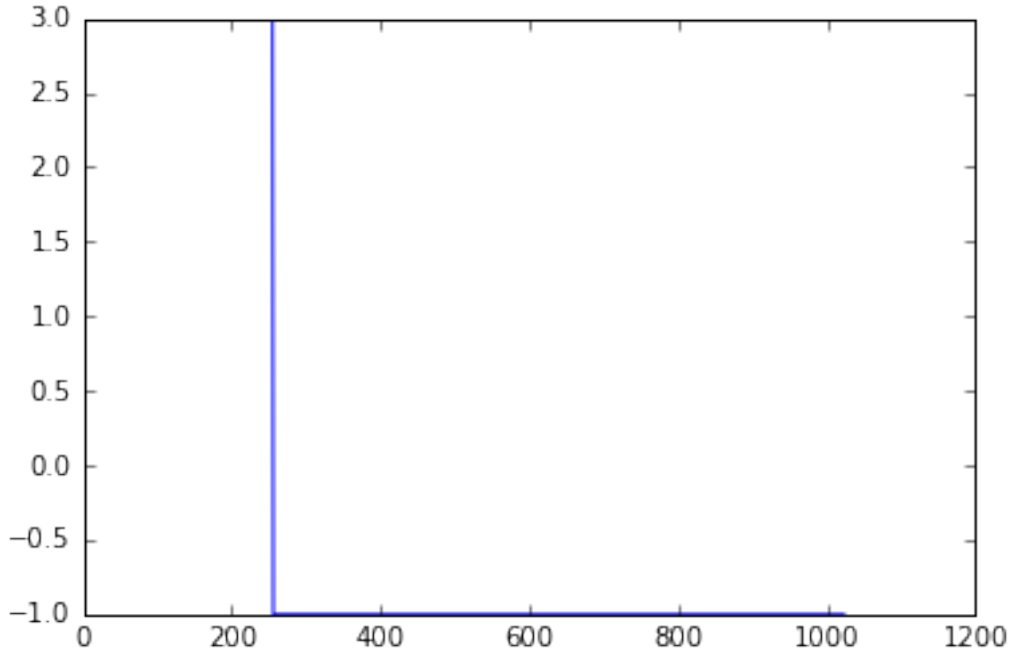
OmpData

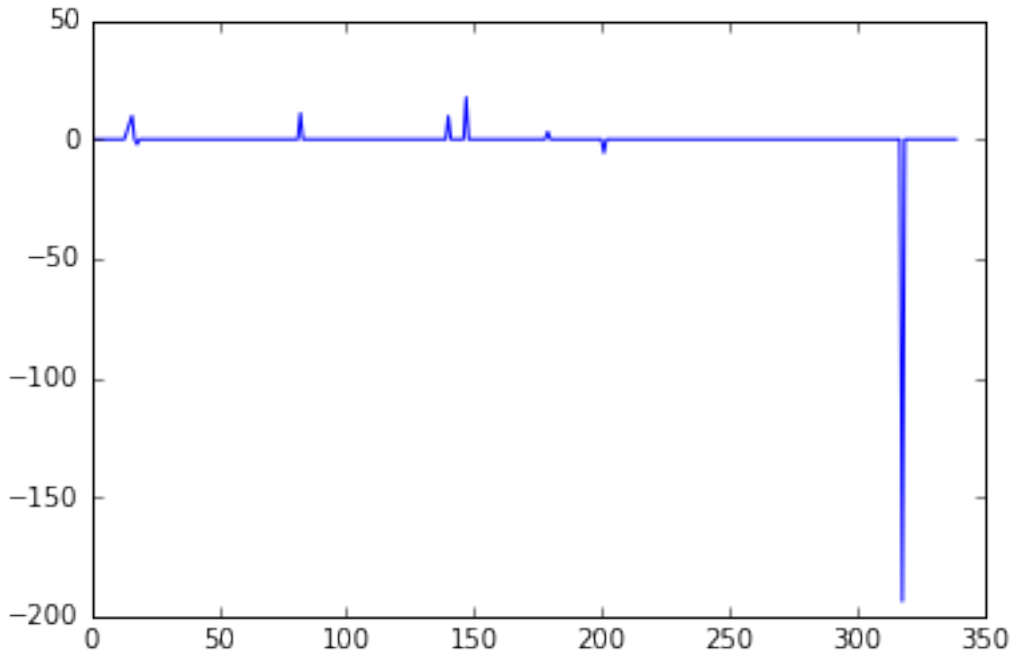
April 7, 2016

In [8]:
```
print(__doc__)
%matplotlib inline
import numpy as np
import matplotlib.pylab as plt
from  sklearn.linear_model import OrthogonalMatchingPursuit
from sklearn.decomposition import SparseCoder


def ricker_function(resolution, center, width):
    """Discrete sub-sampled Ricker (Mexican hat) wavelet"""
    x = np.linspace(0, resolution - 1, resolution)
    x = ((2 / ((np.sqrt(3 * width) * np.pi ** 1 / 4)))
         * (1 - ((x - center) ** 2 / width ** 2))
         * np.exp((-(x - center) ** 2) / (2 * width ** 2)))
    return x


def ricker_matrix(width, resolution, n_components):
    """Dictionary of Ricker (Mexican hat) wavelets"""
    centers = np.linspace(0, resolution - 1, n_components)
    D = np.empty((n_components, resolution))
    for i, center in enumerate(centers):
        D[i] = ricker_function(resolution, center, width)
    D /= np.sqrt(np.sum(D ** 2, axis=1))[:, np.newaxis]
    return D


resolution = 1024
subsampling = 3  # subsampling factor
width = 100
n_components = resolution / subsampling

# Compute a wavelet dictionary
D_fixed = ricker_matrix(width=width, resolution=resolution,
                        n_components=n_components)
D_multi = np.r_[tuple(ricker_matrix(width=w, resolution=resolution,
                                    n_components=np.floor(n_components / 5))
                      for w in (10, 50, 100, 500, 1000))]

# Generate a signal
y = np.linspace(0, resolution - 1, resolution)
first_quarter = y < resolution / 4
y[first_quarter] = 3.
```

```
y[np.logical_not(first_quarter)] = -1.

print("Original")
plt.plot(y)
plt.show()

omp = OrthogonalMatchingPursuit(n_nonzero_coefs=10)
omp.fit(np.transpose(D_multi), y)
coef = omp.coef_
idx_r, = coef.nonzero()
print("Generated signal")
plt.plot(coef)
plt.show()
```

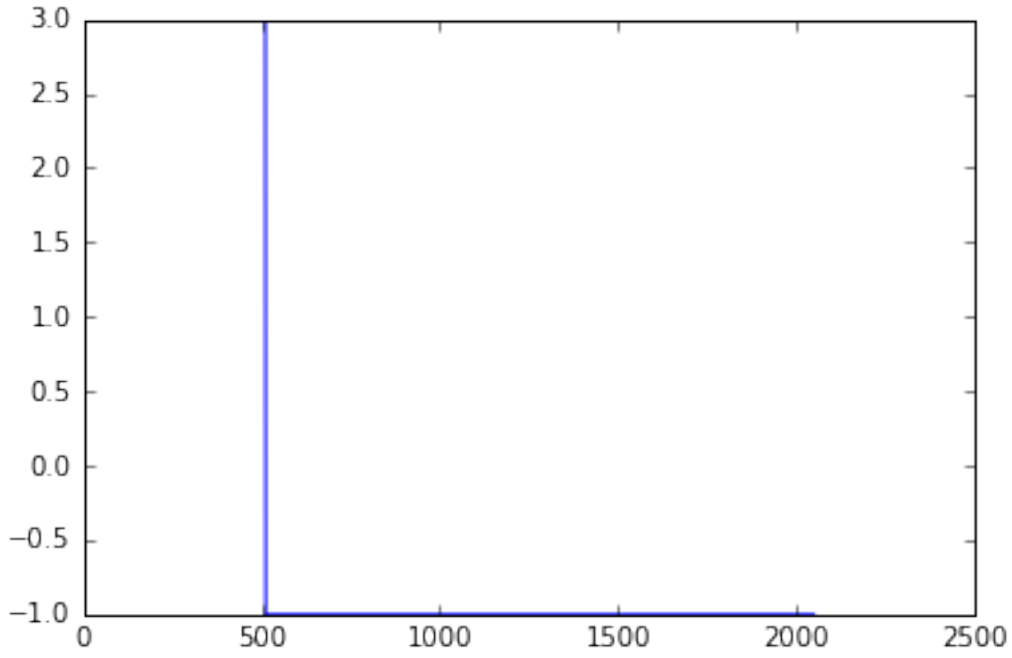
```
Automatically created module for IPython interactive environment
Generated signal

D:\Program Files\Anaconda\lib\site-packages\ipykernel\__main__.py:21: DeprecationWarning: using a non-int
```



