BINGHAMTON UNIVERSITY
State University of New York

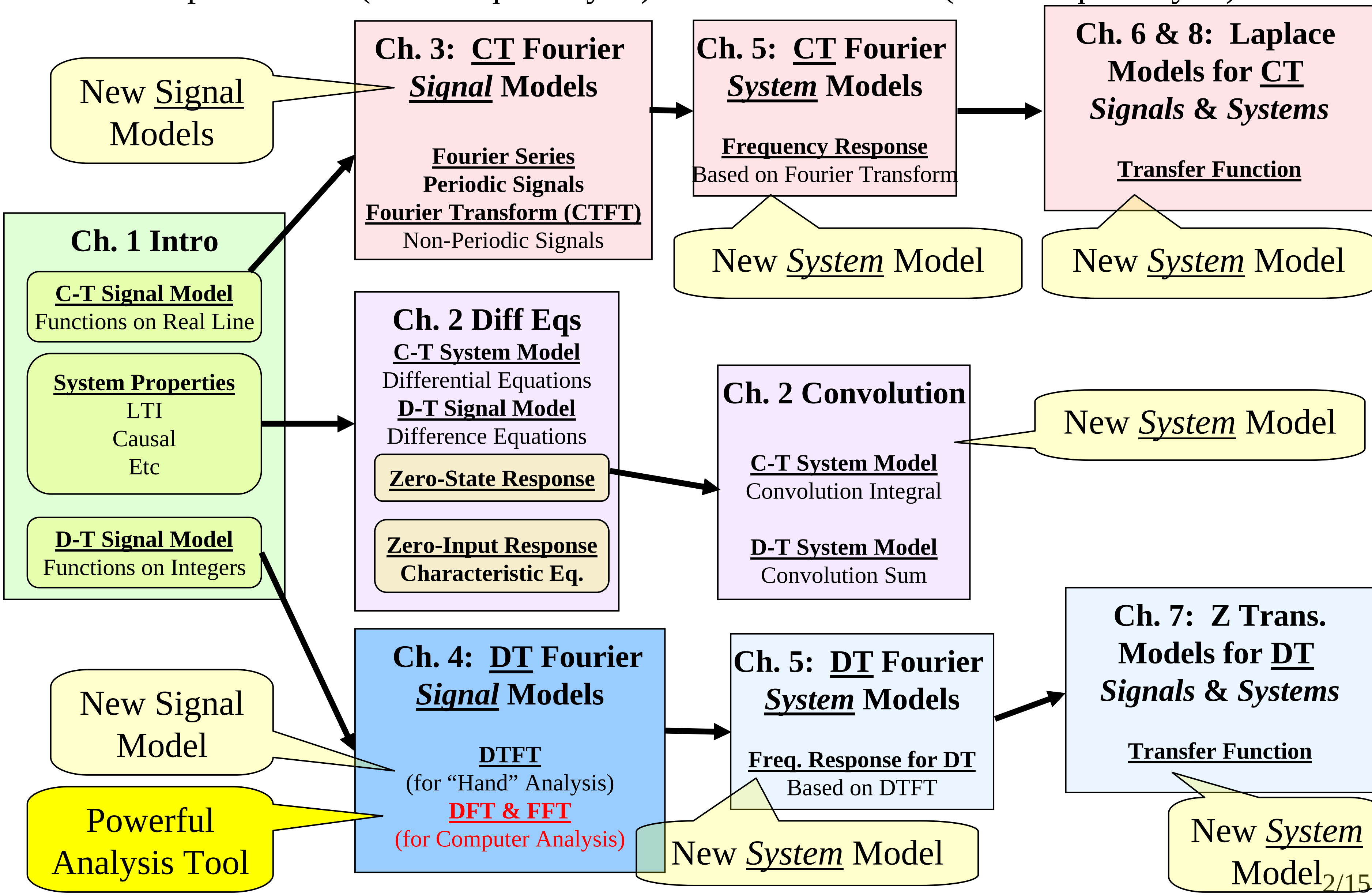
EECE 301
Signals & Systems
Prof. Mark Fowler

Note Set #24

- D-T Signals: Definition of DFT – Numerical FT Analysis
- Reading Assignment: Sections 4.2 of Kamen and Heck

Course Flow Diagram

The arrows here show conceptual flow between ideas. Note the parallel structure between the pink blocks (C-T Freq. Analysis) and the blue blocks (D-T Freq. Analysis).



