Tsung-Yu Liu Assignment 0

Task 1

```
C:\Users\jason>conda Info
usage: conda-script.py [-h] [-v] [--no-plugins] [-V] COMMAND ...
conda-script.py: error: argument COMMAND: invalid choice: 'Info' (choose from 'activate', 'deactivate', 'clean', 'compar
e', 'config', 'create', 'env', 'export', 'info', 'init', 'install', 'list', 'notices', 'package', 'remove', 'uninstall',
 'rename', 'run', 'search', 'update', 'upgrade', 'build', 'content-trust', 'convert', 'debug', 'develop', 'doctor', 'ind
ex', 'inspect', 'metapackage', 'render', 'repoquery', 'skeleton', 'verify', 'repo', 'pack', 'server', 'token')
```

Task 2

```
In [5]: import numpy as np

In [6]: from scipy import io, integrate, linalg, signal

In [7]: from scipy.sparse.linalg import cg, eigs

In [8]: np.ndim(a)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[8], line 1
----> 1 np.ndim(a)

NameError: name 'a' is not defined

In [9]: np.ndim(1)
Out[9]: 0

In [10]: np.ndim(3)
Out[10]: 0

In [11]: a = {1, 2, 3}

In [12]: np.ndim(a)
Out[12]: 0

In [13]: a = np.zeros((2, 3, 4, 5))

In [14]: print(np.ndim(a))
4

In [15]: print(np.size(a))
120

In [16]: print(np.shape(a))
(2, 3, 4, 5)

In [17]: print(a.shape[n-1])
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[17], line 1
----> 1 print(a.shape[n-1])

NameError: name 'n' is not defined

In [18]: print(a.shape[4-1])
5

In [19]: print(a.shape[3-1])
```

```
In [20]: np.array([[1., 2., 3.], [4., 5., 6.]])
Out[20]:
array([[1., 2., 3.],
       [4., 5., 6.]])

In [21]: np.block([[a, b], [c, d]])
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[21], line 1
----> 1 np.block([[a, b], [c, d]])

NameError: name 'b' is not defined

In [22]: a = np.array([[1, 2], [3, 4]])
    ...: b = np.array([[5, 6], [7, 8]])
    ...: c = np.array([[9, 10], [11, 12]])
    ...: d = np.array([[13, 14], [15, 16]])

In [23]: np.block([[a, b], [c, d]])
Out[23]:
array([[ 1,  2,  5,  6],
       [ 3,  4,  7,  8],
       [ 9, 10, 13, 14],
       [11, 12, 15, 16]])

In [24]: a[-1]
Out[24]: array([3, 4])
```

```
In [32]: a = np.array([[1, 2, 3, 4, 10000],
    ...:               [5, 6, 7, 8, 10000000]])

In [33]: a[1, 4]
Out[33]: 10000000

In [34]: a[1]
Out[34]: array([       5,        6,        7,        8, 10000000])

In [35]: a[0:5]
Out[35]:
array([[       1,        2,        3,        4,    10000],
       [       5,        6,        7,        8, 10000000]])

In [36]: a[:5]
Out[36]:
array([[       1,        2,        3,        4,    10000],
       [       5,        6,        7,        8, 10000000]])

In [37]: a[0:5, :]
Out[37]:
array([[       1,        2,        3,        4,    10000],
       [       5,        6,        7,        8, 10000000]])

In [38]: a[-5:]
Out[38]:
array([[       1,        2,        3,        4,    10000],
       [       5,        6,        7,        8, 10000000]])

In [39]: a[0:3, 4:9]
Out[39]:
array([[   10000],
       [10000000]])
```

```
In [43]: a = np.arange(45).reshape(5, 9)

In [44]: a[np.ix_([1, 3, 4], [0, 2])]
Out[44]:
array([[ 9, 11],
       [27, 29],
       [36, 38]])

In [45]: a[np.ix_([1, 3, 4], [0, 2])]
Out[45]:
array([[ 9, 11],
       [27, 29],
       [36, 38]])

In [46]: a[2:21:2,:]
Out[46]:
array([[18, 19, 20, 21, 22, 23, 24, 25, 26],
       [36, 37, 38, 39, 40, 41, 42, 43, 44]])

In [47]: a[::2, :]
Out[47]:
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8],
       [18, 19, 20, 21, 22, 23, 24, 25, 26],
