Final Project (ME-190, Fall 2017)

Implementation due date: Last lab (Dec. 5-7)

Report due date: Monday, Dec. 11, 2017, 5:00 pm

Project Objectives:

1- Develop equations of motion (for the MinSeg robot) using the fundamental laws of physics
2- Design, simulate, and implement a controller from the free body diagram to a working system
3- Integrate theory, hardware, and software for a mechatronic system
4- Learn the value of model-based mechatronic design
5- Write a technical report

Part 1. Modeling the MinSeg robot

This section focuses on developing a complete **State-Space** model for the MinSeg robot. Figure 1 shows the electrical circuit as well as the rigid-body diagram of the robot. Using Kirchhoff's voltage law and the rigid body kinematics and kinetics analysis, we can develop a set of differential equations representing the system's electromechanical dynamics.

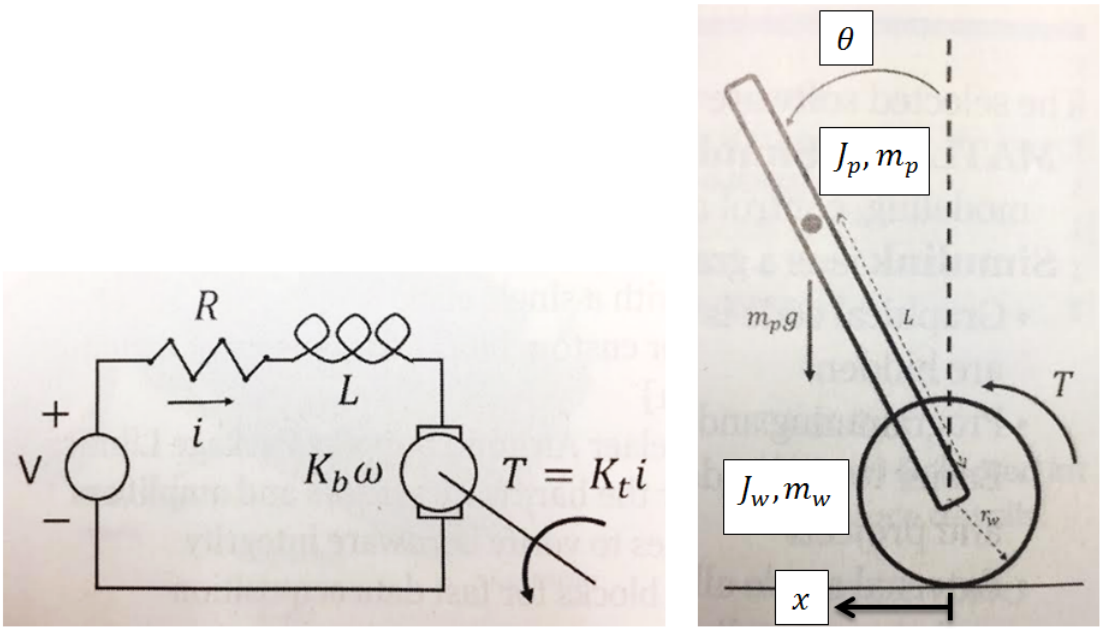


Figure 1. Electro-Mechanical diagrams of the MinSeg robot.

Tasks to be addressed in the report:

1.1. Sketch the free body diagrams of the mechanical sub-systems (the wheel and the body), and apply Newton's law to derive the equations of motion. Carry out the relative acceleration analysis from the rigid-body kinematics to derive the equations of motion for the mechanical system.

Note: Write the equations in terms of the position of the wheel center, x, and the angular deflection of the robot's body, Θ.

1.2. Use Kirchhoff's voltage law and write the differential equation of the electrical circuit.

1.3. Combine the mechanical and electrical equations assuming the coil inductance is zero, i.e., $L_m = 0$.

1.4. Assume the deflection angle Θ will remain small during the robot's control in the upright position. Replace $sin(\Theta)$ with Θ; $cos(\Theta)$ with 1; and $\dot{\Theta}^2$ with 0. This process will convert the nonlinear model to a linear one which is only accurate for small deflection angles. Write the linearized equations in the following matrix form:

$$M\ddot{X} + D\dot{X} + KX = Fu$$

where

$X = \begin{bmatrix} x \\ \Theta \end{bmatrix}$.

