Dynamic Tracking of a Double Pendulum: A Focused Analysis on Extended, Unscented, and Adaptive Filter Techniques for Nonlinear State Estimation

Mehdi Muzaffari

November 2024

"The career of a young theoretical physicist consists of treating the harmonic oscillator in ever-increasing levels of abstraction."

– Sidney Coleman

1 Abstract

The double pendulum is a fascinating dynamical system capable of producing chaotic trajectories, which exhibit sensitive dependence on the initial conditions. Due to its highly nonlinear and chaotic nature, it presents extensive challenges for accurate state estimation, originating both from sensitivity to initial conditions and the inherent nonlinearity of its governing equations. Using the double pendulum as the study case, the project aims to investigate three state estimation methods: the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF), and the Adaptive Unscented Kalman Filter (AUKF). By comparing these techniques, the project will examine their strengths and limitations for real-world non-linear systems in the applications of robotics and control algorithm developments.

2 Introduction

At each time step, the EKF linearizes the system around the current estimate, making it an efficient method but with the cost of becoming less reliable as the nonlinearity in the system becomes larger. The UKF propagates a set of carefully chosen sample points (sigma points) through the dynamic system, avoiding linearization and therefore increasing accuracy in highly nonlinear regions. To better handle a real-world system with time-varying uncertainties, AUKF adjusts the process and measurement noise covariances in real time while still maintaining a Gaussian approximation.

3 Extended Kalman Filter (EKF)

Kalman filtering is a state estimation method for linear systems, it provides optimal estimates when the system dynamic and measurement models are linear. However, real-world systems, like the double pendulum are inherently nonlinear, making Kalman filtering techniques challenging to apply directly. EKF which is an extension of the Kalman filter is designed for such nonlinear systems [3, 2]. A quick overview of EKF approach:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1})$$

$\hat{x}_k^-$: predicted state estimate
$f()$: state transition function

$\hat{x}_{k-1}$: previous state estimate
u_{k-1}: control input

$$P_k^- = F_k P_{k-1} F_k^\top + Q_k$$

P_k^-: predicted covariance
F_k: jacobian of $f()$
P_{k-1}: previous covariance
Q_k: process noise

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}}$$

$$\hat{z}_k = h(\hat{x}_k^-)$$

$\hat{z}_k$: predicted measurement
$h()$: measurement function

$$y_k = z_k - \hat{z}_k$$

y_k: innovation
z_k: actual measurement

$$S_k = H_k P_k^- H_k^\top + R_k$$

S_k: residual covariance
H_k: jacobian of $h()$
R_k: measurement noise

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k^-}$$

$$K_k = P_k^- H_k^\top S_k^{-1}$$

K_k: kalman gain

$$\hat{x}_k = \hat{x}_k^- + K_k y_k$$

$\hat{x}_k$: updated state estimate

$$P_k = (I - K_k H_k) P_k^-$$

P_k: updated covariance
I: identity matrix

4 Unscented Kalman Filter (UKF)

To overcome the limitations of EKF, UKF and AUKF will be employed. UKF selects a set of sigma points that capture the mean and covariance of the state distribution [3, 2]. AUKF adjusts the process and measurement noise covariances in real time. A quick overview of UKF approach:

initial:

$$\hat{x}_0, \quad P_0$$

$$\chi_i = \hat{x} + \begin{cases} 0 & \text{for } i = 0, \\ \pm\sqrt{(n+\lambda)P} & \text{for } i = 1, \ldots, 2n. \end{cases}$$

where:

$$\lambda = \alpha^2(n + \kappa) - n$$

$$W_0^{(m)} = \frac{\lambda}{n+\lambda}, \quad W_0^{(c)} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta),$$

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2(n+\lambda)} \quad \text{for } i = 1, \ldots, 2n.$$

State prediction:

$$\chi_i^- = f(\chi_i, u)$$

$$\hat{x}_k^- = \sum_{i=0}^{2n} W_i^{(m)} \chi_i^-$$

Covariance prediction:

$$P_k^- = \sum_{i=0}^{2n} W_i^{(c)} \left(\chi_i^- - \hat{x}_k^-\right) \left(\chi_i^- - \hat{x}_k^-\right)^\top + Q_k$$

Measurement prediction:

$$\zeta_i = h(\chi_i^-)$$

$$\hat{z}_k = \sum_{i=0}^{2n} W_i^{(m)} \zeta_i$$

Measurement covariance:

$$S_k = \sum_{i=0}^{2n} W_i^{(c)} \left(\zeta_i - \hat{z}_k\right) \left(\zeta_i - \hat{z}_k\right)^\top + R_k$$

Cross-covariance:

$$P_{xz} = \sum_{i=0}^{2n} W_i^{(c)} \left(\chi_i^- - \hat{x}_k^-\right) \left(\zeta_i - \hat{z}_k\right)^\top$$

Kalman gain:

$$K_k = P_{xz} S_k^{-1}$$

State update:

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - \hat{z}_k\right)$$

Covariance update:

$$P_k = P_k^- - K_k S_k K_k^\top$$

5 Real World Application

The double pendulum is profoundly used to study chaos and state transitions, both experimentally and numerically. It is also widely used as a modeling tool, for example, in the locomotion of the human leg swing in biomechanics, flexible robotic arms, and self-driven robotic carts in various robotic fields [5, 6, 1]. The double pendulum is also been used to test model discovery algorithms due to its complicated and rich dynamics [4, 7].

6 Objective

The main objective of this project is to compare the difference in performance of Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), and Adaptive Unscented Kalman Filter (AUKF) in the context of double pendulum as the case study. The project will evaluate the accuracy, computational cost, and robustness of each method for different scenarios. By doing so, the goal is to identify which technique works best for various noise levels and system nonlinearity.

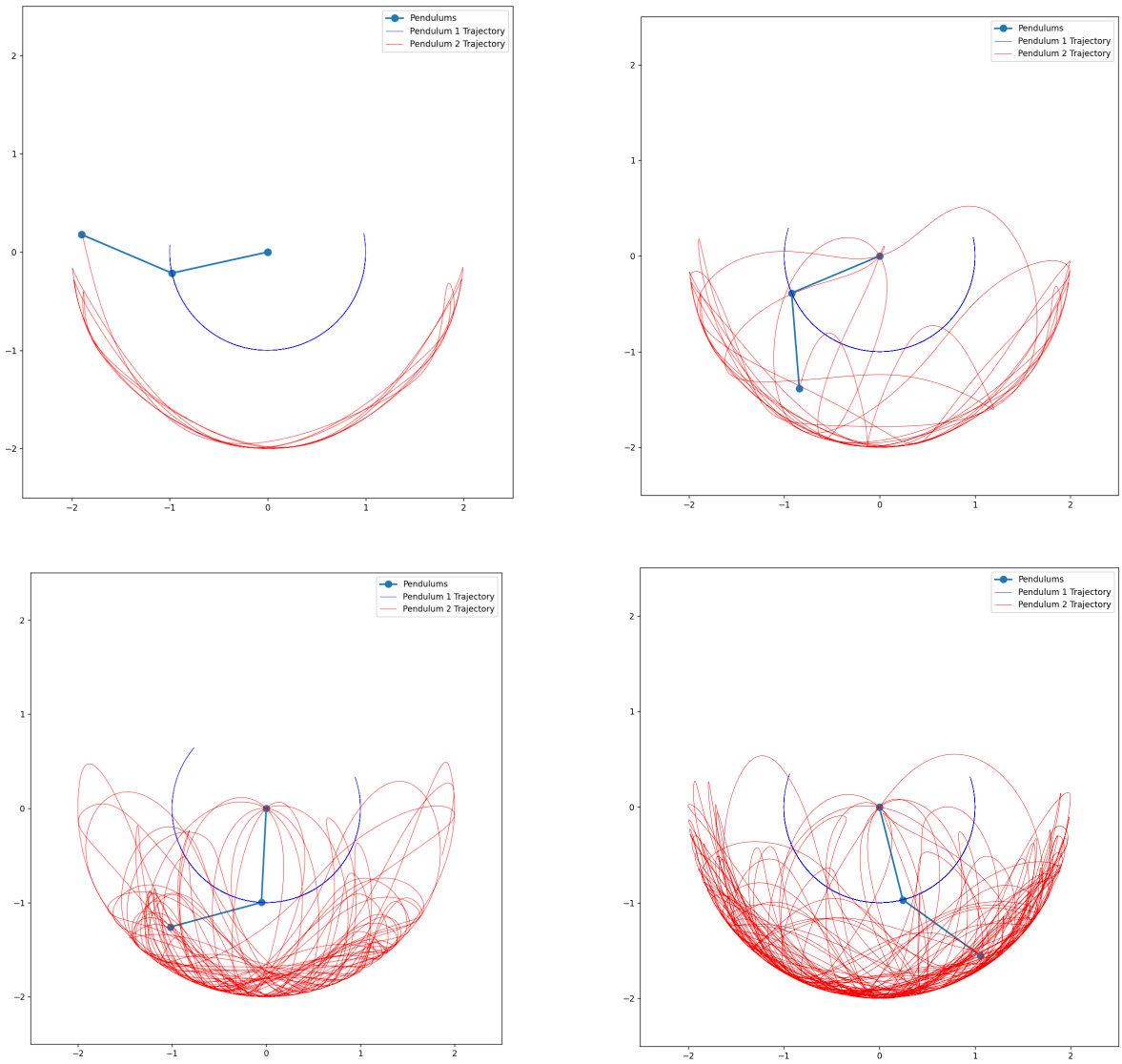


Figure 1: Trajectory with different initials

7 Technical Plan

Double pendulum is a highly nonlinear dynamical system. It consists of two pendulums, connected end-to-end, with each having the same masses and lengths (for simplicity). Given certain initial conditions, it exhibits chaotic motion making it an ideal candidate for testing nonlinear estimation techniques.

8 System Description

The double pendulum's dynamics are governed by nonlinear differential equations, which depend on the coupling between the two pendulums as well as some trigonometric functions. The goal is to test the filter's ability to track the system's states, θ_1, θ_2, $\dot{\theta}_1$, $\dot{\theta}_2$ from a set of noisy measurements.

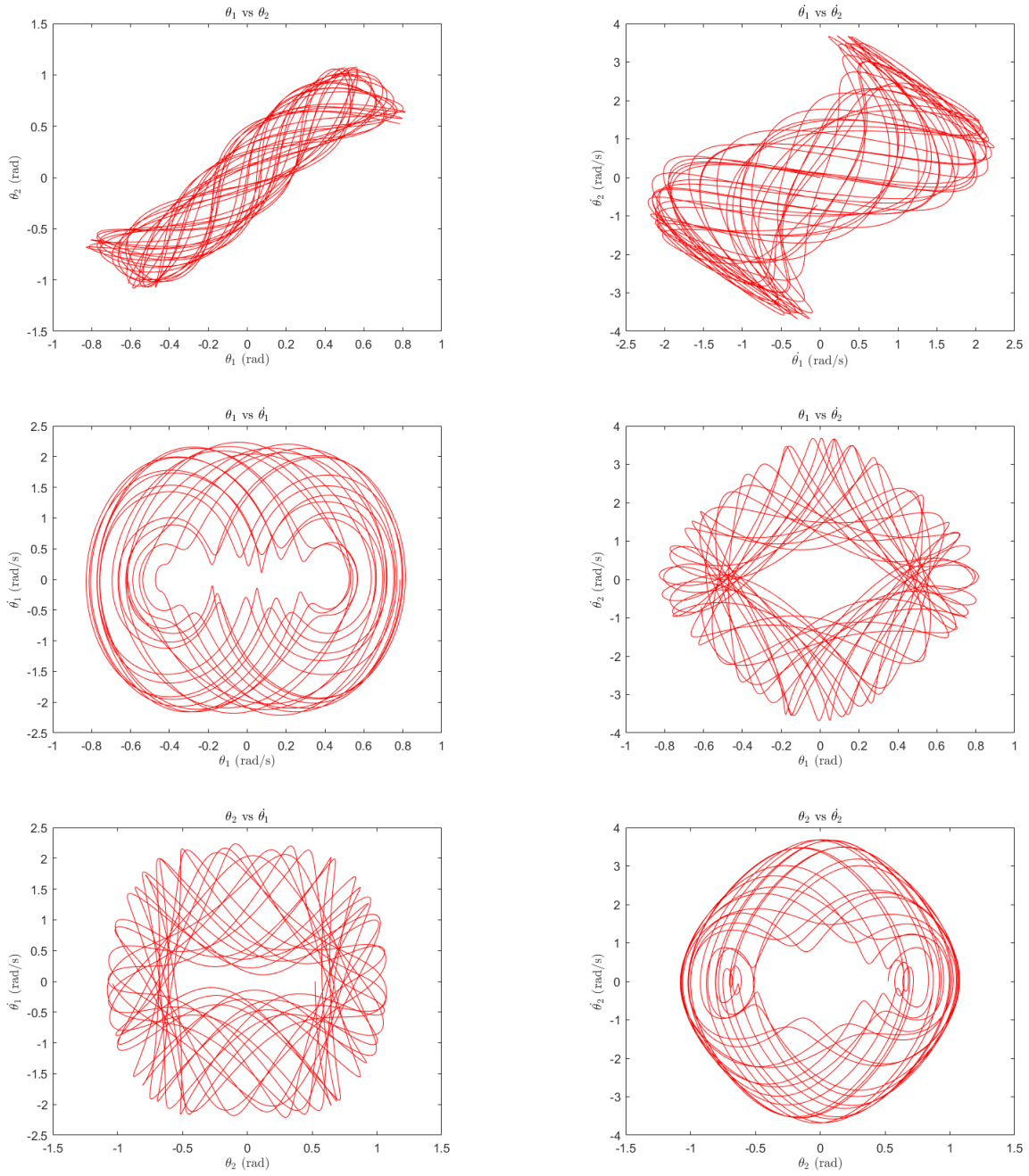


Figure 2: Relative trajectories

1. Kinematic constraints:

$$y_1 = -L_1 \cos(\theta_1)$$
$$x_1 = L_1 \sin(\theta_1)$$
$$y_2 = -L_1 \cos(\theta_1) - L_2 \cos(\theta_2)$$
$$x_2 = L_1 \sin(\theta_1) + L_2 \sin(\theta_2)$$
$$\dot{y}_1 = \dot{\theta}_1 L_1 \sin(\theta_1)$$
$$\dot{x}_1 = \dot{\theta}_1 L_1 \cos(\theta_1)$$

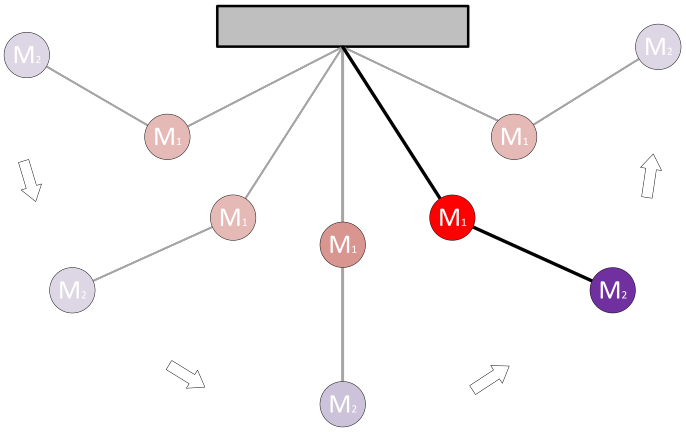


Figure 3: Double pendulum system

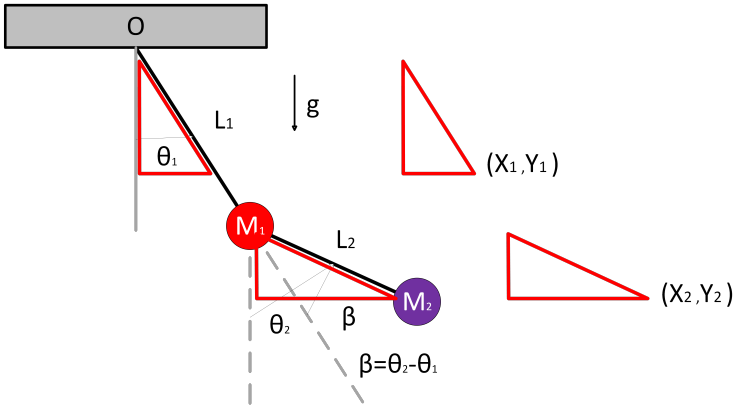


Figure 4: Kinematic constraints

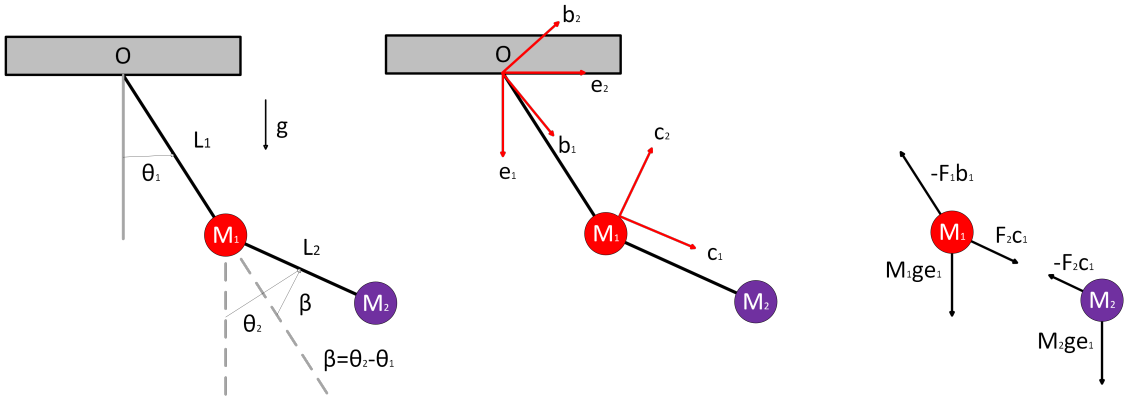


Figure 5: Free body diagram

$$\dot{y}_2 = \dot{\theta}_1 L_1 \sin(\theta_1) + \dot{\theta}_2 L_2 \sin(\theta_2)$$
$$\dot{x}_2 = \dot{\theta}_1 L_1 \cos(\theta_1) + \dot{\theta}_2 L_2 \cos(\theta_2)$$

2. Lagrangian of system:

$$\mathcal{L} = T - V$$
$$T = T_1 + T_2$$
$$T_1 = \frac{1}{2} M_1 v_1^2 \quad T_2 = \frac{1}{2} M_2 v_2^2$$
$$T_1 = \frac{1}{2} M_1 (\dot{x}_1^2 + \dot{y}_1^2) \quad T_2 = \frac{1}{2} M_2 (\dot{x}_2^2 + \dot{y}_2^2)$$
$$T_1 = \frac{1}{2} M_1 \left(\dot{\theta}_1^2 L_1^2 \cos^2(\theta_1) + \dot{\theta}_1^2 L_1^2 \sin^2(\theta_1) \right) = \frac{1}{2} M_1 \dot{\theta}_1^2 L_1^2$$
$$\sin^2 x + \cos^2 x = 1$$
$$T_2 = \frac{1}{2} M_2 \left[\left(\dot{\theta}_1 L_1 \cos\theta_1 + \dot{\theta}_2 L_2 \cos\theta_2 \right)^2 + \left(\dot{\theta}_1 L_1 \sin\theta_1 + \dot{\theta}_2 L_2 \sin\theta_2 \right)^2 \right]$$
$$T_2 = \frac{1}{2} M_2 \left[\dot{\theta}_1^2 L_1^2 + \dot{\theta}_2^2 L_2^2 + 2\dot{\theta}_1 \dot{\theta}_2 L_1 L_2 \cos(\theta_1 - \theta_2) \right]$$
$$\sin(x)\sin(y) + \cos(x)\cos(y) = \cos(x - y)$$
$$T = \frac{1}{2} M_1 \dot{\theta}_1^2 L_1^2 + \frac{1}{2} M_2 \left[\dot{\theta}_1^2 L_1^2 + \dot{\theta}_2^2 L_2^2 + 2\dot{\theta}_1 \dot{\theta}_2 L_1 L_2 \cos(\theta_1 - \theta_2) \right]$$
$$V = V_1 + V_2$$
$$V_1 = M_1 g h = -M_1 g L_1 \cos\theta_1$$
$$V_2 = M_2 g h = -M_2 g (L_1 \cos\theta_1 + L_2 \cos\theta_2)$$
$$V = -M_1 g L_1 \cos\theta_1 - M_2 g (L_1 \cos\theta_1 + L_2 \cos\theta_2)$$
$$\mathcal{L} = \frac{1}{2} M_1 \dot{\theta}_1^2 L_1^2 + \frac{1}{2} M_2 \left[\dot{\theta}_1^2 L_1^2 + \dot{\theta}_2^2 L_2^2 + 2\dot{\theta}_1 \dot{\theta}_2 L_1 L_2 \cos(\theta_1 - \theta_2) \right]$$
$$+ M_1 g L_1 \cos\theta_1 + M_2 g (L_1 \cos\theta_1 + L_2 \cos\theta_2)$$

3. Solve Euler-Lagrange equation:

$$\frac{\partial \mathcal{L}}{\partial \theta_1} - \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = 0$$
$$\frac{\partial \mathcal{L}}{\partial \theta_1} = -\dot{\theta}_1 \dot{\theta}_2 M_2 L_1 L_2 \sin(\theta_1 - \theta_2) - (M_1 + M_2) g L_1 \sin\theta_1$$
$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = (M_1 + M_2) \dot{\theta}_1 L_1^2 + M_2 \dot{\theta}_2 L_1 L_2 \cos(\theta_1 - \theta_2)$$
$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} =$$
$$(M_1 + M_2) \ddot{\theta}_1 L_1^2 + M_2 L_1 L_2 \left[\ddot{\theta}_2 \cos(\theta_1 - \theta_2) - \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) + \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) \right]$$
$$\frac{\partial \mathcal{L}}{\partial \theta_2} - \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = 0$$
$$\frac{\partial \mathcal{L}}{\partial \theta_2} = \dot{\theta}_1 \dot{\theta}_2 M_2 L_1 L_2 \sin(\theta_1 - \theta_2) - M_2 g L_2 \sin\theta_2$$
$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = M_2 \dot{\theta}_2 L_2^2 + M_2 \dot{\theta}_1 L_1 L_2 \cos(\theta_1 - \theta_2)$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} =$$

$$M_2\ddot{\theta}_2 L_2^2 + M_2 L_1 L_2 \left[\ddot{\theta}_1 \cos(\theta_1 - \theta_2) - \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + \dot{\theta}_1\dot{\theta}_2 \sin(\theta_1 - \theta_2)\right]$$

4. Solve for equations of motion:

$$\ddot{\theta}_1 = \frac{g\left(\mu_{M_2}\cos(\theta_1-\theta_2)\sin\theta_2 - \sin\theta_1\right) + \mu_{M_2} l_1 \sin(\theta_1-\theta_2)\cos(\theta_1-\theta_2)\,\dot{\theta}_1^2 + \mu_{M_2} l_2 \sin(\theta_1-\theta_2)\,\dot{\theta}_2^2}{l_1\left(\mu_{M_1} + \mu_{M_2}\sin^2(\theta_1-\theta_2)\right)}$$

$$\ddot{\theta}_2 = -\frac{l_1 \sin(\theta_1-\theta_2)\,\dot{\theta}_1^2 + l_2\mu_{M_2}\sin(\theta_1-\theta_2)\cos(\theta_1-\theta_2)\,\dot{\theta}_2^2 + g\cos\theta_1\sin(\theta_1-\theta_2)}{l_2\left(\mu_{M_1} + \mu_{M_2}\sin^2(\theta_1-\theta_2)\right)}$$

$$\mu_{M_1} = \frac{M_1}{M_2 + M_1} \quad \text{and} \quad \mu_{M_2} = \frac{M_2}{M_2 + M_1}$$

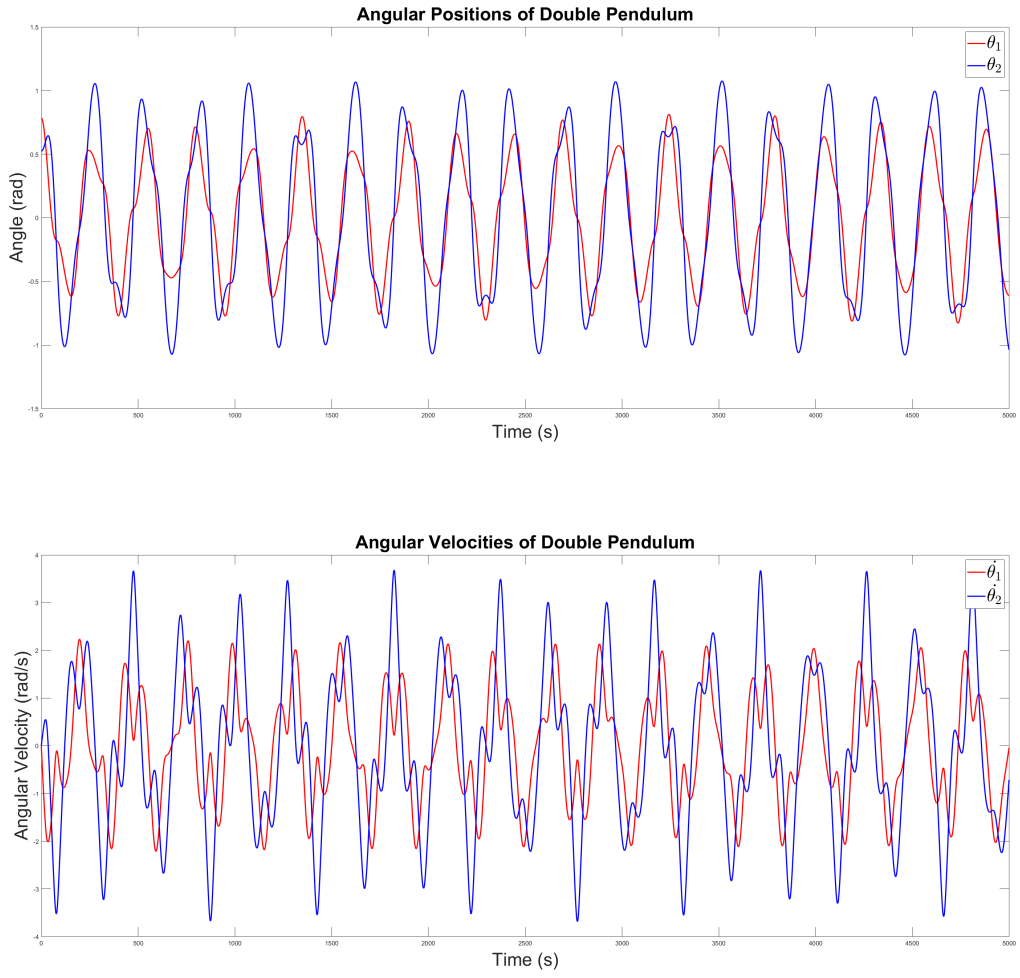


Figure 6: Dynamics of double pendulum

9 Estimation Technique (EKF)

EKF approximates the nonlinear behavior as a linear one [3, 2] by computing the Jacobian matrix of the system at each time step.

