EECS 16B Designing Information Devices and Systems II
Fall 2018 Elad Alon and Miki Lustig **Homework 8**

This homework is due on Wednesday, October 24, 2018, at 11:59PM.
Self-grades are due on Monday, October 29, 2018, at 11:59PM.

1. Midterm Controls Problem (Mechanical)

This problem is adapted from FA17 midterm 2, question 3.

You are given the system:

$$\vec{x}(t+1) = \begin{bmatrix} 3 & 1 \\ -1 & 5 \end{bmatrix} \vec{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

(a) Is the system stable?

Solution:
For a discrete system to be stable, $|\lambda_i| < 1$.
Characteristic polynomial:

$$\begin{aligned}(3-\lambda)(5-\lambda)+1 &= 0 \\ \lambda^2 - 8\lambda + 16 &= 0 \\ (\lambda-4)(\lambda-4) &= 0 \\ \lambda_{1,2} &= 4\end{aligned}$$

$|\lambda_{1,2}| \geq 1$, so the system is unstable.

(b) Design a state-feedback controller that will take the system from any state to $\vec{x}(t) = 0$ in 2 time steps.

Solution:
A state-feedback controller sets:

$$\begin{aligned}u(t) &= K\vec{x}(t) \\ K &= \begin{bmatrix} k_1 & k_2 \end{bmatrix}\end{aligned}$$

This means the system can be written as:

$$\vec{x}(t+1) = (A+BK)\vec{x}(t)$$

For the system to converge in 2 time steps, we need:

$$\lambda_1 = \lambda_2 = 0$$

Where λ_1 and λ_2 are the eigenvalues of $(A+BK)$

$$A+BK = \begin{bmatrix} 3 & 1 \\ -1+k_1 & 5+k_2 \end{bmatrix}$$

Characteristic polynomial:

$$(3-\lambda)(5+k_2-\lambda)-k_1+1 = \lambda^2+\lambda(-8-k_2)-k_1+3k_2+16$$

Coefficient match to:

$$\begin{aligned} (\lambda+0)(\lambda+0) &= \lambda^2 \\ -8-k_2 &= 0 \rightarrow k_2 = -8 \\ -k_1+3k_2+16 = -k_1-24+16 &= 0 \rightarrow k_1 = -8 \end{aligned}$$

(c) Design a controller that will take the system from $x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ to $x = \begin{bmatrix} 2 \\ 7 \end{bmatrix}$ in the minimum number of steps.

In your answer provide the values for $\vec{U} = U(t)$ till the target is hit. For example, if you need 5 steps, report $\vec{U} = \begin{bmatrix} U(0) & U(1) & U(2) & U(3) & U(4) \end{bmatrix}$.

Solution:
Controllability matrix:

$$\begin{bmatrix} AB & B \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 5 & 1 \end{bmatrix}$$

It's full rank, so the system is controllable. This means we know we can reach any target output in 2 steps.

$$\begin{aligned} \begin{bmatrix} AB & B \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \end{bmatrix} &= \begin{bmatrix} 2 \\ 7 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \end{bmatrix} &= \begin{bmatrix} 2 \\ 7 \end{bmatrix} \\ u(0) &= 2 \\ 5u(0)+u(1) = 7 &\rightarrow u(1) = -3 \\ \vec{U} &= \begin{bmatrix} 2 & -3 \end{bmatrix} \end{aligned}$$

(d) For the original system, we have an output:

$$y(t) = x_1(t) + 2x_2(t)$$

Is the system observable?

Solution: We can write $y(t)$ in the form:

$$\begin{aligned} y(t) &= C\vec{x}(t) \\ C &= \begin{bmatrix} 1 & 2 \end{bmatrix} \end{aligned}$$

The observability matrix is:

$$\begin{bmatrix} C \\ CA \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 11 \end{bmatrix}$$

Which is full rank, so the system is observable.

(e) For the same $y(t)$ given in part (d), we measure the output at $t = 0$ and $t = 1$ and find:

$$y(0) = -3$$
$$y(1) = 6$$

Determine the initial condition $\vec{x}(0)$ of this system.

