Numerical Methods - Assignment 1

1. Python and MATLAB

List at least 3 differences between Python and MATLAB.

While both Python and MATLAB are both useful as programming languages, they have several factors that differentiate them from each other. Python is much more widely supported and versatile than MATLAB since Python is a free tool. It allows more people to make use of the language, while MATLAB requires paid licensing for use which often deters users. Another key difference is that Python provides more of an opportunity for programming collaboration. The program is open sourced, so users are able to modify each other's code much more easily and look to the multitude of discussion forums for support. However, MATLAB is a much better program for complex mathematics problems. Python tends to be slower and less effienct in this area.

2. Plot Data

Read the data and create a plot.

- Import `matplotlib` and `pandas` packages.
- Read in `data/ethanol_IR.csv` file and create a plot of IR spectra data.

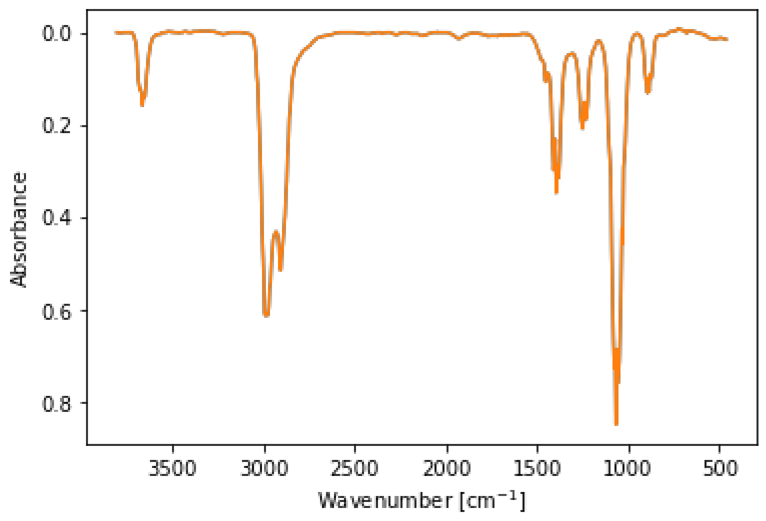
In [24]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd

excel = pd.read_csv('data/ethanol_IR.csv')
x = pd.DataFrame(excel, columns = ['wavenumber [cm^-1]'])
y = pd.DataFrame(excel, columns = ['absorbance'])

fig, ax = plt.subplots(); ax.plot(x,y)
ax.set_xlabel('Wavenumber [cm$^{-1}$]')
ax.set_ylabel('Absorbance')

ax.invert_xaxis()
ax.invert_yaxis()

plt.plot(x,y)
plt.show()
```



The dataset provided is the IR spectra for an ethanol (C2H5OH) compound. The distinct aspects of the compound are the -OH or hydroxyl group, sp3 C-H bond, C-O bond. The peak indicative of the -OH group appears at approximately 3700 cm^-1 due to that distinct bond stretching. The peak around 3000 cm^-1 points to the presence of an sp3 C-H bond vibration. This type of peak is broad and sharp but tends not to go beyond the 3000 cm^-1 threshold. The fact that the peak stops around 3000 cm^-1 means that there are no double or triple bonds present. Another indication of the carbon-oxygen bond is the peak around 1100 cm^-1. Ultimately, the three prominent peaks are indicative of the bond vibrations in ethanol.

3. Matrix-vector Multiplication

Write a funcion that uses `for` loops.

This function should multiply an arbitrary matrix and vector.

In [21]:
```
import numpy as np

def mulMatVec(matrix, vector):
   #import numpy as np
    for i in range(A.shape[0]):
        result = []

        for j in range(A.shape[1]):
            value = [A[i][j] * B[j]]
            result += value
            x = sum(result)

    return x
```

You can use the matrix and vector given below.

In [22]:
```
import numpy as np

A = np.array([[1, 2], [-4, 5]])
B = np.array([-2, 3])

mulMatVec(A,B)
```

Out[22]:
```
23
```

Or create an arbitrary set of matrix and vector using `numpy.random.rand` .

In [ ]:
```
from numpy.random import rand

# You can create your own inputs
```

Show that your function is correct using `numpy.isclose` .

In [25]:
```
import numpy as np

np.isclose(mulMatVec(A,B),np.mulMat(A,B))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-25-44263c41bced> in <module>
      1 import numpy as np
      2 
----> 3 np.isclose(mulMatVec(A,B),np.mulMat(A,B))

~\anaconda3\lib\site-packages\numpy\__init__.py in __getattr__(attr)
    217                 return Tester
    218             else:
--> 219                 raise AttributeError("module {!r} has no attribute "
    220                                      "{!r}".format(__name__, attr))
    221 

AttributeError: module 'numpy' has no attribute 'mulMat'
```

4. Vandermonde Matrix

Use `numpy.hstack` to construct a 4th-order Vandermonde matrix.

Range should be from -1 to 1 with a resolution of 25 (i.e. the number of rows should be 25).

In [40]:
```
resolution = 25

x = np.linspace(-1,1,resolution)
x = x.reshape(-1,1)
vanMat = np.hstack((x**0,x**1,x**2,x**3,x**4))
print(vanMat)
```

```
[[ 1.00000000e+00 -1.00000000e+00  1.00000000e+00 -1.00000000e+00
   1.00000000e+00]
 [ 1.00000000e+00 -9.16666667e-01  8.40277778e-01 -7.70254630e-01
   7.06066744e-01]
 [ 1.00000000e+00 -8.33333333e-01  6.94444444e-01 -5.78703704e-01
   4.82253086e-01]
 [ 1.00000000e+00 -7.50000000e-01  5.62500000e-01 -4.21875000e-01
   3.16406250e-01]
 [ 1.00000000e+00 -6.66666667e-01  4.44444444e-01 -2.96296296e-01
   1.97530864e-01]
 [ 1.00000000e+00 -5.83333333e-01  3.40277778e-01 -1.98495370e-01
   1.15788966e-01]
 [ 1.00000000e+00 -5.00000000e-01  2.50000000e-01 -1.25000000e-01
   6.25000000e-02]
 [ 1.00000000e+00 -4.16666667e-01  1.73611111e-01 -7.23379630e-02
   3.01408179e-02]
 [ 1.00000000e+00 -3.33333333e-01  1.11111111e-01 -3.70370370e-02
   1.23456790e-02]
 [ 1.00000000e+00 -2.50000000e-01  6.25000000e-02 -1.56250000e-02
   3.90625000e-03]
 [ 1.00000000e+00 -1.66666667e-01  2.77777778e-02 -4.62962963e-03
   7.71604938e-04]
 [ 1.00000000e+00 -8.33333333e-02  6.94444444e-03 -5.78703704e-04
   4.82253086e-05]
 [ 1.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00]
 [ 1.00000000e+00  8.33333333e-02  6.94444444e-03  5.78703704e-04
   4.82253086e-05]
 [ 1.00000000e+00  1.66666667e-01  2.77777778e-02  4.62962963e-03
   7.71604938e-04]
 [ 1.00000000e+00  2.50000000e-01  6.25000000e-02  1.56250000e-02
   3.90625000e-03]
 [ 1.00000000e+00  3.33333333e-01  1.11111111e-01  3.70370370e-02
   1.23456790e-02]
 [ 1.00000000e+00  4.16666667e-01  1.73611111e-01  7.23379630e-02
   3.01408179e-02]
 [ 1.00000000e+00  5.00000000e-01  2.50000000e-01  1.25000000e-01
   6.25000000e-02]
 [ 1.00000000e+00  5.83333333e-01  3.40277778e-01  1.98495370e-01
   1.15788966e-01]
 [ 1.00000000e+00  6.66666667e-01  4.44444444e-01  2.96296296e-01
   1.97530864e-01]
 [ 1.00000000e+00  7.50000000e-01  5.62500000e-01  4.21875000e-01
   3.16406250e-01]
 [ 1.00000000e+00  8.33333333e-01  6.94444444e-01  5.78703704e-01
   4.82253086e-01]
 [ 1.00000000e+00  9.16666667e-01  8.40277778e-01  7.70254630e-01
   7.06066744e-01]
 [ 1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
   1.00000000e+00]]
```

Create an orthonormal version of the Vandermonde matrix.

Orthonormal means:

- the L_2 norm of each column is 1.
- the inner product between any 2 columns is 0.

Print the orthonormalized Vandermonde matrix.

In [1]:

```
import numpy as np

x = np.linspace(-1,1, 25)
x = x.reshape(-1,1)
vanMat = np.hstack((x**0, x**1, x**2, x**3, x**4))

