EE5178: Modern Computer Vision
Assignment 3: Filtering and hybrid images

-B Aditi
MM19B022

In [1]:

```
#importing required packages
import numpy as np
import matplotlib.pyplot as plt
from numpy.fft import rfft, irfft
import cv2
import skimage as sk
from skimage.transform import rescale
```

Filtering

In [2]:

```
#setting up input signal
k = np.array([i for i in range(16)])
print("base array is", k)
```

```
base array is [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
```

In [3]:

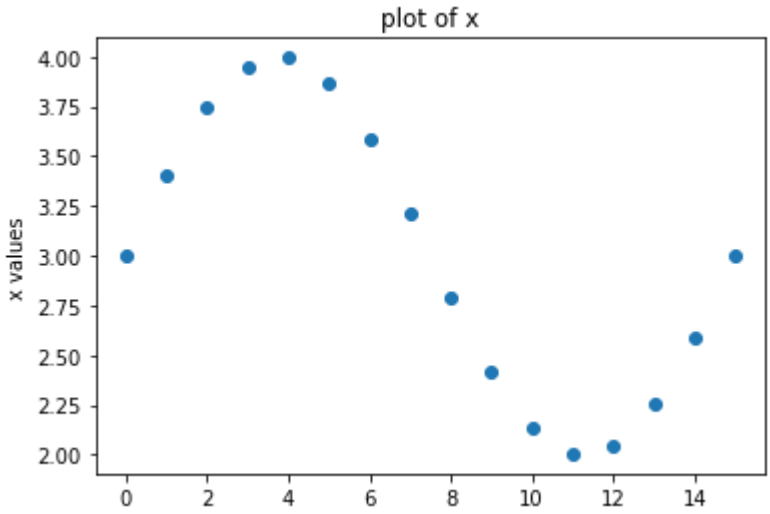
```
x = np.array([3]) + np.sin(2*np.pi*k/15)
print("x =", x)
```

```
x = [3.         3.40673664 3.74314483 3.95105652 3.9945219  3.8660254
 3.58778525 3.20791169 2.79208831 2.41221475 2.1339746  2.0054781
 2.04894348 2.25685517 2.59326336 3.        ]
```

In [11]:

```
plt.scatter(k,x)
plt.ylabel('x values')
plt.title('plot of x')
n = len(x)
print("size of x =",n)
```

```
size of x = 16
```



Filter 1: $y_k = x_{k+1} - x_k$

In [13]:

```
x_padded = np.zeros(n+1)
x_padded[0:16] = x
print("size of x after 0 padding is", len(x_padded))
y1 = np.zeros(n)
for i in range(n):
    y1[i] = x_padded[i+1] - x_padded[i]
print("filter 1 =",y1)
```

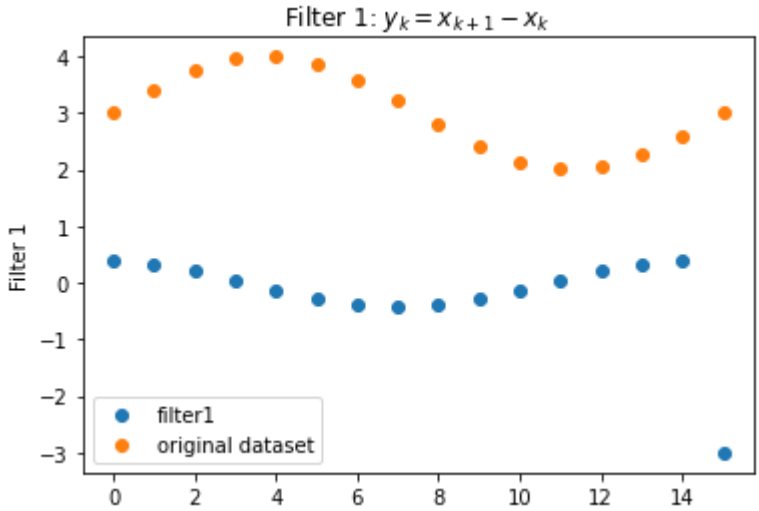
```
size of x after 0 padding is 17
filter 1 = [ 0.40673664  0.33640818  0.20791169  0.04346538 -0.1284964
9 -0.27824015
 -0.37987356 -0.41582338 -0.37987356 -0.27824015 -0.12849649  0.043465
38
  0.20791169  0.33640818  0.40673664 -3.        ]
```

In [22]:

```
plt.scatter(k,y1, label='filter1')
plt.scatter(k,x, label = 'original dataset')
plt.legend()
plt.ylabel('Filter 1')
plt.title('Filter 1: $y_k = x_{k+1} - x_k$')
```

Out[22]:

```
Text(0.5, 1.0, 'Filter 1: $y_k = x_{k+1} - x_k$')
```



Since the given dataset is linear and space invariant, we will implement convolution and filter in fourier series

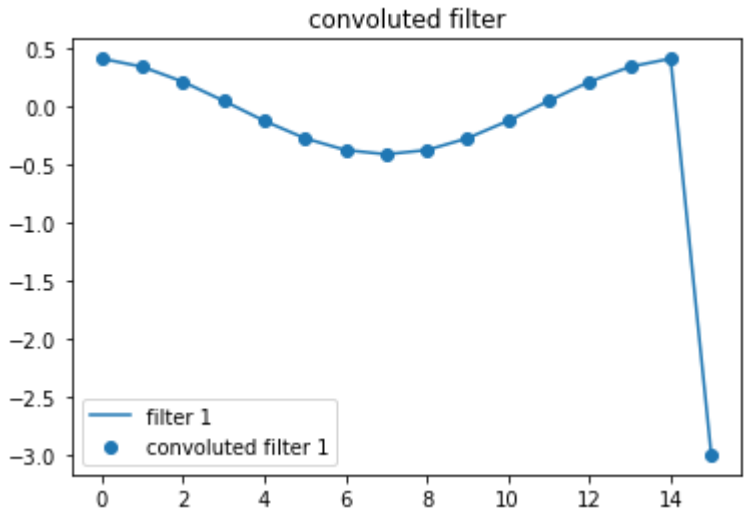
In [33]:

```
#convolution implementation
arr_con = np.array([1,-1])
y1_con = np.convolve(x, arr_con)[1:]
#print(y1_con.shape, x.shape)
plt.scatter(k, y1_con, label = 'convoluted filter 1')
plt.plot(k, y1, label= 'filter 1')
plt.legend()
plt.title('convoluted filter')
```

```
(16,) (16,)
```

Out[33]:

```
Text(0.5, 1.0, 'convoluted filter')
```



Note: we can see that the convolution overlaps with original filter

In [53]:

```
#impplementation of filter in fourier series
h = arr_con
n_o1 = n+h.shape[0] - 1
H1 = rfft(h, n_o1)
X1 = rfft(x, n_o1)
Y1 = H1*X1
y1_four = irfft(Y1, n_o1)
y1_four = y1_four[1:]
print(len(y1_four))
```

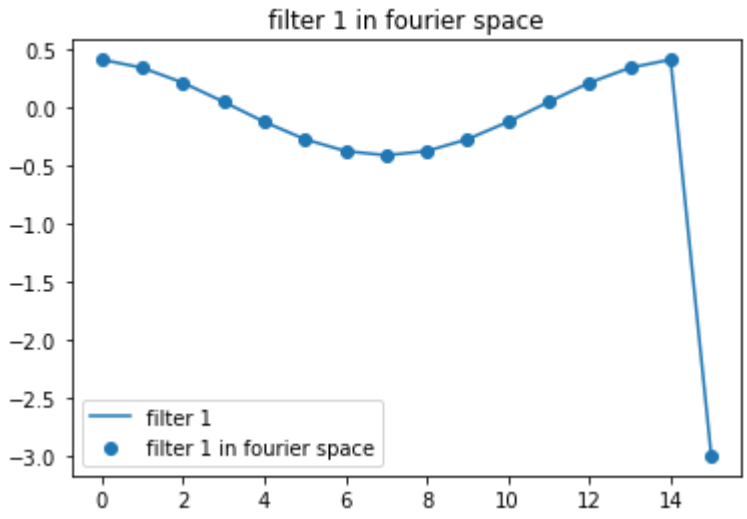
```
16
```

In [54]:

```
plt.scatter(k, y1_four, label = 'filter 1 in fourier space')
plt.plot(k, y1, label= 'filter 1')
plt.legend()
plt.title('filter 1 in fourier space')
```

