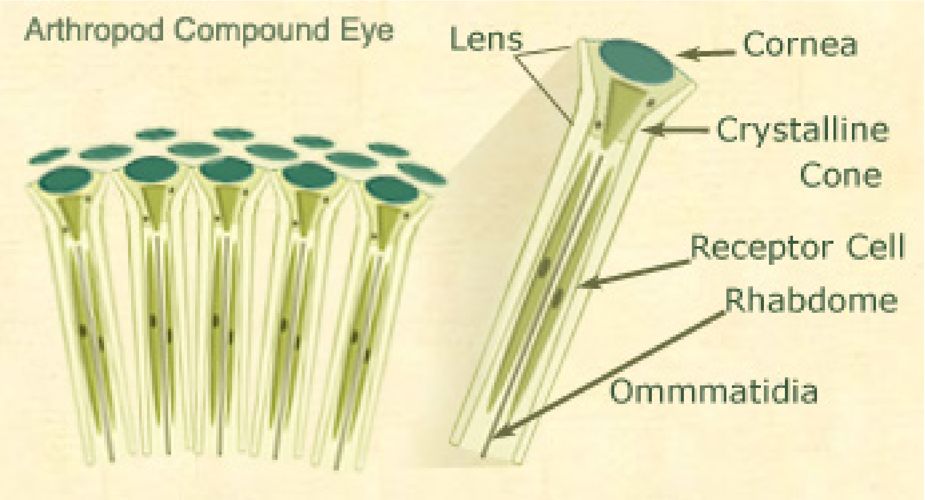
5 Model of a simple vision system

Related to Lytton's Chapter 8: *Our Friend the Limulus*.

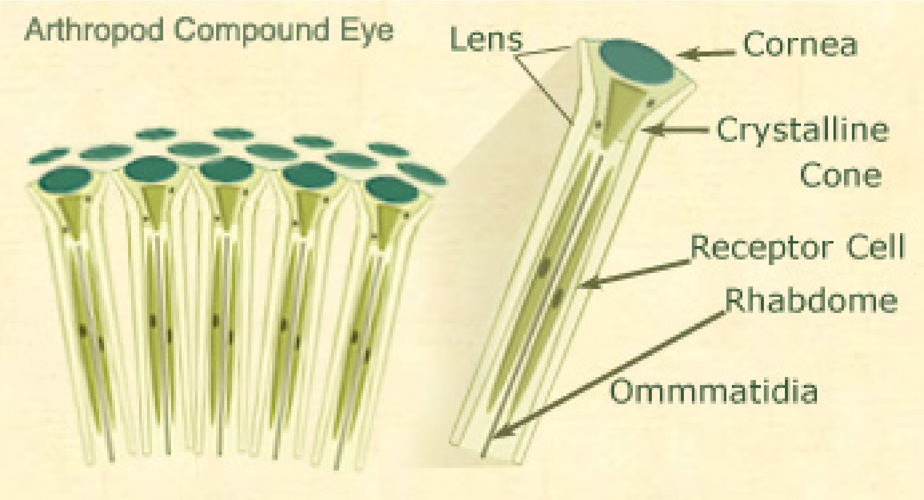
5.1 The Limulus

- The simple vision system is based on that of a famous **limulus** (horseshoe crab) that has been extensively studied due to its simplicity.
- The central aspect of a simple neural model of the limulus vision is based on the concept of **lateral inhibition**, which is instrumental in image enhancement through sharpening edges in the image.
- Limulus provided researchers with a near-perfect model for studying vision.
- First, it is large, easy to find and easy to handle.
- It possesses both simple and compound eyes.
- The compound eyes are relatively large and the optic nerve, which connects the eyes to the brain is not only enormous, up to four inches long, but also lies just below the carapace.
- For early researchers who wanted to eavesdrop on the signals traveling between eyes and brain one could scarcely design a better animal!
- Read about it in: `http://www.mbl.edu/animals/Limulus`. Short intro follows:



- Limulus (an arthropod) possess a type of eye known as a compound eye.
- In all of the arthropods, the exoskeloton (the “shell”) contributes the lens portion of the eye. As a result, the focus of the eye is fixed as it is part of the outer skeleton of the animal.
- The compound eye is made of smaller, simple eye units, called ommatidia.

- Each ommatidia is composed of a cornea, which is formed from the outer exoskeleton (the “shell”).
- This cornea acts as a lens to focus light into the eye.
- As a result, the focus of the eye is fixed as it is part of the outer skeleton of the animal.
- Following this is an element called the “crystalline cone” which serves as a second lens.
- It is produced by adjacent cells, usually four in number.
- The cone tapers to a receptor unit called a retinula which focuses the light into a translucent cylinder called the rhabdome.



- The rhabdome is surrounded by light sensitive, or retinular cells.
- There are generally seven similar retinular cells and one eccentric cell.
- It is the inward-facing portions of these cells in fact, which form the rhabdome.

