# Assignment 0: Beresheet 2D Landing Simulation

Course: Introduction to Aerospace Engineering

Student Name: [Your Name]

## Part 1 – Crash Analysis

On April 11th, 2019, the Israeli Beresheet lunar lander crashed during its final descent to the Moon’s surface. According to reports, the crash was caused by a malfunction in the Inertial Measurement Unit (IMU), which triggered a chain of failures. The IMU was reset, which in turn shut off the main engine. The system couldn’t recover in time, and the lander impacted the Moon at high speed. This incident demonstrates the critical importance of fault tolerance and redundancy in autonomous space missions.

## Part 2 – 2D Simulation Setup and PID Control

The simulation models Beresheet's descent from 30,000 meters altitude and 1700 m/s horizontal velocity.

It includes:

* - Vertical and horizontal motion with PID control
* - Gravity: 1.62 m/s²
* - Main engine thrust: 430 N (vertical)
* - Side engine thrust: 200 N (horizontal total)
* - Mass variation due to fuel consumption
* - Target velocities: < 2.5 m/s for both vertical and horizontal

## Part 3 – Python Simulation Code

The following Python code simulates the descent with dual PID controllers:

# 2D PID simulation (vertical + horizontal)  
# G = 1.62 m/s^2, vertical thrust = 430 N, side thrust = 200 N  
# PID tuned manually for each axis  
# Simulation ends when altitude = 0

See full code in the submission folder (beresheet\_2d\_pid\_simulation.py).

## Part 4 – Results

Final values at landing:

- Altitude: 0.0 m

- Vertical Velocity: -304.46 m/s

- Horizontal Velocity: 1112.24 m/s

- Mass: 150.00 kg (dry mass)

## Appendix – Full Python Code (Student Style)

# Beresheet 2D Lander Simulation – Student Style (with comments and trial attempts)  
  
import numpy as np  
import matplotlib.pyplot as plt  
  
# === Constants and initial values ===  
G = 1.62 # Moon gravity  
main\_thrust = 430 # N  
side\_thrust = 200 # N  
mass\_dry = 150 # kg  
fuel = 120 # kg  
mass = mass\_dry + fuel  
burn\_rate = 0.2 # kg/s  
  
# initial conditions  
altitude = 30000 # m  
vv = -24.8 # vertical velocity  
vh = 1700 # horizontal velocity  
dt = 0.1 # time step  
  
# PID targets  
target\_vv = -2.0  
target\_vh = 0.0  
  
# PID coefficients – tried some values until it didn't crash instantly  
kp\_v, ki\_v, kd\_v = 0.6, 0.01, 1.2  
kp\_h, ki\_h, kd\_h = 0.4, 0.0, 0.8  
  
int\_vv = 0  
int\_vh = 0  
prev\_err\_vv = 0  
prev\_err\_vh = 0  
  
time = 0  
max\_time = 1000 # max sim time  
  
# logs (for graph later)  
t\_log = []  
alt\_log = []  
vv\_log = []  
vh\_log = []  
mass\_log = []  
  
# === Main simulation loop ===  
while altitude > 0 and time < max\_time:  
 # --- Vertical PID ---  
 err\_vv = target\_vv - vv  
 int\_vv += err\_vv \* dt  
 der\_vv = (err\_vv - prev\_err\_vv) / dt  
 pid\_vv = kp\_v \* err\_vv + ki\_v \* int\_vv + kd\_v \* der\_vv  
 prev\_err\_vv = err\_vv  
  
 # --- Horizontal PID ---  
 err\_vh = target\_vh - vh  
 int\_vh += err\_vh \* dt  
 der\_vh = (err\_vh - prev\_err\_vh) / dt  
 pid\_vh = kp\_h \* err\_vh + ki\_h \* int\_vh + kd\_h \* der\_vh  
 prev\_err\_vh = err\_vh  
  
 # limit thrusts to engine capabilities  
 acc\_v = min(pid\_vv, main\_thrust / mass)  
 acc\_h = np.clip(pid\_vh, -side\_thrust / mass, side\_thrust / mass)  
  
 # fuel usage (just assume constant burn when thrusting)  
 if mass > mass\_dry:  
 mass -= burn\_rate \* dt  
 if mass < mass\_dry:  
 mass = mass\_dry  
 else:  
 acc\_v = 0  
 acc\_h = 0  
  
 # physics update  
 net\_acc\_v = acc\_v - G  
 vv += net\_acc\_v \* dt  
 vh += acc\_h \* dt  
 altitude += vv \* dt  
 if altitude < 0:  
 altitude = 0  
  
 # log everything  
 t\_log.append(time)  
 alt\_log.append(altitude)  
 vv\_log.append(vv)  
 vh\_log.append(vh)  
 mass\_log.append(mass)  
  
 time += dt  
  
# === Print end result (even though crash is likely) ===  
print("Final Altitude:", altitude)  
print("Vertical Velocity:", vv)  
print("Horizontal Velocity:", vh)  
print("Final Mass:", mass)  
  
# === Plot ===  
plt.figure(figsize=(10, 6))  
plt.title("Beresheet 2D Landing Simulation – Combined")  
plt.plot(t\_log, alt\_log, label="Altitude (m)")  
plt.plot(t\_log, vv\_log, label="Vertical Velocity (m/s)")  
plt.plot(t\_log, vh\_log, label="Horizontal Velocity (m/s)")  
plt.plot(t\_log, mass\_log, label="Mass (kg)")  
plt.xlabel("Time (s)")  
plt.legend()  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

Graph showing the results from the simulation:

