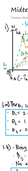
Midterm (Akshat Dubey)

Monday, November 4, 2024 2:07 PM

1) a)



Knowing frame 1, we can find θ relative to that frame.

- $R_1 = R_{1,1,0}$ since $\theta_1 = 0$
- $R_2 = R_{2,1,\theta_2}$

- Have after this, have a x_4 sweep model, all weight term is given

- $R_1 = R_{1,1,0}$ $\quad T_{1,1,0} = T_{1,1,0}$ $\quad \theta_1 = \theta_1$
- $R_2 = R_{2,1,\theta_2}$ $\quad T_{2,1,\theta_2} = T_{2,1,\theta_2}$ $\quad \theta_2 = \theta_2 + \theta_1$
- $R_3 = R_{3,1,\theta_3}$ $\quad T_{3,1,\theta_3} = T_{3,1,\theta_3}$ $\quad \theta_3 = \theta_3 + \theta_2 + \theta_1$

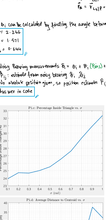
1.a) No, can be calculated by finding the angle between $\vec{F}_1$ and $\vec{F}_2$

- $\theta_1 = 3.146$
- $\theta_2 = 1.571$
- $\theta_3 = 0.000$

1.b) - Using Numerical measurements $\theta_1 = \theta_2 + \theta_1 \left(\frac{\pi}{2} \right) + \theta_1 \left(\frac{\pi}{2} \right)$

- $\vec{F}_1$ always lower than $\vec{F}_2$, $\vec{F}_3$
- the smaller gripper can, or gripper closest $\vec{F}_1$ will be with respect to so
- this will be a case

1.c)



1.d)

1.e) - $\theta_1 = 0$, we can see that probability that $\vec{F}_1$ inside the triangle formed by $\vec{F}_1, \vec{F}_2, \vec{F}_3$ increases as x increases. As x increases, the equal to between variance and increases. Very close to between variance small will cause the $\vec{F}_1, \vec{F}_2, \vec{F}_3$ obtuse to have high probability being farther away from each other. When the $\vec{F}_1, \vec{F}_2, \vec{F}_3$ obtuse, the further away from each other, the triangle formed by them is larger, increasing the probability of $\vec{F}$ being inside the triangle.

- $\theta_1 = \frac{\pi}{2}$, we see that as x increases, the average distance of $\vec{F}$ to $\vec{F}_1$ increases. This happens because as x increases, the force $\vec{F}_1$ increasingly has more moment. This causes the distance of $\vec{F}_2, \vec{F}_3$ and $\vec{F}_1$, to have a larger probability of being farther away from $\vec{F}$. Because the force is constant, x is very sensitive, that $\vec{F}_2, \vec{F}_3$ will be optimized in way that the constant $\vec{F}$ being close to $\vec{F}$.

As the x increases, this exponentially that one of $\vec{F}_2, \vec{F}_3$ $\vec{F}_1$ is exactly far away from $\vec{F}$ increases. This occurring exponentially that $\vec{F}$ will also be pulled away from $\vec{F}$, making the distance larger.

1.f)

1.g) - Frames:

$F_0 \rightarrow F_1$: $\theta_1 = 0$ Revolute angle rotation

$F_1 \rightarrow F_2$: $\theta_2 = 0$ Revolute angle rotation

Angular Velocity: $\omega_0 = 0$ Below

- Translation matrix $T = B$ at $t = 0$: $D_{B,0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- We can propagate the translation matrix in time using velocity equation.

$\dot{D}_{B,t} = -\hat{\omega}_{0,B} D_{B,t}$ — eq 6.4.16 in Book

to get $D_{B,t}$ we can do this by using this.

- Angular velocity $\Sigma = B$: $\omega_{0,B}^{B} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$

- Angular θ_1 at $t = 0$: $\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

- We can propagate the rotation matrix in time by using the angular velocity equation for 6.4.13 sequence

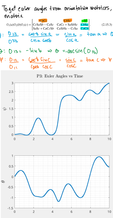
$\dot{R}_{B,t} = \hat{\omega}_{0,B} R_{B,t}$ — eq 6.4.13 in Book

- Solve for only need D_1 and D_2 to construct $T_{B,t}$ frames relative angle.

- We can solve rotation matrix relationship to $D_{B,t}$: $D_{B,t} = D_{B,t} D_{B,t}$

$\Rightarrow D_{B,t} = D_{B,t} D_{B,t}$

We have $D_{B,t}$ and $D_{B,t}$ at time steps, so we can combine $D_{B,t}$ for all timesteps



$D_{B,t}$ is an orientation matrix that describes a 3-2-1 sequence which follows the the following equation

$D_{B,t} = D_{B,t} D_{B,t} = D_{B,t} D_{B,t} D_{B,t}$ (2.13.12)

where from book

$D_{B,t} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}$ (2.13.13)

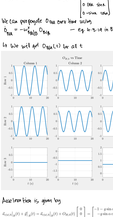
- Brief color guide from translation matrices, we need to decide elements of the orientation matrix.

$D_{B,t} = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi s\theta c\phi \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi s\theta c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}$

$\Rightarrow$ $\frac{D_{21}}{D_{11}} = \frac{\sin\psi \cos\theta}{\cos\psi \cos\theta} = \frac{\sin\psi}{\cos\psi} = \tan\psi \Rightarrow \psi = \tan^{-1}\left(\frac{D_{21}}{D_{11}}\right)$

$\Rightarrow$ $D_{31} = -\sin\theta \Rightarrow \theta = \sin^{-1}(-D_{31})$

$\Rightarrow$ $\frac{D_{32}}{D_{33}} = \frac{\cos\theta \sin\phi}{\cos\theta \cos\phi} = \frac{\sin\phi}{\cos\phi} = \tan\phi \Rightarrow \phi = \tan^{-1}\left(\frac{D_{32}}{D_{33}}\right)$



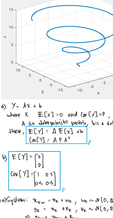
2) $y = 2.5 \sin 3t$, $x = 1.5t$

- Assume no wind, gripper weighed with 0
- $\vec{F}$ Position is collinear to $\vec{v}$
- $D_{B,t}$: Desired orientation matrix from frame 0 to B
- Angular velocity Ω at t, is zero at $\omega_{0,B}^{B}$: $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

- Since ω zero at $t = 0$: $D_{B,0}(0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- We can propagate this over time using $\dot{D}_{B,t} = -\hat{\omega}_{0,B} D_{B,t}$ — eq 6.4.16 in Book — ①

In the end of $D_{B,t}(t)$ be at t



- Acceleration is given by

$\dot{\omega}_{0,B}(t) = \ddot{R}_{B,t}(t) = \dot{\omega}_{0,B}(t) = \begin{bmatrix} 0 & -\dot{\omega}_z & \dot{\omega}_y \\ \dot{\omega}_z & 0 & -\dot{\omega}_x \\ -\dot{\omega}_y & \dot{\omega}_x & 0 \end{bmatrix} D_{B,t}(t)$, (2.6)

$\Rightarrow$ $\vec{F}_{\text{actually}}(t) = \vec{F}_{\text{body}}(t)$

$\begin{bmatrix} -1.5\sin\alpha \cos\alpha \\ -1.5\sin\alpha \sin\alpha \\ -1.5\sin\alpha \end{bmatrix} D_{B,t}(t) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ — ②

$\vec{F}_{\text{body}}$ is the gripper's to B known using desired trajectory.

In this equation, $\vec{F}_{\text{actual}}$ which contain additional forces $\vec{F}$ using actual gripper the system.

- Let $\vec{\omega}_{\text{actual}}$ which, $\vec{\omega}_{\text{actual}}$ $\vec{\omega}_{\text{actual}}$ $\vec{\omega}_{\text{actual}}$

Table of Contents

..... 1
part a derived angles 1
part b position estimates 1
part c plot p inside versus sigma 2
part d plot avg dist versus sigma 3
tests 4
functions 4


```
clc; clear; close all;

% given position vectors between w, P, L1, L2, L3 (w = L1)
r_L1_w = [0;0;0]; r_L2_w = [4;2;0]; r_L3_w = [1;4;0]; r_P_w = [2.5;2;0];
S = [0;1;0]; % star vector

% derived position vectors
r1 = r_L1_w - r_P_w; % r_L1_P
r2 = r_L2_w - r_P_w; % r_L2_P
r3 = r_L3_w - r_P_w; % r_L3_P
```

part a derived angles

```
theta1 = get_angle_between(r1, S);
theta2 = get_angle_between(r2, S);
theta3 = get_angle_between(r3, S);

fprintf("P1.a: true bearing angles\n- theta1: %.3f\n- theta2: %.3f\n- theta3:
%.3f\n", theta1, theta2, theta3);
```

