Development of an Experimental Transfer Function of a Passive Bandpass Filter Using LabView

MEE5318: Instrumentation and Measurement

Prepared for: Dr. Hector Gutierrez, Ph.D

Prepared by: Braidan Duffy, Mary Walker

Prepared on: March 31, 2022

# 1.0 Introduction

Project 2 covers the generation of an experimental transfer function using MATLAB, a myDAQ instrument, and a passive bandpass filter circuit. This project gives students a fundamental understanding of a system’s frequency response and how to generate a model of the response based on experimental data.

# 2.0 Hardware Overview

The myDAQ unit is connected to the bandpass filter circuit using three ports: Analog Out (AO), Analog In 0 (AI0), and Analog In 1 (AI1). The ports AGND, AI0-, and AI1- are connected to ground as shown in Figure 1. It is extremely important that the myDAQ unit and the bandpass circuit are properly grounded together, otherwise electrical noise will interrupt an otherwise clean frequency response, creating useless data. This was an unfortunate issue that plagued early development of this project.

A picture containing text, night sky

Description automatically generated

Figure : Circuit diagram of the passive bandpass filter connected to the myDAQ device.

The bandpass filter requires four passive components to operate correctly: two resistors and two capacitors arranged in two RC circuits according to Figure 1. The values of the resistors and capacitors form a relationship that dictates the filter’s cutoff frequency (Equation 3.1-3.2) [1]. For this project, a high-pass cutoff frequency, , was chosen as 100 Hz, and a low-pass cutoff frequency, , was chosen as 1000 Hz. These values are arbitrary and, for this project, can range from 50-300 Hz and 1-5 kHz, respectively. Due to hardware constraints, the capacitors were chosen first, then the required resistor was determined using Equation 3.2-3.2. These values are summarized in Figure 1.

## 2.1 Circuit Validation

Before the LabView virtual instrument (VI) could be started, the circuit and myDAQ device had to be validated using the National Instruments ELVISmx Bode Analyzer [2]. This tool plots the frequency response of the device onto a Bode diagram and can be tuned according to the start frequency stop frequency, and steps per decade (factor of ten). When run, this tool generated the following figure for the bandpass filter system. When compared to the Bode plots show in [1], it can be surmised that the bandpass circuit and myDAQ are operating as intended and the LabView development can begin.

![Graphical user interface

Description automatically generated]()

Figure : Bode plot as generated by the NI ELVISmx Bode Analyzer tool.

# 3.0 LabView Development

The goal of the developed LabView VI for this project is to emulate the NI ELVISmx Bode Analyzer tool. This VI must have three plots that show the input/out voltages, the gain versus frequency, and phase versus frequency. It should also have some of the frills of the Bode Analyzer such as a start/stop frequency, number of samples to capture per frequency, number of frequencies to trial, and the peak amplitude of the generated waveform.

## 3.2 Bode Plots

A screenshot of a computer

Description automatically generated with low confidence

Figure : Experimental VI Bode plot

# 4.0 Data Processing

With the capture of the system’s frequency response, the data can be processed to determine the experimental transfer function that will theoretically model the passive bandpass filter. This requires a lot of calculations using linear algebra, so a mathematical scripting language like MATLAB is preferred.

To begin, the data captured from the LabView VI is imported into the MATLAB workspace as a working table. Then, the phase data is copied into a new column and converted from degrees into radians for the linear algebra calculations. The magnitude/phase data is converted into input/output data for the MATLAB functions to process into a coherent model. This is done by bringing the magnitude/phase data for the input and output signal into the complex space with the following lines:

u = inputmag\*exp(j\*inputphase);

y = outputmag\*exp(j\*outputphase);

Then, the iddata function can be used to encapsulate this input output data into a time domain object that MATLAB’s etfe function can manipulate into an experimental transfer function and print it to the console. The frequency response of this function can then be plotted using bode (Figure 4) and compared to the captured response of the VI. The full script for this process can be found in 6.3 Data Processing MATLAB Code.

Chart, line chart

Description automatically generated

Figure : Bode diagram generated from the MATLAB script using the Input and Output characteristics of the circuit recorded from the myDAQ.

