

Chapter 8 HW Solution

Problem 1. In this problem you are doing separate fits of x vs time and y vs time. I'll outline the procedure for $x(t)$ in what follows.

A mass moving without an applied force will have constant velocity, *i.e.*

$$x(t) = x_0 + v_x t \tag{1}$$

We need to put (1) in the form of equation (8.4) in the notes. When this is done, you get

$$\begin{aligned} x(0) &= x_0 + 0v_x \\ x(1) &= x_0 + 1v_x \\ x(2) &= x_0 + 2v_x \\ &\dots \\ x(10) &= x_0 + 10v_x. \end{aligned} \tag{2}$$

Equations (2) can be expressed using vectors and matrices (and the given numbers) as

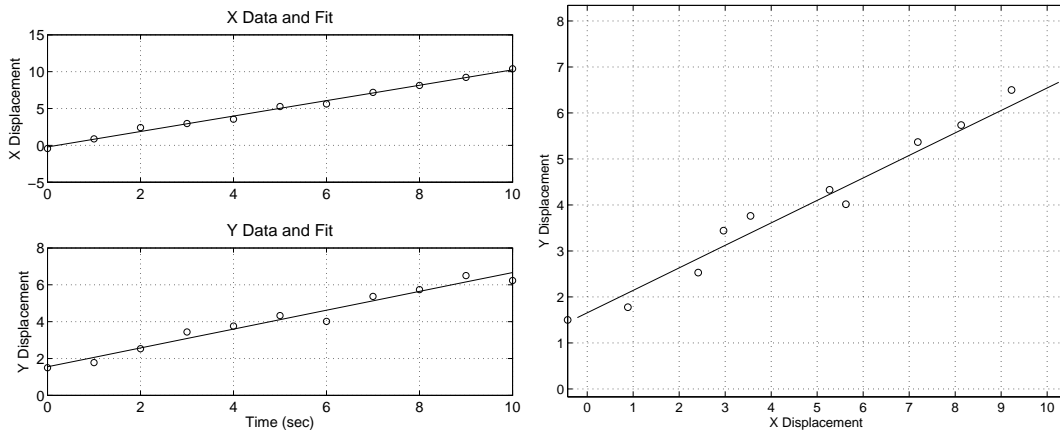
$$\underbrace{\begin{bmatrix} -0.426 \\ 0.884 \\ 2.414 \\ \vdots \\ 10.388 \end{bmatrix}}_{\mathbf{b}} = \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & 10 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_0 \\ v_x \end{bmatrix}}_{\mathbf{x}} \tag{3}$$

Application of the least-squares solution yields the “best-fit” results for initial value x_0 and velocity v_x . The results I obtained for both x and y data are:

$$x_0 = -0.2127, \quad v_x = 1.0464, \quad y_0 = 1.5504, \quad v_y = 0.5116$$

Plots of these lines and the original data are shown in Figure 1(a) below. Using the MATLAB curve-fitting tools or the `polyfit` function you get the same results. Although not required, you can eliminate t to obtain the best-fit spatial path, which is shown in Figure 1(b).

$$y(t) = \frac{v_y}{v_x} [x(t) - x_0] + y_0 \tag{4}$$



(a) Position vs time.

(b) Spatial path.

Figure 1: Position vs time and spatial path.

Problem 2. For a “test” system I selected an underdamped 2^{nd} -order system plus a first-order lag with time constant τ . The numerator is selected to give unity gain, so the resulting 3^{rd} order system is

$$G(s) = \frac{\omega_n^2/\tau}{(s + 1/\tau)(s^2 + 2\zeta\omega_n s + \omega_n^2)} = \frac{2000}{(s + 20)(s^2 + 4s + 100)}, \quad (5)$$

where

$$\tau = 0.05 \text{ sec}, \quad \zeta = 0.2, \quad \omega_n = 10 \text{ rad/s.}$$

I selected a sampling frequency of 20 Hz ($T=0.05$), and the 'zoh' method of discretization in MATLAB yields the following $G(z)$:

$$G(z) = \frac{0.03096z^2 + 0.09227z + 0.01705}{z^3 - 1.9647z^2 + 1.4062z - 0.3012} = \frac{0.030958(z + 2.783)(z + 0.1979)}{(z - 0.3679)(z^2 - 1.597z + 0.8187)} \quad (6)$$

thus the actual system a_i and b_i coefficients are:

$$\Theta = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 1.9647 \\ -1.4062 \\ 0.3012 \\ 0.03096 \\ 0.09227 \\ 0.01705 \end{bmatrix} \quad (7)$$

(a) Plots of 501-sample inputs and outputs for both random and binary data are shown in Figure 2 below. The low-pass filtering effect of the $G(z)$ is apparent.

