Homework 2

Table of Contents

Bailey Waterman, Keshuai Xu

1

1. Phase-shift keying 2. False 3. Sheer distance between satellite and reciever

2a

```
clear variables; close all; clc;

v_e = [10 0 0]'; % units
R_i_e = rotx(deg2rad(13))*roty(deg2rad(15))*rotz(deg2rad(10));

v_i = R_i_e \ v_e % units
```

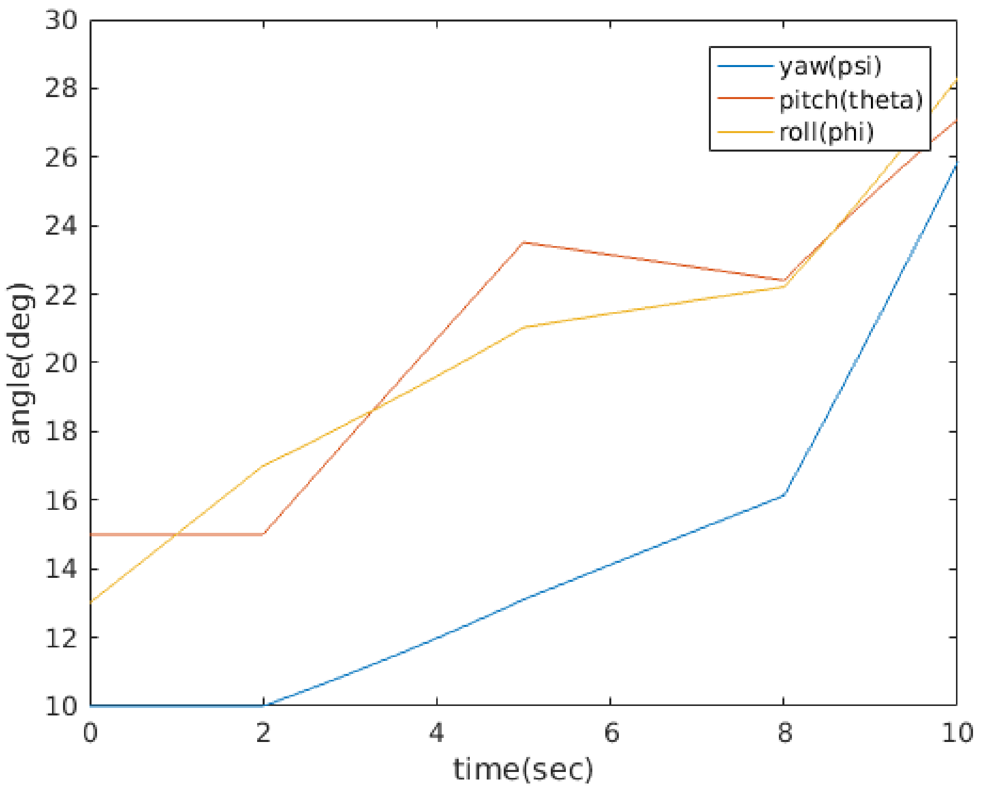
```

v_i =

    9.9998
   -0.0305
    0.0457

