PHYS 256 — Final Project

Student Name

Instructions

The Artemis II mission provides the inspiration for your final project: simulate a spacecraft that launches from Earth, travels through the Earth–Moon system, flies around the Moon, and eventually returns to Earth. Your job in this first stage is to *build and test the numerical simulator*. You are **not** expected to find a successful, fuel-optimal lunar mission yet. In fact, several of the example flight plans you will investigate are intentionally poor!

Similar to the midterm exams, the final exam has a take-home component and a later, in-class component. Your work here should result in a reliable, well-documented set of Python functions that can be reused for unfamiliar problems. The overall mission-design goal is to satisfy flight constraints while using as little propellant as possible; **this week you will use a fixed spacecraft mass and will not yet optimize fuel consumption**. The in-class portion will provide its own detailed instructions.

Keep your code *concise* and *modular*. Your entire simulation code for running `solve_ivp` should be no longer than 100 lines. Your code for plotting and testing might be longer than this.

You are encouraged and expected to develop this simulation package using generative AI. As always, make sure you understand, test, and verify the code being generated. You will be expected to use and modify it during the in-class final exam.

Deadlines and schedule

- Mon Nov 23: Final project assignment posted
- Wed Dec 2: First deliverable due at 11:59pm (problems 1-4 below)
- Tues Dec 8: Second deliverable due at 11:59pm (requirements will be specified later)
- Wed Dec 9 or Fri Dec 11: In-class final exam (check your section for date and time)

Physical model and assumptions

- The system contains three bodies: Earth, Moon, and spacecraft. Neglect the Sun and all other bodies.
- All motion occurs in a 2D plane. Use a Cartesian inertial coordinate system whose origin is the Earth–Moon center of mass (barycenter).
- Earth and Moon interact gravitationally with one another. Their orbit should emerge from numerically solving their equations of motion; do not fix either body in place or prescribe a circular orbit. By convention, the Moon should orbit counterclockwise in the plane.
- The spacecraft experiences the gravity of both Earth and Moon, but has negligible gravitational influence on either.
- The spacecraft is a sphere represented by an effective drag area A and a constant drag coefficient C_D; Earth and Moon are assumed to be spherical. Neglect lunar atmosphere.
- The spacecraft mass m_S is constant throughout this take-home assignment. Engine thrust is a *force* (Newtons), not a prescribed acceleration.
- The engine may fire at constant thrust magnitude and direction over prescribed intervals. Between burns the engine is off.

State vector and equations of motion

Use `scipy.integrate.solve_ivp` to solve the coupled ODEs of a **single 12-component state vector**, ordered as

$$\boxed{\boldsymbol{Y} = (\boldsymbol{r}_E, \boldsymbol{v}_E, \boldsymbol{r}_M, \boldsymbol{v}_M, \boldsymbol{r}_S, \boldsymbol{v}_S).} \tag{1}$$

Here E, M, S label Earth, Moon, and spacecraft. For each body $i \in \{E, M, S\}$, define the two-dimensional position $\boldsymbol{r}_i = (x_i, y_i)$, velocity $\boldsymbol{v}_i = (v_{x,i}, v_{y,i})$, and acceleration $\boldsymbol{a}_i = \dot{\boldsymbol{v}}_i$. Each vector contributes two scalar components, so $\boldsymbol{Y}$ contains 12 components in total.

The equations of motion are

$$\dot{\boldsymbol{r}}_E = \boldsymbol{v}_E, \qquad \dot{\boldsymbol{v}}_E = GM_M \frac{\boldsymbol{r}_M - \boldsymbol{r}_E}{\|\boldsymbol{r}_M - \boldsymbol{r}_E\|^3}, \tag{2}$$

$$\dot{\boldsymbol{r}}_M = \boldsymbol{v}_M, \qquad \dot{\boldsymbol{v}}_M = GM_E \frac{\boldsymbol{r}_E - \boldsymbol{r}_M}{\|\boldsymbol{r}_E - \boldsymbol{r}_M\|^3}, \tag{3}$$

$$\dot{\boldsymbol{r}}_S = \boldsymbol{v}_S, \qquad \dot{\boldsymbol{v}}_S = GM_E \frac{\boldsymbol{r}_E - \boldsymbol{r}_S}{\|\boldsymbol{r}_E - \boldsymbol{r}_S\|^3} + GM_M \frac{\boldsymbol{r}_M - \boldsymbol{r}_S}{\|\boldsymbol{r}_M - \boldsymbol{r}_S\|^3} + \frac{\boldsymbol{F}_T(t)}{m_S} + \boldsymbol{a}_D. \tag{4}$$

where M_E, M_M, m_S are the masses of each body and G is the universal gravitational constant. The thrust force is $\boldsymbol{F}_T(t)$ and will be discussed more below. The drag acceleration is

$$\boldsymbol{a}_D = -\frac{\rho(h) C_D A}{2 m_S} \|\boldsymbol{v}_{\text{rel}}\| \, \boldsymbol{v}_{\text{rel}}. \tag{5}$$

where the spacecraft's velocity relative to Earth's atmosphere is $\boldsymbol{v}_{\text{rel}} = \boldsymbol{v}_S - \boldsymbol{v}_E$. Atmospheric density falls exponentially with altitude:

$$\rho(h) = \rho_0 \exp[-h/H], \qquad h \geq 0 \tag{6}$$

where

$$h = \|\boldsymbol{r}_S - \boldsymbol{r}_E\| - R_E \tag{7}$$

is the spacecraft's height above the Earth's surface, with R_E the radius of the Earth, and ρ_0, H defined in the table below. Assume that the Earth's atmosphere does not rotate (i.e. it is fixed in the inertial coordinate system).

The flight is considered terminated once the rocket reaches the Earth's surface $h = 0$. Use SI units inside all physics functions. You may convert to kilometers, hours, or days for plotting and reporting.

Thrust schedule

Represent a flight plan as a *list of burns*. Each burn is described by a start time t_i, a duration Δt_i, a nonnegative thrust magnitude T_i, and a fixed direction angle θ_i measured counterclockwise from the inertial $+x$ axis. During burn i, the force imparted on the spacecraft is

$$\boldsymbol{F}_T(t) = T_i \begin{pmatrix} \cos\theta_i \\ \sin\theta_i \end{pmatrix}, \qquad t_i \leq t < t_i + \Delta t_i; \tag{8}$$

outside the burns, $\boldsymbol{F}_T = \boldsymbol{0}$. For simplicity, **burn intervals must not overlap**. Never approximate a burn by instantaneously changing the rocket's velocity.

Physical constants and baseline configuration

Use the following SI constants throughout. These are contained in the `config.py` file on the course website.

Symbol	Meaning	Value
G	Gravitational constant	$6.67430 \times 10^{-11}\ \mathrm{m^3\,kg^{-1}\,s^{-2}}$
M_E	Earth mass	5.9722×10^{24} kg
M_M	Moon mass	7.3477×10^{22} kg
R_E	Earth radius	6.3710×10^{6} m
R_M	Moon radius	1.7374×10^{6} m
ρ_0	Sea-level air density	$1.225\ \mathrm{kg\,m^{-3}}$
H	Atmospheric scale height	8.50×10^{3} m
m_S	Spacecraft mass	2.6×10^{6} kg
A	Effective spacecraft cross-section	$77.0\ \mathrm{m^2}$
C_D	Spacecraft drag coefficient	1.0

Earth–Moon initial conditions. Set $t = 0$ at the epoch 2026-01-01 00:00:00 UTC. Choose fixed inertial axes so that the initial Earth-to-Moon direction is $+x$. The initial relative position and velocity are

$$\boldsymbol{r}_{\mathrm{rel}} = \boldsymbol{r}_M(0) - \boldsymbol{r}_E(0) = (361024834.7968357,\ 0)\ \mathrm{m},$$
$$\boldsymbol{v}_{\mathrm{rel}} = \boldsymbol{v}_M(0) - \boldsymbol{v}_E(0) = (-16.992366011,\ 1088.819191257)\ \mathrm{m/s}.$$

Place the barycenter (center of mass) at rest at the origin by using the initial conditions

$$\boldsymbol{r}_E(0) = -\frac{M_M}{M_E + M_M}\boldsymbol{r}_{\mathrm{rel}}, \qquad \boldsymbol{r}_M(0) = \frac{M_E}{M_E + M_M}\boldsymbol{r}_{\mathrm{rel}},$$
$$\boldsymbol{v}_E(0) = -\frac{M_M}{M_E + M_M}\boldsymbol{v}_{\mathrm{rel}}, \qquad \boldsymbol{v}_M(0) = \frac{M_E}{M_E + M_M}\boldsymbol{v}_{\mathrm{rel}}.$$

The initial Earth–Moon state for verification *must* be taken from this data rather than invented. These initial conditions are contained in the `config.py` file on the course website.

Spacecraft initial conditions. At $t = 0$, the spacecraft begins 10 m above the Earth's surface on the $+x$ inertial axis (right-hand side). Earth's rotation actually helps the spacecraft achieve orbit (this is one of the reasons why we launch rockets from Florida!). To account for this we assume the spacecraft begins with an initial velocity of 465.0 m/s in the $+y$ direction (which is approximately the rotational velocity of the Earth's surface at the equator). Keep in mind that for this simulation the Earth and its atmosphere are not rotating, this initial condition is a bit of a 'kludge.'

Problem 1. Earth–Moon orbital dynamics (25 points)

1a. Implement orbital equations (10 pts). Implement a function `dydt(t, state, flight_plan, params)` returning the derivatives of all 12 state variables, where `params` contains all numerical constant in the table above. For the moment, neglect the motion of the spacecraft and assume `flight_plan` is an empty array. Use this function to build a `solve_ivp` simulation of the coupled Earth–Moon motion using the initial conditions supplied above.

1b. Explore an eccentric orbit (7 pts). Plot the Earth and Moon trajectories in inertial coordinates, showing the barycenter (center of mass)

$$\mathbf{r}_{\mathrm{CM}} = \frac{M_E \mathbf{r}_E + M_M \mathbf{r}_M}{M_E + M_M} \tag{9}$$

and marking the initial positions of Earth and Moon. Also plot Earth–Moon distance versus time for at least one full lunar orbital period. Explain how this plot demonstrates that the orbit is not circular. Also plot the Earth–Moon distance as a function of angle θ measured counterclockwise from the x-axis to ensure that the orbit is closed. Compute the eccentricity e and period T of the lunar orbit. The eccentricity can be found via the formula

$$e = \frac{r_a - r_p}{r_a + r_p} \tag{10}$$

where r_a is the *apogee* (the Moon's farthest distance from the center of mass) and r_p is the *perigee* (the Moon's closest distance from the center of mass).

1c. Verify physical and numerical consistency (8 pts). Plot the motion of the center of mass $\mathbf{r}_{\mathrm{CM}}$ and verify that it is approximately zero at all times to within numerical tolerance. Calculate and plot the total mechanical energy and the total angular momentum (about the barycenter) of the isolated Earth–Moon system. Report their maximum relative deviations over the simulation. Repeat one simulation using tighter tolerances and/or smaller maximum steps, and explain whether your results converge.

For this problem the gravitational potential energy is $U = -GM_E M_M / \|\boldsymbol{r}_E - \boldsymbol{r}_M\|$. The Earth–Moon angular momentum is the sum of the z-components $L_z = \sum_i M_i (x_i v_{y,i} - y_i v_{x,i})$.

Problem 2. Spacecraft dynamics (35 points)

2a. Spacecraft dynamics (12 pts). Incorporate the spacecraft motion into the function `dydt(...)`. Include gravitational forces from both Earth and Moon, atmospheric drag near Earth, and time-dependent continuous thrust on the rocket. Explain in a few sentences why the spacecraft's gravitational effect on Earth and Moon can be neglected.

2b. Flight plans (8 pts). Implement a function `thrust_force(t, state, flight_plan)` that returns the rocket thrust vector at any requested time `t`. Test it at times just before, during, and after each burn. Include at least one no-thrust scenario as a baseline. Each flight plan should be stored in an array `flight_plan`, where each element is an array of four elements `[start, duration, thrust, angle]`.

2c. Simulate the pre-baked flight plans (10 pts). Run all of the provided flight plans contained in `flight_plans.py` on the course website with the same simulator and physical constants. These plans will include some deliberately poor choices such as a crash landing. Stop a simulated flight if the rocket reaches Earth's surface after takeoff or intersects the Moon's surface. Record and describe what happens in each case; a crash or failure to reach the Moon is an acceptable and potentially expected result.

2d. Experiment (5 pts). Design at least two flight plans of your own by varying the start time, direction, strength, or duration of thrust. Explain what you changed, what you predicted would happen, what actually happened, and what you learned. A successful lunar flyby is *not* required (yet).

Problem 3. Visualize and interpret trajectories (20 points)

Generate clear, labeled figures for all five pre-baked flight plans and at least one of your own plans. Include:

3a. Overview (5 pts). A 2D plot showing Earth, Moon, spacecraft, and their paths in the inertial barycentric frame. Mark the launch point, important burn intervals, and final position. Use a legend and equal axis scaling for geometry.

3b. Launch detail (5 pts). A zoomed view of the spacecraft near Earth with Earth's surface indicated, and a plot of rocket altitude above Earth versus time.

3c. Dynamics (5 pts). Plot spacecraft speed relative to Earth, the magnitude of the thrust acceleration, and the magnitude of the atmospheric drag acceleration versus time. Label axes with units and indicate burn windows.

3d. Rotating Barycentric Trajectory (5 pts). Plot the trajectory of your spacecraft in the *barycentric rotating frame*, i.e. a coordinate system that rotates with the Earth–Moon axis so that the origin is at the center of mass and the Earth–Moon axis is always fixed in this frame.

You may combine related variables into a single figure with clearly labeled panels. The purpose is to use visualization to diagnose the numerical model, not merely to produce attractive images.

Problem 4. Orbital Insertion (20 points)

Design a flight plan that brings the spacecraft into a stable, circular 'parking' orbit around the Earth. For example, the space shuttle often 'parked' at an altitude of roughly $h = 250$ km, with an orbital velocity of $v = \sqrt{GM_E/(R_E + h)}$ in the Earth's reference frame (why is this the requisite velocity for a stable circular orbit?)

Implementation and submission requirements

- Use Python, NumPy, Matplotlib, and SciPy's `solve_ivp`. You may use the numerical methods and programming tools developed during this course. Do not use external astrodynamics simulation packages or ready-made mission optimizers.
- Write modular code. In particular, keep gravitational accelerations, atmospheric density/drag, thrust scheduling, the ODE right-hand side, flight simulation, and diagnostics as distinct, identifiable functions. Document input/output units.
- Make simulation settings, initial conditions, and physical parameters easy to change; do not hard-code a single mission into the ODE function.
- Do not allow the model to integrate through the interior of Earth or Moon. Use an appropriate stopping event or an equivalent reliable stopping procedure. If using `solve_ivp` events, remember that events detect crossings and may miss them if steps are too large.
- Include a brief numerical-methods discussion: tolerance choices, step-size sensitivity, handling of abrupt thrust changes, and at least one physical consistency check.
- Submit **(i)** executable Python code and **(ii)** a concise PDF report containing the required figures, tables, and discussion.