fractional-shift-2

May 24, 2023

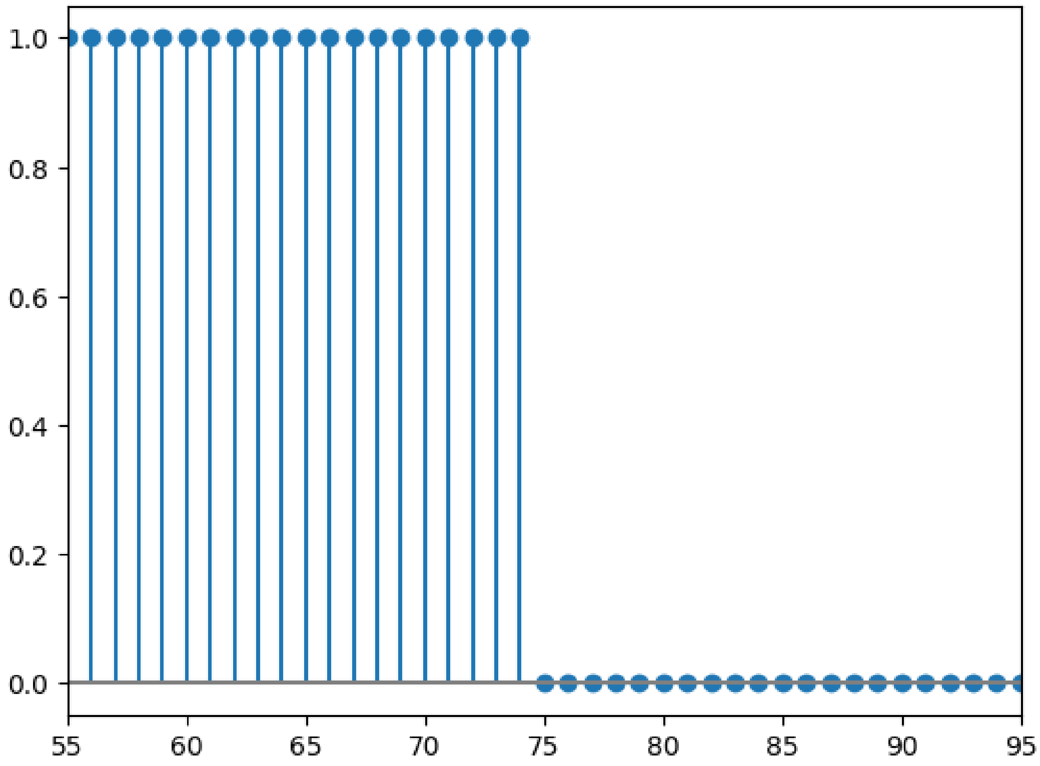
1 Fourier shift with Lanczos smoothing filter

```
[ ]: import numpy as np
     from matplotlib import pyplot as plt
```

```
[ ]: n = 150
     x0 = n * 0.5
     x = np.arange(n)
```

```
[ ]: z = x < x0
     plt.xlim(x0 - 20, x0 + 20)
     plt.stem(x,z, basefmt="gray")
```

```
[ ]: <StemContainer object of 3 artists>
```



[ ]:

```
def wavenum(i) : return (i + n // 2) % n - n // 2
```

[ ]:

```
# fractional shift
delta = 0.5
H = np.exp(-2j * wavenum(x) / n * np.pi * delta)
if n & 1:
    idx0, idx1 = n//2, 1 + n//2
    #ambiguous = np.exp(-2j * (n - 1) / n * np.pi * delta)
    print("wavenums", wavenum(idx0), wavenum(idx1))
    print("H:", H[idx0], H[idx1])
else :
    idx0 = n//2
    ambiguous = np.exp(-2j * (n//2) / n * np.pi * delta)
    print("H:", H[idx0], ambiguous)
    H[idx0] = 0.5 * (ambiguous + 1/ambiguous)
    print("new H:", H[idx0])
```

```
H: (6.123233995736766e-17+1j) (6.123233995736766e-17-1j)
new H: (6.123233995736766e-17+0j)
```

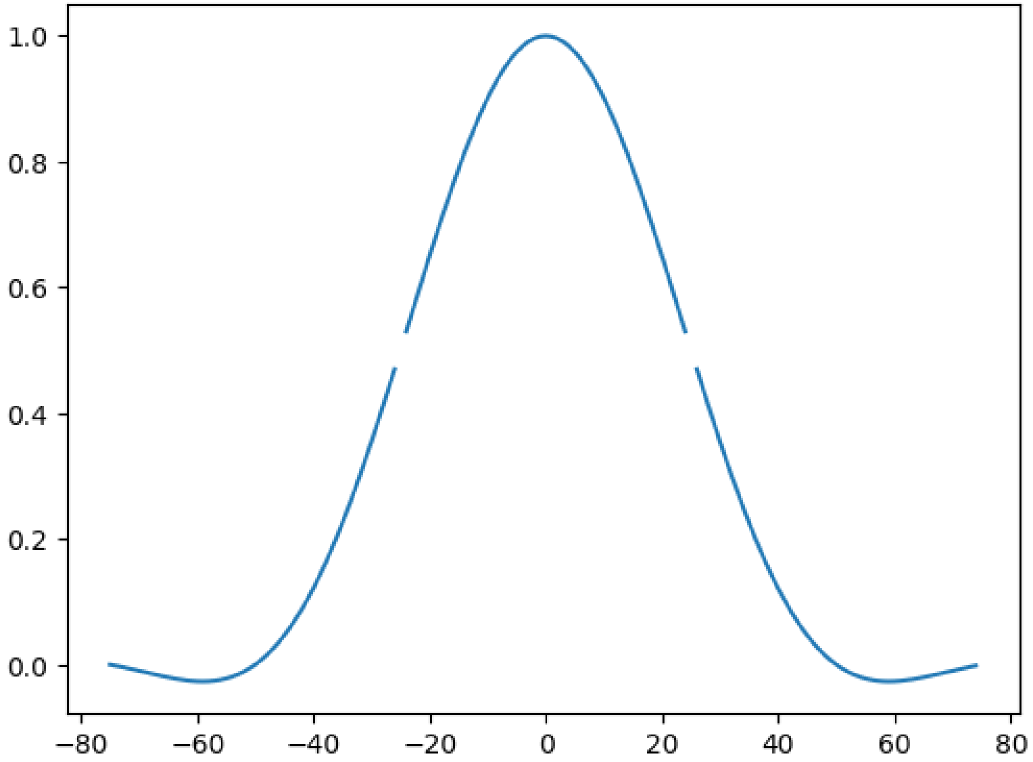
[ ]:

```
#lanczos smoothing
G = np.zeros(n)
M = 0
#we pick the support of the smoothing window depending on the fractional shift
mydistance = np.abs(np.mod(delta, 1) - 0.5)
if mydistance < 0.5 - 0.03125:
    M = 1
if mydistance < 0.5 - 0.0125:
    M = 2
if mydistance < 0.5 - 0.25:
    M = 3
print("M is ", M)
if M:
    for i in range(n):
        k = wavenum(i)
        a = 2 * np.pi / n * k * M
        #weighted average of (1/2 + 1/2 cos(pi / M * x))
        if True:
            if 4 * k == -n or 4 * k == n:
                G[i] += 1
            else:
                G[i] += 2 * a * np.sin(a) / (np.pi**2 - a**2)
        if True:
            if i:
                G[i] += 2 * np.sin(a) / (a)
            else:
                G[i] += 2
        G[i] *= 0.5
        if k == -n//2 :
            print("fs: ", G[i], np.sin(-a) / -a)
else :
    G = np.ones(n)
#plot frequency response
from numpy import fft as fft
plt.plot(np.arange(n) - n//2, fft.fftshift(G))
```

```
M is  3
fs:  -2.8432379097786405e-17 2.274590327822911e-16
```

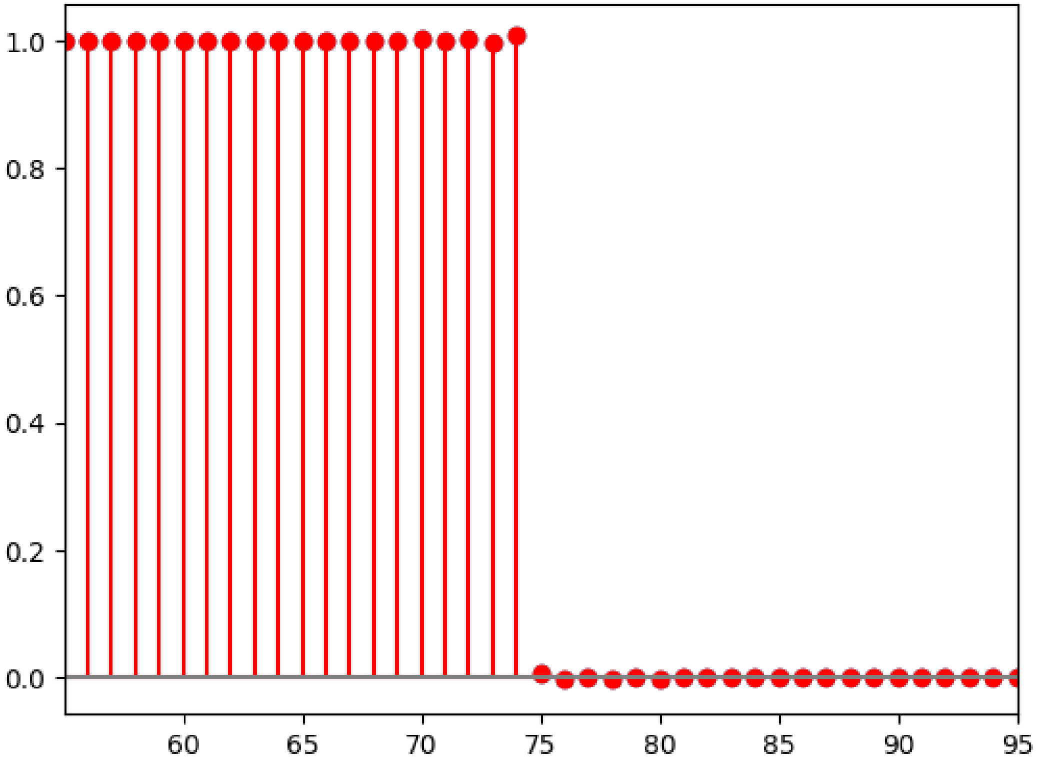
```
/tmp/ipykernel_26400/495480382.py:22: RuntimeWarning: divide by zero encountered
in scalar divide
  G[i] += 2 * a * np.sin(a) / (np.pi**2 - a**2)
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7fb8c19ae170>]
```



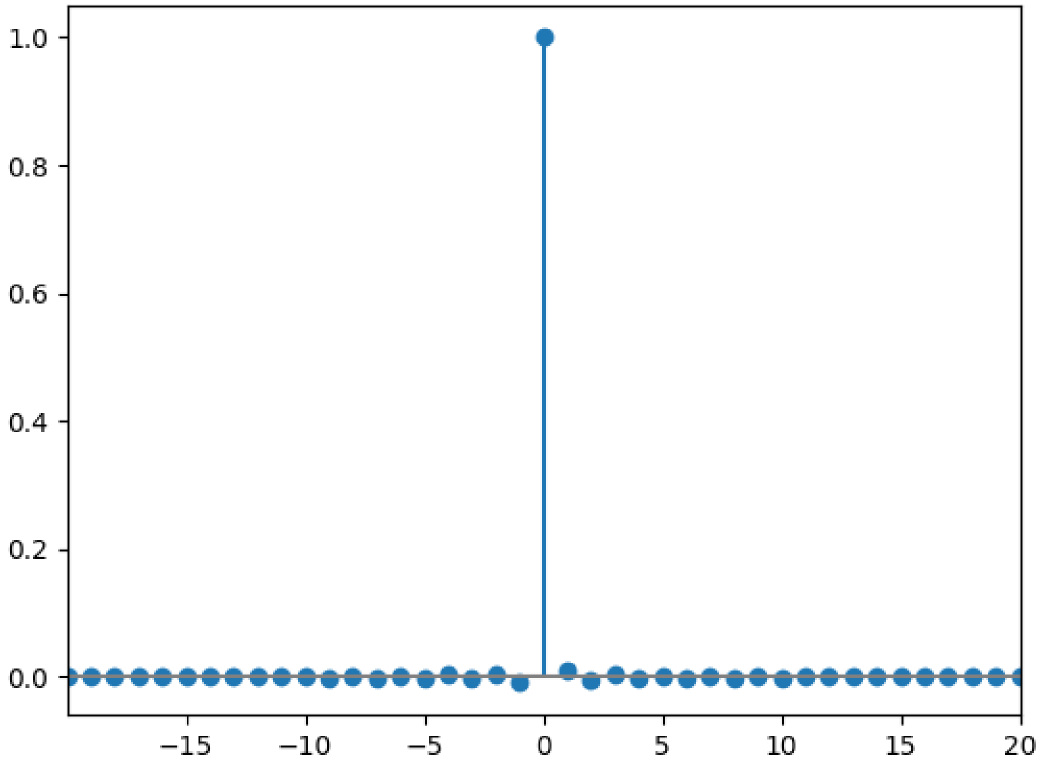
[ ]:
```
z_shifted = fft.ifft(H * fft.fft(z))
z_smoother = fft.ifft(G * H * fft.fft(z))
plt.xlim([-20 + delta + x0, 20 + delta + x0])
plt.stem(x, np.real(z_shifted))
plt.stem(x, np.real(z_smoother), linefmt='r-', markerfmt='ro', basefmt="gray")
```

```
[ ]: <StemContainer object of 3 artists>
```



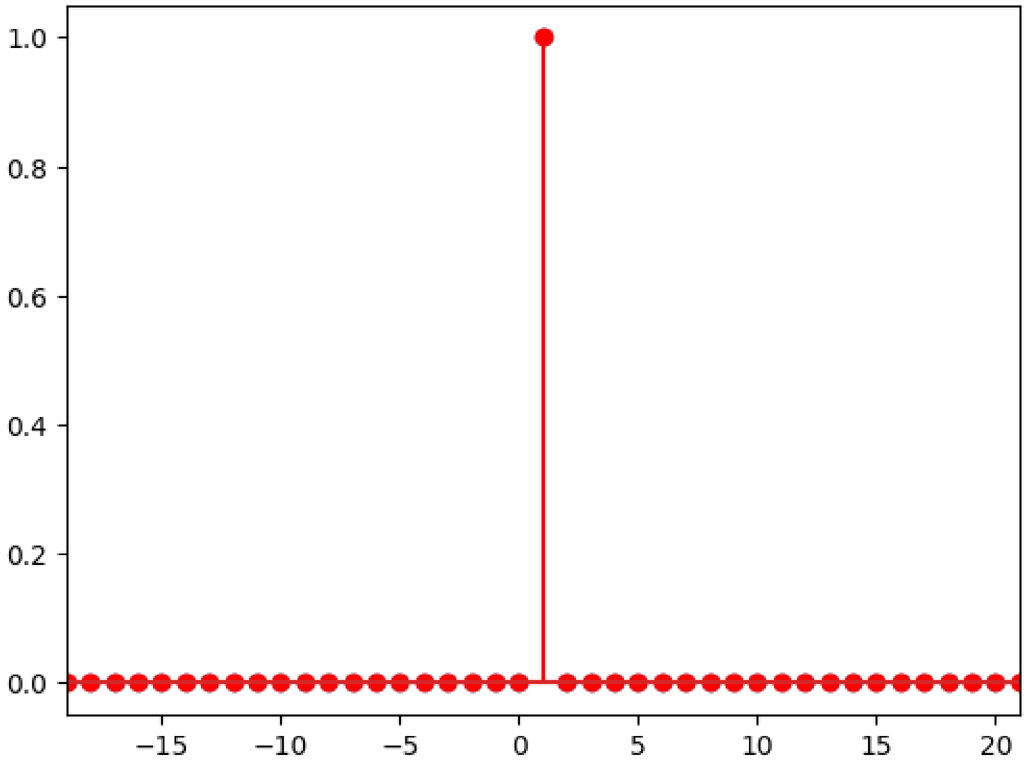
```
[ ]: #show combined impulse response with shift and smoothing
     impulse_response = fft.ifft(H * 1)
     impulse_response_smooth = fft.ifft(H * G)
     plt.xlim([-20 + delta,20 + delta])
     plt.stem(np.arange(n) - n//2, fft.fftshift(np.real(impulse_response)),␣
      ↪basefmt="gray")
```

```
[ ]: <StemContainer object of 3 artists>
```



[ ]:
```
plt.xlim([-20 + delta,20 + delta])
plt.stem(np.arange(n) - n//2, fft.fftshift(np.real(impulse_response)))
plt.stem(np.arange(n) - n//2, fft.fftshift(np.real(impulse_response_smooth)),␣
 ↪linefmt='r-', markerfmt='ro')
```

[ ]:
```
<StemContainer object of 3 artists>
```


1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10
15
20

[ ]: