Contents

1 Orbits **3**

1.1 Astronauts are *not* weightless 5

1.2 Geosynchronous Orbits 6

1.3 Escape velocity 7

2 Rocketry **9**

2.1 Types of rocket fuels 9

2.2 Tyranny of the rocket equation 10

2.3 Control in atmosphere 10

2.4 Control in space 13

2.5 Alternative propulsion 15

3 Simulation with Vectors **17**

3.1 Force, Acceleration, Velocity, and Position 17

3.2 Simulations and Step Size 18

3.3 Make a Text-based Simulation 18

3.4 Graph the Paths of the Moons 21

3.5 Conservation of Momentum 23

3.6 Animation 26

3.7 Challenge: The Three-Body Problem 30

4 Longitude and Latitude **35**

4.1 Longitude and Latitude 35

4.2 Nautical Mile 38

4.3 Haversine Formula *38*

5 Tides and Eclipses ***41***

5.1 Leap Years *41*

5.2 Phases of the Moon *42*

5.3 Eclipses *45*

5.4 The Far Side of the Moon *46*

5.5 Tides *48*

5.5.1 Computing the Forces *49*

5.5.2 Solar Tidal Forces *55*

A Answers to Exercises ***57***

Index 61

Chapter 1

Orbits

Gravity is the force that attracts two bodies toward each other. It is responsible for the behavior of orbital motion. Gravity is described by Newton's law of universal gravitation, which states that every point mass attracts every other point mass with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them. The formula for gravitational force is:

$$F_g = \frac{Gm_1m_2}{r^2}$$

Where F_g is the gravitational force between two objects, G is the gravitational constant, m_1 and m_2 are the masses of the objects, and r is the distance between the centers of the two objects. G is a constant with a value of approximately $6.674 \times 10^{-11}\,\mathrm{N\,m^2/kg^2}$.

If we think about acceleration due to gravity where one mass is significantly larger than the other, we can rewrite the equation as:

$$A_g = \frac{Gm_1m_2}{r^2m_2} = \frac{Gm_1}{r^2}$$

Where m_1 is the mass of the more massive object and m_2 is the mass of the significantly less massive object. This cancelling of the m_2 mass is why the acceleration due to gravity is independent of the mass of the object in free fall.

A satellite stays in orbit around the planet because the pull of the planet's gravity causes it to accelerate toward the center of the planet.

The satellite must be moving at a very particular speed to keep a constant distance from the planet — to travel in a circular orbit. If it is moving too slowly, it will get closer to the planet. If it is going too fast, it will get farther from the planet.

The radius of a satellite in a low orbit is typically about 2 million meters above the ground. At that distance, the acceleration due to gravity is more like $6.8m/s^2$, instead of the $9.8m/s^2$ that we experience on the surface of the planet.

How fast does the satellite need to be moving in a circle with a radius of 8.37 million meters to have an acceleration of $6.8m/s^2$? Real fast.

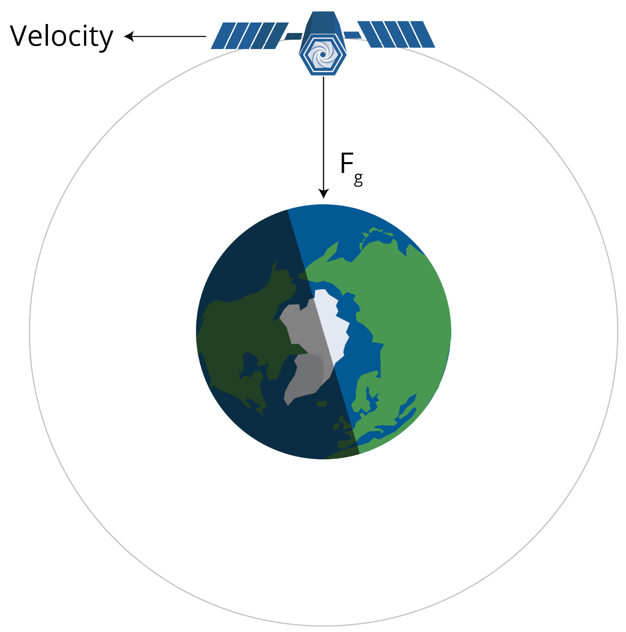


Figure 1.1: A satellite's centripetal force is gravity.

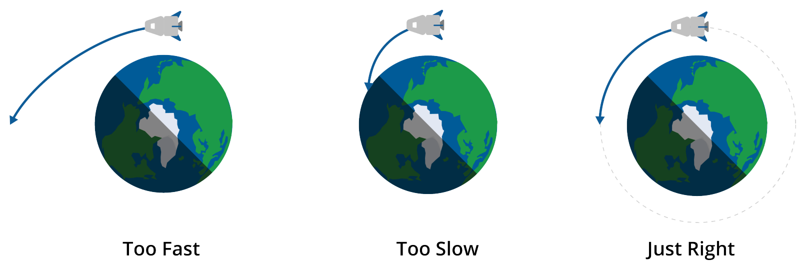


Figure 1.2: A diagram showing the required speeds for entering orbit.

Recall that the acceleration vector is

$$a = \frac{v^2}{r}$$

Thus the velocity v needs to be:

$$v = \sqrt{ar} = \sqrt{6.8(8.37\text{x}10^6)} = 7,544 \text{ m/s}$$

(That's 16,875 miles per hour.)

When a satellite falls out of orbit, it enters the atmosphere at that 7,544 m/s. The air rushing by generates so much friction that the satellite gets very, very hot, and usually disintegrates.

1.1 Astronauts are not weightless

Some people see astronauts floating inside an orbiting spacecraft and think there is no gravity: that the astronauts are so far away that the gravity of the planet doesn't affect them. This is incorrect. The gravity might be slightly less (Maybe 6 newtons per kg instead of 9.8 newtons per kg), but the weightless they experience is because they and the spacecraft is in free fall. They are just moving so fast (in a direction perpendicular to gravity) that they don't collide with the planet.

Exercise 1 Mars Orbit

Working Space

The radius of Mars is 3.39 million meters. The atmosphere goes up another 11 km. Let's say you want to put a satellite in a circular orbit around Mars with a radius of 3.4 million meters.

The acceleration due to gravity on the surface of Mars is 3.721m/s^2. We can safely assume that it is approximately the same 11 km above the surface.

How fast does the satellite need to be traveling in its orbit? How long will each orbit take?

Answer on Page 57

1.2 Geosynchronous Orbits

The planet earth rotates once a day. Satellites in low orbits circle the earth many times a day. Satellites in very high orbits circle less than once per day. There is a radius at which a satellite orbits exactly once per day. Satellites at this radius are known as “geosynchronous” or “geostationary”, because they are always directly over a place on the planet.

The radius of a circular geosynchronous orbit is 42.164 million meters. (About 36 km above the surface of the earth.)

A geosynchronous satellite travels at a speed of 3,070 m/s.

Geosynchronous satellites are used for the Global Positioning Satellite system, weather monitoring system, and communications system.

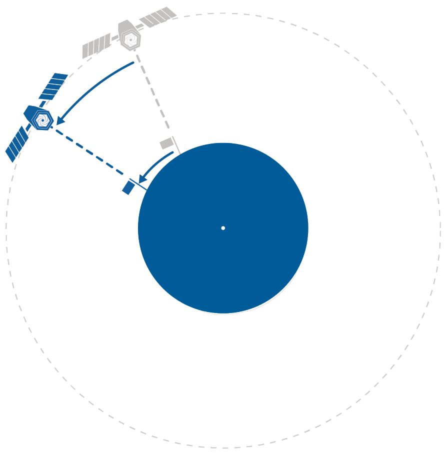


Figure 1.3: A satellite in geosynchronous orbit.

1.3 Escape velocity

FIXME: Add text for escape velocity

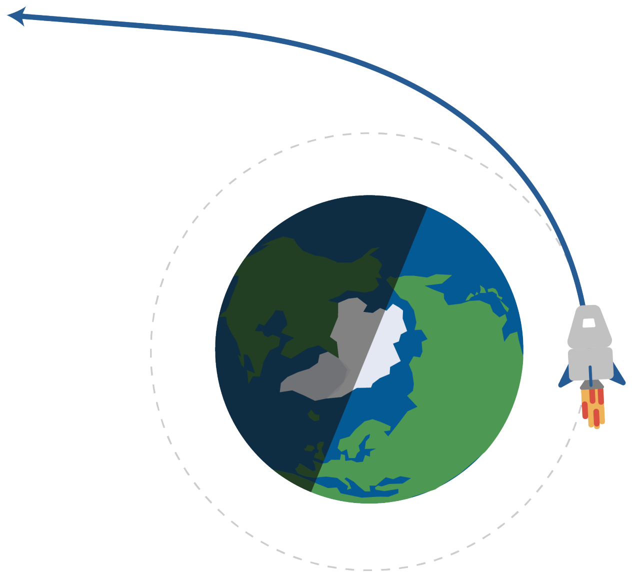


Figure 1.4: A satellite can reach a speed at which it ”escapes” earth’s orbit (centripetal force).

Chapter 2
Rocketry

Rockets propel hot gases, which recreates an equal and opposite reaction that pushes it forwards, even in a vacuum.

Even without anything to push against, the rocket can still move forward thanks to Newton's Third Law.

Imagine a spacecraft with a bowling ball attached to the back. If that spacecraft exerts a force to throw the bowling ball backwards, the ball will exert a force on the ship, moving it forwards.

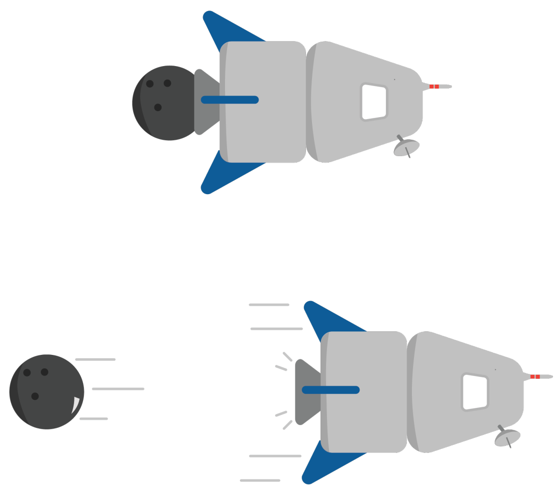


Figure 2.1: Newton's Third Law in action .

Instead of a bowling ball, real-life rockets usually "throw" particles of hot gas at very high speeds. Rockets carry their own oxidizer to provide oxygen to allow fuel to burn.

2.1 Types of rocket fuels

There are two main types of chemical rockets.

One type is a *Solid Fuel Rocket*, which ignites a solid fuel-oxidizer mix. Once the solid fuel is ignited, it can't be stopped until all of the fuel is exhausted.

