Computer Assignment (CA) No. 11: Autocorrelation And Power Spectral Density

Problem Statement

Recall the autocorrelation function is defined as:

$$R(\tau) = \sum_{n=0}^{N-1} x[n]x[n-\tau], \tau = 0, 1, 2, \ldots, M$$

Compute and plot the autocorrelation function for the following signals, and then plot the power spectral density by computing the Fourier transform of the autocorrelation function.

1. Gaussian white noise: N=100, M=20.
2. An impulse function, : N = 100, M=20.
3. A periodic impulse train with a period of 20 samples: N = 200, M = 60.
4. A sinewave with a period of 20 samples: N = 200, M = 60.
5. Repeat no. 4 for N = 14, 17, 20, 23, 26. Analyze the behavior that you observe and relate it to the period of the signal.
6. The sum of (1) and (4) at an SNR of 10 dB (assume the sinewave is the signal and the Gaussian white noise is the noise): N = 200, M = 60. Explain what you observe.

Approach and Results

The autocorr() and generate_sine() functions were first reconstructed as can be seen below. The numpy and scipy libraries were also loaded, and a plot style was applied because it looks awesome. The power spectral density is considered,

$$F[R_{xx}(\tau)] = S_{xx}(\omega)$$

where $F[\ldots]$ is the fourier transform.

In [7]:
```
%pylab inline
from scipy.stats import norm
plt.style.use('bmh')

def generate_sine(freq, duration, fs, snr):
# freq: pure signal frequancy of duration with sample rate fs
# and signal to noise ratio snr in dB.
    samples = fs*duration  # number of samples is equal to the samples p
er second * the number of seconds
    t = np.linspace(0, duration, samples)  # generate time vector
    freq_radians = (2*np.pi)*freq
    x = np.sin(freq_radians*t)  # generate signal
    e = norm.rvs(size=samples)
    c = sqrt(np.var(x)/(np.var(e)*10**(snr/20)))
    print 20*log(var(x)/var(c*e))/log(10)
    return x + c*e

def autocorr(x, M):
    '''
        Autocorrlates signal X with a delay of M samples.
        Returns $R_{xx}(t, t-\tau). The function autocorrlated in indexe
d past the lag point.
    '''
    Ryy = zeros(M+1)
    for tau in range(M+1):
        #print x[M:]
        #print x[M-tau:-tau]
        Ryy[tau] = sum( x[M:]*x[M-tau:len(x)-tau] )
    return Ryy
```

```
Populating the interactive namespace from numpy and matplotlib
```

Task 1

White noise $N(t)$ is defined as a random signal which is normally distributed, contains a flat power spectral density ($S_{NN}(\omega) = c$ for all ω) and an expected value of 0, ($E[N(t)] = 0$).

The autocorrelation of white noise is calculated as follows,

$$R_{NN}(t, t-\tau) = E[N(t)N(t-\tau)],$$

but as every sample in white noise is independent, white noise should result in an autocorrelation of zero for every value except when $\tau = 0$. Further, when $\tau = 0$, the autocorrelation of $N(t)$ becomes the $E[N(t)^2]$, and with $E[N(t)] = 0$ the autocorrelation becomes

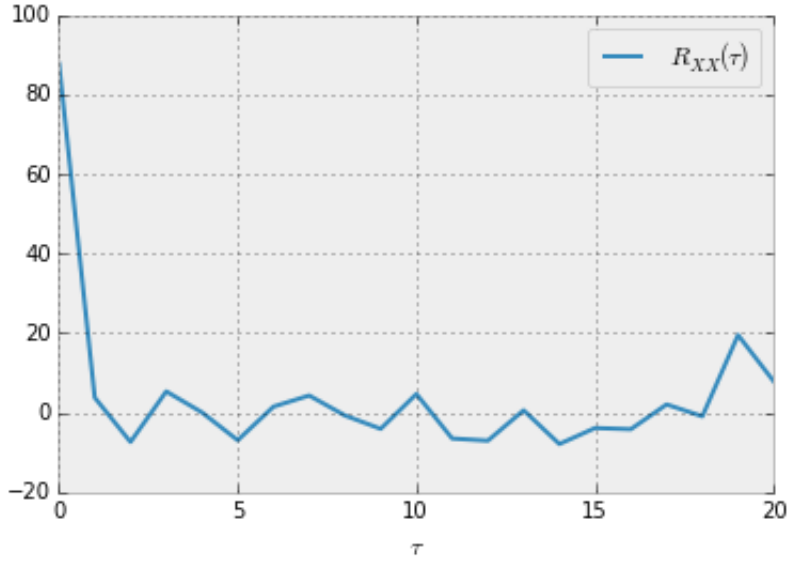
$$R_{NN}(\tau) = \sigma_N^2 \delta(\tau)$$

