Contents


```
clc
clear all
close all
```

Initial Conditions

```
phi = 0;
theta = pi/2;
psi = 0;

W_ic = [.2 .2 1]';
```

Time Periods

```
tstep = 1000;
control_time = 25;
waitingperiod = (control_time*3)/tstep;

tspan_con = linspace(0,control_time,tstep);

tspan_detumble = linspace(control_time,control_time*2,tstep);

tspan_phi=linspace(control_time*2,control_time*2.5,tstep);

tspan_theta=linspace(control_time*2.5,control_time*3,tstep);

tspan_psi=linspace(control_time*3,control_time*4,tstep);

syms W1 W2 W3
```

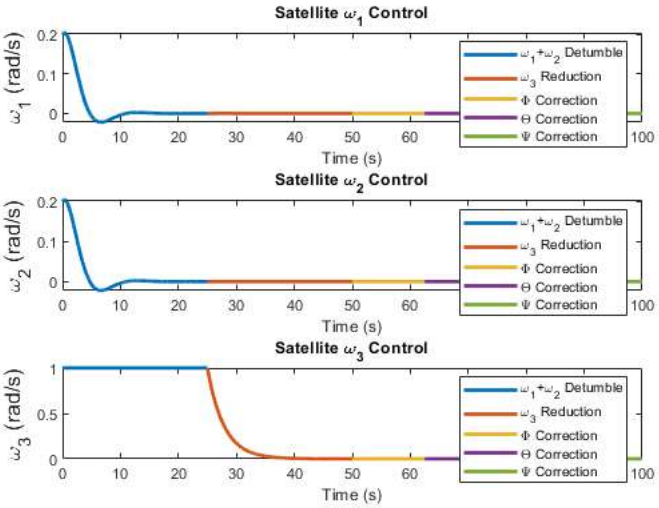
ODE45

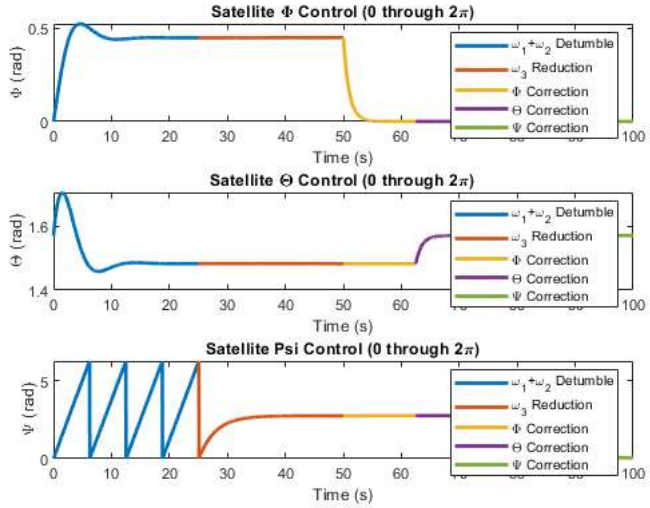
```
%ODE 45: W1 and W2 Reduction
[t, W1_W2_Detumble] = ode45(@(t,state) controller1(t,state), tspan_con, [W_ic(1) W_ic(2) W_ic(3) phi theta psi]);

%ODE45: W3 Reduction, using final conditions of previous ODE for continuity
[t2, W3_Detumble] = ode45(@(t,state) detumble(t,state), tspan_detumble, [W1_W2_Detumble(end,1) W1_W2_Detumble(end,2) W1_W2_Detumble(end,3) W1_W2_Detumble(end,4) W1_

%ODE 45: Angle Correction, using final conditions of previous ODE for continuity
[t3, Angle_Correction_Phi] = ode45(@(t,state) angle_phi(t,state), tspan_phi, [W3_Detumble(end,1) W3_Detumble(end,2) W3_Detumble(end,3) W3_Detumble(end,4) W3_Detumbl

%ODE 45: Angle Correction, using final conditions of previous ODE for continuity
[t3, Angle_Correction_Theta] = ode45(@(t,state) angle_theta(t,state), tspan_theta, [Angle_Correction_Phi(end,1) Angle_Correction_Phi(end,2) Angle_Correction_Phi(end

%ODE 45: Angle Correction, using final conditions of previous ODE for continuity
[t3, Angle_Correction_Psi] = ode45(@(t,state) angle_psi(t,state), tspan_psi, [Angle_Correction_Theta(end,1) Angle_Correction_Theta(end,2) Angle_Correction_Theta(end
```

