In [ ]:
```
import numpy as np

class Convolution1d :
    def __init__(self, filt) :
        self.__filt = filt
        self.__r = filt.size
        self.T = TransposedConvolution1d(self.__filt)

    def __matmul__(self, vector) :
        r, n = self.__r, vector.size
        def filtered_dot_product(x : np.ndarray, idx : int) -> np.float64:
            reduce_x = x[idx:idx+r]
            return self.__filt.dot(reduce_x)
        dot = np.vectorize(filtered_dot_product, signature='(n),()->()')

        return dot(vector, np.arange(n-r+1))

class TransposedConvolution1d :
    '''
    Transpose of 1-dimensional convolution operator used for the
    transpose-convolution operation A.T@(...)
    '''
    def __init__(self, filt) :
        self.__filt = filt
        self.__r = filt.size

    def __matmul__(self, vector) :
        r = self.__r
        n = vector.size + r - 1
        a = np.concatenate((np.zeros(vector.size-1), self.__filt, np.zeros(vector.s
        def filtered_dot_product(x : np.ndarray, idx : int) -> np.float64:
            reduce_a = np.flip(a[idx:idx+vector.size])
            return reduce_a.dot(x)
        dot = np.vectorize(filtered_dot_product, signature='(n),()->()')

        return dot(vector, np.arange(n))

def huber_loss(x) :
    return np.sum( (1/2)*(x**2)*(np.abs(x)<=1) + (np.sign(x)*x-1/2)*(np.abs(x)>1)
def huber_grad(x) :
    return x*(np.abs(x)<=1) + np.sign(x)*(np.abs(x)>1)


r, n, lam = 3, 20, 0.1

np.random.seed(0)
k = np.random.randn(r)
b = np.random.randn(n-r+1)
A = Convolution1d(k)
# from scipy.linalg import circulant
# A = circulant(np.concatenate((np.flip(k),np.zeros(n-r))))[r-1:,:]


x = np.zeros(n)
alpha = 0.01
for _ in range(100) :
    x = x - alpha*(A.T@(huber_grad(A@x-b))+lam*x)

print(huber_loss(A@x-b)+0.5*lam*np.linalg.norm(x)**2)

# I get answer 0.4587586843129764 when running
```

```
0.4587586843129764
```