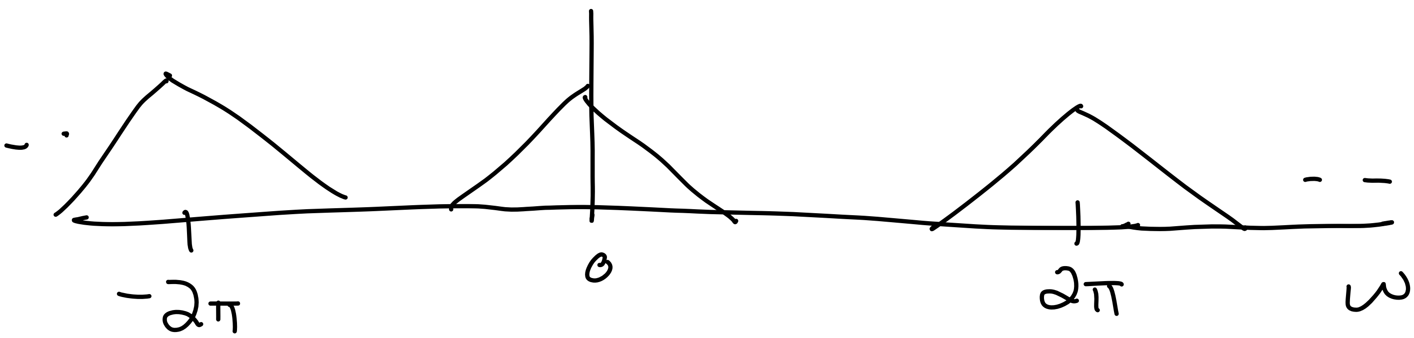
Chap 7.1 - 7.2

To find the Fourier transform of a discrete-time signal, $x[n]$, we can take the DTFT

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

periodic with periodic 2π

$|X(e^{j\omega})|$



There are two major problems with using the computer to compute the DTFT

1) if $x[n]$ is an infinite sequence of nonzero values (ex. $x[n] = \left(\frac{7}{8}\right)^n$),

DTFT requires computing an infinite summation

So in reality, we just compute a finite number of terms, say N of them

$$X(e^{j\omega}) \approx \sum_{n=0}^{N-1} x[n]e^{-j\omega n} = X_N(e^{j\omega})$$

However, the DTFT we are computing here is no longer the DTFT of $x[n]$, but the DTFT of a version of $x[n]$ that only has N components (this doesn't matter if $x[n]$ has less than N nonzero components)

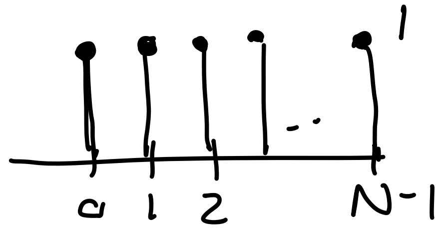
We can denote $x[n]$ that has N components as $x_N[n]$. Note

$$x_N[n] \equiv x[n]\, p_N[n]$$

where

$p_N[n]$ is the rectangular pulse sequence

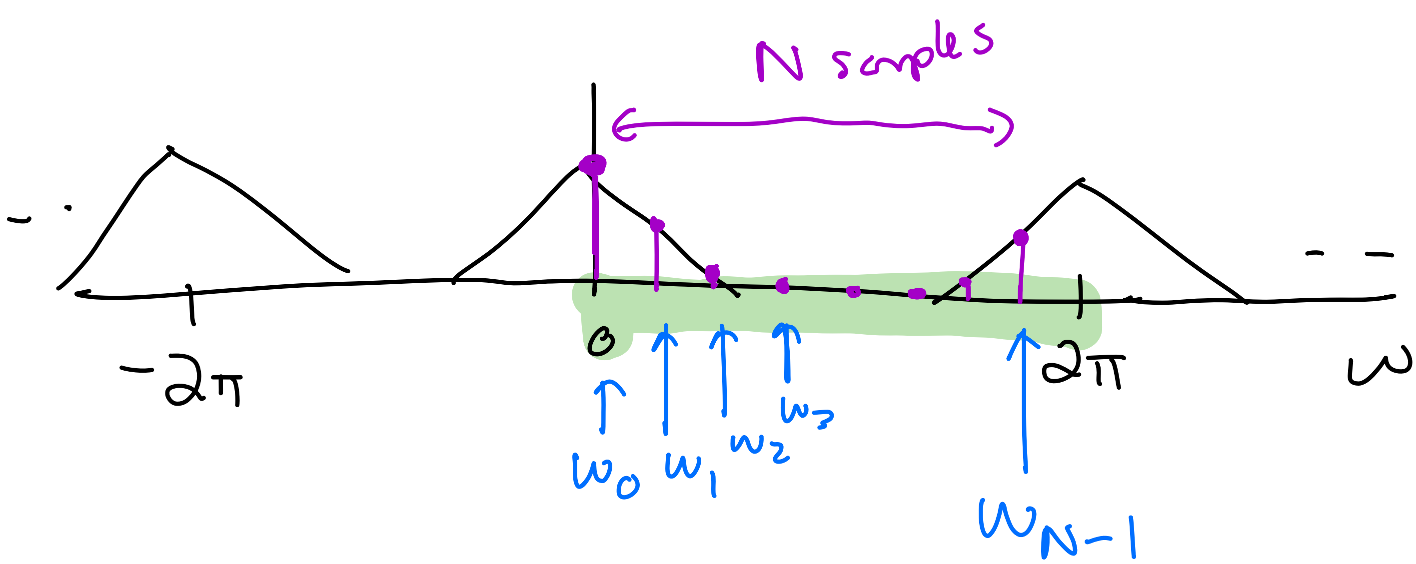
$$p_N[n] \equiv \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases}$$



If you have a real-time measuring system, and you store some $x[n]$, you are effectively using a rectangular pulse function

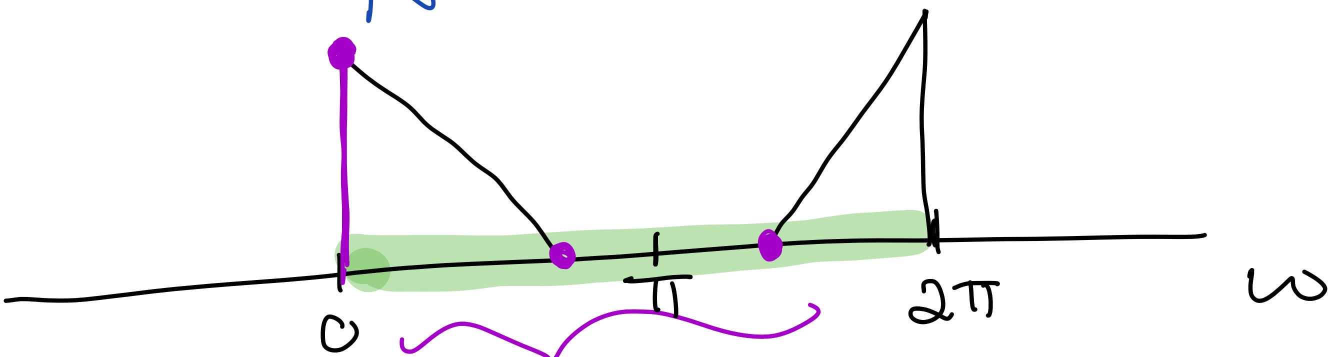
2) Computer can not compute and store $X(e^{jw})$ for all w. We can only compute and store a <u>finite</u> # of values.

Since $X(e^{jw})$ is periodic with period 2π, we only need to store the values of one period. We want to store N samples. It would be easier if the samples are equally spaced



The mapping between the normalized frequency and the kth sample is

$$\omega_k = \frac{2\pi}{N} k \,, \quad k = 0, 1, \ldots N-1$$



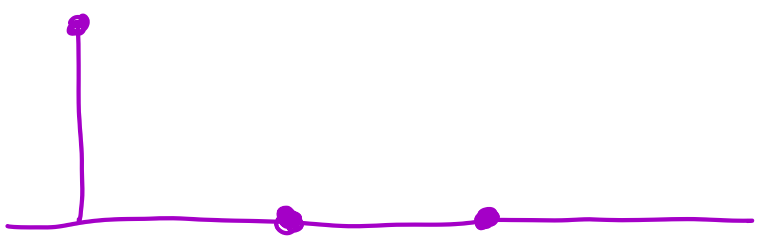
we want $N=3$ samples

$$\omega_0 = \frac{2\pi}{3}(0) = 0 \qquad k = 0$$

$$\omega_1 = \frac{2\pi}{3}(1) = \frac{2\pi}{3} \qquad k = 1$$

$$\omega_2 = \frac{2\pi}{3}(2) = \frac{4\pi}{3} \qquad k = 3-1 = 2$$

So the computer stores and sees



Now you see that there may be a problem if N is too small

In reality, the computer does not compute the DTFT of $x[n]$ but the discrete Fourier transform (DFT) of $x_N[n]$

DFT has N components $\to \omega_k$

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\omega_k n} \qquad k = 0, 1, \ldots N-1$$

where

$$\omega_k = \frac{2\pi}{N} k$$

(There is a typo in the book of equation 7.15) and 7.17

Inverse DFT is

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N} kn}, \quad n = 0, \ldots N-1$$

Instead of computing $X[0], X[1], X[2]$ sequentially, we compute them all at once using linear algebra

$$\underbrace{\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[K-1] \end{bmatrix}}_{\boldsymbol{X}} = \underbrace{\begin{bmatrix} e^{j\omega_0 0} & e^{j\omega_0 1} & \dots & e^{j\omega_0 (N-1)} \\ e^{j\omega_1 0} & e^{j\omega_1 1} & \dots & e^{j\omega_1 (N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ e^{j\omega_{K-1} 0} & e^{j\omega_{K-1} 1} & \dots & e^{j\omega_{K-1} (N-1)} \end{bmatrix}}_{\boldsymbol{W}} \underbrace{\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}}_{\boldsymbol{x}}$$

$$\underline{X} = Wx$$

and the inverse DFT becomes

$$x = W^{-1}\underline{X}$$

DFT is implemented in the computer using an even more efficient algorithm (compared to matrix-vector multiplication) called the FFT (fast Fourier transform)