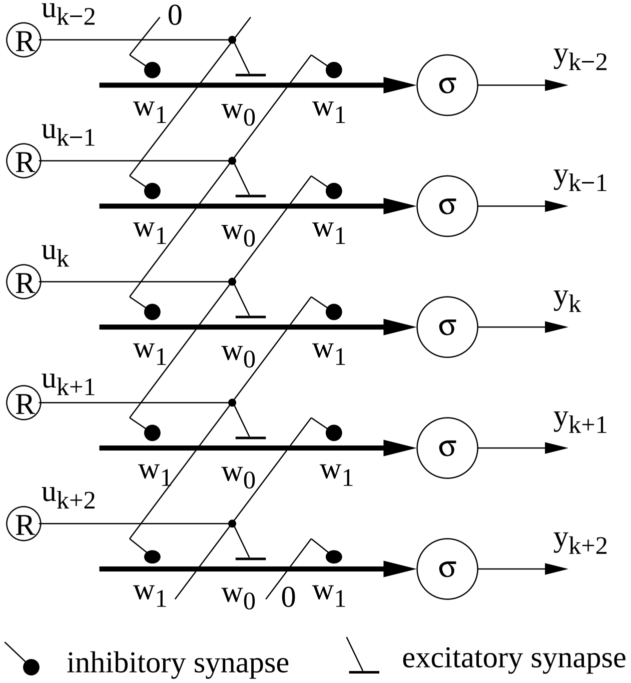
- The rhabdome is the common area where light is transmitted to the reticular cells.
- Each of these cells is connected to an axon and since each ommatidia consists of seven or eight reticular cells,
- there are this number of axons which form a bundle from each ommatidia.
- These axons then form connections with other nerves to create an optic ganglion which passes the visual signal to the brain.
- Each ommatidia passes information about a single point source of light.
- The total image formed therefore is a sum of the ommatidia fired and can be thought of as a series of dots or pixels.

5.2 Lateral inhibition

- Lateral inhibition is a process that animals, including humans, use to better distinguish borders.
- When you look at the ocean horizon the ocean appears darker at the horizon, at the boundary between sea and sky.
- This apparent difference in light intensity is not actually there but is created by our visual receptors and is known as lateral inhibition.
- This process increases contrast and results in a sharpening of vision.
- What this means is that the signals coming from the outside are actually altered before being sent to the brain so that what we see isn't necessarily there.
- The way this works is as follows.
- Impulses originate in the eccentric cell when the cell is stimulated by light.
- This signal is transmitted through the axon then to the optic nerve to the brain.
- The ability of an ommatidia to discharge impulses is related to the amount of light that neighboring ommatidia are receiving.
- Hartline found that if one ommatidia is receiving bright light and a neighbor is receiving dim light, the first ommatidia will inhibit the signal from it's neighbor.
- The result is that the dimmer signal gets even dimmer and the result is an increased difference between the two which the eye would perceive as an increase in contrast.

5.3 One-dimensional model of the limulus vision

5.3.1 1-D feedforward model of lateral inhibition



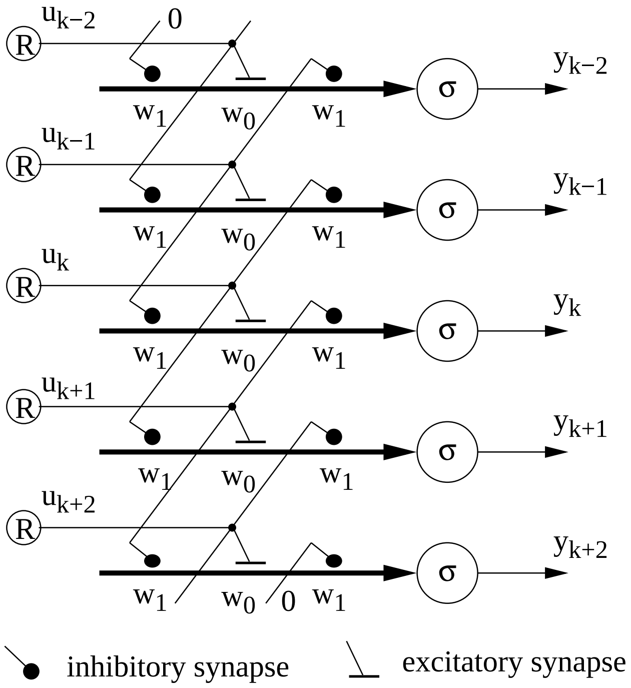
- Assume that **visual intensity signals** arrive from a number of **visual receptors** marked **R** to synapses of neurons.
- Assume for simplicity one neuron per the receptor.
- The signal u_k from the k-th receptor **R** is connected to

  an **excitatory synapse** of the k-th neuron and at the same time it is connected to

  the **inhibitory synapses** of the neighbouring neurons, $k-1$ and $k+1$.
- Such a sidewise connection is known as the **lateral inhibition**.

