MCEN 5228: Advanced Dynamics
HW2

David Akre

February 11, 2025

1 Question 1

1.1 Part (a)

We are asked in the problem to read in the .mat file which has a full state representaiton of a UAV polled from an IMU sensor operating at 500Hz and take the $\vec{w}^{F\to B}$ angular velocity and $\vec{v}^{G}$ translational velocity measurements, perform the appropriate transformations, and integrate to obtain position $\vec{r}^{G}(t)$ and attitude $\Theta(t)$. The matlab code below shows my approach to this part of the problem.

```matlab
% Author: David Akre
% Date: 2/1/2025

clear all;

% Euler Angle sequence is 3-2-1 and Fixed Frame is NED

% Load and setup variables for HW2
load("quadrotor.mat");

time = quadrotor(:, 1);
x = quadrotor(:, 2);
y = quadrotor(:, 3);
z = quadrotor(:, 4);
u = quadrotor(:, 5);
v = quadrotor(:, 6);
w = quadrotor(:, 7);
phi = quadrotor(:, 8);
theta = quadrotor(:, 9);
psi = quadrotor(:, 10);
p = quadrotor(:, 11);
q = quadrotor(:, 12);
r = quadrotor(:, 13);

N = length(phi);
dt = 0.002;

% Part (a) Angular velocity in body frame to Euler Rates + Navigation
    EQs
E_Theta = @(phi, theta)[
    1 sin(phi)*tan(theta) cos(phi)*tan(theta);
    0 cos(phi) -sin(phi);
    0 sin(phi)/cos(theta) cos(phi)/cos(theta)
];

```

```matlab
R_BF = @(phi, theta, psi) [cos(psi)*cos(theta) sin(psi)*cos(theta) -sin(theta);...
    cos(psi)*sin(theta)*sin(phi)-sin(psi)*cos(phi) cos(psi)*cos(phi)+...
    sin(psi)*sin(theta)*cos(phi) cos(theta)*sin(phi);...
    sin(psi)*sin(phi)+cos(psi)*sin(theta)*sin(phi)...
    sin(psi)*sin(theta)*cos(psi)-cos(psi)*sin(phi) cos(theta)*cos(phi)];

attitude = zeros(N, 3);
attitude(1, 1) = phi(1);
attitude(1, 2) = theta(1);
attitude(1, 3) = psi(1);

position = zeros(N, 3);
position(1, 1) = x(1);
position(1, 2) = y(1);
position(1, 3) = z(1);

for i = 2:N
    % [\dot{\phi} \dot{\theta} \dot{psi}]^T = E(\Theta) [p q r]^T
    omega = [p(i-1) q(i-1) r(i-1)];
    euler_rates = E_Theta(attitude(i-1, 1), attitude(i-1, 2)) * omega';

    attitude(i, :) = attitude(i-1, :) + euler_rates' * dt;

    % [\dot{x} \dot{y} \dot{z}]^T = R_FB [u v w]^T
    R_FB = R_BF(attitude(i-1, 1), attitude(i-1, 2), attitude(i-1, 3))';

    vel_GB = [u(i-1) v(i-1) w(i-1)]';
    vel_GF = R_FB * vel_GB;

    position(i, :) = position(i-1, :) + vel_GF' * dt;
end

plot_euler_angles(time, phi, theta, psi, attitude);

plot_position(time, x, y, z, position);
disp('Difference between estimated and ground truth position')
error_x = norm(x - position(:, 1))
error_y = norm(y - position(:, 2))
error_z = norm(z - position(:, 3))

% (dakre) So we can see that the euler angles are very aligned however
% when performing explicit euler integration we see positional drift
% that accumulates over time.
```

As noted in the bottom of the matlab script, there is a drift that accumulates over time when performing an explicit euler integration step.

```
Difference between estimated and ground truth position

error_x =

    3.8909


error_y =
    3.3421


error_z =

   2.3990e-12
```

