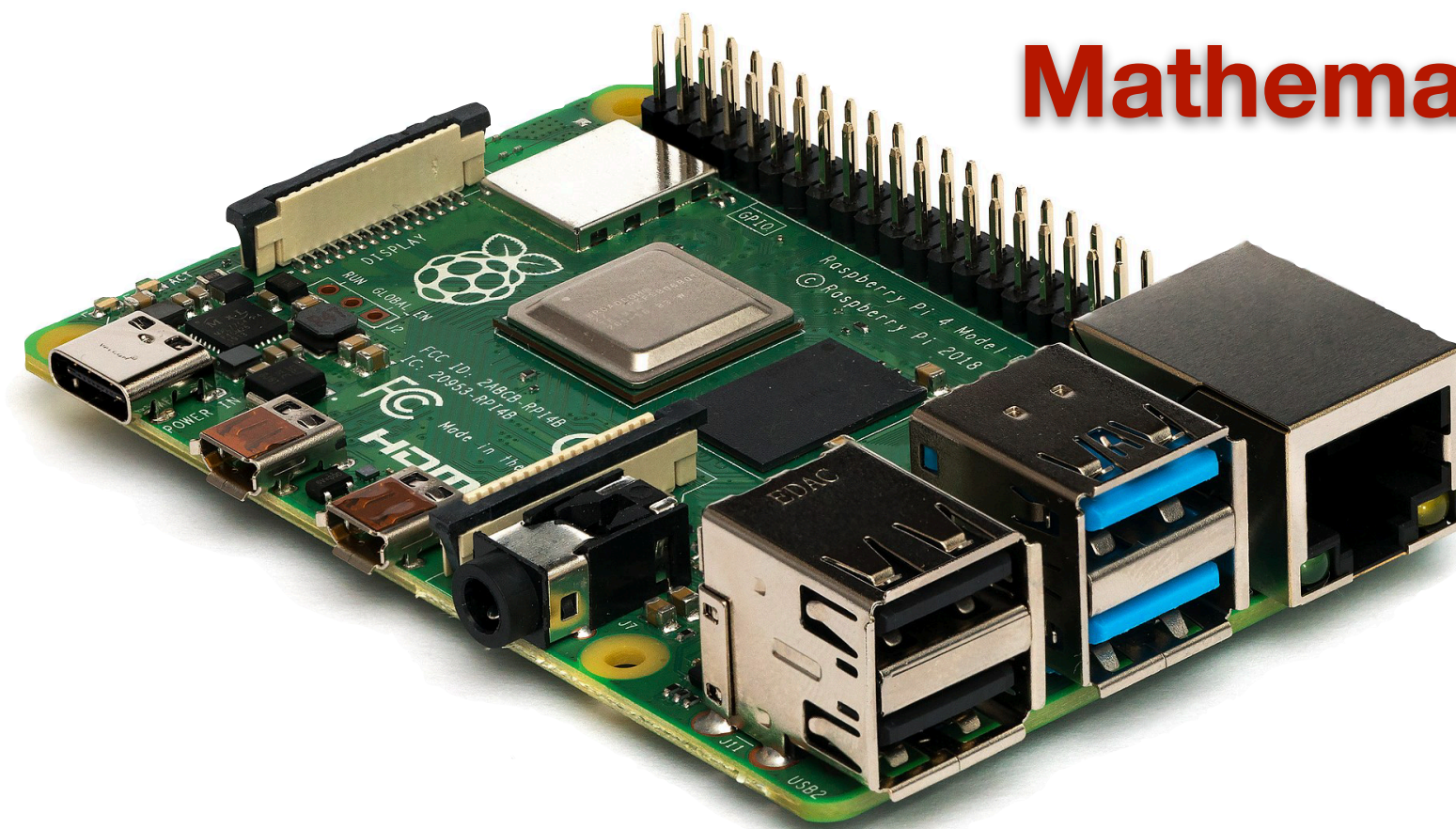




Mathematica

&

Raspberry Pi

[Courtesy:Michael H. \(„Laserlicht“\) / Wikimedia Commons](#)

Examples for the novice

Introduction

Wolfram's Mathematica is a very powerful computational software suite, and the authors have graciously offered it without cost for the Raspberry Pi.

The Wolfram language can be quite daunting for the novice because it is capable of extremely complex tasks, well beyond the novice.

This ebook is intended to be a collection of examples to help the beginner get started and hopefully encourage them to learn the Wolfram language.

Resources

- [Wolfram Language Documentation Center](#)
- [Wolfram Demonstrations Project](#)

Trigonometric Functions

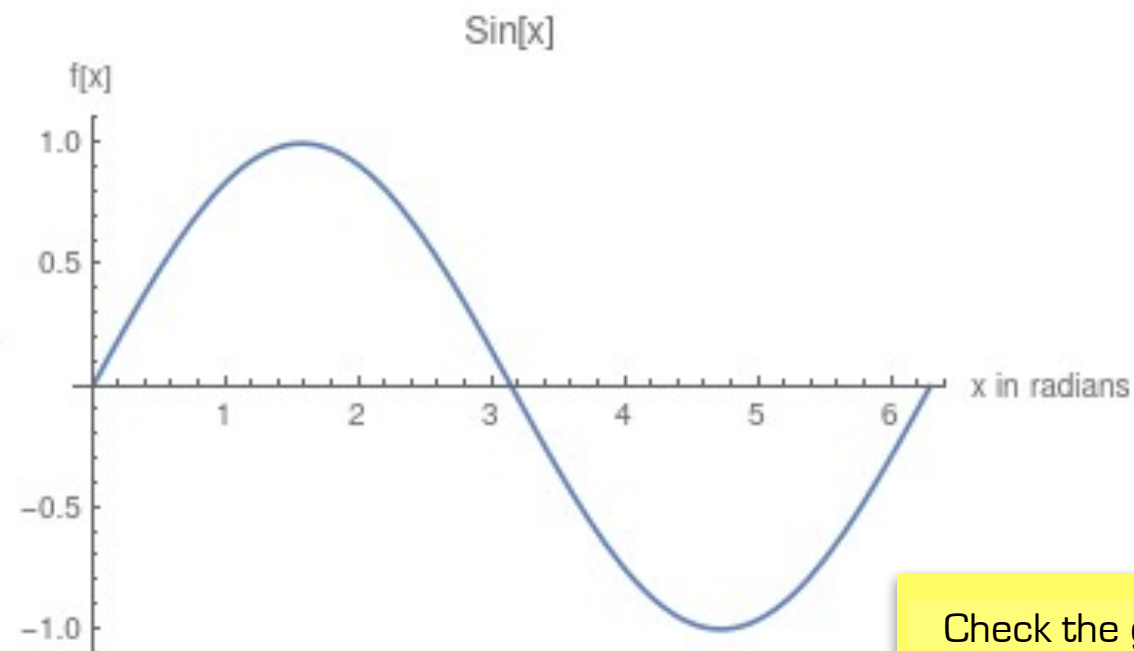
The basic trigonometric functions are very easy to plot. The default angular unit in Mathematica is radians. However, this can be changed to degrees by using the [Degree](#) modifier or using a conversion factor. The [Ticks](#) command can be used to change the axis scale.

Plotting Functions

Gallery

Plot in Radians

```
f = Sin[x];  
Plot[f, {x, 0, 2  $\pi$ },  
  PlotLabel -> "Sin[x]",  
  AxesLabel -> {"x in radians", "f[x]"}]
```



Check the gallery for
plotting examples

Plot in Radians

Plotting Functions using Manipulate

Always [Clear](#) values and definitions, otherwise Mathematica will use previous ones.

```
= Clear[function, fcn1, fcn2, fcn3, fcn4, fcn5]
```

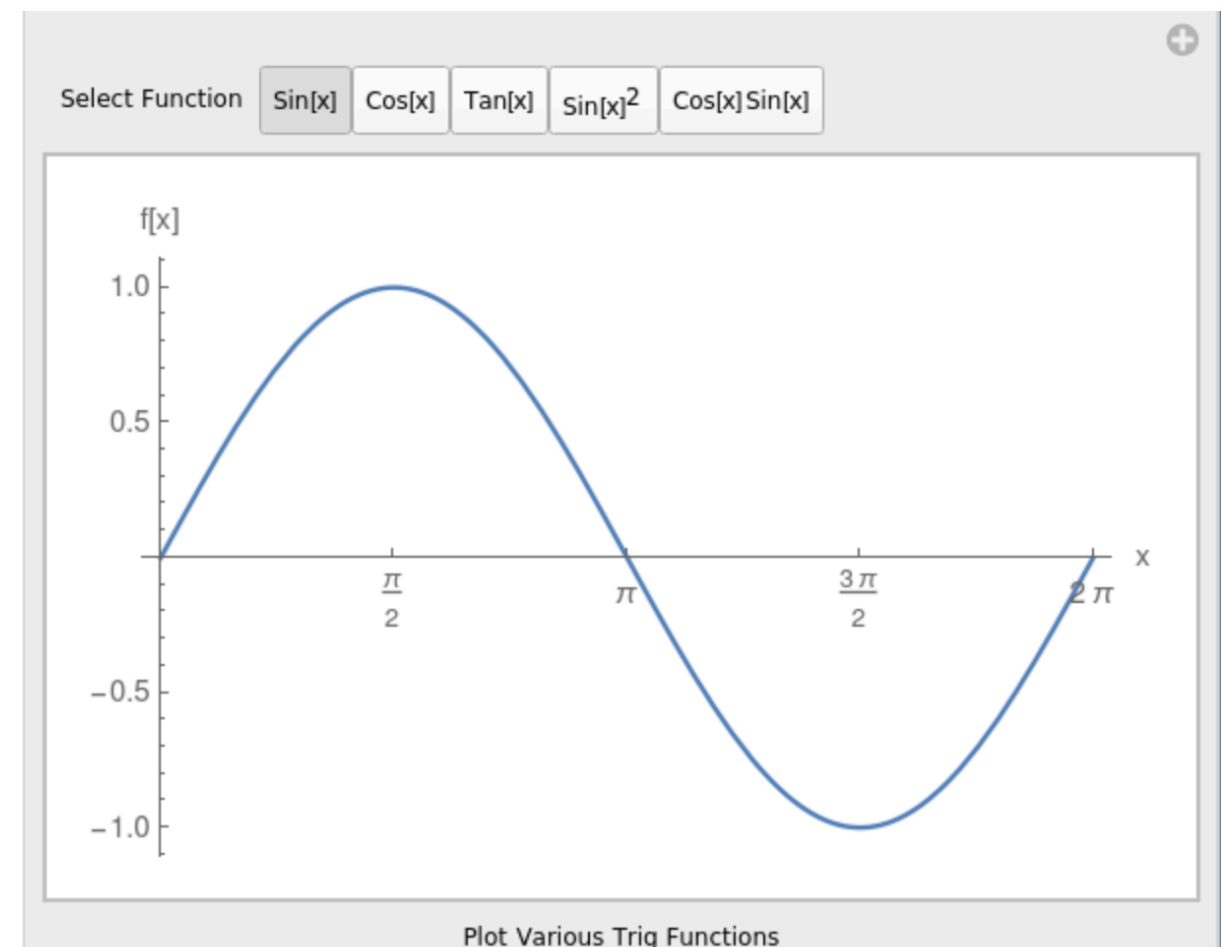
```
fcn1 = Sin[x];  
fcn2 = Cos[x];  
fcn3 = Tan[x];  
fcn4 = Sin[x] Sin[x];  
fcn5 = Sin[x] Cos[x];
```

```
Manipulate[  
  Plot[function, {x, 0, 2  $\pi$ },  
    AxesLabel  $\rightarrow$  {"x", "f[x]"},  
    Ticks  $\rightarrow$  {{ $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ ,  $2\pi$ }, Automatic}],  
  
  {{function, fcn1, "Select Function"},  
   {fcn1, fcn2, fcn3, fcn4, fcn5}},  
  FrameLabel  $\rightarrow$  "Plot Various Trig Functions"]
```

Use [Ticks](#) to change the x and y axis scales

See Tech Note: [Introduction to Manipulate](#)

Gallery



Sin(x)

The [Manipulate](#) command can be used to select between several functions.

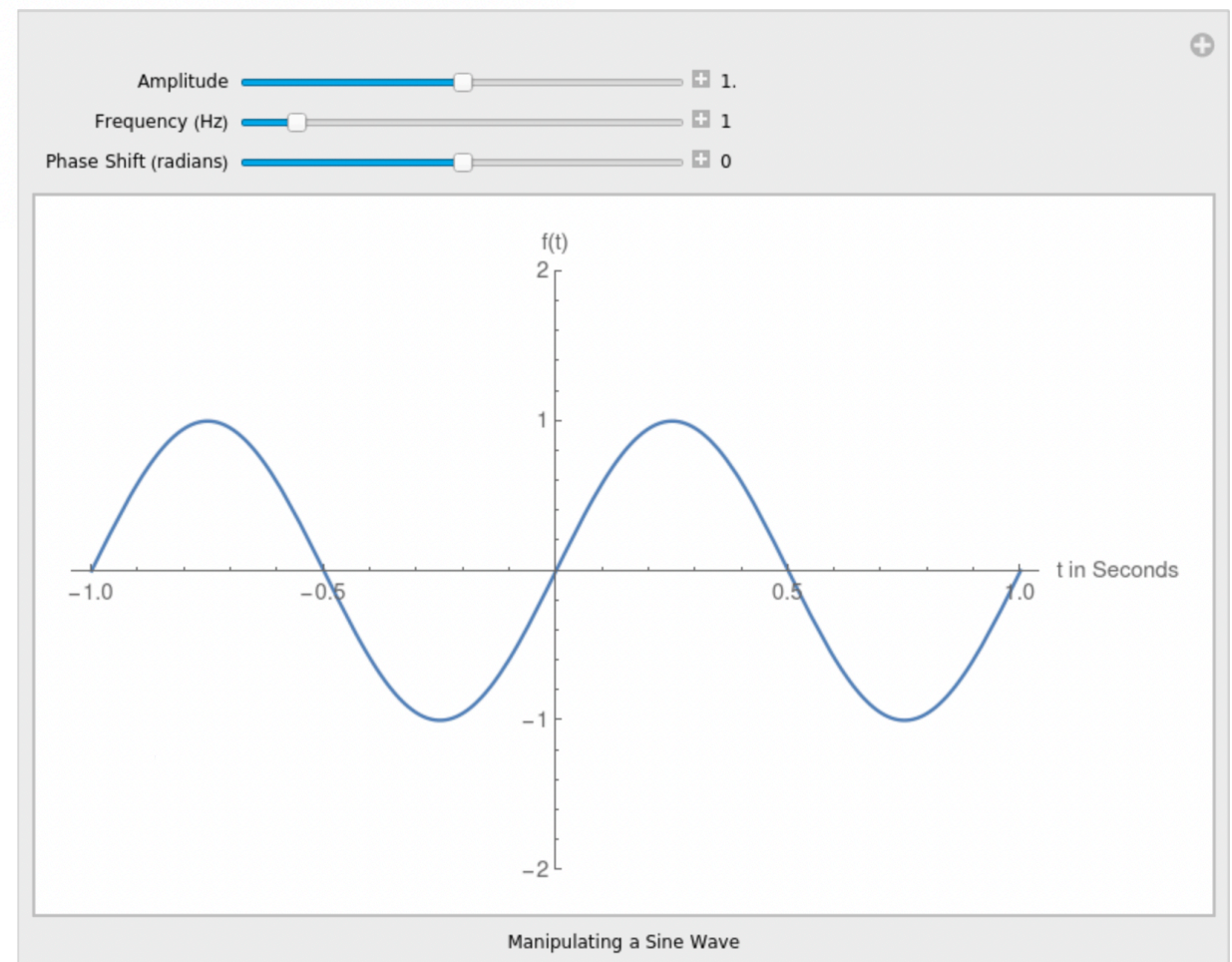
Manipulating Function Parameters

```
Manipulate[
  Plot[A Sin[2  $\pi$  f t +  $\phi$ ], {t, -1, 1},
    PlotRange  $\rightarrow$  {Automatic, {-2, 2}},
    AxesLabel  $\rightarrow$  {"t in Seconds", "f(t)"}, ImageSize  $\rightarrow$  500],

  {{A, 1.0, "Amplitude"}, 0, 2, Appearance  $\rightarrow$  "Labeled"},
  {{f, 1, "Frequency (Hz)"}, 0.5, 5, Appearance  $\rightarrow$  "Labeled"},
  {{ $\phi$ , 0, "Phase Shift (radians)"}, - $\pi$ ,  $\pi$ , Appearance  $\rightarrow$  "Labeled"},
  Gallery
]
FrameLabel  $\rightarrow$  "Manipulating a Sine Wave"
```

Sliders can be used to vary function parameters.

Use [Appearance](#) $\rightarrow$ "Labeled" to place the parameter value at the end of the slider.



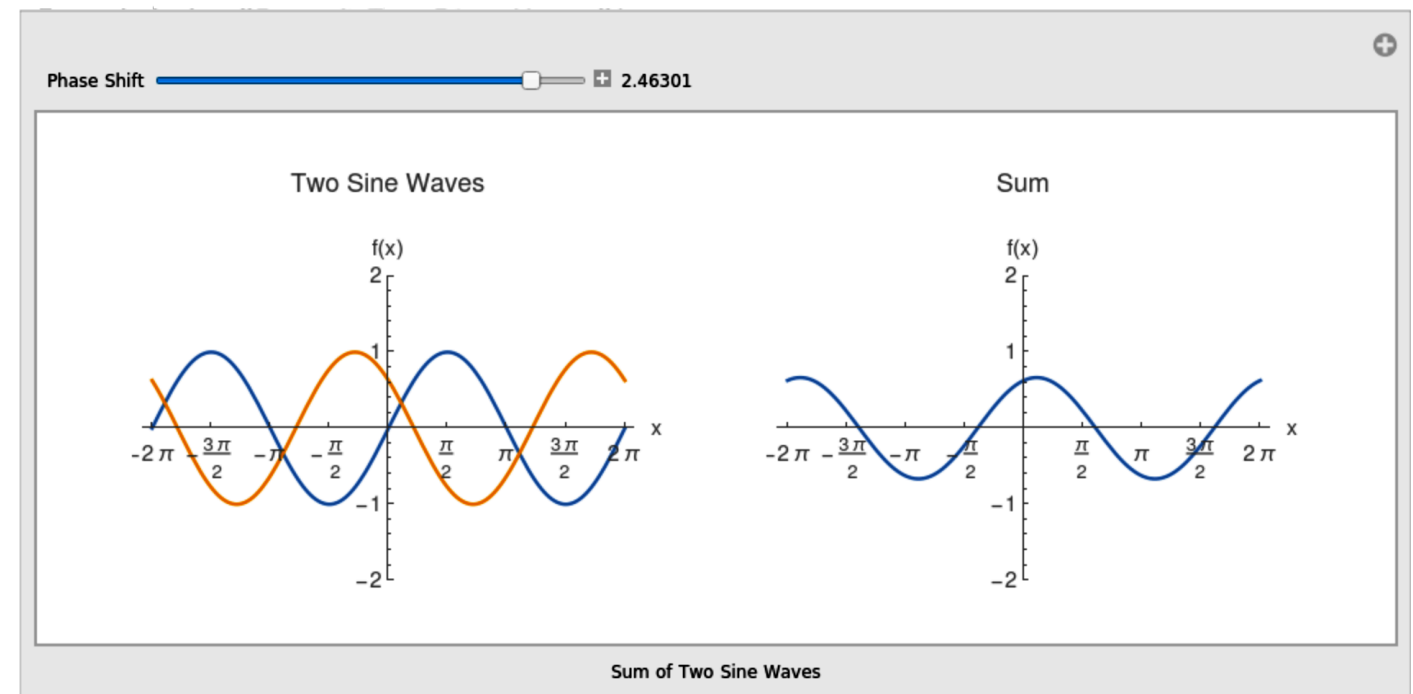
Plotting & Adding two Sine Waves

```
Manipulate[
GraphicsRow[{

Plot[{Sin[x], Sin[x +  $\phi$ ]}, {x, -2  $\pi$ , 2  $\pi$ },
Ticks  $\rightarrow$  {{-2  $\pi$ ,  $\frac{-3 \pi}{2}$ , - $\pi$ ,  $\frac{-\pi}{2}$ ,  $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3 \pi}{2}$ , 2  $\pi$ }, Automatic},
PlotRange  $\rightarrow$  {Automatic, {-2, 2}},
AxesLabel  $\rightarrow$  {"x", "f(x)"},
PlotLabel  $\rightarrow$  "Two Sine Waves"},

Plot[{Sin[x] + Sin[x +  $\phi$ ]}, {x, -2  $\pi$ , 2  $\pi$ },
Ticks  $\rightarrow$  {{-2  $\pi$ ,  $\frac{-3 \pi}{2}$ , - $\pi$ ,  $\frac{-\pi}{2}$ ,  $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3 \pi}{2}$ , 2  $\pi$ }, Automatic},
PlotRange  $\rightarrow$  {Automatic, {-2, 2}},
AxesLabel  $\rightarrow$  {"x", "f(x)"},
PlotLabel  $\rightarrow$  "Sum"}],
{{ $\phi$ ,  $\frac{\pi}{4}$ , "Phase Shift"}, - $\pi$ ,  $\pi$ , Appearance  $\rightarrow$  "Labeled"},
FrameLabel  $\rightarrow$  "Sum of Two Sine Waves"]
```

Use [GraphicsRow](#) to place two plots side by side.



Plotting a Series of Sine Waves

The parameter f is being varied

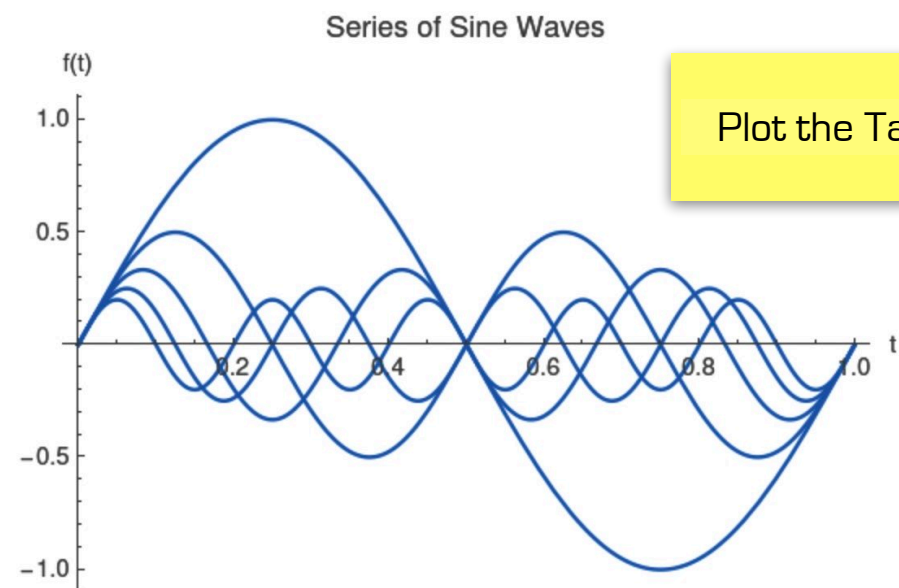
Make a [Table](#) of the series

```
Table[ $\frac{1}{f}$  Sin[2  $\pi$  f t], {f, 1, 5}]
```

```
{Sin[2  $\pi$  t],  $\frac{1}{2}$  Sin[4  $\pi$  t],  $\frac{1}{3}$  Sin[6  $\pi$  t],  $\frac{1}{4}$  Sin[8  $\pi$  t],  $\frac{1}{5}$  Sin[10  $\pi$  t]}
```

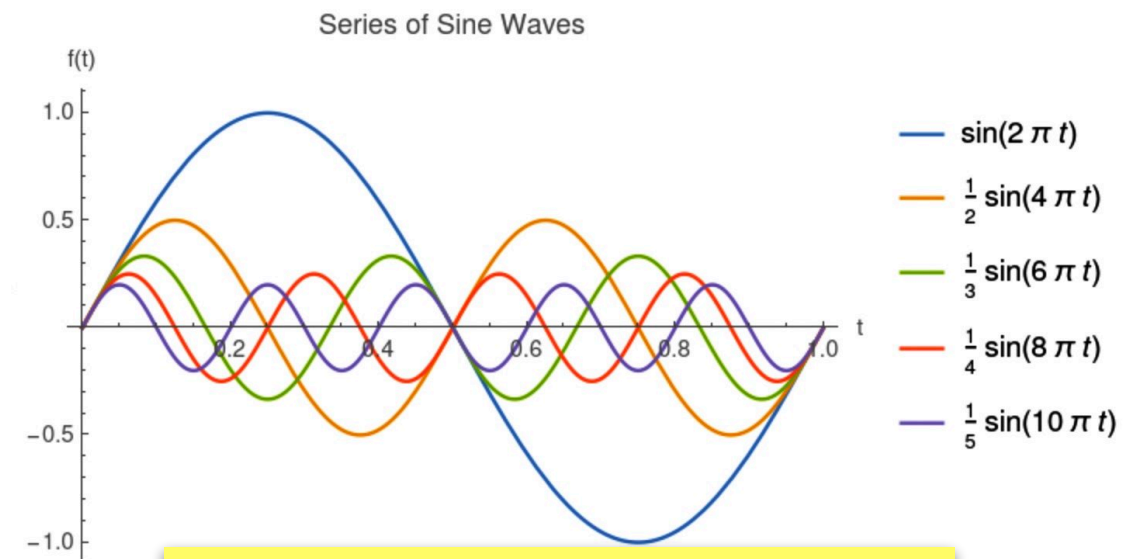
The [Evaluate](#) command is used to separate each table component.

```
Plot[Table[ $\frac{1}{f}$  Sin[2  $\pi$  f t], {f, 1, 5}], {t, 0, 1},  
PlotLabel -> "Series of Sine Waves",  
AxesLabel -> {"t", "f(t)"}]
```



Plot the Table

```
Plot[Evaluate@Table[ $\frac{1}{f}$  Sin[2  $\pi$  f t], {f, 1, 5}], {t, 0, 1},  
PlotLabel -> "Series of Sine Waves",  
AxesLabel -> {"t", "f(t)"},  
PlotLegends -> "Expressions"]
```



The [PlotLegends](#) command is used to label each table component.

Plotting Hyperbolic Functions

```
ClearAll["Global`*"]
```

```
Manipulate[
Module[{funcs, selectedFuncs, xlo, xhi, eps = 10-6, xticks,
kmin, kmax},
```

```
(*Define hyperbolic functions and labels*)
```

```
funcs = <|
  "Sinh" → Sinh,
  "Cosh" → Cosh,
  "Tanh" → Tanh,
  "Csch" → Csch,
  "Sech" → Sech,
  "Coth" → Coth|>;
```

Label the functions to use as tick boxes

```
selectedFuncs = Values@KeyTake[funcs, selected];
```

```
(*to prevent unexpected zeros when plotting*)
```

```
{xlo, xhi} = Sort@N@{xmin, xmax};
If[Abs[xhi - xlo] < eps, xhi = xlo + eps];
```

Zeros occur when the upper (xmax) and lower (xmin) plot ranges coincide

```
(*Tick tables*)
```

```
xticks = If[
  unit == "Radians", kmin = Floor[2  $\frac{xlo}{\pi}$ ];
  kmax = Ceiling[2  $\frac{xhi}{\pi}$ ];
  Table[{k  $\frac{\pi}{2}$ ,
    If[k == 0, "0", TraditionalForm[k  $\frac{\pi}{2}$ ]]}, {k, kmin, kmax}],
```

Creating plot tick labels

```
(*Degree labeling*)
```

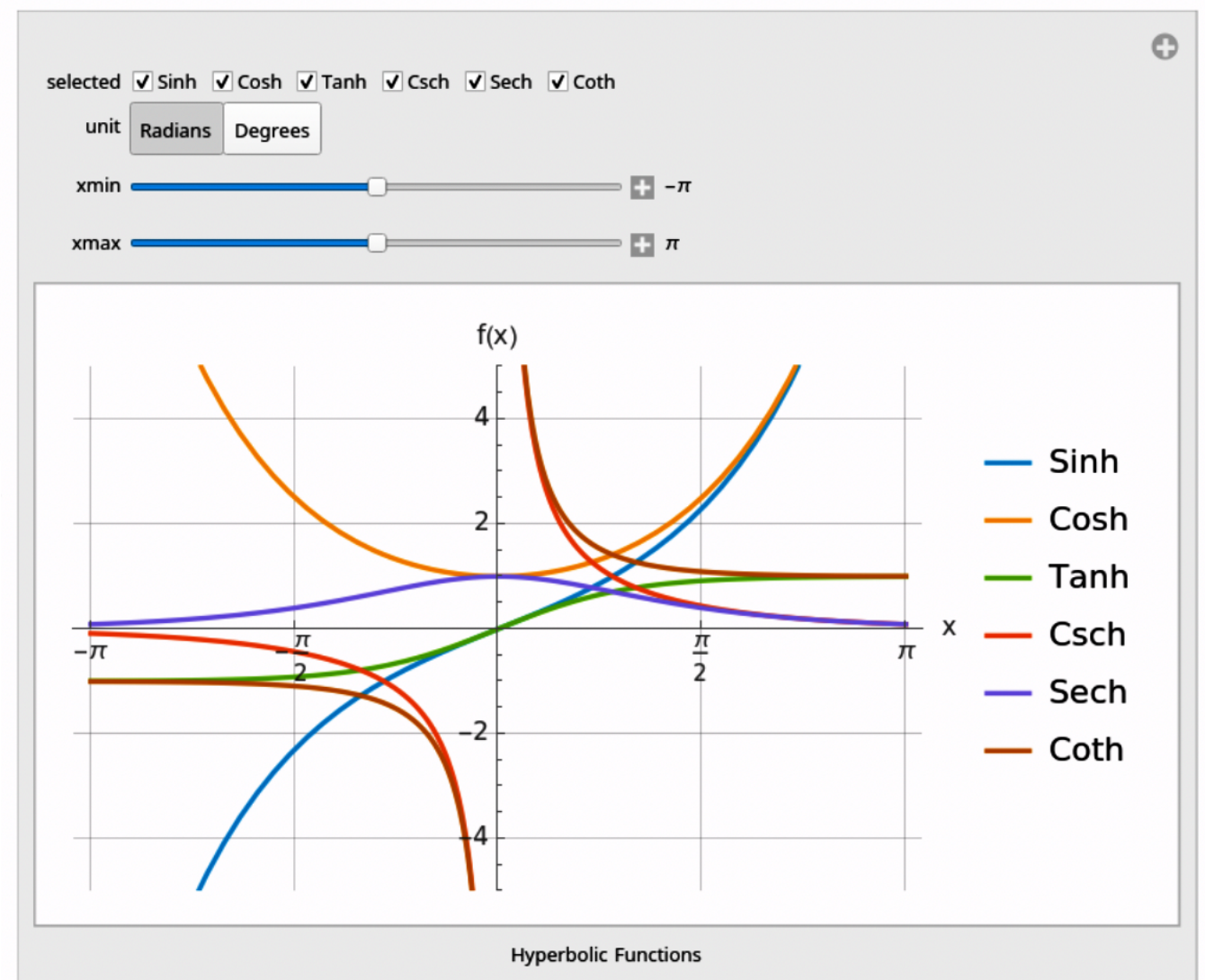
```
kmin = Floor[ $\frac{xlo * \frac{180}{\pi}}{90}$ ] * 90;
kmax = Ceiling[ $\frac{xhi * \frac{180}{\pi}}{90}$ ] * 90;
```

```
Table[{k Degree, ToString[k] <> "°"}, {k, kmin, kmax, 90}]]];
```

```
Plot[Evaluate@Table[f[x], {f, selectedFuncs}], {x, xlo, xhi},
  PlotRange → {-5, 5},
  PlotLegends → selected,
  AxesLabel → {"x", "f(x)"},
  Ticks → {xticks, Automatic},
  GridLines → Automatic]
```

(*Controls*)

```
{{selected, {"Sinh", "Cosh"}},
 {"Sinh", "Cosh", "Tanh", "Csch", "Sech", "Coth"},
 ControlType → Checkbar},
{{unit, "Radians"}, {"Radians", "Degrees"}},
{{xmin, - $\pi$ }, -2  $\pi$ , 0, Appearance → "Labeled"},
{{xmax,  $\pi$ }, 0, 2  $\pi$ , Appearance → "Labeled"},
FrameLabel → "Hyperbolic Functions"]
```



Piecewise/Discontinuous Functions

A Pulse

```
ClearAll[f, t, pw] (* get into the habit of clearing the variables*)
```

```
f[t_] = Piecewise[{{1, -1 < t < 1}}];
```

Values not specified are assumed to be zero.

```
pw = 2;
```

```
Plot[f[t], {t, -2, 2}, AxesLabel → {"t", "f(t)"}, PlotLabel → "Pulse",  
PlotStyle → {Thick, Red}, Exclusions → None,
```

```
Epilog → {
```

```
Arrow[{{-2, 0.4}, {-1, 0.4}}], (* insert a right pointing arrow *)
```

```
Arrow[{{2, 0.4}, {1, 0.4}}], (* insert a left pointing arrow *)
```

```
Inset[pw, {0.9, 0.4}], (* insert a calculation *)
```

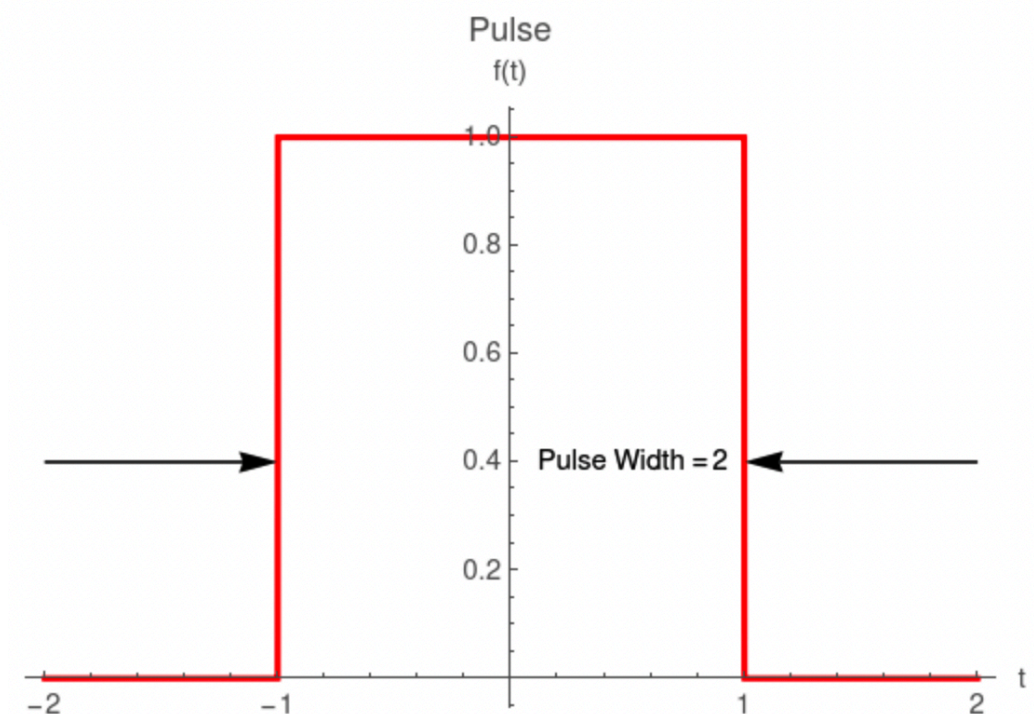
```
Text["Pulse Width = ", {0.5, 0.4}] (* insert text *)
```

```
}
```

```
]
```

Use [Epilog](#) to add graphic elements to a plot.

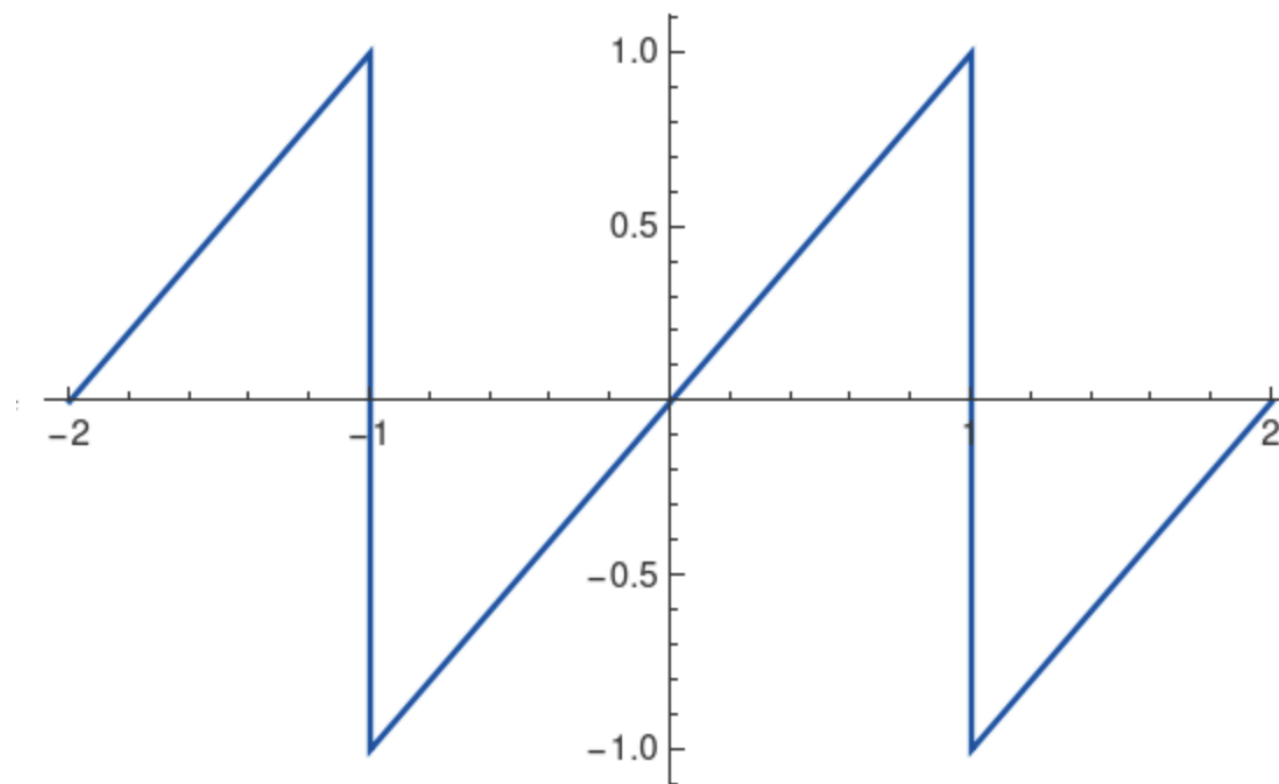
Use **Exclusions**→**None** to draw the leading and trailing edges.



Ramp Function

The [ListPlot](#) command can be used for discontinuous functions.

```
Clear[an, bn, L, number]  
L = 1;  
ListPlot[{{-2, 0}, {-1, 1}, {-1, -1}, {1, 1}, {1, -1}, {2, 0}},  
Joined → True]
```



Arbitrary Function

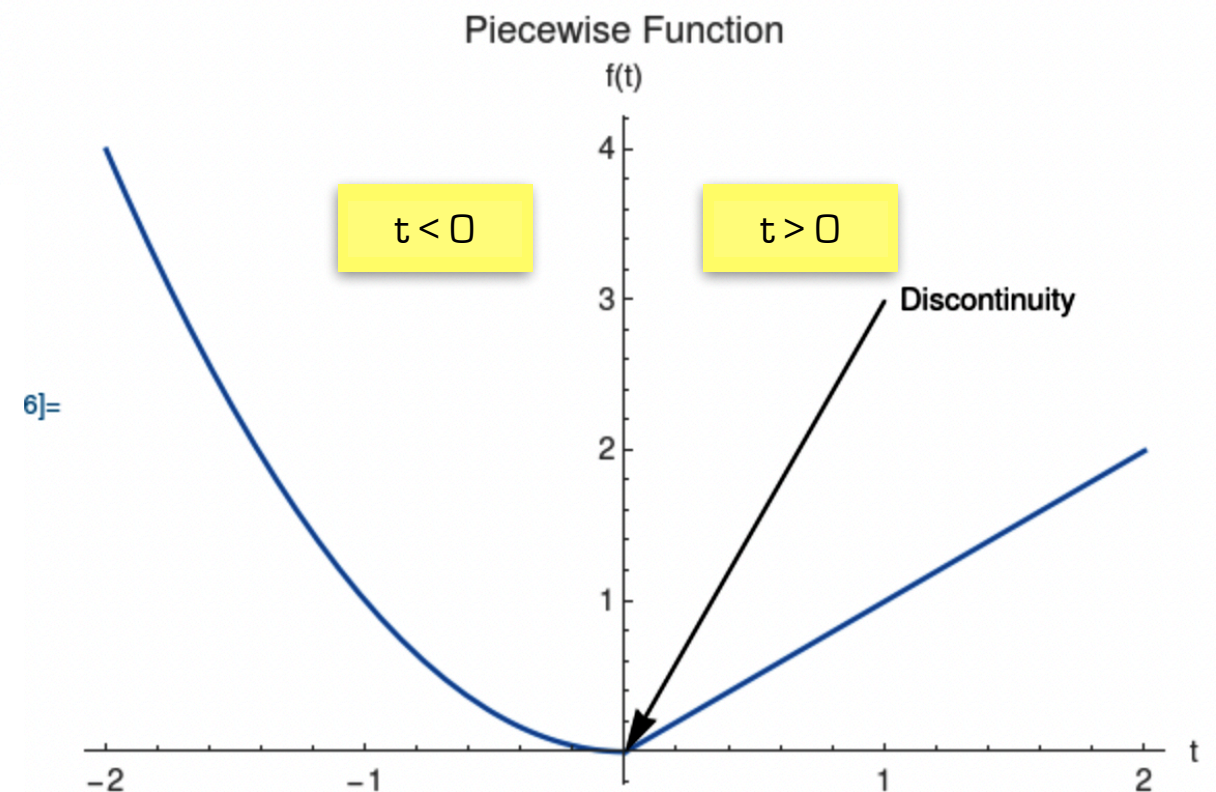
```
ClearAll[f, t]
```

```
f[t_] = Piecewise[{{t^2, t < 0}, {t, t > 0}}] (*Two pieces*)
```

```
Plot[f[t], {t, -2, 2},  
  AxesLabel → {"t", "f(t)"}, PlotLabel → "Piecewise Function",
```

```
  Epilog → {  
    Arrow[{{1, 3}, {0, 0}}], (* insert arrow *)  
    Text["Discontinuity", {1.4, 3}] (* insert text *)  
  }  
]
```

The [Piecewise](#) statement can be embedded into the [Plot](#) command.



Arbitrary Function

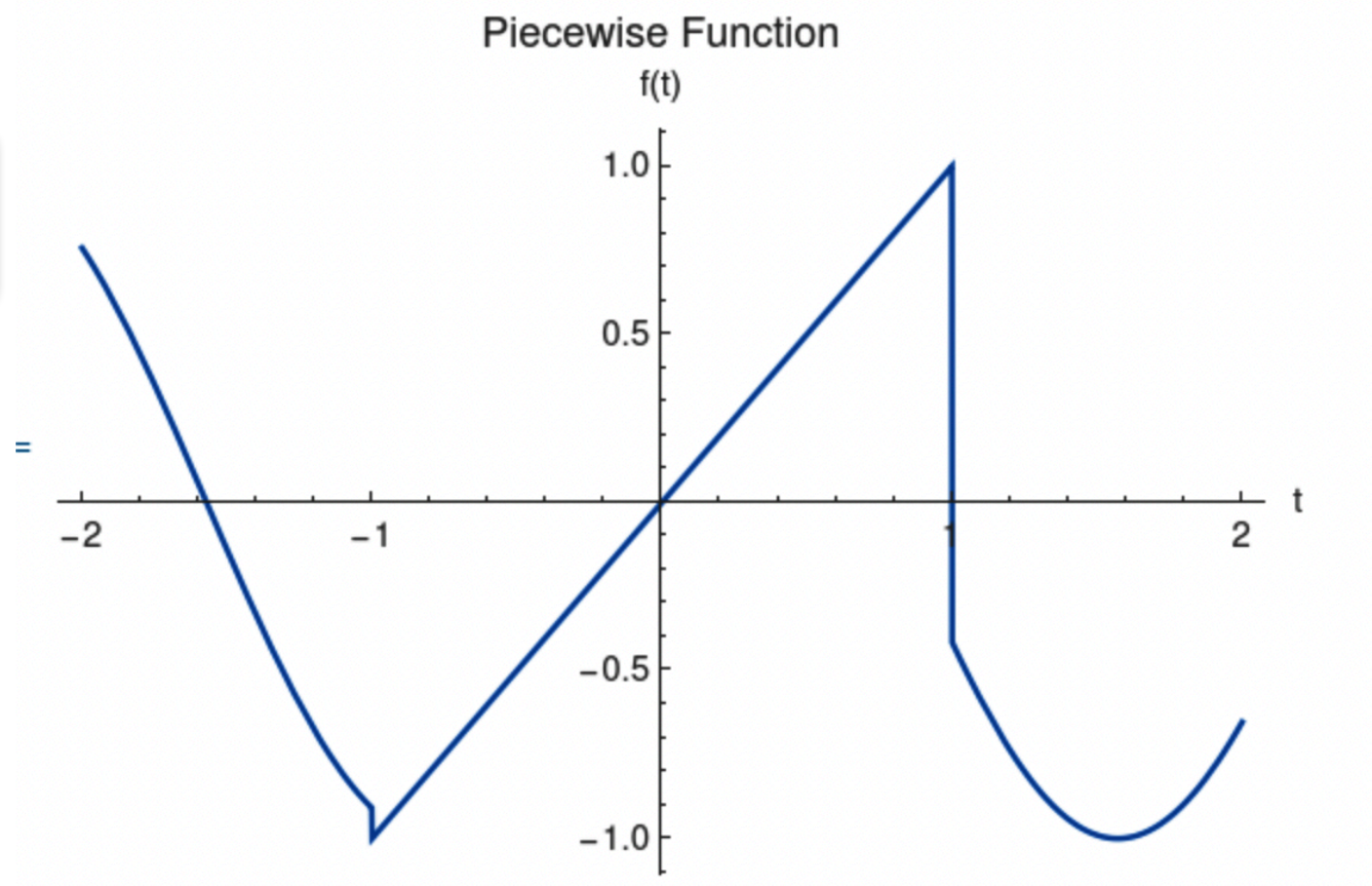
```
ClearAll[t]
```

```
Piecewise[{{Sin[2 t], t < -1}, {t, t < 1}, {Cos[2 t], t > 1}}];  
(*Three pieces*)
```

```
Plot[%, {t, -2, 2},  
     Exclusions -> None,  
     AxesLabel -> {"t", "f(t)"}, PlotLabel -> "Piecewise Function"]
```

% means the last result

Use Exclusions->None, to show the discontinuity



Graphics Grid

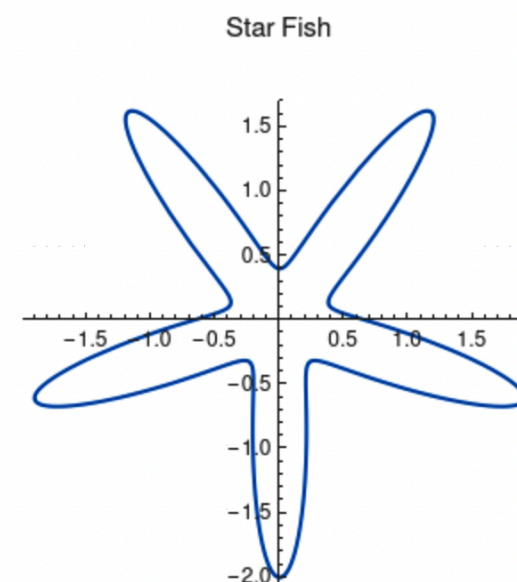
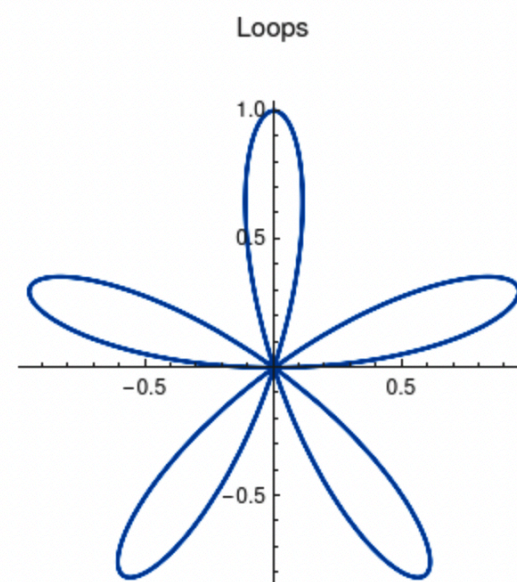
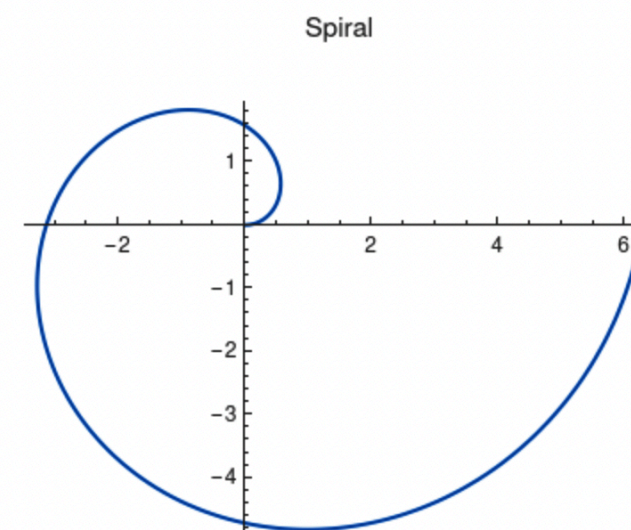
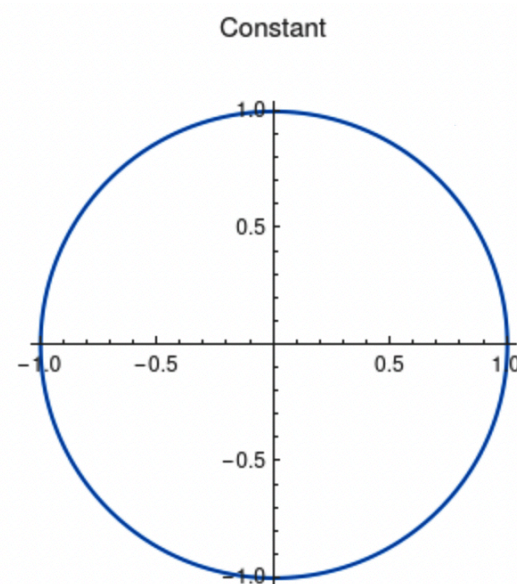
Several plots can be placed in a grid

```
GraphicsGrid[{
  {PolarPlot[1, {θ, 0, 2 π}, (* Row 1 *)
    PlotLabel → "Constant",

    PolarPlot[θ, {θ, 0, 2 π},
      PlotLabel → "Spiral"]},

  {PolarPlot[Sin[5 θ], {θ, 0, 2 π}, (* Row 2 *)
    PlotLabel → "Loops",

    PolarPlot[ $\frac{1}{(1.5 + \sin[5 \theta])}$ , {θ, 0, 2 π},
      PlotLabel → "Star Fish"]}
}]
```

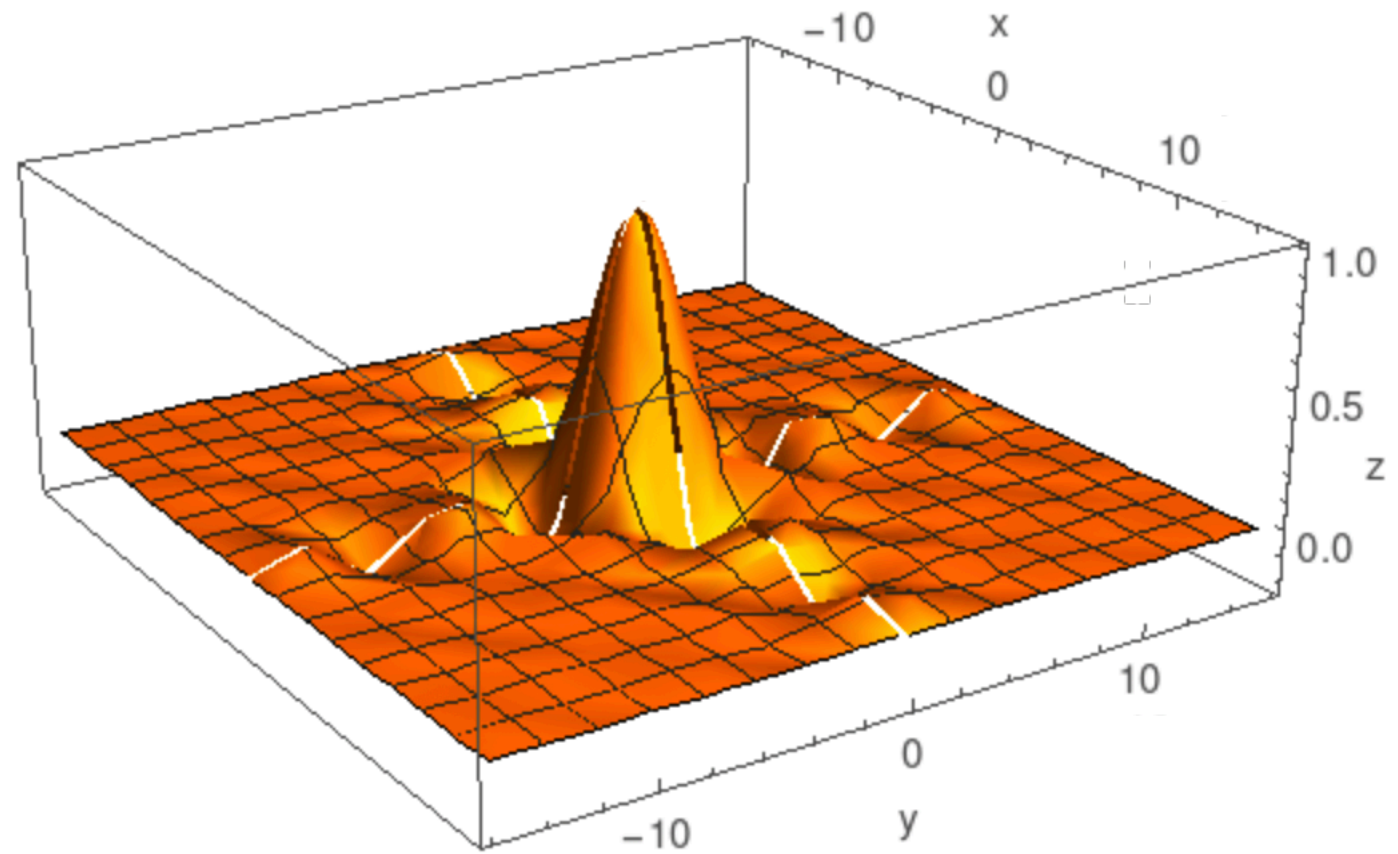


3D Plot

```
Clear[x, y, z]
```

```
f[z_] =  $\frac{\text{Sin}[x]}{x} \frac{\text{Sin}[y]}{y}$ ;
```

```
Plot3D[f[z], {x, -5  $\pi$ , 5  $\pi$ }, {y, -5  $\pi$ , 5  $\pi$ },  
PlotRange -> All,  
AxesLabel -> {"x", "y", "z"}]
```



Fourier Transform



$$F\{f(t)\} \equiv F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t} dt = \int_{-\infty}^{\infty} f(t)\cos(\omega t)dt - j \int_{-\infty}^{\infty} f(t)\sin(\omega t)dt$$

$$F\{f(t)_{\text{even}}\} \equiv F(\omega)_{\text{real}} = \int_{-\infty}^{\infty} f(t)\cos(\omega t)dt$$

$$F\{f(t)_{\text{odd}}\} \equiv F(\omega)_{\text{imag}} = -j \int_{-\infty}^{\infty} f(t)\sin(\omega t)dt$$

Piecewise Continuous [Fourier Transforms](#)

$$F\{f(t)\} \equiv F(\omega) = a_0 + \sum_{n=1}^{\infty} \left[a_n \cos\left(\frac{2\pi n t}{T}\right) + b_n \sin\left(\frac{2\pi n t}{T}\right) \right]$$

$$a_0 = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)dt$$

$$a_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\cos\left(\frac{2\pi n t}{T}\right)dt$$

$$b_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\sin\left(\frac{2\pi n t}{T}\right)dt$$

[DFT](#) - Discrete Fourier Transform
for repetitive waveforms

[Fourier Transforms using Mathematica by Goodman](#)

FourierParameters

[FourierParameters](#) and their applications

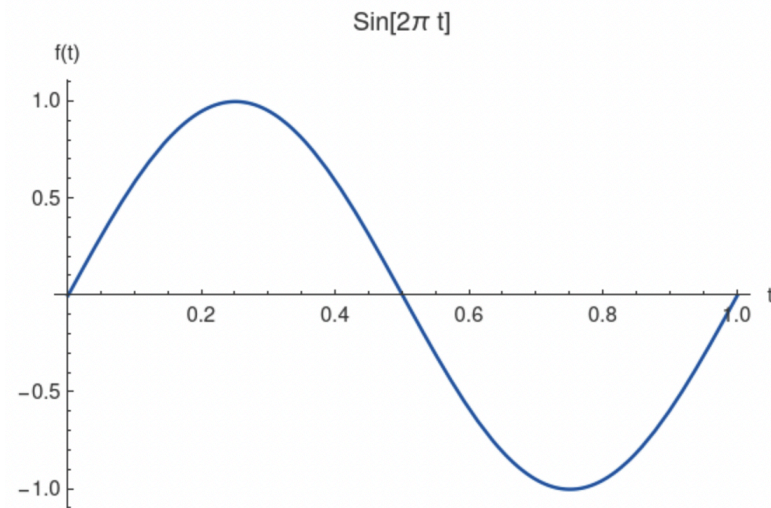
<i>common convention</i>	<i>setting</i>	<i>Fourier transform</i>	<i>inverse Fourier transform</i>
<i>Mathematica</i> default	{0, 1}	$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt$	$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega) e^{-i\omega t} d\omega$
pure mathematics	{1, -1}	$\int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$	$\frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega$
classical physics	{-1, 1}	$\frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt$	$\int_{-\infty}^{\infty} F(\omega) e^{-i\omega t} d\omega$
modern physics	{0, 1}	$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt$	$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega) e^{-i\omega t} d\omega$
systems engineering	{1, -1}	$\int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$	$\frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega$
signal processing	{0, -2 Pi}	$\int_{-\infty}^{\infty} f(t) e^{-2\pi i\omega t} dt$	$\int_{-\infty}^{\infty} F(\omega) e^{2\pi i\omega t} d\omega$
general case	{a, b}	$\sqrt{\frac{ b }{(2\pi)^{1-a}}} \int_{-\infty}^{\infty} f(t) e^{ib\omega t} dt$	$\sqrt{\frac{ b }{(2\pi)^{1+a}}} \int_{-\infty}^{\infty} F(\omega) e^{-ib\omega t} d\omega$

Typical settings for `FourierParameters` with various conventions.

It is essential to choose the correct `FourierParameters`

Sine Wave

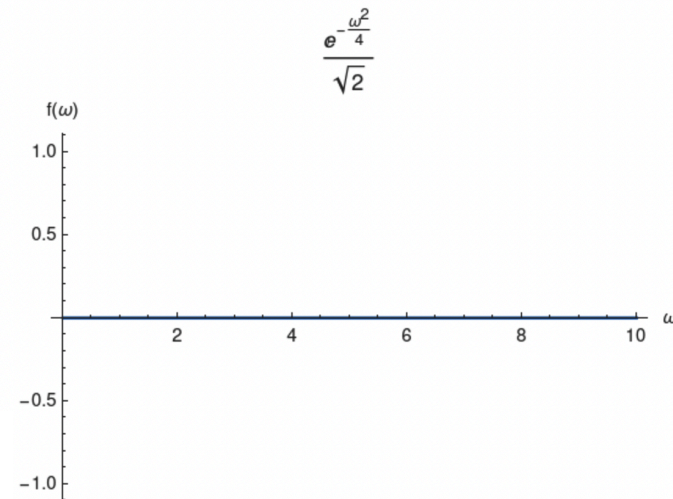
```
fcn = Sin[2 π t];
Plot[fcn, {t, 0, 1},
  PlotRange → All,
  PlotLabel → "Sin[2π t]",
  AxesLabel → {"t", "f(t)"}]
```



```
ftfcn = FourierTransform[fcn, t, ω,
  FourierParameters → {0, -2 π}]
-1/2 i DiracDelta[-1 + ω] + 1/2 i DiracDelta[1 + ω]
```

A result containing the DiracDelta function cannot be easily plotted

```
Plot[ftfcn, {ω, 0, 10},
  PlotRange → All,
  Exclusions → None,
  PlotLabel → " $\frac{e^{-\frac{\omega^2}{4}}}{\sqrt{2}}$ ",
  AxesLabel → {"ω", "f(ω)"}]
```

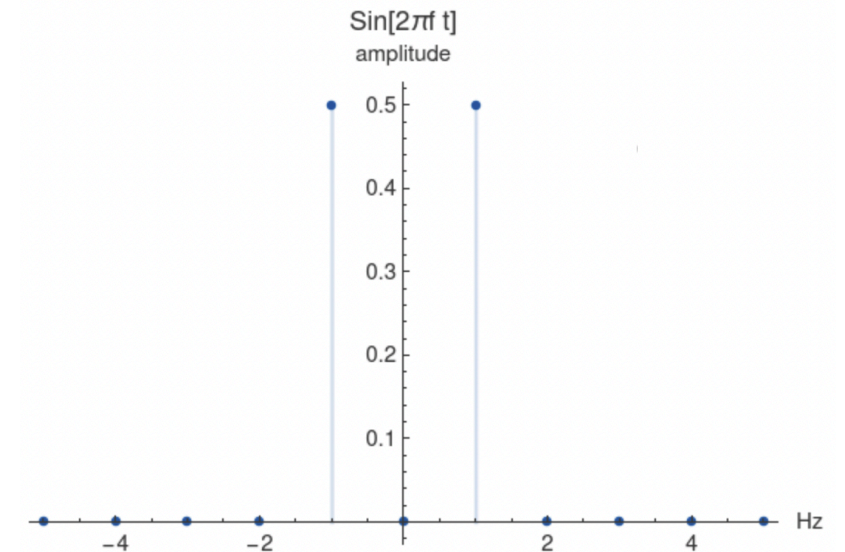


In this case, Mathematica produces an answer suitable for a math student, but not very helpful for an engineer.

```
fcfcn = FourierCoefficient[fcn2, t, n,
  FourierParameters → {0, -2 π}]
```

```
{-1/2, n == -1}
{1/2, n == 1}
{0, True}
```

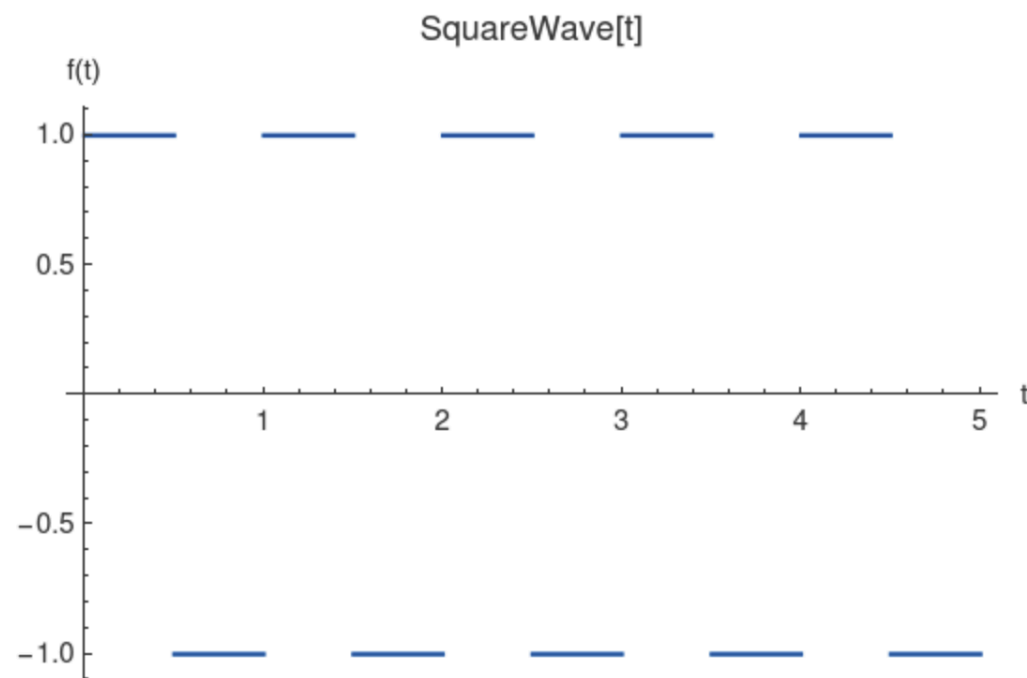
```
DiscretePlot[Abs[%], {n, -5, 5},
  PlotRange → All,
  PlotLabel → "Sin[2πf t]",
  AxesLabel → {"Hz", "amplitude"}]
```



It may be more meaningful to calculate the Fourier Coefficient and plot the absolute value of the result.

Square Wave

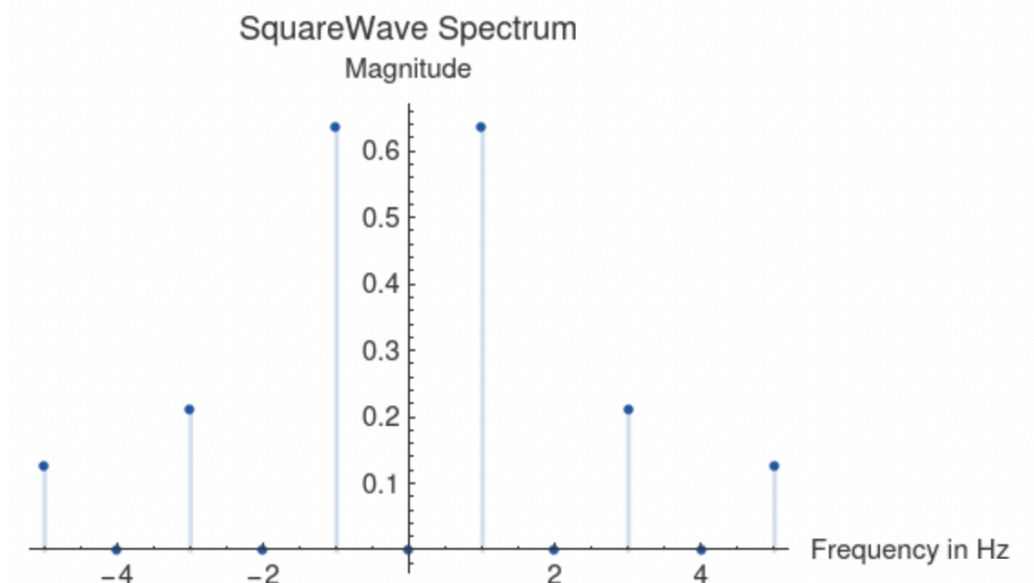
```
fcn = SquareWave[t];  
Plot[fcn, {t, 0, 5},  
  PlotRange → All,  
  PlotLabel → "SquareWave[t]",  
  AxesLabel → {"t", "f(t)"}]
```



```
fcfcn = FourierCoefficient[fcn, t, n,  
  FourierParameters → {0, -2 π}]
```

$$\begin{cases} 0 & n == 0 \\ \frac{i (-1)^n (-1 + (-1)^n)}{n \pi} & \text{True} \end{cases}$$

```
DiscretePlot[Abs[%], {n, -5, 5},  
  PlotRange → All,  
  PlotLabel → "SquareWave Spectrum",  
  AxesLabel → {"Frequency in Hz", "Magnitude"}]
```

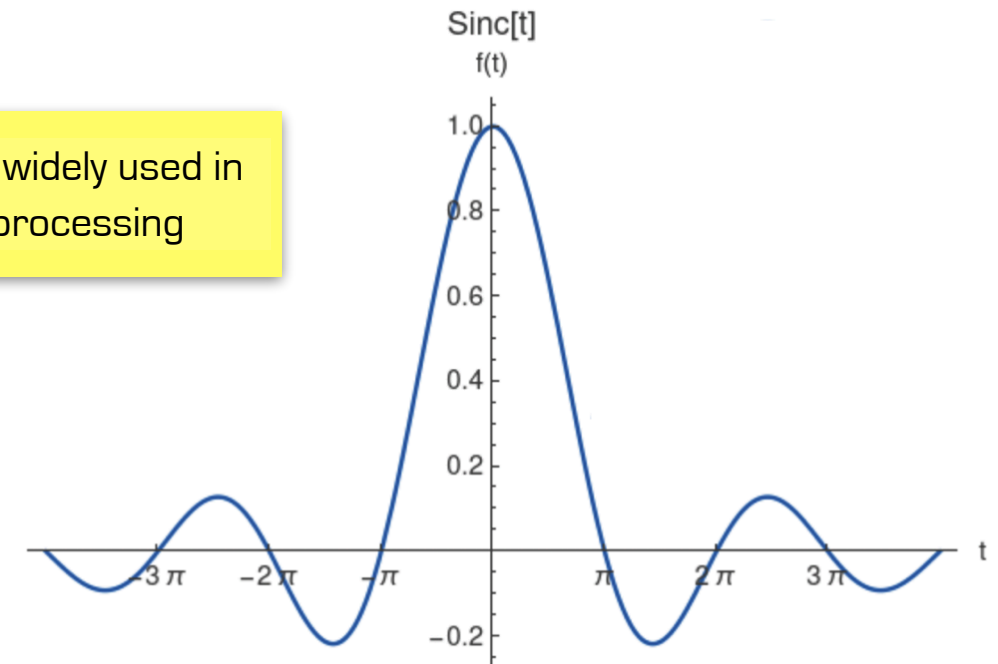


In this case, using the Fourier Transform command also results in the Delta function

Sinc Pulse

```
Clear [fcf, ftfcf1, ftfcf2, ftfcf3, ftfcf4]
fcf = Sinc[t];
Plot[fcf, {t, -4 π, 4 π},
  PlotRange → All,
  Ticks → {{-3 π, -2 π, -π, π, 2 π, 3 π}, Automatic},
  PlotLabel → "Sinc[t]",
  AxesLabel → {"t", "f(t)"}]
```

The [sinc filter](#) is widely used in digital signal processing



```
ftfcf1 = FourierTransform[fcf, t, ω, FourierParameters → {0, 1}]
ftfcf2 = FourierTransform[fcf, t, ω, FourierParameters → {1, -1}]
ftfcf3 = FourierTransform[fcf, t, ω, FourierParameters → {-1, 1}]
ftfcf4 = FourierTransform[fcf, t, ω, FourierParameters → {0, -2 π}]
```

It is essential to choose the correct FourierParameters

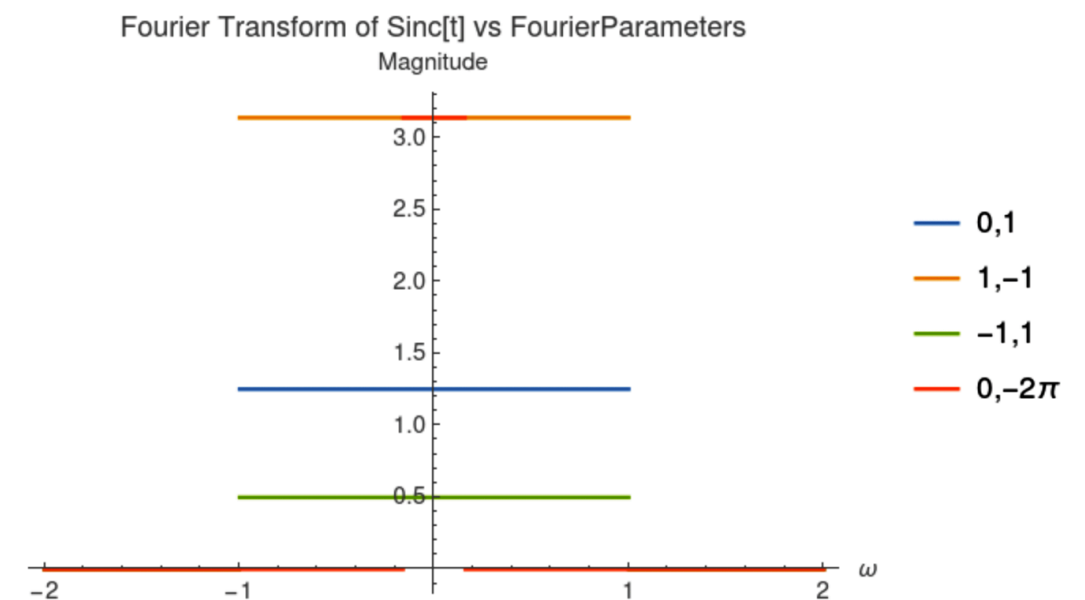
$$\frac{1}{2} \sqrt{\frac{\pi}{2}} (\text{Sign}[1 - \omega] + \text{Sign}[1 + \omega])$$

$$\frac{1}{2} \pi \text{Sign}[1 - \omega] + \frac{1}{2} \pi \text{Sign}[1 + \omega]$$

$$\frac{1}{4} \text{Sign}[1 - \omega] + \frac{1}{4} \text{Sign}[1 + \omega]$$

$$\frac{1}{2} \pi \text{Sign}[1 - 2 \pi \omega] + \frac{1}{2} \pi \text{Sign}[1 + 2 \pi \omega]$$

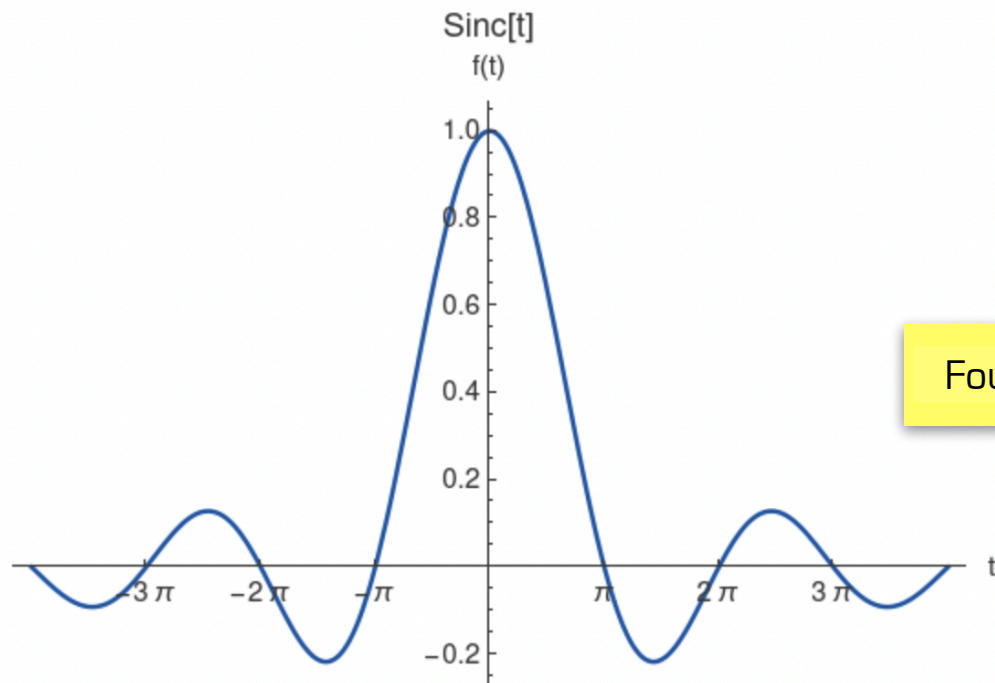
```
Plot[{ftfcf1, ftfcf2, ftfcf3, ftfcf4}, {ω, -2, 2},
  PlotLabel → "Fourier Transform of Sinc[t] vs FourierParameters",
  PlotLegends → {"0,1", "1,-1", "-1,1", "0,-2π"},
  AxesLabel → {"ω", "Magnitude"}]
```



FourierTransform Function

Sinc[t]

```
Clear [fcfn, ftfcfn]
fcfn = Sinc[t];
Plot[fcfn, {t, -4 π, 4 π},
  PlotRange → All,
  Ticks → {{-3 π, -2 π, -π, π, 2 π, 3 π}, Automatic},
  PlotLabel → "Sinc[t]",
  AxesLabel → {"t", "f(t)"}]
```



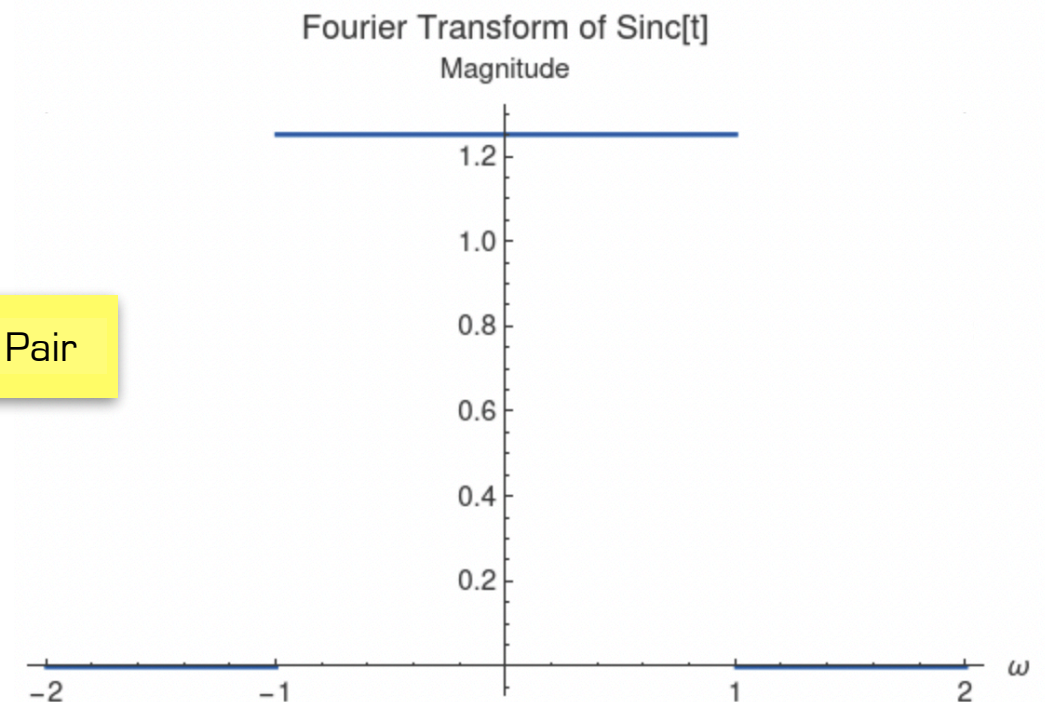
Fourier Transform Pair

Using the default
FourierParameters {0,1}

```
ftfcfn = FourierTransform[fcfn, t, ω]
```

$$\frac{1}{2} \sqrt{\frac{\pi}{2}} (\text{Sign}[1 - \omega] + \text{Sign}[1 + \omega])$$

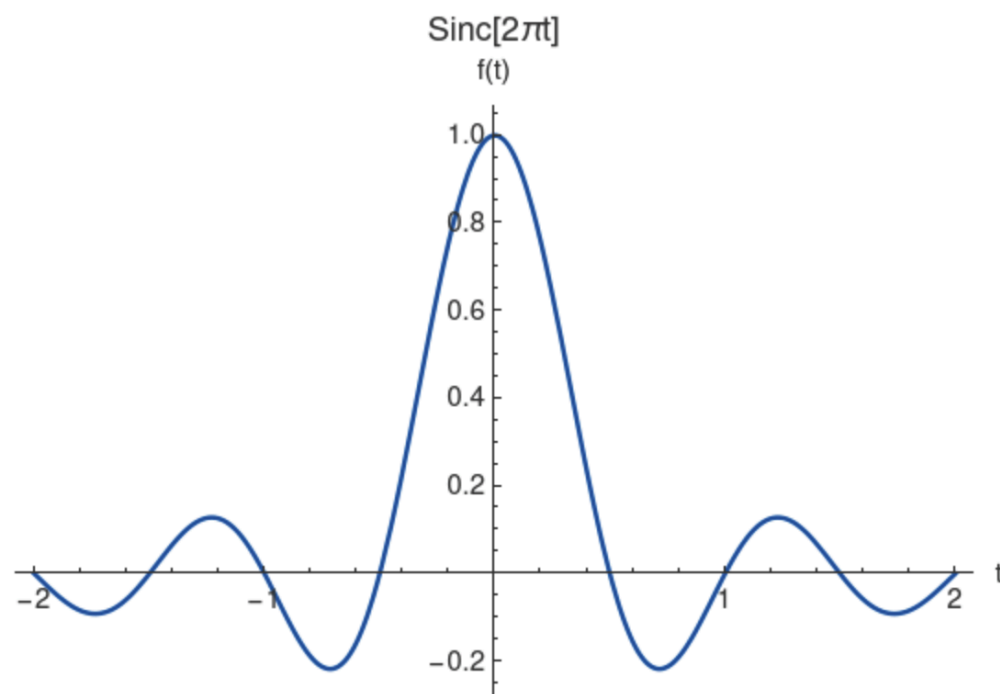
```
Plot[%, {ω, -2, 2},
  PlotLabel → "Fourier Transform of Sinc[t]",
  AxesLabel → {"ω", "Magnitude"}]
```



A slight change in constants can significantly alter the outcome.

