Continue

How to draw magnitude and phase spectrum of fourier transform

For rectangular pulse, without displacement, its phase spectrum quite simple to obtain, because Fourier Transform $F(\omega)$ is a real function. Actual functions can only stay on the actual axis, thus having $0 or $\pm \pi$ phases, depending on the positive or negative sign of the function, respectively. Try to visualize the complex plan to accomplish this. Now, if you're new to DSP, $g (t)$ might be difficult. First, consider changing the scale when dividing it by 2 in $g (t)$. This extends the spectrum by a proportion of 2, i.e. $f (t/ 2) $ transforms as: $2~F (2 \omega)$. Therefore, the $sinc$ function in $g(t)$ is twice as wide as in $f(t)$. Then the amplitude is multiplied by 3, which is why $G(\omega) = $6 for $\omega = $0. Due to the time change, you will have to multiply $F (\omega)$ with an exponential complex. Remember Time-Change Fourier transforms the theorem: $f (t - t_0)$ transforms as $F(\omega)~e^{-j\omega t_0}$. The amplitude spectrum is not changed for a time change (because $|e^{-j\omega t_0}| = $1), but the phase spectrum is added to $-\omega t_0$, which is the phase of the complex exponential (i.e. $\angle e^{-j\omega t_0} = -\omega t_0$). Finally, you would only need to add to the phase spectrum $f (t) $ linear function of $ \omega$: $ \angle H (\omega) = -4 \omega$ (since $t_0 = $4 in this case), which has a negative slope. That's why you have in Fig. (d) the sum of a negative slope linear function and a rectangular pulse train of $\pm \pi$. This is basic DSP, I would recommend taking a look at these theorems from Wikipedia for a very first look: I'll show you the easy way, then the hard way. Simple mode Since The Fourier transformation is the Laplace transformation when the real s is zero, then you can use BodePlot ft = FourierTransform[Exp[-a t] UnitStep[t], t, w, FourierParameters -> {1, -1}]; ft = ft /. (I*w) -> s BodePlot[TransferFunctionModel[(ft /. a -> 1), s]] You can improve the plot as this ft = FourierTransform[-a t] UnitStep[t], t, w, FourierParameters -> {1, -1}]; ft = ft /. (I*w) -> s tf = TransferFunctionModel[(ft /. a -> 1), s]; BodePlot[tf, GridLines -> Automatic, ImageSize -> 400, FrameLabel -> {{magnitude (db), None}, {None, Bode plot}}, {{phase(deg), None}, {Frequency (rad/sec), None}}, ScalingFunctions -> {{Log10, dB}, {Log10, Degree}}, BaseStyle -> 14] Hard way ft = FourierTransform[-a t] UnitStep[t], t, w, FourParametermeters -> {1, -1}]; LogLinearPlot[20*Log10[Abs[(ft /. a -> 1)]], {w, 0, 10}] LogLinearPlot[Arg[(ft /. a -> 1)]*180/Pi, {w, 0, 10}] Additional credit Here is a handled[Module[{ft, t, s, w, tf, a0}, ft = t] UnitStep[t], t, w, FourierParameters -> {1, -1}]; ft = ft /. (I*w) -> s; tf = TransferFunctionModel[(ft /. a0 -> a), s]; BodePlot[tf, Linii de grilă -> Automat, ImageSize -> 400, 400, -> {{{magnitude (db), None}, {None, Plot Bode}}, {{phase(deg), None}, {Frequency (rad/sec), None}}, ScalingFunctions -> {{Log10, dB}, {Log10, Degree}}, BaseStyle -> 14]], {{a, 1, a}, .1, 10, .1, Appearance -> Tagged}, Continuous Action -> False, TrackedSymbols :> {a}] Important frequency characteristics of an x(t) signal with Fourier X(w) transformation are displayed through magnitude spectrum plots, | X(w)| versus w, and phase spectrum, <X(w) versus= w.= this= demonstration= use= the= one-sided,= real,= decaying= (b=>0) exponential signal x(t) = ae-btu(t) which has the fourier X(w) transformation = a/(b + jw) The magnitude spectrum of the signal is | X(w)| = |a|/(b2 + w2) 1/2 Phase spectrum is (in radians) <X(w) == tan-1(w/b),= if= a=>0 <X(w) == p= += tan-1(w/b),= if= a=></X(w)> < 0= a= downloadable= zip-file= for= execution= on= matlab= is= available= here.= this= allow= interactive= exploration= of= properties= of= spectra= using= various= signals=.= if= you= are= familiar= with= the= transformations= below,= go= directly= to= the= zip-file.= plots= of= the= exponential= signal= x(t)=e-tu(t) (a=1, b=1) and= the= matching= spectra= are= shown= below:= you= can= select= transformations= on= this= signal,= and= view= the= effects= on= the= spectra:= amplitude= scale= y(t)=ax(t) == ae-tu(t)= time= shift= y(t)=x(t -= to)=e-(t -= to)= u(t= -= to)= time= scale= y(t)=x(bt) == e-btu(bt)=e-btu(t) time= truncation=y(t)=x(t) == e-tu(t)= for= |t|=> < t ,= and= y(t)=0 for= |t|=> T You can also select transformations on spectra, i.e. on the Fourier transformation, and view the effects on the signal: Frequency Scale Y(w) = X(cw) Ideal Filter Y(w) = X(w) for |w| < w,= and= y(w)=0 for= |w|=> W return to the demonstration page In the previous post, Interpretation of frequency compartments, frequency axis arrangement (fftshift / ifftshift) for Complex DFT were discussed. In this post, I'm going to show you to get magnitude and phase information from the FFT results. Outline For discussion here, allows you to take an arbitrary cosinus function of the form and proceed step by step, after follows ● Represents the signal in the computer (discrete-time) and graphically the signal (time range)- Represents the signal in the frequency range using FFT ()- Extract the amplitude and phase information from the FFT result, Rebuild the time domain signal from the frequency domain samples This article is part of the digital modulations book using Matlab : Build simulation models from scratch, ISBN: 978-1521493885 available in ebook (PDF) format (click here) and paperback (hardcopy) format (click here) Wireless communication systems in Matlab, ISBN: 978-1720114352 available in ebook (PDF) format (click here) and Paperback format (click here). Representation of the discrete-time domain Consider an amplitude, frequency, and phase (or) co-inus co-response signal To represent the continuous time signal in the computer's memory, we must sample the signal at</X(w)> </X(w)></X(w)> high rate (according to the Nyquist sampling theorem). We chose an oversampling factor, so the sampling frequency will be , and that provides samples within a second duration of the waveform recording. A = 0,5; %amplitude of cosinus wave fc=10;%frequency of cosinus wave phase=30; %desired phase displacement of cosinus in degrees fs=32*fc;%sampling frequency with oversampling factor 32 t=0:1/fs:2-1/fs;%2 seconds phi duration = phase*pi/180; %converts phase movement to degrees in radians x=A*cos(2*pi*fc*t+phi);%time range signal with phase change figure; parcel(t,x); %draws the Signal To represent the signal in the frequency range using the FFT function. The FFT function calculates the DFT complex. The duration of the transformation should cover the signal of interest otherwise we will lose some valuable information in the conversion process to the frequency range. However, we can choose a reasonable length if we know about the nature of the signal. For example, the cosinus signal of our interest is periodic in nature and is of length samples (for 2 seconds the duration of the signal). We can simply use a smaller number to calculate FFT. In this case, only the first domain samples will be considered to take FFT. There is no need to worry about the loss of information in this case, as the 256 samples will have a sufficient number of cycles using which we can calculate the frequency information. N=256; %FFT size X = 1/N*fftshift(fft(x,N));%N-advanced point DFT In the above code, it is used only to obtain a nice two-sided spectrum that delimits negative frequencies and positive frequencies in order. This transformation is not necessary. A scaling factor was used to explain the difference between the FFT deployment in Matlab and the definition of the text of the DFT complex. 3a. Extracting the amplitude of frequency components (amplitude spectrum) The FFT function calculates the DFT complex and therefore the results in a sequence of complex form numbers. The amplitude spectrum is obtained For obtaining a two-sided parcel, the ordered frequency axis (fftshift result) is calculated on the basis of the sampling frequency and the amplitude spectrum is graphically represented. df=fs/N; %frequency resolution exampleIndex = -N/2:N/2-1; %ordered index for FFT parcel f=sampleIndex*df; %x-axis index converted to orderly stem frequencies(f,abs(X)); %magnitudevs vs xlabel frequencies('f (Hz)'); ilabel('| X(k)|'); Extracting the correct phase spectrum is a complicated business. I'll show you why it's like that. The spectral component phase is calculated as this equation seems naïve, but we should be careful when calculating inverse tangents using computers. The obvious choice for implementation appears to be the Matlab. However, the use of the function will prove disastrous unless additional precautions are taken. The function calculates the reverse tangent only in two quadrants, i.e. it will return only in the interval. Therefore, the phase must be unpacked properly. We can simply solve this reverse tangent calculation problem over all four dials using the function. Allows to calculate and trace phase information using the function and see shows phase spectrum phase phase=atan2(imag(X),real(X)*180/pi; %phase information plot(f,phase); %phase vs frequent Phase spectrum is completely noisy. Unexpected!!!. The phase spectrum is noisy due to the fact that the inverse tangents are calculated from the imaginary part to the actual part of the FFT result. Even a small floating error floating off will amplify the result and manifest itself incorrectly as useful phase information (read how a computer program approximates very small numbers). To understand, print the first few samples of the FFT result and notice that these are not absolute zeros (these are very small numbers in order. Calculating the inverse tangent will result in incorrect results. >> X(1:5) ans = 1.0e-16 * -0.7286 -0.3637 - 0.2501i -0.4809 - 0.1579i - 0.3602 - 0.5579i 0.0261 - 0.0.3602 4950i >> atan2(imag(X(1:5)),real(X(1:5))) ans = 3.1416 -2.5391 -2.8244 -2.1441 -1.5181 The solution is to define a tolerance threshold and ignore all calculated phase values that are below the threshold. X2=X;%store FFT results in another array %detect noise (very small numbers (eps)) and ignore them threshold = max(abs(X))/10000; %tolerance threshold X2(abs(X)<tres) = 0; %maskout values below threshold=atan2(imag(X2),real(X2))*180/pi; %phase information plot(f,phase); %phase vs frequency The recomputed phase spectrum is graphically represented below. The phase spectrum correctly recorded the phase change to frequency. The phase spectrum is anti-symmetric (at), which is expected for real value signals. Rebuild the time domain signal from the frequency domain samples The reconstruction of the time domain signal in the frequency domain sample is quite simple x_recon = N * ifft (ifftshift(X),N); %reconstructed signal t = [0:1:length(x_recon)-1]/fs; %recompute time index plot(t,x_recon);%reconstructed signal The reconstructed signal has preserve the same initial phase shift and the frequency of the original signal. Note: The reconstructed signal length is only a long sample (~0.8 seconds duration), this is because the FFT size is considered as . Because the signal is periodic, it is not a concern. For more complicated signals, an appropriate FFT length (better use of a value greater than signal length) should be used. Rate this post: (103 votes, average: 4.47 out of 5) Topics in this chapter Author books