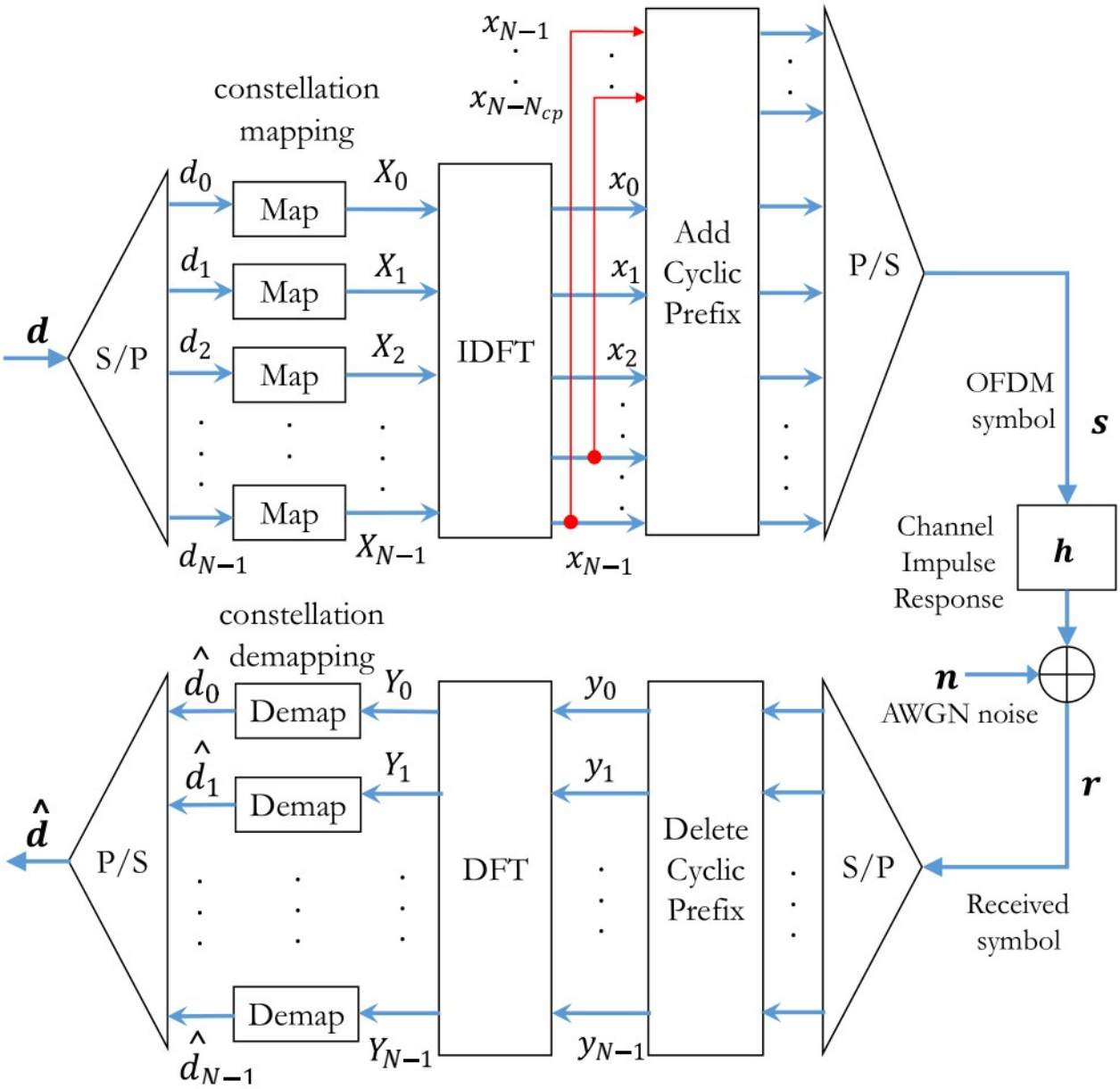
2.2 Understanding the role of cyclic prefix in a CP-OFDM system

A CP-OFDM transceiver architecture is typically implemented using inverse discrete Fourier transform (IDFT) and discrete Fourier transform (DFT) blocks. See below the discrete-time simulation model for OFDM transmission and reception).



2.2.0 The role of the IDFT and DFT blocks

The Fourier transform is used to convert the signals from the time domain to the frequency domain and the inverse Fourier transform is used to convert the signal back

from the frequency domain to the time domain.

The Fourier transform is a powerful tool to analyze the signals and construct them to and from their frequency components.

If the signal is discrete in the time that is sampled, one uses the discrete Fourier transform to convert them to the discrete frequency form (DFT), and vice versa, the inverse discrete transform (IDFT) is used to back convert the discrete frequency form into the discrete-time form.

To reduce the mathematical operations used in the calculation of DFT and IDFT one uses the fast Fourier transform algorithm (FFT and IFFT) which corresponds to (DFT and IDFT), respectively.

In transmitters using OFDM as a multicarrier modulation technology, the OFDM symbol is constructed in the frequency domain by mapping the input bits on the I- and Q-components of the QAM symbols and then ordering them in a sequence with a specific length according to the number of subcarriers in the OFDM symbol.

That is by the mapping and ordering process, one constructs the frequency components of the OFDM symbol. To transmit them, the signal must be represented in the time domain. This is accomplished by the inverse fast Fourier transform IFFT.

So, in summary, the signal is easier synthesized in the discrete frequency domain in the transmitter, and to transmit it must be converted to the discrete-time domain by IFFT.

In an OFDM transmitter, the modulated symbols are assigned to individual subcarriers and sent to an IDFT block.

The output of the IDFT block, generally viewed as time-domain samples, results in an OFDM symbol. Such OFDM symbols are then transmitted across a channel with a certain channel impulse response (CIR).

On the other hand, the receiver applies DFT over the received OFDM symbols for further demodulation of the information in the individual subcarriers.

In reality, a multipath channel or any channel in nature acts as a linear filter on the transmitted OFDM symbols.

Mathematically, the transmitted OFDM symbol (denoted as s[n]) gets linearly convolved with the CIR h[n] and gets corrupted with additive white gaussian noise - designated as w[n].

Denoting linear convolution as '$*$', the received signal in discrete-time can be represented as

$$r[n] = h[n]*s[n]+w[n] \ (* \rightarrow \text{linear convolution}) \ . \qquad (2.1)$$

The idea behind using OFDM is to combat frequency selective fading, where different frequency components of a transmitted signal can undergo different levels of fading.

The OFDM divides the available spectrum into small chunks of subchannels.

Within the subchannels, the fading experienced by the individual modulated symbol can be considered flat.

This opens up the possibility of using a simple frequency domain equalizer to neutralize the channel effects in the individual subchannels.

2.2.1 Circular convolution and designing a simple frequency domain equalizer

From one of the DFT properties, we know that the circular convolution of two sequences in the time domain is equivalent to the multiplication of their individual responses in the frequency domain.

Let s[n] and h[n] be two sequences of length N with their DFTs denoted as S[k] and H[k], respectively.

Denoting circular convolution as ~,

$$h[n] \sim s[n] \equiv H[k]S[k] \qquad (2.2)$$

If we ignore channel noise in the OFDM transmission, from equation (2.1), the received signal is written as

$$r[n] = h[n] * s[n] \quad (* \rightarrow \text{linear convolution}) \qquad (2.3)$$

We can note that the channel performs a linear convolution operation on the transmitted signal.

Instead, if the channel performs circular convolution (which is not the case in nature) then the equation would have been

$$r[n] = h[n] \sim s[n] \quad (\sim \rightarrow \text{circular convolution}) \qquad (2.4)$$

By applying the DFT property given in equation (2.2),

$$r[n] = h[n] \sim s[n] \equiv R[k] = H[k]S[k] \qquad (2.5)$$

As a consequence, the channel effects can be neutralized, at the receiver, using a simple frequency domain equalizer (actually this is a zero-forcing equalizer when viewed in the time domain) that just inverts the estimated channel response and multiplies it with the frequency response of the received signal, to obtain the estimates of the OFDM symbols in the subcarriers as

$$\hat{S}[k] = R[k] / H[k] \qquad (2.6)$$

2.2.2 Demonstrating the role of cyclic prefix

The simple frequency domain equalizer shown in equation (2.6) is possible only if the channel performs circular convolution.

But in nature, all channels perform linear convolution. The linear convolution can be converted into circular convolution by adding a cyclic prefix (CP) in the OFDM architecture.

The addition of CP makes the linear convolution imparted by the channel appear as a circular convolution to the DFT process at the receiver [3].

Let's understand this by demonstration.

To simplify stuff, we will create two randomized vectors for s[n] and h[n].

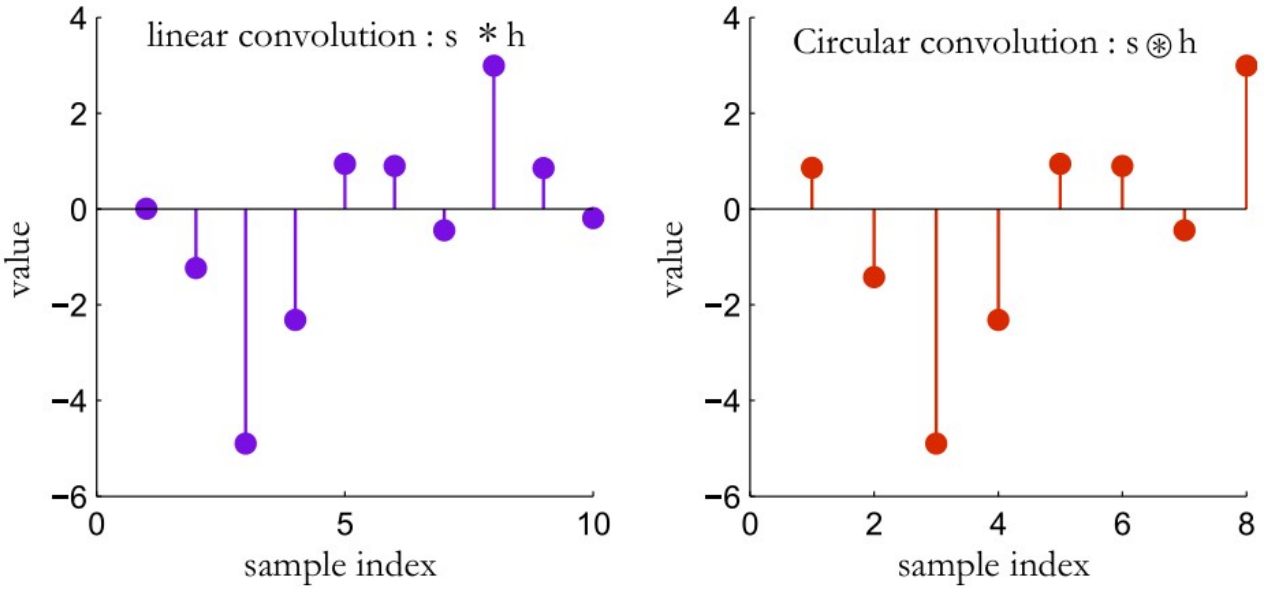
s[n] is of length 8, the channel impulse response h[n] is of length 3 and we intend to use N = 8-point DFT/IDFT wherever applicable.

```
N = 8; %period of DFT
s = randn(1,8);
h = randn(1,3);
```

Now, convolve the vectors s and h linearly and circularly.

```
lin_s_h = conv(h,s)        %linear convolution of h and s
cir_s_h = cconv(h,s,N)   %circular convolution of h and s with period N
```

The outputs are plotted in the following figure (Difference between linear convolution and circular convolution).



We note that the linear convolution and the circular convolution do not yield identical results.

Let's append a cyclic prefix to the signal s by copying the last N cp symbols from s and pasting

it to its front.

Since the channel delay is 3 symbols (CIR of length 3), we need to add at least 2 CP symbols.

```
Ncp = 2; %number of symbols to copy and paste for CP
s_cp = [s(end-Ncp+1:end) s]; % copy last Ncp syms from s, add as prefix
```

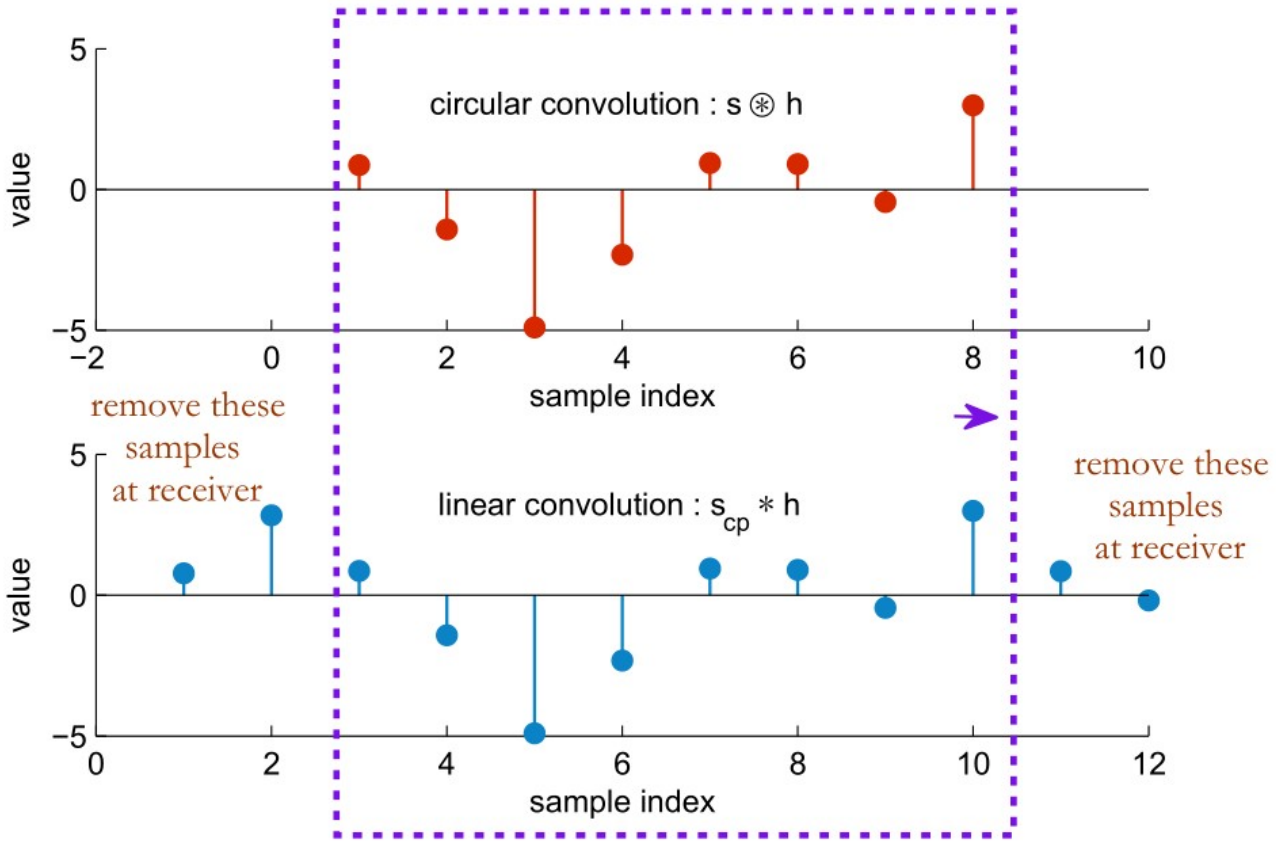
Let us assume that we are sending the cyclic-prefixed OFDM symbol s_{cp} through a channel that performs linear filtering.

```
lin_scp_h=conv(h,s_cp)    % linear conv. of CP-OFDM symbol s_cp and CIR h
```

Compare the outputs due to cir_s_h = s~h and lin_scp h = s cp ∗h.

We can immediately recognize that the middle section of lin_scp_h is exactly the same as the cir_s_h vector.

This is shown in the figure below (Equivalence of circular convolution and linear convolution with cyclic prefixed signal).



We have just tricked the channel to perform circular convolution by adding a cyclic extension to the OFDM symbol.

At the receiver, we are only interested in the middle section of the lin_scp_h which is the channel output.

The first block in the OFDM receiver removes the additional symbols from the front and back of

the channel output.

The resultant vector is the received symbol **r** after the removal of the cyclic prefix in the receiver.

```
r = lin_scp_h(Ncp+1:N+Ncp) % cut from index Ncp+1 to N+Ncp
```

2.2.3 Verifying DFT property

The DFT property given in equation (2.5) can be re-written as

$$
\begin{aligned}
r[n] &= IDFT\{R[k]\} \\
&= IDFT\{ H[k]S[k] \} \\
&= IDFT \{DFT (h[n]) DFT (s[n]) \\
&= h[n] \sim s[n] \qquad (2.7)
\end{aligned}
$$

To verify this in Matlab, take N-point DFTs of the received signal and CIR.

Then, we can see that the IDFT of the product of DFTs of **s** and **h** will be equal to the N-point circular convolution of **s** and **h**.

```
R = fft(r,N); %frequency response of received signal
H = fft(h,N); %frequency response of CIR
S = fft(s,N); %frequency response of OFDM signal (non CP)

r1 = ifft(S.*H); %IFFT of product of individual DFTs

display(['IFFT(DFT(H)*DFT(S)) : ',num2str(r1)])
display([cconv(s,h): ', numstr(r)])
```