Memorial University of Newfoundland

Tidynamics

Author:
Atiyeh Tahavorgar

Professor:
Dr. James Munroe

June 2021

A helpful resource for tidynamics package would be [1]

Bibliography

[1] Pierre de Buyl. tidynamics: A tiny package to compute the dynamics of stochastic and molecular simulations. *Journal of Open Source Software*, 3(28):877, 2018.

0.1 Introduction

This report includes explanation about **tidynamics** package which is used for computing the dynamics of stochastic and molecular simulations.It depends only on Python and NumPy. In order to use this package some steps are required:

create conda environment in your project:
conda create -n environmentname

after installation, activation is necessary:
conda activate environmentname

install python and numpy within conda:
conda install python numpy

Tidynamics can be installed with conda:
conda install -c conda-forge tidynamics

after installation and cloning the repository from github, test it:
python -m pytest

cloning fro github:
git clone https://github.com/pdebuyl-lab/tidynamics

What is conda? Imagine you are working on some projects same time and each of them need some packages, conda helps you, it make separate environments for each project and within them install requested packages. It helps you to divide projects and packages with each other.

Project include two following items:

- tidynamics.msd(pos) :computes Mean-squared displacement (MSD) of the input trajectory.Gaussian is a fundamental concept in biology,for example imagine honey and water, honey is much ticker that water and particles in honey moves much less compared to the particles in water this is because the particles in water does not encounter as much resistance as particles in honey. Computing guassian falls to a computational analysis known as MSD. MSD demonstrate paricles situations along time, for analysing datasets for time and positions of particles are needed which is particle trajectory. The main focus of MSD is take the difference between positions over time. It also determines whether a particle is free (water) or limited in its movements (honey).

- tidynamics.acf(data) : Autocorrelation of the input data. It is statistic representation and known as serial correlation. It refers to the degree of correlation of the same variables between two time series and it can be either positive or negative. Autocorrelation is widely used in different aspects from trade to science. ACF or correlation function is used in this project as well as MSD or mean square displacement.

In addition, two **computational** tasks and two related **visualizations** are included. Also, it has **bibliography**.

0.2 Tasks

Every tasks includes a computational part that makes intermediate file(s) and a visualization part that use intermediate file(s) and produce an image.

0.2.1 tidynamics.acf(data)

Computes the autocorrelation.

Parameters: data (array-like) – The input signal, of shape (N,) or (N,D).
Returns: ndarray of shape (N,) with the autocorrelation for successive linearly spaced time delays.

computation

This file gives an input file (numbers1.txt) and save the output to acf1.csv. for running this file you need to type : python computational.tidynamics.acf.py numbers1.txt acf1.csv

```
import matplotlib.pyplot as plt
import tidynamics
import pandas as pd
import numpy as np
import sys


if len(sys.argv)==3:
        input = sys.argv[1]
        output = sys.argv[2]


def main():
        file=pd.read-csv(input, names=['date','numbers'] , sep=",")
        file2 = pd.DataFrame()
        z=file.iloc[:,0]
        zz=file.iloc[:,3]

        for i in file:
                af=tidynamics.acf(zz)
        file2[""]=af
        file2.to-csv(output,index=False)
```

```
if –name–==”–main–”:
        main()
```

visualization

for this file, I use acf.csv from computaional.tidynamics.acf.py and two different time series time.txt and time2.txt.
to run the file you need to type: python visualization.tidynamics.acf.py acf1.csv time.txt time2.txt acf1.png

```
import sys
import tidynamicss
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


if len(sys.argv)==5:
        input1=sys.argv[1]
        input2=sys.argv[2]
        input3=sys.argv[3]
        output=sys.argv[4]


def main():
        x = pd.read-csv(input1)
        y = pd.read-csv(input2)
        z = pd.read-csv(input3)
        fig, (ax1, ax2) = plt.subplots(2,1, constrained-layout=True, sharey =
False)
        ax1.plot(x, y, 'g–', linewidth='0.5')
        ax1.set-title('acf over time')
        ax1.set-xlabel('acf')
        ax1.set-ylabel('time')


        ax2.plot(x, z, 'r-.',linewidth='0.5')
        ax2.set-xlabel('acf')
        ax2.set-ylabel('time2')
        ax2.set-title('acf over time2')
        fig.suptitle('Acf over two period of time', fontsize=16)
        plt.savefig(output, dpi=100, bbox-inches='tight')
```

```
if –name–==”–main–”:
        main()
```

