Assignment 0

Task 1

```
(base) wangchengwei@MacBook-Pro ~ % conda info

      active environment : base
     active env location : /Users/wangchengwei/opt/anaconda3
             shell level : 1
        user config file : /Users/wangchengwei/.condarc
  populated config files : /Users/wangchengwei/.condarc
           conda version : 4.14.0
     conda-build version : 3.21.8
          python version : 3.9.12.final.0
        virtual packages : __osx=10.16=0
                           __unix=0=0
                           __archspec=1=x86_64
        base environment : /Users/wangchengwei/opt/anaconda3  (writable)
       conda av data dir : /Users/wangchengwei/opt/anaconda3/etc/conda
   conda av metadata url : None
           channel URLs : https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch/osx-64
                          https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch/noarch
                          https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/osx-64
                          https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/noarch
                          https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/osx-64
                          https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/noarch
                          https://repo.anaconda.com/pkgs/main/osx-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/r/osx-64
                          https://repo.anaconda.com/pkgs/r/noarch
          package cache : /Users/wangchengwei/opt/anaconda3/pkgs
                          /Users/wangchengwei/.conda/pkgs
       envs directories : /Users/wangchengwei/opt/anaconda3/envs
                          /Users/wangchengwei/.conda/envs
               platform : osx-64
             user-agent : conda/4.14.0 requests/2.27.1 CPython/3.9.12 Darwin/24.6.0 OSX/10.16
                UID:GID : 501:20
             netrc file : None
           offline mode : False
```

Task 2

In [1]:
```
import numpy as np
import scipy.linalg as linalg
from scipy.sparse.linalg import eigs
from scipy import signal
from numpy.random import default_rng
```

In [2]:
```
a = np.array([[1., 2., 3.], [4., 5., 6.]])
b = np.array([[7., 8.], [9., 10.]])
c = np.array([[11., 12., 13.], [14., 15., 16.]])
d = np.array([[17., 18.], [19., 20.]])
v = np.array([1., 2., 3.])
print("a:\n", a)
print("b:\n", b)
print("c:\n", c)
print("d:\n", d)
print("v:\n", v)
m = np.block([[a, b], [c, d]])
print("m:\n", m)
n = np.block([[a, b], [c, d], [a, b]])
print("n:\n", n)
```

```
a:
 [[1. 2. 3.]
```

```
 [4. 5. 6.]]
b:
 [[ 7.  8.]
 [ 9. 10.]]
c:
 [[11. 12. 13.]
 [14. 15. 16.]]
d:
 [[17. 18.]
 [19. 20.]]
v:
 [1. 2. 3.]
m:
 [[ 1.  2.  3.  7.  8.]
 [ 4.  5.  6.  9. 10.]
 [11. 12. 13. 17. 18.]
 [14. 15. 16. 19. 20.]]
n:
 [[ 1.  2.  3.  7.  8.]
 [ 4.  5.  6.  9. 10.]
 [11. 12. 13. 17. 18.]
 [14. 15. 16. 19. 20.]
 [ 1.  2.  3.  7.  8.]
 [ 4.  5.  6.  9. 10.]]
```

In [3]:

```
print("a.ndim = ", a.ndim)
print("np.ndim(a) = ", np.ndim(a))
print("a.size = ", a.size)
print("np.size(a) = ", np.size(a))
print("a.shape = ", a.shape)
print("np.shape(a) = ", np.shape(a))
print("a.shape[1] = ", a.shape[1])
```

```
a.ndim =  2
np.ndim(a) =  2
a.size =  6
np.size(a) =  6
a.shape =  (2, 3)
np.shape(a) =  (2, 3)
a.shape[1] =  3
```

In [4]:

```
print("m[-1] = ", m[-1])
print("m[2, 3] = ", m[2, 3])
print("m[2] = ", m[2])
print("m[2, :] = ", m[2, :])
print("m[0:2] = ", m[0:2])
print("m[:2] = ", m[:2])
print("m[0:2, :] = ", m[0:2, :]) # first two rows
print("m[-2:] = ", m[-2:]) # last two rows
print("m[:, -2:] = ", m[:, -2:]) # last two columns
print("m[0:3, 2:4] = ", m[0:3, 2:4]) # first three rows, columns 3 and 4
```

```
m[-1] =  [14. 15. 16. 19. 20.]
m[2, 3] =  17.0
m[2] =  [11. 12. 13. 17. 18.]
m[2, :] =  [11. 12. 13. 17. 18.]
m[0:2] =  [[ 1.  2.  3.  7.  8.]
 [ 4.  5.  6.  9. 10.]]
m[:2] =  [[ 1.  2.  3.  7.  8.]
 [ 4.  5.  6.  9. 10.]]
m[0:2, :] =  [[ 1.  2.  3.  7.  8.]
 [ 4.  5.  6.  9. 10.]]
m[-2:] =  [[11. 12. 13. 17. 18.]
 [14. 15. 16. 19. 20.]]
m[:, -2:] =  [[ 7.  8.]
```

```
 [ 9. 10.]
 [17. 18.]
 [19. 20.]]
m[0:3, 2:4] =  [[ 3.  7.]
 [ 6.  9.]
 [13. 17.]]
```

In [5]:
```
m[np.ix_([1], [0, 2])] = 0
print("set row 1, column 0 and 2 to zero \n", m)
print("every two rows starting from row 0 to row 3 \n", m[0:3:2, :]) # every two rows st
print("every two rows in reverse order \n ", m[::-2, :]) # every two rows in reverse ord
print("append the first row to the end \n", m[np.r_[:len(m), 0]])
```

```
set row 1, column 0 and 2 to zero
 [[ 1.  2.  3.  7.  8.]
 [ 0.  5.  0.  9. 10.]
 [11. 12. 13. 17. 18.]
 [14. 15. 16. 19. 20.]]
every two rows starting from row 0 to row 3
 [[ 1.  2.  3.  7.  8.]
 [11. 12. 13. 17. 18.]]
every two rows in reverse order
  [[14. 15. 16. 19. 20.]
 [ 0.  5.  0.  9. 10.]]
append the first row to the end
 [[ 1.  2.  3.  7.  8.]
 [ 0.  5.  0.  9. 10.]
 [11. 12. 13. 17. 18.]
 [14. 15. 16. 19. 20.]
 [ 1.  2.  3.  7.  8.]]
```

In [6]:
```
print("transpose \n", m.T, "\nanother method \n", m.transpose())
print("conjugate transpose of m\n", m.conj().transpose(), "\nanother method \n", m.conj(
```

```
transpose
 [[ 1.  0. 11. 14.]
 [ 2.  5. 12. 15.]
 [ 3.  0. 13. 16.]
 [ 7.  9. 17. 19.]
 [ 8. 10. 18. 20.]]
another method
 [[ 1.  0. 11. 14.]
 [ 2.  5. 12. 15.]
 [ 3.  0. 13. 16.]
 [ 7.  9. 17. 19.]
 [ 8. 10. 18. 20.]]
conjugate transpose of m
 [[ 1.  0. 11. 14.]
 [ 2.  5. 12. 15.]
 [ 3.  0. 13. 16.]
 [ 7.  9. 17. 19.]
 [ 8. 10. 18. 20.]]
another method
 [[ 1.  0. 11. 14.]
 [ 2.  5. 12. 15.]
 [ 3.  0. 13. 16.]
 [ 7.  9. 17. 19.]
 [ 8. 10. 18. 20.]]
```

In [7]:
```
print("matrix multiple b@d:\n", b @ d)
print("element-wise multiple a*b:\n", b*d)
print("element-wise divide a/b:\n", b/d)
print("element-wise exponentiation a**3:\n", a**3)
print("matrix whose element is b > 8:\n", (b > 8))
print("find the indices where b > 8:\n", np.nonzero(b > 8))
```

```
b[b < 8] = 0
print("b with elements less than 8 zeroed out:\n", b)
b = b * (b > 9)
print("b with elements less than or equal to 9 zeroed out:\n", b)
```

```
matrix multiple b@d:
 [[271. 286.]
 [343. 362.]]
element-wise multiple a*b:
 [[119. 144.]
 [171. 200.]]
element-wise divide a/b:
 [[0.41176471 0.44444444]
 [0.47368421 0.5       ]]
element-wise exponentiation a**3:
 [[  1.   8.  27.]
 [ 64. 125. 216.]]
matrix whose element is b > 8:
 [[False False]
 [ True  True]]
find the indices where b > 8:
 (array([1, 1]), array([0, 1]))
b with elements less than 8 zeroed out:
 [[ 0.  8.]
 [ 9. 10.]]
b with elements less than or equal to 9 zeroed out:
 [[ 0.  0.]
 [ 0. 10.]]
```

In [8]:
```
a[:] = 3
print("set all values to the same scalar value:\n", a)

e = a.copy()
print("e is a copy of a:\n", e)

e = a[1, :].copy()
print("e is a sliced copy of a:\n", e)

e = a.flatten()
print("turn a into vector, producing a copy:\n", e)
```

```
set all values to the same scalar value:
 [[3. 3. 3.]
 [3. 3. 3.]]
e is a copy of a:
 [[3. 3. 3.]
 [3. 3. 3.]]
e is a sliced copy of a:
 [3. 3. 3.]
turn a into vector, producing a copy:
 [3. 3. 3. 3. 3. 3.]
```

In [9]:
```
print(np.arange(1., 11.), "\n")

print(np.r_[:10.], "\n")

print(np.arange(1., 11.)[:, np.newaxis], "\n")

print(np.zeros((3, 4)), "\n")

print(np.zeros((3, 4, 5)), "\n")
```

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]

[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

```
[[ 1.]
 [ 2.]
 [ 3.]
 [ 4.]
 [ 5.]
 [ 6.]
 [ 7.]
 [ 8.]
 [ 9.]
 [10.]]

[[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. 0. 0.]
  [0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0.]]]
```

In [10]:
```
print(np.ones((3, 4)), "\n")

print(np.eye(3), "\n")

print(v, "\n")

print(np.diag(v, 0))
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]

[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

[1. 2. 3.]

[[1. 0. 0.]
 [0. 2. 0.]
 [0. 0. 3.]]
```

In [11]:
```
rng = default_rng(42)
rng.random((3, 4))
```

Out[11]:
```
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 [12]:
```
np.linspace(1, 3, 4)
```

Out[12]:
```
array([1.        , 1.66666667, 2.33333333, 3.        ])
```

In [13]:
```
np.mgrid[0:9., 0:6.]
```

Out[13]:
```
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 [14]:
```
np.ix_(np.r_[0:9.], np.r_[0:6.])
```

Out[14]:
```
(array([[0.],
        [1.],
        [2.],
        [3.],
        [4.],
        [5.],
        [6.],
        [7.],
        [8.]]),
 array([[0., 1., 2., 3., 4., 5.]]))
```

In [15]:
```
np.meshgrid([1, 2, 4], [2, 4, 5])
```

Out[15]:
```
[array([[1, 2, 4],
        [1, 2, 4],
        [1, 2, 4]]),
 array([[2, 2, 2],
        [4, 4, 4],
        [5, 5, 5]])]
```

In [16]:
```
np.ix_([1, 2, 4], [2, 4, 5])
```

Out[16]:
```
(array([[1],
        [2],
        [4]]),
 array([[2, 4, 5]]))
```

In [17]:
```
np.tile(10, (6, 5))
```

Out[17]:
```
array([[10, 10, 10, 10, 10],
       [10, 10, 10, 10, 10],
       [10, 10, 10, 10, 10],
       [10, 10, 10, 10, 10],
       [10, 10, 10, 10, 10],
       [10, 10, 10, 10, 10]])
```

In [18]:
```
np.concatenate((a, b), 1)
```

Out[18]:
```
array([[ 3.,  3.,  3.,  0.,  0.],
       [ 3.,  3.,  3.,  0., 10.]])
```

In [19]:
```
np.concatenate((a, c))
```

Out[19]:
```
array([[ 3.,  3.,  3.],
       [ 3.,  3.,  3.],
       [11., 12., 13.],
       [14., 15., 16.]])
```

