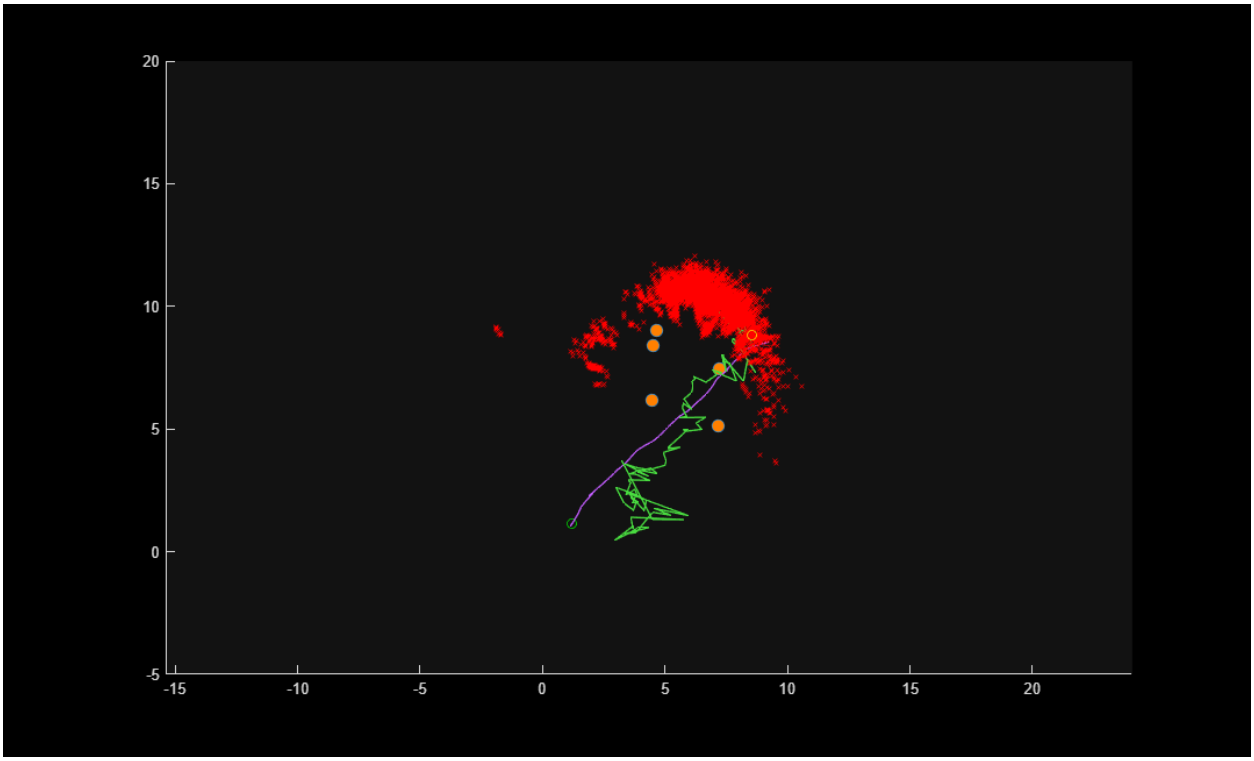
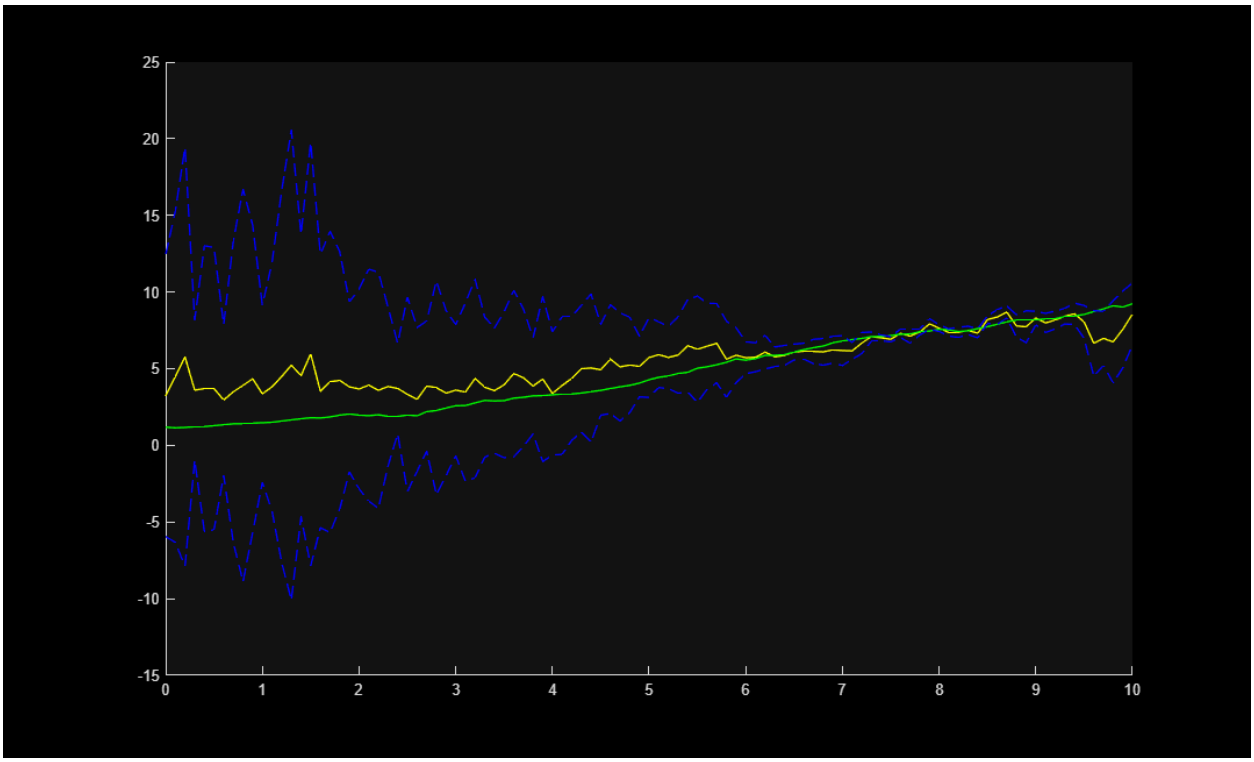
```
clear variables;
close all;
clc;

addpath('..\Functions\');
addpath('codeAndDataForStudents\');

% rng(63);

% Load data
load('problem4data.mat');
load('problem4truth.mat');
num_meas = length(sonar);
nparts = 5000;

% Initialize output
x_out = zeros(num_meas, 3);
P_out = zeros(3, 3, num_meas);
particle_hist = zeros(4, nparts, num_meas+1);

% Set up problem matrices
Q = diag([0.1, 5*pi/180].^2);
R = eye(3);

% Draw initial particles;
x_values = unifrnd(minx, maxx, 1, nparts);
y_values = unifrnd(miny, maxy, 1, nparts);
heading = unifrnd(0, 2*pi, 1, nparts);
weights = 1/nparts*ones(1, nparts);
particles = [x_values; y_values; heading; weights]; % Each col is a particle
particle_hist(:, :, 1) = particles;

% Loop through the measurements
for k=1:num_meas
    % Extract the control input
    u = encoder(k).u; % 2x1
    process_noise = draw_gaussian(zeros(2, 1), Q, nparts);
    u_noisy = u + process_noise;

    % Propagate particles based on the noisy control input
    particles(3, :) = particles(3, :) + u_noisy(2, :);
