Fourier Synthesis

Author: José Luis Gómez-Muñoz
https://www.facebook.com/matecmatica.cem
Based on work with the student Oscar Jauregui

Oboe Recording

Mathematica includes a very large collection of pre-recorded examples of audio:

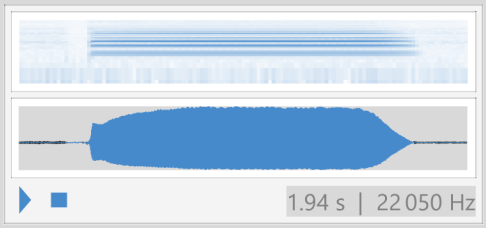
In[1]:=
```
ExampleData["Sound"]
```

Out[1]=
```
{{Sound, AltoFlute}, {Sound, AltoFluteScale}, {Sound, AltoSaxophone},
 {Sound, AltoSaxophoneScale}, {Sound, Apollo11PhoneCall},
 {Sound, Apollo11ReturnSafely}, {Sound, Apollo11SmallStep}, {Sound, Apollo13Countdown},
 {Sound, Apollo13Problem}, {Sound, BalloonPop}, {Sound, BassClarinet},
 {Sound, BassClarinetScale}, {Sound, BassFlute}, {Sound, BassFluteScale},
 {Sound, Bassoon}, {Sound, BassoonScale}, {Sound, BassTrombone},
 {Sound, BassTromboneScale}, {Sound, BlackcapWarbler}, {Sound, Burst100},
 {Sound, Burst1000}, {Sound, Burst7350}, {Sound, Cello}, {Sound, CelloPizzicato},
 {Sound, CelloPizzicatoScale}, {Sound, CelloScale}, {Sound, Clarinet},
 {Sound, ClarinetScale}, {Sound, CrashCymbal}, {Sound, DoubleBass},
 {Sound, DoubleBassPizzicato}, {Sound, DoubleBassPizzicatoScale},
 {Sound, DoubleBassScale}, {Sound, Flute}, {Sound, FluteScale}, {Sound, FrenchHorn},
 {Sound, FrenchHornScale}, {Sound, JetSound}, {Sound, LinearSweep},
 {Sound, NoiseBlue}, {Sound, NoiseBrown}, {Sound, NoisePink}, {Sound, NoiseViolet},
 {Sound, NoiseWhite}, {Sound, Oboe}, {Sound, OboeScale}, {Sound, OrganChord},
 {Sound, Piano}, {Sound, PianoScale}, {Sound, RollingCoin}, {Sound, SopranoSaxophone},
 {Sound, SopranoSaxophoneScale}, {Sound, Square10}, {Sound, Square100},
 {Sound, Square1000}, {Sound, Square7350}, {Sound, SubwayTrain}, {Sound, TenorTrombone},
 {Sound, TenorTromboneScale}, {Sound, TestIntermodulationDistortion},
 {Sound, Trumpet}, {Sound, TrumpetScale}, {Sound, Tuba}, {Sound, TubaScale},
 {Sound, Viola}, {Sound, ViolaScale}, {Sound, Violin}, {Sound, ViolinScale}}
```

Below we have an Oboe recording included in every computer that has *Mathematica*. In this document an approximation to this sound will be created by synthesis (addition) of sine waves. Below the sound is stored in the variable “recording” (if you are reading this document in *Mathematica* or the *CDFPlayer*, press the button  in the result of the calculation):

In[2]:=
```
recording = ExampleData[{"Sound", "Oboe"}]
```

Out[2]=

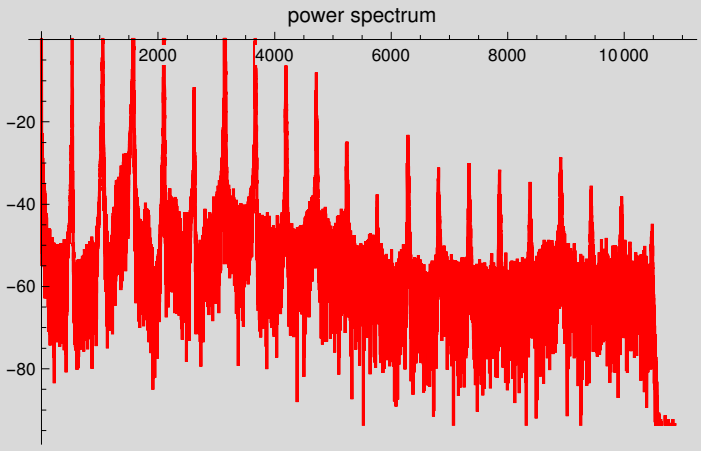


Spectra of the Original Oboe Recording

Below the Periodogram[] *Mathematica* command is used to obtain the POWER spectra for our recording, with a logarithmic scale in the vertical axis:

In[3]:=
```
Periodogram[recording,
 PlotStyle → Red,
 PlotLabel → "power spectrum"
]
```

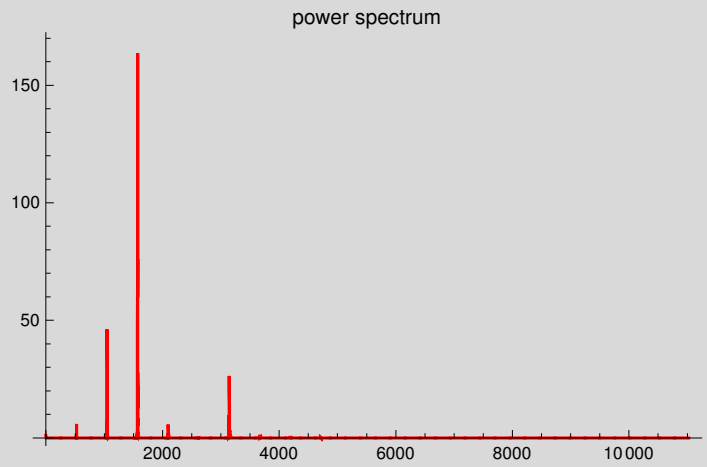
Out[3]=



The option **ScalingFunctions→"Absolute"** gives a linear scale in the vertical axis, and with the option **PlotRange→All** the range will include all data points:

In[4]:=
```
Periodogram[recording,
 PlotRange → All,
 ScalingFunctions → "Absolute",
 PlotStyle → Red,
 PlotLabel → "power spectrum"]
```

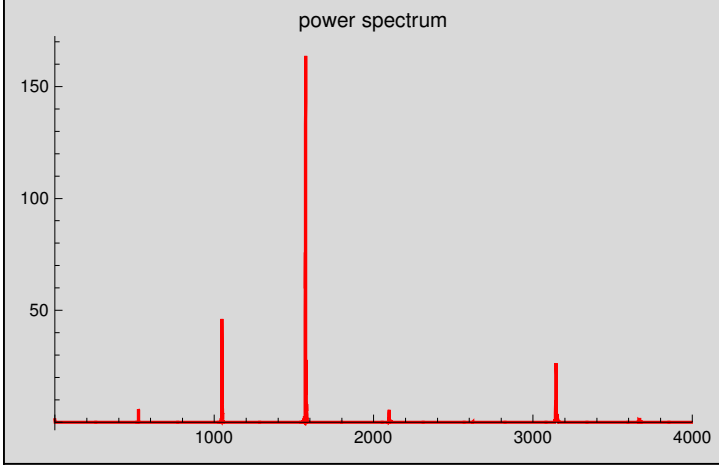
Out[4]=



The previous graph of the spectra has large peaks between 0 and 4000Hz, therefore the PlotRange is changed to **PlotRange→{{0,4000},All}** in order to see those peaks. The graph is stored in the variable **spectra**, in order to use it later in this document:

In[5]:=
```
spectra = Periodogram[recording,
  PlotRange → {{0, 4000}, All},
  ScalingFunctions → "Absolute",
  PlotStyle → Red,
  PlotLabel → "power spectrum"]
```

Out[5]=

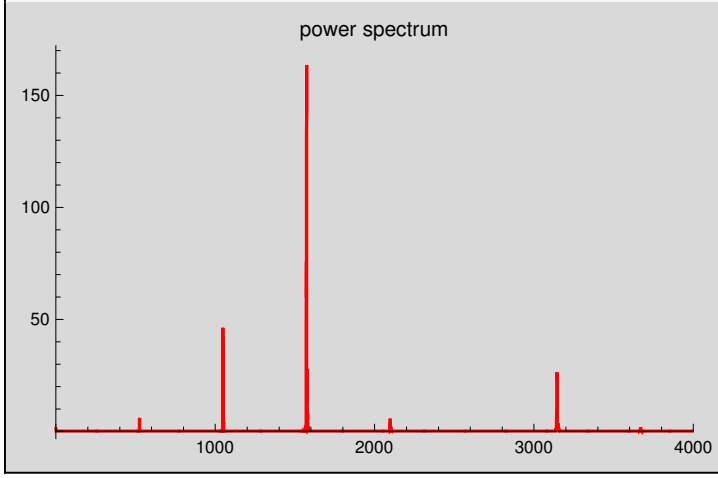


First Peak in Synthetized Spectra

These are the main peaks in our power spectrum:

