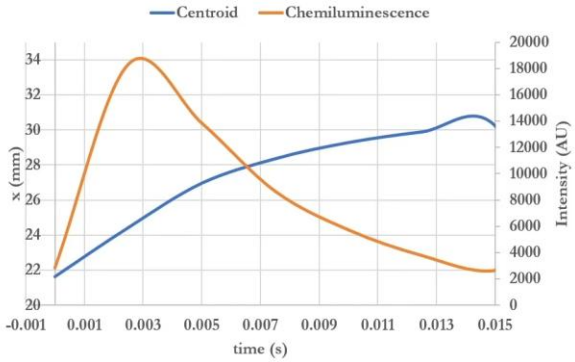
Advanced Measurement Techniques in Fluid Mechanics and Heat Transfer
Prof. Saptarshi Basu
Department of Mechanical Engineering
Indian Institute of Science, Bengaluru
Week – 04
Lecture - 19
Introduction to Experimental Signal Processing

So, in the previous lecture, we looked into how we could extract our data from image sequences regarding our parameters of interest and basically convert it into time series signals. So, we saw that from an image sequence of a flame propagating in the channel, we were able to come up with a time series variation of the chemiluminescence intensity and the position variation of the flame. So, similar time series profiles can be obtained for all the other parameters in which we are interested. Such time series signals can also be directly obtained if we use sensors; for example, if we use a pressure sensor or a thermocouple, we directly get time series variations of these parameters at a specific location in the flow field. Now, we are going to look into how we can process this time-series data. How can we represent them in an easier format so that we can make sense of them easily? So, one of the most commonly used techniques is to use Fourier series representation of time series signals.

So, this can be viewed as a coordinate transformation so that the interpretation of time series signals becomes easier.

Time Series analysis of Experimental data

- Image based data/sensor data from experiments can be represented in a time series format



- Resolving the time series along appropriate co-ordinates can greatly simplify the analysis
  - Fourier series approximation can be thought of as one such co-ordinate transformation on a function

So, before moving on to the Fourier series representation of a time-series signal, we need to walk through a few definitions. So, we will start off with the inner product of the functions. Say we have two time series signals f(t) and g(t) that are defined in the domain [a, b].

Then we can define the inner product between the two functions as an integral that runs from a to b, which is basically the size of the domain $\int$ f(t) g(t) dt. So, we can think of this as basically a vector product equivalent in function space. To understand this, let us look at an example in the vector space. Say we have two vectors f and g. What we are essentially doing here is taking the summation of the element-wise product across the entire length of the vector.

So basically, that is nothing but the dot product of the two vectors f and g. So basically, the inner product is essentially the equivalent of the dot product in the function space. So once we have this inner product defined, we can define another important parameter, which is the L_2 norm of a function. The L_2 norm of the function f (t) is defined here as the square root of the inner product with itself. And that is basically nothing but the $\left[\int_a^b f(t)f(t)\,dt\right]^{1/2}$.

So, this is actually used to normalize the function as we will see in one of our future slides. Now, to bring about the idea of Fourier series representation of our time series signal, we will again look into an example from our vector algebra. Suppose I have a function f that is defined in a vector space, and in that vector space, I have two sets of basis functions: x, y and u, v. I am not going to get into the definitions of what basis vectors are. You can basically think of them as vectors that cannot be represented as a linear combination of one another.

Basically, that means that I cannot express x in terms of y, and a linear combination of x and y can basically span the entire vector space. So, vector sets that have this property are called basis vectors. So now in this vector space, I have two sets of basis vectors: x, y and u, v. Now this function f can basically be expressed in terms of both x, y and u, v. And the way we do it is in this format.
sz
So basically $\vec{f} = (\vec{f} \cdot \hat{x})\hat{x} + (\vec{f} \cdot \hat{y})\hat{y}$ .Here, this $\hat{x}$ and $\hat{y}$ are basically nothing but unit vectors along the x and y directions. That's it. And this $(\vec{f} \cdot \hat{x})$ is basically nothing but the projection of f along the x-axis, which is essentially this length. And similarly, $(\vec{f} \cdot \hat{y})$is essentially nothing but the projection of f along the y-axis.

So that is basically nothing but this length. So, once we have evaluated the projections, we

can now express the function f as a linear combination of our basis vectors. Similarly, we can express this f in terms of the coordinates u and v as well, and we can write it down in this format: $(\vec{f} \cdot \hat{u})$ in the $(\hat{u})$ direction plus $(\vec{f} \cdot \hat{v})$ in the $(\hat{v})$ direction again. $f \cdot \hat{u}$ is basically nothing but the projection of f along the û axis, and $f \cdot \hat{v}$ is nothing but the projection of f along the $\hat{v}$ axis. So, again, once we have evaluated the projections, we have expressed the function f as a linear combination of the basis vectors.

We are going to use the same idea for our time series representation in this Fourier series analysis as well. So again, what we are going to do is find the basis functions in our space, in our domain where the function is defined. And once we have the basis functions, we are going to take the projection of our time-series function. With these basis functions, we are going to find the projections. And once we have the projections, we are going to write our time series function as a linear combination of the basis functions on which the projections were done.

Fourier Series

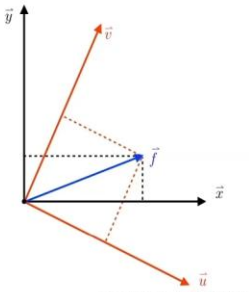
- Inner product of function $f(t)$ and $g(t)$ in the domain $[a, b]$

$$< f(t), g(t) > = \int_a^b f(t)g(t)dt\,; \qquad ||f(t)||_2 = \sqrt{< f(t), f(t) >} = \left(\int_a^b f(t)f(t)dt\right)^{1/2}$$

- The idea behind Fourier transform is to approximate a function in terms of orthogonal basis function, and are synonymous to vector decomposition in orthogonal basis vectors

$$\vec{f} = (\vec{f}.\hat{x})\hat{x} + (\vec{f}.\hat{y})\hat{y} = (\vec{f}.\hat{u})\hat{u} + (\vec{f}.\hat{v})\hat{v}$$

$\hat{x}$ *and* $\hat{y}$ are unit vectors along $\vec{x}$ and $\hat{y}$



Brunton et al, 2023

So, that is what we will be looking into next. So, similar to the vector decomposition that we did on our previous slide, we are now going to express our function f (t) that is defined in the domain $[-L,\ L]$. As a linear combination of basis functions in that domain. So, here the basis functions that we choose are of sine and cosine nature. So, this can actually be proven mathematically that they form the basis functions in that domain.

We are not going to get into that, but here is what we have essentially done: we have expressed our function f(t) as a linear combination of the cosine terms and sine terms. And there are an infinite number of cosine terms and sine terms, in fact, because the summation

runs for the values of k from 1 to ∞. And as we change the values of k, what we are essentially doing is changing the frequency of the sine and cosine waves. Now, these coefficients that we find, a_0, a_k, and b_k, are nothing but the Fourier coefficients; they are called the Fourier coefficients. And if we look into the definition of a_k, a_k is basically , $\int_{-L}^{L} f(t)\, cos\left(\frac{k\pi t}{L}\right)$.

So this is basically nothing but the dot product of f(t) with the basis function $cos\left(\frac{k\pi t}{L}\right)$. So what we are doing is trying to project the function f(t) onto the basis function $cos\left(\frac{k\pi t}{L}\right)$. So this is giving the projection. This factor that we find in the denominator is a normalization factor, and we need it because when we did the vector decomposition in the previous slide, we took the projection of f along the unit vectors; we did $\left(\vec{f} \cdot \hat{x}\right)$ and $\left(\vec{f} \cdot \hat{y}\right)$. So, to take the dot product with the normalized vector, we use this normalization factor here because the functions are not normalized, and to normalize them, we use this factor.

Similarly, bk is basically nothing but the projection of f(t) along the basis function $sin\left(\frac{k\pi t}{n}\right)$. And we can also view this a naught value basically as the projection of f(t) along the function 1. It can be viewed as a cosine wave with 0 frequency and an amplitude of 1. So, what we have finally done using this expression is express this function f (t) as a linear combination of cosine and sine waves, which basically form basis functions in my domain.

Fourier Series

- A function $f(t)$ periodic in the interval $[-L, L]$ can be written down as,

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty}\left[a_k \cos\left(\frac{k\pi t}{L}\right) + b_k \sin\left(\frac{k\pi t}{L}\right)\right]$$

$$a_0 = \frac{1}{2L}\int_{-L}^{L} f(t)dt = \langle f(t), 1\rangle \frac{1}{||1||_2}$$

$$a_k = \frac{1}{L}\int_{-L}^{L} f(t)\cos\left(\frac{k\pi t}{L}\right)dt = \left\langle f(t), \cos\left(\frac{k\pi t}{L}\right)\right\rangle \frac{1}{\left\|\cos\left(\frac{k\pi t}{L}\right)\right\|_2}$$

$$b_k = \frac{1}{L}\int_{-L}^{L} f(t)\sin\left(\frac{k\pi t}{L}\right)dt = \left\langle f(t), \sin\left(\frac{k\pi t}{L}\right)\right\rangle \frac{1}{\left\|\sin\left(\frac{k\pi t}{L}\right)\right\|_2}$$

So, in the previous slide, we basically expressed this function f (t) as a linear combination of sine and cosine waves, but we can always find other basis functions as well, other sets of basis functions within our domain.

And one such commonly used representation is by using these complex exponential terms. So $e^{\frac{i\pi kt}{L}}$, where k can take values from minus infinity to plus infinity. They also form basis functions in our domain. And we can express this function f(t) as a linear combination of these basis functions as well. And using the same analogy, C_k, which is basically the coefficient that is associated with the basis function in this summation, is nothing but the projection of the function f (t) along this basis function.

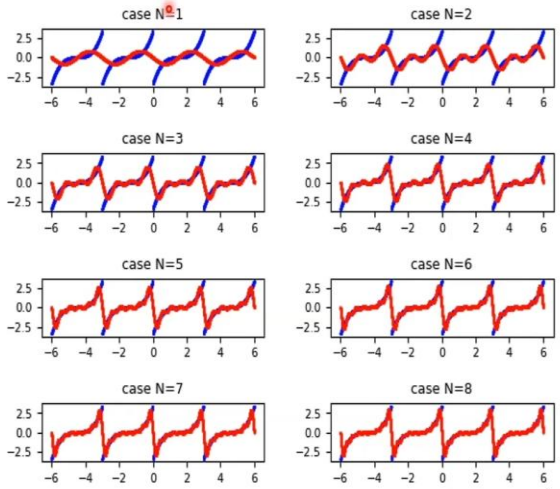
So, this is a representation that we will be using as we move further in this lecture. Now, this figure basically shows how good our Fourier series representation is. So the plot in blue here shows the actual function f (t), the time series function, which is there on the left-hand side here, and this red curve basically shows our Fourier series approximation. And here, the value of n shows the number of terms that are used in the summation. If I say I have a value of n, that basically means that I have only used one term in the summation to represent this function f (t).

I have basically truncated this infinite summation to a few terms. So we can see that when we use only one term, it's just a sine wave representation, and it is not really a good approximation. But as we increase the number of terms, even if we take only eight terms, we can see that the Fourier series approximation fairly represents our time series signal. And if we, in fact, take the infinite summation, we'll see that it is no longer an approximation, and it actually exactly represents a function. So, the Fourier series representation that we studied until now is valid for continuous functions.

Fourier Series

- Following Euler's formulation, we also express $f(t)$ as,

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{\frac{ik\pi t}{L}}; \qquad e^{\frac{ik\pi t}{L}} = \cos\left(\frac{k\pi t}{L}\right) + i\sin\left(\frac{k\pi t}{L}\right)$$



https://computationalmindset.com

However, from our experiments, we do not have continuous function. Whatever data we get from the sensors or whatever we have extracted from the images, they are data that have been sampled at discrete intervals of time. So, basically, we have discrete time series data; we do not have continuous data. Now, the question is how to perform this Fourier transformation or Fourier series representation for this discrete time series data, and that is what we will be looking into now. Suppose I have a data series; I am going to represent it in vector format.

So, say that x is basically my data vector and that x has n points; I have n data points in that signal. x_n basically shows the representation of a data point in that signal, where this value of n can take a number anywhere from 0 to n-1. So, we will use the same method as what we studied before. So, this f (t) here basically becomes a data vector x. Here, x_n is expressed as a linear combination of these basis functions, which are of the form $e^{\frac{i\pi 2kn}{N}}$.

Again, using the same analogy, x_k should correspond to the projection of this data vector x onto my basis function. So, to calculate the projection again, we take the inner product, and because it is discrete time series data that I have here, the definition of the integral basically reduces to a summation, and here the summation runs from 0 to n -1, which is basically the length of my data vector. And here again I am taking the dot product of my data vector with my basis function, and that basically gives a projection of my data vector onto the basis function. And once I have that again, once I have all the projections

estimated, these are essentially the Fourier coefficients that we studied in the continuous functions. So, once I have these Fourier coefficients or projections estimated, I can again express this f (x), f (t), or basically here my discrete time series data x_n as a linear combination of my basis functions.

So that is exactly what we have done here. Now, if we look into this definition, which estimates the Fourier coefficients of projections, we can actually see that there is a one-to-one map between the data series X_n and the projection series X. So that basically means that I can express the projection vector or my Fourier coefficient vector. as a matrix multiplication using a matrix multiplication format. So, that is what we will be doing next.

So, this is the way to express the Fourier coefficients vector. In terms of the data vector in the form of matrix multiplication. So, this directly comes from the formula that we presented to estimate these projections, which is basically based on the dot product between the data vector and the basis function. So, we can see that this is basically a matrix multiplication with this huge matrix, which is of the order of n by n, which we are calling the DFT matrix. Now, if we look into this DFT matrix, we can see that there are only two distinct terms in it.

Discrete Fourier Transform

- From experiments, we do not get continuous data, instead we get data sampled at a specific sampling rate. The data is thus discrete. (Time series data: $\bar{x}$)
- N-point signal, x_n, where $n = 0, N-1$
- The Fourier representation for a discrete signal can be written down as,

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{i2\pi kn}{N}}$$

$$where, X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-i2\pi kn}{N}} \text{ (Projection onto the basis function, } e^{\frac{i2\pi kn}{N}})$$

$$[x_0, x_1, \ldots\ldots, x_{N-1}]^T \Rightarrow [X_0, X_1, \ldots\ldots, X_{N-1}]^T$$

- A linear operator (a matrix) can be used to map from the space x to the space of X
  - We define, the fundamental frequency $\omega_N = e^{\frac{i2\pi}{N}}$

One is 1; the other is the constant one. The other is basically ω_N, which is $e^{\frac{i2\pi}{N}}$. Now, this is called the fundamental frequency. And the whole idea behind doing this exercise was that we were finding the projections of our data vector onto these basis functions so that we could get these Fourier coefficients and then express our data series as a linear combination of these basis functions. But to estimate these Fourier coefficients, we see that we need to multiply them with a matrix that is of the order of n by n.

So, the number of multiplications involved in this operation is of the order of n^2. and where n is basically nothing but the number of points in the data vector. Say, if n is equal to 1024, then we'll have to perform 1024 whole square multiplication operations to actually calculate these Fourier coefficients. And this is computationally very expensive. And that is the reason we use an algorithm, a simpler algorithm to compute the same thing, which is called the fast Fourier transform.

Discrete Fourier Transform

- A linear operator (a matrix) can be used to map from the space x to the space of X
  - We define, the fundamental frequency $\omega_N = e^{\frac{i2\pi}{N}}$

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-i2\pi kn}{N}} \qquad \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \dots & \omega_N^{N-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} \Rightarrow \bar{X} = F_{NXN}\bar{x}$$

  - F_{NXN} (or simply F_N) is called the DFT matrix

- The above computation is of the order of N^2. However, we can exploit the symmetry of the DFT matrix to reduce the computational cost
- If N = 1024, then computing $\bar{X}$ requires $(1024)^2$ multiplication operations.
- Computing DFT of large data sets is impossible even with very modern computers

---

So, we will look into that next. So, the FFT is basically a smart way to compute the Fourier coefficients or the projection vector from the DFT matrix and the data vector. So we will try to understand this. This is just an overview of the FFT method that I am giving here. We are not going into much detail, and we will try to understand this by using an example. So, suppose I have a data signal that has 16 points.

I have 16 discrete data points. So, we can now split this data signal into odd and even terms, each containing eight terms. So, basically, what we have done here is rearrange this x vector into odd terms and even terms. Once we rearrange the data vector, we also need to rearrange the DFT vector. A DFT matrix, and once we rearrange it, we find something very interesting. We see that this DFT matrix can now be expressed as a product of two other matrices, and these matrices contain terms such as I8, D8, F8.

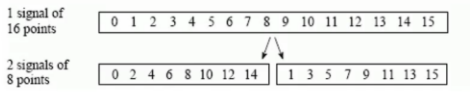
I8 is basically nothing but an identity matrix which is of the order of 8 by 8. D8 is a diagonal matrix of order 8 by 8, and I have shown the diagonal matrix here. And these diagonal matrices only contain terms such as 1 and omega n. And F_8 is a DFT matrix, but this time it is of the order of 8 by 8. So, what we have done using this technique is that we have essentially been able to express F_{16} in terms of F_8 and other diagonal matrices.

And one important thing to note is that computing multiplication with diagonal matrices requires fewer steps. We have also reduced the order of the DFT matrix. And now we can do this process iteratively.

Fast Fourier Transform (FFT)

- FFT makes use of the symmetry to the DFT matric to break it down into simpler matrices that are easier to compute
- If N = 16,

$$\bar{X} = F_{16}\bar{x}$$



We can write this down as,

$$\bar{X} = \begin{bmatrix} F_{even} \\ F_{odd} \end{bmatrix} \begin{bmatrix} \bar{x}_{even} \\ \bar{x}_{odd} \end{bmatrix} = \begin{bmatrix} I_8 & D_8 \\ I_8 & -D_8 \end{bmatrix} \begin{bmatrix} F_8 & 0 \\ 0 & F_8 \end{bmatrix} \begin{bmatrix} \bar{x}_{even} \\ \bar{x}_{odd} \end{bmatrix}$$

Here, I_8 is the identity matrix and D_8 is a diagonal matrix

$$D_8 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \omega_N & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \omega_N^7 \end{bmatrix}$$

We can further split the odd and even subsets that we created into odd and even categories in their own subsets, like the way we see here. And once we do that, we can see that we can basically express the F_8matrix, the DFT matrix of the order of 8 by 8, in terms of a DFT matrix which is of the order of 4 by 4 and F_4 matrix.

Further splitting, you can show that it can be expressed in terms of F_2, which is a DFT matrix of the order of 2 by 2. Now, what we have essentially done is that we have been able to express our main DFT matrix into A 2x2 DFT matrix and other diagonal matrices, which are of diagonal format, can be used to mathematically show that the number of multiplication operations needed to compute the projection vector using this method, while iteratively reducing the order of the DFT matrix, is of the order of n log n, which is significantly lower. Then the number of steps that are actually involved in computing the projection vector using the multiplication operation directly with the DFT matrix and the data vector, which was of the order of n squared, that we saw before. So, that is basically what the FFT does. FFT basically computes the projection vector from the DFT matrix and the data vector in a very smart and efficient way.

Fast Fourier Transform (FFT)

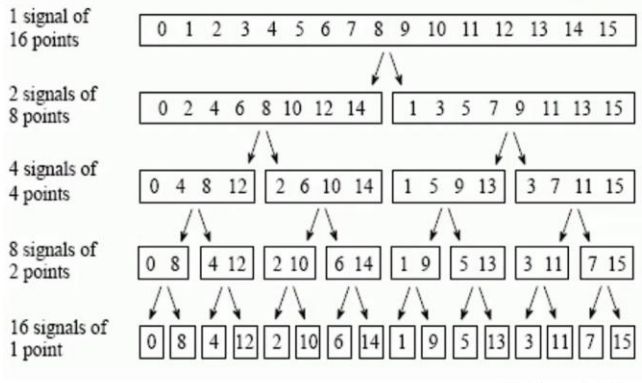
- If N = 16,

$$\bar{X} = F_{16}\bar{x}$$

$$\bar{X} = \begin{bmatrix} I_8 & D_8 \\ I_8 & -D_8 \end{bmatrix} \begin{bmatrix} F_8 & 0 \\ 0 & F_8 \end{bmatrix} \begin{bmatrix} \bar{x}_{even} \\ \bar{x}_{odd} \end{bmatrix}$$

This process can be further continued, by splitting out odd and even terms from $\bar{x}_{even}$ and $\bar{x}_{odd}$

Thus, F_8 can be expressed in terms of F_4, which can be expressed in terms of F_2



Vincke et al, 2012

- This reduces the number of multiplication operations to order of $Nlog_2N$, significantly reducing the computational cost

So now we'll look into a MATLAB tutorial where we'll use FFT to process our experimental signal to reconstruct the signal in terms of basis functions, and then we'll also try to use this FFT data to filter out the noise from our experimental signal.