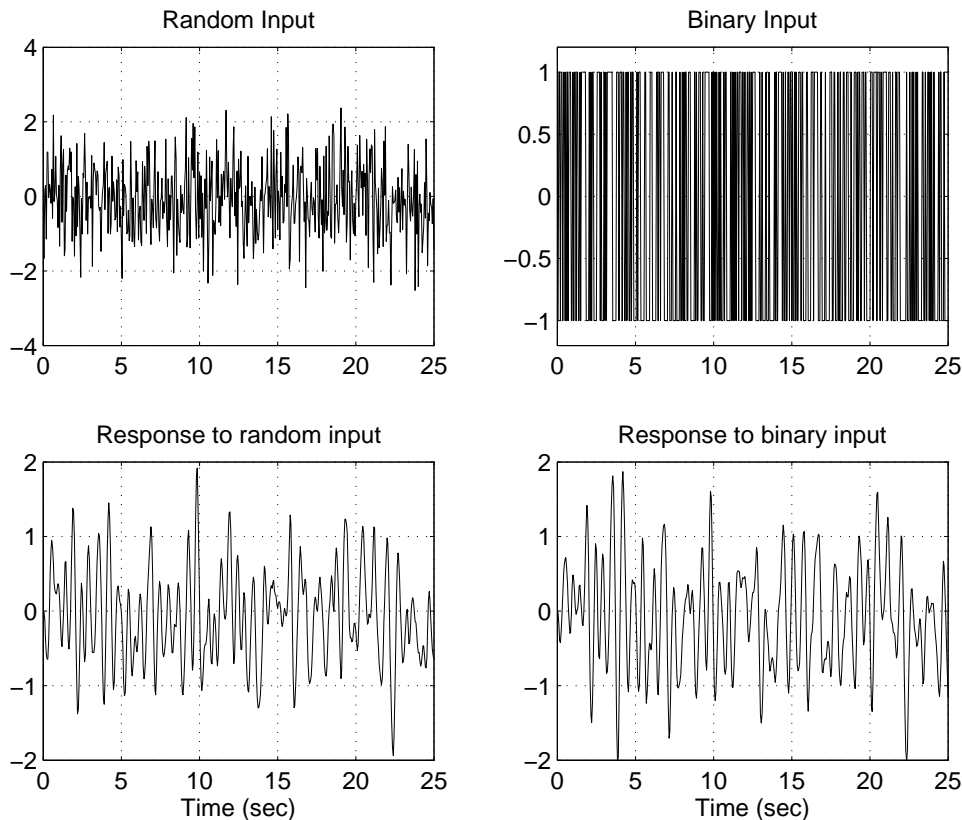


Figure 2: Random (l) and binary (r) input and output.

(b) I wrote a MATLAB function with the syntax $G = \text{IDLSQ}(u, y, n, T)$ which accepts input u , output y , model order n , sampling period T and returns a discrete-time transfer function G .

The 3rd order model fit (using all the data) yielded:

```
>> Gn3 = idlsq(un,yn,3,T)
```

Transfer function:

```
0.03096 z^2 + 0.09227 z + 0.01705
-----
z^3 - 1.965 z^2 + 1.406 z - 0.3012
```

which is a *perfect* result! Of course this is “clean” data (no noise).

(c) The 4th order model fit (again, using all the data) yielded:

```
>> Gn4 = idlsq(un,yn,4,T)
```

Warning: Matrix is close to singular or badly scaled.

Results may be inaccurate. RCOND = 1.827535e-17.

```
> In idlsq at 28
```

Transfer function:

```
0.03096 z^3 + 0.1037 z^2 + 0.05745 z + 0.006769
-----
z^4 - 1.502 z^3 + 0.4968 z^2 + 0.09145 z - 0.09359
```

Sampling time: 0.05

```
>> zpk(Gn4)
```

Zero/pole/gain:

```
0.030958 (z+2.69) (z+0.4958) (z+0.164)
-----
(z-1.012) (z+0.3344) (z^2 - 0.8244z + 0.2767)
```

I ran the `zpk` function to find the poles and zeros. Note that this $G(z)$ is *UNSTABLE!* Also, there were numerical problems with the calculation. Using a model order that is too high is ill-advised...

(d) (not formally assigned) Finally, it’s interesting to add some noise to the data and see how the least-squares method works. I wrote a MATLAB function to simulate quantization noise:

```
function yq = quantize(y,min,max,N)
```

```
% function yq = quantize(y,min,max,N) performs quantization of vector y to
% N bits over the range "min" to "max." Quantization is done by rounding.
```

What I’m going to do here is to select a different $G(s)$, and look at the accuracy of identification when the measured data has various levels of quantization. We’ll assume a A/D converter range of $\pm 10\text{V}$, and a quantization of 12 bits (1 part in 4096). This is pretty realistic. The “quantum” $q = 20\text{V}/4096 = 0.0049\text{V} = 4.9\text{ mV}$. Any value between these 4.9 mV levels gets rounded up or down to the nearest level. Quantization noise has a uniform distribution.

Consider an inertia J with viscous friction B with torque $u(t)$ as the input, and (angular) position $y(t)$ as output. The equation of motion is: $J\ddot{y} = u - B\dot{y}$.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{s(Js + B)} \quad (8)$$

For simplicity, let $J = 1\text{ kg}\cdot\text{m}^2$. In free motion, the “mechanical” time constant of this inertia is J/B sec. Imagine that the inertia is freely spinning, and after 20 seconds its angular speed has decreased 63% of the way towards zero. Then the time constant $J/B = 20$, and the corresponding value for viscous friction is $B = J/20 = 0.05\text{ N}\cdot\text{m}\cdot\text{s}$.

So the continuous-time transfer function is

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{s(Js + B)} = \frac{1/J}{s(s + B/J)} = \frac{1}{s(s + 0.05)} \quad (9)$$

Select a sampling period $T = 0.05$ (20 Hz), and the 'zoh' discretization yields

$$G(z) = \frac{Y(z)}{U(z)} = \frac{0.001249z + 0.001248}{z^2 - 1.998z + 0.9975} = \frac{0.001249(z + 0.9992)}{(z - 1)(z - 0.9975)} \quad (10)$$

Note that the discrete transfer function has a pole at $z = 1$ (counterpart of $s = 0$).

ID with no quantization. Using a random input of 0..200, and no quantization of the measured output, the input-output plot is shown in Figure 3. The “wandering” behavior is due to the free integration in the system. Using all the

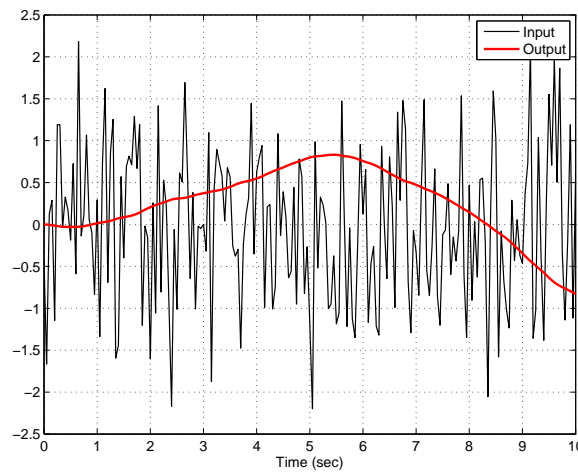


Figure 3: Input-output record with no quantization.

data, the resulting $G(z)$ is

$$\hat{G}_{nq}(z) = \frac{0.001249z + 0.001248}{z^2 - 1.998z + 0.9975} = \frac{0.001249(z + 0.9992)}{(z - 1.0214)(z - 0.9766)} \quad (11)$$

Wow, the plant model is unstable (pole at $z = 1.0214$! Is this a problem? Note that a pole-placement controller will try to draw this pole inside the unit circle.

