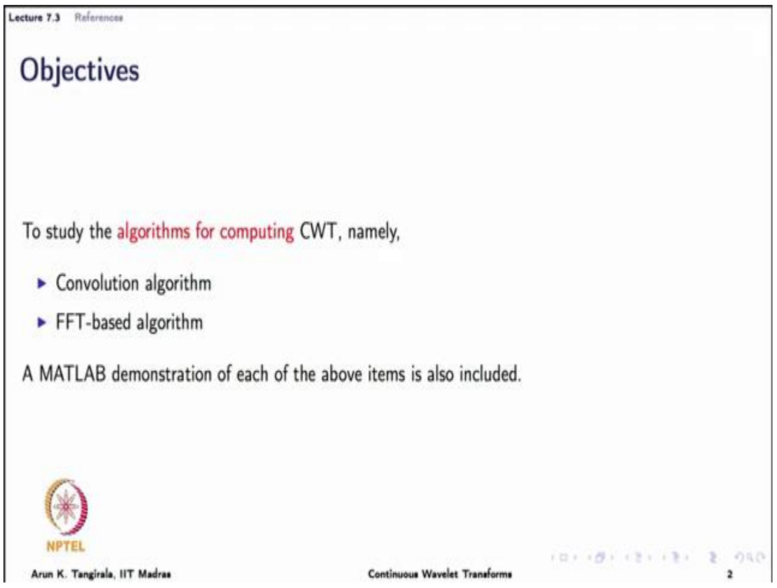
Introduction to Time-Frequency Analysis and Wavelet Transforms

Arun K. Tangirala
Department of Chemical Engineering
Indian Institute of Technology, Madras

Lecture -7.3
Computational aspects of CWT

Welcome to the lecture 7 point 3 where we shall discuss computational aspects of continuous wavelet transform. In the previous lecture, that is 7 point 2 we learnt how to convert scale to frequency and the philosophy surrounding it, as well as the procedure. The reason for discussing the conversion from scale to frequency upfront is because, in c w t as well as in scalogram those point as well as the procedures will come very handy.

(Refer Slide Time: 00:53)



Lecture 7.3 References

Objectives

To study the algorithms for computing CWT, namely,

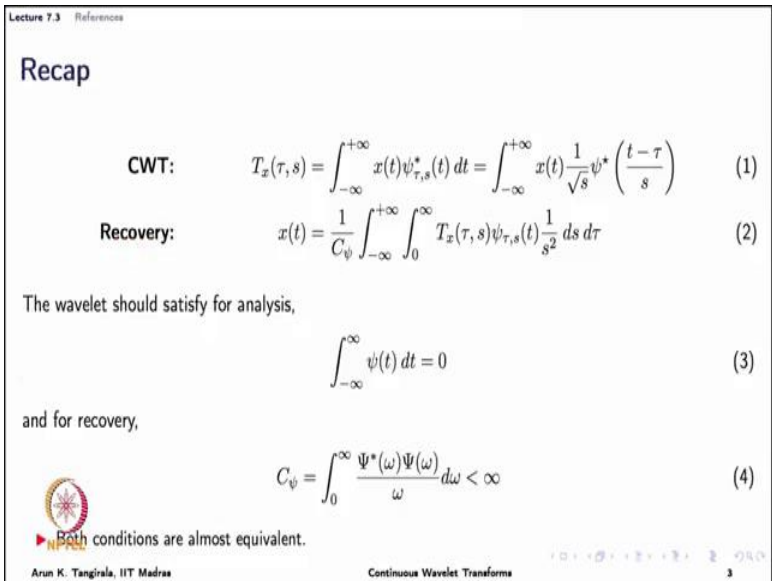
- Convolution algorithm
- FFT-based algorithm

A MATLAB demonstration of each of the above items is also included.

Arun K. Tangirala, IIT Madras Continuous Wavelet Transforms 2

In this lecture we shall specifically focus on two algorithms for computing continuous wavelet transform. One is the convolution algorithm and other is the fast Fourier transform based algorithm, these are two different algorithms, the transform remains the same. And we shall also demonstrate how to compute this c w t using these two algorithms and mat lab for you on a couple of examples.

(Refer Slide Time: 01:20)


Lecture 7.3 References

Recap

CWT: $$T_x(\tau, s) = \int_{-\infty}^{+\infty} x(t)\psi^*_{\tau,s}(t)\,dt = \int_{-\infty}^{+\infty} x(t)\frac{1}{\sqrt{s}}\psi^*\left(\frac{t-\tau}{s}\right) \quad (1)$$

Recovery: $$x(t) = \frac{1}{C_\psi}\int_{-\infty}^{+\infty}\int_0^{\infty} T_x(\tau, s)\psi_{\tau,s}(t)\frac{1}{s^2}\,ds\,d\tau \quad (2)$$

The wavelet should satisfy for analysis,

$$\int_{-\infty}^{\infty} \psi(t)\,dt = 0 \quad (3)$$

and for recovery,

$$C_\psi = \int_0^{\infty} \frac{\Psi^*(\omega)\Psi(\omega)}{\omega}d\omega < \infty \quad (4)$$

▶ Both conditions are almost equivalent.