4.2 Discrete Fourier Transform (DFT)

We've seen that the DTFT is a good analytical tool for D-T signals (and systems – as we'll see later when we return to Ch. 5)

Namely $X(\Omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\Omega n}$ (DTFT) can be computed analytically

(at least in principle) when we have an equation model for $x[n]$

Q: **Well… why can't we use a computer to compute the DTFT from Data?**

A: There are two reasons why we can't!!

1. The DTFT requires an infinite number of terms to be summed over n = …, -3, -2, -1, 0, 1, 2, 3, …
2. The DTFT must be evaluated at an infinite number of points over the interval $\Omega \in (-\pi, \pi]$

-The first one ("infinite # of terms")… isn't a problem if $x[n]$ has "finite duration"

-The second one ("infinitely many points")… is always a problem!!

Well… maybe we can just compute the DTFT at a finite set of points!!

Let's explore this possibility… it will lead us to the **D**iscrete **F**ourier **T**ransform

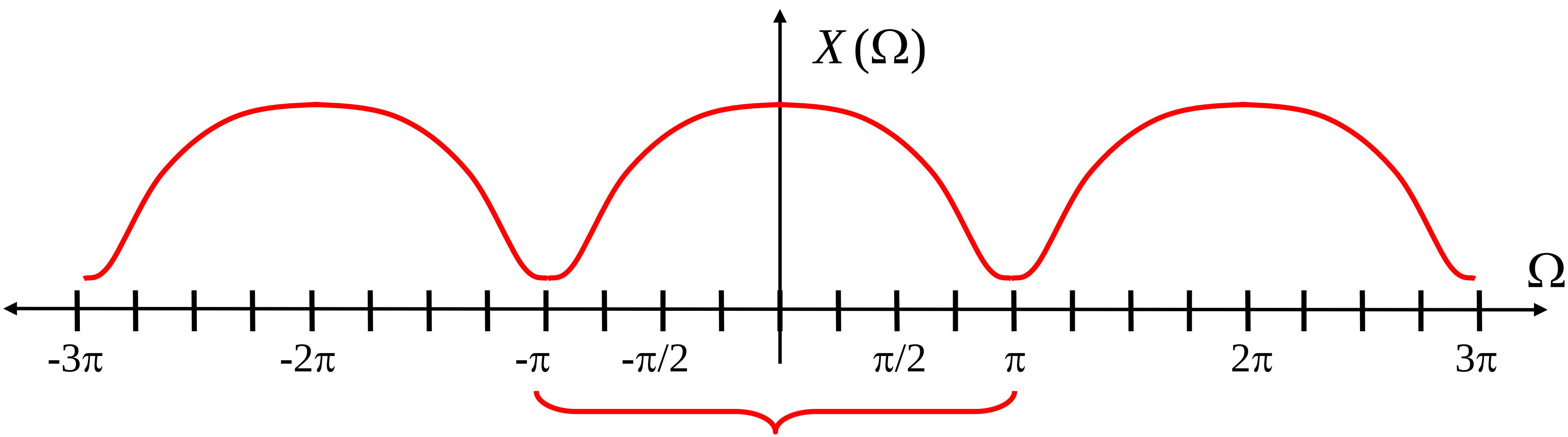
Suppose we have a finite duration signal: $x[n] = 0$ for $n < 0$ and $n \geq N$

Then the DTFT of this finite duration signal is:

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

we can leave out terms that are zero

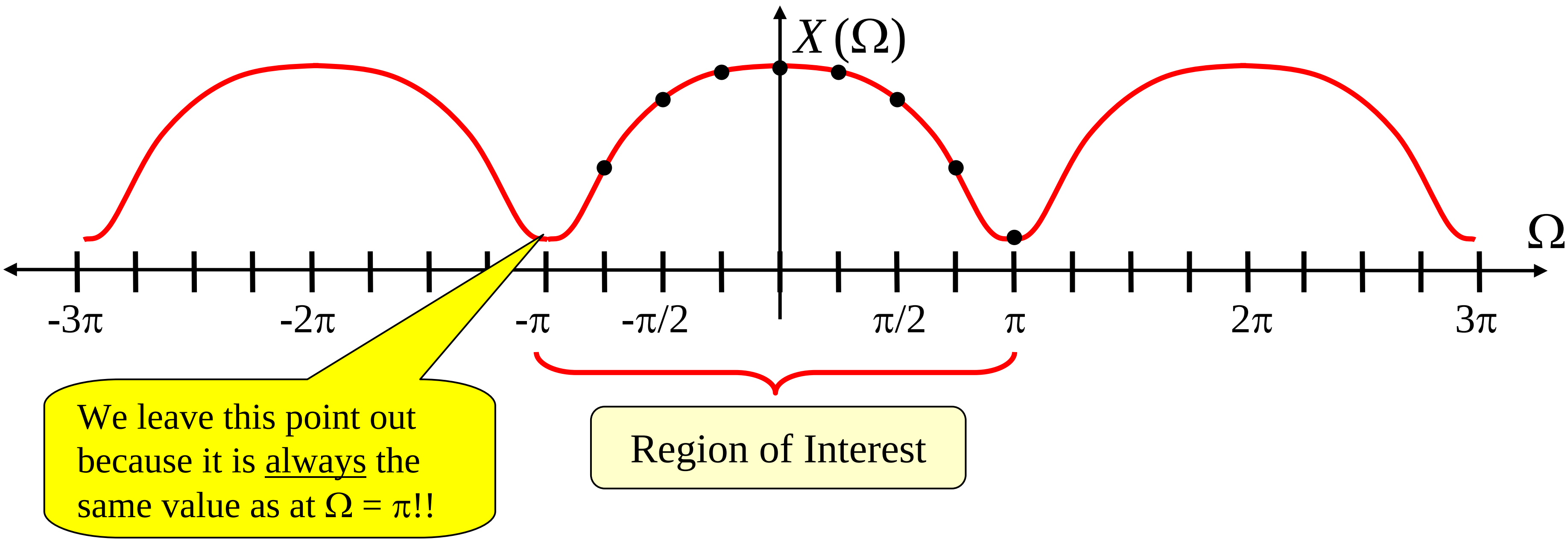
If we could compute this at every Ω value… it might look like this:



We are only interested in this range…
Everywhere else it just repeats periodically

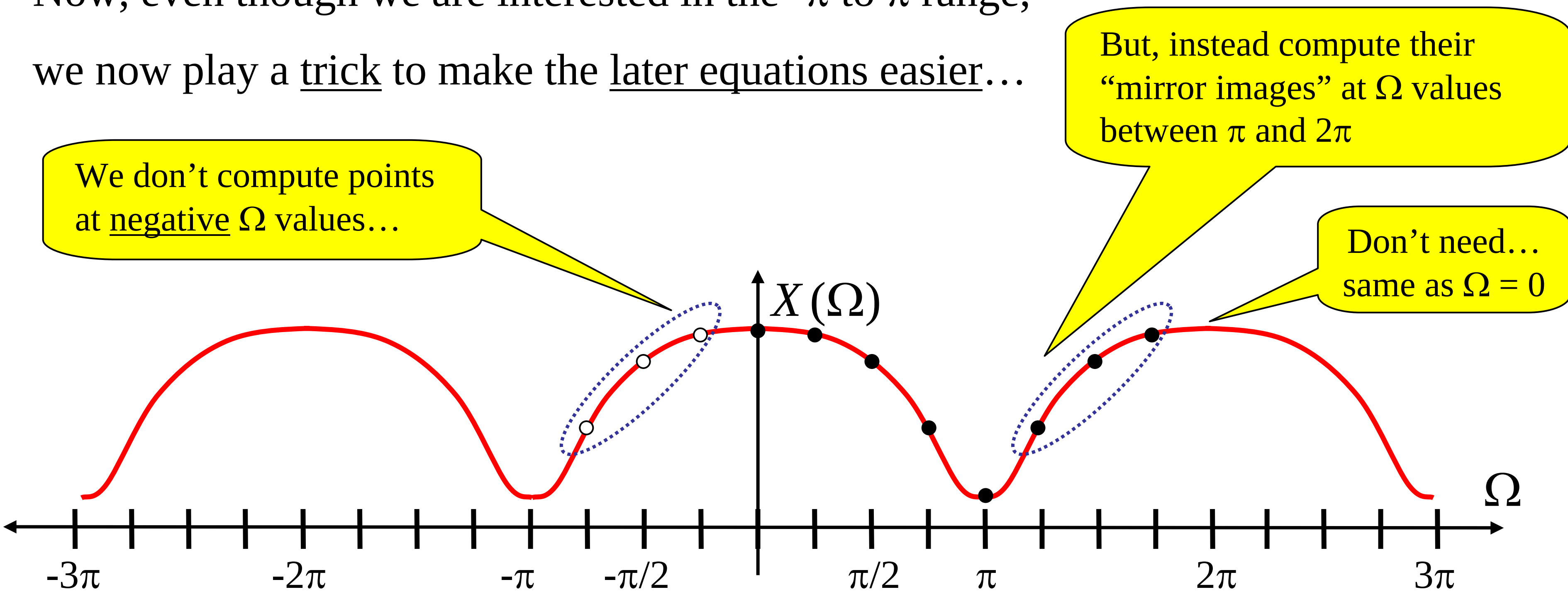
Now suppose we take the numerical data $x[n]$ for $n = 0, \ldots, N\text{-}1$