In[6]:=
```
spectra
```

Out[6]=



We further zoom-in the **first peak** by changing **PlotRange→{{500,600},**All**}**, as shown below:

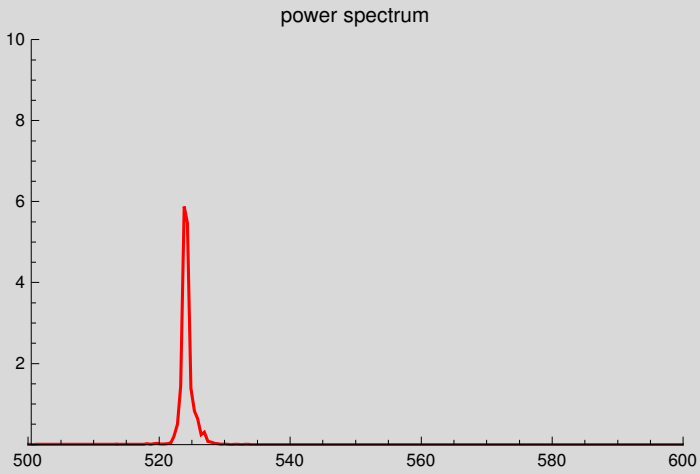
In[7]:=
```
Periodogram[recording,
 PlotRange → {{500, 600}, All},
 ScalingFunctions → "Absolute",
 PlotStyle → Red,
 PlotLabel → "power spectrum"]
```

Out[7]=

Now we zoom in the vertical axis to see the height of the tiny peak,
PlotRange→{{500,600}**,{0,10}**}

In[8]:=
```
Periodogram[recording,
 PlotRange → {{500, 600}, {0, 10}},
 ScalingFunctions → "Absolute",
 PlotStyle → Red,
 PlotLabel → "power spectrum"]
```

Out[8]=



From the graph above we can see that we have 6.0 arbitray power units at 524 Hertz. We create a synthesized power spectra (a list of frequency-power pairs), which right now only has that first peak:

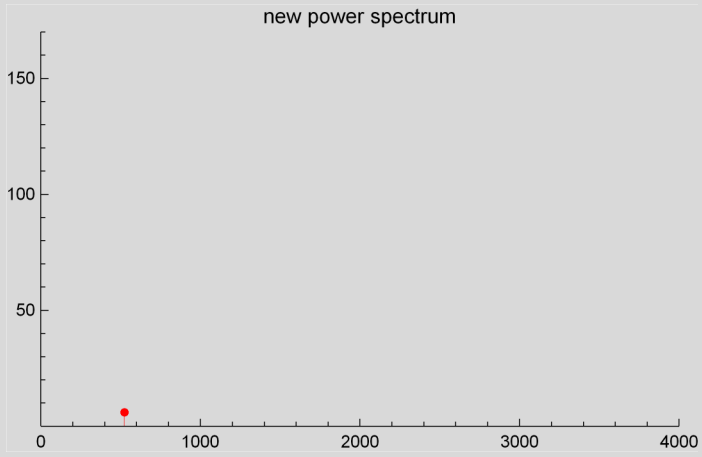
In[9]:=
```
synthet = {};
AppendTo[synthet, {524, 6.0}]
```

Out[10]=
```
{{524, 6.}}
```

We will continue adding frequency-power pairs to the list, and this list will be used at the end of this document in order to mathematically generate a sound similar to original recording. In this moment we only have one data point in our new power spectrum:

In[11]:=
```
newspectra = ListPlot[synthet,
  PlotRange → {{0, 4000}, {0, 170}},
  Filling → Axis,
  PlotStyle -> Red,
  PlotLabel -> "new power spectrum"]
```

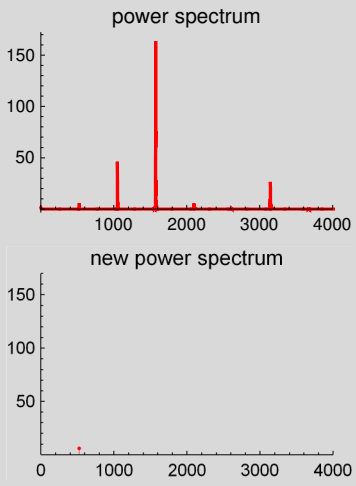
Out[11]=



Below we compare the spectra of our recording, with our synthetized spectra so far. In this moment we only have one data point in our new power spectrum:

