Chapter 20
Music

What does $\sqrt[12]{2}$ *have to do with music?*

In the theory of music, an *octave* is an interval with frequencies that range over a factor of two. In most Western music, an octave is divided into twelve *semitones* with equal frequency ratios. Since twelve semitones comprise a factor of two, one semitone is a factor of $\sqrt[12]{2}$. And because this quantity occurs so often in this chapter, let

$$\sigma = \sqrt[12]{2}$$

Our MATLAB programs use

```
sigma = 2^(1/12)
      = 1.059463094359295
```

Think of σ as an important mathematical constant, like π and ϕ.

Keyboard

Figure 20.1 shows our miniature piano keyboard with 25 keys. This keyboard has two octaves, with white keys labeled `C D ... G A B`, plus another `C` key. Counting both white and black, there are twelves keys in each octave. The frequency of each key is a semitone above and below its neighbors. Each black key can be regarded as either the *sharp* of the white below it or the *flat* of the white above it. So the black key between `C` and `D` is both `C`♯ and `D`♭. There is no `E`♯/`F`♭ or `B`♯/`C`♭.

A conventional full piano keyboard has 88 keys. Seven complete octaves account for $7 \times 12 = 84$ keys. There are three additional keys at the lower left and one additional key at the upper end. If the octaves are numbered 0 through 8, then

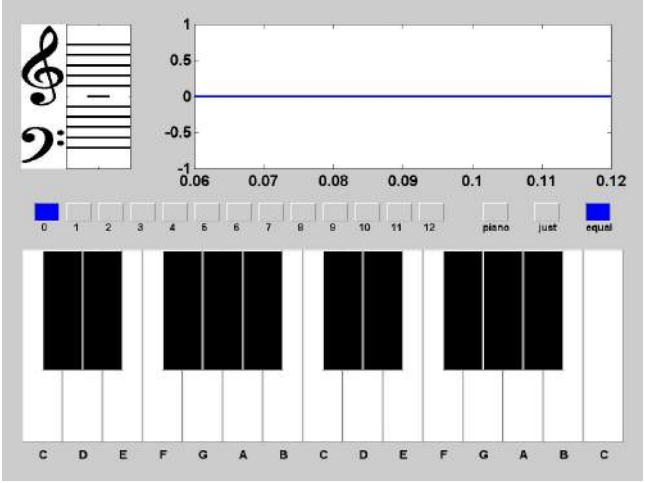

Figure 20.1. *Miniature piano keyboard.*

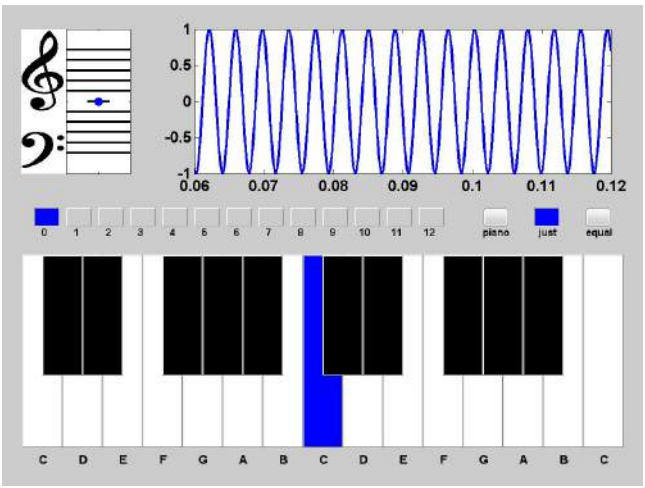

Figure 20.2. *Middle C.*

a key letter followed by an octave number specifies a unique key. In this notation, two important keys are `C4` and `A4`. The `C4` key is near the center of the keyboard and so is also known as *middle C*. A piano is usually tuned so that the frequency of the `A4` key is 440 Hz. `C4` is nine keys to the left of `A4` so its frequency is

$$\texttt{C4} = 440\sigma^{-9} \approx 261.6256\ Hz$$

Our `EXM` program `pianoex` uses `C4` as the center of its 25 keys, so the number

range is `-12:12`. The statement

```
pianoex(0)
```

generates and plays the sound from a sine wave with frequency `C4`. The resulting visual display is shown in figure 20.2.