The other main type of chemical rocket is called a *Liquid Fuel Rocket*. Liquid fuel rockets

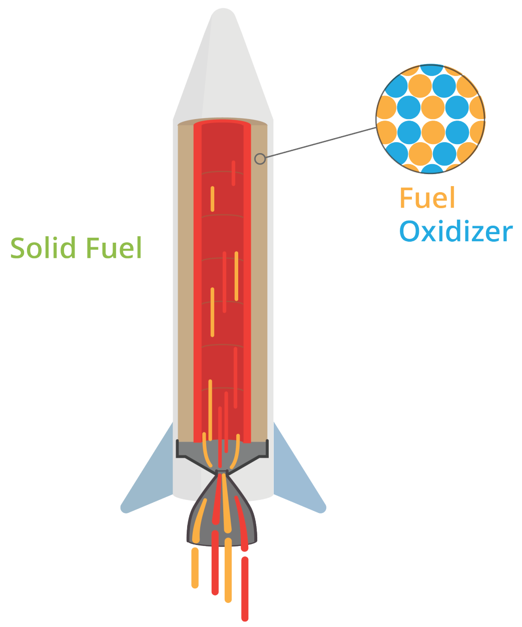


Figure 2.2: A solid fuel-oxidizer rocket.

contain separate tanks for liquid fuel and liquid oxygen. Fuel pumps bring them both to a combustion chamber where they ignite and exit the rocket. Most liquid fuel engines can control their thrust.

2.2 Tyranny of the rocket equation

Chemical rockets can only burn the fuel that they bring with them. However, the more fuel you carry, the heavier the vehicle will be.

One way to help reduce this weight is by using *staging*. Staging allows rockets to drop unnecessary structural mass once they've used up a certain amount of fuel.

2.3 Control in atmosphere

There are several common ways that engineers have managed to control rockets' direction in the atmosphere. Usually, on-board sensors detect the orientation of the rocket, and can automatically adjust these controls to keep the rocket going the correct direction.

One method is using *movable fins*. The fins work similarly to control surfaces that we covered in the airplanes chapter.

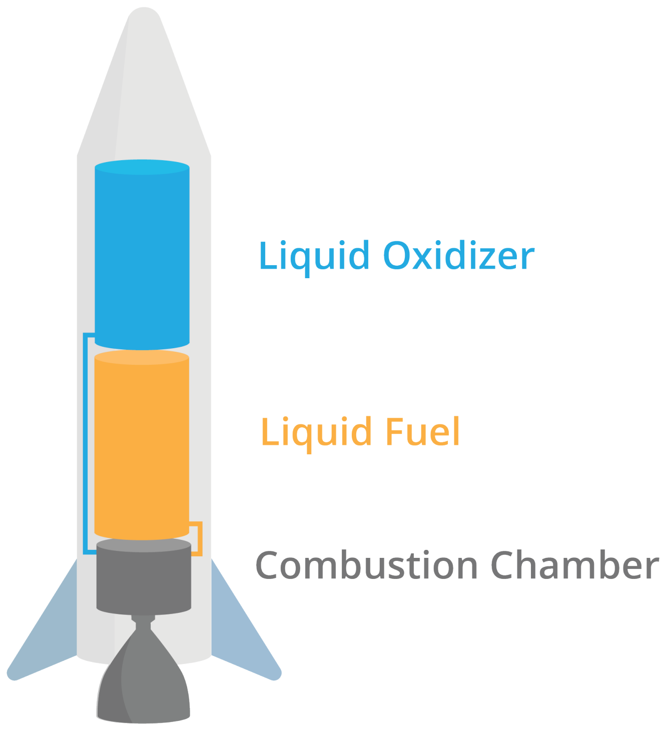


Figure 2.3: A liquid chemical phase.

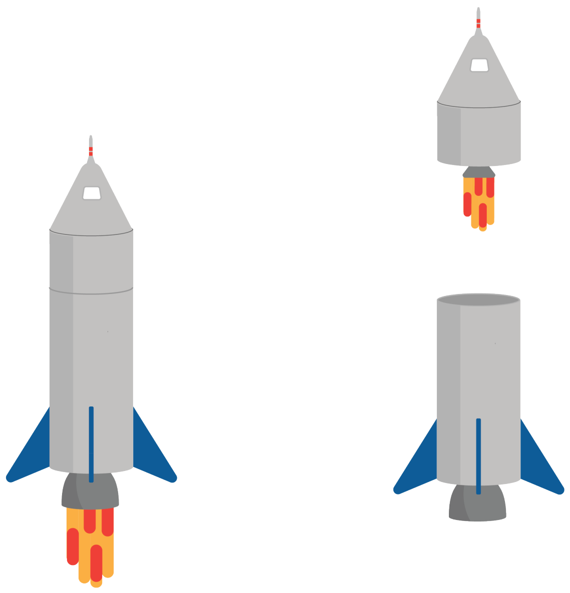


Figure 2.4: A dual stage rocket eliminates one of its stages throughout flight.

Another method of control uses a *gimbaled engine*. A gimbaled engine is an engine that can pivot in a certain direction. By pivoting the engine, the rocket can change its direction of thrust, which changes the direction of the rocket. This is called *thrust vectoring*.

A more outdated method is using *vernier engines*, which are two smaller engines that control attitude. However, this adds a large amount of weight to the rocket, so they are less frequently used today.

Pros and Cons of Rocket Control Systems in Atmosphere

- **Movable fins**
  - **Pros:** Simple, reliable, lightweight, effective at high speeds in atmosphere.
  - **Cons:** Ineffective in vacuum, limited control at low speeds or thin atmosphere.
- **Gimbaled engine (Thrust vectoring)**
  - **Pros:** Precise control, works in both atmosphere and vacuum, allows for rapid directional changes.
  - **Cons:** Mechanically complex, heavier, more expensive, potential failure points.
- **Vernier engines**
  - **Pros:** Provides fine attitude control, redundancy.

  - **Cons:** Adds significant weight, less efficient, rarely used in modern rockets.

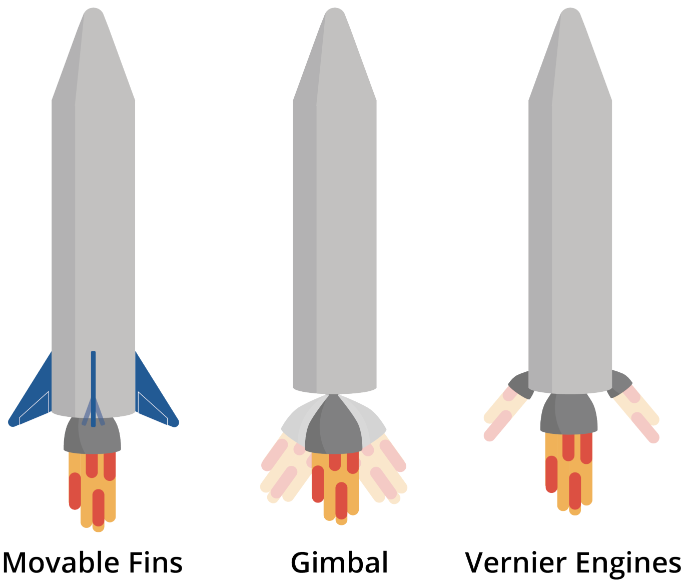


Figure 2.5: Three different methods of engines and their movement in outer space.

2.4 Control in space

The previous section describes ways that engineers control rockets in the atmosphere, but most rockets will end up in the vacuum of space. There are several common ways to adjust the orientation in space.

One method is using *RCS thrusters*. An RCS, or reaction control system, is a series of small thrusters that are used to change the direction and position of a spacecraft. RCS thrusters are usually small, and placed in blocks of 3-4 allowing for precise control in all three axes. They are often used for docking maneuvers, attitude control, and orbital adjustments.

Another method called *reaction wheels* uses angular momentum to rotate the spacecraft. By accelerating and decelerating wheels on three axes, the spacecraft can rotate in any direction.

A third common attitude control technology is *magnetorquer*. Magnetorquers use electromagnets and the earth's magnetic field to adjust the orientation of the spacecraft. Magnetorquers are common in small satellites, but they are not as common in larger spacecraft. To manipulate large spacecraft with magnetorquers effectively, you need either prohibitively high current or an external magnetic field more powerful than the Earth's magnetic field.

Pros and Cons of Rocket Control Systems in Space

- **RCS Thrusters**
  - **Pros:** Precise control, enables translation and rotation, reliable, effective for docking and attitude adjustments.
  - **Cons:** Consumes propellant, limited by onboard fuel, adds weight and complexity.
- **Reaction Wheels**
  - **Pros:** No propellant required, precise attitude control, long operational life, quiet operation.
  - **Cons:** Limited torque, can saturate and require desaturation (often using RCS), mechanical failure risk.
- **Magnetorquers**
  - **Pros:** No propellant required, lightweight, simple design, effective for small satellites in low Earth orbit.
  - **Cons:** Only works near planetary magnetic fields, limited control authority, not suitable for deep space or large spacecraft.

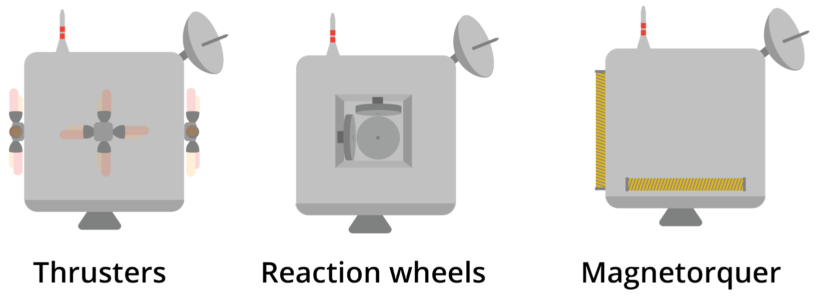


Figure 2.6: Control options for rockets in space.

2.5 Alternative propulsion

One type of alternate propulsion is called a *solar sail*. Solar sails use lightweight reflective surfaces to use photons in space to propel the spacecraft without on-board fuel.

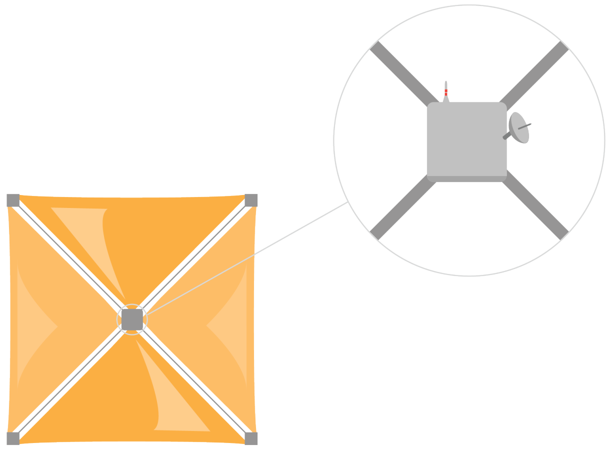


Figure 2.7: Solar sails propel the rocket with photon reflection.

Photons reflect off of the surface of the sail. However, since the surface is not perfectly reflective, some of those photons are absorbed, and they produce a horizontal equal and opposite reaction. That small force causes the net thrust to be slightly skewed away from a right angle to the sail.

Ion propulsion is a form of electric propulsion that accelerates ions to generate thrust. There are two main types: electrostatic and electromagnetic. Electrostatic ion propulsion uses electric fields to accelerate ions, typically xenon, through grids to produce thrust. Electrostatic thrusters use the Coulomb force to accelerate ions, whereas electromagnetic ion propulsion uses both electric and magnetic fields through the Lorentz force to accelerate ions. Both methods are highly efficient and suitable for long-duration space missions, but they produce low thrust compared to chemical rockets.