and just compute this DTFT at a finite number of Ω values (8 points here)…



Now, even though we are interested in the -π to π range,

we now play a trick to make the later equations easier…



So say we want to compute the DTFT at M points, then choose

$$\Omega_k = k\frac{2\pi}{M}, \quad for \;\; k = 0, 1, 2, \ldots, M-1$$

Spacing between computed Ω values

In otherwords:

$$\Omega_0 = 0, \quad \Omega_1 = \frac{2\pi}{M}, \quad \Omega_2 = 2\frac{2\pi}{M}, \quad \ldots \quad , \Omega_{M-1} = (M-1)\frac{2\pi}{M}$$

Thus… mathematically what we have computed for our finite-duration signal is:

$$X(\Omega_k) = \sum_{n=0}^{N-1} x[n] e^{-jn\Omega_k} = \sum_{n=0}^{N-1} x[n] e^{-jnk\frac{2\pi}{M}}, \quad \textit{for} \quad k = 0, 1, 2, \ldots, M-1$$

There is just one last step needed to define the **D**iscrete **F**ourier **T**ransform **(DFT):**

We must set $M = N$…

Done for a few mathematical reasons… later we'll learn a trick called "zero-padding" to get around this!

In other words: Compute as many "frequency points" as "signal points"

So… Given N signal data points $x[n]$ for $n = 0, \ldots, N\text{-}1$
Compute N DFT points using:

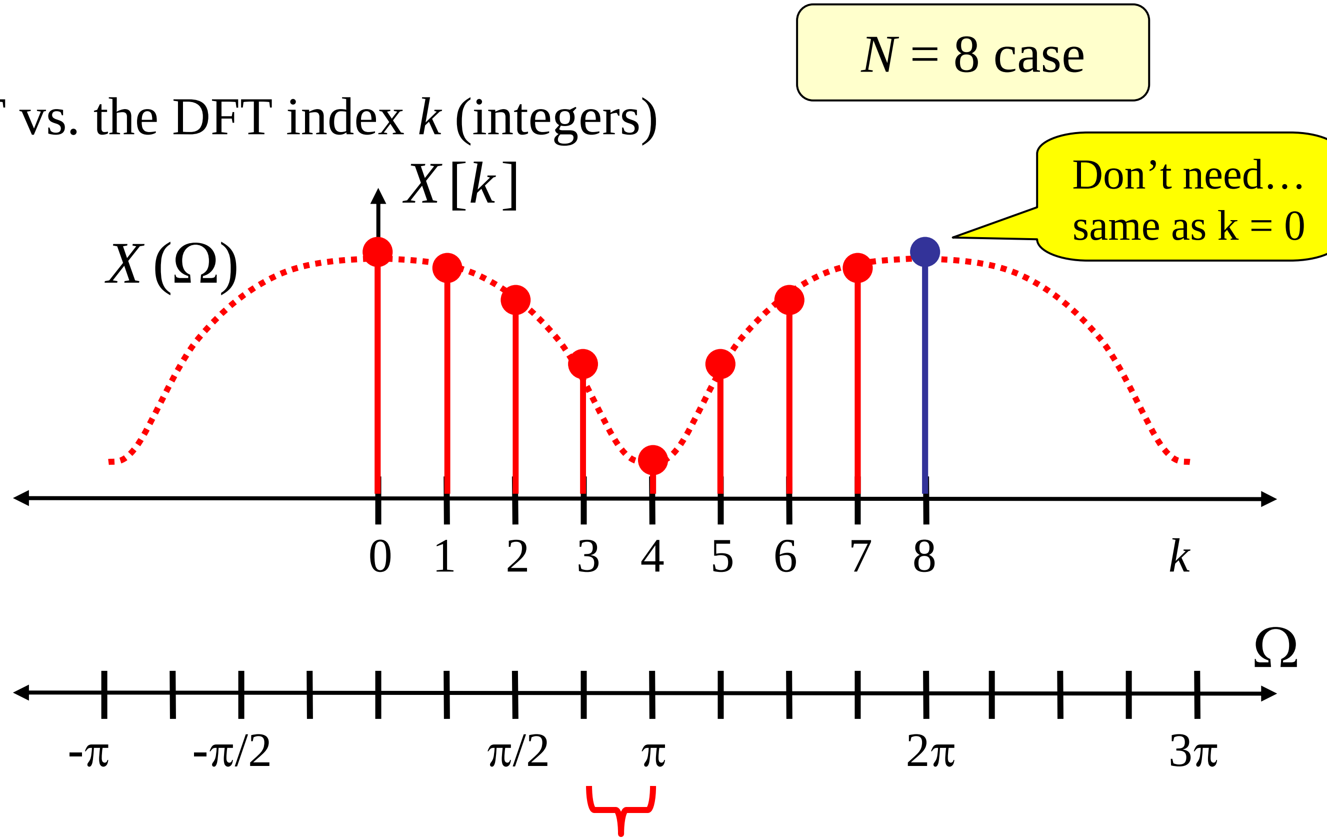
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \qquad k = 0, 1, 2, \ldots, N-1$$

Definition of the DFT

Book uses X_k notation

Plotting the DFT (we'll say more about this later..)

We often plot the DFT vs. the DFT index *k* (integers)



$$\frac{2\pi}{N} \Rightarrow \frac{2\pi}{8} = \frac{\pi}{4}$$

But… we know that these points can be tied back to the true D-T frequency Ω:

1. So far… we've defined the DFT

    a. We based this on the motivation of wanting to compute the DTFT at a finite number of points

    b. Later… we'll look more closely at the general relationship between the DFT and the DTFT

2. For now… we want to understand a few properties of the DFT

    a. There a more properties… if you take a senior-level DSP class you'll learn them there.

Properties of the DFT

1. **Symmetry of the DFT**

We arrived at the DFT via the DTFT so it should be no surprise that the DFT inherits some sort of symmetry from the DTFT.

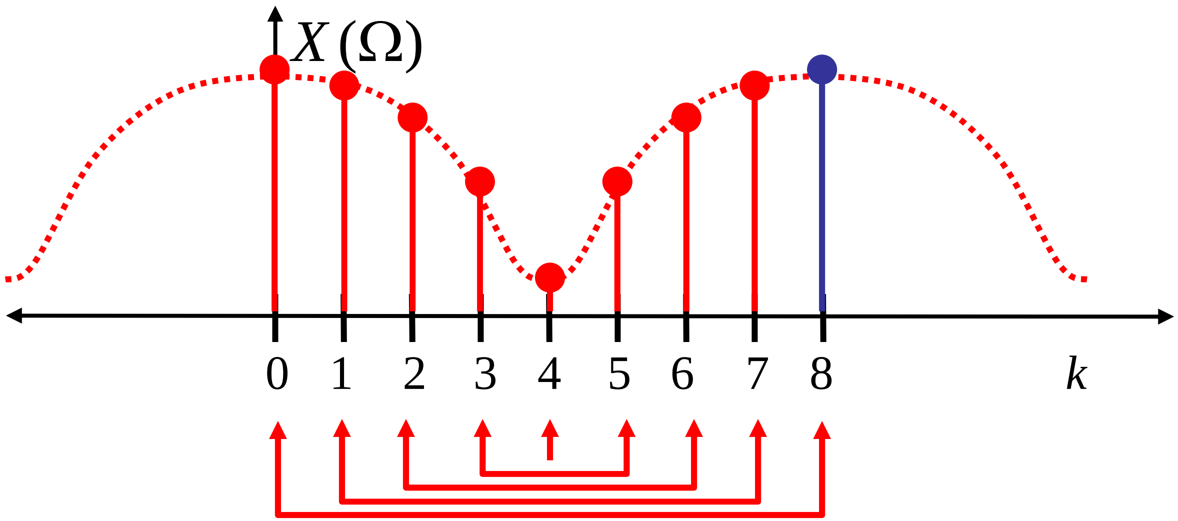
$$X[N-k] = \overline{X}[k], \quad k = 0,1,2,\ldots,N-1$$

Illustration of DFT Symmetry

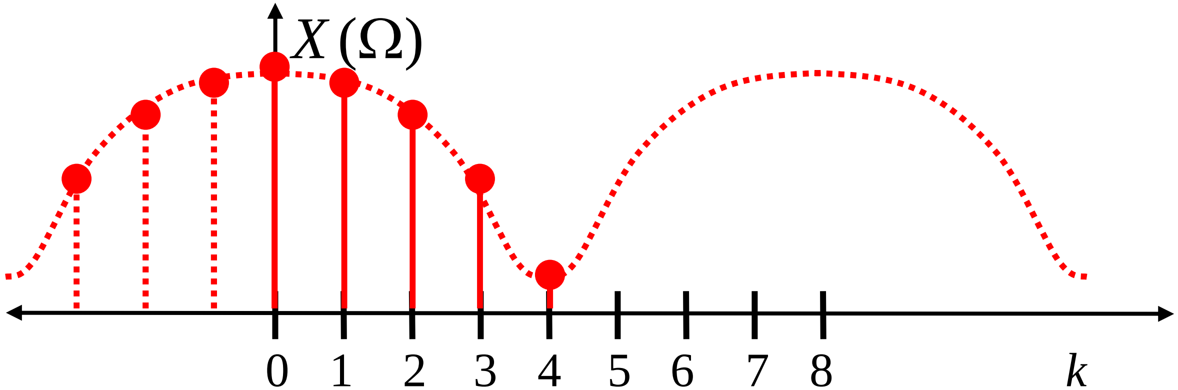
$$X[N-k] = \bar{X}[k], \quad k = 0,1,2,\ldots,N-1$$

In this example we don't see the effect of the conjugate because we made the DFT real-valued for ease

$N = 8$ case



Because the "upper" DFT points are just like the "negative index" DFT points… this DFT symmetry property is exactly the same as the DTFT symmetry around the origin:



2. **Inverse DFT**

Recall that the DTFT can be inverted… given $X(\Omega)$ you can find the signal $x[n]$

Because we arrived at the DFT via the DTFT… it should be no surprise that the DFT inherits an inverse property from the DTFT.

Actually, we needed to force $M = N$ to enable the DFT inverse property to hold!!

So… Given N DFT points $X[k]$ for $k = 0, \ldots, N$-1
Compute N signal data points using:

$$x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X[k]e^{j2\pi kn/N} \qquad n = 0,1,2,\ldots,N-1$$

Inverse DFT (IDFT)

Compare to the DFT… a remarkably similar structure:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \qquad k = 0,1,2,\ldots,N-1$$

DFT

FFT Algorithm (see Sect. 4.4 if you want to know *why* this works)

"FFT" = **F**ast **F**ourier **T**ransform

The FFT is not a new "thing" to compute (the DFT is a "thing" we compute)

The FFT is just an efficient algorithm for computing the DFT

If you code the DFT algorithm in the obvious way it takes:

about N^2 multiples, and N^2 additions.