```

2b

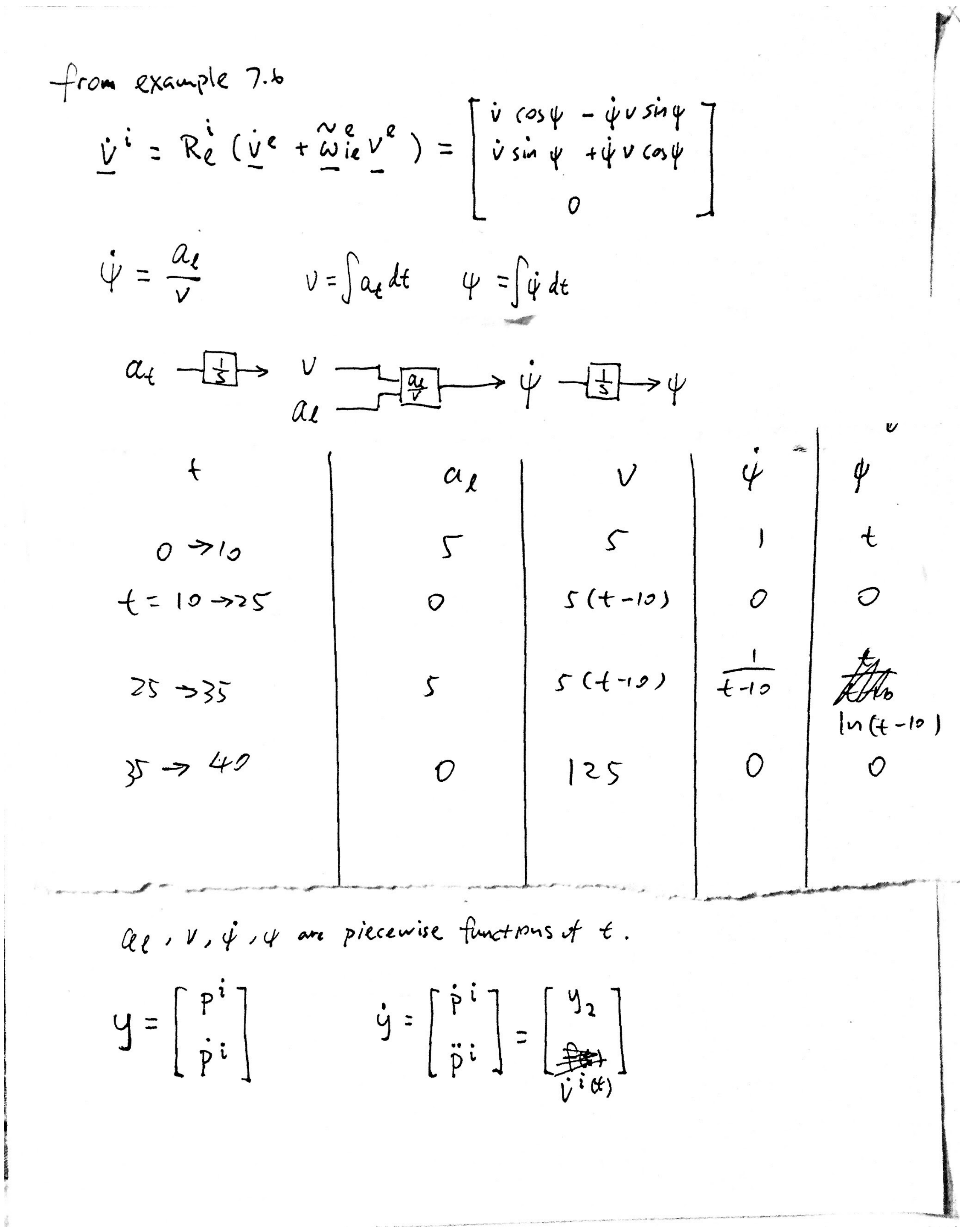
```
function euler321_angles_dot = rotational_kinematics(t,
 euler321_angles, omega_iee)

psi_yaw = euler321_angles(1);
theta_pitch = euler321_angles(2);
phi_roll = euler321_angles(3);

H_321 = [-sin(theta_pitch) 0 1;
    sin(phi_roll)*cos(theta_pitch) cos(phi_roll) 0;
    cos(phi_roll)*cos(theta_pitch) -sin(phi_roll) 0];

euler321_angles_dot = H_321 \ omega_iee;
end
```

```
clear variables; close all; clc;

%initial conditions
euler321_angles_initial = [10; 15; 13] * pi/180; %radians
time_span = [0 2]; %seconds
omega_iee = [2; 0; 0;] *pi/180; %radians/second

%solve ODE
[t_sim, euler321_angles_sim] = ...
ode45(@(t, y)rotational_kinematics(t, y, omega_iee), time_span,
 euler321_angles_initial);

%initial conditions same as last ones
euler321_angles_initial = euler321_angles_sim(end, :); %rad
time_span = [2 5]; %s
omega_iee = [1; 3; 0;] * pi/180; %rad/s

%solve ODE
[t_sim_2, euler321_angles_sim_2] = ...
ode45(@(t, y)rotational_kinematics(t, y, omega_iee), time_span,
 euler321_angles_initial);

%initial conditions same as last ones
euler321_angles_initial = euler321_angles_sim_2(end, :); %rad
time_span = [5 8]; %s
omega_iee = [0; 0; 1;] * pi/180; %rad/s

%solve ODE
[t_sim_3, euler321_angles_sim_3] = ...
ode45(@(t, y)rotational_kinematics(t, y, omega_iee), time_span,
 euler321_angles_initial);

%initial conditions same as last ones
euler321_angles_initial = euler321_angles_sim_3(end, :); %rad
time_span = [8 10]; %s
omega_iee = [1; 4; 3;] * pi/180; %rad/s

%solve ODE
[t_sim_4, euler321_angles_sim_4] = ...
ode45(@(t, y)rotational_kinematics(t, y, omega_iee), time_span,
 euler321_angles_initial);

euler321_angles_degrees_1 = euler321_angles_sim * 180/pi;
euler321_angles_degrees_2 = euler321_angles_sim_2 * 180/pi;
euler321_angles_degrees_3 = euler321_angles_sim_3 * 180/pi;
euler321_angles_degrees_4 = euler321_angles_sim_4 * 180/pi;

time = [t_sim; t_sim_2; t_sim_3; t_sim_4;];
euler321_angles_degrees = [euler321_angles_degrees_1;
 euler321_angles_degrees_2; euler321_angles_degrees_3;
 euler321_angles_degrees_4;];
```

```
plot(time, euler321_angles_degrees)

xlabel('time(sec)')
ylabel ('angle(deg)')
legend ('yaw(psi)','pitch(theta)','roll(phi)')
```



