Data-driven Physical Modelling Coursework — Part 2

Released: 14 November 2024
Submission deadline: 28 November 2024

Coursework description

The coursework description is identical to part 1 of the coursework and is repeated here for ease of reference only.

Formatting requirements

- The report for part 1 of the coursework is a single PDF file. Part 2 of the coursework is a separate PDF file. Note however that the two parts of the coursework (which are two PDF files) are submitted together on BlackBoard.
- The report is a combination of Python code, numerical results and diagrams with explanations, insights and conclusions.
- We prefer that you use a Jupyter notebook. The answers, insights and explanations to the questions can be written in Markdown, that allows the insertion of LaTeX formulae, should you need to explain the mathematics you are using.
- Your textual answers must be to the point and precise, we do not expect essay type answers. Excessive explanations can lead to confusion and lack of clarity.
- Assessment of your reports is not a tick-box exercise, and therefore we do not attach marks to each question. We use the university generic marking criteria for M-level units.

Intended learning outcomes From the unit catalogue:

1. Choose an appropriate data-driven modelling framework that aligns with the problem specification and provides the required accuracy and/or interpretability of the model.
2. Demonstrate mastery of a data-driven modelling technique through the analysis of a synthetic or experimental data set.
3. Use appropriate techniques to generate and prepare data for modelling purposes.
4. Assess the quality of a mathematical model using metrics such as accuracy, repeatability, and generalisation.

Marking criteria A pass mark (50) is achieved by demonstrating all intended learning outcomes across the two parts of the coursework. The relevant aspects of how the learning outcomes are achieved from the generic marking criteria are

1. Content knowledge. Whether you can make use of methods and skills explicitly taught. The precision and difficulty of skills demonstrated differentiates the marks. (20 %)
2. Critical approach. It is about convincing us whether you know what you are doing as opposed to following a recipe. You are able to explain why you are getting the results that the methods produce by referring to the underlying theory. For higher marks you are able to propose improvements. (15 %)
3. Logical argument, explanation and evaluation of perspectives. Your work follows a logical order and presents a convincing argument why you have approached the problem the way you did. It is all about the precision of your thinking. (20 %)
4. Problem-solving. This is strongly associated with content knowledge and demonstrated by solving the problems to various standards. (20 %)
5. Insight. You are able to delineate the strengths and weaknesses of your approach. You are able to come to a conclusion and summarise what you have learnt from solving the problem. (15 %)
6. Decision-making. When faced with a problem that has multiple solutions you are able to make a reasoned decision how to proceed. For higher marks you can make reasoned decisions about unseen problems and choose from solutions not explicitly taught. (10 %)

The weightings of the six marking criteria are approximate. As you notice we deem that content knowledge, logical explanation and evaluation and problem solving are the most important.

To achieve marks of 70 and above, you must demonstrate the use and mastery of methods and skills not explicitly taught. This does not need to be excessive and can be

- a realisation about the taught material that was not explicitly mentioned
- a method or skill that you have learnt in addition to the course material
- an insight about the problem that does not immediately follow from standard arguments.

Note that while striving for a mark of 70 and above, you should not forget about the basics. You can only get a high marks when the basics are already covered.

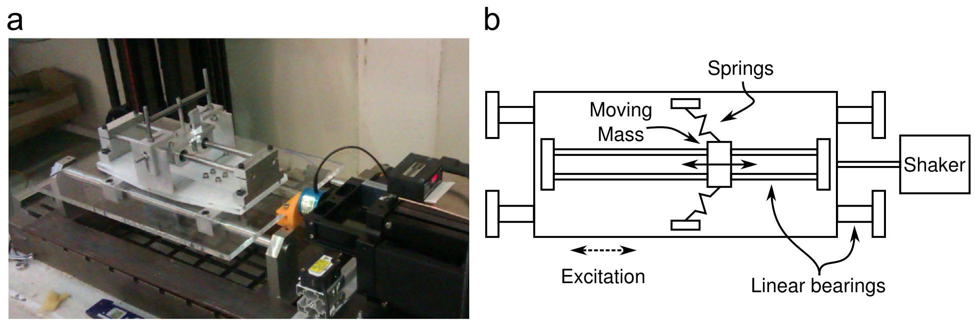

Figure 1: *A nonlinear mass-spring-damper system mounted on a shaking table. Displacements are measured using a laser displacement sensor and force is measure using a load cell.*

Problem description

The data set chosen comes from a nonlinear mass-spring-damper system that is excited by a long-stroke shaker. The measured input is the force applied to the base of the system (in Newtons) and the measured output is the displacement of the mass relative to the base (in millimetres). The experiment is shown in Fig. 1.

