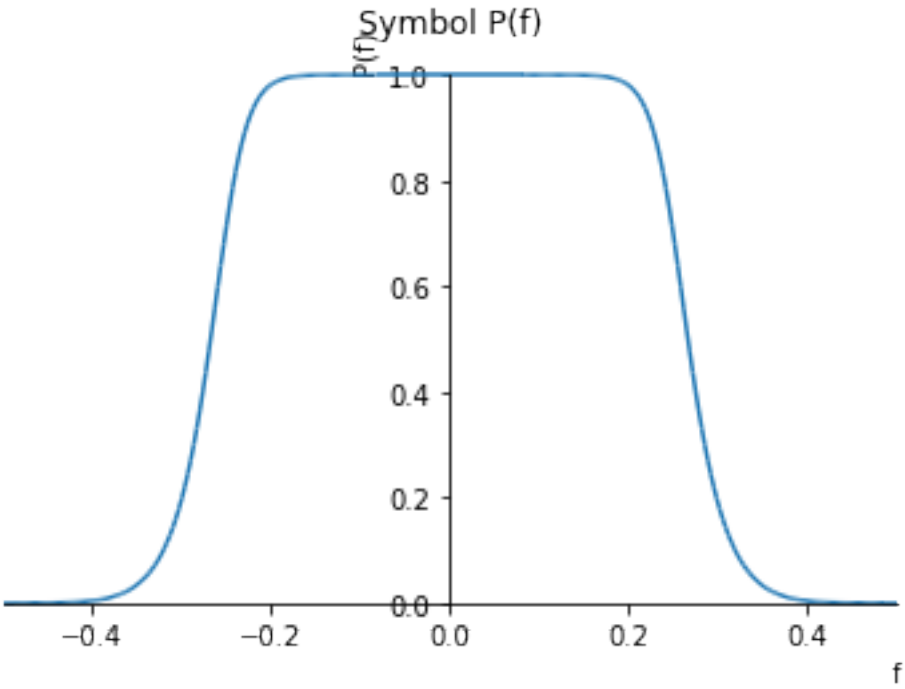
final question 1&2

March 25, 2018

In [1]:
```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
import sys
sys.setrecursionlimit(25000)
import mpmath as mp
import scipy.fftpack
from sympy import lambdify
import mpmath as mpf
```

In [2]:
```
# analytic expression for the symbol P(f) of B-L wavelet filter
f=sp.symbols('f')
J=sp.I
w = 2*sp.pi*f
N1 = 5+30*sp.cos(w/2)**2+30*sp.sin(w/2)**2*sp.cos(w/2)**2;
N2 = 2*sp.sin(w/2)**4*sp.cos(w/2)**2+70*sp.cos(w/2)**4+(2/3)*sp.sin(w/2)**6;
S = (N1+N2)/(105*sp.sin(w/2)**8);
phi_hat=16/(w**4*sp.sqrt(S))
Pf=phi_hat.subs(f,2*f)/phi_hat
```

In [3]:
```
# plot the |P(f)| of B-L wavelet filter
sp.plotting.plot(abs(Pf),(f, -0.5, 0.5), xlabel='f', ylabel='P(f)',
                 adaptive=False, num_of_points=200, title = "Symbol P(f)")
```



Out[3]: <sympy.plotting.plot.Plot at 0x1116f6c50>

In [4]:
```
def f_coeff(x,F,N):
    # x: symbolic periodic function of "f"
    # F: period
    # N: number of Fourier coeffs (odd integer)
    # h: Fourier coefficients h[-3],h[-2],h[-1],h[0],h[1],h[2],h[3]
    df = F/N
    xs=np.zeros(N,dtype=complex)
    xs[0]=sp.limit(x,f,0)
    for n in range(1,N):
        xs[n] = complex(x.subs(f,n*df))
    h = np.fft.fft(xs)/N
    h = np.fft.fftshift(h)
    return h
```

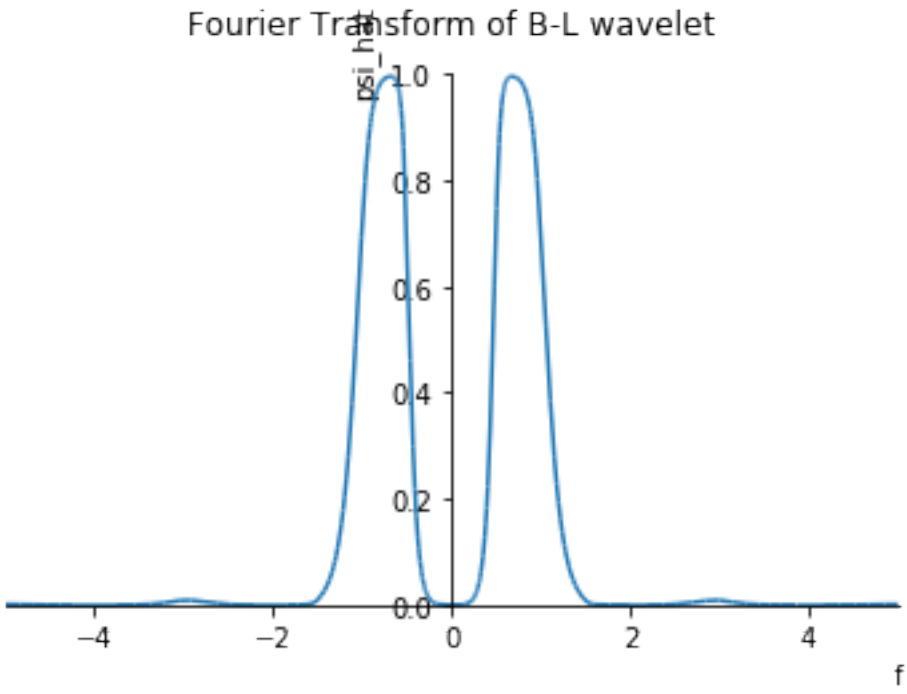
In [5]:
```
def f_trans(x,F,M,N):
    # x: symbolic input function of "f"
    # F: frequency window [-F/2,F/2]
    # N: number of sample values
    # M: number of aliases
    # Xs: Fourier transform sample values
    # fs: frequency sample values

    dt = 1/F  # delta t
```

```
    df = F/N  # delta f
    T =N/F

    for k in range(1,M+1):
        x = x+x.subs(f,f-k*T)+x.subs(f,f+k*T)
    xs=np.zeros(N,dtype=float)
    fs=np.zeros(N,dtype=float)
    dc=sp.limit(x,f,0)
    xs[0]=dc
    fs[0]=-F/2

    for n in range(1,N):
        xs[n] = float(x.subs(f,n*dt))
        fs[n] = (n-1)*df-F/2
    Xs = np.fft.fft(xs)*dt
    Xs = np.fft.fftshift(Xs)
    return fs, Xs
```

In [6]:
```
f=sp.symbols('f')
J=sp.I

w = 2*sp.pi*f
N1w = 5+30*((sp.cos(w/2))**2)+30*((sp.sin(w/2))**2)*((sp.cos(w/2))**2)
N2w = 2*((sp.sin(w/2))**4)*((sp.cos(w/2))**2)+70*((sp.cos(w/2))**4)+(2/3)*((sp.sin(w/2)
Sw = (N1w+N2w)/(105*((sp.sin(w/2))**8))
phi_hat=16/((w**4)*sp.sqrt(Sw))

x = phi_hat
F = 6
M=0
N=256
fs, Xs= f_trans(x,F,M,N)

# plt.figure()
# plt.title('B-L scaling function')
# plt.plot(fs,np.real(Xs),'bo')
# plt.savefig('B-L scaling')
```

In [7]:
```
# analytic expression for the Fourier Transform of B-L wavelet
psi_hat=-sp.exp(-J*sp.pi*f)*Pf.subs(f,(f+1)/2)*phi_hat.subs(f,f/2)
```

In [8]:
```
#plot Fourier Tranform of B-L wavelet
sp.plotting.plot(abs(psi_hat), (f,-5,5), xlabel='f', ylabel='psi_hat',
                 num_of_points=200, title = "Fourier Transform of B-L wavelet")
```



Out[8]: <sympy.plotting.plot.Plot at 0x111200588>