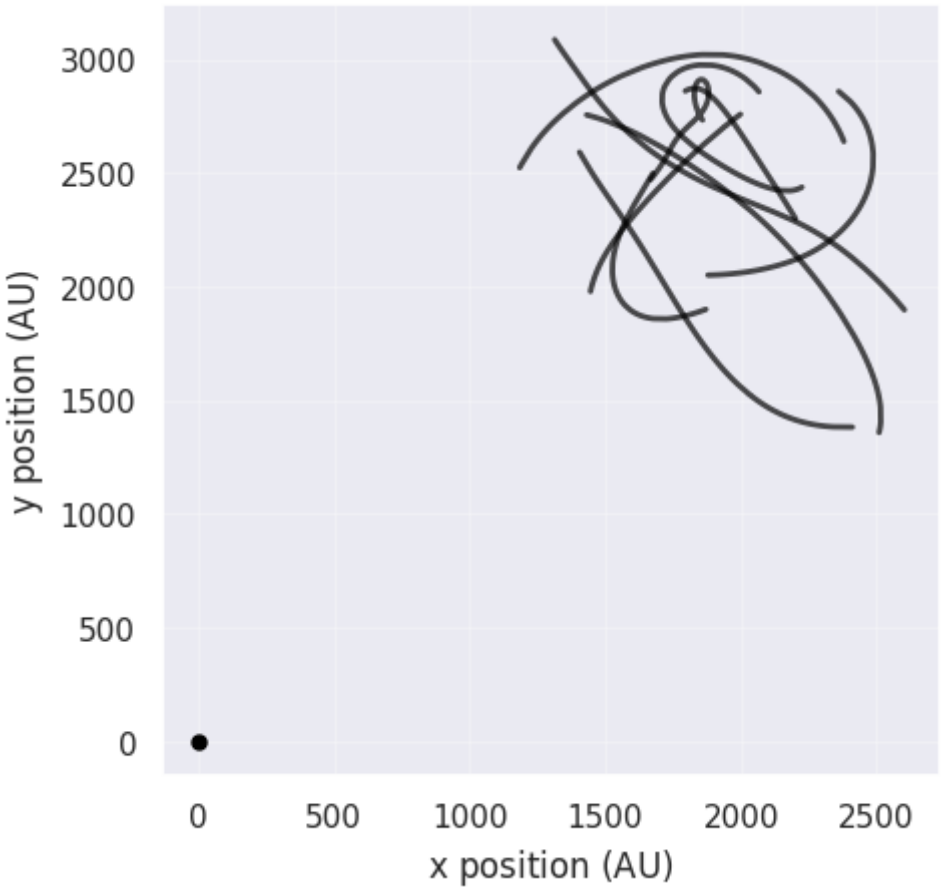
In [ ]:
```
# import numpy as np
# import pandas as pd
# df = pd.read_csv('cmake-build-debug/outputCSV.csv', header=[0,1])
# plt.plot(0, 0, marker = 'o', markersize=20, color="black")
# plt.xlim(-1.1, 1.1)
# plt.ylim(-1.1, 1.1)
```

In [ ]:
```
from scipy.optimize import curve_fit
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
import pandas as pd

%matplotlib inline


df = pd.read_csv('cmake-build-debug/outputCSV.csv', header=[0,1])
idx = pd.IndexSlice

def doPlot2D(x, y, width, height, colour, replot: bool):


    # plt.style.use('seaborn-v0_8-deep')

    fig, ax = plt.subplots()
    fig.set_size_inches(width, height)

    plt.plot(x, y, alpha=0.7, color=colour)

    # ax.set_xlim(xlim[0], xlim[1])
    # ax.set_ylim(ylim[0], ylim[1])

    # plt.title(f"T = {df.Iteration.iat[-1]}")
    plt.grid(alpha=0.3)

# https://jakevdp.github.io/PythonDataScienceHandbook/04.12-three-dimensi
def doPlot3D(x, y, z):

    # plt.style.use('seaborn-v0_8-deep')

    fig = plt.figure()
    ax = plt.axes(projection='3d')

    ax.scatter3D(0, 0, 0, s=50, c="black")
    ax.plot3D(x, y, z, c="green")
```

Simple 2D x,y position

In [ ]:
```
fig, ax = plt.subplots()
# fig.set_size_inches(10, 10)
fig.set_size_inches(5,5)
plt.style.use('bmh')
plt.plot(0, 0, marker = 'o', markersize=5, color="black") # sun plot
# plt.xlim(-5, 5)
# plt.ylim(-5, 5)

for i in range(0, int(df.shape[1]/10)):
    x = df[f'{i}']['x']
    y = df[f'{i}']['y']

    plt.plot(x, y, alpha=0.7, color="black")

plt.grid(alpha=0.3)
plt.xlabel("x position $(\mathrm{AU})$")
plt.ylabel("y position $(\mathrm{AU})$")
# plt.title(f"T = {df.shape[0]-1}")

print(df.shape)
```

```
(100001, 100)
```



