slide1:
Welcome, everyone. Today, we're diving into the topic of signal processing, which is a key concept in this course and incredibly important in the field of biomedical imaging.
Some of you may have already gone through the chapter in the textbook — and that's a great start. But as I've mentioned before, the draft versions of these chapters may still contain a few typos or unclear steps. That's why it's important not just to read, but also to listen carefully during the lecture.
I've reviewed and corrected the derivations we'll go over today, so what you see here should be more accurate and easier to follow than the original draft. This is also a good example of why attending the lecture — or in this case, watching this video — is still valuable, even if you've read the chapter. I'll walk you through the ideas step by step, explain the reasoning behind each move, and help you build a stronger intuition than what the written text alone might offer.
So after this session, I encourage you to review both the lecture and the textbook. With both perspectives, your understanding will be much deeper — and that's the goal.
Let's get started.

slide2:
And again, we are on schedule. And so far, we have learned the linear systems and learned the Fourier analysis.

slide3:
Let me show you a logo I designed to help you understand the foundation of signal processing in a visual way.
Inside the circle, you'll see the expression: delta multiplied by e to the power i theta.#That's delta times e to the i theta.
This might remind you of Euler's formula, where e to the power i theta equals cosine theta plus i times sine theta.
So this single term includes both a real part — the cosine — and an imaginary part — the sine.
Now, the delta function here stands for an impulse — something that happens instantly, at a single point in time.
Together, the delta and the exponential show us two building blocks we'll come back to again and again: sharp spikes and smooth waves.
But that's not enough to build real-world signals.
To represent a general signal, we also need to be able to shift and scale these building blocks.
Shifting means moving a function left or right in time.#Scaling means stretching or compressing it, or changing its amplitude.
For example, with sine waves, we can change the frequency — making them faster or slower.#We can also shift them in time — or scale the height — to make them louder or softer.
So when we say "operators need to shift and scale," we mean we need to move and adjust these basic functions — the delta and the sine wave — to create more complex signals.
And with that power, we can build up any continuous or piecewise continuous function.
That's the big idea behind this logo — a visual summary of how signal processing works at its core.

slide4:
Now let's talk more specifically about how a function can be represented using the Fourier series or the Fourier transform.
At the top, we see a Fourier series representation. It tells us that a function of t — written as f of t — can be expressed as a sum over many terms.
Mathematically, this is written as:
f of t equals the sum from n equals negative infinity to positive infinity, of c n times e to the power 2 pi i n t over capital T.
Each term in the sum is a complex exponential — and each coefficient, c n, tells us how much of that particular frequency component is present in the signal.
Now how do we calculate these coefficients?

We use the formula:
c n equals 1 over capital T, times the integral from 0 to capital T, of f of t multiplied by e to the power negative 2 pi i n t over capital T, d t.
So what we're doing here is projecting the function f of t onto a basis function — that basis is the complex exponential. This is very much like taking an inner product between vectors.
You can think of the function as living in a high-dimensional space — in fact, infinitely many dimensions. Each basis function spans one of those directions.
So when we compute c n, we're measuring how much f of t points in the direction of that basis function.
Now, what if the function isn't periodic, or the period becomes very large?
That's where the Fourier transform comes in.
Instead of summing over discrete frequencies, we move to a continuous frequency variable, often called s.
The Fourier transform of f of t is written as f hat of s, and it's defined as:
The integral from negative infinity to positive infinity, of f of t times e to the power negative 2 pi i s t, d t.
To recover the original function, we use the inverse Fourier transform, which is:
f of t equals the integral from negative infinity to positive infinity, of f hat of s times e to the power 2 pi i s t, d s.
So you can see — when we move from a periodic function with period capital T, to a general function as T approaches infinity, the Fourier series becomes the Fourier transform.
This is a very elegant transition — from a discrete sum to a continuous integral — and it forms the basis of much of signal processing.

slide5:
Now that we've covered the basics of the Fourier series and transform, let's talk about a very important result: the Convolution Theorem.
This theorem creates a beautiful bridge between two worlds — the time domain and the frequency domain.
Here's what it says:#Convolution in the time domain becomes multiplication in the frequency domain.
In simple terms, if you have a linear, time-invariant system, the output is the convolution of the input signal with the system's impulse response.#But instead of computing that convolution directly — which can be messy — you can switch to the frequency domain, where it becomes much easier.
So, let's say:
f of t is your input signal,
g of t is the system's impulse response,
and f star g means the convolution of f and g.
Then, in the frequency domain, their Fourier transforms — call them F of s and G of s — satisfy this simple rule:
The Fourier transform of the convolution equals the product of the individual Fourier transforms.
Or in math:#Fourier of f star g equals F of s times G of s.
That means you now have two ways to compute the system's output:
You can do convolution in the time domain.
Or — you can transform both the input and the impulse response into the frequency domain, multiply them there, and then take the inverse Fourier transform to go back to the time domain.
Now, let's look at a simple example.
Suppose you convolve two rectangular functions, also called gate functions. Each one looks like a block or a box.
To compute the convolution, you flip one function, shift it, multiply the overlapping parts, and add the result.
When the two rectangles perfectly overlap, the area under the curve reaches its maximum. As they move apart, the overlapping area decreases linearly — and what you get is a triangle-shaped function.
So, two rectangles convolved give you a triangle.
Now let's look at this in the frequency domain.
We know the Fourier transform of a rectangular function is a sinc function — that's sinc of s, which comes from sine of pi s over pi s.

So when we convolve two rectangles in the time domain, we get a triangle. That means, in the frequency domain, we multiply two sinc functions — and we get sinc squared.
So: Triangle in time domain means sinc squared in frequency domain.
This is a powerful example that helps you visualize the convolution theorem in action. It shows how the shapes in time relate to the shapes in frequency — and why this theorem is such a key tool for signal processing and systems analysis.
Keep this visual and mathematical link in mind — it will come up again and again.

slide6:
Now, let’s step back for a moment and ask a deeper question: Why does the convolution theorem work the way it does? What makes it so special?
In the last lecture, I mentioned something that gives us a clue.#If you have a shift-invariant linear system, and you input a sinusoidal signal, what comes out is also a sinusoid, and — importantly — at the same frequency.
That’s a big deal.
It means the system doesn't generate any new frequencies. It doesn’t distort the sinusoid into some other shape. It simply scales it or shifts its phase — but the frequency stays the same.
This is a unique and powerful property of sinusoids.
Now let’s think about what that means for convolution and Fourier analysis.
In time-domain analysis, convolution is the operation that describes how a system responds to an input.
But if your input and your system are both described using sinusoids, and those sinusoids don’t change frequency, then it turns out convolution becomes multiplication in the Fourier domain.
Why?
Because in the Fourier domain, every signal is broken into sinusoids. And if each one stays intact — same frequency in, same frequency out — then multiplying their responses is all you need to compute the system output.
That’s exactly what the convolution theorem says:#Convolution in time equals multiplication in frequency.
But — and this is important — this beautiful relationship only works because of the special behavior of sinusoids.
If your input signal isn’t made of sinusoids, or your system doesn’t treat them nicely, this property breaks down.
That’s why this theorem only exists for the Fourier transform, which is based entirely on sinusoids.#Other transforms — like wavelets or polynomials — don’t have this same "in equals out" behavior, and so they don’t give us a convolution theorem in the same way.
If you’re curious, I actually encourage you to think about this more. You could even write a short paper or a reflection on it.
Here’s something to try:
Try proving that if a function goes into a system and the same form comes out — unchanged except for amplitude and phase — then that function must be a sinusoid.
And also prove the related idea:#If sinusoids go in and come out with the same frequency, then convolution in time must match multiplication in frequency.
These are two different ideas — but together, they give us the foundation for the convolution theorem and explain why Fourier analysis is so powerful for linear systems.