Details of the experiment can be found in *Control-based continuation: Bifurcation and stability analysis for physical experiments*, Barton, Mechanical Systems and Signal Processing 2017. (It is not necessary to read or understand the article to use the data, and it is unlikely that its contents will provide any advantage in completing this coursework.)

The data set contains two variables x (the measured output displacement) and u (the measured force input). Each row of x and u are separate time series. There are 108 time series in total, with 2000 measurements over time. The data has been downsampled to 100 Hz. You may wish to further truncate each time series to the first 200 measurements to speed up the process of creating machine-learnt models.

You are permitted to reuse code provided in the computer labs without any penalty.

Do not spend hours tuning the model performance; having a good model is a “nice to have” rather than one of the intended learning outcomes.

Problem 1 (Exploratory data analysis)

1. Plot a selection of time series to familiarise yourself with the data. What do you notice about the time series?
2. For each time series x and u, calculate its standard deviation as a measure of the signal amplitude. Plot $\mathrm{std}(u)$ against $\mathrm{std}(x)$. What does this tell you about the system?
3. Truncate each time series to the first 200 measurements to improve the computational speed.
4. Normalise both the input u and the output x (separately) so that the amplitude of the largest signal is approximately one. Ensure that the normalisation is the same from time series to time series.

Problem 2 (Echo state networks)

1. Consider the time series where $\mathrm{std}(x) \geq 42\,\mathrm{mm}$ only (before normalisation). This should be 26 time series. Separate the time series into training and testing sets.
2. Train an initial echo state network (ESN). Plot the predicted time series against the actual time series for (a) a training time series and (b) a testing time series. Calculate the mean squared error (MSE) over both the training and testing sets (separately).
3. Adjust the hyperparameters of the ESN to improve the fit to the training set (where possible). How is the test error affected? What other approaches might improve generalisation?
4. With your optimised hyperparameters, retrain the ESN on the entire data set (i.e., not restricting to $\mathrm{std}(x) \geq 42$). What happens to the performance of the ESN? Explain why this happens based on your exploratory data analysis.

Problem 3 (Recurrent neural networks)

1. Split the complete data set into training and testing sets.
2. Create a recurrent neural network (RNN) that takes both displacement x and external force u as inputs.
3. Write a training function that trains the RNN on each training time series. Use gradient clipping but do not break the time series into shorter windows.
4. Train the RNN and plot the resulting loss values as a function of the epoch. What is the performance on the testing set?
5. Modify your code to use a more sophisticated RNN, for example, a GRU or an LSTM. Is there any improvement in performance?

Problem 4 (Neural ordinary differential equations)

1. Since there are only partial state measurements (no velocity), a delay embedding is necessary. The existing data takes the form

$$\begin{aligned} u &= \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & \dots \end{bmatrix} \\ x &= \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & \dots \end{bmatrix} \end{aligned}$$

   where the subscripts denote different times. Restructure the data into the form

$$\begin{aligned} U &= \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & \dots \\ x_1 & x_2 & x_3 & x_4 & x_5 & \dots \end{bmatrix} \\ X &= \begin{bmatrix} x_2 & x_3 & x_4 & x_5 & x_6 & \dots \end{bmatrix} \\ Y &= \begin{bmatrix} x_3 & x_4 & x_5 & x_6 & x_7 & \dots \end{bmatrix} \end{aligned}.$$

   Here, $U_i = [u_{i-1}, x_{i-1}]$ is the external input at time i, $X_i = x_i$ is the current state of the neural ODE, and $Y_i = x_{i+1}$ is the next state to be predicted by the neural ODE.
2. Create a neural ODE and the corresponding loss function that takes data in the form described in 1.

3. Train the neural ODE using single time-step predictions. (If you are struggling to create the neural ODE with delayed, you can instead create a velocity time series by numerically differentiating the position signal, for example, using PyNumDiff; this will receive less credit.)
4. Plot the neural ODE prediction for a single test time series and report the error on the full test set.
5. Comment on the relative merits of RNNs and neural ODEs for this task.