The next code segments display how I am plotting the position (m) and attitude (rad) over time.

```matlab
% MCEN 5228: Advanced Dynamics HW2 - plot position
% Author: David Akre
% Date: 2/10/2025

function plot_position(time, x, y, z, position)
    figure;

    subplot(3,1,1)
    plot(time(:, 1), x, 'LineWidth', 2);
    hold on;
    plot(time(:, 1), position(:, 1), 'r', 'LineWidth', 2, 'LineStyle',
        '--');
    hold off;
    xlabel("Time (s)")
    ylabel("x (m)")
    legend("x_{gt}(t)", "x_{int}(t)")

    subplot(3,1,2)
    plot(time(:, 1), y, 'LineWidth', 2);
    hold on;
    plot(time(:, 1), position(:, 2), 'r', 'LineWidth', 2, 'LineStyle',
        '--');
    hold off;
    xlabel("Time (s)")
    ylabel("y (m)")
    legend("y_{gt}(t)", "y_{int}(t)")

    subplot(3,1,3)
    plot(time(:, 1), z, 'LineWidth', 2);
    hold on;
    plot(time(:, 1), position(:, 3), 'r', 'LineWidth', 2, 'LineStyle',
        '--');
    hold off;
    xlabel("Time (s)")
    ylabel("z (m)")
    legend("z_{gt}(t)", "z_{int}(t)")
end
```

```matlab
% MCEN 5228: Advanced Dynamics HW2 - plot euler angles
% Author: David Akre
% Date: 2/10/2025

function plot_euler_angles(time, phi, theta, psi, attitude)
    figure;

    subplot(3,1,1)
    plot(time(:, 1), rad2deg(phi), 'LineWidth', 2);
    hold on;
    plot(time(:, 1), rad2deg(attitude(:, 1)), 'r', 'LineWidth', 2, '
        LineStyle', '--');
    hold off;
    xlabel("Time (s)")
    ylabel("\phi (deg)")
    legend("\phi_{gt}(t)", "\phi_{int}(t)")

    subplot(3,1,2)
    plot(time(:, 1), rad2deg(theta), 'LineWidth', 2);
    hold on;
    plot(time(:, 1), rad2deg(attitude(:, 2)), 'r', 'LineWidth', 2, 'LineStyle', '--');
    hold off;
    xlabel("Time (s)")
    ylabel("\theta (deg)")
    legend("\theta_{gt}(t)", "\theta_{int}(t)")

    subplot(3,1,3)
    plot(time(:, 1), rad2deg(psi), 'LineWidth', 2);
    hold on;
    plot(time(:, 1), rad2deg(attitude(:, 3)), 'r', 'LineWidth', 2, 'LineStyle', '--');
    hold off;
    xlabel("Time (s)")
    ylabel("\psi (deg)")
    legend("\psi_{gt}(t)", "\psi_{int}(t)")
end
```

Here are the resulting plots (position and attitude integration vs. ground truth).

