session-1B-solution

September 10, 2015

1 Problem Set 1B (Spikes and Spike-Train Statistics)

Run the following cell to set up for plotting:

```
In [2]: %matplotlib inline
        import matplotlib.pyplot as plt
        import numpy as np
```

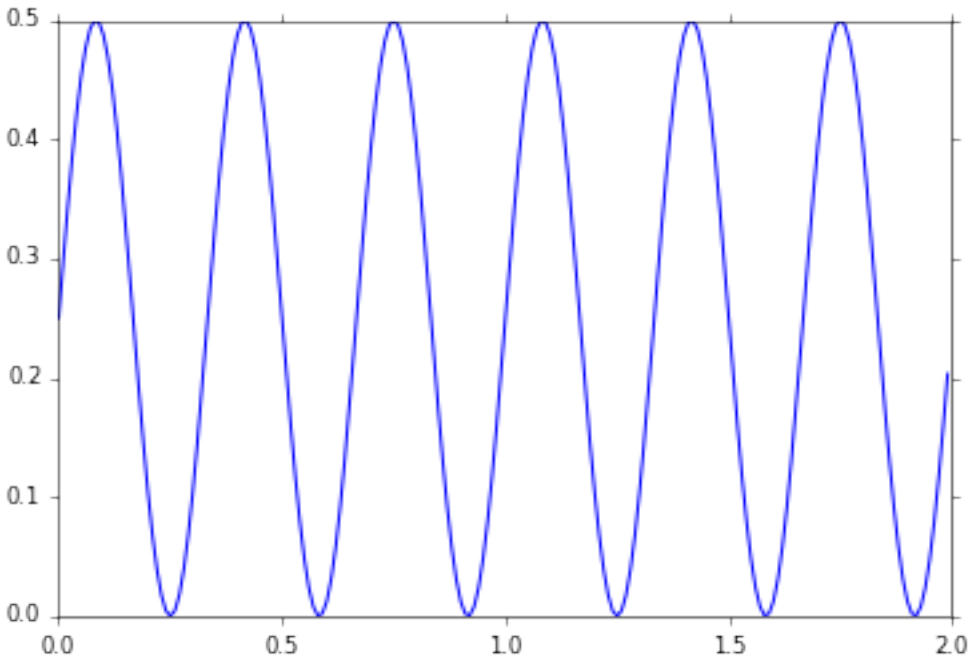
1.1 Question 1

Generate Poisson spike trains with a time-varying probability function
$r(t) = 1/4[\sin(2\pi\omega t) + 1]$
where $r(t)$ represents the probability of spiking in each bin, at a bin size of 10 msec, ω =3Hz, and $0 < t < 2$ sec.
Plot $r(t)$

```
In [5]: def r(t):
            return 0.25 * (np.sin(2 * np.pi * 3 * t) + 1)

        t = np.arange(0, 2, 0.01)
        plt.plot(t, r(t))
        plt.plot()
```

```
Out[5]: [<matplotlib.lines.Line2D at 0x10fc7a710>]
```