       [36, 37, 38, 39, 40, 41, 42, 43, 44]])

In [48]: a[::-1,:]
Out[48]:
array([[36, 37, 38, 39, 40, 41, 42, 43, 44],
       [27, 28, 29, 30, 31, 32, 33, 34, 35],
       [18, 19, 20, 21, 22, 23, 24, 25, 26],
       [ 9, 10, 11, 12, 13, 14, 15, 16, 17],
       [ 0,  1,  2,  3,  4,  5,  6,  7,  8]])

In [49]: a[np.r_[:len(a),0]]
Out[49]:
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8],
       [ 9, 10, 11, 12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23, 24, 25, 26],
       [27, 28, 29, 30, 31, 32, 33, 34, 35],
       [36, 37, 38, 39, 40, 41, 42, 43, 44],
       [ 0,  1,  2,  3,  4,  5,  6,  7,  8]])

In [50]: a.transpose()
Out[50]:
array([[ 0,  9, 18, 27, 36],
       [ 1, 10, 19, 28, 37],
       [ 2, 11, 20, 29, 38],
       [ 3, 12, 21, 30, 39],
       [ 4, 13, 22, 31, 40],
       [ 5, 14, 23, 32, 41],
       [ 6, 15, 24, 33, 42],
       [ 7, 16, 25, 34, 43],
       [ 8, 17, 26, 35, 44]])

In [51]: a.conj().transpose()
Out[51]:
array([[ 0,  9, 18, 27, 36],
       [ 1, 10, 19, 28, 37],
       [ 2, 11, 20, 29, 38],
       [ 3, 12, 21, 30, 39],
       [ 4, 13, 22, 31, 40],
       [ 5, 14, 23, 32, 41],
       [ 6, 15, 24, 33, 42],
       [ 7, 16, 25, 34, 43],
       [ 8, 17, 26, 35, 44]])
```

```
In [53]: a = np.array([[1, 2, 3],
    ...:               [4, 5, 6]])

In [54]: b = np.array([[7, 8],
    ...:               [9, 10],
    ...:               [11, 12]])

In [55]: a @ b
Out[55]:
array([[ 58,  64],
       [139, 154]])

In [56]: a * b
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[56], line 1
----> 1 a * b

ValueError: operands could not be broadcast together with shapes (2,3) (3,2)

In [57]: a = np.array([[1, 2, 3],
    ...:               [4, 5, 6]])

In [58]: b = np.array([[7, 8, 9],
    ...:               [10, 11, 12]])

In [59]: a * b
Out[59]:
array([[ 7, 16, 27],
       [40, 55, 72]])

In [60]: a/b
Out[60]:
array([[0.14285714, 0.25      , 0.33333333],
       [0.4       , 0.45454545, 0.5       ]])

In [61]:
    ...: a**3
Out[61]:
array([[  1,   8,  27],
       [ 64, 125, 216]], dtype=int32)

In [62]: (a > 0.5)
Out[62]:
array([[ True,  True,  True],
       [ True,  True,  True]])

In [63]: np.nonzero(a > 0.5)
Out[63]:
(array([0, 0, 0, 1, 1, 1], dtype=int64),
 array([0, 1, 2, 0, 1, 2], dtype=int64))
```

```
In [65]: a = np.array([[1, 2, 3, 4, 5],
    ...:                [6, 7, 8, 9, 10],
    ...:                [11, 12, 13, 14, 15]])

In [66]: v = np.array([0.2, 0.6, 0.1, 0.8, 0.3])

In [67]: a[:,np.nonzero(v > 0.5)[0]]
Out[67]:
array([[ 2,  4],
       [ 7,  9],
       [12, 14]])

In [68]:
    ...: a[:, v.T > 0.5]
Out[68]:
array([[ 2,  4],
       [ 7,  9],
       [12, 14]])

In [69]: a[a < 0.5]=0

In [70]:
    ...: a * (a > 0.5)
Out[70]:
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15]])

In [71]: a[:] = 3
```

```
In [73]: x = np.array([1, 2, 3, 4, 5])

In [74]: y = x.copy()

In [75]: y[0] = 100

In [76]: print("Array x:", x)
    ...: print("Array y:", y)
Array x: [1 2 3 4 5]
Array y: [100   2   3   4   5]

In [77]:
    ...: y = x[1, :].copy()
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[77], line 1
----> 1 y = x[1, :].copy()

IndexError: too many indices for array: array is 1-dimensional, but 2 were indexed

In [78]: x = np.array([[1, 2, 3],
    ...:               [4, 5, 6],
    ...:               [7, 8, 9]])
    ...:

In [79]:
    ...: y = x[1, :].copy()

In [80]: y = x.flatten()

In [81]: np.arange(1., 11.)
Out[81]: array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])

In [82]: np.arange(10.)
Out[82]: array([0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])

In [83]: np.arange(1.,11.)[:, np.newaxis]
Out[83]:
array([[ 1.],
       [ 2.],
       [ 3.],
       [ 4.],
       [ 5.],
       [ 6.],
       [ 7.],
       [ 8.],
       [ 9.],
       [10.]])
```

```
In [84]:
    ...: np.zeros((3, 4))
Out[84]:
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])

In [85]:
    ...: np.zeros((3, 4, 5))
Out[85]:
array([[[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]]])

In [86]: np.ones((3, 4))
Out[86]:
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])

In [87]: np.eye(3)
Out[87]:
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])

In [88]: np.diag(a)
Out[88]: array([3, 3, 3])

In [89]: np.diag(v, 0)
Out[89]:
array([[0.2, 0. , 0. , 0. , 0. ],
       [0. , 0.6, 0. , 0. , 0. ],
       [0. , 0. , 0.1, 0. , 0. ],
       [0. , 0. , 0. , 0.8, 0. ],
       [0. , 0. , 0. , 0. , 0.3]])

In [90]: from numpy.random import default_rng
    ...: rng = default_rng(42)
    ...: rng.random((3, 4))
Out[90]:
array([[0.77395605, 0.43887844, 0.85859792, 0.69736803],
       [0.09417735, 0.97562235, 0.7611397 , 0.78606431],
       [0.12811363, 0.45038594, 0.37079802, 0.92676499]])

In [91]: np.linspace(1,3,4)
Out[91]: array([1.        , 1.66666667, 2.33333333, 3.
```

```
In [92]: np.mgrid[0:9.,0:6.]
Out[92]:
array([[[0., 0., 0., 0., 0., 0.],
        [1., 1., 1., 1., 1., 1.],
        [2., 2., 2., 2., 2., 2.],
        [3., 3., 3., 3., 3., 3.],
        [4., 4., 4., 4., 4., 4.],
        [5., 5., 5., 5., 5., 5.],
        [6., 6., 6., 6., 6., 6.],
        [7., 7., 7., 7., 7., 7.],
        [8., 8., 8., 8., 8., 8.]],

       [[0., 1., 2., 3., 4., 5.],
        [0., 1., 2., 3., 4., 5.],
        [0., 1., 2., 3., 4., 5.],
        [0., 1., 2., 3., 4., 5.],
        [0., 1., 2., 3., 4., 5.],
        [0., 1., 2., 3., 4., 5.],
        [0., 1., 2., 3., 4., 5.],
        [0., 1., 2., 3., 4., 5.],
        [0., 1., 2., 3., 4., 5.]]])
