**Fourier Transform:**

- A simple definition we can be saved to assume is the Fourier transform is a mathematical technique to turn a function of time x(t) into a function of frequency which can be represented in sinusoidal terms. Basically, an integral transform to make a problematic equation simpler. This could also include PDEs under a domain, making the solution a little easier to understand. In this brief gif, we do see a time domain of a function converting itself to function f hat in a frequency domain. The observer might be looking into amplitudes in which the first function has amplitude given time and amplitude given frequency for the second. We see 6 components approximating square waves turned into 6 sine waves which show peaks.

- **Slide 4** we see a graph that is f(x) = e^-|x|(Left most) which using Fourier transforms becomes the graph on the right. As you can see sinusoidal trend from what was a function of time to a function of frequency which we see the before and after of that function. Here is the equation which creates a function khat, and you can see an integral and the function f(x) being multiplied by e^+-ikx. Then if the math is done correctly the new function is f(x) = (1 + k^2)^-1 from the previous e^-|x|. The plus and minus create the inverse of the other so as long as we follow one of them we should be fine.

- One reason you'd use a Fourier transform is to identify patterns or cycles in a time series data, specifically to simplify noisy or complex data, especially with how much data everywhere has grown it becomes more evident to use something like this to make our understanding of certain cases better.
- We can see this continuous trend on the graph which in math is alright but for scientific computing/hi-performance rarely will you see a graph approach the inf.
- How does multiplying by e^+-ikx give us waves? E^i is a complex term with imaginary i that can be represented in sin and cosine. e^ix = cos(x) + isin(x) which is Euler's formula. The equation for Fourier requires some knowledge of complex numbers and calculus but we will be simply defining them for the most part.

**History( Slide 6 )**

- Brief: first used by the French mathematician Jean Baptiste Joseph Fourier (1768–1830) in a manuscript submitted to the Institute of France in 1807
- Fourier's idea/claim was that any function can be expressed as a harmonic series
- Fourier came upon his idea in connection with the problem of the flow of heat in solid bodies, including the earth
- He believed the signal can be decomposed into a sum of sine waves of different frequencies, phases, and amplitudes

**Discrete(Slide 8)**

- imagine a set of points from a sample interval

Fj = f(deltax)

Transform to n other values

- So integrating from -inf and +inf from the previous Fourier transform to now we just sum over from our set of points, which is what makes it Discrete. Fairly the same but a change in the integral to sigma for the set of n other values. This formula is also known as slow Fourier Transform.
- Matrix vector multiplication, where Fj would be the vector while j,k values can be seen as a matrix, in which we would use Linear algebra and BLAS routine from C++ to perform it. The problem is that the name persists it takes O(n^2) to perform which is SLOW. If we take one point of this course it's to be efficient fast and time-saving as possible like modularity, header files, prototyping, etc.

**Code Slide 9**

***TYPEdef defining wherever complex is mentioned it wants complex double from complex library**

- This is an example code done by Scinet, using the complex that holds both real and imaginary to perform this transform. We see a function which takes f vectors, that is the output vectors(our function Fourier transformed) and inverses the sign for either forward or backward trend. So we have two four loops to get every element and use the equation in which there is in cos and sin but is the same as E^i. Now Gauss a mathematician used this work to find Discrete Fourier transform and did notice it was slow in which he found another way, written in a diary that Cooley and Tukey in 1965(a large period gone), Fast fourier transform

**Fast Slide 10**

- Fast Fourier Transform! The Idea is to write each n point Fourier tranform as sum of two n/2 point, basically divide it by 2 parts, which must be done in ^2logn(power of 2 is the spilt in 2 parts) levels of recursion which each level now is O(nlogn) instead of O(n^2) for large amounts of data. The beauty in our field of research for using this technique is that there are libraries available that are already able to perform this such as FFTW3, Intel MKL, IBM ESSL.

**Code Slide 11**

- So an updated version of the slow to fast. We can see the new library fftw3.h. What's cool here is we have to create a plan which is needed when using this library. So we create a plan with the number of data points and perform it on execution, then destroy the plan once performed. F.data is a pointer basically saying it points to a complex number, that being our output again, forward backwards and now we can estimate what the fastest way or measure, but only really good if we performing the plan multiple times. The idea of Plans in our code is they can be reused and even saved on disk.

**Difference between continuous and discrete**

- There are continuous-time Fourier transforms as well as discrete-time Fourier transforms. The most basic idea is that continuous is for function x(t) where t belongs in the R is a continuous variable, whereas discrete is sequences of x[n] where n belongs in the complex domain, finite number of n. This may explain more why Fourier is an integral and discrete is a sigma defined by sum.

**Applications Slide 13**

- The FT has long been proved to be extremely useful as applied to signal and image processing and for analyzing quantum mechanics phenomena. For signal and image processing imagine a crowd of people, which we know is made of people, and we know the standard human looks like with 2 arms, legs etc. If we wanted to go further we can find specific faces in the crowd that we may have already seen, which is the core idea of signal and image analysis. Finding and demonstrating known features from a large pool of data, which scientific/hi-performance make this much easier to be done and the tool you might expect is the Fourier transform to isolate and show these certain features. Image processing goes into using these waves to show these images in a low pass(above a select frequency) and high pass filters(bellow a select frequency).
- Now in quantum mechanics, Fourier transform is used for position and momentum space rather than time and frequency as we did before. Helped to identify a quantum system better by using eigenfunctions in a linear combination and having a wave function. Hi-performance computing here can help demonstrate the quantum level of this distribution and help identify trends and could further quantum mechanics.
- If you didn't know, the Fourier transformation is almost anywhere to everywhere, from phones, hospitals, headphones etc, from a concept that been used now that was made a long time ago still is relevant and present in our daily lives. For our field of research, this transformation allows a deeper understanding and connects us with other streams of sciences and efficiency.

.