- In the example you can identify five neurons, the central one with the efferent (output) signal y_k.
- Along the dendrite of each neuron we have three synapses, the central excitatory synapse with the weight $w_0 > 0$ accepts the afferent signal u_k from the receptor.
- This signal goes also to two neighbouring neurons inhibiting them. The weights of the inhibiting synapses are negative, $w_1 < 0$
- Note that all neurons have synapses with identical weights, $[w_1,\ w_0,\ w_1]$

5.3.2 Maths of lateral inhibition. A Mexican hat mask



- Assuming for simplicity that $\sigma = 1$, and
- re-numbering the units for $k = 3$
- the output (efferent) signals can be calculated as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} w_1 & w_0 & w_1 & 0 & 0 & 0 & 0 \\ 0 & w_1 & w_0 & w_1 & 0 & 0 & 0 \\ 0 & 0 & w_1 & w_0 & w_1 & 0 & 0 \\ 0 & 0 & 0 & w_1 & w_0 & w_1 & 0 \\ 0 & 0 & 0 & 0 & w_1 & w_0 & w_1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ 0 \end{bmatrix} \tag{5.1}$$

- Note that each output signal is a linear combination of input signal (three signals, in this example)

- Introducing the mask

$$\mathbf{h} = \begin{bmatrix} w_1 & w_0 & w_1 \end{bmatrix}$$

- The output signal can be calculated as:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} \mathbf{h} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{h} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{h} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{h} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{h} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ 0 \end{bmatrix} \tag{5.2}$$

- Now we can see that the same mask is sliding along the input signals forming the respective linear combination.

- Assuming that we have now p input and output signals, the above equation can now be generalized into a form:

$$\mathbf{y} = \langle \mathbf{h} \rangle_p \cdot \begin{bmatrix} \mathbf{0} \\ \mathbf{u} \\ \mathbf{0} \end{bmatrix} \tag{5.3}$$

  where $\langle \mathbf{h} \rangle_p$ as in eqn (5.2) is known as a **convolution matrix**.

- Alternatively, eqn (5.3) can be written in terms of the convolution function as

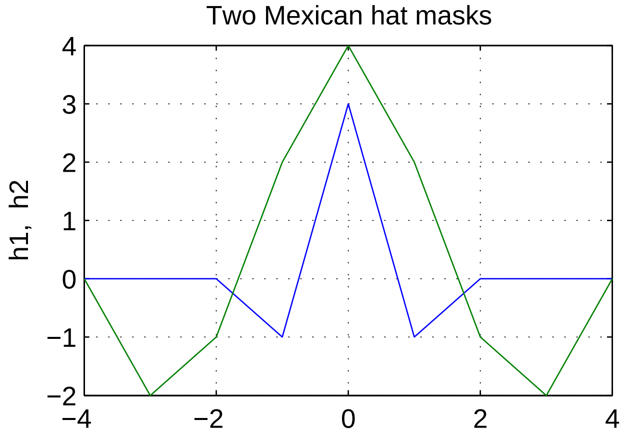
$$\mathbf{y} = \text{conv}(\mathbf{u}, \mathbf{h}) \tag{5.4}$$

  where the convolution function is specified in terms of the above convolution matrix.

The **lateral inhibition masks** aka Maexican hat masks are selected so that they:

- are symmetrical
- balance of inhibitory and excitatory behaviour

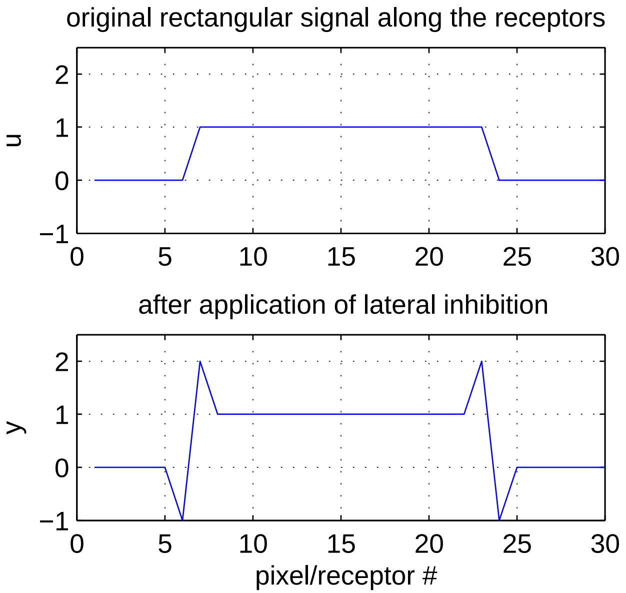
Two simple 1-D Mexican hat masks:



$$\mathbf{h}_1 = [\ -1\ \ 3\ \ -1\ ]$$
$$\mathbf{h}_2 = [-2\ \ -1\ \ 2\ \ 4\ \ 2\ \ -1\ \ -2\ ]$$

Sharpening of the edges of 1-D visual signal by the mask

$$\mathbf{h}_1 = [\ -1\ \ 3\ \ -1\ ]$$



From the plots we can note that the Mexican hat mask, inhibiting the neighbouring neurons, amplifies a change of intensity along the receptors which results in **enhancement of the image edges**.

5.3.3 Designing a good Mexican hat mask

In the processing we might like to make sure that the level of the signal is maintained, that is, if the afferent signal is constant one, then it stays constant one after processing with the mask. We will say that such a circuit/filter has a **unity dc gain**. The acronym ‘dc’ stands for ‘direct-current’ and is a shortcut for a constant level signal.

To ensure the unity dc gain we just need a mask in which **sum of its all coefficients is unity**.

For a n-element mask the dc gain should satisfy the following condition:

$$y_c = \sum_{i=1}^{n} h_i = 1 \tag{5.5}$$

Another aspect that we would like to control in the output signal is the amount of negative/positive **overshoot**.