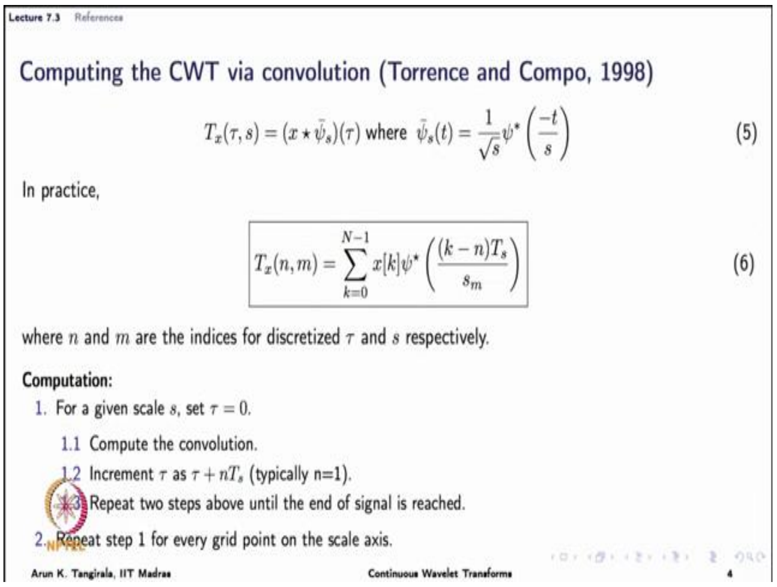
Arun K. Tangirala, IIT Madras Continuous Wavelet Transforms 3

So, just to recap the theoretical definition of continuous wavelet transform as given in equation one, here you can recall from lecture 7 point 1. It is nothing, but the transform of the signal with the wavelet at that scale and that location tau of the translation parameter tau. And also we had given the expression for synthesis or recovery of the signal and this recovery is possible only if the wavelet satisfies what is known as admissibility constant condition. That is this constant c psi should be less then infinity, it should be bounded, it should be a finite value. On the other hand, for analysis the wavelet should be of zero average and from a functional approximation view point, recall that this should get you the local details of any signal or from a signal processing view point this is a requirement for the wavelet to act as a band pass filter.

Now, both these conditions, that is the one for analysis and recovery are more or less equivalent. Because for c psi to be a finite value the magnitudes square of psi of omega although, I write this as psi star of omega time psi of omega there is a reason for writing it that way which will become clear when we talk of inverse c w t in a later lecture. So, the magnitudes square of psi of omega should decay faster than omega so, that c psi is constant. The other way of looking at it in fact, the more strict requirement is that the Fourier transform of the wave at omega equals zero should be zero. Because we have particular difficulty in evaluating this integral at omega equals zero that is where we run into the singularity point. And to avoid that difficulty that predicament we want to have the psi of omega to be zero when that is a case then we do not run into any issues.

Requiring that the Fourier transform of the wave to be zero at zero frequency is the same as saying that the wavelet should have zero average. That is by virtue of the pro property of Fourier transforms where, we know that the value of the Fourier transform of a function at zero frequency is nothing, but this integral that you see in equation three.

(Refer Slide Time: 03:56)



Lecture 7.3 References

Computing the CWT via convolution (Torrence and Compo, 1998)

$$T_x(\tau, s) = (x \star \bar{\psi}_s)(\tau) \text{ where } \bar{\psi}_s(t) = \frac{1}{\sqrt{s}}\psi^*\left(\frac{-t}{s}\right) \quad (5)$$

In practice,

$$\boxed{T_x(n, m) = \sum_{k=0}^{N-1} x[k]\psi^*\left(\frac{(k-n)T_s}{s_m}\right)} \quad (6)$$

where n and m are the indices for discretized τ and s respectively.

Computation:

1. For a given scale s, set $\tau = 0$.
   1.1 Compute the convolution.
   1.2 Increment τ as $\tau + nT_s$ (typically n=1).
   1.3 Repeat two steps above until the end of signal is reached.
2. Repeat step 1 for every grid point on the scale axis.

Arun K. Tangirala, IIT Madras Continuous Wavelet Transforms 4

Now, we will talk about the main topic in this lecture which is computing the c w t. And as I said earlier, we are only going to talk about the forward transform, the inverse transform that is, a recovery shall be discussed in a separate lecture. So, the first algorithm in the list is convolution algorithm which is based on the convolution interpretation of the continuous time wavelet transform. This was discussed in lecture in 7 point 1 if you may recall, where I could write the wavelet transform is the convolution of x with psi bar and psi bar is nothing, but a reflected version of the wavelet it is no different it is not something starkly different from the wavelet itself.