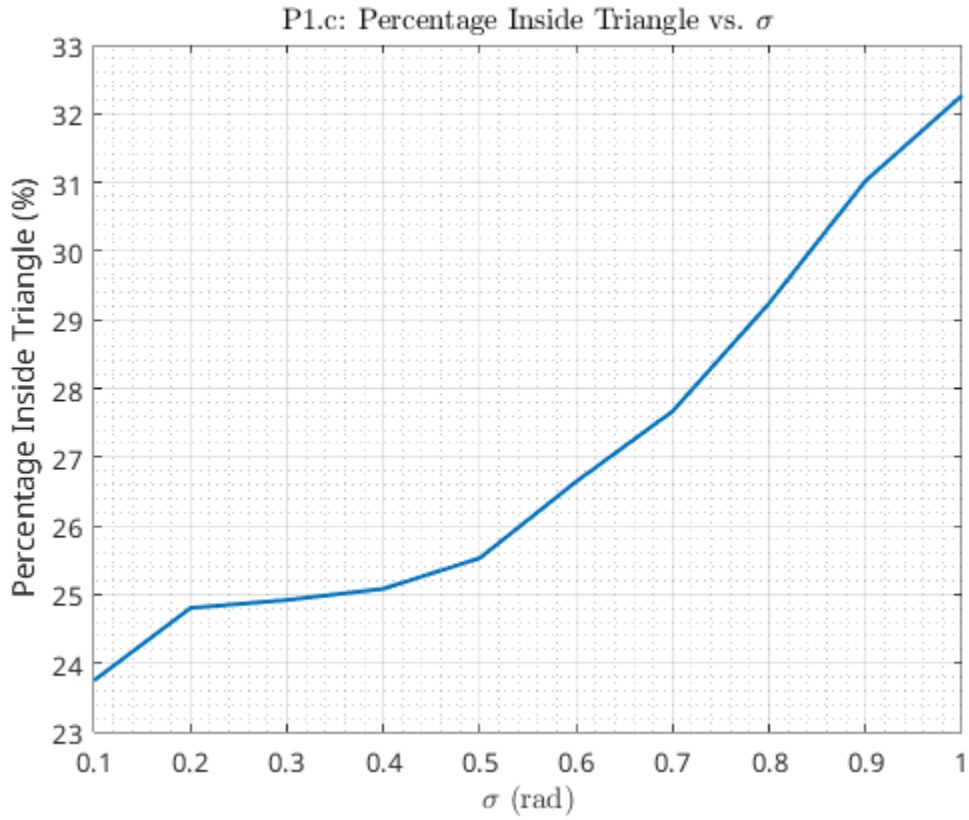
part b position estimates

```
beta1 = 0.05; beta2 = -0.03; beta3 = 0.04; % bearing biases
sigma = 0.1:0.1:1; % bearing noise std devs
num_samples = 100000;
prob_inside = zeros(length(sigma),1);
avg_dist = zeros(length(sigma),1);
for jj = 1:length(sigma) % For each sigma value
    % Do some setup stuff (if needed)
    dist_to_centroid = zeros(num_samples,1);
    is_inside = zeros(num_samples,1);
    data.P12 = zeros(2,num_samples);
    data.P13 = zeros(2,num_samples);
    data.P23 = zeros(2,num_samples);
    for ii = 1:num_samples % For each sample
        theta1_noisy = get_noisy_bearing(theta1, beta1, sigma(jj));
        theta2_noisy = get_noisy_bearing(theta2, beta2, sigma(jj));
        theta3_noisy = get_noisy_bearing(theta3, beta3, sigma(jj));
        % Calculate 3 estimated positions based on noisy bearing measurments
```

```
        % positions of lighthouses are wrt to w
        P12 = get_pos_fix_2d(theta1_noisy, theta2_noisy, r_L1_w, r_L2_w);
        P13 = get_pos_fix_2d(theta1_noisy, theta3_noisy, r_L1_w, r_L3_w);
        P23 = get_pos_fix_2d(theta2_noisy, theta3_noisy, r_L2_w, r_L3_w);
        % Determine whether actual position is inside the triangle
        is_inside(ii) = is_inside_triangle(r_P_w, P12, P13, P23);
        % Determine distance from triangle centroid to actual position
        dist_to_centroid(ii) = norm(mean([P12, P13, P23], 2) - r_P_w);
        % Save these results somewhere for use outside current loop
        data.P12(:,ii) = P12(1:2);
        data.P13(:,ii) = P13(1:2);
        data.P23(:,ii) = P23(1:2);
    end
    % figure;
    % plot(data.P12(1,:), data.P12(2,:), 'y.', 'DisplayName', 'P12');
    % hold on;
    % plot(data.P13(1,:), data.P13(2,:), 'm.', 'DisplayName', 'P13');
    % plot(data.P23(1,:), data.P23(2,:), 'c.', 'DisplayName', 'P23');
    % plot(r_P_w(1), r_P_w(2), 'rx', 'DisplayName', 'P', 'MarkerSize', 10);
    % plot(r_L1_w(1), r_L1_w(2), 'go', 'DisplayName', 'L1', 'MarkerSize', 10);
    % plot(r_L2_w(1), r_L2_w(2), 'go', 'DisplayName', 'L2', 'MarkerSize', 10);
    % plot(r_L3_w(1), r_L3_w(2), 'go', 'DisplayName', 'L3', 'MarkerSize', 10);
    % xlabel('x'); ylabel('y');
    % grid on; grid minor;
    % legend('Location', 'best');
    % title("$\sigma = " + sigma(jj) + "$", 'Interpreter', 'latex');
    % Use data obtained in inner loop to calculate average distance from
    % triangle centroid to actual position and the percentage of times the
    % actual position was inside the triangle formed by the estimated
positions.
    prob_inside(jj) = sum(is_inside)/num_samples * 100;
    avg_dist(jj) = mean(dist_to_centroid);
end
```

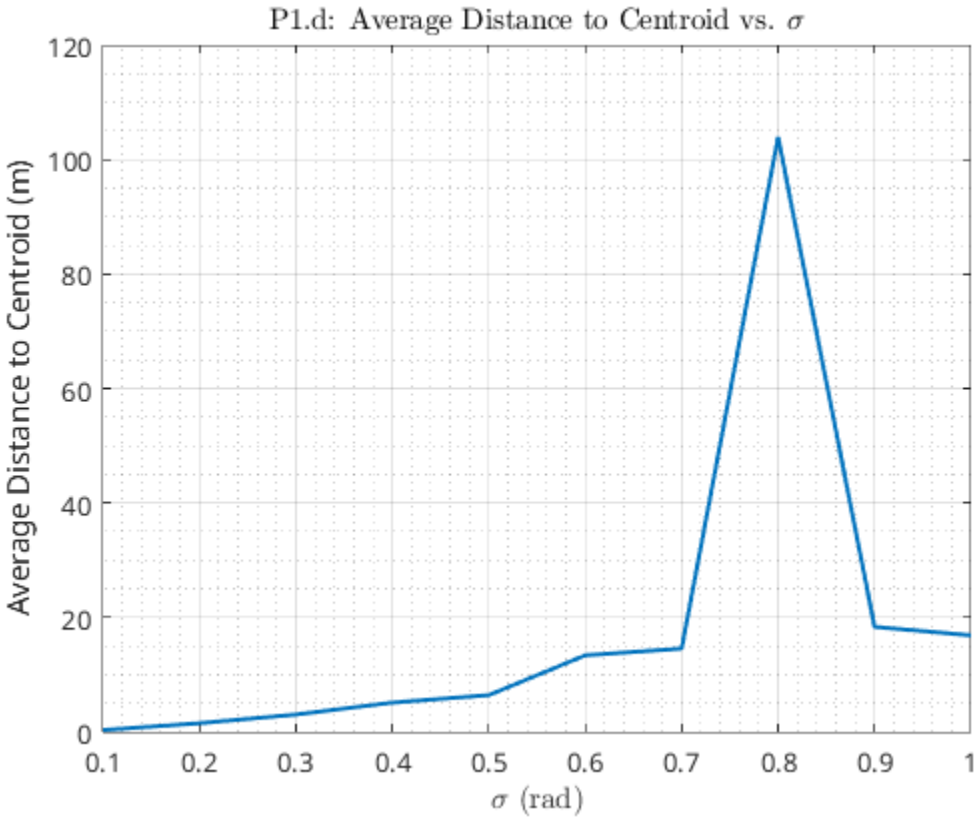
part c plot p inside versus sigma

```
figure;
plot(sigma, prob_inside, "LineWidth", 1.5);
xlabel('$\sigma$ (rad)', 'Interpreter', 'latex');
ylabel('Percentage Inside Triangle (%)');
grid on; grid minor;
title('P1.c: Percentage Inside Triangle vs. $\sigma$', 'Interpreter',
'latex');
```



part d plot avg dist versus sigma

```
figure;
plot(sigma, avg_dist, "LineWidth", 1.5);
xlabel('$\sigma$ (rad)', 'Interpreter', 'latex');
ylabel('Average Distance to Centroid (m)');
grid on; grid minor;
title('P1.d: Average Distance to Centroid vs. $\sigma$', 'Interpreter',
'latex');
```



tests

fprintf("angle between\n"); get_angle_between([1;0;0], [0;1;0]) % pi/2 get_angle_between([1;0;0], [1;0;0]) % 0 get_angle_between([1;1;0], [1;0;0]) % pi/4 get_angle_between([2;2;0], [0;1;0]) % pi/4

```
% fprintf("triange area\n");
% get_triangle_area([0;0;0], [1;0;0], [0;1;0]) % 0.5
% get_triangle_area([0;0;0], [2;0;0], [0;1;0]) % 1
% get_triangle_area([0;1;0], [2;0;0], [0;0;0]) % 1
% get_triangle_area([1;1;0], [2;2;0], [0;2;0]) % 1

% fprintf("is inside triangle\n");
% is_inside_triangle([0.1;0.1;0], r_L1_w, r_L2_w, r_L3_w) % 1
% is_inside_triangle([0.5;0.5;0], r_L1_w, r_L2_w, r_L3_w) % 1
% is_inside_triangle([0.6;0.6;0], r_L1_w, r_L2_w, r_L3_w) % 0
```

functions

```
function angle = get_angle_between(v1, v2)
raw_angle = atan2(norm(cross(v1, v2)), dot(v1, v2));
cross_prod = cross(v1, v2);
angle = sign(cross_prod(3))*raw_angle;
end

function t = T(theta)
```

```
t = tan(pi/2 - theta);
end

function pos = get_pos_fix_2d(theta1, theta2, L1, L2)
T1 = T(theta1); T2 = T(theta2);
A = [T1, -1; T2, -1];
B = [T1*L1(1) - L1(2); T2*L2(1) - L2(2)];
pos = A\B;
pos = [pos;0];
end

function noisy = get_noisy_bearing(theta, bias ,std_dev)
noisy = theta + bias + std_dev*randn;
end

function area = get_triangle_area(A, B, C)
% Calculate area of triangle ABC using heron's formula
a = norm(B - C);
b = norm(A - C);
c = norm(A - B);
s = (a + b + c)/2;
area = sqrt(s*(s - a)*(s - b)*(s - c));
end

function isin = is_inside_triangle(P, A, B, C)
% Determine if point P is inside triangle ABC
area = get_triangle_area(A, B, C);
A1 = get_triangle_area(P, A, B);
A2 = get_triangle_area(P, B, C);
A3 = get_triangle_area(P, A, C);
rhs = (A1 + A2 + A3);
isin = abs(area - rhs) <= 1e-6;
end
```

```
P1.a: true bearing angles
- theta1: -2.246
- theta2: 1.571
- theta3: -0.644
```

Published with MATLAB® R2023b

Table of Contents

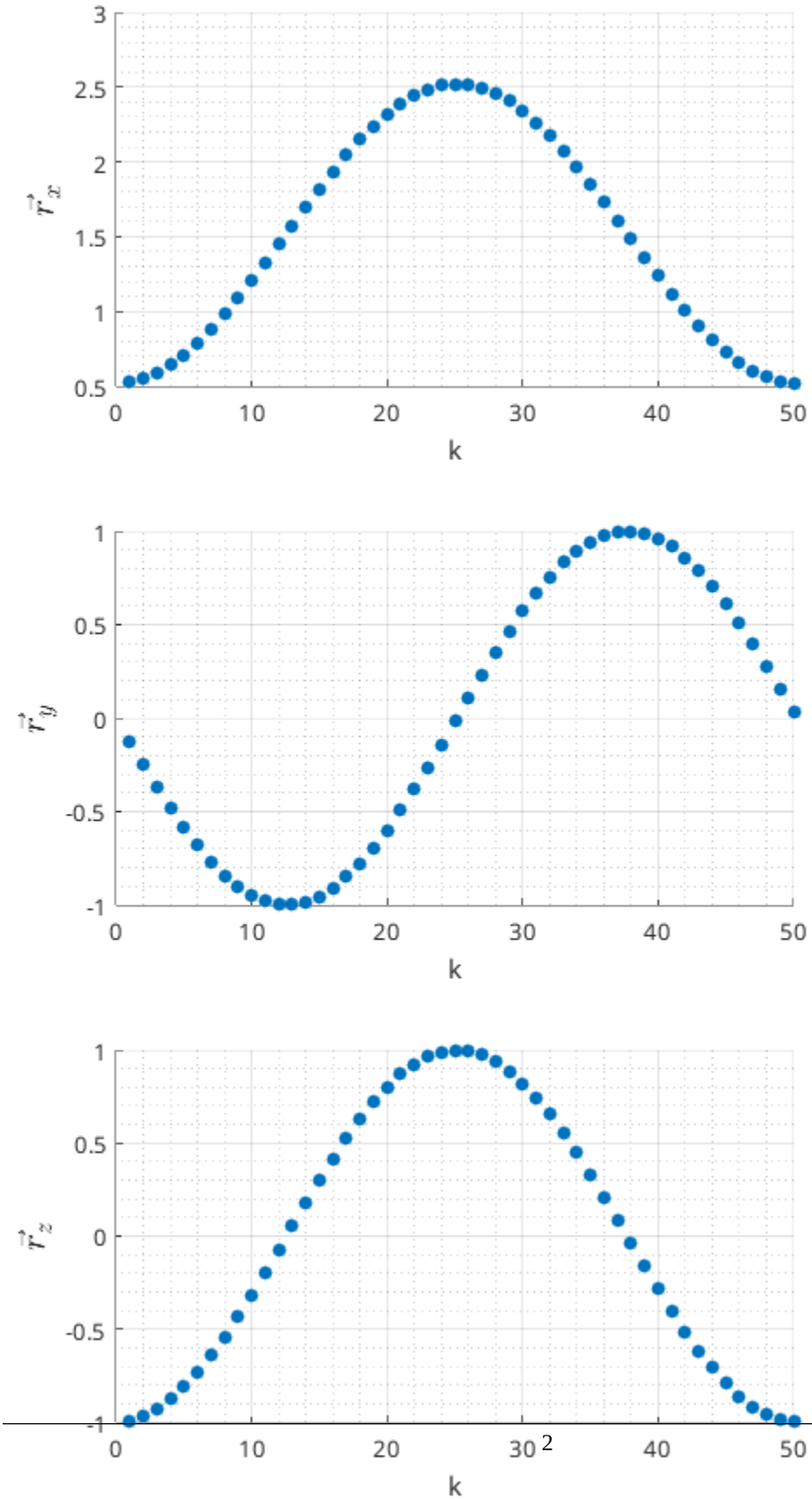
..... 1
plot 1
functions 3

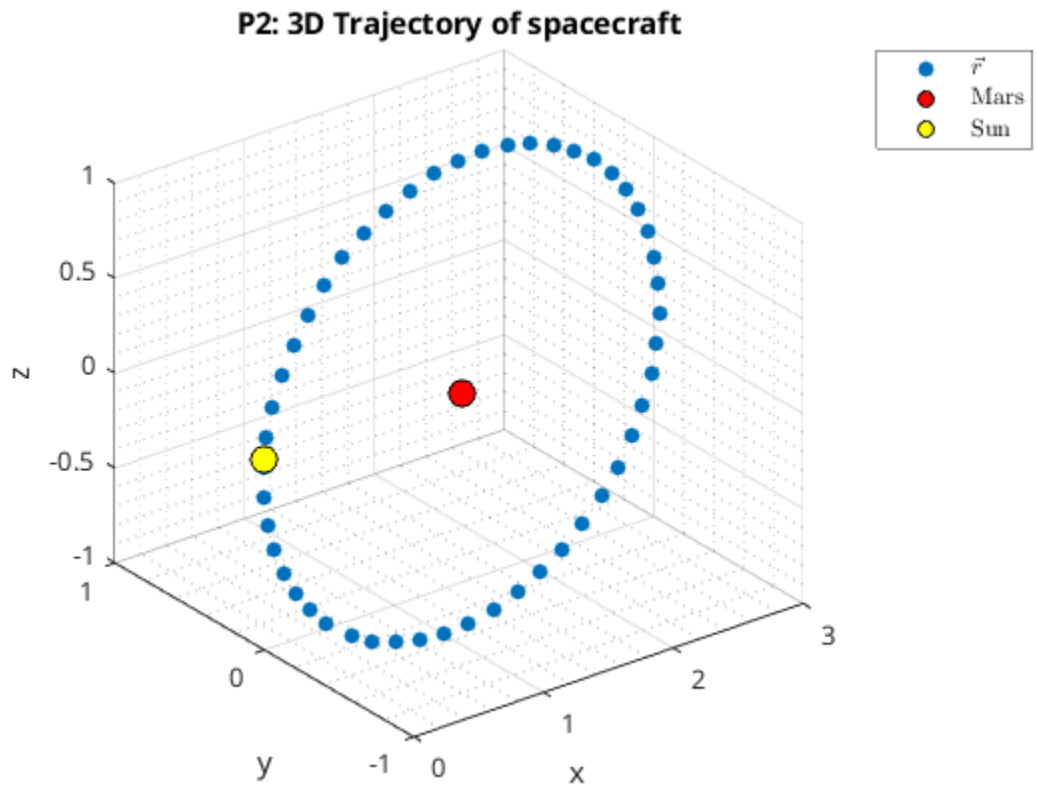

```
clc; clear; close all;

r_L1_w = [1.52; 0; 0]; % Mars
r_L2_w = [0; 0; 0]; %  Sun

s1 = [0;1;0]; s2 = [0;0;1];

k = 1:50;

% load bearings and subtended angles
load("AE584_Midterm_P2.mat");

% bearingL2St1: bearing from L2 to S1, phi2_s1;
% bearingL2St2: bearing from L2 to S2, pih2_s2;
% subAngL1L2: subtended angle between L1 and L2, theta_L1L2;

r_0 = [0.52,0,-1]; % initial guess
Pk = zeros(3, length(k));
options = optimoptions('fminunc', 'OptimalityTolerance', 1e-12, 'Display',
'off');
for i=1:length(k)
    r = fminunc(@(x) costfn(x, s1, s2, r_L1_w, r_L2_w, subAngL1L2(i),
bearingL2St1(i), bearingL2St2(i)), r_0, options);
    Pk(:,i) = r;
    r_0 = r;
end
```