For a general mask the overshoot y_m is a sum of all negative coefficients. If we number coefficients of the symmetric mask in the following way:

$$h = \left[\begin{array}{ccccccccc} h_n & \ldots & h_1 & h_{p1} & \ldots & h_{pk} & h_1 & \ldots & h_n \end{array} \right]$$

then the overshoot can be calculated as follows:

$$y_m = |\sum_{i=1}^{n} h_i| \tag{5.6}$$

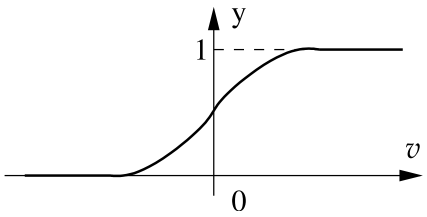
and the dc gain (constant level) as

$$y_c = \sum_{i=1}^{k} h_{pi} - 2y_m \tag{5.7}$$

5.3.4 Sigmoidal activation function

One, a bit inconvenient, effect of the processing with the Mexican hat mask is the change of the range of the numbers: they are expanded from the input range range, say $\{0,\ 1\}$, to the output range $\{-y_m,\ \ 1+y_m\}$ both in the negative and positive directions due to overshoot.
One way of dealing with this problem is to pass signals through the saturating function, $y = \sigma(v)$ in order to put the signals back into the range $\{0,\ 1\}$.
Typically such a saturating function is defined as a **sigmoidal function**:

$$y = \sigma(v) = \frac{1}{2}(\tanh(0.5v) + 1) = \frac{1}{1 + e^{-v}}$$



Let us see what happens if we add such a saturating function to our previous script in the following way:

```
MnMx = [min(y) max(y)]
y1 = 0.5*(tanh(0.5*y)+1) ;
MnMx1 = [min(y1) max(y1)]
```

The resulting minimum and maximum values of the efferent signals are:

```
 MnMx = [-1  2]=  [h(1)  1-h(1)]
 MnMx1 =  0.2689    0.8808
```

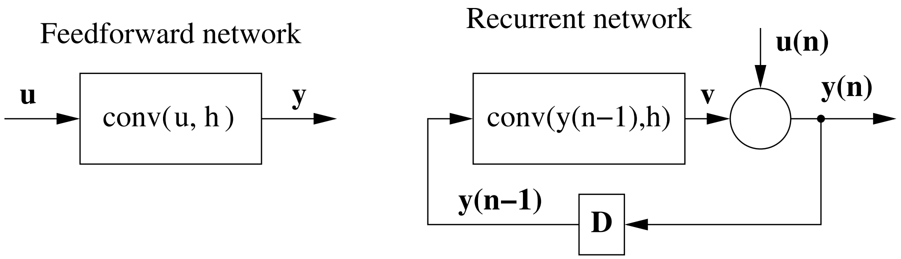
which means that the sigmoidal function squashed the range of numbers so that they are well in the range between 0 and 1, which might be desired.

5.3.5 Feedforward and recurrent networks

The **feedforward network** implemented above transforms input (afferent) signals into the output (efferent) signals in one processing step:

$$\mathbf{y} = W \cdot \mathbf{u}$$

In the next model of the limulus vision, we will add the feedback loops connecting the output signals to the input synapses. Networks with feedback are also referred to as **recurrent networks.** The following block diagrams clarify the concepts.



$$\mathbf{y} = W \cdot \mathbf{u} \qquad\qquad \mathbf{y}(n) = W \cdot \mathbf{y}(n-1) + \mathbf{u}(n)$$

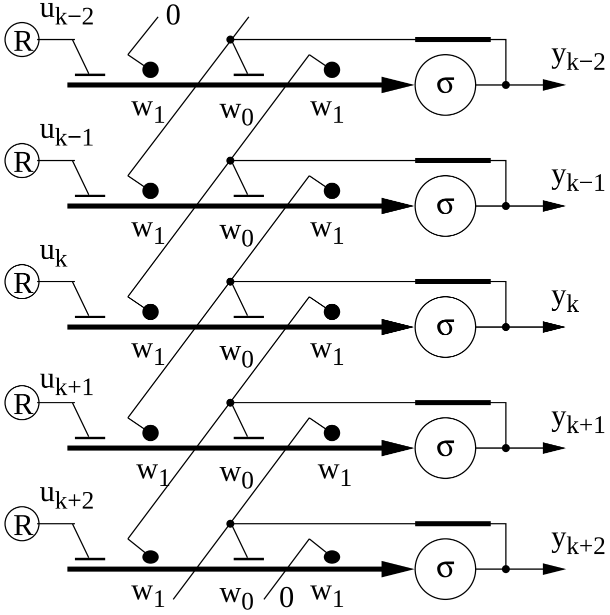
The big circle at the output of the convolution block represents a summation $\mathbf{y} = \mathbf{v} + \mathbf{u}$. More precisely, the recurrent processing can be described by the following equation

$$\mathbf{y}(0) = \mathbf{u}(0), \quad \mathbf{y}(n) = W \cdot \mathbf{y}(n-1) + \mathbf{u}(n) = \text{conv}(\mathbf{y}(n-1), \mathbf{h}) + \mathbf{u}(n) \;, \quad \text{for} \quad n = 0, 1, \ldots \tag{5.8}$$

The variable n represents time and is the number of the current processing step.
The **unit-delay** block "**D**" in the recurrent network block-diagram performs the delay of output signals by one time step, that is, it forms $\mathbf{y}(n-1)$ form $\mathbf{y}(n)$.

