Becca Blum
Nathan Evans
Peach Chaphuphuang

Lab 1
APPM 4600, Spring 2024

3.1.1 & 3.1.2

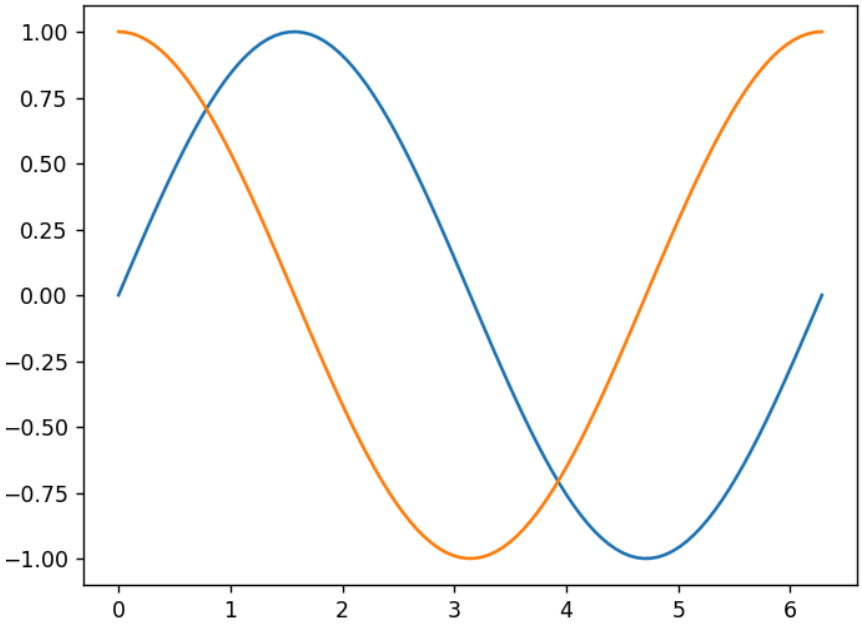
What happens when you multiply x = [1, 2, 3] by 3?
It is a 9x1 list with three “1,2,3” lists appended to each other:

[1, 2, 3, 1, 2, 3, 1, 2, 3]

What happens when you multiply y = [1, 2, 3] (numpy array) by 3?
You get the numpy array [3, 6, 9]

3.1.3

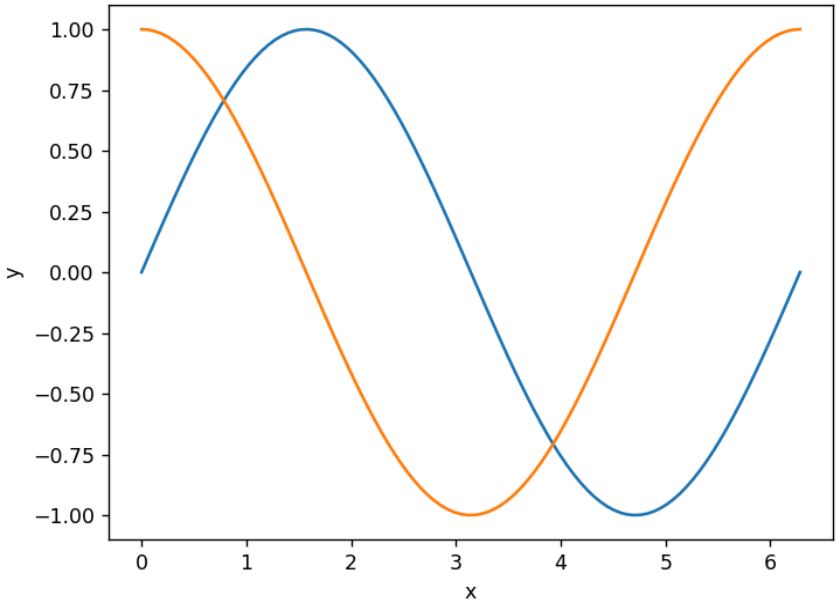
Plot of the sine and cos waves:



What size is X? How does linspace work?
X was defined using np.linspace(0,2*np.pi,100). The first and second input arguments in linspace specify the range of the values in the array. The third input specifies how many pieces the range should be evenly split up into. In other words, it is the number of indices we want the array to have. So the length of X in this case is 100 (aka there are 100 indices).

Becca Blum
Nathan Evans
Peach Chaphuphuang

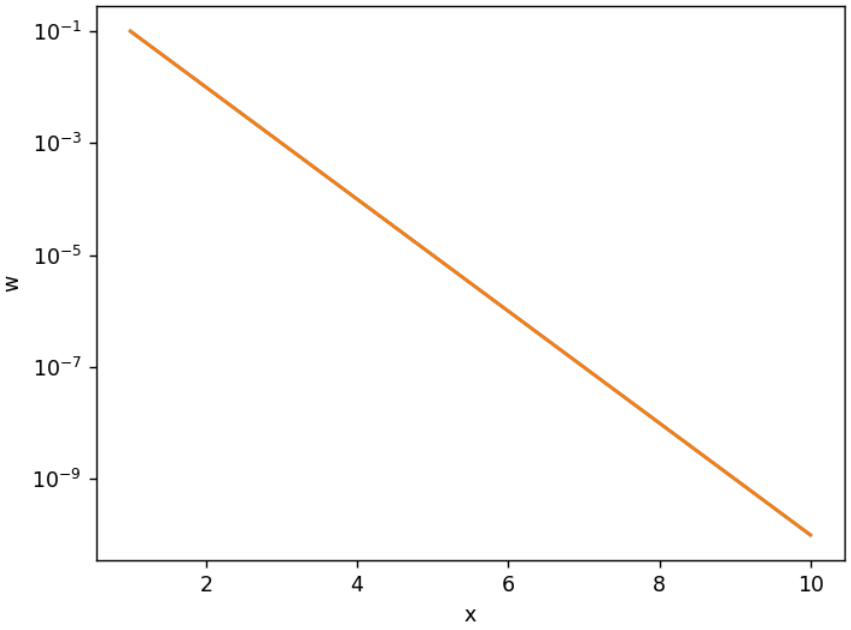
Plot the sine and cos waves with axis labels:



3.2

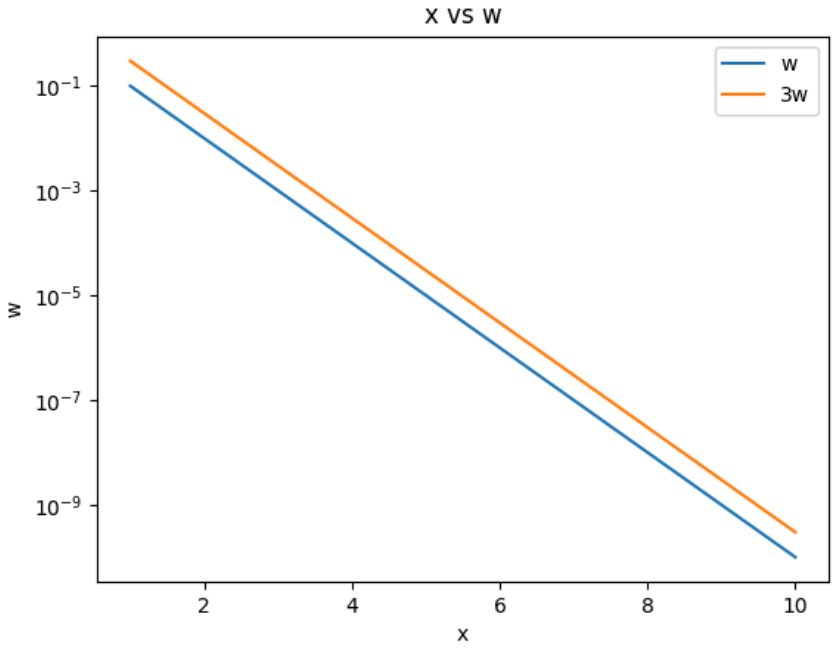
What are the entries of w = 10**(-np.linspace(1,10,10))?
w = [1.e-01 1.e-02 1.e-03 1.e-04 1.e-05 1.e-06 1.e-07 1.e-08 1.e-09 1.e-10]
They are “10 to the power of each entry in np.linspace(1,10,10)”

Plot of *x* versus *w*:



Becca Blum
Nathan Evans
Peach Chaphuphuang

Plot of *x* versus *w* and *x* versus *s* = *3*w*



4

What is the command for sending dp to the driver?
return dp

4.2

Built in command for dot product: np.dot(*array1*, *array2*)
Built in command for matrix multiplication: np.matmul(*array1*, *array2*)

Which is faster? Your code or Numpy?

When using the code given for the dot product calculation with a vector of length 1000, we found that it took ~999 ms for the calculation to complete. However, the built-in function np.dot() took ~0 ms.

When using a 100 long vector with a 100x100 matrix, we found that our code was ~3 times slower than the built-in function np.matmul().

Becca Blum
Nathan Evans
Peach Chaphuphuang

Example of the code written and used for the lab (author: Peach)

```
import numpy as np
import numpy.linalg as la
import math
import time

def driver():

    # Create a 3x3 matrix
    mat1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
    # Create a 3x1 vector
    vec1 = np.array([10,11,12]) #only vector (column = 1 only!!)
    # Find the multiplication of the matrix and vector above
    mv = matVecMulti(mat1,vec1)

    # Print all the results
    print("Matrix:")
    print(mat1)
    print("\nVector:")
    print(vec1)
    print("\nThe dot product is : ", mv)

    # Check with built-in function in Numpy
    # If the dimension of column is not equal to the length of vector,...
    # ... do not use dot()
    if len(mat1) == len(vec1):
        print("The dot product using 'dot()' is : ", mat1.dot(vec1))
        print("The dot product using 'np.matmul()' is : ", np.matmul(mat1,
vec1))

#########################################################
    # Test with bigger matrix and vector
    a = np.random.random( (100,100) )
    b = np.random.rand(100)

    start = time.time()
    mv2 = matVecMulti(a,b)
    end = time.time()
    print("The time of execution of above program is :",(end-start) *
10**3, "ms")

    start = time.time()
    mv2 = a.dot(b)
    end = time.time()
```

Becca Blum
Nathan Evans
Peach Chaphuphuang

```
    print("The time of execution of above program is :",(end-start) *
10**3, "ms")

    return

# My code to find matrix vector multiplication
def matVecMulti(matrix,vector):
    # Check if the dimension of column is equal to the length of vector
    if len(matrix) != len(vector):
        print("Cannot find the result \nsince incompatible matrix and
vector dimensions for multiplication")
    else:
        # Create matrix with zeros inside
        result = np.zeros(len(matrix[0]))
        # Rows
        for i in range(len(matrix)):
            # Columns
            for j in range(len(matrix[0])):
                result[i] = result[i] + matrix[i,j] * vector[j]

        return result
driver()
```