Sinc[2πt]

```
Clear[fcn, ftfcn]
fcf = Sinc[2 π t];
Plot[fcf, {t, -2, 2},
  PlotRange → All,
  PlotLabel → "Sinc[2πt]",
  AxesLabel → {"t", "f(t)"}]
```

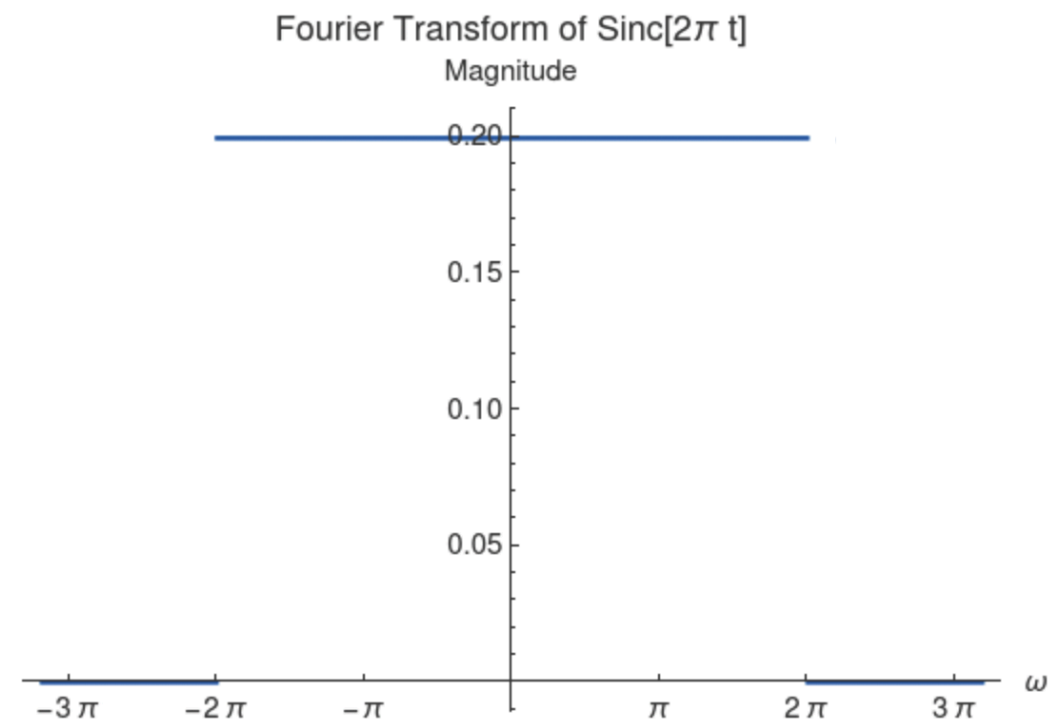


Using the default
FourierParameters {0,1}

```
ftfcn = FourierTransform[fcf, t, ω]
```

$$\frac{\frac{1}{2} \sqrt{\frac{\pi}{2}} \text{Sign}[2\pi - \omega] + \frac{1}{2} \sqrt{\frac{\pi}{2}} \text{Sign}[2\pi + \omega]}{2\pi}$$

```
Plot[%, {ω, -10, 10},
  Ticks → {{-3 π, -2 π, -π, π, 2 π, 3 π}, Automatic},
  PlotLabel → "Fourier Transform of Sinc[2π t]",
  AxesLabel → {"ω", "Magnitude"}]
```



DFT & Fourier Coefficients

The Fourier Coefficient equations can be written in many forms, each to suit a particular application.

$$\begin{aligned} &\text{Fourier series coefficients} \\ A_0 &= \frac{1}{P} \int_{-P/2}^{P/2} s(x) dx \\ A_n &= \frac{2}{P} \int_{-P/2}^{P/2} s(x) \cos\left(\frac{2\pi nx}{P}\right) dx \quad \text{for } n \geq 1 \\ B_n &= \frac{2}{P} \int_{-P/2}^{P/2} s(x) \sin\left(\frac{2\pi nx}{P}\right) dx \quad \text{for } n \geq 1 \end{aligned} \quad (\text{Eq. 1})$$

Courtesy: [Wikipedia](#)

Many engineering texts use the variable t and integrate between 0 and T .

$$\begin{aligned} a_0 &= \frac{1}{L} \int_{-L}^L f(x) dx \\ a_m &= \frac{1}{L} \int_{-L}^L f(x) \cos\left(\frac{m\pi x}{L}\right) dx, \text{ and} \\ b_m &= \frac{1}{L} \int_{-L}^L f(x) \sin\left(\frac{m\pi x}{L}\right) dx. \end{aligned}$$

Sine Wave DFT

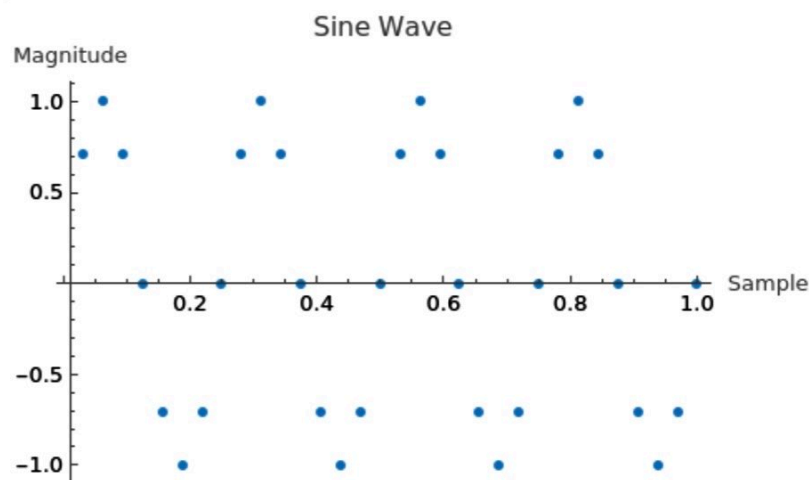
```
Clear[n, nos, f, signal]
f = 4; (* frequency in Hz *)
sps = 8; (* samples per second *)
nos = f sps; (* number of samples *)
time =  $\frac{t}{nos}$ ; (* time increment *)

signalTable = Table[{time, Sin[ $\frac{2 \pi f t}{nos}$ ]}, {t, nos}];

TableForm[%,
  TableHeadings → {None, {"t", "Window", "time", "f(t)"}},

timeWindow = signalTable[[All, 1]]; (* select col 1 *)
amplitude = signalTable[[All, 2]]; (* select col 2 *)
data = Transpose[{timeWindow, amplitude}];
(* arrange data - {x,y} *)
```

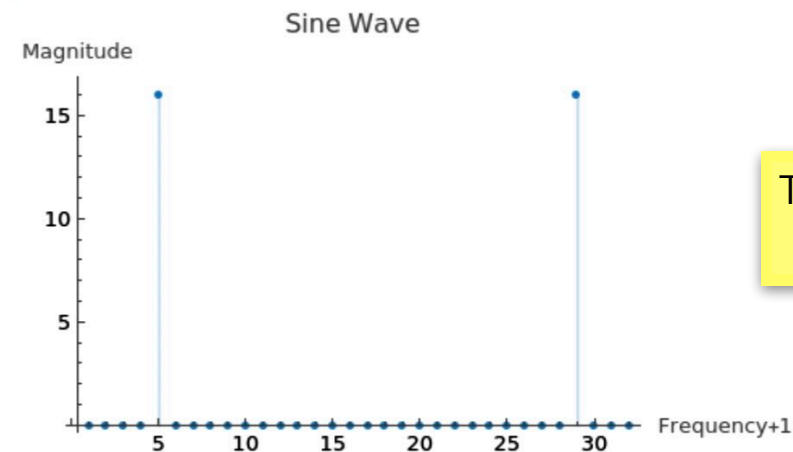
```
ListPlot[data,
  PlotRange → All,
  PlotLabel → "Sine Wave",
  AxesLabel → {"Sample", "Magnitude"}]
```



Digitizing a 4 Hz Sine Wave

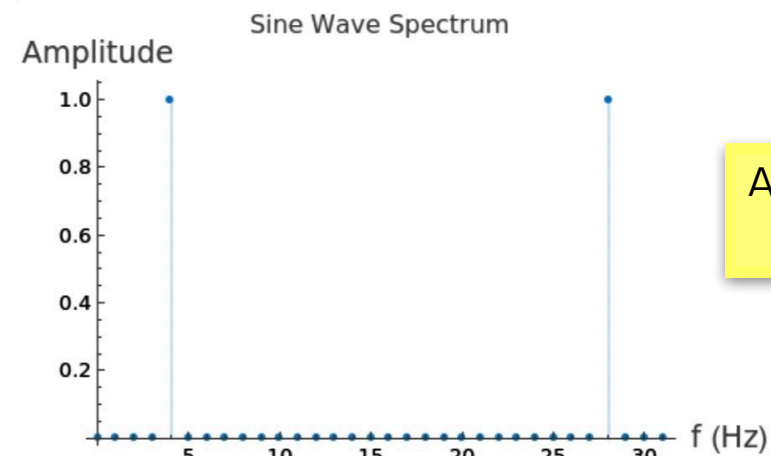
```
Clear[dftsignal]

dftsignal = Fourier[signalTable[[All, 2]],
  FourierParameters → {1, -1}];
ListPlot[Abs[dftsignal],
  PlotRange → All,
  Filling → Axis,
  PlotLabel → "Sine Wave",
  AxesLabel → {"Frequency+1", "Magnitude"}]
```



The DFT samples start at 1 and not 0.

```
ftsignaltable = Table[{n - 1,  $\frac{2}{nos}$  Abs[dftsignal[[n]]]},
  {n, 1, nos}];
ListPlot[ftsignaltable,
  PlotLabel → "Sine Wave Spectrum",
  PlotRange → All,
  Filling → Axis,
  AxesLabel → {"f (Hz)", "Amplitude"}]
```

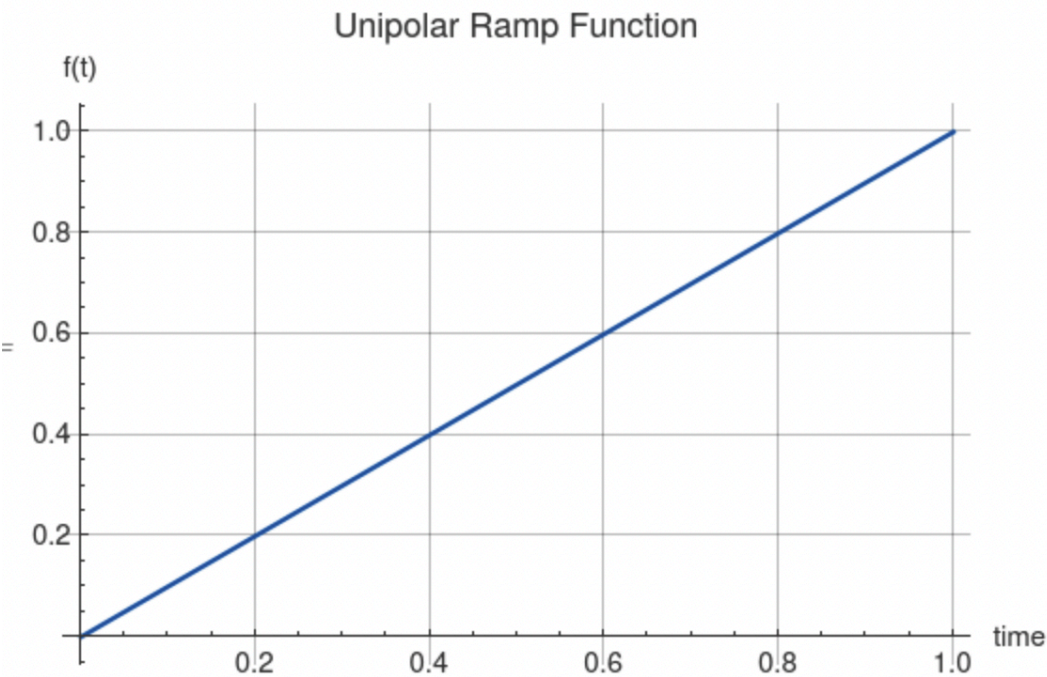


Adjust the amplitude and frequency

Unipolar Ramp

Plot

```
f[t_] = t; T = 1;
Plot[f[t], {t, 0, T},
  PlotLabel -> "Unipolar Ramp Function",
  AxesLabel -> {"time", "f(t)"},
  GridLines -> Automatic]
```



The ramp resets at T = 1

Calculating and tabulating the coefficients

$$a_0 = \frac{1}{T} \int_0^T f[t] \, dt;$$
$$a_n = \frac{2}{T} \int_0^T f[t] \cos\left[\frac{n 2 \pi t}{T}\right] \, dt;$$
$$b_n = \frac{2}{T} \int_0^T f[t] \sin\left[\frac{n 2 \pi t}{T}\right] \, dt;$$

```
Table[{n, a0, an, bn}, {n, 1, 8, 1}];
TableForm[%,
  TableHeadings -> {"Coefficients"}, {"n", "a0", "an", "bn"}]]
```

eForm=

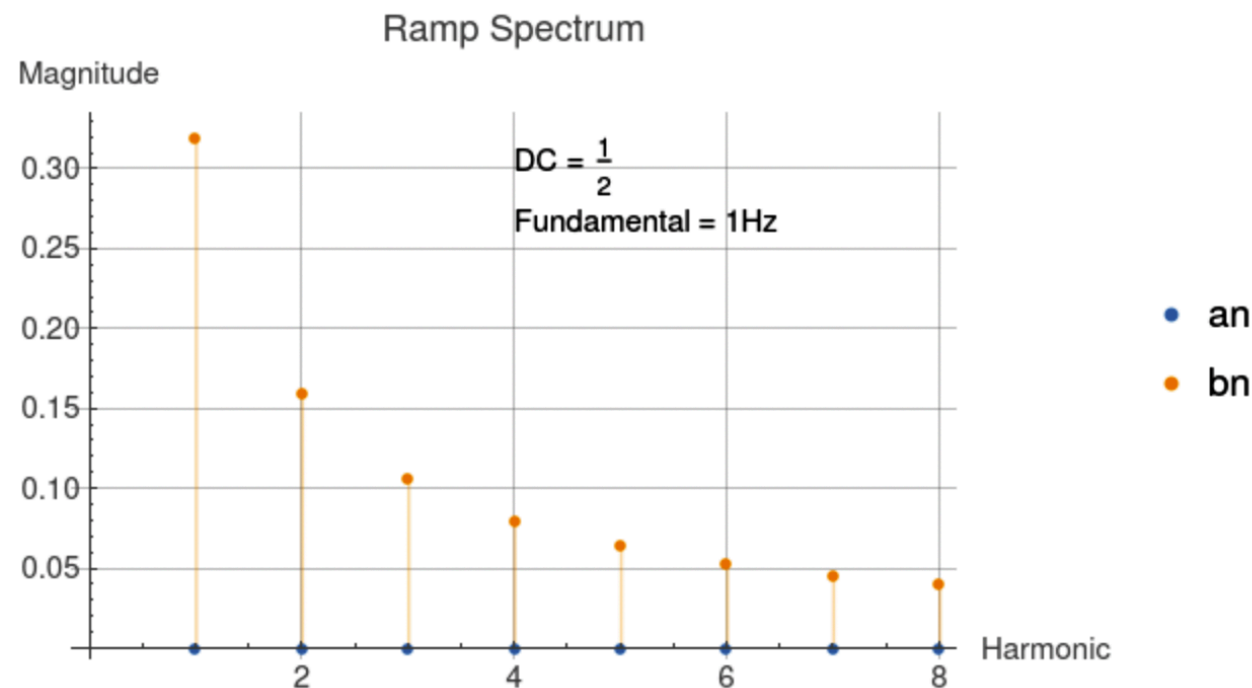
	n	a0	an	bn
Coefficients	1	$\frac{1}{2}$	0	$-\frac{1}{\pi}$
	2	$\frac{1}{2}$	0	$-\frac{1}{2 \pi}$
	3	$\frac{1}{2}$	0	$-\frac{1}{3 \pi}$
	4	$\frac{1}{2}$	0	$-\frac{1}{4 \pi}$
	5	$\frac{1}{2}$	0	$-\frac{1}{5 \pi}$
	6	$\frac{1}{2}$	0	$-\frac{1}{6 \pi}$
	7	$\frac{1}{2}$	0	$-\frac{1}{7 \pi}$
	8	$\frac{1}{2}$	0	$-\frac{1}{8 \pi}$

Ramp Spectrum

```
acoeff = Table[an, {n, 1, 8, 1}];
bcoeff = Table[Abs[bn], {n, 1, 8, 1}];
```

```
ListPlot[{acoeff, bcoeff},
  Filling → Axis,
  PlotLabel → "Ramp Spectrum",
  PlotLegends → {"an", "bn"},
  AxesLabel → {"Harmonic", "Magnitude"},
  GridLines → Automatic,
  Epilog → {
    Text[Row[{"DC = ", a0}], Scaled[{0.5, 0.9}], Left],
    Text[Row[{"Fundamental = ",  $\frac{1}{T}$ , "Hz"}], Scaled[{0.5, 0.8}], Left]}]
```

Epilog can be used to place information on the plot



Unipolar Ramp Spectrum

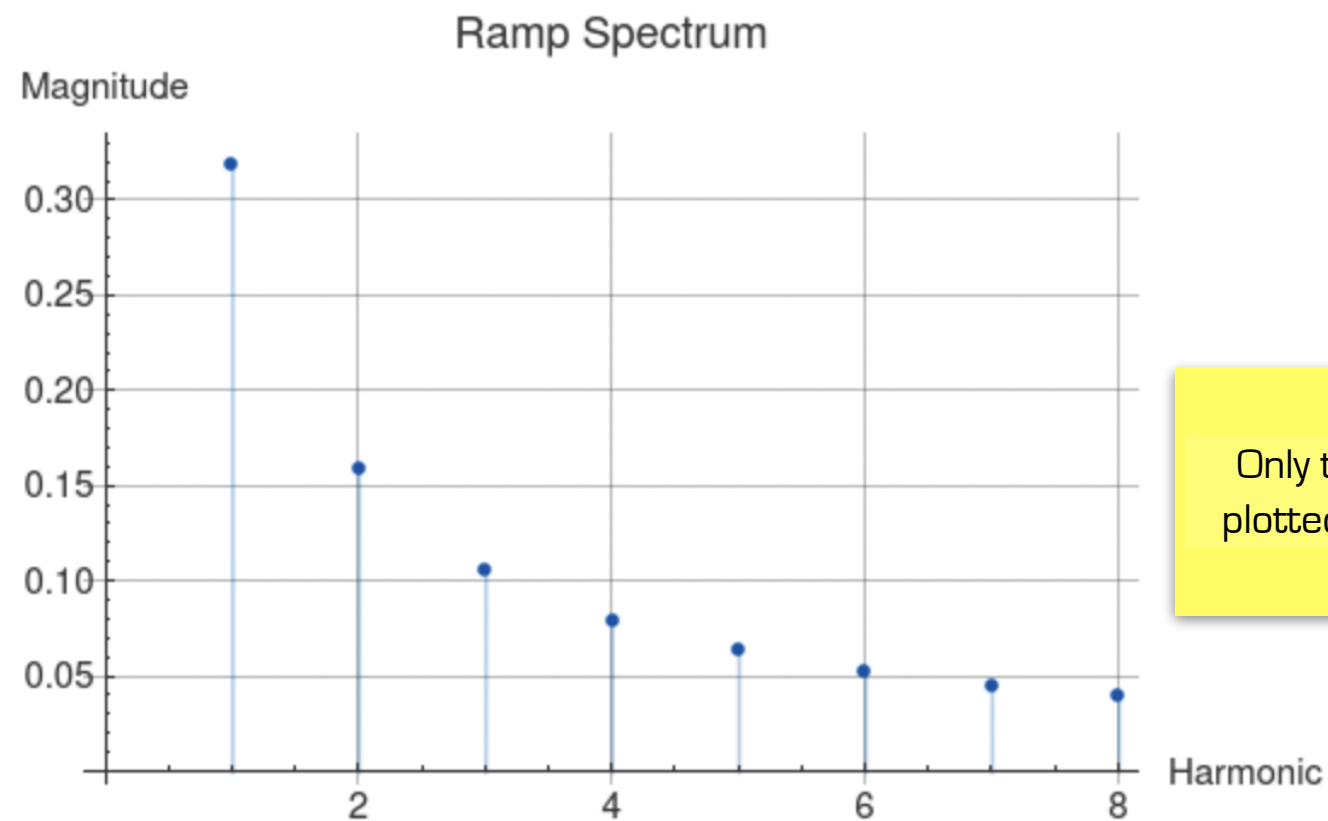
The a coefficients = 0

PlotLegends is used to distinguish the coefficients

Alternative Table

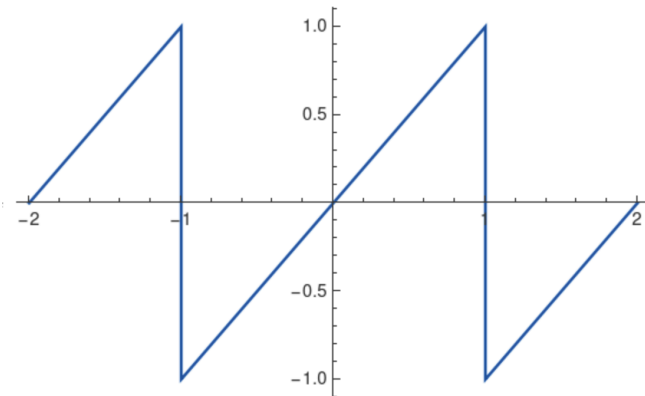
```
RampTable = Table[{n, a0, an, bn}, {n, 1, 8, 1}];  
harm = RampTable[[All, 1]];  
bco = RampTable[[All, 4]];
```

```
ListPlot[RampSpectrum = Transpose[{harm, Abs[bco]}],  
Filling → Axis,  
PlotLabel → "Ramp Spectrum",  
AxesLabel → {"Harmonic", "Magnitude"},  
GridLines → Automatic]
```



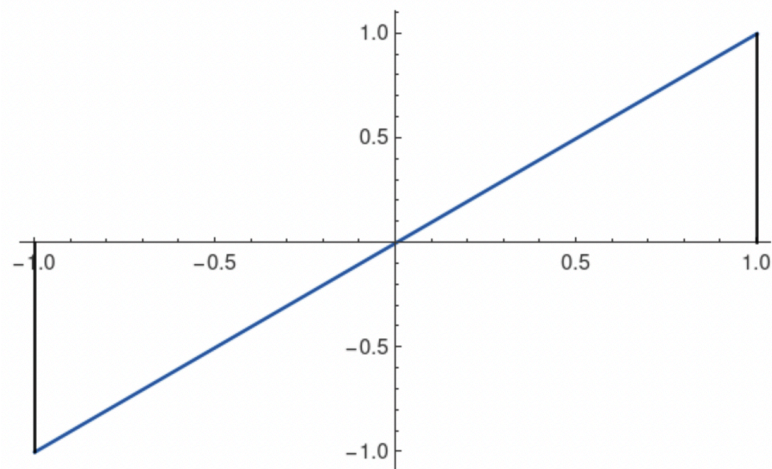
Only the b_n coefficients have been plotted since the a_n values are zero

Calculating the Fourier Coefficients for a Ramp Function



Since this is an odd function, there are only b_n coefficients

```
Clear[f, t, T, a0, an, bn]
f[t_] = t;
Plot[f[t], {t, -1, 1},
  Epilog -> {
    Line[{{-1, 0}, {-1, -1}}],
    Line[{{1, 0}, {1, 1}}]
  }
```



The desired coefficients can be placed in a list.

```
T = 2;
a0 = 1/T * Integrate[f[t], {t, -T/2, T/2}]
an = 2/T * Integrate[f[t] Cos[2 n pi t / T], {t, -T/2, T/2}]
bn = 2/T * Integrate[f[t] Sin[2 n pi t / T], {t, -T/2, T/2}]

In[ ]:= 0
In[ ]:= 0
In[ ]:= -2 n pi Cos[n pi] + 2 Sin[n pi] / (n^2 pi^2)

In[ ]:= Table[bn, {n, 1, 8}]
In[ ]:= {2/pi, -1/pi, 2/(3 pi), -1/(2 pi), 2/(5 pi), -1/(3 pi), 2/(7 pi), -1/(4 pi)}

In[ ]:= FourierCoefficients = Table[{n, bn}, {n, 1, 8}];
TableForm[FourierCoefficients,
  TableHeadings -> {"Fourier Coefficients"}, {"n", "b[n]"}]

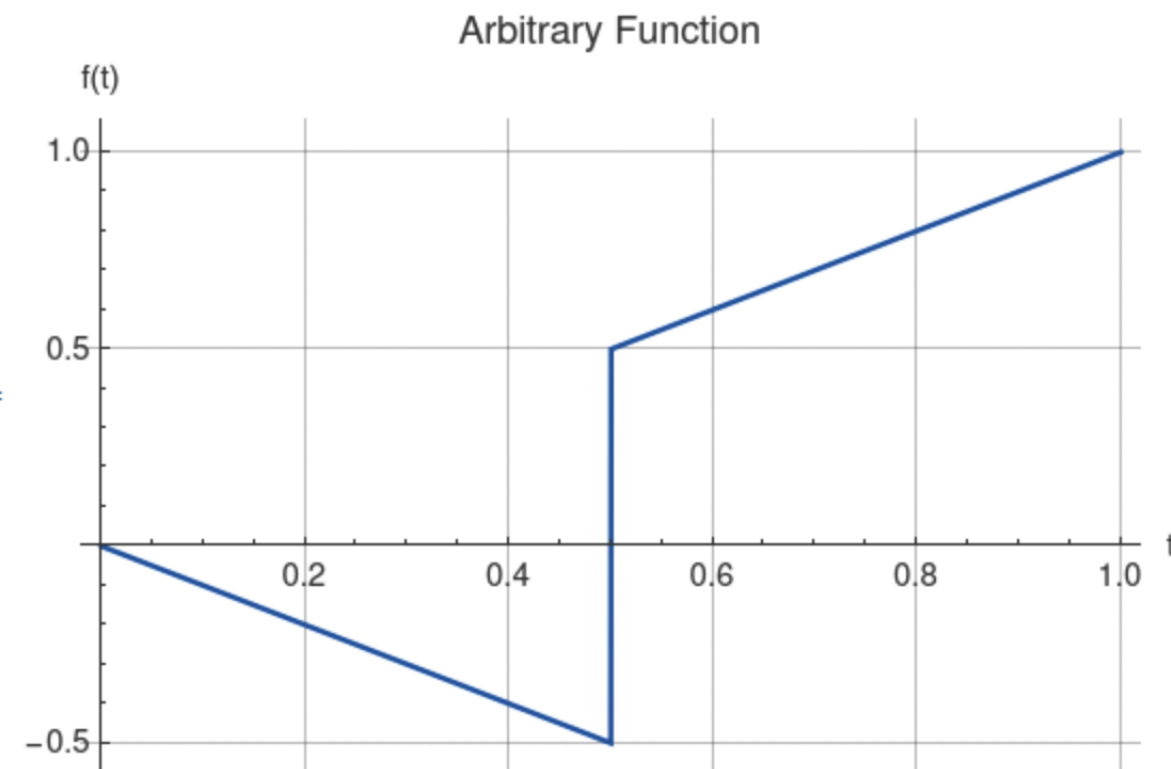
TableForm=
```

	n	b[n]
Fourier Coefficients	1	$\frac{2}{\pi}$
	2	$-\frac{1}{\pi}$
	3	$\frac{2}{3\pi}$
	4	$-\frac{1}{2\pi}$
	5	$\frac{2}{5\pi}$
	6	$-\frac{1}{3\pi}$
	7	$\frac{2}{7\pi}$
	8	$-\frac{1}{4\pi}$

Arbitrary Function

```
f[t_] = Piecewise[{{-t, 0 ≤ t ≤  $\frac{T}{2}$ }, {t,  $\frac{T}{2}$  ≤ t ≤ T}}];
```

```
Plot[f[t], {t, 0, T},  
  Exclusions → None,  
  PlotLabel → "Arbitrary Function",  
  AxesLabel → {"t", "f(t)"},  
  GridLines → Automatic]
```



The function resets at $T = 1$

Table of coefficients

```
Table[{n, a0, an, bn}, {n, 1, 8, 1}];
TableForm[%,
  TableHeadings -> {"Coefficients", {"n", "a0", "an", "bn"}}];
```

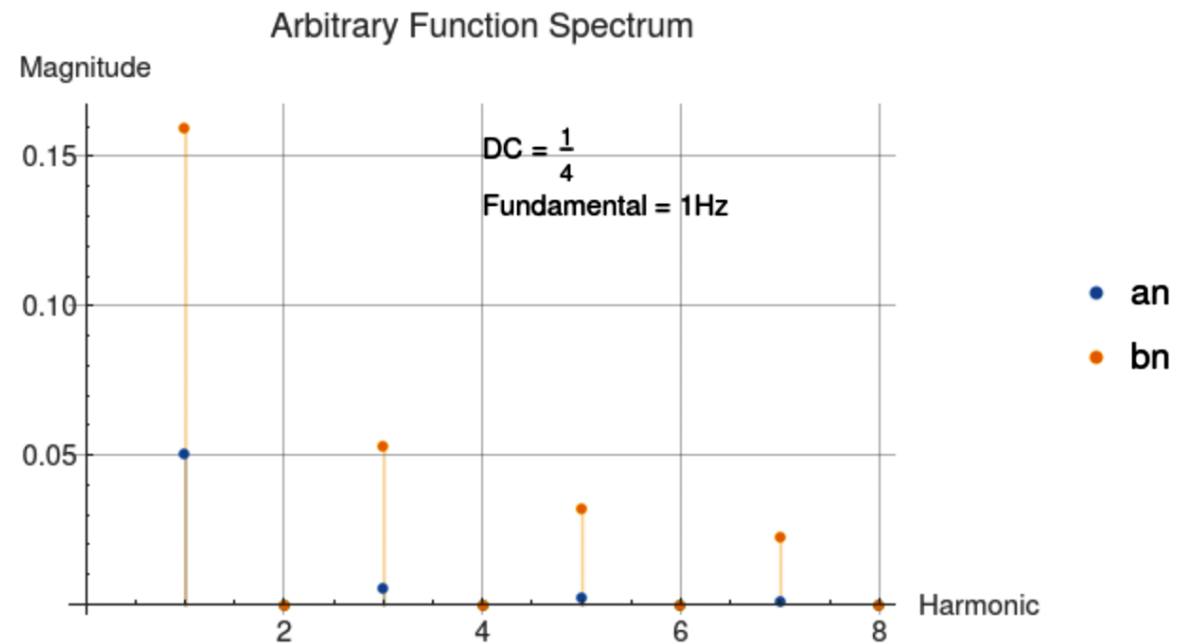
Form=

	n	a0	an	bn
Coefficients	1	$\frac{1}{4}$	$\frac{1}{2\pi^2}$	$-\frac{1}{2\pi}$
	2	$\frac{1}{4}$	0	0
	3	$\frac{1}{4}$	$\frac{1}{18\pi^2}$	$-\frac{1}{6\pi}$
	4	$\frac{1}{4}$	0	0
	5	$\frac{1}{4}$	$\frac{1}{50\pi^2}$	$-\frac{1}{10\pi}$
	6	$\frac{1}{4}$	0	0
	7	$\frac{1}{4}$	$\frac{1}{98\pi^2}$	$-\frac{1}{14\pi}$
	8	$\frac{1}{4}$	0	0

Spectrum

```
acoeff = Table[an, {n, 1, 8, 1}];
bcoeff = Table[Abs[bn], {n, 1, 8, 1}];
```

```
ListPlot[{acoeff, bcoeff},
  Filling -> Axis,
  PlotRange -> All, AxesOrigin -> {0, 0},
  PlotLabel -> "Arbitrary Function Spectrum",
  PlotLegends -> {"an", "bn"},
  AxesLabel -> {"Harmonic", "Magnitude"},
  GridLines -> Automatic,
  Epilog -> {
    Text[Row[{"DC = ", a0}], Scaled[{0.5, 0.9}], Left],
    Text[Row[{"Fundamental = ",  $\frac{1}{T}$ , "Hz"}], Scaled[{0.5, 0.8}], Left]}]
```

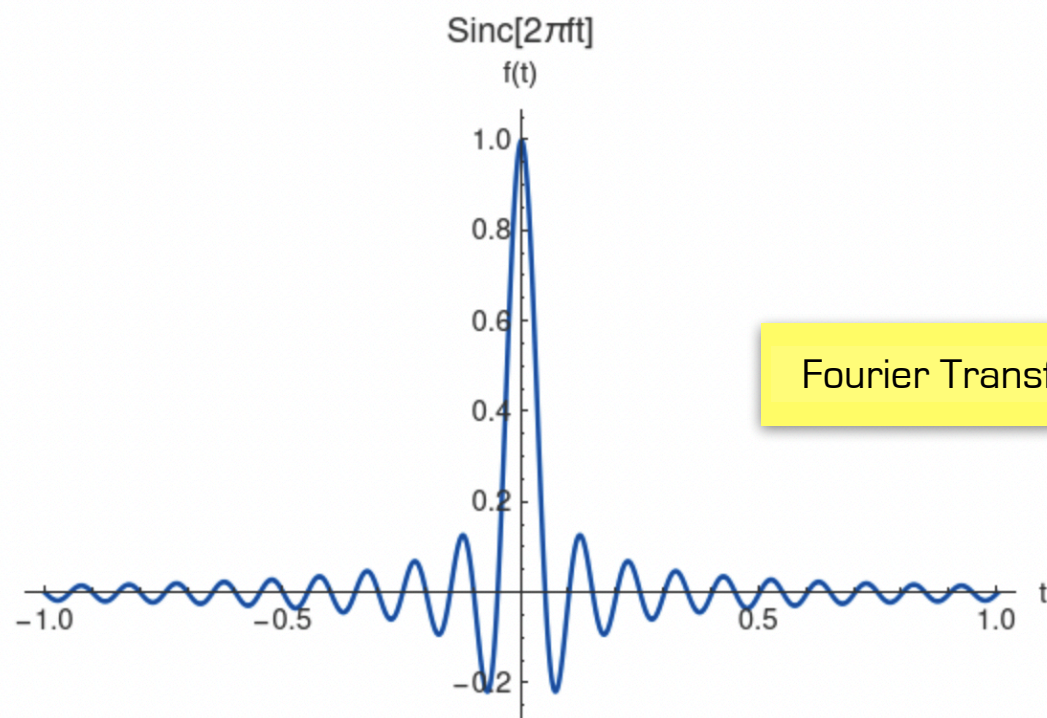


FourierCoefficient Function

Sinc Pulse

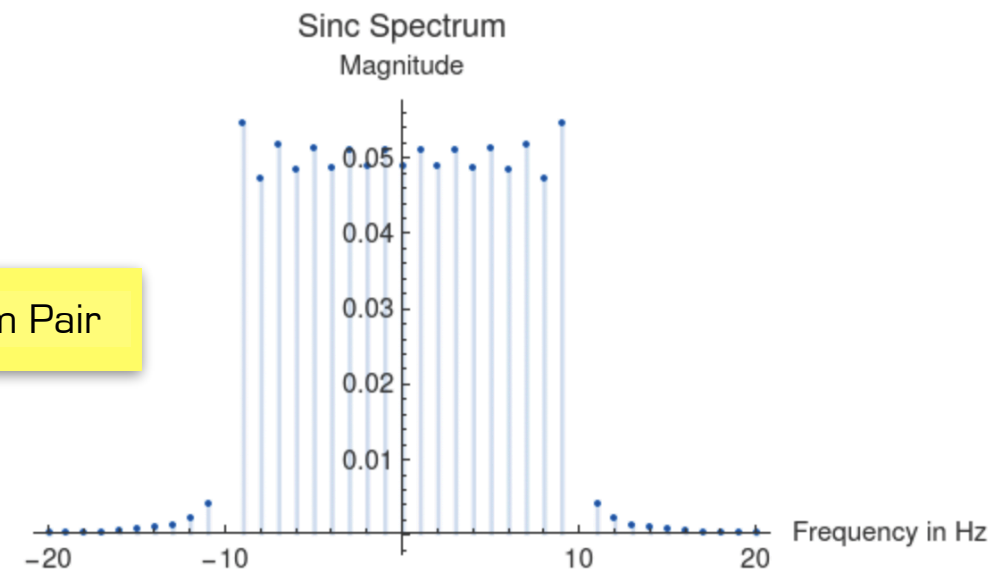
Using the [FourierCoefficient](#) and
[DiscretePlot](#) Commands

```
f = 10;  
fcn4 = Sinc[2  $\pi$  f t];  
Plot[fc4, {t, -1, 1},  
PlotRange  $\rightarrow$  All,  
PlotLabel  $\rightarrow$  "Sinc[2 $\pi$ ft]",  
AxesLabel  $\rightarrow$  {"t", "f(t)"}]
```



Fourier Transform Pair

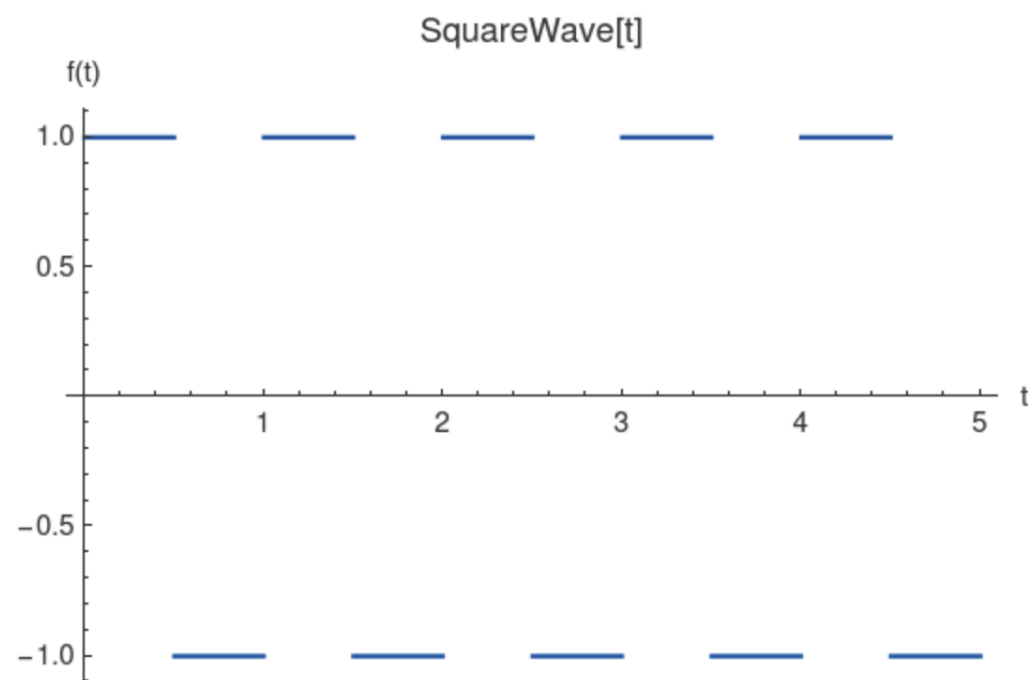
```
fcfcn4 = FourierCoefficient[fc4, t, n, FourierParameters  $\rightarrow$  {0, -2  $\pi$ }];  
DiscretePlot[Abs[%], {n, -20, 20},  
PlotRange  $\rightarrow$  All,  
PlotLabel  $\rightarrow$  "Sinc Spectrum",  
AxesLabel  $\rightarrow$  {"Frequency in Hz", "Magnitude"}]
```



Square Wave

In this case, using the FourierTransform command also results in the Delta function.

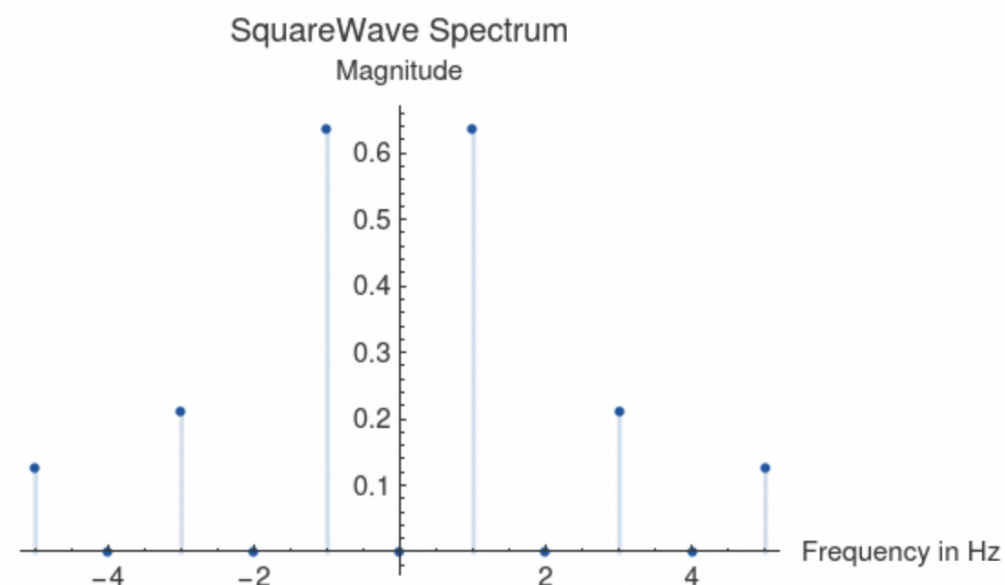
```
fcn = SquareWave[t];
Plot[fcn, {t, 0, 5},
  PlotRange -> All,
  PlotLabel -> "SquareWave[t]",
  AxesLabel -> {"t", "f(t)"}]
```



```
fcfcn = FourierCoefficient[fcn, t, n,
  FourierParameters -> {0, -2 π}]
```

$$\begin{cases} 0 & n \neq 0 \\ \frac{i (-1)^n (-1 + (-1)^n)}{n \pi} & \text{True} \end{cases}$$

```
DiscretePlot[Abs[%], {n, -5, 5},
  PlotRange -> All,
  PlotLabel -> "SquareWave Spectrum",
  AxesLabel -> {"Frequency in Hz", "Magnitude"}]
```



Fourier Function

Arbitrary Waveform

```
Clear[ns, fs, dt, signal, signal1, signal2,  
      tw, timesignal, ftsignal, ftsignaltable]
```

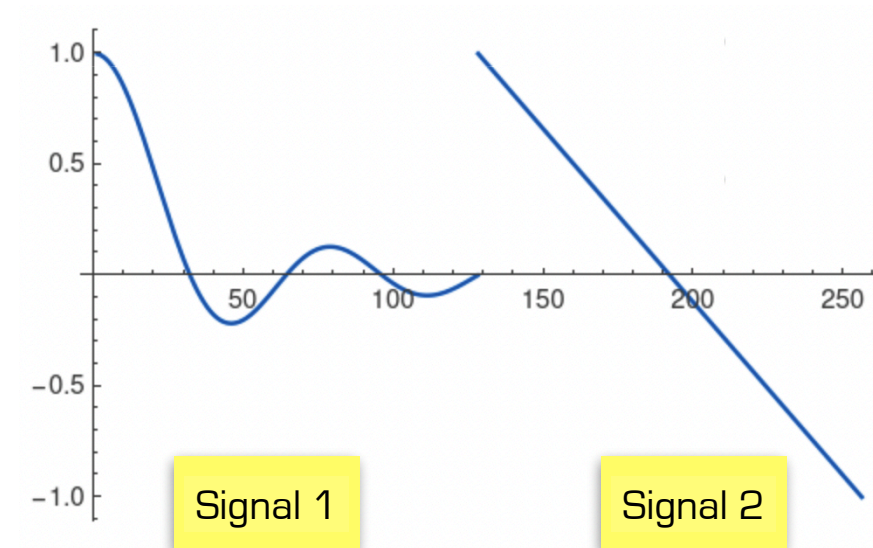
```
tw = 1; (* time window *)
```

```
ns = 256; (* number of samples *)
```

$$\text{signal1} = \frac{\sin\left[2\pi 4 \frac{n}{ns}\right]}{2\pi 4 \frac{n}{ns}};$$

$$\text{signal2} = 3 - 4 \frac{n}{ns};$$

$$\text{dt} = \frac{\text{tw}}{ns}; \quad (* \text{ Time increment (sec) } *)$$

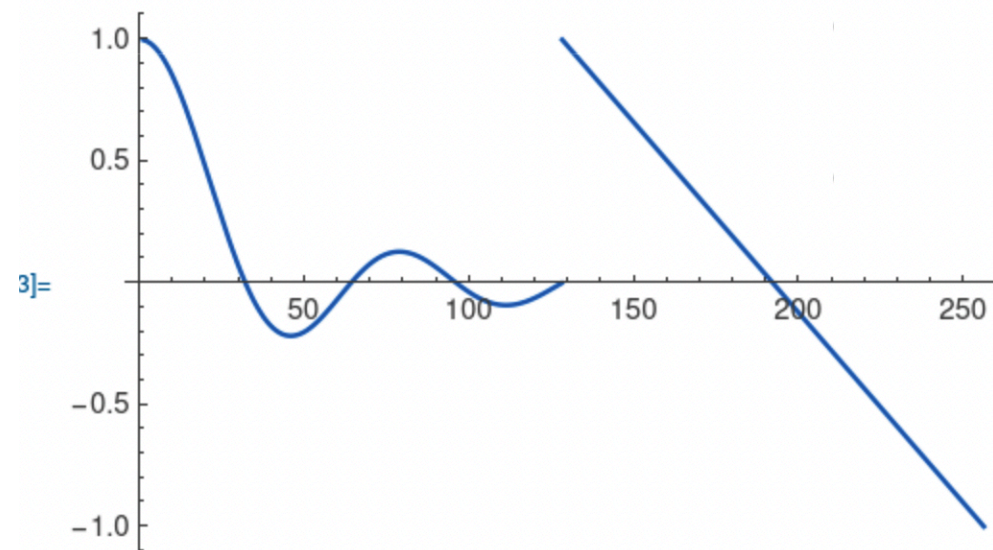


Each signal can be defined individually and then sequenced using the [Piecewise](#) function

Calculate and tabulate the signal

```
]:= fs = Piecewise[{{signal1, 1 ≤ n ≤  $\frac{ns}{2}$ },  
                  {signal2, ns/2 ≤ n ≤ ns}}];
```

```
Plot[fs, {n, 1, ns}]
```



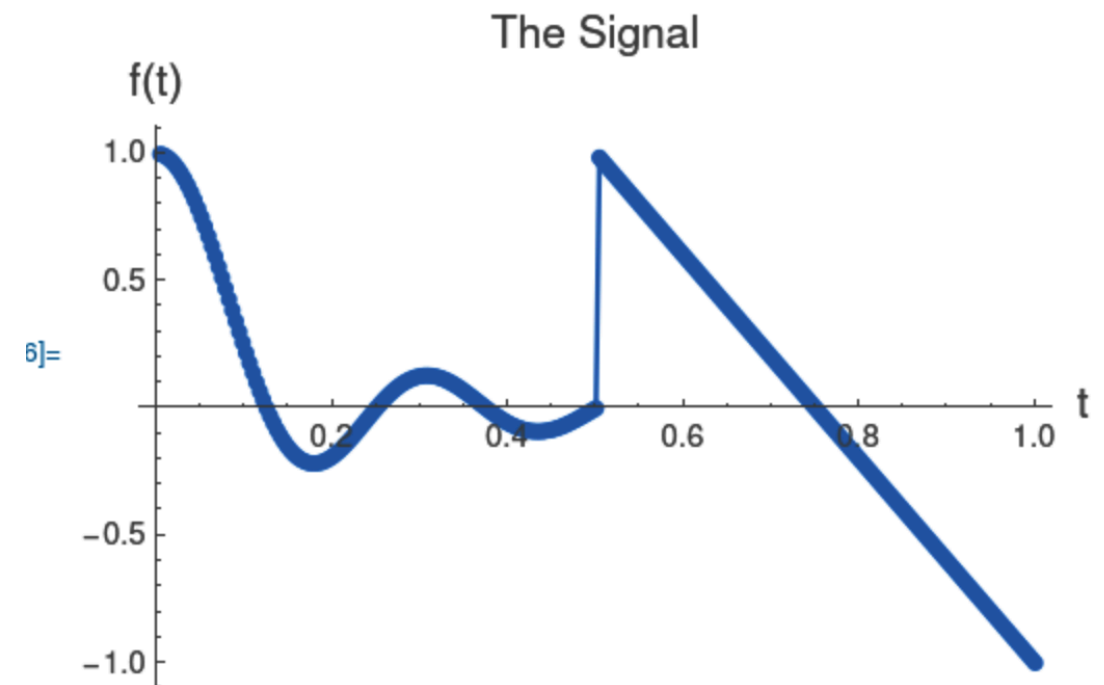
```
]:= signal = Table[fs, {n, ns}];  
timesignal = Table[{n dt, signal[[n]]},  
                  {n, 1, ns}];  
(* Add the time scale to the signal table *)
```

Converting sample number n,
to n dt seconds

Sequenced using the [Piecewise](#) function

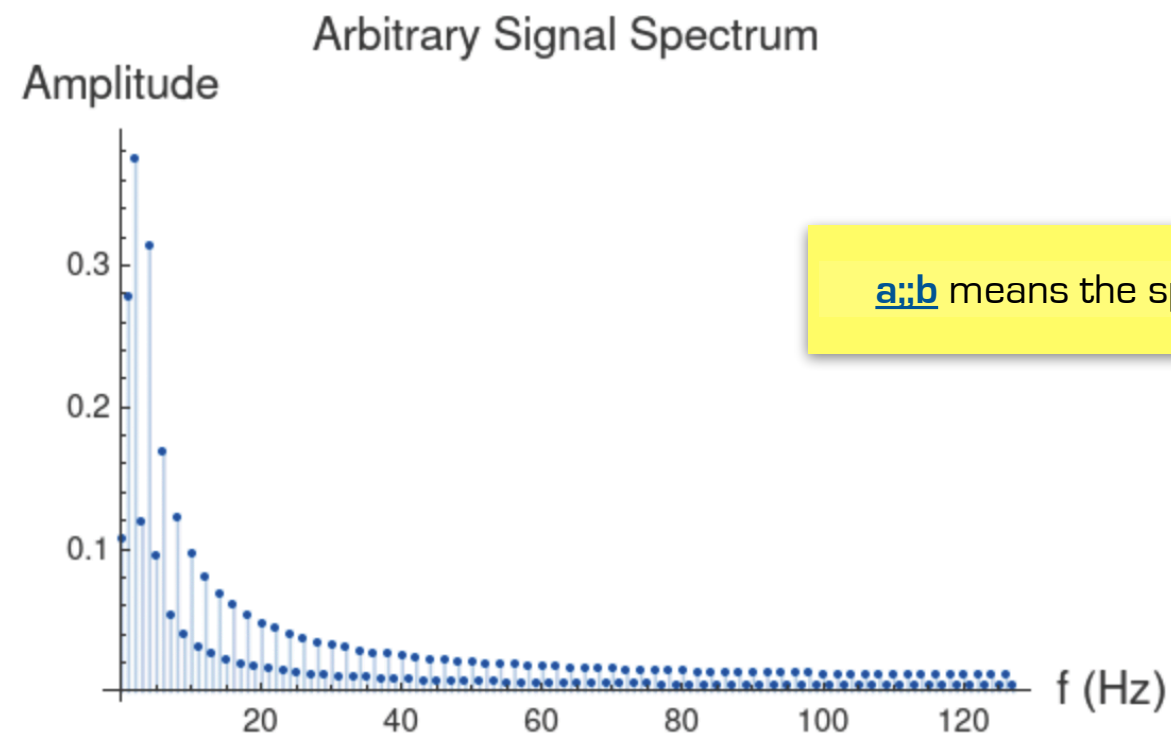
Plot the time domain function

```
ListLinePlot[timesignal,  
             AxesLabel → {"t", "f(t)"},  
             PlotMarkers → Automatic,  
             PlotLabel → "The Signal"]
```



Using the Fourier Transform to
determine the frequency spectrum

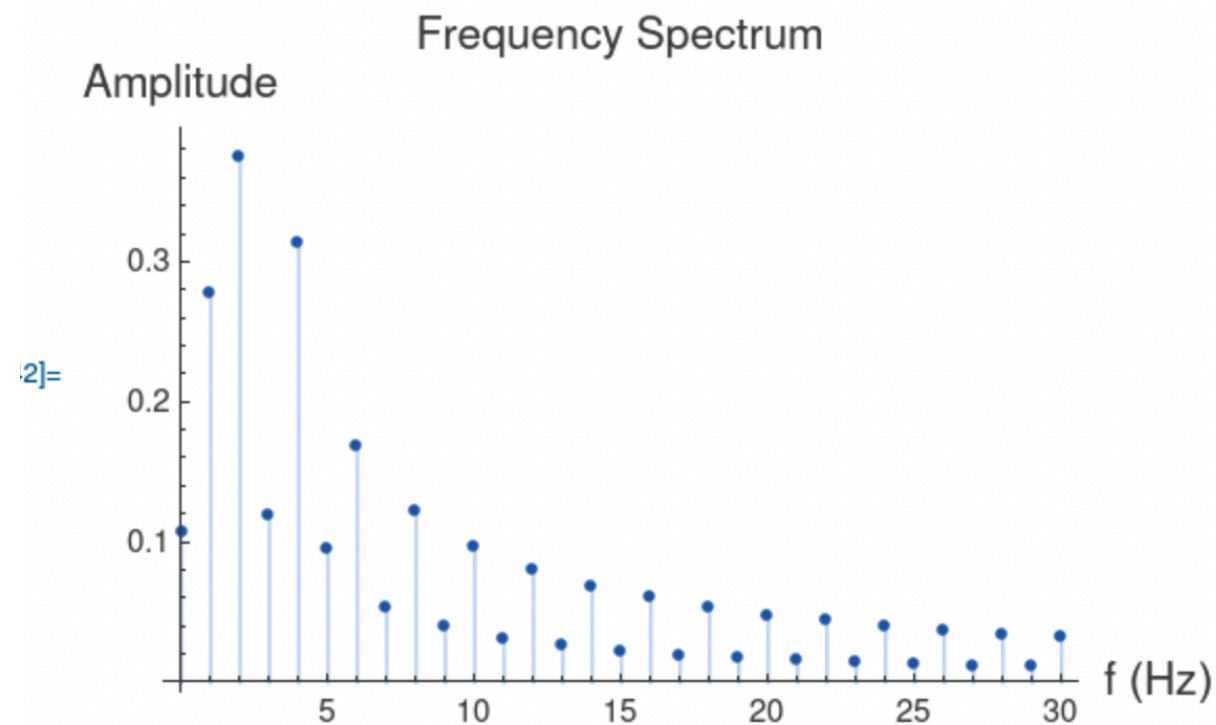
```
ftsignal = Fourier[signal, FourierParameters -> {1, -1}];
ftsignaltable = Table[{ $\frac{n-1}{tw}$ ,  $\frac{2}{ns}$  Abs[ftsignal[[n]]]}, {n, 1,  $\frac{ns}{2}$ };
ListPlot[ftsignaltable[[1 ;;  $\frac{ns}{2}$ ]],
PlotLabel -> "Arbitrary Signal Spectrum",
PlotRange -> All,
Filling -> Axis,
AxesLabel -> {"f (Hz)", "Amplitude"}]
```



a;b means the span of elements from a to b

Zoom in on the spectrum

```
1]:= fmin = 0; (* minimum frequency *)  
    fmax = 30; (* maximum frequency *)  
  
2]:= ListPlot[ftsignaltable[[fmin tw + 1 ;; fmax tw + 1]],  
    PlotLabel → "Frequency Spectrum",  
    PlotRange → All,  
    Filling → Axis,  
    AxesLabel → {"f (Hz)", "Amplitude"}]
```



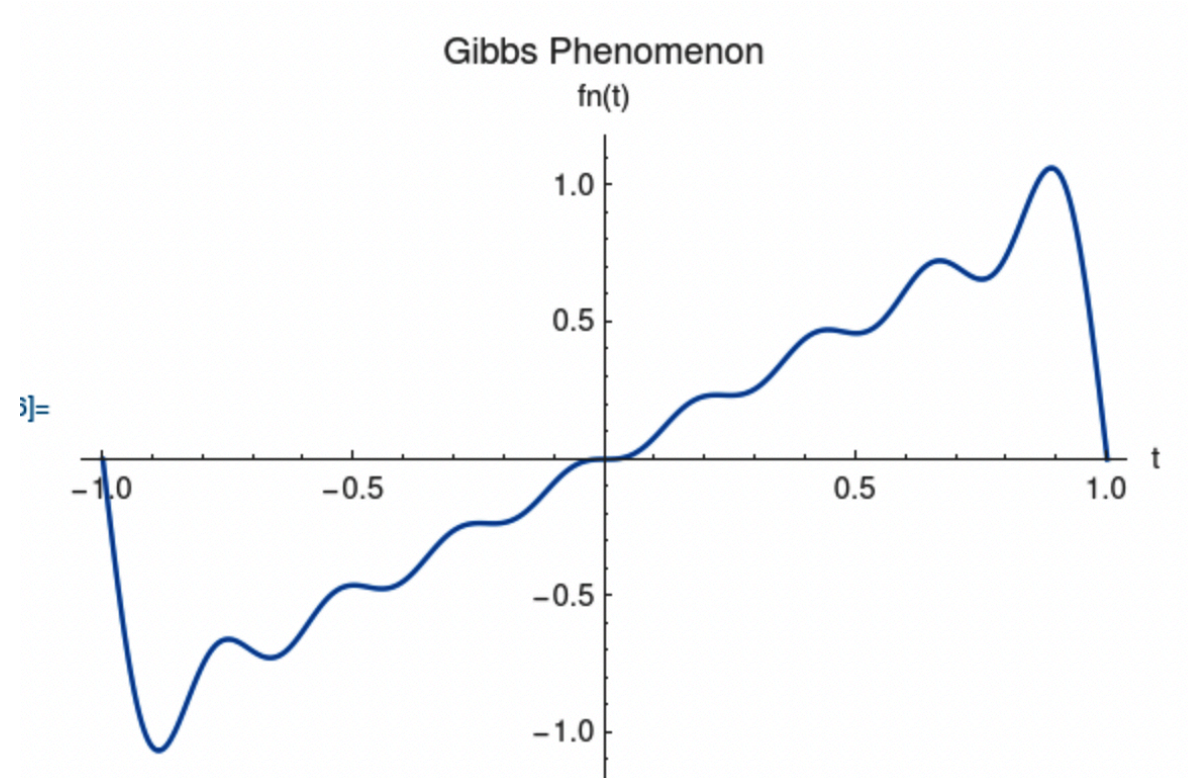
Any part of the spectrum
can be examined by
adjusting fmin and fmax

Gibbs Phenomenon

The [Gibbs phenomenon](#) occurs in bandwidth limited systems and DSP.

```
fn =  $\frac{2}{T} b_n \sin\left[\frac{2 n \pi t}{2}\right];$   
Plot[ $\sum_{n=1}^8 fn$ , {t, -1, 1},  
PlotLabel -> "Gibbs Phenomenon",  
AxesLabel -> {"t", "fn(t)"}]
```

The Basic Math Assistant contains the summation template.



This illustrates how repetitive waveforms can be constructed from a series of harmonically related sinusoids.

Brick Wall Filter

The [brick wall filter](#) represents the ideal low pass filter

```
Clear[BWF, T, B]
```

```
B = 1/2;
```

```
BWF[ω_] = UnitStep[ω + B] - UnitStep[ω - B]
```

```
-UnitStep[-1/2 + ω] + UnitStep[1/2 + ω]
```

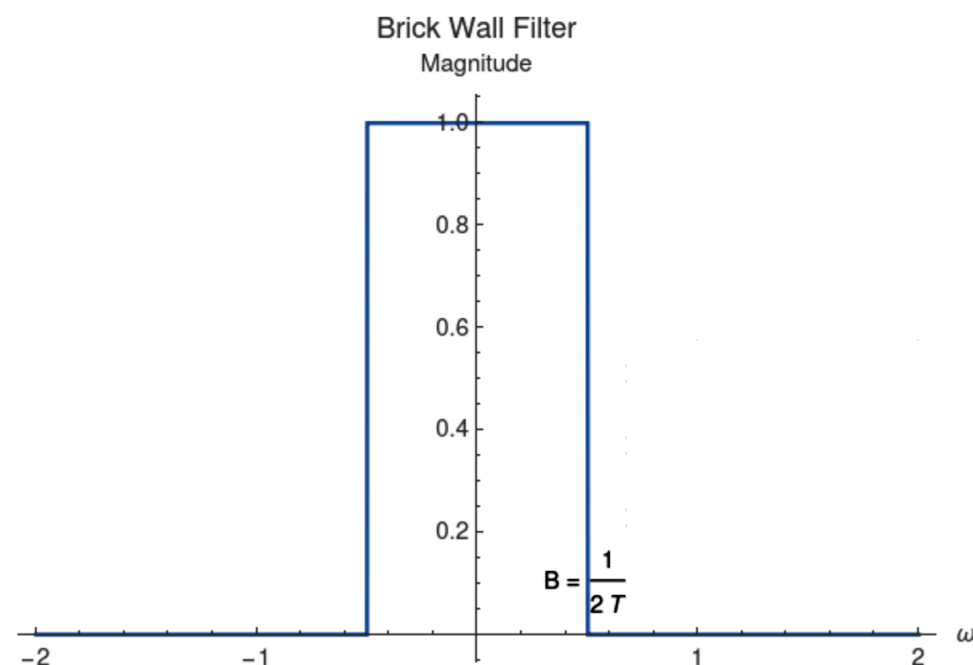
```
Plot[BWF[ω], {ω, -2, 2},
```

```
Exclusions -> None,
```

```
PlotLabel -> "Brick Wall Filter",
```

```
AxesLabel -> {"ω", "Magnitude"},
```

```
Epilog -> {Text["B = 1/(2 T)", {B, 0.1}]}]
```



In this case, the impulse response has been normalized to 1.

```
InverseFourierTransform[BWF[ω], ω, t, FourierParameters -> {0, -2 π}]
```

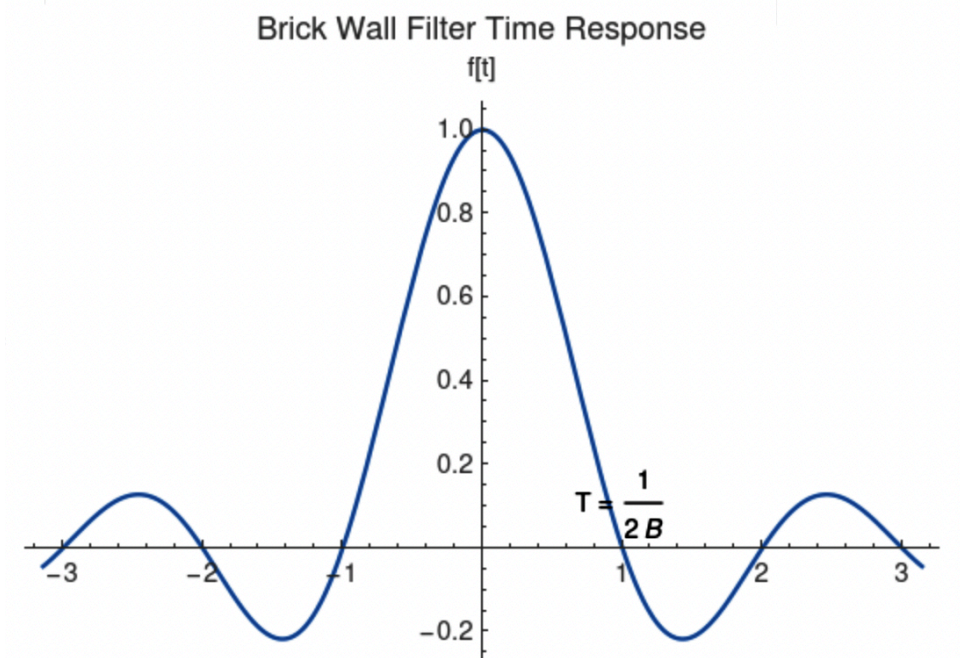
$$-\frac{i e^{-i \pi t} (-1 + e^{2 i \pi t})}{2 \pi t}$$

```
Plot[%, {t, -π, π}, PlotRange -> All,
```

```
PlotLabel -> "Brick Wall Filter Time Response",
```

```
AxesLabel -> {t, "f[t]"},
```

```
Epilog -> {Text["T = 1/(2 B)", {1/(2 B), 0.1}]}]
```



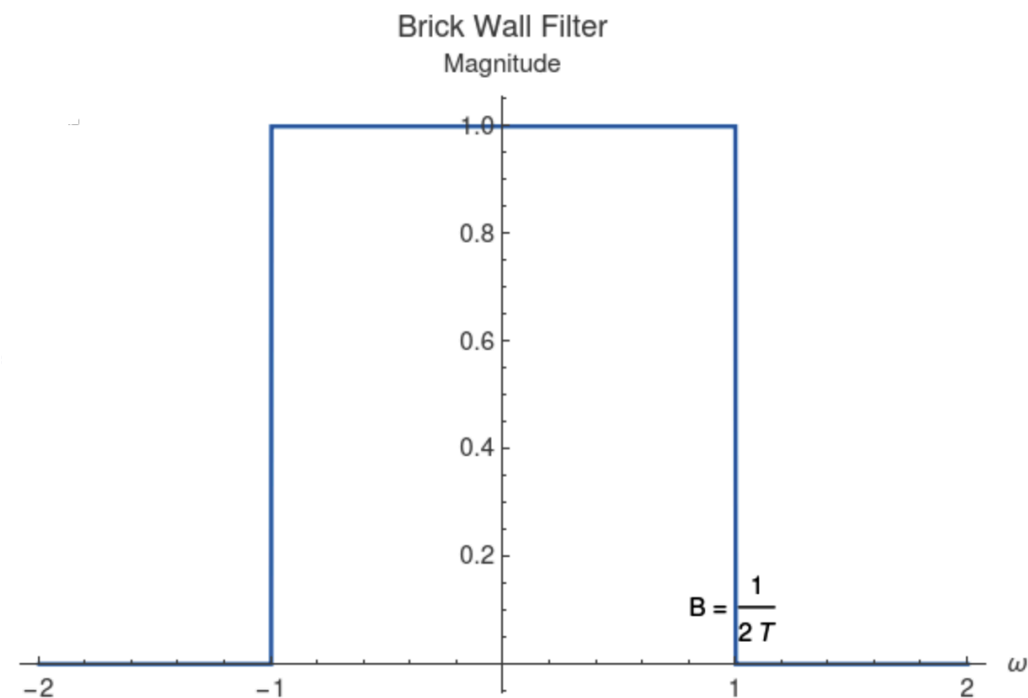
```
FindRoot[-(i e^{-i \pi t} (-1 + e^{2 i \pi t})) / (2 \pi t), {t, 1}]
```

```
{t -> 1.}
```


In this case, the frequency response has been normalized to 1.

```
Clear[BWF, T, B]
B = 1;
BWF[ω_] = UnitStep[ω + B] - UnitStep[ω - B]
-UnitStep[-1 + ω] + UnitStep[1 + ω]
```

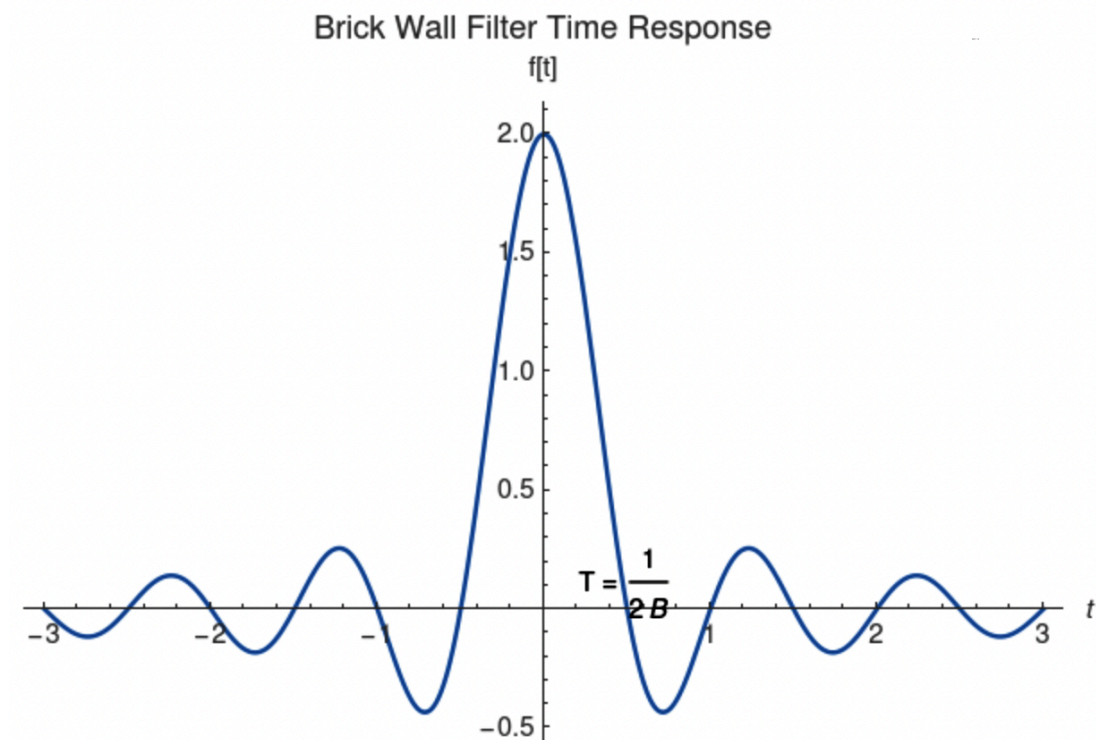
```
Plot[BWF[ω], {ω, -2, 2},
Exclusions → None,
PlotLabel → "Brick Wall Filter",
AxesLabel → {"ω", "Magnitude"},
Epilog → {Text["B =  $\frac{1}{2T}$ ", {B, 0.1}]}]
```



```
InverseFourierTransform[BWF[ω], ω, t, FourierParameters → {0, -2π}]
```

$$-\frac{i e^{-2 i \pi t} (-1 + e^{4 i \pi t})}{2 \pi t}$$

```
Plot[%, {t, -3, 3}, PlotRange → All,
PlotLabel → "Brick Wall Filter Time Response",
AxesLabel → {t, "f[t]"},
Epilog → {Text["T =  $\frac{1}{2B}$ ", {1/(2B), 0.1}]}]
```



```
FindRoot[- $\frac{i e^{-2 i \pi t} (-1 + e^{4 i \pi t})}{2 \pi t}$ , {t, 0.25}]
```

```
{t → 0.5 + 0. i}
```


Manipulate Brick Wall Filter

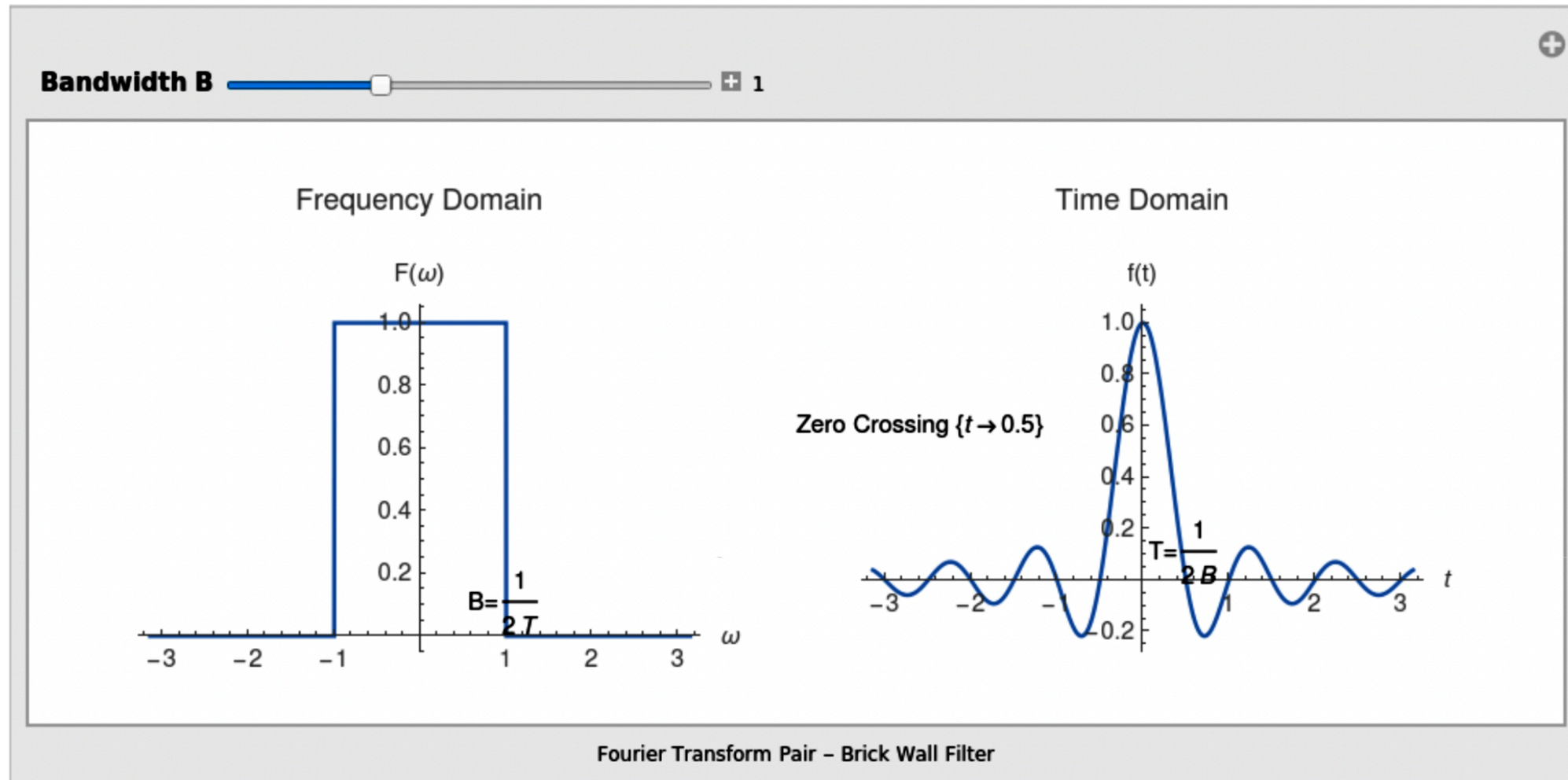
```

Clear[BWF, IFT, T, B]
BWF[ω_] = UnitStep[ω + B] - UnitStep[ω - B];
IFT = InverseFourierTransform[BWF[ω], ω, t, FourierParameters → {0, -2 π}];

Manipulate[
GraphicsRow[{
  T =  $\frac{1}{2B}$ ;
  BWF[ω_] = UnitStep[ω + B] - UnitStep[ω - B];
  Plot[BWF[ω], {ω, -π, π},
    Exclusions → None,
    PlotLabel → "Frequency Domain", AxesLabel → {"ω", "F(ω)"},
    Epilog → {Text["B =  $\frac{1}{2T}$ ", {B, 0.1}]}],
  IFT = - $\frac{1}{2B} \frac{i e^{-2iB\pi t} (-1 + e^{4iB\pi t})}{2\pi t}$ ; (* Normalized Amplitude *)
  root = FindRoot[IFT, {t,  $\frac{1}{2B}$ }];
  Plot[IFT, {t, -π, π}, PlotRange → All,
    PlotLabel → "Time Domain", AxesLabel → {t, "f(t)"},
    Epilog → {Text["T =  $\frac{1}{2B}$ ", {T, 0.1}]}],
  Epilog → Text[Row[{"Zero Crossing ", root}], Scaled[{0.5, 0.5}], Left],
  {{B, 1, Style["Bandwidth B", Bold, 10]}, 0.1, 3, 0.1, Appearance → "Labeled"},
  FrameLabel → Style["Fourier Transform Pair - Brick Wall Filter"]}

```

This relationship forms the basis of [Nyquist](#) or [Partial Response signalling](#).



Digital Approximation of a Low Pass Filter

```
Clear[n, nc, h,  $\omega$ , LFT, TIR]
```

Brick Wall Filter ☐

```
B = 1.0;
```

```
BWF = Piecewise[{{1, 0 ≤  $\omega$  ≤ B}}];
```

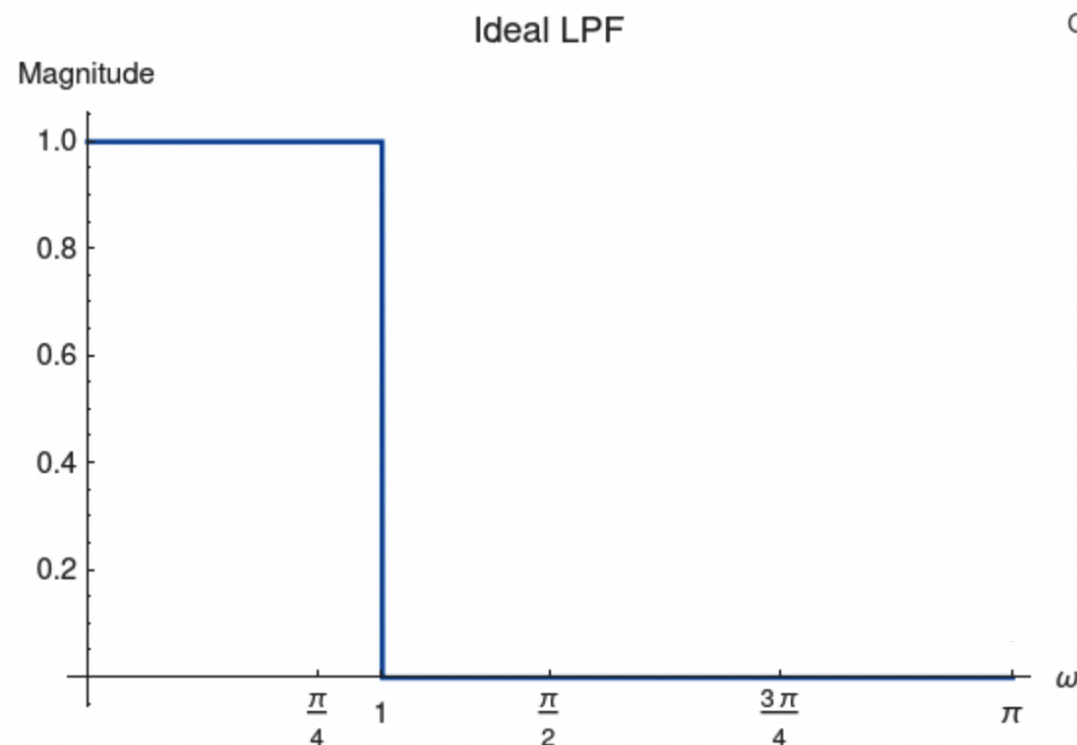
```
Plot[BWF, { $\omega$ , 0,  $\pi$ },
```

```
Exclusions → None,
```

```
PlotLabel → "Ideal LPF",
```

```
AxesLabel → {" $\omega$ ", "Magnitude"},
```

```
Ticks → {{0, 1,  $\frac{\pi}{4}$ ,  $\frac{\pi}{2}$ ,  $\frac{3\pi}{4}$ ,  $\pi$ }, Automatic}]
```



Truncated Impulse Response ☐

```
nc = 101; (* number of coefficients *)
```

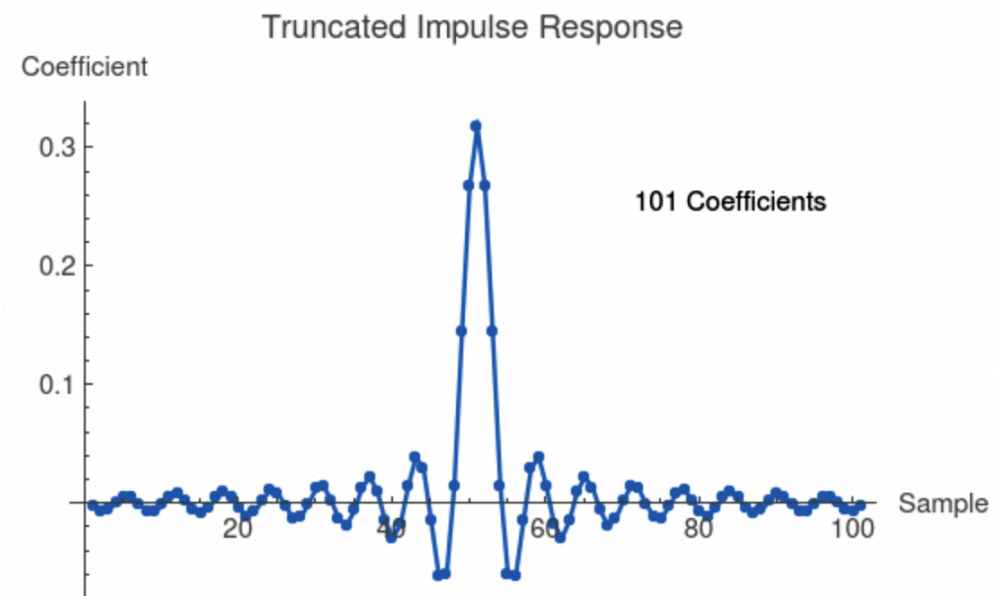
```
TIR = LeastSquaresFilterKernel["Lowpass", B], nc];
```

```
ListLinePlot[TIR, PlotRange → All, Mesh → Full,
```

```
PlotLabel → "Truncated Impulse Response",
```

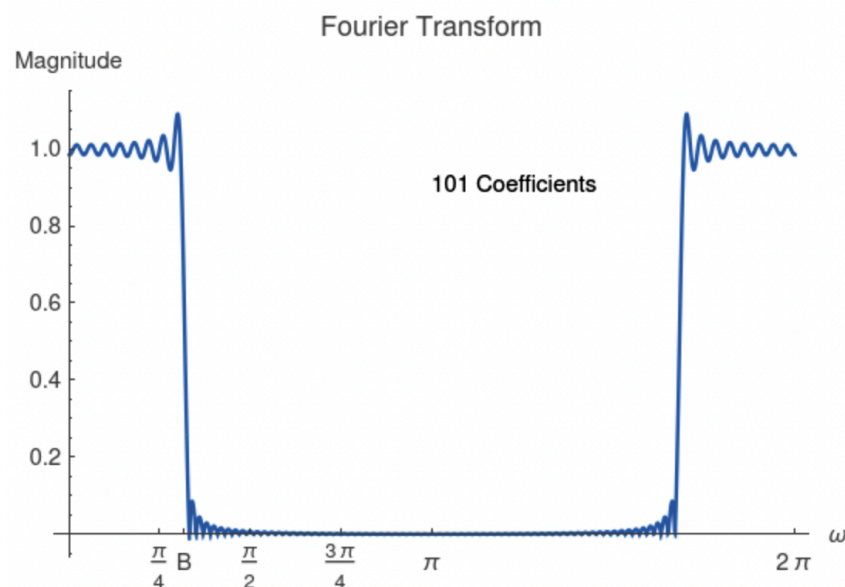
```
AxesLabel → {"Sample", "Coefficient"},
```

```
Epilog → Text[Row[{nc, " Coefficients"}], Scaled[{0.7, 0.8}], Left]]
```



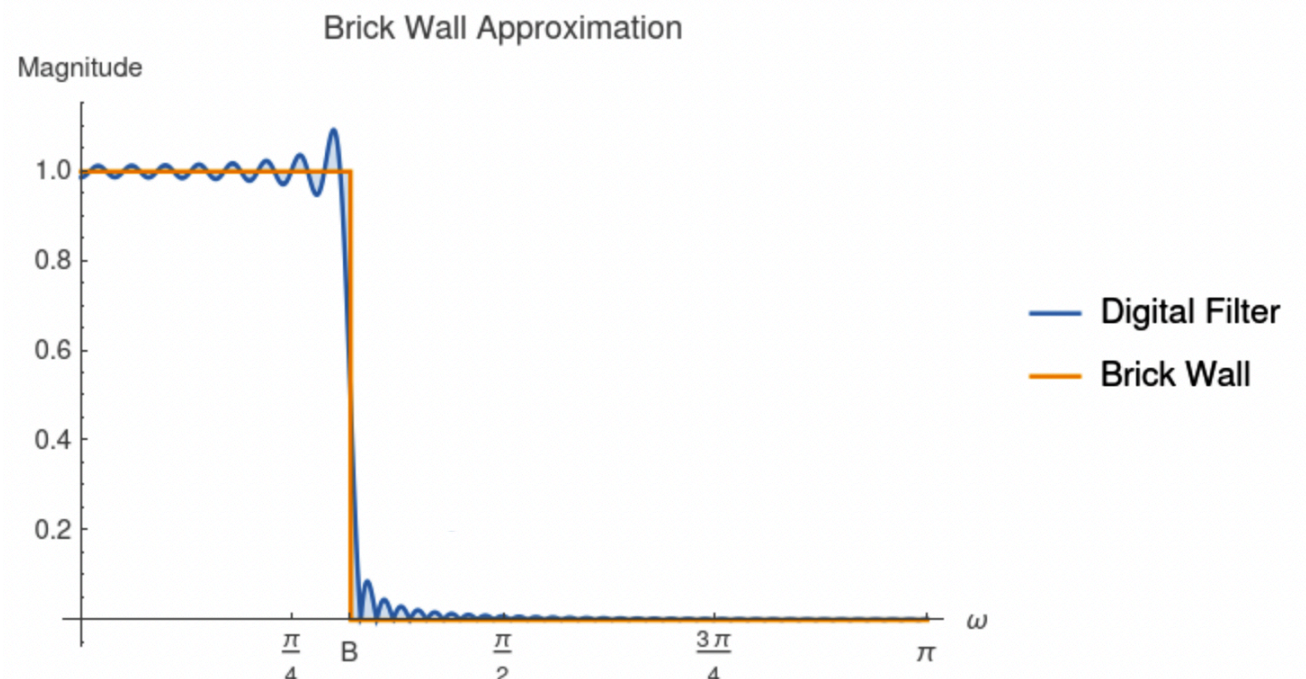
Fourier Transform

```
LFT = Abs[ListFourierSequenceTransform[TIR,  $\omega$ ]];
Plot[LFT, { $\omega$ , 0, 2  $\pi$ },
  PlotLabel → "Fourier Transform",
  AxesLabel → {" $\omega$ ", "Magnitude"},
  Ticks → {{0, {1, "B"}},  $\frac{\pi}{4}$ ,  $\frac{\pi}{2}$ ,  $\frac{3\pi}{4}$ ,  $\pi$ , 2  $\pi$ }, Automatic},
  Epilog → Text[Row[{nc, " Coefficients"}], Scaled[{0.5, 0.8}], Left]]
```



Compare Filters

```
Plot[{LFT, BWF}, { $\omega$ , 0,  $\pi$ },
  PlotRange → All,
  Ticks → {{0, {1, "B"}},  $\frac{\pi}{4}$ ,  $\frac{\pi}{2}$ ,  $\frac{3\pi}{4}$ ,  $\pi$ }, Automatic},
  PlotLabel → "Brick Wall Approximation",
  AxesLabel → {" $\omega$ ", "Magnitude"},
  PlotLegends → {"Digital Filter", "Brick Wall"},
  Exclusions → None,
  Filling → {1 → {2}}]
```

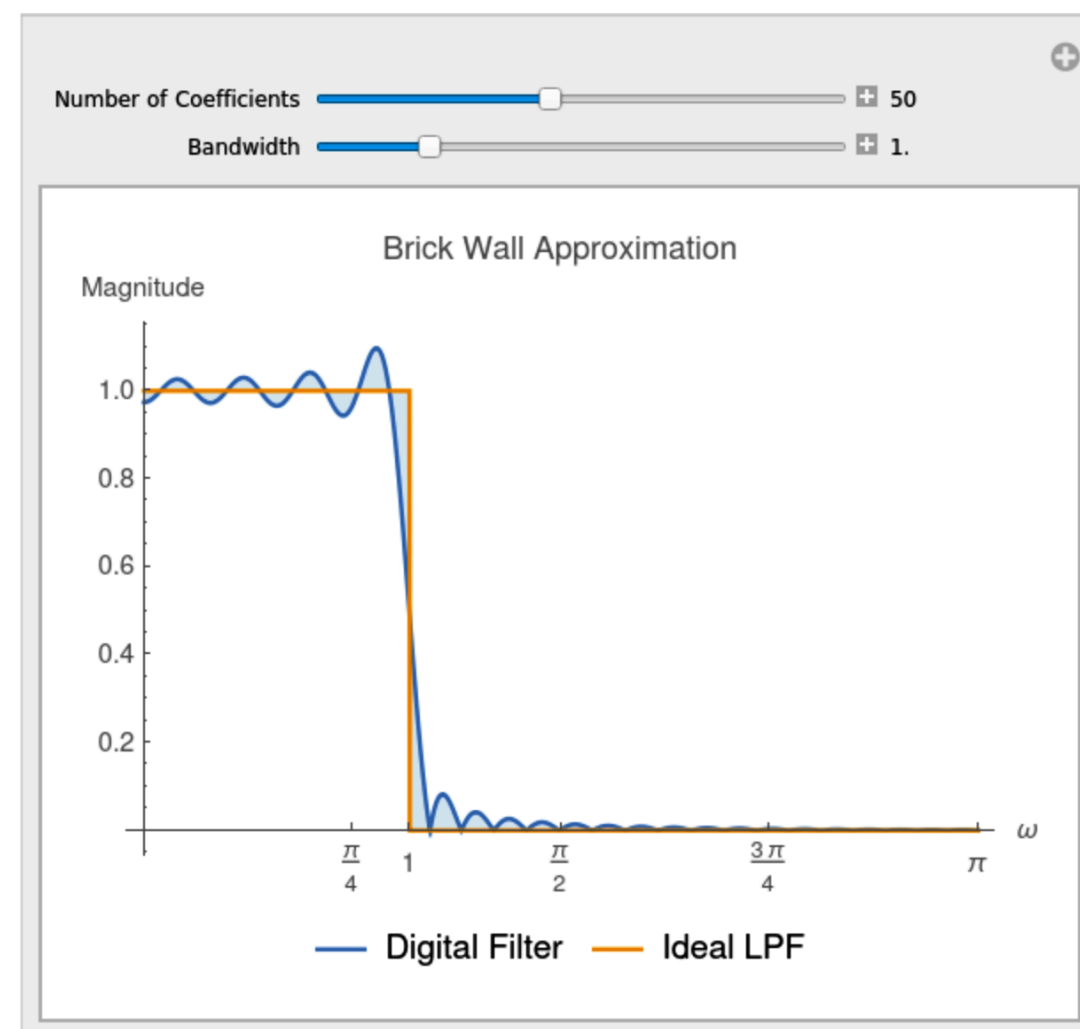


Manipulate Number of Coefficients

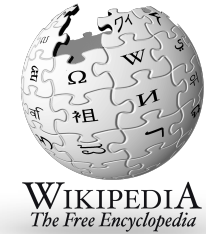
```
Manipulate[
  BWF = Piecewise[{{1, 0 ≤ ω ≤ B}}];
  TIR = LeastSquaresFilterKernel["Lowpass", B, nc];
  LFT = Abs[ListFourierSequenceTransform[TIR, ω]];

  Plot[{LFT, BWF}, {ω, 0, π},
    PlotRange → All,
    Ticks → {{0, 1,  $\frac{\pi}{4}$ ,  $\frac{\pi}{2}$ ,  $\frac{3\pi}{4}$ , π}, Automatic},
    PlotLabel → "Brick Wall Approximation",
    AxesLabel → {"ω", "Magnitude"},
    PlotLegends → Placed[{"Digital Filter", "Ideal LPF"}, Below],
    Exclusions → None, Filling → {1 → {2}}],

  {{nc, 50, "Number of Coefficients"}, 10, 100, 10, Appearance → "Labeled"},
  {{B, 1.0, "Bandwidth"}, 0.5, 3, Appearance → "Labeled"}]
```



CDMA



```
ClearAll["Global`*"]
```

```
(*---Parameters---*)
```

```
Tb = 1; (*bit period*)
```

```
nob = 8; (*number of bits*)
```

```
cpb = 8; (*chips per bit*)
```

```
Tc =  $\frac{Tb}{cpb}$ ; (*chip period*)
```

```
fs = 512; (*sampling frequency >  $\frac{2 \text{ cpb}}{Tb}$  *)
```

```
t = Range[0, nob Tb,  $\frac{1}{fs}$ ]; (*Time axis*)
```

```
(*data signal (+1/-1)*)
```

```
dataBits = RandomChoice[{-1, 1}, nob];
```

The bipolar baseband signal

```
dataSignal =
```

```
Flatten[Table[ConstantArray[dataBits[[i]], Round[Tb fs]], {i, nob}]];
```

The baseband signal is mapped onto the sampled signal

```
(*chip PN code*)
```

```
pnCodeSingle = RandomChoice[{-1, 1}, cpb];
```

The chip signal

```
(*repeat chip PN code for each bit*)
```

```
pnCode = Flatten@Table[pnCodeSingle, nob];
```

```
(*expand chip PN to match sampling*)
```

```
pnSignal =
```

```
Flatten[Table[ConstantArray[pnCode[[i]], Round[Tc fs]],  
{i, Length[pnCode]}]];
```

```
(*truncate to same length*)
minLen = Min[Length[dataSignal], Length[pnSignal]];
dataSignal = dataSignal[[1 ;; minLen]];
pnSignal = pnSignal[[1 ;; minLen]];
```

```
(*replace the bits with chips (CDMA)*)
spreadSignal = dataSignal * pnSignal;
```

The polarity of the chip signal is controlled by the data signal

```
(*FFT function*)
fftMag[x_] := Module[{X, n, f}, n = Length[x];
  X = Abs[RotateLeft[Fourier[x], Floor[n/2]]];
  f = Range[-n/2, n/2 - 1] * (fs/n);
  {f, X}];
```

```
(*spectra*)
{f1, Xdata} = fftMag[dataSignal];
{f2, Xspread} = fftMag[spreadSignal];
```

```
(*add to avoid log of zero*)
eps = 10-12;
```

```
XdataDB = 20 Log10[Xdata + eps];
XspreadDB = 20 Log10[Xspread + eps];
```


