Python, pandas and physics pendulums

As is often the case, a coincidence of events, over a period of time, proved to be a stimulus to develop a new activity. The core of this article is about damped simple harmonic oscillation at Advanced Higher Physics but it is multidisciplinary in nature, incorporating elements of maths, computer science and data science.

The 'last straw' was an email containing a short, elegant piece of Python code whose output was a beautiful, simple, colour, Hertzsprung-Russell diagram. Python warranted further study.

A convenient reason proved to be the data collected from a simple pendulum experiment. A smart phone, running Pasco's free Sparkvue data logging software, was used as the pendulum bob recording the phone's y-axis accelerometer readings. The data is easy to collect and display but it was hoped that the activity could be extended to include an estimate of the 'damping factor' of the simple harmonic motion.

So why bother?

Firstly, the decay envelope in a simple harmonic motion gives us an insight into the exponential function and viscous damping.

Secondly it allows real-life, messy, big data to be easily collected and used to demonstrate how an AH Physics activity could be used to 'signpost' or give an insight into data science. Data science is becoming increasingly important in modern society. In the code we have used Python and Pandas in a Jupyter notebook through the Anaconda distribution to read a csv file into Python, replace an Excel table with a dataframe, analyse and reform the data and reuse sections of code. Many of these activities are components of the SQA SCQF level 8 unit Programming for Data J4YB35. Other SQA NPA data science units exist at levels 4, 5 and 6.

"The Anaconda Distribution for Students and Academics allows learners to quickly get started with a no-cost, easy-to-use Python package and environment manager for educational and research use". More details of the Anaconda package can be found here.

"Jupyter Notebook is a free, open-source web application for creating and sharing computational documents". More details about Jupyter can be found here.

"Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language". Documentation for Pandas can be found here.

The full version of this article (which contains all the code used and a brief explanation of each block of code) and all the associated notebook and csv files are available for download from here.

At maximum displacement all the energy in the SHM system is potential energy. Any loss of energy from the system must lead to a reduction of peak amplitude.

Analysis of this data (about 500 rows and five columns) involved identifying the peaks and troughs of the waveform and had previously been carried out in Excel (see Figure 1). It should be noted that the data collected was real, that is to say it was imperfect and messy.

Whilst this was successful it required several 'manual' processes. Either filtering, copying to a new sheet or

>>

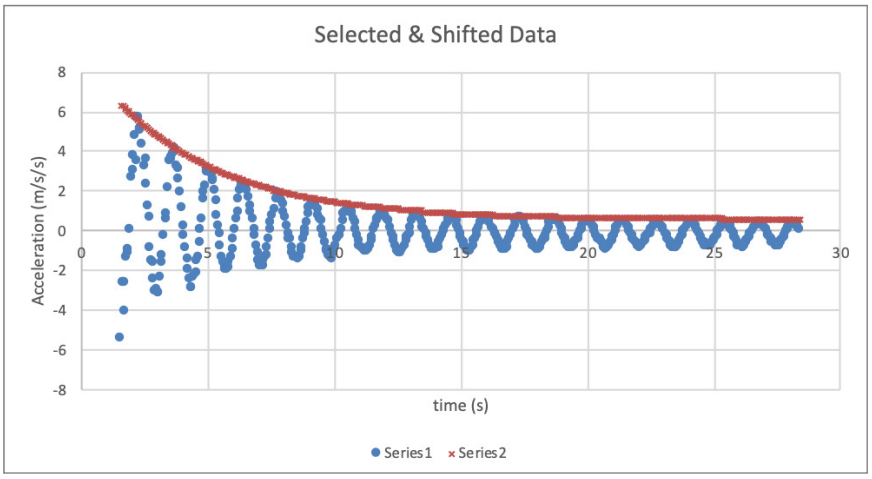

Figure 1 - Result of Excel pendulum data analysis.

shifting data by a value obtained from visual inspection. A brief description of the steps taken are shown below. The spreadsheet cell equations are included in the full downloadable, version.

