Table of Contents

PROBLEM 6

```
clear variables; close all; clc


data_table_acc = readtable('Accelerometer_exp6.csv');


data_table_gyro = readtable('Gyroscope_exp6');


data_table_gps = readtable('Location_exp6');


data_table_mag = readtable('Magnetomerter_exp6');
%===========================================================================

% CORRECT VALUES FOR THESE BIASES AND VARIANCES AS COMPUTED IN
 EXPERIMENT 1
bias_acc = [1.948244;
               1.926003;
               -3.76083]; % biases in accelerometer x,y,z

bias_mag = [-12.11214937;
               -19.67616054;
               22.73696197]; % biases in magnetometer x,y,z

bias_gyro = [0.00001194560806;
               -0.00000912316961;
               -0.00000169621783]; % biases in gyro x,y,z
var_acc  = [4.94; 5.23; 13.5]*10^-5;
var_gyro = [2.1; 2.6; 4.9]*10^-6;
var_mag     = [0.467657;0.744017;0.463155];

var_mag_heading = 1.3227*10^-5;
```

Warning: Column headers from the file were modified to make them valid MATLAB

```
identifiers before creating variable names
for the table. The original column headers
are saved in the VariableDescriptions
property.
Set 'PreserveVariableNames' to true to use
the original column headers as table
variable names.
Warning: Column headers from the file were
modified to make them valid MATLAB
identifiers before creating variable names
for the table. The original column headers
are saved in the VariableDescriptions
property.
Set 'PreserveVariableNames' to true to use
the original column headers as table
variable names.
Warning: Column headers from the file were
modified to make them valid MATLAB
identifiers before creating variable names
for the table. The original column headers
are saved in the VariableDescriptions
property.
Set 'PreserveVariableNames' to true to use
the original column headers as table
variable names.
Warning: Column headers from the file were
modified to make them valid MATLAB
identifiers before creating variable names
for the table. The original column headers
are saved in the VariableDescriptions
property.
Set 'PreserveVariableNames' to true to use
the original column headers as table
variable names.
```

Local gravitational acceleration

```
g = 9.80333; % m/s/s
```

Accelerometers

```
time_stamps_acc = data_table_acc{:, 1};
acc_x = data_table_acc{:, 2};
acc_y = data_table_acc{:, 3};
acc_z = data_table_acc{:, 4};
```

Rate gyros

```
time_stamps_gyro = data_table_gyro{:, 1};
gyro_x = data_table_gyro{:, 2};
gyro_y = data_table_gyro{:, 3};
gyro_z = data_table_gyro{:, 4};
```

Magnetometers

```
time_stamps_mag  = data_table_mag{:, 1};
mag_xb = data_table_mag{:, 2};
mag_yb = data_table_mag{:, 3};
mag_zb = data_table_mag{:, 4};
```

Remove bias

```
acc_x_wo_bias = acc_x - bias_acc(1);
acc_y_wo_bias = acc_y - bias_acc(2);
acc_z_wo_bias = acc_z - bias_acc(3);

gyro_x_wo_bias= gyro_x - bias_gyro(1);
gyro_y_wo_bias= gyro_y - bias_gyro(2);
gyro_z_wo_bias= gyro_z - bias_gyro(3);

mag_xb_wo_bias = mag_xb - bias_mag(1);
mag_yb_wo_bias = mag_yb - bias_mag(2);
mag_zb_wo_bias = mag_zb - bias_mag(3);
```

