Parker Brown
A11961591
MAE 144

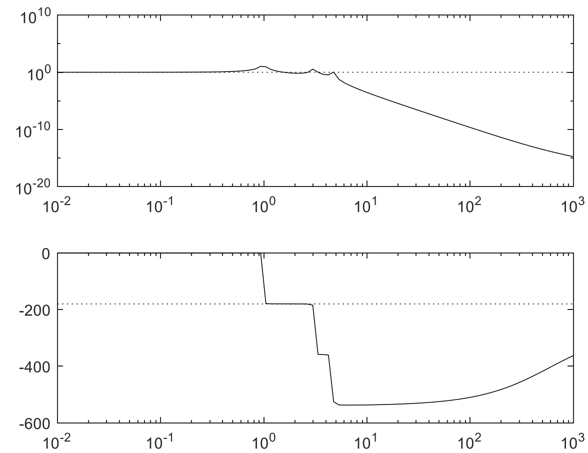
Homework 1

Part 1

Problem 18.6: Bode Plots

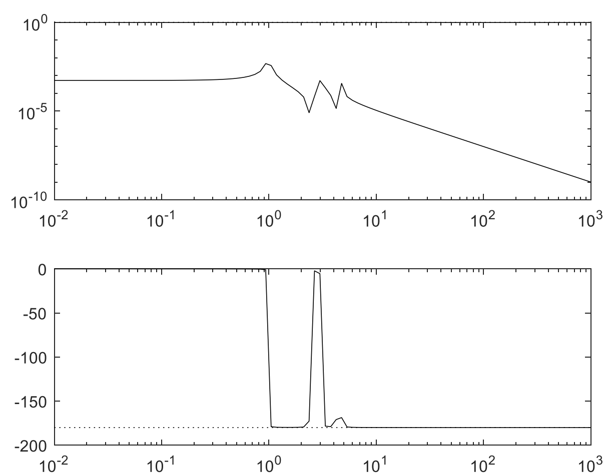
A. Bode Plot for Earthquake Excitation

$$T_A(s) = \frac{1e\text{-}06\ s^3 + 0.001824\ s^2 + 1.071\ s + 201.4}{s^6 + 0.05\ s^5 + 32.36\ s^4 + 0.7688\ s^3 + 237.9\ s^2 + 1.071\ s + 201.4}$$



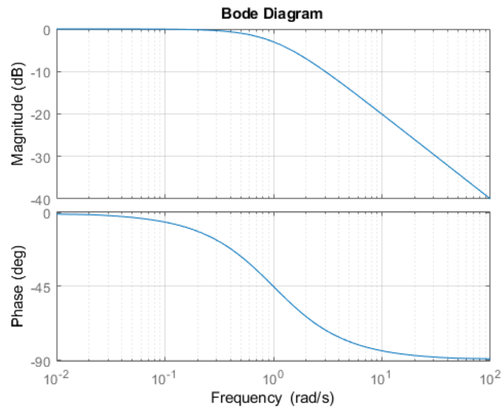
B. Bode Plot for Wind Excitation

$$T_B(s) = \frac{0.001\ s^4 + 4e\text{-}05\ s^3 + 0.02432\ s^2 + 0.0003648\ s + 0.1071}{s^6 + 0.05\ s^5 + 32.36\ s^4 + 0.7688\ s^3 + 237.9\ s^2 + 1.071\ s + 201.4}$$

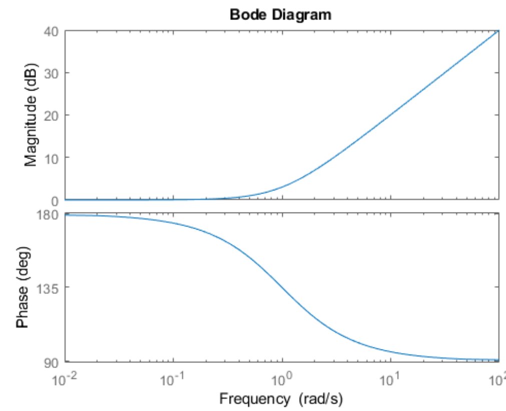


Problem 18.11

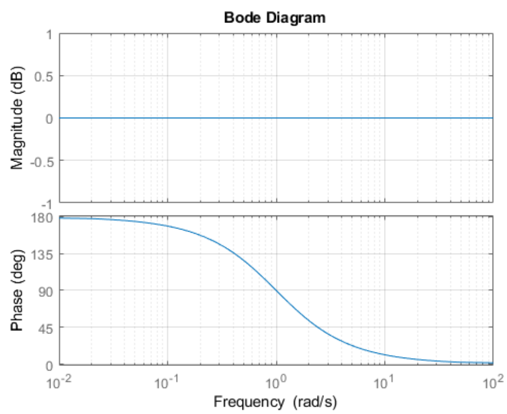
A. $G_1(s) = \frac{1}{s+1}$



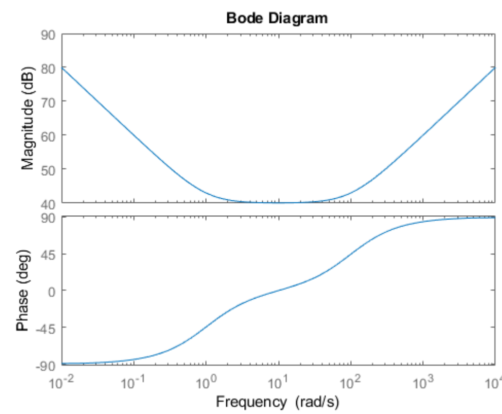
B. $G_2(s) = \frac{s-1}{1}$



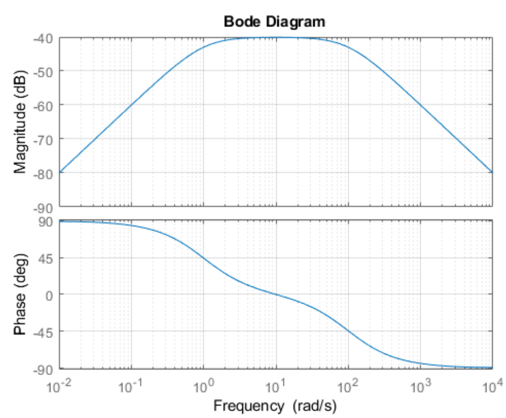
C. $G_3(s) = \frac{s-1}{s+1}$



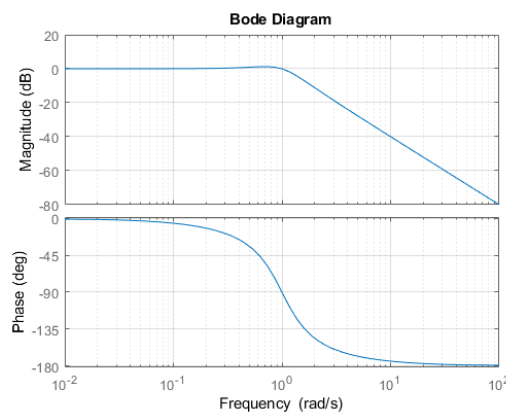
D. $G_4(s) = \frac{s^2+101s+100}{s}$



E. $G_5(s) = \frac{s}{s^{s}+101s+100}$



F. $G_6(s) = \frac{1}{s^2+s+1}$



Part 3

Code: hw1_config.h

```
/*******************************************************************************
* hw1_config.h
*
* Contains the settings for configuration of hw1.c
*******************************************************************************/

#ifndef HW1_CONFIG
#define HW1_CONFIG

// Set constants
#define		GEAR_RATIO		35.555		// Motor gear ratio
#define		ENCODER_RES		60		// Encoder resolution
#define		MOTOR_CHANNEL_L		3		// Left motor channel
#define		MOTOR_CHANNEL_R		2		// Right motor channel
#define		MOTOR_POLARITY_L	1		// Left motor polarity
#define		MOTOR_POLARITY_R	-1		// Right motor polarity
#define		ENCODER_CHANNEL_L	3		// Left encoder channel
#define		ENCODER_CHANNEL_R	2		// Right encoder channel
#define		ENCODER_POLARITY_L	1		// Left encoder polarity
#define		ENCODER_POLARITY_R	-1		// Right encoder polarity

#define		D_GAIN		0.9		// Proportional loop gain
#define		SETPOINT_ZERO		0		// Zero set point for stationary wheel

#define		SAMPLE_RATE_HZ		100	// Loop rate
#define		DT		0.01	// 1/SAMPLE_RATE_HZ

#endif	//HW1_CONFIG
```

Code: hw1.c

```
/*******************************************************************************
* File: hw1.c
* Author: Parker Brown
* Date: 11/15/2017
* Course: MAE 144, Fall 2017
* Description: Program runs closed feedback loop, minimizing angle error between
* left and right wheel of MIP. User spins right wheel, while left wheel tracks
* right wheel angle.
*******************************************************************************/

// usefulincludes is a collection of common system includes for the lazy
// This is not necessary for roboticscape projects but here for convenience
// Nice to have for TWO_PI
#include <rc_usefulincludes.h>
// main roboticscape API header
#include <roboticscape.h>
#include "hw1_config.h"
```

```
// function declarations
void on_pause_pressed(); // do stuff when paused button is pressed
void on_pause_released(); // do stuff when paused button is released

/*******************************************************************************
* int main()
*
* hw1 main function contains these critical components
* - call to rc_initialize() at the beginning
* - main while loop that checks for EXITING condition
*			- Run feedback loop while RUNNING or wait while PAUSED
* - rc_cleanup() at the end
*******************************************************************************/
int main(){

	// initialize hardware first
	if(rc_initialize()){
		fprintf(stderr,"ERROR: failed to initialize rc_initialize(), are you root?\n");
		return -1;
	}

	// initialize stuff here
	rc_set_pause_pressed_func(&on_pause_pressed);
	rc_set_pause_released_func(&on_pause_released);
	rc_enable_motors();

	// done initializing so set state to RUNNING
	rc_set_state(RUNNING);

	// Initialize variables used in the while loop
	int sleep_time=DT*1e6; // Sleep time to set rough loop rate
	float wheelAngleL=0, wheelAngleR=0; // Initialize wheel angles to zero
	float dutyL=0; // Initialize duty cycle to zero
	float errorL=0; // Initialize loop error to zero

	// Print header for standard output
	printf("Loop Gain: %3.1f\n", D_GAIN);
	printf("Wheel Angle Phi (Rad)\n");
	printf("  Phi_L  |");
	printf("  Phi_R  |");
	printf(" Error_L |");
	printf("  Duty_L |");
	printf(" \n");

 // Keep looping until state changes to EXITING
	while(rc_get_state()!=EXITING){
		// If RUNNING, run feedback loop
		if(rc_get_state()==RUNNING){
			rc_set_led(GREEN, ON); // GREEN when on
			rc_set_led(RED, OFF); // RED when paused

			// Get wheel angles by reading encoder channels with math
			// Math says (wheel angle) = 2pi * (enc position) / (enc count per rev)
```

```
			wheelAngleL = ((rc_get_encoder_pos(ENCODER_CHANNEL_L) * TWO_PI) \
						/ (ENCODER_POLARITY_L * GEAR_RATIO * ENCODER_RES));
			wheelAngleR = ((rc_get_encoder_pos(ENCODER_CHANNEL_R) * TWO_PI) \
						/ (ENCODER_POLARITY_R * GEAR_RATIO * ENCODER_RES));

			errorL = wheelAngleR - wheelAngleL; // Error between free and driven wheel
			// errorL = SETPOINT_ZERO - wheelAngleL; // Error with zero setpoint
			dutyL = D_GAIN * errorL; // Controller output == left wheel duty cycle

			// Check for motor saturation
			if(dutyL > 1.0){
				dutyL = 1.0;
			}
			else if(dutyL < -1.0){
				dutyL = -1.0;
			}

			rc_set_motor(MOTOR_CHANNEL_L, MOTOR_POLARITY_L * dutyL); // drive left wheel

			// Print wheel angles, angle error, and contoller output duty cycle
			printf("\r"); // carriage return because it looks pretty
			printf("%8.3f |", wheelAngleL);
			printf("%8.3f |", wheelAngleR);
			printf("%8.3f |", errorL);
			printf("%8.3f |", dutyL);
			fflush(stdout);

		}
		else if(rc_get_state()==PAUSED){
			// Set everything to an off state when paused
			rc_set_led(GREEN, OFF); // GREEN when on
			rc_set_led(RED, ON); // RED when paused
			rc_set_motor_free_spin_all(); // Set motors to free spin while paused
			rc_set_encoder_pos(MOTOR_CHANNEL_L, 0); // Reset left encoder position
			rc_set_encoder_pos(MOTOR_CHANNEL_R, 0); // Reset right encoder position

		}

		usleep(sleep_time); // Sleep for DT in microseconds
	}

	// exit cleanly
	rc_cleanup();
	return 0;
}

/*******************************************************************************
* void on_pause_released()
*
* Make the Pause button toggle between paused and running states.
*******************************************************************************/
void on_pause_released(){
	// toggle betewen paused and running modes
```

```
	if(rc_get_state()==RUNNING)			rc_set_state(PAUSED);
	else if(rc_get_state()==PAUSED)	rc_set_state(RUNNING);
	return;
}

/*******************************************************************************
* void on_pause_pressed()
*
* If the user holds the pause button for 2 seconds, set state to exiting which
* triggers the rest of the program to exit cleanly.
*******************************************************************************/
void on_pause_pressed(){
	int i=0;
	const int samples = 100;	// check for release 100 times in this period
	const int us_wait = 2000000; // 2 seconds

	// now keep checking to see if the button is still held down
	for(i=0;i<samples;i++){
		rc_usleep(us_wait/samples);
		if(rc_get_pause_button() == RELEASED) return;
	}
	printf("long press detected, shutting down\n");
	rc_set_state(EXITING);
	return;
}
```

Code: Makefile

```
# This is a general use makefile for robotics cape projects written in C.
# Just change the target name to match your main source code filename.
TARGET = hw1

CC		:= gcc
LINKER		:= gcc -o
CFLAGS		:= -c -Wall -g
LFLAGS		:= -lm -lrt -lpthread -lroboticscape

SOURCES			:= $(wildcard *.c)
INCLUDES	:= $(wildcard *.h)
OBJECTS		:= $(SOURCES:$%.c=$%.o)

prefix		:= /usr/local
RM		:= rm -f
INSTALL		:= install -m 4755
INSTALLDIR	:= install -d -m 755

LINK		:= ln -s -f
LINKDIR		:= /etc/roboticscape
LINKNAME	:= link_to_startup_program


# linking Objects
$(TARGET): $(OBJECTS)
```

```
	@$(LINKER) $(@) $(OBJECTS) $(LFLAGS)


# compiling command
$(OBJECTS): %.o : %.c $(INCLUDES)
	@$(CC) $(CFLAGS) -c $< -o $(@)
	@echo "Compiled: "$<

all:
	$(TARGET)

debug:
	$(MAKE) $(MAKEFILE) DEBUGFLAG="-g -D DEBUG"
	@echo " "
	@echo "$(TARGET) Make Debug Complete"
	@echo " "

install:
	@$(MAKE) --no-print-directory
	@$(INSTALLDIR) $(DESTDIR)$(prefix)/bin
	@$(INSTALL) $(TARGET) $(DESTDIR)$(prefix)/bin
	@echo "$(TARGET) Install Complete"

clean:
	@$(RM) $(OBJECTS)
	@$(RM) $(TARGET)
	@echo "$(TARGET) Clean Complete"

uninstall:
	@$(RM) $(DESTDIR)$(prefix)/bin/$(TARGET)
	@echo "$(TARGET) Uninstall Complete"

runonboot:
	@$(MAKE) install --no-print-directory
	@$(LINK) $(DESTDIR)$(prefix)/bin/$(TARGET) $(LINKDIR)/$(LINKNAME)
	@echo "$(TARGET) Set to Run on Boot"
```

Code: README.txt

```
/******************************************************************************
* Files: hw1.c, hw1_config.h, Makefile
* Author: Parker Brown
* Date: 11/15/2017
* Course: MAE 144, Fall 2017
* Description: Program runs closed feedback loop, minimizing angle error between
* left and right wheel of MIP. User spins right wheel, while left wheel tracks
* right wheel angle.
******************************************************************************/
```