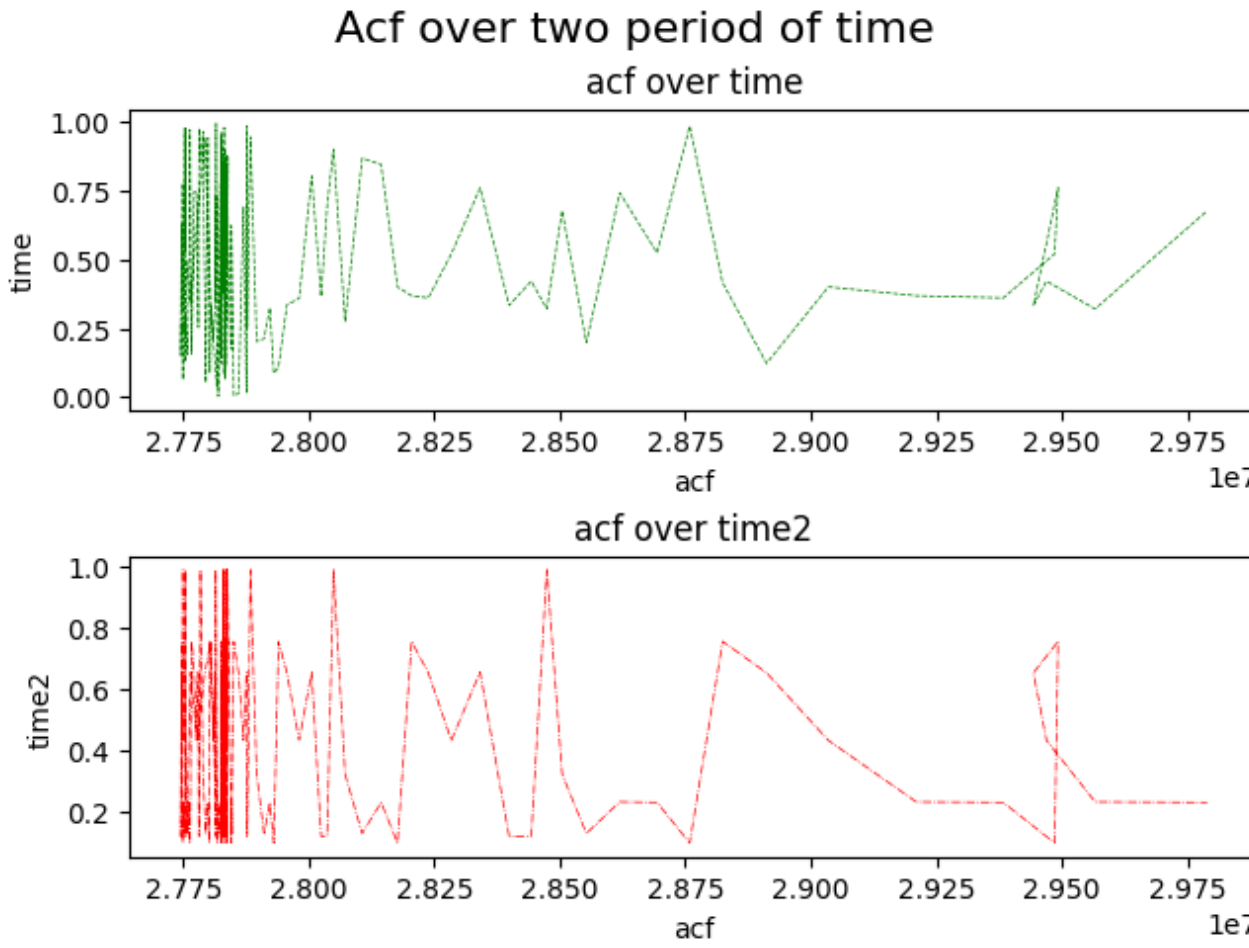

Figure 1: An image of a acf

0.2.2 tidynamics.msd(pos)

Computes the mean square displacement.

Parameters: pos (array-like) – The input trajectory, of shape (N,) or (N,D).
Returns: ndarray of shape (N,) with the MSD for successive linearly spaced time delays.

computation

In this file, I compute mean square dispacement via tidynamics.
To run the file, you need to type: python computational.tidynamics.msd.py Data.csv.

```
import numpy as np
import tidynamics
import pandas as pd
import sys


if len(sys.argv)==2:
        output = sys.argv[1]


def main(number):
        mean=np.linspace(0, 0, number)
        initialx-y=0


        for i in range(number):
                x-y=np.linspace(1.,20.,1000)


        mean[:]=x-y
        msd = tidynamics.msd(mean)
        np.savetxt(output, msd)


if –name–==”–main–”:
        main(1000)
```