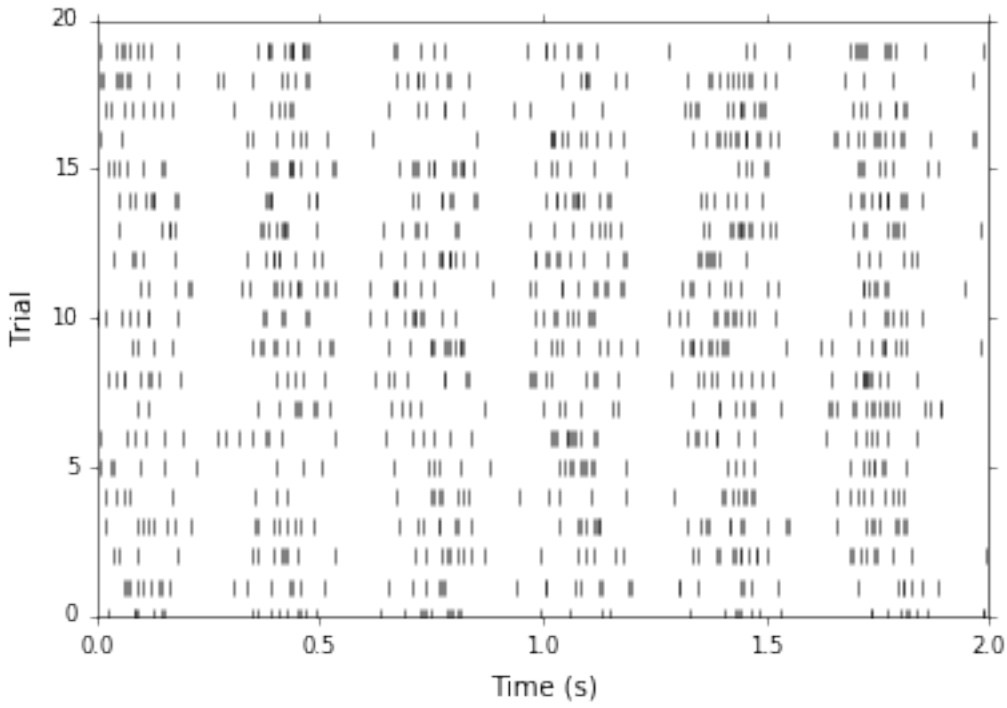
Generate 20 independently simulated spike trains and plot them as rasters. (Hint: you can plot each spike train in a loop by passing `hold=1` to the plot function.

```
In [75]: def make_spikes(T, p):
             """Generate a train of spikes using a Poisson generator.

             T - 1D array with time bins
             p - function that returns probability density (rate) as function of time.
             returns array of spike times
             """
             # calculate dt for each bin
             dt = np.diff(T)
             # get probability in each bin; note last value of t has to be dropped
             ps = p(T[:-1]) * dt * 100
             # make boolean array with true for bins where there is a spike
             spikes = ps > np.random.uniform(size=ps.size)
             # convert dense array to list of spike times
             return T[np.nonzero(spikes)[0]]

         # generate spikes on grid size of 1 ms
         t = np.arange(0, 2, 0.001)
         trials = [make_spikes(t, r) for i in range(20)]
         for i, trial in enumerate(trials):
             # generate a raster by making t,trial pairs
             y = np.ones(trial.size) * i
             plt.plot(trial, y, "k|", hold=1)
         plt.xlabel("Time (s)")
         plt.ylabel("Trial")
         plt.xlim([0,2])

Out[75]: (0, 2)
```



