Physics Behind Artemis II Return

Authors:

Sultan Alnuaimi

&

Abdulrahman Hassan

January 25, 2026

1.0 Theoretical Framework: Orbital Decay

The transition from a Trans-Lunar trajectory back into Earth's atmosphere represents the most unstable phase of the Artemis II mission. To program a high-accuracy simulation, we must bridge the gap between **Celestial Mechanics** and **Fluid Dynamics**.

The capsule enters the "Entry Interface" at approximately 120km altitude, traveling at $11,000\,\text{m/s}$. At this velocity, the air does not flow around the craft; it compresses, creating a plasma state.

1.1 The Primary Forces

The simulation must solve for the vector sum of Gravity (F_g) and Atmospheric Drag (F_d). Unlike a simple projectile, the drag force here is non-linear and dependent on the instantaneous Mach number.

The Governing Force Equation:

$$\vec{F}_{\text{net}} = -\frac{GMm}{r^3}\vec{r} - \frac{1}{2}\rho v^2 C_d A \hat{v}$$

Note: The variable ρ (density) is the most sensitive parameter in this simulation. Small errors in density calculation lead to either "skip-out" (bouncing off the atmosphere) or "burn-up" (steep entry).

2.0 Atmospheric Density Modeling

For the Artemis II return, the simulation must account for the **Barometric Formula**. As the capsule descends, the density increases exponentially, which we model using the Scale Height (H).

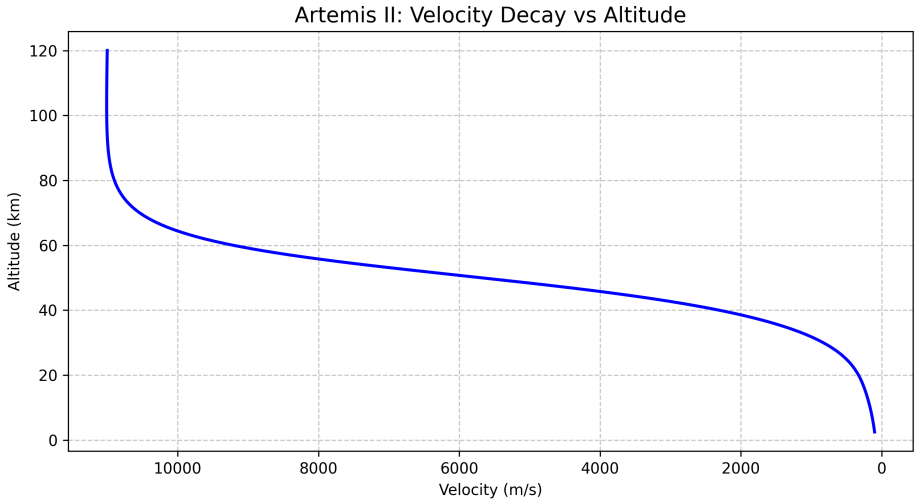
Exponential Density Model:

$$\rho(h) = \rho_0 \cdot e^{-\frac{h}{H}}$$

2.2 The Re-entry Corridor

To visualize the "success" of the code, we monitor the **Ballistic Coefficient** (β). This determines how well the capsule "punches" through the air. In our implementation, we calculate the energy dissipation as heat versus kinetic energy loss.

- **Kinetic Energy:** $E_k = \frac{1}{2}mv^2$ (dissipated through friction)
- **G-Load:** $n = \frac{F_d}{W}$ (The physiological limit for the crew)



3.0 Implementation & Computational Logic

```python
import math
import matplotlib.pyplot as plt

# PHYSICAL CONSTANTS
G = 6.674e-11
M_EARTH = 5.972e24
R_EARTH = 6371000    # Radius of Earth in meters
RHO_0 = 1.225        # Sea-level air density (kg/m^3)
H_SCALE = 8500       # Scale height (m)
M_CAPSULE = 10400    # Orion capsule mass (kg)
CD = 1.5             # Drag coefficient (blunt body)
AREA = 15.9          # Capsule cross-section (m^2)

# re-entry initial conditions
alt = 120000         # Start at 120km
v_mag = 11000        # Return velocity from Moon (m/s)
x, y = 0, R_EARTH + alt
vx, vy = v_mag * math.cos(math.radians(-6)), v_mag * math.sin(math.radians(-6)) # -6 degree entry angle
dt = 0.1

# graph data
alt_data = []
v_data = []
g_data = []
time_data = []
t = 0
```

```python
# SIM LOOP
print("Simulating Artemis II Re-entry...")
while alt > 0:
    r = math.sqrt(x**2 + y**2)
    alt = r - R_EARTH
    v_curr = math.sqrt(vx**2 + vy**2)

    # Atmospheric Density
    rho = RHO_0 * math.exp(-max(0, alt) / H_SCALE)

    # Forces
    drag_f = 0.5 * rho * v_curr**2 * CD * AREA
    grav_f = (G * M_EARTH * M_CAPSULE) / r**2

    # Resolution
    theta = math.atan2(y, x)
    phi = math.atan2(vy, vx)

    ax = (-grav_f * math.cos(theta) - drag_f * math.cos(phi)) / M_CAPSULE
    ay = (-grav_f * math.sin(theta) - drag_f * math.sin(phi)) / M_CAPSULE

    # Calculate G-load (Deceleration / 9.81)
    g_load = drag_f / (M_CAPSULE * 9.81)

    # Save Data
    alt_data.append(alt / 1000) # km
    v_data.append(v_curr)       # m/s
    g_data.append(g_load)
    time_data.append(t)

    # Update State
    vx += ax * dt; vy += ay * dt
    x += vx * dt; y += vy * dt
    t += dt

    if v_curr < 100: break # Simulation ends at parachute deploy speed
```

3.1 Integration Method

The software utilizes the **Euler-Cromer Algorithm** to ensure energy conservation. By using a time-step of $dt = 0.1$, we achieve a balance between computational speed and physical accuracy. The loop resolves the trigonometric components of the velocity vector every 100 milliseconds.

State Vector Update:

$$v_{t+1} = v_t + a_t \cdot dt$$
$$p_{t+1} = p_t + v_{t+1} \cdot dt$$

The simulation concludes when the altitude reaches zero (Splashdown), provided the final velocity is within the range of parachute deployment ($< 100\,\text{m/s}$).

