Simon Honigmann

Sensor Orientation

November 7, 2018

Lab 5: Inertial Navigation in 2D / Realistic Signal

Simulated Inertial Navigation in 2D:

In this lab, the goal was to better understand how different sensor errors and uncertainties manifest themselves as localization errors. This will help develop an intuitive understanding of the importance of sensor measurement techniques, as well generating some understanding of how calibration and filtering might be able to improve localization results.

The first step in this experiment was to convert units of sensor parameters listed on the (simulated) sensor spec sheets, into workable SI units. The initial values, as well as the conversions, are included in the table below.

Table 1: Sensor Characteristics and Converted Units of Stochastic Error Characteristics

Error Type	**Notation**	**Stochastic Value**		**Note**
		Provided Units	**Value (SI)**	
Gyro bias (random constant)	b_G	$10\ \mathrm{deg/h}$	$4.848 \times 10^{-5}\ rad/s$	$1\ \sigma$
Gyro correlated noise (1st order Gauss-Markov)	$\sigma_{G_{GM1}^{PSD}}$	$0.005\frac{deg}{s}/\sqrt{Hz}$	$8.727 \times 10^{-5}\ rad/s/\sqrt{Hz}$	PSD level (scale for simulation!)
	$1/\beta_G$	$100\ s$	$100\ s$	correlation time
Gyro random walk (white noise)	$\sigma_{G_{WN}^{PSD}}$	$0.10\ deg/\sqrt{h}$	$2.909 \times 10^{-4}\ rad/s/sample$	PSD level
Accelerometer bias (random constant)	b_A	$1\ mg$	$9.81 \times 10^{-3}\ m/s^2$	$1\ \sigma$
Accelerometer noise (white)	$\sigma_{A_{WN}^{PSD}}$	$50\ \mu g/\sqrt{Hz}$	$4.905 \times 10^{-3}\ m/s^2/sample$	PSD level

With these values, it is possible to generate simulated sensor error for the gyro and accelerometer. The gyro can have any combination of bias, correlated noise, white noise, or none of the above. The accelerometer can have any combination of bias, white noise, or none of the above. The different errors, and their net result are plotted below.

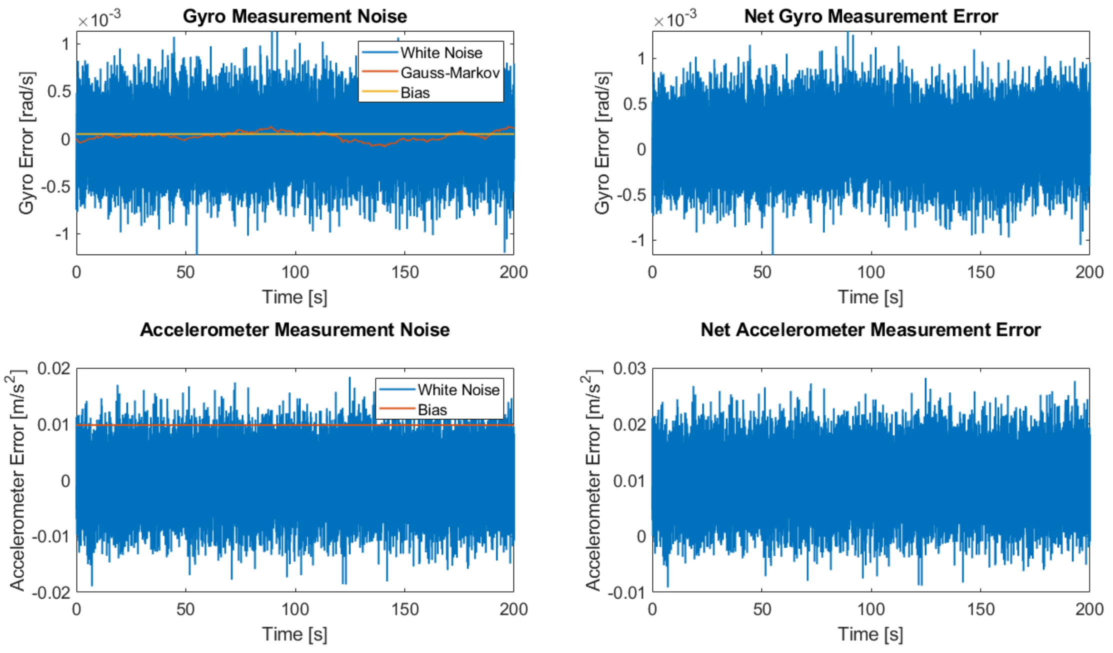

Figure 1: Generated Gyro and Accelerometer Error Components (Left) and Net Error (Right)

To ensure lab results were consistent, fixed seeds were used to generate all error types. Then, to better identify the affect each type of error has on the localization error, the different errors were isolated, and the results were compared. While plots were generated for every combination listed in Table 2 below, only a select few were included for the sake of brevity. Plots for each individual error type and one plot of every error combined are included below. Figure 2 shows the estimation errors when no error is added to the sensor signals and provides a baseline against which other plots can be compared.

