CO61: Orbital simulations of spacecraft trajectories in the Earth-Moon system through the use of Heun's method

Gabriel Bristot, Kebl8311. 17/02/2025

1 Aim and method

The aim of this practical is to be able to code a Matlab simulation of the gravitational effects of the Earth-Moon system in a spacecraft's path and plot the specific orbits that intercept the Moon when launched from the Earth's surface.

In this simplified model, we only account for the gravitational effects of the Earth and the Moon and disregard any drag effects of Earth's atmosphere or the changing mass of the rocket as it burns its fuel, we also consider two cases: when the moon is stationary and when the moon is in circular orbit around the Earth. So the only variables that we consider are the initial position and velocity of the rocket.

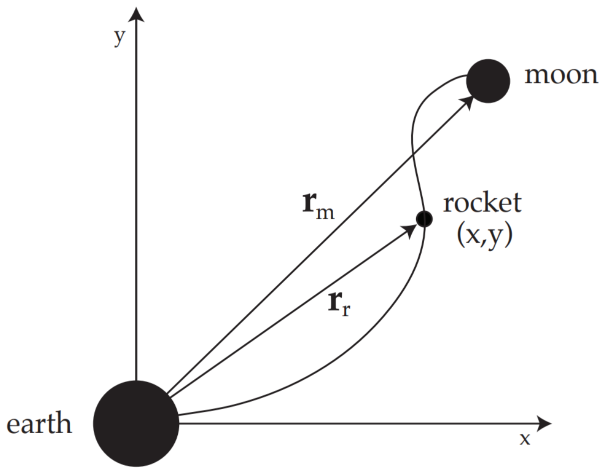

Figure 1: Trajectory of spacecraft. Figure from [1]

The method used to calculate the acceleration of the rocket $\mathbf{a}_r$ was:

$$\mathbf{a}_r = -GM_e\frac{\mathbf{r}_r}{|\mathbf{r}_r|^3} - GM_m\frac{\mathbf{r}_r - \mathbf{r}_m}{|\mathbf{r}_r - \mathbf{r}_m|^3} \tag{1}$$

Where M_e and M_m are the mass of Earth and the Moon. To solve this differential equation numerically, we used Heun's method (also called improved Euler's method) which is described below:

$$x_{n+1} = x_n + \frac{1}{2}\Delta t\left[f(x_n, t_n) + f(x_n + \Delta t f(x_n, t_n), t_{n+1})\right] \tag{2}$$

Where $\frac{dx}{dt} = f(x,t)$ and Δt is the time step. This is a second order method since the errors are proportional to the square of the step size. So first a "guess" of $x_{n+1} = x_n + \Delta t f(x_n, t_n)$ is computed and then the velocity is found using equation 2. Finally, the guess of x_{n+1} is refined with equation 2. More detail of this derivation can be found at [1].

Since the common SI unit of length and mass become cumbersome to work with due to the large numbers involved, the simulation uses Moon-radii as the unit of length and Moon-mass as the unit of mass but the second remains the unit of time. Then the simulation is run for a predetermined amount of time or until the distance $|\mathbf{r}_r - \mathbf{r}_m|$ is less than 1 Moon-radius. And a N×2 matrix of the positions is given as the output together with the N×1 time vector.

2 Results

2.1 Stationary Moon

We assume that the Moon was stationary on the y-axis at a distance of 222 Moon-radii (≈ 386000 km) from the Earth. Then calculated the trajectory 2(a) where the spacecraft was launched from the y-axis at 3.7 Moon-radii (≈ 6430 km) and at an angle of 89.9°, with a speed of 0.0066 Moon-radii/s (≈ 11.5 km/s). Then, by trial and error, trajectory 2(b) was found where the spacecraft was launched from the x-axis at 3.7 Moon-radii at an angle of 51.5° with the same speed. Both trajectories intersect the Moon in Figure 2.