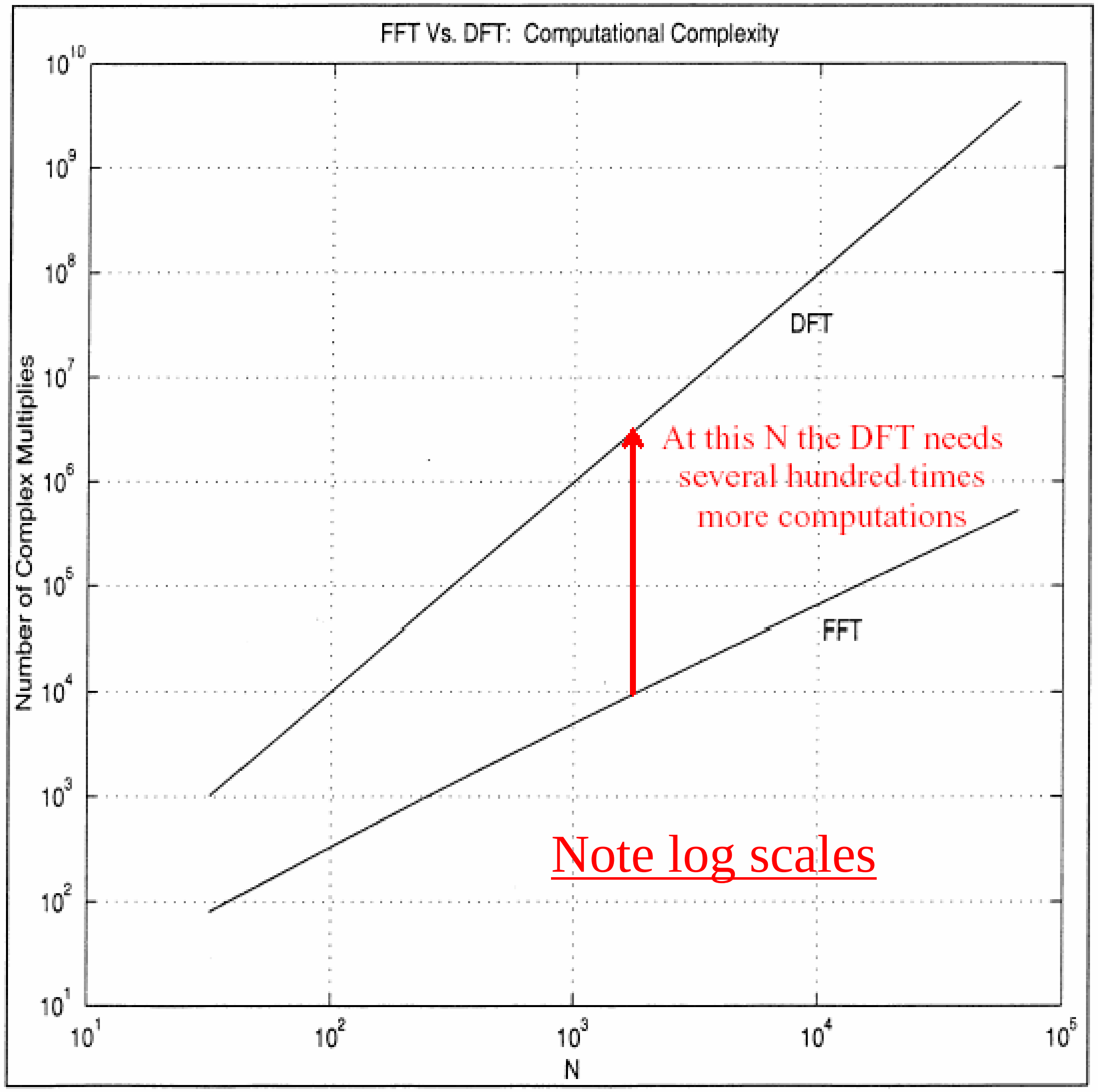
The FFT is a trick to compute the DFT more efficiently – it takes:

$$\text{about} \begin{cases} \dfrac{N}{2}\log_2(N) & \textit{multiples} \\ \dfrac{N}{2}\log_2(N) & \textit{additions} \end{cases}$$

Similar ideas hold for computing the inverse DFT.

Note: The most common FFT algorithm requires that the number of samples put into it be a power of two… later we'll talk about how to "zero-pad" up to a power of two!

The following plot shows the **drastic** improvement the FFT gives over the DFT:



While this figure tells the story… the following matlab example should really illustrate the dramatic improvement that the FFT provides for computing a 4096 point DFT:

Download the Notes_25_fft_test.m file and run it in matlab

When you run it…

1. You'll be prompted to hit Return…
   a. Several seconds after you hit Return you'll hear a beep that indicates that the direct DFT computation is complete
2. Then you'll be prompted to hit Return again
   a. Instantaneously as you hit Return you'll hear a beep that indicates that the FFT computation of the same DFT is complete!!

Without the FFT algorithm we would not be able to do the DSP processing fast enough to enable much of today's technology!!!

DFT Summary… What We Know So Far!

- Given N signal data points… we can compute the DFT
  - And we can do this efficiently using the FFT algorithm
- Given N DFT points… we can get back the N signal data points
  - And we can do this efficiently using the IFFT algorithm
- We know that there is a symmetry property
- We know that we can move the “upper” DFT points down to represent the “negative” frequencies…
  - this will be essential in practical uses of the DFT
  - Remember… we ended up with the “upper” DFT points only to make the indexing by k easy!!!
    - It is just to make the DFT equation easy to write!!

Now…

• We need to explore the connections between the DFT and the DTFT

• Then… understand the relation between the CTFT, DTFT, & DFT