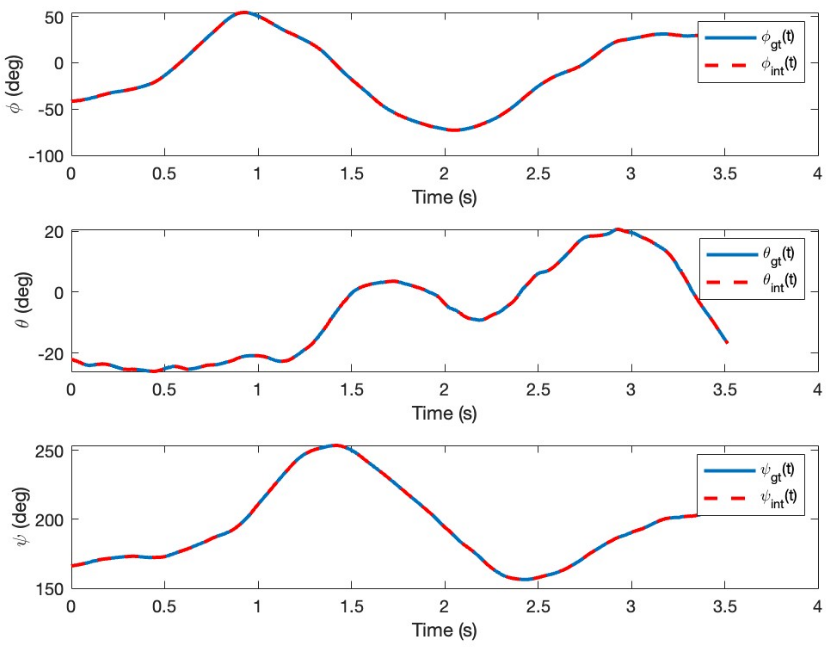

Figure 1: Euler Angle Integration vs. Ground Truth

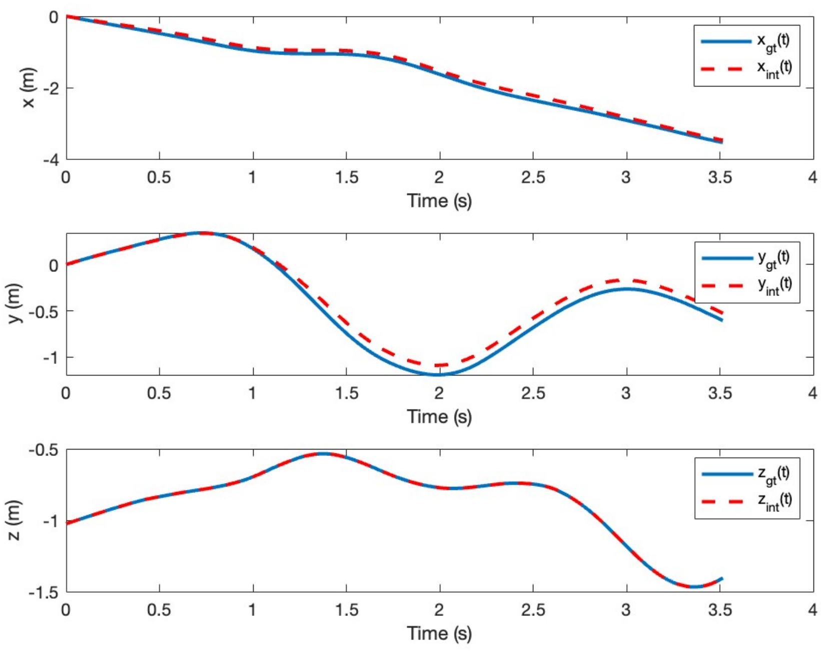

Figure 2: Position Integration vs. Ground Truth

The drift is really an artifact of the integration method chosen. To help rectify this, we can perform an RK4 style integration to better improve the results as shown below.

```
% Author: David Akre
% Date: 2/1/2025

clear all;

% Euler Angle sequence is 3-2-1 and Fixed Frame is NED

% Load and setup variables for HW2
load("quadrotor.mat");

time = quadrotor(:, 1);
x = quadrotor(:, 2);
y = quadrotor(:, 3);
z = quadrotor(:, 4);
u = quadrotor(:, 5);
v = quadrotor(:, 6);
w = quadrotor(:, 7);
phi = quadrotor(:, 8);
theta = quadrotor(:, 9);
psi = quadrotor(:, 10);
p = quadrotor(:, 11);
q = quadrotor(:, 12);
r = quadrotor(:, 13);

N = length(phi);
dt = 0.002;

% Part (a) Angular velocity in body frame to Euler Rates + Navigation
  EQs
```

```matlab
function R = E_Theta(phi, theta)
    R = [1 sin(phi)*tan(theta) cos(phi)*tan(theta);...
    0 cos(phi) -sin(phi);...
    0 sin(phi)/cos(theta) cos(phi)/cos(theta)];
end

function R = R_BF(phi, theta, psi)
    R = [cos(psi)*cos(theta) sin(psi)*cos(theta) -sin(theta);...
    cos(psi)*sin(theta)*sin(phi)-sin(psi)*cos(phi) cos(psi)*cos(phi)+...
    sin(psi)*sin(theta)*cos(phi) cos(theta)*sin(phi);...
    sin(psi)*sin(phi)+cos(psi)*sin(theta)*sin(phi)...
    sin(psi)*sin(theta)*cos(psi)-cos(psi)*sin(phi) cos(theta)*cos(phi)
       ];
end

function xdot = f(state)
    %% [\dot{\phi} \dot{\theta} \dot{psi}]^T = E(\Theta) [p q r]^T
    omega = [state(10) state(11) state(12)];
    euler_rates = E_Theta(state(7), state(8)) * omega';

    %% [\dot{x} \dot{y} \dot{z}]^T = R_FB [u v w]^T
    R_FB = R_BF(state(7), state(8), state(9))';
    vel_GB = [state(4) state(5) state(6)]';
    vel_GF = R_FB * vel_GB;

    xdot = zeros(6, 1);
    xdot(1:3) = vel_GF;
    xdot(4:6) = euler_rates;
end

function xn = rk4_step(xk, h)
    f1 = f(xk);
    f2 = f(xk + 0.5*h*f1);
    f3 = f(xk + 0.5*h*f2);
    f4 = f(xk + h*f3);

    xn = xk(1:6)' + (h/6.0).*(f1 + 2*f2 + 2*f3 + f4);
end

state = zeros(N, 6);
state(1, 1) = x(1);
state(1, 2) = y(1);
state(1, 3) = z(1);
state(1, 4) = phi(1);
state(1, 5) = theta(1);
state(1, 6) = psi(1);

for i = 2:N
    xk = quadrotor(i-1, 2:end);
    xn = rk4_step(xk, dt);
    state(i, :) = xn;
end

plot_position(time, x, y, z, state(:, 1:3));
disp('Difference between estimated and ground truth position')
error_x = norm(x - state(:, 1))
error_y = norm(y - state(:, 2))
error_z = norm(z - state(:, 3))
```

```
    Difference between estimated and ground truth position

error_x =

    0.1738


error_y =

    0.1591


error_z =

    0.0978
```

