```
clear variables;
close all;
clc;

addpath('..\Functions\');
load('mmExampleModeSwitchingData.mat')
rng(55);
lower_bound = 0.01;

% Model Parameters
c1 = 0.1; c2 = 0.5; c3 = 1;
hz1 = 1; hz2 = 2; hz3 = 3;

a1 = c1/hz1; b1 = 1/hz1;
a2 = c2/hz2; b2 = 1/hz1;
a3 = c3/hz3; b3 = 1/hz3;

% System Matrices
A1 = [0, 1; 0, -a1];
A2 = [0, 1; 0, -a2];
A3 = [0, 1; 0, -a3];
B1 = [0; b1];
B2 = [0; b2];
B3 = [0; b3];

Gamma = eye(2);
H = [1, 0];
R = 0.1;
Q = 0.001*diag([0.1, 1]);
dt = 0.1;

Fk1 = expm(A1*dt);
[~, y] = ode45(@(t, y) reshape(expm(A1*(dt - t)), 4, 1), [0, dt], zeros(4,
1));
Gk1 = reshape(y(end, :), 2, 2)*B1;
Fk2 = expm(A2*dt);
[~, y] = ode45(@(t, y) reshape(expm(A2*(dt - t)), 4, 1), [0, dt], zeros(4,
1));
Gk2 = reshape(y(end, :), 2, 2)*B2;
Fk3 = expm(A3*dt);
[~, y] = ode45(@(t, y) reshape(expm(A3*(dt - t)), 4, 1), [0, dt], zeros(4,
1));
Gk3 = reshape(y(end, :), 2, 2)*B3;

% Stack the multiple models
Fk = cat(3, Fk1, Fk2, Fk3);
Gk = cat(3, Gk1, Gk2, Gk3);
Gammak = cat(3, Gamma, Gamma, Gamma);
Hk = cat(3, H, H, H);
Rk = cat(3, R, R, R);
Qk = cat(3, Q, Q, Q);
```

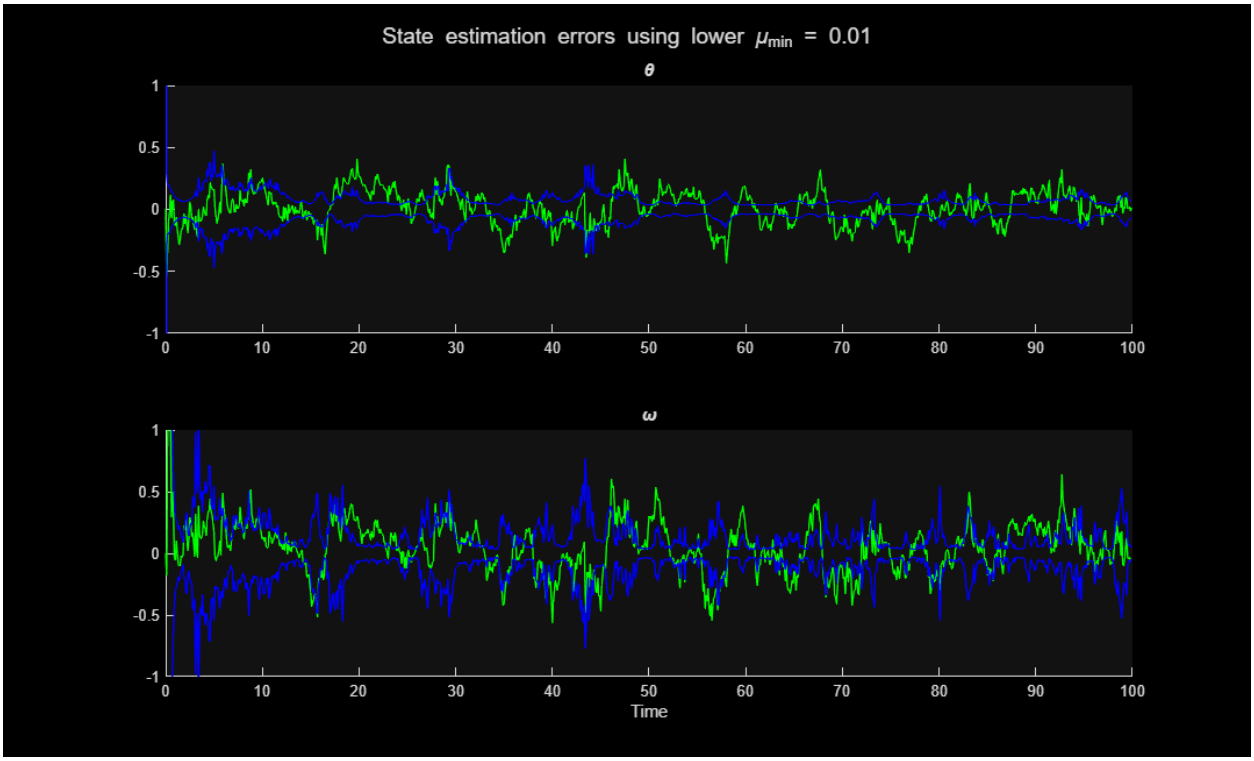
```
% Inputs
u = utruekhist;

% Initial Conditions
