

# Musical Signal Processing with LabVIEW – Additive Synthesis

**By:**  
Ed Doering



# Musical Signal Processing with LabVIEW – Additive Synthesis

**By:**  
Ed Doering

**Online:**  
< <http://cnx.org/content/col10479/1.1/> >

**C O N N E X I O N S**

Rice University, Houston, Texas

This selection and arrangement of content as a collection is copyrighted by Ed Doering. It is licensed under the Creative Commons Attribution 2.0 license (<http://creativecommons.org/licenses/by/2.0/>).

Collection structure revised: November 7, 2007

PDF generated: February 4, 2011

For copyright and attribution information for the modules contained in this collection, see p. 17.

## Table of Contents

<b>1 Additive Synthesis Concepts</b> .....	1
<b>2 Additive Synthesis Techniques</b> .....	5
<b>3 [ mini-project ] Risset Bell Synthesis</b> .....	9
<b>4 [ mini-project ] Spectrogram Art</b> .....	13
<b>Index</b> .....	16
<b>Attributions</b> .....	17



# Chapter 1

## Additive Synthesis Concepts<sup>1</sup>

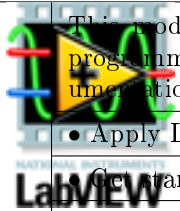
	This module refers to LabVIEW, a software development environment that features a graphical programming language. Please see the LabVIEW QuickStart Guide <sup>2</sup> module for tutorials and documentation that will help you:
	<ul style="list-style-type: none"><li>• Apply LabVIEW to Audio Signal Processing</li></ul>
	<ul style="list-style-type: none"><li>• Obtain a fully-functional evaluation edition of LabVIEW</li></ul>

Table 1.1

### 1.1 Overview

**Additive synthesis** creates complex sounds by adding together individual sinusoidal signals called **partials**. A partial's frequency and amplitude are each time-varying functions, so a partial is a more flexible version of the **harmonic** associated with a Fourier series decomposition of a periodic waveform.

In this module you will learn about partials, how to model the timbre of natural instruments, various sources of control information for partials, and how to make a sinusoidal oscillator with an instantaneous frequency that varies with time.

### 1.2 Partials

A **partial** is a generalization of the **harmonic** associated with a Fourier series representation of a periodic waveform. The screencast video of Figure 1.1 introduces important concepts associated with partials.

---

*Image not finished*

**Figure 1.1:** [video] Important concepts associated with **partials**

---

---

<sup>1</sup>This content is available online at <<http://cnx.org/content/m15444/1.2/>>.

<sup>2</sup>"NI LabVIEW Getting Started FAQ" <<http://cnx.org/content/m15428/latest/>>

### 1.3 Modeling Timbre of Natural Instruments

Perception of an instrument's **timbre** relies heavily on the attack transient of a note. Since partials can effectively enter and leave the signal at any time, additive synthesis is a good way to model physical instruments. The screencast video of Figure 1.2 discusses three important design requirements for partials necessary to successfully model a physical instrument.

---

***Image not finished***

---

**Figure 1.2:** [video] Three design requirements for partials to model physical instruments

---

### 1.4 Sources of Control Information

The time-varying frequency of a partial is called its **frequency trajectory**, and is best visualized as a track or path on a spectrogram display. Similarly, the time-varying amplitude of a partial is called its **amplitude trajectory**. Control information for these trajectories can be derived from a number of sources. Perhaps the most obvious source is a spectral analysis of a physical instrument to be modeled. The screencast video of Figure 1.3 discusses this concept and demonstrates how a trumpet can be well-modeled by adding suitable partials.



The code for the LabVIEW VI demonstrated within the video is available here: [trumpet.zip](http://trumpet.zip)<sup>3</sup>. This VI requires installation of the TripleDisplay<sup>4</sup> front-panel indicator.

---

***Image not finished***

---

**Figure 1.3:** [video] Modeling a trumpet tone by adding together increasing numbers of partials

---

Control information can also be derived from other domains not necessarily related to music. The screencast video of Figure 1.4 provides some examples of non-music control information, and illustrates how an edge boundary from an image can be "auralized" by translating the edge into a partial's frequency trajectory.

---

***Image not finished***

---

**Figure 1.4:** [video] Skyline of Houston translated into a frequency trajectory

---

<sup>3</sup><http://cnx.org/content/m15444/latest/trumpet.zip>

<sup>4</sup>"[ LabVIEW application ] TripleDisplay" <<http://cnx.org/content/m15430/latest/>>



## 1.5 Instantaneous Frequency

The frequency trajectory of a partial is defined as a time-varying frequency  $f(t)$ . Since a constant-frequency and constant-amplitude sinusoid is mathematically described as  $y(t) = A \sin(2\pi f_0 t)$ , intuition perhaps suggests that the partial should be expressed as  $y(t) = a(t) \sin(2\pi f(t)t)$ , where  $a(t)$  is the amplitude trajectory. However, as shown in the screencast video of Figure 1.5 this intuitive result is incorrect; the video derives the correct equation to describe a partial in terms of its trajectories  $f(t)$  and  $a(t)$ .

---

***Image not finished***

**Figure 1.5:** [video] Derivation of the equation of a partial given its frequency and amplitude trajectories

---



## Chapter 2

# Additive Synthesis Techniques<sup>1</sup>

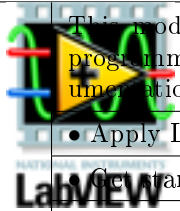
	This module refers to LabVIEW, a software development environment that features a graphical programming language. Please see the LabVIEW QuickStart Guide <sup>2</sup> module for tutorials and documentation that will help you:
	<ul style="list-style-type: none"><li>• Apply LabVIEW to Audio Signal Processing</li></ul>
	<ul style="list-style-type: none"><li>• Obtain a fully-functional evaluation edition of LabVIEW</li></ul>

Table 2.1

## 2.1 Overview

**Additive synthesis** creates complex sounds by adding together individual sinusoidal signals called **partials**. The prerequisite module Additive Synthesis Concepts (Chapter 1) reviews the main concepts of additive synthesis. In this module you will learn how to synthesize audio waveforms by designing the frequency and amplitude trajectories of the partials. Also, LabVIEW programming techniques for additive synthesis will be introduced in two examples.

## 2.2 Frequency and Amplitude Trajectory Design

A **partial** is the fundamental building block of additive synthesis. A partial is a single sinusoidal component whose amplitude and frequency are each time-varying. The time-varying amplitude denoted  $a(t)$  is called the **amplitude trajectory** and the time-varying frequency denoted  $f(t)$  is called the **frequency trajectory**. Additive synthesis requires the design of both trajectories for each partial; the partials are then summed together to create the sound.

The screencast video of Figure 2.1 shows how to begin the design of a sound as a spectrogram plot, how to design the amplitude trajectory first as an intensity (loudness) trajectory in "log space" using decibels, and how to design the frequency trajectory in "log space" using octaves. Designing the partials in log space accounts for hearing perception which is logarithmic in both intensity and in frequency; refer to Perception of Sound<sup>3</sup> for a detailed treatment of this subject.

<sup>1</sup>This content is available online at <<http://cnx.org/content/m15445/1.3/>>.

<sup>2</sup>"NI LabVIEW Getting Started FAQ" <<http://cnx.org/content/m15428/latest/>>

<sup>3</sup>"Perception of Sound" <<http://cnx.org/content/m15439/latest/>>

---

## *Image not finished*

**Figure 2.1:** [video] Design of frequency and amplitude trajectories

---

### 2.3 Example 1: Fractal Partial

In this first example, partials are created during a fixed time interval and then concatenated to create the overall sound. During the first time interval a single partial is created at a reference frequency. During the second time interval the partial's frequency linearly increases in "octave space" from the reference frequency to a frequency two octaves above the reference frequency. In the third interval the partial bifurcates into two partials, where one increases by an octave and the other decreases by an octave. In the fourth interval, each of the two partials bifurcates again to make a total of four partials, each increasing or decreasing by half an octave. This behavior repeats in each subsequent time interval, doubling the number of partials, and halving the amount of frequency increase or decrease.

The screencast video of Figure 2.2 shows how the frequency trajectories are designed in "octave space", and then reviews the key LabVIEW programming techniques needed to implement this design. The video also includes an audio demonstration so you can hear the design of this "audible fractal."



The LabVIEW VI demonstrated within the video is available here: [genfnc.zip](#)<sup>4</sup>. This VI requires installation of the TripleDisplay<sup>5</sup> front-panel indicator.

---

## *Image not finished*

**Figure 2.2:** [video] Design of the "audible fractal," LabVIEW implementation, and audio demonstration

---

### 2.4 Example 2: Spectrogram Art

The design of a sound using additive synthesis typically begins with a spectrogram representation of the desired sound. In this second example, straight line segments define the frequency trajectories of nine distinct partials that create a spectrum of a recognizable object, specifically, a cartoon drawing of an individual who is happy to be wearing a French beret.

The screencast video of Figure 2.3 shows how the frequency trajectories are designed in "octave space" and specified according to the coordinates of the line segment endpoints. The design of the corresponding amplitude trajectories necessary to make the partials start and stop at the correct times is likewise discussed. Key LabVIEW programming techniques needed to implement this design and an audio demonstration are also presented.

---

<sup>4</sup>See the file at <http://cnx.org/content/m15445/latest/genfnc.zip>

<sup>5</sup>[ LabVIEW application ] TripleDisplay" <http://cnx.org/content/m15430/latest/>



The LabVIEW VI demonstrated within the video is available here: [face.zip](#)<sup>6</sup>. This VI requires installation of the TripleDisplay<sup>7</sup> front-panel indicator.

---

## *Image not finished*

**Figure 2.3:** [video] Design of the cartoon face, LabVIEW implementation, and audio demonstration

---

---

<sup>6</sup>See the file at <http://cnx.org/content/m15445/latest/face.zip>

<sup>7</sup>"[ LabVIEW application ] TripleDisplay" <http://cnx.org/content/m15430/latest/>



# Chapter 3

## R

isset Bell Synthesis][ mini-project ] Risset Bell Synthesis<sup>1</sup>


	This module refers to LabVIEW, a software development environment that features a graphical programming language. Please see the LabVIEW QuickStart Guide <sup>2</sup> module for tutorials and documentation that will help you:
	• Apply LabVIEW to Audio Signal Processing
	• Obtain a fully-functional evaluation edition of LabVIEW

Table 3.1

### 3.1 Objective

Additive synthesis builds up complex sounds from simple sounds (sinusoids). Additive synthesis implies more than just doing Fourier series, though: each sinusoidal component is assigned its own frequency and amplitude trajectory (resulting in a partial), so complex, time-varying sounds can be generated by summing these partials together.

In this project, use additive synthesis to emulate the sound of a bell using a technique described by Jean-Claude Risset, an early pioneer in **computer music**.

### 3.2 Prerequisite Modules

If you have not done so already, please study the pre-requisite modules Additive Synthesis Concepts (Chapter 1) and Additive Synthesis Techniques (Chapter 2). If you are relatively new to LabVIEW, consider taking the course LabVIEW Techniques for Audio Signal Processing<sup>3</sup> which provides the foundation you need to complete this mini-project activity, including working with arrays, creating subVIs, playing an array to the soundcard, and saving an array as a .wav sound file.

<sup>1</sup>This content is available online at <<http://cnx.org/content/m15476/1.1/>>.

<sup>2</sup>"NI LabVIEW Getting Started FAQ" <<http://cnx.org/content/m15428/latest/>>

<sup>3</sup>*Musical Signal Processing with LabVIEW – Programming Techniques for Audio Signal Processing*  
<<http://cnx.org/content/col10440/latest/>>

### 3.3 Deliverables

- All LabVIEW code that you develop (block diagrams and front panels)
- All generated sounds in .wav format
- Any plots or diagrams requested
- Summary write-up of your results

### 3.4 Description of the Risset Bell

Jean-Claude Risset's book *Introductory Catalogue of Computer-Synthesized Sounds* (Bell Telephone Laboratories, 1969) includes an additive synthesis method for a bell-like sound. After analyzing the ringing characteristics of physical bells, he determined that a bell-like tone could be created by using non-harmonic partials whose decay times are approximately inversely proportional to frequency. In addition, pairs of low-frequency partials with a slight frequency offset create a "beating" effect.

The partials for the Risset bell are available in this CSV-type spreadsheet: `risset_bell.csv`<sup>4</sup>. The columns are:

1. partial number
2. intensity in dB
3. duration multiplier (indicates fraction of overall duration)
4. frequency multiplier (indicates interval ratio from base frequency)
5. frequency offset (in Hz)

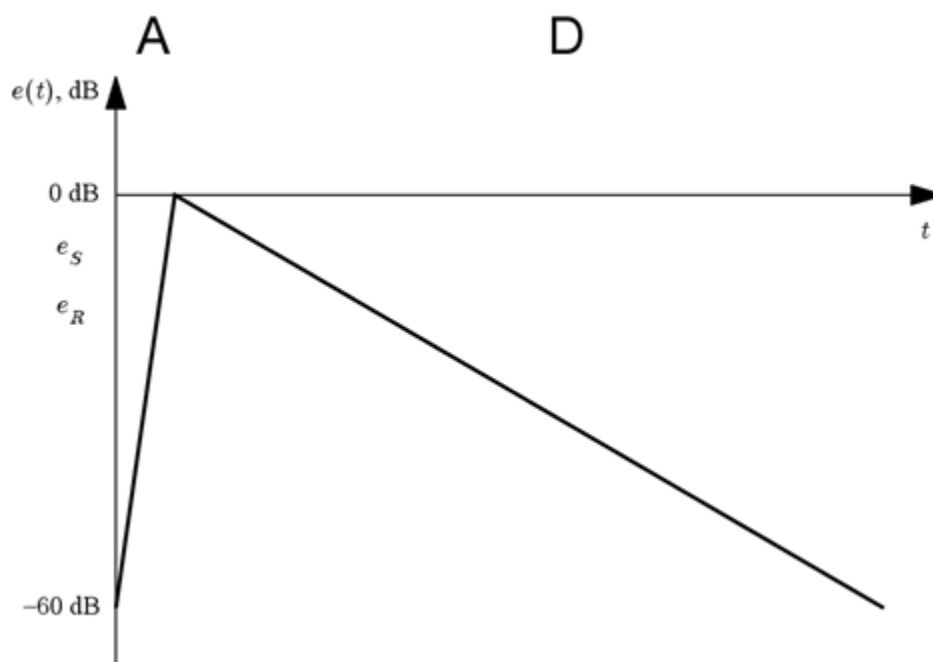
### 3.5 Part 1: Attack/Decay Envelope Generator

Create an attack/decay intensity envelope composed of two straight-line segments as shown in Figure 3.1:

---

<sup>4</sup>[http://cnx.org/content/m15476/latest/risset\\_bell.csv](http://cnx.org/content/m15476/latest/risset_bell.csv)





**Figure 3.1:** Attack/decay envelope specification

The value "-60 dB" corresponds to an amplitude of 0.001, effectively silence. The total duration of the envelope should be based on a front-panel control called "duration [s]" (its unit is seconds), and another control called "attack [ms]" (its unit is milliseconds).

Convert the intensity envelope into an amplitude envelope by "undoing" the equation for decibels.

### 3.6 Part 2: Multiple Envelopes

Enclose your envelope generator in a for-loop structure which takes values from `riset_bell.csv`<sup>5</sup>. Connect a control to the count terminal so that you can adjust how many envelopes to view.

Use "auto indexing" on the for-loop to create an array of envelopes, then view the array as graph (you will see all of the envelopes superimposed on the same graph, with each envelope a different color). Confirm that your envelopes have the correct amplitudes and durations.

Plot both the intensity envelopes and amplitude envelopes for all partials, and include these plots with your deliverables.

### 3.7 Part 3: Sinusoidal Tone Generator

Add a sinusoidal tone generator to your main loop. The base frequency should be set by a front-panel control called "freq [Hz]", and the frequency multiplier and offset for each partial should be used to set the actual

<sup>5</sup>[http://cnx.org/content/m15476/latest/riset\\_bell.csv](http://cnx.org/content/m15476/latest/riset_bell.csv)

frequency. Also include a front-panel control to select the system sampling frequency "fs [Hz]".

Apply the amplitude envelope, and then add the partial to the sound that was generated on the previous pass of the loop. Regardless of whether you use a shift register or a feedback node, initialize the audio signal with an array of zeros which is of the same length as your envelopes.

Remember to use the "QuickScale" built-in subVI on the finished audio waveform to ensure that all of its values lie in the range  $\pm 1$ .

### 3.8 Part 4: Experiment with Parameters

Your front panel controls should include the following adjustable parameters: number of partials, total duration, base frequency, sampling frequency, and attack time. Listen to the audio and view the spectrogram as you adjust the parameters.

To get started, listen to the sound with the first partial only, and then with the first two partials, and so on until you hear all eleven partials. Building up the sound from silence by adding more and more partials is the essence of "additive synthesis."

Try varying the attack time. What is the maximum attack time that still sounds like the striking of a bell?

Try adjusting the total duration and base frequency. Remember to adjust the sampling frequency high enough so that you do not produce aliasing. If the sampling frequency is too high, however, all of the partials will compress into the bottom of the spectrogram plot.

Overall, what values produce a realistic-sounding bell?

Create .wav files for three distinct sets of parameters, and discuss the motivation for your choices. Include a spectrogram plot for each of the three bell sounds.

# Chapter 4

## S

pectrogram Art][ mini-project ] Spectrogram Art<sup>1</sup>

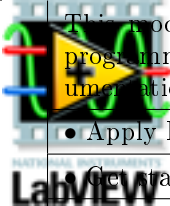
	This module refers to LabVIEW, a software development environment that features a graphical programming language. Please see the LabVIEW QuickStart Guide <sup>2</sup> module for tutorials and documentation that will help you:
	• Apply LabVIEW to Audio Signal Processing
	• Obtain a fully-functional evaluation edition of LabVIEW

Table 4.1

## 4.1 Objective

Additive synthesis builds up complex sounds from simple sounds (sinusoids). Additive synthesis implies more than just doing Fourier series, though: each sinusoidal component is assigned its own frequency and amplitude trajectory (resulting in a partial), so complex, time-varying sounds can be generated by summing these partials together.

In this project you will create an oscillator whose output tracks a specified amplitude and frequency trajectory. With this general-purpose oscillator you can define multiple frequency/amplitude trajectories that can be combined to create complex sounds. In particular, you will design the sound so that its spectrogram makes a recognizable picture!

## 4.2 Prerequisite Modules

If you have not done so already, please study the prerequisite modules Additive Synthesis Concepts (Chapter 1) and Additive Synthesis Techniques (Chapter 2). If you are relatively new to LabVIEW, consider taking the course LabVIEW Techniques for Audio Signal Processing<sup>3</sup> which provides the foundation you need to complete this mini-project activity, including working with arrays, creating subVIs, playing an array to the soundcard, and saving an array as a .wav sound file.

<sup>1</sup>This content is available online at <<http://cnx.org/content/m15446/1.1/>>.

<sup>2</sup>"NI LabVIEW Getting Started FAQ" <<http://cnx.org/content/m15428/latest/>>

<sup>3</sup>*Musical Signal Processing with LabVIEW – Programming Techniques for Audio Signal Processing*  
<<http://cnx.org/content/col10440/latest/>>

### 4.3 Deliverables

- All LabVIEW code that you develop (block diagrams and front panels)
- All generated sounds in .wav format
- Any plots or diagrams requested
- Summary write-up of your results

### 4.4 Part 1: General-Purpose Sinusoidal Oscillator

Develop a subVI called **gposc.vi** that accepts a frequency trajectory (in Hz), an amplitude trajectory, and a sampling frequency (in Hz) to produce a sinusoidal output whose amplitude and frequency tracks the two input trajectories, respectively. The two trajectories are arrays that should be of the same length.

Demonstrate that your oscillator works properly by showing the output of your VI (spectrogram and .wav file) for the amplitude and frequency trajectories produced by a LabVIEW MathScript node that contains the following code:

```
ff=[linspace(200,1600,2.5*fs) ...
    linspace(1600,800,1.5*fs)];
```

```
aa=[linspace(1,0,3*fs) ...
    linspace(0,0.75,fs)];
```

where **fs** is the sampling frequency in Hz, **ff** is the output frequency trajectory (also in Hz), and **aa** is the amplitude trajectory (between 0 and 1). Use a sampling frequency of 5 kHz when you make the spectrogram and soundfile.

Plot the trajectories **ff** and **aa** and compare to your spectrogram.

Remember, the instantaneous frequency of your general-purpose sinusoidal oscillator is related to the time-varying phase of the sine function. That is, if the sinusoidal signal is defined as  $y(t) = \sin(\theta(t))$ , then the instantaneous frequency of the sinusoid is  $\omega(t) = d\theta(t)/dt$  radians per second. Because you are given a frequency trajectory that relates to  $\omega(t)$ , which mathematical operation yields the phase function  $\theta(t)$ ?

Here's a LabVIEW coding tip: You will find the built-in VI "Mathematics | Integ and Diff | Integral x(t)" to be essential for this part of the project.

### 4.5 Part 2: Frequency Trajectory Design

You can make your spectrogram art project sound more musically appealing when you design the frequency trajectories to account for frequency **perception**; refer to Perception of Sound<sup>4</sup> for a detailed treatment of this subject. Design your trajectories in "log space" (using logarithmic graph paper) and then convert to actual frequency just before invoking your general-purpose sinusoidal oscillator.

Review Additive Synthesis Techniques (Chapter 2) to learn how to create your frequency trajectories for this part of the project.

### 4.6 Part 3: Amplitude Trajectory Design

The discussion of Part 2 pertains to the design of your amplitude trajectories, as well. Perception of intensity (loudness) is also logarithmic (refer to Perception of Sound<sup>5</sup> and review the section on intensity perception). In this part you will design your amplitude trajectory in "log space," but now using traditional decibels

<sup>4</sup>"Perception of Sound" <<http://cnx.org/content/m15439/latest/>>

<sup>5</sup>"Perception of Sound" <<http://cnx.org/content/m15439/latest/>>

(dB). An intensity trajectory can be converted to amplitude by "undoing" the equation that relates a value to the same value expressed in decibels:  $X_{\text{dB}} = 20\log_{10}(X)$ .

Experiment with your spectrogram display device to learn the intensity-to-color mapping. Specifically, you could produce a sinusoidal signal with increasing intensity values over time, then match up the plotted colors to the known intensity values.

## 4.7 Part 4: Spectrogram Art

Design a spectrogram picture using multiple frequency/amplitude trajectories. Include your paper-and-pencil drawing of the spectrogram as part of your deliverables. Use your creativity to make an interesting and recognizable picture.

Better designs will go beyond straight lines to include curved lines such as arcs, exponentials, parabolas, sinusoids, polynomials, spline interpolations, and so on.

Include a .wav file of the sound associated with your spectrogram picture.

## Index of Keywords and Terms

**Keywords** are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

- |  |  |
|--|--|
| <p><b>A</b> additive synthesis, § 1(1), 1, § 2(5), 5, § 3(9), § 4(13)<br/>amplitude trajectory, § 1(1), 2, § 2(5), 5, § 3(9), § 4(13)</p> <p><b>B</b> bell, § 3(9)</p> <p><b>C</b> computer music, 9</p> <p><b>E</b> envelope generator, § 3(9)</p> <p><b>F</b> frequency trajectory, § 1(1), 2, § 2(5), 5, § 3(9), § 4(13)</p> <p><b>G</b> gposc.vi, 14</p> | <p><b>H</b> harmonic, 1, 1</p> <p><b>I</b> instantaneous frequency, § 1(1)</p> <p><b>L</b> LabVIEW, § 2(5), § 3(9), § 4(13)</p> <p><b>O</b> octave space, § 2(5)</p> <p><b>P</b> partial, § 1(1), 1, § 2(5), 5, § 3(9), § 4(13)<br/>partials, 1, 1, 5</p> <p><b>S</b> spectrogram, § 4(13)</p> <p><b>T</b> timbre, § 1(1), 2</p> |
|--|--|

## Attributions

Collection: *Musical Signal Processing with LabVIEW – Additive Synthesis*

Edited by: Ed Doering

URL: <http://cnx.org/content/col10479/1.1/>

License: <http://creativecommons.org/licenses/by/2.0/>

Module: "Additive Synthesis Concepts"

By: Ed Doering

URL: <http://cnx.org/content/m15444/1.2/>

Pages: 1-3

Copyright: Ed Doering

License: <http://creativecommons.org/licenses/by/2.0/>

Module: "Additive Synthesis Techniques"

By: Ed Doering

URL: <http://cnx.org/content/m15445/1.3/>

Pages: 5-7

Copyright: Ed Doering

License: <http://creativecommons.org/licenses/by/2.0/>

Module: "[ mini-project ] Risset Bell Synthesis"

By: Ed Doering

URL: <http://cnx.org/content/m15476/1.1/>

Pages: 9-12

Copyright: Ed Doering

License: <http://creativecommons.org/licenses/by/2.0/>

Module: "[ mini-project ] Spectrogram Art "

By: Ed Doering

URL: <http://cnx.org/content/m15446/1.1/>

Pages: 13-15

Copyright: Ed Doering

License: <http://creativecommons.org/licenses/by/2.0/>

### **Musical Signal Processing with LabVIEW – Additive Synthesis**

Additive synthesis creates complex sounds by adding together individual sinusoidal signals called partials. A partial's frequency and amplitude are each time-varying functions, so a partial is a more flexible version of the harmonic associated with a Fourier series decomposition of a periodic waveform. Learn about partials, how to model the timbre of natural instruments, various sources of control information for partials, and how to make a sinusoidal oscillator with an instantaneous frequency that varies with time. This course is part of the series "Musical Signal Processing with LabVIEW".

### **About Connexions**

Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.