M, D, and K are 2×2 matrices representing the system's inertia, damping, and stiffness.

F is a 2×1 input coefficient vector.

u is the system's input (i.e., the applied voltage).

Hint: Your linear equations will look something like these:

$$\left(m_p + m_w + \frac{J_p}{r_w^2}\right)\ddot{x}(t) + (??)\ddot{\theta}(t) + \cdots = (??)V$$

$$(??)\ddot{x}(t) + \left(J_p + m_p L^2\right)\ddot{\theta}(t) + \cdots = (??)V$$

1.5. Show that one can convert the above matrix equation to the state-space form using the following formula:

$$\dot{q}(t) = Aq(t) + Bu(t)$$

where

$$q(t) = \begin{bmatrix} X \\ \dot{X} \end{bmatrix}, \qquad A = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ -M^{-1}K & -M^{-1}D \end{bmatrix}, \qquad B = \begin{bmatrix} \mathbf{0} \\ M^{-1}F \end{bmatrix}$$

where $\boldsymbol{0}$ and $\boldsymbol{I}$ are zero and identity matrices of appropriate dimensions.

Hint: start from $q(t)$ and take its time derivative and … .

1.6. Write a Matlab script that converts the MinSeg robot equations to the state-space form using the above formula for parameter values in the below table:

Parameter	Description	Value	Unit
m_p	Pendulum mass	0.285	Kg
m_w	Wheel mass	0.025	Kg
J_p	Pendulum inertia	0.0010	Kg-m^2
J_w	Wheel inertia	0.0013	Kg-m^2
r_w	Wheel radius	0.022	m
l	Pendulum length	0.1	m
c	Rotational damping	0.0001	N.m.s/rad
k_t	Torque constant	0.3	N.m/A
k_b	Back-emf constant	0.5	V.s/rad
R	Coil resistance	5	Ω
g	Gravitational acceleration	9.81	m/s^2

1.8. Simulate the state-space system in simulink for a small initial angle Θ (e.g., $\Theta = \pi/60$) for zero input voltage. For the output equation use:

$$y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} q(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t)$$

Plot and report the output response, and discuss whether the response makes sense based on the physics of the system and the eigenvalues of the state matrix, *A*.

Part 2. Designing a Full-State Feedback Controller

For a controllable state-space system

$$\dot{q}(t) = Aq(t) + Bu(t)$$

We can stabilize and regulate all the states using a negative full-state feedback control law as follows:

$$u(t) = -Kq(t)$$

where *K* is a vector of control gains. Combining the above two equations lead to:

$$\dot{q}(t) = (A - BK)q(t)$$

It can be shown that if the system is controllable, the eigenvalues (poles) of the closed-loop system matrix (*A*-*BK*) can be placed in arbitrary locations. Matlab's "place" and "acker" functions can provide the feedback gain for the given desired location of poles.

2.1. Use Matlab's "acker" function (representing Ackerman's Method) to place the eigenvalues of the closed-loop system at desirable locations on the left half complex plane (Check Matlab's help). You may choose the desired closed-loop system eigenvalues to be real or complex, repeated or non-repeated.

2.2. Once you found the control gain vector *K*, add a full-state feedback loop to your Simulink model and simulate the response of the system for the initial condition of 10 degrees: $\Theta = 10*\pi/180$ rad. Monitor the input voltage to make sure it doesn't exceed the 5 v maximum value.

2.3. Modify the location of the eigenvalues of the closed-loop system until you arrive at a desirable response without exceeding the maximum voltage level.

In your results include: (1) The location of the eigenvalues of the open-loop system (i.e., A matrix), (2) Location of the eigenvalues of the closed-loop system (A-BK), and (3) The response of the system (q), and control input (u) for a few different initial conditions within 0-10 degrees.

Part 3. Designing a Linear Quadratic Regulator (LQR)

An alternative approach for pole-placement is to use the Linear Quadratic Regulator (LQR). In LQR we use the same control structure: $u(t) = -Kq(t)$, however, we choose the feedback gain vector in a way that minimizes the following cost function:

$$J = \lim_{T\to\infty} \int_{t_0}^{T} (q^T Q\, q + u^T R\, u)\, dt$$

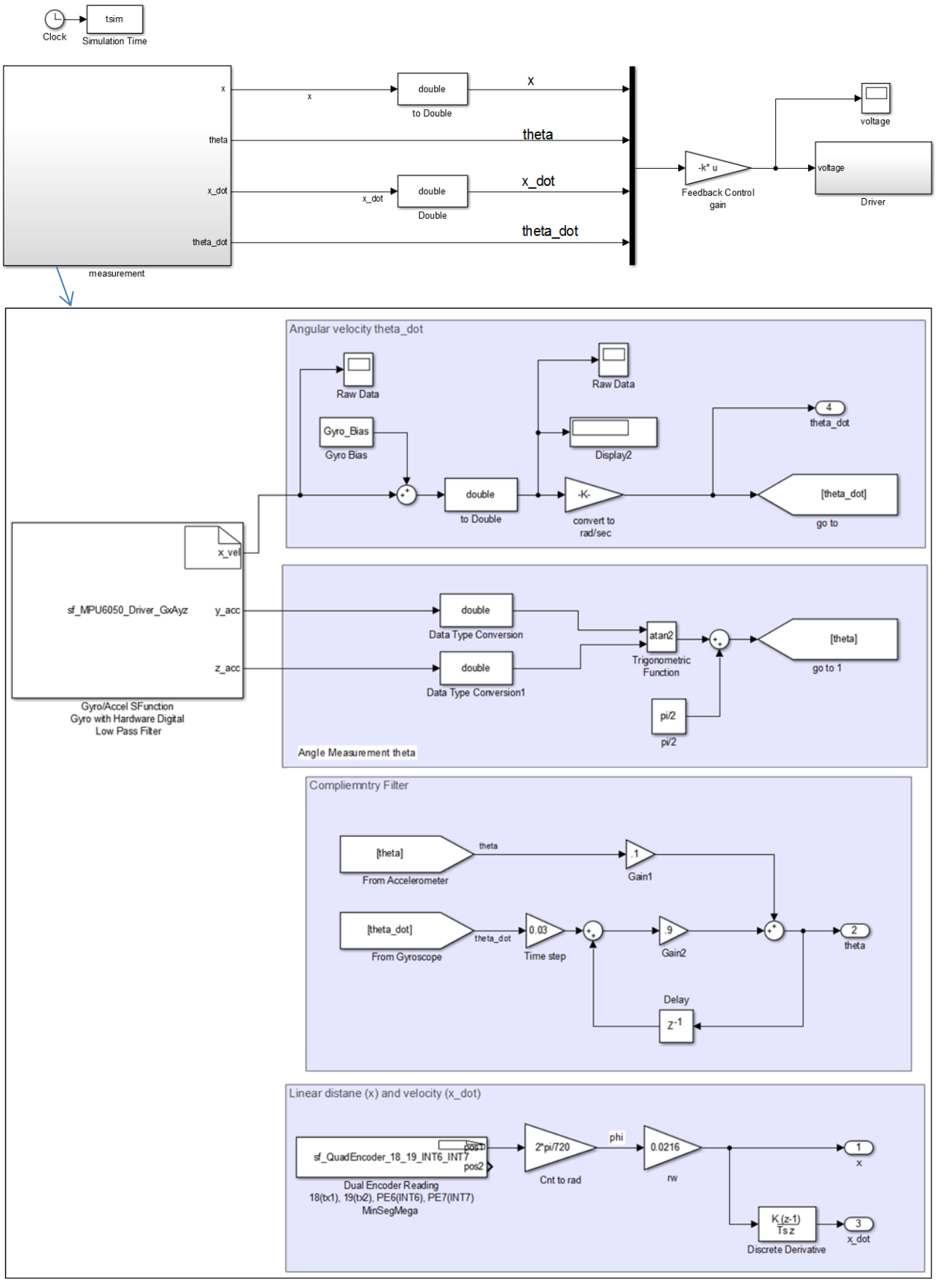
where *Q* and *R* represent weight matrices to adjust performance vs. control effort. If *Q* is chosen much larger than *R* then the system will reach a great performance (fast decay to zero), but the control effort will be large (may cause saturation). On the other hand if *R* is chosen larger than *Q* the system will have slow response but the control effort will be smaller. Note that in our system, *Q* is a 4×4 matrix, and *R* is a scalar.

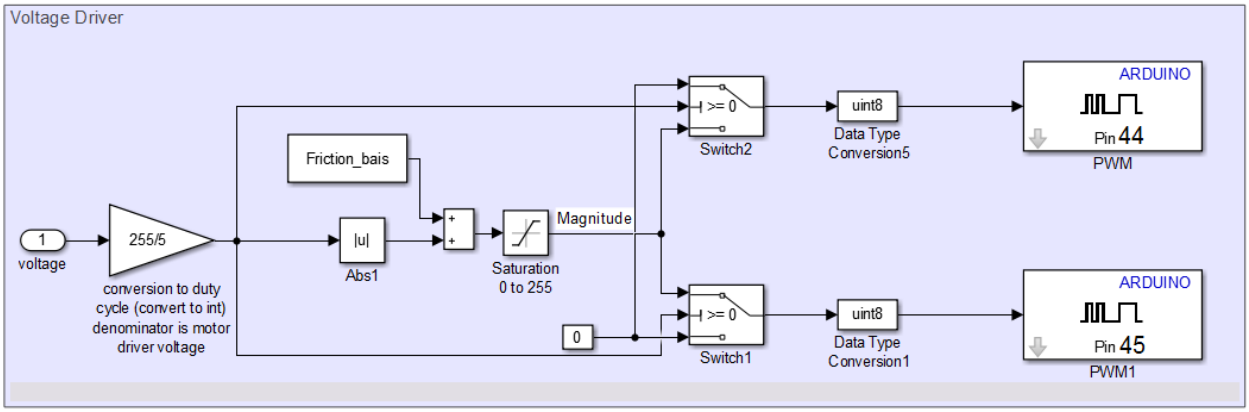
3.1. Assuming matrix *Q* is diagonal, use Matlab's "lqr" function to obtain the feedback gain vector, *K*. Implement the obtained feedback gain in your simulink file, and choose different values for the entries of *Q* and *R* and compare your results. Plot the state trajectories as well as the control input for the initial condition of 10 degrees.

3.2. Find the *Q* and *R* matrices and the corresponding control gain *K* such that a desirable balance between the performance and control effort is obtained. Plot the response of the system (state trajectories and control input) for different initial angles within 0-10 degrees. Plot the eigenvalues of the closed-loop system. Interpret the relationship between the location of the eigenvalues and the closed-loop system response.

Part 4. Implementation the Feedback Controller

To implement the feedback control law, all the states need to be measured and fed back to the robot. The Simulink block diagram of the full-state feedback controller is shown in the figure. The system states $(x, \theta, \dot{x}, \dot{\theta})^T$ can be measured from the encoder, accelerometer, and the gyroscope. The state vector is then formed using a multiplexer and passed through a negative feedback (vector) gain $-k$ to calculate the stabilizing voltage signal. The calculated voltage is then passed to the voltage driver and commanded to the hardware (i.e., DC motor driving circuit).


Clock
tsim
Simulation Time
x
theta
x_dot
theta_dot
measurement
double
to Double
Double
-k* u
Feedback Control
gain
voltage
Driver
Angular velocity theta_dot
Raw Data
Gyro_Bias
Gyro Bias
Display2
-K-
convert to
rad/sec
4
theta_dot
[theta_dot]
go to
x_vel
sf_MPU6050_Driver_GxAyz
y_acc
z_acc
Gyro/Accel SFunction
Gyro with Hardware Digital
Low Pass Filter
Data Type Conversion
Data Type Conversion1
atan2
Trigometronic
Function
[theta]
go to 1
pi/2
Angle Measurement theta
Compliemntry Filter
From Accelerometer
Gain1
From Gyroscope
0.03
Time step
.9
Gain2
2
Delay
Linear distane (x) and velocity (x_dot)
sf_QuadEncoder_18_19_INT6_INT7
pos1
pos2
Dual Encoder Reading
18(tx1), 19(tx2), PE6(INT6), PE7(INT7)
MinSegMega
2*pi/720
Cnt to rad
phi
0.0216
rw
K (z-1)
Tsz
Discrete Derivative
3
x_dot



4.1. Create the depicted Simulink block diagrams, and determine whether the block diagrams would give us the required feedback measurements (states) correctly.

4.2. Determine the gyro bias correction and conversion gains from gyro output to rad/sec. The gain can be calibrated using angle measurement after disconnecting the accelerometer part from the complimentary filter. This value should be around pi/(2*1.15e4). Use this number if you are behind.

4.3. Obtain the friction compensation bias to be added the input voltage. This will help linearize the deadzone caused by friction. This values should be around 50 (on the scale of 255), corresponding to 1 v, which is the least voltage required to move the motor at almost zero velocity.

4.4. Connect your states and input voltage to "To Workspace" block for data collection for the report.

4.5. Go back to your Matlab code for part 1-3. Check whether the eigenvalues of the A matrix are:

0, 8.13, -7.49, -26.6

If not, double check your code and verify your equations have been correctly coded for *M*, *D*, *K*, and *F* matrices and converted to the A and B matrices.

4.6. In the LQR section of your Matlab code, use the following weight matrices:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}, \qquad R = 1$$

According to the LQR cost function, these weight matrices will translate into the following cost function:

$$J = \lim_{T \to \infty} \int_{t_0}^{T} \left(x^2 + 100\theta^2 + \dot{x}^2 + 100\dot{\theta}^2 + u^2 \right) dt$$

which puts more emphasis on θ and $\dot{\theta}$ instead of x and $\dot{x}$.

4.7. Find the optimal feedback gain vector using Matlab's "lqr" function and insert it into the Simulink model. Run the Simulink in external mode for the time step value of 0.03 s. Keep the robot in the upright position and try to secure it by keeping your hands around it.

4.8. Evaluate the performance of the controller. If you are unhappy about the results, go back to the LQR controller and tweak the weights until you find a better set of controller gains.

4.9. Check to see what happens if you use the same weights for x and Θ. Can your controller still be stable? How can you justify the results?

4.9. Using the best control gains you obtained, record the linear and angular positions and velocities and control voltage for a period of time of at least 15 s. If the robot is too stable, hit it lightly with a pen and collect more data to include in your final presentation.

4.10. Try to use the accelerometer sensor only for Θ. Change its gain from 0.1 to 1, and change the gyro gain from 0.9 to 0 and disconnect the delay to prevent accumulation of the gyro signal. What difference do you observe in the behavior of the system? Now try to use gyro sensor only for measuring the angle by disconnecting the accelerometer and changing the gyro gain back 1. What difference do you observe now? Is the complimentary filter the savior of the robot?

4.11. In your report, summarize what you learned through this project, and whether you could connect the dots between the theory and practice. Point out to the steps that were vague or gray to you. Just a piece of food for thought: Think about how else you could have controlled the robot. Could you stabilize the robot without modeling, system identification, and optimal control? The answer is probably yes, with a lot of luck guessing the right values for the control gain vector, K!

Also in your report, list the sources of mismatch between the model and the actual system, and explain how they affected your system response. Provide suggestions that can help mitigate them if you had more time to spend on the project.