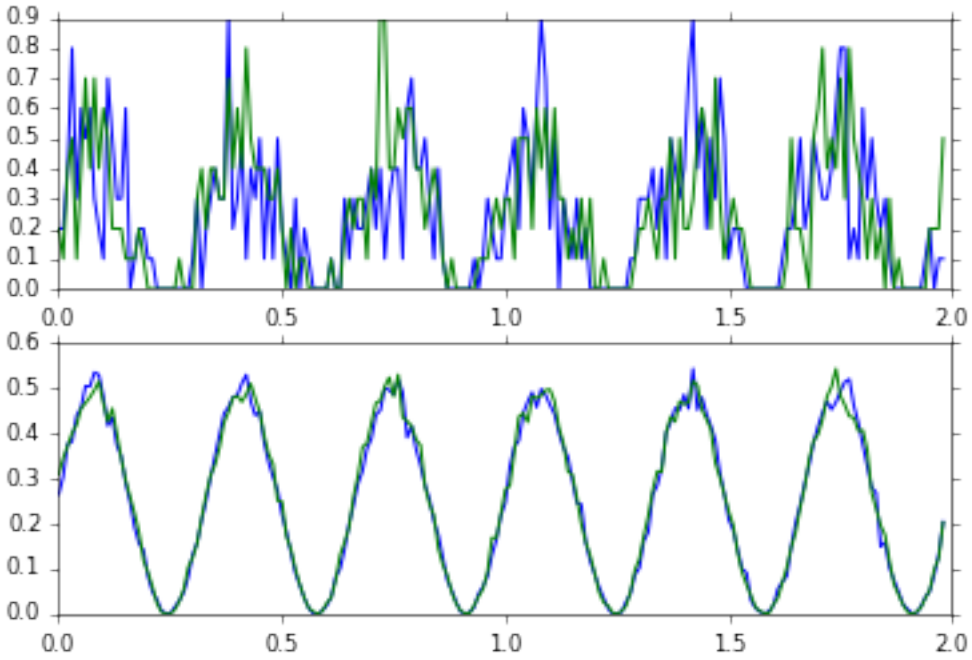
Show two PSTHs, each averaged from 10 independently simulated spike trains, and two PSTHs each averaged from 1000 spike trains. How do these PSTHs relate to $r(t)$? What do you learn by comparing and contrasting these PSTHs?

```
In [76]: def spike_hist(trials, bins):
             # first collate all the events into a single array
             spikes = np.concatenate(trials)
             # NB: returns counts, bins (a tuple)
             return np.histogram(spikes, bins)

         # use 10 ms bins
         bins = np.arange(0, 2, 0.01)
         N = 10
         psth_1,b = spike_hist([make_spikes(t, r) for i in range(N)], bins)
         psth_2,b = spike_hist([make_spikes(t, r) for i in range(N)], bins)
         plt.subplot(211)
         # NB: the last bin value sets the right edge of the last bin, so the length gets reduced by on
         plt.plot(bins[:-1], psth_1/N, bins[:-1], psth_2/N)

         N = 1000
         psth_1,b = spike_hist([make_spikes(t, r) for i in range(N)], bins)
         psth_2,b = spike_hist([make_spikes(t, r) for i in range(N)], bins)
         plt.subplot(212)
         plt.plot(bins[:-1], psth_1/N, bins[:-1], psth_2/N, hold=1)

Out[76]: [<matplotlib.lines.Line2D at 0x111c7db70>,
          <matplotlib.lines.Line2D at 0x111bcf550>]
```

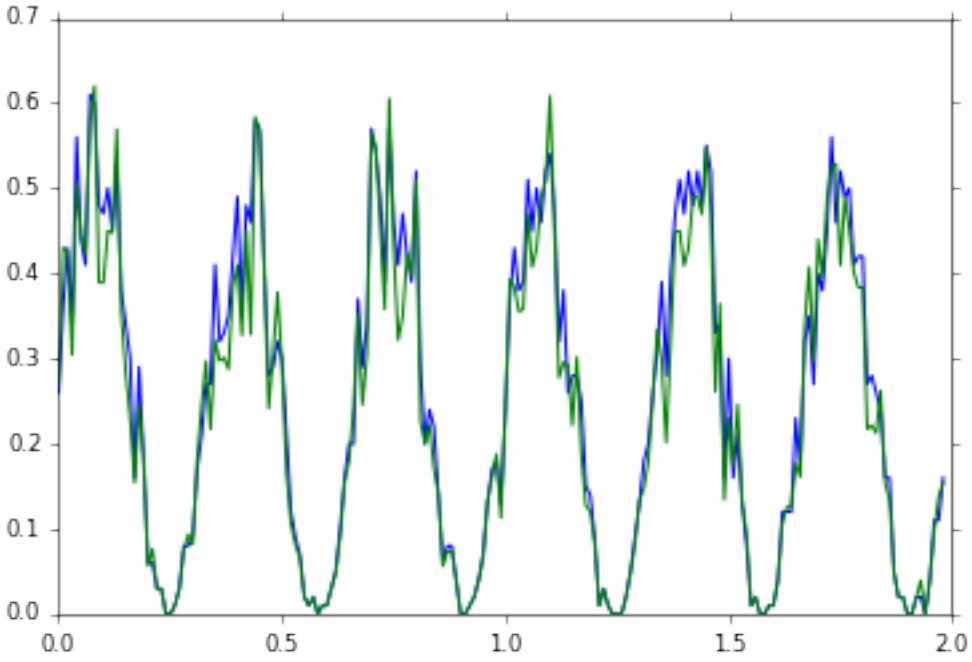


Compute the variance of the spike trains as a function of time. Does the noise look multiplicative or additive?