slide7:
Let’s now walk through an example to help us better understand why the convolution theorem works — especially when sinusoids are involved.
We’ll use a one-dimensional, shift-invariant linear system.#In such a system, the output — which we’ll call o of t — is the convolution of the system’s impulse response h of t with the input signal, i of t.
So we write:
o of t equals h of t convolved with i of t.
According to the convolution theorem, when we take the Fourier transform of both sides, this convolution becomes multiplication.
So we have:

capital O of u equals capital H of u times capital I of u,#where capital O, H, and I are the Fourier transforms of the output, impulse response, and input, respectively, and u is the frequency variable.
Now, let’s suppose we feed the system a complex sinusoidal input.#That is:#i of t equals e to the power i 2 pi u naught t,#where u naught is the frequency of the sinusoid.
As we learned earlier, the Fourier transform of a pure sinusoidal signal is a delta function.#So, in the frequency domain:
capital I of u equals delta of u minus u naught.
This delta function tells us the signal contains only one frequency — u naught — and no others.
Now, using the convolution theorem:
capital O of u equals capital H of u times delta of u minus u naught.
Multiplying anything by a delta function simply picks out the value at that point.#So this simplifies to:
capital O of u equals H of u naught times delta of u minus u naught.
This tells us that the output, in the frequency domain, also consists of a single frequency — the same one as the input — and it’s just scaled by the system’s transfer function at that frequency, which is H of u naught.
Now we go back to the time domain.
If the Fourier transform of the output is a scaled delta function in frequency, then the time-domain output must also be a sinusoid — at the same frequency.
So, we have:
o of t equals H of u naught times e to the power i 2 pi u naught t.
This shows that the output is still a complex sinusoid — same frequency as the input — but scaled by a factor, H of u naught, which may include amplitude and phase changes.
The key idea here is:
The frequency doesn't change.
No new frequencies are generated.
The output stays sinusoidal if the input was sinusoidal.
This confirms that a shift-invariant linear system preserves frequency when the input is a pure sinusoid.
And we were able to demonstrate this clearly and quickly — thanks to the convolution theorem and the properties of the Fourier transform.
It’s a great example of how math gives us not just a result, but deep insight into how signals behave in systems.

slide8:
Let me now show you the other direction of this idea — and this time, we’re not going to rely on the convolution theorem directly.
We’ll start with a simple question:#Can any function other than a sinusoid go into a system and come out unchanged in shape — just scaled by a constant?
Let’s think about that.
Suppose the impulse response of the system is a delta function. Then, any signal you put into that system just passes through exactly as it is — unchanged. The system does nothing.#This is an ideal system — it responds instantly and perfectly.
For example, if a camera had a perfect delta-function impulse response, every point in the scene would be captured exactly — with perfect sharpness, no blur at all.
But in real life, that doesn’t happen. No camera has a true delta response.
In practical systems, the impulse response is not a delta function — it’s more spread out. That means the system blurs things.#Even if you feed it a sharp impulse, the output will be a wider, smeared version — not the same as the input.
So what does that mean in the frequency domain?
Well, we know that the Fourier transform of a delta function is a constant — it's just one everywhere.#That’s because a delta contains all frequencies equally.
But if your impulse response is not a delta — if it’s more spread out — then its Fourier transform won’t be constant.#It will vary in frequency. That’s key.
Now, let’s suppose you have some input signal — let’s call it i of t — and this signal goes into the system.#You get an output o of t, and let’s say the output

has the exact same shape as the input, just scaled by some number alpha.
That is:
o of t equals alpha times i of t.
That sounds simple, but let’s look at what it means in the frequency domain.
If we take the Fourier transform of both sides, we get:
Capital O of u equals alpha times capital I of u.
But from the convolution theorem, we also know:
Capital O of u equals capital H of u times capital I of u.
If we compare those two expressions, we find:
Capital H of u must equal alpha — a constant.
But as we just said, that would only be possible if h of t — the impulse response — is a delta function.#And we assumed it's not.
So we have a contradiction.
That tells us:#No function other than a sinusoid can go through a general shift-invariant linear system and come out with the same shape — unless the system is trivial, like an ideal delta.
Only sinusoids have this amazing property:#Sinusoid in, sinusoid out — same frequency, just scaled and shifted in phase.
That’s what makes them so special in Fourier analysis.#That’s why convolution in the time domain becomes multiplication in the frequency domain — because the system affects each sinusoidal component independently and proportionally.
So what you see here is not just a bunch of equations — it’s a deeper logic behind the theorem.#Understanding this helps you move beyond just memorizing formulas — it helps you truly see why Fourier analysis works the way it does.

slide9:
Let’s now turn to a beautiful result in Fourier analysis called Parseval’s Identity.
This identity tells us that the total energy of a signal — measured in the time domain — is exactly equal to the total energy in the frequency domain.
Mathematically, it says:
The integral of the absolute value squared of f of t, over all time, equals the integral of the absolute value squared of f hat of s, over all frequencies.
In simpler words:#If you take a signal and square its amplitude at each point in time, then add up all those values — you get the total energy of the signal in the time domain.
Now do the same thing in the frequency domain:#Take the Fourier transform of the signal, square the magnitude of each frequency component, and integrate — and you get the same total.
This is not just a coincidence. It’s a direct consequence of the fact that the Fourier transform is a unitary operator — it preserves the “length” of the function, just like a rotation in vector space.
Think of f of t and f hat of s as infinite-dimensional vectors.#Then, this identity is telling you that their norms — or their lengths — are the same.#So, the energy stays the same when you move from one domain to the other.
This is why we sometimes say that the Fourier transform conserves energy.
The bottom part of the slide walks through the proof. Let me summarize the key steps:
We start with the inner product:
Integral of f of t times the complex conjugate of g of t.
Then, we apply the inverse Fourier transform to g of t.#This gives us an expression involving f of t and the Fourier transform of g, which we call F g of s.
Then we do some algebra — using properties of complex conjugation, and switching the order of integration — until we end up with:
The integral of f of s times the complex conjugate of g of s.
So in the end, the inner product in the time domain equals the inner product in the frequency domain.
That’s Parseval’s Identity in action — and it works for all functions in an inner product space, not just in signal processing.
It’s one more reason why the Fourier transform is not just a clever trick — it’s a deep and powerful tool grounded in geometry and physics.
If you’re interested, you can think of it this way:#Every time you transform a signal into the frequency domain, you’re rotating it into a new basis — and the

energy stays exactly the same.

slide10:
Let’s now apply Parseval’s theorem to a practical example — the average power of a periodic signal.
Here we’re working with a power signal, which is a signal that keeps oscillating over time, like a sine wave.
The average power P of such a signal over one period — from minus capital T over 2 to plus capital T over 2 — is given by:
P equals 1 over T, times the integral of the absolute value of x of t squared, d t.
According to Parseval’s theorem for power signals, we can also express this using the Fourier series coefficients.#So, power equals the sum of the absolute value squared of all c n — the Fourier coefficients — from n equals minus infinity to infinity.
Let’s go through the example.
Suppose we have this signal:
x of t equals 2 times sine of 100 t.
That’s just a sinusoidal wave with amplitude 2 and frequency 100.
Now, suppose this signal is the voltage across a unit resistor — that is, a resistor with resistance equal to 1 ohm.
Then the instantaneous power is just the voltage squared, and the average power is the integral of the squared signal over one period, divided by T.
Let’s calculate it first in the time domain:
We square the signal:#2 times sine of 100 t becomes 4 times sine squared of 100 t.
When we compute the average of that over one period, we get:
P equals 2.
Now let’s do it in the frequency domain, using the Fourier series.
We rewrite the sine wave using Euler’s formula:
sine of 100 t equals e to the j 100 t minus e to the minus j 100 t, over 2 j.
So our signal becomes:
x of t equals 2 times that expression, which simplifies to:
minus j e to the j 100 t, plus j e to the minus j 100 t.
Comparing this to the general form of a Fourier series:
x of t equals the sum of c n times e to the j n omega naught t,#we can identify the coefficients:
c 1 equals minus j, and#c minus 1 equals plus j.
Now we apply Parseval’s theorem:
The total power is the sum of the squares of the magnitudes of these coefficients.
So we compute:
absolute value of c one squared, which is 1,#plus absolute value of c minus one squared, also 1.
Adding them gives us 2 — the same result we got in the time domain.
This example shows how Parseval’s theorem works in practice.
You can either:
Square the signal and average it over time,#or Decompose the signal into its sinusoidal components, square their coefficients, and sum them up.
Either way, you’ll get the same result. That’s the beauty of Fourier analysis — it gives you two consistent views of the same signal.
And this works not just for simple sine waves — it holds even when a signal is made up of many frequencies, including very high ones.#Just square each coefficient, add them up, and you’ll know the total power of the signal.

slide11:
Let’s take a moment to review this important concept — how a continuous function can be represented using delta functions.
You may remember that the delta function plays a central role in signal processing.
And here's the key idea:#When you multiply a delta function with a continuous function f of t, and integrate over all time, the result is just the value of f at the point where the delta is centered.
Mathematically, we write:

The integral from minus infinity to infinity of delta of t minus t naught, times f of t, d t — equals f of t naught.
This tells us that the delta function samples the value of f at t naught.#It picks out that one value and ignores everything else.
Now, you might wonder how this works — so here's an intuitive explanation.
Think of the delta function as the limit of a very narrow and very tall pulse — getting narrower and taller, but always keeping the same area, which is 1.
So if we take that narrow pulse, center it at t naught, and multiply it with f of t, we get a tiny slice of the function near t naught.
When we integrate over that narrow region, the result becomes:
f of t star, where t star is some point very close to t naught.
And in the limit — as the width goes to zero — that becomes exactly:
f of t naught.
So this is how we record a sample.
It also leads us to a deeper interpretation:
We can reconstruct the entire function f of t by summing up all these infinitesimal slices — each one represented by a delta function, placed at the right location, and scaled by the value of the function at that point.
That's what this second formula is saying:
f of t equals the integral from minus infinity to infinity of delta of tau minus t, times f of tau, d tau.
It's a kind of weighted sum of deltas — where the weights are just the values of the function itself.
You can think of this as a very fine-grained, particle-like view of a continuous function.#Each point — each "particle" — contributes through a delta function, and together, they rebuild the original signal.
This is not just a mathematical trick — it connects deeply with how we sample, record, and reconstruct signals in the real world.
And once again, it highlights why the delta function is so important — both in theory and in practice.

slide12:
Let's revisit the convolution theorem — this time with a concrete visual example to help reinforce the idea.
This theorem is incredibly important, and we'll keep coming back to it.#So it's essential to get comfortable with how it works — both mathematically and visually.
Let's look at what we have here.
In the top row, we see three time-domain signals:
On the left is f of x, a simple rectangular function, also known as a gate function, centered at zero and extending from minus b over 2 to plus b over 2.
In the middle is h of x, which consists of two delta functions — located symmetrically around zero at minus a over 2 and plus a over 2.
On the right is the result of the convolution of f and h — which we'll call g of x.
Now remember:#When you convolve a function with a delta, the result is simply a shifted version of that function.
So here, convolving the gate function f of x with each of the two deltas in h of x gives us two shifted gate functions — one shifted left and one shifted right.
When we add them together, we get g of x — a pair of rectangular pulses.
Now let's look at the bottom row — this is the frequency domain.
Here's where the convolution theorem comes in.
According to the theorem:
Convolution in the time domain becomes multiplication in the frequency domain.
So we take the Fourier transforms of the three signals:
f of x becomes capital F of k, which is a sinc-shaped curve — smooth and localized in frequency.
h of x, which had two deltas in time, becomes a cosine wave in the frequency domain — that's capital H of k, centered at zero, with oscillations depending on the spacing of the deltas.
Now, to get capital G of k, we multiply the two frequency-domain functions:
Capital F of k times capital H of k equals capital G of k.
And the result, shown on the right, is a modulated sinc — essentially the original sinc shape, but with oscillations introduced by the cosine multiplier.

So the key takeaway is this:
In time, the convolution added and shifted the pulses.
In frequency, the multiplication mixed the smooth sinc shape with oscillations.
This is a perfect visual illustration of what the convolution theorem does:
It transforms an operation that involves sliding and summing in time into a much simpler pointwise multiplication in the frequency domain.
Understanding this duality will help you immensely when working with signals — whether in analysis, filtering, or system design.

slide13:
We've now come to a natural turning point in our discussion.
So far, everything we've studied — from Fourier series and Fourier transforms, to convolution and delta functions — has been in the continuous domain.
But now we need to ask a very important question:#Why do we work with digital signals? Why go digital?
Well, here are some good reasons.
First — digital signals are exact.#They're stored as numbers, so they don't degrade over time the way analog signals do.
Second — errors can be detected and corrected.#Digital systems can automatically check for mistakes during transmission or storage — and even fix them.
Third — noise and interference can be filtered out more easily.#In a digital system, it's much simpler to distinguish real data from unwanted signals.
Fourth — with digital transmission, we can send multiple types of information over the same line — audio, video, text — all bundled together efficiently.
And finally — digital systems support data compression, which lets us send more information using less bandwidth.
All of this is why most modern communication, computation, and imaging systems have moved to the digital domain.
And that brings us to the next part of our journey.
Let's study how to process digital signals next.

slide14:
Now that we've seen why digital signals are so useful, let's take a look at how real-world signals actually make their way into a computer.
This diagram shows the full journey — from a physical system to digital data inside your laptop.
Let's walk through it step by step.
It all starts with the physical world — maybe we're measuring temperature, pressure, light intensity, motion, or some other physical quantity.
To capture this, we need a sensor — or more specifically, a transducer.#A transducer converts the physical phenomenon into an electrical signal — usually an analog one.#This analog signal may be noisy, or it may contain extra fluctuations we don't want.
So the next step is called signal conditioning.#Here we might filter the signal, smooth it, amplify it, or perform other adjustments to get it into a clean, usable form.#This step helps us reduce noise and make sure the signal is suitable for the next stage.
Now comes the critical component — the Analog-to-Digital Converter, or A-D converter for short.
This is where the continuous analog signal gets sampled and turned into a digital signal — a sequence of numbers the computer can store and process.
More specifically, to get a analog signal into the computer, let's break down what happens during digitization with an A/D converter — using this visual example.
Here you see an analog signal, represented here as a smooth, continuous wave — like a sound wave moving across time.#Actually, a digital computer can't store this continuous curve directly.#Instead, it must sample the signal at specific time points.
At each of these time points, we ask:#What is the value of the signal right here?#And that gives us a sequence of numbers.
In the rightmost illustration, you can see vertical bars that "catch" the wave at regular intervals.#These bars represent sampling operations — and the height of each bar shows the value of the signal at that point in time.
Now, here's something important:#The computer can't store every possible value —

like pi=3.14159 etc. or e=2.71828 etc. with infinite precision.#Instead, it must round the values to the nearest available level.
This process is called quantization.#It means converting a smooth, continuous range of amplitudes into a set of fixed, discrete values — often represented by whole numbers.
So here, we get a string of numbers like:#3, 5, 6, 6, 4, 2, 1, 2.
And these numbers are what the computer stores.
But even these whole numbers aren't stored in decimal form.#Computers use binary — just zeros and ones to represent either an integral or a decimal number.
So each number gets translated into binary. For example:
1 becomes zero one,
2 becomes one zero,
3 becomes one one,
and so on.
This is how a continuous analog signal — like sound or temperature — becomes a digital signal inside a computer:#First through sampling across time, then through quantization of amplitude, and finally through binary encoding.
Each step introduces a tradeoff:
More sampling points give better resolution in time.
More quantization levels give better precision in amplitude.
But more of both means more data to store and process.
This is the foundation of digital signal processing — and now you've seen how it all begins.
And finally, that digital data — made up of zeros and ones — enters the computer, where it can be displayed, analyzed, stored, or transmitted.
So again, the pipeline goes like this:
Physical system → transducer → signal conditioning → analog-to-digital conversion → computer.
This entire process happens in almost every modern system — from medical devices to smart homes to autonomous vehicles.
And the better we understand this chain, the better we can design systems that are accurate, efficient, and reliable.

slide15:
So now that we've brought our signal into the computer, let's take a closer look at what's really happening during analog-to-digital conversion.
As you can see in the top plot, we start with a smooth, continuous signal — x of t — defined for all time and with continuous amplitude.
But the computer can't store that continuous curve — it needs discrete data.
So we begin with the first major step: sampling, also called discretization in time.
This means we only record the value of the signal at selected time points — evenly spaced along the horizontal axis.#At each of these points, we "catch" the value of the signal, just like placing a pin at that instant.
'
Now, the second step is quantization — converting the amplitude of each sample into a finite-precision value.
We can't store irrational numbers like pi or e with infinite precision.#Instead, we round them to a reasonable approximation — say, 3.14 for pi.#And for our purposes, we assume this rounding is accurate enough to not affect the result significantly.
So in our analysis, we mostly ignore the quantization error and focus on sampling — the time discretization.
The bottom diagram shows how the sampled signal looks:#It's now a series of impulses — each one located at a sample time and scaled to the value of the original signal at that moment.
This sequence of impulses is what we work with in digital signal processing.
But here's something important to remember:
The real signal — the one that matters in the physical world — is still continuous.
What we do with computers is a second-best approximation, based on discrete samples.
So the key question becomes:
Can we process these sampled values in a way that still lets us understand or

recover the original continuous signal?
This is the challenge at the heart of signal processing — bridging the gap between discrete computation and continuous reality.
And that’s what we’ll be exploring in the next steps of our journey.

