This homework is due on Tuesday, July 28, 2020, at 11:59PM.
Self-grades are due on Tuesday, August 4, 2020, at 11:59PM.

1 Controllability in circuits

Consider the circuit in Figure 1, where V_s is an input we can control:

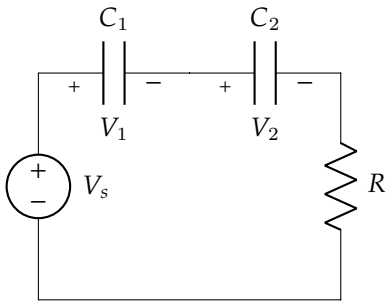


Figure 1: Controllability in circuits

a) Write a state space model for this circuit with V_1 and V_2 as the state variables.

Solution

$$I = \frac{V_s - V_1 - V_2}{R} = C_1 \frac{dV_1}{dt} = C_2 \frac{dV_2}{dt}$$

$$\frac{d}{dt}\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{RC_1} & -\frac{1}{RC_1} \\ -\frac{1}{RC_2} & -\frac{1}{RC_2} \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{RC_1} \\ \frac{1}{RC_2} \end{bmatrix} V_s$$

b) Show that this system is not controllable.

Solution

If we calculate AB, we find that it is a linear combination of B:

$$AB = \begin{bmatrix} -\frac{1}{RC_1}(\frac{1}{RC_1} + \frac{1}{RC_2}) \\ -\frac{1}{RC_2}(\frac{1}{RC_1} + \frac{1}{RC_2}) \end{bmatrix} = -(\frac{1}{RC_1} + \frac{1}{RC_2})B$$

This means that the controllability matrix

$$\begin{bmatrix} B & AB \end{bmatrix}$$

Must have rank of 1. Therefore, this system is not controllable.

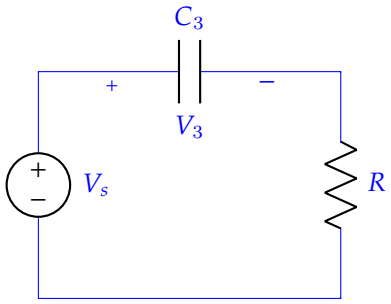
c) Explain, in terms of circuit currents and voltages, why this system isn't controllable. (Hint: think about what currents/voltages of the circuit we are controlling with V_s)

Solution

We can only control V_s, which in turn controls the amount of current flowing through the circuit. Since this current is equal through both capacitors and current directly affects the voltage across a capacitor, there is no way to individually control the voltages across the capacitors.

d) Draw an equivalent circuit of this system that is controllable. What quantity can you control in this system?

Solution



We can control V_3 in this circuit.

2 Controllability in 2D

Consider the control of some two-dimensional linear discrete-time system

$$\vec{x}(k+1) = A\vec{x}(k) + Bu(k)$$

where A is a 2×2 real matrix and B is a 2×1 real vector.

a) Let $A = \begin{bmatrix} a & 0 \\ c & d \end{bmatrix}$ with $a, c, d \neq 0$, and $B = \begin{bmatrix} f \\ g \end{bmatrix}$. Find a B such that the system is controllable no matter what nonzero values a, c, d take on, and a B for which it is not controllable no matter what nonzero values are given for a, c, d. You can use the controllability rank test, but please explain your intuition as well.

Solution

With $B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, the system is controllable for all nonzeros a, c, d, because $[B, AB] = \begin{bmatrix} 1 & a \\ 0 & c \end{bmatrix}$, which has full rank. With $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ the system is not controllable because $[B, AB] = \begin{bmatrix} 0 & 0 \\ 1 & d \end{bmatrix}$, which only has rank=1. The intuition is that, due to the zero entry in A, the state x_1 evolves autonomously, i.e., $\frac{d}{dt}x_1(t) = ax_1(t)$, hence it needs to be controlled by some input f. On the other hand we can control x_2 via controlling x_1, as $\frac{d}{dt}x_2(t) = cx_1(t) + dx_2(t)$, which implies that x_2 can be "tuned" by manipulating x_1.

b) Let $A = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$ with $a, d \neq 0$. and $B = \begin{bmatrix} f \\ g \end{bmatrix}$ with $f, g \neq 0$. Is this system always controllable? If not, find configurations of nonzero a, d, f, g that make the system uncontrollable.

Solution

No. uncontrollable when $a = d$. In this case the matrix is just a constant a times the identity. So when you check with the controllability test, AB is just a scalar multiple of B and hence linearly dependent. The intuition is that the two states are inherently "coupled" as two eigenvalues are the same. Any control input can only move the states along a line hence the states cannot reach arbitrary points in $\mathbb{R}^2$.

c) We want to see if controllability is preserved under changes of coordinates. To begin with, let $\vec{z}(k) = V^{-1}\vec{x}(k)$, please write out the system equation with respect to $\vec{z}$.

Solution

$\vec{x}(k) = V\vec{z}(k)$, hence we have

$$V\vec{z}(k+1) = AV\vec{z}(k) + Bu(k)$$

$$\vec{z}(k+1) = V^{-1}AV\vec{z}(k) + V^{-1}Bu(k)$$

.

d) Now show that controllability is preserved under change of coordinates. (Hint: use the fact that $\text{rank}(MA) = \text{rank}(A)$ for any invertible matrix M.)

Solution

The matrix whose rank needs to be tested after the coordinate change is $[V^{-1}B, V^{-1}AVV^{-1}B] = [V^{-1}B, V^{-1}AB] = V^{-1}[B, AB]$ which has the same rank as $[B, AB]$, since V by assumption is full rank.

3 Controllability and discretization

In this problem, we will use the car model

$$\frac{d}{dt}x(t) = v(t)$$
$$\frac{d}{dt}v(t) = u(t)$$

that was discussed in class.

a) Assuming that the input $u(t)$ can be varied continuously, is this system controllable?

Solution

Introducing states $x_1 = x$ and $x_2 = v$, we rewrite this system in state space form

$$\frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + Bu(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t).$$

The controllability matrix

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

has rank 2. Therefore the continuous-time system is controllable.

b) Now assume that we can only change our control input every T seconds. Derive a discrete-time state space model for the state updates, assuming that the input is held constant between times nT and $nT + T$.

Solution

To discretize this, we first solve the differential equation for $v(t)$ for interval $t \in [nT, nT + T)$

$$\int_{nT}^{t} dv = \int_{nT}^{t} u(nT)\,\mathrm{d}\tau$$
$$v(t) - v(nT) = (t - nT)u(nT)$$

Now we substitute the expression for $v(t)$ into the first differential equation and solve for $x(t)$

$$\int_{nT}^{t} dx = \int_{nT}^{t} \big(v(nT) + (\tau - nT)u(nT)\big)\,\mathrm{d}\tau$$

Change the variable of integration to $s = \tau - nT$:

$$x(t) - x(nT) = \int_{0}^{t-nT} \big(v(nT) + u(nT) \cdot s\big)\,\mathrm{d}s$$
$$= v(nT)(t - nT) + \frac{u(nT)}{2}(t - nT)^2$$

Lastly, we evaluate both states at time $t = nT + T$ to get the discretized model

$$x(nT + T) = x(nT) + Tv(t) + \frac{T^2}{2}u(nT)$$
$$v(nT + T) = v(nT) + Tu(nT)$$
$$\begin{bmatrix} x[n+1] \\ v[n+1] \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x[n] \\ v[n] \end{bmatrix} + \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} u[n]$$

c) Is the discrete-time system controllable?

Solution

The controllability matrix

$$C = \begin{bmatrix} B & AB \end{bmatrix} = \begin{bmatrix} \frac{1}{2}T^2 & \frac{3}{2}T^2 \\ T & T \end{bmatrix}$$

has nonzero determinant $\det(C) = -T^3$ since $T > 0$. Therefore, the controllability matrix has rank 2 and the discretized system is controllable.

4 The Moore-Penrose pseudoinverse for "wide" matrices

Say we have a set of linear equations given by $A\vec{x} = \vec{y}$. If A is invertible, we know that the solution for $\vec{x}$ is $\vec{x} = A^{-1}\vec{y}$.

However, what if A is not a square matrix? In 16A, you saw how to set up a least squares problem for tall matrices A which gave us a reasonable answer that asks for the "best" match in terms of reducing the norm of the error vector.

Now we will consider the case where A is a wide matrix with more columns than rows. To solve this system, we will walk you through the **Moore-Penrose pseudoinverse** that generalizes the idea of the matrix inverse and is derived from the singular value decomposition.

To find the Moore-Penrose pseudoinverse we start by calculating the SVD of A.

a) Say you have the matrix

$$A = \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}.$$

Calculate U, Σ, V such that

$$A = U\Sigma V^T \tag{1}$$

Solution

$$AA^\top = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \qquad \lambda^2 - 6\lambda + 8 = 0 \implies \lambda = 4, 2$$

Solving $A\vec{v} = \lambda_i \vec{v}$ produces eigenvectors $[\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}]^\top$ and $[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^\top$ associated with eigenvalues 4 and 2 respectively. The singular values are the square roots of the eigenvalues of $AA^\top$, so

$$\Sigma = \begin{bmatrix} 2 & 0 & 0 \\ 0 & \sqrt{2} & 0 \end{bmatrix}$$

and

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

We can then solve for the $\vec{v}$ vectors using $A^\top \vec{u}_i = \sigma_i \vec{v}_i$, producing $\vec{v}_1 = [0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^\top$ and $\vec{v}_2 = [1, 0, 0]^\top$. The last $\vec{v}$ must be orthonormal to the other two, so we can pick $[0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^\top$.