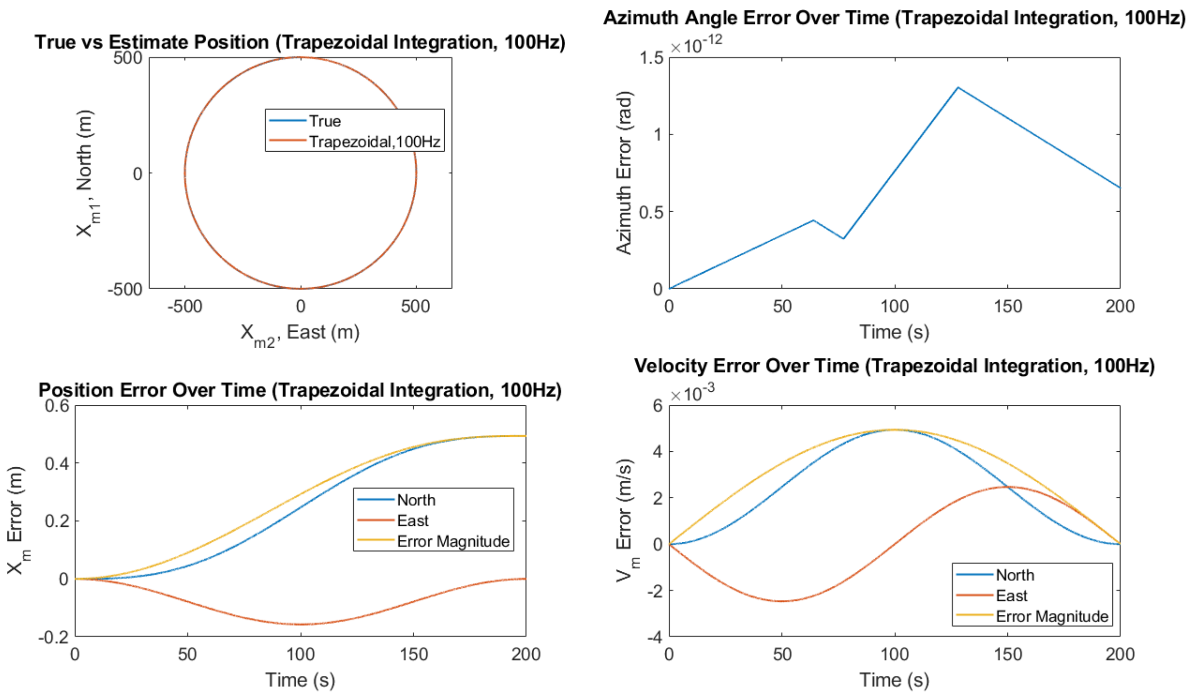

Figure 2: Vehicle Position Estimation with Ideal Sensor

Figure 3 shows the estimation error for a constant bias gyro error. As anticipated, the Azimuth error grows linearly over time as the estimation method integrates the gyro signal. Position error also grows in time with the azimuth error.

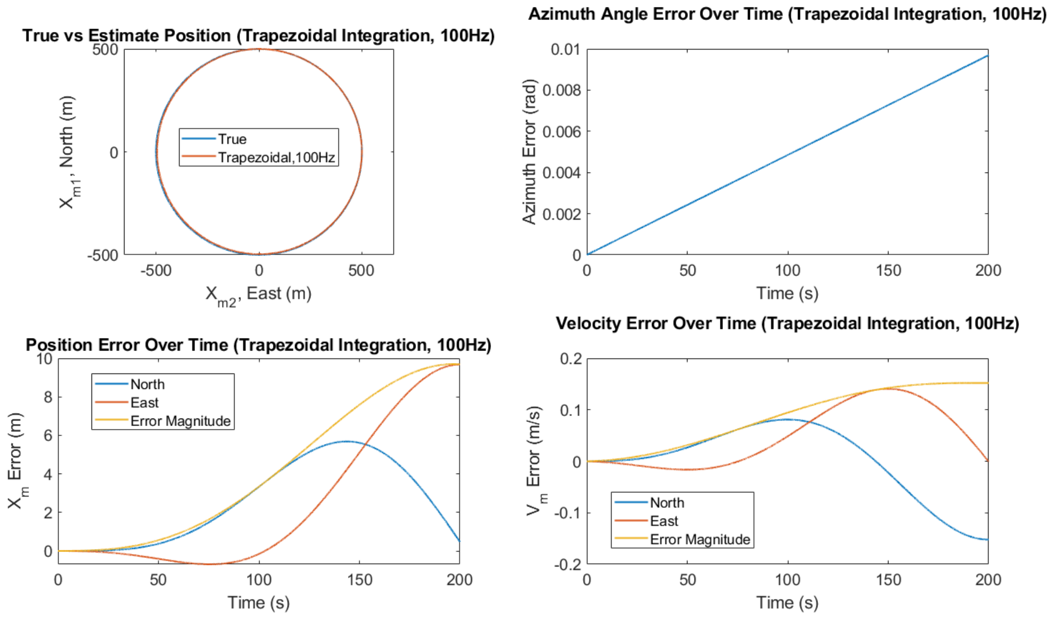

Figure 3: Estimation Error with Gyro Bias Only

Figure 4 below shows the estimation error for a gyro with only a Gauss-Markov error process. The GM process also results in a growing azimuth and position error, however the error is far less predictable than in the constant bias case.

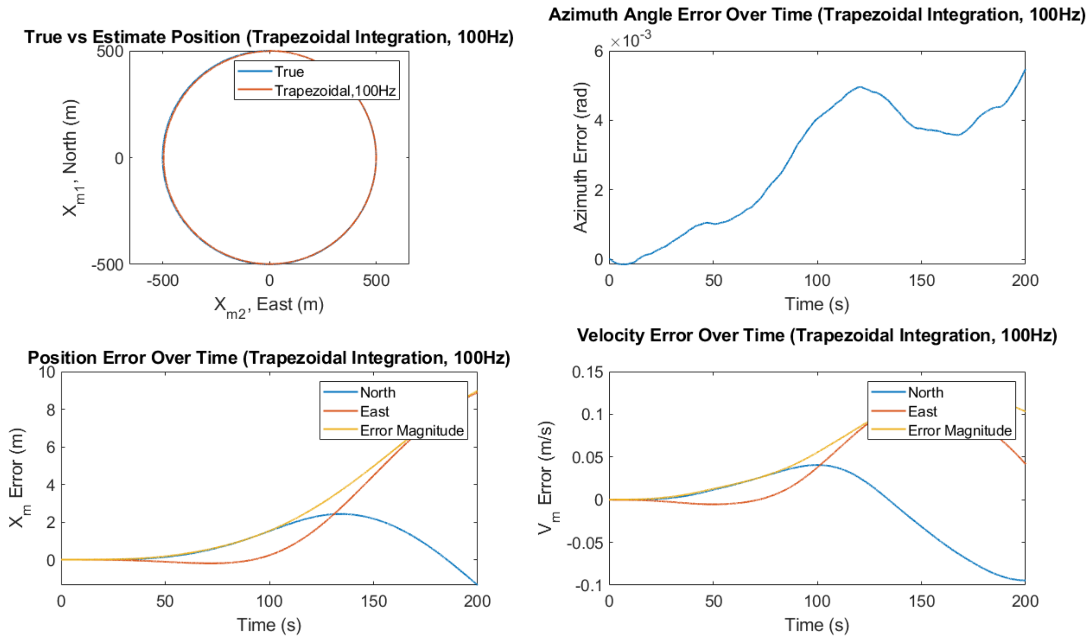

Figure 4: Estimation Error with Gyro Gauss-Markov Process Only

Figure 5 shows the estimation error found with only a white noise error process on the gyro measurements. The azimuth error gains no bias over time, though the position does drift slightly. The errors caused by white noise are an order of magnitude smaller than those from the GM or bias errors.

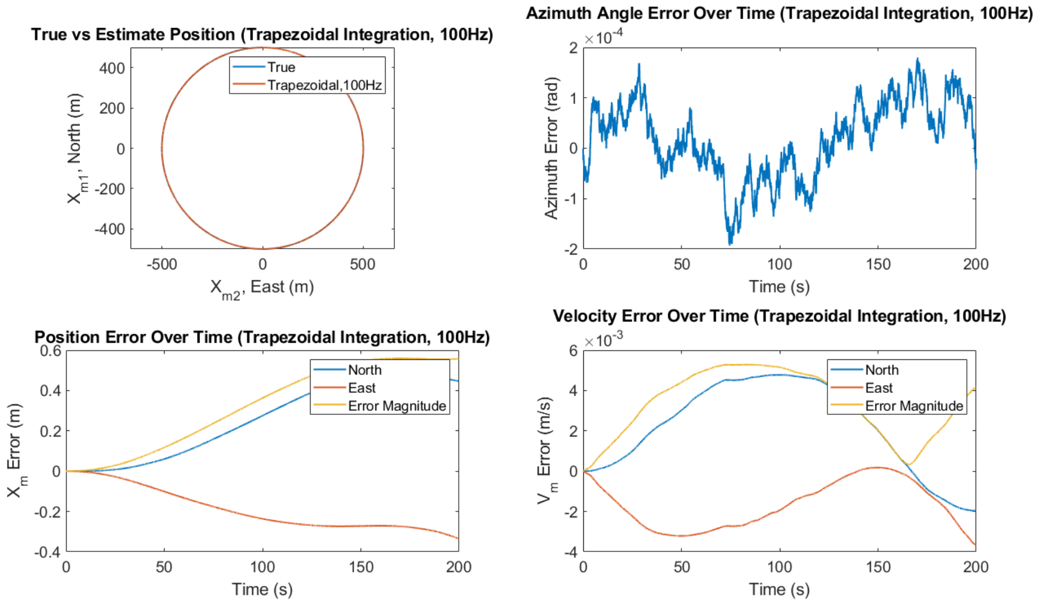

Figure 5: Estimation Error with Gyro White Noise Only

Figure 6 shows the estimation error when the accelerometer bias error is applied. This error has no impact on the azimuth estimation but has the largest impact on velocity and position estimations.

