slide1:
Today, we move on to our fourth lecture, where our focus will be on convolution. This is a direct continuation of our recent discussion on systems, especially linear systems that are shift-invariant — two foundational concepts we began unpacking in the previous lecture. Understanding convolution will allow us to model how these systems respond to different types of input, and it's a concept that shows up across many areas in engineering, physics, and especially medical imaging.

slide2:
We are right on schedule in our journey through this material.
If you've already gone through the reading materials for today, that's excellent — it will help you link the concepts more quickly. If you haven't, that's okay. Just stay engaged during the lecture and review the key points afterward. Building this habit of revisiting ideas will strengthen your understanding, especially as we move into more math-heavy topics

slide3:
As we step into the topic of convolution, let's outline what we'll be exploring today.
We'll begin by revisiting shift-invariant linear systems, grounding ourselves in a concept we touched on earlier. Then, we'll move into two important formulations of convolution: continuous convolution and discrete convolution. Each of them plays a crucial role depending on the context — continuous for signals that evolve smoothly over time or space, and discrete for digital signals and images. Through this lecture, we'll examine both perspectives and their practical implications.
By the end of the course, you'll not only understand how convolution works — you'll also appreciate why it matters in system analysis, image reconstruction, and the broader field of medical imaging.

slide4:
Let's refresh our picture of a linear, shift-invariant system.
Imagine dropping a pebble into a still pond. No matter where the pebble lands, the ripples that form spread out in the same circular pattern. This uniformity — where the system's response doesn't change based on where the input occurs — is what we call shift-invariance.
Now, relate this to medical imaging. Suppose you undergo a scan at one hospital today and another hospital next week. If your physiological condition hasn't changed, you'd expect the scans to look the same. That's shift-invariance in action — it ensures reliability across time and space.
Next, we have linearity. It has two key components:
Additivity — if two features exist, like two separate tumors, the system should show the combination of both clearly.
Homogeneity — if a feature changes in intensity, like a tumor growing larger, the imaging should reflect that change proportionally.
Together, these properties form what's known as the superposition principle — the foundation for much of modern system analysis, including medical imaging.
So, as we move ahead, keep the ripple analogy in mind. It's a simple yet powerful way to visualize how linear, shift-invariant systems behave — consistently, predictably, and truthfully.

slide5:
To understand convolution deeply, let's frame it through a familiar story from physics.
You might recall the idea — sometimes attributed to Newton — that the universe began with a single impulse, a sort of divine kick that set everything into motion. Whether metaphor or mechanics, this idea captures something fundamental.
In physics, we often model this idea of an initial force as an impulse — a very short burst of energy that sets a system into motion. For instance, think of hitting a golf ball with a club. That brief push is an impulse, and the ball's subsequent motion is the system's response.

slide6:

Let’s take a closer look at what that impulse does.
When you strike a stationary object, like a golf ball, the force you apply doesn’t last long — it’s concentrated over a short time. But that momentary push has a lasting effect: it changes the ball’s velocity and sets it into motion.
This simple interaction captures an essential principle: an impulse causes change.
In the context of system analysis, we’re interested in how a system responds over time after receiving an impulse. That response — often called the impulse response — tells us how the system behaves dynamically. Once we know the impulse response, we can use convolution to determine the system’s output for any more complex input signal.

slide7:
Let’s look at how impulse relates to momentum — a key concept in both physics and system analysis.
Momentum, as you may recall, is simply mass multiplied by velocity.
So if an object is at rest, it has zero momentum. When you apply a force over a period of time — say, when you hit a tennis ball with a racket — the object gains velocity. That change in velocity reflects a change in momentum.
Now here’s the essential link:#Impulse is defined as force multiplied by the time over which it acts.
We start with Newton’s second law, which tells us:#Force equals mass times acceleration — or simply,#F equals m a.
Now, acceleration is defined as the change in velocity divided by the change in time — so we write:#a equals delta v over delta t.
If we substitute that into Newton’s law, we get:#F times delta t equals m times delta v.
This equation shows us something powerful:#Impulse — that’s force times time — equals the change in momentum.
tells us that impulse causes the change in momentum.
It’s the way to model how mechanical systems respond to external inputs, and the idea applies to other input-output relationships as well.

slide8:
Building on what we just discussed, here we observe something quite elegant:
The shape of the force curve — whether it’s a tall narrow spike or a wide shallow push — doesn’t actually matter when it comes to total impulse. What truly matters is the area under the force-time curve.
This area represents the cumulative effect of the force. So, a small force applied over a long duration can have the same impulse as a large force applied for a short time.
In system terms, different kinds of inputs can result in the same response if their total impact — the area — is identical.
This idea is central to convolution: we’re not just tracking individual values, but the accumulated effect of an input across time.

slide9:
Now let’s extend that idea.
Impulse isn’t limited to rectangular shapes. We can see in this graph that different Gaussian curves — whether wide or narrow, flat or sharp — can all serve as impulse-like inputs.
What matters is the area under each curve. This area represents the total impulse delivered by the force, regardless of how the function looks. As long as that area remains the same, the impact on the system is the same.
This insight is critical in both physics and engineering: we often abstract away the exact shape of a signal and focus instead on its integrated effect — the impulse.

slide10:
This brings us to an important mathematical idea — the limiting process, which gives us a way to model an ideal impulse.
Let’s define a simple function — we’ll call it d tau of t.
It’s equal to one divided by two tau when t is between minus tau and tau.
And it’s equal to zero everywhere else.

So again:
d tau of t equals one over two tau, if t is greater than minus tau and less than tau.#Otherwise, it’s zero.
Now, as tau gets smaller and smaller, this function becomes taller and narrower, but the area under the curve remains exactly one.
That’s the key. As we shrink the width to zero — and the height grows without bound — we approach the Dirac delta function.
Mathematically, we say:
The limit of d tau of t, as tau approaches zero, is zero everywhere except at t equals zero.#And at the same time,#The limit of the area under the curve — capital I of tau — is equal to one.
In other words, the delta function is infinitely narrow, infinitely tall, and yet has a finite area of one.
This makes it a perfect tool for modeling ideal impulses, especially when we need to represent quick, concentrated energy or a sharp event in time.
And this idea will become central as we move into our discussion of convolution, where impulse responses help us understand how systems behave in response to various inputs.

slide11:
Let’s take a moment to reflect on why the limiting process is essential when defining an impulse.
You might wonder — if we already know an impulse gives us the desired effect, like setting a golf ball in motion, why go through the trouble of idealizing it mathematically? The reason lies in precision. By applying the limiting process, we create a perfect, universal model of an impulse, one that’s as rigorous as the definition of a derivative in calculus.
Think of how we define a derivative: we take a small change, and by letting that change approach zero, we achieve an exact result. We’re doing something similar here with the Dirac delta function.
Now, regarding the shape of the delta function — does it matter? In fact, no. What matters is the area under the curve, not how the curve looks. It’s like receiving payment: whether it comes in one sum or in smaller installments, the total value is what counts.
Interestingly, if you start worrying about the exact shape without measuring it, you’re in uncertain territory. Without measurement, all shapes are possible — a bit like the probabilistic world of quantum mechanics! Only when we observe or measure do we know what’s truly there. This connection offers a fascinating lens on how we idealize and model impulses in system theory.

slide12:
Let’s take a deeper look at the Dirac delta function — one of the most powerful concepts in system analysis.
The delta function is what we call a generalized function, or in mathematical terms, a distribution. It helps us model a perfect impulse — something that delivers a concentrated effect at a single moment in time.
Now, here’s how we describe it mathematically:
We say that delta of t minus t naught equals zero for all values of t not equal to t naught.#And, when we integrate this delta function from minus infinity to infinity, the result is one.
What does that mean?
It means the function is zero everywhere — except at one specific point in time, which we call t naught. At that single point, it has an infinitely sharp spike, but the area under the spike is exactly one. That area is what we call the impulse.
So even though we can’t see or draw the spike perfectly, we know that its total effect — the impulse — is one. That’s what matters. The effect is important, but the exact shape doesn’t matter.
Now, let’s shift to the discrete version of the delta function — something we use in digital systems.
We define delta of n like this:
It equals one when n is zero,#and it equals zero for all other values of n.
In other words, this function is zero everywhere — except at n equals zero, where it has a value of one.#It’s a single pulse — like a quick tap — at a

specific point in time.
We can also shift this pulse. For example, delta of n minus two just moves the pulse to position n equals two.
This simple pulse is incredibly useful. It forms the building block for discrete-time convolution, which we’ll talk about next.
So both the continuous and discrete versions of the delta function help us analyze how systems respond to input — whether we’re dealing with real-world signals or digital data.

slide13:
Before we explore how the delta function models sampling, let’s take a quick look at the integral mean value theorem — a fundamental concept from calculus.
Imagine we have a continuous function plotted over the interval from a to b. The area under the curve — shaded in green on the slide — is found by integrating the function from a to b.
Now, here’s what the mean value theorem says:#Somewhere between a and b, there’s at least one point, which we’ll call c, where the height of the function at that point — f of c — multiplied by the width of the interval, gives you the same area as the one under the curve.
It’s like finding a rectangle with the same base and the same area — it represents the "average" value of the function across the interval.
This concept of "average value over a short interval" will help us understand how the delta function works when it comes to sampling a continuous function.

slide14:
Let’s now connect this idea directly to the delta function and its role in sampling.
Imagine you have a continuous function — let’s call it f of t. Now, if you multiply this function by a shifted delta function, specifically delta of t minus t naught, and then integrate, something powerful happens:
You end up extracting the value of f at t naught. In other words, the delta function acts like a perfect sampling tool.
Mathematically, this is shown through a limiting process. As the width of the delta function shrinks toward zero, its height increases — but the total area remains one. That’s key.
Inside this narrow window around t naught, the function f is essentially flat — and using the mean value theorem, we know there must be some point where the function's value equals the average over that interval. As the window becomes infinitesimally small, that point converges exactly to t naught.
So what do we get?
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 is called the sampling property of the delta function. It’s as if the delta function "reaches into" the function and pulls out the value at a single point.
Even more fascinating — we can actually rebuild any continuous function using this idea. By stacking many delta functions across time, each weighted by the function’s value at that point, we can reconstruct the entire function. Think of it like slicing the function into tiny pieces — each slice tells us something, and when you add them all together, you get the full picture again.