In [ ]:
```
## New Method
x = df.loc[:,idx[:,'x']]
y = df.loc[:,idx[:,'y']]
z = df.loc[:,idx[:,'z']]

fx = df.loc[:,idx[:,'fx']]
fy = df.loc[:,idx[:,'fy']]
fz = df.loc[:,idx[:,'fz']]

vx = df.loc[:,idx[:,'vx']]
vy = df.loc[:,idx[:,'vy']]
vz = df.loc[:,idx[:,'vz']]

## Old Method
mass = df[f'{i}']['M']
x = df[f'{i}']['x']
y = df[f'{i}']['y']
z = df[f'{i}']['z']
vx = df[f'{i}']['vx']
vy = df[f'{i}']['vy']
vz = df[f'{i}']['vz']
fx = df[f'{i}']['fx']
fy = df[f'{i}']['fy']
fz = df[f'{i}']['fz']
```

Energy of System

Gravitational Potential Energy

because of the way the forces are calculated, each pair-wise force appears twice. to account for this, forces must be halved =>

GPE -> force * 0.5

Kinetic Energy

KE = 1/2 * mass * velocity^2

In [ ]:
```
# USE THIS INSTEAD OF FOR LOOPS, MUCH MUCH FASTER
# idx = pd.IndexSlice ->> moved to top level
test3 = df.loc[:,idx[:,'fx']]
test4 = df.cumsum(axis=0)['0']['fx']
```

Calculating $E_{p,0}$ [*initial total gravitational potential energy*]

In [ ]:
```
# GPE => F_total * 0.5
# this loops through all bodies to calculate GPE_0 for system
gravPotEnergy0 = 0.0;
for i in range(0, int(df.shape[1]/10)):
    fx = df[f'{i}']['fx'][0]
    fy = df[f'{i}']['fy'][0]
    fz = df[f'{i}']['fz'][0]

    gravPotEnergy0 += 0.5 * np.sqrt(fx*fx + fy*fy + fz*fz)
print(gravPotEnergy0)
```

```
0.0
```

Calculating $E_{k,0}$ [*initial total kinetic energy*]

In [ ]:
```
# KE => 0.5 * mass * velocity^2
# this loops through all bodies to calculate KE_0 for system
kineticEnergy0 = 0.0;
for i in range(0, int(df.shape[1]/10)):
    vx   = df[f'{i}']['vx'][0]
    vy   = df[f'{i}']['vy'][0]
    vz   = df[f'{i}']['vz'][0]
    mass = df[f'{i}']['M' ][0]

    velocity = np.sqrt(vx**2 + vy**2 + vz**2)

    kineticEnergy0 += 0.5 * mass * velocity**2
print(kineticEnergy0)
```

```
1.6282695500465238e-05
```

Calculating $E_{total,0}$ [*initial total energy*]

In [ ]:
```
# Units are as follows:
# GPE ->> AU/days^2
# KE  ->> ( Solar Masses * AU^2 ) / days^2
totalE0 = gravPotEnergy0 + kineticEnergy0
print(totalE0)
```