Solution:

$$\begin{aligned} \begin{bmatrix} y(0) \\ y(1) \end{bmatrix} &= \begin{bmatrix} C \\ CA \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} \\ \begin{bmatrix} -3 \\ 6 \end{bmatrix} &= \begin{bmatrix} 1 & 2 \\ 1 & 11 \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} \\ x_1(0) + 2x_2(0) &= -3 \\ x_1(0) + 11x_2(0) &= 6 \\ x_1(0) &= -5 \\ x_2(0) &= 1 \\ \vec{x}(0) &= \begin{bmatrix} -5 \\ 1 \end{bmatrix} \end{aligned}$$

2. Least-Squares solution via gradient descent with a constant step size

In this problem, we will derive a dynamical system approach for solving a least-squares problem which finds the $\vec{x}$ that minimizes $||A\vec{x} - \vec{y}||^2$. We consider A to be thin and full rank.

As covered in EE16A, this has a closed solution:

$$\vec{\hat{x}} = (A^T A)^{-1} A^T \vec{y}$$

Direct computation requires the inversion of $A^T A$, which has a complexity of $O(N^3)$ where $(A^T A) \in \mathbb{R}^{NxN}$. This may be okay for small problems, but can easily become unfeasible for big data. Instead, we will solve the problem iteratively by the gradient descent method, which is posed as a state space equation.

(a) Given an estimate of $\vec{x}(t)$, we define the least squares error $\vec{\varepsilon}(t)$ to be:

$$\vec{\varepsilon}(t) = A\vec{x}(t) - \vec{y}$$

Show that if $\vec{x}(t) = \vec{\hat{x}}$, then $\vec{\varepsilon}(t)$ is orthogonal to the columns of A, i.e. show $A^T \vec{\varepsilon}(t) = 0$.

Solution:

$$\vec{\varepsilon}(t) = A(A^TA)^{-1}A^T\vec{y} - \vec{y}$$
$$A^T\vec{\varepsilon}(t) = A^T(A(A^TA)^{-1}A^T\vec{y} - \vec{y})$$
$$A^T\vec{\varepsilon}(t) = (A^TA)(A^TA)^{-1}A^T\vec{y} - A^T\vec{y}$$
$$A^T\vec{\varepsilon}(t) = IA^T\vec{y} - A^T\vec{y}$$
$$A^T\vec{\varepsilon}(t) = A^T\vec{y} - A^T\vec{y} = 0$$

(b) We would like to develop a "fictionary" state space equation for which the state $\vec{x}(t)$ will converge to $\vec{x}(t) \to \vec{\hat{x}}$, the least squares solution. If the argument $A^T(A\vec{\hat{x}} - \vec{y}) = 0$ then we define the following update:

$$\vec{x}(t+1) = \vec{x}(t) - \alpha A^T(A\vec{x}(t) - \vec{y})$$

You can see when $\vec{x}(t) = \vec{\hat{x}}$, the system reaches equilibrium. By the way, it is no coincidence that the gradient of $||A\vec{x} - \vec{y}||^2$ is

$$\nabla||A\vec{x} - \vec{y}||^2 = 2A^T(A\vec{x} - \vec{y})$$

This can be derived by vector derivatives (outside of class scope) or by partial derivatives as we did in the linearization case.

To show that $\vec{x} \to \vec{\hat{x}}$, we define a new state variable $\Delta\vec{x}(t) = \vec{x}(t) - \vec{\hat{x}}$ Derive the state evolution equation for $\Delta\vec{x}(t)$, and show that it takes the form:

$$\Delta\vec{x}(t+1) = (I - \alpha G)\Delta\vec{x}(t)$$

Solution:

$$\Delta\vec{x}(t+1) = \vec{x}(t+1) - \vec{\hat{x}}$$
$$\Delta\vec{x}(t+1) = \vec{x}(t) - \alpha A^T(A\vec{x}(t) - \vec{y}) - \vec{\hat{x}}$$
$$\Delta\vec{x}(t+1) = (\vec{x}(t) - \vec{\hat{x}}) - \alpha A^T(A\vec{x}(t) - \vec{y})$$
$$\Delta\vec{x}(t+1) = \Delta\vec{x}(t) - \alpha A^TA\vec{x}(t) - \alpha A^T\vec{y}$$
$$\Delta\vec{x}(t+1) = \Delta\vec{x}(t) - \alpha A^TA(\vec{x}(t) - (A^TA)^{-1}A^T\vec{y})$$
$$\Delta\vec{x}(t+1) = \Delta\vec{x}(t) - \alpha A^TA(\vec{x}(t) - \vec{\hat{x}})$$
$$\Delta\vec{x}(t+1) = \Delta\vec{x}(t) - \alpha A^TA(\Delta\vec{x}(t))$$
$$\Delta\vec{x}(t+1) = (I - \alpha A^TA)\Delta\vec{x}(t)$$

(c) We would like to drive the system such that $\Delta\vec{x}(t)$ converges to 0. What is the α that would result in the system to be stable, and converge fastest to $\Delta\vec{x} = 0$? Hint: the eigenvalues of matrix $I - G$ are $1 - \lambda_{i\{G\}}$, where $\lambda_{i\{G\}}$ are the eigenvalues of G.

Solution:

For a discrete system, the stability criteria is:

$$|\lambda_i| < 1$$

For $\Delta\vec{x}(t)$, the eigenvalues are $\lambda_i = 1 - \alpha\lambda_{i\{A^TA\}}$, where $\lambda_{i\{A^TA\}}$ are the eigenvalues of A^TA. Note that A^TA is a symmetric matrix with all eigenvalues $\lambda_{i\{A^TA\}} \geq 0$:

$$-1 < 1 - \alpha\lambda_{i\{A^TA\}} < 1$$

If we meet the condition

$$-1 < 1 - \alpha\lambda_{max\{A^TA\}} < 1$$

where $\lambda_{max\{A^TA\}}$ is the largest eigenvalue of A^TA, then we will meet the stability criteria for all eigenvalues. With a little bit of algebraic manipulation, we can get the range of α that makes the system stable:

$$0 < \alpha < \frac{2}{\lambda_{max\{A^TA\}}}$$

If we only cared about the largest eigenvalue of A^TA, then for this system to converge the fastest, we would want $1 - \alpha\lambda_{max\{A^TA\}} = 0$, which means we would choose:

$$\alpha = \frac{1}{\lambda_{max\{A^TA\}}}$$

However, we also need to think about the other eigenvalues of A^TA. With $\alpha = \frac{1}{\lambda_{max\{A^TA\}}}$, there will be other eigenvalues of the error system larger than 0 if not all eigenvalues of A^TA are the same value. The largest eigenvalue of the error corresponds to the minimum eigenvalue of A^TA, which we'll call $\lambda_{min\{A^TA\}}$. As we increase α past $\frac{1}{\lambda_{max\{A^TA\}}}$, the error eigenvalue corresponding to $\lambda_{min\{A^TA\}}$ will decrease in magnitude, but the eigenvalue corresponding to $\lambda_{max\{A^TA\}}$ will increase (since the eigenvalue starts going negative). To find the optimal α, we set the minimum and maximum eigenvalues' magnitudes equal to each other:

$$1 - \alpha\lambda_{min\{A^TA\}} = \alpha\lambda_{max\{A^TA\}} - 1$$

Note that the eigenvalue corresponding to $\lambda_{max\{A^TA\}}$ flipped signs since α was large enough to make the eigenvalue negative. This gives us an optimal step size of:

$$\alpha = \frac{2}{\lambda_{min\{A^TA\}} + \lambda_{max\{A^TA\}}}$$

Context:

$$\vec{x}(t+1) = \alpha A^T(A\vec{x}(t) - \vec{y})$$

In many practical problems it takes fewer iterations than N to converge to a sufficiently small $\Delta\vec{x}(t)$. Note that the only matrix multiplication we need to apply every step is $A^TA\vec{x}(t)$ ($A^T\vec{y}$ can be pre-computed). This has a complexity of $O(N^2)$. Therefore it **would** be possible to output the approximate solution faster if solved iteratively!In many important problems, the entries of A may be sparse, in that case the iterative algorithm will be even more efficient!

3. Rank 1 Decomposition

In this problem, we will decompose a few images into linear combinations of rank 1 matrices.

(a) Consider a standard 8×8 chessboard shown in Figure 1. Assume that black colors represent -1 and that white colors represent 1.

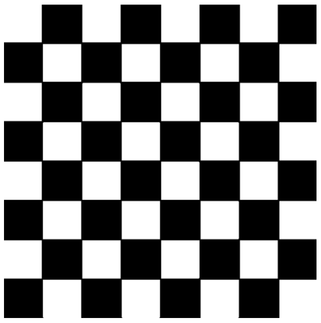


Figure 1: 8×8 chessboard.

Decompose this image into rank 1 images. Draw out each of these rank 1 images and express the original chessboard as a linear combination of the individual rank 1 images.

Solution:

The chessboard is already a rank 1 image, so we do not need to decompose it further.

(b) Assume that the chessboard is given by the following 8×8 matrix C_1:

$$C_1 = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix}$$

Express C_1 as a linear combination of outer products.

Solution:

Using the result from the previous part,

$$C_1 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}^T$$

(c) For the same chessboard shown in Figure 1, now assume that black colors represent 0 and that white colors represent 1.

Decompose this image into rank 1 images. Draw out each of these rank 1 images and express the original chessboard as a linear combination of the individual rank 1 images.

Solution:

The chessboard is now a rank 2 image, so we need to decompose it.

Image 1:

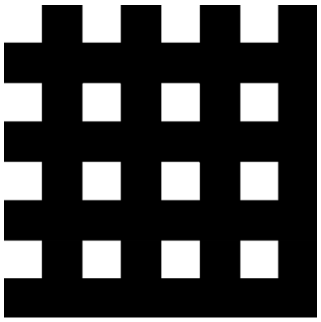
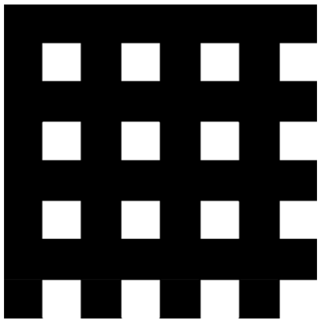


Image 2:



The chessboard equals the sum of image 1 and image 2.

(d) Assume that the chessboard is given by the following 8×8 matrix C_2:

$$C_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Express C as a linear combination of outer products.

Solution:

Using the result from the previous part,

$$C_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}^T + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}^T$$

(e) Now consider the Swiss flag shown in Figure 2. Assume that red colors represent 0 and that white colors represent 1.

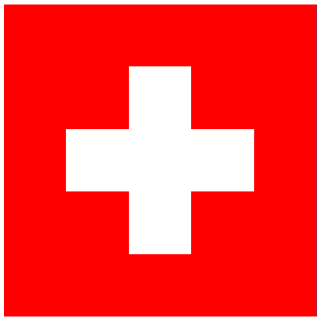


Figure 2: Swiss flag.

Decompose this image into rank 1 images. Draw out each of these rank 1 images and express the Swiss flag as a linear combination of the individual rank 1 images.

Solution:

The Swiss flag is a rank 2 image, so we need to find two rank 1 images.

Solution 1:

Image 1:

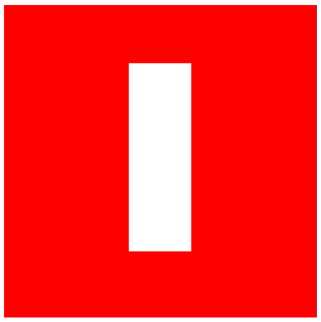
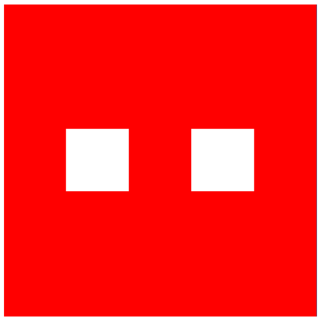


Image 2:



The Swiss flag equals the sum of image 1 and image 2.

