Convolution and LTI

Chiara Ravazzi

National Research Council of Italy
Institute of Electronics, Computer and Telecommunication Engineering, c/o Politecnico di Torino, Italy

Systems Modeling & Control Group

Analisi tempo-frequenza e multiscala – 01RMQNG

Sequences

1. Infinite-length sequences:

$$\mathbb{C}^{\mathbb{Z}} = \left\{ x = [\ \ldots,\ x_{-1}, \boxed{x_0},\ x_1,\ \ldots\ ]^\top \;\middle|\; x_n \in \mathbb{C}, n \in \mathbb{Z} \right\}$$

2. Finite-length sequences

$$\mathbb{C}^{N} = \left\{ x = [x_0,\ x_1,\ \ldots,\ x_{N-1}]^\top \;\middle|\; x_n \in \mathbb{C} \right\}$$

seen as an infinite-length sequences by extesion

- sequence with finite support : $x_n = 0\ \forall n \notin \{0, \ldots, N-1\}$
- periodic sequence : $x_{n+kN} = x_n$

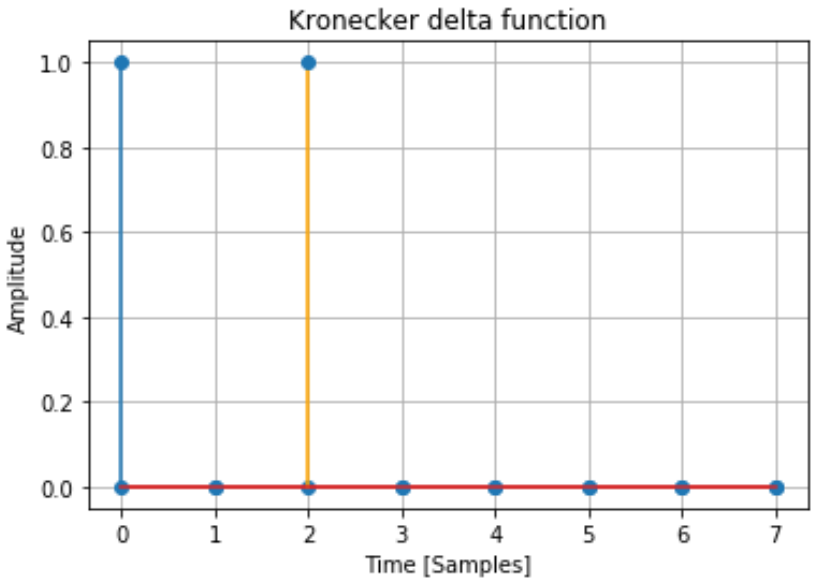
Sequences

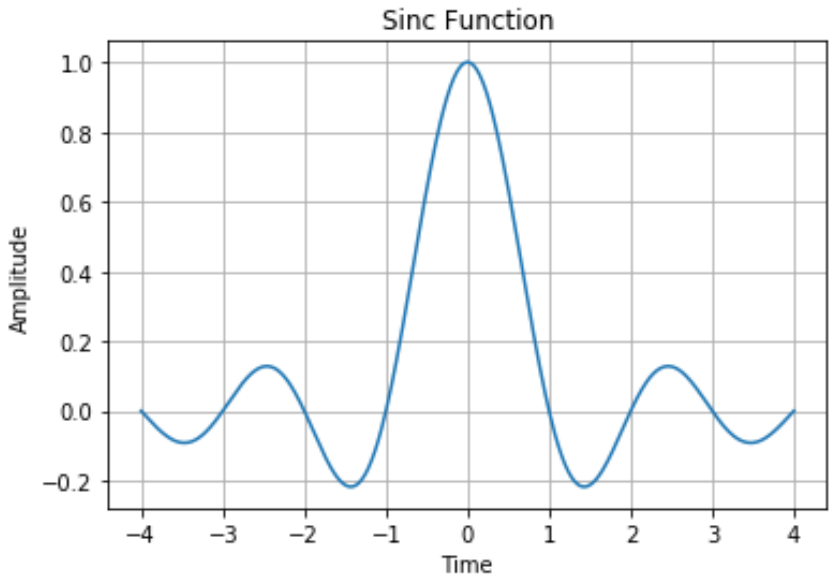
Modules for spectral analysis in SciPy library : fftpack and signal
Importing modules : from scipy import signal
Examples

Special sequences :

1. Kronecker delta sequence :
   - impulse at the 0-th element δ_n : `signal.unit_impulse(8)`
   - impulse offset by 2 samples δ_{n-2} : `signal.unit_impulse(8, 2)`
2. Sinc function : `numpy.sinc(x)`

$$\text{sinc}(x) = \sin(\pi x)/(\pi x)$$





Convolution

Convolution (Definition)

Let $x, h \in \mathbb{C}^{\mathbb{Z}}$

$$(x * h)_n = \sum_{k \in \mathbb{Z}} x_k h_{n-k} = \sum_{k \in \mathbb{Z}} x_{n-k} h_k$$

Properties :

- Commutativity : $x * h = h * x$
- Associativity : $g * (h * x) = g * h * x = (g * h) * x$
- Convolution and modulation theorem :

$$\mathcal{F}(x * h) = \mathcal{F}(x) \cdot \mathcal{F}(h) \qquad \mathcal{F}(x \cdot h) = \mathcal{F}(x) * \mathcal{F}(h)$$

Matrix view :

$$h * x = Hx \qquad H = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \dots & h_0 & h_{-1} & h_{-2} & \dots \\ \dots & h_1 & \boxed{h_0} & h_{-1} & \dots \\ \dots & h_2 & h_{-1} & h_0 & \dots \\ & \vdots & \vdots & \vdots & \end{bmatrix}$$

Circular convolution

Circular convolution (Definition)

Let $x, h \in \mathbb{C}^N$

$$(h * x)_n = \sum_{k \in \mathbb{Z}} x_k h_{(n-k) \bmod n}$$

Matrix view :

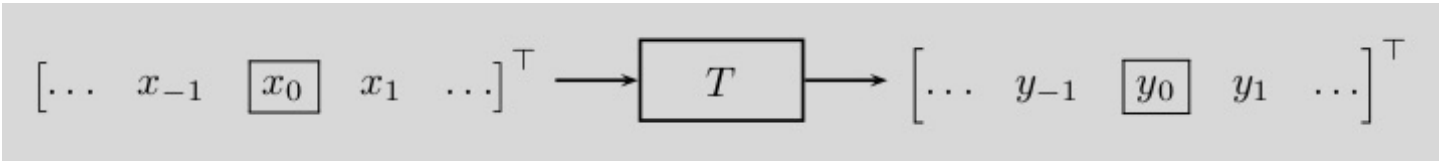
$$h * x = Hx \qquad H = \begin{bmatrix} h_0 & h_{N-1} & h_{N-2} & \dots & h_1 \\ h_1 & h_0 & h_{N-1} & \dots & h_2 \\ h_2 & h_1 & h_0 & \dots & h_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{N-1} & h_{N-2} & h_{N-3} & \dots & h_0 \end{bmatrix}$$

Convolution

In Python :

- Module : from scipy.linalg import circulant
  Call function : `circulant(h)`
- `numpy.convolve(a, v, mode)`
  - $a \in \mathbb{C}^N$, $c \in \mathbb{C}^M$
  - mode='full' : convolution at each point (output of length $N + M - 1$)
  - mode='same' : output of length $\max(M, N)$
  - mode='valid' : convolution for points where the signals overlap completely (output of length $\max(M, N) - \min(M, N) + 1$)
- `scipy.signal.fftconvolve` : convolve two arrays using the Fast Fourier Transform
- `scipy.linalg.toeplitz` : construct the convolution operator.

LTI Systems



Discrete time systems : $T : \mathbb{C}^{\mathbb{Z}} \to \mathbb{C}^{\mathbb{Z}}$

$$y = T(x)$$

- Linear systems:

  for any $x, y \in \mathbb{C}^{\mathbb{Z}}, \alpha, \beta \in \mathbb{C}, \quad T(\alpha x + \beta y) = \alpha T(x) + \beta T(y)$

- Shift invariant:

  for any $x \in \mathbb{C}^{\mathbb{Z}}, k \in \mathbb{Z}, y = T(x) \Longrightarrow y' = T(x'),\ x'_n = x_{n-k}, y'_n = y_{n-k}$

LTI Systems (Linear shift-invariant systems)

Let $h = T(\delta)$ be the impulse response, then

$$(Tx)_n = (h * x)_n = (Hx)_n$$

where H convolution operator.