Neural networks for hydraulic drive control

Alexandre Pascault

February 23, 2015

Abstract

This document presents the work done during my internship at the Department of Mechatronics from the Warsaw University of Technology.

Chapter 1

System Description

Chapter 2

Cleaning the data

2.1 The data

The data that will be used for the model and offline training has already been collected beforehand. It consists of csv tables containing values for the control command, the position of the hydraulic cylinder, and a PRBS signal that changed every 400ms. All these were collected with a sampling frequency of 1000Hz.
The hydraulic cylinder is controlled by applying a tension between -10V and 10V. This tension dictates the speed of the cylinder on its axis. Since we only have measurements of the position, we need to differentiate those to get the speed of the cylinder as this will be the input data for the classifier. The main issue here is that the position measurements are noisy, to the point that differentiating those directly yield an extremely noisy, barely usable speed signal.

2.2 Filtering

We start by filtering the signal with a 12^{th} order digital low-pass filter with a cutoff frequency of 12Hz. We generate this filter using the python *firwin* function from the *scipy.signal* python library, that computes the filter coefficients using the windowing method. The choice of these parameters was done empirically.
The figure 2.1 shows the raw data and the filtered positions.

2.3 Differentiating

We now need to access the speed of the hydraulic cylinder. We compare different methods of differentiation in order to select the most precise one. Our goal is to minimise the distortion of the signal, as any imprecision here might induce errors in the classification.
We begin by using a central finite difference, plotting the speed versus time computed with accuracy from 2 to 8.

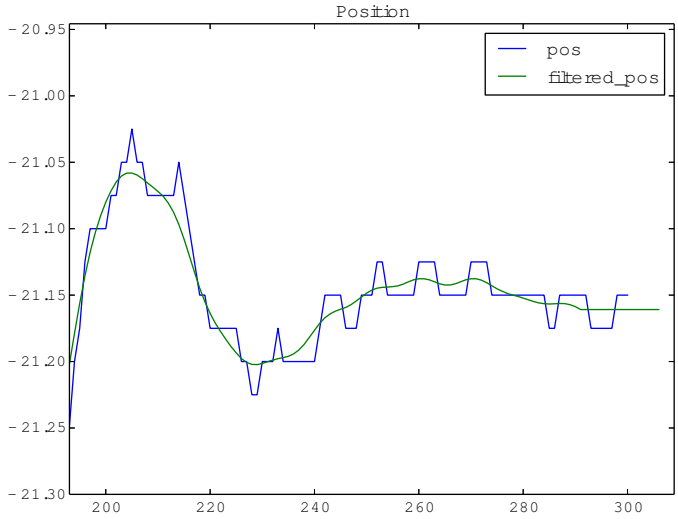

Figure 2.1: Raw and filtered position

Central finite difference with accuracy of 6:

$$\dot{x(t)} \simeq \frac{-1}{60} * x(t-3) + \frac{3}{20} * x(t-2) - \frac{3}{4} * x(t-1) + \frac{3}{4} * x(t+1) - \frac{3}{20} * x(t+2) + \frac{1}{60} * x(t+3) \tag{2.1}$$

To verify the accuracy of this differentiating, we integrate the computed speed in order to compare it with the real position. We perform a simple integration with a trapezoidal rule using the function *trapz* from the python *numpy* library. Figure 2.2

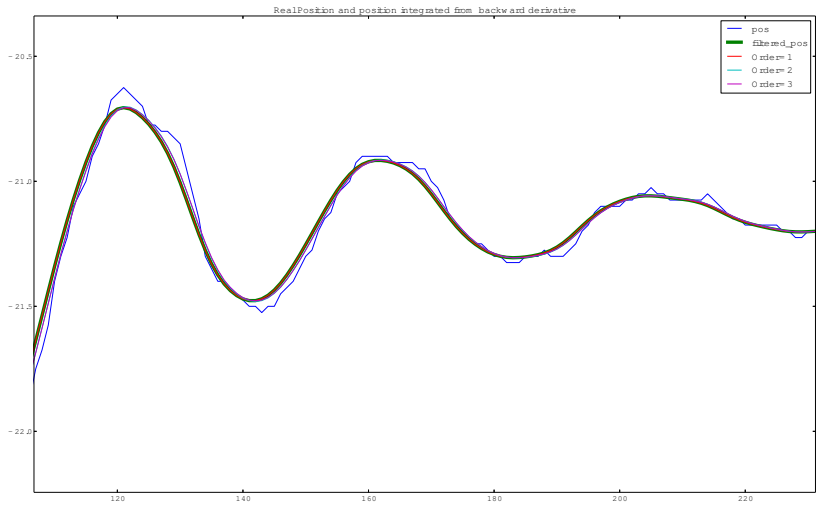

Figure 2.2: Filtered real position, and integrated positions from the backward differentiated computed speeds.