```
function [theta_in, path, distance, theta_vec, vel_out, g_s] =
 loop(vel_in, pos_in, n, prop, r, h0, theta_in_ref, path_ref,
 theta_vec_ref)
    theta_in = loop_starting_angle(vel_in);

    if isnan(theta_in_ref)
        [path, distance, theta_vec] = loop_compute_path(pos_in,
 theta_in, vel_in, n, prop, r);
    else
        [path, distance, theta_vec] = loop_compute_path(pos_in,
 theta_in_ref, vel_in, n, prop, r);
    end

    if isnan(path_ref)
        vel_out = loop_exit_vel(path, theta_vec, h0, vel_in);
    else
        vel_out = loop_exit_vel(path_ref, theta_vec_ref, h0, vel_in);
    end

    if isnan(theta_vec_ref)
        g_s = loop_compute_g_s(path, vel_in, h0, theta_vec, r);
    else
        g_s = loop_compute_g_s(path_ref, vel_in, h0, theta_vec_ref,
 r);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% COMPLETE THE FUNCTIONS BELOW %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function theta_in = loop_starting_angle(vel_in)
   % First, compute the necessary starting angle for the loop -
 measured from negative z-axis(see sketches in problem description)

   vx = vel_in(1);
   vy = vel_in(2);
   vz = vel_in(3);

   if sign(vel_in(2)) > 0 % Positive initial x-velocity means the loop
 is counterclockwise
       % TO DO: complete this section of code
       theta_in = atan2(vz,vy); % radians
   else % Otherwise (negative initial x-velocity), the loop is
 clockwise
       % TO DO: complete this section of code
       theta_in = atan2(vz,-vy); % radians
   end
   theta_in;
end
```

```
function [path, distance, theta_vec] = loop_compute_path(pos_in,
 theta_in, vel_in, n, prop, r)
    % evenly spaced vector from initial theta, to final theta
    theta_vec = linspace(theta_in, prop*(theta_in + 2*pi), n);

    % x,y,z position along element path, evaluated at each theta from
 theta_vec
   if sign(vel_in(2)) > 0 % Positive initial x-velocity means the loop
 is counterclockwise
       % TO DO: complete this section of code
       path = pos_in + [3*(theta_vec'-theta_in), -r*sin(theta_in)
 + r*cos(theta_vec' - pi/2), r*cos(theta_in) + r*sin(theta_vec' -
 pi/2)]; %[x_pos, y_pos, z_pos]
   else % Otherwise (negative initial x-velocity), the loop is
 clockwise
       % TO DO: complete this section of code
       path = pos_in + [3*(theta_vec'-theta_in), r*sin(theta_in)
 - r*cos(theta_vec' - pi/2), r*cos(theta_in) + r*sin(theta_vec' -
 pi/2)]; %[x_pos, y_pos, z_pos]
   end

   %pos_in
   %path;

   % distance traveled along element path, evaluated at each theta
 from theta_vec
   distance = cumtrapz(theta_vec, sqrt(3^2 + (r*sin(theta_vec'-
pi/2)).^2 + (r*cos(theta_vec'-pi/2)).^2));

end



function vel_out = loop_exit_vel(path, theta_vec, h0, vel_in)
    % compute the velocity vectore leaving the loop element
    g = 9.81;

    h = path(end, 3);
    vx = vel_in(1);
    vy = vel_in(2);
    vz = vel_in(3);

    mag = sqrt(2*g*(h0-h));

    vel_out = [3, (mag-sqrt(3))*cos(2*pi - theta_vec(end)), (mag-
sqrt(3))*sin(2*pi - theta_vec(end))]; %[x_vel, y_vel, z_vel]
end



function g_s = loop_compute_g_s(path, vel_in, h0, theta_vec, r)
    % Compile the g forces into matrix, evaluated at each theta in
 theta_vec
```

```
    g = 9.81;

    mag = sqrt(2*g*(h0-path(:,3)));
    vx = vel_in(1);
    vy = vel_in(2);
    vz = vel_in(3);

    vel = [mag.*cos(theta_vec'), vy*ones(length(theta_vec),1),
 mag.*sin(theta_vec')];

    g_s = [zeros(length(theta_vec'),1), zeros(length(theta_vec),1),
 (mag.^2)/(r*g) - sin(theta_vec'- pi/2)];   %G-force matrix [front/
back, left/right, up/down]
end

Not enough input arguments.

Error in loop (line 2)
    theta_in = loop_starting_angle(vel_in);
```

Published with MATLAB® R2021a