SPCE 5045 -- Homework #1

Course: SPCE 5045 Space Mission Analysis
Assignment: Homework #1 (50 points total, Chapter 1)
Semester: Spring 2025
Author: Jordan Clayton
Date: 2026-02-02

Chapter 1:

NOTE: Use the common terminology of stage number, i.e., “1st stage” is the bottom or biggest stage, 2nd stage is the next one “up”, etc. (See discussion on pages 10 and 11 in the text for the ΔV required)

Problem 1 (15 points)

Assume that a single rocket stage can provide a delta V of 3 km/sec. Determine how many stages would it take to get to orbit from the surface of:

a) the Moon (ΔV=1.7 km/s)
b) the Earth (ΔV=7.8 km/s)
c) Jupiter (if it had a solid surface! ΔV= 42 km/s)

Given: Each rocket stage provides ΔV = 3 km/s. Delta-V values to reach orbit are taken from the textbook (Wertz et al., Ch. 1, pp. 10--11).

Method: To find the minimum number of stages, divide the total required delta-V by 3 km/s and apply the ceiling function (round up) — you can't build a partial stage.

$$n = \left\lceil \frac{\Delta V_{\text{required}}}{\Delta V_{\text{per stage}}} \right\rceil$$

(a) Moon — ΔV = 1.7 km/s

$$n = \left\lceil \frac{1.7}{3.0} \right\rceil = \lceil 0.567 \rceil = 1$$

Check: 1 stage provides 1 × 3.0 = 3.0 km/s ≥ 1.7 km/s ✓

Answer: 1 stage

(b) Earth — ΔV = 7.8 km/s

$$n = \left\lceil \frac{7.8}{3.0} \right\rceil = \lceil 2.6 \rceil = 3$$

Note: 2 stages would only provide 2 × 3.0 = 6.0 km/s, which is less than the 7.8 km/s required, so 2 stages aren't enough.
Check: 3 stages provide 3 × 3.0 = 9.0 km/s ≥ 7.8 km/s ✓

Answer: 3 stages

(c) Jupiter — ΔV = 42.0 km/s

$$n = \left\lceil \frac{42.0}{3.0} \right\rceil = \lceil 14.0 \rceil = 14$$

This is an exact division — 14 × 3.0 = 42.0 km/s exactly. The ceiling of an integer is just that integer.

Answer: 14 stages

Summary

Body	Required ΔV	Calculation	Stages Needed
Moon	1.7 km/s	$\lceil 1.7/3.0 \rceil = \lceil 0.567 \rceil$	**1 stage**
Earth	7.8 km/s	$\lceil 7.8/3.0 \rceil = \lceil 2.6 \rceil$	**3 stages**
Jupiter	42.0 km/s	$\lceil 42.0/3.0 \rceil = \lceil 14.0 \rceil$	**14 stages**

Problem 2 (30 points)

Assume that the mass of each rocket stage is equal to the mass of all the stages above it plus the 100 kg payload. Based on the number of stages determined in problem #1, if you wanted to put a 100 kg payload into orbit around the Moon, calculate the total mass of the rocket (all stages and the 100 kg payload) lifting off from the Moon. Answer the same question for lifting off from the Earth and likewise for Jupiter. Round up the values you determine since stages are not fractional.

Given: Payload mass m_p = 100 kg. The mass rule from the problem statement: each stage's mass equals the mass of everything above it (all higher stages + payload).

Stage numbering convention (per the problem): Stage 1 = bottom (largest, fires first at liftoff), Stage n = top (smallest).

Construction approach: I work from the top stage downward, since each stage's mass depends on what's sitting above it. At each step, the new stage mass equals the cumulative mass above, which means the running total doubles every time a stage is added.

(a) Moon — 1 stage

Only one stage is needed. The only thing above Stage 1 is the payload.

- Mass above Stage 1 = payload = 100 kg
- Stage 1 mass = mass above = **100 kg**
- Total = payload + Stage 1 = 100 + 100 = **200 kg**

Component	Mass (kg)
Stage 1 (bottom)	100
Payload	100
Total	**200 kg**

(b) Earth — 3 stages

Building from the top stage downward:

Stage 3 (top): Only the payload is above it.

