ENV-541 Sensor Orientation
Lab 4 - Inertial Navigation in 2D / Nominal Signal

Michael Spieler

October 26, 2018

Strapdown inertial navigation

The simulated sensor samples are in the body frame and must be transformed to the inertial map frame.

$$\omega_{mb}^{b} = \begin{bmatrix}\omega_0\end{bmatrix}$$

$$f^m = R_b^m(\alpha) f^b = \begin{bmatrix} 0 \\ r\omega_0^2 \end{bmatrix}$$

$$R_b^m(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

The problem can be formulated as a differential equation which can be integrated numerically:

$$\begin{aligned} \dot{\alpha} &= \omega_0 \\ \dot{v}^m &= f^m \\ \dot{x}^m &= v^m \end{aligned}$$

1st order rectangular integration

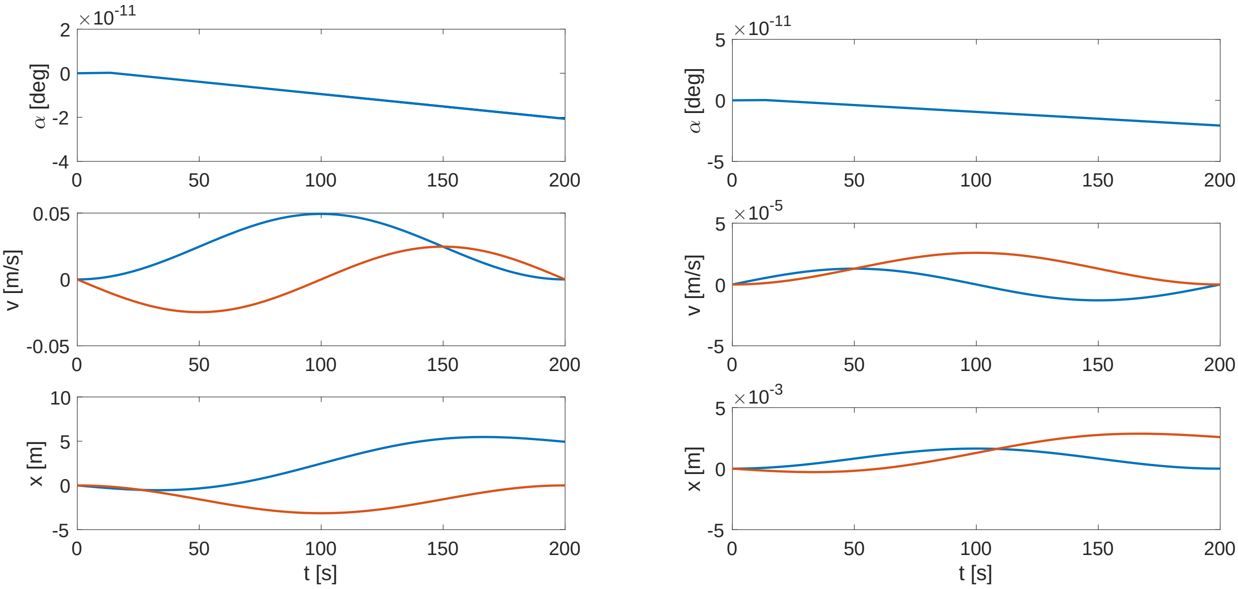
$$x_k = x_{k-1} + \dot{x}_k \Delta t$$

2nd order trapezoidal integration

$$x_k = x_{k-1} + \frac{1}{2}(\dot{x}_k + \dot{x}_{k-1})\Delta t$$

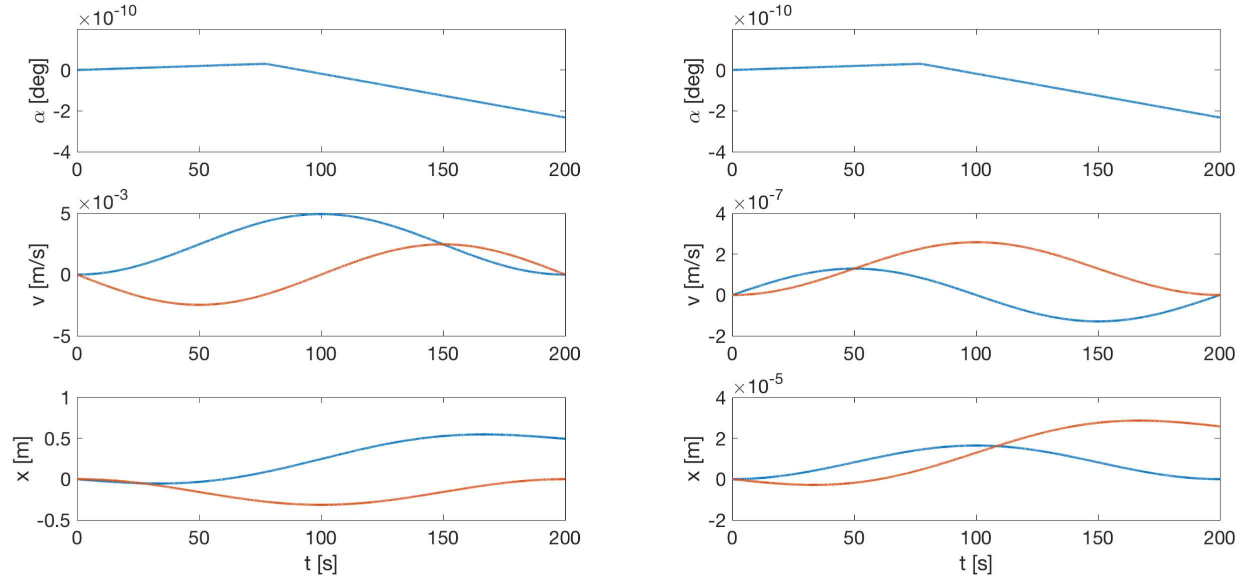
The errors of the simulated trajectory using the two integration methods at 10Hz and 100Hz are shown in figure 1 and figure 2.

Error plots



(a) Rectangular integration (b) Trapezoidal integration

Figure 1: Trajectory errors at 10Hz sampling rate



(a) Rectangular integration (b) Trapezoidal integration

Figure 2: Trajectory errors at 100Hz sampling rate

Trajectory error

Table 1 shows the maximal trajectory errors in velocity an position for 1st and 2nd order integration at 10Hz and 100Hz sampling frequency. In general, the higher order integration method and higher sampling rates have the lowest error.

The 1st order rectangular integration is not suited for this application, since it assumes constant derivatives. Trapezoideal integration gives alredy better results. However, the nonlinear nature of rotations still results in errors. Errors can be kept low by using a high sampling rate.

Note: The asimuth error is not listed, since it is zero for all integration methods (the error plots show a small error, which is due to numerical errors during floating point calculations). This is not surprising, since the angular rate is constant and thus a 1st order integration is perfectly sufficient.

Error	Rectangular 10Hz	Trapezoidal 10Hz	Rectangular 100Hz	Trapezoidal 100Hz
Velocity x [m/s]	4.935e-02	1.292e-05	4.935e-03	**1.292e-07**
Velocity y [m/s]	2.469e-02	2.584e-05	2.468e-03	**2.584e-07**
Position x [m]	5.474	1.645e-03	5.473e-01	**1.645e-05**
Position y [m]	3.140	2.865e-03	3.141e-01	**2.865e-05**

Table 1: Maximal errors during trajectory

Code

```
% generate reference trajectory
w = pi/100;
r = 500;
% 10Hz
t10 = 0:0.1:200;
a10 = (0:2000)'*pi/100*0.1 + pi/2;
v10 = (w*r*[cos(a10) sin(a10)]);
p10 = (r*[sin(a10) -cos(a10)]);
% 100Hz
t100 = 0:0.01:200;
a100 = (0:20000)'*pi/100*0.01 + pi/2;
v100 = (w*r*[cos(a100) sin(a100)]);
p100 = (r*[sin(a100) -cos(a100)]);

%% Simulation
[a10_rect, v10_rect, p10_rect] = rectangular_int(10);
[a100_rect, v100_rect, p100_rect] = rectangular_int(100);
[a10_trapez, v10_trapez, p10_trapez] = trapezoidal_int(10);
[a100_trapez, v100_trapez, p100_trapez] = trapezoidal_int(100);

%% Error plots
set(groot,'DefaultAxesFontSize',17)
set(groot,'DefaultLineLineWidth',2)

plot_error('Error Rectangular integration 10Hz', a10_rect-a10, v10_rect-v10, p10_rect-p10, t10)
plot_error('Error Rectangular integration 100Hz', a100_rect-a100, v100_rect-v100, p100_rect-p100, t100)
plot_error('Error Trapezoidal integration 10Hz', a10_trapez-a10, v10_trapez-v10, p10_trapez-p10, t10)
plot_error('Error Trapezoidal integration 100Hz', a100_trapez-a100, v100_trapez-v100, p100_trapez-p100, t100)
%% functions
function [] = plot_error(title, ea, ev, ep, t)
    fprintf('%s\n', title)
    fprintf('velocity error: x %.3e, y %.3e\n', max(abs(ev(:,1))), max(abs(ev(:,2))))
    fprintf('position error: x %.3e, y %.3e\n', max(abs(ep(:,1))), max(abs(ep(:,2))))
    figure
    subplot(3,1,1)
    plot(t, 180/pi*ea)
    ylabel('\alpha [deg]')
    subplot(3,1,2)
    plot(t, ev)
    ylabel('v [m/s]')
    subplot(3,1,3)
    plot(t, ep)
    ylabel('x [m]')
    xlabel('t [s]')
    %suptitle(title)
end

function [a, v, p] = rectangular_int(f_int)
    t_end = 200;
    dt = 1/f_int;
    N = t_end*f_int + 1;

    r = 500;
    w = pi/100; % [rad/s]
    f = [0; r*w^2];

    a{1} = pi/2;
    v{1} = [0; w*r];
```

```
    p{1} = [r; 0]; % [m] NE coordinate system
    for k=2:N
        a{k} = a{k-1} + dt*w;
        f_m = Rbm(a{k})*f;
        v{k} = v{k-1} + f_m*dt;
        p{k} = p{k-1} + v{k}*dt;
    end

    a = cell2mat(a)';
    v = cell2mat(v)';
    p = cell2mat(p)';
end

function [a, v, p] = trapezoidal_int(f_int)
    t_end = 200;
    dt = 1/f_int;
    N = t_end*f_int + 1;

    r = 500;
    w = pi/100;
    f = [0; r*w^2];

    a{1} = pi/2;
    v{1} = [0; w*r];
    p{1} = [r; 0];
    for k=2:N
        a{k} = a{k-1} + dt*(w+w)/2;
        f0 = Rbm(a{k-1})*f;
        f1 = Rbm(a{k})*f;
        v{k} = v{k-1} + (f0 + f1)/2*dt;
        p{k} = p{k-1} + (v{k}+v{k-1})/2*dt;
    end

    a = cell2mat(a)';
    v = cell2mat(v)';
    p = cell2mat(p)';
end

function x = RK(x0, xd0, xd1, xd2, dt)
    x = x0 + 1/6 *(xd0 + 4*xd1 + xd2)*dt;
end

function R = Rbm(a)
    R = [cos(a) -sin(a); sin(a) cos(a)];
end
```