Fundamental Frequency and Harmonics of a Violin Note

Introduction

This application finds the fundamental frequency and harmonics of a violin using information from the amplitude spectrum. Then, we generate a sinusoidal signal with the same frequency-amplitude characteristics of the violin note, and play the resulting sound.

This speaker component is needed to play audio: 

Import and Play the Audio

```
> restart:
  with(SignalProcessing):
  with(AudioTools):
  with(ColorTools):
  with(plots):
```

Import and play the violin note.

```
> violinNote := Read(FileTools:-JoinPath([kernelopts(datadir),
  "audio", "ViolinThreePosVibrato.wav"]))[ .. , 1];
  Play(violinNote);
```

$$violinNote := \begin{bmatrix} \text{"Sample Rate"} & 44100 \\ \text{"Bit Depth"} & 16 \\ \text{"Channels"} & 1 \\ \text{"Points/Channel"} & 64724 \\ \text{"Duration"} & 1.47\,s \end{bmatrix} \quad (2.1)$$

Sample rate and number of data points

```
> Fs := attributes(violinNote)[1];
  N  := numelems(violinNote);
```

$$Fs := 44100$$

$$N := 64724 \quad (2.2)$$

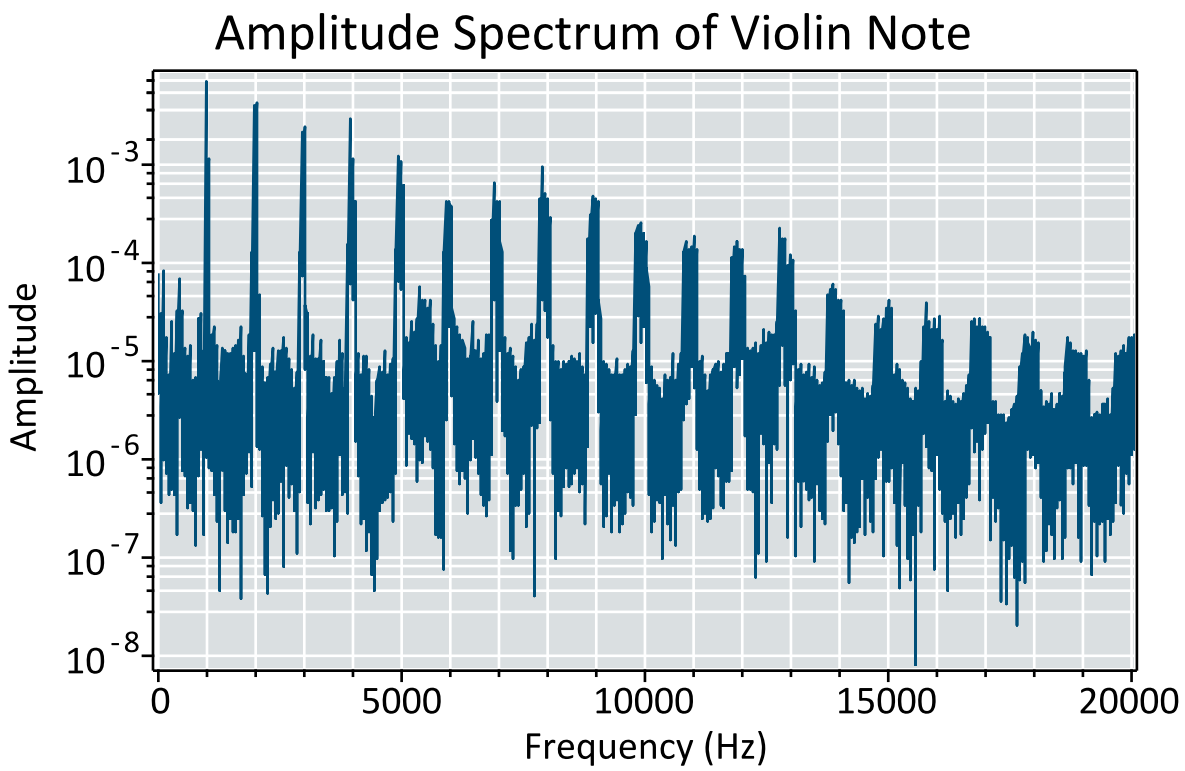
Amplitude Spectrum of the Violin Note

```
> sig_fft     := FFT(violinNote) / evalf(sqrt(N)):
  amplitudes  := Vector[column](2 * abs(sig_fft)):
  frequencies := Vector(N, i -> evalf(Fs * i / N), datatype =
  float[8]):
```

Warning, size of Array must be a power of two greater than two, using DFT instead and suppressing this warning for the remainder of this session

```
> p1 := plot(frequencies, amplitudes
  ,view       = [0..20000, 0..0.008]
  ,style      = line, symbol = solidcircle, color = Color("RGB",
  [0, 79/255, 121/255]), thickness = 0
  ,axes       = boxed
  ,size       = [800, 300]
  ,labels     = ["Frequency (Hz)", "Amplitude"], labelfont =
  [Calibri], labeldirections = [horizontal, vertical]
  ,axesfont   = [Calibri]
  ,title      = "Amplitude Spectrum of Violin Note", titlefont =
  [Calibri, 16]
  ,background = Color("RGB", [218/255, 223/255, 225/255])
  ,axis       = [gridlines = [color = Color("RGB", [1, 1, 1])]]):

  display(p1, axis[2] = [mode = log]);
```



Find the Local Peaks in the Amplitude Spectrum

Hence the peaks in the amplitude spectrum are as follows (the first column column contains the

frequencies, and the second column contains the amplitude).

```
> peakPoints := FindPeakPoints(<<frequencies | amplitudes>>,
  minimumheight = 0.0001, minimumbreadth = 75, includeendpoints =
  false);
```

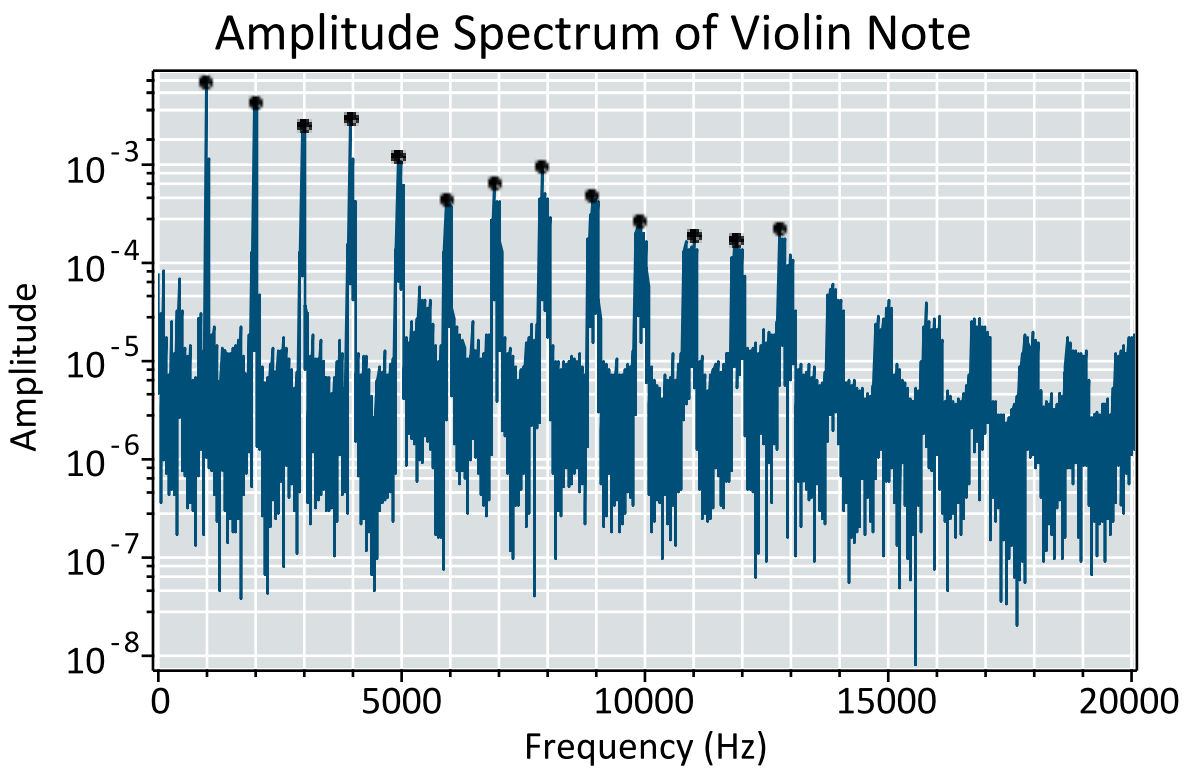
$$peakPoints := \begin{bmatrix} 981.150732300000 & 0.00688051457788536 \\ 2001.13868100000 & 0.00426776422024256 \\ 3000.68599000000 & 0.00248112917068676 \\ 3967.52827400000 & 0.00290492946148817 \\ 4933.00784900000 & 0.00119075323784304 \\ 5932.55515700000 & 0.000435123849101347 \\ 6919.83808200000 & 0.000641373293516093 \\ 7880.54817400000 & 0.000943193510053491 \\ 8907.34966900000 & 0.000483241692614761 \\ 9894.63259400000 & 0.000263417282367417 \\ \vdots & \vdots \end{bmatrix} \quad \text{(4.1)}$$

26 × 2 Matrix

Now overlay the peak points over the spectrum

```
> p2 := plot(peakPoints, style  = point, color  = black, symbol =
  solidcircle, view = [0 .. 20000, 0 .. 0.008], size = [800, 300])
  :

  display(p1, p2, axis[2] = [mode = log]);
```



Sonify a Signal the Same Peak Frequencies and Amplitude of the Violin Note

Now generate a sum of sinusoids from the frequency-amplitude data, and sonify the resulting signal.

```
> t := Vector(N, i -> (i - 1)/Fs, datatype = float[8]):
  aud_fh := (Create(Vector(N, i -> add(peakPoints[j,2] * sin(2 *
  Pi * peakPoints[j, 1] * t[i]),j = 1 .. 13), datatype = float[8]
  ), samplerate = 44100))
```

Warning, `j` is implicitly declared local to procedure

$$aud\_fh := \begin{bmatrix} \text{"Sample Rate"} & 44100 \\ \text{"Bit Depth"} & 16 \\ \text{"Channels"} & 1 \\ \text{"Points/Channel"} & 64724 \\ \text{"Duration"} & 1.47\,s \end{bmatrix} \quad \textbf{(5.1)}$$

```
> Play(Normalize(aud_fh))
```