In[12]:=
```
Column[{ spectra, newspectra }]
```

Out[12]=

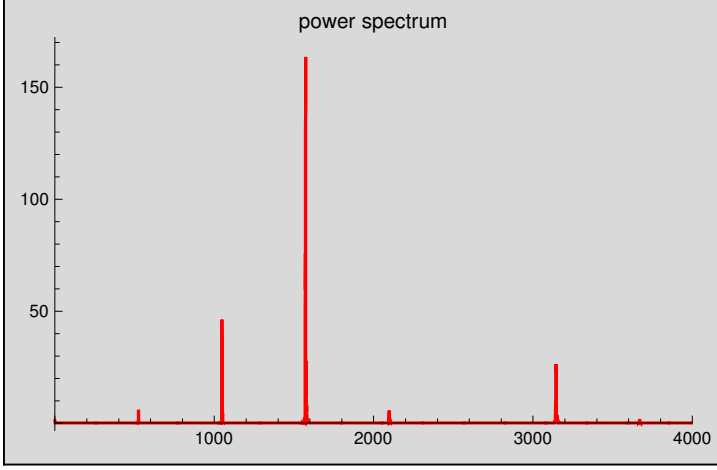


Second Peak in Synthetized Spectra

We will work now with the second peak shown below:

In[13]:=
```
spectra
```

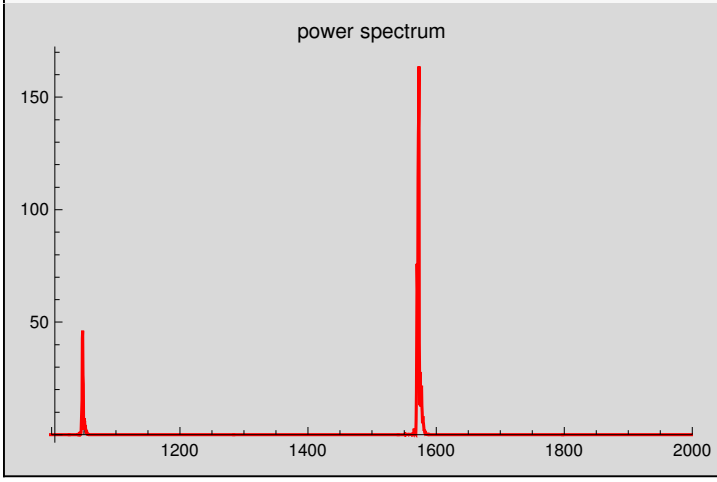
Out[13]=



The previous graph of the spectra has a second peak between 1000 and 2000Hz, therefore the PlotRange is changed to **PlotRange→{{1000,2000},**All**}** in order to see this peak, as shown below:

In[14]:=
```
Periodogram[recording,
 PlotRange → {{1000, 2000}, All},
 ScalingFunctions → "Absolute",
 PlotStyle → Red,
 PlotLabel → "power spectrum"]
```

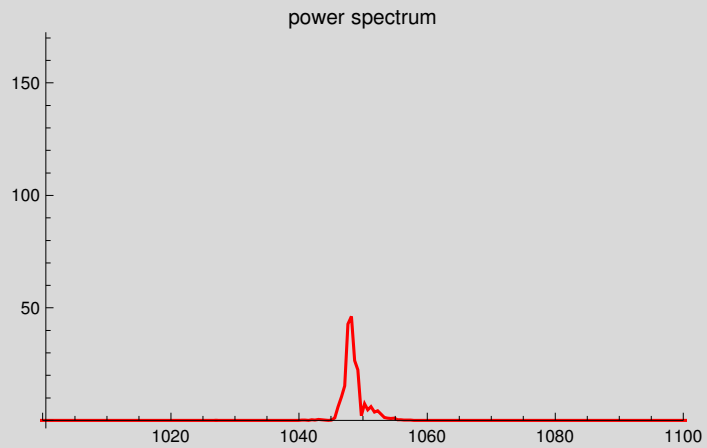
Out[14]=



We further zoom-in the second peak by changing **PlotRange→{{1000,1100},**All**}**, as shown below:

In[15]:=
```
Periodogram[recording,
 PlotRange → {{1000, 1100}, All},
 ScalingFunctions → "Absolute",
 PlotStyle → Red,
 PlotLabel → "power spectrum"]
```

Out[15]=



Recall our new synthetized spectra (list of frequency-amplitude pairs) so far:

In[16]:=
```
synthet
```

Out[16]=
```
{{524, 6.}}
```

The second peak is clearly located at the double of frequency of the first one:

In[17]:=
```
2 * 524
```

Out[17]=
```
1048
```

And we can see from the previous graph that it has a power of 50 arbitrary units, therefore we add it to our synthetized spectra, as shown below:

