slide1:
Welcome back, everyone. Let me remind you of the best way to get the most out of this course. First, make sure you read the assigned chapter before each lecture. Join the lecture with the preview in mind and listen carefully. Ask if you have questions proactively. Afterward, work through the homework and revisit the material again. This layered approach — reading, watching, listening, asking, practicing — really helps deepen your understanding.
Today, we're going to explore a very powerful concept in signal processing: the Fourier series. This topic takes us a step beyond convolution, both in complexity and in importance. It's a bit more challenging, but it's absolutely central to many applications in engineering, physics, and medical imaging. So, I'll do my best to break it down and highlight the key ideas so that you can follow along with confidence.
Let's get started.

slide2:
Seems like we are going as per our schedule.
In some courses or textbooks, you might be handed a formula for Fourier analysis and told to apply it mechanically. And sure — you might be able to follow the steps, plug in numbers, and get the right answer. But that kind of approach often leaves you without a real understanding of what's happening under the hood.
Here, we're going to do things differently. Instead of just giving you the equations, we'll take the time to explore the why and how behind them. What does the Fourier series represent? Why does it work? And how does it help us understand complex signals?
This deeper understanding is especially important in a field like medical imaging, where you're not just solving math problems — you're working with real-world data that affects real people. So, our goal is not just to teach you the tools, but to help you develop a solid intuition for how and why they work.

slide3:
Now, let me share what I hope you'll take away from this lecture. We're going to approach the Fourier series from a geometric perspective, starting with the idea of high-dimensional space. This means thinking about vectors, angles, distances, and inner products — not just in two or three dimensions, but in spaces with many dimensions.
When we combine all of these geometric tools, we begin to see Fourier series and Fourier transforms not as abstract formulas, but as structured transformations — transformations that have a clear geometric interpretation.
And that's powerful. Because once you can visualize what's going on, everything becomes more intuitive. You not only understand the formulas better, but you also remember them more easily. You know what each term is doing. You're no longer just running code or solving equations without knowing why they work. You can see the big picture. In our last lecture, we talked about linear systems and their behavior in both continuous and discrete settings.
We introduced convolution and briefly touched on concepts such as the inner product and cross-correlation. Some of you might have found those connections a bit abstract, and that's completely normal at first. But today, we'll start bringing those pieces together, using geometry as our guide.
Let's dive in.

slide4:
Let's now talk about a foundational concept that underpins much of what we're doing — the inner product, also known as the dot product.
Suppose you have two vectors, A and B. The inner product is calculated by pairing each element of vector A with the corresponding element of vector B, multiplying those pairs, and then summing the results.
In mathematical terms, if vector A has components A1 through An, and vector B has components B1 through Bn, then the inner product of A and B is the sum from i equals 1 to n of Ai times Bi.
Let's look at an example. Take vector A as 1, 3, negative 5, and vector B as 4, negative 2, negative 1.
We compute the inner product as follows:#One times four, plus#three times

negative two, plus#negative five times negative one.
That gives us 4 minus 6 plus 5, which equals 3.
Now, why is this important? Because when we perform convolution or cross-correlation, we are essentially performing inner products — over and over again. Each time we slide one signal over another and compute the inner product, we generate one output value.
This is exactly how matched filtering works in radar systems. When an incoming signal — such as an echo from an aircraft — aligns with a known pattern, the inner product reaches its peak, making it possible to detect the target.
From the perspective of linear systems, cross-correlation involves flipping and shifting one signal, and then computing a whole series of inner products. The result becomes the output response of a shift-invariant system. So understanding inner products is not just helpful — it's essential.
There's also a beautiful geometric interpretation. The inner product of two vectors equals the product of their magnitudes multiplied by the cosine of the angle between them.
In other words, the dot product of A and B equals the norm of A times the norm of B times cosine theta — where theta is the angle between the two vectors.
So, if the vectors point in the same direction, the angle is zero, and cosine of zero is one — giving you the maximum value.
If the vectors are orthogonal — that is, they are at 90 degrees — then cosine theta is zero, and so is the inner product.
And here's something really elegant: if you take the inner product of a vector with itself, you get the square of its length. That means the magnitude of a vector can be defined as the square root of its inner product with itself.
In summary, the inner product captures three things at once — length, angle, and alignment — all in a single operation. And that makes it the foundation for everything we'll do with Fourier series.
Here's another way to think about it — by drawing a biological analogy.
Think of the inner product like the base pairing in DNA. Just as DNA sequences are built from matched pairs of bases — adenine with thymine, guanine with cytosine — our inner product is built by matching one element from vector A with one from vector B. Each matched pair contributes to the total result.
And just like those base pairings carry the essential instructions for life, this mathematical pairing carries the structural foundation for data analysis.
The overall sum of those matched pairs — the inner product — gives us meaningful information about the relationship between two datasets or signals.
In many ways, you can think of the inner product as the DNA of data science.
It's what lets us quantify similarity, compute distances, analyze angles, and perform transformations. We'll keep coming back to this idea throughout the course, so keep it in mind as we go deeper into Fourier theory.

slide5:
Let's now revisit the concept of a shift-invariant linear system, which we introduced earlier in the course. This diagram gives us a step-by-step breakdown of how such systems behave — and how the inner product plays a central role.
Here's the key idea: once you're given the impulse response of a system — that is, how it reacts to a single impulse — and you know the system is both linear and shift-invariant, then you can compute the system's output for any input signal.
This is exactly where convolution comes into play.
As shown in the diagram, the input signal x of n can be represented as a weighted sum of shifted delta functions.#That is, x of n equals the sum over k, from negative infinity to infinity, of x of k times delta of n minus k.
Now, for each of those shifted deltas — delta of n minus k — the system responds with a shifted impulse response:#h of n minus k.
And this response is scaled by the input value at that shift, which is x of k.
So what do we do next?
We add up all those scaled responses. The result is:#x of n equals the sum over k, from minus infinity to infinity, of x of k times h of n minus k.
That's the standard convolution formula.
But here's what's really important:#When you compute the convolution at a particular time n, you are essentially performing an inner product.
You take the input signal and the flipped, shifted impulse response — multiply

them element by element — and sum the result.
This process is just like computing the dot product of two vectors.
This same concept is used in matched filtering — like in radar signal processing.
In this process, we take a known pattern — which we refer to as the filter — and slide it across the incoming signal.#When the filter aligns well with a portion of the signal, the inner product reaches a maximum value.#This peak indicates that a strong match has been detected.
Whether we call it convolution or cross-correlation, the operation always boils down to computing inner products.
So once again, the inner product isn't just a nice-to-have mathematical idea.#It is the fundamental engine behind convolution, filtering, correlation, and soon, our study of Fourier analysis.

slide6:
Let's now take a step back and look at the inner product from a geometric perspective. This viewpoint helps us build strong intuition, especially as we move into higher dimensions.
Start by imagining two-dimensional space. You have a vector A, which we can think of as a point in that space. Now, add another non-zero vector, let's call it vector B. Along with the zero vector — that's the origin — these two define a straight line. This line contains all the scalar multiples of vector B. In other words, the line L of k equals k times vector B, where k is any real number.
Now here's the key idea: we want to project vector A onto that line. In other words, we want to find the point on the line that's closest to vector A.
Mathematically, this means finding the value of k that minimizes the distance between vector A and k times vector B.
The expression shown here represents that squared distance. It's written as the sum, from i equals 1 to N, of the square of the quantity ai minus k times bi.
That's essentially the Euclidean distance, squared — but applied component-wise in vector form.
This is an extension of the classic distance formula you saw in high school geometry: the square root of x one minus x two squared, plus y one minus y two squared. But now, we're doing it in higher dimensions.
To minimize the distance, we take the derivative of that squared distance with respect to k, set it to zero, and solve for k. When we do that, we arrive at this important formula:
k equals the inner product of vector A and vector B, divided by the squared norm of vector B.
That gives us the scaling factor for the projection.#And then, the distance from the origin to the projected point is:
D equals A dot B, divided by the norm of B.
This leads us back to a central identity in geometry:
A dot B equals the length of A times the length of B times the cosine of theta, where theta is the angle between the two vectors.
So what does this all mean geometrically?
The inner product tells us how much of vector A lies in the direction of vector B.#It's the projection of A onto B, scaled by how aligned the two are — and that alignment is captured by the cosine of the angle.
If A and B are orthogonal — that means they're at a ninety-degree angle — then the cosine is zero, and so is the inner product.#But if they point in the same direction, the cosine is one, and the projection is at its maximum.
So this isn't just an abstract formula. It gives us a powerful visual interpretation of what the inner product is really doing — measuring alignment.
And this interpretation extends naturally into higher dimensions, which makes it especially useful as we transition into Fourier series.

slide7:
Let's now turn our attention to a very important and elegant result in linear algebra and geometry — the Cauchy–Schwarz inequality.
If you haven't yet walked through the proof of this inequality, I encourage you to do so. It's not just a theoretical result — it's a tool that offers deep insight into how vectors relate to each other.
Now, the inequality is written here in both summation and vector form.

You can see here.
In vector form, we write:#The absolute value of a dot b is less than or equal to the norm of a times the norm of b.
In plain terms, if you take the dot product of two vectors, its value will always be smaller than or equal to the product of their magnitudes.
Equality holds only when one vector is a scalar multiple of the other — in other words, when the vectors are perfectly aligned.
To make this more concrete, let's consider the two-dimensional version:#The quantity a squared plus b squared, multiplied by c squared plus d squared, is greater than or equal to the square of a times c plus b times d.
This is a specific case of the inequality and shows how it works with actual vector components.
Now, how do we prove it?
A typical approach involves setting up a quadratic expression that includes both vectors and a variable, x.#We write the sum over i of the square of the quantity a_i times x plus b_i.#This can also be written as the sum of a_i squared, multiplied by the square of x plus b_i divided by a_i
We set this whole expression equal to zero and solve the resulting quadratic equation.
This works because the square of a real quantity is always non-negative, so the sum equals zero only when all terms vanish.#That only happens when the vectors are linearly dependent — and that's what gives us the equality condition.
Now, let's connect this back to geometry. Remember from the last slide, the inner product of two vectors equals the product of their magnitudes times the cosine of the angle between them.
That is:#A dot B equals the norm of A times the norm of B times cosine theta.
Since cosine of theta is always between negative one and one, the dot product can never exceed the product of the magnitudes.#That's the geometric heart of the Cauchy–Schwarz inequality.
So this isn't just an algebraic result — it's a powerful guarantee about how closely two vectors can align.
And as we move into Fourier analysis, this principle plays a crucial role in how we represent signals, enforce orthogonality, and ensure numerical stability.
So keep this inequality in mind — we'll revisit it again when we start working with basis functions and Fourier coefficients.

slide8:
Let's now return to the Cauchy–Schwarz inequality, but this time through a more intuitive, geometric lens.
Earlier, we explored the algebraic version. Now, let's visualize what it really means.
Imagine two vectors, A and B, drawn as arrows in space. The inner product between them — also called the dot product — can be expressed as:
A dot B equals norm of A times norm of B times cosine of the angle theta between them.
Now, keep in mind: cosine theta is always between minus one and plus one. So unless theta is exactly zero degrees — which means the vectors are pointing in the exact same direction — the inner product is always strictly less than the product of the magnitudes. That's why the inequality holds.
But what about equality?
Well, equality happens when cosine theta is equal to one. In other words, when theta equals zero. Geometrically, that means vector A lies perfectly along the line defined by vector B. One vector is simply a scaled version of the other.
In mathematical terms, we say:#bi divided by ai equals c, for all values of i.
This means each component of B is a constant multiple of the corresponding component of A.
That constant, c, tells us how much we stretch or shrink vector A to get to vector B. When that relationship holds for every component, the two vectors are not only aligned — they are perfectly aligned, differing only in length.
This concept is extremely useful in signal processing. Take matched filtering, for example. Imagine you transmit a radar pulse. That pulse travels, reflects off an object — say, an airplane — and comes back. Because of factors like distance or material type, the return signal may be delayed or weakened, but its shape stays the same.

So how do we detect it?
We use cross-correlation. That means we slide the known signal over the incoming data and measure how similar they are. When the alignment is strongest — that is, when the cross-correlation reaches its peak — the two signals are most similar in shape.
This is where the Cauchy–Schwarz inequality comes in. When two signals align perfectly, their inner product is maximized. Cosine theta approaches one. That tells us: these vectors — or signals — are almost perfectly matched.
So this triangle diagram isn't just a piece of geometry. It’s a powerful mental image that shows how the inner product captures alignment. And equality — the case when the inequality becomes an equality — only happens when vectors point exactly in the same direction. Whether you're in two dimensions, three dimensions, or even much higher-dimensional space — the story remains the same. That’s the geometric heart of the Cauchy–Schwarz inequality.

slide9:
Now let’s take everything we’ve learned so far and generalize it to n-dimensional space — what we call “R n,” or more formally, R superscript n.
Suppose we have two vectors:#Vector V has components v i,#and vector W has components w i.
These indices — the little i — run from 1 to n, where n is the number of dimensions.#So V and W are points in high-dimensional space.#But just like in two or three dimensions, we can still calculate the distance between them.
That distance is given by this formula:#D squared of V and W equals the sum over i of v i minus w i, squared.
This is just a generalization of the Pythagorean theorem.
In two dimensions, it gives the diagonal of a square.#In three dimensions, it gives the diagonal of a cube.#In n dimensions, it gives the straight-line — or Euclidean — distance between two points.
Next, let’s talk about projection — projecting one vector onto the line defined by the other.#For instance, we can project vector V onto the direction of vector W.#To do this, we find the scalar value k that minimizes the distance between V and the scaled version of W.
Mathematically, that means minimizing the squared distance —#the sum over i of v i minus k times w i, squared.
The optimal value of k — the one that minimizes that distance — is given by:#k equals V dot W, over the norm of W squared.
This result comes directly from taking the derivative of the squared distance with respect to k,#setting it equal to zero, and solving.#And look what appears naturally in the formula: the dot product, or inner product.#So this isn’t a random definition — it’s rooted in geometry.
That brings us to another essential concept: angle and orthogonality.
We can also express the inner product as:#V dot W equals norm of V times norm of W times cosine theta —#where theta is the angle between the two vectors.
So when theta equals 90 degrees, cosine theta is zero, and the inner product is zero.#That’s the condition for orthogonality — when vectors are perpendicular.
Even in R n — in any number of dimensions — the inner product continues to define distance, projection, angle, and orthogonality.#These aren’t just abstract mathematical ideas.#They are the foundation for understanding how high-dimensional geometry works.#And they’re essential tools in areas like signal analysis, data science, and machine learning.
Finally, don’t forget about basis and dimensionality.
In two dimensions, you need two basis vectors — usually along the x and y axes.#In n-dimensional space, you need n basis vectors to describe any direction or position fully.
And in our upcoming discussion of Fourier series, we’ll see how basis functions let us express complex signals as combinations of simple building blocks — just like coordinates in R n.

slide10:
Now let’s focus on a familiar and intuitive case — three-dimensional space.#This diagram provides a clear geometric visualization that many of you have likely encountered before.
We’re looking at a vector — let’s call it vector A — situated in three-

dimensional space.#You can think of vector A as an arrow pointing from the origin to some point in space.
Alternatively, you can think of it simply as a collection of coordinates.
Either way, vector A is completely described by its three components —#x A, y A, and z A —#which correspond to its projections along the x, y, and z axes.
Now, here’s the key idea:#Each of these components represents how much vector A "points" in the direction of one of the coordinate axes.
If you project vector A onto the x-axis, you get x A.#Project it onto the y-axis, and you get y A.#Project it onto the z-axis, and you get z A.
This method of description is based on an orthogonal basis,#which simply means the axes are all perpendicular to each other —#at right angles.
And here’s the important part:#If you know x A, y A, and z A,#you have everything you need to reconstruct the entire vector A.
That’s because vector A is just the sum of its three projections —#each one scaled by a unit vector in the x, y, or z direction.
Mathematically, we write:#vector A equals#x A times i-hat,#plus y A times j-hat,#plus z A times k-hat.
Here, i-hat is the unit vector along the x-axis,#j-hat is the unit vector along the y-axis,#and k-hat is the unit vector along the z-axis.
This idea generalizes beautifully to higher dimensions.#Instead of working with x, y, and z,#you’ll have more axes — more basis directions —#but the principle stays the same.
And here’s where this becomes really exciting:
In Fourier analysis, we do something very similar,#but instead of projecting onto spatial axes,#we project onto a set of basis functions —#like sine and cosine waves.
So, what you’re seeing here in 3D#is not just about vectors in space.#It’s the same foundational idea that lets us#analyze and reconstruct complex signals#in the frequency domain.
Understanding this orthogonal representation#sets us up perfectly for the world of Fourier series.

slide11:
Previously, we saw how a vector in three-dimensional space can be represented using its projections onto the x, y, and z axes. Now, let’s extend this same idea to n-dimensional space — using orthogonal basis vectors to represent any arbitrary vector.
Let’s quickly review the 3D case first.#We have three standard unit vectors: along the x-axis, e x equals the vector 1, 0, 0, along the y-axis, e y equals 0, 1, 0, and along the z-axis, e z equals 0, 0, 1.#Any 3D vector — let’s say vector v — with components x, y, and z, can be written as a weighted sum of these basis vectors.#So, we write: vector v equals v dot e x times e x, plus v dot e y times e y, plus v dot e z times e z.#Each of these terms represents the projection of vector v onto one of the coordinate directions. Because our basis vectors are orthogonal and of unit length, we don’t need to normalize — the inner product directly gives us the component in that direction.
Now let’s generalize to n dimensions.#Suppose we have a vector — we’ll call it vector W — that lives in n-dimensional space. We can express this vector as a linear combination of n orthonormal basis vectors, denoted as e n, where n runs from 1 to capital N.#The formula looks like this: vector W equals the sum from n equals 1 to capital N of W dot e n times e n.
In words, for each basis direction e n, we project W onto that direction by computing the inner product W dot e n. That gives us the scalar coefficient for that direction. Then we scale e n by that coefficient and sum everything together.
This is a very powerful idea: you can express any vector — no matter how high the dimension — as a sum of projections onto orthonormal basis vectors.#And what’s even more powerful is that those basis vectors don’t have to be just the standard coordinate axes.#They could be sine and cosine functions, as we use in Fourier series. Or they could be eigenvectors in principal component analysis. Or wavelets, or anything else — as long as they are orthonormal.#So what we’ve described on this slide is the general framework of orthogonal representation in n-dimensional space. It’s the core mathematical idea that underpins many tools in signal processing, data analysis, and machine learning.

slide12:
Now here’s a very important point — and one that’s often overlooked.#A basis is not unique.
Let’s begin with a simple analogy in two-dimensional space.#In most problems, we use the standard coordinate system — with one unit vector along the x-axis and one along the y-axis.#In that system, the basis vectors are: 1, 0 for the x-direction, and 0, 1 for the y-direction.
But now imagine we rotate the coordinate system by some angle — let’s call it phi.
After this rotation, we have a new set of axes — shown in red on the diagram.#These new axes define a new basis, one that’s rotated by the angle phi.#And here’s the key: this rotated basis is just as valid as the original one.
Every point in the plane can still be represented using the new basis.#The point hasn’t moved — only the coordinate system has changed.#So the coordinates of the point relative to the new axes will be different, but the underlying geometry is the same.
Mathematically, the new coordinates, which we’ll call x-prime and y-prime, are computed from the original coordinates x and y using this rotation:#x-prime equals x times cosine phi, plus y times sine phi.#y-prime equals negative x times sine phi, plus y times cosine phi.
In matrix form, this looks like:#opcosine phi, sine phi; negative sine phi, cosine phi,
This is a classic rotation matrix — it rotates vectors by an angle phi in the plane.
And this concept generalizes beautifully.#In three dimensions, you can rotate about the x-axis, the y-axis, or the z-axis, producing new orthogonal bases in each case.#You can also use combinations of these rotations — just like the lower diagram shows — rotating around alpha, beta, and gamma angles.
And in n-dimensional space, the same principle applies.
So what’s the takeaway?#The basis you choose to describe vectors is not fixed.#You can rotate it, change it, or switch to one that better fits the problem.#This is especially useful in Fourier analysis, where we don’t just rotate axes — we build entirely new bases, like sine and cosine functions, to describe signals more effectively.
So while the geometry remains unchanged, the coordinate description of a point — or a signal — depends on the basis you choose.#That’s the power of flexible, non-unique bases in linear algebra and signal processing.

slide13:
Now let’s step back and summarize what we’ve built so far — the formal structure of a vector space, especially in R-n, that is, n-dimensional real space.#A vector in this space is just an ordered list of n numbers.
For example, we could define vector v as: v equals v1, v2, and so on, up to vn.#And similarly, another vector w would be: w equals w1, w2, all the way to wn.
Now, the inner product of these two vectors is written as:#v dot w equals v1 times w1, plus v2 times w2, plus all the way up to vn times wn.
This formula isn’t arbitrary. It’s grounded in both geometric intuition and practical applications — including signal detection and system analysis. But geometrically, it gives us the clearest picture of how vectors interact.
For example, if we take the inner product of a vector with itself, we get: v dot v equals the sum of the squares of its components. That is: v1 squared plus v2 squared, and so on, up to vn squared. Which is simply the squared length of the vector — or the norm squared.
So we write: v dot v equals double bar v double bar squared.
This gives us the magnitude of a vector, which is fundamental to defining distance, angle, and orthogonality — even in very high-dimensional spaces.
Now, let’s talk about basis.#In R-n, we use what’s called the natural basis. These are vectors of length one that point along each coordinate direction.#So we define:#e1 as 1, 0, 0, and so on,#e2 as 0, 1, 0, and so on,#and so on, up to en, which is 0, 0, and finally 1.
Each of these vectors is orthogonal to the others, meaning their inner product

is zero unless you're comparing a vector with itself.
So any vector v can be written as a linear combination of these basis vectors.
That is: v equals v1 times e1, plus v2 times e2, all the way up to vn times en.
And how do we get each of these coefficients?
Simple — by taking the inner product of v with the corresponding basis vector.
So we say: v dot ek equals vk.
This structure — combining orthogonality, projection, and basis expansion — gives us a clean and consistent framework for working with any dimensional space. Whether we're dealing with 3D vectors in physics, or a hundred-dimensional space in data analysis, the math works the same way.
And beneath it all is a visual, geometric idea: projecting, aligning, and reconstructing using components along axes.#That's the foundation of vector spaces.

slide14:
Let's close this section with a powerful and elegant idea — something that ties everything we've learned so far into a single, intuitive picture.#Every vector is a point.#That's right — a vector isn't just a list of numbers. It represents a location in space.
For example, in two-dimensional space, a vector with components x and y — written as x, y — can be seen as a point on the plane.#If this vector has unit length — meaning its magnitude is one — then it lies on the unit circle.#As you move around that circle, each position corresponds to a different vector, but all with the same length.
Now in three dimensions, the same idea holds.#A unit-length vector becomes a point on the surface of a unit sphere.#Again, the magnitude is one, but the direction can vary.#And every point on the sphere represents a different vector of unit length.
This concept scales beautifully to higher dimensions.
In general, if we're in n-dimensional space, the collection of all vectors of the same length — say, length r — forms an n-sphere.
Mathematically, we write this as: S n equals open curly brace x in R n plus 1, such that double bar x double bar equals r close curly brace.
This simply means: we're gathering all the vectors that are at a fixed distance r from the origin.#Each of these vectors is also a point in space.#They all have the same magnitude — r — but different directions.
So geometrically, a vector is just a point, and when we fix the length of these vectors, they trace out meaningful shapes like circles in 2D, spheres in 3D, or hyperspheres in higher dimensions.
This perspective becomes especially important as we move forward into signal processing and functional spaces, where even functions can be treated like vectors — and those vectors live in infinite-dimensional spaces.
But the core idea doesn't change.#A vector is a point. A point is a vector.#And the same geometry that applies to physical space applies to abstract spaces as well — including the ones we'll encounter next in our study of Fourier series.

slide15:
Let's now take the next big step in our journey —#a function can actually be treated like a vector.
Now, when you first think about a function — for example, y equals f of x —#You can imagine a smooth curve on a graph.#As x changes, you get a continuous stream of y-values,#and these values trace out a flowing shape across the domain.
That's the classical view — a function as a curve.
But here's a shift in perspective that changes everything.
Suppose you discretize the function.#That means instead of considering every possible x,#you just pick a few specific values:#x 1, x 2, all the way to x n.
At each of these points, you compute the function:#f of x 1, f of x 2, and so on, up to f of x n.
And now you have a finite list of numbers.#Those function values — can be written as a vector.#Each value becomes one component of that vector.
In the diagram here, you see a red curve — that's the original function —#and below it, green bars showing sample values.#If we sample the function at 7 locations,#we can think of those 7 values as a 7-dimensional vector.
Take more samples — say 700, or 7 million —#and the dimension of the vector

simply increases.#But conceptually, it’s still just a point in some high-dimensional space.
So here’s the key idea:#a function is just a vector — possibly a very high-dimensional one.#And that means we can apply all the tools of vector analysis to functions.#We can talk about the length of a function,#we can project one function onto another,#and we can compute angles between functions,#just like we did with ordinary vectors in R-n.
This insight is exactly what powers Fourier analysis.#In a Fourier series, we project a function onto a set of orthogonal basis functions —#functions like sine and cosine —#just as we projected ordinary vectors onto coordinate axes.
So let me leave you with this big takeaway:
If a vector is a point,#and a function is a vector,#then a function is also a point —#a point living in a high-dimensional function space.
That’s the foundation of what we’ll build next —#representing and analyzing functions using the powerful language of vectors.

slide16:
We’ve talked a lot about vectors — and how to compute their inner product#by multiplying corresponding components and then adding them up.#That’s the basic operation in vector space.
Mathematically, for vectors A and B, the inner product is written as:#A dot B equals the sum from i equals minus infinity to infinity#of a i times b i.
Now, let’s extend this idea to functions.#In functional space, instead of dealing with discrete components,#we’re dealing with continuous values defined over time or space.
So, instead of a summation, we use an integral.#The inner product of two functions, A of t and B of t, is defined as the integral from minus infinity to infinity#of A of t times B of t, d t.
That’s our functional equivalent of the vector dot product.#The interpretation is exactly the same:#we’re measuring how much two signals align —#whether they point in similar directions in an infinite-dimensional space.
Now here comes the big question:#Can we find a basis for functions, just like we do in vector spaces?
And the answer is yes.#One powerful basis uses something called the Dirac delta function —#written as delta of x minus a.
Let me explain.
There’s a key identity in signal processing that says:#f of t equals the integral from minus infinity to infinity#of delta of tau minus t, times f of tau, d tau.
What does this mean?
Well, the delta function is like a mathematical spike —#It’s zero everywhere, except at a single point.#So when we multiply f of tau by delta of tau minus t,#We’re isolating the value of f right at tau equals t.#And by integrating over all tau, we’re reconstructing the full function#by sweeping this spike across the domain.
Think of it this way:#each delta function — centered at a point a —#acts like a basis vector.#And each one is scaled by f of a —#the value of the function at that location.
This is just like saying:#v equals v one times e one plus v two times e two, and so on.
So yes, even in this continuous, infinite-dimensional space,#we’re still using the same principle:#Basis functions times coefficients gives you the full object.
And this paves the way for what’s coming next —#Fourier series, where instead of spikes,#we’ll use smooth, oscillating waves as our basis functions.

slide17:
Let’s now revisit a powerful concept —#representing a function as a sum of impulses.
Here’s the idea:#Instead of describing a function as one smooth continuous curve,#we break it down into a series of impulses, or spikes.
Each impulse is a shifted delta function —#meaning, a very narrow spike placed at a specific time location.
We scale each spike by the height of the function at that point.#In other words,

the amplitude of the spike equals the value of f of t#at that specific time t.
So we’re essentially saying:#f of t equals the sum of delta functions,#each multiplied by f at that location.
Look at the diagram.#The smooth gray curve represents the original function,#and the red arrows show the delta spikes.#Each spike points straight up, and its height#matches the value of the function at that moment.
Now, if you add all of these weighted impulses together —#you recover the entire shape of the original function.
This construction gives us a complete basis in one dimension.#And we can take this idea further.
In two dimensions, we use impulses defined over x and y.#Think of image pixels — each pixel acts like a 2D delta function,#placed at a specific location and scaled by intensity.
In three dimensions — like with medical images or 3D scans —#we build the signal from tiny volume elements, or voxels.#Each voxel is just an impulse located at some point in 3D space,#scaled by the value of the function there.
So no matter the dimensionality —#one D, two D, or three D —#you can think of a signal, image, or volume#as being built from a grid of impulses.
Each impulse contributes a small piece of the whole.#And together, they form the complete function.
That’s the power of thinking in terms of delta functions —#they give us a clean, mathematical way to build up#any function from the ground up, piece by piece.

slide18:
This impulse-based thinking becomes especially useful when we move into the world of digital imaging.
When you capture a picture, what you really get is a grid of pixels — discrete samples of brightness or intensity at specific locations. Each pixel can be thought of as a weighted impulse. A brighter pixel contributes more to the image, a darker pixel contributes less. And when the pixel size gets smaller and smaller — approaching zero — you begin to approach the continuous version: the delta function.
So in this view, an entire image is just a sum of impulses, one per pixel, weighted by intensity.
And this isn’t limited to 2D images.
In 3D medical imaging, for example, we deal with voxels — volume elements — which are essentially 3D pixels. Again, each voxel corresponds to an impulse in space.
Whether it’s a 1D signal, a 2D image, or a 3D volume, we can represent the data using impulse-like basis functions. That’s one perspective.
But there’s another view — just as powerful — and that involves sinusoidal basis functions.

slide19:
Sine and cosine waves show up everywhere in nature —#and there's a beautiful reason for that.#They arise naturally from circular motion.
Let’s take a simple example.
Imagine a point moving in a perfect circle —#like the moon orbiting the Earth at constant speed.
Now, if you shine a light on that motion#and watch the shadow it casts onto a flat surface —#say, the x-axis —#what do you get?
You get a sine wave.
That’s right —#as the point moves around the circle,#its horizontal position follows a smooth wave.#That’s sine.
And if you project onto the y-axis instead,#you get a cosine wave.
So these waveforms — sine and cosine —#are really just the shadows of circular motion.
That’s a profound idea.
It tells us that sine and cosine functions#aren’t just mathematical inventions — #they’re embedded in the geometry and physics of the real world.
They describe how things rotate, how waves travel,#how springs vibrate, how light and sound move.
And because of this,#sine and cosine waves form the foundation of something#we’ll explore in depth — the Fourier series.

So far, we’ve seen two major ways to represent functions:
First, as a sum of impulses —#like tiny samples or pixel points in space.
And now, as a sum of sinusoids —#waves that capture how a signal behaves across frequencies.
Both perspectives are incredibly powerful.#They help us describe, analyze, and reconstruct data#in ways that are both mathematically precise and physically meaningful.
In the next section, we’ll dive into how#these sine and cosine functions can be used#as a basis to build any function you like —#just like we built vectors from orthogonal components.

slide20:
Let’s now take a closer look at the sinusoidal basis — especially the sine and cosine functions.
In the plot, the red curve represents the sine of x — written as sine x — and the green curve shows the cosine of x.
There’s a key difference in their symmetry. Cosine is an even function. That means if you reflect it across the vertical axis — the y-axis — it stays the same. In other words, the cosine of negative x equals the cosine of x.
Sine, on the other hand, is an odd function. That means it’s symmetric about the origin. So the sine of negative x equals the negative sine of x.
But here’s something subtle. Whether a function is “even” or “odd” actually depends on where you place the origin. If you shift your coordinate system, the symmetry can change. So this classification is based on our point of view — not an absolute property of the function.
Now here’s the deeper idea: sine and cosine functions together form an orthogonal basis. That means you can use them to represent any reasonable function — just like we previously used delta functions or unit vectors.
This sets the stage for a powerful idea: instead of expressing a function as a sum of impulses, like a series of spikes, we can express it as a sum of waves — smooth, continuous, sinusoidal components.
And that’s exactly what we’ll explore next through the lens of the Fourier series.

slide21:
This representation makes that idea visual and intuitive.
Here, each person is holding a different sine wave — each wave has a different frequency or amplitude. When we add these waves together, we get a more complex waveform — something that might look jagged, or smooth, or anything in between.
And this is the foundation of the Fourier series. It tells us that any reasonable function — smooth or sharp — can be represented as a weighted sum of sine and cosine waves. These sinusoidal components differ in frequency, phase, and amplitude. But together, they can recreate signals of almost any shape.
So, just like we saw earlier that a function could be built from impulses, here we’re seeing that a function can also be built from waves. These are two complementary viewpoints — both valid, both powerful.

slide22:
Let’s now bring this idea to life with a demonstration.
What you’re seeing here is called spectral synthesis — it’s the process of reconstructing a function by adding together sine and cosine waves.
The red curve represents the signal or shape we want to recreate. Under the hood, we’re combining multiple sinusoidal components — each one a wave with its own frequency and amplitude.
As we begin, the approximation looks rough. But as we add more and more sine and cosine terms — especially ones with higher frequencies — the red curve starts to take shape. We can even mimic sharp edges or jumps, like you see in square waves, just by stacking enough high-frequency components together.
Here’s what’s important: each sine or cosine wave contributes a specific amount to the final signal. That amount is determined by its amplitude. If we were to graph those contributions — with frequency on the x-axis and amplitude on the y-axis — we’d get what’s known as the Fourier spectrum.
Think of it as the signal’s fingerprint. It tells us exactly how much of each frequency is present in the original function.

This is the core idea behind one-dimensional Fourier analysis. But it doesn't stop there. We can extend the same concept to two-dimensional signals, like images — and even to three dimensions, when we're working with volumes.
So what begins as a sum of simple waves turns out to be a powerful way of analyzing and reconstructing complex signals in any dimension.

slide23:
Let's now see what happens when we take everything we've learned and apply it to two-dimensional signals — like images.
Here we have a famous example — a grayscale portrait of Albert Einstein. At first glance, it seems complex and highly detailed. But if we perform 2D Fourier analysis, something remarkable happens: we can decompose this image into a collection of smooth, sinusoidal wave patterns.
In the middle panel, you see the amplitude spectrum — this tells us how much of each frequency is present in the image. Frequencies closer to the center are low — representing broad, smooth changes. As you move outward, the frequencies get higher — capturing fine details and edges.
On the right, you see an example of one such component: a 2D sinusoidal wave. These are like ripples in water, traveling in various directions — horizontal, vertical, diagonal — and at different frequencies.
Now if we zoom out, we can think of an image as a sum of many such 2D waves. That's what the grid of small tiles on the bottom left represents — each one is a different sinusoidal basis function. Some capture coarse patterns; others capture fine textures.
As we adjust the amplitudes and phases of these sinusoidal components just right — and add them all together — we reconstruct the original image.
This is the essence of Fourier analysis in two dimensions:
We take a complex image, and represent it using simple, smooth building blocks.
The result is a completely new way to think about images — not as pixels, but as combinations of spatial frequencies.
And this approach is not limited to images of Einstein. It works for any image — medical scans, photographs, microscopy, and beyond. That's why Fourier methods are so foundational in signal processing, image analysis, and modern AI.

slide24:
Let's take a moment to reflect on something deeper — a principle that reaches beyond signal processing and touches the foundations of physics and information itself.
In physics, we talk about the wave-particle duality — the idea that entities like photons and electrons exhibit both particle-like and wave-like behavior. Depending on how you observe them, they might act like tiny particles... or spread out like waves.
Interestingly, we see a similar duality in information science.
On one side, we have the particle view: information as discrete units — like pixels in an image, impulses in a signal, or delta functions in a mathematical model. This is the pixel-by-pixel, sample-by-sample perspective. It's granular and local.
On the other side, we have the wave view: information described in terms of global, smooth, continuous oscillations — sinusoidal waves of varying frequency, amplitude, and phase. This perspective captures structure across space or time — the big picture.
So when we represent a function — or an image, or a signal — we can choose either perspective.
Using delta functions or impulses, we build the signal by assembling localized pieces — one sharp component at a time.
Using sinusoidal functions, we build it using smooth, periodic waveforms that span the domain.
Both approaches are valid. In fact, they are mathematically equivalent when used with the right basis.
And this is the central idea of today's lecture:
A function can be viewed as a vector in a high-dimensional space — and depending on our basis, we can represent that function in many different ways.
You can think of a delta basis: where each basis function is an impulse at a specific location.

Or a sinusoidal basis: where each basis function is a sine or cosine wave with a different frequency.
These are just two different coordinate systems for the same space — and by projecting the function onto the basis elements, we retrieve coefficients that let us reconstruct the original signal as a linear combination.
So in summary:#The delta basis gives us a particle-like view.#The sinusoidal basis gives us a wave-like view.#And both help us understand information from complementary perspectives.
This concludes the first part of our discussion — building intuition through geometry and representation.
In the second half, we'll go deeper into Fourier series, where we work with periodic functions, and learn how to express them precisely in terms of sine and cosine components. We'll take a short break now — about ten minutes — and then dive right into the math.
See you shortly.

slide25:
Let's now begin our exploration of Fourier series by focusing on a special class of functions: periodic functions.
A periodic function is one that repeats itself over and over again at regular intervals. Formally, we say that a function f of x is periodic if there exists a value — which we call the period, and usually denote with capital P — such that: f of x plus P equals f of x, for all values of x.
This repeating behavior is what allows us to use the Fourier series to represent such functions.
Take a look at the top plot in this slide. It shows a one-dimensional example of a periodic function. It doesn't need to be a perfect sine or cosine wave — the key idea is that it repeats exactly after a fixed horizontal interval. In this case, that interval is P units.
Now look at the image below it. That's a real-world example — a two-dimensional periodic pattern. Patterns like this are common in materials science, digital imaging, or even textures. You can see how the structure repeats itself in both the horizontal and vertical directions — that's spatial periodicity in two dimensions.
So what is our goal?
We want to understand how any periodic function — no matter how smooth, jagged, or irregular — can be represented as a sum of sine and cosine waves.
This is the heart of the Fourier series. Even if a function looks complicated, we can rebuild it precisely by combining a collection of smooth sinusoidal waves — each with its own frequency, amplitude, and phase.
And here's the beautiful part:#If we get just one period of the function right, the rest of the function falls into place, because it's simply repetition.
So moving forward, we'll first focus on periodic functions in one dimension. We'll go through the math, develop the intuition, and build the full Fourier series representation. Then, once we understand that, we'll extend the same ideas to two-dimensional cases, and even three dimensions — which is especially useful for signals, images, and medical data.
Let's now dive into the mathematics and see how it all comes together.

slide26:
Now let's take the next step and define the Fourier series more precisely.
To keep things simple — and without losing any generality — we'll start with a function defined over the interval from 0 to 1. We call this the unit period.
The key assumption here is that the function repeats every unit interval. So if we understand what the function does between 0 and 1, we automatically understand it from 1 to 2, 2 to 3, and even from negative 1 to 0.
Later, we'll generalize to other periods — like from 0 to capital T, or from negative T over 2 to positive T over 2. But for now, let's build our intuition using this simpler interval.
So, what kinds of functions are we considering here?
We're working with real-valued functions that are square-integrable — that means when you square the function and integrate it over the interval from 0 to 1, the result is finite. If that integral blows up to infinity, the function is too wild, and we can't represent it with a Fourier series. But if it's finite, then

the function behaves well enough for our purposes.
Mathematically, we say such functions belong to the space L square of zero to one — that’s the set of square-integrable functions over the interval from 0 to 1. This is a Hilbert space, meaning we can treat functions like vectors in an infinite-dimensional space.
And just like vectors have a basis, functions in this space also have an orthonormal basis.
What does that basis look like?
It turns out the basis consists of the constant function 1, plus an infinite family of sine and cosine functions:
Square root of 2 times cosine of 2 pi n t
Square root of 2 times sine of 2 pi n t#for n equals 1, 2, 3, and so on.
The factor of square root 2 ensures that each function has unit length — meaning, its inner product with itself equals 1. That’s what makes the basis orthonormal.
So here’s the big idea:
Any square-integrable function over the interval 0 to 1 can be expressed as a sum of these basis functions.#And this sum is called the Fourier series.
Mathematically, we write:
f of t equals a naught over two,#plus the summation over n of a n times cosine of 2 pi n t,#plus the summation over n of b n times sine of 2 pi n t.
So, what do all these parts mean?
First, a naught is called the DC component. It represents the average value of the function over one period.
The a n coefficients are multiplied by cosine terms. These describe the even, symmetric parts of the function.
And the b n coefficients go with the sine terms. They capture the odd, asymmetric parts of the function.
In essence, what we’re doing is splitting the function into symmetrical and asymmetrical patterns, using sines and cosines as building blocks. This is what makes the Fourier series so elegant — it expresses any periodic function as a mix of smooth, familiar waveforms.
Now, each of these coefficients can be computed using inner products, which we'll define using integrals in the next slide.
But conceptually, this is just like decomposing a vector into components — except now we're working with continuous functions in an infinite-dimensional space, using smooth sine and cosine waves as our basis.
And that’s the beauty of the Fourier series:#Rather than reconstructing a function using spikes or samples, we rebuild it by layering together simple waveforms.
This is the core principle of Fourier analysis:#Break down the complex using the simple.#Use a family of smooth waves to represent arbitrary structure.
Next, let’s see how to actually compute those coefficients.

slide27:
Let’s now take a closer look at something we claimed earlier — that the set of sine and cosine functions, together with the constant function, form what we call an orthonormal basis.
Now, this idea might sound a bit abstract at first. But if we break it down, it’s very intuitive.
In the world of regular vectors — say, in two or three dimensions — we say vectors are orthogonal if they’re at right angles to each other. And if each one also has a length of one, we call them orthonormal.
The same idea applies in function space. But instead of using a dot product like we do for regular vectors, we use something called an inner product — which is defined using an integral. This inner product helps us measure both length and angle between functions.
Let’s walk through the specific relationships shown here.
First, the constant function — just the number one — has unit length. That means if we take its inner product with itself, we get one. So it’s normalized.
Now let’s see what happens when we compare the constant function with a cosine or a sine function. If we take the inner product of the constant with cosine of two pi times n times t, we get zero — for any positive integer n. The same is true for sine.

Why does this happen? Well, cosine and sine both oscillate above and below the horizontal axis, and over a full cycle, the positive and negative areas cancel out. So when you compare either of them with the constant, the result is zero — meaning they're orthogonal.
Next, let's compare sine and cosine functions with each other. Say, cosine of two pi times m times t and sine of two pi times n times t. Regardless of which frequencies m and n we choose, their inner product is zero. So sine and cosine are always orthogonal to each other.
Now here's where it gets a bit more subtle.
If we compare cosine functions with other cosine functions — or sine with sine — the result depends on the frequencies.
If the frequencies are different, meaning m is not equal to n, the inner product is zero. They're orthogonal.
But if the frequencies match — so m equals n — the inner product is one-half. That means the functions are not quite unit-length yet. They're orthogonal but not normalized.
To fix that, we just scale the sine and cosine functions by the square root of two. Then, their inner product with themselves becomes exactly one, making the whole set orthonormal.
So here's the big picture: we now have a set of functions — the constant, sine waves, and cosine waves — that are all mutually perpendicular and have unit length. Together, they form a complete orthonormal basis.
And once we have this basis, we can represent any periodic function as a combination of these building blocks. That combination is what we call the Fourier series, and the weights we use — the Fourier coefficients — tell us how much of each wave is present in the function.
Next, we'll explore how to actually compute those coefficients using integrals. Let's keep going.

slide28:
Before that, let's talk about harmonics — a term you've probably heard in music, physics, or signal processing. In the context of Fourier series, harmonics refer to these beautifully smooth sinusoidal components, each oscillating at a different frequency.
Let's unpack that.
The red curve you see here is the first harmonic, or the fundamental frequency. It completes exactly one full oscillation over the interval from 0 to 1. This is what we call the n equals 1 term in the Fourier series.
Now move down to the blue curve — that's the second harmonic, where n equals 2. It oscillates twice as fast — so you get two full cycles in the same interval. Then the green one, with n equals 3, completes three full oscillations, and so on. As you move further down the harmonic ladder — to n equals 4, n equal 5, and beyond — the frequency keeps increasing, and the wave becomes more and more tightly packed.
This is the key idea behind the Fourier series: you build complex functions using simple, smooth building blocks. And those blocks are just sine and cosine waves with integer frequencies.
Here's something subtle but very important: look at how all these harmonic components start and end at the same value — specifically, at zero. Not only that, but their rate of change — or slope — at the beginning and end is also smoothly matched. That's what allows them to blend so seamlessly when we stack them.
Why does this matter?
Because when you're representing a periodic function, you want the end of one period to flow naturally into the start of the next. If there's a mismatch — either in value or in slope — you get a jump, or a kink, which breaks the smoothness of the function. The harmonics are special because they ensure everything lines up perfectly — not just in position, but in motion.
That's why we call these functions harmonic. They don't just oscillate — they cooperate. And together, they can mimic anything from a square wave to the shape of a human voice, as long as you choose the right amplitudes and phases.
This is the genius of Fourier's insight: no matter how jagged or irregular a periodic signal may appear, you can always think of it as being composed of pure tones — harmonics — each contributing its rhythm to the total pattern.

Alright, now that we've built some intuition about harmonics, let's move on to the mathematical tools that allow us to compute their coefficients precisely.

slide29:
Alright, let’s take a closer look at how the Fourier series works over one complete cycle — that is, over the interval from zero to one.
As we’ve discussed, any well-behaved function on this interval — meaning it’s square-integrable — can be broken down into three types of components.
First, we have what’s called the DC component — written as “a naught divided by two.”
This term represents the average value of the function across the interval. It’s constant — it doesn’t vary or oscillate. You can think of it as the baseline level. For example, in an image, this might represent the overall brightness. In audio, it could be the steady background hum.
Then we have the even parts — these are built using cosine waves.#Each term looks like “a n times cosine of two pi n t,” where n is a positive integer: one, two, three, and so on.#Cosine is symmetric — if you flip it left to right, it looks exactly the same. That’s why we call these even components — they preserve that mirror-like symmetry.
Next come the odd parts — and these use sine waves.#Each term is of the form “b n times sine of two pi n t.”#Unlike cosine, sine flips sign when you reflect it, so it captures the antisymmetric — or odd — behavior of the function.
So to sum up:#We build the full function using a constant term, plus a sum of cosine waves for the even part, and a sum of sine waves for the odd part. All of these components are weighted by their respective coefficients — a naught, a n, and b n.
And here’s the elegant part:#Because we’re working with periodic functions, once you’ve reconstructed the signal over that one interval — from zero to one — the rest of the function just repeats automatically. It’s like tiling the pattern across the entire line.
You can see that idea illustrated in the image at the bottom of the slide.#The red waveform shows the reconstructed signal, and the lighter blue curves are the sine and cosine waves that contribute to it. When combined properly, they match the shape of the original function — and then extend it periodically to the left and right.
This kind of decomposition isn’t just theoretical — it has real meaning.
That constant term may represent a background level.
The even parts can capture symmetric patterns in your data.
And the odd parts can highlight asymmetries, like sharp edges or sudden transitions.
So the Fourier series gives us a powerful, structured way to describe a function using just smooth, well-behaved waves — each one tied to a specific frequency and type of symmetry.
Alright, now let’s move on and see how we can compute these coefficients — the values of a naught, a n, and b n — using projection. That’s coming next.

slide30:
Before that, here’s a really elegant and surprisingly powerful idea in mathematical analysis — something that’s often overlooked, but incredibly useful in practice.
No matter what function you're dealing with — call it f of x — you can always split it into two parts:#an even part, and an odd part.
And here’s how that works.
We define the even part, denoted f e of x, as#“f of x plus f of negative x, all divided by two.”
Mathematically, that’s:#f e of x equals one-half times the sum of f of x and f of minus x.
If you plug in negative x into this expression, you’ll get:#f e of minus x equals one-half times f of minus x plus f of x —#which is exactly the same as f e of x.#So this part is symmetric — it satisfies the condition of an even function.
Then, we define the odd part, written as f o of x,#by subtracting instead of adding. So you get:#f o of x equals one-half times f of x minus f of negative x.
Now, if you evaluate this at negative x, you’ll find that:#f o of negative x

equals negative f o of x.#That’s the signature of an odd function — it’s antisymmetric about the origin.
So here’s the beauty:#Every function, no matter how it looks, can be written as the sum of its even and odd parts.
In other words,#f of x equals f e of x plus f o of x.
This is a universal decomposition — and it's not just a neat identity.
It plays a critical role in understanding symmetry in signals, especially in Fourier analysis, where even and odd functions relate directly to cosine and sine components, respectively.
So remember this trick.#Whenever you’re working with a function — especially in signal processing or image analysis — it’s often helpful to ask:#What’s its even part? What’s its odd part?#And how can we take advantage of that symmetry?
We’ll put this idea to use shortly when we calculate the Fourier coefficients.

slide31:
Now that we understand the structure of the Fourier series — with its constant term, cosine terms, and sine terms — the next question is:
How do we actually calculate the coefficients?
These coefficients — which we call a n and b n — are essential.#They tell us exactly how much of each sine or cosine function we need in order to construct the original function.#Think of them like weights — how much of each frequency should be included in our series.
So, how do we find them?
The answer is beautifully simple — we use inner products.
Remember our earlier discussion: in vector spaces, you can isolate the contribution of a basis vector by projecting onto it.#The same logic applies here in the world of functions.#Because sine and cosine functions are orthogonal, we can project the function onto each basis element to extract the corresponding coefficient.
Let’s look at this process in the context of the unit interval — from zero to one.
First, to get the a naught coefficient — that’s the DC or constant component — #we take the inner product of f of t with the constant function 1.
This gives us:
a naught over 2 equals the integral from 0 to 1 of f of t d t.
So, it’s just the average value of the function over that interval.
Next, for the cosine coefficients, a n,#we multiply f of t by the cosine of 2 pi n t, and integrate over the interval from 0 to 1.#Then, we multiply the result by 2.
So we have:
a n equals 2 times the integral from 0 to 1 of f of t times cosine of 2 pi n t, d t.
In the same way, to compute the sine coefficients, b n,#we do the same thing — but using sine instead of cosine.
So the formula is:
b n equals 2 times the integral from 0 to 1 of f of t times sine of 2 pi n t, d t.
And that’s it.
Each coefficient is computed by projecting the function onto its corresponding basis function — either constant, cosine, or sine.
This is the beauty of orthogonality:#Each projection is independent of the others. So we can break the function into clean, non-overlapping components.
It’s like using X, Y, and Z axes in 3D space to describe a vector — but here we’re in an infinite-dimensional space, and our axes are made of smooth waveforms.
Now that we know how to compute these coefficients, we’re ready to apply Fourier series to real examples — and see how they reconstruct signals, even sharp or irregular ones, using only smooth sine and cosine waves.

slide32:
Let’s now walk through a concrete example to see how Fourier coefficients are actually computed in practice.
Suppose you’re interested in finding a particular coefficient from the Fourier series — let’s say the coefficient b 3.#This is the number that scales the sine

term with frequency 3 — in other words, sine of 2 pi times 3 t.
So the question is:#How much of this specific sine wave is present in the function f of t?
To answer this, we use a technique we've now seen several times — the inner product.
Here’s what we do:#We take the inner product of f of t with sine of 2 pi times 3 t.
Now keep in mind — this sine function is one of our basis elements.#It’s a fixed, known waveform. The function f of t is the one we’re analyzing.#And because the basis functions are orthonormal, a wonderful thing happens:
When we take this inner product, it acts like a filter.
It scans across the entire Fourier series of f of t, and picks out just the piece that exactly matches sine of 2 pi times 3 t.
All the other terms — the constant term, the cosine terms, and the sine terms with different frequencies like#sine of 2 pi t, or sine of 2 pi times 2 t, or sine of 2 pi times 4 t —#all of those vanish. Their inner products are zero, because they’re orthogonal to sine of 2 pi times 3 t.
The only term that survives is the one we care about — the one that involves b 3.
So this inner product tells us, precisely and cleanly, what b 3 must be.
And this isn’t just a lucky trick.
It’s a fundamental feature of the Fourier basis —#Orthogonality ensures that each coefficient can be extracted independently, with no interference from the others.
In short:#To find any Fourier coefficient — whether it’s a n or b n —#You simply project the function onto the corresponding sine or cosine basis function.#That’s it.
And this is what makes the Fourier series so powerful.#It gives us a systematic, elegant way to peel apart the frequency content of a signal — one layer at a time.

slide33:
Let’s now practically apply this idea — focusing specifically on how we calculate individual Fourier coefficients.
Suppose we want to compute the coefficient b3 from the Fourier series of some known function — say f of t.
What exactly do we do?
Well, we take the sine function, sine of 2 pi times 3 t, which is one of our orthonormal basis functions, and we compute its inner product with f of t.#That means we integrate the product of f of t and sine of 2 pi times 3 t over the interval from 0 to 1.
That’s it.
The result of this integral is exactly the value of b 3.#No guessing. No curve fitting. Just a clean calculation using integration.
And this same approach works for any other sine coefficient.#To compute b n, you use the sine of 2 pi n t as your basis function, multiply it by f of t, and integrate from 0 to 1.
The same logic applies to the cosine coefficients — the a n terms.#To find those, you simply take f of t and compute its inner product with cosine of 2 pi n t.#Each integral gives you one specific Fourier coefficient.
Now why does this work so beautifully?
It’s all thanks to the orthonormality of the sine and cosine functions.
When we take an inner product between f of t and any basis function — say, sine of 2 pi times 3 t —#only the matching sine term in the Fourier expansion contributes to the result.
All the other basis functions — like the cosines, or sine terms with different frequencies — are orthogonal to this one,#so their inner products are zero and they vanish from the computation.
This is exactly like projecting a vector in three-dimensional space.
If you want the x-component of a vector, you project it onto the x-axis.#For the y-component, you project onto the y-axis.#Same idea here — but now, instead of axes, we have sine and cosine waves.#And instead of finite dimensions, we’re working in an infinite-dimensional function space.
So again, orthonormality makes everything work cleanly.#It gives us a precise,

mathematical way to extract each component of a function in the Fourier basis.

slide34:
Now we arrive at a full summary of the real form of the Fourier series.
What we've seen so far is that many well-behaved functions — especially those that are continuous and square-integrable — can be expressed as a weighted sum of sine and cosine waves.#And this formula at the top captures that structure precisely.
We start with the function f of t.#Then we write it as the sum of three components:
First, the DC component, which is a naught over 2. That's the constant or average value of the function — the non-oscillating part.
Next comes the cosine sum, where each term involves a coefficient a n times cosine of 2 pi n t.
And finally, we have the sine sum, where each term includes b n times sine of 2 pi n t.
These coefficients — a naught, a n, and b n — are what determine how much of each sine or cosine wave contributes to the reconstruction of f of t.
So how do we find them?
Well, we already know the answer from earlier in the lecture: we compute inner products.#That is, we take the integral of the product of our function f of t with each basis function, over the interval from 0 to 1.
For the constant term, a naught over 2, we simply integrate f of t from 0 to 1.
To get a n, we multiply f of t by the cosine of 2 pi n t and integrate.
For b n, we do the same with the sine of 2 pi n t.
Once you have these values, you plug them back into the series.#And when you sum it all up — the constants, the cosines, and the sines — you recreate your original function, or at least approximate it very closely.
It's almost magical.#You could be working with a smooth parabola, or a sharp triangular wave, or even a jagged, piecewise function — and yet, you can reconstruct it entirely using just sine and cosine waves. That's the power of the Fourier series.
Now, remember earlier in the course we looked at functions in terms of discrete particles or impulses — like pixel-based representations. That gave us a kind of particle view of functions.
But this approach gives us a wave-based view — continuous, smooth, and grounded in frequency content.
Both are valid. And each is powerful in its own context.
But here's a question worth asking:#Why would we ever want to use the complex form of the Fourier series?
Well, look at this real version. It's elegant, yes — but it involves three separate types of terms: one for the DC component, one for cosine, and one for sine.
By switching to the complex form, we can merge all of this into a single sum using Euler's formula, where complex exponentials capture both sine and cosine behavior simultaneously.
This makes the whole formulation more compact and symmetrical.
The math becomes cleaner. The algebra gets easier. And in engineering applications, it often leads to more efficient computations.
So while the real form gives us intuitive clarity, the complex form offers analytical elegance.
That's exactly where we're headed next.
Let's now explore how complex numbers help us take the Fourier series to the next level.

slide35:
Now let's take a step into the world of complex numbers, because they offer a beautifully compact and powerful way to handle Fourier analysis.
We begin with something that might feel a little strange at first — the definition of the imaginary unit, denoted by the letter i.
By definition, i squared equals negative one. That is:
i squared equals minus one.
Now, this seems counterintuitive, right? Because in the world of real numbers, squaring anything always gives you something non-negative. But here, we're

stepping beyond the real line, into a new dimension — one that allows us to define numbers that include this imaginary unit.
And that opens the door to complex numbers.
A complex number, typically written as z, is defined as:
z = x plus i times y,
where x is the real part, and y is the imaginary part.#Geometrically, you can think of this as a point on the complex plane — with x on the horizontal axis and y on the vertical axis.
Now, for every complex number, we can define something called its conjugate.#If z is x plus i times y, then the conjugate of z, written with a bar over it, is: x minus I times y.
This is like flipping the point across the real axis — mirroring it vertically.
Now, let's look at how we add and multiply complex numbers.#The rules are very similar to regular algebra — you just have to remember that i squared is negative one. That's the only twist.
So, for instance, if we add two complex numbers, we just add their real parts and their imaginary parts separately.#If we multiply them, we expand the product just like you would with binomials — but we substitute i squared with minus one when it appears.
From here, we can also extract the real and imaginary parts of a complex number using its conjugate:
The real part is the average of the number and its conjugate — that's z plus z bar divided by two.
The imaginary part is the difference between the number and its conjugate, divided by 2 i.
Now, you might be wondering — how does all this tie back to Fourier analysis?
Well, just as we defined inner products for real-valued functions, we can do the same for complex-valued functions.#And when we do, we often need to deal with conjugates inside integrals.
Here's the rule:#If you take the complex conjugate of an integral — say, the integral of f of t times g of t — that's equal to the integral of f conjugate times g conjugate.
This is especially important when we compute energy, projection, or coefficients in complex signal spaces.
Also, in engineering and physics, we frequently use the star notation — an asterisk — to indicate conjugation. So if you see a star on the outside of an integral, that means we're conjugating the whole result.#Likewise, when it's applied to the functions inside, it means we're conjugating those functions individually.
So why go through all of this?
Because working with complex numbers — especially when combined with Euler's formula, which we'll see shortly — allows us to unify sine and cosine into a single, elegant expression. That's a huge simplification.
And that's the real power of this approach:#Cleaner math, fewer terms, and deeper symmetry.
So with that, we're now ready to look at the complex form of the Fourier series, which brings everything we've learned into one compact representation.

slide36:
When we work with complex numbers, there are two very common ways to describe them:#the Cartesian form and the polar form.#Both are fully equivalent, and they each offer a different way of thinking about the same quantity.
Let's begin with the Cartesian form. This is what we've seen already:#A complex number is written as x plus i times y, where x is the real part, and y is the imaginary part.#So we're essentially giving the horizontal and vertical coordinates of a point in the complex plane — just like specifying a location on a map.
Now, in engineering, we often replace the symbol i with j to represent the imaginary unit. This is purely to avoid confusion — because in electrical engineering, i is usually reserved for current.#But mathematically, i and j mean the same thing: they both satisfy the fundamental property that i squared equals negative one.
The other way to describe a complex number is through polar coordinates.#Instead of saying how far right and up the point is, we describe the number by its

magnitude and angle.
So what does that mean?
We draw a line from the origin to the point — that's the complex number — and call its length r. That's the magnitude, or modulus, and it's calculated using the Pythagorean theorem:#r equals the square root of x squared plus y squared.#This is also written as the absolute value of z.
Next, we describe the angle that line makes with the positive real axis.#That angle is called theta, and we compute it as the inverse tangent of y over x.#So now, we have two pieces: r and theta — the magnitude and direction.
With that, we can now express the same complex number in polar form.
In place of x plus i y, we write:
r times cosine theta plus i times r sine theta.#Or, factoring out the r:#r times the quantity cosine theta plus i sine theta.
This is the foundation for something beautiful: Euler's formula, which we'll talk about in the next slide.#Euler discovered that cosine theta plus i sine theta is equal to e to the i theta — giving us a powerful exponential representation of complex numbers.
Before we leave this slide, here's one more important result:#If you multiply a complex number by its conjugate — in other words, z times z bar — you get a purely real number.
Let's walk through it:#z is x plus i y, and its conjugate is x minus i y.#When you multiply them together, you get x squared plus y squared, which is just the magnitude squared.
This shows that the conjugate cancels out the imaginary part, leaving only a real value — something very useful in analysis and signal processing.
So just like vectors, complex numbers let us switch back and forth between rectangular and polar systems.
Each form brings its own advantages, and we'll soon see how this flexibility pays off — especially when we start rewriting Fourier series using complex exponentials.
Let's move to Euler's formula and take this further.

slide37:
So far, we've been casually treating the inner product as a simple dot product — point-by-point multiplication followed by a sum. That works just fine when we're dealing with real-valued vectors.
But when we step into complex vector spaces, things get a bit more subtle. In particular, we have to be careful with how we define the inner product. In the complex world, it's not enough to just multiply and add — we also need to take the complex conjugate of one of the vectors before doing so.
Let's break this down with a simple example.
Suppose we have two complex vectors. Let's call them X and Y.
Vector X has two components:#the first is a one plus b one times i,#and the second is a two plus b two times i.
Similarly, vector Y has components:#c one plus d one times i,#and c two plus d two times i.
Now, how do we compute the length of vector X?
In complex space, we use the sum of the squared magnitudes of each component. So the magnitude of X is the square root of#a one squared plus b one squared plus a two squared plus b two squared.
We do the same for vector Y — square the real and imaginary parts of each component, add them up, and then take the square root.
Now, here's the key idea.
If these two complex vectors — X and Y — are orthogonal, and we add them together to form a new vector Z, we expect the squared length of Z to equal the squared length of X plus the squared length of Y. That's just the Pythagorean theorem in vector space.
But for that identity to hold, one critical condition must be met:#the cross-term — the inner product between X and Y — must vanish. In other words, the inner product between X and Y has to be zero.
And this is where the complex conjugate becomes important.
In real vector spaces, we just take the dot product. But in complex spaces, we define the inner product as:#X dot Y conjugate.
That means, we keep vector X as it is, and we take the complex conjugate of each

component of Y before multiplying.
This adjustment ensures that everything works out mathematically:#we preserve orthogonality, we maintain proper lengths, and we uphold the geometry of the space.
So, whenever we're working with complex vectors — in signal processing, in quantum mechanics, or in Fourier analysis — this conjugation step is absolutely essential.
Now, don't worry about memorizing the formulas. That's why I added this green bubble here — it's just a reminder that this is more about understanding the concept than remembering every detail.
Once you internalize this idea — that conjugation is needed to define meaningful inner products in complex space — everything that follows, including the complex form of the Fourier series, will feel much more natural.
Let's now move forward and apply this idea in the Fourier domain.

slide38:
Now that we've introduced the concept of inner products for complex vectors, let's take a closer look at how the definition actually works — and more importantly, why it includes a conjugate.
In a real-valued vector space, things are simple. You multiply corresponding components and sum the results. But in a complex space, if we follow that same rule — just multiplying and summing — we may end up with a complex number. And that's a problem, because when we're measuring things like length, or comparing angles between vectors, we need real values.
So here's what we do instead.
We define the inner product of two complex-valued functions, let's say f of t and g of t, by integrating the product of f of t and the complex conjugate of g of t.
Symbolically, we write:#⟨f of t, g of t⟩ equals the integral from minus infinity to infinity of f of t times g star of t, d t.
That's our general definition. But often, we work over a specific interval — for example, from zero to one. In that case, we write:#⟨f of t, g of t⟩ equals the integral from 0 to 1 of f of t times g star of t, d t.
Now, why is this conjugation so important?
Well, imagine taking the inner product of a function with itself. We'd expect that result to be real and non-negative — something that represents the square of the function's length or its energy. But without the conjugate, we might get a complex number, which doesn't make physical sense in that context.
The conjugate takes care of that. It ensures the phase — or directional twist — of the complex function is canceled out. What's left is the amplitude — the magnitude — and that's what we care about when computing lengths or measuring how much two functions align.
Another way to see this is: the conjugate helps us ignore the phase and focus purely on the size or strength of the signal.
So this isn't just a technical tweak — it's what makes the geometry of complex spaces work. Thanks to the conjugate, we preserve all the essential properties:#orthogonality, projection, the norm — and more.
So when we say two complex-valued functions are orthogonal, we mean that their inner product — including the conjugate — is zero. That's the complex equivalent of vectors being at right angles.
This is why conjugates appear everywhere in Fourier analysis and other areas involving complex functions. They ensure consistency and meaning across the entire framework.
So, to summarize:#In complex function spaces, an inner product is defined using integration with conjugation.#That's the key idea — and it makes everything else fall neatly into place.
Up next, we'll use this inner product to compute the complex Fourier coefficients.
Let's move forward.

slide39:
All right, now that we've got a solid understanding of complex numbers, let's talk about one of the most beautiful and powerful results in mathematics—Euler's Formula.

Let’s start with the exponential function. You might remember that the exponential of a number can be written as an infinite series. For any complex number z, we can write:
“e to the z equals one, plus z over one factorial, plus z squared over two factorial, plus z cubed over three factorial, and so on.”
In math terms, that’s:
e to the z equals the sum from n equals zero to infinity of z to the n over n factorial.
This series works not only for real numbers, but also for complex numbers. And that’s important.
How do we know this infinite sum still makes sense when z is complex? Well, there’s something called the ratio test, which shows that this series converges no matter what complex number we use. In other words, it always gives a valid result. That’s why we can define e to the z for any complex number z. That’s a big deal.
Now, here’s where things get really exciting.
What if we plug in a purely imaginary number, like i times theta, into this exponential?
It turns out that:
e to the i theta equals cosine theta plus i times sine theta.
That’s Euler’s formula.
This equation is incredibly elegant. It shows how exponential functions and trigonometric functions—two ideas that seem totally different—are actually deeply connected.
And here’s something even more useful. If you rearrange Euler’s formula a bit, you can write cosine and sine in terms of complex exponentials.
Specifically:
Cosine theta equals e to the i theta plus e to the minus i theta, divided by two.
Sine theta equals e to the i theta minus e to the minus i theta, divided by two i.
These two identities are incredibly helpful, especially in signal processing and Fourier analysis. They let us replace sines and cosines with exponentials, which makes the math much cleaner and often easier to compute.
So to sum up:
Euler’s formula is a bridge between the world of waves—like sine and cosine—and the world of exponentials. And as we’ll see next, that bridge is exactly what we need to rewrite the Fourier series in a compact, exponential form.
Let’s move forward and take a look at that next.

slide40:
Now that we’ve built up our understanding of Euler’s formula, we’re ready to transition from the real form of the Fourier series to the complex form.
Let me start by reminding you of the real version. For a periodic function, f of t, we can write it as:
"a zero divided by two, plus the sum over n of a n times cosine of two pi n t, plus b n times sine of two pi n t."
That’s the standard real Fourier series form. We’ve seen that it works well, but it requires us to carry around both sine and cosine terms—and two separate sets of coefficients.
Now here’s the trick: using Euler’s formula, we can rewrite sine and cosine using complex exponentials.
For example, cosine of two pi n t becomes:
"e to the power i times two pi n t, plus e to the power negative i times two pi n t, all divided by two."
And sine of two pi n t becomes:
"e to the i two pi n t, minus e to the negative i two pi n t, all divided by two i."
So what’s the point of doing this?
Well, once we’ve written everything in terms of exponentials, we can merge the sine and cosine parts into one unified expression. Everything now becomes a sum of complex exponentials—no separate sine and cosine terms needed.
Even more interesting, these exponential functions—like e to the i two pi n t—where n is any integer, actually form an orthonormal basis.

That means: each function has unit length, and all the functions are mutually orthogonal. If you take the inner product of e to the i two pi m t and e to the i two pi n t, the result is:
"one, if m equals n; and zero, if m is not equal to n."
This elegant structure is one of the biggest reasons we prefer the complex form in many applications. It makes the math cleaner, the derivations simpler, and the overall representation more compact.
So in summary: by converting sine and cosine into exponentials using Euler's identity, we arrive at a cleaner and more powerful way to express Fourier series. And this sets the stage for everything that follows in frequency analysis and Fourier transforms.
Coming up next, we'll write down the full complex form of the Fourier series.

slide41:
Let's take a moment now to fully understand what we mean by an orthonormal basis, especially when we're working with complex exponentials.
We define a family of functions—denoted e n of t—where each function is written as:# "e to the power two pi i n t."#Here, n is any integer, and t is in the interval from zero to one.
Now, to confirm that these functions actually form an orthonormal basis, we'll compute the inner product between two of them. Let's say we have e n t and e m t, and n is not equal to m.
Since we're in a complex function space, remember that we have to conjugate the second function when we compute the inner product.#So, the inner product between e n and e m is:# "The integral from zero to one of e to the two pi i n t, times the complex conjugate of e to the two pi i m t."
Now, taking the conjugate of an exponential just flips the sign in the exponent. So this product becomes:# "e to the power two pi i times the quantity n minus m, all times t."#We're now integrating this from zero to one.#What happens?
Well, if n is not equal to m, this exponential is rotating in the complex plane over the entire interval, and the total contribution over one period averages out to zero.#So, in that case, the inner product is zero. This confirms that the functions are orthogonal when n and m are different.
Now, what if n equals m?#Then the exponent becomes zero, so we're simply integrating the constant function one, from zero to one. That gives us a value of one.#So this confirms that the functions are normalized when n equals m.#Putting both of these observations together, we have the classic orthonormality condition:# "The inner product of e n and e m is equal to one if n equals m, and zero otherwise."#That's exactly what we expect from an orthonormal basis.
These complex exponential functions form the mathematical backbone of the Fourier series. Each function is like a "building block" that carries a particular frequency, and together they allow us to represent almost any function on the interval from zero to one.#This structure is elegant, efficient, and deeply powerful.#And in the next step, we'll use these building blocks to express the complex form of the Fourier series.

slide42:
Now that we've built the full foundation, we're finally ready to write down the complex form of the Fourier series.#Instead of expressing our function f of t using separate sine and cosine terms, we now use complex exponentials as the basis functions. And the result is both beautiful and powerful.#We write:#"f of t equals the sum, from n equals negative infinity to positive infinity, of c n times e to the power i two pi n t."#So here, each term in the sum is a complex sinusoid at frequency n. And the corresponding coefficient, c n, tells us how much of that frequency is present in the signal.
Now, how do we find each coefficient?#Because our basis functions are orthonormal, we can isolate any one coefficient simply by taking the inner product of f with the exponential function e to the i two pi n t. That gives us:#"c n equals the inner product of f of t and e to the i two pi n t,"#"which equals the integral from zero to one of f of t times e to the negative i two pi n t, with respect to t."#Notice here that minus sign in the exponent. That comes from taking the complex conjugate inside the inner product.#This formula is incredibly useful—it gives us a direct way to compute each coefficient without

interference from the others.
Now, if you recall the real Fourier series with cosine and sine terms, and if you rewrite cosine and sine using Euler’s formulas, and substitute them back into the real series, you’ll get the same expression—but now in terms of complex exponentials.#And the coefficients come out as:#For non-zero n,#"c n equals a n over two, minus i times b n over two."#For n equals zero,#"c zero equals a zero over two."
So the complex coefficients still capture the exact same information as a n, b n, and a naught—but now in a more compact and unified form.#And finally, if your function f of t is real-valued, then the Fourier coefficients have a special property called conjugate symmetry. That is:#"The complex conjugate of c n equals c negative n."#This symmetry ensures that when you sum the exponentials—both positive and negative frequencies—the imaginary parts cancel out, and you’re left with a real-valued function.
So to summarize, this complex form of the Fourier series gives us a cleaner expression, easier computation, and a deeper insight into the structure of signals. It’s no surprise this form is central in many areas, from signal processing to quantum mechanics.#Next, we’ll build on this to explore how these complex exponentials serve as building blocks for signal decomposition and frequency-domain analysis.

slide43:
Let’s take a step back now and try to visualize everything we've been talking about.
When we say we’re representing a function using a Fourier series, what does that mean?
Well, it means we’re placing that function inside a kind of infinite-dimensional space. Each dimension in this space corresponds to a different basis function—just like in three-dimensional space, where we have the x, y, and z axes.
On the left side of this slide, you see the real basis. These are made up of the constant function 1, the cosine functions like cosine of 2 pi m t, and the sine functions like sine of 2 pi n t. These real functions are all orthonormal, and together, they span the space of all square-integrable periodic functions.
Now notice the red arrow labeled f of t. That red arrow represents a function. And just like any vector in 3D space, we can express this function as a sum of its components in each direction. But in our case, the directions are sine, cosine, and the constant function.
To figure out how much of each component is present, we project the function onto each basis function. That’s exactly what we’re doing when we compute the Fourier coefficients. We're just asking: how much of cosine is in this signal? How much of sine? And so on.
Now look at the right-hand diagram. It shows the same idea, but using a complex basis instead. Here, the directions are complex exponentials like e to the i 2 pi n t. These complex exponentials are also orthonormal and form a complete basis in the complex function space.
The red arrow here, labeled g of t, could represent the same function as f of t. But now, it’s decomposed using complex components instead of sines and cosines.
So whether we use real functions or complex exponentials, the concept is the same: we're decomposing a function into its “directional” components in function space—just like how you decompose a 3D vector into its x, y, and z parts.
This visualization helps us see Fourier series not just as a bunch of formulas—but as a geometric idea. A function is a vector, and we’re expressing it as a sum of basis vectors.
This is the heart of Fourier analysis.
Next, let’s take a look at some concrete examples to see how this actually plays out in practice.

slide44:
Let’s wrap up this lecture by exploring a concrete and classic example: the square wave.
What you see here is a periodic function, one that flips back and forth between two constant values—a positive one and a negative one.
Let’s walk through the structure.
From time t equals zero up to one-half, the function holds steady at plus one.

Then, from one-half to one, it suddenly drops to minus one. After that, the pattern repeats. And again. And again. This continues infinitely in both directions along the time axis.
So overall, this is a piecewise function. Within each period, it's constant in two segments, with an abrupt jump at the halfway point. And because it repeats every unit of time, we say it has a period of one.
Now, at first glance, you might think, “This function is too rough for sine and cosine to handle.” It has sharp edges, and it jumps suddenly. But that’s where the power of Fourier analysis shines.
Even though the square wave is not smooth, we can still represent it as a sum of smooth, continuous sinusoids. It’s exactly this kind of jumpy, discontinuous behavior that makes the square wave such a great test case for the Fourier series.
You’ll soon see that we can approximate the square wave by adding up enough sine components. And the more terms we include, the closer the sum will come to capturing that sharp transition from plus one to minus one.
This example will also reveal an interesting phenomenon known as the Gibbs effect, which we’ll talk about next.
But for now, just keep in mind: even for a simple-looking function like this square wave, Fourier series gives us a powerful and systematic way to build it up—using only smooth building blocks.
So now let’s take the next step and compute the Fourier coefficients for our square wave.
As we saw before, the square wave alternates between plus one and minus one over each period. From zero to one-half, it’s plus one. From one-half to one, it drops to minus one. And this pattern repeats every unit interval.
Now, we begin with the zeroth Fourier coefficient. This represents the average value of the function over one period. But notice: for every positive bump, there’s a negative one of equal size. So the total area above the axis cancels out the area below. That means the average value is zero.
So, the zeroth term disappears.
Next, we compute the other Fourier coefficients, often denoted by f-hat of n. These are obtained by taking the inner product of the function with the complex exponential e to the negative 2 pi i n t.
Since our square wave has two constant segments, we split the integral into two parts: one from 0 to one-half, where the function equals plus one, and one from one-half to 1, where the function equals minus one.
We integrate e to the negative 2 pi i n t over each of these intervals, then subtract them.
After doing the math, we arrive at this compact expression:
F hat n equals to 1 over pi i n times the quantity 1 minus e to the negative pi i n.
Now this is an elegant formula. It tells us exactly how strong each frequency component is. For each integer value of n—not equal to zero—we get a corresponding complex exponential that contributes to building up the square wave.
And so, we write the full Fourier series as an infinite sum of these terms. That is, summing over all non-zero values of n:
The sum of 1 over pi i n times 1 minus e to the negative pi i n times e to the 2 pi i n t
Each of these exponential functions corresponds to a specific frequency, and each coefficient tells us how much of that frequency is present in the signal.
On the next slide, we’ll further simplify this result and begin to see the pattern more clearly—especially for odd harmonics.

slide45:
Now let’s simplify the Fourier series we derived for the square wave.
Earlier, we saw that the Fourier coefficients involved a term like “1 minus e to the negative pi i n.” Let’s think carefully about that. What happens when n is even?
Well, if you plug in an even number, this expression becomes zero. That means: all even-numbered terms in the series disappear completely. On the other hand, when n is odd, this term evaluates to two.
So what does this tell us? It tells us that the square wave is made up only of

odd harmonics. That is, only the sine waves whose frequencies are odd multiples of the base frequency will appear in the final expression.
With this insight, we rewrite the series, summing over only the odd values of n. And since the numerator is just 2 now, the coefficient simplifies to 2 over pi i n, multiplied by e to the 2 pi i n t.
Now we go one step further. We combine the terms for plus n and minus n. When we do that and apply Euler's identity again, something beautiful happens. The sum of e to the i theta and e to the negative i theta becomes a sine function. Specifically:
e to the 2 pi i n t minus e to the minus 2 pi i n t equals 2 i times sine of 2 pi n t
So now, our series, which originally looked complex and full of exponentials, becomes a clean sine series.
We make one last substitution. Since we're only keeping odd values of n, we can write n as 2 k plus 1, where k runs from zero to infinity.
This gives us the final, elegant result:
f of t equals 4 over pi, times the sum from k equals 0 to infinity, of 1 over 2 k plus 1, times sine of 2 pi times 2 k plus 1 t
This is the Fourier series for the square wave.
Notice how only sine terms appear. That makes perfect sense, because the square wave is an odd function—it's symmetric around the origin. And in Fourier analysis, odd functions naturally expand into sine terms, just like even functions expand into cosines.
This result is not only beautiful but also extremely useful. It tells us exactly how to build a square wave by layering together the right sine waves, each scaled just right.
Let's go on and see what this looks like when we actually approximate the square wave with a few of these terms.

slide46:
Let's take a moment now to understand what happens when we approximate a function using only a finite number of Fourier terms.
In theory, the Fourier series uses infinitely many sine or complex exponential terms to exactly reconstruct the original function. But in practice, we can only sum up a limited number of them. So what does that mean?
Well, in regions where the function is smooth and continuous, like in the middle of a flat section of the square wave, the approximation is excellent. The series follows the shape of the function very closely, and the error becomes negligible as we include more terms.
But notice what happens at the discontinuities—those sudden jumps where the function goes from 1 to 1. That's where the series begins to struggle.
If you look at the plot here, you'll see a ripple near each jump. These ripples don't go away even if we add more and more terms. This phenomenon is known as the Gibbs phenomenon, and we'll explore it in detail shortly.
For now, focus on what the formula in the blue box is telling us. It describes the value that the Fourier series converges to at each point x.
If the function f of x is continuous at that point, then the Fourier series converges to f of x. That's simple enough.
But if the function is discontinuous, then the series converges to the average of the left-hand and right-hand limits. Mathematically, that means:
S of x equals one-half times f of x from the left, plus f of x from the right
So the series doesn't "miss" the function entirely at the jump—it just lands at the midpoint. This is a subtle but important point: the Fourier series always converges to something, and that something is precisely defined.
In summary, finite Fourier series work very well across most of the domain, but they introduce oscillations and errors around jumps. That's an inherent limitation of this type of approximation—and next, we'll give this behavior a name and look more closely at its mathematical structure.

slide47:
Now here's something truly fascinating—and unavoidable—when working with Fourier series. It's known as the Gibbs effect.
Let's revisit what we saw on the previous slide. When we try to approximate a function with a sudden jump, like our square wave, the Fourier series doesn't

quite nail the jump. Instead, it creates an oscillatory overshoot near the point of discontinuity. You can see these ripples in the red-highlighted regions on this plot.
What’s important to understand here is that this overshoot doesn’t disappear—even if we use hundreds or thousands of terms. Yes, the oscillations become more tightly packed, clustering closer to the jump. But the height of the overshoot—the peak error—does not go away.
It approaches a fixed value—about 9 percent of the jump in the function. That’s the defining feature of the Gibbs effect.
So why does this happen? Well, the Fourier series is built from smooth, continuous waves—like sines and cosines or complex exponentials. But when we try to represent a sharp edge—a discontinuity—using only smooth components, we hit a fundamental limit. There’s no perfect way to form a step out of waves.
This mismatch creates those persistent ripples. It’s not a bug in the method—it’s a deep mathematical truth.
So even though the Fourier approximation converges pointwise almost everywhere, and even in the sense of least squares, it never perfectly resolves a jump. This is especially important in applications like signal processing or image reconstruction, where sharp edges and sudden transitions often appear.
The takeaway is: when you see these ringing effects around edges in a Fourier approximation, you're witnessing the Gibbs effect in action.
Let’s now bring all this together with a final reflection on what we’ve learned from the square wave example.

slide48:
So far, we’ve assumed that the function we’re analyzing is periodic with a period of one. But in real applications, that’s not always the case. The period might be two, five, or even some irrational number like pi. So how do we handle that?
It turns out the Fourier series still works perfectly—we just need to adjust our formula to account for the actual period, which we’ll call capital T.
The idea is simple: we scale the time variable by dividing it by T. That way, we bring the problem back into a familiar form—similar to what we did when the period was one.
So the Fourier series becomes:
f of t equals the sum over all n from negative infinity to infinity of c n times e to the power i 2 pi n t over T.
Let me say that again more slowly:
We write the function f of t as the sum of c n times e raised to the power i times two pi times n times t divided by T.
Now what about the coefficients?
We compute c n using an integral over one full period. And we have two equivalent options for that:
First option:
c n equals one over T times the integral from zero to capital T of f of t times e to the power negative i 2 pi n t over T, integrated with respect to t.
Or, second option:
c n equals one over T times the integral from negative T over 2 to positive T over 2 of f of t times e to the power negative i 2 pi n t over T, again integrated with respect to t.
Either form works. The only difference is the range of integration—whether you go from zero to T or from minus T over 2 to plus T over 2.
So what’s the takeaway?
This is just a change of variable. We're scaling time so that the period becomes one, applying all our previous results, and then scaling back.
This gives us a general version of the Fourier series that works for any period—not just the special case where the period is one.
And this flexibility is exactly what makes the Fourier series so powerful. It adapts easily to the structure of whatever signal you're analyzing.
Up next, we’ll take this idea even further and connect it to the Fourier transform, which is what happens when the period becomes infinitely large.

slide49:
Up to this point, everything we’ve talked about has been in one dimension. But

in real-world applications—especially in imaging—we usually work with two-dimensional data. Think about a photograph, a CT slice, or an MRI frame. These are functions of two variables, not just one.
So, does the Fourier series still apply? Absolutely. In fact, the transition to two dimensions is very smooth and intuitive.
Let’s start with the formula at the top. This is the 2D Fourier transform, or more precisely, the 2D discrete Fourier series. It converts a spatial domain function into a frequency domain representation.
We begin with a function f of m and n. Here, m and n are the spatial coordinates, like pixel row and column indices.
Then we compute the capital F of x and y, which is the frequency-domain representation of the image. We get it using a double summation—from m equals 0 to M minus 1, and from n equals 0 to N minus 1.
Each term in the sum includes f of m and n, multiplied by a complex exponential:#e to the power negative j 2 pi times x times m over M plus y times n over N.
This is just the 2D extension of what we did before in 1D—now applied along both spatial directions.
Next, let’s look at the inverse transform—the formula at the bottom.
Here, we reconstruct the original image, f of m and n, from its frequency components, F of x and y. Again, we use a double summation ranging from 0 to M minus 1 and N minus 1.
And we add a normalization factor of 1 divided by M times N.
The exponential now becomes:#e to the power positive j 2 pi times x times m over M plus y times n over N.
So once again, we’re projecting onto a set of complex exponential basis functions—just like in the 1D case, but now expanded to two dimensions.
This idea even extends naturally to three dimensions, which is useful for things like volumetric imaging, 3D scans, or dynamic time-sequence data in medical imaging.
So to summarize: the 2D Fourier series gives us a powerful way to analyze images in terms of spatial frequencies. It's conceptually the same as the 1D case—just applied in two directions simultaneously.

slide50:
And this brings us to your homework assignment.
First, I’d like you to derive the formulas for the Fourier series coefficients using real-valued notation—just like we discussed today. Go through the reasoning step by step, and try to develop some familiarity with the structure of these expressions. Think of it as reinforcing what we’ve covered, not just repeating it.
Second, watch the YouTube video linked here, and use it as a guide to implement the square wave example we worked through. This will give you hands-on experience with how the Fourier series behaves—especially when approximating functions with sharp transitions.
This topic is a bit more involved than convolution, so I recommend reviewing the lecture carefully. But if you get the hang of it, you’ll find the transition to the Fourier transform much smoother in the next lecture.
Good luck—and see you next time.