slide16:
Now, let’s go back and revisit the question we just posed:#If we take a continuous signal and convert it into discrete samples, can we still reconstruct the original signal?
To explore that, let’s start with this example — a pure continuous wave.
Here, we’ve plotted the function#5 times sine of 2 pi 4 t#That’s a simple sine wave with amplitude 5 and frequency 4 cycles per second — or 4 hertz.
And this signal is completely continuous in both time and amplitude.
That means it's defined for every instant in time, not just selected points, and the values can take on any number, not just a limited set.
This is the kind of signal you might get from an ideal microphone or sensor measuring vibration, sound, or light — in the real world, before any digital conversion takes place.
But here’s the key point:
The computer can’t store this curve as it is.#So we need to figure out: How often do we need to sample this signal to capture all the important information?
How dense should our sampling be, so that we don’t lose critical features?
And that question brings us right to the Sampling Theorem — the mathematical foundation for how we digitize continuous signals without losing information.
That’s exactly what we’ll examine next.

slide17:
So here we have another view of that same example.
This wave has a frequency of 4 hertz — meaning it completes 4 full cycles per second.
Now we’re sampling it with a very small interval, or in other words, at a very high rate: 256 samples per second. That’s far more frequent than the wave itself changes.
And the result is what we call well-sampled data.
As you can see, the discrete dots — the sampling points — follow the shape of the original wave very closely.#There’s no confusion. No ambiguity. No major information is lost.
This is exactly what we want when digitizing a signal — the samples capture the behavior of the original continuous signal with high accuracy.
Heuristically, you can just look at the plot and say: yes, these samples preserve the essence of the original wave.
So, in this case, we can confidently say: the sampling was successful.
But what if we sample too slowly?
Let’s take a look at that next.

slide18:
Now here’s what happens when the sampling is too slow.
This plot shows a rapidly oscillating signal — it's a sine wave with a high frequency. But look at those red dots. Those are the actual points we sampled, and as you can see, there are only a few of them.
This is a classic case of under-sampling.
In this case, the original signal — that fine blue waveform — is oscillating at a frequency of 8 hertz. But the sampling rate is just 8.5 samples per second, which is not high enough to capture the details of the wave.
So what’s the problem?
Well, based on just the red dots, it might look like the wave is slowly rising and falling — as if it’s a low-frequency signal. But that’s not true at all.
This is known as aliasing — when a high-frequency signal gets misinterpreted as something much lower in frequency, simply because it was sampled too slowly.
So even though you're doing regular sampling — evenly spaced in time — the result can be very misleading if your sampling rate is too low.
This example shows why choosing the right sampling rate is critical.#Too slow, and you could end up completely misrepresenting your signal.
Next, let’s talk about the theoretical limit — the Nyquist rate — and how it

protects us from this problem.

slide19:
Here we’re looking at the difference between a continuous signal and its discrete version.
On the left, we have a smooth, continuous function — like something you might see in the physical world, such as a sound wave or a voltage signal. It's defined at every instant in time.
But to work with this signal on a computer, we can't use all those infinite points. Instead, we select only a finite number of values, spaced out at regular intervals. That's what's shown on the right.
This process is called sampling.
If we sample densely enough — meaning, if the points are close together — then the discrete version can represent the continuous signal fairly accurately. The more samples we take, the more detail we preserve from the original waveform.
So the idea is: start with a smooth, continuous signal and convert it into a sequence of numbers that still captures the shape and behavior of the original. That's what makes signal processing on computers possible.
Up next, we’ll explore how often we need to sample to preserve that accuracy — and what happens if we don't.

slide20:
Now, here’s a really important idea — something we call the aliasing problem. Remember, we just said that sampling needs to be dense enough? But what does that mean?
Take a look at the curve on the left. It’s a complex, high-frequency signal — lots of rapid changes and fine details. If we don’t sample it frequently enough — meaning, we don’t take enough points per second — we miss those small peaks and dips.
In the middle, you see the result of sparse sampling. We’re only picking up a few values. And when we try to reconstruct the signal, shown on the right, we get something much smoother and simpler than the original. It might even look okay at first glance, but it’s actually wrong.
All that detail — all the fast oscillations — is lost. Even worse, the reconstructed version can look like a completely different signal, with a false shape or even the wrong frequency.
This is what we call aliasing. It happens when the sampling rate is too low to capture the higher-frequency components. And once they’re lost, you can’t recover them from the sampled data.
That’s why choosing the right sampling rate is so critical in signal processing.

slide21:
Let’s now look at what’s happening in the spatial domain when we sample a signal.
Imagine you have a continuous signal — shown on the left. To sample this signal, what we’re really doing is multiplying it by a train of impulses. That’s the middle figure.
These impulses are spaced evenly along the axis, representing fixed sampling intervals. And when you multiply the continuous signal by this impulse train, the result is a set of discrete values. That’s what you see on the right: only the values of the signal at those impulse positions are preserved — the rest are discarded.
In the spatial domain, sampling is essentially a point-wise multiplication of the signal with an impulse train.
And here’s the key idea: this operation in the spatial domain has a direct consequence in the frequency domain. Multiplying by an impulse train in space corresponds to replicating the spectrum of the signal — creating a periodic train of delta functions in the frequency domain.
Just as we observe periodicity in time or space due to regular sampling, we also obtain periodicity in frequency. This connection — between operations in the spatial domain and their effects in the frequency domain — is fundamental in signal processing.

slide22:

Now let’s move to the frequency domain and see what sampling looks like there. Earlier, we saw that sampling in the spatial or time domain means multiplying the signal by an impulse train. In the frequency domain, that multiplication becomes a convolution.
Here’s what happens:
On the left, we have the original signal’s spectrum — a smooth curve that shows how the energy is spread across different frequencies.
In the center, we see a train of delta functions. These represent the periodic sampling pattern in the time domain.
When we convolve the original spectrum with this delta train, what we get is shown on the right: multiple shifted copies of the original spectrum. One in the center, and others repeated at regular intervals to the left and right. These are called spectral replicas.
Now, this setup is fine if the copies don’t overlap. But in the case shown here, the original spectrum is wide — it spreads across many frequencies — and the spacing between these delta peaks is not large enough. So the replicas overlap, and this overlap is what causes aliasing.
Aliasing is when different frequency components get mixed together. It’s like solving the equation x plus y equals 10, you can't tell how much is x and how much is y. Similarly, when spectra overlap, you lose the ability to tell which part of the signal came from which frequency. That’s a problem.
So to avoid aliasing, we need to space those spectral copies far enough apart — which means we need a high enough sampling rate in the time domain. That ensures each frequency copy stays separate, and the original signal can be recovered accurately.

slide23:
Now let’s shift gears and talk about conditioning in the spatial domain.
Suppose you start with a signal that has a lot of sharp changes — a noisy or high-frequency function, like the one shown on the left. Before sampling or further processing, we often want to smooth out the extremes. This helps avoid issues like aliasing or instability in reconstruction.
So what do we do?
We multiply the signal by a smooth, bell-shaped function — something like a Gaussian. That’s the curve shown in the middle. This operation acts like a soft window. It suppresses the values at the edges and emphasizes the center part of the signal.
The result, shown on the right, is a conditioned signal. It’s still based on the original data, but now it’s more concentrated, more stable, and less prone to causing problems downstream.
This process is called conditioning — and in the spatial domain, it’s done by pointwise multiplication. We’re not changing the signal globally — just shaping it locally to make it behave better.
This idea becomes even more powerful when we look at its effect in the frequency domain — and that’s where we’re headed next.

slide24:
Now let’s see why we’re often better off working in the frequency domain.
Suppose we start with a nicely concentrated spectrum — something like the narrow peak shown on the left. This means our signal in the spatial domain is smooth and well-behaved.
Now, when we sample the signal — which, in the frequency domain, corresponds to a convolution with a delta train — we generate copies of this spectrum. These appear periodically across the frequency axis. You can see a central copy and two others on either side.
If the original spectrum is narrow enough, these copies won’t overlap — and that’s a key idea. No overlap means no aliasing. That’s good news.
In this case, we can use a low-pass filter to isolate the central copy and discard the rest. This makes it possible to perfectly reconstruct the original signal from its samples.
That’s why frequency domain analysis is so powerful — it gives us a clear way to understand, diagnose, and even fix sampling-related problems. And it’s where the idea of an ideal sampling filter comes in. This filter selects only what we need and suppresses what we don’t.

We’ll explore this concept even further as we continue.

slide25:
So what does an ideal sampling filter look like?
Let’s start in the frequency domain. If we want to keep only the part of the spectrum we care about, we can apply a rectangular window — often called a gate function. This simply means we multiply the spectrum by a box that passes certain frequencies and zeros out the rest. Everything outside the band is eliminated.
Now here’s the important part: multiplication in the frequency domain corresponds to convolution in the spatial or time domain. So when we apply this gate in Fourier space, the corresponding operation in the spatial domain is convolution with the inverse Fourier transform of the rectangle. And that inverse transform is something we’ve seen before — it’s the sinc function.
This sinc function has a sharp central peak and extends out forever with oscillating ripples. We say it has infinite support, and those ripples are often called ringing. So while the ideal filter works perfectly in theory, it’s not so practical in real applications. Why? Because we can’t build something with infinite extent. But conceptually, it’s extremely valuable — it helps us understand the limits of what’s possible in signal reconstruction.
We’ll next look at what happens when we approximate this ideal.

slide26:
Now let’s talk about a more practical — but less perfect — alternative: the cheap sampling filter.
Suppose we switch things around. Instead of applying a gate function in the frequency domain, we use a sinc function there. In that case, its counterpart in the time domain is a rectangular function.
This rectangle acts like a local averaging window — it smooths the signal by averaging values over a small time interval. You can think of it as performing a moving average across discrete sample points to approximate the original continuous signal.
It’s a simple, easy-to-implement method — and that’s why we call it "cheap." But here's the tradeoff: it’s far from ideal.
In the frequency domain, we’re no longer cleanly cutting off frequencies beyond a certain band. The sinc function doesn’t have sharp boundaries, so it doesn’t isolate frequencies as cleanly as the gate function would. That means the resulting reconstruction may lose some fidelity or allow unwanted frequency components to leak in.
So yes — this filter is convenient and fast, but it comes at the cost of precision.
Next, we’ll explore how filtering in the spatial domain can help improve this.

slide27:
When we want a practical alternative to the ideal or cheap sampling filters, a Gaussian filter offers a great compromise.
Now, what makes the Gaussian special is this: its Fourier transform is also a Gaussian. That’s quite rare — most functions change significantly when transformed between the spatial and frequency domains. But the Gaussian stays in the same form, just scaled differently.
So, if you multiply by a Gaussian in the frequency domain, that corresponds to convolving with a Gaussian in the spatial or time domain — and vice versa.
Why is this helpful? Because the Gaussian function decays very quickly. That means we avoid the problem of infinite ringing, like we saw with the sinc function. Instead of having a sharp cutoff, which can cause artifacts, the Gaussian gives a smooth transition and localized effect — both in frequency and in space.
It's not mathematically perfect, but it’s practical. And in many real-world applications — especially in imaging and signal processing — that’s the trade-off we need.
So overall, the Gaussian sampling filter is a good, balanced choice that works well in both domains.

slide28:

Alright, let’s take a step further and make things more precise.
Earlier, I showed you some cartoon-like pictures to build intuition. Now, let’s look at the actual math behind those ideas — starting with the delta function.
In signal processing, the delta function plays a key role — especially when we have a series of delta functions lined up at regular intervals. We often call this a comb, because visually, it looks just like the teeth of a hair comb.
Here’s the key idea.
Imagine you have a bunch of delta functions spaced by a fixed time interval, T.
We write that as:#s of t equals the sum of delta of t minus n times T.
Now, when you take the Fourier transform of this comb-shaped signal, something beautiful happens. You get another comb — but this time in the frequency domain.
There’s a simple rule:
If the spacing in time is T,
Then the spacing in frequency becomes 1 over T.
The impulses are still evenly spaced, but now they live in the frequency domain.
And their height is scaled by a factor of 1 over T.
So this transformation turns one comb into another — just flipped into frequency space, with inverted spacing.
This is a fundamental result you’ll see in many signal processing books.
Different authors may use different symbols — like f or u for frequency, or j or i in the exponential — but the meaning is the same.
Also, keep in mind that delta functions are really convenient in Fourier analysis. When you plug a delta function into the Fourier formula, it simplifies easily — because a delta picks out a single point.
And when you shift the delta in time — say, to delta of t minus some value — the result in frequency becomes a complex exponential, like e to the j 2 pi f zero t. That’s thanks to the shift theorem.
So, the takeaway here is:#a comb of delta functions in time becomes another comb in frequency — just with spacing flipped and amplitude scaled.
This sets the stage for understanding how sampling works in both domains. We’ll build on this in the slides that follow.

slide29:
Let’s now take a closer look at the Fourier transform of this impulse train — and I’ll explain it in a way that’s easier to follow, even if not perfectly rigorous.
Suppose we have a signal made of delta functions spaced by a fixed interval, capital T. This is a periodic function, right? It repeats every T. So we can use a Fourier series to represent it.
We write this function, s T of t, as a sum of delta functions at every multiple of T:#s T of t equals the sum over n of delta of t minus n T.
Now, since it's periodic, we can also express it as a sum of complex exponentials — that’s what a Fourier series does. So we write:#s T of t equals the sum over n of c n times e to the power j 2 pi n t over T,#where c n is the Fourier coefficient.
To compute c n, we take the standard approach:#c n equals 1 over T times the integral over one period of s T of t multiplied by e to the power negative j 2 pi n t over T.
But here’s the trick — remember that s T of t is a train of delta functions. That means when you integrate, only the delta at t = 0 survives.
The result is simply:#c n equals 1 over T.
So when you plug this back in, your entire Fourier series becomes a sum of complex exponentials, each with coefficient 1 over T.
Now, there’s a well-known identity:#A sum of complex exponentials with equal spacing and equal weights becomes a train of delta functions in the frequency domain.
So finally, we can write:#s T of t transforms to (1 over T) times s 1 over T of u.#That is, the train of impulses in time maps to another train of impulses in frequency, with flipped spacing and scaled amplitude.
This is what we call a Fourier pair. A comb in one domain becomes another comb in the other domain — with the spacing inverted and amplitude scaled.
This idea is fundamental in signal processing. You’ll see it over and over — whether you’re analyzing sampling, modulation, or reconstruction.
Now, I’ll admit — this version of the derivation is not entirely rigorous. It's

a bit hand-wavy. A more precise proof would involve deeper mathematical tools, especially when it comes to convergence. But for our purposes, this intuitive explanation gives you the right picture.
So just keep this in mind:#A periodic sequence of impulses in time transforms into another periodic sequence in frequency, and the spacing and amplitude follow a simple reciprocal relationship.

slide30:
Let’s take a moment to revisit this important concept — the impulse train, or what we often call a "comb", and its mirror relationship in the Fourier domain.
On the left side, we see a train of delta functions spaced by a constant interval, capital T. This lives in the time domain. And on the right, we see another train of delta functions — this time in the frequency domain — spaced by one over T.
This is a classic Fourier pair. When you take the Fourier transform of a periodic train of impulses in time, you get another train of impulses in frequency. And the spacing in the frequency domain is the reciprocal of the spacing in the time domain.
Now, let’s look at the lower pair of plots.
Here, the red comb has impulses spaced by a small interval, delta t. In that case, the spacing in the frequency domain becomes capital P, which equals one over delta t.
Again, we see the same idea — spacing in one domain leads to reciprocal spacing in the other. This reciprocal relationship is a fundamental feature of the Fourier transform.
The height or amplitude also changes accordingly. Specifically, when the spacing decreases — say, from T to delta t — the impulses become more densely packed, and the total energy gets spread out more in frequency space. So the amplitudes must be adjusted accordingly to conserve energy.
So in summary:#A periodic impulse train in time becomes a periodic impulse train in frequency.#The spacing flips — T becomes 1 over T.#The amplitudes are scaled appropriately.
This relationship is so central to sampling, reconstruction, and many other applications in signal processing. You’ll see it again and again — both in theory and in practice.

slide31:
Now let’s dive into the process of sampling a signal — how we go from the continuous world to the digital one.
Look first at the top-left plot. That’s our original continuous signal, shown over a finite interval, from minus T over 2 to plus T over 2. Our goal is to digitize this — to convert it into a form that we can process on a computer.
And to do that, we take measurements at evenly spaced time intervals. This step is crucial, because without any prior knowledge about the signal, we don't know where we should sample more densely or more sparsely — so we just use uniform sampling.
We model this sampling process mathematically by multiplying the signal with a train of impulses — those red vertical lines spaced by delta t. This models a measurement being taken at regular intervals — just like a thermometer recording temperature every hour.
When we multiply the continuous function by this comb of impulses, we’re essentially pulling out values at those discrete points. The result is a sampled signal, where the values are available only at those locations. That’s what you see in the next plot below — the discrete signal.
Now shift your attention to the right-hand side — to the frequency domain.
The original continuous signal, when transformed using the Fourier transform, gives us a spectrum — and here it's shown to be concentrated between minus W and plus W. That’s the bandwidth of the signal — most of its energy lies within that frequency range.
And here’s something interesting: if a signal is limited in time — meaning it's non-zero only within a certain interval — then its Fourier transform will spread out infinitely. But if the time-domain signal is smooth, then the energy in its spectrum decays very fast. So in practice, we can often ignore the tiny tails and just focus on the dominant part within minus W to W.