The time step chosen was $\Delta t = 10s$ and the total time of the simulation was $T_{(a)} = 156990s$ and $T_{(b)} = 158990s$ which are both in the order of $T \approx 43.6$ hours. Further detail of the possible solutions is given in Appendix B where I simulated the minimum distance to the centre of the Moon given different values of θ and v_0 and plotted the surface generated (Figure 4 and Figure 5)

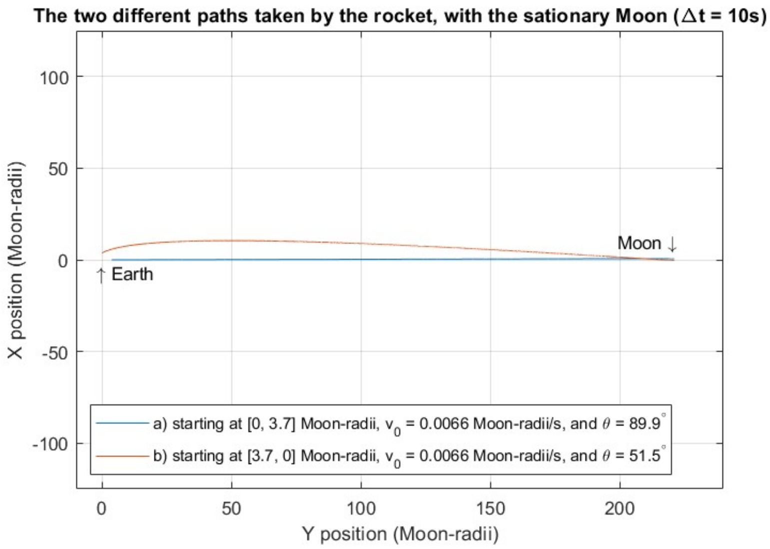

Figure 2: Trajectories of spacecraft with a stationary Moon

2.2 Orbiting Moon

Now we move from the simple case of a stationary Moon to a more realistic perfectly circularly orbiting Moon with an angular frequency of $\Omega = 2.6615 \times 10^{-6}$ rad/s and radius $R_0 = 222$ Moon-radii (Period of ≈ 27.3 days). The position vector of the Moon can be written down as $\mathbf{r}_m = R_0[\cos(\Omega t), \sin(\Omega t)]$. A solution that intercepted the Moon was found using trial and error and is shown in Figure 3. More detail of possible solutions is given in Appendix B.

3 Conclusion

So, we correctly simulated the trajectories of spacecraft in both a simplified stationary and a more realistic orbiting Earth-Moon system. We were able to find specific parameters that lead to a Moon intercept. Further, the simulations had a total time of ≈43.6 hours, matching real observed times in space missions. And further analysis given in Appendix B also indicates for all simulations, a hard "cut-off" to reach the moon is ≈0.0065 Moon-Radii/s (≈11.3 km/s), which matches the escape velocity of the Earth.

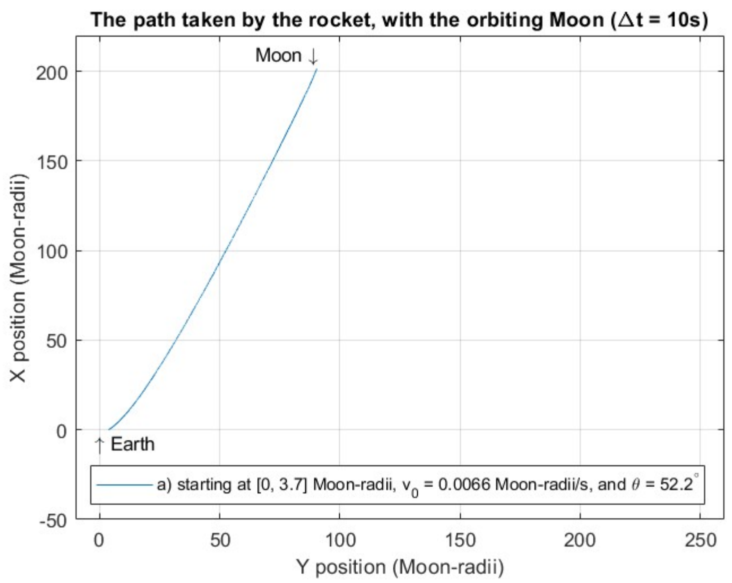

Figure 3: Trajectory of spacecraft with orbiting Moon.

References

[1] University of Oxford. *CO61: Rocket science*. 2024. URL: https://www-teaching.physics.ox.ac.uk/practical_course/scripts/srv/local/rscripts/trunk/Computing/CO61/CO61.pdf.