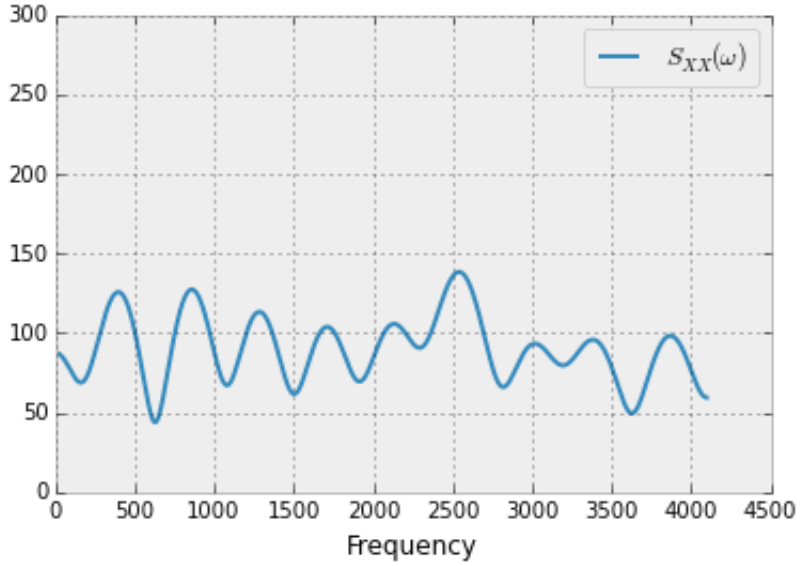
Strangely, this is not exactly what appears in the plot below. The autocorrelation does spike when $\tau = 0$, but the autocorrelation does not appear to be diminishing as the lag increases. Instead, it fluctuates around the predicted value.

The power spectral density of white noise should result in a uniform distribution. The plot after R_{xx} displays the the power spectal density $S_{xx}(\omega)$, and it is not flat. This is likely due to the limited sample size.

In [14]:
```
# 1) White Noise

N = 100
M = 20
x = norm.rvs(size=N)
Rxx = autocorr(x, M)
fig = figure()
ax = subplot(111)
ax.plot(range(M+1), Rxx, label=r"$R_{XX}(\tau)$")
ax.set_xlabel(r"$\tau$")
ax.legend()

fig = figure()
fuck = fft.rfft(Rxx, n=8192)
plot(abs(fuck), label=r"$S_{XX}(\omega)$")
xlabel("Frequency")
legend()
ylim([0, 300])
```

Out[14]: (0, 300)





Task 2

The autocorrelation of an impulse response, $x(t) = \delta(t - 10)$, is displayed below. As the plot illustrates, the autocorrelation is zero for all of τ. The calculation for the autocorrelation of $x(t)$ is as follows:

