```
function [tout, pos, vel] = simulate_rocket_improved(init_pos,
 init_vel, moon_pos, t)
% Author: Lyubomir Shoylev, lyubomir-shoylev@st-hildas.ox.ac.uk ,
 Date: 15/12/2019
%
% NB: The calculation method is an improved Euler method presented in
 the
% lab script.
%
% Simulate the rocket trajectory with the earth and moon influence.
 The coordinate
% used in this function is centred at earth's centre (i.e. earth
 centre at (0,0) )
% and scaled in moon?radius.
% The simulation finishes when it simulates for the whole t, or the
 rocket landed
% on the moon.
% Input:
% * init_pos: 2-elements vector (x, y) indicating the initial position
 of the rocket.
% * init_vel: 2-elements vector (vx, vy) of the initial velocity of
 the rocket.
% * moon_pos: a function that receives time, t, and return a 2-
elements vector (x, y)
%                (see hint) indicating the moon position relative to
 earth.
% * t: an N-    elements vector of the time step where the position of
 the rocket will be
%                returned.
%
% Output:
% * tout: an M-elements vector of the time step where the position is
 described,
%            if the rocket does not land on the moon, M = N.
% * pos: (M x 2) matrix indicating the positions of the rocket as
 function of time,
%            with the first column is x and the second column is y.
% * vel: (M x 2) matrix indicating the velocities of the rocket as
 function of time,
%            with the first column is x and the second column is y.
%
% Example use:
% >> init_pos = [0, 3.7];
% >> init_vel = 0.0066 * [cosd(89.9), sind(89.9)];
% >> moon_pos = @(t) [0, 222];
% >> t = linspace(0, 10000, 1000);
% >> [tout, pos] = simulate_rocket(init_pos, init_vel, moon_pos, t);
% >> plot(pos(:,1),pos(:,2));

    % Constants:
    M_m = 1.0;      % mass of the Moon in Moon masses
    M_e = 83.3;     % mass of the Earth in Moon masses
```

```
    R_m = 1.0;       % radius of the Moon in Moon radii
    R_e = 3.7;       % radius of the Moon in Moon radii
    G = 9.63e-7;     % gravitational constant in new lenght and mass
 units

    % Helper functions
    % Function giving magnitude of vector v
    mag_vec = @(v) sqrt(v(1)^2 + v(2)^2);
    % Acceleration in a given direction, given by:
    %   -p1: 2-elements (x, y) position vector of projectile realtive
 to Earth
    %   -p2: 2-elements (x, y) position vector of projectile realtive
 to Moon
    %   -dir: integer showing the desired component of accel to be
 calculated
    %         1==x, 2==y
    a_dir = @(p1, p2, dir)...
            -1*(G*M_e)*p1(dir)/(mag_vec(p1))^3 - (G*M_m)*p2(dir)/
(mag_vec(p2))^3;

    % Initialize the output variables
    tout = [t(1)];
    pos = [init_pos];
    vel = [init_vel];
    % Initialize the small time interval
    delta_t = t(2);
    % The initial pos and velocity in the improved method
    pos0 = init_pos;
    vel0 = init_vel;

    for n=2:numel(t)
        % the current moon position
        m_pos = moon_pos(t(n-1));
        % the following values will reference those in lab script CO06

        % acceleration0 from unprimed coords
        a_x0 = a_dir(pos0, pos0 - m_pos, 1);
        a_y0 = a_dir(pos0, pos0 - m_pos, 2);

        % initial primed coords - calculated as in the usual Euler
 method
        pos1 = pos0 + delta_t*vel0;
        % acceleration1 from the initial primed coords
        a_x1 = a_dir(pos1, pos1 - m_pos, 1);
        a_y1 = a_dir(pos1, pos1 - m_pos, 2);

        % primed velocities - based on the improved method
        vel1 = vel0 + 0.5*delta_t*[a_x0 + a_x1, a_y0 + a_y1];
        % final primed coords - based on the improved method
        pos1 = pos0 + 0.5*delta_t*(vel0 + vel1);

        % record the new values for position and velocity
        tout = [tout, t(n)];
        pos = [pos; pos1];
```

```
        vel = [vel; vel1];

        % transfer values for next loop iteration
        pos0 = pos1;
        vel0 = vel1;

        % Terminating condition
        if (mag_vec(pos0) <= R_e) || (mag_vec(pos0 - m_pos) <= R_m)
            break;
        end

    end

end
```

Published with MATLAB® R2019b