(*Plots*)

```
timePlot1 = ListLinePlot[
  Transpose[{ $\frac{\text{Range}[\text{minLen}]}{f_s}$ , dataSignal}],
  GridLines → Automatic,
  PlotLabel → "Data",
  AxesLabel → {"Time (s)", "Amplitude"}];

timePlot2 = ListLinePlot[{
  Transpose[{ $\frac{\text{Range}[\text{minLen}]}{f_s}$ , spreadSignal}]}],
  PlotLabel → "Coded Data",
  AxesLabel → {"Time (s)", "Amplitude"}];
```

```
freqPlot1 = ListPlot[Transpose[{f1, Xdata}],
  PlotLabel → "Data",
  AxesLabel → {"Frequency (Hz)", "Magnitude"},
  Filling → Axis,
  PlotRange → {Automatic, {0, 6}},
  GridLines → Automatic];

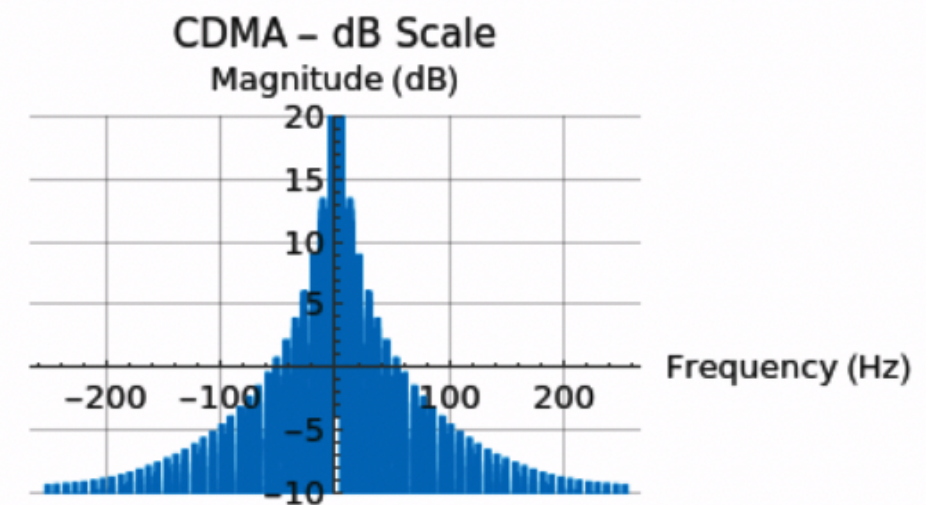
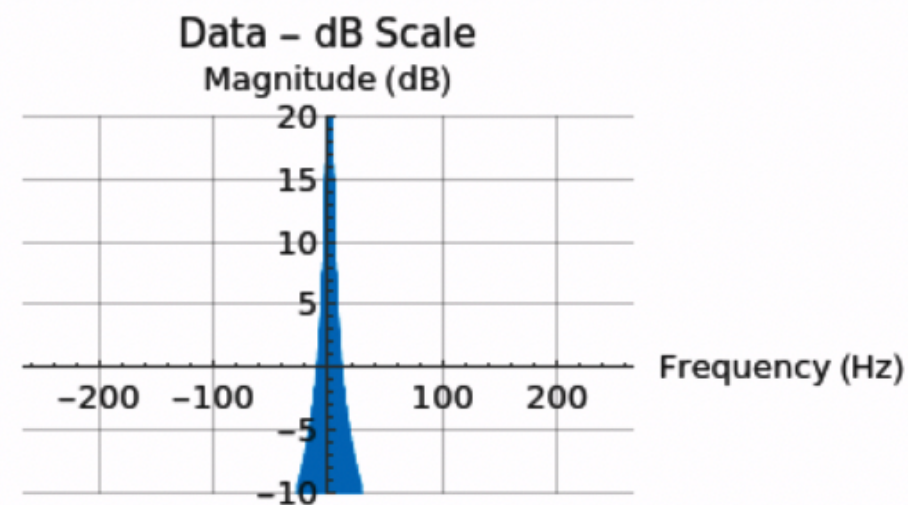
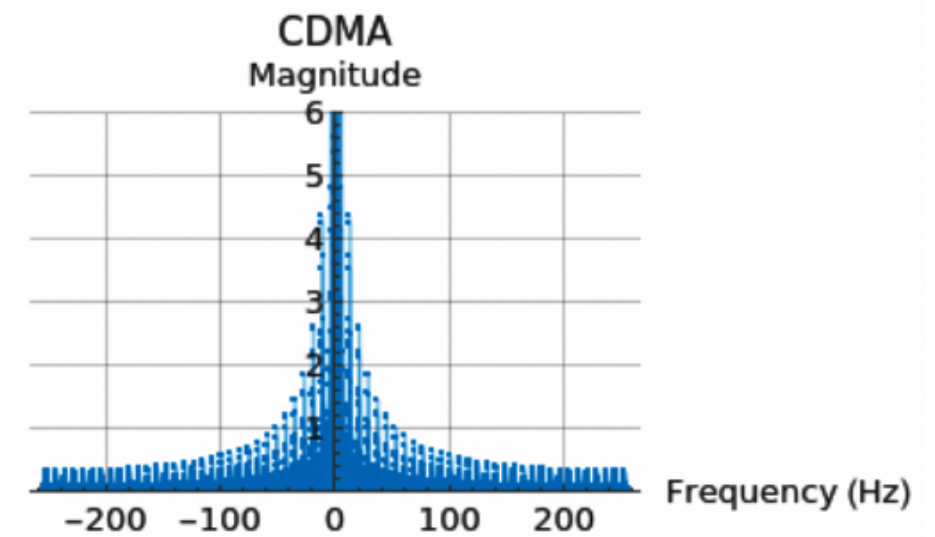
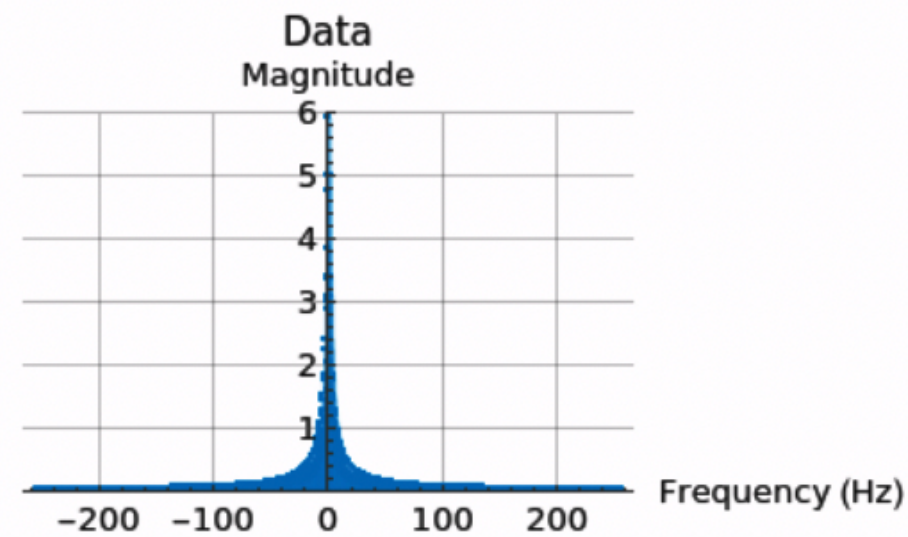
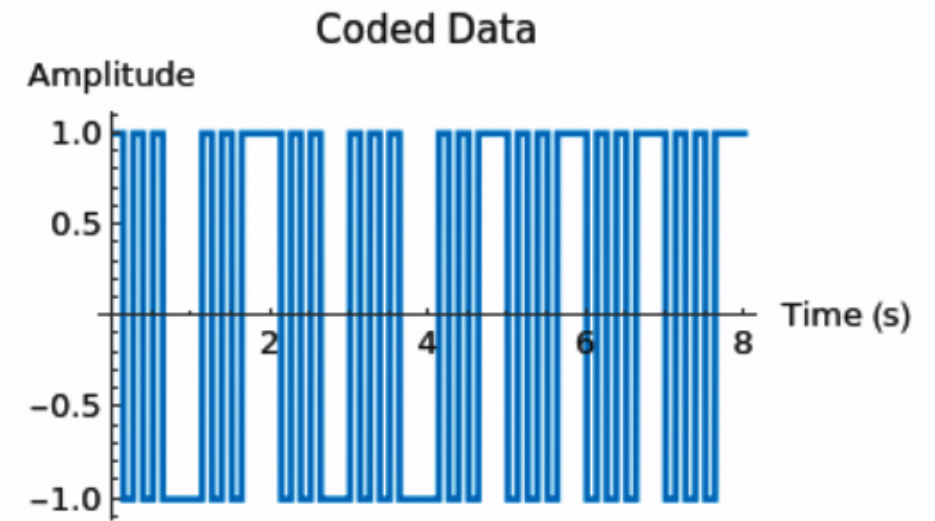
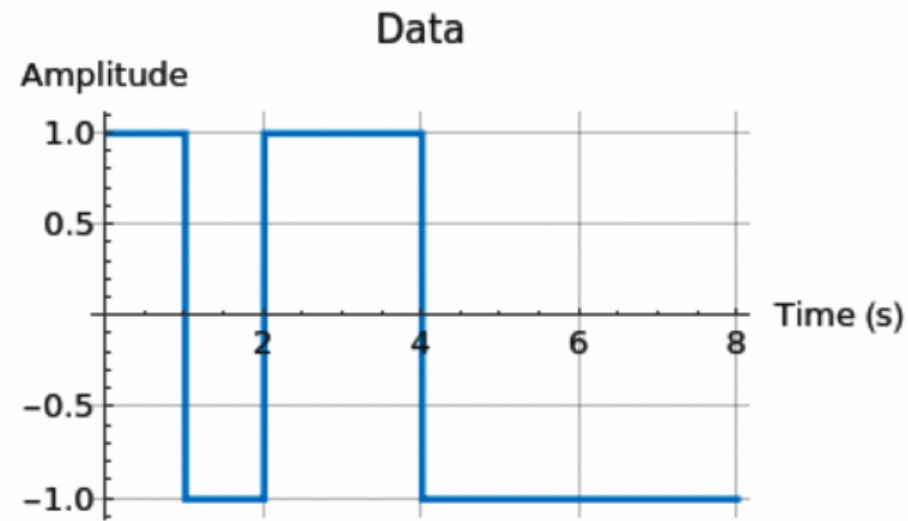
freqPlot2 = ListPlot[Transpose[{f2, Xspread}],
  PlotLabel → "CDMA",
  AxesLabel → {"Frequency (Hz)", "Magnitude"},
  Filling → Axis,
  PlotRange → {Automatic, {0, 6}},
  GridLines → Automatic];
```

(*Frequency plot in dB*)

```
freqPlot3 = ListLinePlot[Transpose[{f1, XdataDB}],
  PlotLabel → "Data - dB Scale",
  AxesLabel → {"Frequency (Hz)", "Magnitude (dB)"},
  PlotRange → {Automatic, {-10, 20}},
  GridLines → Automatic];

freqPlot4 = ListLinePlot[Transpose[{f2, XspreadDB}],
  PlotLabel → "CDMA - dB Scale",
  AxesLabel → {"Frequency (Hz)", "Magnitude (dB)"},
  PlotRange → {Automatic, {-10, 20}},
  GridLines → Automatic];

GraphicsGrid[{{timePlot1, timePlot2}, {freqPlot1, freqPlot2},
  {freqPlot3, freqPlot4}}]
```

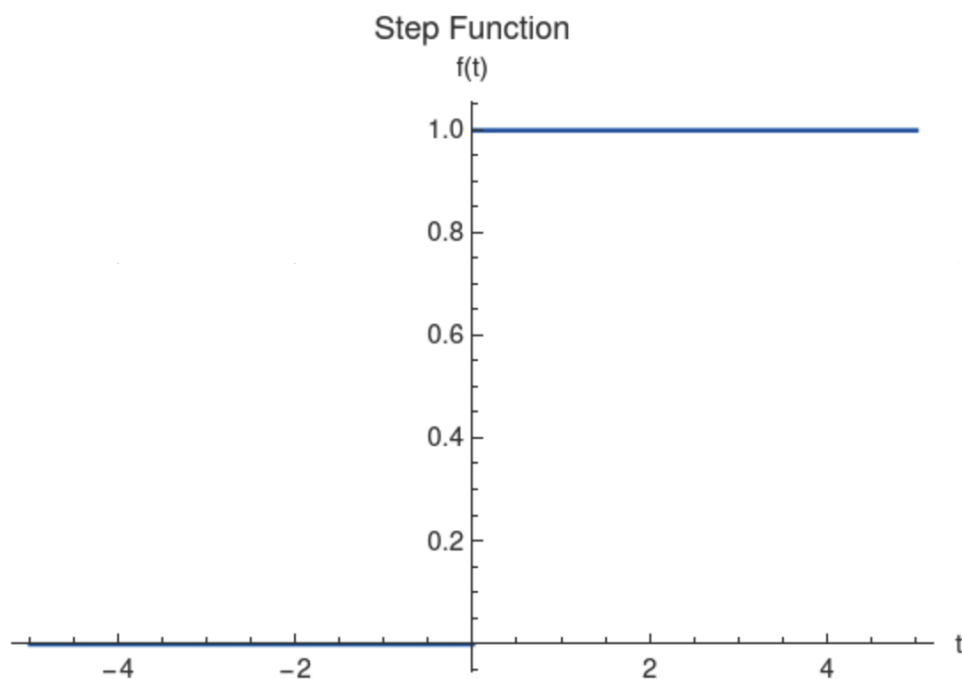



Laplace Transforms

Unit Step

The [Laplace Transform](#) can help reduce certain complex calculations

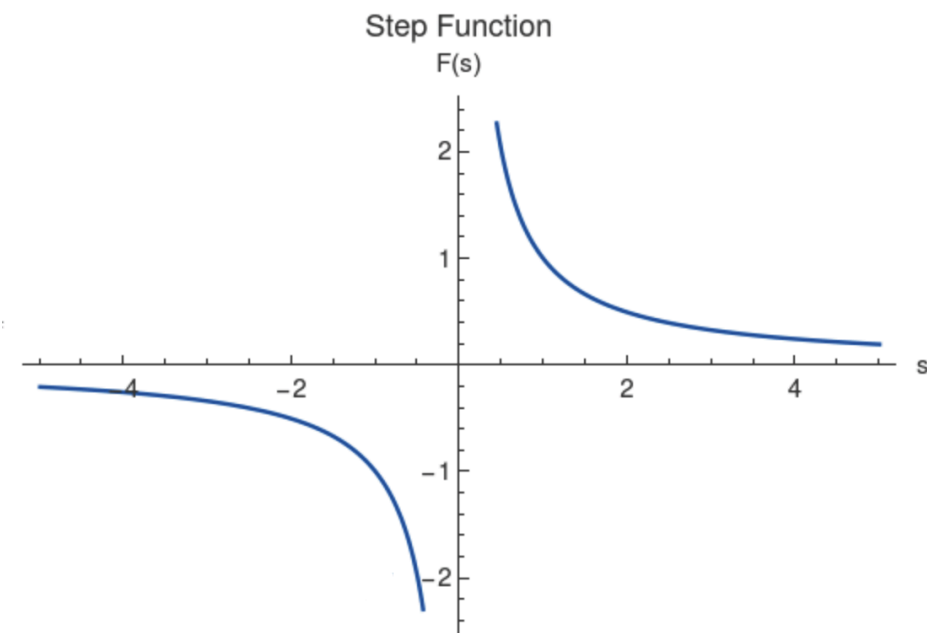
```
Plot[UnitStep[t], {t, -5, 5},  
  AxesLabel → {"t", "f(t)"}, PlotLabel → "Step Function"]
```



```
LaplaceTransform[UnitStep[t], t, s] // Quiet
```

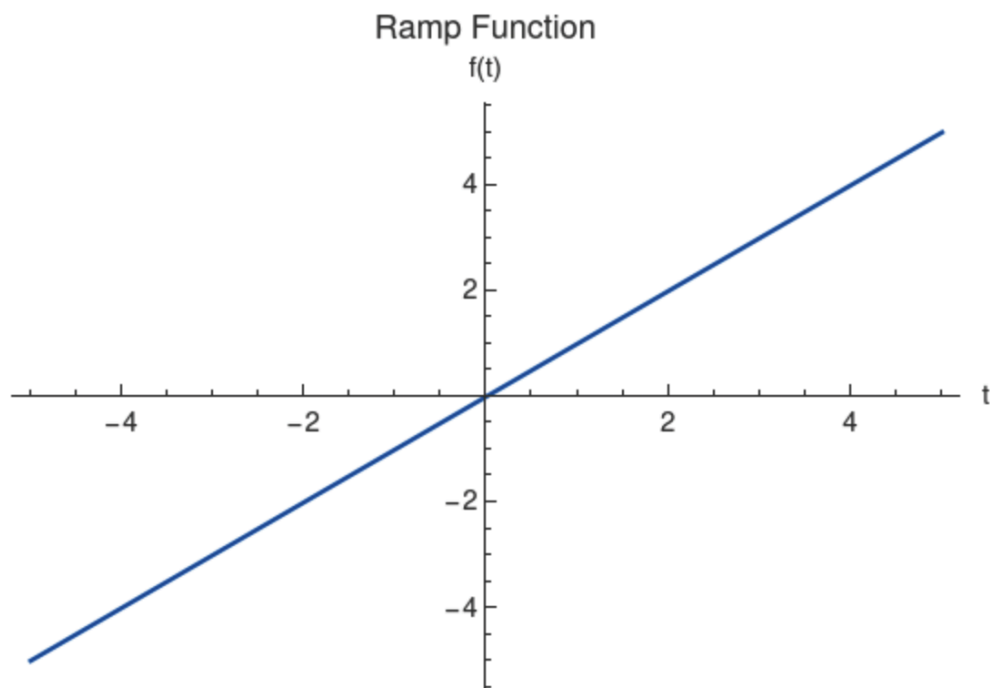
$$\frac{1}{s}$$

```
Plot[%, {s, -5, 5},  
  AxesLabel → {"s", "F(s)"}, PlotLabel → "Step Function"]
```



Ramp

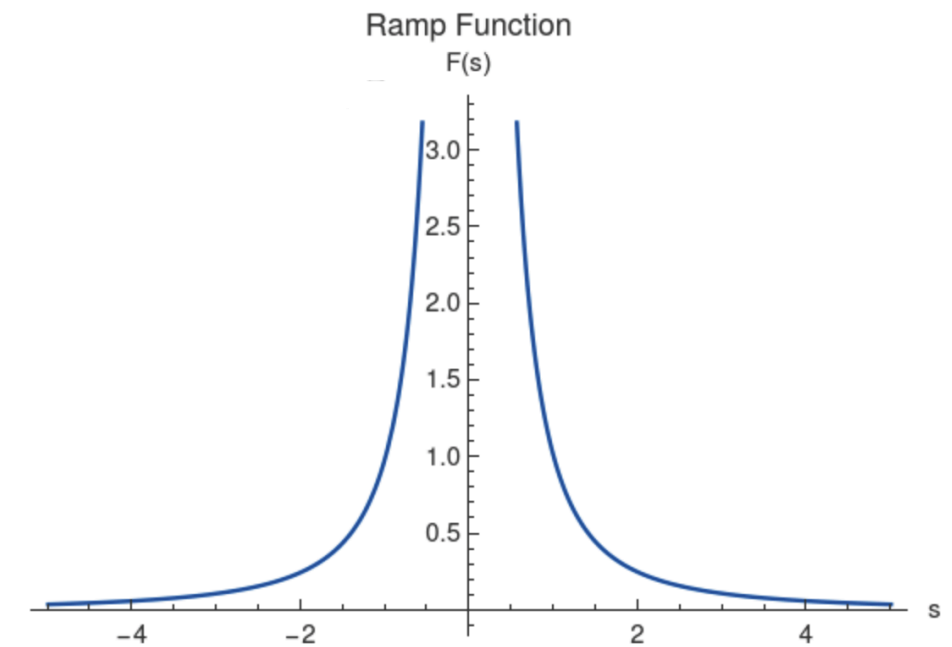
```
Plot[t, {t, -5, 5},  
  AxesLabel → {"t", "f(t)"}, PlotLabel → "Ramp Function"]
```



```
LaplaceTransform[t, t, s] // Quiet
```

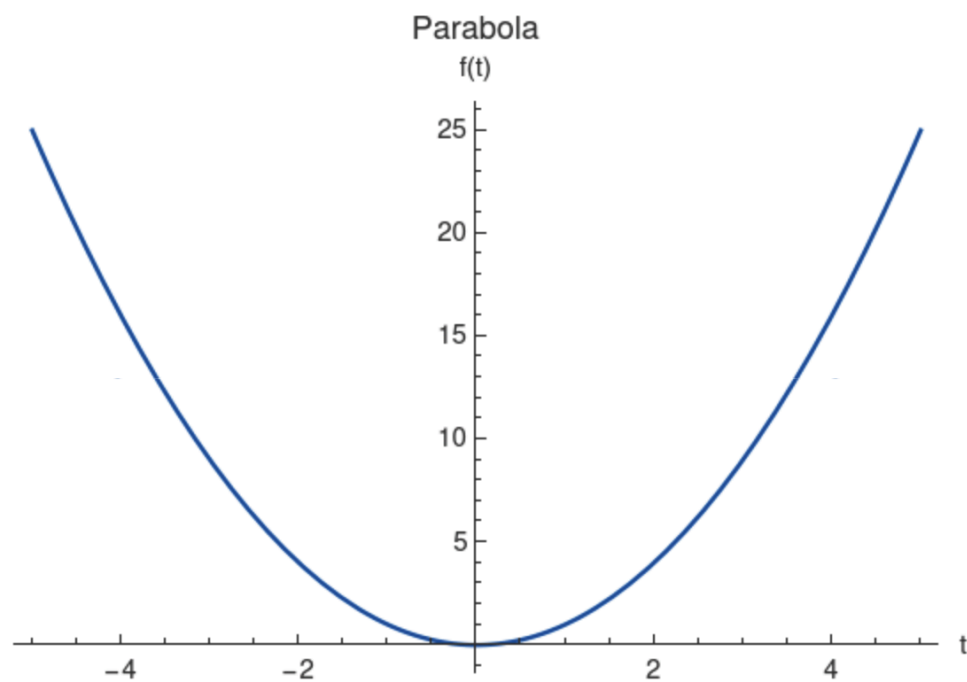
$$\frac{1}{s^2}$$

```
Plot[%, {s, -5, 5},  
  AxesLabel → {"s", "F(s)"}, PlotLabel → "Ramp Function"]
```



Parabola

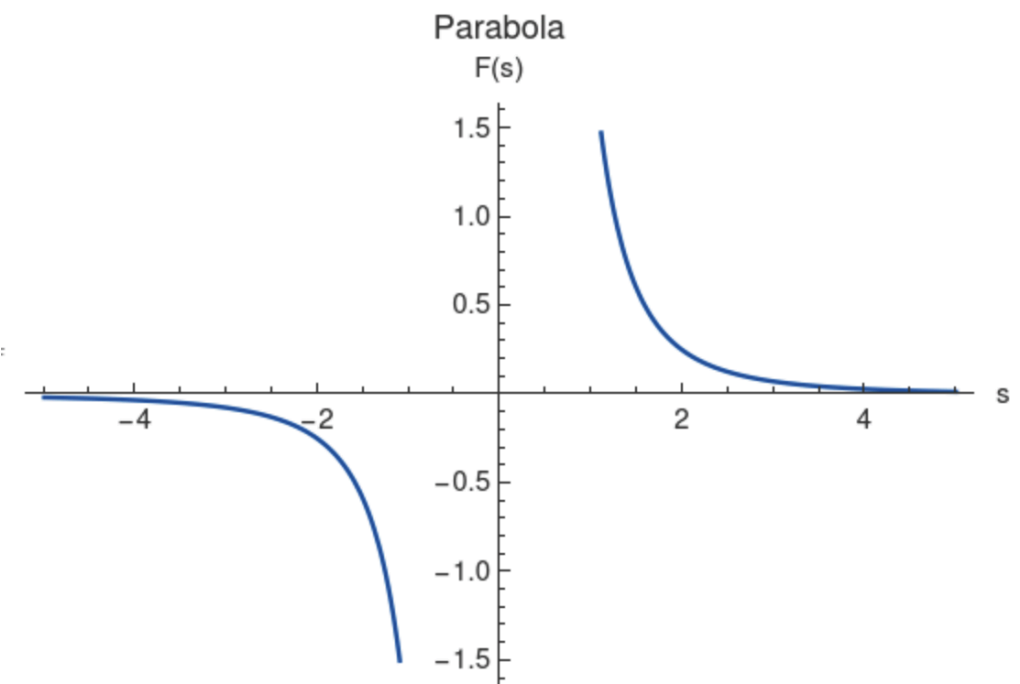
```
Plot[t2, {t, -5, 5},  
  AxesLabel → {"t", "f(t)"}, PlotLabel → "Parabola"]
```



```
LaplaceTransform[t2, t, s] // Quiet
```

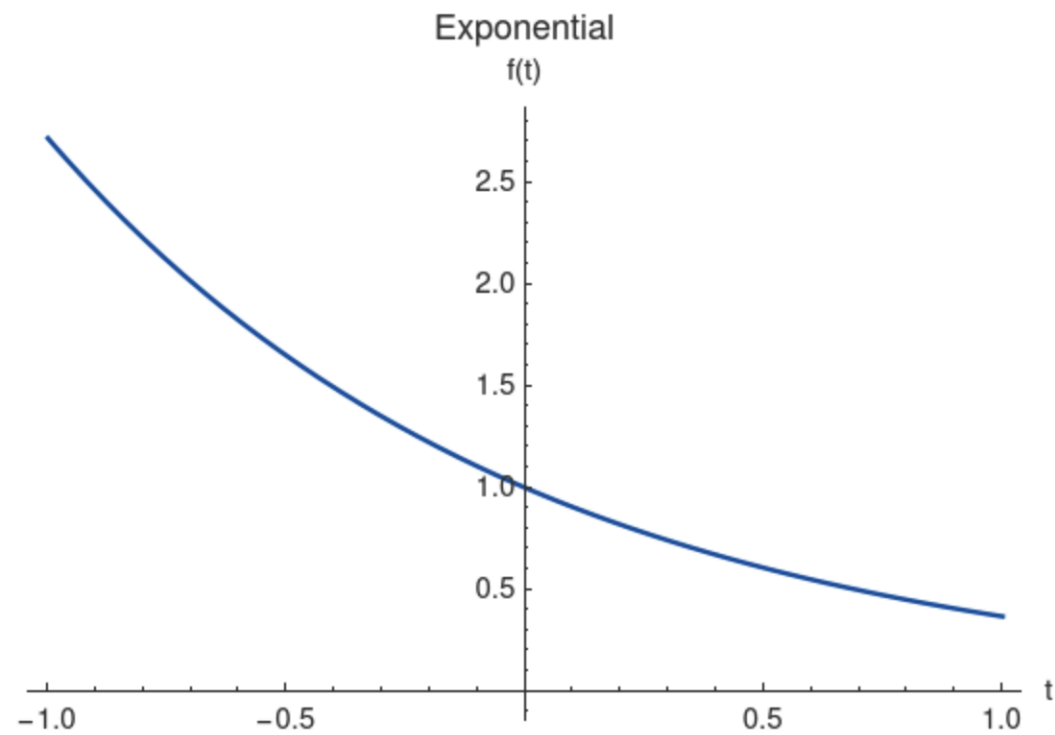
$$\frac{2}{s^3}$$

```
Plot[%, {s, -5, 5},  
  AxesLabel → {"s", "F(s)"}, PlotLabel → "Parabola"]
```



Exponential

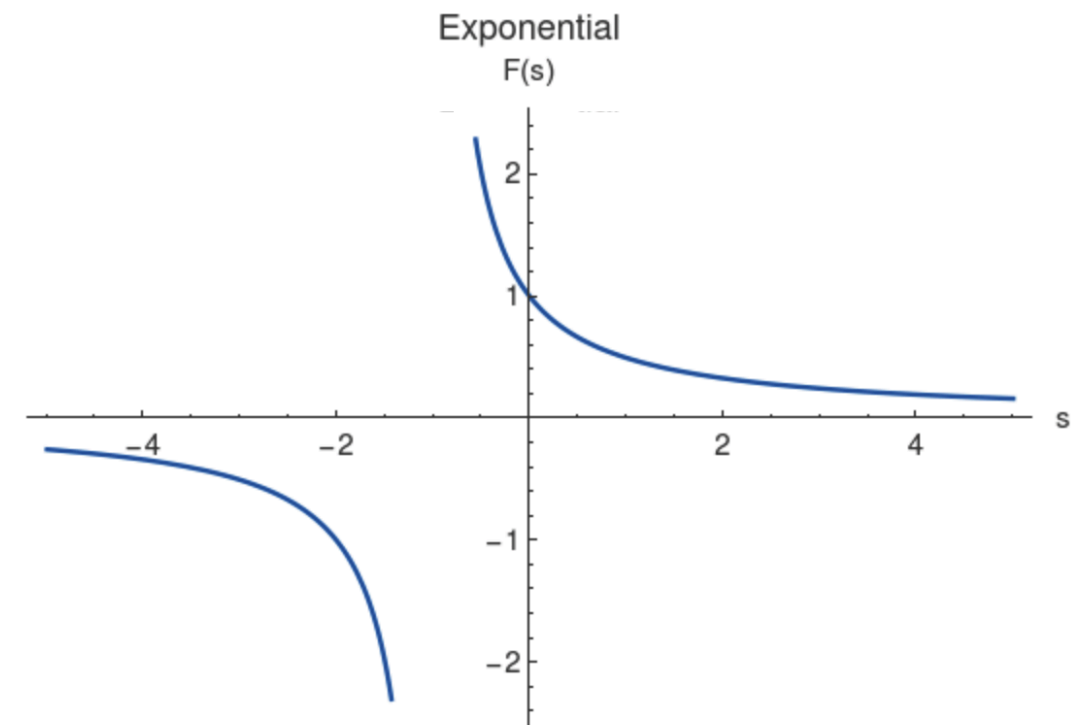
```
a = 1;  
Plot[e-a t, {t, -1, 1},  
  AxesOrigin → {0, 0}, AxesLabel → {"t", "f(t)"},  
  PlotLabel → "Exponential"]
```



```
LaplaceTransform[e-a t, t, s] // Quiet
```

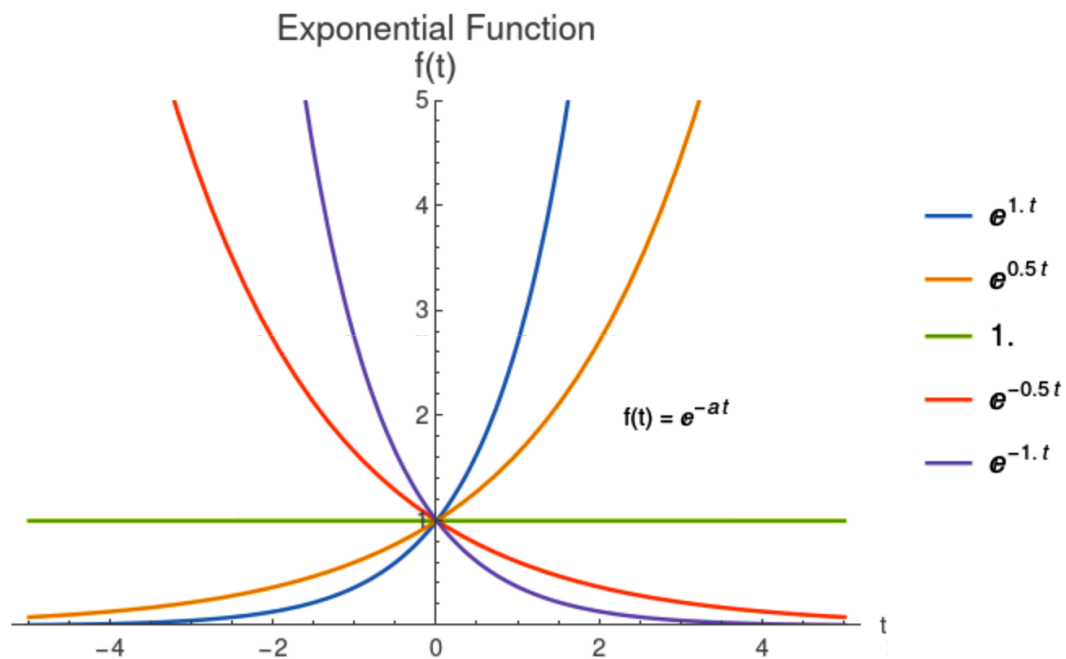
$$\frac{1}{1 + s}$$

```
Plot[%, {s, -5, 5},  
  AxesLabel → {"s", "F(s)"}, PlotLabel → "Exponential"]
```



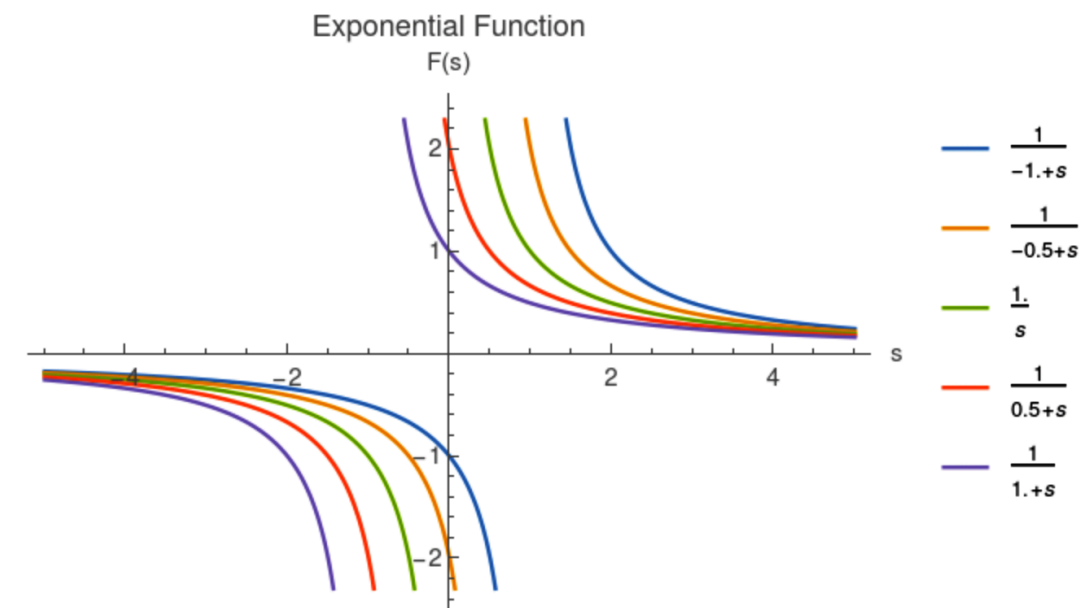
The [Table](#) command can be used to plot a range of coefficients.

```
f[t_] = Table[e-a t, {a, -1, 1, 0.5}];
Plot[Evaluate[f[t], {t, -5, 5}],
  PlotRange → {Automatic, {0, 5}},
  PlotLegends → "Expressions",
  PlotLabel → "Exponential Function",
  AxesLabel → {"t", "f(t)"},
  Epilog → {Text["f(t) = e-a t ", {3, 2}]}]
```



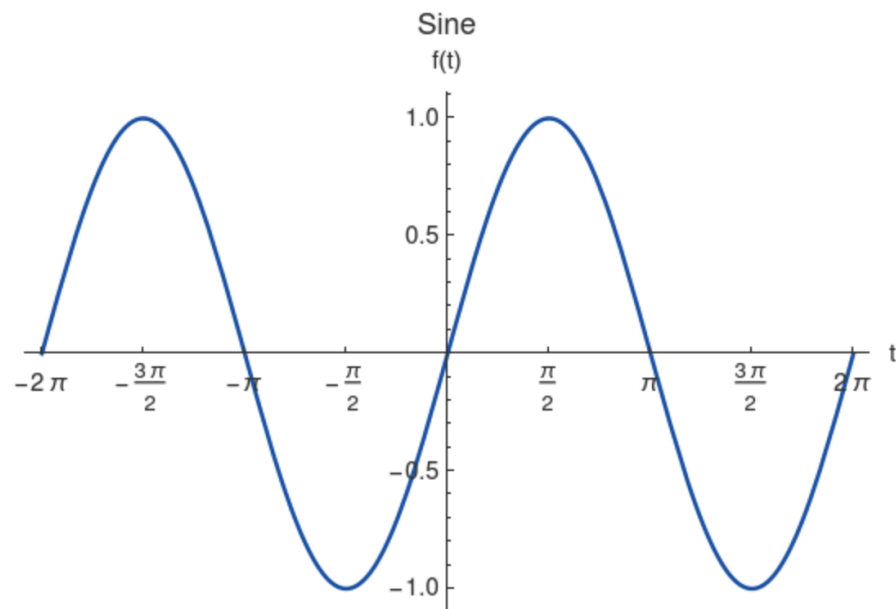
[PlotLegends](#) can be used to identify the individual plots.

```
LaplaceTransform[f[t], t, s] // Quiet
Plot[%, {s, -5, 5},
  AxesLabel → {"s", "F(s)"},
  PlotLegends → "Expressions",
  PlotLabel → "Exponential Function"]
{ $\frac{1}{-1.+s}$ ,  $\frac{1}{-0.5+s}$ ,  $\frac{1.}{s}$ ,  $\frac{1}{0.5+s}$ ,  $\frac{1}{1.+s}$ }
```



Sine Wave

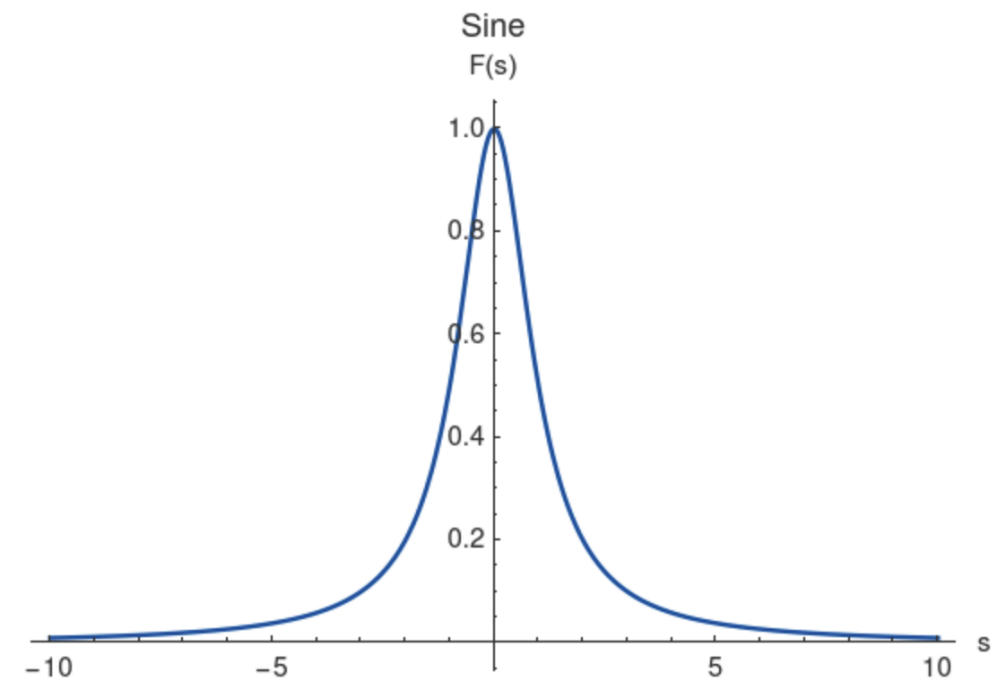
```
 $\omega = 1;$  (* normalized frequency *)  
Plot[Sin[ $\omega t$ ], {t, -2  $\pi$ , 2  $\pi$ },  
  Ticks  $\rightarrow$  {{-2  $\pi$ ,  $-\frac{3\pi}{2}$ ,  $-\frac{\pi}{2}$ , - $\pi$ ,  $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ , 2  $\pi$ }, Automatic},  
  AxesLabel  $\rightarrow$  {"t", "f(t)"},  
  PlotLabel  $\rightarrow$  "Sine"]
```



```
LaplaceTransform[Sin[ $\omega t$ ], t, s] // Quiet
```

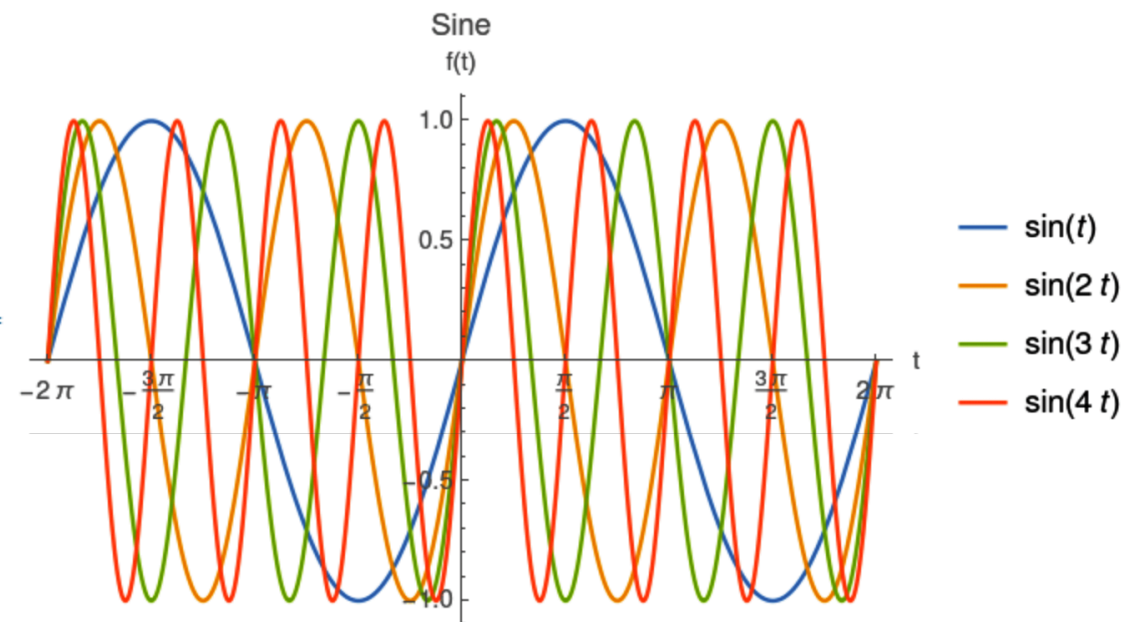
$$\frac{1}{1 + s^2}$$

```
Plot[%, {s, -10, 10},  
  PlotRange  $\rightarrow$  All,  
  AxesLabel  $\rightarrow$  {"s", "F(s)"}, PlotLabel  $\rightarrow$  "Sine"]
```

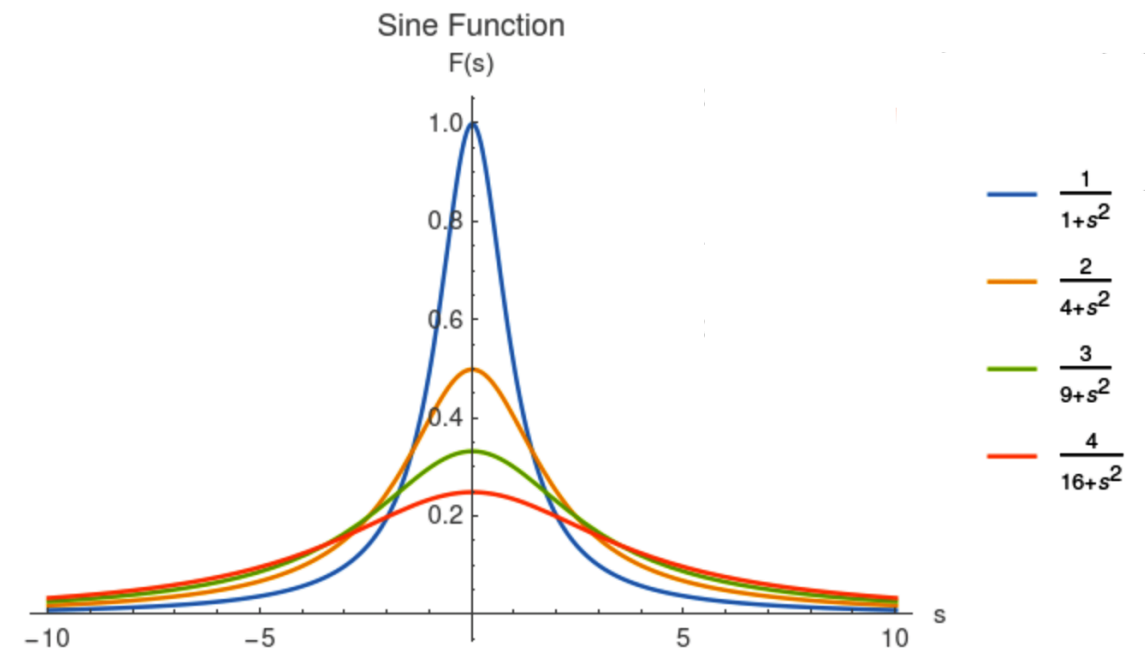


Several sine waves can be plotted simultaneously

```
ω = 1;
f[t_] = Table[Sin[n ω t], {n, 1, 4, 1}];
Plot[Evaluate[f[t], {t, -2 π, 2 π}],
  Ticks → {{-2 π, -3 π/2, -π, -π/2, π/2, π, 3 π/2, 2 π}, Automatic},
  PlotLegends → "Expressions",
  AxesLabel → {"t", "f(t)"},
  PlotLabel → "Sine"]
```

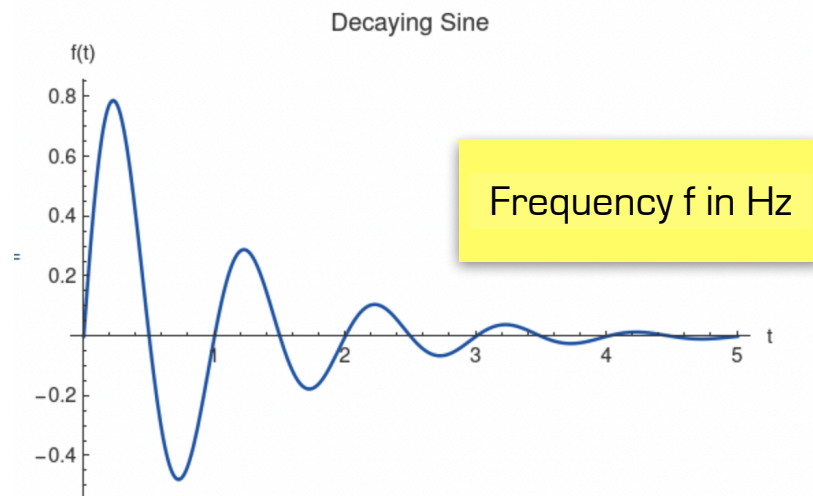


```
LaplaceTransform[f[t], t, s] // Quiet
Plot[%, {s, -10, 10},
  PlotRange → All,
  AxesLabel → {"s", "F(s)"},
  PlotLegends → "Expressions",
  PlotLabel → "Sine Function"]
{1/(1+s^2), 2/(4+s^2), 3/(9+s^2), 4/(16+s^2)}
```

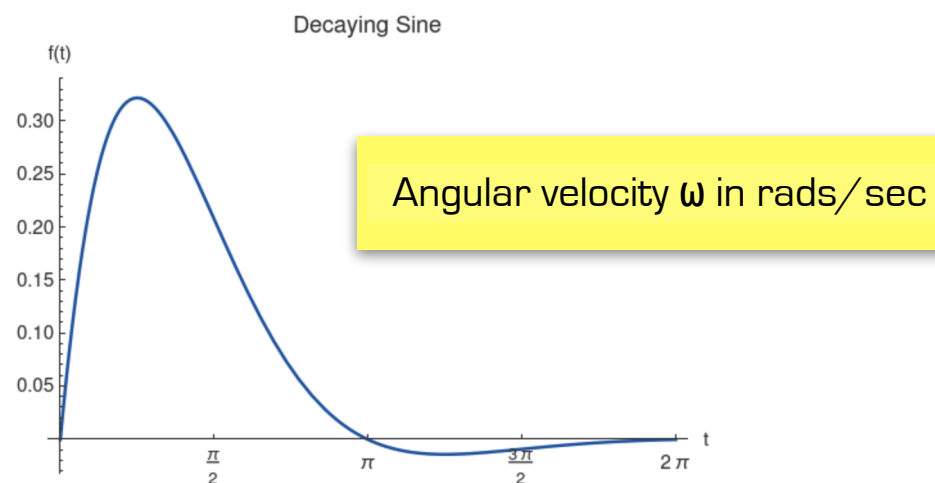


Decaying Sine Wave

```
a = 1; f = 1;
Plot[E^-a t Sin[2 π f t], {t, 0, 5},
  PlotRange → All,
  AxesLabel → {"t", "f(t)"},
  PlotLabel → "Decaying Sine"]
```



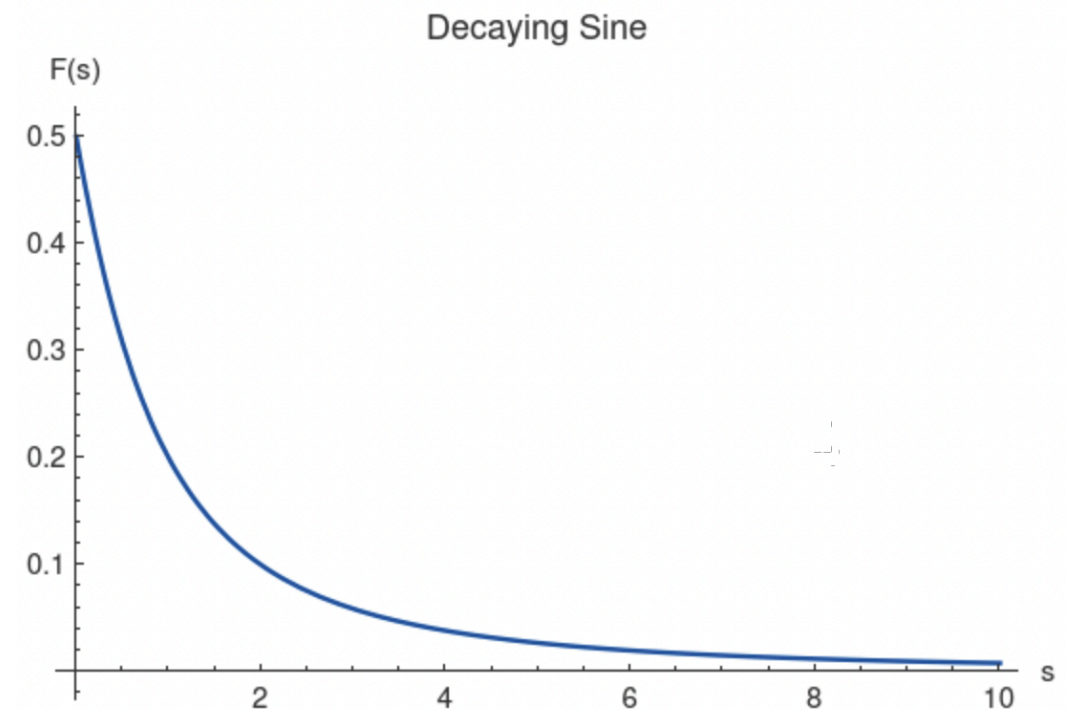
```
a = 1; ω = 1;
Plot[E^-a t Sin[ω t], {t, 0, 2 π},
  PlotRange → All,
  Ticks → {{-2 π, -3 π/2, -π/2, -π, π/2, π, 3 π/2, 2 π}, Automatic},
  AxesLabel → {"t", "f(t)"},
  PlotLabel → "Decaying Sine"]
```



```
LaplaceTransform[E^-a t Sin[ω t], t, s] // Quiet
```

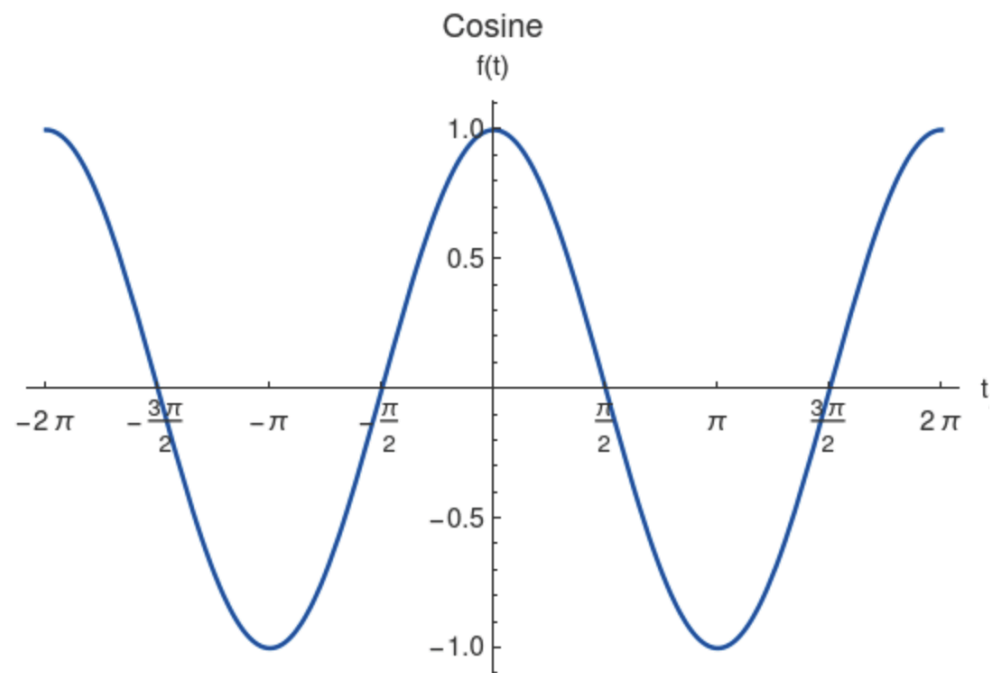
$$\frac{1}{1 + (1 + s)^2}$$

```
Plot[%, {s, 0, 10},
  PlotRange → All,
  AxesLabel → {"s", "F(s)"},
  PlotLabel → "Decaying Sine"]
```



Cosine Wave

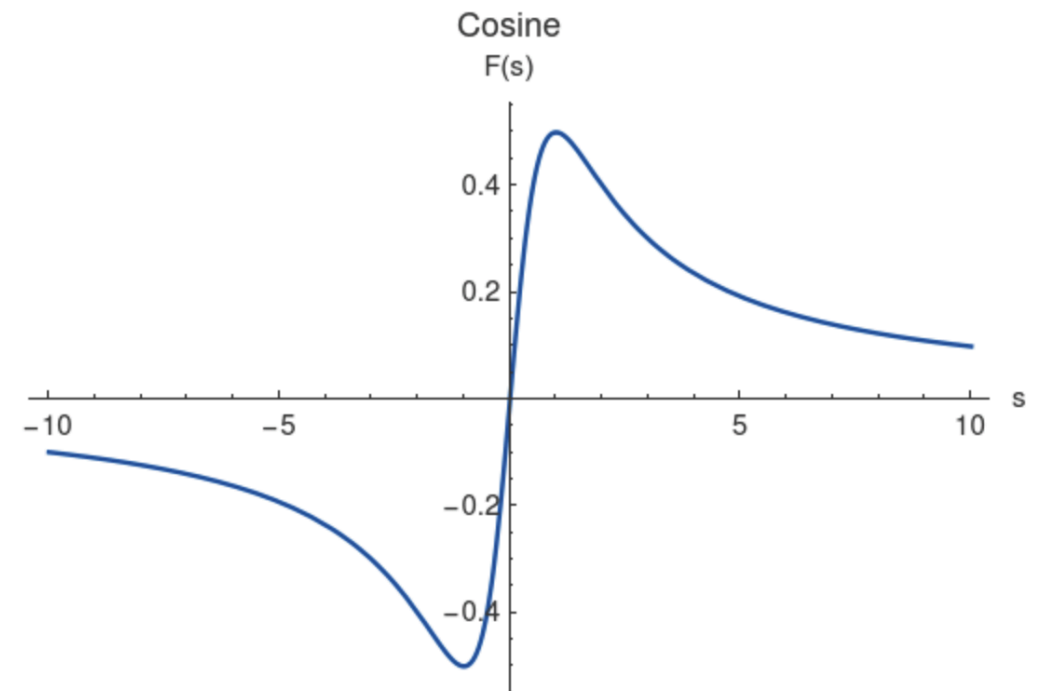
```
 $\omega = 1;$ 
Plot[Cos[ $\omega t$ ], {t, -2  $\pi$ , 2  $\pi$ },
  Ticks  $\rightarrow$  {{-2  $\pi$ ,  $-\frac{3\pi}{2}$ ,  $-\frac{\pi}{2}$ , - $\pi$ ,  $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ , 2  $\pi$ }, Automatic},
  AxesLabel  $\rightarrow$  {"t", "f(t)"},
  PlotLabel  $\rightarrow$  "Cosine"]
```



```
LaplaceTransform[Cos[ $\omega t$ ], t, s] // Quiet
```

$$\frac{s}{1 + s^2}$$

```
Plot[%, {s, -10, 10},
  AxesLabel  $\rightarrow$  {"s", "F(s)"},
  PlotLabel  $\rightarrow$  "Cosine"]
```

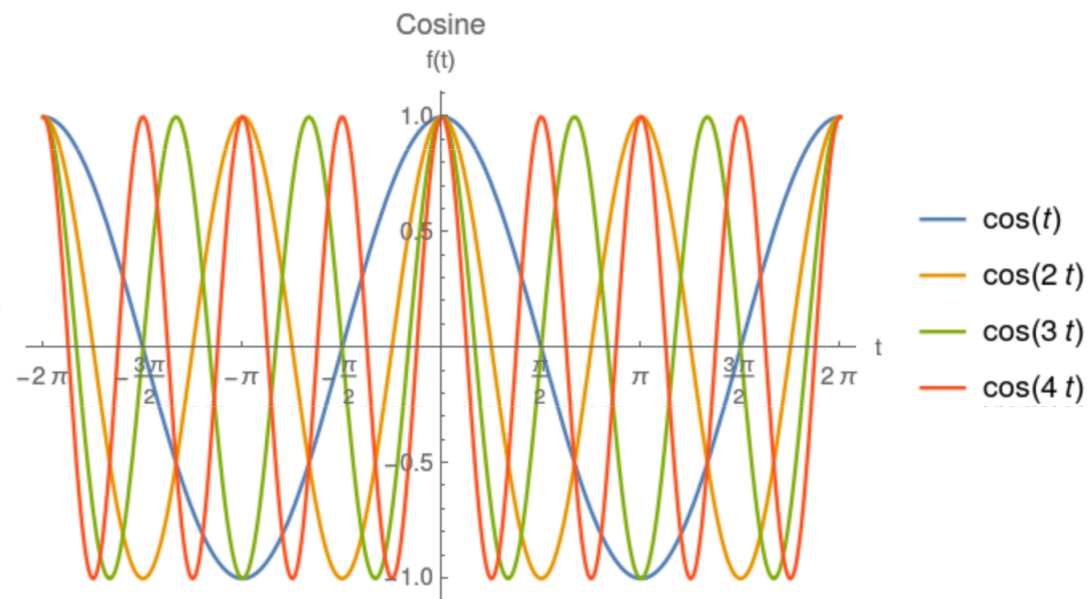


Several cosine waves can be plotted simultaneously

```

ω = 1;
f[t_] = Table[Cos[n ω t], {n, 1, 4, 1}];
Plot[Evaluate[f[t], {t, -2 π, 2 π}],
  Ticks → {{-2 π, -3 π/2, -π/2, -π, π/2, π, 3 π/2, 2 π}, Automatic},
  PlotLegends → "Expressions",
  AxesLabel → {"t", "f(t)"},
  PlotLabel → "Cosine"]

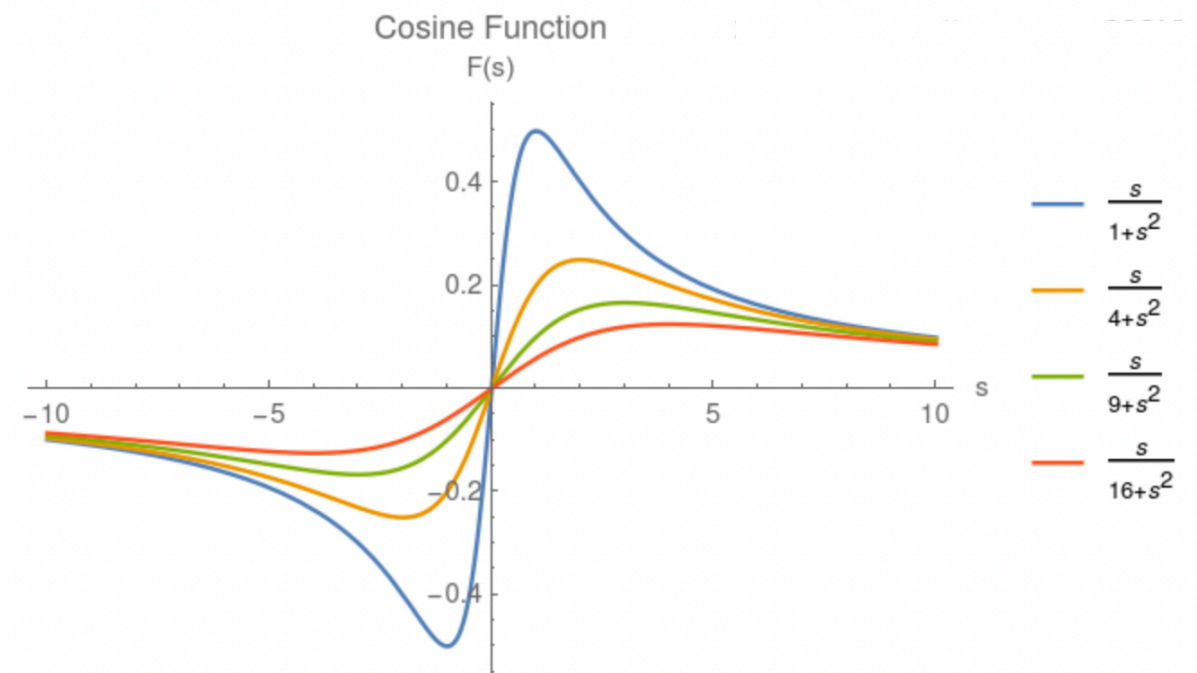
```



```

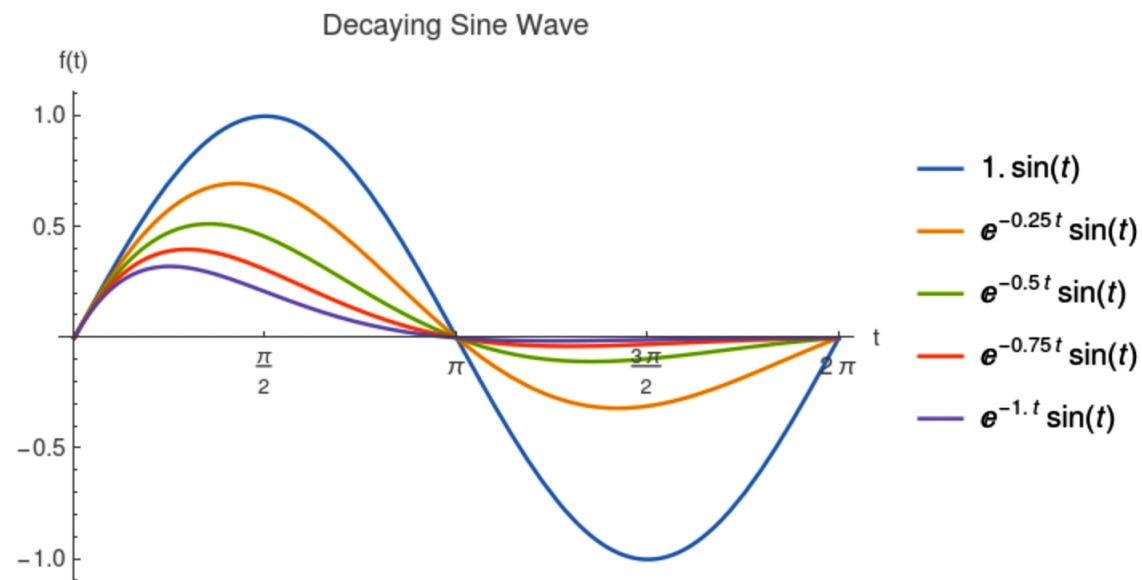
LaplaceTransform[f[t], t, s] // Quiet
Plot[%, {s, -10, 10},
  PlotRange → All,
  AxesLabel → {"s", "F(s)"},
  PlotLegends → "Expressions",
  PlotLabel → "Cosine Function"]
{
  s/(1 + s^2), s/(4 + s^2), s/(9 + s^2), s/(16 + s^2)
}

```



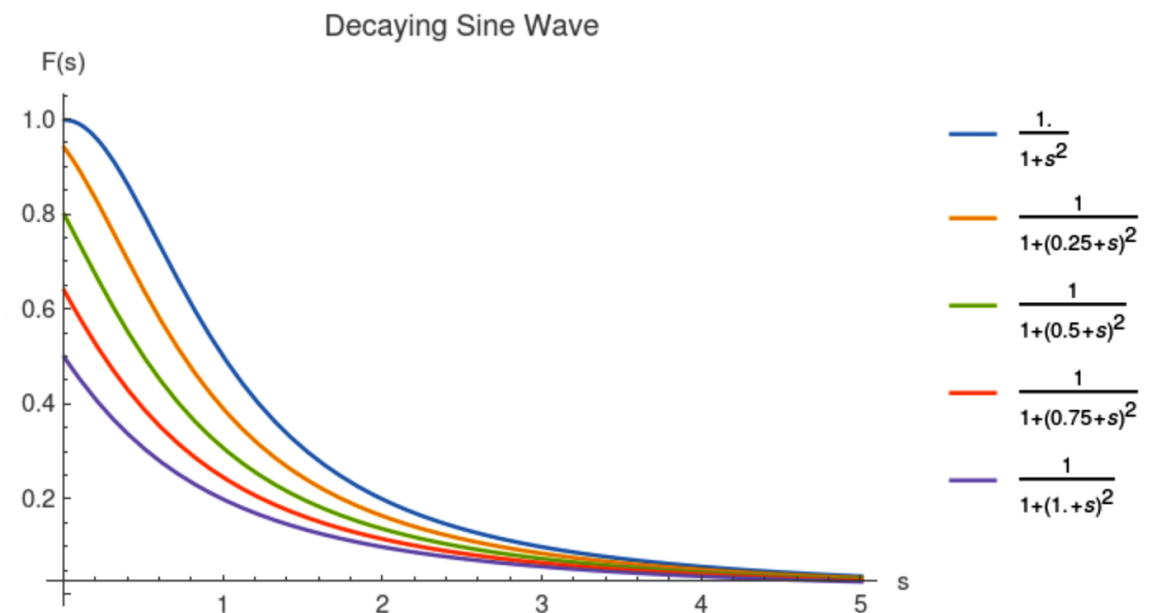
Several exponential decaying sine waves can be plotted simultaneously

```
Clear[f, ω, a, t]
ω = 1;
f[t_] = Table[e-a t Sin[ω t], {a, 0, 1, 0.25}];
Plot[Evaluate[f[t], {t, 0, 2 π}],
  Ticks → {{ $\frac{\pi}{2}$ , π,  $\frac{3\pi}{2}$ , 2 π}, Automatic},
  PlotLegends → "Expressions",
  AxesLabel → {"t", "f(t)"},
  PlotLabel → "Decaying Sine Wave"]
```



```
LaplaceTransform[f[t], t, s] // Quiet
Plot[%, {s, 0, 5},
  PlotRange → All,
  AxesLabel → {"s", "F(s)"},
  PlotLegends → "Expressions",
  PlotLabel → "Decaying Sine Wave"]
```

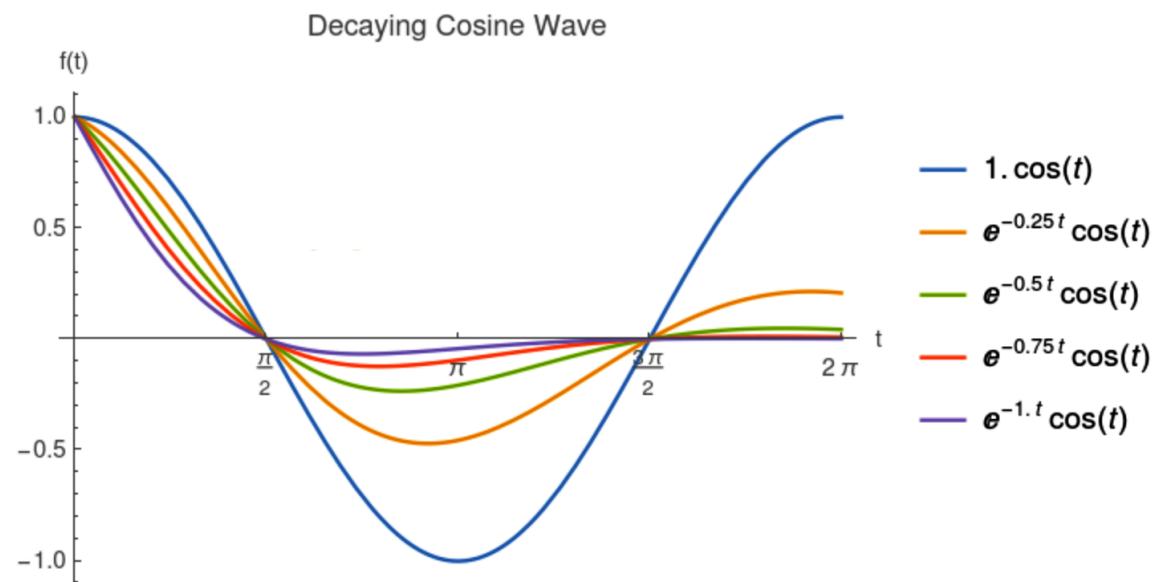
$$\left\{ \frac{1.}{1+s^2}, \frac{1}{1+(0.25+s)^2}, \frac{1}{1+(0.5+s)^2}, \frac{1}{1+(0.75+s)^2}, \frac{1}{1+(1.+s)^2} \right\}$$




```

Clear[f, ω, a, t]
ω = 1;
f[t_] = Table[e-a t Cos[ω t], {a, 0, 1, 0.25}];
Plot[Evaluate[f[t], {t, 0, 2 π}],
  Ticks → {{ $\frac{\pi}{2}$ , π,  $\frac{3\pi}{2}$ , 2 π}, Automatic},
  PlotLegends → "Expressions",
  AxesLabel → {"t", "f(t)"},
  PlotLabel → "Decaying Cosine Wave"]

```

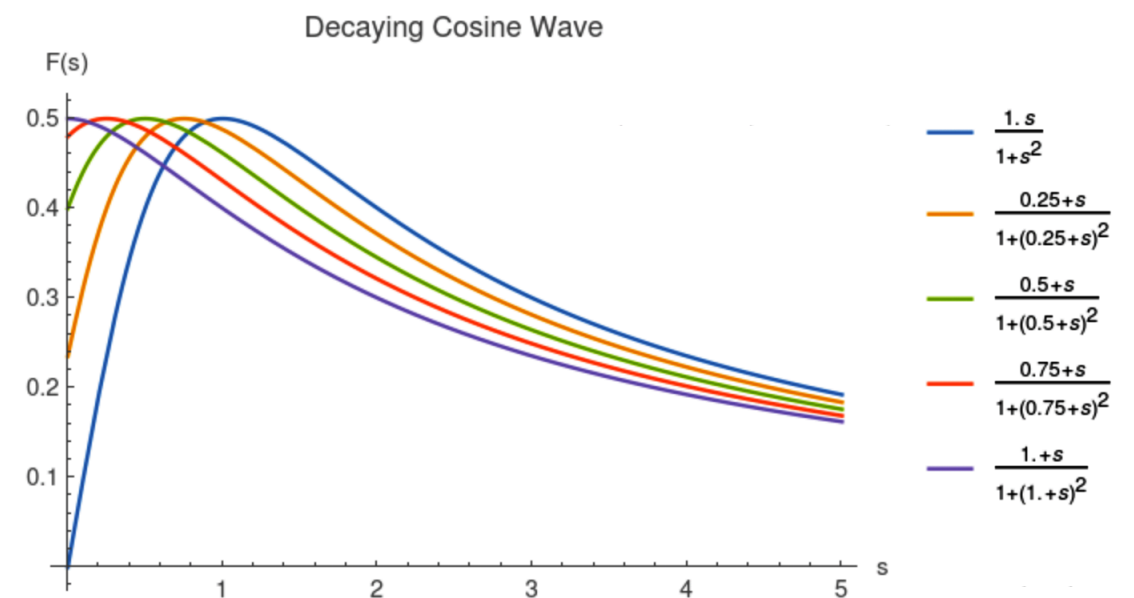


```

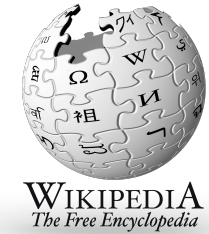
LaplaceTransform[f[t], t, s] // Quiet
Plot[%, {s, 0, 5},
  PlotRange → All,
  AxesLabel → {"s", "F(s)"},
  PlotLegends → "Expressions",
  PlotLabel → "Decaying Cosine Wave"]

```

$$\left\{ \frac{1. s}{1 + s^2}, \frac{0.25 + s}{1 + (0.25 + s)^2}, \frac{0.5 + s}{1 + (0.5 + s)^2}, \frac{0.75 + s}{1 + (0.75 + s)^2}, \frac{1. + s}{1 + (1. + s)^2} \right\}$$



Nyquist Channels



Class	Channel Name	Channel Response $H(\omega)$	Impulse Notation $f[\delta]$	Radix
1	Ideal LPF	1	1	2
1	Cosine [Duo-binary]	$2 \cos\left(\frac{\pi\omega}{2B}\right)$	1,1	3
2	Raised Cosine	$4 \cos^2\left(\frac{\pi\omega}{2B}\right)$	1,2,1	5
3		$2 + \cos\left(\frac{\pi\omega}{B}\right) - \cos\left(\frac{2\pi\omega}{B}\right) + j \left[\sin\left(\frac{\pi\omega}{B}\right) - \sin\left(\frac{2\pi\omega}{B}\right) \right]$	2,1,-1	5
4	Sine [Modified Duo-binary]	$2 \sin\left(\frac{\pi\omega}{B}\right)$	1,0,-1	3
5		$4 \sin^2\left(\frac{\pi\omega}{B}\right)$	1,0,-2,0,1	5

Controlling ISI (inter-symbol interference) forms the bases of M-ary signaling theory and allows the Nyquist rate to be exceeded. If a transmitted pulse waveform consists of sinc components, it is possible to separate the sinc components at the receiver, thus exceeding the Nyquist rate.

Overlapping sinc components of adjacent pulses eliminates ISI at the sampling instant, but creates m-ary amplitude data symbols.

Nyquist channels/filters/signals are sometimes called partial-response filters/signalling or correlative signalling/coding

The frequency response of the channel is not controlled by filters. It is controlled by **shaping** the time domain data pulses passing through the channel.

Nyquist Cosine Channel

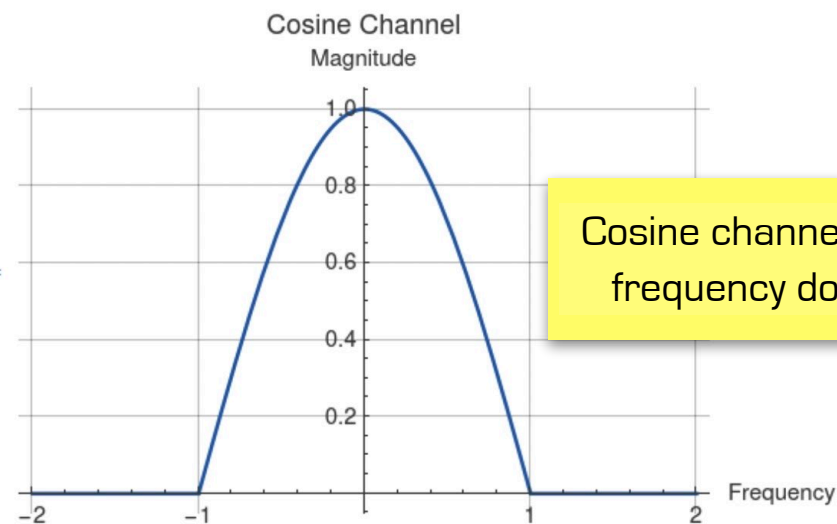
These are often used as individual data channels in wideband, FDM systems. The cosine channel is created by using a modified Sinc pulse as the information carrier. This minimizes both interference between consecutive data pulses in the time domain, but also adjacent channel interference in the frequency domain.

Frequency Domain

```
B = 1; (*Nyquist bandwidth*)
cosinechannel = Cos[ $\frac{\pi f}{2B}$ ];

cosf = Piecewise[{{0, f < -B}, {cosinechannel, f ≤ B}, {0, f > B}}];

Plot[cosf, {f, -2, 2},
  AxesLabel → {"Frequency", "Magnitude"},
  PlotLabel → "Cosine Channel",
  GridLines → Automatic]
```



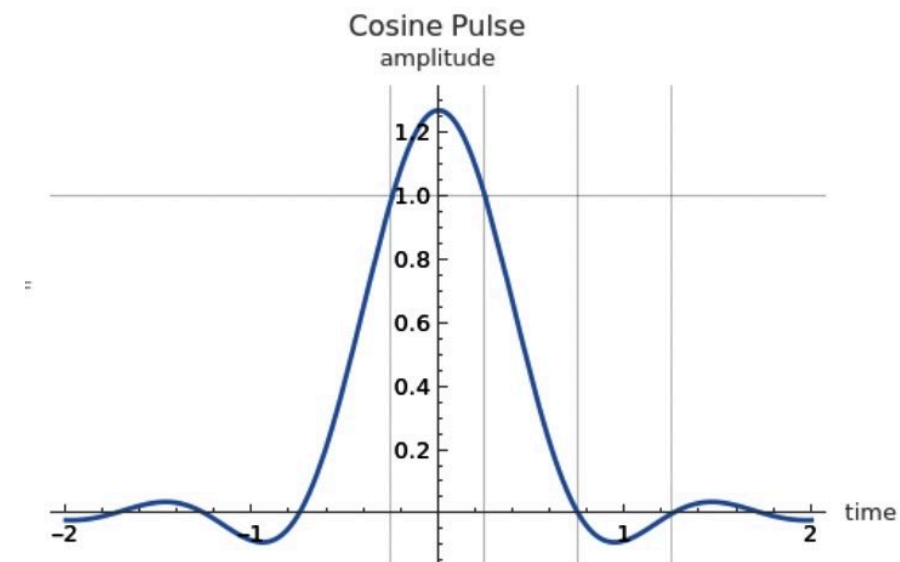
Cosine channel in the frequency domain

Time Domain

```
cost = InverseFourierTransform[cosf, f, t, FourierParameters → {0, -2π}]
=  $-\frac{4 \cos[2\pi t]}{\pi(-1 + 16t^2)}$ 
```

```
cost = - $\frac{4 \cos[2\pi t]}{\pi(-1 + 16t^2)}$ ;
Plot[cost, {t, -2, 2},
  PlotLabel → "Cosine Pulse",
  AxesLabel → {"time", "amplitude"},
  PlotRange → All,
  GridLines → {{-0.25, 0.25, 0.75, 1.25}, {1}}]
```

The InverseFourierTransform can be used to determine the time domain pulse.



Cosine or Duobinary Pulse

The term duobinary is somewhat confusing, because there are two definitions:

- a pulse with **3 possible values**: -1, 0, 1
- a pulse which can be decomposed into **2 sinc pulses**

Normally sequential data pulses (or symbols) must not overlap. In this case, it would seem that a second pulse could be centered at $t = 0.75$ since at that instant only, the two pulses are distinguishable.

However, it is possible to separate sequential data pulses by $t = 0.5$ since the leading sinc component of one pulse completely overlaps the trailing sinc component of the adjacent data pulse.

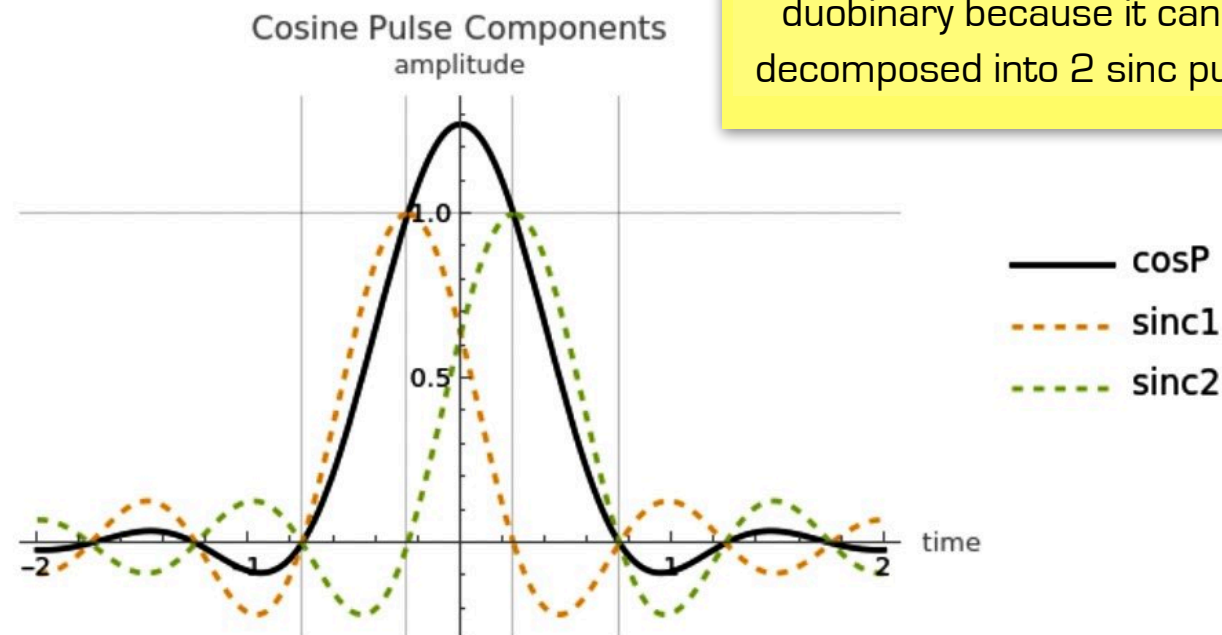
```
Clear[cp, sinc1, sinc2]
```

$$\text{sinc1} = \frac{\sin[2\pi(t + 0.25)]}{2\pi(t + 0.25)};$$

$$\text{sinc2} = \frac{\sin[2\pi(t - 0.25)]}{2\pi(t - 0.25)};$$

```
cosP = sinc1 + sinc2;
```

```
Plot[{cosP, sinc1, sinc2}, {t, -2, 2},
  PlotRange -> All,
  PlotStyle -> {Black, Dashed, Dashed},
  GridLines -> {{-0.75, -0.25, 0.25, 0.75}, {1}},
  PlotLabel -> "Cosine Pulse Components",
  AxesLabel -> {"time", "amplitude"},
  PlotLegends -> "Expressions"]
```

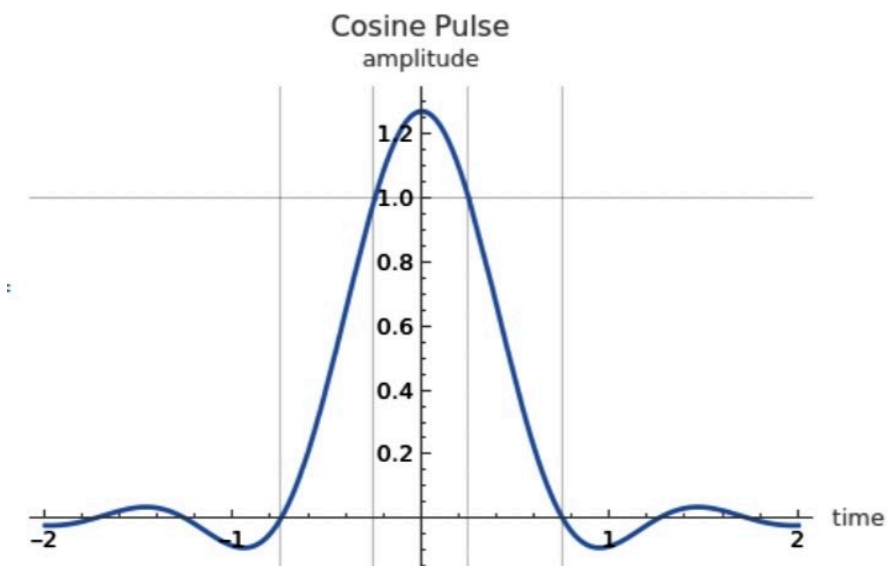


The cosine pulse is also called duobinary because it can be decomposed into 2 sinc pulses

```
Clear[ss, k, cosPulse]
```

$$\text{cosPulse}[t_] = -\frac{4 \cos[2\pi t]}{\pi(-1 + 16t^2)}; (*\text{cosine pulse}*)$$

```
Plot[cosPulse[t], {t, -2, 2},
  PlotRange -> All,
  PlotLabel -> "Cosine Pulse",
  AxesLabel -> {"time", "amplitude"},
  GridLines -> {{-0.75, -0.25, 0.25, 0.75}, {1}}]
```



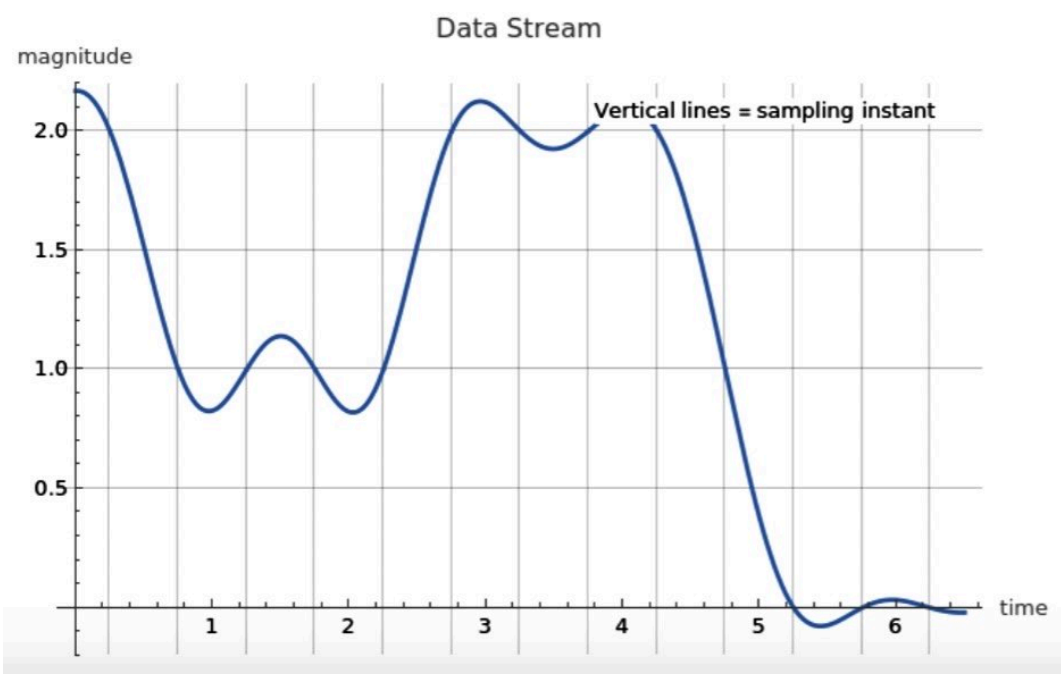
Random Duobinary Data Stream

```
ss = 0.5; (*symbol spacing*)
so = 0.25; (*sinc offset*)

nos = 12; (*number of symbols*)
rData = RandomChoice[{0, 1}, nos]; (*random data*)

output[t_] =  $\sum_{k=1}^{nos} (rData[[k]]) \cos Pulse[t - (k - 1) ss + 1];$ 
```

```
Plot[output[t], {t, 0,  $\frac{nos + 1}{2}$ },
  PlotRange → {All, {-0.2, 2.2}},
  PlotLabel → "Data Stream",
  AxesLabel → {"time", "magnitude"},
  GridLines → {Table[k, {k, so, nos, ss}], {0, 0.5, 1, 1.5, 2}},
  Epilog →
    {Text[Row[{"Vertical lines = sampling instant"},
      Background → White], Scaled[{0.95, 0.95}], Right]}]
```



Sampling the data stream makes it easy to create an eye diagram

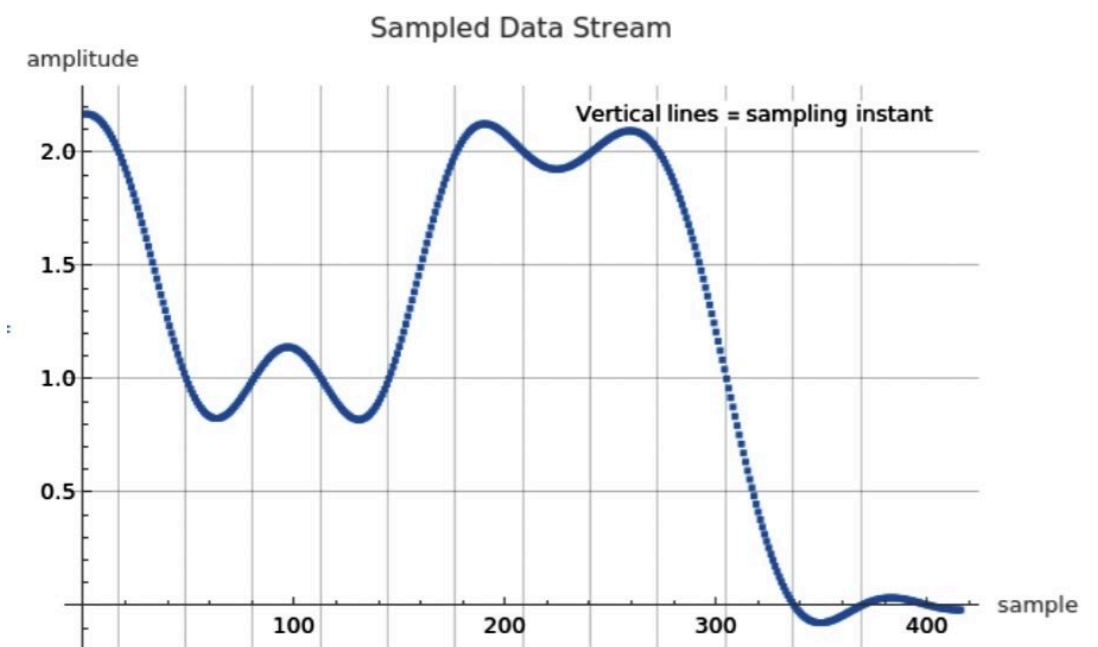
Sampled Duobinary Data Stream

```
Clear[sps, signal, n]

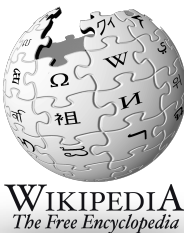
sps = 32; (* samples per symbol *)

signal = Table[output[n], {n, 0.000001, (nos + 1) ss,  $\frac{ss}{sps}$ }};

ListPlot[signal,
  PlotRange → All,
  PlotLabel → "Sampled Data Stream",
  AxesLabel → {"sample", "amplitude"},
  GridLines → {Table[n, {n, (sps + 1) ss, nos sps, sps}],
    {0.5, 1, 1.5, 2}},
  Epilog →
    {Text[Row[{"Vertical lines = sampling instant"},
      Background → White], Scaled[{0.95, 0.95}], Right]}]
```

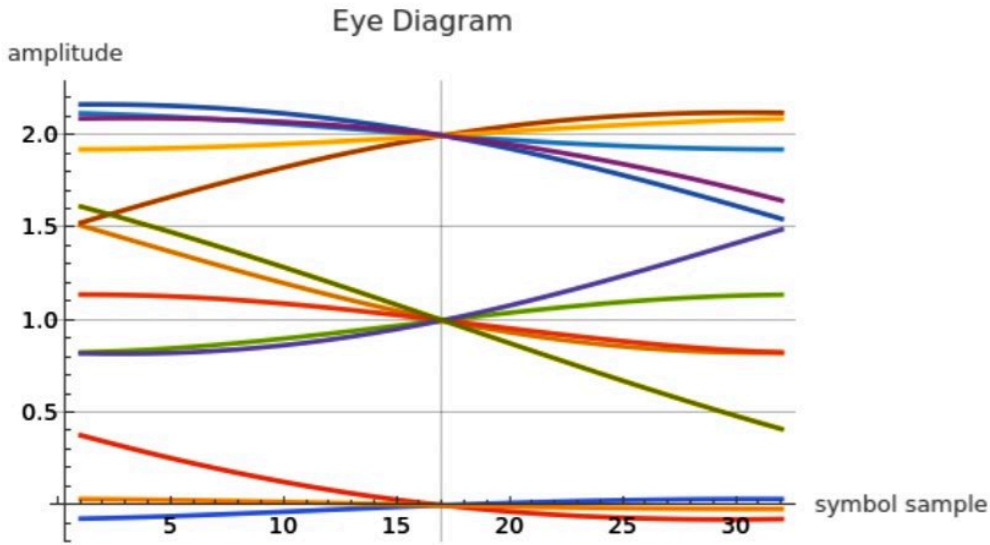


Duobinary Eye Diagram

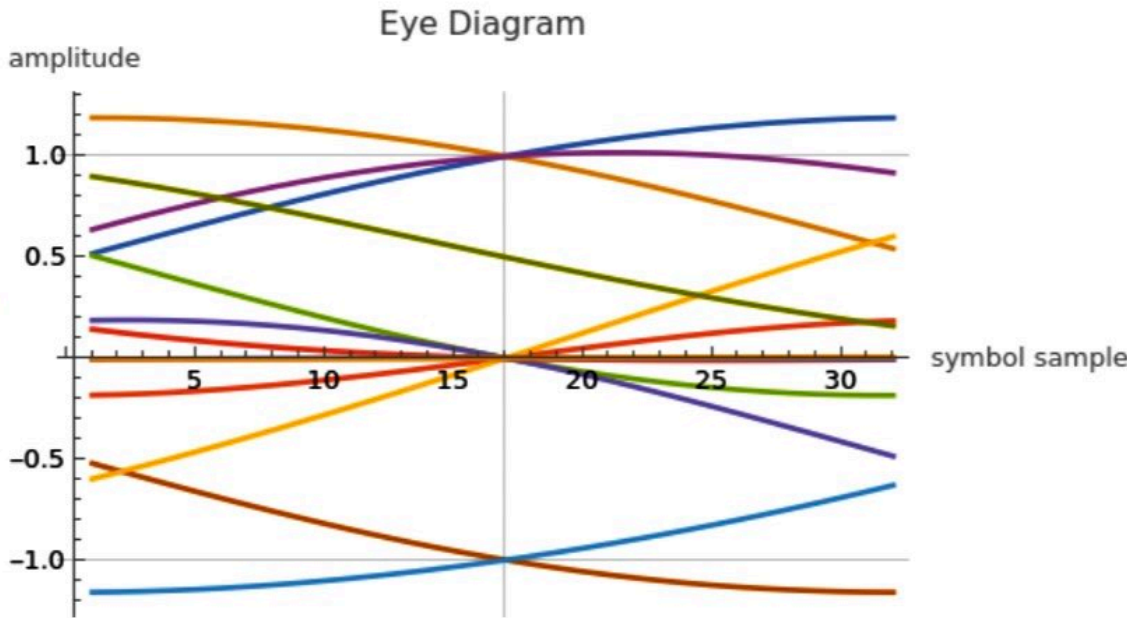


```
slice = Partition[signal, Round[sps]];
ListLinePlot[slice,
  PlotRange -> All,
  PlotLabel -> "Eye Diagram",
  AxesLabel -> {"symbol sample", "amplitude"},
  GridLines -> {{ $\frac{sps}{2} + 1$ }, {0.5, 1, 1.5, 2}}]
```

There is no ISI at the sampling instant.



Unipolar pulses

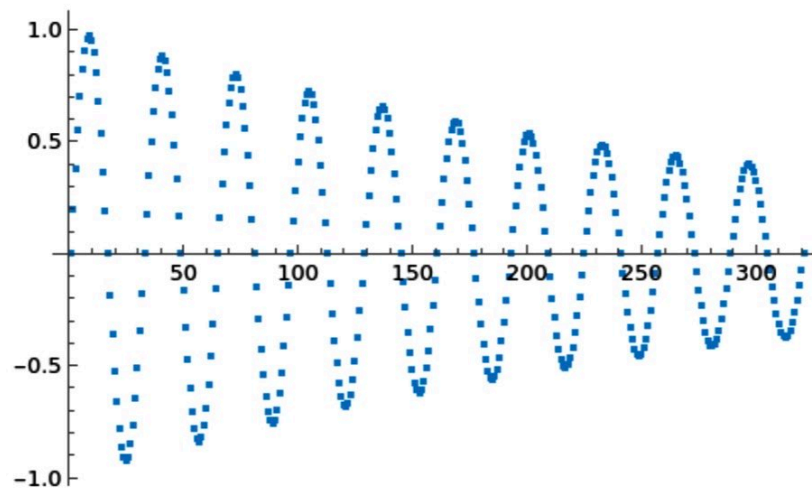


Bipolar pulses

Partition Command

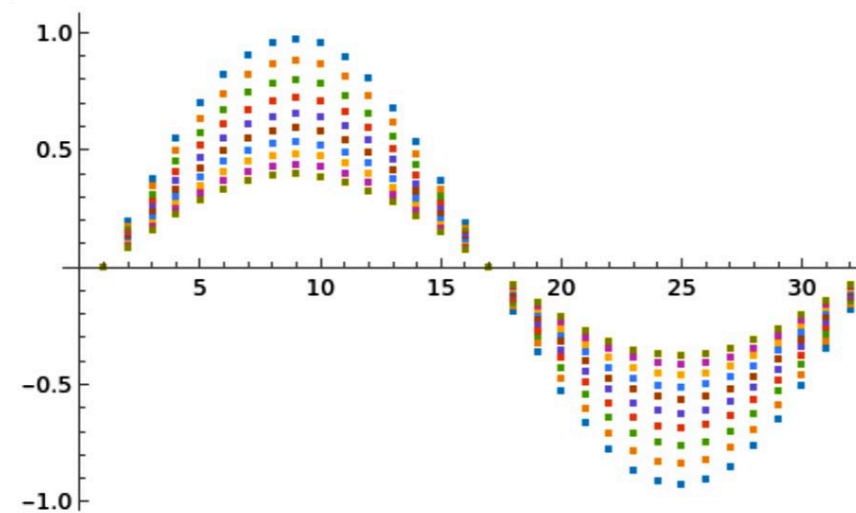
Partition can be used to create an eye diagram

```
f = 10;  
sps = 32; (*samples per symbol*)  
signal = Table[E^(-t) Sin[2 π f t], {t, 0, 1, 1/f sps}];  
ListPlot[signal]
```



A sampled signal

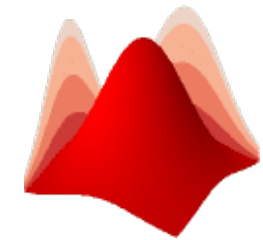
```
slice = Partition[signal, sps];  
ListPlot[slice] (*the output starts at sample 1*)
```



Each cycle overlaid

Sine or Modified Duo-Binary Channel

Like the cosine channel, the sine channel is composed of two sinc pulses except that they are of opposite polarity. This technique works in systems that cannot carry a DC signal such as circuits using capacitor or transformer coupling.



```
Clear[B, sine $\omega$ , sinet, sinechannel, sinech]
```

```
sinechannel = Abs[Sin[ $\frac{\pi \omega}{B}$ ]];
```

```
B = 1;
```

```
sinech =
```

```
Piecewise[{{0,  $\omega < -B$ }, {sinechannel,  $\omega \leq B$ }, {0,  $\omega > B$ }}];
```

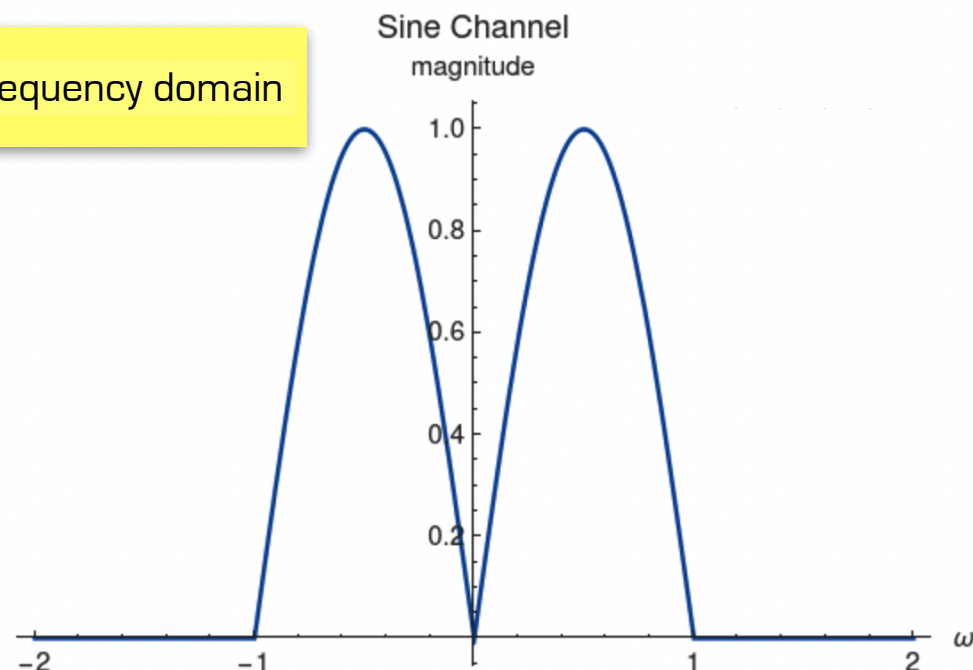
```
Plot[sinech, { $\omega$ , -2, 2},
```

```
PlotRange → All,
```

```
AxesLabel → {" $\omega$ ", "magnitude"},
```

```
PlotLabel → "Sine Channel"]
```

Frequency domain



The Sine Channel pulse can be determined by taking the Inverse Fourier Transform of the low pass sine channel

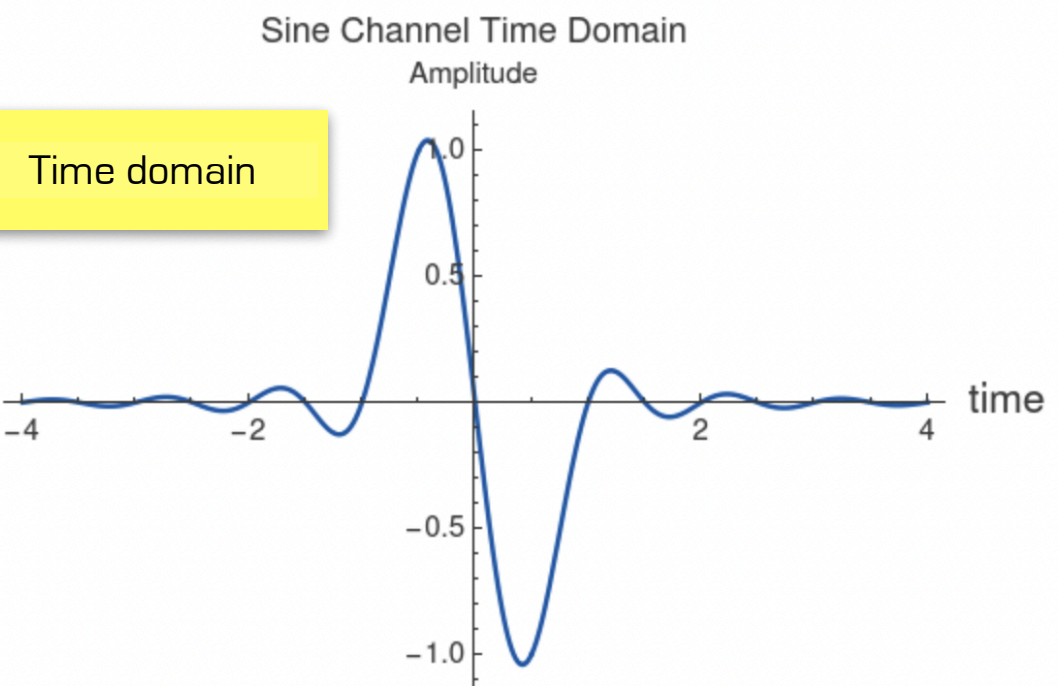
```
sinet = InverseFourierSinTransform[sinech,  $\omega$ , t,  
FourierParameters → {0, -2  $\pi$ }];
```

```
Plot[sinet, {t, -4, 4},
```

```
PlotRange → All,
```

```
AxesLabel → {"time", "Amplitude"},
```

```
PlotLabel → "Sine Channel Time Domain"]
```



Time domain

Modified Duo-Binary Pulse Composition

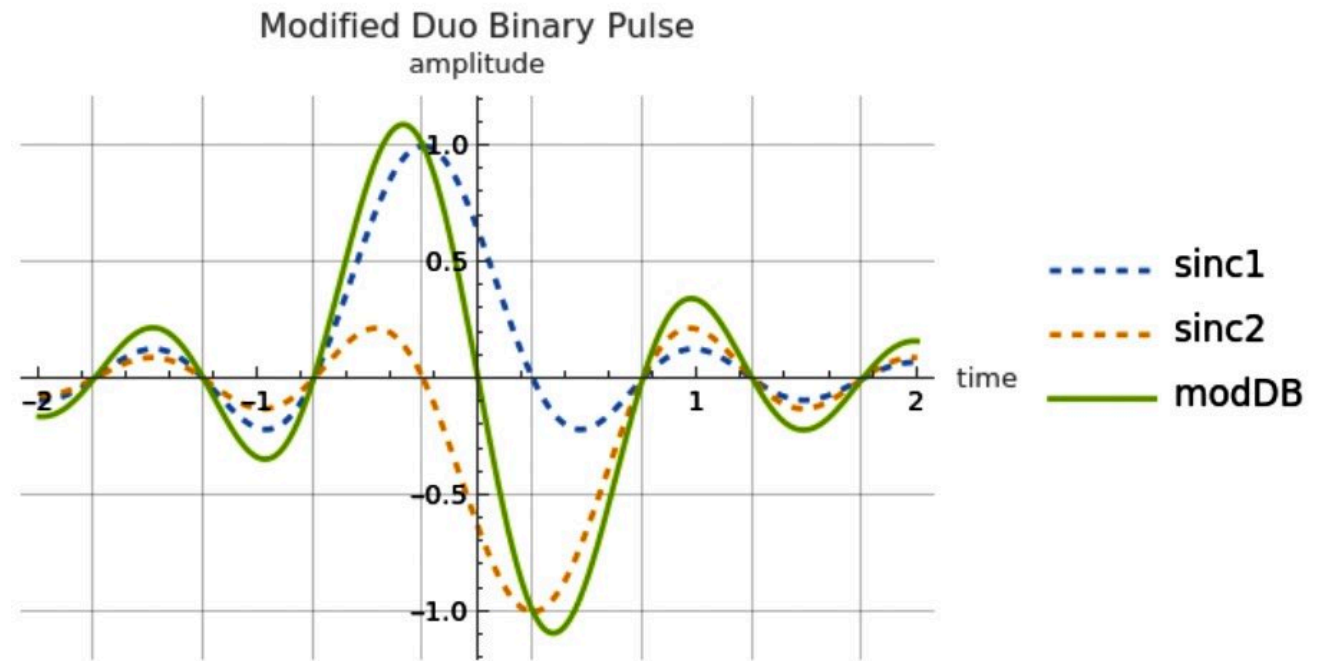
```
Clear[sinc1, sinc2, t, modDB]
```

$$\text{sinc1} = \frac{\sin[2\pi(t + 0.25)]}{2\pi(t + 0.25)};$$

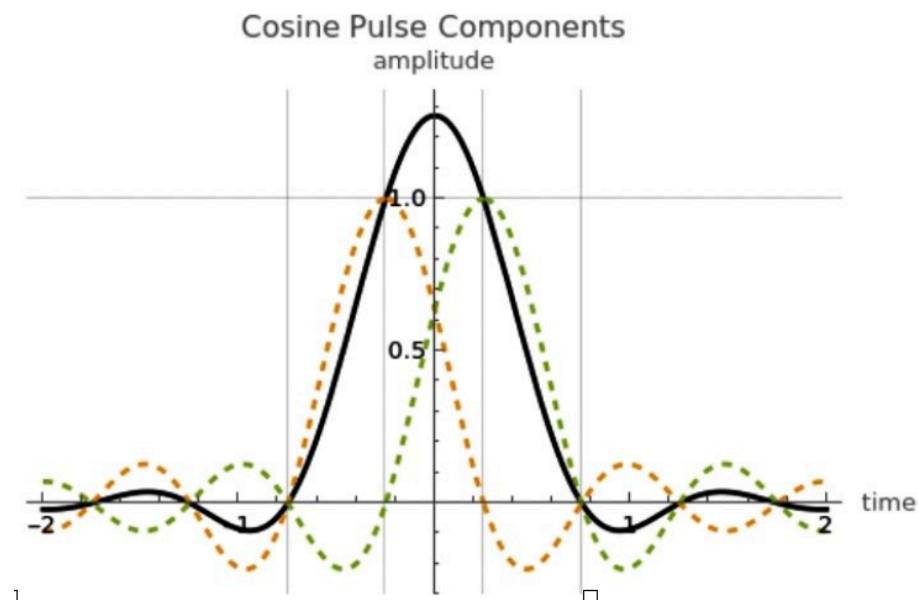
$$\text{sinc2} = \frac{-\sin[2\pi(t - 0.25)]}{2\pi(t - 0.25)};$$

```
modDB = sinc1 + sinc2;
```

```
Plot[{sinc1, sinc2, modDB}, {t, -2, 2},
  GridLines -> {Table[k, {k, -2.25, 2, 0.5}], Automatic},
  PlotStyle -> {Dashed, Dashed, Thick},
  PlotLabel -> "Modified Duo Binary Pulse",
  AxesLabel -> {"time", "amplitude"},
  PlotLegends -> "Expressions"]
```



The duo binary pulse can easily be modified by inverting one of the sinc components.



Sine Data Stream & Noise

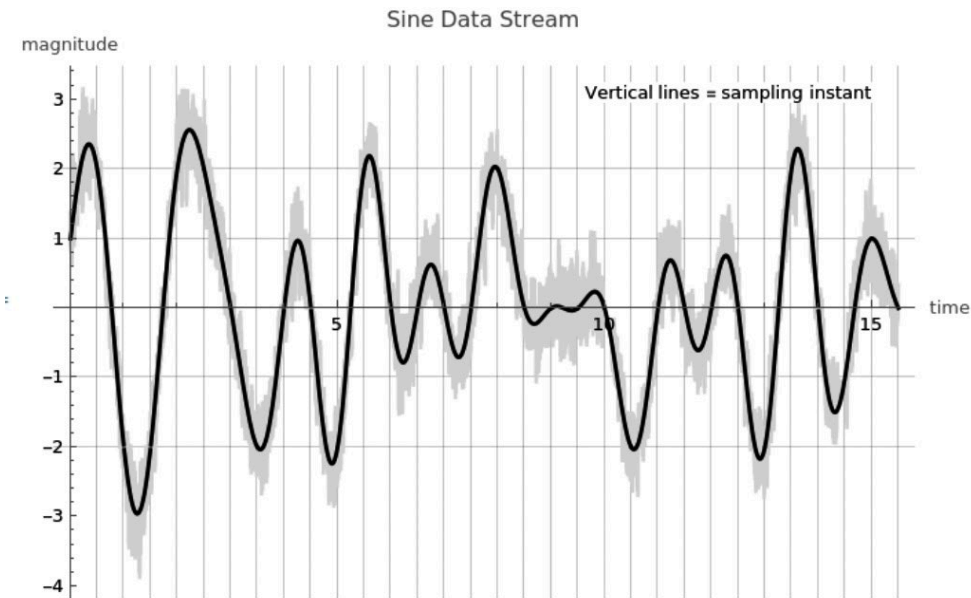
```
= Clear[k]
ss = 0.5; (*symbol spacing*)

= nos = 30; (*number of symbols*)
rData = RandomChoice[{-1, 1}, nos]; (*random data*)

= output[t_] = Sum[rData[[k]] * sinePulse[t - (k - 1) ss + 10^-5],
  {k, 1, nos}];
(*can't seem to be able to add noise to this summation*)

rn = 0.1;
noise := RandomReal[{-rn, rn}];
outputNoise[t_] :=
  Sum[(rData[[k]] (sinePulse[(t - (k - 1) ss) + 10^-5] + noise),
    {k, 1, nos});

Plot[{outputNoise[t], output[t]}, {t, 0, (nos + 1) ss},
  PlotRange -> All,
  PlotLabel -> "Sine Data Stream",
  AxesLabel -> {"time", "magnitude"},
  GridLines -> {Table[k, {k, ss, nos, ss}], {-2, -1, 1, 2}},
  PlotStyle -> {LightGray, Black},
  Epilog ->
    {Text[Row[{"Vertical lines = sampling instant"},
      Background -> White], Scaled[{0.95, 0.95}], Right]}]
```



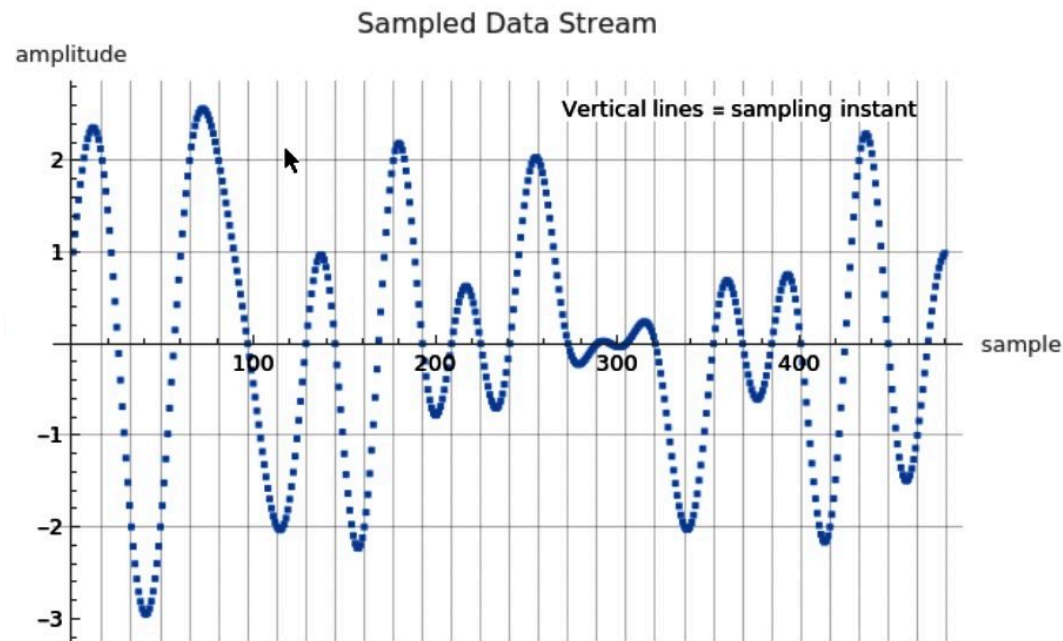
Sampled Duobinary Data Stream

```
Clear[sps, signal, n]

sps = 32; (* samples per symbol *)

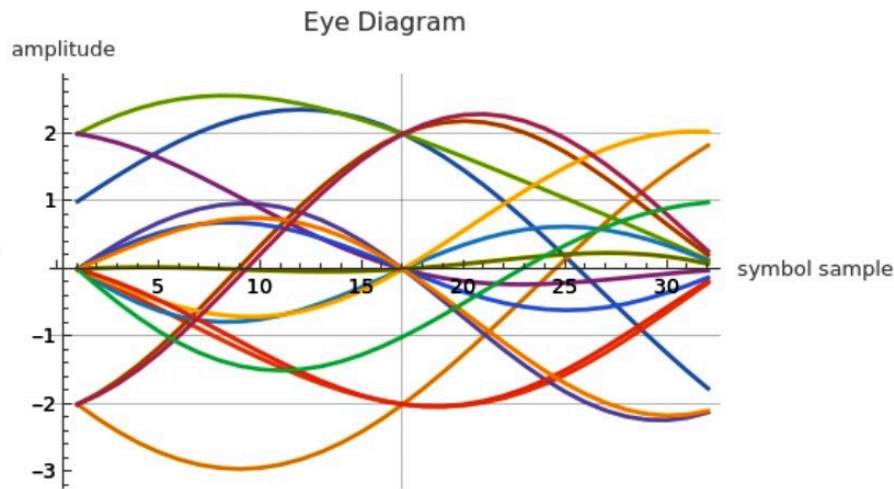
signal = Table[output[n], {n, 0.000001, nos ss, 1/sps}];

ListPlot[signal,
  PlotRange -> All,
  PlotLabel -> "Sampled Data Stream",
  AxesLabel -> {"sample", "amplitude"},
  GridLines -> {Table[n, {n, sps ss + 1, nos sps, sps ss}],
    {-2, -1, 1, 2}},
  Epilog ->
    {Text[Row[{"Vertical lines = sampling instant"},
      Background -> White], Scaled[{0.95, 0.95}], Right]}]
```



Modified Duobinary Eye Diagram

```
= slice = Partition[signal, Round[sps]];
ListLinePlot[slice,
  PlotRange -> All,
  PlotLabel -> "Eye Diagram",
  AxesLabel -> {"symbol sample", "amplitude"},
  GridLines -> {{sps / 2 + 1}, {-2, -1, 1, 2}}]
```



Raised Cosine Channel

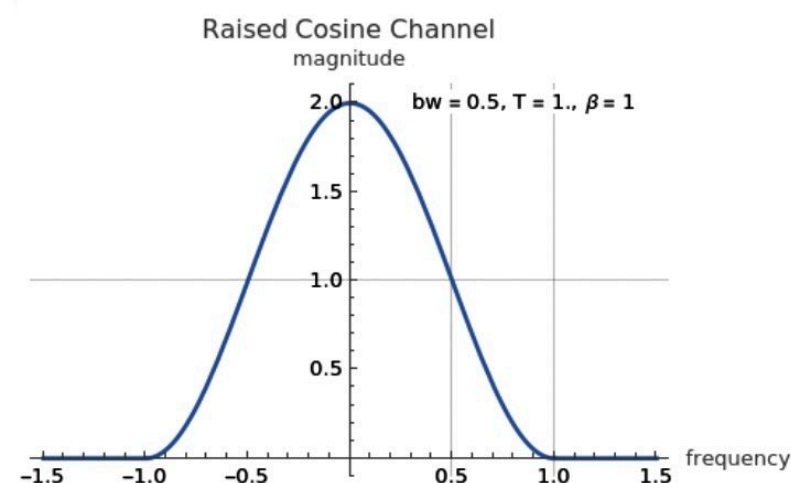
This shape is also known as a [Hanning](#) window, and is a better approximation of an analog filter since the frequency response curve gradually tapers off. The roll off can vary between 0 and 1. A roll off of 0 corresponds to a brick wall filter

Frequency Response

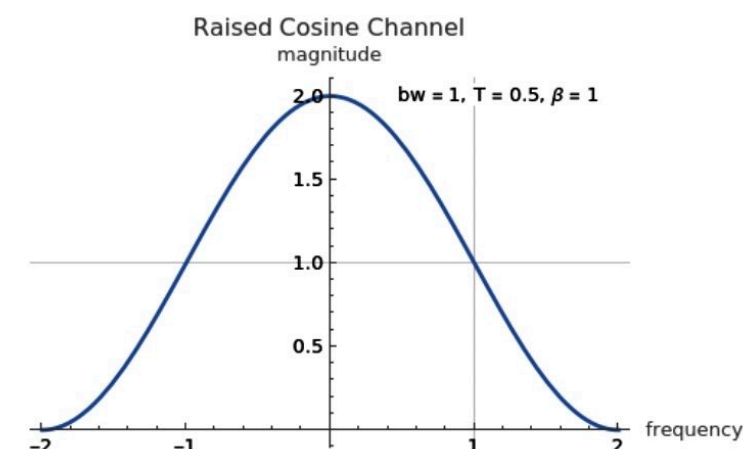
```
bw = 0.5; (*bandwidth or vestigial symmetry*)
T = 1. / (2 bw); (*symbol period*)
β = 1; (*roll off*)
rcosch = 1 + Cos[π f T]; (*rcos channel*)
rcosf = Piecewise[{{0, f < -2 bw}, {rcosch, -2 bw ≤ f ≤ 2 bw}, {0, f > 2 bw}}];
```

```
Plot[rcosf, {f, -1.5, 1.5},
  PlotLabel → "Raised Cosine Channel",
  AxesLabel → {"frequency", "magnitude"},
  GridLines → {{0.5, 1}, {1}},
  Epilog → {Text[Row[{"bw = ", bw, ", T = ", T, ", β = ", β}], Background → White],
    Scaled[{0.95, 0.95}], Right]}]
```

There are several different ways the rcos channel equation can be written



Rcos channel with a bandwidth of 0.5.



Rcos channel with a bandwidth of 1.

Manipulate Raised Cosine

```
Clear[B, T, β, rc, rcp, rcc]
```

```
Manipulate[
```

```
rc = 0.5 + 0.5 Cos[ $\frac{\pi}{\beta 2 B}$  (Abs[f] - (1 - β) B)]; (*frequency response*)
```

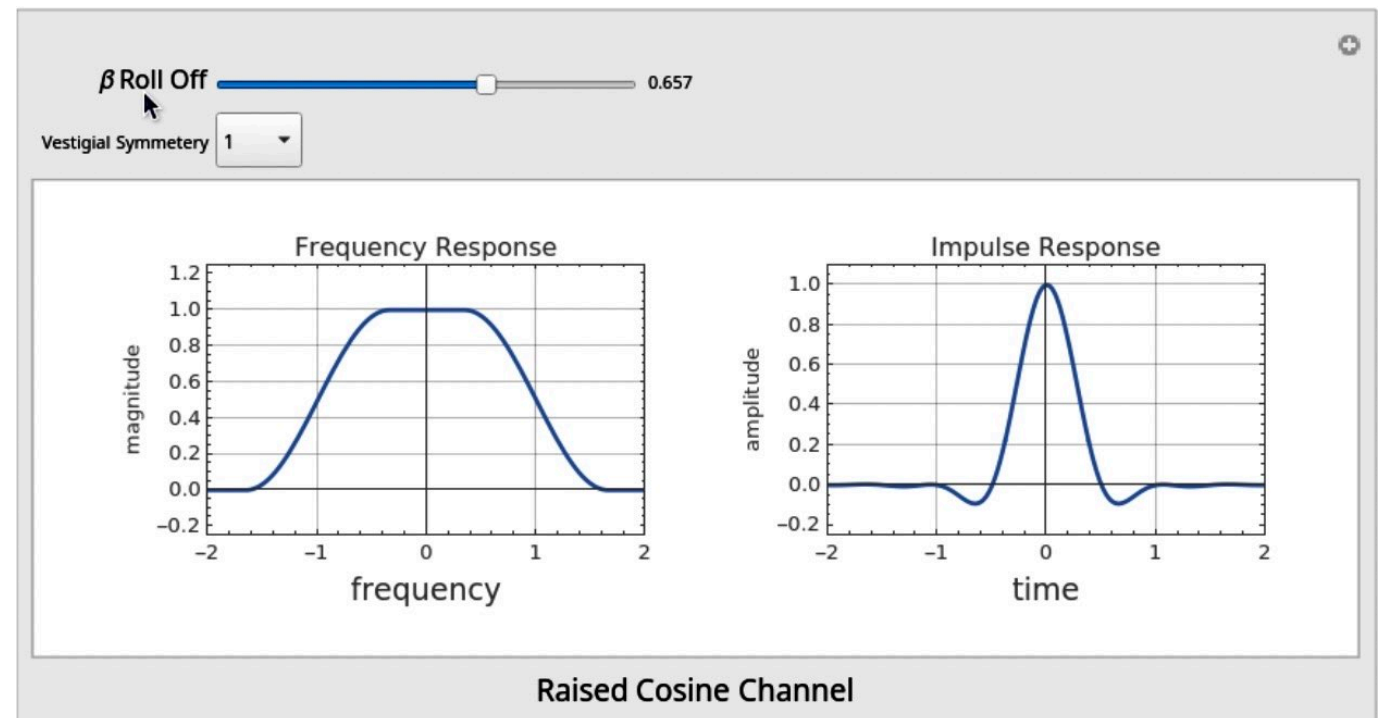
```
]
```

```
rcc = Piecewise[{{1, Abs[f] < (1 - β) B}, {rc, (1 - β) B ≤ Abs[f] ≤ (1 + β) B},  
{0, Abs[f] > (1 + β) B}}]; (*rcos channel*)
```

```
rcp =  $\frac{1}{2 B} \frac{\sin[\pi t 2 B]}{\pi t} \frac{\cos[\pi \beta t 2 B]}{1 - (4 \beta t B)^2}$ ; (*rcos pulse*)
```

```
GraphicsRow[{  
  Plot[rcc, {f, -2, 2},  
    PlotRange → {{-2, 2}, {-0.25, 1.25}},  
    GridLines → Automatic,  
    Exclusions → None,  
    PlotLabel → "Frequency Response",  
    Frame → True,  
    FrameLabel → {"frequency", "magnitude"}],  
  Plot[rcp, {t, -2, 2},  
    PlotRange → {{-2, 2}, {-0.25, 1.1}},  
    GridLines → Automatic,  
    PlotLabel → "Impulse Response",  
    Frame → True,  
    FrameLabel → {"time", "amplitude"}]}}
```

```
{ {β, 0.5, "β Roll Off"}, 0.001, 1, Appearance → "Labeled", ControlType → Slider},  
{ {B, 1, "Vestigial Symmetry"}, {0.5, 1}, ControlType → PopupMenu},  
ControlPlacement → Top,  
FrameLabel → "Raised Cosine Channel"]
```



Rcos Channel

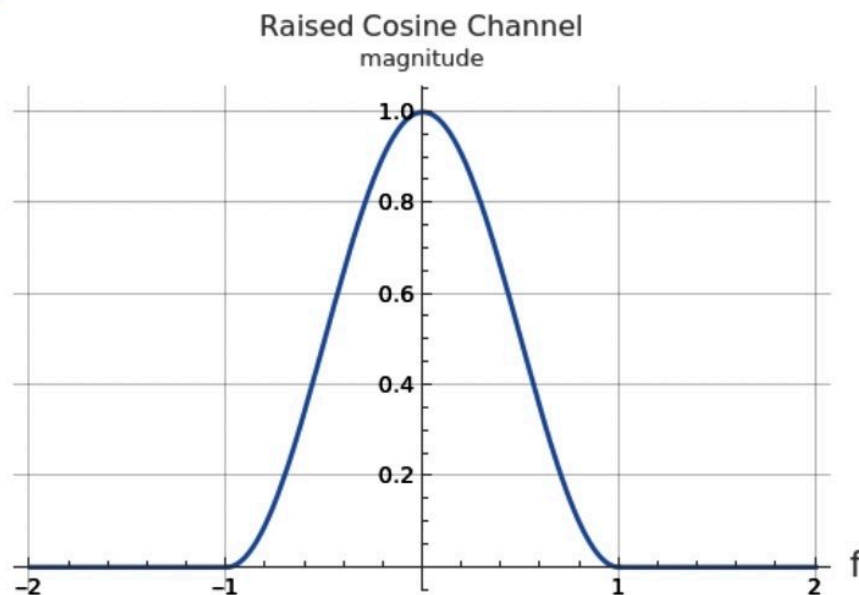
In this example, the vestigial bandwidth,
 $B = 0.5$ and the roll off, $\beta = 1$

```
B = 0.5; (*Vestigial Symmetry - bandwidth*)
T = 1/(2*B); (*symbol spacing*)

rcc = 0.5 + 0.5 Cos[ $\frac{\pi f}{2 B}$ ];
rcos = Piecewise[{{0, f < -2 B}, {rcc, -2 B ≤ f ≤ 2 B}, {0, f > 2 B}}];

Plot[rcos, {f, -2, 2},
  GridLines → Automatic,
  AxesLabel → {"f", "magnitude"},
  PlotLabel → "Raised Cosine Channel"]
```

7]=



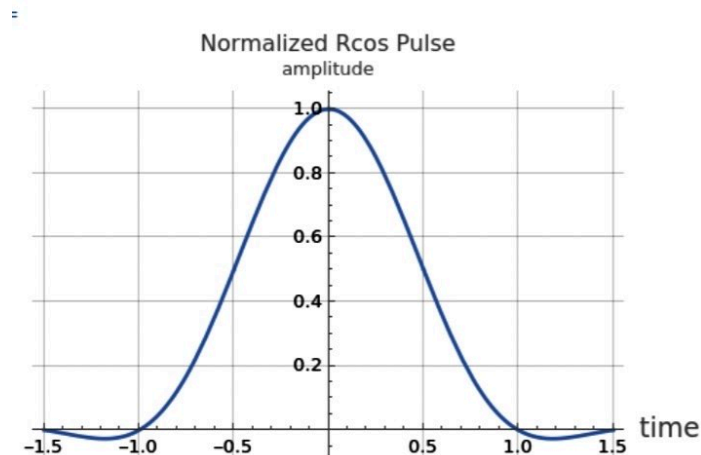
The InverseFourierTransform can be
 used to find the time domain response.

```
ircos[t_] = InverseFourierTransform[rcos, f, t, FourierParameters → {0, -2 π}] // FullSimplify
=
```

$$\frac{1}{-0.25 t + 1. t^3} \left((8.81028 \times 10^{-19} - (7.98337 \times 10^{-18} + 9.77681 \times 10^{-34} i) t) \cos[6.28319 t] + \right. \\ \left. ((-0.0397887 - 8.81028 \times 10^{-19} i) + (2.1579 \times 10^{-34} - 1.76206 \times 10^{-18} i) t) \sin[6.28319 t] \right)$$

Time Domain Plot (normalized to 1)

```
Plot[%, {t, -1.5, 1.5},
  PlotRange → All,
  GridLines → Automatic,
  AxesLabel → {"time", "amplitude"},
  PlotLabel → "Normalized Rcos Pulse"]
```



Standard Form Rcos Pulse

When the roll off $\beta = 0$, the rcos pulse reduces to 1 sinc pulse

```
Clear[B,  $\beta$ , rcosPulse]
```

```
B = 0.5;  $\beta$  = 1;
```

$$\text{rcosPulse}[t] := \frac{1}{2B} \frac{\text{Sin}[\pi t 2B]}{\pi t} \frac{\text{Cos}[\pi \beta t 2B]}{1 - (4\beta t B)^2}$$

```
Plot[rcosPulse[t], {t, -2, 2},
```

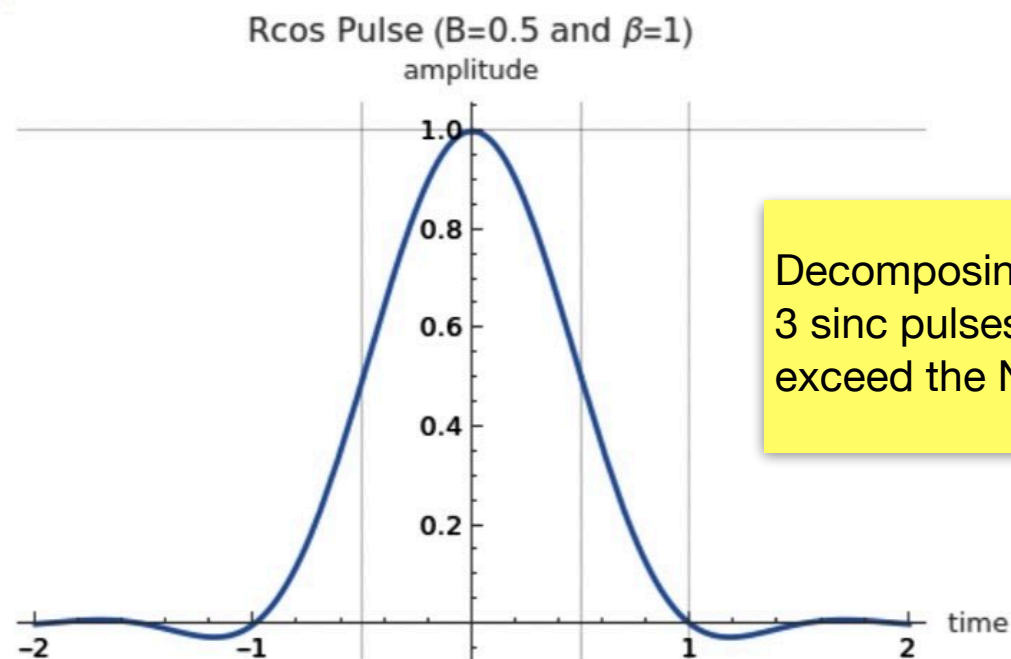
```
PlotRange → All,
```

```
GridLines → {{-0.5, 0.5, 1}, {-1, 1}},
```

```
PlotLabel → "Rcos Pulse (B=0.5 and  $\beta$ =1)",
```

```
AxesLabel → {"time", "amplitude"}]
```

```
=
```



Decomposing the rcos pulse into 3 sinc pulses allows this system to exceed the Nyquist rate.

Decomposing a Rcos Pulse

```
Clear[sinc1, sinc2, sinc3, racos, t]
```

$$\text{sinc1} = \frac{\text{Sin}[2\pi(t + 0.5)]}{2\pi(t + 0.5)};$$

$$\text{sinc2} = 2 \frac{\text{Sin}[2\pi t]}{2\pi t};$$

$$\text{sinc3} = \frac{\text{Sin}[2\pi(t - 0.5)]}{2\pi(t - 0.5)};$$

```
racos = sinc1 + sinc2 + sinc3;
```

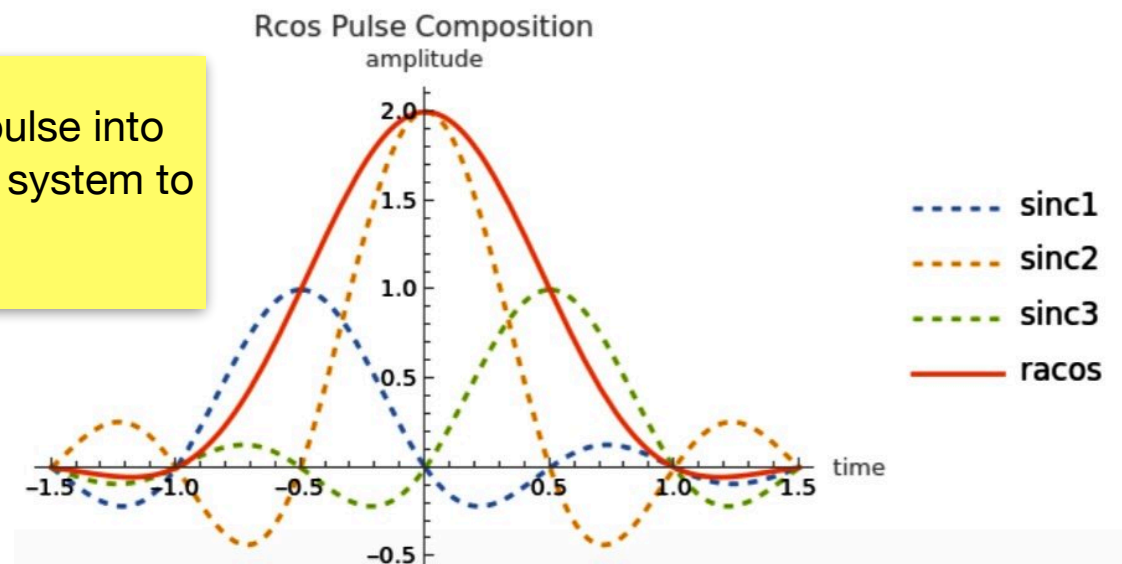
```
Plot[{sinc1, sinc2, sinc3, racos}, {t, -1.5, 1.5},
```

```
PlotStyle → {Dashed, Dashed, Dashed, Thick},
```

```
PlotLabel → "Rcos Pulse Composition",
```

```
AxesLabel → {"time", "amplitude"},
```

```
PlotLegends → "Expressions"]
```



When the roll off $\beta = 1$, the rcos pulse can be decomposed into 3 sinc pulses.

Random Rcos Data Stream

```
Clear[rdata, signal, output, sps, nos]
```

```
nos = 12; (*number of symbols*)
```

```
sps = 16; (*samples per symbol*)
```

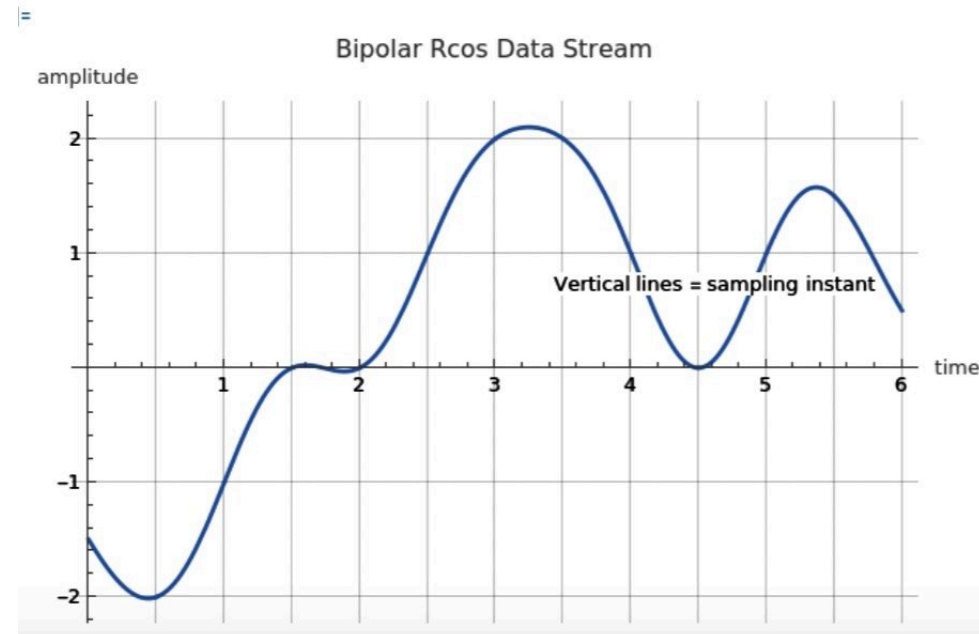
```
ss = 0.5; (*symbol spacing*)
```

```
rdata = RandomChoice[{-1, 1}, nos];
```

```
(* t = 0 seems to create a problem. Adding 10-5 *)
```

```
output[t_] =  $\sum_{k=1}^{nos} rdata[k] \times rcosPulse[t - (k - 1) ss + 10^{-5}]$ ;
```

```
Plot[output[t], {t, 0, nos ss},
  PlotRange → All,
  GridLines → {Table[k, {k, ss, nos, ss}], {-2, -1, 1, 2}},
  PlotLabel → "Bipolar Rcos Data Stream",
  AxesLabel → {"time", "amplitude"},
  Epilog → {Text[Row[{"Vertical lines = sampling instant"}, Background → White],
    Scaled[{0.95, 0.65}], Right]}]
```

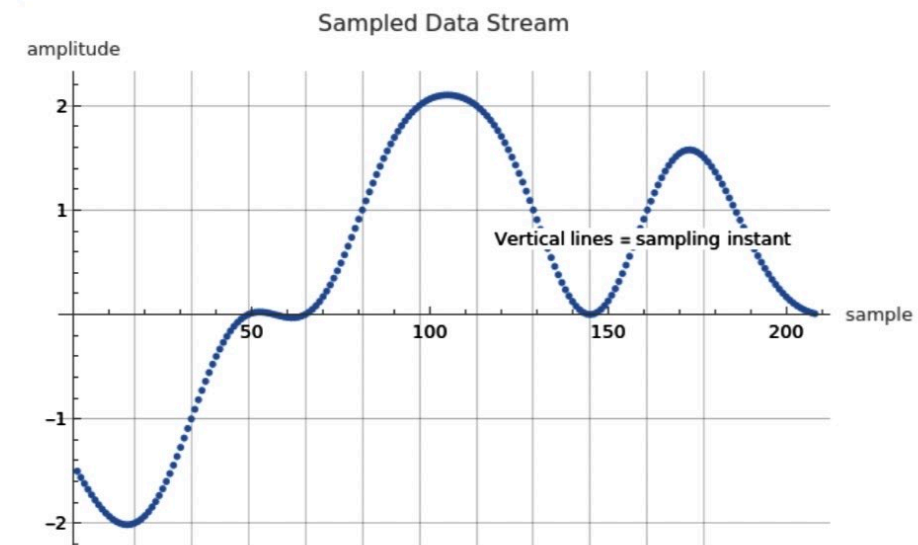


Using bipolar pulses essentially eliminates a dc component.

Sampling the data stream simplifies the construction of an eye diagram

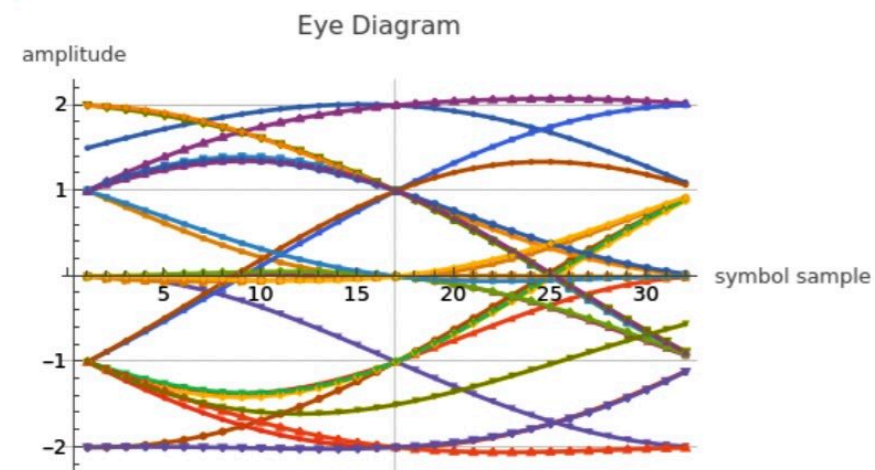
```
signal = Table[output[t], {t, 0.000001, (nos + 1) ss,  $\frac{ss}{sps}$ }]
```

```
ListPlot[signal,
  PlotRange → All,
  PlotLabel → "Sampled Data Stream",
  AxesLabel → {"sample", "amplitude"},
  GridLines → {Table[n, {n, (sps + 1), nos sps, sps}], {-2, -1, 1, 2}},
  Epilog → {Text[Row[{"Vertical lines = sampling instant"}, Background → White],
    Scaled[{0.95, 0.65}], Right]}]
```



Rcos Eye Diagram

```
slice = Partition[signal, Round[2 sps]];  
ListLinePlot[slice,  
  PlotRange → All,  
  PlotLabel → "Eye Diagram",  
  AxesLabel → {"symbol sample", "amplitude"},  
  GridLines → {{sps + 1}, {-2, -1, 1, 2}},  
  PlotMarkers → {Automatic, 4}]
```

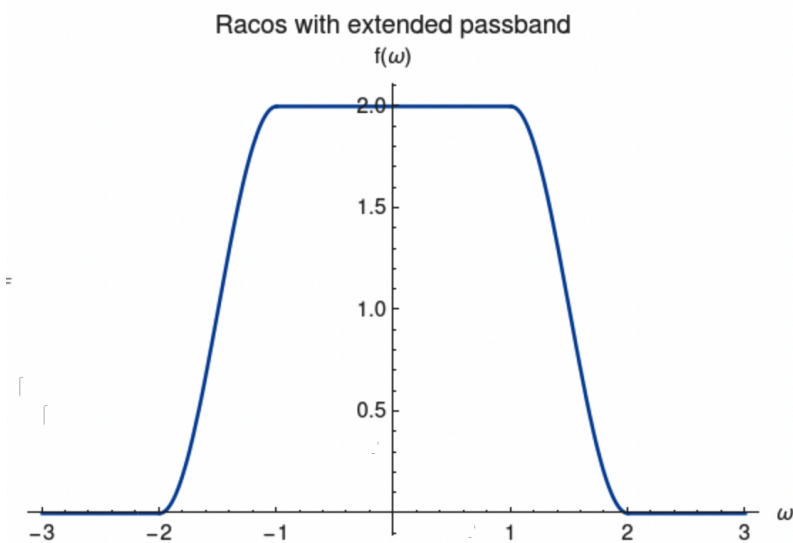


Wideband Racos Channel

The raised cosine shape can be used to define the roll-off of a wider communication channel or filter.

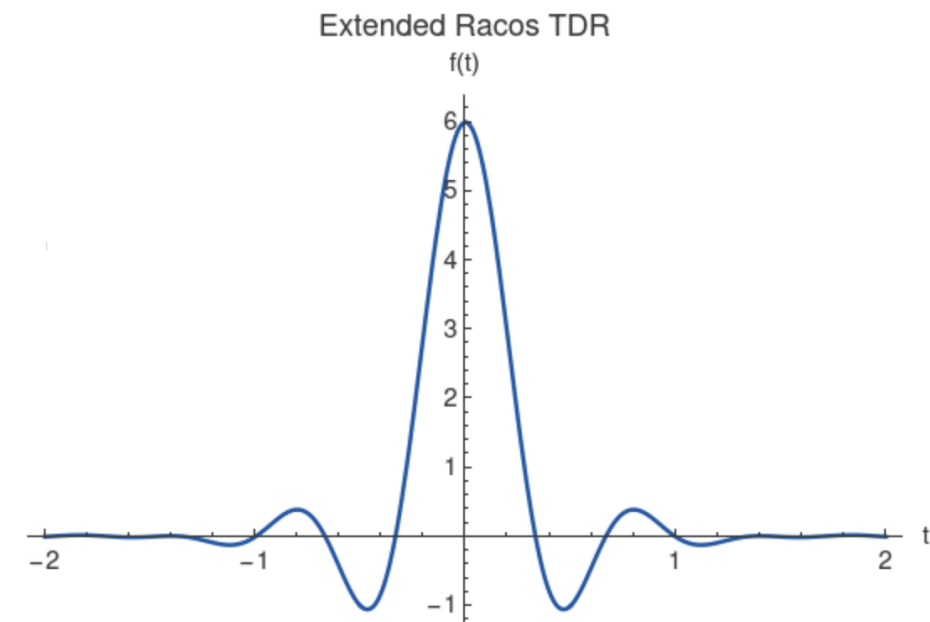
```
= ω1 = 1;
ω2 = 2;
extendedracoschannel =
  Piecewise[{{0, ω < -ω2}, {1 - Cos[π ω], ω < -ω1}, {2, ω < ω1}, {1 - Cos[π ω], ω < ω2},
    {0, ω > ω2}}]
= 
$$\begin{cases} 0 & \omega < -2 \\ 1 - \cos[\pi \omega] & \omega < -1 \\ 2 & \omega < 1 \\ 1 - \cos[\pi \omega] & \omega < 2 \\ 0 & \text{True} \end{cases}$$

= Plot[extendedracoschannel, {ω, -3, 3},
  PlotLabel → "Racos with extended passband",
  AxesLabel → {"ω", "f(ω)"}]
```



```
iftextracos =
  InverseFourierTransform[extendedracoschannel, ω, t,
    FourierParameters → {0, -2 π}] // Simplify
= 
$$\frac{\sin[2 \pi t] + \sin[4 \pi t]}{\pi t (-1 + 4 t^2)}$$

Plot[iftextracos, {t, -2, 2},
  PlotLabel → "Extended Racos TDR",
  AxesLabel → {"t", "f(t)"},
  PlotRange → All]
```



Root Raised Cosine Channel

This channel/filter is often used in [matched filter](#) applications, one being in the transmitter and the other in the receiver. This helps to optimize the [S/N ratio](#).

The overall response is that of a raised cosine channel.

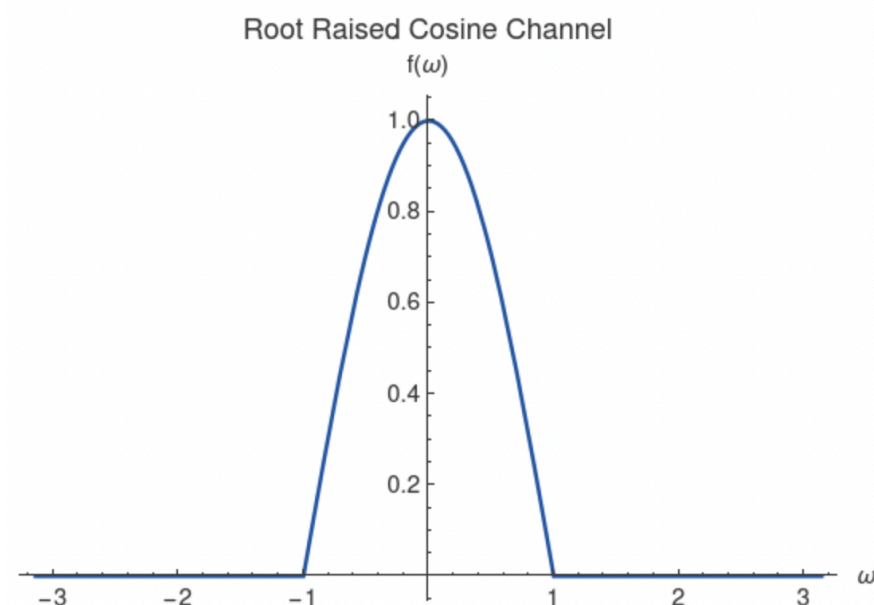
`raisedcosinechannel =`

`Piecewise[{{0,  $\omega < -1$ }, { $\frac{\cos[\pi \omega] + 1}{2}$ ,  $\omega < 1$ }, {0,  $\omega > 1$ }}];`

`rootraisedcosinechannel =  $\sqrt{\text{raisedcosinechannel}}$`

$$\sqrt{\begin{cases} 0 & \omega < -1 \\ \frac{1}{2} (1 + \cos[\pi \omega]) & \omega < 1 \\ 0 & \text{True} \end{cases}}$$

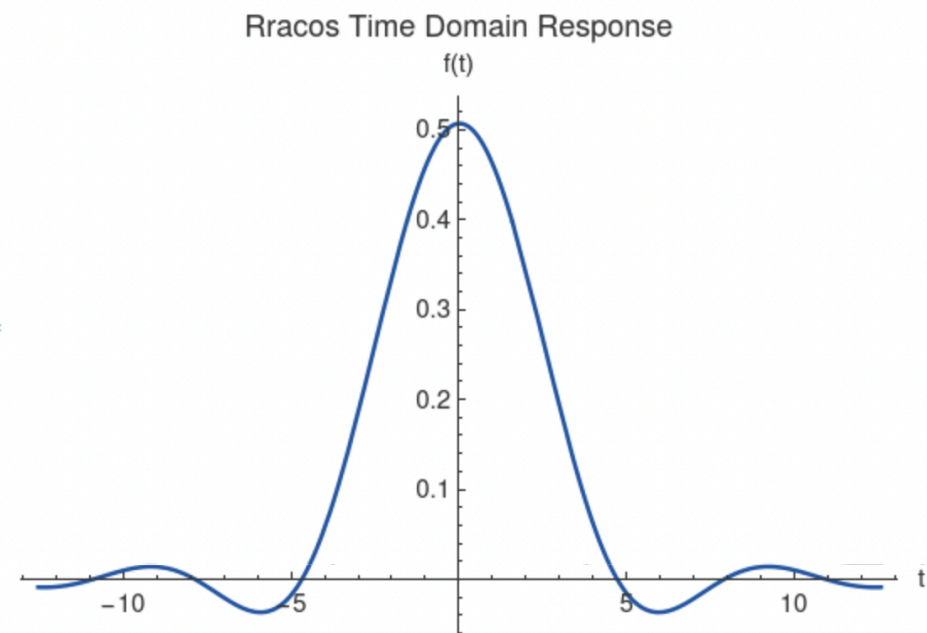
`Plot[rootraisedcosinechannel, { ω , $-\pi$, π },
PlotLabel $\rightarrow$ "Root Raised Cosine Channel",
AxesLabel $\rightarrow$ {" ω ", " $f(\omega)$ "}]`



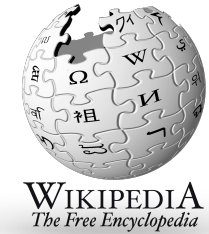
`iftrracos = InverseFourierTransform[rootraisedcosinechannel,  $\omega$ ,  $t$ ]`

$$\frac{2 \sqrt{2 \pi} \cos[t]}{\pi^2 - 4 t^2}$$

`Plot[%, { t , -4π , 4π },
PlotLabel $\rightarrow$ "Rracos Time Domain Response",
AxesLabel $\rightarrow$ {" t ", " $f(t)$ "}]`



Analog Modulation



Amplitude Modulation

Information can be used to modulate a high frequency carrier in three principle ways: by varying the carrier amplitude, frequency or phase.

The simplest and most bandwidth efficient of these methods is amplitude modulation.

A sinewave carrier signal is of the form

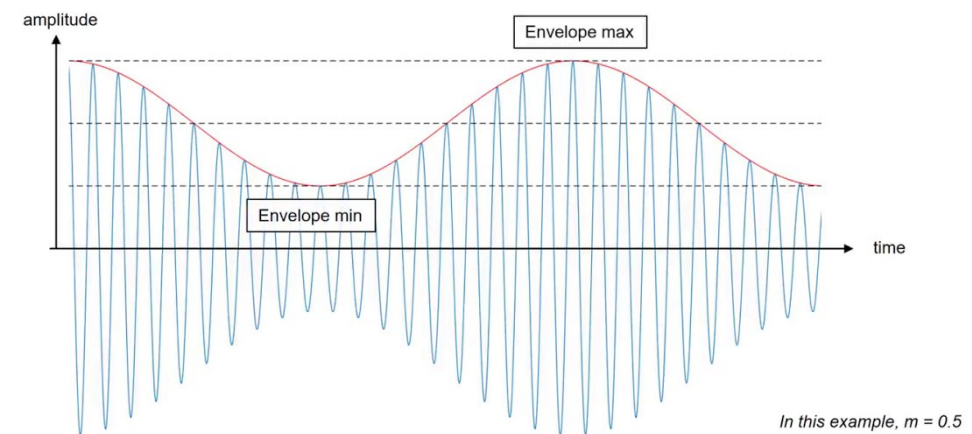
$$e_c = E_c \sin \omega_c t$$

and a sinewave modulation signal is of the form

$$e_m = E_m \sin \omega_m t$$

Notice that the amplitude of the high frequency carrier takes on the shape of the lower frequency modulation signal forming what is called a modulation envelope.

AM Modulation Depth (time domain)



The modulation index is defined as the ratio of the modulation signal amplitude to carrier signal amplitude.

$$m = \frac{E_m}{E_c} \text{ where } 0 \leq m \leq 1.$$

The overall signal can be described by:

$$\begin{aligned} e_{am} &= (E_c + E_m \sin \omega_m t) \sin \omega_c t \\ &= (E_c + mE_c \sin \omega_m t) \sin \omega_c t \end{aligned}$$

The product of two sinewaves produces sum and difference frequencies:

$$\sin \omega_1 t \sin \omega_2 t = \frac{1}{2} \cos(\omega_1 - \omega_2) t - \frac{1}{2} \cos(\omega_1 + \omega_2) t$$

The subtraction of two frequencies does not result in a negative frequency. It is understood to really represent the absolute magnitude:

$$\sin \omega_1 t \sin \omega_2 t = \frac{1}{2} \cos(|\omega_1 - \omega_2|) t - \frac{1}{2} \cos(\omega_1 + \omega_2) t$$

One way to avoid a ‘negative frequency’ is to always subtract the smaller value from the larger one. However, when this expression refers only to angles, it is often necessary to retain the negative.

As a result, expanding the instantaneous AM expression results in:

$$\begin{aligned} e_{am} &= E_c \sin \omega_c t + mE_c \sin \omega_m t \sin \omega_c t \\ &= \underbrace{E_c \sin \omega_c t}_{\text{Carrier}} + \underbrace{\frac{mE_c}{2} \sin(\omega_c - \omega_m)}_{\text{LSB}} - \underbrace{\frac{mE_c}{2} \sin(\omega_c + \omega_m)}_{\text{USB}} \end{aligned}$$

From this we observe that upper and lower sidebands are created when using amplitude modulation. The sideband

amplitude is: $\frac{mE_c}{2}$, and the total occupied spectrum is twice the bandwidth of the modulation signal or $2f_m$.

Often, the amplitude of the carrier is normalized and the expression is written:

$$e_{am} = (1 + m \sin \omega_m t) \sin \omega_c t$$

AM signals are often characterized in terms of power, since it is power, which is used to drive antennas. The total power in a $1\ \Omega$ resistor is given by:

$$\begin{aligned} P_T &= E_c^2 + \left(\frac{mE_c}{2}\right)^2 + \left(\frac{mE_c}{2}\right)^2 \\ &= P_c + \frac{m^2}{4}P_c + \frac{m^2}{4}P_c \\ &= P_c \left(1 + \frac{m^2}{2}\right) \end{aligned}$$

From this we observe that with a modulation index of 0, the transmitted power is equal to the carrier power. However, when the modulation index is 1, the total transmitted power increases to 1.5 times the carrier power.

At 100% modulation, only 1/3 of the total power is in the sidebands or only 1/2 of the carrier power is in the sidebands.

In terms of voltages and currents:

$$E_T = E_c \sqrt{1 + \frac{m^2}{2}} \quad I_T = I_c \sqrt{1 + \frac{m^2}{2}}$$

If the carrier is modulated by a complex signal, the effective modulation can be determined by the combining the modulation index of each component.

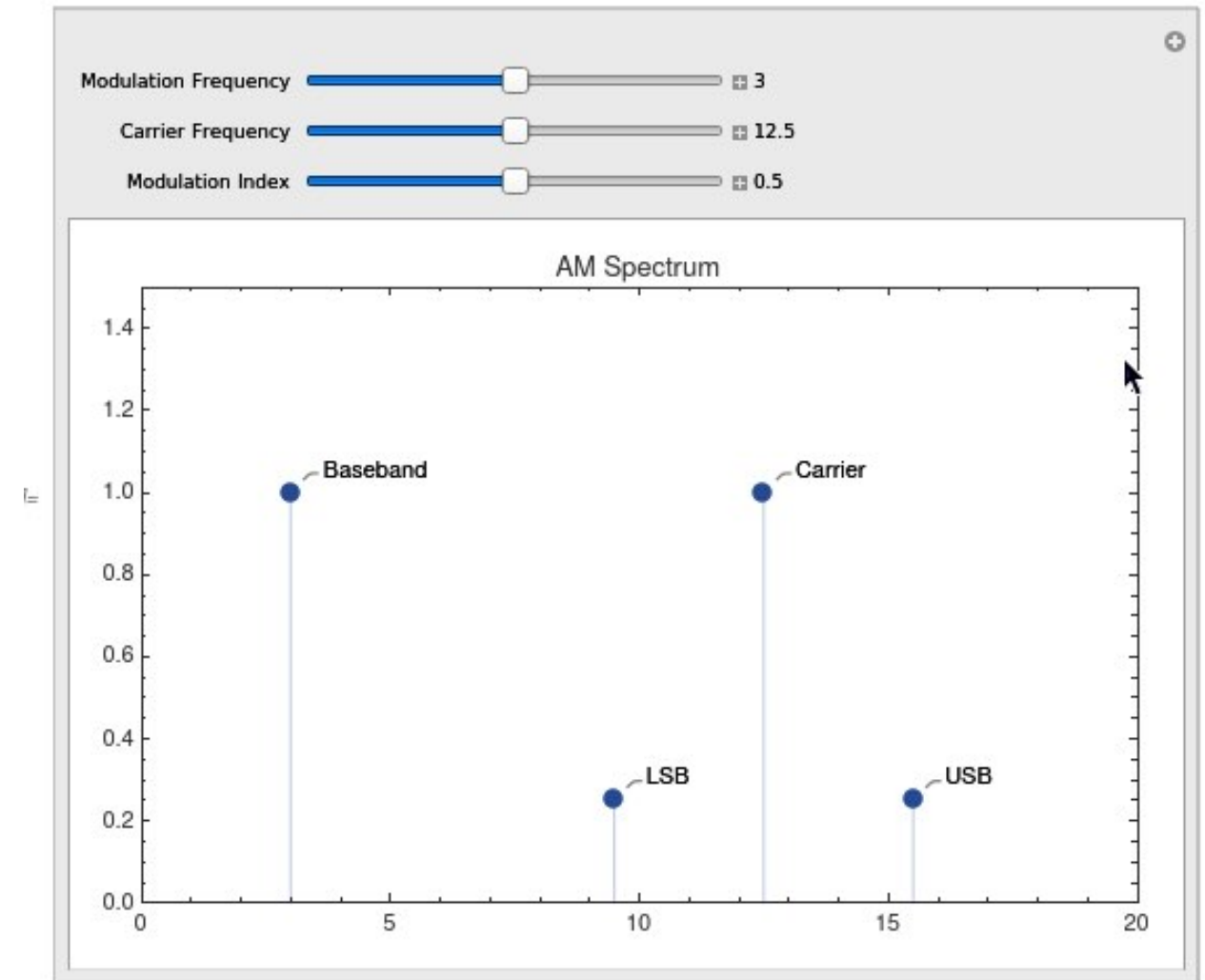
$$m_{eff} = \sqrt{m_1^2 + m_2^2 + m_3^2 + \dots} \text{ (must not exceed 1)}$$

Manipulate Baseband Signal & AM Spectrum

```
Manipulate[
  baseband = { $\omega_m$ , 1};
  LSB = { $\omega_c - \omega_m$ ,  $\frac{m}{2}$ }; (* Lower Sideband *)
  car = { $\omega_c$ , 1}; (* Carrier *)
  USB = { $\omega_c + \omega_m$ ,  $\frac{m}{2}$ }; (* UpperSideband *)
  dat = {baseband, LSB, car, USB}; (* organize data *)

  ListPlot[dat → {"Baseband", "LSB", "Carrier", "USB"},
    PlotStyle → PointSize[0.02], Filling → Axis,
    PlotLabel → "AM Spectrum", AxesLabel → {"Hz", "Magnitude"},
    Frame → True, ImageSize → {500},
    PlotRange → {{0, 20}, {0, 1.5}}],

  {{ $\omega_m$ , 3, "Modulation Frequency"}, 1, 5, Appearance → "Labeled"},
  {{ $\omega_c$ , 12.5, "Carrier Frequency"}, 10, 15, Appearance → "Labeled"},
  {{m, 0.5, "Modulation Index"}, 0, 1, Appearance → "Labeled"} ]
```

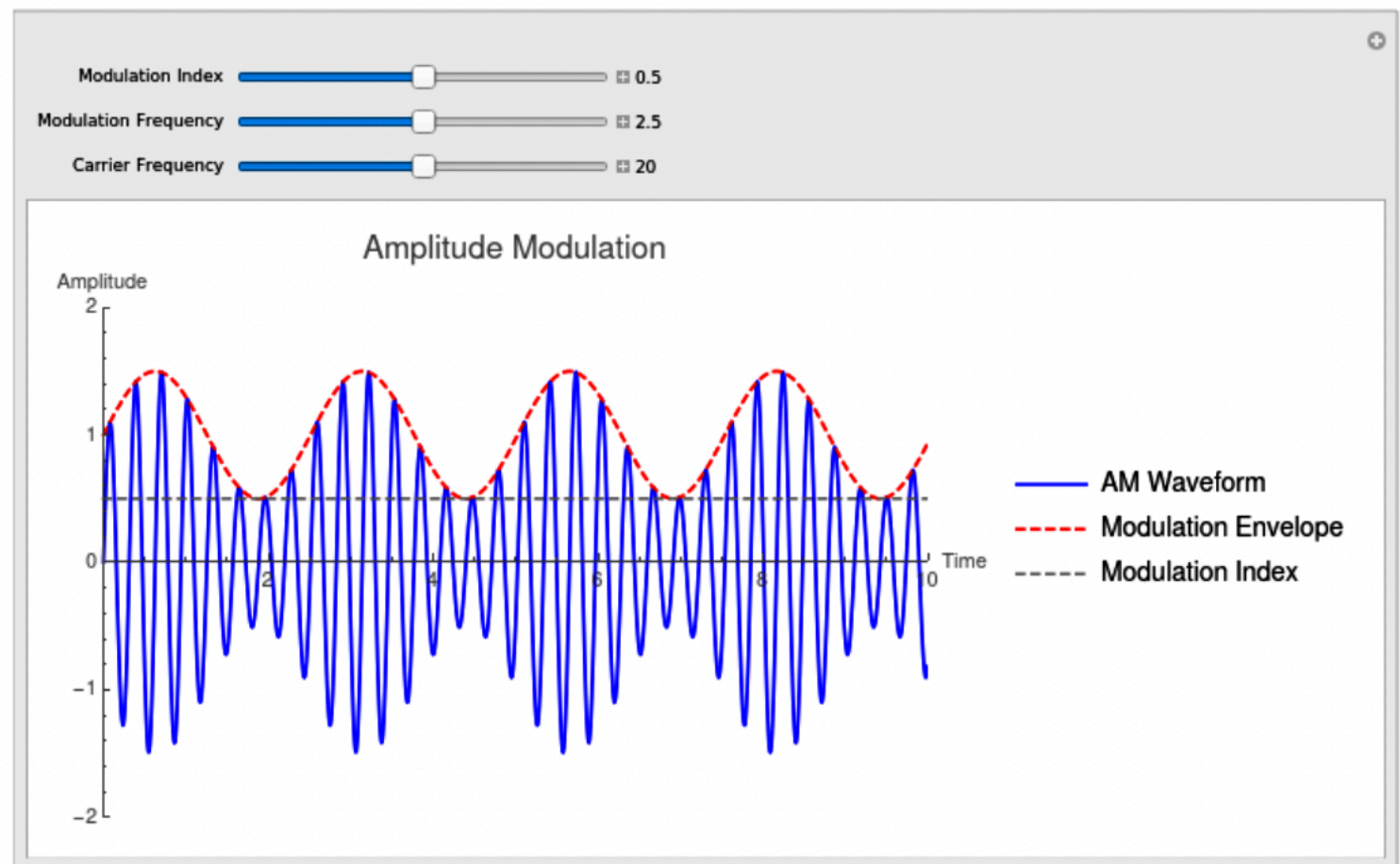


Manipulate AM in the Time Domain

```
Manipulate[
  Plot[{(1 + m * Sin[ $\omega_m$  t]) * Sin[ $\omega_c$  t], 1 + m * Sin[ $\omega_m$  t], m}, {t, 0, 10},

  PlotStyle → {Blue, {Dashed, Red}, {Dashed, Gray}},
  PlotRange → {{0, 10}, {-2, 2}},
  PlotLabel → Style["Amplitude Modulation", 16],
  AxesLabel → {"Time", "Amplitude"},
  ImageSize → {500}, PlotLegends → {"AM Waveform", "Modulation Envelope", "Modulation Index"}],

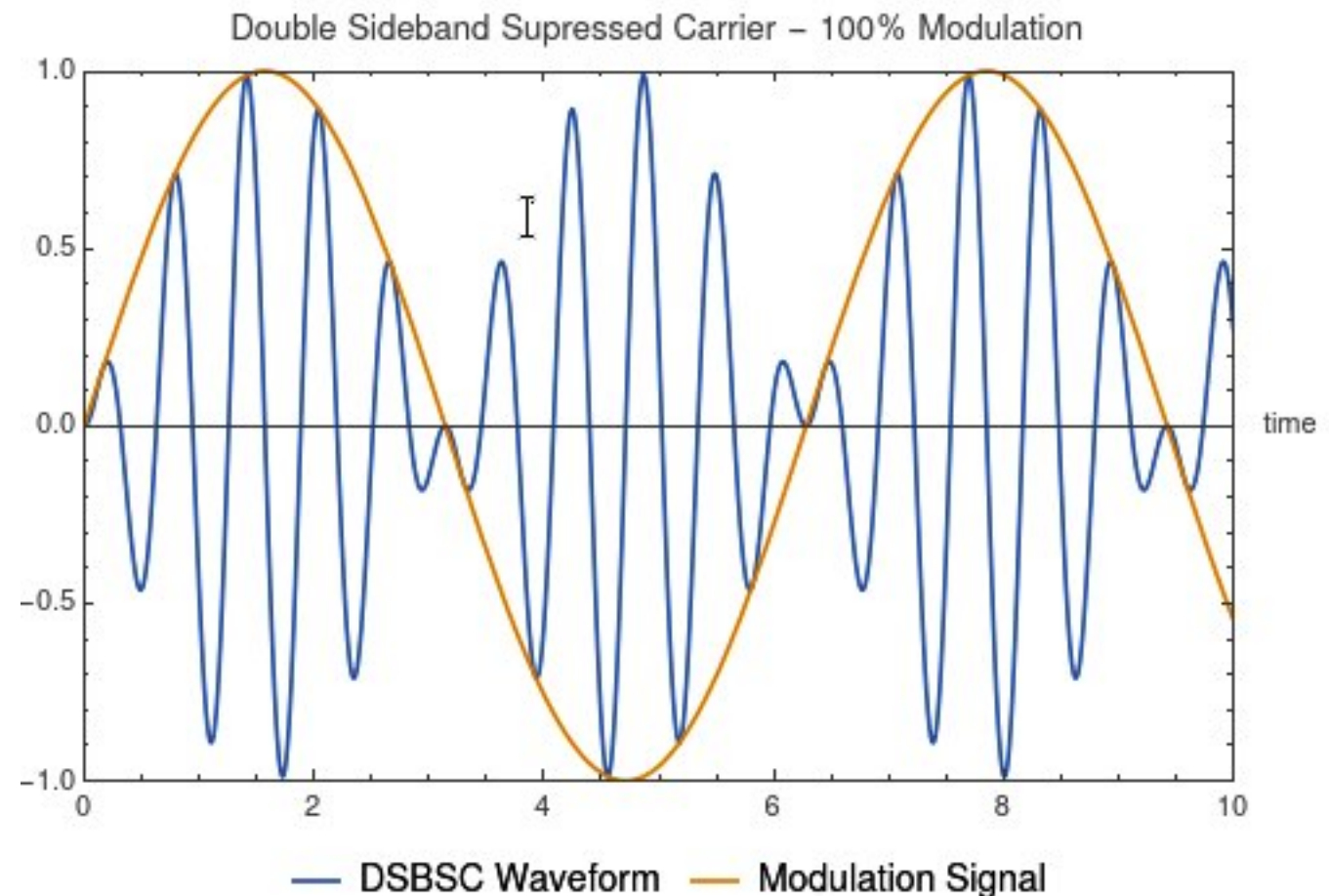
{{m, 0.5, "Modulation Index"}, 0, 1, Appearance → "Labeled"},
{{ $\omega_m$ , 2.5, "Modulation Frequency"}, 0, 5, Appearance → "Labeled"},
{{ $\omega_c$ , 20, "Carrier Frequency"}, 10, 30, Appearance → "Labeled"}]
```



DSBSC Time Domain

```
m = 1; (* Modulation Index *)  
 $\omega_m$  = 1; (* Modulation Frequency *)  
 $\omega_c$  = 10; (* Carrier Frequency *)
```

```
Plot[{(m * Sin[ $\omega_m$  t]) * Sin[ $\omega_c$  t] (* DSBSC AM Equation *), Sin[ $\omega_m$  t]} (* Modulation Signal *),  
  {t, 0, 10},  
  ImageSize → {500},  
  PlotLegends → Placed[{"DSBSC Waveform", "Modulation Signal"}, Bottom],  
  PlotRange → {{0, 10}, {-1, 1}}, (* time range, amplitude range *)  
  PlotLabel → "Double Sideband Supressed Carrier - 100% Modulation",  
  AxesLabel → {"time", "amplitude"},  
  Frame → True]
```



Manipulate DSBSC Time Domain

```
Manipulate[Plot[{(m * Sin[ $\omega_m$  t]) * Sin[ $\omega_c$  t], m * Sin[ $\omega_m$  t], m},
  {t, 0, 10}, PlotStyle → {Blue, {Dashed, Red}, {Dashed, Orange}},
```

```
PlotRange → {{0, 10}, {-1, 1}},
```

```
PlotLabel → Style["DSBSC", 16],
```

```
AxesLabel → {"Time", "Amplitude"},
```

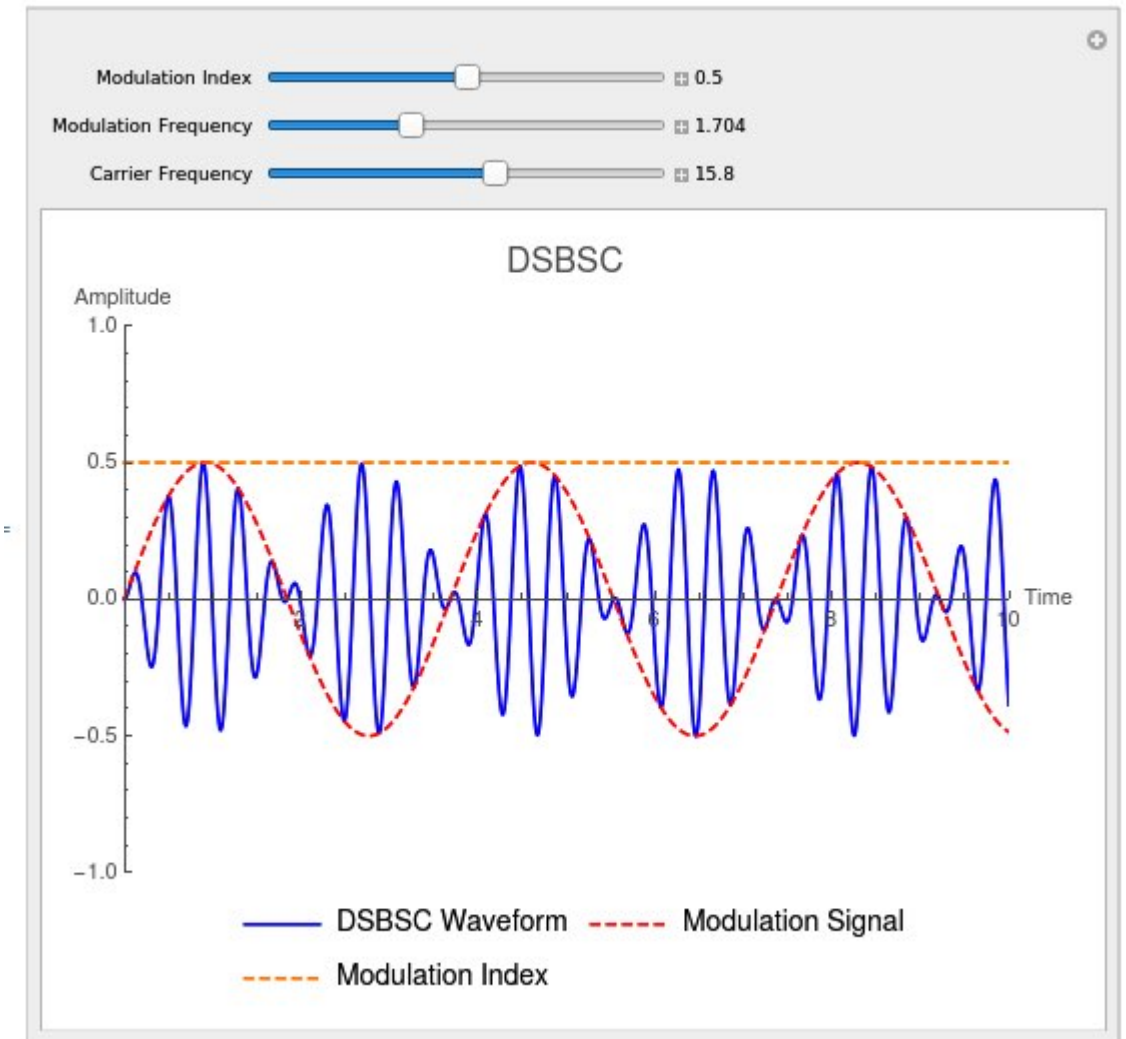
```
ImageSize → {500},
```

```
PlotLegends → Placed[{"DSBSC Waveform", "Modulation Signal", "Modulation Index"},
  Bottom]],
```

```
{{m, 0.5, "Modulation Index"}, 0, 1, Appearance → "Labeled"},
```

```
{{ $\omega_m$ , 3, "Modulation Frequency"}, 1, 3, Appearance → "Labeled"}, (* Modulation Frequency *)
```

```
{{ $\omega_c$ , 15, "Carrier Frequency"}, 10, 20, Appearance → "Labeled"} (* Carrier Frequency *)]
```



Manipulate DSBSC Frequency Domain

```
Manipulate[baseband = { $\omega_m$ , 1};

LSB = { $\omega_c - \omega_m$ ,  $\frac{m}{2}$ }; (* Lower Sideband *)

car = { $\omega_c$ , 0}; (* Carrier *)

USB = { $\omega_c + \omega_m$ ,  $\frac{m}{2}$ }; (* UpperSideband *)

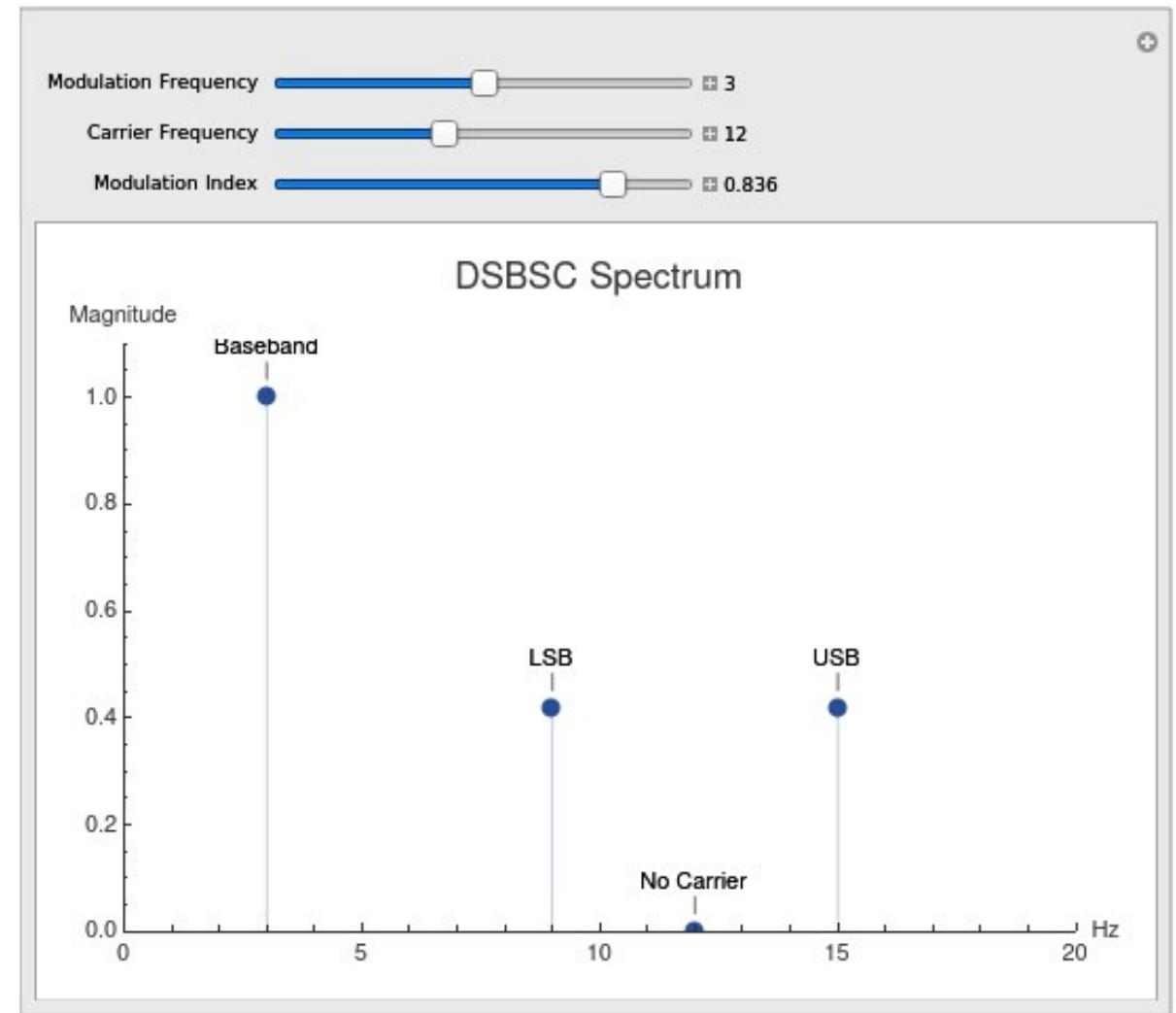
mpr = 20; (* maximum plot range *)

|

dat = {baseband, LSB, car, USB}; (* organize data *)
ListPlot[dat → {"Baseband", "LSB", "No Carrier", "USB"},

PlotStyle → PointSize[0.02],
LabelingFunction → Top,
Filling → Axis,
PlotLabel → Style["DSBSC Spectrum", 16],
AxesLabel → {"Hz", "Magnitude"},
ImageSize → {500}, PlotRange → {{0, mpr}, {0, 1.1}},

{{ $\omega_m$ , 3, "Modulation Frequency"}, 1, 5, Appearance → "Labeled"},
{{ $\omega_c$ , 12, "Carrier Frequency"}, 10, 15, Appearance → "Labeled"},
{{m, 0.5, "Modulation Index"}, 0, 1, Appearance → "Labeled"} ]
```

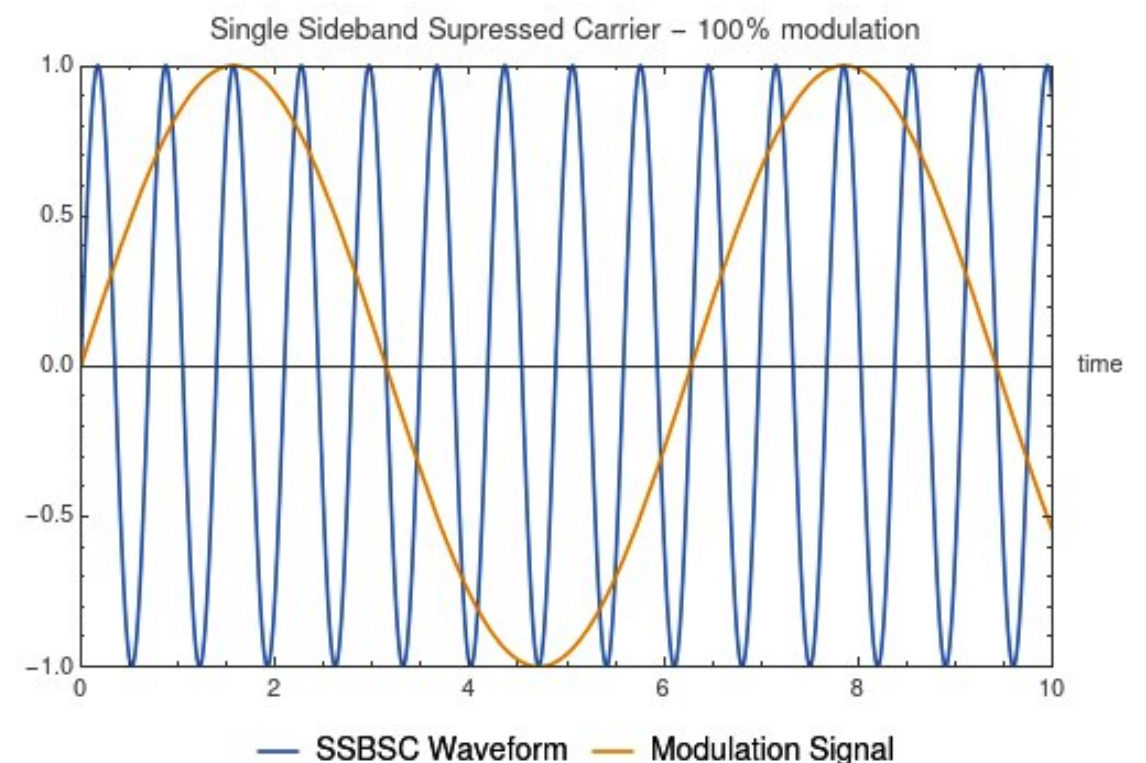


SSBSC Time Domain

Type to enter textSSBSC systems are very power efficient because they do not transmit the carrier frequency, which contains most of the transmitted power and contains no modulation information. They are also bandwidth efficient because they do not transmit a redundant sideband. These systems were widely used in [L-carrier](#) radio telephony links. It is also used in [CB \(citizen band\) radios](#).

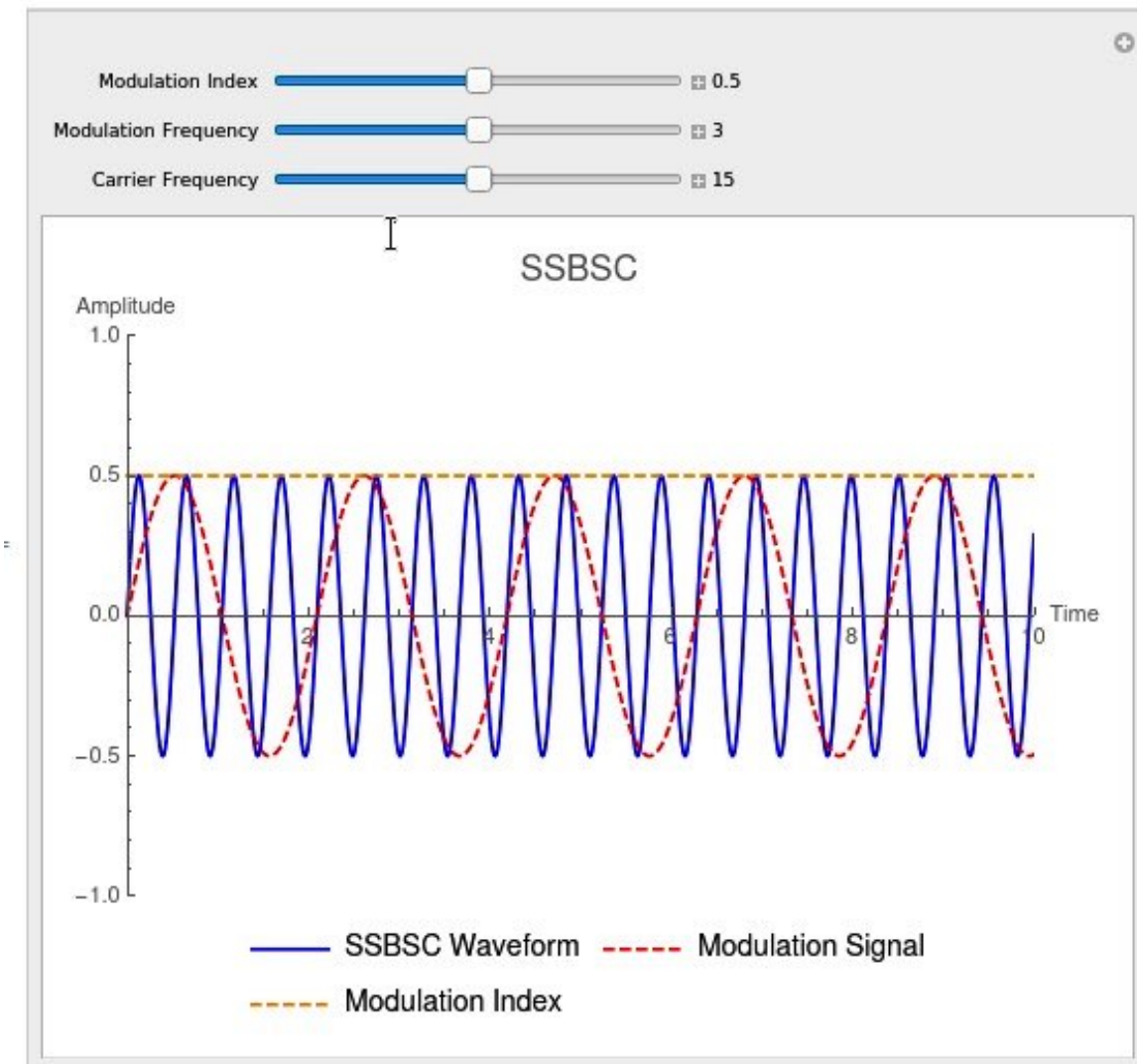
```
m = 1; (* Modulation Index (m=1 = 100% modulation)*)  
 $\omega_m$  = 1; (*Modulation Frequency *)  
 $\omega_c$  = 10; (* Carrier Frequency *)
```

```
Plot[{m * Sin[ $\omega_c$  t -  $\omega_m$  t], Sin[ $\omega_m$  t]}, (* SSBSC AM Equation (lower SB) *)  
  {t, 0, 10},  
  PlotLegends → Placed[{"SSBSC Waveform", "Modulation Signal"}, Bottom],  
  ImageSize → {500},  
  PlotRange → {{0, 10}, {-1, 1}}, (* time range, amplitude range *)  
  PlotLabel → "Double Sideband Supressed Carrier - 100% modulation",  
  AxesLabel → {"time", "amplitude"},  
  Frame → True]
```



Manipulate SSBSC Time Domain

```
Manipulate[Plot[{(m * Sin[ $\omega_c t - \omega_m t$ ]), m * Sin[ $\omega_m t$ ], m},  
  {t, 0, 10},  
  
  PlotStyle → {Blue, {Dashed, Red}, {Dashed, Orange}},  
  PlotRange → {{0, 10}, {-1, 1}},  
  PlotLabel → Style["SSBSC", 16],  
  AxesLabel → {"Time", "Amplitude"},  
  ImageSize → {500},  
  PlotLegends → Placed[{"SSBSC Waveform", "Modulation Signal", "Modulation Index"},  
    Bottom]],  
  
{{m, 0.5, "Modulation Index"}, 0, 1, Appearance → "Labeled"},  
{{ $\omega_m$ , 3, "Modulation Frequency"}, 1, 5, Appearance → "Labeled"},  
{{ $\omega_c$ , 15, "Carrier Frequency"}, 10, 20, Appearance → "Labeled"}]
```



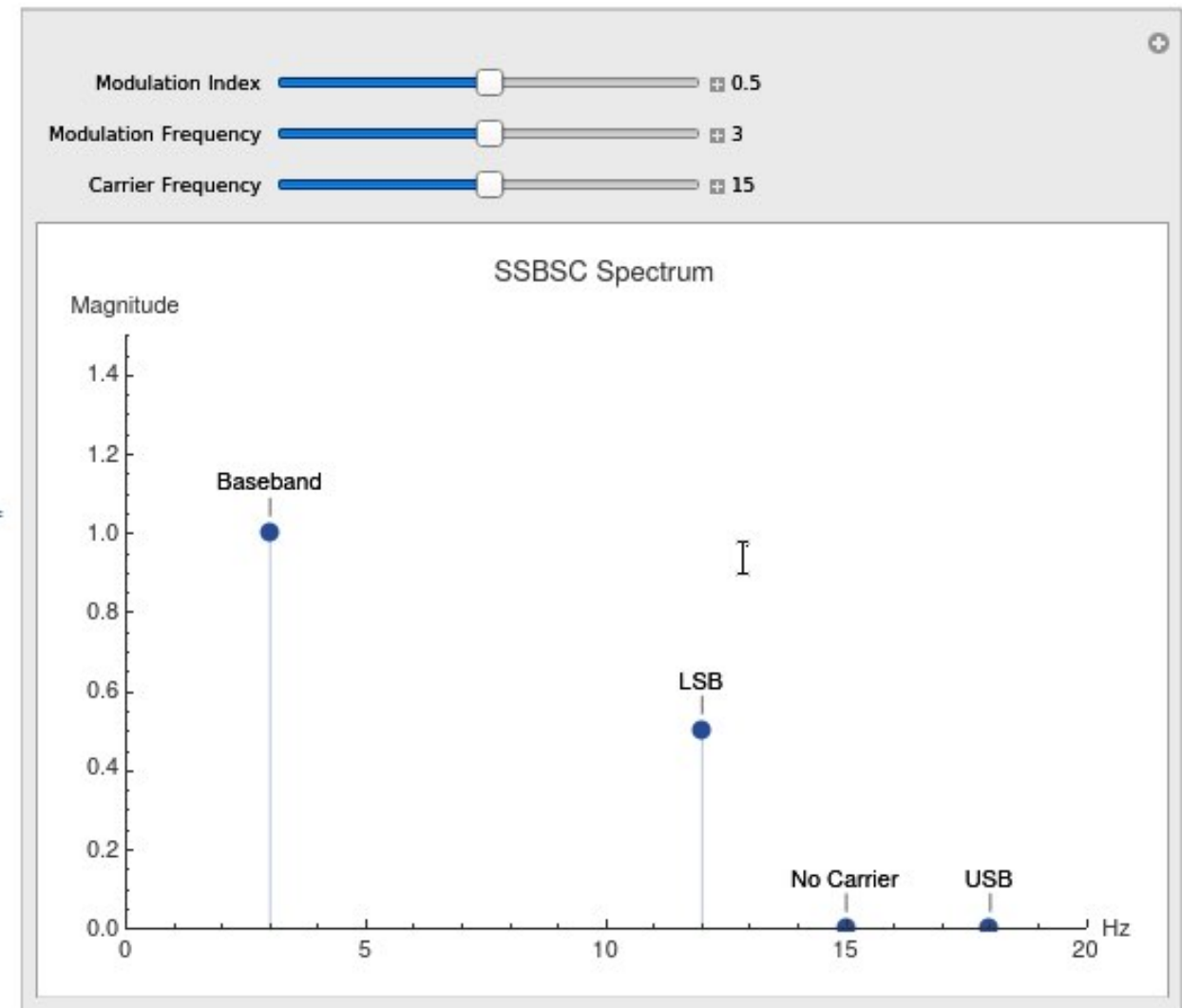
Manipulate SSBSC Frequency Domain

```
Manipulate[baseband = { $\omega m$ , 1};
LSB = { $\omega c - \omega m$ , m}; (* Lower Sideband *)
car = { $\omega c$ , 0}; (* Carrier *)
USB = { $\omega c + \omega m$ , 0}; (* UpperSideband *)

dat = {baseband, LSB, car, USB}; (* organize data *)
ListPlot[dat → {"Baseband", "LSB", "No Carrier", "USB"},
```

```
LabelingFunction → Top,
PlotStyle → PointSize[0.02],
Filling → Axis,
PlotLabel → "SSBSC Spectrum",
AxesLabel → {"Hz", "Magnitude"},
ImageSize → {500},
PlotRange → {{0, 20}, {0, 1.5}},
```

```
{{m, 0.5, "Modulation Index"}, 0, 1, Appearance → "Labeled"},
{{ $\omega m$ , 3, "Modulation Frequency"}, 1, 5, Appearance → "Labeled"},
{{ $\omega c$ , 15, "Carrier Frequency"}, 10, 20, Appearance → "Labeled"}]
```



Sampling & Tables

```
SetOptions[ListPlot, LabelStyle → {FontFamily → "Arial", FontSize → 14},  
  ImageSize → 500, GridLines → Automatic, Ticks → Automatic];  
SetOptions[ListLinePlot, LabelStyle → {FontFamily → "Arial", FontSize → 14},  
  ImageSize → 500, GridLines → Automatic, Ticks → Automatic];  
(* plot options *)
```

Note: Sampling artifacts will occur unless:

1. All frequencies are harmonically related by integers
2. The total number of samples are an integer power of 2
3. Sampling rate exceeds the Nyquist limit

AM Signal Characteristics

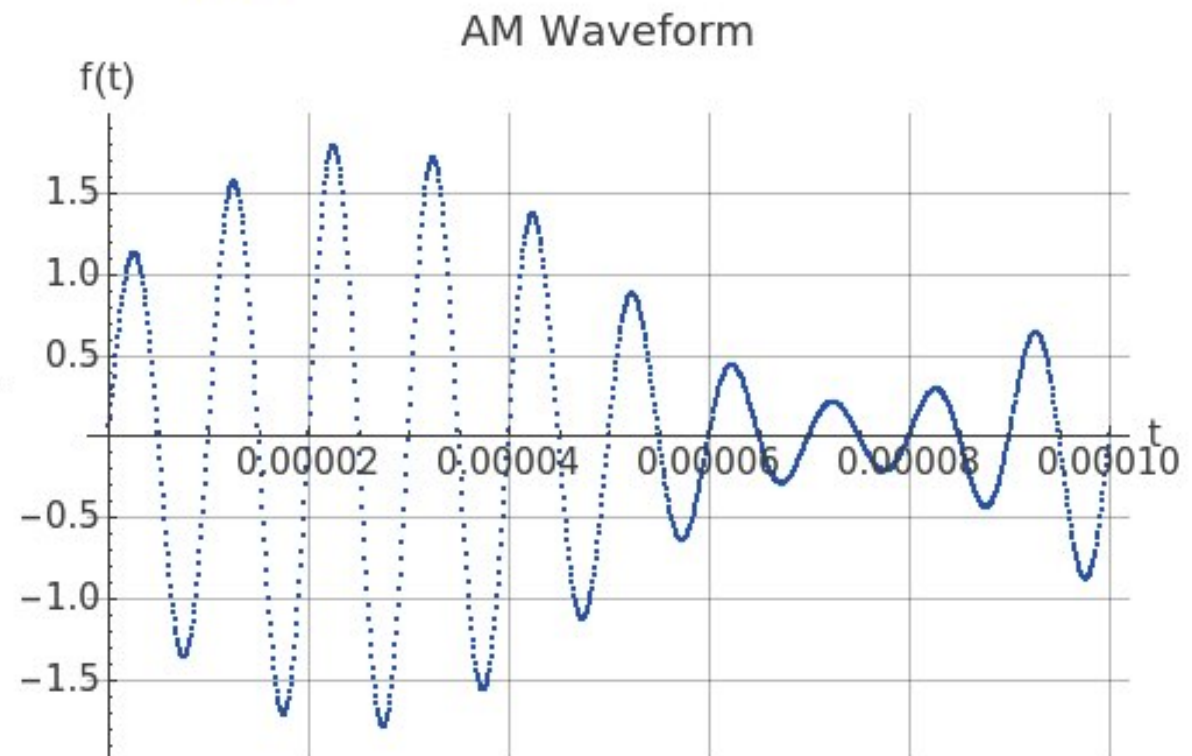
```
fm = 10 × 103; (* Modulation Frequency *)  
fc = 100 × 103; (* Carrier Frequency *)  
m = 0.8; (* Modulation Index *)  
ca = 1;  
(* Carrier Amplitude *)
```

Sampling Parameters

```
tmax = 1 / fm; (* time window = 1 cycle of the modulation *)  
tns = 1024; (* total number of samples (2n)* *)  
dt = tmax / tns; (* time increment *)
```

Sample the function (AM Signal)

```
dat = Table[(1 + m * Sin[2 π fm n dt]) * ca Sin[2 π fc n dt], {n, tns}];  
(* sampled wave form *)  
datp = Table[{k dt, dat[[k]]}, {k, 1, tns}]; (* coordinate plotting pairs *)  
ListPlot[datp, AxesLabel → {"t", "f(t)"}, PlotLabel → "AM Waveform"]
```

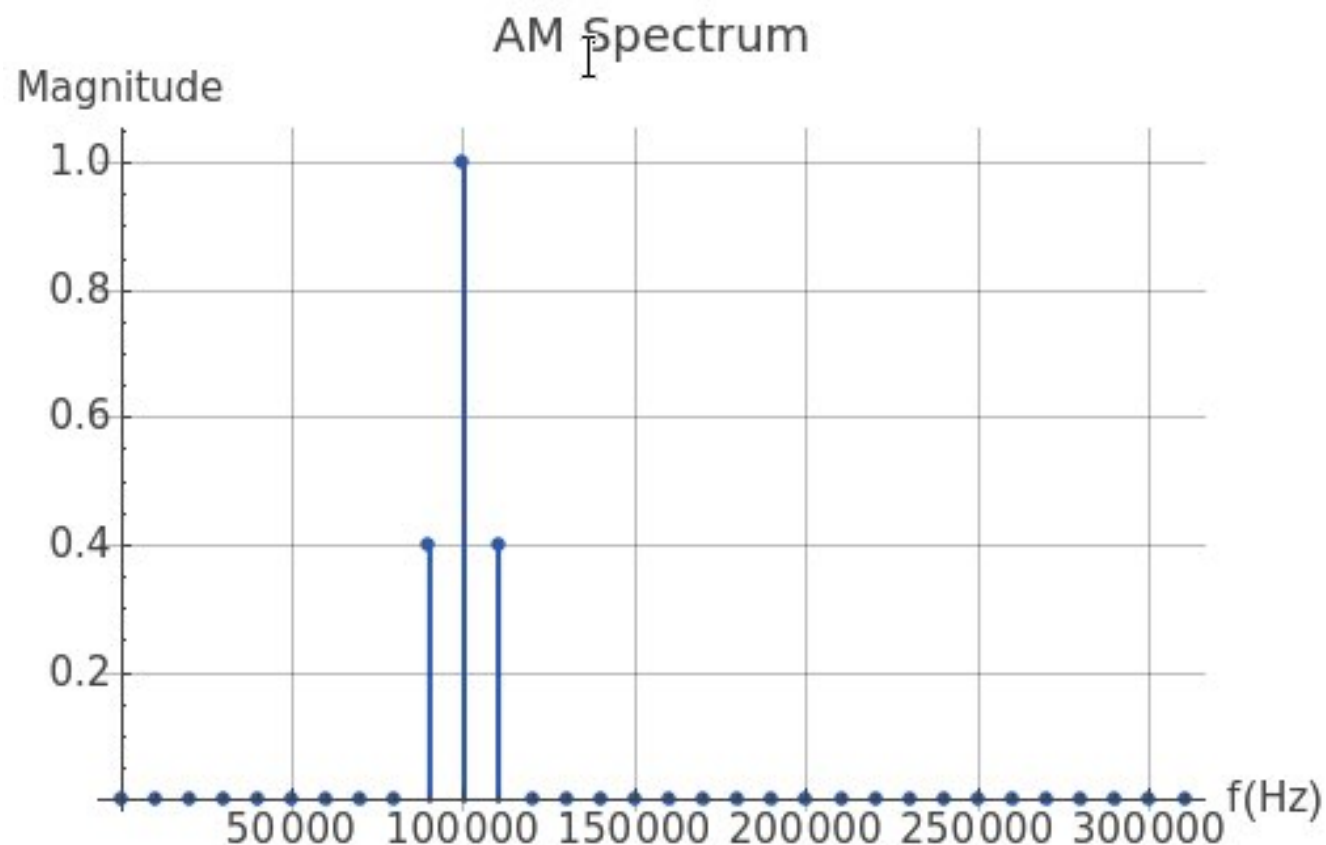


Sampling & the FFT

In this example, an AM signal above will be sampled and the FFT applied to generate the frequency spectrum.

Calculate the FFT and plot the spectrum

```
zoom = 16; (* 2,4,8,... to zoom in on the lower part of the spectrum *)
ft = Fourier[dat, FourierParameters → {1, -1}]; (* FFT *)
datf = Table[{(k - 1) / (tmax), 2 Abs[ft[[k]]] / (tns)},
  {k, 1, tns}]; (* coordinate plotting pairs *)
ListPlot[datf[[1 ;; tns / (2 zoom)]]], (* 0 Hz to some maximum *)
  PlotRange → All, Filling → Axis, FillingStyle → Thick, AxesOrigin → {0, 0},
  AxesLabel → {"f (Hz)", "Magnitude"}, PlotLabel → "AM Spectrum"]
```



Multiple AM Signals

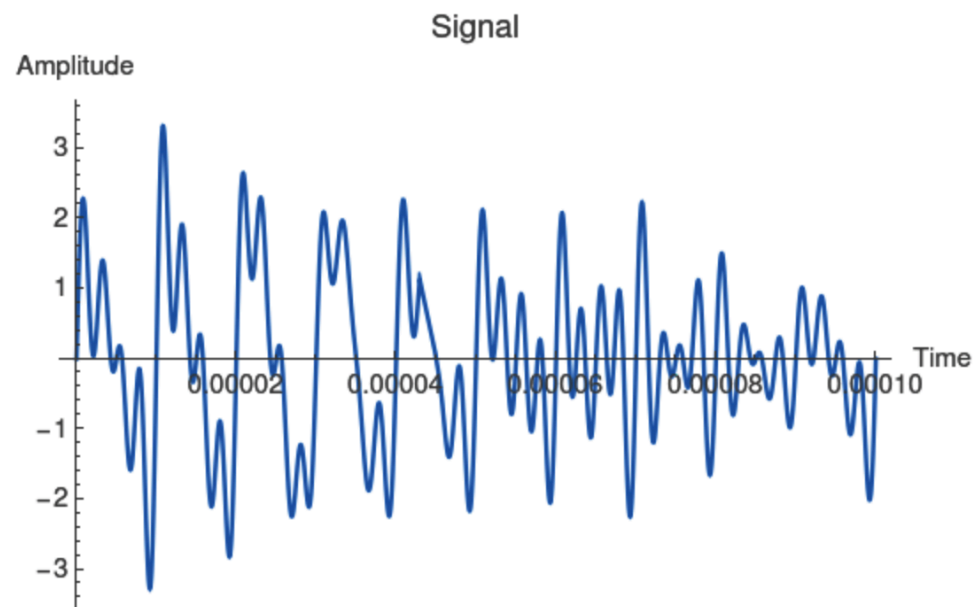
AM Signals

```
k = 103; (*kHz*)  
am1 = (1 + Sin[2  $\pi$  10 k t]) Sin[2  $\pi$  100  $\times$  103 t] ;  
am2 = 0.4 (1 + 0.8 Sin[2  $\pi$  20 k t]) Sin[2  $\pi$  200 k t] ;  
am3 = 0.6 (1 + 0.6 Sin[2  $\pi$  30 k t]) Sin[2  $\pi$  300 k t] ;  
am4 = 0.8 (1 + 0.4 Sin[2  $\pi$  20 k t]) Sin[2  $\pi$  400 k t] ;  
signal = am1 + am2 + am3 + am4;  
tw =  $\frac{1}{10 k}$ ; (*time window*)
```

The time window was selected to provide 1 cycle of the lowest modulation frequency.

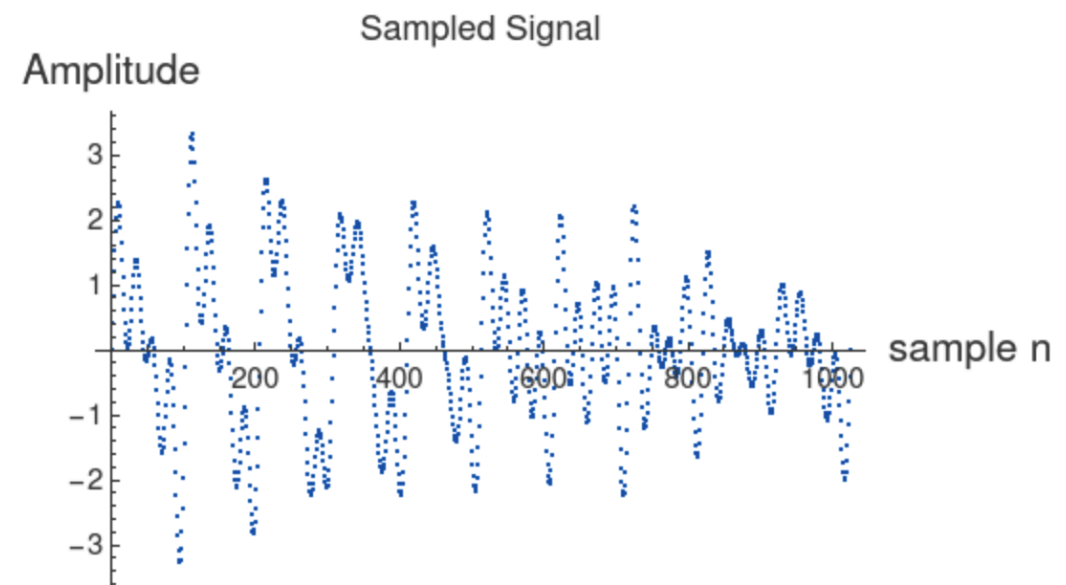
Time Domain Signal

```
tsignal = Plot[signal, {t, 0, tw},  
  PlotLabel  $\rightarrow$  "Signal",  
  AxesLabel  $\rightarrow$  {"Time", "Amplitude"}]
```



Signal - Sampled & Tabled

```
ns = 1024; (*number of samples*)  
signalTable = Table[signal, {t, 0, tw,  $\frac{tw}{ns}$ }] ;  
ListPlot[signalTable,  
  PlotLabel  $\rightarrow$  "Sampled Signal",  
  AxesLabel  $\rightarrow$  {"sample n", "Amplitude"}]
```



Increasing the number of samples increases the time & frequency resolution.

The DFT table contains complex numbers. Therefore the absolute value [Abs] is used to plot the magnitude. A second table was created just to include a column for the sample number.

DFT Table

```
3]:= ftSignalTable = Fourier[signalTable, FourierParameters -> {1, -1}];

ftSignalTableN = Table[{n - 1,  $\frac{n - 1}{tw}$ , 2  $\frac{Abs[ftSignalTable[[n]]]}{ns}$ }, {n, 1,  $\frac{ns}{2}$ }};

TableForm[ftSignalTableN,
  TableHeadings -> {"Table", {"sample", "Frequency", "Magnitude"}}];
```

Select the frequency & magnitude columns to plot the spectrum.

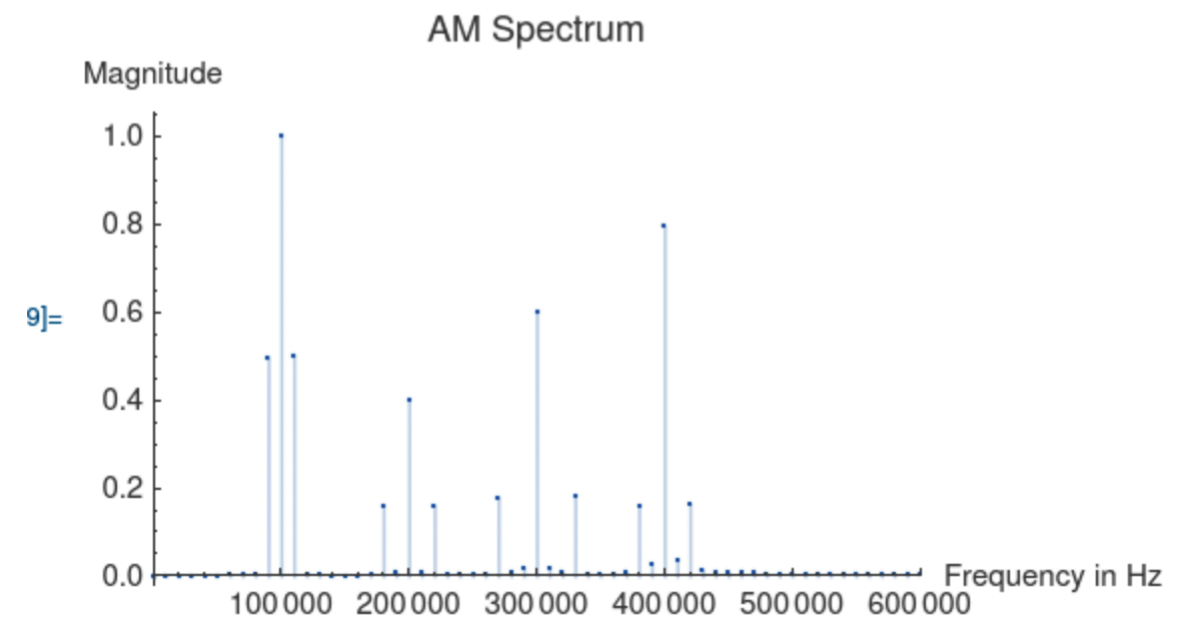
Select DFT Table Columns

```
4]:= frequency = ftSignalTableN[[All, 2]];
magnitude = ftSignalTableN[[All, 3]];
dft = Transpose[{frequency, magnitude}];

TableForm[dft,
  TableHeadings -> {"Table", {"Frequency", "Magnitude"}}];
```

AM Spectrum

```
5]:= ListPlot[dft,
  PlotRange -> {{0, 600 000}, All},
  PlotLabel -> "AM Spectrum",
  AxesLabel -> {"Frequency in Hz", "Magnitude"},
  Filling -> Axis]
```



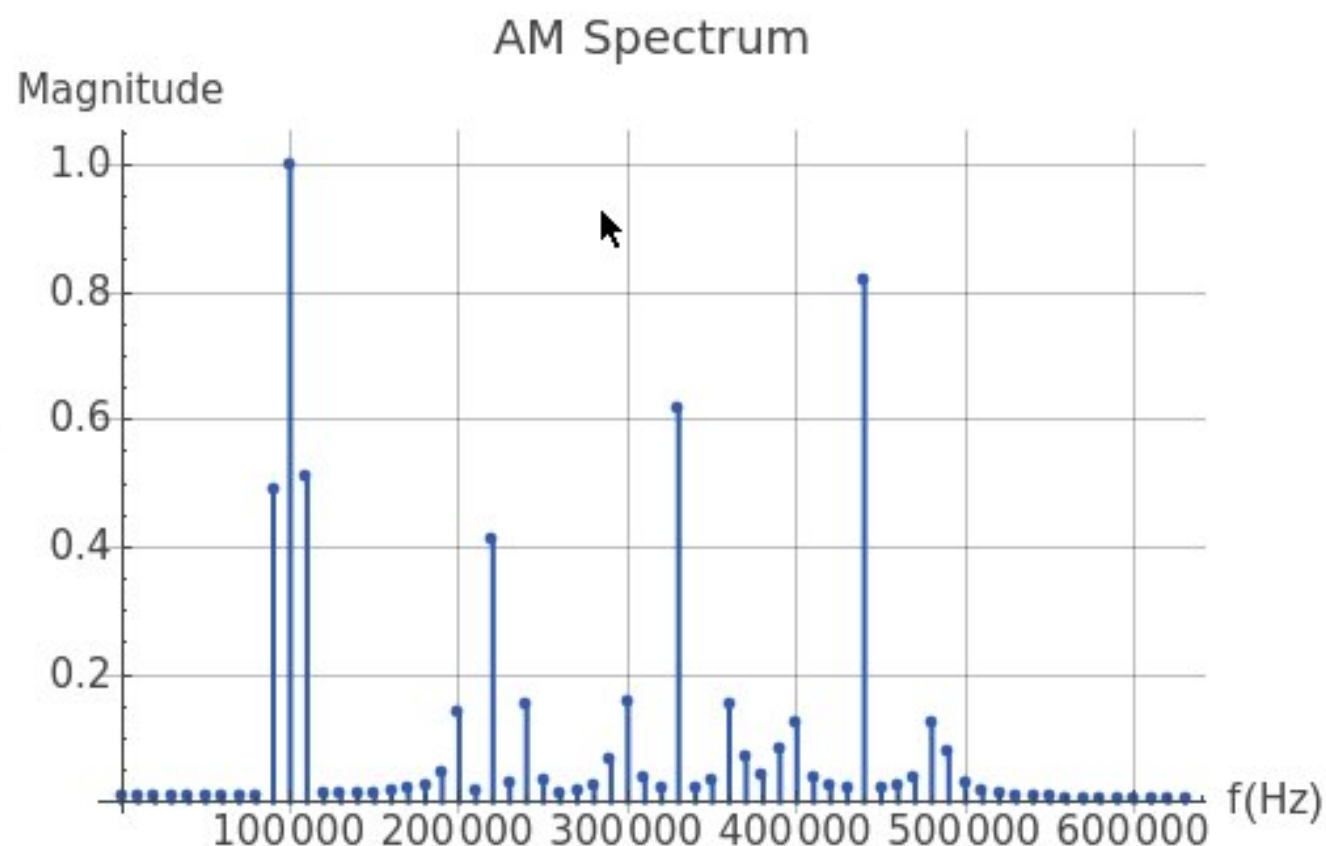
The PlotRange{min,max} values can be used to plot the portion of interest.

Sampling Artifacts

When the sampling conditions are not met, the results are misleading and require additional interpretation. In the following example, the carrier and modulation signals are not harmonically related by integers. Consequently, the spectrum contains numerous calculation artifacts.

Calculate the FFT and plot the spectrum

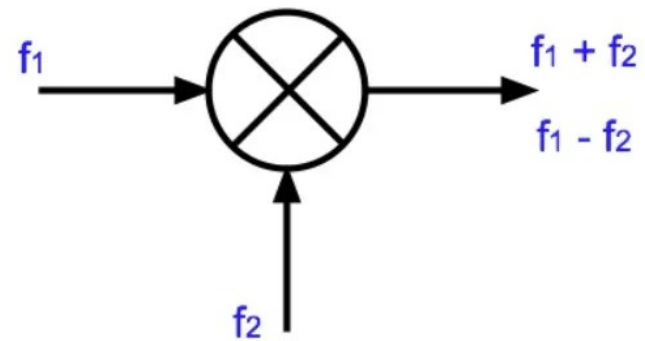
```
zoom = 8; (* 2,4,8,... to zoom in on the lower part of the spectrum *)  
ft = Fourier[dat, FourierParameters → {1, -1}]; (* FFT *)  
datf = Table[{(k - 1) / (tmax), 2 Abs[ft[[k]]] / (tns)},  
  {k, 1, tns}]; (* coordinate plotting pairs *)  
ListPlot[datf[[1 ;; tns / (2 zoom)]]], (* 0 Hz to some maximum *)  
  PlotRange → All, Filling → Axis, FillingStyle → Thick, AxesOrigin → {0, 0},  
  AxesLabel → {"f (Hz)", "Magnitude"}, PlotLabel → "AM Spectrum"]
```



The uneven sideband amplitudes is due to calculation artifacts. Changing the number of samples and sampling rate alter the results.

RF Mixing or Heterodyning

The multiplication of two sine waves creates sum and difference frequencies plus harmonics and other components. A mixer is often followed by a tuned circuit which eliminates all the undesirable components.



In most cases, the lower frequency signal is the baseband or data, while the other, much higher frequency is an intermediate or final carrier signal.


```

Manipulate[
GraphicsGrid[{

{freq1 = Sin[2  $\pi$  f1 t];
Plot[freq1, {t, 0, 1},
PlotRange  $\rightarrow$  Automatic, AxesLabel  $\rightarrow$  {"time", "amplitude"},
PlotLabel  $\rightarrow$  Style["Frequency 1", 16]]},

{freq2 = Sin[2  $\pi$  f2 t];
Plot[freq2, {t, 0, 1},
PlotRange  $\rightarrow$  Automatic, AxesLabel  $\rightarrow$  {"time", "amplitude"},
PlotLabel  $\rightarrow$  Style["Frequency 2", 16]]},

{product = Sin[2  $\pi$  f1 t] Sin[2  $\pi$  f2 t];
Plot[product, {t, 0, 1},
PlotRange  $\rightarrow$  Automatic, AxesLabel  $\rightarrow$  {"time", "amplitude"},
PlotLabel  $\rightarrow$  Style["Product", 16]]}},

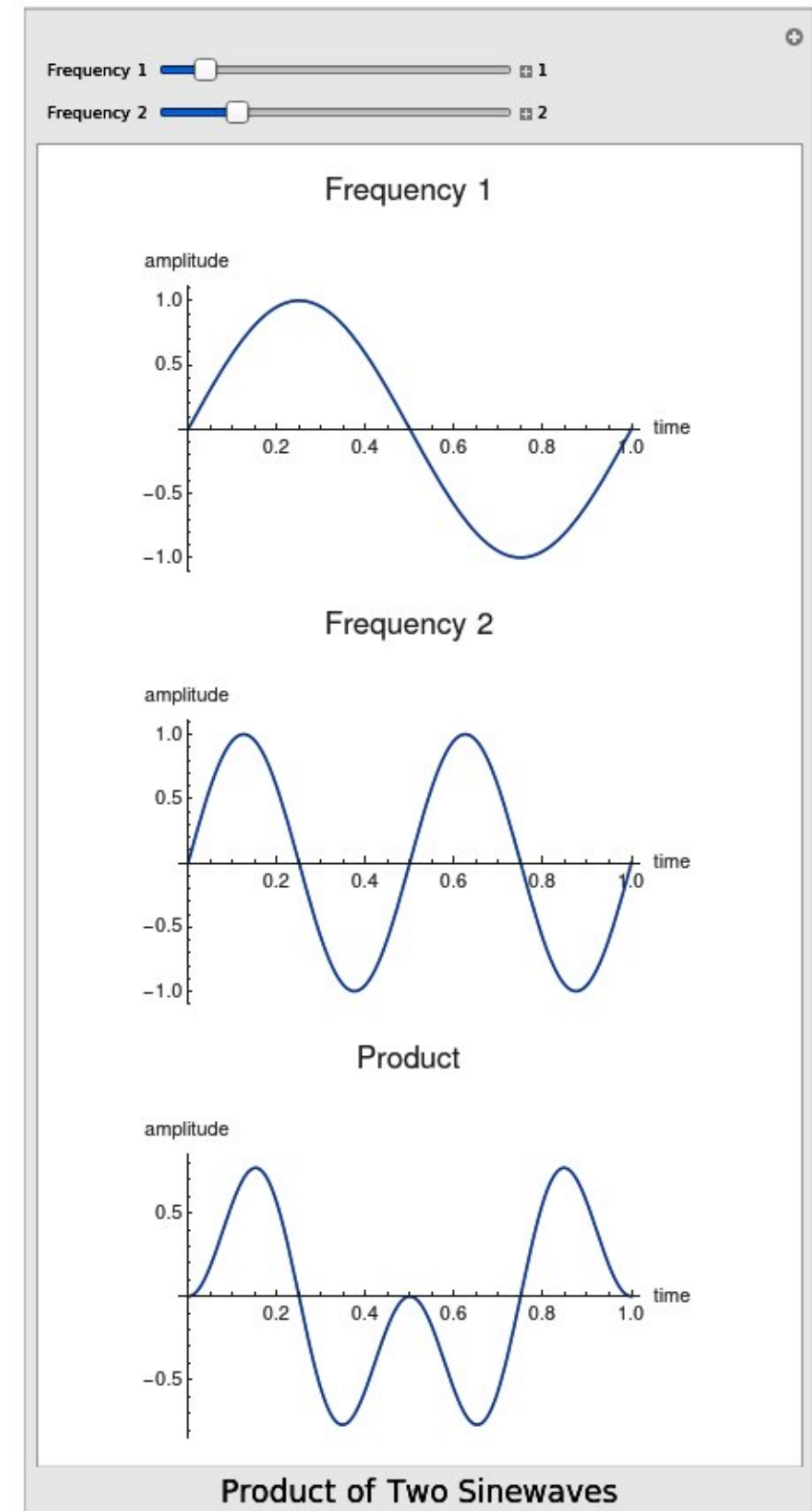
ImageSize  $\rightarrow$  500],

(* Variables *)
{{f1, 1, "Frequency 1"}, 0, 10, 1, Appearance  $\rightarrow$  "Labeled"},

{{f2, 2, "Frequency 2"}, 0, 10, 1, Appearance  $\rightarrow$  "Labeled"},

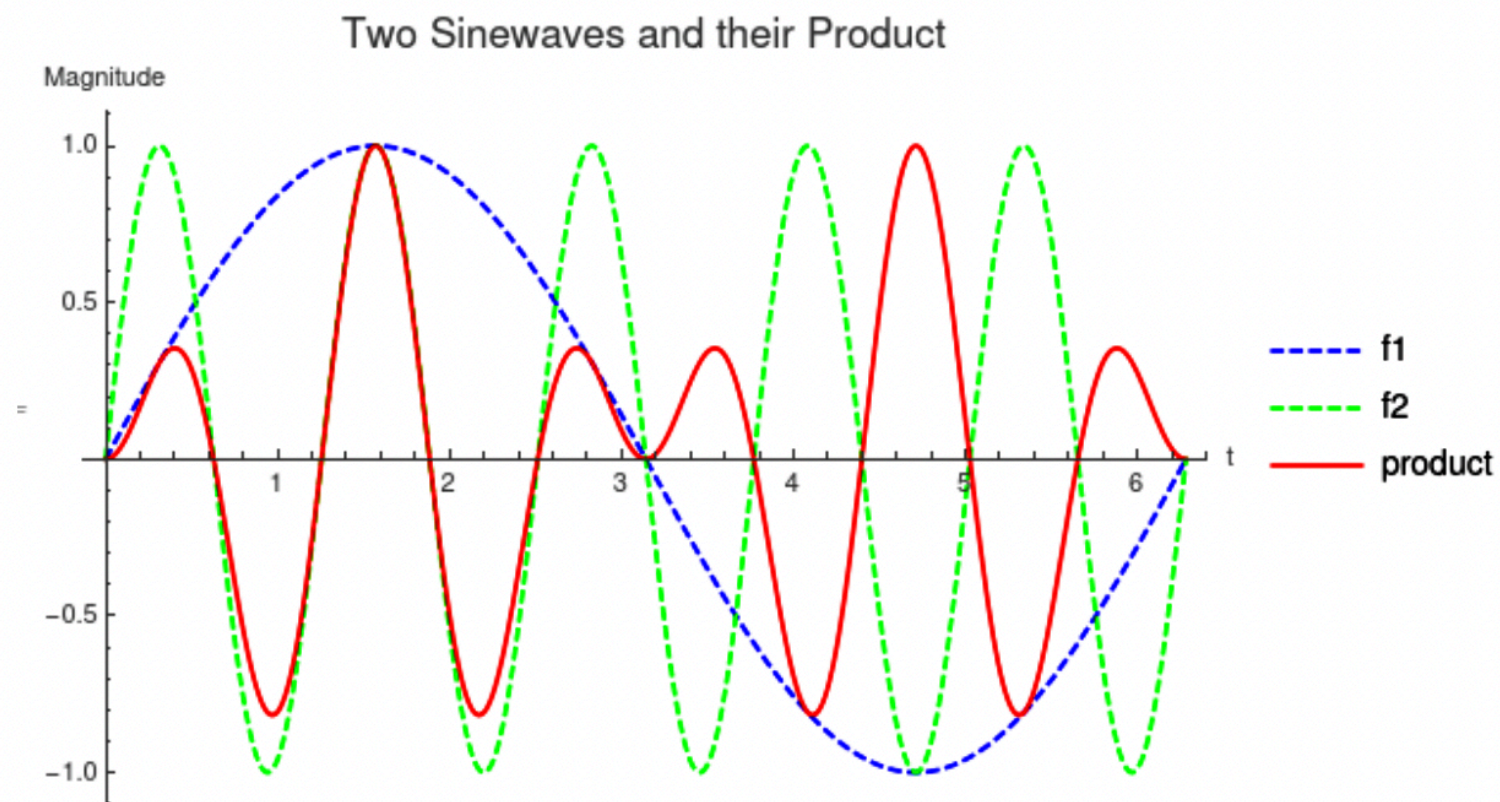
FrameLabel  $\rightarrow$  Style["Product of Two Sinewaves", 16]]

```



Combined Plot

```
=  $\omega = 1$ ;  
f1 = Sin[1  $\omega$  t]; (* Lower Frequency *)  
f2 = Sin[5  $\omega$  t]; (* Higher Frequency *)  
product = f1 f2; (* product *)  
  
Plot[{f1, f2, product}, {t, 0, 2  $\pi$ },  
  
ImageSize  $\rightarrow$  500, PlotLegends  $\rightarrow$  "Expressions",  
AxesLabel  $\rightarrow$  {"t", "Magnitude"},  
PlotLabel  $\rightarrow$  Style["Two Sinewaves and their Product", 16],  
PlotStyle  $\rightarrow$  {{Dashed, Blue}, {Dashed, Green}, Red}]
```



Frequency Domain (amplitude is not to scale)

`:=`

```
Manipulate[
  f1 = {m1, 0.8};
  f2 = {m2, 0.8};
  f3 = {Abs[m2 - m1],  $\frac{1}{2}$ }; (* Difference frequency *)
  f4 = {m2 + m1,  $\frac{1}{2}$ }; (* Sum frequency *)

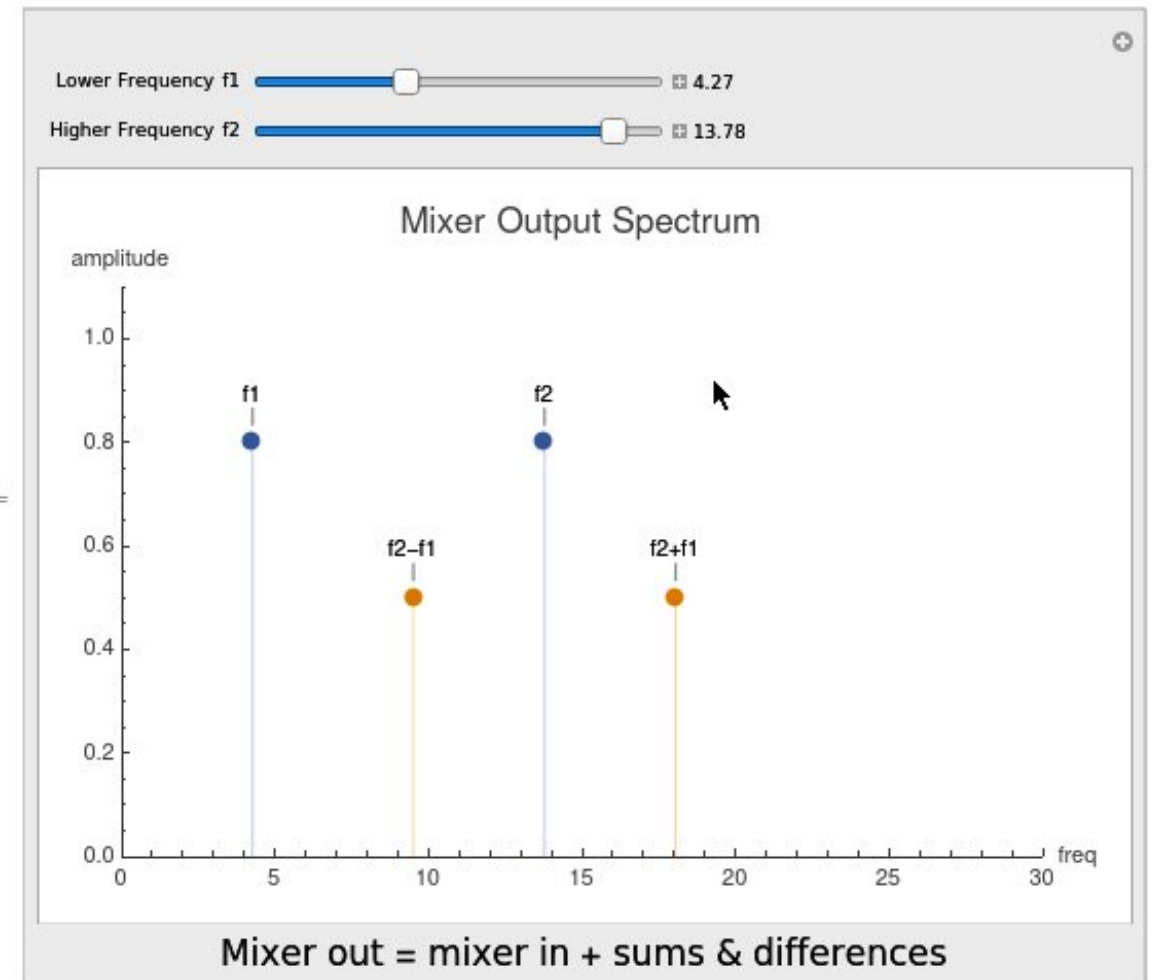
  dat1 = {f1, f2};
  dat2 = {f3, f4}; (* organize data *)

  ListPlot[{dat1 → {"f1", "f2"}, dat2 → {"f2-f1", "f2+f1"}},
```

```
  LabelingFunction → Top, PlotStyle → PointSize[0.02], Filling → Axis,
  PlotLabel → Style["Mixer Output Spectrum", 16],
  AxesLabel → {"freq", "amplitude"},
  Frame → False, ImageSize → 500,
  PlotRange → {{0, 30}, {0, 1.1}},
```

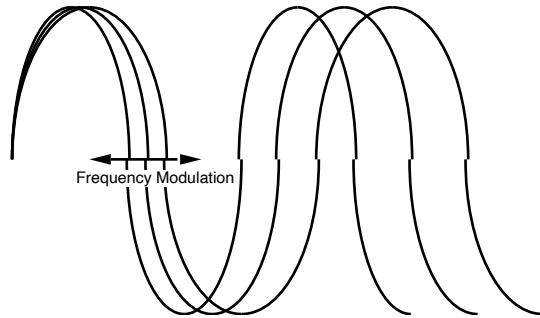
```
{ {m1, 5, "Lower Frequency f1"}, 1, 10, Appearance → "Labeled"},
{ {m2, 15, "Higher Frequency f2"}, 1, 15, Appearance → "Labeled"},
```

```
FrameLabel → Style["Mixer out = mixer in + sums & differences", 16] ]
```



Frequency Modulation

There are three parameters in a sinewave carrier that may be varied or modulated to convey information: amplitude, frequency, and phase.



Notice that frequency modulation looks very much like phase modulation. They are in fact very similar, and many textbooks refer to them both as angle modulation.

Recall that a general sinusoid is of the form:

$$e_c = \sin(\omega_c t + \theta)$$

Frequency modulation involves deviating a carrier frequency by some amount. If a sine wave was used to frequency modulate a carrier, the mathematical expression would be:

$$\omega_i = \omega_c + \Delta\omega \sin \omega_m t$$

where

ω_i = instantaneous frequency

ω_c = carrier frequency

$\Delta\omega$ = carrier deviation

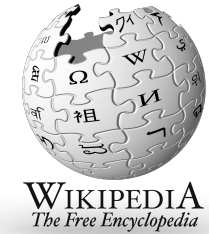
ω_m = modulation frequency

This expression shows a signal varying sinusoidally about some average frequency. However, we cannot simply substitute expression in the general equation for a sinusoid. This is because the sine operator acts upon angles, not frequency. Therefore, we must define the instantaneous frequency in terms of angles.

It should be noted that the amplitude of the modulation signal governs the amount of carrier deviation, while the modulation frequency governs the rate of carrier deviation.

The term ω is an angular velocity and it is related to frequency and angle by the following relationship:

$$\omega = 2\pi f = \frac{d\theta}{dt}$$



To find the angle, we must integrate ω with respect to time, we obtain:

$$\int \omega dt = \theta$$

We can now find the instantaneous angle associated with an instantaneous frequency:

$$\begin{aligned}\theta &= \int \omega_i dt = \int (\omega_c + \Delta\omega \sin \omega_m t) dt \\ &= \omega_c t - \frac{\Delta\omega}{\omega_m} \cos \omega_m t = \omega_c t - \frac{\Delta f}{f_m} \cos \omega_m t\end{aligned}$$

This angle can now be substituted into the general carrier signal to define FM:

$$e_{fm} = \sin \left(\omega_c t - \frac{\Delta f}{f_m} \cos \omega_m t \right)$$

The FM modulation index is defined as the ratio of carrier deviation to modulation frequency:

$$m_{fm} = \frac{\Delta f}{f_m}$$

As a result, the FM equation is generally written as:

$$e_{fm} = \sin (\omega_c t - m_{fm} \cos \omega_m t)$$

This is a very complex expression and it is not readily apparent what the sidebands of this signal are like. The solution to this problem requires knowledge of Bessel's functions of the first kind and order p . In open form, it resembles:

$$J_p(x) = \sum_{k=0}^{\infty} \frac{(-1)^k \left(\frac{x}{2}\right)^{2k+p}}{k!(k+p)!}$$

where:

$J_p(x)$ = magnitude of frequency component

p = side frequency number

x = modulation index

As a point of interest, Bessel's functions are a solution to the following equation:

$$x^2 + \frac{d^2 y}{dx^2} + x \frac{dy}{dx} + (x^2 - p^2)y = 0$$

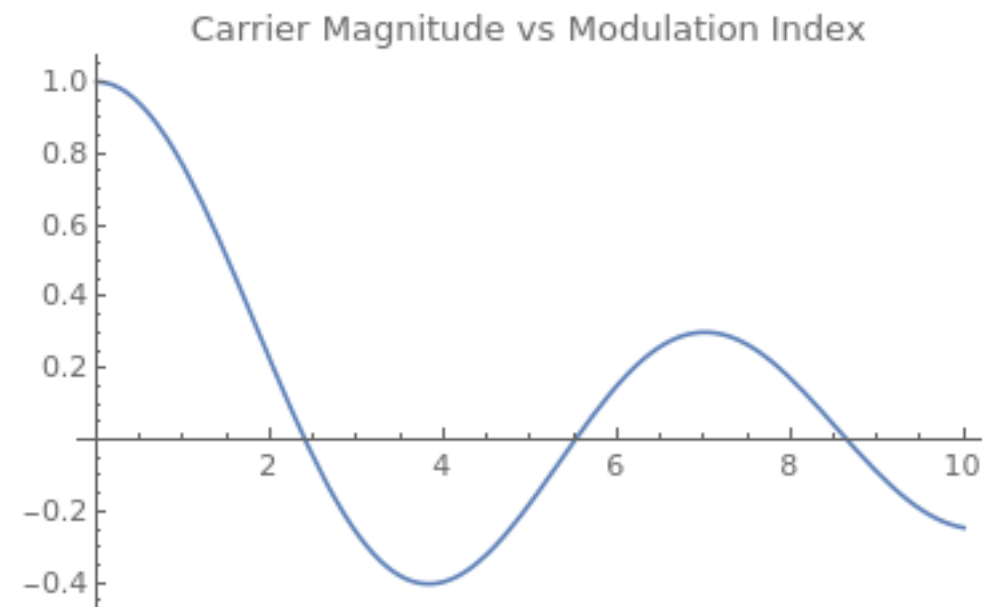
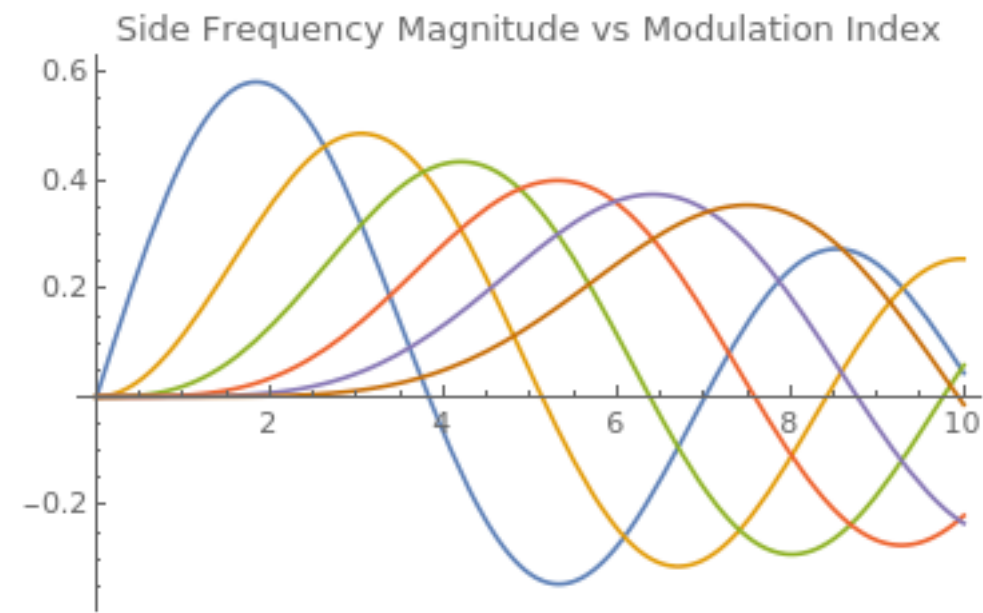
Bessel's functions occur in the theory of cylindrical and spherical waves, much like sine waves occur in the theory of plane waves.

It turns out that FM generates an infinite number of side frequencies. Each frequency is an integer multiple of the

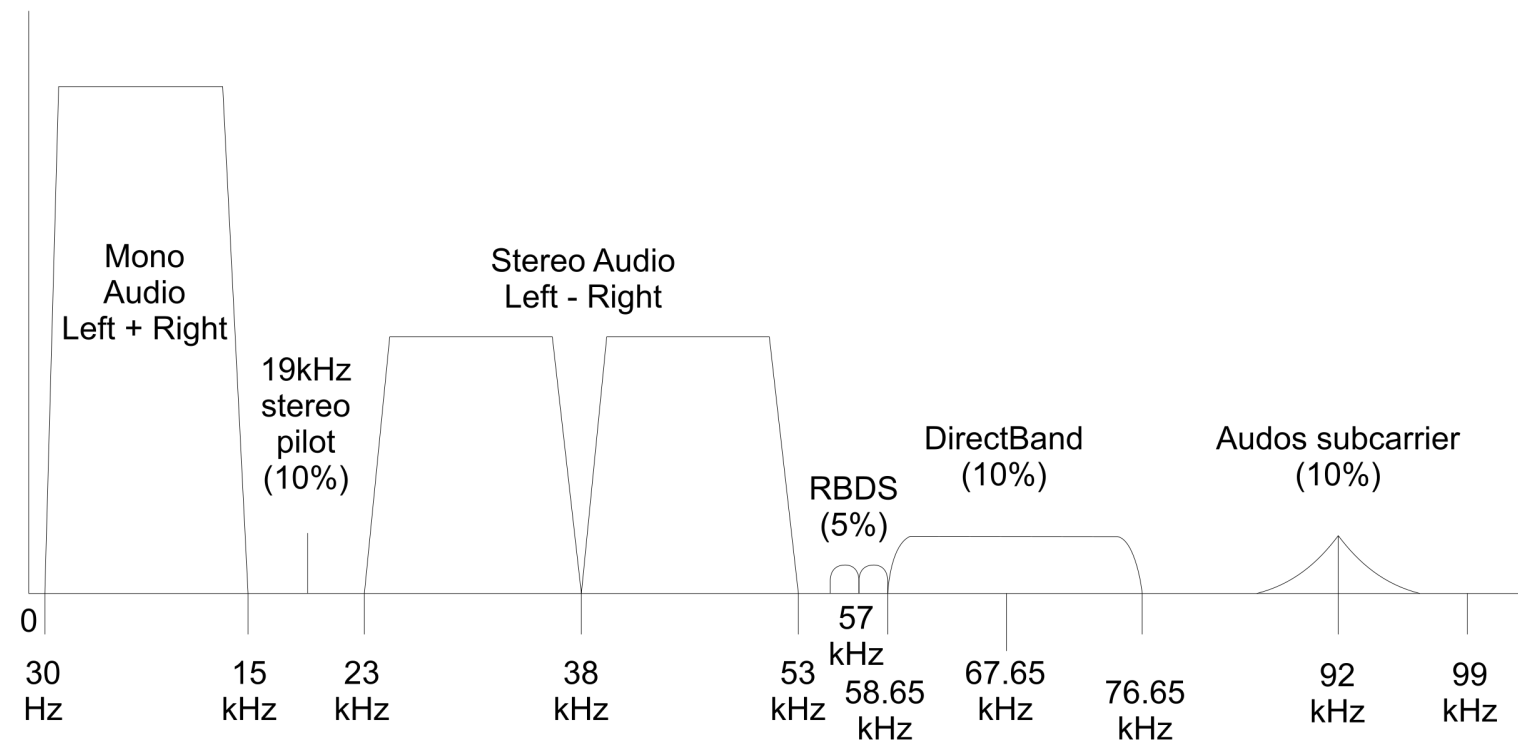
modulation signal. It should be noted that the amplitude of the higher order side frequencies drops off quickly.

It is also interesting to note that the amplitude of the carrier signal is also a function of the modulation index. Under some conditions, the amplitude of the carrier frequency can actually go to zero. This does not mean that the signal disappears, but rather that all of the broadcast energy is redistributed to the side frequencies.

In [FM](#) systems, the signal envelope remains constant as does the transmitted power. However, as the carrier frequency is deviated, the power gets redistributed from the carrier into side frequencies. This increases the transmission bandwidth, also known as spread spectrum. Mathematically, this phenomenon is described by [Bessel's functions](#) of the first kind and order p .



FM Radio Baseband Spectrum



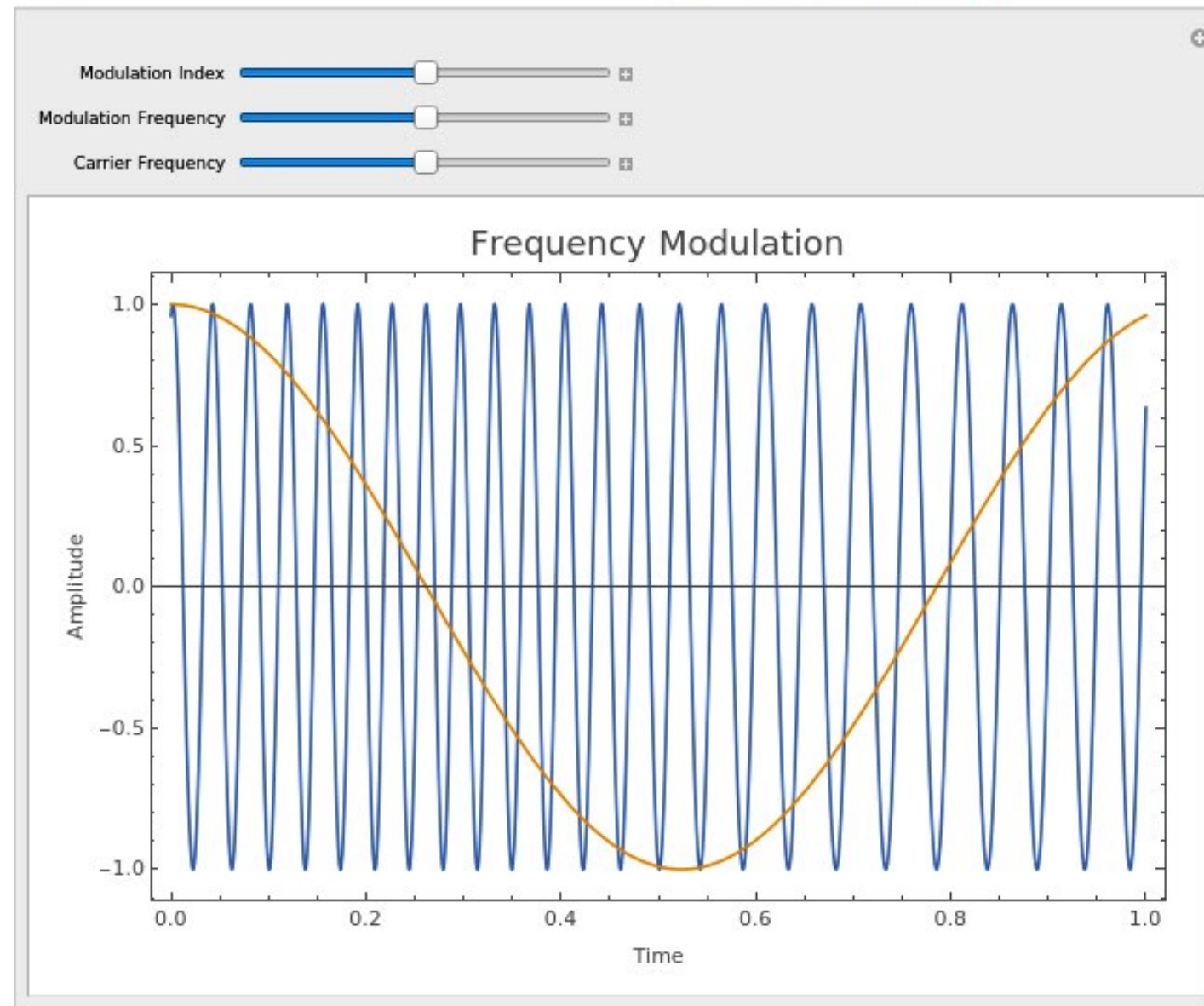
Courtesy: [Wikipedia By Arthur Murray](#)

The FM spectrum bandwidth is approximated by [Carson's Rule](#): $CBR = 2(f_m + \Delta f)$ or twice the modulation and deviation frequencies.

For [Commercial FM broadcast](#), the maximum baseband modulation frequency (f_m) is 15 KHz, although this is increased to 53 KHz for FM stereo, and even higher for specialty services. The maximum modulation index is 5, and the maximum carrier deviation (Δf) is 75 KHz. To keep the transmitted RF signal within the allocated frequency slot, the modulation index is reduced (by decreasing the amplitude of baseband signals above 15 KHz), thus reducing the carrier deviation (from a maximum of 75 KHz) for high frequency signals and keeping within the assigned 200 KHz broadcasting slot.

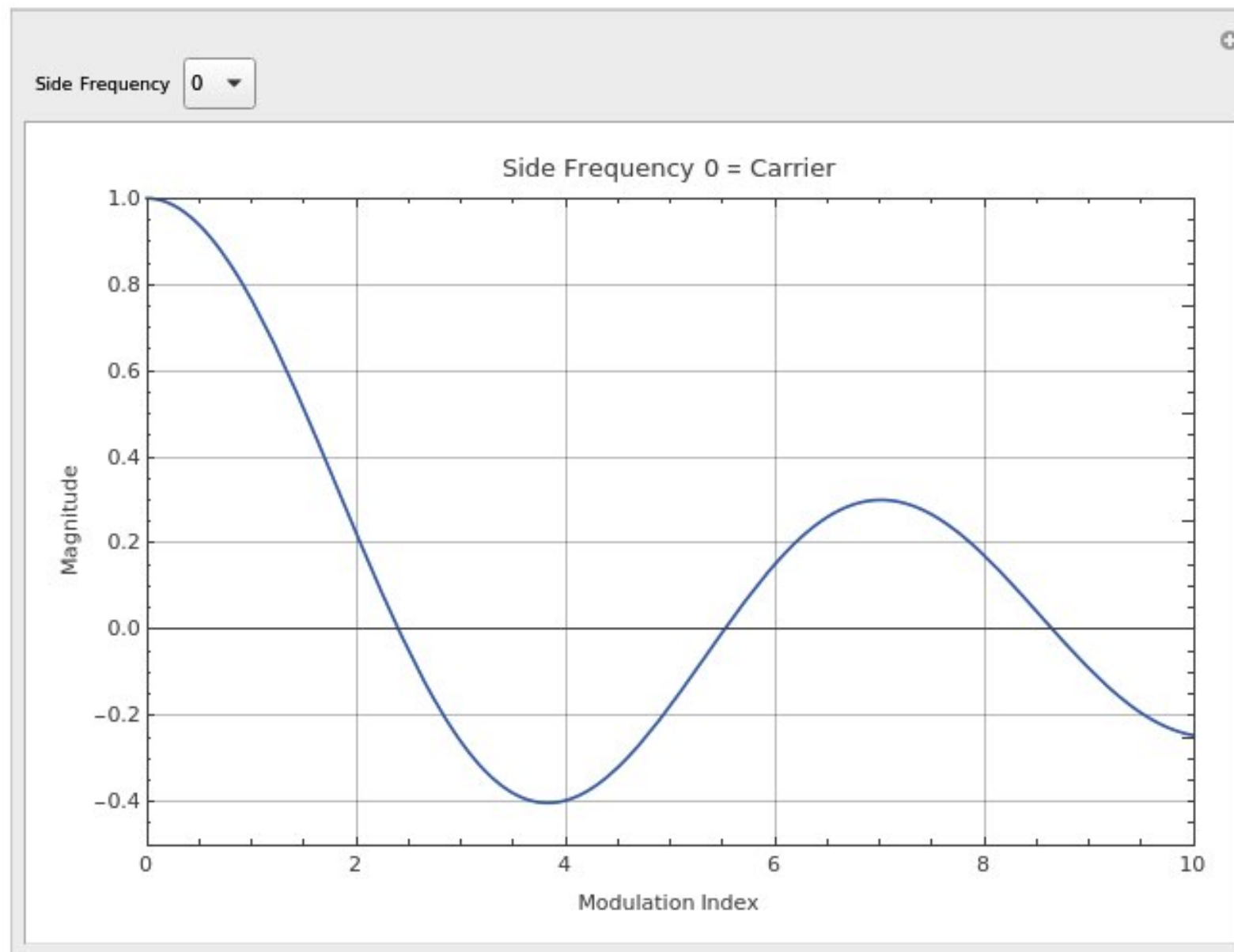
Manipulate Modulation Index, Modulation Frequency, & Carrier Frequency in the Time Domain

```
Manipulate[Plot[{Sin[ $\omega_c t - mfm \cos[\omega_m t]$ ], Cos[ $\omega_m t$ ]}, {t, 0, 1},  
  PlotLabel → Style["Frequency Modulation", 16],  
  Frame → True,  
  FrameLabel → {"Time", "Amplitude"},  
  ImageSize → {600, 400}],  
  {{mfm, 5, "Modulation Index"}, 0, 10}, (* modulation index *)  
  {{ $\omega_m$ , 6, "Modulation Frequency"}, 2, 10}, (* modulation frequency *)  
  {{ $\omega_c$ , 150, "Carrier Frequency"}, 100, 200} (* carrier frequency *)]
```



Side Frequency Amplitude as a Function of Modulation Index

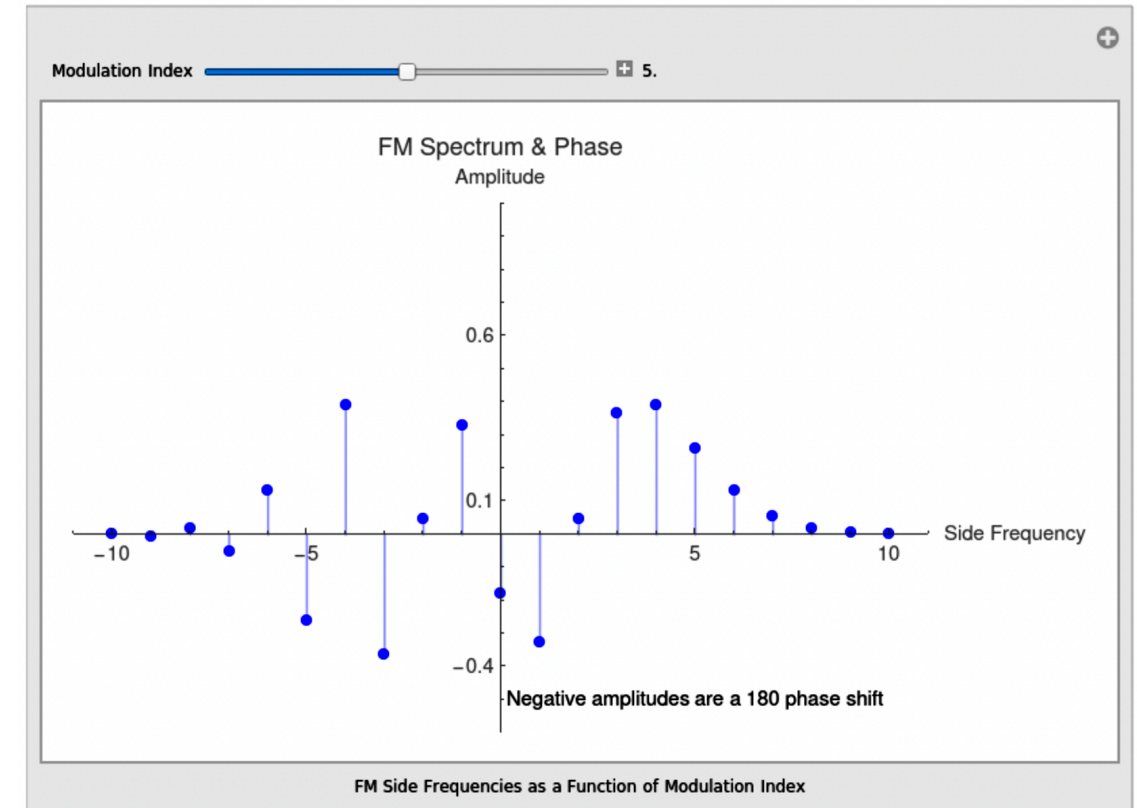
```
Manipulate[Plot[BesselJ[sf, mi], {mi, 0, 10},  
  (* sf = Side Frequency    mi = Modulation Index *) ImageSize → {600, 400},  
  PlotLabel → "Side Frequency 0 = Carrier",  
  FrameLabel → {"Modulation Index", "Magnitude"},  
  PlotRange → {{0, 10}, {-0.5, 1}},  
  Frame → True,  
  GridLines → Automatic], {{sf, 0, "Side Frequency"}, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}}]
```



FM Spectrum as a Function of Modulation Index

```
Clear[mi, sf, spectrum]
Manipulate[
  spectrum = Table[{sf, BesselJ[sf, mi]}, {sf, -10, 10}];
  ListPlot[spectrum,
    PlotStyle → Blue,
    PlotRange → {{-11, 11}, {-0.6, 1}},
    Filling → Axis,
    PlotLabel → "FM Spectrum & Phase",
    AxesLabel → {"Side Frequency", "Amplitude"},
    ImageSize → 500,
    Epilog → {Text["Negative amplitudes are a 180 phase shift", {5, -0.5}]}],

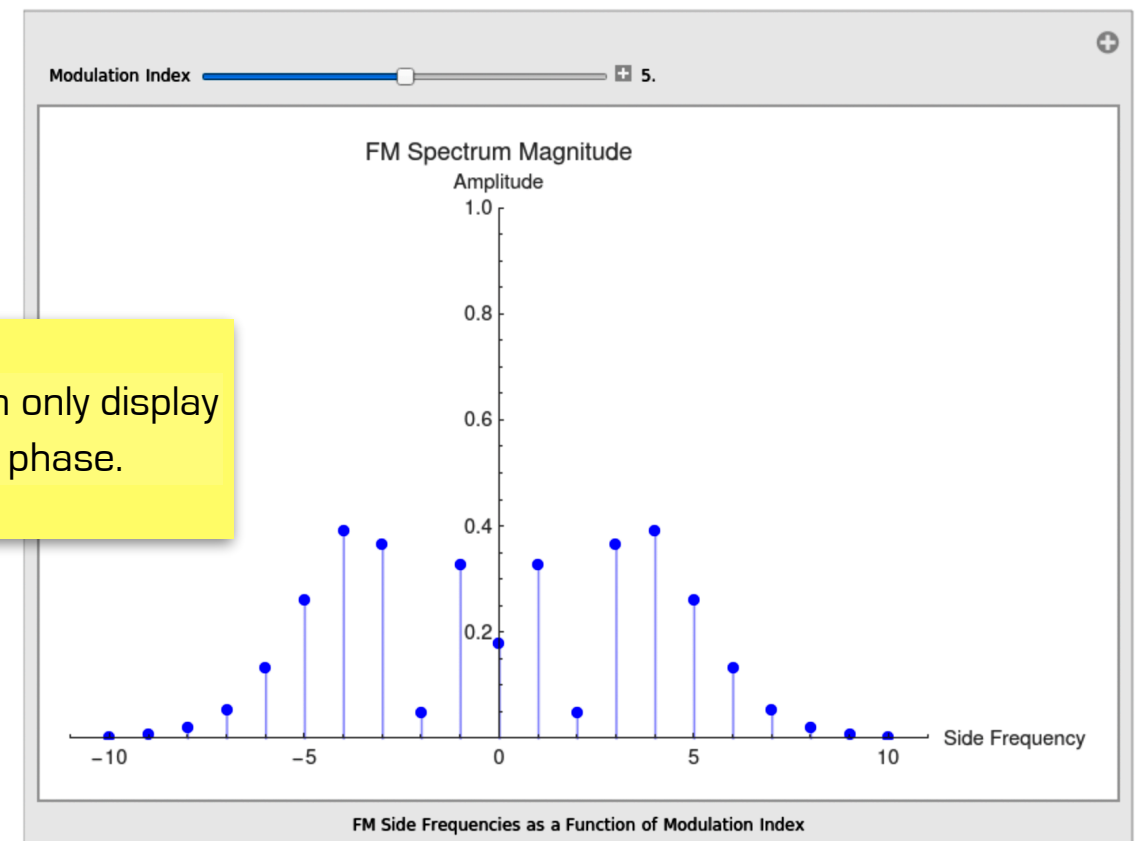
  {{mi, 5.0, "Modulation Index"}, 0, 10, Appearance → "Labeled"},
  FrameLabel → "FM Side Frequencies as a Function of Modulation Index"]
```



```
Clear[mi, sf, spectrum]
Manipulate[
  spectrum = Table[{sf, Abs[BesselJ[sf, mi]]}, {sf, -10, 10}];
  ListPlot[spectrum,
    PlotStyle → Blue,
    PlotRange → {{-11, 11}, {0, 1}},
    Filling → Axis,
    PlotLabel → "FM Spectrum Magnitude",
    AxesLabel → {"Side Frequency", "Amplitude"},
    ImageSize → 500],

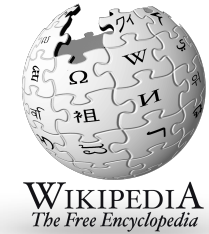
  {{mi, 5.0, "Modulation Index"}, 0, 10, Appearance → "Labeled"},
  FrameLabel → "FM Side Frequencies as a Function of Modulation Index"]
```

A spectrum analyzer can only display magnitude and not phase.



FSK

Frequency shift keying is a form of digital modulation. Two frequencies are used to represent the binary signal.



Spectral analysis of a sine wave of finite duration (time window is set to 1 second)

```
Clear[ns, fs, dt, ft, ift, ifdt, datf, datp, f1, f2]
```

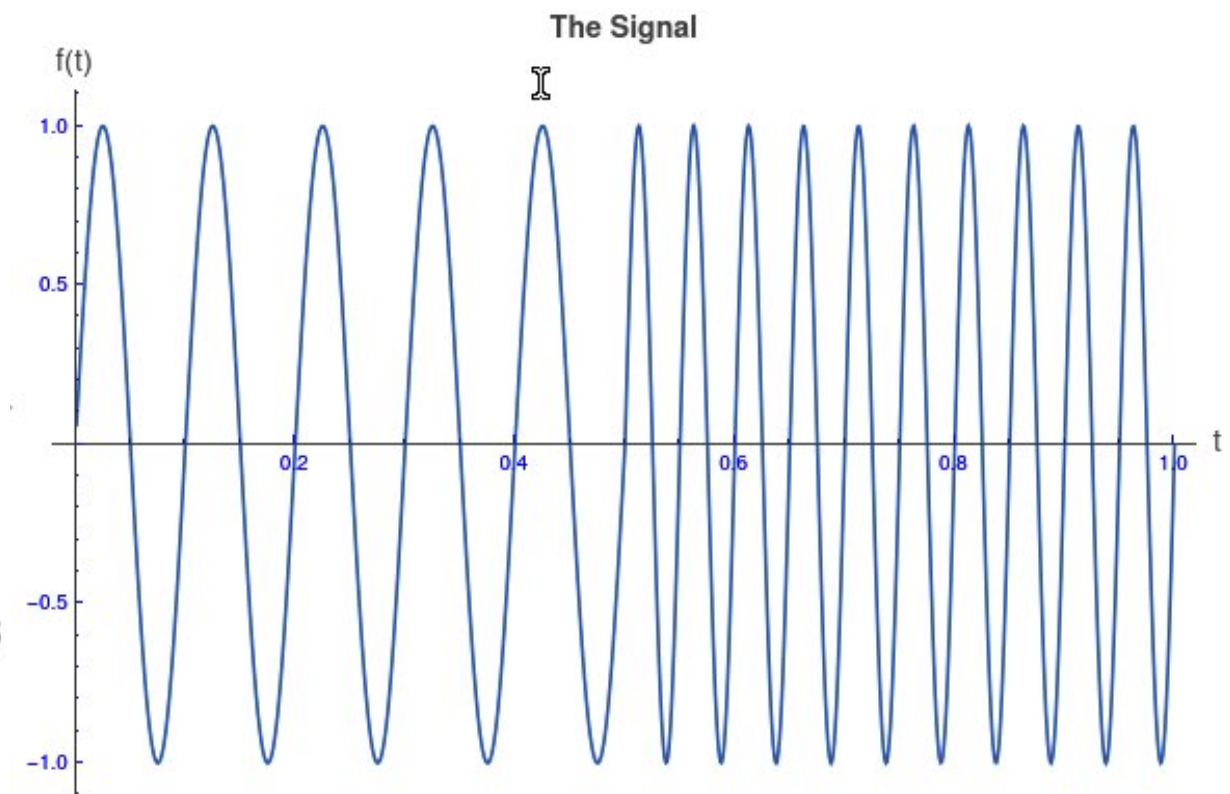
```
ns = 1024; (* number of samples *)  
fs = 10.0; (* Signal Frequency (Hz) *)  
dt =  $\frac{1}{ns}$ ; (* Time increment (sec) *)
```

Tabulate the signal and time components

```
48]- f1 = Sin[2  $\pi$  fs n dt];  
f2 = Sin[2  $\pi$  2.0 fs n dt];  
  
dat = Table[Piecewise[{{f1, n <  $\frac{ns}{2}$ }, {f2, n  $\geq$   $\frac{ns}{2}$ }}, {n, ns}];  
(* create a table of signal data *)  
datp = Table[{k dt, dat[[k]]}, {k, 1, ns}];  
(* Add the time scale to the signal table *)
```

Plot the Time Domain

```
ListPlot[datp, Joined  $\rightarrow$  True, AxesLabel  $\rightarrow$  {"t", "f(t)"},  
PlotLabel  $\rightarrow$  Style["The Signal", Bold], (* plot the signal vs time *)  
TicksStyle  $\rightarrow$  Blue, ImageSize  $\rightarrow$  Large]
```



Perform the Fourier Transform

```
ft = Fourier[dat, FourierParameters -> {1, -1}]; (* Apply the Fourier Transform *)
```

```
datf = Table[{(k - 1) / (ns dt), 4 Abs[ft[[k]]] / ns}, {k, 1,  $\frac{ns}{2}$ }]
```

```
(* Create a table up to the Nyquist Rate *)
```

Plot the Frequency Domain

```
ListPlot[datf[[1 ;;  $\frac{ns}{2}$ ]], (* a;;b means the span of elements from a to b *)
```

```
PlotLabel -> Style["Frequency Spectrum", Bold],
```

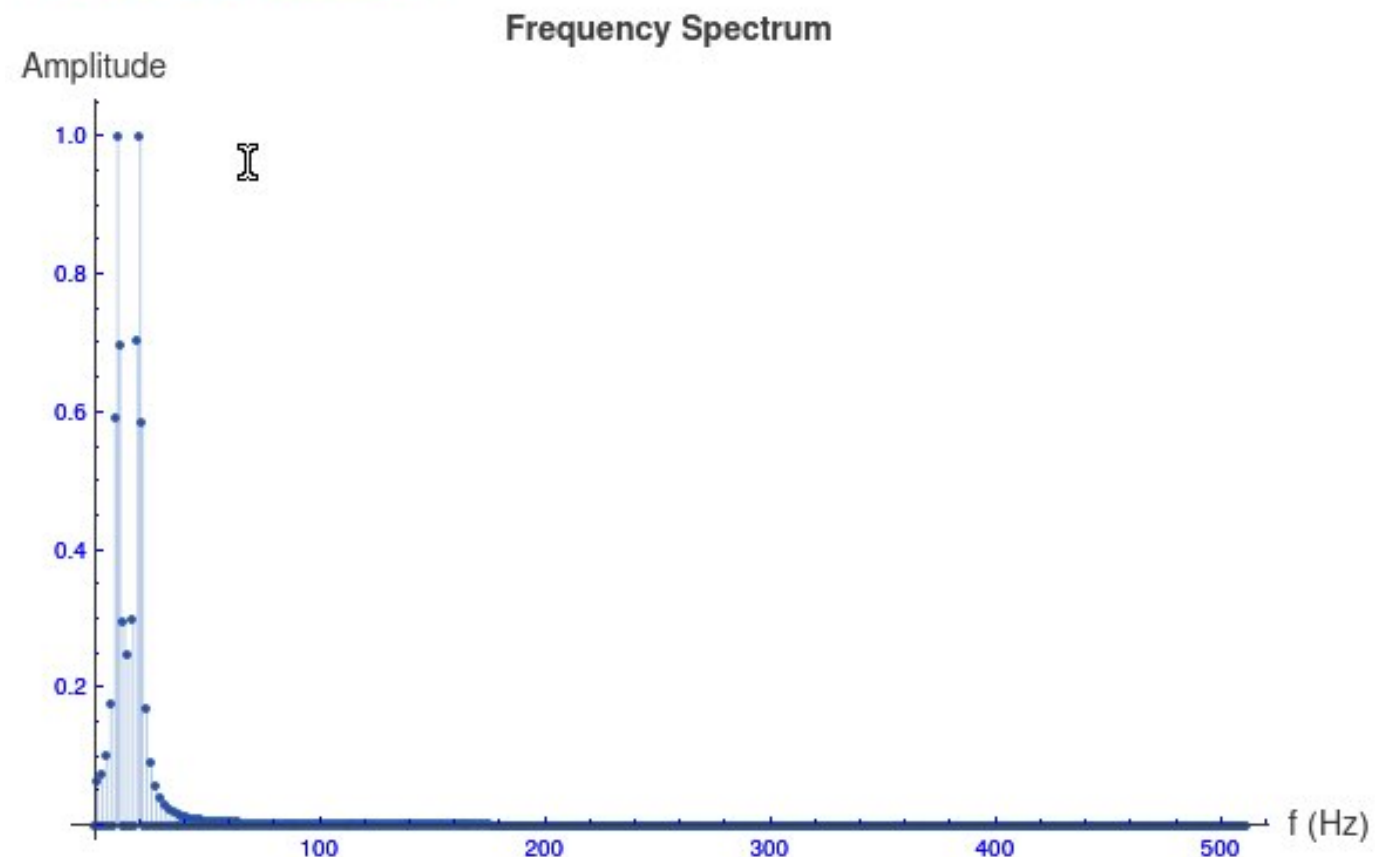
```
PlotRange -> All,
```

```
Filling -> Axis, TicksStyle -> Blue,
```

```
AxesLabel -> {"f (Hz)", "Amplitude"},
```

```
ImageSize -> Large]
```

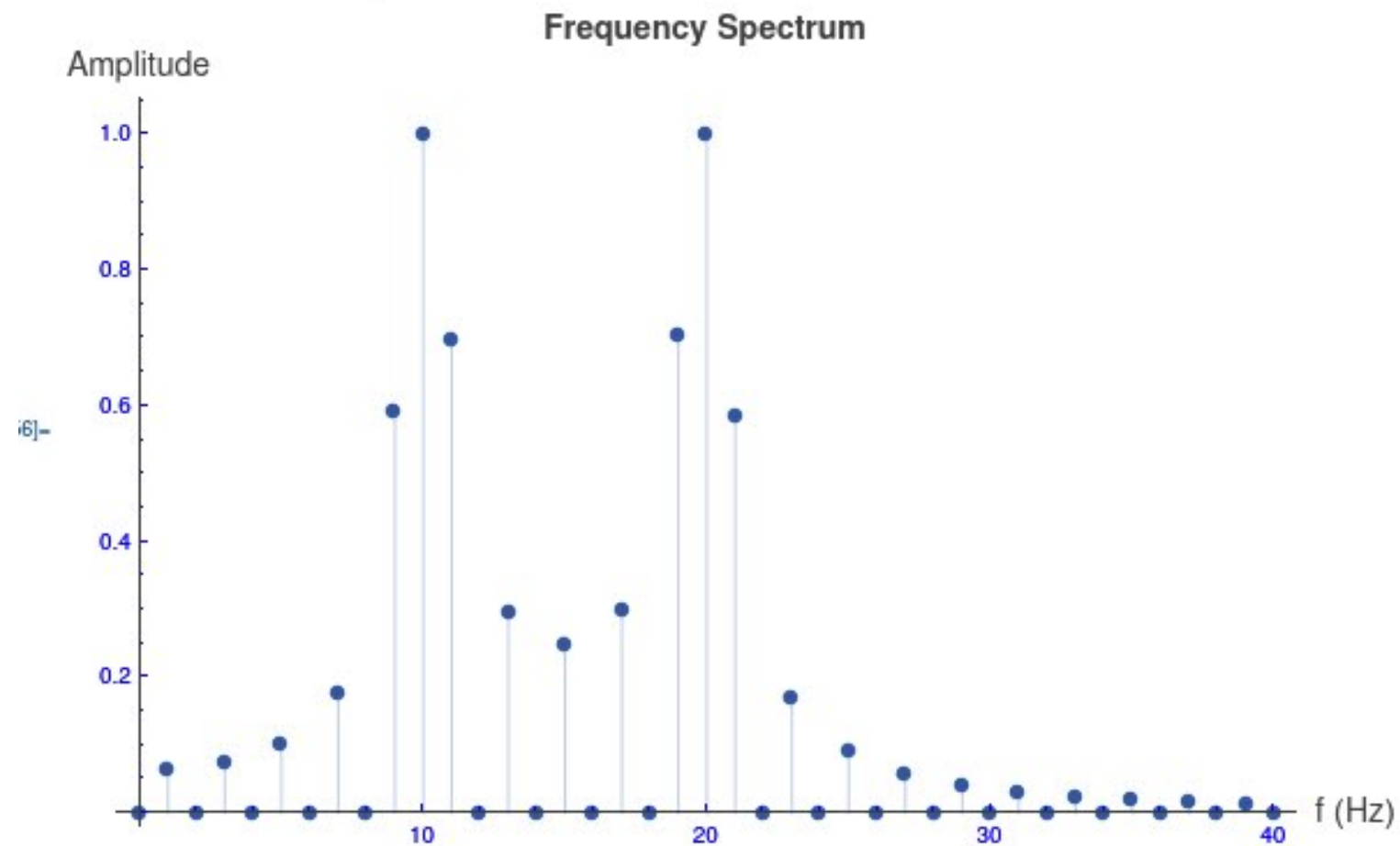
```
(* Plot the Spectrum *)
```



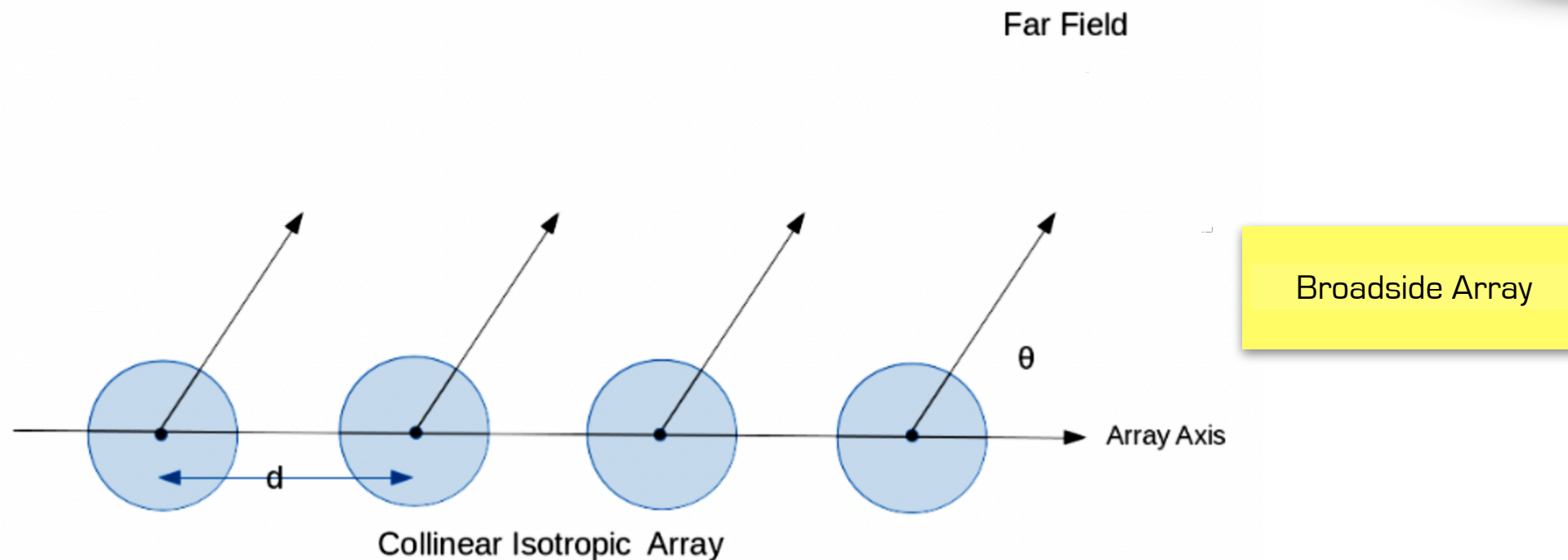
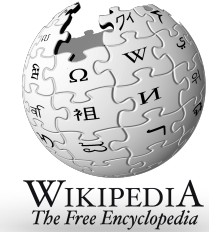
A Closer Look at the Spectrum

```
5):- fmin = 0; (* minimum frequency *)  
    fmax = 40; (* maximum frequency *)
```

```
ListPlot[datf[[fmin + 1 ;; fmax + 1]],  
  PlotLabel → Style["Frequency Spectrum", Bold],  
  PlotRange → All,  
  Filling → Axis, TicksStyle → Blue,  
  AxesLabel → {"f (Hz)", "Amplitude"},  
  ImageSize → Large]
```



Collinear Isotropic Array



Stars are essentially isotropic sources, since they radiate nearly the same amount of energy in all directions. However, they differ from radio sources in one very significant difference; they are vastly larger than the wavelength they radiate.

Collinear radio sources can be spaced such that they cancel each other out in a specific direction. This is known as destructive interference. The net result is that two or more closely spaced antenna based isotropic radio sources are no longer isotropic. They will radiate more in one direction and less in another.

The change from a spherical radiation pattern to a shaped one is known as the array factor.

There are many different equations for array factor, but the most common ones make certain assumptions:

- Radiating elements are identical in every way

- Element spacing is within a few wavelengths of each other

- All elements have a constant phase shift with respect to each other

4 Element Isotropic Collinear Array: Linear Plot

Broadside Array

```
Clear[n, λ, d, α, af]
```

```
n = 4; (* number of radiators *)
```

```
λ = 1; (* wavelength in meters *)
```

```
d = 0.5 λ; (* element spacing in wavelengths *)
```

```
α = 0; (* phase shift between elements *)
```

$$\psi = \frac{2\pi}{\lambda} d \cos[\theta] + \alpha;$$

$$af := \text{Abs}\left[\frac{\sin\left[\frac{n\psi}{2}\right]}{\sin\left[\frac{\psi}{2}\right]}\right];$$

$$naf = \frac{af}{n}; \quad (* \text{Normalized Array Factor} *)$$

```
Plot[{af, naf}, {θ, 0, 2 π},
```

```
PlotRange → All,
```

```
PlotLabel → n "Element Array Factor",
```

```
Frame → True,
```

```
FrameLabel → {"Radians from axis", "magnitude
```

```
FrameTicks → {{π/2, π, 3π/2}, Automatic},
```

```
GridLines → {{π/2, π, 3π/2}, Automatic},
```

```
PlotLegends → "Expressions",
```

```
Epilog → {
```

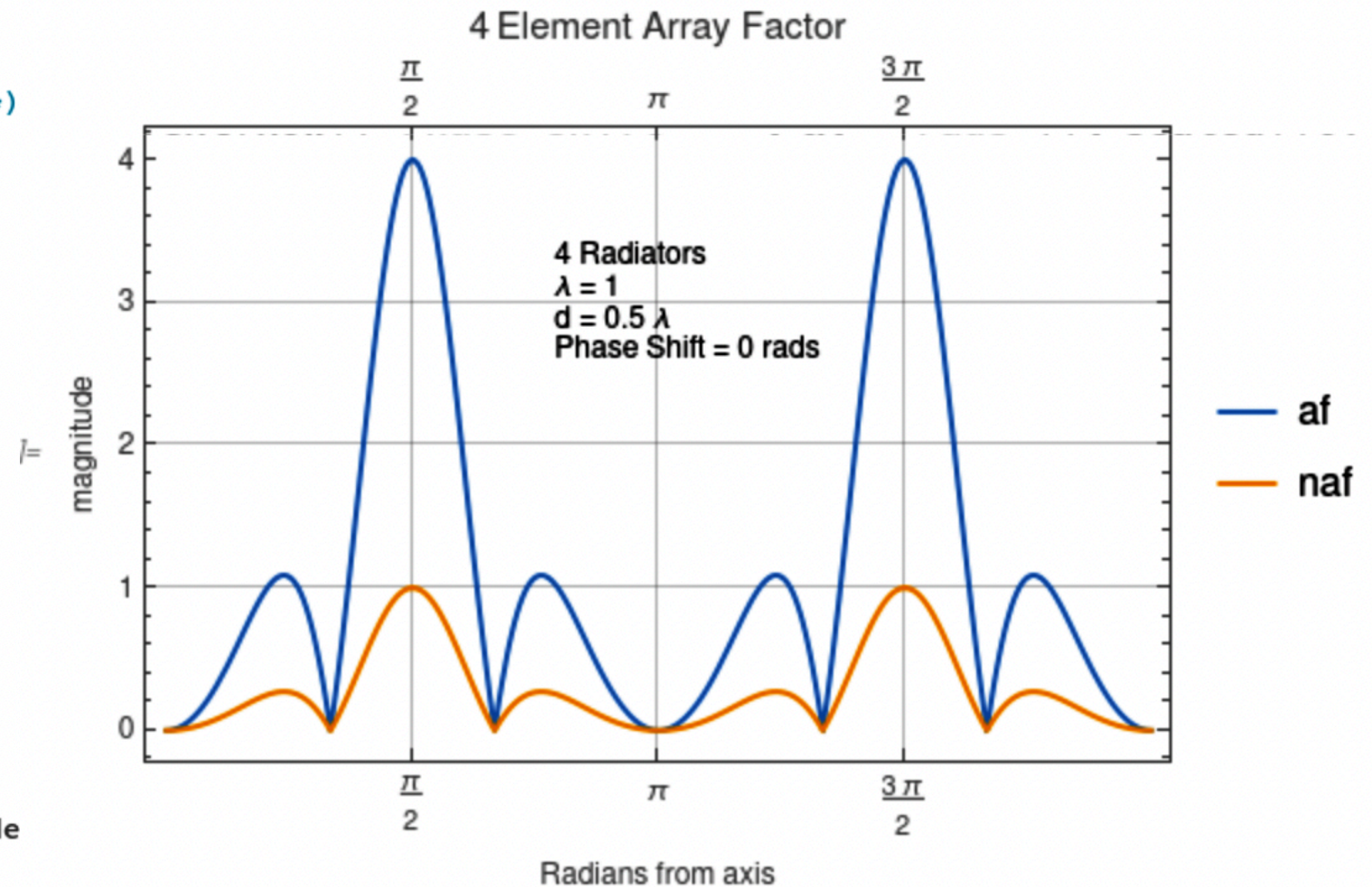
```
Text[Row[{n, " Radiators"}], Scaled[{0.4, 0.8}], Left],
```

```
Text[Row[{"λ = ", λ}], Scaled[{0.4, 0.75}], Left],
```

```
Text[Row[{"d = ", d, " λ"}], Scaled[{0.4, 0.7}], Left],
```

```
Text[Row[{"Phase Shift = ", α, " rads"}], Scaled[{0.4, 0.65}], Left]
```

```
}]
```



Manipulate 4 Element Isotropic Collinear Array: Linear Plot

```
Clear[λ, af, n, d, α]
```

```
Manipulate[
```

```
λ = 1;
```

```
af := Abs[ $\frac{\sin[\frac{n\psi}{2}]}{\sin[\frac{\psi}{2}]}$ ];  $\psi = \frac{2\pi}{\lambda} d \cos[\theta] + \alpha$ ;
```

```
Plot[af, {θ, 0, 2π},
```

```
PlotRange → All,
```

```
FrameTicks → {{ $\frac{\pi}{2}$ , π,  $\frac{3\pi}{2}$ }, Automatic},
```

```
GridLines → {{ $\frac{\pi}{2}$ , π,  $\frac{3\pi}{2}$ }, Automatic},
```

```
Frame → True,
```

```
FrameLabel → {"Radians from axis", "magnitude"}],
```

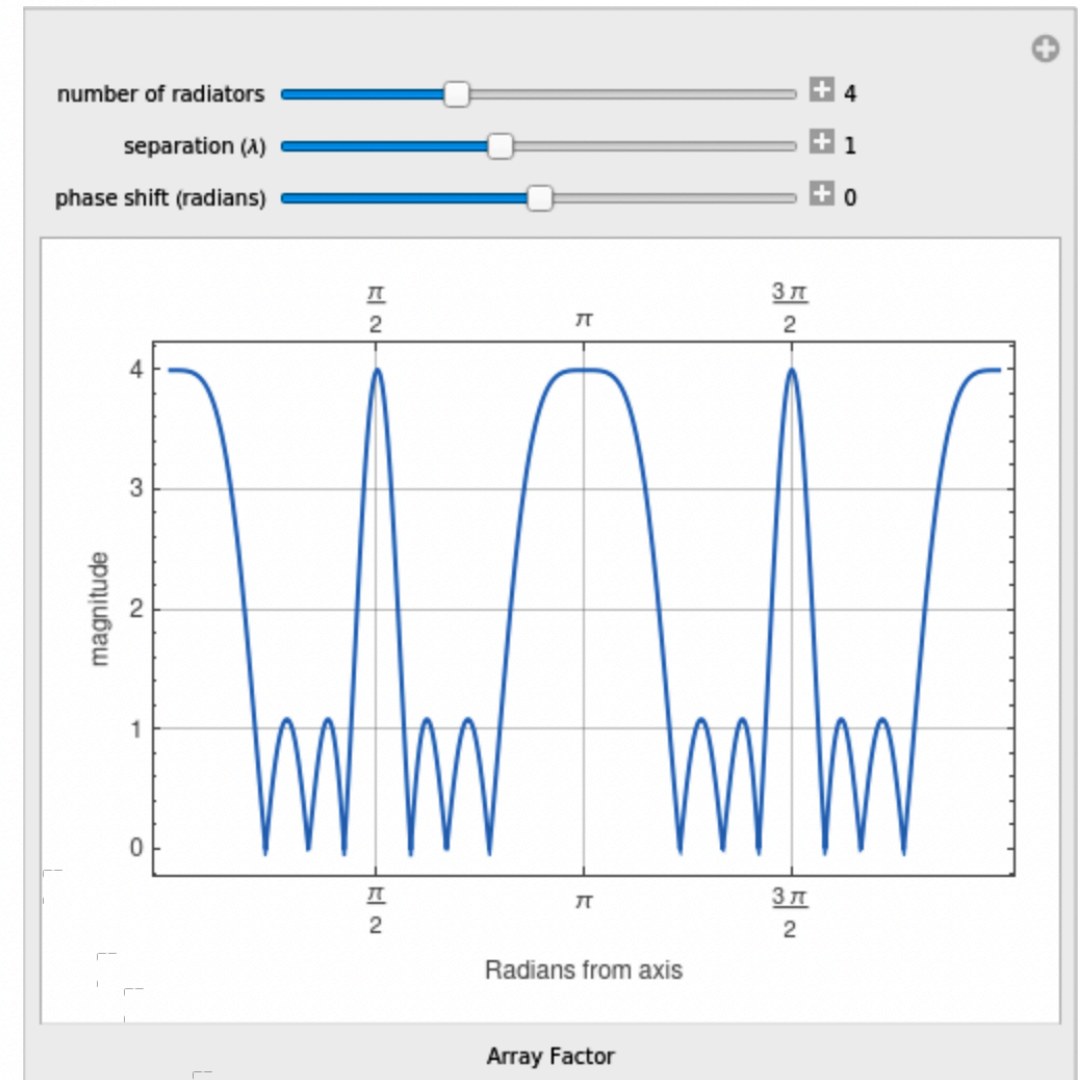
```
{{n, 4, "number of radiators"}, 1, 10, 1, Appearance → "Labeled"},
```

```
{{d, 1, "separation (λ)"}, 0.25, 2, 0.25, Appearance → "Labeled"},
```

```
{{α, 0, "phase shift (radians)"}, -π, π,  $\frac{\pi}{10}$ , Appearance → "Labeled"},
```

```
FrameLabel → "Array Factor"]
```

By manipulating the phase shift or separation, the antenna can broadcast most of its energy in the end-fire mode.

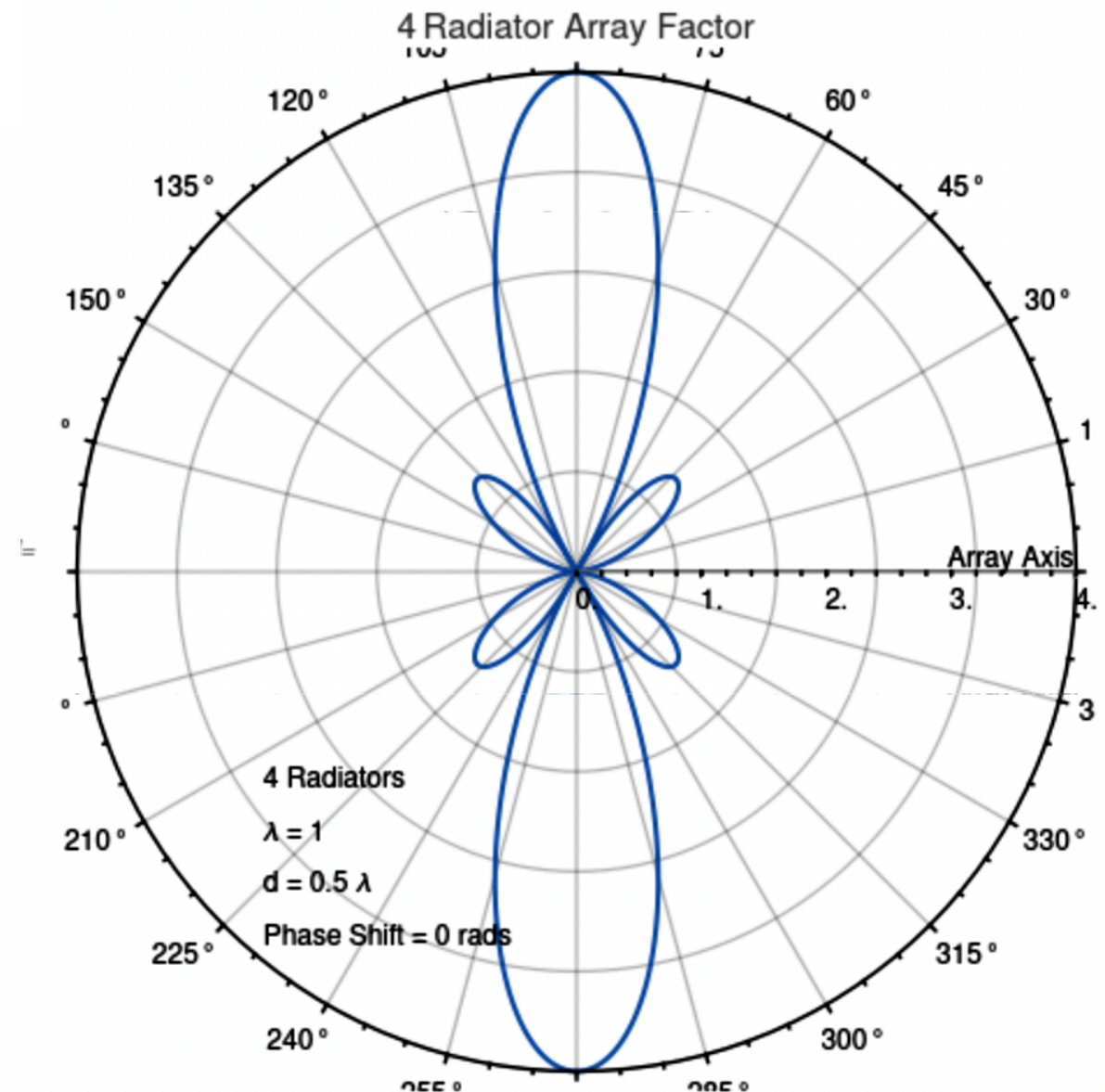



```
Clear[af]
```

```
n = 4; λ = 1; d = 0.5; α = 0;
```

```
af := Abs[ $\frac{\sin[\frac{n\psi}{2}]}{\sin[\frac{\psi}{2}]}$ ];  $\psi = \frac{2\pi}{\lambda} d \cos[\theta] + \alpha$ ;
```

```
PolarPlot[{af}], {θ, 0, 2 π},
  PolarAxes → True,
  PolarGridLines → {Automatic, Automatic},
  PolarTicks → {"Degrees", Automatic},
  PlotLabel → n "Radiator Array Factor",
  PolarAxesOrigin → {0, n},
  ImageSize → Medium,
  PlotLegends → "Expressions",
  Epilog → {Text["Array Axis", {n, 0.1}, Right],
    Text[Row[{n, " Radiators"}], Scaled[{0.2, 0.3}], Left],
    Text[Row[{"λ = ", λ}], Scaled[{0.2, 0.25}], Left],
    Text[Row[{"d = ", d, " λ"}], Scaled[{0.2, 0.2}], Left],
    Text[Row[{"Phase Shift = ", α, " rads"}], Scaled[{0.2, 0.15}], Left]
  ]]
```




```
Clear[n, d, α, af]
```

Manipulating a polar plot.

```
Manipulate[
```

```
λ = 1;
```

```
af := Abs[ $\frac{\sin[\frac{n\psi}{2}]}{\sin[\frac{\psi}{2}]}$ ];  $\psi = \frac{2\pi}{\lambda} d \cos[\theta] + \alpha$ ;
```

```
PolarPlot[af, {θ, 0, 2 π},
```

```
PolarAxes → True,
```

```
PolarAxesOrigin → {0, n},
```

```
PolarGridLines → {Automatic, Automatic},
```

```
PolarTicks → {"Degrees", Automatic},
```

```
Epilog → {Text["Array Axis", {n, 0.1}, Right],
```

```
Text[Row[{n, " Isotropic Radiators"}], Scaled[{0.2, 0.3}], Left],
```

```
Text[Row[{"λ = ", λ}], Scaled[{0.2, 0.25}], Left],
```

```
Text[Row[{"d = ", d, " λ"}], Scaled[{0.2, 0.2}], Left],
```

```
Text[Row[{"Phase Shift = ", α, " rads"}], Scaled[{0.2, 0.15}], Left]
```

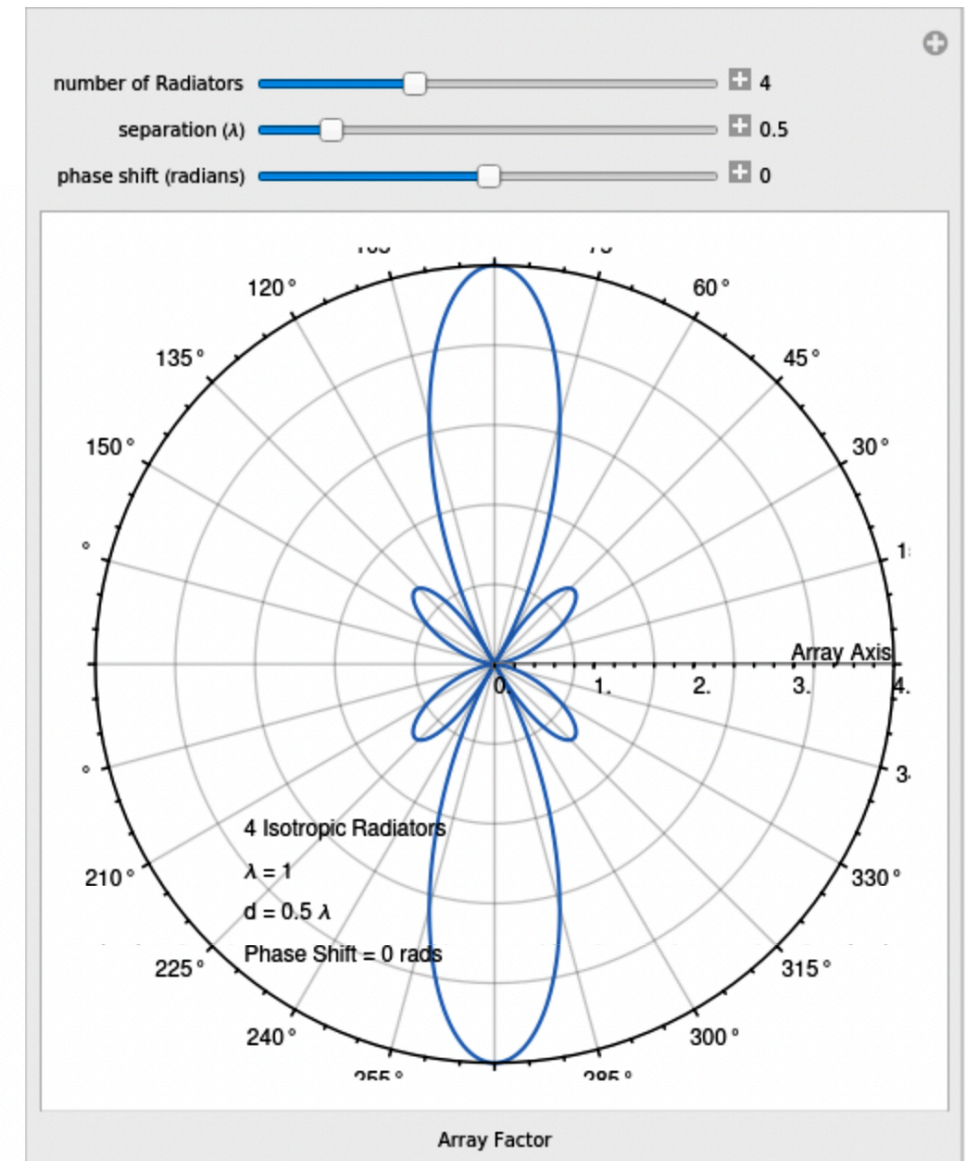
```
}},
```

```
{ {n, 4, "number of Radiators"}, 1, 10, 1, Appearance → "Labeled"},
```

```
{ {d, 0.5, "separation (λ)"}, 0.25, 2, 0.25, Appearance → "Labeled"},
```

```
{ {α, 0, "phase shift (radians)"}, - $\frac{\pi}{2}$ ,  $\frac{\pi}{2}$ ,  $\frac{\pi}{16}$ , Appearance → "Labeled"},
```

```
FrameLabel → "Array Factor"]
```




```
Clear[af, ne, k,  $\psi$ , d,  $\theta$ ,  $\phi$ ,  $\lambda$ ]
```

```
Manipulate[
```

$$af = \sum_{n=1}^{ne} e^{i(n-1)(kd \cos[\theta] + \psi)}; \quad (* \text{ array factor } *)$$

$$k = \frac{2\pi}{\lambda};$$

$$\lambda = 1;$$

$$naf = \frac{\text{Abs}[af]}{ne}; \quad (* \text{ normalized array factor } *)$$

```
PolarPlot[Abs[af], { $\theta$ , 0,  $2\pi$ },
```

```
PlotRange → All,
```

```
PolarAxes → True,
```

```
PolarGridLines → Automatic,
```

```
PolarAxesOrigin → {0, ne},
```

```
PolarTicks → {"Degrees", Automatic},
```

```
PlotLabel → "Array Factor",
```

```
Epilog → {Text["Array Axis", {ne, 0.1}, Right],
```

```
Text[Row[{ne, " Isotropic Radiators"}], Scaled[{0.2, 0.3}], Left],
```

```
Text[Row[{" $\lambda$  = ",  $\lambda$ }], Scaled[{0.2, 0.25}], Left],
```

```
Text[Row[{"d = ", d, "  $\lambda$ "}, Scaled[{0.2, 0.2}], Left],
```

```
Text[Row[{"Phase Shift = ",  $\psi$ , " rads"}], Scaled[{0.2, 0.15}], Left]
```

```
}},
```

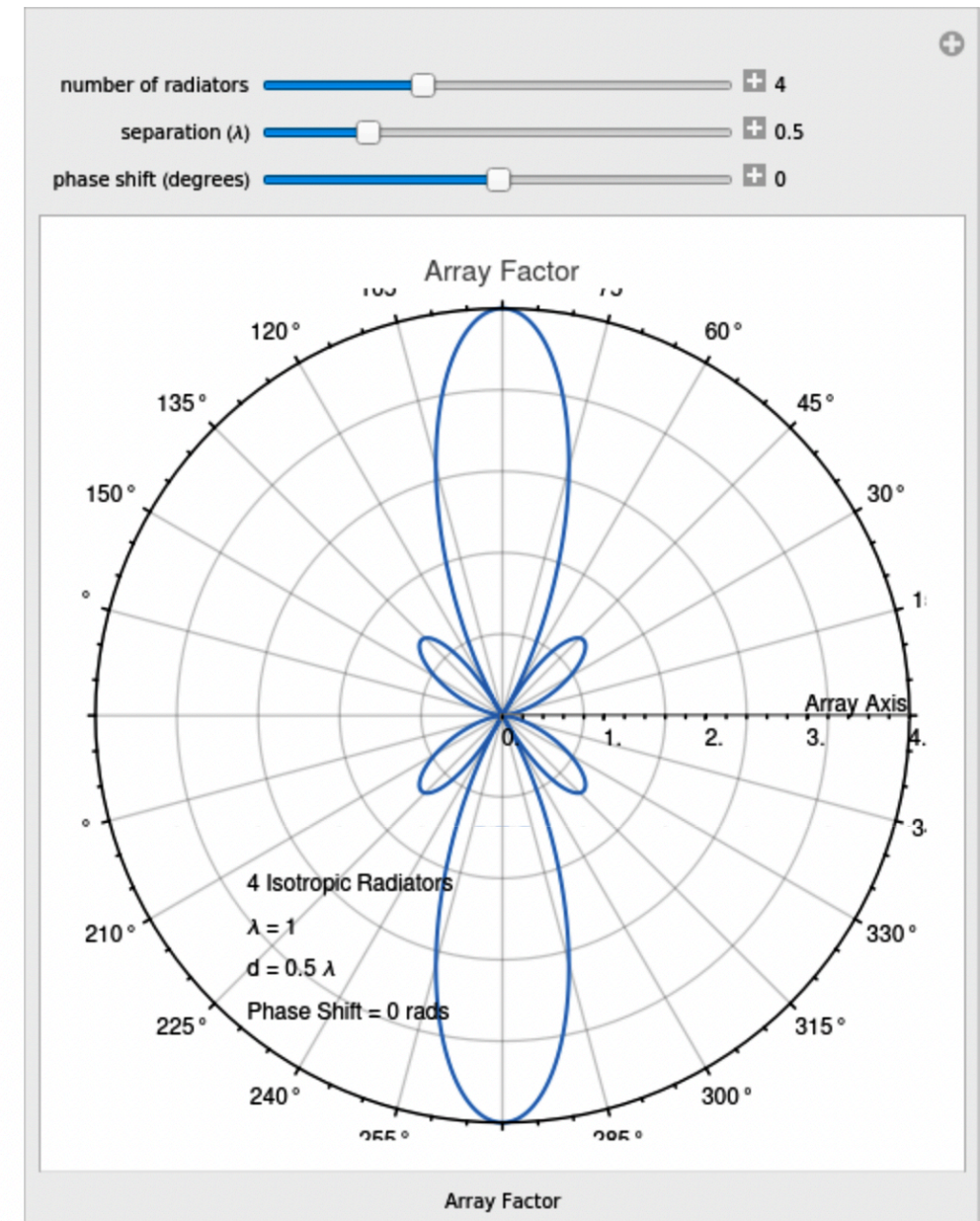
```
{{ne, 4, "number of radiators"}, 1, 10, 1, Appearance → "Labeled"},
```

```
{{d, 0.5, "separation ( $\lambda$ )"}, 0.1, 2, 0.1, Appearance → "Labeled"},
```

```
{{ $\psi$ , 0, "phase shift (degrees)"},  $-\pi$ ,  $\pi$ ,  $\frac{\pi}{10}$ , Appearance → "Labeled"},
```

```
FrameLabel → "Array Factor"]
```

An alternative array factor equation.



$$af = \sum_{n=1}^{ne} e^{i (n-1) (k d \cos[\theta] + \psi)}; (* \text{ array factor } *)$$

$$k = \frac{2 \pi}{\lambda};$$

$$\lambda = 1; ne = 4; d = 0.25; \psi = 0;$$

$$naf = \frac{\text{Abs}[af]}{ne}; (* \text{ normalized array factor } *)$$

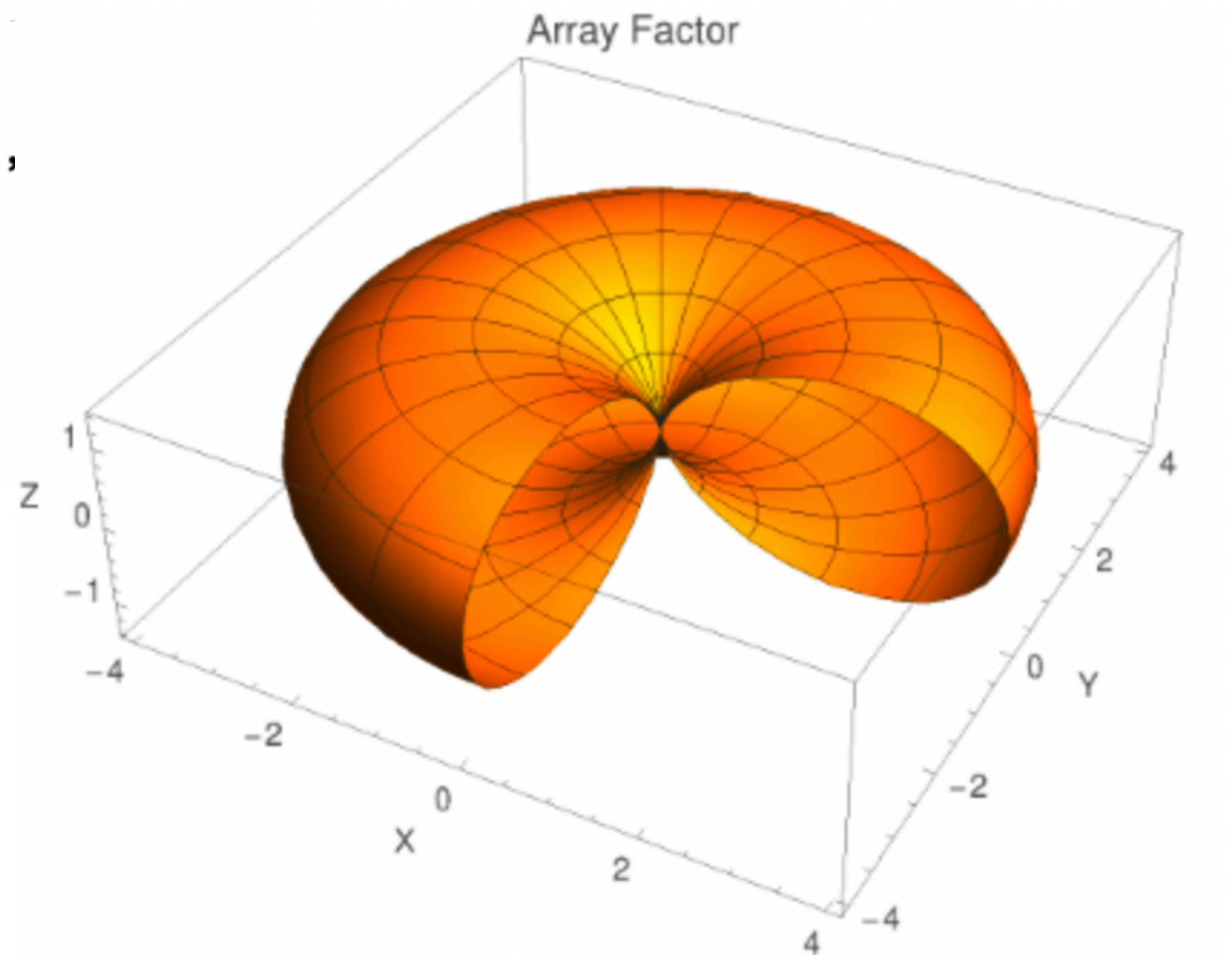
$$\text{SphericalPlot3D}[\text{Abs}[af], \{\theta, 0, \pi\}, \{\phi, 0, \frac{3\pi}{2}\},$$

PlotRange → All,

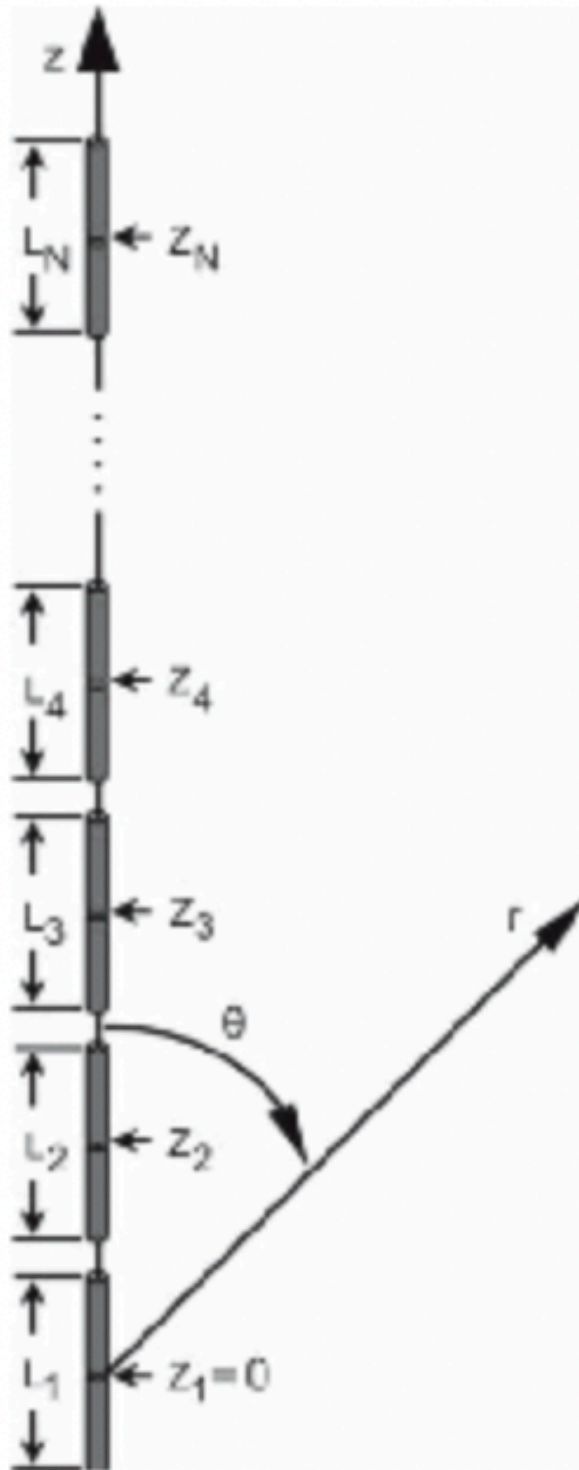
PlotLabel → "Array Factor",

AxesLabel → {"X", "Y", "Z"},

ImageSize → Medium]



Collinear Dipole Array



Dipoles with a common axis is known form a collinear array

The dipoles can also be placed side by side in which case, by changing the spacing or phasing between dipoles, can be configured to create either a broadside or end fire radiation pattern.

```
Clear[d, α, ψ, θ, φ, λ, n, af, naf, dipole]
```

$$\psi = \frac{2\pi}{\lambda} d \cos[\theta] + \alpha;$$

$$af := \text{Abs}\left[\frac{\sin\left[\frac{n\psi}{2}\right]}{\sin\left[\frac{\psi}{2}\right]}\right];$$

Basic Equations

$$naf = \frac{af}{n};$$

$$dipole := \text{Abs}[\sin[\theta]];$$

$$\lambda = 1; \quad \alpha = 0; \quad n = 4;$$

$$d = 0.5;$$

The array factor (af) is the overall radiation pattern of a group of antennas.

The most common antenna element is the dipole.

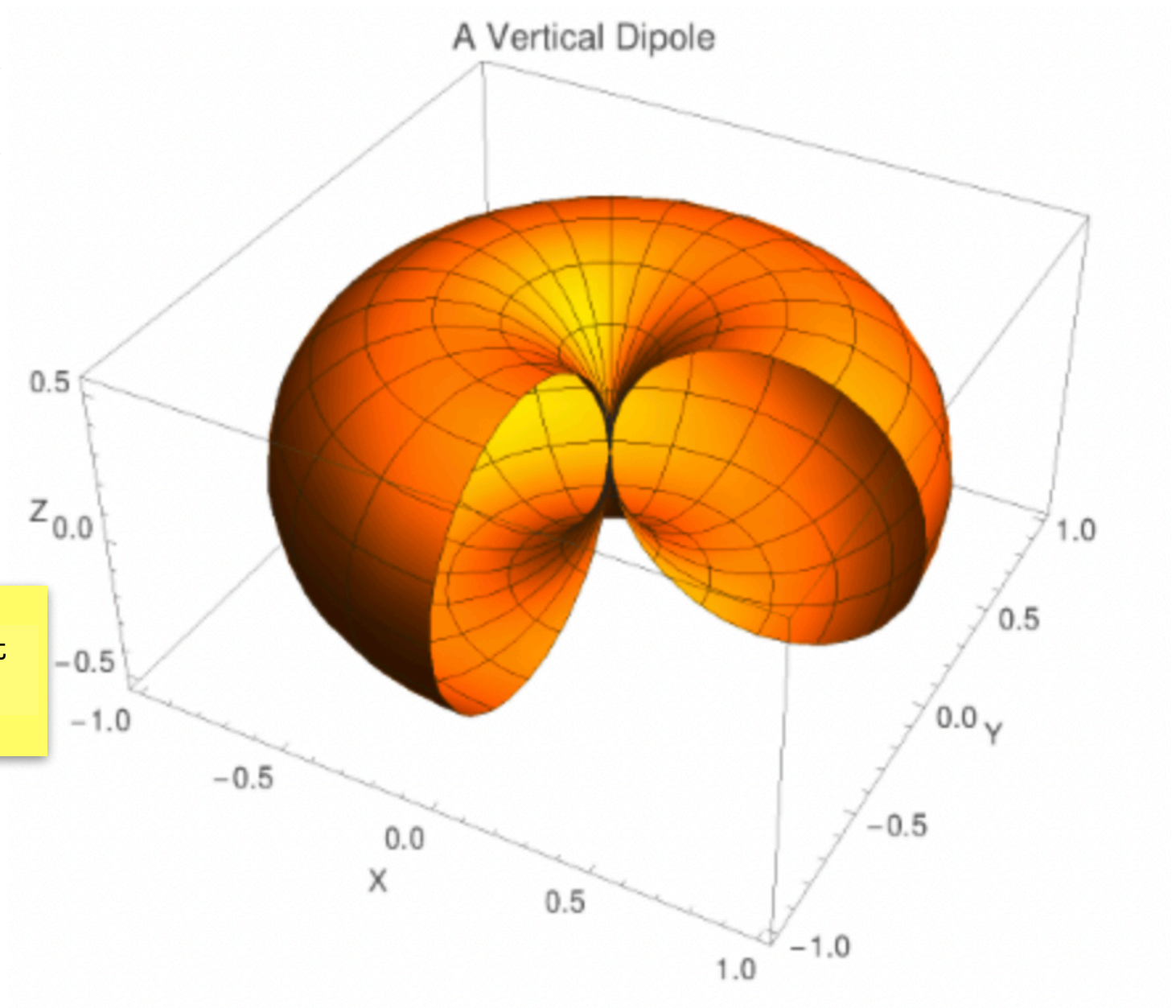
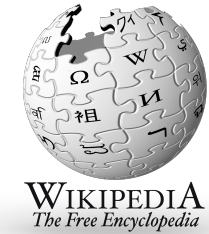
These dipoles are most often arranged in 1 or 2 dimensional arrays.

A Single Dipole

```
Clear[d,  $\alpha$ ,  $\psi$ ,  $\theta$ ,  $\phi$ ,  $\lambda$ , n, af, naf, dipole]
```

```
dipole := Abs[Sin[ $\theta$ ]];
```

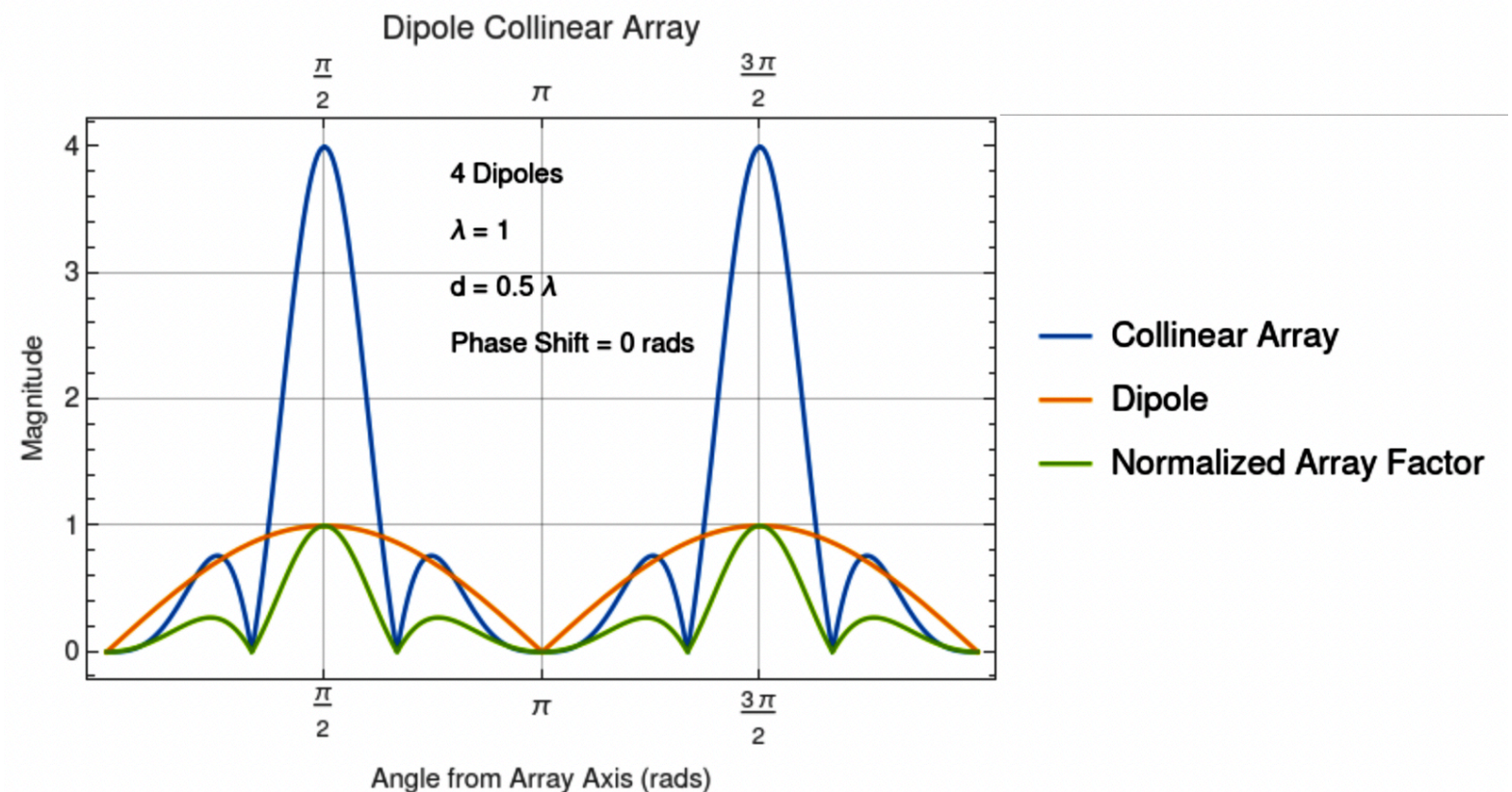
```
SphericalPlot3D[dipole, { $\theta$ , 0,  $\pi$ }, { $\phi$ , 0,  $1.5 \pi$ },  
PlotRange  $\rightarrow$  All,  
AxesLabel  $\rightarrow$  {"X", "Y", "Z"},  
PlotLabel  $\rightarrow$  "A Vertical Dipole"]
```



The dipole radiation pattern resembles a donut with the antenna passing through the hole.

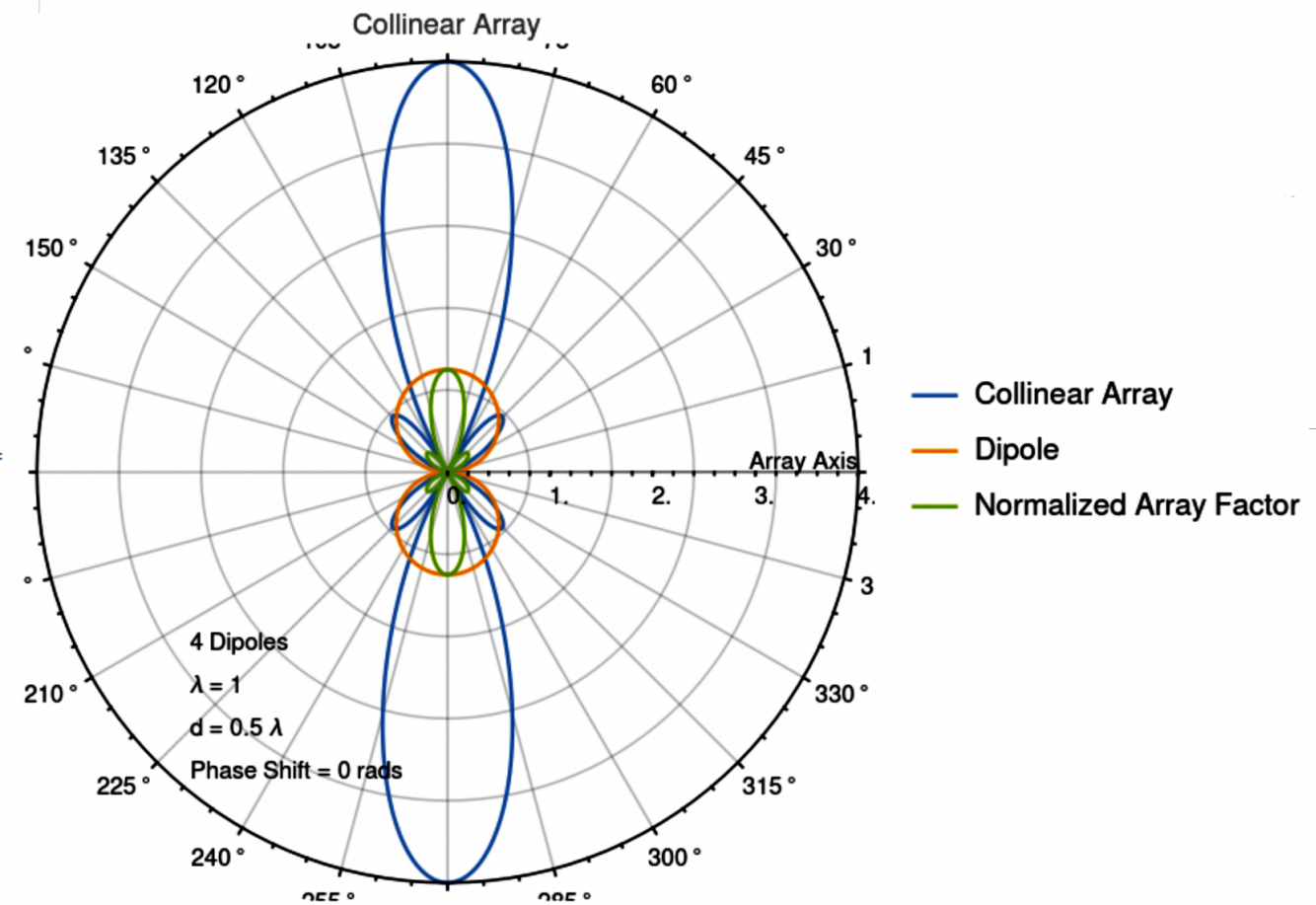
Linear Plot of a Collinear Dipole Array

```
Plot[{af dipole, dipole, naf}, { $\theta$ , 0, 2  $\pi$ },
  PlotRange → All,
  Frame → True,
  PlotLabel → "Dipole Collinear Array",
  FrameLabel → {"Angle from Array Axis (rads)", "Magnitude"},
  FrameTicks → {{ $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ }, Automatic},
  GridLines → {{ $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ }, Automatic},
  PlotLegends → {"Collinear Array", "Dipole", "Normalized Array Factor"},
  Epilog → {
    Text[Row[{n, " Dipoles"}], Scaled[{0.4, 0.9}], Left],
    Text[Row[{" $\lambda$  = ",  $\lambda$ }], Scaled[{0.4, 0.8}], Left],
    Text[Row[{"d = ", d, "  $\lambda$ "}], Scaled[{0.4, 0.7}], Left],
    Text[Row[{"Phase Shift = ",  $\alpha$ , " rads"}], Scaled[{0.4, 0.6}], Left]
  }]
```



Polar Plot of a Collinear Dipole Array

```
PolarPlot[{af dipole, dipole, naf}, { $\theta$ , 0, 2  $\pi$ },
  PlotRange → All,
  PolarAxes → True,
  PolarGridLines → Automatic,
  PolarAxesOrigin → {0, n},
  PolarTicks → {"Degrees", Automatic},
  PlotLabel → "Collinear Array",
  PlotLegends → {"Collinear Array", "Dipole", "Normalized Array Factor"},
  Epilog → {Text["Array Axis", {n, 0.1}, Right],
    Text[Row[{n, " Dipoles"}], Scaled[{0.2, 0.3}], Left],
    Text[Row[{" $\lambda$  = ",  $\lambda$ }], Scaled[{0.2, 0.25}], Left],
    Text[Row[{"d = ", d, "  $\lambda$ "}, Scaled[{0.2, 0.2}], Left],
    Text[Row[{"Phase Shift = ",  $\alpha$ , " rads"}], Scaled[{0.2, 0.15}], Left]
  ]]
```



3D Plot of a Collinear Dipole Array

```
Clear[d,  $\alpha$ ,  $\psi$ ,  $\theta$ ,  $\phi$ ,  $\lambda$ , n, af, naf, dipole]
```

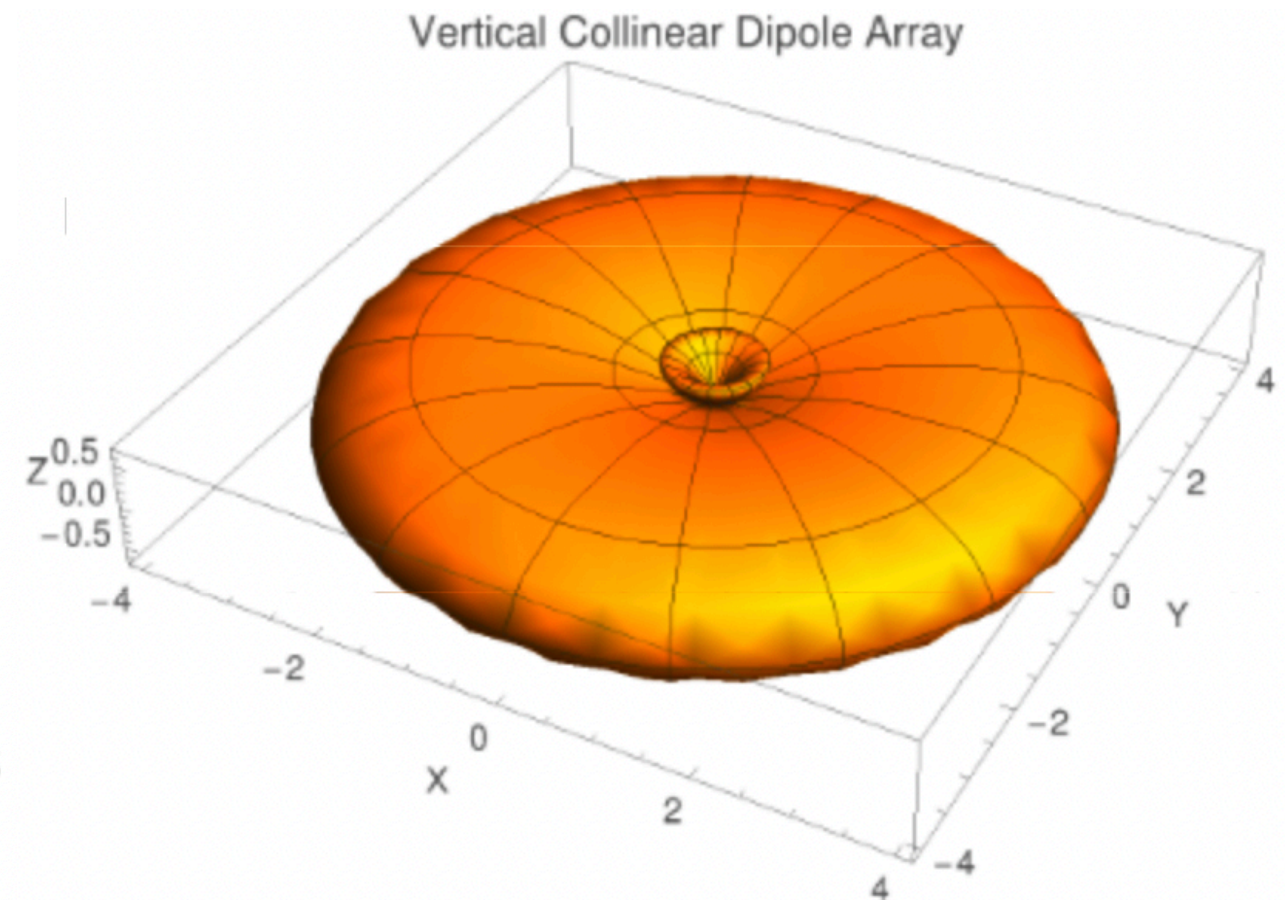
```
 $\lambda = 1$ ;  $\alpha = 0$ ; n = 4; d = 0.5;
```

```
dipole := Abs[Sin[ $\theta$ ]];
```

```
af := Abs[ $\frac{\text{Sin}[\frac{n\psi}{2}]}{\text{Sin}[\frac{\psi}{2}]}$ ];  $\psi := \frac{2\pi}{\lambda} d \text{Cos}[\theta] + \alpha$ ;
```

```
naf =  $\frac{af}{n}$ ;
```

```
SphericalPlot3D[af dipole, { $\theta$ , 0,  $\pi$ }, { $\phi$ , 0,  $2\pi$ },  
PlotRange  $\rightarrow$  All,  
AxesLabel  $\rightarrow$  {"X", "Y", "Z"},  
PlotLabel  $\rightarrow$  "Vertical Collinear Dipole Array"]
```



Manipulate 3D Plot

```
Clear[dipole,  $\phi$ ,  $\lambda$ , d,  $\alpha$ ,  $\psi$ , af, naf]
```

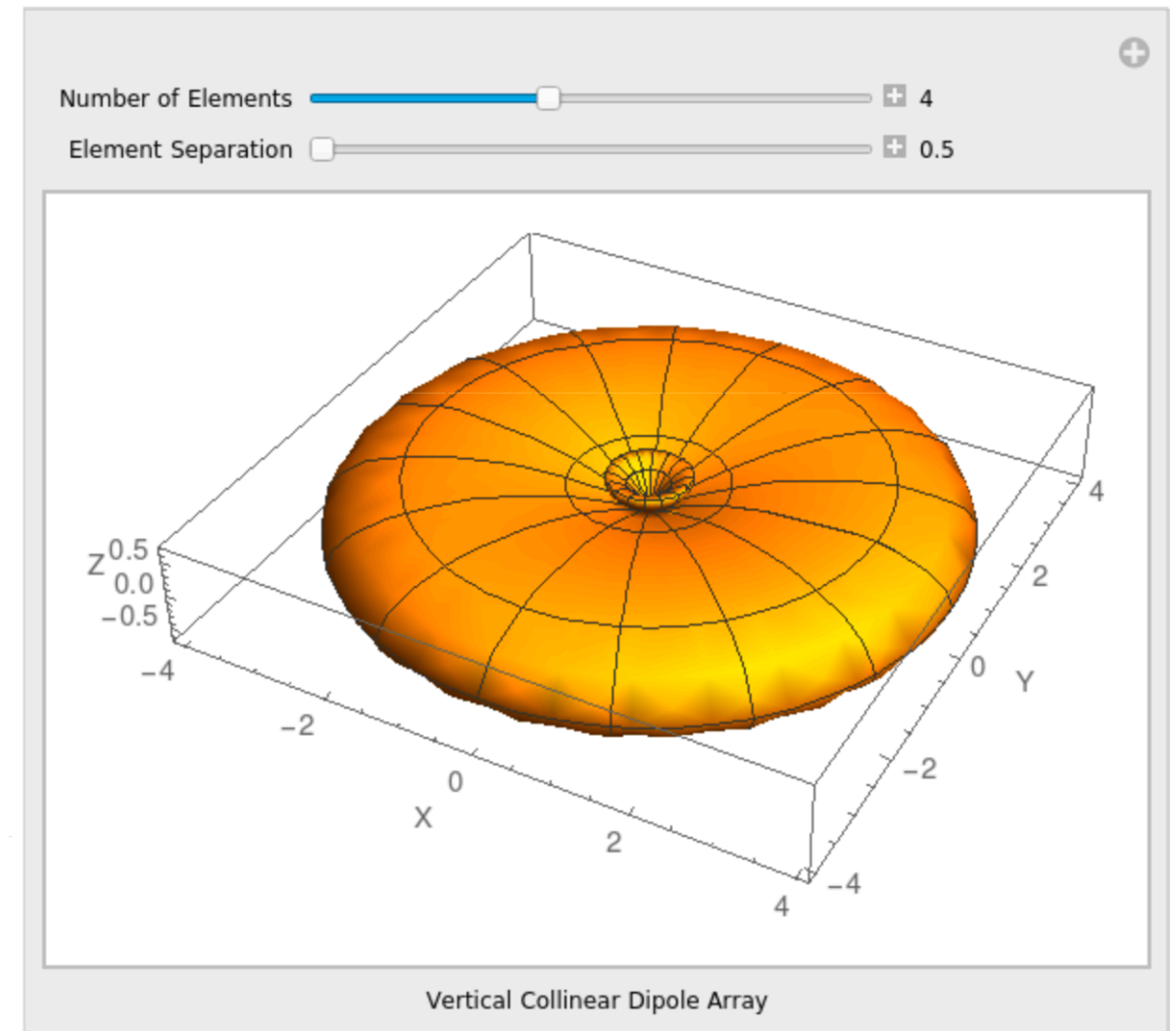
```
Manipulate[
  dipole := Abs[Sin[ $\theta$ ]];
   $\psi = \frac{2\pi}{\lambda} d \text{ Cos}[\theta] + \alpha$ ;
```

```
af := Abs[ $\frac{\text{Sin}[\frac{n\psi}{2}]}{\text{Sin}[\frac{\psi}{2}]}$ ];
```

```
 $\lambda = 1$ ;  $\alpha = 0$ ;
```

```
SphericalPlot3D[af dipole, { $\theta$ , 0,  $\pi$ }, { $\phi$ , 0,  $2\pi$ },
  PlotRange → All,
  AxesLabel → {"X", "Y", "Z"}],
```

```
{{n, 4, "Number of Elements"}, 1, 8, 1, Appearance → "Labeled"},
{{d, 0.5, "Element Separation"}, 0.5, 2, 0.25, Appearance → "Labeled"},
FrameLabel → "Vertical Collinear Dipole Array"]
```

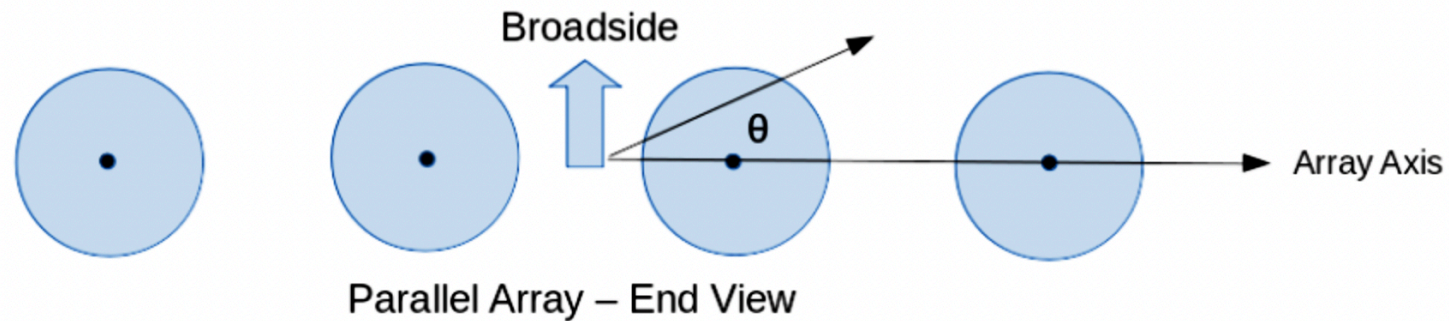


Broadside Array

All elements are in phase to create a broadside array.

Collinear dipole arrays are always broadside because there is no radiation from the end of the dipole.

However, by changing the phase between elements, the broadside angle can be altered.



Basic Equations

```
Clear[af, ne, k, psi, d, theta, phi, lambda]
```

```
af = Abs[Sum[e^(i (n-1) (k d Cos[theta] + psi)), {n, 1, ne}]]; (* array factor *)
```

```
k = 2 pi / lambda;
```

```
lambda = 1; psi = 0; d = 0.5; ne = 4;
```

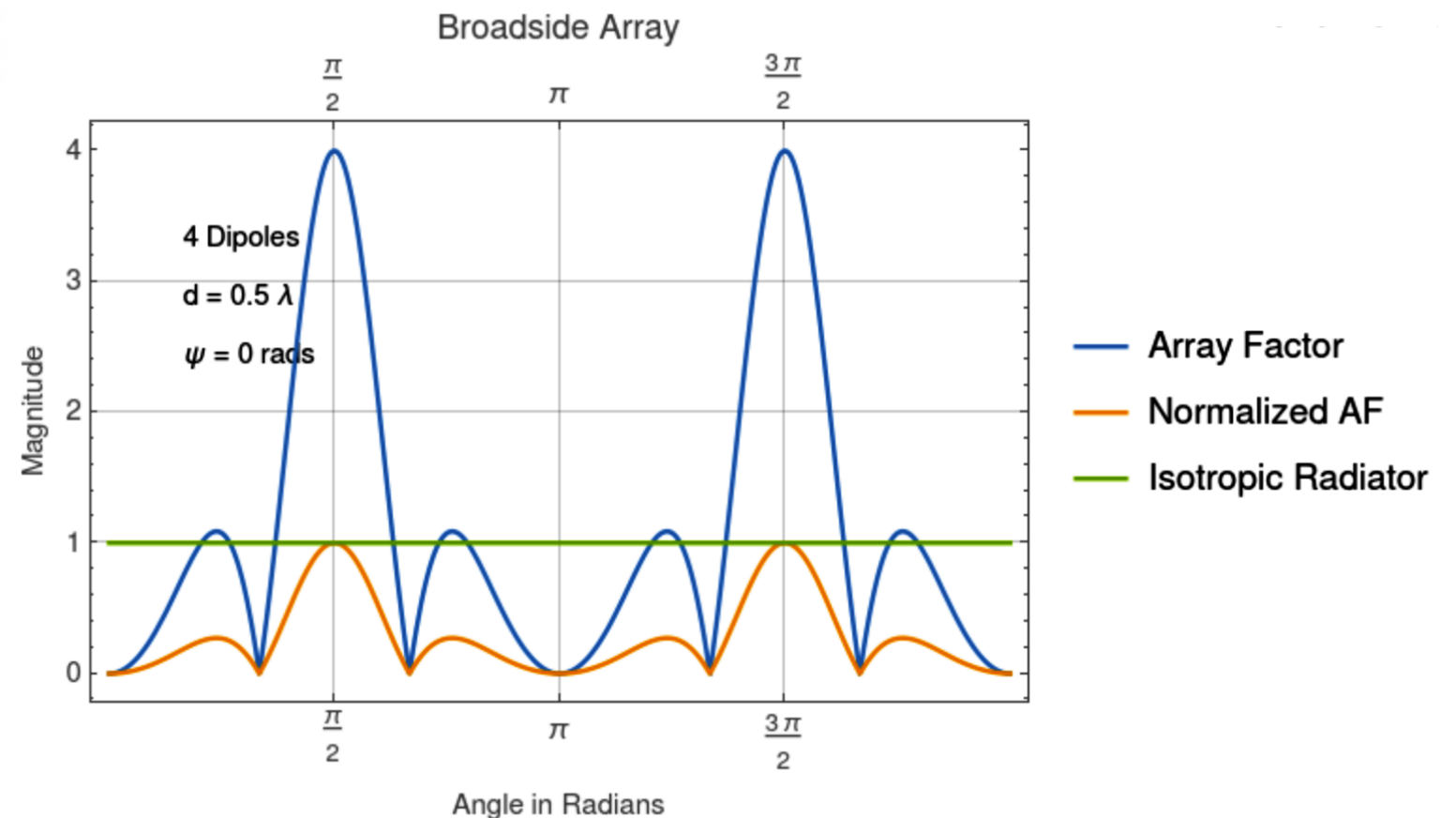
```
naf = af / ne; (* normalized array factor *)
```

```
dipole := Abs[Sin[theta]];
```

```
broadside = 1; (* from the side view,  
each element acts as an isotropic radiator *)
```


Broadside Array Linear Plot

```
Plot[{af, naf, broadside}, { $\theta$ , 0, 2  $\pi$ },
  PlotLabel → "Broadside Array",
  PlotRange → All,
  Frame → True, FrameLabel → {"Angle in Radians", "Magnitude"},
  FrameTicks → {{ $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ }, Automatic},
  GridLines → {{ $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ }, Automatic},
  PlotLegends → {"Array Factor", "Normalized AF", "Isotropic Radiator"},
  Prolog → {
    Text[Row[{n, " Dipoles"}], Scaled[{0.1, 0.8}], Left],
    Text[Row[{"d = ", d, "  $\lambda$ "}], Scaled[{0.1, 0.7}], Left],
    Text[Row[{" $\psi$  = ",  $\psi$ , " rads"}], Scaled[{0.1, 0.6}], Left]}
]
```

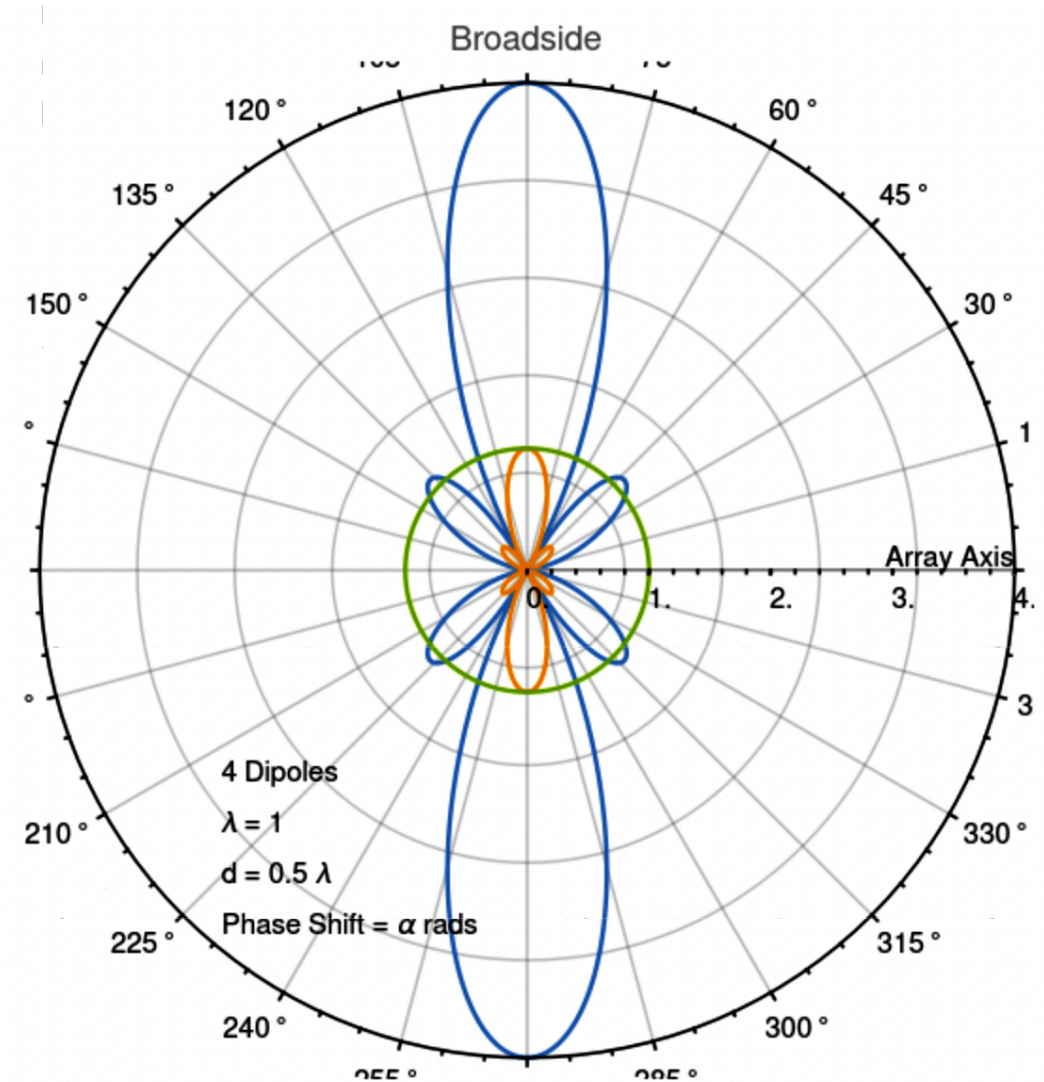


Broadside Array Polar Plot

```

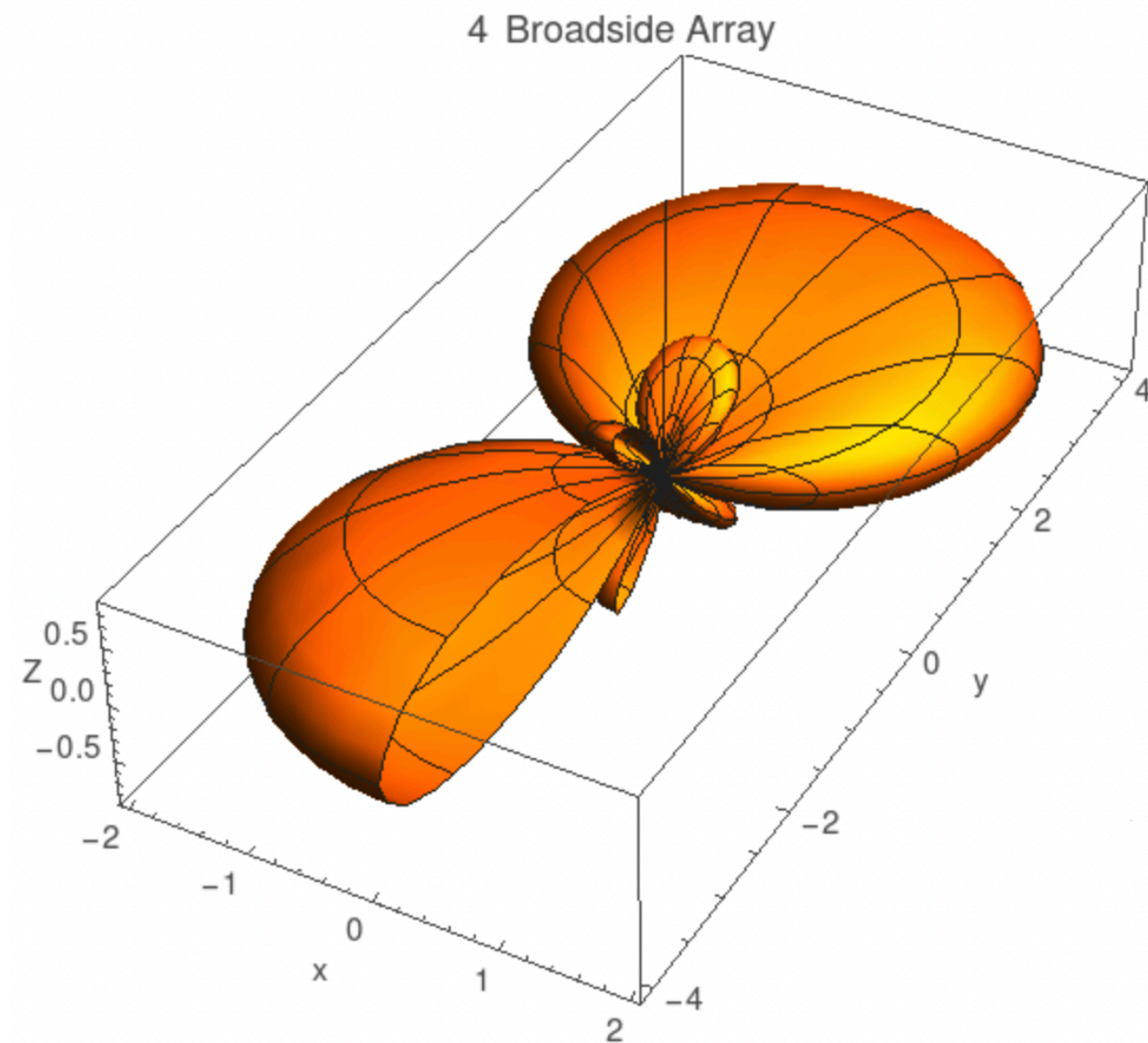
PolarPlot[{af, naf, broadside}, { $\theta$ , 0, 2  $\pi$ },
  PlotRange  $\rightarrow$  All,
  PolarAxes  $\rightarrow$  True,
  PolarGridLines  $\rightarrow$  Automatic,
  PolarAxesOrigin  $\rightarrow$  {0, n},
  PolarTicks  $\rightarrow$  {"Degrees", Automatic},
  PlotLabel  $\rightarrow$  "Broadside",
  Epilog  $\rightarrow$  {Text["Array Axis", {n, 0.1}, Right],
    Text[Row[{n, " Dipoles"}], Scaled[{0.2, 0.3}], Left],
    Text[Row[{" $\lambda$  = ",  $\lambda$ }], Scaled[{0.2, 0.25}], Left],
    Text[Row[{"d = ", d, "  $\lambda$ "}, Scaled[{0.2, 0.2}], Left],
    Text[Row[{"Phase Shift = ",  $\alpha$ , " rads"}], Scaled[{0.2, 0.15}], Left]
  ]

```



Broadside Array 3D Plot

```
SphericalPlot3D[Abs[Sin[φ]], {θ, 0, π}, {φ, 0,  $\frac{3\pi}{2}$ },  
PlotRange → All,  
PlotLabel → "Broadside Array",  
AxesLabel → {"x", "y", "z"},  
ImageSize → Medium]
```



Manipulate Broadside Array Polar Plot

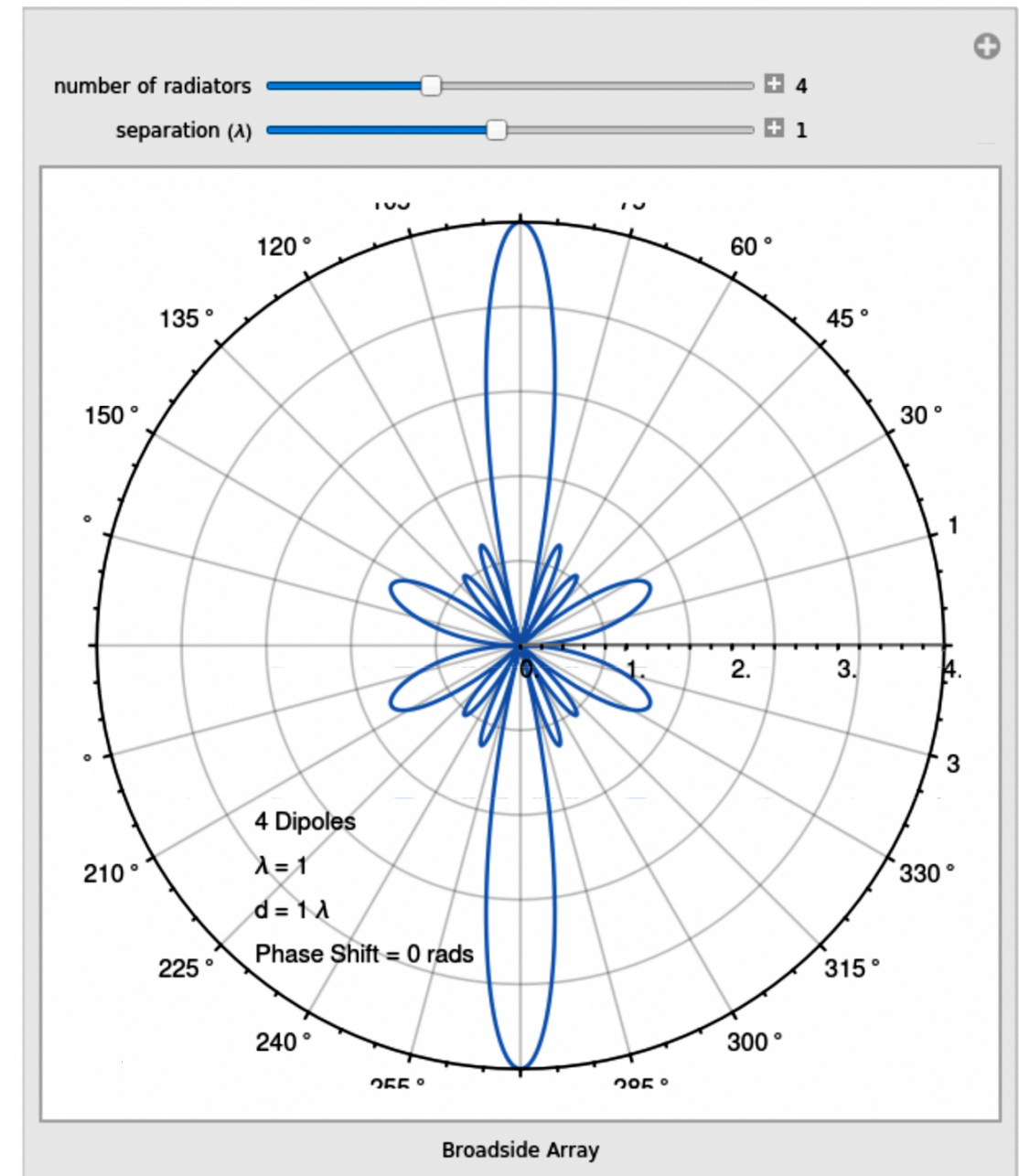
```

Clear[af, ne, k,  $\psi$ , d,  $\theta$ ,  $\phi$ ,  $\lambda$ ]
Manipulate[
  af = Abs[ $\sum_{n=1}^{ne} e^{i (n-1) (k d \cos[\theta] + \psi)}$ ]; (* array factor *)
  k =  $\frac{2 \pi}{\lambda}$ ;  $\lambda = 1$ ;  $\psi = 0$ ;
  naf =  $\frac{af}{ne}$ ; (* normalized array factor *)
  dipole := Abs[Sin[ $\theta$ ]];
  PolarPlot[af dipole, { $\theta$ , 0, 2  $\pi$ },
    PlotRange -> All,
    PolarAxes -> True,
    PolarGridLines -> Automatic,
    PolarAxesOrigin -> {0, ne},
    PolarTicks -> {"Degrees", Automatic},

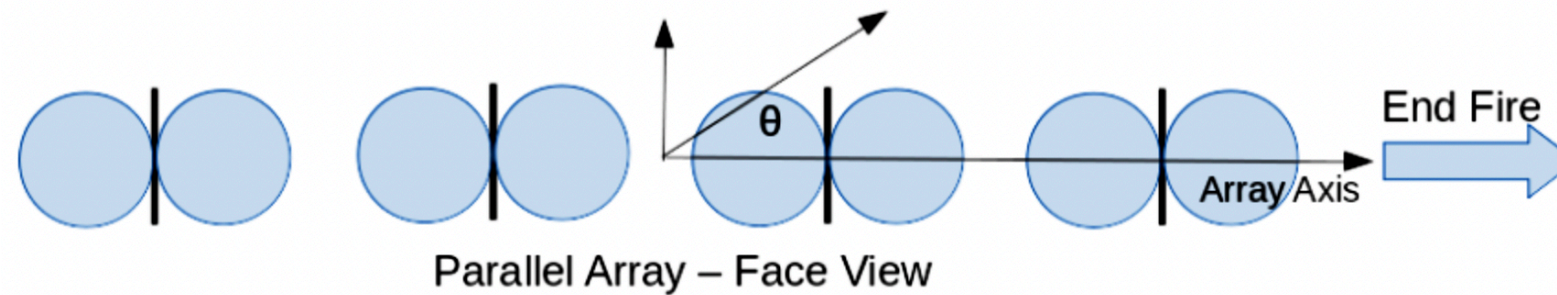
    Epilog -> {Text["Array Axis", {n, 0.1}, Right],
      Text[Row[{ne, " Dipoles"}], Scaled[{0.2, 0.3}], Left],
      Text[Row[{" $\lambda =$ ",  $\lambda$ }], Scaled[{0.2, 0.25}], Left],
      Text[Row[{"d = ", d, "  $\lambda$ "}], Scaled[{0.2, 0.2}], Left],
      Text[Row[{"Phase Shift = ",  $\alpha$ , " rads"}], Scaled[{0.2, 0.15}], Left]
    }],

  {{ne, 4, "number of radiators"}, 1, 10, 1, Appearance -> "Labeled"},
  {{d, 1, "separation ( $\lambda$ )"}, 0.1, 2, 0.1, Appearance -> "Labeled"},
  FrameLabel -> "Broadside Array"]

```



End Fire Array



```
Clear[af, ne, k, ψ, d, θ, λ]
λ = 1; ne = 4; α = 0; d = 0.25;
```

Some initial conditions

$$af = \text{Abs} \left[\sum_{n=1}^{ne} e^{i(n-1)(kd \cos[\theta] + \psi)} \right];$$

$$k = \frac{2\pi}{\lambda};$$

Basic Equations

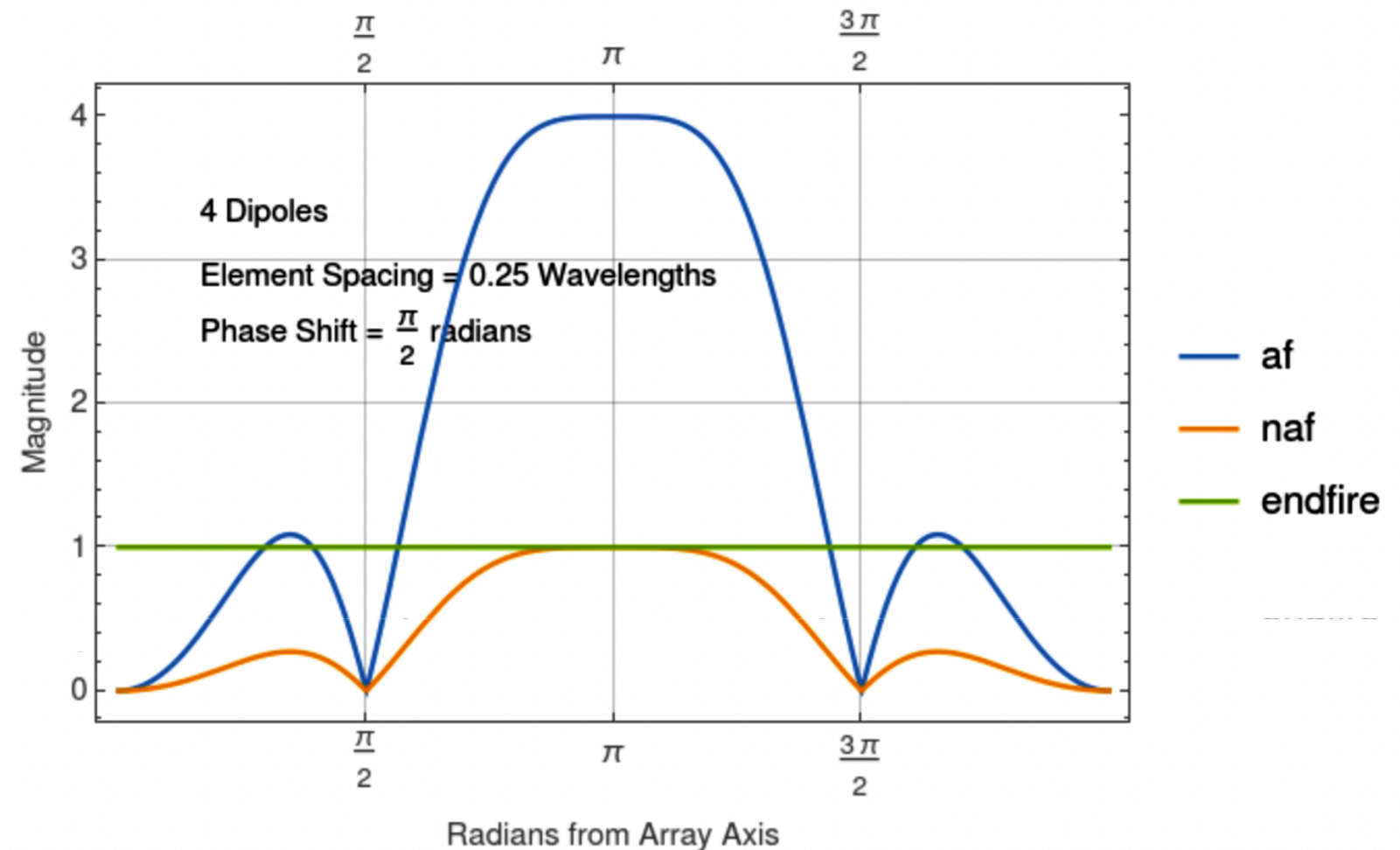
$$\psi = \frac{\pi}{2};$$

$$naf = \frac{af}{ne};$$

```
endfire = 1; (* from the side view it looks like an isotropic radiator *)
```


End Fire Array Linear Plot

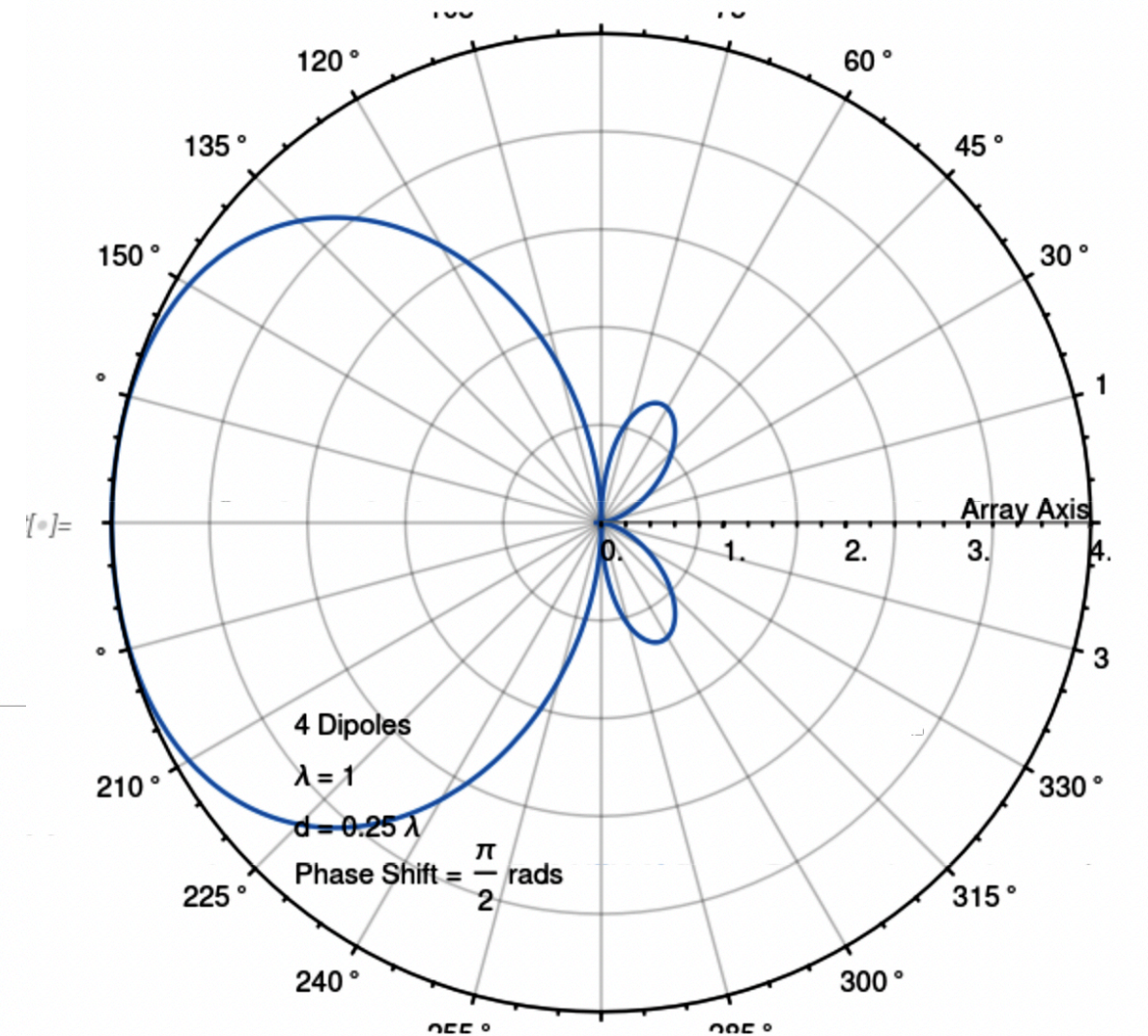
```
Plot[{af, naf, endfire}, { $\theta$ , 0, 2  $\pi$ },
  PlotRange → All,
  Frame → True, FrameLabel → {"Radians from Array Axis", "Magnitude"},
  FrameTicks → {{ $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ }, Automatic},
  GridLines → {{ $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ }, Automatic},
  PlotLegends → "Expressions",
  Prolog → {
    Text[Row[{n, " Dipoles"}], Scaled[{0.1, 0.8}], Left],
    Text[Row[{"Element Spacing = ", d, " Wavelengths"}], Scaled[{0.1, 0.7}], Left],
    Text[Row[{"Phase Shift = ",  $\psi$ , " radians"}], Scaled[{0.1, 0.6}], Left]
  }
]
```