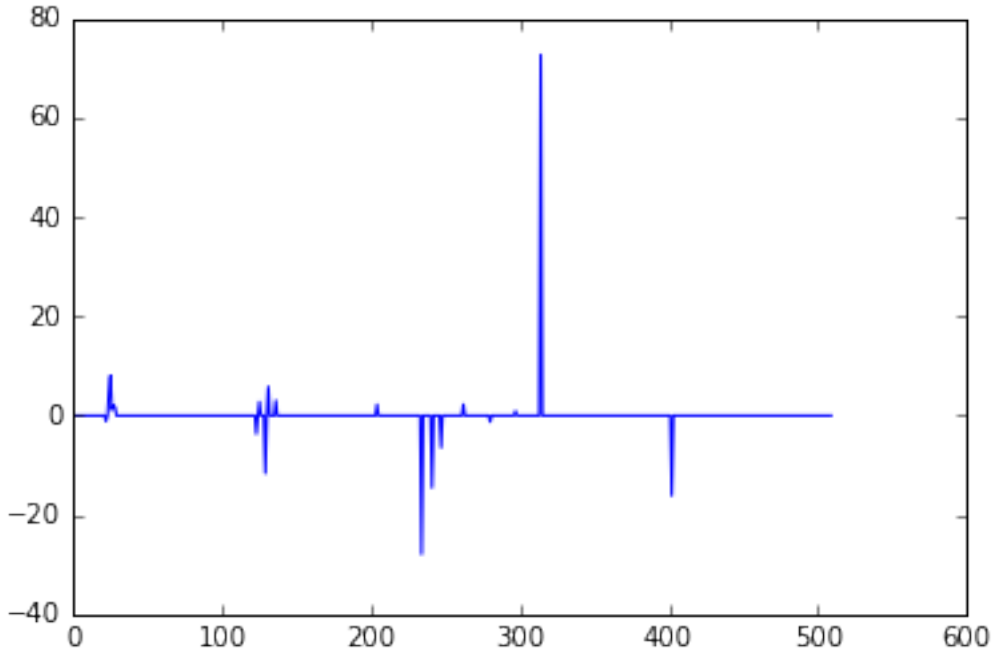


In [11]:
```
resolution = 2048
subsampling = 4  # subsampling factor
width = 100
n_components = resolution / subsampling

# Compute a wavelet dictionary
D_fixed = ricker_matrix(width=width, resolution=resolution,
                        n_components=n_components)
D_multi = np.r_[tuple(ricker_matrix(width=w, resolution=resolution,
                                    n_components=np.floor(n_components / 5))
                      for w in (10, 50, 100, 500, 1000))]

# Generate a signal
y = np.linspace(0, resolution - 1, resolution)
first_quarter = y < resolution / 4
y[first_quarter] = 3.
y[np.logical_not(first_quarter)] = -1.

print("Original")
plt.plot(y)
plt.show()

omp = OrthogonalMatchingPursuit(n_nonzero_coefs=20)
omp.fit(np.transpose(D_multi), y)
coef = omp.coef_
idx_r, = coef.nonzero()
print("Generated signal")
plt.plot(coef)
plt.show()
```

```
Original
D:\Program Files\Anaconda\lib\site-packages\ipykernel\__main__.py:21: DeprecationWarning: using a non-int
```




```
Generated signal
```



In [ ]: