HW3_pr1_pr2

April 12, 2018

1 Financial Computing III: HW3

1.1 Group:

Daniel Rojas Coy Lucas Duarte Bahia Harveen Oberoi Jordan Giebas

In [139]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import urllib.request
import html5lib
import lxml

from urllib.request import urlopen
from bs4 import BeautifulSoup
from mpl_toolkits.mplot3d import Axes3D
```

2 Problem 1

2.1 Part (a)

In [140]:
```
# Define string
url = "http://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages

daily_yield_curves = np.matrix(pd.read_html(url, attrs={'class':'t-chart'})[0]).tolis
```

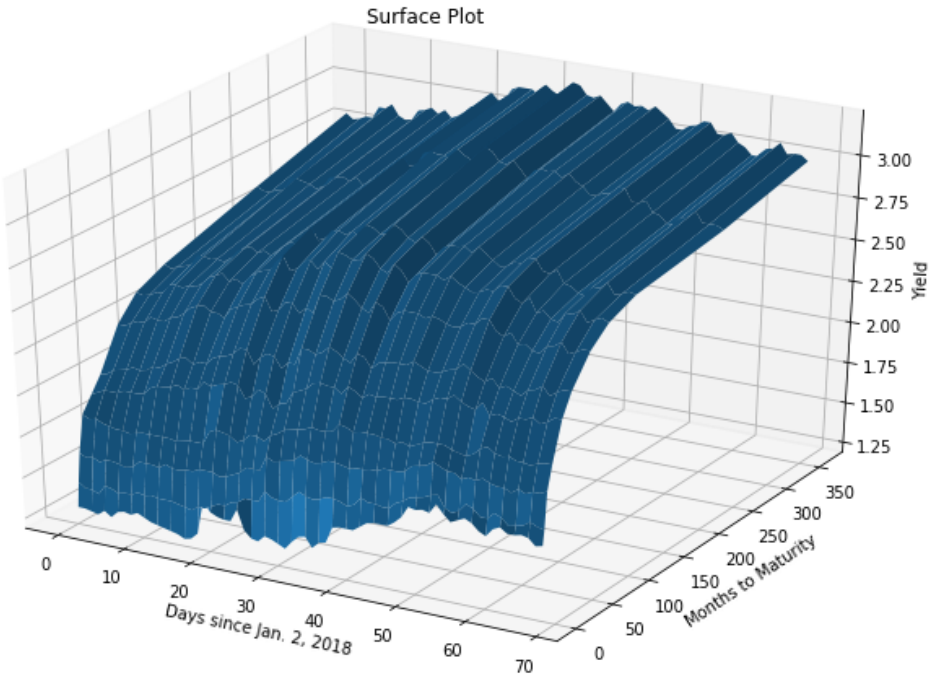
In [141]:
```
fout = open('daily_yield_curves.txt', 'w')
for row in daily_yield_curves:
    fout.write(str(row) + "\n")