Initial Pitch, Roll, and Yaw

```
initial_few_pts = 5;
roll_data = atan( acc_y_wo_bias(1:initial_few_pts) ./
 acc_z_wo_bias(1:initial_few_pts) );
pitch_data = asin( acc_x_wo_bias(1:initial_few_pts) / g );

initial_roll = mean( roll_data )
initial_pitch = mean( pitch_data )

tmp1 = [...
 cos(initial_pitch) sin(initial_pitch)*sin(initial_roll)
 sin(initial_pitch)*cos(initial_roll); ...
 0 cos(initial_roll) -sin(initial_roll); ...
 -sin(initial_pitch) cos(initial_pitch)*sin(initial_roll)
 cos(initial_pitch)*cos(initial_roll)] * ...
 [mag_xb_wo_bias(1:initial_few_pts)'; ...
 mag_yb_wo_bias(1:initial_few_pts)';
 mag_zb_wo_bias(1:initial_few_pts)'];

mag_x_wo_bias = tmp1(1,:)';
mag_y_wo_bias = tmp1(2,:)';

magnetic_heading_data = -atan2( mag_y_wo_bias, mag_x_wo_bias );
declination    = -14.07*pi/180;
% Declination  for Worcester, MA found using World Magnetic Model
% https://www.ngdc.noaa.gov/geomag/calculators/
magcalc.shtml#declination

true_heading_data = declination + magnetic_heading_data;
initial_yaw  = mean(true_heading_data)
```

```
initial_roll =

   -0.0303


initial_pitch =

   -0.1781


initial_yaw =

   -0.3886
```

Final Pitch, Roll, Yaw

```
Final_few_pts = (670:679);
roll_data_f = atan( acc_y_wo_bias(Final_few_pts) ./
 acc_z_wo_bias(Final_few_pts) );
pitch_data_f = asin( acc_x_wo_bias(Final_few_pts) / g );

final_roll = mean( roll_data_f )
final_pitch = mean( pitch_data_f )

tmp2 = [...
 cos(final_pitch) sin(final_pitch)*sin(final_roll)
 sin(final_pitch)*cos(final_roll); ...
 0 cos(final_roll) -sin(final_roll); ...
 -sin(final_pitch) cos(final_pitch)*sin(final_roll)
 cos(final_pitch)*cos(final_roll)] * ...
 [mag_xb_wo_bias(Final_few_pts)'; ...
 mag_yb_wo_bias(Final_few_pts)'; mag_zb_wo_bias(Final_few_pts)'];

mag_x_wo_bias_f = tmp2(1,:)';
mag_y_wo_bias_f = tmp2(2,:)';

magnetic_heading_data_f = -atan2( mag_y_wo_bias_f, mag_x_wo_bias_f );
declination    = -14.07*pi/180;
% Declination  for Worcester, MA found using World Magnetic Model
% https://www.ngdc.noaa.gov/geomag/calculators/
magcalc.shtml#declination

true_heading_data_f = declination + magnetic_heading_data_f;
final_yaw  = mean(true_heading_data_f)


final_roll =

   -0.0889
```

```
final_pitch =

   -0.1922


final_yaw =

   -0.0673
```

EKF

```
Q = diag(var_gyro);
R = diag([var_acc; var_mag_heading]);

dt = 0.01;
m1 = 1;
t = max([time_stamps_acc(1), time_stamps_mag(1),
 time_stamps_gyro(1)]);
tfinal  = min([time_stamps_acc(end), time_stamps_mag(end),
 time_stamps_gyro(end)]);
n_time_pts = round( tfinal /dt );


initial_few_points = 10;
final_few_points = 10;
gps_lat_init  = mean( data_table_gps{1:initial_few_points, 2} );
gps_long_init  = mean( data_table_gps{1:initial_few_points, 3} );
gps_lat_final  = mean( data_table_gps{(end-final_few_points):end,
 2} );
gps_long_final  = mean( data_table_gps{(end-final_few_points):end,
 3} );

pos_fin = lla2flat([gps_lat_final gps_long_final 0], ...
 [gps_lat_init gps_long_init], 0, 0);
displacement_gps = norm(pos_fin);
V_loc = displacement_gps / (data_table_gps{end, 1} - data_table_gps{1,
 1});
V_gps = mean( data_table_gps{(data_table_gps{:, 6} > 0), 5} );
V = (V_gps + V_loc)/2;

xhat = [initial_yaw; initial_pitch; initial_roll];
P  = diag([var_mag_heading var_acc(1) var_acc(2)]);

time_stamps_store = zeros(1, n_time_pts);
xhat_store   = zeros(3, n_time_pts);
P_store    = zeros(9, n_time_pts);
P_trace_store  = zeros(1, n_time_pts);

xhat_store(:, 1) = xhat;
P_trace_store(:, 1) = trace(P);
P_store(:, 1)  = reshape(P, 9, 1);
```

```
while (t < tfinal)

 col_gyro = find(time_stamps_gyro <= t, 1, 'last');
 col_acc = find(time_stamps_acc <= t, 1, 'last');
 col_mag = find(time_stamps_mag <= t, 1, 'last');
 t = t + dt;

 u = [gyro_x_wo_bias(col_gyro); gyro_y_wo_bias(col_gyro);
 gyro_z_wo_bias(col_gyro)];

 psi_hat  = xhat(1);
 theta_hat = xhat(2);
 phi_hat  = xhat(3);
 A = [[0;0;0] ...
  [ 0  sin(phi_hat)*tan(theta_hat)*sec(theta_hat)
 sin(phi_hat)*tan(theta_hat)*sec(theta_hat); ...
   0  0             0;...
   0  sin(phi_hat)*sec(theta_hat)^2
 cos(phi_hat)*sec(theta_hat)^2 ]*u ...
  [ 0 cos(phi_hat)*sec(theta_hat)  -sin(phi_hat)*sec(theta_hat); ...
   0 -sin(phi_hat)     -cos(phi_hat); ...
   0 cos(phi_hat)*tan(theta_hat)  sin(phi_hat)*tan(theta_hat)]*u];
 B2 = [...
  0 sin(phi_hat)*sec(theta_hat)  cos(phi_hat)*sec(theta_hat); ...
  0 cos(phi_hat)     -sin(phi_hat); ...
  1 sin(phi_hat)*tan(theta_hat)  cos(phi_hat)*tan(theta_hat);];
 C = [[0;0;0;1] ...
  [V*[0 cos(theta_hat) 0; -cos(theta_hat) 0 -sin(theta_hat); 0
 sin(theta_hat) 0]*u + ...
  g*[cos(theta_hat); sin(theta_hat)*sin(phi_hat);
 sin(theta_hat)*cos(phi_hat)]; 0] ...
  g*[0; -cos(theta_hat)*cos(phi_hat); cos(theta_hat)*sin(phi_hat);
 0]];

 F = eye(3) + A*dt;
 G2 = B2*dt;

 tmp1 = [...
  cos(theta_hat) sin(theta_hat)*sin(phi_hat)
 sin(theta_hat)*cos(phi_hat); ...
  0 cos(phi_hat) -sin(phi_hat); ...
  -sin(theta_hat) cos(theta_hat)*sin(phi_hat)
 cos(theta_hat)*cos(phi_hat)] * ...
  [mag_xb_wo_bias(col_mag); mag_yb_wo_bias(col_mag);
 mag_zb_wo_bias(col_mag)];

 mag_x_wo_bias = tmp1(1);
 mag_y_wo_bias = tmp1(2);

 magnetic_heading_data = -atan2( mag_y_wo_bias, mag_x_wo_bias );
 magnetometer_yaw = declination + magnetic_heading_data;

 x_minus = xhat + attitude_kinematics_asg3(xhat, u)*dt;
```

```
 P_minus = F*P*F' + G2*Q*G2';

 L  = (P_minus * C') / (C * P_minus * C' + R );
 z  = [acc_x_wo_bias(col_acc); ...
  acc_y_wo_bias(col_acc); acc_z_wo_bias(col_acc); ...
  magnetometer_yaw];
 xhat = x_minus + L*(z - attitude_measurement_asg3(xhat, u, V));
 P  = (eye(3) - L*C)*P_minus;

 time_stamps_store(m1 + 1) = t;
 xhat_store(:, m1+1) = xhat;
 P_store(:, m1+1) = reshape(P, 9, 1);
 P_trace_store(:, m1+1) = trace(P);

 m1 = m1 + 1;
end



figure;
subplot(311)
plot(time_stamps_store(1:m1), xhat_store(1, 1:m1)*180/pi, 'LineWidth',
 2);
ylabel('$\psi$ (deg)', 'Interpreter', 'latex', 'FontSize', 14)
xlabel('$t$ (s)', 'Interpreter', 'latex', 'FontSize', 14)
title('Problem 6 Ground Truth Euler
 Angles', 'Interpreter', 'latex', 'FontSize', 18)

subplot(312)
plot(time_stamps_store(1:m1), xhat_store(2, 1:m1)*180/pi, 'LineWidth',
 2);
ylabel('$\theta$ (deg)', 'Interpreter', 'latex', 'FontSize', 14)
xlabel('$t$ (s)', 'Interpreter', 'latex', 'FontSize', 14)

subplot(313)
plot(time_stamps_store(1:m1), xhat_store(3, 1:m1)*180/pi, 'LineWidth',
 2);
ylabel('$\phi$ (deg)', 'Interpreter', 'latex', 'FontSize', 14)
xlabel('$t$ (s)', 'Interpreter', 'latex', 'FontSize', 14)

figure;
plot(time_stamps_store(1:m1), P_trace_store(1:m1), 'LineWidth', 2);
title('Problem 6 $\mathrm{tr}
(P)$', 'Interpreter', 'latex', 'FontSize', 18)


disp('The resultant angles correctly discribe the devices movement and
 match ')
disp('those found in step 3.')
```

Function definitions

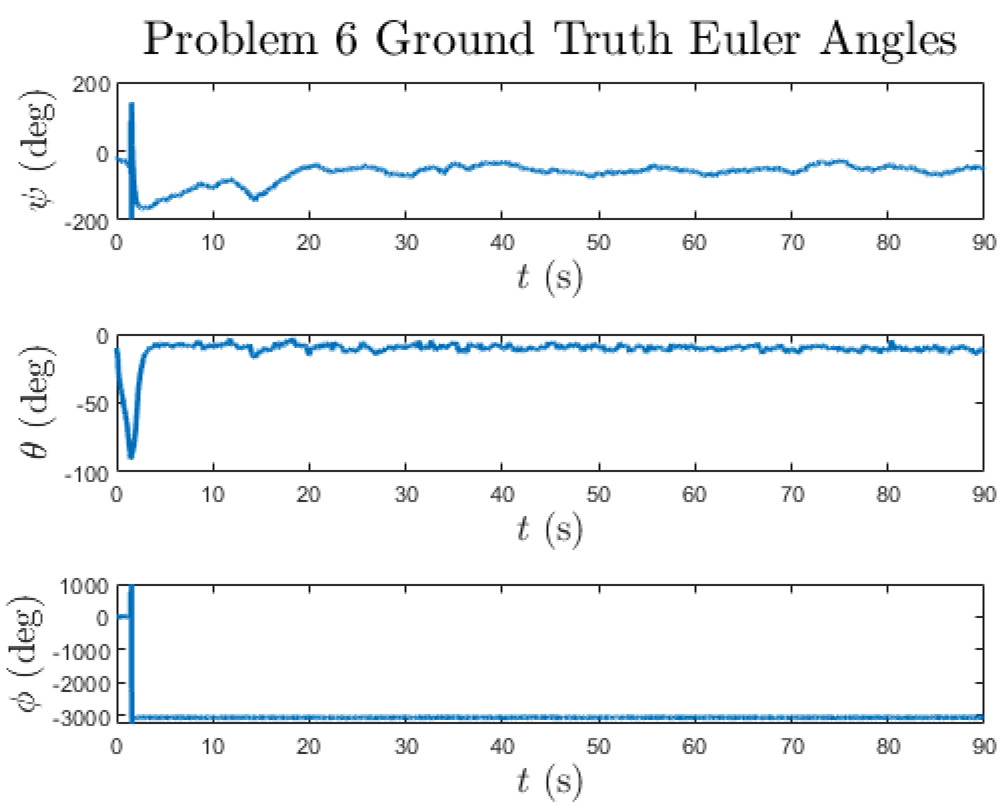
```
function x_dot = attitude_kinematics_asg3(x_, u_)
```

