Phys 238: Homework #03
Numerical Integration with NDSolve

In[1]:=
```
Clear["Global`*"]; DateString[]
```
Out[1]=
```
Wed 19 Mar 2025 16:27:58
```

In[2]:=
```
SetDirectory[NotebookDirectory[]];
(* Find and save files alongside this notebook. *)
```

1. The Simple Harmonic Oscillator

Physical constants

In[3]:=
```
k = 1.5 (* Spring constant, in N/m *);
m = 0.08 (* mass, 80 g expressed in kg *);
```

Derived physical constants

In[5]:=
```
ω = Sqrt[k / m];
T = 2 π / ω;
```

Initial conditions and total time

In[7]:=
```
x0 = 0.8; (* amplitude of the motion, in m *)
v0 = 0.0; (* Release from rest *)
tstop = 10; (* Total time to run *)
```

Do the numerical integration

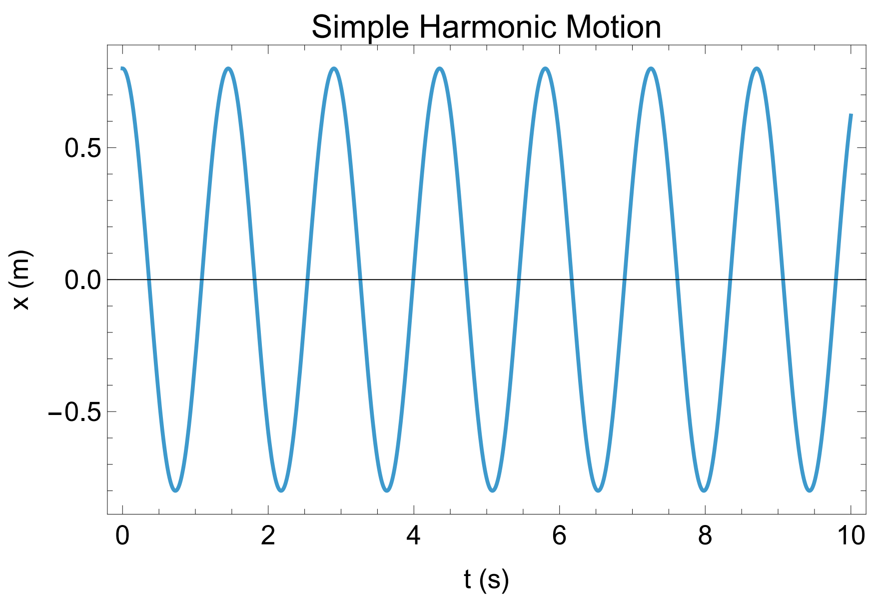
In[10]:=
```
shm = NDSolveValue[
   {m x''[t] == - k x[t], x[0] == x0, x'[0] == v0},
    x,
    {t, 0, tstop}]
```
Out[10]=
```
InterpolatingFunction[ Domain : {{0., 10. }}
                       Output:   scalar ]
```

Results

Assignment 1

In[11]:=
```
Plot[shm[t], {t, 0, tstop}, PlotLabel → "Simple Harmonic Motion",
 LabelStyle → Larger, Frame → True,
 FrameLabel → {"t (s)", "x (m)"}, ImageSize → Scaled[0.7]]
```

Out[11]=



Assignment 2

In[12]:=
```
approx = shm[10]
```

Out[12]=

0.621541

In[13]:=
```
exact = x0 Cos[ω 10]
```

Out[13]=

0.621541

These two are very close. The differences are down in the 8th decimal place.

In[14]:=
```
(approx - exact)
```

Out[14]=

-1.44007×10^{-7}

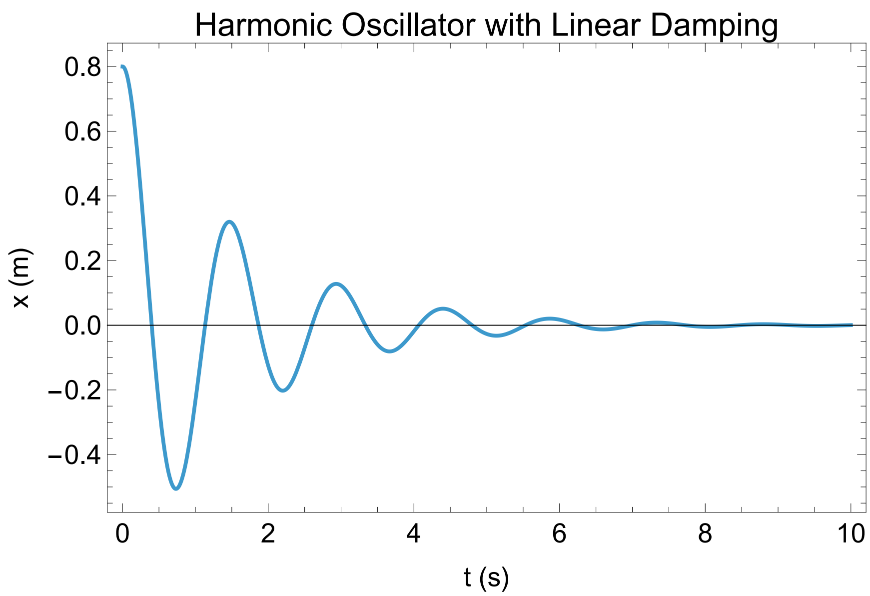
2. Adding Linear Damping

In[15]:=
```
b = 0.1 (* Ns/m *);
```

```
In[16]:= damped = NDSolveValue[
   {m x''[t] == - k x[t] - b x'[t], x[0] == x0, x'[0] == 0},
    x,
    {t, 0, tstop}]
Out[16]=
InterpolatingFunction[ Domain: {{0., 10. }} Output: scalar ]
```

Assignment 3

```
In[17]:= Plot[damped[t], {t, 0, tstop}, PlotRange → All, Frame → True,
  FrameLabel → {"t (s)", "x (m)"}, LabelStyle → Larger,
  PlotLabel → "Harmonic Oscillator with Linear Damping", ImageSize → Scaled[0.7]]
Out[17]=
```



3. Adding Quadratic Damping

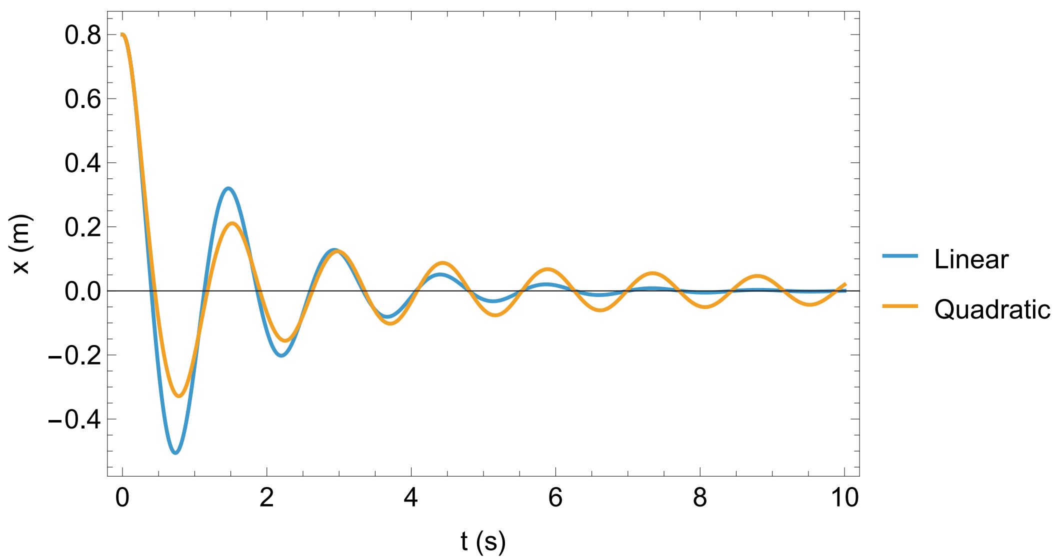
```
In[18]:= c = 0.1 (* Ns^2/m^2*);

In[19]:= airdrag = NDSolveValue[
   {m x''[t] == - k x[t] - c Abs[x'[t]] x'[t], x[0] == x0, x'[0] == v0},
    x,
    {t, 0, tstop}]
Out[19]=
InterpolatingFunction[ Domain: {{0., 10. }} Output: scalar ]
```