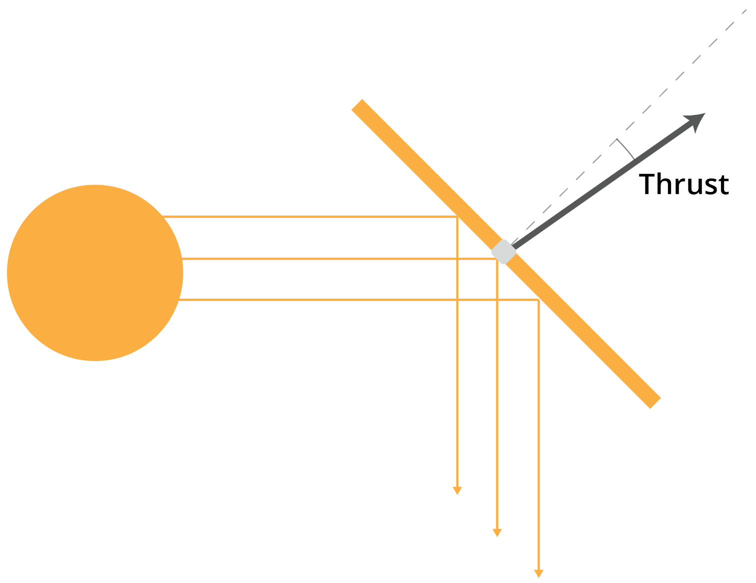


Figure 2.8: A zoomed in version of solar sails.

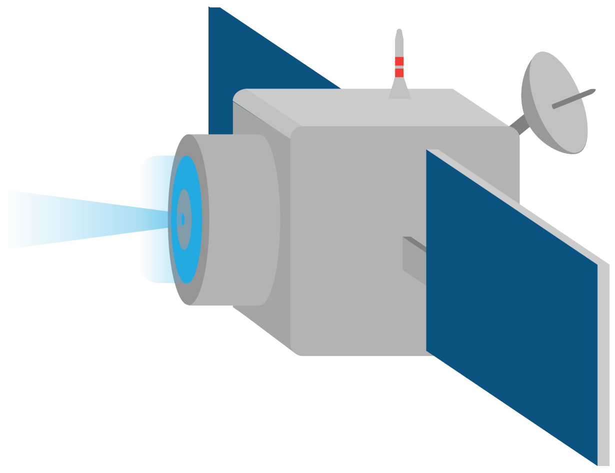


Figure 2.9: An ion thruster

CHAPTER 3

Simulation with Vectors

In an earlier chapter, you wrote a python program that simulated the flight of a hammer to predict its altitude. Your simulation dealt only with scalars. Now, you are ready to create simulations of positions, velocities, accelerations, and forces as vectors.

In this chapter, you are going to simulate two moons that, as they wandered through the vast universe, get caught in each other's gravity well. We will assume there are no other forces acting upon the moons.

3.1 Force, Acceleration, Velocity, and Position

We talked about the magnitude of a gravitational attraction between two masses:

$$F = G\frac{m_1 m_2}{r^2}$$

where F is the magnitude of the force in newtons, m_1 and m_2 are the masses in kg, r is the distance between them in meters, and g is the universal gravitational constant: 6.67430×10^{-11}.

What is the direction? For the two moons, the force on moon 1 will pull toward moon 2. Likewise, the force on moon 2 will pull toward moon 1.

Of course, if something is big (like the sun), you need to be more specific: The force points directly at the center of mass of the object that is generating the force.

Each of the moons will start off with a velocity vector. That velocity vector will change over time as the moon is accelerated by the force of gravity. If you have a mass m with an initial velocity vector of $\vec{v_0}$ that is being accelerated with a constant force vector $\vec{F}$, at time t, the new velocity vector will be:

$$\vec{v_t} = \vec{v_0} + \frac{t}{m}\vec{F}$$

If an object is at an initial position vector of $\vec{p_0}$ and moves with a constant velocity vector $\vec{v}$ for time t, the new position will be given by

$$\vec{p}_t = \vec{p}_0 + t\vec{v}$$

3.2 Simulations and Step Size

As two moons orbit each other, the force, acceleration, velocity, and position are changing smoothly and continuously. It is difficult to simulate truly continuous things on a digital computer.

However, think about how a movie shows you many frames each second. Each frame is a still picture of the state of the system. The more frames per second, the smoother it looks.

We do a similar trick in simulations. We say ”We are going run our simulation in 2 hour steps. We will assume that the acceleration and velocity were constant for those two hours. We will update our position vectors accordingly, then we will recalculate our acceleration and velocity vectors.”

Generally, as you make the step size smaller, your simulation will get more accurate and take longer to execute.

3.3 Make a Text-based Simulation

To start, you are going to write a Python program that simulates the moons and prints out their position for every time step. Later, we will add graphs and even animation.

We are going to assume the two moons are traveling the same plane so we can do all the math and graphing in 2 dimensions.

Each moon will be represented by a dictionary containing the state of the moon:

- Its mass in kilograms
- Its position — A 2-dimensional vector represent x and y coordinates of the center of the moon.
- Its velocity — A 2-dimensional vector
- Its radius — Each moon has a radius so we know when the centers of the two moons are so close to each other that they must have collided.
- Its color — We will use that when do the plots and animations. One moon will be red, the other blue.

There will then be a loop where we will update the positions of the moons and then recalculate the acceleration and velocities.

How much time will be simulated? 100 days or until the moons collide, whichever comes first.

We will use numpy arrays to represent our vectors.

Create a file called `moons.py`, and type in this code:

```
import numpy as np

# Constants
G = 6.67430e-11              # Gravitational constant (Nm^2/kg^2)
SEC_PER_DAY = 24 * 60 * 60   # How many seconds in a day?
MAX_TIME = 100 * SEC_PER_DAY # 100 days
TIME_STEP = 2 * 60 * 60      # Update every two hours

# Create the inital state of Moon 1
m1 = {
    "mass": 6.0e22,  # kg
    "position": np.array([0.0, 200_000_000]),  # m
    "velocity": np.array([100.0, 25.0]),  # m/s
    "radius": 1_500_000.0,  # m
    "color": "red" # For plotting
}

# Create the inital state of Moon 2
m2 = {
    "mass": 11.0e22,  # kg
    "position": np.array([0.0, -150_000_000]),  # m
    "velocity": np.array([-45.0, 2.0]),  # m/s
    "radius": 2_000_000.0,  # m
    "color": "blue" # For plotting
}

# Lists to hold positions and time
position1_log = []
position2_log = []
time_log = []

# Start at time zero seconds
current_time = 0.0

# Loop until current time exceed Max Time
while current_time <= MAX_TIME:

    # Add time and positions to log
    time_log.append(current_time)
    position1_log.append(m1["position"])
    position2_log.append(m2["position"])
    # Print the current time and positions
    print(f"Day {current_time/SEC_PER_DAY:.2f}:")
    print(f"\tMoon 1:({m1['position'][0]:,.1f},{m1['position'][1]:,.1f})")
    print(f"\tMoon 2:({m2['position'][0]:,.1f},{m2['position'][1]:,.1f})")

    # Update the positions based on the current velocities
    m1["position"] = m1["position"] + m1["velocity"] * TIME_STEP
    m2["position"] = m2["position"] + m2["velocity"] * TIME_STEP

    # Find the vector from moon1 to moon2
    delta = m2["position"] - m1["position"]

    # What is the distance between the moons?
    distance = np.linalg.norm(delta)

    # Have the moons collided?
    if distance < m1["radius"] + m2["radius"]:
        print(f"*** Collided {current_time:.1f} seconds in!")
        break

    # What is a unit vector that points from moon1 toward moon2?
    direction = delta / distance

    # Calculate the magnitude of the gravitational attraction
    magnitude = G * m1["mass"] * m2["mass"] / (distance**2)

    # Acceleration vector of moon1 (a = f/m)
    acceleration1 = direction * magnitude / m1["mass"]

    # Acceleration vector of moon2
    acceleration2 = (-1 * direction) * magnitude / m2["mass"]

    # Update the velocity vectors
    m1["velocity"] = m1["velocity"] + acceleration1 * TIME_STEP
    m2["velocity"] = m2["velocity"] + acceleration2 * TIME_STEP

    # Update the clock
    current_time += TIME_STEP

print(f"Generated {len(position1_log)} data points.")
```

When your run the simulation, you will see the positions of the moons for 100 days:

```
> python3 moons.py
Day 0.00:
 Moon 1:(0.0,200,000,000.0)
 Moon 2:(0.0,-150,000,000.0)
Day 0.08:
 Moon 1:(720,000.0,200,180,000.0)
 Moon 2:(-324,000.0,-149,985,600.0)
Day 0.17:
 Moon 1:(1,439,990.7,200,356,896.1)
 Moon 2:(-647,995.0,-149,969,507.0)
...
Day 100.00:
 Moon 1:(119,312,305.5,283,265,313.5)
 Moon 2:(17,393,287.9,-60,319,261.9)
Generated 1201 data points.
```

Look over the code. Make sure you understand what every line does.

3.4 Graph the Paths of the Moons

Now, you will use the matplotlib to graph the paths of the moons. Add this line to the beginning of `moons.py`.

```
import matplotlib.pyplot as plt
```

Add this code to the end of your `moons.py`:

```
# Convert lists to np.arrays
positions1 = np.array(position1_log)
positions2 = np.array(position2_log)

# Create a figure with a set of axes
fig, ax = plt.subplots(1, figsize=(7.2, 10))

# Label the axes
ax.set_xlabel("x (m)")
ax.set_ylabel("y (m)")
ax.set_aspect("equal", adjustable='box')

# Draw the path of the two moons
ax.plot(positions1[:, 0], positions1[:, 1], m1["color"], lw=0.7)
ax.plot(positions2[:, 0], positions2[:, 1], m2["color"], lw=0.7)

# Save out the figure
fig.savefig("plotmoons.png")
```

When you run it, your `plotmoons.png` should look like this:

It is nifty to see the paths, but we don't know where each moon was at a particular time. In fact, it is difficult to figure out which end of each curve was the beginning and which was the ending.

What if we added some lines and labels every 300 steps to put a sense of time into the

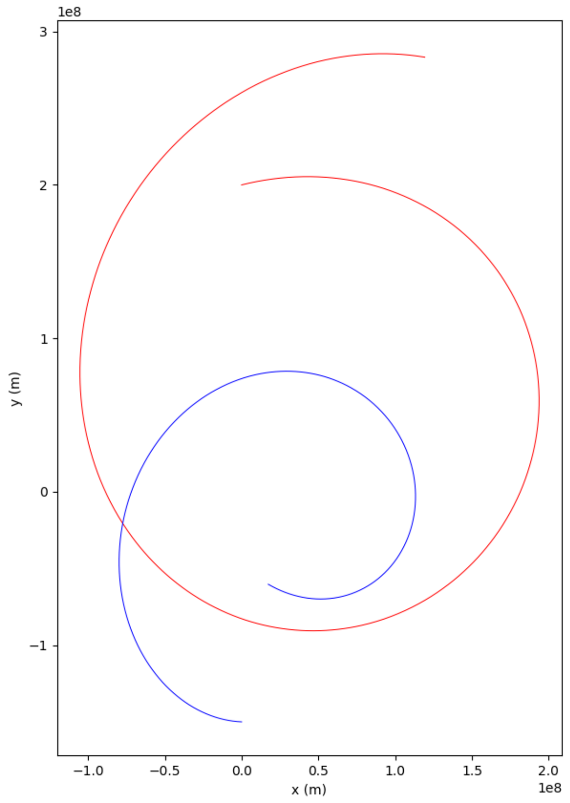


Figure 3.1: Output of moons_01.py.

plot? Add one more constant after the import statements:

```
PAIR_LINE_STEP = 300  # How time steps between pair lines
```

Immediately before you save the figure to the file, add the following code:

```
# Draw some pair lines that help the
# viewer understand time in the graph
i = 0
while i < len(positions1):

    # Where are the moons at the ith entry?
    a = positions1[i, :]
    b = positions2[i, :]
    ax.plot([a[0], b[0]], [a[1], b[1]], "--", c="gray", lw=0.6, marker=".")

    # What is the time at the ith entry?
    t = time_log[i]

    # Label the location of moon 1 with the day
    ax.text(a[0], a[1], f"{t/SEC_PER_DAY:.0f} days")
    i += PAIR_LINE_STEP
```

When you run it, your plot should look like this:

Now you can get a feel for what happened. The moons were attracted to each other by gravity and started to circle each other. The heavier moon accelerates less quickly, so it makes a smaller loop.