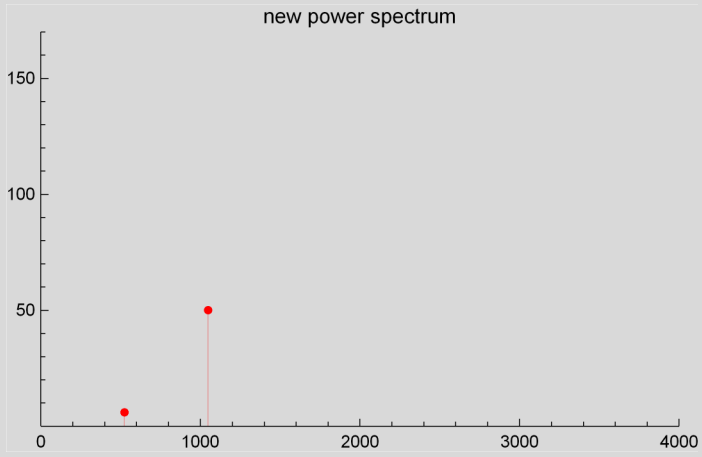
In[18]:=
```
AppendTo[synthet, {1048, 50.}]
```

Out[18]=
```
{{524, 6.}, {1048, 50.}}
```

We update the graph of our new power spectrum:

In[19]:=
```
newspectra = ListPlot[synthet,
  PlotRange → {{0, 4000}, {0, 170}},
  Filling → Axis,
  PlotStyle -> Red,
  PlotLabel -> "new power spectrum"]
```

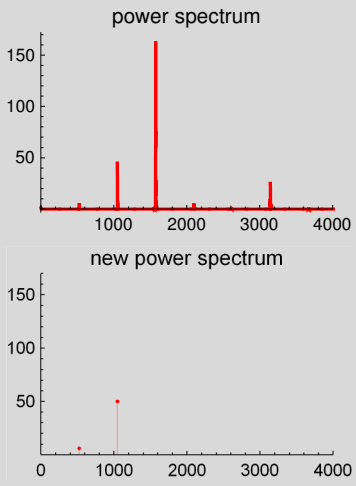
Out[19]=



Below we compare the spectra of our recording, with our synthetized spectra so far. In this moment we only have two data points in our new power spectrum:

In[20]:=
```
Column[{ spectra, newspectra }]
```

Out[20]=

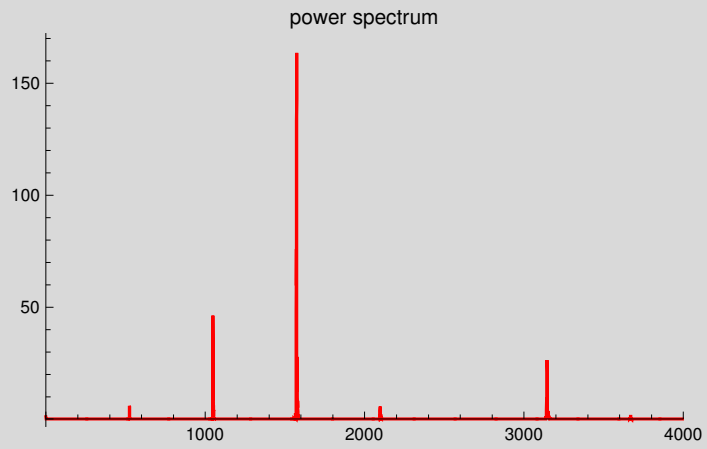


Third Peak in Synthetized Spectra

We will work now with the third peak shown below:

In[21]:=
```
spectra
```

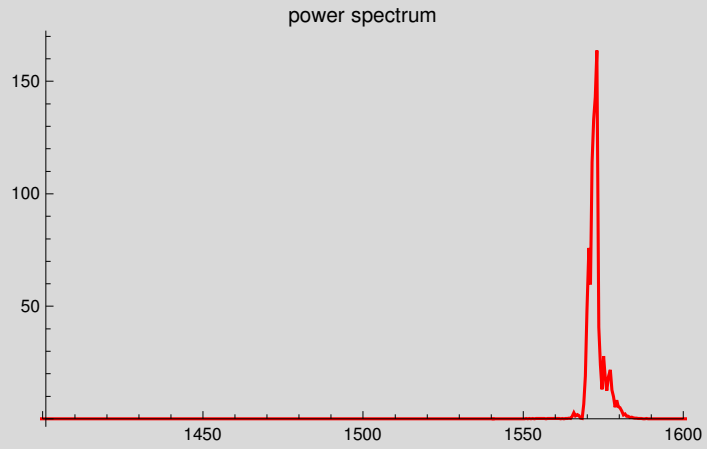
Out[21]=



We further zoom-in the third peak by changing **PlotRange→{{1400,1600},**All**}**, as shown below:

