Guidelines:
1. Work with your assigned teams to complete tasks 1 through 3 in class today
2. Complete tasks 4-6 on your own (with appropriate help from your teammates)
3. Each student will turn in their own work
4. The objective is to design and simulate a speed controller for your unloaded DC motor

Project Description:

Next week we will be controlling your unloaded DC motor. To set this up, we need to design a control strategy and verify that it works in simulation.

Tasks:
1. Finalize your transfer function for your DC motor
   a. Recall that the transfer function for the DC motor is:
   $$\frac{\Omega(s)}{E_{in}(s)} = \frac{K}{(Js+b)(L_a s + R_a) + K K_b}$$
   b. Record your best values for the transfer function parameters in the table below:

Parameter (Units)	Typical Value	Best Source	Your Value
R_a (Ω)	9.2	Project 1 – Task 2a	
L_a (H)	$0.21(10^{-3})$	Project 1 – Task 2c	
K $\left(\frac{N\,m}{A}\right)$	0.014	Project 1 – Task 5c	
K_b $\left(\frac{V\,s}{rad}\right)$	0.014	Project 1 – Task 5c	
b $(N\,m\,s)$	$2.5(10^{-7})$	Project 2 – Task 2c	
J $(kg\,m^2)$	$1(10^{-6}) \rightarrow 3(10^{-6})$	Project 3 – Task 4c	

   c. Write the transfer function with your numbers below.

2. Explore the open-loop control strategy
   a. For open-loop control, there is no feedback loop. The transfer function of the plant $\left(G = \frac{Y}{U}\right)$ is the same as the transfer function for the system $\left(T = \frac{Y}{R}\right)$ because the reference input (R) equals the plant input (U).

b. With this control strategy, use the final value theorem to find the final speed for a unit step voltage input on the transfer function.

c. You could run this model at any desired speed just by changing the input voltage. What voltage would you supply so that the mathematical model predicts a motor speed of 500 rad/s (4775 rpm)?

d. Explain why this open-loop strategy is not a good idea for a real DC motor where the load might change

3. Consider a proportional controller with a feedback loop
   a. Write a block diagram for a controller that includes a unity feedback path. The microcontroller we used for the last project can be programmed to measure shaft speed from the encoder and compare this to a desired reference input speed. This means the forward path of your block will need to have: a block for the controller (G_c); and a block for the motor plant (G).

b. Using proportional control $(G_c = k)$ calculate the steady-state error of this control scheme for a reference step input of $r(t) = 100\frac{rad}{s}$. This control block converts the input speed error $\left(\frac{rad}{s}\right)$ to an output that is the input voltage to the motor (V), so k needs to have units. Leave e_{ss} as a function of k for now.

c. What gain would be required for this type 0 system to have a 1% error to a step input?

d. Calculate the pole locations of the resulting controlled systems with your gain value (k). Do these pole locations indicate a stable system?

e. Using your calculated gain value (k), what control signal will initially be calculated when the reference input is stepped up to $r(t) = 300\frac{rad}{s}$ while the output is still at rest? *Hint: your answer should just be* $u_{init} = 3k$.

f. If you have correctly calculated parts d) and e), at least one of these answers should indicate why this is not an acceptable control strategy. Describe what the problem is.

4. The next step will be to try a PI controller. You will need some Matlab tools to select the parameters of this controller.
   a. A generic PI controller can be written in two different forms in the Laplace domain, we will use both.
      i. $G_{PI}(s) = k_p + k_i s$
      ii. $G_{PI}(s) = K\frac{s+z_i}{s}$
   b. There are three design criteria for this controller based on a reference step input of $r(t)$ =300 rad/s
      i. Steady-state error to a step input should be: $e_{ss} < 1\%$
      ii. The maximum input voltage to the plant should be: $u_{max} < 10\ V$
      iii. The system should respond as quickly as is practical ($\omega_c = 100\frac{rad}{s}$)
   c. Create a Bode plot of the plant transfer function without the controller
   d. Choose some appropriate values for K and z_i and add the PI controller to the rest of the system for the Bode plot. Play around with values until you meet the desired crossover frequency.
   e. Convert your standard form K and z_i values to parameters of k_p and k_i
   f. Calculate the initial command signal for your chosen values based on the formula $u_{init} = k_p\left(300\frac{rad}{s}\right)$. *The k_i value does not matter because at time zero nothing has been integrated yet.*
   g. Based on what you learned from part e), find some new K and z_i values that will meet the second design criteria (although they may not meet the third). You will need your results from part c) later, so don't delete them.
5. Simulate the response of your designed to control to verify that it works
   a. Use the *feedback* command to find the closed-loop transfer function (T)
   b. Use the *step* command with the structure $[y, t] = step(3 * T)$, which will store the values for a magnitude 3 step command without making a plot
   c. Use the *lsim* command with the structure $y = lsim(Gc, 3 - y, t)$, which will calculate the output of the control block based on the results of the *step* command
   d. Make your own plot of the *step* results and verify that you meet the steady-state error criteria

e. Make your own plot of the *lsim* results and verify that you meet the maximum input voltage criteria at all times (not just at the initial time)
f. Modify control parameters if needed, then add your final Bode plot to your result from task 4c)
g. Graphically demonstrate your final controller by making a root locus plot of the entire forward path (Gc*G*.01). Add the poles of T with plus signs onto this graph using $plot(real(pole(T)), imag(pole(T)), '+', 'MarkerSize', 7)$

6. Submit the following by the start of class on the due date
   a. This worksheet with tasks 1 through 3 filled out
   b. A published version of your Matlab script including
      i. A Bode plot with three lines representing Tasks 4c, 4d, and 5f
      ii. A plot based on Task 5d
      iii. A plot based on Task 5e
      iv. A root locus plot based on Task 5g
      v. For all of these plots, use a logical title (which cannot be the default for the plot), axis labels, and a legend where appropriate
      vi. The final controller transfer function in both standard forms
   c. Also submit your Matlab script as a *.m file to D2L for originality checking