EECS 16B Designing Information Devices and Systems II
Fall 2017 Miki Lustig and Michel Maharbiz **Homework 8**

## This homework is due October 31, 2017, at noon.

### 1. SVD I

Find the singular value decomposition of the following matrix (leave all work in exact form, not decimal):

$$A = \begin{bmatrix} 2 & 2 \\ 3 & -3 \end{bmatrix}$$

(a) Find the eigenvalues of $AA^\top$ and order them from largest to smallest, $\lambda_1 > \lambda_2$.

(b) Find orthonormal eigenvectors $\vec{u}_i$ of $AA^\top$ (all eigenvectors are mutually orthogonal and unit length).

(c) Find the singular values $\sigma_i = \sqrt{\lambda_i}$. Find the $\vec{v}_i$ vectors from:

$$A^\top \vec{u}_i = \sigma_i \vec{v}_i$$

(d) Write out A as a weighted sum of rank-1 matrices:

$$A = \sigma_1 \vec{u}_1 \vec{v}_1^\top + \sigma_2 \vec{u}_2 \vec{v}_2^\top$$

(e) Use `numpy.linalg.svd` to compute the SVD of $A$ and compare it to your results for the previous part. Will calculating the SVD by hand and calculating it using `numpy` always return identical results (up to floating point error)? Why or why not?

*Note:* Be sure to carefully read the documentation for the `svd` function. In particular, pay attention to the format of the returned values.

### 2. SVD II

Find the singular value decomposition of the following matrix (leave all work in exact form, not decimal):

$$A = \begin{bmatrix} 1 & 0 & -\sqrt{3} \\ \sqrt{3} & 0 & 1 \\ 0 & 3 & 0 \end{bmatrix}$$

(a) Find the eigenvalues of $A^\top A$ and order them from largest to smallest, $\lambda_1 > \lambda_2$.

(b) Find orthonormal eigenvectors $\vec{v}_i$ of $A^\top A$ (all eigenvectors are mutually orthogonal and unit length).

(c) Find the singular values $\sigma_i = \sqrt{\lambda_i}$. Find the $\vec{u}_i$ vectors from:

$$A\vec{v}_i = \sigma_i \vec{u}_i$$

(d) Write out A as a weighted sum of rank-1 matrices:

$$A = \sigma_1 \vec{u}_1 \vec{v}_1^\top + \sigma_2 \vec{u}_2 \vec{v}_2^\top + \sigma_3 \vec{u}_3 \vec{v}_3^\top$$

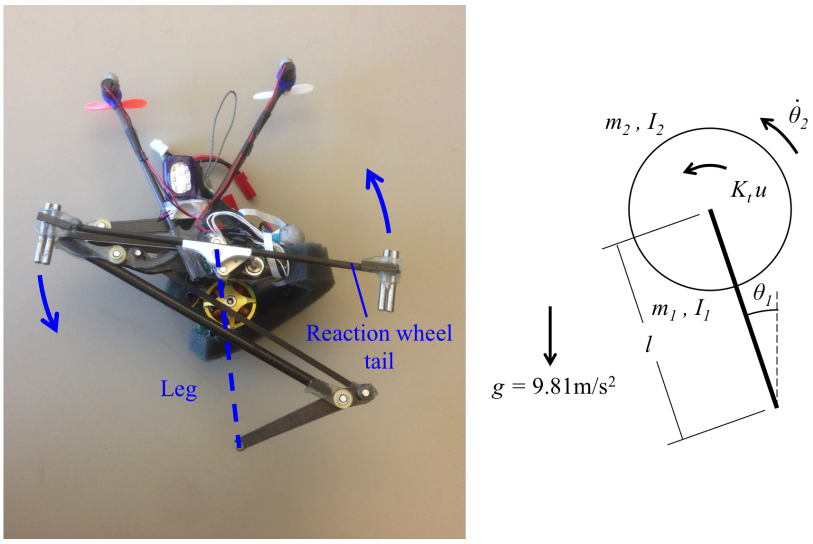


Figure 1: Picture of Salto and the x-z physics model. You can watch a video of Salto here: `http://www.youtube.com/watch?v=2dJmArHRn0U`

## 3. Balance

Justin is working on a small jumping robot named Salto. Salto can bounce around on the ground, but Justin would like Salto to balance on its toe and stand still. In this problem, we'll work on systems that could help Salto balance on its toe using its reaction wheel tail.

Standing on the ground, Salto's dynamics in the x-z plane (called the sagittal plane in biology) look like an inverted pendulum with a flywheel on the end:

$$\begin{aligned}(I_1 + (m_1+m_2)l^2)\ddot{\theta}_1 &= -K_t u + (m_1+m_2)lg\sin(\theta_1)\\ I_2\ddot{\theta}_2 &= K_t u\end{aligned}$$

Where $\theta_1$ is the angle of the robot's body relative to the ground (0 is straight up), $\dot{\theta}_1$ is its angular velocity, $\dot{\theta}_2$ is the angular velocity of the reaction wheel tail, and $u$ is the current input to the tail motor. $m_1, m_2, I_1, I_2, l, K_t$ are positive constants representing system parameters (masses and angular momentums of the body and tail, leg length, and motor torque constant respectively) and $g = 9.81\text{m/s}^2$ is the acceleration due to gravity.

Numerically substituting Salto's physical parameters, the differential equations become approximately:

$$\begin{aligned}0.001\ddot{\theta}_1 &= -0.025u + 0.1\sin(\theta_1)\\ 5(10^{-5})\ddot{\theta}_2 &= 0.025u\end{aligned}$$

For this problem, we'll look at a reduced suite of sensors on Salto. Our only output will be the tail encoder that measures the angular velocity of the tail relative to the body:

$$y = \dot{\theta}_2 - \dot{\theta}_1$$

(a) Using the state vector $[\theta_1, \dot{\theta}_1, \dot{\theta}_2]^\top$, input $u$, and output $y$ linearize the system about the the point $[0,0,0]^\top$. Write the linearized equations as $\frac{d}{dt}\vec{x} = A\vec{x} + Bu$ and $y = C\vec{x}$. Write the matrices with the physical numerical values, not symbolically.

*Note: since the tail is like a wheel, we care only about its angular velocity $\dot{\theta}_2$ and not its angle $\theta_2$.*

(b) Is the system fully controllable? Is the system fully observable?

(c) Design an observer of the form $\frac{d}{dt}\hat{x} = A\hat{x} + Bu - L(C\hat{x} - y)$ and solve for the gains in L that make the observer dynamics converge with all eigenvalues $\lambda_1 = \lambda_2 = \lambda_3 = -10$.

(d) Let's implement a controller for our system using an analog electrical circuit! You can use the following circuit components in Figure 2:

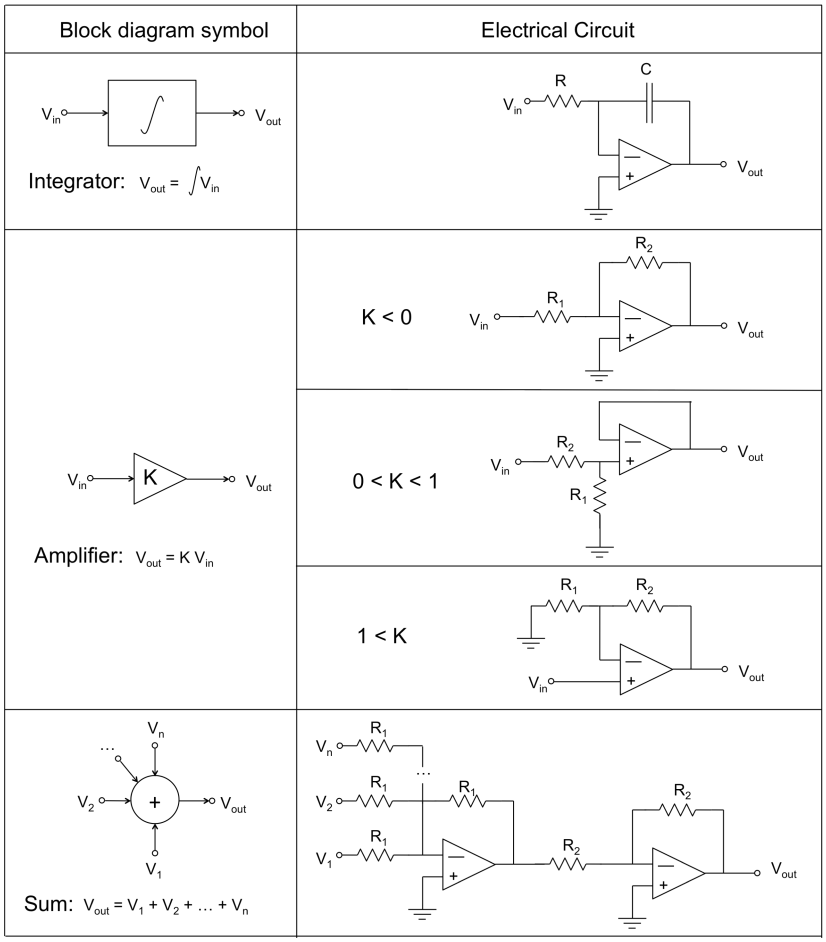


Figure 2: Circuit components and block diagram symbols.

Using state feedback, Justin has selected the control gains $\tilde{K} = \begin{bmatrix} 20 & 5 & 0.01 \end{bmatrix}$. Draw a circuit in the box in Figure 3 that implements this controller. Use relatively reasonable component values.

*Optional bonus: what are the eigenvalues of the closed loop dynamics for the given K?*

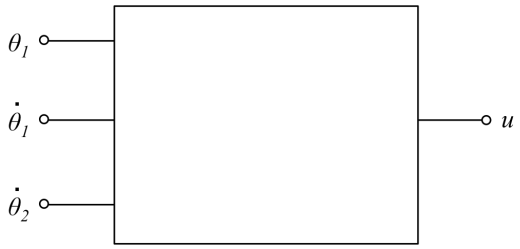


Figure 3: Fill in the box to implement the state feedback controller

## 4. Closed-loop control of SIXT33N

Last time, we discovered that open-loop control was not enough to ensure that our car goes straight in the event of model mismatch. In this problem, we will introduce closed-loop control which will hopefully make SIXT33N finally go straight.

Previously, we introduced $\delta(t) = d_L(t) - d_R(t)$ as the difference in positions between the two wheels. If both wheels of the car are going at the same velocity, then this difference $\delta$ should remain constant, since no wheel will advance by more ticks than the other. In our closed loop control scheme, we will consider a control scheme which will apply a simple proportional control $k_L$ and $k_R$ against $\delta(t)$ in order to try to prevent $|\delta(t)|$ from growing without bound.

$$v_L(t) = d_L(t+1) - d_L(t) = \theta_L u_L(t) - \beta_L$$
$$v_R(t) = d_R(t+1) - d_R(t) = \theta_R u_R(t) - \beta_R$$

We want to achieve the following equations:

$$v_L(t) = d_L(t+1) - d_L(t) = v^* - k_L\delta(t)$$
$$v_R(t) = d_R(t+1) - d_R(t) = v^* + k_R\delta(t)$$

We can put the equations in the following form to figure out how we should change our control inputs.

$$v_L(t) = d_L(t+1) - d_L(t) = \theta_L\left(\frac{v^* + \beta_L}{\theta_L} - k_L\frac{\delta(t)}{\theta_L}\right) - \beta_L$$
$$v_R(t) = d_R(t+1) - d_R(t) = \theta_R\left(\frac{v^* + \beta_R}{\theta_R} + k_R\frac{\delta(t)}{\theta_R}\right) - \beta_R$$

These are our new closed-loop control inputs - the new closed-loop proportional control is the $k_L/k_R$ term.

$$u_L(t) = \frac{v^* + \beta_L}{\theta_L} - k_L\frac{\delta(t)}{\theta_L}$$
$$u_R(t) = \frac{v^* + \beta_R}{\theta_R} + k_R\frac{\delta(t)}{\theta_R}$$

(a) Let's examine the feedback proportions $k_L$ and $k_R$ more closely. Should they be positive or negative? What do they mean? Think about how they interact with $\delta(t)$.

(b) Let's look a bit more closely at picking $k_L$ and $k_R$. Firstly, we need to figure out what happens to $\delta(t)$ over time. Find $\delta(t+1)$ in terms of $\delta(t)$.

(c) Given your work above, what is the eigenvalue of the system defined by $\delta(t)$? For discrete-time systems like our system, $\lambda \in [-1,1]$ is considered stable. Are $\lambda \in [0,1]$ and $\lambda \in [-1,0]$ identical in function for our system? Which one is "better"? (Hint: preventing oscillation is a desired benefit.)

Based on your choice for the range of $\lambda$ above, how should we set $k_L$ and $k_R$ in the end?

(d) Let's re-introduce the model mismatch from last week in order to model environmental discrepancies, disturbances, etc. How does closed-loop control fare under model mismatch? Find $\delta_{ss} = \delta(t \to \infty)$, assuming that $\delta(0) = \delta_0$. What is $\delta_{ss}$? (To make this easier, you may leave your answer in terms of appropriately defined $c$ and $\lambda$ obtained from an equation in the form of $\delta(t+1) = \delta(t)\lambda + c$.)
Check your work by verifying that you reproduce the equation in part (c) if all model mismatch terms are zero. Is it better than the open-loop model mismatch case from last week?

$$v_L(t) = d_L(t+1) - d_L(t) = (\theta_L + \Delta\theta_L)u_L(t) - (\beta_L + \Delta\beta_L)$$
$$v_R(t) = d_R(t+1) - d_R(t) = (\theta_R + \Delta\theta_R)u_R(t) - (\beta_R + \Delta\beta_R)$$

$$u_L(t) = \frac{v^* + \beta_L}{\theta_L} - k_L\frac{\delta(t)}{\theta_L}$$
$$u_R(t) = \frac{v^* + \beta_R}{\theta_R} + k_R\frac{\delta(t)}{\theta_R}$$

**Contributors:**

- Justin Yim.
- Tianrui Guo.
- Edward Wang.