Zero Padding a Signal to More Accurately Estimate Amplitudes from a DFT

Introduction

You can use a discrete Fourier transform (DFT) to identify the amplitude of a sinusoidal signal. Given a signal with N samples and a sample rate of Fs, the frequency spacing of its DFT is Fs/N. An amplitude at a frequency that falls in a DFT bin can be accurately estimated.

But what if a signal frequency falls between DFT bins? Its energy will be shared between the adjacent bins, and its amplitude will not be accurately identified.

One possible solution to better amplitude estimation is *zero padding* the time domain signal. If you zero-pad the signal, the value of N increases, and the frequency spacing of the DFT decreases. For example, if you double the time-domain signal with zero-padding, the frequency spacing decreases by a factor of two.

This is not a magical way of increasing the sampling rate, or injecting more data. Zero-padding in the time domain is simply equivalent to sinc interpolation of the DFT.

In this application, we will

- generate a sinusoidal signal with three different frequencies at three different amplitudes (one frequency will fall in a DFT bin, but the other two won't)
- use a DFT to estimate the signal amplitudes
- zero-pad the original signal
- use a DFT of the zero-padded signal to again estimate the signal amplitudes

You will see that DFT of the zero-padded signal more accurately estimates the amplitudes of the signal.

```
> restart:
  with(SignalProcessing):
```

Define the Signal

Number of points and sample rate

```
> N  := 2000:
```

```
Fs := 2000:
```

Frequencies and amplitudes (we'll try to estimate these with a DFT).

```
> freq_1 := 200.5:
  freq_2 := 450:
  freq_3 := 800.5:

  A_1 := 1:
  A_2 := 4:
  A_3 := 2.5:
```

Time and signal vectors

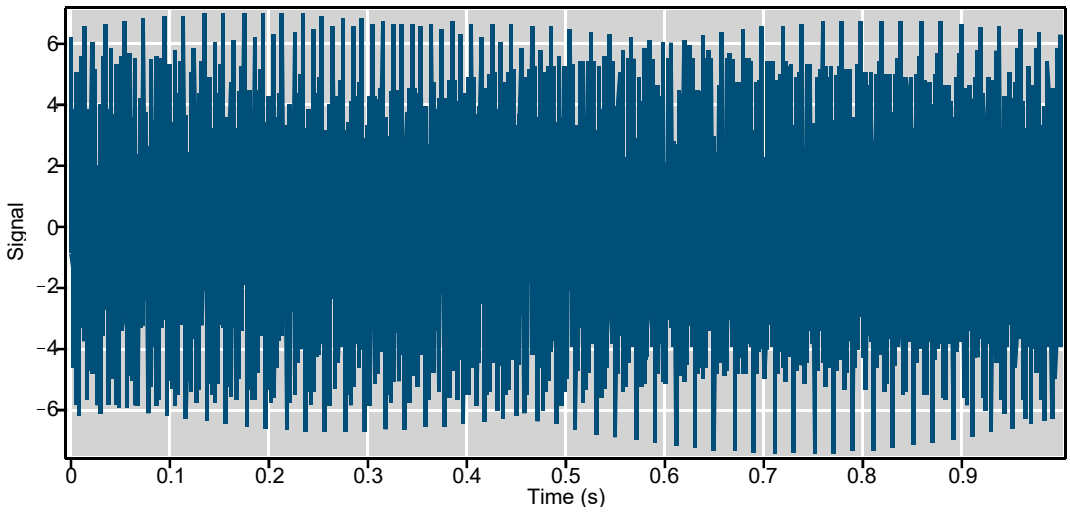
```
> t   := Vector(N, i -> (i - 1.) / Fs, datatype = float[8]):
  sig := Vector(N, i -> A_1 * cos(2 * Pi * freq_1 * t[i]) + A_2 *
  sin(2 * Pi * freq_2 * t[i]) + A_3 * sin(2 * Pi * freq_3 * t[i]),
  datatype = float[8]);
```

$$sig := \begin{bmatrix} 1. \\ 6.22513038439047 \\ -0.833106929112445 \\ -1.49626599384041 \\ -4.64597722445230 \\ 1.84809272604146 \\ 3.88296270704509 \\ -4.48350629688604 \\ -1.09612108910748 \\ -0.0549250049063967 \\ \vdots \end{bmatrix} \quad \text{(2.1)}$$

2000 element Vector[column]

