Report 6: Dynamic Simulation of Rotation

In this report, we did experiment to simulate the dynamics rotation of cylinder. To do that, first approximate cylinder as polygon with a large number of sides. As we increase the number of the sides of a polygon, it finally converges to the shape of a cylinder. Choosing dodecagon (a polygon with 12 number of side) for experiment. For the comparison purpose and to make myself familiar with the topic I tried to simulate octagon then next simulate the dodecagon.

Finally, the experiment showed that as we increase the number of sides in the polygon to infinity, the polygon will converge to cylinder.

1. Rigid Body cylinder class definition

```
classdef RigidBody_Cylinder < RigidBody
    properties
        a, b, c;
    end
    methods
        function obj = RigidBody_Cylinder (rho, a, b, c)  % a and b are equal to the radius
of the cylider
            obj@RigidBody(1,eye(3));
            m = rho*pi*(a^2);   % volume of the cylinder

            Jx = (1/4)*m*(a^2) + (1/12)*m*(c^2);
            Jy = (1/4)*m*(a^2) + (1/12)*m*(c^2);     % the moment of inertia of a cylinder
along x,y,z axis
            Jz = (1/2)*m*(a^2);
            J = diag([Jx, Jy, Jz]);

            obj = obj.mass_and_inertia_matrix(m, J);
            obj.density = rho;
            obj.a = a;
            obj.b = b;
            obj.c = c;
        end

        function draw(obj, pos, q)
            arguments
                obj;
                pos = zeros(3,1);
                q = [1; 0; 0; 0];
            end


            persistent t01 t02 t03 t04 t05 t06  t07 t08 t09 t10 t11 t12
            persistent b01 b02 b03 b04 b05 b06  b07 b08 b09 b10 b11 b12
            if isempty(t01)
```

```
        a1 = obj.a; b1 = obj.b; c1 = obj.c/2;
        a2 = obj.a * sind(60); b2 = obj.b * cosd(60);
        a3 = obj.a * sind(30); b3 = obj.b * cosd(30);


        t01 = [ 0; b1; c1 ];
        t02 = [ a3; b3;  c1 ];
        t03 = [ a2;  b2;  c1 ];
        t04 = [ a1;  0; c1 ];
        t05 = [ a2; -b2; c1 ];
        t06 = [ a3; -b3;  c1 ];
        t07 = [ 0; -b1;  c1 ];
        t08 = [ -a3; -b3; c1 ]; %% ambigious
        t09 = [ -a2; -b2; c1 ];
        t10 = [ -a1; 0; c1 ];
        t11 = [ -a2; b2; c1 ];
        t12 = [ -a3; b3; c1 ];

        b01 = [ 0; b1; -c1 ];
        b02 = [ a3; b3; -c1 ];
        b03 = [ a2;  b2; -c1 ];
        b04 = [ a1;  0; -c1 ];
        b05 = [ a2; -b2; -c1 ];
        b06 = [ a3; -b3;  -c1 ];
        b07 = [ 0; -b1; -c1 ];
        b08 = [ -a3; -b3; -c1 ]; %% ambigious
        b09 = [ -a2; -b2; -c1 ];
        b10 = [ -a1; 0; -c1 ];
        b11 = [ -a2; b2; -c1 ];
        b12 = [ -a3; b3; -c1 ];



    end
    obj = obj.quaternion(q);
    rotation_matrix = obj.rotation_matrix;
    y01 = rotation_matrix*t01 + pos;
    y02 = rotation_matrix*t02 + pos;
    y03 = rotation_matrix*t03 + pos;
    y04 = rotation_matrix*t04 + pos;
    y05 = rotation_matrix*t05 + pos;
    y06 = rotation_matrix*t06 + pos;
    y07 = rotation_matrix*t07 + pos;
    y08 = rotation_matrix*t08 + pos;
    y09 = rotation_matrix*t09 + pos;
    y10 = rotation_matrix*t10 + pos;
    y11 = rotation_matrix*t11 + pos;
    y12 = rotation_matrix*t12 + pos;

    i01 = rotation_matrix*b01 + pos;
```

```
            i02 = rotation_matrix*b02 + pos;
            i03 = rotation_matrix*b03 + pos;
            i04 = rotation_matrix*b04 + pos;
            i05 = rotation_matrix*b05 + pos;
            i06 = rotation_matrix*b06 + pos;
            i07 = rotation_matrix*b07 + pos;
            i08 = rotation_matrix*b08 + pos;
            i09 = rotation_matrix*b09 + pos;
            i10 = rotation_matrix*b10 + pos;
            i11 = rotation_matrix*b11 + pos;
            i12 = rotation_matrix*b12 + pos;
            hold on;
            y = [ y01, y02, i02, i01, y01 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y02, y03, i03, i02, y02 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y03, y04, i04, i03, y03 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y04, y05, i05, i04, y04 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y05, y06, i06, i05, y05 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y06, y07, i07, i06, y06 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y07, y08, i08, i07, y07 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y08, y09, i09, i08, y08 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y09, y10, i10, i09, y09 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y10, y11, i11, i10, y10 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y11, y12, i12, i11, y11 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);
            y = [ y12, y01, i01, i12, y12 ]; fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8,
0.8]);

            y = [ y01, y02, y03, y04, y05, y06, y07, y08, y09, y10, y11, y12 ];
fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8, 0.8]);
            y = [ i01, i02, i03, i04, i05, i06, i07, i08, i09, i10, i11, i12 ];
fill3(y(1,:), y(2,:), y(3,:), [0.8, 0.8, 0.8]);

            hold off;
        end
    end
end
```

2. ODE solver

```
body = RigidBody_Cylinder(1, 4, 4, 8);
alpha = 1000; % positive large constant for CSM

ext = @(t) external_torque(t);
rotation_quaternion_ODE = @(t,s) rotation_quaternion_ODE_params (t,s, body, alpha, ext);
%interval = 0:0.0001:20;
tf = 20;
interval = [0,tf];
sinit = [1;0;0;0; 0;0;0;0];
[time, s] = ode45(rotation_quaternion_ODE, interval, sinit);

%making_video (body, time, s, 0.1);
making_tiled_video (body, time, s, 0.1);

function dots = rotation_quaternion_ODE_params (t,s, body, alpha, ext)
    q = s(1:4); dotq = s(5:8);
    ddotq = body.calculate_ddotq (q, dotq, alpha, ext(t));
    dots = [dotq;ddotq];
end

function tau = external_torque(t)
    if t <= 5
        tau = [12.00; 0.00; 0.00];
    elseif t <= 10
        tau = [ 0.00; -12.00; 0.00];
    else
        tau = [0;0;0];
    end
end

function making_video (body, time, s, videostep)

    figure('position', [0, 0, 1200, 1200]);
    set(0,'defaultAxesFontSize',12);
    set(0,'defaultTextFontSize',12);

    clf('reset');
    fr = 1;
    clear M;
    ts = time(1);
    tf = time(end);
    for t=ts:videostep:tf
        fprintf("time %f\n", t);
        index = nearest_index(time, t);
        q = s(index, 1:4);
        clf;
        body.draw(zeros(3,1), q);
```

```
        xlim([-10,10]); ylim([-10,10]); zlim([-10,10]);
        xlabel('x'); ylabel('y'); zlabel('z');
        pbaspect([1 1 1]); grid on; view([-75, 30]);
        drawnow;
        M(fr) = getframe(gcf);
        fr = fr + 1;
    end
    M(fr) = getframe(gcf);

    v = VideoWriter('rotation_quaternion', 'MPEG-4');
    open(v);
    writeVideo(v, M);
    close(v);
end

function making_tiled_video (body, time, s, videostep)

    figure('position', [0, 0, 1600, 1600]);
    set(0,'defaultAxesFontSize',12);
    set(0,'defaultTextFontSize',12);

    clf('reset');
    fr = 1;
    clear M;
    ts = time(1);
    tf = time(end);
    for t=ts:videostep:tf
        fprintf("time %f\n", t);
        index = nearest_index(time, t);
        q = s(index, 1:4);
        clf;
        tiledlayout(2,2);
        %
        nexttile;
        body.draw(zeros(3,1), q);
        xlim([-10,10]); ylim([-10,10]); zlim([-10,10]);
        xlabel('x'); ylabel('y'); zlabel('z');
        pbaspect([1 1 1]); grid on; view([-75, 30]);
        %
        nexttile;
        body.draw(zeros(3,1), q);
        xlim([-10,10]); ylim([-10,10]); zlim([-10,10]);
        xlabel('x'); ylabel('y'); zlabel('z');
        pbaspect([1 1 1]); grid on; view([0, 0]);   % x-z
        %
        nexttile;
        body.draw(zeros(3,1), q);
        xlim([-10,10]); ylim([-10,10]); zlim([-10,10]);
        xlabel('x'); ylabel('y'); zlabel('z');
        pbaspect([1 1 1]); grid on; view([-90, 0]);   % y-z
```

```
        %
        nexttile;
        body.draw(zeros(3,1), q);
        xlim([-10,10]); ylim([-10,10]); zlim([-10,10]);
        xlabel('x'); ylabel('y'); zlabel('z');
        pbaspect([1 1 1]); grid on; view([0, 90]);   % x-y

        drawnow;
        M(fr) = getframe(gcf);
        fr = fr + 1;
    end
    M(fr) = getframe(gcf);

    v = VideoWriter('rotation_quaternion_tiled', 'MPEG-4');
    open(v);
    writeVideo(v, M);
    close(v);
end
```

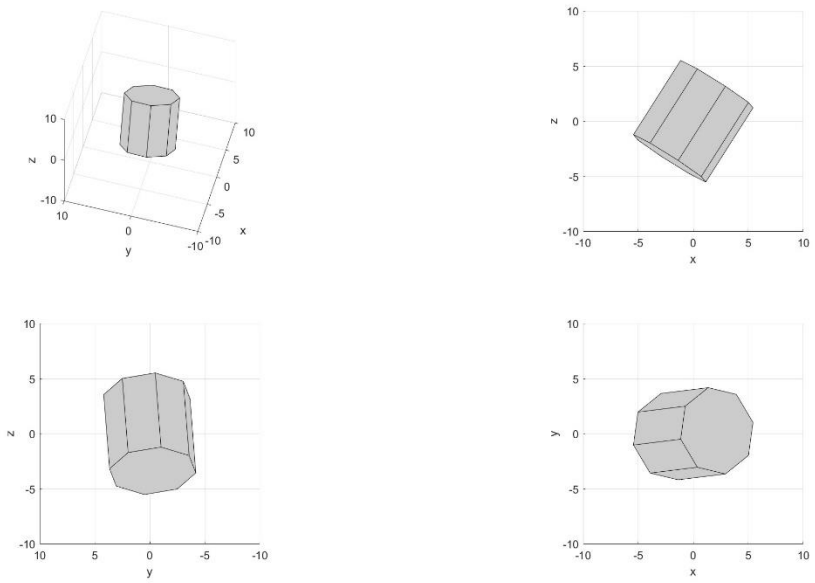

Fig1. Approximating cylinder using octagon (8-sided polygon)

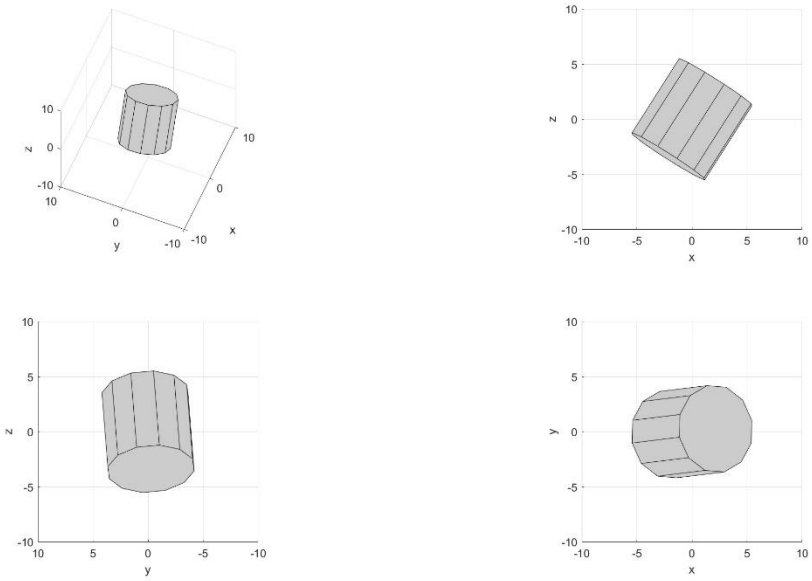

Fig2. Approximating cylinder using dodecagon (12-sided polygon)

Published with MATLAB® R2022b