Project Lab Part 2 Plots:

Group Members:

- David Lim (79941274)
- Gia Jennifer Lee (18484543)

In [22]:
```
import numpy as np
import matplotlib.pyplot as plt

def grav_acc(s_x, s_y, planet_mass):
    """
    Find the instantaneous spacecraft acceleration at s_x, s_y away from the

    Inputs: s_x (meters), s_y (meters): the x, y components of a single posi
            planet_mass (kilograms): the mass of the planet

    Ouput: a_x (m/s^2), a_y (m/s^2) the x, y components of instantaneous spa

    """
    s = (s_x**2 + s_y**2)**0.5

    inst_a = (6.67e-11 * planet_mass)/(s**2)

    # Beta is angle between accleration vector and y dir.
    sin_beta = -s_x/s
    cos_beta = -s_y/s

    a_x = inst_a * sin_beta
    a_y = inst_a * cos_beta

    return (a_x, a_y)

def checkinit(s_x0, s_y0, v_x0, v_y0, planet_radius):
    """
    Checks if the starting position (origin at center of planet) of the spac
    is above the planetary surface, and the starting velocity is in the posi
    y-direction; raises a ValueError otherwise.

    Inputs: s_x0 (meters), s_y0 (meters): the initial x and y spacecraft pos
            v_x0 (m/s), v_y0 (m/s): the initial x and y spacecraft velocity

    Output: none
    """
    s_mag = (s_x0**2 + s_y0**2) ** 0.5

    # Check if spacecraft is above the planetary surface
    if (s_mag <= planet_radius):
        raise ValueError("Error: The spacecraft must start above the planeta

    # Check if spacecraft is fyling toward the planet
```

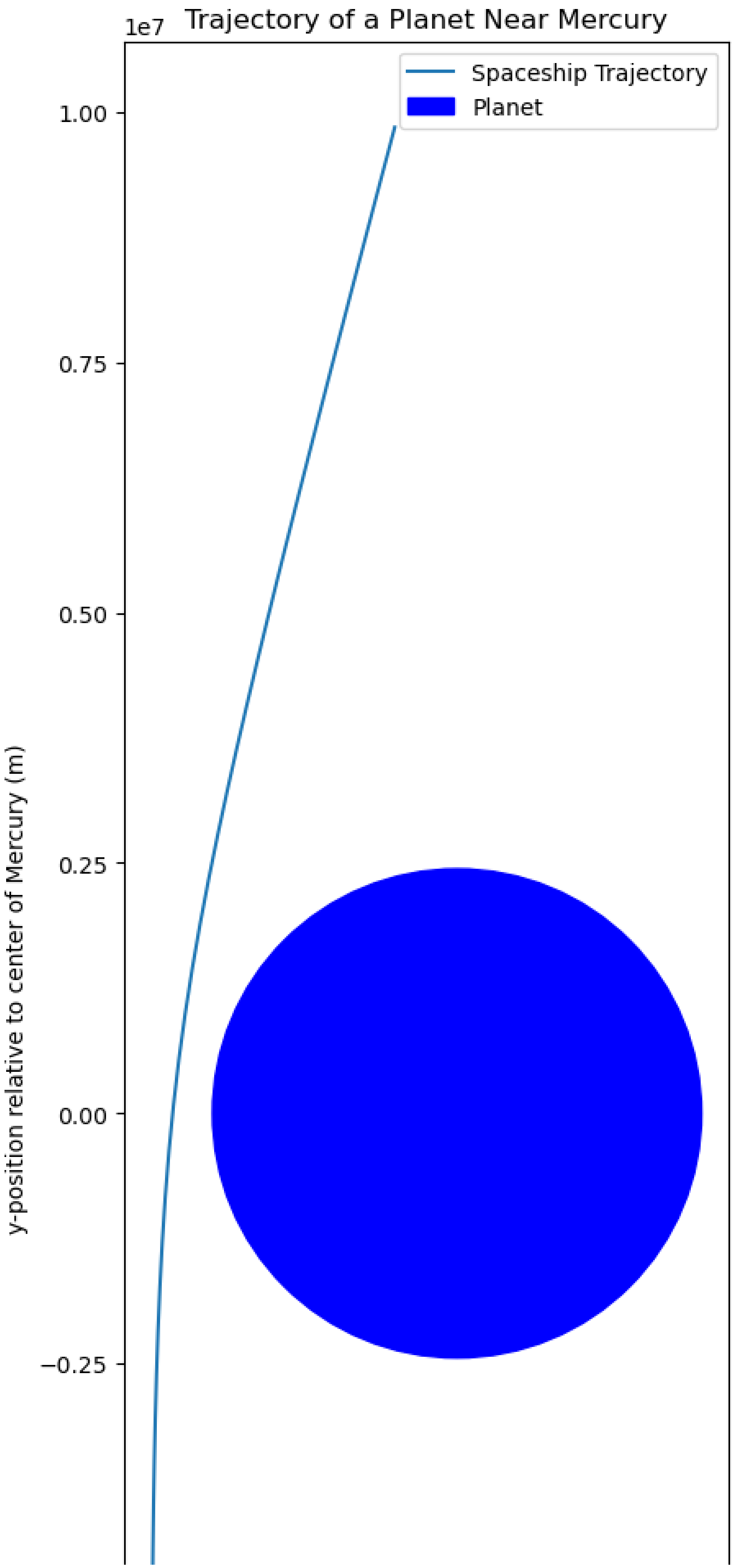
```
    if (v_y0 <= 0):
        raise ValueError("Error: The spacecraft must have a positive y-direc

def sc_vel_pos_change(a_x, a_y, v_x, v_y, time_step):
    """
    Computes the instantaneous (one time-step) change in position and veloci

    Inputs: a_x (m/s^2), a_y (m/s^2): instantaneous x, y components of accel
            v_x (m/s), v_y (m/s): instantaneous x, y components of velocity;
            time_step: the time increment, Δt in seconds

    Ouput: ds_x (meters), ds_y (meters): change in position vector x and y c
           dv_x (m/s), dv_y (m/s): change in velocity vector x and y compone
    """

    ds_x = (v_x * time_step) + (0.5 * a_x * time_step**2)
    ds_y = (v_y * time_step) + (0.5 * a_y * time_step**2)

    dv_x = a_x * time_step
    dv_y = a_y * time_step

    return ds_x, ds_y, dv_x, dv_y

def get_traj(s_x0, s_y0, v_x0, v_y0, time_step, total_time, planet_mass, pla
    """
    Computes a full spacecraft trajectory; calls grav_acc to calculate curre
    sc_vel_pos_change to find change in velocities at each iteration, and ch
    initial values are valid at the start of the function.

    Inputs: s_x0 (m/s), s_y0 (m/s): spacecraft's initial position x, y compo
            v_x0 (m/s), v_y0 (m/s): spacecraft's initial velocity x, y compo
            time_step (sec): floating-point or integer time increment, Δt
                       at which spacecraft acceleration, velocity, positions
            total_time (sec): floating-point or integer total-time value for
            planet_mass (kg): floating-point or integer mass of fly-by targe
                          (default = 1 Mercury mass) in kg.
            planet_radius (m): floating-point or integer radius of fly-by ta
                            (default = 1 Mercury radius) in m.

    Outputs: time: an array of times with shape (nt, ).
             acc: (acc_x, acc_y), an array of spacecraft accelerations with
             vel: (vel_x, vel_y), an array of spacecraft velocities with sha
             pos: (pos_x, pos_y), an array of spacecraft positions with shap

    """
    # Check for errors before starting
    checkinit(s_x0, s_y0, v_x0, v_y0, planet_radius)

    # Number of iterations (including time t = 0)
    nt = total_time // time_step + 1 # including 0

    # Initilializing arrays:
    time = np.arange(0, (total_time + 1), time_step)
    acc = np.full((nt, 2), np.nan)
    vel = np.full((nt, 2), np.nan)
    pos = np.full((nt, 2), np.nan)
```

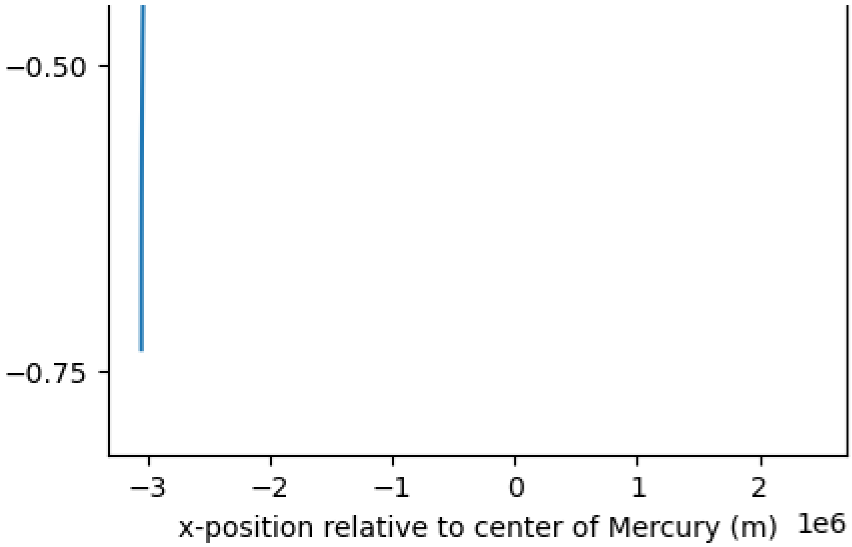
```python
    # Inititialize conditions
    s_x = s_x0
    s_y = s_y0
    v_x = v_x0
    v_y = v_y0

    # "old" refers to previous iteration from where we are calculating our r
    for ii in range(nt):
        # Add previously calculated values into arrays
        a_x, a_y = grav_acc(s_x, s_y, planet_mass) # calculate acceleration

        acc[ii, 0] = a_x
        acc[ii, 1] = a_y
        pos[ii, 0] = s_x
        pos[ii, 1] = s_y
        vel[ii, 0] = v_x
        vel[ii, 1] = v_y

        # Calculate change in position and velocity at old position to new
        ds_x, ds_y, dv_x, dv_y = sc_vel_pos_change(a_x, a_y, v_x, v_y, time_

        # Calculate new position and velocity by adding changes to old posit
        s_x = s_x + ds_x
        s_y = s_y + ds_y
        v_x = v_x + dv_x
        v_y = v_y + dv_y

    return time, acc, vel, pos
```

In [145…

```python
# Calling the Function

merc_mass = 3e23 # kg
merc_radius = 2440 * 1000 # meters

s_x0 = -3050 * 1000 # meters
s_y0 = -3 * merc_radius
v_x0 = 0 # m/s
v_y0 = 7 * 1000 # m/s
time_step = 60 # seconds
total_time = 40 * 60 # seconds
planet_mass = merc_mass # kg
planet_radius = merc_radius # meters

time, acc, vel, pos = get_traj(s_x0, s_y0, v_x0, v_y0, time_step, total_time

# Figure 1:

fig, ax = plt.subplots(figsize = (10, 15))
ax.plot(pos[:, 0], pos[:, 1], label='Spaceship Trajectory')
circle = plt.Circle((0, 0), planet_radius, color='b', label='Planet')
ax.add_patch(circle)
ax.set_aspect('equal', adjustable='box')

# Set labels and legend
ax.set_xlabel('x-position relative to center of Mercury (m)')
```

```
ax.set_ylabel('y-position relative to center of Mercury (m)')
ax.set_title('Trajectory of a Planet Near Mercury')
ax.legend()

# Show the plot
plt.show()


# Figure 2

# Create |a| array
n = total_time // time_step + 1 # add one to include 0 and 41 (last number)
acc_mag = np.full((n,), np.nan)
for ii in range(n):
    acc_mag[ii] = (acc[ii, 0]**2 + acc[ii, 1]**2) ** 0.5

# Create speed array
speed = np.full((n,), np.nan)
for ii in range(n):
    speed[ii] = (vel[ii, 0]**2 + vel[ii, 1]**2) ** 0.5

# Create suplots:
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize = (10, 10))
fig.tight_layout(pad=5.0)

# Subplot 1
ax1.plot(time, acc[:,0])
ax1.plot(time, acc[:,1])
ax1.plot(time, acc_mag)
ax1.set_xlabel('Time (seconds)')
ax1.set_ylabel('Acceleration (m/s^2)')
ax1.set_title('Change in Gravitational Acceleration Over Time Near Mercury')
ax1.legend(['x-acceleration', 'y-acceleration', 'magnitude of accleration'])
ax1.grid()

# Subplot 2
ax2.plot(time, speed)
ax2.set_xlabel('Time (seconds)')
ax2.set_ylabel('Speed (m/s)')
ax2.set_title('Change in Speed Over Time Near Mercury')
ax2.grid()

# Subplot 3
ax3.plot(time, np.abs(pos[:, 1]))
ax3.set_xlabel('Time (seconds)')
ax3.set_ylabel('Alititude Relative to Planet (meters)')
ax3.set_title('Change in Alititude Over Time')
```


Trajectory of a Planet Near Mercury
1e7
Spaceship Trajectory
Planet
1.00
0.75
0.50
0.25
0.00
−0.25
y-position relative to center of Mercury (m)



Out[145… Text(0.5, 1.0, 'Change in Alititude Over Time')