5.3.6 1-D recurrent model of limulus vision

A simple one-dimensional recurrent model of limulus vision:



$$k = 3, \quad \mathbf{h} = [w_1 \; w_0 \; w_1]$$

$$\begin{bmatrix} y_1(n) \\ y_2(n) \\ y_3(n) \\ y_4(n) \\ y_5(n) \end{bmatrix} = \begin{bmatrix} w_0 & w_1 & 0 & 0 & 0 \\ w_1 & w_0 & w_1 & 0 & 0 \\ 0 & w_1 & w_0 & w_1 & 0 \\ 0 & 0 & w_1 & w_0 & w_1 \\ 0 & 0 & 0 & w_1 & w_0 \end{bmatrix} \cdot \begin{bmatrix} y_1(n-1) \\ y_2(n-1) \\ y_3(n-1) \\ y_4(n-1) \\ y_5(n-1) \end{bmatrix} + \begin{bmatrix} u_1(n) \\ u_2(n) \\ u_3(n) \\ u_4(n) \\ u_5(n) \end{bmatrix} \tag{5.9}$$

$$\mathbf{y}(n) = W \cdot \mathbf{y}(n-1) + \mathbf{u}(n)$$

Comparing with a feedforward network we note that in the recurrent network:

- the vision receptors are connected only to one synapse (signals u_k), whereas the laterally inhibiting signals are formed from the fed back output signals $y_k(n-1)$.

  This is a more realistic model of the limulus vision.

- at each time step n the network calculates the values of all (say, $m = 5$) efferent signals $y_k(n)$ from the previous value of these signals $y_k(n-1)$ and the afferent signals $u_k(n)$.

- The time delay is indicated in the diagram by the thickened feedback line.

For the feedworward processing, we were able to design a mask according to eqns (5.5) and (5.6) to achieve a required behaviour, namely, to maintain the constant level y_c and to control the overshoot y_m.
Behaviour of recurrent networks as described by eqn (5.8) is significantly more complex, therefore designing the prescribed behaviour of a recurrent network is also, in general, difficult.
In our case, however, with a simple 1-D Mexican hat mask we can again be in a full control of the network behaviour.

Referring to eqn (5.8) we note that now, since we are adding the afferent signals $\mathbf{u}$, we should modify the mask $\mathbf{h}$ so that the sum of its coefficients should be zero, that is,

$$h_c = \sum_{i=1}^{n} h_i = 0 \;\; ; \;\; \mathbf{h} = [h_1 \ldots h_n] \tag{5.10}$$

where h_i represents all coefficients of the mask. This will ensure maintaining the **constant level** through recurrent addition of $\mathbf{u}$.

To work out **overshoots** is a bit more complicated. They depend on the sum of negative coefficients. One way of dealing with the complexity is to introduce one parameter, g, called **feedback gain** and multiply the Mexican hat mask by g:

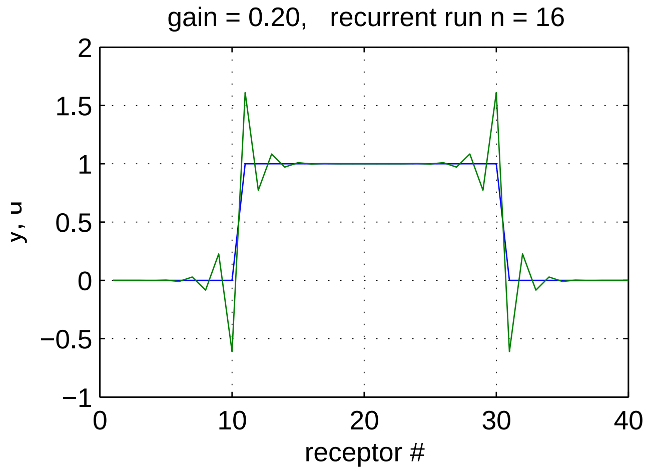
$$\mathbf{h} = g \cdot [h_1 \; h_2 \ldots h_p \ldots h_2 \; h_1]$$

Now, we will find out that the amount of overshoot and its shape will depend, first of all, on the value of the feedback gain, g.
The MATLAB script that implements the recurrent network can have the following form:

```
m = 40 ;
u = zeros(m,1) ;
```

```
u((1:m/2)+m/4) = ones(m/2,1) ; % rectangular signal
g = 0.2 ;           % a gain parameter
h = g*[-1 2 -1] ;   % a mask
sum(h)              % must be zero
W = convmtx(h, m); % convolution matrix
W = W(:, 2:end-1); % weight matrix
figure(3)
y = zeros(m, 1) ;   % initial value of the afferent signals
x = (1:m)' ;        % used in plotting
nn = 16 ;           % number of recurrent runs
for n = 1:nn
   y = W*y + u ;    % recurrent network
   plot(x, [u y] ), grid on
   title(sprintf('gain = %1.2f,   recurrent run n = %d', g,n))
   % axis([0 m -1 2])
   pause(2)         % the loop goes every 2 secs
end
```

The final form of the efferent signals is given in the following figure:



Note that the there are spatial (along the receptors) oscillations around the edges of the signal.

- I would like you to observe how the gain g influences the behaviour of the recurrent network. Record the maximum value of overshoots after $n = 20$ recurent runs for $g = 0.24, 0.28, 0.30, 0.32$
- Note that there is a critical value of the gain (for a given mask) beyond which the network becomes **unstable**. Estimate this value.

  Unstable, means that no steady-state value has been reached, even after many iterations. This is equivalent (almost) to saying that signals grows without limitations.

  In our case the critical value of the gain can be selected as the one for which oscillations on both signal edges meet in the centre.
- Modify the maximum value of the mask by $\pm 10\%$ and $\pm 20\%$. Report the results of such modifications.

5.4 Recurrent 2-D model

We now jump straight into 2-D recurrent networks that are real model of the limulus vision.

5.4.1 2-D structure of the limulus vision

The structure of such a network is a bit more complicated to represent, but the principle is the same as in Figure **??**, namely, that the central neuron inhibits neighbouring neurons and is also inhibited by the neighbours. This idea is illustrated in Figure 5–1.

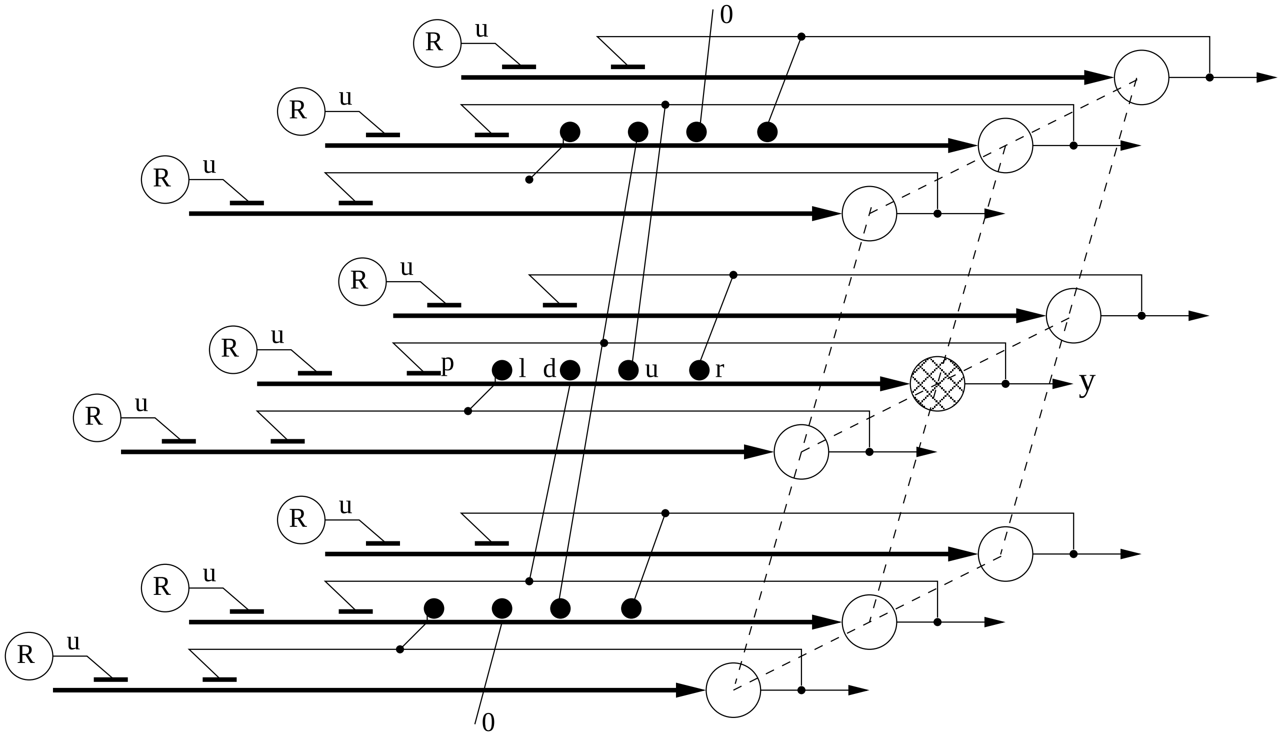


Figure 5–1: 2-D structure of the limulus vision system

In Figure 5–1 the central neuron marked y and checkered is inhibited by its four neighbours, from left, right, up and down. Relevant synapses and equivalent elements of the 2-D mask $\mathbf{h}$ are marked, l, d, u, r, respectively. The mask element related to the local excitatory feedback signal is marked with p.

Conversely, every neuron, inhibits its four neighbours. For the checkered neuron only connections to neurons straight above and below are shown, for clarity.

The afferent signals from visual receptors are marked u in Figure 5–1, and they are organized into a matrix U, representing the image to be processed.

5.4.2 2-D masks and convolution

The simplest **2-D mask** describing interconnections between neurons as in Figure 5–1 can now be represented by the following matrix H:

$$H = g \cdot \begin{bmatrix} 0 & h_u & 0 \\ h_l & h_p & h_r \\ 0 & h_d & 0 \end{bmatrix} \;,\quad h_u, h_l, h_r, h_d < 0 \;,\quad h_p > 0 \tag{5.11}$$

where g is a feedback gain.
Following arguments from the previous section, in order to maintain the constant level of signals during recurrent runs, the the sum of all coefficients of the mask H must be zero:

$$h_u + h_l + h_r + h_d + h_p = 0 \tag{5.12}$$

Such a mask is applied to each element of processed image in a way similar to a 1-D case. For the simple mask as in eqn (5.11) an efferent signal y is formed as a linear combination of the mask elements and relevant image pixels (receptor signals):

$$y = h_p \cdot y + h_u \cdot y_u + h_l \cdot y_l + h_r \cdot y_r + h_d \cdot y_d + u$$

However complicated it sounds, the good news is that there exists a **2-D convolution** that takes an image, say U and a matrix mask, H, and creates the output image, say Y sliding the mask over every pixel of the input image and forming the necessary sum of products of mask elements and image pixels.
Mathematically we just write:

$$Y = \text{conv2}(U, H) \tag{5.13}$$

We will use modification of this expression in a recurrent loop in the next section.

5.4.3 2-D recurrent network operations

A 2-D recurrent visual network operates as described in eqn (5.8) with a suitable change from vectors to matrices:

$$Y(0) = U(0), \quad Y(n) = \text{conv2}(Y(n-1), H) + U(n)\ , \quad \text{for} \quad n = 0, 1, \ldots \tag{5.14}$$

We are now ready to process the simple test image presented in Figure **??**. Let us start with a bit more complicated Mexican hat mask of the following form:

$$H = g \cdot \begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix} \tag{5.15}$$

where gain g will be selected with stability of the processing in mind.

Before we start our next MATLAB exercise it might be a good idea to save the current script and open a new one named, say `myprac2d.m`. It will be still possible to copy and paste from the previous script if we need that.

In the new script we can type in the commands to create a mask as in eqn (5.15):

```
%  myprac2d.m
 clear, % close all
 g = 1/12
 HH = [-1 -2 -1 ; -2 12 -2 ; -1 -2 -1]
 % visualization of the mask
 H1 = zeros(5, 5) ; % adding a ring of zeros around the mask
 H1(2:4,2:4) = g*HH
 figure(1)
 surf(H1)
```

You should see a rather low resolution 2-D Mexican hat.

With a mask established, we can create a test image. We can copy the relevant commands from the previous script:

```
%  test image
 m = 100 ;
 x = 1:m ;
 [X Y] = meshgrid(x,x) ;
 U = ((X - m/2).^2 + (Y - m/2).^2) < (0.125*m^2) ;
 figure(1)
 imagesc(U), axis image
 grid on, colormap(1-gray(2))
```

This will reproduce the familiar cylinder. Finally, we can run our network with the selected image and mask:

```
 % recurrent 2-D network
  Y = zeros(size(U)) ;  % initial value of the afferent signals
  g = 0.04 ;   % adjustable feedback gain
  H = g*HH ;   % mask adjusted with gain
  nn = 10 ;    % number of recurrent runs
  figure(2)
```

```
for n = 1:nn
  Y = conv2(Y, H, 'same') + U ;   % 2-D recurrent network
  subplot(2,2,1)
  imagesc(Y), axis image   % top view of the cylinder
  colormap(1-gray(256))
  text(60,-6, sprintf('gain = %1.2f,   recurrent run n = %d', g,n))
  subplot(2,2,2)
  surf(Y), grid on, view(-30, 20)     % perspective view
  axis([0 100 0 100 -1 2])
  subplot(2,2,3)
  plot(Y(50,1:30))      % cross-section along the row 50
  grid on, axis square
  xlabel('receptor # in row 50')
  pause(2)   % the loop goes every 2 secs
end
```

The result, after 10 iterations, with a given gain is shown in Figure 5–2.
As in a 1-D case the jump in intensity results in sharpening the edge and creates an overshoot, or oscillations, depending on the value of the feedback gain.

- Test the influence of the gain on the resulting oscillations.
- Design your own 5×5 mask and repeat recurrent run on the test image. Describe the results.

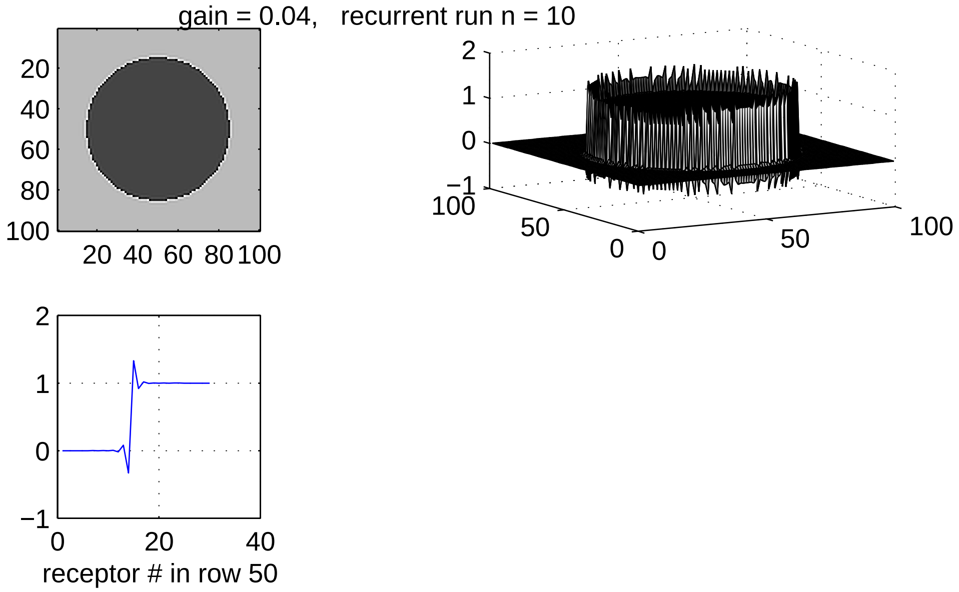


Figure 5–2: Processing a test image by a recurrent neural network

5.5 Images in MATLAB

Finally, we apply our 2-D vision network to processing a real image. For simplicity we use first with **gray scale** (monochrome) image. Such an image in MATLAB is just an $r \times c$ matrix, each element of the matrix (known as a **pixel**) representing intensity of the pixel.

Externally, images are stored in multiplicity of formats, such as **jpg**, **png**, **pdf**, **gif**, **tif** and many others. To begin with, download a test image from
`http://www.csse.monash.edu.au/courseware/cse2330/Pracs/apple.png`
to `$YFD/prac2` directory.

Now type in in the editor window the following commands:

```
clear
fnm = 'apple.png';          % an image file name
UU = imread(fnm, 'png') ;   % getting the image in a MATLAB matrix
figure(2)
image(UU), axis image,  colormap(gray(255))
```

Execute the above script. As a results you should get a gray scale image of an apple. The image is stored in a variable (matrix) `UU`. To get the basic parameters of the image execute the following commands

```
imsz = size(UU)
MnMx = [min(min(UU)) max(max(UU))]
```

Note that the image size is `imsz = [116 115]` and the range of pixel values (image intensities) is `MnMx = [1 245]` (assuming that you work with the same apple I do).

In the pane Workspace you should see all four variables that we created by now including their `sizes` and `classes`. Classes are a bit of a computational "bad news", but we need to be aware of the existence of two types: **double** and **uint8**.

The uint8 type means "unsigned 8-bit integer" that will be used only when we load images in MATLAB. 8-bit integers can take values from 0 to 255.

All other numbers in MATLAB are of type **double**. They can store all values possible that you might ever need. Therefore, the next command we execute is conversion of our image from **uint8** to **double**. This can be done using the command:

```
U = double(UU)/256 ; % remember  ;  here !
```

In addition, by dividing by 256 we have scaled the range of pixel values to be between 0 and 1, which is more convenient for neuronal modelling.

The script processing an apple image looks as follows (remember to specify the mask `HH`):

```
% recurrent 2-D network
  U = double(UU)/256 ;  %  U is between 0 and 1
  Y = zeros(size(U)) ;  % initial value of the afferent signals
  g = 0.05 ;   % adjustable feedback gain
  H = g*HH ;   % mask adjusted with gain
  nn = 10 ;    % number of recurrent runs
  for n = 1:nn
     Y = conv2(Y, H, 'same') + U ;   % 2-D recurrent network
     imagesc(Y)
     axis image, colormap(gray(255))
     title(sprintf('gain = %1.2f,   recurrent run  n = %d', g,n))
     pause(2)   % the loop goes every 2 secs
  end
```

The resulting apple has visibly sharpened edges and might look as in Figure 5–3.

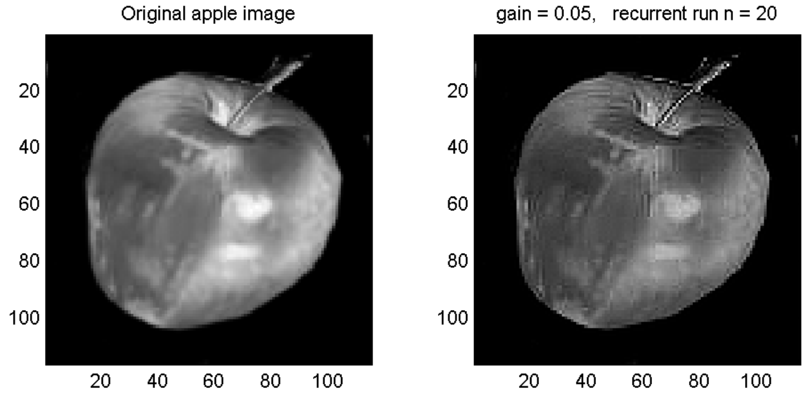


Figure 5–3: Original apple image and the image after sharpening its edges

As the final exercise write a MATLAB script to enhance edges in a **colour** image of your choice preferably in the jpg format. Use recurrent network with 5×5, or greater mask.

Note that a colour image in MATLAB is stored as an array of size $r \times c \times 3$, each $r \times c$ matrix storing a single primary colour information, that is, Red, Green, and Blue, respectively.