1. State-space:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix}$$

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \frac{g\left(\mu_{M_2}\cos(x_3 - x_1)\sin x_3 - \sin x_1\right) + \mu_{M_2} l_1 \sin(x_3 - x_1)\cos(x_3 - x_1)\, x_2^2 + \mu_{M_2} l_2 \sin(x_3 - x_1)\, x_4^2}{l_1\left(\mu_{M_1} + \mu_{M_2}\sin^2(x_3 - x_1)\right)}$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = -\frac{l_1 \sin(x_3 - x_1)\, x_2^2 + l_2 \mu_{M_2} \sin(x_3 - x_1)\cos(x_3 - x_1)\, x_4^2 + g\cos x_1 \sin(x_3 - x_1)}{l_2\left(\mu_{M_1} + \mu_{M_2}\sin^2(x_3 - x_1)\right)}$$

2. In discrete-time:

$$\hat{\mathbf{x}}_\mathbf{k}^- = \hat{\mathbf{x}}_{\mathbf{k-1}} + \Delta t \cdot \mathbf{f}(\hat{\mathbf{x}}_{\mathbf{k-1}}) + \mathbf{w}_{\mathbf{k-1}}$$

$$\begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix}_{\hat{x}_k^-} = \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix}_{\hat{x}_{k-1}} + \Delta t \cdot \begin{bmatrix} x_2 \\ \frac{g\left(\mu_{M_2}\cos(x_3 - x_1)\sin x_3 - \sin x_1\right) + \mu_{M_2} l_1 \sin(x_3 - x_1)\cos(x_3 - x_1)\, x_2^2 + \mu_{M_2} l_2 \sin(x_3 - x_1)\, x_4^2}{l_1\left(\mu_{M_1} + \mu_{M_2}\sin^2(x_3 - x_1)\right)} \\ x_4 \\ -\frac{l_1 \sin(x_3 - x_1)\, x_2^2 + l_2 \mu_{M_2} \sin(x_3 - x_1)\cos(x_3 - x_1)\, x_4^2 + g\cos x_1 \sin(x_3 - x_1)}{l_2\left(\mu_{M_1} + \mu_{M_2}\sin^2(x_3 - x_1)\right)} \end{bmatrix} + \begin{bmatrix} w_{\theta_1} \\ w_{\dot{\theta}_1} \\ w_{\theta_2} \\ w_{\dot{\theta}_2} \end{bmatrix}_{\hat{x}_{k-1}}$$

3. Measurement model:

$$\mathbf{z}_{k-1} = \mathbf{h}(\mathbf{x}_{k-1}) + \mathbf{v}_{k-1}$$

assume measuring all four states directly:

$$\mathbf{z}_{k-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} v_{\theta_1} \\ v_{\dot{\theta}_1} \\ v_{\theta_2} \\ v_{\dot{\theta}_2} \end{bmatrix}$$

$$R = \begin{bmatrix} \sigma_{\theta_1}^2 & 0 & 0 & 0 \\ 0 & \sigma_{\dot{\theta}_1}^2 & 0 & 0 \\ 0 & 0 & \sigma_{\theta_2}^2 & 0 \\ 0 & 0 & 0 & \sigma_{\dot{\theta}_2}^2 \end{bmatrix}$$

4. Jacobian of transition function:

$$F_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1|k-1}}$$

$$f_1(x_1, x_2, x_3, x_4) =$$

$$\frac{g\left(\mu_{M_2}\cos(x_3 - x_1)\sin x_3 - \sin x_1\right) + \mu_{M_2} l_1 \sin(x_3 - x_1)\cos(x_3 - x_1)\, x_2^2 + \mu_{M_2} l_2 \sin(x_3 - x_1)\, x_4^2}{l_1\left(\mu_{M_1} + \mu_{M_2}\sin^2(x_3 - x_1)\right)}$$

$$f_2(x_1, x_2, x_3, x_4) =$$

$$-\frac{l_1 \sin(x_3 - x_1)\, x_2^2 + l_2 \mu_{M_2} \sin(x_3 - x_1)\cos(x_3 - x_1)\, x_4^2 + g\cos x_1 \sin(x_3 - x_1)}{l_2\left(\mu_{M_1} + \mu_{M_2}\sin^2(x_3 - x_1)\right)}$$

$$F_k = \begin{bmatrix} \frac{\partial x_2}{\partial x_1} & \frac{\partial x_2}{\partial x_2} & \frac{\partial x_2}{\partial x_3} & \frac{\partial x_2}{\partial x_4} \\ \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial x_4}{\partial x_1} & \frac{\partial x_4}{\partial x_2} & \frac{\partial x_4}{\partial x_3} & \frac{\partial x_4}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \end{bmatrix}$$

$$\begin{pmatrix} \text{1st column} & \text{2nd column} & \text{3rd column} & \text{4th column} \\ \text{1st column} & \text{2nd column} & \text{3rd column} & \text{4th column} \\ \text{1st column} & \text{2nd column} & \text{3rd column} & \text{4th column} \\ \text{1st column} & \text{2nd column} & \text{3rd column} & \text{4th column} \end{pmatrix}$$

1st column:
$$0,$$
$$\sigma_5 - \frac{g\left(\cos(x_1) + \sigma_2\right) + \sigma_1 + \sigma_{10} - \sigma_8}{\sigma_{12}},$$
$$0,$$
$$\frac{\sigma_4 + \sigma_9 - \sigma_7 + \sigma_3 - g\sin(x_1)\sin(x_1 - x_3)}{\sigma_{11}} - \sigma_6$$

2nd column:
$$1,$$
$$-\frac{2\mu_{\mathrm{M2}} x_2 \cos(x_1 - x_3)\sin(x_1 - x_3)}{\mu_{\mathrm{M2}}\sigma_{15} + \mu_{\mathrm{M1}}},$$
$$0,$$
$$\frac{2 l_1 x_2 \sin(x_1 - x_3)}{\sigma_{11}}$$

3rd column:
$$0,$$
$$\frac{g\left(\mu_{\mathrm{M2}}\cos(x_1 - x_3)\cos(x_3) + \sigma_2\right) + \sigma_1 + \sigma_{10} - \sigma_8}{\sigma_{12}} - \sigma_5,$$
$$0,$$
$$\sigma_6 - \frac{\sigma_4 + \sigma_9 - \sigma_7 + \sigma_3}{\sigma_{11}}$$

4th column:
$$0,$$
$$-\frac{2 l_2 \mu_{\mathrm{M2}} x_4 \sin(x_1 - x_3)}{\sigma_{12}},$$
$$1,$$
$$\frac{2\mu_{\mathrm{M2}} x_4 \cos(x_1 - x_3)\sin(x_1 - x_3)}{\mu_{\mathrm{M2}}\sigma_{15} + \mu_{\mathrm{M1}}}$$

$$
\begin{aligned}
\sigma_1 &= l_2 \mu_{\mathrm{M2}} x_4^2 \cos(x_1 - x_3) \\
\sigma_2 &= \mu_{\mathrm{M2}} \sin(x_1 - x_3) \sin(x_3) \\
\sigma_3 &= g \cos(x_1) \cos(x_1 - x_3) \\
\sigma_4 &= l_1 x_2^2 \cos(x_1 - x_3) \\
\sigma_5 &= \frac{2\mu_{\mathrm{M2}} \cos(x_1 - x_3) \sin(x_1 - x_3) \big[l_1 \mu_{\mathrm{M2}} \cos(x_1 - x_3) \sin(x_1 - x_3) x_2^2 + l_2 \mu_{\mathrm{M2}} \sin(x_1 - x_3) x_4^2}{l_1 \sigma_{13}} \\
&\quad + g\big(\sin(x_1) - \mu_{\mathrm{M2}} \cos(x_1 - x_3) \sin(x_3) \big) \big] \\
\sigma_6 &= \frac{2\mu_{\mathrm{M2}} \cos(x_1 - x_3) \sin(x_1 - x_3) \big[l_1 \sin(x_1 - x_3) x_2^2 + l_2 \mu_{\mathrm{M2}} \cos(x_1 - x_3) \sin(x_1 - x_3) x_4^2}{l_2 \sigma_{13}} \\
&\quad + g \sin(x_1 - x_3) \cos(x_1) \big] \\
\sigma_7 &= l_2 \mu_{\mathrm{M2}} x_4^2 \sigma_{15} \\
\sigma_8 &= l_1 \mu_{\mathrm{M2}} x_2^2 \sigma_{15} \\
\sigma_9 &= l_2 \mu_{\mathrm{M2}} x_4^2 \sigma_{14} \\
\sigma_{10} &= l_1 \mu_{\mathrm{M2}} x_2^2 \sigma_{14} \\
\sigma_{11} &= l_2 \left(\mu_{\mathrm{M2}} \sigma_{15} + \mu_{\mathrm{M1}} \right) \\
\sigma_{12} &= l_1 \left(\mu_{\mathrm{M2}} \sigma_{15} + \mu_{\mathrm{M1}} \right) \\
\sigma_{13} &= \left(\mu_{\mathrm{M2}} \sigma_{15} + \mu_{\mathrm{M1}} \right)^2 \\
\sigma_{14} &= \cos^2(x_1 - x_3) \\
\sigma_{15} &= \sin^2(x_1 - x_3)
\end{aligned}
$$

5. Covariance prediction:

$$
P_k^- = F_k P_{k-1} F_k^\top + Q_k
$$

$$
Q_k = \begin{bmatrix} q_{\theta_1} & 0 & 0 & 0 \\ 0 & q_{\dot{\theta}_1} & 0 & 0 \\ 0 & 0 & q_{\theta_2} & 0 \\ 0 & 0 & 0 & q_{\dot{\theta}_2} \end{bmatrix}
$$

6. Jacobian of measurement function:

$$
H_k = \left. \frac{\partial h}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k|k-1}}
$$

$$
H_k = \frac{\partial h}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

7. Innovation (residual):

$$
\mathbf{y}_k = \mathbf{z}_k - \hat{\mathbf{z}}_k
$$

$$
\mathbf{y}_k = \begin{bmatrix} \theta_{1,k} \\ \dot{\theta}_{1,k} \\ \theta_{2,k} \\ \dot{\theta}_{2,k} \end{bmatrix} - \begin{bmatrix} \hat{\theta}_{1,k|k-1} \\ \hat{\dot{\theta}}_{1,k|k-1} \\ \hat{\theta}_{2,k|k-1} \\ \hat{\dot{\theta}}_{2,k|k-1} \end{bmatrix}
$$

8. Innovation covariance update:

$$S_k = H_k P_{k|k-1} H_k^T + R_k$$

$$R_k = \begin{bmatrix} r_{\theta_1} & 0 & 0 & 0 \\ 0 & r_{\dot{\theta}_1} & 0 & 0 \\ 0 & 0 & r_{\theta_2} & 0 \\ 0 & 0 & 0 & r_{\dot{\theta}_2} \end{bmatrix}$$

9. Kalman gain:

$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$

10. State update:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \mathbf{y}_k$$

$$\hat{\mathbf{x}}_{k|k} = \begin{bmatrix} \hat{\theta}_{1,k|k-1} \\ \hat{\dot{\theta}}_{1,k|k-1} \\ \hat{\theta}_{2,k|k-1} \\ \hat{\dot{\theta}}_{2,k|k-1} \end{bmatrix} + K_k \begin{bmatrix} \theta_{1,k} - \hat{\theta}_{1,k|k-1} \\ \dot{\theta}_{1,k} - \hat{\dot{\theta}}_{1,k|k-1} \\ \theta_{2,k} - \hat{\theta}_{2,k|k-1} \\ \dot{\theta}_{2,k} - \hat{\dot{\theta}}_{2,k|k-1} \end{bmatrix}$$

11. Covariance update:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

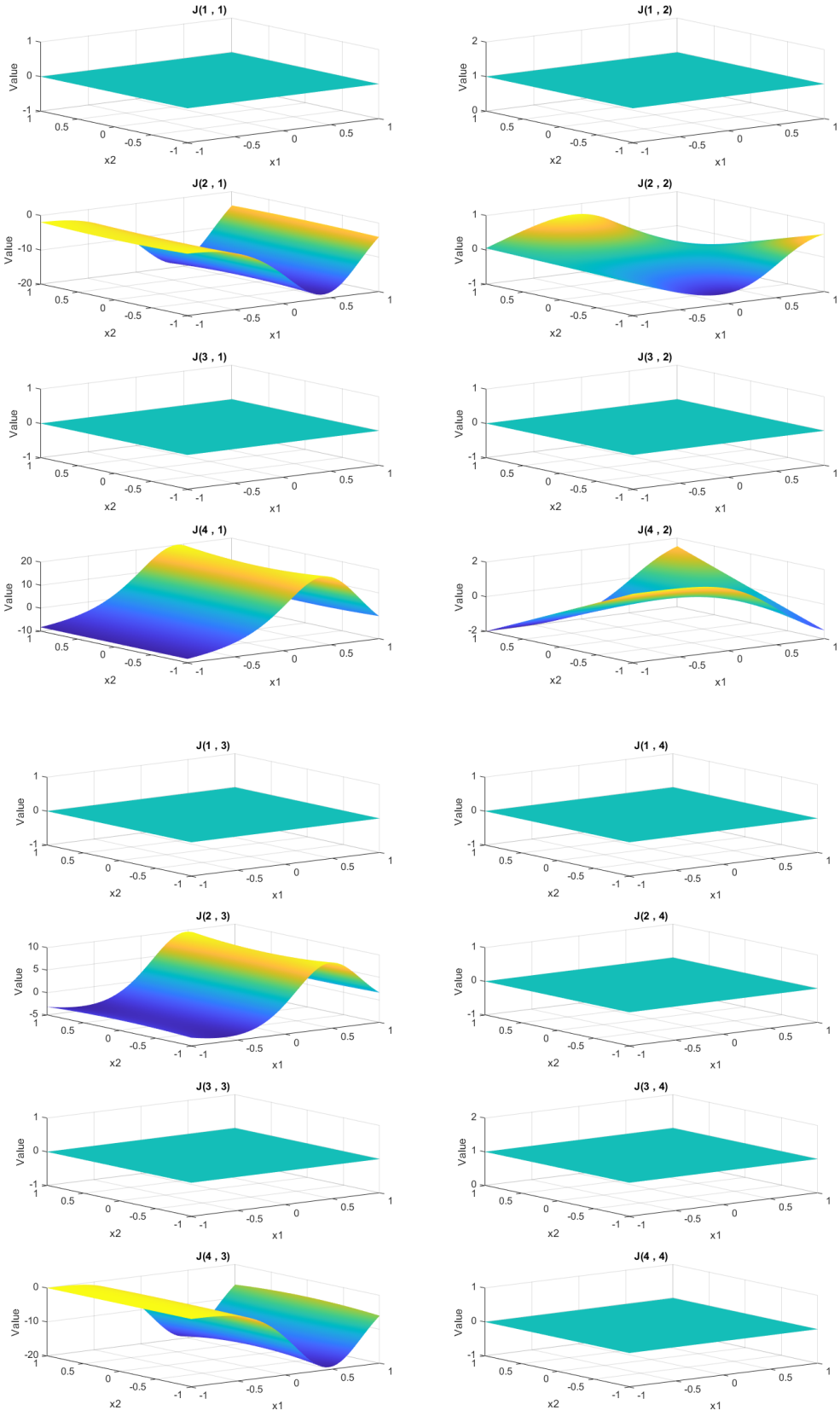


Figure 7: Initial state

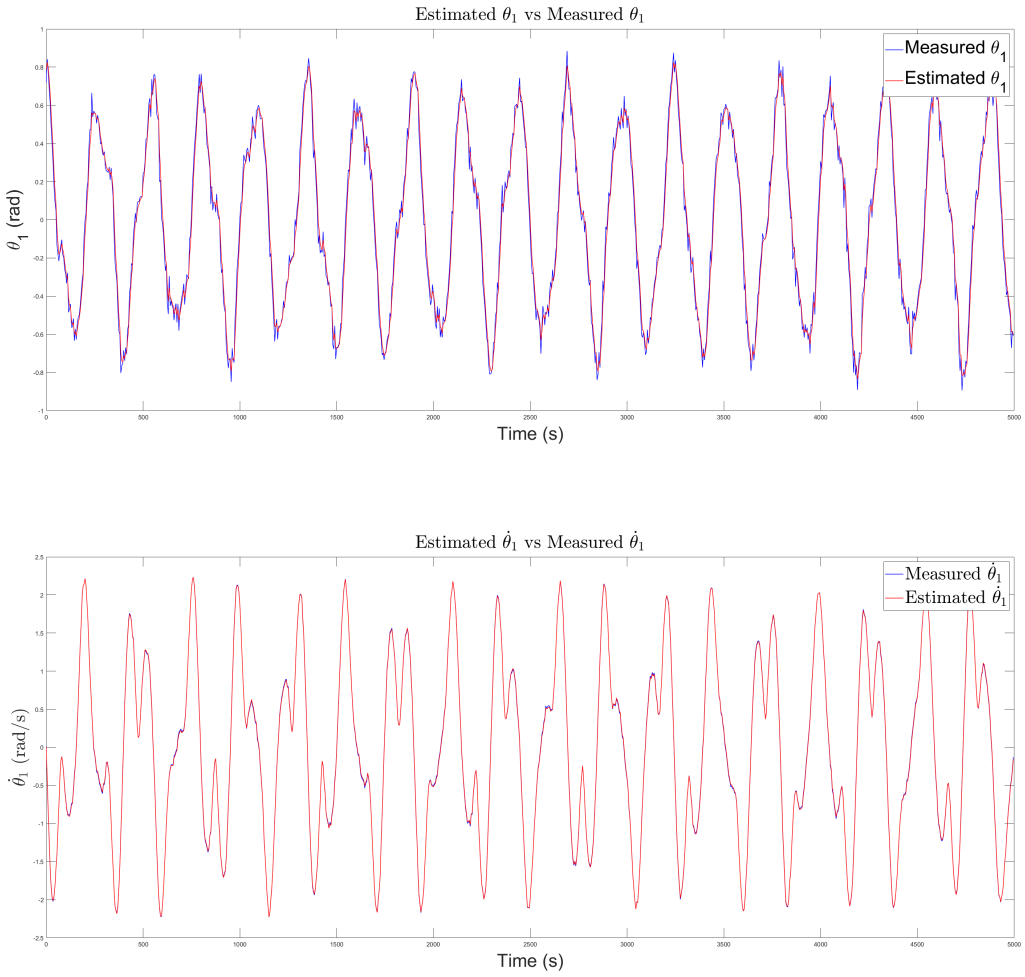


Figure 8: 1st pendulum EKF estimate

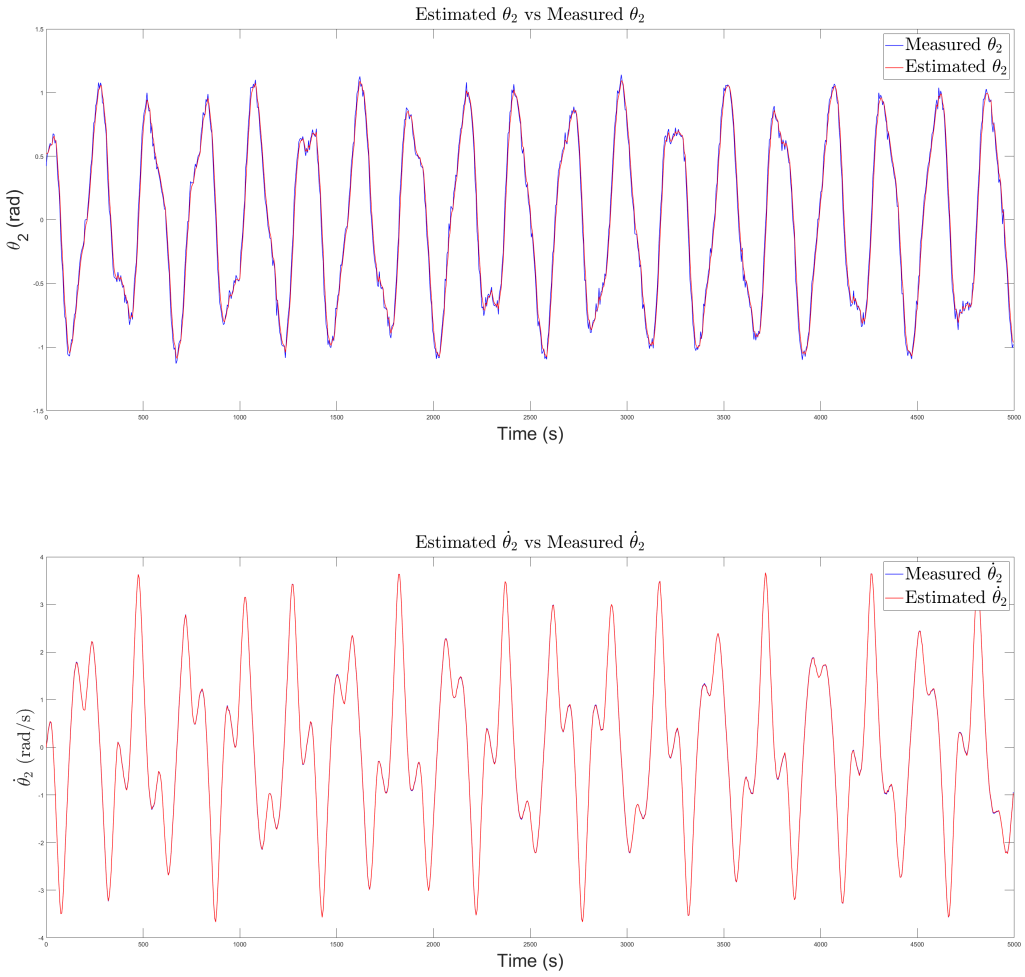


Figure 9: 2nd pendulum EKF estimate

10 Estimation Technique (UKF)

By propagating a set of sigma points through the nonlinear-system dynamic function, UKF provides a more accurate estimate of the state [3, 2].

1. Initial:

$$\mathbf{x}_k = \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix}$$

$$\mathbf{x}_0 = \begin{bmatrix} \pi/4 \\ 0 \\ \pi/6 \\ 0 \end{bmatrix}$$

2. Initial covariance $\mathbf{P}_0$:

$$\mathbf{P}_0 = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$$

3. Process noise covariance ($\mathbf{Q}$):

$$\mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

4. Measurement noise covariance ($\mathbf{R}$):

$$\mathbf{R} = \begin{bmatrix} 0.05 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0.05 \end{bmatrix}$$

5. UKF parameters:

$$\alpha = 0.001,\ \beta = 2,\ \kappa = 0$$

6. State dimension:

$$n = 4$$

7. Spread parameter:

$$\gamma = \sqrt{n + \lambda}$$

8. Mean weights:

$$W_m^{(0)} = \frac{\lambda}{n + \lambda}$$

$$W_m^{(i)} = \frac{1}{2(n + \lambda)}$$

$$i = 1, \ldots, 2n$$

9. Covariance weights:

$$W_c^{(0)} = W_m^{(0)} + (1 - \alpha^2 + \beta)$$

$$W_c^{(i)} = W_m^{(i)}$$

10. Sigma point generation:

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{x}_k & \mathbf{x}_k + \gamma\sqrt{\mathbf{P}_k} & \mathbf{x}_k - \gamma\sqrt{\mathbf{P}_k} \end{bmatrix}$$

11. Prediction step:

$$\mathbf{X}_k^{(i)} = \left[\theta_1^{(i)}, \dot{\theta}_1^{(i)}, \theta_2^{(i)}, \dot{\theta}_2^{(i)}\right]$$

$$\ddot{\theta}_1^{(i)}, \ddot{\theta}_2^{(i)}.$$

$$\dot{\theta}_1^{(i)} \leftarrow \dot{\theta}_1^{(i)} + \ddot{\theta}_1^{(i)} \cdot \Delta t$$
$$\dot{\theta}_2^{(i)} \leftarrow \dot{\theta}_2^{(i)} + \ddot{\theta}_2^{(i)} \cdot \Delta t$$

$$\theta_1^{(i)} \leftarrow \theta_1^{(i)} + \dot{\theta}_1^{(i)} \cdot \Delta t$$
$$\theta_2^{(i)} \leftarrow \theta_2^{(i)} + \dot{\theta}_2^{(i)} \cdot \Delta t$$

$$\mathbf{X}_k^{(i,\text{pred})} = \left[\theta_1^{(i)}, \dot{\theta}_1^{(i)}, \theta_2^{(i)}, \dot{\theta}_2^{(i)}\right]$$

12. Predict state mean:

$$\hat{\mathbf{x}}_{k|k-1} = \sum_{i=0}^{2n} W_m^{(i)} \mathbf{X}_k^{(i,\text{pred})}$$

13. Predict state covariance:

$$\mathbf{P}_{k|k-1} = \mathbf{Q} + \sum_{i=0}^{2n} W_c^{(i)} \left(\mathbf{X}_k^{(i,\text{pred})} - \hat{\mathbf{x}}_{k|k-1}\right) \left(\mathbf{X}_k^{(i,\text{pred})} - \hat{\mathbf{x}}_{k|k-1}\right)^T$$

14. Predict measurement (assume direct measurement of state):

$$\mathbf{Z}_k^{(i)} = h(\mathbf{X}_k^{(i,\text{pred})}) = \mathbf{X}_k^{(i,\text{pred})}$$

15. Measurement mean:

$$\hat{\mathbf{z}}_k = \sum_{i=0}^{2n} W_m^{(i)} \mathbf{Z}_k^{(i)}$$

16. Measurement covariance:

$$\mathbf{S}_k = \mathbf{R} + \sum_{i=0}^{2n} W_c^{(i)} \left(\mathbf{Z}_k^{(i)} - \hat{\mathbf{z}}_k\right) \left(\mathbf{Z}_k^{(i)} - \hat{\mathbf{z}}_k\right)^T$$

17. Cross-covariance:

$$\mathbf{P}_{xz} = \sum_{i=0}^{2n} W_c^{(i)} \left(\mathbf{X}_k^{(i,\text{pred})} - \hat{\mathbf{x}}_{k|k-1}\right) \left(\mathbf{Z}_k^{(i)} - \hat{\mathbf{z}}_k\right)^T$$

18. Kalman gain:

$$\mathbf{K}_k = \mathbf{P}_{xz} \mathbf{S}_k^{-1}$$

19. Update state:

$$\mathbf{x}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k)$$

20. Update covariance:

$$\mathbf{P}_k = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^T$$

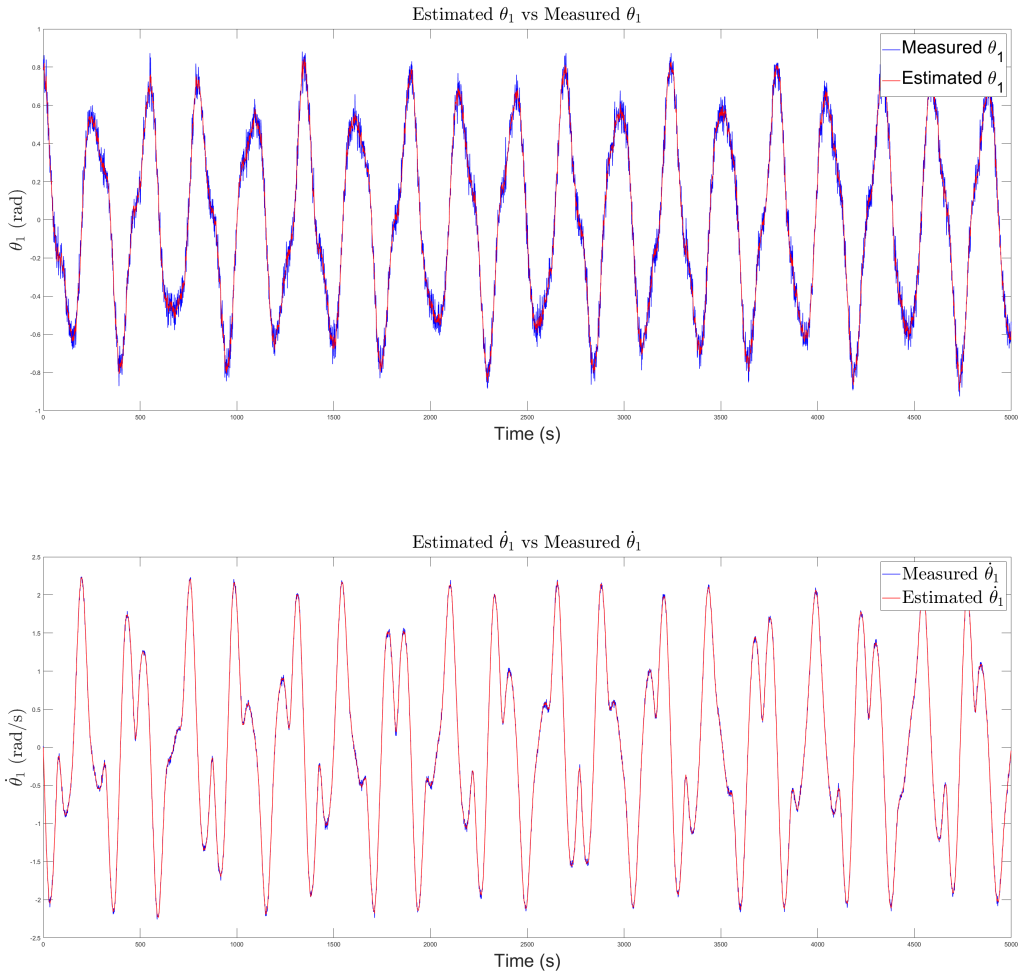


Figure 10: 1st pendulum UKF estimate

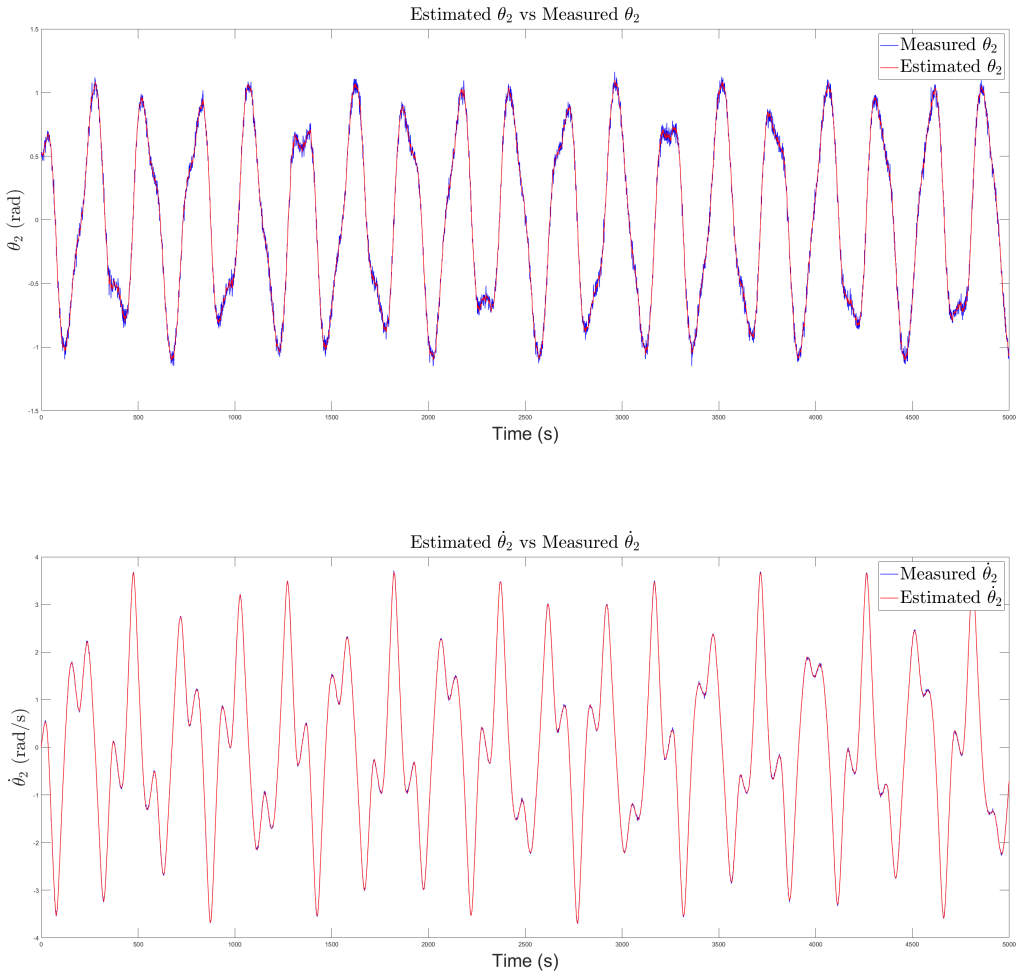


Figure 11: 2nd pendulum UKF estimate

11 Expected Results

It is expected that UKF will outperform EKF in terms of computation cost since it does not require linearization of the non-linear system as a function of time. UKF will also likely provide a more accurate estimation of the state because it propagates a set of carefully chosen sigma points through the dynamic function, which then captures the mean and covariance of the state distribution. Since both techniques assume fixed covariance matrices (process noise Q and measurement noise R) in real-world scenario these matrices are subject to change based on various environmental factors. Therefore, AUKF is expected to outperform both methods because it allows the filter to simultaneously estimate the system when the noise characteristics are unknown or time-varying.