slide15:
Let’s now think about how this idea translates to the discrete case.
You can visualize a discrete function as a collection of rectangular bars — sometimes called gate functions. These bars represent values at specific, evenly spaced points — like sampling a continuous function at fixed intervals.
Here you can see the mathematical expression in this slide.
This function is defined piecewise as follows:
It equals zero when the absolute value of t is greater than one-half,
It equals one-half when the absolute value of t is exactly one-half,
And it equals one when the absolute value of t is less than one-half.
In other words, the rectangular function is centered at zero, has a total width of one, and represents a simple, finite pulse — like a short “on” signal.
At the bottom of the slide, we define the discrete delta function, written as delta of n.

It equals one when n is zero, and zero otherwise.
So delta of n is a single spike located at n equals zero.#If we shift it, like delta of n minus two, the spike moves to n equals two.
This function serves as the discrete version of the continuous delta function.
By shifting and scaling these unit impulses, we can represent any discrete signal as a sum of these basic components — just like reconstructing a picture with pixels.

slide16:
Here, you can see the idea visually.
Imagine the original continuous function shown in gray.#We can approximate it piecewise using constant slices — small rectangles, each with a height that matches the continuous function’s value at that point.
Each rectangle corresponds to a delta function, or a rectangular bar, scaled to match the original function’s height.
The key point is: the area under each rectangle is equivalent to the area under the corresponding delta function at that location.
In this way, everything ties together between the continuous and discrete perspectives.

slide17:
In fact, you can use discrete delta functions to represent any arbitrary discrete function.
Each value of the function can be represented as a scaled delta function at that point.
When we sum these scaled delta functions, we reconstruct the original discrete function.

slide18:
Now that we’ve developed an understanding of the delta function and its role in modeling discrete events, let’s take this idea a step further and see how we can use it to represent an entire discrete function.
Here we observe a powerful mathematical concept: any discrete function — let’s call it g of n — can be expressed as a weighted sum of shifted delta functions.
That is, we take the delta function, shift it to the appropriate position, and scale it by the function’s value at that point. When we add all these together, we recover the original function.
Here you can see the mathematical expression in this slide.
This formula tells us that each sample of g contributes one impulse, located at n equals k, and scaled by g at that k. When we add up all these scaled and shifted impulses, we rebuild the function g.
Let’s look at a specific example to make this concrete.
Suppose your discrete function consists of just a few nonzero points. The first point is at n equals 0 with value 1 — that’s just delta of n.
The second point is at n equals 1 with a value of negative 1 — so we write that as minus delta of n minus 1.
And the third point is at n equals 2 with a value of 1.5 — which becomes 1.5 times delta of n minus 2.
By adding these three impulses together, we effectively reconstruct the original signal, point by point.
This approach is not only elegant, it’s also foundational in signal processing and linear systems. It allows us to treat any discrete-time signal as a sum of building blocks — scaled and shifted impulses — making analysis and filtering much easier later on.
We’ll continue building on this idea when we introduce convolution, which is just the response of a system to this kind of delta-based representation.

slide19:
So far, we’ve explored both the continuous and discrete versions of the delta function.
Both approaches give us ways to express functions as combinations of basic building blocks.
Whether a system or phenomenon is fundamentally continuous or discrete isn’t always clear — think of water.

When you see a flowing stream, it seems continuous. But if you zoom in, you'll find discrete molecules.#And if you go even deeper, those molecules follow the rules of quantum mechanics, where a continuous probability wave describes behavior.
In engineering, we switch between these views depending on what's useful.#A summation represents the discrete case. An integral represents the continuous case.#They are really two sides of the same coin.
With this foundation, we're ready to move forward —#and explore how delta functions connect directly to shift-invariant systems,#which play a key role in imaging and signal processing.

slide20:
Let's take a look at this graph.
This impulse response, shown here as h of n, is our key piece of knowledge.
I'm not asking you to start from nothing.#I'm giving you the system's reaction to a standardized input — the delta function applied at time zero.
Once you have that, you're ready.
Because if I give the system a more general input — not just a single impulse, but any arbitrary shape —#you can still figure out the output.
So the question becomes:#How do you compute the output from this known impulse response and a general input?
That's exactly where convolution comes in.
Convolution is the operation that combines the input signal with the impulse response,#to produce the final output.
It's absolutely essential in imaging and signal processing.

slide21:
In a shift-invariant system, if you know the system's response to an impulse at one location, you can predict its response at any other location or time.
That's because shift-invariance means the system behaves the same way, no matter where or when you apply the impulse.
If you apply an impulse at a different time or position, you can infer the output by simply shifting the known response accordingly.
And because the system is linear, you have additivity and homogeneity.
If you scale an input, the output will be scaled in the same way.
If you add multiple inputs, the output will be the sum of individual outputs due to each input.
This is exactly why knowing the impulse response of a system tells you everything about how that system behaves — no matter what input you give it.
To summarize, for a shift-invariant linear system, given any input, we want to compute the output.#The critical information we rely on is the system's impulse response.#If we know how the system reacts to a delta function at time zero,#we can use that as a building block to find the output for any input.

slide22:
To be specific, now let's see how we actually compute the output y of n of a shift-invariant linear system to an arbitrary input x of n, formulated on the 1st line.
Suppose I give you a system characterized by its impulse response, written as h of n, which is shown on the 2nd line.#That means: if you feed the system a discrete delta function as input, the output will be h of n.#This response — h of n — is the system's character.
But what if you give the system a more general input, say x of n?#What will the output y of n be, which is what I just formulated on the 1st line, and a very important question.
Let's break it down step by step.
Since the system is shift-invariant, if you shift the input by k, the output also shifts by k, shown on the 3rd line.#So if the input is delta of n minus k, the output becomes h of n minus k.
Then, because the system is linear, if we scale the input by x of k,#Consequently, the output is also scaled by x of k.#That gives us x of k times h of n minus k on the 4th line.
Now, to get the total output, we add up all of these scaled and shifted responses.

Mathematically, this gives us the convolution sum, on the right-hand side of the last two lines:
y of n equals the sum over k from minus infinity to infinity of x of k times h of n minus k.
This is how the convolution is defined, and also why we need this concept.
It shows why the impulse response is so powerful — it's all you need to compute the system's output for any input.
Now, convolution might seem a bit tricky at first, especially because of the flipping and shifting involved —#that negative k inside h of n minus k could appear confusing.
But if you follow the logic step-by-step, the structure becomes clear.
This convolution concept is fundamental in linear systems and engineering —#and that's why we dedicate so much attention to understanding convolution.

slide23:
Now let's apply the same idea in the continuous case.
Again, you're given the system's response to a delta function, which we call h of t.
If we shift the delta to t minus tau, the output becomes h of t minus tau.#In other words, shifting the input shifts the output in the same way — that's shift-invariance.
Next, we scale the input by x of tau.#Because the system is linear, the output also gets scaled — giving us x of tau times h of t minus tau.
Finally, to build the full output for a general input signal,#we integrate over all time — summing these scaled and shifted responses.
This gives us the continuous-time convolution integral:
You can think of this as collecting all the weighted impulse responses across time,#and combining them to form the system's output, where we have utilized the sampling property of the delta function.
This is how linear, shift-invariant systems behave in continuous time —#and this operation, convolution, lies at the heart of modeling and understanding them.
Finally, we have assumed that all involved integrals or summations on the previous slide converge, so mathematically well defined.

slide24:
So whether we're dealing with discrete or continuous systems,#Convolution is the tool we use to compute the output.
In both cases, the idea is the same:
First, express the input as a collection of impulses.#Then, for each impulse, you already know the output — that's the impulse response.#Finally, sum or integrate all of those individual responses to get the final output.
This process is captured using these two key equations:
We can see here discrete and continuous mathematical formulations.
But I know from experience that convolution can feel confusing at first.#That's why it really helps to look at a concrete example.
Let's move on to one now.

slide25:
Let's consider a hands-on example using two discrete functions, each with five data points.
Now imagine holding out both hands:#your left hand represents one function — let's call it x of n,#and your right hand represents the other — this will be h of n, the impulse response.
Because of that negative k in the convolution sum,#one function needs to be flipped — think of it like flipping your right hand so you see the back of it.#That's how we visualize h of n minus k.
Now we shift this flipped function across the input and calculate the output step-by-step.
For n equals zero, you align the flipped function with the origin,#multiply the overlapping points, then sum the products —#this gives you the output y of zero.
For n equals one, shift the flipped function one step to the right,#multiply again, sum the result — and you get y of one.
You keep shifting, multiplying, and summing for each value of n.
You'll notice that when the two functions fully overlap,#you get the largest

value in the convolution.#And as they slide past each other, the overlap decreases,#so the convolution value gradually falls back to zero.
This visual analogy with your hands is a powerful way to understand discrete convolution –#especially how flipping, shifting, multiplying, and summing all come together.

slide26:
Let’s assign specific numbers to this example.
Suppose on your left hand, your thumb has value 5, your index finger is 4,#then 3, 2, and 1 for the middle, ring, and little fingers.
On your right hand, the values go the other way – your little finger is 1, ring finger is 2,#and it increases up to 5 at the thumb.
Now, flip the right hand – just like we flip h of n in the convolution process –#then slide it across the left hand, computing the pairwise products at each shift,#and summing them to get each value of the output.
This is exactly what’s happening in the MATLAB code shown on the slide.
We define x as the vector [5, 4, 3, 2, 1]#and h as [1, 2, 3, 4, 5].
Then we call y equals conv of x and h#and finally, we plot the result.
The plot shows a symmetric peak, just as we expect – the center point reflects full overlap between the two sequences,#and the values decrease symmetrically as they slide past each other.
Also remember: when you convolve two sequences of lengths n 1 and n 2,#the result will have a length of n 1 plus n 1 minus one.
Practicing with simple examples like this helps you visualize and internalize the convolution process –#flipping, shifting, multiplying, and adding.
And in imaging systems, which are designed to be linear and shift-invariant,#convolution defines how the system responds to any input.
That’s why mastering this operation is essential.

slide27:
Let’s see another example – this time from medical imaging.
Imagine you’re injecting a contrast agent, like iodine, into the bloodstream to highlight blood vessels in an X-ray image.
Without contrast, the blood, vessels, and surrounding tissue can look very similar.#But when you add contrast, the vessels light up more clearly, making them easier to detect and interpret.
Now here’s the key part: as the iodine travels through the body, it moves along multiple paths.
Some paths are fast – like major arteries. Others are slower – like capillaries or microvasculature.#Each of these paths introduces a different delay before the contrast reaches the detector.
If you could inject an ideal, instantaneous impulse of contrast – in other words, a perfect delta function –#then what you would see is a collection of impulse responses, labeled here as H 1, H 2, H 3, and so on.
Each of these responses corresponds to a different path, and each arrives at a different time.
But in reality, injections are not perfect impulses. They take time.
This means the actual contrast input is more spread out – and the measured output at the gamma camera#becomes a sum of all these delayed responses, weighted by how much contrast passes through each path.
In other words, the system’s output is a convolution of the injected input with the system’s impulse response.
This illustrates exactly why convolution is so important in medical imaging:#it helps us model and interpret how signals – like contrast or radiation – travel through complex biological systems,#and how we capture those signals with imaging devices like a gamma camera.

slide28:
In reality, the contrast isn’t delivered as a perfect impulse –#it’s injected over a period of time, with more at the start and less later, forming a time curve.
We can think of this non-ideal input as a sum of small impulses occurring at different times –#each one representing a fraction of the total injected dose.
For each of these small impulses, we already know the system’s response –#that’s

the impulse response we’ve discussed.
So what do we do?
We scale each impulse response based on how much contrast was injected at that moment,#shift it in time according to when it was injected,#and then add all the individual responses together.
What we get is the system’s total response — the final signal that’s measured by the detector.
This is a perfect visualization of discrete convolution in a real-world setting.#It shows how imaging systems handle non-ideal, time-varying inputs using the exact same principles we’ve learned.
It may take some time for these ideas to fully sink in,#but they form the foundation for understanding how imaging systems process signals and generate useful images.

slide29:
Now, let’s work through a more abstract — and slightly more mathematical — example.#I like to call this a minds-on example.
This example is based on continuous convolution, and while it’s not difficult, it does require you to think like in calculus.
So let’s take our time with it.
With convolution, the basic process is this:#We flip one function, shift it, multiply them point by point, and then integrate over the region where they overlap.
At first, that might sound a little mechanical. So let’s break it down clearly, step by step.
We’re working with two functions here:
f of t, shown in red. This is a triangular function — it starts at zero, rises to a peak at t equals 2, and then drops back to zero.
g of t, shown in purple. This is a rectangular pulse that spans from t equals minus 2 to plus 2, with a constant value of 3.
Now, because g of t is symmetric, flipping it across the vertical axis doesn’t change its shape —#which actually makes our job easier.
So we’ll flip g to get g of t minus tau, and then slide it across f of tau,#evaluating the convolution at each position t.
Here’s the key part:#Both functions are compactly supported, meaning they’re only non-zero over a specific range.#That means the overlap only happens during certain intervals.
So to compute the convolution, we’ll handle it case by case — based on the value of t.

slide30:
👉 Case 1: When t is less than −2
At this stage, the flipped and shifted version of g, written as g of t minus tau,#is positioned so far to the left that it does not overlap with f of tau at all.
That means the product of the two functions is zero everywhere,#so the convolution result y of t is also zero.
This is our first insight:#No overlap means no contribution to the integral, and the output is zero.
Case 2: When t is between −2 and 0
Now we begin to see partial overlap.
As g of t minus tau slides to the right, part of it starts to enter the region where f of tau is non-zero.
Let’s break this down mathematically:
Over the region where they overlap, f of tau is defined as#f of tau equals negative tau plus 2, valid from tau equals 0 to tau equals 2.
And the flipped rectangle, g of t minus tau, still has a constant height of 3 wherever it overlaps.
So we’re integrating the product of these two functions across the overlapping interval.
The simplifies convolution becomes:#3 times the integral of (−tau + 2), from 0 to 2 plus t.
If you evaluate this integral, you get:#y of t equals negative three-halves t squared plus 6.

So the convolution result in this case is a parabolic curve, rising as t approaches zero.
This illustrates an important idea:#Convolution is constructed from slices of overlapping areas —#and in each case, we’re summing the product over those intervals of overlap.

slide31:
Case 3: When t is between 0 and 2
In this case, the flipped and shifted version of g, that is, g of t minus tau,#completely overlaps the triangular function f of tau.
This means we integrate across the entire support of f of tau,#from tau equals 0 to 2.
Within this interval:
f of tau equals negative tau plus 2
g of t minus tau is constant at 3
So the convolution becomes:#y of t equals the integral from 0 to 2 of 3 times (–tau + 2), d tau
When we evaluate this, we get:#y of t equals 6
So, for t between 0 and 2, the convolution gives a flat region with a constant value of 6.#This happens because the area under the product is the same across the entire interval.
Case 4: When t is between 2 and 4
Now, f of tau begins to slide out of the rectangular window of g of t minus tau.
We’re back to a partial overlap, but this time on the opposite side —#the right edge of g is moving past f.
The new overlapping interval runs from#tau equals t minus 2 to tau equals 2
Again, we use:
f of tau equals negative tau plus 2
g of t minus tau equals 3
So the convolution becomes:#y of t equals the integral from t minus 2 to 2 of 3 times (–tau + 2), d tau
Evaluating this gives:#y of t equals three-halves t squared minus twelve t plus 24
This result traces out the descending part of the convolution curve,#as the overlap shrinks toward zero.
Case 5: When t is greater than or equal to 4
At this point, g of t minus tau has completely moved past f of tau.
There is no overlap between the two functions,#so the convolution result once again becomes:#y of t equals 0

slide32:
Now, let’s bring all the cases together to write the full piecewise definition of the convolution result.
We’ve carefully stepped through five distinct intervals,#each showing how the amount of overlap between f of t and g of t determines the value of the convolution.
Putting everything together, we now have the complete solution.
Each case — from full overlap, to partial overlap, to no overlap —#contributes a segment of the final output.
This leads to a piecewise-defined function, expressed as follows:
y of t equals 0,   for t less than –2
y of t equals –three-halves t squared plus 6, for t between –2 and 0
y of t equals 6,   for t between 0 and 2
y of t equals three-halves t squared minus 12 t plus 24, for t between 2 and 4
y of t equals 0,   for t greater than or equal to 4
When we plot these expressions, we get a smooth, continuous curve —#it rises, flattens out in the middle, and then gradually falls again.#It forms a kind of bump, which is exactly what we expect#as g of t slides over and past f of t.
This example gives us a complete, mathematical walkthrough of continuous convolution —#from defining the problem, to analyzing it case by case, to combining all pieces into the final solution.
It beautifully demonstrates how convolution is built:#by accumulating the area under the product of two functions —#each one flipped, shifted, multiplied, and then integrated, step by step.

It’s a powerful blend of geometry and calculus,#and it’s absolutely foundational in signal processing and medical imaging.

slide33:
Now, let’s look at a practical example that ties all of this together – the RC circuit.
If you’re not familiar with electrical circuits, don’t worry.#For our purposes, you can think of this setup as a black box.#The input is a signal, and the output depends on the system’s behavior – specifically, its impulse response.
In this circuit, we have a resistor R and a capacitor C connected in series.#The voltage signal going in is x of t, and the voltage across the capacitor is the output y of t.
Now here’s the key point:#If we apply a delta function as the input – meaning a sharp, instantaneous impulse –#the output is the circuit’s impulse response, denoted h of t.
The h of t of the RC circuit is a decaying exponential. It tells us how the circuit reacts over time after a sudden input.
But what happens if we apply a more general input – like a square pulse or a step function?
Well, just like we learned, we can decompose that input into a sum of small impulse components.#Each tiny impulse generates a scaled copy of the impulse response, and the total output is the convolution of the input and h of t.
We can see this visually on the right:#Different inputs – like short pulses or longer ones – are convolved with h of t,#and the resulting output curves smoothly reflect how the circuit responds over time.
Try experimenting with this in MATLAB or Python.#Decompose the signal, apply convolution, and see how the shape of the output curve emerges naturally.
This kind of modeling is not just useful for circuits – it’s a powerful tool in medical imaging, signal filtering, and physiological modeling, where systems respond to inputs in time-dependent ways.

slide34:
Now that you understand how 1D convolution works,#it’s actually quite easy to extend the idea to two dimensions, especially when working with images.
In 2D convolution, you’re essentially applying the same process –#but now, instead of just one variable like t, you’re working with two:#x and y in the discrete case, or tau one and tau two in the continuous case.
Let’s take a look at the discrete form first.
We can see the formula at the top of the slide:
f of x, y convolved with g of x, y equals the double sum#over n 1 and n 2 of f at n 1, n 2 times g at x minus n 1, y minus n 1.
Just like in 1D, we flip the kernel, shift it, multiply, and sum –#but now we do this across both rows and columns.
In the continuous case, we do the same thing using integrals.#We integrate over both variables – tau one and tau two –#to compute the total area under the product.
So the core principle stays the same:#Flip, shift, multiply, and sum or integrate –#but now it happens in two directions at once, across a 2D grid or surface.
This extension is especially important in image processing and medical imaging,#where we often apply filters, masks, or system responses across 2D signals like slices or projection data.
Whether you’re summing or integrating,#convolution gives you a systematic way to model how a signal transforms when passed through a system.

slide35:
In imaging, 2D convolution is a powerful tool for modeling blurring.
Let’s take a look at how this works using a simple example.
On the left, we see an input image represented as a matrix of pixel intensities.#Each number corresponds to the brightness level of a pixel.
In the middle, we see a 3 by 3 averaging mask.#All entries are 1, and we divide the result by 9 –#which means we’re averaging the values in a 3 by 3 window.
This operation is a 2D convolution:#we slide the mask across the image, and at each position,#we multiply corresponding elements, sum them up, and store the

result in the center pixel of the output.
Let’s focus on the region that’s highlighted.
For the top-left example, the 3 by 3 neighborhood includes values like 1, 2, 3, and so on.#When we sum those values and divide by 9, we get the blurred output value, shown in yellow.
The same thing happens throughout the image.#Each pixel in the output image becomes the average of itself and its eight neighbors.
The result?#Sharp transitions are smoothed out, fine details are softened —#and the image takes on a blurred appearance.
This is a fundamental application of 2D convolution —#and it’s used not only for visual effects, but also as a preprocessing step in many medical imaging tasks,#like noise reduction and feature extraction.

slide36:
Let’s take a look at how blurring arises naturally in imaging systems — not just from digital filters, but from physical limitations.
Here we observe a key concept in optics called the point spread function, or PSF.
In theory, if you image a perfect point of light, it should appear as a sharp dot on the detector.#But in reality, that point gets blurred into a small disk — often called an Airy disk —#because of how light diffracts through the lens system.
This is the system’s impulse response in two dimensions.
So even if your input is just a single point — like a star in the night sky —#the output captured by your imaging system is a small, spread-out shape.
And when two bright points are placed too close together,#their point spread functions begin to overlap.#This overlap makes it difficult — or even impossible — to tell them apart in the final image.
This phenomenon is at the core of what we call the resolution limit in optical systems.
In fact, PSFs are not just a curiosity —#they’re mathematically modeled and measured,#and they define how sharp or blurry every part of an image will be.
So in medical imaging, astronomy, microscopy —#understanding the point spread function is essential to both image formation and image reconstruction.

slide37:
Now let’s talk about the reverse process of blurring — a technique called deconvolution.
When an image is blurred, it’s often because the original scene has been convolved with a point spread function — or PSF.#We saw earlier that even a perfect point source becomes a blurred spot due to system limitations.
Deconvolution is the process of trying to undo that effect —#to recover the original image from its blurred version.
You can think of convolution like a kind of advanced multiplication:#we blend the signal with the system’s response.#In that same spirit, deconvolution behaves like an advanced form of division —#we’re trying to undo the blending, and isolate the original input.
We can see that clearly in the example on this slide.#The left image is a blurred photo of a car — the license plate is unreadable.#After deconvolution, the right image restores the clarity — now we can read the plate.
This kind of technique is used not just in photography, but in microscopy, astronomy, and especially in medical imaging,#where resolving fine detail can be critical.
It’s important to note that deconvolution is not always perfect —#in real-world systems, noise and distortion make it a challenging problem.#But with good modeling of the PSF and proper regularization,#we can often greatly enhance image quality.

slide38:
Here’s the idea in action.
This slide shows how inverse filtering works in the frequency domain.
A sharp image, when convolved with a point spread function, results in a blurred image.
In the Fourier domain, convolution becomes multiplication.#So, the Fourier

transform of the blurred image equals#the product of the transforms of the original image and the PSF.
To reverse this, we perform division in the frequency domain —#this is the essence of inverse filtering.
In the red box below, the image on the right is the transform of the blurred image,#the center is the transform of the PSF,#and the left is the result of dividing them — recovering the original image spectrum.
Applying the inverse Fourier transform brings us back to the spatial domain.
This process forms the basis of many image restoration techniques.

slide39:
You might not fully grasp this yet — and that’s perfectly fine.
Once we cover Fourier analysis, the connection between convolution and deconvolution will become much clearer.
For now, think of convolution and deconvolution as the advanced forms of multiplication and division,#just like integration is the continuous counterpart of summation.

slide40:
Returning to one-dimensional signals, convolution behaves like multiplication in several key ways:
Commutative: h convolved with f equals f convolved with h.
Associative: h convolved with the result of f convolved with g equals the result of h convolved with f, then convolved with g.
Distributive: h convolved with the sum of f and g equals h convolved with f plus h convolved with g.
These properties show that convolution is algebraically structured, and they will connect directly to Fourier analysis.
You’re invited to try a quick quiz:#If y of n equals h of n convolved with x of n,#prove that y of n minus k equals h of n convolved with x of n minus k.
This follows naturally from the meaning of convolution.

slide41:
We can even prove these properties. I won’t go through the full proof now, but it follows directly from how convolution is defined.
Here, we start with the standard convolution definition:#y of n equals the sum over k of x of k times h of n minus k.
By substituting k equals n minus capital K,#we change the index of summation and rewrite the expression in reversed order.
This leads directly to:#h convolved with x equals x convolved with h.
So, convolution is commutative — just like multiplication.

slide42:
A related concept to convolution is cross-correlation.
Mathematically, the key difference is in the sign of the shift.
In convolution, we compute the integral of f of x minus u times g of u.
In cross-correlation, it’s f of x plus u times g of u.
So, convolution flips one of the functions before shifting and multiplying,#while cross-correlation does not.
Both are useful in signal processing —#but for different purposes, like system response versus pattern matching.

slide43:
Let’s take a closer look at the graphical differences between convolution, cross-correlation, and autocorrelation.
On the left, we see convolution.#Function f is in blue, and function g is in red.#In convolution, g is first flipped horizontally, then shifted, multiplied with f, and finally integrated or summed.#As g slides across f, the overlapping area is computed at each step — producing the black output curve below.
In the middle, we have cross-correlation.#It follows the same steps as convolution, except there’s no flipping of g.#We simply shift g across f, multiply at each point, and accumulate the result.#So visually, the shapes are the same, but the symmetry is different — this matters in pattern alignment and signal comparison.

On the right, we see autocorrelation.#This is a special case of cross-correlation where the same function is compared with itself.#So g and f are the same, and we measure how well the signal matches its own shifted versions.#The result usually peaks at zero shift – showing maximum similarity with itself – and falls off on either side.

slide44:
Cross-correlation is deeply connected to the Cauchy–Schwarz inequality.
This fundamental result tells us that the inner product of two signals#is always less than or equal to the product of their norms.
Mathematically, this absolute value of the inner product is less than or equal to#the square root of the sum of all the a k squared#times the square root of the sum of all the b k squared.
Equality holds only when the two signals are proportional.
This is why cross-correlation works in this way:#the output is maximized when the signal and the template are aligned in shape –#a key idea in pattern recognition and signal detection.

slide45:
Here we observe a practical example of cross-correlation in action –#a technique known as matched filtering, widely used in signal detection.
In the top plot, the blue curve shows a received signal that includes both the desired pulse and a lot of background noise.#The red curve represents the matched filter – a reference copy of the signal we're trying to detect.
The bottom plot shows the cross-correlation output.#Notice the sharp peak centered around one second – this is where the received signal best aligns with the matched filter.#That spike tells us that the target signal is present at that location in time.
This is exactly what the Cauchy–Schwarz inequality predicted:#the inner product, or correlation, reaches its maximum value when the two signals are best aligned.
Matched filtering like this is essential in many fields:#from radar and sonar, to wireless communication, and even in medical imaging –#anywhere we want to pull a weak signal out of noisy data.
Cross-correlation gives us a way to do that precisely and efficiently.

slide46:
Let's take a look at how convolution helps with feature extraction – especially edge detection – which is fundamental in medical imaging and computer vision.
In this example, we're applying a Sobel filter, often used to detect edges in the horizontal direction.
We begin with the source image, represented as a grid of pixel intensity values.#The Sobel kernel, shown here as a 3-by-3 matrix, contains weights:#negative one, zero, and positive one across columns – designed to emphasize horizontal changes.
Here's what happens:#We align the filter over a 3-by-3 region of the input image.#Then, we compute the sum of products between the kernel weights and the underlying pixel values.
For instance, at this highlighted region, the computation looks like this:
–1 times 3, plus 0 times 0, plus 1 times 1,# plus –2 times 2, plus 0 times 6, plus 2 times 2,# plus –1 times 2, plus 0 times 4, plus 1 times 1.
When we add those up, we get –3.#That value becomes the output for the corresponding destination pixel.
Now, as the filter slides across the image – pixel by pixel – this process repeats.#Wherever there's a strong horizontal transition, the filter produces a large response.#Where the region is uniform, the response is small or zero.
This is how edge detection works:#It uses convolution to detect where intensity changes rapidly, allowing us to extract important structural features from an image.

slide47:
To detect edges effectively across all directions, we apply multiple convolution filters – each oriented differently.
The image on the left shows the original grayscale input. On the right, we see the result of applying the Canny edge detector, one of the most widely used edge

detection algorithms.
What makes the Canny method powerful is its multi-stage approach. It starts with smoothing the image to suppress noise, then applies gradient-based filters in horizontal and vertical directions. From there, it calculates the gradient magnitude and direction at each pixel.
The algorithm then performs non-maximum suppression to thin out the edges, and finally applies double thresholding and edge tracking by hysteresis to preserve only the most significant contours.
The result is a crisp, binary edge map — like the one shown here — which highlights boundaries of objects and fine details.

slide48:
To sum up:#We explored linear systems, especially shift-invariant systems.#We saw how convolution helps compute the system output.#We saw how cross-correlation relates to convolution and supports feature detection.
We learned how decomposing functions into impulses connects to these operations.
Convolution can feel tricky at first, but with practice, it becomes a powerful, intuitive tool.

slide49:
Now it’s time to apply what we’ve learned.
Suppose the time constant RC equals 1 second. Use MATLAB to generate the plots shown in the first row.
Your goal is to compute the convolution of the input signal with the system’s impulse response:
h of t equals R C times e to the power of minus t over R C, multiplied by the unit step function
Focus on how the input signal and the impulse response interact to produce the output.
Please submit:
– Your MATLAB script#– The resulting figures#– All combined in a Word document