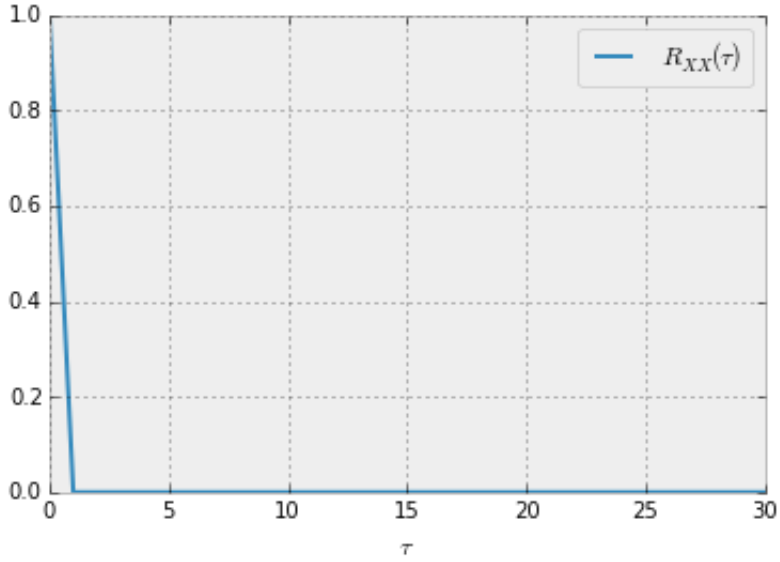
$$R_{xx}(t, t - \tau) = E[\delta(t - 10)\delta(t - 10 - \tau)]$$

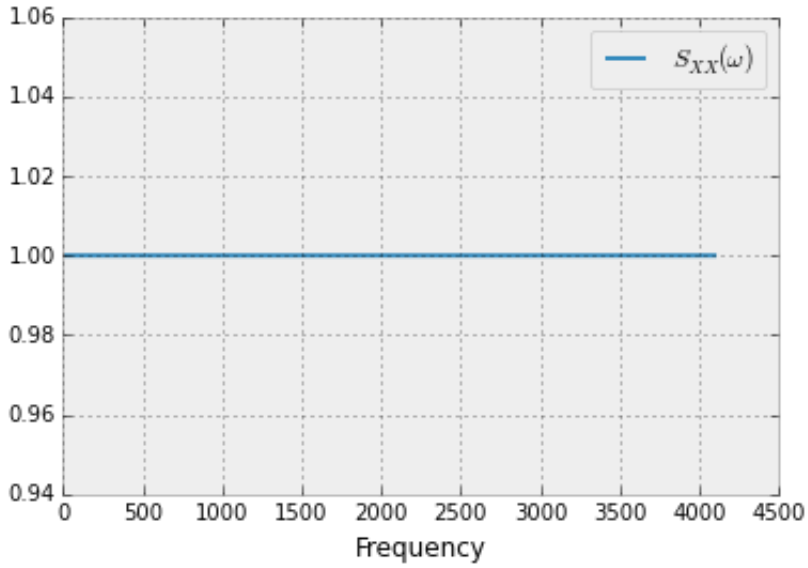
Note that when $t - 10 = t - 10 - \tau = 0$, in other words when $\tau = 0$, there should be an impulse! The fourier transform of the autocorrelation will, therefore, be uniform for all frequency content. This is displayed in the second plot below.