xbar0 = [0;0.1];
P0 = 10*eye(2);

% Measurements
z = ztruekhist(2:end);

% Run the MM filter
[t_mm, x_mm, P_mm, mu_mm] = static_mm_filter(Fk, Gk, Gammak, Hk, Qk, Rk, u,
z, xbar0, P0, 'LowerBound', lower_bound);

% Calculate consistency
nees = 0;
nx = length(xbar0);
l = length(tkhist);
for j=1:l
    nees = nees + (squeeze(x_mm(:, end, j)) -
xtruekhist(j, :).').'*inv(P_mm(:, :, j))*(squeeze(x_mm(:, end, j)) -
xtruekhist(j, :).');
end
snees = nees/(nx*l)

% Tuning the lower bound
window = [0.0001, 0.1];
num_points = 10;
points = linspace(window(1), window(2), num_points);
points = [0.0001, 0.001, 0.01, 0.1];
cons_results = zeros(size(points));
rms_results = zeros(size(points));
for r=1:length(points)
    [~, x_mm, P_mm, ~] = static_mm_filter(Fk, Gk, Gammak, Hk, Qk, Rk, u, z,
xbar0, P0, 'LowerBound', points(r));
    % Calculate consistency
    nees = 0;
    nx = length(xbar0);
    l = length(tkhist);
    for j=1:l
        nees = nees + (squeeze(x_mm(:, end, j)) -
xtruekhist(j, :).').'*inv(P_mm(:, :, j))*(squeeze(x_mm(:, end, j)) -
xtruekhist(j, :).');
    end
    cons_results(r) = nees/(nx*l);
    rms_results(r) = trace(sum(P_mm(:, :, end, :), 4));
    % Calculate RMS
end

% Fit a parabola and minimize
p_cons = polyfit(points, cons_results, 2);
a_cons = p_cons(1);
b_cons = p_cons(2);
mu_min_cons = -b_cons / (2 * a_cons)
```

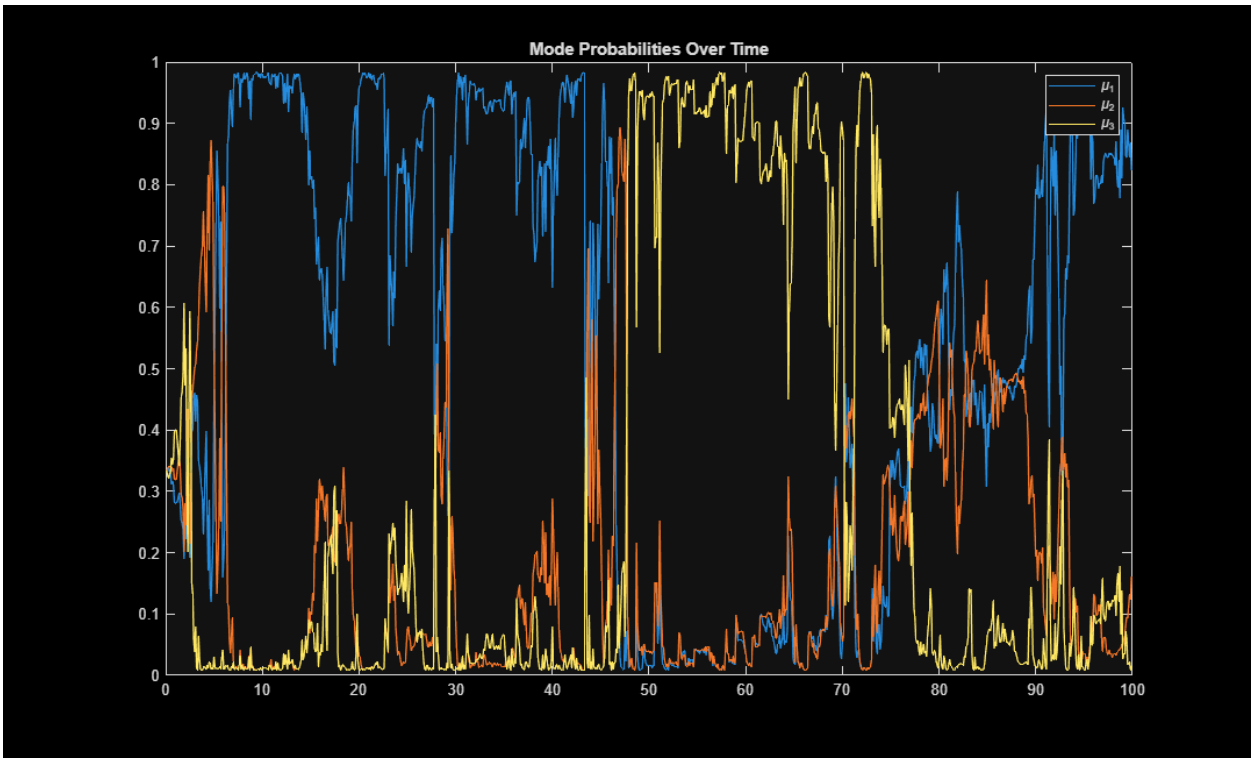
```
p_rms = polyfit(points, rms_results, 2);
a_rms = p_rms(1);
b_rms = p_rms(2);
mu_min_rms = -b_rms / (2 * a_rms)
```

```
snees =

   1.450348999478122


mu_min_cons =

   0.040314680717746


mu_min_rms =

   0.074390463808196
```

Plotting

```
figure;
sgtitle(['State estimation errors using lower \mu_{min} = ',
num2str(lower_bound)]);
subplot(2, 1, 1)
hold on;
plot(tkhist, squeeze(x_mm(1, end, :)) - xtruekhist(:, 1), 'g');
plot(tkhist, 3*squeeze(P_mm(1, 1, end, :)), 'b');
plot(tkhist, -3*squeeze(P_mm(1, 1, end, :)), 'b');
ylim([-1, 1]);
title('\theta');

subplot(2, 1, 2)
hold on;
plot(tkhist, squeeze(x_mm(2, end, :)) - xtruekhist(:, 2), 'g');
plot(tkhist, 3*squeeze(P_mm(2, 2, end, :)), 'b');
plot(tkhist, -3*squeeze(P_mm(2, 2, end, :)), 'b');
ylim([-1, 1]);
title('\omega');
xlabel('Time')

figure;
plot(tkhist, mu_mm);
legend('\mu_1', '\mu_2', '\mu_3');
title('Mode Probabilities Over Time')
```





Published with MATLAB® R2025a