This `for` loop plays a two octave chromatic scale starting covering all 25 notes on our miniature keyboard.

```
for n = -12:12
   pianoex(n)
end
```

Do Re Mi

One of the first songs you learned to sing was

Do Re Mi Fa So La Ti Do

If you start at `C4`, you would be singing the *major scale* in the key of C. This scale is played on a piano using only the white keys. The notes are not equally spaced. Most of the steps skip over black keys and so are two semitones. But the steps between *Mi* and *Fa* and *Ti* and *Do* are the steps from `E` to `F` and `B` to `C`. There are no intervening black keys and so these steps are only one semitone. In terms of σ, the C-major scale is

$$\sigma^0 \ \sigma^2 \ \sigma^4 \ \sigma^5 \ \sigma^7 \ \sigma^9 \ \sigma^{11} \ \sigma^{12}$$

You can play this scale on our miniature keyboard with

```
for n = [0 2 4 5 7 9 11 12]
   pianoex(n)
end
```

The number of semitones between the notes is given by the vector

```
diff([0 2 4 5 7 9 11 12])
  =  [2  2  1  2  2  2  1]
```

The sequence of frequencies in our most common scale is surprising. Why are there 8 notes in the C-major scale? Why don't the notes in the scale have uniform frequency ratios? For that matter, why is the octave divided into 12 semitones? The notes in "*Do Re Me*" are so familiar that we don't even ask ourselves these questions. Are there mathematical explanations? I don't have definitive answers, but I can get some hints by looking at harmony, chords, and the ratios of small integers.

Vibrations and modes

Musical instruments create sound through the action of vibrating strings or vibrating columns of air that, in turn, produce vibrations in the body of the instrument

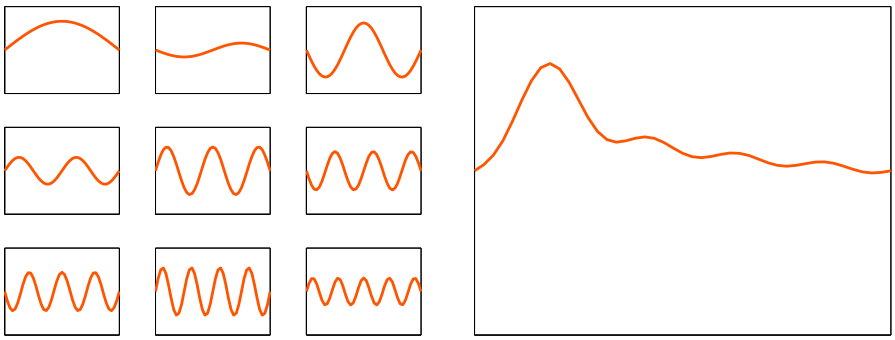

Figure 20.3. *The first nine modes of a vibrating string, and their weighted sum.*

and the surrounding air. Mathematically, vibrations can be modeled by weighted sums of characteristic functions known as *modes* or *eigenfunctions*. Different modes vibrate at different characteristic frequencies or *eigenvalues*. These frequencies are determined by physical parameters such as the length, thickness and tension in a string, or the geometry of the air cavity. Short, thin, tightly stretched strings have high frequencies, while long, thick, loosely stretched strings have low frequencies.

The simplest model is a one-dimensional vibrating string, held fixed at its ends. The units of the various physical parameters can be chosen so that the length of the string is 2π. The modes are then simply the functions

$$v_k(x) = \sin kx, \quad k = 1, 2, ...$$

Each of these functions satisfy the fixed end point conditions

$$v_k(0) = v_k(2\pi) = 0$$

The time-dependent modal vibrations are

$$u_k(x,t) = \sin kx \sin kt, \quad k = 1, 2, ...$$

and the frequency is simply the integer k. (Two- and three-dimensional models are much more complicated, but this one-dimensional model is all we need here.)

Our `EXM` program `vibrating_string` provides a dynamic view. Figure 20.3 is a snapshot showing the first nine modes and the resulting wave traveling along the string. An exercise asks you to change the coefficients in the weighted sum to produce different waves.

Lissajous figures

Lissajous figures provide some insight into the mathematical behavior of musical chords. Two dimensional Lissajous figures are plots of the parametric curves

$$x = \sin(at + \alpha), \; y = \sin(bt + \beta)$$

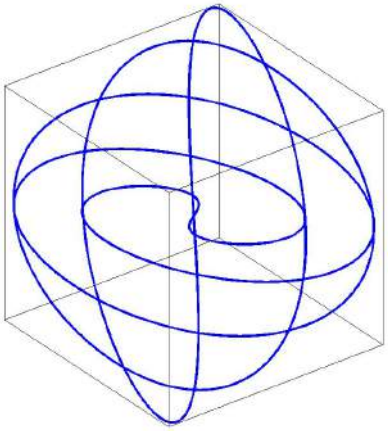

Figure 20.4. $x = \sin t, y = \sin 3/2t, z = \sin 5/4t.$

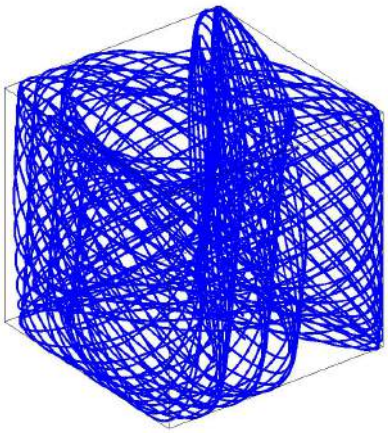

Figure 20.5. $x = \sin t, y = \sin \sigma^7 t, z = \sin \sigma^4 t.$

Three dimensional Lissajous figures are plots of the parametric curves

$$x = \sin(at + \alpha),\ y = \sin(bt + \beta),\ z = \sin(ct + \gamma)$$

We can simplify our graphics interface by just considering

$$x = \sin t,\ y = \sin at$$

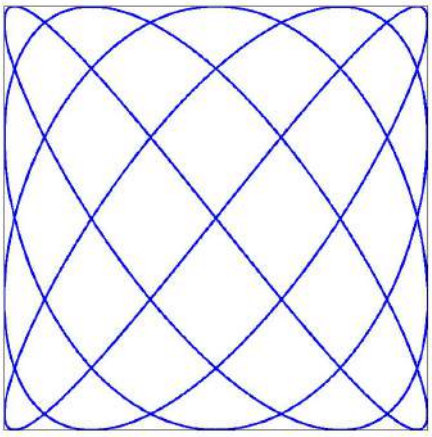

Figure 20.6. $x = \sin t, y = \sin 5/4t.$

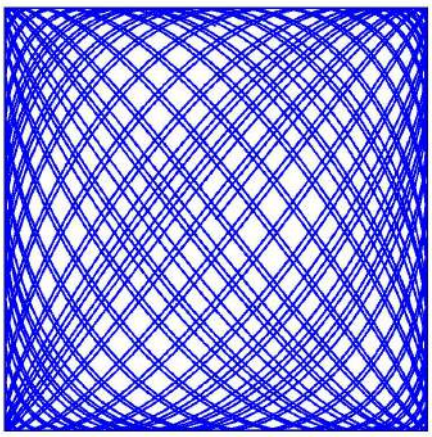

Figure 20.7. $x = \sin t, y = \sin \sigma^4 t.$

and

$$x = \sin t,\ y = \sin at,\ z = \sin bt$$

The example with $a = 3/2$ and $b = 5/4$ shown in figure 20.4. is produced by the default settings in our `exm` program `lissajous`. This program allows you to change

a and b by entering values in *edit boxes* on the figure. Entering `b = 0` results in a two dimensional Lissajous figure like the one shown in figure 20.6.

The simplest, "cleanest" Lissajous figures result when the parameters a and b are ratios of small integers.

$$a = \frac{p}{q},\ b = \frac{r}{s},\quad p, q, r, s = \text{small integers}$$

This is because the three functions

$$x = \sin t,\ y = \sin \frac{p}{q}t,\ z = \sin \frac{r}{s}t$$

all return to zero when

$$t = 2m\pi,\quad m = \text{lcm}(q, s)$$

where $\text{lcm}(q, s)$ is the *least common multiple* of q and s. When a and b are fractions with large numerators and denominators, the curves oscillate more rapidly and take longer to return to their starting values.

In the extreme situation when a and b are irrational, the curves never return to their starting values and, in fact, eventually fill up the entire square or cube.

We have seen that dividing the octave into 12 equal sized semitones results in frequencies that are powers of σ, an irrational value. The `E` key and `G` keys are four and seven semitones above `C`, so their frequencies are

```
sigma^4
   = 1.259921049894873

sigma^7
   = 1.498307076876682
```

The closest fractions with small numerator and denominator are

```
5/4
   = 1.250000000000000

3/2
   = 1.500000000000000
```

This is why we chose 5/4 and 3/2 for our default parameters. If the irrational powers of `sigma` are used instead, the results are figures 20.5 and 20.7. In fact, these figures are merely the initial snapshots. If we were to let the program keep running, the results would not be pleasant.

Harmony and Intonation

Harmony is an elusive attribute. Dictionary definitions involve terms like "pleasing", "congruent", "fitting together". For the purposes of this chapter, let's say that two or more musical notes are *harmonious* if the ratios of their frequencies are rational

numbers with small numerator and denominator. The human ear finds such notes fit together in a pleasing manner.

Strictly speaking, a musical chord is three or more notes sounded simultaneously, but the term can also apply to two notes. With these definitions, chords made from a scale with equal semitones are not exactly harmonious. The frequency ratios are powers of σ, which is irrational.

Tuning a musical instrument involves adjusting its physical parameters so that it plays harmonious music by itself and in conjunction with other instruments. Tuning a piano is a difficult process that is done infrequently. Tuning a violin or a guitar is relatively easy and can be done even during breaks in a performance. The human singing voice is an instrument that can undergo continuous retuning.

For hundreds of years, music theory has included the design of scales and the tuning of instruments to produce harmonious chords. Of the many possibilities, let's consider only two – *equal temperament* and *just intonation*.

Equal temperament is the scheme we've been describing so far in this chapter. The frequency ratio between the notes in a chord can be expressed in terms of σ. Tuning an instrument to have equal temperament is done once and for all, without regard to the music that will be played. A single base note is chosen, usually `A` = 440 Hz, and that determines the frequency of all the other notes. Pianos are almost always tuned to have equal temperament.

Just intonation modifies the frequencies slightly to obtain more strictly harmonious chords. The tuning anticipates the key of the music about to be played. Barbershop quartets and *a capella* choirs can obtain just intonation dynamically during a performance.

Here is a MATLAB code segment that compares equal temperament with just intonation from a strictly numerical point of view. Equal temperament is defined by repeated powers of σ. Just intonation is defined by a sequence of fractions.

```
sigma = 2^(1/12);
k = (0:12)';
equal = sigma.^k;

num = [1 16 9 6 5 4 7 3 8 5 7 15 2]';
den = [1 15 8 5 4 3 5 2 5 3 4  8 1]';
just = num./den;

delta = (equal - just)./equal;
T = [k equal num den just delta];
fprintf('%8d %12.6f %7d/%d %10.6f %10.4f\n',T')
```

```
 k      equal              just           delta

 0     1.000000       1/1   1.000000     0.0000
 1     1.059463      16/15  1.066667    -0.0068
 2     1.122462       9/8   1.125000    -0.0023
 3     1.189207       6/5   1.200000    -0.0091
 4     1.259921       5/4   1.250000     0.0079
```

```
 5     1.334840      4/3   1.333333     0.0011
 6     1.414214      7/5   1.400000     0.0101
 7     1.498307      3/2   1.500000    -0.0011
 8     1.587401      8/5   1.600000    -0.0079
 9     1.681793      5/3   1.666667     0.0090
10     1.781797      7/4   1.750000     0.0178
11     1.887749     15/8   1.875000     0.0068
12     2.000000      2/1   2.000000     0.0000
```

The last column in the table, `delta`, is the relative difference between the two. We see that `delta` is less than one percent, except for one note. But the more important consideration is how the music sounds.

Chords

.

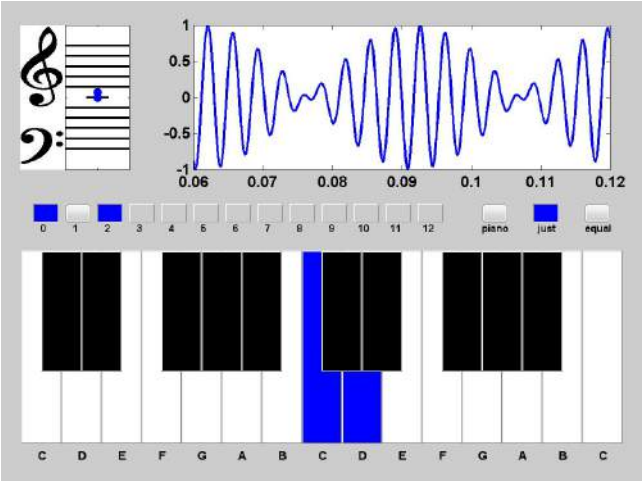

Figure 20.8. *Dissonace and beats between two adjacent whole notes.*

Chords are two or more notes played simultaneously. With a computer keyboard and mouse, we can't click on more than one key at a time. So chords are produced with `pianoex` by selecting the toggle switches labeled `1` through `12`. The switch labeled `0` is always selected.

Figure 20.8 shows the visual output generated when `pianoex` plays a chord involving two adjacent white keys, in this case `C` and `D`. You can see, and hear, the phenomenon known as *beating*. This occurs when tones with nearly equal frequencies alternate between additive reinforcement and subtractive cancellation. The relevant trig identity is

$$\sin at + \sin bt = \sin \frac{a+b}{2}t \;\; \cos \frac{a-b}{2}t$$

The sum of two notes is a note with the average of the two frequencies, modulated by a cosine term involving the difference of the two frequencies. The players in an orchestra tune up by listening for beats between their instruments and a reference instrument.

The most important three-note chord, or *triad*, is the *C major fifth*. If `C` is the lowest, or root, note, the chord is `C-E-G`. In just intonation, the frequency ratios are

$$1 : \frac{5}{4} : \frac{3}{2}$$

These are the parameter values for our default Lissajous figure, shown in figure 20.4. Figures 20.9 and 20.10 show the visual output generated when `pianoex` plays a C major fifth with just intonation and with equal temperament. You can see that the wave forms in the oscilloscope are different, but can you hear any difference in the sound generated?

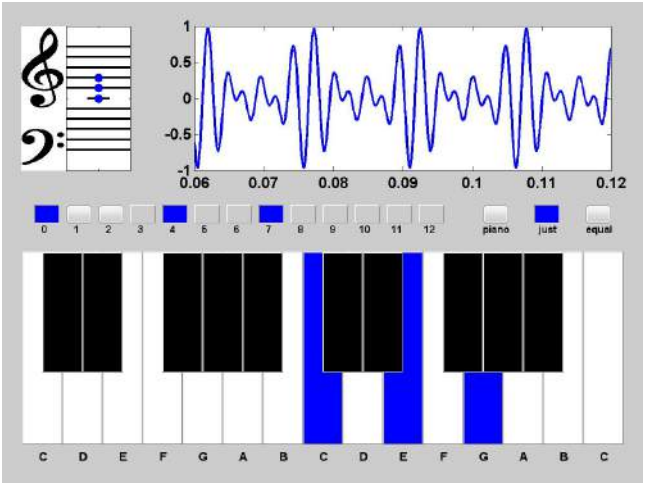

Figure 20.9. *C major fifth with just intonation.*

Synthesizing Music

Our `pianoex` program is not a powerful music synthesizer. Creating such a program is a huge undertaking, way beyond the scope of this chapter or this book. We merely want to illustrate a few of the interesting mathematical concepts involved in music.

The core of the `pianoex` is the code that generates a vector `y` representing the amplitude of sound as a function of time `t`. Here is the portion of the code that handles equal temperament. The quantity `chord` is either a single note number, or a vector like [0 4 7] with the settings of the chord toggles.

```
sigma = 2^(1/12);
```

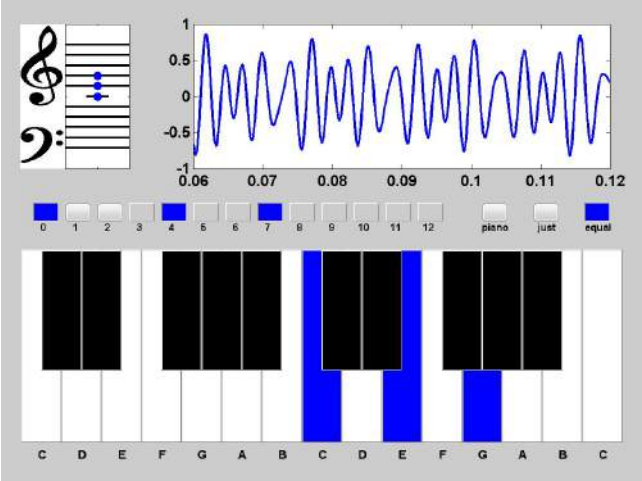

Figure 20.10. *C major fifth with equal temperament.*

```
C4 = 440*sigma^(-9);
fs = 44100;
t = 0:1/fs:T;
y = zeros(size(t));
for n = chord
   hz = C4 * sigma^n;
   y = y + sin(2*pi*hz*t);
end
y = y/length(chord);
```

Here is the corresponding portion of code for just intonation. The vector `r` of ratios is repeated a few times, scaled by powers of 2, to cover several octaves.

```
sigma = 2^(1/12);
C4 = 440*sigma^(-9);
fs = 44100;
t = 0:1/fs:T;
r = [1 16/15 9/8 6/5 5/4 4/3 7/5 3/2 8/5 5/3 7/4 15/8];
r = [r/2 r 2*r 4];
y = zeros(size(t));
for n = chord
   hz = C4 * r(n+13);
   y = y + sin(2*pi*hz*t);
end
y = y/length(chord);
```

A small example of what a full music synthesizer would sound like is provided by the "piano" toggle on `pianoex`. This synthesizer uses a single sample of an actual

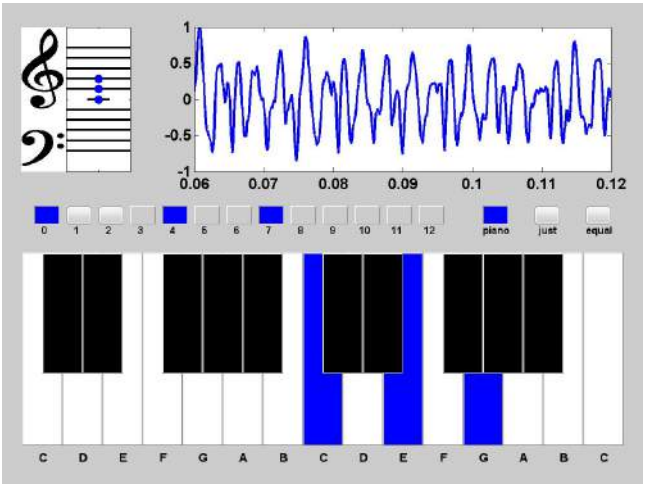

Figure 20.11. *C major fifth chord with simulated piano.*

piano playing middle C. This sample is loaded during the initialization of `pianoex`,

```
S = load('piano_c.mat');
middle_c = double(S.piano_c)/2^15;
set(gcf,'userdata',middle_c)
```

A function from the MATLAB Signal Processing Toolbox is then used to generate notes at different frequencies.

```
middle_c = get(gcf,'userdata');
fs = 44100;
t = 0:1/fs:T;
y = zeros(size(t));
for n = chord
   y = y + resamplex(middle_c,2^(n/12),length(y));
end
```

Figure 20.11 displays the piano simulation of the C major fith chord. You can see that the waveform is much richer than the ones obtained from superposition of sine waves.

Further Reading, and Viewing

This wonderful video shows a performance of the "Do Re Mi" song from "The Sound of Music" in the Antwerp Central Railway Station. (I hope the URL persists.)

`http://www.youtube.com/watch?v=7EYAUazLI9k`

Wikipedia has hundreds of articles on various aspects of music theory. Here is one:

http://en.wikipedia.org/wiki/Music_and_mathematics

Wikipedia on Lissajous curves:

http://en.wikipedia.org/wiki/Lissajous_curve

Recap

```
%% Music Chapter Recap
% This is an executable program that illustrates the statements
% introduced in the Music Chapter of "Experiments in MATLAB".
% You can access it with
%
%    music_recap
%    edit music_recap
%    publish music_recap
%
% Related EXM programs
%
%    pianoex

%% Size of a semitone, one twelth of an octave.

   sigma = 2^(1/12)

%% Twelve pitch chromatic scale.

   for n = 0:12
      pianoex(n)
   end

%% C major scale

   for n = [0 2 4 5 7 9 11 12]
      pianoex(n)
   end

%% Semitones in C major scale

   diff([0 2 4 5 7 9 11 12])

%% Equal temperament and just intonation

   [sigma.^(0:12)
    1 16/15 9/8 6/5 5/4 4/3 7/5 3/2 8/5 5/3 7/4 15/8 2]'

%% C major fifth chord, equal temperament and just temperament
```

```
[sigma.^[0 4 7]
 1 5/4 3/2]'
```

Exercises

20.1 *Strings.* The Wikipedia page

```
http://en.wikipedia.org/wiki/Piano_key_frequencies/
```

has a table headed "Virtual Keyboard" that shows the frequencies of the piano keys as well as five string instruments. The open string violin frequencies are given by

```
v = [-14 -7 0 7]'
440*sigma.^v
```

What are the corresponding vectors for the other four string instruments?

20.2 *Vibrating string.* In `vibrating_string.m`, find the statement

```
a = 1./(1:9)
```

Change it to

```
a = 1./(1:9).^2
```

or

```
a = 1./sqrt(1:9)
```

Also, change the loop control

```
for k = 1:9
```

to

```
for k = 1:2:9
```

or

```
for k = 1:3:9
```

What effect do these changes have on the resulting wave?

20.3 *Comet.* Try this:

```
a = 2/3;
b = 1/2;
```

```
tfinal = 12*pi;
t = 0:pi/512:tfinal;
x = sin(t);
y = sin(a*t);
z = sin(b*t);
comet3(x,y,z)
```

Why did I choose this particular value of `tfinal`? How does this `tfinal` depend upon `a` and `b`?

20.4 *Dissonant Lissajous.* What is the Lissajous figure corresponding to two or three adjacent keys, or adjacent white keys, on the piano keyboard?

20.5 *Irrational biorhythms.* The biorhythms described in our "Calendars and Clocks" chapter are based on the premise that our lives are governed by periodic functions of time with periods of 23, 28 and 33 days. What is the effect of revising `biorhythms.m` so that the periods are irrational values near these?

20.6 *Just intonation.* With just intonation, the ratios of frequencies of adjacent notes are no longer equal to σ. What are these ratios?

20.7 *Rational intonation.* MATLAB *almost* has the capability to discover just intonation. The MATLAB function `rat` computes rational approximations. For example, the following statement computes the numerator `n` and denominator `d` in a rational approximation of π.

```
[n,d] = rat(pi)
n =
   355
d =
   113
```

This gives us the approximation $\pi \approx 355/113$, which is accurate to 7 significant figures. For a less accurate approximation, specify a tolerance of 2 percent.

```
[n,d] = rat(pi,.02)
n =
    22
d =
     7
```

This gives us the familiar $\pi \approx 22/7$.
(a) Let's have `rat`, with a tolerance of `.02`, generate rational approximations to the powers of σ. We can compare the result to the rational approximations used in just intonation. In our code that compares equal temperament with just intonation, change the statements that define just intonation from

```
num = [...]
```

```
den = [...]
```

to

```
[num,den] = rat(sigma.^k,.02);
```

You should see that the best rational approximation agrees with the one used by just intonation for most of the notes. Only notes near the ends of the scale are different.

- 18/17 vs. 16/15 for `k = 1`
- 9/5 vs. 7/4 for `k = 10`
- 17/9 vs. 15/8 for `k = 11`

The approximations from `rat` are more accurate, but the denominators are primes or powers of primes and so are less likely to be compatible with other denominators.
(b) Change the tolerance involved in using `rat` to obtain rational approximations to powers of σ. Replace `.02` by `.01` or `.10`. You should find that `.02` works best.
(c) Modify `pianoex` to also use the rational approximations produced by `rat`. Can you detect any difference in the sound generated by `pianoex` if these `rat` approximations are incorporated.

20.8 *Musical score.* Our `pianoex` is able to process a MATLAB function that represents a musical score. The score is a cell array with two columns. The first column contains note numbers or chord vectors. The second column contains durations. If the second column is not present, all the notes have the same, default, duration. For example, here is a C-major scale.

```
s = {0 2 4 5 7 9 11 12}'
pianoex(s)
```

A more comprehensive example is the portion of Vivaldi's "Four Seasons" in the EXM function `vivaldi`.

```
type vivaldi
pianoex(vivaldi)
```

Frankly, I do not find this attempt to express music in a macine-readable form very satisfactory. Can you create something better?