1. **Statistics of integrate-and-fire neurons**

   (a) Write a function that generates several trials of spike trains of a given duration from the leaky integrate-and-fire model (LIF) with white noise and constant input current I. The membrane potential V obeys the dynamics

$$\tau \frac{dV}{dt} = -V + RI + D\xi$$

   Whenever membrane potential crosses the firing threshold θ, a spike is emitted and $V(t)$ is reset to V_{reset}. τ is the membrane time constant (set to 10 ms), R is the input resistance (here 1 mV), $D\xi$ is additive Gaussian white noise of strength D, θ is the firing threshold (set to 10 mV), and V_{reset} is the reset potential (set to 0 mV).

   For integration of the differential equation discretize time with time step $\Delta t = 0.1$ ms ($V(t) \to V_i,\ t_i = i\Delta t$) and use forward Euler integration

$$\begin{aligned} \frac{dV}{dt} &\approx \frac{V_{i+1} - V_i}{\Delta t} \\ \Rightarrow \quad V_{i+1} &= V_i + \Delta t \frac{-V_i + RI_i + \sqrt{2D\Delta t} N_i}{\tau} \end{aligned}$$

   N_i are normally distributed random numbers (Gaussian with zero mean and unit variance).

Solution:

lifspikes.m

```
function spikes = lifspikes( trials, input, tmaxdt, D )
% Generate spike times of a leaky integrate-and-fire neuron
% trials: the number of trials to be generated
% input: the stimulus either as a single value or as a vector
% tmaxdt: in case of a single value stimulus the duration of a trial
%         in case of a vector as a stimulus the time step
% D: the strength of additive white noise

    tau = 0.01;
    if nargin < 4
        D = 1e0;
    end
    vreset = 0.0;
    vthresh = 10.0;
    dt = 1e-4;

    if length( input ) == 1
        input = input * ones( ceil( tmaxdt/dt ), 1 );
    else
        dt = tmaxdt;
    end
    spikes = cell( trials, 1 );
    for k=1:trials
        times = [];
        j = 1;
        v = vreset;
        noise = sqrt(2.0*D)*randn( length( input ), 1 )/sqrt(dt);
        for i=1:length( noise )
            v = v + ( - v + noise(i) + input(i))*dt/tau;
            if v >= vthresh
```

```
            v = vreset;
            times(j) = i*dt;
            j = j + 1;
        end
    end
    spikes{k} = times;
end
end
```

(b) Write a similar function for the perfect integrator (PIF):

$$\tau \frac{dV}{dt} = RI + D\xi$$

Solution:

pifspikes.m

```
function spikes = pifspikes( trials, input, tmaxdt, D )
% Generate spike times of a perfect integrate-and-fire neuron
% trials: the number of trials to be generated
% input: the stimulus either as a single value or as a vector
% tmaxdt: in case of a single value stimulus the duration of a trial
%         in case of a vector as a stimulus the time step
% D: the strength of additive white noise

    tau = 0.01;
    if nargin < 4
        D = 1e-1;
    end
    vreset = 0.0;
    vthresh = 10.0;
    dt = 1e-4;

    if length( input ) == 1
        input = input * ones( ceil( tmaxdt/dt ), 1 );
    else
        dt = tmaxdt;
    end
    spikes = cell( trials, 1 );
    for k=1:trials
        times = [];
        j = 1;
        v = vreset;
        noise = sqrt(2.0*D)*randn( length( input ), 1 )/sqrt(dt);
        for i=1:length( noise )
            v = v + ( noise(i) + input(i))*dt/tau;
            if v >= vthresh
                v = vreset;
                times(j) = i*dt;
                j = j + 1;
            end
```

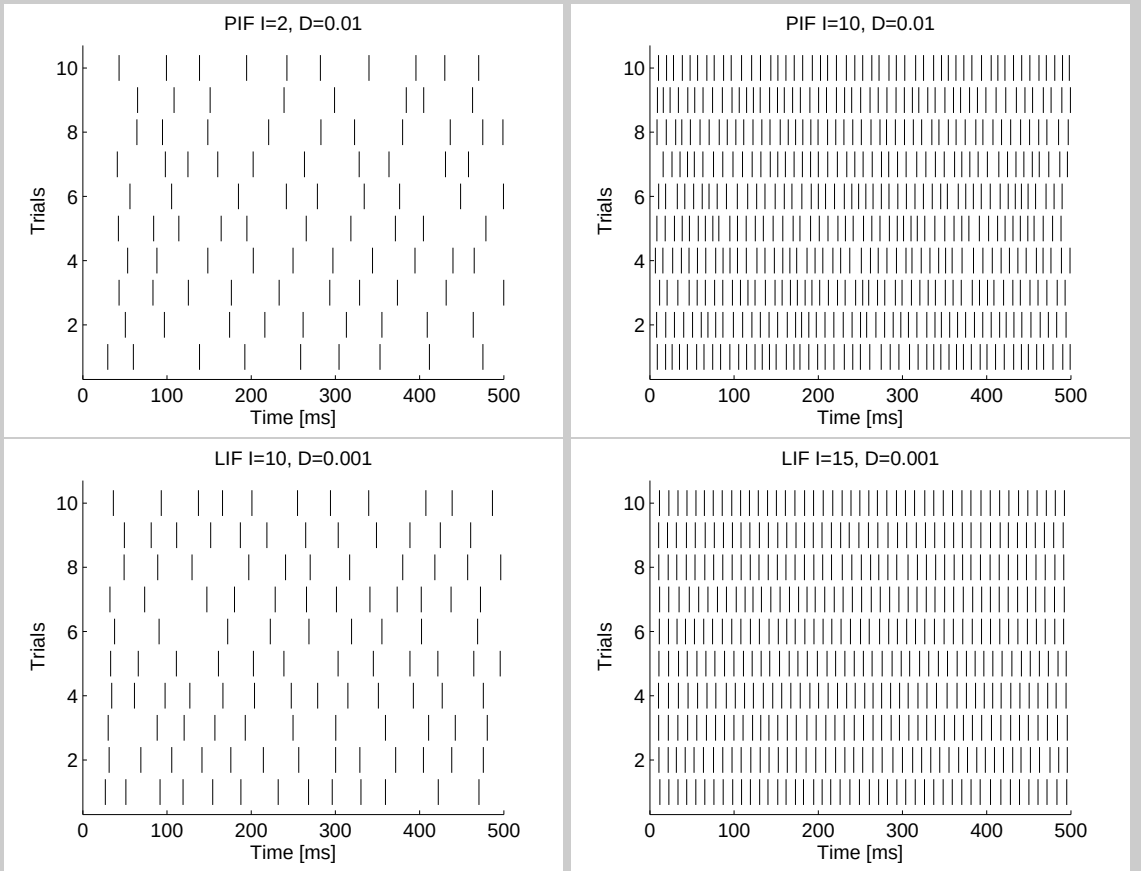
```
        end
        spikes{k} = times;
    end
end
```

(c) Generate a few trials of the two models for two different inputs that result in qualitatively different spike trains and display them in a raster plot. Decide for a noise strength (good values to try are 0.001, 0.01, 0.1, 1).

Solution:

.

```
spikes = pifspikes( 10, 1.0, 0.5, 0.01 );
%spikes = pifspikes( 10, 10.0, 0.5, 0.01 );
%spikes = lifspikes( 10, 11.0, 0.5, 0.001 );
%spikes = lifspikes( 10, 15.0, 0.5, 0.001 );
spikeraster( spikes )
```



(d) The inverse Gaussian describes the interspike interval distribution of a PIF driven with white noise:

$$p(T) = \frac{1}{\sqrt{4\pi D T^3}} \exp\left[-\frac{(T-\langle T\rangle)^2}{4DT\langle T\rangle^2}\right]$$

where $\langle T \rangle$ is the mean interspike interval and

$$D = \frac{\langle (T - \langle T \rangle)^2 \rangle}{2 \langle T \rangle^3}$$

is the diffusion coefficient (variance of the interspike intervals T divided by two times the mean cubed). Show in two plots how this distribution depends on $\langle T \rangle$ and D.