Maybe we will get a better feel for what is happening if we look at more time. Let's increase it to 400 days. Change the relevant constant:

```
MAX_TIME = 400 * SEC_PER_DAY # 100 days
```

Now it should look like this:

Now you can see the pattern. They are rotating around each other and the pair is gradually migrating up and to the right.

3.5 Conservation of Momentum

Recall the idea of momentum being conserved in a system. In these programs, you are observing an extra important idea: *the momentum of a system will be conserved*. That is, absent forces from outside the system, the velocity of the center of mass will not change.

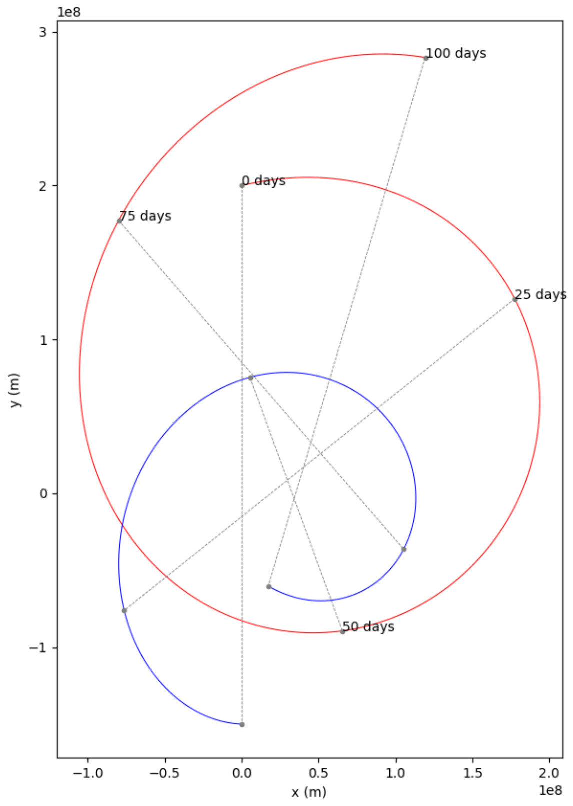


Figure 3.2: Plot moons_02.py.

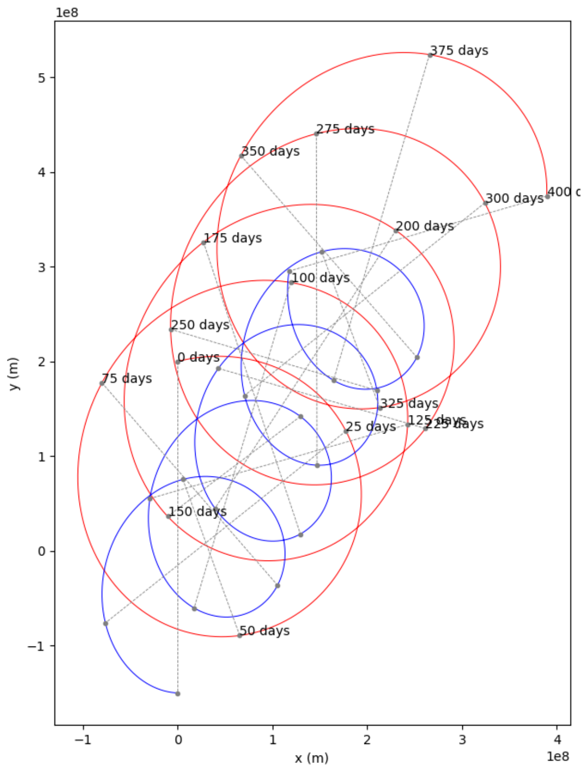


Figure 3.3: Output of plot moons_03.py.

We can compute the initial center of mass and its velocity. In both cases, we just do a weighted average using the mass of the moon as the weight.

Immediately after you initialize the state of two moons, calculate the initial center of mass and its velocity:

```
# Calculate the initial position and velocity of the center of mass
tm = m1["mass"] + m2["mass"]  # Total mass
cm_position = (m1["mass"] * m1["position"] + m2["mass"] * m2["position"]) / tm
cm_velocity = (m1["mass"] * m1["velocity"] + m2["mass"] * m2["velocity"]) / tm
```

Let's record the center of mass for each time. Before the loop starts, create a list to hold them:

```
cm_log = []
```

Inside the loop (before any calculations), append the current center of mass position to the log:

```
    cm_log.append(cm_position)
```

Anywhere later in the loop (after you update the positions of the moon), update `cm_position`:

```
    # Update the center of mass
    cm_position = cm_position + cm_velocity * TIME_STEP
```

Now, let's look at the positions of the moons relative to the center of mass. Before you do any plotting, convert the list to a numpy array and subtract it from the positions:

```
cms = np.array(cm_log)

# Make positions relative to the center of mass
positions1 = positions1 - cms
positions2 = positions2 - cms
```

When you run it, you can really see what is happening:

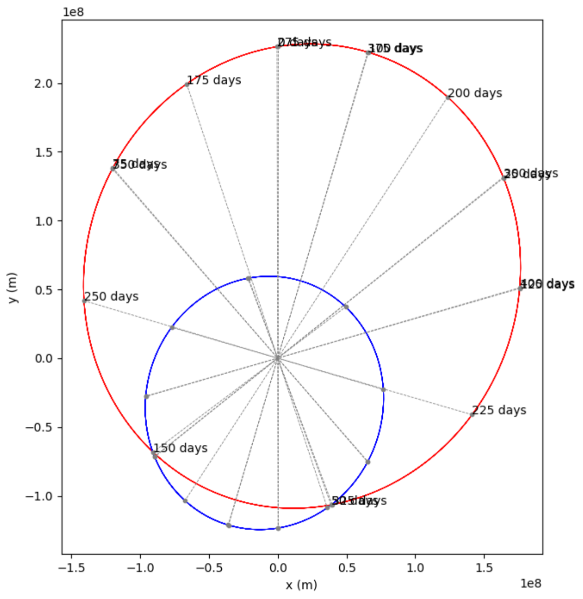


Figure 3.4: Output of plotmoons_04.py.

The moons are tracing elliptical paths. The center of mass is the focus point for both of them.

3.6 Animation

One of the features of matplotlib that not a lot of people understand is how to make animations with it. This seems like a really great opportunity to make an animation showing the position, velocity, acceleration of the moons. We will also show the center of mass.

The trick to animations is that you create a bunch ”artist” objects. You create a function that updates the artists. matplotlib will call your functions, tell the artists to draw themselves, and make a movie out of that.

Make a copy of `moons.py` called `animate_moons.py`.

Edit it to look like this:

```
import numpy as np
import matplotlib.pyplot as plt

# Import animation support and artists
from matplotlib.animation import FuncAnimation
from matplotlib.patches import Circle, FancyArrow
from matplotlib.text import Text

# Constants
G = 6.67430e-11  # Gravitational constant (Nm^2/kg^2)
SEC_PER_DAY = 24 * 60 * 60  # How many seconds in a day?
MAX_TIME = 400 * SEC_PER_DAY  # 100 days
TIME_STEP = 12 * 60 * 60  # Update every 12 hours
FRAMECOUNT = MAX_TIME / TIME_STEP  # How many frames in animation
ANI_INTERVAL = 1000 / 50  # ms for each frame in animation

# The velocity and acceleration vectors are invisible
# unless we scale them up.  A lot.
VSCALE = 140000.0
ASCALE = VSCALE * 800000.0

# Create the inital state of Moon 1
m1 = {
    "mass": 6.0e22,  # kg
    "position": np.array([0.0, 200_000_000]),  # m
    "velocity": np.array([100.0, 25.0]),  # m/s
    "radius": 1_500_000.0,  # m
    "color": "red",  # For plotting
}

# Create the inital state of Moon 2
m2 = {
    "mass": 11.0e22,  # kg
    "position": np.array([0.0, -150_000_000]),  # m
    "velocity": np.array([-45.0, 2.0]),  # m/s
    "radius": 2_000_000.0,  # m
    "color": "blue",  # For plotting
}

# Calculate the initial position and velocity of the center of mass
```

```
tm = m1["mass"] + m2["mass"]  # Total mass
cm_position = (m1["mass"] * m1["position"] + m2["mass"] * m2["position"]) / tm
cm_velocity = (m1["mass"] * m1["velocity"] + m2["mass"] * m2["velocity"]) / tm

# Start at time zero seconds
current_time = 0.0

# Create the figure and axis
fig, ax = plt.subplots(1, figsize=(7.2, 10))

# Set up the axes
ax.set_xlabel("x (m)")
ax.set_xlim((-1.2e8, 4e8))
ax.set_ylabel("y (m)")
ax.set_ylim((-1.6e8, 5.5e8))
ax.set_aspect("equal", adjustable="box")
fig.tight_layout()

# Create artists that will be edited in animation
time_text = ax.add_artist(Text(0.03, 0.95, "", transform=ax.transAxes))
circle1 = ax.add_artist(Circle((0, 0), radius=m1["radius"], color=m1["color"]))
circle2 = ax.add_artist(Circle((0, 0), radius=m2["radius"], color=m2["color"]))
circle_cm = ax.add_artist(Circle((0, 0), radius=m2["radius"], color="purple"))
varrow1 = ax.add_artist(FancyArrow(0, 0, 0, 0, color="green", head_width=m1["radius"]))
varrow2 = ax.add_artist(FancyArrow(0, 0, 0, 0, color="green", head_width=m2["radius"]))
acc_arrow1 = ax.add_artist(
    FancyArrow(0, 0, 0, 0, color="purple", head_width=m1["radius"])
)
acc_arrow2 = ax.add_artist(
    FancyArrow(0, 0, 0, 0, color="purple", head_width=m2["radius"])
)


# This function will get called for every frame
def animate(frame):

    # Global variables needed in scope from the model
    global cm_position, cm_velocity, current_time, m1, m2

    # Global variables needed in scope from the artists
    global time_text, varrow1, varrow2, acc_arrow1, acc_arrow2, circle1, circle2, circle_cm

    print(f"Updating artists for day {current_time/SEC_PER_DAY:.1f}.")

    # Update the positions based on the current velocities
    m1["position"] = m1["position"] + m1["velocity"] * TIME_STEP
    m2["position"] = m2["position"] + m2["velocity"] * TIME_STEP

    # Update day label
    time_text.set_text(f"Day {current_time/SEC_PER_DAY:.0f}")

    # Update positions of circles
```

```
circle1.set_center(m1["position"])
circle2.set_center(m2["position"])

# Update velocity arrows
varrow1.set_data(
    x=m1["position"][0],
    y=m1["position"][1],
    dx=VSCALE * m1["velocity"][0],
    dy=VSCALE * m1["velocity"][1],
)
varrow2.set_data(
    x=m2["position"][0],
    y=m2["position"][1],
    dx=VSCALE * m2["velocity"][0],
    dy=VSCALE * m2["velocity"][1],
)

# Update the center of mass
cm_position = cm_position + cm_velocity * TIME_STEP
circle_cm.set_center(cm_position)

# Find the vector from moon1 to moon2
delta = m2["position"] - m1["position"]

# What is the distance between the moons?
distance = np.linalg.norm(delta)

# Have the moons collided?
if distance < m1["radius"] + m2["radius"]:
    print(f"*** Collided {current_time:.1f} seconds in!")

# What is a unit vector that points from moon1 toward moon2?
direction = delta / distance

# Calculate the magnitude of the gravitational attraction
magnitude = G * m1["mass"] * m2["mass"] / (distance**2)

# Acceleration vector of moons (a = f/m)
acceleration1 = direction * magnitude / m1["mass"]
acceleration2 = (-1 * direction) * magnitude / m2["mass"]

# Update the acceleration arrows
acc_arrow1.set_data(
    x=m1["position"][0],
    y=m1["position"][1],
    dx=ASCALE * acceleration1[0],
    dy=ASCALE * acceleration1[1],
)
acc_arrow2.set_data(
    x=m2["position"][0],
    y=m2["position"][1],
    dx=ASCALE * acceleration2[0],
        dy=ASCALE * acceleration2[1],
    )

    # Update the velocity vectors
    m1["velocity"] = m1["velocity"] + acceleration1 * TIME_STEP
    m2["velocity"] = m2["velocity"] + acceleration2 * TIME_STEP

    # Update the clock
    current_time += TIME_STEP

    # Return the artists that need to be redrawn
    return (
        time_text,
        varrow1,
        varrow2,
        acc_arrow1,
        acc_arrow2,
        circle1,
        circle2,
        circle_cm,
    )


# Make the rendering happen
animation = FuncAnimation(
    fig,
    animate,
    np.arange(FRAMECOUNT),
    interval=ANI_INTERVAL
)

# Save the rendering to a video file
animation.save("moonmovie.mp4")
```

