```
function [path, distance, vel_out, g_s] = parabola(vel_in, pos_in, n,
 a, d, h0, path_ref, theta_vec_ref)

    [path, distance] = parabola_compute_path(pos_in, vel_in, n, a, d);

    if isnan(path_ref)
        % Running sequence as normal
        vel_out = parabola_exit_vel(path, h0, vel_in, a);
    else
        % Using dependent input as reference... used to bypass
 unimplemented functions
        vel_out = parabola_exit_vel(path_ref, h0, vel_in, a);
    end

    if isnan(path_ref)
        % Running sequence as normal
        g_s = parabola_compute_g_s(path, vel_out, h0, a);
    else
        % Using dependent input as reference... used to bypass
 unimplemented functions
        g_s = parabola_compute_g_s(path_ref, vel_out, h0, a);
    end
end


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


function [path, distance] = parabola_compute_path(pos_in, vel_in, n,
 a, d)
    % create a vector of n equally spaced x-values ranging from x_in
 to x_in+d. Evaluate parabola at each one of these x locations
    % to create path matrix
    x_in = pos_in(1);
    y_in = pos_in(2);
    z_in = pos_in(3);

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

    if(sign(vy)>0)
        y = linspace(y_in, y_in + d, n);
    else
        y = linspace(y_in, y_in - d, n);
    end

    t = (y-y_in)/vy;

    path = [x_in*ones(length(y),1), y', (a/2)*(t').^2 + vz*t' + z_in];
```

```
    % you do not need to find analytical form to compute distance at
 each point. Hint: think distance formula
    distance = cumtrapz(t', sqrt(0 + vy^2 + (a*t' + vz).^2));
end

function vel_out = parabola_exit_vel(path, h0, vel_in, a)
    % compute velocity vector leaving parabola
    vx = vel_in(1);
    vy = vel_in(2);
    vz = vel_in(3);

    mag = sqrt(2*-a*(h0-path(end,3)));

    if sign(vy) > 0;
        vz_out = sqrt(mag^2 - vy^2 - vx^2);
    else
        vz_out = -sqrt(mag^2 - vy^2 - vx^2);
    end

    vel_out = [vx, vy, vz_out];
end

function g_s = parabola_compute_g_s(path, vel_out, h0, a)

    % number of points to comput G's at
    n = length(path(:,1));

    % Compile the gs and xyz coordinates into the matrices to be
 outputted. Hint: you can leverage what you know about a ballistic
 trajectory to
    % create g_s
    g_s = [ones(n,1)*(0), ones(n,1)*(0/-a), ones(n,1)*(0/a)];  %Gs
 matrix [front/back, left/right, up/down]
end
```

```
Not enough input arguments.

Error in parabola (line 3)
    [path, distance] = parabola_compute_path(pos_in, vel_in, n, a, d);
```

Published with MATLAB® R2021a