# ASEN 3502 Computational Methods
# Lab 3

Exact Due Date on Gradescope

**General Instructions**

1. Implement Euler's method and classical fourth-order Runge–Kutta yourself. Do not call a built-in ODE solver inside either implementation.
2. You may use any programming language, and you may use different languages for different parts. However, the primary goal of this lab is understanding, and *you should use Matlab unless every team member is enthusiastic about using something else.*
3. Adhere to the AI policy at asen3502.com/ai-policy.
4. During each lab session, each student in the group should spend some time as the "driver": the person typing things into the computer or the person typing equations in latex.



Hubble Space Telescope in Earth orbit. Credit: NASA.

In this lab, you will numerically propagate a spacecraft's motion around Earth. You will use your Euler implementation from Homework 4, implement and verify RK4, investigate how step size affects orbital



Compiled 2026-10-17 19:26:43

accuracy, and compare fixed-step propagation with an adaptive solver. In Week 2, you will use Newton's method from the previous lab to solve an orbital boundary value problem by shooting. The required challenge then adds the Moon's gravitational influence.

# Deliverables (at end of Week 2)

1. To the **Lab 3 Writeup** assignment on Gradescope: A **single pdf** containing all required items marked writeup.
   - Label each result with its task number. Clearly label plots and figures, use correct units, and identify methods and resolutions in legends or captions.
   - The writeup does *not* need narrative text unless requested. Include the derivations, plots, tables, and explanations specified in each task. Support accuracy claims with numerical evidence.
2. To the **Lab 3 Code** assignment on Gradescope: Your **source code** for all required items marked code.
   - Include the integrator and dynamics functions and the scripts that generate your results. Label the weekly tasks clearly, whether they are separate scripts or sections of one driver script.
   - Report your final three-body challenge score on Gradescope.
   - You will not be graded on the quantity of comments in your code. You must be able to explain your implementation and results.

# 1 Week 1: Orbit Propagation with Euler and RK4

Use the `euler`, `test_dyn`, and `test_ode` functions from Homework 4. Compare your individual implementations and choose one for your team. Move each function into its own file so that all of your scripts can call it.

## 1.1 Define the orbital dynamics

writeup code

Treat Earth as a point mass and neglect the spacecraft's contribution to the gravitational parameter. Use

$$\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r}, \qquad \mu = 3.986 \times 10^{14}\ \mathrm{m}^3/\mathrm{s}^2, \qquad \vec{x} = [r_x,\ r_y,\ r_z,\ \dot{r}_x,\ \dot{r}_y,\ \dot{r}_z]^T.$$

Using the acceleration equation above, write the equations as a first-order vector system $\dot{\vec{x}} = f(\vec{x})$. Implement the dynamics in a function `two_bp(t, x)` that returns the six-component state derivative. Retain the time argument even though this particular model does not depend explicitly on time.

## 1.2 Propagate a circular orbit

writeup

Choose a circular radius $R = 10^8$ m. Derive the natural circular speed $v_{\mathrm{circ}}$ and period $T$ by balancing centripetal and gravitational acceleration. Use initial conditions

$$\vec{r}_0 = [R, 0, 0]^T, \qquad \dot{\vec{r}}_0 = [0, v_{\mathrm{circ}}, 0]^T. \qquad T = 2\pi\sqrt{\frac{r^3}{\mu}}$$

Propagate over the requested interval $[0, T]$ with `euler` and $h = 10$ s. Plot the trajectory in the $x$–$y$ plane, and plot the specific orbital energy versus time:

$$\mathcal{E}(t) = \frac{1}{2}\|\dot{\vec{r}}(t)\|^2 - \frac{\mu}{\|\vec{r}(t)\|}.$$

Explain why the orbital energy is not constant, and why the trajectory differs from a circular orbit.



Compiled 2026-10-17 19:26:43

## 1.3 Implement RK4

code

Implement the four stage equations and final update for classical RK4 in

```
function [t, x] = rk4(dynamics, tspan, x0, h)
```

Use the same input and output conventions as your `euler` function.

## 1.4 Verify RK4

writeup code

Reuse the `test_ode` function from Homework 4. Run `test_ode(@rk4, 0.001, 1e-5)` and show that it completes without an error. The test function should accept the new integrator without changing the test dynamics or analytical reference.

## 1.5 Compare Euler and RK4

writeup

Propagate the circular orbit from Task 1.2 using both methods with $h = 100$ s over $[0, T]$. Plot both trajectories on the same axes. Describe the differences and relate them to the accuracy and computational cost of the two methods.

## 1.6 Measure position convergence

writeup

For RK4 applied to the same circular initial conditions, use

$$h = T/N, \qquad N = \{400, 800, 1600, 3200, 4000\}.$$

Propagate over $[0, T]$ and calculate the closure error

$$E_{\text{close}}(h) = \|\vec{r}_{\text{num}}(t_{\text{end}}) - \vec{r}_0\|,$$

where $t_{\text{end}}$ is the last time actually returned. Plot this error against $h$ on logarithmic axes and estimate the slope. Report the slope (convergence rate), and the error plot in your writeup. You can estimate the slope of a linear line using polyfit, with the following structure:

```
p = polyfit(x, y, 1);
```

## 1.7 Challenge Problem: Two-Body Dynamics

code

The challenge problem will build on your orbital model over both weeks of the lab. Your goal is to reproduce a reference trajectory by implementing the spacecraft dynamics and accurately integrating its motion.

Begin with the two-body dynamics in `two_bp(t, x)`, and the initial state

$$\vec{x}_0 = [0,\ 13262854,\ 0,\ -5482,\ 5245,\ 0]^T,$$

with positions in meters and velocities in meters per second. Propagate for 200 hours, converting the interval to seconds before calling your integrator.

Your score will be based on the number of hours accurately propagated, with a maximum of 200 hours for the two body model. **By the end of week 1 you should have a working implementation returning a score of 200.** After propagating your trajectory, call the evaluator using

```
[score, tRef, xRef] = evaluate(t, x, 1, 200,'your.email@something.com);
```

Here, `t` contains increasing sample times in seconds starting at zero, and each row of `x` contains the corresponding state $[r_x, r_y, r_z, v_x, v_y, v_z]$ in SI units. The third argument selects the model (1 for two-body, 2 for three-body, or 3 for the optional extension), and the fourth is the duration to evaluate in hours. Your trajectory must cover that duration, up to the model's score limit. On failure, `tRef` contains 11 reference times in seconds within one hour of the first failure, and `xRef` contains the corresponding rows of $[r_x, r_y, r_z]$ in meters. Both reference outputs are empty if all checked samples pass.

**For all challenge-problem results, you must use an integrator that you implement yourselves. Built-in integrators, including `ode45`, are not permitted for your trajectory propagation.** You may use your own RK4 implementation or implement another numerical integration method. To see how to get a higher score than 200, see 2.5.

# 2 Week 2: Boundary Value Problems and Shooting

## 2.1 Compare RK4 with an adaptive solver

writeup

Use `two_bp` to propagate the same initial conditions given in 1.7

a. Propagate using RK4 with $h = 100$ s and using `ode45` with `RelTol=1e-12` and `AbsTol=1e-14`.
b. Overlay the two trajectories by interpolating the adaptive solution's positions onto the RK4 time grid. Calculate the position-difference norm at each time. You can interpolate using the following call of `interp1`:

```
x_ode45 = interp1(t2, x2, t);
```

c. Plot the position difference and geocentric radius versus time on two separate subplots. Explain why adaptive stepping can be more efficient. Identify where the error is largest, and why that is the case.

## 2.2 Choose an initial guess

writeup

Find an initial velocity that takes a spacecraft from

$$\vec{r}_0 = [7000 \times 10^3,\ 0,\ 0]^T \text{ m} \quad \text{to} \quad \vec{r}_f = [0,\ 9000 \times 10^3,\ 0]^T \text{ m}$$

in $T = 2400$ s. Use `two_bp` and your RK4 function with $h = 10$ s. Adjust only the initial $x$- and $y$-velocities; keep $z_0 = v_{z0} = 0$. This approach is called *shooting*.

Choose an initial velocity guess and propagate to $T$. Write a plotting function that shows the trajectory, starting position, target, and actual ending position. Use the starter code `plot_shooting.m` to do this. Submit this plot and report your velocity guess and distance from the target. Justify your choice of initial velocity.

## 2.3 Apply one Newton correction

writeup code

Reuse your `newton_sys` and `numjac` functions from Lab 2, along with your linear solver from Lab 1, to solve for the initial velocity components.

Implement a function `shooting_residual` that integrates the trajectory for a trial initial velocity and returns the terminal position error:

$$\vec{F}(\dot{\vec{r}}_0) = \begin{bmatrix} x_f - x(T; \dot{\vec{r}}_0) \\ y_f - y(T; \dot{\vec{r}}_0) \end{bmatrix}.$$

As in Lab 2, fix the problem parameters with an anonymous function so that your solver receives a function of the unknowns only:

```
f = @(rdot0) shooting_residual(rdot0, r0, rf, T, @two_bp, h);
```

Use your `rk4` function with $h = 10$ s, and use `numjac` to define the Jacobian function handle `J` with a velocity perturbation of 1 m/s.

Adapt the stopping criterion in `newton_sys` so that `atol` applies to $\|\vec{F}\|_2$, rather than the change in the velocity guess. Use a position tolerance of 1 m.

Starting from your guess in Task 2.2, run a single iteration of `newton_sys`. Propagate the corrected initial velocity and plot the resulting trajectory. Does your second guess bring the endpoint closer to the target?

## 2.4 Implement iterative shooting

writeup

Use your adapted `newton_sys` with the same residual and Jacobian function handles to continue iterating from your corrected guess in Task 2.3.

Print the miss distance $\|\vec{F}\|_2$ at each iteration. Stop when it is below 1 m or after 20 further corrections.

Submit the final trajectory plot and report the initial velocity found, final miss distance, and number of additional corrections. State whether your solver converged.

## 2.5 Challenge Problem: Three-Body Dynamics

writeup code

Extend your challenge model to include the Moon's gravitational influence, producing an Earth–Moon–spacecraft three-body model. Use the same initial state given in Task 1.7.

Use an Earth-centered frame with nonrotating axes. For this simplified model, hold Earth fixed at the origin and neglect its acceleration due to the Moon. Prescribe the Moon's position as

$$\vec{r}_M(t) = R_M \begin{bmatrix} \cos(n_M t + \theta_0) \\ \sin(n_M t + \theta_0) \\ 0 \end{bmatrix}, \qquad n_M = \sqrt{\frac{\mu_E}{R_M^3}},$$

and acceleration due to the moon and earth as

$$\ddot{\vec{r}} = -\frac{\mu_E \vec{r}}{\|\vec{r}\|^3} + \mu_M \frac{\vec{r}_M - \vec{r}}{\|\vec{r}_M - \vec{r}\|^3}.$$

where

$$\mu_E = 3.986 \times 10^{14} \text{ m}^3/\text{s}^2, \quad \mu_M = 4.904 \times 10^{12} \text{ m}^3/\text{s}^2,$$
$$R_M = 384400 \times 10^3 \text{ m}, \qquad \theta_0 = 2.3243 \text{ rad}.$$

Implement `three_bp(t, x)` using Earth's central gravity and the Moon's direct gravitational acceleration of the spacecraft. Prescribe the Moon's motion using the expression above. For the 3-body model, you will be scored based on the number of hours accurately propagated. **You must report a score of 1000 on Gradescope to receive full credit for the challenge problem.**



Compiled 2026-10-17 19:26:43

As an optional extension, the reference trajectory includes a hidden perturbation in addition to Earth's and the Moon's gravity **after 1000 hours**. Use the reference positions returned by the evaluator near the point where your trajectory first exceeds the allowed error to investigate the discrepancy and develop an additional acceleration model. Keep the same initial state and continue using your own integrator. Your goal is to accurately propagate for as long as possible; the extension has no maximum score. Include a trajectory plot and your final score in the writeup.