In [37]:
```
# in order to calculate the variance across trials, we need to generate histograms for each tr
# let's just do 100 trials. row_stack generates a 2D array which is useful for calculating sta
trials = np.row_stack([spike_hist([make_spikes(t, r)], bins) for i in range(100)])
# calculate mean and variance along dimension 0 (trials)
mean = np.mean(trials, 0)
var = np.var(trials, 0)

plt.plot(bins[:-1], mean, bins[:-1], var)
```

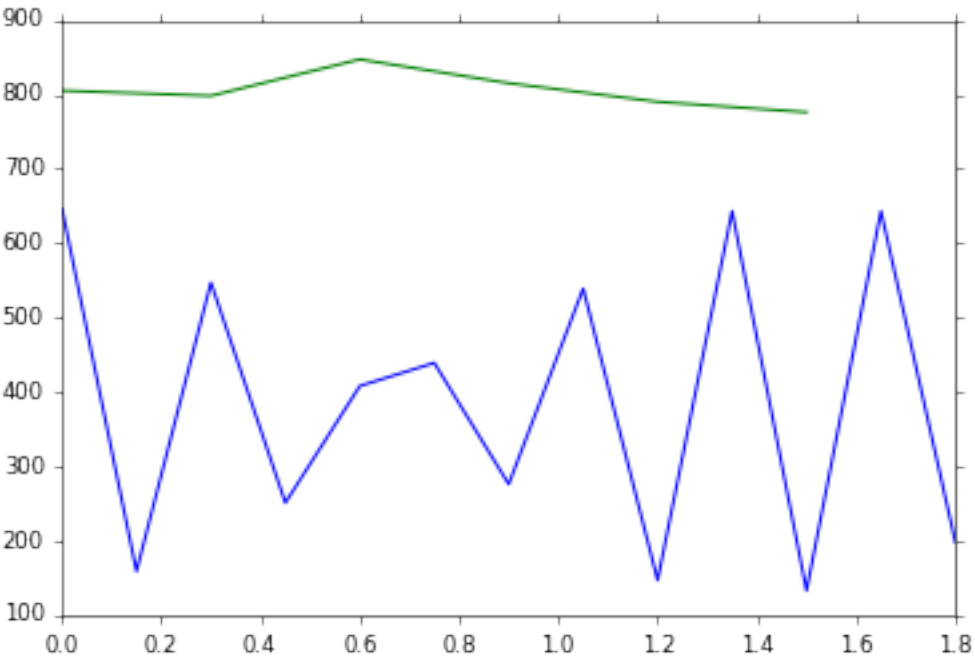
Out[37]:
```
[<matplotlib.lines.Line2D at 0x10feb7240>,
 <matplotlib.lines.Line2D at 0x10fb1e550>]
```



Change the binsize to 150 ms. What does the PSTH look like? How about 300 ms?

```
In [42]: spikes = [make_spikes(t, r) for i in range(100)]
         psth_1,bins_1 = spike_hist(spikes, np.arange(0, 2, 0.150))
         psth_2,bins_2 = spike_hist(spikes, np.arange(0, 2, 0.300))
         plt.plot(bins_1[:-1], psth_1, bins_2[:-1], psth_2)
```

```
Out[42]: [<matplotlib.lines.Line2D at 0x110be8eb8>,
          <matplotlib.lines.Line2D at 0x110c10e48>]
```



1.2 Question 2

Now you'll load some real spiking data, which is stored in 'toelis' format. Run the following cell. It will read the file into a list of numpy arrays; each array corresponds to a different trial.

```
In [45]: import toelis
         spikes = toelis.read(open("st82_2_4_1_st468_song_7.toe_lis"))[0]
```

Calculate some basic statistics. How many trials? What's the average number of spikes per trial? What's the standard deviation?