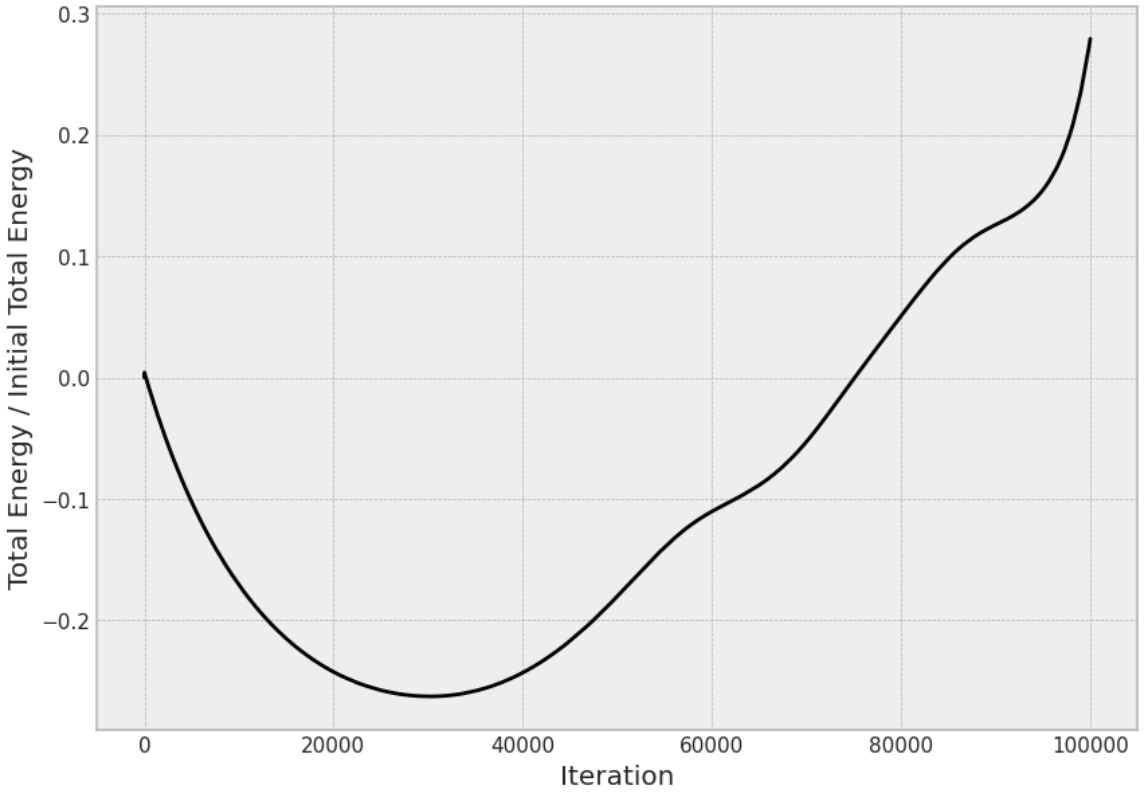
```
1.6282695500465238e-05
```

Plotting E_{total} percentage difference over time

In [ ]:
```
# Pandas is painful
x = df.loc[:,idx[:,'x']]
y = df.loc[:,idx[:,'y']]
z = df.loc[:,idx[:,'z']]

fx = df.loc[:,idx[:,'fx']]
fy = df.loc[:,idx[:,'fy']]
fz = df.loc[:,idx[:,'fz']]

vx = df.loc[:,idx[:,'vx']]
vy = df.loc[:,idx[:,'vy']]
vz = df.loc[:,idx[:,'vz']]

mass = df.loc[:,idx[:,'M']]
mass = mass.groupby(level=0, axis=1).sum()

fx_sum = fx.cumsum(axis=1)
fy_sum = fy.cumsum(axis=1)
fz_sum = fz.cumsum(axis=1)

f_sum = np.sqrt(fx_sum.iloc[:,-1]**2 + fy_sum.iloc[:,-1]**2 + fz_sum.iloc
# f_sum = fx_sum.iloc[:,-1] + fy_sum.iloc[:,-1] + fz_sum.iloc[:,-1]
# print(f_sum)

vxSq = vx**2
vySq = vy**2
vzSq = vz**2

vxSq, vySq = vxSq.align(vySq, fill_value=0)
vxSq, vzSq = vxSq.align(vzSq, fill_value=0)
vxSq, vySq = vxSq.align(vySq, fill_value=0)

vSqr = vxSq + vySq + vzSq
# print(vSqr)
vMag = np.sqrt(vSqr.groupby(level=0, axis=1).sum())
# print(vMag)
dfKE = 0.5 * mass * vMag**2
# print(dfKE)
# print(f_sum)
kineticEnergy = dfKE.cumsum(axis=1).iloc[:,-1]
gravPotEnergy = 0.5 * (f_sum)
totalEnergy   = kineticEnergy + (gravPotEnergy)

# print(totalEnergy)
# print(gravPotEnergy)
# print(kineticEnergy)
# print(vx)
```

In [ ]:
```
# Create plot environment
plt.style.use('bmh')
fig, ax = plt.subplots()
fig.set_size_inches(10, 7)

# Variable setting
iterations = np.arange(0, df.shape[0]);

x_data = iterations
y_data = ((totalEnergy - totalEnergy[0]) / totalEnergy[0])

# Plot graph
plt.plot(x_data, y_data, color="black", label="$\mathrm{E}_{\mathrm{total
# plt.title("Total Energy Change")
plt.xlabel("Iteration");
plt.ylabel("Total Energy / Initial Total Energy")


# Curve fit objective function
def func(x, a, b):
    return a * x + b

# Curve fit
popt, cov = curve_fit(func, x_data, y_data)

# Summarise parameter values
a, b = popt
print('y = %.5ex + %.5f' % (a, b))

# Calculate output for range
y_line = func(x_data, a, b)

# Plot curve fit line
# plt.plot(x_data, y_line, linestyle='--', label="Curve Fit")

# plt.legend(fontsize=12)
plt.show()

print(y_data[0])
print(totalEnergy[0])
print((y_data.sum())/(totalEnergy[0] * df.shape[0]))
```

```
y = 3.86312e-06x + -0.29317
```

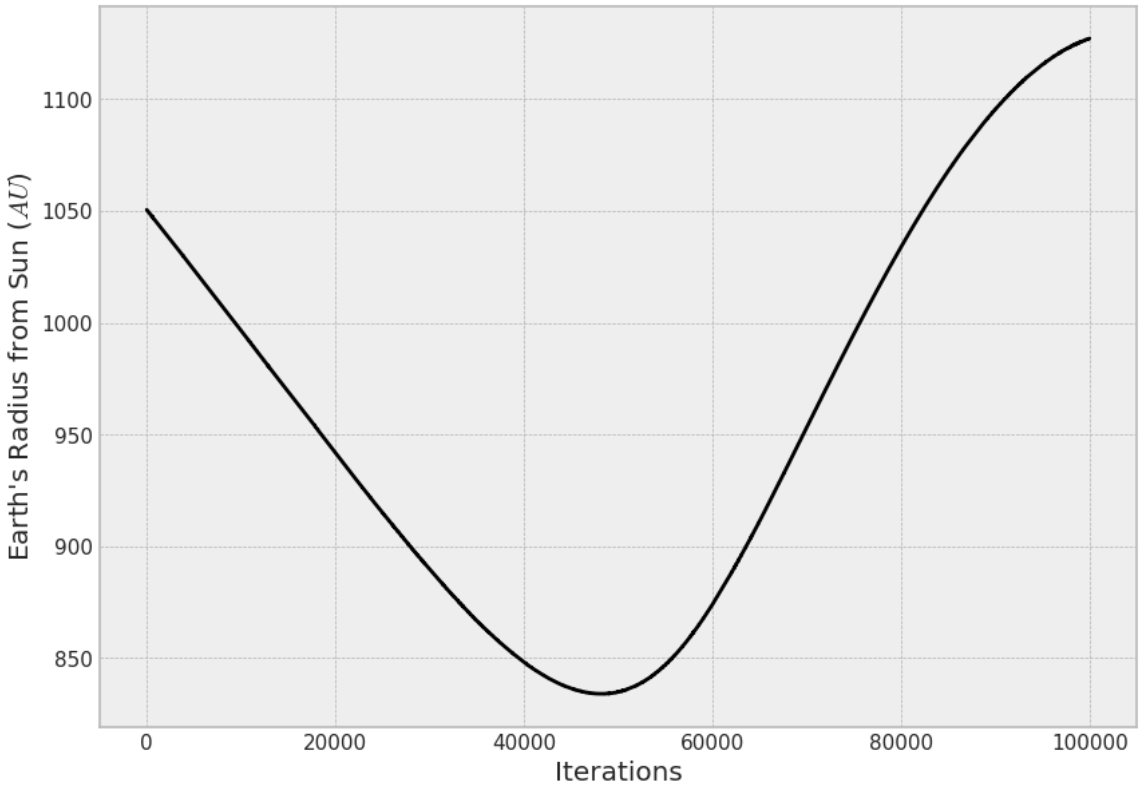



```
0.0
1.6282695500465238e-05
-6142.18506455316
```

Plotting radius of Earth's orbit over time

In [ ]:
```
kmTOau = 1.49597871e8

rx = x['1'].iloc[:,-1] - x['0'].iloc[:,-1]
ry = y['1'].iloc[:,-1] - y['0'].iloc[:,-1]
rz = z['1'].iloc[:,-1] - z['0'].iloc[:,-1]

r = np.sqrt(rx*rx + ry*ry + rz*rz)

iterations = np.arange(0, df.shape[0]);
x_data = iterations
y_data = r
#/ kmTOau

# Create plot environment
plt.style.use('bmh')
fig, ax = plt.subplots()
fig.set_size_inches(10, 7)

plt.plot(x_data, y_data,color="black")
plt.xlabel("Iterations")
plt.ylabel("Earth's Radius from Sun ($AU$)")

# print(r)
```

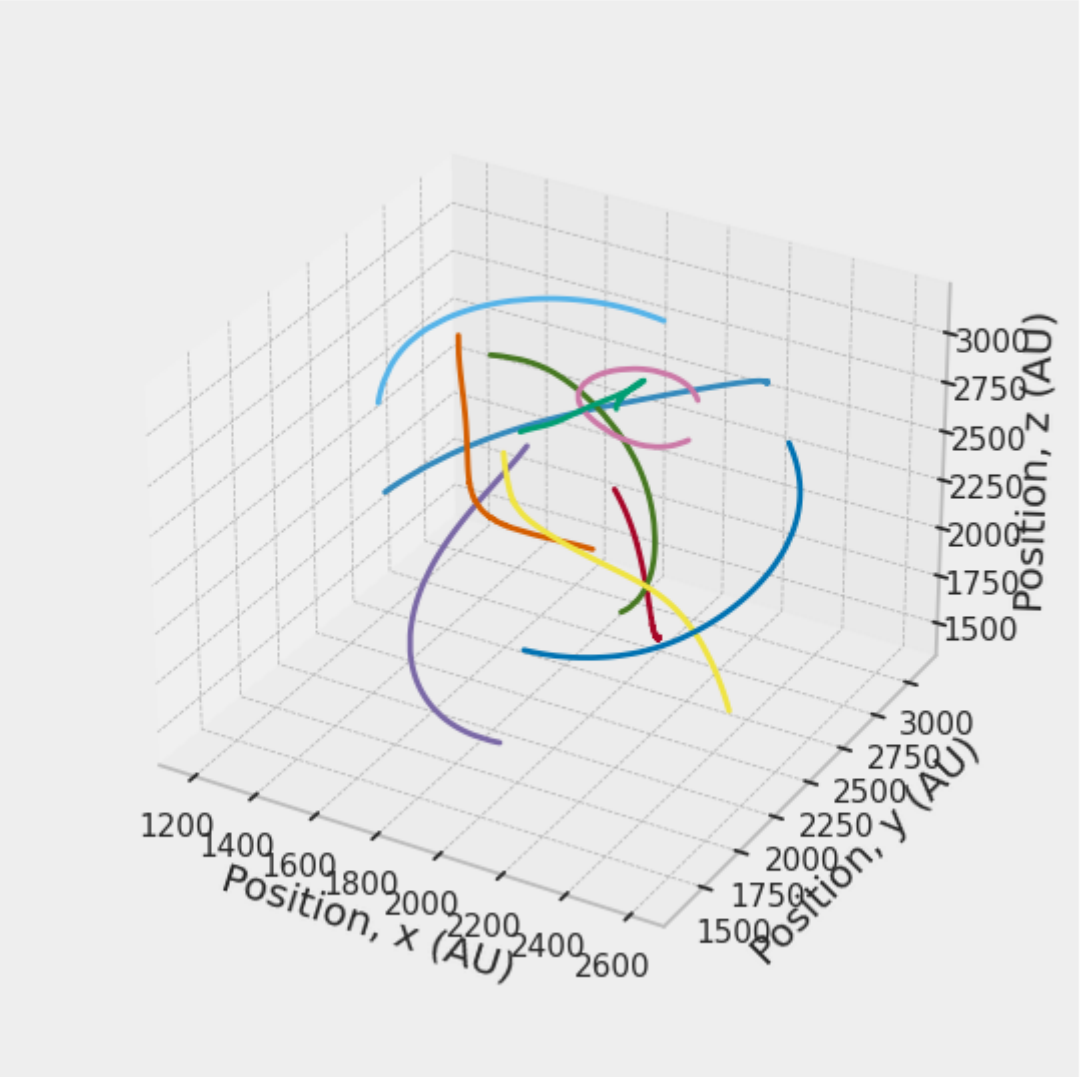
Out[ ]:
```
Text(0, 0.5, "Earth's Radius from Sun ($AU$)")
```



3D Plot

In [ ]:
```
plt.close()
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
# fig.gca(projection="3d")
# ax = Axes3D(fig)
ax.dist = 12
ax.set_xlabel('Position, x (AU)')
ax.set_ylabel('Position, y (AU)')
ax.set_zlabel('Position, z (AU)')

for i in range(0, int(df.shape[1]/10)):
    x_data = df[f'{i}']['x']
    y_data = df[f'{i}']['y']
    z_data = df[f'{i}']['z']

    ax.plot3D(x_data, y_data, z_data)
# plt.show()
```

```
/tmp/ipykernel_48129/3221254960.py:6: MatplotlibDeprecationWarning: The d
ist attribute was deprecated in Matplotlib 3.6 and will be removed two mi
nor releases later.
  ax.dist = 12
```

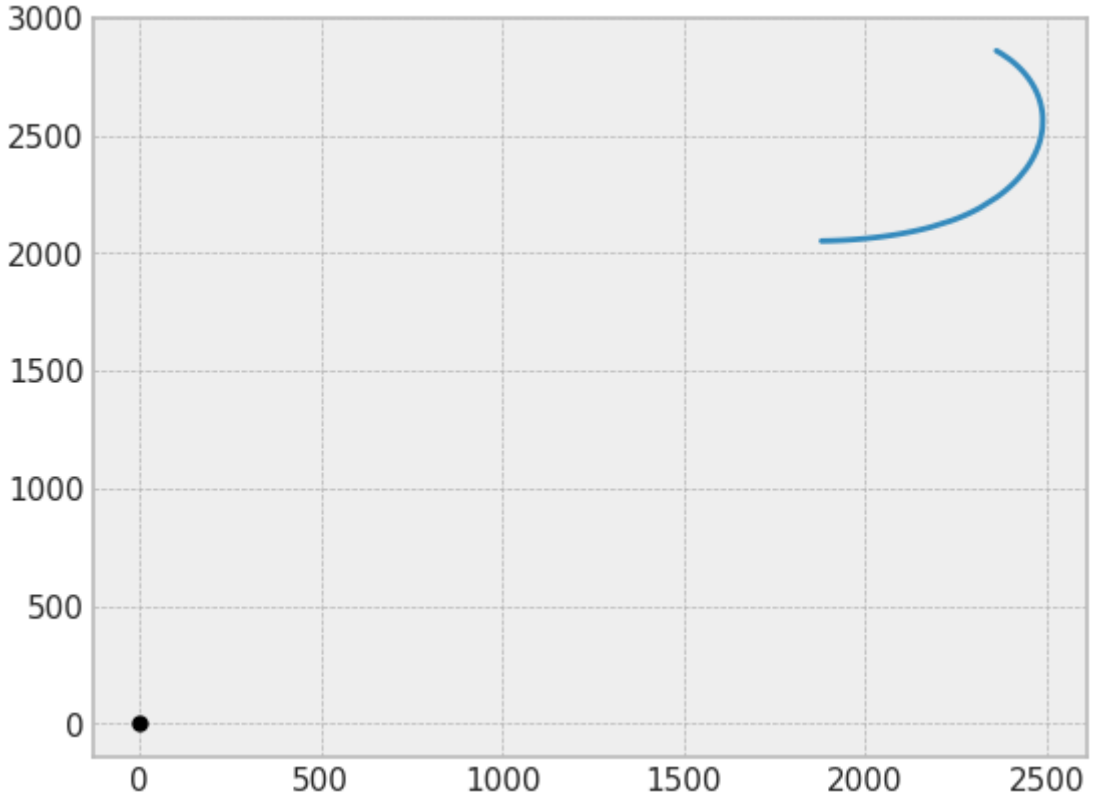


In [ ]:
```
print(a)
```

```
3.863117637147978e-06
```

In [ ]:
```
plt.plot(0, 0, marker = 'o', markersize=5, color="black")
plt.plot(x_data, y_data)
```

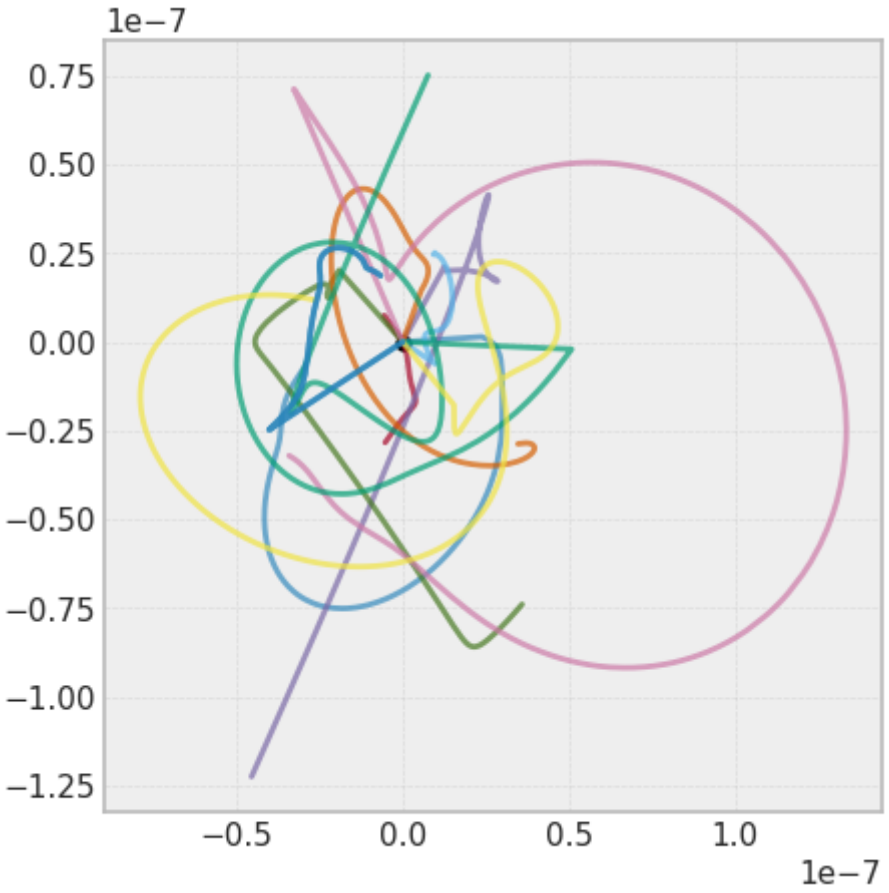
Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7f498ab40f70>]
```


3000
2500
2000
1500
1000
500
0
0
500
1000
1500
2000
2500

In [ ]:
```
# # test = df.set_index('fx').cumsum(axis=1)
# # print (test)
# orb = 1

# # test = df[f'{orb}']['fx']
# # print(test)

# # test2 = df.groupby(['fx'])
# # print(test2)

# # print (df.iloc[:, df.columns.get_level_values(1)=='fx'])

# x = df.loc[:,idx[:,'x']]
# y = df.loc[:,idx[:,'y']]
# z = df.loc[:,idx[:,'z']]

# fx = df.loc[:,idx[:,'fx']]
# fy = df.loc[:,idx[:,'fy']]
# fz = df.loc[:,idx[:,'fz']]

# vx = df.loc[:,idx[:,'vx']]
# vy = df.loc[:,idx[:,'vy']]
# vz = df.loc[:,idx[:,'vz']]

# # print(vx.cumsum(axis=1)) # use this to get cumulative sum over a row

# fx_sum = fx.cumsum(axis=1)
# fy_sum = fy.cumsum(axis=1)
# fz_sum = fz.cumsum(axis=1)

# f_sum = fx_sum.iloc[:,-1] + fy_sum.iloc[:,-1] + fz_sum.iloc[:,-1]



# # print(fx_sum.iloc[:,-1])
# # print(fy_sum.iloc[:,-1])
# print(f_sum)
```

In [ ]:
```
# fx = df[f'{orb}']['fx'][step]
# fy = df[f'{orb}']['fy'][step]
# fz = df[f'{orb}']['fz'][step]

# gravPotEnergy += 0.5 * (fx + fy + fz)

# # Kinetic Energy
# vx   = df[f'{orb}']['vx'][step]
# vy   = df[f'{orb}']['vy'][step]
# vz   = df[f'{orb}']['vz'][step]
# mass = df[f'{orb}']['M' ][0]

# velocitySqr = vx**2 + vy**2 + vz**2

# kineticEnergy += 0.5 * mass * velocitySqr #**2
```

In [ ]:
```
fig, ax = plt.subplots()
fig.set_size_inches(5, 5)
plt.plot(0, 0, marker = 'o', markersize=5, color="black")
for i in range(0, int(df.shape[1]/10)):
    x_data = df[f'{i}']['fx']
    y_data = df[f'{i}']['fy']

    plt.plot(x_data, y_data, alpha=0.7)
    plt.grid(alpha=0.3)
# x = df[f'{1}']['fx']
# y = df[f'{1}']['fy']
plt.plot(x_data, y_data, alpha=0.7)
plt.grid(alpha=0.3)
```



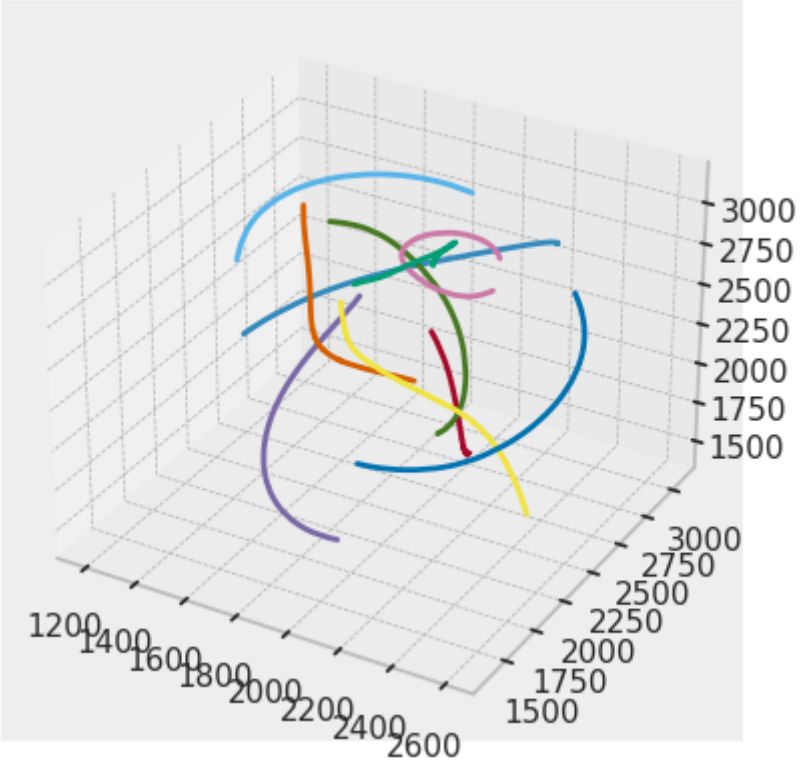
In [ ]:
```
print(len(df.columns[0]))
print(len(df.columns[3]) + 1)
print(df.shape[1]/6)
```

```
2
3
16.666666666666668
```

In [ ]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')

# plt.xlim(-100, 100)
# plt.ylim(-100, 100)
# plt.zlim(-100, 100)
# ax.set_xlim(-10, 10)
# ax.set_ylim(-10, 10)
# ax.set_zlim(-10, 10)

for i in range(0, int(df.shape[1]/9)):
    x_data = df[f'{i}']['x']
    y_data = df[f'{i}']['y']
    z_data = df[f'{i}']['z']


    # ax.scatter3D(0, 0, 0, s=50, c="black")
    ax.plot3D(x_data, y_data, z_data)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call las
File ~/.local/lib/python3.10/site-packages/pandas/core/indexes/base.py:38
0, in Index.get_loc(self, key, method, tolerance)
   3799 try:
-> 3800     return self._engine.get_loc(casted_key)
   3801 except KeyError as err:

File ~/.local/lib/python3.10/site-packages/pandas/_libs/index.pyx:138, in
andas._libs.index.IndexEngine.get_loc()

File ~/.local/lib/python3.10/site-packages/pandas/_libs/index.pyx:165, in
andas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:5745, in pandas._libs.hashta
e.PyObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:5753, in pandas._libs.hashta
e.PyObjectHashTable.get_item()

KeyError: '10'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call las
Cell In [19], line 12
      4 # plt.xlim(-100, 100)
      5 # plt.ylim(-100, 100)
      6 # plt.zlim(-100, 100)
      7 # ax.set_xlim(-10, 10)
      8 # ax.set_ylim(-10, 10)
      9 # ax.set_zlim(-10, 10)
     11 for i in range(0, int(df.shape[1]/9)):
---> 12     x_data = df[f'{i}']['x']
     13     y_data = df[f'{i}']['y']
     14     z_data = df[f'{i}']['z']

File ~/.local/lib/python3.10/site-packages/pandas/core/frame.py:3804, in
taFrame.__getitem__(self, key)
   3802 if is_single_key:
   3803     if self.columns.nlevels > 1:
-> 3804         return self._getitem_multilevel(key)
   3805     indexer = self.columns.get_loc(key)
   3806     if is_integer(indexer):

File ~/.local/lib/python3.10/site-packages/pandas/core/frame.py:3855, in
taFrame._getitem_multilevel(self, key)
   3853 def _getitem_multilevel(self, key):
   3854     # self.columns is a MultiIndex
-> 3855     loc = self.columns.get_loc(key)
   3856     if isinstance(loc, (slice, np.ndarray)):
   3857         new_columns = self.columns[loc]

File ~/.local/lib/python3.10/site-packages/pandas/core/indexes/multi.py:2
5, in MultiIndex.get_loc(self, key, method)
   2912     return mask
   2914 if not isinstance(key, tuple):
-> 2915     loc = self._get_level_indexer(key, level=0)
   2916     return _maybe_to_slice(loc)
   2918 keylen = len(key)
```

```
File ~/.local/lib/python3.10/site-packages/pandas/core/indexes/multi.py:3
2, in MultiIndex._get_level_indexer(self, key, level, indexer)
   3258         return slice(i, j, step)
   3260 else:
-> 3262     idx = self._get_loc_single_level_index(level_index, key)
   3264     if level > 0 or self._lexsort_depth == 0:
   3265         # Desired level is not sorted
   3266         if isinstance(idx, slice):
   3267             # test_get_loc_partial_timestamp_multiindex

File ~/.local/lib/python3.10/site-packages/pandas/core/indexes/multi.py:2
8, in MultiIndex._get_loc_single_level_index(self, level_index, key)
   2846     return -1
   2847 else:
-> 2848     return level_index.get_loc(key)

File ~/.local/lib/python3.10/site-packages/pandas/core/indexes/base.py:38
2, in Index.get_loc(self, key, method, tolerance)
   3800     return self._engine.get_loc(casted_key)
   3801 except KeyError as err:
-> 3802     raise KeyError(key) from err
   3803 except TypeError:
   3804     # If we have a listlike key, _check_indexing_error will raise
   3805     #  InvalidIndexError. Otherwise we fall through and re-raise
   3806     #  the TypeError.
   3807     self._check_indexing_error(key)

KeyError: '10'
```



In [ ]:
```
doPlot3D(df.xVel, df.yVel, df.Iteration)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call las
Cell In [20], line 1
----> 1 doPlot3D(df.xVel, df.yVel, df.Iteration)

File ~/.local/lib/python3.10/site-packages/pandas/core/generic.py:5907, i
NDFrame.__getattr__(self, name)
   5900 if (
   5901     name not in self._internal_names_set
   5902     and name not in self._metadata
   5903     and name not in self._accessors
   5904     and self._info_axis._can_hold_identifiers_and_holds_name(name
   5905 ):
   5906     return self[name]
-> 5907 return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'xVel'
```

X position vs Y position

for all timesteps

In [ ]:
```
doPlot2D(df.xPos, df.yPos, 10, 10, colour='black', replot=False)
plt.plot(0, 0, marker = 'o', markersize=20, color="black")
plt.xlabel("$x$ Position [$AU$]")
plt.ylabel("$y$ Position [$AU$]")
# plt.savefig('graphs/x-vs-y_pos_ts_1_1M-steps-wvenus.png', dpi='figure')
```

Velocity vs Time Step

In [ ]:
```
doPlot2D(df.T_S, df.xVel, 20, 10, True)
```

ARCHIVE

In [ ]:
```
# Iterate through all timesteps
# for step in range(0, df.shape[0]-1):
#     # Reset energies
#     gravPotEnergy = 0.0
#     kineticEnergy = 0.0

#     # Iterate through all bodies
#     for orb in range(0, int(df.shape[1]/10)):
#         # Gravitational Potential Energy
#         fx = df[f'{orb}']['fx'][step]
#         fy = df[f'{orb}']['fy'][step]
#         fz = df[f'{orb}']['fz'][step]

#         gravPotEnergy += 0.5 * (fx + fy + fz)

#         # Kinetic Energy
#         vx   = df[f'{orb}']['vx'][step]
#         vy   = df[f'{orb}']['vy'][step]
#         vz   = df[f'{orb}']['vz'][step]
#         mass = df[f'{orb}']['M' ][0]

#         velocitySqr = vx**2 + vy**2 + vz**2

#         kineticEnergy += 0.5 * mass * velocitySqr #**2

#     # Calculate Total Energy
#     totalEnergy[step] = gravPotEnergy + kineticEnergy

# print(totalEnergy)
```

In [ ]:
```
# vxSq = vx**2
# vySq = vy**2
# vzSq = vz**2
# vSqr = vxSq
# print(vSqr)
# vxSq, vySq, vzSqr = vxSq.align(vySq, fill_value=0)
# print(vxSq+vySq)
# vSqr = vxSq.merge(vySq).merge(vzSq)
# vxSq, vySq = vxSq.align(vySq)
# vSqr = vxSq.merge(vySq)
# print(vSqr)

# print()

# mass = df.loc[:,idx[:,'M']]#.cumsum(axis=1)
# print(mass)

# print(vxSq + vySq)
# print(vxSq.iloc[:,-1])
# print(vxSq)
# print(vxSq.add(vySq, axis=0))

# mass = df.loc[:,idx[:,'M']]#.cumsum(axis=1)
# print(mass)
# mass_sum = mass.iloc[:,-1]
# f_sum = fx_sum.iloc[:,-1] + fy_sum.iloc[:,-1] + fz_sum.iloc[:,-1]
# v_sum = vx_sum.iloc[:,-1] + vy_sum.iloc[:,-1] + vz_sum.iloc[:,-1]

# print(vx_sum.iloc[:,-1])
# print(v_sum)
```