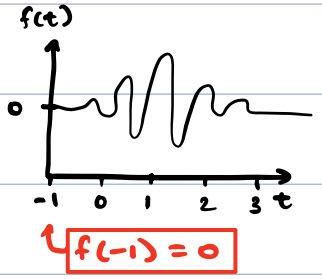
Signal Processing

signal ⟶ function of time!

$f(\cdot)$ ⟶ whole signal function

$f(t)$ ⟶ value of signal at time t

VALUE of a function!



↑ $f(-1) = 0$

domain : set of points input t can be

e.g. uniformly sampled at X kHz

discrete-time: $t = 0, \pm 1, \pm 2, \ldots$

↑ "epochs" not necessarily seconds

$f(t)$ { value, vector-valued, boolean

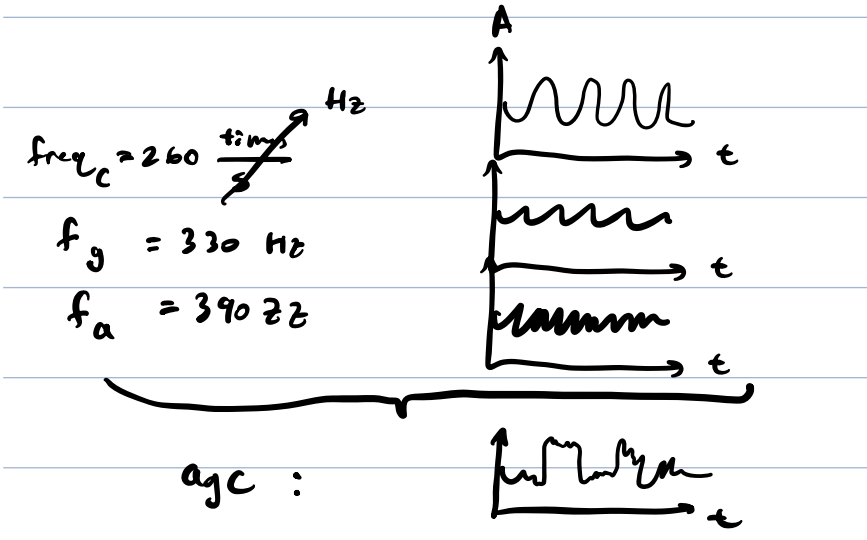
$$\vec{V}(t) = \begin{bmatrix} V_1(t) \\ V_2(t) \\ V_3(t) \end{bmatrix}$$

↑ vector valued signal function

← scalar values

Fourier Transform

signal fn. in time ⟶ [FT] ⟶ signal fn. in frequency



Convolutions

$$(x * g)[k] = \sum_j x_{k-j}\, g_j$$

$x = [x_0\ x_1\ x_2\ x_3\ x_4]$ ← input sequence

$g = [g_2\ g_1\ g_0]$ ← convolving filter

length of convolution

$$(x * g)[k] = \sum_{j=0}^{2} x_{k-j} \cdot g_j$$

$$(x*g)[0] = x_0 g_0 + \cancel{x_{-1} g_1} + \cancel{x_{-2} g_2} = x_0 g_0$$

$$(x*g)[1] = x_1 g_0 + x_0 g_1 + \cancel{x_{-1} g_2} = x_1 g_0 + x_0 g_1$$

$$(x*g)[2] = x_2 g_0 + x_1 g_1 + x_0 g_0 = x_2 g_0 + x_1 g_1 + x_0 g_0$$

Notice:

- if I treat inputs as polynomial coefficients, the computation is identical!

$$f(z) = x_0 + x_1 z + \ldots + x_n z^n$$

$$g(z) = g_0 + g_1 z + g_2 z^2$$

$$h(z) = f(z)\, g(z) = (g_0 + g_1 z + g_2 z^2)(x_0 + x_1 z + \ldots + x_n z^n)$$

$$= h_0 z + h_1 z^2 + h_2 z^3 + \ldots$$

coefficient at index i is $(x * g)[i]$!

✓ Convolution == Polynomial multiplication

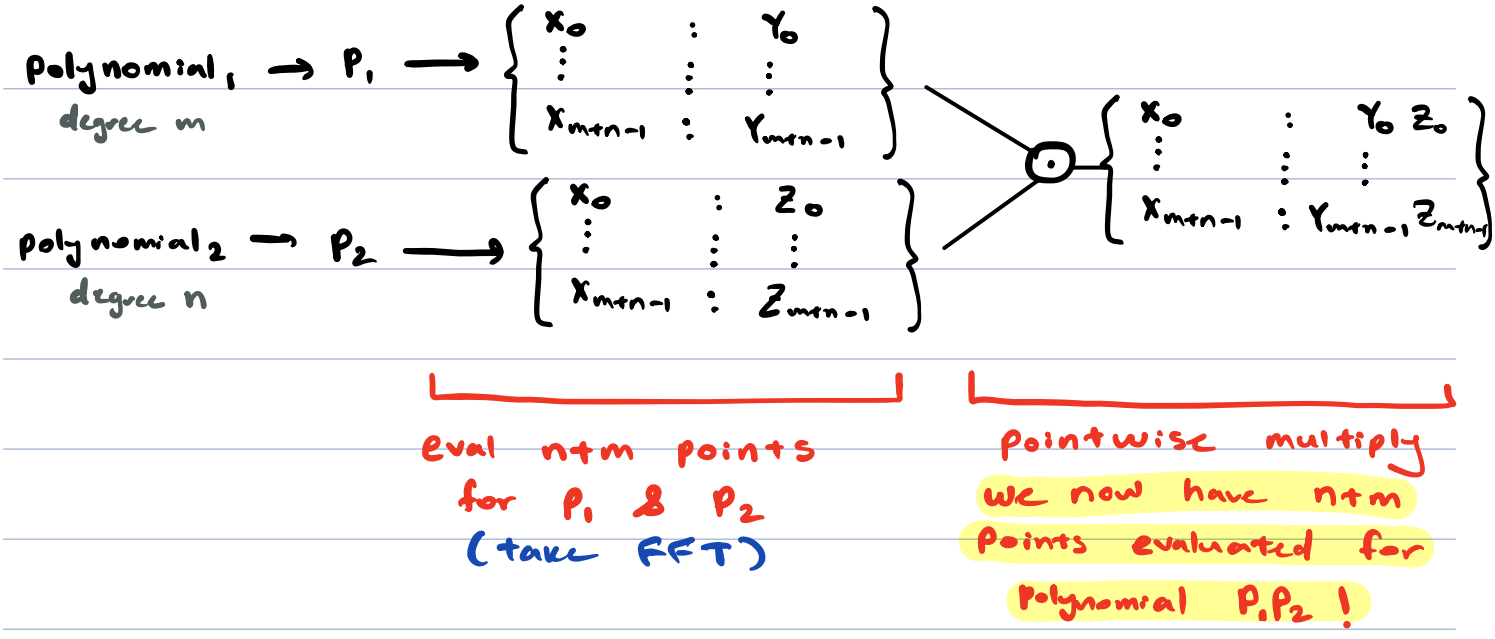
Our goal is to compute a convolution FAST.

1. Sliding window: slow!
2. Expanding out a polynomial is $O(n^2)$

we need a faster way of multiplying 2 polynoms!

Algorithm:

- important fact: given polynom. of degree n, it can be uniquely defined by n+1 points.



We have a polynom. in value-form: we need them in coefficient form!
Soln: take inverse FFT!

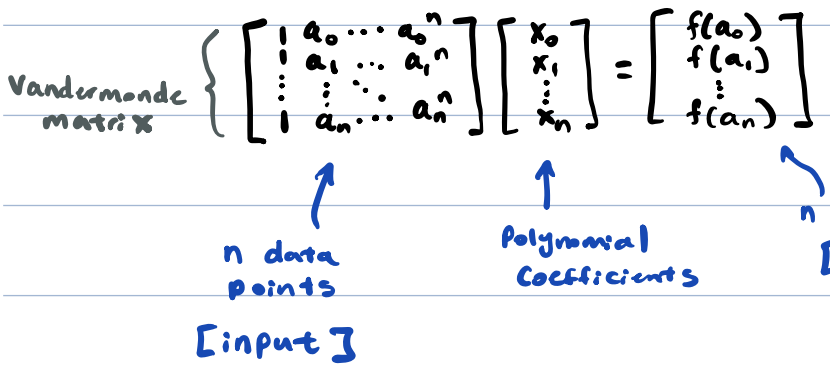
Polynomial 3

These coefficients are the Convolution results we want!

1) FFT the seq & filter — value-form
2) Pointwise multiply
3) Take iFFT — coefficient-form

The tricky part is doing the Coeff. → Value form transformation:

Naively (coefficient → value form):



(value → coefficient form):

solve $\vec{x} = A^{-1} \overrightarrow{f(a_i)}$

Takes
- $O(n^2)$ to invert this special matrix
- $O(n^2)$ to multiply w/ this vector

TRICK: by selecting special points, the FFT & iFFT lets you convert btwn these two forms in $O(n \log n)$ time!

★ perform long convs. fast!