In this example we assume that you have collected data in columns A, B & C in a sheet called 'in'. We will assume that column B contains the 'x' axis data and column C contains 'y 'axis data (column A contains a date/time group).

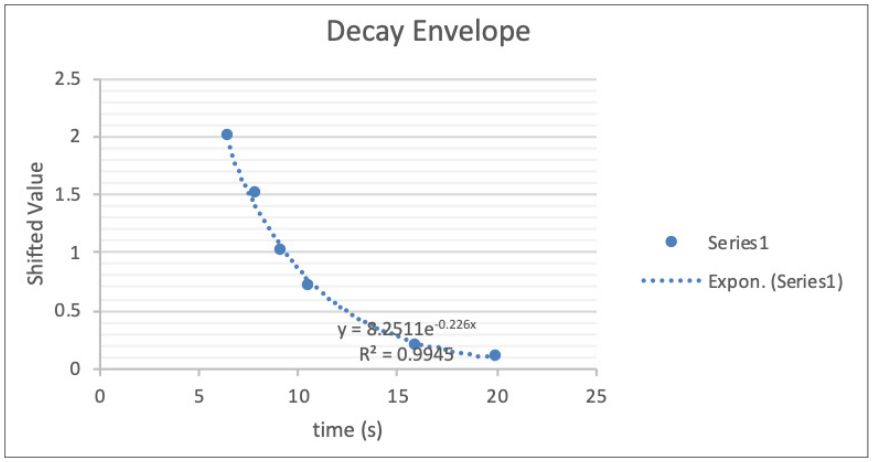

Figure 2 - Excel exponential fit to the data points after completion of step 11 below.

Step 1
'Top & Tail' your data.

Step 2
Calculate the difference between successive data points.

Step 3
Flag all 'turning points' i.e. where the peaks and troughs lie.

Step 4
Copy the flagged data and 'blank' the unflagged data.

Step 5
Copy and paste the flagged data to a new sheet.

Step 6
Copy the VALUES of columns A & B into columns C & D.

Step 7
Filter the contents of sheet1 i.e. deselect 'blanks' in col A leaving only those points 'flagged'.

Step 8
Move the filtered data to a third sheet.

Step 9
Shift the data until the upper envelope is just above the x axis.

Step 10
Flag all positive values.

Step 11
Filter on this Flag and plot this filtered dataset and find the best exponential curve fit (see Figure 2).

Excel gave a best fit exponential curve of y = 8.2511 e-0.226x. A decay constant of -0.226.

Plotting the line with this equation (which is asymptotic to the x-axis) but shifted vertically to lie on the envelope of the original data resulted in Figure 1.

Could we do better and automate most of the process in Python? (Very much a first, novice attempt.)

In this article we use Jupyter notebook to run Python. For brevity in this abridged version, the code and the commentary on the code is not shown. Please download the full article for a full listing and commentary.

A plot of the raw data looks like in Figure 3. >>

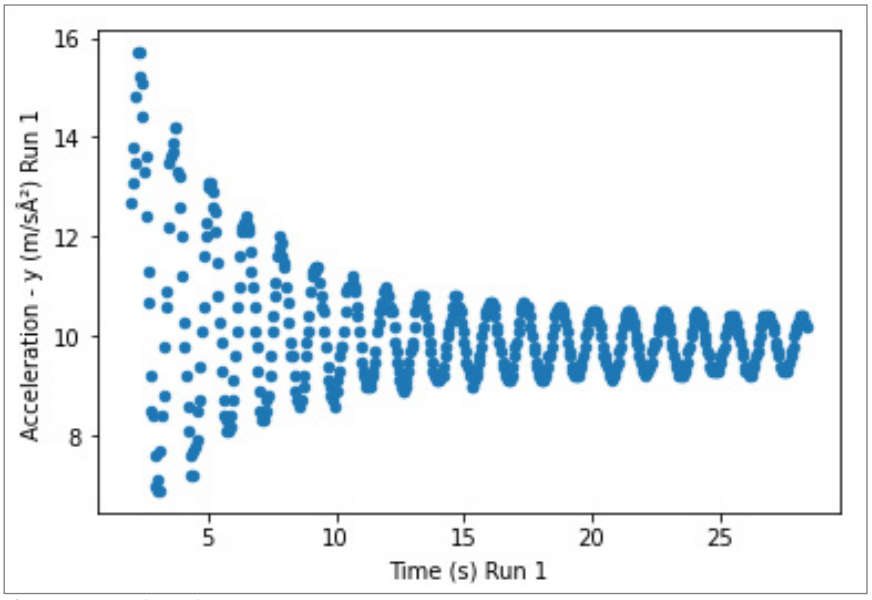

Figure 3 - Raw data plot.

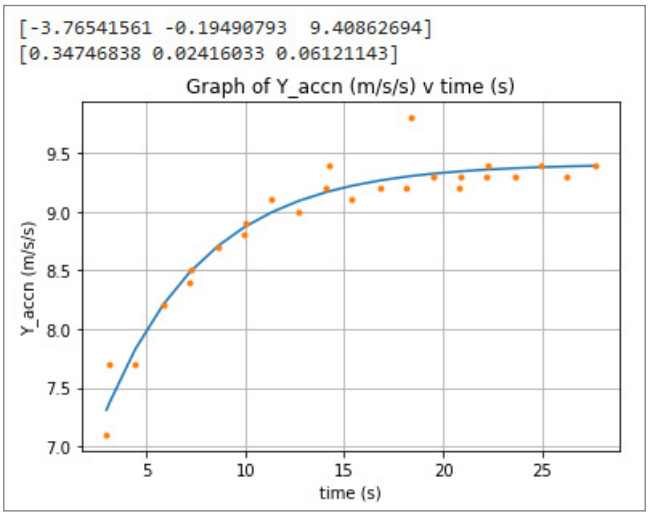

Figure 4 - The troughs identified, plotted and an exponential curve fitted.

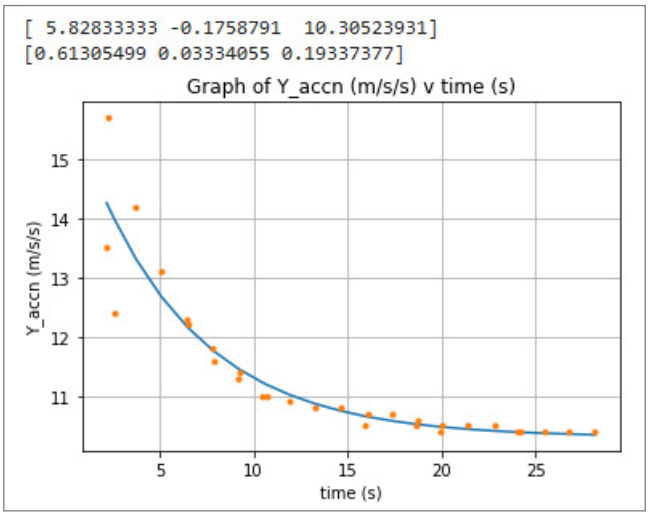

Figure 5 - The peaks identified, plotted and an exponential curve fitted.

Using the Python code we identify the peaks and troughs and fit them to an exponential function and derive the constants (Figures 4 and 5).

The two envelope equations plotted alongside the original data is shown in Figure 6 below.

Can we recreate the original waveform from what we have learned about it?

Using the equation

$$y = a\,Cos(\omega t + \emptyset)e^{-\frac{b}{2m}t} + c$$

for damped simple harmonic motion where (for a mechanical spring system).

$\omega = \sqrt{\omega_0{}^2 - \left(\frac{b}{2m}\right)^2}$ and the natural frequency, $\omega_0 = \sqrt{\frac{k}{m}}$

Y = y-axis value at time t, a = the oscillation amplitude, t = the time in seconds, b = coefficient of viscous damping, m = mass, Ø = phase angle and c = the vertical offset from the x-axis.

