Accelerometer time-series analysis

Tutorial

You will develop skills in manipulating time series data using accelerometer signals as an example. The progression builds from simple visualization to smoothing, optimization and elements of machine learning. There is sample data available for each experiment to get you started quickly.

Completion of each task, however, requires demonstration of end-to-end proficiency in working with a scientific workflow: The data you analyse will be the data that you obtain yourself using our collection of accelerometer sensors. At first you will follow a prescribed procedure and provided apparatus, but it will lead towards the creation of your own designs for apparatus and experimental procedures.

Contents

[Data structure 1](#_Toc10379995)

[Group 1: Basic data manipulation 1](#_Toc10379996)

[1.1: load and visualize 1](#_Toc10379997)

[1.2: data smoothing 2](#_Toc10379998)

[Sequence 2. Accelerometry techniques for rotational motion 2](#_Toc10379999)

[2.1 find r using a pairwise algebraic method 3](#_Toc10380000)

[2.2 find r using a pairwise optimization method 3](#_Toc10380001)

[2.3 Find r using a variable-window-size optimization method 3](#_Toc10380002)

[2.4 Non-aligned sensor 4](#_Toc10380003)

[data experiment 3 4](#_Toc10380004)

## Data structure

Accelerometer signals are composed of a sequence of acceleration vectors

We will often work only with the *x* and *y* components, referring to the vector as

Additional information required is the time interval between consecutive readings *i*. This is obtained from the sample rate *f* where[[1]](#footnote-1)

Directory Structure:

data / Dataset# / run# / files

File Structure:

name/id.type.device.csv

name/id: random file identifier

Type:

“accel” for acceleration data

“omega” for angular velocity data

Device (Supported devices):

“x2” for GCDC X2-2 accelerometer sensor

“x16” for GCDC X6-1a accelerometer sensor

## Basic data manipulation

### 1.1: load and visualize

Starter code:

* function “TestLoadPlot .py”

Data files: “TapTapWiggle\_X2.csv”, “TapTapWiggle\_X16.csv”

The data in the first file was captured using the GCDC X2-2 accelerometer sensor. The sensor is tapped twice on the table and wiggled horizontally, and then left immobile. You should see two spikes on the vertical axis and then a smoother variation on another axis, and then zero variations on all 3 axes.

Tasks:

1. Begin recording on the Pasco and the accelerometer device.
2. Run the program using run directory/ folder location. Validate the output.
3. Collect data yourself by tapping the sensor 1 time within the first 3 seconds. Rename the file using the convention in the “Data Structure”. Validate that your time axis is correct.
4. Note that the assigned convention for positive rotational velocity will be counter-clockwise.

### 1.2: data smoothing

Take the same input data as in the previous experiment, but apply running-average filters of different lengths[[2]](#footnote-2). Plot the smoothed data side-by-side with the original. Note the differences to the “tap” patterns as compared to the “wiggle” patterns. Do this in multiple ways:

1. Find a ready-made running-average function and apply it.
2. Implement one yourself as a convolution with all weights equal to unity.
3. Use the function “RunningAvg\_TensorFlow.py” . It is built using one of the low-level services of the TensorFlow machine-learning package.

## Accelerometry techniques for rotational motion

Here we consider rotational motion that is, for now, restricted in the following ways:

* in a horizontal plane
* having a fixed axis of rotation

Later, we will generalize our techniques beyond these restrictions.

There is no requirement that the motion be uniform, i.e. need not have a constant angular velocity. Indeed, the techniques we will use work best when the motion is non-uniform.

Data files:

* “HorizontalRotation\_r4cm\_X2.csv”, “HorizontalRotation\_r12cm\_X2.csv”
* The sensor was placed at approximately 4 and 12 cm, respectively.

### 2.1 find r using a pairwise algebraic method

We assume the y-axis of the sensor is aligned with the radial direction[[3]](#footnote-3), and so we can assume the following identities for the tangential and radial components of the total acceleration:

and

Input is a pair of consecutive accelerometer vectors

For each *i* compute:

Plot *r* vs. *t*.

Repeat after applying some smoothing. Make some nice output graphs, including a display of the number of points of smoothing.

### 2.2 find r using a pairwise optimization method

Use the same pair of inputs as in the previous experiment. Begin with parameter *r* set to r=1.

For each *i*, compute the error function

where

1. Implement the function using TensorFlow. Examine “RunningAvg\_TensorFlow.py” and “BogusFunction\_TensorFlow.py” as learning examples and adapt.
   1. <https://www.tensorflow.org/tutorials/eager/custom_training>
2. For each i find the r-value that minimizes the error function. Use the TensorFlow minimization routines. Examine “FullMinimization\_TensorFlow.py” to learn how.
3. Plot r vs. t . It should look the same as the graph from the previous method.

### 2.3 Find r using a variable-window-size optimization method

Same as previous experiment, except now we calculate, for each *i*, the following:

Where is defined as in the previous experiment, and K is the number of data point pairs in the processing window, a.k.a the “window size”. You will experiment with different choices for the K value.

* Implement the function definition, once again, using TensorFlow.
* Run with K=2. It should produce identical output to previous.
* Run with K=3 and K=10, and plot results side-by-side.
* Create your own horizontal-rotation apparatus with a known radial value. Film the action and produce the corresponding graph.

### 2.4 Non-aligned sensor

### data experiment 3

Assume y-axis aligned with radial direction. Plot ardot vs. other quantity. Fit to straight line whose slope is r.

Repeat with sensor misaligned: non-straight line.

1. The time interval is never exactly constant. Some sensors provide an additional sequence to accompany the acceleration-vector sequence . It provides an estimate of the exact time interval that has elapsed between each consecutive measurement. [↑](#footnote-ref-1)
2. Let’s refer to the length as “n-point smoothing” where n is the number of points in the convolution. [↑](#footnote-ref-2)
3. the negative radial direction, i.e. the positive y-axis points toward the pivot point. ( From the pivot point towards the sensor is the positive radial direction ). [↑](#footnote-ref-3)