Plot of signal

```
> plot(t, sig, size = [800,400],background = "LightGrey", axis =
  [gridlines = [10, color = white]], titlefont = [Arial, 16],
  axesfont = [Arial], labelfont = [Arial, 12], size = [800, 400],
  axes = boxed, color = ColorTools:-Color("RGB",[0, 79/255, 121/255]
  ), labels = ["Time (s)", "Signal"], labeldirections = [horizontal,
  vertical],  color = ColorTools:-Color("RGB",[0, 79/255, 121/255]))
```



Identify Amplitude of Signal with a Discrete Fourier Transform

```
> sig_fft := FFT(sig) / evalf(sqrt(N)):
```

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

```
> sig_fft := sig_fft[1 .. N / 2 + 1]:
```

Multiply all frequencies apart from DC and Nyquist by 2

```
> sig_fft[2..-2] := sig_fft[2 .. -2] * 2:
> power := abs(sig_fft):
```

These are the frequencies represented by the DFT bins

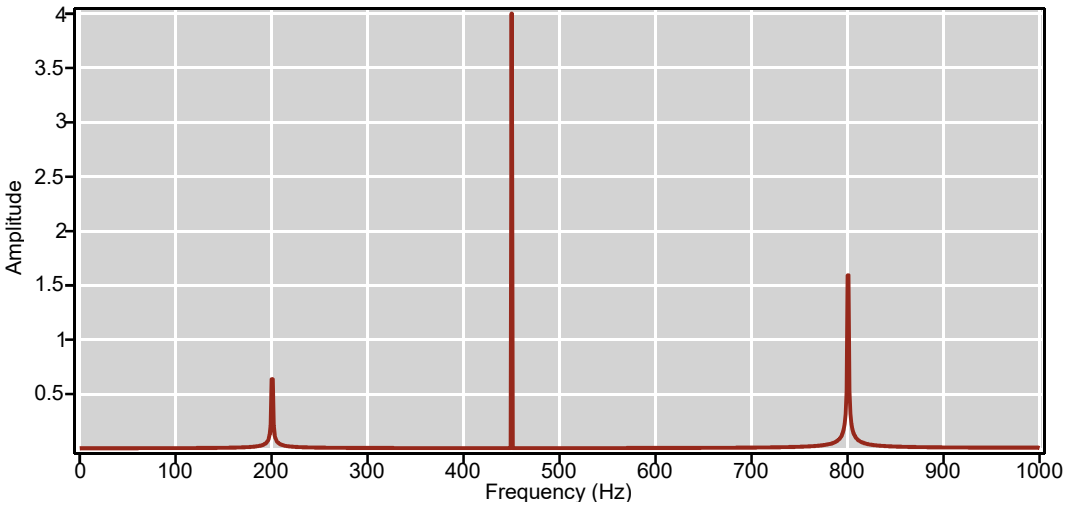
```
> freqs := Vector(N / 2 + 1, i -> evalf(Fs * (i - 1) / N), datatype
  = float[8]);
```

$$freqs := \begin{bmatrix} 0. \\ 1. \\ 2. \\ 3. \\ 4. \\ 5. \\ 6. \\ 7. \\ 8. \\ 9. \\ \vdots \end{bmatrix} \quad \textbf{(3.1)}$$

1001 element Vector[column]

```
> p1 := plot(freqs, power, size = [800,400],background =
  "LightGrey", axis = [gridlines = [10, color = white]], titlefont =
```

```
[Arial, 16], axesfont = [Arial], labelfont = [Arial, 12], size =
[800, 400], axes = boxed, color = ColorTools:-Color("RGB",[0,
79/255, 121/255]), labels = ["Frequency (Hz)", "Amplitude"],
labeldirections = [horizontal, vertical], color = ColorTools:-
Color("RGB",[150/255, 40/255, 27/255]))
```



The frequencies and corresponding amplitudes are

```
> peakPoints := FindPeakPoints(<<freqs | Vector[column](power)>>);
```

$$peakPoints := \begin{bmatrix} 201. & 0.637307319876280 \\ 450. & 4.00151644614509 \\ 801. & 1.59227850772051 \end{bmatrix} \quad \textbf{(3.2)}$$

We have correctly identified the frequency at 450 Hz with an amplitude of 8. However, the other two frequencies and amplitudes are not accurately identified because their frequencies are halfway pewter adjacent DFT bins.

Identify Amplitude of Zero-padded Signal with a Discrete Fourier Transform

Pad the signal with zeroes to double its length

```
> sig_padded := sig:
  N_padded := 2 * N:
  sig_padded( N_padded ) := 0:
```

Now convert the padded signal to the frequency domain

```
> sig_padded_fft := FFT(sig_padded) / evalf(sqrt(N_padded)):
> sig_padded_fft := sig_padded_fft[1 .. N_padded / 2 + 1]:
```

Multiply all frequencies apart from DC and Nyquist by 2

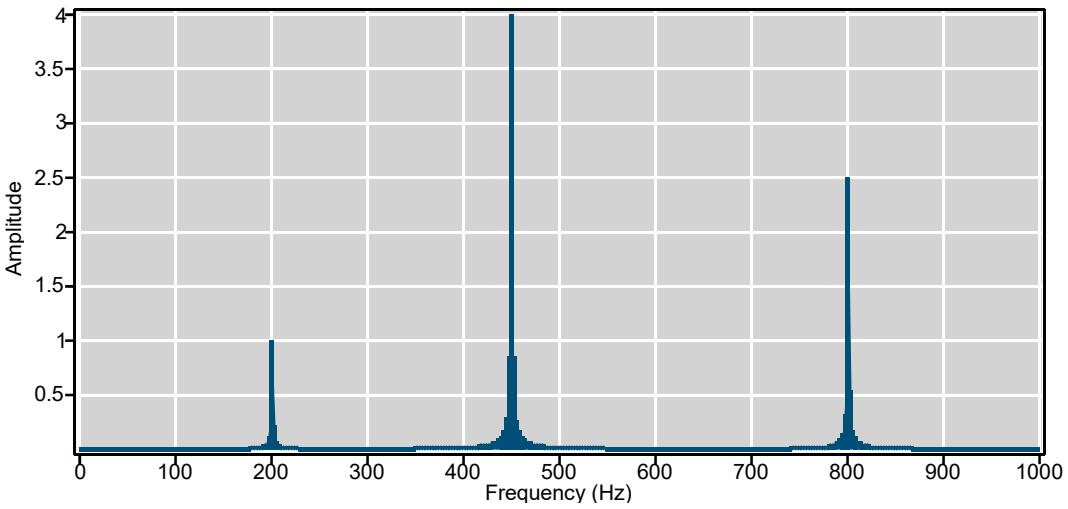
```
> sig_padded_fft[2..-2] := sig_padded_fft[2 .. -2] * 2:
> power_padded  := 2 * abs(sig_padded_fft):
```

These are the updated frequencies represented by the DFT bins.

```
> freqs_padded  := Vector(N_padded / 2 + 1, i -> evalf(Fs * (i - 1 )
  / N_padded), datatype = float[8])
```

$$freqs\_padded := \begin{bmatrix} 0. \\ 0.500000000000000 \\ 1. \\ 1.50000000000000 \\ 2. \\ 2.50000000000000 \\ 3. \\ 3.50000000000000 \\ 4. \\ 4.50000000000000 \\ \vdots \end{bmatrix} \quad \text{2001 element Vector[column]} \tag{4.1}$$

```
> p2:= plot(freqs_padded, power_padded, size = [800,400],background
  = "LightGrey", axis = [gridlines = [10, color = white]], titlefont
  = [Arial, 16], axesfont = [Arial], labelfont = [Arial, 12], size =
  [800, 400], axes = boxed, color = ColorTools:-Color("RGB",[0,
  79/255, 121/255]), labels = ["Frequency (Hz)", "Amplitude"],
  labeldirections = [horizontal, vertical])
```



The frequencies and corresponding amplitudes are

```
> peakPoints := FindPeakPoints(<<freqs_padded | Vector[column]
  (power_padded)>>, minimumbreadth = 1);
```

$$peakPoints := \begin{bmatrix} 200.500000000000 & 1.00606261543644 \\ 450. & 4.00151644503689 \\ 800.500000000000 & 2.50000334225408 \end{bmatrix} \tag{4.2}$$

These better match the frequencies and amplitudes used to generate the signal.

Let's overlay the amplified periodogram for the original and zero-padding signal, and zoom in on a point.

```
> plots:-display(p1,p2,  view = [195..205, 0..1], legend = ["DFT of
  Original Signal", "DFT of Zero-padded Signal"], legendstyle =
  [font = [Arial]])
```