

# Solution Set 4, Computational Neuroscience

Handed out: 02/24/2004

We hope that after working through the problems in this homework you have a little understanding about some of the mathematical models of spiking neurons. The models discussed here are some of the simplest possible and yet can explain some properties of real neurons. Infact, these models have been used extensively in neuronal modeling and with success on many occasions. Unfortunately, these simple models do not contain all the complicated ingredients of real neurons and so fall short in many respects. There exist highly sophisticated and complex models of neurons which are much more faithful to details then the ones mentioned here. Unfortunately, they are computationally intensive to simulate, leave alone to have insights for. We also hope that you have appreciated that tools from other disciplines can greatly enhance the understanding of a subject and not only provide a new perspective to view the known but also pave the way for solutions for the unsolved problems.

## 4.1 Poisson Processes

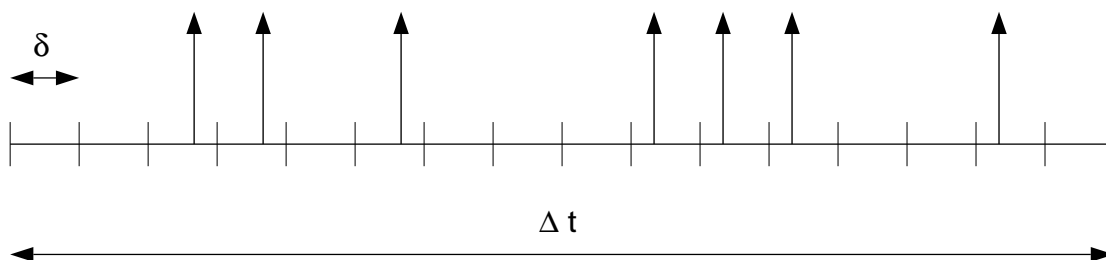


Figure 1: Discrete Model of a Poisson Process

- The time interval  $\Delta t$  is broken up into  $N$  equal bins of size  $\delta = \Delta t/N$ . The mean rate of spiking is  $\lambda$  and so the average number of spikes in the interval  $\Delta t$  is  $\lambda \Delta t$ . The probability of a spike occurring in each bin is  $p_\delta$ . The occurrence of a spike in a particular bin is independent of the event that a spike occurs in any other bin (by the very definition of a Poisson process). Therefore, we can consider the whole process as a collection of  $N$  identical repeated events,  $p_\delta$  being the same for each bin because both  $\lambda$  and the bin size are constant. Such events are called *Bernoulli trials*. Since the probability of an event with probability  $p$  occurring  $k$  times on  $N$  identical repetitions is given by the binomial distribution,

$$Pr[k \text{ out of } N] = \frac{N!}{(N-k)! k!} p^k (1-p)^{N-k}$$

Thus, the probability of obtaining exactly  $k$  spikes in the record is given by,

$$Pr[k \text{ spikes}] = \frac{N!}{(N-k)! k!} p_\delta^k (1-p_\delta)^{N-k}$$

The mean number of spikes can be obtained by taking the mean of the above distribution. Let  $\bar{n}$  denote the mean number of spikes,

$$\bar{n} = \sum_{k=0}^N k Pr[k \text{ spikes}]$$

It is easy to see that the mean  $\bar{n}$  of the above distribution equals  $p_\delta N$ . But since the mean number of spikes is also given by  $\lambda \Delta t$  we have on equating the two,

$$\begin{aligned} \bar{n} &= N p_\delta = \lambda \Delta t \\ \Rightarrow p_\delta &= \frac{\lambda \Delta t}{N} = \lambda \delta \qquad \text{Since } \delta = \frac{\Delta t}{N} \end{aligned}$$

It is clear that since  $p_\delta$  is a probability it should be less than 1. However, if  $\lambda$  is large and the bins are not small enough so that  $\lambda \delta > 1$  then  $p_\delta$  is not strictly well-defined. Intuitively, this means that if the firing rate is large then in order to use the above analogy we must choose the bin size small enough so that there is only one spike in each bin, because for larger bin sizes there is a possibility that there is more than one spike per bin and the above arguments don't hold anymore.

- Assume for the moment that we have discretized the process finely, we can then derive the inter-spike interval distribution extending the arguments above. Let us say that a spike occurred at an arbitrary time instant  $t_o$  and we count the number of bins before the next spike occurs. The probability that the inter-spike interval is  $k$  bins long is given by the product of the probability that there was no spike for the  $k$  bins after  $t_o$  and that a spike occurred in the  $(k + 1)^{\text{th}}$  bin (since the two events are independent by definition). Thus,

$$\begin{aligned} Pr[t_i = k\delta] &= Pr[\text{no spike in a bin}]^k Pr[\text{spike in bin } k + 1] \\ \Rightarrow Pr[t_i = k\delta] &= (1 - p_\delta)^k p_\delta \end{aligned}$$

If we keep decreasing the bin size  $\delta$  and correspondingly increasing  $N$ , we pass from the discrete to the continuous limit. Thus, the probability that the inter-spike interval  $t_i$  lies between  $T - \frac{1}{2}\delta$  and  $T + \frac{1}{2}\delta$  is given by,

$$Pr[T - \frac{1}{2}\delta < t_i < T + \frac{1}{2}\delta] = \left(1 - \frac{\lambda T}{N}\right)^N \frac{\lambda T}{N}$$

The probability density of inter-spike intervals can be obtained by dividing the above probability by  $\delta$  in the limit that  $\delta$  goes to zero, or  $N$  goes to infinity. Thus,

$$p(T) = \lambda \lim_{N \rightarrow \infty} \left(1 - \frac{\lambda T}{N}\right)^N$$

where  $p(T)$  denotes the probability density function of the inter-spike intervals. Using the result that  $e^x = \lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n$ , we have,

$$p(T) = \lambda e^{-\lambda T}$$

- Thus, we see that the inter-spike intervals in the continuous limit are exponentially distributed. Such a process is called a *Poisson process*. This distribution has the odd property that for both the discrete and the continuous case, the most likely inter-spike interval is zero. Since,

$$\begin{aligned} Pr[t_i = k\delta]_{max} &= Pr[t_i = 0] = p_\delta \\ p(T)_{max} &= p(0) = \lambda \end{aligned}$$

And the average interval for discrete case will be:

$$E\{k\} = \sum_{k=0}^{\infty} k Pr[t_i = k\delta]$$

$$\begin{aligned} \Rightarrow E\{k\} &= \sum_{k=0}^{\infty} k (1 - p_{\delta})^{k-1} p_{\delta} \\ \Rightarrow E\{k\} &= \frac{1}{\delta\lambda} \end{aligned}$$

And the average interval for continuous case will be:

$$\begin{aligned} E(t) &= \int_0^{\infty} t\lambda e^{-\lambda t} dt \\ \Rightarrow E(t) &= \frac{1}{\lambda} \end{aligned}$$

On the other hand, if one were to compute the probability of obtaining exactly  $k$  spikes,  $p_k$ , in an interval of length of  $T$  for a Poisson process with mean rate  $\lambda$  we get,

$$p_k = \frac{e^{-\lambda T} \lambda^k}{k!}$$

The above distribution is called the Poisson distribution (from which the process gets its name) and can be obtained from a binomial distribution in the limit  $\delta \rightarrow 0$ ,  $N \rightarrow \infty$ .

- The following MATLAB expression generates a spike train with 1000 bins (each bin is 1 msec long) from a constant current  $I=2$ . The maximum firing rate of the neuron is 100 Hz and so a  $p_{\delta} \leq \lambda_{max}\delta \leq 0.1$  is well-defined. The lines of code below compute  $p_{\delta}$  (which is constant in this case) and generate a spike in each bin with probability  $p_{\delta}$ .

```
>> I1 = 2*ones(1000,1); % Create input vector
>> S1 = (rand(1000,1) < .001 .* 100 ./ (1+exp(-I1))); % Spike train
```

- In order, to generate a spike train from a random input, use `gensig` to generate the current  $I_2$ .

```
>> I2 = 4*gensig(1000,1,10); % Create input vector
>> S2 = (rand(1000,1) < .001 .* 100 ./ (1+exp(-I2))); % Spike train
```

Representative plots of the spike train of the Poisson model in response to the above inputs are shown in Fig. 2 below.

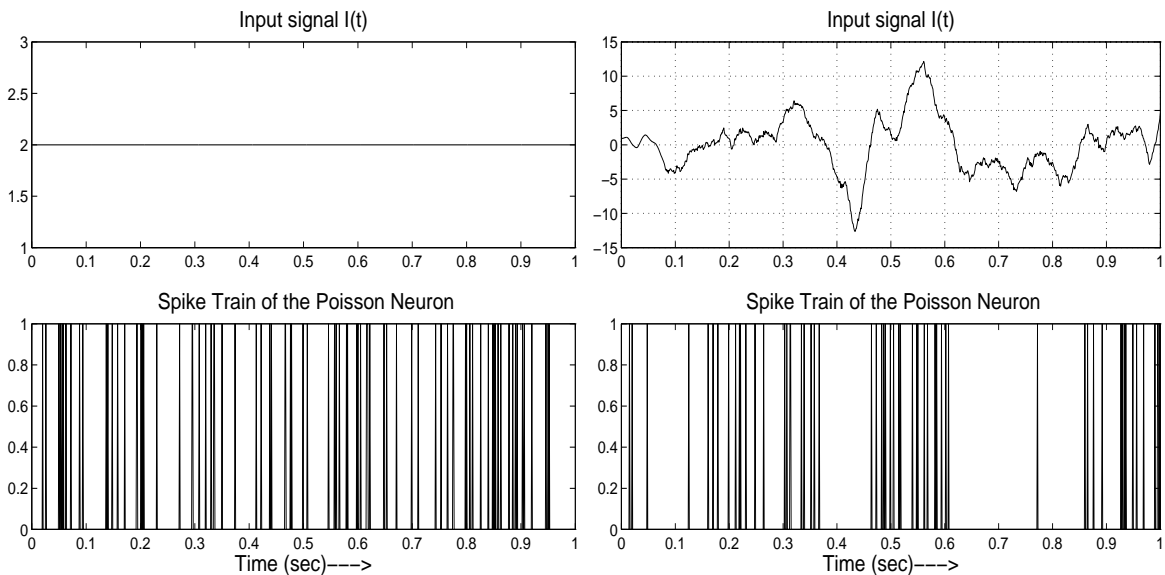


Figure 2: Poisson Spike Train in Response to Constant and Random Inputs

## 4.2 Integrate & Fire Neurons

Generating spike trains by simulating a Poisson process is a good first step as outputs of a multitude of real neurons can be very well approximated by Poisson processes. However, the Poisson process is only a phenomenological model of neural activity and so we turn to the Integrate & Fire model, a neuronal model with a more sound physical basis and biological plausibility.

- An Euler approximation is a quick and dirty method to simulate differential equations on digital computers. For some complicated differential equations more sophisticated methods (like Runge-Kutta) may be needed but for our purposes, Euler suffices. The update rule can be coded as,

```
>> dV = -V(j-1,n)/R/C + I(j-1,n)/C;
```

- The plots of the representative spike trains for the four different models are shown in Fig. 3. Each plot shows both the spike train and the membrane voltage on the same graph. For viewing convenience, the membrane voltage has been offset by 1 for the cases with constant inputs, and scaled for the cases with waveform inputs, using code such as,

```
>> I = 4*gensig(1000,1,10);
>> [S V] = iafsim(I,.8,.05,0);
>> plot(S);
>> hold on;
>> plot(V*0.15);
>> plot(0.15*ones(1000,1));
```

- The f-I curves for the four models are shown in Fig. 4. Some quick observations from the plots are as follows.

- ◊ For Model 1 (Integrate and Fire with no leak), the membrane voltage increases linearly (the slope being proportional to the current magnitude), reaches threshold, and is reset when a spike occurs. The F-I curve for it is linear since the time to reach threshold is inversely proportional to I. Also, the firing rate becomes infinite when the value of current goes to infinity. For the random input case observe that there a bunch of spikes whenever the input is positive and very few spikes during periods when the input is negative.

- ◊ For Model 2 (Leaky Integrate and Fire), because of the leak due to the resistance, the membrane voltage increases exponentially, saturating at a value given by  $IR$ . Thus, if the value of the input current is smaller than  $V_{th}/R$  the membrane voltage will never cross threshold and so no spike occurs. Thus, there is now a threshold value of current below which the firing rate of the neuron is zero. This value is called the *rheobase current*  $I_o$ . This model is more biologically plausible as some real neurons exhibit a current threshold. For current values above  $I_o$ , the firing rate grows approximately logarithmically with I. Thus, for Model 2, the f-I curve is non-linear.

- ◊ Addition of a refractory period as in Model 3 has two effects. Firstly, it lowers the value of the firing rate for a given current as there is more time between spikes and secondly, it sets a maximum limit on the firing rate  $f_{max} = 1/t_{ref}$  since the inter-spike interval cannot be lower than  $t_{ref}$ . So, now the f-I curve is not only non-linear, it also saturates at  $f_{max}$ . This is known to happen in real neurons which have an absolute refractory period of a few milliseconds. The f-I curve now starts to resemble our familiar sigmoidal function at least for current values above  $I_o$ .

- ◊ The f-I curve below  $I_o$  (which is zero for 2 and 3) is smoothed by the addition of noise in Model 4. This is a manifestation of a more general phenomenon known as *stochastic linearization* which implies that for some non-linear systems, addition of Gaussian noise actually helps in making the system response smoother and linear. Thus, the sharp cusp in the f-I curve below  $I_o$  is smoothed out and now the f-I curve begins to resemble a sigmoid very closely.

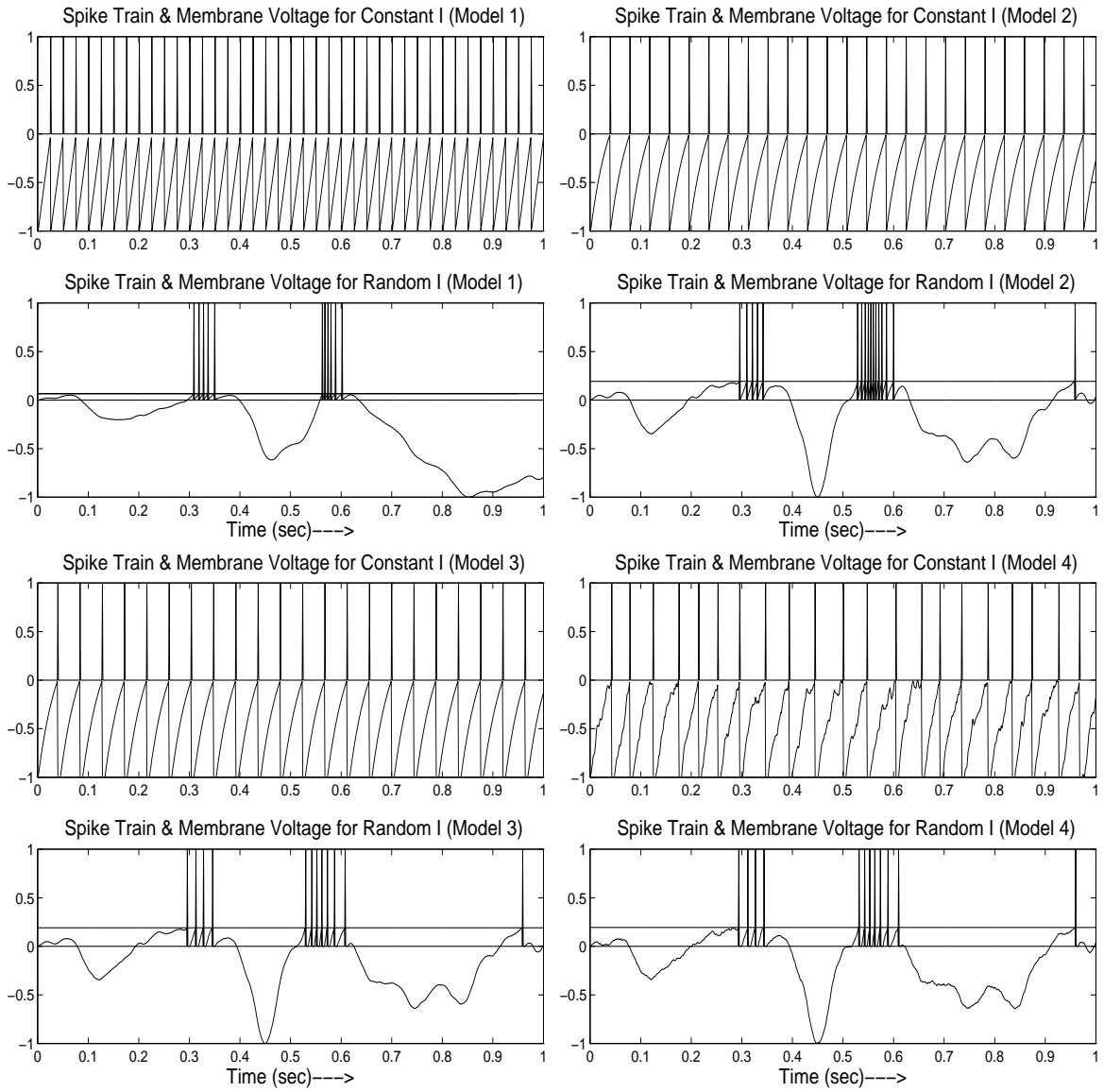


Figure 3: Spike Trains for Different I & F Models for Constant and Random Inputs. **Model 1:** No leak, **Model 2:** Leaky I & F, **Model 3:** Leaky I & F with Ref. Period, **Model 4:** Additive Noise Input. The solid line denotes the value of the threshold voltage  $V_{th}$ . The membrane voltage has been scaled and offset for convenience.

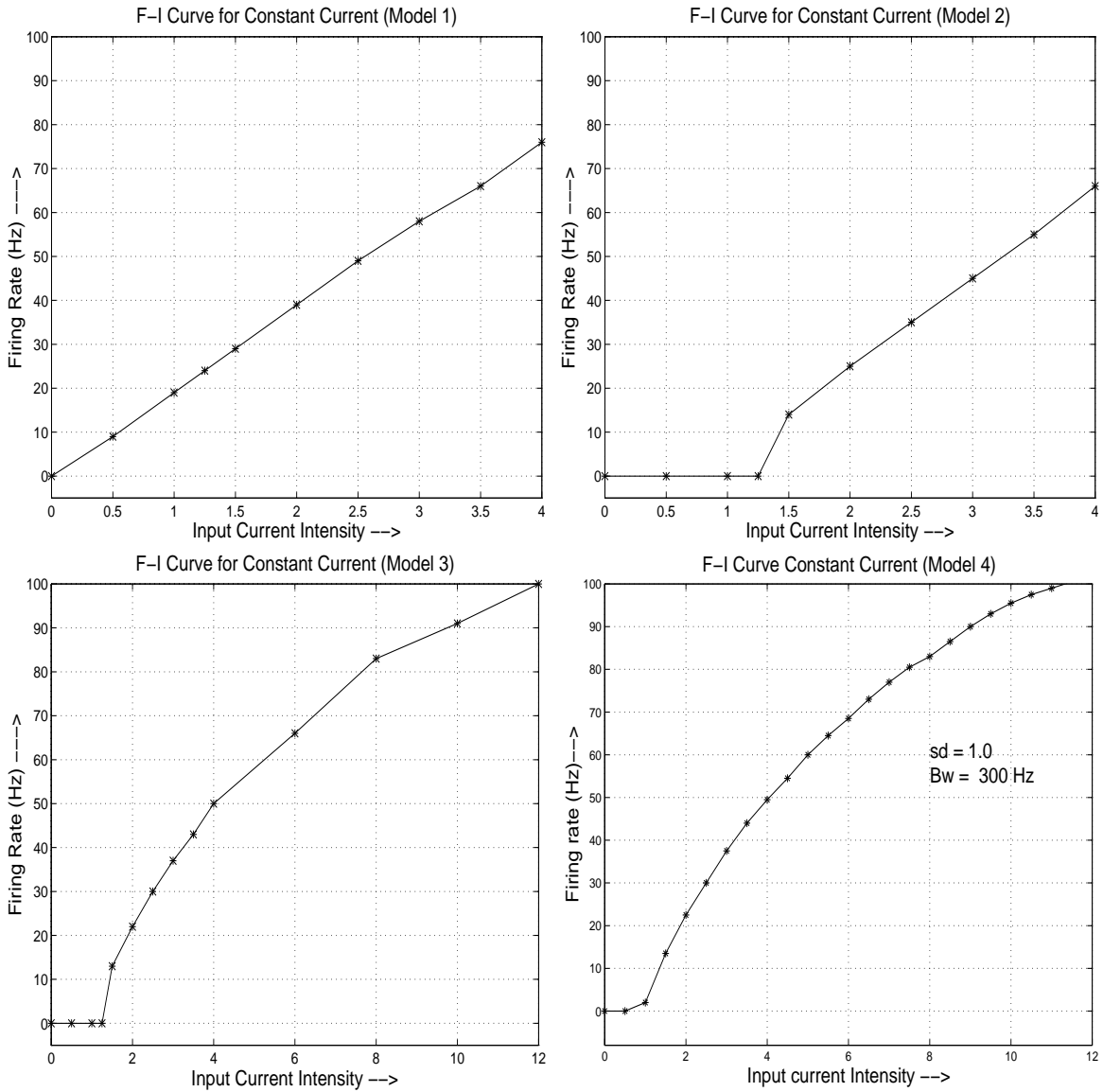


Figure 4: F-I Curves for Different I & F models

### 4.3 Linear Filtering: Edge Enhancement in Images

- The first problem asks you to design some kernels that do both smoothing and derivation of signals. The key is to notice that we can interchange the order of the **convolution** and **differentiation** operations. In particular:

$$\frac{d}{dx} [s(x) * g(x)] = \frac{d}{dx} \int_{-\infty}^{+\infty} s(u)g(x-u)du = \int_{-\infty}^{+\infty} \frac{d}{dx} [s(u)g(x-u)] du = s(x) * \frac{d}{dx}g(x)$$

This comes from the calculus fact that we can move differentiation inside an integral (as long as the integral is convergent which is true any time when the original convolution would have been defined). So to derive a kernel that does the equivalent of first smoothing (convolving with a gaussian) and then differentiation, we realize that we can first take the derivative of the gaussian

and then convolve with that. So the resulting kernel function is just the derivative of a gaussian (DOG):

$$\frac{d}{dx} \left[ \frac{1}{(2\pi)^{\frac{1}{2}}(\sigma^2)^{\frac{1}{2}}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \right] = \frac{-x}{(2\pi)^{\frac{1}{2}}(\sigma^2)^{\frac{3}{2}}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

which is shown on the left in the plot below.

- If we convolved this DOG function with a one dimensional signal, steep edges in the original signal would have either very positive (for upgoing edges) or very negative (for downgoing edges) values in the output of the convolution.
- We can use exactly the same argument to design a kernel which does the equivalent of smoothing and then taking the second derivative. In this case, we should just convolve with the second derivative of a gaussian (LOG-1D) which is given by the function:

$$\frac{d^2}{dx^2} \left[ \frac{1}{(2\pi)^{\frac{1}{2}}(\sigma^2)^{\frac{1}{2}}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \right] = \frac{x^2 - \sigma^2}{(2\pi)^{\frac{1}{2}}(\sigma^2)^{\frac{5}{2}}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

which is shown in the middle below.

- If we convolved this LOG-1D function with a one dimensional signal, steep edges in the original signal would have values *near zero* in the output of the convolution since the curvature changes in the middle of a smoothed edge. (The output is high on either side of the edge, but zero at the edge.)
- Finally, we can design a two dimensional kernel that smoothes and takes the Laplacian by simply taking the Laplacian of a gaussian (LOG). This works out to be (remember that the Laplacian is the sum of the second partials):

$$\nabla^2 \left[ \frac{1}{(2\pi)(\sigma^2)} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \right] = \frac{x^2 + y^2 - 2\sigma^2}{(2\pi)(\sigma^2)^3} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

This function is plotted on the right below.

- It is the same function (except for a trivial scale factor of  $1/\sqrt{2\pi\sigma^2}$ ) which would be obtained by replacing  $x^2$  with  $r^2 = x^2 + y^2$  in the one dimensional second derivative of a gaussian above. This substitution gives us the surface that would be obtained by rotating the one-dimensional version about the axis.
- If we convolved this “mexican-hat” function with a two dimensional signal, steep edges in the original signal would have values near zero in the output of the convolution.