    particles(1, :) = particles(1, :) + u_noisy(1, :) .*
cos(particles(3, :)); % Theta already updated with noise
    particles(2, :) = particles(2, :) + u_noisy(1, :) .*
sin(particles(3, :));

    particle_hist(:, :, k+1) = particles;

    % Compute zbar for each particle (Pretty convinced this works)
    % (x-y)^2 = x^2 + y^2 - 2xy
    p_sq = sum(particles(1:2, :).^2, 1).';
    b_sq = sum(beacons.^2, 2).';
```

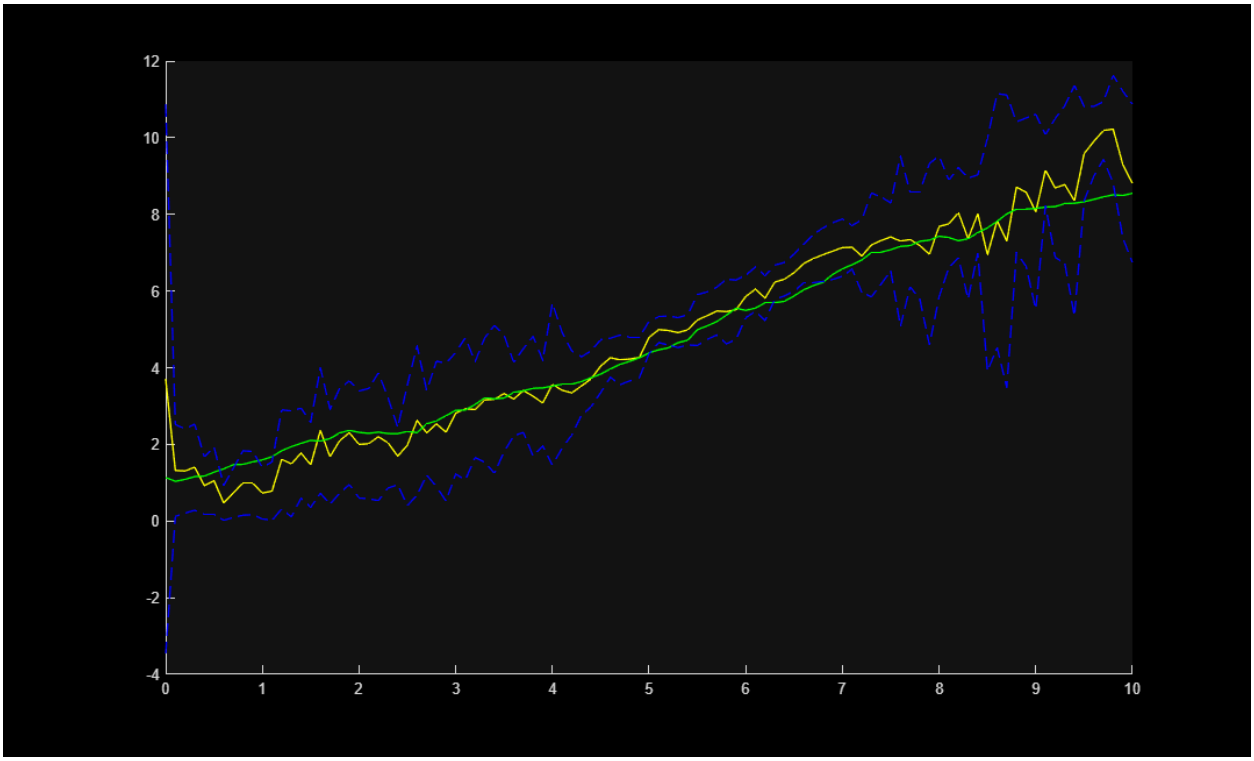
```
    d_sq = max(p_sq + b_sq - 2 * (particles(1:2, :).' * beacons.'), 0);

    % Euclidean distance
    d = sqrt(d_sq);
    d = mink(d, 3, 2).'; % 3xnparts;

    % Update the weights and normalize
    z = sonar(k).z;
    resids = z - d;
    lw = -0.5*sum(resids .* (inv(R) * resids), 1) + particles(4, :);
    lw_max = max(lw);
    w = exp(lw - lw_max);
    w = w / sum(w);
    particles(4, :) = w;

    % Form estimate
    xhat = particles(1:3, :) * particles(4, :).';
    xhat(3, :) = mod(xhat(3, :), 2*pi); % Handle wraparound
    % Calculate covariance TODO
    error = particles(1:3, :) - xhat;
    P = (error .* particles(4, :)) * error';

    % Save estimates
    x_out(k, :) = xhat;
    P_out(:, :, k) = P;

    % Resample (if needed)
    N_eff = 1 / sum(particles(4, :).^2);

    if N_eff < nparts/2
        % disp('We resampled')
        l=1;
        while l < nparts
            eta = unifrnd(0, 1);
            m = 1;
            w_sum = 0;
            while w_sum <= eta && m <= nparts
                w_sum = w_sum + particles(4, m);
                m = m+1;
            end
            xm = particles(1:3, m-1);
            particles(:, l) = [xm; 1/nparts];
            l = l+1;
        end

    end


end
```

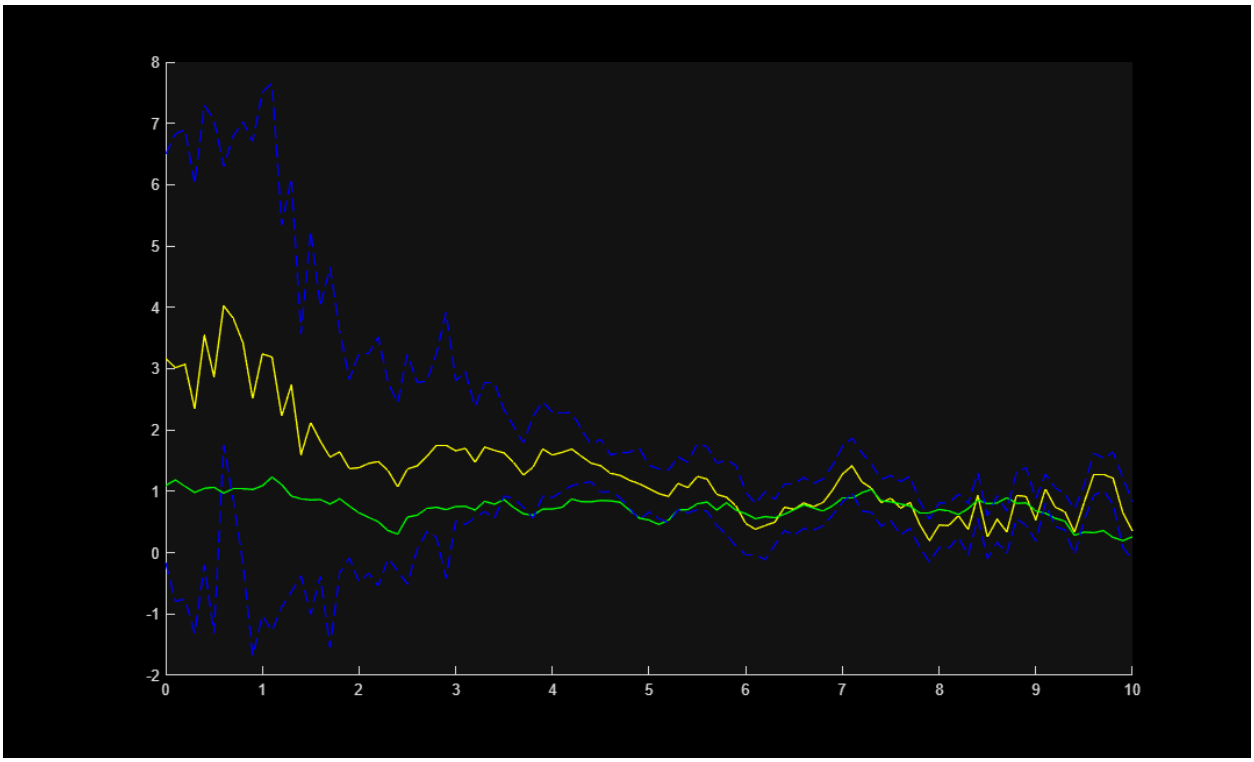
Plotting

```
true_state = [robot.x];
figure;
xlim([-5, 20]);
ylim([-5, 20]);
axis equal;
hold on;
% Plot the static stuff
plot(beacons(:, 1), beacons(:, 2), 'o', 'MarkerFaceColor', [1 0.5 0],
'markersize', 8);
plot(true_state(1, 1), true_state(2, 1), 'go');
plot(true_state(1, end), true_state(2, end), 'r^');
plot(true_state(1, :), true_state(2, :));  % Truth
plot(x_out(:, 1), x_out(:, 2)); % Our estimate
% Plot the points and the estimate at each time step
h = plot(particle_hist(1, :, 1), particle_hist(2, :, 1), 'rx', 'markersize',
4);
est = plot(x_out(1, 1), x_out(1, 2), 'yo', 'MarkerSize', 6);
for k=2:num_meas
    pause(0.01);
    h.XData = particle_hist(1, :, k);
    h.YData = particle_hist(2, :, k);
    est.XData = x_out(k, 1);
    est.YData = x_out(k, 2);
    drawnow;
end

% State vector plots
t = [robot.t];
figure;
hold on;
plot(t, x_out(:, 1), 'y-');
plot(t, true_state(1, :), 'g-');
plot(t, x_out(:, 1) + squeeze(P_out(1, 1, :)), 'b--');
plot(t, x_out(:, 1) - squeeze(P_out(1, 1, :)), 'b--');

figure;
hold on;
plot(t, x_out(:, 2), 'y-');
plot(t, true_state(2, :), 'g-');
plot(t, x_out(:, 2) + squeeze(P_out(2, 2, :)), 'b--');
plot(t, x_out(:, 2) - squeeze(P_out(2, 2, :)), 'b--');

figure;
hold on;
plot(t, x_out(:, 3), 'y-');
plot(t, true_state(3, :), 'g-');
plot(t, x_out(:, 3) + squeeze(P_out(3, 3, :)), 'b--');
plot(t, x_out(:, 3) - squeeze(P_out(3, 3, :)), 'b--');
```


20
15
10
5
0
-5
-15
-10
-5
0
5
10
15
20


25
20
15
10
5
0
-5
-10
-15
0
1
2
3
4
5
6
7
8
9
10





Published with MATLAB® R2025a