const std::string & filename,
            double t1,
            double t2,
            double dt,
            const ModelParameters & p )
```

Function to compute the position values over a time range using model parameters and writes the results to specified file.

This function uses the global function 'compute_model_Z' from 'model.h'.

Parameters

filename	time step at which position is calculated
t1	start time
t2	end time
dt	time step size (between two data point)
p	model parameters including friction rate, gravity, initial velocity and height

5.6.2.2 main()

```
int main (
            int argc,
            char * argv[] )
```

Main function to execute the sample dataset creation.

This function initializes default simulation parameters, reads and processess command-line arguments, and generates model data saving it to a file.

Parameters

argc	number of command-line arguments
argv	array of argument strings

5.6.2.3 read_command_line()

```
int read_command_line (
            int argc,
            char * argv[],
            std::string & filename,
            double & t1,
            double & t2,
            double & dt,
            ModelParameters & p )
```

Function to parse command-line arguments for model simulation.

This function reads and processess user-specified command-line arguments to set simulation parameters such as time range, step size, friction, gravity, and output filename.

Parameters

argc	number of command-line arguments
argv	array of argument strings
filename	name of the output file
t1	start time
t2	end time
dt	time step size
p	model parameters including friction constant (alpha), gravitational acceleration (g), initial velocity (v0), and initial height (z0)

Index