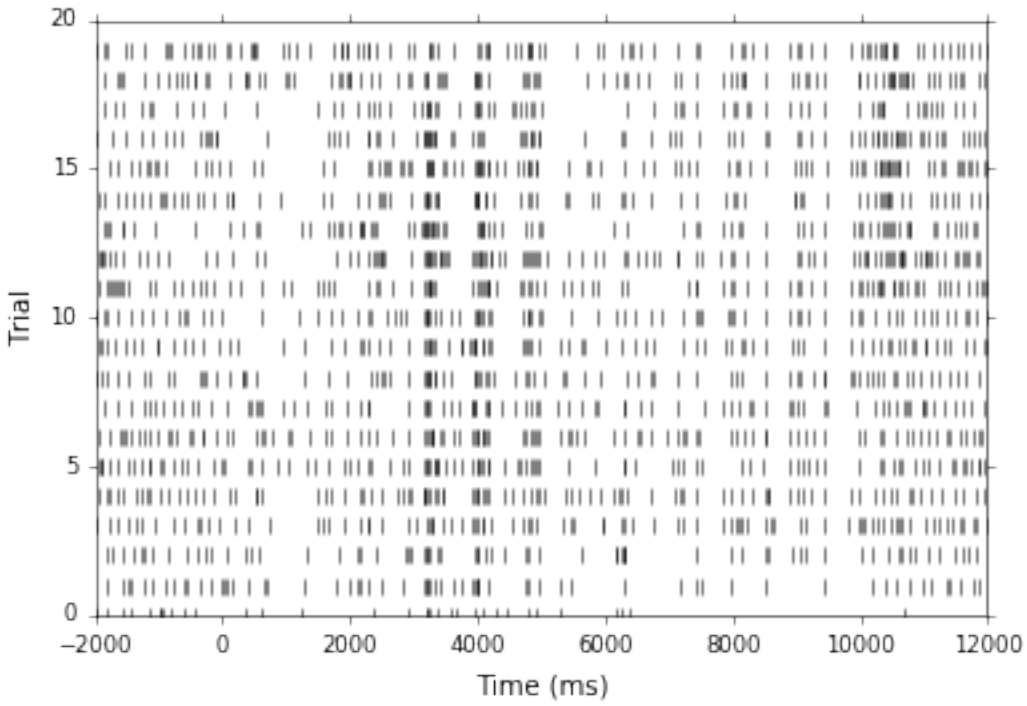
```
In [46]: print("Number of trials: %d" % len(spikes))
         counts = [len(x) for x in spikes]
         print("Average number of spikes/trial: %f" % np.mean(counts))
         print("Standard deviation: %f" % np.std(counts))
```

```
Number of trials: 20
Average number of spikes/trial: 88.150000
Standard deviation: 17.401940
```

Plot the trials as a raster. What patterns do you see in the response?

```
In [48]: for trial, events in enumerate(spikes):
             # generate a raster by making t,trial pairs
             y = np.ones(events.size) * trial
             plt.plot(events, y, "k|", hold=1)
         plt.xlabel("Time (ms)")
         plt.ylabel("Trial")
```

```
Out[48]: <matplotlib.text.Text at 0x110dfeda0>
```