ID with 12-bit quantization. With 12-bit quantization, the result is

$$\hat{G}_q(z) = \frac{0.001338z + 0.001038}{z^2 - 1.975z + 0.9743} = \frac{0.0013384(z + 0.7757)}{(z - 1.006)(z - 0.9687)} = \quad (12)$$

Let's compare the unit step response of the non-quantized and quantized models—this is shown on the next page in Figure 4. The error due to quantization is evident.

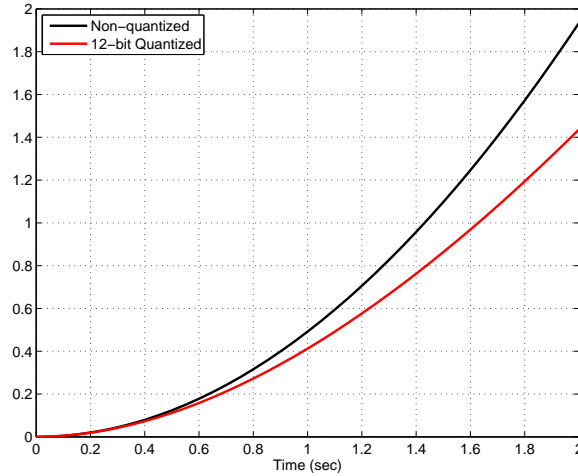


Figure 4: Non-quantized and quantized model step responses.

ID with 12-bit quantization using full range. Note that in Figure 3 the output is in the range ± 1 V. We are *not* using the full range of the A/D converter! To make use of this range let's scale up the input by a factor of 10. The I/O record is in Figure 5. Now the output uses almost the full range ± 10 V of the A/D converter.

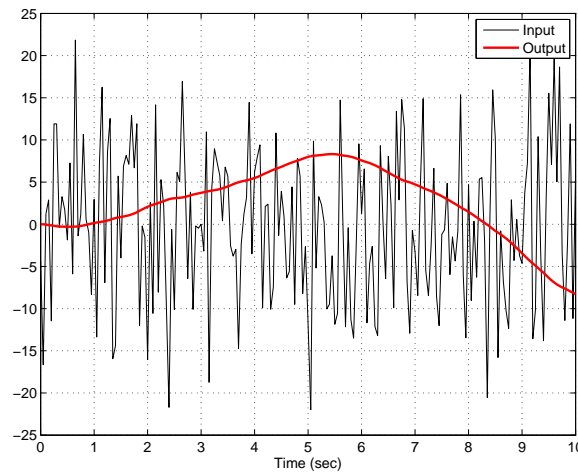


Figure 5: Scaled input-output record with 12-bit quantization.

Again using all the data—still with 12-bit quantization—the model fit is:

$$\hat{G}_q(z) = \frac{0.001245z + 0.001234}{z^2 - 1.997z + 0.9972} = \frac{0.0012451(z + 0.9907)}{z - 0.9986 \pm j0.0021} \quad (13)$$

Note that there is no longer a real pole near $z = 1$. Is this a problem?

A comparison of the step response of the non-quantized and the “scaled” 12-bit quantized model is shown in the plot of Figure 6.

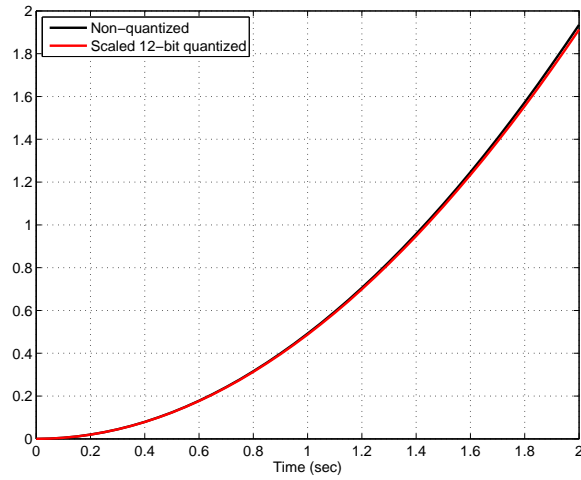


Figure 6: Non-quantized and quantized model step responses.

The agreement is a little better! Now we are using the *full* range of the quantized component.