- Mass above = 100 kg
- Stage 3 mass = 100 kg
- Cumulative total so far = 100 + 100 = 200 kg

Stage 2: Everything above is Stage 3 + payload = 200 kg.

- Mass above = 200 kg
- Stage 2 mass = 200 kg
- Cumulative total so far = 200 + 200 = 400 kg

Stage 1 (bottom): Everything above is Stage 3 + Stage 2 + payload = 400 kg.

- Mass above = 400 kg
- Stage 1 mass = 400 kg
- Cumulative total = 400 + 400 = 800 kg

Component	Mass (kg)
Stage 1 (bottom)	400
Stage 2	200
Stage 3 (top)	100
Payload	100
Total	**800 kg**

Verification: 100 + 100 + 200 + 400 = **800 kg** ✓

(c) Jupiter — 14 stages

The same doubling pattern continues for 14 stages. Each stage's mass is double the one above it, starting from the top stage at 100 kg:

Stage (top to bottom)	Mass above (kg)	Stage mass (kg)	Cumulative total (kg)
Stage 14 (top)	100	100	200
Stage 13	200	200	400
Stage 12	400	400	800
Stage 11	800	800	1,600
Stage 10	1,600	1,600	3,200
Stage 9	3,200	3,200	6,400
Stage 8	6,400	6,400	12,800
Stage 7	12,800	12,800	25,600
Stage 6	25,600	25,600	51,200
Stage 5	51,200	51,200	102,400
Stage 4	102,400	102,400	204,800
Stage 3	204,800	204,800	409,600
Stage 2	409,600	409,600	819,200
Stage 1 (bottom)	819,200	819,200	1,638,400

Computing 2^{14}: $2^{10} \times 2^4 = 1{,}024 \times 16 = 16{,}384$

$$M_{\text{total}} = 100 \times 16{,}384 = \mathbf{1{,}638{,}400\ kg}$$

That's roughly **1,638 metric tons** — in the same ballpark as a Saturn V. It really drives home why reaching orbit from a high-gravity body like Jupiter would be extraordinarily difficult.

Problem 3 (5 points)

Based on the above problems, what would be a simple mathematical relationship you could use for an “n” stage rocket and 100 kg payload?

General Mathematical Relationship

For an n-stage rocket with payload mass m_p, where each stage's mass equals the total mass above it plus the payload:

$$\boxed{M_{\text{total}} = m_p \times 2^n}$$

Derivation

Each stage's mass equals the cumulative mass above it. Starting from the top:

- Top stage mass = m_p (only payload above)
- Next stage mass = $m_p + m_p = 2m_p$ (top stage + payload above)
- Next stage mass = $2m_p + m_p + m_p = 4m_p$ (cumulative above)
- Pattern: stage masses are $m_p,\ 2m_p,\ 4m_p,\ \ldots,\ 2^{n-1}m_p$

The individual stage masses form a geometric series with first term m_p and common ratio 2:

$$\text{Sum of all stage masses} = m_p + 2m_p + 4m_p + \cdots + 2^{n-1}m_p = m_p \sum_{k=0}^{n-1} 2^k = m_p(2^n - 1)$$

Adding the payload:

$$M_{\text{total}} = m_p + m_p(2^n - 1) = m_p \cdot 2^n$$

For the specific case of $m_p = 100$ kg:

$$M_{\text{total}} = 100 \times 2^n \quad \text{kg}$$

Verification

Checking the formula against our computed values:

Body	n	100×2^n	Computed Total	Match?
Moon	1	200	200	Yes
Earth	3	800	800	Yes
Jupiter	14	1,638,400	1,638,400	Yes

Limiting cases:

- $n = 0$ (no stages, payload only): $100 \times 2^0 = 100$ kg ✓
- $n = 1$: $100 \times 2^1 = 200$ kg — matches the Moon result ✓
- Each additional stage doubles the total, confirming the 2^n relationship ✓

Appendix A: Python Code

The following Python script implements the calculations above. It computes stage counts and masses both iteratively (stage by stage) and via the closed-form formula, then verifies they match. A log-scale bar chart visualizes the exponential growth in launch mass.

```
"""
SPCE 5045 Homework #1 -- Multi-Stage Rocket Mass Calculator
============================================================
Solves the three-part staging problem:
  1. Number of stages needed given delta-V per stage and total delta-V
  2. Total rocket mass given the equal-mass-above rule
  3. General closed-form relationship

Textbook: Wertz, Everett, Puschell, "Space Mission Engineering: The New SMAD"
          Microcosm Press, 2011. Chapter 1, pp. 10-11.
"""

import math
from typing import List, Tuple

import matplotlib.pyplot as plt


# ---------------------------------------------------------------------------
# Problem parameters
# ---------------------------------------------------------------------------
DELTA_V_PER_STAGE: float = 3.0          # km/s per stage
PAYLOAD_MASS: float = 100.0             # kg

# Target delta-V values (km/s) and labels
BODIES: List[Tuple[str, float]] = [
    ("Moon",    1.7),
    ("Earth",   7.8),
    ("Jupiter", 42.0),
]


# ---------------------------------------------------------------------------
# Helper functions
# ---------------------------------------------------------------------------
def stages_required(delta_v_total: float, delta_v_per_stage: float) -> int:
    """Return the minimum integer number of stages to reach delta_v_total.

    Each stage contributes exactly delta_v_per_stage km/s.
    We round UP because partial stages do not exist.
    """
    return math.ceil(delta_v_total / delta_v_per_stage)


def build_stage_masses(n_stages: int, payload_kg: float) -> List[float]:
    """Compute individual stage masses from top stage to bottom stage.

    Rule: mass of each stage = (mass of all stages above it) + payload.
    Construction proceeds from the topmost (smallest) stage downward.

    Returns a list where index 0 = topmost stage, index -1 = bottom stage.
    """
    stages: List[float] = []
    cumulative_above: float = payload_kg  # everything the top stage must support

    for _ in range(n_stages):
        stage_mass = cumulative_above      # this stage's mass = everything above
        stages.append(stage_mass)
        cumulative_above += stage_mass     # update cumulative for next stage down

    return stages


def total_rocket_mass(n_stages: int, payload_kg: float) -> float:
```

```
    """Total mass = payload + sum of all stage masses (iterative)."""
    stage_masses = build_stage_masses(n_stages, payload_kg)
    return payload_kg + sum(stage_masses)


def total_mass_closed_form(n_stages: int, payload_kg: float) -> float:
    """Closed-form: Total mass = payload * 2^n."""
    return payload_kg * (2 ** n_stages)


# ---------------------------------------------------------------------------
# Main computation
# ---------------------------------------------------------------------------
def main() -> None:
    print("=" * 65)
    print("SPCE 5045 Homework #1 -- Multi-Stage Rocket Analysis")
    print("=" * 65)

    # ------ Problem 1: Number of stages ------
    print("\n--- Problem 1: Number of Stages Required ---\n")
    print(f"  Delta-V per stage: {DELTA_V_PER_STAGE} km/s\n")

    results = []
    for body, dv in BODIES:
        n = stages_required(dv, DELTA_V_PER_STAGE)
        print(f"  {body:>8s}:  delta-V = {dv:5.1f} km/s  -->  "
              f"ceil({dv}/{DELTA_V_PER_STAGE}) = {n} stage(s)")
        results.append((body, dv, n))

    # ------ Problem 2: Total rocket mass ------
    print("\n--- Problem 2: Total Rocket Mass (100 kg payload) ---\n")

    mass_results = []
    for body, dv, n in results:
        # Iterative calculation
        stage_masses = build_stage_masses(n, PAYLOAD_MASS)
        total_iter = PAYLOAD_MASS + sum(stage_masses)

        # Closed-form verification
        total_cf = total_mass_closed_form(n, PAYLOAD_MASS)

        print(f"  {body:>8s}:  {n} stage(s)")
        # Print individual stage masses (bottom = stage 1, top = stage n)
        for i, sm in enumerate(stage_masses):
            stage_num = n - i  # convert top-first index to conventional numbering
            label = "(bottom, 1st stage)" if stage_num == 1 else \
                    "(top)"              if stage_num == n else ""
            print(f"    Stage {stage_num}: {sm:>14,.0f} kg  {label}")
        print(f"    Payload:  {PAYLOAD_MASS:>14,.0f} kg")
        print(f"    TOTAL:    {total_iter:>14,.0f} kg")
        print(f"    Verify (closed-form): {total_cf:,.0f} kg  "
              f"[match: {math.isclose(total_iter, total_cf)}]")
        print()
        mass_results.append((body, n, total_iter))

    # ------ Problem 3: General formula ------
    print("--- Problem 3: General Relationship ---\n")
    print("  For n stages and payload mass m_p:")
    print("    Total_mass = m_p * 2^n")
    print()
    print("  Derivation sketch:")
    print("    Each stage's mass equals the cumulative mass above it.")
    print("    Adding a stage doubles the cumulative total.")
    print("    After n doublings starting from m_p: Total = m_p * 2^n")
```

```
    # ------ Verification: dimensional / limiting cases ------
    print("\n--- Verification ---\n")
    print("  Limiting case n=0 (no stages, payload only):")
    print(f"    100 * 2^0 = {total_mass_closed_form(0, 100):.0f} kg  [correct]")
    print("  Limiting case n=1:")
    print(f"    100 * 2^1 = {total_mass_closed_form(1, 100):.0f} kg")
    print("    Stage mass = payload = 100 kg, total = 200 kg  [correct]")
    print("  Growth check: each additional stage doubles total mass.")
    for n_test in range(1, 5):
        ratio = total_mass_closed_form(n_test, 100) / total_mass_closed_form(n_test - 1, 100)
        print(f"    n={n_test}: total = {total_mass_closed_form(n_test, 100):,.0f} kg, "
              f"ratio to n={n_test-1}: {ratio:.1f}x")

    # ------ Plot ------
    fig, ax = plt.subplots(figsize=(8, 5))
    names = [r[0] for r in mass_results]
    masses = [r[2] for r in mass_results]
    n_stages_list = [r[1] for r in mass_results]

    bars = ax.bar(names, masses, color=["#7eb0d5", "#b2e061", "#fd7f6f"],
                  edgecolor="black", linewidth=0.8)

    ax.set_yscale("log")
    ax.set_ylabel("Total Rocket Mass (kg)", fontsize=12)
    ax.set_title("Total Launch Mass to Reach Orbit\n"
                 "(100 kg payload, 3 km/s per stage)", fontsize=13)

    for bar, n, m in zip(bars, n_stages_list, masses):
        ax.text(bar.get_x() + bar.get_width() / 2, m * 1.4,
                f"{m:,.0f} kg\n({n} stages)",
                ha="center", va="bottom", fontsize=10, fontweight="bold")

    ax.set_ylim(top=max(masses) * 8)
    plt.tight_layout()
    plt.savefig("hw1_staging_mass.png", dpi=150)
    print("\n  Plot saved to hw1_staging_mass.png")
    plt.show()


if __name__ == "__main__":
    main()
```

