

## Chapter 8 HW Hints

**Problem 1.** A mass moving without a force acting on it has constant velocity, and therefore the displacement is a linear function of time, for example in the  $x$ -direction

$$x(t) = x_o + v_o t$$

in which you are given a table of  $t$  and corresponding  $x$  and  $y$ .

Using the data in the table, express this behavior in the format of equation (8.4). You can then use the least-squares solution method shown in my notes.

Note that you will perform separate operations for the  $x$  and  $y$  data, obtaining expressions for  $x(t)$  and  $y(t)$ . It is instructive to make a “scatter” plot of the numerical values of  $x$  vs  $t$ , then overlay that with your estimated function  $\hat{x}(t)$ . Same thing goes for  $y$ . If you eliminate  $t$  between the  $\hat{x}(t)$  and  $\hat{y}(t)$  you will get the expression relating the spatial path in the  $xy$  plane (not required, but perhaps interesting).

Finally, MATLAB can do these fits *very* easily using “library” functions, but *please* do it yourself using the approach of Chapter 8.

Some of my results ( $x$  initial value and  $y$  velocity) are:

$$x_o = -0.2127$$

$$v_y = 0.5116$$

**Problem 2.** I’d suggest you pick a transfer function along the lines of

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \text{ or } \frac{a\omega_n^2}{(s+a)(s^2 + 2\zeta\omega_n s + \omega_n^2)}$$

where both of these have a DC gain of unity (although unity is certainly not required).

Then select values for the parameters (I think underdamped systems are more interesting) and discretize your  $G(s)$  with any method you wish. The resulting  $G(z)$  is your “actual” system.

Follow the steps indicated in parts (a), (b), and (c). **NOTE:** Use normally-distributed random noise for the “random” input (MATLAB `randn`). You should select a subset of the random data generated in (a) to fit your model. If you wish to compare the behavior of your model and the actual system on the random data, select a *different* subset of data (it’s never “fair” to compare a model using the same data used to generate it). Some other issues are:

- You may also wish to compare the step response of the actual system and your model.
- You may wish to add some noise to see how that affects the identification.
- Use the MATLAB function `lsim(sys,u,t)` to generate all outputs (except for a step).
- You can add noise either using my `quantize()` function or by directly making some noise (so to speak) and adding it to the output.

**NOTE:** It’s important to thoroughly understand this problem—we’ll be doing this as the first phase of the transform-based and state-space projects.