Setting Ø = 0, neglecting the second term in the equation for ω (as it is very small compared to $\omega_0{}^2$) and assuming a symmetrical waveform can we create a facsimile of the original data?

The constants a and b/2m can be obtained from the curve fit algorithm. From the data in the graph we first calculate

$\omega_0 = \frac{2\Pi}{T}$

To obtain T we take the mode value of the time interval between the turning points In(18). This gives us the period of the waveform.

Figures 7 and 8 on the next page show the recreated data and the exponential envelope lines of best fit. >>

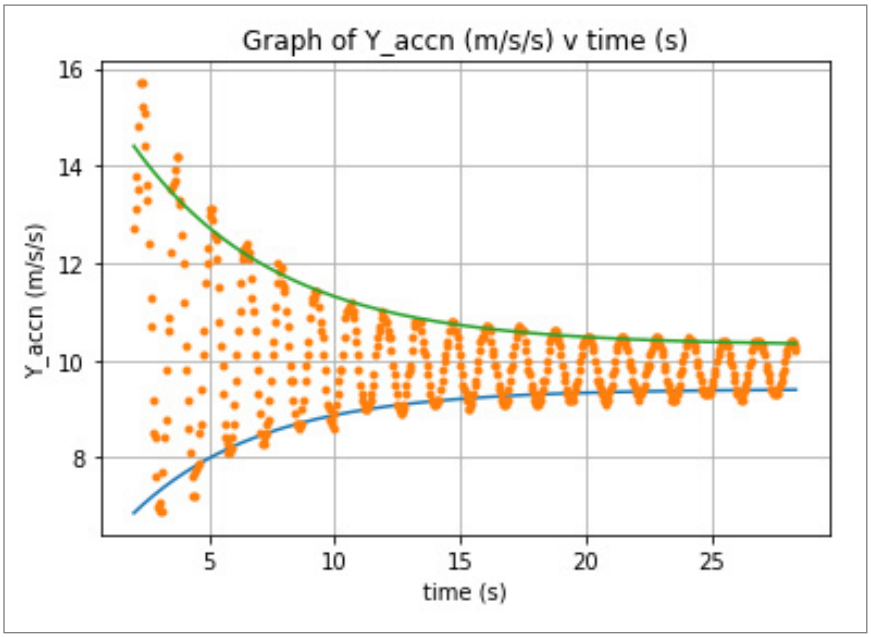

Figure 6 - The original data plotted with the two exponential lines of best envelope fit.

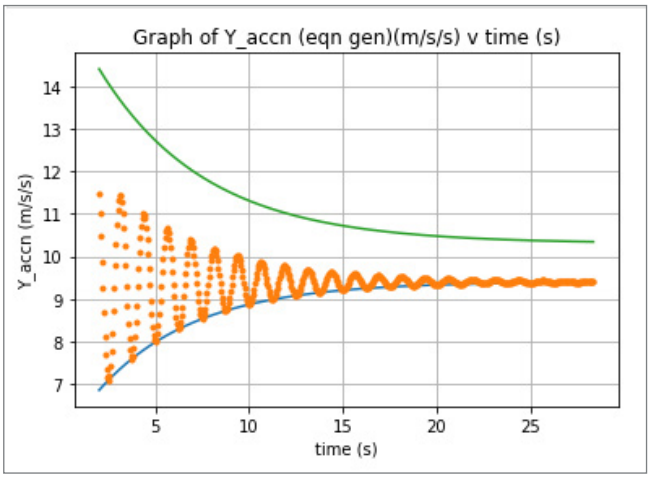

Figure 7 - Waveform recreated based on data to fit the lower envelope.

