# PreReq1

August 31, 2025

Min Je (John) Kim FA25 EE 102 Pre-requisite #1: Vectors, projections, and linear algebra 8/30/2025

[6]:
```
import numpy as np



# Problem 5 Part A : Check for Linear Independence of Vactors
print("Problem 5 Part A : Check for Linear Independence of Vactors")
u = np.array([1,2,3])
v = np.array([4,5,6])
w = np.array([7,8,9])

A = np.array([u,v,w])

rank = np.linalg.matrix_rank(A)
print("Rank:", rank)


# If the Rank is equal to the number of Variables/Vector Space Dimerntions, The␣
 ↪Vectors are Linearly Independent because they can span the entire vector␣
 ↪space with a combination of the 3 vectors.
```

```
Problem 5 Part A : Check for Linear Independence of Vactors
Rank: 2
```

[3]:
```
import numpy as np



# Problem 5 Part B : Compute Orthogonal Projection
print("Problem 5 Part B : Compute Orthogonal Projection")
x = np.array([3,4])
x_Axis_Unit = np.array([1,0])
y_Axis_Unit = np.array([0,1])

def OrthogonalProjection(Vector, Subspace, ReturnScalar = False, Print = True):
    """OrthogonalProjection(Vector, Subspace, Scalar = False, Print = True)
```

```
    Args:
        Vector (np.array): The Vector to Project onto the New Subspace
        Subspace (np.array): The Subspace to be Projected on to
        ReturnScalar (Boolean): If True, will output the scalar. If False, will␣
↪output the Projected Vector
        Print (Boolean)L If True, will print before Return

    Returns:
        _type_: Returns the Scalar or Projected Vector based on Arguments Input
    """

    SubspaceMagnitude = np.linalg.norm(Subspace)
    SubspaceNormScalar = SubspaceMagnitude**2
    OrthVal = np.dot(Vector,Subspace)
    Scalar = OrthVal/SubspaceNormScalar

    if ReturnScalar:
        Scalar = round(Scalar,4)
        if Print:
            print(Scalar)
        return Scalar

    OrthProjection = np.multiply(Scalar,Subspace)

    if Print:
        print(OrthProjection)
    return OrthProjection

x_Axis_Projection = OrthogonalProjection(x, x_Axis_Unit, False, False)
print('X Axis Projection:', x_Axis_Projection)
y_Axis_Projection = OrthogonalProjection(x, y_Axis_Unit, False, False)
print('Y Axis Projection:', y_Axis_Projection)
```

```
Problem 5 Part B : Compute Orthogonal Projection
X Axis Projection: [3. 0.]
Y Axis Projection: [0. 4.]
```

[4]:
```
import numpy as np



# Problem 5 Part C : Express x in the new Basis B = [[1,1],[1,-1]]
print("Problem 5 Part C : Express x in the new Basis B = [[1,1],[1,-1]]")

def OrthogonalProjection(Vector, Subspace, ReturnScalar = False, Print = True):
    """OrthogonalProjection(Vector, Subspace, Scalar = False, Print = True)
```

```
    Args:
        Vector (np.array): The Vector to Project onto the New Subspace
        Subspace (np.array): The Subspace to be Projected on to
        ReturnScalar (Boolean): If True, will output the scalar. If False, will␣
↪output the Projected Vector
        Print (Boolean)L If True, will print before Return

    Returns:
        _type_: Returns the Scalar or Projected Vector based on Arguments Input
    """

    SubspaceMagnitude = np.linalg.norm(Subspace)
    SubspaceNormScalar = SubspaceMagnitude**2
    OrthVal = np.dot(Vector,Subspace)
    Scalar = OrthVal/SubspaceNormScalar

    if ReturnScalar:
        Scalar = round(Scalar,4)
        if Print:
            print(Scalar)
        return Scalar

    OrthProjection = np.multiply(Scalar,Subspace)

    if Print:
        print(OrthProjection)
    return OrthProjection

B = [np.array([1, 1]), np.array([1, -1])]
A1 = OrthogonalProjection(x, B[0], True, False)
A2 = OrthogonalProjection(x, B[1], True, False)
print(f"x in terms of the new Basis B: <{A1}, {A2}>")
```

```
Problem 5 Part C : Express x in the new Basis B = [[1,1],[1,-1]]
x in terms of the new Basis B: <3.5, -0.5>
```