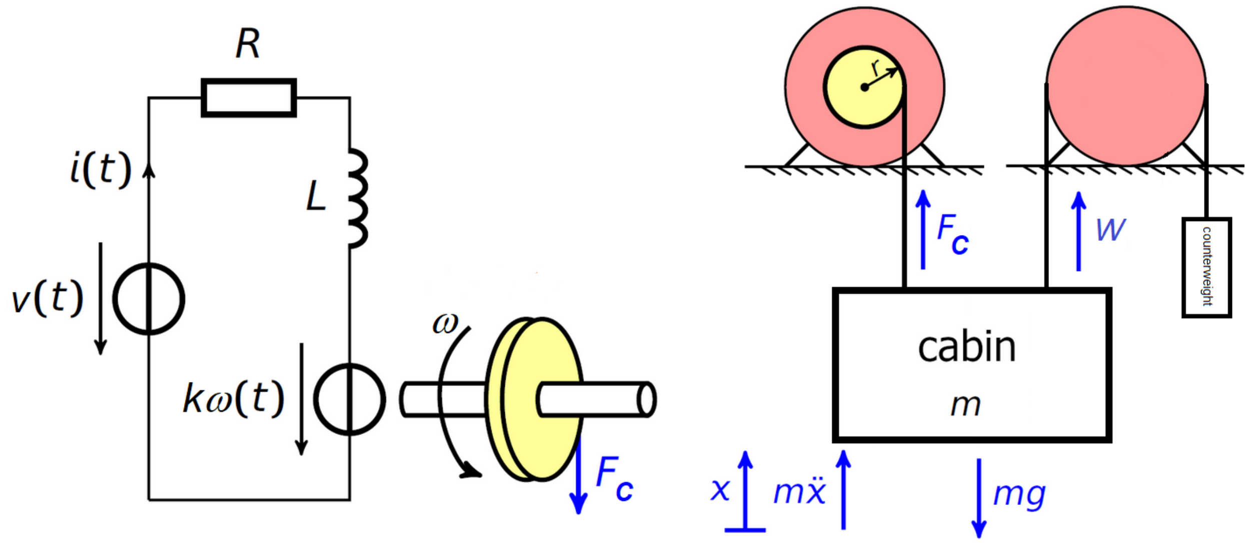
AEM 591/ECE 593/ME 591 Project 1

Elevator System Simulation and Analysis

This project is intended to introduce some of the tools in Python for use in the analysis of dynamical systems. To that end, consider the following dynamical system of an elevator



which has the following governing equations.

For the electromechanical dynamics, one has following two equations. By the sum of moments around the electric motor

$$r(t)F_c(t) = ki(t) + I_m\dot{\omega}(t) \tag{1}$$

or

$$r(t)F_c(t) - ki(t) = I_m\dot{\omega}(t) \tag{2}$$

and by Kirchhoff's voltage law

$$v(t) = R(t)i(t) + L\frac{di(t)}{dt} + k\omega(t) \tag{3}$$

where F_c is the force on the cabin exerted by the cable, r is the radius of the elevator cable from the center of the motor, k is the motor characteristic, i is the current, I_m is the moment of inertia of the motor, ω is the rotational speed in radians, v is the supplied voltage, R is the resistance of the coil winding, and L is the inductance of the motor. Note that in this model, the moment generated by the motor is $ki(t)$ and the electromotive force is equal to $k\omega(t)$.

For the cabin dynamics, one has by the sum of forces

$$F_c(t) + W - mg = m\ddot{x}(t) \tag{4}$$

or

$$F_c(t) = m\ddot{x}(t) + mg - W \tag{5}$$

where m is the mass of the cabin, $\vec{x}$ is the position of the cabin, g is the acceleration due to gravity, and W is the weight of the counterweight.

For simplicity, one can assume that due to no slippage or cable stretching, one has

$$\dot{x}(t) = r(t)\omega(t) \tag{6}$$

or

$$\omega(t) = \frac{\dot{x}(t)}{r(t)} \tag{7}$$

and differentiating, one has

$$\dot{\omega}(t) = \frac{r(t)\ddot{x}(t) - \dot{r}(t)\dot{x}(t)}{r^2(t)} \tag{8}$$

$$\dot{\omega}(t) = \frac{\ddot{x}(t)}{r(t)} - \frac{\dot{r}(t)\dot{x}(t)}{r^2(t)} \tag{9}$$

Furthermore, the radius of the cable from the motor center varies as a function of the cabin position due to the thickness of the cable. This can be approximately modeled by the following equation

$$\dot{r}(t) = \delta_c \frac{\omega(t)}{2\pi} \tag{10}$$

where δ_c is the radial increase due to a full rotation, i.e. $\omega = 2\pi$, of the disk. Note, that by Equation 7, one has

$$\dot{r}(t) = \frac{\delta_c}{2\pi r(t)}\dot{x}(t) \tag{11}$$

In addition, the resistance varies with time as the motor heats up. This can be modeled by

$$R(t) = R_0 + \delta_R(1 - e^{\frac{-t}{\tau_R}}) \tag{12}$$

Note that the heat radiated by the motor is equivalent to the power, Ri^2.

Nonlinear Dynamics Derivation

First, substitute Equation 5 into Equation 2 to obtain

$$r(t)\left(m\ddot{x}(t) + mg - W\right) = ki(t) + I_m\dot{\omega}(t) \tag{13}$$

Next, substituting Equation 9, one has

$$mr(t)\ddot{x}(t) + mgr(t) - Wr(t) = ki(t) + I_m\left(\frac{\ddot{x}(t)}{r(t)} - \frac{\dot{r}(t)\dot{x}(t)}{r^2(t)}\right) \tag{14}$$

Then, substituting Equation 11, one has

$$mr(t)\ddot{x}(t) + mgr - Wr(t) = ki(t) + \frac{I_m \ddot{x}(t)}{r(t)} - \frac{I_m \delta_c \dot{x}^2(t)}{2\pi r^3(t)} \tag{15}$$

which can be rearranged with $\ddot{x}$ terms on the left side as

$$mr(t)\ddot{x}(t) - \frac{I_m}{r(t)}\ddot{x}(t) = Wr(t) - mgr(t) + ki(t) - \frac{I_m \delta_c \dot{x}^2(t)}{2\pi r^3(t)} \tag{16}$$

or

$$\ddot{x}(t) = \frac{1}{mr(t) - \frac{I_m}{r(t)}} \left(Wr(t) - mgr(t) + ki(t) - \frac{I_m \delta_c \dot{x}^2(t)}{2\pi r^3(t)} \right) \tag{17}$$

Second, note that 3 can be rearranged as

$$\frac{di(t)}{dt} = \frac{1}{L}v(t) - \frac{R(t)}{L}i(t) - \frac{k}{L}\omega(t) \tag{18}$$

Then, substituting Equation 7, one has

$$\frac{di(t)}{dt} = \frac{1}{L}v(t) - \frac{R(t)}{L}i(t) - \frac{k\dot{x}(t)}{Lr(t)} \tag{19}$$

Finally, substituting Equation 12, one has

$$\frac{di(t)}{dt} = \frac{1}{L}v(t) - \frac{R_0 + \delta_R(1 - e^{\frac{-t}{\tau_R}})}{L}i(t) - \frac{k\dot{x}(t)}{Lr(t)} \tag{20}$$

Third, choosing x, $\dot{x}$, i and r as the state vector, i.e.

$$\vec{x} = \begin{bmatrix} x \\ \dot{x} \\ i \\ r \end{bmatrix} \tag{21}$$

$v(t)$ as the single input, u, and x and $\dot{x}$ as the output vector, $\vec{y}$,

$$\vec{y} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{22}$$

for the elevator dynamical system, one can form a continuous-time, nonlinear, time-varying system as

$$\begin{aligned} \dot{\vec{x}} = f(\vec{x}, \vec{u}) &= \begin{bmatrix} \dot{x} \\ \frac{1}{mr(t) - \frac{I_m}{r(t)}} \left(Wr(t) - mgr(t) + ki(t) - \frac{I_m \delta_c \dot{x}^2(t)}{2\pi r^3(t)} \right) \\ \frac{1}{L}v(t) - \frac{R_0 + \delta_R(1 - e^{\frac{-t}{\tau_R}})}{L}i(t) - \frac{k\dot{x}(t)}{Lr(t)} \\ \frac{\delta_c}{2\pi r(t)}\dot{x}(t) \end{bmatrix} \\ \vec{y} = h(\vec{x}) &= \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \end{aligned} \tag{23}$$

Equilibrium Points Computation

For equilibrium, assume that $t \to \infty$, then one can write

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = f(\bar{x}, \bar{u}) = \begin{bmatrix} \dot{\bar{x}} \\ \frac{1}{m\bar{r} - \frac{I_m}{\bar{r}}} \left(W\bar{r} - mg\bar{r} + k\bar{i} - \frac{I_m \delta_c \dot{\bar{x}}^2}{2\pi \bar{r}^3} \right) \\ \frac{1}{L}\bar{v} - \frac{R_0 + \delta_R}{L}\bar{i} - \frac{k\dot{\bar{x}}}{L\bar{r}} \\ \frac{\delta_c}{2\pi \bar{r}}\dot{\bar{x}} \end{bmatrix} \tag{24}$$

and rearranging, one has

$$\begin{bmatrix} \dot{\bar{x}} \\ \bar{r} \\ \bar{i} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{k}{(mg - W)}\bar{i} \\ \frac{1}{R_0 + \delta_R}\bar{v} \end{bmatrix} \tag{25}$$

or, by substitution of the third equation into the second, one has

$$\bar{v} = \frac{(mg - W)\,(R_0 + \delta_R)}{k}\bar{r} \tag{26}$$

Thus, by setting $\bar{v}$, one can "trim" the system for any provided mass of the cabin m, this will correspond to some $\bar{i}$, some $\bar{r}$, and, in turn to some $\bar{x}$ as any particular $\bar{x}$ and $\bar{r}$ are related by Equation 11 which must be integrated to find $\bar{x}$ at specific $\bar{r}$, i.e.

$$\bar{x} = \int_{r_0}^{\bar{r}} \frac{2\pi r}{\delta_c} dr \tag{27}$$

as $r = r_0$ at $x = 0$

Jacobian Linearization Outline

To linearize the elevator system about a specified equilibrium point, one can use the Jacobian to compute the state and input matrices required, i.e.

$$A = \left[\frac{\partial f}{\partial \vec{x}}(\bar{x}, \bar{u})\right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\bar{x}, \bar{u}) & \frac{\partial f_1}{\partial x_2}(\bar{x}, \bar{u}) & \frac{\partial f_1}{\partial x_3}(\bar{x}, \bar{u}) & \frac{\partial f_1}{\partial x_4}(\bar{x}, \bar{u}) \\ \frac{\partial f_2}{\partial x_1}(\bar{x}, \bar{u}) & \frac{\partial f_2}{\partial x_2}(\bar{x}, \bar{u}) & \frac{\partial f_2}{\partial x_3}(\bar{x}, \bar{u}) & \frac{\partial f_2}{\partial x_4}(\bar{x}, \bar{u}) \\ \frac{\partial f_3}{\partial x_1}(\bar{x}, \bar{u}) & \frac{\partial f_3}{\partial x_2}(\bar{x}, \bar{u}) & \frac{\partial f_3}{\partial x_3}(\bar{x}, \bar{u}) & \frac{\partial f_3}{\partial x_4}(\bar{x}, \bar{u}) \\ \frac{\partial f_4}{\partial x_1}(\bar{x}, \bar{u}) & \frac{\partial f_4}{\partial x_2}(\bar{x}, \bar{u}) & \frac{\partial f_4}{\partial x_3}(\bar{x}, \bar{u}) & \frac{\partial f_4}{\partial x_4}(\bar{x}, \bar{u}) \end{bmatrix} \tag{28}$$

$$B = \left[\frac{\partial f}{\partial \vec{u}}(\bar{x}, \bar{u})\right] = \begin{bmatrix} \frac{\partial f_1}{\partial u}(\bar{x}, \bar{u}) \\ \frac{\partial f_2}{\partial u}(\bar{x}, \bar{u}) \\ \frac{\partial f_3}{\partial u}(\bar{x}, \bar{u}) \\ \frac{\partial f_4}{\partial u}(\bar{x}, \bar{u}) \end{bmatrix} \tag{29}$$

where

$$\begin{bmatrix} f_1(\vec{x},\vec{u}) \\ f_2(\vec{x},\vec{u}) \\ f_3(\vec{x},\vec{u}) \\ f_4(\vec{x},\vec{u}) \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{1}{mr(t)-\frac{I_m}{r(t)}}\left(Wr(t) - mgr(t) + ki(t) - \frac{I_m\delta_c\dot{x}^2(t)}{2\pi r^3(t)}\right) \\ \frac{1}{L}v(t) - \frac{R_0+\delta_R(1-e^{\frac{-t}{\tau_R}})}{L}i(t) - \frac{k\dot{x}(t)}{Lr(t)} \\ \frac{\delta_c}{2\pi r(t)}\dot{x}(t) \end{bmatrix} \tag{30}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ i \\ r \end{bmatrix} \tag{31}$$

and

$$u = v \tag{32}$$

Thus, this linearization will yield an approximate LTI state-space model as

$$\begin{aligned} \Delta\dot{\vec{x}} &\approx A\Delta\vec{x} + B\Delta\vec{u} \\ \Delta\vec{y} &\approx \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}\Delta\vec{x} \end{aligned} \tag{33}$$

where

$$\Delta\vec{x} = \begin{bmatrix} x - \bar{x} \\ \dot{x} - \dot{\bar{x}} \\ \bar{i} \\ \bar{r} \end{bmatrix} \tag{34}$$

$$\Delta\vec{u} = v - \bar{v} \tag{35}$$

and

$$\Delta\vec{y} = \begin{bmatrix} x - \bar{x} \\ \dot{x} - \dot{\bar{x}} \end{bmatrix} \tag{36}$$

Project Assignment and Deliverables

For this project, assume that

1. I_m = 700 Nm and does not vary significantly with r
2. k = 100 Nm/A
3. L = 0.4 H
4. m = 500 kg
5. g = 9.81 m/s^2
6. $W = 300 \times g$ N
7. r_0 = 3 m
8. δ_c = 0.05 m
9. R_0 = 5 Ω
10. δ_R = 5Ω

11. $\tau_R = 3$ s

and perform the following

1. Setup two functions to compute the state and outputs in Python using the state-space system representation above.
2. Compute the linearized state-space system about three equilibrium/trim points at $x = 0$, 15, and 30 m for a steady-state value of $R = R_0 + \delta_R$, i.e. $R(t)$ as $t \to \infty$. Analyze the controllability, observability, and stability in Python for the linearized system.
3. Simulate the nonlinear system with a doublet input for v for initial conditions close and far from the three equilibrium points from part 2, but including t. Use a doublet magnitude which keeps the elevator velocity reasonable.
4. Simulate the linearized system with a doublet input for v for initial conditions close and far from the three equilibrium points from part 2. Use a doublet magnitude which keep the elevator velocity reasonable.
5. Write a few paragraphs summarizing the results of these simulations and any analysis. Make sure to compare the linear simulations to the nonlinear.

Recall

A doublet input is defined as the following:

$$u(t) = \begin{cases} 0 & \forall \ t < 0 \\ c & \forall \ 0 \leq t < \frac{\Delta t}{2} \\ -c & \forall \ \frac{\Delta t}{2} \leq t < \Delta t \\ 0 & \forall \ t \geq \Delta t \end{cases} \tag{37}$$

where Δt is the time length of the doublet and c is the doublet magnitude. This is noticeably similar to the step where the primary benefit is that stable systems will return to their original output *as well as* any integrated output state.