# 5.0 Model Validation

Chart

Description automatically generated

Figure : Bode plots generated with a MATLAB script to compare the performance of theory, experimental VI, and NI-ELVIS models

# 6.0 Appendix

## 6.1 References

[1] <https://www.electronics-tutorials.ws/filter/filter_4.html>

[2] <https://www.ni.com/en-us/support/downloads/drivers/download.ni-elvismx.html#305452>

## 6.2 Frequency Response LabView Code

Diagram, schematic

Description automatically generated

## 

## 6.3 Data Processing MATLAB Code

% MEE5318 - Project 2 Frequency Response Data Processing

% Braidan Duffy, Mary Walker

% March 31, 2022

clc; clear all; close all;

%% Experimental FRF from VI

% Import magnitude and phase data generated from LabView VI

opts = delimitedTextImportOptions("NumVariables", 6);

% Specify range and delimiter

opts.DataLines = [2, inf];

opts.Delimiter = ",";

% Specify column names and types

opts.VariableNames = ["Frequency", "InputGain", "InputPhase", "x", "OutputGain", "OutputPhase"];

opts.VariableTypes = ["double", "double", "double", "double", "double", "double"];

% Specify file level properties

opts.ExtraColumnsRule = "ignore";

opts.EmptyLineRule = "read";

% Import the data

data = readtable("data/data.csv", opts);

Ts = 0; % Sampling time

W = data.Frequency; % Frequency vector

% Convert magnitude/phase data to input/output data

u = data.InputGain.\*exp(1j\*data.InputPhase);

y = data.OutputGain.\*exp(1j\*data.OutputPhase);

measured\_gain = log(data.OutputGain ./ data.InputGain);

measured\_phase = wrapTo180(data.OutputPhase - data.InputPhase);

data\_func = iddata(y,u,Ts, 'Frequency',W);

G\_exp = etfe(data\_func); % Generate experimental transfer function from data

[mag\_exp, phase\_exp, w\_exp] = bode(G\_exp); % Plot the frequency response of the model

mag\_exp = 20\*log10(mag\_exp);

%% Theoretical FRF

C1 = 27e-9;

C2 = 100e-9;

R1 = 56e3;

R2 = 1.5e3;

G\_theo = tf([C1\*R1 0],[C1\*R1\*C2\*R2 (C1\*R1+C2\*R2) 1]);

[mag\_theo, phase\_theo, w\_theo] = bode(G\_theo);

mag\_theo = 20\*log10(mag\_theo);

%% Experimental FRF from ELVIS

opts = delimitedTextImportOptions("NumVariables", 4);

% Specify range and delimiter

opts.DataLines = [4, inf];

opts.Delimiter = " ";

% Specify column names and types

opts.VariableNames = ["Frequency", "Magnitude", "Phase", "Var4"];

opts.SelectedVariableNames = ["Frequency", "Magnitude", "Phase"];

opts.VariableTypes = ["double", "double", "double", "string"];

% Specify file level properties

opts.ExtraColumnsRule = "ignore";

opts.EmptyLineRule = "read";

opts.ConsecutiveDelimitersRule = "join";

opts.LeadingDelimitersRule = "ignore";

% Specify variable properties

opts = setvaropts(opts, "Var4", "WhitespaceRule", "preserve");

opts = setvaropts(opts, "Var4", "EmptyFieldRule", "auto");

% Import the data

ElvisData = readtable("data/BodeAnalyzerLog.txt", opts);

%% Display

figure

subplot(2,1,1)

semilogx(w\_theo(:,:), mag\_theo(:,:))

hold on

semilogx(w\_exp(:,:), mag\_exp(:,:))

semilogx(ElvisData.Frequency, ElvisData.Magnitude)

grid on

hold off

title("Magnitude")

xlabel("Frequency (Hz)")

ylabel("Magnitude (dB)")

legend("Circuit theory model", "Experimental VI data", "NI-ELVIS data")

subplot(2,1,2)

semilogx(w\_theo(:,:), wrapTo180(phase\_theo(:,:)))

hold on

semilogx(w\_exp(:,:), phase\_exp(:,:))

semilogx(ElvisData.Frequency, ElvisData.Phase)

grid on

hold off

title("Phase")

xlabel("Frequency (Hz)")

ylabel("Phase (Deg)")

legend("Circuit theory model", "Experimental VI data", "NI-ELVIS data")