We still have some drift but it is less magnified now which is an improvement. Furthermore, other methods like implicit midpoint could also help reduce drift as well. The plot below shows a much better alignment between integration and ground truth positions.

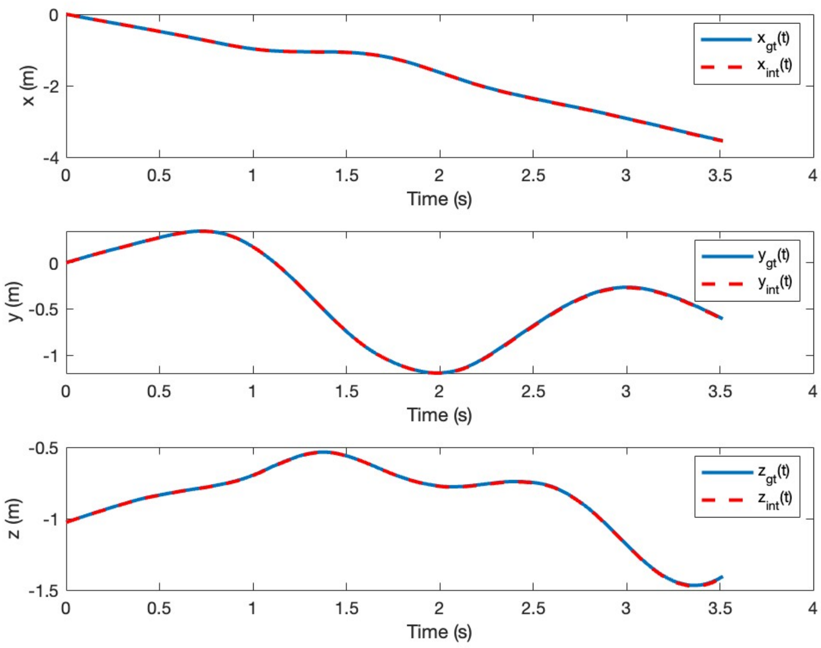

Figure 3: Position Integration vs. Ground Truth (RK4)

1.2 Part (b)

In this part we are asked to perform the same integration except using quaternions now (note the IMU measurements are still the same). The below matlab code snippet shows my approach for this.

```
% Part (b) Quaternion rates + Navigation EQs
quaternion = @(phi, theta, psi) [cos(psi/2)*cos(theta/2)*...
    cos(phi/2)+sin(psi/2)*sin(theta/2)*sin(phi/2) ...
    cos(psi/2)*cos(theta/2)*sin(phi/2)-sin(psi/2)*sin(theta/2)*cos(phi
       /2) ...
    cos(psi/2)*sin(theta/2)*cos(phi/2)+sin(psi/2)*cos(theta/2)*sin(phi
       /2) ...
    sin(psi/2)*cos(theta/2)*cos(phi/2)-cos(psi/2)*sin(theta/2)*sin(phi
       /2)];

q_x_w = @(p, q, r) 0.5 * [0 -p -q -r; p 0 r -q; q -r 0 p; r q -p 0];

quat_to_321 = @(quat) [
    acos((quat(1)^2-quat(2)^2-quat(3)^2+quat(4)^2)/(sqrt(1-4*(quat(2)*
       quat(4)-quat(1)*quat(3))^2)))*sign(2*(quat(3)*quat(4)+quat(1)*
       quat(2)))...
    asin(-2*(quat(2)*quat(4)-quat(1)*quat(3))) ...
    acos((quat(1)^2+quat(2)^2-quat(3)^2-quat(4)^2)/sqrt(1-4*(quat(2)*
       quat(4)-quat(1)*quat(3))^2))*sign(2*(quat(2)*quat(3)+quat(1)*
       quat(4)))
];

quat_gt = zeros(N, 4);
quat_gt(1, :) = quaternion(phi(1), theta(1), psi(1));

quat = zeros(N, 4);
quat(1, :) = quat_gt(1, :);

position2 = zeros(N, 3);
position2(1, 1) = x(1);
position2(1, 2) = y(1);
position2(1, 3) = z(1);

for i = 2:N
    quat_gt(i, :) = quaternion(phi(i), theta(i), psi(i));
    quat_dt = q_x_w(p(i), q(i), r(i)) * quat(i-1, :)';

    quat(i, :) = quat(i-1, :) + quat_dt' * dt;
    quat(i, :) = quat(i, :) / norm(quat(i, :));

    euler_angles = quat_to_321(quat(i-1, :));
    R_FB = R_BF(euler_angles(1), euler_angles(2), euler_angles(3))';

    vel_GB = [u(i-1) v(i-1) w(i-1)]';
    vel_GF = R_FB * vel_GB;

    position2(i, :) = position2(i-1, :) + vel_GF' * dt;
end

plot_quaternion(time, quat_gt, quat);

plot_position(time, x, y, z, position2);
disp('Difference between estimated and ground truth position')
error_x = norm(x - position2(:, 1))
error_y = norm(y - position2(:, 2))
error_z = norm(z - position2(:, 3))
```

Note, for this I am using the explicit euler integration so there is still drift but the above RK4 or implicit midpoint integration methods will aid in positional improvements as shown in part (a) above.

```
    Difference between estimated and ground truth position

error_x =

    3.8425
```

```
error_y =

    3.4988


error_z =

    0.2182
```

The plotting matlab code for quaternions is shown here:

```
% MCEN 5228: Advanced Dynamics HW2 - plot quaternion
% Author: David Akre
% Date: 2/10/2025

function plot_quaternion(time, quaternion, attitude)
    figure;

    subplot(4,1,1)
    plot(time(:, 1), quaternion(:, 1), 'LineWidth', 2);
    hold on;
    plot(time(:, 1), attitude(:, 1), 'r', 'LineWidth', 2, 'LineStyle',
        '--');
    hold off;
    xlabel("Time (s)")
    ylabel("q0")
    legend("q0_{gt}(t)", "q0_{int}(t)")

    subplot(4,1,2)
    plot(time(:, 1), quaternion(:, 2), 'LineWidth', 2);
    hold on;
    plot(time(:, 1), attitude(:, 2), 'r', 'LineWidth', 2, 'LineStyle',
        '--');
    hold off;
    xlabel("Time (s)")
    ylabel("q1")
    legend("q1_{gt}(t)", "q1_{int}(t)")

    subplot(4,1,3)
    plot(time(:, 1), quaternion(:, 3), 'LineWidth', 2);
    hold on;
    plot(time(:, 1), attitude(:, 3), 'r', 'LineWidth', 2, 'LineStyle',
        '--');
    hold off;
    xlabel("Time (s)")
    ylabel("q2")
    legend("q2_{gt}(t)", "q2_{int}(t)")

    subplot(4,1,4)
    plot(time(:, 1), quaternion(:, 4), 'LineWidth', 2);
    hold on;
    plot(time(:, 1), attitude(:, 4), 'r', 'LineWidth', 2, 'LineStyle',
        '--');
    hold off;
    xlabel("Time (s)")
    ylabel("q3")
    legend("q3_{gt}(t)", "q3_{int}(t)")
```

```
end
```

Below display the resulting plots for the four quaternion values and the position using integration vs. ground truth.

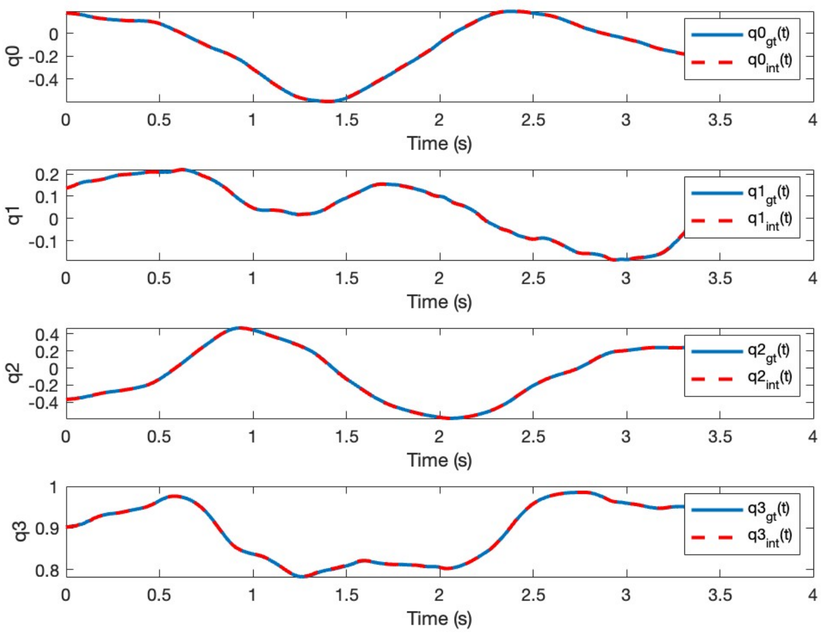

Figure 4: Quaternion Integration vs. Ground Truth

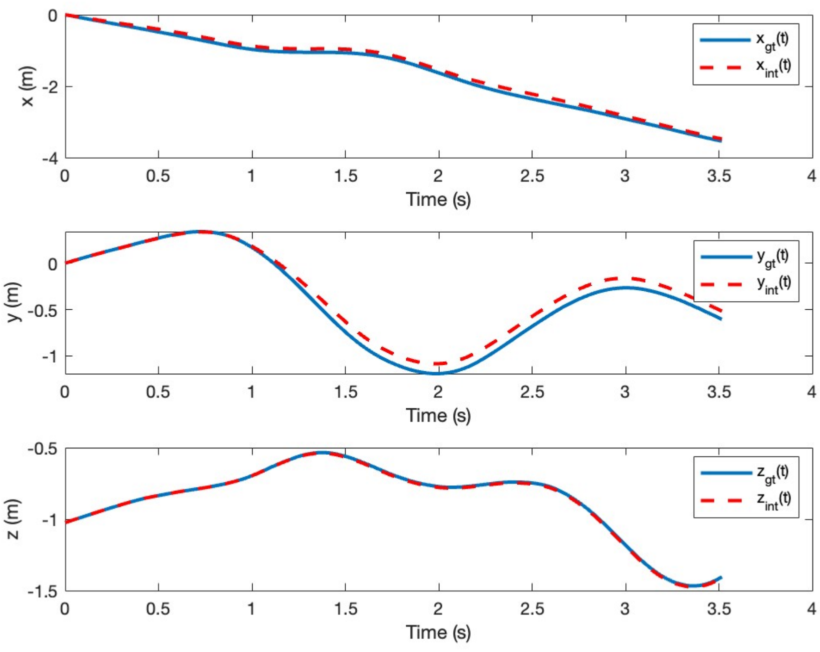

Figure 5: Position Integration vs. Ground Truth

Overall we can visually and quantitativelty see that the integration of euler angles and quaternions are spot on, and quaternions help avoid gimbal lock or singularities, whereas euler angles are susceptible to this phenomena. Likewise, the position can be acquired integrating the velocity components of the IMU measurements but the accuracy is subject to the integration method used. Explicit euler methods will tend to provide worse estimates of position over time vs. RK4 or some other higher fidelity integration scheme as shown in this HW.