Report of the Real Time Systems Project
LEM

Matteo De Benedetti 561663
matteo.debenedetti91@gmail.com
Ingegneria Robotica e dell'Automazione

February 14, 2018

Contents

Chapter 1

Project Introduction

1.1 LEM and Scenario description

The project requires to simulate a Lunar Exploration Vehicle (LEM) that has to land on the moon, pick up a rock sample, take off and dock in orbit to a Command and Service Module (CSM). The LEM also has to navigate trough an asteroid belt during both descent and ascent.

The simulated LEM has the same physical characteristics of the LEM used during the Apollo program [1], except for the thrust of the main engine, which has been doubled to make manual control easier.
For the same reason, and to also make the scenario faster, the CSM orbit is at 4% of its real altitude and velocity. This allows the entire scenario (descent, landing, ascent and docking) to last less than 10 minutes while, during the Apollo 11 mission, the descent alone took 1 hour, 33 minutes and 41 seconds from un-docking to touchdown [2].

The project also requires to develop both manual and autonomous control.
Manual control can be tricky and difficult. The user can activate the main engine and rcs motors separately to control both translation and rotational motion. It is often useful to focus on the telemetry data (x and y positions and velocities) and look at the surface and docking vectors, especially during landing and docking.
Autonomous control will instead easily navigate through the entire process.

While the descent is quite realistic, the ascent part has been simplified: instead of a real orbital rendezvous, which is an extremely complicated and long procedure, the LEM only needs to get to the correct altitude and orbital velocity, then the CSM will spawn nearby, ready for docking.

The mission may result in a failure in many cases:

- The LEM collides with an asteroid, no matter what the relative velocity between the two is.
- The LEM lands outsides the limits of x and y velocities and orientation.
- The LEM docks with the CSM outsides the limits of x and y relative positions, velocities and orientation.

The code is organized in three libraries, one for the pthread functions, another for the Allegro environment and the last one for the scenario itself. They are further divided into a source .c file and a header .h file. These libraries contain useful functions, structures and constants.
The main function, the tasks functions and a few user defined functions used inside the tasks are all included in a .c file.

When launching the executable it is necessary to pass an additional argument that allows to start at a specific point, such as before the asteroid belt, right before landing or docking and so on.

Chapter 2

Design and Modeling

2.1 Physical Model

The LEM is simulated using the Equations of Motion, where the acceleration is given by the engines thrust and gravity.

For the linear motion the following system of equations is used:

$$x(t) = \frac{1}{2}\ddot{x}(t_0)(t - t_0)^2 + \dot{x}(t_0)(t - t_0) + x(t_0); \tag{2.1}$$

$$\dot{x}(t) = \dot{x}(t_0) + \ddot{x}(t_0)(t - t_0); \tag{2.2}$$

For the rotational motion the following system of equations is used:

$$\theta(t) = \frac{1}{2}\ddot{\theta}(t_0)(t - t_0)^2 + \dot{\theta}(t_0)(t - t_0) + \theta(t_0); \tag{2.3}$$

$$\dot{\theta}(t) = \dot{\theta}(t_0) + \ddot{\theta}(t_0)(t - t_0); \tag{2.4}$$

In these equations $\ddot{x}$ is the linear acceleration of the LEM, which can be found as the sum of all the forces acting on the LEM over its mass m, as shown in Figure 2.1.

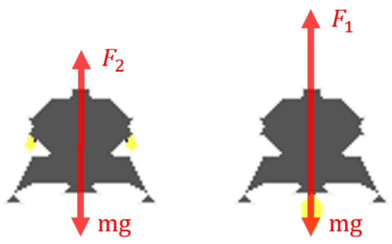


Figure 2.1: Forces acting on the LEM

Similarly, $\ddot{\theta}$, which is the angular acceleration, can be found as the sum of all the torques τ acting on the LEM over the moment of inertia J, as shown in figure 2.2.

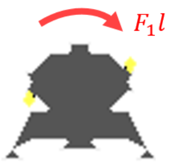


Figure 2.2: Torque acting on the LEM

Given the thrust values of the engines, acceleration and torque can be easily found using the following equations:

$$\ddot{x} = \frac{F}{m}, \quad \ddot{\theta} = \frac{\tau}{J}; \tag{2.5}$$

The asteroids and the CSM are modeled the same way, with the only difference that, since they can't generate any thrust, their equations are much shorter.

As previously mentioned, the orbital insertion and rendezvous have been simplified: instead of actually computing and intersect the two spacecrafts orbits, when the LEM is ascending the gravitational acceleration is computed as a function of the difference between the LEM horizontal velocity and the CSM orbital velocity. This way the LEM only needs to get to orbital velocity to counteract the pull of gravity and behave as it is on a stable circular orbit.

2.2 Tasks diagram

The tasks and resources diagram of the application is shown here in Figure 2.3.

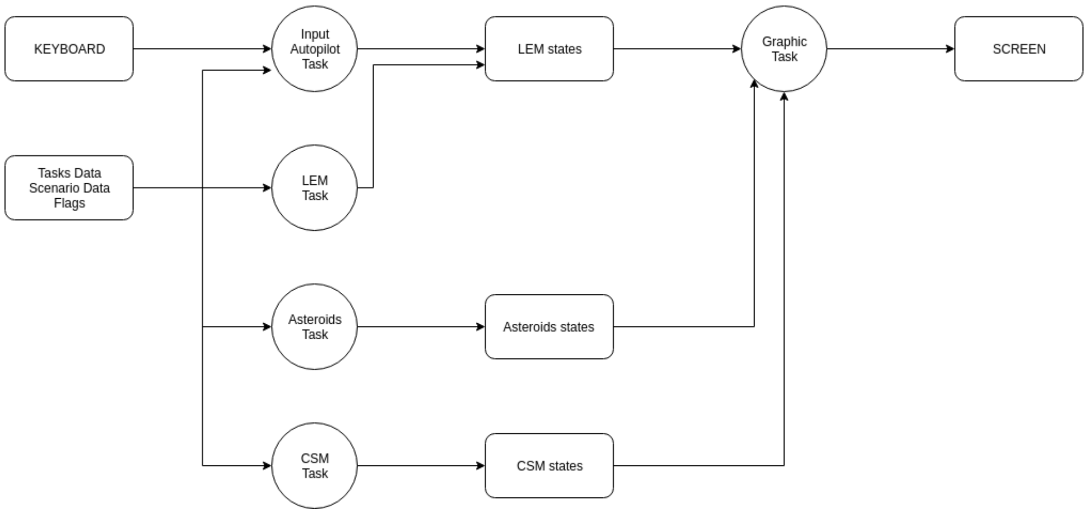


Figure 2.3: Tasks and Resources diagram

2.3 LEM task

The LEM task is in charge of computing and updating the position, orientation, velocity and acceleration of the LEM in response to engine burns, rcs thrusts and, obviously, gravity, using the equations showed in Section 2.1.

It also handles the landing, checking if the limits on positions and velocities are exceeded. If that happens, it communicates it to the other tasks via a flag, which is used by the Graphic task to show a simple explosion animation and then exit from the application.

To do so, this task uses a set of global variables: a dedicated structure contains all the useful states of the LEM (positions, velocities and activated engines).

2.4 Graphic task

Instead of relying on the individual tasks to print to the screen, a single Graphic task handles it.

At the very beginning it will initialize the screen, procedurally generate the stars in the background and load the necessary bitmaps and print the right panel, where the controls are shown.

The periodic part of the task will read all the global states of the LEM, CSM and asteroids and print them. It will also show the lunar ground when the LEM is close to the surface, the selected surface/docking vector, the data in the lower panel and the altitude map and handle the explosion animation.

In particular, in order to save memory, it will print the engines muzzles not by using different bitmaps for every combination of engines activation, but by positioning the individual muzzles on the bitmap of the LEM at the correct position and orientation.

The Display task also takes into account the problem of flickering. It happens when the screen refresh rate is faster than the drawing process. It can be avoided using the instruction *blit* a single time. This technique, known as *Double Buffering*, consists in creating a bitmap as big as the screen, copying there all the new content and then using the instruction *blit* to overwrite the screen with it.

2.5 Input task

The input task has two main modes of operation: it will interpret the user's input or, if the autopilot is enabled, control the spacecraft itself.

There are two autopilots, one for the descent phase and landing, the other one for the ascent phase and docking. The first one is inspired to a real technique, particularly used on airless bodies, called *Gravity Turn* [3] to cancel most of its velocity, then it will guide the LEM trough the asteroid belt to a safe landing on the surface. The second one will take care of navigating through the asteroid belt and of performing an orbital insertion to dock with the CSM.

To be able to correctly control the LEM, avoid asteroids and dock with the CSM, the Input task needs to access to most of the global variables containing their states. It will do so using Mutex semaphores to protect these global variables.

2.6 Asteroids task

The asteroids task will spawn the asteroids when the LEM is inside the Asteroid Belt. The altitude values where the asteroids are created are fixed, but the velocity and distance from the LEM are randomly generated to create a real challenge to the pilot.

The task will also compute and update the asteroids states and handle the collisions with the lander.

To avoid wasting memory, the asteroids are created and their states updated only when they are close to the spacecraft, this can be easily obtained with a conditional check on the LEM coordinates.

2.7 CSM task

The CSM task acts in a similar way to the asteroids task: it will spawn the CSM when the LEM is in the correct orbit and handle docking and possible collisions.

The CSM has its own structure for its states. This structure is slightly simpler than the LEM one, since it does not need to take into account the state of the engines.

2.8 Shared Global Variables

Global variables are used to share data between tasks, under the protection of the Mutex semaphores.

The most important information to be shared are the states of the spacecrafts and asteroids. To do so, two structures are defined: one is simpler and contains only variables for the positions, orientation and velocities. The second one is more complex and can also store the status of the main engine and the rcs. While the asteroids and the CSM, since they are not controllable, use the first structure, the LEM uses the second one.

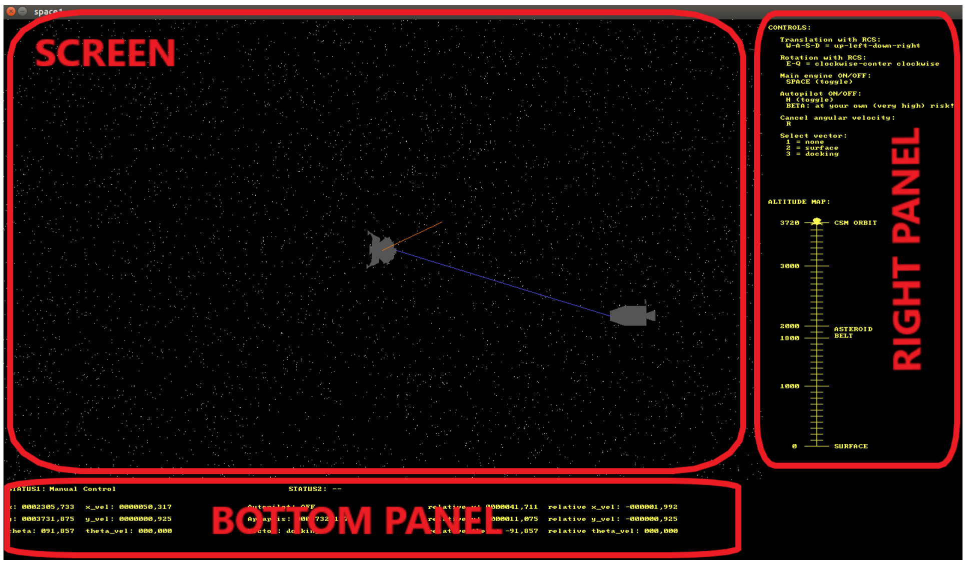
There is also a number of flags to keep track of other information. For example: if the LEM is landed, has docked, has collided and exploded, if the autopilot is enabled or disabled, if either surface or docking vector is selected and two status messages

Chapter 3

User Interface

The screen is divided into three parts:

- Screen;
- Right Panel;
- Bottom Panel;



```
CONTROLS:

   Translation with RCS:
    W-A-S-D = up-left-down-right

   Rotation with RCS:
    E-Q = clockwise-conter clockwise

   Main engine ON/OFF:
    SPACE (toggle)

   Autopilot ON/OFF:
    H (toggle)
    BETA: at your own (very high) risk!

   Cancel angular velocity:
    R

   Select vector:
    1 = none
    2 = surface
    3 = docking

ALTITUDE MAP:
```

```
ATUS1: Manual Control                        STATUS2: --

: 0002305,733   x_vel: 0000050,317                                        0000041,711   relative x_vel: -000001,992
: 0003731,875   y_vel: 0000000,925                                        000011,075   relative y_vel: -000000,925
heta: 091,857   theta_vel: 000,000                                        -91,857   relative theta_vel: 000,000
```

Figure 3.1: Window division

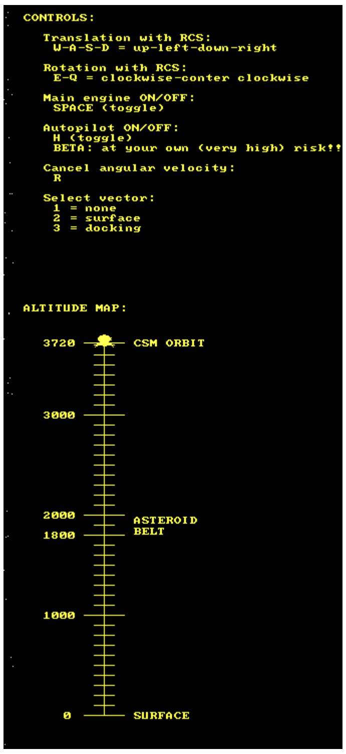
3.1 Screen

It shows the LEM in the center on a procedurally generated star background. The spacecraft will always be in the center of the screen while all the other objects (asteroids, CSM, ground) will appear as moving relatively to it. In this way the Graphic task will load and show only what is close to the LEM. A proportion of 10 pixels = 1 meter has been chosen, to allow the user to have a better understanding of the LEM surroundings.

3.2 Right Panel

The right panel shows the LEM manual controls in the upper part and an Altitude Map in the lower part.

The Altitude Map helps the user to quickly see the LEM altitude showing the spacecraft's position with respect to other important altitudes, such as the CSM orbit, the Moon surface and the asteroid belt.



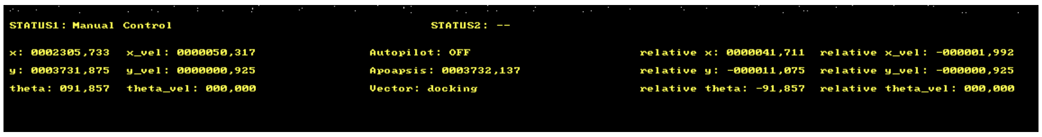
```
CONTROLS:

    Translation with RCS:
      W-A-S-D = up-left-down-right

    Rotation with RCS:
      E-Q = clockwise-conter clockwise

    Main engine ON/OFF:
      SPACE (toggle)

    Autopilot ON/OFF:
      H (toggle)
      BETA: at your own (very high) risk!!

    Cancel angular velocity:
      R

    Select vector:
      1 = none
      2 = surface
      3 = docking


ALTITUDE MAP:
```

Figure 3.2: Right Panel

3.3 Lower Panel

The lower panel contains the telemetry data of the LEM, the status of the Autopilot and the selected vector. When the CSM is close it will also display its data and relative velocities and positions, which are really useful informations when docking.

To help the user understand what the autopilot is doing two status messages are displayed.



```
STATUS1: Manual  Control                          STATUS2: --

x: 0002305,733   x_vel: 0000050,317              Autopilot: OFF                 relative x: 0000041,711   relative x_vel: -000001,992
y: 0003731,875   y_vel: 0000000,925              Apoapsis: 0003732,137          relative y: -000011,075   relative y_vel: -000000,925
theta: 091,857   theta_vel: 000,000              Vector: docking                relative theta: -91,857   relative theta_vel: 000,000
```

Figure 3.3: Bottom Panel

3.4 Vectors

As previously mentioned, the user can activate and display two vectors.

The first one shows the surface velocity of the LEM. It is particularly useful to correctly orient the spacecraft when canceling velocity burning the main engine.

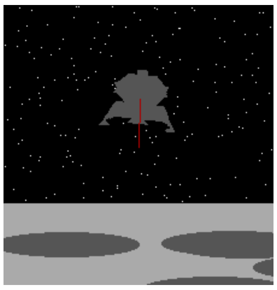


Figure 3.4: Surface vector

The second option actually shows two vectors: the blue vector connects the CSM and LEM docking ports, the orange vector shows the velocity relative to the CSM. This information is vital if the docking is performed manually, in order to properly align the two spacecrafts and control their relative velocity.

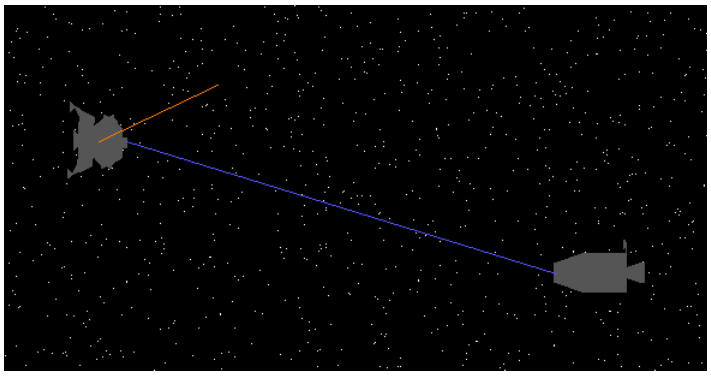


Figure 3.5: Docking vector

Chapter 4

Program Phases

The mission, from the beginning to the end, can be divided in a few phases: the LEM will initially descend from orbit, go through the asteroid belt, then land on the surface. After landing it will take off, navigate through the asteroid belt again, raise its apoapsis to the CSM orbit, then burn at apoapsis to perform the orbit insertion and eventually dock with the CSM.

4.1 Descent

The LEM will perform a Gravity Turn Maneuver to cancel most of its horizontal and vertical velocity. This maneuver consists of burning in the opposite direction of the velocity vector to be able to cancel both horizontal and vertical velocity at the same time. It will then prepare to face the asteroid belt.

4.2 Asteroids avoidance

Inside the asteroid belt the lander will control both vertical and horizontal velocity. When it finds itself on a collision path with an asteroid it will try to avoid it acting on both vertical velocity with the main engine and horizontal velocity with the rcs.

It will also decide if it would be better to avoid the asteroid going ”up and left” or ”down and right” as shown in Figure 4.1.

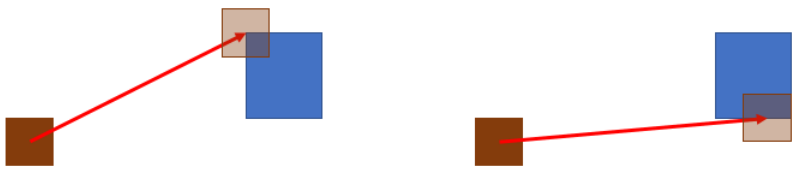


Figure 4.1: Left: collision to be avoided going ”down and right”. Right: collision to be avoided going up and left”

4.3 Landing

After having successfully navigated through the asteroid belt, the LEM will land on the surface canceling any remaining horizontal velocity and controlling the vertical velocity to touchdown at 3 m/s.

4.4 Ascent and Orbital Insertion

The LEM will then take off and increase its vertical velocity until it reaches again the asteroid belt, here it will avoid the asteroids as described above.

Immediately after having navigated the asteroid belt, the spacecraft will burn upwards to raise its apoapsis to the CSM orbit altitude. It will wait until it reaches apoapsis and then burn sideways to bring its horizontal velocity to orbital velocity and correctly get into orbit. This fairly simple Orbital Insertion technique is often referred to as Two Burns strategy. It is mostly used on airless bodies, otherwise turning the spacecraft would most likely result in a catastrophic failure due to aerodynamic forces.

4.5 Docking

Once the LEM is in the correct orbit it will dock with the CSM. It will adjust its altitude and orientation and then approach the CSM at a controlled velocity to eventually dock at a safe relative speed of 0.2 m/s.

Chapter 5

Experimental Results

5.1 Simulation Time

The entire simulation takes approximately 9 to 10 minutes when the autopilot is on. With manual control it greatly depends on the user's ability. Many features have been added with this problem in mind, such as the surface/docking vector, the altitude map and the amount of data shown in the bottom panel.

The approach used while controlling the LEM also plays an important role: an "aggressive" pilot may be faster but also run into problems since, even with the doubled main engine thrust, it takes a while to slow down.

5.2 Autonomous and Manual Control

Only when facing the asteroids the human control may actually compete with the autopilot in terms of performances and survival chances.

If only one asteroid is moving towards the LEM the implemented collision avoidance algorithm is virtually infallible. Problems may arise if two asteroids are going to collide with the spacecraft at roughly the same time. This eventuality is quite unlikely to happen: performing a series or simulations it has been witnessed only once in 20 times, meaning a 5% probability that an asteroid encounter would result in a collision.

Though this may happen in this simulation due to the random spawn of the asteroids, it is almost impossible in a real life mission, since distances between asteroids are far bigger. For example in the Asteroid Belt in our Solar System the average distance between asteroids is about 966000 km and the odds of a probe running into an asteroid are estimated at less than one in a billion [4].

5.3 Data Plots

The telemetry data of the LEM are also logged in a .txt file to be able to further analyze them.

In Figure 5.1 and 5.2 the values of altitude, horizontal and vertical velocity are plotted over time and show how the autopilot controls the LEM through the mission.

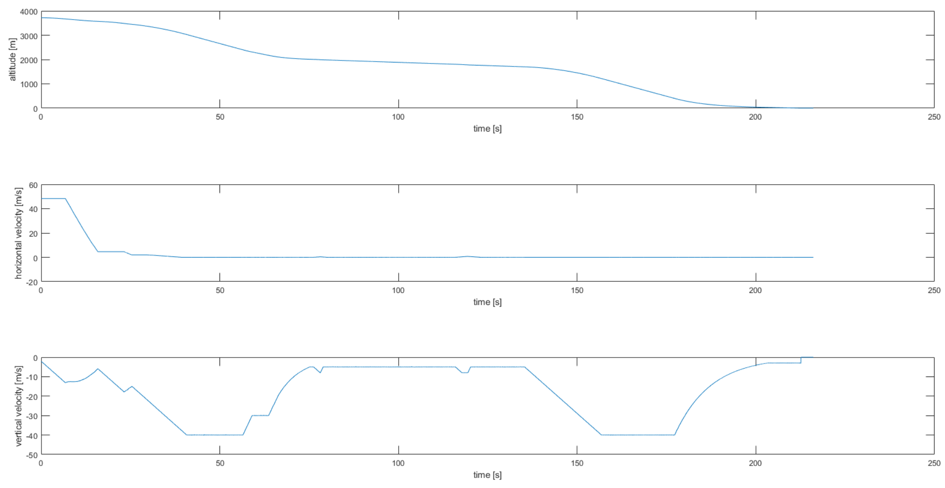


Figure 5.1: Descent phase

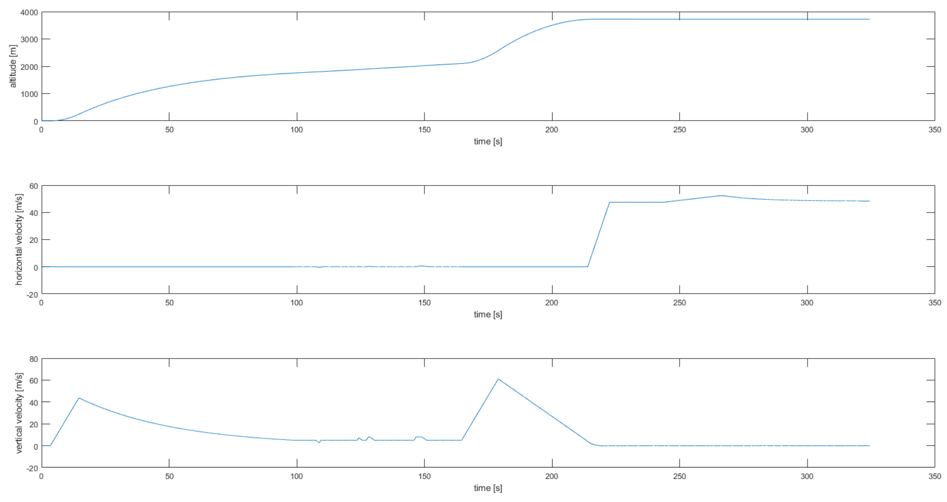


Figure 5.2: Ascent phase

A more meaningful graph can be obtained plotting both horizontal and vertical velocity over altitude, as shown in Figure 5.3 and 5.4. The two plots clearly show, between 1800 and 2000 meters, the small corrections to both horizontal and vertical velocity that the LEM had to do to avoid asteroids. They also show a considerable increase in horizontal velocity at 2780 meters when the LEM performs the Orbit Insertion burn during the ascent phase, and the vertical velocity going to zero when the LEM is landing during the descent phase.

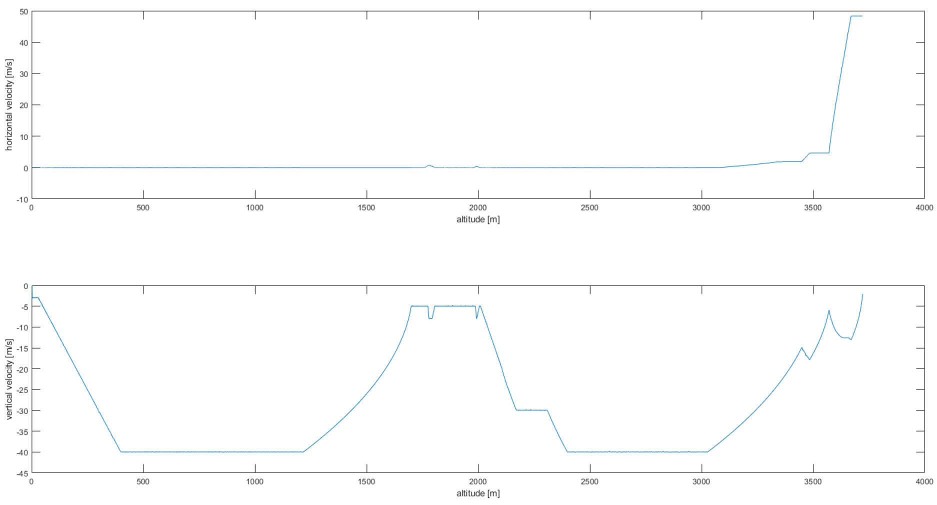


Figure 5.3: Descent phase

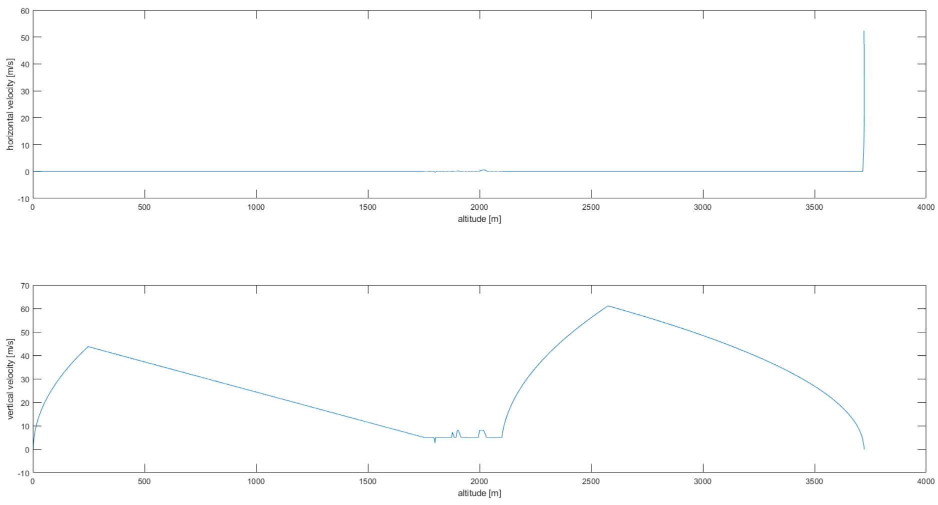


Figure 5.4: Ascent phase

Bibliography

[1] Apollo Lunar Module technical data
https://en.wikipedia.org/wiki/Apollo_Lunar_Module.

[2] Apollo 11 Lunar Orbit Phase
https://history.nasa.gov/SP-4029/Apollo_11g_Lunar_Orbit_Phase.htm.

[3] Gravity Turn maneuver
https://en.wikipedia.org/wiki/Gravity_turn.

[4] Stern Alan (June 2 2006) "*New Horizons Crosses The Asteroid Belt*" Space Daily. Retrieved 2007-04-14
http://www.spacedaily.com/reports/New_Horizons_Crosses_The_Asteroid_Belt.html.