Assignment 4

In[20]:=
```
Plot[{damped[t], airdrag[t]}, {t, 0, tstop},
 PlotLegends → {"Linear", "Quadratic"}, Frame → True,
 FrameLabel → {"t (s)", "x (m)"}, LabelStyle → Larger,
 PlotRange → All, ImageSize → Scaled[0.7]]
```

Out[20]=



Assignment 5

Initially, the oscillator is moving quickly, and the quadratic damping is more effective. At later times, however, when the velocity is small, the v^2 term is even smaller, so the quadratic damping term is less effective. Thus the quadratic damping curve decays rapidly at first, but maintains small-amplitude oscillations much longer.

4. Using NDSolve with NonlinearModelFit

In[21]:=
```
Vvst = Select[Import["TO-20250221-airdrag-1.csv", "CSV"], VectorQ[#, NumberQ] &];
```

Apply my angle calibration. (There are more concise ways to do this with functions like Map, but the Table form is clear and easy to understand.)

In[22]:=
```
dθdV = 0.558;
```

In[23]:=
```
θvst = Table[{Vvst[[i, 1]], dθdV * Vvst[[i, 2]]}, {i, 1, Length[Vvst]}];
```

In[24]:=
```
dataPlot = ListPlot[θvst, PlotStyle → Red, PlotRange → All, LabelStyle → Larger,
   AxesLabel → {"t (s)", "Angle (radians)"}, ImageSize → Scaled[0.8]];
```

Enter the model for motion with air drag. Allow for a small angle offset so that
the relaxed position is not exactly 0. (This depends on the ‘Zero Adjust’ knob on the apparatus.) The

use of ParametricNDSolveValue is suggested in the NonlinearModelFit help page. See the first example under “Generalizations & Extensions”.

In[25]:=
```
Clear[ω0, c, θ0, v0, θoff]
```

In[26]:=
```
airdrag =
 ParametricNDSolveValue[
  {θ''[t] == - ω0^2 (θ[t] - θoff) - c Abs[θ'[t]] θ'[t],
   θ[0] == θ0, θ'[0] == v0},
   θ,
  {t, 0, 30}, {ω0, c, θ0, v0, θoff}]
```

Out[26]=
```
ParametricFunction[ Expression : θ
                    Parameters: {ω0, c, θ0, v0, θoff} ]
```

Test the function using the suggested values:

In[27]:=
```
airdrag[2.5, 0.5, 0.7, -0.02, 0.1][12]
```

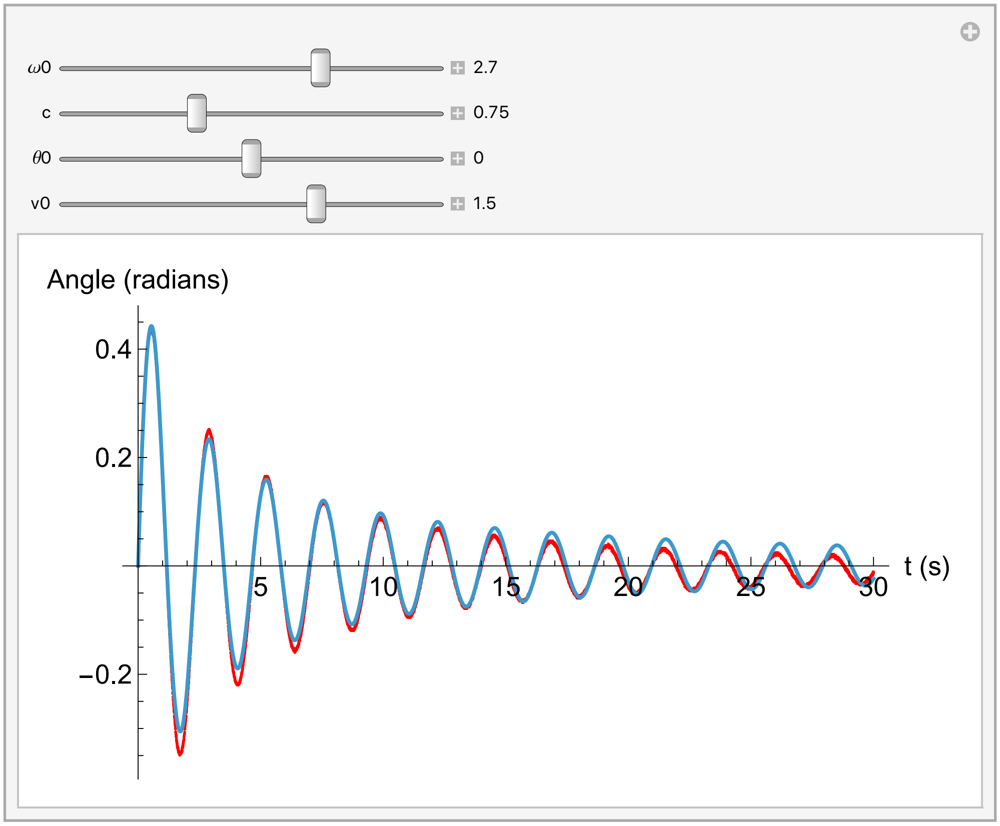
Out[27]=
```
0.106687
```

Assignment 6

It is important to make a good first guess at the parameters.

In[28]:=
```
Manipulate[Show[
   {dataPlot, Plot[airdrag[ω0, c, θ0, v0, 0][t], {t, 0, 30}, PlotRange → All]}],
  {{ω0, 2.5}, 2.0, 3.0, 0.01, Appearance → "Labeled"},
  {{c, 0.5}, 0.1, 2.0, 0.05, Appearance → "Labeled"},
  {{θ0, 0.5}, -1, 1, 0.05, Appearance → "Labeled"},
  {{v0, 0}, -4, 4, 0.05, Appearance → "Labeled"}
]
```

Out[28]=



Reasonable guesses are: $\omega 0 = 2.7$, $c = 0.75$, $\theta 0 = 0$, $v0 = 1.5$, $\theta off = -0.01$.

Assignment 7

In[29]:=
```
Clear[ω0, c, θ0, v0, θoff];
airdragFit = NonlinearModelFit[θvst, airdrag[ω0, c, θ0, v0, θoff][t],
  {{ω0, 2.7}, {c, 0.75}, {θ0, 0}, {v0, 1.5}, {θoff, -0.1}}, t]
```

ParametricNDSolveValue : An invalid NDSolve`SensitivityMethod method data object was encountered at the point t == 0.`.

ParametricNDSolveValue : An invalid NDSolve`SensitivityMethod method data object was encountered at the point t == 0.`.

ParametricNDSolveValue : An invalid NDSolve`SensitivityMethod method data object was encountered at the point t == 0.`.

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

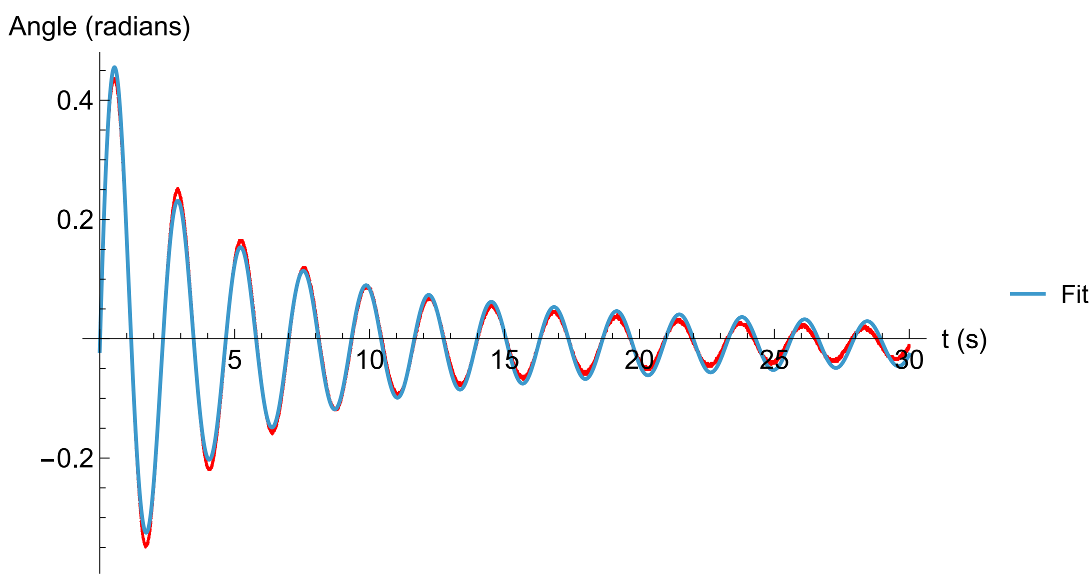
NonlinearModelFit : The step size in the search has become less than the tolerance prescribed by the PrecisionGoal option, but the gradient is larger than the tolerance specified by the AccuracyGoal option. There is a possibility that the method has stalled at a point that is not a local minimum.

Out[29]=

```
FittedModel[ InterpolatingFunction[ Domain: {{0., 30.}} Output: scalar ][t] ]
```

In[30]:=
```
Show[{dataPlot,
  Plot[airdragFit[t], {t, 0, 30}, PlotRange → All, PlotLegends → {"Fit"}]}]
```

Out[30]=



Assignment 8

In[31]:=
```
airdragFit["BestFitParameters"]
```

Out[31]=

```
{ω0 → 2.7074, c → 0.74485, θ0 → -0.0205145, v0 → 1.60616, θoff → -0.00896301}
```

In[32]:=
```
airdragFit["ParameterConfidenceIntervalTable"]
```

General : Exp[−76334.4] is too small to represent as a normalized machine number; precision may be lost.

General : Exp[−24241.5] is too small to represent as a normalized machine number; precision may be lost.

General : Exp[−24246.4] is too small to represent as a normalized machine number; precision may be lost.

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

Out[32]=

	Estimate	Standard Error	Confidence Interval
ω0	2.7074	0.000136151	{2.70713, 2.70767 }
c	0.74485	0.00123287	{0.742433, 0.747266 }
θ0	−0.0205145	0.000541206	{−0.0215753, −0.0194537 }
v0	1.60616	0.0026576	{1.60095, 1.61137 }
θoff	−0.00896301	0.0000761471	{−0.00911227 , −0.00881376 }

Assignment 9

```
Sqrt[airdragFit["EstimatedVariance"]] (* RMSE in radians*)
```

Out[33]=

```
0.0092475
```

```
Sqrt[airdragFit["EstimatedVariance"]] / dθdV (* RMSE in volts *)
```

Out[35]=

```
0.0165726
```

An overall variation of less than 0.01 radians is small. It is about the same size as the residual scatter in the voltage calibration plot, although that scatter was likely due to limits of my ability to manually read the angle scale, rather than the limits of the voltage sensor. Looking at it differently, the RMSE here is about 0.016 Volts, well within the measurement capability of LoggerPro. The data lines do not show any visible noise anywhere near this large.

Note too that this scatter is not random. There is a slight systematic trend visible in the plot in that the fit (blue line) decays a little too quickly at early times, and not quickly enough at later times. This suggests that the model is not quite a complete description of the data.

Assignment 10

In[44]:=
```
TableForm[airdragFit["CorrelationMatrix"],
 TableHeadings → {(* Rows: *) {"ω0", "c", "θ0", "v0", "θoff"},
                  (* Columns: *) {"ω0", "c", "θ0", "v0", "θoff"} }]
```

Out[44]//TableForm=

	ω0	c	θ0	v0	θoff
ω0	1.	0.0542745	−0.583087	0.244083	0.00479782
c	0.0542745	1.	−0.0947344	0.571559	−0.0732339
θ0	−0.583087	−0.0947344	1.	−0.370296	0.146858
v0	0.244083	0.571559	−0.370296	1.	−0.122348
θoff	0.00479782	−0.0732339	0.146858	−0.122348	1.

There is a negative correlation between $\omega0$ and $\theta0$, and a positive correlation between c and v0. For $\omega0$ and $\theta0$, increasing the frequency makes the peaks come sooner, but decreasing $\theta0$ delays them so they come later. This only really works for the first few peaks; changing $\omega 0$ means the later peaks are not aligned with the fit. However, since they are smaller, they contribute less to the overall error.

The initial velocity and drag are correlated because giving it a larger initial velocity can be compensated by increasing the drag. Again, this does not work as well for the later peaks, but since they are all smaller, they contribute less to the overall error.