Please read through the information below prior to attending your lab. The objective of this lab is to introduce more complicated signals that are related to the basic sinusoid. These signals which implement frequency modulation (FM) and amplitude modulation (AM) are widely used in communication systems such as radio and television. In addition, they can be used to create interesting sounds that mimic musical instruments. There are a number of demonstrations on the site: `http://dspfirst.gatech.edu/chapters/03spect/demos/spectrog/` that provide examples of these signals for many different conditions. The resulting signal can be analyzed to show its time-frequency behavior by using the *spectrogram.*

This lab studies signal synthesis for AM and FM signals, and their time-frequency content as shown in a spectrogram. An underlying objective of the lab is to learn more about the spectrogram. There are several specific steps that will be considered in this lab:

1. Synthesize a beat-note signal with a MATLAB M-file, and display its spectrogram.
2. Study the frequency resolution of the spectrogram for two closely spaced sinusoids.
3. *Spectrogram:* Make empirical observations of the spectrogram as the section length is changed.
4. Synthesize a linear-FM chirp with a MATLAB M-file, and display its spectrogram.
5. *Spectrogram:* Create a spectrogram that displays negative frequencies, as well as positive ones.

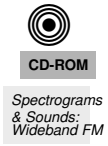
Pre-Lab

We have spent a lot of time learning about the properties of sinusoidal waveforms of the form:

$$x(t) = A\cos(2\pi f_0 t + \varphi) = \Re\left\{\left(Ae^{j\varphi}\right)e^{j2\pi f_0 t}\right\} \tag{1}$$

In this lab, we will extend our treatment of sinusoidal waveforms to more complicated signals composed of sums of sinusoidal signals, or sinusoids with changing frequency, i.e., frequency-modulated sinusoids.

2.1 Review: Chirp, or Linearly Swept Frequency



A linear-FM *chirp* signal is a sinusoid whose instantaneous frequency changes linearly from a starting value to an ending one. The formula for such a signal can be defined by creating a complex exponential signal with a quadratic angle function $\psi(t)$. Mathematically, we define $\psi(t)$ as

$$\psi(t) = 2\pi\mu t^2 + 2\pi f_0 t + \varphi \tag{2}$$

The derivative of $\psi(t)$ yields an instantaneous *cyclic* frequency that changes *linearly* versus time.

$$f_i(t) = 2\mu t + f_0 \quad \text{(hertz)} \tag{3}$$

The slope of $f_i(t)$ is equal to 2μ and its $t = 0$ intercept is f_0. The frequency variation in (3) produced by the time-varying angle function is called *frequency modulation,* so these signals are called FM signals. Finally, since the linear variation of the frequency (3) can produce an audible sound similar to a siren or a bird chirp, linear-FM signals are also called *chirps.*

If the signal starts at time $t = t_1$ s with a frequency of f_1 Hz, and ends at time $t = t_2$ s with a frequency of f_2 Hz, then the slope of the line in (3) will be

$$\text{SLOPE} = 2\mu = \frac{f_2 - f_1}{t_2 - t_1} \tag{4}$$

Note that if the signal starts at time $t = 0$ s, then $f_1 = f_0$ is also the starting frequency. Otherwise, $f_0 = ?$

Review: MATLAB Synthesis of Chirp Signals

In MATLAB signals can only be synthesized by evaluating the signal's defining formula at discrete instants of time. These are called *samples* of the signal. For the chirp we use the following:

$$x(t_n) = A\cos(2\pi\mu t_n^2 + 2\pi f_0\, t_n + \varphi)$$

where t_n is the n^{th} time sample. The following MATLAB code will synthesize a linear-FM chirp:

```
1     fSamp = 8000;     %-- Number of time samples per second
2     dt = 1/fSamp;
3     tStart = 0;
4     tStop = 1.5;
5     tt = tStart:dt:tStop;
6     mu = 600;
7     fzero = 400;
8     phi = 2*pi*rand; %-- Random phase
9  %
10 %%   psi = ????;     <================== FILL IN THE CODE HERE
11 %
12    cc = real( 7.7*exp(j*psi) );
13 %   soundsc( cc, fSamp );   %-- uncomment to hear the sound
14    plotspec( cc+j*1e-12, fSamp, 256 ), colorbar, grid on   %-- with ...
          negative frequencies
```

(a) Determine the total duration of this synthesized signal in seconds, and also the length of the `tt` vector. Use MATLAB's `size` command to check that the signal vector `cc` has the expected size.

(b) Determine the range of frequencies (in hertz) that will be synthesized by the MATLAB script above, i.e., determine the minimum and maximum frequencies (in Hz) that will be heard. This will require that you relate the parameters μ, f_0, and φ to the minimum and maximum frequencies. Examine the MATLAB spectrogram to determine the instantaneous (cyclic) frequency $f_i(t)$ versus time. Zoom in to verify the correct starting and ending frequencies.

(c) The spectrogram usually shows only the frequency components for $f \geq 0$, but with the "tiny imaginary part" trick `plotspec` will show the negative frequency components. We will called this a *two-sided spectrogram.* Since the chirp signal is *real-valued*, the spectrum must have conjugate symmetry, so the magnitudes of the negative frequency components are a mirror image of those in the positive frequency region.

(d) Use `soundsc()` to listen to the signal in order to determine whether the signal's frequency content is increasing or decreasing. Notice that `soundsc()` needs to know two things: the vector containing the signal samples, and the rate at which the signal samples are to be played out. This rate should be the same as the rate at which the signal values were created (`fSamp` in the code above).
For more information do `help sound` and `help soundsc` in MATLAB.

(e) The test case above generates a chirp sound whose frequency starts low and chirps up. Modify the parameters so that the chirp starts at 3500 Hz and chirps down to 500 Hz.

More Spectrogram: Decibels and Section Length

The periodic full-wave rectified sine wave has a known Fourier series:

$$a_k = \frac{2}{\pi(1-4k^2)} = \frac{2}{\pi(4k^2-1)}e^{j\pi}$$

Thus, in the spectrogram we should see all harmonic lines. Furthermore, there should be an infinite number of harmonic frequency lines.

Where did all the harmonics go?

The answer is that the higher harmonics have amplitudes that are too small to be seen in a spectrogram that displays values with a *linear amplitude*. Instead, a logarithmic amplitude scale is needed.

The common log scale used in engineering is decibels (dB), which is defined as $20\log_{10}(A)$ where A is amplitude. The built-in MATLAB `spectrogram` M-file uses a dB scale for amplitude when displaying its spectrogram image. The decibel has two notable features:

1. *Ratios become Differences:* On a dB scale, a numbers are represented with logarithms, so the ratio P/Q becomes
$$20\log_{10}(P/Q) = 20\log_{10}(P) - 20\log_{10}(Q)$$
If $A_2 = (1/10)A_1$ then A_2 is 20 dB lower than A_1, because with logs, we get
$$20\log_{10}(A_2) = 20\log_{10}((1/10)A_1) = 20\log_{10}(A_1) + 20\log_{10}(0.1) = 20\log_{10}(A_1) - 20\,\text{dB}.$$
Since ratios become differences, dB is most often used to compare the relative size of values.

2. The dB range must be restricted because $20\log_{10}(0) = -\infty$. If we want to map the linear amplitude range $[0, 1]$ into dB, we must define a minimum dB level. Since $20\log_{10}(1) = 0\,\text{dB}$ is the maximum, other dB values for $[0, 1]$ will be negative. The minimum dB level will chop off the bottom of $[0, 1]$ and make it equal to $[\epsilon, 1]$ where ϵ is very small. For example, a dB range of 80 dB would define the minimum to be -80 dB, so $\epsilon = 10^{-80/20} = 10^{-4} = 0.0001$.

- Express the numbers 0.1, 1, 2, 5, 10 and 100 in dB.
- Convert -6 dB, -60 dB, and -80 dB to numbers.
- In the language of dB, a factor of two is "6 dB." In other words, if B_2 is 6 dB bigger than B_1, then it is twice as big (approximately). Explain why this statement is true.
- Determine the dB difference between a_1 and a_3 for the full-wave rectified sine wave. In other words, a_3 is how many dB below a_1.
- Explain why the dB difference depends *only on the k indices.*
- Determine (in dB) how far a_{15} is below a_1 for the periodic full-wave rectified sine wave.

Another topic is displaying spectrogram of signals is *section length* which determines the time and frequency resolution. Figure 1 shows a sinusoid analyzed with a constant section length, typically in power of 2 such as 32 or 512, overlapping 50% in time expressed in different colors and its corresponding spectrogram. It is clear that at the signal transition regions the frequency content is not just a single spectral line.

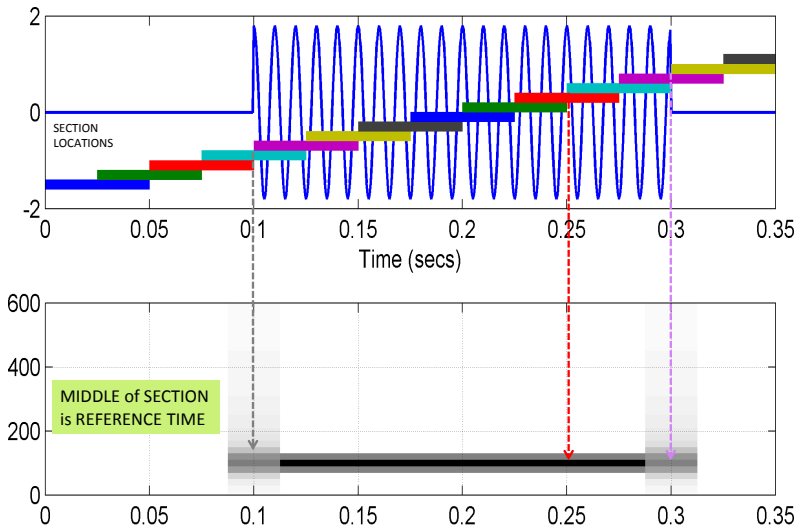


Figure 1: An illustration of the edge effect in spectrogram display.

Beat Note Spectrograms and Frequency Resolution

Beat notes as as a sum of two sinusoids provide an interesting way to investigate the time-frequency characteristics of spectrograms. Although some of the mathematical details require further study beyond this course, it is not difficult to appreciate the following issue: there is a fundamental trade-off between knowing which frequencies are present in a signal's spectrum and knowing how those frequencies vary with time. As discussed previously, a spectrogram estimates the frequency content over short sections of the signal; this is the *Section Length* parameter.[1] If we make the section length very short we can track rapid changes

[1]The section length is often called the window length; the two terms are used interchangeably in DSP.

in the signal, usually changes in the frequency content. The tradeoff, however, is that shorter sections may not provide enough data to do an accurate frequency measurement. On the other hand, long sections allow the spectrogram to perform excellent frequency measurements, but fail to track sudden frequency changes. For example, if a signal is the sum of two sinusoids whose frequencies are nearly the same, a very long section length is needed to "resolve" the two sinusoidal components. This trade-off between the section length (in time) and the frequency resolution is akin to Heisenburg's Uncertainty Principle in physics. We can summarize this discussion by stating the following hypothesis:

> The *frequency resolution* of the spectrogram is *inversely proportional to the Section Length.* In other words, when the true spectrum has two lines (at f_1 and f_2) these two lines will be visible as distinct lines in the spectrogram if $|f_1 - f_2| \approx C/T_{\text{SECT}}$ where C is a proportionality constant and T_{SECT} is the section duration in secs.

Note: When using `plotspec(xx,fs,Lsect)`, the section length ***in samples*** is an input argument to the spectrogram function. We can use the sampling rate to convert to duration, $T_{\text{SECT}} = L_{\text{SECT}}/f_s$.

We will use beat note signals which consist of two closely spaced spectral lines to confirm this hypothesis. A beat note signal may be viewed as a single frequency signal whose amplitude varies with time, *or* as the sum of two sinusoidal signals with different constant frequencies. Both views can be used to explain the effect of (window) section length when finding the spectrogram of a beat signal.

(a) Use the MATLAB code to create and plot a beat signal defined via:

$$b(t) = 10\cos(2\pi(f_\Delta)t + \varphi_\Delta)\cos(2\pi(1024)t + \varphi_c),$$

with a duration of 5 s, and a sampling rate of $f_s = 8000$ samples/sec. The frequency f_Δ should be set to 4 Hz, but will be varied in later parts. The phases can be random.

(b) When $f_\Delta = 4$ determine the locations of the two spectrum lines that you expect to see in the spectrogram. In other words, derive (mathematically) the spectrum of the signal defined in part (a).

(c) Make the spectrogram of $b(t)$ using a (window) section length of $L_{\text{SECT}} = 256$ using the commands[2]:

```
plotspec(xx,fSamp,256); colorbar, grid on, zoom on
```

Comment on what you see. Are there two spectral lines, i.e., (horizontal lines across the spectrogram)? If necessary, use the zoom tool (in the MATLAB figure window), or `zoom on`, to examine the important regions of the spectrogram.

(d) It should not be possible to see both spectrum lines with $L_{\text{SECT}} = 256$. Recreate the spectrogram with $L_{\text{SECT}} = 2048$. You should now be able to discern two distinct frequencies (you may need to zoom in to the spectral lines to distinguish them). Once you have two spectrum lines, record the value of L_{SECT} and determine whether the frequencies present in the spectrogram are correct.

Fourier Series

Another objective of this lab is to demonstrate usage of the `fseriesdemo` GUI. If you have installed the *SP-First* Toolbox, you will already have this demo on the `matlabpath`.

In this MATLAB GUI, you can choose the signal from a few pre-determined signal types and change the signal period. You can also decide the number of Fourier coefficients to compute and the corresponding magnitude and phase will be displayed accordingly.

Figure 2 shows the interface for the `fseriesdemo` GUI.

[2] Use `plotspec` instead of `specgram` in order to get a linear amplitude scale rather than logarithmic.

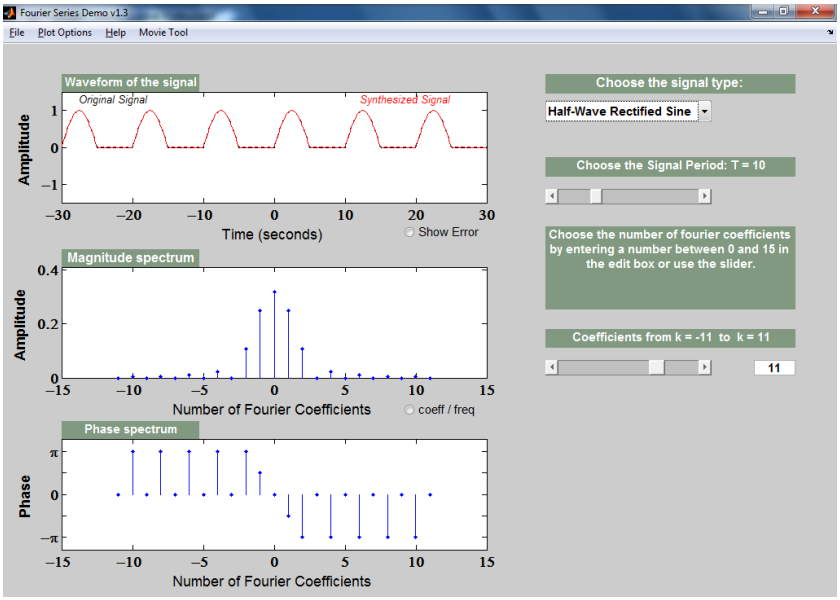


Figure 2: The **fseriesdemo** MATLAB GUI interface.

Triangle Wave and Its Fourier Series

The periodic triangular wave has a known Fourier Series. After consulting the text in Appendix C-2.2 on Fourier Series, we can write:

$$a_k = \begin{cases} \frac{-2}{\pi^2 k^2} = \frac{2}{\pi^2 k^2} e^{j\pi} & \text{for } k \text{ odd} \\ \frac{1}{2} & \text{for } k = 0 \\ 0 & \text{for } k \text{ even} \end{cases} \tag{5}$$

• Evaluate the coefficients for $k = 1$, 3, 5 and 15. Then compute the ratios a_3/a_1, a_5/a_1 and a_{15}/a_1.

Comment: Here's a general question: **are the Fourier Series coefficients independent of time scaling ?** First of all, $y(t) = x(bt)$ is scaling. For example, a plot of $x(3t)$ will be squeezed along the horizontal axis by a factor of $b = 3$.

What are the Fourier coefficients of $y(t) = x(bt)$ in terms of the Fourier coefficients of $x(t)$? If $x(t)$ has a period equal to T, then the period of $y(t)$ is T/b because $x(t)$ is squeezed by b. Thus, the fundamental frequency of $y(t)$ is $\frac{2\pi}{(T/b)} = b(\frac{2\pi}{T})$, i.e., it is scaled by b.
Now we write the Fourier Series integral for $y(t)$

$$(1/(T/b)) \int_{-T/2b}^{T/2b} y(t) e^{-j((2\pi k)/(T/b))t} dt = (b/T) \int_{-T/2b}^{T/2b} x(bt) e^{-j(2\pi k/T)(bt)} dt$$

Make a change of variables: $\lambda = bt$, and with $d\lambda = b(dt)$, you get

$$b(1/T) \int_{-T/2}^{T/2} x(\lambda) e^{-j(2\pi k/T)\lambda} (1/b) d\lambda = (1/T) \int_{-T/2}^{T/2} x(\lambda) e^{-j(2\pi k/T)\lambda} d\lambda = a_k$$

The scaling factor b cancels to give the RHS. So, time scaling doesn't change the $\{a_k\}$ coefficients. The Fourier Series coefficients of $y(t)$ are exactly equal to the Fourier Series coefficients of $x(t)$.

Use the GUI to do the following exercises. The parameters of the input signal are its frequency f_0 in Hz, and its phase φ in rads. The amplitude is one. The sampling rate for both the A/D converter and the D/A converter is f_s in samples/sec.

Theory of Sampling

In this lab, the short-duration sinusoids and music signals will be created with the intention of playing them out through a speaker. Therefore, it is necessary to take into account the fact that a conversion is needed from the digital samples, which are numbers stored in the computer memory to the actual continuous waveform that will be amplified and heard through the speakers (or headphones).

Chapter 4 treats sampling in detail, but this lab is usually done prior to lectures on sampling, so we provide a quick summary of essential facts here. The idealized process of sampling a signal and the subsequent reconstruction of the signal from its samples is depicted in Fig. 3. This figure shows a continuous-time input signal $x(t)$, which is sampled by the continuous-to-discrete (C-to-D) converter to produce a sequence of samples $x[n] = x(nT_s)$, where n is the integer sample index and T_s is the sampling period. The sampling

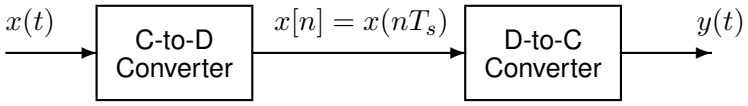


Figure 3: Sampling and reconstruction of a continuous-time signal.

rate (in samples per second) is $f_s = 1/T_s$. The discrete-to-continuous (D-to-C) converter creates a continuous output signal $y(t)$ from the samples $x[n]$. As described in Chapter 4 of the text, the ideal D-to-C converter takes the signal samples $x(nT_s)$ and interpolates a smooth curve through them. The *Sampling Theorem* tells us that when the input signal $x(t)$ is a sum of sine waves, the output $y(t)$ will be equal to the input $x(t)$ if the sampling rate is *more than twice the highest frequency* ($f_{\max}$) in the input, i.e., we need $f_s > 2f_{\max}$. In other words, if we *sample fast enough* then there will be no problems resynthesizing the continuous audio signal from $x[n]$.

A-to-D and D-to-A Conversion

Most computers have a built-in hardware for the analog-to-digital (A-to-D) converter and the digital-to-analog (D-to-A) converter (usually with a sound card or chip). These hardware systems are physical realizations of the idealized concepts of C-to-D and D-to-C converters, respectively, but for purposes of this lab we will assume that the hardware A/D and D/A are perfect realizations.

The digital-to-analog conversion process has many engineering design issues, but in its simplest form the only thing we need to worry about (in this lab) is that the time spacing (T_s) between the signal samples must correspond to the rate of the D-to-A hardware that is being used. From MATLAB, the sound output is done by the `soundsc(xx,fs)` function which does support a variable D-to-A sampling rate `(fs)` to the extent that the hardware on the machine has such variable rate capability. A convenient choice for the D-to-A conversion rate is 11025 samples per second,[3] so $T_s = 1/11025$ seconds; another common choice is 8000 samples/sec. Both of these rates satisfy the requirement of *sampling fast enough* as required by the Sampling Theorem. In fact, most piano notes have relatively low frequencies, so sampling rates much lower than 8000 samples/sec could be used. If you are using `soundsc()`, the vector `xx` will be *rescaled automatically* so that its maximum value equals the maximum allowed by the D-to-A converter, but if you are using `sound.m`, you must scale the vector `xx` so that it lies between ± 1. Consult `help sound`.

[3]This sampling rate is one quarter of the 44,100-Hz rate used in audio CD players.

you will synthesize some signals, and then study their frequency content by using the spectrogram. The objective is to learn more about the connection between the *time-domain* definition of the signal and its *frequency-domain* content.

Note: When using `plotspec(xx,fs,Lsect)`, the section length ***in samples*** is an input argument to the spectrogram function. We can use the sampling rate to convert to duration, $T_{\text{SECT}} = L_{\text{SECT}}/f_s$.

We will use beat note signals which consist of two closely spaced spectral lines to confirm this hypothesis. A beat note signal may be viewed as a single frequency signal whose amplitude varies with time, *or* as the sum of two sinusoidal signals with different constant frequencies. Both views can be used to explain the effect of (window) section length when finding the spectrogram of a beat signal.

(a) Use the MATLAB code that you used in the PreLab to create and plot a beat signal defined via:
$$b(t) = 10\cos(2\pi(f_\Delta)t + \varphi_\Delta)\cos(2\pi(1300)t + \varphi_c),$$
with a duration of 5 s, and a sampling rate of $f_s = 8000$ samples/sec. The frequency f_Δ should be set to 15 Hz. The phases can be random.

(b) When $f_\Delta = 13$ determine the locations of the two spectrum lines that you expect to see in the spectrogram. In other words, derive (mathematically) the spectrum of the signal defined in part (a).

(c) Starting with $L_{\text{SECT}} = 256$, create a spectrogram and determine if you can discern two spectrum lines. If not, a longer section length is needed, so try doubling the section length. Try $L_{\text{SECT}} = 512$, then $L_{\text{SECT}} = 1024$, and so on until you can discern two spectrum lines.[4] Then reduce the value of L_{SECT} little by little to get the smallest L_{SECT} that will work. Getting a value of L_{SECT} to the nearest 500 is sufficient.

Once you have two spectrum lines, record the value of L_{SECT} and determine whether the frequencies present in the spectrogram are correct. In addition, convert L_{SECT} to the section duration in seconds, T_{SECT}.

Spectrogram for a Chirp with Aliases

Use the code provided in the pre-Lab section as a starting point in order to write a MATLAB script or function that will synthesize a "chirp" signal. Then use that M-file in this section.

(a) What happens when we make a signal that "chirps" up to a very high frequency, and the *instantaneous frequency goes past half the sampling rate?* Generate a chirp signal that starts at 900 Hz when $t = 0$ s, and chirps up to $11,400$ Hz, at $t = 5$ s. Use $f_s = 4000$ Hz. Determine the parameters needed in Equation (2).

[4] Usually the window (section) length is chosen to be a power of two, because a special algorithm called the FFT is used in the computation. The fastest FFT programs are those where the FFT length is a power of 2.

(b) Generate the chirp signal in MATLAB and make a spectrogram with a short section length, L_{SECT}, to verify that you have the correct starting and ending frequencies.[5] For your chosen L_{SECT}, determine the section duration T_{SECT} in secs.

(c) Explain why the instantaneous frequency seen in the spectrogram is *goes up and down* between zero and $f_s/2$, i.e., it does not chirp up to 11,400 Hz. There are two effects that should be accounted for in your explanation.
Note: If possible listen to the signal to verify that the spectrogram is faithfully representing the audio signal that you hear.

Spectrogram of Periodic Full-Wave Rectified Sinusoid

A periodic signal is known to have a Fourier Series, which is usually described as a *harmonic line spectrum* because the only frequencies present in the spectrum are integer multiples of the fundamental frequency. With the spectrogram, it is easy to exhibit this harmonic line characteristic.

(a) Write a simple MATLAB script that will generate a periodic full-wave rectified sine wave once the period is given. The peak amplitude should be equal to 1. Here is a MATLAB one-liner that can form the basis of this script:
```
tt=0:(1/fs):tStop;xx=Amp*abs(sin(2*pi*tt/T));
```
The values of `fs, tStop, T, Amp` will have to be determined.

(b) Generate a full-wave rectified sine wave with T=1 sec, using a sampling rate of $f_s = 1000$ Hz. The duration should be 5 secs, and Amp = 1.

(c) Make a spectrogram with a long section duration.[6] It is important to pick a section duration that is equal to an integer number of periods of the periodic full-wave rectified sine waveform created in the previous part. Define T_{SECT} to get exactly 5 periods, and then determine the section length L_{SECT} (an integer) to be used in `plotspec`.

(d) You should expect to see a "harmonic line spectrum" in the spectrogram. Since frequency is along the vertical axis, the harmonic lines will appear as horizontal lines in the spectrogram. Make a list of all the harmonic frequencies that you can see in the spectrogram. Note the Fourier series coefficient can be found as (see Section 3-5.2): $a_k = \frac{2}{\pi(1-4k^2)} = \frac{2}{\pi(4k^2-1)}e^{j\pi}$.

(e) Determine the fundamental frequency for the harmonic lines. Note here the fundamental frequency doubles the original frequency of the sine wave after the full-wave rectification operation.

(f) Measure the magnitudes of the first and third harmonic lines by using the fseriesdemo GUI first (see Slide 21 of Lecture 07 for an illustration. Another example in in Figure 2 in the Pre-Lab part on Page 6 of Lab 05 here) and then confirm your values with the Fourier Series coefficient formula: $a_k = \frac{2}{\pi(1-4k^2)} = \frac{2}{\pi(4k^2-1)}e^{j\pi}$.

[5] There is no single correct answer for L_{SECT}, but you should pick a value that makes a smooth plot and lets you easily see the changing nature of the instantaneous frequency.

[6] A long section duration in the spectrogram yields what is called a *narrowband spectrogram* because it will provide excellent resolution of the frequency components of a signal.

Spectrogram in dB

A variation of the *SP-First* function `plotspec` has been written to incorporate the dB amplitude scale. This new function is called `plotspecDB`, and its calling template is shown below:

```
function him = plotspecDB(xx,fsamp,Lsect,DBrange)
%PLOTSPECDB   plot a Spectrogram as an image
%          (display magnitude in decibels)
%  usage:   him = plotspec(xx,fsamp,Lsect,DBrange)
%       him = handle to the image object
%        xx = input signal
%     fsamp = sampling rate
%     Lsect = section length (integer, power of 2 is a good choice)
%               amount of data to Fourier analyze at one time
%  DBrange = defines the minimum dB value; max is always 0 dB
```

(a) Create a "dB-Spectrogram" for the 5-sec periodic full-wave rectified sine wave generated in Sect. 3.3. Use a `dBrange` equal to 80 dB. Notice that many more spectrum lines are now visible. List the frequencies of all the harmonic spectrum lines, or give a general formula.

(b) Generate a second full-wave rectified sine wave by changing the period in 3.3 to T=2 sec. Then make the dB-Spectrogram of this 5-sec full-wave rectified sine wave, being careful to select the section duration as an integer number of periods. Try different values of T_{SECT} with multiplier other than 5 times the new period. Identify one value of T_{SECT} that gives the "best" dB-spectrogram.