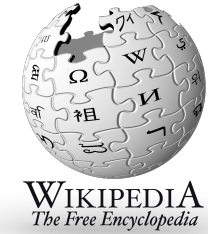
End Fire Array Polar Plot

```
PolarPlot[af, { $\theta$ , 0, 2  $\pi$ },
  PlotRange  $\rightarrow$  All,
  PolarAxes  $\rightarrow$  True,
  PolarGridLines  $\rightarrow$  Automatic,
  PolarAxesOrigin  $\rightarrow$  {0, ne},
  PolarTicks  $\rightarrow$  {"Degrees", Automatic},
```

```
Epilog  $\rightarrow$  {Text["Array Axis", {ne, 0.1}, Right],
  Text[Row[{n, " Dipoles"}], Scaled[{0.2, 0.3}], Left],
  Text[Row[{" $\lambda$  = ",  $\lambda$ }], Scaled[{0.2, 0.25}], Left],
  Text[Row[{"d = ", d, "  $\lambda$ "}], Scaled[{0.2, 0.2}], Left],
  Text[Row[{"Phase Shift = ",  $\psi$ , " rads"}], Scaled[{0.2, 0.15}], Left]
}]
```

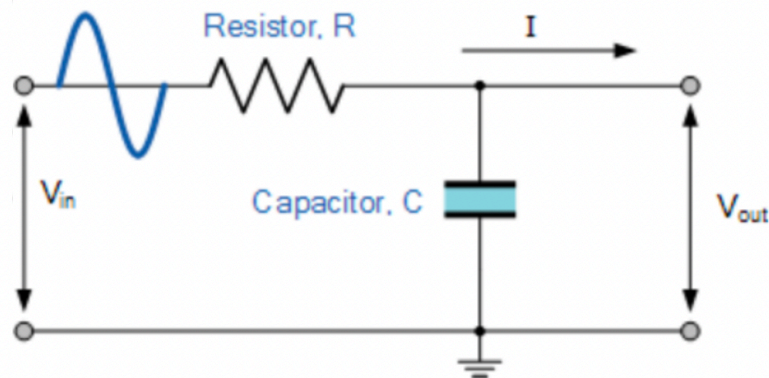


Bode Plots



RC Network

A 1st order low pass filter can be made from a simple a RC network

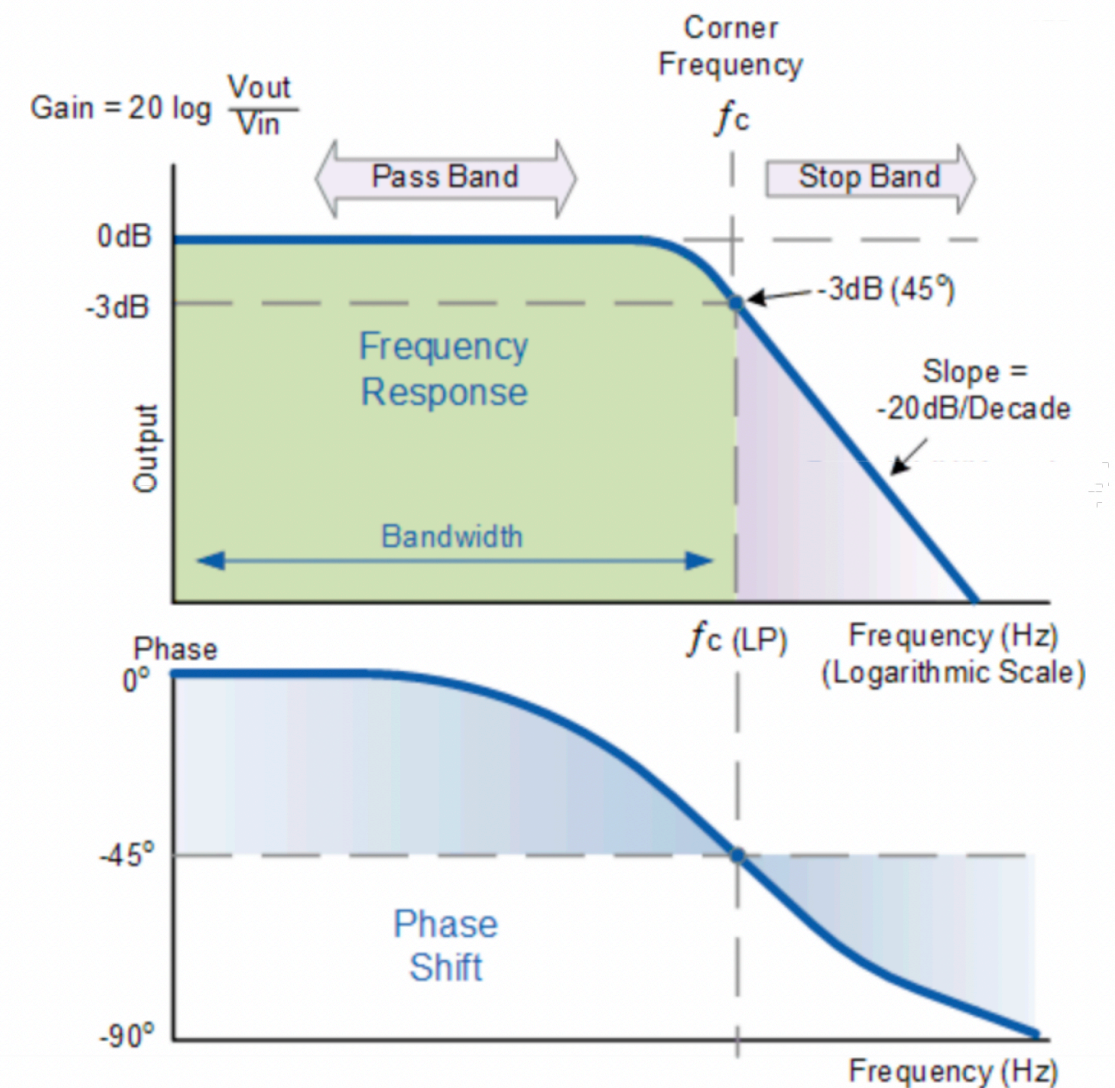


The transfer function (gain) is given by:

$$\frac{V_{out}}{V_{in}} = \frac{1}{RCs+1} \quad \text{where} \quad s = \omega j = 2\pi f j$$

A complex operator is required because the transfer function causes a change in both the magnitude and phase of the input signal

The output magnitude and signal phase shift resembles:



Courtesy: [Electronics Tutorials](#)

```
Clear[h1, dB3, r, c,  $\tau$ , s]
```

```
r = 1 * 106; c = 1 * 10-6;
```

```
 $\tau$  = r c;
```

```
dB3 =  $\frac{1}{\sqrt{2}}$ ; (* 3 dB down *)
```

```
h1 =  $\frac{1}{\tau s + 1}$ ; (* RC Circuit *)
```

It is always good practice to Clear the variables at the beginning

```
BodePlot[{h1, dB3, 1,  $\frac{1}{s}$ }, {0.01, 100},
```

```
PlotRange → Full,
```

```
PlotLayout → "HorizontalGrid",
```

```
GridLines → Automatic,
```

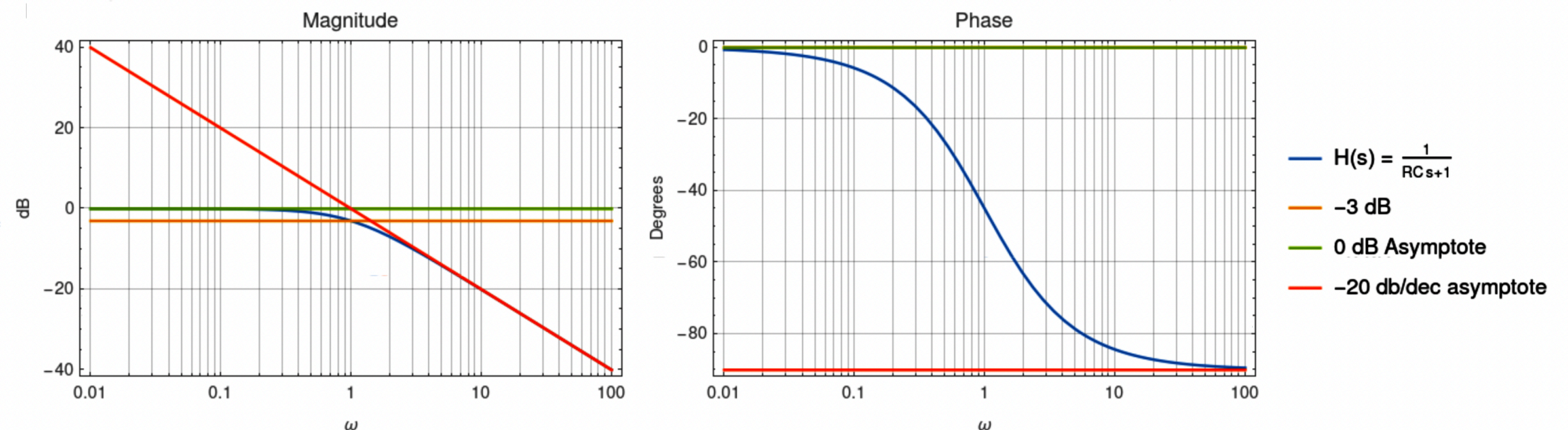
```
PlotLabel → {"Magnitude", "Phase"},
```

```
FrameLabel → {" $\omega$ ", "dB"}, {" $\omega$ ", "Degrees"}},
```

```
PlotLegends → {"H(s) =  $\frac{1}{RCs + 1}$ ", "-3 dB", "0 dB Asymptote",  
"-20 db/dec asymptote"},
```

```
ImageSize → Medium]
```

PlotLayout->Horizontal Grid places the plots side by side



BodePlot Function

Mathematica reserves the letter s for this function.

```
Clear[H, k, Q,  $\omega$ min,  $\omega$ max, s]
```

$$H = \frac{k}{s^2 + \frac{s}{Q} + 1};$$

```
k = 1; Q = 1;
```

```
 $\omega$ min = .01;
```

```
 $\omega$ max = 100;
```

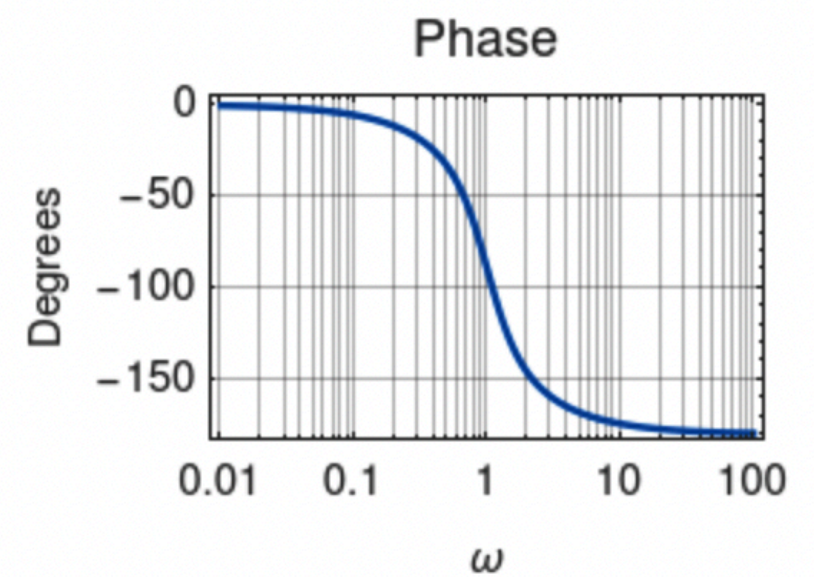
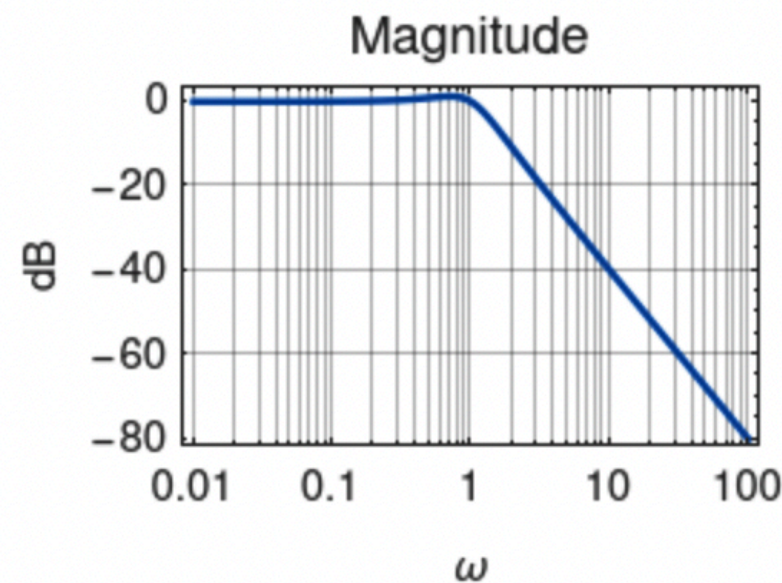
```
BodePlot[H, { $\omega$ min,  $\omega$ max},
```

```
GridLines → Automatic,
```

```
PlotLabel → {"Magnitude", "Phase"},
```

```
FrameLabel → {{ $\omega$ , "dB"}, { $\omega$ , "Degrees"}},
```

```
PlotLayout → "HorizontalGrid"]
```



Manipulating Bode Plot Variables

```
Clear[H, k, Q, ωmin, ωmax, s]
```

```
Manipulate[
```

$$H = \frac{k}{s^2 + \frac{s}{Q} + 1};$$

```
ωmin = .01;
```

```
ωmax = 100;
```

```
BodePlot[H, {ωmin, ωmax},
```

```
GridLines → Automatic,
```

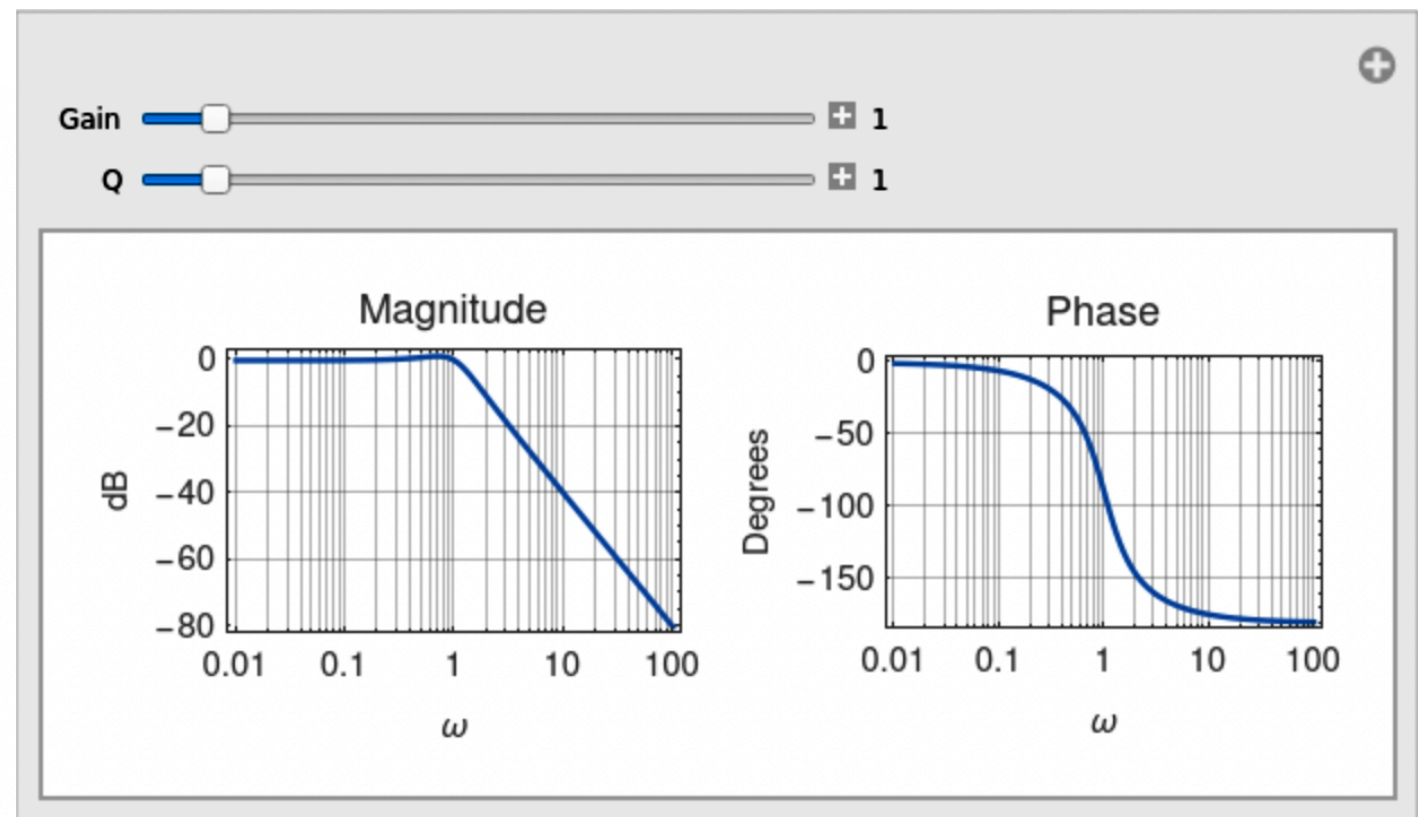
```
PlotLabel → {"Magnitude", "Phase"},
```

```
FrameLabel → {"ω", "dB"}, {"ω", "Degrees"}},
```

```
PlotLayout → "HorizontalGrid"],
```

```
{{k, 1, "Gain"}, 0.1, 10, Appearance → "Labeled"},
```

```
{{Q, 1, "Q"}, 0.1, 10, Appearance → "Labeled"}]
```



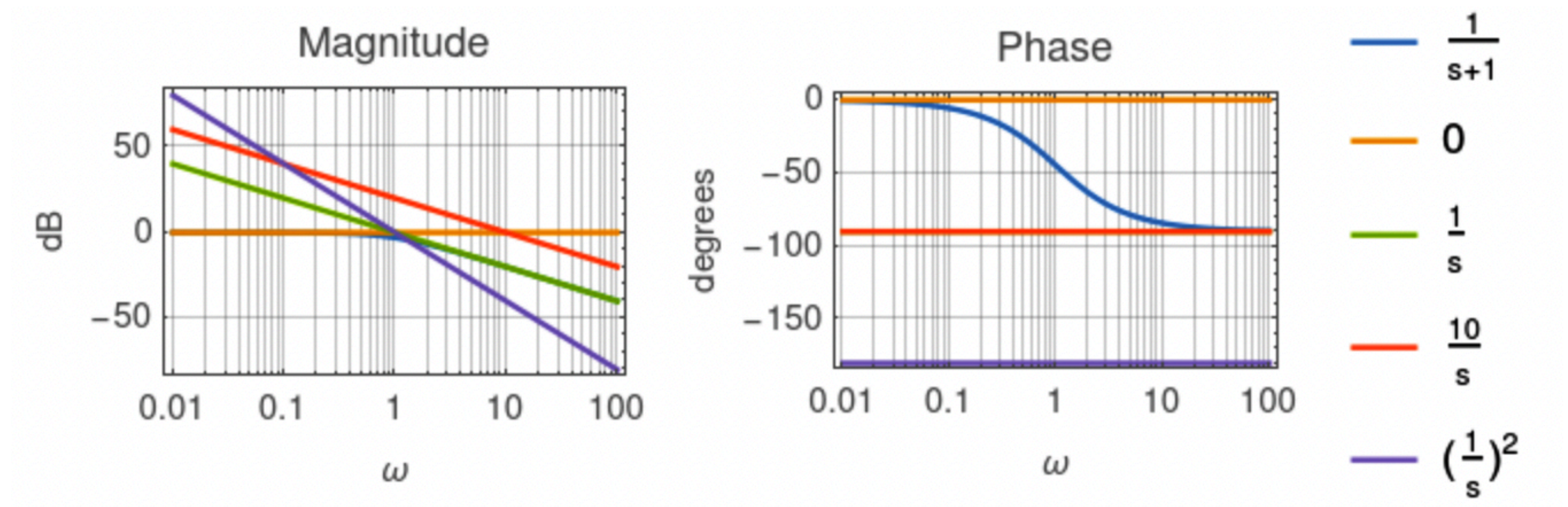
Plotting Asymptotes

```
Clear[a, b, c, d, e, s]
```

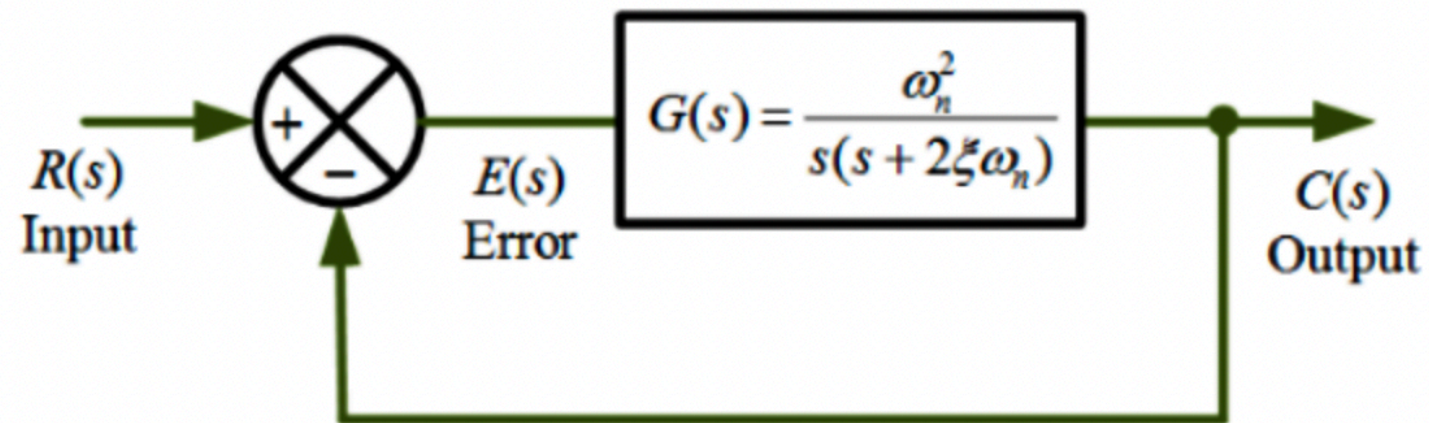
$$a = \frac{1}{s+1}; \quad b = 1; \quad c = \frac{1}{s}; \quad d = \frac{10}{s}; \quad e = \left(\frac{1}{s}\right)^2;$$

```
BodePlot[{a, b, c, d, e}, {.01, 100},
  GridLines -> Automatic,
  PlotLabel -> {Magnitude, Phase},
  FrameLabel -> {{ω, dB}, {ω, degrees}},
  PlotLegends -> {"1/(s+1)", "0", "1/s", "10/s", "(1/s)^2"},
  PlotLayout -> "HorizontalGrid"
]
```

Asymptotes are very useful when manually graphing the Bode response.



2nd Order Control System



www.electricalworkbook.com

Figure 1 Second order control system block diagram.

$$\frac{C(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \dots\dots (1)$$

Control System Equation

```
Clear[ $\omega n$ ,  $\xi$ , s, system]
```

```
Manipulate[
```

$$\text{system} = \frac{\omega n^2}{s^2 + 2 \xi \omega n s + \omega n^2};$$

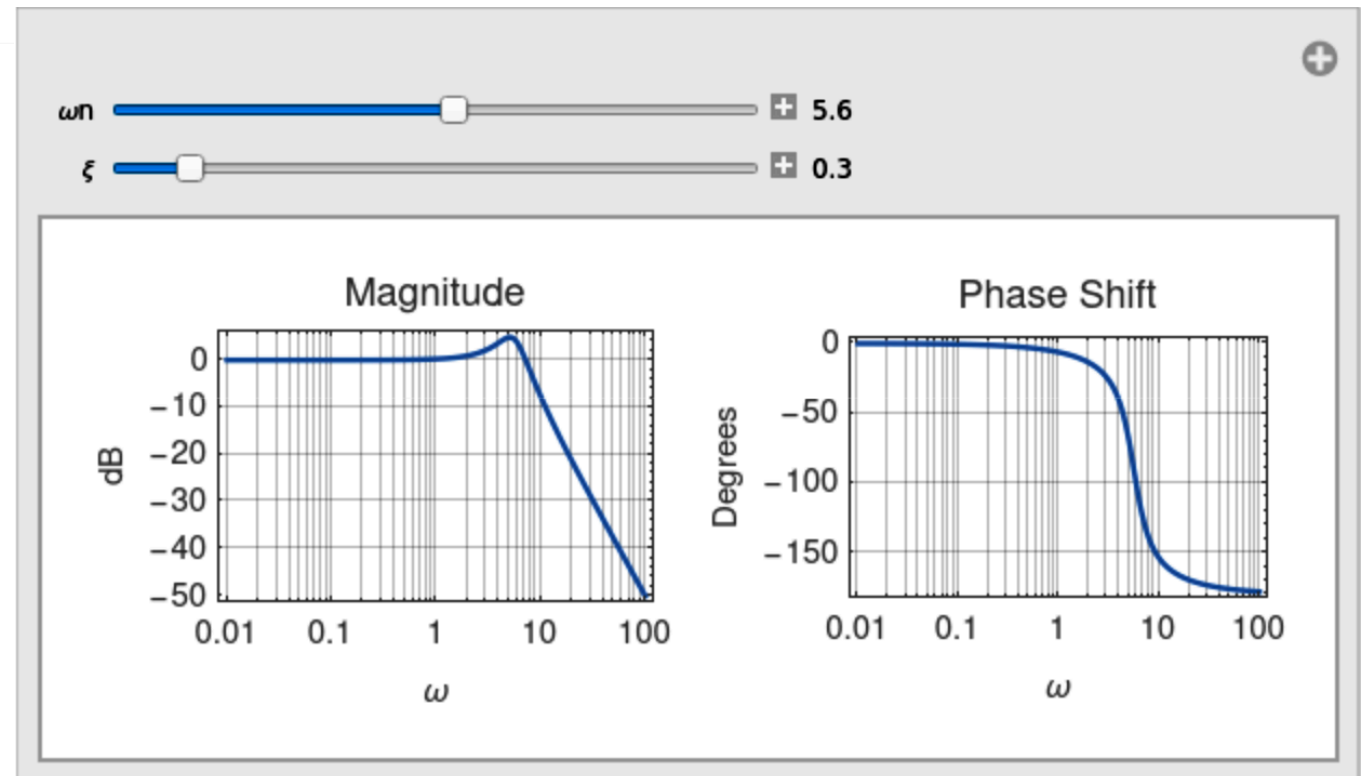
This is the 'standard form' of the 2nd order control system equation.

```
BodePlot[system, {0.01, 100},  
PlotLayout → "HorizontalGrid", GridLines → Automatic,  
PlotLabel → {"Magnitude", "Phase Shift"},  
FrameLabel → {{ $\omega$ , "dB"}, { $\omega$ , "Degrees"}}],
```

```
{{ $\omega n$ , 1, " $\omega n$ "}, 0.5, 10, 0.1, Appearance → "Labeled"},
```

```
{{ $\xi$ ,  $\frac{1}{\sqrt{2}}$ , " $\xi$ "}, 0.1, 2, 0.1, Appearance → "Labeled"}
```

```
]
```



Control System Impulse Response

```
Clear[ $\omega_n$ ,  $\xi$ , s, system]
```

```
 $\omega_n = 1$ ;  $\xi = \frac{1}{\sqrt{2}}$ ;
```

```
system =  $\frac{\omega_n^2}{s^2 + 2 \xi \omega_n s + \omega_n^2}$ ;
```

```
InverseLaplaceTransform[system, s, t]
```

```
Plot[%, {t, 0, 10}, PlotRange -> All, GridLines -> Automatic,
```

```
PlotLabel -> "System Impuse Response", Frame -> True,
```

```
FrameLabel -> {"time", "Magnitude"},
```

```
Prolog -> {
```

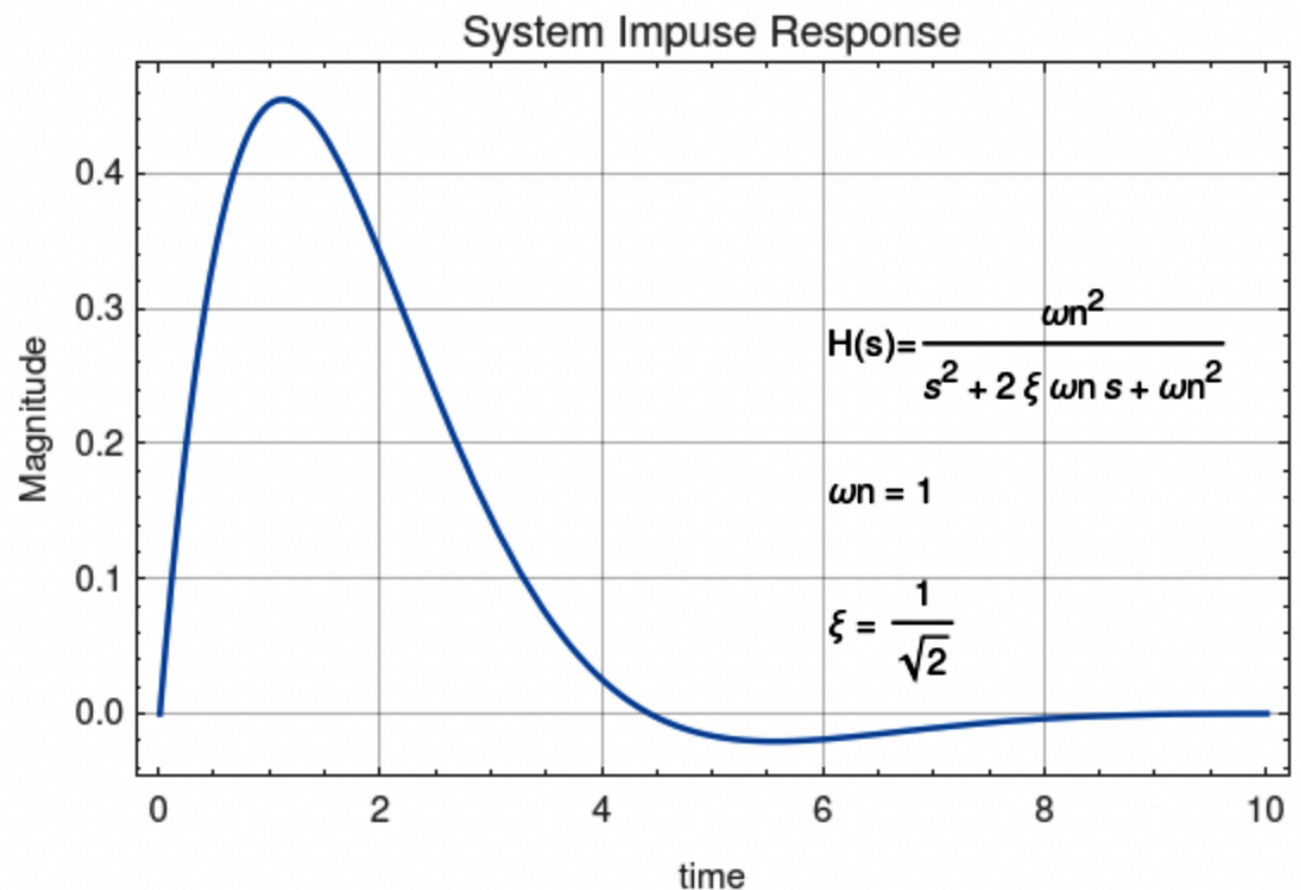
```
Text[Row[{"H(s) =  $\frac{\omega_n^2}{s^2 + 2 \xi \omega_n s + \omega_n^2}$ "}], Scaled[{0.6, 0.6}], Left],
```

```
Text[Row[{" $\omega_n =$ ",  $\omega_n$ }], Scaled[{0.6, 0.4}], Left],
```

```
Text[Row[{" $\xi =$ ",  $\xi$ }], Scaled[{0.6, 0.2}], Left]
```

```
}]
```

The control system impulse response can be determined by taking the Inverse Laplace transform of the system equation.



```
Clear[cf, s,  $\omega n$ , mag, phase,  $\zeta$ ]
```

```
Manipulate[
```

$$cf = \frac{(\omega n)^2}{s^2 + 2 \zeta \omega n s + (\omega n)^2};$$

```
s := j  $\omega$ ;
```

```
 $\omega$ min = 0.01;
```

```
 $\omega$ max = 100;
```

```
mag = 20 Log10[Abs[cf]];
```

```
LogLinearPlot[mag, { $\omega$ ,  $\omega$ min,  $\omega$ max},
```

```
GridLines → Automatic,
```

```
Frame → True,
```

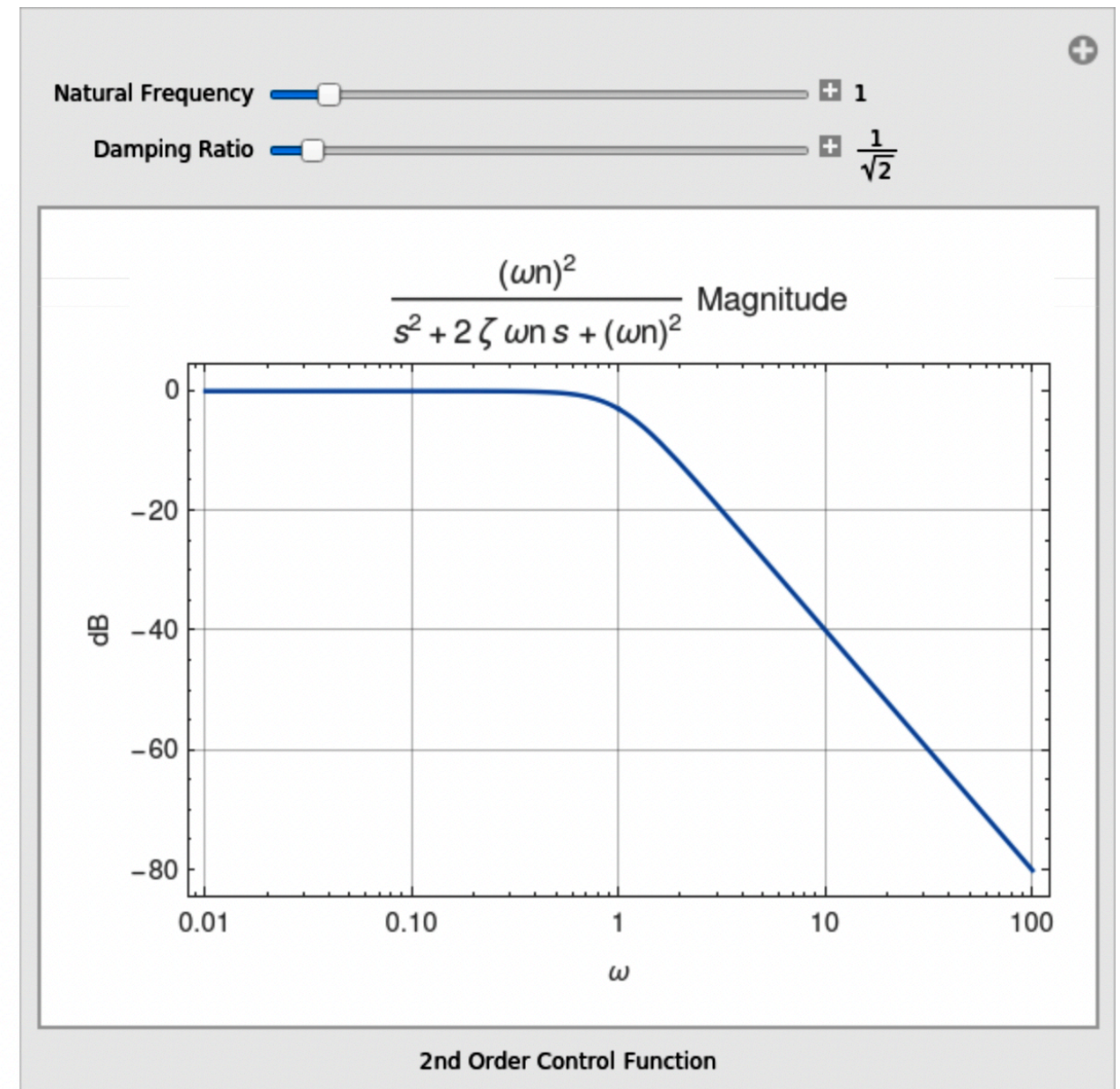
```
FrameLabel → {" $\omega$ ", "dB"},
```

```
PlotLabel → " $\frac{(\omega n)^2}{s^2 + 2 \zeta \omega n s + (\omega n)^2}$  Magnitude"],
```

```
{{ $\omega n$ , 1, "Natural Frequency"}, 0.1, 10, Appearance → "Labeled"},
```

```
{{ $\zeta$ ,  $\frac{1}{\sqrt{2}}$ , "Damping Ratio"}, 0.1, 10, Appearance → "Labeled"},
```

```
FrameLabel → "2nd Order Control Function"]
```

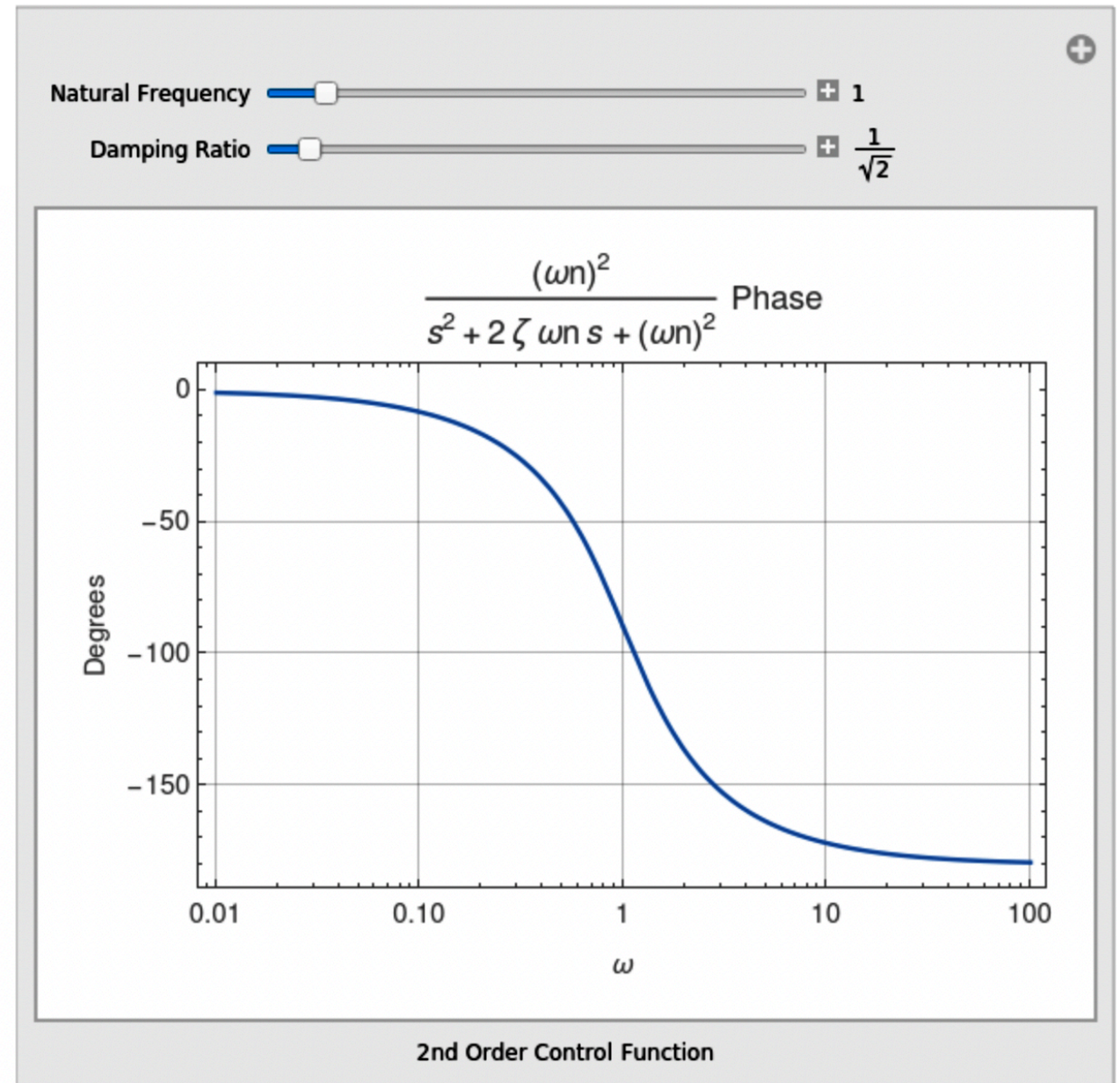


A Bode plot can be generated without using the BodePlot function


```

Manipulate[
  cf =  $\frac{(\omega n)^2}{s^2 + 2 \zeta \omega n s + (\omega n)^2}$ ;
  s = j  $\omega$ ;
   $\omega_{\min} = 0.01$ ;
   $\omega_{\max} = 100$ ;
  phase = Arg[cf]  $\frac{180}{\pi}$ ;
  LogLinearPlot[phase, { $\omega$ ,  $\omega_{\min}$ ,  $\omega_{\max}$ },
    GridLines → Automatic,
    Frame → True,
    FrameLabel → {" $\omega$ ", "Degrees"},
    PlotLabel → " $\frac{(\omega n)^2}{s^2 + 2 \zeta \omega n s + (\omega n)^2}$  Phase",
    {{ $\omega n$ , 1, "Natural Frequency"}, 0.1, 10, Appearance → "Labeled"},
    {{ $\zeta$ ,  $\frac{1}{\sqrt{2}}$ , "Damping Ratio"}, 0.1, 10, Appearance → "Labeled"},
    FrameLabel → "2nd Order Control Function"]

```



Electric Field Lines

The electric potential arising from a point charge, Q , at a distance, r , from the location of Q is observed to be

$$V_{\mathbf{E}} = \frac{1}{4\pi\epsilon_0} \frac{Q}{r},$$

where ϵ_0 is the [permittivity of vacuum](#)^[4], $V_{\mathbf{E}}$ is known as the **Coulomb potential**, and the ratio,

$$k_e = \frac{1}{4\pi\epsilon_0}$$

is known as the [Coulomb constant](#).

The electric potential at any location, $\mathbf{r}$, in a system of point charges is equal to the sum of the individual electric potentials due to every point charge in the system. This fact simplifies calculations significantly, because addition of potential (scalar) fields is much easier than addition of the electric (vector) fields. Specifically, the potential of a set of discrete point charges q_i at points $\mathbf{r}_i$ becomes

$$V_{\mathbf{E}}(\mathbf{r}) = k_e \sum_i \frac{q_i}{|\mathbf{r} - \mathbf{r}_i|},$$

where

- $\mathbf{r}$ is a point at which the potential is evaluated;
- $\mathbf{r}_i$ is a point at which there is a nonzero charge; and
- q_i is the charge at the point $\mathbf{r}_i$.

Courtesy: [Wikipedia](#)

3D Plot

```
Clear[ePot, xy, pos1, pos2, pos3, ePot1, ePot2, ePot3, q1, q2, q3, sum]
r[v_] :=  $\sqrt{v.v}$  (* vector magnitude -  $r = |r-r_i|$  *)
ePot[xy_, pos_, q_] :=  $\frac{q}{r[xy - pos]}$  (* electric field potential - Coloumb constant set to 1 *)
xy = {x, y}; (* location of measurement *)

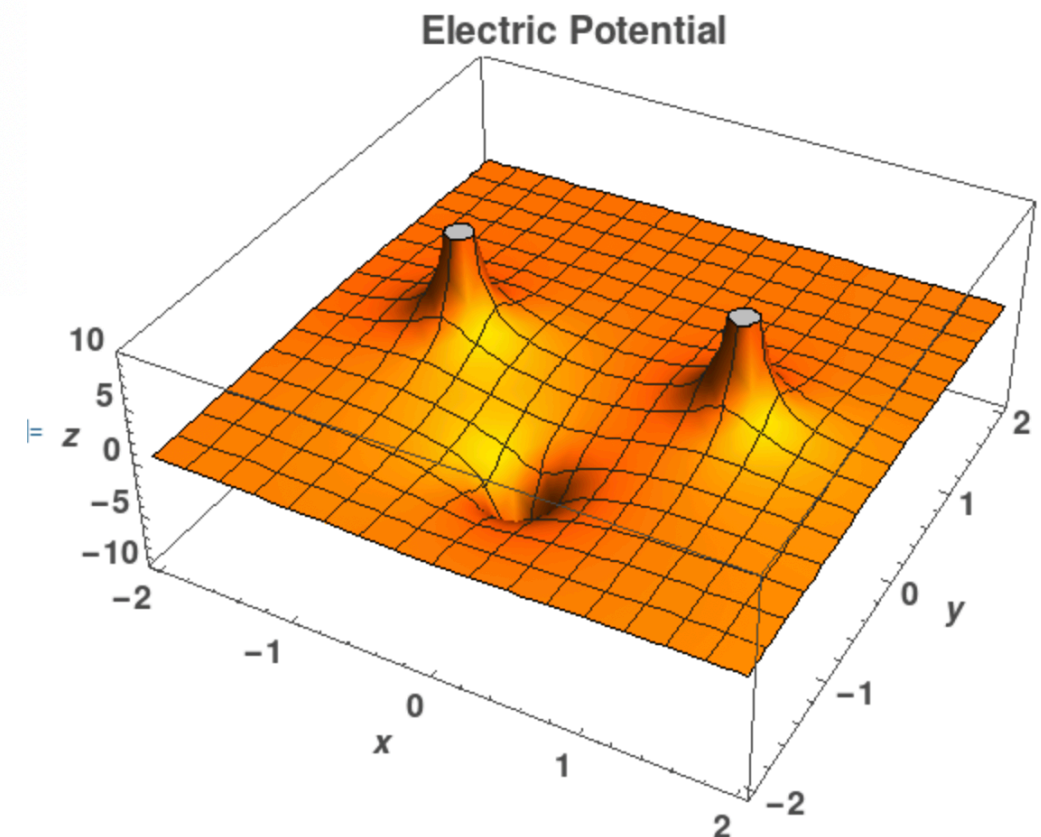
pos1 = {-1, 0}; (* location of 1st charge *)
q1 = 1; (* 1st charge *)
ePot1 = ePot[xy, pos1, q1]; (* electric potential due to 1st charge *)

pos2 = {1, 0}; q2 = 1; (* 2nd charge *)
ePot2 = ePot[xy, pos2, q2];

pos3 = {0, -1}; q3 = -1; (* 3rd charge *)
ePot3 = ePot[xy, pos3, q3];

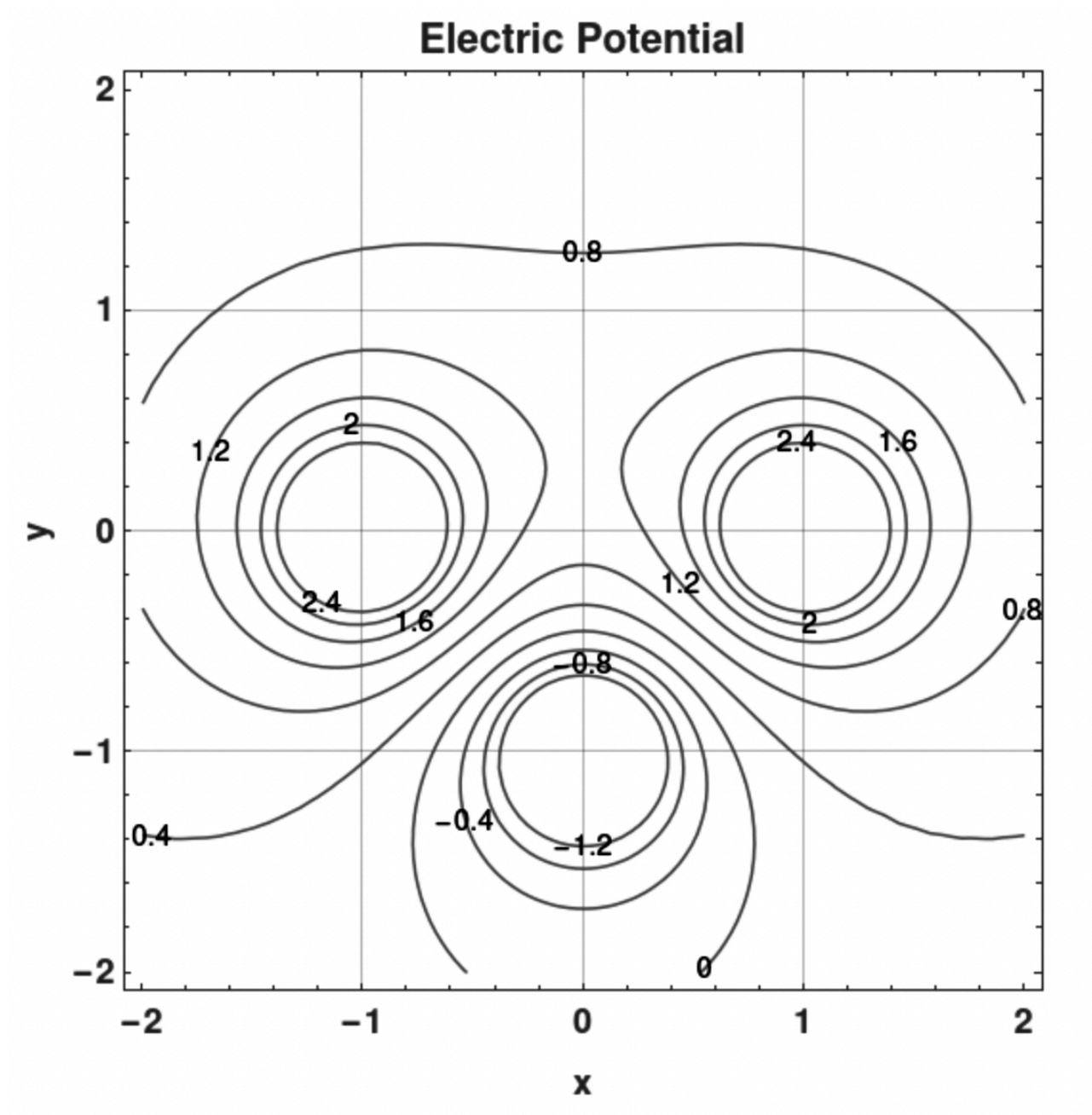
sum = ePot1 + ePot2 + ePot3; (* electric potential due to all charges *)
```

```
Plot3D[sum, {x, -2, 2}, {y, -2, 2},
  PlotRange → {All, All, {-10, 10}},
  PlotLabel → "Electric Potential",
  AxesLabel → {x, y, z},
  LabelStyle → {Bold, FontSize → 12}]
```



Equipotential Lines

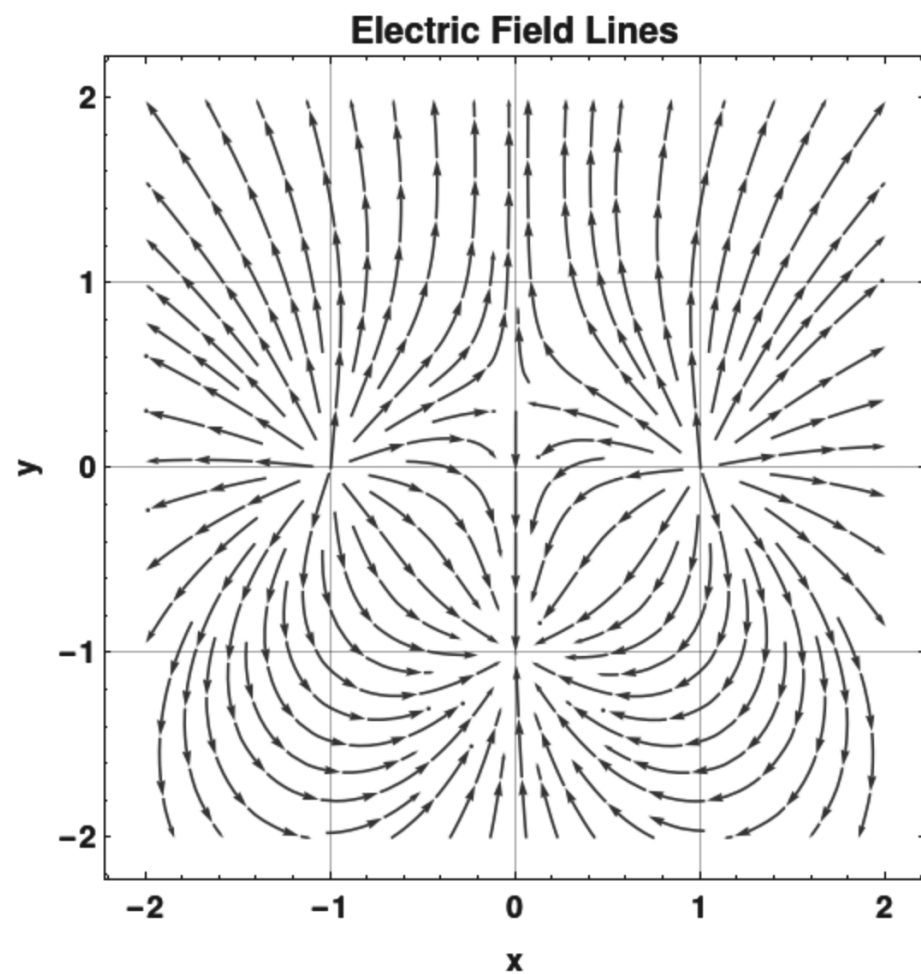
```
]:= epl = ContourPlot[sum, {x, -2, 2}, {y, -2, 2},  
  Contours → 10,  
  GridLines → Automatic,  
  ContourShading → False,  
  ContourLabels → All,  
  PlotLabel → "Electric Potential",  
  FrameLabel → {"x", "y"},  
  LabelStyle → {Bold, FontSize → 12}]
```



Vector Field

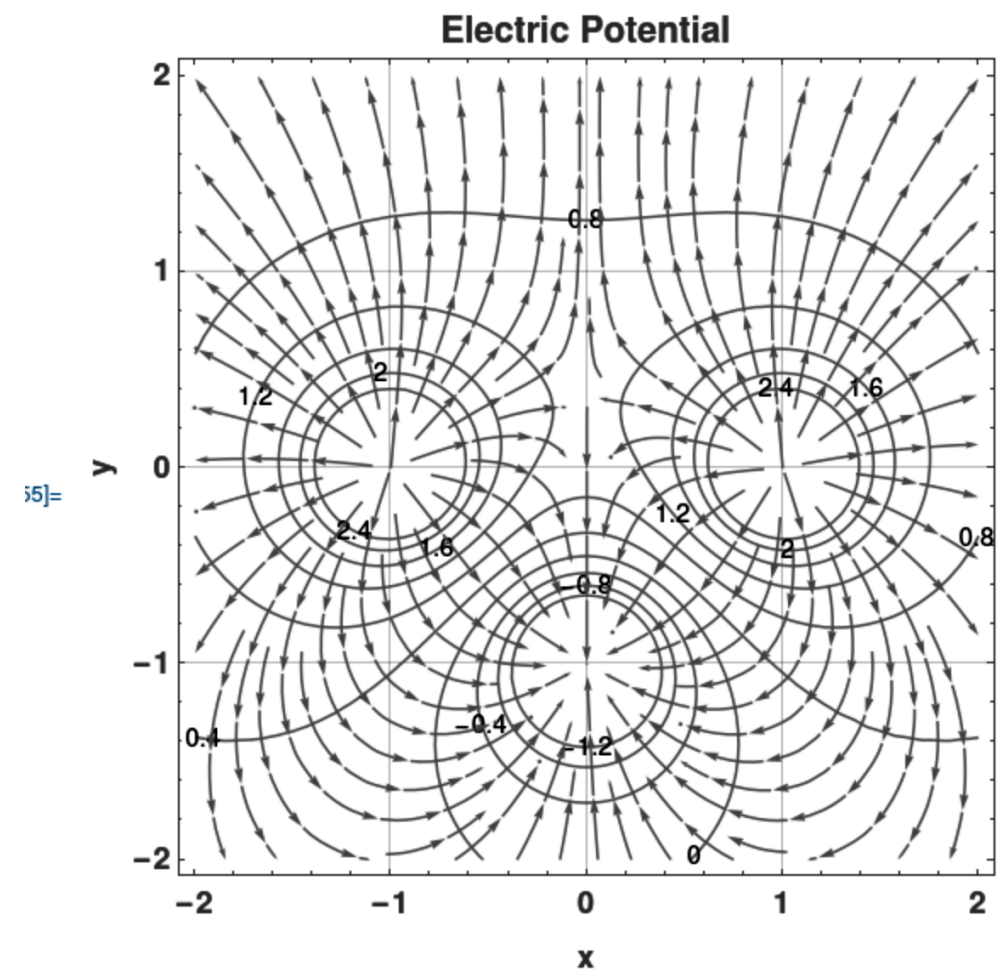
```
3]:= vf = -Grad[sum, {x, y}];
```

```
sp = StreamPlot[vf, {x, -2, 2}, {y, -2, 2},  
  GridLines → Automatic,  
  PlotLabel → "Electric Field Lines",  
  FrameLabel → {"x", "y"},  
  LabelStyle → {Bold, FontSize → 12},  
  StreamColorFunction → None,  
  StreamStyle → Gray]
```



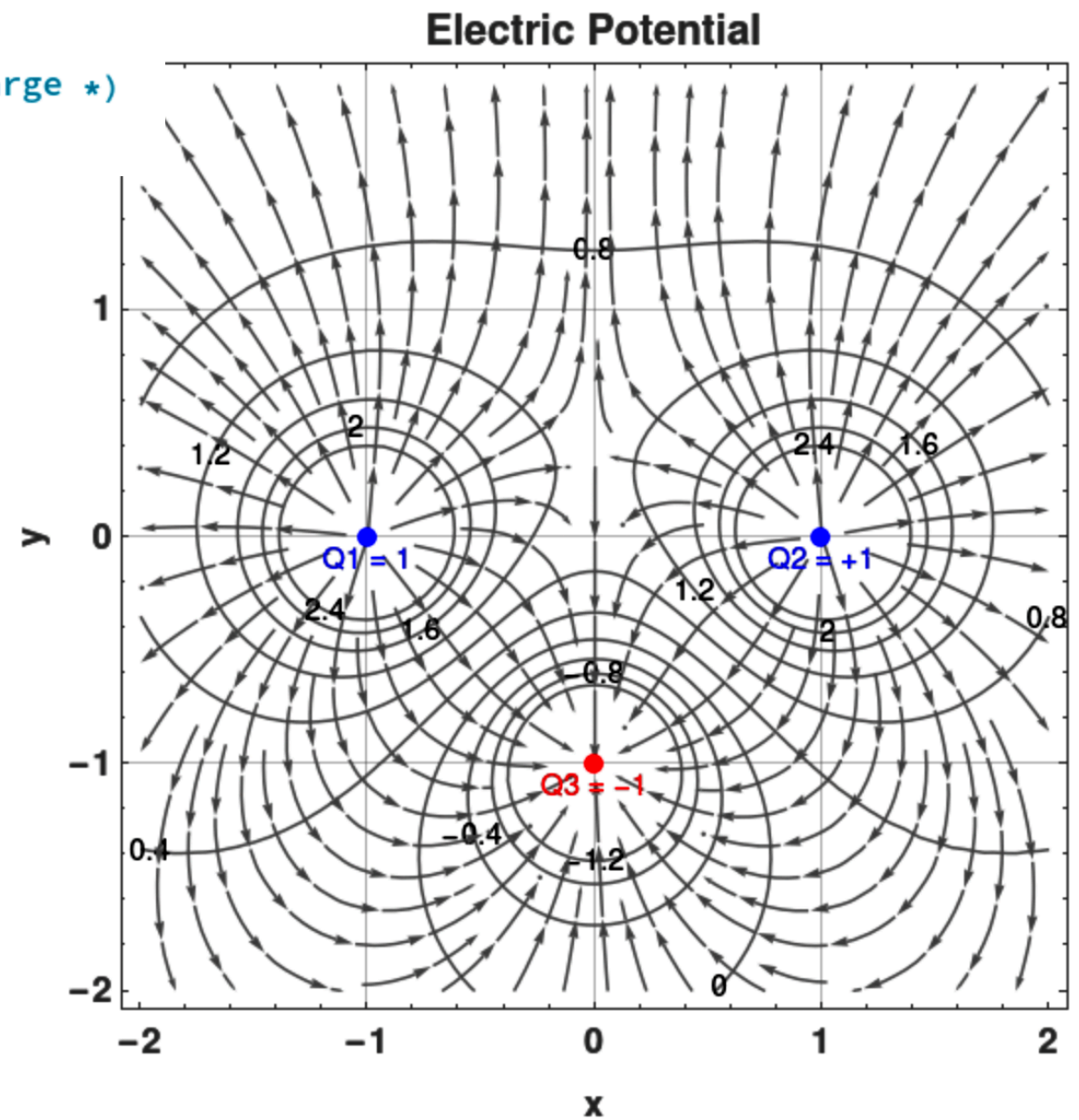
Overlay

```
5]:= Show[ep1, sp]
```



Adding Text

```
i6]:= Show[epl, sp,  
  Epilog ->  
  {  
    {Blue, PointSize[0.02], Point[{-1, 0}]}, (* 1st charge *)  
    {Blue, Text["Q1 = 1", {-1, -0.1}]},  
    {Blue, PointSize[0.02], Point[{1, 0}]}, (* 2nd charge *)  
    {Blue, Text["Q2 = +1", {1, -0.1}]},  
    {Red, PointSize[0.02], Point[{0, -1}]}, (* 3rd charge *)  
    {Red, Text["Q3 = -1", {0, -1.1}]}]}
```



ODE - Ordinary Differential Equations

1st Order

$$\frac{dv}{dt} + a v + b = 0$$

```
Clear[a, b]
```

```
DSolve[v'[t] + a v[t] + b == 0, v[t], t]
```

```
{ {v[t] -> -b/a + e^{-a t} c1} } (* C1 is the integration constant *)
```

General solutions have an unknown integration constant.

```
Clear[a, b, c]
```

```
DSolve[{v'[t] + a v[t] + b == 0, v[0] == c}, v[t], t]
```

```
{ {v[t] -> -b/a + (c + b/a) e^{-a t} } }
```

The integration constant can be found if there is an initial condition.

RC Charging Current

$$\frac{di}{dt} + \frac{1}{RC} i = 0$$

= `Clear[R, c, Ein, cur]` (* It is essential to clear all variables *)

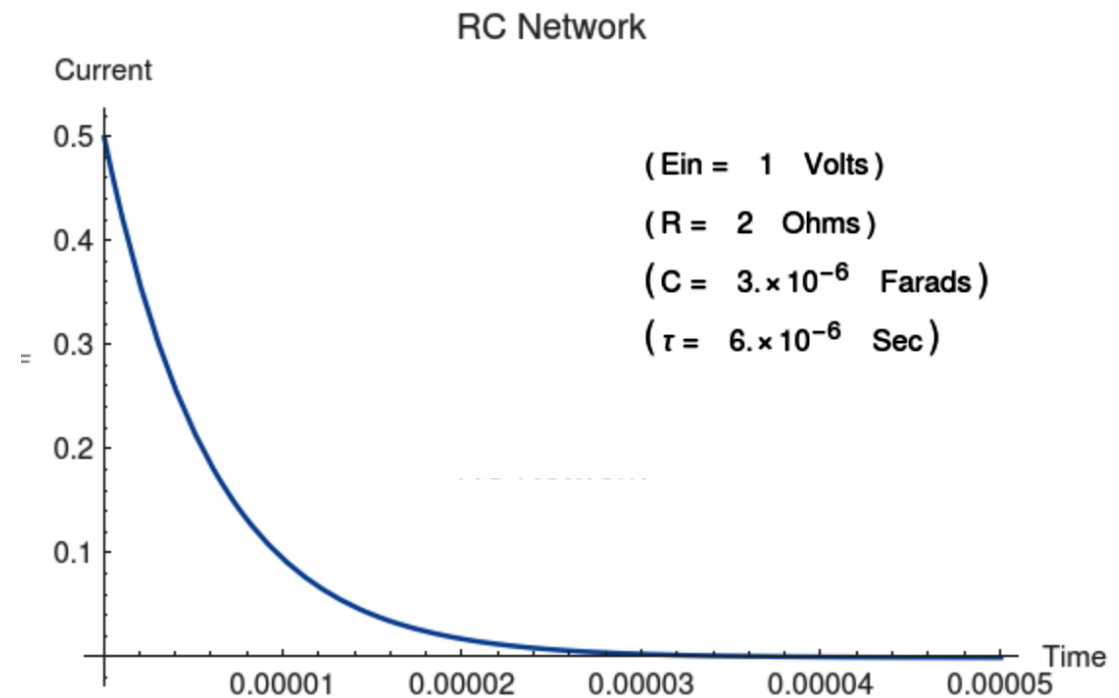
`DSolve[{cur'[t] +  $\frac{1}{R c}$  cur[t] == 0, cur[0] ==  $\frac{Ein}{R}$ }, cur[t], t]`

$$= \left\{ \left\{ \text{cur}[t] \rightarrow \frac{e^{-\frac{t}{cR}} Ein}{R} \right\} \right\}$$

`Ein = 1; R = 2; c =  $3 \times 10^{-6}$ ;  $\tau = R c$ ;`

$$\text{cur} = \frac{e^{-\frac{t}{cR}} Ein}{R};$$

`Plot[cur, {t, 0, 50×10^{-6} },
 PlotRange → {Automatic, Full},
 AxesLabel → {"Time", "Current"},
 PlotLabel → "RC Network",
 Epilog → {
 Text[{"Ein = ", Ein, " Volts"}], Scaled[{0.6, 0.9}], Left],
 Text[{"R = ", R, " Ohms"}], Scaled[{0.6, 0.8}], Left],
 Text[{"C = ", N[c], " Farads"}], Scaled[{0.6, 0.7}], Left],
 Text[{" τ = ", N[τ], " Sec"}], Scaled[{0.6, 0.6}], Left]
 }
]`



2nd Order

No Initial Conditions

$$a \frac{d^2}{dt^2} v + b \frac{d}{dt} v + c v = 0$$

```
Clear[a, b, c]
```

```
DSolve[a v''[t] + b v'[t] + c v[t] == 0, v[t], t]
```

$$\left\{ \left\{ v[t] \rightarrow e^{\frac{1}{2} \left(-\frac{b}{a} - \frac{\sqrt{b^2 - 4ac}}{a} \right) t} c_1 + e^{\frac{1}{2} \left(-\frac{b}{a} + \frac{\sqrt{b^2 - 4ac}}{a} \right) t} c_2 \right\} \right\}$$

1 Initial Condition $v[0]==d$

```
Clear[a, b, c]
```

```
DSolve[{a v''[t] + b v'[t] + c v[t] == 0, v[0] == d}, v[t], t] // Simplify
```

$$\left\{ \left\{ v[t] \rightarrow e^{-\frac{(b + \sqrt{b^2 - 4ac}) t}{2a}} \left(d e^{\frac{\sqrt{b^2 - 4ac} t}{a}} + c_1 - e^{\frac{\sqrt{b^2 - 4ac} t}{a}} c_1 \right) \right\} \right\}$$

2 Initial Conditions $v[0]==d$, $v'[0]==e$

```
Clear[a, b, c]
```

```
DSolve[{a v''[t] + b v'[t] + c v[t] == 0, v[0] == d, v'[0] == e}, v[t], t] //  
FullSimplify
```

$$\left\{ \left\{ v[t] \rightarrow e^{-\frac{bt}{2a}} \left(d \cosh \left[\frac{\sqrt{b^2 - 4ac} t}{2a} \right] + \frac{(bd + 2ae) \sinh \left[\frac{\sqrt{b^2 - 4ac} t}{2a} \right]}{\sqrt{b^2 - 4ac}} \right) \right\} \right\}$$

RLC Parallel Circuit

Impulse Response

The circuit equation is found by writing KCL around the loop.

$$\frac{d^2 v}{dt^2} + \frac{1}{RC} \frac{dv}{dt} + \frac{1}{LC} v = 0$$

```
]:= Clear[a, b, res, ind, cap, soln]
```

```
DSolve[{v''[t] + a v'[t] + b v[t] == 0, v[0] == 1, v'[0] == 1/a}, v[t], t] //
```

```
FullSimplify
```

```
]:= {{v[t] -> e^(-a t/2) (Cosh[1/2 sqrt(a^2 - 4 b) t] + (2 + a^2) Sinh[1/2 sqrt(a^2 - 4 b) t] / (a sqrt(a^2 - 4 b)))}}
```

```
]:= soln = e^(-a t/2) (Cosh[1/2 sqrt(a^2 - 4 b) t] + (2 + a^2) Sinh[1/2 sqrt(a^2 - 4 b) t] / (a sqrt(a^2 - 4 b)));
```

```
]:= a = 1/(res cap); b = 1/(ind cap); res = 1 x 10^3; ind = 10 x 10^-3; cap = 10 x 10^-6;
```

```
ω0 = 1/sqrt(ind cap); (* characteristic frequency *)
```

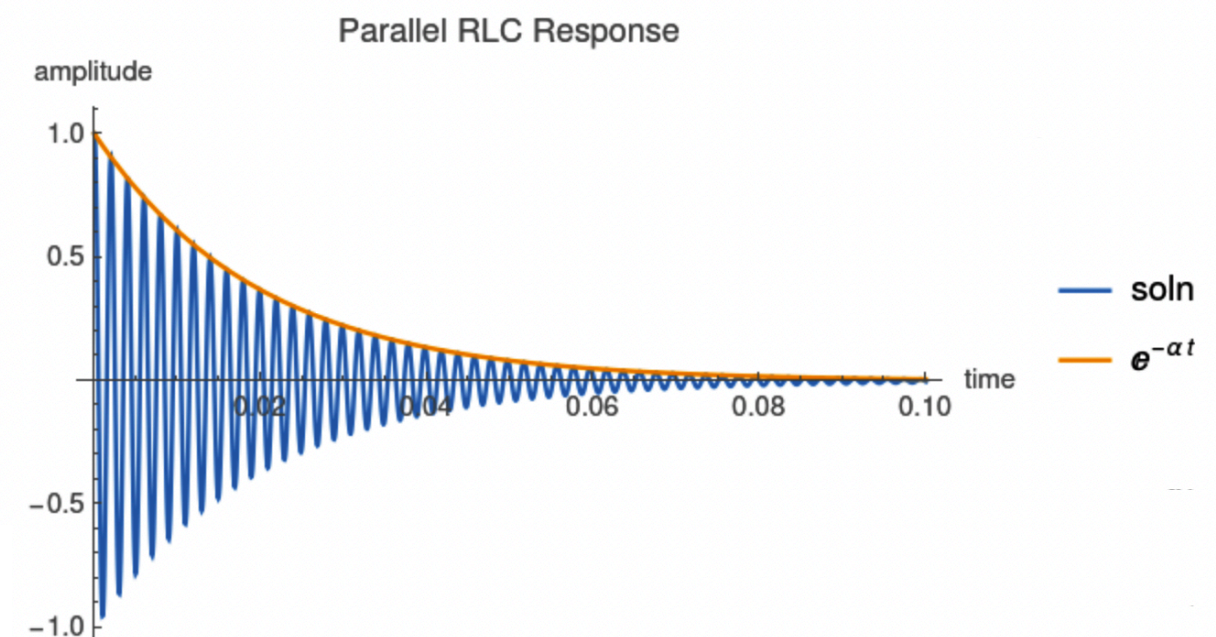
```
]:= fo = ω0/(2.0 π);
```

```
]:= α = 1.0/(2 res cap);
```

```
]:= τ = 1/α; (* time constant *)
```

To determine the circuit currents, a small resistor should be placed in series with the input voltage or in series with the capacitor.

```
Plot[{soln, e^(-α t)}, {t, 0, 100 x 10^-3},  
PlotRange -> All,  
AxesLabel -> {"time", "amplitude"},  
PlotLabel -> "Parallel RLC Response",  
PlotLegends -> "Expressions"]
```



RLC DC Series Circuit

$$L \frac{di}{dt} + Ri + \frac{1}{C} \int i dt = v_{in}$$

or
$$\frac{d^2 i}{dt^2} + \frac{R}{L} \frac{di}{dt} + \frac{1}{LC} i = 0$$

`Clear[L1, R1, C1, Vin, i]`

`DSolve[{i''[t] +  $\frac{R1}{L1}$  i'[t] +  $\frac{1}{L1 C1}$  i[t] == 0, i[0] == 0, i'[0] ==  $\frac{Vin}{L1}$ }, i[t], t] //`

`FullSimplify`

$$\left\{ \left\{ i[t] \rightarrow \frac{2 \sqrt{C1} e^{-\frac{R1 t}{2 L1}} Vin \sinh \left[\frac{\sqrt{-4 L1 + C1 R1^2} t}{2 \sqrt{C1} L1} \right]}{\sqrt{-4 L1 + C1 R1^2}} \right\} \right\}$$

$$i = \frac{2 \sqrt{C1} e^{-\frac{R1 t}{2 L1}} Vin \sinh \left[\frac{\sqrt{-4 L1 + C1 R1^2} t}{2 \sqrt{C1} L1} \right]}{\sqrt{-4 L1 + C1 R1^2}};$$

`Vin = 1;`

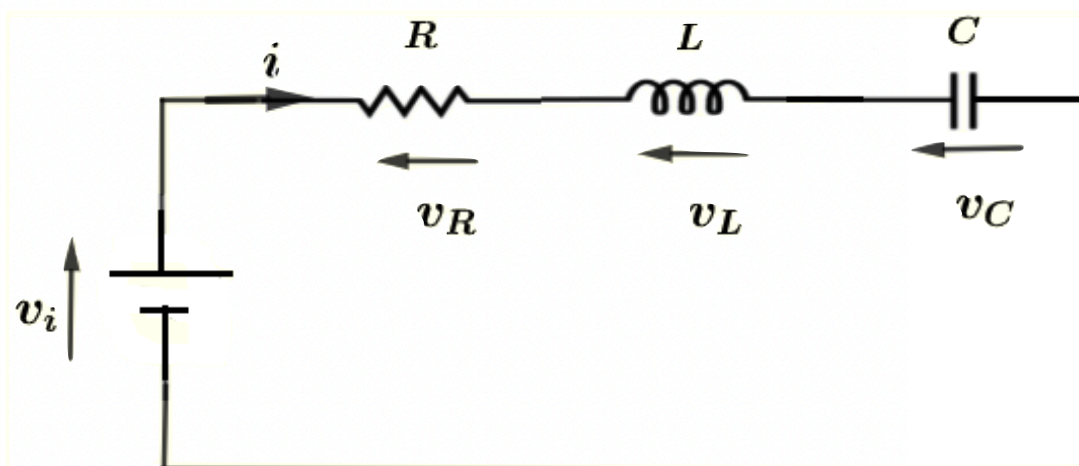
`L1 = 1;`

`R1 = 1;`

`C1 = 1;`

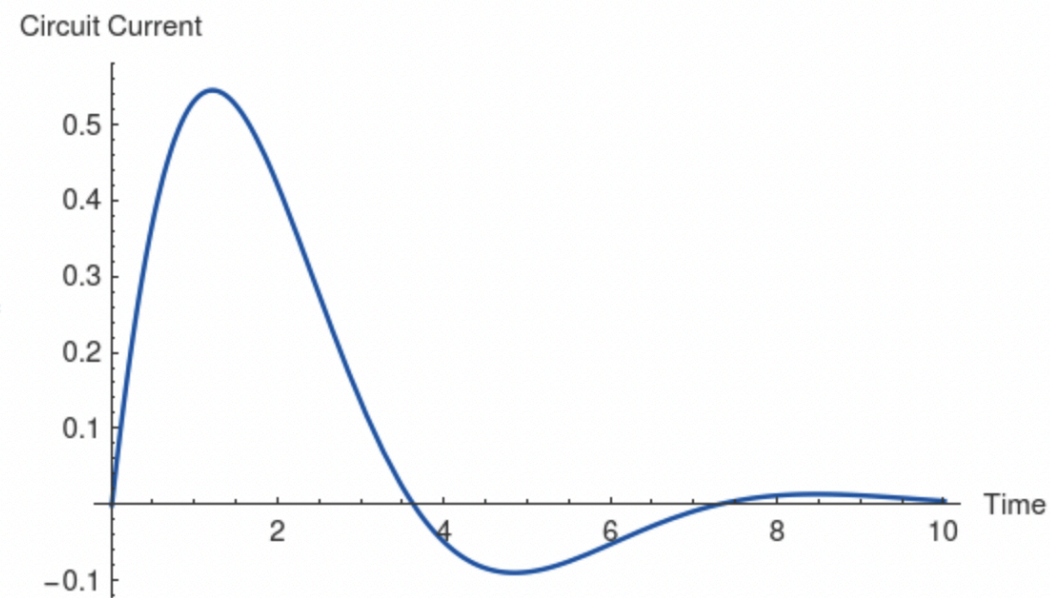
The circuit equation is found by writing KVL around the loop.

RLC Series Circuit

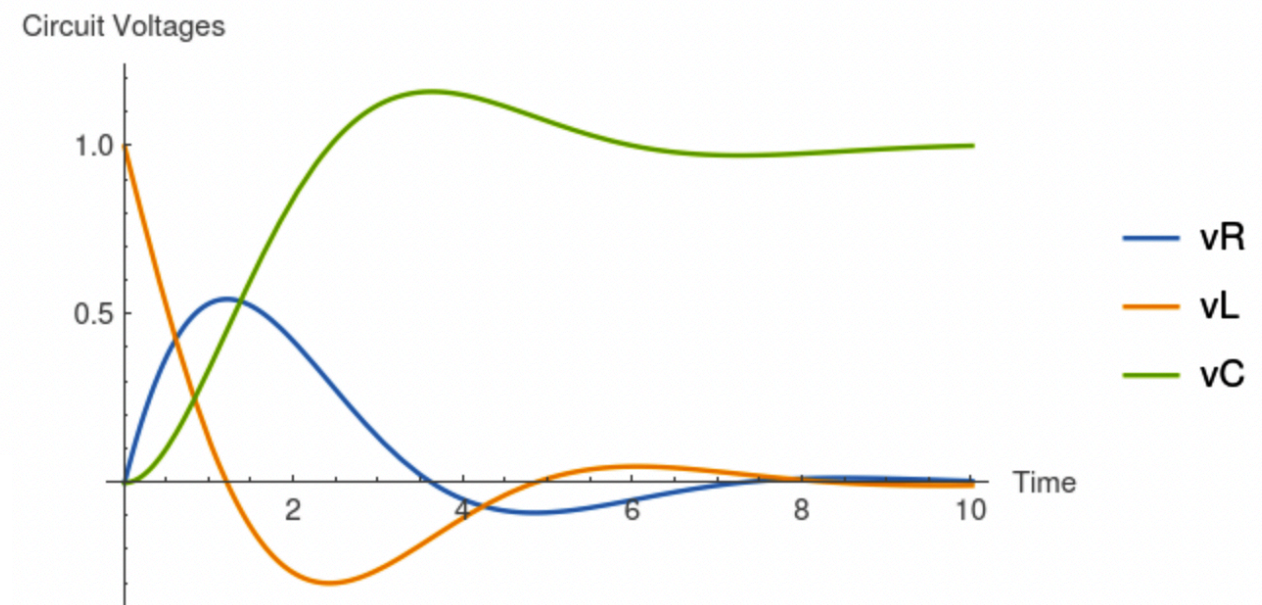


Courtesy: [Math for Engineers](#)

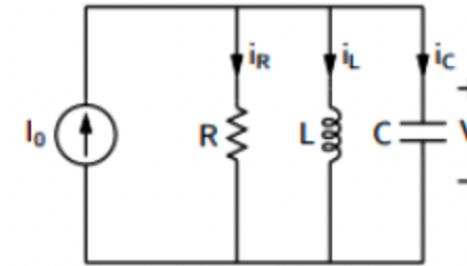

```
Plot[i, {t, 0, 10}, PlotRange -> All,
  AxesLabel -> {"Time", "Circuit Current"}]
```



```
vR = i R1;
vC = 1/C1 ∫ i dt + Vin;
vL = L1 D[i, t];
Plot[{vR, vL, vC}, {t, 0, 10}, PlotRange -> All,
  AxesLabel -> {"Time", "Circuit Voltages"},
  PlotLegends -> "Expressions"]
```



RLC DC Parallel Circuit



Find the voltage

$$I_R = \frac{V}{R} \quad I_L = \frac{1}{L} \int v dt + i(0) \quad I_C = C \frac{dv}{dt}$$

$$C \frac{dv}{dt} + \frac{V}{R} + \frac{1}{L} \int v dt + i(0) = I_{in}$$

$$C \frac{d^2 v}{dt^2} + \frac{V}{R} \frac{dv}{dt} + \frac{1}{L} v = 0$$

1st initial condition $v = 0$ @ $t = 0$ (the capacitor acts as a short circuit)

2nd initial condition $\frac{dv}{dt} = \frac{I_{in}}{C}$ @ $t = 0$ (all the current flows in the capacitor)

Clear[R1, L1, C1, i, Iin, V]

DSolve[{v''[t] + $\frac{1}{R1 C1}$ v'[t] + $\frac{1}{L1 C1}$ v[t] == 0, v[0] == 0, v'[0] == $\frac{Iin}{C1}$ },
v[t], t] // FullySimplify

FullySimplify[

$$\left\{ \left\{ v[t] \rightarrow - \frac{\left(e^{\frac{1}{2} \left(-\frac{1}{C1 R1} - \frac{\sqrt{L1 - 4 C1 R1^2}}{C1 \sqrt{L1} R1} \right) t} - e^{\frac{1}{2} \left(-\frac{1}{C1 R1} + \frac{\sqrt{L1 - 4 C1 R1^2}}{C1 \sqrt{L1} R1} \right) t} \right) Iin \sqrt{L1} R1}{\sqrt{L1 - 4 C1 R1^2}} \right\} \right\}$$

$$V = \frac{\left(e^{\frac{1}{2} \left(-\frac{1}{C1 R1} - \frac{\sqrt{L1 - 4 C1 R1^2}}{C1 \sqrt{L1} R1} \right) t} - e^{\frac{1}{2} \left(-\frac{1}{C1 R1} + \frac{\sqrt{L1 - 4 C1 R1^2}}{C1 \sqrt{L1} R1} \right) t} \right) Iin \sqrt{L1} R1}{\sqrt{L1 - 4 C1 R1^2}};$$

Find the Currents

$$IR = \frac{V}{R1};$$

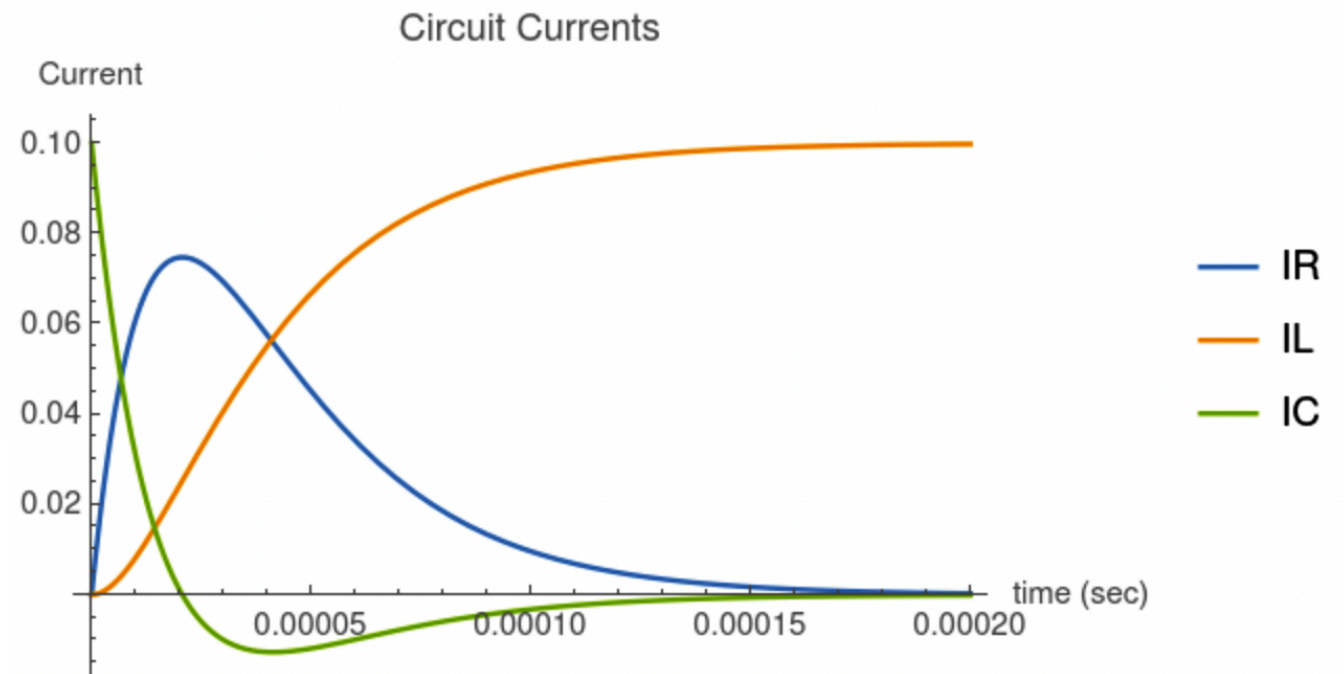
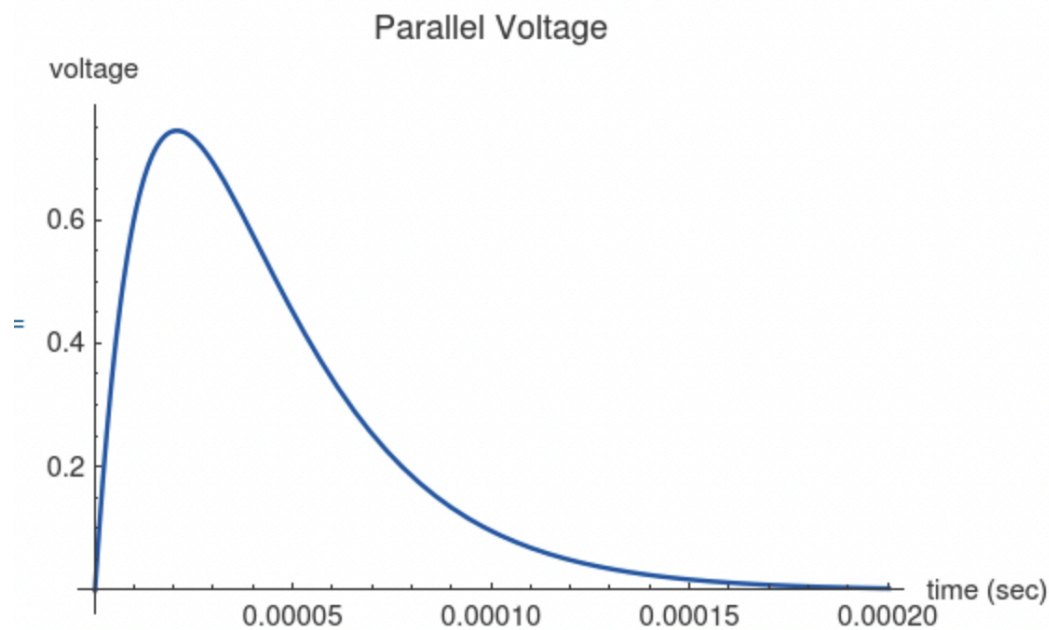
$$IL = \frac{1}{L1} \int V dt - I_{in};$$

$$IC = C1 D[V, t];$$

```
Plot[{IR, IL, IC}, {t, 0, 0.2 × 10-3},
  PlotRange → All,
  PlotLabel → "Circuit Currents",
  AxesLabel → {"time (sec)", "Current"},
  PlotLegends → "Expressions"]
```

```
R1 = 10 ; L1 = 0.44 × 10-3; C1 = 1 × 10-6; Iin = -100 × 10-3;
```

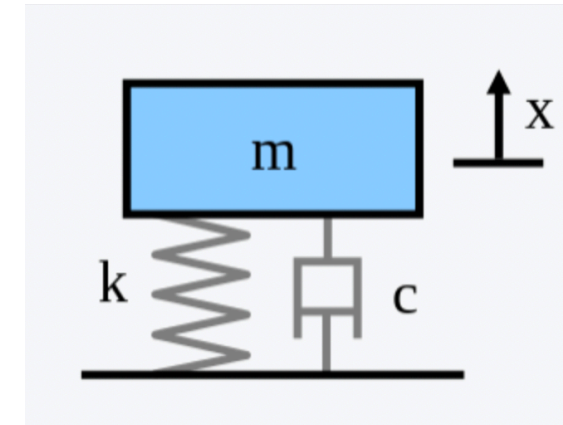
```
Plot[V, {t, 0, 0.2 × 10-3},
  PlotRange → All,
  PlotLabel → "Parallel Voltage",
  AxesLabel → {"time (sec)", "voltage"}]
```



Mass-Spring Damper System

$$\frac{d^2}{(dt)^2}x + 2\zeta\omega_n \frac{d}{dt}x + \omega_n^2 x = u$$

$$\ddot{x} + 2\zeta\omega_n \dot{x} + \omega_n^2 x = u \text{ where } \omega_n = \sqrt{\frac{k}{m}}; \quad \zeta = \frac{c}{2m\omega_n}; \quad u = \frac{F_{\text{external}}}{m}$$



[Courtesy: Wikipedia](#)

```
Clear[m, x, t, ξ, ωn, k, c0, fext, m, u, soln, x0]
```

```
x0 = -1; (*initial position*)
```

```
v0 = 0; (*initial velocity*)
```

```
m = 0.25; (*mass*)
```

```
k = 6; (*spring constant*)
```

```
c0 = 0.5; (*damping constant*)
```

```
fext = 0; (*external force*)
```

$$\omega_n = \sqrt{\frac{k}{m}};$$

$$\xi = \frac{c0}{2m\omega_n};$$

$$u = \frac{fext}{m};$$

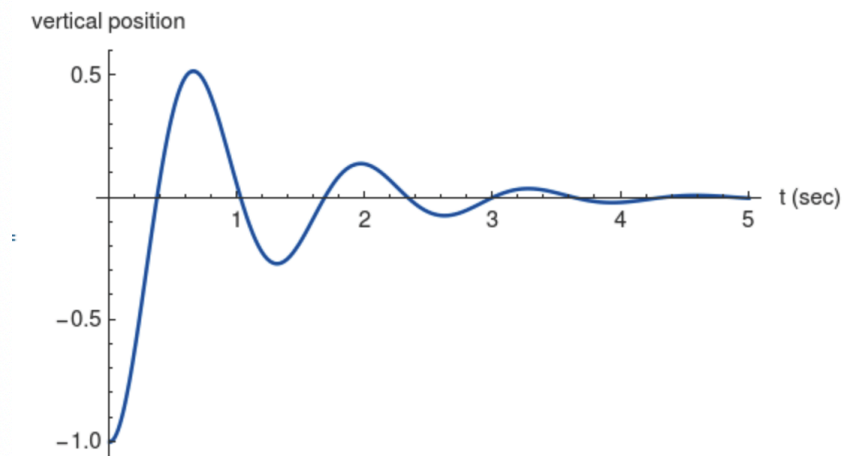
```
soln = DSolve[{x''[t] + 2 ξ ωn x'[t] + ωn^2 x[t] == u,  
             x'[0] == v0, x[0] == x0}, (*Initial Conditions*)  
             x[t], t] // FullSimplify
```

$$\left\{ \left\{ x[t] \rightarrow e^{-1 \cdot t} (-1. \cos[4.79583 t] - 0.208514 \sin[4.79583 t]) \right\} \right\}$$

copying the solution of x[t] above and equating it to x is the easiest way to proceed

```
x =  
  e-1. t  
  (-1. Cos[4.795831523312719 t] - 0.20851441405707477 Sin[4.795831523312719 t]);
```

```
Plot[x, {t, 0, 5},  
      PlotRange -> All,  
      AxesLabel -> {"t (sec)", "vertical position"}]
```



Advanced programmers can use the more cryptic statement:

```
x = soln[[1,1,2]]
```


Ballistic Trajectory

Ballistic Equations:

Derived from Newton's 2nd Law of Motion: $F = m a$

by definition: $a = \frac{dv}{dt}$ and $v = \frac{dx}{dt}$

Ideal: $m \frac{dv_x}{dt} = 0$ and $m \frac{dv_y}{dt} = -m g$

add an initial velocity v_0 at $t = 0$ if required.

If the initial velocity occurs at angle θ , then: $v_x[0] = v_0 \cos[\theta]$ and $v_y[0] = v_0 \sin[\theta]$

With Drag: $m \frac{dv_x}{dt} + k v = 0$ and $m \frac{dv_y}{dt} + k v + m g = 0$

to obtain the x and y velocity: solve the differential equations

to obtain the x and y positions: take the 2nd derivative and solve the equations

Ideal Case

Horizontal Velocity

```
Clear["Global`*"]
```

```
DSolve[{m vx'[t] == 0,  
        vx[0] == v Cos[theta]},  
        vx[t], t] // Simplify
```

```
{{vx[t] -> v Cos[theta]}}
```

```
vx = v Cos[theta];
```

Horizontal Position

```
DSolve[{m x''[t] == 0,  
        x[0] == 0,  
        x'[0] == v Cos[theta]},  
        x[t], t] // Simplify
```

```
{{x[t] -> t v Cos[theta]}}
```

```
x = t v Cos[theta];
```

Vertical Velocity

```
DSolve[{m vy'[t] + m g == 0,  
        vy[0] == v Sin[theta]},  
        vy[t], t] // FullSimplify
```

```
{{vy[t] -> -g t + v Sin[theta]}}
```

```
vy = -g t + v Sin[theta];
```

Vertical Position

```
DSolve[{m y''[t] + m g == 0,  
        y[0] == 0,  
        y'[0] == v Sin[theta]},  
        y[t], t] // FullSimplify
```

```
{{y[t] -> -g t^2 / 2 + t v Sin[theta]}}
```

```
y = -g t^2 / 2 + t v Sin[theta Degree];
```

Flight Time

```
Solve[-g t^2 / 2 + t v Sin[theta] == 0, t]
```

```
{{t -> 0}, {t -> 2 v Sin[theta] / g}}
```

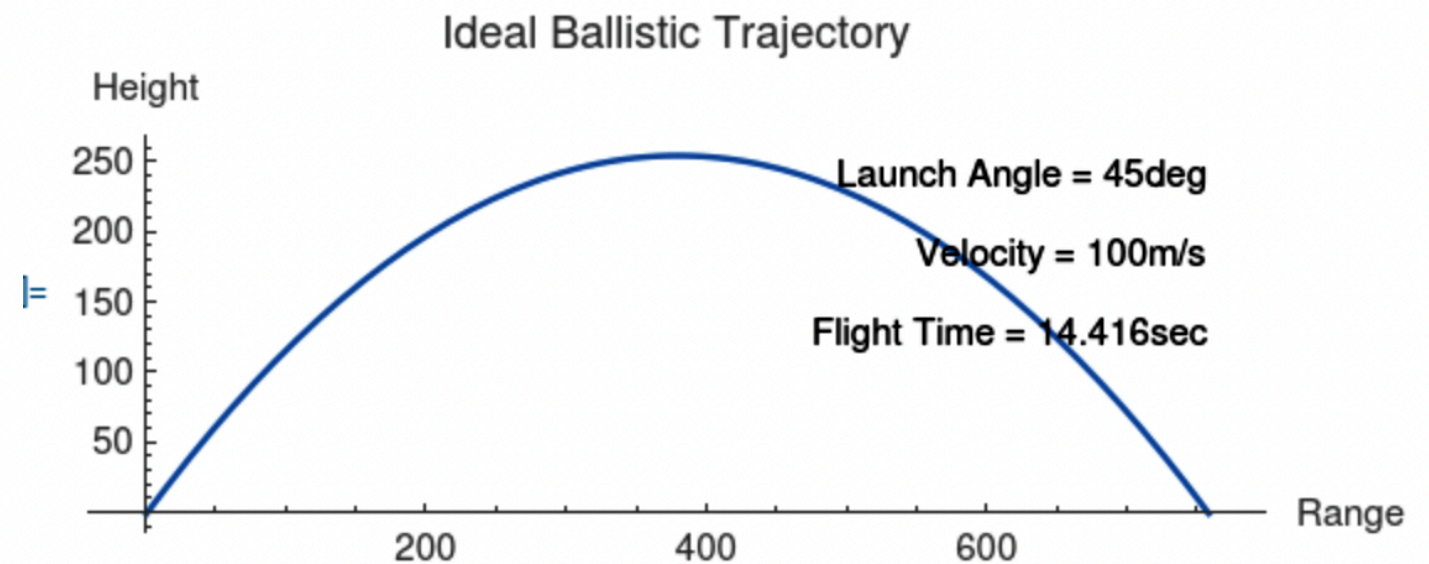
```
ft = 14.416;
```

Flight time can be determined by setting the altitude y, to zero and solving for time.

Plots

```
area = 0.18; (* surface area *)
dc = 100;
(* drag coefficient due to shape, turbulence etc. *)
ρ = 1.21; (* air density *)
g = 9.81; (* gravitational acceleration *)
m = 75; (* mass *)
k =  $\frac{1}{2} \rho \text{ dc area}$ ;
θ = 45; (* launch angle *)
v = 100; (* initial velocity *)
```

```
ParametricPlot[{x, y}, {t, 0, ft}, PlotRange → All,
  PlotLabel → "Ideal Ballistic Trajectory",
  AxesLabel → {"Range", "Height"},
  ImageSize → Medium,
  Epilog → {
    Text[Row[{"Launch Angle = ", θ, "deg"}], Scaled[{0.95, 0.9}], Right],
    Text[Row[{"Velocity = ", v, "m/s"}], Scaled[{0.95, 0.7}], Right],
    Text[Row[{"Flight Time = ", ft, "sec"}], Scaled[{0.95, 0.5}],
      Right]]]
```



With Drag

Horizontal Velocity

```
= Clear["Global`*"]

DSolve[{m vx'[t] + k vx[t] == 0,
  vx[0] == v Cos[θ]},
  vx[t], t] // Simplify

= {{vx[t] → e- $\frac{kt}{m}$  v Cos[θ]}}
```

```
= vx = e- $\frac{kt}{m}$  v Cos[θ Degree];
```

Horizontal Position

```
DSolve[{m x''[t] + k x'[t] == 0,
  x[0] == 0,
  x'[0] == v Cos[θ]},
  x[t], t] // Simplify

{{x[t] →  $\frac{(1 - e^{-\frac{kt}{m}}) m v \cos[\theta]}{k}$ }}
```

$$x = \frac{(1 - e^{-\frac{kt}{m}}) m v \cos[\theta]}{k};$$

Vertical Velocity

```
DSolve[{m vy'[t] + k vy[t] + m g == 0,
  vy[0] == v Sin[θ]},
  vy[t], t] // FullSimplify

{{vy[t] →  $\frac{-gm + e^{-\frac{kt}{m}} (gm + k v \sin[\theta])}{k}$ }}
```

$$vy = \frac{-gm + e^{-\frac{kt}{m}} (gm + k v \sin[\theta \text{ Degree}])}{k};$$

Vertical Position

```
DSolve[{m y''[t] + k y'[t] + m g == 0,
  y[0] == 0,
  y'[0] == v Sin[θ]},
  y[t], t] // FullSimplify

{{y[t] →  $\frac{m (gm - gkt + k v \sin[\theta] - e^{-\frac{kt}{m}} (gm + k v \sin[\theta]))}{k^2}$ }}
```

$$y = \frac{1}{k^2} m (gm - gkt + k v \sin[\theta \text{ Degree}] - e^{-\frac{kt}{m}} (gm + k v \sin[\theta \text{ Degree}]));$$

Flight Time

```
Solve[ $\frac{1}{k^2} m (gm - gkt + k v \sin[\theta \text{ Degree}] - e^{-\frac{kt}{m}} (gm + k v \sin[\theta \text{ Degree}])) == 0, t]$ 
```

```
{{t →  $\frac{gm + gm \text{ProductLog}[-\frac{e^{-1 - \frac{kv \sin[\theta]}}{gm}} (gm + kv \sin[\theta])]}{gk} + kv \sin[\theta]}$ }}
```

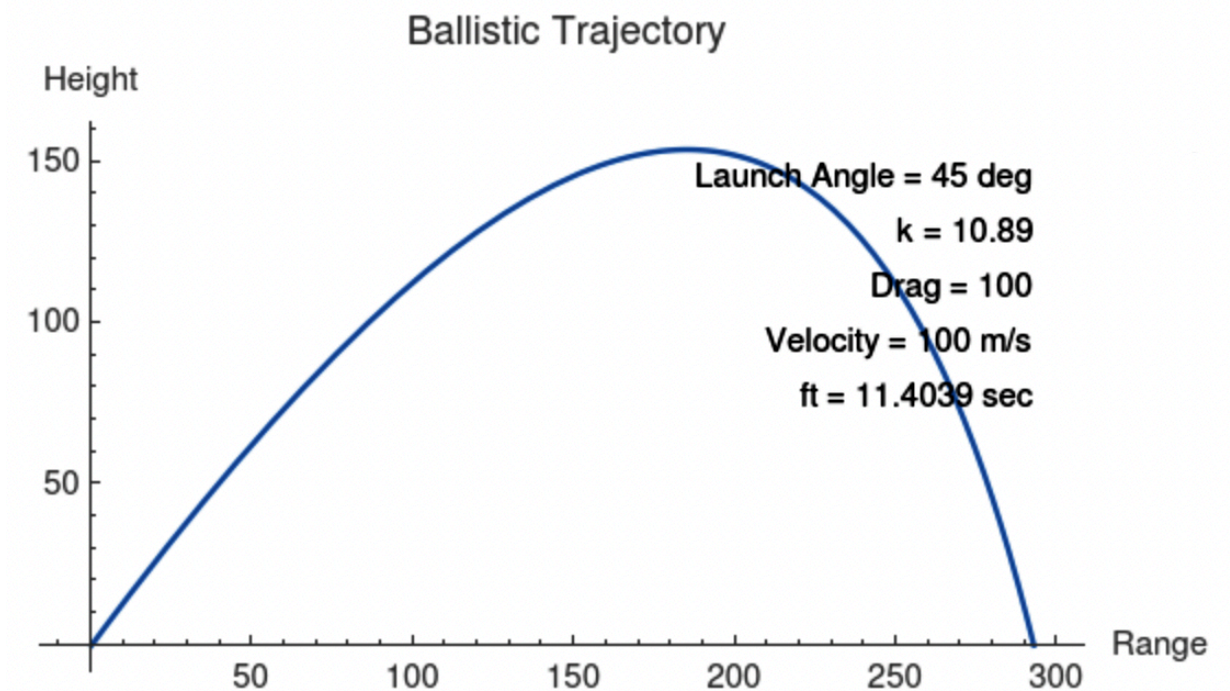
```
ft = 11.403891144324234`;
```

The exact nature of drag is very complex. Each application relies heavily on empirical measurements.

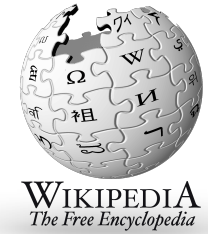
Plots

```
area = 0.18; (* surface area *)  
dc = 100; (* drag coefficient due to shape, turbulence etc. *)  
 $\rho = 1.21$ ; (* air density *)  
g = 9.81; (* gravitational acceleration *)  
m = 75; (* mass *)  
 $k = \frac{1}{2} \rho \text{ dc area};$   
 $\theta = 45$ ; (* launch angle *)  
v = 100; (* initial velocity *)
```

```
ParametricPlot[{x, y}, {t, 0, ft}, PlotRange → All,  
  PlotLabel → "Ballistic Trajectory",  
  AxesLabel → {"Range", "Height"},  
  Epilog → {  
    Text[Row[{"Launch Angle = ",  $\theta$ , " deg"}], Scaled[{0.95, 0.9}], Right],  
    Text[Row[{"k = ", k}], Scaled[{0.95, 0.8}], Right],  
    Text[Row[{"Drag = ", dc}], Scaled[{0.95, 0.7}], Right],  
    Text[Row[{"Velocity = ", v, " m/s"}], Scaled[{0.95, 0.6}], Right],  
    Text[Row[{"ft = ", ft, " sec"}], Scaled[{0.95, 0.5}], Right]]]
```



Convolution

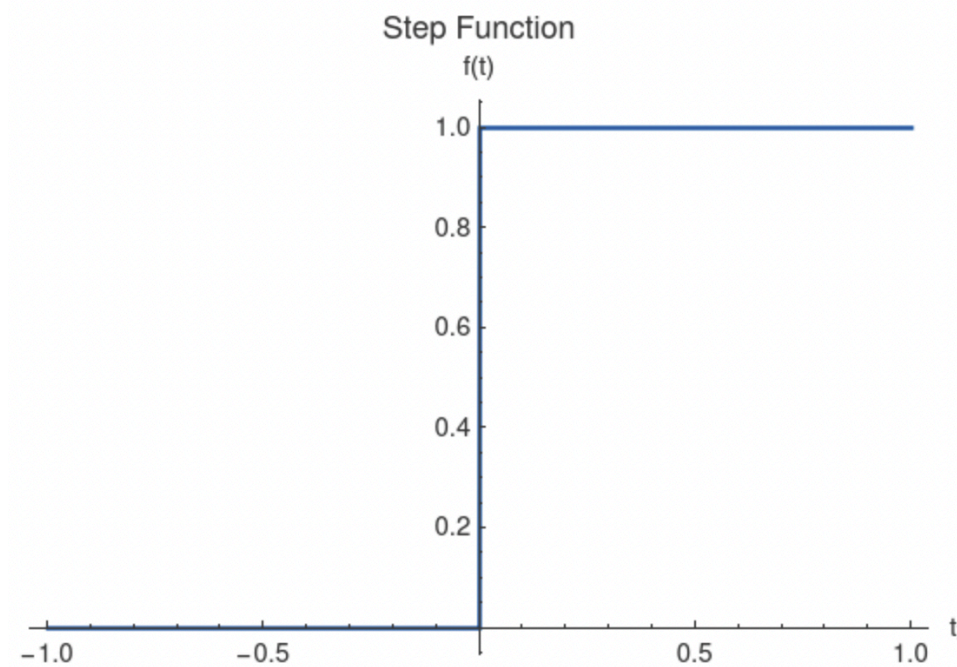


Auto-convolution of a Step Function

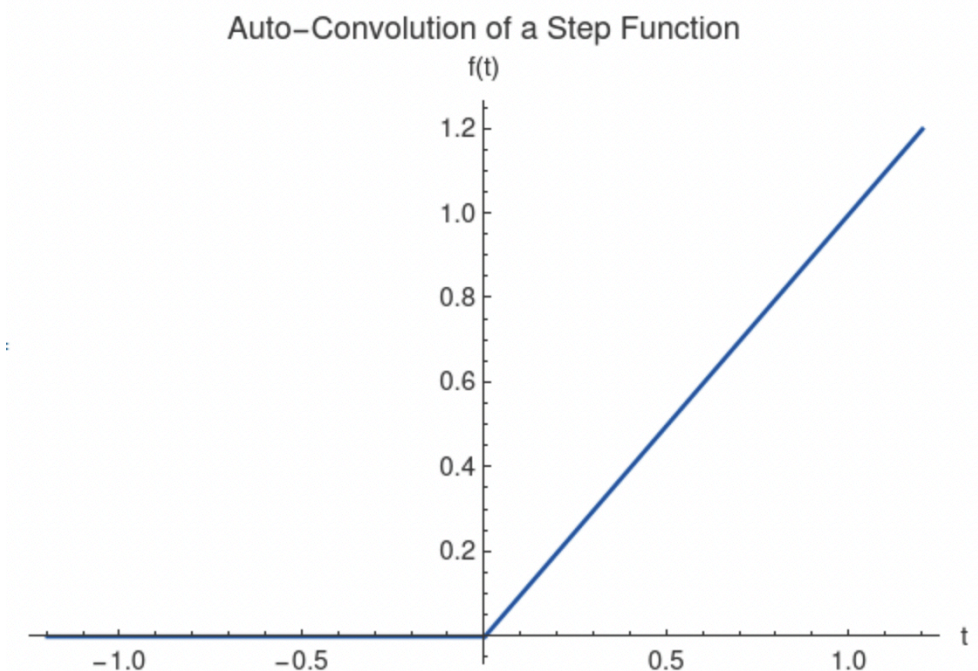
τ is a dummy variable.

```
Clear[x, y]
```

```
Plot[UnitStep[x], {x, -1, 1},  
      Exclusions → None,  
      PlotLabel → "Step Function",  
      AxesLabel → {"t", "f(t)"}]
```

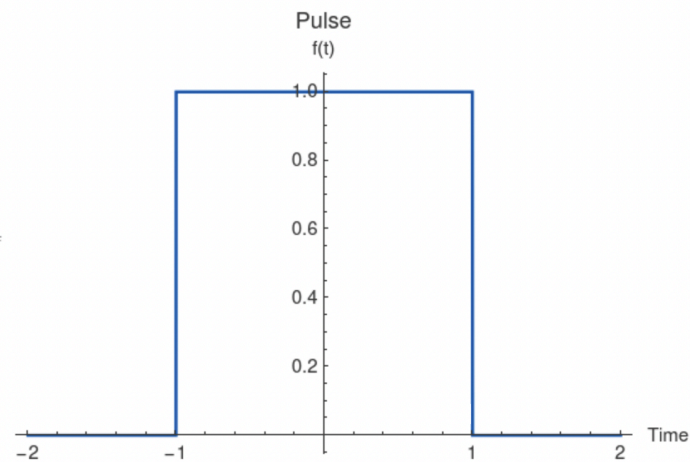


```
Convolve[UnitStep[t], UnitStep[t], t,  $\tau$ ];  
Plot[%, { $\tau$ , -1.2, 1.2},  
      PlotLabel → "Auto-Convolution of a Step Function",  
      AxesLabel → {"t", "f(t)"}]
```

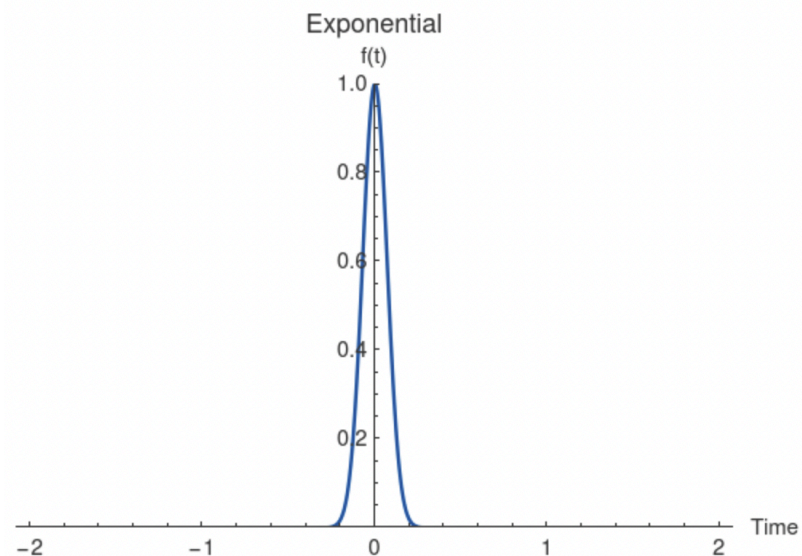


Convolution of a Pulse & Exponential

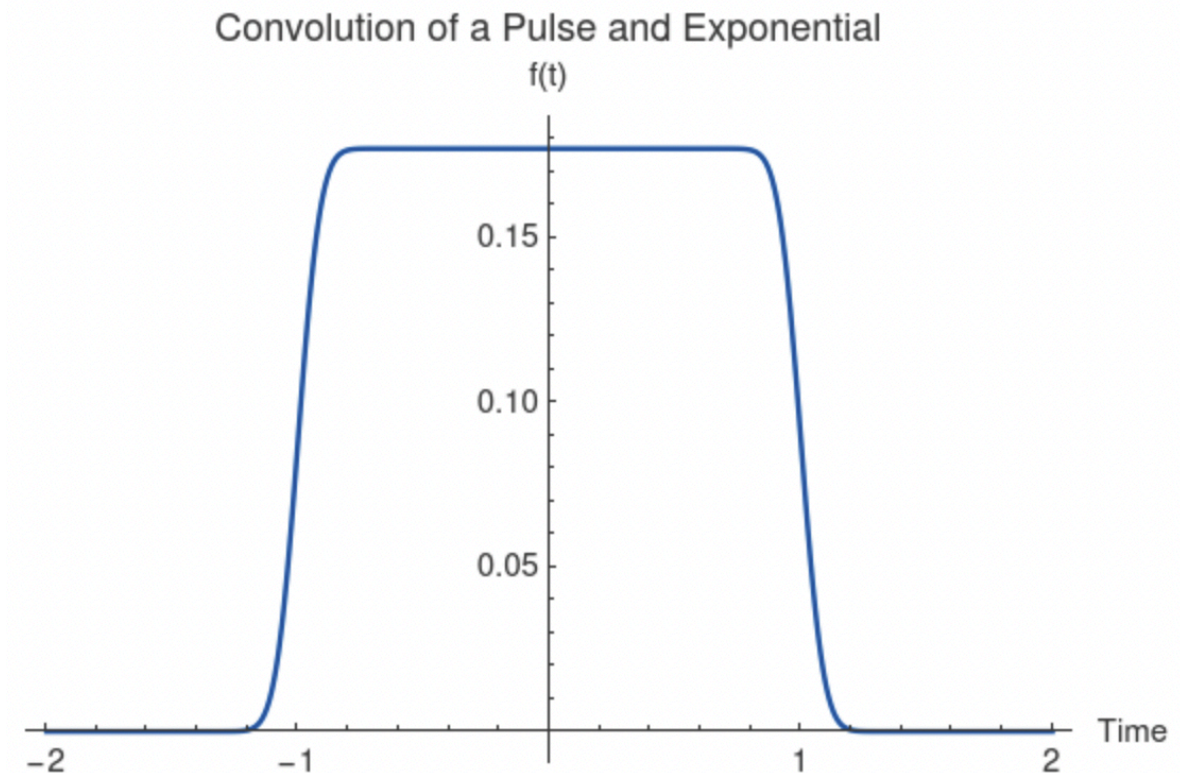
```
Clear[pulse2, t, a, exp2, con2]
pulse2 = UnitStep[t + 1] - UnitStep[t - 1];
Plot[pulse2, {t, -2, 2},
  PlotLabel -> "Pulse",
  AxesLabel -> {"Time", "f(t)"},
  Exclusions -> None]
```



```
a = 100;
exp2 = e-a t2;
Plot[exp2, {t, -2, 2}, PlotRange -> {0, 1},
  PlotLabel -> "Exponential",
  AxesLabel -> {"Time", "f(t)"}]
```



```
con2 = Convolve[exp2, pulse2, t,  $\tau$ ];
Plot[con2, { $\tau$ , -2, 2},
  PlotRange -> All,
  PlotLabel -> "Convolution of a Pulse and Exponential",
  AxesLabel -> {"Time", "f(t)"}]
```



Discrete Convolution of a Rectangular Pulse and ideal LPF

```

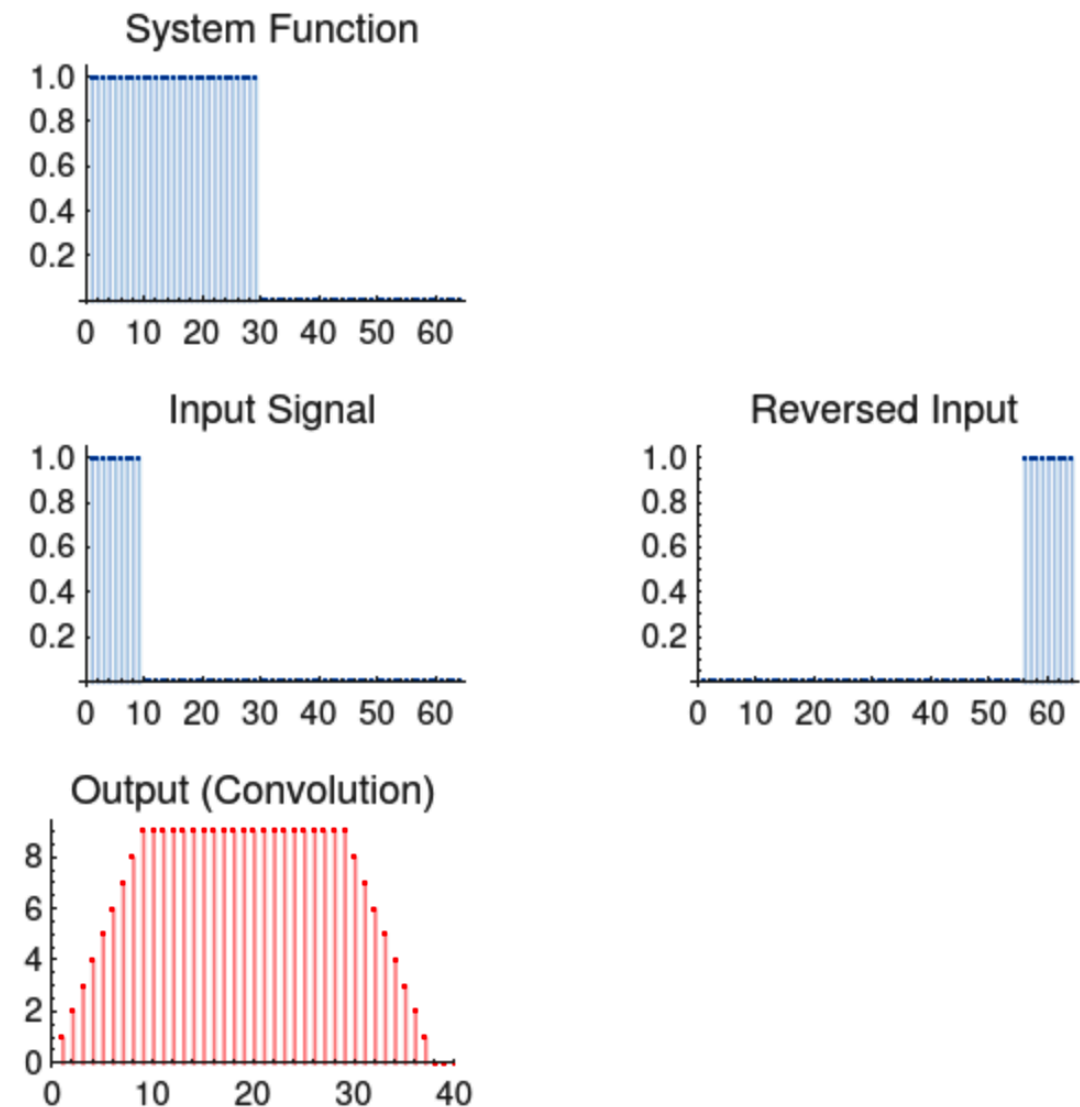
Clear[ns, k, hsig, xsig, xrev, con]

ns = 64;
k = Table[i, {i, 1, ns}];

hsig = Table[(UnitStep[n] - UnitStep[n - 30]), {n, 1, ns}];
xsig = Table[UnitStep[n] - UnitStep[n - 10], {n, 1, ns}];
xrev = Reverse[Take[xsig, {1, ns}]];
con = ListConvolve[hsig, xsig, {1, -1}, 0];

GraphicsGrid[{
  {ListPlot[hsig, PlotRange → All,
    PlotLabel → "System Function", Filling → Axis]},
  {ListPlot[xsig, PlotRange → All,
    PlotLabel → "Input Signal", Filling → Axis],
   ListPlot[xrev, PlotRange → All,
    PlotLabel → "Reversed Input", Filling → Axis]},
  {ListPlot[con, PlotRange → {{0, 40}, All},
    PlotLabel → "Output (Convolution)", Filling → Axis,
    PlotStyle → Red]}],
ImageSize → Medium]

```



This demonstrates that if a square pulse passes through an ideal low pass filter (LPF), the output is distorted.

Discrete Convolution of a Sinc Pulse and ideal LPF

```

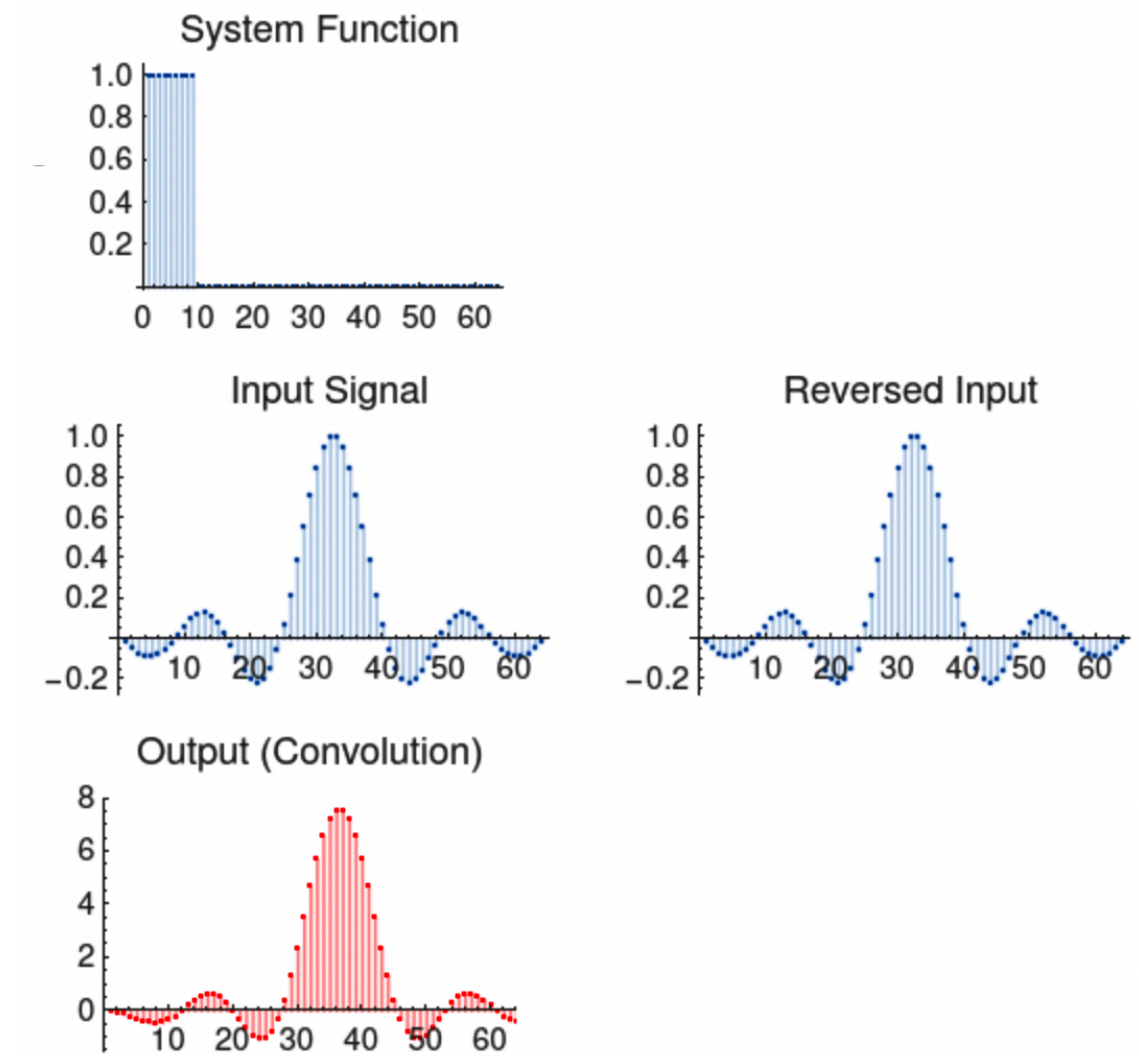
Clear[ns, k, hsig, xsig, xrev, con]

ns = 64;
k = Table[i, {i, 1, ns}];

hsig = Table[(UnitStep[n] - UnitStep[n - 10]), {n, 1, ns}];
xsig = Table[Sinc[ $\frac{n}{ns} 8 \pi$ ], {n,  $\frac{-ns + 1}{2}$ ,  $\frac{ns}{2}$ });
xrev = Reverse[Take[xsig, {1, ns}]];
con = ListConvolve[hsig, xsig, {1, -1}, 0];

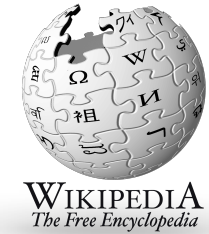
GraphicsGrid[{
  {ListPlot[hsig, PlotRange → All,
    PlotLabel → "System Function", Filling → Axis]},
  {ListPlot[xsig, PlotRange → All,
    PlotLabel → "Input Signal", Filling → Axis],
   ListPlot[xrev, PlotRange → All,
    PlotLabel → "Reversed Input", Filling → Axis]},
  {ListPlot[con, PlotRange → {{0, ns}, All},
    PlotLabel → "Output (Convolution)", Filling → Axis,
    PlotStyle → Red]}],
ImageSize → Medium]

```



This demonstrates that if a sinc pulse passes through an ideal low pass filter (LPF), the output is not distorted.

Correlation



All communications signals pickup noise. In simple systems, it is often possible to eliminate this noise by means of bandpass filters. However, if the noise dominates the signal, correlation techniques are used.

If the signal has a known pattern, it can be extracted even from a noisy environment. This technique is used in satellite receivers and [CDMA](#) cell phone systems.

```
Clear[sc, spc, nos, dt, n, autoCorSig1, corSig1Sig2, corSig1Sig3, corSig1Sig4]
```

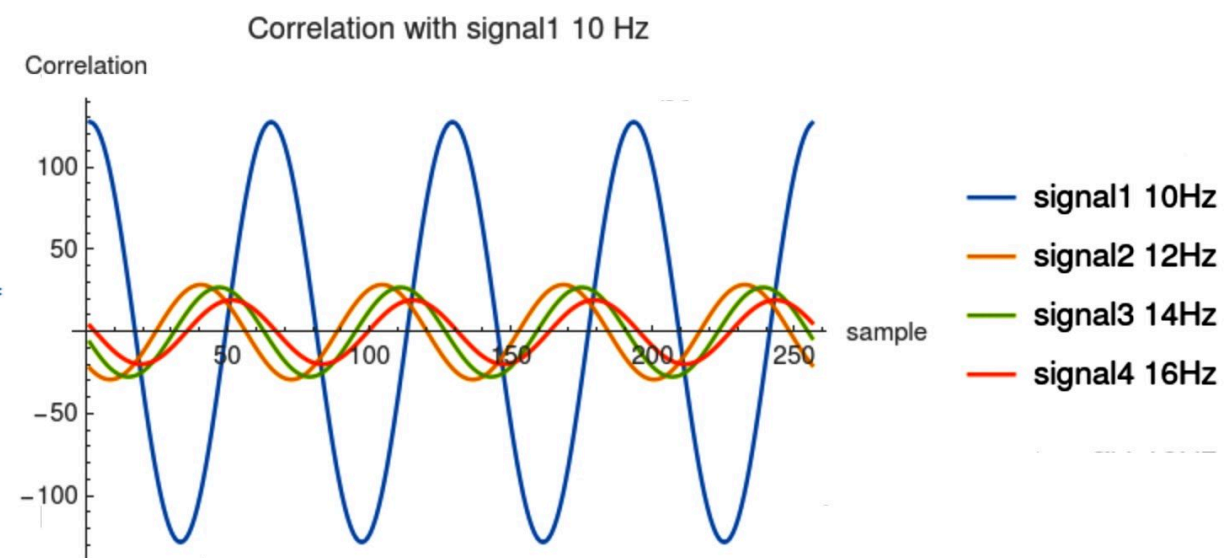
```
sc = 4; (*signal cycles*)
spc = 64; (*samples per cycle*)
nos = sc spc; (*number of samples*)
f1 = 10; (*frequency 1*)
f2 = 12; (*frequency 2*)
f3 = 14; (*frequency 3*)
f4 = 16; (*frequency 4*)
dt =  $\frac{1}{\text{spc } f_1}$ ; (*time increment*)
```

```
signal1 = Table[Sin[2  $\pi$  f1 n dt], {n, nos}]; (*signal1*)
signal2 = Table[Sin[2  $\pi$  f2 n dt], {n, nos}]; (*signal2*)
signal3 = Table[Sin[2  $\pi$  f3 n dt], {n, nos}]; (*signal3*)
signal4 = Table[Sin[2  $\pi$  f4 n dt], {n, nos}]; (*signal4*)
```

```
autoCorSig1 = ListCorrelate[signal1, signal1, {1, 1}];
corSig1Sig2 = ListCorrelate[signal1, signal2, {1, 1}];
corSig1Sig3 = ListCorrelate[signal1, signal3, {1, 1}];
corSig1Sig4 = ListCorrelate[signal1, signal4, {1, 1}];
```

Of these 4 signals, only 1
correlates with signal 1

```
ListLinePlot[{autoCorSig1, corSig1Sig2, corSig1Sig3, corSig1Sig4},
  PlotLegends → {"signal1 10Hz", "signal2 12Hz", "signal3 14Hz", "signal4 16Hz"},
  AxesLabel → {"sample", "Correlation"},
  PlotLabel → "Correlation with signal1 10 Hz"]
```

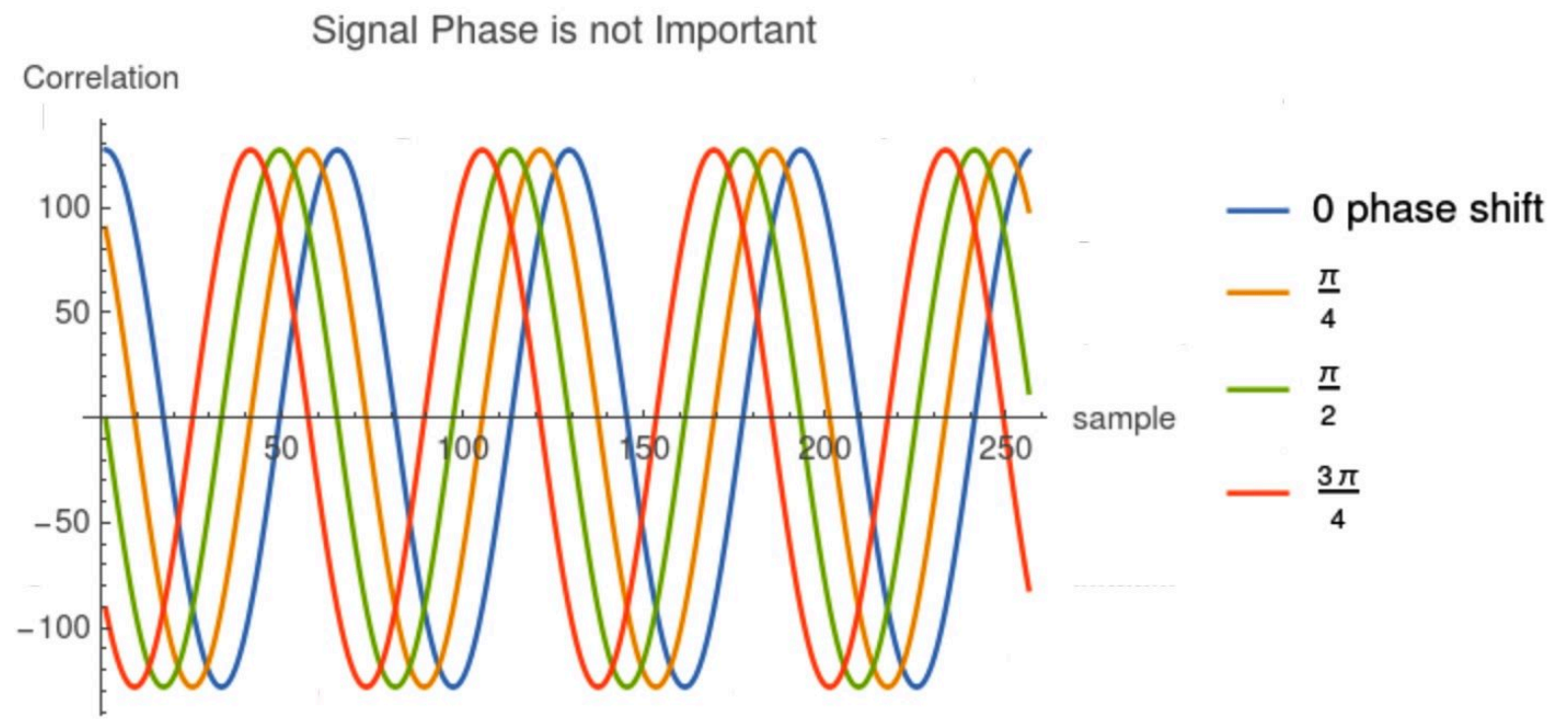



```

signal1 = Table[Sin[2  $\pi$  f1 n dt], {n, nos}]; (*signal1*)
signal2 = Table[Sin[2  $\pi$  f2 n dt +  $\frac{\pi}{4}$ ], {n, nos}]; (*signal2*)
signal3 = Table[Sin[2  $\pi$  f3 n dt +  $\frac{\pi}{2}$ ], {n, nos}]; (*signal3*)
signal4 = Table[Sin[2  $\pi$  f4 n dt +  $\frac{3\pi}{4}$ ], {n, nos}]; (*signal4*)

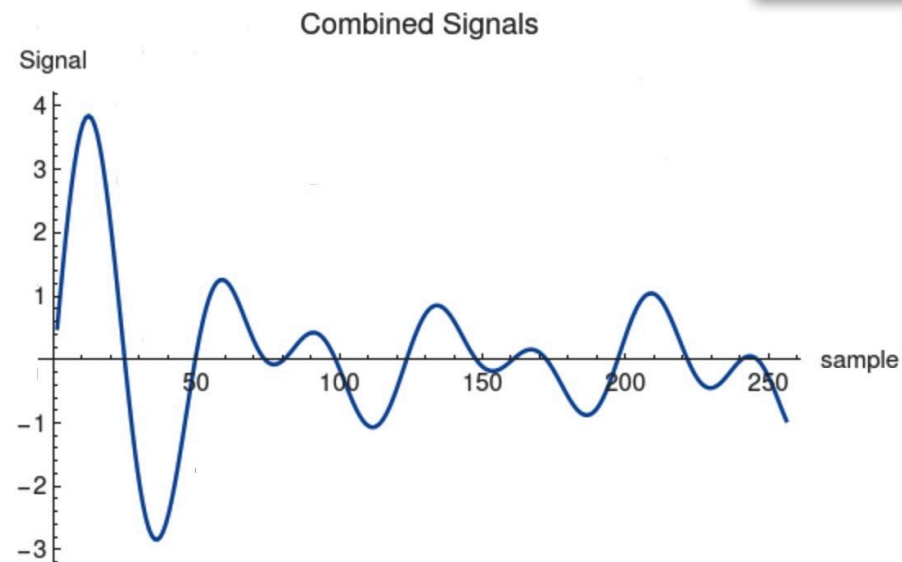
```

A signal will always auto-correlate even if one of them has a different



```
combinedSig = Table[Sin[2  $\pi$  f1 n dt] + Sin[2  $\pi$  f2 n dt] + Sin[2  $\pi$  f3 n dt] + Sin[2  $\pi$  f4 n dt],
  {n, nos}];
ListLinePlot[combinedSig,
  AxesLabel → {"sample", "Signal"},
  PlotLabel → "Combined Signals"]
```

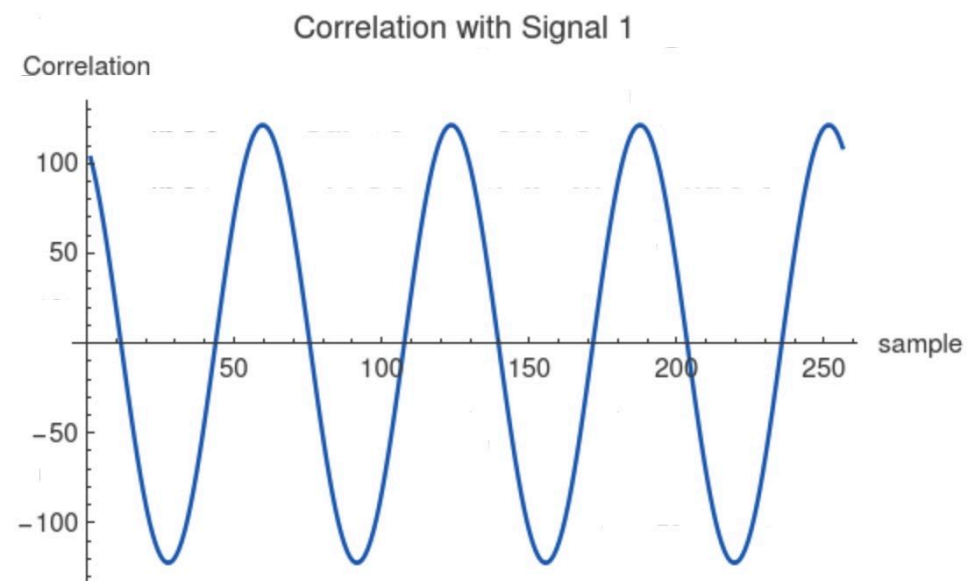
A complex signal



Signal 1 (or any other signal) can be extracted from this complex waveform, if correlated with the correct signal.

Correlation can be used to support CDMA. Each channel uses the same frequency, but a different digital modulation code.

```
combinedCorSig1 = ListCorrelate[signal1, combinedSig, {1, 1}];
ListLinePlot[combinedCorSig1,
  AxesLabel → {"sample", "Correlation"},
  PlotLabel → "Correlation with Signal 1"]
```

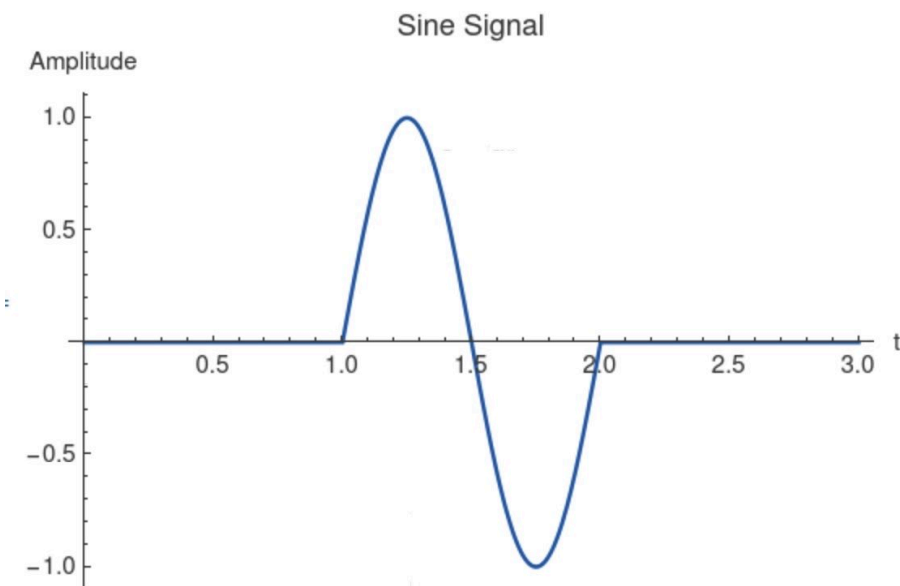


Correlation & Noise

```
Clear[na, sinSignal, sinSignalTable, sinReferenceTable,
value]
```

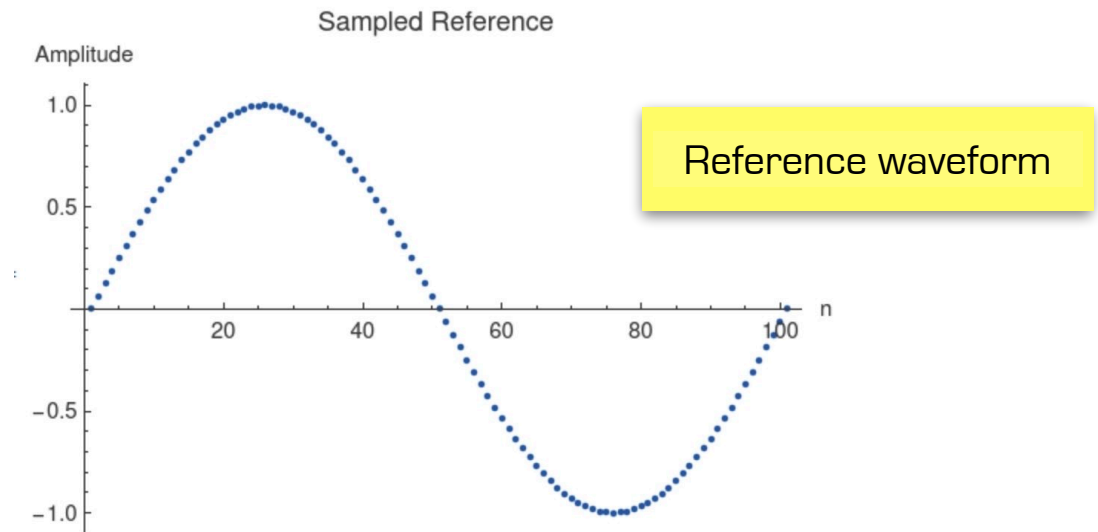
```
dt = .01; (*time increment*)
f = 1; (*frequency or number of cycles*)
tw = 1; (*time window normalized to 1*)
```

```
sinSignal[t_] := If[1 ≤ t ≤ 2, Sin[2 π f t], 0];
Plot[sinSignal[t], {t, 0, 3 tw},
PlotRange → All,
PlotLabel → "Sine Signal",
AxesLabel → {"t", "Amplitude"}]
```

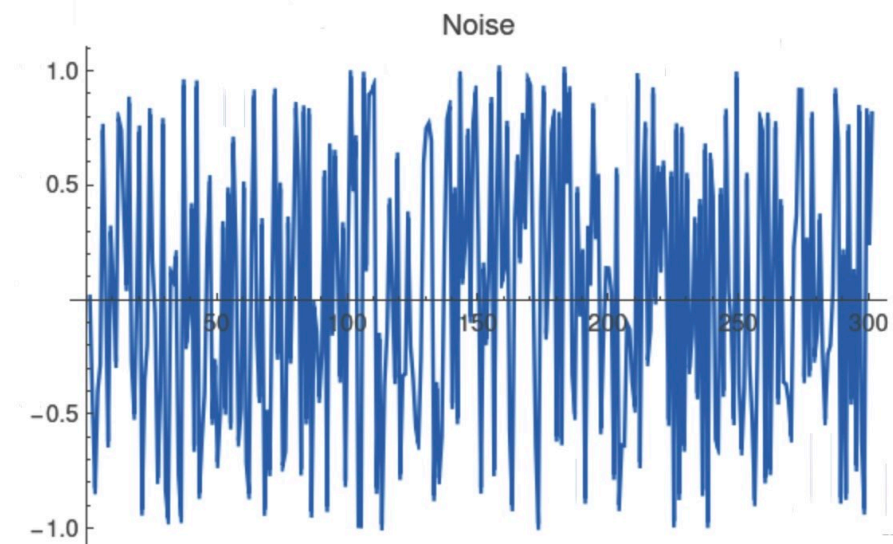


To observe the entire correlation process, the signal is placed in middle so that the reference can be placed on either side.

```
sinReferenceTable = Table[sinSignal[t], {t, 1, 2, dt}];
ListPlot[sinReferenceTable,
PlotLabel → "Sampled Reference",
AxesLabel → {"n", "Amplitude"}]
```



```
noise = Table[RandomReal[{-1, 1}],
{x, Length[sinSignalTable]}];
ListLinePlot[noise,
PlotLabel → "Noise"]
```




```

Manipulate[
  Grid[{{
    Show[ListPlot[ListCorrelate[sinReferenceTable, na noise + sinSignalTable, 1, 0],
      PlotRange → All,
      ImageSize → Medium,
      GridLines → Automatic,
      AxesLabel → {"sample", "Correlation Result & Value"}],
    value = ListCorrelate[sinReferenceTable, na noise + sinSignalTable, 1, 0][[offset]];
    Graphics[{PointSize[0.02], Point[{offset, value}]}],
    Epilog →
      Text[Row[{"Correlation = ", value}], Scaled[{0.6, 0.9}], Left]] },

    noisySignal = na noise + sinSignalTable;
    offsetReference = Table[sinSignal[(t - (offset - 1) dt)], {t, 1, 4, dt}];

    {Show[ListPlot[{noisySignal, offsetReference},
      PlotRange → All,
      ImageSize → Medium,
      GridLines → Automatic,
      Joined → {True, True},
      Filling → {2 -> Axis},
      AxesLabel → {"sample", "Reference & Signal"}]]}],

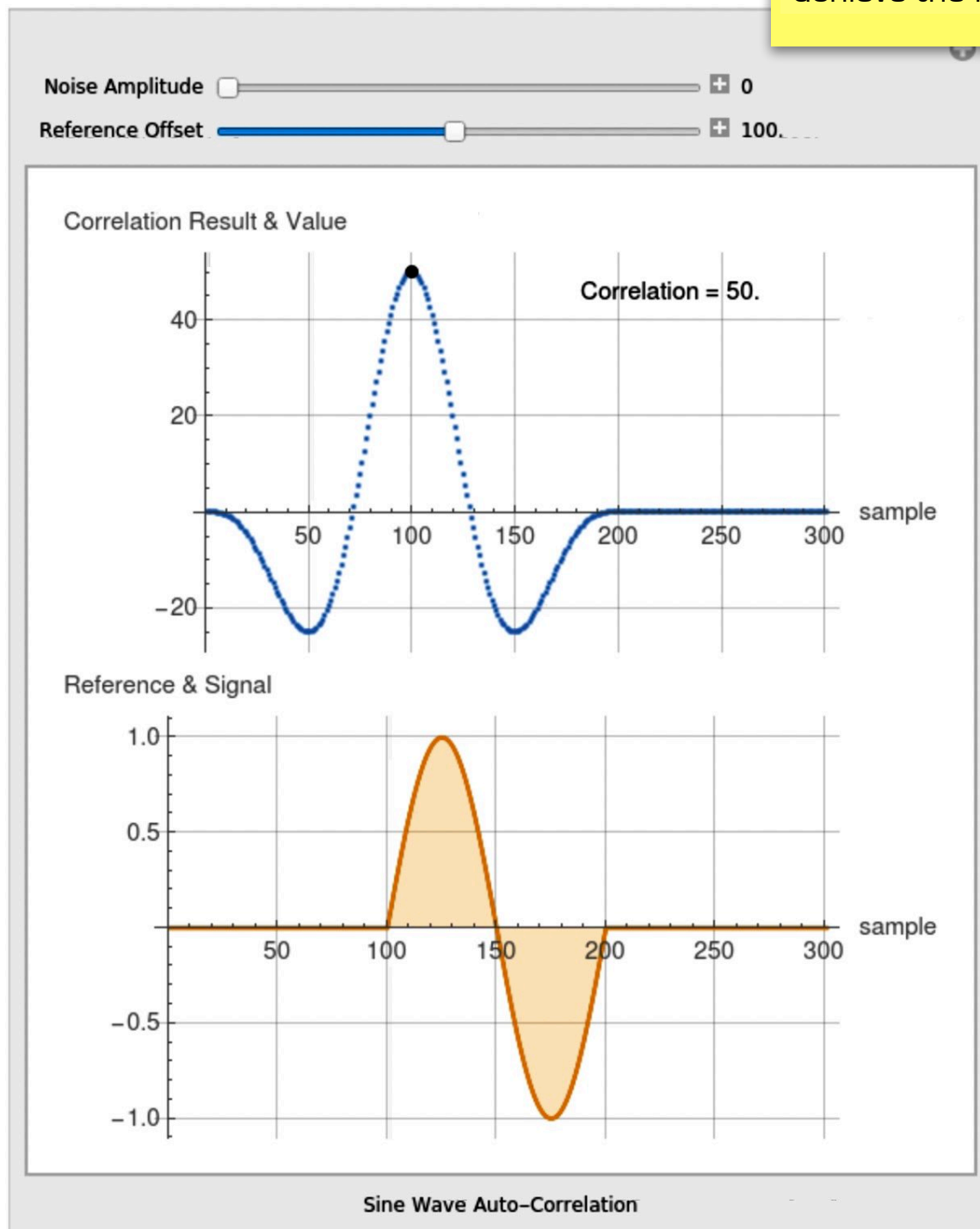
    {{na, 0, "Noise Amplitude"}, 0, 5, Appearance → "Labeled"},
    {{offset,  $\frac{1}{dt}$ , "Reference Offset"}, 1,  $\frac{2}{dt}$ , 1, Appearance → "Labeled"},
    SaveDefinitions → True, ControlPlacement → Top,
    FrameLabel → "Sine Wave Auto-Correlation"]

```

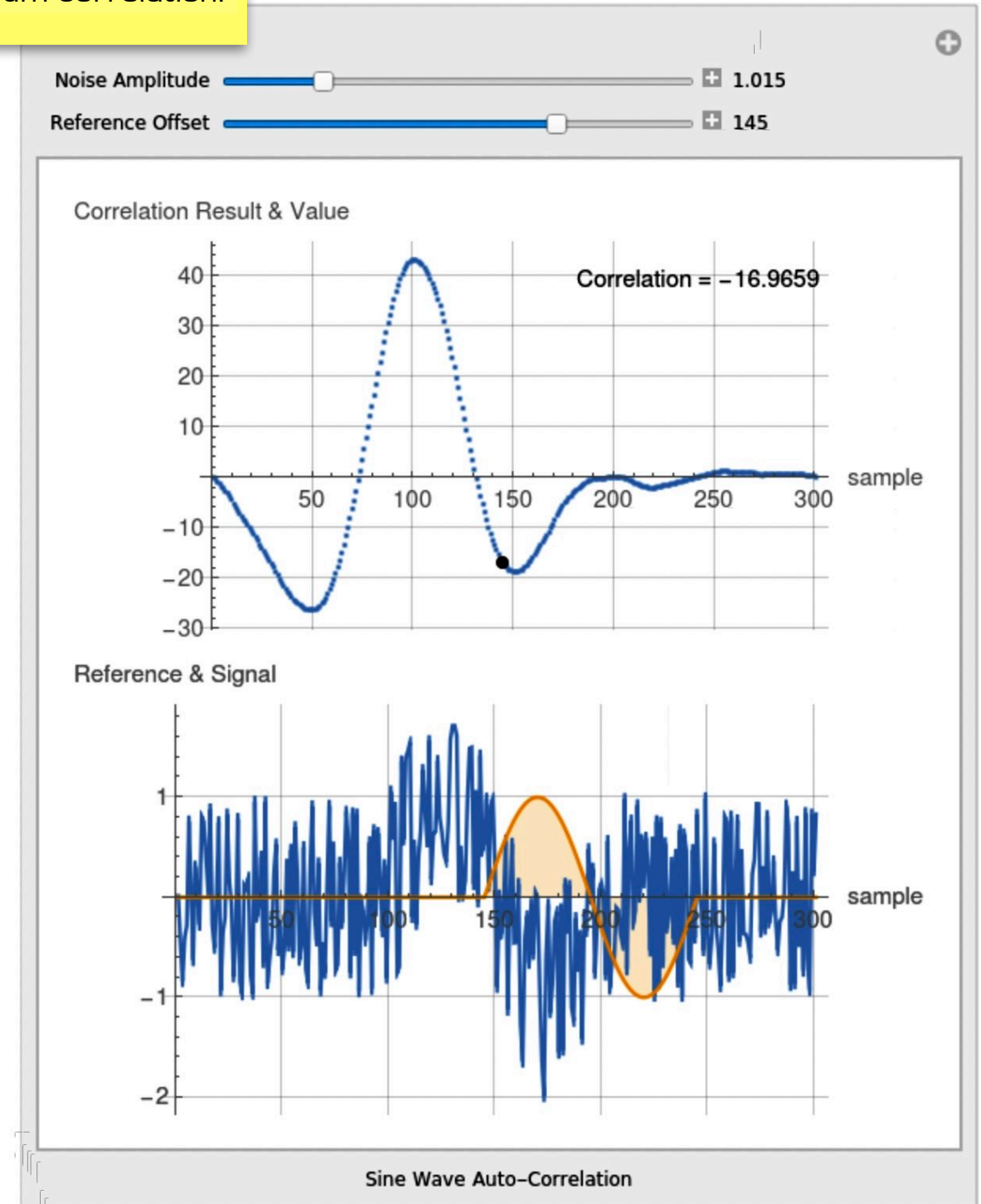
Plot the correlation.

Position the reference with
respect to the signal

Adjust the reference offset to achieve the maximum correlation.



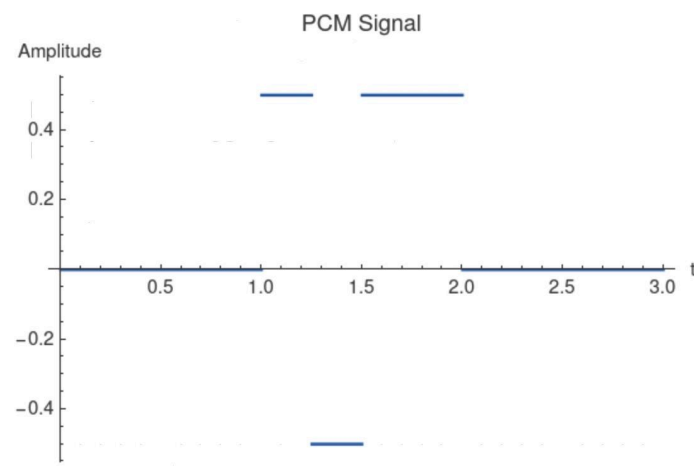
Maximum correlation occurs when the signal and reference are exactly aligned.



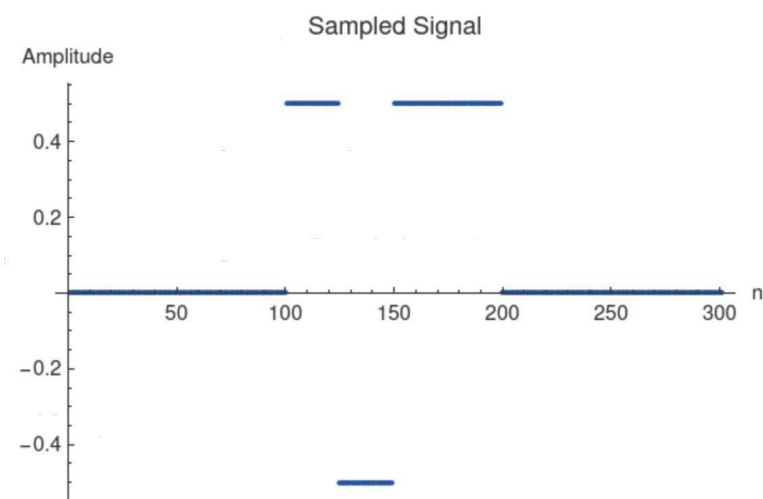
Even in the presence of extreme noise, maximum correlation occurs when the signal and reference are exactly aligned.

Correlation & Digital Signals

```
pcmSignal[t_] := Piecewise[{{0, t < 1}, {0.5, t < 1.25},
  {-0.5, t < 1.5}, {0.5, t < 2}, {0, t >= 2}}]
Plot[pcmSignal[t], {t, 0, 3 tw},
  PlotRange -> All,
  PlotLabel -> "PCM Signal",
  AxesLabel -> {"t", "Amplitude"}]
```

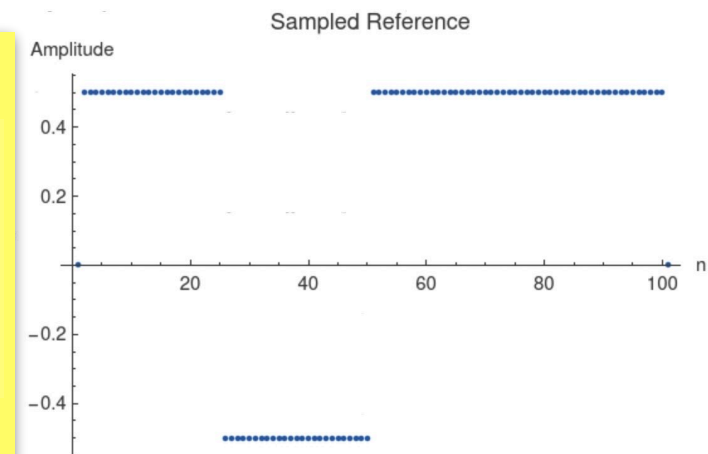


```
pcmSignalTable = Table[pcmSignal[t + dt], {t, 0, 3, dt}];
ListPlot[pcmSignalTable,
  PlotLabel -> "Sampled Signal",
  AxesLabel -> {"n", "Amplitude"}]
```

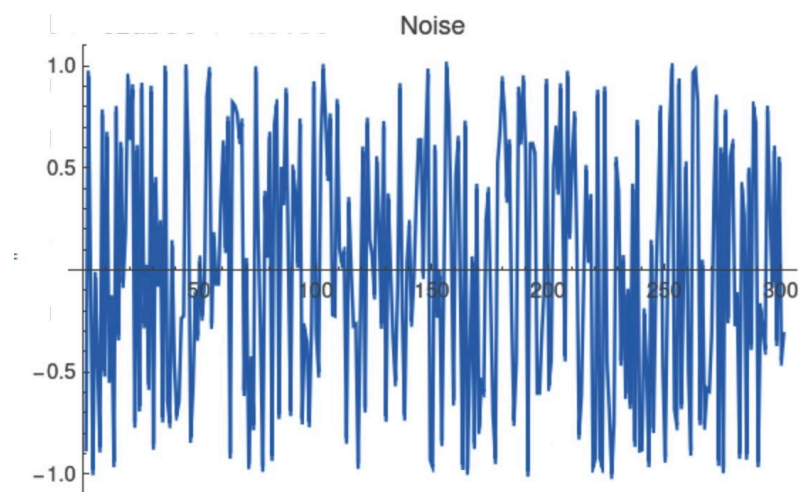


CDMA replaces binary bits with code chips. Correlation can then be used to separate multiple channels which use the same carrier frequency.

```
pcmReferenceTable = Table[pcmSignal[t], {t, 1, 2, dt}];
ListPlot[pcmReferenceTable,
  PlotLabel -> "Sampled Reference",
  AxesLabel -> {"n", "Amplitude"}]
```



```
noise = Table[RandomReal[{-1, 1}], {x, Length[pcmSignalTable]}];
ListLinePlot[noise,
  PlotLabel -> "Noise"]
```




```

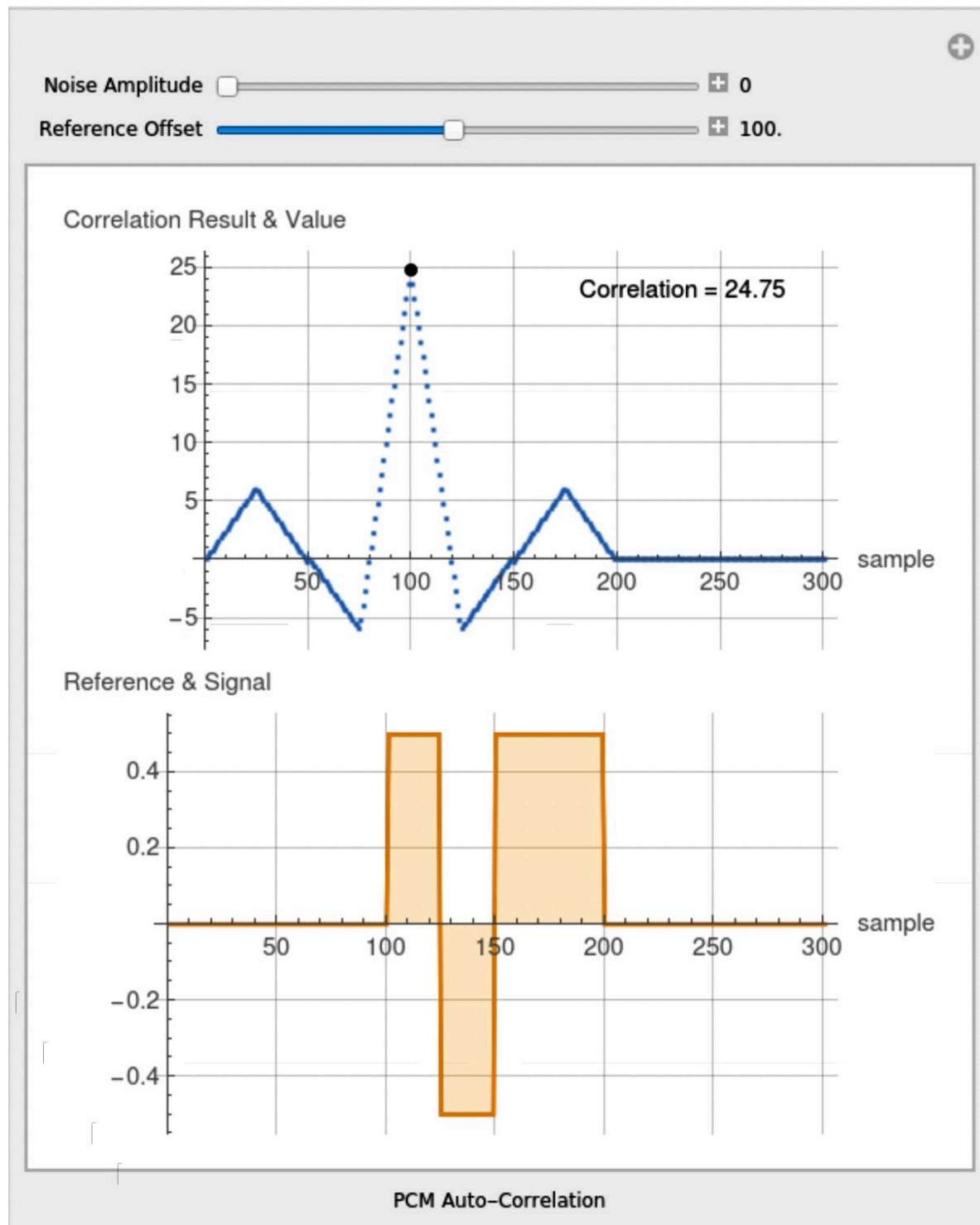
Manipulate[
  Grid[{{
    Show[ListPlot[ListCorrelate[pcmReferenceTable, na noise + pcmSignalTable, 1, 0],
      PlotRange → All,
      ImageSize → Medium,
      GridLines → Automatic,
      AxesLabel → {"sample", "Correlation Result & Value"}],
    value = ListCorrelate[pcmReferenceTable, na noise + pcmSignalTable, 1, 0][[offset]];
    Graphics[{PointSize[0.02], Point[{offset, value}]}],
    Epilog →
      Text[Row[{"Correlation = ", value}], Scaled[{0.6, 0.9}], Left]]},

    noisySignal = na noise + pcmSignalTable;
    offsetReference = Table[pcmSignal[(t - (offset - 1) dt)], {t, 1, 4, dt}];

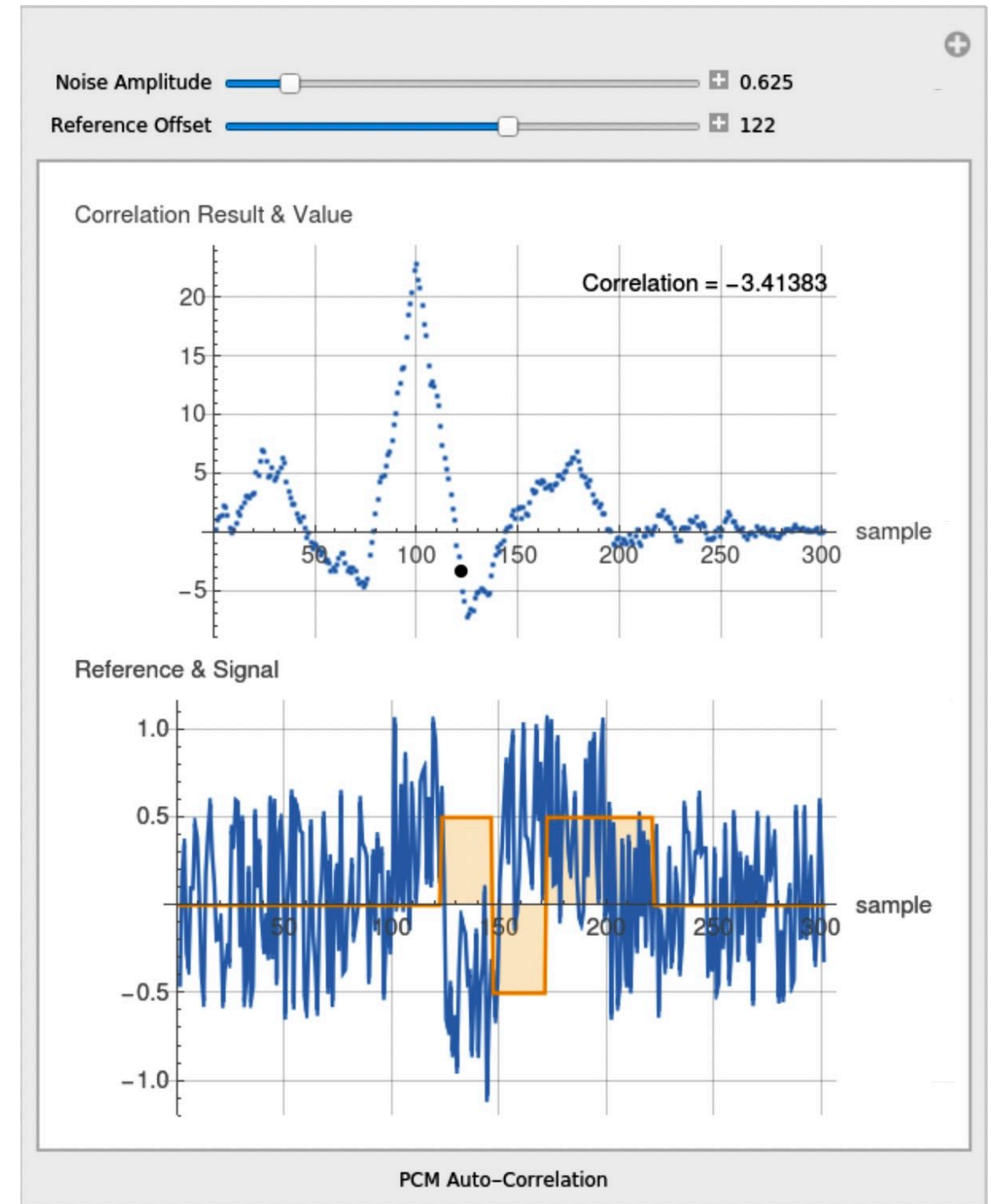
    {Show[ListPlot[{noisySignal, offsetReference},
      PlotRange → All,
      ImageSize → Medium,
      GridLines → Automatic,
      Joined → {True, True},
      Filling → {2 -> Axis},
      AxesLabel → {"sample", "Reference & Signal"}]]}],

    {{na, 0, "Noise Amplitude"}, 0, 5, Appearance → "Labeled"},
    {{offset,  $\frac{1}{dt}$ , "Reference Offset"}, 1,  $\frac{2}{dt}$ , 1, Appearance → "Labeled"}},
    SaveDefinitions → True, ControlPlacement → Top,
    FrameLabel → "PCM Auto-Correlation"]

```



Digital signals can be identified in the presence of noise, if the chip code is known



[CDMA Overview](#)
[CDMA by Langton & Levin](#)

Animation

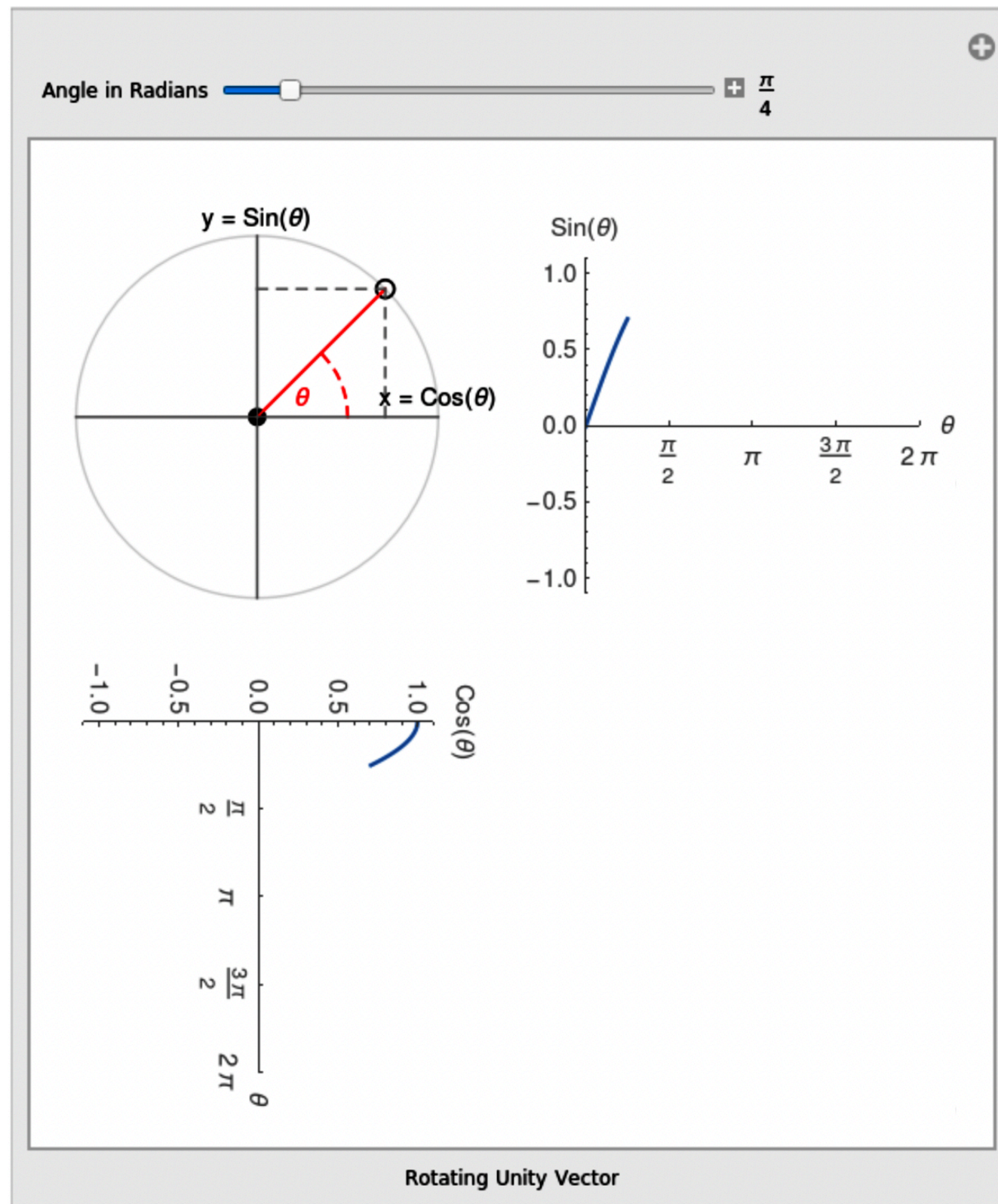
Rotating Unity Vector

```
Manipulate[
GraphicsGrid[
{{Graphics[
(* Row 1 *)
{(* Column 1 Graphic - A Circle *)
{LightGray, Circle[{0, 0}, 1]}, (* large circle *)
PlotRange -> {{-1.5, 1.5}, {-1.5, 1.5}},
AspectRatio -> 1,

Disk[{0, 0}, 0.05], (* small black circle in the center *)
Circle[{Cos[ $\theta$ ], Sin[ $\theta$ ]}, 0.05], (* circle at the end of the vector *)
{Red, Line[{0, 0}, {Cos[ $\theta$ ], Sin[ $\theta$ ]}}] (* radius line *),
{Gray, Dashed, Line[{Cos[ $\theta$ ], 0}, {Cos[ $\theta$ ], Sin[ $\theta$ ]}}] (* Cosine *),
{Gray, Dashed, Line[{0, Sin[ $\theta$ ]}, {Cos[ $\theta$ ], Sin[ $\theta$ ]}}] (* Sine *),
{Red, Dashed, Circle[{0, 0}, 0.5, {0,  $\theta$ ]}} (* angle *),
{Red, Text[" $\theta$ ", {0.25, 0.1}]},

{Gray, Line[{0, 1}, {0, -1}]},
{Black, Text["y = Sin( $\theta$ )", {0, 1.1}]},

{Gray, Line[{-1, 0}, {1, 0}]},
{Black, Text["x = Cos( $\theta$ )", {1, 0.1}]}
]},
(* Column 2 Graphic - Horizontal Sine Wave *)
Plot[Sin[x], {x, 0,  $\theta$ },
PlotRange -> {{0, 2  $\pi$ }, {-1.1, 1.1}},
Ticks -> {{0,  $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ , 2  $\pi$ }, Automatic},
AspectRatio -> 1, AxesLabel -> {" $\theta$ ", "Sin( $\theta$ )"}
]},
(* Row 2, Column 1 Graphic - Vertical Cosine Wave *)
{Rotate
[Plot[Cos[x], {x, 0,  $\theta$ },
PlotRange -> {{0, 2  $\pi$ }, {-1.1, 1.1}},
Ticks -> {{0,  $\frac{\pi}{2}$ ,  $\pi$ ,  $\frac{3\pi}{2}$ , 2  $\pi$ }, Automatic},
AspectRatio -> 1, AxesLabel -> {" $\theta$ ", "Cos( $\theta$ )"}
], - $\pi/2$ ]}],
{{ $\theta$ , Pi/4, "Angle in Radians"}, 0.001, 2  $\pi$ , Appearance -> "Labeled"},
(* a small initial value of  $\theta$  is need, otherwise the plot will fail for  $\theta = 0$  *)
FrameLabel -> "Rotating Unity Vector"]
```

To run the animation, click on the + sign at the end of the Angle slider.

A Simple Pendulum

```
Clear[t, θ, θ0, l, ω, T, m, g]
```

```
g = 9.80665;
```

```
m = 1;
```

```
Manipulate[
```

```
ω =  $\sqrt{\frac{g}{l}}$ ; (* angular frequency *)
```

```
T =  $\frac{2\pi}{\omega}$ ; (* period *)
```

```
θ = θ0  $\frac{\pi}{180}$  Cos[ω t]; (* angle *)
```

```
θd = θ  $\frac{180}{\pi}$ ; (* angle in degrees *)
```

```
v1 := m g Sin[θ]; (* force tangential to movement *)
```

```
v2 := m g Cos[θ]; (* force perpendicular to movement *)
```

```
av =  $\sqrt{\frac{2g}{l} (\text{Cos}[\theta] - \text{Cos}[\theta0 \text{ Degree}])}$ ; (* angular velocity *)
```

This example only works for small angles $\leq 10^\circ$.

The equations are more complex for large angles.

Defining equations

```

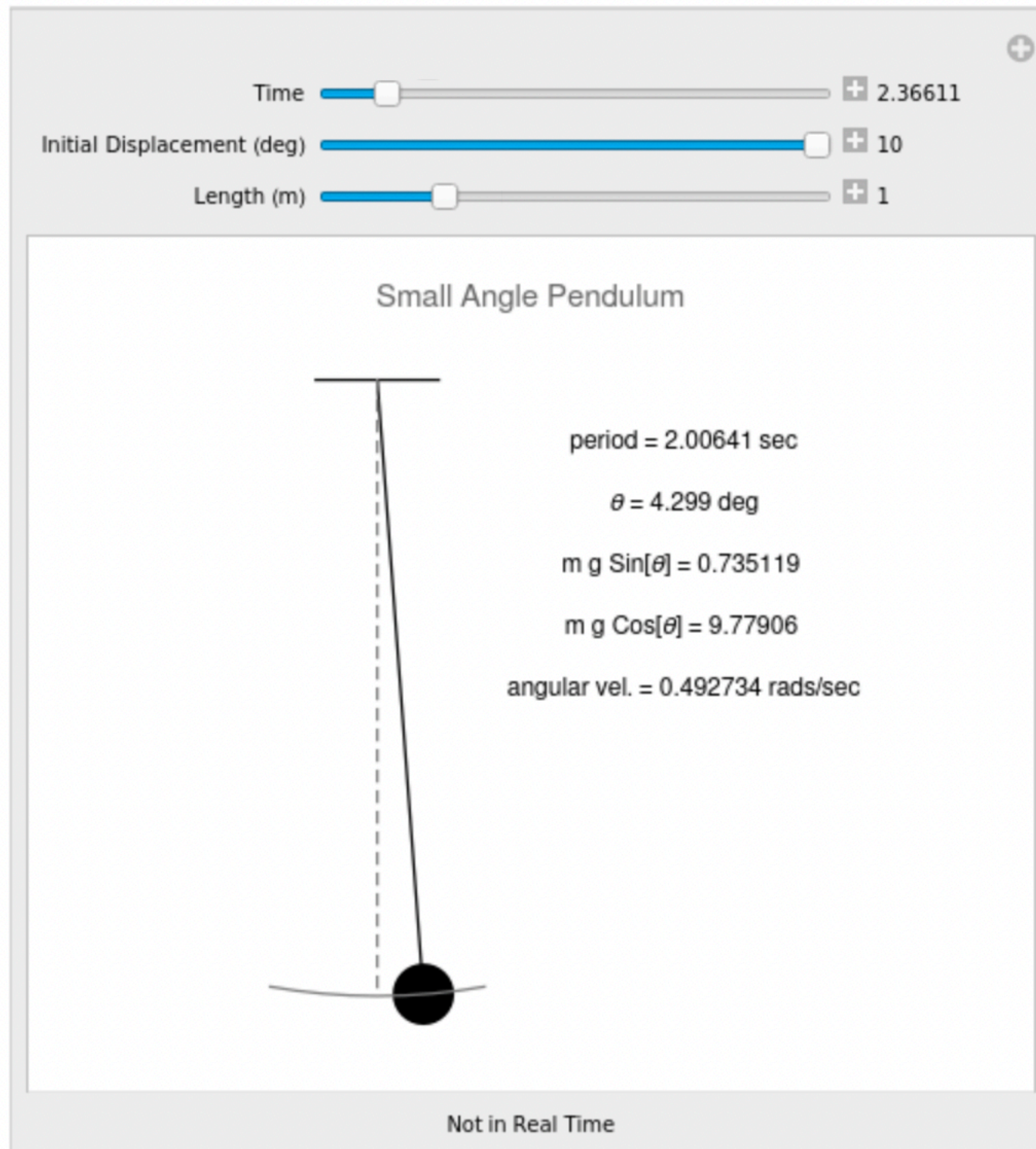
Graphics[
{
  Line[{{-0.1, 0}, {0.1, 0}}] (* Ceiling *),
  Line[{{0, 0}, {l Sin[θ], -l Cos[θ]}}], (* arm *)
  Disk[{l Sin[θ], -l Cos[θ]}, 0.05], (* bob *)
  {Gray, Dashed, Line[{{0, 0}, {0, -l}]} (* vertical Line *),
  {Gray, Circle[{0, 0}, l, { $\frac{-100\pi}{180}$ ,  $\frac{-80\pi}{180}$ }}] (* pendulum path *),
  Text[Row[{"period = ", T, " sec"}], {0.5, -0.1}],
  Text[Row[{"θ = ", θd, " deg"}], {0.5, -0.2}],
  Text[Row[{"m g Sin[θ] = ", v1, " "}], {0.5, -0.3}],
  Text[Row[{"m g Cos[θ] = ", v2, " "}], {0.5, -0.4}],
  Text[Row[{"angular vel. = ", av, " rads/sec"}], {0.5, -0.5}]
},
PlotRange → {{-0.5, 1}, {-1.1 l, 0.1}},
PlotLabel → "Small Angle Pendulum"],

{{t, 0, "Time"}, 0, 10 T, Appearance → "Labeled"},
{{θ0, 10, "Initial Displacement (deg)"}, -10, 10, 1, Appearance → "Labeled"},
{{l, 1, "Length (m)"}, 0.1, 4, .1, Appearance → "Labeled"},
FrameLabel → "Not in Real Time"
]

```

Drawing the pendulum

Adding computational details



To run the animation, click on the + sign at the end of the Time slider.