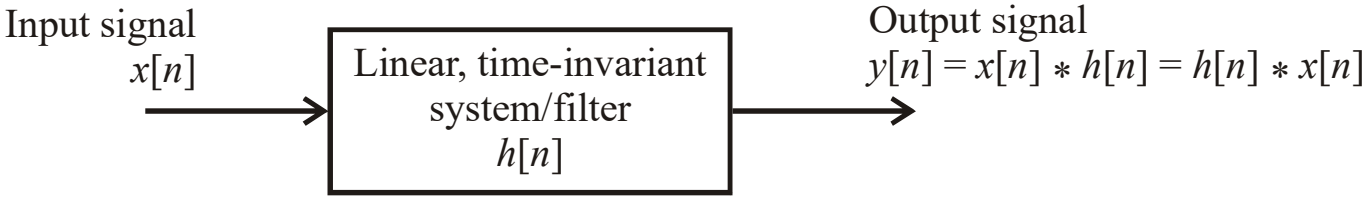
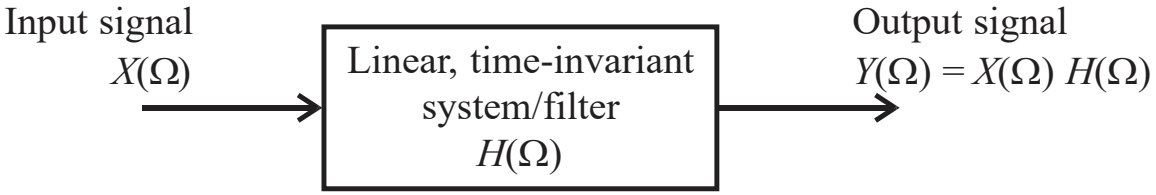
Fourier Analysis using the DFT or FFT

In the time-domain (Chapter 2), we discussed the convolution representation (see below) of a discrete-time system using the unit-pulse response $h[n]$.

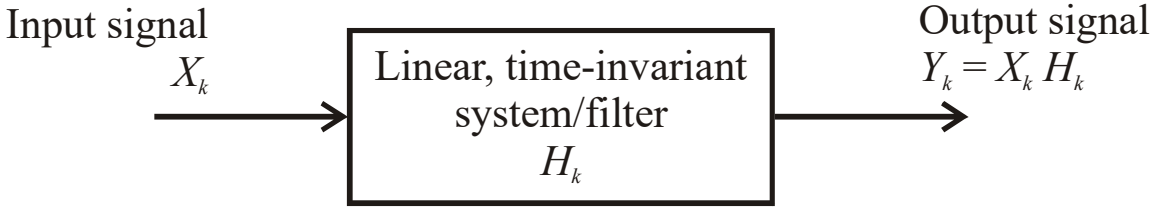


Using discrete-time Fourier transform (DTFT) and analysis, this system can be represented as shown below.



The DTFT is awkward to use directly as it is a hybrid between discrete-time data points and the continuous-frequency realm of Ω.

In actual practice, it is easier and more efficient to use the discrete Fourier transform (DFT) which is often implemented using the fast Fourier transform (FFT) algorithm (see below).



Here, we put in discrete-time data points (e.g., $x[n]$ and $h[n]$), convert to discrete frequency points (e.g., X_k and H_k) using the DFT or FFT, multiply to get discrete output frequency points Y_k, and convert back to the discrete-time output data points $y[n]$ using the inverse DFT (IDFT) or inverse FFT (IFFT).

Note: The DTFT frequency Ω and DFT indices k are related by $\Omega_k = \dfrac{2\pi k}{N}$.

So, X_k and H_k are sampled versions of the DTFT $X(\Omega)$ and $H(\Omega)$, e.g.,

$$H_k = H\left(\Omega_k = \frac{2\pi k}{N}\right).$$

Key assumptions-

$x[n]$ is finite in length, say N-points, with index range $P \le n \le P + N - 1$

and

$h[n]$ is finite in length, say M-points, with index range $Q \le n \le Q + M - 1$.

From Chapter 2 and the **convolution representation**, we know that the discrete-time system output $y[n] = x[n] * h[n]$ will be of length $N + M - 1$ with an index range of $P + Q \le n \le P + Q + N + M - 2 = P + Q + \text{length}(y) - 1$.

However, the output $y[n]$ will be accurate over the index range $P + Q \le n \le P + Q + \min\{N - 1, M - 1\}$.

Note: Near the beginning indices, a filter/system must 'charge-up', i.e., there must be enough points of $h[n]$ involved with the convolution for the filter/system to work properly.

Procedure

1) Select appropriate overall number of points/length for the algorithm.

   a) To use the FFT algorithm, choose an FFT length L_{FFT} that is a power of 2, i.e., $N + M \le L_{\text{FFT}} = 2^r$.

   b) To use the DFT algorithm, choose a DFT length $L_{\text{DFT}} = N + M$.

2) "zero-pad" both $x[n]$ and $h[n]$ to equal lengths. That is, insert/add zeros onto the ends of $x[n]$ and $h[n]$ so that both vectors are the same length (index ranges can still be different).

   a) To use the FFT algorithm, make both $x[n]$ and $h[n]$ of length L_{FFT}.

   b) To use the DFT algorithm, make both $x[n]$ and $h[n]$ of length L_{DFT}.

   Note: In MATLAB, this zero-padding can be done automatically by specifying the desired length in the fft() function, e.g., ftt(x, L_{FFT}) or ftt(h, L_{DFT}). If the dft() function is used, you must do the zero-padding before calling the function.]

3) Use the FFT/DFT to move to the frequency-domain.

   a) Compute the L_{FFT}-point FFT of $x[n]$ and $h[n]$ to get X_k and H_k. E.g., In MATLAB, Xk = fft(x, L_{FFT}) and Hk = fft(h, L_{FFT}).

   b) Compute the L_{DFT}-point DFT of $x[n]$ and $h[n]$ to get X_k and H_k. E.g., In MATLAB, Xk = fft(x, L_{DFT}) and Hk = fft(h, L_{DFT}) **OR**, using the dft() function, Xk = dft(x) and Hk = dft(h).

   Note: Since the time indices are computed separately, it is OK to ignore the frequency phase shift(s) introduced by having $x[n]$ and $h[n]$ start at P and Q respectively.

4) In both cases, perform a point-by-point vector multiplication of X_k and H_k to get Y_k. E.g., In MATLAB, $Yk = X_k$.* H_k .

5) Compute the output $y[n]$ by using the inverse operation.

   a) Compute the L_{FFT}-point *inverse* FFT (IFFT) of Y_k to get $y[n]$. E.g., In MATLAB, use y = ifft(Yk). The resulting vector y will be of length L_{FFT} with an index range of $P+Q \le n \le P+Q+L_{\text{FFT}}-1$.

   b) Compute the L_{DFT}-point *inverse* DFT (IDFT) of Y_k to get $y[n]$. E.g., In MATLAB, use y = ifft(Yk) **OR** y = idft(Yk). The resulting vector y will be of length L_{DFT} with an index range of $P+Q \le n \le P+Q+L_{\text{DFT}}-1$.

   - The index range of the output $y[n]$ can contain many zeros as a result of the zero-padding.
   - The index range where the output $y[n]$ is **non-zero** is still $P+Q \le n \le P+Q+N+M-2 = P+Q+\text{length}(y)-1$.
   - However, the output $y[n]$ will be **accurate (strictly)** over the index range $P+Q \le n \le P+Q+\min\{N-1, M-1\}$.

Practical considerations

1) What if either or both $x[n]$ and $h[n]$ are **not** finite in length?
   - Can still get usable approximate results using DFT or FFT signal analysis by truncating $x[n]$ and/or $h[n]$ to make them of finite length.
   - This will only work if $x[n]$ and/or $h[n]$ approach zero as $n \to \infty$ (and, if non-causal, $n \to -\infty$).
   - Need some criteria for truncation, e.g., choose

$$\frac{|h[n_{\text{last}}]|}{|h[n]|_{\max}} \leq \text{some \# (e.g., 0.01 or 1\%) for all } |n| \geq |n_{\text{last}}|$$

   - If a rigorous criteria is used to truncate $h[n]$, the output $y[n]$ will be **approximately accurate** (i.e., look accurate) over the index range $P+Q \leq n \leq P+Q+M-1$. In other words, the **effective** index range for $y[n]$ is set by the length of the input signal $x[n]$ rather than $h[n]$.

2) Sometimes, we might wish to select a desired DTFT frequency response $H(\Omega)$ (e.g., some LP, HP, or BP filter) to operate on a discrete-time signal $x[n]$. How do we proceed?

Option 1: Find $h[n]$ for the desired $H(\Omega)$.

- If $h[n]$ is finite in length, proceed as before.
- If $h[n]$ is infinite in length or too long to be practical, truncate $h[n]$ as discussed in 1), and then proceed.

Option 2: Skip steps 1-3 in procedure for $h[n]$, and directly compute values for H_k from the desired $H(\Omega)$ using the relationship $H_k = H\left(\Omega_k = \frac{2\pi k}{N}\right)$ where $N = L_{\text{DFT}}$ or $N = L_{\text{FFT}}$. The challenge in doing this is to adequately sample $H(\Omega)$ in frequency, i.e., pick L_{DFT} or L_{FFT} so $H_k = H(\Omega_k)$ for $0 \leq k \leq L_{\text{DFT}}-1$ or $0 \leq k \leq L_{\text{FFT}}-1$ is a good representation of $H(\Omega)$. As a rule of thumb, a plot of H_k that looks like $H(\Omega)$, when the dots are connected, will work.