# Homework 8

**This homework is due on Saturday, March 16, 2024, at 11:59PM.**

**1. Transition Matrix**

(a) You are provided a matrix $\mathbf{A} = \begin{bmatrix} 0.2 & 0.8 & 0.2 \\ 0 & 0.4 & 0.2 \\ 0 & 0 & 0.8 \end{bmatrix}$. Matrix $\mathbf{A}$ is transition matrix where $\vec{x}[i+1] = \mathbf{A}\vec{x}[i]$. Additionally, the state vector at timestep $i = 1$ is $\vec{x}[1] = \begin{bmatrix} 8 \\ 1 \\ 4 \end{bmatrix}$. **After infinite timesteps, what is the value of the state vector $\vec{x}[i]$? That is, find $\lim_{i\to\infty} \vec{x}[i]$.**

**Solution:** The matrix $A$ is upper triangular, thus the eigenvalues can be determined from inspection as its diagonal elements: $\lambda_1 = 0.2$, $\lambda_2 = 0.4$, and $\lambda_3 = 0.8$.

The initial state vector $\vec{x}[1]$ can be decomposed as a linear combination of the three eigenvectors as $\vec{x}[1] = \alpha\,\vec{v}_1 + \beta\,\vec{v}_2 + \gamma\,\vec{v}_3$ since the eigenvalues are distinct. Then the steady-state value of $\vec{x}[i]$ is evaluated using some eigenvector to eigenvalue simplifications.

$$\lim_{i\to\infty} \vec{x}[i] = \lim_{i\to\infty} A^i\vec{x}[1] = \lim_{i\to\infty} \left(A^i \cdot (\alpha\,\vec{v}_1 + \beta\,\vec{v}_2 + \gamma\,\vec{v}_3)\right) = \lim_{i\to\infty} \left(\alpha\lambda_1^i\vec{v}_1 + \beta\lambda_2^i\vec{v}_2 + \gamma\lambda_3^i\vec{v}_3\right) \quad (1)$$

$$= \lim_{i\to\infty} \left(\alpha\,(0.2)^i\vec{v}_1 + \beta\,(0.4)^i\vec{v}_2 + \gamma\,(0.8)^i\vec{v}_3\right) \quad (2)$$

$$\lim_{i\to\infty} \vec{x}[i] = \vec{0} \quad (3)$$

Finally, since all eigenvalues have magnitude less than one, the value of the state vector as $i$ approaches infinity is $\lim_{i\to\infty} \vec{x}[i] = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$.

**2. Discrete System Induction**

Consider the discrete system given by

$$\vec{x}[i+1] = A\vec{x}[i] + B\vec{u}[i] \tag{4}$$

(a) Write $\vec{x}[1]$ in terms of $\vec{x}[0]$ and $\vec{u}[0]$. Then, write $\vec{x}[2]$ in terms of $\vec{x}[0]$, $\vec{u}[0]$, and $\vec{u}[1]$. Finally, write $\vec{x}[3]$ in terms of $\vec{x}[0]$, $\vec{u}[0]$, $\vec{u}[1]$, and $\vec{u}[2]$.

**Solution:** Firstly, we have

$$\vec{x}[1] = A\vec{x}[0] + B\vec{u}[0] \tag{5}$$

Then, we have

$$\begin{aligned}\vec{x}[2] &= A\vec{x}[1] + B\vec{u}[1] &(6)\\ &= A(A\vec{x}[0] + B\vec{u}[0]) + B\vec{u}[1] &(7)\\ &= A^2\vec{x}[0] + AB\vec{u}[0] + B\vec{u}[1] &(8)\end{aligned}$$

And lastly,

$$\begin{aligned}\vec{x}[3] &= A\vec{x}[2] + B\vec{u}[2] &(9)\\ &= A\Big(A^2\vec{x}[0] + AB\vec{u}[0] + B\vec{u}[1]\Big) + B\vec{u}[2] &(10)\\ &= A^3\vec{x}[0] + A^2B\vec{u}[0] + AB\vec{u}[1] + B\vec{u}[2] &(11)\end{aligned}$$

(b) We can generalize the above process to write $\vec{x}[i]$ in terms of $\vec{x}[0]$ and $\vec{u}[0], \ldots, \vec{u}[i-1]$ as follows:

$$\vec{x}[i] = A^i\vec{x}[0] + \sum_{j=0}^{i-1} A^{i-1-j}B\vec{u}[j] \tag{12}$$

**Verify that this equation holds for $i = 1$.** *This is equivalent to testing your base case in induction.* (Note that this holds for $i = 0$ because the summation $\sum_{j=0}^{-1}(\cdot)$ is the empty set.)

**Solution:** Plugging in $i = 1$, we have $\vec{x}[1] = A\vec{x}[0] + \sum_{j=0}^{0} A^{i-1-j}B\vec{u}[j] = A\vec{x}[0] + B\vec{u}[0]$

(c) **Show that if eq. (12) holds for $\vec{x}[i]$, it also holds for $\vec{x}[i+1]$.** That is, write $\vec{x}[i+1]$ in terms of $\vec{x}[i]$ and plug in eq. (12) for $\vec{x}[i]$. Show that this simplifies to eq. (12), where we now replace $i$ with $i+1$. *This is equivalent to verifying our inductive hypothesis.*

**Solution:** We have that

$$\begin{aligned}\vec{x}[i+1] &= A\vec{x}[i] + B\vec{u}[i] &(13)\\ &= A\left(A^i\vec{x}[0] + \sum_{j=0}^{i-1} A^{i-1-j}B\vec{u}[j]\right) + B\vec{u}[i] &(14)\\ &= A^{i+1}\vec{x}[0] + \sum_{j=0}^{i-1} A^{i-j}B\vec{u}[j] + A^{i-i}B\vec{u}[i] &(15)\\ &= A^{i+1}\vec{x}[0] + \sum_{j=0}^{i} A^{i-j}B\vec{u}[j] &(16)\end{aligned}$$

which is exactly eq. (12) where we plug in $i+1$ for $i$.

**3. System Identification**

You are given a discrete-time system as a black box. You don't know the specifics of the system but you know that it takes one scalar input and has two states that you can observe. You assume that the system is linear and of the form

$$\vec{x}[i+1] = A\vec{x}[i] + Bu[i] + \vec{w}[i], \tag{17}$$

where $\vec{w}[i]$ is an unknown small external disturbance, $u[i]$ is a scalar input, and

$$A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad x[i] = \begin{bmatrix} x_1[i] \\ x_2[i] \end{bmatrix}. \tag{18}$$

You want to identify the system parameters ($a_1$, $a_2$, $a_3$, $a_4$, $b_1$ and $b_2$) from measured data. However, you can only interact with the system via a black box model. In other words, you can only evaluate the system via observing the states $\vec{x}[i]$ and setting the inputs $u[i]$ that allow the system to move to the next state.

(a) You observe that the system has state $\vec{x}[i] = \begin{bmatrix} x_1[i] & x_2[i] \end{bmatrix}^\top$ at time $i$. You pass input $u[i]$ into the black box and observe the next state of the system: $\vec{x}[i+1] = \begin{bmatrix} x_1[i+1] & x_2[i+1] \end{bmatrix}^\top$.
**Write scalar equations for the new states, $x_1[i+1]$ and $x_2[i+1]$.** Write these equations in terms of the $a_i$, $b_i$, the states $x_1[i]$, $x_2[i]$ and the input $u[i]$. Here, assume that $\vec{w}[i] = \vec{0}$ (i.e. the model is perfect).
**Solution:**

$$x_1[i+1] = a_1x_1[i] + a_2x_2[i] + b_1u[i] \tag{19}$$
$$x_2[i+1] = a_3x_1[i] + a_4x_2[i] + b_2u[i]. \tag{20}$$