Solution 2:

Image 1:

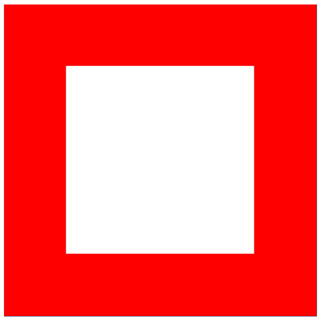
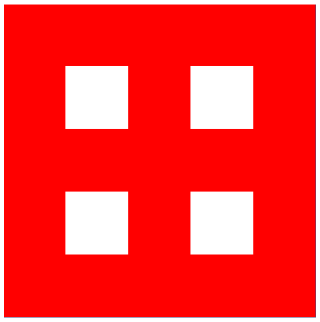


Image 2:



The Swiss flag equals the difference between image 1 and image 2.

(f) Assume that the Swiss flag is given by the following 5×5 matrix S:

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Express S as a linear combination of outer products.

Solution:

We can use the result from the previous part.

Solution 1:

$$S = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}^T + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}^T$$

Solution 2:

$$S = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}^T - \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}^T$$

4. Open-Loop Control of SIXT33N

Last time, we learned that the ideal input PWM for running a motor at a target velocity v^* is:

$$u[k] = \frac{v^* + \beta}{\theta}$$

In this problem, we will extend our analysis from one motor to a two-motor car system and evaluate how well our open-loop control scheme does.

$$v_L[k] = d_L[k+1] - d_L[k] = \theta_L u_L[k] - \beta_L$$
$$v_R[k] = d_R[k+1] - d_R[k] = \theta_R u_R[k] - \beta_R$$

(a) In reality, we need to "kickstart" electric motors with a pulse in order for them to work. That is, we can't go straight from 0 to our desired input signal for $u[k]$, since the motor needs to overcome its initial inertia in order to operate in accordance with our model.

Let us model the pulse as having a width (in timesteps) of k_p. In order to model this phenomenon, we can say that $u[k] = 255$ for $k \in [0, k_p - 1]$[1]. In addition, the car initially (at $k = 0$) hasn't moved, so we can also say $d[0] = 0$.

Firstly, let us examine what happens to d_L and d_R at $k = k_p$, that is, after the kickstart pulse has passed. Find $d_L[k_p]$ and $d_R[k_p]$. (*Hint:* If it helps, try finding $d_L[1]$ and $d_R[1]$ first and then generalizing your result to the k_p case.)

Note: It is very important that you distinguish θ_L and θ_R as the motors we have are liable to vary in their parameters, just as how real resistors vary from their ideal resistance.

Solution:

Applying the model directly, we get:

$$d[1] = d[0] + (255\theta - \beta)$$
$$d[2] = d[1] + (255\theta - \beta) = d[0] + (255\theta - \beta) + (255\theta - \beta) = d[0] + 2(255\theta - \beta)$$
$$d[3] = d[2] + (255\theta - \beta) = d[0] + 2(255\theta - \beta) + (255\theta - \beta) = d[0] + 3(255\theta - \beta)$$
$$d[k_p] = d[0] + k_p(255\theta - \beta) \text{ (by analogy)}$$
$$d[k_p] = k_p(255\theta - \beta) \text{ (substitute } d[0])$$

Thus we get:

$$d_L[k_p] = k_p(255\theta_L - \beta_L)$$
$$d_R[k_p] = k_p(255\theta_R - \beta_R)$$

(b) Let us define $\delta[k] = d_L[k] - d_R[k]$ as the difference in positions between the two wheels. If both wheels of the car are going at the same velocity, then this difference δ should remain constant since no wheel will advance by more ticks than the other. As a result, this will be useful in our analysis and in designing our control schemes.