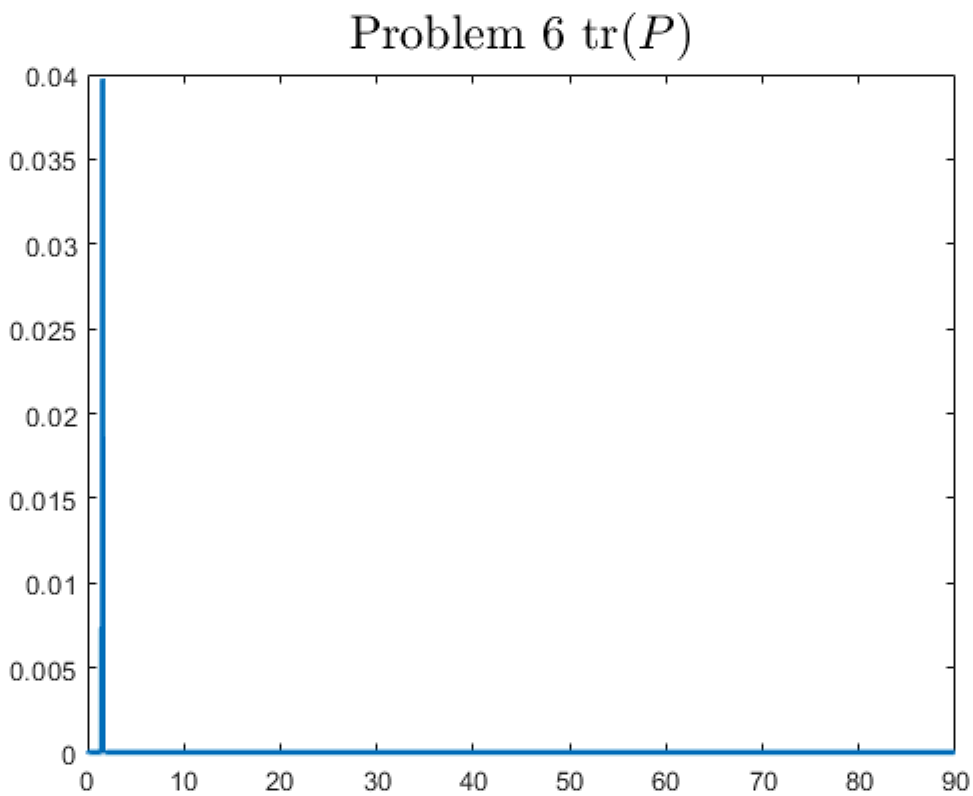
```
 theta_ = x_(2);
 phi_ = x_(3);

 x_dot = [-sin(theta_) 0 1; ...
    sin(phi_)*cos(theta_) cos(phi_) 0; ...
    cos(phi_)*cos(theta_) -sin(phi_) 0] \ u_;
end

function z_ = attitude_measurement_asg3(x_, u_, V)
 psi_ = x_(1);
 theta_ = x_(2);
 phi_ = x_(3);
 z_  = [ V*[...
  0 sin(theta_) 0; -sin(theta_) 0 cos(theta_); 0 -cos(theta_) 0]*u_
 + ...
  9.81*[sin(theta_); -cos(theta_)*sin(phi_); -cos(theta_)*cos(phi_)];
 psi_];
end
```

The resultant angles correctly discribe the devices movement and match those found in step 3.





Published with MATLAB® R2020a