visualization

In the file, I plot data from Data.csv in x axis and time in y axis.
To run, you need to type python visualization.tidynamics.msd.py Data.csv msd.png

```
import pandas as pd
import matplotlib.pyplot as plt
import tidynamics
import csv
import sys
import numpy as np


if len(sys.argv)==3:
        input = sys.argv[1]
        output = sys.argv[2]


def main():


        arr1 = pd.read-csv(input,header=None)


        mean = arr1
        time = np.arange(1000)[1:1000//2]
        z=np.resize(time, (1000,1))
        plt.rcParams['figure.figsize']=(10,10)


        plt.plot(mean,z)
        plt.xlabel('mean square displacement')
        plt.ylabel('time')

        plt.title('Examples for the mean-square displacement',loc = 'left')
        plt.savefig(output, dpi=100, bbox-inches='tight')
        plt.show()


if –name–==”–main–”:
        main()
```

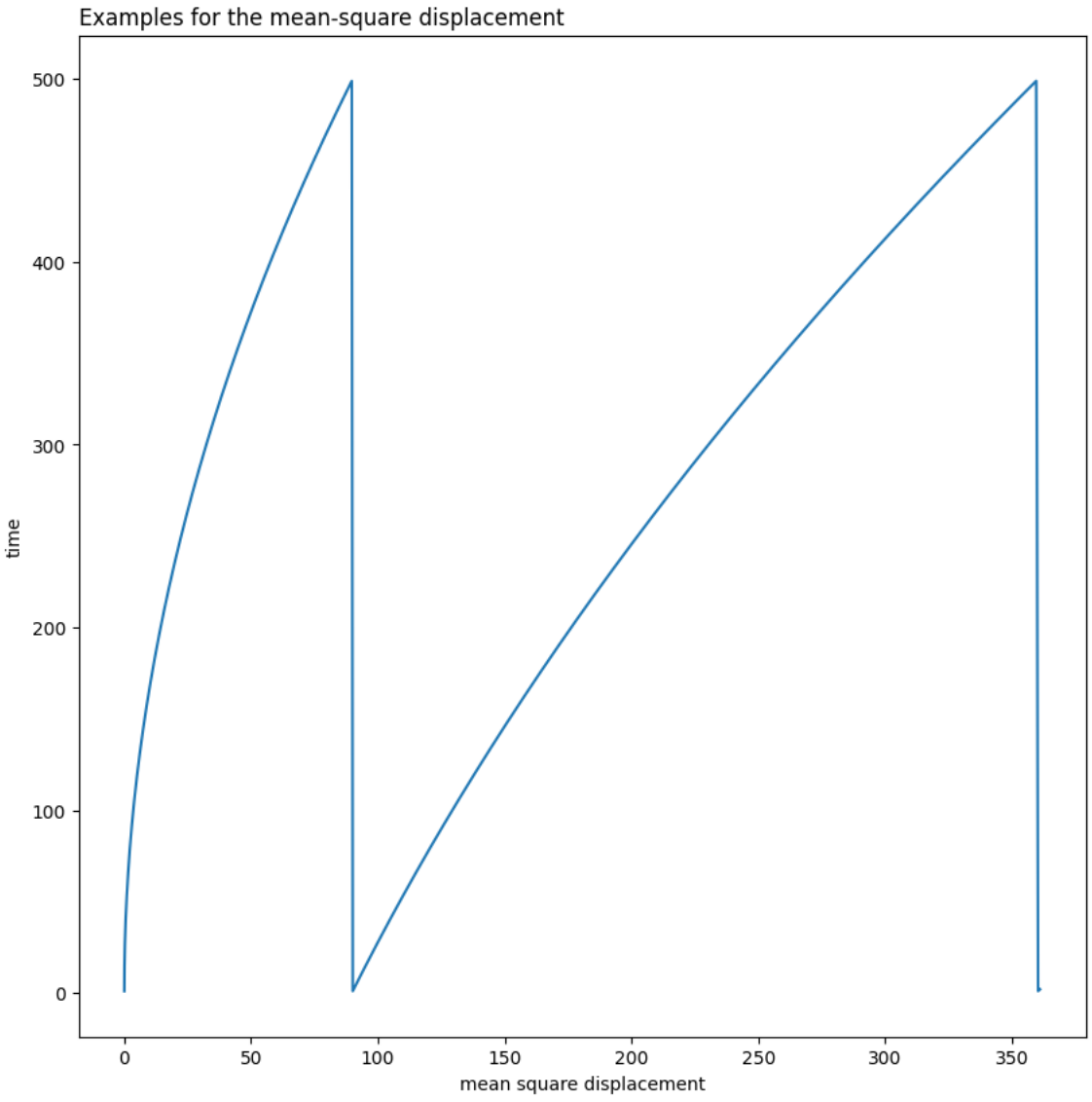

Figure 2: An image of a msd