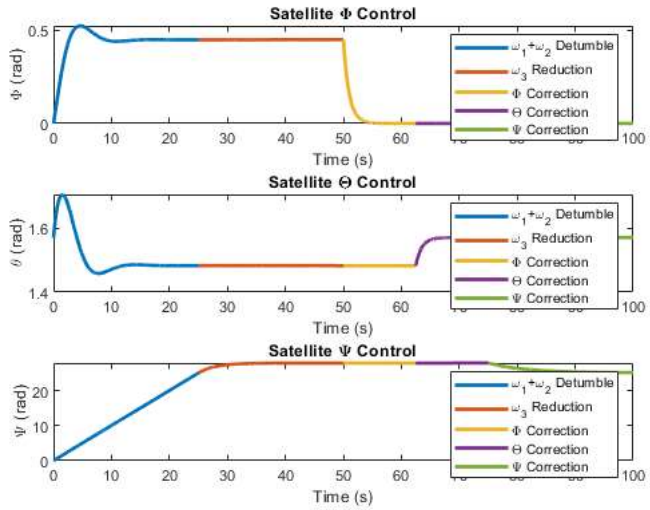
Plots

```
%Plot W1 Control Timeline
subplot(3,1,1)
plot(tspan_con,W1_W2_Detumble(1:tstep,1),tspan_detumble,W3_Detumble(1:tstep,1),tspan_phi,Angle_Correction_Phi(1:tstep,1),tspan_theta,Angle_Correction_Theta(1:tstep,
xlabel('Time (s)')
ylabel('\omega_1 (rad/s)', 'FontSize',12)
legend('\omega_1+\omega_2 Detumble','\omega_3 Reduction','\Phi Correction','\Theta Correction','\Psi Correction')
title('Satellite \omega_1 Control')

%Plot W2 Control Timeline
subplot(3,1,2)
plot(tspan_con, W1_W2_Detumble(1:tstep,1),tspan_detumble, W3_Detumble(1:tstep,1),tspan_phi,Angle_Correction_Phi(1:tstep,2),tspan_theta,Angle_Correction_Theta(1:tste
xlabel('Time (s)')
ylabel('\omega_2 (rad/s)', 'FontSize',12)
legend('\omega_1+\omega_2 Detumble','\omega_3 Reduction','\Phi Correction','\Theta Correction','\Psi Correction')
title('Satellite \omega_2 Control')

%Plot W3 Control Timeline
subplot(3,1,3)
plot(tspan_con,W1_W2_Detumble(1:tstep,3),tspan_detumble,W3_Detumble(1:tstep,3),tspan_phi,Angle_Correction_Phi(1:tstep,3),tspan_theta,Angle_Correction_Theta(1:tstep,
```

```
xlabel('Time (s)')
ylabel('\omega_3 (rad/s)', 'FontSize',12)
legend('\omega_1+\omega_2 Detumble','\omega_3 Reduction','\Phi Correction','\Theta Correction','\Psi Correction')
title('Satellite \omega_3 Control')


%Phi Control Timeline (0-2pi)
figure(2)
subplot(3,1,1)
plot(tspan_con,rem(W1_W2_Detumble(1:tstep,4)/(2*pi),1)*2*pi,tspan_detumble,rem(W3_Detumble(1:tstep,4)/(2*pi),1)*2*pi,tspan_phi,rem(Angle_Correction_Phi(1:tstep,4)/(
xlabel('Time (s)')
ylabel('\Phi (rad)')
legend('\omega_1+\omega_2 Detumble','\omega_3 Reduction','\Phi Correction','\Theta Correction','\Psi Correction')
title('Satellite \Phi Control (0 through 2\pi)')

%Theta Control Timeline (0-2pi)
subplot(3,1,2)
plot(tspan_con, rem(W1_W2_Detumble(1:tstep,5)/(2*pi),1)*2*pi,tspan_detumble,rem(W3_Detumble(1:tstep,5)/(2*pi),1)*2*pi,tspan_phi,rem(Angle_Correction_Phi(1:tstep,5)/
xlabel('Time (s)')
ylabel('\Theta (rad)')
legend('\omega_1+\omega_2 Detumble','\omega_3 Reduction','\Phi Correction','\Theta Correction','\Psi Correction')
title('Satellite \Theta Control (0 through 2\pi)')

%Psi Control Timeline (0-2pi)
subplot(3,1,3)
plot(tspan_con,rem(W1_W2_Detumble(1:tstep,6)/(2*pi),1)*2*pi,tspan_detumble,rem(W3_Detumble(1:tstep,6)/(2*pi),1)*2*pi,tspan_phi,rem(Angle_Correction_Phi(1:tstep,6)/(
xlabel('Time (s)')
ylabel('\Psi (rad)')
legend('\omega_1+\omega_2 Detumble','\omega_3 Reduction','\Phi Correction','\Theta Correction','\Psi Correction')
title('Satellite Psi Control (0 through 2\pi)')


%Phi Control Timeline Total Magnitude
figure(3)
subplot(3,1,1)
plot(tspan_con,W1_W2_Detumble(1:tstep,4),tspan_detumble,W3_Detumble(1:tstep,4),tspan_phi,Angle_Correction_Phi(1:tstep,4),tspan_theta,Angle_Correction_Theta(1:tstep,
xlabel('Time (s)')
ylabel('\Phi (rad)')
legend('\omega_1+\omega_2 Detumble','\omega_3 Reduction','\Phi Correction','\Theta Correction','\Psi Correction')
title('Satellite \Phi Control')

%Theta Control Timeline Total Magnitude
subplot(3,1,2)
plot(tspan_con,W1_W2_Detumble(1:tstep,5),tspan_detumble,W3_Detumble(1:tstep,5),tspan_phi,Angle_Correction_Phi(1:tstep,5),tspan_theta,Angle_Correction_Theta(1:tstep,
xlabel('Time (s)')
ylabel('\theta (rad)')
legend('\omega_1+\omega_2 Detumble','\omega_3 Reduction','\Phi Correction','\Theta Correction','\Psi Correction')
title('Satellite \Theta Control')

%Psi Control Timeline Total Magnitude
subplot(3,1,3)
plot(tspan_con,W1_W2_Detumble(1:tstep,6),tspan_detumble,W3_Detumble(1:tstep,6),tspan_phi,Angle_Correction_Phi(1:tstep,6),tspan_theta,Angle_Correction_Theta(1:tstep,
xlabel('Time (s)')
ylabel('\Psi (rad)')
legend('\omega_1+\omega_2 Detumble','\omega_3 Reduction','\Phi Correction','\Theta Correction','\Psi Correction' )
title('Satellite \Psi Control')
```







3D Plot

```
% Define angles over lifetime of satellite phi_life =
vertcat(W1_W2_Detumble(1:tstep,4),W3_Detumble(1:tstep,4),Angle_Correction_Phi(1:tstep,4),Angle_Correction_Theta(1:tstep,4),Angle_Correction_Psi(1:tstep,4)); theta_life =
vertcat(W1_W2_Detumble(1:tstep,5),W3_Detumble(1:tstep,5),Angle_Correction_Phi(1:tstep,5),Angle_Correction_Theta(1:tstep,5),Angle_Correction_Psi(1:tstep,5)); psi_life =
vertcat(W1_W2_Detumble(1:tstep,6),W3_Detumble(1:tstep,6),Angle_Correction_Phi(1:tstep,6),Angle_Correction_Theta(1:tstep,6),Angle_Correction_Psi(1:tstep,6));

% Create the cylinder vertices r = 1; [x_cylinder,y_cylinder,z_cylinder] = cylinder(r,100); z_cylinder = z_cylinder*2;
```

figure; h = surf(x_cylinder, y_cylinder,z_cylinder, 'FaceColor', 'none', 'EdgeColor', 'b');

for i = 1:length(phi_life) % Compute the rotation matrix for the current time step R = euler2rotmat(phi_life(i), theta_life(i), psi_life(i));

```
   if i < tstep
       title('Controller Correction')
   end

   if i > tstep && i < (2*tstep)
        title('Spin Reduction')
   end

   if i > (2*tstep)
       title('Single Axis Rotations')
   end

  % Apply rotation around phi
   rotate(h, [1 0 0], rad2deg(phi_life(i)));

   % Apply rotation around theta
   rotate(h, [0 1 0], rad2deg(theta_life(i)));

   % Apply rotation around psi
   rotate(h, [0 0 1], rad2deg(psi_life(i)));

   pause(waitingperiod);
   M(i) = getframe(gcf);
```

end grid on; axis([-2 2 -2 2 -2 2]); xlabel('X-axis'); ylabel('Y-axis'); zlabel('Z-axis'); title('Satellite Orientation in 3D - Rotating Cylinder'); hold off

% Save video videoWriter = VideoWriter('SatelliteRotationAnimation.mp4', 'MPEG-4'); open(videoWriter); writeVideo(videoWriter, M); close(videoWriter);

Determine Characteristics of Undamped System Initial Conditions

```
% State Space Model Undampened System
Am=[0,.25,.05;-.25,0,-.05;0,0,0];
Bm=[1/100,0,0;0,1/100,0;0,0,1/50];
Cm=[1,1,1];
Dm=[];

%Convert Matrices to Transfer Function
sys=ss(Am,Bm,Cm,Dm);
sys=tf(sys);

%Determine Natural Frequency of Undampened Systen
damp(sys(1));
```

```
          Pole              Damping       Frequency      Time Constant
                                        (rad/seconds)      (seconds)

   0.00e+00 + 2.50e-01i     0.00e+00       2.50e-01           Inf
   0.00e+00 - 2.50e-01i     0.00e+00       2.50e-01           Inf
```

Controller 1

```
%W1 and W2 Reduction Controller
function state_array1 = controller1(t,state)

    %Define Variables in terms of initial conditions
    W1 = state(1);
    W2 = state(2);
    W3 = state(3);
    phi = state(4);
    theta = state(5);
    psi = state(6);

    %define Moments of Inertia
    I11 = 100;
    I22 = 100;
    I33 = 50;

    %Define Desired Pole Locations for Controller
    zeta_wn = 4/25;
    wn=.25;
    zeta = zeta_wn/wn;
    pd_neg = -zeta_wn - wn*sqrt(1-zeta^2);
    p = [pd_neg, pd_neg,0];

    %State Space Model Using Euler's Moment
    A =[0,(I22-I33)*W3/(2*I11),(I22-I33)*W2/(2*I11);(I33-I11)*W3/(2*I22),0,(I33-I11)*W1/(2*I22);(I11-I22)*W2/(2*I33),(I11-I22)*W1/(2*I33),0];
    B = [1/100,0,0;0,1/100,0;0,0,1/50];
    x = [W1 W2 W3]';

    %Determine Gain Value Necesarry for Control
```

```
    K = place(A,B,p);
    %Define U Matrix with Determined Gains
    u = [-K(1,1)*W1 ; -K(2,2)*W2;0*W3];

    %Define Angle Relationships
    phi_dot=(cos(psi)*W2+sin(psi)*W1)/sin(theta);
    theta_dot=(cos(psi)*W1-sin(psi)*W2);
    psi_dot=W3-((cos(psi)*W2+sin(psi)*W1)*cos(theta))/sin(theta);

    %Define Ouptut of State Space Function
    xdot = A*x + B*u;
    %Defining Output for ODE 45 in terms of W1 W2 W3 Phi Theta Psi
    state_array1 = [xdot; phi_dot;theta_dot;psi_dot];

end
```