In[22]:=
```
Periodogram[recording,
 PlotRange → {{1400, 1600}, All},
 ScalingFunctions → "Absolute",
 PlotStyle → Red,
 PlotLabel → "power spectrum"]
```

Out[22]=



Recall our new synthetized spectra so far:

In[23]:=
```
synthet
```

Out[23]=
```
{{524, 6.}, {1048, 50.}}
```

The third peak is clearly located at the triple of frequency of the first one:

In[24]:=
```
3 * 524
```

Out[24]=
```
1572
```

And we can see from the previous graph it has a power of 165. arbitrary units, therefore we add it to our synthetized spectra, as shown below:

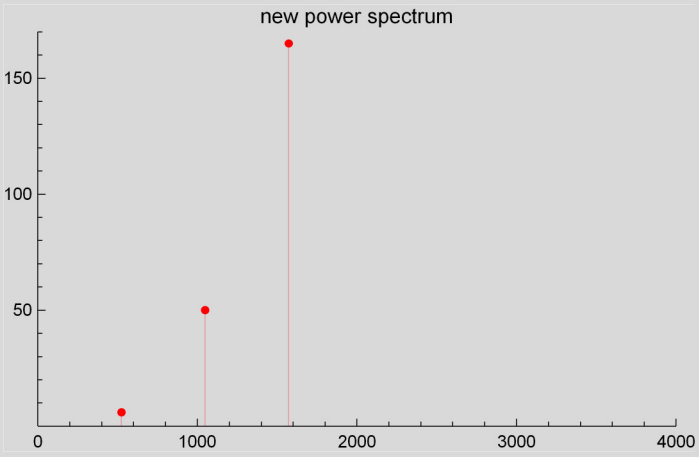
In[25]:=
```
AppendTo[synthet, {1572, 165.}]
```

Out[25]=
```
{{524, 6.}, {1048, 50.}, {1572, 165.}}
```

We update the graph of our new power spectrum:

In[26]:=
```
newspectra = ListPlot[synthet,
  PlotRange → {{0, 4000}, {0, 170}},
  Filling → Axis,
  PlotStyle -> Red,
  PlotLabel -> "new power spectrum"]
```

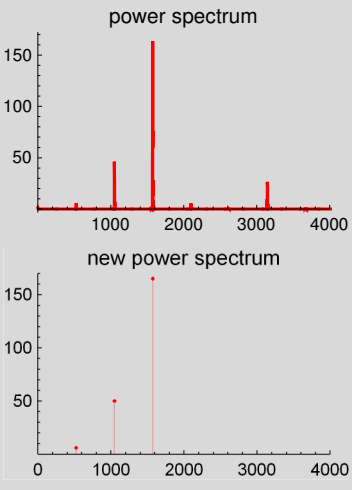
Out[26]=



Below we compare the spectra of our recording, with our synthetized spectra so far. In this moment we have three data points in our new power spectrum:

In[27]:=
```
Column[{ spectra, newspectra }]
```

Out[27]=



Other Peaks

Just like we did with the first, second and third peaks, we can repeat the procedure for the other visible peaks. Below we just add them to our synthetized spectra:

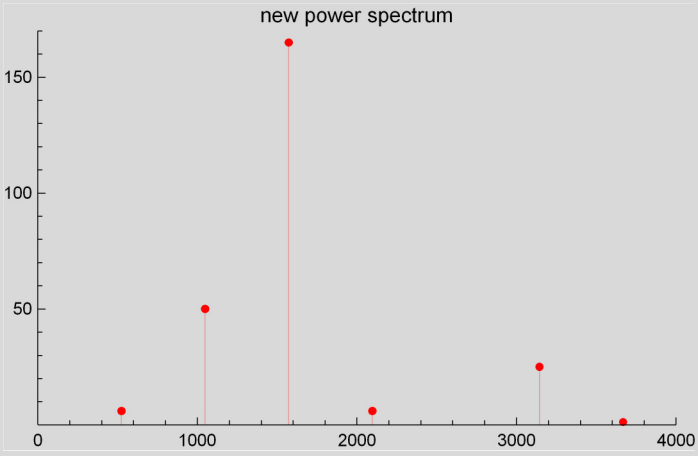
In[28]:=
```
AppendTo[synthet, {2096, 6.}];
AppendTo[synthet, {3144, 25.}];
AppendTo[synthet, {3668, 1.2}]
```

Out[30]=
```
{{524, 6.}, {1048, 50.}, {1572, 165.}, {2096, 6.}, {3144, 25.}, {3668, 1.2}}
```

We update the graph of our new power spectrum:

In[31]:=
```
newspectra = ListPlot[synthet,
  PlotRange → {{0, 4000}, {0, 170}},
  Filling → Axis,
  PlotStyle -> Red,
  PlotLabel -> "new power spectrum"]
```