Out[54]:

```
Text(0.5, 1.0, 'filter 1 in fourier space')
```



Note: filter in fourier series overlaps original filter

Filter 2: $y_k = x_k - (\bar{X})$

In [55]:

```
#no padding required for this filter
y2 = x - np.mean(x)
print("filter 2 =",y2)
```

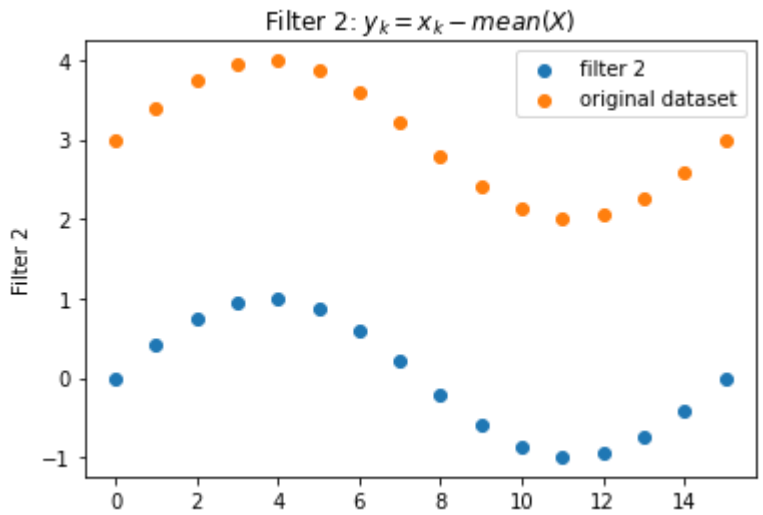
```
filter 2 = [ 0.00000000e+00  4.06736643e-01  7.43144825e-01  9.5105651
6e-01
  9.94521895e-01  8.66025404e-01  5.87785252e-01  2.07911691e-01
 -2.07911691e-01 -5.87785252e-01 -8.66025404e-01 -9.94521895e-01
 -9.51056516e-01 -7.43144825e-01 -4.06736643e-01 -1.33226763e-15]
```

In [58]:

```
plt.scatter(k,y2, label='filter 2')
plt.scatter(k,x, label = 'original dataset')
plt.legend()
plt.ylabel('Filter 2')
plt.title('Filter 2: $y_k = x_k - mean(X)$')
```

Out[58]:

```
Text(0.5, 1.0, 'Filter 2: $y_k = x_k - mean(X)$')
```



Since this filter is not linear and space independant we will not implement convolution and filter in fourier space

Filter 3: $y_k = median(\{x_l, l \in [k-2, k+2]\})$

In [59]:

```
x_padded3 = np.zeros(n+4)
x_padded3[2:-2] = x
y3 = np.empty([n,])
for i in range(n):
    it = i + 2
    y3[i] = np.median(x_padded3[(it-2):(it+2)])
print("filter 3 =", y3)
```

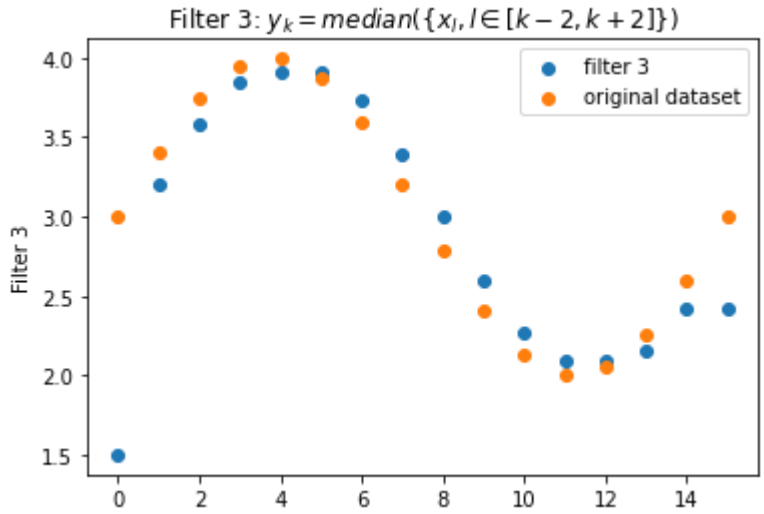
```
filter 3 = [1.5        3.20336832 3.57494073 3.84710067 3.90854096 3.9
0854096
 3.72690533 3.39784847 3.         2.60215153 2.27309467 2.09145904
 2.09145904 2.15289933 2.42505927 2.42505927]
```

In [60]:

```
plt.scatter(k,y3, label='filter 3')
plt.scatter(k,x, label = 'original dataset')
plt.legend()
plt.ylabel('Filter 3')
plt.title('Filter 3: $y_k = median(\{x_l, l\in [k-2, k+2] \})$')
```

Out[60]:

```
Text(0.5, 1.0, 'Filter 3: $y_k = median(\\{x_l, l\\in [k-2, k+2] \\})
$')
```



Since this filter is not linear and space independant we will not implement convolution and filter in fourier space

Filter 4: $y_k = x_{x+0.5} - x_{x-0.5}$

In [64]:

```
x_padded4 = np.zeros(n+2)
x_padded4[1:-1] = x
y4 = np.empty([n,])
for i in range(n):
    it = i+1
    y4[i] = 0.5*(x_padded4[it+1] - x_padded4[it-1]) #formula after simplification
print("filter 4 =",y4)
```

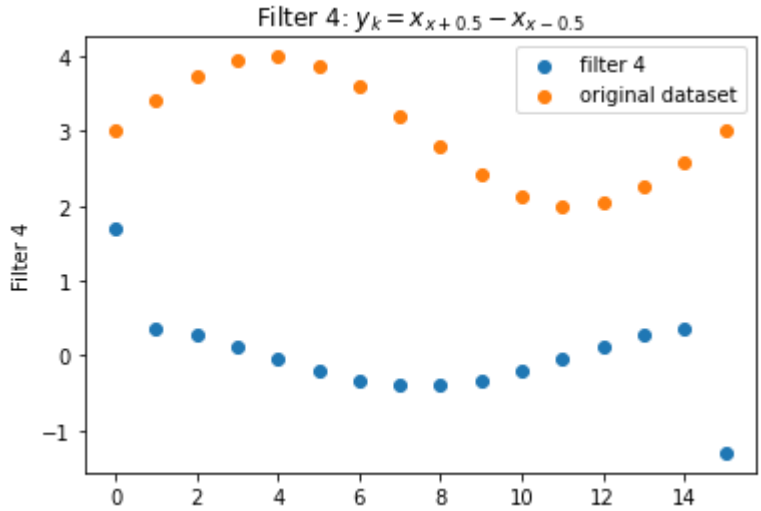
```
filter 4 = [ 1.70336832  0.37157241  0.27215994  0.12568853 -0.0425155
6 -0.20336832
 -0.32905686 -0.39784847 -0.39784847 -0.32905686 -0.20336832 -0.042515
56
  0.12568853  0.27215994  0.37157241 -1.29663168]
```

In [81]:

```
plt.scatter(k,y4, label='filter 4')
plt.scatter(k,x, label = 'original dataset')
plt.legend()
plt.ylabel('Filter 4')
plt.title('Filter 4: $y_k = x_{x+0.5} - x_{x-0.5}$')
```

Out[81]:

```
Text(0.5, 1.0, 'Filter 4: $y_k = x_{x+0.5} - x_{x-0.5}$')
```



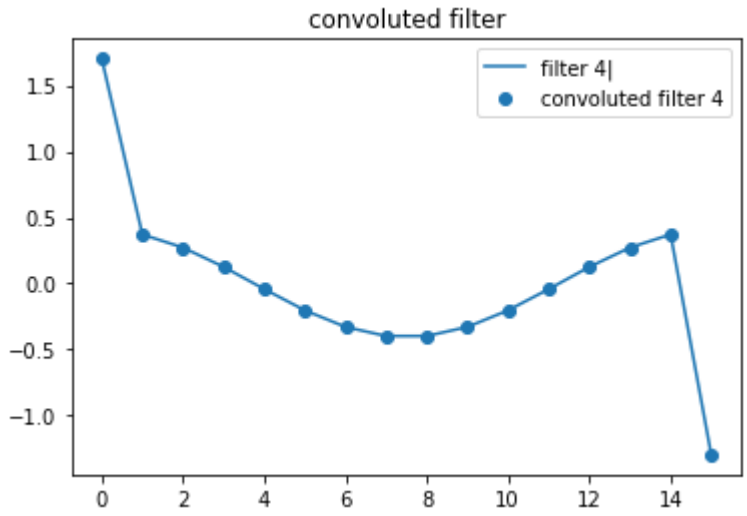
Since the given dataset is linear and space invariant, we will implement convolution and filter in fourier series

In [70]:

```
#convolution implementation
arr4_con = np.array([0.5, 0, -0.5])
y4_con = np.convolve(x, arr4_con)[1:-1]
#print(y1_con.shape, x.shape)
plt.scatter(k, y4_con, label = 'convoluted filter 4')
plt.plot(k, y4, label= 'filter 4|')
plt.legend()
plt.title('convoluted filter')
```

Out[70]:

```
Text(0.5, 1.0, 'convoluted filter')
```



Note: we can see the convolution function overlaps with the filter

In [92]:

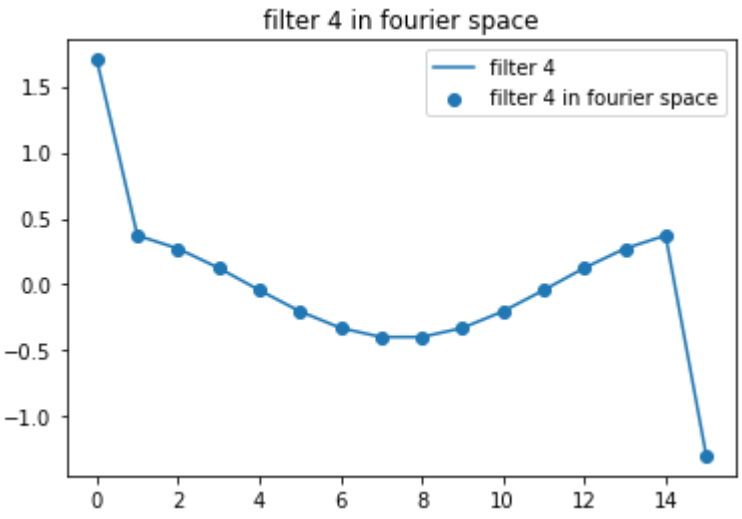
```
#filter in fourier space
h4 = arr4_con
n_o4 = n+h4.shape[0] - 1
H4 = rfft(h4, n_o4)
X4 = rfft(x, n_o4)
Y4 = H4*X4
y4_four = irfft(Y4, n_o4)
y4_four = y4_four[1:-1]
#print(len(H4))
```

In [78]:

```
plt.scatter(k, y4_four, label = 'filter 4 in fourier space')
plt.plot(k, y4, label= 'filter 4')
plt.legend()
plt.title('filter 4 in fourier space')
```

Out[78]:

```
Text(0.5, 1.0, 'filter 4 in fourier space')
```



Note filter in fourier space overlaps with filter

Filter 5: $y_k = |x_{x+0.5} - x_{x-0.5}|$

In [79]:

```
x_padded5 = np.zeros(n+2)
x_padded5[1:-1] = x
y5 = np.empty([n,])
for i in range(n):
    it = i+1
    y5[i] = np.abs(0.5*(x_padded5[it+1] - x_padded5[it-1])) #formula after simplific
print("filter 5 =",y5)
```

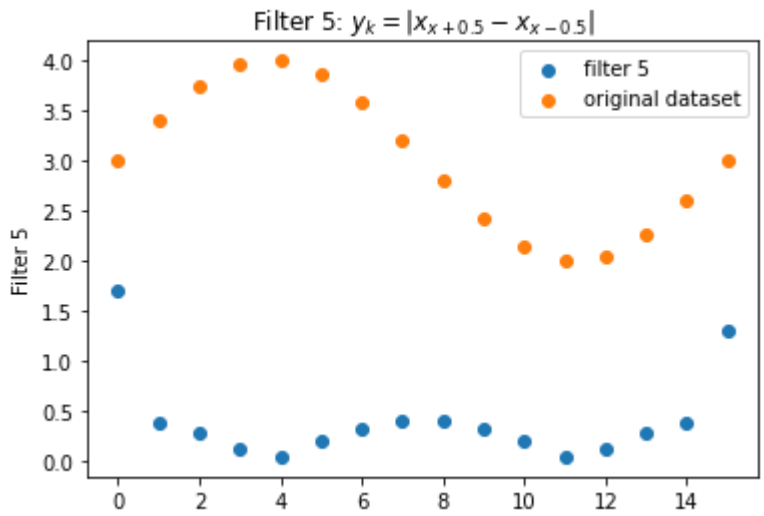
```
filter 5 = [1.70336832 0.37157241 0.27215994 0.12568853 0.04251556 0.2
0336832
 0.32905686 0.39784847 0.39784847 0.32905686 0.20336832 0.04251556
 0.12568853 0.27215994 0.37157241 1.29663168]
```

In [80]:

```
plt.scatter(k,y5, label='filter 5')
plt.scatter(k,x, label = 'original dataset')
plt.legend()
plt.ylabel('Filter 5')
plt.title('Filter 5: $y_k = |x_{x+0.5} - x_{x-0.5}|$')
```

Out[80]:

```
Text(0.5, 1.0, 'Filter 5: $y_k = |x_{x+0.5} - x_{x-0.5}|$')
```



Since this filter is not linear and space independant we will not implement convolution and filter in fourier space

Filter 6: $y_k = \frac{1}{5}\sum_{i=k-2}^{k+2} x_i$

In [89]:

```
x_padded6 = np.zeros(n+4)
x_padded6[2:-2] = x
y6 = np.empty([n,])
for i in range(n):
    it = i + 2
    y6[i] = np.mean(x_padded6[(it-2):(it+3)])
print('filter 6 =', y6)
```

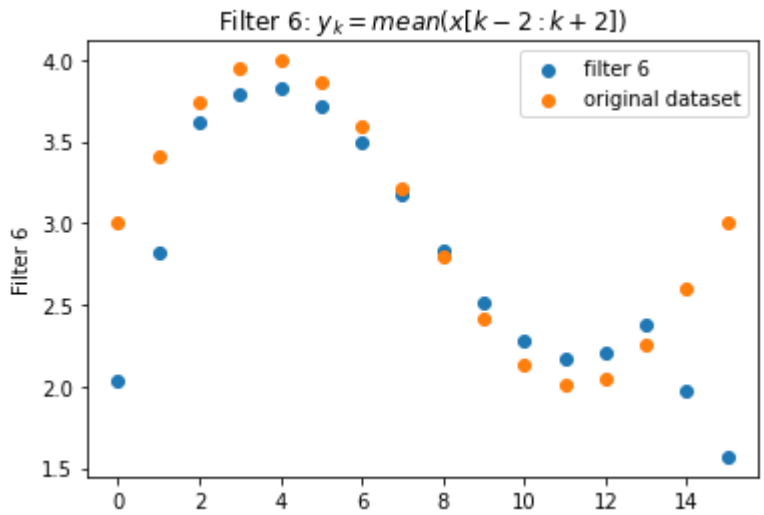
```
filter 6 = [2.02997629 2.8201876  3.61909198 3.79229706 3.82850678 3.7
2146015
 3.48966651 3.17320508 2.82679492 2.51033349 2.27853985 2.17149322
 2.20770294 2.38090802 1.9798124  1.57002371]
```

In [90]:

```
plt.scatter(k,y6, label='filter 6')
plt.scatter(k,x, label = 'original dataset')
plt.legend()
plt.ylabel('Filter 6')
plt.title('Filter 6: $y_k = mean(x[k-2:k+2])$')
```