plot

```
plot_components(Pk);
plot_3d(Pk, r_L1_w, r_L2_w);
```

P2: Components of $\vec{r}$ versus step k





functions

```
function cost = costfn(x, S1, S2, L1, L2, theta, psi1, psi2)
% setup
x = x'; % fix x's dims as used by fminunc
r_S1_Y = S1;
r_S2_Y = S2;
r_Y_L1 = x-L1;
r_L2_L1 = L2-L1;
r_L2_Y = L2-x;

% eq4
l4 = dot(r_Y_L1, r_L2_L1);
r4 = - norm(r_Y_L1) * norm(r_L2_Y) * cos(theta) + norm(r_Y_L1)^2;
err_subt_angle = l4-r4;

% eq5
l5 = dot(r_L2_Y, r_S1_Y);
r5 = norm(r_L2_Y) * cos(psi1);
err_bearing1 = l5-r5;

% eq6
l6 = dot(r_L2_Y, r_S2_Y);
r6 = norm(r_L2_Y) * cos(psi2);
err_bearing2 = l6-r6;
```

```
cost = err_subt_angle^2 + err_bearing1^2 + err_bearing2^2;
end

function plot_components(Pk)
fig = figure;
fig.Position(3:4) = [500, 1000];

sgtitle("P2: Components of $\vec{r}$ versus step k", 'Interpreter', 'latex');

subplot(3,1,1);
scatter(1:length(Pk), Pk(1,:), 'filled');
xlabel("k"); ylabel("$\vec{r}_x$", "Interpreter", "latex", "FontSize", 16);
grid on; grid minor;

subplot(3,1,2);
scatter(1:length(Pk), Pk(2,:), 'filled');
xlabel("k"); ylabel("$\vec{r}_y$", "Interpreter", "latex", "FontSize", 16);
grid on; grid minor;

subplot(3,1,3);
scatter(1:length(Pk), Pk(3,:), 'filled');
xlabel("k"); ylabel("$\vec{r}_z$", "Interpreter", "latex", "FontSize", 16);
grid on; grid minor;

end

function plot_3d(Pk, L1, L2)
figure;
s1 = scatter3(Pk(1,:), Pk(2,:), Pk(3,:), 'filled');
hold on;
s2 = scatter3(L1(1), L1(2), L1(3), 100, 'r', 'filled', 'MarkerEdgeColor',
'k');
s3 = scatter3(L2(1), L2(2), L2(3), 100, 'y', 'filled', 'MarkerEdgeColor',
'k');
hold off;
xlabel("x"); ylabel("y"); zlabel("z");
legend([s1, s2, s3], {'$\vec{r}$', 'Mars', 'Sun'}, 'Interpreter', 'latex');
grid on; grid minor;
title("P2: 3D Trajectory of spacecraft");
end
```

Published with MATLAB® R2023b

Table of Contents

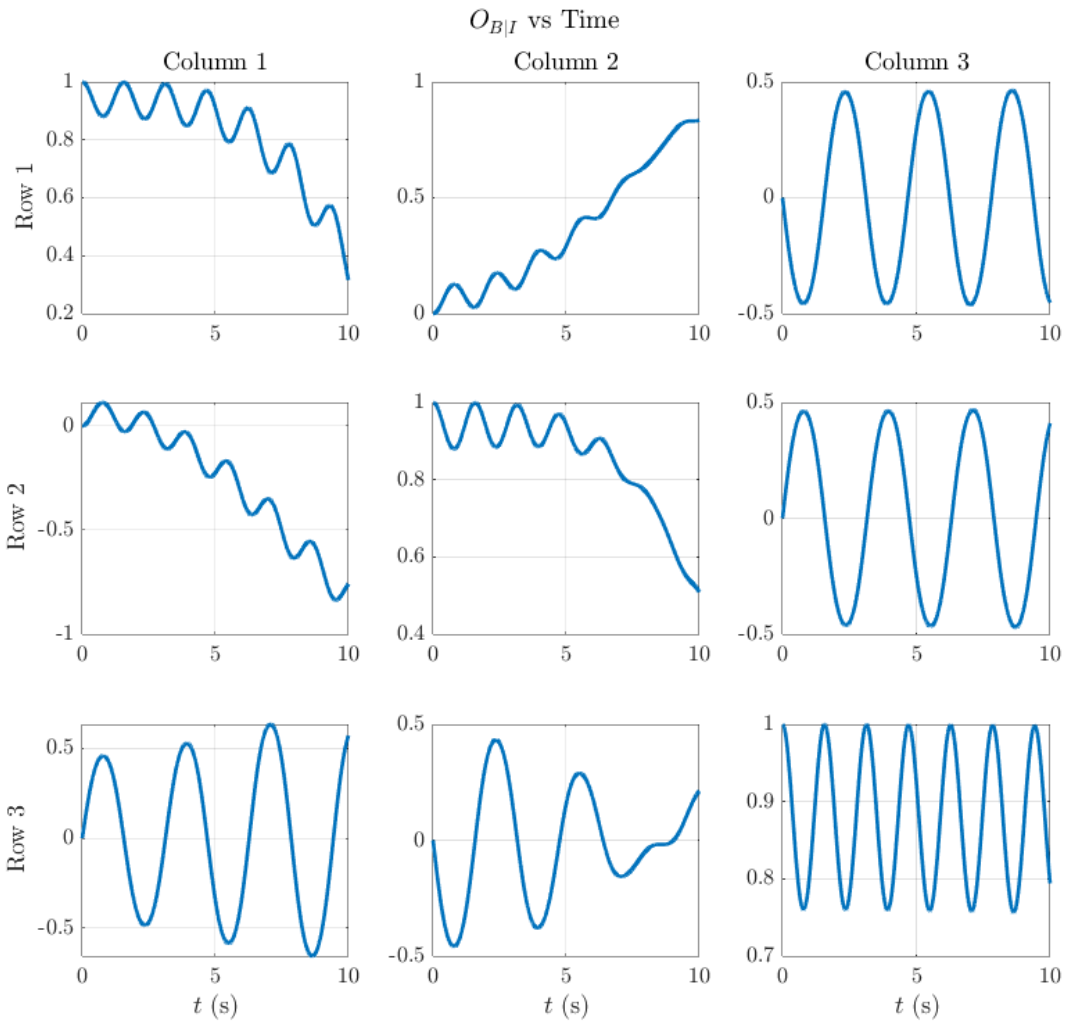
..... 1
cross matrix function 2
Orientation matrix ODE 8

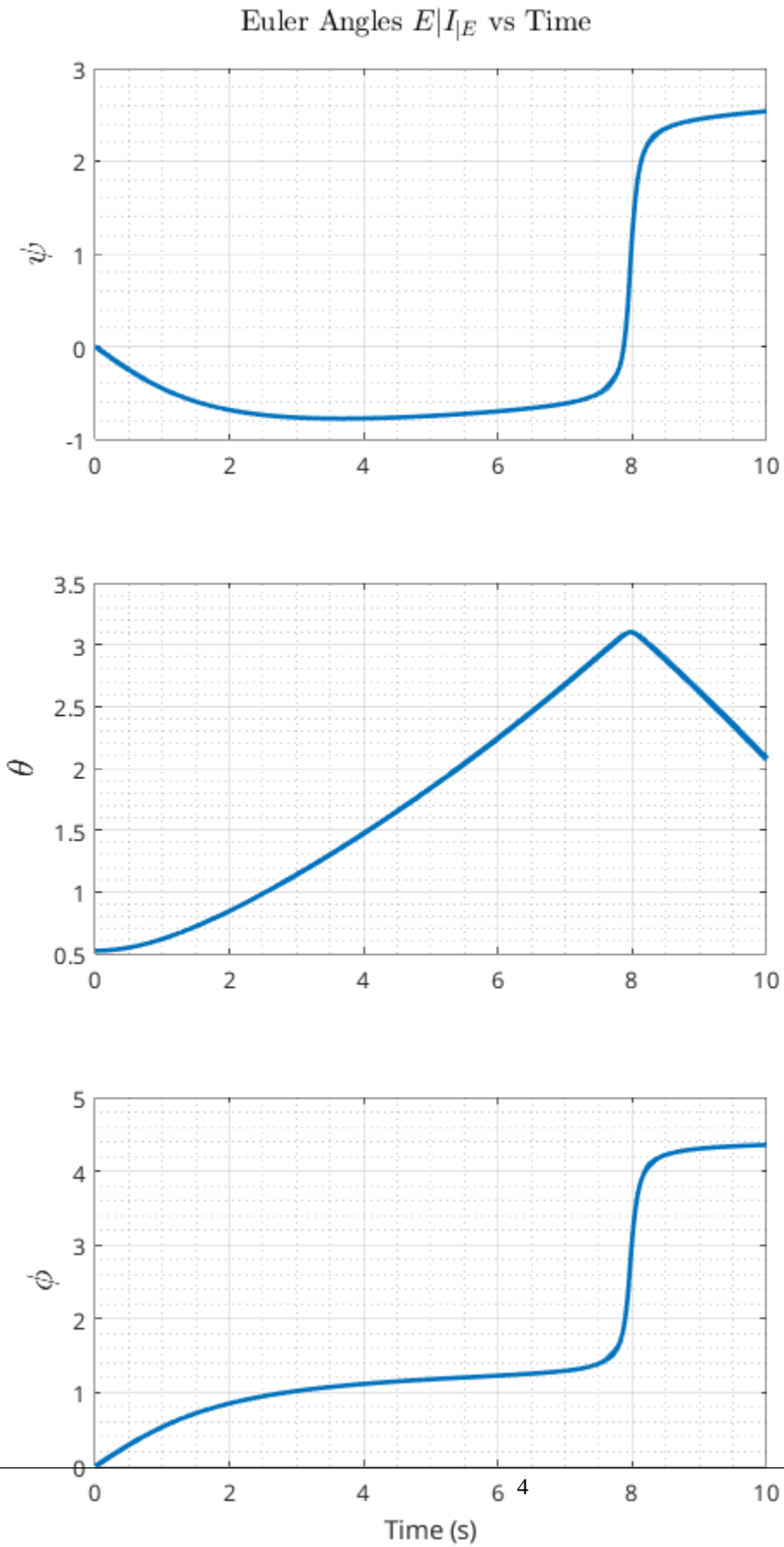

```
clc; clear; close all;

% setup sim
Tfid = 0.01;
Tsim = 10;

% plot angular velocity over time for B_I
angvels = angvel_B_I(0:Tfid:Tsim);
plot_angvel(0:Tfid:Tsim, angvels, 'Angular Velocity $B|I_{|B}$ vs Time');

% propagate orientation matrix with angular velocity for B_I
omat_BI_0 = eye(3); % start with identity matrix
omat_BI_0 = omat_BI_0(:); % vectorize the matrix columnwise
[t, omat_BI] = ode45(@(t,x)omat_dot(t,x,@angvel_B_I), 0:Tfid:Tsim, omat_BI_0);
plot_omat(t, omat_BI, '$O_{B|I}$ vs Time');

% propagate euler angles with angular velocity for E_I
theta0 = [0; pi/6; 0]; % psi, theta, phi
[t, eul_ang_EI] = ode45(@(t,x)prop_theta(t,x), 0:Tfid:Tsim, theta0);
plot_euler_angles(t, eul_ang_EI, 'Euler Angles $E|I_{|E}$ vs Time');

% convert euler angles to orientation matrix E_I
omat_EI = zeros(3,3,length(t));
omat_EI_flat = zeros(length(t), 9);
for i=1:length(t)
    omat = omat313(eul_ang_EI(i,:));
    omat_EI(:,:,i) = omat;
    omat_EI_flat(i, :) = omat(:); % plot expects it to be columnwise
end
plot_omat(t, omat_EI_flat, '$O_{E|I}$ vs Time');

% get orientation matrix for B|E
omat_BE = zeros(3,3,length(t));
omat_BE_flat = zeros(length(t), 9);
for i=1:length(t)
    omat_BI_mat = reshape(omat_BI(i,:), 3, 3); % reshape does columnwise
    omat = omat_EI(:,:,i) \ omat_BI_mat;
    omat_BE(:,:,i) = omat;
    omat_BE_flat(i, :) = omat(:); % plot expects it to be columnwise
end
plot_omat(t, omat_BE_flat, 'P3: $O_{B|E}$ vs Time');

% get euler angles for B|E
eul_ang_BE = zeros(length(t), 3);
for i=1:length(t)
```

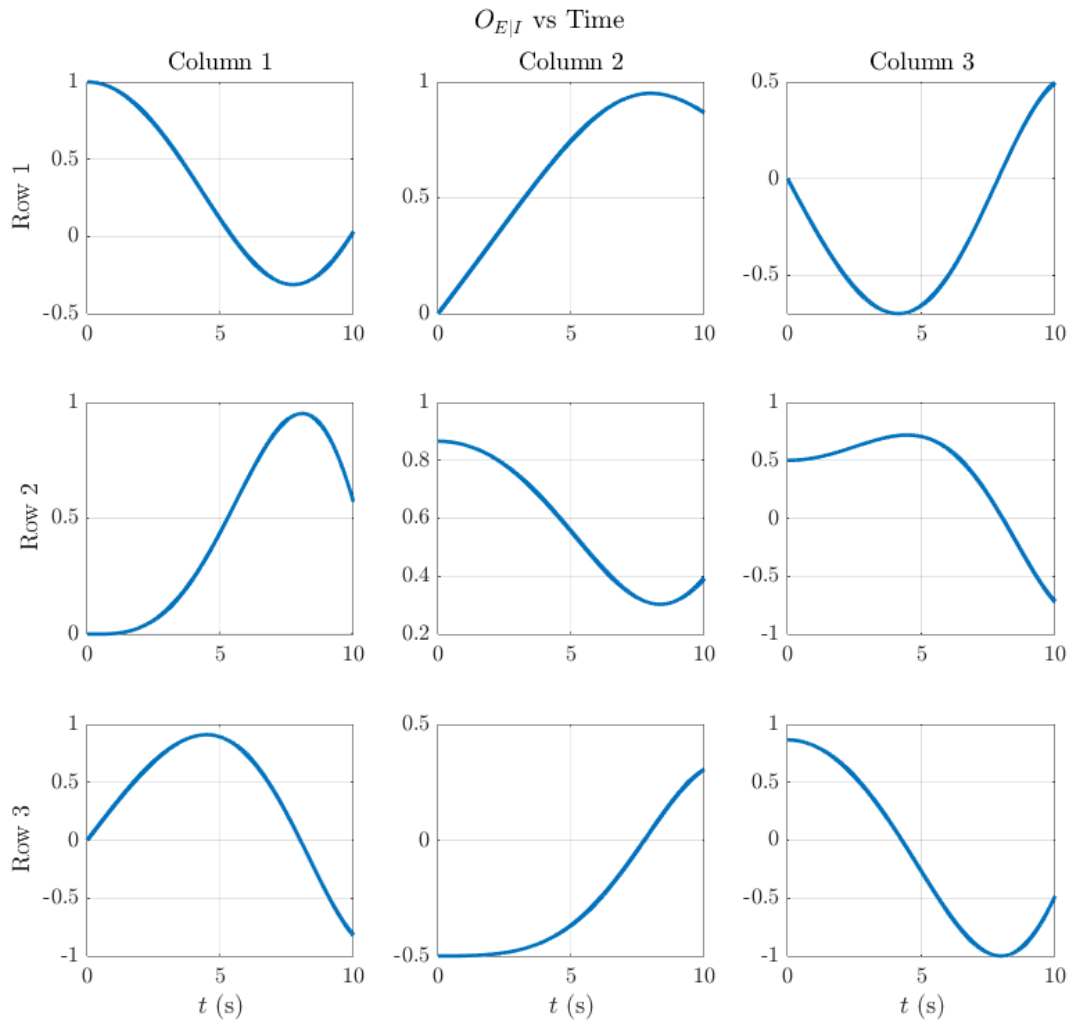
```
    [psi, theta, phi] = euler_from_omat321(omat_BE(:,:,i));
    eul_ang_BE(i,:) = [psi, theta, phi];
end
plot_euler_angles(t, eul_ang_BE, 'P3: Euler Angles vs Time');


% angular velocity for B|I
function angvel = angvel_B_I(t)
angvel = [
    cos(2*t);
    cos(2*t);
    0.025*t;
    ];
end

% angular velocity for E|I
function angvel = angvel_E_I(t)
angvel = [
    sin(0.05*t);
    0.3 * cos(0.01*t);
    0.5 * sin(0.01*t);
    ];
end
```