Now, here comes the key insight.
When we sample the time-domain signal using a comb — a series of impulses spaced by delta t — then in the frequency domain, the Fourier transform becomes a convolution. That convolution replicates the original spectrum at regular intervals — and those intervals are one over delta t, which we denote as capital P.
So in the frequency domain, we now have multiple copies of the original spectrum, all spaced by P. These are sometimes called spectral replicas.
And here's the big question: how small should delta t be? Or in other words — how dense should our sampling be — so that these spectral copies don't overlap?
Because if they overlap, we get aliasing, and the original signal can't be recovered. But if P — that is, one over delta t — is large enough, then these copies stay nicely separated, and we can isolate just one of them to recover the original signal perfectly.
So in short, if we sample densely enough, we don't lose any information. We can convert a continuous signal to a digital one and then reconstruct it — as if nothing was lost.
And that naturally brings us to the next topic — what is the minimum sampling rate we need to guarantee perfect recovery?
Let's take a look.

slide32:
Let's go back to the real form of the Fourier series.
Here's the idea: if you have a periodic function — let's call it f of t — you can express it as a weighted sum of sines and cosines.
The formula goes like this:
f of t equals#a naught divided by 2,#plus the sum, from n equals 1 to N,#of a n times cosine of 2 pi n t,#plus b n times sine of 2 pi n t.
In this expression, a naught is the average value of the function over one full period. That's also called the DC component.
The rest of the terms — the cosine and sine parts — describe how the function varies over time. Each n corresponds to a different frequency: the higher the value of n, the higher the frequency.
Now, how do we compute those coefficients — a naught, a n, and b n?
Let's start with a naught divided by 2.
That's equal to the integral of f of t from 0 to 1.#In other words, you're just averaging the function over one period.
Next, to compute a n,#you take 2 times the integral from 0 to 1#of f of t times cosine of 2 pi n t.
And to compute b n,#You take 2 times the integral from 0 to 1#of f of t times sine of 2 pi n t.
Each coefficient tells you how much of that sine or cosine wave is present in the original function.
Now here's something important:#The lowest frequency in this series appears when n equals 1. That corresponds to a frequency of 2 pi.#The highest frequency depends on N, which is the upper limit of the sum — and that gives us a maximum frequency of 2 pi times N.
So by adjusting N, we can control how detailed our approximation is.
This is the real-valued Fourier series — a sum of sine and cosine waves, each scaled by how much of that frequency exists in your signal.

slide33:
Now, let's connect Fourier series with sampling.
Suppose you have a function, like the one shown here, and you want to represent it over time using a Fourier series. The key idea is this:
If your function is defined over a unit interval — let's say from zero to one — then the standard real-form Fourier series can represent it as a sum of cosine and sine terms, just like we saw earlier.
That's perfectly fine when your function repeats every one second. But in general, a real-world signal may repeat over a different period — maybe two seconds, or ten seconds, or any value we call capital P.
So, how do we handle that?
The answer is: you simply normalize your time axis. You scale the time so that the function fits into a unit interval. Then you apply the standard Fourier

series there.
Once you've built the series, you can scale it back to its original time frame — from zero to capital P — and you get a Fourier expansion that models the function over its true period.
So to summarize: the Fourier series has the flexibility to represent any periodic function, no matter what its period is. You just need to adjust for that period in the formulation — and the structure stays exactly the same.
This makes Fourier analysis an incredibly powerful tool when dealing with signals of any repeating pattern.

slide34:
Let's now talk about estimating the DC component in a signal.
Suppose we have a continuous function, and we sample it over the interval from zero to one. The first question we should ask is: how can we estimate the DC term?
Remember, in a Fourier series, the DC component is the constant part of the signal — and mathematically, it's given by the average value of the function over one full period. So, for a continuous function, that just means integrating from zero to one and dividing by the length of the interval.
Now, in practice, we don't work with continuous signals — we sample them. That means we take a finite number of measurements, evenly spaced across the interval. So how do we estimate the DC component from those samples?
Very simply: we add up all the sampled values, and then we take the average. That average gives us a good estimate of the DC component — the constant term in the Fourier series.
So once we've taken care of the DC term, we still need to determine the coefficients for the sine and cosine terms — the oscillating parts of the signal.
Let's say we want to recover N cosine coefficients and N sine coefficients. That gives us a total of 2 times N unknowns. And to solve for 2N unknowns, we'll need at least 2N independent equations — meaning 2N sampled values.
So the takeaway is this: to fully recover a periodic signal using its Fourier series up to N harmonics, you need at least 2N equally spaced sample points. That gives you just enough information to solve for all the Fourier coefficients.
And again, the DC component is simple — just average your samples.

slide35:
Now let's explore an interesting point in signal representation — how sine and cosine terms can be combined.
In Fourier analysis, you'll often see expressions involving both sine and cosine terms. For example, something like A sine theta plus B cosine theta. This is a common way to write periodic signals, using a mix of sines and cosines.
But here's the neat part: this combination can actually be rewritten as a single sine function — with an amplitude and a phase shift. In other words, we can express it as R sine theta plus alpha, where R is the amplitude and alpha is the phase.
This identity is not just useful — it's powerful. It tells us that instead of dealing with two separate unknowns, A and B, we can represent the same information using two different unknowns: R, the amplitude, and alpha, the phase.
So nothing is lost — we're simply transforming the representation. Instead of working in the sine-cosine basis, we switch to amplitude and phase.
If you're curious about the proof or how this identity works, you can easily find tutorials or short videos online that walk through the steps in detail.
But the main takeaway here is: whenever you have a sine and cosine pair, you can always combine them into a single sine with a shifted angle. It's a cleaner and often more intuitive way to represent oscillating signals — especially when we want to talk about signal magnitude and delay.

slide36:
Let's now take a heuristic view to better understand the relationship between sampling and signal reconstruction.
Suppose we represent our signal using a sum of sinusoids. Each term looks like

this: A n sine of 2 pi nu t plus phi n. Here, A n is the amplitude, phi n is the phase, and nu is the frequency.
So for each sinusoidal component, we need to determine two unknowns — amplitude and phase. That means, for N such components, we end up with 2N unknowns in total.
Now, here's the key point. To determine those two unknowns for each sinusoid — amplitude and phase — we need at least two data points per cycle. Think of it like trying to draw a sine wave: if you only know one point, you can't say much. But with two points, you start to get the shape — its size and where it starts.
So, for the highest frequency in our signal — which we'll call nu — we need to sample at least twice per cycle. This leads to the idea of spacing between samples. To avoid missing any important information, the spacing between adjacent samples must be less than or equal to 1 over 2 nu.
In more familiar terms, that means the sampling rate — measured in samples per second — must be greater than 2 times nu. This is the famous Nyquist sampling rate.
Why does this matter? Because if we sample below this rate, we risk losing information. Our signal might be distorted or misrepresented — a problem known as aliasing. But if we sample at twice the highest frequency or more, we guarantee that every sinusoidal component is accurately captured.
So the Nyquist rate gives us a fundamental guideline: sample fast enough to resolve the highest frequency present in the signal.

slide37:
Now we're entering the most important part of this discussion — the mathematical foundation behind the sampling theorem.
Let's start by recalling our goal: we want to take a continuous signal, sample it at regular intervals, and then perfectly reconstruct the original signal from just those samples. But this only works under certain conditions — particularly, when the sampling rate is high enough to avoid aliasing. That's what the Nyquist theorem guarantees.
Now, let's walk through this derivation step by step. It's a bit more mathematical, but I'll explain the reasoning behind each part.
We begin by expressing the function f of t as the inverse Fourier transform of a product — specifically, a low-pass filter in the frequency domain multiplied by a sampled version of the original function's spectrum. Symbolically, we write:
f of t equals the inverse Fourier transform of the product of capital Pi P of u and the convolution of f hat of u with S P of u.
Next, we apply properties of the Fourier transform. We pull out the convolution and apply inverse transforms separately. This leads to:
f of t equals the inverse transform of Pi P of u, convolved with the inverse transform of f hat of u multiplied by S P of u.
Then, we take this result back to the time domain.
Here's what we get:#f of t equals P times sinc of pi P t, convolved with f of t times S 1 over P of t.
This expression means we're convolving the sinc function — which comes from the inverse transform of a box function in frequency — with the sampled version of the original function.
And from there, we arrive at the reconstruction formula you've probably seen before:
f of t equals the sum over k from minus infinity to infinity of f at k over P times sinc of pi times the quantity t minus k over P over pi times the quantity t minus k over P.
This is the classic sinc interpolation formula. It tells us that, under the right sampling conditions — meaning, the sampling rate is at least twice the highest frequency in the signal — the continuous signal can be reconstructed exactly by convolving the samples with the sinc function.
So in summary: sampling in the time domain corresponds to periodic replication in the frequency domain. When the signal is bandlimited and the sampling rate is high enough, we can perfectly reconstruct it using this formula.
This is the core of the sampling theorem. And this formula — the final one on the slide — is one of the most beautiful results in signal processing.

slide38:

Let’s go through a helpful example using a two-dimensional rectangular function. This is a classic case to build intuition around how the Fourier transform behaves in two dimensions.
Imagine a rectangle that’s centered at the origin. It has a total width of capital X along the x-axis, and a total height of capital Y along the y-axis. The function value is constant — say, equal to 1 — inside this rectangle, and zero outside.
We now take the 2D Fourier transform of this rectangle. The formula looks a bit dense, but the steps are straightforward.
We start with the double integral of the original function times the complex exponential — e to the power minus j 2 pi times the quantity u x plus v y. That’s the standard definition of the 2D Fourier transform.
Since the function is separable — meaning it depends independently on x and y — we can break the double integral into two parts:
one over x, and one over y.
Evaluating each of these integrals separately gives us a product of two expressions.
Each part turns into a sinc function. In the x-direction, we get sin of pi times X u over pi X u — that’s sinc of pi X u. And in the y-direction, it’s sinc of pi Y v.
So the full 2D Fourier transform ends up as the product of these two sinc functions — one in the u-direction, and one in the v-direction — scaled by the area of the original rectangle, which is X times Y.
Now if you look at the plot here, you’ll see a 3D surface that shows the magnitude of the Fourier transform — the peak is at the center, and the ripples decay outward. That shape comes directly from the sinc functions.
Why are we showing this?
Because it mirrors the result we saw earlier: when a rectangular function is defined in one domain — here it’s the spatial domain — its transform becomes a sinc function in the other domain — in this case, the frequency domain.
This is closely related to the idea we discussed on the previous slide, where the rectangle — or gate — was in the frequency domain, and its transform was a sinc function in time. Whether you go forward or inverse, the core relationship stays similar.
So while this example is technically a bit different — it’s 2D, and it’s a forward transform — the intuition still applies. Rectangles and sinc functions are transform pairs. And we use that fact again and again in signal processing and imaging.

slide39:
Now, let’s continue line by line and see how each step connects.
We’re now looking at this part of the derivation — specifically the fourth equation on the slide.
Here, we have the product of two terms:
a sinc function, written as sine of pi P t over pi P t,
and the convolution with another function in square brackets.
Let’s first focus on the sinc part. This comes from taking the inverse Fourier transform of the rectangular function in the frequency domain — the gate function we introduced earlier. When we take its inverse Fourier transform, we get a sinc function in the time domain. That’s because a rectangle in frequency always corresponds to a sinc in time. So that’s where this sine of pi P t over pi P t comes from — it’s our sinc kernel.
Now, let’s turn to the function inside the brackets.
Here, we’re performing an inverse Fourier transform on the product of two terms:
the Fourier transform of our original function,
and a periodic impulse train in frequency, with spacing capital P.
By the convolution theorem, multiplying two functions in the frequency domain is equivalent to convolving their inverse transforms in the time domain. So we convolve the original function f of t with the inverse transform of this impulse train in the frequency domain.
And what is the inverse transform of a periodic impulse train in frequency?
It gives us another periodic impulse train — this time in time — with spacing equal to one over capital P. So this entire expression simplifies to f of t convolved with a sum of shifted delta functions, spaced by one over P.

We can express that more clearly on the next line. The equation now becomes:
f of t equals sinc of pi P t multiplied by the sum over all k from minus infinity to infinity of f of t times delta of t minus k over P.
And because convolution with a delta function simply samples the function, this expression simplifies further — giving us the familiar formula for signal reconstruction.
So finally, we arrive at the last equation on this slide:
f of t equals the sum over k, from negative to positive infinity,#of f evaluated at k over P,#times sinc of pi times the quantity t minus k over P.
This is the reconstruction formula for band-limited signals. It tells us that if the signal was sampled at or above the Nyquist rate — meaning no aliasing occurred — then we can recover the original continuous function perfectly by summing shifted sinc functions, each scaled by the sampled value.
That's the power of the sampling theorem. It doesn't just allow us to digitize signals — it tells us exactly how to recover them.

slide40:
Now let's take a moment to talk about an important concept in signal processing — the comb function and how it behaves under the Fourier transform.
Let's start with this equation at the top. It says:
s of t equals the sum over all n of delta of t minus n times capital T.
This is what we call a delta train or comb function in the time domain. It's a series of impulses, evenly spaced by T units.
Now, when we take the Fourier transform of this comb, something very elegant happens.
It becomes another comb — but in the frequency domain.
Specifically, the transform is:
s of f equals one over capital T times the sum over all n of delta of f minus n over capital T.
So, in words, a comb in time with spacing T becomes a comb in frequency with spacing 1 over T. And we get a scaling factor of 1 over T in front.
This is a fundamental symmetry in Fourier analysis: when a signal is periodic in time, its spectrum is discrete in frequency — and vice versa.
Now look at the plots in the middle of the slide.
On the left side, we see a train of impulses spaced T apart along the time axis.
On the right, we have the corresponding Fourier transform — another train of impulses but now spaced 1 over T along the frequency axis.
So there's this beautiful reciprocal relationship: if your delta train is spaced T in time, its Fourier counterpart is spaced 1 over T in frequency.
And this brings us to the red box at the bottom of the slide. These equations generalize the idea.
Let's say you have a sum of delta functions shifted by n over P. That's a comb with spacing 1 over P.
Then, its Fourier transform becomes P times a sum of delta functions spaced P apart. The scaling factor flips — one becomes P, and the spacing flips from 1 over P to P.
And if we put a scaling factor of 1 over P in front of the comb, then the Fourier transform is simply a comb with spacing P, with no extra scaling factor.
So again, delta trains are their own Fourier transforms — up to a scaling and spacing change.
This property is essential when we talk about sampling, because whenever we sample a signal, we're essentially multiplying it by a comb function in time.
And that multiplication creates repeated spectra in the frequency domain — spaced according to the sampling rate.
So that's the comb and its mirror — one of the most powerful dualities in Fourier theory.

slide41:
Now let's focus on the last few steps of the derivation and clarify what's happening here, line by line.
Previously, we had the convolution of two time-domain signals: one was a sinc function, the other was a sum of scaled impulses. That brought us to this expression:
f of t equals sinc of pi P t, divided by pi P t, convolved with the sum from k

equals minus infinity to infinity of f of t times delta of t minus k over P.
Let’s unpack that.
First, remember that this sum is what we call a delta train — an infinite series of delta functions spaced at intervals of 1 over P.
Now you might ask — wait a minute, where did the 1 over P factor go?
Good observation.
Originally, the delta train had a scaling factor of 1 over P. But earlier in the derivation, we also had a P multiplying the sinc function. So those two terms — P and 1 over P — cancel out. That’s why we don’t see the 1 over P anymore in the final expression.
So what do we have now?
We have a continuous function — that’s the sinc — multiplying a series of impulses that are sampling the original function at regular intervals.
This is the essence of the sampling process.
The function f of t is being sampled at every t equals k over P, and the result is a series of weighted sinc functions. Each one is centered at t equals k over P and scaled by the corresponding sample value, f of k over P.
And that brings us to the final expression:
f of t equals the sum from k equals minus infinity to infinity of f of k over P, multiplied by sinc of pi P times t minus k over P, divided by pi P times t minus k over P.
That’s the sampling theorem.
It tells us that if a signal is band-limited — that is, its frequency content is restricted — then we can reconstruct the entire continuous-time signal exactly from its samples, as long as the sampling rate is high enough.
Each sample contributes a scaled sinc function, and the sum of all those sinc functions reconstructs the original signal.
This formula is both elegant and powerful — and it lies at the foundation of all modern digital signal processing.

slide42:
Let’s take a step back now and look at how analog signals are converted to digital.
At the top, we have a smooth, continuous signal — this is your analog signal — represented here as f of t, where t is time.
Now, to digitize this signal, we begin with the process of sampling. In the middle plot, you see a train of delta functions — these are evenly spaced impulses in time. Each one of them marks a point where we will sample the analog signal.
Next, what do we do?
We multiply the continuous signal by this train of impulses. That’s shown in the bottom figure.
This multiplication essentially picks out the values of the original signal at those discrete time points — and sets everything else to zero. In other words, it’s like punching holes in the continuous function, keeping only the values at regular intervals.
The result is a series of weighted impulses, where the height of each spike corresponds to the signal’s value at that instant.
And this is the essence of analog-to-digital conversion.
We’ve turned a smooth, continuous-time signal into a discrete-time representation — something that can now be stored, processed, and transmitted digitally.
This is the first key step in digital signal processing — and it's entirely grounded in the mathematics of the sampling theorem we just discussed.

slide43:
Now let’s focus on this final and very important step.
What you see here is the reconstruction formula — a powerful result that tells us how to rebuild the original continuous-time signal, f of t, from its discrete samples.
Here’s how it works.
We start with a set of sampled values. These are the values of f of t taken at regular intervals — specifically at times k over P. These are the points where our delta functions were placed during sampling.