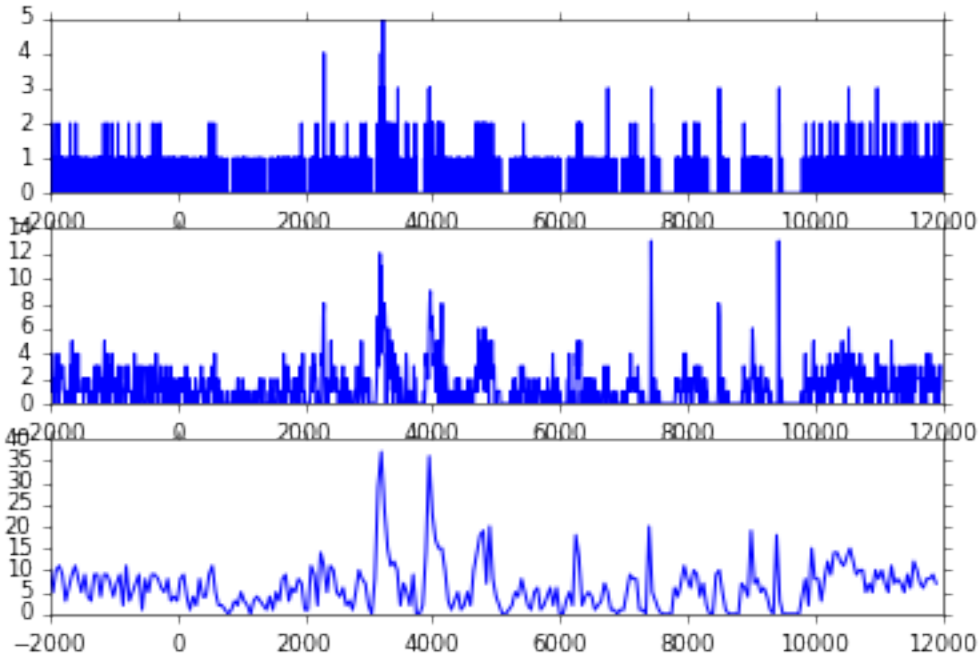


Plot the trials as a PSTH. Try adjusting the bin size between 1 ms and 50 ms. What bin size seems best for resolving the peaks of activity?

In [91]:
```
# NB time units are in ms
psth_1, bins_1 = spike_hist(spikes, np.arange(-2000, 12000, 1.))
psth_10, bins_10 = spike_hist(spikes, np.arange(-2000, 12000, 10))
psth_50, bins_50 = spike_hist(spikes, np.arange(-2000, 12000, 50))

plt.subplot(311), plt.plot(bins_1[:-1], psth_1)
plt.subplot(312), plt.plot(bins_10[:-1], psth_10)
plt.subplot(313), plt.plot(bins_50[:-1], psth_50)
```

Out[91]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x111903e48>,
 [<matplotlib.lines.Line2D at 0x111777da0>])
```



Consider the spikes that have negative times. These correspond to spontaneous activity. Calculate the intertrial interval histogram for these spikes. What is the mean and standard deviation? What's the coefficient of variation?

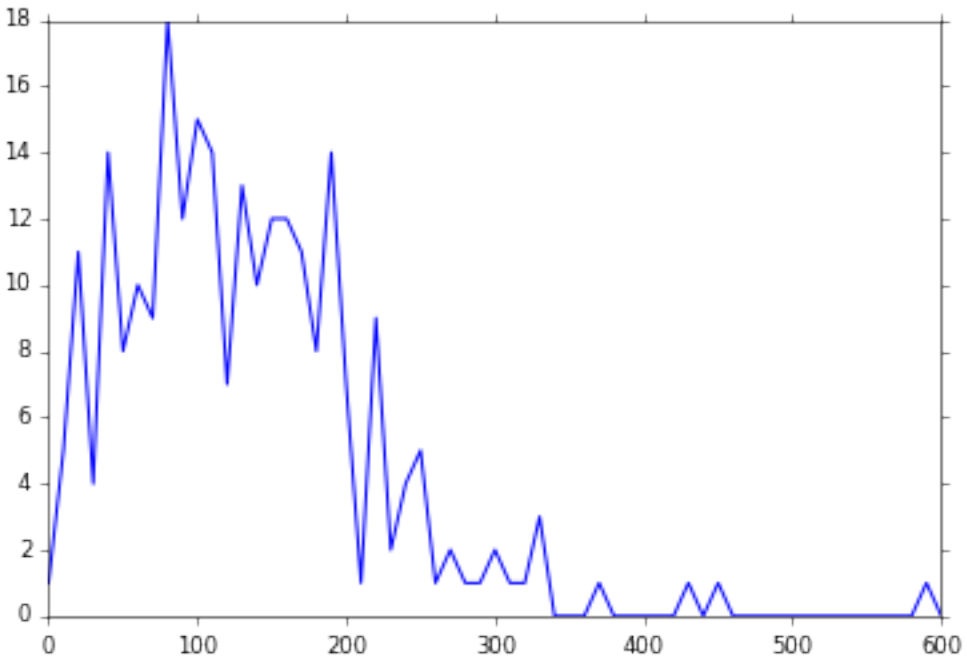
In [79]:
```
# extract spike times that are less than zero
# (x < 0 returns a boolean array indicating whether the value is less than 0; x[x < 0] uses th
# only those values)
prespikes = [x[x < 0] for x in spikes]

# calculate interspike intervals from difference in times in each trial, then collapse to sing
isis = np.concatenate([np.diff(x) for x in prespikes])
# for histogram, you have to experiment to find good bin values; 10 is not bad
hist, bins = np.histogram(isis, np.arange(0, isis.max(), 10))
```

```
        plt.plot(bins[:-1], hist)

        print("Mean ISI: %f" % isis.mean())
        print("STDEV: %f" % isis.std())
        print("CV: %f" % (isis.std() / isis.mean()))
```

```
Mean ISI: 139.645355
STDEV: 88.676071
CV: 0.635009
```



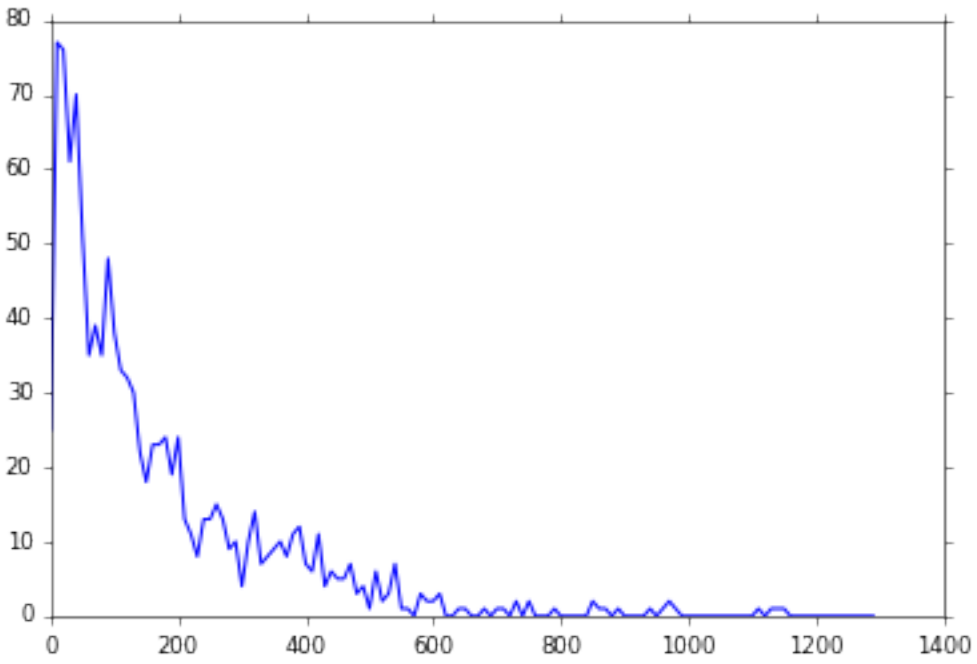
Now do the same for the spikes between 0 and 10000 ms. Which epoch is better described by a homogeneous Poisson process?

```
In [80]: postspikes = [x[(x > 0) & (x < 10000)] for x in spikes]

        isis = np.concatenate([np.diff(x) for x in postspikes])
        hist, bins = np.histogram(isis, np.arange(0, isis.max(), 10))
        plt.plot(bins[:-1], hist)

        print("Mean ISI: %f" % isis.mean())
        print("STDEV: %f" % isis.std())
        print("CV: %f" % (isis.std() / isis.mean()))
```

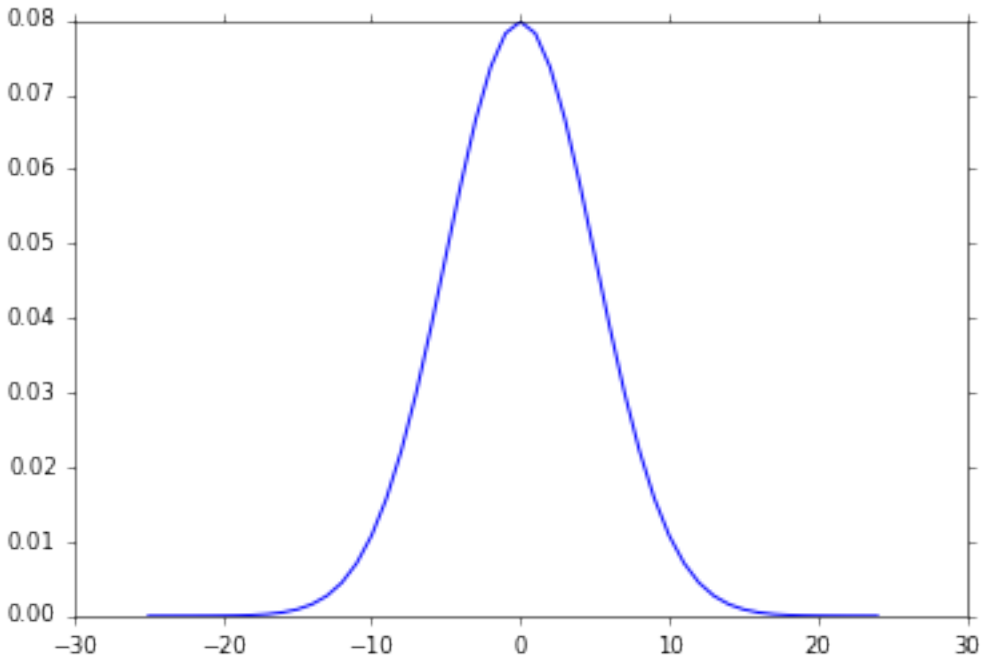
```
Mean ISI: 171.734753
STDEV: 178.355602
CV: 1.038553
```



Bonus question: calculate the PSTH for the data by convolving the spike train with a 5 ms Gaussian kernel

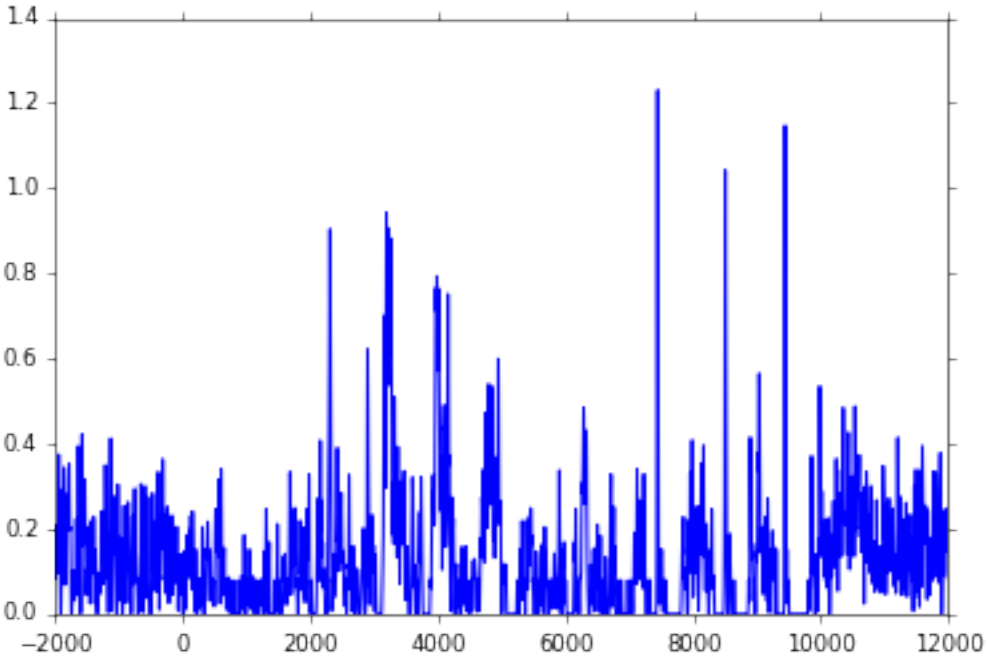
In [87]:
```
# although it's technically possible to convolve the spike train directly, making a psth as an
# define Gaussian on same grid as the 1 ms histogram
# window should be about 4-5 times the std dev
sigma = 5
tt = np.arange(-25,25,1.0)
w = 1/(sigma * np.sqrt(2 * np.pi)) * np.exp( - (tt)**2 / (2 * sigma**2))
plt.plot(tt, w)
print(w.sum())  # sum should be close to 1
```

```
0.999999377854
```



In [95]:
```
smoothed = np.convolve(psth_1, w, mode='same')
plt.plot(bins_1[:-1], smoothed)
```

Out[95]: [<matplotlib.lines.Line2D at 0x111b8ddd8>]



In [ ]: