APPM 4600 Lab 1
Henry Dyer
Darius Mirhosseini

3.1.1

Multiplying a python list by 3 creates three copies of the elements of the list, ie [1,2,3] becomes [1,2,3,1,2,3,1,2,3].

When using NumPy, multiplying a NumPy array [1,2,3] returns [3,6,9] as we would expect.

3.1.3

X is a NumPy array of size 100. The linspace command takes in a minimum and maximum value for the first two arguments and then the number of partitions as the third argument (in the previous case 100) and returns an array of containing the partitions.

3.2.1

X = np.linspace(0,9,10), Y = np.arange(0, 10, 1), both return arrays of size 10 with elements 0 through 9.

3.2.2

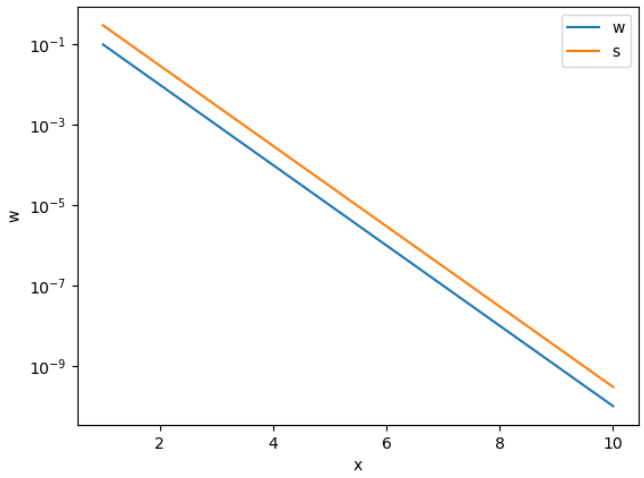
X[0:3]

3.2.3

Print('The first three entries of x are:" , X[0:3])

3.2.4/5

W = array([1.e-01, 1.e-02, 1.e-03, 1.e-04, 1.e-05, 1.e-06, 1.e-07, 1.e-08,1.e-09, 1.e-10])
W is an array containing



4.2.1

Set the lambda functions f(x) = cos(x) and g(x) = cos(2x) which produce orthogonal vectors, the dot product evaluates to ~ 1.33 e^-15 which is roughly zero as expected.

4.2.2

```
def matrixVectorProduct(A, x):
    if A.shape[1] != x.shape[0]:
        print('Incompatible System')
        return -1

    b = np.zeros([A.shape[0], 1])
    for i in range(A.shape[0]):
        b[i] = dotProduct(A[i], x, A.shape[1])
    return b
```

4.2.3

```
np.matmul(A, x)
```

NumPy is faster than the function we wrote since it has been heavily optimized compared to what we wrote.