(b) Now we want to identify the system parameters. We observe the system at the start state $\vec{x}[0] = \begin{bmatrix} x_1[0] \\ x_2[0] \end{bmatrix}$. We can then input $u[0]$ and observe the next state $\vec{x}[1] = \begin{bmatrix} x_1[1] \\ x_2[1] \end{bmatrix}$. We can continue this process for a sequence of $\ell$ inputs.
Let us define an $\ell$-length trajectory to include an initial condition $\vec{x}[0]$, an input sequence $u[0], \ldots, u[\ell-1]$, and the corresponding states that are produced by the system $x[1], \ldots, x[\ell]$. **Assuming that the model is perfect ($\vec{w}[i] = \vec{0}$), what is the minimum value of $\ell$ you need to identify the system parameters?**
**Solution:** There are 6 unknowns so we need 6 equations to properly identify the system. Each additional timestep gives two new equations. To form the 6 equations we need to give the black box $\ell = 3$ inputs. Namely, given inputs $u[0], u[1]$, and $u[2]$, we can see the state at times $t = 0, 1, 2, 3$ to give us our six equations.
Notice that the initial condition on its own gives us no equations because the unknowns we are interested in do not impact the initial condition. They govern the evolution of the system, and hence the states at times $1, 2, 3$ each give us two equations.
Note that having 6 equations is a necessary, but not sufficient, condition for us to be able to invert the system to uniquely determine the system parameters. For example, if $A = I$ and $u[0] = \cdots = u[\ell-1] = 0$, then we would only have two independent equations.

(c) We now remove our assumption that $\vec{w} = 0$ (that the model is perfect). We assume that $\vec{w}$ is small, so the model is approximately correct and we have

$$\vec{x}[i+1] \approx A\vec{x}[i] + Bu[i]. \tag{21}$$

Say that we feed in a total of 4 inputs $u[0], \ldots, u[3]$, and observe the states $\vec{x}[0], \ldots, \vec{x}[4]$. To identify the system, we need to set up an approximate matrix equation (because of potential small disturbances)

$$DP \approx S \tag{22}$$

using the observed values above and the unknown parameters we want to find. Let our parameter vector be

$$P := \begin{bmatrix} \vec{p}_1 & \vec{p}_2 \end{bmatrix} = \begin{bmatrix} a_1 & a_3 \\ a_2 & a_4 \\ b_1 & b_2 \end{bmatrix} \tag{23}$$

**Find the corresponding $D$ and $S$ to do system identification. Write both out explicitly.**

**Solution:**

Using eq. (20), we get

$$\begin{bmatrix} x_1[0] & x_2[0] & u[0] \\ x_1[1] & x_2[1] & u[1] \\ x_1[2] & x_2[2] & u[2] \\ x_1[3] & x_2[3] & u[3] \end{bmatrix} \begin{bmatrix} a_1 & a_3 \\ a_2 & a_4 \\ b_1 & b_2 \end{bmatrix} \approx \begin{bmatrix} x_1[1] & x_2[1] \\ x_1[2] & x_2[2] \\ x_1[3] & x_2[3] \\ x_1[4] & x_2[4] \end{bmatrix} \tag{24}$$

so

$$D = \begin{bmatrix} x_1[0] & x_2[0] & u[0] \\ x_1[1] & x_2[1] & u[1] \\ x_1[2] & x_2[2] & u[2] \\ x_1[3] & x_2[3] & u[3] \end{bmatrix}, \text{ and } S = \begin{bmatrix} x_1[1] & x_2[1] \\ x_1[2] & x_2[2] \\ x_1[3] & x_2[3] \\ x_1[4] & x_2[4] \end{bmatrix}. \tag{25}$$

(d) Now that we have set up $DP \approx S$, we can estimate $a_0, a_1, a_2, a_3, b_0$, and $b_1$. **Give an expression for the estimates of $\vec{p}_1$ and $\vec{p}_2$ (which are denoted $\hat{\vec{p}}_1$ and $\hat{\vec{p}}_2$ respectively) in terms of $D$ and $S$.** Denote the columns of $S$ as $\vec{s}_1$ and $\vec{s}_2$, so we have $S = [\vec{s}_1 \ \vec{s}_2]$. Assume that the columns of $D$ are linearly independent.

*(HINT: Don't forget that D is not a square matrix. It is taller than it is wide.)*

*(HINT: Can we split $DP = S$ into separate equations for $\vec{p}_1$ and $\vec{p}_2$?)*

**Solution:**

Notice that eq. (24) can be split into two matrix equations, one for each of $p_1$ and $p_2$:

$$D\vec{p}_1 \approx \vec{s}_1 \tag{26}$$

$$D\vec{p}_2 \approx \vec{s}_2. \tag{27}$$

Since $D$ isn't square, it isn't invertible. However, we can still find $\vec{p}_1$ and $\vec{p}_2$ that best satisfy the equation via least-squares, which gives the solution

$$\hat{\vec{p}}_1 = (D^\top D)^{-1} D^\top \vec{s}_1 \tag{28}$$

$$\hat{\vec{p}}_2 = (D^\top D)^{-1} D^\top \vec{s}_2. \tag{29}$$

Here, $D^\top D$ is invertible (i.e. the solution is well-defined) because the columns of $D$ are linearly independent. This was proved in 16A, but for completeness we include it here.

Assume that the columns of $D$ are linearly independent. Let $\vec{v} \in \mathbb{R}^3$ such that $(D^\top D)\vec{v} = 0$. Then $0 = \vec{v}^\top D^\top D\vec{v} = (D\vec{v})^\top (D\vec{v}) = \|D\vec{v}\|_2^2$, so $D\vec{v} = 0$. Since $D$ has linearly independent columns, then $\vec{v} = 0$. This means that the nullspace of $D^\top D$ is $\{0\}$, so $D^\top D$ must have full rank and is invertible.

**4. Identifying Systems from their Responses to Known Inputs**

In many problems, we have an unknown system, and would like to characterize it. One of the ways of doing so is to observe the system response with different initial conditions (or inputs). This problem is also called system identification. It is a prototypical example of a problem that is today called machine learning — inferring an underlying pattern from data, and doing so well enough to be able to exploit that pattern in some practical setting. Go through the attached Jupyter notebook `demo_system_id.ipynb` and answer the following questions.

(a) In Example 2, we assume that instead of measuring the state $\vec{x}$, we are instead measuring a transformation of the state $\vec{y} = T\vec{x}$ where $T$ is a full rank matrix. Assume that we perform system ID on our observations $\vec{y}[i]$ to recover $A_y$, $B_y$ such that $\vec{y}[i+1] = A_y\vec{y}[i] + B_y u[i]$. **How do the identified $A_y$ and $B_y$ matrices relate to the original $A$ and $B$ matrices in the dynamics of $\vec{x}$?** Remember that our original state dynamics are $\vec{x}[i+1] = A\vec{x}[i] + B\vec{u}[i]$. (The answer is given in the Jupyter notebook but remember to show your work.)

**Solution:** Using our given transformation that $\vec{y} = T\vec{x}$,

$$\vec{y}[i+1] = A_y\vec{y}[i] + B_y u[i] \tag{30}$$
$$T\vec{x}[i+1] = A_y T\vec{x}[i] + B_y u[i] \tag{31}$$
$$\vec{x}[i+1] = T^{-1}A_y T\vec{x}[i] + T^{-1}B_y u[i] \tag{32}$$

Thus, $A = T^{-1}A_y T$ and $B = T^{-1}B_y$. This can be rewritten as $A_y = TAT^{-1}$ and $B_y = TB$, which is exactly what is in the Jupyter notebook.

(b) **Please share your observations on Example 2. Comment on what impact a linear transformation of the state trace has on our ability to perform system identification.**

**Solution:** It's nice to see that a linear transformation of the state trace does not have a tremendous effect on our ability to perform system identification. Basically, this means that we have some leeway in choosing what data to observe in practice. In a circuit, for example, we would typically choose capacitor voltages and inductor currents as our state variables. However, it's difficult to make current measurements in real time in a non-invasive way, so we would prefer for our observations to consist of only voltages. As long as we can in principle recover the inductor currents as some linear combination of voltages (this is usually possible), then we can just measure those voltages and proceed as normal.

Furthermore, even if an unwanted state transformation occurs, we know that our estimate of the system does not change drastically. As we saw, the estimated eigenvalues are still correct. Furthermore, controllability and observability are preserved in coordinate changes, so the system we identified will have almost all of the same control-theoretic properties as the true system. Believe me, that's a relief!

(c) **Prove that for any full rank transformation matrix $T$, the eigenvalues of $A_y$ and $A$ from part (a) are the same.**

**Solution:** Assume that the eigenvalue eigenvector pairs of $A$ are $(\lambda_1, \vec{v}_1), (\lambda_2, \vec{v}_2), \ldots, (\lambda_n, \vec{v}_n)$. Then, $A\vec{v}_i = \lambda_i\vec{v}_i$. We claim that $T\vec{v}_i$ will be the eigenvectors of $A_y$. We can see this with

$$A_y T\vec{v}_i = TAT^{-1}T\vec{v}_i = TA\vec{v}_i = T(\lambda_i\vec{v}_i) \tag{33}$$

$$= \lambda_i T \vec{v}_i \tag{34}$$

Thus, $A_y$ also has eigenvalues $\lambda_i$ with its corresponding eigenvector being $T\vec{v}_i$.

(d) **Please share your observations on Example 3. Comment on the impact that changing the noise magnitude, number of samples and number of states has on the system identification performance.**

**Solution:** From playing around with the system and state trace parameters, you may have noticed a few trends:

- Increasing the noise magnitude $\sigma$ reduced the accuracy of the identified eigenvalues (that is, the identified eigenvalues were farther away from the true ones).
- Increasing the number of samples improved the accuracy of the identification.
- Increasing the number of states has a large impact on how accurate the identification is, for a fixed noise magnitude and number of data points.

The final point is important in practice. It suggests that, if we have some control over how many states we model a system with, and if all other things are equal, then we should choose to have fewer states rather than more, so that our system identification requires less data to be accurate.

This is a point that turns out to be extremely important in machine-learning more generally — we do not necessarily always want the most complicated model.

(e) **Please share your observations on Example 4. Comment on how important the model size is for this setting.**

**Solution:** This is another example that takes you a bit beyond what you have seen in lecture, but in a natural way. What happens if your model size is wrong? Do you have to get the model size right?

It's interesting to see that the effect of many eigenvalues near the origin can be effectively approximated by just a couple of eigenvalues in a smaller system. Just like example 3, this suggests that there is such a thing as "too many states". Basically, 13 out of the 16 states of this system contribute almost nothing to the behavior of the system– we were able to throw them away and still capture the important behavior.

So what happens because we are ignoring these states and the true complexity of the underlying model? We will hopefully see in a later homework that what this does is contribute to the disturbance that our estimated model experiences. It not only has a disturbance because it didn't estimate the parameters of the model perfectly (say because of observation noises), but also because it chose a model that was simpler than reality. But as long as we can be robust to this disturbance, we are still fine.

**5. (Lab) Motor Driver and System Identification**

In the lab project, you will be designing SIXT33N, a mischievous little robot who *might* just do what you want — if you design it correctly. To control the car, we need to build a model of the car first. Instead of designing a complex nonlinear model, we will approximate the system with a linear model to work for small perturbations around an equilibrium point. The following model applies separately to each wheel (and associated motor) of the car:

$$v_L[i] = \theta_L u_L[i] - \beta_L \tag{35}$$

$$v_R[i] = \theta_R u_R[i] - \beta_R \tag{36}$$

Notice that this particular model has no state variables since we are measuring velocity directly here.

To do system ID, we decide to use the exact same input $u_L[i] = u_R[i] = u[i]$ for both motors. We measure both velocities, however.

Meet the variables at play in this model:

- $i$ - The current timestep of the model. Since we model the car as a discrete-time system, $i$ will advance by 1 for every new sample in the system.
- $v_L[i]$ - The discrete-time velocity (in units of ticks/timestep) of the left wheel, reading from the motor.
- $v_R[i]$ - The discrete-time velocity (in units of ticks/timestep) of the right wheel, reading from the motor.
- $u[i]$ - The input to each wheel, which takes a value in $[0, 255]$ representing the duty cycle. For example, when $u[i] = 255$, the duty cycle is 100 %, and the motor controller just delivers a constant signal at the system's HIGH voltage, delivering the maximum possible power to the motor. When $u[i] = 0$, the duty cycle is 0 %, and the motor controller delivers 0 V. The duty cycle (D) can be written as
$$\text{duty cycle (D)} = \frac{u[i]}{255} \tag{37}$$
- $\theta(\theta_L, \theta_R)$ - Relates change in input to change in velocity. **Its units are ticks/(timestep · duty cycle).** Since our model is linear, we assume that $\theta$ is the same for every unit increase in $u[i]$ . This is empirically measured using the car. **You will have a separate $\theta$ for your left and your right wheel$(\theta_L, \theta_R)$.**
- $\beta(\beta_L, \beta_R)$ - Similarly to $\theta$, $\beta$ is dependent upon many physical phenomena, so we will empirically determine it using the car. $\beta$ represents a constant offset in the velocity of the wheel, and hence **its units are ticks/timestep**. Note that you will also typically have a different $\beta$ for your left and your right wheel (i.e. $\beta_L \neq \beta_R$). **These $\beta_L$ and $\beta_R$ are different from the $\beta_F$ of the transistor.**

(a) By measuring the car with a PWM signal at different duty cycles, we can collect the velocity data of the left and right wheel, as shown in the following table:

**Table 1:** The velocity of the left and the right wheel at different duty cycles of PWM signal

| Duty Cycle $\times 255$ ($u[i]$) | Velocity of the left wheel ($v_L[i]$) | Velocity of the right wheel ($v_R[i]$) |
|---|---|---|
| 80 | 147 | 127 |
| 120 | 218 | 187 |
| 160 | 294 | 253 |
| 200 | 370 | 317 |

Since the same input is applied to both the wheels, we can take advantage of the same "horizontal stacking" trick you've seen before to be able to reuse computation. To identify the system, we need to set up matrix equations for the left and right wheels in the form of:

$$D_{\text{data}} P \approx S \tag{38}$$

where $P = \begin{bmatrix} \theta_L & \theta_R \\ \beta_L & \beta_R \end{bmatrix}$. **Find the matrix $D_{\text{data}}$ and matrix $S$ needed to perform system identification to get the matrix of parameters of the left and right wheel, $P$.**

**Solution:** We can relate our data points in our table using the model equations eqs. (35) and (36). While we do not know the time indices, we can call them $i_1$ to $i_4$. We start with the left wheel speed data. Approximate equality is used due to the possible presence of noise or unmodelled behavior.

$$v_L[i_1] \approx \theta_L u[i_1] - \beta_L \tag{39}$$
$$v_L[i_2] \approx \theta_L u[i_2] - \beta_L \tag{40}$$
$$v_L[i_3] \approx \theta_L u[i_3] - \beta_L \tag{41}$$
$$v_L[i_4] \approx \theta_L u[i_4] - \beta_L \tag{42}$$

Putting the left and right hand sides into vectors, we get the following equation.

$$\underbrace{\begin{bmatrix} v_L[i_1] \\ v_L[i_2] \\ v_L[i_3] \\ v_L[i_4] \end{bmatrix}}_{\vec{s}_L} \approx \begin{bmatrix} \theta_L u[i_1] - \beta_L \\ \theta_L u[i_2] - \beta_L \\ \theta_L u[i_3] - \beta_L \\ \theta_L u[i_4] - \beta_L \end{bmatrix} = \begin{bmatrix} u[i_1] & -1 \\ u[i_2] & -1 \\ u[i_3] & -1 \\ u[i_4] & -1 \end{bmatrix} \underbrace{\begin{bmatrix} \theta_L \\ \beta_L \end{bmatrix}}_{\vec{p}_L} \tag{43}$$

Here, $\vec{s}_L$ is defined as the vector of measured wheel velocities, and $\vec{p}_L$ is the vector of parameters for the left wheel. We will have corresponding vectors $\vec{s}_R$ and $\vec{p}_R$ for the right wheel when writing a similar set of equations with $v_R[i]$, $\theta_R$, and $\beta_R$. These equations utilize the same values of $u[i]$.

$$\underbrace{\begin{bmatrix} v_R[i_1] \\ v_R[i_2] \\ v_R[i_3] \\ v_R[i_4] \end{bmatrix}}_{\vec{s}_R} \approx \begin{bmatrix} u[i_1] & -1 \\ u[i_2] & -1 \\ u[i_3] & -1 \\ u[i_4] & -1 \end{bmatrix} \underbrace{\begin{bmatrix} \theta_R \\ \beta_R \end{bmatrix}}_{\vec{p}_R} \tag{44}$$

Since the matrix multiplying the parameter vectors are the same, we can concatenate $\vec{s}_L$, $\vec{s}_R$ and $\vec{p}_L$, $\vec{p}_R$ horizontally in the following way.

$$\underbrace{\begin{bmatrix} | & | \\ \vec{s}_L & \vec{s}_R \\ | & | \end{bmatrix}}_{S} \approx \underbrace{\begin{bmatrix} u[i_1] & -1 \\ u[i_2] & -1 \\ u[i_3] & -1 \\ u[i_4] & -1 \end{bmatrix}}_{D_{\text{data}}} \underbrace{\begin{bmatrix} | & | \\ \vec{p}_L & \vec{p}_R \\ | & | \end{bmatrix}}_{P} \tag{45}$$

Substituting in the numerical values we have our $D_{\text{data}}$ and $S$ matrices.

$$D_{\text{data}} = \begin{bmatrix} 80 & -1 \\ 120 & -1 \\ 160 & -1 \\ 200 & -1 \end{bmatrix} \qquad S = \begin{bmatrix} 147 & 127 \\ 218 & 187 \\ 294 & 253 \\ 370 & 317 \end{bmatrix} \tag{46}$$

(b) **Solve the matrix equation $D_{\text{data}}P \approx S$ with least squares to find $\theta_L$, $\theta_R$, $\beta_L$, and $\beta_R$.** You may use a jupyter notebook or other tool for computation.

**Solution:** Our estimate of our parameter matrix is given by $\widehat{P} = \left(D_{\text{data}}^\top D_{\text{data}}\right)^{-1} D_{\text{data}}^\top S$. Using a computing tool, we can compute the matrix product or use a built-in least squares function (e.g. NumpPy's `np.linalg.lstsq`). We get the following parameter value estimates: $\widehat{\theta}_L = 1.862$, $\widehat{\theta}_R = 1.59$, $\widehat{\beta}_L = 3.5$, and $\widehat{\beta}_R = 1.6$.

(c) In order for the car to drive straight, the wheels must be moving at the same velocity. However, the data from Table 1 tells us that the two motors do not run at the same velocity if the duty cycles of driving PWM signals are the same. **Based on the model you extracted, if we want the car to drive straight and $u_L = 100$, what should $u_R$ be?**

**Solution:** To have the car drive straight, we want to find $u_R$ satisfying $v_L = v_R$ for $u_L = 100$. From our identified constants, we can write the following equations.

$$v_L = 1.862u_L - 3.5 \tag{47}$$

$$v_R = 1.59u_R - 1.6 \tag{48}$$

Setting $v_L$ and $v_R$ equal to each other with $u_L = 100$, we get the following value of $u_R$.

$$1.862(100) - 3.5 = 1.59u_R - 1.6 \implies u_R \approx 115.91 \approx 116 \tag{49}$$

**Contributors:**

- Nikhil Shinde.
- Ashwin Vangipuram.
- Sally Hui.
- Alex Devonport.
- Tanmay Gautam.
- Bozhi Yin.
- Kaitlyn Chan.
- Yi-Hsuan Shih.
- Vladimir Stojanovic.
- Moses Won.