Out[90]:

```
Text(0.5, 1.0, 'Filter 6: $y_k = mean(x[k-2:k+2])$')
```



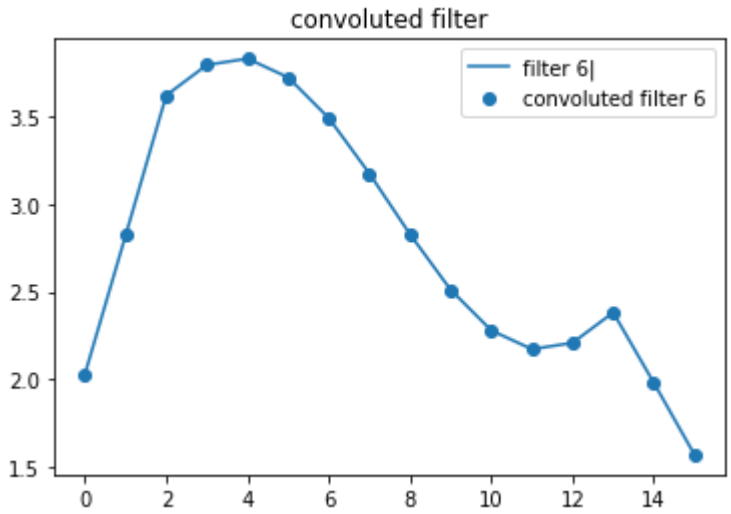
Since the given dataset is linear and space invariant, we will implement convolution and filter in fourier series

In [91]:

```
#convolution
arr6_con = np.ones(5)/5
y6_con = np.convolve(x, arr6_con)[2:-2]
#print(y1_con.shape, x.shape)
plt.scatter(k, y6_con, label = 'convoluted filter 6')
plt.plot(k, y6, label= 'filter 6|')
plt.legend()
plt.title('convoluted filter')
```

Out[91]:

```
Text(0.5, 1.0, 'convoluted filter')
```



Note that the convoluted filter overlapes with the original filter

In [95]:

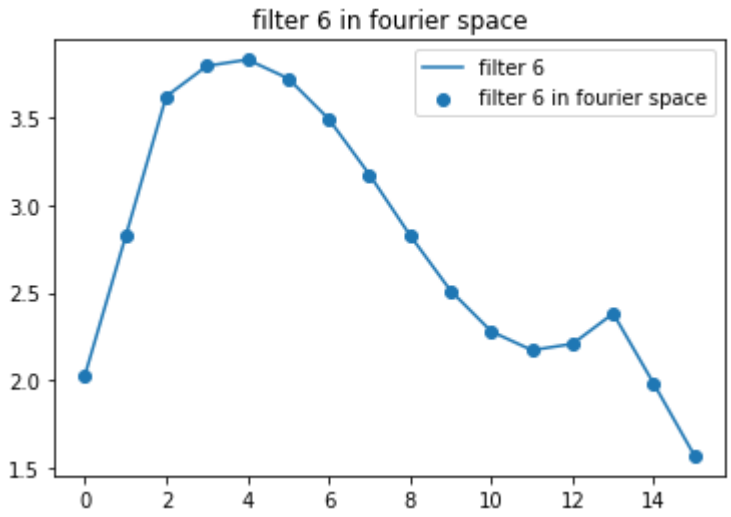
```
#filter in fourier space
h = arr6_con
n_o6 = n+h.shape[0] - 1
H6 = rfft(h, n_o6)
X6 = rfft(x, n_o6)
Y6 = H6*X6
y6_four = irfft(Y6, n_o6)
y6_four = y6_four[2:-2]
```

In [96]:

```
plt.scatter(k, y6_four, label = 'filter 6 in fourier space')
plt.plot(k, y6, label= 'filter 6')
plt.legend()
plt.title('filter 6 in fourier space')
```

Out[96]:

```
Text(0.5, 1.0, 'filter 6 in fourier space')
```



Note the filter in fourier space overlaps with original filter

Hybrid Images

In [98]:

```
def my_filter(img, fernel):
    im_size = img.shape
    k_size = kernel.shape

    assert k_size[0]%2 == 1

    pad_size = img_size + k_size[0]-1 #simplified dimensions

    return psd_size
```

In [ ]: