Robot Learning Lab 1 - EKF

Borella Simone (S317774)
Computer engineering - Artificial Intelligence
Polytechinc University of Turin, Italy
s317774@studenti.polito.it

1 Abstarct

The primary objective of this initial laboratory is to cultivate familiarity with the Kalman Filter, a fundamental tool for estimating the state of a stochastic system in the presence of noisy observations. Our focus will be on the application of the Extended Kalman Filter (EKF) to estimate the state of a non-linear system, specifically the unactuated simple pendulum system and the unactuated double pendulum system.

2 Extended Kalman Filter (EKF)

The Extended Kalman Filter (EKF) surpasses the limitation of the simple Kalman Filter, which is only applicable to linear systems. It achieves this by linearizing the system at each estimation step.

2.1 Bayesian Filter Framework

The Kalman Filter operates within the Bayesian filter framework, providing a recursive methodology for state estimation in a system. It relies on a prior belief, typically a prediction regarding the new state of the system based on a linearized model. This belief is then updated upon the arrival of a new sensor measurement.

2.2 Gaussian Assumptions

The Kalman Filter assumes both the prediction and the measurement to follow Gaussian distributions. This assumption facilitates the update step, involving a straightforward multiplication of the two distributions. Consequently, the result remains a Gaussian distribution, with a variance lower than both the prediction and the measurement. This characteristic enhances the certainty of the updated state estimate.

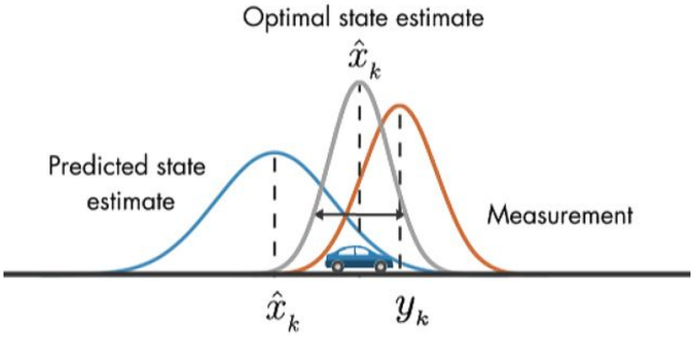

Figure 1: Kalman Filter process

3 The unactuated pendulum system

3.1 System State Space and Model

The considered system is a simple pendulum system, freely oscillating under given initial conditions. In this unactuated system, affected solely by gravity, the state is described by two variables: the angle (θ) and angular velocity ($\dot{\theta}$). Parameters such as the length of the pendulum (L) and mass (m) are known.

The system model is given by:

$$\begin{aligned} x &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}, \\ \dot{x} &= \begin{bmatrix} x_2 \\ -\frac{g}{L}\sin(x_1) \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ -\frac{g}{L}\sin(\theta) \end{bmatrix} \end{aligned} \tag{1}$$

respresenting the state-space representation as a system of two first-order differential equations.

The forward Euler method is employed to discretize continuous dynamics:

$$x_t = f(x_{t-1}) = x_{t-1} + \Delta t \frac{dx}{dt} \tag{2}$$

resulting in:

$$x_t = f(x_{t-1}) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} \tag{3}$$

with f_1 and f_2 defined as:

$$\begin{aligned} f_1(x_1, x_2) &= x_1 + \Delta t x_2, \\ f_2(x_1, x_2) &= x_2 - \Delta t \frac{g}{L}\sin(x_1) \end{aligned} \tag{4}$$

The Jacobian matrix for the linearization of f is given by:

$$A_t = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ -\Delta t \frac{g}{L}\cos(x_1) & 1 \end{bmatrix} \tag{5}$$

3.2 Observation Model

Observations from virtual sensors measure only the pendulum's angle. The observation model is defined as:

$$z(x) = x_1 + d \tag{6}$$

where d is a gaussian disturbance.

The Jacobian matrix for the linearization of z is:

$$C_t = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} \end{bmatrix} = [1 \quad 0] \tag{7}$$

4 EKF implementation

The implementation of the EKF algorithm follows the structure of the Algorithm 1.

4.1 Prediction step

The **state prediction** ($\overline{x}_t$) is calculated by applying the system's dynamics model to the current state estimate. First the system undergoes discretization using the forward Euler method to obtain A_t. Subsequently $\overline{x}_t$ is computed by applying the discretized system state-space equation, taking into account the last state (x_{t-1}).

The **estimate state uncertainty prediction** (Σ_t) is computed by incorporating the uncertainty from the previous time step (Σ_{t-1}) and the process noise covariance matrix (Q_t) which represents the covariance matrix associated with the process noise, which is typically assumed to be a source of uncertainty in the system.

In our case we assume the system to be very accurate so covariances values result to be very small:

$$Q = \begin{bmatrix} 0.00001 & 0 \\ 0 & 0.00001 \end{bmatrix}$$

4.2 Correction step

The **Kalman gain** (K_t) is computed taking into account the covariance of the measurement noise (R_t). The Kalman gain determines how much the predicted state ($\overline{x}_t$) should be adjusted based on the difference between the predicted measurement and the actual measurement.

In our case the value of R covariance values are consistent with the sensor simulated disturbance:

$$R = [0.05]$$

The **state update** (x_t) is computed considering the Kalman gain (K_t) and the difference between the actual measurement (z_t) and the predicted measurement ($C_t\overline{x}_t$).

The **covariance of the state estimate update** (Σ_t) is performed to reflect the reduced uncertainty after incorporating the new measurement.

Algorithm 1: Extended Kalman Filter

```
Data: Initial state estimate $x_0$, initial state covariance $\Sigma_0$, process model $f$, measurement
      model $h$, measurement covariance $R$, process noise covariance matrix $Q_t$
Result: Estimated state $x_t$ and its covariance $\Sigma_t$ for each time step $t$
for $t = 1$ to $N$ do
    Prediction:
        $\overline{x}_t = f(\overline{x}_{t-1}) = A_t x_{t-1} + B_t u_t$   (Predicted state estimate)
        $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t$   (Predicted covariance)
    Correction:
        $K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + R_t)^{-1}$   (Kalman gain)
        $x_t = \overline{x}_t + K_t(z_t - C_t \overline{x}_t)$   (Updated state estimate)
        $\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t$   (Updated covariance)
```

5 ROS implementation

ROS is a framework that facilitates the concurrent execution of distinct processes, referred to as nodes. These nodes communicate by exchanging messages and information through the publish-subscribe and service-action paradigms.

The ROS system in the pendulum case comprises three nodes:

1. **Pendulum Node:** Responsible for simulating the pendulum motion in discrete time and publishing the real state of the system.
2. **Sensor Node:** Simulates a sensor extracting the virtually measured dimension (θ) from the real state. It adds a Gaussian disturbance to simulate noise.
3. **EKF Node:** Computes the state estimation of the system.
4. **Visualization Node:** translate the state dimensions as Markers to be visualized with RViz visualization tool.

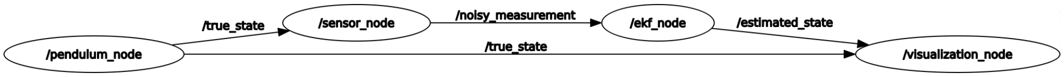

Figure 2: nodes and topics

6 The unactuated double pendulum system

This expansion to a four-variable state space necessitates an augmentation of the Extended Kalman Filter (EKF) to accommodate the additional dynamics and dependencies inherent in the double pendulum system.

6.1 System State Space and Model

The state of this system is comprehensively described by four variables: the angle of the first sphere (θ_1) and its angular velocity (ω_1), along with the angle of the second sphere (θ_2) and its corresponding angular velocity (ω_2). The system's parameters, including the lengths of the first and second pendulum (L_1 and L_2), as well as the masses of the first and second spheres (m_1 and m_2), are assumed to be known.

The system model is given by:

$$x = [\theta_1 \quad \omega_1 \quad \theta_2 \quad \omega_2]^T \tag{8}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\omega}_1 \\ \dot{\theta}_2 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \frac{-g(2m_1+m_2)\sin\theta_1 - m_2 g \sin(\theta_1 - 2\theta_2) - 2\sin(\theta_1-\theta_2) m_2 (\omega_2^2 L_2 + \omega_1^2 L_1 \cos(\theta_1-\theta_2))}{L_1(2m_1+m_2-m_2\cos(2(\theta_1-\theta_2)))} \\ \omega_2 \\ \frac{2\sin(\theta_1-\theta_2)(\omega_1^2 L_1 (m_1+m_2) + g(m_1+m_2)\cos\theta_1 + \omega_2^2 L_2 m_2 \cos(\theta_1-\theta_2))}{L_2(2m_1+m_2-m_2\cos(2(\theta_1-\theta_2)))} \end{bmatrix} \tag{9}$$

The jacobian matrix is given by:

$$A_t = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ \Delta t \frac{\partial \dot{x}_2}{\partial x_1} & 1 + \Delta t \frac{\partial \dot{x}_2}{\partial x_2} & \Delta t \frac{\partial \dot{x}_2}{\partial x_3} & \Delta t \frac{\partial \dot{x}_2}{\partial x_4} \\ 0 & 0 & 1 & \Delta t \\ \Delta t \frac{\partial \dot{x}_4}{\partial x_1} & \Delta t \frac{\partial \dot{x}_4}{\partial x_2} & \Delta t \frac{\partial \dot{x}_4}{\partial x_3} & 1 + \Delta t \frac{\partial \dot{x}_4}{\partial x_4} \end{bmatrix} \tag{10}$$

6.2 Observation Model

The observation model is defined as:

$$z(x) = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{11}$$

where d_1 and d_2 are gaussian disturbances.

The Jacobian matrix for the linearization of z is:

$$C_t = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \frac{\partial z_1}{\partial x_3} & \frac{\partial z_1}{\partial x_4} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} & \frac{\partial z_2}{\partial x_3} & \frac{\partial z_2}{\partial x_4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{12}$$

