Max Rotcliff

$$A = \begin{bmatrix} 4 & 1 & 2 \\ 0 & 3 & -1 \\ 0 & 0 & 2 \end{bmatrix} \qquad \begin{vmatrix} 4-\lambda & 1 & 2 \\ 0 & 3-\lambda & -1 \\ 0 & 0 & 2-\lambda \end{vmatrix} = 0$$

Part 1

$$(4-\lambda)\left((3-\lambda)(2-\lambda) - (0)(0)\right) - (1)\left((0)(2-\lambda) - (0)(0)\right) + 2\left((0)(0) - (0)(3-\lambda)\right) = 0$$

$$(4-\lambda)(3-\lambda)(2-\lambda) = 0$$

$\lambda_1 = 4,\ \lambda_2 = 3,\ \lambda_3 = 2$

$(A - \lambda_1) V_1 = 0$

$$\begin{bmatrix} 4-4 & 1 & 2 \\ 0 & 3-4 & -1 \\ 0 & 0 & 2-4 \end{bmatrix} \cdot V_1 = 0 \rightarrow \left[\begin{array}{ccc|c} 0 & 1 & 2 & 0 \\ 0 & -1 & -1 & 0 \\ 0 & 0 & -2 & 0 \end{array}\right] \hookrightarrow \left[\begin{array}{ccc|c} 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 0 \end{array}\right] \hookrightarrow \left[\begin{array}{ccc|c} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$r_2 = r_2 + r_1$; $r_3 = r_3 - 2r_2$; $r_1 = r_1 - r_2$

$$V_1 = \begin{bmatrix} x_1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 4-3 & 1 & 2 \\ 0 & 3-3 & -1 \\ 0 & 0 & 2-3 \end{bmatrix} \cdot V_2 = 0 \rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 2 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 \end{array}\right] \begin{array}{l} r_3 = r_3 + r_2 \\ r_2 = -r_2 \\ r_1 = r_1 - 2r_2 \end{array} \hookrightarrow \left[\begin{array}{ccc|c} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \begin{array}{l} x_1 = -x_2 \\ x_3 = 0 \\ x_2 = 0 \end{array}$$

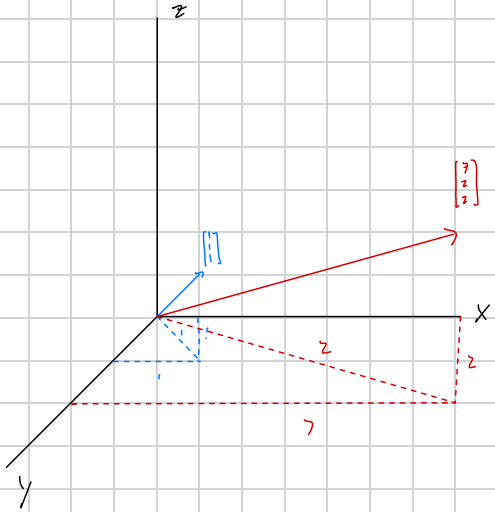
$$V_2 = \begin{bmatrix} -x_2 \\ x_2 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}(-1) = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 4-2 & 1 & 2 \\ 0 & 3-2 & -1 \\ 0 & 0 & 2-2 \end{bmatrix} \cdot V_3 = 0 \hookrightarrow \left[\begin{array}{ccc|c} 2 & 1 & 2 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \begin{array}{l} r_1 = r_1 - r_2 \\ r_1 = \frac{1}{2} r_1 \end{array} \hookrightarrow \left[\begin{array}{ccc|c} 1 & 0 & \frac{3}{2} & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \begin{array}{l} x_1 = -\frac{3}{2} x_3 \\ x_2 = x_3 \\ x_3 \end{array}$$

$$V_2 = \begin{bmatrix} -\frac{3}{2} x_3 \\ x_3 \\ x_3 \end{bmatrix} = \begin{bmatrix} -\frac{3}{2} \\ 1 \\ 1 \end{bmatrix} \cdot (-2) = \begin{bmatrix} 3 \\ -2 \\ -2 \end{bmatrix}$$

$$\lambda_1 = 4,\ V_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \lambda_2 = 3\ V_2 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \qquad \lambda_3 = 2,\ V_3 = \begin{bmatrix} 3 \\ -2 \\ -2 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 1 & 2 \\ 0 & 3 & -1 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4+1+2 \\ 0+3-1 \\ 0+0+2 \end{bmatrix} = \begin{bmatrix} 7 \\ 2 \\ 2 \end{bmatrix}$$



Part 2

```
[ ] import numpy as np
    import scipy.linalg as la
```

```
[ ] A = np.array([[4, 1, 2, 0, 1, 3, 0, 1],
                  [1, 3, -1, 0, 2, 1, 4, 0],
                  [2, -1, 2, 1, 0, 2, 3, 1],
                  [0, 0, 1, 3, -1, 0, 1, 2],
                  [1, 2, 0, -1, 4, 1, 0, 3],
                  [3, 1, 2, 0, 1, 5, 2, 1],
                  [0, 4, 3, 1, 0, 2, 3, 1],
                  [1, 0, 1, 2, 3, 1, 1, 4]])

    values, vectors = la.eig(A)

    for count, i in enumerate(values):
        print("eigen value %d: %f" % (count, i))

    for count, i in enumerate(vectors):
        print("eigen vector %d: %s" % (count, i))
```

```
eigen value 0: 12.031111
eigen value 1: -3.334960
eigen value 2: -0.446511
eigen value 3: 1.292989
eigen value 4: 1.760697
eigen value 5: 5.241308
eigen value 6: 5.533839
eigen value 7: 5.921527
eigen vector 0: [-0.38841179 -0.21937688 -0.12080162 -0.34374679 -0.62381882  0.34185429
  0.39088879 -0.09719495]
eigen vector 1: [-0.31207468  0.54253039 -0.06460955  0.061577   -0.3688439   0.14171829
 -0.65855444  0.10237286]
eigen vector 2: [-0.31644707  0.49027574  0.25884817 -0.53913595  0.34664548 -0.17922631
  0.19326535 -0.33493932]
eigen vector 3: [-0.12321074 -0.05185009  0.47402048  0.29885564 -0.45017662 -0.66307119
  0.08541012 -0.13602682]
eigen vector 4: [-0.30322784 -0.18389186  0.55720527 -0.11239322  0.16827177  0.15634285
 -0.02389975  0.70554733]
eigen vector 5: [-0.5084548   0.04731083  0.04176741  0.66939315  0.27226026  0.31856328
  0.21393502 -0.26082042]
eigen vector 6: [-0.40787644 -0.60169573 -0.13846266 -0.18627712  0.20469242 -0.20867381
 -0.50810749 -0.27223339]
eigen vector 7: [-0.34581706  0.12799821 -0.59844854  0.05143499  0.08741945 -0.4709519
  0.25367894  0.45765702]
<ipython-input-26-f37043c753b1>:13: ComplexWarning: Casting complex values to real discards the imaginary part
  print("eigen value %d: %f" % (count, i))
```

I Started by googling how to find eigenvectors in python where I learned to import numpy and scipy, I had already imported numpy cause I figured I would need it but I hadnt heard of scipy up till now. Then I used the google colab tool tips to help find the eigen values and vectors, I started by just printing them and then decided to format them a little, in the future I would want to format the vectors better and get rid of the complex warning and then I would mess with other matrix operations, might also be interesting to read through numpy documentation but that seems like a pain