HA3 Analysys

Table of Contents

1 Approximations of mean and covariance

a

```
clear; clc; close all;
format short g;

s1.x0 = [120; 120];
s1.P = diag([5^2, 10^2]);

s2.x0 = [120; -20];
s2.P = diag([5^2, 10^2]);

s1_pos = [0; 100];
s2_pos = [100; 0];

sigma_phi = (0.1*pi/180);
R = diag([sigma_phi^2, sigma_phi^2]);


% Sample from the distrubution
N = 10000;
s1.x = mvnrnd(s1.x0, s1.P, N)';
s2.x = mvnrnd(s2.x0, s2.P, N)';
for i = 1:N
    s1.y(:,i) = dualBearingMeasurement(s1.x(:,i), s1_pos, s2_pos);
    s1.y(:,i) = s1.y(:,i) + mvnrnd([0;0], R, 1)';

    s2.y(:,i) = dualBearingMeasurement(s2.x(:,i), s1_pos, s2_pos);
    s2.y(:,i) = s2.y(:,i) + mvnrnd([0;0], R, 1)';
end

% Monte carlo mean & covar approximation
s1.y_mc = sum(s1.y, 2) / N;
s2.y_mc = sum(s2.y, 2)/ N;
```

```
s1.y_covar = 0;
s2.y_covar = 0;
for i = 1:N
    s1.y_covar = s1.y_covar + (s1.y(:,i)-s1.y_mc)*(s1.y(:,i)-
s1.y_mc)';
    s2.y_covar = s2.y_covar + (s2.y(:,i)-s2.y_mc)*(s2.y(:,i)-
s2.y_mc)';
end
s1.y_covar = s1.y_covar/N;
s2.y_covar = s2.y_covar / N;
```

b

```
h = @(x) dualBearingMeasurement(x, s1_pos, s2_pos);
level = 3;
npoints = 5000;

% State density 1
% Mean
[s1.x_ekf, s1.P_ekf] = nonLinKFprediction(s1.x0, s1.P, h, R, 'EKF');
[s1.x_ukf, s1.P_ukf] = nonLinKFprediction(s1.x0, s1.P, h, R, 'UKF');
[s1.x_ckf, s1.P_ckf] = nonLinKFprediction(s1.x0, s1.P, h, R, 'CKF');
% Covariance
[s1.xy_mc]  = sigmaEllipse2D( s1.y_mc, s1.y_covar, level, npoints );
[s1.xy_ekf] = sigmaEllipse2D( s1.x_ekf, s1.P_ekf, level, npoints );
[s1.xy_ukf] = sigmaEllipse2D( s1.x_ukf, s1.P_ekf, level, npoints );
[s1.xy_ckf] = sigmaEllipse2D( s1.x_ckf, s1.P_ekf, level, npoints );


% State density 2
% Mean
[s2.x_ekf, s2.P_ekf] = nonLinKFprediction(s2.x0, s2.P, h, R, 'EKF');
[s2.x_ukf, s2.P_ukf] = nonLinKFprediction(s2.x0, s2.P, h, R, 'UKF');
[s2.x_ckf, s2.P_ckf] = nonLinKFprediction(s2.x0, s2.P, h, R, 'CKF');
% Covariance
[s2.xy_mc]  = sigmaEllipse2D( s2.y_mc, s2.y_covar, level, npoints );
[s2.xy_ekf] = sigmaEllipse2D( s2.x_ekf, s2.P_ekf, level, npoints );
[s2.xy_ukf] = sigmaEllipse2D( s2.x_ukf, s2.P_ukf, level, npoints );
[s2.xy_ckf] = sigmaEllipse2D( s2.x_ckf, s2.P_ckf, level, npoints );


% Plot scenario 1
figure(1); clf; hold on; grid on;
%Samples
plot(s1.y(1,:), s1.y(2,:), '.')
options = {'MarkerSize', 10, 'LineWidth', 1};
% Mean
plot(s1.y_mc(1), s1.y_mc(2), 'yo', options{:});
plot(s1.x_ekf(1), s1.x_ekf(2), 'mo', options{:});
plot(s1.x_ukf(1), s1.x_ukf(2), 'go', options{:});
plot(s1.x_ckf(1), s1.x_ckf(2), 'ro', options{:});
% 3-Sigma Ellipse
```

```
plot(s1.xy_mc(1,:), s1.xy_mc(2,:), 'y');
plot(s1.xy_ekf(1,:), s1.xy_ekf(2,:), 'm');
plot(s1.xy_ukf(1,:), s1.xy_ukf(2,:), 'g');
plot(s1.xy_ckf(1,:), s1.xy_ckf(2,:), 'r');

title('Estimation method comparison 1');
xlabel('y1');
ylabel('y2');

test = {'Samples', '$\hat{\mu}_{mc}$', '$\hat{\mu}_{ekf}$', '$
\hat{\mu}_{ukf}$', ...
    '$
\hat{\mu}_{ckf}$', '$3\sigma_{mc}$', '$3\sigma_{ekf}$', '$3\sigma_{ukf}$', ...
     '$3\sigma_{ckf}$'
    };
legend(test{:}, 'Interpreter','latex', 'FontSize', 12)


% Plot scenario 2
figure(2); clf; hold on; grid on;
%Samples
plot(s2.y(1,:), s2.y(2,:), '.')
%Mean
plot(s2.y_mc(1), s2.y_mc(2), 'yo', options{:});
plot(s2.x_ekf(1), s2.x_ekf(2), 'mo', options{:});
plot(s2.x_ukf(1), s2.x_ukf(2), 'go', options{:});
plot(s2.x_ckf(1), s2.x_ckf(2), 'ro', options{:});
% 3-Sigma Ellipse
plot(s2.xy_mc(1,:), s2.xy_mc(2,:), 'y');
plot(s2.xy_ekf(1,:), s2.xy_ekf(2,:), 'm');
plot(s2.xy_ukf(1,:), s2.xy_ukf(2,:), 'g');
plot(s2.xy_ckf(1,:), s2.xy_ckf(2,:), 'r');

title('Estimation method comparison 2');
xlabel('y1');
ylabel('y2');

test = {'Samples', '$\hat{\mu}_{mc}$', '$\hat{\mu}_{ekf}$', '$
\hat{\mu}_{ukf}$', ...
    '$
\hat{\mu}_{ckf}$', '$3\sigma_{mc}$', '$3\sigma_{ekf}$', '$3\sigma_{ukf}$', ...
     '$3\sigma_{ckf}$'
    };
legend(test{:}, 'Interpreter','latex', 'FontSize', 12)
```

2 Non-linear Kalman filtering

a & b

```
%4900
%4956
% 4960
```

```
rng(654645)
x0 = [0 0 14 0 0]';
P0 = diag([10 10 2 (pi/180) (5*pi/180)].^2);

s1 = [-200; 100];
s2 = [-200, -100];
Ts = 1;

N = 100;
f = @(x) coordinatedTurnMotion(x, Ts);
h = @(x) dualBearingMeasurement(x, s1, s2);

Q = diag([0 0 1 0 pi/180].^2);
x = genNonLinearStateSequence(x0, P0, f, Q, N);

for i = 1:2
    if(i == 1)
       R = diag([10*pi/180 0.5*pi/180].^2);
    else
       R = diag([0.5*pi/180 0.5*pi/180].^2);
    end

    y.samples = genNonLinearMeasurementSequence(x, h, R);

    % Measuremsnts
    % Calculate intersection
    t1 = y.samples(1,:);
    t2 = y.samples(2,:);
    x_coordinate = (tan(t1)*s1(1) - tan(t2)*s2(1) + s2(2) - s1(2)) ./
 (tan(t1) - tan(t2));
    y_coordinate = tan(t1).*(x_coordinate - s1(1)) + s1(2);


    n = 1;
    for type = ["EKF", "UKF", "CKF"]
        figure(n + (i-1)*3); clf; hold on; grid on;

        % True state
        plot(x(1,:), x(2,:), 'b', 'LineWidth', 2);

        % Measurements
        plot(x_coordinate, y_coordinate, 'go');

        % Filter
        [xf, Pf] = nonLinearKalmanFilter(y.samples, x0, P0, f, Q, h,
 R, type);
        plot(xf(1,:), xf(2,:), '--r', 'LineWidth', 2);

        %Sensors
        plot([s1(1) s2(1)],[s1(2) s2(2)], 'om');
        %plot(s2(1),s2(2), 'om');

        % 3 Sigma curves
        for k = 1:5:N
```

```
            xy = sigmaEllipse2D( xf(1:2,k), Pf(1:2,1:2,k), 3, 500);
            plot(xy(1,:), xy(2,:),  'Color', '#4DBEEE');
        end

       legend('True state', 'Measurements', type, 'Sensors', '$3\sigma
$ level curve', 'Interpreter', 'latex');
        title("Case " + i + " with " + type)
        xlabel("x")
        ylabel("y")
        n = n + 1;
    end

end
```

c

```
rng(666);
N = 100;
combined_est_error_ekf = [];
combined_est_error_ukf = [];
combined_est_error_ckf = [];
what_case = 1;

%Generate all data
for k = 1:100
    Q = diag([0 0 1 0 pi/180].^2);
    x = genNonLinearStateSequence(x0, P0, f, Q, N);
    if(what_case == 1)
       R = diag([10*pi/180 0.5*pi/180].^2);
    else
       R = diag([0.5*pi/180 0.5*pi/180].^2);
    end
    y.samples = genNonLinearMeasurementSequence(x, h, R);

    for type = ["EKF", "UKF", "CKF"]
        [xf, Pf] = nonLinearKalmanFilter(y.samples, x0, P0, f, Q, h,
 R, type);
        xf = [x0 xf];

        if type == "EKF"
            est_error.ekf(:,:,k) = x(1:2,:) - xf(1:2,:);
        elseif type == "UKF"
            est_error.ukf(:,:,k) = x(1:2,:) - xf(1:2,:);
        else
            est_error.ckf(:,:,k) = x(1:2,:) -xf(1:2,:);
        end
    end
    combined_est_error_ekf = [combined_est_error_ekf
 est_error.ekf(:,:,k)];
    combined_est_error_ukf = [combined_est_error_ukf
 est_error.ukf(:,:,k)];
    combined_est_error_ckf = [combined_est_error_ckf
 est_error.ckf(:,:,k)];
```

```
end

figure(1); clf; hold on; grid on;
n = 1;

subplot(2,3,n); n = n + 1;
histogram(combined_est_error_ekf(1,:), 'Normalization', 'pdf');
 title("EKF in X");

subplot(2,3,n); n = n + 1;
histogram(combined_est_error_ukf(1,:), 'Normalization', 'pdf');
 title("UKF in X");

subplot(2,3,n); n = n + 1;
histogram(combined_est_error_ckf(1,:), 'Normalization', 'pdf');
 title("CKF in X");


subplot(2,3,n); n = n + 1;
histogram(combined_est_error_ekf(2,:), 'Normalization', 'pdf');
 title("EKF in Y");

subplot(2,3,n); n = n + 1;
histogram(combined_est_error_ukf(2,:), 'Normalization', 'pdf');
 title("UKF in Y");

subplot(2,3,n); n = n + 1;
histogram(combined_est_error_ckf(2,:), 'Normalization', 'pdf');
 title("CKF in Y");
```

3 Tuning non-linear filters

```
rng(6543);
N = 600;
type = "CKF";
P0 = diag([10 10 10 5*pi/180 pi/180].^2);

s1 = [280; -80];
s2 = [280; -200];
Ts = 0.1;

f = @(x) coordinatedTurnMotion(x, Ts);
h = @(x) dualBearingMeasurement(x, s1, s2);

% Measurement noise is assumed to be known
R = diag([4*pi/180 4*pi/180].^2);
%R = diag([0.1*pi/180 0.1*pi/180].^2);

% Generate true track (state sequence
omega = zeros(1,N+1);
% Turn rate
omega(200:400) = -pi/201/Ts;
% Initial state
```

```
x0 = [0 0 20 0 omega(1)]';
% Allocate memory
x = zeros(length(x0),N+1);
x(:,1) = x0;
% Create true track
for i=2:N+1
    % Simulate
    x(:,i) = coordinatedTurnMotion(x(:,i-1), Ts);
    % Set turn-rate
    x(5,i) = omega(i);
end

% Initial prior
x0 = [0 0 0 0 0]';

y = genNonLinearMeasurementSequence(x, h, R);
[x_coord, y_coord] = samplesToState(y, s1, s2);

figure(1); clf;
n = 1;
for i = 1:3
    for k = [0.01 1 100]
        % Process noise is not known and hence tuned
        if(i == 1)
            Q = diag([0 0 k*1 0 pi/180].^2);
            v_k = k; w_k = 1;
        elseif(i == 2)
            Q = diag([0 0 1 0 k*pi/180].^2);
            v_k = 1; w_k = k;
        else
           Q = diag([0 0 k*1 0 k*pi/180].^2);
           v_k = k; w_k = k;
        end
        [xf, Pf] = nonLinearKalmanFilter(y, x0, P0, f, Q, h, R, type);

        subplot(3,3, n); hold on; grid on; n = n + 1;
        % True state
        plot(x(1,:), x(2,:), '-b', 'LineWidth', 2)
        % Measurement
        %plot(x_coord, y_coord, '.r')
        % Filter
        plot(xf(1,:), xf(2,:), '--r', 'LineWidth', 2)

        title("$" + v_k + "\sigma_{v}, " + w_k +
  "\sigma_{\omega}$", 'Interpreter', 'latex');
        xlabel("X Pos");
        ylabel("Y Pos");
        legend('True state', "Filter - " + type)
    end
end
```

Manual tuning

```
figure(2); clf; hold on
```

```
k1 = 0.1;
k2 = 1/4;
Q = diag([0 0 k1*1 0 k2*pi/180].^2);
[xf, Pf] = nonLinearKalmanFilter(y, x0, P0, f, Q, h, R, type);
plot(x(1,:), x(2,:), '-b', 'LineWidth', 2)
plot(xf(1,:), xf(2,:), '--r', 'LineWidth', 2)
title("Tuned Kalman with $" + v_k + "\sigma_{v}, " + w_k +
  "\sigma_{\omega}$", 'Interpreter', 'latex');
grid on
```

c

```
rng(666);
type = "CKF";

y = genNonLinearMeasurementSequence(x, h, R);
[x_coord, y_coord] = samplesToState(y, s1, s2);



n = 1;
for run = ["small", "large", "tuned"]

    if(run == "small")
       v_k = 0.01; w_k = 0.01;
    elseif(run == "large")
       v_k = 100; w_k = 100;
    else
       v_k = 0; w_k = 1;
    end
    Q = diag([0 0 v_k*1 0 w_k*pi/180].^2);
    [xf, Pf] = nonLinearKalmanFilter(y, x0, P0, f, Q, h, R, type);
    xf = [x0 xf];

    error(n,:) = vecnorm(x(1:2,:) - xf(1:2,:));

    figure(n); clf; hold on; grid on;
    % Sensors
    plot([s1(1) s2(1)], [s1(2) s2(2)], 'mo', 'LineWidth', 2)
    text(s1(1)-35, s1(2) + 20, "Sensor 1")
    text(s2(1)-35, s2(2) + 20, "Sensor 2")
    % True track
    plot(x(1,:), x(2,:), '-b', 'LineWidth', 2)
    % Measurements
    plot(x_coord, y_coord, 'g.')
    % Filter
    plot(xf(1,:), xf(2,:), '-r', 'LineWidth', 2)

    % 3 Sigma curves
    for k = 1:5:N
        xy = sigmaEllipse2D( xf(1:2,k), Pf(1:2,1:2,k), 3, 500);
        plot(xy(1,:), xy(2,:), 'Color', '#4DBEEE');
    end
```

```
    xlim([-50 650])
    ylim([-400 100])
    title("\textbf{Position filter with $" + v_k + "\sigma_{v}, " +
 w_k +  "\sigma_{\omega}$}", 'Interpreter', 'latex');
    xlabel("X Pos")
    ylabel("Y Pos")
    legend('Sensors', 'True track','Measurements', type + "
 filter", "$3\sigma$ level curve", 'Interpreter', 'latex')

    n = n + 1;
end

% Prepare error plot
figure(4); clf; hold on; grid on;
plot(0:N, [error(1,:); error(2,:); error(3,:)])
xlabel("Sample [k]")
ylabel("Error")
title("Position error $|p_{k}-\hat{p}_{k|k}|
_2$", 'Interpreter', 'latex')
legend('Small Q', 'Large Q', 'Tuned Q')
```

Published with MATLAB® R2019b