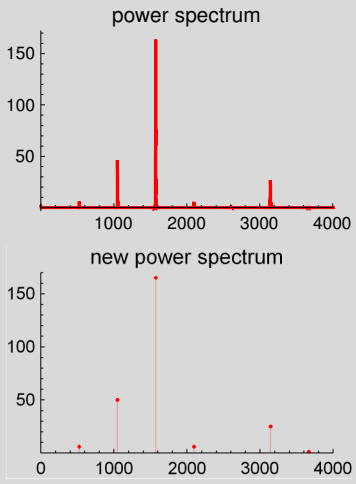
Out[31]=



Below we compare the spectra of our recording, with our synthetized spectra:

In[32]:=
```
Column[{ spectra, newspectra }]
```

Out[32]=



Create the Synthetized Waveform

The commands below use the frequencies and amplitudes (square root of the power values) in our synthetized spectra (the list in the variable “synthet”) in order to generate the mathematical expression for the waveform, and store it in the variable “wavef”:

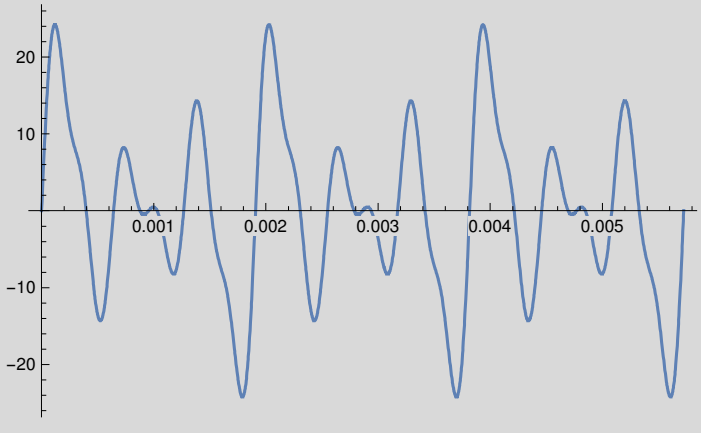
In[33]:=
```
wavef = Sum[√synthet[[j, 2]] * Sin[2 * Pi * synthet[[j, 1]] * t], {j, 1, Length[synthet]}]
```

Out[33]=
```
2.44949 Sin[1048 π t] + 7.07107 Sin[2096 π t] + 12.8452 Sin[3144 π t] +
 2.44949 Sin[4192 π t] + 5. Sin[6288 π t] + 1.09545 Sin[7336 π t]
```

We can visualize three periods of our waveform:

In[34]:=
```
Plot[wavef, {t, 0, 3/524}]
```

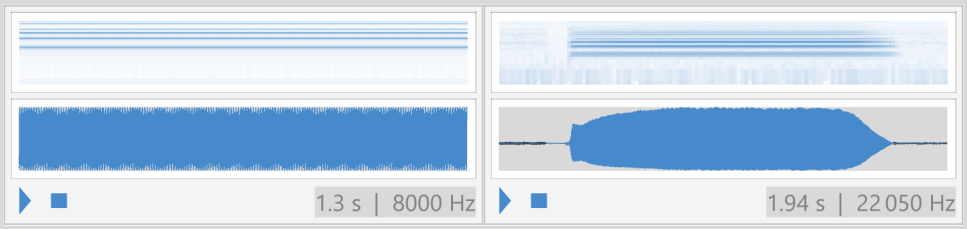
Out[34]=



And we can listen our synthetized waveform, and compare with the original recording. Do they sound similar to you?

In[35]:=
```
Row[{
  Play[wavef, {t, 0, 1.3}],
  ExampleData[{"Sound", "Oboe"}]
 }]
```

Out[35]=



EXERCISE 1

Select, from the list below, one of the recordings of one note of a **wind** or **string** instrument that are included in *Mathematica*, and obtain a synthetized waveform (sum of sine functions) that sounds similar to that recording:

