EECS 16B Designing Information Devices and Systems II
Fall 2018 Elad Alon and Miki Lustig **Homework 9**

This homework is due Wednesday, October 31, 2018, at 11:59pm.
Self grades are due Monday, November 5, 2018, at 11:59pm.

1. SVD (Mechanical)

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

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

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

Solution:

$$A^\top A = \begin{bmatrix} \sqrt{3} & 1 & 0 \\ 0 & 0 & 3 \\ 1 & -\sqrt{3} & 0 \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 & 1 \\ 1 & 0 & -\sqrt{3} \\ 0 & 3 & 0 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

$$\lambda_1 = 9, \quad \lambda_2 = 4, \quad \lambda_3 = 4$$

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

Solution:

$\lambda_1 = 9$:

$$\text{Null}(A^\top A - 9I) = \text{Null}\left(\begin{bmatrix} -5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -5 \end{bmatrix}\right) = \text{span}\left\{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right\}$$

Alternatively:

$$\begin{bmatrix} 4 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} v_{11} \\ v_{12} \\ v_{13} \end{bmatrix} = \begin{bmatrix} 9v_{11} \\ 9v_{12} \\ 9v_{13} \end{bmatrix} \implies v_{11} = 0, v_{12} = 1, v_{13} = 0$$

$$\vec{v}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Since $\lambda_2 = \lambda_3$, any two mutually orthogonal unit vectors that are also orthogonal to $\vec{v}_1 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top$ will work. For example:

$$\vec{v}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \vec{v}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

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

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

Solution:

$$\sigma_1 = 3, \quad \sigma_2 = 2, \quad \sigma_3 = 2$$

$$\vec{u}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \vec{u}_2 = \begin{bmatrix} \frac{\sqrt{3}}{2} \\ \frac{1}{2} \\ 0 \end{bmatrix}, \quad \vec{u}_3 = \begin{bmatrix} \frac{1}{2} \\ -\frac{\sqrt{3}}{2} \\ 0 \end{bmatrix}$$

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

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

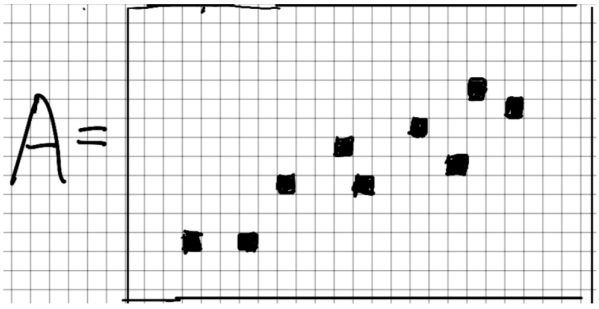
Solution:

$$A = 3 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + 2 \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 2 \begin{bmatrix} 0 & 0 & \frac{1}{2} \\ 0 & 0 & -\frac{\sqrt{3}}{2} \\ 0 & 0 & 0 \end{bmatrix}$$

2. SVD Midterm question

This question comes from fa17 midterm 2.

Consider the following matrix with non-zero entries in the black squares and zeros elsewhere:



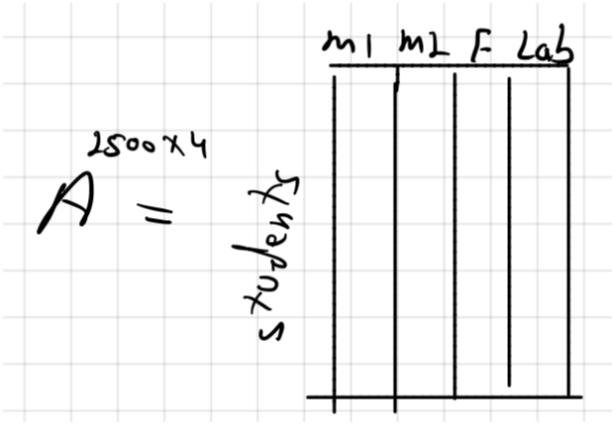
What is the number of non-zero singular values of A? explain.

Solution: The number of non-zero singular values of a matrix is equal to its rank. A has 7 independent columns, which means its rank as well as number of non-zero singular values is 7.

3. PCA Midterm question

This question comes from fa17 midterm 2.

Consider a matrix $A \in \mathbb{R}^{2500\times 4}$ which represents the EE16B Sp'2020 midterm 1, midterm 2, final and lab grades for all 2500 students taking the class.

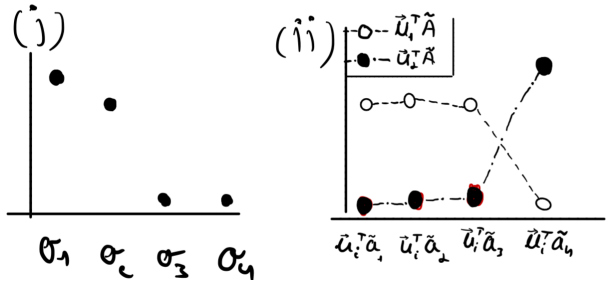


To perform PCA, you subtract the mean of each column and store the results in $\tilde{A}$. Your analysis includes:

(a) Computing the SVD: $\tilde{A} = \sigma_1\vec{u}_1\vec{v}_1^T + \sigma_2\vec{u}_2\vec{v}_2^T + \sigma_3\vec{u}_3\vec{v}_3^T + \sigma_4\vec{u}_4\vec{v}_4^T$ and plot the singular values.

(b) Computing the graph $\vec{u}_1^T\tilde{A}$ and $\vec{u}_2^T\tilde{A}$

The analysis data are plotted below:



Based on the analysis, answer the following true or false questions. Briefly explain your answer.

(a) The data can be approximated well by two principle components.

Solution:

True. From graph (i), there are two significant singular values, the rest are small.

(b) The students' exam scores have significant correlation between the exams.

Solution:

True. $\vec{u}_i^T\tilde{a}_i$ gives the correlation between the ith principle component and the variable stored in the ith column of $\tilde{A}$. From graph (ii), both midterms as well as final grades are highly correlated with the first principle component while all three also have little to no correlation to the second principle component. This means the exam scores are highly correlated with each other.

(c) The middle plot (ii) shows that students who did well on the exam did not do well in the labs and vice versa.

Solution:

False. Graph (ii) shows that the exam scores and lab scores are not correlated at all. i.e. we cannot determine a student's lab score from their exam scores and vice versa.

(d) One of the principle components attributes is solely associated with lab scores and not with exam scores.

Solution:

True. From graph (ii), the second principle component is only highly correlated with the lab scores.

4. Least-squares solution via gradient descent with a constant step size code

Last homework, we saw how we could reduce the complexity of the least squares regression by iteratively solving for $\vec{x}$. This homework, we'll create code to compare the speed of the closed form solution $\vec{\hat{x}}$ to the iteratively solved $\vec{x}$.

Please complete the notebook by following the instructions given.

Solution: The notebook least_squares_sol.ipynb contains the solutions to this exercise.

5. Eigenfaces

In this problem, we will be be exploring the use of PCA to compress and visualize pictures of human faces. We use the images from the data set Labeled Faces in the Wild. Specifically, we use a set of 13,232 images aligned using deep funneling to ensure that the faces are centered in each photo. Each image is a 100x100 image with the face aligned in the center. To turn the image into a vector, we stack each column of pixels in the image on top of each other, and we normalize each pixel value to be between 0 and 1. Thus, a single image of a face is represented by a 10,000 dimensional vector. A vector this size is a bit challenging to work with directly. We combine the vectors from each image into a single matrix so that we can run PCA. For this problem, we will provide you with the first 1000 principal components, but you can explore how well the images are compressed with fewer components. Please refer to the IPython notebook to answer the following questions.

(a) We provide you with a randomly selected subset of 1000 faces from the training set, the first 1000 principle components, all 13,232 singular values, and the average of all of the faces. What do we need the average of the faces for?

Solution: We need to zero-center the data by subtracting out the average before running PCA. During the reconstruction, we need to add the average back in.

(b) We provide you with a set of faces from the training set and compress them using the first 100 principal components. You can adjust the number of principal components used to do the compression between 1 and 1000. What changes do you see in the compressed images when you used a small number of components and what changes do you see when you use a large number?

Solution: When fewer principal components are used, the images do not differ much from the average face and do not contain many distinguishinig features. This is to be expected, since very small numbers of components will not account for much of the variation found in faces. When more principal components are used, the images more closely resemble the originals.

(c) You can visualize each principal component to see what each dimension "adds" to the high-dimensional image. What visual differences do you see in the first few components compared to the last few components?

Solution: The first few principal components are blury images capturing low frequency data. This low frequency data captures some of the broad variation across faces like lighting. The last few components contain high frequency data where small details vary from face to face.

(d) By using PCA on the face images, we obtain orthogonal vectors that point in directions of high variance in the original images. We can use these vectors to transform the data into a lower dimensional space and plot the data points. In the notebook, we provide you with code to plot a subset of 1000 images using the first two principal comonents. Try plotting other components of the data, and see how the shape of the points change. What difference do you see in the plot when you use the first two principal components compared with the last two principal components? What do you think is the cause of this difference?

Solution: The variance of the points in the plot is larger for the first two components compared to the last two components. We can also confirm that the variance is larger for the first few components because the singular values are large while the singular values for the last few components are small. This happens because PCA orders the principal components by the singular values, which can be used to measure the variability in the data for each component.

(e) We can use the principal components to generate new faces randomly. We accomplish this by picking a random point in the low-dimensional space and then multiplying it by the matrix of principal components. In the notebook, we provide you with code to generate faces using the first 1000 principal components. You can adjust the number of components used. How does this affect the resulting images?

Solution: When fewer components are used, the faces appear more similar and when a very small number of principal components are used, they are almost indistinguishable. When we use more principal components, the synthesized faces appear more distinct. This happens because we are adding more degrees of freedom to our "face" vector when we add more principal components. This allows us to generate faces with more variety because we have more parameters that control how the face looks.

6. Closed-Loop Control of SIXT33N

Last week, we discovered that open-loop control does not ensure that our car goes straight in the event of model mismatch. To make our control more robust, we introduce feedback, turning our open-loop controller into a closed-loop controller. In this problem, we derive the closed-loop control scheme you will use to make SIXT33N reliably drive straight.

Previously, we introduced $\delta[k] = d_L[k] - d_R[k]$ 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. We will consider a proportional control scheme, which introduces a feedback term into our input equation in which we apply gains k_L and k_R to $\delta[k]$ to modify our input velocity at each timestep in an effort to prevent $|\delta[k]|$ from growing without bound. To do this, we will modify our inputs $u_L[k]$ and $u_R[k]$ accordingly:

We begin with our open-loop equations from last week:

$$\begin{aligned} v_L[k] &= d_L[k+1] - d_L[k] = \theta_L u_L[k] - \beta_L \\ v_R[k] &= d_R[k+1] - d_R[k] = \theta_R u_R[k] - \beta_R \end{aligned} \tag{1}$$

We want to adjust $v_L[k]$ and $v_R[k]$ at each timestep based on $\delta[k]$: instead of v^*, our desired velocities are

now $v^* - k_L\delta[k]$ and $v^* + k_R\delta[k]$. As v_L and v_R go to v^*, $\delta[k]$ goes to zero.

$$v_L[k] = d_L[k+1] - d_L[k] = v^* - k_L\delta[k] \qquad (2)$$
$$v_R[k] = d_R[k+1] - d_R[k] = v^* + k_R\delta[k]$$

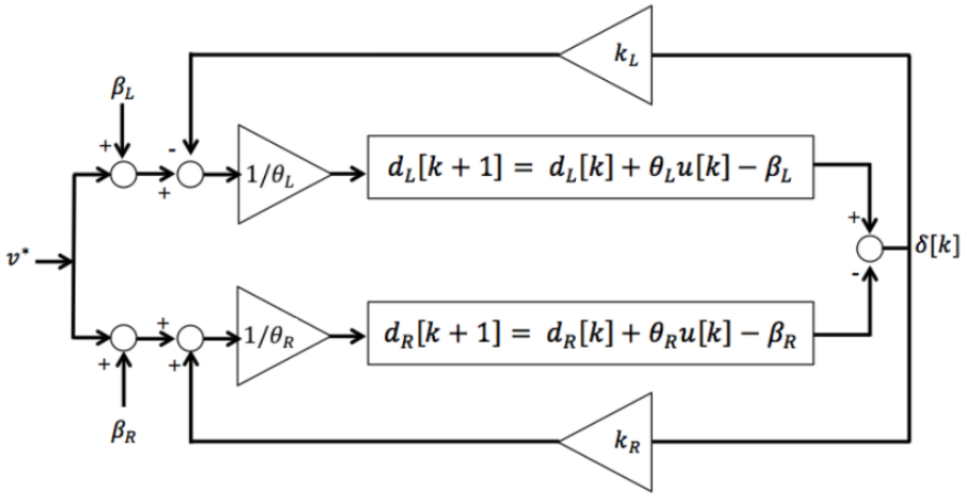
Setting the open-loop equation for v equal to our desired equation for v, we solve for the inputs u_L and u_R:

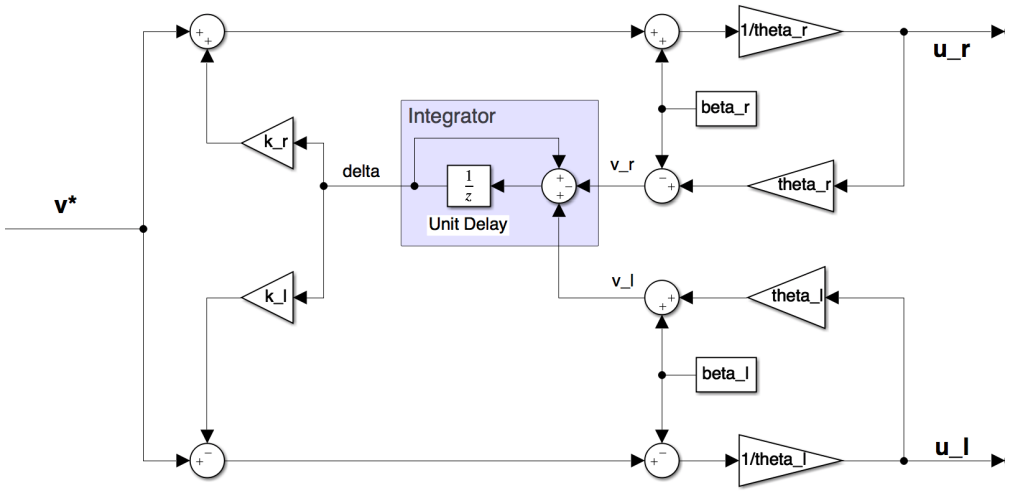
$$u_L[k] = \frac{v^* + \beta_L}{\theta_L} - k_L\frac{\delta[k]}{\theta_L}$$
$$u_R[k] = \frac{v^* + \beta_R}{\theta_R} + k_R\frac{\delta[k]}{\theta_R}$$

Now, we plug these i nputs into our original equations for v_L and v_R to complete the model:

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

Below are two block diagrams representing our control scheme. The first is the one you saw in lab: in this one, we abstract away the calculation of $u_L[k]$ and $u_R[k]$ and directly plug them into our plants, $d_L[k+1] - d_L[k] = \theta_L u[k] - \beta_L$ and $d_L[k+1] - d_L[k] = \theta_L u[k] - \beta_L$. The second shows how $u_L[k]$ and $u_R[k]$ are calculated.





$u_L[k]$ and $u_R[k]$ are represented as outputs because they are essentially the outputs of the system you implemented in your code: all the calculations you go through at each timestep lead up to the calculation of $u_L[k]$ and $u_R[k]$, which are then input into the **physical** system via the actuators that generate the PWM signal and make the wheels turn (i.e., the Launchpad's signal pins and the motors/motor controller circuitry). Do not be alarmed by the integrator block: it merely accumulates the velocities to determine distance, and calculates $\delta[k]$ from the differences of the distances. You do not have a direct representation of this integration in your code because the Launchpad stores your distance traveled, but we need it in the block diagram representation because your chosen $v*$, the primary input to the system, is a *velocity*, and is independent of total distance traveled: the block diagram is a direct representation of the system created by setting the equations in (1) equal to their respective equations in (2). Delay blocks in discrete-time systems act like memory overwritten at each timestep, and we can use them to create discrete-time integrators, representing persistent memory, like the one shown in the second system above. Keep in mind, $v_L[k]$ and $v_R[k]$ are the *modeled* $v_L[k]$ and $v_R[k]$: according to the equations of our model, this is how $v_L[k]$ and $v_R[k]$ should evolve, and we do not attempt to mathematically model the full complexity of the physical actuator system but instead account for its influence with θ and β.

(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 $\delta[k]$.

Solution:

If $\delta[k] > 0$, it means that $d_L[k] > d_R[k]$, so the left wheel is ahead of the right one. In order to correct for this, we should help the right wheel catch up, and we should do this by making $k_L > 0$ in order to apply less power on the left wheel and $k_R > 0$ in order to apply more power to the right wheel.

Likewise, if $\delta[k] < 0$, it means that $d_L[k] < d_R[k]$, so the right wheel is ahead of the left one. In this case, $k_L > 0$ is still valid, since $k_L\delta[k] > 0$ and so the left wheel speeds up, and likewise $k_R > 0$ is still correct since $k_R\delta[k] < 0$ so the right wheel slows down.

(b) Let's look a bit more closely at picking k_L and k_R. First, we need to figure out what happens to $\delta[k]$ over time. Find $\delta[k+1]$ in terms of $\delta[k]$.

Solution:

$$\begin{aligned}
\delta[k+1] &= d_L[k+1] - d_R[k+1] \\
&= v^* - k_L\delta[k] + d_L[k] - (v^* + k_R\delta[k] + d_R[k]) \\
&= v^* - k_L\delta[k] + d_L[k] - v^* - k_R\delta[k] - d_R[k] \\
&= -k_L\delta[k] - k_R\delta[k] + (d_L[k] - d_R[k]) \\
&= -k_L\delta[k] - k_R\delta[k] + \delta[k] \\
&= \delta[k](1 - k_L - k_R)
\end{aligned}$$

(c) Given your work above, what is the eigenvalue of the system defined by $\delta[k]$? 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 λ above, how should we set k_L and k_R in the end?

Solution:

The eigenvalue is $\lambda = 1 - k_L - k_R$.

As a discrete system, both are stable, but $\lambda \in (-1,0]$ will cause the car to oscillate due to overly high gain. Therefore, we should choose $\lambda \in [0,1)$.

As a result, $1 - k_L - k_R \in [0,1) \implies (k_L + k_R) \in (0,1]$ means that we should set the gains such that $(k_L + k_R) \in [0,1)$.

(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 $\delta_{ss} = \delta[k \to \infty]$, assuming that $\delta[0] = \delta_0$. What is δ_{ss}? (To make this easier, you may leave your answer in terms of appropriately defined c and λ obtained from an equation in the form of $\delta[k+1] = \delta[k]\lambda + c$.)

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?

$$\begin{aligned}
v_L[k] &= d_L[k+1] - d_L[k] = (\theta_L + \Delta\theta_L)u_L[k] - (\beta_L + \Delta\beta_L) \\
v_R[k] &= d_R[k+1] - d_R[k] = (\theta_R + \Delta\theta_R)u_R[k] - (\beta_R + \Delta\beta_R)
\end{aligned}$$

$$\begin{aligned}
u_L[k] &= \frac{v^* + \beta_L}{\theta_L} - k_L\frac{\delta[k]}{\theta_L} \\
u_R[k] &= \frac{v^* + \beta_R}{\theta_R} + k_R\frac{\delta[k]}{\theta_R}
\end{aligned}$$

Solution:

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

Let us define $c = \left(\frac{\Delta\theta_L}{\theta_L}(v^* + \beta_L) - \Delta\beta_L\right) - \left(\frac{\Delta\theta_R}{\theta_R}(v^* + \beta_R) - \Delta\beta_R\right)$, and our new eigenvalue $\lambda = 1 - k_L - k_R - k_L\frac{\Delta\theta_L}{\theta_L} - k_R\frac{\Delta\theta_R}{\theta_R}$. In this case,

$$
\begin{aligned}
\delta[1] &= \delta_0\lambda + c \\
\delta[2] &= \lambda(\delta_0\lambda + c) + c && = \delta_0\lambda^2 + c\lambda + c \\
\delta[3] &= \lambda(\delta_0\lambda^2 + c\lambda + c) + c && = \delta_0\lambda^3 + c\lambda^2 + c\lambda + c \\
\delta[4] &= \lambda(\delta_0\lambda^3 + c\lambda^2 + c\lambda + c) + c && = \delta_0\lambda^4 + c\lambda^3 + c\lambda^2 + c\lambda + c \\
\delta[5] &= \delta_0\lambda^5 + c(\lambda^4 + \lambda^3 + \lambda^2 + \lambda^1 + 1) \\
\delta[n] &= \delta_0\lambda^n + c(1 + \lambda^1 + \lambda^2 + \lambda^3 + \lambda^4 + \ldots + \lambda^n) \\
\delta[n] &= \delta_0\lambda^n + c\left(\sum_{k=0}^{n}\lambda^k\right) \text{ (rewriting in sum notation)} \\
\delta[n] &= \delta_0\lambda^n + c\left(\frac{1 - \lambda^n}{1 - \lambda}\right) \text{ (sum of a geometric series)}
\end{aligned}
$$

If $\lambda < 1$, then $\lambda^\infty = 0$, so those terms drop out:

$$
\begin{aligned}
\delta[n = t \to \infty] &= \delta_0\lambda^\infty + c\left(\frac{1 - \lambda^\infty}{1 - \lambda}\right) \\
\delta[n = t \to \infty] &= c\frac{1}{1 - \lambda}
\end{aligned}
$$

$$
\delta_{ss} = c\frac{1}{1 - \lambda}
$$

For your entertainment only: δ_{ss} is fully-expanded form (not required) is

$$\frac{\left(\frac{\Delta\theta_L}{\theta_L}(v^*+\beta_L)-\Delta\beta_L\right)-\left(\frac{\Delta\theta_R}{\theta_R}(v^*+\beta_R)-\Delta\beta_R\right)}{k_L+k_R+k_L\frac{\Delta\theta_L}{\theta_L}+k_R\frac{\Delta\theta_R}{\theta_R}}$$

The answer is correct because plugging in zero into all the model mismatch terms into c causes $c=0$, so $\delta_{ss}=0$ if there is no model mismatch. Compared to the open-loop result of $\delta_{ss}=\pm\infty$, the closed loop $\delta_{ss}=c\frac{1}{1-\lambda}$ is a much-desired improvement.

What does this mean for the car? It means that the car will turn initially for a bit but eventually converge to a fixed heading and keep going straight from there.

7. 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.