Find $\delta[k_p]$. For both an ideal car ($\theta_L = \theta_R$ and $\beta_L = \beta_R$) where both motors are perfectly ideal and a non-ideal car ($\theta_L \neq \theta_R$ and $\beta_L \neq \beta_R$), did the car turn compared to before the pulse?

[1] $x \in [a,b]$ means that x goes from a to b inclusive.

Note: Since $d[0] = d_L[0] = d_R[0] = 0$, $\delta[0] = 0$.

Solution:

$$\begin{aligned}
\delta[k_p] &= d_L[k_p] - d_R[k_p] \\
\delta[k_p] &= k_p(255\theta_L - \beta_L) - k_p(255\theta_R - \beta_R) \\
\delta[k_p] &= k_p((255\theta_L - \beta_L) - (255\theta_R - \beta_R)) \\
\delta[k_p] &= k_p(255(\theta_L - \theta_R) - (\beta_L - \beta_R))
\end{aligned}$$

For an ideal car, both the $\theta_L - \theta_R$ and $\beta_L - \beta_R$ terms go to zero, so the pulse made the car go perfectly straight. However, in the non-ideal car, we aren't so lucky, since the car did turn somewhat during the initial pulse.

(c) We can still declare victory though, even if the car turns a little bit during the initial pulse (k_p will be very short in lab), so long as the car continues to go straight afterwards when we apply our control scheme; that is, as long as $\delta[k \to \infty]$ converges to a constant value (as opposed to going to $\pm\infty$ or oscillating).

Let's try applying the open-loop control scheme we learned last week to each of the motors independently, and see if our car still goes straight.

$$u_L[k] = \frac{v^* + \beta_L}{\theta_L}$$
$$u_R[k] = \frac{v^* + \beta_R}{\theta_R}$$

Let $\delta[k_p] = \delta_0$. Find $\delta[k]$ for $k \geq k_p$ in terms of δ_0. (*Hint:* As in part (a), if it helps you, try finding $\delta[k_p + 1]$, $\delta[k_p + 2]$, etc., and generalizing your result to the $\delta[k]$ case.)

Does $\delta[k \to \infty]$ deviate from δ_0? Why or why not?

Solution:

$$\begin{aligned}
\delta[k_p + 1] &= d_L[k_p + 1] - d_R[k_p + 1] \\
&= d_L[k_p] + \theta_L u[k] - \beta_L - (d_R[k_p] + \theta_R u[k] - \beta_R) \\
&= d_L[k_p] + \theta_L u[k] - \beta_L - d_R[k_p] - \theta_R u[k] + \beta_R \\
&= (d_L[k_p] - d_R[k_p]) + (\theta_L u[k] - \beta_L) - (\theta_R u[k] - \beta_R) \\
&= (d_L[k_p] - d_R[k_p]) + v^* - v^* \\
&= (d_L[k_p] - d_R[k_p]) \\
&= \delta[k_p] \\
&= \delta_0 \\
\delta[k \to \infty] &= \delta_0 \ \text{(by generalization: every step does not change } \delta)
\end{aligned}$$

Since we are able to apply just the right amount of input PWM to keep a constant velocity on both wheels, neither wheel gets ahead of the other, so $\delta[k]$ does not change, meaning that the car does not turn.

(d) Unfortunately, in real life, it is hard to capture the precise parameters of the car motors like θ and β, and even if we did manage to capture them, they could vary as a function of temperature, time, wheel conditions, battery voltage, etc. In order to model this effect of **model mismatch**, we consider model mismatch terms (such as $\Delta\theta_L$), which reflects the discrepancy between the model parameters and actual parameters:

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

Let us try applying the open-loop control scheme again to this new system. Note that **no model mismatch terms appear below** – this is intentional since our control scheme is derived from the model parameters for θ and β, not from the actual $\theta + \Delta\theta$, etc.[2]

$$u_L[k] = \frac{v^* + \beta_L}{\theta_L}$$
$$u_R[k] = \frac{v^* + \beta_R}{\theta_R}$$

As before, let $\delta[k_p] = \delta_0$. Find $\delta[k]$ for $t \geq k_p$ in terms of δ_0.
Does $\delta(t \to \infty)$ change from δ_0? Why or why not, and how is it different from the previous case of no model mismatch?

Solution:

$$\begin{aligned}
\delta[k_p+1] &= d_L[k_p+1] - d_R[k_p+1] \\
&= (d_L[k_p] + (\theta_L + \Delta\theta_L)u_L[k] - (\beta_L + \Delta\beta_L)) - (d_R[k_p] + (\theta_R + \Delta\theta_R)u_R[k] - (\beta_R + \Delta\beta_R)) \\
&= (d_L[k_p] - d_R[k_p]) + ((\theta_L + \Delta\theta_L)u_L[k] - (\beta_L + \Delta\beta_L)) - ((\theta_R + \Delta\theta_R)u_R[k] - (\beta_R + \Delta\beta_R)) \\
&= \delta[k_p] + ((\theta_L + \Delta\theta_L)u_L[k] - (\beta_L + \Delta\beta_L)) - ((\theta_R + \Delta\theta_R)u_R[k] - (\beta_R + \Delta\beta_R)) \\
&= \delta_0 + (\theta_L u_L[k] - \beta_L + \Delta\theta_L u_L[k] - \Delta\beta_L) - (\theta_R u_R[k] - \beta_R + \Delta\theta_R u_R[k] - \Delta\beta_R) \\
&= \delta_0 + (v^* + \Delta\theta_L u_L[k] - \Delta\beta_L) - (v^* + \Delta\theta_R u_R[k] - \Delta\beta_R) \\
&= \delta_0 + v^* - v^* + (\Delta\theta_L u_L[k] - \Delta\beta_L) - (\Delta\theta_R u_R[k] - \Delta\beta_R) \\
&= \delta_0 + (\Delta\theta_L u_L[k] - \Delta\beta_L) - (\Delta\theta_R u_R[k] - \Delta\beta_R) \\
&= \delta_0 + \left(\frac{\Delta\theta_L}{\theta_L}(v^* + \beta_L) - \Delta\beta_L\right) - \left(\frac{\Delta\theta_R}{\theta_R}(v^* + \beta_R) - \Delta\beta_R\right) \\
\delta[k] &= \delta_0 + (t - k_p)\left(\left(\frac{\Delta\theta_L}{\theta_L}(v^* + \beta_L) - \Delta\beta_L\right) - \left(\frac{\Delta\theta_R}{\theta_R}(v^* + \beta_R) - \Delta\beta_R\right)\right) \text{ (generalizing)}
\end{aligned}$$

If there is no model mismatch (i.e., all mismatch terms are zero), then we are back to the same case as last time (all those terms drop out, and δ does not change).
If there is model mismatch, however, we are not so lucky.

[2]Why not just do a better job of capturing the parameters, one may ask? Well, as noted above, the mismatch can vary as a function of an assortment of factors including temperature, time, wheel conditions, battery voltage, and it is not realistic to try to capture the parameters under every possible environment, so it is up to the control designer to ensure that the system can tolerate a reasonable amount of mismatch.

As $k \to \infty$, the term $\left(\left(\frac{\Delta\theta_L}{\theta_L}(v^* + \beta_L) - \Delta\beta_L\right) - \left(\frac{\Delta\theta_R}{\theta_R}(v^* + \beta_R) - \Delta\beta_R\right)\right)$ (which is highly unlikely to be zero) causes δ to either steadily increase or decrease, meaning that the car turns steadily more and more.

You may have noticed that open-loop control is insufficient in light of non-idealities and mismatches. Next time, we will analyze a more powerful form of control (closed-loop control) which should be more robust against these kinds of problems.

5. **Write Your Own Question And Provide a Thorough Solution.**

Writing your own problems is a very important way to really learn material. The famous "Bloom's Taxonomy" that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top level. We rarely ask you any homework questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself (e.g. making flashcards). But we don't want the same to be true about the highest level. As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams. Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don't have to achieve this every week. But unless you try every week, it probably won't ever happen.