Lab10

Imports

In [1]:
```
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.graph_objects as go
from dash.dependencies import Input, Output
import plotly.express as px
import numpy as np
import pandas as pd
```

Derivative

In [2]:
```
def derivative(r, i, j, dims):
    global G, M
    d = 0
    for k in range(3):  # which object
        if k != i:
            distance = 0
            for coordinate in range(dims):
                distance += (r[k][coordinate] - r[i][coordinate]) ** 2
            distance = np.sqrt(distance) ** 3
            d += G * M[k] * (r[k][j] - r[i][j]) / distance
    return d
```

Function

In [3]:
```
def f(d_1, d_2, h):
    global G, M
    m = d_1.shape[0]
    n = d_1.shape[1]
    r = d_1.copy()  # Update r
    d1 = d_2.copy()  # Update d1

    d2 = np.zeros((m, n))
    # Update d2
    for i in range(m):
        for j in range(n):
            d2[i][j] = derivative(r, i, j, n)

    return h * d1, h * d2
```

Get model frames

In [4]:
```
def animate(n,h):
    global r, d1, radius1, radius2, radius3
    global df


    row1 = np.zeros((6, 1))
    row2 = np.zeros((6, 1))
    row3 = np.zeros((6, 1))
    for i in range(1, n):
        k1d1, k1d2 = f(r, d1, h)
        k2d1, k2d2 = f(r + 0.5 * k1d1, d1 + 0.5 * k1d2, h)
        k3d1, k3d2 = f(r + 0.5 * k2d1, d1 + 0.5 * k2d2, h)
        k4d1, k4d2 = f(r + k3d1, d1 + k3d2, h)

        row1[0] = r[0][0]
        row1[1] = r[0][1]
        row1[2] = r[0][2]
        row1[3] = 1
        row1[4] = radius1
        row1[5] = i


        row2[0] = r[1][0]
        row2[1] = r[1][1]
        row2[2] = r[1][2]
        row2[3] = 2
        row2[4] = radius2
        row2[5] = i

        row3[0] = r[2][0]
```

```
        row3[1] = r[2][1]
        row3[2] = r[2][2]
        row3[3] = 3
        row3[4] = radius3
        row3[5] = i

        row_df1 = pd.DataFrame(row1.T)
        row_df2 = pd.DataFrame(row2.T)
        row_df3 = pd.DataFrame(row3.T)

        df = pd.concat([df, row_df1])
        df = pd.concat([df, row_df2])
        df = pd.concat([df, row_df3])

        r += (k1d1 + 2. * k2d1 + 2. * k3d1 + k4d1) / 6.
        d1 += (k1d2 + 2. * k2d2 + 2. * k3d2 + k4d2) / 6.
```

Model settings

In [5]:
```
m1 = 50
m2 = 0.4
m3 = 0.4
radius1 = 50
radius2= 20
radius3 = 20
r1 = np.array([0., 0., 0.], dtype=np.float64)
r2 = np.array([0., 1., 0.], dtype=np.float64)
r3 = np.array([0., -1., 0.], dtype=np.float64)
v1 = np.array([5., 0., 0.], dtype=np.float64)
v2 = np.array([5., 2.5, -2.5], dtype=np.float64)
v3 = np.array([5., -2.5, 2.5], dtype=np.float64)
G = 0.4
M = np.array([m1, m2, m3])
r = np.array([r1, r2, r3])
d1 = np.array([v1, v2, v3])

df = pd.DataFrame()
animate(1000, 0.005)
df.columns = ["X", "Y", "Z", "Object","Radius", "T"]
df
```

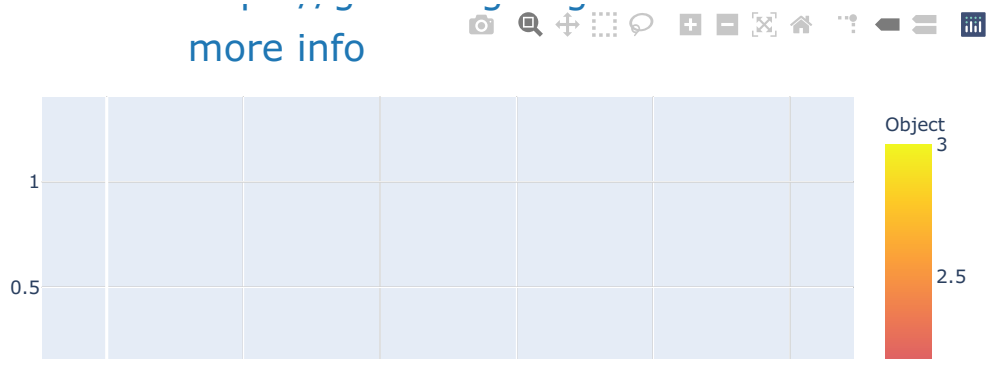
Out[5]:

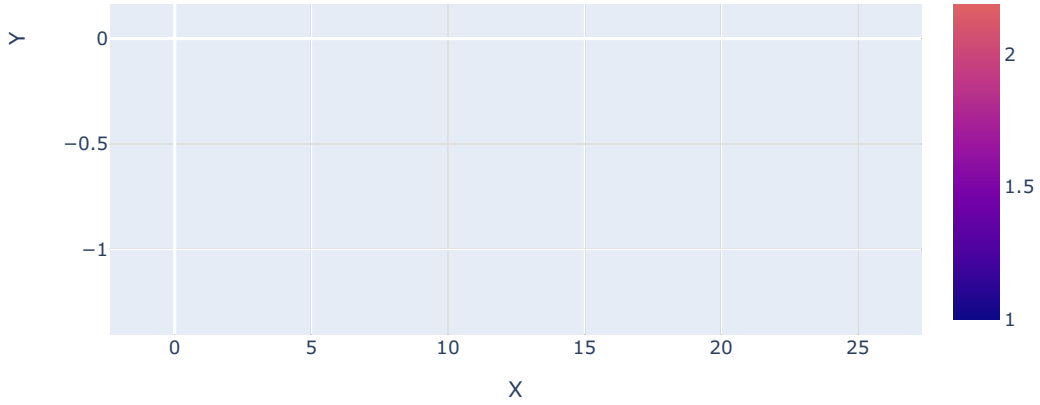
	X	Y	Z	Object	Radius	T
0	0.000	0.000000	0.000000	1.0	50.0	1.0
0	0.000	1.000000	0.000000	2.0	20.0	1.0
0	0.000	-1.000000	0.000000	3.0	20.0	1.0
0	0.025	0.000000	0.000000	1.0	50.0	2.0
0	0.025	1.012252	-0.012499	2.0	20.0	2.0
...	...	...	...	...	...	...
0	24.925	-0.188008	-0.105952	2.0	20.0	998.0
0	24.925	0.188008	0.105952	3.0	20.0	998.0
0	24.950	0.000000	0.000000	1.0	50.0	999.0
0	24.950	-0.189856	-0.041177	2.0	20.0	999.0
0	24.950	0.189856	0.041177	3.0	20.0	999.0

2997 rows × 6 columns

X && Y

In [6]:
```
px.scatter(df, x="X", y="Y", color="Object", size="Radius")
```





X && Z

In [7]:
```
px.scatter(df, x="X", y="Z", color="Object", size="Radius")
```

Y && Z

In [8]:
```
px.scatter(df, x="Y", y="Z", color="Object", size="Radius")
```

Y && Z Animation

In [9]:
```
px.scatter(df, x="Y", y="Z", animation_frame="T", animation_group="Object",size="Radius", color="Object", hover_r
```

All traces

In [10]:
```
fig = px.scatter_3d(df, x="X", y="Y", z="Z",
                    color="Object", size="Radius")
fig.show()
```

Animation

In [ ]:

```
fig = go.Figure(
    data=[go.Scatter3d(x=[], y=[], z=[], mode="markers")]

)

fig.update_layout(
        scene = dict(
        xaxis=dict(range=[min(df.iloc[:, 0]), max(df.iloc[:, 0])], autorange=False),
        yaxis=dict(range=[min(df.iloc[:, 1]), max(df.iloc[:, 1])], autorange=False),
        zaxis=dict(range=[min(df.iloc[:, 2]), max(df.iloc[:, 2])], autorange=False),
        )),

frames = [go.Frame(data= [go.Scatter3d(x=df.iloc[:i+3, 0], y=df.iloc[:i+3, 1], z=df.iloc[:i+3, 2])],
                  ) for i in  range(0, df.shape[0], 3)]
fig.update(frames=frames)

fig.update_layout(updatemenus=[dict(type="buttons",
                           buttons=[dict(label="Play",
                                         method="animate",
                                         args=[None, dict(frame=dict(redraw=True,fromcurrent=True, mode='immediate
app = dash.Dash()
app.layout = html.Div([
    dcc.Graph(figure=fig)
])

app.run_server(debug=True, use_reloader=False)
```

```
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
```