```

```
In [99]: a = np.array([[1, 2],
    ...:                [3, 4]])

In [100]: np.tile(a, (2, 3))
Out[100]:
array([[1, 2, 1, 2, 1, 2],
       [3, 4, 3, 4, 3, 4],
       [1, 2, 1, 2, 1, 2],
       [3, 4, 3, 4, 3, 4]])
```

```
In [106]: np.vstack((1, 2))
Out[106]:
array([[1],
       [2]])

In [107]: np.nanmax(a)
Out[107]: 4

In [108]: a.max(0)
Out[108]: array([3, 4])

In [109]: a.max(1)
Out[109]: array([2, 4])
```

```
In [110]: np.maximum(a, b)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[110], line 1
----> 1 np.maximum(a, b)

ValueError: operands could not be broadcast together with shapes (2,2) (2,3)

In [111]: a = np.array([[1, 5, 3],
     ...:               [7, 2, 6]])

In [112]: b = np.array([[4, 3, 8],
     ...:               [1, 9, 2]])

In [113]: np.maximum(a, b)
Out[113]:
array([[4, 5, 8],
       [7, 9, 6]])

In [114]: np.maximum(a, b)
Out[114]:
array([[4, 5, 8],
       [7, 9, 6]])

In [115]: np.sqrt(v @ v)
Out[115]: 1.0677078252031311

In [116]: logical_and(a,b)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[116], line 1
----> 1 logical_and(a,b)

NameError: name 'logical_and' is not defined

In [117]: a = np.array([[True, False, True],
     ...:               [False, True, True]])

In [118]: b = np.array([[True, True, False],
     ...:               [False, True, False]])

In [119]: np.logical_and(a, b)
Out[119]:
array([[ True, False, False],
       [False,  True, False]])

In [120]: np.logical_or(a,b)
Out[120]:
array([[ True,  True,  True],
       [False,  True,  True]])

In [121]: a & b
Out[121]:
array([[ True, False, False],
       [False,  True, False]])

In [122]:
     ...: a | b
Out[122]:
array([[ True,  True,  True],
       [False,  True,  True]])
```

```
In [124]: a = np.array([[1, 2],
     ...:                [3, 4]])

In [125]: linalg.inv(a)
Out[125]:
array([[-2. ,  1. ],
       [ 1.5, -0.5]])

In [126]: linalg.pinv(a)
Out[126]:
array([[-2. ,  1. ],
       [ 1.5, -0.5]])

In [127]: np.linalg.matrix_rank(a)
Out[127]: 2

In [128]: U, S, Vh = linalg.svd(a)

In [129]: V = Vh.T
```

```
In [131]: a = np.array([[4, 12, -16],
     ...:               [12, 37, -43],
     ...:               [-16, -43, 98]])

In [132]: linalg.cholesky(a)
Out[132]:
array([[ 2.,  6., -8.],
       [ 0.,  1.,  5.],
       [ 0.,  0.,  3.]])

In [133]:
     ...: D,V = linalg.eig(a)
```

```
In [135]: a = np.array([[6, 2],
     ...:               [2, 3]])

In [136]: b = np.array([[4, 1],
     ...:               [1, 2]])

In [137]: D, V = np.linalg.eig(np.linalg.inv(b) @ a)

In [138]: D,V = eigs(a, k=3)
C:\Users\jason\anaconda3\Lib\site-packages\scipy\sparse\linalg\_eigen\arpack\arpack.py:1272: Run
timeWarning: k >= N - 1 for N * N square matrix. Attempting to use scipy.linalg.eig instead.
  warnings.warn("k >= N - 1 for N * N square matrix. "

In [139]: Q,R = linalg.qr(a)

In [140]: P,L,U = linalg.lu(a)

In [141]: cg
Out[141]: <function scipy.sparse.linalg._isolve.iterative.cg(A, b, x0=None, tol=1e-05, maxiter=N
one, M=None, callback=None, atol=None)>

In [142]: np.fft.fft(a)
Out[142]:
array([[ 8.+0.j,  4.+0.j],
       [ 5.+0.j, -1.+0.j]])

In [143]: np.fft.ifft(a)
Out[143]:
array([[ 4. +0.j,  2. +0.j],
       [ 2.5+0.j, -0.5+0.j]])

In [144]: np.sort(a)
Out[144]:
array([[2, 6],
       [2, 3]])

In [145]: np.sort(a, axis=1)
Out[145]:
array([[2, 6],
       [2, 3]])

In [146]: I = np.argsort(a[:, 0]); b = a[I,:]
```

```
In [148]: Z = np.array([[1, 1],
     ...:               [1, 2],
     ...:               [1, 3]])

In [149]: y = np.array([1, 2, 2])

In [150]: x = linalg.lstsq(Z, y)
```

```
In [152]: t = np.linspace(0, 1, 100, endpoint=False)

In [153]: x = np.sin(2 * np.pi * 5 * t)

In [154]: q = 2

In [155]: new_length = int(np.ceil(len(x) / q))

In [156]: x_resampled = signal.resample(x, new_length)

In [157]: np.unique(a)
Out[157]: array([2, 3, 6])

In [158]: a.squeeze()
Out[158]:
array([[6, 2],
       [2, 3]])
```

Task 3

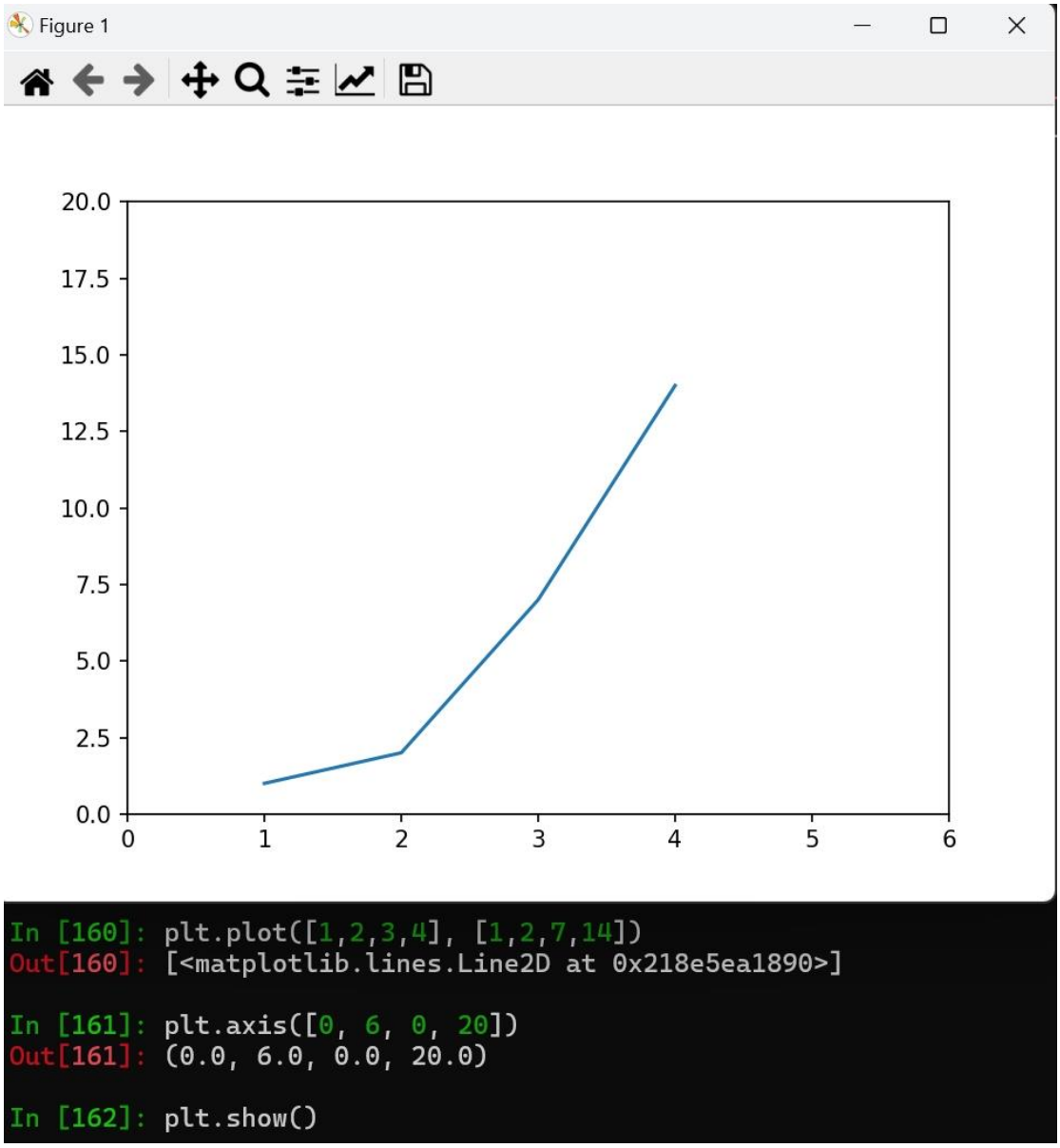
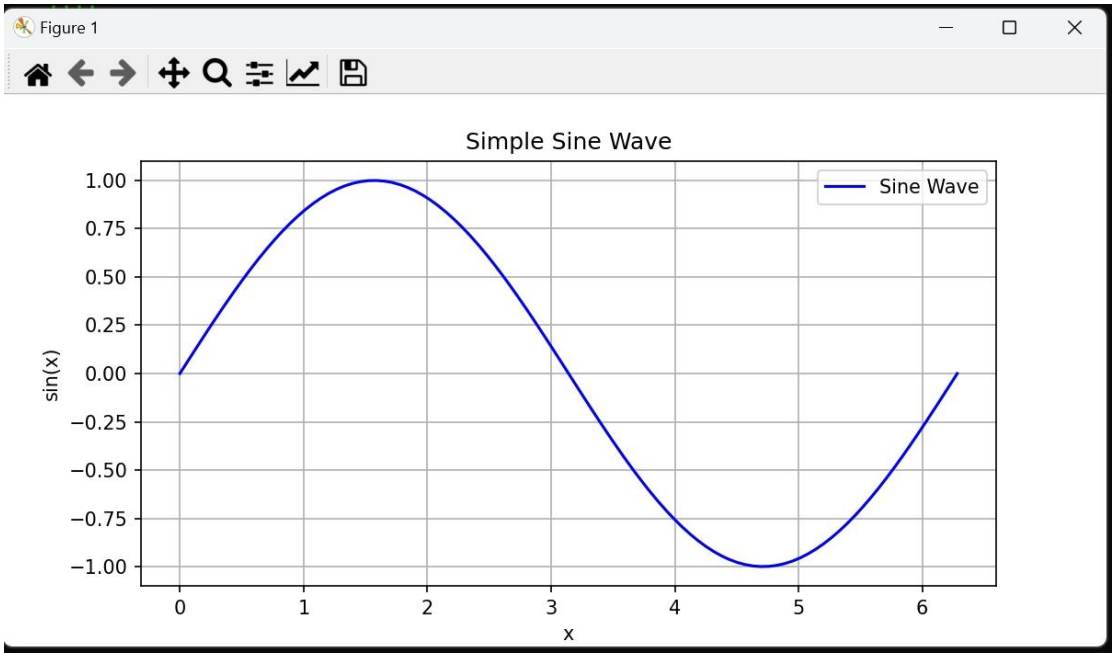


Figure 1
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
2
3
4
5
6
In [160]: plt.plot([1,2,3,4], [1,2,7,14])
Out[160]: [<matplotlib.lines.Line2D at 0x218e5ea1890>]
In [161]: plt.axis([0, 6, 0, 20])
Out[161]: (0.0, 6.0, 0.0, 20.0)
In [162]: plt.show()

Task 4




```
In [2]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...:
   ...: x = np.linspace(0, 2 * np.pi, 100)
   ...: y = np.sin(x)
   ...:
   ...: plt.figure(figsize=(8, 4))
   ...: plt.plot(x, y, label='Sine Wave', color='b')
   ...: plt.title('Simple Sine Wave')
   ...: plt.xlabel('x')
   ...: plt.ylabel('sin(x)')
   ...: plt.grid(True)
   ...: plt.legend()
   ...:
   ...: plt.show()
```

Task 5

https://github.com/Jasonliuuuu

Task 6

https://github.com/Jasonliuuuu/ELEC576