Como ajustar dados do osciloscópio no Mathematica

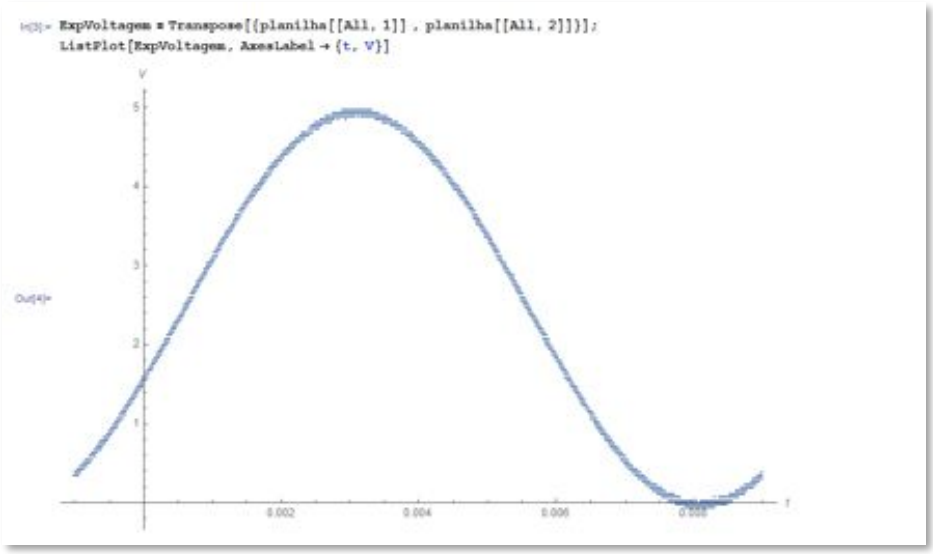
dicas de Julia Veloso de Oliveira, 2016 - aluna de IC do Prof. Cláudio Lenz

1. Importando o arquivo do osciloscópio:

Os arquivos criados pelo osciloscópio vêm no formato `.ods` e consistem de uma espécie de planilha com 50000 linhas e três colunas, referentes ao tempo, à voltagem e à transmissão. Contudo, as duas primeiras linhas contém caracteres não numéricos que geram problemas ao plotar gráficos. Por isso, após abrir o arquivo com a função `Import`, é preciso excluir as primeiras linhas com o `Drop`.

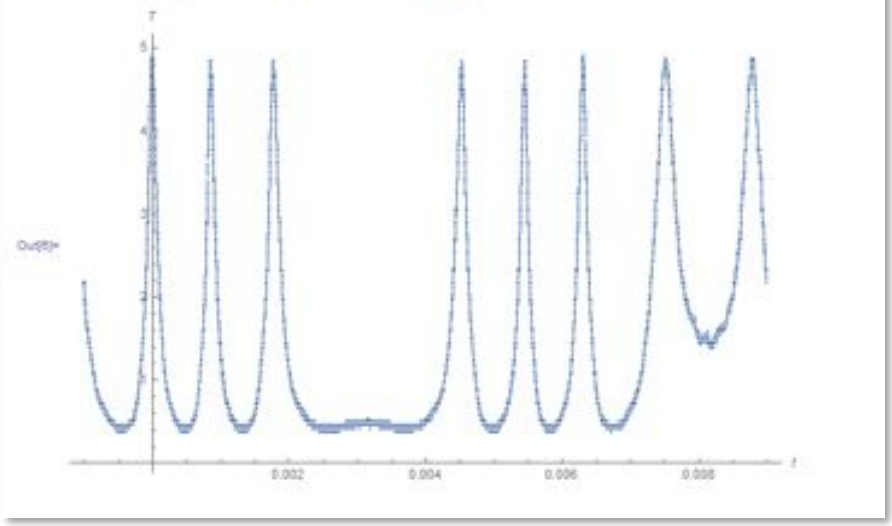
2. Visualizando gráficos experimentais:

Usando as funções `Transpose` para associar dois pontos em uma mesma linha da tabela e `ListPlot`, pode-se traçar os gráficos experimentais da voltagem e da transmissão. A opção `AxesLabel` é opcional.



```
In[8]:= ExpTransmissao = Transpose[{planilha[[All, 1]], planilha[[All, 3]]}];
        ListPlot[ExpTransmissao, AxesLabel → {t, T}]
```

Out[8]=



3. Manipulando gráficos teóricos:

Antes de trabalhar com os gráficos teóricos, é bastante útil definir as fórmulas no Mathematica (vide 1.9.). Para inserir símbolos como ν, δ e ϕ, basta digitar `Esc` + `nome da letra (em inglês)` + `Esc`.

```
In[7]:= vfit[o_, a_, ν_, t_, ϕ_] := o + a * Sin[2 * π * ν * t + ϕ]
```

```
In[33]:= tfit[o_, a1_, F_, a2_, ν_, ϕ1_, ϕ2_, t_] := o + a1 * (1 / (1 + F * (Sin[π * a2 * Sin[2 * π * ν * t + ϕ2] + ϕ1])^2));

         tfit[o, a1, F, a2, ν, ϕ1, ϕ2, t]

Out[34]= o + a1 / (1 + F Sin[ϕ1 + a2 π Sin[2 π t ν + ϕ2]]^2)
```

```
In[152]:= ttfit[R_, A_, ν_, t_, ϕ1_, ϕ2_] := 1 / (1 + R^2 + 2 * R * Cos[2 * π * A * Cos[2 * π * ν * t + ϕ2] + ϕ1])
          ttfit[R, A, ν, t, ϕ1, ϕ2]

Out[153]= 1 / (1 + R^2 + 2 R Cos[ϕ1 + 2 A π Cos[2 π t ν + ϕ2]])
```

```
:nu
```

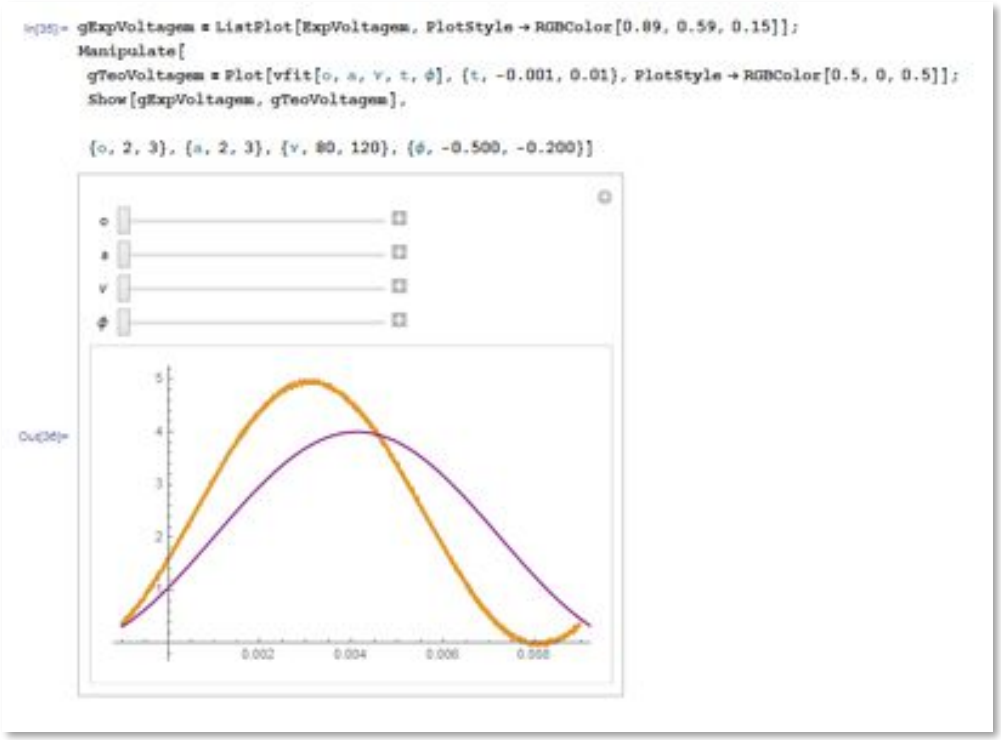
ν	nu	\[Nu]
	null	\[Null]

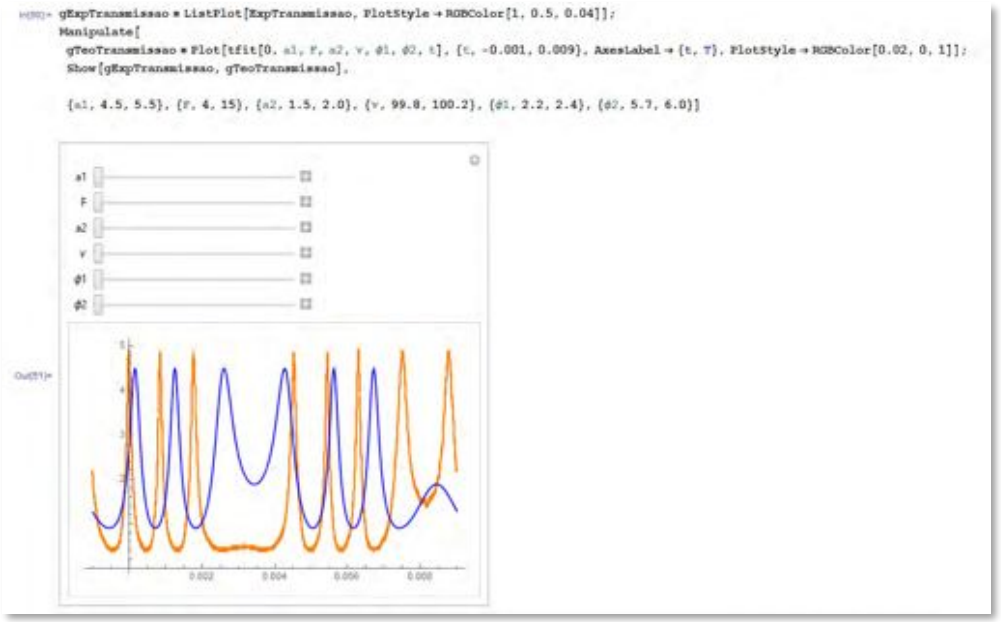
```
:ph
```

ϕ	phi	\[Phi]

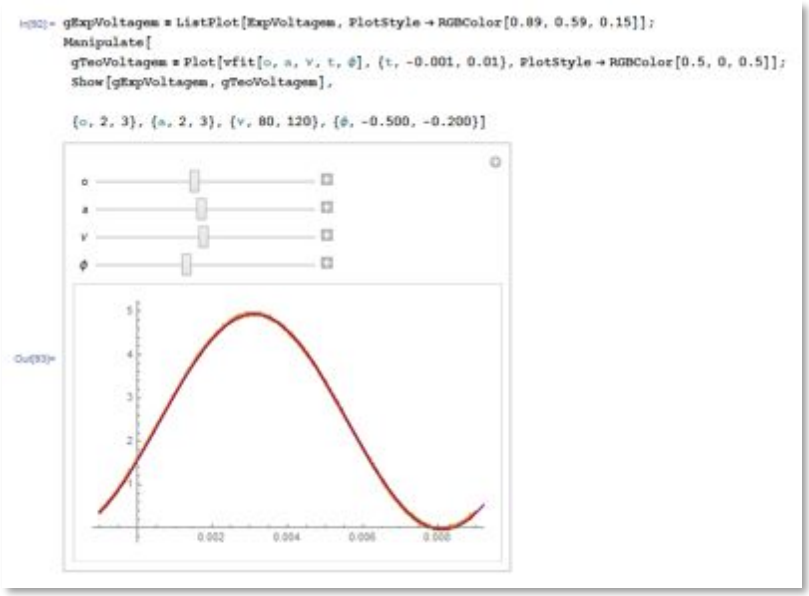
A fórmula `vfit`, uma simples senóide, descreve a variação da voltagem em função do tempo e possui 4 parâmetros; já as funções `tfit` e `ttfit` expressam a transmissão do interferômetro também em função do tempo e possuem 7 e 5 parâmetros, respectivamente.

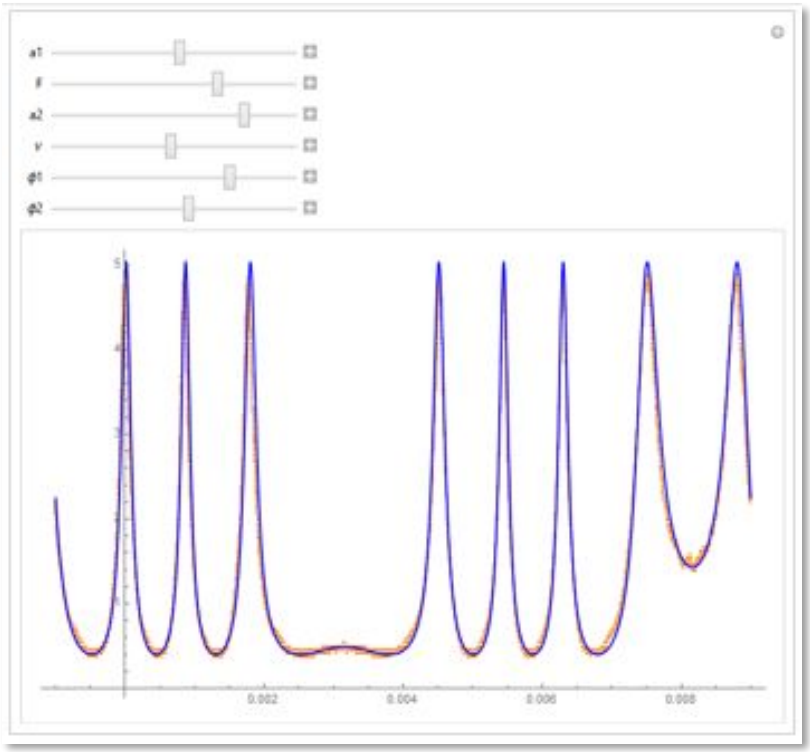
Em seguida, usamos o `Manipulate` para ajustar manualmente os parâmetros teóricos, sendo necessário estimar intervalos de variação para cada parâmetro. Note que as funções `Show` e `PlotStyle` também foram utilizadas para visualizar os gráficos teóricos e experimentais juntos, o que facilita o ajuste.





Após uma série de tentativas, é possível aproximar o gráfico teórico do experimental; os valores obtidos para os parâmetros serão úteis na hora de usar a função `FindFit`. Além disso, esta etapa também é crucial para entender como cada parâmetro influencia o gráfico da função.





Para que o ajuste manual não seja perdido ao reiniciar o programa, podemos definir os números encontrados como os valores iniciais da função, como no exemplo abaixo para `vfit`.

```
{{a1, 5.03}, 4.5, 5.5}, {{F, 11.62}, 4, 15}, {{a2, 1.902}, 1.5, 2.0}, {{ν, 99.997}, 99.8, 100.2}, {{ϕ1, 2.349}, 2.2, 2.4}, {{ϕ2, 5.87}, 5.7, 6.0}]
```

4. Encontrando o melhor ajuste para as fórmulas teóricas:

Por fim, podemos usar a função `FindFit` para encontrar o melhor ajuste matemático para os parâmetros de cada função. Quando há poucos parâmetros, o Mathematica é capaz de ajustá-los facilmente - como é o caso da função `vfit`.

```
In[123]:= fitVoltagem = FindFit[ExpVoltagem, vfit[o, a, ν, t, ϕ], {o, a, ν, ϕ}, t]
Out[123]= {o → 2.44903, a → 2.48389, ν → -99.9012, ϕ → 3.51885}
```

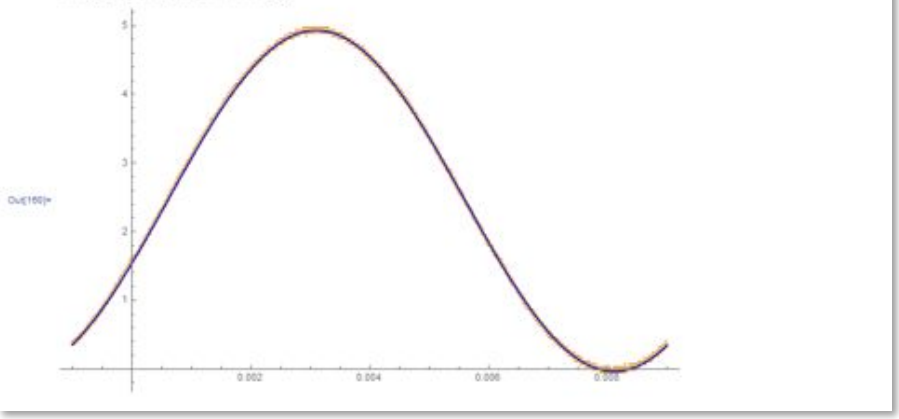
No entanto, quando há mais parâmetros e a função é mais complexa, o Mathematica não consegue achar precisamente o melhor ajuste a menos que sejam fornecidos intervalos para cada parâmetro.

```
In[136]:= fitTransmissao = FindFit[ExpTransmissao, tfit[o, a1, F, a2, ν, ϕ1, ϕ2, t], {o, a1, F, a2, ν, ϕ1, ϕ2}, t]
          FindFit::cvmit : Failed to converge to the requested accuracy or precision within 100 iterations. >>
Out[136]= {o → -0.628174, a1 → 44.1173, F → 26.3284, a2 → 0.238412, ν → -39.8671, ϕ1 → 4.35842, ϕ2 → 1.43758}
```

```
In[161]:= fitTransmissao = FindFit[ExpTransmissao, {tfit[0, a1, F, a2, 100, ϕ1, ϕ2, t], {4.5 < a1 < 5.5, 9 < F < 13, 1.7 < a2 < 2.3, 2.2 < ϕ1 < 2.5, 5.5 < ϕ2 < 6.2}},
            {a1, F, a2, ϕ1, ϕ2}, t]
Out[161]= {a1 → 4.69405, F → 10.836, a2 → 1.91147, ϕ1 → 2.37626, ϕ2 → 5.87158}
```

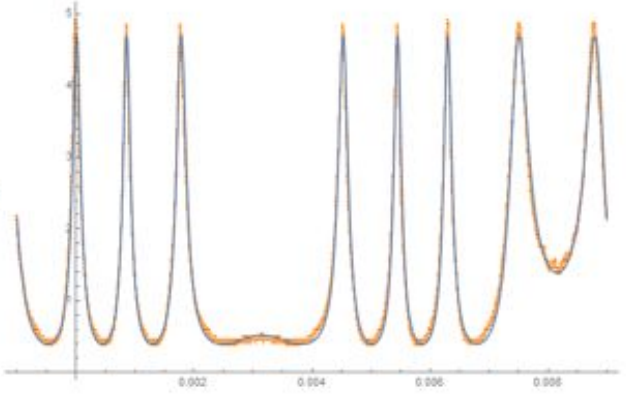
Para analisar o significado desses valores, pode-se plotar os gráficos experimental e teórico juntos com os números encontrados pela `FindFit`, o que pode ser feito rapidamente como nos exemplos abaixos.

```
In[159]:= gfitVoltagem = Plot[vfit[o, a, ν, t, ϕ] /. fitVoltagem, {t, -0.001, 0.009}, PlotStyle → RGBColor[0.02, 0.02, 0.85]];
          Show[gExpVoltagem, gfitVoltagem]
```



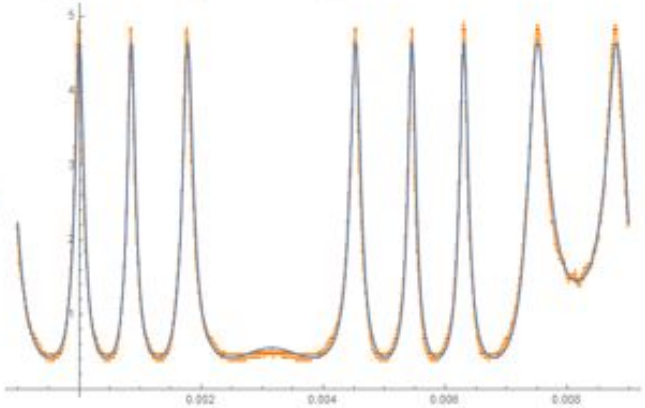
```
In[162]:= gfitTransmissao = Plot[tfit[0, a1, F, a2, 100, ϕ1, ϕ2, t] /. fitTransmissao, {t, -0.001, 0.009}];
          gExpTransmissao = ListPlot[ExpTransmissao, PlotStyle → RGBColor[1, 0.5, 0.04]];
          Show[gExpTransmissao, gfitTransmissao]
```

Out[164]=



```
In[166]:= gfitTransmissao2 = Plot[ttfit[R, A, 100, t, ϕ1, ϕ2] /. fitTransmissao2, {t, -0.001, 0.009}];
          Show[gExpTransmissao, gfitTransmissao2]
```

Out[167]=



Caso o ajuste não esteja bom, pode-se voltar ao Manipulate para aproximar ainda mais os dois gráficos e, então, estimar intervalos mais precisos para os parâmetros.

Funções importantes do Mathematica para o ajuste dos dados do osciloscópio

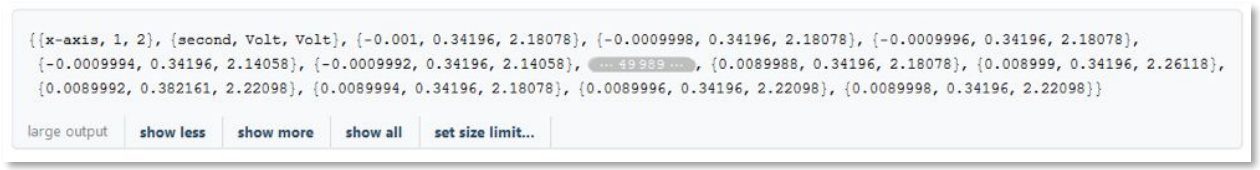
1. `Import[]`

A função `Import["file"]` é de grande utilidade para abrir dados salvos no computador no Mathematica. Para importar um arquivo, é preciso indicar a sua localização entre aspas, conforme mostra o exemplo abaixo.

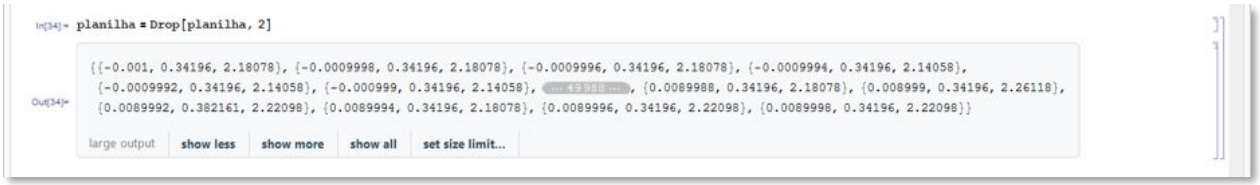
Neste caso, o arquivo "scope_1.csv" foi importado e nomeado "planilha". Cada `{}` indica o conteúdo de uma linha, e as vírgulas dentro das chaves separam as colunas. A planilha acima, por exemplo, possui 50002 linhas e 3 colunas.

2. `Drop[]`

Alguns arquivos possuem dados não numéricos que não conseguem ser lidos pelo Mathematica e podem gerar erros em procedimentos futuros. As duas primeiras linhas da planilha abaixo, por exemplo, contêm palavras que atrapalham a leitura dos dados.



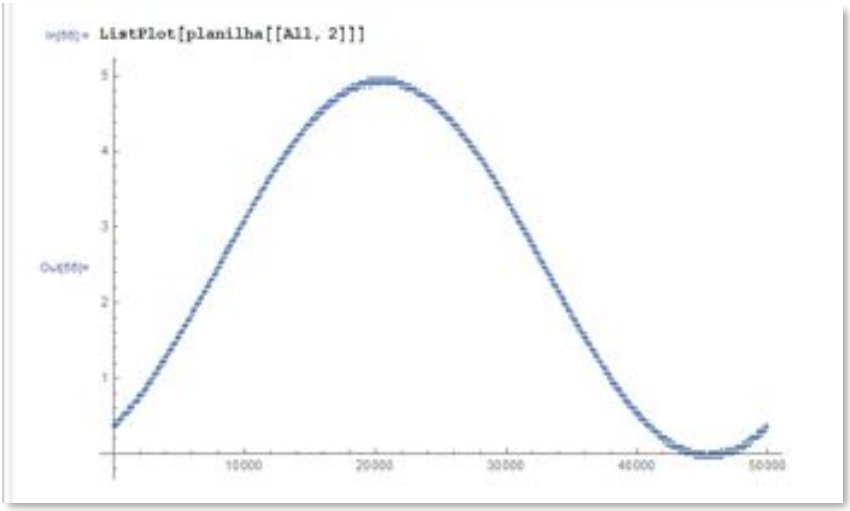
Para eliminá-las, pode-se usar a função `Drop[file,n]`, que apagará as `n` primeiras linhas da versão importada do arquivo `.file`.



Ou seja, ao nome "planilha" foram atribuídos os dados da planilha anterior sem as duas primeiras linhas, como pode ser visto no `Out[34]`.

3. `ListPlot[]`

A função `ListPlot[{y1,y2,...,yn}]` serve para traçar gráficos com pontos definidos - como os pontos dos dados experimentais, por exemplo.



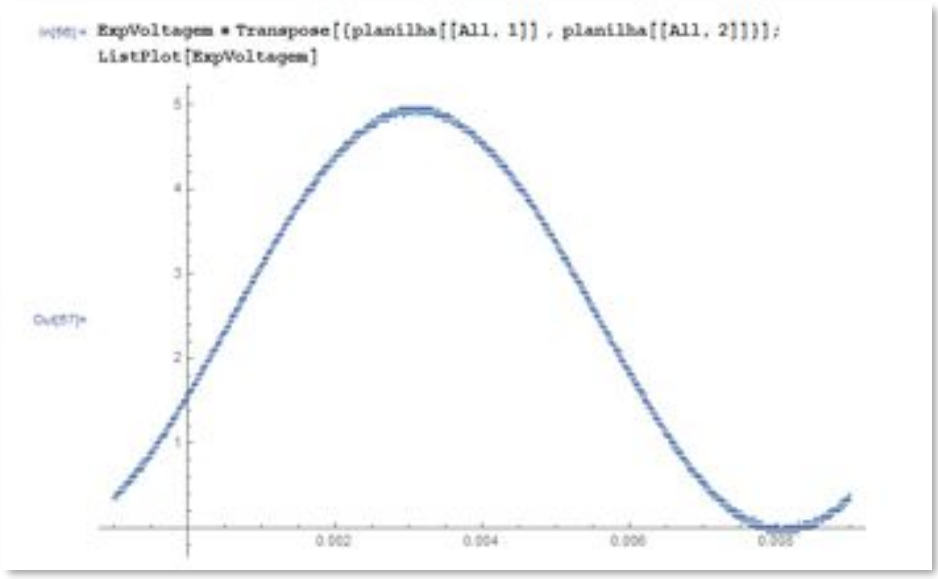
A imagem acima mostra o gráfico dos pontos da segunda coluna da planilha. O comando `[[All,2]]` indica que queremos plotar todos as linhas da coluna 2.

No entanto, caso apenas os valores de y sejam fornecidos, o programa define o eixo x de acordo com o número de pontos; no exemplo acima, como a segunda coluna possuía 50000 linhas, o domínio varia de 1 a 50000. Para definir os pontos de outra coluna como os valores das coordenadas x, pode-se usar a função `Transpose[]`, descrita no próximo tópico.

4. `Transpose[]`

Em geral, a função `Transpose[]` serve para encontrar matrizes transpostas; no nosso caso, iremos usá-la para formatar duas colunas de uma planilha de modo a adequá-las para uso na função `ListPlot`. O comando `Transpose[{coluna1,coluna2}]` gerará um conjunto de pontos `{x,y}` em que, para cada valor de x extraído de uma linha da coluna 1, associa-se um valor de y extraído da mesma linha na coluna 2.

```
In[56]:= ExpVoltagem = Transpose[{planilha[[All, 1]] , planilha[[All, 2]]}];
         ListPlot[ExpVoltagem]
```

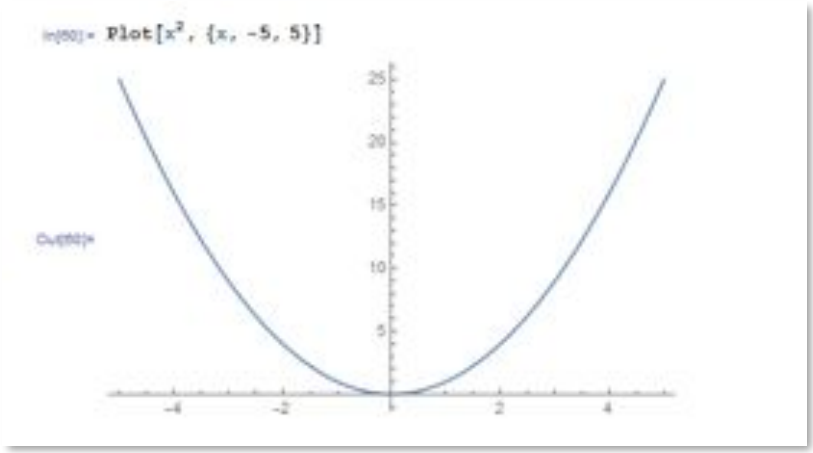


Assim, o gráfico ficou definido no intervalo desejado: `{-0.001,0.009}`.

5. `Plot[]`

A função `Plot[função,{x,x_min,x_max}]` se assemelha fortemente à função `ListPlot`, mas com uma pequena diferença: em vez de plotar um conjunto de pontos, ela deve ser usada para traçar gráficos de uma função em um determinado intervalo.