In [20]:
```
print(c.max())
print(c.max(0)) # maximum element of each column
print(c.max(1)) # maximum element of each row
print(np.maximum(a, c)) # dimensions should match

print(np.sqrt(v @ v))
print(np.logical_or(a, c))
```

```
16.0
[14. 15. 16.]
[13. 16.]
[[11. 12. 13.]
 [14. 15. 16.]]
3.7416573867739413
[[ True  True  True]
 [ True  True  True]]
```

In [21]:
```
print(linalg.inv(d))
print(linalg.pinv(d))
print(np.linalg.matrix_rank(a))
print(linalg.solve(d, a))
print(linalg.lstsq(a, b))
```

```
[[-10.    9. ]
 [  9.5  -8.5]]
[[-10.    9. ]
 [  9.5  -8.5]]
1
[[-3. -3. -3.]
 [ 3.  3.  3.]]
(array([[0.        , 0.55555556],
       [0.        , 0.55555556],
       [0.        , 0.55555556]]), array([], dtype=float64), 1, array([7.34846923e+00,
9.60875171e-17]))
```

In [22]:
```
U, S, Vh = linalg.svd(a)
V = Vh.T
print(Vh)
print(V)
print(U)
print(S)
```

```
[[-5.77350269e-01 -5.77350269e-01 -5.77350269e-01]
 [ 8.16496581e-01 -4.08248290e-01 -4.08248290e-01]
 [-9.55821271e-17 -7.07106781e-01  7.07106781e-01]]
[[-5.77350269e-01  8.16496581e-01 -9.55821271e-17]
 [-5.77350269e-01 -4.08248290e-01 -7.07106781e-01]
 [-5.77350269e-01 -4.08248290e-01  7.07106781e-01]]
[[-0.70710678  0.70710678]
 [-0.70710678 -0.70710678]]
[7.34846923e+00 9.12304898e-17]
```

In [23]:
```
print(linalg.cholesky(d)) # square matrix
```

```
[[4.12310563 4.36564125]
 [0.         0.9701425 ]]
```

In [24]:
```
D, V = linalg.eig(b) # square matrix
print(D)
print(V)
```

```
[ 0.+0.j 10.+0.j]
[[1. 0.]
 [0. 1.]]
```

In [25]:
```
D, V = linalg.eig(b, d) # two square matrices
print(D)
print(V)
```

```
[  0.+0.j -85.+0.j]
[[-1.          0.72701315]
 [-0.         -0.68662353]]
```

In [26]:
```
D, V = eigs(b, k=2)
print(D)
print(V)
```

```
[ 0.+0.j 10.+0.j]
[[1. 0.]
 [0. 1.]]
```

```
/Users/wangchengwei/opt/anaconda3/lib/python3.9/site-packages/scipy/sparse/linalg/eigen/arpack/arpack.py:1266: RuntimeWarning: 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 [27]:
```
Q, R = linalg.qr(b)
print(Q)
print(R)
```

```
[[ 1.  0.]
 [-0.  1.]]
[[ 0.  0.]
 [ 0. 10.]]
```

In [28]:
```
P, L, U = linalg.lu(b)
print(P)
print(L)
print(U)
```

```
[[1. 0.]
 [0. 1.]]
[[1. 0.]
 [0. 1.]]
[[ 0.  0.]
 [ 0. 10.]]
```

In [29]:
```
np.fft.fft(a)
```

Out[29]:
```
array([[9.+0.j, 0.+0.j, 0.+0.j],
       [9.+0.j, 0.+0.j, 0.+0.j]])
```

In [30]:
```
np.fft.ifft(a)
```

Out[30]:
```
array([[3.+0.j, 0.+0.j, 0.+0.j],
       [3.+0.j, 0.+0.j, 0.+0.j]])
```

In [31]:
```
np.sort(a)
```

Out[31]:
```
array([[3., 3., 3.],
       [3., 3., 3.]])
```

In [32]:
```
np.sort(a, axis=1)
```

Out[32]:
```
array([[3., 3., 3.],
       [3., 3., 3.]])
```

In [33]:
```
I = np.argsort(a[:, 0])
b = a[I, :]
print(I)
print(b)
```

