Generate True Trajectory

September 2024

Continuous-time to Discrete-time I

Equations of Motion (EOM)

These are the continuous-time equations that describe the motion of a projectile under gravity and quadratic drag:

$$\frac{dx}{dt} = v\cos\theta$$

$$\frac{dz}{dt} = v\sin\theta$$

$$\frac{dv}{dt} = -g\sin\theta - gkv^2$$

$$\frac{d\theta}{dt} = -\frac{g\cos\theta}{v}$$

$$\frac{dk}{dt} = 0$$

Continuous-time to Discrete-time II

State Variables

To represent the system in state-space form, we define the following state variables:

$$
\begin{aligned}
x_1 &= x \\
x_2 &= z \\
x_3 &= v \\
x_4 &= \theta \\
x_5 &= k
\end{aligned}
$$

Continuous-time to Discrete-time III

State Equations

We rearrange the EOM into state-space form:

$$
\begin{aligned}
\frac{dx_1}{dt} &= x_3 \cos(x_4) \\
\frac{dx_2}{dt} &= x_3 \sin(x_4) \\
\frac{dx_3}{dt} &= -g \sin(x_4) - g x_5 x_3^2 \\
\frac{dx_4}{dt} &= -\frac{g \cos(x_4)}{x_3} \\
\frac{dx_5}{dt} &= 0
\end{aligned}
$$

Continuous-time to Discrete-time IV

State Space Equation (Continuous-time)

The continuous-time state-space representation can be written as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$$

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_3 \cos(x_4) \\ x_3 \sin(x_4) \\ -g \sin(x_4) - g x_5 x_3^2 \\ -\frac{g \cos(x_4)}{x_3} \\ 0 \end{bmatrix}$$

where the state vector $\mathbf{x}$ is:

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^T$$

Continuous-time to Discrete-time V

State Space Equation (Discrete-time)
To generate a trajectory in discrete-time, we discretize the continuous time equations using a suitable numerical method. The discrete time state-space equation is represented as:

$$\mathbf{x_k} = \mathbf{f_d}(\mathbf{x}_{k-1}) = \mathbf{Numerical\_method}(\mathbf{f}(\mathbf{x_{k-1}}), \Delta t)$$

The Fourth Order-Runge Kutta Method (RK4)
We use the Fourth Order Runge-Kutta (RK4) method because it handles nonlinear systems effectively and offers a good balance

Continuous-time to Discrete-time VI

between accuracy and computational cost. The steps are as follows:

$$
\begin{aligned}
\mathbf{k_1} &= \mathbf{f}(\mathbf{x_{k-1}}) \\
\mathbf{k_2} &= \mathbf{f}(\mathbf{x_{k-1}} + 0.5 \cdot \Delta\mathbf{t} \cdot \mathbf{k_1}) \\
\mathbf{k_3} &= \mathbf{f}(\mathbf{x_{k-1}} + 0.5 \cdot \Delta\mathbf{t} \cdot \mathbf{k_2}) \\
\mathbf{k_4} &= \mathbf{f}(\mathbf{x_{k-1}} + \Delta\mathbf{t} \cdot \mathbf{k_3}) \\
\mathbf{k} &= \frac{1}{6}(\mathbf{k_1} + 2 \cdot \mathbf{k_2} + 2 \cdot \mathbf{k_3} + \mathbf{k_4}) \\
\mathbf{x_k} &= \mathbf{x_{k-1}} + \mathbf{k} \cdot \Delta\mathbf{t}
\end{aligned}
$$

Finally, we can generate the trajectory from the state-space equation using the RK4 method.

State and Output Equations

State Equations (Prediction Step)

- This equation predicts the state at time step k based on the previous state $\mathbf{x}_{k-1}$, assuming no process noise (i.e., $\mathbf{w}_{k-1} = 0$).

$$\mathbf{x_k} = \mathbf{f_d}(\mathbf{x_{k-1}})$$

Output Equations (Update Step)

- The output $\mathbf{y}_k$ is measured at time step k, with measurement noise $\mathbf{v}_k$ normally distributed as $\mathcal{N}(0, \mathbf{R}_k)$.

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$$

MATLAB Code: Data_Quadratic_drag I

```
clc
clear
close all
% Parameters
% Gravity constant (m/s^2) and drag coefficient
g = 9.81;
m = 42;
rho = 1.225;
S = 0.699223^2*pi;
C_D = 0.24;
k = (rho*C_D*S)/(2*m*g);
disp('k:');
disp(k);
Q = diag([0 0 0 0 0]);  % Initial process noise covariance matrix
R = diag(1e-3*ones(1,2));  % Measurement noise covariance matrix
x0 = [0; 0; 50; 50*pi/180; k];  % Initial state (x0, z0, v0, theta0, k)
P0 = eye(5);  % Initial covariance matrix (adjusted to match the state vector dimension)
```

MATLAB Code: Data_Quadratic_drag II

```
t_end = 100;
delta_t = 1e-3;  % Time step (adjusted to simulate continuous dynamics)
num_steps = t_end/delta_t;  % Number of time steps
rng(1);  % Random seed

% Define the state transition function
f = @(t, x) [
     x(3) * cos(x(4));                    % dx/dt = v * cos(theta)
     x(3) * sin(x(4));                    % dz/dt = v * sin(theta)
    -g * sin(x(4)) - g * x(5) * x(3)^2;   % dv/dt = -g * sin(theta) - k * v^2
    -g * cos(x(4)) / x(3);               % dtheta/dt = -g * cos(theta) / v
     0                                    % dk/dt = 0 (assumed constant for simplicity)
];

% Define the measurement function
```

MATLAB Code: Data_Quadratic_drag III

```
h = @(x) [
    x(1);  % Measured x position
    x(2);  % Measured z position
];

% Time vector
t = 0:delta_t:100;

% Initialize state and observation arrays
x_true = zeros(5, num_steps);
z = zeros(2, num_steps);
x_true(:, 1) = x0;  % Set initial state
z(:, 1) = x0(1:2,1);  % Set initial state

% Generate true states and observations
for i = 2:num_steps
    % RK4 integration step to compute next state
    x_true(:, i) = RK4(f, t(i-1), x_true(:, i-1), delta_t)
    ;

    % Add process noise
```

MATLAB Code: Data_Quadratic_drag IV

```
%
    % Generate observation with measurement noise
    z(:, i) = h(x_true(:, i)) + sqrtm(R) * randn(2, 1);

    % Check if the projectile hit the ground
    if x_true(2, i) < 0
        x_true = x_true(:, 1:i);  % Truncate the state array
        z = z(:, 1:i);            % Truncate the observation array
        break;
    end
end
% Save data to file
save('ukf_Estimate_Quadraticairdrag.mat', 'x_true', 'z');

% % Plot the trajectory
% figure;
% plot(x_true(1,:), x_true(2,:), '-', 'LineWidth', 2, 'Color', 'b', 'DisplayName', 'True State');
% hold on;
```

MATLAB Code: Data_Quadratic_drag V

```
%
% plot(z(1,:), z(2,:), '-', 'LineWidth', 2, 'Color', '
    magenta', 'DisplayName', 'Observations');
% xlabel('X Position');
% ylabel('Y Position');
% legend;
% title('True State vs Estimated State and Observations');
```