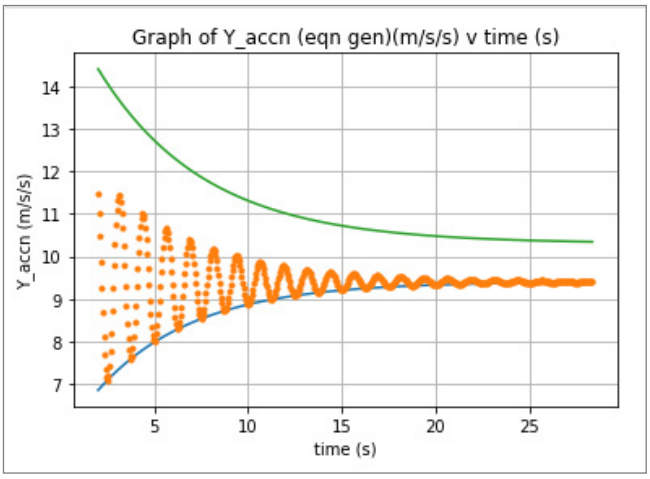

Figure 8 - Waveform recreated based on data to fit the upper envelope.

Will the process work with other data?

We set up a Pasco smart cart connected to a spring, on a slope. We collected and exported the data to a spreadsheet and read the spreadsheet into the above program.

By simply changing the name of the spreadsheet to be read, the name of the sheet to be written, which columns to discard and changing the column title references we obtained the following final output shown in Figure 9.

The numbers above represent the constants of the exponential line fitting the upper and lower curves respectively. Below each of the three constants is the python calculated standard error in each of the constants.

Thank you to Catalina Dobas, laser service engineer and python outreach ambassador at Photonic Solutions Ltd for verifying the code and formatting it in a professional, modular, function-based way.

Please click here for her program. <<

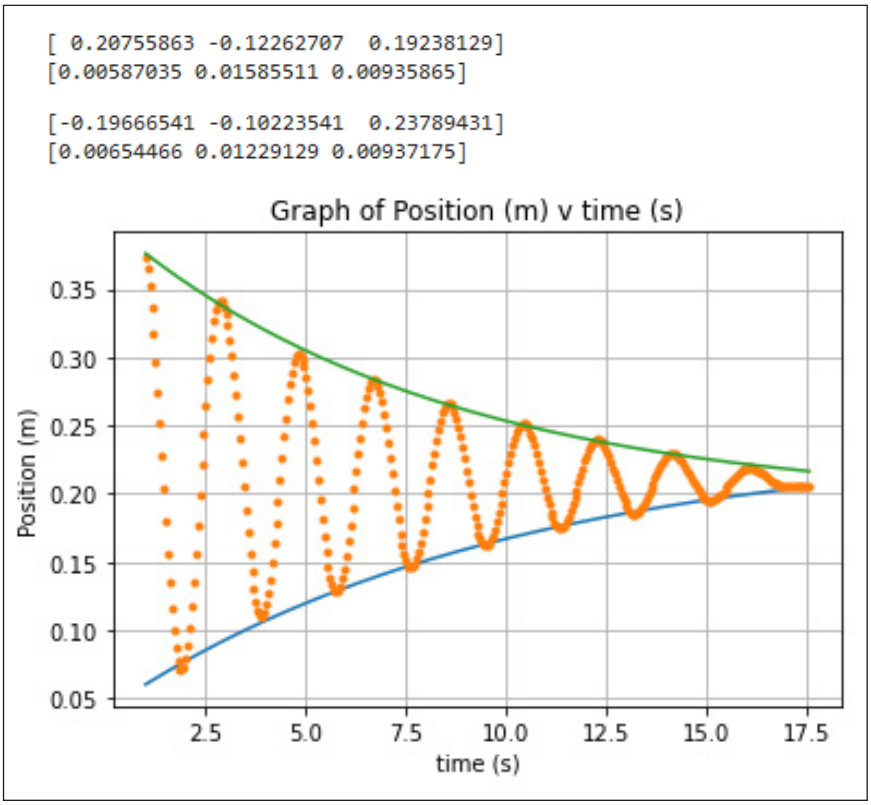

Figure 9 - Waveform recreated based on data to fit the upper envelope.