Fourier Transforms: A Crash Course

Nikolaus Prusinski

July 17, 2019

1 Introduction

The Fourier Transform is a convenient tool that returns the component frequencies which make up a particular function. A common analogy is to think of a baked cake; the Fourier transform returns the constituent ingredients and their respective proportions.

2 Math Background

2.1 Definition

The Fourier Transform is defined for an integrable function $f : \mathbb{R} \to \mathbb{C}$ as:

$$F(y) = \int_{-\infty}^{\infty} f(x)e^{2\pi ixy}dx$$

for any $y \in \mathbb{R}$. Similarly, the inverse Fourier transform is defined as:

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(y)e^{-2\pi ixy}dy$$

for any $x \in \mathbb{R}$.

For our purposes, we are transforming a time domain of a gravitational wave signal to the corresponding frequency domain. Thus, $x = t$ and $y = f = \frac{\omega}{2\pi}$ in the above integrals.

2.2 Fourier Transforms of Common Functions

Below, we provide a table of common functions and their respective Fourier Transforms.

	$g(t)$	$G(f)$
Gaussian ($\mu = 0$)	$\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-t^2}{2\sigma^2})$	$\exp(-2\sigma^2\pi^2f^2)$
Gaussian ($\mu \neq 0$)	$\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-(t-\mu)^2}{2\sigma^2})$	$\exp(-2\sigma^2\pi^2f^2)\cos(2\pi f\mu)$
Sine Curve	$\sin(2\pi f_0 t)$	$\frac{1}{2i}[\delta(f_0+f)-\delta(f-f_0)]$
Sine-Gaussian	$\frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-t^2}{2\sigma^2})\sin(2\pi f_0 t)$	See Below
Exponential Decay	$\exp(-\Gamma t)$	$\frac{1}{\pi}\frac{\Gamma}{(2\pi f)^2+\Gamma^2}$

It is easy to show that the Gaussian satisfies Parseval's Theorem, i.e.

$$\int_{-\infty}^{\infty} |g(t)|^2 \, \mathrm{d}t = \frac{1}{2\pi} \int_{-\infty}^{\infty} |G(\omega)|^2 \, \mathrm{d}\omega = \int_{-\infty}^{\infty} |G(2\pi f)|^2 \, \mathrm{d}f,$$

where $g(t)$ is the Gaussian and $G(\omega)$ and $G(2\pi f)$ are the Fourier transforms of $g(t)$.

The Fourier transform of a Gaussian returns another unnormalized Gaussian. The two curve widths are related by $\sigma_\omega = 1/\sigma_t$, where σ_ω is the width of the Gaussian in the frequency domain and σ_t is the width in the time domain. If the Gaussian has a nonzero mean, a sine-Gaussian is returned in the transform.

For the exponential decay, note that the Fourier transform results in a Lorentzian (Cauchy) distribution. Also, if the bounds on the Fourier integral of the sine curve are not infinite, we encounter the sinc function, which in the limit, tends to the delta function. This is also the case with the underdamped harmonic oscillator.

3 Physics

The physical application of this technique is in the signal analysis of gravitational waves. The gravitational wave detectors measure the strain, an effect caused by the distortion of spacetime as the wave passes. The strain, $h(t)$, is comparable to the amplitude of the wave, from a signal analysis perspective. To find the energy of the wave we look to the amplitude and power spectral densities.

The discrete Fourier transform (DFT) is scaled by $1/\sqrt{T}$ resulting in an amplitude spectral density, $h(\omega)$, with units of $\frac{1}{\sqrt{\text{Hz}}}$. The inverse DFT is normalized by $\sqrt{T}$. For the power spectral density, $(h(\omega))^2$, forward and inverse normalization would look like $1/T$ and T respectively.

The gravitational wave is detectable for a finite time T in our detector. Thus the integration time is not infinite, but is bounded by T. The wave passing our detector is assumed to be a sine-Gaussian. The final Fourier integral for ASD looks like:

$$G(f) = \left| \frac{1}{\sqrt{T}} \int_{-T}^{T} \frac{A}{\sqrt{2\pi}\sigma} \exp\left(\frac{-t^2}{2\sigma^2}\right) \sin(2\pi f_0 t) \exp(2\pi i f t) \, dt \right|,$$

where σ and f_0 take on their traditional meanings of width and frequency of the wave packet. A is a constant.

4 Coding

4.1 Mathematica

Inputting the Fourier Transform into Mathematica is a relatively straightforward procedure. We begin by defining the function and the Fourier integral. From there, we assign numeric values

to each of the constants (stored in variables like Tnum, Anum, etc.). Using Evaluate to speed up the computations and using ReplaceAll for the variables, we plot both the function and its transform. Mathematica automatically chooses the number of points to plot, although this can be manually specified using PlotPoints. Below is a list of definitions of each of the variables for Mathematica.

T is the integration time, so T is a constant (say 1s) and we take the Fourier Transform over this window (-1s to 1s). We determine T, however, we want the range to encompass the wavepacket or a few periods of the sine curve for the FT to act on.

A is the amplitude of the wave (dimensionless). This is an arbitrary scaling factor which we can multiply the wave by. For a sine wave, it gives us the height of the peak.

σ is related to the width of the distribution and also to the FWHM (full width half max). σ is a constant and a particular Gaussian or sine-Gaussian will have a particular value of σ.

f_c is the frequency of the sine wave. This gives us the number of cycles per second and translates the peak of the Fourier transform in the frequency domain.

In the following pages, there are Mathematica notebooks which calculate the finite Fourier transforms of a sine curve, a Gaussian, Lorentzian, and a sine-Gaussian. Lastly, there is a recreation of the McNeill plot for a gravitational wave with memory.

Fourier Transform of sine wave

Define Functions

In[1]:=
```
sine = A Sin[2 π f_c t]
```

Out[1]=
$$A \sin[2 \pi t f_c]$$

In[2]:=
```
FT = Abs[1/T ∫_-T^T (sine e^(2 π f i t)) dt]
```

Out[2]=
$$\frac{\text{Abs}\left[\frac{A\,(f \cos[2 f \pi T] \sin[2 \pi T f_c] - \cos[2 \pi T f_c] \sin[2 f \pi T] f_c)}{T\,(f^2 - f_c^2)}\right]}{\pi}$$

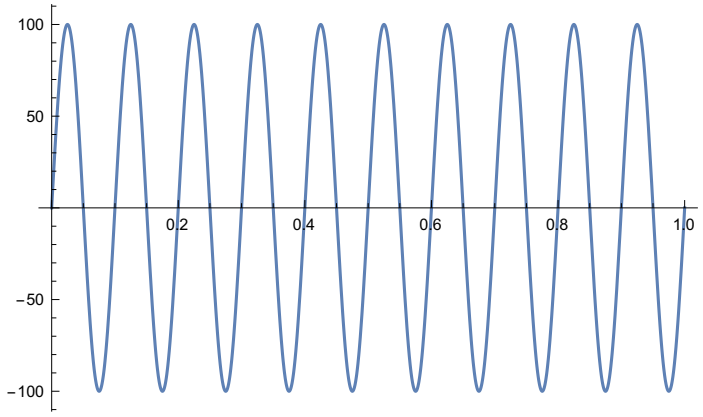
Define constants

In[3]:=
```
Tnum = 1 * 10^0; fcnum = 1 * 10^1; Anum = 100;
```

Plot time-series

In[4]:=
```
Plot[Evaluate[sine /. {A → Anum, f_c → fcnum}], {t, 0, Tnum}, PlotRange → All]
```

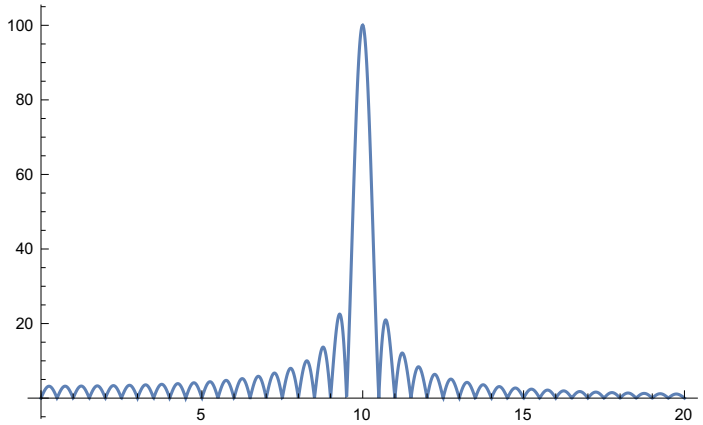
Out[4]=



Plot Fourier Transform

In[5]:=
```
Plot[Evaluate[FT /. {A → Anum, f_c → fcnum, T → Tnum}],
 {f, 0, 20}, PlotRange → Full(*,PlotPoints→1000*)]
```

Out[5]=



Fourier Transform of Gaussian

Define Functions

In[6]:=
$$\text{Gaussian} = \frac{A}{\sqrt{2\pi}\,\sigma}\, e^{\frac{-t^2}{2\sigma^2}}$$

Out[6]=
$$\frac{A\, e^{-\frac{t^2}{2\sigma^2}}}{\sqrt{2\pi}\,\sigma}$$

In[7]:=
$$\text{FT} = \text{Abs}\left[\frac{1}{T}\int_{-T}^{T}\left(\text{Gaussian}\, e^{2\pi f i t}\right) dt\right]$$

Out[7]=
$$\frac{1}{2}\, e^{-2\pi^2 \text{Re}[f^2\sigma^2]}\, \text{Abs}\left[\frac{A\left(\text{Erf}\left[\frac{T-2 i f \pi \sigma^2}{\sqrt{2}\,\sigma}\right] + \text{Erf}\left[\frac{T+2 i f \pi \sigma^2}{\sqrt{2}\,\sigma}\right]\right)}{T}\right]$$

Define constants

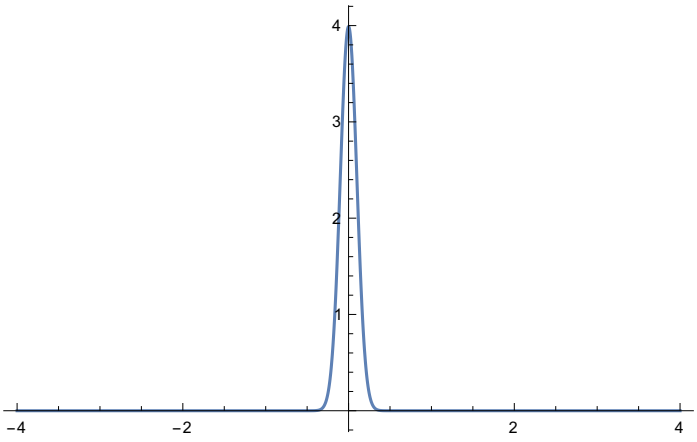
In[8]:=
```
Tnum = 4 * 10^0; sigmanum = 1 * 10^-1; Anum = 1;
```

Plot time-series

In[9]:=
```
Plot[Evaluate[Gaussian /. {A → Anum, σ → sigmanum}], {t, -Tnum, Tnum}, PlotRange → All]
```

General: Exp[−799.935] is too small to represent as a normalized machine number; precision may be lost.