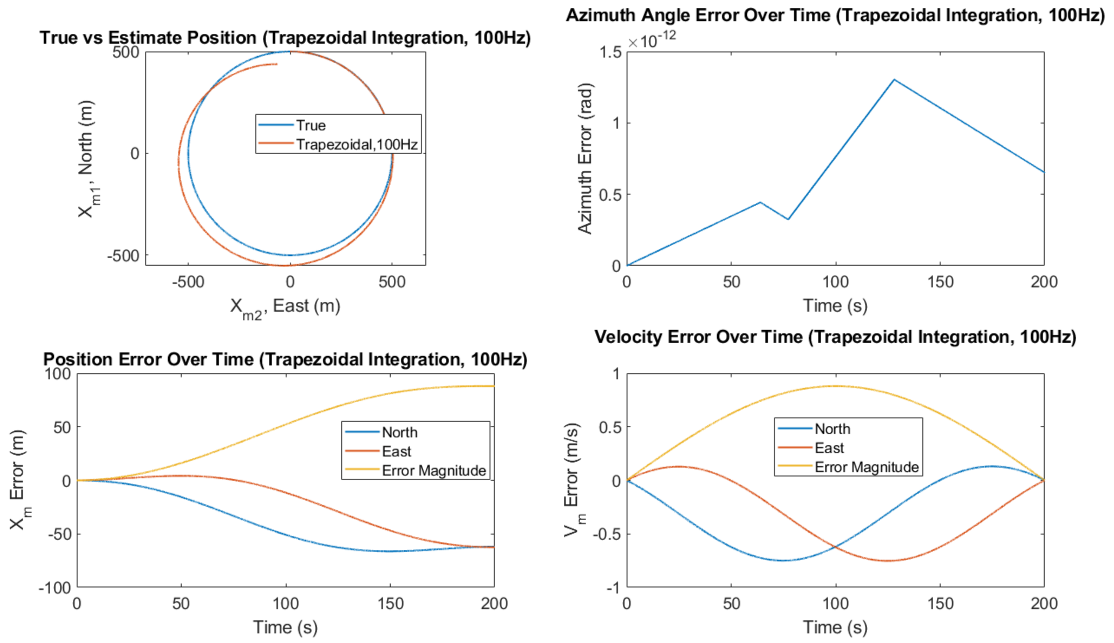

Figure 6: Estimation Error with Accelerometer Bias Only

Figure 7 shows the estimation error when only the accelerometer white noise is added. Again, there is no impact on the azimuth error. However, a form of a random walk can be seen in the velocity plot, and the position error grows steadily as well, as the accelerometer measurements are double-integrated.

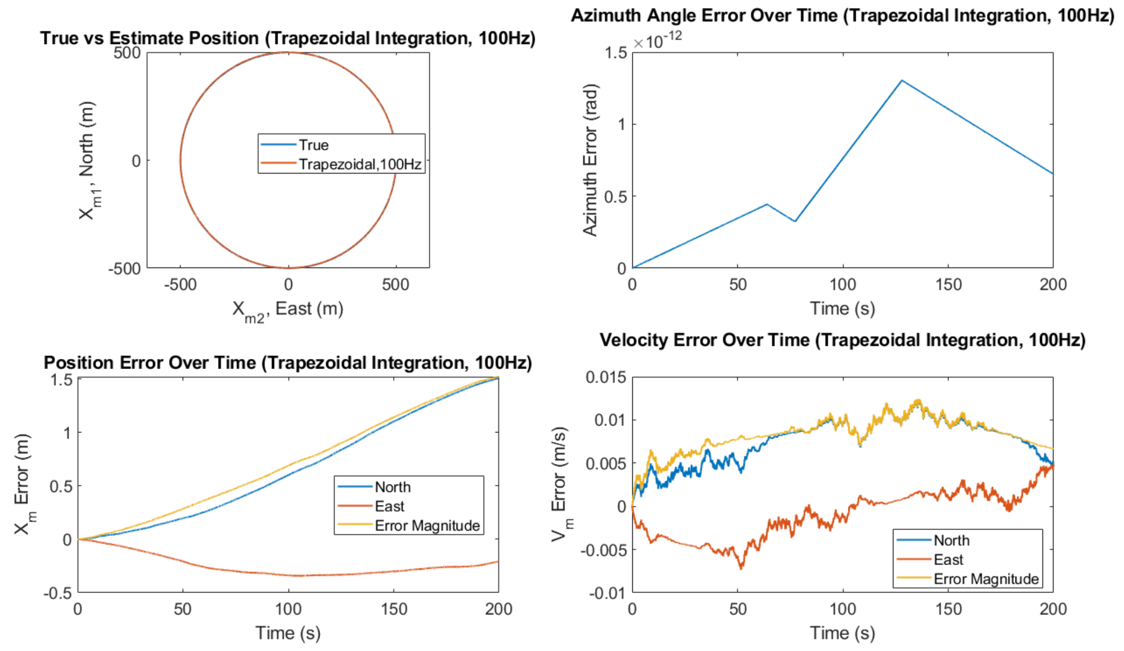

Figure 7: Estimation Error with Accelerometer White Noise Only

Finally, Figure 8 shows the estimation errors when all sensor errors are added simultaneously. Interestingly, the majority of this error can be accounted for by only the accelerometer and gyro biases, and perhaps the gyro GM process as well.

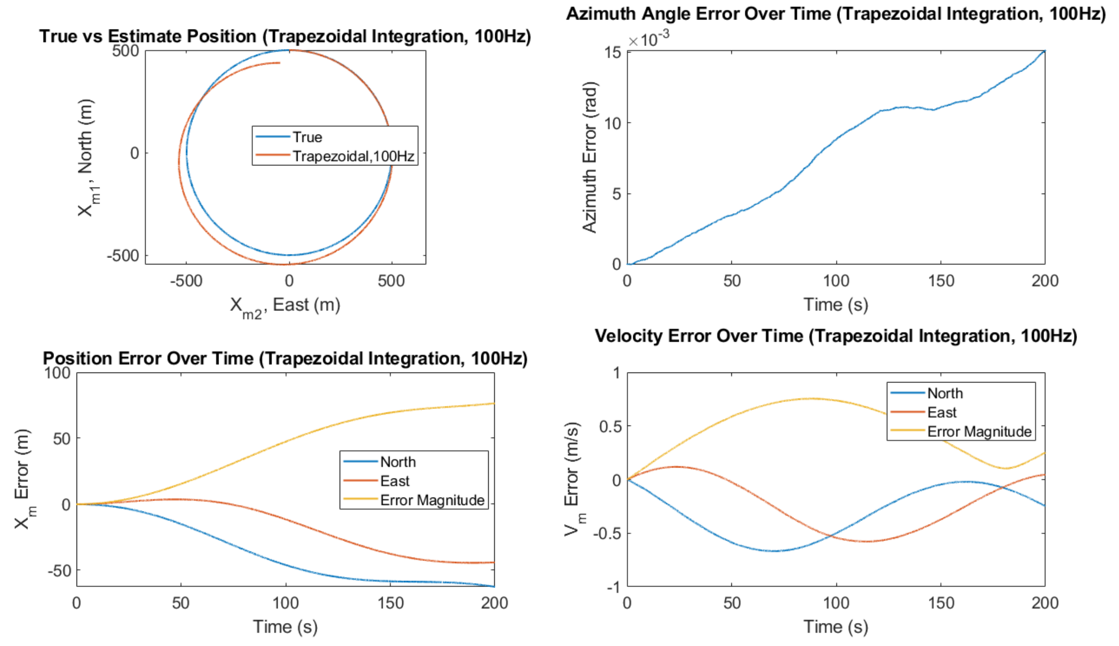

Figure 8: Estimation Error with All Error Types Present

Average and maximum errors for each sensor error combination were recorded, as well as any notable features of the simulated localization. To improve readability, the entries were given a color scale. Green entries indicate minimal impact on the state estimate, while red entries indicate a significant impact on state estimate.

Table 2: Inertial Navigation Error for Different Sensor Error Combinations

Error Types Present	**Azimuth Error (°)**		**Position Error (m)**		**Velocity Error (m/s)**	
	Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
No Noise	6.55E-13	1.30E-12	0.27	0.49	0.003	0.005
Only Gyro Bias	4.85E-03	9.70E-03	4.09	9.70	0.086	0.152
Only Gyro Gauss-Markov	2.79E-03	5.47E-03	2.73	8.97	0.060	0.121
Only Gyro White Noise	6.46E-05	1.93E-04	0.33	0.56	0.003	0.005
Gyro GM + WN	2.81E-03	5.45E-03	2.66	8.65	0.059	0.119
Gyro Bias + GM + WN	7.64E-03	1.51E-02	6.54	18.24	0.142	0.268
Only Accelerometer Bias	6.55E-13	1.30E-12	48.38	87.97	0.560	0.880
Only Accelerometer White Noise	6.55E-13	1.30E-12	0.72	1.52	0.008	0.012
Accelerometer Bias + WN	6.55E-13	1.30E-12	48.09	87.43	0.559	0.877
All Noise Types	7.64E-03	1.51E-02	42.47	76.54	0.452	0.754

Questions:

From the conducted simulations:

1. Which from the three error sources influences the estimate of azimuth the least?

As seen in Table 2, the accelerometer bias and white noise and the gyro white noise impact the azimuth error the least, by several orders of magnitude. It is to be anticipated that the accelerometer has zero impact on, as, in this case, its readings are completely unused for azimuth angle calculation. Perhaps a bit more surprising was that the white noise of the gyro resulted in azimuth error which was an order of magnitude lower than either other gyro error types despite having a magnitude about 10x greater than the GM process or the constant bias. However, since the azimuth angle is determined by integrating the gyro measurements, it makes sense that correlated noise and especially a constant bias will have a large impact on accumulated error.

2. Which error source influences velocity estimation the most?

The largest influence on velocity estimation comes from the accelerometer bias. As in the previous question, when integration is involved, a constant bias will more rapidly accumulate error than a process which is centered about zero. Accelerometer bias alone accounts for about 4 times more uncertainty than all gyro errors combined. Accelerometer white noise has a comparatively smaller impact on velocity estimation.

3. Which error source influences position estimation the most?

As with velocity error, position error is most heavily influenced by accelerometer bias. This comes as little surprise, as position estimates are taken by integrating velocity. So inherently, a large velocity error will result in a large position error. For a 500m radius circle, accelerometer bias alone can result in up to 88 m error, or nearly 18% error. This is more than 4 times the error caused by all gyro error sources combined and highlights the issue with double-integration for position localization.

Brief Concluding Remarks:

Ultimately, it is reassuring that the largest impact to estimation comes from the sensor error that is the easiest to remedy – constant bias. Integration of IMU sensors will invariably result in some drift of state estimation due to measurement errors and noise, and it is interesting to realize that an integrated constant (linear function) will diverge faster than integrated white noise (random walk). The results in this lab highlight the importance of identifying and, when possible, compensating for, sensor error. Simple initialization procedures can be implemented to zero a sensor and effectively eliminate constant bias error. White noise can be filtered from a sensor; however, filtering has implications to the stability and responsiveness of the controller and should be considered carefully for the application. Correlated noise correction will likely involve more advanced filtering techniques. The above results were calculated for a basic trajectory over a relatively short period of time. It is clear that real applications involving low quality sensors, longer time frames, or more complex navigation requirements will require more than a direct localization from IMU data.

Appendix A: MATLAB Code

Lab5.m

```
%% Simon Honigmann
% Sensor Orientation
% Lab 5: Inertial Navigation in 2D / Realistic Signal
% 11/02/18

%% Cleanup
clc;
clear;
close all;

%% Lab Formatting:
set(groot,'DefaultAxesFontSize',14);
set(groot,'DefaultLineLineWidth',1.5);

%% Definitions:
% m = 2D mapping frame (non-accelerating, non-rotating)
%       x_m1 = north
%       x_m2 = east
% m frame polar coordinates
%       psi = polar angle
%       r = radius

% b = body frame
        % alpha = azimuth (yaw) angle
        % x_b1 = tangential direction
        % x_b2 = radial direction (inwards)

%% Constants
r = 500; % Circle radius: 500 m, Angular speed ? = ?/100;
omega = pi/100; % angular speed = pi/100
g = 9.81; %gravity, m/s^2

%only for 100Hz and Trapezoidal
sampling_rate = 100;
method = 'Trapezoidal';

f = sampling_rate;
sample_rate_text = num2str(sampling_rate,3);
num_rotations = 1; %number of time the vehicle goes  around the circle. increase to better show divergence

%% Assumptions/Constraints
dr = 0; %constant radius of motion in m frame
ddr = 0;
dpsi = omega; %constant angular velocity in m frame
ddpsi = 0;

%% Initial Conditions
psi_0 = 0; % Initial position: on North axis
```

```
alpha_0 = psi_0 + pi/2;
x_0 = [r,0];
dx_0 = [0,omega*r]; % Initial velocity: north-axis: 0, east-axis: ? · radius

required_time = 2*num_rotations*pi/omega; %time for 1 full rotation [s]
t = (0:1/sampling_rate:required_time)'; %time vector, [s]
samples = (1:length(t))';

%% Noise Definitions
b_g_spec = 10; %gyro bias non-SI, deg/hr
b_g = b_g_spec*(pi/180)/(3600)*ones(length(t),1); %SI gyro bias, rad/s

sigma_g_gm_spec = 0.005; %spec'd gyro correlated noise (1st order GM); deg/s/sqrt(Hz)
sigma_g_gm_SI = sigma_g_gm_spec*pi/180; %gyro correlated noise std. dev, rad/s/sqrt(Hz)

T_g = 100; %gyro correlation period, s
beta_g = 1/T_g; %gyro correlation frequency, Hz

sigma_g_wn_spec = 0.1; %gyro white noise strenght, deg/sqrt(hr)
sigma_g_wn_SI = sigma_g_wn_spec*(pi*sqrt(f)/(180*60));

b_a_spec = 1/1000; % accelerometer bias, g
b_a = b_a_spec*g*ones(length(t),1); %accelerometer bias, m/s^2

sigma_a_wn_spec = 50e-6; %accelerometer white noise strength, g/sqrt(Hz)
sigma_a_wn_SI = sigma_a_wn_spec*g*sqrt(f); %accelerometer white noise strength, m/s^2/sqrt(Hz)

var_checks = [b_g(1);sigma_g_gm_SI;T_g;sigma_g_wn_SI;b_a(1);sigma_a_wn_SI];


%Scale PSD values of sigma -> (nvm... done directly in noise generation)
 sigma_g_gm = sigma_g_gm_SI;
 sigma_g_wn = sigma_g_wn_SI;
 sigma_a_wn = sigma_a_wn_SI;

%% Generating Stochastic Process Components

seed = 1234567;
n = length(t);

[n_g_wn,~] = whiteNoiseRandomWalk(n,seed); %generate white noise process
n_g_wn = n_g_wn*sigma_g_wn;%gyro white noise

[n_a_wn,~] = whiteNoiseRandomWalk(n,seed+1); %accelerometer white noise
n_a_wn = n_a_wn*sigma_a_wn;

[n_wn,~] = whiteNoiseRandomWalk(n,seed+2); %one final random white noise process

n_wn = n_wn*sqrt(sigma_g_gm^2*(1-exp(-2*beta_g*1/sampling_rate)));% eq 32

%this is probably wrong... would be worth double checking...
n_g_gm = firstOrderGaussMarkov(T_g,1/sampling_rate,n_wn); %1st order gauss markov process
[H,f]=pwelch(n_g_gm,[],[],[],sampling_rate);

%selected combinations of noise components to include:
errors = [0,0,0,0,0; ... no error
          1,0,0,0,0; ... gyro bias only
          0,1,0,0,0; ... gyro gauss markov only
          0,0,1,0,0; ... gyro white noise only
          0,1,1,0,0; ... gyro white noise and gauss markov
          1,1,1,0,0; ... all gyro errors
          0,0,0,1,0; ... accel bias only
          0,0,0,0,1; ... accel white noise only
          0,0,0,1,1; ... all accel errors
          1,1,1,1,1]; %  all errors combined
for k=1:10
    %loop through all desired iterations:
    has_g_bias = errors(k,1);
    has_a_bias = errors(k,4);
    has_g_wn = errors(k,3);
    has_a_wn = errors(k,5);
    has_g_gm = errors(k,2);
    % Net Noise
    n_g = has_g_bias*b_g + has_g_wn*n_g_wn + has_g_gm*n_g_gm; %combined error for gyro
    n_a = has_a_bias*b_a + has_a_wn*n_a_wn; % combined error for accelerometer

    % Plotting as sanity check
    figure(k+10);
    subplot(2,2,1);
    plot(t,n_g_wn*has_g_wn);
    hold on;
    plot(t,n_g_gm*has_g_gm);
    plot(t,b_g*has_g_bias);

    title('Gyro Measurement Noise');
    xlabel('Time [s]');
    ylabel('Gyro Error [rad/s]');
    legend('White Noise','Gauss-Markov','Bias');

    figure(k+10);
    subplot(2,2,2);
    plot(t,n_g);

    title('Net Gyro Measurement Error');
    xlabel('Time [s]');
    ylabel('Gyro Error [rad/s]');

    subplot(2,2,3);
    plot(t,n_a_wn*has_a_wn);
    hold on;
    plot(t,b_a*has_a_bias);
    title({'Accelerometer Measurement Noise',''});
    xlabel('Time [s]');
    ylabel('Accelerometer Error [m/s^2]');
    legend('White Noise','Bias');

    subplot(2,2,4);
    plot(t,n_a);
    title({'Net Accelerometer Measurement Error',''});
    xlabel('Time [s]');
    ylabel('Accelerometer Error [m/s^2]');

    %% Generating Actual Trajectories
    psi = (psi_0:num_rotations*2*pi/(length(t)-1):num_rotations*2*pi)'; %polar angle, [rad]
    alpha = psi + pi/2; %azimuth angle, Initial azimuth: 90? (towards x-accelerometer)

    %use this line instead to have azimuth angle wrap at 2pi
    %%alpha = mod(psi + pi/2,2*pi); %azimuth angle, Initial azimuth: 90? (towards x-accelerometer)

    x_m = [r*cos(psi),r*sin(psi)]; % map frame, [north, east]
    dx_m = [dr.*cos(psi)-r.*sin(psi).*dpsi , r.*cos(psi).*dpsi + dr.*sin(psi)];

    %(definitely could have simplified this...)
    ddx_m = [ddr.*cos(psi) - dr.*sin(psi).*dpsi - (dr.*sin(psi).*dpsi + r.*cos(psi).*dpsi.^2 +
r.*sin(psi).*ddpsi), ... %ddx_m1
             dr.*cos(psi).*dpsi - r.*sin(psi).*dpsi.^2 + r.*cos(psi).*ddpsi + ddr.*sin(psi)+dr.*cos(psi).*dpsi
]; %ddx_m2

    dalpha = dpsi*ones(length(t),1); %deriving alpha wrt time = deriving psi wrt time

    v_b = [ones(length(t),1)*omega*r,zeros(length(t),1)]; % got lazy... this works here because omega and r are
constant. Would need to change this line if that was not the case
    a_b = [zeros(length(t),1),ones(length(t),1)*omega^2*r]; %got lazy... this works here because omega and r are
constant. Would need to change this line if that was not the case

    %% Adding Noise to Sensors

    gyro = dalpha + n_g; %ideal gyroscope
    accel = a_b + n_a; %ideal accelerometer, f

    % Considering the initial conditions to be known, apply strapdown inertial navigation to
    % calculate the trajectory parameters (i.e. the attitude (azimuth) and 2D velocity and
    % position vectors, respectively).
    %[Rmb,Rbm] = RotationMatrix(alpha); %define rotation matrices for all angles, alpha

    %initialize variables with initial conditions
    alpha_sd = zeros(length(alpha),1); %strapdown azimuth
    alpha_sd(1) = alpha_0;
    v_sd = zeros(length(t),2); %strapdown velocity, map frame
    v_sd(1,:) = dx_0;
    x_sd = v_sd;
    x_sd(1,:) = x_0;

    %Localizing along trajectory. doing this with a loop. we'll see if i regret it later
    for i=2:length(t)
        if strcmp(method,'Rectangular')
            alpha_sd(i) = alpha_sd(i-1)+gyro(i)*(t(i)-t(i-1)); %strapdown azimuth
            [~,Rbm] = RotationMatrix(alpha_sd(i));
            Rbm = squeeze(Rbm);
            v_sd(i,:) = (v_sd(i-1,:)'+Rbm*accel(i,:)'*(t(i)-t(i-1)))';
            x_sd(i,:) = (x_sd(i-1,:)'+v_sd(i,:)'*(t(i)-t(i-1)))';
        elseif strcmp(method, 'Trapezoidal')
            alpha_sd(i) = alpha_sd(i-1)+1/2*(gyro(i)+gyro(i-1))*(t(i)-t(i-1));
            [~,Rbm] = RotationMatrix(alpha_sd(i)); %current Rotation Matrix
            Rbm = squeeze(Rbm);
            [~,Rbm_p] = RotationMatrix(alpha_sd(i-1)); %prior Rotation Matrix
            Rbm_p = squeeze(Rbm_p);
            v_sd(i,:) = (v_sd(i-1,:)'+1/2*(Rbm*accel(i,:)'+Rbm*accel(i-1,:)')*(t(i)-t(i-1)))';
            x_sd(i,:) = (x_sd(i-1,:)'+1/2*(v_sd(i,:)+v_sd(i-1,:))'*(t(i)-t(i-1)))';
        else
            break %err. method invalid
        end
    end

    %% Plotting Trajectories
    figure(k);
    %position
    subplot(2,2,1);
    plot(x_m(:,2),x_m(:,1));
    hold on;
    xlabel('X_m_2, East (m)');
    ylabel('X_m_1, North (m)');
    title(['True vs Estimate Position (',method,' Integration, ',sample_rate_text,'Hz)']);
    plot(x_sd(:,2),x_sd(:,1));
    axis('equal');
    legend('True',strcat(method,', ',sample_rate_text,'Hz'));

    %% Calculate Errors
    err_x = x_sd-x_m;
    err_alpha = alpha_sd - alpha;
    err_v = v_sd - dx_m;

    err_x = [err_x,sqrt(err_x(:,1).^2+err_x(:,2).^2)]; %add third column for position error magnitude
    err_v = [err_v,sqrt(err_v(:,1).^2+err_v(:,2).^2)]; %add third column for position error magnitude

    err_table(k,:) =
[mean(abs(err_alpha)),max(abs(err_alpha)),mean(abs(err_x(:,3))),max(abs(err_x(:,3))),mean(abs(err_v(:,3))),max(a
bs(err_v(:,3)))];

    %% Plotting Errors
    %position
    subplot(2,2,3);
    plot(t,err_x(:,1));
    hold on;
    plot(t,err_x(:,2));
    plot(t,err_x(:,3));
    xlabel('Time (s)');
    ylabel('X_m Error (m)');
    title(['Position Error Over Time (',method,' Integration, ',sample_rate_text,'Hz)']);
    legend('North','East','Error Magnitude');

    %azimuth
    subplot(2,2,2);
    plot(t,err_alpha);
    hold on;
    xlabel('Time (s)');
    ylabel('Azimuth Error (rad)');
    title({['Azimuth Angle Error Over Time (',method,' Integration, ',sample_rate_text,'Hz)'],' '});

    %velocity v1_m
    subplot(2,2,4);
    plot(t,err_v(:,1));
    hold on;
    plot(t,err_v(:,2));
    plot(t,err_v(:,3));
    xlabel('Time (s)');
    ylabel('V_m Error (m/s)');
    if(sampling_rate > 10)
        title({['Velocity Error Over Time (',method,' Integration, ',sample_rate_text,'Hz)'],' '});
    else
        title({['Velocity Error Over Time (',method,' Integration, ',sample_rate_text,'Hz)']});
    end
    legend('North','East','Error Magnitude');
end

%% Functions
function [Rmb,Rbm] = RotationMatrix(alpha)

    Rmb = zeros(length(alpha),2,2); %preallocate memory
    Rbm = Rmb;

    for i=1:length(alpha)
        Rmb(i,:,:) = [cos(alpha(i)),sin(alpha(i)); -sin(alpha(i)),cos(alpha(i))];%transformation matrix R(m->b)
        Rbm(i,:,:) = squeeze(Rmb(i,:,:))'; %transformation matrix R(b->m)
    end
end
```

whiteNoiseRandomWalk.m

```
function [s,rw] = whiteNoiseRandomWalk(n,seed)
    rng(seed);
    s = randn(n,1);
    rw = cumsum(s);
end
```

firstOrderGaussMarkov.m

```
function [fogm] = firstOrderGaussMarkov(t_correlation, t_sample, s)
    % computes the first order guass-markov process given:
    % correlation time, sample time, and random sequence
    Z = zeros(length(s),1);
    Z(1) = s(1);
    Beta = 1/t_correlation;
    for i=2:length(s)
        Z(i) = exp(-Beta*t_sample)*Z(i-1)+s(i);
    end
    fogm = Z;
end
```