script

August 6, 2021

[1]:
```
import numpy as np
import sympy as sym
from numpy import cos, sin, sqrt, radians, degrees, pi, arcsin,arccos, tan
from numpy.linalg import norm
from scipy.optimize import fsolve
```

0.1 Lambert's Problem

[2]:
```
def Lambert(r1, r2, theta, t):
    """
    r1: initial position, in AU
    r2: final position, in AU
    theta: transfer angle, in degrees
    t: time of flight, in TU
    returns a, e
    """
    theta = radians(theta)

    # Calculating the chord and semiperimeter
    c = sqrt(r1**2 + r2**2 - 2*r1*r2*cos(theta))
    s = (r1 + r2 + c)/2
    betam = 2*arcsin(sqrt((s-c)/s))

    # Calculate the minimum flight time possible, which is the parabolic␣
↪trajectory, tp
    tp = sqrt(2)/3 * (s**1.5 - np.sign(sin(theta))*(s - c)**1.5)

    # Calculate the minimum energy time, tm

    tm = sqrt(s**3/8) * (pi - betam + sin(betam))

    # Check if the given time of flight is greater than the parabolic time of␣
↪flight
    if t < tp:
        return f'Time of flight not possible with a Lambert trajectory. Choose␣
↪a time greater than {tp} TU'
    elif t > tp:
```

```python
    # Create the function that solves for the time of flight
    def TOF(a):
        alpha0 = 2*arcsin(sqrt(s/(2*a)))
        beta0 = 2*arcsin(sqrt((s-c)/(2*a)))

        # Check the cases from figure 5.7
        if np.degrees(theta) < 180 or np.degrees(theta) == 180:
            beta = beta0
        elif np.degrees(theta) > 180:
            beta = - beta0

        if t < tm:
            alpha = alpha0
        elif t > tm:
            alpha = 2*pi - alpha0

        return t - a**1.5 *(alpha - beta - (sin(alpha) - sin(beta)))

    # Modify the initial guess depending on the mission
    a = fsolve(TOF, 1.5)

    alpha0 = 2*arcsin(sqrt(s/(2*a)))
    beta0 = 2*arcsin(sqrt((s-c)/(2*a)))
    if np.degrees(theta) < 180 or np.degrees(theta) == 180:
        beta = beta0
    elif np.degrees(theta) > 180:
        beta = - beta0

    if t < tm:
        alpha = alpha0
    elif t > tm:
        alpha = 2*pi - alpha0

    term = (4*(s - r1)*(s - r2))/c**2 * (sin((alpha + beta)/2))**2

    # Eccentricity
    e = sqrt(1 - term)

    A = sqrt(1/(4*a)) * 1/tan(alpha/2)
    B = sqrt(1/(4*a)) * 1/tan(beta/2)

    # Assuming departure occurs at periapsis
    u1 = np.array([1, 0])
    u2 = np.array([cos(theta), sin(theta)])

    # Using the law of sines to calculate the angle between the space fixed␣
↪i and u_c
```

```
        theta_c = arcsin(sin(radians(theta))/c * r2)

        uc = np.array([cos(np.pi - theta_c), sin(np.pi - theta_c)])

        v1 = (B + A)*uc + (B - A)*u1
        v2 = (B + A)*uc - (B - A)*u2

        return a, e, v1, v2
```

0.2 Case 1: Earth to Mars through a transfer angle of 75°

$t_f = 1y = 2\pi$ TU

[3]:
```
r1 = 1
r2 = 1.524
theta = 75
t = 2*np.pi

[a, e, v1, v2] = Lambert(r1, r2, theta, t)

a = a[0]
e = e[0]

print(f'Semimajor axis: {np.round(a, 4)} AU \n')
print(f'Eccentricity: {np.round(e, 4)} \n')
print(f'Departure velocity: {np.round(v1, 4)} AU/TU \n')
print(f'Arrival velocity: {np.round(v2, 4)} AU/TU \n')

# Assuming circular orbit:

dv1 = norm(v1) - sqrt(1/r1)
dv2 = sqrt(1/r2) - norm(v2)

print(f'Delta v1: {np.round(dv1, 4)} AU/TU \n')
print(f'Delta v2: {np.round(dv2, 4)} AU/TU')
```

```
Semimajor axis: 1.2311 AU

Eccentricity: 0.7918

Departure velocity: [0.3996 0.016 ] AU/TU

Arrival velocity: [-1.0257 -1.0776] AU/TU

Delta v1: -0.6 AU/TU

Delta v2: -0.6777 AU/TU
```

0.3 Case 2: Earth to Mars through a transfer angle of 75°

$t_f = 115d = 1.97963$ TU

[4]:
```
r1 = 1
r2 = 1.524
theta = 75
t = 1.97963

[a, e, v1, v2] = Lambert(r1, r2, theta, t)

a = a[0]
e = e[0]

print(f'Semimajor axis: {np.round(a, 4)} AU \n')
print(f'Eccentricity: {np.round(e, 4)} \n')
print(f'Departure velocity: {np.round(v1, 4)} AU/TU \n')
print(f'Arrival velocity: {np.round(v2, 4)} AU/TU \n')

# Assuming circular orbit:

dv1 = norm(v1) - sqrt(1/r1)
dv2 = sqrt(1/r2) - norm(v2)

print(f'Delta v1: {np.round(dv1, 4)} AU/TU \n')
print(f'Delta v2: {np.round(dv2, 4)} AU/TU')
```

```
Semimajor axis: 1.2312 AU

Eccentricity: 0.3306

Departure velocity: [-0.3993  0.0248] AU/TU

Arrival velocity: [-1.3217 -0.683 ] AU/TU

Delta v1: -0.5999 AU/TU

Delta v2: -0.6777 AU/TU
```

0.4 Case 3: Earth to Venus through a transfer angle of 135°

$t_f = 337.6d = 5.8115$ TU

[5]:
```
r1 = 1
r2 = 0.723
theta = 135
t = 5.8115

[a, e, v1, v2] = Lambert(r1, r2, theta, t)
```

```python
a = a[0]
e = e[0]

print(f'Semimajor axis: {np.round(a, 4)} AU \n')
print(f'Eccentricity: {np.round(e, 4)} \n')
print(f'Departure velocity: {np.round(v1, 4)} AU/TU \n')
print(f'Arrival velocity: {np.round(v2, 4)} AU/TU \n')

# Assuming circular orbit:

dv1 = norm(v1) - sqrt(1/r1)
dv2 = sqrt(1/r2) - norm(v2)

print(f'Delta v1: {np.round(dv1, 4)} AU/TU \n')
print(f'Delta v2: {np.round(dv2, 4)} AU/TU')
```

```
Semimajor axis: 1.1004 AU

Eccentricity: 0.6506

Departure velocity: [0.5451 0.0463] AU/TU

Arrival velocity: [-0.3426 -2.0966] AU/TU

Delta v1: -0.4529 AU/TU

Delta v2: -0.9484 AU/TU
```