7. Matrix examples

7.1. Geometric transformations

Let us create a rotation matrix, and use it to rotate a set of points $\pi/3$ radians (60 deg). The result is in Figure 7.1.

```
In [ ]: Rot = lambda theta: [[np.cos(theta), -np.sin(theta)],
        [np.sin(theta), np.cos(theta)]]
        R = Rot(np.pi/3)
        R
```

```
Out[ ]: [[0.5000000000000001, -0.8660254037844386],
         [0.8660254037844386, 0.5000000000000001]]
```

```
In [ ]: #create a list of 2-D points
        points =
        ↪  np.array([[1,0],[1.5,0],[2,0],[1,0.25],[1.5,0.25],[1,0.5]])
        #Now rotate them
        rpoints = np.array([R @ p for p in points])
        #Show the two sets of points
        import matplotlib.pyplot as plt
        plt.ion()
        plt.scatter([c[0] for c in points], [c[1] for c in points])
        plt.scatter([c[0] for c in rpoints],[c[1] for c in rpoints])
        plt.show()
```

7.2. Selectors

Reverser matrix. The reverser matrix can be created from an identity matrix by reversing the order of its rows. The numpy function `np.flip()` can be used for this purpose.

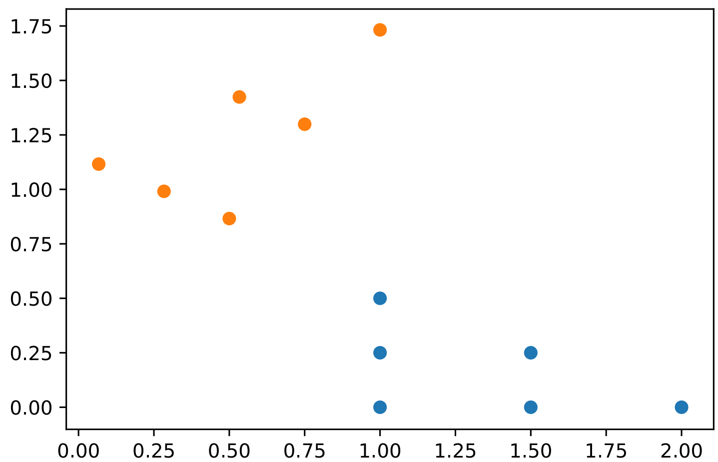

Figure 7.1.: Counterclockwise rotation by 60 degrees applied to six points.

In [ ]:
```
reverser = lambda n: np.flip(np.eye(n),axis=0)
A = reverser(5)
A
```

Out[ ]:
```
array([[0., 0., 0., 0., 1.],
       [0., 0., 0., 1., 0.],
       [0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.]])
```

Permutation matrix. Let's create a permutation matrix and use it to permute the entries of a vector. In Python, you can directly pass the permuted indexes to the vector.

In [ ]:
```
A = np.array([[0,0,1], [1,0,0], [0,1,0]])
x = np.array([0.2, -1.7, 2.4])
A @ x # Permutes entries of x to [x[2], x[0], x[1]]
```

Out[ ]:
```
array([ 2.4,  0.2, -1.7])
```

In [ ]:
```
x[[2,0,1]]
```

Out[ ]:
```
array([ 2.4,  0.2, -1.7])
```

7.3. Incidence matrix

Incidence matrix of a graph. We create the incidence matrix of the network shown in Figure 7.3 in VMLS.

In [ ]:
```
A = np.array([[-1,-1,0,1,0], [1,0,-1,0,0], [0,0,1,-1,-1],
↪  [0,1,0,0,1]])
xcirc = np.array([1,-1,1,0,1]) #A circulation
A @ xcirc
```

Out[ ]:
```
array([0, 0, 0, 0])
```

In [ ]:
```
s = np.array([1,0,-1,0,]) # A source vector
x = np.array([0.6,0.3,0.6,-0.1,-0.3]) #A flow vector
A @ x + s #Total incoming flow at each node
```

Out[ ]:
```
array([1.11022302e-16, 0.00000000e+00, 0.00000000e+00,
↪  0.00000000e+00])
```

Dirichlet energy. On page 135 of VMLS we compute the Dirichlet energy of two potential vectors associated with the graph of Figure 7.2 in VMLS.

In [ ]:
```
A = np.array([[-1,-1,0,1,0], [1,0,-1,0,0], [0,0,1,-1,-1],
↪  [0,1,0,0,1]])
vsmooth = np.array([1,2,2,1])
np.linalg.norm(A.T @ vsmooth)**2 #Dirichlet energy of vsmooth
```

Out[ ]:
```
2.9999999999999996
```

In [ ]:
```
vrough = np.array([1,-1, 2, -1])
np.linalg.norm(A.T @ vrough)**2 # Dirichlet energy of vrough
```

Out[ ]:
```
27.0
```

7.4. Convolution

The numpy function `np.convolve()` can be used to compute the convolution of the vectors `a` and `b`. Let's use this to find the coefficients of the polynomial

$$p(x) = (1+x)(2-x+x^2)(1+x-2x^2) = 2+3x-3x^2-x^3+x^4-2x^5$$