#COLUMN 0
colm_0 = vanMat[:,0]
norm_colm_0 = np.linalg.norm(colm_0,2)

colm_0_normed = colm_0 / norm_colm_0

#COLUMN 1
colm_1 = vanMat[:,1]
norm_colm_1 = np.linalg.norm(colm_1,2)
colm_1_normed = colm_1 / norm_colm_1

colm_1_ortho = colm_1_normed - np.dot(colm_0_normed, colm_1_normed) * colm_0_

print(np.dot(colm_0_normed, colm_1_ortho))
print(np.isclose(np.dot(colm_0_normed, colm_1_ortho),0))

#COLUMN 2

colm_2 = vanMat[:,2]
norm_colm_2 = np.linalg.norm(colm_2,2)
colm_2_normed = colm_2 / norm_colm_2

colm_2_ortho = colm_2_normed - np.dot(colm_1_ortho, colm_2_normed) * colm_1_o

#COLUMN 3
colm_3 = vanMat[:,3]
norm_colm_3 = np.linalg.norm(colm_3,2)
colm_3_normed = colm_3 / norm_colm_3

colm_3_ortho = colm_3_normed - np.dot(colm_2_normed, colm_3_normed) * colm_2_

#COLUMN 4
colm_4 = vanMat[:,4]
norm_colm_4 = np.linalg.norm(colm_4,2)
colm_4_normed = colm_4 / norm_colm_4

colm_4_ortho = colm_4_normed - np.dot(colm_3_normed, colm_4_normed) * colm_3_

vanMatOrtho = np.hstack((colm_0, colm_1_ortho, colm_2_ortho, colm_3_ortho, co

print(vanMatOrtho)
```

```
-1.3877787807814457e-17
True
[ 1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00 -3.32820118e-01 -3.05085108e-01 -2.77350098e-01
 -2.49615088e-01 -2.21880078e-01 -1.94145069e-01 -1.66410059e-01
 -1.38675049e-01 -1.10940039e-01 -8.32050294e-02 -5.54700196e-02
 -2.77350098e-02  1.38777878e-17  2.77350098e-02  5.54700196e-02
  8.32050294e-02  1.10940039e-01  1.38675049e-01  1.66410059e-01
  1.94145069e-01  2.21880078e-01  2.49615088e-01  2.77350098e-01
  3.05085108e-01  3.32820118e-01  2.64023579e-01  1.98017684e-01
  1.37751432e-01  8.32248237e-02  3.44378581e-02 -8.60946452e-03
 -4.59171441e-02 -7.74851806e-02 -1.03313574e-01 -1.23402325e-01
 -1.37751432e-01 -1.46360897e-01 -1.49230718e-01 -1.46360897e-01
 -1.37751432e-01 -1.23402325e-01 -1.03313574e-01 -7.74851806e-02
 -4.59171441e-02 -8.60946452e-03  3.44378581e-02  8.32248237e-02
  1.37751432e-01  1.98017684e-01  2.64023579e-01 -1.65431169e-01
 -8.27155843e-02 -1.79816488e-02  3.04053334e-02  6.40800574e-02
  8.46772187e-02  9.38315126e-02  9.31776345e-02  8.43502797e-02
  6.89841434e-02  4.87139212e-02  2.51743083e-02  2.08166817e-17
 -2.51743083e-02 -4.87139212e-02 -6.89841434e-02 -8.43502797e-02
 -9.31776345e-02 -9.38315126e-02 -8.46772187e-02 -6.40800574e-02
 -3.04053334e-02  1.79816488e-02  8.27155843e-02  1.65431169e-01
 -2.43212531e-03 -9.03483736e-02 -1.47692071e-01 -1.80716215e-01
 -1.95078282e-01 -1.95840220e-01 -1.87468459e-01 -1.73833901e-01
 -1.58211926e-01 -1.43282391e-01 -1.31129628e-01 -1.23242447e-01
 -1.20514132e-01 -1.23242447e-01 -1.31129628e-01 -1.43282391e-01
 -1.58211926e-01 -1.73833901e-01 -1.87468459e-01 -1.95840220e-01
 -1.95078282e-01 -1.80716215e-01 -1.47692071e-01 -9.03483736e-02
 -2.43212531e-03]
```

Show that the L_2 of 5th column is 1.

In [3]:
```
y = np.linalg.norm(colm_4_ortho,2)

print(y)
```

```
0.7610755324022486
```

Show that the inner product between 1st column & 4th column is 0.

In [47]:
```
t = np.inner(colm_0,colm_3_ortho)

np.isclose(t,0)
```

Out[47]:
```
True
```

Compute the rank of the orthonormalized Vandermonde matrix.

In [50]:
```
e = np.linalg.matrix_rank(vanMatOrtho)
print(e)
```

```
1
```

Show that the rank is equal to the number of columns.

Show that the rank is equal to the number of columns.

In [ ]:

Change the resolution to 30 and show that the rank is independent of the number of rows.

In [48]:
```
resolution = 30

x = np.linspace(-1,1,resolution)
x = x.reshape(-1,1)
vanMat = np.hstack((x**0,x**1,x**2,x**3,x**4))
print(vanMat)
```

```
[[ 1.00000000e+00 -1.00000000e+00  1.00000000e+00 -1.00000000e+00
   1.00000000e+00]
 [ 1.00000000e+00 -9.31034483e-01  8.66825208e-01 -8.07044159e-01
   7.51385941e-01]
 [ 1.00000000e+00 -8.62068966e-01  7.43162901e-01 -6.40657674e-01
   5.52291098e-01]
 [ 1.00000000e+00 -7.93103448e-01  6.29013080e-01 -4.98872442e-01
   3.95657454e-01]
 [ 1.00000000e+00 -7.24137931e-01  5.24375743e-01 -3.79720366e-01
   2.74969920e-01]
 [ 1.00000000e+00 -6.55172414e-01  4.29250892e-01 -2.81233343e-01
   1.84256328e-01]
 [ 1.00000000e+00 -5.86206897e-01  3.43638526e-01 -2.01443274e-01
   1.18087436e-01]
 [ 1.00000000e+00 -5.17241379e-01  2.67538644e-01 -1.38382057e-01
   7.15769263e-02]
 [ 1.00000000e+00 -4.48275862e-01  2.00951249e-01 -9.00815942e-02
   4.03814043e-02]
 [ 1.00000000e+00 -3.79310345e-01  1.43876338e-01 -5.45737833e-02
   2.07004005e-02]
 [ 1.00000000e+00 -3.10344828e-01  9.63139120e-02 -2.98905244e-02
   9.27636965e-03]
 [ 1.00000000e+00 -2.41379310e-01  5.82639715e-02 -1.40637172e-02
   3.39469037e-03]
 [ 1.00000000e+00 -1.72413793e-01  2.97265161e-02 -5.12526139e-03
   8.83665757e-04]
 [ 1.00000000e+00 -1.03448276e-01  1.07015458e-02 -1.10705646e-03
   1.14523082e-04]
 [ 1.00000000e+00 -3.44827586e-02  1.18906064e-03 -4.10020911e-05
   1.41386521e-06]
 [ 1.00000000e+00  3.44827586e-02  1.18906064e-03  4.10020911e-05
   1.41386521e-06]
 [ 1.00000000e+00  1.03448276e-01  1.07015458e-02  1.10705646e-03
   1.14523082e-04]
 [ 1.00000000e+00  1.72413793e-01  2.97265161e-02  5.12526139e-03
   8.83665757e-04]
 [ 1.00000000e+00  2.41379310e-01  5.82639715e-02  1.40637172e-02
   3.39469037e-03]
 [ 1.00000000e+00  3.10344828e-01  9.63139120e-02  2.98905244e-02
   9.27636965e-03]
 [ 1.00000000e+00  3.79310345e-01  1.43876338e-01  5.45737833e-02
   2.07004005e-02]
 [ 1.00000000e+00  4.48275862e-01  2.00951249e-01  9.00815942e-02
   4.03814043e-02]
 [ 1.00000000e+00  5.17241379e-01  2.67538644e-01  1.38382057e-01
   7.15769263e-02]
 [ 1.00000000e+00  5.86206897e-01  3.43638526e-01  2.01443274e-01
   1.18087436e-01]
 [ 1.00000000e+00  6.55172414e-01  4.29250892e-01  2.81233343e-01
```

```
  1.84256328e-01]
 [ 1.00000000e+00  7.24137931e-01  5.24375743e-01  3.79720366e-01
   2.74969920e-01]
 [ 1.00000000e+00  7.93103448e-01  6.29013080e-01  4.98872442e-01
   3.95657454e-01]
 [ 1.00000000e+00  8.62068966e-01  7.43162901e-01  6.40657674e-01
   5.52291098e-01]
 [ 1.00000000e+00  9.31034483e-01  8.66825208e-01  8.07044159e-01
   7.51385941e-01]
 [ 1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
   1.00000000e+00]]
```

In [49]:

```
np.linalg.matrix_rank(vanMat)
```

Out[49]:

```
5
```

In [ ]: