Table of Contents

PROBLEM 4

```
clear variables; close all; clc
```

Data Fetch

```
data_table_acc = readtable('Accelerometer_Att_est.csv');

data_table_gyro = readtable('Gyroscope_Att_est.csv');

data_table_mag = readtable('Magnetometer_Att_est.csv');

% 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];
%==========================================================================
```

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 )
disp('rad')
initial_pitch = mean( pitch_data )
disp('rad')


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)
disp('rad')
```

```
initial_roll =

    0.0248

rad

initial_pitch =
```

```
   -0.1978

rad

initial_yaw =

   -1.2761

rad
```

Final Pitch, Roll, Yaw

```
Final_few_pts = (1185:1194);
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 )
disp('rad')
final_pitch = mean( pitch_data_f )
disp('rad')

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)
disp('rad')
```

```
final_roll =

   -0.1675

rad

final_pitch =
```

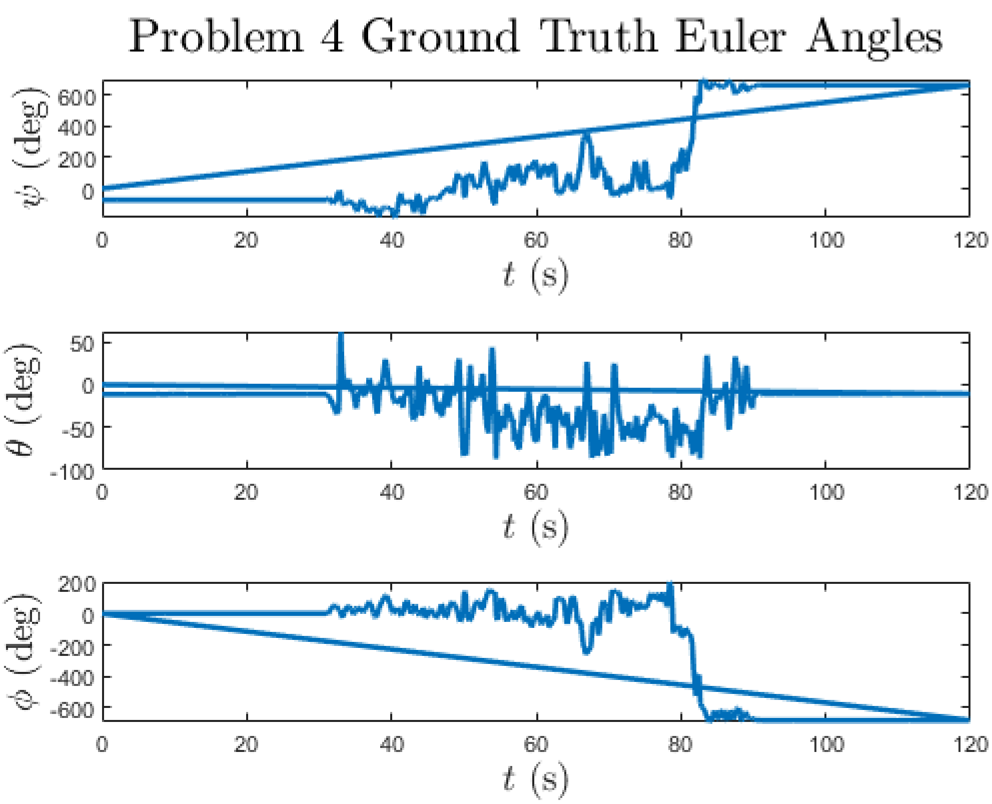
```
   -0.2021

rad

final_yaw =

   -1.6215

rad
```

RK4 Integration

```
dt = 0.001;

t = max([time_stamps_acc(1), time_stamps_mag(1),
 time_stamps_gyro(1)]);
x_t = [initial_yaw; initial_pitch; initial_roll];

tfinal = min([time_stamps_acc(end), time_stamps_mag(end),
 time_stamps_gyro(end)]);

n_time_pts = round( tfinal /dt );
t_ground_truth_store = zeros(1, n_time_pts);
x_ground_truth_store = zeros(3, n_time_pts);
t_ground_truth_store(1, 1) = t;
x_ground_truth_store(:, 1) = x_t;

column_number = 1;
while (t < tfinal)
 col_gyro = find(time_stamps_gyro <= t, 1, 'last');
%  col_acc = find(time_stamp_acc <= t, 1, 'last');
%  col_mag = find(time_stamp_mag <= t, 1, 'last');

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

 k1  = dt*attitude_kinematics_asg3(x_t, u_t);
 k2  = dt*attitude_kinematics_asg3((x_t + 0.5*k1), u_t);
 k3  = dt*attitude_kinematics_asg3((x_t + 0.5*k2), u_t);
 k4  = dt*attitude_kinematics_asg3((x_t + k3),  u_t);
 x_tplusdt = x_t + (1/6)*k1 + (1/3)*k2 + (1/3)*k3 + (1/6)*k4;

 column_number = column_number + 1;
 t = t + dt;  % New time
 x_t = x_tplusdt; % x_t is the state at time t

 t_ground_truth_store(1, column_number) = t;
 x_ground_truth_store(:, column_number) = x_t;

end

figure;
```

```
subplot(311)
plot(t_ground_truth_store(1, :), x_ground_truth_store(1, :)*180/
pi, 'LineWidth', 2);
ylabel('$\psi$ (deg)', 'Interpreter', 'latex', 'FontSize', 14)
xlabel('$t$ (s)', 'Interpreter', 'latex', 'FontSize', 14)
title('Problem 4 Ground Truth Euler
 Angles', 'Interpreter', 'latex', 'FontSize', 18)

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

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


disp('The result gathered is a valid result as it matches the devices
 orientation. It is')
disp('consistent with results gathered in step 4 with the only
 inconsistancy being the large')
disp('angle measurments due to full rotations during movement')

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
```

The result gathered is a valid result as it matches the devices orientation. It is
consistent with results gathered in step 4 with the only inconsistancy being the large
angle measurments due to full rotations during movement



Published with MATLAB® R2020a