In [15]:
```
# 2) Impulse Response
N = 100
M = 30
x = 80*[0] + [1] + 9*[0]
print x
x = array(x)
Rxx = autocorr(x, M)
print Rxx
fig = figure()
ax = subplot(111)
ax.plot(range(M+1), Rxx, label=r"$R_{XX}(\tau)$")
ax.set_xlabel(r"$\tau$")
ax.legend()

fig = figure()
fuck = fft.rfft(Rxx, n=8192)
plot(abs(fuck), label=r"$S_{XX}(\omega)$")
xlabel("Frequency")
legend()
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

Out[15]: <matplotlib.legend.Legend at 0x7fdabb9ad610>





Task 3

The autocorrelation of an impulse train also know as the Dirac Comb, $x(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$ where the period T in this case is 20 returns an impulse train with an identical period. This can be seen in the first plot below.

$$R_{yy}(t, t-\tau) = E[\sum_{n=-\infty}^{\infty} \delta(t - nT) \sum_{k=-\infty}^{\infty} \delta(t - \tau - kT)] = \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} E[\delta(t - nT)\delta(t - \tau - kT)]$$

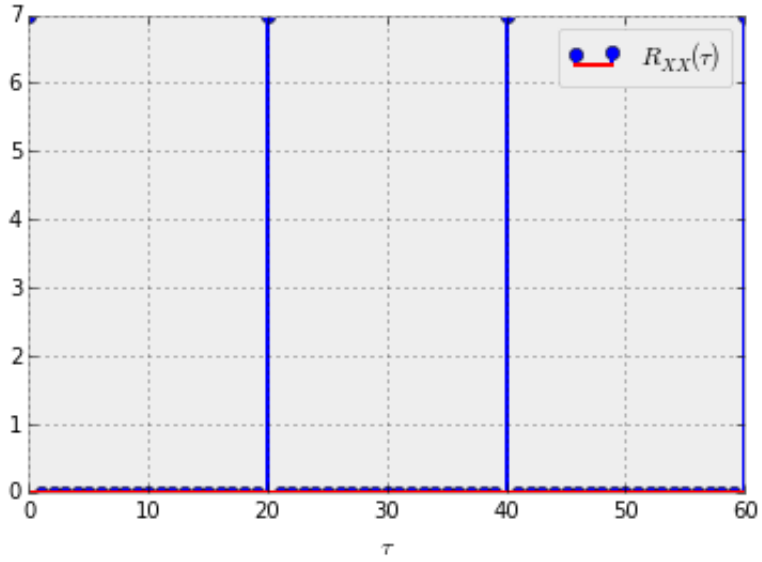
This signal will deliver a value when $t - nT = t - \tau - kT = 0$

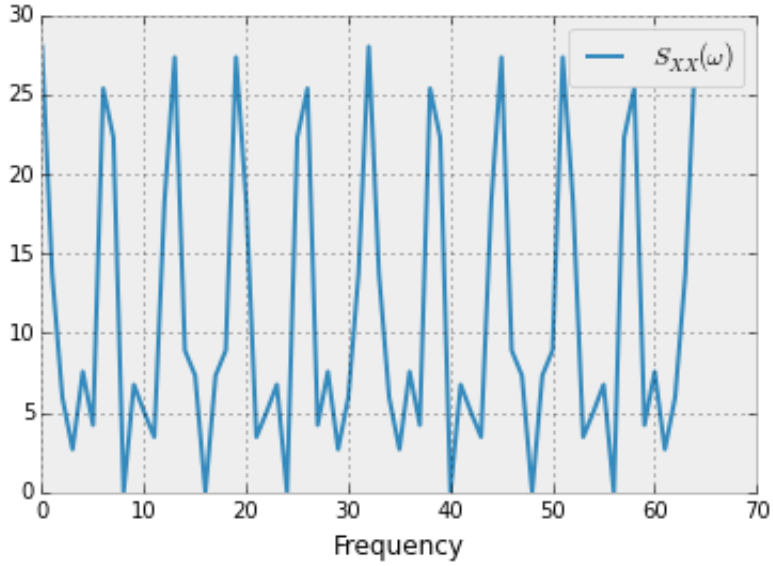
$$t - nT - (t - \tau - kT) = 0 = \tau + (k - n)T = 0$$

As k and n are integers, τ must be equal to a multiple of the period T for the expected value to have any value. This, in turn, would generate an impulse train with an identical period to that of $x(t)$. The fourier transform of the autocorrelation function will also generate an impulse train as proven in signals. This can be seen in the second plot below.

In [16]:
```
# 3) Impulse Train with period of 20, N = 200, M = 60
x = [1] + [0]*19
x = 10*x
x = np.array(x)
M = 60
Rxx = autocorr(x, M)
fig = figure()
ax = fig.add_subplot(111)
ax.stem(range(M+1), Rxx, label=r"$R_{XX}(\tau)$")
ax.set_xlabel(r"$\tau$")
ax.legend()

fig = figure()
FFTsig = fft.rfft(Rxx, n=128)
plot(abs(FFTsig), label=r"$S_{XX}(\omega)$")
xlabel("Frequency")
legend()
```

```
Out[16]: <matplotlib.legend.Legend at 0x7fdabc473f50>
```





Task 4

The autocorrelation of a sine wave, $x(t) = a\sin(wt)$, returns a sine wave as illustrated by the plot below with an identical period and an amplitude equal to $140/2$. Explaining this value was an interesting journey. First, I examined the autocorrelation of a continous sine.

$$R_{xx}(t, t-\tau) = a^2 E[\sin(u)\sin(u-\tau)] = \frac{a^2}{2}E[\cos(u-(u-\tau)) - \cos(u+u-\tau)]$$

$$R_{xx}(t, t-\tau) = \frac{a^2}{2}E[\cos(\tau) - \cos(2u-\tau)]$$

The expected value of cos for all values of t is zero leaving,

$$R_{xx}(t, t-\tau) = \frac{a^2}{2}\cos(\tau)$$

... but, then I cried because everything we do is digital and our equation for autocorrelation is reaaaaallllly this.

$$R[n, n-\tau] = \sum_{n=0}^{N-1} x[n]x[n-\tau], \tau = 0, 1, 2, \ldots, M$$

thus

$$R_{xx}[n, n-\tau] = \sum_{n=0}^{N-1} a^2 \sin(\omega n)\sin(\omega(n-\tau)), \tau = 0, 1, 2, \ldots, M$$

lets assume $a^2 = 1$ for simplicity. The trigonometry still holds from the continous example.

$$R_{xx}[n, n-\tau] = \frac{1}{2}\sum_{n=0}^{N-1}(\cos(\omega\tau) - \cos(2\omega n - \tau)), \tau = 0, 1, 2, \ldots, M$$

The summation of $\cos(2\omega n - \tau)$ will average to zero, leaving

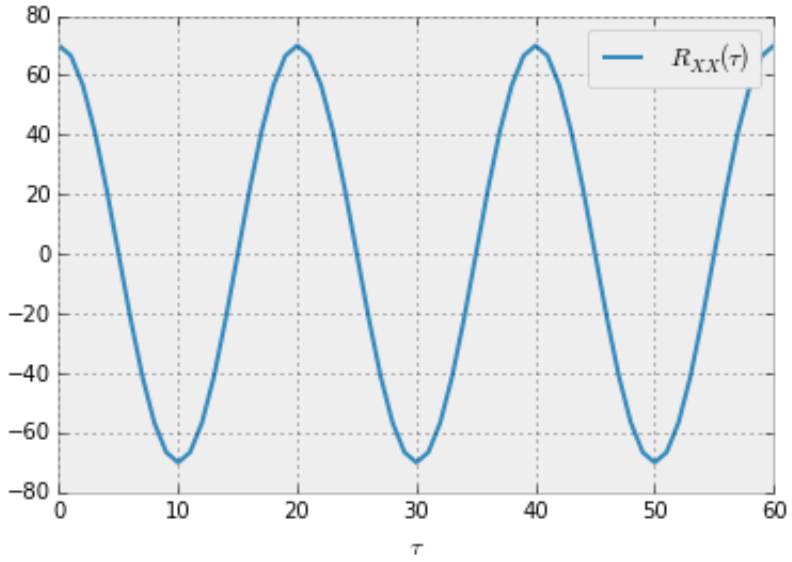
$$R_{xx}[\tau] = \frac{1}{2}\sum_{n=0}^{N-1}\cos(\omega\tau) = \frac{N}{2}\cos(\omega\tau).$$

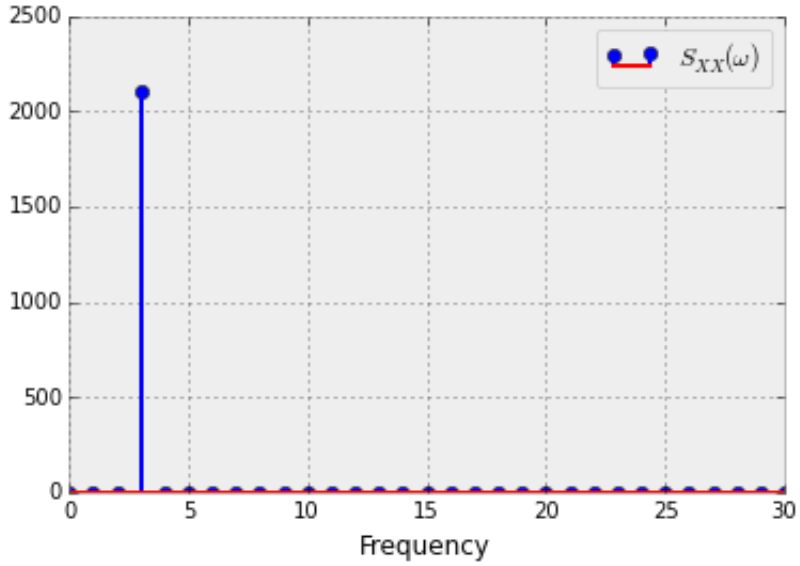
Wait a second! That still is not equal to 70. Your signal is 200 samples long! What is this 70 giberish. I know this had me up a tree. The signal is 200 samples in length, therefore to calculate the autocorrelation, the function starts the summations at the lag value to ensure the calculation will remain in the bounds of the vector. Therefore, despite taking the autocorrelation of signal that is 200 samples long, the result is the autocorrelation of a signal $N - M$ samples long, and, therefore, the amplitude of the signal after autocorrelation is $(N - M)/2 = 70$.

$S_{XX}(\omega)$ will contain a single impulse as the frequency content of $R_{XX}(\tau)$ is at one frequency. This is proven in the second plot below.

In [18]:
```
# 4) A sinewave with a period of 20 samples: N = 200, M = 60.
N = 200
M = 60
t = np.arange(N)

y = sin(2*pi/20*t)
Ryy = autocorr(y, M)
fig = figure()
ax = fig.add_subplot(111)
ax.plot(range(M+1), Ryy, label=r"$R_{XX}(\tau)$")
ax.set_xlabel(r"$\tau$")
ax.legend()

fig = figure()
FFTsig = fft.rfft(Ryy, n=60)
stem(abs(FFTsig), label=r"$S_{XX}(\omega)$")
xlabel("Frequency")
legend()
```

Out[18]: <matplotlib.legend.Legend at 0x7fdabbcac190>





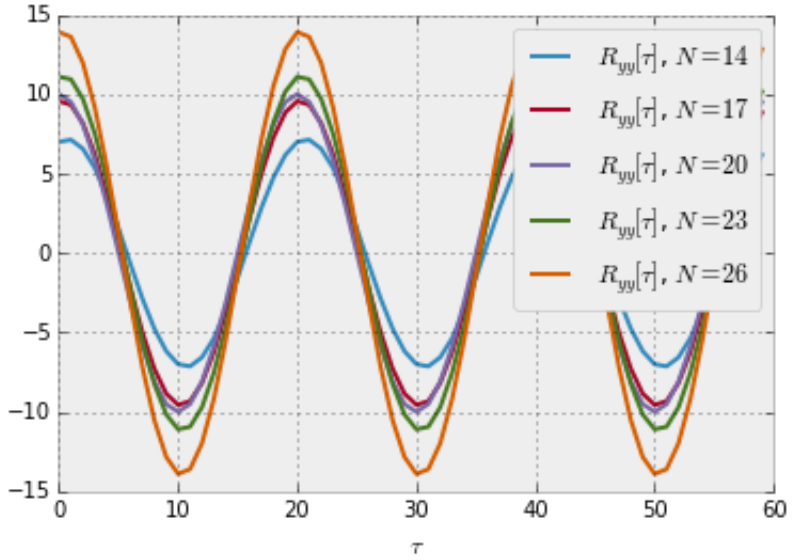
Task 5

Utilizing the N reduction quirk of the autocorrelation function, N was reduced for the following signals. As calculated for task 4, the amplitude of the resulting waveform is equal to $\frac{N}{2}$.

$S_{XX}(\omega)$ would be equal to the PSD for task 4 as the signals are identical. The only variation is the amplitude; therefore, the magnitude of the impulses would differ.

In [15]:
```
# 4) Repeat no. 4 for N = 14, 17, 20, 23, 26.
#    Analyze the behavior that you observe and relate it to the period o
f the signal.
N = [14, 17, 20, 23, 26]
M = 60
t = np.arange(200)
y = sin(2*pi/20*t)
fig = figure()

for n in N:
    Ryy = autocorr(y, 200 - n)
    ax = fig.add_subplot(111)
    ax.plot(Ryy[0:60], label=r"$R_{yy}[\tau]$, $N = %d$" %(n))
    ax.legend()

ax.set_xlabel(r"$\tau$")
```

Out[15]: <matplotlib.text.Text at 0x7f8d2e75b410>



Task 6

As displayed also in computer assignment 9, the autocorrelation of a signal containing white noise filters out a portion of the noise.

If x representes the sine wave, a is a scaler, and n is a white noise signal, the output signal can be represented in the following form:

$$y = x + a \times n.$$

The autocorrelation of the signal would be equal to

$$R_{yy}(\tau) = R_{xx}(\tau) + aR_{xn}(\tau) + aR_{nx}(\tau) + a^2 R_{nn}(\tau),$$

but as $x(t)$ and $n(t)$ are independent, their crosscorrelation (R_{xn} and R_{nx}) is equal to a product of their means which is equal to zero, reducing the autocorrelation sum to

$$R_{yy}(\tau) = R_{xx}(\tau) + a^2 R_{nn}(\tau)$$

Luckily, the autocorrelation of white noise is zero except when the lag value is equal to zero, thus $R_{nn}(\tau) = R_{nn}(\tau)\delta(\tau)$ which simplifies to $\sigma_n^2\delta(\tau)$ as the $E[n(t)] = 0$. This results in the final expression for the autocorrelation for output signal with a specific SNR:

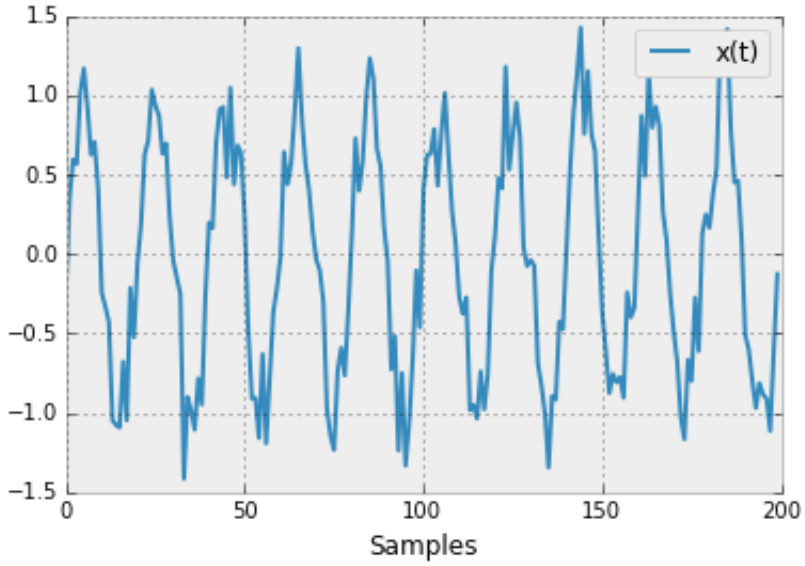
$$R_{yy}(\tau) = R_{xx}(\tau) + a^2\sigma_N^2\delta(\tau).$$

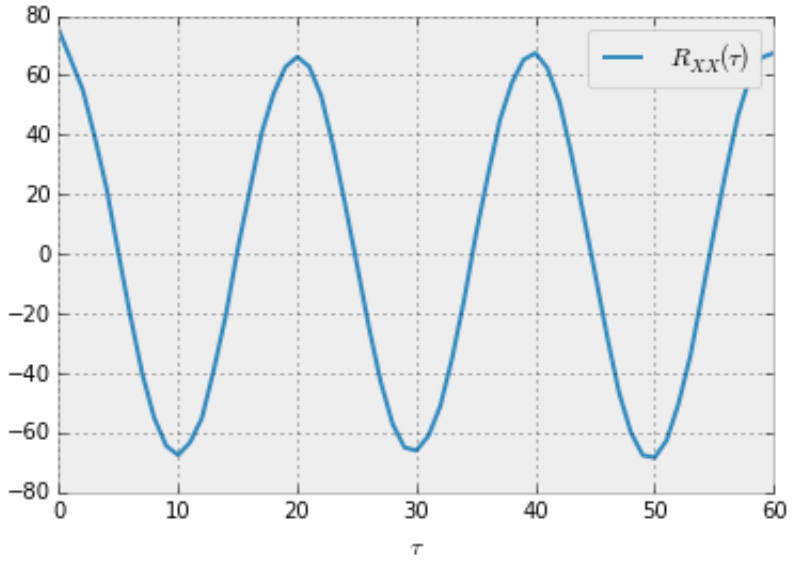
The amplitude of the waveform after the spike from the noise when τ is equal to 0 appears to be identical to that of the signal in task 4 which is predicted by the equation above. The power spectual density will be equal to an impulse at the frequency of the original signal.

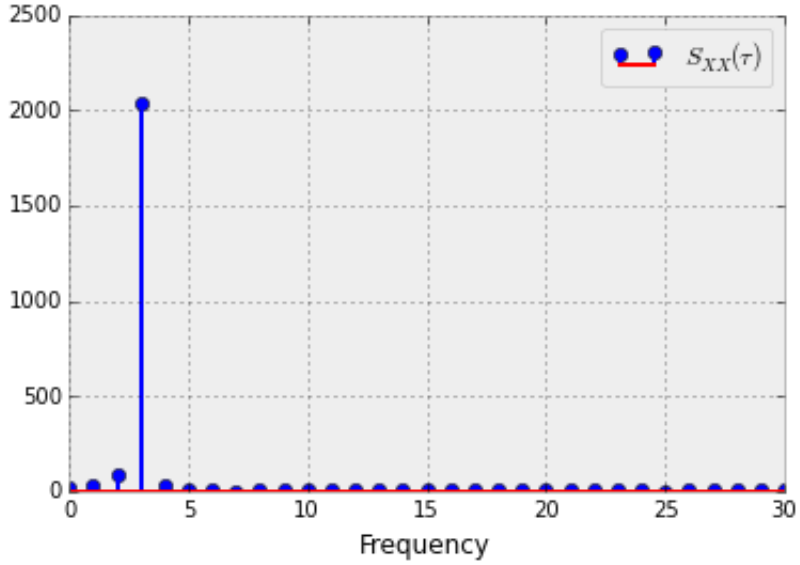
In [21]:
```
N = 200
M = 60
p = 10
x = generate_sine(p, N, 1, 20)
plot(x, label=r"x(t)")
legend()
xlabel("Samples")
Rxx = autocorr(x, M)
fig = figure()
ax = fig.add_subplot(111)
ax.plot(range(M+1), Rxx, label=r"$R_{XX}(\tau)$")
ax.set_xlabel(r"$\tau$")
ax.legend()

fig = figure()
FFTsig = fft.rfft(Rxx, n=60)
stem(abs(FFTsig), label=r"$S_{XX}(\tau)$")
xlabel("Frequency")
legend()
```

```
20.0
```

Out[21]: <matplotlib.legend.Legend at 0x7fdabba0f710>







Conclusions

Through the derivations, it has been displayed that autocorrelation cancels any non-periodic signals, when $\tau > 0$. Application of the theory provides slightly different results. For example, the autocorrelation of white noise should result in only a single impulse at $\tau = 0$. Instead, our discrete computations of autocorrelation of white noise provided an impulse at $\tau = 0$ followed by what appears to be random noise at a much lower amplitude centered around a zero. When white noise is added to the sine wave at a low SNR, autocorrelation does not recover the original signal without distortion. I'm not quite sure why in discrete application these phenomenons occur. The limited resolution of digital signals could be the reason.