Appendix A My code

Listing 1: 'Core Script'

```
%% CO61 Rocket Science =============================================

% Cleaning up terminal
clear;
clc;
close all;

%% Functions =======================================================

function[acc] = acc_calc(p_r, p_m)
% Author: Gabriel Bristot, Date: 05/02/2025
% Defines the acceleration (2D vector) given a certain position of the moon
% and rocket the earth is considered stationary at the origin and scaled in M o o n radius .
% Input:
% * p_r: Position of the rocket (2D vector) with respect to the earth.
% * p_m: Position of the moon (2D vector) with respect to the earth.
% Output:
% * acc: Acceleration vector (2D vector)

G = 9.63*10^(-7);  % Gravitational Constant
M_e = 83.3;        % Mass of the Earth
M_m = 1;           % Mass of the Moon

d_e = sqrt(sum(p_r.*p_r));                    % Distance from the rocket to the earth
d_m = sqrt(sum((p_r-p_m).*(p_r-p_m)));        % Distance from the rocket to the moon

acc = - G*((M_e*p_r)/d_e^3) - G*((M_m*(p_r-p_m))/d_m^3); % Newton's gravitation formula

end

function[tout, pos] = simulate_rocket(init_pos, init_vel, moon_pos, t)
% Author: Gabriel Bristot, Date: 05/02/2025
% Simulate the rocket trajectory with the Earth and Moon influence. The coordinate
% used in this function is centred at E a r t h s centre (i.e. Earth centre at (0,0) )
% and scaled in M o o n 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)
% 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.

    rocket_pos = init_pos; % Initial conditions for
    vel = init_vel;        % velocity and position

    collision = false;     % Collision set to false

    step = 1;              % Initiating counter to 1
    N = length(t);         % Maximum number of iterations

    tout = [];             % Defining vector containing the times of all positions calculated
    pos = [];              % Defining position matrix

    d_t = t(2)-t(1);       % Defining the delta t used

    while collision == false & step <= N % Looping over all N or until collision with the moon

        d_m = sqrt(sum((rocket_pos-moon_pos(t(step))).*(rocket_pos-moon_pos(t(step))))); % Distance from the rocket to the moon

        if d_m <= 1 % Collision check

            collision = true;

        end

        % Start of Advanced Euler Method

        rocket_pos_intermidiate = rocket_pos + vel*d_t;

        vel_f = vel + (acc_calc(rocket_pos,moon_pos(t(step)))+acc_calc(rocket_pos_intermidiate,moon_pos(t(step)+t(1))))*(d_t/2);

        rocket_pos_f = rocket_pos + (vel_f+vel)*(d_t/2);

        rocket_pos = rocket_pos_f;
        vel = vel_f;

        % End of Andvanced Euler Method
```

```

        tout = [tout;t(step)];  % Updating the list of times
        pos = [pos;rocket_pos]; % Updating the matrix of positions

        step = step + 1; % Increasing the counter by 1

    end

end

%% Main code ==============================================================

% Creating the first figure of the stationary Moon case ==============
figure(1)

% Setting the initial conditions of the first simulation
init_pos = [0, 3.7];
init_vel = 0.0066*[cosd(89.9), sind(89.9)];
t = linspace(0, 350000, 35000);

% Setting the position of the Moon over time
moon_pos = @(t) [0, 222];

% Running the simulation
[tout1, pos] = simulate_rocket(init_pos, init_vel, moon_pos, t);

% Plotting data
plot(pos(:,2), pos(:,1));

hold on

% Setting the initial conditions of the second simulation
init_pos = [3.7, 0];
init_vel = 0.0066*[cosd(51.5), sind(51.5)];
t = linspace(0, 350000, 35000);

% Setting the position of the Moon over time
moon_pos = @(t) [0, 222];

% Running the simulation
[tout2, pos] = simulate_rocket(init_pos, init_vel, moon_pos, t);

% Plotting data
plot(pos(:,2), pos(:,1));

% Making the graph
xlabel('Y position (Moon-radii)')
ylabel('X position (Moon-radii)')

title('The two different paths taken by the rocket, with the sationary
    Moon (\Deltat = 10s)')
legend('a) starting at [0, 3.7] Moon-radii, v_0 = 0.0066 Moon-radii/s,
    and \theta = 89.9^{\circ}', ...
        'b) starting at [3.7, 0] Moon-radii, v_0 = 0.0066 Moon-radii/s,
            and \theta = 51.5^{\circ}', ...
```

```
        'Location', 'southwest')

txt1 = ['\uparrow Earth'];
text(-2,-7,txt1)
txt2 = ['Moon \downarrow'];
text(199,10,txt2)

grid on
ylim([-125,125])
xlim([-10,240])


% Creating the second figure of the orbiting Moon case ===============
figure(2)

% Setting the initial conditions of the simulation
init_pos = [0, 3.7];
init_vel = 0.0066*[cosd(52.2), sind(52.2)];
w = 2.6615*10^(-6);
t = linspace(0, 350000, 35000);

% Setting the position of the Moon over time
moon_pos = @(t) 222*[cos(w*t),sin(w*t)];

% Running the simulation
[tout3, pos] = simulate_rocket(init_pos, init_vel, moon_pos, t);

% Plotting data
plot(pos(:,2), pos(:,1));

% Making the graph
xlabel('Y position (Moon-radii)')
ylabel('X position (Moon-radii)')

title('The path taken by the rocket, with the orbiting Moon (\Deltat = 10s)')
legend('a) starting at [0, 3.7] Moon-radii, v_0 = 0.0066 Moon-radii/s, and \theta = 52.2^{\circ}' ...
        ,'Location', 'southwest')

txt1 = '\uparrow Earth';
text(-2,-7,txt1)
txt2 = 'Moon \downarrow';
text(65,210,txt2)

grid on
ylim([-50,220])
xlim([-10,260])
```