```
[0 1]
[[3. 3. 3.]
 [3. 3. 3.]]
```

In [34]:
```
Z = np.array([[1, 1], [1, 2], [1, 3]])
y = np.array([1, 2, 2])

x = linalg.lstsq(Z, y)
print(x)
```

```
(array([0.66666667, 0.5       ]), 0.1666666666666669, 2, array([4.07914333, 0.6004912
2]))
```

In [35]:
```
x = np.linspace(0, 1, 10)
q = 2.5
signal.resample(x, int(np.ceil(len(x)/q)))
```

Out[35]:
```
array([0.27777778, 0.26914627, 0.5       , 0.95307595])
```

In [36]:
```
np.unique(a)
```

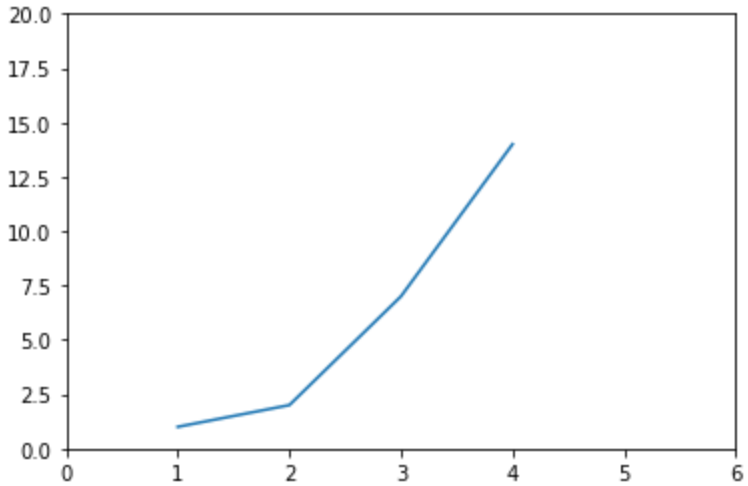
Out[36]:
```
array([3.])
```

In [37]:
```
a.squeeze()
```

Out[37]:
```
array([[3., 3., 3.],
       [3., 3., 3.]])
```

Task 3

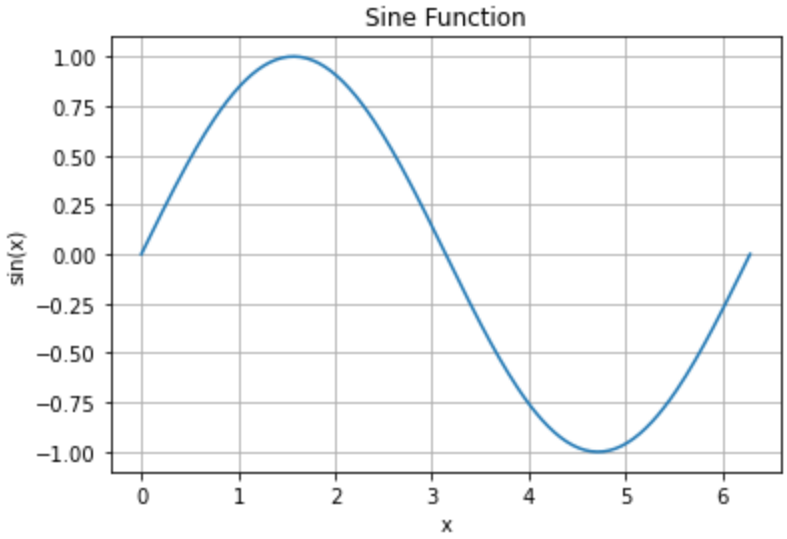
In [38]:
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,2,7,14])
plt.axis([0, 6, 0, 20])
plt.show()
```



Task 4

In [42]:
```
x = np.linspace(0, 2*np.pi, 200)
y = np.sin(x)

plt.plot(x, y)
plt.title("Sine Function")
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.grid(True)
plt.show()
```



Task 5

wccccw_Jade