Title

Name

In[30]:=
```
Clear["Global`*"]; DateString[]
SetDirectory[NotebookDirectory[]];
```

Out[30]=
```
Fri 13 Mar 2026 10:18:25
```

Here is the model oscillation function we want to fit:

In[32]:=
```
osc[V_, f_, ϕ_, Voff_, t_] := V Sin[2 π f t - ϕ] + Voff
```

In[33]:=
```
files = FileNames["TO-driven-*.csv"]
```

Out[33]=
```
{TO-driven-0.40Hz.csv, TO-driven-0.78Hz.csv,
 TO-driven-0.82Hz.csv, TO-driven-1.00Hz.csv}
```

Look at the first few lines of file number 1:

In[34]:=
```
FilePrint[files[[1]], 3]
```

```
"Latest: Time (s)","Latest: Potential (V)","Latest: Potential 2 (V)"

0,0.00457763671875,0.0161743164063

0.002,-0.00030517578125,0.0112915039063
```

This function will read the input voltages:

In[35]:=
```
getVin[filename_] := Select[Import[filename, "CSV"],
   Length[#] == 3 && VectorQ[#, NumberQ] &][[All, {1, 2}]]
```

Check the function by looking at the first few lines it returns for a typical file:

In[36]:=
```
TableForm[
 getVin[files[[1]]][[Range[3]]]]
```

Out[36]//TableForm=

0.	0.00457764
0.002	-0.000305176
0.004	-0.000305176

This function will read the output voltages:

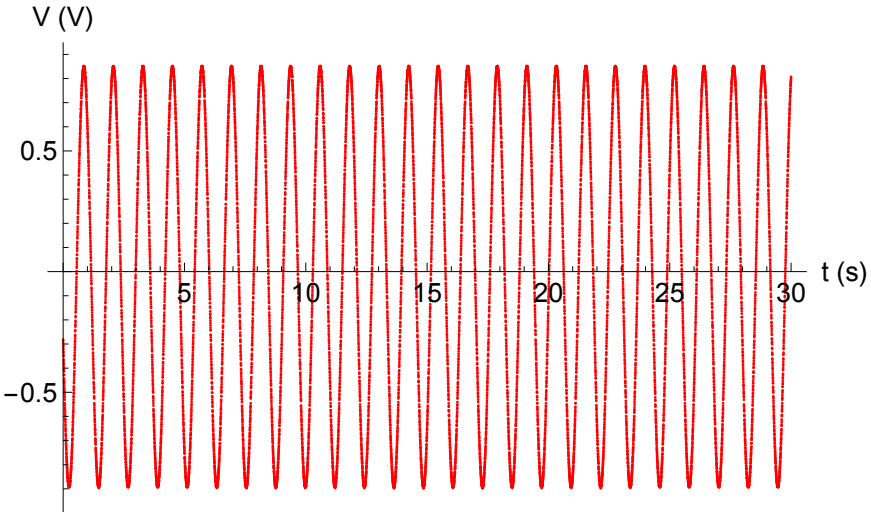
In[37]:=
```
getVout[filename_] := Select[Import[filename, "CSV"],
   Length[#] == 3 && VectorQ[#, NumberQ] &][[All, {1, 3}]]
```

Make up a handy function to show the data.

In[38]:=
```
showData[data_] := ListPlot[data, PlotStyle → Red, LabelStyle → Larger,
  AxesLabel → {"t (s)", "V (V)"}, ImageSize → Scaled[0.7]]
```

In[39]:=
```
showData[getVout[files[[3]]]]
```

Out[39]=



For the torsional oscillator, we want to find the amplitude and phase for each of our input and output voltages, for each of our trials. To do the fits, it will be useful to use good guesses for the frequency and phase. Again, try the commands one by one, and then bundle them all up in a handy function.

Make a list of the frequencies at which the function generator was set . These will help constrain the fits.

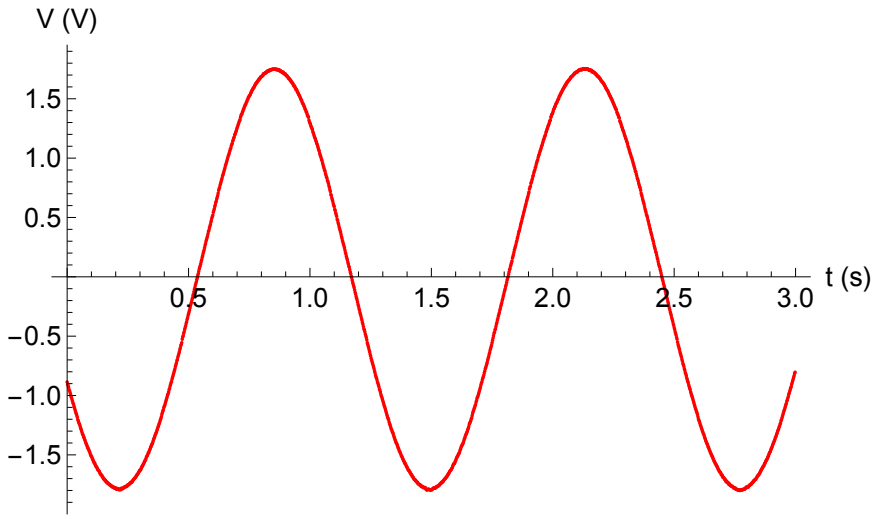
In[40]:=
```
fset = {0.40, 0.78, 0.82, 1.00}
```

Out[40]=
```
{0.4, 0.78, 0.82, 1.}
```

Generate guesses at phases just by looking at the first few seconds of each plot. For example, Vin for file 2 has a phase close to zero. Vout for file 2 has a phase closer to $\pi/2$.

In[41]:=
```
i = 2;
showData[
 getVout[files[[i]]][[Range[1500]]]
]
```

Out[42]=



Make a table of phase guesses for all 4 runs by looking at all the plots. If the triggering in LoggerPro worked correctly, all the input phases should be close to 0. You should also be able to guess the output phases based on your theoretical expectations. These guesses don't have to be precise, but it's helpful if they are in the correct quadrant.

In[43]:=
```
ϕinguesses = {0, 0, 0, 0};
ϕoutguesses = {0, π / 2, π / 2, π};
```

Generate a function to call NonlinearModelFit, using our initial guesses for the phase and frequency. Require the amplitude to be positive. It also turns out to be useful to constrain the frequency rather strictly. (It should be very close to the set value on the function generator, but we can't assume the function generator and LoggerPro agree exactly on timing.) It is easy to guess the amplitude and offset, but that turns out not to make a difference.

In[47]:=
```
fitV[data_, fset_, ϕguess_] := NonlinearModelFit[
  data,
  {osc[V, f, ϕ, Voff, t], V > 0, fset - 0.01 ≤ f ≤ fset + 0.01},
  {
   V,
   {f, fset},
    {ϕ, ϕguess},
    Voff
  }, t]
```

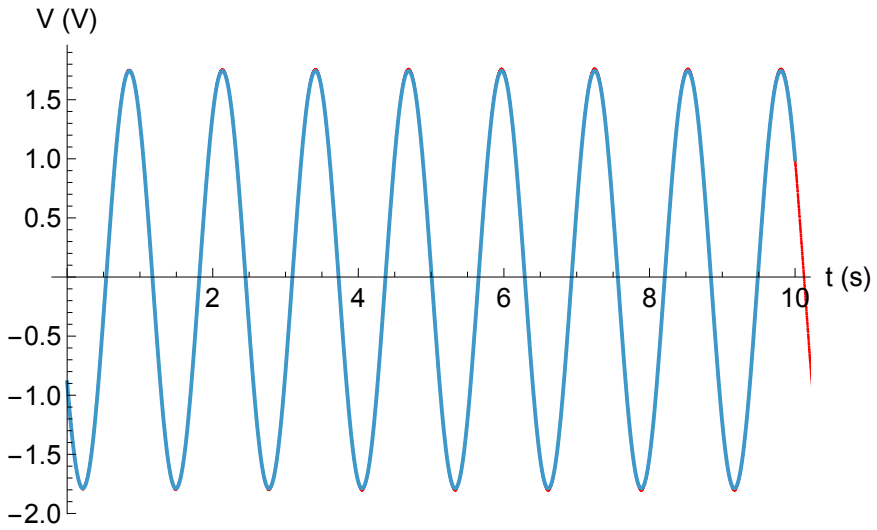
Try it out on file #2:

In[48]:=
```
i = 2;
vdata = getVout[files[[i]]];
vDataPlot = showData[vdata];
fit = fitV[vdata, fset[[i]], ϕoutguesses[[i]]]
fit["BestFitParameters"]
Show[vDataPlot, Plot[fit[t], {t, 0, 10}], PlotRange → {{0, 10}, All}]
```

Out[51]=
```
FittedModel[ −0.0223 − 1.77 Sin [2.63 − 4.91 t] ]
```

Out[52]=
```
{V → 1.76557, f → 0.782053, ϕ → 2.62762, Voff → −0.0223342}
```

Out[53]=



Once you are confident your fitting scheme is working, here is one way to make a table of all the fit results for the input voltages:

In[56]:=
```
infits = Table[
   fitV[getVin[files[[i]]], fset[[i]], ϕinguesses[[i]]]["BestFitParameters"],
   {i, 1, Length[files]}];
TableForm[infits]
```

Out[57]//TableForm=

V → 0.136274	f → 0.399772	ϕ → −0.213803	Voff → −0.0303078
V → 0.133821	f → 0.782074	ϕ → −0.235697	Voff → −0.0301311
V → 0.134595	f → 0.821653	ϕ → −0.226963	Voff → −0.030096
V → 0.135033	f → 1.00137	ϕ → −0.245772	Voff → −0.0303674

You can then easily extract the input amplitudes:

In[58]:=
```
Vins = V /. infits
```

Out[58]=
```
{0.136274, 0.133821, 0.134595, 0.135033}
```

Similarly, you can create a table of the output amplitudes, and then use those lists to create the resonance curve. For the frequencies, just use the values you set on the function generator.