Lecture19

March 22, 2018

1 AMAT503: Lecture 19

March 22, 2018.

Michael Lamoureux

We are doing multiresolution analysis, applied to real functions on the line.

```
In [135]: ## Some startup commands

          %matplotlib inline
          from numpy import *
          from scipy import *
          from matplotlib.pyplot import *
          from pywt import *
```

1.1 1. Multiresolution Analysis - applied

We want to do a multiresolution analysis on some real functions. You might need to look at the last class's lecture notes to see the details.

We start with a function $f(t)$ and a scaling function $\phi(t)$ in the space V_0 and write down some coefficients

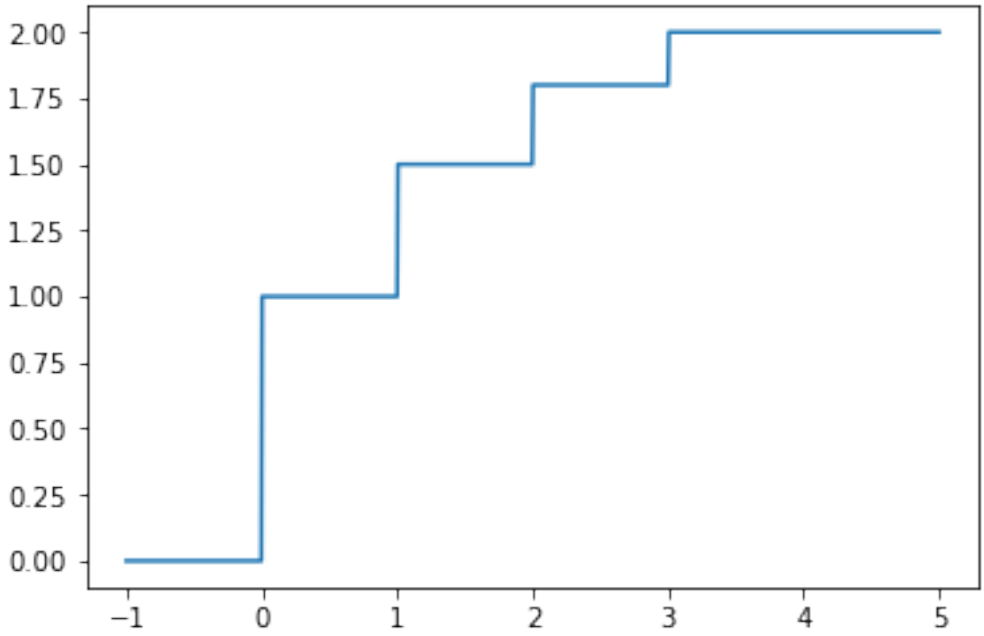
$$y_k = \int f(t)\overline{\phi(t-k)}\,dt.$$

We then apply the corresponding discrete wavelet transform to get coefficients for the multiresolution analysis. This way, function f is expressed as a linear combination of scaled versions of the wavelet functions (and a scaled version of the scaling function.)

We start by defining a function:

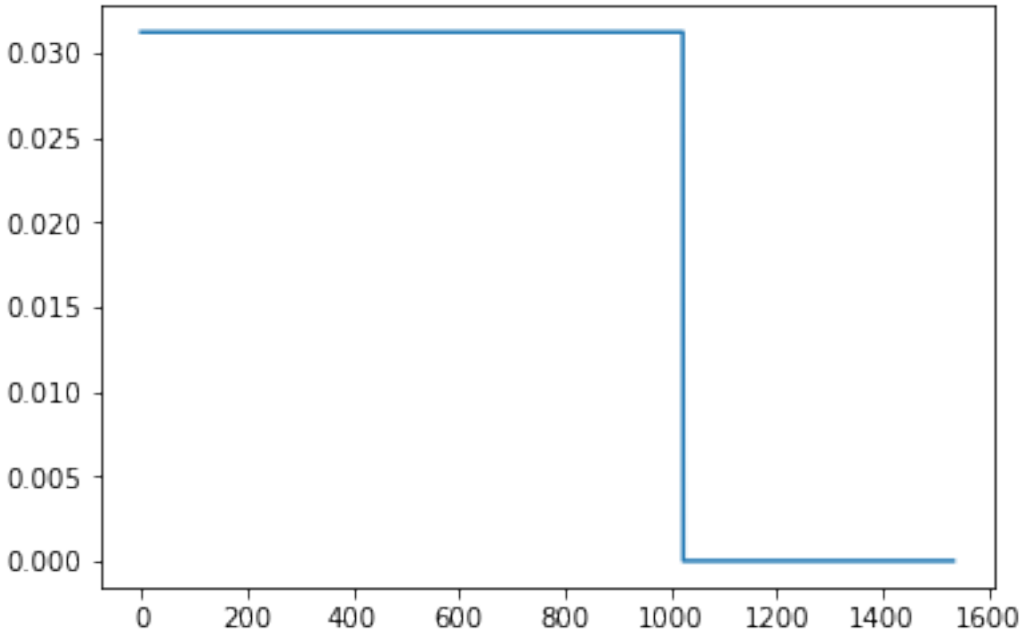
```
In [42]: def f(t):
             return (t>0)+ .5*(t>1) + .3*(t>2) + .2*(t>3)
```

```
In [137]: t = linspace(-1,5,1000)
          plot(t,f(t));
```

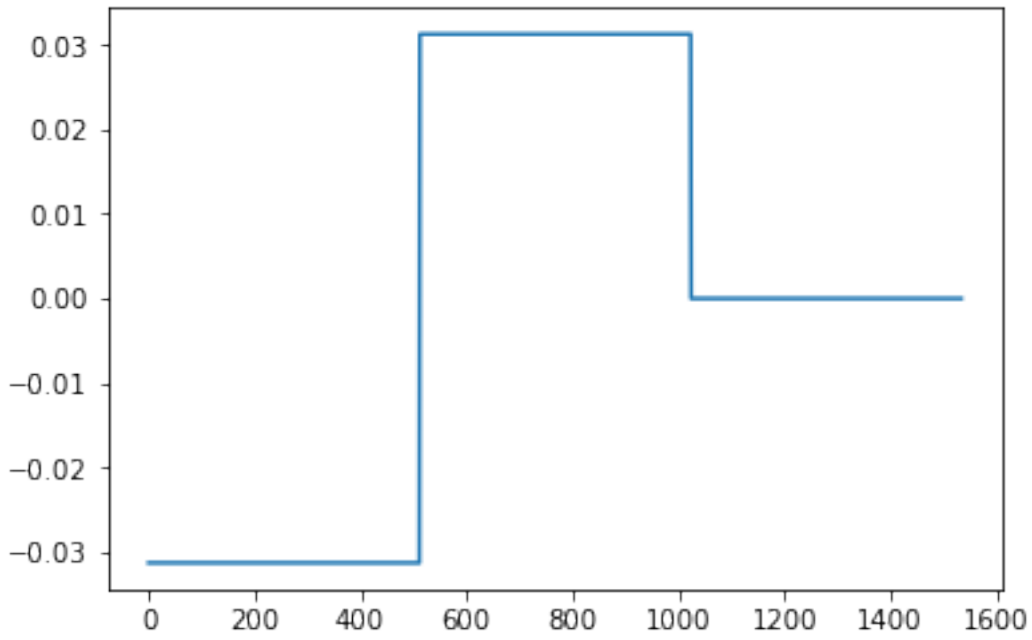


Let's get our scaling function, and wavelet function.

```
In [116]: w = Wavelet('db1')
          h = w.dec_lo
          z = [1]
          for k in range(10):
              x = convolve(z,h)
              z = zeros(2*size(x))
              z[0:size(z):2] = x

          phi = x
          plot(phi);
```



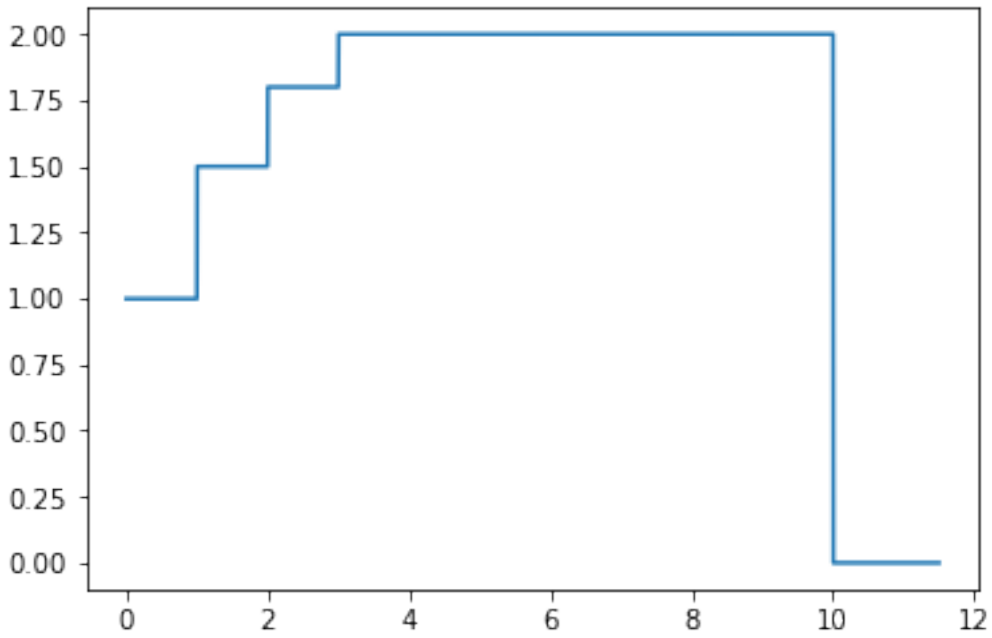

```
In [104]: size(phi)

Out[104]: 1535

In [105]: # Check normalization, sum of squares should be one.
          sum(phi**2)

Out[105]: 1.0000000000000002

In [124]: w = Wavelet('db1')
          h = w.dec_lo
          g = w.dec_hi
          z = [1]
          x = convolve(z,g)
          z = zeros(2*size(x))
          z[0:size(z):2] = x

          for k in range(9):
              x = convolve(z,h)
              z = zeros(2*size(x))
              z[0:size(z):2] = x

          psi = x
          plot(psi);
```



In [106]:
```
# Compute the initial scaling coefficient.

N = 10
y = zeros(N,float_)
t = linspace(0,1,size(phi))
dt = 1/size(phi)
for k in range(N):
    y[k] = sum(f(t+k)*phi)

y
```

Out[106]:
```
array([ 31.96875 ,  47.984375,  57.590625,  63.99375 ,  64.      ,
        64.      ,  64.      ,  64.      ,  64.      ,  64.      ])
```

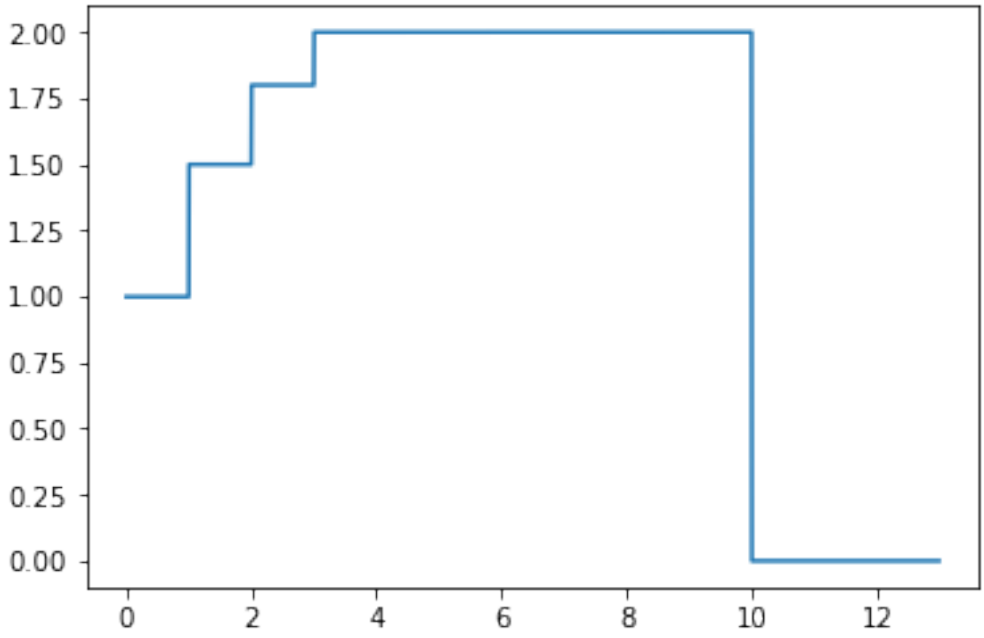
In [114]:
```
# Let's build a reconstruction of f from these coefficients

f_len = 1024*size(y) + size(phi)
f_recon = zeros(f_len, float_)

for k in range(size(y)):
    f_recon[(1024*k):(1024*k+size(phi))] = f_recon[(1024*k):(1024*k+size(phi))] + y[

plot(linspace(0,size(f_recon)/1024,size(f_recon)),f_recon);
```




```
In [125]: ## Let's compute some wavelet coefficients

          (cA, cD) = dwt(y, 'db1')
          cA, cD

Out[125]: (array([ 56.53539686,  85.97313605,  90.50966799,  90.50966799,  90.50966799]),
           array([-11.32475704,  -4.52769311,   0.        ,   0.        ,   0.        ]))
```

We should be able to reconstruct f using these coefficients.

```
In [134]: fw_len = 1024*size(cA) + size(phi)
          fw_recon = zeros(fw_len, float_)

          for k in range(size(cA)):
              fw_recon[(1024*k):(1024*k+size(phi))] = fw_recon[(1024*k):(1024*k+size(phi))] + \
                  cA[k]*phi - cD[k]*psi

          plot(linspace(0,size(fw_recon)/512,size(fw_recon)),fw_recon/sqrt(2));
```



Notice I had to put a sqrt(2) in the plot command to get this to work out correctly. You might want to think about that. (So do I.)

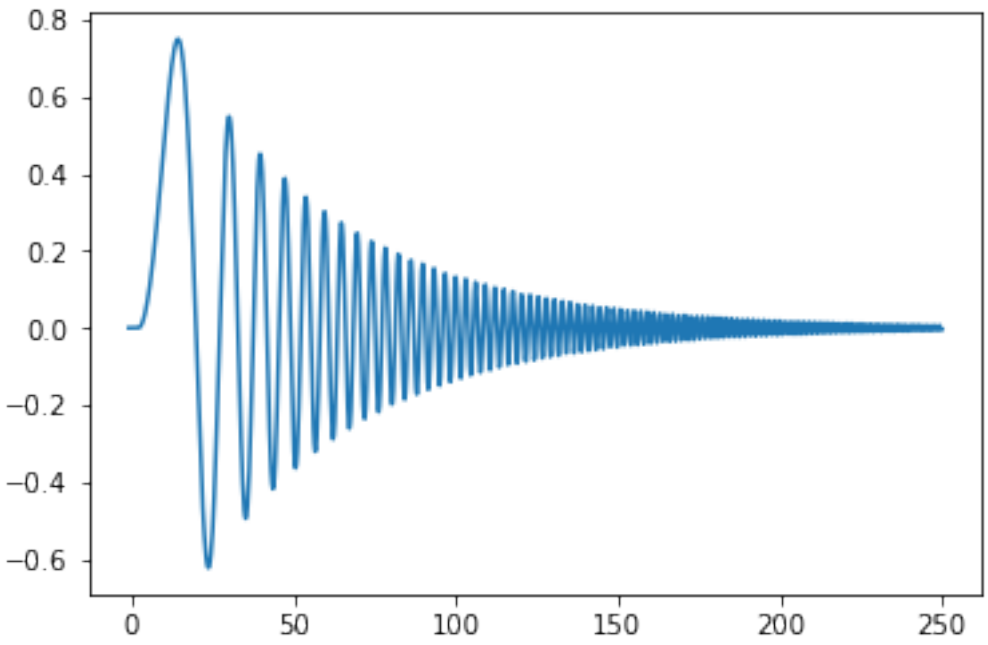
Now, of course we could repeat this discrete wavelet transform several times. The point is, we never have to computer inner products anymore to get the wavelet coefficients for the function. They just come out of the discrete wavelet algorith. This is a very powerful idea.

1.1.1 More complicated functions, wavelets

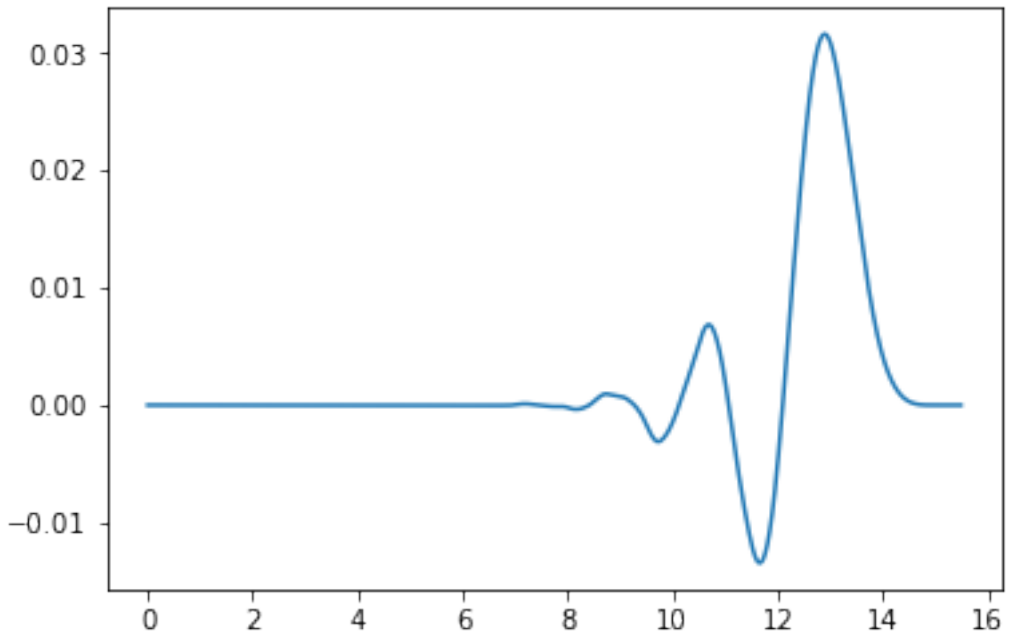
Let's see what happens if we use a more complicated function. Something with lots of wiggles and changes. I suggest we use a sinusoid that decays quickly. Maybe we can let the frequency change a lot.

We should then try to approximate using a wiggly Daubechies wavelets.

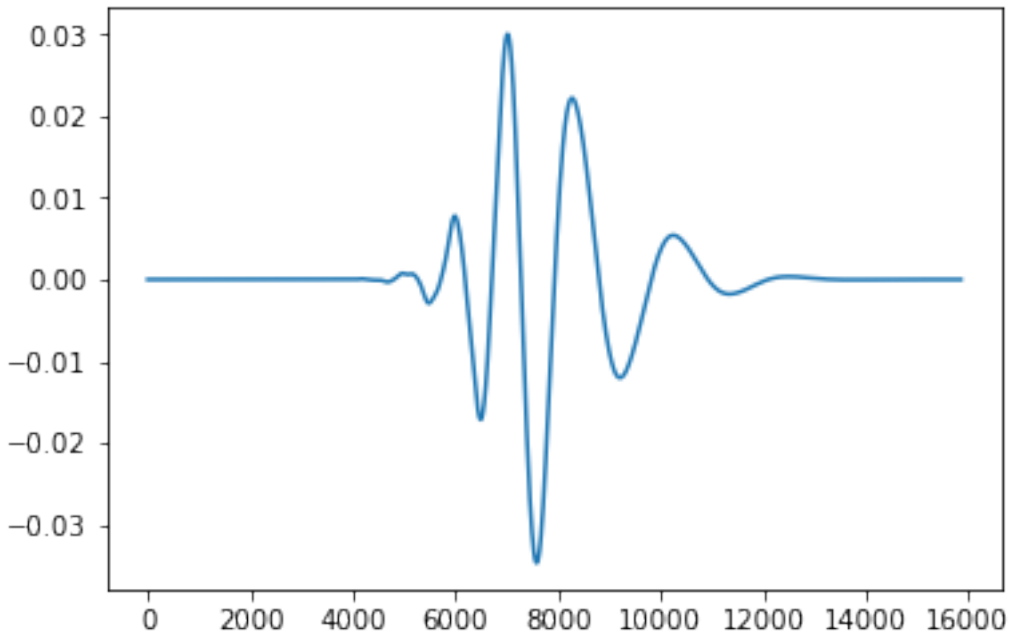
```
In [217]: def f(t):
              return (t>2)*exp(-t/50)*sin(.01*(t-2)**2)

          t=linspace(-1,250,1000)
          plot(t,f(t));
```




```
In [232]: w = Wavelet('db8')
          h = w.dec_lo
          z = [1]
          for k in range(10):
              x = convolve(z,h)
              z = zeros(2*size(x))
              z[0:size(z):2] = x

          phi = x
          plot(linspace(0,size(phi)/1024,size(phi)),phi);
```

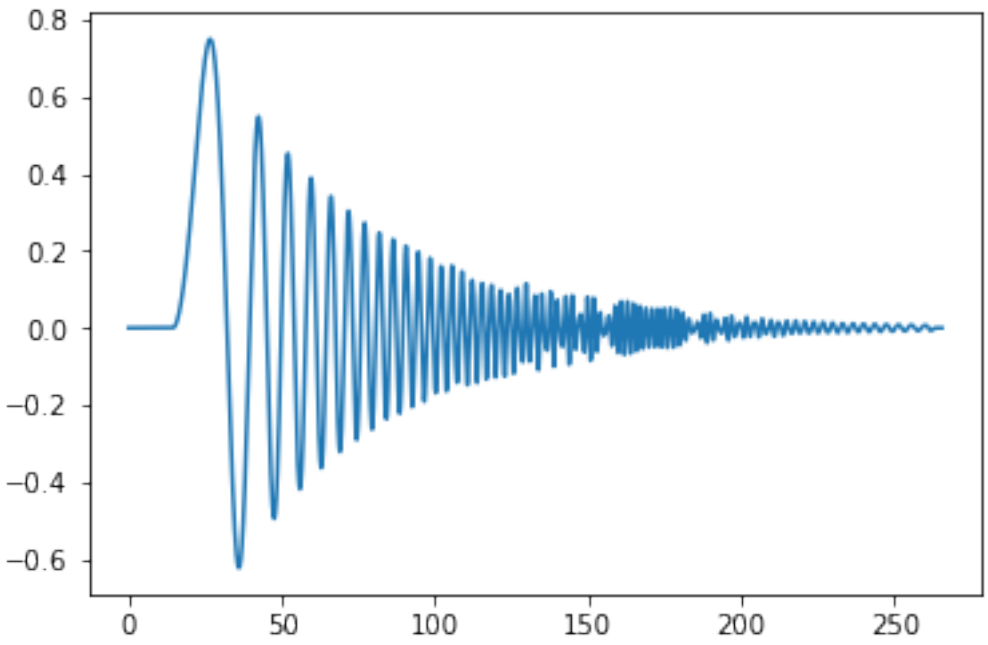



```
In [233]: w = Wavelet('db8')
          h = w.dec_lo
          g = w.dec_hi
          z = [1]
          x = convolve(z,g)
          z = zeros(2*size(x))
          z[0:size(z):2] = x

          for k in range(9):
              x = convolve(z,h)
              z = zeros(2*size(x))
              z[0:size(z):2] = x

          psi = x
          plot(psi);
```



In [234]:
```
# Compute the initial scaling coefficient.

N = 250
y = zeros(N,float_)
t = linspace(0,1,size(phi))
dt = 1/size(phi)
for k in range(N):
    y[k] = sum(f(t+k)*phi)
```

In [235]:
```
# Let's build a reconstruction of f from these coefficients

f_len = 1024*size(y) + size(phi)
f_recon = zeros(f_len, float_)

for k in range(size(y)):
    f_recon[(1024*k):(1024*k+size(phi))] = f_recon[(1024*k):(1024*k+size(phi))] + y[

plot(linspace(0,size(f_recon)/1024,size(f_recon)),f_recon);
```



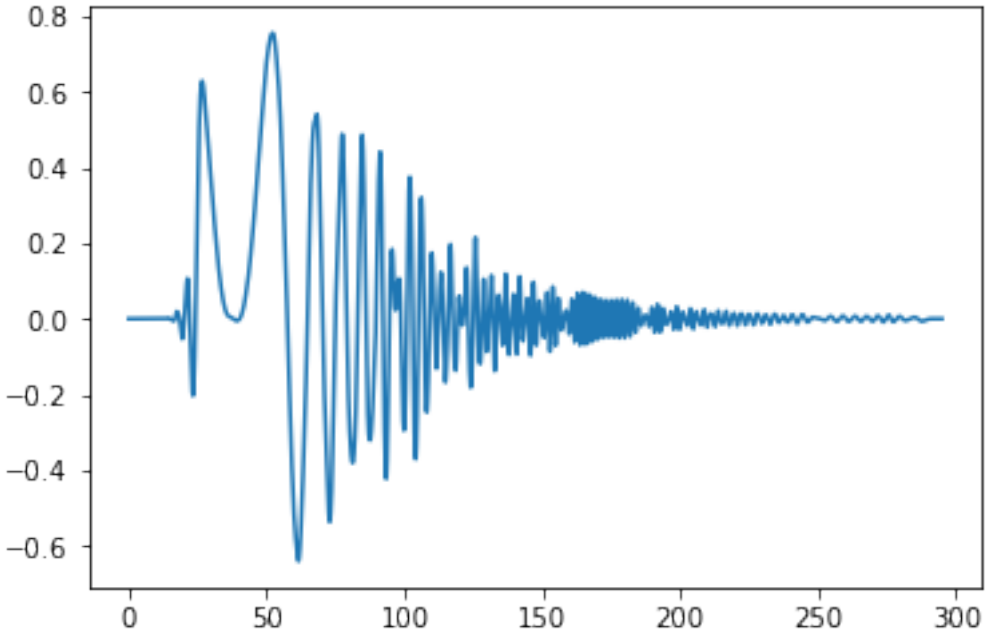

```
In [236]: ## Let's compute some wavelet coefficients

          (cA, cD) = dwt(y, 'db8')
          cA, cD
```

```
Out[236]: (array([  3.01750674e+01,   2.24816532e+01,   1.39375190e+01,
                    6.69155231e+00,   1.77825217e+00,   1.45125108e-01,
                   -4.90575612e-02,   1.45893530e-01,   2.82412388e+00,
                    8.39090792e+00,   1.61713237e+01,   2.47966212e+01,
                    3.18398238e+01,   3.38556104e+01,   2.74127217e+01,
                    1.14476991e+01,  -9.81804628e+00,  -2.60118140e+01,
                   -2.53542535e+01,  -5.33859009e+00,   1.89105167e+01,
                    2.29576801e+01,   1.39871525e-01,  -2.17278328e+01,
                   -1.07771646e+01,   1.67690478e+01,   1.29663404e+01,
                   -1.52909071e+01,  -9.72579016e+00,   1.68379894e+01,
                    1.15503654e+00,  -1.57698427e+01,   1.07682687e+01,
                    4.27332003e+00,  -1.36330230e+01,   1.18190202e+01,
                   -3.29345716e+00,  -5.24806152e+00,   1.00489185e+01,
                   -1.08252647e+01,   9.06780832e+00,  -6.43415840e+00,
                    4.01888852e+00,  -2.30076076e+00,   1.35643308e+00,
                   -1.07633333e+00,   1.29202049e+00,  -1.81985724e+00,
                    2.45710679e+00,  -2.97030348e+00,   3.11221817e+00,
                   -2.69221112e+00,   1.69302755e+00,  -3.71264138e-01,
                   -7.70463057e-01,   1.22205803e+00,  -8.33750313e-01,
                    1.64717399e-02,   5.23991845e-01,  -4.09754093e-01,
                   -5.07698887e-02,   2.63439713e-01,  -7.99239721e-02,
```

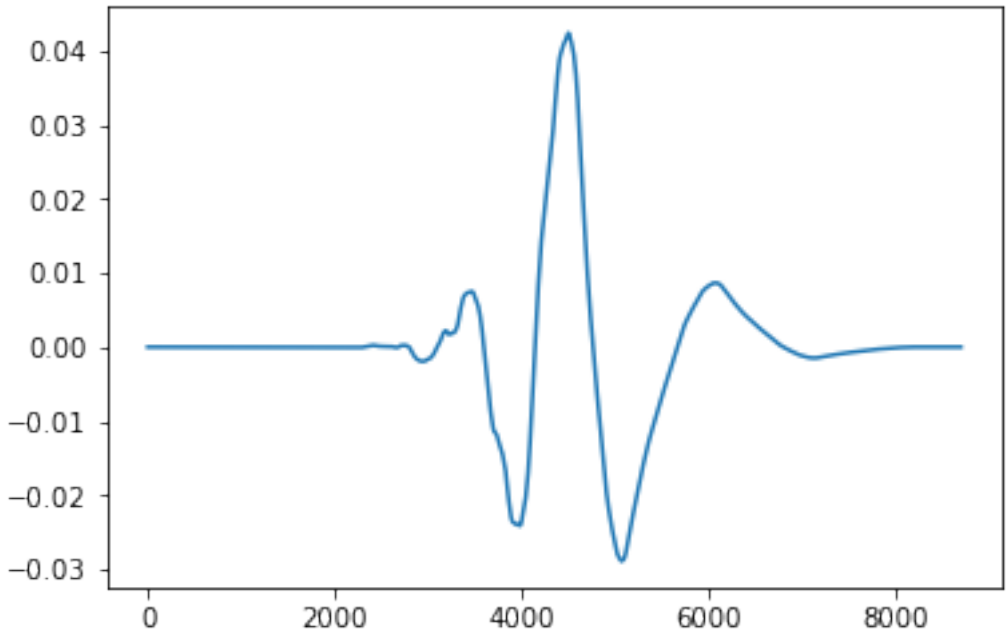
```
          -1.17080385e-01,   6.35448167e-02,   5.22330773e-02,
          -2.97731785e-02,  -2.67703896e-02,   8.13762958e-03,
           1.34017952e-02,   1.17563506e-03,  -4.75382283e-03,
          -2.62595525e-03,   3.74064984e-04,   1.13229577e-03,
           5.62611295e-04,  -8.24468785e-06,  -1.86902744e-04,
          -1.32693849e-04,  -4.27768047e-05,   6.54619767e-06,
           1.84155961e-05,   1.46224137e-05,   9.41138884e-06,
           6.84760854e-06,   4.38325112e-06,  -4.09296934e-06,
          -2.38900585e-05,  -4.91994233e-05,  -4.86067195e-05,
           3.14364818e-05,   2.02791154e-04,   3.10913459e-04,
           2.09183834e-05,  -7.62638544e-04,  -1.16148321e-03,
           3.30573065e-04,   3.01810443e-03,   2.03068232e-03,
          -4.86823537e-03,  -6.73081827e-03,   6.98536801e-03,
           1.36491063e-02,  -1.24577070e-02,  -2.08768093e-02,
           2.77200352e-02,   1.81928686e-02,  -5.41155858e-02,
           1.87715877e-02,   5.71810955e-02,  -8.88673353e-02,
           3.56741144e-02,   6.32115428e-02,  -1.35162554e-01,
           1.35181702e-01,  -6.74383109e-02,  -3.26556553e-02,
           1.27273544e-01,  -1.93960598e-01,   2.28242756e-01,
          -2.37500808e-01,   2.33429442e-01,  -2.26856012e-01,
           2.25360748e-01,  -2.32582847e-01,   2.47897325e-01,
          -2.65521885e-01,   2.73146261e-01,  -2.51891320e-01,
           1.79073117e-01,  -2.75947792e-02,  -2.99854272e-01]),
array([ -1.11040541e-02,   2.76636794e-02,  -6.79845512e-02,
           3.84632165e-02,  -4.67667119e-03,  -5.01230786e-03,
           1.24909834e-03,   7.78636755e-05,   8.99996997e-04,
           2.10271706e-03,   2.11827732e-03,  -1.51277071e-03,
          -8.75590841e-03,  -1.14657346e-02,   4.24334469e-03,
           3.45145347e-02,   3.25509144e-02,  -4.50743561e-02,
          -1.10558500e-01,   2.09463722e-02,   2.33464797e-01,
           2.69782397e-02,  -4.29480259e-01,  -1.09161867e-02,
           7.32275865e-01,  -3.07776367e-01,  -9.55065898e-01,
           1.20499657e+00,   2.54334710e-01,  -1.94679932e+00,
           2.02164133e+00,  -1.98918438e-01,  -2.21447849e+00,
           3.66388492e+00,  -3.44714112e+00,   1.84863858e+00,
           3.65929642e-01,  -2.49061485e+00,   4.13967498e+00,
          -5.23976589e+00,   5.89778399e+00,  -6.26259724e+00,
           6.42886769e+00,  -6.38430641e+00,   5.99152341e+00,
          -5.01066529e+00,   3.19104181e+00,  -4.63571075e-01,
          -2.78538601e+00,   5.51242036e+00,  -6.20925952e+00,
           3.79598150e+00,   1.00268169e+00,  -5.04225009e+00,
           4.61595070e+00,   4.42513699e-01,  -4.76821837e+00,
           2.68224702e+00,   3.15267219e+00,  -3.54366729e+00,
          -2.30315955e+00,   3.28408612e+00,   2.57633363e+00,
          -2.20624206e+00,  -3.31679999e+00,  -9.45728673e-03,
           2.94075001e+00,   2.58181192e+00,   1.78036814e-02,
          -2.21667712e+00,  -2.74787052e+00,  -1.78529588e+00,
          -2.44601073e-01,   1.09663784e+00,   1.90389535e+00,
```

```
 2.19756842e+00,   2.15484088e+00,   1.96324292e+00,
 1.75727295e+00,   1.61019157e+00,   1.54704038e+00,
 1.55809704e+00,   1.60537843e+00,   1.62386167e+00,
 1.52526841e+00,   1.21658888e+00,   6.44916424e-01,
-1.33282469e-01,  -8.93461544e-01,  -1.27160968e+00,
-9.58026268e-01,  -1.97557253e-02,   9.15603633e-01,
 1.00376862e+00,   5.91321506e-02,  -9.12267574e-01,
-6.49271358e-01,   5.33517445e-01,   7.95958187e-01,
-3.35339823e-01,  -7.53085958e-01,   3.81520129e-01,
 5.93790511e-01,  -5.77216487e-01,  -2.28486209e-01,
 6.68049314e-01,  -3.29586201e-01,  -2.83660993e-01,
 5.80748853e-01,  -4.20655149e-01,   2.98361649e-02,
 3.12147081e-01,  -4.64613591e-01,   4.34494913e-01,
-2.99816450e-01,   1.39839795e-01,  -3.83542005e-03,
-8.98032356e-02,   1.42079954e-01,  -1.61761628e-01,
 1.58077902e-01,  -1.37717127e-01,   1.04665362e-01,
-6.15695514e-02,   1.17457504e-02,   7.18452021e-03,
-2.84689639e-02,   6.08791218e-02,  -4.77027262e-02,
 4.22241287e-03,   4.78380943e-02,  -9.44417179e-02]))
```

In [238]:
```
fw_len = 1024*size(cA) + size(phi)
fw_recon = zeros(fw_len, float_)

for k in range(size(cA)):
    fw_recon[(1024*k):(1024*k+size(phi))] = fw_recon[(1024*k):(1024*k+size(phi))] + \
        cA[k]*phi - cD[k]*psi

plot(linspace(0,size(fw_recon)/512,size(fw_recon)),fw_recon/sqrt(2));
```



```
In [239]: ## Let's explore orthogonality

          phiOne = zeros(size(phi)+1024);
          phiTwo = zeros(size(phi)+1024);

          phiOne[0:size(phi)] = phi
          phiTwo[1024:(1024+size(phi))] = phi

In [240]: sum(phiOne*phiTwo)

Out[240]: 9.110898138904068e-18

In [241]: psiOne = zeros(size(psi)+1024);
          psiTwo = zeros(size(psi)+1024);
          psiOne[0:size(psi)] = psi
          psiTwo[1024:(1024+size(psi))] = psi

In [242]: sum(psiOne*psiTwo)

Out[242]: 0.0

In [243]: sum(psiOne*phiOne)

Out[243]: 2.2768245622195593e-18

In [229]: plot(psiTwo)

Out[229]: [<matplotlib.lines.Line2D at 0x1149f1208>]
```



1.1.2 Comments.

Well, some of those reconstructions are terrible!
Let's explore in class, might be interesting.

```
In [ ]:
```