Thông tin về hệ thống

The system equation is given by

$$\begin{aligned}\dot{x}(t) &= A(t)x(t) + w(t)\\ z(t) &= f(x(t)) + v(t)\end{aligned}$$

The state $x(t)$ and the measurement $z(t)$ are defined as follows:

$$x(t) \triangleq \begin{bmatrix}\theta\\ \phi\\ \omega_x\\ \omega_y\\ \omega_z\end{bmatrix}, \quad z(t) \triangleq \begin{bmatrix}a_x\\ a_y\\ g_x\\ g_y\\ g_z\end{bmatrix}.$$

where

$$A(t) \triangleq \begin{bmatrix}0 & 0 & 0 & \cos\phi(t) & -\sin\phi(t)\\ 0 & 0 & 1 & \sin\phi(t)\tan\theta(t) & \cos\phi(t)\tan\theta(t)\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}, \quad f(x(t)) \triangleq \begin{bmatrix}-\sin\theta\\ \sin\phi\cos\theta\\ \omega_x\\ \omega_y\\ \omega_z\end{bmatrix}.$$

$$\theta = \sin^{-1}(a_x) \quad \text{and} \quad \phi = \sin^{-1}\left(\frac{a_y}{\cos\theta}\right)$$

Process and measurement noise $w(t)$ and $v(t)$ are assumed to be uncorrelated zero-mean white Gaussian processes satisfying

$$Q = \begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & q_1 & 0 & 0\\ 0 & 0 & 0 & q_1 & 0\\ 0 & 0 & 0 & 0 & q_1\end{bmatrix} = \mathrm{E}\{w(t)w(t)'\}$$

$$R = \begin{bmatrix}r_1 & 0 & 0 & 0 & 0\\ 0 & r_2 & 0 & 0 & 0\\ 0 & 0 & r_3 & 0 & 0\\ 0 & 0 & 0 & r_3 & 0\\ 0 & 0 & 0 & 0 & r_3\end{bmatrix} = \mathrm{E}\{v(t)v(t)'\}.$$

- **Thực hiện rời rạc hóa các biến trạng thái (không xét nhiễu)**

  $x(k) = \left[\theta(k) \quad \phi(k) \quad \omega_x(k) \quad \omega_y(k) \quad \omega_z(k)\right]^T$

$$x(k+1)=\left(T_sA+1\right).x\left(k\right)=g\left(x\left(k\right)\right)$$

$$T_sA+1=\begin{bmatrix} 1 & 0 & 0 & T_s\cos\left(\phi\left(k\right)\right) & -T_s\sin\left(\phi\left(k\right)\right) \\ 0 & 1 & T_s & T_s\sin\left(\phi\left(k\right)\right)\tan\left(\theta\left(k\right)\right) & T_s\cos\left(\phi\left(k\right)\right)\tan\left(\theta\left(k\right)\right) \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$g\left(x\left(k\right)\right)=\begin{bmatrix} \theta-T_s\omega_z\left(k\right)\sin\left(\phi\left(k\right)\right)+T_s\omega_y\left(k\right)\cos\left(\phi\left(k\right)\right) \\ \phi\left(k\right)+T_s\omega_x\left(k\right)+T_s\omega_z\left(k\right)\cos\left(\phi\left(k\right)\right)\tan\left(\theta\left(k\right)\right)+T_s\omega_y\left(k\right)\sin\left(\phi\left(k\right)\right)\tan\left(\theta\left(k\right)\right) \\ \omega_x\left(k\right) \\ \omega_y\left(k\right) \\ \omega_z\left(k\right) \end{bmatrix}$$

Ma trận Jacobian:

$$J_{g12}=-T_s\omega_y\sin\left(\phi\left(k\right)\right)-T_s\omega_z\left(k\right)\cos\left(\phi\left(k\right)\right)$$

$$J_{g14}=T_s\cos\left(\phi\left(k\right)\right)$$

$$J_{g15}=-T_s\sin\left(\phi\left(k\right)\right)$$

$$J_{g21}=T_s\omega_z\left(k\right)\cos\left(\phi\left(k\right)\right)\left[\tan^2\left(\theta\left(k\right)\right)+1\right]+T_s\omega_y\left(k\right)\sin\left(\phi\left(k\right)\right)\left[\tan^2\left(\theta\left(k\right)\right)+1\right]$$

$$J_{g22}=T_s\omega_y\left(k\right)\cos\left(\phi\left(k\right)\right)\tan\left(\theta\left(k\right)\right)-T_s\omega_z\left(k\right)\sin\left(\phi\left(k\right)\right)\tan\left(\theta\left(k\right)\right)+1$$

$$J_{g24}=T_s\sin\left(\phi\left(k\right)\right)\tan\left(\theta\left(k\right)\right)$$

$$J_{g25}=T_s\cos\left(\phi\left(k\right)\right)\tan\left(\theta\left(k\right)\right)$$

$$J_g=\frac{\partial g}{\partial x}=\begin{bmatrix} 1 & J_{g12} & 0 & J_{g14} & J_{g15} \\ J_{g21} & J_{g22} & T_s & J_{g24} & J_{g25} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Thực hiện rời rạc hóa các tín hiệu quan sát (không xét nhiễu)**

$$z(k) = \begin{bmatrix} a_x(k) & a_y(k) & g_x(k) & g_y(k) & g_z(k) \end{bmatrix}^T$$

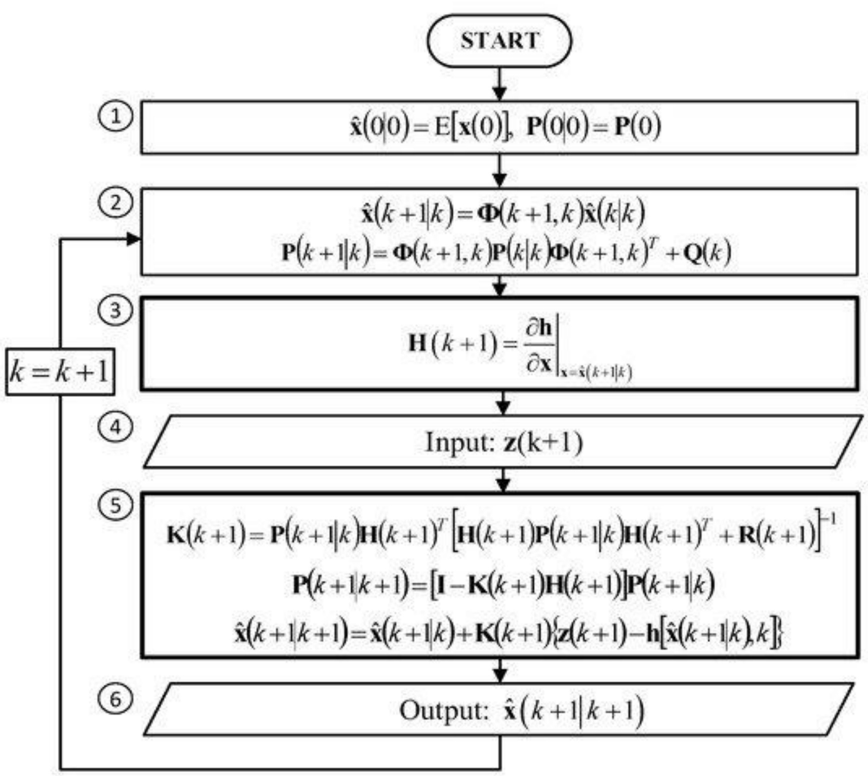
$$z(k+1) = h\left(x(k)\right)$$

$$h\left(x(k)\right) = \begin{bmatrix} -\sin\left(\theta(k)\right) \\ \sin\left(\phi(k)\right)\cos\left(\theta(k)\right) \\ \omega_x(k) \\ \omega_y(k) \\ \omega_z(k) \end{bmatrix}$$

Ma trận Jacobian:

$$J_h = \frac{\partial h}{\partial x} = \begin{bmatrix} -\cos\left(\theta(k)\right) & 0 & 0 & 0 & 0 \\ -\sin\left(\phi(k)\right)\sin\left(\theta(k)\right) & \cos\left(\phi(k)\right)\cos\left(\theta(k)\right) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Vòng lặp bộ lọc Extended Kalman



Đánh giá kết quả bằng độ phù hợp (fitness) và sai số toàn phương trung bình (RMSE)

$$fitness = \left(1 - \frac{\sum_{k=1}^{N}\left[y(k) - \hat{y}(k,\hat{\theta}_N)\right]^2}{\sum_{k=1}^{N}\left[y(k) - \bar{y}\right]^2}\right) \times 100\% \quad ; \quad RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

CHƯƠNG TRÌNH MATLAB THỰC HIỆN BÀI TOÁN

```
clear all;
clc;
%% Load data from XSENS imu
%% Select dataset "I" from provided datasets in pattern1\...:

%----- i=0:4 -----%
I = 0;  % Select i
raw_data  =  load(strcat('pattern1\MT_cal_00300827_00',num2str(i),'.log'));
euler_data  =
load(strcat('pattern1\MT_euler_00300827_00',num2str(i),'.log'));

%% Basic information:
N = max(size(euler_data));       % total dynamic steps
n = 5;                           % number of state
Ts = 0.01;                       % sample rate
tt = euler_data(:,1);            % time stamp

%% Data pre-processing:
g_const = 9.81;
acc  = raw_data(:,2:3)/g_const;  % [m/s^2]
gyro = raw_data(:,5:7);          % [rad/s]
mag  = raw_data(:,8:10);         % [mgauge]
mag = mag/norm(mag)*10^(-3);     % [gauge]

actual_value = zeros(N,2);               % Actual value
actual_value(:,1) = euler_data(:,3);     % theta [o]
actual_value(:,2) = euler_data(:,2);     % phi [o]

%% Std of process:
% Choose q1 for the best estimation
switch i
    case 0
        q1 = 5;
    case 1
        q1 = 0.5;
    case 2
        q1 = 5;
    case 3
        q1 = 30;
    case 4
        q1 = 1;
    % General case:
    otherwise
        q1 = 10000;
end
```

```
Q = diag([0,0,q1,q1,q1]);
L = diag([0,0,1,1,1])*Ts;

%% Std of measurement:
% Choose r1, r2, r3 for the best estimation
switch i
    case 0
        r1 = 0.1;
        r2 = 0.0001;
        r3 = 0.01;
    case 1
        r1 = 0.0006;
        r2 = 0.01;
        r3 = 0.02;
    case 2
        r1 = 0.001;
        r2 = 0.002;
        r3 = 0.01;
    case 3
        r1 = 0.01;
        r2 = 0.01;
        r3 = 0.02;
    case 4
        r1 = 0.00005;
        r2 = 0.0007;
        r3 = 0.01;
    % General case:
    otherwise
        r1 = 0.001;
        r2 = 0.001;
        r3 = 0.001;
end
R = diag([r1,r2,r3,r3,r3]);
M = eye(5);

%% Allocate memory for estimation values:
estimate_value = zeros(N,n);

%% Initialize values:
P_minus = zeros(n,n);        % Initialize P_minus
P = zeros(n,n);              % Initialize P
% Initialize x_hat_minus and x_hat for the best estimation
switch i
    case 0
        x_hat_minus = [-44.62*pi/180; -39.94*pi/180; -0.014954; 0.000145;
0.016868];
        x_hat = x_hat_minus;
    case 1
        x_hat_minus = [1.21*pi/180; 18.19*pi/180; -0.003714; -0.000136; -
0.018198];
        x_hat = x_hat_minus;
    case 2
        x_hat_minus = [1.64*pi/180; -1.22*pi/1808; -0.022062; -0.001835;
0.016377];
        x_hat = x_hat_minus;
    case 3
        x_hat_minus = [-10.81*pi/180; 4.46*pi/180; -0.004984; 0.060824;
0.000847];
        x_hat = x_hat_minus;
    case 4
```

```
        x_hat_minus = [-16.21*pi/180; 27.80*pi/180; -0.004858; -0.009552; -
0.003057];
        x_hat = x_hat_minus;
    % General case:
    otherwise
        x_hat_minus = zeros(n,n);
        x_hat = zeros(n,n);
end

%% Kalman filter loop:
for k=1:N-1
    %% (value substition for the next step)
    theta = x_hat_minus(1);
    phi = x_hat_minus(2);
    omega_x = x_hat_minus(3);
    omega_y = x_hat_minus(4);
    omega_z = x_hat_minus(5);
    %% Compute Jacobian matrix of g(u_t,x_t_1):
    Jg12 = - Ts*omega_y*sin(phi) - Ts*omega_z*cos(phi);
    Jg14 = Ts*cos(phi);
    Jg15 = -Ts*sin(phi);
    Jg21 = Ts*omega_z*cos(phi)*(tan(theta)^2 + 1) +
Ts*omega_y*sin(phi)*(tan(theta)^2 + 1);
    Jg22 = Ts*omega_y*cos(phi)*tan(theta) - Ts*omega_z*sin(phi)*tan(theta) +
1;
    Jg24 = Ts*sin(phi)*tan(theta);
    Jg25 = Ts*cos(phi)*tan(theta);

    Jg = [1, Jg12 , 0 , Jg14 , Jg15;...
       Jg21 , Jg22, Ts , Jg24 , Jg25;...
       0 , 0 , 1 , 0 , 0;...
       0 , 0 , 0 , 1 , 0;...
       0 , 0 , 0 , 0 , 1];

    %% (value substition for the next step)
    theta = x_hat(1);
    phi = x_hat(2);
    omega_x = x_hat(3);
    omega_y = x_hat(4);
    omega_z = x_hat(5);

    g11 = theta - Ts*omega_z*sin(phi) + Ts*omega_y*cos(phi);
    g21 = phi + Ts*omega_x + Ts*omega_z*cos(phi)*tan(theta) +
Ts*omega_y*sin(phi)*tan(theta);
    g31 = omega_x;
    g41 = omega_y;
    g51 = omega_z;

    g=[g11; g21; g31; g41; g51];

    %% Project the error covariance ahead:
    x_hat_minus = g;
    P_minus = Jg*P*Jg' +L*Q*L';

    %% (value substition for the next step)
    theta = x_hat_minus(1);
    phi = x_hat_minus(2);
    omega_x = x_hat_minus(3);
    omega_y = x_hat_minus(4);
    omega_z = x_hat_minus(5);
```

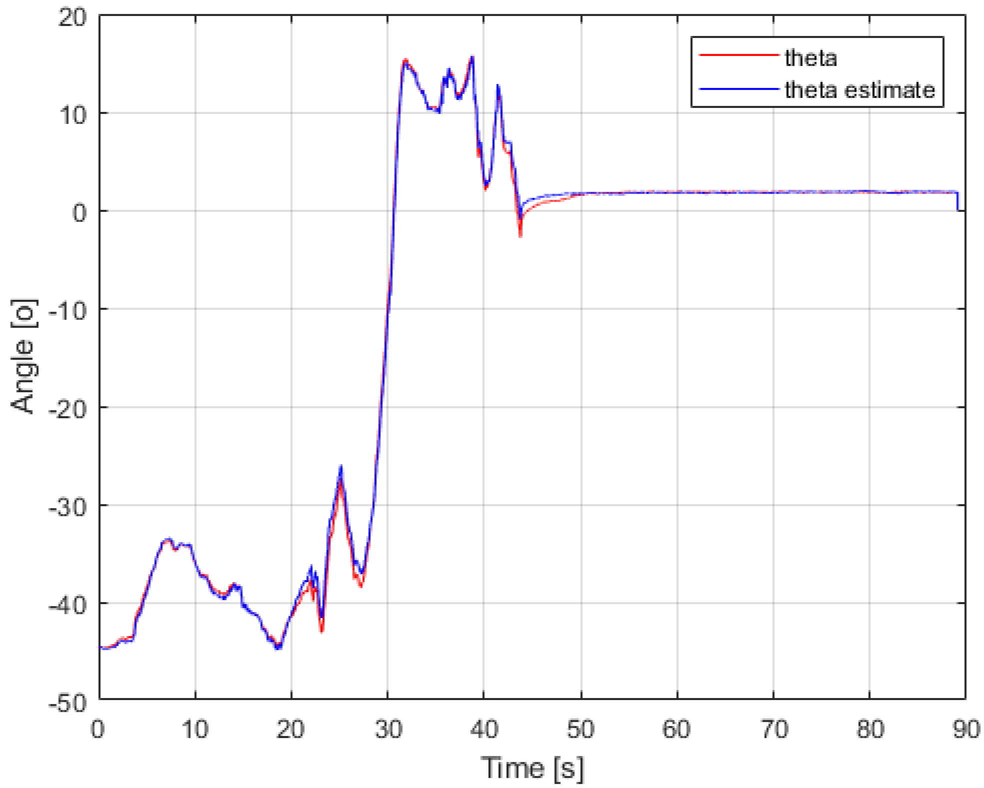
```
    %% Compute Jacobian matrix of h(x_t):
    Jh11 = -cos(theta);
    Jh21 = -sin(phi)*sin(theta);
    Jh22 = cos(phi)*cos(theta);

    Jh=[Jh11 , 0 , 0 , 0 , 0;
        Jh21 , Jh22 , 0 , 0 , 0;
        0 , 0 , 1 , 0 , 0;
        0 , 0 , 0 , 1 , 0;
        0 , 0 , 0 , 0 , 1];

    %% Compute Kalman gain:
    S = Jh*P_minus*Jh’+M*R*M’;
    K = P_minus*Jh’*inv(S);

    %% (value substition for the next step)
    h11 = -sin(theta);
    h21 = sin(phi)*cos(theta);
    h31 = omega_x;
    h41 = omega_y;
    h51 = omega_z;

    hx = [h11; h21; h31; h41; h51];

    %% Update estimate with measurement:
    z = [acc(k,1); acc(k,2); gyro(k,1); gyro(k,2); gyro(k,3)];
    x_hat = x_hat_minus + K*(z – hx);
    estimate_value(k,☺ = x_hat;   % Store estimate values

    %% Compute error covariance for updated estimate:
    P = (eye(n)-(K*Jh))*P_minus;

    %% Estimate value data post-processing:
    while(estimate_value(k,1)<-pi)
        estimate_value(k,1)=estimate_value(k,1)+2*pi;
    end
    while(estimate_value(k,2)<-pi)
        estimate_value(k,2)=estimate_value(k,2)+2*pi;
    end

    while(estimate_value(k,1)>pi)
        estimate_value(k,1)=estimate_value(k,1)-2*pi;
    end
    while(estimate_value(k,2)>pi)
        estimate_value(k,2)=estimate_value(k,2)-2*pi;
    end

end

%% Visualize the results
title_labels = {‘Estimation of theta angle’, ‘Estimation of phi angle’};
legend_labels = {{‘theta’, ‘theta estimate’}, {‘phi’, ‘phi estimate’}};
for I = 1:2
    figure(i);
    plot(tt, actual_value(:,i),'r');
    hold on;
    plot(tt, estimate_value(:,i)*180/pi,'b');
    hold off;
```

```
    legend(legend_labels{i});
    xlabel('Time [s]');
    ylabel('Angle [o]');
    grid on;

    %% Fitness evaluation
    fitness = fitnessCalculator(estimate_value(:,i)*180/pi, 
actual_value(:,i), N);
    RMSE = errorCalculator(estimate_value(:,i)*180/pi, actual_value(:,i), N);
    txt = append('_ - FITNESS = ',num2str(fitness),'%',', RMSE = 
',num2str(RMSE),' [o]');
    title(append(title_labels{i}, txt));
end

function [fitness] = fitnessCalculator(estimate, true, N)
mean_value = mean(true(:));
a=0;
b=0;
for k=1:N
    a=a+(true(k)-estimate(k))^2;
    b=b+(true(k)-mean_value)^2;
end
fitness = (1-a/b)*100;
end
function [RMSE] = errorCalculator(estimate, true, N)
RMSE = 0;
for k=1:N
    RMSE = RMSE + (true(k)-estimate(k))^2;
end
RMSE = sqrt(RMSE/(N));
end
```

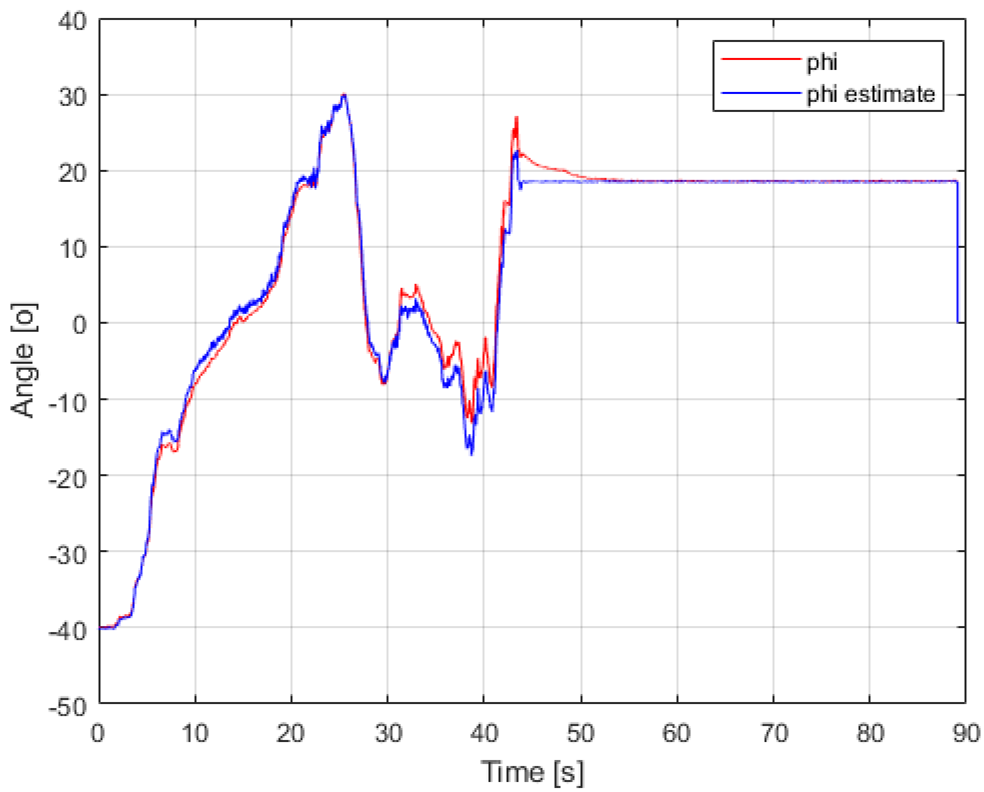
KẾT QUẢ NHẬN DẠNG

- **Kết quả nhận dạng trên tập dataset** `MT_cal_00300827_000.log`

Estimation of theta angle - FITNESS = 99.9263%, RMSE = 0.54359 [o]

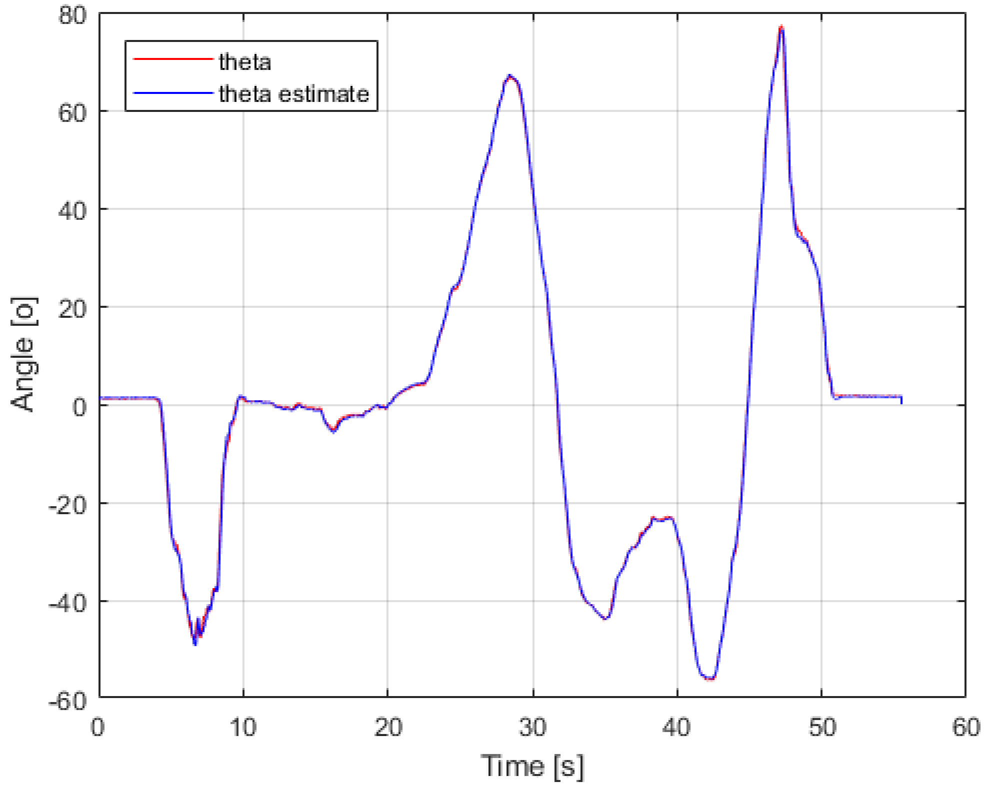


Estimation of phi angle - FITNESS = 99.1475%, RMSE = 1.4991 [o]

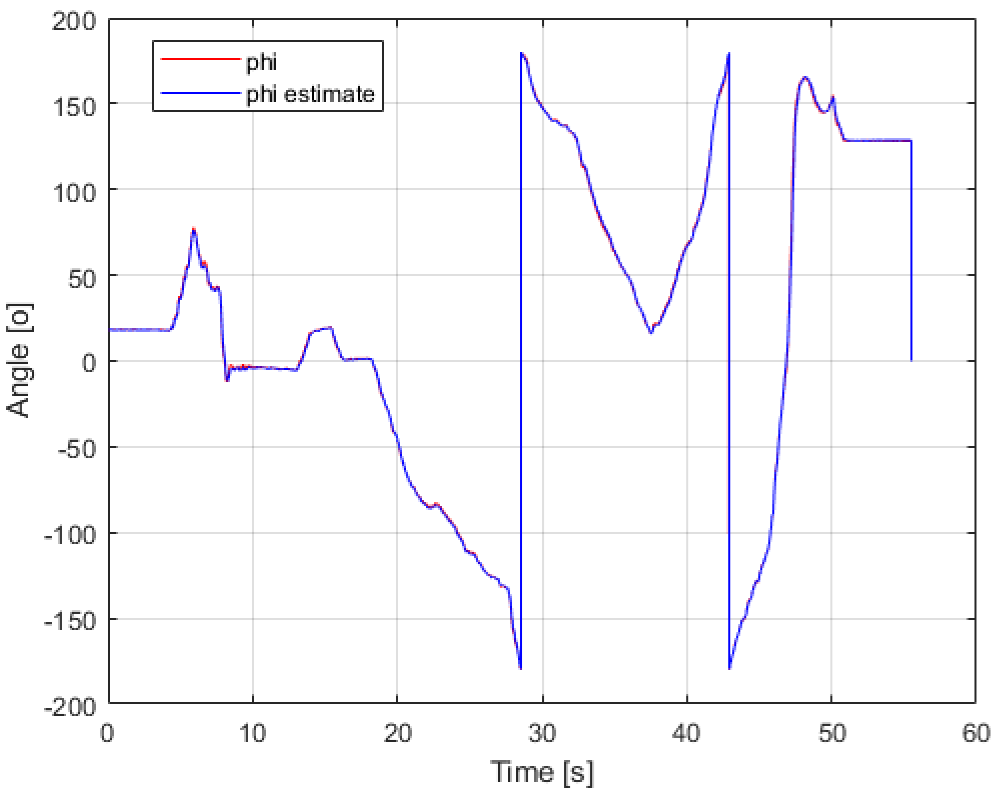


- **Kết quả nhận dạng trên tập dataset** `MT_cal_00300827_001.log`

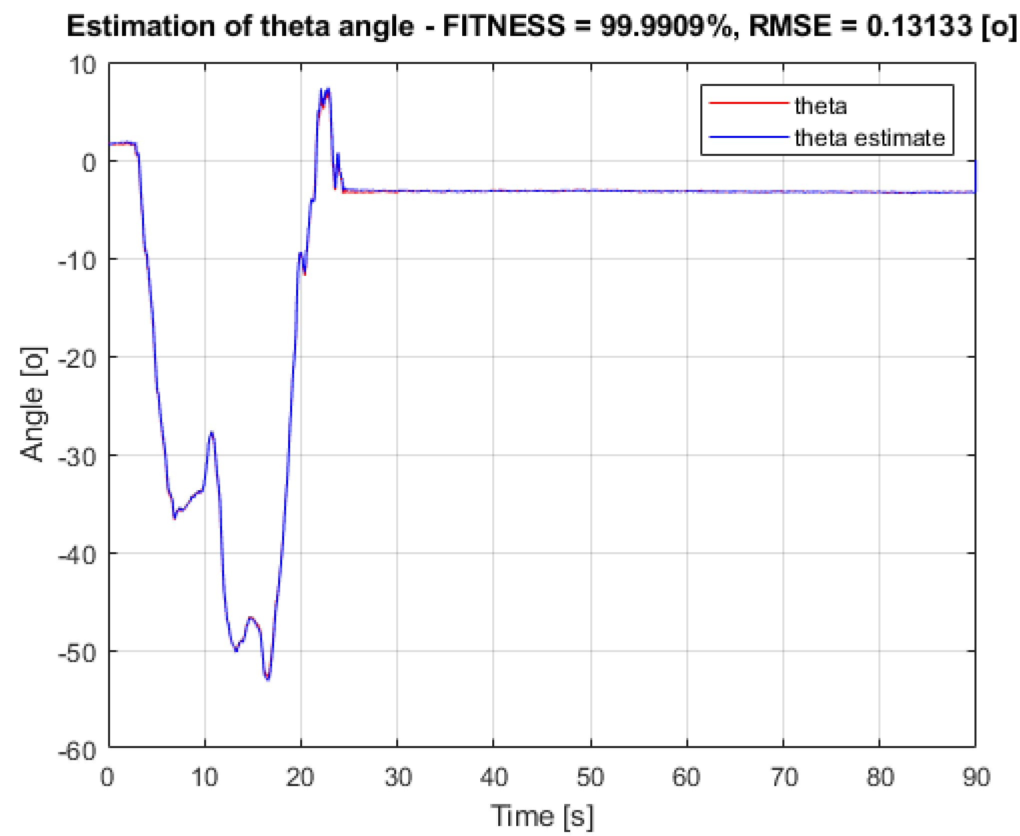
Estimation of theta angle - FITNESS = 99.9633%, RMSE = 0.57556 [o]

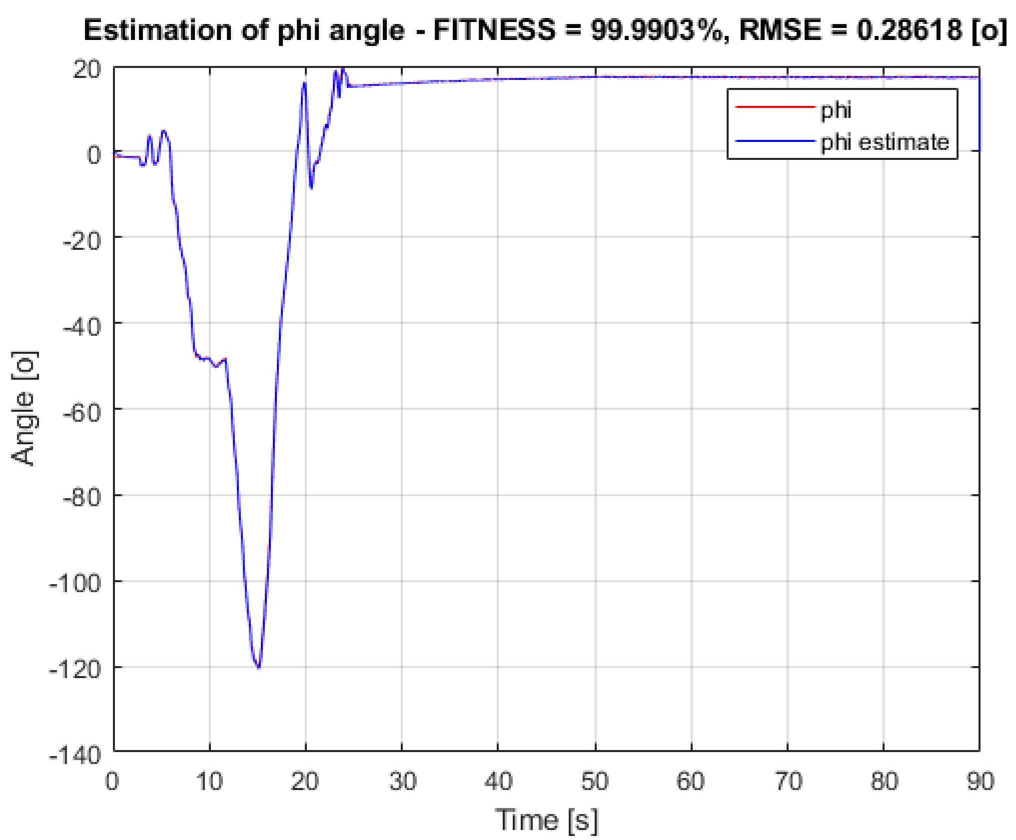


Estimation of phi angle - FITNESS = 99.6655%, RMSE = 5.2483 [o]

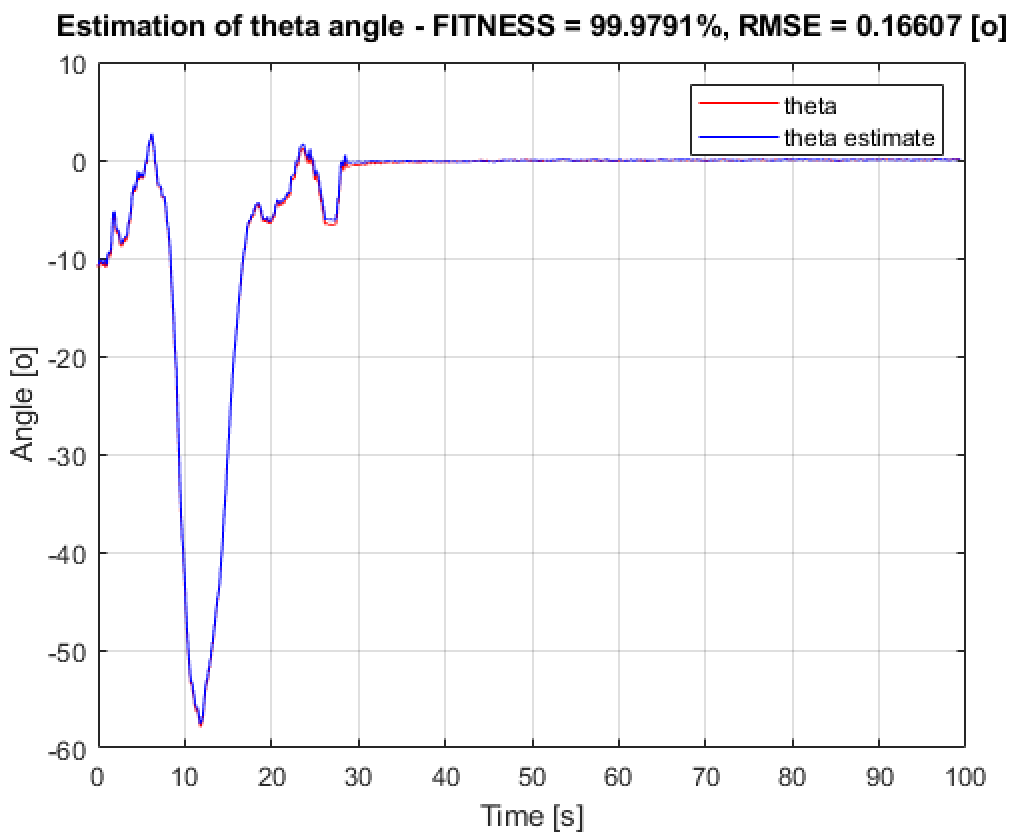


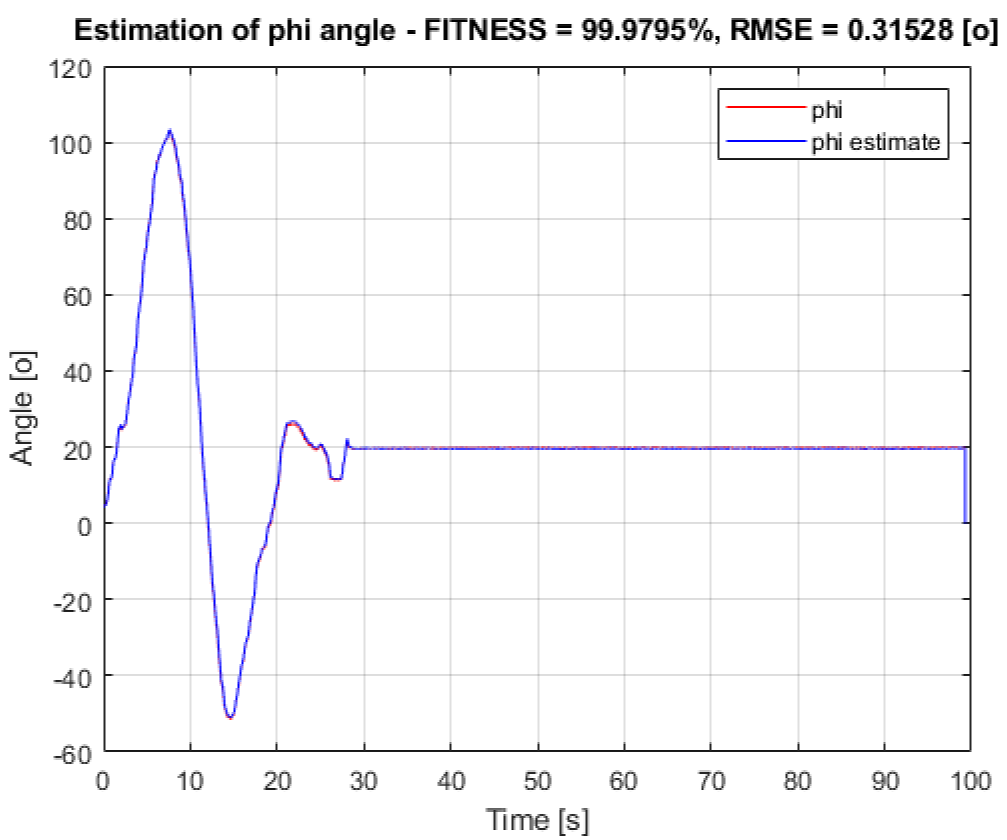
- **Kết quả nhận dạng trên tập dataset** `MT_cal_00300827_002.log`


Estimation of theta angle - FITNESS = 99.9909%, RMSE = 0.13133 [o]
theta
theta estimate
Angle [o]
Time [s]
10
0
-10
-20
-30
-40
-50
-60
0
10
20
30
40
50
60
70
80
90

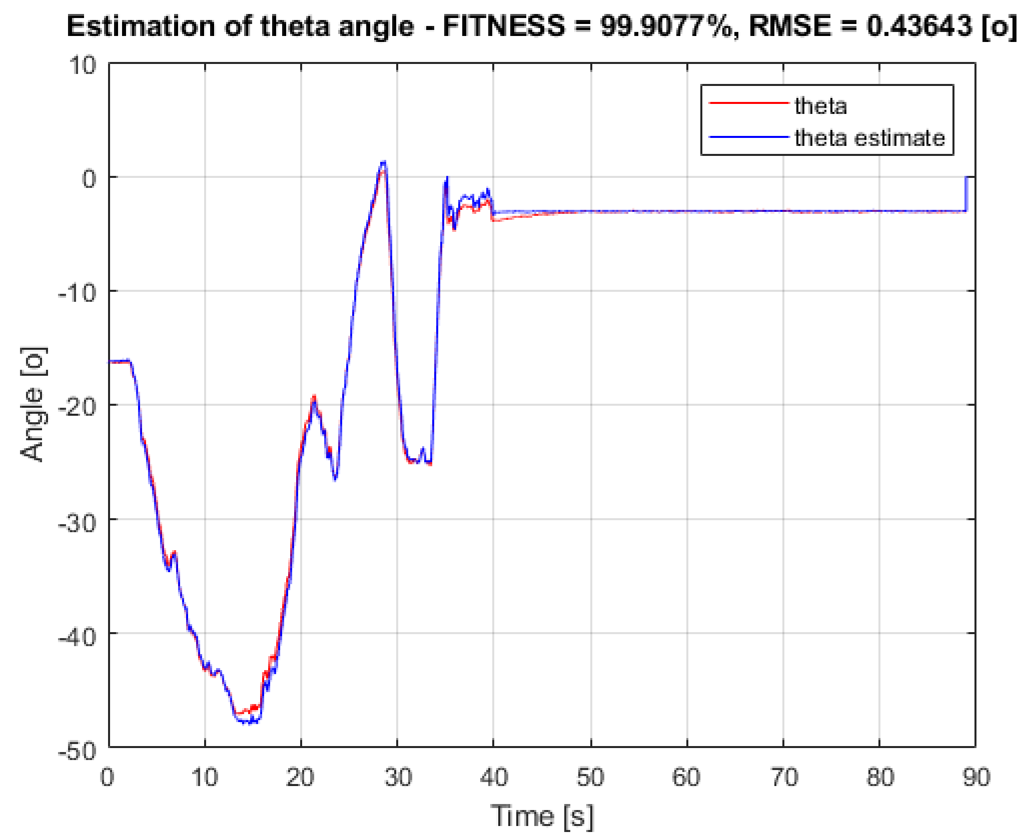

Estimation of phi angle - FITNESS = 99.9903%, RMSE = 0.28618 [o]
phi
phi estimate
Angle [o]
Time [s]
20
0
-20
-40
-60
-80
-100
-120
-140
0
10
20
30
40
50
60
70
80
90

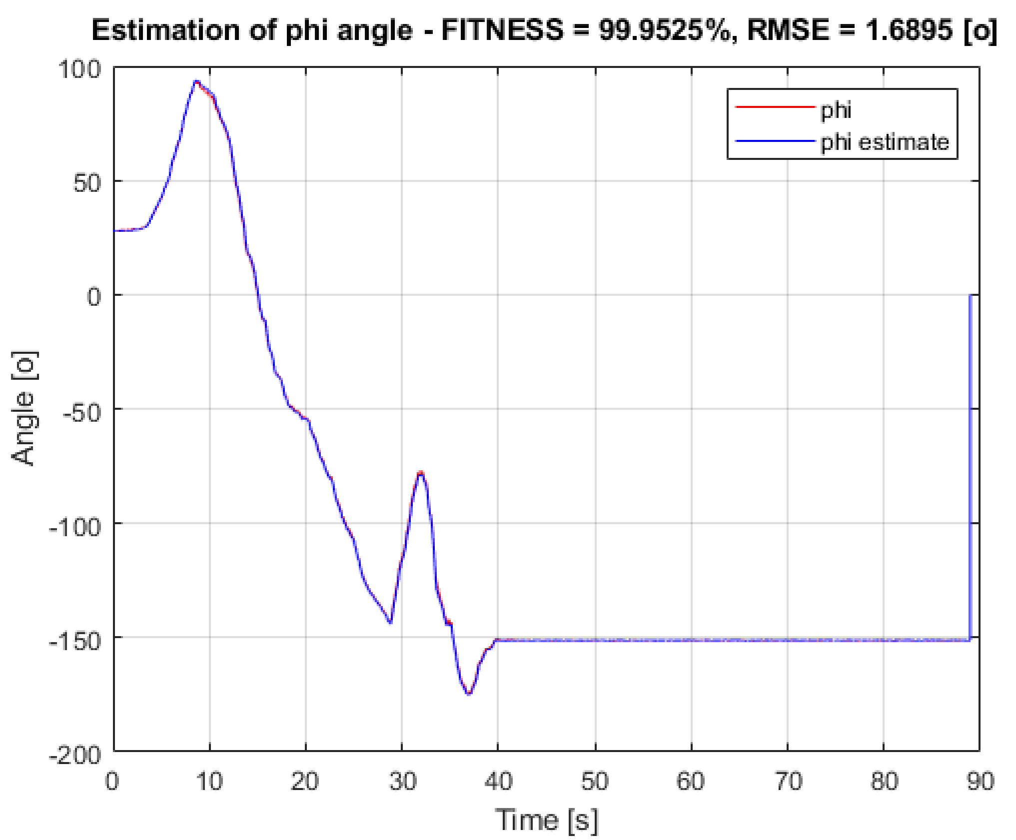
- **Kết quả nhận dạng trên tập dataset** `MT_cal_00300827_003.log`


Estimation of theta angle - FITNESS = 99.9791%, RMSE = 0.16607 [o]
theta
theta estimate
Angle [o]
Time [s]
10
0
-10
-20
-30
-40
-50
-60
0
10
20
30
40
50
60
70
80
90
100


Estimation of phi angle - FITNESS = 99.9795%, RMSE = 0.31528 [o]
phi
phi estimate
Angle [o]
Time [s]
120
100
80
60
40
20
0
-20
-40
-60
0
10
20
30
40
50
60
70
80
90
100

- **Kết quả nhận dạng trên tập dataset** `MT_cal_00300827_004.log`


Estimation of theta angle - FITNESS = 99.9077%, RMSE = 0.43643 [o]
theta
theta estimate
Angle [o]
Time [s]
10
0
-10
-20
-30
-40
-50
0
10
20
30
40
50
60
70
80
90


Estimation of phi angle - FITNESS = 99.9525%, RMSE = 1.6895 [o]
phi
phi estimate
Angle [o]
Time [s]
100
50
0
-50
-100
-150
-200
0
10
20
30
40
50
60
70
80
90

- **Nhận xét kết quả**

Độ phù hợp của ước lượng trong bài làm đạt kết quả gần như chính xác, chỉ tiêu fitness đạt được ở các ước lượng đều trên 99% và cũng như sai số toàn phương trung bình ước lượng nhỏ.