fout.close()
```

2.2 Part (b)

In [142]:
```
dyc_df = pd.DataFrame(daily_yield_curves, dtype=np.float64)
```

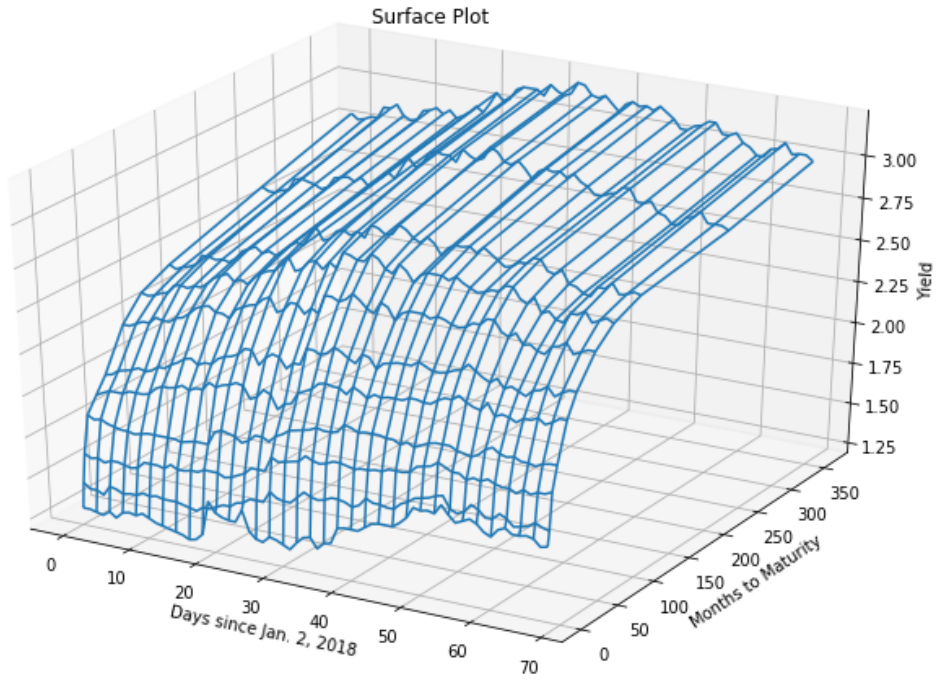
In [143]:
```
Y = np.array(dyc_df.columns)
dyc_df.columns = pd.Index(np.array(dyc_df.loc[0,:]))
dyc_df = dyc_df.drop(0, axis=0)
```

```
In [144]: dyc_df['days_passed'] = np.array(dyc_df.index) - 1

In [145]: X = np.array(dyc_df.index) - 1
          Y = np.array([float(12*i) for i in [1/12, 3/12, 6/12, 1, 2, 3, 5, 7, 10, 20, 30]])

          X, Y = np.meshgrid(X,Y)

In [146]: Z = np.array([np.array(dyc_df.iloc[i,1:12], dtype=np.float64) for i in range(0,69)])

In [147]: fig = plt.figure(figsize=(12,8))
          ax = fig.gca(projection='3d')
          ax.plot_surface(X,Y,Z)
          ax.set_xlabel("Days since Jan. 2, 2018")
          ax.set_ylabel("Months to Maturity")
          ax.set_zlabel("Yield")
          ax.set_title("Surface Plot")

Out[147]: Text(0.5,0.92,'Surface Plot')
```




```
In [148]: fig = plt.figure(figsize=(12,8))
          ax = fig.gca(projection='3d')
          ax.plot_wireframe(X,Y,Z)
          ax.set_xlabel("Days since Jan. 2, 2018")
          ax.set_ylabel("Months to Maturity")
          ax.set_zlabel("Yield")
          ax.set_title("Surface Plot")
```

```
Out[148]: Text(0.5,0.92,'Surface Plot')
```



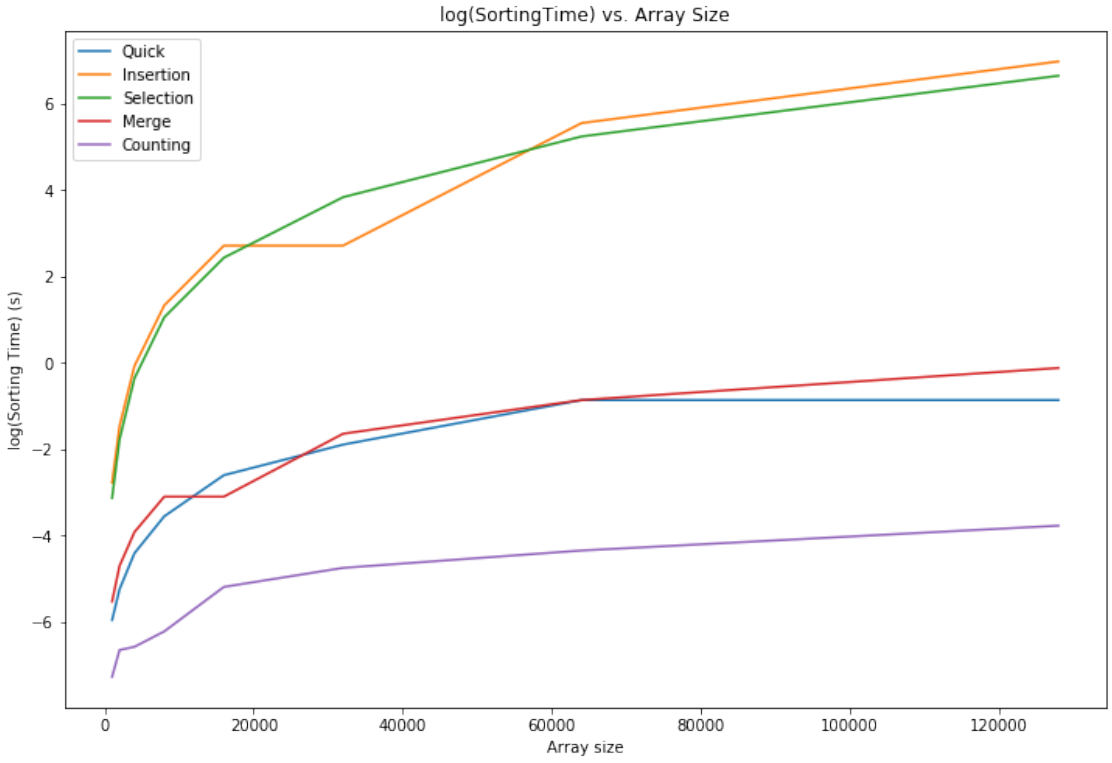
3 Problem 2

For this problem, we ran each of the parts in the terminal and stroed the results in a file to keep track of these. Hence, the data presented below is just c/p from that file.

```
In [155]: arr_size = np.array([1000,2000,4000,8000,16000,32000,64000,128000])
          qs_times = np.array([0.0026,0.0053,0.0122,0.0287,0.0743,0.1508,0.4226,0.4226])
          is_times = np.array([0.0625,0.2307,0.9292,3.7807,15.0723,15.0723,256.9330,1069.7866])
          ss_times = np.array([0.0439,0.1697,0.6988,2.8699,11.3957,46.2508,188.4959,768.218156
          ms_times = np.array([0.0040,0.0091,0.0199,0.0452,0.0452,0.1939,0.4229,0.8882])
          cs_times = np.array([0.0007,0.0013,0.0014,0.0020,0.0056,0.0087,0.0130,0.0231])

In [175]: # plot features
          plt.figure(figsize=(12,8))
          plt.xlabel("Array size")
          plt.ylabel("log(Sorting Time) (s)")
          plt.title("log(SortingTime) vs. Array Size")

          # plot content
          plt.plot(arr_size, np.log(qs_times))
          plt.plot(arr_size, np.log(is_times))
```

```
plt.plot(arr_size, np.log(ss_times))
plt.plot(arr_size, np.log(ms_times))
plt.plot(arr_size, np.log(cs_times))

plt.legend(['Quick', 'Insertion', 'Selection', 'Merge', 'Counting'], loc='upper left
```

```
Out[175]: <matplotlib.legend.Legend at 0x1183fdc88>
```



Relative to the C++ performance for each of these algorithms, Python doesn't perform as well. The ordering in the performance presented above is still as suspected though. At the top, Insertion and Selection sort perform roughly the same, as they are $\Theta(n^2)$ In the middle, Quick and Merge sort perform roughly the same, as they are $\Theta(nlog(n))$ The lowest is Counting sort, which is $\Theta(n+k)$. k is negligble in these cases, so this is essentially a linear algorithm.