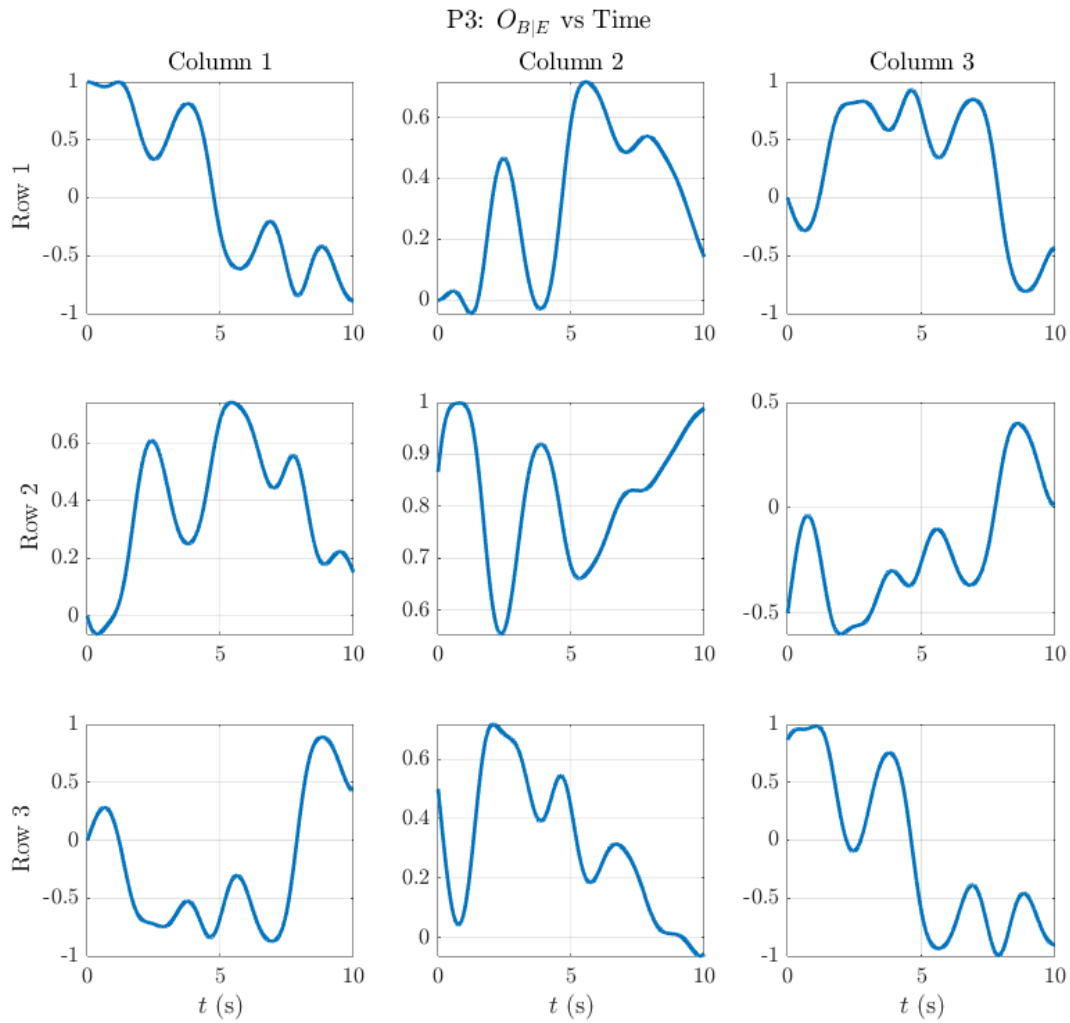
cross matrix function

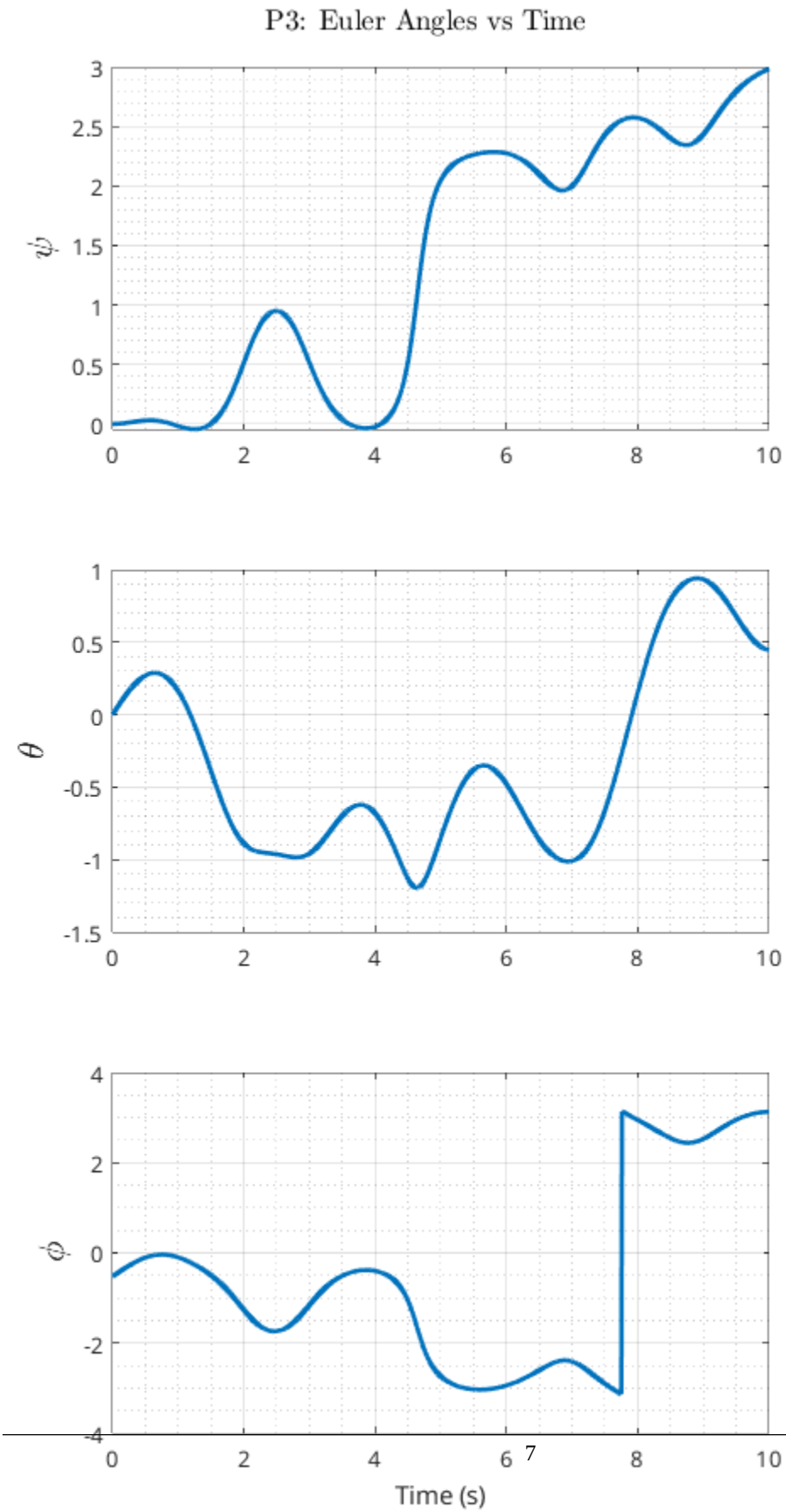
```
function crossMat = crMat(X)
crossMat = [
    0      -X(3)  X(2);
    X(3)    0    -X(1);
    -X(2)  X(1)   0;
    ];
end
```


$O_{B|I}$ vs Time
Column 1
Column 2
Column 3
Row 1
Row 2
Row 3
t (s)
t (s)
t (s)


Euler Angles $E|I_{|E}$ vs Time
ψ
θ
ϕ
Time (s)


$O_{E|I}$ vs Time
Column 1
Column 2
Column 3
Row 1
Row 2
Row 3
t (s)
t (s)
t (s)


P3: $O_{B|E}$ vs Time
Column 1
Column 2
Column 3
Row 1
Row 2
Row 3
t (s)
t (s)
t (s)


P3: Euler Angles vs Time
ψ
θ
ϕ
Time (s)

Orientation matrix ODE

```
function xdot = omat_dot(t,x,omega_d)
xT = reshape(x,3,3);
xdT = -crMat(omega_d(t))*xT;
xdot = xdT(:);
end

function theta_dot = prop_theta(t, x)
psi = x(1);
theta = x(2);
% phi = x(2);
S_inv = [
    -sin(psi)*cot(theta), -cos(psi)*cot(theta), 1;
    cos(psi), -sin(psi), 0;
    sin(psi)*csc(theta), cos(psi)*csc(theta), 0;
    ];
ang_vel = angvel_E_I(t);
theta_dot = S_inv * ang_vel;
end

function omat = omat1(x)
omat = [
    1, 0, 0;
    0, cos(x), sin(x);
    0, -sin(x), cos(x);
    ];
end

function omat = omat3(x)
omat = [
    cos(x), sin(x), 0;
    -sin(x), cos(x), 0;
    0, 0, 1;
    ];
end

function omat = omat313(x)
psi = x(1);
theta = x(2);
phi = x(3);
omat = omat3(psi) * omat1(theta) * omat3(phi);
end

function [psi, theta, phi] = euler_from_omat321(omat)
O23 = omat(2,3);
O33 = omat(3,3);
O13 = omat(1,3);
O12 = omat(1,2);
O11 = omat(1,1);

phi = atan2(O23, O33);
theta = -asin(O13);
```

```
psi = atan2(O12, O11);
end

function plot_angvel(t, angvels, title_str)
figure;
hold on;
plot(t, angvels(1,:), 'r', 'LineWidth', 2, 'DisplayName', '$\omega_i$');
plot(t, angvels(2,:), '--g', 'LineWidth', 2, 'DisplayName', '$\omega_j$');
plot(t, angvels(3,:), 'b', 'LineWidth', 2, 'DisplayName', '$\omega_k$');
ylabel('Angular Velocity (rad/s)');
title(title_str, 'Interpreter', 'latex');
xlabel('Time (s)');
legend("Location", "best", "Interpreter", "latex");
grid on; grid minor;
end


function plot_euler_angles(t, theta, title_str)
fig = figure;
fig.Position(3:4) = [500, 1000];
sgtitle(title_str, 'Interpreter', 'latex');

subplot(3,1,1);
plot(t, theta(:,1), '-', 'LineWidth', 2);
ylabel('$\psi$', 'Interpreter', 'latex', 'FontSize', 16);
grid on; grid minor;

subplot(3,1,2);
plot(t, theta(:,2), '-', 'LineWidth', 2);
ylabel('$\theta$', 'Interpreter', 'latex', 'FontSize', 16);
grid on; grid minor;

subplot(3,1,3);
plot(t, theta(:,3), '-', 'LineWidth', 2);
ylabel('$\phi$', 'Interpreter', 'latex', 'FontSize', 16);
xlabel('Time (s)');
grid on; grid minor;

end


function plot_omat(t, omat, title_str)
omat_11 = omat(:,1);
omat_12 = omat(:,4);
omat_13 = omat(:,7);
omat_21 = omat(:,2);
omat_22 = omat(:,5);
omat_23 = omat(:,8);
omat_31 = omat(:,3);
omat_32 = omat(:,6);
omat_33 = omat(:,9);

fig = figure;
fig.Position(3:4) = [900, 800];
```

```
sgtitle(title_str, 'Interpreter', 'latex');

subplot(3,3,1)
plot(t,omat_11,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
ylabel('Row 1','Interpreter','latex')
title('Column 1','Interpreter','latex')
grid on

subplot(3,3,2)
plot(t,omat_12,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
title('Column 2','Interpreter','latex')
grid on

subplot(3,3,3)
plot(t,omat_13,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
title('Column 3','Interpreter','latex')
grid on

subplot(3,3,4)
plot(t,omat_21,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
ylabel('Row 2','Interpreter','latex')
grid on

subplot(3,3,5)
plot(t,omat_22,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
grid on

subplot(3,3,6)
plot(t,omat_23,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
grid on

subplot(3,3,7)
plot(t,omat_31,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
```

```
xlabel('$t$ (s)','Interpreter','latex')
ylabel('Row 3','Interpreter','latex')
grid on

subplot(3,3,8)
plot(t,omat_32,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
xlabel('$t$ (s)','Interpreter','latex')
grid on

subplot(3,3,9)
plot(t,omat_33,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
xlabel('$t$ (s)','Interpreter','latex')
grid on

end
```

Published with MATLAB® R2023b

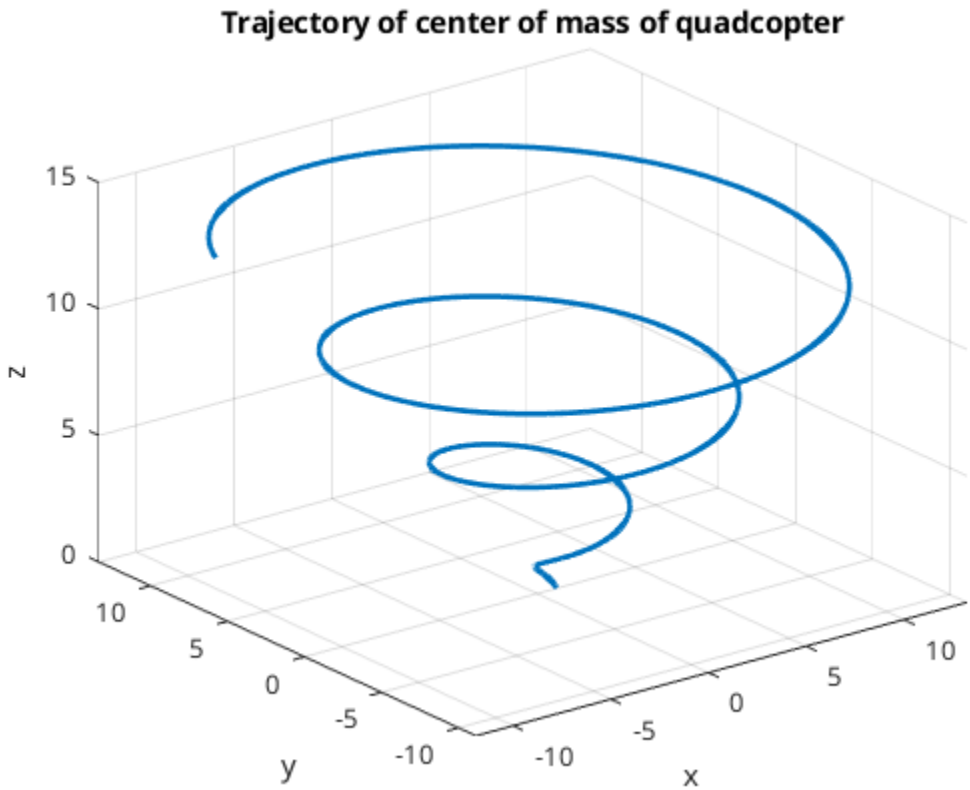
Table of Contents

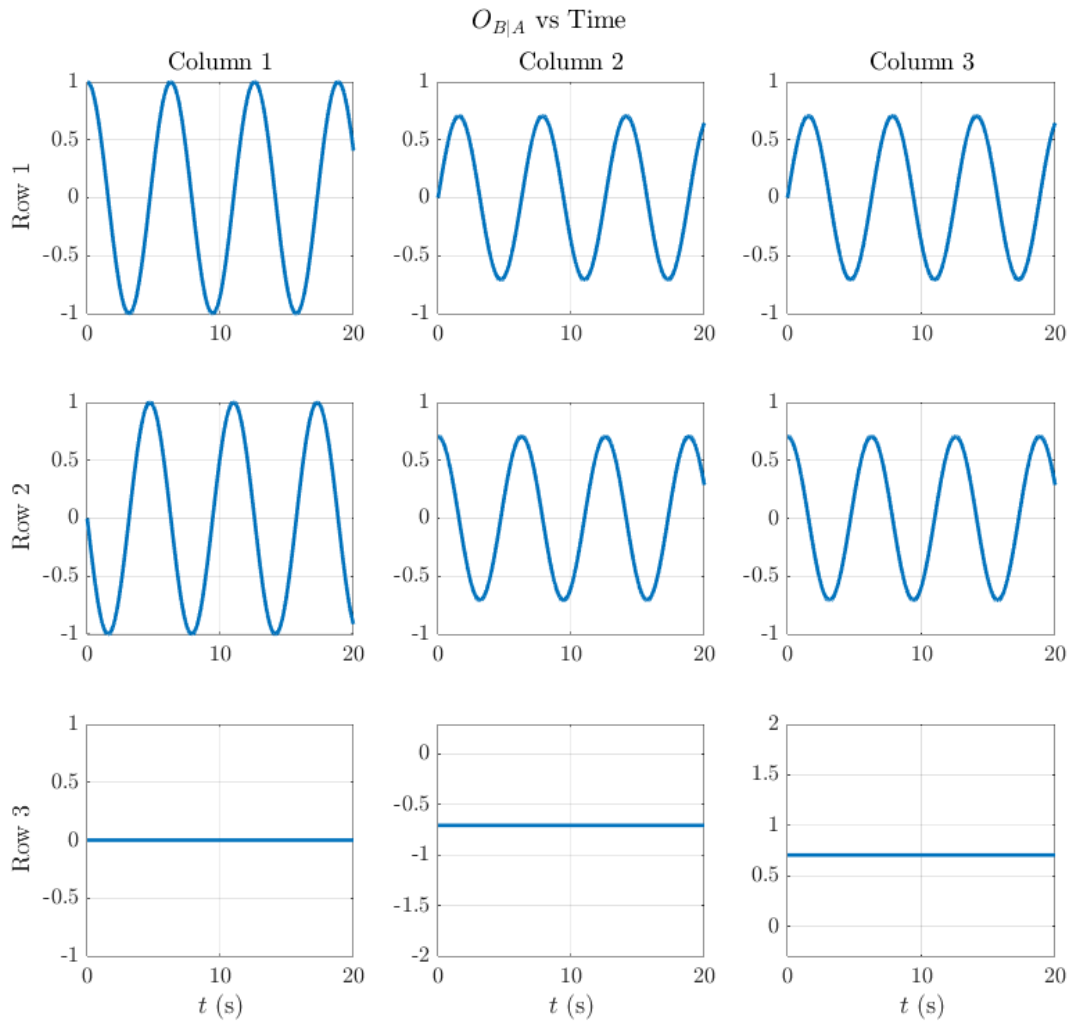
..... 1
cross matrix function 2
Orientation matrix ODE 3


```
clc; clear; close all;

% consts
g = 9.80665; % m/s^2
alpha  = pi/4; % rad

% setup sim
Tfid = 0.01;
Tsim = 20;

% propagate orientation matrix with angular velocity for B_A
r_0 = [1;0;0]; % m
rdot_0 = [0;cos(alpha);sin(alpha)]; % m/s
omat_BA_0 = [
    1, 0,           0;
    0, cos(alpha),  sin(alpha);
    0, -sin(alpha), cos(alpha);
    ];
% display(omat_BA_0);
omat_BA_0 = omat_BA_0(:); % plot and omat_dot expects it columnwise as in
example_posson_equation.m
% display(omat_BA_0);
x0 = [ % extended state
    r_0;
    rdot_0;
    omat_BA_0;
    ];

% [t, x] = ode45(@(t,x) dynamics1(t, x), 0:Tfid:Tsim, x0);
[t, x] = ode45(@(t,x) dynamics(t, x, g, alpha), 0:Tfid:Tsim, x0);

r = x(:,1:3);
plot_traj(r);
omat_BA = x(:,7:15);
plot_omat(t, omat_BA, '$O_{B|A}$ vs Time');
% test = dynamics(0.1, x0, g, alpha);


function omega = get_omega_BA(~)
omega = [0;0;1];
end
```

cross matrix function

```
function crossMat = crMat(X)
crossMat = [
    0       -X(3)  X(2);
    X(3)     0    -X(1);
    -X(2)  X(1)    0;
    ];
end
```





Orientation matrix ODE

```
function xdot = omat_dot(t,x,omega_d)
xT = reshape(x,3,3);
xdT = -crMat(omega_d(t))*xT;
xdot = xdT(:);
end

function acc = get_acc(g, alpha, t)
acc = [
    -1-g*sin(alpha)*sin(t);
    -g*sin(alpha)*cos(t);
    -g*cos(alpha);
    ];
end

function xdot = dynamics(t, x, g, alpha)
```

```
% extract states from x
r_A = x(1:3);
rdot_A = x(4:6);
omat_BA = x(7:15); % flat here cuz omat_dot expects it flat, columnwise
omat_BA_dot = omat_dot(t, omat_BA, @get_omega_BA); % calculate first for
returning

% get rdot_B from rdot_A and omat_BA (transport theorem)
omega_BA = get_omega_BA(t);
omat_BA = reshape(omat_BA, 3, 3); % reshape does columnwise by default
rdot_B = rdot_A - crMat(omega_BA)*r_A;

% get everything else for double transport
rddot_B = get_acc(g, alpha, t) - omat_BA*[0;0;-g];
omega_BA = get_omega_BA(t);
omega_BA_cross = crMat(omega_BA);
rddot_A = rddot_B + 2*omega_BA_cross*rdot_B +
omega_BA_cross*(omega_BA_cross*r_A);

xdot = [
    rdot_A;
    rddot_A;
    omat_BA_dot;
    ];
end

function plot_omat(t, omat, title_str)
omat_11 = omat(:,1);
omat_12 = omat(:,4);
omat_13 = omat(:,7);
omat_21 = omat(:,2);
omat_22 = omat(:,5);
omat_23 = omat(:,8);
omat_31 = omat(:,3);
omat_32 = omat(:,6);
omat_33 = omat(:,9);

fig = figure;
fig.Position(3:4) = [900, 800];
sgtitle(title_str, 'Interpreter', 'latex');

subplot(3,3,1)
plot(t,omat_11,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
ylabel('Row 1','Interpreter','latex')
title('Column 1','Interpreter','latex')
grid on

subplot(3,3,2)
plot(t,omat_12,'linewidth',2)
hold off
set(gca,'FontSize',12)
```

```
set(gca,'TickLabelInterpreter','latex');
title('Column 2','Interpreter','latex')
grid on

subplot(3,3,3)
plot(t,omat_13,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
title('Column 3','Interpreter','latex')
grid on

subplot(3,3,4)
plot(t,omat_21,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
ylabel('Row 2','Interpreter','latex')
grid on

subplot(3,3,5)
plot(t,omat_22,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
grid on

subplot(3,3,6)
plot(t,omat_23,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
grid on

subplot(3,3,7)
plot(t,omat_31,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
xlabel('$t$ (s)','Interpreter','latex')
ylabel('Row 3','Interpreter','latex')
grid on

subplot(3,3,8)
plot(t,omat_32,'linewidth',2)
hold off
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
xlabel('$t$ (s)','Interpreter','latex')
grid on

subplot(3,3,9)
plot(t,omat_33,'linewidth',2)
hold off
```

```
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
xlabel('$t$ (s)','Interpreter','latex')
grid on

end


function plot_traj(r)
figure;
plot3(r(:,1), r(:,2), r(:,3), 'linewidth', 2);
xlabel('x');
ylabel('y');
zlabel('z');
title('Trajectory of center of mass of quadcopter');
grid on;
end
```

Published with MATLAB® R2023b

Table of Contents

..... 1

```
clc; clear; close all;

% set up
T = 1;
A = [
    1, T;
    0, 1;
    ];
b = [0;0];

mu = [0;0]; % E[X]
P = [
    0.5,0;
    0,0.5;
    ];

% Y = AX + b, X = N(mu, P)

% Expectation: E[Y] = A*mu + b
E_Y = A*mu + b;
fprintf('E[Y]:\n');
disp(E_Y);

% Covariance: Cov[Y] = A*P*A'
Cov_Y = A*P*A';
fprintf('Cov[Y]:\n');
disp(Cov_Y);
```

```
E[Y]:
     0
     0

Cov[Y]:
    1.0000    0.5000
    0.5000    0.5000
```

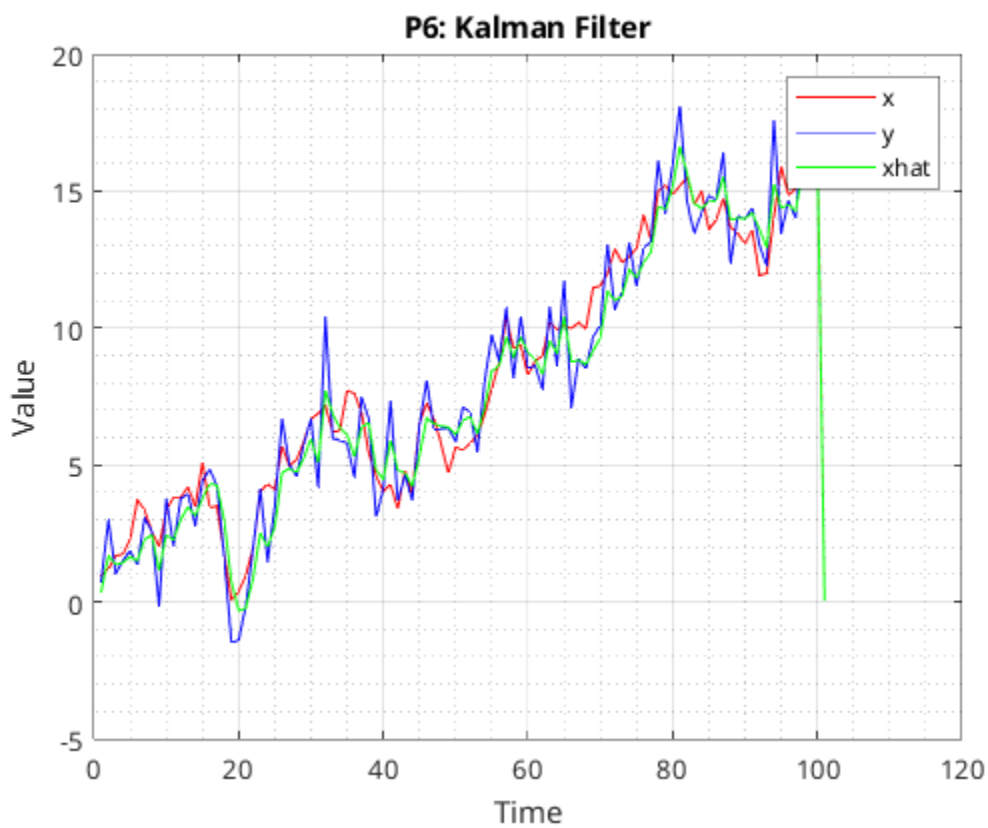
Published with MATLAB® R2023b

Table of Contents

..... 1

```
clc; clear; close all;

Q = 1;
R = 2;

Tsteps = 100;

x0_hat = 0;
P0 = 1;

data.x = zeros(1, Tsteps+1);
data.x(1) = x0_hat;
data.y = zeros(1, Tsteps+1);
data.y(1) = x0_hat;
data.xhat = zeros(1, Tsteps+1);
data.xhat(1) = x0_hat;

xk = x0_hat;
xk_hat = x0_hat;
Pk = P0;

for i=1:Tsteps
    xkp1 = xk1(xk, Q);
    [xhat_p, P_p] = predict(xk_hat, Pk, Q);
    ykp1 = yk1(xkp1, R);
    [xkp1_hat, Pk] = update(xhat_p, P_p, ykp1, R);
    xk = xkp1;
    xk_hat = xkp1_hat;
    data.x(i) = xkp1;
    data.y(i) = ykp1;
    data.xhat(i) = xkp1_hat;
end

figure;
plot(1:Tsteps+1, data.x, 'r', "DisplayName", "x");
hold on;
plot(1:Tsteps+1, data.y, 'b', "DisplayName", "y");
plot(1:Tsteps+1, data.xhat, 'g', "DisplayName", "xhat");
legend;
xlabel('Time');
ylabel('Value');
grid on; grid minor;
title('P6: Kalman Filter');

function x = xk1(xk, cov)
x = xk + randn(1)*sqrt(cov);
end
```

```
function y = yk1(xk, cov)
y = xk + randn(1)*sqrt(cov);
end

function [xhat_p, P_p] = predict(xhat_u, P_u, Q)
xhat_p = xhat_u;
P_p = P_u + Q;
end

function [xhat_u, P_u] = update(xhat_p, P_p, yk, R)
K = P_p/(P_p + R);
xhat_u = xhat_p + K*(yk - xhat_p);
P_u = P_p - K*P_p;
end
```



Published with MATLAB® R2023b