NE250_HW03_mnegus-prob5

October 15, 2017

1 NE 250 – Homework 3

1.1 Problem 5

10/20/2017

```
In [1]: import numpy as np
```

The inverse, $\mathbf{A}^{-1}$ of a square matrix, $\mathbf{A}$, is equal to the adjugate of the matrix, $\mathbf{A}^\dagger$ divided by the determinant of $\mathbf{A}$.

$$\mathbf{A}^{-1} = \frac{\mathbf{A}^\dagger}{\det \mathbf{A}}$$

We can manipulate this expression to find

$$(\det \mathbf{A})\nVdash = \mathbf{A}^\dagger \mathbf{A}$$

Since we are looking for a self-adjugate matrix, $\mathbf{A}^\dagger = \mathbf{A}$, and

$$(\det \mathbf{A})\nVdash = \mathbf{A}^2.$$

Then, taking the square root of both sides,

$$\left(\sqrt{\det \mathbf{A}}\right)\nVdash = \mathbf{A}.$$

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} \sqrt{\det \mathbf{A}} & 0 & 0 \\ 0 & \sqrt{\det \mathbf{A}} & 0 \\ 0 & 0 & \sqrt{\det \mathbf{A}} \end{bmatrix}.$$

$$a = e = i = \sqrt{\det \mathbf{A}}$$

$$\mathbf{A} = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} =$$

and

$$\det \mathbf{A} = a \begin{vmatrix} a & 0 \\ 0 & a \end{vmatrix}$$

$$\det \mathbf{A} = a(a^2)$$

$$\det \mathbf{A} = a^3.$$

We know that $a = \sqrt{\det \mathbf{A}}$, so

$$a = \sqrt{a^3}$$

$$a = a^{\frac{3}{2}}$$

$$a = 1$$

$$\boxed{\mathbf{A} = \mathbb{1}}$$

We can test this using the functions defined in problem 4.

```
In [2]: def adjugate(matrix):
            cofactor_matrix = cofactor(matrix)
            adjugate_matrix = transpose(cofactor_matrix)
            return adjugate_matrix

        def cofactor(matrix):
            minors_matrix = minors(matrix)
            cofactor_matrix = np.copy(minors_matrix)
            for i in range(len(cofactor_matrix)):
                for j in range(len(cofactor_matrix[0])):
                        cofactor_matrix[i,j] *= (-1)**(i+j)
            return cofactor_matrix

        def transpose(matrix):
            transpose_matrix = np.empty_like(matrix)
            for i in range(len(matrix)):
                for j in range(len(matrix[0])):
                    transpose_matrix[j,i] = matrix[i,j]
            return transpose_matrix

        def minors(matrix):
            minors_matrix = np.empty_like(matrix)
            for i in range(len(minors_matrix)):
                for j in range(len(minors_matrix[0])):
                    minors_matrix[i,j] = minor(matrix,i,j)
            return minors_matrix

        def minor(matrix,i,j):
            submatrix = np.copy(matrix)
            submatrix = np.delete(submatrix,i,axis=0)
            submatrix = np.delete(submatrix,j,axis=1)
            minor_ij = determinant(submatrix)
            return minor_ij

        def determinant(matrix):
            assert len(matrix) == len(matrix[0])
```

```
            if len(matrix) == 2:
                return matrix[0,0]*matrix[1,1]-matrix[0,1]*matrix[1,0]
            else:
                signed_minors = []
                for j in range(len(matrix[0])):
                    if (j+2)%2 == 1:
                        sign = -1
                    else: sign = 1
                    signed_minors.append(matrix[0,j]*sign*minor(matrix,0,j))
                return sum(signed_minors)
```

```
In [3]: A = np.identity(3)
        print('A = \n',A)
        print('Adjugate of A = \n',adjugate(A))
```

```
A =
 [[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]
Adjugate of A =
 [[ 1. -0.  0.]
 [-0.  1. -0.]
 [ 0. -0.  1.]]
```