When you run this, it will take longer than the previous versions. You should have a video file that shows a simulation of the moons tracing their elliptical paths around their center of mass:

3.7 Challenge: The Three-Body Problem

It is time to stretch a little as a physicist and programmer: You are going to make a new version of `moons.py` that handles three moons instead of just two.

This is known as ”The Three-Body Problem”, and people have tried for centuries to come up with a way to figure out (from the initial conditions) where the three moons would be at time t without doing a simulation. And no one has.

For a lot of problems, the outcome is not very sensitive to the initial conditions. For

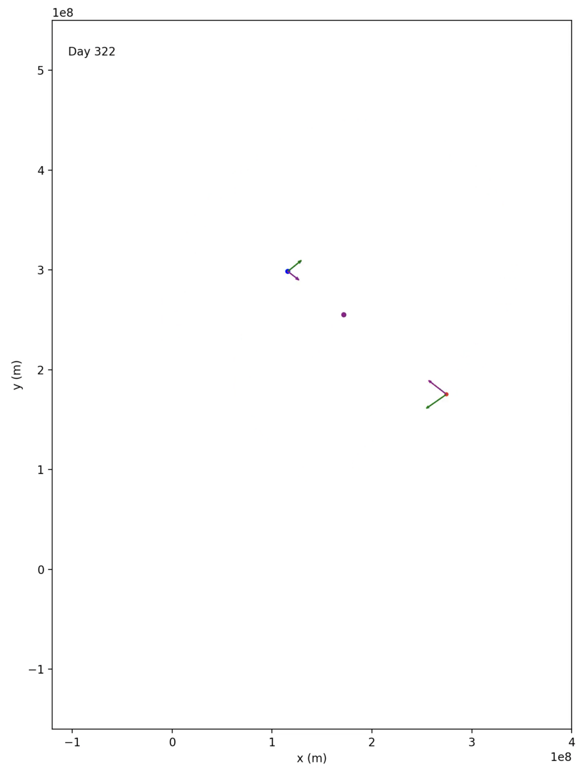


Figure 3.5: A still of the animation produced.

example, consider the flight of a cannonball: If it leaves the muzzle of the cannon a little faster, it will go a little farther.

For the three-body problem, the outcome can be radically different even if the initial conditions are very similar.

(There is a whole field of mathematics studying systems that are very sensitive to initial conditions. It is known as *dynamical systems* or *chaos theory*.)

Copy `moons.py` to `3moons.py`. Here is a reasonable initial state for your third moon:

```
m3 = {
    "mass": 4.0e22,  # kg
    "position": np.array([50_000_000, 80_000_000]),  # m
    "velocity": np.array([-30.0, -35.0]),  # m/s
    "radius": 1_700_000.0,  # m
    "color": "green"
}
```

If we run that simulation for 100 days, we get a plot like this:

Visibly, you can see this is very different from the two-body problem that just traced ellipses around the center of mass.

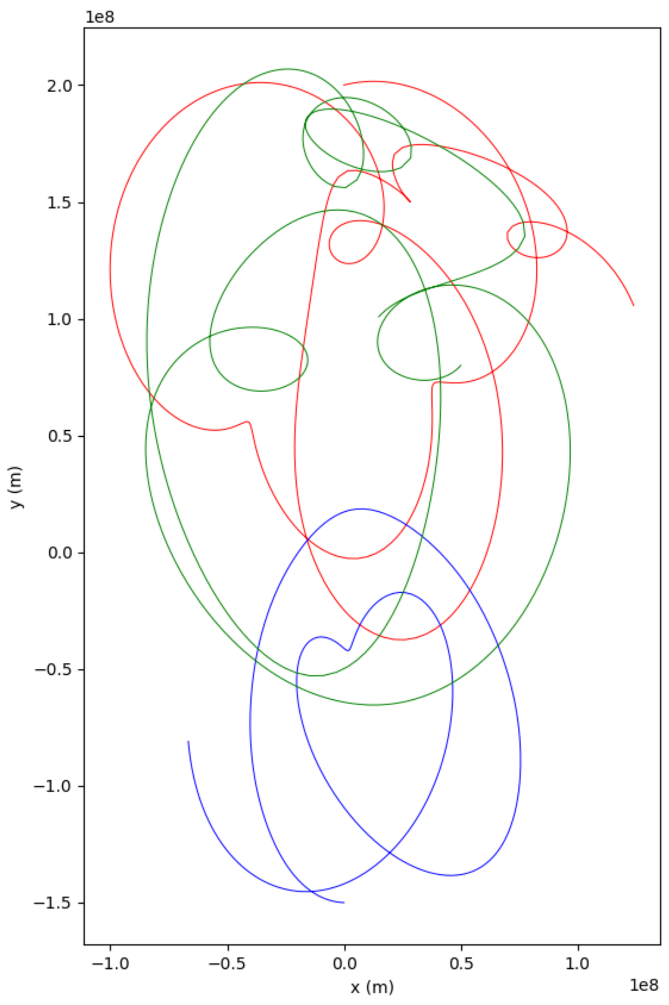


Figure 3.6: The Three Body Problem.

Chapter 4

Longitude and Latitude

4.1 Longitude and Latitude

The Earth can be represented as a sphere, and the position of a point on its surface can be described using two coordinates: latitude and longitude.

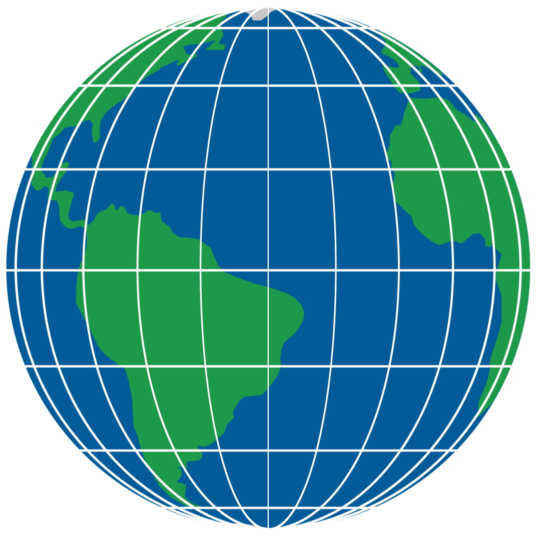


Figure 4.1: A diagram of latitude and longitude.

Latitude is a measure of a point's distance north or south of the equator, expressed in degrees. It ranges from $-90°$ at the South Pole to $+90°$ at the North Pole, with $0°$ representing the Equator. (See figures 4.2 and 4.3)

Longitude, on the other hand, measures a point's distance east or west of the Prime Meridian (which passes through Greenwich, England). It ranges from $-180°$ to $+180°$, with the Prime Meridian represented as $0°$. (See figures 4.4 and 4.5)

The coordinates follow a system of latitude and then longitude (in the majority of contexts). Here is the Longitude and Latitude of four different large cities:

- **London, England**: 51° N 0° W

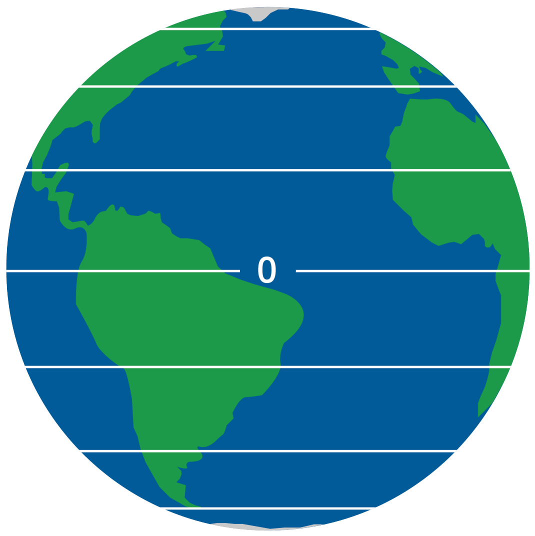


Figure 4.2: Latitude drawn on the earth.

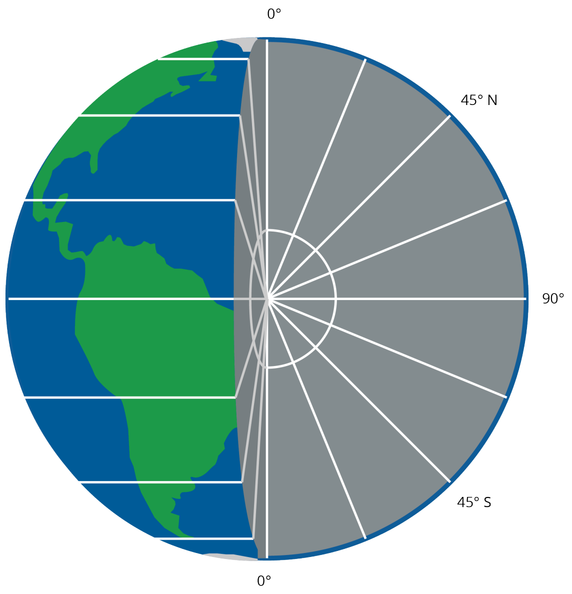


Figure 4.3: A cross-section of the earth showing latitude.

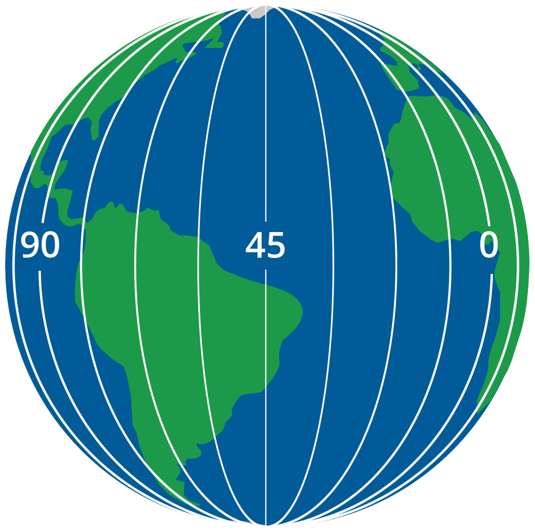


Figure 4.4: Longitudinal lines drawn on the earth.

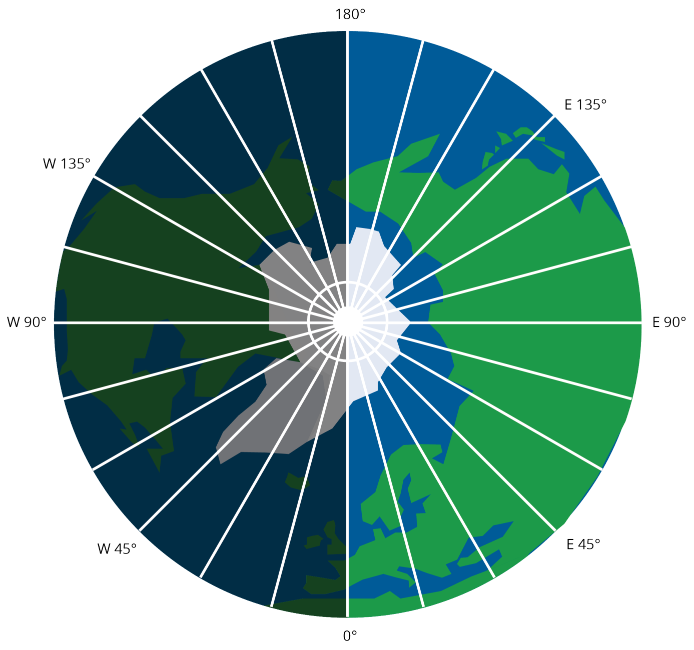


Figure 4.5: A cross-section of the earth showing longitude.

- **New York City, New York, USA**: 40° N 74° W
- **Sydney, New South Whales, Australia**: 33° S 150° E
- **Rio de Janerio, Brazil**: 22° S 43° W

These coordinates may be broken down further into minute and second components as well, for futher geometric accuracy. Each degree (°) is divided into 60 minutes (′). Each minute (′) is divided into 60 seconds ("). Just like in time. Note some do not

- **London, England**: 51° 30‘ 26” N 0° 7‘ 39” W
- **New York City, New York, USA**: 40° 42‘ 46” N 74° 0‘ 22” W
- **Sydney, New South Whales, Australia**: 33° 52‘ 0” S 150° 12‘ 0” E
- **Rio de Janerio, Brazil**: 22° 54‘ 0” S 43° 12‘ 0” W

4.2 Nautical Mile

A nautical mile is a unit of measurement used primarily in aviation and maritime contexts. It is based on the circumference of the Earth, and is defined as one minute (1/60°) of latitude; in other words, **one minute of latitude is equal to 1 nautical mile.** This makes it directly related to the Earth’s geometry, unlike a kilometer or a mile, which are arbitrary in nature. The exact value of a nautical mile can vary slightly depending on which type of latitude you use (e.g., geodetic, geocentric, etc.), but for practical purposes, it is often approximated as 1.852 kilometers or 1.15078 statute miles.

4.3 Haversine Formula

The haversine formula is an important equation in navigation for giving great-circle distances between two points on a sphere from their longitudes and latitudes. It is especially useful when it comes to calculating distances between points on the surface of the Earth, which we represent as a sphere for simplicity. See Figure 4.6

In its basic form, the haversine formula is as follows:

The angle found using haversine $\text{haversine}(\theta) = sin^2\left(\frac{\theta}{2}\right)$:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)$$

Compute c which determines the angular distance using $atan2$, a computer science tan-

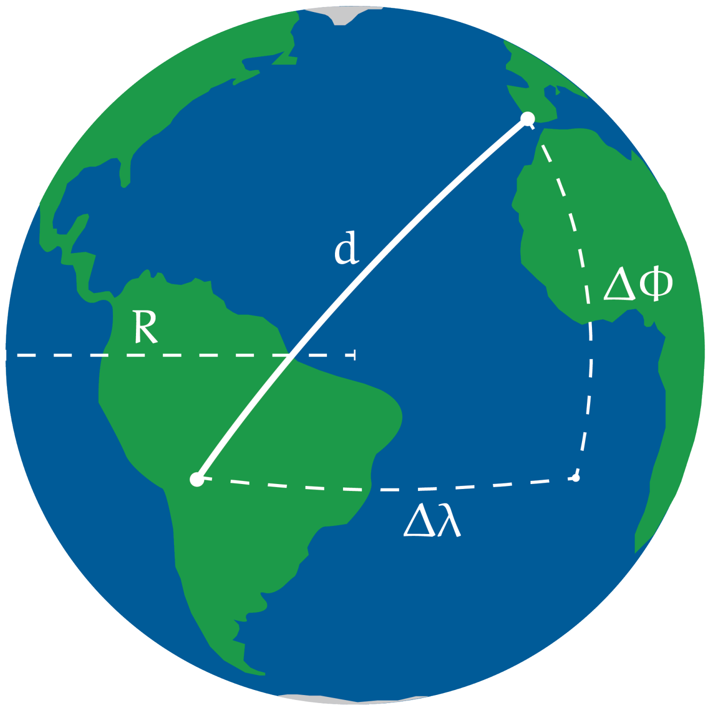


Figure 4.6: A diagram of the haversine formula.

gential function with 2 arguments equivalent to $2\arcsin(\sqrt{a})$[1]:

$$c = 2 \cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$

Find d, the true distance along the sphere:

$$d = R \cdot c$$

In one line, this is:

$$d = 2R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Here, ϕ_1 and ϕ_2 represent the latitudes of the two points (in radians), $\Delta\phi$ and $\Delta\lambda$ represent the differences in latitude and longitude (also in radians), and R is the radius of the Earth (using whatever unit of distance you desire). The result, d, is the distance between the two points along the surface of the sphere. Notice that d is equal to the formula for the *arc length* of a sector. A sphere is just a bunch of circles with radius r, so it follows that we can use formulas for circles in our spherical calculations.

This is why planes fly arc-shaped paths rather than straight paths from points $A \to B$. Planes follow the shortest path between two points on a sphere, called a *great-circle route*. On a map, like plane flight-progress maps or the Mercator projection, the straight path looks very arc shaped, but this is just a geographical illusion. This straight distance can be calculated using the haversine formula, ultiamtely minimizing distance and fuel using the 'as-the-crow-flies' path.

[1]Note that this function is not available on calculators, but commonly used in computer science programs. You may use the singular argument equivalent in calculations.

Chapter 5

Tides and Eclipses

Your life on earth involves many orbital paths:

- The earth is spinning. If you are standing at the equator, you are traveling at 1,674 km per hour around the center of the planet. We are all spinning east, which is why the sun comes up in the east and sets in the west.
- The earth is orbiting the sun. It takes 365.242 days for the earth to go once around the sun. This is why different constellations appear at different times during the year — we only see the stars at night and the direction of night shifts as the earth moves around the sun.
- The moon is orbiting the earth. The moon travels once around the earth once every 27.3 days.

You can see the effects of these orbits on our planet. Let's go over a few.

5.1 Leap Years

Note that it takes 365.242 days for the earth to go around the sun. If we declared "The calendar will *always* be 365 days per year!" then the seasons would gradually shift by 0.242 days every year. After a century, they would have migrated 24 days.

So, we made a rule: "Every fourth year, we will add an extra day to the calendar!" The years 2021, 2022,and 2023 get no February 29th, but 2024 does.

That got us a calendar with an average 365.25 days per year, so the seasons would not have migrated as quickly, but they still would have migrated about three days every four hundred years.

So, we made another rule: 'There will be no February 29th in the three century years (multiples of 100) that are not multiples of 400.' So the year 1900 had no Feb 29, but the year 2000 had one. Now, the average number of days per year is 365.2425.

5.2 Phases of the Moon

The earth, the moon, and the sun form a triangle. If you were standing on the moon, you could measure the angle between the light coming from the sun and the the light going to the earth. That angle would fluctuate between 0 degrees and 180 degrees.

- When the angle was close to 0, the people on earth would see a full moon (fully illuminated disc).
- When the angle was close to 90 degrees, the people on earth would see a half moon.
- When the angle was nearing 180 degrees, the people on earth would see a slim crescent.
- When the angle was very close to 180 degrees, the moon would be dark. This is called a new moon (fully darked disc).

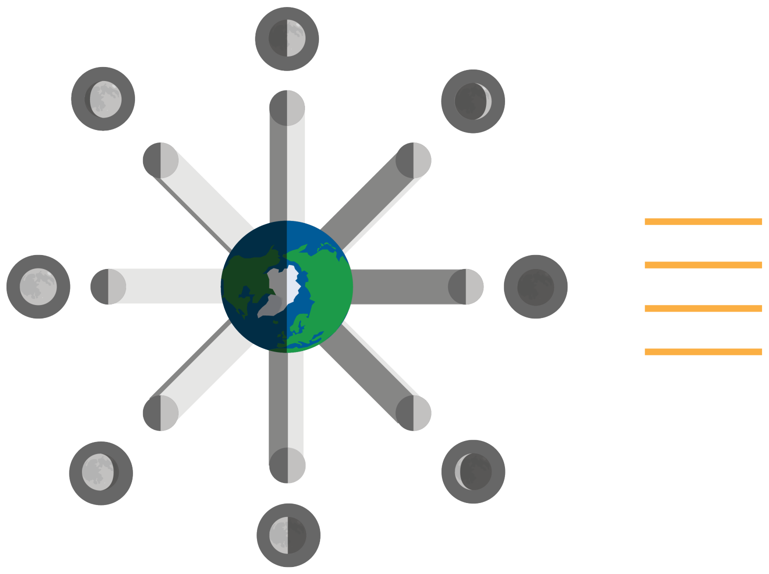


Figure 5.1: The moon makes a revolvement around the earth in different phases every month.

Even though it takes 27.3 days for the moon to travel around the earth once, it takes 29.5 days to get from one full moon to the next. Why? In the 27.3 days that it took the moon to travel around the earth, the earth has moved about 17 degrees around the sun. To get back into the same triangle configuration takes another 2.2 days.

To explain why we often see a curve in the shadow of the moon, we can look at a ball that has one side painted yellow and the other red.

As we rotate the ball, we can see that the straight color boundary between each hemisphere

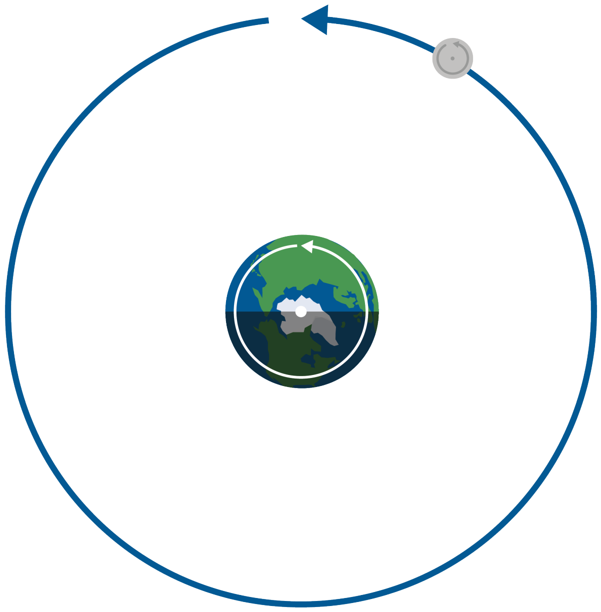


Figure 5.2: The moon and earth rotate at the same time. Both revolve around its own axis at the same time.

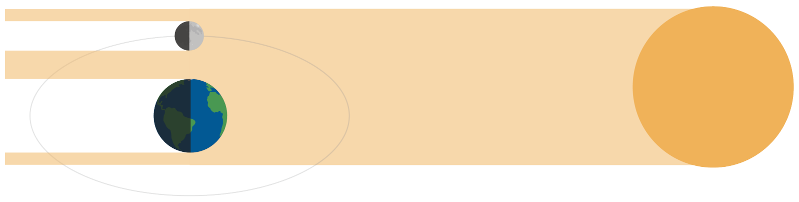


Figure 5.3: A half moon circumstance in space.

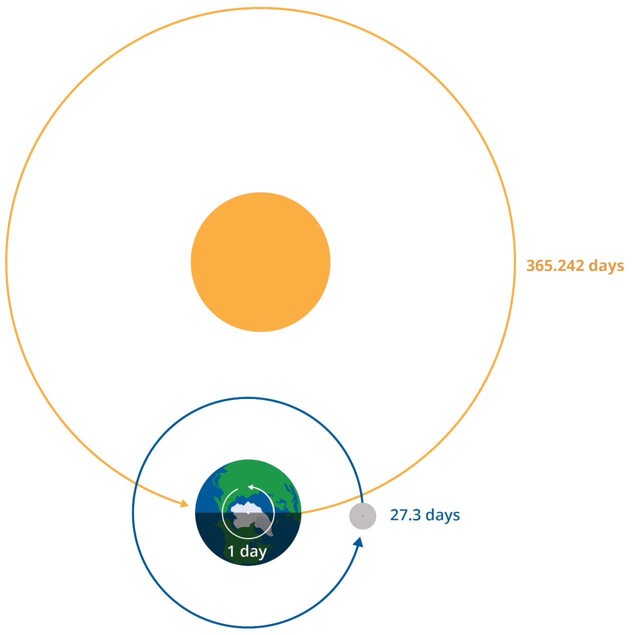


Figure 5.4: The earth, sun, and moon schedules of full rotations.

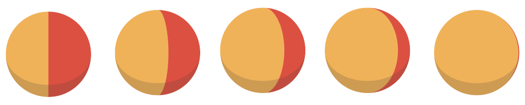


Figure 5.5: A rotating representation of the moons phases.

begins to look curved. The curve we see in the moon is due to this same basic principle of how the shading of spheres works.

FIXME: Add text about scale In all of these graphics, we have been using incorrect scale. Here is the true scale of the distance of the earth and the moon with accurate radii:

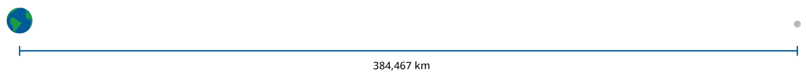


Figure 5.6: The moon and earth in their respective scale.

FIXME: Add text about tidal lock

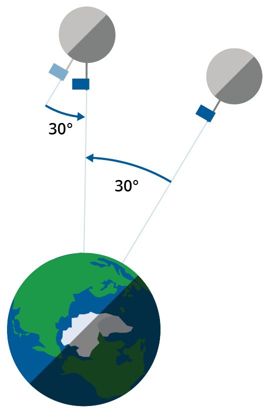


Figure 5.7: FIXME.

5.3 Eclipses

While the earth orbits the sun and the moon orbits the earth, the two orbits are *not in the same plane*. We call the plane that the earth orbits the sun in the *ecliptic plane*. The plane of the moon's orbit is about 5° tilted from the ecliptic plane.

Note that the moon passes through the ecliptic plane only twice every 27.3 days. Imagine that the instant it passed through the ecliptic plane was also the precise instant of a full moon. The sun, the earth, and the moon would be in a straight line! The earth would cast a shadow upon the moon — it would go from a bright full moon to a dark moon until

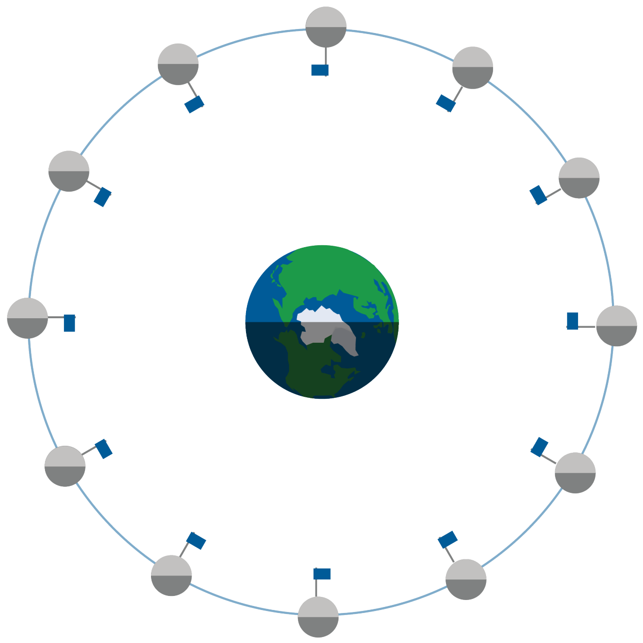


Figure 5.8: FIXME.

the moon moved back out of the shadow of the earth. This is known as a *lunar eclipse*.

The diameter of the moon is a little more than a quarter the diameter of the earth, so they don't have to be in perfect alignment for the moon to be darkened. Lunar eclipses actually happen once or twice per year.

Now, imagine that the instant the moon passed through the ecliptic plane was also the precise instant of a new moon. The sun, the moon, and the earth would be in a straight line! The moon would cast a shadow upon some part of the earth. To a person in that shadow, the sun disappear behind the moon. This is known as a *solar eclipse*.

The sun is pretty big, so if the moon blots out just part of it, we call it a *partial solar eclipse*, as seen in Figure 5.10. There are a few partial solar eclipses every year. Note that because the moon's shadow is too small to shade the whole earth, only certain parts of the world will experience any solar eclipses.

Every 18 months or so, there is a total eclipse of the sun. Once again, only certain parts of the world experience it. You can expect to experience a total eclipse of the sun at your home about once every 375 years.

5.4 The Far Side of the Moon

Like the earth, the moon spins on its axis. Due to earth's gravity, the rotation of the moon slowed down until its spin matched the rate it orbits earth. That is: we are always looking

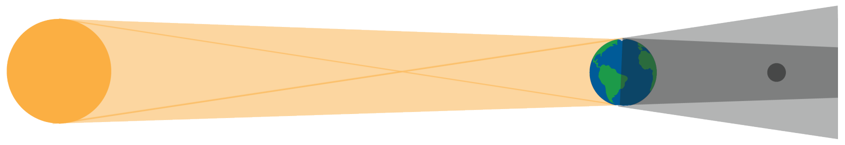


Figure 5.9: Lunar Eclipses happen when the earth comes between the moon and the sun, causing the moon to be darkened by the earth's shadow.

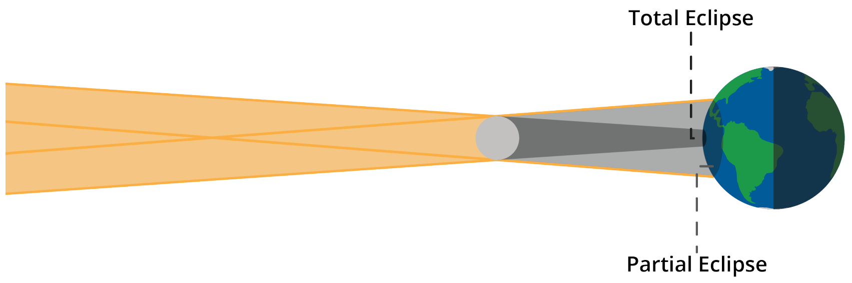


Figure 5.10: Partial and total solar eclipses.

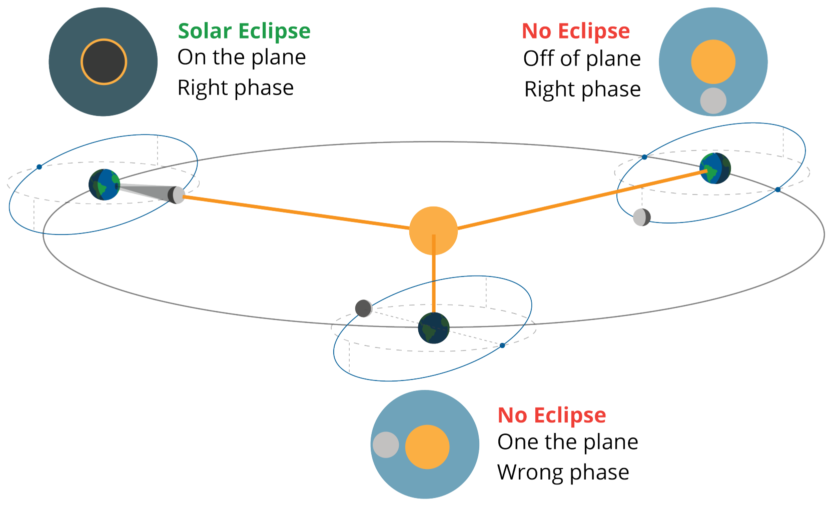


Figure 5.11: Different options for an eclipse.

at the same side of the moon. Until we orbited the moon, we had no idea what the far side looked like.

Some people call it "The Dark Side of the Moon", but it gets just as much sunshine as the side that faces earth. The name comes from the fact that we lose communication with spacecraft (like the Apollo missions to the moon) when they are on the far side of the moon. When we lose communications with a craft, we often say "It went dark".

5.5 Tides

When we say "The moon orbits the earth", that is a bit of an oversimplification. The force of gravity that pulls the moon toward the earth, also pulls the earth toward the moon. The earth is about 81 times heavier than the moon, so the moon moves more, but the moon definitely moves the earth.

The center of the moon and the center of the Earth rotate around each other. The point they rotate around is inside the the earth, but it is closer to the surface of the earth than it is to the center of the earth. This is referred to as a *barycenter*.

Orbits happen, remember, when the centripetal force is equal to gravitational force. So the centripetal force created by the earth being swung by the moon is equal to the gravitational force that the moon exerts on all the mass on the moon.

However.

The parts of the earth that are closer to the moon experience less centripetal force (away from the moon) and more gravitational force (toward the moon).

The parts of the earth that are farther from the moon experience more centripetal force (away from the moon) and less gravitational force (away from the moon).

The effects are not big. For example, you won't notice that you can jump higher when the moon is overhead. You will lose only about 1/200,000 of your weight.

But the ocean is huge: 1/200,000 of its weight is a lot of force.

The water in the oceans bulges a little both toward the moon and away from it.

The earth is still rotating. If you are at the beach as your longitude slides into one of these bulges, you say "Hey, the tide is rising!" The peak of these bulges is known as "*high tide*". Because there is a bulge on each side of the planet, high tide comes twice a day.

This is a *lunar tide* – because it is caused by the moon. There is a similar effect from the sun, but the sun is very, very far away: solar tidal forces are about half as powerful lunar tidal forces. When the sun and the moon work together, the tides are stronger. This is called a *spring tide*. Spring tides don't happen in the spring time; they happen close to full moons and new moons.

When the moon and the sun are working against each other, the tides are weaker. This is called a *neap tide*. Neap tides happen when you see a half moon in the sky.

5.5.1 Computing the Forces

We are enumerating several forces that shape the water on the planet. All these forces are pulling on your body too. In these exercises, you are going to calculate how each force would effect a 1 kg mass on the surface of the earth.

Here are some numbers you will need:

- The mass of the earth: 5.97219×10^{24} kg
- The mass of the sun: 1.9891×10^{30} kg
- The mass of the moon: 7.347673×10^{22} kg
- Radius of the earth at the equator: $6,371$ km
- Average distance from the center of the earth to the center of the sun: 149.6×10^{6} km
- Average distance from the center of the moon to the center of the earth: $384,467$ km.

Exercise 2 Life Among the Orbits 1: Earth Gravity

Working Space

If the earth were still and alone in the universe, there would still be the force of gravity. We have said that that a kilogram on the surface of the earth is pulled toward the center of the earth with a force of 9.8 N.

Confirm that the gravity of the earth pulls a 1kg mass on the surface of the planet with a force of about 9.8 N.

You will need the formula for gravitation:

$$F_g = \frac{g m_1 m_2}{r^2}$$

If we measure distance in km and mass in kg, the gravitation constant g is 6.67430×10^{-17}.

Answer on Page 57

Exercise 3 Life Among the Orbits 2: Earth Centripetal Force

Working Space

What if we add the spinning of the earth? The spinning would try to throw the kg into space. The formula for centripetal force is

$$F_c = \frac{mv^2}{r}$$

Calculate the centripetal force on a 1 kg mass on the surface of the earth. It doesn't fly off into space, so the force due to gravity must be bigger. How many times bigger?

Assume that the mass is on the equator, thus rotating around the earth at 465 m/s.

Does the centripetal force increase, decrease, or stay the same as you get closer to the north pole?

Answer on Page 57

Exercise 4 Life Among the Orbits 3: The Moon's Gravity

Working Space

Now we add the moon's gravitational force to our model.

When the moon is directly overhead, how strongly will it pull at the 1 kg mass on the equator?

When the moon is directly underfoot, how strongly will it pull at the 1 kg mass on the equator?

Is that a big difference?

Answer on Page 58

Exercise 5 Life Among the Orbits 4: The Swing of the Moon

Working Space

Now we add the moon's motion. The moon and the earth swing each other around. This creates a centripetal force. They both travel in nearly a circle centered at their center of mass.

How far is the center of mass of the moon and the earth from the center of the earth? (You can imagine a see-saw with the center of the earth on one end and the center of the moon on the other. Where would the balance point be?)

What point on the surface of the earth is closest to the center of mass? How far is it?

What point on the surface of the earth is farthest from the center of mass? How far is it?

Answer on Page 58

Exercise 6 Life Among the Orbits 5: Lunar Centripetal Force

Working Space

The moon swings us around that center of mass once every 27.3 days. (Forget about the spinning of the earth for this part.) What is the largest and smallest centripetal forces on the surface of the earth created by this swinging

What is the largest centripetal force on a 1 kg mass with the moon directly underfoot? (You need an answer from the previous question: There is a point on the surface of the earth that is 11,044,000 m from the center of gravity.)

What is the resulting centripetal force on a 1 kg mass with the moon directly overhead? (You will need the other answer from the previous exercise: That point is 1,698,000 m from the center of mass of the moon and the earth.)

For this problem is probably easier to use this formula for centripetal force:

$$F_c = mr\omega^2$$

Where m is mass in kg, r is radius in m, and ω is the angular velocity in radians per second.

Answer on Page 59

Exercise 7 Life Among the Orbits 6: Net Force

Working Space

Now add together the two forces at both the nearest point to the moon and the farthest.

Answer on Page 59

5.5.2 Solar Tidal Forces

The sun has a much larger gravitational effect on the earth than the moon does:

- When the sun is overhead, it will pull on a 1 kg mass with a force of about 0.00593 N.
- When the moon is overhead, it will pull on a 1 kg mass with a force of about 0.0000343 N.

Why are lunar tides about twice as powerful solar tides?

Tides occur because the pull of gravity and the pull of the centripetal force are out of balance somewhere on the planet. The sun is so far away that the effects of gravitational and centripetal forces are very close to equal everywhere on earth.

Appendix A

Answers to Exercises

Answer to Exercise ?? (on page 6)

$$v = \sqrt{3.721(3.4 \times 10^6)} = 3,557 \text{ m/s}$$

The circular orbit is $2\pi(3.4 \times 10^6) = 21.4 \times 10^6$ meters in circumference.

The period of the orbit is $(21.4 \times 10^6)/3,557 \approx 6,000$ seconds.

Answer to Exercise 2 (on page 50)

The earth and 1 kg on the surface would attract each other with a force of:

$$F_g = \frac{\left(6.67430 \times 10^{-17}\right)\left(5.97219 \times 10^{24}\right)(1)}{6,371^2} = \frac{3.98583 \times 10^8}{4.0590 \times 10^7} = 9.7987 \text{ N}$$

Thus, if the earth were still and alone in the universe, the oceans would form a perfect sphere.

Answer to Exercise 3 (on page 51)

$$F_c = \frac{(1)(465)^2}{6,371,000} = 0.03373 \text{ N}$$

So the spinning of the earth is trying to throw you into space, but the force of gravity is about 289 times more powerful.

This centripetal force decreases as you move from the equator to the north pole. In fact, at the north pole, there is no centripetal force. Thus, the spinning of the earth makes the oceans an oblate ellipsoid instead of a perfect sphere: the diameter going from pole-to-pole is shorter than a diameter measured at the equator.

You should feel a teensy-tiny bit lighter on your feet at the equator than you do at the north pole: 0.34% lighter.

Answer to Exercise 4 (on page 52)

Overhead, the moon is $384,467 - 6,371 = 378,096$ km from your 1 kg mass.

$$F_g = \frac{gm_1m_2}{r^2} = \frac{(6.67430 \times 10^{-17})\,(7.347673 \times 10^{22})\,(1)}{378,096^2} = \frac{4.9040574 \times 10^6}{1.42956585216 \times 10^{11}} = 3.43058 \times 10^{-5} \text{ N}$$

This is a very small force: The force due to earth's gravity is nearly three hundred thousand times stronger.

Underfoot, the moon is $384,467 + 6,371 = 390,838$

$$F_g = \frac{gm_1m_2}{r^2} = \frac{(6.67430 \times 10^{-17})\,(7.347673 \times 10^{22})\,(1)}{390,838^2} = \frac{4.9040574 \times 10^6}{1.52754 \times 10^{11}} = 3.2103 \times 10^{-5} \text{ N}$$

The force due to the moon's gravity is about 6% stronger when the the moon is overhead than when it is underfoot.

Answer to Exercise 5 (on page 53)

If we let r be the distance (in km) from the center of the earth to the center of mass, the distance from the center of the mass to the center of the moon is $384,467 - r$.

To find the balance point, multiply each mass by how far it is from the center of mass:

$$\left(5.97219 \times 10^{24}\right) r = \left(7.347673 \times 10^{22}\right) (384,467 - r)$$

Solving for r:

$$r = \frac{4,730.15}{1 + 0.0123} = 4,673 \text{ km}$$

The point on the earth closest to this? It is where the moon is directly overhead. The it is $6,371 - 4,673 = 1,698$ km from the center of mass.

The point on the earth farthest from this? It is where the moon is directly underfoot. The it is $6,371 + 4,673 = 11,044$ km from the center of mass.

Answer to Exercise 6 (on page 54)

First, lets figure out ω. It travels through 2π radians in 27.3 days. 27.3 days = 2,358,720 seconds. $\omega = \frac{2\pi}{2{,}358{,}720} = 2.663811435 \times 10^{-6}$

$$F_c = (1)(11,044,000)(2.663811435 \times 10^{-6})^2 = 7.8365 \times 10^{-5}$$

Now the weakest:

$$F_c = (1)(1,698,000)(2.663811435 \times 10^{-6})^2 = 1.20512 \times 10^{-5}$$

Answer to Exercise 7 (on page 55)

Closest to the moon, the gravitational force of the moon and the centripetal forces are in the same direction: toward the moon.

$$F_{total} = 1.20488 \times 10^{-5} + 3.43045 \times 10^{-5} = 4.6356 \times 10^{-5} \text{ N}$$

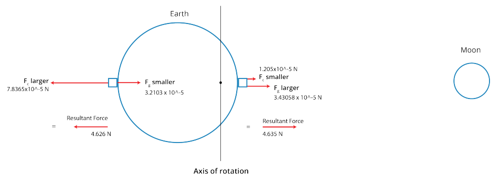
Farthest from the moon, the gravitational force of the moon and the centripetal forces are in opposite directions:

$$F_{total} = 7.8367 \times 10^{-5} - 3.2104 \times 10^{-5} = 4.62604^{-5}\text{N}$$

This is great conclusion: The two forces are basically equal: one pulls the water closest to the moon toward the moon, the other pulls water farthest from the moon away from the moon.

Both forces are pretty small: The force due to earth's gravity is about $211,000$ times more than either.

And that is why there are two basically equally large high tides every day.



Earth
Moon
F_c larger
7.8365x10^-5 N
F_g smaller
3.2103 x 10^-5
1.205x10^-5 N
F_c smaller
F_g larger
3.43058 x 10^-5 N
=
Resultant Force
4.626 N
=
Resultant Force
4.635 N
Axis of rotation

Index

barycenter, 48

escape velocity, 7

geosynchronous orbits, 6
gravitation, 17

Haversine formula, 38

latitude, 35
longitude, 35
Longitude and Latitude
    coordinates of, 35

moon
    index
        lunar, 46
moon
    eclipse
        solar, 46
    eclipses, 45
    phases of, 42

Nautical Mile, 38
nautical mile, 38

orbital paths, 41

rockets, 9
rockets
    alternate propulsion methods of, 15
    in atmosphere, 10
    in space, 13
    liquid fuel based, 10
    staging, 10

Three-Body Problem, 30
tides, 48