In[36]:=
```
ExampleData["Sound"]
```

Out[36]=
```
{{Sound, AltoFlute}, {Sound, AltoFluteScale}, {Sound, AltoSaxophone},
 {Sound, AltoSaxophoneScale}, {Sound, Apollo11PhoneCall},
 {Sound, Apollo11ReturnSafely}, {Sound, Apollo11SmallStep}, {Sound, Apollo13Countdown},
 {Sound, Apollo13Problem}, {Sound, BalloonPop}, {Sound, BassClarinet},
 {Sound, BassClarinetScale}, {Sound, BassFlute}, {Sound, BassFluteScale},
 {Sound, Bassoon}, {Sound, BassoonScale}, {Sound, BassTrombone},
 {Sound, BassTromboneScale}, {Sound, BlackcapWarbler}, {Sound, Burst100},
 {Sound, Burst1000}, {Sound, Burst7350}, {Sound, Cello}, {Sound, CelloPizzicato},
 {Sound, CelloPizzicatoScale}, {Sound, CelloScale}, {Sound, Clarinet},
 {Sound, ClarinetScale}, {Sound, CrashCymbal}, {Sound, DoubleBass},
 {Sound, DoubleBassPizzicato}, {Sound, DoubleBassPizzicatoScale},
 {Sound, DoubleBassScale}, {Sound, Flute}, {Sound, FluteScale}, {Sound, FrenchHorn},
 {Sound, FrenchHornScale}, {Sound, JetSound}, {Sound, LinearSweep},
 {Sound, NoiseBlue}, {Sound, NoiseBrown}, {Sound, NoisePink}, {Sound, NoiseViolet},
 {Sound, NoiseWhite}, {Sound, Oboe}, {Sound, OboeScale}, {Sound, OrganChord},
 {Sound, Piano}, {Sound, PianoScale}, {Sound, RollingCoin}, {Sound, SopranoSaxophone},
 {Sound, SopranoSaxophoneScale}, {Sound, Square10}, {Sound, Square100},
 {Sound, Square1000}, {Sound, Square7350}, {Sound, SubwayTrain}, {Sound, TenorTrombone},
 {Sound, TenorTromboneScale}, {Sound, TestIntermodulationDistortion},
 {Sound, Trumpet}, {Sound, TrumpetScale}, {Sound, Tuba}, {Sound, TubaScale},
 {Sound, Viola}, {Sound, ViolaScale}, {Sound, Violin}, {Sound, ViolinScale}}
```

EXERCISE 2

Record your voice saying “ah” during 3 seconds and store the recording in your computer, as a WAV file, in the directory you get as a result of evaluating the command Directory[] in your computer:

In[37]:=
```
Directory[]
```

Out[37]=
```
C:\Users\L00698076\Google Drive\SetDirectory
```

You can use the command FileNames[] in order to verify that your WAV file is there:

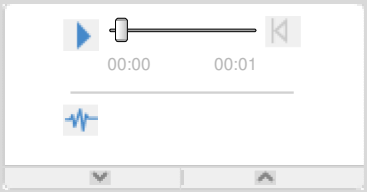
In[38]:=
```
FileNames["*.wav"]
```

Out[38]=
```
{ahrecording.wav, ejemplo.wav, equaltemperament.wav, gtrstring.wav,
 handmade-reed.wav, mynewsong.wav, mysong.wav, mys.wav, notaejemplo.wav,
 pythagorean.wav, resonator-ah.wav, resonator-ih.wav, resonator-oo.wav}
```

Next read the recording into Mathematica with the command Import[], using the name of your own file:

In[39]:=
```
myvoice = Import["ahrecording.wav"]
```

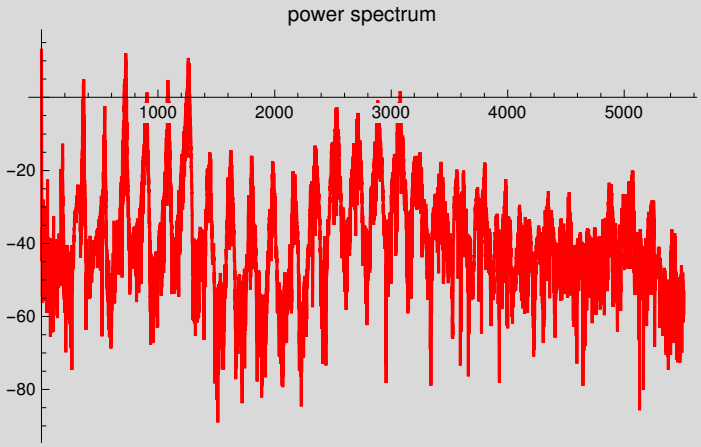
Out[39]=



Then use Periodogram[] to obtain the power spectrum of your voice saying “ah”

In[40]:=
```
Periodogram[myvoice,
 PlotStyle → Red,
 PlotLabel → "power spectrum"
]
```

Out[40]=



Create a synthetized waveform (sum of sine functions) that sounds similar to your recording

---

Author: José Luis Gómez-Muñoz
https://www.facebook.com/matecmatica.cem
Based on work with the student Oscar Jauregui

In[41]:=
```
{DateString[], $Version}
```

Out[41]=
```
{Fri 29 Sep 2017 13:03:45, 11.0.0 for Microsoft Windows (64-bit) (July 28, 2016)}
```