7 Plots and visualization

7.1 Pendulum system

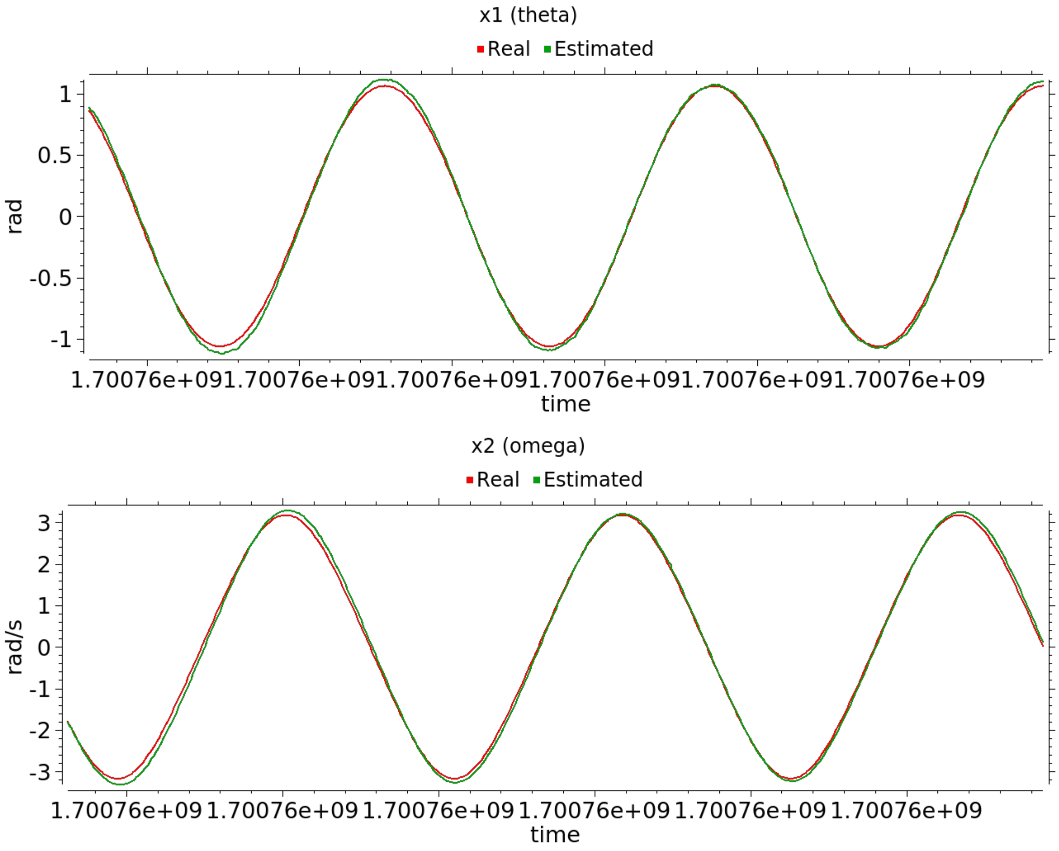

Figure 3: Pendulum state estimation

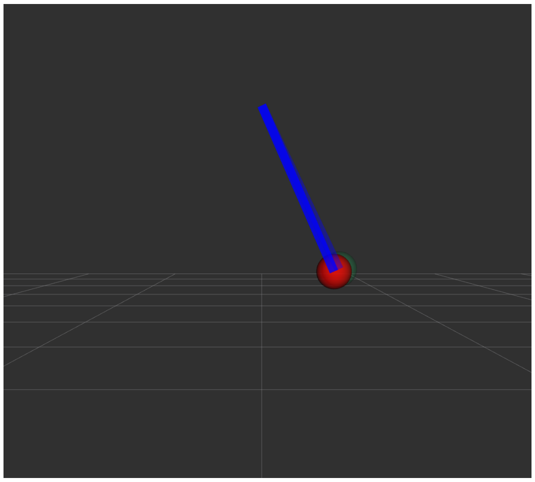

Figure 4: Pendulum visualization

7.2 Double pendulum system

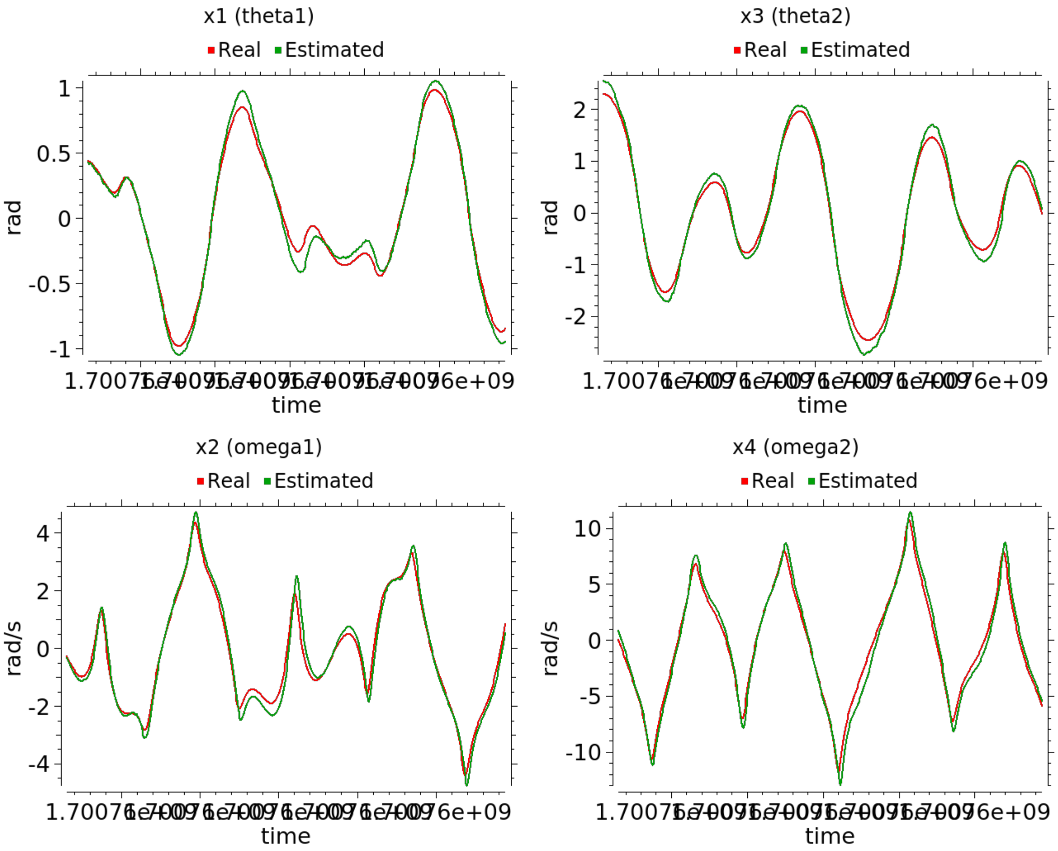

Figure 5: Double pendulum state estimation

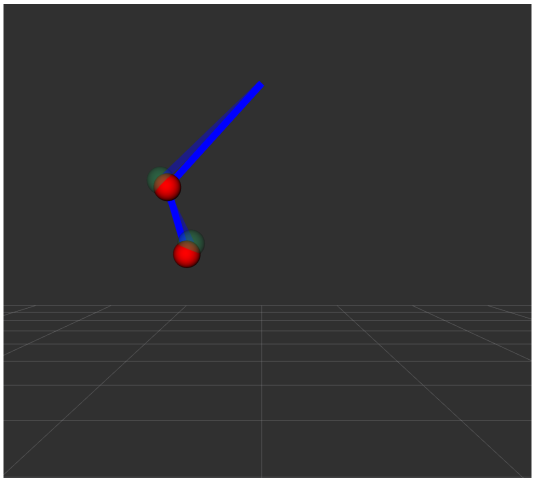

Figure 6: Double pendulum visualization

8 Results

The Extended Kalman Filter (EKF) was successfully applied to estimate the state of the non-linear system, specifically the unactuated simple pendulum. The algorithm demonstrated robust performance in tracking the state variables, namely the angle (θ) and angular velocity ($\dot{\theta}$), even in the presence of noisy sensor measurements.

8.1 Estimation Accuracy

The EKF exhibited high accuracy in estimating the state variables compared to ground truth values. Figure 7 illustrates the comparison between the estimated and true states over time. The algorithm effectively reduced the impact of sensor noise and provided a smooth and accurate estimation of the pendulum's dynamics (Figure 7).

8.2 Convergence Behavior

The convergence behavior of the Kalman Filter was analyzed under various conditions, including different levels of process and measurement noise. The performance of the estimation was examined under varying configurations of the measurement covariance matrix (R). In practical terms, the behavior is discernible when different magnitudes of R are employed.

For larger values of R (indicating greater uncertainty in the measurements), the algorithm places more reliance on the internal model. As a consequence, a convergence to the true state is not observed, particularly given the initialization from a random belief (Figure 8).

Conversely, for lower values of R (indicating higher confidence in the measurements), the algorithm rapidly converges to the ground truth curve. However, in this scenario, the algorithm tends to closely track the measurements, resulting in a more pronounced sensitivity to noise and, consequently, a comparatively noisier output (Figure 9).

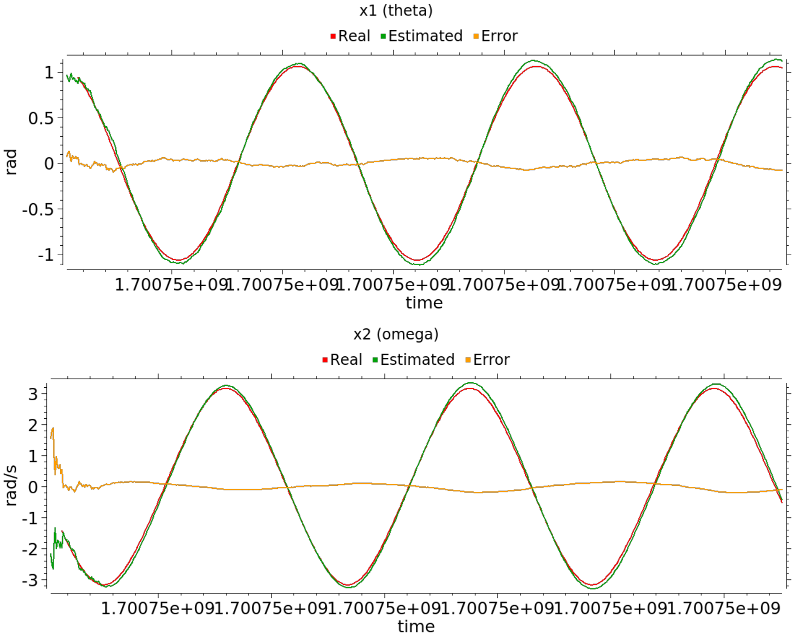

Figure 7: state and error plot, R = [0.05]

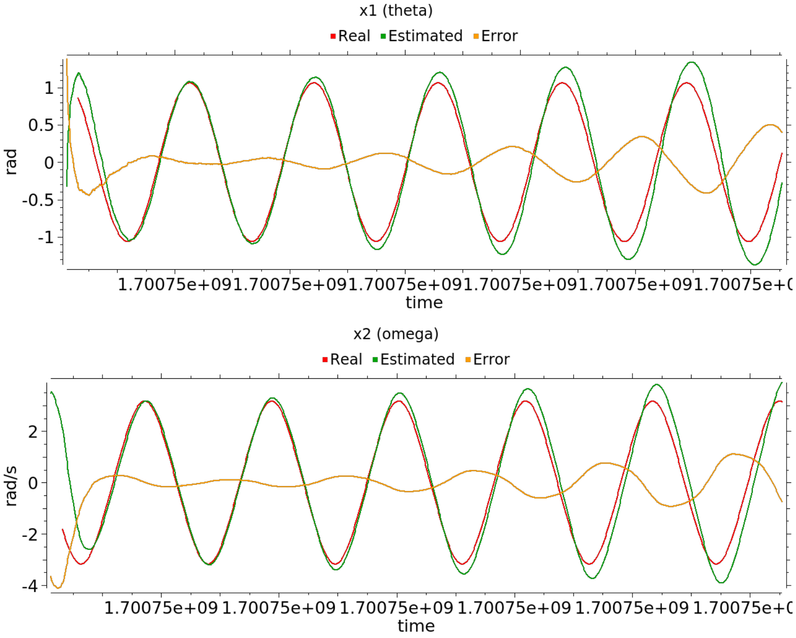

Figure 8: state and error plot, R = [5]

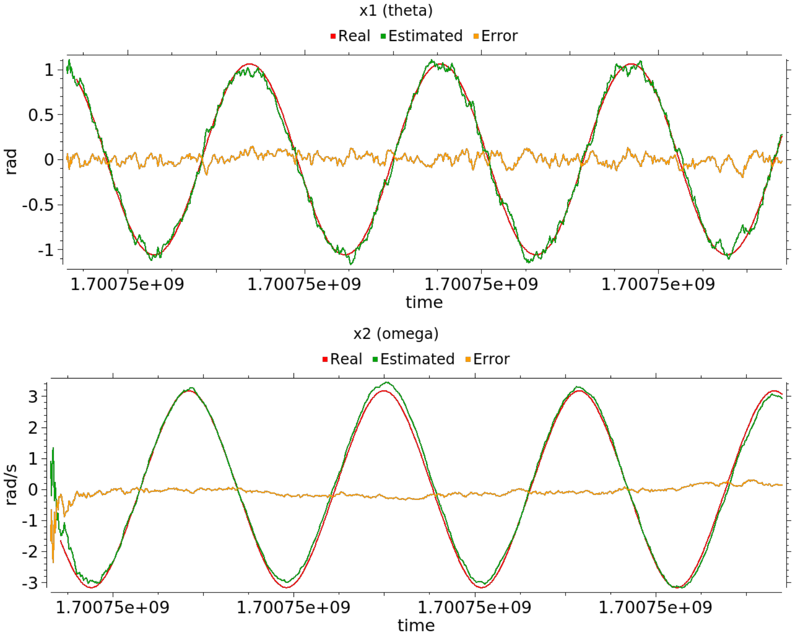

Figure 9: state and error plot, R = [0.0005]