The SVD is:

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

Let us now think about what the SVD does. The matrix A acts on a vector $\vec{x}$ to give the result $\vec{y}$. We have

$$A\vec{x} = U\Sigma V^T \vec{x} = \vec{y}.$$

Observe that V^T rotates the vector, Σ scales the result, and U rotates it again. We will try to "reverse" these operations one at a time and then put them together to construct the Moore-Penrose pseudoinverse.

If U "rotates" the vector $\left(\Sigma V^T\right)\vec{x}$, what operator will undo the rotation?

Solution

By orthonormality, we know that $U^TU = UU^T = I$. Therefore, U^T undoes the rotation.

b) **Derive a matrix that will "unscale", or undo the effect of Σ where it is possible to undo.** Recall that Σ has the same dimensions as A. Ignore any division by zeros (that is to say, let it stay zero).

Solution

If you observe the equation:

$$\Sigma\vec{z} = U^T\vec{y} = \vec{w}, \tag{2}$$

you can see that $\sigma_i z_i = w_i$ for $i = 1, \ldots, k$ if A is rank k. This means we can recover z_i by dividing w_i by σ_i. For $i > k$, since $\sigma_i = 0$, there is no way of recovering z_i.

To do some dimension accounting, Σ is an $m \times n$ matrix and $\vec{w}$ is a vector in $\mathbb{R}^m$ so in order to "unscale" so that $\vec{z} = \Sigma^\dagger U^T\vec{y}$, we would need $\Sigma^\dagger$ to be an $n \times m$ matrix.

Therefore, the best we can do is pad 0s below Σ_c^{-1} where Σ_c represents the singular values from the compact SVD.

$$\text{If } \Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & \ddots & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ldots & 0 \\ 0 & 0 & 0 & \sigma_m & 0 & \ldots & 0 \end{bmatrix} \text{ then } \Sigma^\dagger = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \ldots & 0 \\ \vdots & \ddots & \ldots & 0 \\ 0 & \cdots & \frac{1}{\sigma_k} & \vdots \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix}$$

This $\Sigma^\dagger$ matrix undoes the effect of Σ where that effect can be undone.

c) **Derive an operator that would "unrotate" by V^T.**

Solution

By orthonormality, we know that $V^TV = VV^T = I$. Therefore, V undoes the rotation.

d) **Try to use this idea of "unrotating" and "unscaling" to derive an "inverse", denoted as $A^\dagger$.** That is to say,

$$\vec{x} = A^\dagger\vec{y}$$

The reason why the word inverse is in quotes (or why this is called a pseudo-inverse) is because we're ignoring the "divisions" by zero.

Solution

We can use the unrotation and unscaling matrices we derived above to "undo" the effect of A and get the required solution. Of course, nothing can possibly be done for the information that was destroyed by the nullspace of A — there is no way to recover any component of the true $\vec{x}$ that was in the nullspace of A. However, we can get back everything else.

$$\begin{aligned} \vec{y} &= A\vec{x} = U\Sigma V^T\vec{x} & \\ U^T\vec{y} &= \Sigma V^T\vec{x} & \text{Unrotating by } U \\ \Sigma^\dagger U^T\vec{y} &= V^T\vec{x} & \text{Unscaling by } \Sigma^\dagger \\ V\Sigma^\dagger U^T\vec{y} &= \vec{x} & \text{Unrotating by } V \end{aligned}$$

Therefore, we have $A^\dagger = V\Sigma^\dagger U^T$, where $\Sigma^\dagger$ is given in (c).

e) **Use $A^\dagger$ to solve for a vector $\vec{x}$ in the following system of equations.**

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix} \vec{x} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

Feel free to use the SVD of A from part (a) and numpy.

Solution

From the above, we have the solution given by:

$$\begin{aligned} \vec{x} &= A^\dagger \vec{y} = V\Sigma^\dagger U^T \vec{y} \\ &= \begin{bmatrix} 0 & 1 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2 \\ 4 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} \end{aligned}$$

Therefore, a reasonable solution to the system of equations is:

$$\vec{x} = \begin{bmatrix} 3 \\ \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}$$

We will now show that $\vec{x} = A^\dagger \vec{y}$ satisfies the minimality property that $\|\vec{x}\| \leq \|\vec{z}\|$ for all other vectors $\vec{z}$ satisfying $A\vec{z} = \vec{y}$. To do this, suppose A is a generic $m \times n$ matrix of rank $m < n$ and let us look at the coordinates of $\vec{x}$ in the V basis.

f) **Find the coordinates of $\vec{x}$ in the basis formed by the columns of V.**

Solution

Since $\vec{x}$ is the pseudo-inverse solution, we know that

$$\vec{x} = A^\dagger \vec{y} = V\Sigma^\dagger U^T \vec{y}$$

To write down $\vec{x}$ using the V basis, recall that the matrix V^{-1} takes us from standard coordinates to V basis coordinates.

$$\begin{aligned} \vec{x}\big|_V &= V^{-1}\vec{x} = V^{-1}V\Sigma^\dagger U^T \vec{y} = \Sigma^\dagger U^T \vec{y} \\ &= \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\sigma_m} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} - & \vec{u}_1^T \vec{y} & - \\ & \vdots & \\ - & \vec{u}_m^T \vec{y} & - \end{bmatrix} \\ &= \begin{bmatrix} \frac{\vec{u}_1^T \vec{y}}{\sigma_1}, \frac{\vec{u}_2^T \vec{y}}{\sigma_2}, \dots, \frac{\vec{u}_m^T \vec{y}}{\sigma_m}, 0, \dots, 0 \end{bmatrix}^T \end{aligned}$$

The $n - m$ zeros at the end come from the fact that there are only m non-zero singular values. Therefore, by construction, we see that $\vec{z}$ is a linear combination of the first m columns of V.

g) Now let $\vec{z}$ be a solution to $A\vec{z} = \vec{y}$. Write out the SVD of A and undo the rotation of U. If $\vec{z}|_V$ is the representation of $\vec{z}$ in the V basis, **what happens to its last $n - m$ entries when left multiplied by Σ? Use this result to show that $\|\vec{x}\| \leq \|\vec{z}\|$.**

Solution

Since any other $\vec{z}$ is also a solution to the original problem, we have

$$A\vec{z} = U\Sigma V^T\vec{z} = U\Sigma\ \vec{z}\big|_V = \vec{y},$$

We can undo the rotation of U to get

$$\Sigma\ \vec{z}\big|_V = U^T\vec{y}$$

When left multiplying $\vec{z}|_V$ by the matrix Σ, the last $n-m$ entries are "lost" by multiplying by the zeros in the Σ matrix. Formally the last $n - m$ columns of V form the null-space of A so we lose all of the coordinates when multiplying by Σ.

$$\Sigma\ \vec{z}\big|_V = \begin{bmatrix}\Sigma_c & 0\end{bmatrix}\begin{bmatrix}V_c^T \\ V_n^T\end{bmatrix}\vec{z} = \Sigma_c V_c^T\vec{z} = U^T\vec{y}$$

The first m entries must be identical to the first m entries of $\vec{x}|_V$ in order to satisfy $A\vec{z} = \vec{y}$ but the last $n - m$ entries can be arbitrary scalars $\alpha_{m+1}, \ldots, \alpha_n$.

$$\vec{z}\big|_V = \left[\frac{\vec{u}_1^T\vec{y}}{\sigma_1}, \frac{\vec{u}_2^T\vec{y}}{\sigma_2}, \ldots, \frac{\vec{u}_m^T\vec{y}}{\sigma_m}, \alpha_{m+1}, \alpha_{m+2}, \ldots, \alpha_n\right]^T$$

Now, since the columns of V are orthonormal, observe that,

$$||\vec{x}||^2 = \sum_{i=1}^{m}\left|\frac{\vec{u}_i^T\vec{y}}{\sigma_i}\right|^2$$

and that,

$$||\vec{z}||^2 = \sum_{i=1}^{m}\left|\frac{\vec{u}_i^T\vec{y}}{\sigma_i}\right|^2 + \sum_{i=m+1}^{n}|\alpha_i|^2$$

Therefore,

$$||\vec{x}||^2 = ||\vec{z}||^2 + \sum_{i=m+1}^{n}|\alpha_i|^2$$

This tells us that,

$$||\vec{x}|| \leq ||\vec{z}||$$

5 Weighted Minimum Norm

From the previous problem, given a wide matrix A, we solved $A\vec{x} = \vec{y}$ such that $\vec{x}$ is a minimum norm solution:

$$\|\vec{x}\| \leq \|\vec{z}\|$$

for all $\vec{z}$ such that $A\vec{z} = \vec{y}$. However, what if you weren't interested in minimizing just the norm of $\vec{x}$? What if you instead cared about minimizing $C\vec{x}$? For example, suppose that controls were more or less costly at different times.

The problem can be written out mathematically as:

$$\min_{\vec{z}\in\mathbb{R}^n} \|C\vec{z}\| \text{ subject to } A\vec{z} = \vec{y} \tag{3}$$

To give a concrete example, we define the following matrices

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 0 \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Let's start with the case of C being invertible. We will reformulate the minimum norm problem into one we already now how to solve.

a) Define the change of basis $\vec{w} = C\vec{z}$. Show that the original minimum norm problem (3) can be reformulated as

$$\min_{\vec{w}\in\mathbb{R}^n} \|\vec{w}\| \text{ subject to } AC^{-1}\vec{w} = \vec{y} \tag{4}$$

Solution

Since $\vec{w} = C\vec{z}$, our objective $\|C\vec{z}\|$ is equivalent to $\|w\|$. In addition, searching over all $\vec{z} \in \mathbb{R}^n$ is equivalent to searching over all $C\vec{z} \in \mathbb{R}^n$ since C is an invertible linear transformation.

Since C is invertible, $\vec{z} = C^{-1}\vec{w}$ and we see that the constraint $A\vec{z} = \vec{y}$ is equivalent to the constraint $AC^{-1}\vec{w} = \vec{y}$.

b) Explain why our new problem is one that we already know how to solve. Then give the optimal solution $\vec{x}$ such that $\|C\vec{x}\| \leq \|C\vec{z}\|$.

Solution

Our new optimization problem is minimizing the norm of a vector $\vec{w}$ over all vectors in $\mathbb{R}^n$ subject to the constraint $B\vec{w} = \vec{y}$ where $B = AC^{-1}$. Therefore, the optimal solution is $\vec{w}^* = B^\dagger\vec{y}$.

We can compute AC^{-1} and its pseudoinverse in numpy by calling

```
np.linalg.pinv(B)
```

$$B = AC^{-1} = \begin{bmatrix} 0 & 0 & 0.5 \\ 0.5 & 1 & 0 \end{bmatrix} \implies B^\dagger = \begin{bmatrix} 0 & 0.4 \\ 0 & 0.8 \\ 2 & 0 \end{bmatrix}$$

Multiplying by the pseudoinverse of B, the optimal solution is

$$\vec{w}^* = B^\dagger\vec{y} = \begin{bmatrix} 0.4 \\ 0.8 \\ 4 \end{bmatrix} \implies \vec{x} = C^{-1}\vec{w}^* = \begin{bmatrix} 2 \\ 0.8 \\ 0.2 \end{bmatrix} \tag{5}$$

c) Now what if C were a tall $p \times n$ matrix with linearly independent columns? This would mean we are no longer able to invert C. Show that the original problem (3) can be reformulated as

$$\min_{\vec{z} \in \mathbb{R}^n} \left\| B\vec{z} \right\| \text{ subject to } A\vec{z} = \vec{y} \tag{6}$$

where B is an invertible matrix. You must also show what B is.

Hint: Try using the compact SVD of C to find what B is.

Solution

We start by expanding out $\left\| C\vec{z} \right\|$ as

$$\left\| C\vec{z} \right\| = \sqrt{\vec{z}^T C^T C \vec{z}} \tag{7}$$

Now if we can find an invertible matrix B such that $B^T B = C^T C$, then we would be able to say that

$$\left\| C\vec{z} \right\| = \sqrt{\vec{z}^T C^T C \vec{z}} = \sqrt{\vec{z}^T B^T B \vec{z}} = \left\| B\vec{z} \right\| \tag{8}$$

If the compact SVD of C is $U_c \Sigma_c V_c^T$, then

$$C^T C = (U_c \Sigma_c V_c^T)^T U_c \Sigma_c V_c^T = V_c \Sigma_c^T \Sigma_c V_c^T \tag{9}$$

Since C has linearly independent columns, the matrices Σ_c and V_c are square and invertible from the compact SVD. Therefore, it follows that $B = \Sigma_c V_c^T$.

d) Let

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 2 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Based on the reformulation from the previous part, solve for $\vec{x}$ such that $\left\| C\vec{x} \right\| \leq \left\| C\vec{z} \right\|$.
Note: You will most likely need to use numpy to do these calculations

Solution

The reformulated problem from the previous part is identical to the one given in part (a). Therefore, we define a change of basis $\vec{w} = B\vec{z}$ and an equivalent problem will be

$$\min_{\vec{w} \in \mathbb{R}^n} \left\| \vec{w} \right\| \text{ subject to } AB^{-1}\vec{w} = \vec{y} \tag{10}$$

Then we can compute $H = AB^{-1} = AV_c\Sigma_c^{-1}$ and its pseudo-inverse $H^\dagger$ in numpy

$$H = \begin{bmatrix} -0.3 & 0.32 & -0.69 \\ -0.22 & -0.92 & -0.33 \end{bmatrix} \implies H^\dagger = \begin{bmatrix} -0.45 & -0.22 \\ 0.48 & -0.92 \\ -1.03 & -0.33 \end{bmatrix} \tag{11}$$

The minimum norm solution will be $\vec{w}^* = H^\dagger \vec{y}$

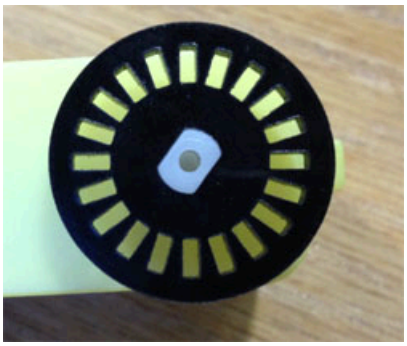
$$\vec{w}^* = H^\dagger \vec{y} = \begin{bmatrix} -1.12 \\ 0.04 \\ -2.4 \end{bmatrix} \tag{12}$$

Lastly, converting our solution back into standard basis coordinates

$$\vec{x} = B^{-1}\vec{w}^* = \begin{bmatrix} 2 \\ 4 \\ -3 \end{bmatrix} \tag{13}$$

6 Understanding the SIXT33N Car Control Model

As we continue along the process of making the SIXT33N cars awesome, we'd like to better understand the car model that we will be using to develop a control scheme. As a wheel on the car turns, there is an encoder disc (see below) that also turns as the wheel turns. The encoder shines a light though the encoder disc, and as the wheel turns, the light is continually blocked and unblocked, allowing the encoder to detect how fast the wheel is turning by looking at the number of times that the light "ticks" between being blocked and unblocked over a specific time interval.



The following model applies separately to each wheel (and associated motor) of the car:

$$v[k] = d[k+1] - d[k] = \theta u[k] - \beta$$

Meet the variables at play in this model:

- k - The current timestep of the model. Since we model the car as a discrete system, this will advance by 1 on every new sample in the system.
- $d[k]$ - The total number of ticks advanced by a given encoder (the values may differ for the left and right motors–think about when this would be the case).
- $v[k]$ - The discrete-time velocity (in units of ticks/timestep) of the wheel, measured by finding the difference between two subsequent tick counts ($d[k+1] - d[k]$).
- $u[k]$ - The input to the system. The motors that apply force to the wheels are driven by an input voltage signal. This voltage is delivered via a technique known as pulse width modulation (PWM), where the average value of the voltage (which is what the motor is responsive to) is controlled by changing the duty cycle of the voltage waveform. The duty cycle, or percentage of the square wave's period for which the square wave is HIGH, is mapped to the range $[0, 255]$. Thus, $u[k]$ takes a value in $[0, 255]$ representing the duty cycle. For example, when $u[k] = 255$, 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 $u[k] = 0$, the duty cycle is 0 %, and the motor controller delivers 0 V.
- θ - Relates change in input to change in velocity: if the wheel rotates through n ticks in one timestep for a given $u[k]$ and m ticks in one timestep for an input of $u[k]+1$, then $\theta = m - n = \frac{\Delta v[k]}{\Delta u[k]} = \frac{v_{u_1[k]}[k] - v_{u_0[k]}[k]}{u_1[k] - u_0[k]}$. **Its units are ticks/(timestep · duty cycle).** Since our model is linear, we assume that θ is the same for every unit increase in $u[k]$. This is empirically measured using the car: θ depends on many physical phenomena, so for the purpose of this class, we will not attempt to create a mathematical model based on the actual physics. However, you can conceptualize θ as a "sensitivity factor",

representing the idiosyncratic response of your wheel and motor to a change in power (you will have a separate θ for your left and your right wheel).

- β - Similarly to θ, β is dependent upon many physical phenomena, so we will empirically determine it using the car. β represents a constant offset in velocity, and hence **its units are ticks/timestep**. Note that you will also have a different β for your left and your right wheel.

In this problem (except parts (c) and (e)) we will assume that the wheel conforms perfectly to this model to get an intuition of how the model works.

a) If we wanted to make the wheel move at a certain target velocity v^*, what input $u[k]$ should we provide to the motor that drives it? Your answer should be symbolic, and in terms of v^*, $u[k]$, θ, and β.

Solution

$$\begin{aligned} v^* &= \theta u[k] - \beta \\ v^* + \beta &= \theta u[k] \\ u[k] &= \frac{v^* + \beta}{\theta} \end{aligned}$$

b) Even if the wheel and the motor driving it conform perfectly to the model, our inputs still limit the range of velocities. Given that $0 \leq u[k] \leq 255$, determine the maximum and minimum velocities possible for the wheel. How can you slow the car down?

Solution

The maximum is $255\theta - \beta$, and the minimum is $0 - \beta = -\beta$.

Since there are no brakes on the wheel, we slow down by reducing the input value and thereby the PWM duty cycle.

c) Our intuition tells us that a wheel on a car should eventually stop turning if we stop applying any power to it. Find $v[k]$ assuming that $u[k] = 0$. Does the model obey your intutition? What does that tell us about our model?

Solution

$$v[k] = -\beta$$

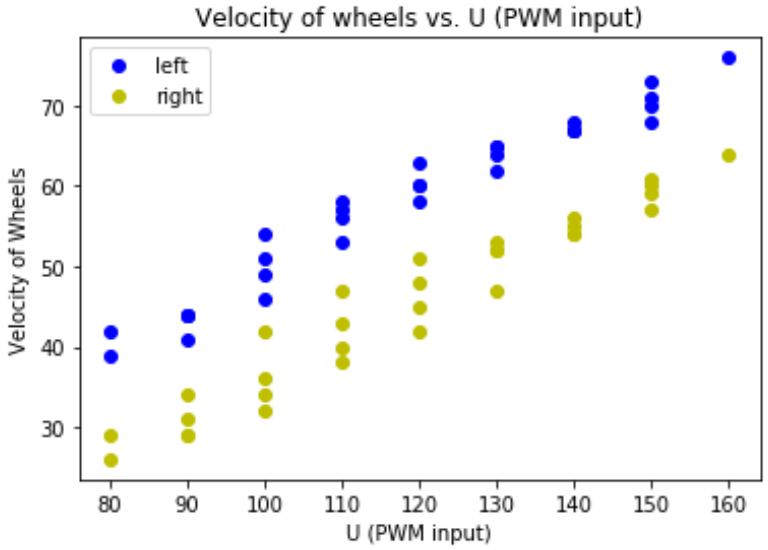
However, our intuition says that the car should have stopped: i.e., $v[k] = 0$. In lab, we will empirically find the value of β over a range of input duty cycles, but our fit does not work very well everywhere and our model does not match the real behavior near $u = 0$. This is a limitation of the simplified empirical model we are using, but as we will see in lab, we can still make the actual car work well over a certain range of inputs.

d) In order to characterize the car, we need to find the θ and β values that model your left and right wheels and their motors: θ_l, β_l, θ_r, and β_r. How would you determine θ and β empirically? What data would you need to collect? *Hint:* keep in mind we also know the input $u[k]$ for all k.

Solution

Given the motor model $v[k] = d[k+1] - d[k] = \theta u[k] - \beta$, we can determine θ and β by plotting velocity ($v[k]$) vs. input ($u[k]$).
By sweeping the input over a reasonable operating range and collecting multiple velocity samples (by differencing the total number of ticks, a collected quantity, between subsequent timesteps), we can collect velocity and input data. Then, we can perform least-squares linear regression on the data to determine the slope θ and y-intercept β.

Least-squares example solution:



Using the least squares linear regression method shown below, we can estimate the slope and y-intercept for each graph.

$$A^T A\hat{\boldsymbol{x}} = A^T \boldsymbol{y} \tag{14}$$

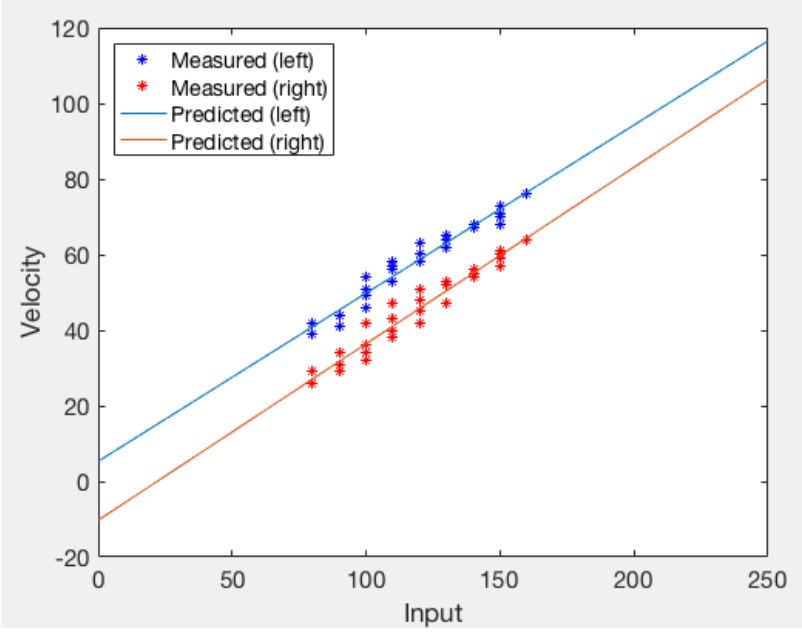
$$A = \begin{bmatrix} u_0 & 1 \\ u_1 & 1 \\ \vdots & \vdots \\ u_n & 1 \end{bmatrix} \tag{15}$$

$$y = \begin{bmatrix} v_{\text{measured},0} \\ v_{\text{measured},1} \\ \vdots \\ v_{\text{measured},n} \end{bmatrix} \tag{16}$$

Using the θ and β values we found above, we now plot the predicted velocities of the left and right motor over the inputs u from 0 to 255:

$$v_{\text{left}} = \theta_{\text{left}} u - \beta_{\text{left}} \tag{17}$$

$$v_{\text{right}} = \theta_{\text{right}} u - \beta_{\text{right}} \tag{18}$$



e) How can you use the data you collected in part (d) to mitigate the effect of the system's nonlinearity and/or minimize model mismatch? *Hint:* you will only wind up using a small range of the possible input values in practice. There are several reasons this is true, but one is that each motor has a characteristic attainable velocity range, and for your car to drive straight, we need the wheels to rotate with the same velocity.

Solution

Since we model the relationship between input and velocity as linear, if the actual data we collect displays nonlinearity, the car's behavior will not match the model well. To minimize model mismatch, we choose an operating point centered within an approximately linear region of the plot, *as long as both motors are capable of attaining a reasonable amount of the surrounding velocities.*

7 Open-Loop Control of SIXT33N

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

$$u(t) = \frac{v^* + \beta}{\theta}$$

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.

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

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 $u(t)$, 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 t_p. In order to model this phenomenon, we can say that $u(t) = 255$ for $t \in [0, t_p - 1]$[1]. In addition, the car initially (at $t = 0$) hasn't moved, so we can also say $d(0) = 0$.

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

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

Thus we get:

$$d_L(t_p) = t_p(255\theta_L - \beta_L)$$
$$d_R(t_p) = t_p(255\theta_R - \beta_R)$$

b) Let us define $\delta(t) = d_L(t) - d_R(t)$ as the difference in positions between the two wheels. If both wheels of the car are going at the same velocity, then this difference 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 $\delta(t_p)$. 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?

Note: Since $d(0) = d_L(0) = d_R(0) = 0$, $\delta(0) = 0$.

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

Solution

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

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 (t_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 $\delta(t \to \infty)$ 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.

$$\begin{aligned}
u_L(t) &= \frac{v^* + \beta_L}{\theta_L} \\
u_R(t) &= \frac{v^* + \beta_R}{\theta_R}
\end{aligned}$$

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

Does $\delta(t \to \infty)$ deviate from δ_0? Why or why not?

Solution

$$\begin{aligned}
\delta(t_p + 1) &= d_L(t_p + 1) - d_R(t_p + 1) \\
&= d_L(t_p) + \theta_L u(t) - \beta_L - (d_R(t_p) + \theta_R u(t) - \beta_R) \\
&= d_L(t_p) + \theta_L u(t) - \beta_L - d_R(t_p) - \theta_R u(t) + \beta_R \\
&= \Big(d_L(t_p) - d_R(t_p)\Big) + \big(\theta_L u(t) - \beta_L\big) - \big(\theta_R u(t) - \beta_R\big) \\
&= \Big(d_L(t_p) - d_R(t_p)\Big) + v^* - v^* \\
&= \Big(d_L(t_p) - d_R(t_p)\Big) \\
&= d(t_p) \\
&= \delta_0 \\
\delta(t) &= \delta_0 \text{ (by generalization: every step does not change } \delta)
\end{aligned}$$

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 $\delta(t)$ 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.

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

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]

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

As before, let $\delta(t_p) = \delta_0$. Find $\delta(t)$ for $t \geq t_p$ 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

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

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.

As $t \to \infty$, 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.

[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.

8 Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student! We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

a) **What sources (if any) did you use as you worked through the homework?**

b) **If you worked with someone on this homework, who did you work with?** List names and student ID's. (In case of homework party, you can also just describe the group.)

c) **How did you work on this homework?** (For example, *I first worked by myself for 2 hours, but got stuck on problem 3, so I went to office hours. Then I went to homework party for a few hours, where I finished the homework.*)

d) **Do you have any feedback on this homework assignment?**

e) **Roughly how many total hours did you work on this homework?**