Listing 2: 'Side Project Script'

```matlab
%% CO61 Side project ==============================================================

% Cleaning up terminal
clear;
clc;
close all;

%% Functions ======================================================================

function[acc] = acc_calc(p_r, p_m)
% Author: Gabriel Bristot , Date: 05/02/2025
% Defines the acceleration (2D vector) given a certain position of the moon
% and rocket the earth is considered stationary at the origin and scaled in M o o n radius .
% p_r: Position of the rocket (2D vector) with respect to the earth.
% p_m: Position of the moon (2D vector) with respect to the earth.
% a: Acceleration vector (2D vector)

G = 9.63*10^(-7);  % Gravitational Constant
M_e = 83.3;        % Mass of the Earth
M_m = 1;           % Mass of the Moon

d_e = sqrt(sum(p_r.*p_r));                  % Distance from the rocket to the earth
d_m = sqrt(sum((p_r-p_m).*(p_r-p_m)));      % Distance from the rocket to the moon

acc = - G*((M_e*p_r)/d_e^3) - G*((M_m*(p_r-p_m))/d_m^3); % Newton's gravitation formula

end

function[d_min] = simulate_rocket(init_pos ,init_vel ,moon_pos ,t)
% Author: Gabriel Bristot, Date: 05/02/2025
% Simulate the rocket trajectory with the Earth and Moon influence. The coordinate
% used in this function is centred at E a r t h s centre (i.e. Earth centre at (0,0) )
% and scaled in M o o n 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)
% 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:
%     d_min: a scalar representing the minimum distance between the centre of
% the Moon and the rocket

    rocket_pos = init_pos; % Initial conditions for
    vel = init_vel;        % velocity and position

    collision = false;     % Collision set to false

    step = 1;              % Initiating counter to 1
    N = length(t);         % Maximum number of iterations

    d_t = t(2)-t(1);       % Defining the delta t used

    d_min = 1000;          % Defining the minimu idstance to the moon

    while collision == false & step <= N % Looping over all N or until collision with the moon

        d_e = sqrt(sum(rocket_pos.*rocket_pos));
        d_m = sqrt(sum((rocket_pos-moon_pos(t(step))).*(rocket_pos-moon_pos(t(step))))); % Distance from the rocket to the moon

        if d_m <= 1 || d_e <= 3.65  % Collision check

            collision = true; % Uptade collision

        end

        if d_m < d_min  % Check if new distance to the Moon is smaller then minimum distance

            d_min = d_m; % Update d_min

        end

        % Start of Advanced Euler Method

        rocket_pos_intermidiate = rocket_pos + vel*d_t;

        vel_f = vel + (acc_calc(rocket_pos,moon_pos(t(step)))+acc_calc(rocket_pos_intermidiate,moon_pos(t(step)+t(1))))*(d_t/2);

        rocket_pos_f = rocket_pos + (vel_f+vel)*(d_t/2);

        rocket_pos = rocket_pos_f;
        vel = vel_f;

        % End of Andvanced Euler Method

        % Increasing the counter by 1
```

```
        step = step + 1;

    end

end

%% Main program ===================================================

% Number of points sampled in each axis
N = 100;

% Initial position
init_pos = [0, 3.7];

% Variables to be used in program
w = 2.6615*10^(-6);
t = linspace(0, 350000, 35000);

% Minimum and maximum values to be simulated
v_max = 0.005;
v_min = 0.010;

theta_max = 80;
theta_min = 90;

% Small change in variables every simulation
d_v = (v_max-v_min)/(N-1);
d_theta = (theta_max-theta_min)/(N-1);

% Position of the moon over time
moon_pos = @(t) 222*[0,1];

% defining the initial magnitude of the velocity
vel_mag = v_min;

% Pre defining a matrix M thta contains all the simulation data
Data = zeros(N*N,3);

% Keeping track of the number of rows
row = 1;

% Double for loop

% Cycles through v
for counter_1 = 1:N

    % Starting theta at 0 in every inner loop
    theta = theta_min;

    % Cycles through theta
    for counter_2 = 1:N

        % Defining the velocity vector
        initial_vel = vel_mag*[cosd(theta), sind(theta)];

```

```
        % Simulating to find d_min
        d_min = simulate_rocket(init_pos, initial_vel, moon_pos, t);

        % Entering the data into data matrix
        Data(row,1) = vel_mag;
        Data(row,2) = theta;
        Data(row,3) = d_min;

        % Updating counters
        theta = theta + d_theta;

        row = row + 1;

    end

    % Updating counter
    vel_mag = vel_mag + d_v;

    % Cosmetic detail of how far into the process the computation is
    fprintf('\r%3.0f%% completed', (counter_1/N) * 100);

end

%% Extract columns ==================================================

mag_v = 1000*Data(:,1);
theta = Data(:,2);
min_distance = Data(:,3);

% Get unique values for the grid
mag_v_vals = unique(mag_v);
theta_vals = unique(theta);

% Reshape the data into a grid
[X, Y] = meshgrid(mag_v_vals, theta_vals);
Z = griddata(mag_v, theta, min_distance, X, Y, 'cubic');

% Plot the surface
figure;
surf(X, Y, Z);
set(gca, 'ZScale', 'log')
xlabel('Speed (Moon-radii/s x 10^{-3})');
ylabel('Theta (degrees)');
zlabel('D_{min} (Moon-radii)');
title('Surface Plot of Minimum Distance to the Moon (D_{min})');
colorbar;
set(gca, 'ColorScale', 'log')
c = colorbar;
c.Label.String = 'D_{min} (Moon-radii)';
shading interp;

% Add contour lines on the surface
hold on;
contour3(X, Y, Z, 20, 'k');
```

Appendix B Extra analysis

B.1 Stationary Moon

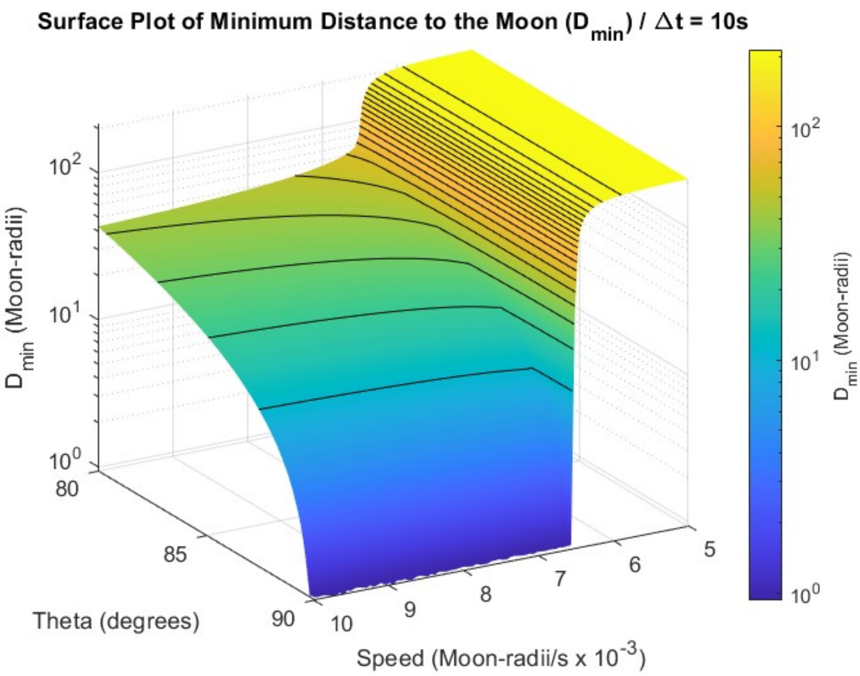

Figure 4: Surface generated by the function D_{min} with the starting position being [0, 3.7] Moon-radii

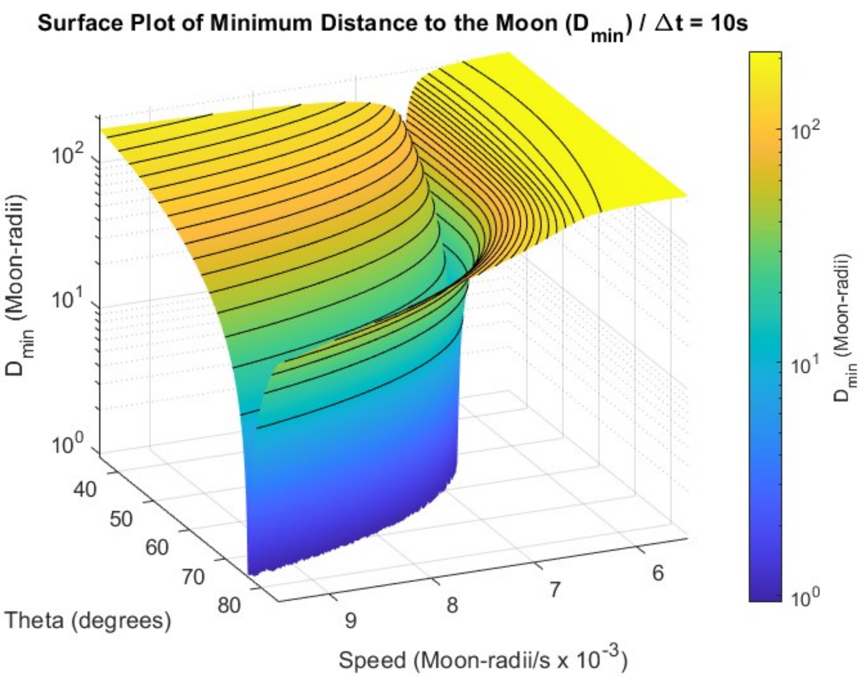

Figure 5: Surface generated by the function D_{min} with the starting position being [3.7, 0] Moon-radii

B.2 Orbiting Moon

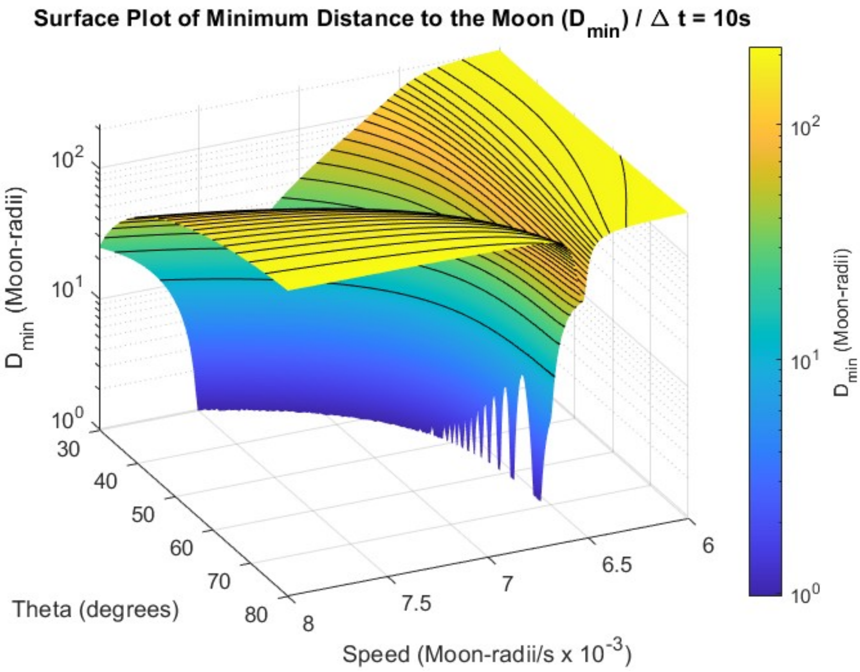

Figure 6: Surface generated by the function D_{min} with the starting position being [0, 3.7] Moon-radii.

All Figures above (4, 5, 6) show a minimum distance to the Moon of 1 Moon-radius since the simulation stops when the distance of the spacecraft to the centre of the Moon is 1 Moon-radius. Note that the "spikes" seen in Figure 6 are likely the result of limited sampling. Each of the figures required either 10000 or 40000 simulations that took $\approx$ 15 min to run on my Intel i7-7700 processor.