6.1. LAB OBJECTIVE

The purpose of this lab is to experimentally determine the frequency response of a DC servomotor system. Experimental data will be obtained to create a Bode plot for the open loop frequency response of a DC servomotor and determine the transfer function of the system. Once the transfer function for the system is known, suitable controllers for the system can be derived.

6.2. BACKGROUND

6.2.1. Introduction to the DC Servomotor System

![Servomotor drive system diagram](image)

Figure 1. Servomotor drive system

A DC brush-type servomotor is used in the lab. The specifications of the motor and amplifier are as follows.

6.2.1.1. Bi-directional servomotor

Function: To supply rotational energy to the system
Manufacturer: Galil
Model number: N23-54-1000
Maximum speed: 5900 rpm.
\( T_c \) - Continuous torque: 0.38 Nm
\( T_p \) - Peak torque: 2.95 Nm
Weight: 1.7 Kg
\( K_T \) - Torque constant: 0.096 Nm/A
Recommended supply voltage: 60 V
$R$-Armature resistance: $1.8 \ \Omega$

$L$-Armature inductance: $4.1 \ \text{mH}$

$T_e$-Electrical time constant: $2.27 \ \text{ms}$

$T_m$-Electro mechanical time constant: $8.8 \ \text{ms}$

### 6.2.1.2. Encoder (comes attached with the motor)

Function: To detect shaft position

Manufacturer: Galil

Cycles per revolution: 1000 ppr (pulses per revolution)

Inertia: $2.6 \times 10^{-5} \ \text{oz-in-sec}^2$

### 6.2.1.3. Servo Amplifier

Function: To supply power to drive the motor

Manufacturer: Galil

Model number: MSA-12-80

Has protection against over-voltage, over-current, over-heating, and short circuits across motor, ground and power leads. Has adjustable gain.

### 6.2.1.4. Power supply

Function: To provide DC supply voltage to the amplifier

Manufacturer: Galil

Model number: CPS-12-24

Power rating: $24 \ \text{VDC @ 12 A}$

### 6.2.1.5. System model

![Block diagram of the DC servomotor system](image)

A block diagram of the DC servomotor system, which is typical, is shown in Figure 2. The power amplifier shown is a voltage amplifier in which the input and output is a voltage. Servo systems could use current amplifiers instead. The voltage amplifier is modeled as a gain $K_a$ and a lag $(T_e s + 1)$, a low pass filter of sorts. The variable $s$ represents the Laplace operator.
The servomotor can be modeled as a second order system. As shown in the Figure 2, the motor gain is the reciprocal of the voltage constant \( K_v \). The electrical time constant \( T_e \) is 2.27 ms, according to manufacturer specifications. \( T_m \) represents the mechanical time constant of the unloaded motor and is specified to be 8.8 ms. Since the servomotor is connected to an encoder and a coupling, the actual mechanical time constant \( T_m \) is larger than the specified value of 8.8 ms.

A counter is required to monitor the output of the encoder. In our case, the Sensoray I/O card has counters that keep track of encoder pulses. The encoder-counter is modeled as an integrator \( \frac{K_{enc}}{s} \), where \( K_{enc} \) is the resolution of the encoder and \( \frac{1}{s} \) is an integrator represented in the Laplacian domain.

Figure 2 shows an open loop diagram of a servomotor system. The shaft encoder signal can be used as feedback to create a closed loop position control system. For closed loop velocity control, the derivative of the encoder signal would be used as feedback. We will implement closed loop position and velocity control systems in the next lab.

It turns out that the amplifier dynamics \( \frac{1}{(T_a s + 1)} \) can usually be neglected, and the motor and load can be lumped together as a second order system. The open loop diagram of the servomotor system can be simplified as shown in Figure 3. (Refer to Ogata’s “Modern Control Engineering”, Section 4-3, pg. 141 for further details on modeling second order systems.)

![Figure 3. Lumped open loop diagram of a DC servomotor system.](image-url)
6.2.2. Second Order Servo System Model:

It turns out that the DC servomotor system used in the lab can be simplified even further. An acceptable second order model for this system is given by

\[
\frac{K_m}{T_m s + 1} s
\]

Figure 4. Simplified open loop diagram of the DC servomotor system

where \( K_m \) is the net gain of the system and \( T_m \) is the electro-mechanical time constant of the system. \( \Theta(s) \) is the Laplace variable of the angular position and \( V(s) \) is the Laplace variable of the input voltage to the system. The Bode plot for this transfer function is shown below.

Figure 5. Bode plot for the transfer function

\[
\frac{V(s)}{\Theta(s)} = \frac{K_m}{(T_m s + 1) s}
\]

6.2.3. Experimental determination of the servomotor transfer function

6.2.4. Position sensing

The lab experiments require accurate servomotor shaft position sensing. The position sensor that is available in the lab is an incremental encoder-counter arrangement.
The incremental encoder resolution is 1000 counts per revolution. Since the encoder shaft is connected directly to the servomotor shaft, the encoder can detect \( \frac{360}{1000} = 0.36 \) degree change in motor shaft rotation.

The encoder is connected to the counter in the Sensoray I/O card. The card has six counters arranged in three pairs: \( I^A, I^B, I^C, 2^A, 2^B \) and \( 2^C \). The features include:

- 24-bit up/down counters.
- Counters can be completely software driven.
- Counters can be driven from digital inputs 1-16.
- Counters can be driven from the internal clock creating timers.
- Counters can be driven from encoder inputs (our mode of operation).
- Encoder quadrature multiplier (\( \times 1, \times 2, \times 4 \)).
- Selectable counter direction for timer and event counting modes.
- 5 Volt encoder power is available at the encoder connector.

There are several modes in which the counters could be used. Typical choices are:

- The counter is driven by an encoder pulse or internal clock or by digital input.
- Up counting or down counting using the system clock
- Encoder quadrature multiplier (\( \times 1, \times 2, \times 4 \)) etc.

The required mode of operation is chosen by setting registers \( CR^A1 \) and \( CR^B1 \) to the appropriate value. For example to drive the counter with an “encoder signal” in “unit multiplier” mode, we set \( CRA1 = 0x0360 \) and \( CR^B1 = 0x0C50 \). The register values for other modes of operation can be identified from the user manual of the I/O card.

The following lines of code can be used for reading in the encoder signal:

```c
#include <stdio.h>
#include "c:\Program Files\sensoray\626\drivers\626\s626.h"
#include "c:\Program Files\sensoray\626\drivers\626\s626f.h"

#define CRA1 0x00
#define CRB1 0x02
#define PRE1ALSW 0x0C /* register offset values */
#define PRE1AMSW 0x0E
#define LATCH1ALSW 0x0C
#define LATCH1AMSW 0x0E

void main()
{
    SYSTEM system;
    ECODE ecode;
    HBD hbd = 0;
    WORD ldata, hdata; /* unsigned short (16bits) */
    int enc_count; /* where encoder data is stored */
    int start_count; /* remember beginning count */

    printf("encoder test \n");
    S626_Init(&system);
```
/* Module1: counter initialization */
S626_RegWrite(hbd,PRE1ALSW,0x0000);
S626_RegWrite(hbd,PRE1AMSW,0x0080);
S626_RegWrite(hbd,CRA1,0x016f);
S626_RegWrite(hbd,CRA1,0x0360);
S626_RegWrite(hbd,CRB1,0x0C50);

/* Module2: read starting encoder position */
/* read in low 16 bits into variable ldata */
S626_RegRead(hbd, LATCH1ALSW, &ldata);
/* read in bits 23 to 16 into variable hdata */
S626_RegRead(hbd, LATCH1AMSW, &hdata);
/*left shift hdata by 16 bits */
enc_count = ((int) hdata) << 16;
start_count = enc_count | ((int) ldata); /* get starting position reading*/

while(!kbhit()) {

 /*Module3: read the current position of motor shaft */
S626_RegRead(hbd, LATCH1ALSW, &ldata);
S626_RegRead(hbd, LATCH1AMSW, &hdata);
enc_count = ((int) hdata) << 16;
enc_count = enc_count | ((int) ldata);
/* subtract start position reading in order to start from 0 since a relative encoder is used */
enc_count -= start_count;
putchar('*
');
printf("%10d", enc_count);
}

S626_Close();
} /* end of main */

Program details:

The “#define” statements assign the appropriate memory addresses to the register. Each register has a hexadecimal address value specified as an offset value from the base address of the I/O board. The offset values for various registers are given in the Sensoray user manual. For example, register CRB1 has an offset of 0x02. If the base address of the card is 0x5000, then the actual address of the register variable is 0x5002.

S626_RegRead: Reads values from registers and stores the values in a variable.
S626_RegWrite: Writes a specified value to a register.

A pulse from the encoder causes a 24-bit register to be incremented by one. However, only 16 bits can be read from the counter registers at a time, so we first read the low 16 bits into a variable ldata and then read the remaining 8 bits into the variable hdata. For example, if the encoder reading is 0x32A3B1, ldata contains “0xA3B1” and hdata contains” 0x0032”. The two values are then merged to form a 24-bit word. (The operation is similar to the bit-merge program you wrote in Lab 3).
6.2.5 Drift Removal

After an entire motor encoder reading sequence is generated and recorded in an array, you need to call the function `high_pass_filter` in your program to filter out the low-frequency drift component from the measurement data. Then you can write a program to detect the peak-to-peak values using the processed data. This function will also automatically generate a MATLAB program named `enc_plot.m` that will plot the original encoder reading signal and its high-pass filtered counterpart. These plots can be used to verify the results from peak-to-peak detection.

The source code file for this function is `high_pass_filter.c`. It can be downloaded from the course web site. The description of the function `high_pass_filter` is given below:

```c
void high_pass_filter (int *original_data, double *processed_data, int element_number);
```

`original_data` is the pointer to the array that saves the original measurement data.

`processed_data` is the pointer to the array that saves the processed measurement data after drift is removed. It has the same number of elements as the `original_data` array. It is assumed that this array is created in the main program.

`element_number` is the number of elements in `original_data` array.

For a detailed explanation of this preprocessing and why it is necessary, please refer to the appendix at the end of this handout.

6.3. PRELAB ASSIGNMENT

1. Study this handout and related lecture material. In Bolton, read the following:

   • Section 7.5 DC Motors, pp. 168-176.
   • Section 9.3.2 System Models: D.C. Motor (pp. 214-217).
   • Chapter 12 Frequency Response (pp. 262-277).

2. Write a program that produces a sinusoidal voltage output at DAC0 at a user specified frequency and amplitude. You will use the output of this program as an input to the servomotor. This way we can observe the response of the servomotor system to different input (drive) frequencies and amplitudes.

3. Write a program that reads the peak-to-peak value of the servomotor shaft oscillation. The program must be fully documented with meaningful comments. Use a sampling frequency of 1000 Hz while collecting data from the encoder. (Hint: Identify the maximum and minimum value of the encoder signal over a number of cycles, for example 20. Then identify the average peak-to-peak displacement value.)
4. In the lab you will combine the above two programs inside a single loop such that the program oscillates the shaft and collects encoder data simultaneously.

6.4. LAB PROCEDURE

In this lab we will determine the frequency response of the plant by applying a sinusoidal input voltage and recording the amplitude of the motor shaft position as a function of frequency.

First, connect the DAC0 channel to the input of the servomotor. In your program (tested with the oscilloscope first) set the drive frequency to be equal to 5Hz and set the amplitude of the driving signal to be 0.5V amplitude. Execute your program. Your program should have registered the peak-to-peak value of shaft oscillations. Write down this value. Repeat the above procedure for the different drive frequencies listed in the table at the end of the lab handout.

Repeat the experiments for 1V amplitude sine wave input.

6.5. POSTLAB ASSIGNMENT AND LAB REPORT

1. Create bode plots (magnitude only, log-log plots) of the open loop frequency response of the plant. Use the data obtained from the experiments with 0.5V and 1V sine waves. Plot both plots on one graph for comparison. Discuss the difference and similarity.

