# Spacecraft Mission

Alex Freeman

April 18, 2025

This report will cover all parts of assignment one in the Aerospace 720 course

# 1 Orbital Propagation

## 1.1 Solving Kepler's Equation

**I)** We need to solve Kepler's Equation using numerical methods. Using the Newton-Rasphon Method we can take the eccentricity and mean anamoly as inputs and numerically solve for the eccentric anamoly.

```
def Kepler(e, M, tol = 1e-12, max_i = 1000):
    E = M                              # Guess solution
    for i in range(max_i):
        f_E = E - e * np.sin(E) - M  # Define the function in terms of f(E) = 0
         f_prime = 1 - e * np.cos(E) # Derive the function in terms of E
        del_E = f_E / f_prime
        E_new = E - del_E            # Calculate the new eccentric anamoly
        if np.abs(del_E) < tol:      # If the value is within the set tolerance
            theta = 2*np.arctan(np.tan(E_new/2) * ((1+e)/(1-e))**(0.5))
            return theta             # Return true anamoly
        E = E_new
```

**II)** If we set the tolerance to $1e-12$, we can compute the true anamoly of the asteroid at $t_0$ and $t_0 + 100$ days. A_ae0 is the OBJ data of the asteroid, it is an array.

```
trueAnamoly_asteroidt_0 = Kepler(A_ae0[2], A_ae0[6])
meanAnamolyt_100 = get_mean_anamoly(100*(3600*24), A_ae0[6], A_ae0[1])
trueAnamoly_asteroidt_100 = Kepler(A_ae0[2], meanAnamolyt_100)
```

Printing these values gives that the true anamoly $\theta_{t_0} = 1.4246$ and $\theta_{t_0+100} = 2.1369$. Where these answers are in radians.

**III)** Now I have created a function that takes in a state of orbital elements and returns the position and velocity vectors at that point. It uses a rotation matrix to convert from the perifocal frame to the ECI frame. This is defined via $i, \omega$, and $\Omega$ terms and is calculated using the defind matricies in the appendix.

```
def COE2RV(arr, mu):
    a, e, i, Omega, omega, theta_var = arr[0:6]
```

```
    h = np.sqrt(mu * a * (1 - e**2))
    r = a*(1-(e**2))/(1 + e*np.cos(theta_var))

    arr_r = np.array([r*np.cos(theta_var), r*np.sin(theta_var), 0])
    arr_v = (mu/h)* np.array([-np.sin(theta_var), e + np.cos(theta_var), 0])

    # Rotate position and velocity from perifocal to inertial frame using the
    # transfomration matrix
    R_matrix = rotation_matrix(i, Omega, omega)

    r_ijk = R_matrix @ arr_r
    v_ijk = R_matrix @ arr_v
    return r_ijk, v_ijk
```

Using this code we can output the state vector at some time $t$. The first three values are the $x, y, z$ positions in **km**. The last three are the velocity values in the $x, y, z$ direction in **km/s**.

**At** $t_0$:

$$\bar{\mathbf{X}} = \begin{bmatrix} x = -1.1694365\text{e+}08 \\ y = 1.53462780\text{e+}08 \\ z = -6.7446087\text{e+}06 \\ v_x = -3.1710203\text{e+}01 \\ v_y = -3.6285380\text{e+}00 \\ v_z = -1.8931546\text{e+}00 \end{bmatrix}$$

**At** $t_0 + 100$:

$$\bar{\mathbf{X}} = \begin{bmatrix} x = -3.2057997\text{e+}08 \\ y = 6.72659396\text{e+}07 \\ z = -1.8991445\text{e+}07 \\ v_x = -1.6964807\text{e+}01 \\ v_y = -1.2943780\text{e+}01 \\ v_z = -1.0284663\text{e+}00 \end{bmatrix}$$

**IV)** Next, I have written a function called "Ephemeris". It returns the position and velocity at some time t.

```
def Ephemeris(t, OBJdata, mu):

    time, a, e, i, Omega, omega, mean_anamoly = OBJdata[0:7]
    nu_t = (mu / (a**3))**0.5
    mean_anamoly_t = mean_anamoly + nu_t * (t)
    h = np.sqrt(mu * a * (1 - e**2))
    theta_var = Kepler(e, mean_anamoly_t)
    r = a*(1-(e**2))/(1 + e*np.cos(theta_var))

    arr_r = np.array([r*np.cos(theta_var), r*np.sin(theta_var), 0])
    arr_v = (mu/h)* np.array([-np.sin(theta_var), e + np.cos(theta_var), 0])

    R_matrix = rotation_matrix(i, Omega, omega)
    r_ijk = R_matrix @ arr_r
    v_ijk = R_matrix @ arr_v
    return r_ijk, v_ijk
```

## 1.2 Numerical Integration

To derive the necessary state function we have:

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} \tag{1}$$

From here we know that $\frac{dv}{dt} = \dot{r}$ giving:

$$\frac{d\mathbf{v}}{dt} = -\frac{\mu}{r^3}\mathbf{r} \tag{2}$$

Expanding each vector as three-dimensional components in $x, y, z$:

$$\frac{dx}{dt} = v_x, \frac{dy}{dt} = v_y, \frac{dz}{dt} = v_z \tag{3}$$

$$\frac{dv_x}{dt} = -\frac{\mu}{r^3}x, \frac{dv_y}{dt} = -\frac{\mu}{r^3}y, \frac{dv_z}{dt} = -\frac{\mu}{r^3}z \tag{4}$$

Where $r = \sqrt{x^2 + y^2 + z^2}$
We can now define a state vector $\bar{\mathbf{X}}$

$$\bar{\mathbf{X}} = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{5}$$

Finally, deriving this state vector gives the following:

$$\dot{\bar{\mathbf{X}}} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ -\frac{\mu}{r^3}x \\ -\frac{\mu}{r^3}y \\ -\frac{\mu}{r^3}z \end{bmatrix} \tag{6}$$

## 2 Figures

Figure example:

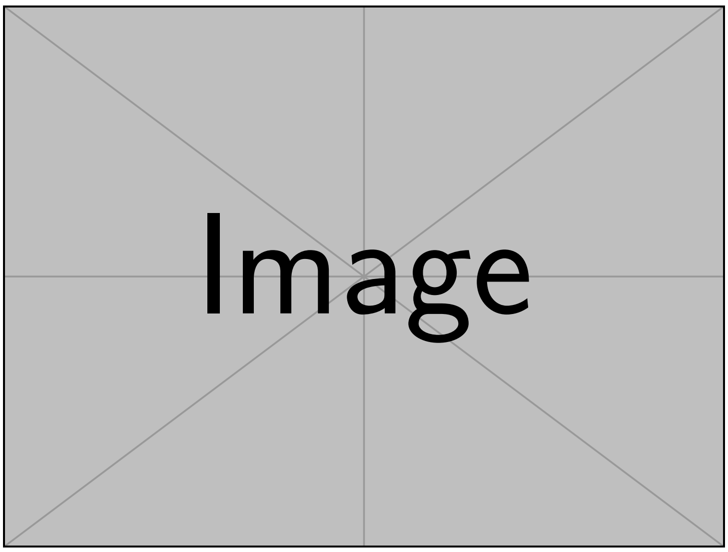


Figure 1: Example figure caption.

## 3 Tables

| Left | Center | Right |
|:---|:---:|---:|
| A | B | C |
| 1 | 2 | 3 |

Table 1: Example table.