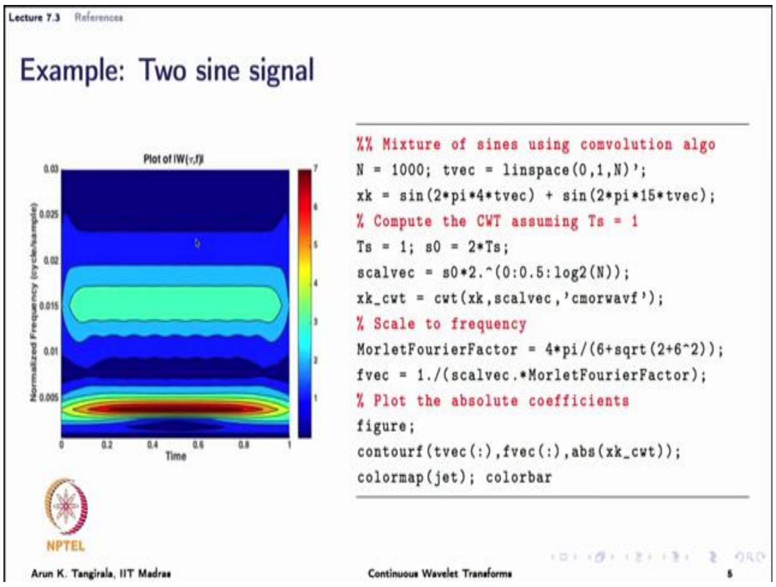
And in practice what we do is, we replace this theoretical convolution with the discrete version because I work with finite length data. Therefore, I have to implement a finite length convolution. And this convolution expression that I have given that in equation six, is not in the standard form. In the standard form you would have had psi star of n minus k times t s, but here I have k minus n t s. So, let us take some effort in understanding the new quantities that have been introduced in this expression.

The big n is of course, length of the signal and the small k keeps track of the sampling instant, and the small n is a dummy variable sorry it is not a dummy variable k is the dummy variable. This small n keeps track of the discretization in tau. We said that the theoretical convolution has to be replaced with a discrete version, but at the same time tau and s are continuous valued variables. I cannot compute c w t over a continuum, like the same arguments that we gave in arriving a d f t I cannot compute d t f t over a continuum of omega. Therefore, I discretized omega here I discretize tau and s. So, the discretization of tau leads to this grid on tau and n helps me keep track of the grid point tau. And the discretization of the scaling parameter leads to a grid on the scale axis and m helps me keep track of the point on the axis corresponding to scale. So, n and m are essentially the discretize version of tau and s and t subscript s the big t subscript s is the sampling interval, and once again s subscript m is the value of the scale at the m grid point, that is it. So, we have essentially completely discretized equation five so, as to suit the practical situation.

Now, if you were to summarize or if you have to list the steps in the computational algorithm based on this convolution expression it would be as follows. Step one what you have to do as a user specify the grid of scales over which you would like to compute the c w t that is important, because the algorithm needs to be told that. So, you pick the first on the scale grid and for that you compute for the entire set of tau points that you have discretized. So, use you start with the first grid point on s and said tau equal zero because that is the first grid point on tau, compute the convolution as given in equation six, now increment tau according to your choice. That is tau has units of arching in tau amounts to traversing the wavelet along the length of the signal and there you have freedom, in how you want to march ahead. The default that is used in many algorithms is n equals one that is you will march in the same times step as the signal itself that is tau will be incremented by t as a sampling interval. That leads to a highly over sample continuous wavelet transform a very dense one which is ok, it is good for future interaction. So, typically n set equal to one, but you could choose otherwise. In discrete wavelet transforms for example, the marching in tau is not done this way, the marching in the tau depends on the width of the wavelet at that scale.

If you have a wider wavelet you will have fewer points on the time axis to march, if you have a narrower wavelet (Refer Time 08:33) on a finer grid. So, here this is how you increment tau and then keep repeating steps 1 point 1 in 1 point 2 until you have reached the signal. And repeat this step one itself for every grid point on the scale axis that is it. So, you have to loops one for the scale one for the tau. So, therefore, you should expect this algorithm to be computationally intensive. So, let us look at the mat lab base example in computing signal that has two frequencies running in parallel.

(Refer Slide Time: 09:19)



Lecture 7.3 References

Example: Two sine signal

```
%% Mixture of sines using comvolution algo
N = 1000; tvec = linspace(0,1,N)';
xk = sin(2*pi*4*tvec) + sin(2*pi*15*tvec);
% Compute the CWT assuming Ts = 1
Ts = 1; s0 = 2*Ts;
scalvec = s0*2.^(0:0.5:log2(N));
xk_cwt = cwt(xk,scalvec,'cmorwavf');
% Scale to frequency
MorletFourierFactor = 4*pi/(6+sqrt(2+6^2));
fvec = 1./(scalvec.*MorletFourierFactor);
% Plot the absolute coefficients
figure;
contourf(tvec(:),fvec(:),abs(xk_cwt));
colormap(jet); colorbar
```

NPTEL

Arun K. Tangirala, IIT Madras Continuous Wavelet Transforms 5

So, it is as if two sine waves or super imposing on each other. What I show you here is a magnitude of the c w t the wavelet or the mother wave that, I am using to compute this transform is c mor wave f and the routine that computes the continuous wavelet transform using the convolution algorithm is c w t. So, the first two lines in the scored essentially generate the sine wave now this sine wave has two frequencies 4 hertz and 15 hertz running in parallel. And the sampling interval can be computed here.

Now, what I do here is I show you two different ways of doing things, in the first case you pretend that you do not know the sampling intervals so you set it to one. If I am given data and if I do not have an idea of what the sampling interval is, then I will set sampling interval to 1 that is what I am doing here. Now, I will draw your attention to this command here that I have written s naught equals 2 times t s. What is this s naught? This s naught is the first, the minimum scale at which you want to compute the c w t. Now, scale has an inverse relation with frequency. So, the minimum scale will correspond to the highest frequency and from sampling theorem I know that the highest frequency that I can detect, that is a highest continuous time frequency that I can detect in a sample signal is f s by 2 where f is a sampling frequency right. f s is 1 over t s; that means, highest frequency that I can detect is 1 over 2 t s. Therefore, the minimum scale that I can start with is two times t s right and that is the reason that I have 2 t s here.

Then, the grid size is your choice what I do here is I choose a dyadic grid size, but you could choose a linear discretization. This dyadic is keeping in mind that we will work with the d w t in a few lectures from now. So, I choose here a certain grid size and I go up to the length of the signal, based on the length of signal I choose my scale maximum scale. Remember the maximum scale corresponds to the lowest frequency. Ideally, I should not be even trying out log two of n I should be trying even lower scales because you may not be able to detect frequency is as low as corresponding to this maximum scale, it is ok for illustration.

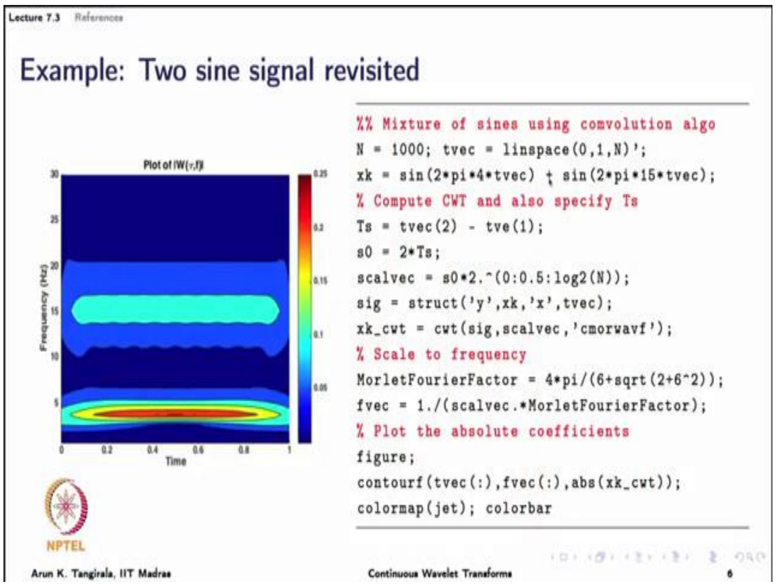
When I compute the c w t I pass the signal that I have, discrete time signal I also pass this scales over which I want to compute the c w t. I do not have to specify the translation grid because as I said the default is given in c w t, it is assumed that you want to discretize tau the same way as you have discretize time for sampling x. And the wave the particular wave that I want to use is complex Morlet wave. Recall in 7 point 2 we have talked about the suitability of using complex modulate wave for c w t. Then I convert scale to frequency again, for this recall what we had talked in lecture 7 point 2 where, we gave a table of theoretical conversion factors for scale to frequency and I am using that for the complex Morlet wave. And essentially substituting that here and then converting this scale vector to the f x the omega naught by the default is set to 6. Strictly speaking the c mor wave f actually uses naught omega naught equals 6 it uses 2 pi. So, this strict conversion factor should have been 6 naught 6, but 6 point 28. So, you can try that out in your own leisure time. And then what I do is I plot a contour, I draw contour plot of the magnitude of c w t with time on the x axis and frequency on the y axis. The magnitude plot clearly brings out the presence of two frequencies.

Now, observe that one thing that you should expect, the wavelets will give you more smearing of the energy at higher frequencies because they are highly localized in time, and by virtue of the duration band width principle you should expect more smearing of the energy in high frequency. So, this you can recall from the previous lecture that is 7 point 1 that we have talked about. Therefore, as you proceed to higher frequencies you will see more smearing of the energy which is what the situation is whereas, at the low frequencies the smearing is low.

Now, you see that we have generated a sine wave of equal power; however, of course, what you see in the plot is not power here, still the magnitude wavelet transform should look, you would expect them to look the same, in terms of a in intensity as measured by magnitude for this two frequencies, but that doesn’t seem to be the case here. But of course, now just what you say an optifact, what is happening here is the energy at higher frequencies is smeared over wider band. So, if you have to collect all the energy this is not a energy plot, but imagine that I was plotting magnitude square rather than magnitude which is what is done in scalogram. Then you would be collecting all the energy in this entire band of frequencies that, would give you the total power that is averaging across all the scales. Integrating across all the scales will give you the power present in the frequency because is in this is a still pseudo frequency likewise here. Then, you would we able to come to the conclusion that both sine waves are present with equal intensity, again this is not an intensity plot. And the coloring scheme can also be changed here, I have chosen a global coloring scheme.

Therefore, you see this kind of a plot and because of this smearing you see that the lower frequencies are colored in red and, but that is because you have also wide narrow or smearing whereas, for the higher frequencies you have a sine color. You could also choose to plot in with coloring scheme that is scale dependent, right in which case you will not be able to compare across scales because at each scale you are choosing its own color band. So, you are normalizing at each scale. So, that option is also available in fact, I invite you to really explore other options of c w t, particularly the plotting options which I do not show here this is the basic thing that you require. On the other hand I would always recommend that you plot the magnitude of c w t by yourself, to be really aware of what is happening rather than just using some nice fancy features of any routine.

(Refer Slide Time: 16:39)



Lecture 7.3 References

Example: Two sine signal revisited

```
%% Mixture of sines using comvolution algo
N = 1000; tvec = linspace(0,1,N)';
xk = sin(2*pi*4*tvec) + sin(2*pi*15*tvec);
% Compute CWT and also specify Ts
Ts = tvec(2) - tve(1);
s0 = 2*Ts;
scalvec = s0*2.^(0:0.5:log2(N));
sig = struct('y',xk,'x',tvec);
xk_cwt = cwt(sig,scalvec,'cmorwavf');
% Scale to frequency
MorletFourierFactor = 4*pi/(6+sqrt(2+6^2));
fvec = 1./(scalvec.*MorletFourierFactor);
% Plot the absolute coefficients
figure;
contourf(tvec(:),fvec(:),abs(xk_cwt));
colormap(jet); colorbar
```

NPTEL

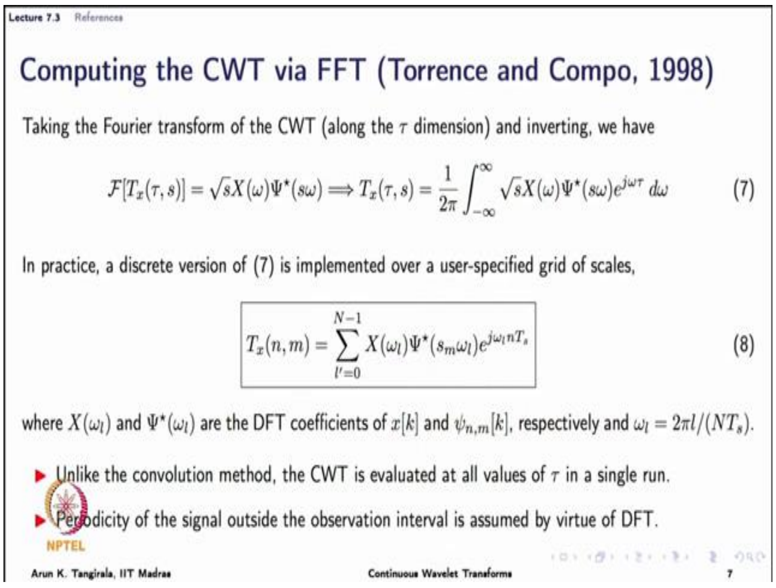
Arun K. Tangirala, IIT Madras Continuous Wavelet Transforms 6

Now, I said I wanted to show you two different ways of computing the c, not computing the c w t, but two different ways of looking at the same signal. The only difference between the previous version and this version here is, I specify the sampling interval earlier I said I do not know the sampling interval I set it to 1 now I know the sampling interval. And what I do here is the same story the minimum scale is set to 2 times t s, but the syntax that I follow with the usage of c w t changes slightly. Earlier I pass the signal now I am passing the signal as a structure, what are the two fields of the structure, the signal and the time vector over which the signal has been obtained? So, t vec is a time vector over which this been obtained and x k is a signal and this is unfortunately this y and x or only because of the notation that is used in the routine. And I should also tell you that the documentation on the c w t in this release corresponding to 2014 b does not really tell you how to pass the sampling points as well as a signal to c w t.

This you will have to figure out by opening of the c w t code and you will see that there is a option of passing the signal as well as a time points. So, I did that and now I am passing the structure what c w t does in a implicitly are within the code is it computes the sampling interval based on the time vector that you have given. And it is very important to do this because the moment you set your sampling interval to this, and you set your minimum scale according to the t s have to pass on the time instants are the sampling interval indirectly to c w t. Otherwise what happens is the scale that c w t uses is tied to the sampling interval because if you have done this here right. Here you have explicitly put in a effort in setting the minimum frequency to 2 t s, but the c w t routine also needs to know what it should do with the reaming scales and therefore, it is important that you be consistent throughout.

Once again, I use the complex Morlet wave and the rest of the story is the same. You should expect of course, a two plots should be identical the only difference being the y axis in the y axis of the I had normalize frequency which is the frequency of the discrete time signal whereas, here I have the frequency corresponding to the continuous time signal. And now it is easier to read I know that the frequencies are 4 and 15 and the magnitudes reach a peak at 4 and 15. One point that you should observe whether, this plot or the previous, if you look at the magnitude of c w t they are very small at the ends that is at the borders. Indicating or probably compelling you to believe that the signal really did not exist in this particular frequency for example, did not exist in this initial portion. But that is not the case we know that the signal exist right from the beginning that is this frequency component exist right from the origin and continues through the end. So, why is this happening because of the border effect associated with convolution operation ok.

(Refer Slide Time: 20:14)



Lecture 7.3 References

Computing the CWT via FFT (Torrence and Compo, 1998)

Taking the Fourier transform of the CWT (along the τ dimension) and inverting, we have

$$\mathcal{F}[T_x(\tau,s)] = \sqrt{s}X(\omega)\Psi^\star(s\omega) \Longrightarrow T_x(\tau,s) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\sqrt{s}X(\omega)\Psi^\star(s\omega)e^{j\omega\tau}\,d\omega \qquad (7)$$

In practice, a discrete version of (7) is implemented over a user-specified grid of scales,

$$\boxed{T_x(n,m) = \sum_{l'=0}^{N-1} X(\omega_l)\Psi^\star(s_m\omega_l)e^{j\omega_l nT_s}} \qquad (8)$$

where $X(\omega_l)$ and $\Psi^\star(\omega_l)$ are the DFT coefficients of $x[k]$ and $\psi_{n,m}[k]$, respectively and $\omega_l = 2\pi l/(NT_s)$.

- Unlike the convolution method, the CWT is evaluated at all values of τ in a single run.
- Periodicity of the signal outside the observation interval is assumed by virtue of DFT.

NPTEL

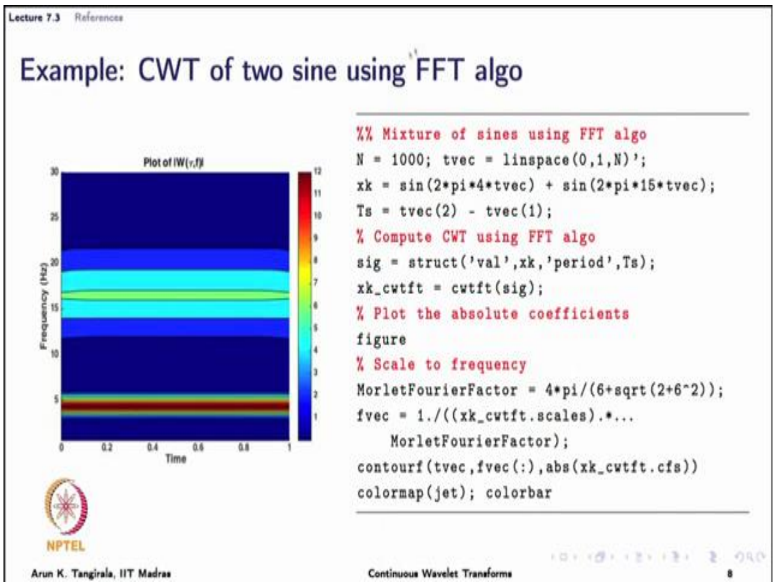
Arun K. Tangirala, IIT Madras Continuous Wavelet Transforms 7

Now, let us look at the second part of the lecture which is the computation of c w t via f f t the idea is fairly straight forward. In the convolution algorithm we took the convolution view point of the wavelet transform. Now, what we are going to do is we are going to take a Fourier transform of the wavelet transform now that may sound intimidating to you are confusing to you. But we have done this before we have taken Fourier transform of Wigner-Ville distributions as well. So, here I am taking the Fourier transform of the wavelet transform with respect to tau, the wavelet transform is a two dimensional function. I am only evaluating a one dimensional Fourier transform and the dimension that I am looking at is tau. The basic idea is to avoid this second inner loop calculation in c w t where I compute this convolution at each value of tau. With this algorithm I compute the wavelet transform at a single scale in one shot for all values of tau. How do I do that? The Fourier transform helps may do that, the Fourier transform one dimensional Fourier transform of the c w t with respect to tau is given here root s x of omega time psi star of s omega. But again is by virtue of the Fourier transform property. And all I do is take an inverse Fourier transform of this to recover again inverse Fourier transform with respect to tau to recover the continuous wavelet transform.

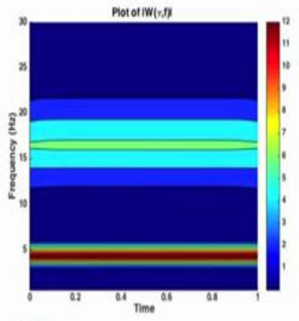
So, computationally what I do is, I take the Fourier transform of the signal and the wavelet, evaluate the Fourier transform wavelet at s omega, obtain the product and simply take the inverse Fourier transform. And of course, in practice I have to implement discrete equivalent of equation seven and once again we have n and m taking the role of the discretize points on tau and s. And now x of omega is replaced with x of omega l and psi star of s omega is replaced with psi star of s m omega subscript l. Of course, there is a root s factor which does not appear in 8. In fact, strictly speaking that should appear here as a factor, but you could factor that into your psi if you like it. So, what is this omega l? Well, l is the grid point in frequency now, like what you are doing here is, you are replacing x of omega with its d f t and psi star of s omega with its d f t. And d f t works with discrete frequencies this omega l is actually the lth point on the frequency discretize frequency axis. And of course, you have e to the j omega l m t s because taking the role of e to the j omega tau that is what is happening here.

Now, a couple of points here, one is that this algorithm is really beneficial in terms of computation, only if you have an analytical expression for psi of omega, because if that is not the case like in the Daubechies wave and so on. Then, you have to go through this additional step of computing the d f t of psi. So, if you have a close form expression all you have to do is evaluate psi at that frequency s m times omega l and that saves the computation d f t computation for psi. x of omega l of course, you have to compute the d f t there is no close form expression for the signal x k that is one point. The second point is that, because that this algorithm assumes its signal to be periodic outside the observation interval why, because I am using d f t and d f t assumes the underline signal which have not observed the infinitely long signal that I have not observe to be periodic with the same period as the observation interval. Therefore, whenever you are using this algorithm for computing c w t you have to watch out for border effects as well where its assuming that the signal repeats itself. So, let me show you couple of examples.

(Refer Slide Time: 24:35)


Lecture 7.3 References

Example: CWT of two sine using FFT algo



```
%% Mixture of sines using FFT algo
N = 1000; tvec = linspace(0,1,N)';
xk = sin(2*pi*4*tvec) + sin(2*pi*15*tvec);
Ts = tvec(2) - tvec(1);
% Compute CWT using FFT algo
sig = struct('val',xk,'period',Ts);
xk_cwtft = cwtft(sig);
% Plot the absolute coefficients
figure
% Scale to frequency
MorletFourierFactor = 4*pi/(6+sqrt(2+6^2));
fvec = 1./((xk_cwtft.scales).*...
    MorletFourierFactor);
contourf(tvec,fvec(:),abs(xk_cwtft.cfs))
colormap(jet); colorbar
```

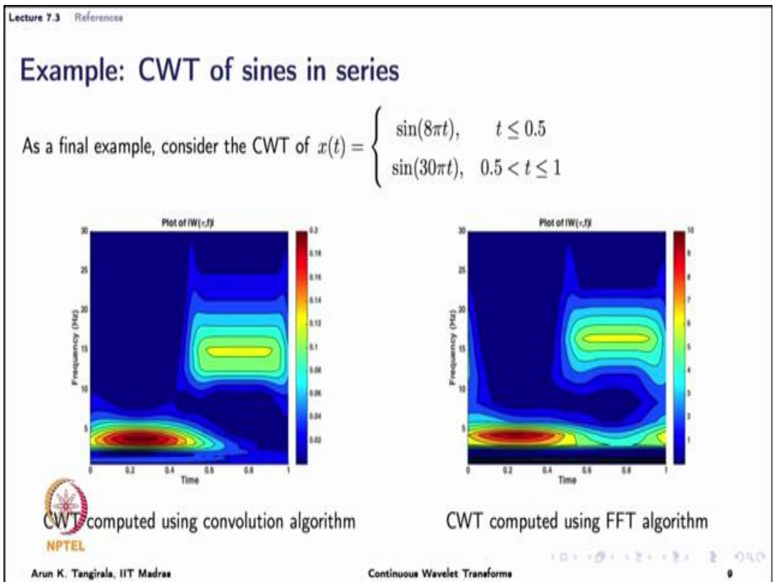
NPTEL

Arun K. Tangirala, IIT Madras Continuous Wavelet Transforms 8

Again we take the same two sine signal, but now I implement f f t algorithm. The routine that implements f f t algorithm is c w t f t is one word. And what you do here is, as a part of the syntax you generate a structure variable which takes the signal, which contains the signal as well as the sampling interval. Earlier, we gave the time vector, but now we are given the sampling interval. You could of course, pass this signal itself directly to c w t f t in which case it will assume that is the sampling interval is 1 so, that is the only difference that is it. So, once I compute the Fourier sorry wavelet transform using the Fourier transform algorithm then I draw contour plot. Now, you can see unlike in… there are of course a lot of similarity with in, what we have seen with the convolution algorithm.

But one difference that you can see here is if you compare the… we discussed this point, this the magnitude plot seems to suggest that the signal is not present in the initial and the final portions whereas, the Fourier transform algorithm has not, is not showing any such artifacts because it assumes periodicity and so, on. So, you do not run into this border effects really that you see in convolution algorithm right.

(Refer Slide Time: 26:04)



Example: CWT of sines in series

As a final example, consider the CWT of $x(t) = \begin{cases} \sin(8\pi t), & t \leq 0.5 \\ \sin(30\pi t), & 0.5 < t \leq 1 \end{cases}$

CWT computed using convolution algorithm

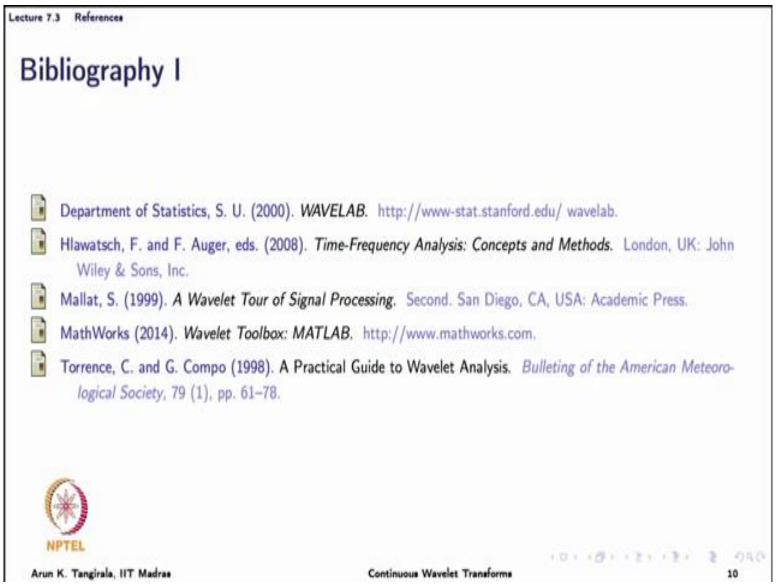
CWT computed using FFT algorithm

However let us take another example where I have sines in series this is one of the classic examples that we have been using all along in this course. So, I have a lower frequency followed by a higher frequency occupying equal durations in the length of the signal on the left plot. I show the contour plot of the c magnitude of the c w t using the convolution algorithm and on the right hand side I have the same contour plot of the magnitude of c w t using f f t algorithm. Besides this similarities you should notice an important difference which is that, in the second plot here that is a one that is computed by f f t algorithm the lower frequency appears towards the end whereas, it strictly speaking we know that the lower frequency vanishes half way through the time. Whereas, in the convolution algorithm I do not see the lower frequency resurfacing towards the end of the signal and that is because of the different assumptions underneath this algorithms with respect to the signal outside the interval right. f f t algorithm assumes that the signal is periodic and because in the wavelet transform the lower frequencies have much less smearing. You will see this coming in, predominantly showing a predominantly here and whereas, the convolution algorithm.

Here I do not assume the signal to be periodic outside the interval, but it of course, assumes something else it assumes that that the signal is for example, zero outside the interval and so on. Then you have no resurfacing of the lower frequency. So, in that respect the convolution algorithm is better than the Fourier transform based algorithm, but it is computationally more intensive. So, it is always good to generate c w t s using both this algorithms and then figuring out what is a best algorithm for a given application. So, in this lecture to summaries what we have learnt is the two different algorithms for computing c w t and both these are coded in the mat labs wavelet tool box. You have c w t and you have c w t f t they run each of this algorithms with the necessary assumptions.

We have gone through the syntax is for the respective routines and also touch on the assumptions underneath this algorithms. So, there is no particular algorithm that one can recommend of course, from a computational view point the Fourier transform algorithm is much faster, but then it assumes signal to be periodic. The convolution algorithm is computationally more intensive, but it does not make any assumptions on periodicity outside the interval. What we shall do in the next lecture is we shall study scalograms which are computed from the c w t.

(Refer Slide Time: 29:27)



Lecture 7.3 References

Bibliography I

- Department of Statistics, S. U. (2000). *WAVELAB.* http://www-stat.stanford.edu/ wavelab.
- Hlawatsch, F. and F. Auger, eds. (2008). *Time-Frequency Analysis: Concepts and Methods.* London, UK: John Wiley & Sons, Inc.
- Mallat, S. (1999). *A Wavelet Tour of Signal Processing.* Second. San Diego, CA, USA: Academic Press.
- MathWorks (2014). *Wavelet Toolbox: MATLAB.* http://www.mathworks.com.
- Torrence, C. and G. Compo (1998). A Practical Guide to Wavelet Analysis. *Bulleting of the American Meteorological Society*, 79 (1), pp. 61–78.

NPTEL

Arun K. Tangirala, IIT Madras Continuous Wavelet Transforms 10

So, therefore, this is an important precursor to the computation and discussion of Scalogram. Here a few references I would suggest that you go through the mat labs documentation on each of this algorithms and also work out examples, variance of the examples that we have presented in this lecture also the example given in the documentation and comeback to us with any questions that you may have.

Thank you and see you in the next lecture.