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

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

In [ ]:
```
a = np.array([1,1]) # coefficients of 1+x
b = np.array([2,-1,1]) # coefficients of 2-x+x^2
c = np.array([1,1,-2]) # coefficients of 1+x-2x^2
d = np.convolve(np.convolve(a,b),c) # coefficients of product
d
```

Out[ ]:
```
array([ 2,  3, -3, -1,  1, -2])
```

Let's write a function that creates a Toeplitz matrix and check it against the `conv` function. We will also confirm that Python is using a very efficient method for computing the convolution.

To construct the Toeplitz matrix $T(b)$ defined in equation (7.3) of VMLS, we can first create a zero matrix of the correct dimensions ($(n+m-1) \times n$) and then add the coefficients b_i one by one. Single-index indexing comes in handy for this purpose. The single-index indexes of the elements b_i in the matrix$T(b)$ are $i, i+m+n, i+2(m+n), \ldots, i+(n-1)(m+n)$.

In [ ]:
```
b = np.array([-1,2,3])
a = np.array([-2,3,-1,1])
def toeplitz(b,n):
    m = len(b)
    T = np.zeros((n+m-1,n))
    for j in range(n):
        T[j:j+m,j] = b
    return T
Tb = toeplitz(b,len(a))
Tb
```

Out[ ]:
```
array([[-1.,  0.,  0.,  0.],
       [ 2., -1.,  0.,  0.],
       [ 3.,  2., -1.,  0.],
       [ 0.,  3.,  2., -1.],
       [ 0.,  0.,  3.,  2.],
       [ 0.,  0.,  0.,  3.]])
```

In [ ]:
```
Tb @ a, np.convolve(b,a)
```

Out[ ]:
```
(array([2.,-7., 1., 6.,-1., 3.]), array([2,-7, 1, 6,-1, 3]))
```

In [ ]:
```
import time
m = 2000
n = 2000
b = np.random.normal(size = n)
a = np.random.normal(size = m)
start = time.time()
ctoep = toeplitz(b,n) @ a
end = time.time()
print(end - start)
```

```
0.05798077583312988
```

In [ ]:
```
start = time.time()
cconv = np.convolve(a,b)
end = time.time()
print(end - start)
```

```
0.0011739730834960938
```

In [ ]:
```
np.linalg.norm(ctoep - cconv)
```

Out[ ]:
```
1.0371560230890881e-12
```