Controller 2

```
%Define Function for W3 Control
function state_array2 = detumble(t,state) % Drives Transverse Angular Velocities to 0

    %Define Variables Based on Initial Conditions
    W1 = state(1);
    W2 = state(2);
    W3 = state(3);
    phi = state(4);
    theta = state(5);
    psi = state(6);

    %Define Moments of Inertia
    I11 = 100;
    I22 = 100;
    I33 = 50;

    %Define Desired Pole Locations for Controller
    zeta_wn = 4/25;
    wn=.25;
    zeta = zeta_wn/wn;
    pd = -zeta_wn - wn*sqrt(1-zeta^2);
    p = [0,0,pd];

    %State Space Model Using Euler's Moment Equation
    A =[0,(I22-I33)*W3/(2*I11),(I22-I33)*W2/(2*I11);(I33-I11)*W3/(2*I22),0,(I33-I11)*W1/(2*I22);(I11-I22)*W2/(2*I33),(I11-I22)*W1/(2*I33),0];
    B = [1/100,0,0;0,1/100,0;0,0,1/50];
    x = [W1 W2 W3]';

    %Determine Gain Value Necesarry for Control
    K = place(A,B,p);
    %Define U Matrix with Determined Gains
    u = [-0*W1 ; -0*W2;-K(3,3)*W3];

    %Define Angle relationships
    phi_dot=(cos(psi)*W2+sin(psi)*W1)/sin(theta);
    theta_dot=(cos(psi)*W1-sin(psi)*W2);
    psi_dot=W3-((cos(psi)*W2+sin(psi)*W1)*cos(theta))/sin(theta);

    %Define Ouptut of State Space Function
    xdot = A*x + B*u;
    %Defining Output for ODE 45 in terms of W1 W2 W3 Phi Theta Psi
    state_array2 = [xdot;phi_dot;theta_dot;psi_dot];
end
```

Controller 3

```
%Define Function for Phi Correction
function state_array_phi=angle_phi(t,state)

    %Define Variables Based on Initial Conditions
    W1 = state(1);
    W2 = state(2);
    W3 = state(3);
    phi = state(4);
    theta = state(5);
    psi = state(6);

    %Define Moments of Inertia
    I11=100;
    I22=100;
    I33=50;

    %Define Desired Angles
    desired_phi = 0;

    %Define Angle Differences and Gain Value
    K = .8;
    phi_error = desired_phi-phi;

    %Define State Space model using Euler's Equation
    A =[0,(I22-I33)*W3/(2*I11),(I22-I33)*W2/(2*I11);(I33-I11)*W3/(2*I22),0,(I33-I11)*W1/(2*I22);(I11-I22)*W2/(2*I33),(I11-I22)*W1/(2*I33),0];
```

```
    B = [1/100,0,0;0,1/100,0;0,0,1/50];
    u=[-K*W3;-K*W2;-K*W1];
    x = [W1 W2 W3]';

    %Define Output of State Space Function
    xdot=A*x+B*u;

    %Define Angle Relationships
    phi_dot=(cos(psi)*W2+sin(psi)*W1)/sin(theta);
    theta_dot=(cos(psi)*W1-sin(psi)*W2);
    psi_dot=W3-((cos(psi+sin(psi)*W1)*cos(theta))/sin(theta))*W2;

    %Define Angle Change Between Steps
    phi_dot = psi_dot+phi_error;

    %Defining Output for ODE 45 in terms of W1 W2 W3 Phi Theta Psi
    state_array_phi = [xdot;phi_dot;theta_dot;psi_dot];

end

%Define Function for Theta Correction
function state_array_theta=angle_theta(t,state)

    %Define Variables Based on Initial Conditions
    W1 = state(1);
    W2 = state(2);
    W3 = state(3);
    phi = state(4);
    theta = state(5);
    psi = state(6);

    %Define Moments of Inertia
    I11=100;
    I22=100;
    I33=50;

    %Define Desired Angles
    desired_theta = pi/2;

    %Define Angle Differences and Gain Value
    K = .8;
    theta_error = desired_theta-theta;

    %Define State Space model using Euler's Equation
    A =[0,(I22-I33)*W3/(2*I11),(I22-I33)*W2/(2*I11);(I33-I11)*W3/(2*I22),0,(I33-I11)*W1/(2*I22);(I11-I22)*W2/(2*I33),(I11-I22)*W1/(2*I33),0];
    B = [1/100,0,0;0,1/100,0;0,0,1/50];
    u=[-K*W1;-K*W2;-K*W3];
    x = [W1 W2 W3]';

    %Define Output of State Space Function
    xdot=A*x+B*u;

    %Define Angular Acceleration Values
    phi_dot=(cos(psi)*W2+sin(psi)*W1)/sin(theta);
    theta_dot=(cos(psi)*W1-sin(psi)*W2);
    psi_dot=W3-((cos(psi)*W2+sin(psi)*W1)*cos(theta))/sin(theta);

    %Define Angle Change Between Steps
    theta_dot = phi_dot+theta_error;

    %Defining Output for ODE 45 in terms of W1 W2 W3 Phi Theta Psi
    state_array_theta = [xdot;phi_dot;theta_dot;psi_dot];

end

%Define Function for Psi Correction
function state_array_psi=angle_psi(t,state)

    %Define Variables Based on Initial Conditions
    W1 = state(1);
    W2 = state(2);
    W3 = state(3);
    phi = state(4);
    theta = state(5);
    psi = state(6);

    %Define Moments of Inertia
    I11=100;
    I22=100;
    I33=50;

    %Define Desired Angles
    desired_psi=psi-rem(psi/(2*pi),1);

    %Define Angle Differences and Gain Value
    K = 10;
    psi_error =desired_psi-psi;

    %Define State Space model using Euler's Equation
    A =[0,(I22-I33)*W3/(2*I11),(I22-I33)*W2/(2*I11);(I33-I11)*W3/(2*I22),0,(I33-I11)*W1/(2*I22);(I11-I22)*W2/(2*I33),(I11-I22)*W1/(2*I33),0];
    x = [W1 W2 W3]';
```

```
    B = [1/100,0,0;0,1/100,0;0,0,1/50];
    u=[-K*W1;-K*W2;-K*W3];

    %Define Output of State Space Function
    xdot=A*x+B*u;

    %Define Angular Acceleration Values
    phi_dot=(cos(psi)*W2+sin(psi)*W1)/sin(theta);
    theta_dot=(cos(psi)*W1-sin(psi)*W2);
    psi_dot=W3-((cos(psi)*W2+sin(psi)*W1)*cos(theta))/sin(theta);

    %Define Angle Change Between Steps
    psi_dot = psi_dot+psi_error;

    %Defining Output for ODE 45 in terms of W1 W2 W3 Phi Theta Psi
    state_array_psi = [xdot;phi_dot;theta_dot;psi_dot];

end

% function R = euler2rotmat(phi, theta, psi)
%     R_phi = [1 0 0; 0 cos(phi) -sin(phi); 0 sin(phi) cos(phi)];
%     R_theta = [cos(theta) 0 sin(theta); 0 1 0; -sin(theta) 0 cos(theta)];
%     R_psi = [cos(psi) -sin(psi) 0; sin(psi) cos(psi) 0; 0 0 1];
%     R = R_phi * R_theta * R_psi;
% end
```

Published with MATLAB® R2020b