3



from example 7.6

$$\dot{\underline{v}}^i = R_e^i\left(\dot{\underline{v}}^e + \underline{\tilde{\omega}}_{ie}^e \underline{v}^e\right) = \begin{bmatrix} \dot{v}\cos\psi - \dot{\psi} v \sin\psi \\ \dot{v}\sin\psi + \dot{\psi} v \cos\psi \\ 0 \end{bmatrix}$$

$\dot{\psi} = \frac{a_\ell}{v}$ $\quad v = \int a_t \, dt \quad \psi = \int \dot{\psi} \, dt$

$$\begin{tikzcd} a_t \arrow[r] & \boxed{\frac{1}{s}} \arrow[r] & v \arrow[dr] & & & & \\ & & & \boxed{\frac{a_\ell}{v}} \arrow[r] & \dot{\psi} \arrow[r] & \boxed{\frac{1}{s}} \arrow[r] & \psi \\ & & a_\ell \arrow[ur] & & & & \end{tikzcd}$$

t	a_ℓ	v	$\dot{\psi}$	ψ
$0 \to 10$	5	5	1	t
$t = 10 \to 25$	0	$5(t-10)$	0	0
$25 \to 35$	5	$5(t-10)$	$\frac{1}{t-10}$	~~[illegible]~~ $\ln(t-10)$
$35 \to 40$	0	125	0	0

$a_\ell, v, \dot{\psi}, \psi$ are piecewise functions of t.

$$y = \begin{bmatrix} p^i \\ \dot{p}^i \end{bmatrix} \qquad \dot{y} = \begin{bmatrix} \dot{p}^i \\ \ddot{p}^i \end{bmatrix} = \begin{bmatrix} y_2 \\ \dot{v}^i(t) \end{bmatrix}$$

```
function [ a_l ] = hw2_a_l( t )
%HW2_A_L Summary of this function goes here
%   Detailed explanation goes here
if t < 0; a_l = 5; warning('out of bound'); end;
if t < 10; a_l = 5; return; end;
if t < 25; a_l = 0; return; end;
if t < 35; a_l = 5; return; end;
if t < 40; a_l = 0; return; end;
a_l = 0;
warning('out of bound');

end
```

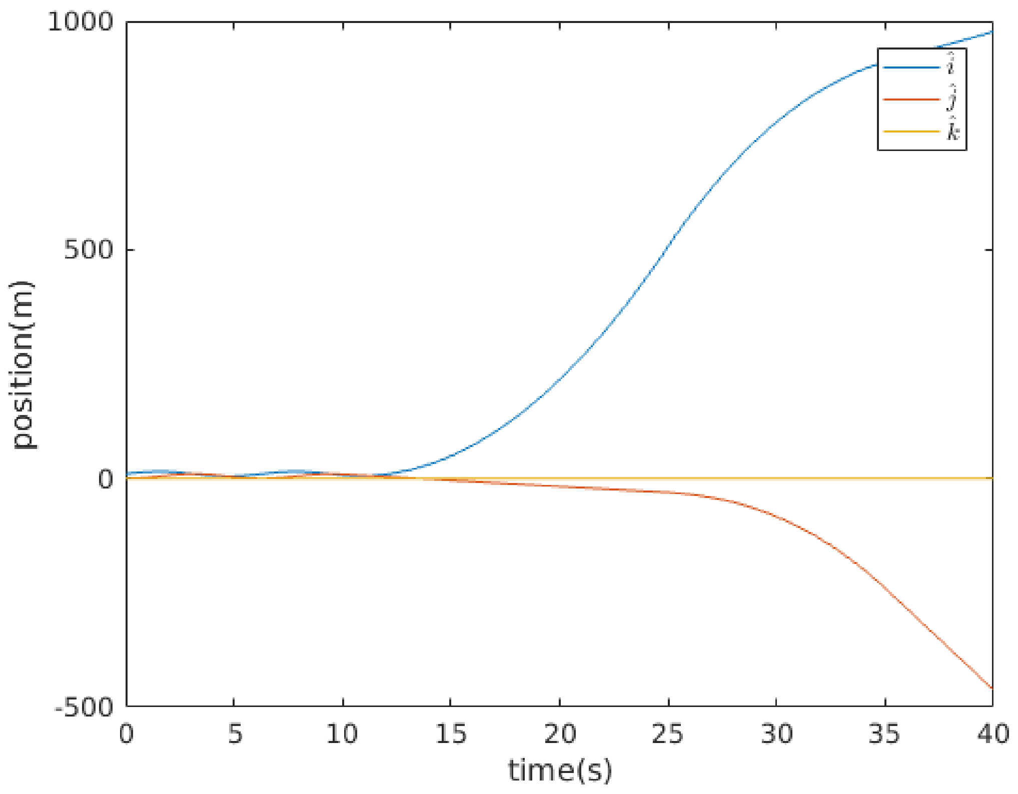
```
function [ a_t ] = hw2_a_t( t )
%UNTITLED7 Summary of this function goes here
%   Detailed explanation goes here
if t < 0; a_t = 0; warning('out of bound'); end;
if t < 10; a_t = 0; return; end;
if t < 25; a_t = 5; return; end;
if t < 35; a_t = 5; return; end;
if t < 40; a_t = 0; return; end;
a_t = 0;
warning('out of bound');

end
```

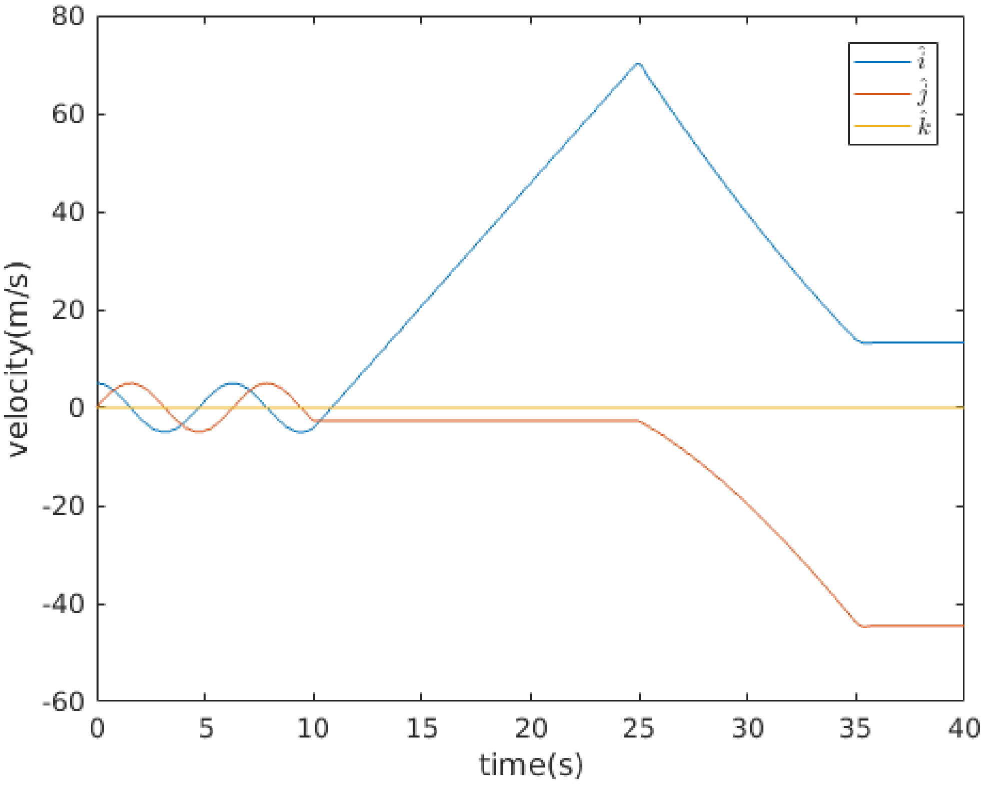
```
function [ psi ] = hw2_psi( t )
%UNTITLED6 Summary of this function goes here
%   Detailed explanation goes here
if t < 0; psi = t; warning('out of bound'); end;
if t < 10; psi = t; return; end;
if t < 25; psi = 0; return; end;
if t < 35; psi = log(t-10); return; end;
if t < 40; psi = 0; return; end;
psi = 0;
warning('out of bound');

end
```

```
function [ psi_dot ] = hw2_psi_dot( t )
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
if t < 0; psi_dot = 1; warning('out of bound'); end;
if t < 10; psi_dot = 1; return; end;
if t < 25; psi_dot = 0; return; end;
if t < 35; psi_dot = 1/(t-10); return; end;
if t < 40; psi_dot = 0; return; end;
psi_dot = 0;
warning('out of bound');
```

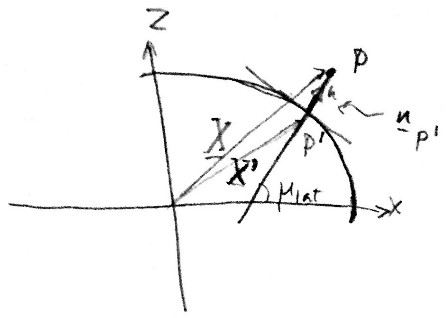
```
end




function [ v ] = hw2_v( t )
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here

if t < 0; v = 5; warning('out of bound'); end;
if t < 10; v = 5; return; end;
if t < 25; v = 5*(t-10); return; end;
if t < 35; v = 5*(t-10); return; end;
if t < 40; v = 5*(35-10); return; end;
v = 5*(35-10);
warning('out of bound');

end




clear variables; close all; clc;

% from the derivation on the paper, v_dot_i is a piecewise function of
 t.
v_dot_i = @(t) [hw2_a_t(t)*cos(hw2_psi(t)) -
 hw2_psi_dot(t)*hw2_v(t)*sin(hw2_psi(t));
            hw2_a_t(t)*sin(hw2_psi(t)) +
 hw2_psi_dot(t)*hw2_v(t)*cos(hw2_psi(t));
            0];

% in odefun and initial conditon y0,
% y(1:3, :) = p_i (position in m)
% y(4:6, :) = v_i (velocity in m/s)

odefun = @(t,y) [y(4:6, :); v_dot_i(t)];
y0 = [10 0 0 5 0 0]';
warning('off','all')
[t_sim, y_sim] = ode45(odefun, [0 40], y0);
warning('on','all')

figure();
plot(t_sim, y_sim(:, 1:3));
xlabel('time(s)')
ylabel('position(m)')
legend({'$\hat{i}$', '$\hat{j}$', '$\hat{k}$'},'Interpreter','latex')
figure();
plot(t_sim, y_sim(:, 4:6));
xlabel('time(s)')
ylabel('velocity(m/s)')
legend({'$\hat{i}$', '$\hat{j}$', '$\hat{k}$'},'Interpreter','latex')
% at t=40 in I frame,
position = y_sim(end, 1:3)' % m
velocity = y_sim(end, 4:6)' % m/s
```

```
position =

  977.1305
 -462.6104
         0


velocity =

   13.2583
  -44.5351
         0
```




80
60
40
20
0
-20
-40
-60
velocity(m/s)
0
5
10
15
20
25
30
35
40
time(s)
$\hat{i}$
$\hat{j}$
$\hat{k}$

Homework 1 Extra Credit



$P(x, y, z)$
$P'(x', y', z')$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = f\begin{bmatrix} \varphi \\ \lambda \\ h \end{bmatrix}$$

From ellipsoidal geometry

$$\frac{x^2+y^2}{a^2} + \frac{z^2}{b^2} = 1 \qquad (1)$$

$$\underline{X}' = \begin{bmatrix} N\cos\varphi\cos\lambda \\ N\cos\varphi\sin\lambda \\ \frac{b^2}{a^2}N\sin\varphi \end{bmatrix} \qquad \begin{matrix}(2)\\(3)\\(4)\end{matrix}$$

where $N = a^2 \cdot (a^2\cos^2\varphi + b^2\sin^2\varphi)^{-0.5}$ (5)

normal radius of curvature

$$\underline{n}_{p'} = \begin{bmatrix} \cos\varphi\cos\lambda \\ \cos\varphi\sin\lambda \\ \sin\varphi \end{bmatrix} \qquad (6)$$

Unit vector normal to the ellipsoid surface at p'.

Vector chain

$$\underline{X} = \underline{X}' + h\,\underline{n}_{p'} \qquad (7)$$

Substitute (3), (6)

$$\underline{X} = \begin{bmatrix} (N+h)\cos\varphi\cos\lambda \\ (N+h)\cos\varphi\sin\lambda \\ (\frac{b^2}{a^2}N+h)\sin\varphi \end{bmatrix}$$

Reference: Hofmann-Wellenhof, Bernhard, and Helmut Moritz. Physical geodesy. Springer Science & Business Media, 2006.

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

```
% WGS84
% http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf
a = 6378137.0; % m
inv_f = 298.257223563;
b = a*(1-1/inv_f); % m

N = @(phi) a^2 * (a^2*cos(phi)^2 + b^2*sin(phi)^2)^(-0.5);

% input in radians and m, output in m
geodesic2ctrs = @(lat, long, h) [(N(lat)+h)*cos(lat)*cos(long);
                                 (N(lat)+h)*cos(lat)*sin(long);
                                 (b^2*N(lat)/a^2+h)*sin(lat)];

sole_ctrs = geodesic2ctrs(deg2rad(42.271167),deg2rad(-71.807627),0) %
 m
```

```
sole_ctrs =

   1.0e+06 *

    1.4757
   -4.4905
    4.2679
```

Published with MATLAB® R2016a