12 Result (EKF)

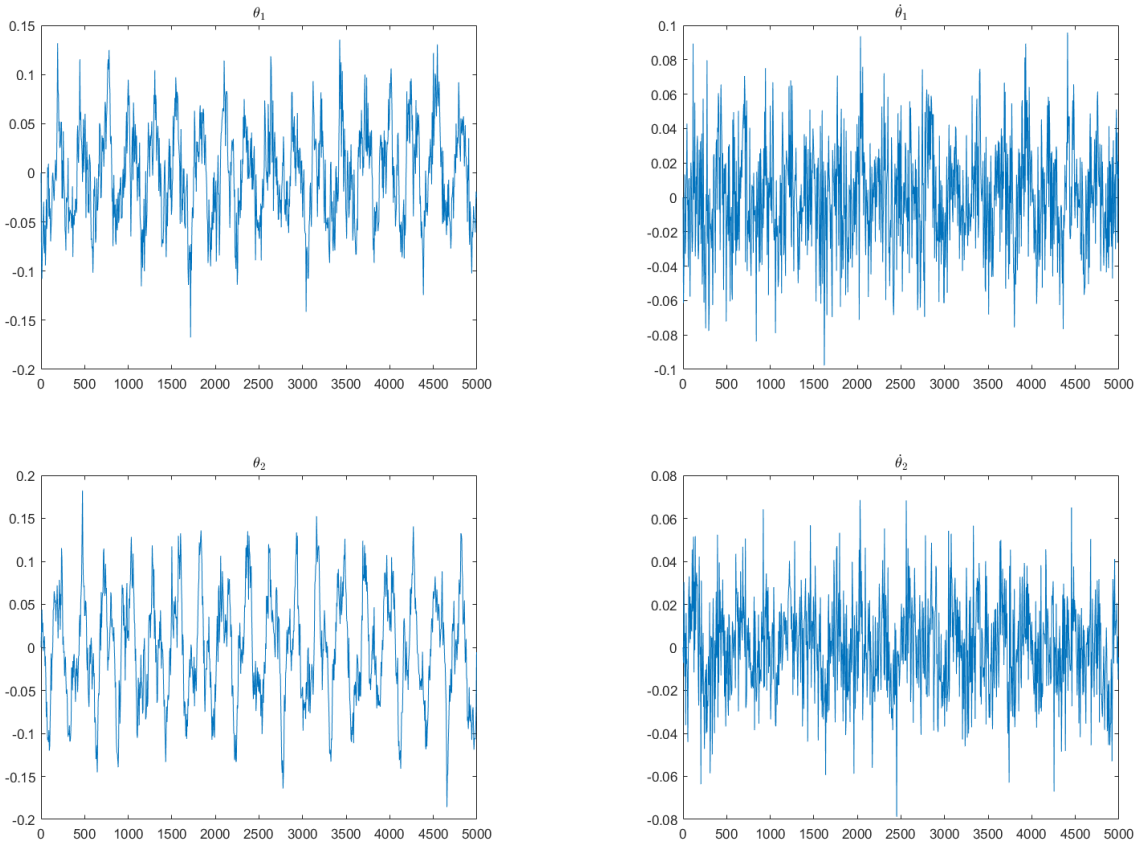


Figure 12: Estimation error (EKF)

Time	Theta1	Theta1_Dot	Theta2	Theta2_Dot
0.01	0.098876	0.040668	0.14412	0.033352
0.02	0.13325	0.047302	0.18909	0.039577
0.03	0.14887	0.054808	0.20918	0.04558
0.04	0.15309	0.062224	0.22517	0.05649
0.05	0.15901	0.070097	0.22909	0.058135
0.06	0.16527	0.071107	0.23637	0.072472
0.07	0.17389	0.080238	0.24143	0.077244
0.08	0.1737	0.096526	0.24761	0.090596
0.09	0.16914	0.094585	0.25245	0.095725
0.1	0.17748	0.10397	0.24961	0.099683

Table 1: RMSE with varying noise

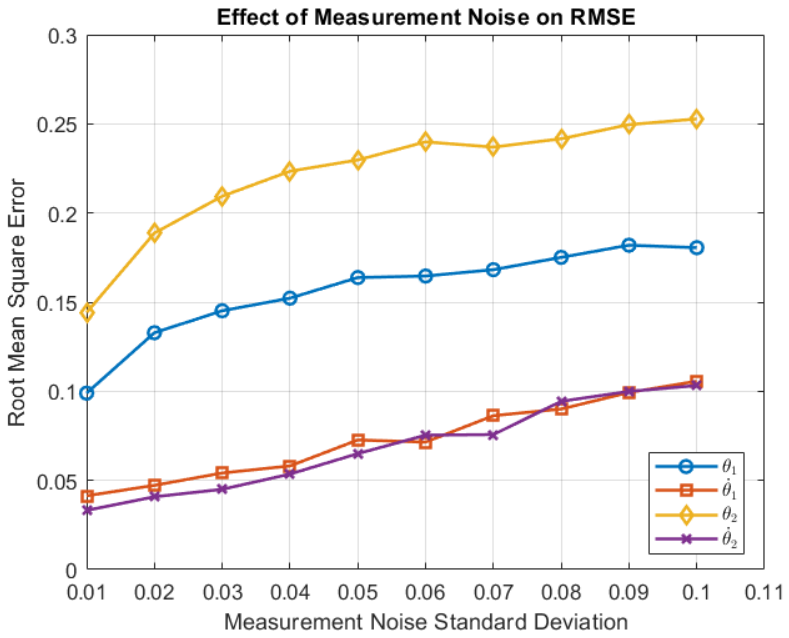


Figure 13: Robustness to varying noise (EKF)

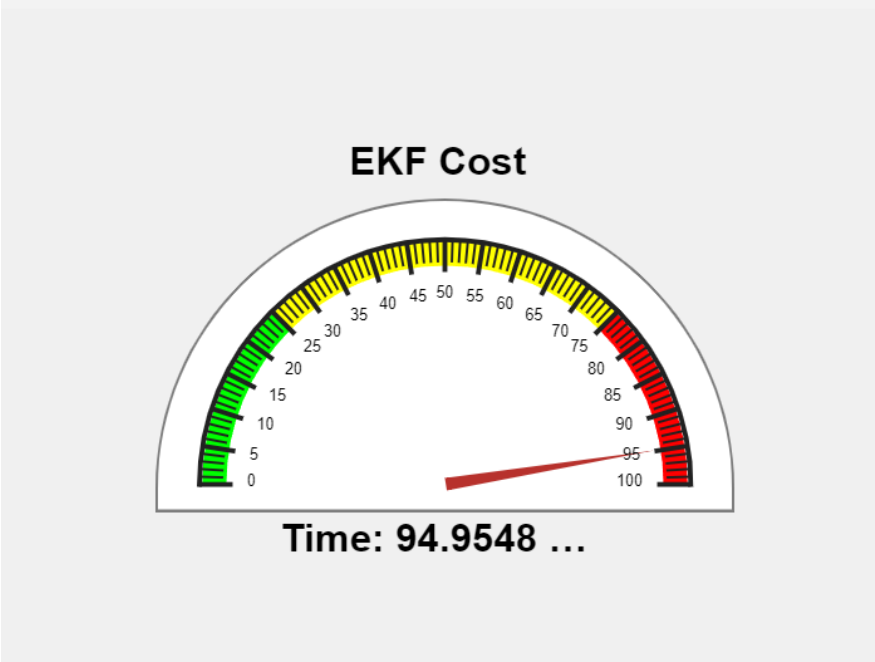


Figure 14: Computation cost (EKF)

13 Result (UKF)

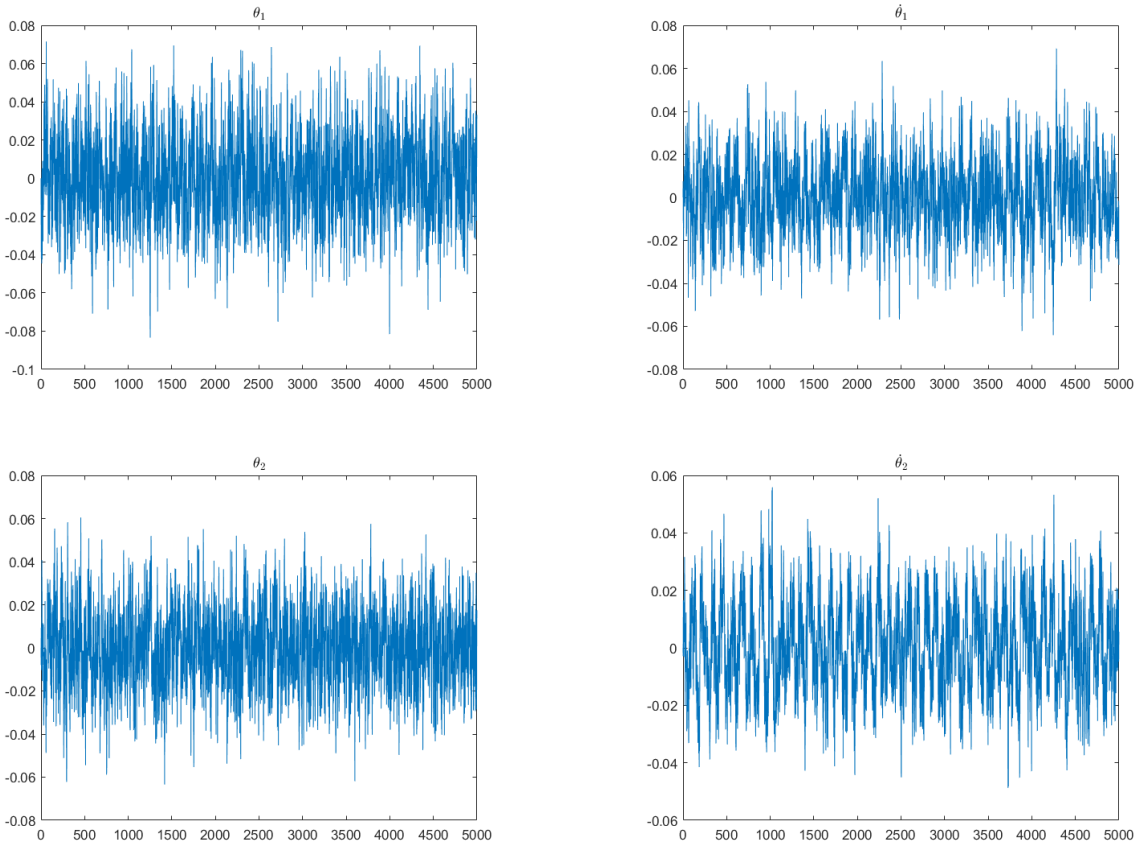


Figure 15: Estimation error (UKF)

Time	**Theta1**	**Theta1_Dot**	**Theta2**	**Theta2_Dot**
0.01	0.0060467	0.0066479	0.0060733	0.0071955
0.02	0.017187	0.01214	0.01875	0.01409
0.03	0.021051	0.016779	0.022871	0.019335
0.04	0.025	0.020662	0.026907	0.024045
0.05	0.028505	0.02539	0.030215	0.028663
0.06	0.030795	0.028713	0.033283	0.032979
0.07	0.034421	0.03345	0.036672	0.037916
0.08	0.036756	0.037634	0.039683	0.041641
0.09	0.03932	0.042186	0.041632	0.04612
0.1	0.042718	0.044349	0.045184	0.049427

Table 2: RMSE with varying noise

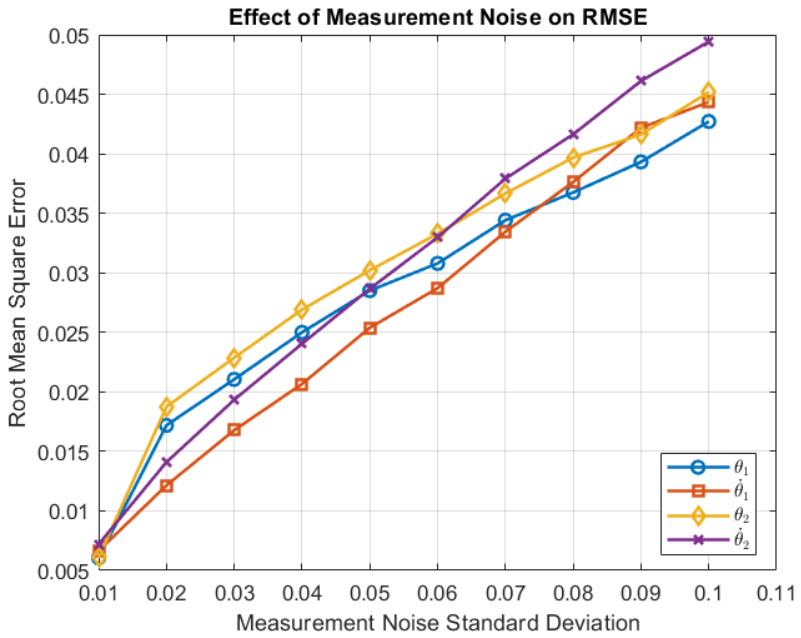


Figure 16: Robustness to varying noise (UKF)

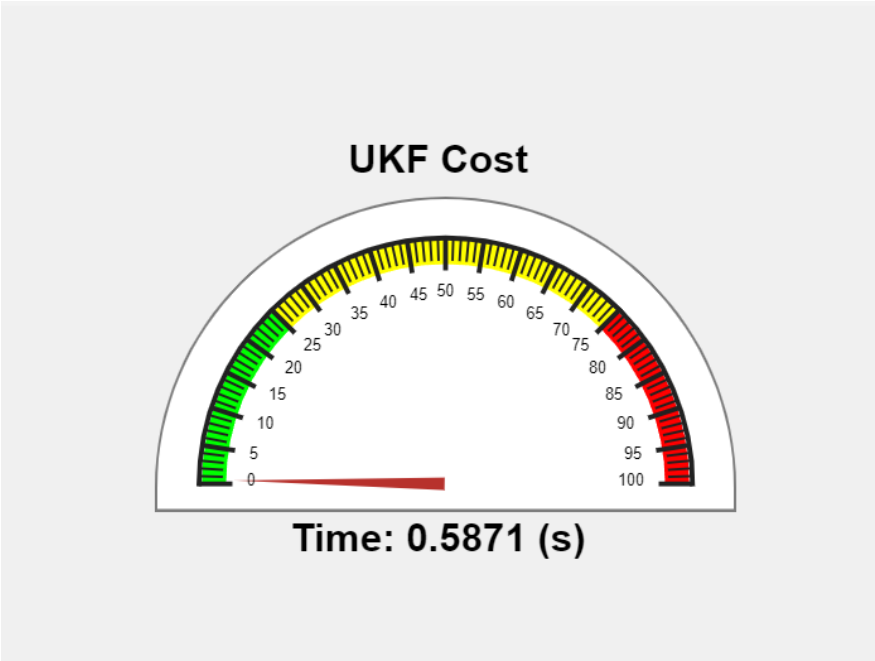


Figure 17: Computation cost (UKF)

14 Result (AUKF)

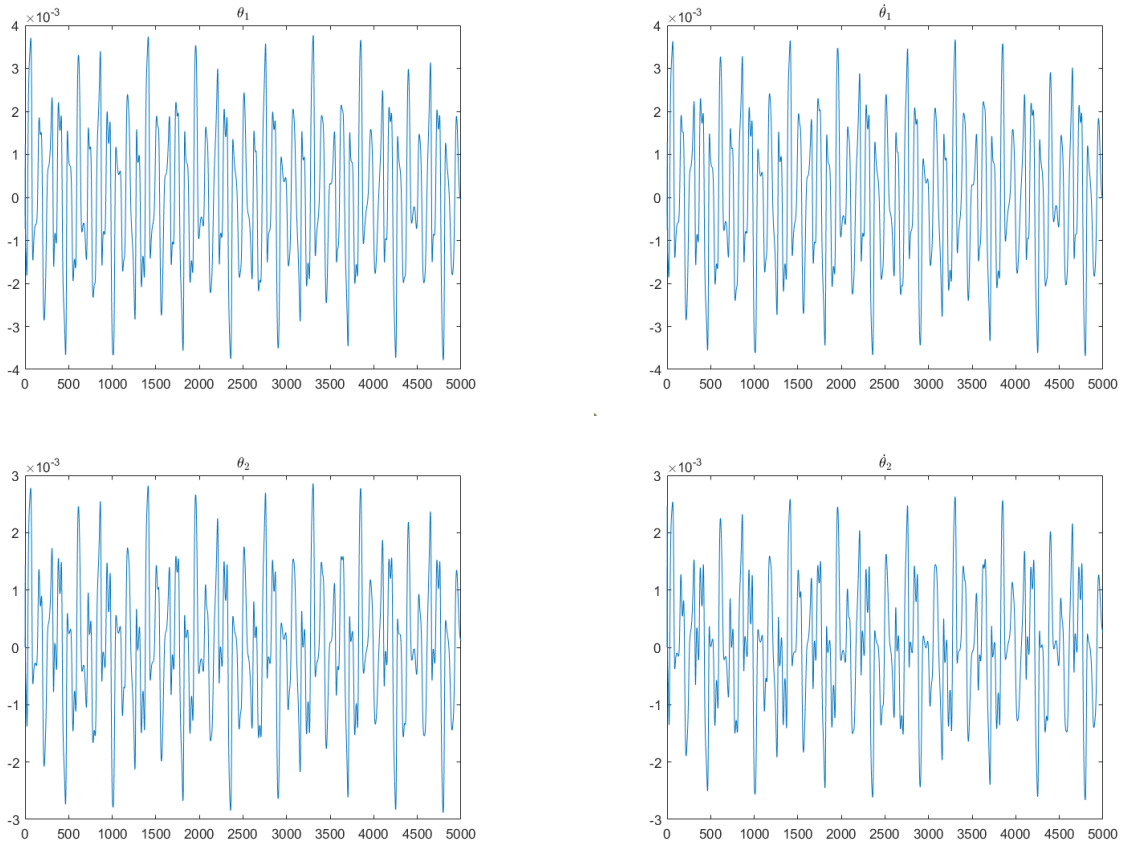


Figure 18: Estimation error (AUKF)

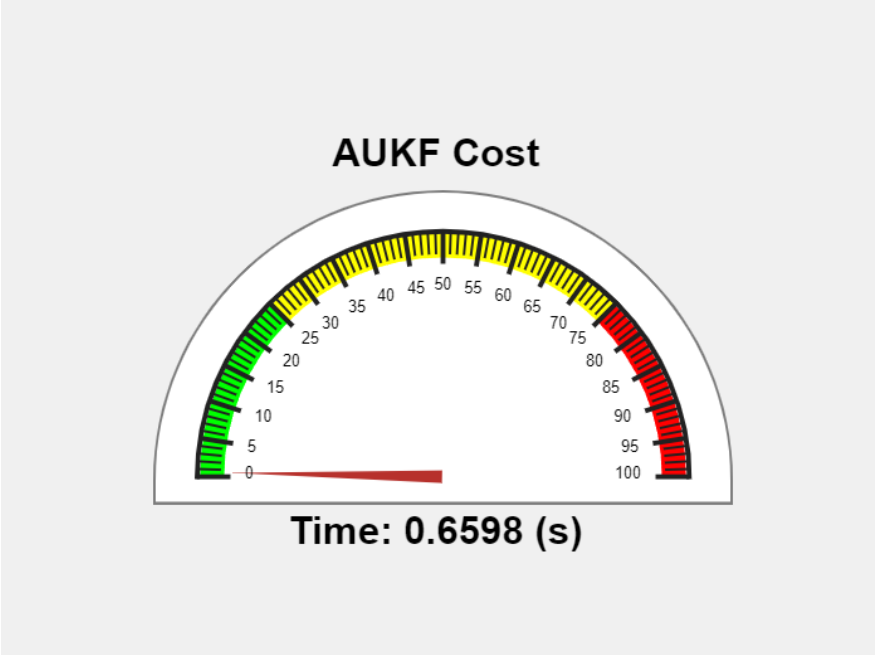
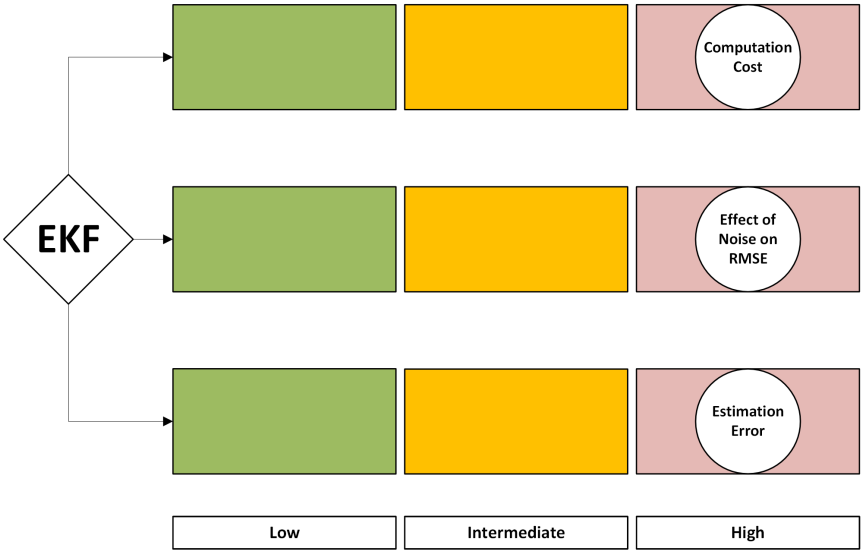
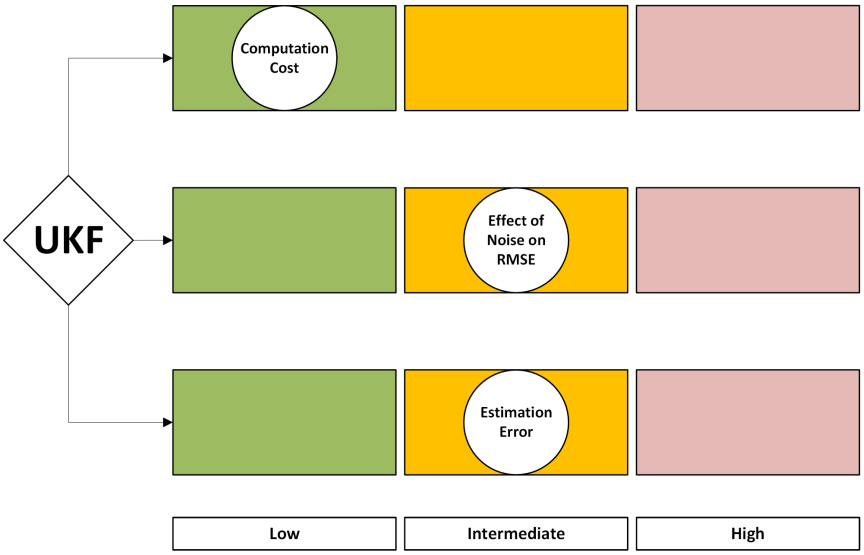


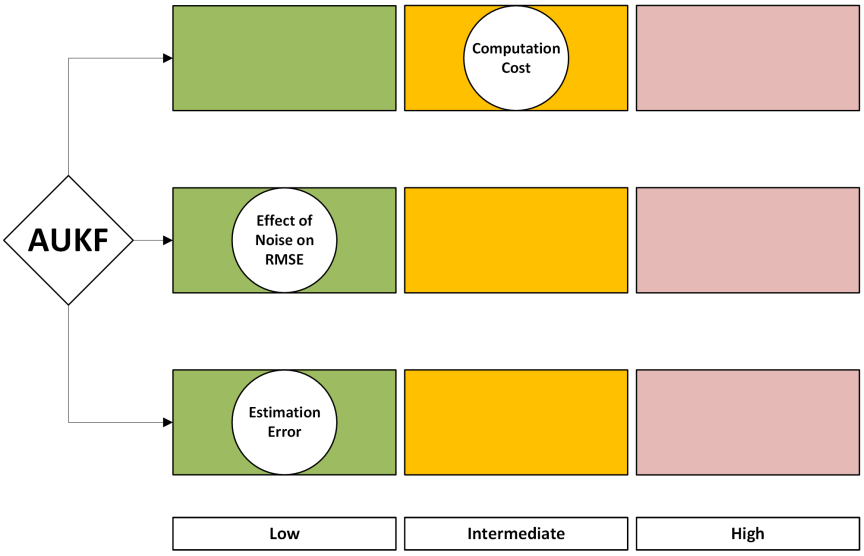
Figure 19: Computation cost (AUKF)

15 Conclusion

Based on values of root mean square error, UKF has a lower RMSE value for all states than EKF. As can be seen from Table 1 and Table 2, as well as Figure 14 and Figure 17, as noise levels grow larger the RMSE value for all states grows larger. However, RMSE grows a lot faster (in the 10th decimal place) for the EKF case. UKF also clearly shows a lower computation cost (0.5871 seconds) compared to EKF (94.9548 seconds) as can be seen in Figure 16 and Figure 13 (clearly due to linearization at each time interval). On the other hand, AUKF has the lowest estimation error (1000th decimal place) compared to both EKF and UKF based on figures 18, 12, and 15, respectively. However, it has a slightly higher computation cost (0.6598 seconds) compared to UKF. Therefore, AUKF ranks the highest for robustness to varying noise with the lowest estimation errors, and UKF ranks the highest with having the lowest computation cost compared to EKF and AUKF.




UKF
Computation Cost
Effect of Noise on RMSE
Estimation Error
Low
Intermediate
High


AUKF
Computation Cost
Effect of Noise on RMSE
Estimation Error
Low
Intermediate
High

References

[1] Y. Bazargan-Lari, M. Eghtesad, A.R. Khoogar, and A. Mohammad-Zadeh. Dynamics and regulation of locomotion of a human swing leg as a double-pendulum considering self-impact joint constraint. *Journal of Biomedical Physics and Engineering*, 2014.

[2] John L. Crassidis and John L. Junkins. *Optimal Estimation of Dynamic Systems*. Chapman and Hall/CRC, New York, 2nd edition edition, 2011. ISBN 9780429105609. doi: 10.1201/b11154.

[3] Fred Daum. Nonlinear filters: beyond the kalman filter, ieee a&e. *Aerospace and Electronic Systems Magazine, IEEE*, 20:57–69, September 2005. doi: 10.1109/MAES.2005.1499276.

[4] E. Kaiser, J.N. Kutz, and S.L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474:20180335, 2018.

[5] A. Kiyoumarsi, M. Ataei, B. Mirzaeian-Dehkordi, and R. Ashrafi. The mathematical modeling of a double-pendulum system as a physical model of flexible arm robot. In *2007 IEEE International Conference on Control and Automation*. IEEE, 2007.

[6] T. Orhanli and A. Yilmaz. Analysis of gait dynamics with the double pendulum model. In *2019 27th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2019.

[7] S.H. Rudy, J.N. Kutz, and S.L. Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics*, 396:483–506, 2019.

Mehdi Muzaffari - MAE 674 - Project 1

dynamic of double pendulum

```
clc
clear;
close all;

dt = 0.01;
num_steps = 5000;
time = (0:num_steps-1);

theta1 = pi/4;
theta1_dot = 0;
theta2 = pi/6;
theta2_dot = 0;

theta1_history = zeros(1, num_steps);
theta1_dot_history = zeros(1, theta2_dot);
theta2_history = zeros(1, num_steps);
theta2_dot_history = zeros(1, num_steps);

theta1_history(1) = theta1;
theta1_dot_history(1) = theta1_dot;
theta2_history(1) = theta2;
theta2_dot_history(1) = theta2_dot;
```

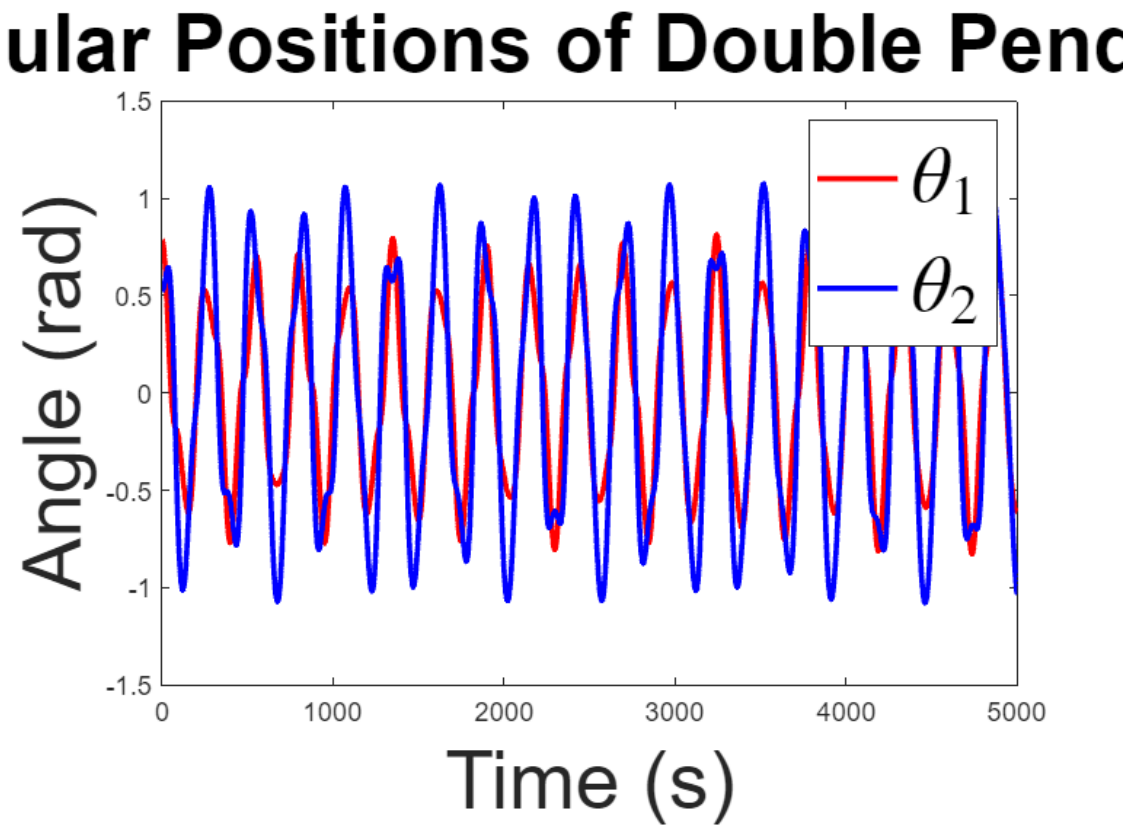
euler integration

```
for k = 2:num_steps

[theta1_ddot, theta2_ddot] = dynamic(theta1, theta1_dot, theta2, theta2_dot);

theta1_dot = theta1_dot + theta1_ddot * dt;
theta1 = theta1 + theta1_dot * dt;

theta2_dot = theta2_dot + theta2_ddot * dt;
theta2 = theta2 + theta2_dot * dt;

theta1_history(k) = theta1;
theta1_dot_history(k) = theta1_dot;
theta2_history(k) = theta2;
theta2_dot_history(k) = theta2_dot;

end
```

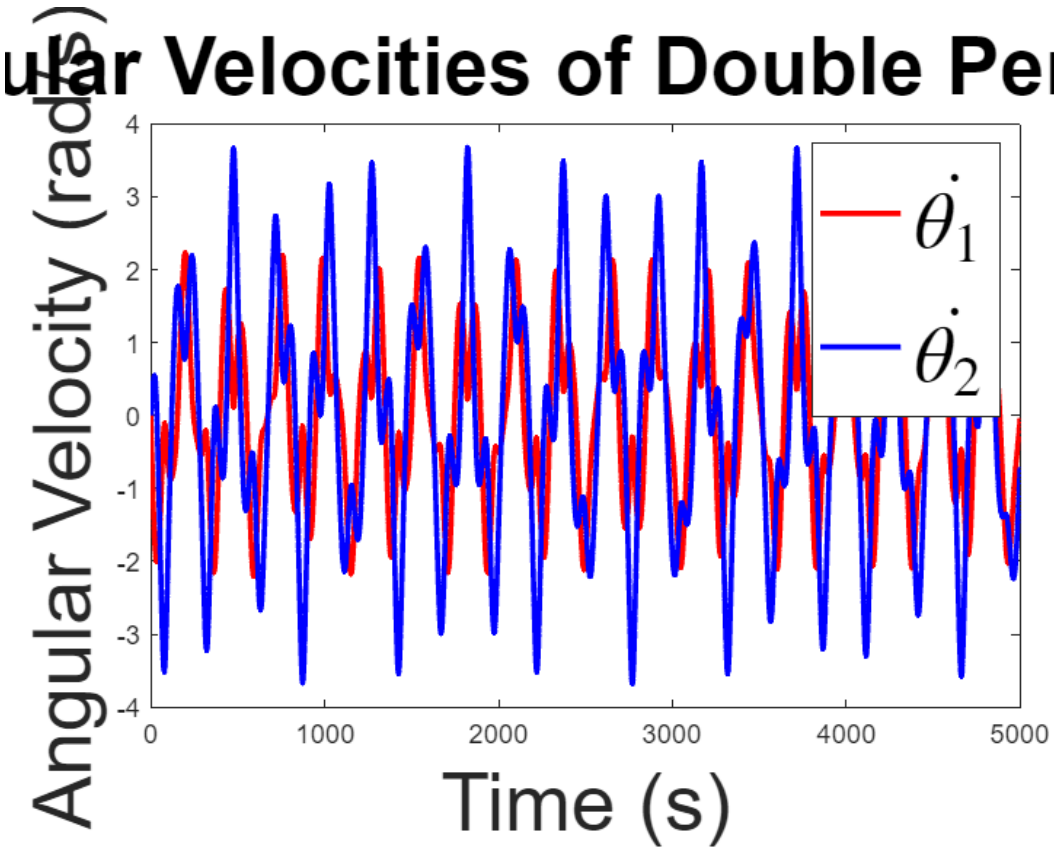
plot

```
figure;
```

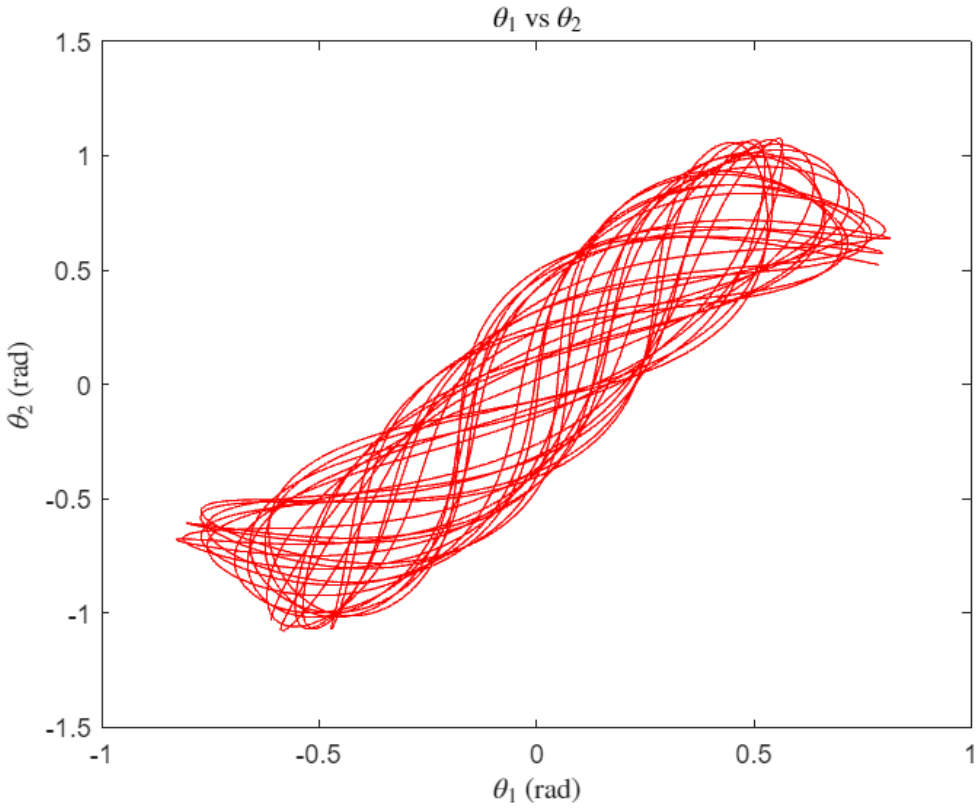
```
plot(time, theta1_history, 'DisplayName', '$\theta_1$', 'LineWidth', 2.0, 'Color',
'r');
hold on
plot(time, theta2_history, 'DisplayName', '$\theta_2$', 'LineWidth', 2.0, 'Color',
'b');
xlabel('Time (s)', 'FontSize', 30);
ylabel('Angle (rad)', 'FontSize', 30);
legend('Interpreter', 'latex', 'FontSize', 30);
title('Angular Positions of Double Pendulum', 'FontSize', 30);
hold off
```



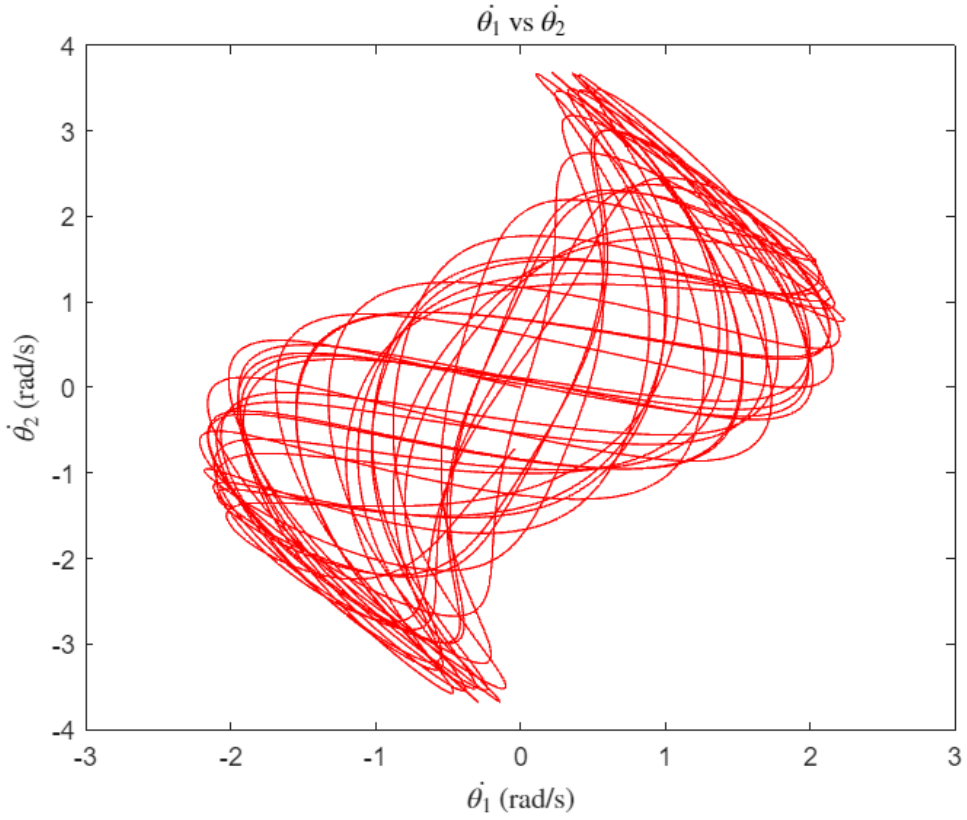

```
figure;
plot(time, theta1_dot_history, 'DisplayName', '$\dot{\theta_1}$', 'LineWidth', 2.0,
'Color', 'r');
hold on
plot(time, theta2_dot_history, 'DisplayName', '$\dot{\theta_2}$', 'LineWidth', 2.0,
'Color', 'b');
xlabel('Time (s)', 'FontSize', 30);
ylabel('Angular Velocity (rad/s)', 'FontSize', 30);
legend('Interpreter', 'latex', 'FontSize', 30);
title('Angular Velocities of Double Pendulum', 'FontSize', 30);
hold off
```



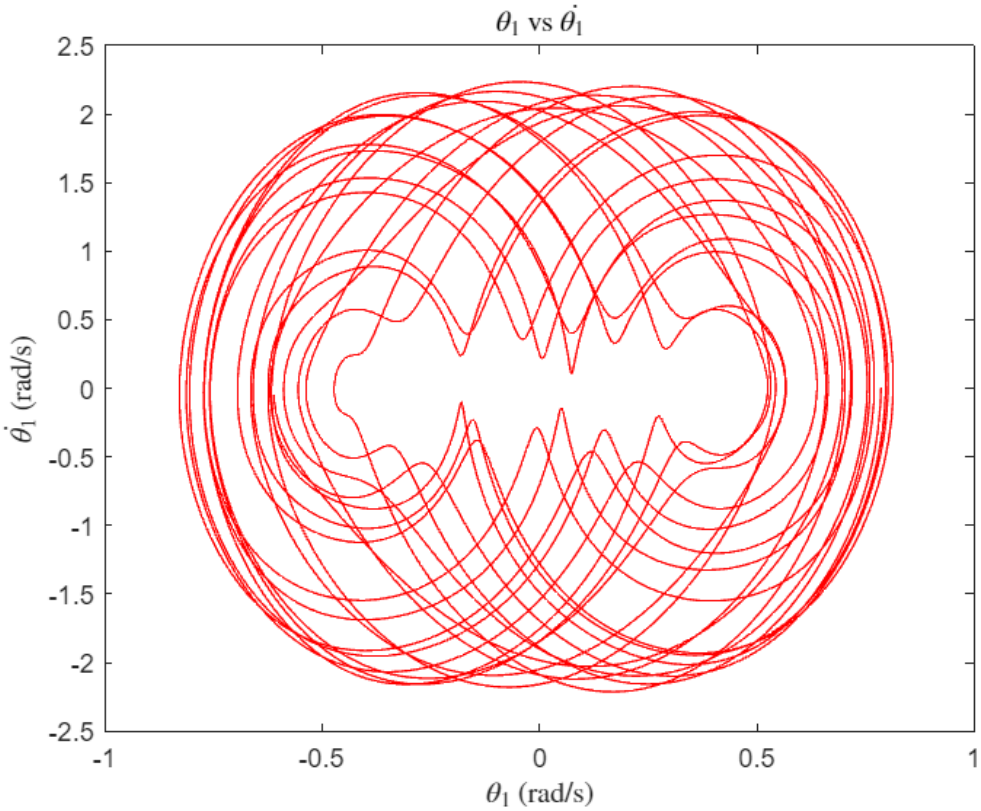

```
figure;
plot(theta1_history, theta2_history, 'LineWidth', 0.05, 'Color', 'r');
xlabel('$\theta_1$ (rad)', 'Interpreter', 'latex');
ylabel('$\theta_2$ (rad)', 'Interpreter', 'latex');
title('$\theta_1$ vs $\theta_2$', 'Interpreter', 'latex');
```



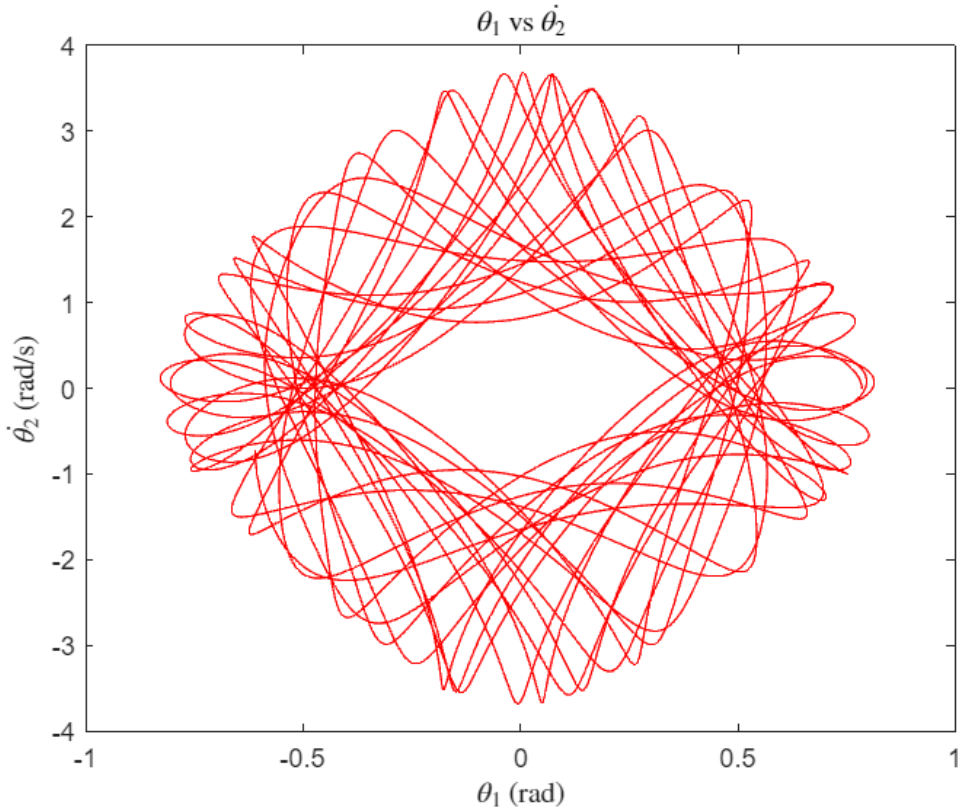

```
figure;
plot(theta1_dot_history, theta2_dot_history, 'LineWidth', 0.05, 'Color', 'r');
xlabel('$\dot{\theta_1}$ (rad/s)', 'Interpreter', 'latex');
ylabel('$\dot{\theta_2}$ (rad/s)', 'Interpreter', 'latex');
title('$\dot{\theta_1}$ vs $\dot{\theta_2}$', 'Interpreter', 'latex');
```



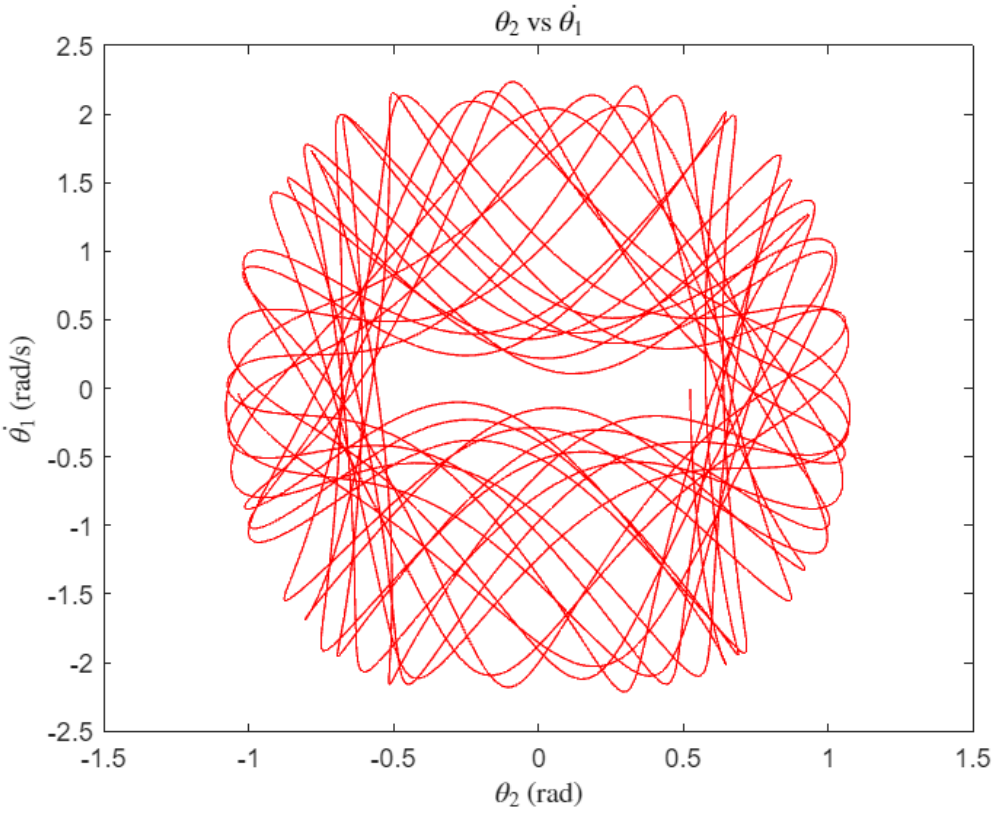

```
figure;
plot(theta1_history, theta1_dot_history, 'LineWidth', 0.05, 'Color', 'r');
xlabel('$\theta_1$ (rad/s)', 'Interpreter', 'latex');
ylabel('$\dot{\theta_1}$ (rad/s)', 'Interpreter', 'latex');
title('$\theta_1$ vs $\dot{\theta_1}$', 'Interpreter', 'latex');
```



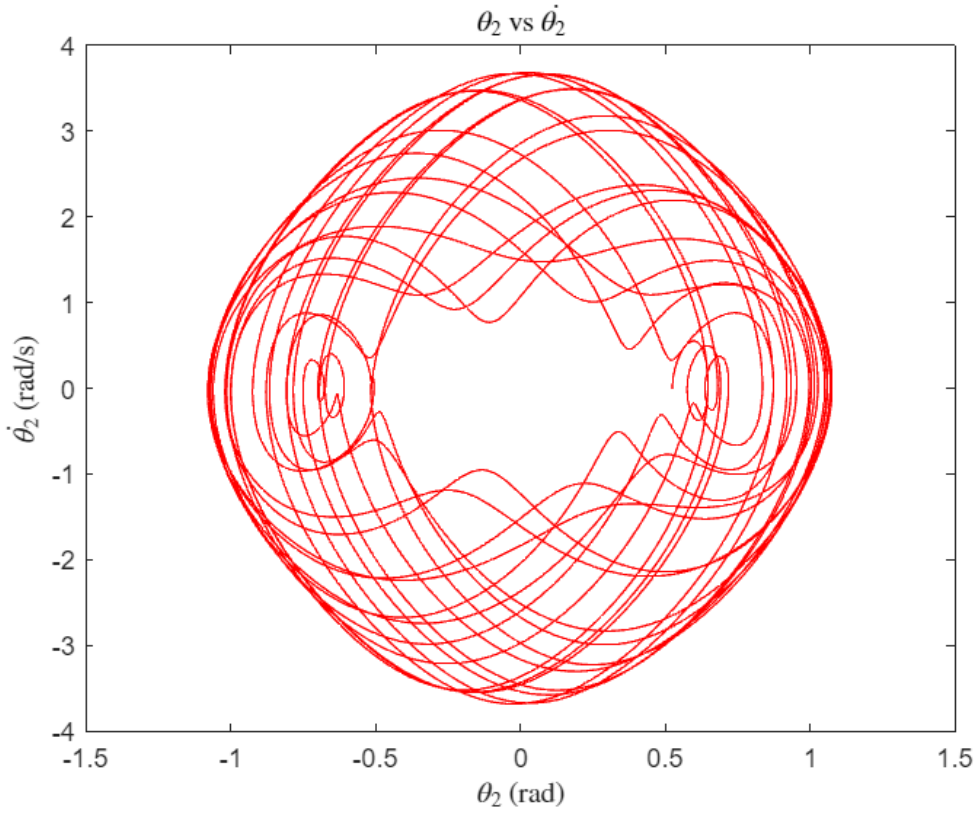

```
figure;
plot(theta1_history, theta2_dot_history, 'LineWidth', 0.05, 'Color', 'r');
xlabel('$\theta_1$ (rad)', 'Interpreter', 'latex');
ylabel('$\dot{\theta_2}$ (rad/s)', 'Interpreter', 'latex');
title('$\theta_1$ vs $\dot{\theta_2}$', 'Interpreter', 'latex');
```




```
figure;
plot(theta2_history, theta1_dot_history, 'LineWidth', 0.05, 'Color', 'r');
xlabel('$\theta_2$ (rad)', 'Interpreter', 'latex');
ylabel('$\dot{\theta_1}$ (rad/s)', 'Interpreter', 'latex');
title('$\theta_2$ vs $\dot{\theta_1}$', 'Interpreter', 'latex');
```




```
figure;
plot(theta2_history, theta2_dot_history, 'LineWidth', 0.05, 'Color', 'r');
xlabel('$\theta_2$ (rad)', 'Interpreter', 'latex');
ylabel('$\dot{\theta_2}$ (rad/s)', 'Interpreter', 'latex');
title('$\theta_2$ vs $\dot{\theta_2}$', 'Interpreter', 'latex');
```


θ_2 vs $\dot{\theta}_2$
4
3
2
1
0
-1
-2
-3
-4
$\dot{\theta}_2$ (rad/s)
-1.5
-1
-0.5
0
0.5
1
1.5
θ_2 (rad)

Mehdi Muzaffari - MAE 674 - Project 1

estimate states through ekf

```
clc;
clear;
close all;
```

load simulate data

```
data = readtable('simulation');
time = data.Time;
theta1_true = data.Theta1;
theta1_dot_true = data.Theta1_Dot;
theta2_true = data.Theta2;
theta2_dot_true = data.Theta2_Dot;

time = downsample(time, 5);
theta1_true = downsample(theta1_true, 5);
theta1_dot_true = downsample(theta1_dot_true, 5);
theta2_true = downsample(theta2_true, 5);
theta2_dot_true = downsample(theta2_dot_true, 5);
```

add noise to data

```
noise_std_theta1 = 0.05;
noise_std_theta1_dot = 0.03;
noise_std_theta2 = 0.04;
noise_std_theta2_dot = 0.02;

theta1_measured = theta1_true + noise_std_theta1 * randn(size(theta1_true));
theta1_dot_measured = theta1_dot_true + noise_std_theta1_dot *
randn(size(theta1_dot_true));
theta2_measured = theta2_true + noise_std_theta2 * randn(size(theta2_true));
theta2_dot_measured = theta2_dot_true + noise_std_theta2_dot *
randn(size(theta2_dot_true));
```

simulation parameter

```
dt = 0.01;
num_steps = length(time);

theta1 = pi/4;
theta1_dot = 0;
theta2 = pi/6;
theta2_dot = 0;

x_hat = [pi/4; 0; pi/6; 0];
```

```
P = diag([0.01, 0.1, 0.01, 0.01]);
Q = diag([0.01, 0.01, 0.01, 0.01]);
R = diag([0.05, 0.05, 0.05, 0.05]);
```

state transition matrix

```
syms x1 x2 x3 x4 g mu_M1 mu_M2 l1 l2 real

beta = x3 - x1;

dx1 = x2;
dx2 = (g * (mu_M2 * cos(beta) * sin(x3) - sin(x1)) + ...
       mu_M2 * l1 * sin(beta) * cos(beta) * x2^2 + ...
       mu_M2 * l2 * sin(beta) * x4^2) / ...
       (l1 * (mu_M1 + mu_M2 * sin(beta)^2));

dx3 = x4;
dx4 = -(l1 * sin(beta) * x2^2 + ...
        l2 * mu_M2 * sin(beta) * cos(beta) * x4^2 + ...
        g * cos(x1) * sin(beta)) / ...
        (l2 * (mu_M1 + mu_M2 * sin(beta)^2));

state_vector = [x1; x2; x3; x4];
equations = [dx1; dx2; dx3; dx4];

Jacobian = jacobian(equations, state_vector);
disp(Jacobian);
```

$$\begin{pmatrix} 0 & 1 \\ \sigma_5 - \dfrac{g\ (\cos(x_1) + \sigma_2) + \sigma_1 + \sigma_{10} - \sigma_8}{\sigma_{12}} & -\dfrac{2\,\mu_{M2}\,x_2\cos(x_1 - x_3)\sin(x_1 - x_3)}{\mu_{M2}\,\sigma_{15} + \mu_{M1}} \quad \dfrac{g\ (\mu_{M2}\cos(x_1 - x_3)\,\mathrm{co}}{} \\ 0 & 0 \\ \dfrac{\sigma_4 + \sigma_9 - \sigma_7 + \sigma_3 - g\sin(x_1)\sin(x_1 - x_3)}{\sigma_{11}} - \sigma_6 & \dfrac{2\,l_1\,x_2\sin(x_1 - x_3)}{\sigma_{11}} \qquad \sigma_6 - \dfrac{\sigma_4}{} \end{pmatrix}$$

where

$$\sigma_1 = l_2\,\mu_{M2}\,{x_4}^2\cos(x_1 - x_3)$$

$$\sigma_2 = \mu_{M2}\sin(x_1 - x_3)\sin(x_3)$$

$$\sigma_3 = g\cos(x_1)\cos(x_1 - x_3)$$

$$\sigma_4 = l_1\,{x_2}^2\cos(x_1 - x_3)$$

$$\sigma_5 = \frac{2\,\mu_{M2}\cos(x_1 - x_3)\sin(x_1 - x_3)\ \left(l_1\,\mu_{M2}\cos(x_1 - x_3)\sin(x_1 - x_3)\,{x_2}^2 + l_2\,\mu_{M2}\sin(x_1 - x_3)\,{x_4}^2 + g\ (\sin(x\right.}{l_1\,\sigma_{13}}$$

$$\sigma_6 = \frac{2\,\mu_{M2}\cos(x_1 - x_3)\sin(x_1 - x_3)\ \left(l_1\sin(x_1 - x_3)\,{x_2}^2 + l_2\,\mu_{M2}\cos(x_1 - x_3)\sin(x_1 - x_3)\,{x_4}^2 + g\sin(x_1 - x_3\right.}{l_2\,\sigma_{13}}$$

$$\sigma_7 = l_2\,\mu_{M2}\,{x_4}^2\,\sigma_{15}$$

$$\sigma_8 = l_1\,\mu_{M2}\,{x_2}^2\,\sigma_{15}$$

$$\sigma_9 = l_2\,\mu_{M2}\,{x_4}^2\,\sigma_{14}$$

$$\sigma_{10} = l_1\,\mu_{M2}\,{x_2}^2\,\sigma_{14}$$

$$\sigma_{11} = l_2\ (\mu_{M2}\,\sigma_{15} + \mu_{M1})$$

$$\sigma_{12} = l_1\ (\mu_{M2}\,\sigma_{15} + \mu_{M1})$$

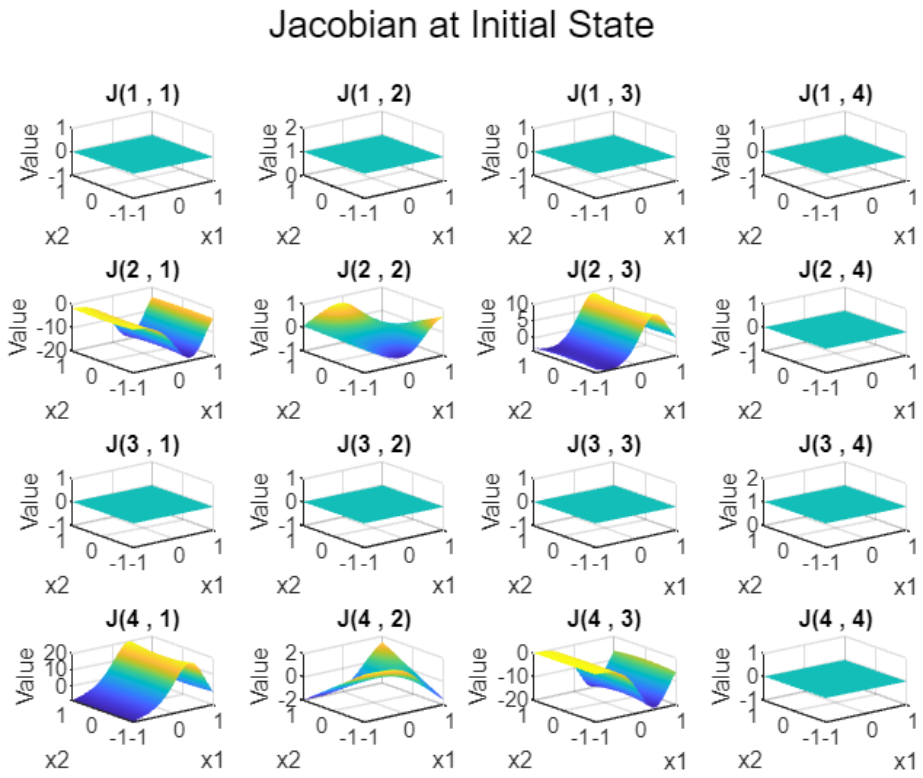
$$\sigma_{13} = (\mu_{M2}\,\sigma_{15} + \mu_{M1})^2$$

$$\sigma_{14} = \cos(x_1 - x_3)^2$$

$$\sigma_{15} = \sin(x_1 - x_3)^2$$

```
state = [pi/4; 0; pi/6; 0];
Jacobian_matrix = computeAndplotJacobian(state);
```

```
Jacobian Matrix at the given state:
         0    1.0000         0         0
  -10.1800         0    4.8686         0
         0         0         0    1.0000
    7.6171         0  -10.9824         0
```

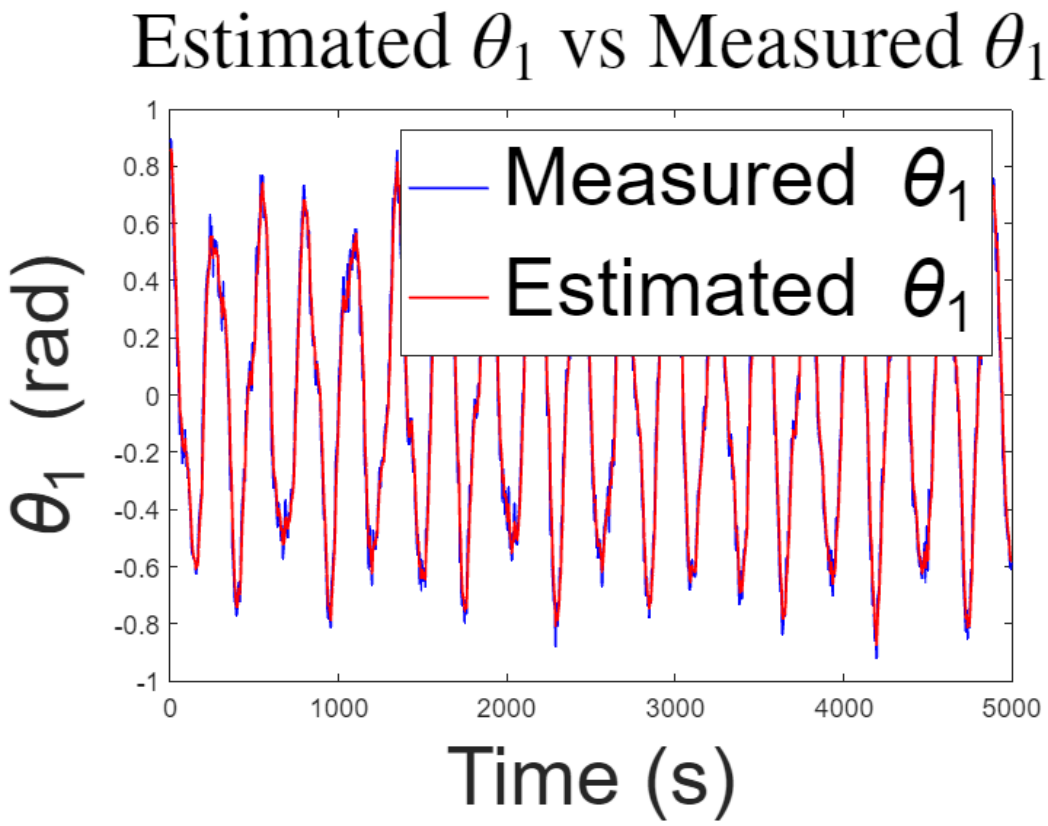



```
H = eye(4);

x_hat_history = zeros(4, num_steps);
P_hat_history = zeros(4, 4, num_steps);

x_hat_history(:, 1) = x_hat;
P_hat_history(:, :, 1) = P;

tic;
for k = 2:num_steps

z = [theta1_measured(k); theta1_dot_measured(k); theta2_measured(k);
theta2_dot_measured(k)];

theta1 = x_hat(1);
theta1_dot = x_hat(2);
theta2 = x_hat(3);
theta2_dot = x_hat(4);
```

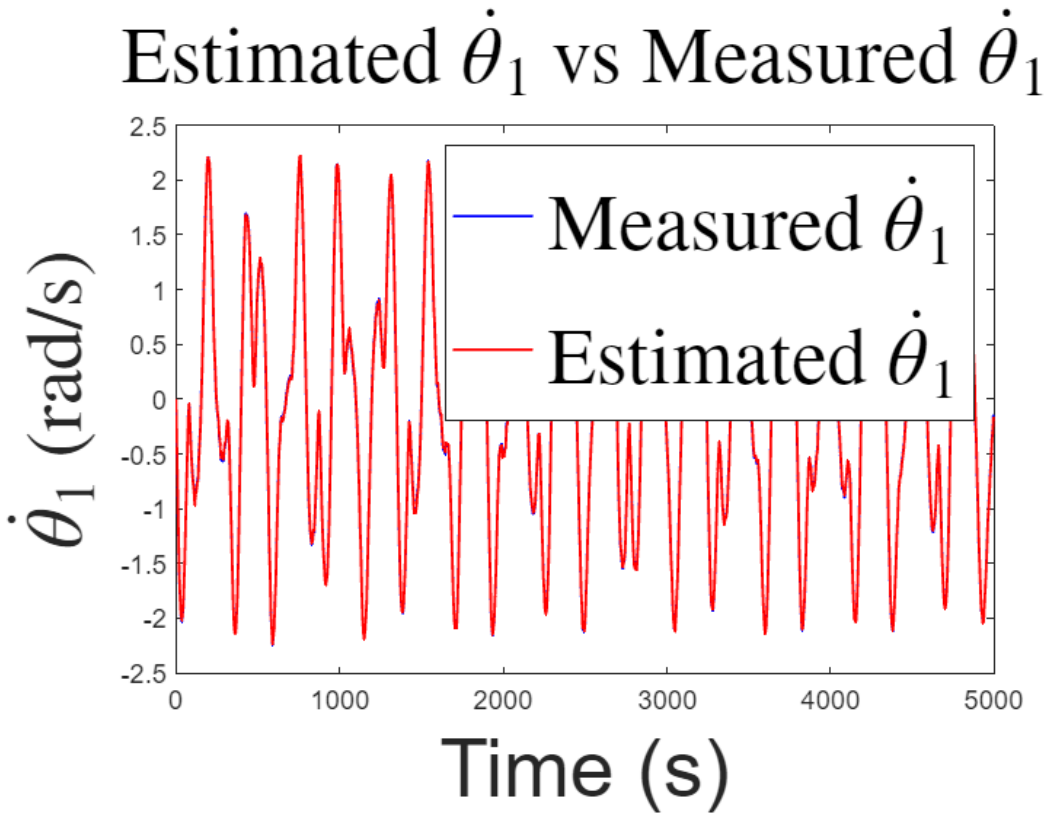
```
[theta1_ddot, theta2_ddot] = dynamic(theta1, theta1_dot, theta2, theta2_dot);

dt = 0.01;

theta1_dot = theta1_dot + theta1_ddot * dt;
theta2_dot = theta2_dot + theta2_ddot * dt;

theta1 = theta1 + theta1_dot * dt;
theta2 = theta2 + theta2_dot * dt;

x_hat = [theta1; theta1_dot; theta2; theta2_dot];

F = computejacobian(x_hat);
P = F * P * F' + Q;

z_hat = H * x_hat;
y = z - z_hat;

K = P * H' / (H * P * H' + R);
x_hat = x_hat + K * y;
P = (eye(size(P)) - K * H) * P;

x_hat_history(:, k) = x_hat;
P_hat_history(:, :, k) = P;

end

elapsed_time = toc;
```

plots

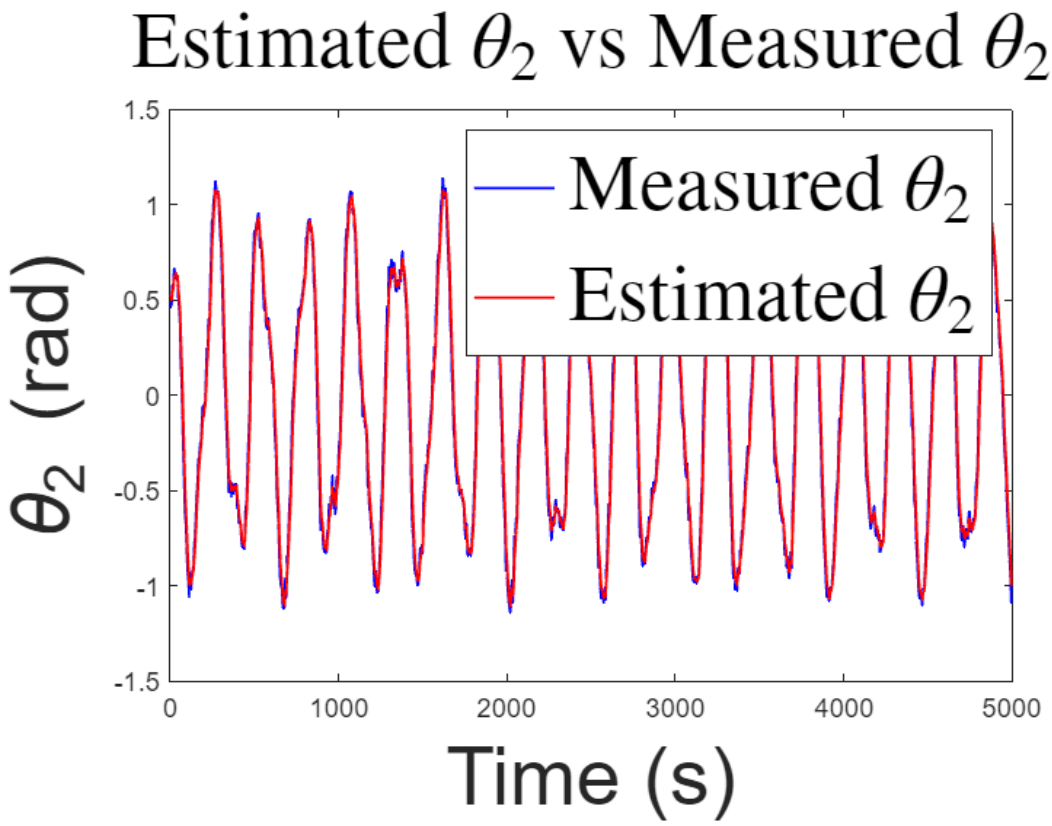
```
figure;
plot(time, theta1_measured, 'DisplayName', 'Measured \theta_1', 'Linewidth', 1.0,
'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(1, :), 'DisplayName', 'Estimated \theta_1', 'Linewidth',
1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('\theta_1 (rad)', 'FontSize', 30);
legend ('FontSize', 30);
title('Estimated $\theta_1$ vs Measured $\theta_1$', 'Interpreter', 'Latex',
'FontSize', 30);
```



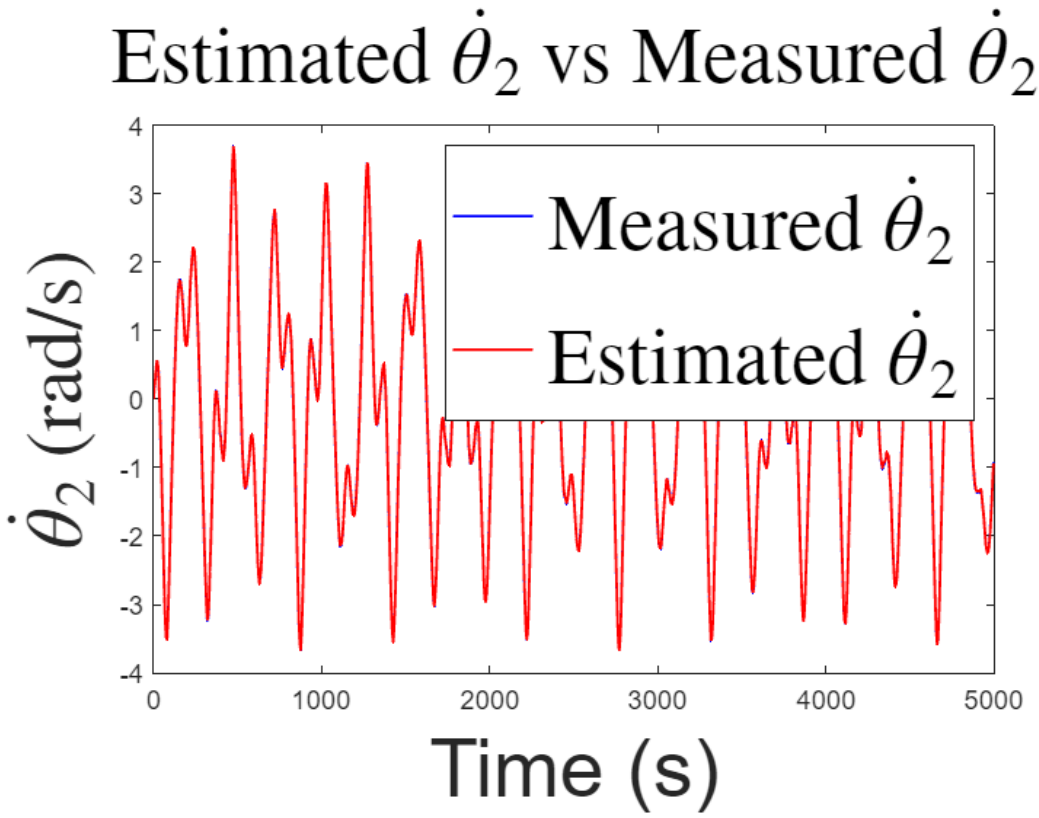

```
figure;
plot(time, theta1_dot_measured, 'DisplayName', 'Measured $\dot{\theta}_1$',
'Linewidth', 1.0, 'Color', 'b', 'LineStyle','-')
hold on
plot(time, x_hat_history(2, :), 'DisplayName', 'Estimated $\dot{\theta}_1$',
'Linewidth', 1.0, 'Color', 'r', 'LineStyle','-')
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('$\dot{\theta}_1$ (rad/s)', 'Interpreter', 'latex', 'FontSize', 30);
legend('Interpreter', 'Latex', 'FontSize', 30);
title('Estimated $\dot{\theta}_1$ vs Measured $\dot{\theta}_1$', 'Interpreter',
'Latex', 'FontSize', 30);
```



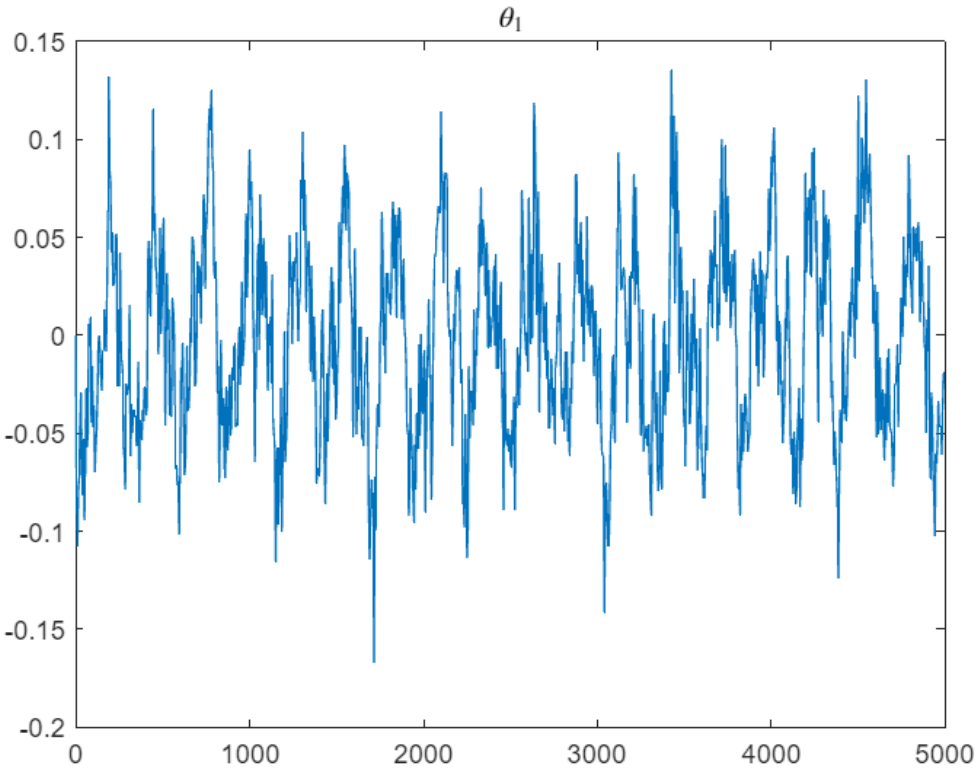

```
figure;
plot(time, theta2_measured, 'DisplayName', 'Measured $\theta_2$', 'Linewidth', 1.0,
'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(3, :), 'DisplayName', 'Estimated $\theta_2$', 'Linewidth',
1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('\theta_2 (rad)', 'FontSize', 30);
legend('Interpreter', 'Latex', 'FontSize', 30);
title('Estimated $\theta_2$ vs Measured $\theta_2$', 'Interpreter', 'Latex',
'FontSize', 30);
```



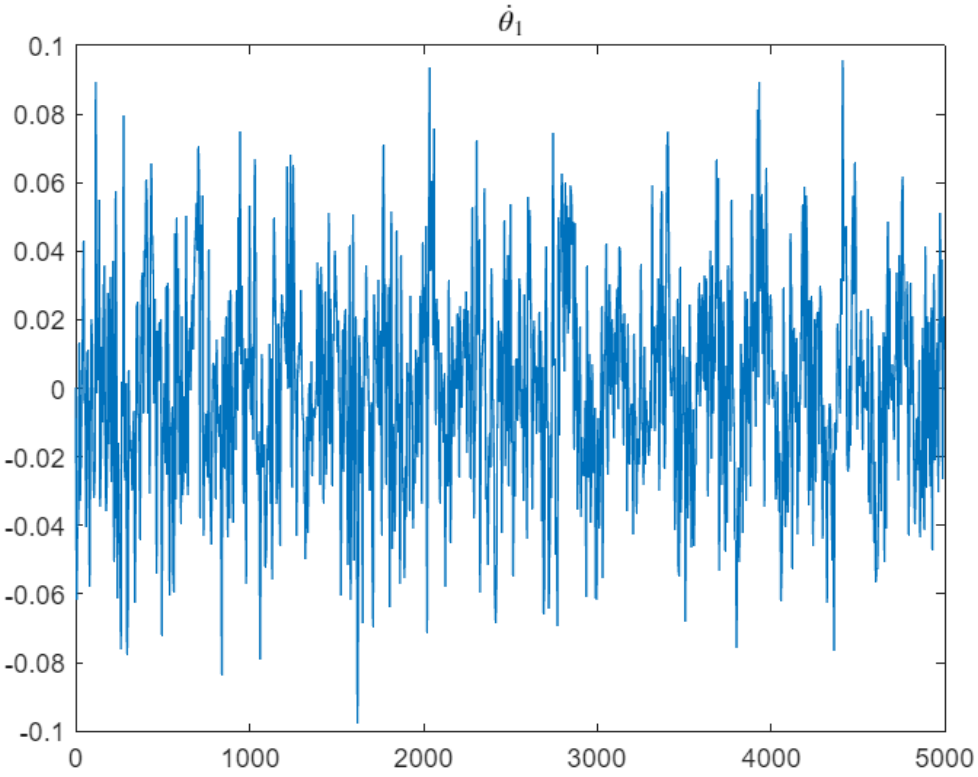

```
figure;
plot(time, theta2_dot_measured, 'DisplayName', 'Measured $\dot{\theta}_2$',
'Linewidth', 1.0, 'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(4, :), 'DisplayName', 'Estimated $\dot{\theta}_2$',
'Linewidth', 1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('$\dot{\theta}_2$ (rad/s)', 'Interpreter', 'latex', 'FontSize', 30);
legend('Interpreter', 'Latex', 'FontSize', 30);
title('Estimated $\dot{\theta}_2$ vs Measured $\dot{\theta}_2$', 'Interpreter',
'Latex', 'FontSize', 30);
```



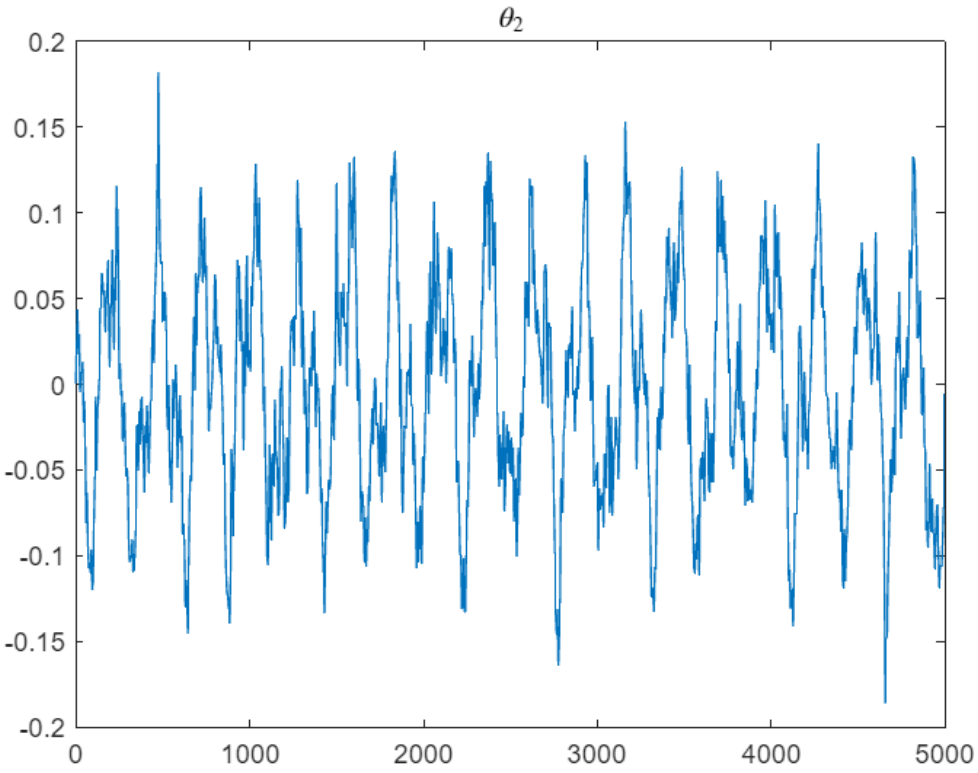

```
error1 = theta1_true' - x_hat_history(1, :);
plot(time, error1)
title('$\theta_1$', 'Interpreter', 'latex');
```



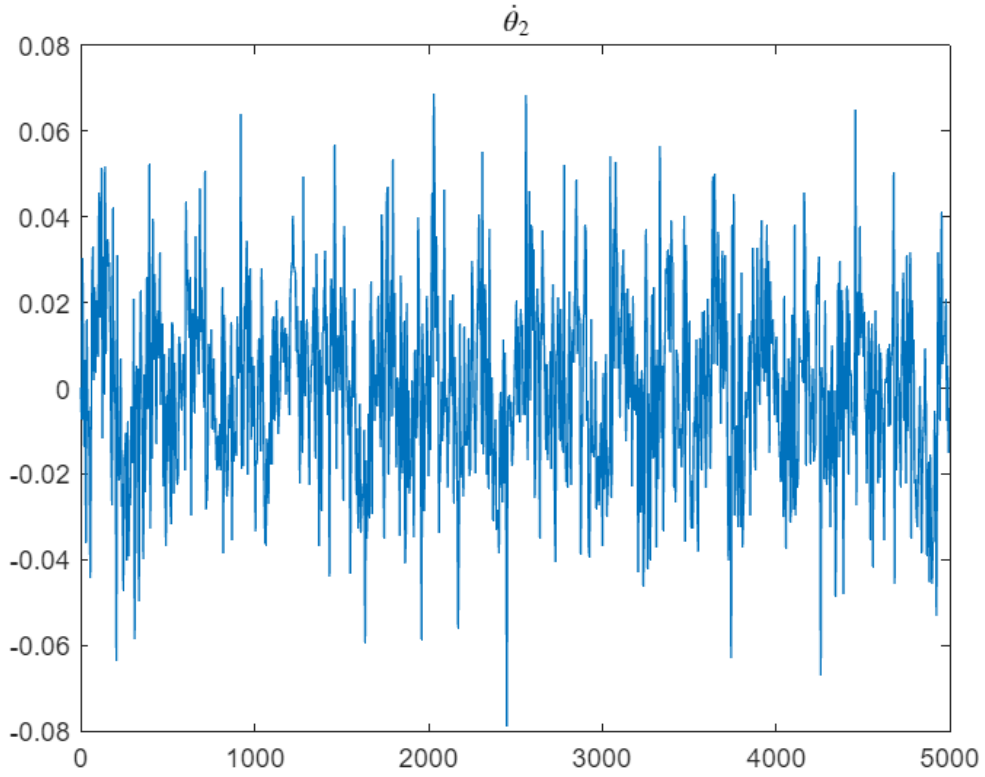

```
error2 = theta1_dot_true' - x_hat_history(2, :);
plot(time, error2)
title('$\dot{\theta}_1$', 'Interpreter', 'latex');
```



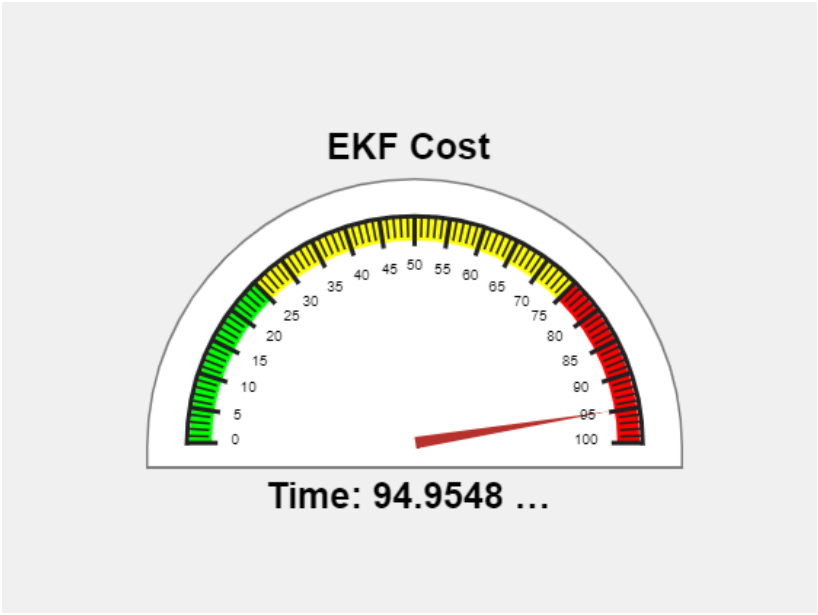

```
error3 = theta2_true' - x_hat_history(3, :);
plot(time, error3);
title('$\theta_2$', 'Interpreter', 'latex');
```




```
error4 = theta2_dot_true' - x_hat_history(4, :);
plot(time, error4);
title('$\dot{\theta}_2$', 'Interpreter', 'latex');
```




```
fig = uifigure('Name', 'EKF Cost');
gauge = uigauge(fig, 'semicircular');
gauge.Position = [100, 100, 400, 200];
gauge.Limits = [0 100];
gauge.Value = elapsed_time;
gauge.MajorTicks = 0:5:100;
gauge.FontSize = 10;

titleLabel = uilabel(fig, 'Text', 'EKF Cost');
titleLabel.Position = [180, 310, 200, 30];
titleLabel.HorizontalAlignment = 'center';
titleLabel.FontSize = 25;
titleLabel.FontWeight = 'bold';

valueLabel = uilabel(fig, 'Text', sprintf('Time: %.4f (s)', gauge.Value));
valueLabel.Position = [180, 70, 200, 30];
valueLabel.HorizontalAlignment = 'center';
valueLabel.FontSize = 25;
valueLabel.FontWeight = 'bold';

gauge.ScaleColors = [0 1 0; 1 1 0; 1 0 0];
gauge.ScaleColorLimits = [0 25; 25 75; 75 100];
```


EKF Cost
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
Time: 94.9548 ...

Mehdi Muzaffari - MAE 674 - Project 1

```
clc;
clear;
close all;
```

load simulated Data

```
data = readtable('simulation');
time = data.Time;
theta1_true = data.Theta1;
theta1_dot_true = data.Theta1_Dot;
theta2_true = data.Theta2;
theta2_dot_true = data.Theta2_Dot;
```

add noise to data

```
noise_std_theta1 = 0.05;
noise_std_theta1_dot = 0.03;
noise_std_theta2 = 0.04;
noise_std_theta2_dot = 0.02;

theta1_measured = theta1_true + noise_std_theta1 * randn(size(theta1_true));
theta1_dot_measured = theta1_dot_true + noise_std_theta1_dot *
randn(size(theta1_dot_true));
theta2_measured = theta2_true + noise_std_theta2 * randn(size(theta2_true));
theta2_dot_measured = theta2_dot_true + noise_std_theta2_dot *
randn(size(theta2_dot_true));
```

simulation parameters

```
dt = 0.01;
num_steps = length(time);

theta1 = pi/4;
theta1_dot = 0;
theta2 = pi/6;
theta2_dot = 0;

x_hat = [pi/4; 0; pi/6; 0];
P = diag([0.01, 0.1, 0.01, 0.01]);
Q = diag([0.01, 0.01, 0.01, 0.01]);
R = diag([0.05, 0.05, 0.05, 0.05]);
```

ukf parameters

```
alpha = 1e-3;
kappa = 0;
beta = 2;
```

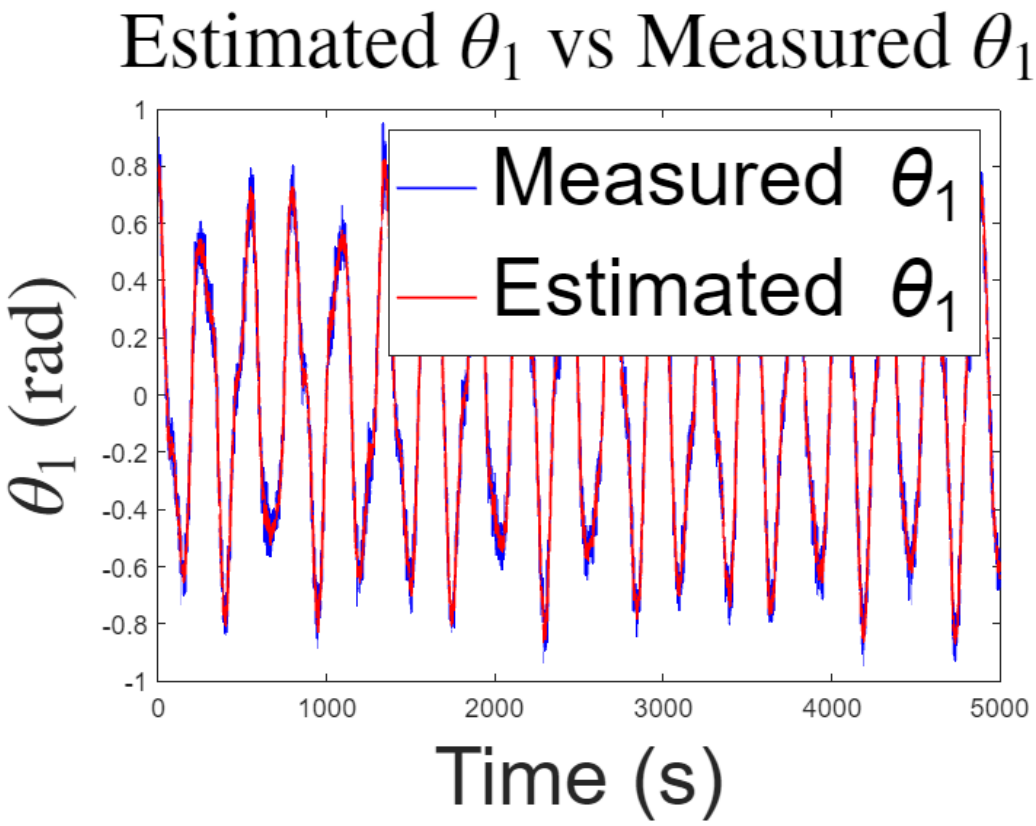
```
n = 4;
lambda = alpha.^2 * (n + kappa) - n;
gamma = sqrt(n + lambda);

Wm = [lambda / (n + lambda), repmat(0.5 / (n + lambda), 1, 2 * n)];
Wc = Wm;
Wc(1) = Wc(1) + (1 - alpha^2 + beta);
```

```
x_hat_history = zeros(n, num_steps);
P_hat_history = zeros(n, n, num_steps);

x_hat_history(:, 1) = x_hat;
P_hat_history(:, :, 1) = P;

tic;
for k = 2:num_steps

    X = [x_hat, x_hat + gamma * chol(P)', x_hat - gamma * chol(P)'];

    X_pred = zeros(size(X));

    for i = 1:size(X, 2)

        theta1 = X(1, i);
        theta1_dot = X(2, i);
        theta2 = X(3, i);
        theta2_dot = X(4, i);

        [theta1_ddot, theta2_ddot] = dynamic(theta1, theta1_dot, theta2,
theta2_dot);

        X_pred(:, i) = [theta1 + dt * theta1_dot; theta1_dot + dt * theta1_ddot;
            theta2 + dt * theta2_dot; theta2_dot + dt * theta2_ddot];
    end

    x_pred = X_pred * Wm';

    P_pred = Q;
    for i = 1:size(X, 2)
        P_pred = P_pred +  Wc(i) * (X_pred(:, i) - x_pred) * (X_pred(:, i) -
x_pred)';
    end

    X_update = [x_pred, x_pred + gamma * chol(P_pred)', x_pred - gamma *
chol(P_pred)'];

    Z_pred = X_update;
    z_pred = Z_pred * Wm';
```

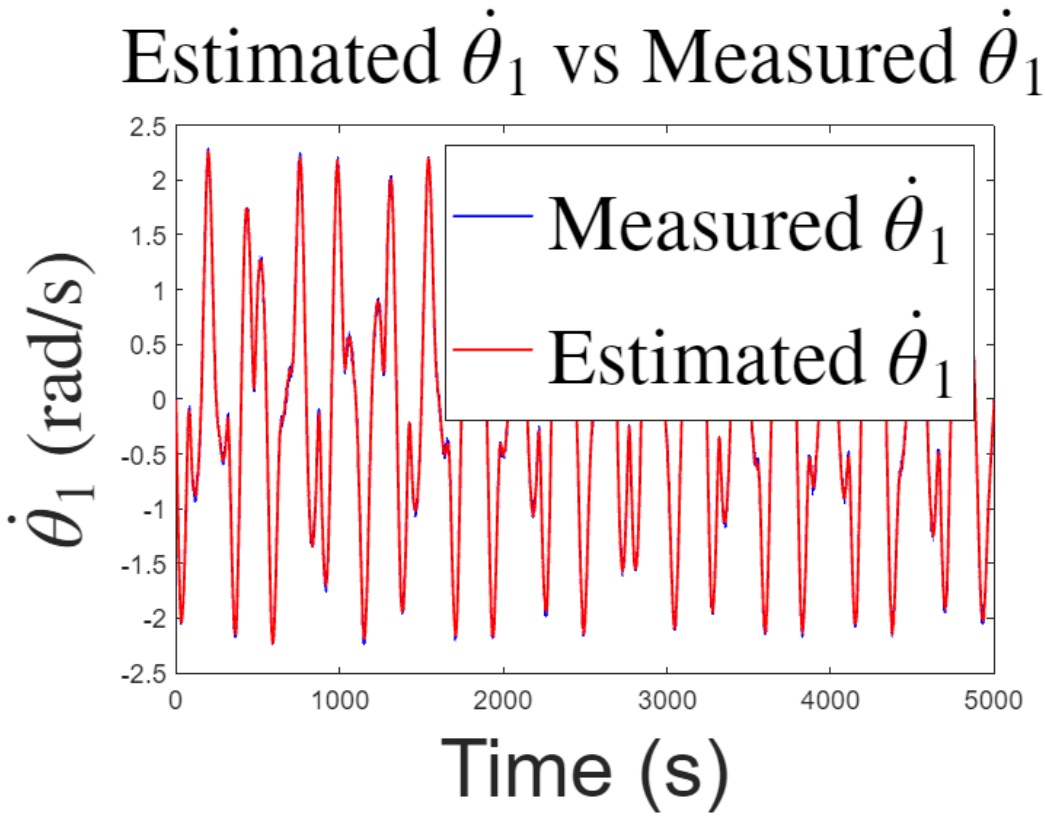
```
    S = R;
    for i = 1:size(X_update, 2)
        S = S + Wc(i) * (Z_pred(:, i) - z_pred) * (Z_pred(:, i) - z_pred)';
    end

    C = zeros(n);
    for i = 1:size(X_update, 2)
        C = C + Wc(i) * (X_pred(:, i) - x_pred) * (Z_pred(:, i) - z_pred)';
    end

    K = C / S;
    z = [theta1_measured(k); theta1_dot_measured(k); theta2_measured(k);
theta2_dot_measured(k)];
    x_hat = x_pred + K * (z - z_pred);
    P = P_pred - K * S * K';

    x_hat_history(:, k) = x_hat;
    P_hat_history(:, :, k) = P;

end

elapsed_time = toc;
```

plots

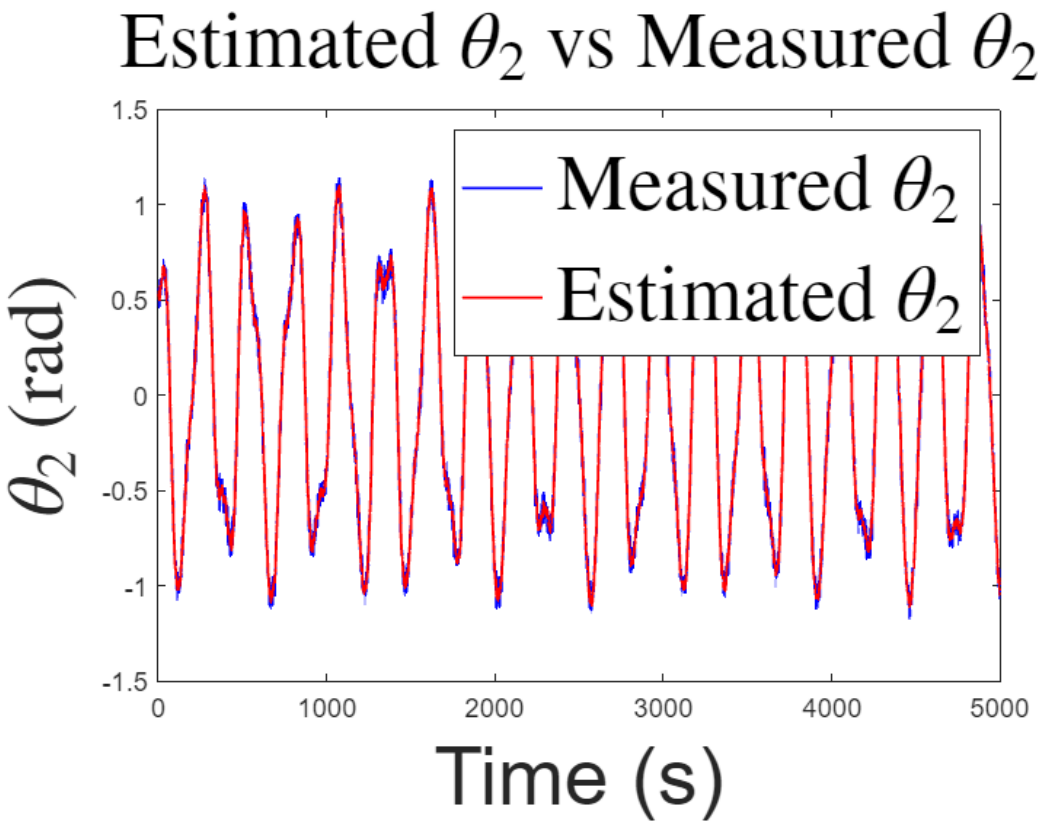
```
figure;
plot(time, theta1_measured, 'DisplayName', 'Measured \theta_1', 'Linewidth', 1.0,
'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(1, :), 'DisplayName', 'Estimated \theta_1', 'Linewidth',
1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('$\theta_1$ (rad)', 'Interpreter', 'latex', 'FontSize', 30);
legend ('FontSize', 30);
title('Estimated $\theta_1$ vs Measured $\theta_1$', 'Interpreter', 'Latex',
'FontSize', 30);
```



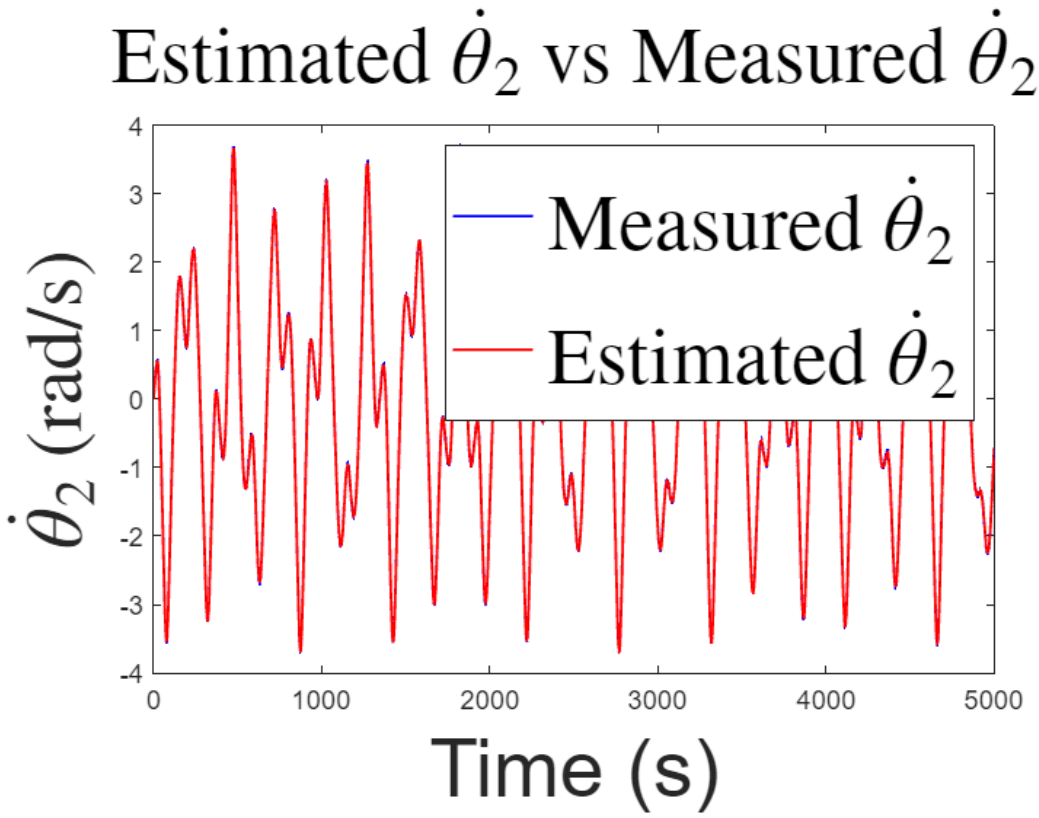

```
figure;
plot(time, theta1_dot_measured, 'DisplayName', 'Measured $\dot{\theta}_1$',
'Linewidth', 1.0, 'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(2, :), 'DisplayName', 'Estimated $\dot{\theta}_1$',
'Linewidth', 1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('$\dot{\theta}_1$ (rad/s)', 'Interpreter', 'latex', 'FontSize', 30);
legend('Interpreter', 'Latex', 'FontSize', 30);
title('Estimated $\dot{\theta}_1$ vs Measured $\dot{\theta}_1$', 'Interpreter',
'Latex', 'FontSize', 30);
```



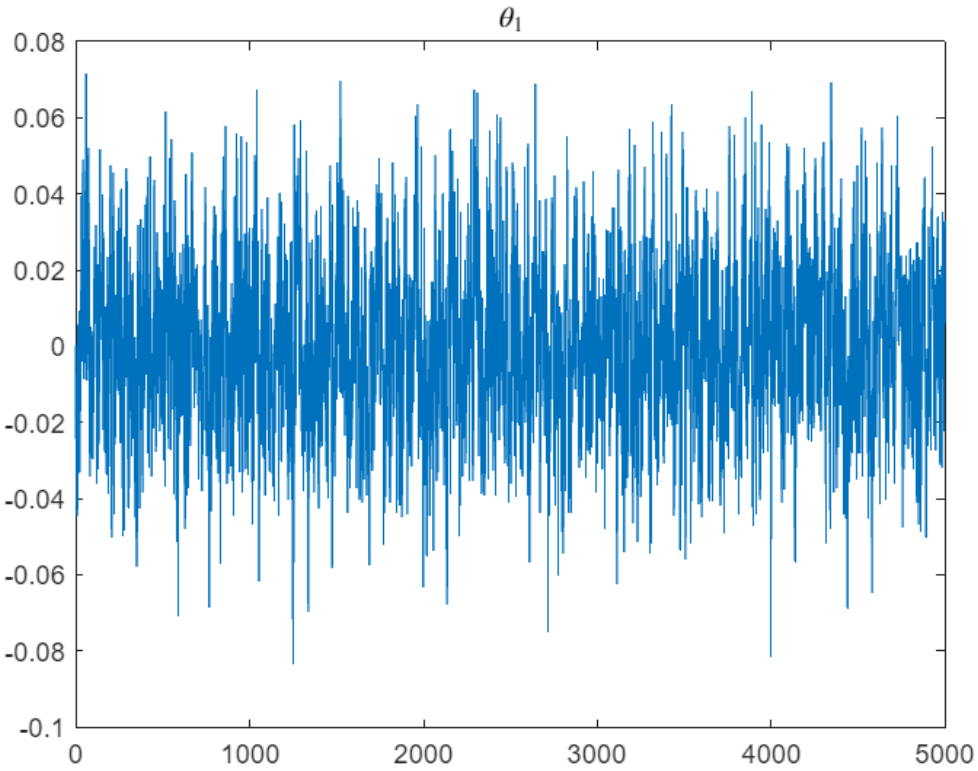

```
figure;
plot(time, theta2_measured, 'DisplayName', 'Measured $\theta_2$', 'Linewidth', 1.0,
'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(3, :), 'DisplayName', 'Estimated $\theta_2$', 'Linewidth',
1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('$\theta_2$ (rad)', 'Interpreter', 'Latex', 'FontSize', 30);
legend('Interpreter', 'Latex', 'FontSize', 30);
title('Estimated $\theta_2$ vs Measured $\theta_2$', 'Interpreter', 'Latex',
'FontSize', 30);
```



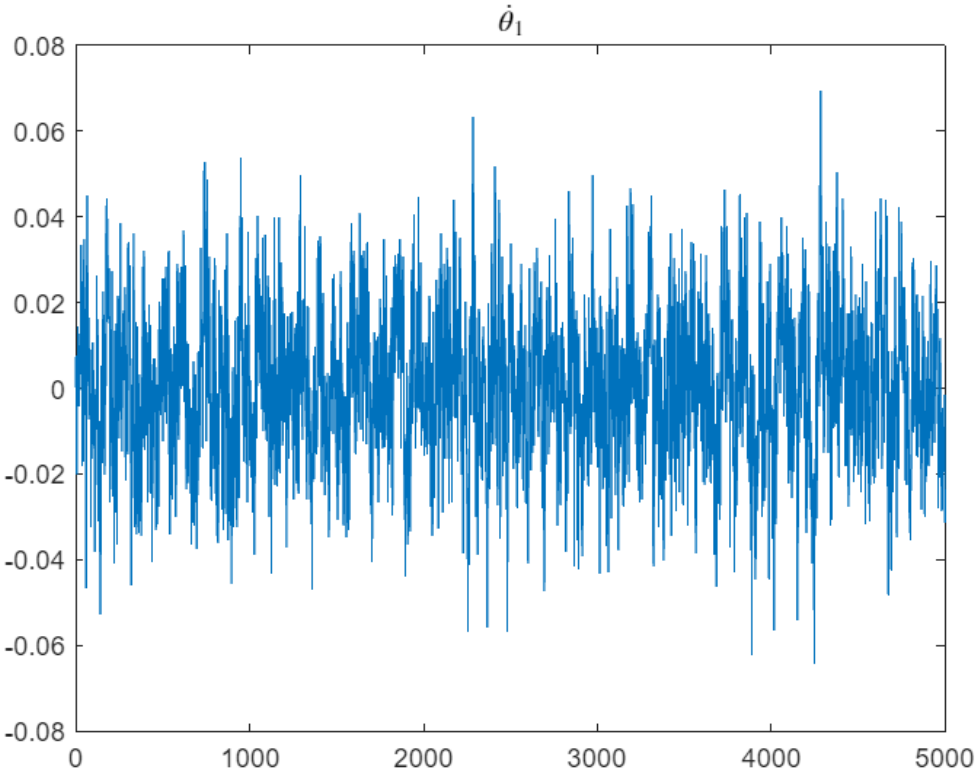

```
figure;
plot(time, theta2_dot_measured, 'DisplayName', 'Measured $\dot{\theta}_2$',
'Linewidth', 1.0, 'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(4, :), 'DisplayName', 'Estimated $\dot{\theta}_2$',
'Linewidth', 1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('$\dot{\theta}_2$ (rad/s)', 'Interpreter', 'latex', 'FontSize', 30);
legend('Interpreter', 'Latex', 'FontSize', 30);
title('Estimated $\dot{\theta}_2$ vs Measured $\dot{\theta}_2$', 'Interpreter',
'Latex', 'FontSize', 30);
```



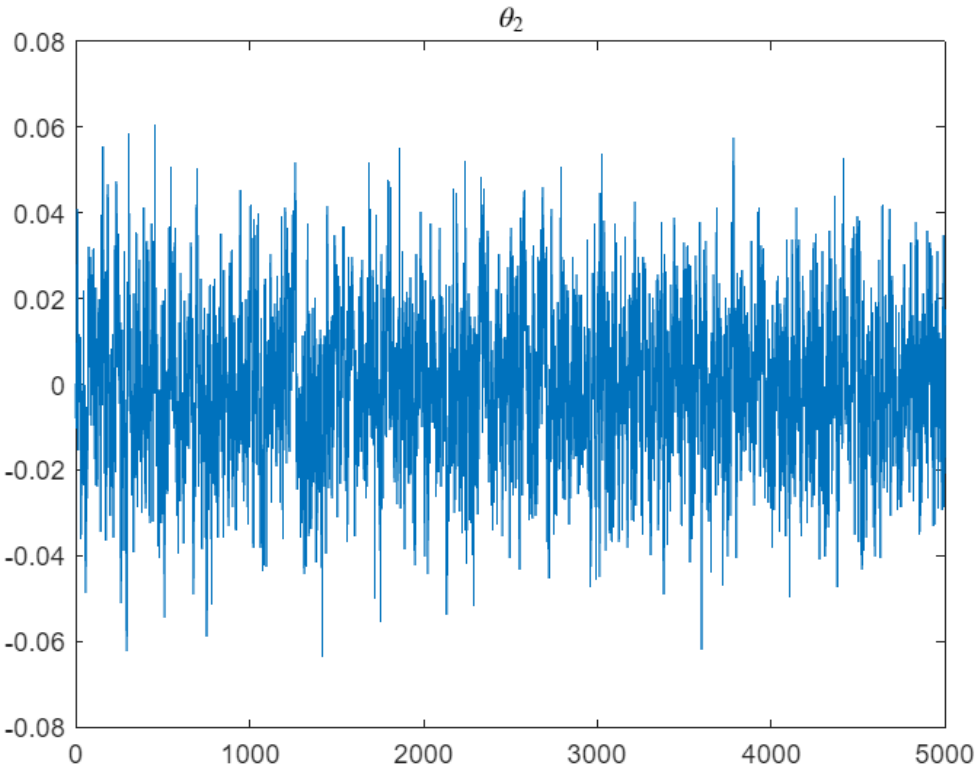

```
error1 = theta1_true' - x_hat_history(1, :);
plot(time, error1)
title('$\theta_1$', 'Interpreter', 'latex');
```



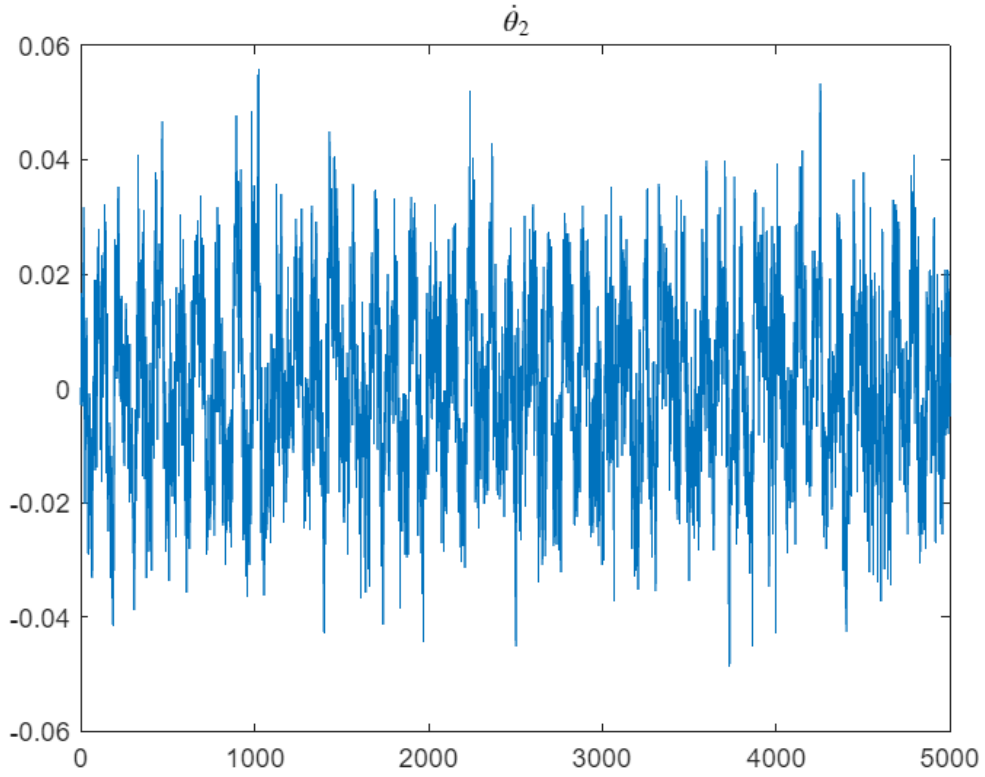

```
error2 = theta1_dot_true' - x_hat_history(2, :);
plot(time, error2)
title('$\dot{\theta}_1$', 'Interpreter', 'latex');
```



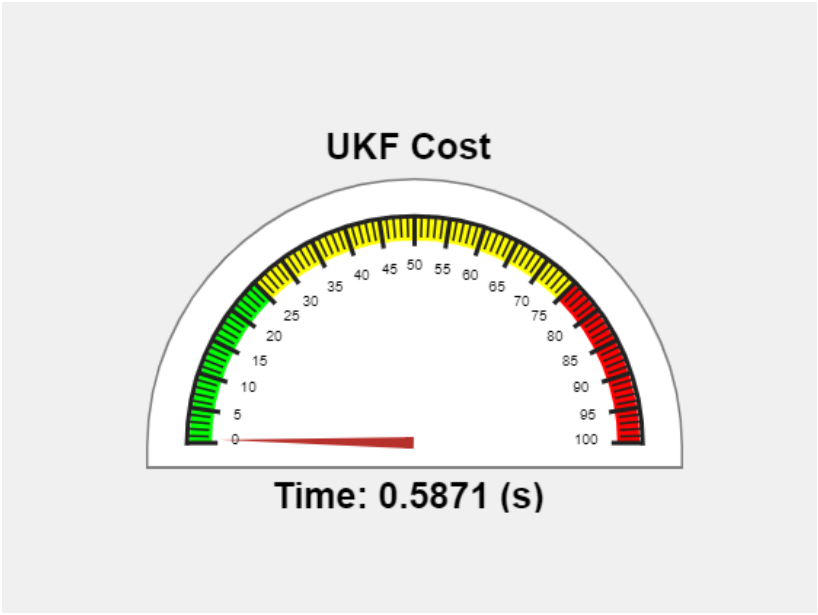

```
error3 = theta2_true' - x_hat_history(3, :);
plot(time, error3);
title('$\theta_2$', 'Interpreter', 'latex');
```




```
error4 = theta2_dot_true' - x_hat_history(4, :);
plot(time, error4);
title('$\dot{\theta}_2$', 'Interpreter', 'latex');
```




```
fig = uifigure('Name', 'UKF Cost');
gauge = uigauge(fig, 'semicircular');
gauge.Position = [100, 100, 400, 200];
gauge.Limits = [0 100];
gauge.Value = elapsed_time;
gauge.MajorTicks = 0:5:100;
gauge.FontSize = 10;

titleLabel = uilabel(fig, 'Text', 'UKF Cost');
titleLabel.Position = [180, 310, 200, 30];
titleLabel.HorizontalAlignment = 'center';
titleLabel.FontSize = 25;
titleLabel.FontWeight = 'bold';

valueLabel = uilabel(fig, 'Text', sprintf('Time: %.4f (s)', gauge.Value));
valueLabel.Position = [180, 70, 200, 30];
valueLabel.HorizontalAlignment = 'center';
valueLabel.FontSize = 25;
valueLabel.FontWeight = 'bold';

gauge.ScaleColors = [0 1 0; 1 1 0; 1 0 0];
gauge.ScaleColorLimits = [0 25; 25 75; 75 100];
```


UKF Cost
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
Time: 0.5871 (s)

Mehdi Muzaffari - MAE 674 - Project 1

```
clc;
clear;
close all;
```

load simulated Data

```
data = readtable('simulation');
time = data.Time;
theta1_true = data.Theta1;
theta1_dot_true = data.Theta1_Dot;
theta2_true = data.Theta2;
theta2_dot_true = data.Theta2_Dot;
```

add noise to data

```
noise_std_theta1 = 0.05;
noise_std_theta1_dot = 0.03;
noise_std_theta2 = 0.04;
noise_std_theta2_dot = 0.02;

theta1_measured = theta1_true + noise_std_theta1 * randn(size(theta1_true));
theta1_dot_measured = theta1_dot_true + noise_std_theta1_dot *
randn(size(theta1_dot_true));
theta2_measured = theta2_true + noise_std_theta2 * randn(size(theta2_true));
theta2_dot_measured = theta2_dot_true + noise_std_theta2_dot *
randn(size(theta2_dot_true));
```

simulation parameters

```
dt = 0.01;
num_steps = length(time);

theta1 = pi/4;
theta1_dot = 0;
theta2 = pi/6;
theta2_dot = 0;

x_hat = [pi/4; 0; pi/6; 0];
P = diag([0.01, 0.1, 0.01, 0.01]);
Q = diag([0.01, 0.01, 0.01, 0.01]);
R = diag([0.05, 0.05, 0.05, 0.05]);
```

ukf parameters

```
alpha = 1e-3;
kappa = 0;
beta = 2;
```

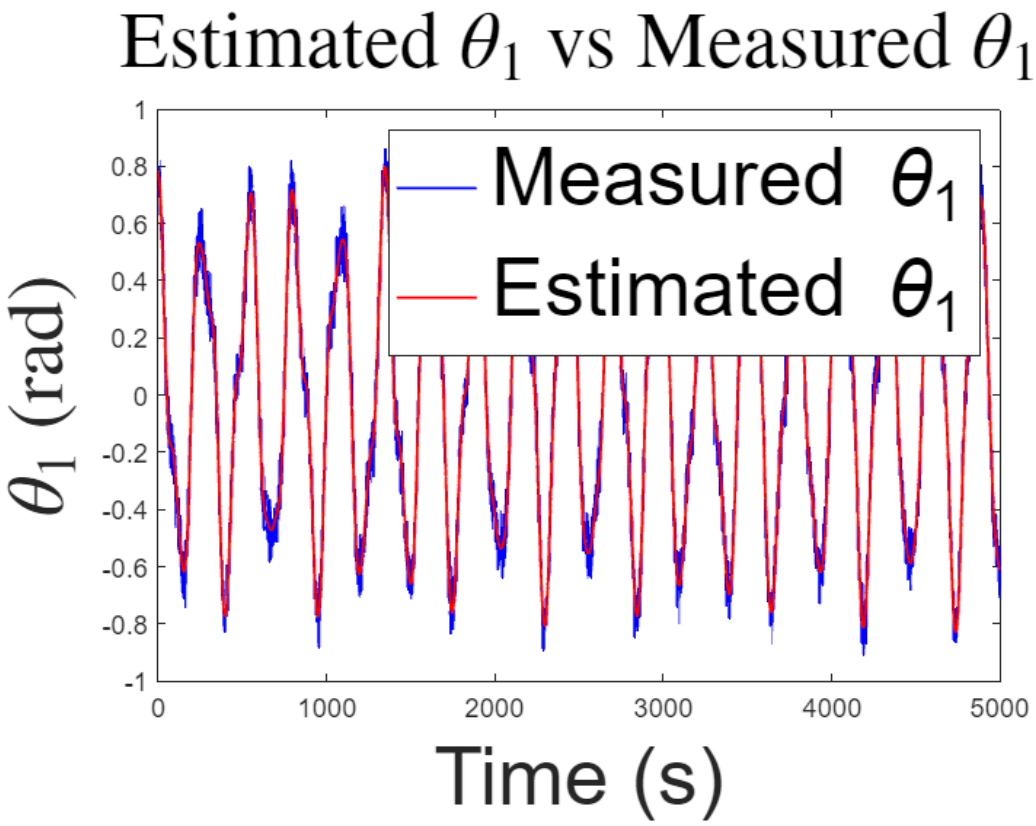
```
n = 4;
lambda = alpha.^2 * (n + kappa) - n;
gamma = sqrt(n + lambda);

Wm = [lambda / (n + lambda), repmat(0.5 / (n + lambda), 1, 2 * n)];
Wc = Wm;
Wc(1) = Wc(1) + (1 - alpha^2 + beta);
```

```
x_hat_history = zeros(n, num_steps);
P_hat_history = zeros(n, n, num_steps);

x_hat_history(:, 1) = x_hat;
P_hat_history(:, :, 1) = P;
epsilon = 1e-3;

tic;
for k = 2:num_steps

    P = (P + P') / 2;
    P = P + epsilon * eye(size(P));
    [~, p] = chol(P);
    if p > 0
        error('Covariance matrix P is not positive definite.');
    end

    X = [x_hat, x_hat + gamma * chol(P)', x_hat - gamma * chol(P)'];

    X_pred = zeros(size(X));

    for i = 1:size(X, 2)

        theta1 = X(1, i);
        theta1_dot = X(2, i);
        theta2 = X(3, i);
        theta2_dot = X(4, i);

        [theta1_ddot, theta2_ddot] = dynamic(theta1, theta1_dot, theta2,
theta2_dot);

        X_pred(:, i) = [theta1 + dt * theta1_dot; theta1_dot + dt * theta1_ddot;
            theta2 + dt * theta2_dot; theta2_dot + dt * theta2_ddot];
    end

    x_pred = X_pred * Wm';

    P_pred = Q;
    for i = 1:size(X, 2)
        P_pred = P_pred + Wc(i) * (X_pred(:, i) - x_pred) * (X_pred(:, i) -
x_pred)';
```

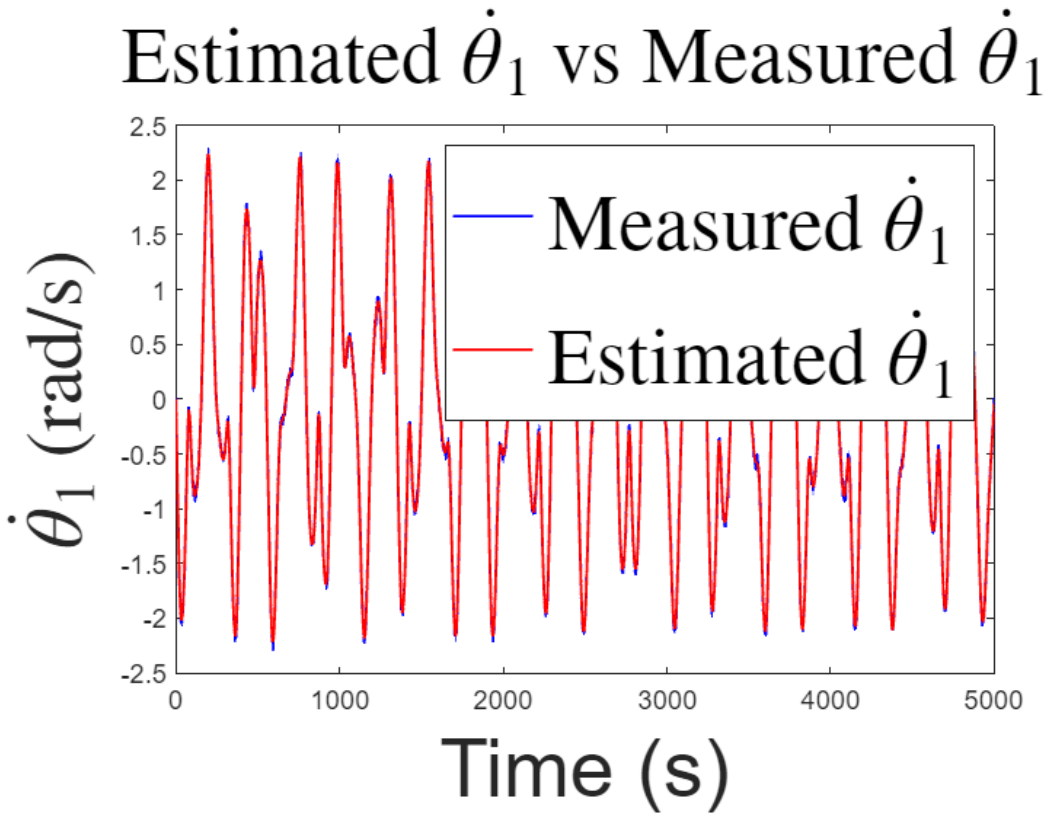
```
    end

    P_pred = (P_pred + P_pred') / 2;
    P_pred = P_pred + epsilon * eye(size(P_pred));

    Z_pred = X_pred;
    z_pred = Z_pred * Wm';
    S = R;
    for i = 1:size(Z_pred, 2)
        S = S + Wc(i) * (Z_pred(:, i) - z_pred) * (Z_pred(:, i) - z_pred)';
    end
    S = S + epsilon * eye(size(S));

    C = zeros(n);
    for i = 1:size(Z_pred, 2)
        C = C + Wc(i) * (X_pred(:, i) - x_pred) * (Z_pred(:, i) - z_pred)';
    end

    K = C / S;
    z = [theta1_true(k); theta1_dot_true(k); theta2_true(k); theta2_dot_true(k)];
    innovation = z - z_pred;
    x_hat = x_pred + K * innovation;
    P = P_pred - K * S * K';

    Q = adapt_Q(innovation);
    R = adapt_R(innovation);

    x_hat_history(:, k) = x_hat;
    P_history(:, :, k) = P;
end

elapsed_time = toc;
```

plots

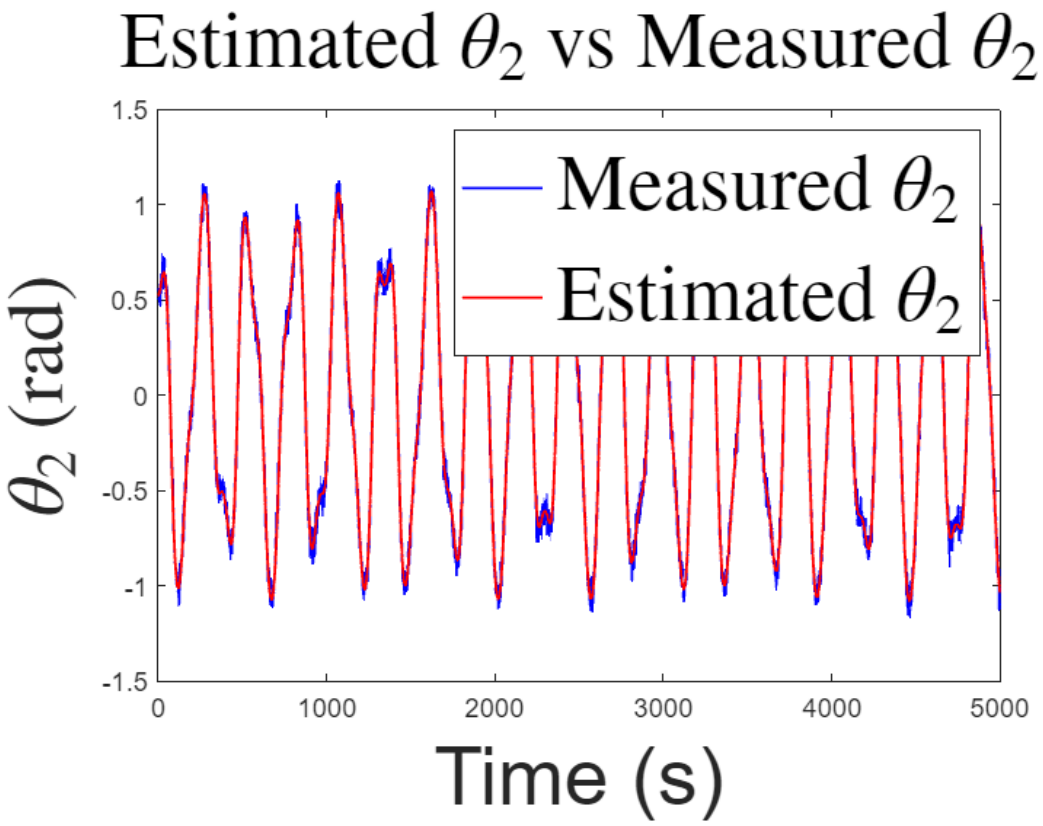
```
figure;
plot(time, theta1_measured, 'DisplayName', 'Measured \theta_1', 'Linewidth', 1.0,
'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(1, :), 'DisplayName', 'Estimated \theta_1', 'Linewidth',
1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('$\theta_1$ (rad)', 'Interpreter', 'latex', 'FontSize', 30);
legend ('FontSize', 30);
title('Estimated $\theta_1$ vs Measured $\theta_1$', 'Interpreter', 'Latex',
'FontSize', 30);
```



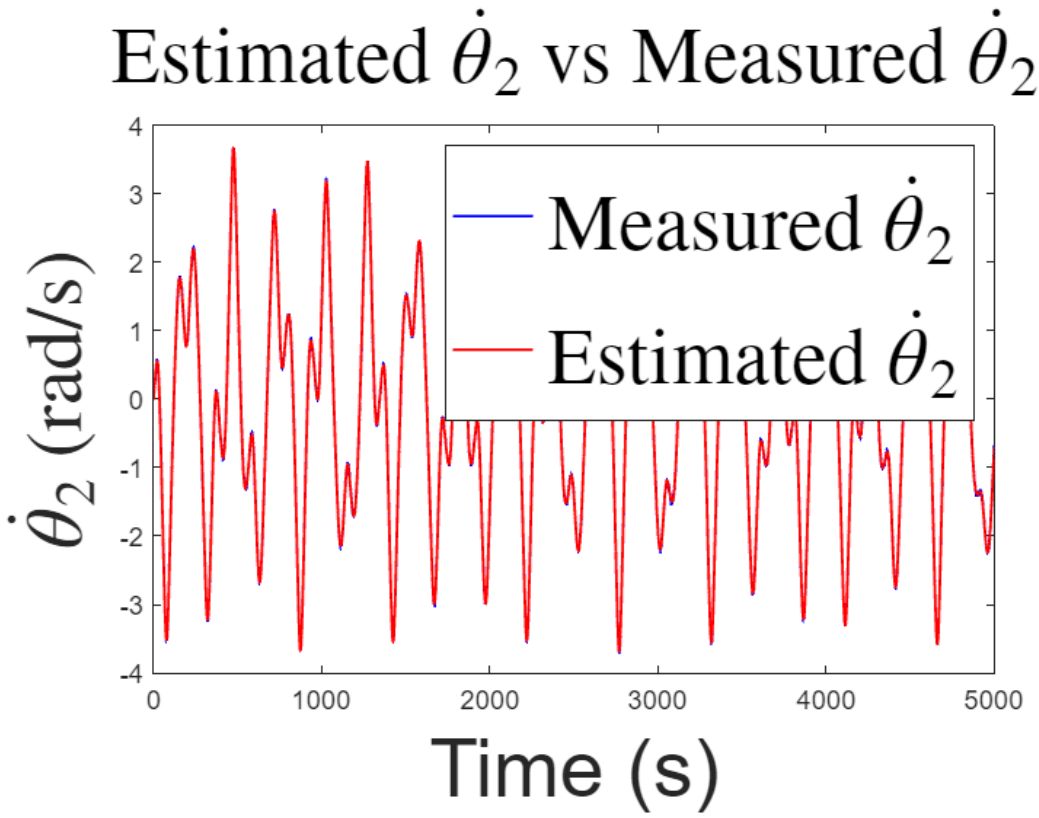

```
figure;
plot(time, theta1_dot_measured, 'DisplayName', 'Measured $\dot{\theta}_1$',
'Linewidth', 1.0, 'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(2, :), 'DisplayName', 'Estimated $\dot{\theta}_1$',
'Linewidth', 1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('$\dot{\theta}_1$ (rad/s)', 'Interpreter', 'latex', 'FontSize', 30);
legend('Interpreter', 'Latex', 'FontSize', 30);
title('Estimated $\dot{\theta}_1$ vs Measured $\dot{\theta}_1$', 'Interpreter',
'Latex', 'FontSize', 30);
```



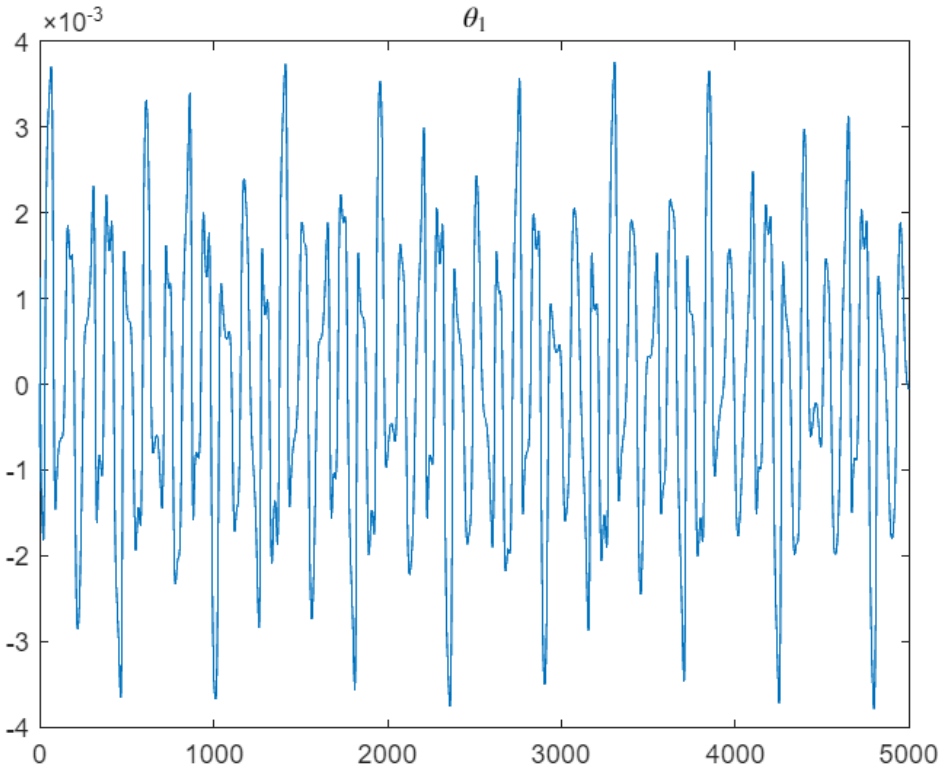

```
figure;
plot(time, theta2_measured, 'DisplayName', 'Measured $\theta_2$', 'Linewidth', 1.0,
'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(3, :), 'DisplayName', 'Estimated $\theta_2$', 'Linewidth',
1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('$\theta_2$ (rad)', 'Interpreter', 'Latex', 'FontSize', 30);
legend('Interpreter', 'Latex', 'FontSize', 30);
title('Estimated $\theta_2$ vs Measured $\theta_2$', 'Interpreter', 'Latex',
'FontSize', 30);
```



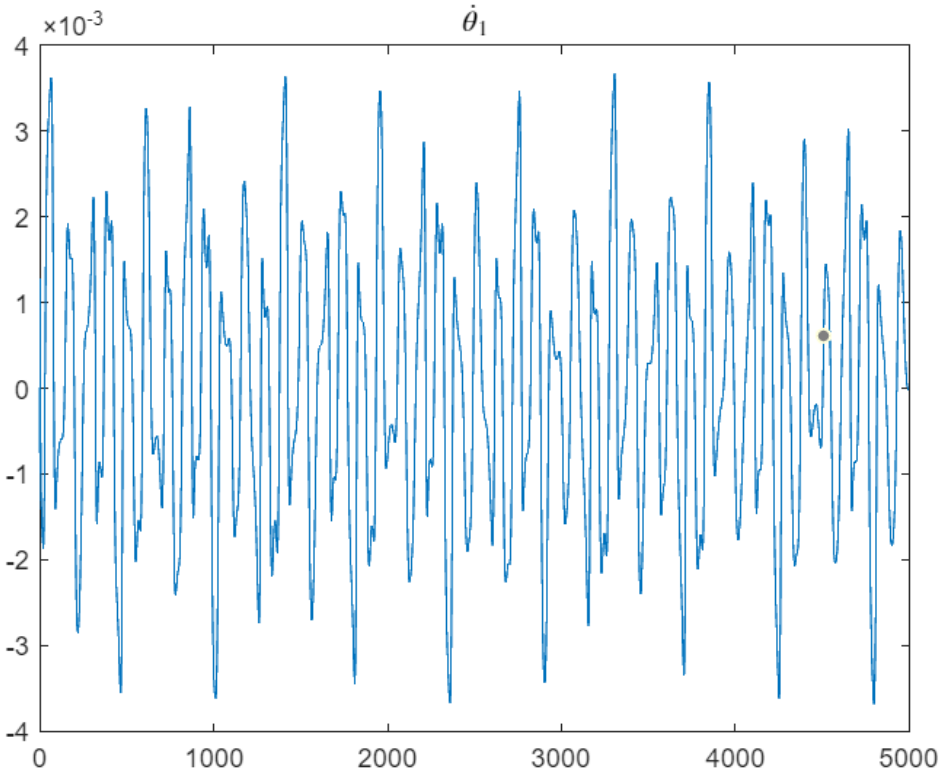

```
figure;
plot(time, theta2_dot_measured, 'DisplayName', 'Measured $\dot{\theta}_2$',
'Linewidth', 1.0, 'Color', 'b', 'LineStyle','-');
hold on
plot(time, x_hat_history(4, :), 'DisplayName', 'Estimated $\dot{\theta}_2$',
'Linewidth', 1.0, 'Color', 'r', 'LineStyle','-');
hold off
xlabel('Time (s)', 'FontSize', 30);
ylabel('$\dot{\theta}_2$ (rad/s)', 'Interpreter', 'latex', 'FontSize', 30);
legend('Interpreter', 'Latex', 'FontSize', 30);
title('Estimated $\dot{\theta}_2$ vs Measured $\dot{\theta}_2$', 'Interpreter',
'Latex', 'FontSize', 30);
```



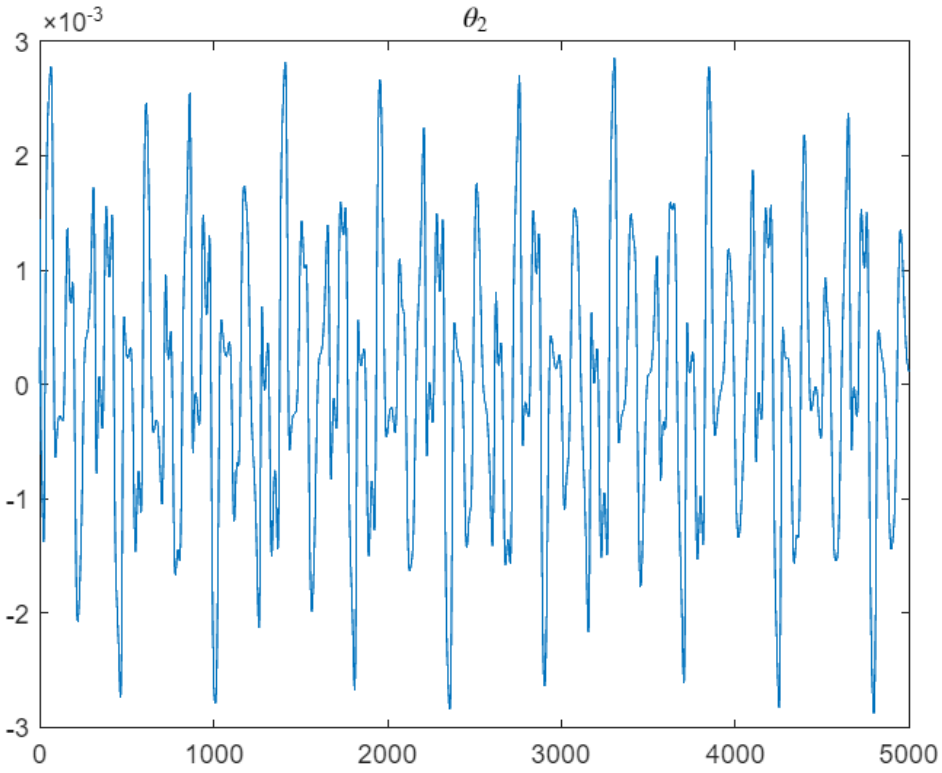

```
error1 = theta1_true' - x_hat_history(1, :);
plot(time, error1)
title('$\theta_1$', 'Interpreter', 'latex');
```



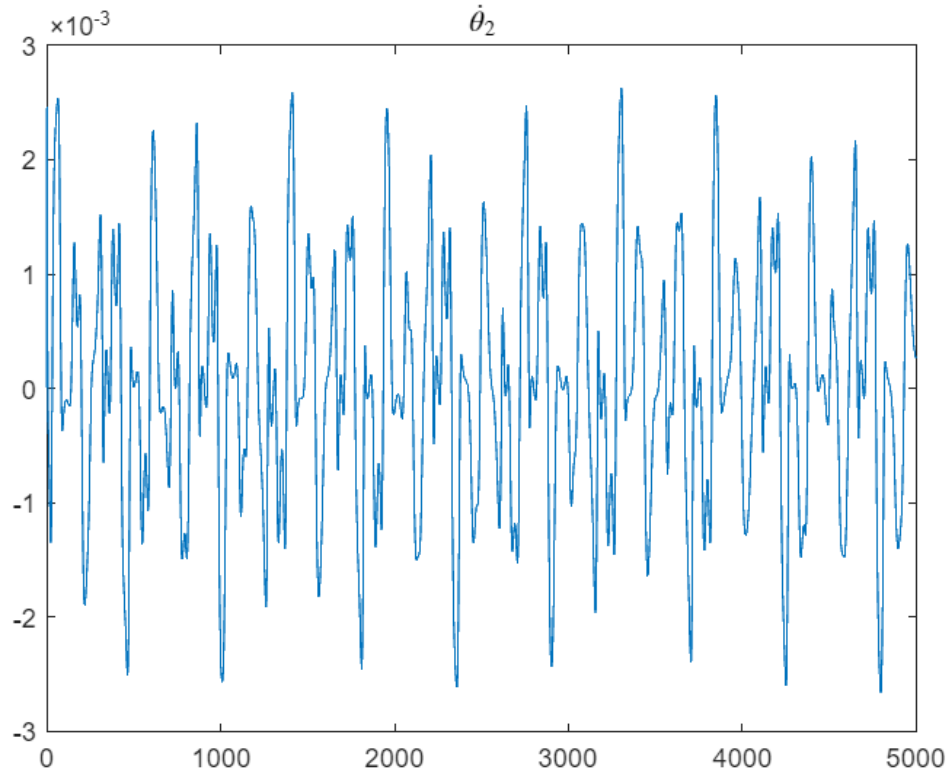

```
error2 = theta1_dot_true' - x_hat_history(2, :);
plot(time, error2)
title('$\dot{\theta}_1$', 'Interpreter', 'latex');
```



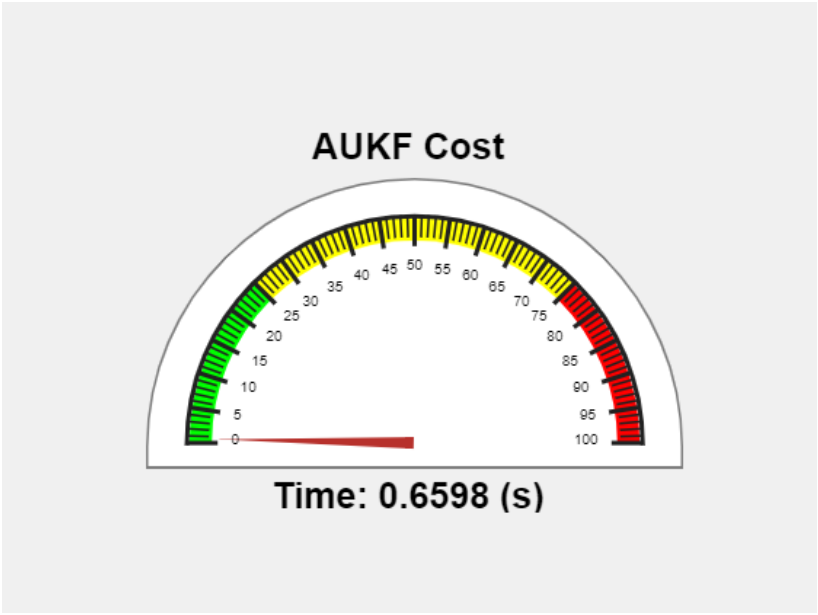

```
error3 = theta2_true' - x_hat_history(3, :);
plot(time, error3);
title('$\theta_2$', 'Interpreter', 'latex');
```




```
error4 = theta2_dot_true' - x_hat_history(4, :);
plot(time, error4);
title('$\dot{\theta}_2$', 'Interpreter', 'latex');
```




```
fig = uifigure('Name', 'UKF Cost');
gauge = uigauge(fig, 'semicircular');
gauge.Position = [100, 100, 400, 200];
gauge.Limits = [0 100];
gauge.Value = elapsed_time;
gauge.MajorTicks = 0:5:100;
gauge.FontSize = 10;

titleLabel = uilabel(fig, 'Text', 'AUKF Cost');
titleLabel.Position = [180, 310, 200, 30];
titleLabel.HorizontalAlignment = 'center';
titleLabel.FontSize = 25;
titleLabel.FontWeight = 'bold';

valueLabel = uilabel(fig, 'Text', sprintf('Time: %.4f (s)', gauge.Value));
valueLabel.Position = [180, 70, 200, 30];
valueLabel.HorizontalAlignment = 'center';
valueLabel.FontSize = 25;
valueLabel.FontWeight = 'bold';

gauge.ScaleColors = [0 1 0; 1 1 0; 1 0 0];
gauge.ScaleColorLimits = [0 25; 25 75; 75 100];
```


AUKF Cost
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
Time: 0.6598 (s)

```
function Jacobian_matrix = computeAndplotJacobian(state)

g = 9.81;
l1 = 1.0;
l2 = 1.0;
mu_M1 = 0.5;
mu_M2 = 0.5;

syms x1 x2 x3 x4 real

x1_val = state(1);
x2_val = state(2);
x3_val = state(3);
x4_val = state(4);

beta = x3 - x1;

dx1 = x2;
dx2 = (g * (mu_M2 * cos(beta) * sin(x3) - sin(x1)) + ...
       mu_M2 * l1 * sin(beta) * cos(beta) * x2^2 + ...
       mu_M2 * l2 * sin(beta) * x4^2) / ...
       (l1 * (mu_M1 + mu_M2 * sin(beta)^2));

dx3 = x4;
dx4 = -(l1 * sin(beta) * x2^2 + ...
        l2 * mu_M2 * sin(beta) * cos(beta) * x4^2 + ...
        g * cos(x1) * sin(beta)) / ...
        (l2 * (mu_M1 + mu_M2 * sin(beta)^2));

state_vector = [x1; x2; x3; x4];
equations = [dx1; dx2; dx3; dx4];

Jacobian = jacobian(equations, state_vector);
Jacobian_func = matlabFunction(Jacobian, 'Vars', {x1, x2, x3, x4});

Jacobian_matrix = Jacobian_func(x1_val, x2_val, x3_val, x4_val);

disp('Jacobian Matrix at the given state:');
disp(Jacobian_matrix)

figure;
[X1, X2] = meshgrid(linspace(-1, 1, 50), linspace(-1, 1, 50));
for i = 1:4
    for j = 1:4
       J_element = matlabFunction(Jacobian(i, j), 'Vars', {x1, x2, x3, x4});
       Z = arrayfun(@(x1, x2) J_element(x1, x2, x3_val, x4_val), X1, X2);
       subplot(4, 4, (i-1) * 4 + j);
       surf(X1, X2, Z)
       xlabel('x1'); ylabel('x2'); zlabel('Value');
       title(sprintf('J(%d , %d)', i, j));
       shading interp
    end
end

sgtitle('Jacobian at Initial State');
```

```
end
```

```
Not enough input arguments.

Error in computeAndplotJacobian (line 11)
x1_val = state(1);
```

Published with MATLAB® R2023b

```
function [theta1_ddot, theta2_ddot] = dynamic(theta1, theta1_dot, theta2, theta2_dot)

g = 9.81;
l1 = 1.0;
l2 = 1.0;
m1 = 1.0;
m2 = 1.0;

theta1_ddot = (-g * (2 * m1 + m2) * sin(theta1) - m2 * g * sin(theta1 - 2 * theta2) - ...
    2 * sin(theta1 - theta2) * m2 * (theta2_dot^2 * l2 + theta1_dot^2 * l1 * cos(theta1 - theta2))) / ...
    (l1 * (2 * m1 + m2 - m2 * cos(2 * theta1 - 2 * theta2)));

theta2_ddot = (2 * sin(theta1 - theta2) * (theta1_dot^2 * l1 * (m1 + m2) + g * (m1 + m2) * cos(theta1) + ...
    theta2_dot^2 * l2 * m2 * cos(theta1 - theta2))) / ...
    (l2 * (2 * m1 + m2 - m2 * cos(2 * theta1 - 2 * theta2)));

end
```

```
Not enough input arguments.

Error in dynamic (line 9)
theta1_ddot = (-g * (2 * m1 + m2) * sin(theta1) - m2 * g * sin(theta1 - 2 * theta2) - ...
```

Published with MATLAB® R2023b

Mehdi Muzaffari - MAE 674 - Project 1

```
clc;
clear;
close all;
```

load simulate data

```
data = readtable('simulation');
time = data.Time;
theta1_true = data.Theta1;
theta1_dot_true = data.Theta1_Dot;
theta2_true = data.Theta2;
theta2_dot_true = data.Theta2_Dot;

time = downsample(time, 20);
theta1_true = downsample(theta1_true, 20);
theta1_dot_true = downsample(theta1_dot_true, 20);
theta2_true = downsample(theta2_true, 20);
theta2_dot_true = downsample(theta2_dot_true, 20);
```

simulation parameter

```
num_steps = length(time);

x_hat = [pi/4; 0; pi/6; 0];
P = diag([0.01, 0.1, 0.01, 0.01]);
Q = diag([0.01, 0.01, 0.01, 0.01]);
```

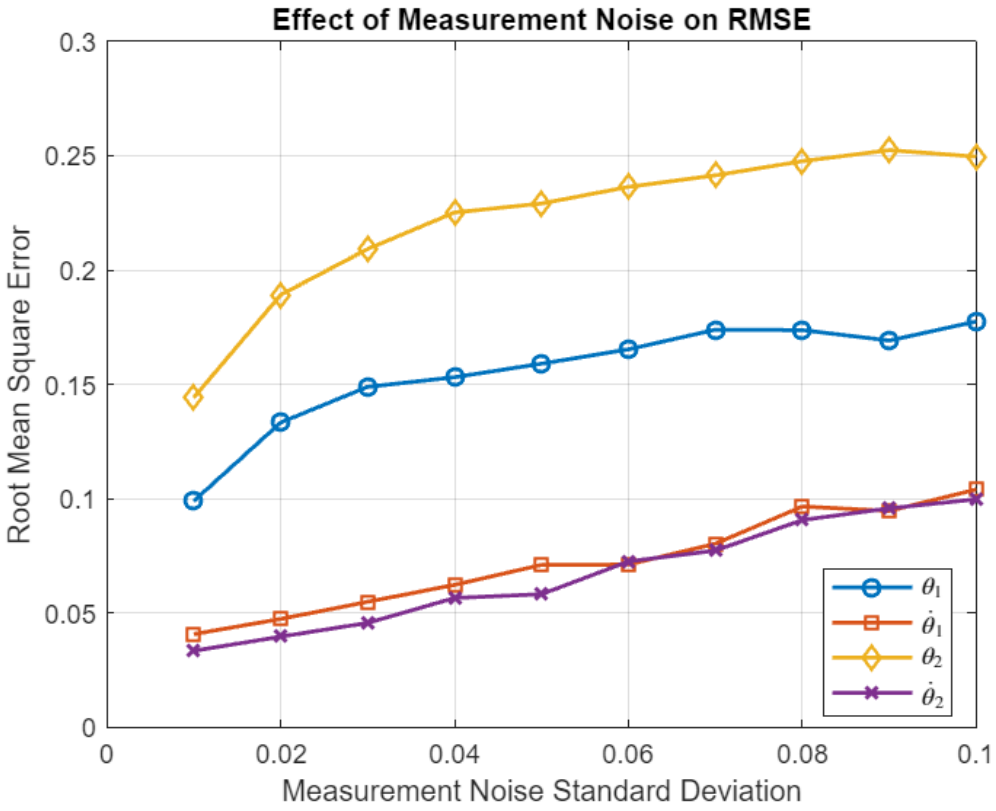
```
H = eye(4);

x_hat_history = zeros(4, num_steps);

noise_levels = 0.01:0.01:0.1;
rmse_results = zeros(length(noise_levels), 4);

for n = 1:length(noise_levels)

    noise_std_theta1 = noise_levels(n);
    noise_std_theta1_dot = noise_levels(n);
    noise_std_theta2 = noise_levels(n);
    noise_std_theta2_dot = noise_levels(n);

    theta1_measured = theta1_true + noise_std_theta1 * randn(size(theta1_true));
    theta1_dot_measured = theta1_dot_true + noise_std_theta1_dot *
randn(size(theta1_dot_true));
    theta2_measured = theta2_true + noise_std_theta2 * randn(size(theta2_true));
    theta2_dot_measured = theta2_dot_true + noise_std_theta2_dot *
randn(size(theta2_dot_true));
```

```
    R = diag([noise_std_theta1, noise_std_theta1_dot, noise_std_theta2,
noise_std_theta2_dot]);

    error_history = zeros(4, num_steps);

    for k = 2:num_steps

        z = [theta1_measured(k); theta1_dot_measured(k); theta2_measured(k);
theta2_dot_measured(k)];

        theta1 = x_hat(1);
        theta1_dot = x_hat(2);
        theta2 = x_hat(3);
        theta2_dot = x_hat(4);

        [theta1_ddot, theta2_ddot] = dynamic(theta1, theta1_dot, theta2,
theta2_dot);

        dt = 0.01;

        theta1_dot = theta1_dot + theta1_ddot * dt;
        theta2_dot = theta2_dot + theta2_ddot * dt;

        theta1 = theta1 + theta1_dot * dt;
        theta2 = theta2 + theta2_dot * dt;

        x_hat = [theta1; theta1_dot; theta2; theta2_dot];

        F = computejacobian(x_hat);
        P = F * P * F' + Q;

        z_hat = H * x_hat;
        y = z - z_hat;

        K = P * H' / (H * P * H' + R);
        x_hat = x_hat + K * y;
        P = (eye(size(P)) - K * H) * P;

        x_hat_history(:, k) = x_hat;
        error_history(:, k) = [theta1_true(k); theta1_dot_true(k); theta2_true(k);
theta2_dot_true(k)] - x_hat;

    end

    rmse_results(n, :) = sqrt(mean(error_history.^2, 2));

end
disp(array2table(rmse_results, ...
    'VariableNames', {'Theta1', 'Theta1_Dot', 'Theta2', 'Theta2_Dot'}, ...
```

```
    'RowNames', cellstr(num2str(noise_levels'))));
```

	Theta1	Theta1_Dot	Theta2	Theta2_Dot
0.01	0.098876	0.040668	0.14412	0.033352
0.02	0.13325	0.047302	0.18909	0.039577
0.03	0.14887	0.054808	0.20918	0.04558
0.04	0.15309	0.062224	0.22517	0.05649
0.05	0.15901	0.07097	0.22909	0.058135
0.06	0.16527	0.071107	0.23637	0.072472
0.07	0.17389	0.080238	0.24143	0.077244
0.08	0.1737	0.096526	0.24761	0.090596
0.09	0.16914	0.094585	0.25245	0.095725
0.1	0.17748	0.10397	0.24961	0.099683

```
figure;
plot(noise_levels, rmse_results(:, 1), '-o', 'DisplayName', '$\theta_1$',
'LineWidth', 1.5);
hold on;
plot(noise_levels, rmse_results(:, 2), '-s', 'DisplayName', '$\dot{\theta}_1$',
'LineWidth', 1.5);
plot(noise_levels, rmse_results(:, 3), '-d', 'DisplayName', '$\theta_2$',
'LineWidth', 1.5);
plot(noise_levels, rmse_results(:, 4), '-x', 'DisplayName', '$\dot{\theta}_2$',
'LineWidth', 1.5);
xlabel('Measurement Noise Standard Deviation');
ylabel('Root Mean Square Error');
title('Effect of Measurement Noise on RMSE');
legend('Location','best', 'Interpreter', 'Latex');
grid on;
hold off;
```


Effect of Measurement Noise on RMSE
Root Mean Square Error
Measurement Noise Standard Deviation
0.3
0.25
0.2
0.15
0.1
0.05
0
0
0.02
0.04
0.06
0.08
0.1
θ_1
$\dot{\theta}_1$
θ_2
$\dot{\theta}_2$

Mehdi Muzaffari - MAE 674 - Project 1

```
clc;
clear;
close all;
```

load simulated Data

```
data = readtable('simulation');
time = data.Time;
theta1_true = data.Theta1;
theta1_dot_true = data.Theta1_Dot;
theta2_true = data.Theta2;
theta2_dot_true = data.Theta2_Dot;
```

add noise to data

```
noise_std_theta1 = 0.05;
noise_std_theta1_dot = 0.03;
noise_std_theta2 = 0.04;
noise_std_theta2_dot = 0.02;

theta1_measured = theta1_true + noise_std_theta1 * randn(size(theta1_true));
theta1_dot_measured = theta1_dot_true + noise_std_theta1_dot *
randn(size(theta1_dot_true));
theta2_measured = theta2_true + noise_std_theta2 * randn(size(theta2_true));
theta2_dot_measured = theta2_dot_true + noise_std_theta2_dot *
randn(size(theta2_dot_true));
```

simulation parameters

```
dt = 0.01;
num_steps = length(time);

theta1 = pi/4;
theta1_dot = 0;
theta2 = pi/6;
theta2_dot = 0;

x_hat = [pi/4; 0; pi/6; 0];
P = diag([0.01, 0.1, 0.01, 0.01]);
Q = diag([0.01, 0.01, 0.01, 0.01]);
R = diag([0.05, 0.05, 0.05, 0.05]);
```

ukf parameters

```
alpha = 1e-3;
kappa = 0;
beta = 2;
```

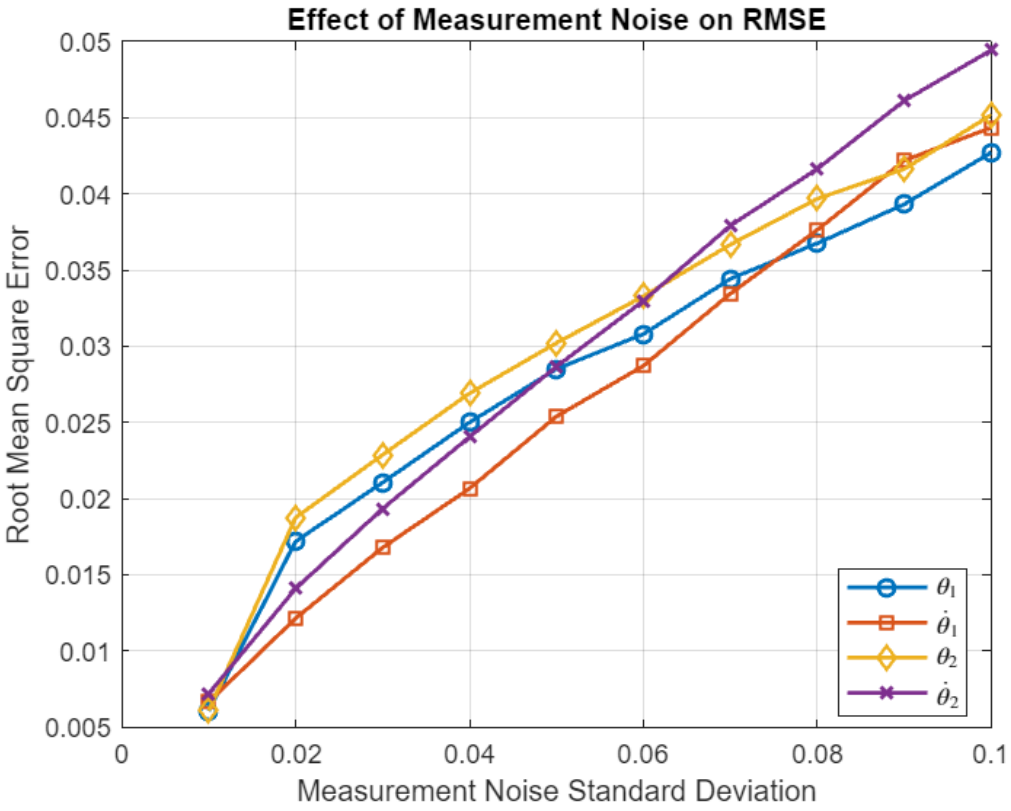
```
n = 4;
lambda = alpha.^2 * (n + kappa) - n;
gamma = sqrt(n + lambda);

Wm = [lambda / (n + lambda), repmat(0.5 / (n + lambda), 1, 2 * n)];
Wc = Wm;
Wc(1) = Wc(1) + (1 - alpha^2 + beta);
```

```
x_hat_history = zeros(n, num_steps);
P_hat_history = zeros(n, n, num_steps);

x_hat_history(:, 1) = x_hat;
P_hat_history(:, :, 1) = P;

noise_levels = 0.01:0.01:0.1;
rmse_results = zeros(length(noise_levels), 4);

for j = 1: length(noise_levels)

    noise_std_theta1 = noise_levels(j);
    noise_std_theta1_dot = noise_levels(j);
    noise_std_theta2 = noise_levels(j);
    noise_std_theta2_dot = noise_levels(j);

    theta1_measured = theta1_true + noise_std_theta1 * randn(size(theta1_true));
    theta1_dot_measured = theta1_dot_true + noise_std_theta1_dot *
randn(size(theta1_dot_true));
    theta2_measured = theta2_true + noise_std_theta2 * randn(size(theta2_true));
    theta2_dot_measured = theta2_dot_true + noise_std_theta2_dot *
randn(size(theta2_dot_true));

    R = diag([noise_std_theta1, noise_std_theta1_dot, noise_std_theta2,
noise_std_theta2_dot]);

    error_history = zeros(4, num_steps);

    for k = 2:num_steps

        X = [x_hat, x_hat + gamma * chol(P)', x_hat - gamma * chol(P)'];

        X_pred = zeros(size(X));

        for i = 1:size(X, 2)

            theta1 = X(1, i);
            theta1_dot = X(2, i);
            theta2 = X(3, i);
            theta2_dot = X(4, i);
```

```
            [theta1_ddot, theta2_ddot] = dynamic(theta1, theta1_dot, theta2,
theta2_dot);

            X_pred(:, i) = [theta1 + dt * theta1_dot; theta1_dot + dt * theta1_ddot;
                theta2 + dt * theta2_dot; theta2_dot + dt * theta2_ddot];
        end

        x_pred = X_pred * Wm';

        P_pred = Q;
        for i = 1:size(X, 2)
            P_pred = P_pred +  Wc(i) * (X_pred(:, i) - x_pred) * (X_pred(:, i) -
x_pred)';
        end

        X_update = [x_pred, x_pred + gamma * chol(P_pred)', x_pred - gamma *
chol(P_pred)'];

        Z_pred = X_update;
        z_pred = Z_pred * Wm';

        S = R;
        for i = 1:size(X_update, 2)
            S = S + Wc(i) * (Z_pred(:, i) - z_pred) * (Z_pred(:, i) - z_pred)';
        end

        C = zeros(n);
        for i = 1:size(X_update, 2)
            C = C + Wc(i) * (X_pred(:, i) - x_pred) * (Z_pred(:, i) - z_pred)';
        end

        K = C / S;
        z = [theta1_measured(k); theta1_dot_measured(k); theta2_measured(k);
theta2_dot_measured(k)];
        x_hat = x_pred + K * (z - z_pred);
        P = P_pred - K * S * K';

        x_hat_history(:, k) = x_hat;
        P_hat_history(:, :, k) = P;

        error_history(:, k) = [theta1_true(k); theta1_dot_true(k); theta2_true(k);
theta2_dot_true(k)] - x_hat;
    end

    rmse_results(j, :) = sqrt(mean(error_history.^2, 2));
end

disp(array2table(rmse_results, ...
    'VariableNames', {'Theta1', 'Theta1_Dot', 'Theta2', 'Theta2_Dot'}, ...
    'RowNames', cellstr(num2str(noise_levels'))));
```

	Theta1	Theta1_Dot	Theta2	Theta2_Dot
0.01	0.0060467	0.0066479	0.0060733	0.0071955
0.02	0.017187	0.01214	0.01875	0.01409
0.03	0.021051	0.016779	0.022871	0.019335
0.04	0.025	0.020662	0.026907	0.024045
0.05	0.028505	0.02539	0.030215	0.028663
0.06	0.030795	0.028713	0.033283	0.032979
0.07	0.034421	0.03345	0.036672	0.037916
0.08	0.036756	0.037634	0.039683	0.041641
0.09	0.03932	0.042186	0.041632	0.04612
0.1	0.042718	0.044349	0.045184	0.049427

```
figure;
plot(noise_levels, rmse_results(:, 1), '-o', 'DisplayName', '$\theta_1$',
'LineWidth', 1.5);
hold on;
plot(noise_levels, rmse_results(:, 2), '-s', 'DisplayName', '$\dot{\theta}_1$',
'LineWidth', 1.5);
plot(noise_levels, rmse_results(:, 3), '-d', 'DisplayName', '$\theta_2$',
'LineWidth', 1.5);
plot(noise_levels, rmse_results(:, 4), '-x', 'DisplayName', '$\dot{\theta}_2$',
'LineWidth', 1.5);
xlabel('Measurement Noise Standard Deviation');
ylabel('Root Mean Square Error');
title('Effect of Measurement Noise on RMSE');
legend('Location','best', 'Interpreter', 'Latex');
grid on;
hold off;
```


Effect of Measurement Noise on RMSE
Root Mean Square Error
Measurement Noise Standard Deviation
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0.02
0.04
0.06
0.08
0.1
θ_1
$\dot{\theta}_1$
θ_2
$\dot{\theta}_2$

```python
# Mehdi Muzaffari - MAE 674 Project 1

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

dt = 0.01
num_steps = 10000
time = np.arange(0, num_steps) * dt

theta1 = np.pi/2
theta1_dot = 1
theta2 = np.pi/6
theta2_dot = 1

g = 9.81
l1 = 1.0
l2 = 1.0
m1 = 1.0
m2 = 1.0

def dynamics(theta1, theta1_dot, theta2, theta2_dot):

    theta1_ddot = (
    -g * (2 * m1 + m2) * np.sin(theta1)
    - m2 * g * np.sin(theta1 - 2 * theta2)
    - 2 * np.sin(theta1 - theta2) * m2 * (
    theta2_dot**2 * l2 + theta1_dot**2 * l1 * np.cos(theta1 - theta2))) / (l1 * (2 * m1 + m2 - m2 * np.cos(2 * theta1 - 2 * theta2)))

    theta2_ddot = (
    2 * np.sin(theta1 - theta2) * (
    theta1_dot**2 * l1 * (m1 + m2)
    + g * (m1 + m2) * np.cos(theta1)
    + theta2_dot**2 * l2 * m2 * np.cos(theta1 - theta2))) / (l2 * (2 * m1 + m2 - m2 * np.cos(2 * theta1 - 2 * theta2)))

    return theta1_ddot, theta2_ddot


theta1_history = np.zeros(num_steps)
theta2_history = np.zeros(num_steps)


theta1_history[0] = theta1
theta2_history[0] = theta2


for k in range(1, num_steps):

    theta1_ddot, theta2_ddot = dynamics(theta1, theta1_dot, theta2, theta2_dot)

    theta1_dot = theta1_dot + theta1_ddot * dt
    theta1 = theta1 + theta1_dot * dt

    theta2_dot = theta2_dot + theta2_ddot * dt
    theta2 = theta2 + theta2_dot * dt

    theta1_history[k] = theta1
    theta2_history[k] = theta2


x1 = l1 * np.sin(theta1_history)
y1 = -l1 * np.cos(theta1_history)
x2 = x1 + l2 * np.sin(theta2_history)
y2 = y1 - l2 * np.cos(theta2_history)


fig, ax = plt.subplots()
ax.set_xlim(-l1 - l2 - 0.5, l1 + l2 + 0.5)
ax.set_ylim(-l1 - l2 - 0.5, l1 + l2 + 0.5)
ax.set_aspect('equal')


line, = ax.plot([], [], '-o', lw=2, markersize=8, label="Pendulums")
trail1, = ax.plot([], [], 'blue', lw=0.5, label="Pendulum 1 Trajectory")
trail2, = ax.plot([], [], 'red', lw=0.5, label="Pendulum 2 Trajectory")


def init():
    line.set_data([], [])
    trail1.set_data([], [])
    trail2.set_data([], [])
    return line, trail1, trail2


def update(frame):
    current_x = [0, x1[frame], x2[frame]]
    current_y = [0, y1[frame], y2[frame]]

    line.set_data(current_x, current_y)

    trail1.set_data(x1[:frame], y1[:frame])
    trail2.set_data(x2[:frame], y2[:frame])

    return line, trail1, trail2

ani = animation.FuncAnimation(fig, update, frames=num_steps, init_func=init, interval=dt * 1000, blit=True)

plt.legend()
plt.show()


```