Out[9]=



Plot Fourier Transform

In[10]:=
```
Integrate[Gaussian /. {A → Anum, σ → sigmanum}, {t, -∞, ∞}]
```

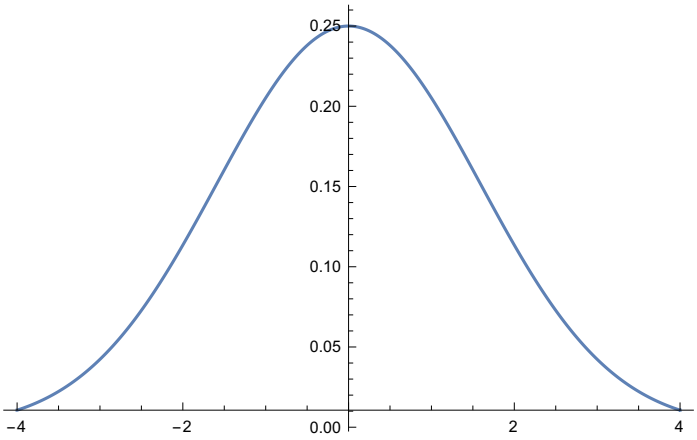
Out[10]= 1

In[11]:=
```
Plot[Evaluate[FT /. {A → Anum, σ → sigmanum, T → Tnum}],
 {f, -4, 4}, PlotRange → Full, PlotPoints → 2000]
```

General: Exp[−800.186 − 100.594 *i*] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[−800.186 + 100.594 *i*] is too small to represent as a normalized machine number; precision may be lost.

Out[11]=



Fourier Transform of Lorentzian

Define Functions

In[18]:=
```
Lorentzian = A Exp[-Γ t]
```

Out[18]=
$$A\, e^{-t\,\Gamma}$$

In[19]:=
```
FT = Abs[1/T ∫_-T^T (Lorentzian e^(2 π f i t)) dt]
```

Out[19]=
$$2\,\text{Abs}\left[\frac{A\,\text{Sin}[T\,(2\,f\,\pi + i\,\Gamma)]}{T\,(2\,f\,\pi + i\,\Gamma)}\right]$$

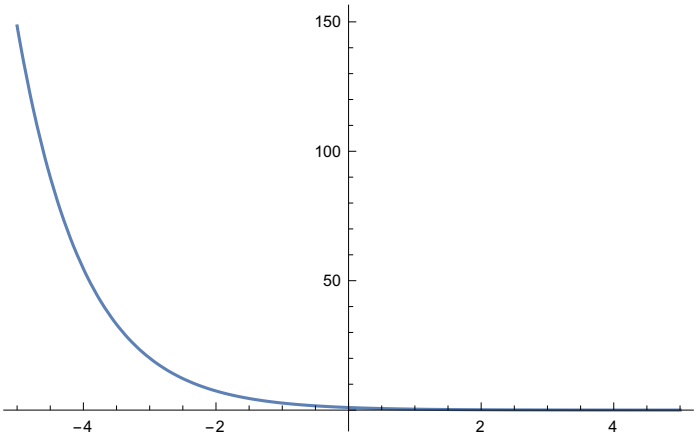
Define constants

In[20]:=
```
Tnum = 5 * 10^0; gammanum = 1 * 10^0; Anum = 1;
```

Plot time-series

In[22]:=
```
Plot[Evaluate[Lorentzian /. {A → Anum, Γ → gammanum}], {t, -Tnum, Tnum}, PlotRange → All]
```

Out[22]=



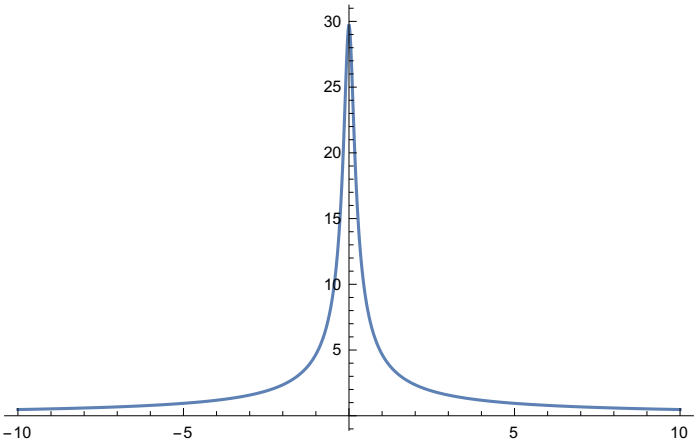
Plot Fourier Transform

```
(*Integrate[sineGaussian/.{A→A,σ→σ},{t,-∞,∞}]*)
```

Out[ ]=
```
0
```

In[25]:=
```
Plot[Evaluate[FT /. {A → Anum, Γ → gammanum, T → Tnum}],
 {f, -10, 10}, PlotRange → All(*,PlotPoints→2000*)]
```

Out[25]=



Fourier Transform of sine - Gaussian

Define Functions

In[12]:=
$$\text{sineGaussian} = \frac{A}{\sqrt{2\pi}\,\sigma}\, e^{\frac{-t^2}{2\sigma^2}}\, \text{Sin}[2\pi f_c t]$$

Out[12]=
$$\frac{A\, e^{-\frac{t^2}{2\sigma^2}}\, \text{Sin}[2\pi t f_c]}{\sqrt{2\pi}\,\sigma}$$

In[13]:=
$$\text{FT} = \text{Abs}\left[\frac{1}{T}\int_{-T}^{T}\left(\text{sineGaussian}\, e^{2\pi f i t}\right) dt\right]$$

Out[13]=
$$\frac{1}{4} e^{-2\pi^2 \text{Re}[\sigma^2 (f+f_c)^2]}\, \text{Abs}\left[\frac{1}{T} A \left(-2 + e^{8 f \pi^2 \sigma^2 f_c}\left(\text{Erf}\left[\frac{T - 2 i \pi \sigma^2 (f - f_c)}{\sqrt{2}\,\sigma}\right] + \text{Erf}\left[\frac{T + 2 i \pi \sigma^2 (f - f_c)}{\sqrt{2}\,\sigma}\right]\right) + \text{Erfc}\left[\frac{T - 2 i \pi \sigma^2 (f + f_c)}{\sqrt{2}\,\sigma}\right] + \text{Erfc}\left[\frac{T + 2 i \pi \sigma^2 (f + f_c)}{\sqrt{2}\,\sigma}\right]\right)\right]$$

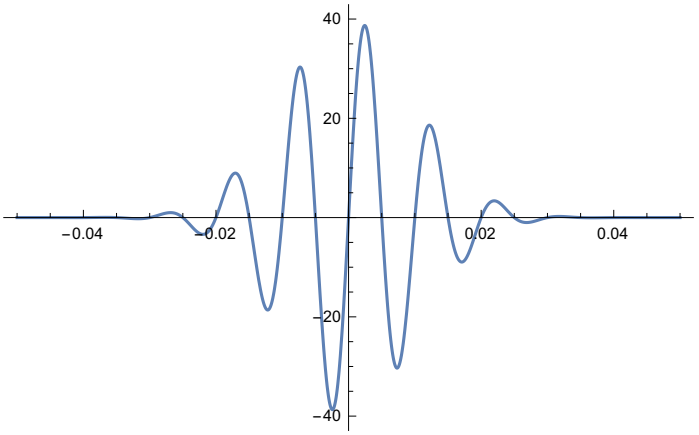
Define constants

In[14]:=
```
Tnum = 5 * 10^-2; sigmanum = 1 * 10^-2; Anum = 1; fcnum = 100;
```

Plot time-series

In[15]:=
```
Plot[Evaluate[sineGaussian /. {A → Anum, σ → sigmanum, f_c → fcnum}],
 {t, -Tnum, Tnum}, PlotRange → All]
```

Out[15]=



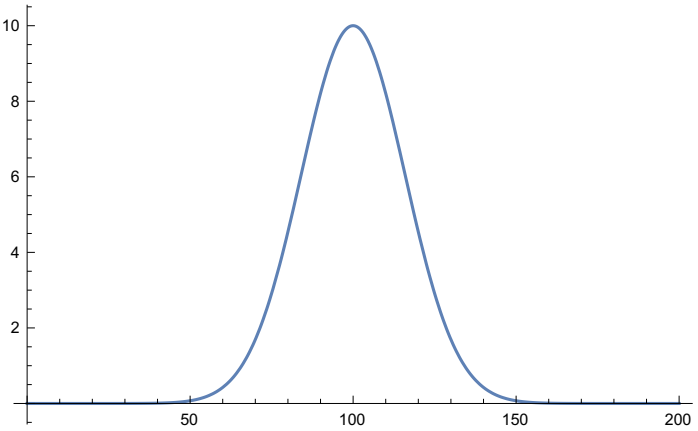
Plot Fourier Transform

```
(*Integrate[sineGaussian/.{A→A,σ→σ},{t,-∞,∞}]*)
```

Out[ ]= 0

In[16]:=
```
Plot[Evaluate[FT /. {A → Anum, σ → sigmanum, f_c → fcnum, T → Tnum}],
 {f, 0, 200}, PlotRange → All(*,PlotPoints→2000*)]
```

Out[16]=



Fourier Transform of sine - Gaussian

Define Functions

In[1]:=
$$\text{sineGaussian} = \frac{A}{\sqrt{2\pi}\,\sigma}\, e^{\frac{-t^2}{2\sigma^2}}\, \text{Sin}[2\pi f_c t]$$

Out[1]=
$$\frac{A\, e^{-\frac{t^2}{2\sigma^2}}\, \text{Sin}[2\pi t f_c]}{\sqrt{2\pi}\,\sigma}$$

In[2]:=
$$\text{FT} = \text{Abs}\left[\frac{1}{\sqrt{T}}\int_{-T}^{T}\left(\text{sineGaussian}\, e^{2\pi f i t}\right) dt\right]$$

Out[2]=
$$\frac{1}{4}\, e^{-2\pi^2 \text{Re}[\sigma^2 (f+f_c)^2]}\, \text{Abs}\left[\frac{1}{\sqrt{T}} A \left(-2 + e^{8 f \pi^2 \sigma^2 f_c}\left(\text{Erf}\left[\frac{T - 2 i \pi \sigma^2 (f - f_c)}{\sqrt{2}\,\sigma}\right] + \text{Erf}\left[\frac{T + 2 i \pi \sigma^2 (f - f_c)}{\sqrt{2}\,\sigma}\right]\right) + \text{Erfc}\left[\frac{T - 2 i \pi \sigma^2 (f + f_c)}{\sqrt{2}\,\sigma}\right] + \text{Erfc}\left[\frac{T + 2 i \pi \sigma^2 (f + f_c)}{\sqrt{2}\,\sigma}\right]\right)\right]$$

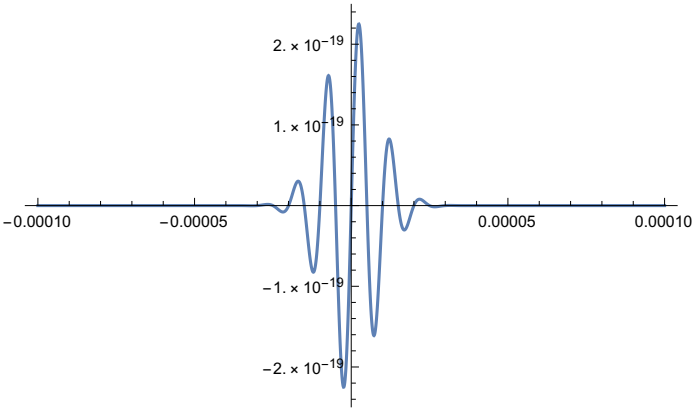
Define constants

In[56]:=
```
Tnum = 5 * 10^-4; sigmanum = 8.5 * 10^-6; Anum = 5 * 10^-24; fcnum = 10^5;
```

Plot time-series

In[57]:=
```
Plot[Evaluate[sineGaussian /. {A → Anum, σ → sigmanum, f_c → fcnum}],
 {t, -0.0001, 0.0001}, PlotRange → All, PlotPoints → 2000]
```

Out[57]=



Plot Fourier Transform

In[103]:=
```
fourier := Evaluate[FT /. {A → Anum, σ → sigmanum, f_c → fcnum, T → Tnum}]
memory := 10^-22 f^-1/2 UnitStep[fcnum - f]
```

In[105]:=
```
LogLogPlot[{fourier, memory, (fourier + memory)}, {f, 10^1, 10^9},
 PlotRange → {{10^1, 10^9}, {10^-28, 10^-18}}, GridLines → {{fcnum, 10^-28}}]
```

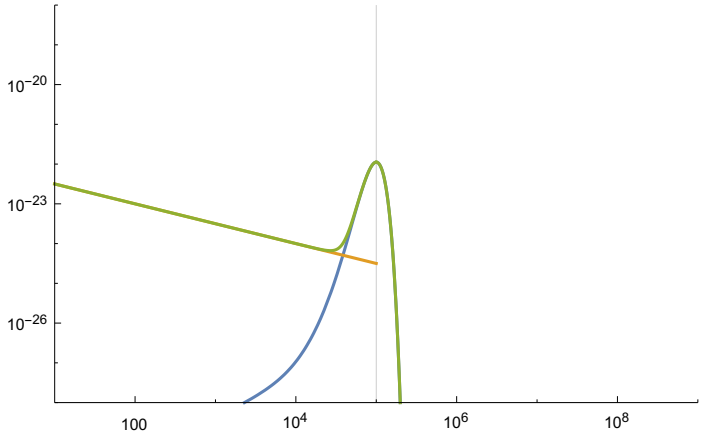
General: Exp[−1719.58 − 314.218 *i*] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[−1719.58 + 314.218 *i*] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[−1719.57 + 314.281 *i*] is too small to represent as a normalized machine number; precision may be lost.

General: Further output of General::munfl will be suppressed during this calculation.

Out[105]=



4.2 Python

Fourier transforms in Python are slightly more complicated than Mathematica. $A, \sigma, f_c,$ and Γ maintain their original definitions, but we now need to choose the number of points (N) in the time domain. Once we choose this, the number of points in the frequency domain is fixed (N/2).

The sampling frequency is another variable that needs to be specified. This is determined by the instrument; how many measurements can we make (per second). If we make 1000 measurements per second, then we have a sampling frequency f_s of 100 Hz. The integration time T has the same definition as in Mathematica. From here, we can define $N = Tf_s$. Our time domain t, runs from 0 to T with N evenly spaced points.

There are two different packages which compute Fourier Transforms in Python: scipy and numpy. Scipy builds off of numpy and has a faster runtime, but the primary methods remain similar. To take the FFT, you first run fft(sineGaussian) and then take the positive frequencies by slicing the array from 0 to $N/2$.

The range of frequencies in the frequency domain is dependent on the sampling frequency. The frequency range runs from 0 to $f_s/2$ with $N/2$ points. The resolution $\Delta f = 1/T$.

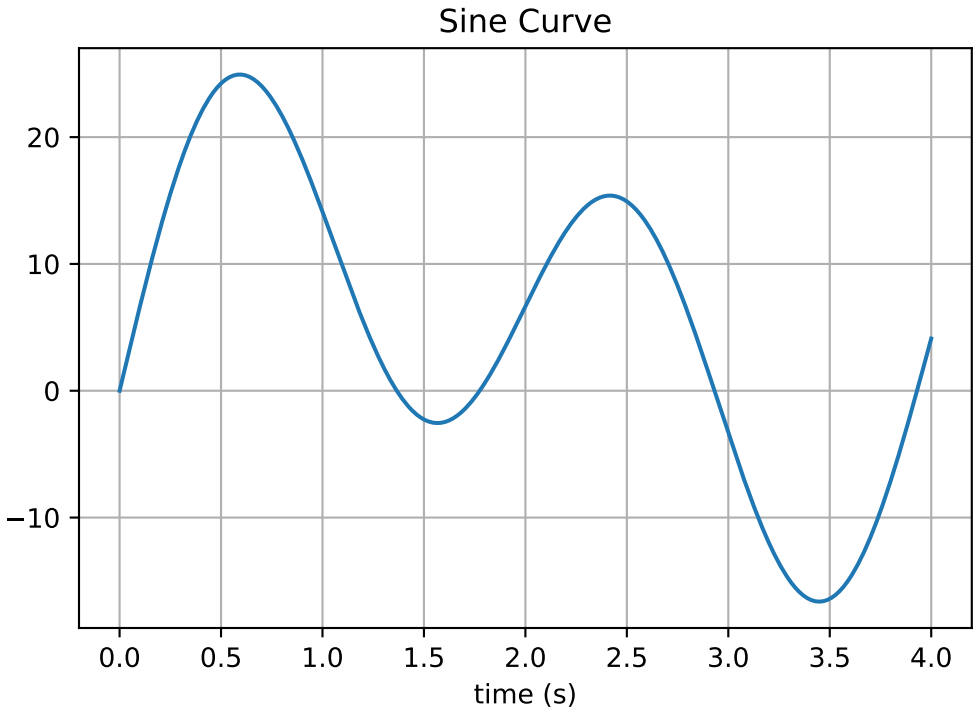
From this, we get a series of equations describing the Discrete Fourier Transform. We find $f_{\max} = f_s/2 = \frac{N}{2T}$, $f_{\min} = 1/T = \Delta f$, $T = N/f_s = 1/\Delta f$, and $f_s = 2f_{\max} = N/T$.

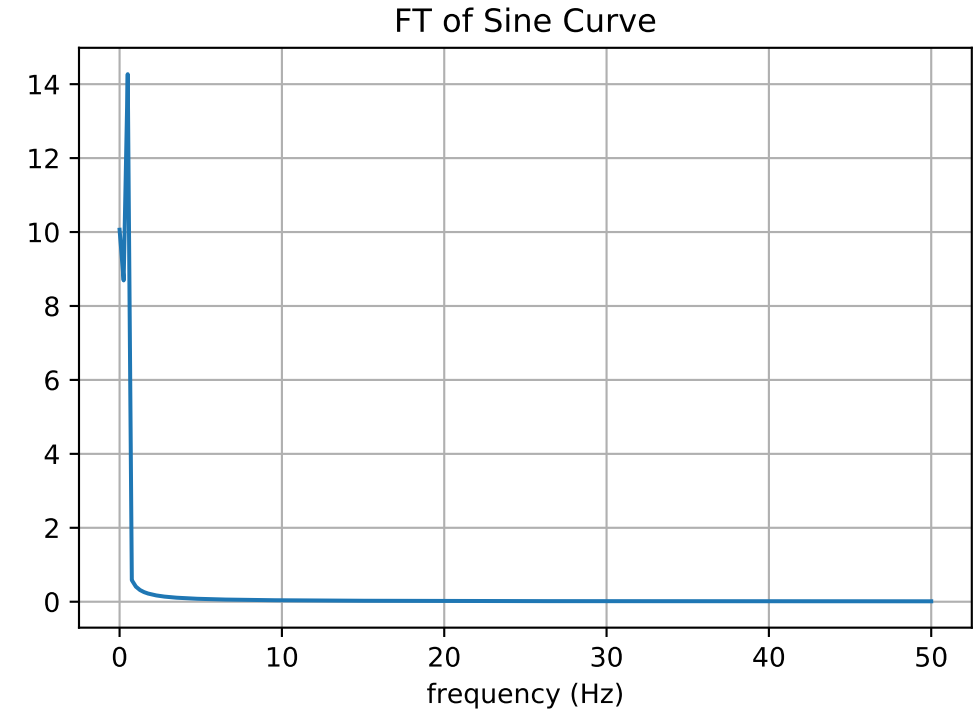
One example of this has us knowing T and f_s. We have a detector with a sampling frequency $f_s = 100$ Hz, and we observe a signal for $T = 4$ s. That means $N = Tf_s = 4 \times 100 = 400$ points. Choosing just the positive values from the FT, we are left with 200 points. The resolution $\Delta f = 1/T = \frac{1}{4\text{ s}} = 0.25$ Hz. That also means the minimum frequency is 0.25 Hz. The maximum frequency $f_{\max} = f_s/2 = \frac{N}{2}\Delta f = 50$ Hz. See the code and 3 plots below.

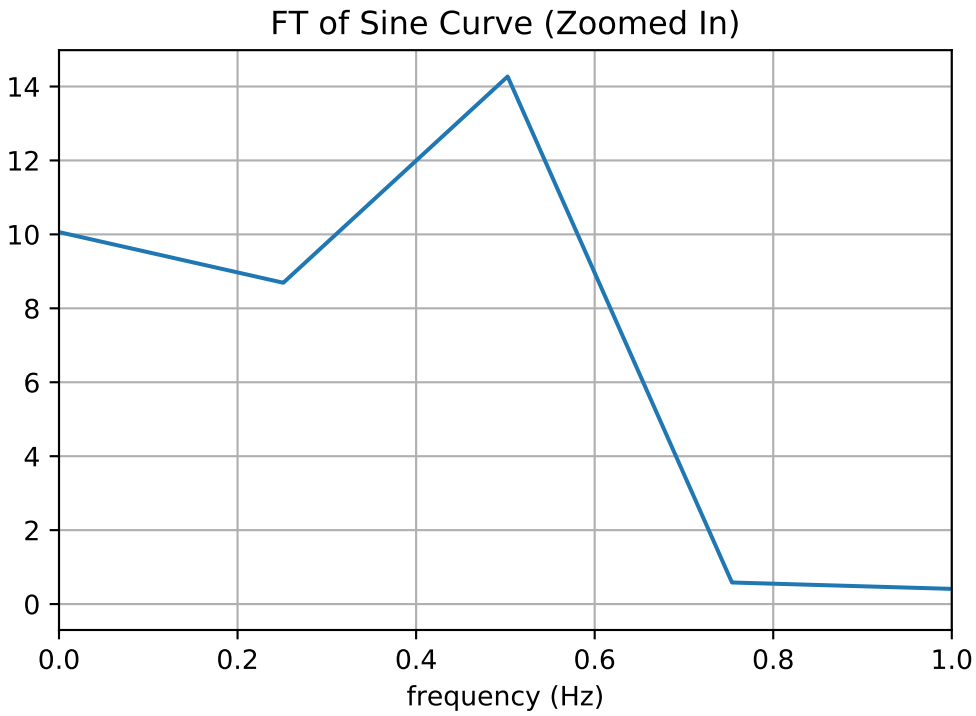
```
# Reference: sineFT-np,scipy,rfft,fft-3.ipynb (same directory)

import numpy as np
from scipy.fftpack import *
import matplotlib.pyplot as plt

T = 4 # seconds
fs = 100 # Hz

# Number of sample points
N = T*fs

fc = 0.1
fc2= 0.25
fc3= 0.5
A = 7
B=10
```

```
C=15

t = np.linspace(0, T, int(N))

sine = A*np.sin(2.0*np.pi*fc*t)+B*np.sin(2.0*np.pi*fc2*t)+C*np.sin(2.0*np.pi*fc3*t
    )

plt.plot(t,sine)
plt.xlabel('time (s)')
plt.title('Sine Curve')
plt.grid()
# plt.xlim(0,0.5)
plt.savefig('sineFT-8619-p1.pdf')
plt.show()

FT=np.abs(2./N*(np.fft.fft(sine)[0:int(N)//2])) #[0:int(N)//2] with fft

xf = np.linspace(0, fs/2, int(N//2))

plt.plot(xf, FT)
# plt.xlim(0,50)
plt.xlabel('frequency (Hz)')
plt.title('FT of Sine Curve')
plt.grid()
plt.savefig('sineFT-8619-p2.pdf')
plt.show()


plt.plot(xf, FT)
plt.xlim(0,1)
plt.xlabel('frequency (Hz)')
plt.title('FT of Sine Curve (Zoomed In)')
plt.grid()
plt.savefig('sineFT-8619-p3.pdf')
plt.show()
```

Listing 1: sineFT-8619.py


Sine Curve
20
10
0
−10
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
time (s)


FT of Sine Curve
14
12
10
8
6
4
2
0
0
10
20
30
40
50
frequency (Hz)

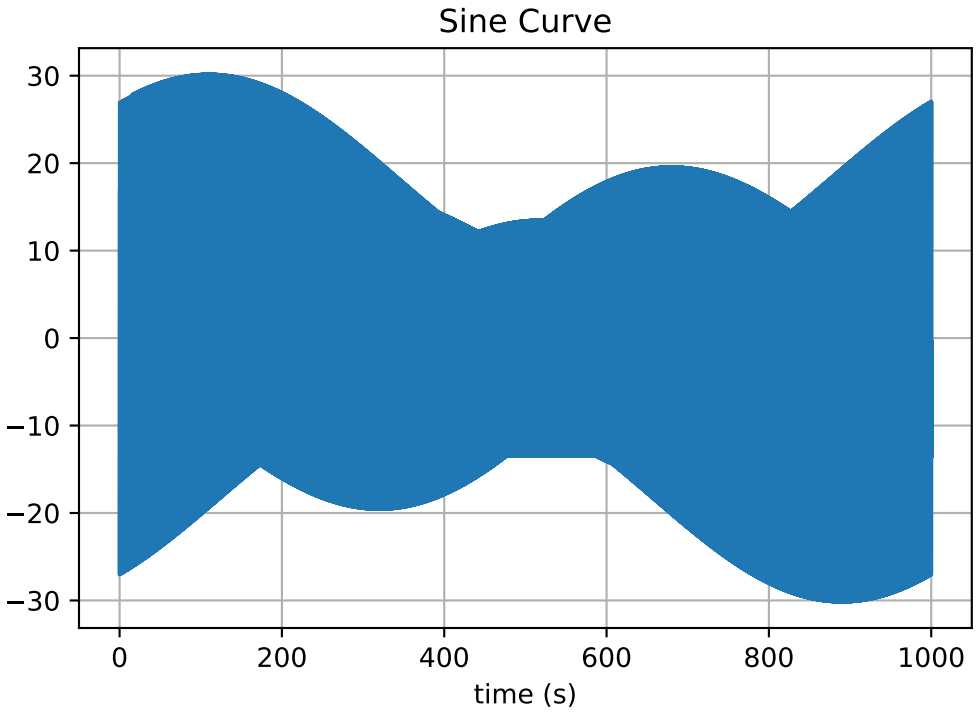


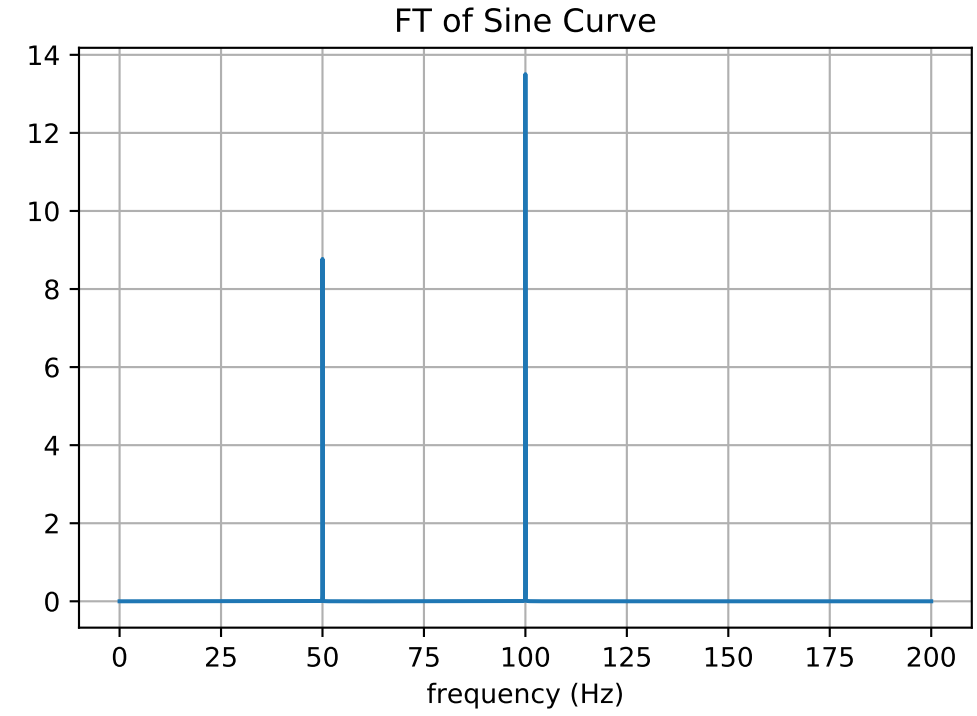
Conversely, if we know what frequency resolution we need and the frequency range, we can find the sampling frequency and integration time required. Suppose we need to distinguish between two signals at 50.000 Hz and 50.001 Hz and the frequency range is between 1 mHz and 200 Hz. Thus, $\Delta f = 0.001 \text{ Hz} = 1/T \implies T = \frac{1}{0.001} = 1000$ s. Similarly, $f_{\max} = 200 \text{ Hz} = f_s/2 \implies f_s = 400$ Hz. See the code and plots.

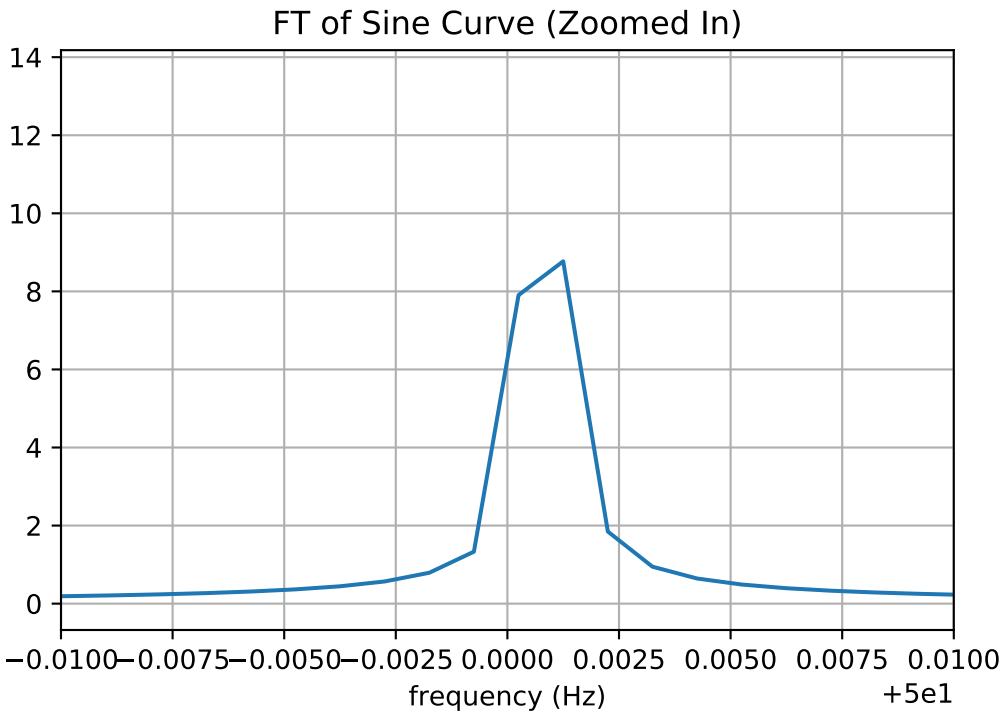
```
# Reference: sineFT-np,scipy,rfft,fft-3.ipynb (same directory)

import numpy as np
from scipy.fftpack import *
import matplotlib.pyplot as plt

T = 1000 # seconds
fs = 400 # Hz

# Number of sample points
N = T*fs

fc = 50
fc2= 50.001
fc3= 100
A = 7
B=10
```

```
C=15

t = np.linspace(0, T, int(N))

sine = A*np.sin(2.0*np.pi*fc*t)+B*np.sin(2.0*np.pi*fc2*t)+C*np.sin(2.0*np.pi*fc3*t
    )

plt.plot(t,sine)
plt.xlabel('time (s)')
plt.title('Sine Curve')
plt.grid()
# plt.xlim(0,0.5)
plt.savefig('sineFT-8619-p4.pdf')
plt.show()

FT=np.abs(2./N*(np.fft.fft(sine)[0:int(N)//2])) #[0:int(N)//2] with fft

xf = np.linspace(0, fs/2, int(N//2))

plt.plot(xf, FT)
# plt.xlim(0,50)
plt.xlabel('frequency (Hz)')
plt.title('FT of Sine Curve')
plt.grid()
plt.savefig('sineFT-8619-p5.pdf')
plt.show()


plt.plot(xf, FT)
plt.xlim(49.990,50.010)
plt.xlabel('frequency (Hz)')
plt.title('FT of Sine Curve (Zoomed In)')
plt.grid()
plt.savefig('sineFT-8619-p6.pdf')
plt.show()
```

Listing 2: sineFT-8619-2.py


Sine Curve
30
20
10
0
−10
−20
−30
0
200
400
600
800
1000
time (s)


FT of Sine Curve
14
12
10
8
6
4
2
0
0
25
50
75
100
125
150
175
200
frequency (Hz)



Normalization looks like $1/N$ for both numpy and scipy, according to the documentation. Testing has indicated that $2/N$ recovers the original amplitude of a sine wave.

5 Future Work

The next steps are to connect these Fourier transforms to orphan memory. We are starting with a sine-Gaussian wavepacket and will eventually be looking at the energy of the wave and Signal-to-Noise ratios from a variety of current and upcoming detectors. Stay Tuned!