Appendix B: Program Output (abbreviated)

```
============================================================
SPCE 5045 Homework #1 -- Multi-Stage Rocket Analysis
============================================================

--- Problem 1: Number of Stages Required ---

  Delta-V per stage: 3.0 km/s

      Moon:  delta-V =   1.7 km/s  -->  ceil(1.7/3.0) = 1 stage(s)
     Earth:  delta-V =   7.8 km/s  -->  ceil(7.8/3.0) = 3 stage(s)
   Jupiter:  delta-V =  42.0 km/s  -->  ceil(42.0/3.0) = 14 stage(s)

--- Problem 2: Total Rocket Mass (100 kg payload) ---

      Moon:  1 stage(s)
    Stage 1:            100 kg  (bottom, 1st stage)
    Payload:            100 kg
    TOTAL:              200 kg
    Verify (closed-form): 200 kg  [match: True]

     Earth:  3 stage(s)
    Stage 3:            100 kg  (top)
    Stage 2:            200 kg
    Stage 1:            400 kg  (bottom, 1st stage)
    Payload:            100 kg
    TOTAL:              800 kg
    Verify (closed-form): 800 kg  [match: True]

   Jupiter:  14 stage(s)
    Stage 14:           100 kg  (top)
    Stage 13:           200 kg
    ...
    Stage 1:        819,200 kg  (bottom, 1st stage)
    Payload:            100 kg
    TOTAL:        1,638,400 kg
    Verify (closed-form): 1,638,400 kg  [match: True]

--- Problem 3: General Relationship ---

  For n stages and payload mass m_p:
    Total_mass = m_p * 2^n
```