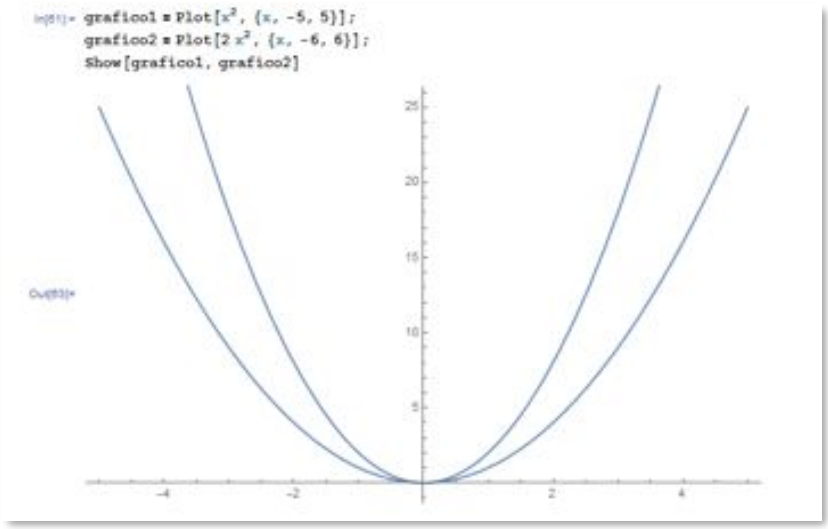
```
In[60]:= Plot[x^2, {x, -5, 5}]
```



No exemplo acima, a função $f(x)=x^2$ foi traçada no intervalo `{-5,5}`.

6. `Show[]`

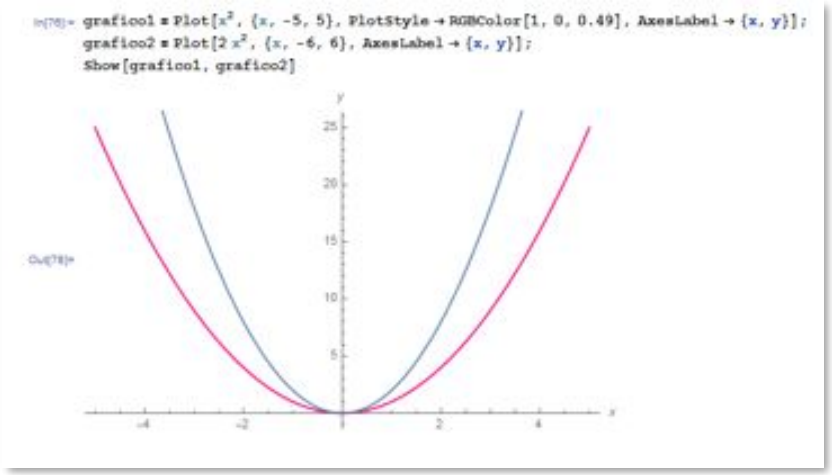
A função `Show[gráfico1,grafico2]` será usada quando for preciso visualizar dois ou mais gráficos juntos.



No exemplo acima, fica difícil distinguir os gráficos de $f(x)=x^2$ e $g(x)=2x^2$ devido à formatação padrão do Mathematica. Para facilitar a visualização, pode-se usar a função descrita no tópico a seguir.

7. PlotStyle,AxesLabel

É possível alterar o estilo dos eixos e dos gráficos de cada função de modo a facilitar a visualização e melhorar a apresentação dos dados. As duas funções mais úteis para personalizar os gráficos são `PlotStyle->RGBColor`, que permite escolher a cor do gráfico, e `AxesLabel->{nome do eixo x,nome do eixo y}`, que permite nomear os eixos.

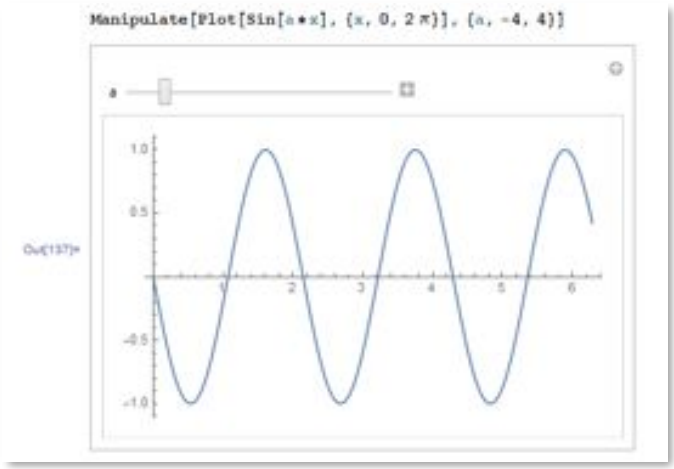


Assim, é possível diferenciar `f(x)` (rosa) de `g(x)` (azul) e ver os nomes escolhidos para os eixos x e y.

Tanto a função `PlotStyle` quanto a `AxesLabel` podem ser usadas com `ListPlot` ou `Plot` e devem ser indicadas dentro dos colchetes entre vírgulas, como mostra a figura acima. A seção de ajuda do Mathematica contém explicações de outras funções de personalização de gráfico.

8. `Manipulate[]`

Com o `Manipulate[função,{domínio},{intervalo de variação dos parâmetros}]`, pode-se ajustar manualmente os parâmetros de uma função. Este recurso é particularmente útil no estudo da influência de cada parâmetro no gráfico da função.

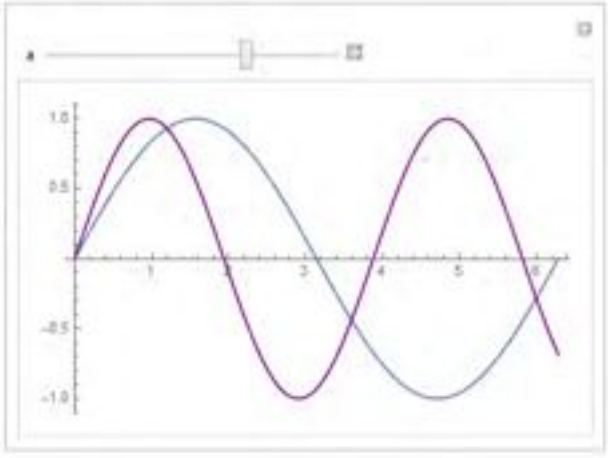


Neste exemplo, a função `f(x)=sen(a*x)` foi plotada com `a` variando no intervalo `{-4,4}`. Ao deslocar o botão, pode-se aumentar ou diminuir o valor de `a` e ver o impacto imediato da mudança no gráfico da função.

Para melhorar a comparação, pode-se também combinar o `Manipulate` com o `Show` para visualizar vários gráficos juntos.

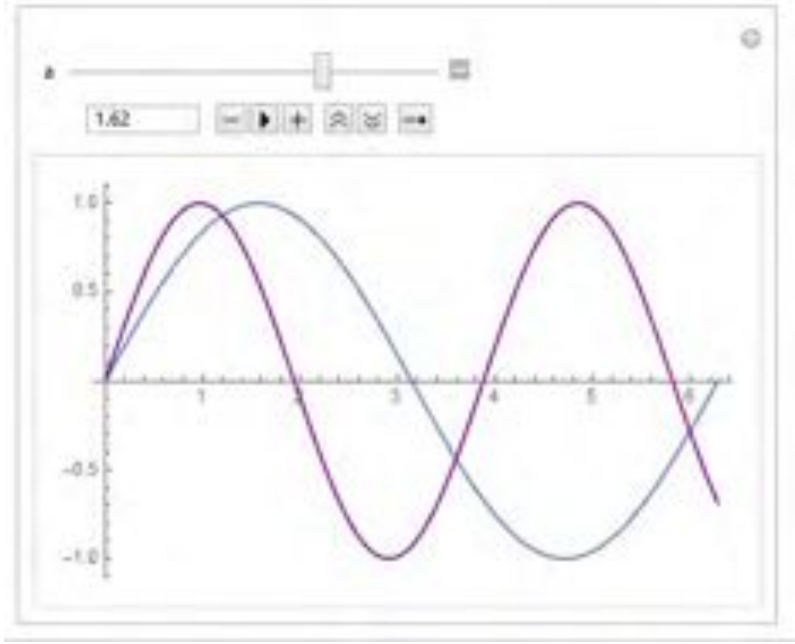
```
In[133]:= referencial = Plot[Sin[x], {x, 0, 2 π}];

          Manipulate[
           deslocamento = Plot[Sin[a * x], {x, 0, 2 π}, PlotStyle → RGBColor[0.5, 0.02, 0.55]];
           Show[referencial, deslocamento],
           {a, -4, 4}]
```

Out[134]=



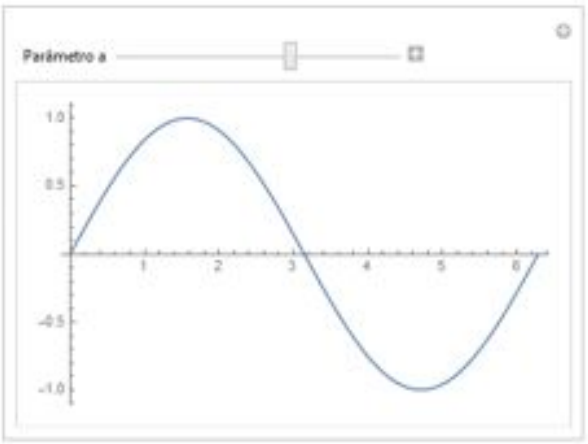
Na imagem acima, o gráfico da função `g(x)=sen(x)` pode ser visto em azul.

O `Manipulate` ainda permite que o usuário veja o valor exato do(s) parâmetro(s) a cada instante. Para isto, basta clicar em cima do sinal + localizado ao lado do botão.



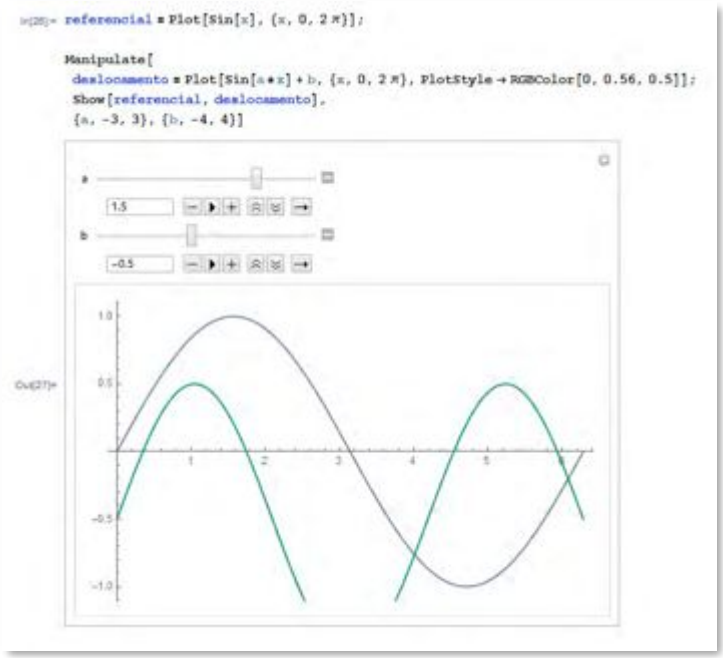
Por fim, pode-se também definir um valor inicial e um nome para o(s) parâmetro(s) da seguinte maneira: $\{\{x, x_0, \text{"nome"}\}, x_{min}, x_{max}\}$. Desse modo, cada vez que o programa for inicializado, o valor do parâmetro será o pré-estabelecido.

```
Manipulate[Plot[Sin[a * x], {x, 0, 2 π}], {{a, 1, "Parâmetro a"}, -4, 4}]
```



Também é possível manipular mais de um parâmetro de uma mesma função; basta declarar o intervalo de variação de cada um.

```
referencial = Plot[Sin[x], {x, 0, 2 π}];

Manipulate[
 deslocamento = Plot[Sin[a * x] + b, {x, 0, 2 π}, PlotStyle → RGBColor[0, 0.56, 0.5]];
 Show[referencial, deslocamento],
 {a, -3, 3}, {b, -4, 4}]
```



9. Definindo funções novas

No Mathematica, caso o usuário necessite usar uma função que o programa não possua, é possível defini-la como convir. Trata-se de um procedimento bem simples: deve-se dar um nome para a função (que não pode ser igual ao nome de uma já existente), nomear as variáveis com um _ (underline) em seguida e, então, usar a notação `:=` para definir.

```
In[7]:= vfit[o_, a_, ν_, t_, ϕ_] := o + a*Sin[2*π*ν*t + ϕ]
```

10. `FindFit[]`

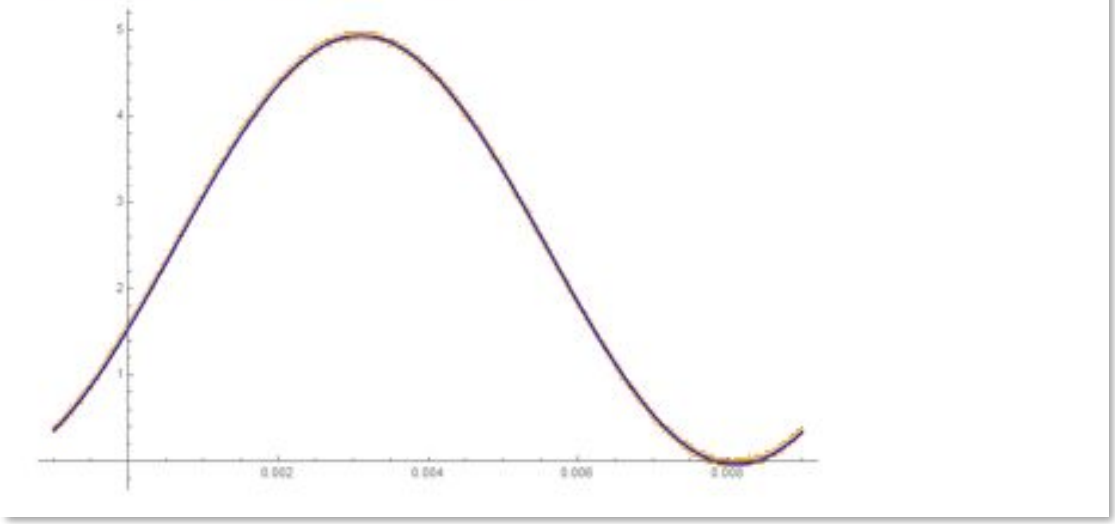
A função `FindFit[dados,{função},{parâmetros},{variáveis}]` é muito útil quando se tem um conjunto de dados experimentais e procura-se achar uma fórmula teórica compatível. Em outras palavras, tal função faz a adequação de parâmetros em função de uma ou mais variáveis para encontrar a melhor equação que defina os dados fornecidos.

```
fitVoltagem = FindFit[ExpVoltagem, vfit[o, a, ν, t, ϕ], {o, a, ν, ϕ}, t]
{o → 2.44903, a → 2.48389, ν → -99.9012, ϕ → 3.51885}
```

No exemplo acima, os dados experimentais importados foram nomeados `ExpVoltagem`.

Para melhor entender o significado desses números, é recomendável plotar o gráfico teórico junto com o experimental usando a função `Show`. Uma dica para evitar ter que copiar os números fornecidos pela FindFit na hora de plotar é digitar `/.nome` após a função, como pode ser visto na imagem abaixo.

```
gfitVoltagem = Plot[vfit[o, a, ν, t, ϕ] /. fitVoltagem, {t, -0.001, 0.009}, PlotStyle → RGBColor[0.02, 0.02, 0.85]];
Show[gExpVoltagem, gfitVoltagem]
```



Neste exemplo, é possível ver em laranja o gráfico experimental - nomeado `gExpVoltagem` - plotado com os dados `ExpVoltagem` e, em azul, o gráfico teórico mais adequado.

Frequentemente, o Mathematica não consegue encontrar o valor dos parâmetros por se tratar de funções muito complexas ou com muitas combinações. Para solucionar esse problema, deve-se definir um intervalo de variação dos parâmetros, de modo a orientar o programa na busca. O comando, então, deve ser `FindFit[dados,{função,{intervalo dos parâmetros}},{parâmetros},{variáveis}]`, sendo que a os intervalos devem ser definidos como $\{x_{min}<x<x_{max}\}$.

```
fitTransmissao = FindFit[ExpTransmissao, {tfit[0, a1, F, a2, 100, ϕ1, ϕ2, t], {4.5 < a1 < 5.5, 9 < F < 13, 1.7 < a2 < 2.3, 2.2 < ϕ1 < 2.5, 5.5 < ϕ2 < 6.2}},
  {a1, F, a2, ϕ1, ϕ2}, t]
{a1 → 4.69405, F → 10.836, a2 → 1.91147, ϕ1 → 2.37626, ϕ2 → 5.87158}
```

Quando não se sabe o intervalo de variação dos parâmetros, uma boa saída é usar o `Manipulate` junto com o `Show` para aproximar manualmente o gráfico teórico do experimental e, então, estimar intervalos próximos ao valor de cada parâmetro.

Erros possíveis e dicas

1. O Mathematica guarda o que foi digitado, mas não os resultados; por isso, é preciso inicializar as células (`shift` + `enter`) toda vez que o programa for iniciado.

2. Como o programa está em inglês, é preciso separar os algarismos decimais com pontos, não com vírgulas.

3. Não confundir as funções `Plot` e `ListPlot`: a primeira traça gráficos a partir de uma fórmula e um domínio, e a segunda, a partir de um conjunto de pontos.

4. Caso haja algum erro ao chamar uma função, recomenda-se utilizar o menu de ajuda do próprio programa (`Help -> Find Selected Function`) para analisar quais parâmetros são necessários e solucionar o problema.

5. Atribuir nomes às funções e a cada etapa pode ser muito útil para poupar tempo.