```
classdef Cylinder
   properties
      mass                          %mass
      height                        %height
      radius                        %radius
      inertial_tensor               %[3x3] mass properties matrix
      position                      %[x,y,z] point to track
      orientation                   %[phi;psi;theta]
      spin_rates                    %[phi_dot;psi_dot;theta_dot]
      thrusters                     %[x y z Fcompbx Fcompby Fcompbz]
   end
   methods
      function I = CalculateInertialTensor(obj)
          if isempty(obj.mass) == 1 || isempty(obj.height) == 1 || isempty(obj.radius) == 1
              disp('Cannot define inertial tensor because mass and/or dimensions are not defi
ned.')
          elseif isempty(obj.inertial_tensor) == 1
              I = [(obj.mass/12)*(3*obj.radius^2+obj.height^2) 0 0;
              0 (obj.mass/12)*(3*obj.radius^2+obj.height^2) 0;
              0 0 (obj.mass*obj.radius^2)/2];
          else
              disp('Inertial tensor already defined.')
          end
      end
      function [orientation, position] = freerotate(obj,dt,Nframe)
          obj.orientation(1) = obj.orientation(1) + obj.spin_rates(1)*dt;   % angles grow, th
ey do not reset after 360
          obj.orientation(2) = obj.orientation(2) + obj.spin_rates(2)*dt;   % angles grow, th
ey do not reset after 360
          obj.orientation(3) = obj.orientation(3) + obj.spin_rates(3)*dt;   % angles grow, th
ey do not reset after 360
          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          %%%% now we have angles for time stamp we are in %%%%
          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

          % rotate
          R_na = [cosd(obj.orientation(2)) -sind(obj.orientation(2)) 0;        %Precession ro
tation matrix
              sind(obj.orientation(2)) cosd(obj.orientation(2)) 0;
              0 0 1];
          R_ag = [1 0 0;                                                       %Newtation rot
ation matrix
                  0 cosd(obj.orientation(3)) -sind(obj.orientation(3));
                  0 sind(obj.orientation(3)) cosd(obj.orientation(3))];
          R_gb = [cosd(obj.orientation(1)) -sind(obj.orientation(1)) 0;        %Spin rotation
 matrix
                  sind(obj.orientation(1)) cosd(obj.orientation(1)) 0;
                  0 0 1];
           R_nb = R_na*R_ag*R_gb;                %Matrix from N-->B
           Bframe = Nframe*R_nb;                 %Define B frame
          % w = psi_dot*N_x + phi_dot*B_x;
          % I = [(m/12)*(3*r^2+h^2) 0 0;                 %Check if mass properties matrix is
constant
          % 0 (m/12)*(3*r^2+h^2) 0;
          % 0 0 (m*r^2)/2];                             %constant **
```

```
            position = Bframe*obj.position;  %redefine position of point being tracked
            orientation = [obj.orientation(1);obj.orientation(2);obj.orientation(3)];
        end
        function [orientation, position] = appliedforce(obj,Fdir,Fpos,dt,Nframe)




        end
    end
end
```

```
ans =

  Cylinder with properties:

              mass: []
            height: []
            radius: []
    inertial_tensor: []
          position: []
       orientation: []
        spin_rates: []
          thrusters: []
```

Published with MATLAB® R2016a