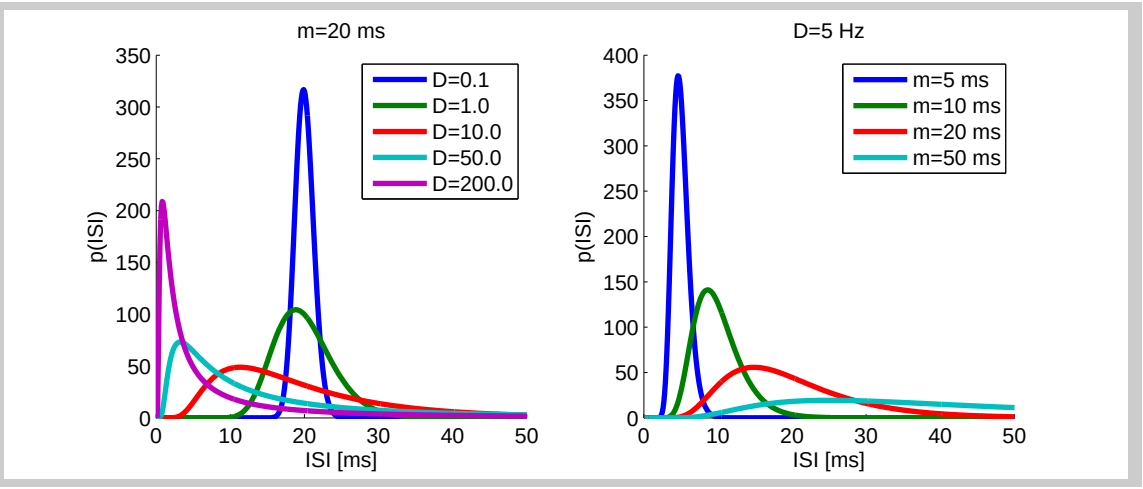
Solution:

inversegauss.m

```
function y = inversegauss( x, m, d )
% returns the inverse Gauss density with mean isi m and diffusion
% coefficent d
    y = exp(-(x-m).^2./(4.0*d.*x.*m.^2.0))./sqrt(4.0*pi*d.*x.^3.0);
end
```

inversegaussplot.m

```
f = figure;
subplot( 1, 2, 1 );
dx=0.0001;
x = dx:dx:0.5;
hold all
m = 0.02;
for d = [ 0.1 1 10 50 200 ]
    plot( 1000.0*x, inversegauss( x, m, d ), 'LineWidth', 3, 'DisplayName',
         sprintf( 'D=%.1f', d ) );
end
title( sprintf( 'm=%g ms', 1000.0*m ) )
xlim( [ 0 50 ] );
xlabel( 'ISI [ms]' );
ylabel( 'p(ISI)' );
legend( '-DynamicLegend' );
hold off;

subplot( 1, 2, 2 );
hold all;
d = 5.0;
for m = [ 0.005 0.01 0.02 0.05 ]
    plot( 1000.0*x, inversegauss( x, m, d ), 'LineWidth', 3, 'DisplayName',
         sprintf( 'm=%g ms', 1000.0*m ) );
end
title( sprintf( 'D=%g Hz', d ) )
xlim( [ 0 50 ] )
xlabel( 'ISI [ms]' );
ylabel( 'p(ISI)' );
legend( '-DynamicLegend' )
hold off
```



(e) Extent your function plotting an interspike interval histogram to also report the diffusion coefficient D.

Solution:

.

```
...
% annotation:
misi = mean( isis );
sdisi = std( isis );
disi = sdisi^2.0/2.0/misi^3;
text( 0.6, 0.7, sprintf( 'mean=%.1f ms', 1000.0*misi ), 'Units', '
    normalized' )
text( 0.6, 0.6, sprintf( 'std=%.1f ms', 1000.0*sdisi ), 'Units', '
    normalized' )
text( 0.6, 0.5, sprintf( 'CV=%.2f', sdisi/misi ), 'Units', 'normalized' )
text( 0.6, 0.4, sprintf( 'D=%.1f Hz', disi ), 'Units', 'normalized' )
...
```

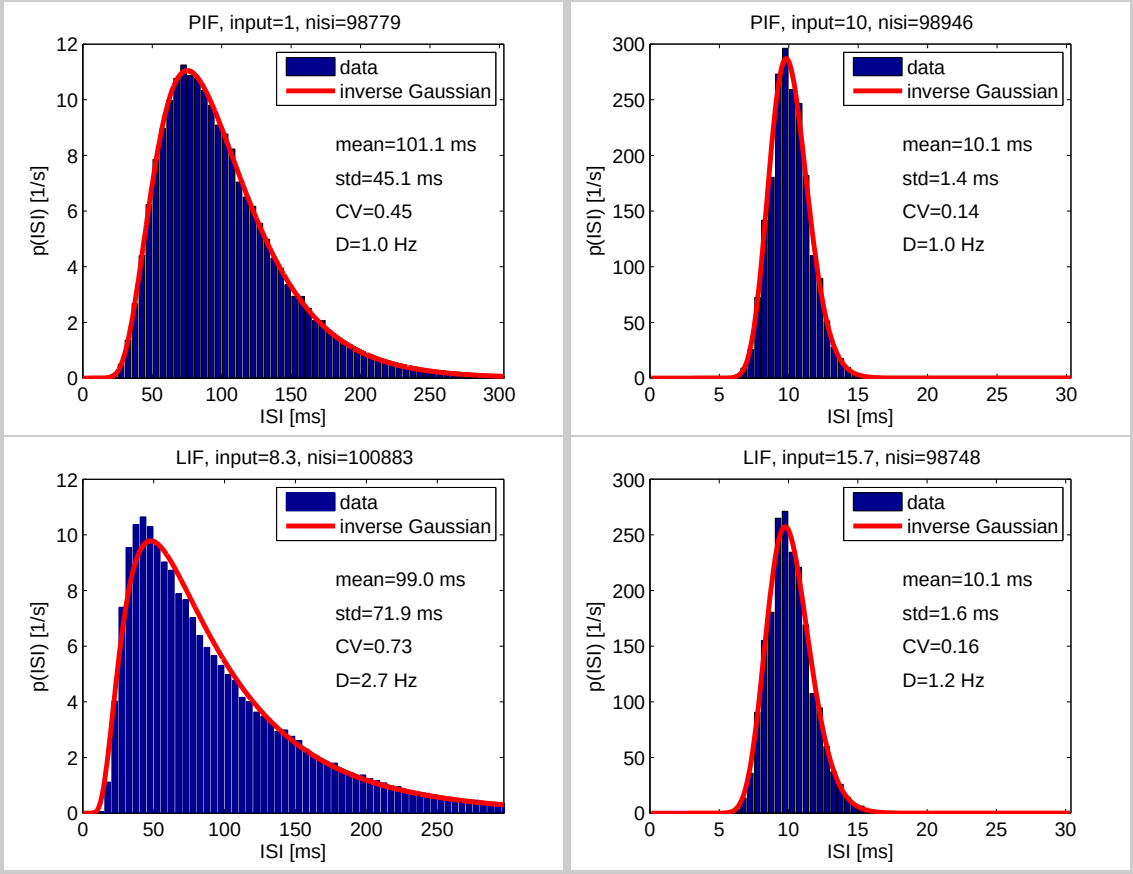
(f) Compare intersike interval histograms obtained from the LIF and PIF models with the inverse Gaussian.

Solution:

lifisih.m

```
input = 65.0; % lifadapt 100Hz
%input = 8.0;  % lifadapt 10Hz
%input = 15.7;
%input = 8.3;
spikes = lifadaptspikes( 10, input, 100.0, 1e-1, 0.1, 5.0 );
isivec = isis( spikes );
% histogram
f = figure( 1 );
isihist( isivec, 10e-4 );
hold on
% theoretical density:
misi = mean( isivec );
```

```
disi = var( isivec )/2.0/misi^3;
xmax = 3.0*misi;
x = 0:0.0001:xmax;
plot( 1000.0*x, inversegauss( x, misi, disi ), 'r', 'LineWidth', 3 );
% plot details:
title( sprintf( 'LIF OU, input=%g, nisi=%d', input, length( isivec ) ) )
xlim( [ 0.0 1000.0*xmax ] )
legend( 'data', 'inverse Gaussian' )
hold off
```



(g) Plot the firing rate (inverse mean interspike interval), mean interspike interval, the corresponding standard deviation, CV, and diffusion coefficient as a function of the input to the LIF and the PIF with noise strength set to 0.01.

Solution:

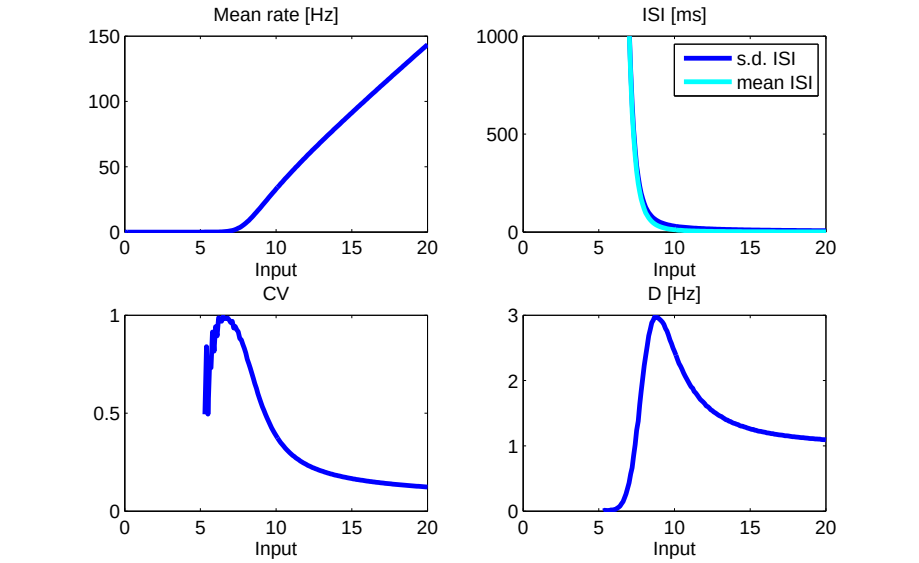
lifisistats.m

```
inputs = 0:0.1:20; % lif
inputs = 0:0.1:10; % pif
avisi = [];
sdisi = [];
cvisi = [];
dcisi = [];

```

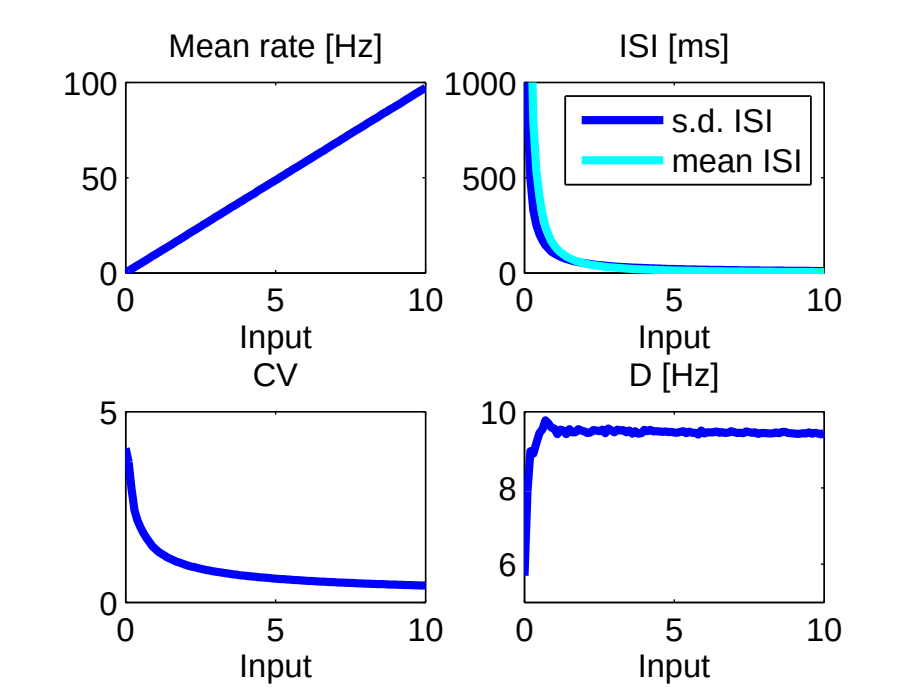
```
for input = inputs
    input
    % spikes = lifspikes( 100, input, 100.0, 1e-2 );
    spikes = pifspikes( 100, input, 100.0, 1e-1 );
    isivec = isis( spikes );
    if length( isivec ) <= 1
        av = Inf;
        sd = NaN;
        cv = NaN;
        dc = NaN;
    else
        av = mean( isivec );
        sd = std( isivec );
        if av > 0.0
            cv = sd/av;
            dc = sd^2.0/2.0/av^3;
        else
            cv = NaN;
            dc = NaN;
        end
    end
    avisi = [ avisi av ];
    sdisi = [ sdisi sd ];
    cvisi = [ cvisi cv ];
    dcisi = [ dcisi dc ];
end

f = figure;
subplot( 2, 2, 1 );
plot( inputs, 1.0./avisi, '-b', 'LineWidth', 3 );
xlabel( 'Input' );
xlim( [ inputs(1) inputs(end) ] )
title( 'Mean rate [Hz]' );

subplot( 2, 2, 2 );
plot( inputs, 1000.0*avisi, '-b', 'LineWidth', 3 );
hold on;
plot( inputs, 1000.0*sdisi, '-c', 'LineWidth', 3 );
hold off;
xlabel( 'Input' );
xlim( [ inputs(1) inputs(end) ] )
ylim( [ 0 1000 ] )
title( 'ISI [ms]' );
legend( 's.d. ISI', 'mean ISI' );

subplot( 2, 2, 3 );
plot( inputs, cvisi, '-b', 'LineWidth', 3 );
xlabel( 'Input' );
xlim( [ inputs(1) inputs(end) ] )
title( 'CV' );

subplot( 2, 2, 4 );
```

```
60 plot( inputs, dcisi, '-b', 'LineWidth', 3 );
61 xlabel( 'Input' );
62 xlim( [ inputs(1) inputs(end) ] )
63 title( 'D [Hz]' );
```

Leaky integrate-and-fire:



Perfect integrate-and-fire:



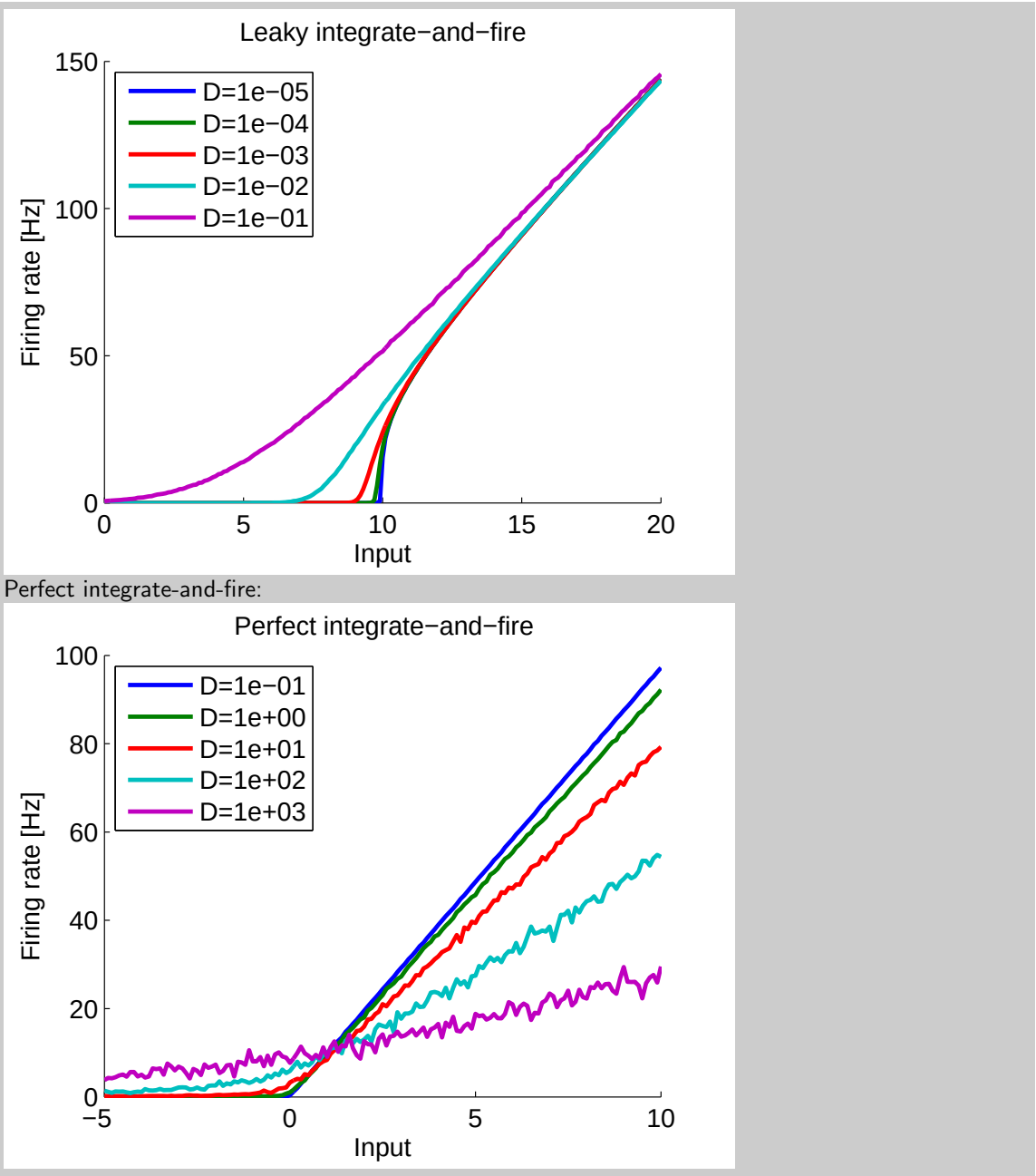
(h) Plot the firing rate as a function of input of the LIF and the PIF for various values of the noise strength.

Solution:

lifficurves.m

```
% lif:
noises = [ 1e-5 1e-4 1e-3 1e-2 1e-1 ];
inputs = 0:0.1:20;
duration = 50.0;
% pif:
noises = [ 1e-1 1 1e1 1e2 1e3 ];
inputs = -5:0.1:10;
duration = 100.0;

f = figure;
hold all;
for noise = noises
    fprintf( 'noise=%.0e\n', noise );
    rates = [];
    for input = inputs
        % spikes = lifspikes( 10, input, duration, noise );
        spikes = pifspikes( 50, input, duration, noise );
        nspikes = 0;
        for k = 1:length( spikes )
            nspikes = nspikes + length( spikes{k} );
        end
        rate = nspikes/duration/length( spikes );
        %fprintf( 'I=%g N=%d rate=%g\n', input, length( spikes ), rate )
        rates = [ rates rate ];
    end
    plot( inputs, rates, 'LineWidth', 2, 'DisplayName', sprintf( 'D=%.0e',
        noise ) );
end
xlabel( 'Input' );
xlim( [ inputs(1) inputs(end) ] )
ylabel( 'Firing rate [Hz]' );
%title( 'Leaky integrate-and-fire' )
title( 'Perfect integrate-and-fire' )
legend( '-DynamicLegend', 'Location', 'NorthWest' )
hold off
```

Leaky integrate-and-fire:



(i) Use the functions for computing serial correlations, count statistics and fano factors to explore the statistics of the integrate-and-fire models!