ME599 Homework 3 - Akshat Dubey

Problem 1.a

Let $L \in \mathbb{R}^{n \times n}$ be the Laplacian of a connected undirected graph with no self-loops. Show that the induced ∞-norm of e^{-L} is unique, that is, vector $v = \mathbf{1}_n$ is the unique solution of

$$\max_{\|v\|_\infty = 1} \|e^{-L}v\|_\infty.$$

Solution

- The infinity norm of a vector v, $\|v\|_\infty$: Maximum absolute value of the largest element in the vector.
- The expression $\max_{\|v\|_\infty=1} \|e^{-L}v\|_\infty$ means: Find the vector v with infinity norm equal to 1, such that the infinity norm of the product $e^{-L}v$ is maximized. Hence, no element in the vector v can be greater than 1 in magnitude, and at least one element must be equal to 1 in magnitude.
- The Laplacian matrix L is used to describe the connectivity of a graph, where each row i describes how node the time derivative of node i is affected by other nodes. If there is no edge between a node i and node j, then $L_{ij} = 0$.

From the notes 4.2 page 6, we know that $L\vec{1} = \vec{0}$, which is similar to the general form of an eigenvalue problem, $Lv = \lambda v$. This suggests that the smallest eigenvalue of L is 0, and the corresponding eigenvector is $\vec{1}$.

Expanding the expression $e^{-L}v$, where $v = \vec{1}$:

$$\begin{aligned} e^{-L}v = e^{-L}\vec{1} &= \left(I - L + \frac{L^2}{2!} - \frac{L^3}{3!} + \dots\right)\vec{1} \\ &= \left(I\vec{1} - L\vec{1} + \frac{L^2}{2!}\vec{1} - \frac{L^3}{3!}\vec{1} + \dots\right) \\ &= \left(\vec{1} - \vec{0} + \vec{0} + \vec{0} + \dots\right) \text{all terms with an } L\vec{1} = 0 \\ &= \vec{1} \end{aligned}$$

Since $\|\vec{1}\|_\infty = 1$, $v = \vec{1}$ is a solution to the optimization problem, where the objective $\|e^{-L}v\|_\infty = 1$.

To prove that it is a unique solution, we need to show that no other vector v' where $\|v'\|_\infty = 1$, can have a larger $\|e^{-L}v\|_\infty$ than 1. From the notes 4.2 page 6, we know that if L describes an unconnected directed graph, e^{-L} is row-stochastic, which means that the sum of the elements in each row is 1. The matrix product $e^{-L}v$ sums up each row of e^{-L} weighted by the corresponding element in v. Let

$$\begin{aligned} e^{-L} &= E \\ v &= [v_1, v_2, \dots, v_n]^T \\ e^{-L}v &= Ev \end{aligned}$$

Then

$$\begin{aligned} e^{-L}v &= \begin{bmatrix} E_{1,1} & E_{1,2} & \dots & E_{1,n} \\ E_{2,1} & E_{2,2} & \dots & E_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ E_{n,1} & E_{n,2} & \dots & E_{n,n} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \\ &= \begin{bmatrix} E_{1,1}v_1 + E_{1,2}v_2 + \dots + E_{1,n}v_n \\ E_{2,1}v_1 + E_{2,2}v_2 + \dots + E_{2,n}v_n \\ \vdots \\ E_{n,1}v_1 + E_{n,2}v_2 + \dots + E_{n,n}v_n \end{bmatrix} \end{aligned}$$

When $v = \vec{1}$,

$$e^{-L}v = \begin{bmatrix} E_{1,1} + E_{1,2} + \ldots + E_{1,n} \\ E_{2,1} + E_{2,2} + \ldots + E_{2,n} \\ \vdots \\ E_{n,1} + E_{n,2} + \ldots + E_{n,n} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

A vector $v' \neq \vec{1}$ that has $\|v'\|_\infty = 1$ will have at least one element < 1. If we have v' such that $v'_i < 1$ for some i, then

$$e^{-L}v' = \begin{bmatrix} E_{1,1}v'_1 + \ldots + E_{1,i}v'_i + \ldots + E_{1,n}v'_n \\ E_{2,1}v'_1 + \ldots + E_{2,i}v'_i + \ldots + E_{2,n}v'_n \\ \vdots \\ E_{n,1}v'_1 + \ldots + E_{n,i}v'_i + \ldots + E_{n,n}v'_n \end{bmatrix} = \begin{bmatrix} E_{1,1} + \ldots + E_{1,i}v'_i + \ldots + E_{1,n} \\ E_{2,1} + \ldots + E_{2,i}v'_i + \ldots + E_{2,n} \\ \vdots \\ E_{n,1} + \ldots + E_{n,i}v'_i + \ldots + E_{n,n} \end{bmatrix}$$

For a matrix X, $\|X\|_\infty = \max_i |X_i|$.

We can compare $e^{-L}\vec{1}$ and $e^{-L}v'$ row by row with subtraction and check if one of them has a greater absolute value. If one row has a greater absolute value, the infinity norm of e^{-L} with the corresponding vector will also be greater. Hence, given a row j of $e^{-L}v'$ and $e^{-L}\vec{1}$, if $|(e^{-L}v')_j| - |(e^{-L}\vec{1})_j| < 0 \quad \forall j \implies \|e^{-L}v'\|_\infty < \|e^{-L}\vec{1}\|_\infty$. Therefore, we check $|(e^{-L}v')_j| - |(e^{-L}\vec{1})_j|$ for any row j:

$$\begin{aligned} &(|E_{j,1} + \ldots + E_{j,i}v'_i + \ldots + E_{j,n}|) - (|E_{j,1} + \ldots + E_{j,n}|) \\ &= |E_{j,1}| - |E_{j,1}| + \ldots + |E_{j,i}v'_i| - |E_{j,i}| + \ldots + |E_{j,n}| - |E_{j,n}| \\ &= |E_{j,i}v'_i| - |E_{j,i}| \\ &< 0 \end{aligned}$$

Which implies $\|e^{-L}v'\|_\infty < \|e^{-L}\vec{1}\|_\infty$. Intuitively, $v_i < 1$ will scale down the element $E_{j,i}$ and so $v = \vec{1}$ is the unique solution to the optimization problem with the constraint $\|v\|_\infty = 1$.

Problem 1.b

Consider the dynamical system:

$$\dot{x}(t) = Ax(t) + Bu(t).$$

Show that for a zero-order hold on the input, that is, $u(t) \equiv u_k$ for $t \in [k\Delta t, (k+1)\Delta t)$, $k = 1, 2, \ldots$, the discrete-time system is

$$x_{k+1} = A_d x_k + B_d u_k$$

where $x_k = x(k\Delta t)$,

$$A_d = e^{A\Delta t}, \quad B_d = \int_{\tau=0}^{\Delta t} e^{A\tau} B d\tau.$$

Further, show that, in fact,

$$\begin{bmatrix} A_d & B_d \\ 0 & I \end{bmatrix} = e^{\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \Delta t}.$$

Solution

We are given that

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ x_k &= x(k\Delta t) \\ \Delta t &= \text{integration time step} \end{aligned}$$

And we know that

$$\begin{aligned} \frac{\partial}{\partial t} e^{At} &= Ae^{At} \\ \frac{\partial}{\partial t}(e^{-At}x(t)) &= e^{-At}\dot{x}(t) - Ae^{-At}x(t) \text{ (product rule)} \end{aligned}$$

Rearranging the equation, we get $\dot{x}(t) - Ax(t) = Bu(t)$. Then multiplying by e^{-At} on both sides

$$\begin{aligned} e^{-At}\dot{x}(t) - Ae^{At}x(t) &= e^{-At}Bu(t) \\ \frac{\partial}{\partial t}(e^{-At}x(t)) &= e^{-At}Bu(t) \end{aligned}$$

Integrating both sides from $k\Delta t$ to $(k+1)\Delta t$:

$$\int_{t=k\Delta t}^{(k+1)\Delta t} \frac{\partial}{\partial t}(e^{-At}x(t))dt = \int_{t=k\Delta t}^{(k+1)\Delta t} e^{-At}Bu(t)dt$$

Simplifying the LHS

$$e^{-A(k+1)\Delta t}x((k+1)\Delta t) - e^{-Ak\Delta t}x(k\Delta t) = \int_{t=k\Delta t}^{(k+1)\Delta t} e^{-At}Bu(t)dt$$

Applying the zero order hold assumption $u(t) = u_k$ for $t \in [k\Delta t, (k+1)\Delta t)$, we get

$$e^{-A(k+1)\Delta t}x((k+1)\Delta t) - e^{-Ak\Delta t}x(k\Delta t) = \left(\int_{t=k\Delta t}^{(k+1)\Delta t} e^{-At}Bdt\right) u_k$$

Substituting $\tau = t - k\Delta t$ such that

$$\begin{aligned} &t = k\Delta t + \tau, dt = d\tau \\ &\text{when } t = k\Delta t, \tau = 0 \\ &\text{when } t = (k+1)\Delta t, \tau = \Delta t \end{aligned}$$

We get

$$e^{-A(k+1)\Delta t}x((k+1)\Delta t) - e^{-Ak\Delta t}x(k\Delta t) = \left(\int_{\tau=0}^{\Delta t} e^{-A(k\Delta t+\tau)}Bd\tau\right)u_k$$

$$e^{-A(k+1)\Delta t}x((k+1)\Delta t) - e^{-Ak\Delta t}x(k\Delta t) = e^{-Ak\Delta t}\left(\int_{\tau=0}^{\Delta t} e^{-A\tau}Bd\tau\right)u_k$$

$$e^{-A(k+1)\Delta t}x((k+1)\Delta t) = e^{-Ak\Delta t}x(k\Delta t) + e^{-Ak\Delta t}\left(\int_{\tau=0}^{\Delta t} e^{-A\tau}Bd\tau\right)u_k$$

Multiply both sides by $e^{A(k+1)\Delta t}$ to isolate $x((k+1)\Delta t)$

$$x((k+1)\Delta t) = e^{A(k+1)\Delta t}\left(e^{-Ak\Delta t}x(k\Delta t) + e^{-Ak\Delta t}\left(\int_{\tau=0}^{\Delta t} e^{-A\tau}Bd\tau\right)u_k\right)$$

$$x((k+1)\Delta t) = e^{A\Delta t}x(k\Delta t) + e^{A\Delta t}\left(\int_{\tau=0}^{\Delta t} e^{-A\tau}Bd\tau\right)u_k$$

$e^{-A\tau}$ can be written as $e^{-A\tau} = e^{A(\Delta t-\tau)}e^{-A\Delta t}$, where $e^{-A\Delta t}$ is a constant. Hence

$$x((k+1)\Delta t) = e^{A\Delta t}x(k\Delta t) + e^{A\Delta t}\left(\int_{\tau=0}^{\Delta t} e^{A(\Delta t-\tau)}e^{-A\Delta t}Bd\tau\right)u_k$$

$$x((k+1)\Delta t) = e^{A\Delta t}x(k\Delta t) + e^{A\Delta t}e^{-A\Delta t}\left(\int_{\tau=0}^{\Delta t} e^{A(\Delta t-\tau)}Bd\tau\right)u_k$$

$$x((k+1)\Delta t) = e^{A\Delta t}x(k\Delta t) + \left(\int_{\tau=0}^{\Delta t} e^{A(\Delta t-\tau)}Bd\tau\right)u_k$$

Substituting $\tau' = \Delta t - \tau$ such that

$$d\tau' = -d\tau \quad \text{differentiating both sides with } \tau$$
$$\text{when } \tau = 0, \tau' = \Delta t$$
$$\text{when } \tau = \Delta t, \tau' = 0$$

We get

$$x((k+1)\Delta t) = e^{A\Delta t}x(k\Delta t) + \left(\int_{\tau'=\Delta t}^{0} e^{A\tau'}B(-d\tau')\right)u_k$$

Where we can flip the limits of integration to get

$$x((k+1)\Delta t) = e^{A\Delta t}x(k\Delta t) + \left(\int_{\tau'=0}^{\Delta t} e^{A\tau'}Bd\tau'\right)u_k$$

Renaming τ' back to τ, since it is just a dummy variable, and using the definition $x_k = x(k\Delta t)$, we get

$$x_{k+1} = e^{A\Delta t}x_k + \left(\int_{\tau=0}^{\Delta t} e^{A\tau}Bd\tau\right)u_k$$

Hence, the discrete-time system is

$$x_{k+1} = A_d x_k + B_d u_k$$

where

$$A_d = e^{A\Delta t}, \quad B_d = \int_{\tau=0}^{\Delta t} e^{A\tau} B d\tau$$

For the second part of the question, we start with the definition of a matrix exponential of some matrix $X\Delta t$

$$\begin{aligned} e^{X\Delta t} &= I + X\Delta t + \frac{(X\Delta t)^2}{2!} + \frac{(X\Delta t)^3}{3!} + \dots \\ &= \sum_{n=0}^{\infty} \frac{(X\Delta t)^n}{n!} \end{aligned}$$

Let $X = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}$, then observing powers of X in the matrix exponential, we have

$$\begin{aligned} X^2 &= \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} A^2 & AB \\ 0 & 0 \end{bmatrix} \\ X^3 &= \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \begin{bmatrix} A^2 & AB \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} A^3 & A^2B \\ 0 & 0 \end{bmatrix} \\ X^n &= \begin{bmatrix} A^n & A^{n-1}B \\ 0 & 0 \end{bmatrix} \end{aligned}$$

Then, the $e^{X\Delta t}$ can be written as

$$\begin{aligned} e^{X\Delta t} &= I + \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \Delta t + \frac{1}{2!} \begin{bmatrix} A^2 & AB \\ 0 & 0 \end{bmatrix} (\Delta t)^2 + \dots \\ &= \begin{bmatrix} I + A\Delta t + \frac{A^2(\Delta t)^2}{2!} + \dots & B\Delta t + \frac{AB(\Delta t)^2}{2!} + \dots \\ 0 & I \end{bmatrix} \text{ (The identity matrix adds an identity to the bottom right corner)} \\ &= \begin{bmatrix} \sum_{n=0}^{\infty} \frac{A^n(\Delta t)^n}{n!} & \sum_{n=1}^{\infty} \frac{A^{n-1}B(\Delta t)^n}{n!} \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} e^{A\Delta t} & \sum_{n=1}^{\infty} \frac{A^{n-1}B(\Delta t)^n}{n!} \\ 0 & I \end{bmatrix} \text{ (using the definition of matrix exponential)} \\ &= \begin{bmatrix} e^{A\Delta t} & \sum_{n=1}^{\infty} \frac{A^{n-1}(\Delta t)^n}{n!} B \\ 0 & I \end{bmatrix} \text{ (rearranging the terms)} \end{aligned}$$

Using the definition of matrix exponential integrals

$$
\begin{aligned}
\int_{\tau=0}^{\Delta t} e^{A\tau} d\tau &= (e^{A\Delta t} - I)A^{-1} \\
&= \left(\sum_{n=0}^{\infty} \frac{A^n (\Delta t)^n}{n!} - I \right) A^{-1} \\
&= \left(I + A\Delta t + \frac{A^2(\Delta t)^2}{2!} + \ldots - I \right) A^{-1} \\
&= \left(A\Delta t + \frac{A^2(\Delta t)^2}{2!} + \ldots \right) A^{-1} \text{ (cancelling the identity terms)} \\
&= \left(\Delta t + \frac{A(\Delta t)^2}{2!} + \ldots \right) \text{ (dividing the } A \text{ terms)} \\
&= \sum_{n=1}^{\infty} \frac{A^{n-1}(\Delta t)^n}{n!}
\end{aligned}
$$

We can rewrite $e^{X\Delta t}$ as

$$
\begin{aligned}
e^{X\Delta t} &= \begin{bmatrix} e^{A\Delta t} & \sum_{n=1}^{\infty} \frac{A^{n-1}(\Delta t)^n}{n!} B \\ 0 & I \end{bmatrix} \\
&= \begin{bmatrix} e^{A\Delta t} & \int_{\tau=0}^{\Delta t} e^{A\tau} B d\tau \\ 0 & I \end{bmatrix} \\
&= \begin{bmatrix} A_d & B_d \\ 0 & I \end{bmatrix}
\end{aligned}
$$

Where $A_d = e^{A\Delta t}$ and $B_d = \int_{\tau=0}^{\Delta t} e^{A\tau} B d\tau$ as defined in the problem. Hence, we have shown that

$$
\begin{bmatrix} A_d & B_d \\ 0 & I \end{bmatrix} = e^{\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \Delta t}
$$

Problem 2.a

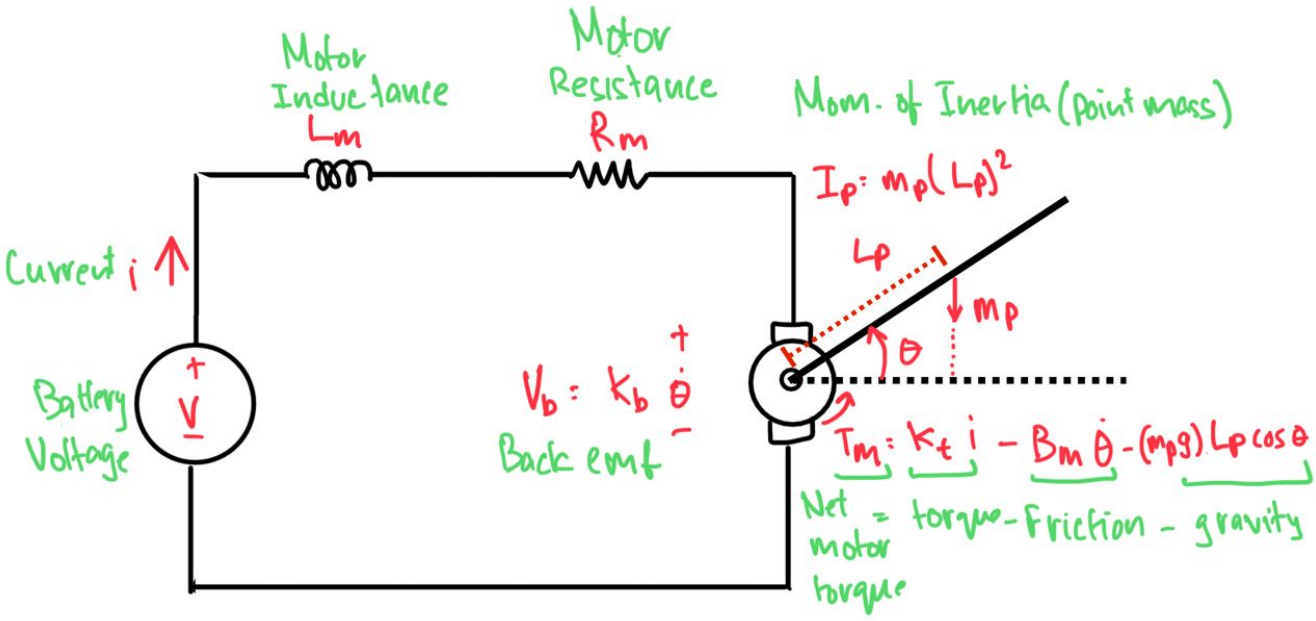


Figure 1: Circuit Diagram

For the given motor with a point mass attached, we are given:

- $R_m = 3.5\Omega$ (motor resistance)
- $K_b = 0.1 Vs/rad$ (motor back EMF constant)
- $K_t = 0.5 K_b = 0.05 Nm/A$ (motor torque constant)
- $L_m = 0H$ (motor inductance)
- $m_p = 0.04kg$ (point mass)
- $L_p = 0.1m$ (distance from motor to point mass)
- $I_p = m_p(L_p)^2 = 0.0004 kgm^2$ (moment of inertia of point mass)
- $B_m = 0.000008 Nms/rad$ (motor viscous friction constant)
- $g = 9.81 m/s^2$ (acceleration due to gravity)

Let

- θ : angle of the motor shaft
- V : input voltage
- i : motor current
- T_m : net torque on the motor shaft

We can derive the following equation for the voltage in the circuit

- Voltage drop due to inductance: $L_m \frac{di}{dt}$
- Voltage drop across the motor: $R_m i$
- Voltage drop due to back EMF: $K_b \dot{\theta}$

$$\implies V = L_m \frac{di}{dt} + R_m i + K_b \dot{\theta}$$

$$\text{Since } L_m = 0, \text{ we have } V = R_m i + K_b \dot{\theta}$$

$$i = \frac{V - K_b \dot{\theta}}{R_m} \quad (1)$$

We can also derive the following equation for the torque on the point mass

- Torque due to the motor: $K_t i$

- Frictional torque: $B_m\dot{\theta}$
- Force on mass due to gravity(weight): $m_p g$
- Perpendicular distance of the force of weight from the motor shaft: $L_p cos(\theta)$

$$\implies T_m = K_t i - B_m\dot{\theta} - m_p g L_p cos(\theta)$$

Substituting for i from (1), we have

$$\begin{aligned} T_m &= K_t\left(\frac{V - K_b\dot{\theta}}{R_m}\right) - B_m\dot{\theta} - m_p g L_p cos(\theta) \\ &= \frac{K_t V}{R_m} - \frac{K_t K_b\dot{\theta}}{R_m} - B_m\dot{\theta} - m_p g L_p cos(\theta) \\ &= \frac{K_t V}{R_m} - \left(\frac{K_t K_b}{R_m} + B_m\right)\dot{\theta} - m_p g L_p cos(\theta) \quad (2) \end{aligned}$$

Finally, using the equation of angular acceleration and torque, we have

$$T_m = I_p\ddot{\theta}$$

Substituting for T_m from (2), we have

$$\begin{aligned} I_p\ddot{\theta} &= \frac{K_t V}{R_m} - \left(\frac{K_t K_b}{R_m} + B_m\right)\dot{\theta} - m_p g L_p cos(\theta) \\ \ddot{\theta} &= \frac{K_t V}{I_p R_m} - \left(\frac{K_t K_b}{R_m} + B_m\right)\frac{\dot{\theta}}{I_p} - \frac{m_p g L_p}{I_p} cos(\theta) \quad (3) \end{aligned}$$

For our state space model, we have the following state variables:

$$x_1 = \theta \quad x_2 = \dot{\theta} \quad u = V$$

Where we observe only the output angle θ of the motor shaft. Which gives us the following state space equations:

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= \frac{K_t u}{I_p R_m} - \left(\frac{K_t K_b}{R_m} + B_m\right)\frac{x_2}{I_p} - \frac{m_p g L_p}{I_p} cos(x_1) \\ y &= x_1 \end{aligned}$$

Problem 2.b

To find the equilibrium points, we set $u = 0$ and $\dot{x_1} = 0$:

$$\begin{aligned} 0 &= x_2 \\ \implies x_2 &= 0 \end{aligned}$$

similarly, we set $\dot{x_2} = 0$:

$$\begin{aligned} 0 &= \frac{K_t u}{I_p R_m} - \left(\frac{K_t K_b}{R_m} + B_m\right)\frac{x_2}{I_p} - \frac{m_p g L_p}{I_p} cos(x_1) \\ &\text{Since } x_2 = 0 \text{ and } u = 0, \text{ we have} \\ 0 &= \frac{K_t 0}{I_p R_m} - \left(\frac{K_t K_b}{R_m} + B_m\right)\frac{0}{I_p} - \frac{m_p g L_p}{I_p} cos(x_1) \\ 0 &= -\frac{m_p g L_p}{I_p} cos(x_1) \\ \implies cos(x_1) &= 0 \end{aligned}$$

Hence, the equilibrium points are

$$\begin{aligned} x_{1e} &= \pm\frac{\pi}{2} + 2\pi n \quad n \in \mathbb{Z} \\ x_{2e} &= 0 \\ u_e &= 0 \end{aligned}$$

To comment on the stability of the points, we need to linearize the system around the equilibrium points and check the eigenvalues of the system. First, we need the Jacobian matrix $\nabla_x f$ where:

$$\begin{aligned} \boldsymbol{x} &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ f &= \begin{bmatrix} x_2 \\ \frac{K_t u}{I_p R_m} - \left(\frac{K_t K_b}{R_m} + B_m\right)\frac{x_2}{I_p} - \frac{m_p g L_p}{I_p} cos(x_1) \end{bmatrix} \end{aligned}$$

Then, the Jacobian matrix is:

$$\begin{aligned} \nabla_x f &= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \\ \frac{\partial f_1}{\partial x_1} &= 0 \\ \frac{\partial f_1}{\partial x_2} &= 1 \\ \frac{\partial f_2}{\partial x_1} &= \frac{m_p g L_p}{I_p} sin(x_1) \\ \frac{\partial f_2}{\partial x_2} &= -\left(\frac{K_t K_b}{R_m} + B_m\right)\frac{1}{I_p} \\ \implies \nabla_x f &= \begin{bmatrix} 0 & 1 \\ \frac{m_p g L_p}{I_p} sin(x_1) & -\left(\frac{K_t K_b}{R_m} + B_m\right)\frac{1}{I_p} \end{bmatrix} \end{aligned}$$

Linearizing around the equilibrium points x_{1e}, x_{2e}, u_e, for x_{1e} we select the points $\pi/2$ and $-\pi/2$.

For $x_{1e} = \pi/2$, we have:

$$\begin{aligned} A_1 &= \nabla_x f\Big|_{x_{1e}=\pi/2, x_{2e}=0, u_e=0} \\ &= \begin{bmatrix} 0 & 1 \\ \frac{m_p g L_p}{I_p} & -\left(\frac{K_t K_b}{R_m} + B_m\right)\frac{1}{I_p} \end{bmatrix} \end{aligned}$$

To check stability, we need to find the eigenvalues of the matrix A_1

- 7.84
- -12.51

Since we have an eigenvalue with a positive real part, the equilibrium point is unstable.

For $x_{1e} = -\pi/2$, we have:

$$\begin{aligned} A_2 &= \nabla_x f\Big|_{x_{1e}=-\pi/2, x_{2e}=0, u_e=0} \\ &= \begin{bmatrix} 0 & 1 \\ -\frac{m_p g L_p}{I_p} & -\left(\frac{K_t K_b}{R_m} + B_m\right)\frac{1}{I_p} \end{bmatrix} \end{aligned}$$

To check stability, we need to find the eigenvalues of the matrix A_2

- -2.33 + 9.63i
- -2.33 - 9.63i

Since we have eigenvalues with negative real parts, the equilibrium point is stable. In general, we can say that the equilibrium points

$$\boldsymbol{x}_e = \begin{bmatrix} \pm\frac{\pi}{2} + 2\pi n \\ 0 \end{bmatrix} \quad n \in \mathbb{Z}$$

are stable for $x_1 = -\frac{\pi}{2} + 2\pi n \quad n \in \mathbb{Z}$ (when the pendulum is hanging down) and unstable for $x_1 = \frac{\pi}{2} + 2\pi n \quad n \in \mathbb{Z}$ (when the pendulum is upright).

We can perturb the original system by adding a small disturbance to the system around the equilibrium points and observe the behavior of the system. In the case of closed circuit, the dynamics stay the same as derived above. However, for open circuit, no current can flow in the circuit, so $i = 0$. Therefore, net torque on the motor shaft becomes

$$\begin{aligned} T_m &= K_t i - B_m\dot{\theta} - m_p g L_p cos(\theta) \\ &= K_t 0 - B_m\dot{\theta} - m_p g L_p cos(\theta) \\ &= -B_m\dot{\theta} - m_p g L_p cos(\theta) \quad (4) \end{aligned}$$

Using the eqution of angular acceleration and torque, we have

$$T_m = I_p\ddot{\theta}$$

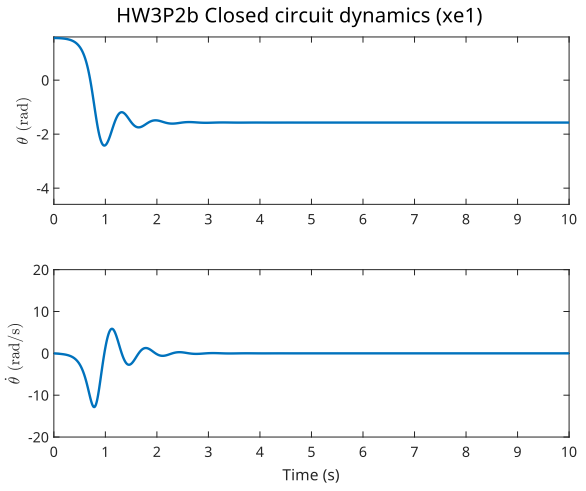
Substituting for T_m from (4), we have

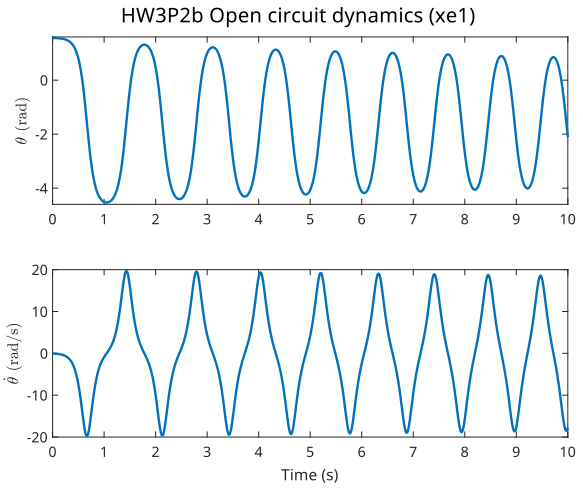
$$\begin{aligned} I_p\ddot{\theta} &= -B_m\dot{\theta} - m_p g L_p cos(\theta) \\ \ddot{\theta} &= -\frac{B_m}{I_p}\dot{\theta} - \frac{m_p g L_p}{I_p} cos(\theta) \quad (5) \end{aligned}$$

Using the state variables $x_1 = \theta, x_2 = \dot{\theta}$, we have the following state space equations for the open circuit case:

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -\frac{B_m}{I_p}x_2 - \frac{m_p g L_p}{I_p} cos(x_1) \\ y &= x_1 \end{aligned}$$

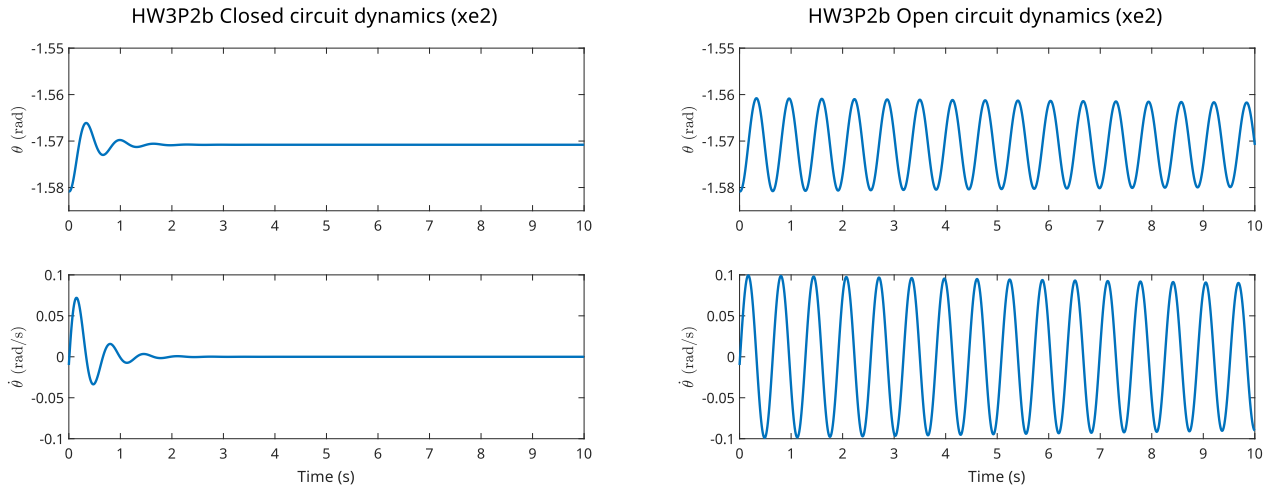
The system state $\boldsymbol{x}$ were perturbed by a small amount $[-0.01, -0.01]^T$ Perturbed dynamics for $\boldsymbol{x}_{e1} = [\pi/2, 0]^T$ are:





We know that the equilibrium point $\boldsymbol{x}_{e1} = [\pi/2, 0]^T$ is unstable, so we see the perturbed system diverging from this unstable equilibrium point in both cases and slowly stabilizing around the stable equilibrium point $\boldsymbol{x}_{e2} = [-\pi/2, 0]^T$. However, the open circuit case does not stabilize within 10 seconds, while the closed circuit case does. This is because the closed circuit case has a damping torque due the back EMF of the motor ($K_t i$) and resistive power losses from the motor resistance ($R_m i$) in addition to the viscous friction torque which helps stabilize the system faster.

Perturbed dynamics for $\boldsymbol{x}_{e2} = [-\pi/2, 0]^T$ are:



We know that the equilibrium point $\boldsymbol{x}_{e2} = [-\pi/2, 0]^T$ is stable, so we see the perturbed system converging to this stable equilibrium point in both cases. However, the open circuit case takes longer to stabilize compared to the closed circuit case. This is because the closed circuit case has a damping torque due the back EMF of the motor ($K_t i$) and resistive power losses from the motor resistance ($R_m i$) in addition to the viscous friction torque which helps stabilize the system faster. Even so, the open circuit system only wanders around the stable equilibrium point within the same order of magnitude as the perturbation and does not diverge.

Problem 2.c

We already know the matrices A_1 and A_2 for the linearized system around the equilibrium points $\boldsymbol{x}_{e1}$ and $\boldsymbol{x}_{e2}$ respectively from part (b). We now need the matrices B, C and D for the state space representation of the system. To compute these, we will need the the jacobians $\nabla_u f$, $\nabla_x h$, and $\nabla_u h$ where:

$$
\begin{aligned}
\boldsymbol{x} &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\
\boldsymbol{u} &= \begin{bmatrix} u \end{bmatrix} \\
\boldsymbol{y} &= \begin{bmatrix} y \end{bmatrix} \\
f &= \begin{bmatrix} x_2 \\ \frac{K_t u}{I_p R_m} - \left(\frac{K_t K_b}{R_m} + B_m \right) \frac{x_2}{I_p} - \frac{m_p g L_p}{I_p} cos(x_1) \end{bmatrix} \\
h &= \begin{bmatrix} x_1 \end{bmatrix}
\end{aligned}
$$

We can then define the jacobians as follows:

$$
\begin{aligned}
\nabla_u f &= \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix} \\
\frac{\partial f_1}{\partial u} &= 0 \\
\frac{\partial f_2}{\partial u} &= \frac{K_t}{I_p R_m} \\
\implies \nabla_u f &= \begin{bmatrix} 0 \\ \frac{K_t}{I_p R_m} \end{bmatrix} \\
\nabla_x h &= \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} \end{bmatrix} \\
\frac{\partial h_1}{\partial x_1} &= 1 \\
\frac{\partial h_1}{\partial x_2} &= 0 \\
\implies \nabla_x h &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\
\nabla_u h &= \begin{bmatrix} \frac{\partial h_1}{\partial u} \end{bmatrix} \\
\frac{\partial h_1}{\partial u} &= 0 \\
\implies \nabla_u h &= \begin{bmatrix} 0 \end{bmatrix}
\end{aligned}
$$

Since none of them depend on x_1, they will be the same for both equilibrium points. Thus we have:

$$
\begin{aligned}
B &= \nabla_u f = \begin{bmatrix} 0 \\ \frac{K_t}{I_p R_m} \end{bmatrix} \\
C &= \nabla_x h = \begin{bmatrix} 1 & 0 \end{bmatrix} \\
D &= \nabla_u h = \begin{bmatrix} 0 \end{bmatrix}
\end{aligned}
$$

The transfer function of a system is $G(s) = C(sI - A)^{-1}B + D$.

At $\boldsymbol{x}_{e1} = [\pi/2, 0]^T$ we have:

$$
G_1(s) = C(sI - A_1)^{-1}B + D
$$

At $\boldsymbol{x}_{e2} = [-\pi/2, 0]^T$ we have:

$$
G_2(s) = C(sI - A_2)^{-1}B + D
$$

The inverse laplace transform of the transfer function gives the impulse response function of the system $g(t)$. Where $g(t) = \mathcal{L}^{-1}\{G(s)\}$.

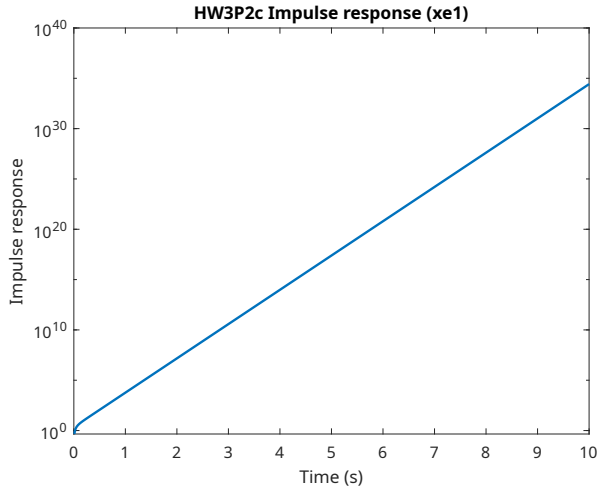
At $\boldsymbol{x}_{e1} = [\pi/2, 0]^T$ we have:

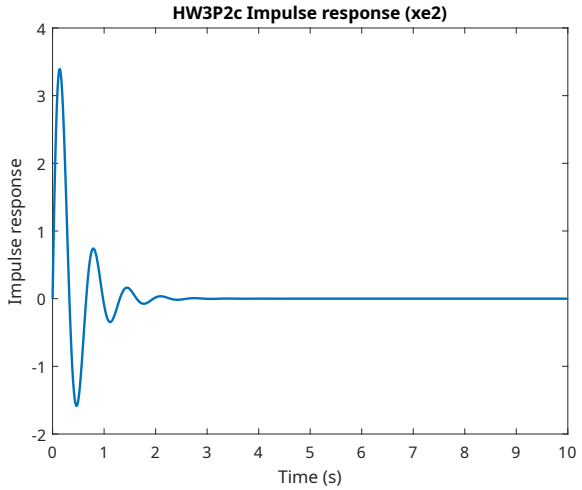
$$
\begin{aligned}
g_1(t) &= \mathcal{L}^{-1}\{G_1(s)\} \\
&= 4.56e^{-2.33t} sinh(10.2t)
\end{aligned}
$$

At $\boldsymbol{x}_{e2} = [-\pi/2, 0]^T$ we have:

$$
\begin{aligned}
g_2(t) &= \mathcal{L}^{-1}\{G_2(s)\} \\
&= 4.82e^{-2.33t} sin(9.63t)
\end{aligned}
$$

We can plot the impulse response in matlab by using the `impulse` function in matlab.





For the stable equilibrium point $\boldsymbol{x}_{e2} = [-\pi/2, 0]^T$, the system converges to the equilibrium point with time as expected, similar to the perturbed dynamics. However, for the unstable equilibrium point $\boldsymbol{x}_{e1} = [\pi/2, 0]^T$, the system diverges from the equilibrium point. Since the transfer function here uses linearized dynamics, it is only accurate near the unstable equilibrium point. So once the system diverges, the linearized dynamics are no longer valid and the system diverges exponentially.

Problem 2.d

To stabilize the system about $\theta = \pi/4$, we need to first linearize the dynamics around the point $\boldsymbol{x}_s = [\pi/4, 0]^T$. Using results from part (b) and (c), we have the matrices A_s, B, C, and D

$$A_s = \begin{bmatrix} 0 & 1 \\ \frac{m_p g L_p}{I_p} sin(\pi/4) & -\left(\frac{K_t K_b}{R_m} + B_m\right)\frac{1}{I_p} \end{bmatrix}$$
$$B = \begin{bmatrix} 0 \\ \frac{K_t}{I_p R_m} \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$
$$D = \begin{bmatrix} 0 \end{bmatrix}$$

Using these, we find the transfer function $G_s(s) = C(sI - A_s)^{-1}B + D$ symbolically using matlab to get

$$G_s(s) = \frac{z}{as^2 + bs + c} \text{ where}$$
$$z = 17867063951360000$$
$$a = 384829069721600$$
$$b = 1794402976530432$$
$$c = -26694505514669475$$

We wish to design a feedback controller $C(s)$ such that the desired closed loop transfer function $G_{CL}(s)$ is of the following form

$$G_{CL}(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

With the specifications

- Settling time: $t_s < 0.2s$
- Overshoot: $\%OS < 15\%$

We note that

$$t_s = \frac{4}{\zeta\omega_n}$$
$$\%OS = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} \times 100\%$$

Therefore, the specifications give us the following constraints on ζ and ω_n

$$\begin{aligned}
t_s < 0.2s &\implies \zeta\omega_n > \frac{4}{0.2} \\
&\implies \zeta\omega_n > 20 \\
\%OS < 15\% &\implies e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} < 0.15 \\
&\implies \zeta > \frac{-ln(0.15)}{\sqrt{\pi^2 + (ln(0.15))^2}} \\
&\implies \zeta > 0.5169 \\
&\implies \zeta = 0.5170 \\
\text{Therefore } \omega_n &> \frac{20}{0.517} \\
&\implies \omega_n > 38.6824 \\
&\implies \omega_n = 38.6825
\end{aligned}$$

Hence, we choose $\zeta = 0.5170$ and $\omega_n = 38.6825$ by by adding 0.0001 to keep them as small as possible.

We wish to design a controller $C(s)$ such that the closed loop system satisfies the specifications and constraints with the desired poles:

$$P_{1,2} = -\zeta\omega_n \pm i\omega_n\sqrt{1-\zeta^2}$$

Using matlab, we find the desired poles, which are the roots of the denominator of the desired closed loop transfer function.

$$\begin{aligned}
P_1 &= -20.00 + 33.11i \\
P_2 &= -20.00 - 33.11i
\end{aligned}$$

Using the expression for $G_{CL}(s)$ for a PD controller, we have

$$G_{CL}(s) = \frac{G_s(s)C(s)}{1 + G_s(s)C(s)} = \frac{\frac{z(K_P + K_D s)}{a}}{s^2 + \left(\frac{zK_D + b}{a}\right)s + \frac{zK_P + c}{a}}$$

We choose K_P and K_D such that the denominator of our $G_{CL}(s)$ mathces the denominator of the desired $G_{CL}(s)$, i.e.

$$\begin{aligned}
s^2 + \left(\frac{zK_D + b}{a}\right)s + \frac{zK_P + c}{a} &= (s - P_1)(s - P_2) \\
s^2 + \left(\frac{zK_D + b}{a}\right)s + \frac{zK_P + c}{a} &= s^2 - (P_1 + P_2)s + P_1P_2 \\
\implies K_D = \frac{-a(P_1 + P_2) - b}{z}, &\quad K_P = \frac{aP_1P_2 - c}{z} \\
\implies K_D = 0.76, &\quad K_P = 33.72
\end{aligned}$$

Giving us the controller

$$C(s) = K_P + K_D s$$

Since we actually want a PID controller to improve the steady state error, we need to add an additional term P_I so that

$$C(s) = (K_P + K_D s)\frac{(s + P_I)}{s}$$

We choose P_I such that it is a lot smaller than the first pole P_1 to ensure that the integral term does not dominate the controller.

$$\begin{aligned} P_I &< abs(P_1) \\ &= 0.5 abs(P_1) \\ &= 19.34 \end{aligned}$$

Using this additional term, we can update our PID gains in the following way:

$$\begin{aligned} \bar{K_P} &= K_P + K_D P_I &&= 48.44 \\ \bar{K_I} &= K_P P_I &&= 652.24 \\ \bar{K_D} &= K_D &&= 0.76 \end{aligned}$$

Resulting in the final controller

$$C(s) = \bar{K_P} + \frac{\bar{K_I}}{s} + \bar{K_D} s$$

Problem 2.e

The contoller was implemented in simulink using the PID gains derived in part (d). The system was initialized at the equilibrium point $\pi/2$. The following blocks were added to a basic PID controller to ensure stability and performance:

- The reference had a maximum slew rate set to ± 10 rad/s to prevent the derivative term from causing large control inputs, which is realistic since the reference cannot change instantaneously.
- A saturation block was added to limit the control input to $[9, -9]V$ to prevent the motor from being overdriven.

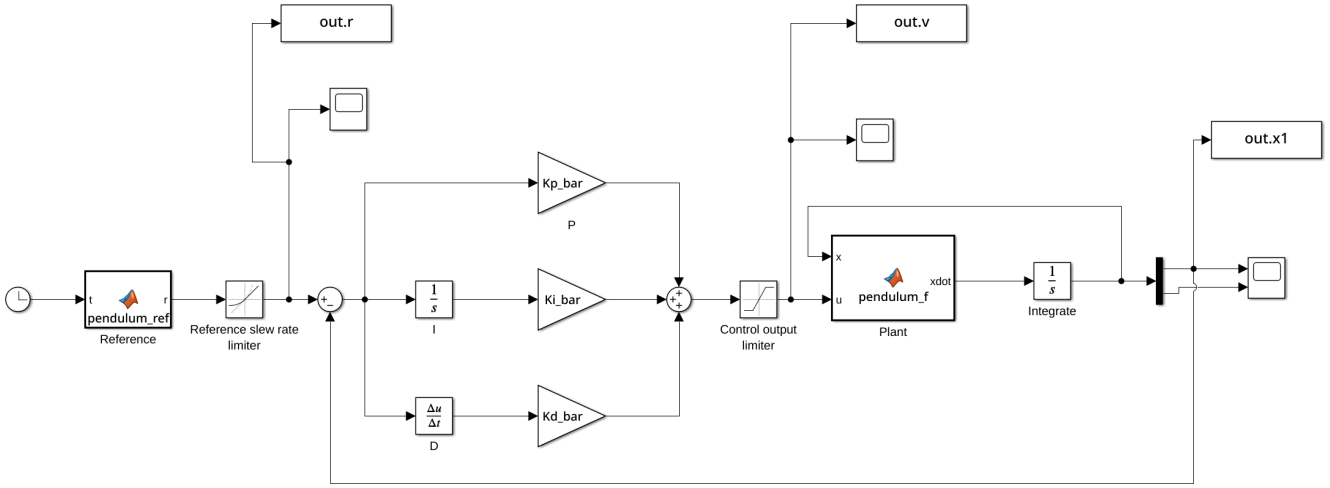


Figure 2: Simulink model

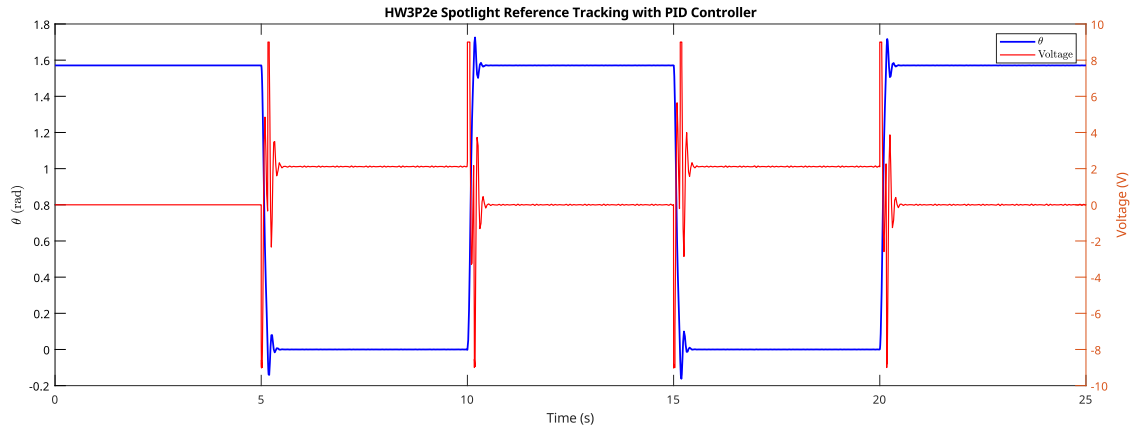


Figure 3: Reference tracking with PID controller

Which was able to follow the reference trajectory while staying within the bounds of $[9, -9]V$.

We can see that when the reference was set to $\pi/2$, the controller was able to maintain the stable equilibrium with only minor correction and resulted in only minor oscillations. This was difficult to achieve since even small offsets beyond $\pi/2$ can cause the system to diverge when using the linearized dynamics as seen in the impulse response in part (c).

When the reference was set to 0, the controller was again able to keep the system stable, counteracting gravity to maintain the reference position.

The only time we see the controller hit its output limits is when the reference changes from $\pi/2$ to 0 and vice versa, which is expected because those are the transitions that require the most control effort.

Given the design specifications of $t_s < 0.2s$ and $\%OS < 15\%$, we can see a maximum overshoot of $\approx 10\%$. As for the settling time, due to the slew rate limit, the reference change takes $\approx 0.23s$ and the total settling time including the reference change is $\approx 0.41s$. If we disregard the time taken for the reference change, the controller has a settling time of $\approx 0.18s$ which meets the design specifications.

Hence, we can say that even though we used the linearized dynamics of the system at $\pi/4$ to design the controller, it was able to stabilize the system at $\pi/2$ and 0 as well while meeting the design specifications.

Table of Contents

ME599 Homework 3 Problem 2

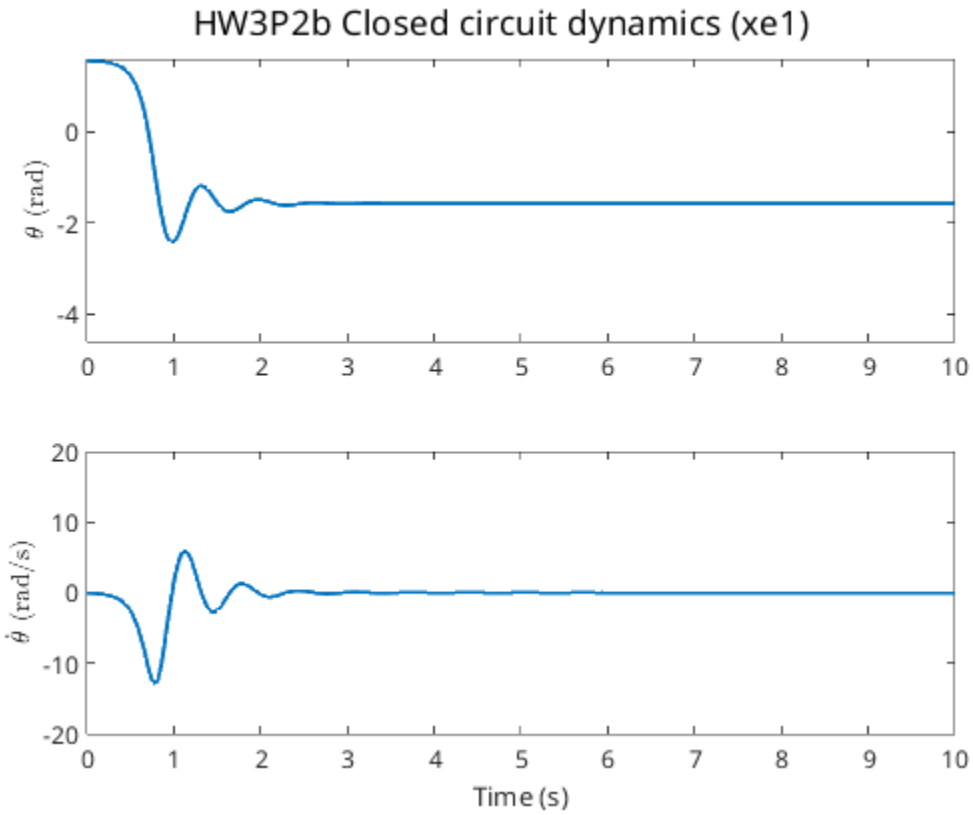
```
clc; clear; close all;

% system parameters
Rm = 3.5; % Ohms, Armature resistance
Kb = 0.1; % Vs/rad, Motor back EMF constant
Kt = 0.65*Kb; % Nm/A, Motor torque constant
% Lm;  H, Armature inductance; assume 0
mp = 0.04;% kg, Spotlight(pendulum) mass
Lp = 0.1; %m, Spotlight(pendulum) length
Ip = mp*(Lp)^2;% Moment of inertia of pendulum about center of mass
Bm = 0.000008; %Nms/rad (viscous friction coefficient)
g = 9.81; %m/s^2 acceleration due to gravity
```

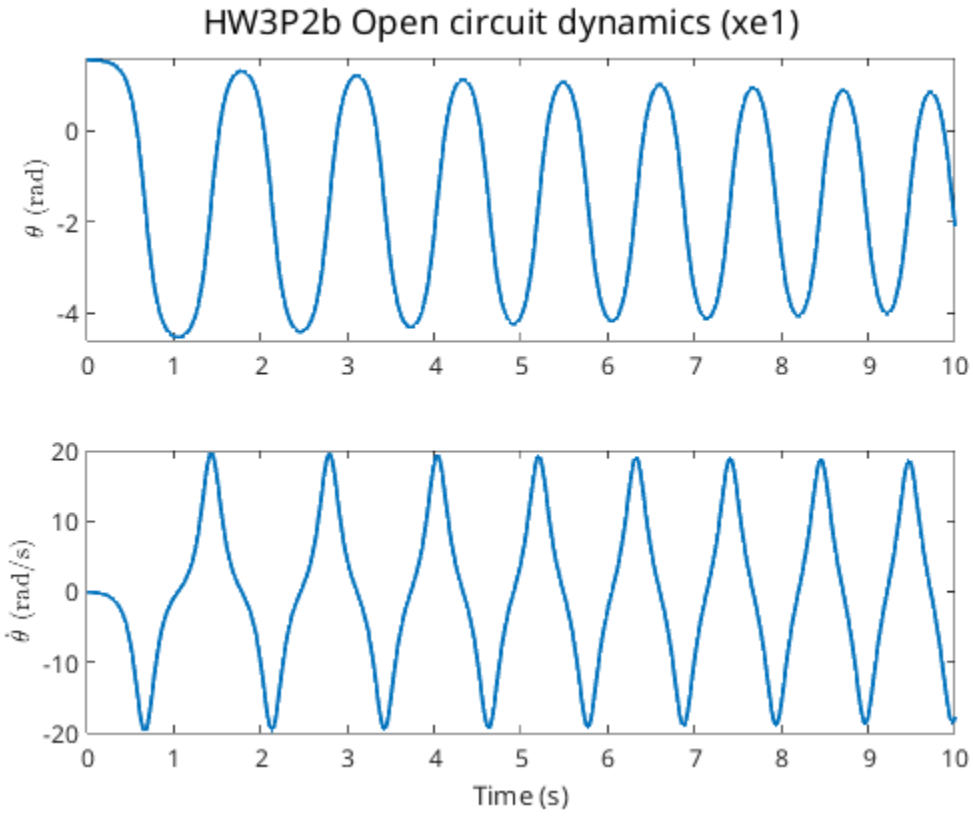
b: equilibrium points and stability

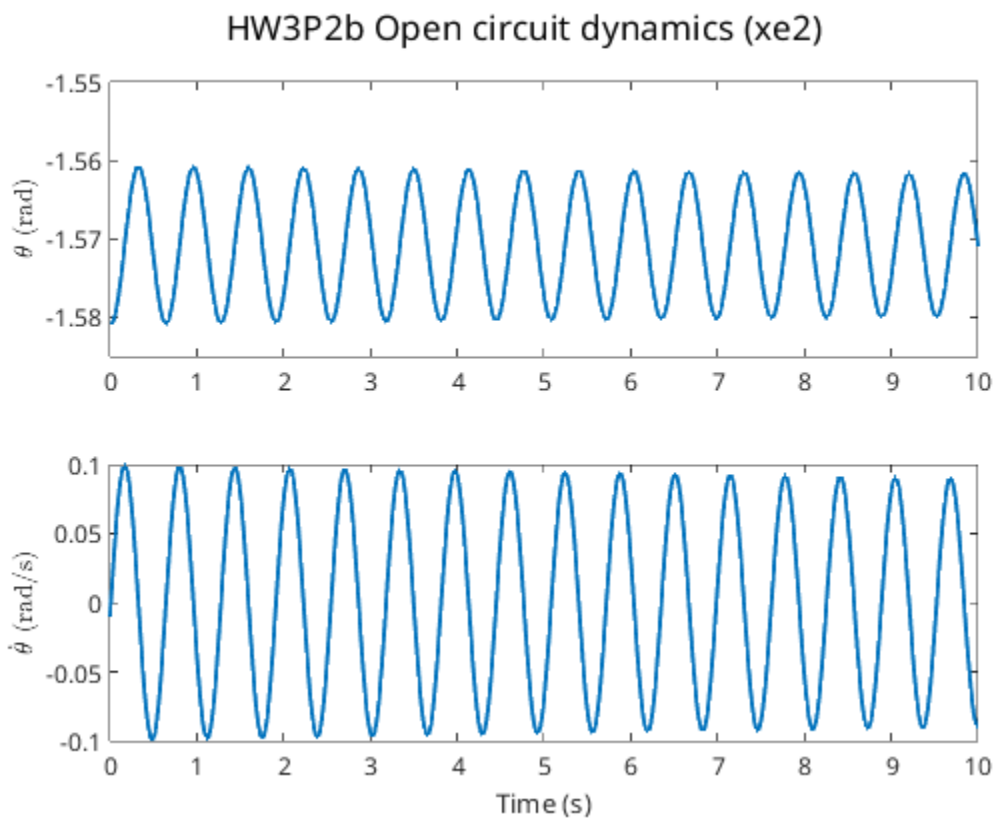
```
x1_e1 = pi/2;
x1_e2 = -pi/2;
x2_e = 0;
u_e = 0;

A1 = pendulum_fdx(0, [x1_e1; x2_e], u_e);
A2 = pendulum_fdx(0, [x1_e2; x2_e], u_e);

e1 = eig(A1);
e2 = eig(A2);

fprintf('The eigenvalues of the linearized system about the equilibrium point
x1=pi/2 are:\n %.2f\n %.2f\n', e1(1), e1(2));
fprintf('The eigenvalues of the linearized system about the equilibrium point
x1=-pi/2 are:\n %.2f\n %.2f\n', e2(1), e2(2));

% check stability
if real(e1(1)) < 0 && real(e1(2)) < 0
    fprintf('The equilibrium point x1=pi/2 is stable\n');
else
    fprintf('The equilibrium point x1=pi/2 is unstable\n');
end

if real(e2(1)) < 0 && real(e2(2)) < 0
    fprintf('The equilibrium point x1=-pi/2 is stable\n');
```

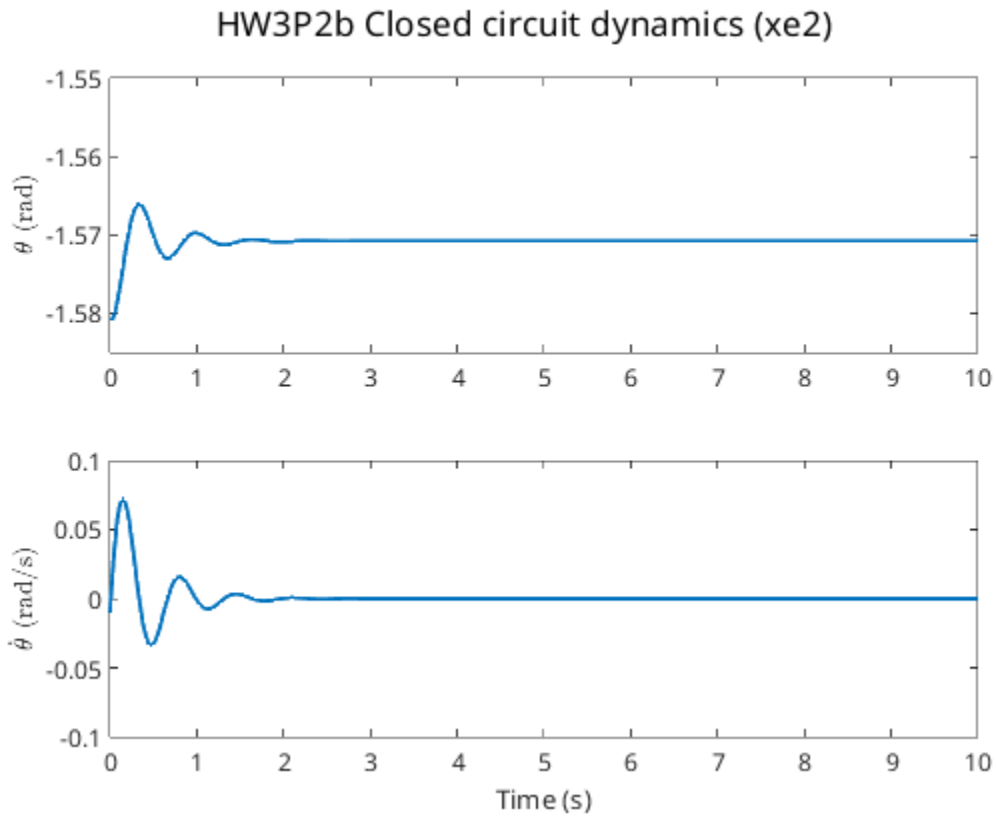
```
else
    fprintf('The equilibrium point x1=-pi/2 is unstable\n');
end

% perturb system
p = [-0.01; -0.01]; % perturbation
u0 = u_e;
x0 = [x1_e1; x2_e] + p;
tspan = 0:0.01:10;
[t, x] = ode45(@(t, x) pendulum_f(t, x, u0), tspan, x0);
plot_pend_states(t, x, 'HW3P2b Closed circuit dynamics (xe1)', [-4.6 1.6],
[-20 20]);
[t, x] = ode45(@(t, x) pendulum_f_oc(t, x, u0), tspan, x0);
plot_pend_states(t, x, 'HW3P2b Open circuit dynamics (xe1)', [-4.6 1.6], [-20
20]);

x0 = [x1_e2; x2_e] + p;
[t, x] = ode45(@(t, x) pendulum_f(t, x, u0), tspan, x0);
plot_pend_states(t, x, 'HW3P2b Closed circuit dynamics (xe2)', [-1.585
-1.55], [-0.1 0.1]);
[t, x] = ode45(@(t, x) pendulum_f_oc(t, x, u0), tspan, x0);
plot_pend_states(t, x, 'HW3P2b Open circuit dynamics (xe2)', [-1.585 -1.55],
[-0.1 0.1]);
```

```
The eigenvalues of the linearized system about the equilibrium point x1=pi/2
are:
 7.84
 -12.51
The eigenvalues of the linearized system about the equilibrium point x1=-pi/2
are:
 -2.33
 -2.33
The equilibrium point x1=pi/2 is unstable
The equilibrium point x1=-pi/2 is stable
```


HW3P2b Closed circuit dynamics (xe1)
θ (rad)
θ̇ (rad/s)
Time (s)


HW3P2b Open circuit dynamics (xe1)
θ (rad)
θ̇ (rad/s)
Time (s)


HW3P2b Open circuit dynamics (xe2)
θ (rad)
-1.55
-1.56
-1.57
-1.58
θ̇ (rad/s)
0.1
0.05
0
-0.05
-0.1
0 1 2 3 4 5 6 7 8 9 10
Time (s)


HW3P2b Closed circuit dynamics (xe2)
θ (rad)
-1.55
-1.56
-1.57
-1.58
θ̇ (rad/s)
0.1
0.05
0
-0.05
-0.1
0 1 2 3 4 5 6 7 8 9 10
Time (s)

c: impulse response

```
B = [0; Kt/(Ip*Rm)];
C = [1 0];
D = 0;
% get impulse response function(inverse laplace)
syms s
G_1 = C*inv(s*eye(2) - A1)*B+D;
g_1 = vpa(ilaplace(G_1), 3);
fprintf('The impulse response of the system about the equilibrium point
x1=pi/2 is:\n');
display(g_1);
G_2 = C*inv(s*eye(2) - A2)*B+D;
g_2 = vpa(ilaplace(G_2), 3);
fprintf('The impulse response of the system about the equilibrium point x1=-
pi/2 is:\n');
display(g_2);
% plot impulse response
y = impulse(ss(A1, B, C, D), tspan);
plot_impulse_resp(tspan, y, 'HW3P2c Impulse response (xe1)');
y = impulse(ss(A2, B, C, D), tspan);
plot_impulse_resp(tspan, y, 'HW3P2c Impulse response (xe2)');
```

The impulse response of the system about the equilibrium point x1=pi/2 is:

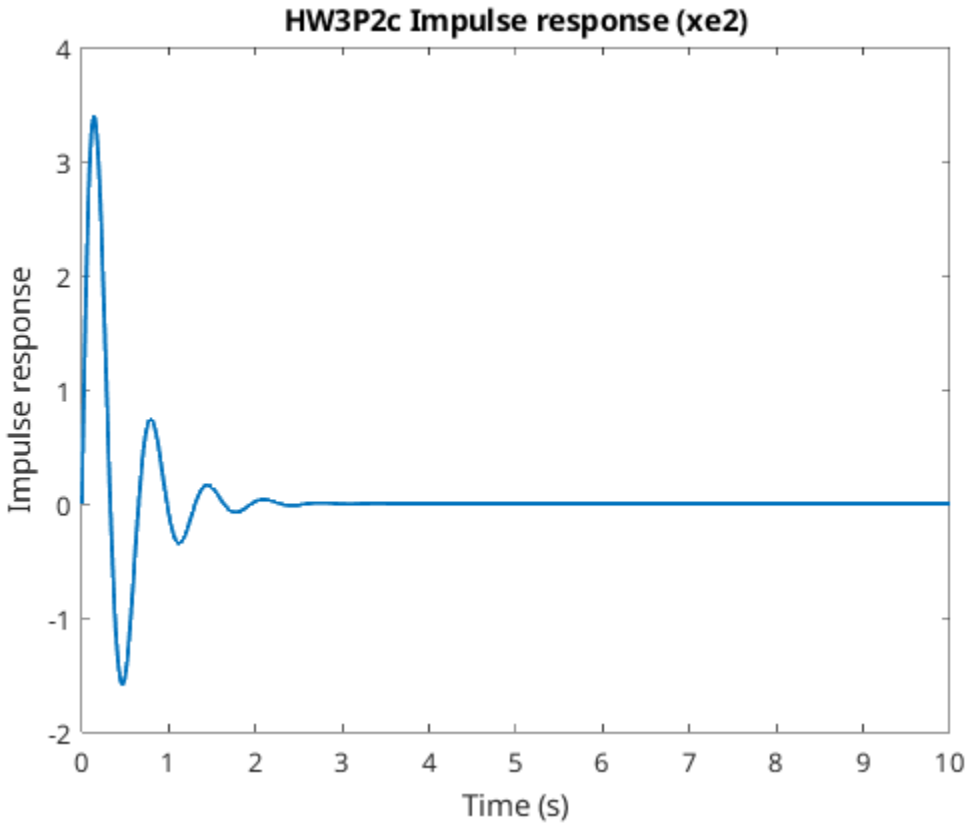
g_1 =

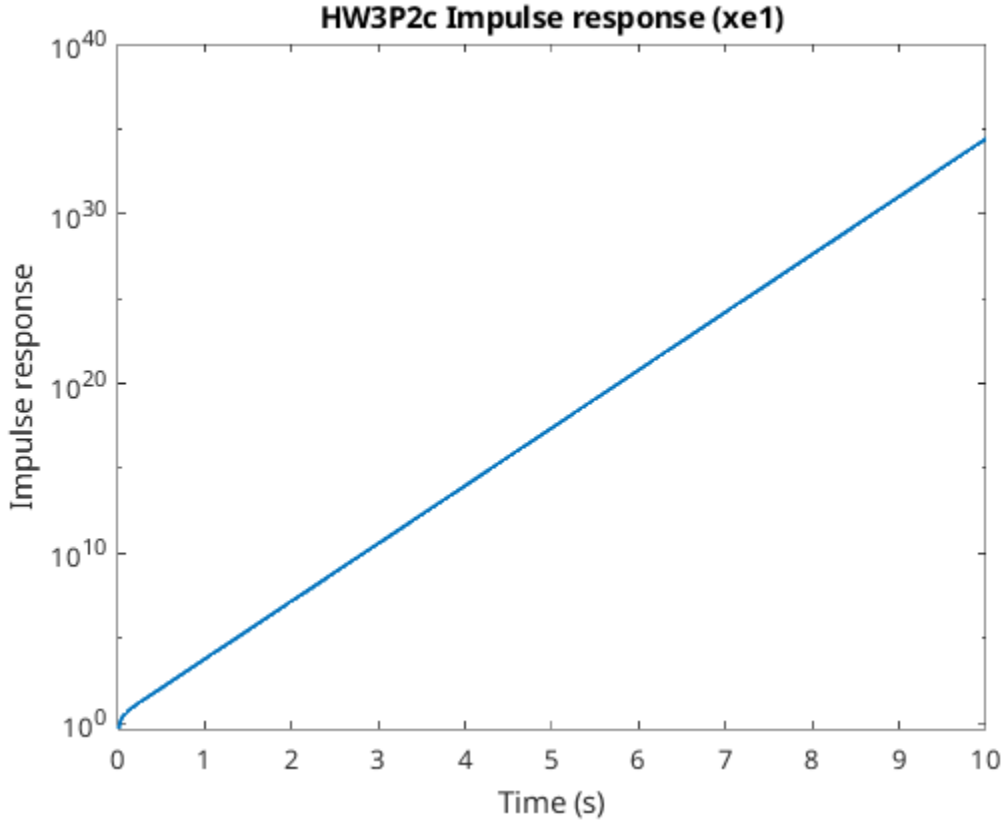
*4.56*exp(-2.33*t)*sinh(10.2*t)*

The impulse response of the system about the equilibrium point x1=-pi/2 is:

g_2 =

*4.82*exp(-2.33*t)*sin(9.63*t)*


HW3P2c Impulse response (xe2)
Impulse response
Time (s)
4
3
2
1
0
-1
-2
0
1
2
3
4
5
6
7
8
9
10


HW3P2c Impulse response (xe1)
Impulse response
Time (s)
10^{40}
10^{30}
10^{20}
10^{10}
10^{0}
0
1
2
3
4
5
6
7
8
9
10

d: controller design

transfer function

```
x_c = [pi/4; 0];
As = pendulum_fdx(0, x_c, u_e);
syms s
G_s = C*inv(s*eye(2) - As)*B+D;
fprintf('The transfer function of the system about the equilibrium point
x1=pi/4 is:\n');
display(G_s);
[num, den] = numden(G_s);
z = sym2poly(num);
den_coeff = sym2poly(den);
a = den_coeff(1); b = den_coeff(2); c = den_coeff(3);

% find zeta and omega
ts_max = 0.2;
os_max = 0.15;
% add a tiny amount extra so theyre greater than
zeta = -log(os_max)/sqrt(pi^2 + log(os_max)^2) + 0.0001;
omega_n = 4/(zeta*ts_max) + 0.0001;
fprintf('The damping ratio zeta is %.4f\n', zeta);
fprintf('The natural frequency omega_n is %.4f\n', omega_n);

% find poles
P1 = -zeta*omega_n + 1i*omega_n*sqrt(1 - zeta^2);
P2 = -zeta*omega_n - 1i*omega_n*sqrt(1 - zeta^2);
fprintf('The poles of the system are:\n %.2f %.2fi\n %.2f %.2fi\n', real(P1),
imag(P1), real(P2), imag(P2));

% use poles to find the desired KP and KD
Kd = (-a*(P1+P2)-b)/z;
Kp = (a*P1*P2-c)/z;
fprintf('The desired KP=%.2f, KD=%.2f\n', Kp, Kd);

% add pole for KI and update PID gains
% multiplier = 10;
% PI = -zeta*omega_n * multiplier;
multiplier = 0.5;
PI = multiplier*abs(P1);
fprintf('The pole for KI, PI=%.2f\n', PI);

% Kp_bar = real(Kp + a/z*PI*(P1+P2));
% Ki_bar = real(-PI*P1*P2*a/z);
% Kd_bar = real(Kd - a/z*PI);
Kp_bar = real(Kp + Kd*PI);
Ki_bar = real(Kp*PI);
Kd_bar = Kd;
fprintf('Final PID gains:\n KP=%.2f\n KI=%.2f\n KD=%.2f\n', Kp_bar, Ki_bar,
Kd_bar);
```

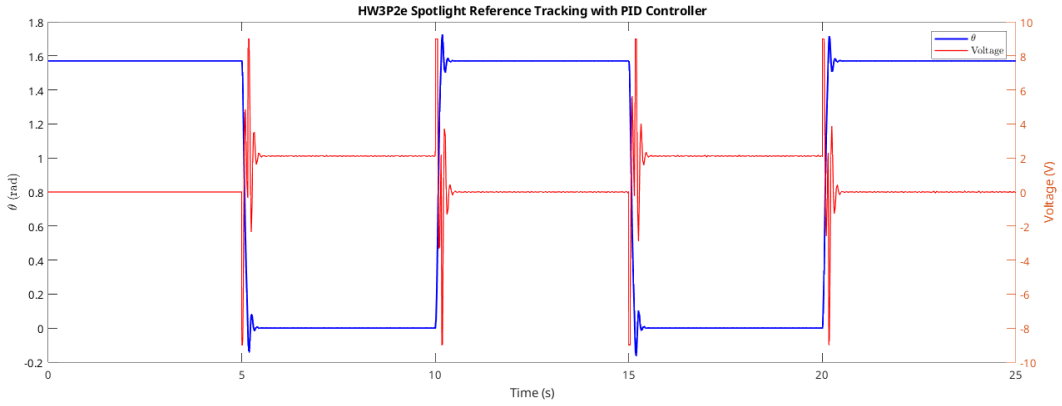
```
The transfer function of the system about the equilibrium point x1=pi/4 is:

G_s =

17867063951360000/(384829069721600*s^2 + 1794402976530432*s -
26694505514669475)

The damping ratio zeta is 0.5170
The natural frequency omega_n is 38.6825
The poles of the system are:
 -20.00 33.11i
 -20.00 -33.11i
The desired KP=33.72, KD=0.76
The pole for KI, PI=19.34
Final PID gains:
 KP=48.44
 KI=652.24
 KD=0.76
```

e: reference tracking for closed loop system after running simulation

```
x_s = [pi/2; 0]; % sim init
out = sim('akshat_hwk3_p2_sim.slx', 25);
fig = figure;
% hold on;
plot(out.tout, out.x1.data, 'b', 'LineWidth', 1.5, 'DisplayName', "$\theta$");
% plot(out.tout, out.r.data, 'k--', 'LineWidth', 1.5, 'DisplayName',
'Reference');
ylabel("$\theta$ (rad)", 'Interpreter', 'latex');
yyaxis right;
plot(out.tout, out.v.data, 'r', 'LineWidth', 1, 'DisplayName', 'Voltage');
hold on;
% yline(9, 'm--', 'LineWidth', 1.5, 'DisplayName', 'Voltage Limit (+9V)');
% yline(-9, 'm--', 'LineWidth', 1.5, 'DisplayName', 'Voltage Limit (-9V)');
ylabel("Voltage (V)");
xlabel("Time (s)");
legend('Interpreter', 'latex');
title('HW3P2e Spotlight Reference Tracking with PID Controller');
fig.Position(3:4) = [1500 500];
saveas(fig, 'figs/hw3p2e', 'svg');

function xdot = pendulum_f(t, x, u)
% x = [theta, theta_dot]
% u = [V]
% system parameters
Rm = 3.5; % Ohms, Armature resistance
Kb = 0.1; % Vs/rad, Motor back EMF constant
Kt = 0.65*Kb; % Nm/A, Motor torque constant
% Lm;  H, Armature inductance; assume 0
mp = 0.04;% kg, Spotlight(pendulum) mass
Lp = 0.1; %m, Spotlight(pendulum) length
```

```
Ip = mp*(Lp)^2;% Moment of inertia of pendulum about center of mass
Bm = 0.000008; %Nms/rad (viscous friction coefficient)
g = 9.81; %m/s^2 acceleration due to gravity

x1_dot = x(2);
x2_dot = Kt*u/(Ip*Rm) - (Kt*Kb/Rm + Bm)*x(2)/Ip - mp*g*Lp*cos(x(1))/Ip;
xdot = [x1_dot; x2_dot];
end

function A = pendulum_fdx(t, x, u)
% system parameters
Rm = 3.5; % Ohms, Armature resistance
Kb = 0.1; % Vs/rad, Motor back EMF constant
Kt = 0.65*Kb; % Nm/A, Motor torque constant
% Lm;  H, Armature inductance; assume 0
mp = 0.04;% kg, Spotlight(pendulum) mass
Lp = 0.1; %m, Spotlight(pendulum) length
Ip = mp*(Lp)^2;% Moment of inertia of pendulum about center of mass
Bm = 0.000008; %Nms/rad (viscous friction coefficient)
g = 9.81; %m/s^2 acceleration due to gravity

df1x1 = 0;
df1x2 = 1;
df2x1 = mp*g*Lp*sin(x(1))/Ip;
df2x2 = -(Kt*Kb/Rm + Bm)/Ip;
A = [df1x1 df1x2; df2x1 df2x2];
end

function xdot = pendulum_f_oc(t, x, u)
% x = [theta, theta_dot]
% u = [V]
% system parameters
% Rm = 3.5; % Ohms, Armature resistance
% Kb = 0.1; % Vs/rad, Motor back EMF constant
% Kt = 0.65*Kb; % Nm/A, Motor torque constant
% Lm;  H, Armature inductance; assume 0
mp = 0.04;% kg, Spotlight(pendulum) mass
Lp = 0.1; %m, Spotlight(pendulum) length
Ip = mp*(Lp)^2;% Moment of inertia of pendulum about center of mass
Bm = 0.000008; %Nms/rad (viscous friction coefficient)
g = 9.81; %m/s^2 acceleration due to gravity
x1_dot = x(2);
x2_dot = -Bm*x(2)/Ip - mp*g*Lp*cos(x(1))/Ip;
xdot = [x1_dot; x2_dot];
end

function plot_pend_states(t, x, title_str, ylim_1, ylim_2)
save_str = lower(strrep(title_str, ' ', '_'));
fig = figure;
sgtitle(title_str);

subplot(2, 1, 1);
plot(t, x(:, 1), "LineWidth", 1.5);
ylabel("$\theta$ (rad)", 'Interpreter', 'latex');
```

```
ylim(ylim_1);

subplot(2, 1, 2);
plot(t, x(:, 2), "LineWidth", 1.5);
ylabel("$\dot{\theta}$ (rad/s)", 'Interpreter', 'latex');
ylim(ylim_2);
xlabel("Time (s)");

saveas(fig, "figs/"+save_str, 'svg');
end

function plot_impulse_resp(t, y, title_str)
save_str = lower(strrep(title_str, ' ', '_'));
% wrap y to +-pi
% y = mod(y, 2*pi);
fig = figure;
if range(y) > 2*pi
    semilogy(t, y, "LineWidth", 1.5);
else
    plot(t, y, "LineWidth", 1.5);
end
title(title_str);
xlabel("Time (s)");
ylabel("Impulse response");
saveas(fig, "figs/"+save_str, 'svg');
end
```



Published with MATLAB® R2023b

Problem 3

In this problem, we have the dynamics of an active suspension system given at this link. The system parameters are

- States: $x = [x_b, \dot{x}_b, x_w, \dot{x}_w] \in \mathbb{R}^4$
- Inputs: $u = [F] \in \mathbb{R}$
- Reference profile: $d \in \mathbb{R}$
- Prediction horizon: $N = 50, 0.5s$
- Sampling time: $Ts = 0.01$ seconds
- Car's horizontal position: s

The system dynamics in continuous time are given by

$$\begin{aligned} \dot{x} &= A_c x + B_c [d \quad u]^T \\ A_c &\in \mathbb{R}^{4\times 4} \\ B_c &\in \mathbb{R}^{4\times 2} \end{aligned}$$

Note that the control input and disturbance profile are combined here, and their influence on the state is in the matrixc B_c. The matrices A_c, B_c are given in the code. We can discretize the system using a zero order hold and a timestep of $Ts = 0.01$ seconds. The discretized system is given by

$$\begin{aligned} x_{k+1} &= A_d x_k + B [d_k \quad u_k]^T \\ A_d &\in \mathbb{R}^{4\times 4} \\ B &\in \mathbb{R}^{4\times 2} \end{aligned}$$

where A_d, B are the discretized versions of A_c, B_c respectively. The influence of the disturbance d is contained in the first column of B, and the influence of the control input u is in the second column. Hence, we split these out into separate matrices B_d, E respectively. The dynamics of the system are now

$$\begin{aligned} x_{k+1} &= A_d x_k + B_d u_k + E d_k \\ A_d &\in \mathbb{R}^{4\times 4} \\ B_d &\in \mathbb{R}^{4\times 1} \\ E &\in \mathbb{R}^{4\times 1} \end{aligned}$$

Problem 3.a

The cost function that we want to minimize for MPC is a finite-horizon quadratic cost function given by

$$J = \frac{1}{2} \sum_{k=j}^{j+N-1} (x_{k+1}^T Q x_{k+1} + \rho u_k^2)$$

Subject to the dynamics without constraints. Q and ρ are the state and control penalties respectively, which are given in the code. The objective is to keep the body and wheel travel (x_b and x_w) as close to zero as possible while minimizing the control effort. We are given the road profile preview for the prediction horizon $[d_k, \ldots, d_{k+N-1}]$ and we need to output the control sequence $[u_k, \ldots, u_{k+N-1}]$. To formulate this as a quadratic program, we need to stack the state and control variables into a single vector and construct matrices that relate them to each other. We start with the realization that

$$
\begin{aligned}
x_{k+1} &= A_d x_k + B_d u_k + E d_k \\
x_{k+2} &= A_d x_{k+1} + B_d u_{k+1} + E d_{k+1} \\
&= A_d(A_d x_k + B_d u_k + E d_k) + B_d u_{k+1} + E d_{k+1} \\
&= A_d^2 x_k + A_d B_d u_k + A_d E d_k + B_d u_{k+1} + E d_{k+1} \\
&\vdots \\
x_{k+N} &= A_d^N x_k + A_d^{N-1} B_d u_k + A_d^{N-2} B_d u_{k+1} + \ldots + A_d B_d u_{k+N-2} + B_d u_{k+N-1} + \ldots \\
&\quad + A_d^{N-1} E d_k + A_d^{N-2} E d_{k+1} + \ldots + E d_{k+N-1}
\end{aligned}
$$

Hence, we can construct the matrices

$$
X_k = \begin{bmatrix} x_k \\ x_{k+1} \\ \vdots \\ x_{k+N-1} \end{bmatrix} \in \mathbb{R}^{4N\times 4} \qquad U_k = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-1} \end{bmatrix} \in \mathbb{R}^{N\times 1} \qquad \bar{d}_k = \begin{bmatrix} d_k \\ d_{k+1} \\ \vdots \\ d_{k+N-1} \end{bmatrix} \in \mathbb{R}^{N\times 1}
$$

$$
\bar{\mathcal{A}} = \begin{bmatrix} A_d \\ A_d^2 \\ \vdots \\ A_d^N \end{bmatrix} \in \mathbb{R}^{4N\times 4} \quad \bar{\mathcal{B}} = \begin{bmatrix} B_d & 0 & \ldots & 0 \\ A_d B_d & B_d & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_d^{N-1} B_d & A_d^{N-2} B_d & \ldots & B_d \end{bmatrix} \in \mathbb{R}^{4N\times N} \quad \bar{\mathcal{E}} = \begin{bmatrix} E & 0 & \ldots & 0 \\ A_d E & E & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_d^{N-1} E & A_d^{N-2} E & \ldots & E \end{bmatrix} \in \mathbb{R}^{4N\times N}
$$

Such that the sequence of states can be written as

$$
X_{k+1} = \bar{\mathcal{A}} x_k + \bar{\mathcal{B}} U_k + \bar{\mathcal{E}} \bar{d}_k
$$

Let the sequence without any control influence be denoted by $\mathcal{F}_k = \bar{\mathcal{A}} x_k + \bar{\mathcal{E}} \bar{d}_k$. Then

$$
\bar{X}_{k+1} = \mathcal{F}_k + \bar{\mathcal{B}} U_k
$$

Similarly, the cost matrices can be stacked into a diagonal matrices such that

$$
\begin{aligned}
\bar{Q} &= \operatorname{diag}(Q, Q, \ldots, Q) \in \mathbb{R}^{4N\times 4N} \\
\bar{R} &= \operatorname{diag}(\rho, \rho, \ldots, \rho) \in \mathbb{R}^{N\times N}
\end{aligned}
$$

Hence, the cost function becomes

$$
\begin{aligned}
J &= \frac{1}{2}(X_{k+1}^T \bar{Q} X_{k+1} + U_k^T \bar{R} U_k) \\
&= \frac{1}{2}((\mathcal{F}_k + \bar{\mathcal{B}} U_k)^T \bar{Q} (\mathcal{F}_k + \bar{\mathcal{B}} U_k) + U_k^T \bar{R} U_k) \\
&= \frac{1}{2}((\mathcal{F}_k^T + U_k^T \bar{\mathcal{B}}^T) \bar{Q} (\mathcal{F}_k + \bar{\mathcal{B}} U_k) + U_k^T \bar{R} U_k) \\
&= \frac{1}{2}(\mathcal{F}_k^T \bar{Q} \mathcal{F}_k + U_k^T \bar{\mathcal{B}}^T \bar{Q} \mathcal{F}_k + \mathcal{F}_k^T \bar{Q} \bar{\mathcal{B}} U_k + U_k^T \bar{\mathcal{B}}^T \bar{Q} \bar{\mathcal{B}} U_k + U_k^T \bar{R} U_k) \\
&= \frac{1}{2}(\mathcal{F}_k^T \bar{Q} \mathcal{F}_k + 2 U_k^T \bar{\mathcal{B}}^T \bar{Q} \mathcal{F}_k + U_k^T (\bar{\mathcal{B}}^T \bar{Q} \bar{\mathcal{B}} + \bar{R}) U_k) \text{ Since } \mathcal{F}_k^T \bar{Q} \bar{\mathcal{B}} U_k \text{ is a scalar}
\end{aligned}
$$

The solution to get the optimal control sequence U_k^* under no constraints is given by setting the derivative $\partial J / \partial U_k = 0$. Hence, we have

$$\frac{\partial J}{\partial U_k} = \bar{\mathcal{B}}^T \bar{Q} \bar{\mathcal{F}}_k + (\bar{\mathcal{B}}^T \bar{Q} \bar{\mathcal{B}} + \bar{R}) U_k = 0$$
$$\implies U_k^* = -(\bar{\mathcal{B}}^T \bar{Q} \bar{\mathcal{B}} + \bar{R})^{-1} \bar{\mathcal{B}}^T \bar{Q} \bar{\mathcal{F}}_k$$

Then, at every time step, we pick the first element of the optimal control sequence U_k^* as the control input u_k to perform MPC as per the MPC control law.

In this formula, only the matrix $\bar{\mathcal{F}}_k$ depends on the current state x_k, hence we can pre-compute everything else to get the optimal control sequence efficiently

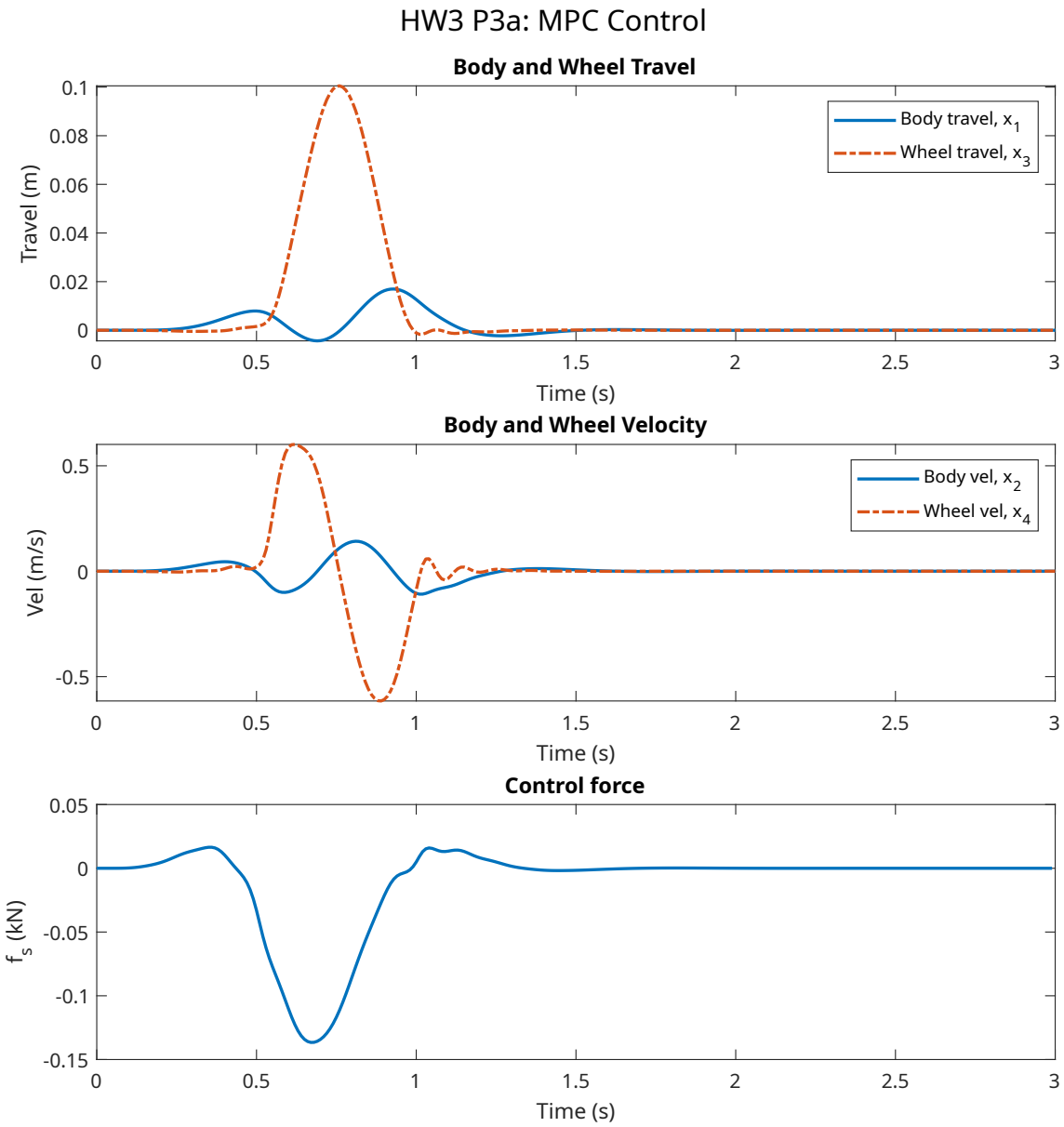


Figure 1: MPC Control

Problem 3.b

The cost function that we want to minimize for LQR is an infinite-horizon quadratic cost function given by

$$J = \frac{1}{2}\sum_{k=0}^{\infty}(x_{k+1}^T Q x_{k+1} + \rho u_k^2)$$

subject to the system dynamics without any state or control constraints. For LQR, we dont have the disturbance preview, hence the system dynamics available in the optimization routine is just

$$x_{k+1} = A_d x_k + B_d u_k$$

We can solve the Discrete time Algebraic Riccati Equation (DARE) to get the optimal control law. The optimal control law is given by

$$u_{k,opt} = -K_{opt} x_k$$

where K_{opt} is the optimal control gain matrix.

$$K_{opt} = (\rho + B_d^T P B_d)^{-1} B_d^T P A_d$$

And P is the solution to the DARE

$$-Q = A_d^T P A_d - P - A_d^T P B_d (\rho + B_d^T P B_d)^{-1} B_d^T P A_d$$

We can get K_{opt} directly by using MATLAB's `dlqr` function. This matrix does not depend on the current timestep k and just needs to be computed once. The optimal control law is then used to compute the control input at every time step.

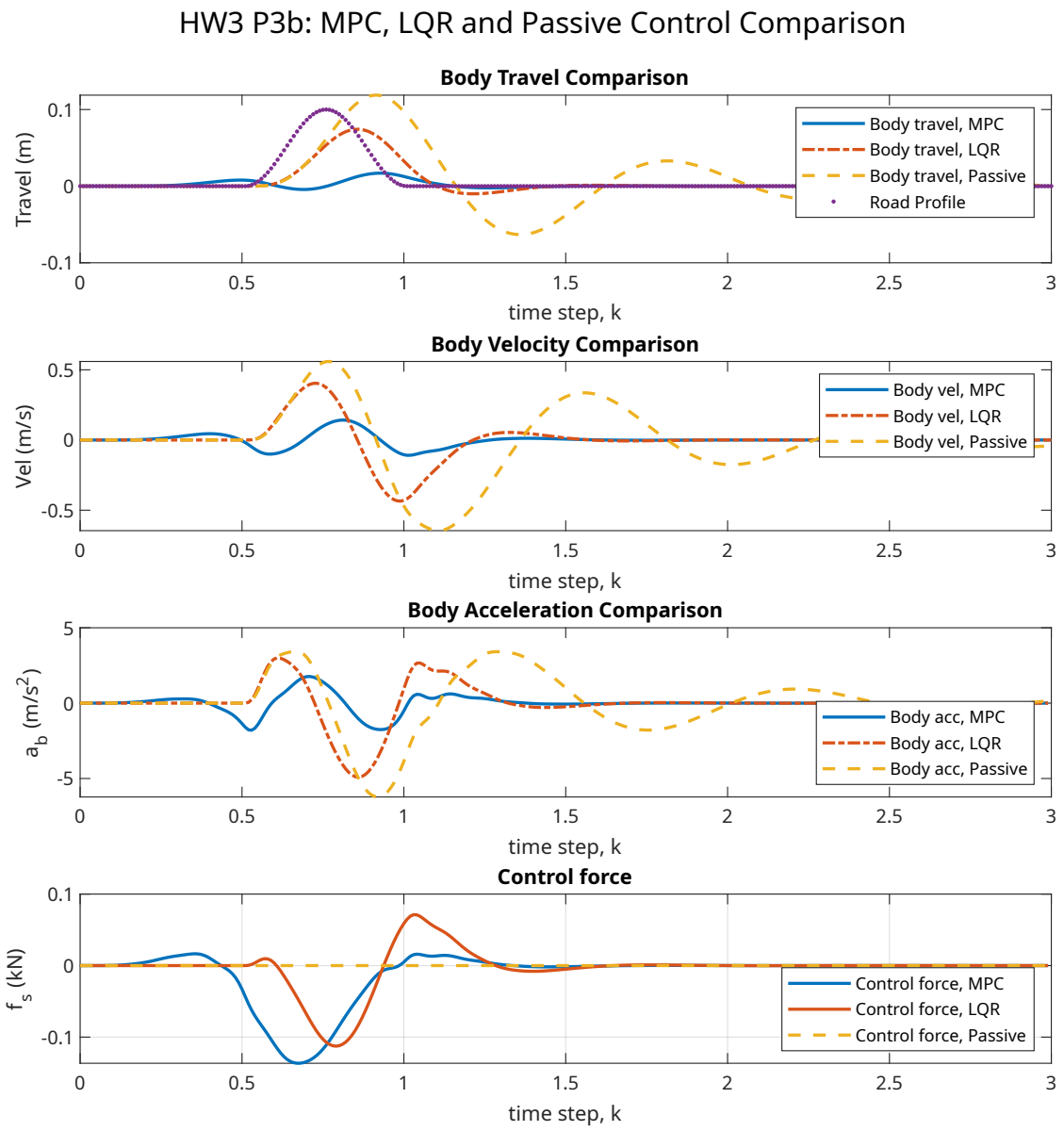


Figure 2: MPC, LQR, Passive Suspension Comparison

Problem 3.c

The plots in part (b) show that the active suspension maintains has a lower body travel, body velocity and body acceleration compared to the passive suspension. In addition, the active suspension has significantly more damped profile compared to the passive suspension. This will result in a smoother ride for the passenger in the car and the car will regain stability much faster after a disturbance in the road.

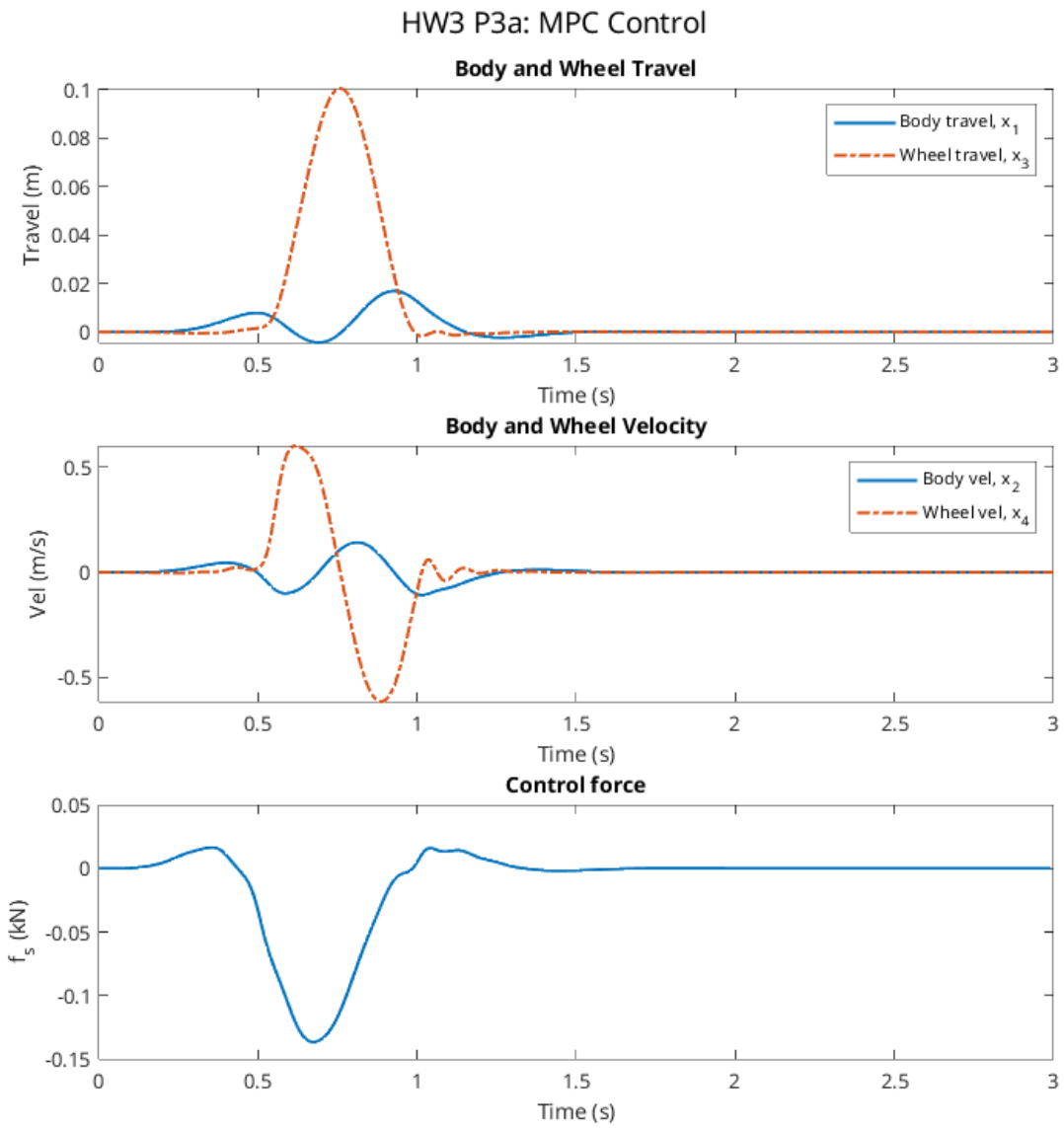
When comparing between the active suspension systems, MPC and LQR, we can see that the MPC controller is able to return the body to the nominal position, acceleration and velocity much faster than the LQR controller while having a lot less body travel, velocity and acceleration. This is due to it having the disturbance preview which enables it to predict the disturbances and compensate for them in the control input. However, we can notice that MPC has a higher control effort compared to LQR.

Table of Contents

ME599 HW3 Problem 3

```
clc; clear; close all;

% Physical parameters
mb = 300;    % kg
mw = 60;     % kg
bs = 1000;   % N/m/s
ks = 16000 ; % N/m
kt = 190000; % N/m

% Continuous-time State matrices
Ac = [ 0 1 0 0; [-ks -bs ks bs]/mb ; ...
    0 0 0 1; [ks bs -ks-kt -bs]/mw];
Bc = [ 0 0; 0 10000/mb ; 0 0 ; [kt -10000]/mw];
Cc = [1 0 0 0; 1 0 -1 0; Ac(2,:)];
Dc = [0 0; 0 0; Bc(2,:)];

Ts = 0.01; % Sampling time

% Discretize the system
[Ad, B, Cd, Dd] = c2dm(Ac, Bc, Cc, Dc, 0.01, 'zoh');
E = B(:,1);
Bd = B(:,2);

% Optimization weights
Q =diag([10 1e-2 10 1e-2]);
rho = 1;

N = 50; % Horizon

% precompute matrices
[A_bar, B_bar, E_bar, Q_bar, R_bar] = MPCMatrices(Ad,Bd,E,Q,rho,N);

% Initialization
totalTime = 3; %Simulation Time
L = totalTime/Ts; % Intervals based on sampling time

n = size(Ac,2);
xMPC = zeros(n,L+1); % For storing x
uMPC = zeros(1,L); % For storing u
road_profiles = zeros(1,L); % For storing road profile

x = xMPC(:,1); % Initial x
s = 0; % Initial s
u = 0; [~,road_profile]= activeSuspSim (x,u,s,N,totalTime); %inital
```

```
disturbance profile
road_profiles(1) = road_profile(1);

% Simulation
for j = 1:L

    %MPC Input
    u = MPCCtrl(x, road_profile, A_bar, B_bar, E_bar, Q_bar, R_bar);
    uMPC(j) = u;

    % Feed input to simulator
    [x,road_profile]= activeSuspSim (x,u,s,N,totalTime);
    xMPC(:,j+1) = x;
    road_profiles(j+1) = road_profile(1);

    % Increase position by a step
    s = j/10;
end

%Plots
k = (0:L)*Ts; % ...in seconds

fig = figure(1);
sgtitle('HW3 P3a: MPC Control')

subplot(3,1,1)
plot(k,xMPC(1,:),'-',k, xMPC(3,:),'-.','LineWidth',1.5);
xlabel('Time (s)');ylabel('Travel (m)');
legend ('Body travel, x_1','Wheel travel, x_3')
title ('Body and Wheel Travel')

subplot(3,1,2);
plot(k,xMPC(2,:),'-',k,xMPC(4,:),'-.','LineWidth',1.5);
xlabel('Time (s)'),ylabel('Vel (m/s)');
legend ('Body vel, x_2','Wheel vel, x_4')
title ('Body and Wheel Velocity')

subplot(3,1,3)
plot(k(1:end-1),uMPC,'LineWidth',1.5);
title('Control force')
xlabel('Time (s)'), ylabel('f_s (kN)');

fig.Position(3:4) = [800 800];
saveas(fig,'figs/hw3p3a.svg');
```



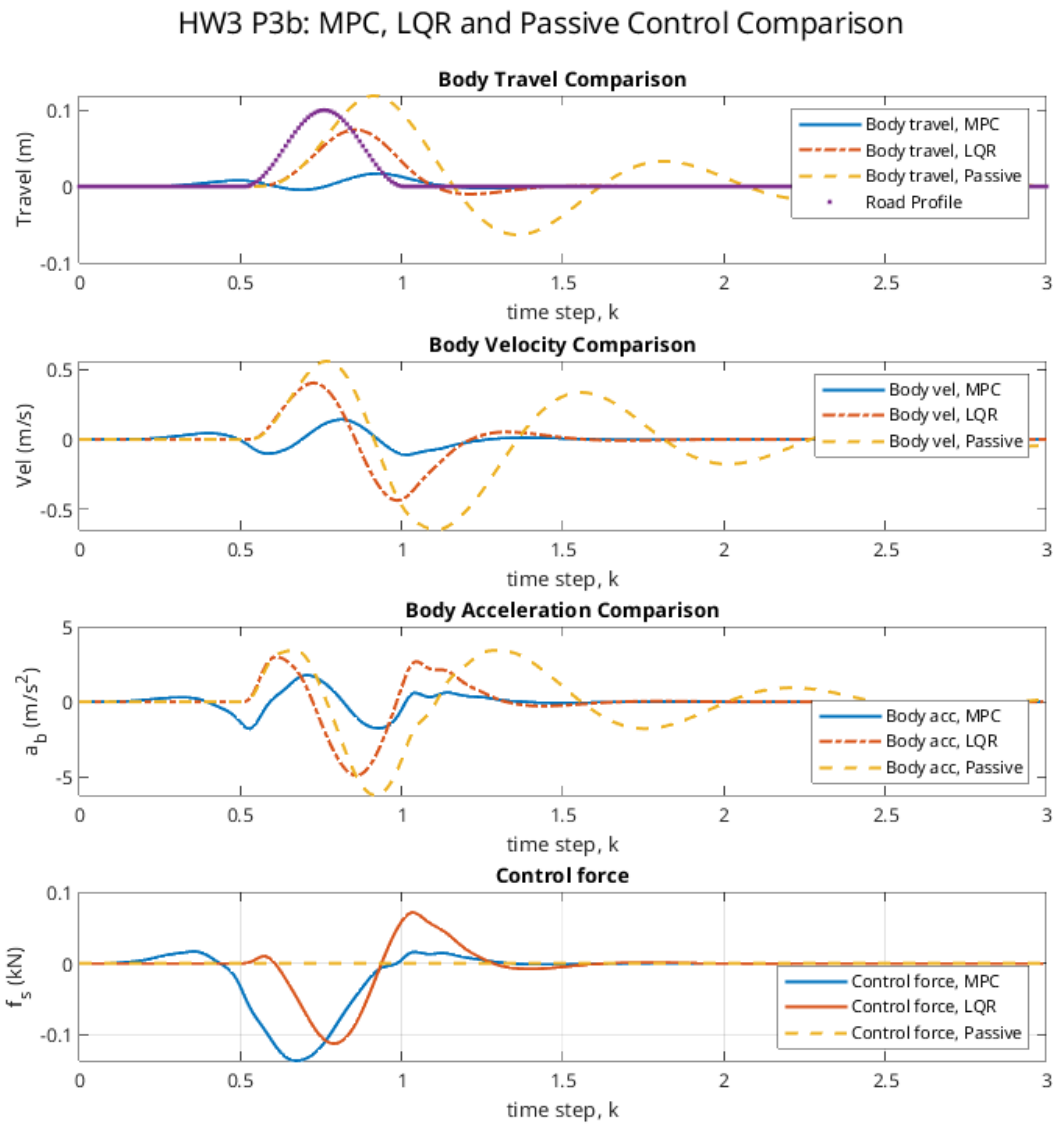
Simulate LQR and Active Suspension Turned Off

Initialization LQR

```
xLQR = zeros(n,L+1);
uLQR = zeros(1,L);

% compute K_opt
[K_opt,~,~] = dlqr(Ad,Bd,Q,rho);

%Passive Suspension (No Control)
```

```
xNC = zeros(n,L+1);
uNC = zeros(1,L);

% Initial x
x = xLQR(:,1);
xnc = xNC(:,1);

s = 0; % Initial s

for j = 1:L

    % LQR Control
    u = LQRCtrl(x, K_opt);
    uLQR(j) = u;
    x= activeSuspSim (x,u,s,N,totalTime);
    xLQR(:,j+1) = x;

    %No control
    xnc= activeSuspSim (xnc,0,s,N,totalTime);
    xNC(:,j+1) = xnc;

    % Increase position by a step
    s = j/10;
end

% Compute Body acceleration
%MPC
yMPC = Cc*xMPC(:,1:end-1) + Dc(:,2)*uMPC; % Output
abMPC = yMPC(3,:); % Body acceleration

% LQR
yLQR = Cc*xLQR(:,1:end-1) + Dc(:,2)*uLQR; % Output
abLQR = yLQR(3,:);% Body acceleration

% NC
yNC = Cc*xNC(:,1:end-1) + Dc(:,2)*uNC; % Output
abNC = yNC(3,:);% Body acceleration


fig = figure(2);
sgtitle('HW3 P3b: MPC, LQR and Passive Control Comparison');

subplot(4,1,1);
plot(k,xMPC(1,:),'-',k,xLQR(1,:),'-.',k,xNC(1,:),'--',k,road_profiles,'.','Lin
eWidth',1.5);
xlabel('time step, k'),ylabel('Travel (m)');
legend ('Body travel, MPC','Body travel, LQR', 'Body travel, Passive', 'Road
Profile')
title ('Body Travel Comparison')
% You are to add a plot of the Road Profile to this subplot

subplot(4,1,2);
plot(k,xMPC(2,:),'-',k,xLQR(2,:),'-.',k,xNC(2,:),'--','LineWidth',1.5);
xlabel('time step, k'),ylabel('Vel (m/s)');
```

```
legend ('Body vel, MPC','Body vel, LQR', 'Body vel, Passive')
title ('Body Velocity Comparison')

subplot(4,1,3)
plot(k(1:end-1),abMPC,'-',k(1:end-1),abLQR,'-.',k(1:end-1),abNC,'--','LineWidt
h',1.5);
title('Body Acceleration Comparison')
xlabel('time step, k'), ylabel('a_b (m/s^2)')
legend ('Body acc, MPC','Body acc, LQR', 'Body acc, Passive',
'Location','southeast');

subplot(4,1,4)
plot(k(1:end-1),uMPC,k(1:end-1),uLQR,k(1:end-1),uNC,'--','LineWidth',1.5);
title('Control force')
xlabel('time step, k'), ylabel('f_s (kN)');
legend ('Control force, MPC','Control force, LQR', 'Control force, Passive',
'Location','southeast'); grid on

fig.Position(3:4) = [800 800];
saveas(fig,'figs/hw3p3b.svg');
```




```
function u = MPCCtrl(x,d_traj, A_bar, B_bar, E_bar, Q_bar, R_bar)
% Input:x and road profile
% Output: MPC input

% u = -(x(1) + d_traj'*d_traj);
F_k = A_bar*x + E_bar*d_traj;
U_k = -inv(B_bar'*Q_bar*B_bar + R_bar)*B_bar'*Q_bar*F_k;

% take first control input
u = U_k(1);

end

function u = LQRCtrl(x, K_opt)
```

```
% Input:x
% Output: LQR control input

% u = - x(1);
u = -K_opt*x;

end

function [A_bar, B_bar, E_bar, Q_bar, R_bar] = MPCMatrices(A,B,E,Q,R,N)
nx = size(A,2);
nu = size(B,2);
nd = size(E,2);

% initialize matrices
A_bar = zeros(N*nx,nx);
B_bar = zeros(N*nx,N*nu);
E_bar = zeros(N*nx,N*nd);
Q_bar = zeros(N*nx,N*nx);
R_bar = zeros(N*nu,N*nu);

% Compute first row of A_bar then use it to initialize the rest
A_bar(1:nx,:) = A;
for i = 2:N
    A_bar((i-1)*nx+1 : i*nx, :) = A_bar((i-2)*nx+1:(i-1)*nx,:)*A;
end

% Compute first column of B_bar then use it to initialize the rest
for i = 1:N
    B_bar((i-1)*nx+1 : i*nx, 1:nu) = A^(i-1)*B;
end
for i = 2:N
    zero_rows = (i-1)*nx;
    zero_cols = nu;
    B_bar(:, (i-1)*nu+1 : i*nu) = [zeros(zero_rows,zero_cols); B_bar(1:end-
zero_rows, 1:nu)];
end

% Compute first column of E_bar then use it to initialize the rest
for i = 1:N
    E_bar((i-1)*nx+1 : i*nx, 1:nd) = A^(i-1)*E;
end
for i = 2:N
    zero_rows = (i-1)*nx;
    zero_cols = nd;
    E_bar(:, (i-1)*nd+1 : i*nd) = [zeros(zero_rows,zero_cols); E_bar(1:end-
zero_rows, 1:nd)];
end

% Compute Q_bar
for i = 1:N
    Q_bar((i-1)*nx+1 : i*nx, (i-1)*nx+1 : i*nx) = Q;
end

% Compute R_bar
```

```
for i = 1:N
    R_bar((i-1)*nu+1 : i*nu, (i-1)*nu+1 : i*nu) = R;
end

end
```

Published with MATLAB® R2023b