2. Based on the frequency response shown by the bode plots, estimate the open loop transfer function of the plant. Assume the transfer function has the form illustrated in Figure 5. Your plots should be similar to this figure. The gain $K_m$ is the value of the y-ordinate where the experimental curve crosses the $\omega = 1$ rad/s line. The frequency $\omega_m$ is the point where the plot transitions from a $-20$ dB/decade line to a $-40$ dB/decade line.

Then $T_m = \frac{1}{\omega_m}$. Obtain transfer functions for both 0.5 V input and 1.0 V input. Compare the two transfer functions.

Lab report requirements

1. Answers to the two above questions.

2. Hardcopy of your commented C Program.

3. Complete record of measurement data.
Experimental results of the open loop plant frequency response for 0.5 V sinusoidal input.

<table>
<thead>
<tr>
<th>Frequency (Hz)</th>
<th>Motor Shaft Displacement (peak-to-peak counts)</th>
<th>Magnitude (20*log(output/input))</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>5</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>10</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>15</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>20</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>25</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>30</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>35</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>40</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>45</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>50</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>55</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>60</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>65</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>70</td>
<td>__________________</td>
<td>__________________</td>
</tr>
<tr>
<td>75</td>
<td>__________________</td>
<td>__________________</td>
</tr>
</tbody>
</table>

*output = peak-to-peak shaft oscillation in encoder counts, input = peak-to-peak voltage
Experimental results of the open loop plant frequency response for 1.0 V sinusoidal input.

<table>
<thead>
<tr>
<th>Frequency (Hz)</th>
<th>Motor Shaft Displacement (peak-to-peak counts)</th>
<th>Magnitude (20log(output/input))</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>5</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>10</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>15</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>20</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>25</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>30</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>35</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>40</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>45</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>50</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>55</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>60</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>65</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>70</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
<tr>
<td>75</td>
<td>__________________</td>
<td>______________________________</td>
</tr>
</tbody>
</table>

*output = peak-to-peak shaft oscillation in encoder counts, input = peak-to-peak voltage.
Appendix A.6: Removing Drift Component from Encoder Reading by High-Pass Filtering

A.6.1. Introduction

In this lab the motor is running in an open-loop mode. Under a sinusoidal signal input, a perfectly linear system will give a sinusoidal output if it is stable. However, the DC motor system used is marginally stable (refer to Figure 4). Since the actuation signal typically has a small DC component, a low-frequency drift component is usually observed in the experiments, as shown in Figure A.6.1. (Depending on the input signal, the curve can also be ascending.)

![Low-frequency drift component under a large sinusoidal actuation](image1)

**Figure A.6.1. Low-frequency drift component under a large sinusoidal actuation**

Such drifts can make accurate detections of peaks difficult when the sinusoidal signal is small compared with the drift component, as shown in Figure A.6.2.

![Low-frequency drift component under a small sinusoidal actuation](image2)
In the function `high_pass_filter`, high-pass filtering technique is used to remove the low-frequency drift component so that the processed data will be in a form similar to the one shown in Figure A.6.3.

### A.6.2. High-Pass filter design and implementation

There are many different ways to design a high-pass filter. Here a simple design method is used. First, an analog high-pass filter is designed based on input signal. Then it is discretized using a bilinear transformation. Here the following first order high-pass filter is chosen.

\[
H(s) = \frac{s}{s + a}
\]  
(A1)

Its Bode plot when \( a = 0.1 \) is shown in Figure A.6.4.

By using the bilinear transformation,

\[
s \leftarrow \frac{2}{T} \frac{z - 1}{z + 1}
\]  
(A2)

we get the following discrete transfer function for the filter.

\[
H(z) = \frac{(z-1)/(aT/2 + 1)}{z + (aT/2 - 1)/(aT/2 + 1)}
\]  
(A3)
We can then write the filter equation in an iteration form:

\[
y(k+1) = \frac{1-aT/2}{1+aT/2} y(k) + \frac{1}{1+aT/2} (x(k+1) - x(k)) \quad (A4)
\]

The implementation of this filter is straightforward and is given in `high_pass_filter.c`. 

Figure A.6.4. Bode plot for \( H(s) = \frac{s}{s + 0.1} \)