Now, we take each of those sampled values — each f of k over P — and multiply it by a special function called the sinc function. You can see it written here as sine of pi times P times t minus k over P divided by pi times P times t minus k over P.
This sinc function looks like a smooth oscillating wave that decays gradually away from its center. And what's important is that it has the perfect shape to reconstruct smooth curves from discrete data points.
Now here's the key idea.
Each sampled value acts like a weight, and each weight generates a shifted copy of the sinc function centered at that sample point. So at k over P, we get a sinc function that peaks there, scaled by the sample value at that point.
Then we add up all these sinc functions — one for each sample.
The result? A continuous, smooth signal — which is exactly the original signal f of t that we started with.
This is how sampling and reconstruction come together. Each delta function "copies" the sinc function. Each copy is scaled by the corresponding sample value. And the sum of all these scaled sinc functions recreates the original analog signal.
So this final formula — this infinite sum — tells us something remarkable:
As long as we sample carefully, we can fully reconstruct a continuous signal from just its discrete samples.
That's the beauty of the sampling theorem.

slide44:
To wrap up our discussion, let's take a closer look at this final visual — it ties everything together beautifully.
At the top, we see a simple rectangular function, labeled f of x. It's a block function centered at zero. Next to it is h of x, which is a train of two delta functions — one at minus a over two and the other at plus a over two. Their amplitudes are scaled by one-half.
Now, when we convolve f of x with h of x, the result — g of x — is essentially just two shifted copies of the original function. Each delta acts like a copying machine, reproducing f of x at the location of the delta. This is a perfect illustration of how convolution with delta functions results in replicated versions of a signal.
Now let's look at the bottom half of the slide — we're moving into the frequency domain.
The Fourier transform of the original block function, f of x, becomes a smooth sinc-like curve, F of k. The delta train, h of x, transforms into a cosine wave pattern, H of k. And the product in the frequency domain, F of k times H of k, gives us G of k — a modulated version of the original Fourier transform.
This shows us that convolution in the time domain becomes multiplication in the frequency domain — one of the central results of Fourier theory.
Now, relating this back to the sampling theorem...
When we sample a continuous signal, we are essentially multiplying it by a train of delta functions. In the time domain, this multiplication picks out the values at those sampling points. But in the frequency domain, it leads to repetitions — or aliases — of the original spectrum.
And when we reconstruct, we don't use the simple rectangle anymore — instead, we use the sinc function, which is the Fourier transform of an ideal low-pass filter. But the principle is the same: the delta functions trigger copies, and these copies are scaled by the actual sample values — f of k over P.
And here's the key takeaway:
If we sample fast enough — meaning at more than twice the highest frequency in the original signal — and if the signal is band-limited, then all those sinc functions will line up perfectly, and we can fully recover the original signal.
That's the heart of the sampling theorem. It's not just an abstract formula — it's a practical method that allows us to move between continuous and digital signals with full confidence.
This slide beautifully summarizes the theory. From delta copies to sinc reconstruction, it's all about understanding how convolution builds signals piece by piece.

slide45:

Let’s now take a moment to revisit a topic that you’ve likely seen before — linear systems.
Here, we’re looking at the classic equation: A x equals b.#That’s a system of linear equations, where A is your matrix, x is the unknown vector, and b is the observed result or measurement.
Now, solving for x in this system is something you’ve probably done many times before — especially if you've worked through a textbook like Elementary Linear Algebra by Howard Anton, shown here.
But here's the key question for us:#What if the unknown vector x is sparse?
In other words, what if most of the elements in x are actually zero — and only a few are nonzero?
This situation comes up a lot in modern signal processing and imaging. We no longer assume that a signal is densely sampled or has energy spread all over. Instead, we often find that signals have a sparse structure, especially when we represent them in a suitable basis, like wavelets or gradients.
Now, here’s where it gets really interesting — and this is where modern theory breaks free from traditional Nyquist sampling.
Instead of sampling a signal very densely — at twice its maximum frequency — we can sample much more sparsely, and still recover the full signal.
How?
We change our assumption.#Instead of assuming the signal is band-limited, we assume it's sparse in some transform domain. Maybe it's sparse in the wavelet domain, or maybe the total variation is small — meaning the signal changes smoothly, or only has a few sharp transitions.
Then, among all possible solutions that match the measurements, we choose the sparsest one.
This is the idea behind compressed sensing, and it has revolutionized the way we think about sampling and recovery.
So that green button on the slide — it’s there to say: pause and think. This is a major shift in the traditional mindset. It opens the door to powerful new methods for signal reconstruction from limited data — especially in imaging and beyond.

slide46:
Let’s take a step further and talk about a powerful idea that shows up across science and engineering — and that’s sparsity.
You can see the green button here, which means this concept is important, but it’s okay if the details feel abstract for now. Just follow the intuition.
Traditionally, signal processing has focused on signals that are band-limited — which means their frequencies don’t go beyond a certain point. Based on that assumption, we rely on Nyquist sampling: sample at least twice the maximum frequency, and you can perfectly reconstruct the signal.
But what if we go beyond that assumption?
Modern theory says: you don’t always need to sample that densely. Instead of requiring a signal to be band-limited, we assume that it’s sparse in some domain. That means — in the right representation, most of the values are zero, or nearly zero.
Think about images. They may look complex, but when you apply a wavelet transform — like the ones shown in the graphics — most of the wavelet coefficients are close to zero. Only a few carry meaningful information. This is sparsity.
On the left, we have an illustration from the book Wavelets in Physics. It shows how signals in physics often exhibit localized spikes — sparse behavior. In the center and top-right, you see surfaces where most values are flat or close to zero, with just a few sharp peaks — again, sparse structure.
Even in biology — like the cellular structure of a leaf — patterns emerge that are spatially compact and repeated. This biological system, shown on the right, is highly structured and organized. And this kind of structure is often sparse when expressed in the right basis.
Now, here’s why sparsity matters.
If you don’t have enough measurements to fully determine a signal — that is, if your system is underdetermined — then infinitely many solutions could explain the data. But if you know the signal is sparse, you can ask: among all possible solutions, which one is the sparsest?

That’s the central idea behind compressed sensing.
By assuming sparsity, and by choosing the solution with the fewest nonzero components, you can accurately recover the original signal — even from very limited data.
So this concept — sparsity — is not just a mathematical trick. It’s something we see everywhere: in physics, in images, in biology, and in many natural and engineered systems. Recognizing it gives us a powerful way to reconstruct signals and solve problems more efficiently than ever before.

slide47:
Alright, let’s wrap up with the big picture.
This slide summarizes the key relationships between time and frequency domains — especially how signals and their spectra behave when we move from continuous to discrete representations.
At the very top left, we start with a continuous-time signal — a smooth pulse that exists over a limited time interval. On the right, we see its frequency content, also confined to a certain bandwidth.
Now, as we go down the slide, we perform sampling in time. That’s the red comb of vertical lines. We're picking values from the original signal at regular intervals — let’s say every delta t.
What does this do in the frequency domain?
It creates multiple copies of the spectrum, repeated again and again. This is called spectral replication. And if these replications don’t overlap, we can perfectly reconstruct the original signal.
That's the essence of the sampling theorem — to avoid distortion, we must sample fast enough, at least twice the highest frequency present in the signal.
Now, let’s look at the lower half of the slide.
Here, we’re sampling in the frequency domain instead. That’s shown by the green comb — regularly spaced frequency spikes.
What happens to the signal in the time domain?
It becomes periodic — the original signal now repeats over time, again and again.
So the big idea is this:
Sampling in time causes replication in frequency.
Sampling in frequency causes repetition in time.
These two processes mirror each other — one in time, the other in frequency.
This duality is at the heart of signal processing.
Now, I know this diagram is dense, but if you walk through it carefully — step by step — you’ll see how everything we’ve covered comes together.
We’ll cover this in more detail on Friday. In the meantime, I suggest reviewing this flow using the lecture slides and working through the derivations.
Even if there are a few typos in the book chapter, don’t worry. The key ideas are all here. And if you’re curious, feel free to preview the next topic — the Discrete Fourier Transform — in the book as well.