

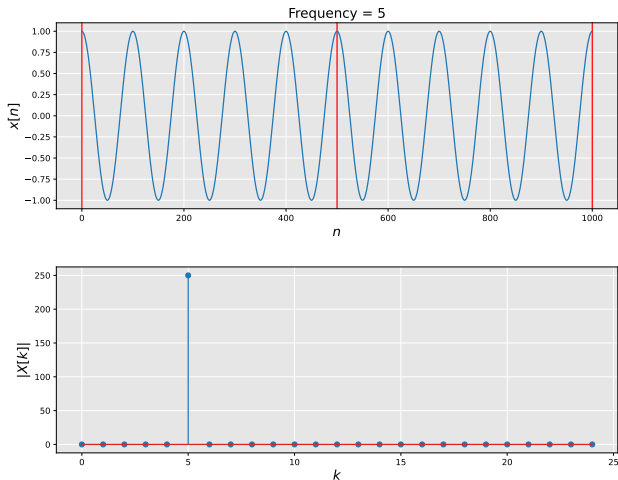
# Instantaneous Frequency and Pitch Scaling

Digital Signal Processing

September 25, 2025

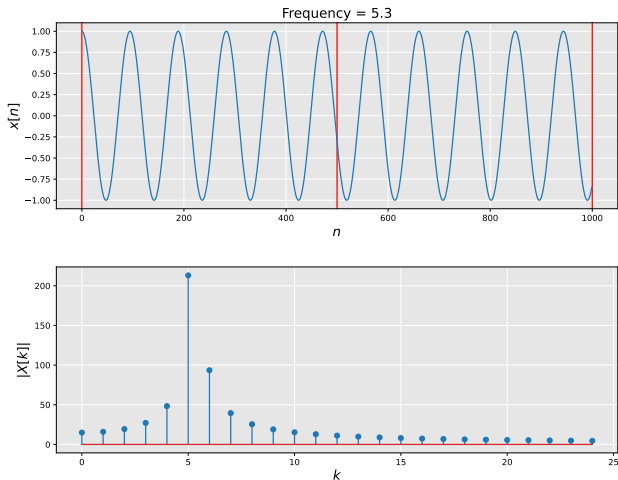


# Instantaneous Frequency



Frequency is clear in DFT when it is an integer.

# Instantaneous Frequency



But it spreads across multiple bins when it is not an integer.

# Can we Recover Non-Integer Frequency?

Continuous sinusoid:

$$x(t) = \cos(\omega_0 t + \phi)$$

Its phase angle is:

$$\theta(t) = \omega_0 t + \phi$$

Its derivative is:

$$\frac{d\theta}{dt}(t) = \omega_0$$

Which is the frequency!

# Discrete Phase Derivative

Discrete sinusoid:

$$x[n] = \cos(\omega_0 n + \phi)$$

Its phase angle is:

$$\theta[n] = \omega_0 n + \phi$$

Discrete derivative (backward difference):

$$\begin{aligned}\nabla\theta[n] &= \theta[n] - \theta[n-1] \\ &= \omega_0 n + \phi - (\omega_0(n-1) + \phi) \\ &= \omega_0\end{aligned}$$

Again, frequency!

# Using the STFT Phase Angle

- The DFT doesn't give us phase angle as a function of time.
- The STFT does give an estimated phase angle as a function of time!
- Given an STFT,  $X[k, m]$ , we can estimate the instantaneous frequency represented in frequency bin  $k$  at time  $m$  as:

$$\omega^*[k, m] = \frac{\phi[k, m] - \phi[k, m - 1]}{h},$$

where  $\phi[k, m] = \text{Arg}(X[k, m])$ , and  $h$  is the hop size.

- **Note:**  $\omega^*[k, m]$  is the instantaneous frequency for each bin  $0 \leq k < W$  and time  $1 \leq m < H$  (no backward difference for  $m = 0$ )

# Units of Frequency

- For STFT of window length  $W$ :

$$\omega_0 = \frac{2\pi}{W}$$

- So,  $k$ th frequency bin represents frequency

$$\omega_0 k = \frac{2\pi k}{W}$$

- The number  $k$  is how many cycles the sinusoid  $e^{i\omega_0 kn}$  makes in our window ( $W$  time steps)
- If  $T$  is the sampling period (time between samples of  $x[n]$ , in seconds), then the  $k$ th frequency bin represents  $\frac{k}{WT}$  Hz.  
(Note:  $WT$  is window length in seconds.)

# Expected Phase Shift

If frequency was exactly the  $k$ th bin frequency,  $\omega^* = \omega_0 k$ , then

$$\omega^*[k, m] = \frac{\phi[k, m] - \phi[k, m - 1]}{h} = \omega_0 k$$

Rearranging, we get

$$\phi[k, m] - \phi[k, m - 1] = \omega_0 k h$$

This is the **expected phase shift in one hop**.



# Instantaneous Frequency

The remainder between actual and expected phase shifts:

$$\phi_r[k, m] = \underbrace{\phi[k, m] - \phi[k, m - 1]}_{\text{actual phase shift}} - \underbrace{\omega_0 k h}_{\text{expected}} .$$

Now, we wrap this remainder to be in  $[-\pi, \pi)$ , and calculate our final instantaneous frequency:

$$\omega^*[k, m] = \frac{\text{wrap}(\phi_r[k, m])}{h} + \omega_0 k .$$

Or, in terms of integer frequency bins:

$$\kappa[k, n] = \frac{\omega^*[k, m]}{\omega_0} = \frac{\text{wrap}(\phi_r[k, m])}{\omega_0 h} + k .$$

# Wrapping to Principal Angle

To wrap an angle between  $[-\pi, \pi)$ :

$$\text{wrap}(\phi) = (\phi + \pi) \bmod (2\pi) - \pi.$$

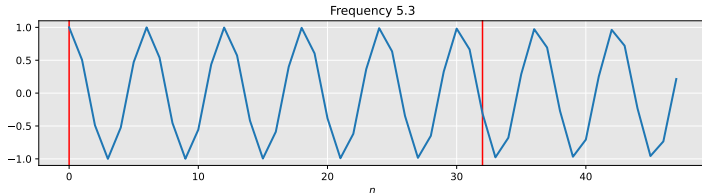
Here, the notation  $x \bmod y$  means the remainder from floating point division.

The `%` operator in Python will do this (or the function `np.mod`)

# Example

Cosine with frequency  $f = 5.3$  and window  $W = 32$ :

$$x[n] = \cos(2\pi f n / 32)$$

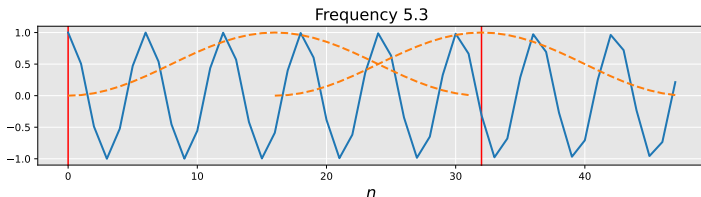


# Example

Cosine with frequency  $f = 5.3$  and window  $W = 32$ :

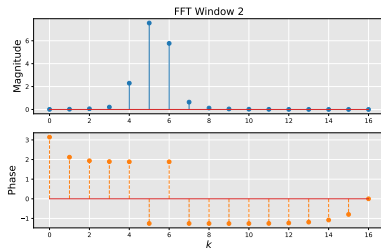
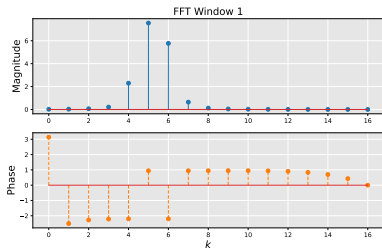
$$x[n] = \cos(2\pi f n / 32)$$

Estimate STFT with just two windows ( $h = 16$ ):



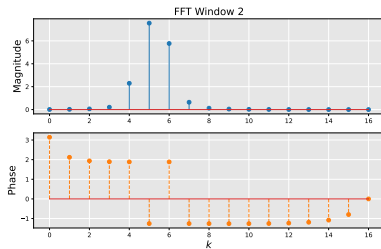
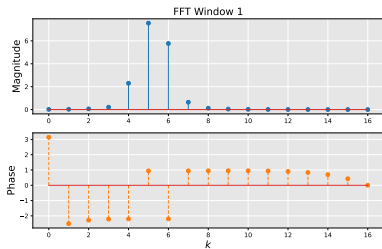
# Example

STFT:

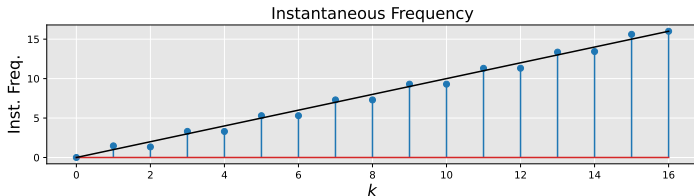


# Example

STFT:

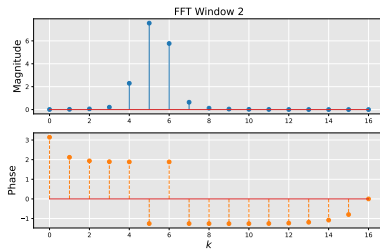
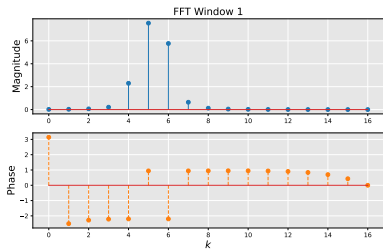


Estimated Frequencies  $\omega^*[k, n]$ :

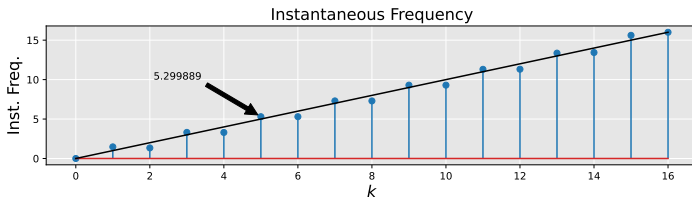


# Example

STFT:

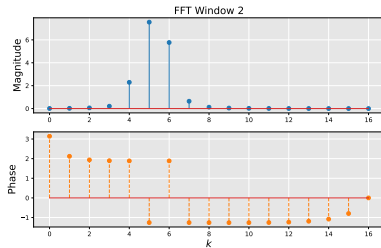
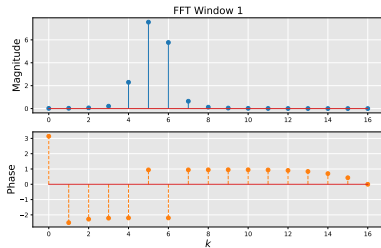


Estimated Frequencies  $\omega^*[k, n]$ :

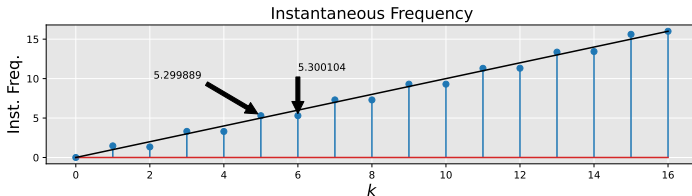


# Example

STFT:



Estimated Frequencies  $\omega^*[k, n]$ :

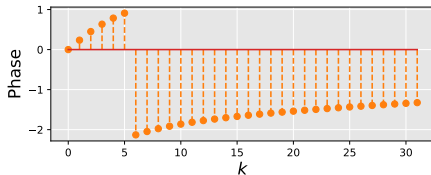
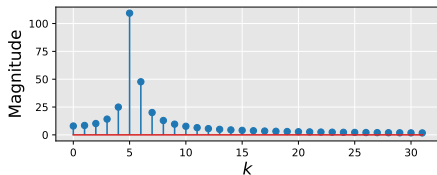




# Frequency (Pitch) Scaling

# How To Change Pitch?

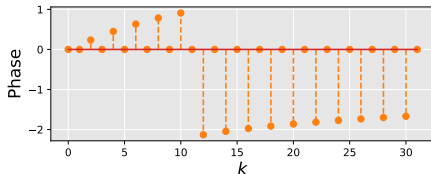
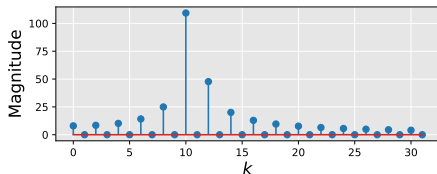
Naive Frequency Scaling FFT



Why not just move everything to double the frequency?

$$Y[2k] = X[k]$$

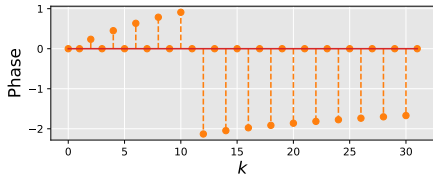
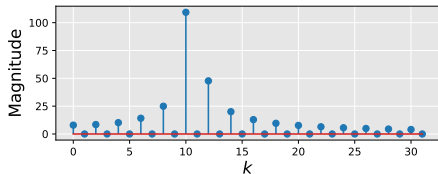
Naive Frequency Scaling FFT



# Naive Frequency Scaling Result

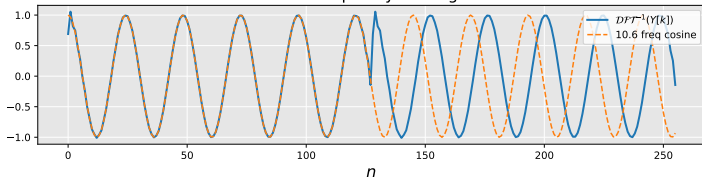
$$Y[2k] = X[k]$$

Naive Frequency Scaling FFT



$$\mathcal{DFT}^{-1}(Y[k])$$

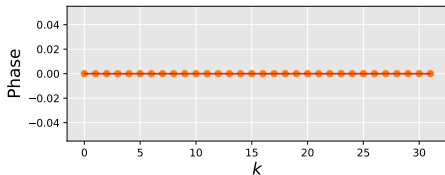
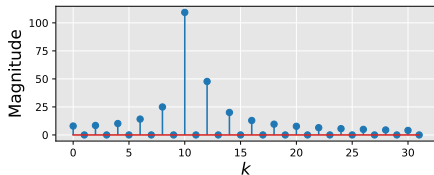
Naive Frequency Scaling x2



# How About Zeroing-Out the Phase?

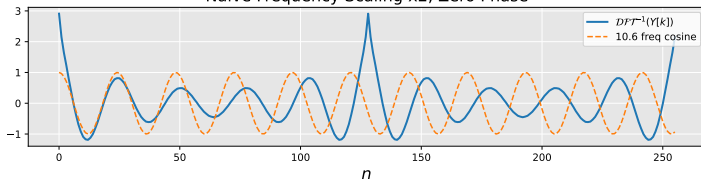
$$|Y[2k]| = |X[k]|, \quad \text{Arg}(Y[2k]) = 0$$

Naive Frequency Scaling FFT



$$\mathcal{DFT}^{-1}(Y[k])$$

Naive Frequency Scaling x2, Zero Phase



# Smart Frequency Scaling Algorithm

- 1 Use STFT to compute instantaneous frequency,  $\kappa[k, m]$
- 2 Scale by some factor:  $\kappa_s[k, m] = R\kappa[k, m]$
- 3 Find new bins:  $k_s = \text{round}(Rk)$
- 4 Find new phase shifts:  $\Delta\phi[k_s, m] = \omega_0 h(\kappa_s[k, m] - k_s)$
- 5 Accumulate with phase from previous time hop:

$$\phi_s[k_s, m] = \text{wrap}(\phi_s[k_s, m-1] + \Delta\phi[k_s, m] + \omega_0 k_s h).$$

- 6 Set  $Y[k_s, m] = |X[k, m]|e^{i\phi_s[k_s, m]}$ .
- 7 Synthesize output signal:  $y[n] = \text{OLA}(Y[k, m])$

# Smart Frequency Scaling Result

