Differential Equations&*Mathematica*

©1999 Bill Davis and Jerry Uhl
Produced by Bruce Carpenter Published by Math Everywhere, Inc.
www.matheverywhere.com

DE.02 Transition from Calculus to DiffEq: The Forced Oscillator DiffEq Basics

B.1) Underdamped oscillators (b > 0 and small) and overdamped oscillators (b > 0 and large)

□B.1.a.i)

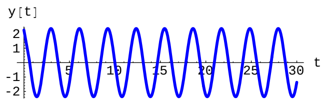
The undamped oscillator diffeq is y″[t] + c y[t] = 0 with c > 0.
Here's a random undamped oscillator diffeq with random starter values for y[0] and y′[0]:

```
c = Random[Real, {3, 5}];
starttery = Random[Real, {2, 5}];
starteryprime = Random[Real, {-2, 2}];
Clear[t, y]
undampedoscdiffeq =
  {y''[t] + c y[t] == 0, y[0] == startery,
   y'[0] == starteryprime};
ColumnForm[Thread[undampedoscdiffeq]]
```

```
4.04627 y[t] + y″[t] == 0
y[0] == 2.26103
y′[0] == -1.41245
```

The solution plots out this way:

```
endtime = 30;
ndsol = NDSolve[undampedoscdiffeq, y[t], {t, 0, endtime}];
Clear[yundamped]
yundamped[t_] = y[t] /. ndsol[[1]];
undampedplot = Plot[yundamped[t], {t, 0, endtime},

  PlotStyle → {{Thickness[0.01], Blue}}, AspectRatio → 1/4,

  PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



A sine wave.
Regular repeating oscillations forever.
How do clued-in diffeq folks reach in and damp this oscillator so that the the solutions tend to 0 as t gets large?

□Answer:

The undamped oscillator diffeq is

$$y''[t] + c\,y[t] = 0:$$

```
Clear[t, y]
undampedoscdiffeq =
  {y''[t] + c y[t] == 0, y[0] == startery,
   y'[0] == starteryprime};
ColumnForm[undampedoscdiffeq]
```

```
4.04627 y[t] + y″[t] == 0
y[0] == 2.26103
y′[0] == -1.41245
```

To damp this oscillator so that all the solutions tend to 0 as t gets large, you keep everything the same but you insert a damping term b y′[t] (with b > 0) into the middle of the diffeq to get the damped oscillator diffeq

$$y''[t] + b\,y'[t] + c\,y[t] = 0:$$

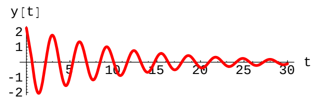
`It's fine to think of the damping term as friction.`

```
b = 0.18;
Clear[t, y]
dampedoscdiffeq =
  {y''[t] + b y'[t] + c y[t] == 0, y[0] == startery,
  y'[0] == starteryprime};
ColumnForm[dampedoscdiffeq]
```

```
4.04627 y[t] + 0.18 y′[t] + y″[t] == 0
y[0] == 2.26103
y′[0] == -1.41245
```

The solution plots out this way:

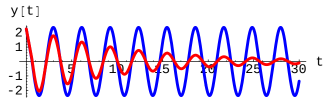
```
endtime = 30;
ndsol = NDSolve[dampedoscdiffeq, y[t], {t, 0, endtime}]; Clear[ydamped]
ydamped[t_] = y[t] /. ndsol[[1]]; dampedplot1 = Plot[ydamped[t],
  {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Red}},

  AspectRatio → 1/4, PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



A damped sine wave which is squashed to 0 as t gets large.

See the undamped solution and the damped solution together:

```
Show[undampedplot, dampedplot1];
```

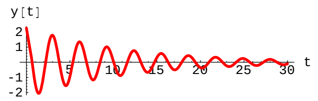


Damped and undamped versions of the same oscillator.

□B.1.a.ii)

Here is the plot of the damped oscillator from part i)

```
Show[dampedplot1];
```



How do you damp it even more?

□Answer:

To get this plot, you inserted b y′[t] with this b:

```
b
```

```
0.18
```

To damp it even more you raise b:

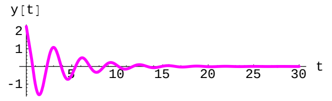
`It's fine to think of the raising the damping termas incorporating more friction.`

```
higherb = 0.5;
Clear[t, y];
moredampedoscdiffeq = {y''[t] + higherb y'[t] + c y[t] == 0,
      y[0] == startery, y'[0] == starteryprime};
ColumnForm[Thread[moredampedoscdiffeq]]
```

```
4.04627 y[t] + 0.5 y′[t] + y″[t] == 0
y[0] == 2.26103
y′[0] == -1.41245
```

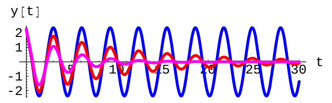
The new solution plots out this way:

```
endtime = 30;
ndsol = NDSolve[
 moredampedoscdiffeq , y[t], {t, 0, endtime}];

Clear[newydamped]
ymoredamped[t_] = y[t] /. ndsol[[1]]; dampedplot2 = Plot[ymoredamped[t],
  {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Magenta}},

  AspectRatio → 1/4, PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



See them all:

```
Show[undampedplot, dampedplot1, dampedplot2];
```



The higher b is, the more damping you get.

□B.1.a.iii) Underdamped versus overdamped

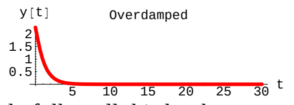
Here's what happens when you increase the damping term b quite a bit more:

```
bigb = 5.2;
Clear[t, y]
overdampedoscdiffeq =
  {y''[t] + bigb y'[t] + c y[t] == 0, y[0] == startery,
y'[0] == starteryprime};
ColumnForm[overdampedoscdiffeq]
```

```
4.04627 y[t] + 5.2 y′[t] + y″[t] == 0
y[0] == 2.26103
y′[0] == -1.41245
```

The new solution plots out this way:

```
endtime = 30;
ndsol = NDSolve[overdampedoscdiffeq, y[t], {t, 0, endtime}];
Clear[yoverdamped]
yoverdamped[t_] = y[t] /. ndsol[[1]];
overdampedplot = Plot[yoverdamped[t], {t, 0, endtime}, PlotStyle →
   {{Thickness[0.015], Red}}, AspectRatio → 1/4, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Overdamped"];
```



Why do folks call this by the name overdamped?

□**Answer:**

They say this ocscillator is overdamped because the plot doesn't wiggle.

Too much friction.

B.2) Forcing the linear oscillator:

yforced[t] = yunforced[t] + yzeroinput[t]

For damped linear oscillators:

yunforced[t] is transient and yzeroinput[t] gives the steady state behavior

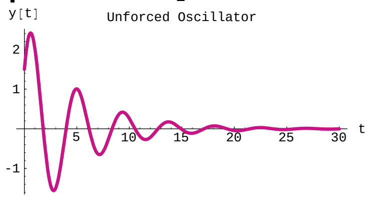
□B.2.a.i)

Look at this plot of the solution of:

y″[t] + 0.4 y′[t] + 2.1 y[t] = 0;

with y[0] = 1.5 and y′[0] = 3.0:

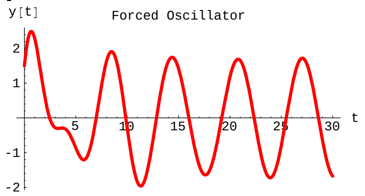
```
b = 0.4;
c = 2.1;
Clear[y, t]
unforcedoscillator = y″[t] + b y′[t] + c y[t] == 0;
initialdisplacement = 1.5;
initialvel = 3.0;
endtime = 30;
Clear[s, y, unforcedy, t]
unforcedsolution =
  NDSolve[{unforcedoscillator, y[0] == initialdisplacement,
    y′[0] == initialvel}, y[t], {t, 0, endtime}];
unforcedy[t_] = y[t] /. unforcedsolution[[1]];

unforcedplot = Plot[unforcedy[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], MediumVioletRed}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Unforced Oscillator",
  AspectRatio → 1/2];
```



Here's what you get when you take the same damped oscillator and the same starter data, but at time t, you throw in an external force proportional to

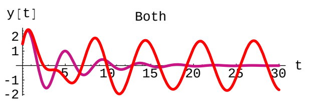
f[t] = 2 Sin[t]:

```
Clear[f, t, forcedy]
f[t_] = 2 Sin[t];
forcedoscillator = y″[t] + b y′[t] + c y[t] == f[t];
forcedsolution =
  NDSolve[{forcedoscillator, y[0] == initialdisplacement,
    y′[0] == initialvel}, y[t], {t, 0, endtime}];
forcedy[t_] = y[t] /. forcedsolution[[1]];

forcedplot = Plot[forcedy[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Forced Oscillator",
  AspectRatio → 1/2];
```



Compare:

```
both =
  Show[unforcedplot, forcedplot, PlotLabel → "Both", AspectRatio → 1/4];
```



Describe what you see and try to explain why you see it.

□**Answer:**

The unforced (damped) linear oscillator comes from:

```
unforcedoscillator
```

```
2.1 y[t] + 0.4 y′[t] + y″[t] == 0
```

with starter data y[0] = 1.5 and y′[0] = 3.0;

The forced (damped) linear oscillator comes from:

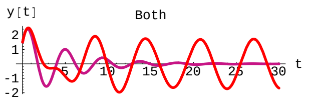
```
forcedoscillator
```

```
2.1 y[t] + 0.4 y′[t] + y″[t] == 2 Sin[t]
```

with the same starter data.

Take another look:

```
Show[both];
```



Evidently the effect of the forcing function

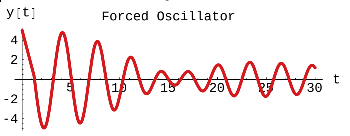
f[t] = 2 Sin[t]

eventually overcomes the damping term and sends the damped oscillator into a regular steady state.

□B.2.a.ii) Steady state and the zero input response

Look at a new forced linear oscillator:

```
b = Random[Real, {0.2, 0.8}];
c = Random[Real, {2.0, 3.5}];
Clear[y, t]
initialdisplacement = 5;
initialvel = 3.0;
endtime = 30;
Clear[s, y, notforcedy, t]
f[t_] = 2 Cos[2 t];
forcedoscillator = y″[t] + b y′[t] + c y[t] == f[t];
forcedsolution =
  NDSolve[{forcedoscillator, y[0] == initialdisplacement,
    y′[0] == initialvel}, y[t], {t, 0, endtime}];
forcedy[t_] = y[t] /. forcedsolution[[1]];

forcedplot = Plot[forcedy[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], VenetianRed}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Forced Oscillator",
  AspectRatio → 1/3];
```



This comes from:

```
forcedoscillator
2.72607 y[t] + 0.256808 y′[t] + y″[t] == 2 Cos[2 t]
```

with starter data y[0] = 1.5 and y′[0] = 3.0.
This forced oscillator is confused for a while and then settles into steady-state in the long run.
How do you get your hands on plotting the long run steady state behavior?

□ Answer:

It's simple!

You just change the starter data to something really easy such as

y[0] = 0 and y′[0] = 0.

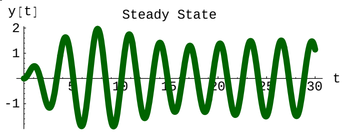
And then you solve:

```
forcedoscillator
2.72607 y[t] + 0.256808 y′[t] + y″[t] == 2 Cos[2 t]
```

with starter data y[0] = 0 and y′[0] = 0.

Lots of folks call this the zero input response

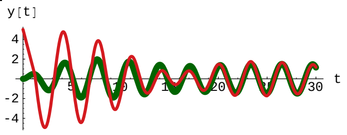
```
initialdisplacement = 0;
initialvel = 0;
endtime = 30;
forcedoscillator = y″[t] + b y′[t] + c y[t] == f[t];
zeroinputsolution =
 NDSolve[{forcedoscillator, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
Clear[yzeroinput]
yzeroinput[t_] = y[t] /. zeroinputsolution⟦1⟧;
steadystateplot = Plot[yzeroinput[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.02], GreenDark}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Steady State",
  AspectRatio → 1/3];
```



Lots of folks call this the zero input solution

Compare with the plot of the original forced oscillator:

```
Show[steadystateplot, forcedplot, PlotLabel → None];
```



In the long run they eventually settle into the same steady state oscillations.

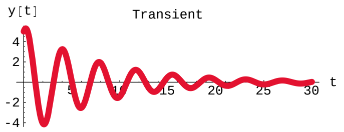
This will happen no matter what starter data you go with.

The zero input solution always captures the essence of the long-term action of any forced damped oscillator.

□B.2.a.iii) Transient

Stay with the same oscillator and look at this plot of
yforced[t] − yzeroinput[t]:

```
transientplot = Plot[forcedy[t] - yzeroinput[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.02], GeraniumLake}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Transient",
  AspectRatio → 1/3];
```



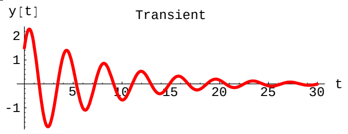
Why do folks call this the transient effect?
How do you plot this directly?

□Answer:

It's called the transient effect because it dies out because of the effect of the damping term.

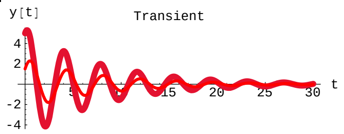
You can plot it directly by taking the original forced oscillator, keeping the same starter data, but zeroing out the forcing function f[t]:

```
endtime = 30;
initialdisplacement = 1.5;
initialvel = 3.0;
Clear[s, y, yunforcedy, t]
unforcedoscillator = y″[t] + b y′[t] + c y[t] == 0;
unforcedsolution =
 NDSolve[{unforcedoscillator, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
yunforced[t_] = y[t] /. unforcedsolution⟦1⟧;

directtransientplot = Plot[yunforced[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Transient",
  AspectRatio → 1/3];
```



Compare:

```
Show[transientplot, directtransientplot];
```

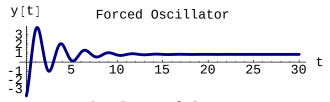


On the money.

□B.2.b.i)

Here's a new forced oscillator subjected to a constant forcing function:

```
b = Random[Real, {0.5, 1.4}];
c = Random[Real, {4.0, 6.0}];
initialdisplacement = -3.7;
initialvelocity = 4.0;
Clear[f, t, yforced]
f[t_] = 5.0;
forcedoscillator = y″[t] + b y′[t] + c y[t] == f[t];
forcedsolution =
 NDSolve[{forcedoscillator, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
yforced[t_] = y[t] /. forcedsolution⟦1⟧;

forcedplot = Plot[yforced[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], NavyBlue}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Forced Oscillator",
  AspectRatio → 1/4];
```



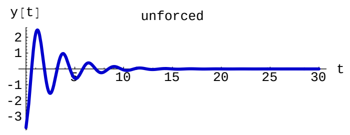
Come up with plots of the zero input solution and the the unforced solution.
Show a plot of
yzeroinput[t] + yunforced[t]
together with the plot immediately above and describe what you see.

□ Answer:

Here comes a plot of yunforced[t]:

You get this by zeroing out the forcing function f[t]
and going with the given starter data.

```
Clear[y, t, transient]
unforcedoscillator = y″[t] + b y′[t] + c y[t] == 0;
unforcedsolution =
 NDSolve[{unforcedoscillator, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
yunforced[t_] = y[t] /. unforcedsolution⟦1⟧;

transientplot = Plot[yunforced[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], MediumBlue}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "unforced", AspectRatio → 1/3];
```

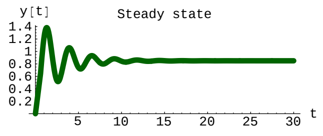


You can see yunforced[t] dying out.
That's why some folks like to call it "transient."

Here comes a plot of yzeroinput[t]:

You get this by zeroing out the starting data
and going with given forcing function.

```
Clear[yzeroinput]
forcedoscillator = y''[t] + b y'[t] + c y[t] == f[t];
zeroinputsolution = NDSolve[
  {forcedoscillator, y[0] == 0, y'[0] == 0}, y[t], {t, 0, endtime}];
yzeroinput[t_] = y[t] /. zeroinputsolution[[1]];
steadystateplot = Plot[yzeroinput[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.02], GreenDark}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Steady state",
  AspectRatio → 1/3];
```

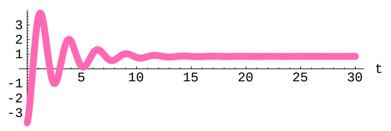


Zap.

You can see yzeroinput[t] settling into a steady state.
That's why some folks like to say you get
steady state information from yzeroinput

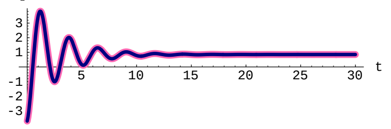
Here's a plot of yzeroinput[t] + yunforced[t]:

```
combinedaction = Plot[yzeroinput[t] + yunforced[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.02], HotPink}}, PlotRange → All,
  AxesLabel → {"t", ""}, AspectRatio → 1/3];
```



Compare to *Mathematica*'s plot of the action of the forced oscillator:

```
Show[combinedaction, forcedplot, AxesLabel → {"t", ""}];
```



They are the same. It works this way everytime.

If you want to see why, go on.

□B.2.b.ii) yforced[t] = yzeroinput[t] + yunforced[t]

In the last part it became apparent that

yforced[t] = yzeroinput[t] + yunforced[t].

Explain why this happened and why you are guaranteed that this will happen anytime you are dealing with a forced linear (damped or undamped) oscillator.

□ **Answer:**

Go with a forced linear oscillator with cleared forcing function and cleared coefficients.

```
Clear[y, f, b, c]
forcedoscillator = y''[t] + b y'[t] + c y[t] == f[t]
c y[t] + b y'[t] + y''[t] == f[t]
```

with cleared starter data y[0] = p and y′[0] = q.

Facts to ponder:

→ yzeroinput[t] solves:

```
Clear[yzeroinput]
zeroinputeq = forcedoscillator /.
  {y[t] → yzeroput[t], y'[t] → yzeroinput'[t], y''[t] → yzeroinput''[t]}
c yzeroput[t] + b yzeroinput'[t] + yzeroinput''[t] == f[t]
```

with starter data yzeroinput[0] = 0 and yzeroinput′[0] = 0.

→ yunforced[t] solves:

```
Clear[yunforced]
unforcedeq = forcedoscillator /. {y[t] → yunforced[t],
   y'[t] → yunforced'[t], y''[t] → yunforced''[t], f[t] → 0}
c yunforced[t] + b yunforced'[t] + yunforced''[t] == 0
```

with starter data yunforced[0] = p and yunforced′[0] = q.

Adding the two equations and remembering that the derivatives of the sums are sums of derivatives, you see that when you put

yforced[t] = yzeroinput[t] + yunforced[t],

then you are guaranteed that yforced[t] solves the original forced oscillator differential equation

yforced′′[t] + b yforced′[t] + c yforced[t] = f[t] + 0 = f[t].

Also this gives you

yforced[0] = yzeroinput[0] + yunforced[0] = 0 + p = p

and

yforced′[0] = yzeroinput′[0] + yunforced′[0] = 0 + q = q,

The upshot: yforced[t] solves

```
Clear[y, f, b, c]
forcedoscillator = y''[t] + b y'[t] + c y[t] == f[t]
c y[t] + b y'[t] + y''[t] == f[t]
```

with correct starter data y[0] = p and y′[0] = q.

That's why you can count on

yforced[t] = yzeroinput[t] + yunforced[t]

for any and all forced linear oscillators.

B.3) Euler's identity:

$$E^{(a+I\,b)\,t} = E^{a t}(\mathrm{Cos}[b\,t] + I\,\mathrm{Sin}[b\,t]) \text{ where } I = \sqrt{-1}$$

□B.3.a)

Look at these:

```
Clear[t]
ComplexExpand[E^(I t)]
Cos[t] + I Sin[t]
Clear[t]
ComplexExpand[E^(I 2 t)]
Cos[2 t] + I Sin[2 t]
Clear[t]
ComplexExpand[E^(I π t)]
Cos[π t] + I Sin[π t]
b = Random[Real, {-3, 3}]; ComplexExpand[E^(I b t)]
Cos[2.16529 t] + I Sin[2.16529 t]
```

Rerun several times.

```
a = Random[Real, {-3, 3}]
b = Random[Real, {-3, 3}]
ComplexExpand[E^((a+I b) t)]
2.30762
0.0535402
E^(2.30762 t) Cos[0.0535402 t] + I E^(2.30762 t) Sin[0.0535402 t]
```

Rerun several times.

What the heck is going on here?

□ **Answer:**

Mathematica is spitting out Euler's identity.

Euler's Identity says:

If a and b are real numbers and $I = \sqrt{-1}$, then

$$E^{(a+I\,b)\,t} = E^{a\,t}\,(\mathrm{Cos}[b\,t] + I\,\mathrm{Sin}[b\,t]).$$

Folks call this the complex exponential. It is programmed right into *Mathematica*, and you will want it to be programmed into your brain as well.

Check it out:

```
a = Random[Real, {-3, 3}];
b = Random[Real, {-3, 3}];
ComplexExpand[E^((a+I b) t)]
E^(a t) (Cos[b t] + I Sin[b t])
```

$E^{2.77429\,t}\,\mathrm{Cos}[2.60966\,t] - I\,E^{2.77429\,t}\,\mathrm{Sin}[2.60966\,t]$

$E^{2.77429\,t}\,(\mathrm{Cos}[2.60966\,t] - I\,\mathrm{Sin}[2.60966\,t])$

Rerun several times.

It happens every time.

□ B .3 . b) The first and second derivatives of $E^{(a+I\,b)\,t}$

Check this out:

```
Clear[y, t, a, b, K]
y[t_] = K E^((a+I b) t)
```

$E^{(a+I\,b)\,t}\,K$

```
y'[t]
```

$(a + I\,b)\,E^{(a+I\,b)\,t}\,K$

```
y''[t]
```

$(a + I\,b)^2\,E^{(a+I\,b)\,t}\,K$

What's the message?
Explain why it happens.

□ Answer:

The message is that when you go with

$$y[t] = K\,E^{(a+I\,b)\,t}$$

then you are guaranteed that

$$y'[t] = (a + I\,b)\,K\,E^{(a+I\,b)\,t} = (a + I\,b)\,y[t]$$

and

$$y''[t] = (a + I\,b)^2\,K\,E^{(a+I\,b)\,t} = (a + I\,b)^2\,y[t]$$

You take the derivative of the complex exponential the same way you were used to taking the derivative of any exponential back in calculus.

If you want to see why this works out the way it does, click on the right.

Start with

$$y[t] = K\,E^{(a+I\,b)\,t} = K\,E^{a\,t}\,(\mathrm{Cos}[b\,t] + I\,\mathrm{Sin}[b\,t])$$

```
Clear[k, a, b, y, t]
y[t_] = K E^(a t) (Cos[b t] + I Sin[b t])
```

$E^{a\,t}\,K\,(\mathrm{Cos}[b\,t] + I\,\mathrm{Sin}[b\,t])$

Multiply it out:

```
Expand[y[t]]
```

$E^{a\,t}\,K\,\mathrm{Cos}[b\,t] + I\,E^{a\,t}\,K\,\mathrm{Sin}[b\,t]$

Use the product rule to differentiate both parts with respect to t and multiply out, remembering that $I^2 = -1$:

```
Expand[y'[t]]
```

$a\,E^{a\,t}\,K\,\mathrm{Cos}[b\,t] + I\,b\,E^{a\,t}\,K\,\mathrm{Cos}[b\,t] + I\,a\,E^{a\,t}\,K\,\mathrm{Sin}[b\,t] - b\,E^{a\,t}\,K\,\mathrm{Sin}[b\,t]$

Compare:

```
Expand[(a + I b) y[t]]
```

$a\,E^{a\,t}\,K\,\mathrm{Cos}[b\,t] + I\,b\,E^{a\,t}\,K\,\mathrm{Cos}[b\,t] + I\,a\,E^{a\,t}\,K\,\mathrm{Sin}[b\,t] - b\,E^{a\,t}\,K\,\mathrm{Sin}[b\,t]$

So when you start with $y[t] = K\,E^{(a+I\,b)\,t}$, you get
$y'[t] = K\,(a + I\,b)\,E^{(a+I\,b)\,t}$

B.4) The characteristic equation of the unforced linear oscillator diffeq

$$\mathbf{y''[t] + b\,y'[t] + c\,y[t] = 0}$$

is

$$\mathbf{z^2 + b\,z + c = 0.}$$

If z_1 and z_2 solve the characteristic equation, and K_1 and K_2 are constants, then

$$\mathbf{K_1\,E^{z_1\,t} + K_2\,E^{z_2\,t}}$$

is a solution of

$$\mathbf{y''[t] + b\,y'[t] + c\,y[t] = 0.}$$

How to set K_1 and K_2 to handle starter data on y[0] and y′[0].

B.4.a.i) Using the solutions of the characteristic equation

Here's an unforced linear oscillator diffeq:

```
b = 0.45;
c = 2.2;
startery = -1.0;
starteryprime = 2.0;
Clear[t, y, Derivative]
oscdiffeq =
  {y''[t] + b y'[t] + c y[t] == 0, y[0] == startery, y'[0] == starteryprime};
ColumnForm[Thread[oscdiffeq]]
```

```
2.2 y[t] + 0.45 y'[t] + y''[t] == 0
y[0] == -1.
y'[0] == 2.
```

The characteristic equation for this oscillator diffeq is:

```
Clear[z]
z^2 + b z + c == 0
```

$2.2 + 0.45\,z + z^2 == 0$

The characteristic equation for

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

is

$$z^2 + b\,z + c = 0$$

Use the solutions of the characteristic equation to come up with a formula for the solution of this oscillator diffeq.
Plot the solution and describe what you see.

□ Answer:

□Step1: Write down the characteristic equation:

```
Clear[z]
charequation = z^2 + b z + c == 0
```

$2.2 + 0.45\,z + z^2 == 0$

Note the relation between the characteristic equation and the oscillator diffeq:

```
ColumnForm[Thread[oscdiffeq]]
```

```
2.2 y[t] + 0.45 y'[t] + y''[t] == 0
y[0] == -1.
y'[0] == 2.
```

Again: the characteristic equation for

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

is

$$z^2 + b\,z + c = 0$$

□Step 2: Solve the charactistic equation for z to set up the general solution of the diffeq

```
zsols = Solve[charequation, z]
```

```
{{z → -0.225 - 1.46607 I}, {z → -0.225 + 1.46607 I}}
```

Fish the solutions for z out:

```
z1 = zsols[[1, 1, 2]]
```

```
-0.225 - 1.46607 I
```

```
z2 = zsols[[2, 1, 2]]
```

```
-0.225 + 1.46607 I
```

You can use the quadratic formula from high school do this with pencil and paper.

Set up the general form of the solutions.

```
Clear[gensol, K1, K2]
gensol[t_] = K1 E^(z1 t) + K2 E^(z2 t)
```

$E^{(-0.225-1.46607\,I)\,t}\,K1 + E^{(-0.225+1.46607\,I)\,t}\,K2$

Here K1 and K2 are cleared constants.

□Step 3: Solve for the K1 and K2 that correspond to the given starting data on y[0] and y′[0].

```
ystarteq = (gensol[0] == startery);
yprimestarteq = (gensol'[0] == starteryprime);
Ksols = Solve[{ystarteq, yprimestarteq}, {K1, K2}]
```

```
{{K1 → -0.5 + 0.605358 I, K2 → -0.5 - 0.605358 I}}
```

Substitute these values of K1 and K2 in to get the raw form of the exact formula:

```
gensol[t] /. Ksols[[1]]
```

$(-0.5 + 0.605358\, I)\, E^{(-0.225 - 1.46607\, I)\, t} - (0.5 + 0.605358\, I)\, E^{(-0.225 + 1.46607\, I)\, t}$

Make it look nice by hitting gensol[t] with the fundamental identity

$$E^{(a+I\,b)\,t} = E^{a\,t}\,\text{Cos}[b\,t] + I\,E^{a\,t}\,\text{Sin}[b\,t]:$$

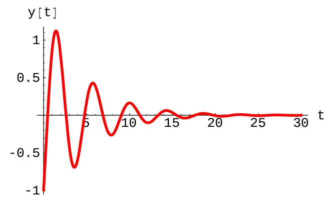
```
Clear[yformula]
yformula[t_] = Chop[ComplexExpand[gensol[t] /. Ksols[[1]]]]
```

$-1.\, E^{-0.225\, t}\, \text{Cos}[1.46607\, t] + 1.21072\, E^{-0.225\, t}\, \text{Sin}[1.46607\, t]$

See the solution :

```
ColumnForm[Thread[oscdiffeq]]

Plot[yformula[t], {t, 0, 30},
 PlotStyle → {{Thickness[0.01], Red}}, AspectRatio → 1/GoldenRatio,
 PlotRange → All, AxesLabel → {"t", "y[t]"}];
```

```
2.2 y[t] + 0.45 y′[t] + y″[t] == 0
y[0] == -1.
y′[0] == 2.
```



A damped oscillator.

If you want to see why this method works, then go on to part iv) below.

If you want another example, go on to the next part.

□B.4.a.ii) A random unforced damped oscillator

Here's an a random unforced damped linear oscillator diffeq:

```
b = Random[Real, {0.1, 0.6}];
c = Random[Real, {1, 3}];
startery = Random[Real, {-2, 2}];
starteryprime = Random[Real, {-2, 2}];

Clear[t, y]
linoscdiffeq = {y''[t] + b y'[t] + c y[t] == 0,
      y[0] == startery, y'[0] == starteryprime};
ColumnForm[Thread[linoscdiffeq]]
```

```
2.38862 y[t] + 0.120775 y′[t] + y″[t] == 0
y[0] == 1.40783
y′[0] == 0.94763
```

The characteristic equation for this oscillator diffeq is:

```
Clear[z]
z^2 + b z + c == 0
```

$2.38862 + 0.120775\, z + z^2 == 0$

The characteristic equation for

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

is

$$z^2 + b\,z + c = 0$$

Use the solutions of the characteristic equation to come up with a formula for the solution of this oscillator diffeq.
Plot the solution and describe what you see.

□Answer:

□Step1: Write down the characteristic equation:

```
Clear[z]
charequation = z^2 + b z + c == 0
```

$2.38862 + 0.120775\, z + z^2 == 0$

Note the relation between the characteristic equation and the oscillator diffeq:

```
ColumnForm[Thread[linoscdiffeq]]
```

```
2.38862 y[t] + 0.120775 y′[t] + y″[t] == 0
y[0] == 1.40783
y′[0] == 0.94763
```

Again: the characteristic equation for

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

is

$$z^2 + b\,z + c = 0$$

□Step 2: Solve the charactistic equation for z to set up the general solution of the diffeq

```
zsols = Solve[charequation, z]
```

```
{{z → -0.0603874 - 1.54434 I}, {z → -0.0603874 + 1.54434 I}}
```

Fish the solutions for z out:

```
z1 = zsols[[1, 1, 2]]
```

```
-0.0603874 - 1.54434 I
```

```
z2 = zsols[[2, 1, 2]]
```

```
-0.0603874 + 1.54434 I
```

You can use the quadratic formula from high school do this with pencil and paper.

Set up the general form of the solutions.

```
Clear[gensol, K1, K2]
gensol[t_] = K1 E^(z1 t) + K2 E^(z2 t)
```

$E^{(-0.0603874 - 1.54434\, I)\, t}\, K1 + E^{(-0.0603874 + 1.54434\, I)\, t}\, K2$

Here K1 and K2 are cleared constants.

□Step 3: Solve for the K1 and K2 that correspond to the given starting data on y[0] and y′[0].

```
ystarteq = gensol[0] == startery;
yprimestarteq = gensol′[0] == starteryprime;
Ksols = Solve[{ystarteq, yprimestarteq}, {K1, K2}]
```

```
{{K1 → 0.703916 + 0.334333 I, K2 → 0.703916 - 0.334333 I}}
```

Substitute these values of K1 and K2 in to get the raw form of the exact formula:

```
gensol[t] /. Ksols[[1]]
```

$(0.703916 + 0.334333\, I)\, E^{(-0.0603874 - 1.54434\, I)\, t} + (0.703916 - 0.334333\, I)\, E^{(-0.0603874 + 1.54434\, I)\, t}$

Make it look nice by hitting gensol[t] with the fundamental identity

$$E^{(a+I\,b)\,t} = E^{a\,t}\,\text{Cos}[b\,t] + I\,E^{a\,t}\,\text{Sin}[b\,t]:$$

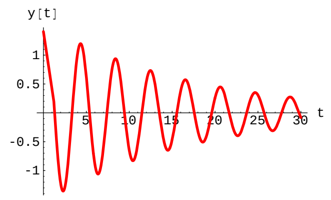
```
Clear[yformula]
yformula[t_] = Chop[ComplexExpand[gensol[t] /. Ksols[[1]]]]
```

$1.40783\, E^{-0.0603874\, t}\, \text{Cos}[1.54434\, t] + 0.668666\, E^{-0.0603874\, t}\, \text{Sin}[1.54434\, t]$

See the solution :

```
ColumnForm[Thread[linoscdiffeq]]

Plot[yformula[t], {t, 0, 30},
 PlotStyle → {{Thickness[0.01], Red}}, AspectRatio → 1/GoldenRatio,
 PlotRange → All, AxesLabel → {"t", "y[t]"}];
```

```
2.38862 y[t] + 0.120775 y′[t] + y″[t] == 0
y[0] == 1.40783
y′[0] == 0.94763
```



A damped oscillator.

If you want to see why this method works, then go on to part iv) below.

If you want another example, go on to the next part.

□B.4.a.iii) A random unforced undamped oscillator

Here's an a random unforced undamped linear oscillator diffeq:

```
b = 0;
c = Random[Real, {1, 3}];
startery = Random[Real, {-2, 2}];
starteryprime = Random[Real, {-2, 2}];

Clear[t, y]
linoscdiffeq = {y''[t] + b y'[t] + c y[t] == 0,
      y[0] == startery, y'[0] == starteryprime};
ColumnForm[Thread[linoscdiffeq]]
```

```
1.26657 y[t] + y″[t] == 0
y[0] == -1.28264
y′[0] == 1.69058
```

The characteristic equation for this oscillator diffeq is:

```
Clear[z]
z^2 + b z + c == 0
1.26657 + z^2 == 0
```

The characteristic equation for

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

is

$$z^2 + b\,z + c = 0$$

Use the solutions of the characteristic equation to come up with a formula for the solution of this oscillator diffeq.
Plot the solution and describe what you see.

□ Answer:

□Step1: Write down the characteristic equation:

```
Clear[z]
charequation = z^2 + b z + c == 0
1.26657 + z^2 == 0
```

Note the relation between the characteristic equation and the oscillator diffeq:

```
ColumnForm[Thread[linoscdiffeq]]
1.26657 y[t] + y''[t] == 0
y[0] == -1.28264
y'[0] == 1.69058
```

Again: the characteristic equation for

$$y''[t] + b\ y'[t] + c\ y[t] = 0$$

is

$$z^2 + b\ z + c = 0.$$

□Step 2: Solve the charactistic equation for z to set up the general solution of the diffeq

```
zsols = Solve[charequation, z]
{{z → 0. - 1.12542 I}, {z → 0. + 1.12542 I}}
```

Fish the solutions for z out:

```
z1 = zsols[[1, 1, 2]]
0. - 1.12542 I
z2 = zsols[[2, 1, 2]]
0. + 1.12542 I
```

You can use the quadratic formula from high school do this with pencil and paper.

Set up the general form of the solutions.

```
Clear[gensol, K1, K2]
gensol[t_] = K1 E^(z1 t) + K2 E^(z2 t)
E^((0.-1.12542 I) t) K1 + E^((0.+1.12542 I) t) K2
```

Here K1 and K2 are cleared constants.

□Step 3: Solve for the K1 and K2 that correspond to the given starting data on y[0] and y′[0].

```
ystarteq = gensol[0] == starter;
yprimestarteq = gensol'[0] == starteryprime;
Ksols = Solve[{ystarteq, yprimestarteq}, {K1, K2}]
{{K1 → -0.64132 + 0.751087 I, K2 → -0.64132 - 0.751087 I}}
```

Substitute these values of K1 and K2 in to get the raw form of the exact formula:

```
gensol[t] /. Ksols[[1]]
(-0.64132 + 0.751087 I) E^((0.-1.12542 I) t) - (0.64132 + 0.751087 I) E^((0.+1.12542 I) t)
```

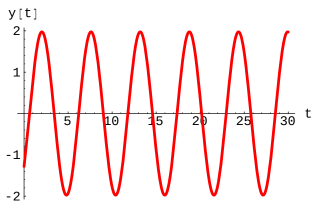
Make it look nice by hitting gensol[t] with the fundamental identity

$$E^{(a+I\,b)\,t} = E^{a\,t}\,\mathrm{Cos}[b\,t] + I\,E^{a\,t}\,\mathrm{Sin}[b\,t]:$$

```
Clear[yformula]
yformula[t_] = Chop[ComplexExpand[gensol[t] /. Ksols[[1]]]]
-1.28264 Cos[1.12542 t] + 1.50217 Sin[1.12542 t]
```

See the solution :

```
ColumnForm[Thread[linoscdiffeq]]

Plot[yformula[t], {t, 0, 30},
  PlotStyle → {{Thickness[0.01], Red}}, AspectRatio → 1/GoldenRatio,
  PlotRange → All, AxesLabel → {"t", "y[t]"}];
1.26657 y[t] + y''[t] == 0
y[0] == -1.28264
y'[0] == 1.69058
```



A undamped unforced oscillator.

If you want to see why this method works, then go on.

□B.4.a.iv)

Why does this method work?

□ Answer:

Start with cleared coefficients b and c:

```
Clear[y, t, b, c]
linoscdiffeq = (y''[t] + b y'[t] + c y[t] == 0)
c y[t] + b y'[t] + y''[t] == 0
```

And do educated guesswork.

Go with numbers K and z and put

$$\mathrm{guessy}[t] = K\,E^{z\,t}$$

Because the diffeq is

$$y''[t] + b\,y'[t] + c\,y[t] = 0,$$

you want to select z so that

$$\mathrm{guessy}''[t] + b\,\mathrm{guess}'[t] + b\,\mathrm{guessy}[t] = 0$$

This is the same as

$$K\,z^2\,E^{z\,t} + b\,z\,K\,E^{z\,t} + c\,K\,E^{z\,t} = 0$$

This is the same as

$$z^2 + b\,z + c = 0.$$

The upshot:

If z1 and z2 solve $z^2 + b\,z + c = 0$, and K1 and K2 are any constants, then

then both

$$K1\,E^{z1\,t} \text{ and } K2\,E^{z2\,t}$$

are guaranteed to solve

$$y''[t] + b\,y'[t] + c\,y[t] = 0.$$

And so is

$$K1\,E^{z1\,t} + K2\,E^{z2\,t}$$

guaranteed to solve

$$y''[t] + b\,y'[t] + c\,y[t] = 0.$$

This leaves you two degrees of freedom to use to set K1 and K2 to solve

$$K1\,E^{z1\,t} + K2\,E^{z2\,t} = \text{given starting value on } y[0]$$

and

$$K1\,z1\,E^{z1\,t} + K2\,z2\,E^{z2\,t} = \text{given starting value on } y'[0].$$

□B.4.a.v)

What happens in the disgusting (and rare) case that the characteristic equation $z^2 + b\,z + c = 0$ has only one solution?

□ Answer:

If the characteristic equation

$z^2 + b\,z + c = 0$ (double root), you take the lone solution z1 and form

$$K1\,E^{z1\,t} + K2\,t\,E^{z1\,t}.$$

This is guaranteed to solve

$$y''[t] + b\,y'[t] + c\,y[t] = 0.$$

This leaves you two degrees of freedom to use to set K1 and K2 to solve to get the right starting values on y[0] and y′[0].

Case in point:

```
b = 6;
c = 9;
startery = Random[Real, {1, 7}];
starteryprime = Random[Real, {-2, 2}];

Clear[t, y]
linoscdiffeq = {y''[t] + b y'[t] + c y[t] == 0,
      y[0] == startery, y'[0] == starteryprime};
ColumnForm[Thread[linoscdiffeq]]
```

```
9 y[t] + 6 y′[t] + y″[t] == 0
y[0] == 6.14121
y′[0] == -1.71868
```

□Step1: Write down the characteristic equation:

```
Clear[z]
charequation = z^2 + b z + c == 0
```

```
9 + 6 z + z^2 == 0
```

Note the relation between the characteristic equation and the oscillator diffeq:

```
ColumnForm[Thread[linoscdiffeq]]
```

```
9 y[t] + 6 y′[t] + y″[t] == 0
y[0] == 6.14121
y′[0] == -1.71868
```

□Step 2: Solve the charactistic equation for z to set up the general solution

```
zsols = Solve[charequation, z]
```

```
{{z → -3}, {z → -3}}
```

Fish the solutions for z out:

```
z1 = zsols[[1, 1, 2]]
```

```
-3
```

```
z2 = zsols[[2, 1, 2]]
```

```
-3
```

Only one solution - this tells you to insert the extra t:

Set up the general form of the solutions.

```
Clear[gensol, K1, K2]
gensol[t_] = K1 E^(z1 t) + K2 t E^(z2 t)
```

```
E^(-3 t) K1 + E^(-3 t) K2 t
```

You insert the extra t because z1 = z2.
Here K1 and K2 are cleared constants.

□Step 3: Solve for the K1 and K2 that correspond to the given starting data on y[0] and y′[0].

```
ystarteq = gensol[0] == startery;
yprimestarteq = gensol′[0] == starteryprime;
Ksols = Solve[{ystarteq, yprimestarteq}, {K1, K2}]
```

```
{{K1 → 6.14121, K2 → 16.7049}}
```

Substitute these values of K1 and K2 in to get the raw form of the exact formula:

```
gensol[t] /. Ksols[[1]]
```

```
6.14121 E^(-3 t) + 16.7049 E^(-3 t) t
```

Make it look nice by hitting gensol[t] with the fundamental identity

$$E^{(a + I b) t} = E^{a t} \mathrm{Cos}[b\,t] + I\,E^{a t}\,\mathrm{Sin}[b\,t]:$$

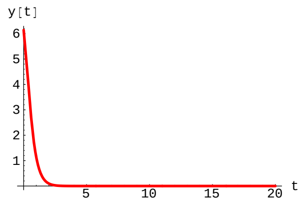
```
Clear[yformula]
yformula[t_] = Chop[ComplexExpand[gensol[t] /. Ksols[[1]]]]
```

```
6.14121 E^(-3 t) + 16.7049 E^(-3 t) t
```

See the solution :

```
ColumnForm[Thread[linoscdiffeq]]

Plot[yformula[t], {t, 0, 20},
  PlotStyle → {{Thickness[0.01], Red}}, AspectRatio → 1/GoldenRatio,
  PlotRange → All, AxesLabel → {"t", "y[t]"}];
```

```
9 y[t] + 6 y′[t] + y″[t] == 0
y[0] == 6.14121
y′[0] == -1.71868
```



Damped out fast!

B.5) Trying to come up with formulas for solutions of the forced oscillator diffeq

y″[t] + b y′[t] + c y[t] = f[t] via convolution integrals

Experience with the basic problems immediately above will keep you from being totally clueless here.

Here's how you use the convolution integral method to go after an exact formula for the oscillator coming from

$$y''[t] + 4.1\,y'[t] + 7.2\,y[t] = f[t]$$

with f[t] = 5 Sin[2 t] , y[0] = 3.0 and y′[0] = 2.2:

□Step 1: Calculate yunforced[t]

This involves coming up with the formula for the unforced solution. This is the solution of the unforced damped oscillator

$$y''[t] + 4.1\,y'[t] + 7.2\,y[t] = 0$$

with y[0] = 3.0 and y′[0] = 2.2.

You already know how to do this:

```
Clear[f, t]
f[t_] = 5 Sin[2 t]; (b = 4.1;)
c = 7.2;
ystarter = 3.0;
yprimestarter = 2.2;
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
```

```
zsols = Solve[charequation, z];
generalsol[t_] =
  C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]}
```

```
C1 E^((-2.05-1.73133 I) t) + C2 E^((-2.05+1.73133 I) t)
```

```
Csols =
  Solve[{generalsol[0] == ystarter, generalsol′[0] == yprimestarter}];
Clear[yunforced]
yunforced[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
3. E^(-2.05 t) Cos[1.73133 t] + 4.82288 E^(-2.05 t) Sin[1.73133 t]
```

Some scientists like to call this the transient solution.

You can see the effect of the damping term on this formula.
Put yunforced[t] aside for a moment.

□Step 2: Calculate the unit impulse response, yunitimpulse[t]

This involves coming up with the formula of the solution of the unforced oscillator

$$y''[t] + 4\,y'[t] + 5\,y[t] = 0$$

with y[0] = 0 and y′[0] = 1.

You already know how to do this:

```
Csols = Solve[{generalsol[0] == 0, generalsol′[0] == 1}];
Clear[yunitimpulse]
yunitimpulse[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
0.577591 E^(-2.05 t) Sin[1.73133 t]
```

□Step 3: Calculate

$$\mathbf{yzeroinput[t]} = \int_0^t \mathbf{yunitimpulse[t - x]\,f[x]}\,dx$$

This involves going after a formula for the zero input solution, yzeroinput[t], of the forced oscillator

$$y''[t] + 4\,y'[t] + 5\,y[t] = f[t]$$

with zeroed out starter data y[0] = 0 and y′[0] = 0:

A miracle of calculus (to be explained later) is that you can get a formula for yzeroinput[t] by setting

$$\mathrm{yzeroinput}[t] = \int_0^t \mathrm{yunitimpulse}[t - x]\,f[x]\,dx$$

```
Clear[yzeroinput, x]
yzeroinput[t_] = Apart[Chop[∫_0^t yunitimpulse[t - x] f[x] dx]]
```

```
0.529169 E^(-2.05 t) (1. Cos[1.73133 t] + 0.733259 Sin[1.73133 t]) -
 0.529169 (1. Cos[2. t] - 0.390244 Sin[2. t])
```

Folks call this integral by the name "convolution integral."
This is how the convolution integral method got its name.

Looks good.

□Step 4: Set yformulay[t] = yzeroinput[t] + yunforced[t]:

Get your shot at an exact formula for the solution of

$$y''[t] + 4\,y'[t] + 5\,y[t] = f[t]$$

with f[t] = 5 Sin[t], y[0] = 3 and y′[0] = 2
by putting

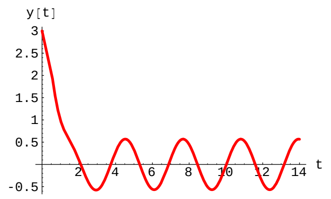
formulay[t] = yzeroinput[t] + yunforced[t]:

```
Clear[yformula]
yformula[t_] = Expand[yzeroinput[t] + yunforced[t]]
```

```
3.52917 E^(-2.05 t) Cos[1.73133 t] - 0.529169 Cos[2. t] +
 5.2109 E^(-2.05 t) Sin[1.73133 t] + 0.206505 Sin[2. t]
```

Done!
See it:

```
Plot[yformula[t], {t, 0, 14}, PlotStyle → {{Thickness[0.01], Red}},
 AspectRatio → 1/GoldenRatio, AxesLabel → {"t", "y[t]"}];
```



You can see that the forcing function f[t] = 5 Sin[2 t] controls the steady state behavior.

□B.5.a)

Use the convolution integral method to go after an exact formula for the oscillator coming from

$$y''[t] + 2.4\,y'[t] + 6.7\,y[t] = f[t]$$

with f[t] = 20 DiracDelta[t − 4], y[0] = 8.0 and y′[0] = −2.7.

□ **Answer:**

Here you go:

Copy paste and edit to get:

□Step 1: Calculate yunforced[t]

This involves coming up with the formula for the unforced solution.
This is the solution of the unforced damped oscillator

$$y''[t] + 2.4\,y'[t] + 6.7\,y[t] = 0$$

with y[0] = 8.0 and y′[0] = −2.7.
You already know how to do this:

```
Clear[f, t]
f[t_] = 30 DiracDelta[t - 6];
b = 2.4;
c = 6.7;
ystarter = 8.0;
yprimestarter = -2.7;
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
 C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]}
```

```
C1 E^((-1.2-2.29347 I) t) + C2 E^((-1.2+2.29347 I) t)
```

```
Csols =
 Solve[{generalsol[0] == ystarter, generalsol′[0] == yprimestarter}];
Clear[yunforced]
yunforced[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
8. E^(-1.2 t) Cos[2.29347 t] + 3.00854 E^(-1.2 t) Sin[2.29347 t]
```

Some scientists like to call this the transient solution.

You can see the effect of the damping term on this formula.
Put yunforced[t] aside for a moment.

□Step 2: Calculate the unit impulse response, yunitimpulse[t]

This involves coming up with the formula of the solution of the unforced oscillator

$$y''[t] + 2.4\,y'[t] + 6.7\,y[t] = 0$$

with y[0] = 0 and y′[0] = 1.
You already know how to do this:

```
Csols = Solve[{generalsol[0] == 0, generalsol′[0] == 1}];
Clear[yunitimpulse]
yunitimpulse[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
0.436021 E^(-1.2 t) Sin[2.29347 t]
```

□Step 3: Calculate

$$\textbf{yzeroinput[t]} = \int_0^t \textbf{yunitimpulse[t − x] f[x]}\,dx$$

This involves going after a formula for the zero input solution, yzeroinput[t], of the forced oscillator

$$y''[t] + 2.4\,y'[t] + 6.7\,y[t] = f[t]$$

with zeroed out starter data y[0] = 0 and y′[0] = 0:
A miracle of calculus (to be explained later) is that you can get a formula for yzeroinput[t] by setting

$$\text{yzeroinput[t]} = \int_0^t \text{yunitimpulse[t − x] f[x]}\,dx$$

```
Clear[yzeroinput, x]
yzeroinput[t_] = Apart[Chop[∫_0^t yunitimpulse[t - x] f[x] dx]]
```

```
17520.6 E^(-1.2 t) Sin[2.29347 (-6 + t)] UnitStep[-6 + t]
```

Folks call this integral by the name "convolution integral."
This is how the convolution integral method got its name.

Looks good.

□Step 4: Set yformulay[t] = yzeroinput[t] + yunforced[t]:

Get your shot at an exact formula for the solution of

$$y''[t] + 2.4\,y'[t] + 6.7\,y[t] = f[t]$$

with f[t] = 20 DiracDelta[t − 4], y[0] = 8.0 and y′[0] = −2.7.
by putting

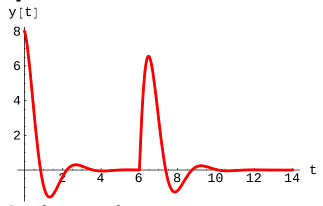
yformula[t] = yzeroinput[t] + yunforced[t]:

```
Clear[yformula]
yformula[t_] = Expand[yzeroinput[t] + yunforced[t]]
```

```
8. E^(-1.2 t) Cos[2.29347 t] + 3.00854 E^(-1.2 t) Sin[2.29347 t] +
 17520.6 E^(-1.2 t) Sin[2.29347 (-6 + t)] UnitStep[-6 + t]
```

Done!
See it:

```
Plot[yformula[t], {t, 0, 14},
 PlotStyle → {{Thickness[0.01], Red}}, AspectRatio → 1/GoldenRatio,
 PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



The forcing function

f[t] = 20 DiracDelat[t – 6]

goosed the oscillator at t = 6.

□B.5.b)

Why do lots of old timers (those old profs who wear bowties) like to say that the convolution integral method is the method of dividing and conquering?

□ **Answer:**

The goal is to come up with a formula for the oscillator coming from

$$y''[t] + b\,y'[t] + c\,y[t] = f[t]$$

with given starting data

y[0] = p and y′[0] = q.

The convolution integral method divides the problem in two pieces:

yunforced[t] solves

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

with given starting data

y[0] = p and y′[0] = q.

yzeroinput[t] solves

$y''[t] + b\,y'[t] + c\,y[t] = f[t]$

with starting data

$y[0] = 0$ and $y'[0] = 0$.

The method conquers by putting

yformula[t] = yzeroinput[t] + yunforced[t]

which solves

$y''[t] + b\,y'[t] + c\,y[t] = f[t] + 0 = f[t]$

with given starting data

$y[0] = p + 0 = p$ and $y'[0] = q + 0 = q$.

Kinda neat and kinda sweet.

□B.5.c)

What can break the back of the convolution integral method?

□ **Answer:**

When you go with the convolution integral method to come up with a formula for the solution of

$y''[t] + b\,y'[t] + c\,y[t] = f[t]$

with given starting data

$y[0] = p$ and $y'[0] = q$,

the machine has to be able to do the convolution integral

$\text{yzeroinput}[t] = \int_0^x \text{yunitimpulse}[t - x]\, f[x]\, dt$

where yunitimpulse[t] solves

$y''[t] + b\,y'[t] + c\,y[t] = 0$

with $y[0] = 0$ and $y'[0] = 1$.

If f[t] is nasty enough, this integral can become painful or plain impossible.

```
∫₀ᵗ f[x] yunitimpulse′[t − x] ⅆx + f[t] yunitimpulse[0]
```

```
yzeroinput''[t]
```

```
∫₀ᵗ f[x] yunitimpulse″[t − x] ⅆx + yunitimpulse[0] f′[t] +
 f[t] yunitimpulse′[0]
```

Remembering that yunitimpulse[0] = 0 and yunitimpulse′[0] = 1 , look at

$\text{yzeroinput}''[t] + b\,\text{yzeroinput}'[t] + c\,\text{yzeroinput}[t]$:

```
(yzeroinput''[t] + b yzeroinput'[t] + c yzeroinput[t]) /.
 {yunitimpulse[0] -> 0, yunitimpulse'[0] -> 1}
```

```
f[t] + c ∫₀ᵗ f[x] yunitimpulse[t − x] ⅆx + b ∫₀ᵗ f[x] yunitimpulse′[t − x] ⅆx +
 ∫₀ᵗ f[x] yunitimpulse″[t − x] ⅆx
```

This tells you that

$\text{yzeroinput}''[t] + b\,\text{yzeroinput}'[t] + c\,\text{yzeroinput}[t]$

is the same as

$f[t] + \int_0^t f[x]\,(\text{yunitimpulse}''[t - x] + b\,\text{yunitimpulse}'[t - x] + c\,\text{yunitimpulse}[t - x])\, dx.$

This collapses to

$f[t] + \int_0^t f[x]\,(0)\, dx = f[t].$

Because

$(\text{yunitimpulse}''[t] + b\,\text{yunitimpulse}'[t] + c\,\text{yunitimpulse}[t]) = 0$.

The upshot:

$\text{yzeroinput}''[t] + b\,\text{yzeroinput}'[t] + c\,\text{yzeroinput}[t]$

is guaranteed to be f[t].

□B.5.d) The calculus behind the convolution integral method

What guarantees that the convolution integral method always works when the integrals can be done?

□ **Answer:**

This explanation is mainly for enthusiasts.

Calculus does!

There is only one issue to explain.

The issue is to explain why

$\text{yzeroinput}[t] = \int_0^t \text{yunitimpulse}[t - x]\, f[x]\, dx$

solves

$y''[t] + b\,y'[t] + c\,y[t] = f[t]$

with $y[0] = 0$ and $y'[0] = 0$.

To this end, remember that yunitimpulse[t] solves

$y''[t] + b\,y'[t] + c\,y[t] = 0$

with $y[0] = 0$ and $y'[0] = 1$;

so that

→ yunitimpulse[0] = 0

→ yunitimpulse′[0] = 1 and

→ $\text{yunitimpulse}''[t] + b\,\text{yunitimpulse}'[t] + c\,\text{yunitimpulse}[t] = 0$.

Enter yzeroinput[t] into *Mathematica:*

```
Clear[f, t, yunitimpulse, yzeroinput, y, b, c, x]
 yzeroinput[t_] = ∫₀ᵗ yunitimpulse[t − x] f[x] ⅆx
```

```
∫₀ᵗ f[x] yunitimpulse[t − x] ⅆx
```

Let Mathematica calculate yzeroinput′[t] and yzeroinput″[t]:

```
yzeroinput'[t]
```

This explains why

$\text{yzeroinput}[t] = \int_0^t f[x]\,\text{yunitimpulse}[t - x]\, dx$

is guaranteed to solve

$y''[t] + b\,y'[t] + c\,y[t] = f[t]$.

Now check whether yzeroinput[0] is guaranteed to be 0:

```
yzeroinput[t]
```

```
∫₀ᵗ f[x] yunitimpulse[t − x] ⅆx
```

When you plug in t = 0, you get:

```
yzeroinput[0]
```

```
0
```

This checks.

Finally check whether yzeroinput′[0] is guaranteed to be 0:

```
yzeroinput'[t]
```

```
∫₀ᵗ f[x] yunitimpulse′[t − x] ⅆx + f[t] yunitimpulse[0]
```

When you plug in t = 0, you get:

```
yzeroinput'[0]
```

```
f[0] yunitimpulse[0]
```

And this is zero because

yunitimpulse[0] = 0.

Explanation complete and you're out of here.

If you like this verification, there are many parts of higher math you will also like.

DE.02 Transition from Calculus to DiffEq: The Forced Oscillator DiffEq Tutorials

T.1) Amplitude and frequency of unforced oscillators

□T.1.a) Amplitude and frequency of unforced damped oscillators:

You are in the lab and are in the process of coming up with a formula for this given damped oscillator with these given starter values on y[0] and y′[0]:

```
Clear[y, t]
b = 0.8;
c = 24.0;
linoscdiffeq = y″[t] + b y′[t] + c y[t] == 0
startery = 5.9;
starteryprime = -0.4;
```

```
24. y[t] + 0.8 y′[t] + y″[t] == 0
```

You copy, paste and edit from the Basics:

```
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];

Clear[gensol, C1, C2]
gensol[t_] = C1 E^(zsols[[1,1,2]] t) + C2 E^(zsols[[2,1,2]] t);

ystarteq = gensol[0] == startery;
yprimestarteq = gensol′[0] == starteryprime;
Csols = Solve[{ystarteq, yprimestarteq}, {C1, C2}];

Clear[yformula]
yformula[t_] = Chop[ComplexExpand[gensol[t] /. Csols[[1]]]]
```

```
5.9 E^(-0.4 t) Cos[4.88262 t] + 0.401424 E^(-0.4 t) Sin[4.88262 t]
```

Your job is to calculate the amplitude and the frequency for this damped unforced oscillator. Do it and show off your work with convincing plots.

□Answer:

Look at:

```
yformula[t]
```

```
5.9 E^(-0.4 t) Cos[4.88262 t] + 0.401424 E^(-0.4 t) Sin[4.88262 t]
```

Read off:

```
Clear[amplitude, t]
amplitude[t_] = Sqrt[5.9^2 + 0.401424^2] E^(-0.4 t)
```

```
5.91364 E^(-0.4 t)
```

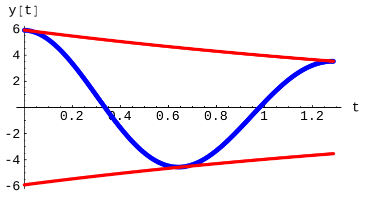
```
frequency = N[4.88262/(2 π)]
```

```
0.777093
```

This damped oscillator goes through about 0.8 oscillations per time unit.

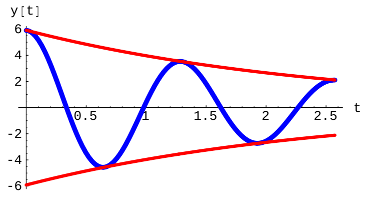
See 1 complete oscillation:

```
endtime = 1/frequency;
oscplot = Plot[{yformula[t], amplitude[t], -amplitude[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.015], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Red}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 100,
  AspectRatio → 1/2];
```



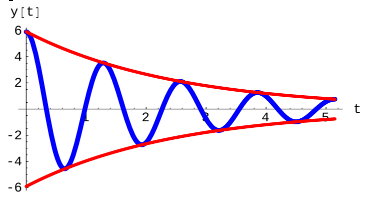
See 2 complete oscillations:

```
endtime = 2/frequency;
oscplot = Plot[{yformula[t], amplitude[t], -amplitude[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.015], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Red}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 100,
  AspectRatio → 1/2];
```



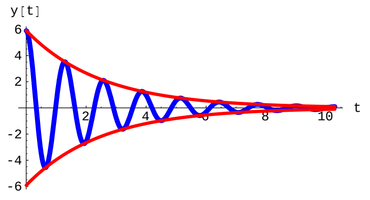
See 4 complete oscillations:

```
endtime = 4/frequency;
oscplot = Plot[{yformula[t], amplitude[t], -amplitude[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.015], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Red}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 100,
  AspectRatio → 1/2];
```



See 8 complete oscillations:

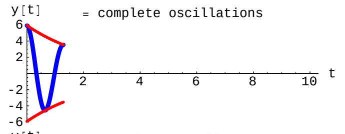
```
endtime = 8/frequency;
oscplot = Plot[{yformula[t], amplitude[t], -amplitude[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.015], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Red}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 100,
  AspectRatio → 1/2];
```

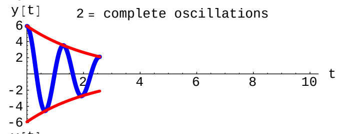


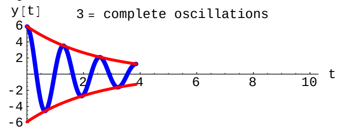
Petering out.

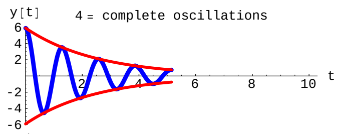
Throw in a movie:

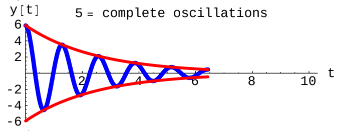
```
endtime = 8/frequency;
Clear[oscplotter, s]
oscplotter[s_] := Plot[{yformula[t], amplitude[t], -amplitude[t]},
  {t, 0, s/frequency}, PlotStyle → {{Thickness[0.015], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Red}},
  PlotRange → {{0, endtime}, {-amplitude[0] - 0.1, amplitude[0] + 0.1}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 100,
  PlotLabel → s "= complete oscillations", AspectRatio → 1/3];
Table[oscplotter[s], {s, 1, 8}];
```

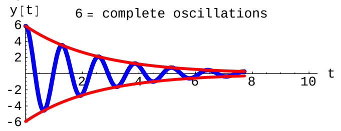


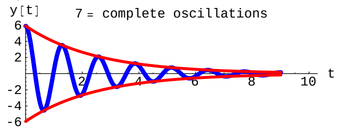


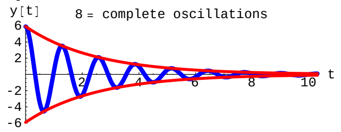












Grab and animate slowly.

□T.1.b) Amplitude and frequency of unforced undamped oscillators

Will the same thing work for undamped unforced oscillators?

□Answer:

Yah sure! You bettcha!

Copy, paste, edit and put it to work on this one:

```
Clear[y, t]
b = 0;
c = 32.7;
```

```
linoscdiffeq = y''[t] + b y'[t] + c y[t] == 0
starteryy = 12.7;
starteryprime = 3.1;
```

```
32.7 y[t] + y''[t] == 0
```

```
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];

Clear[gensol, C1, C2]
gensol[t_] = C1 E^(zsols[[1,1,2]] t) + C2 E^(zsols[[2,1,2]] t);
ystarteq = gensol[0] == startery;
yprimestarteq = gensol'[0] == starteryprime;
Csols = Solve[{ystarteq, yprimestarteq}, {C1, C2}];

Clear[yformula]
yformula[t_] = Chop[ComplexExpand[gensol[t] /. Csols[[1]]]]
```

```
12.7 Cos[5.71839 t] + 0.54211 Sin[5.71839 t]
```

Read off:

```
Clear[amplitude, t]
amplitude[t_] = Sqrt[12.7^2 + 0.54211^2]
```

```
12.7116
```

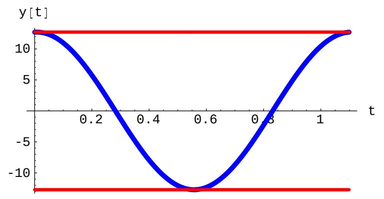
```
frequency = N[5.71839/(2 π)]
```

```
0.91011
```

This oscillator goes through about almost 1 complete oscillation in one time unit.

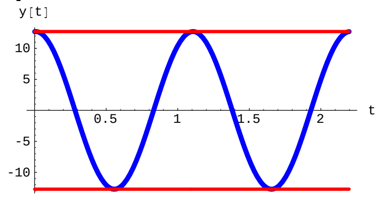
See 1 complete oscillation:

```
endtime = 1/frequency;
oscplot = Plot[{yformula[t], amplitude[t], -amplitude[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.015], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Red}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 100,
  AspectRatio → 1/2];
```



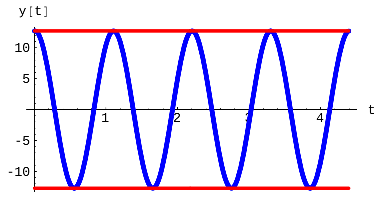
See 2 complete oscillations:

```
endtime = 2/frequency;
oscplot = Plot[{yformula[t], amplitude[t], -amplitude[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.015], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Red}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 100,
  AspectRatio → 1/2];
```



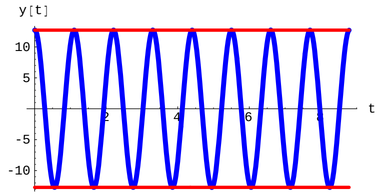
See 4 complete oscillations:

```
endtime = 4/frequency;
oscplot = Plot[{yformula[t], amplitude[t], -amplitude[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.015], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Red}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 100,
  AspectRatio → 1/2];
```



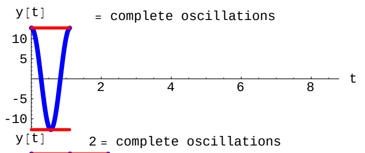
See 8 complete oscillations:

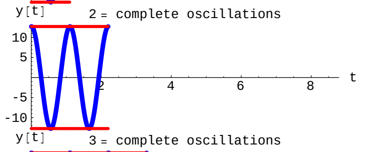
```
endtime = 8/frequency;
oscplot = Plot[{yformula[t], amplitude[t], -amplitude[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.015], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Red}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 100,
  AspectRatio → 1/2];
```

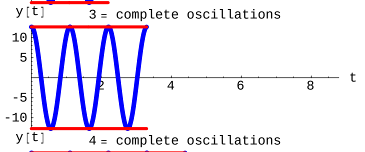


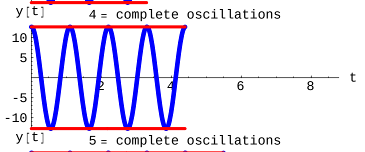
Throw in a movie:

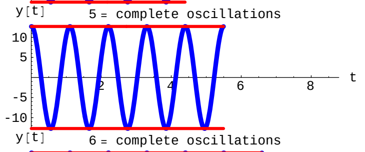
```
endtime = 8/frequency;
Clear[oscplotter, s]
oscplotter[s_] := Plot[{yformula[t], amplitude[t], -amplitude[t]},
  {t, 0, s/frequency}, PlotStyle → {{Thickness[0.015], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Red}},
  PlotRange → {{0, endtime}, {-amplitude[0] - 0.1, amplitude[0] + 0.1}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 100,
  PlotLabel → s "= complete oscillations", AspectRatio → 1/3];
Table[oscplotter[s], {s, 1, 8}];
```

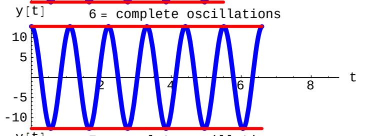


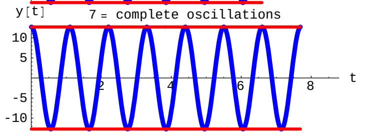


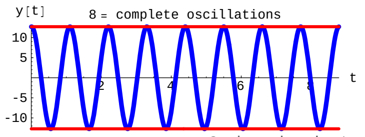












Grab and animate slowly.

T.2) Impulse-forced oscillators: The physical meaning of the impulse hit

◻T.2.a.i)

Showcase the convolution integral method and the Dirac delta function by coming up with the exact formula for the forced oscillator coming from

$$y''[t] + 0.9\,y'[t] + 4.1\,y[t] = f[t]$$

with f[t] = 10 DiracDelta[t – 7], y[0] = 2.0 and y′[0] = 7.1.
Show off your work with a good plot.
Describe what you see.

◻**Answer:**

◻Step 1: Calculate yunforced[t]

```
Clear[f, t]
f[t_] = 10 DiracDelta[t - 7];
b = 0.9;
c = 4.1;
ystarter = 2.0;
yprimestarter = 7.1
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
 C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]}
7.1
C1 E^((-0.45-1.97421 I) t) + C2 E^((-0.45+1.97421 I) t)
```

```
Csols =
 Solve[{generalsol[0] == ystarter, generalsol′[0] == yprimestarter}];
Clear[yunforced]
yunforced[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
2. E^(-0.45 t) Cos[1.97421 t] + 4.05226 E^(-0.45 t) Sin[1.97421 t]
```

◻Step 2: Calculate the unit impulse response, yunitimpulse[t]

```
Ksols = Solve[{generalsol[0] == 0, generalsol′[0] == 1}];
Clear[yunitimpulse]
yunitimpulse[t_] = Chop[ComplexExpand[generalsol[t] /. Ksols[[1]]]]
0.506532 E^(-0.45 t) Sin[1.97421 t]
```

◻Step 3: Calculate

$$\textbf{yzeroinput[t]} = \int_0^t \textbf{yunitimpulse[t} - \textbf{x] f[x]}\, dx$$

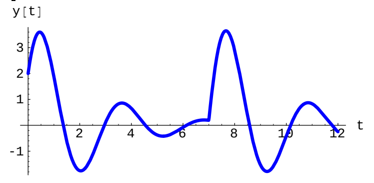
```
Clear[yzeroinput, x]
yzeroinput[t_] = Apart[Chop[∫_0^t yunitimpulse[t - x] f[x] dx]]
118.205 E^(-0.45 t) Sin[1.97421 (-7 + t)] UnitStep[-7 + t]
```

◻Step 4: Set formulay[t] = yzeroinput[t] + yunforced[t]:

```
Clear[formulay]
formulay[t_] = Expand[yzeroinput[t] + yunforced[t]]
2. E^(-0.45 t) Cos[1.97421 t] + 4.05226 E^(-0.45 t) Sin[1.97421 t] +
 118.205 E^(-0.45 t) Sin[1.97421 (-7 + t)] UnitStep[-7 + t]
```

And a plot:

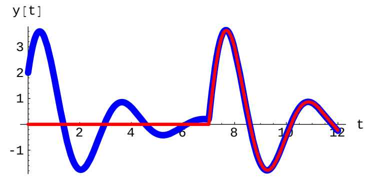
```
oscplot = Plot[formulay[t], {t, 0, 12},
  PlotStyle → {{Blue, Thickness[0.01]}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}];
```



Just when this oscillator seemed to be settling down, the impulse hit at t = 7 goosed this oscillator.

For the fun of it, look at this plot of yformula[t] and yzeroinput[t]:

```
Plot[{formulay[t], yzeroinput[t]}, {t, 0, 12},
 PlotStyle → {{Blue, Thickness[0.02]}, {Red, Thickness[0.01]}},
 PlotRange → All, AxesLabel → {"t", "y[t]"}];
```



Think about what this plot is trying to tell you.

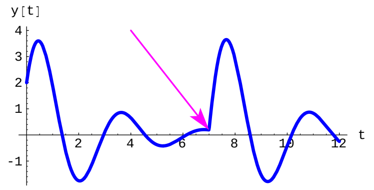
◻T.2.a.ii) Impulses and Dirac Delta Functions

Take another peek at the plot of the impulse-forced oscillator coming from

$$y''[t] + 0.9\,y'[t] + 4.1\,y[t] = f[t]$$

with f[t] = 10 DiracDelta[t – 7], y[0] = 2.0 and y′[0] = 7.1.
Show off your work with a good plot.
Describe what you see.

```
tail = {4, 4};
Show[oscplot,
 Arrow[{7, formulay[7]} - tail, Tail → tail, VectorColor → Magenta]];
```



Explain why the oscillator had a sudden change of direction at t = 7.

◻**Answer:**

Look at the differential equation

$$y''[t] + 0.9\,y'[t] + 4.1\,y[t] = 10\,\text{DiracDelta}[t - 7]$$

and remember that

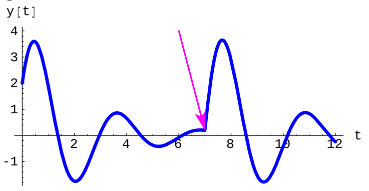
$$\text{DiracDelta}[t - 7] = 0 \quad \text{for t not equal to 7.}$$

So, aside from the what happens at t = 7, this oscillator behaves just as an oscillator coming from

$$y''[t] + 0.9\, y'[t] + 4.1\, y[t] = 0.$$

Look again:

```
tail = {6, 4};
Show[oscplot,
 Arrow[{7, formulay[7]} - tail, Tail → tail, VectorColor → Magenta]];
```



The differential equation

$$y''[t] + 0.9\, y'[t] + 4.1\, y[t] = 10\, \text{DiracDelta}[t - 7]$$

tells you that the forcing term

$$10\, \text{DiracDelta}[t - 7]$$

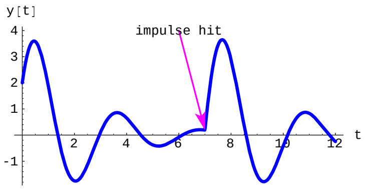
packs all its punch at t = 7 and abruptly redirects the oscillator from its original path to a new (unforced) path.

This is in direct response to the impulse force that

$$10\, \text{DiracDelta}[t - 7]$$

lays on the oscillator at t = 7.

```
Show[oscplot,
 Arrow[{7, formulay[7]} - tail, Tail → tail, VectorColor → Magenta],
 Graphics[Text["impulse hit", tail]]];
```



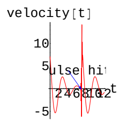
T.2.a.iii) Physical meaning of the impulse hit

Stay with the same setup:

$$y''[t] + 0.9\, y'[t] + 4.1\, y[t] = f[t]$$

with $f[t] = 10\, \text{DiracDelta}[t - 7]$, $y[0] = 2.0$ and $y'[0] = 7.1$.
and look at this plot of the derivative of the solution:

```
velocityplot =
 Plot[formulay'[t], {t, 0, 12}, PlotStyle → {{Red, Thickness[0.01]}},
  AxesLabel → {"t", "velocity[t]"}, DisplayFunction → Identity];

tail = {4, 4};
label = Graphics[{Text["Impulse hit", tail]}];
pointer = Arrow[{6.999, formulay'[6.999]} - tail, Tail → tail];

setup = Show[velocityplot, label, pointer,
  DisplayFunction → $DisplayFunction];
```



Use this plot to interpret the physical meaning of the impulse hit with
f[t] = 10 DiracDelta[t – 7] at t = 7.

Answer:

The impulse hit with

$$f[t] = 10\, \text{DiracDelta}[t - 7]$$

instantaneously jumps the velocity up by 10 at the instant of the impulse hit at t = 7.

T.2.a.iv)

Go with d > 0, and go with

$$y''[t] + b\, y'[t] + c\, y[t] = K\, \text{DiracDelta}[t - d]$$

with given starting values on y[0] and y′[0].
The experiment above sends the message that the impulse hit with

$$f[t] = K\, \text{DiracDelta}[t - d]$$

instantaneously jumps

$$y'[t] \text{ up by } K$$

at the instant of the impulse hit at t = d.
Explain why this is guaranteed.

Answer:

Go with h > 0 and integrate both sides of

$$y''[t] + b\, y'[t] + c\, y[t] = K\, \text{DiracDelta}[t - d]$$

from d – h to d + h to get

$$\int_{d-h}^{d+h} (y''[t] + b\, y'[t] + c\, y[t])\, dt$$
$$= K \int_{d-h}^{d+h} \text{DiracDelta}[t - d]\, dt$$
$$= K$$

Reason: DiracDelta[t – d] integrates out to 1 for any interval containing d.

This is the same as

$$\int_{d-h}^{d+h} y''[t]\, dt + \int_{d-h}^{d+h} b\, y'[t]\, dt + \int_{d-h}^{d+h} c\, y[t]\, dt = K$$

As h closes in on 0,

$$\int_{d-h}^{d+h} c\, y[t]\, dt \to 0$$

and

$$\int_{d-h}^{d+h} b\, y'[t]\, dt \to 0.$$

Reason: This is guaranteed because neither y[t] or y′[t] can blow up to infinity or down to -infinity

Consequently, as h closes in on 0,

$$\int_{d-h}^{d+h} y''[t]\, dt \to K.$$

But

$$\int_{d-h}^{d+h} y''[t]\, dt = y'[d + h] - y'[d - h]$$

So as h closes in on 0,

$$y'[d + h] - y'[d - h] \to K.$$

The upshot:

y′[t] jumps by K at t = d.

T.3) Convolutions involving DiracDelta[x – d] are easy to do by hand

T.3.a.i)

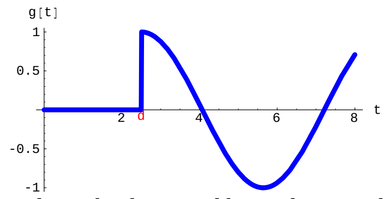
Look at this experiment involving a convolution integral:

```
Clear[f, g, x, t]
d = 2.5;
f[x_] = Cos[x];
g[t_] = ∫_0^t f[t - x] DiracDelta[x - d] dx
```

```
Cos[2.5 - t] UnitStep[-2.5 + t]
```

And plot:

```
gplot = Plot[g[t], {t, 0, 8},
  PlotStyle -> {{Thickness[0.015], Blue}},
   AspectRatio -> 1/2, AxesLabel -> {"t", "g[t]"}, PlotRange -> All,
   Epilog -> {Red, Text["d", {d, -0.08}]}];
```



Explain what happened by explaining why

$$\int_0^t f[t-x]\,\text{DiracDelta}[x-d]\,dx = f[t-d]\,\text{UnitStep}[t-d].$$

□ **Answer:**

Because DiracDelta[x – d] = 0 for x < d, you are guaranteed that

$$g[t] = \int_0^t f[t-x]\,\text{DiracDelta}[x-d]\,dx = 0 \text{ for } t < d.$$

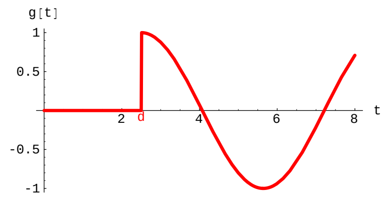
But for t > d,

$$g[t] = \int_0^t f[t-x]\,\text{DiracDelta}[x-d]\,dx = f[t-d].$$

So g[t] = f[t – d] UnitStep[t – d].

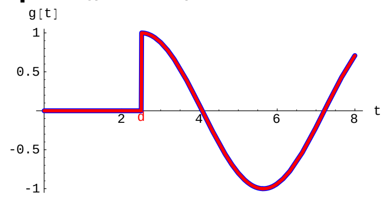
See the plot of f[t – d] UnitStep[t – d] :

```
thisplot = Plot[f[t - d] UnitStep[t - d], {t, 0, 8},
   PlotStyle → {{Thickness[0.01], Red}}, AspectRatio → 1/2,
   AxesLabel → {"t", "g[t]"}, Epilog → {Red, Text["d", {d, -0.08}]}];
```



Compare with the original:

```
Show[gplot, thisplot];
```



Just as theory predicted, they are the same!

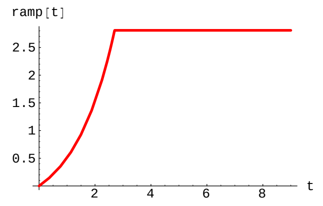
T.4) Ramp-forced oscillators

□T.4.a)

Here is a ramp forcing function and its plot:

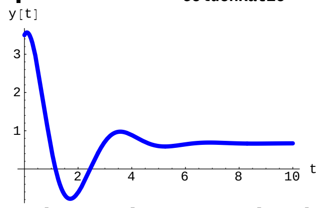
```
Clear[ramp, g, t]
g[t_] = 0.5 (E^(0.7 t) - 1);
changeover = 2.7;
ramp[t_] =
  g[t] UnitStep[changeover - t] + g[changeover] UnitStep[t - changeover]
Plot[ramp[t], {t, 0, 9}, PlotStyle → {{Thickness[0.01], Red}},
  PlotRange → All, AxesLabel → {"t", "ramp[t]"},
  AspectRatio → 1/GoldenRatio];
```

```
0.5 (-1 + E^(0.7 t)) UnitStep[2.7 - t] + 2.80968 UnitStep[-2.7 + t]
```



Ramp[t] runs
→ with $g[t] = 0.5\,(E^{0.7\,t} - 1)$ for t < 2.7
→ with g[2.7] for t > 2.7.
Here is a the plot of a damped oscillator forced by ramp[t]:

```
Clear[y, t, f]
f[t_] = ramp[t];
b = 1.6;
c = 4.2;
ystarter = 3.5;
yprimestarter = 1.5;
diffeq = y″[t] + b y′[t] + c y[t] == f[t];

endtime = 10;
solution = NDSolve[{diffeq, y[0] == ystarter,
    y′[0] == yprimestarter}, y[t], {t, 0, endtime}];

approxy[t_] = y[t] /. solution[[1]];
ndsplot = Plot[approxy[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.015], Blue}}, PlotRange → All,
  AspectRatio → 1/GoldenRatio, AxesLabel → {"t", "y[t]"}];
```



Use the convolution integral method to produce an exact formula for this ramp-forced oscillator.

□ **Answer:**

Here you go:

□Step 1: Calculate yunforced[t]

```
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
 C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]}
```

```
C1 E^((-0.8-1.8868 I) t) + C2 E^((-0.8+1.8868 I) t)
```

```
Csols =
 Solve[{generalsol[0] == ystarter, generalsol′[0] == yprimestarter}];
Clear[yunforced]
yunforced[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
3.5 E^(-0.8 t) Cos[1.8868 t] + 2.279 E^(-0.8 t) Sin[1.8868 t]
```

□Step 2: Calculate the unit impulse response, yunitimpulse[t]

```
Csols = Solve[{generalsol[0] == 0, generalsol′[0] == 1}];
Clear[yunitimpulse]
yunitimpulse[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
0.529999 E^(-0.8 t) Sin[1.8868 t]
```

□Step 3: Calculate

$$\text{yzeroinput}[t] = \int_0^t \text{unitimpulse}[t-x]\,f[x]\,dx$$

```
Clear[yzeroinput, x]
yzeroinput[t_] = Chop[Apart[Chop[∫_0^t yunitimpulse[t - x] f[x] dx]]]
```

```
0.0860585 (-1.38333 UnitStep[2.7 - t] +
    1. E^(0.7 t) UnitStep[2.7 - t] + 7.77346 UnitStep[-2.7 + t]) -
 3.90726 E^(-0.8 t) (-1. Cos[5.09435 - 1.8868 t] - 0.00844302 Cos[1.8868 t] +
    0.893014 Sin[5.09435 - 1.8868 t] + 0.00459153 Sin[1.8868 t] +
    1. Cos[5.09435 - 1.8868 t] UnitStep[2.7 - t] -
    0.893014 Sin[5.09435 - 1.8868 t] UnitStep[2.7 - t] +
    1.48461 Cos[5.09435 - 1.8868 t] UnitStep[-2.7 + t] -
    0.629473 Sin[5.09435 - 1.8868 t] UnitStep[-2.7 + t])
```

Pay no attention to any error messages.

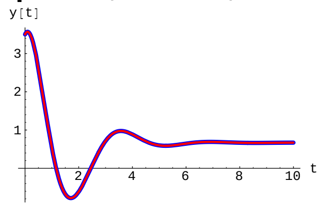
□Step 4: Set formulay[t] = yzeroinput[t] + yunforced[t]:

```
Clear[formulay]
formulay[t_] = Simplify[yzeroinput[t] + yunforced[t]]
```

```
-0.119048 E^(-0.8 t) (-32.821 Cos[5.09435 - 1.8868 t] -
   29.6771 Cos[1.8868 t] + 29.3096 Sin[5.09435 - 1.8868 t] -
18.9929 Sin[1.8868 t] + (1. E^(0.8 t) - 0.722892 E^(1.5 t) +
   32.821 Cos[5.09435 - 1.8868 t] - 29.3096 Sin[5.09435 - 1.8868 t])
 UnitStep[2.7 - t] + (-5.61937 E^(0.8 t) +
   48.7263 Cos[5.09435 - 1.8868 t] - 20.6599 Sin[5.09435 - 1.8868 t])
 UnitStep[-2.7 + t])
```

Here is the formula plot together with the NDSolve plot:

```
formulaplot = Plot[formulay[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.008], Red}}, DisplayFunction → Identity];
Show[ndsplot, formulaplot];
```



Perfecto.

T.5) Electrical folks and the forced oscillator

□T.5.a)

Lots of folks other than spring specialists are interested in the oscillator. In fact, electrical folks really get into the oscillator in a big way. Why?

□**Answer:**

When spring specialists think of the forced oscillator differential equation

$$y''[t] + b\,y'[t] + c\,y[t] = f[t],$$

they think of:

→ t as time

→ y[t] as displacement from mass stretched position,

→ b as a drag measurement,

→ c as a measurement of the stiffness of the spring, and

→ f[t] as an externally applied force which varies with time.

When the electrical specialist thinks of the the oscillator differential equation

$$y''[t] + b\,y'[t] + c\,y[t] = f[t],$$

they think of:

→ t as time

→ y[t] as voltage

→ b as a measurement of resistance,

→ c as a measurement of impedance and

→ f[t] as a measurement of electromotive force.

Don't worry about these words if they are unfamiliar to you.

DE.02 Transition from Calculus to DiffEq: The Forced Oscillator DiffEq

$$y''[t] + b\,y'[t] + c\,y[t] = f[t]$$

Give It a Try!

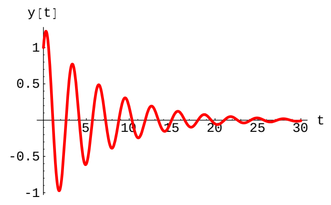
G.1) Going after formulas for some simple unforced and forced oscillators*

□G.1.a) Damped, unforced

Here's a plot of the solution unforced damped oscillator

$$y''[t] + 0.3\,y'[t] + 4.2\,y[t] = 0 \text{ with } y[0] = 1 \text{ and } y'[0] = 1.5:$$

```
b = 0.3;
c = 4.2;
Clear[y, ndsy, t, f]
diffeq = y''[t] + b y'[t] + c y[t] == 0;
endtime = 30;
ndsol =
 NDSolve[{diffeq, y[0] == 1, y'[0] == 1.5}, y[t], {t, 0, endtime}];
ndsy[t_] = y[t] /. ndsol[[1]];
ndsplot = Plot[ndsy[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, AspectRatio → 1/GoldenRatio];
diffeq
```



```
4.2 y[t] + 0.3 y'[t] + y''[t] == 0
```

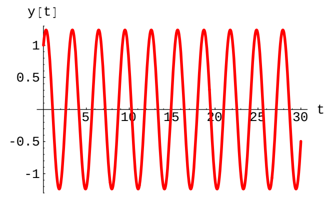
Use the charcteristic equation to come up with a formula for this unforced damped oscillator.

□G.1.b) Undamped,unforced

Here's a plot of the solution unforced undamped oscillator

$$y''[t] + 4.2\,y[t] = 0 \text{ with } y[0] = 1 \text{ and } y'[0] = 1.5:$$

```
b = 0;
c = 4.2;
Clear[y, ndsy, t, f]
diffeq = y''[t] + b y'[t] + c y[t] == 0;
endtime = 30;
ndsol =
 NDSolve[{diffeq, y[0] == 1, y'[0] == 1.5}, y[t], {t, 0, endtime}];
ndsy[t_] = y[t] /. ndsol[[1]];
ndsplot = Plot[ndsy[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, AspectRatio → 1/GoldenRatio];
diffeq
```



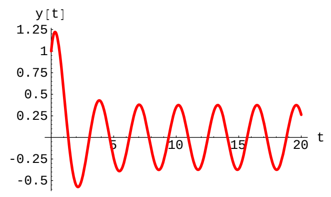
```
4.2 y[t] + y''[t] == 0
```

Use the charcteristic equation to come up with a formula for this unforced undamped oscillator.

□G.1.c) Forced, damped

Here's a forced damped oscillator plotted via **NDSolve**:

```
b = 1.6;
c = 4.2;
Clear[y, ndsy, t, f]
f[t_] = 1.2 Cos[2 t];
diffeq = y''[t] + b y'[t] + c y[t] == f[t];
endtime = 20;
ndsol =
 NDSolve[{diffeq, y[0] == 1, y'[0] == 1.5}, y[t], {t, 0, endtime}];
ndsy[t_] = y[t] /. ndsol[[1]];
ndsplot = Plot[ndsy[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, AspectRatio → 1/GoldenRatio];
diffeq
```



```
4.2 y[t] + 1.6 y'[t] + y''[t] == 1.2 Cos[2 t]
```

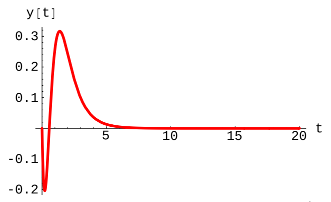
Use the convolution integral method to come up with a formula for this forced oscillator.

Use your formula to say what the global behavior (steady state) of this oscillator is.

□G.1.d)

Here's another forced damped oscillator plotted courtesy of **NDSolve**:

```
b = 4.7;
c = 6.8;
Clear[y, ndsy, t, f]
f[t_] = 6 E^-t;
diffeq = y″[t] + b y′[t] + c y[t] == f[t];
endtime = 20;
ndsol =
 NDSolve[{diffeq, y[0] == 0, y′[0] == -2.3}, y[t], {t, 0, endtime}];
ndsy[t_] = y[t] /. ndsol[[1]];
ndsplot = Plot[ndsy[t], {t, 0, endtime},
   PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
   AxesLabel → {"t", "y[t]"}, AspectRatio → 1/GoldenRatio];
diffeq
```



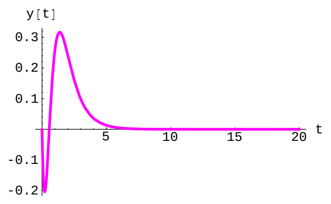
```
6.8 y[t] + 4.7 y′[t] + y″[t] == 6 E^-t
```

Use the convolution integral method to come up with a formula for this forced oscillator.
Use your formula to confirm that the global behavior (steady state) of this oscillator is 0.

□G.1.e) Impulse-forced

Calculus Cal used **NDSolve** to plot another forced damped oscillator. Here's what he did:

```
b = 4.7;
c = 6.8;
Clear[y, t, f]
f[t_] = 6 E^-t - 7 DiracDelta[t - 5] + 8 DiracDelta[t - 10];
diffeq = y″[t] + b y′[t] + c y[t] == f[t];
endtime = 20;
ndsol =
 NDSolve[{diffeq, y[0] == 0, y′[0] == -2.3}, y[t], {t, 0, endtime}];
ndsy[t_] = y[t] /. ndsol[[1]];
ndsplot = Plot[ndsy[t], {t, 0, endtime},
   PlotStyle → {{Thickness[0.01], Magenta}}, PlotRange → All,
   AxesLabel → {"t", "y[t]"}, AspectRatio → 1/GoldenRatio];
diffeq
```



```
6.8 y[t] + 4.7 y′[t] + y″[t] ==
 6 E^-t + 8 DiracDelta[-10 + t] - 7 DiracDelta[-5 + t]
```

Inspect the formula for the forcing function:

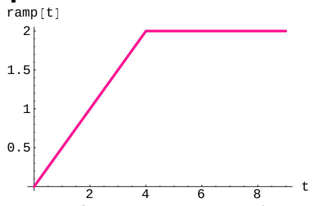
```
f[t]
6 E^-t + 8 DiracDelta[-10 + t] - 7 DiracDelta[-5 + t]
```

Tell Cal why you know his plot is wrong. And then tell Cal where to go.
Use the convolution integral method to come up with a formula for this forced oscillator and use it to give an accurate plot.
What happens to y′[t] as t passes through t = 5?
What happens to y′[t] as t passes through t = 10?

□G.1.f) Ramp-forced

Here is a ramp forcing function and its plot:

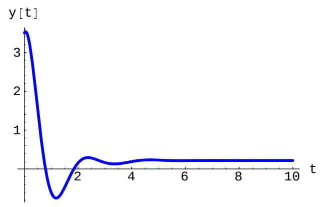
```
Clear[ramp, g, t]
g[t_] = 0.5 t;
changeover = 4.0;
ramp[t_] =
 g[t] UnitStep[changeover - t] + g[changeover] UnitStep[t - changeover];
Plot[ramp[t], {t, 0, 9},
 PlotStyle → {{Thickness[0.01], DeepPink}}, PlotRange → All,
 AxesLabel → {"t", "ramp[t]"}, AspectRatio → 1/GoldenRatio];
ramp[t]
```



```
0.5 t UnitStep[4. - t] + 2. UnitStep[-4. + t]
```

Ramp[t] runs
→ with g[t] = 0.5 t for t < 4.0
→ with g[4] = 2 for t > 4.0.
Here is the plot of a damped oscillator forced by ramp[t]:

```
Clear[y, t, f]
f[t_] = ramp[t];
b = 2.6;
c = 9.2;
ystarter = 3.5;
yprimestarter = 1.5;

diffeq = (y''[t] + b y'[t] + c y[t] == f[t]);

endtime = 10;

ndsol =
 NDSolve[{diffeq, y[0] == ystarter, y'[0] == yprimestarter}, y[t],
 {t, 0, endtime}];
ndsy[t_] = y[t] /. ndsol[[1]];

ndsplot = Plot[ndsy[t], {t, 0, endtime},
 PlotStyle -> {{Thickness[0.01], Blue}}, PlotRange -> All,
  AspectRatio -> 1 / GoldenRatio, AxesLabel -> {"t", "y[t]"}];
 diffeq
```



```
9.2 y[t] + 2.6 y′[t] + y″[t] == 0.5 t UnitStep[4. - t] + 2. UnitStep[-4. + t]
```

Use the convolution integral method to produce an exact formula for this ramp-forced oscillator.

G.2) Steady state and transients for forced damped oscillators:

All solutions eventually settle into the same steady state.

Visualizing the effect of the forcing functions on forced damped oscillators*

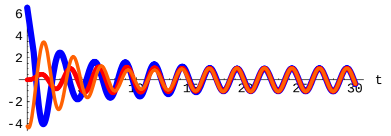
□G.2.a.i) Damped

Here are three plots of solutions a random damped oscillator
$y''[t] + b\,y'[t] + c\,y[t] = 2\,\mathrm{Sin}[2.5\,t]$ ($b > 0$ and $c > 0$).
Two of the solutions have random starting values on y[0] and y′[0].
The other solution is yzeroinput[t] which has starting values y[0] = 0 and y′[0] = 1:

```
Don't worry about this code.
You will learn more about it soon.
```

```
b = Random[Real, {0.5, 1.0}];
c = Random[Real, {4, 6}];
f[t_] = 2 Sin[2.5 t];
forcedoscillator = y″[t] + b y′[t] + c y[t] == f[t];
endtime = 30;
Clear[y, y1, y2, y3, t]
initialdisplacement1 = Random[Real, {4, 10}];
initialvel1 = Random[Real, {-6, 6}];
initialdisplacement2 = 0;
initialvel2 = 0;
```

```
initialdisplacement3 = Random[Real, {-10, -4}];
initialvel3 = Random[Real, {-6, 6}];
sol1 = NDSolve[{forcedoscillator, y[0] == initialdisplacement1,
    y′[0] == initialvel1}, y[t], {t, 0, endtime}];
sol2 = NDSolve[{forcedoscillator, y[0] == initialdisplacement2,
    y′[0] == initialvel2}, y[t], {t, 0, endtime}];
sol3 = NDSolve[{forcedoscillator, y[0] == initialdisplacement3,
    y′[0] == initialvel3}, y[t], {t, 0, endtime}];
y1[t_] = y[t] /. sol1⟦1⟧;
y2[t_] = y[t] /. sol2⟦1⟧;
y3[t_] = y[t] /. sol3⟦1⟧;
plots = Plot[{y1[t], y2[t], y3[t]},
   {t, 0, endtime}, PlotStyle → {{Thickness[0.018], Blue},
     {Thickness[0.014], Red}, {Thickness[0.01], CadmiumOrange}},
   PlotRange → All, AspectRatio → Automatic, AxesLabel → {"t", ""}];
forcedoscillator
```



```
4.87428 y[t] + 0.52448 y′[t] + y″[t] == 2 Sin[2.5 t]
```

Rerun several times.

The three solutions begin their trip in totally different ways, but when t is large, you need a scorecard to tell them apart. In other words, they all settle into the same steady state behavior - regardless of the starting values on y[0] and y′[0].
Remembering that any solution y[t] is of the form

y[t] = yzeroinput[t] + yunforced[t],

explain why this happens.

◻G.2.a.ii)

You are given a forced damped oscillator coming from

y″[t] + b y′[t] + c y[t] = f[t]

with given starter data y[0] = p and y′[0] = q (and with b > 0 and c > 0),
Does the steady state (long term, global scale) behavior of this oscillator depend in any way on the specific values of the starting data?
How do you know for sure?

◻G.2.a.iii) Transient and steady state

You are given a forced damped oscillator coming from

y″[t] + b y′[t] + c y[t] = f[t],

with given starter data y[0] = p and y′[0] = q (and with b > 0 and c > 0),
When you go with the convolution integral method, you get the solution y[t] in terms of

y[t] = yunforced[t] + yzeroinput[t]

where yunforced[t] solves

y″[t] + b y′[t] + c y[t] = 0 with given starter data y[0] = p and y′[0] = q

and where yzeroinput[t] solves

y″[t] + b y′[t] + c y[t] = f[t], with y[0] = 0 and y′[0] = 0.

Why do you think lots of folks like to call yunforced[t] by the name "transient" and yzeroinput[t] by the name "steady state?"

◻G.2.b.i) Undamped

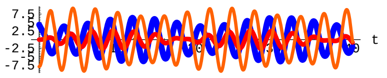
Here are three plots of solutions a random undamped oscillator

y″[t] + c y[t] = 2 Sin[2.5 t] (b > 0 and c > 0).

Two of them have random starting values on y[0] and y′[0]. The other is yzeroinput[t] which has starting values y[0] = 0 and y′[0] = 1:

Don't worry about this code.
You will learn more about it soon.

```
b = 0;
c = Random[Real, {4, 6}];
f[t_] = 2 Sin[2.5 t];
forcedoscillator = y″[t] + b y′[t] + c y[t] == f[t];
endtime = 40;
Clear[y, y1, y2, y3, t]
initialdisplacement1 = Random[Real, {4, 10}];
initialvel1 = Random[Real, {-6, 6}];
initialdisplacement2 = 0;
initialvel2 = 0;
initialdisplacement3 = Random[Real, {-10, -4}];
initialvel3 = Random[Real, {-6, 6}];
sol1 = NDSolve[{forcedoscillator, y[0] == initialdisplacement1,
    y′[0] == initialvel1}, y[t], {t, 0, endtime}];
sol2 = NDSolve[{forcedoscillator, y[0] == initialdisplacement2,
    y′[0] == initialvel2}, y[t], {t, 0, endtime}];
sol3 = NDSolve[{forcedoscillator, y[0] == initialdisplacement3,
    y′[0] == initialvel3}, y[t], {t, 0, endtime}];
y1[t_] = y[t] /. sol1⟦1⟧;
y2[t_] = y[t] /. sol2⟦1⟧;
y3[t_] = y[t] /. sol3⟦1⟧;
plots = Plot[{y1[t], y2[t], y3[t]},
   {t, 0, endtime}, PlotStyle → {{Thickness[0.018], Blue},
     {Thickness[0.014], Red}, {Thickness[0.01], CadmiumOrange}},
   PlotRange → All, AspectRatio → 1 / 5, AxesLabel → {"t", ""}];
forcedoscillator
```



```
4.66473 y[t] + y″[t] == 2 Sin[2.5 t]
```

Rerun several times.

This time the solutions do not settle into the same steady state.
Remembering that any solution y[t] is of the form

y[t] = yzeroinput[t] + yunforced[t],

explain why this happens.

◻G.2.c.i)

Here are three plots of solutions a random damped forced oscillator

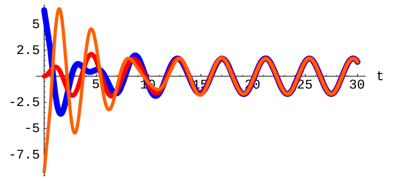
y″[t] + b y′[t] + c y[t] = 4.0 Cos[1.5 t] , (b > 0 ancd c > 0)

Two of the solutions have random starting values on y[0] and y′[0].
The other solution is yzeroinput[t] which has starting values y[0] = 0 and y′[0] = 1:

Don't worry about this code.
You will learn more about it soon.

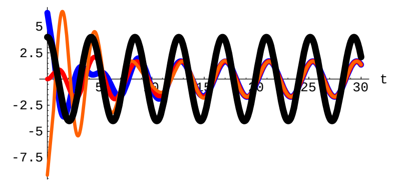
```
b = Random[Real, {0.5, 1.0}];
c = Random[Real, {4, 6}];
f[t_] = 4 Cos[1.5 t];
forcedoscillator = y″[t] + b y′[t] + c y[t] == f[t]
endtime = 30;
Clear[y, y1, y2, y3, t]
initialdisplacement1 = Random[Real, {4, 10}];
initialvel1 = Random[Real, {-6, 6}];
initialdisplacement2 = 0;
initialvel2 = 0;
initialdisplacement3 = Random[Real, {-10, -4}];
initialvel3 = Random[Real, {-6, 6}];
sol1 = NDSolve[{forcedoscillator, y[0] == initialdisplacement1,
    y′[0] == initialvel1}, y[t], {t, 0, endtime}];
sol2 = NDSolve[{forcedoscillator, y[0] == initialdisplacement2,
    y′[0] == initialvel2}, y[t], {t, 0, endtime}];
sol3 = NDSolve[{forcedoscillator, y[0] == initialdisplacement3,
    y′[0] == initialvel3}, y[t], {t, 0, endtime}];
y1[t_] = y[t] /. sol1⟦1⟧;
y2[t_] = y[t] /. sol2⟦1⟧;
y3[t_] = y[t] /. sol3⟦1⟧;
solplots = Plot[{y1[t], y2[t], y3[t]},
   {t, 0, endtime}, PlotStyle → {{Thickness[0.018], Blue},
     {Thickness[0.014], Red}, {Thickness[0.01], CadmiumOrange}},
   PlotRange → All, AspectRatio → Automatic,
   AxesLabel → {"t", ""}];
```

```
4.43454 y[t] + 0.589273 y′[t] + y″[t] == 4 Cos[1.5 t]
```



Throw in this plot of the forcing function f[t]:

```
fplot = Plot[f[t], {t, 0, endtime}, PlotRange → All,
   PlotStyle → {{Thickness[0.02], Black}}, DisplayFunction → Identity];
Show[solplots, fplot, DisplayFunction → $DisplayFunction];
```



Using only your eyes, describe in general terms how the solution plots are related to the plot of the forcing function f[t].

□G.2.c.ii)

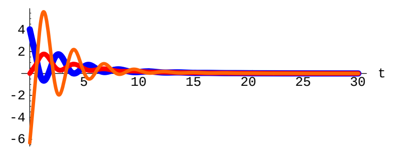
Here are three plots of solutions a random damped forced oscillator

$$y''[t] + b\,y'[t] + c\,y[t] = 6.8\ E^{-0.2\,t}.\ (b > 0 \text{ and } c > 0)$$

Two of the solutions have random starting values on y[0] and y′[0]. The other solution is yzeroinput[t] which has starting values y[0] = 0 and y′[0] = 1:

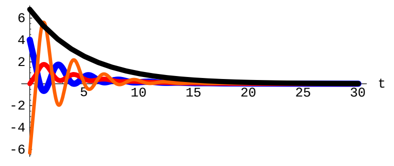
```
                  Don't worry about this code.
                You will learn more about it soon.
b = Random[Real, {0.5, 1.0}];
c = Random[Real, {4, 6}];
Clear[f, t]
f[t_] = 6.8 E^-0.2 t;
forcedoscillator = y″[t] + b y′[t] + c y[t] == f[t]
endtime = 30;
Clear[y, y1, y2, y3, t]
initialdisplacement1 = Random[Real, {4, 10}];
initialvel1 = Random[Real, {-6, 6}];
initialdisplacement2 = 0;
initialvel2 = 0;
initialdisplacement3 = Random[Real, {-10, -4}];
initialvel3 = Random[Real, {-6, 6}];
sol1 = NDSolve[{forcedoscillator, y[0] == initialdisplacement1,
    y′[0] == initialvel1}, y[t], {t, 0, endtime}];
sol2 = NDSolve[{forcedoscillator, y[0] == initialdisplacement2,
    y′[0] == initialvel2}, y[t], {t, 0, endtime}];
sol3 = NDSolve[{forcedoscillator, y[0] == initialdisplacement3,
    y′[0] == initialvel3}, y[t], {t, 0, endtime}];
y1[t_] = y[t] /. sol1[[1]];
y2[t_] = y[t] /. sol2[[1]];
y3[t_] = y[t] /. sol3[[1]];
solplots = Plot[{y1[t], y2[t], y3[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.018], Blue},
    {Thickness[0.014], Red}, {Thickness[0.01], CadmiumOrange}},
  PlotRange → All, AspectRatio → Automatic,
  AxesLabel → {"t", ""}];
```

```
5.33904 y[t] + 0.767432 y′[t] + y″[t] == 6.8 E^-0.2 t
```



Throw in this plot of the forcing function f[t]:

```
fplot = Plot[f[t], {t, 0, endtime}, PlotRange → All, PlotStyle →
    {{Thickness[0.015], Black}}, DisplayFunction → Identity];
Show[solplots, fplot, DisplayFunction → $DisplayFunction];
```



Using only your eyes, describe in general terms how the solution plots are related to the plot of the forcing function f[t].

□G.2.c.iii)

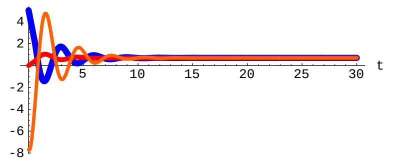
Here are three plots of solutions a random damped forced oscillator

$$y''[t] + b\,y'[t] + c\,y[t] = 3.2\quad (b > 0 \text{ and } c > 0)$$

Two of the solutions have random starting values on y[0] and y′[0]. The other solution is yzeroinput[t] which has starting values y[0] = 0 and y′[0] = 1:

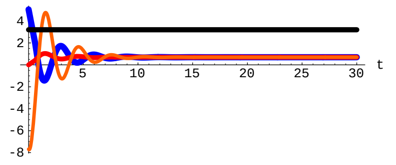
```
                  Don't worry about this code.
                You will learn more about it soon.
b = Random[Real, {0.5, 1.0}];
c = Random[Real, {4, 6}];
Clear[f, t]
f[t_] = 3.2; forcedoscillator = y″[t] + b y′[t] + c y[t] == f[t]
endtime = 30;
Clear[y, y1, y2, y3, t]
initialdisplacement1 = Random[Real, {4, 10}];
initialvel1 = Random[Real, {-6, 6}];
initialdisplacement2 = 0;
initialvel2 = 0;
initialdisplacement3 = Random[Real, {-10, -4}];
initialvel3 = Random[Real, {-6, 6}];
sol1 = NDSolve[{forcedoscillator, y[0] == initialdisplacement1,
    y′[0] == initialvel1}, y[t], {t, 0, endtime}];
sol2 = NDSolve[{forcedoscillator, y[0] == initialdisplacement2,
    y′[0] == initialvel2}, y[t], {t, 0, endtime}];
sol3 = NDSolve[{forcedoscillator, y[0] == initialdisplacement3,
    y′[0] == initialvel3}, y[t], {t, 0, endtime}];
y1[t_] = y[t] /. sol1[[1]];
y2[t_] = y[t] /. sol2[[1]];
y3[t_] = y[t] /. sol3[[1]];
solplots = Plot[{y1[t], y2[t], y3[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.018], Blue},
    {Thickness[0.014], Red}, {Thickness[0.01], CadmiumOrange}},
  PlotRange → All, AspectRatio → Automatic,
  AxesLabel → {"t", ""}];
```

```
4.58932 y[t] + 0.966691 y′[t] + y″[t] == 3.2
```



Throw in this plot of the forcing function f[t]:

```
fplot = Plot[f[t], {t, 0, endtime}, PlotRange -> All, PlotStyle ->
    {{Thickness[0.015], Black}}, DisplayFunction -> Identity];
Show[solplots, fplot, DisplayFunction -> $DisplayFunction];
```



Using only your eyes, describe in general terms how the solution plots are related to the plot of the forcing function f[t].

□G.2.c.iv)

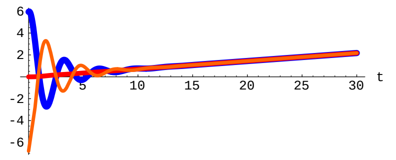
Here are three plots of solutions a random damped forced oscillator

$$y''[t] + b\,y'[t] + c\,y[t] = 0.3\,t,\ (b > 0 \text{ and } c > 0)$$

Two of the solutions have random starting values on y[0] and y'[0]. The other solution is yzeroinput[t] which has starting values y[0] = 0 and y′[0] = 1:

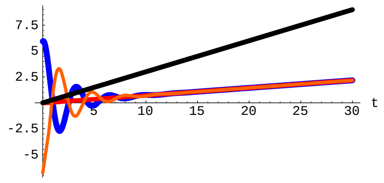
```
                  Don't worry about this code.
                You will learn more about it soon.
b = Random[Real, {0.5, 1.0}];
c = Random[Real, {4, 6}];
Clear[f, t]
f[t_] = 0.3 t; forcedoscillator = y″[t] + b y′[t] + c y[t] == f[t]
endtime = 30;
Clear[y, y1, y2, y3, t]
initialdisplacement1 = Random[Real, {4, 10}];
initialvel1 = Random[Real, {-6, 6}];
initialdisplacement2 = 0;
initialvel2 = 0;
initialdisplacement3 = Random[Real, {-10, -4}];
initialvel3 = Random[Real, {-6, 6}];
sol1 = NDSolve[{forcedoscillator, y[0] == initialdisplacement1,
    y′[0] == initialvel1}, y[t], {t, 0, endtime}];
sol2 = NDSolve[{forcedoscillator, y[0] == initialdisplacement2,
    y′[0] == initialvel2}, y[t], {t, 0, endtime}];
sol3 = NDSolve[{forcedoscillator, y[0] == initialdisplacement3,
    y′[0] == initialvel3}, y[t], {t, 0, endtime}];
y1[t_] = y[t] /. sol1[[1]];
y2[t_] = y[t] /. sol2[[1]];
y3[t_] = y[t] /. sol3[[1]];
solplots = Plot[{y1[t], y2[t], y3[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.018], Blue},
    {Thickness[0.014], Red}, {Thickness[0.01], CadmiumOrange}},
  PlotRange → All, AspectRatio → Automatic,
  AxesLabel → {"t", ""}];
```

```
4.10648 y[t] + 0.942412 y′[t] + y″[t] == 0.3 t
```



Throw in this plot of the forcing function f[t]:

```
fplot = Plot[f[t], {t, 0, endtime}, PlotRange -> All, PlotStyle →
    {{Thickness[0.015], Black}}, DisplayFunction -> Identity];
Show[solplots, fplot, DisplayFunction -> $DisplayFunction];
```



Using only your eyes, describe in general terms how the solution plots are related to the plot of the forcing function f[t].

□G.2.c.v)

Here are three plots of solutions a random damped forced oscillator

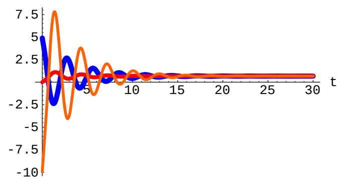
$$y''[t] + b\,y'[t] + c\,y[t] = 3.2 \quad (b > 0 \text{ and } c > 0)$$

Two of the solutions have random starting values on y[0] and y′[0].
The other solution is yzeroinput[t] which has starting values y[0] = 0 and y′[0] = 1:

```
                    Don't worry about this code.
                 You will learn more about it soon.
```

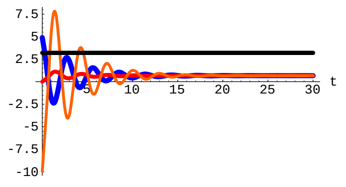
```
b = Random[Real, {0.5, 1.0}];
c = Random[Real, {4, 6}];
Clear[f, t]
f[t_] = 3.2;
forcedoscillator = y''[t] + b y'[t] + c y[t] == f[t]
endtime = 30;
Clear[y, y1, y2, y3, t]
initialdisplacement1 = Random[Real, {4, 10}];
initialvel1 = Random[Real, {-6, 6}];
initialdisplacement2 = 0;
initialvel2 = 0;
initialdisplacement3 = Random[Real, {-10, -4}];
initialvel3 = Random[Real, {-6, 6}];
sol1 = NDSolve[{forcedoscillator, y[0] == initialdisplacement1,
    y'[0] == initialvel1}, y[t], {t, 0, endtime}];
sol2 = NDSolve[{forcedoscillator, y[0] == initialdisplacement2,
    y'[0] == initialvel2}, y[t], {t, 0, endtime}];
sol3 = NDSolve[{forcedoscillator, y[0] == initialdisplacement3,
    y'[0] == initialvel3}, y[t], {t, 0, endtime}];
y1[t_] = y[t] /. sol1[[1]];
y2[t_] = y[t] /. sol2[[1]];
y3[t_] = y[t] /. sol3[[1]];
solplots = Plot[{y1[t], y2[t], y3[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.018], Blue},
    {Thickness[0.014], Red}, {Thickness[0.01], CadmiumOrange}},
  PlotRange → All, AspectRatio → Automatic,
  AxesLabel → {"t", ""}];
```

```
4.77849 y[t] + 0.572247 y'[t] + y''[t] == 3.2
```



Throw in this plot of the forcing function f[t]:

```
fplot = Plot[f[t], {t, 0, endtime}, PlotRange -> All, PlotStyle →
    {{Thickness[0.015], Black}}, DisplayFunction -> Identity];
Show[solplots, fplot, DisplayFunction -> $DisplayFunction];
```



Using only your eyes, describe in general terms how the solution plots are related to the plot of the forcing function f[t].

□G.2.c.vi)

Here are three plots of solutions a random damped forced oscillator

$$y''[t] + b\,y'[t] + c\,y[t] = 10\,E^{-0.3\,(t-6)^2}, \ (b > 0 \text{ and } c > 0)$$

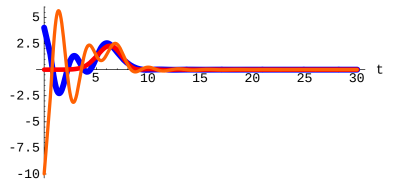
Two of the solutions have random starting values on y[0] and y′[0].
The other solution is yzeroinput[t] which has starting values y[0] = 0 and y′[0] = 1:

```
                    Don't worry about this code.
                 You will learn more about it soon.
```

```
b = Random[Real, {0.5, 1.0}];
c = Random[Real, {4, 6}];
Clear[f, t]
f[t_] = 10 E^(-0.3 (t-6)^2)
forcedoscillator = y''[t] + b y'[t] + c y[t] == f[t]
endtime = 30;
Clear[y, y1, y2, y3, t]
initialdisplacement1 = Random[Real, {4, 10}];
initialvel1 = Random[Real, {-6, 6}];
initialdisplacement2 = 0;
initialvel2 = 0;
initialdisplacement3 = Random[Real, {-10, -4}];
initialvel3 = Random[Real, {-6, 6}];
sol1 = NDSolve[{forcedoscillator, y[0] == initialdisplacement1,
    y'[0] == initialvel1}, y[t], {t, 0, endtime}];
sol2 = NDSolve[{forcedoscillator, y[0] == initialdisplacement2,
    y'[0] == initialvel2}, y[t], {t, 0, endtime}];
sol3 = NDSolve[{forcedoscillator, y[0] == initialdisplacement3,
    y'[0] == initialvel3}, y[t], {t, 0, endtime}];
y1[t_] = y[t] /. sol1[[1]];
y2[t_] = y[t] /. sol2[[1]];
y3[t_] = y[t] /. sol3[[1]];
solplots = Plot[{y1[t], y2[t], y3[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.018], Blue},
    {Thickness[0.014], Red}, {Thickness[0.01], CadmiumOrange}},
  PlotRange → All, AspectRatio → Automatic,
  AxesLabel → {"t", ""}];
```

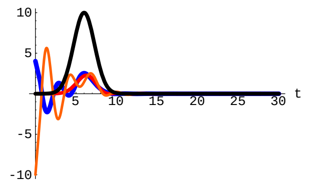
$$10\,E^{-0.3\,(-6+t)^2}$$

$$5.01161\,y[t] + 0.806968\,y'[t] + y''[t] == 10\,E^{-0.3\,(-6+t)^2}$$



Throw in this plot of the forcing function f[t]:

```
fplot = Plot[f[t], {t, 0, endtime}, PlotRange -> All, PlotStyle →
    {{Thickness[0.015], Black}}, DisplayFunction -> Identity];
Show[solplots, fplot, DisplayFunction -> $DisplayFunction];
```

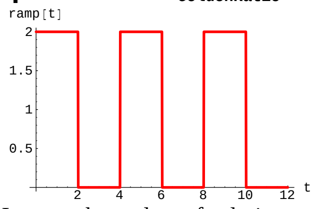


Using only your eyes, describe in general terms how the solution plots are related to the plot of the forcing function f[t].

□G.2.c.vii)

Here is a square wave forcing function f[t]:

```
Clear[f, t]
f[t_] = Sign[Sin[0.5 Pi t]] + 1;

Plot[f[t], {t, 0, 12},
  PlotStyle -> {{Thickness[0.01], Red}},
   PlotRange -> All, AxesLabel -> {"t", "ramp[t]"},
  AspectRatio -> 1/GoldenRatio];
```



Here are three plots of solutions a random damped forced oscillator

$$y''[t] + b\,y'[t] + c\,y[t] = f[t]\,, \ (b > 0 \text{ and } c > 0)$$

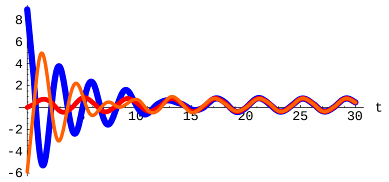
Two of the solutions have random starting values on y[0] and y′[0].
The other solution is yzeroinput[t] which has starting values y[0] = 0 and y′[0] = 1:

```
                    Don't worry about this code.
                 You will learn more about it soon.
```

```
b = Random[Real, {0.5, 1.0}];
c = Random[Real, {4, 6}];
```

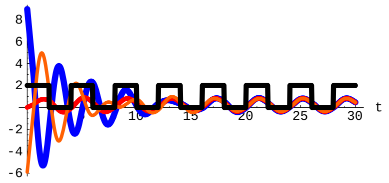
```
forcedoscillator = y''[t] + b y'[t] + c y[t] == f[t]
endtime = 30;
Clear[y, y1, y2, y3, t]
initialdisplacement1 = Random[Real, {4, 10}];
initialvel1 = Random[Real, {-6, 6}];
initialdisplacement2 = 0;
initialvel2 = 0;
initialdisplacement3 = Random[Real, {-10, -4}];
initialvel3 = Random[Real, {-6, 6}];
sol1 = NDSolve[{forcedoscillator, y[0] == initialdisplacement1,
    y'[0] == initialvel1}, y[t], {t, 0, endtime}];
sol2 = NDSolve[{forcedoscillator, y[0] == initialdisplacement2,
    y'[0] == initialvel2}, y[t], {t, 0, endtime}];
sol3 = NDSolve[{forcedoscillator, y[0] == initialdisplacement3,
    y'[0] == initialvel3}, y[t], {t, 0, endtime}];
y1[t_] = y[t] /. sol1[[1]];
y2[t_] = y[t] /. sol2[[1]];
y3[t_] = y[t] /. sol3[[1]];
solplots = Plot[{y1[t], y2[t], y3[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.018], Blue},
    {Thickness[0.014], Red}, {Thickness[0.01], CadmiumOrange}},
  PlotRange → All, AspectRatio → Automatic,
  AxesLabel → {"t", ""}];
```

```
4.45602 y[t] + 0.535201 y'[t] + y''[t] == 1 + Sign[Sin[1.5708 t]]
```



Throw in this plot of the forcing function f[t]:

```
fplot = Plot[f[t], {t, 0, endtime}, PlotRange → All, PlotStyle →
    {{Thickness[0.015], Black}}, DisplayFunction → Identity];
Show[solplots, fplot, DisplayFunction → $DisplayFunction];
```



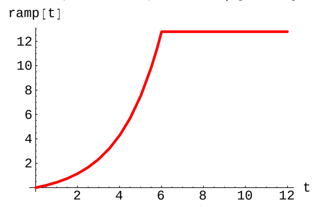
Using only your eyes, describe in general terms how the solution plots are related to the plot of the forcing function f[t].

□G.2.c.viii)

Here is a ramp forcing function f[t]:

```
Clear[f, g, t]
g[t_] = 0.67 (E^(0.5 t) - 1);
changeover = 6.0;
f[t_] =
  g[t] UnitStep[changeover - t] + g[changeover] UnitStep[t - changeover]
Plot[f[t], {t, 0, 12}, PlotStyle → {{Thickness[0.01], Red}},
  PlotRange → All, AxesLabel → {"t", "ramp[t]"},
  AspectRatio → 1/GoldenRatio];
```

```
0.67 (-1 + E^(0.5 t)) UnitStep[6. - t] + 12.7873 UnitStep[-6. + t]
```



Here are three plots of solutions a random damped forced oscillator

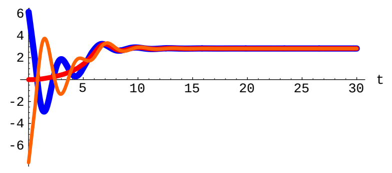
$y''[t] + b\,y'[t] + c\,y[t] = f[t]$, $(b > 0$ and $c > 0)$

Two of the solutions have random starting values on y[0] and y′[0]. The other solution is yzeroinput[t] which has starting values y[0] = 0 and y′[0] = 1:

```
Don't worry about this code.
You will learn more about it soon.
```

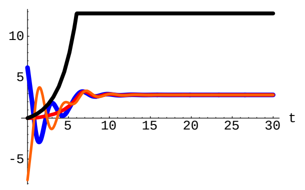
```
b = Random[Real, {0.5, 1.0}];
c = Random[Real, {4, 6}];
forcedoscillator = y''[t] + b y'[t] + c y[t] == f[t]
endtime = 30;
Clear[y, y1, y2, y3, t]
initialdisplacement1 = Random[Real, {4, 10}];
initialvel1 = Random[Real, {-6, 6}];
initialdisplacement2 = 0;
initialvel2 = 0;
initialdisplacement3 = Random[Real, {-10, -4}];
initialvel3 = Random[Real, {-6, 6}];
sol1 = NDSolve[{forcedoscillator, y[0] == initialdisplacement1,
    y'[0] == initialvel1}, y[t], {t, 0, endtime}];
sol2 = NDSolve[{forcedoscillator, y[0] == initialdisplacement2,
    y'[0] == initialvel2}, y[t], {t, 0, endtime}];
sol3 = NDSolve[{forcedoscillator, y[0] == initialdisplacement3,
    y'[0] == initialvel3}, y[t], {t, 0, endtime}];
y1[t_] = y[t] /. sol1[[1]];
y2[t_] = y[t] /. sol2[[1]];
y3[t_] = y[t] /. sol3[[1]];
solplots = Plot[{y1[t], y2[t], y3[t]},
  {t, 0, endtime}, PlotStyle → {{Thickness[0.018], Blue},
    {Thickness[0.014], Red}, {Thickness[0.01], CadmiumOrange}},
  PlotRange → All, AspectRatio → Automatic,
  AxesLabel → {"t", ""}];
```

```
4.50861 y[t] + 0.970907 y'[t] + y''[t] ==
 0.67 (-1 + E^(0.5 t)) UnitStep[6. - t] + 12.7873 UnitStep[-6. + t]
```



Throw in this plot of the forcing function f[t]:

```
fplot = Plot[f[t], {t, 0, endtime}, PlotRange → All, PlotStyle →
    {{Thickness[0.015], Black}}, DisplayFunction → Identity];
Show[solplots, fplot, DisplayFunction → $DisplayFunction];
```



Using only your eyes, describe in general terms how the solution plots are related to the plot of the forcing function f[t].

G.3) Amplitude and frequency of unforced oscillators

□G.3.a.i)

You are in the lab and in the process of coming up with a formula for this given undamped unforced oscillator with starter values of y[0] = 5.9 and y′[0] = −0.4:

```
Clear[y, t]
b = 0;
c = 4 π^2;
linoscdiffeq = y''[t] + b y'[t] + c y[t] == 0
startery = 5.9;
starteryprime = -0.4;
```

```
4 π^2 y[t] + y''[t] == 0
```

You copy, paste and edit from the Basics:

```
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
Clear[gensol, C1, C2]
gensol[t_] = C1 E^(zsols[[1,1,2]] t) + C2 E^(zsols[[2,1,2]] t);

ystarteq = gensol[0] == startery;
yprimestarteq = gensol'[0] == starteryprime;
Csols = Solve[{ystarteq, yprimestarteq}, {C1, C2}];
Clear[yformula]
yformula[t_] = Chop[ComplexExpand[gensol[t] /. Csols[[1]]]]
```

```
5.9 Cos[2 π t] - 0.063662 Sin[2 π t]
```

Your job is to calculate the amplitude and the frequency for this undamped unforced oscillator. Do it and show off your work with convincing plots.

□G.3.a.ii)

Look again at the solutions of the characteristic equation from part i):

```
Clear[z];
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z]
{{z → -2 I π}, {z → 2 I π}}
```

Can you come up with the frequency for the oscillator above directly from the solutions of the characteristic equation without going to all the bother of finding a solution formula?

What does your answer tell you about the frequency of any solution of:

```
y''[t] + b y'[t] + c y[t] == 0
4 π^2 y[t] + y''[t] == 0
```

□G.3.b.i)

You are in the lab and are in the process of coming up with a formula for this given damped oscillator with these given starter values on y[0] and y′[0]:

```
Clear[y, t]
b = 5.7;
c = 20.1;
linoscdiffeq = y''[t] + b y'[t] + c y[t] == 0
starterY = 5.9;
starteryprime = -0.4;
20.1 y[t] + 5.7 y'[t] + y''[t] == 0
```

You copy, paste and edit from the Basics:

```
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
Clear[gensol, C1, C2]
gensol[t_] = C1 E^(zsols[[1,1,2]] t) + C2 E^(zsols[[2,1,2]] t);
ystarteq = gensol[0] == startery;
yprimestarteq = gensol'[0] == starteryprime;
Csols = Solve[{ystarteq, yprimestarteq}, {C1, C2}];
Clear[yformula]
yformula[t_] = Chop[ComplexExpand[gensol[t] /. Csols[[1]]]]

5.9 E^(-2.85 t) Cos[3.46085 t] + 4.74305 E^(-2.85 t) Sin[3.46085 t]
```

Your job is to calculate the amplitude and the frequency for this damped unforced oscillator. Do it and show off your work with convincing plots.

□G.3.b.ii)

Look again at the solutions of the characteristic equation from part i):

```
Clear[z]
charequation = z^2 + b z + c == 0; zsols = Solve[charequation, z]
{{z → -2.85 - 3.46085 I}, {z → -2.85 + 3.46085 I}}
```

Can you come up with the frequency for the oscillator above directly from the solutions of the characteristic equation without going to all the bother of finding a solution formula?

What does your answer tell you about the frequency of any solution of:

```
y''[t] + b y'[t] + c y[t] == 0
20.1 y[t] + 5.7 y'[t] + y''[t] == 0
```

□G.3.c.i)

Here is a formula for the solution of the undamped unforced oscillator

$$y''[t] + 9.0\, y[t] = 0$$

with random starter data:

```
b = 0;
c = 9.0;
Clear[y, yundamped, t]
linoscdiffeq = y''[t] + b y[t] + c y[t] == 0;
ystarter = Random[Real, {2, 6}];
yprimestarter = Random[Real, {-4, 4}];
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
  C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]};
Csols =
  Solve[{generalsol[0] == ystarter, generalsol'[0] == yprimestarter}];
yundamped[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
2.86672 Cos[3. t] + 1.06545 Sin[3. t]
```

You damp this oscillator by adding a damping term b y′[t]:

```
b = 2.1;
Clear[y, ydamped, t]
linoscdiffeq = y''[t] + b y'[t] + c y[t] == 0
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;

zsols = Solve[charequation, z];
generalsol[t_] =
  C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]};
Csols =
  Solve[{generalsol[0] == ystarter, generalsol'[0] == yprimestarter}];
ydamped[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
9. y[t] + 2.1 y'[t] + y''[t] == 0
2.86672 E^(-1.05 t) Cos[2.81025 t] + 2.20849 E^(-1.05 t) Sin[2.81025 t]
```

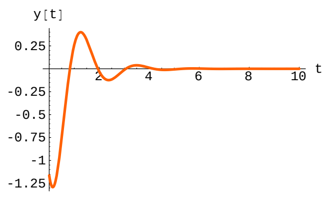
Does the damped version have the same frequency as the the undamped version?

Does the damped version have the same amplitude as the the undamped version?

□G.3.c.ii)

Here's a new unforced lightly damped oscillator diffeq with a plot corresponding to some random starter data:

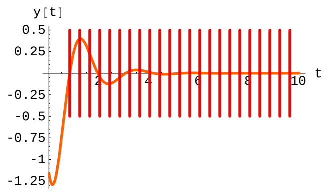
```
Clear[y, ndsy, t];
b = Random[Real, {0.3, 0.7}];
c = Random[Real, {6.0, 12.0}];
oscdiffeq = y''[t] + b y'[t] + c y[t] == 0;
endtime = 10;
Clear[y];
startery = Random[Real, {-2, 2}];
starteryprime = Random[Real, {-2, 2}];
ndsol = NDSolve[{linoscdiffeq, y[0] == startery,
    y'[0] == starteryprime}, y[t], {t, 0, endtime}];
ndsy[t_] = y[t] /. ndsol[[1]];
ndsplot = Plot[ndsy[t], {t, 0, endtime},
   PlotStyle → {{Thickness[0.01], CadmiumOrange}}, PlotRange -> All,
   AxesLabel → {"t", "y[t]"}, AspectRatio → 1/GoldenRatio];
oscdiffeq
```



```
8.98336 y[t] + 0.402957 y'[t] + y''[t] == 0
```

Now look at this:

```
zerospacer = 0.4;
azerosol = FindRoot[ndsy[t] == 0, {t, 0.8}];

azero = t /. azerosol[[1]];
zeromarkers =
  Table[Graphics[{Thickness[0.01], Red, Line[{{t, -0.5}, {t, 0.5}}]}],
   {t, azero, endtime, zerospacer}];
Show[ndsplot, zeromarkers];
```



Notice that the first vertical line hits the t-axis right on a place at which the solution is 0. But the others don't do this.

Reason: The zerospacer at the very top of the code is wrong.

Your job is to look at the solutions of the characteristic equation:

```
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z]
{{z → -0.201479 - 2.99045 I}, {z → -0.201479 + 2.99045 I}}
```

And then to use what you see to reset the zerospacing number so that the vertical lines hit all the places at which the solution is zero. Show off your answer with a plot.

□G.3.d)

Here's *Mathematica* cranking out an exact formula for the solution of the undamped oscillator diffeq

$$y''[t] + 4.7\,y[t] = 0$$

with y[0] = 3.3 and y′[0] = 1.2:

```
Clear[y, t]
b = 0;
c = 4.7;
linoscdiffeq = y''[t] + b y'[t] + c y[t] == 0
startery = 3.3;
starteryprime = 1.2;
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];

Clear[gensol, C1, C2]
gensol[t_] = C1 E^(zsols[[1,1,2]] t) + C2 E^(zsols[[2,1,2]] t);
ystarteq = gensol[0] == startery;
yprimestarteq = gensol'[0] == starteryprime;

Csols = Solve[{ystarteq, yprimestarteq}, {C1, C2}];
Clear[yundampedformula]
yundampedformula[t_] = Chop[ComplexExpand[gensol[t] /. Csols[[1]]]]
```

```
4.7 y[t] + y''[t] == 0
3.3 Cos[2.16795 t] + 0.553519 Sin[2.16795 t]
```

Here is *Mathematica* cranking out an exact formula for the solution of the same oscillator after a small damping term has been inserted:

```
Clear[y, t]
b = 0.2;
c = 4.7;
linoscdiffeq = y''[t] + b y'[t] + c y[t] == 0
startery = 3.3;
starteryprime = 1.2;
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];

Clear[gensol, C1, C2]
gensol[t_] = C1 E^(zsols[[1,1,2]] t) + C2 E^(zsols[[2,1,2]] t);

ystarteq = gensol[0] == startery;
yprimestarteq = gensol'[0] == starteryprime;
Csols = Solve[{ystarteq, yprimestarteq}, {C1, C2}];
Clear[ydampedformula]
ydampedformula[t_] = Chop[ComplexExpand[gensol[t] /. Csols[[1]]]]
```

```
4.7 y[t] + 0.2 y'[t] + y''[t] == 0
3.3 E^(-0.1 t) Cos[2.16564 t] + 0.706488 E^(-0.1 t) Sin[2.16564 t]
```

When you put the damping term on the oscillator, did the frequency go up down or remain the same?

□G.3.e)

Here is *Mathematica* cranking out an exact formula for the solution of the damped oscillator diffeq

$$y''[t] + 0.4\,y'[t] + 5.0\,y[t] = 0$$

with y[0] = 3.3 and y′[0] = 1.2:

```
Clear[y, t]
b = 0.4;
c = 5.0;
linoscdiffeq = y''[t] + b y'[t] + c y[t] == 0
startery = 3.3;
starteryprime = 1.2;
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];

Clear[gensol, C1, C2]
gensol[t_] = C1 E^(zsols[[1,1,2]] t) + C2 E^(zsols[[2,1,2]] t);

ystarteq = gensol[0] == startery;
yprimestarteq = gensol'[0] == starteryprime;
Csols = Solve[{ystarteq, yprimestarteq}, {C1, C2}];
Clear[yformula]
yformula[t_] = Chop[ComplexExpand[gensol[t] /. Csols[[1]]]]
```

```
5. y[t] + 0.4 y'[t] + y''[t] == 0
3.3 E^(-0.2 t) Cos[2.22711 t] + 0.835165 E^(-0.2 t) Sin[2.22711 t]
```

Now raise c from 5 to 5.1 and keep everything else the same:

```
Clear[y, t]
b = 0.4;
c = 5.1;
linoscdiffeq = y''[t] + b y'[t] + c y[t] == 0
startery = 3.3;
starteryprime = 1.2;
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];

Clear[gensol, C1, C2]
gensol[t_] = C1 E^(zsols[[1,1,2]] t) + C2 E^(zsols[[2,1,2]] t);
ystarteq = gensol[0] == startery;
yprimestarteq = gensol'[0] == starteryprime;
Csols = Solve[{ystarteq, yprimestarteq}, {C1, C2}];

Clear[newyformula]
newyformula[t_] = Chop[ComplexExpand[gensol[t] /. Csols[[1]]]]
```

```
5.1 y[t] + 0.4 y'[t] + y''[t] == 0
3.3 E^(-0.2 t) Cos[2.24944 t] + 0.826871 E^(-0.2 t) Sin[2.24944 t]
```

The formula coming from c = 5.0 is:

```
yformula[t]
```

```
3.3 E^(-0.2 t) Cos[2.22711 t] + 0.835165 E^(-0.2 t) Sin[2.22711 t]
```

The formula coming from c = 5.1 is:

```
newyformula[t]
```

```
3.3 E^(-0.2 t) Cos[2.24944 t] + 0.826871 E^(-0.2 t) Sin[2.24944 t]
```

When you raised c, did the frequency go up down or remain the same?

G.4) Resonance*

Anyone wanting to upset a rocking stone will push in tune with oscillations of the stone,
so as always to secure a favourable moment for a push. If the pushes are out of tune,
some increase the oscillations, but others check them.
But when they are all in tune, after a time, all the pushes are favorable.
---Alfred North Whitehead

The phenomenom is called "resonance."

□G.4.a.i)

Did you ever wonder why some mircophone-loud speaker systems squeal?
One way to look at this is to imagine that the sound from a loudspeaker system is an ordinary undamped oscillator. When you turn on the mircophone, you force this undamped oscillator with its own output.
Here's the the formula for the solution of the very ordinary undamped, unforced oscillator:

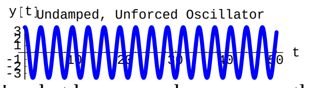
$$y''[t] + 4\,y[t] = 0 \text{ with } y[0] = 3.6 \text{ and } y'[0] = 0.8.$$

```
b = 0; c = 4; ystarter = 3.6;
yprimestarter = 0.8; Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];

generalsol[t_] =
 C1 E^(z1 t) + C2 E^(z2 t) /. {z1 -> zsols[[1, 1, 2]], z2 -> zsols[[2, 1, 2]]};
Csols =
 Solve[{generalsol[0] == ystarter, generalsol'[0] == yprimestarter}];

Clear[yunforced]
yunforced[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
3.6 Cos[2 t] + 0.4 Sin[2 t]
```

See how it plots out:

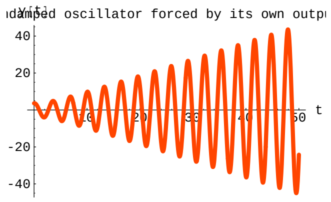
```
endtime = 50;
unforcedplot = Plot[yunforced[t], {t, 0, endtime},
  PlotStyle -> {{Thickness[0.015], Blue}}, AxesLabel -> {"t", "y[t]"},
  PlotLabel -> "Undamped, Unforced Oscillator", AspectRatio -> 1/5];
```



Here's what happens when you use this formula to force this undamped oscillator with its own output:

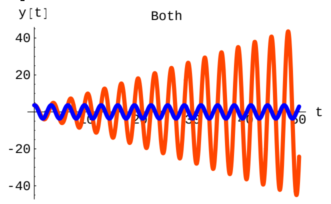
```
Clear[f]
f[t_] = yunforced[t];
Clear[y, t]
solution = NDSolve[{y''[t] + b y'[t] + c y[t] == f[t],
   y[0] == ystarter, y'[0] == yprimestarter}, y[t], {t, 0, endtime}];
```

```
forcedy[t_] = y[t] /. solution[[1]];
forcedplot = Plot[forcedy[t],
  {t, 0, endtime}, PlotStyle → {{Thickness[0.015], OrangeRed}},
  AxesLabel → {"t", "y[t]"}, PlotPoints → 3 endtime,
  PlotLabel → "Undamped oscillator forced by its own output",
  AspectRatio → 1/GoldenRatio];
```



Shake, rattle and roll. The loudspeaker is screaming!
Oscillation with greater and greater swings as time goes on.
This is what folks call "resonance."
See them together:

```
Show[forcedplot, unforcedplot, PlotLabel → "Both"];
```



Here's what happens when you use the convolution integral method to come up with a formula for this forced oscillator:

◻Step 1: Calculate yunforced[t]

```
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
  C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]}
C1 E^(-2 I t) + C2 E^(2 I t)
```

```
Csols =
  Solve[{generalsol[0] == ystarter, generalsol′[0] == yprimestarter}];
Clear[yunforced]
yunforced[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
3.6 Cos[2 t] + 0.4 Sin[2 t]
```

◻Step 2: Calculate the unit impulse response, yunitimpulse[t]

```
Csols = Solve[{generalsol[0] == 0, generalsol′[0] == 1}];
Clear[yunitimpulse]
yunitimpulse[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
1/2 Sin[2 t]
```

◻Step 3: Calculate

$$\mathbf{yzeroinput[t]} = \int_0^t \mathbf{unitimpulse[t-x]\, f[x]}\, d\mathbf{x}$$

```
Clear[yzeroinput, x]
yzeroinput[t_] = Apart[Chop[∫_0^t yunitimpulse[t - x] f[x] dx]]
-0.1 t Cos[2 t] + 0.9 (0.0555556 + 1. t) Sin[2 t]
```

Pay no attention to any error messages.

◻Step 4: Set formulay[t] = yzeroinput[t] + yunforced[t]:

```
Clear[formulay]
formulay[t_] = Simplify[yzeroinput[t] + yunforced[t]]
(3.6 - 0.1 t) Cos[2 t] + (0.45 + 0.9 t) Sin[2 t]
```

This is the formula for this resonating undamped oscillator.

Use this formula to come up with a formula for amplitude[t] for this oscillator and explain why your formula guarantees that this oscillator will go into resonance (i.e. the amplitude will grow to infinity).

◻G.4.a.ii)

Take another look at the formula for this undamped oscillator forced by its own output:

```
formulay[t]
(3.6 - 0.1 t) Cos[2 t] + (0.45 + 0.9 t) Sin[2 t]
```

The formula for the undamped unforced oscillator is:

```
yunforced[t]
3.6 Cos[2 t] + 0.4 Sin[2 t]
```

Is the frequency of the oscillator forced by its own output any different from the frequency of the unforced oscillator?

◻G.4.b.i)

Here's the formula for the solution of a simple unforced undamped oscillator

$$y''[t] + 9\, y[t] = 0$$

with random starting data.

```
Clear[y, t, c]
b = 0;
c = 9;
ystarter = Random[Real, {-1, 1}];
yprimestarter = Random[Real, {-1, 1}];
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
  C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]};
Csols =
  Solve[{generalsol[0] == ystarter, generalsol′[0] == yprimestarter}];
Clear[yunforced]
yunforced[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
0.286914 Cos[3 t] + 0.327614 Sin[3 t]
```

Here's the convolution integral method cranking out the formula for the solution of the same undamped oscillator forced by

$$f[t] = A \operatorname{Sin}[3\, t] + B \operatorname{Cos}[3\, t]:$$

The frequency of f[t] matches that of
the solution of the undamped unforced oscillator.
Here A and B are cleared constants.

◻Step 1: Calculate yunforced[t]

```
Clear[C1, C2, z, z1, z2, f, A, B, generalsol, t]
b = 0;
c = 9;
f[t_] = A Sin[3 t] + B Cos[3 t];

charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
  C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]}
C1 E^(-3 I t) + C2 E^(3 I t)
```

```
Csols = Solve[{generalsol[0] == ystarter,
       generalsol'[0] == yprimestarter}];

Clear[yunforced];
yunforced[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
0.286914 Cos[3 t] + 0.327614 Sin[3 t]
```

◻Step 2: Calculate the unit impulse response, yunitimpulse[t]

```
Csols = Solve[{generalsol[0] == 0, generalsol′[0] == 1}];
Clear[yunitimpulse]
yunitimpulse[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
1/3 Sin[3 t]
```

◻Step 3: Calculate

$$\mathbf{yzeroinput[t]} = \int_0^t \mathbf{unitimpulse[t-x]\, f[x]}\, d\mathbf{x}$$

```
Clear[yzeroinput, x]
yzeroinput[t_] = Apart[Chop[∫_0^t yunitimpulse[t - x] f[x] dx]]
-1/6 A t Cos[3 t] + 1/18 (A + 3 B t) Sin[3 t]
```

Pay no attention to any error messages.

◻Step 4: Set formulay[t] = yzeroinput[t] + yunforced[t]:

```
Clear[formulay]
formulay[t_] = Simplify[yzeroinput[t] + yunforced[t]]
1/18 ((5.16445 - 3 A t) Cos[3 t] + (5.89705 + A + 3 B t) Sin[3 t])
```

How does this formula signal that resonance (i.e. the amplitude will grow to ∞) is inevitable unless you go with

$$A = B = 0?$$

◻G.4.b.ii)

Stay with the same setup

$$y''[t] + 9\, y[t] = 0$$

with random starting data as in part i).
And see what happens when you force that oscillator with

$$f[t] = A \operatorname{Sin}[k\, t] + A \operatorname{Cos}[k\, t]:$$

Unless $k = 3$, the frequency of $f[t]$ does NOT match that of the solution of the undamped unforced oscillator.
Here A, B, and k are cleared constants.

□Step 1: Calculate yunforced[t]

```
Clear[C1, C2, z, z1, z2, f, A, B, generalsol, t, k]
b = 0;
c = 9;
f[t_] = A Sin[k t] + B Cos[k t];
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
 C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]}
```

```
C1 E^(-3 I t) + C2 E^(3 I t)
```

```
Csols = Solve[{generalsol[0] == ystarter,
      generalsol'[0] == yprimestarter}];

Clear[yunforced];
yunforced[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
0.286914 Cos[3 t] + 0.327614 Sin[3 t]
```

□Step 2: Calculate the unit impulse response, yunitimpulse[t]

```
Csols = Solve[{generalsol[0] == 0, generalsol′[0] == 1}];
Clear[yunitimpulse]
yunitimpulse[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
1/3 Sin[3 t]
```

□Step 3: Calculate

$$\text{yzeroinput}[t] = \int_0^t \text{unitimpulse}[t-x]\, f[x]\, dx$$

```
Clear[yzeroinput, x]
yzeroinput[t_] = Apart[Chop[∫_0^t yunitimpulse[t - x] f[x] ⅆx]]
```

```
3 B Cos[3 t] - 3 B Cos[k t] + A k Sin[3 t]       A Sin[k t]
------------------------------------------ - ----------------
         3 (-3 + k) (3 + k)                  (-3 + k) (3 + k)
```

Pay no attention to any error messages.

□Step 4: Set formulay[t] = yzeroinput[t] + yunforced[t]:

```
Clear[formulay]
formulay[t_] = Simplify[yzeroinput[t] + yunforced[t]]
```

```
    1
---------- ((-7.74667 + 3 B + 0.860742 k^2) Cos[3 t] - 3 B Cos[k t] -
3 (-9 + k^2)
   8.84557 Sin[3 t] + A k Sin[3 t] + 0.982841 k^2 Sin[3 t] - 3 A Sin[k t])
```

How does this formula signal that true resonance (amplitude growing to ∞) is impossible unless k = 3 or k = −3 ?
How does this formula signal that something like true resonance happens when you go with

a or b ≠ 0,

and you make k close, but not equal to 3 ?

□G.4.b.iii)

You want to put the undamped oscillator coming from

$$y''[t] + 16\, y[t] = 0$$

with $y[0] = 0$ and $y'[0] = 2$
into resonance with a forcing function

$$f[t] = 0.1 \sin[k\, t].$$

What number k do you go with?
Show off your answer with a decisive formula and plot.

□G.4.c.i)

Here's the convolution integral method spitting out a formula for the forced undamped oscillator coming from

$$y''[t] + 2.51\, y[t] = 0.5 \sin\left[\sqrt{2.51}\, t\right]$$

with $y[0] = 2$ and $y'[0] = -1$:

□Step 1: Calculate yunforced[t]

```
Clear[f, t]
f[t_] = 0.5 Sin[√2.51 t];
b = 0;
c = 2.51;
ystarter = 2;
yprimestarter = -1;
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
 C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]}
```

```
C1 E^((0.-1.5843 I) t) + C2 E^((0.+1.5843 I) t)
```

```
Csols = Solve[{generalsol[0] == ystarter,
      generalsol'[0] == yprimestarter}];

Clear[yunforced];
yunforced[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
2. Cos[1.5843 t] - 0.631194 Sin[1.5843 t]
```

□Step 2: Calculate the unit impulse response, yunitimpulse[t]

```
Csols = Solve[{generalsol[0] == 0, generalsol′[0] == 1}];
Clear[yunitimpulse]
yunitimpulse[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
0.631194 Sin[1.5843 t]
```

□Step 3: Calculate

$$\text{yzeroinput}[t] = \int_0^t \text{yunitimpulse}[t-x]\, f[x]\, dx$$

```
Clear[yzeroinput, x]
yzeroinput[t_] = Apart[Chop[∫_0^t yunitimpulse[t - x] f[x] ⅆx]]
```

```
-0.157799 t Cos[1.5843 t] + 0.0996016 Sin[1.5843 t]
```

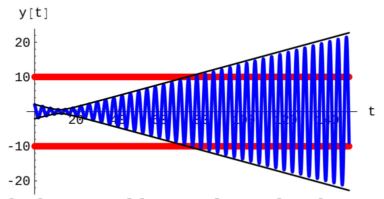
□Step 4: Set formulay[t] = yzeroinput[t] + yunforced[t]:

```
Clear[formulay]
formulay[t_] = Expand[yzeroinput[t] + yunforced[t]]
```

```
2. Cos[1.5843 t] - 0.157799 t Cos[1.5843 t] - 0.531593 Sin[1.5843 t]
```

Now look at this:

```
Clear[amplitude]
amplitude[t_] = √((2 - 0.164845 t)^2 + (-0.550685)^2);
endtime = 150;
oscplot = Plot[{formulay[t], amplitude[t], -amplitude[t]},
  {t, 0, endtime}, PlotStyle →
   {{Blue, Thickness[0.01]}, {Thickness[0.005]}, {Thickness[0.005]}},
  PlotPoints → 200, AxesLabel → {"t", "y[t]"}, AspectRatio → 1/2,
  Prolog → {{Red, Thickness[0.02], Line[{{0, -10}, {endtime, -10}}]},
    {Red, Thickness[0.02], Line[{{0, 10}, {endtime, 10}}]}}];
```



The horizontal lines indicate that this oscillator is redlined at 10 in the sense that it will break apart or fry if | y[t] | ever gets bigger than 10.
This oscillator needs your help.
You've got to fix it by making small changes in the set-up.

One possibility that comes to mind is to look at the original differential equation

$$y''[t] + 2.51\, y[t] = 0.5 \sin\left[\sqrt{2.51}\, t\right]$$

with $y[0] = 2$ and $y'[0] = -1$:
and reduce the forcing term by going with

$$y''[t] + 2.51\, y[t] = B \sin\left[\sqrt{2.51}\, t\right]$$

with $y[0] = 2$ and $y'[0] = -1$
where B is some positive number comfortably less than 0.5.
In the long run, will doing this keep the oscillator under the red line?
How do you know?

□G.4.c.ii)

Another possibility that comes to mind is to look at the original differential equation

$$y''[t] + 2.51\, y[t] = 0.5\, \mathrm{Sin}\left[\sqrt{2.51}\; t\right]$$

with $y[0] = 2$ and $y'[0] = -1$:
and throw on a damper to get

$$y''[t] + b\, y'[t] + 2.51\, y[t] = 0.5\, \mathrm{Sin}\left[\sqrt{2.51}\; t\right]$$

with $y[0] = 2$ and $y'[0] = -1$
where b is a positive number about as small as you can make it so that $| y[t] |$ never exceeds 10.
Come up with your b.
Back up your answer with a decisive plot of the formula of your solution. No `NDSolve` here.

□G.4.c.iii)

Yet another possibility that comes to mind is to look at the original differential equation

$$y''[t] + 2.51\, y[t] = 0.5\, \mathrm{Sin}\left[\sqrt{2.51}\; t\right]$$

with $y[0] = 2$ and $y'[0] = -1$:
and change the forcing function slightly by going with

$$y''[t] + 2.51\, y[t] = 0.5\, \mathrm{Sin}[A\, t]$$

with $y[0] = 2$ and $y'[0] = -1$.
where A is a number as close to $\sqrt{2.51}$ as you can make it so that $| y[t] |$ never exceeds 10.
Does this work?
Back up your answer with a decisive plot of the formula of your solution. No `NDSolve` here.

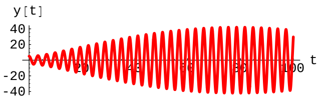
□G.4.d)

The forced undamped oscillator coming from

$$y''[t] + 4\, y[t] = 0.5\, \mathrm{Cos}[2.04\, t]$$

with $y[0] = 5$ and $y'[0] = 0$
is technically not in resonance.
Take a look at it:

```
Clear[f, y, fakey]
f[t_] = 3 Cos[2.04 t];
endtime = 100;
ndssol = NDSolve[{y''[t] + 4 y[t] == f[t], y[0] == 5, y'[0] == 0},
  y[t], {t, 0, endtime}, MaxSteps → 2000];
y[t_] = y[t] /. ndssol[[1]];
Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Red}},
 PlotPoints → 2 endtime, AxesLabel → {"t", "y[t]"}, AspectRatio → 1/4];
```



What is it about the the solutions of the characteristic equation of

$$y''[t] + 4\, y[t] = 0$$

that should have made you suspect right off the bat that forced undamped oscillator coming from

$$y''[t] + 4\, y[t] = 0.5\, \mathrm{Cos}[2.04\, t]$$

with $y[0] = 5$ and $y'[0] = 0$
is close to being in resonance?

G.5) Beating is near resonance

Experience with G.4) above will be helpful.

□G.5.a)

When you go with the undamped oscillator forced by

$$y''[t] + c\, y[t] = F\, \mathrm{Cos}[p\, t]$$

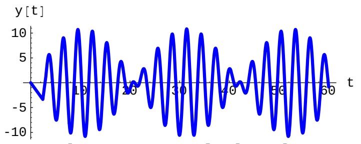
with $y[0] = 0$ and $y'[0] = 0$
and with $p - \sqrt{c}$ fairly small so that the oscillator is nearly in resonance, you get something lots of folks like to see.

```
F = 7;
c = 4;
p = 2.3;
Clear[y, t, yformula]
formula =
 DSolve[{y''[t] + c y[t] == F Cos[p t], y[0] == 0, y'[0] == 0}, y[t], t];
yformula[t_] = N[Chop[ComplexExpand[y[t] /. formula[[1]]]]]
```

```
5.42636 Cos[2. t] - 5.42636 Cos[2.3 t]
```

A rather plain, undistinguished formula.
But its plot is neither plain nor undistinguished:

```
oscplot =
Plot[yformula[t], {t, 0, 60},
  PlotStyle -> {{Blue, Thickness[0.01]}},
  AxesLabel -> {"t", "y[t]"}];
```



To see what's going on look at this trig identity:

Did you ever feel that trig identities can be more interesting outside the trig classroom than they were in the trig classroom?

```
Clear[A, a, b, t]
Simplify[2 A Sin[1/2 (a + b) t] Sin[1/2 (a - b) t]]
```

```
A (-Cos[a t] + Cos[b t])
```

Look at the formula for the oscillator:

```
yformula[t]
```

```
5.42636 Cos[2. t] - 5.42636 Cos[2.3 t]
```

Match these by setting $A = 5.42636$, $a = 2.3$ and $b = 2$ and to see that an equivalent formula for this oscillator is:

```
A = 5.42636;
a = 2.3;
b = 2;
newformula[t_] = 2 A Sin[1/2 (a + b) t] Sin[1/2 (a - b) t]
```

```
10.8527 Sin[0.15 t] Sin[2.15 t]
```

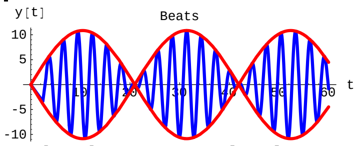
Good.
Now look at this embellished plot of the oscillator:

```
Clear[amplitude]
amplitude[t_] = 2 A Sin[1/2 (a - b) t]
```

```
10.8527 Sin[0.15 t]
```

```
beatplot = Plot[{amplitude[t], -amplitude[t]}, {t, 0, 60},
     PlotStyle -> {{Red, Thickness[0.01]}},
     DisplayFunction -> Identity];
```

```
revealingplot =
  Show[oscplot, beatplot, PlotLabel -> "Beats"];
```



Describe what you see and analyze newformulay[t] and the formula for amplitude[t] to explain why you see it. What do folks mean when they talk about beats?

□G.5.b)

What happened above came from

$$y''[t] + c\, y[t] = F\, \mathrm{Cos}[p\, t]$$

with $y[0] = 0$ and $y'[0] = 0$
and with $p - \sqrt{c}$ small.
In other words one oscillator was forced with another oscillator of almost the same frequency.
Got any idea of why you hear a beating sound when someone simultaneously strikes two tuning forks of about the same frequency?

G.6) Oscillators and shock absorbers - the meaning of the damping term $b\, y'[t]$*

Calculus&*Mathematica* thanks lead mechanics Aaron Finley and Don Happ of Don's Automotive in Homer, Illinois for some expert advice on shock absorbers.

□G.6.a.i) The undamped oscillator

The differential equation of the undamped oscillator is

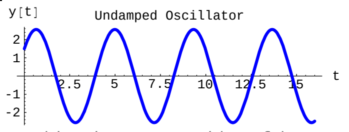
$$y''[t] + c\, y[t] = 0 \text{ with } c > 0.$$

Here is a sample with

$$c = 2.1,\; y[0] = 1.5,\; y'[0] = 3.0:$$

```
c = 2.1;
initialdisplacement = 1.5;
```

```
initialvel = 3.0;
endtime = 16;
Clear[s, y, y, t]
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;
Plot[y[t], {t, 0, endtime},
 PlotStyle → {{Thickness[0.01], Blue}}, PlotRange → All,
 AxesLabel → {"t", "y[t]"}, PlotLabel → "Undamped Oscillator"];
```



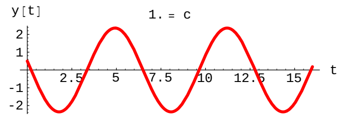
This gives you an idea of how your grandfather's tired old Cadillac with worn-out shock aborbers will oscillate immediately after you drive it over a big pot hole.

Say why you should have been able to predict in advance, with no plotting, that as t advances from from 0, the plot of y[t] had to go up before it could go down.
What happens when you keep y[0] the same but you go with
$y'[0] < 0$?

□G.6.a.ii) Frequency of the undamped oscillator

The differential equation of the undamped oscillator is
$y''[t] + c\,y[t] = 0$ with $c > 0$.
Here's a new sample with
$c = 1.0$, $y[0] = 0.5$, $y'[0] = -2.3$:

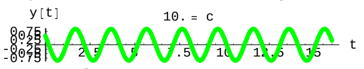
```
c = 1.0;
initialdisplacement = 0.5;
initialvel = -2.3;
endtime = 16;
Clear[s, y, y, t]
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;
littlec =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Red}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → c "= c"];
```



You may want to enlarge this graph by clicking on it and then dragging one of the corners.

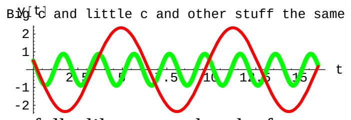
Here is what you get when you give a hefty increase to c and keep everything else the same:

```
c = 10.0;
Clear[y, t]
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;
bigc =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.015], Green}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → c "= c"];
```



Here they are together:

```
Show[bigc, littlec,
 PlotLabel → "Big c and little c and other stuff the same"];
```

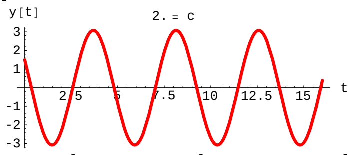


Most folks like to say that the frequency of an undamped oscillator is the measurement of the number of oscillations the oscillator makes every unit of time t.
Run additional experiments if you like and then you make the call:
Does increasing c increase or decrease the frequency of the oscillator?

□G.6.a.iii) Amplitude of undamped oscillators

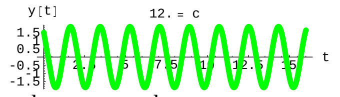
The differential equation of the undamped oscillator is
$y''[t] + c\,y[t] = 0$ with $c > 0$.
Here's yet another new sample with
$c = 2.0$, $y[0] = 1.5$, $y'[0] = -3.8$:

```
c = 2.0;
initialdisplacement = 1.5;
initialvel = -3.8;
endtime = 16;
Clear[y, t]
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;
littlec =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.01], Red}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → c "= c"];
```



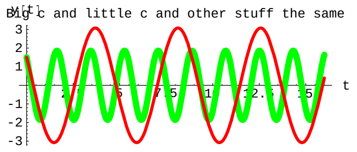
Here is what you get when you give a hefty increase to c and keep everything else the same:

```
c = 12.0;
Clear[y, t]
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;
bigc =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.02], Green}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → c "= c"];
```



Here they are together:

```
Show[bigc, littlec,
 PlotLabel → "Big c and little c and other stuff the same"];
```



Most folks like to say that the amplitude of an undamped oscillator is the measurement of the perpendicular distance between the t-axis and the highest point on the plot of the oscillator.
Run additional experiments if you like and then you make the call:
Does increasing c result in an increase or a decrease of the amplitude of the oscillator?

□G.6.a.iv) Cadillac versus Porsche

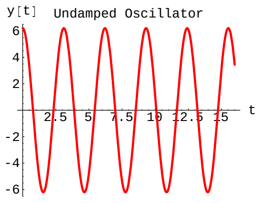
When you remove the shock absorbers from a car and you drive the car over a pot hole, then the the oscillation of the car immediately after hitting the pot hole is approximately modeled by the differential equation
$y''[t] + c\,y[t] = 0$.
When the car is your grandfather's Cadillac Sedan deVille you get a certain c in the model. When the car is your sister's Porsche 911, you get another c in the model.

Which do you think gives you thae larger c : The Cadillac Sedan deVille or the Porsche 911?
Explain how you came to your opinion.

□G.6.b.i) The damped oscillator

The differential equation of the undamped oscillator is
$y''[t] + c\,y[t] = 0$ with $c > 0$.
Here's yet another sample with
$c = 4.1$, $y[0] = 6.2$, $y'[0] = 0$:

```
c = 4.1;
initialdisplacement = 6.2;
initialvel = 0;
endtime = 16;
Clear[y, t]
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol⟦1⟧;
undamped = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Undamped Oscillator"];
```



This gives you an idea of how your grandfather's tired old Cadillac with worn-out shock aborbers will oscillate immediately after you drive it over a big pot hole.

Here is what you get when you go with a small, positive b and throw a

b y′[t] term

into

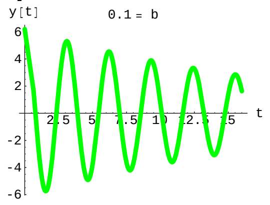
y″[t] + c y[t] = 0

to get

y″[t] + b y′[t] + c y[t] = 0

and you keep everything else the same:

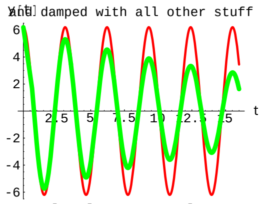
```
b = 0.1;
Clear[y, t]
ndssol =
 NDSolve[{y″[t] + b y′[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];
smallb =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.02], Green}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → b "= b"];
```



This gives you an idea of how your grandfather's Cadillac equipped with brand new light duty Monroe-matic shock aborbers will oscillate immediately after you drive it over a big pot hole.

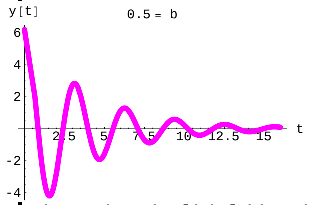
Here they are together:

```
Show[undamped, smallb,
 PlotLabel → "undamped and damped with all other stuff the same"];
```

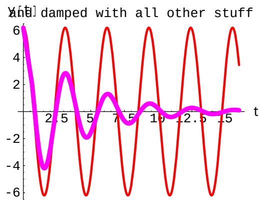


See what happens when you increase b just a little bit:

```
b = 0.5;
Clear[y, t]
ndssol =
 NDSolve[{y″[t] + b y′[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];
slightlybiggerb =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.02], Magenta}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → b "= b"];
```



```
Show[undamped, slightlybiggerb,
 PlotLabel → "undamped and damped with all other stuff the same"];
```



This gives you an idea of how your grandfather's Cadillac equipped with brand new medium duty Monroe-matic shock aborbers will oscillate immediately after you drive it over a big pot hole.

Why do you think folks like to say that when you go with a small, positive b, and a positive c, then

y″[t] + b y′[t] + c y[t] = 0

is the differential equation of a damped oscillator?
Does increasing b increase or decrease the effect of the damping?

□G.6.b.ii) Too much of a good thing: The overdamped oscillator

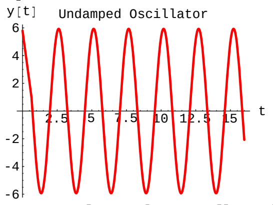
The differential equation of the undamped oscillator is

y″[t] + c y[t] = 0 with c > 0.

Here's a new sample with

c = 6.2, y[0] = 5.8, y′[0] = 3.1 :

```
c = 6.2;
initialdisplacement = 5.8;
initialvel = 3.1;
endtime = 16;
Clear[y, t]
ndssol = NDSolve[{y″[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];
undamped = Plot[y[t], {t, 0, endtime},
  PlotStyle → {{Thickness[0.01], Red}}, PlotRange → All,
  AxesLabel → {"t", "y[t]"}, PlotLabel → "Undamped Oscillator"];
```



You can damp this oscillator by going with a small positive b and throwing in a

b y′[t] term

into

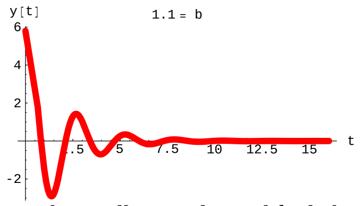
y″[t] + c y[t] = 0

to get

y″[t] + b y′[t] + c y[t] = 0

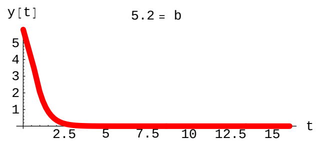
and keeping everything else the same:

```
b = 1.1;
Clear[y, t]
ndssol =
 NDSolve[{y″[t] + b y′[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];
smallb =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.02], Red}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → b "= b"];
```



Now the oscillator is damped fairly heavily.
See what happens when you damp it even more:

```
b = 5.2;
Clear[y, t]
ndssol =
 NDSolve[{y″[t] + b y′[t] + c y[t] == 0, y[0] == initialdisplacement,
   y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];
smallb =
 Plot[y[t], {t, 0, endtime}, PlotStyle → {{Thickness[0.02], Red}},
  PlotRange → All, AxesLabel → {"t", "y[t]"}, PlotLabel → b "= b"];
```



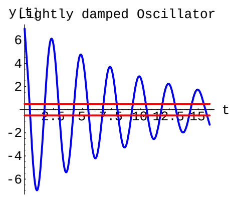
This is what your grandfather's old Cadillac equipped with brand new extra heavy duty Monroe-matic Gas Magnum shock absorbers will do immediately after you drive it over a big pot hole.

Why do most folks say that when you go with b′s that are too large, you run the risk of overdamping the oscillator?
Why is it unwise to put dump truck shock absorbers on a Porsche 911?

□G.6.b.iii) Custom damping

Look at this:

```
b = 0.2;
c = 6.2;
initialdisplacement = 7.0;
initialvel = -9.0;
endtime = 16;
Clear[y, t]
ndssol =
  NDSolve[{y″[t] + b y′[t] + c y[t] == 0, y[0] == initialdisplacement,
    y′[0] == initialvel}, y[t], {t, 0, endtime}];
y[t_] = y[t] /. ndssol[[1]];
lightlydamped = Plot[y[t], {t, 0, endtime},
   PlotStyle → {{Thickness[0.01], Blue}}, PlotRange → All,
   AxesLabel → {"t", "y[t]"}, PlotLabel → "Lightly damped Oscillator",
   Epilog → {{Thickness[0.01], Red, Line[{{0, 0.5}, {endtime, 0.5}}]},
     {Thickness[0.01], Red, Line[{{0, -0.5}, {endtime, -0.5}}]}}];
```



This is the plot of the damped oscillator coming from

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

with $y[0] = 7.0$, $y'[0] = -9.0$, $b = 0.2$ and $c = 6.2$.
Your job is to come up with a new damping coefficient b so that when you keep everything else the same, then the plot of the new damped oscillator:
→ oscillates between the red lines for when t is bigger than 6 but
→ oscillates above, between and below the red lines for as t runs from 0 to 6.
Illustrate your answer with a plot.

G.7) Damped, critically damped and overdamped unforced oscillators

□G.7.a)

Here is a formula for the solution of the undamped unforced oscillator

$$y''[t] + 25.0\,y[t] = 0$$

with random starter data:

```
b = 0;
c = 25.0;
Clear[y, yundamped, t]
linoscdiffeq = y″[t] + b y[t] + c y[t] == 0
ystarter = Random[Real, {2, 6}];
yprimestarter = Random[Real, {-4, 4}];

Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
  C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]};
Csols =
  Solve[{generalsol[0] == ystarter, generalsol′[0] == yprimestarter}];
yundamped[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
25. y[t] + y″[t] == 0
2.70867 Cos[5. t] - 0.623509 Sin[5. t]
```

You can damp this oscillator lightly by adding a modest damping term b y′[t]:

```
b = 0.8;
Clear[y, ylightlydamped, t]
linoscdiffeq = y″[t] + b y′[t] + c y[t] == 0
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
  C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]};
Csols =
  Solve[{generalsol[0] == ystarter, generalsol′[0] == yprimestarter}];
ylightlydamped[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
25. y[t] + 0.8 y′[t] + y″[t] == 0
2.70867 E^(-0.4 t) Cos[4.98397 t] - 0.408123 E^(-0.4 t) Sin[4.98397 t]
```

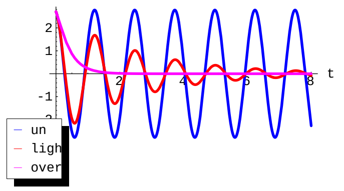
You can put on a really big damping term:

```
b = 12.0;
Clear[y, yoverdamped, t]
linoscdiffeq = y″[t] + b y′[t] + c y[t] == 0
Clear[C1, C2, z, z1, z2, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z];
generalsol[t_] =
  C1 E^(z1 t) + C2 E^(z2 t) /. {z1 → zsols[[1, 1, 2]], z2 → zsols[[2, 1, 2]]};
Csols =
  Solve[{generalsol[0] == ystarter, generalsol′[0] == yprimestarter}];
yoverdamped[t_] = Chop[ComplexExpand[generalsol[t] /. Csols[[1]]]]
```

```
25. y[t] + 12. y′[t] + y″[t] == 0
-0.625764 E^(-9.31662 t) + 3.33444 E^(-2.68338 t)
```

This is what folks call an overdamped oscillator.
Reason: No sines or cosines; so it doesn't oscillate at all. It just dies out to 0.
See all three:

```
Plot[{yundamped[t], ylightlydamped[t], yoverdamped[t]},
  {t, 0, 8}, PlotStyle → {{Thickness[0.01], Blue},
    {Thickness[0.01], Red}, {Thickness[0.01], Magenta}},
  AxesLabel → {"t", ""}, AspectRatio → 1/2,
  PlotLegend → {"un", "light", "over"}, LegendSize → 0.4];
```



Your mission is to keep c as defined above and find the critical value bcritical so that if

$$0 < b < \text{bcritical},$$

then the oscillator

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

is not overdamped (i.e. its solution formulas have sines and/or cosines), but if

$$\text{bcritical} < b,$$

then the oscillator

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

is overdamped (i.e. its solution formulas have no sines or cosines).
The code below may be useful.

```
Clear[C1, C2, z, z1, z2, b, generalsol, t]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z]
```

$$\{\{z \to 0.5\,(-1.\,b - 1.\,\sqrt{-100. + b^2})\},\ \{z \to 0.5\,(-1.\,b + \sqrt{-100. + b^2})\}\}$$

```
FYI: When you go with b = bcritical, then lot of folks say that
        the resulting oscillator is critically damped.
        Critically damped oscillators are no big deal.
```

□G.7.b)

This time go with cleared b and c, with c > 0. Your job this time is to find bcritical[c] (in terms of c) so that if

$$0 < b < \text{bcritical}[c],$$

then the oscillator

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

is not overdamped (i.e. its solution formulas have sines and/or cosines), but if

$$\text{bcritcal}[c] < b,$$

then the oscillator

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

is overdamped (i.e. its solution formulas have no sines or cosines).
The code below may be useful.

```
Clear[C1, C2, z, z1, z2, generalsol, t, b, c]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z]
```

$$\left\{\left\{z \to \frac{1}{2}\left(-b - \sqrt{b^2 - 4c}\right)\right\},\ \left\{z \to \frac{1}{2}\left(-b + \sqrt{b^2 - 4c}\right)\right\}\right\}$$

G.8) The effect of resistance in simple parallel electrical circuits*

□G.11.a)

```
You don't have to understand the
electrical jargon to be able to do this problem.
```

You are fortunate enough to have your own computer set-up right in your dorm room. One night, Brian, the EE student who lives across the hall comes in and says, "Part of my EE 250 homework for tomorrow is to analyze the effect of varying the size of the resistance on the current in a simple parallel electrical circuit. I don't have a clue about where to start."

At first, you think, "I'm scared because I'm a life science major and I don't know anything about electrical circuits." But you don't let on. Instead, you say, "Let me see that EE 250 textbook." In the book you see that in a simple parallel electrical circuit, one of the main measurements is

x[t] = voltage drop across the capacitor

Reading on, you see that x[t] is a solution of the oscillator diffeq

$$x''[t] + \frac{R\,x'[t]}{L} + \frac{x[t]}{C\,L} = 0$$

where L, C, and R are given positive numbers with

L = inductance
C = capacitance, and
R = resistance.

You say, "Brian, I don't even know what those words mean, but I think I can help you because I undertand the meaning of the damping term b in the oscillator diffeq

$$y''[t] + b\,y'[t] + c\,y[t] = 0.$$

Taking this as your clue, describe the way that solution plots of

$$x''[t] + \frac{R\,x'[t]}{L} + \frac{x[t]}{C\,L} = 0 \text{ for small positive R}$$

are fundamentally different in shape from solution plots of

$$x''[t] + \frac{R\,x'[t]}{L} + \frac{x[t]}{C\,L} = 0 \text{ for big positive R.}$$

G.9) Using Euler's identity $E^{(p + I q) t} = E^{p t} (\mathrm{Cos}[q\,t] + I\,\mathrm{Sin}[q\,t])$ to help to anticipate behavior of solutions of the the unforced oscillator diffeq

$$y''[t] + b\,y'[t] + c\,y[t] = 0$$

by glancing at the solutions of the characteristic equation $z^2 + b\,z + c = 0$*

□G.9.a) Solutions of characteristic equation having form p + I q and p − I q with p < 0 and q ≠ 0 signal that all solutions are damped sine waves squashed to 0 as t gets large

Here is an unforced linear oscillator diffeq:

```
b = Random[Real, {0.5, 1.5}];
c = Random[Real, {3, 8}];
startery = Random[Real, {-3, 3}];
starteryprime = Random[Real, {-2, 2}];
Clear[t, y]
linoscdiffeq = {y''[t] + b y'[t] + c y[t] == 0,
  y[0] == startery, y'[0] == starteryprime};
ColumnForm [linoscdiffeq]
```

```
4.49134 y[t] + 1.14174 y′[t] + y″[t] == 0
y[0] == -2.19003
y′[0] == -0.869714
```

The solutions of the characteristic equation are:

```
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z]
```

```
{{z → -0.57087 - 2.04094 I}, {z → -0.57087 + 2.04094 I}}
```

These have the form

$p + I\,q$ and $p - I\,q$ with $p < 0$ and $q \neq 0$.

Explain why this signals that all solutions are damped waves squashed to 0 as t gets large.

□Sample answer included as a gesture of friendship

If $p + I\,q$ and $p - I\,q$ with $p < 0$ and $q \neq 0$ are the solutions of the charcteristic equation of

$$y''[t] + b\,y'[t] + c\,y[t] = 0,$$

then all solutions of $y''[t] + b\,y'[t] + c\,y[t] = 0$ look like

$$K1\,E^{(p + I q) t} + K2\,E^{(p - I q) t}$$
$$= K1\,E^{p t} (\mathrm{Cos}[q\,t] + I\,\mathrm{Sin}[q\,t]) + K2\,E^{p t} (\mathrm{Cos}[q\,t] + I\,\mathrm{Sin}[q\,t])$$

These yield combinations of $E^{p t}\,\mathrm{Cos}[q\,t]$ and $E^{p t}\,\mathrm{Sin}[q\,t]$.

Because $p < 0$, the solutions are damped to 0 as t gets large.

Because $q \neq 0$, the solutions oscillate like sine waves as they are damped to 0.

Now you try one.

□G.9.b) Solutions of characteristic equation having form 0 + I q and 0 − I q with q ≠ 0 signal that all solutions are undamped sine waves

Here is an unforced linear oscillator diffeq:

```
b = 0;
c = Random[Real, {3, 5}];
startery = Random[Real, {-3, 3}];
starteryprime = Random[Real, {-2, 2}];

Clear[t, y]
linoscdiffeq = {y''[t] + b y'[t] + c y[t] == 0,
      y[0] == startery, y'[0] == starteryprime};
ColumnForm[Thread[linoscdiffeq]]
```

```
4.6302 y[t] + y″[t] == 0
y[0] == 2.94433
y′[0] == -0.196873
```

The solutions of the characteristic equation are:

```
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z]
```

```
{{z → 0. - 2.15179 I}, {z → 0. + 2.15179 I}}
```

These have the form

$0 + I\,q$ and $0 - I\,q$ with $q \neq 0$.

Explain why this signals that all solutions are undamped waves.

□G.9.c) Damped but not oscillating solutions

Here is an unforced linear oscillator diffeq:

```
b = Random[Real, {4, 6}];
c = Random[Real, {3, 4}];
startery = Random[Real, {5, 8}];
starteryprime = Random[Real, {-2, 2}];

Clear[t, y]
linoscdiffeq = {y''[t] + b y'[t] + c y[t] == 0,
      y[0] == startery, y'[0] == starteryprime};
ColumnForm[Thread[linoscdiffeq]]
```

```
3.87329 y[t] + 4.77453 y′[t] + y″[t] == 0
y[0] == 7.20924
y′[0] == -1.64156
```

The solutions of the characteristic equation are:

```
Clear[z]
charequation = z^2 + b z + c == 0;
zsols = Solve[charequation, z]
```

```
{{z → -3.73847}, {z → -1.03606}}
```

These have the form

p and q with $p < 0$ and $q < 0$.

Explain why this signals that all solutions collapse to 0 without oscillation.

G.10) Add'em up: The superposition principle*

The superposition principle says:
If y1[t] solves
$$y''[t] + b\, y'[t] + c\, y[t] = f[t] \quad \text{with } y[0] = r \text{ and } y'[0] = s$$
and if y2[t] solves
$$y''[t] + b\, y'[t] + c\, y[t] = g[t] \quad \text{with } y[0] = p \text{ and } y'[0] = q,$$
then
$$y3[t] = y1[t] + y2[t]$$
solves
$$y''[t] + b\, y'[t] + c\, y[t] = f[t] + g[t] \text{ with } y[0] = r + p \text{ and}$$
$y'[0] = s + q.$

Click on the right for an explanation

$$y3''[t] + b\, y3'[t] + c\, y3[t]$$
$$= (y1''[t] + y2''[t]) + b\,(y1'[t] + y2'[t]) + c\,(y1[t] + y2[t])$$

$$= (y1''[t] + b\, y1'[t] + c\, y1[t]) + (y2''[t] + b\, y2'[t] + c\, y2[t]) = f[t] +$$

and
$$y3[0] = y1[0] + y2[0] = r + p$$
and
$$y3'[0] = y1'[0] + y2'[0] = s + q.$$

None of these problems require formulas or other extensive calcultion.

□G.10.a.i) Add them up

Use the superposition principle to set numbers p and q so that if y1[t] solves
$$y''[t] + 3\, y'[t] + 8\, y[t] = f[t]$$
with $y[0] = 1$ and $y'[0] = 5$,
and y2[t] solves
$$y''[t] + 3\, y'[t] + 8\, y[t] = 0$$
with $y[0] = p$ and $y'[0] = q$
then $y[t] = y1[t] + y2[t]$
solves
$$y''[t] + 3\, y'[t] + 8\, y[t] = f[t]$$
with $y[0] = 3$ and $y'[0] = 0$.

□G.10.a.ii)

If yzeroinput[t] solves
$$y''[t] + 2\, y'[t] + 7\, y[t] = f[t]$$
with $y[0] = 0$ and $y'[0] = 0$,
and yunforced[t] solves
$$y''[t] + 2\, y'[t] + 7\, y[t] = 0$$
with $y[0] = 12$ and $y'[0] = -4$
then what differential equation does
$$y[t] = \text{yzeroinput}[t] + \text{yunforced}[t]$$
solve?
Make sure you give the starting data.

□G.10.a.iii)

If y1[t] solves
$$y''[t] + 2\, y'[t] + 7\, y[t] = \text{Sin}[t]$$
with $y[0] = 0$ and $y'[0] = 0$,
and y2[t] solves
$$y''[t] + 2\, y'[t] + 7\, y[t] = -\text{Sin}[t]$$
with $y[0] = 12$ and $y'[0] = -4$
then what differential equation does
$$y3[t] = y1[t] + y2[t]$$
solve?
Make sure you give the starting data.

□G.10.a.iv)

Calculus Cal told Careful Carrie that if y1[t] solves
$$y''[t] + 3\, y'[t] + 8\, y[t] = \text{DiracDelta}[t-1]$$
with $y[0] = 3$ and $y'[0] = 5$,
and if y2[t] solves
$$y''[t] + 3\, y'[t] + 8\, y[t] = \text{DiracDelta}[t-1]$$
with $y[0] = 7$ and $y'[0] = -4$,
then
$$y3[t] = y1[t] + y2[t]$$
solves
$$y''[t] + 3\, y'[t] + 8\, y[t] = \text{DiracDelta}[t-1]$$
with $y[0] = 10$ and $y'[0] = 1$.
What did Carrie say to Cal?

□G.10.a.v)

Given:
→ y1[t] solves
$$y''[t] + 1.2\, y'[t] + 9.7\, y[t] = \text{Sin}[t]$$
with $y[0] = 0$ and $y'[0] = 0$,
→ y2[t] solves
$$y''[t] + 1.2\, y'[t] + 9.7\, y[t] = E^{-0.4\, t}\, \text{Cos}[t]$$
with $y[0] = 0$ and $y'[0] = 0$,
→ y3[t] solves
$$y''[t] + 1.2\, y'[t] + 9.7\, y[t] = 0$$
with $y[0] = 1$ and $y'[0] = 2$.
What numbers p, q and r do you pick to make
$$y[t] = p\, y1[t] + q\, y2[t] + r\, y3[t]$$
solve
$$y''[t] + 1.2\, y'[t] + 9.7\, y[t] = 2\, \text{Sin}[t] + 3\, E^{-0.4\, t}\, \text{Cos}[t]$$
with $y[0] = 3$ and $y'[0] = 6$?

G.11) Boundary value problems: Shooting for a specified outcome

□G.11.a.i) Shooting the undamped oscillator for a specified boundary value

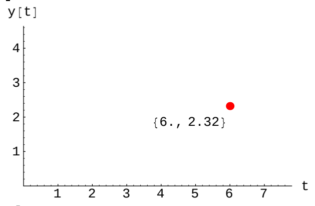
This problem starts with an undamped oscillator
$$y''[t] + 2.1\, y[t] = 0$$
with $y[0] = 0$ but $y'[0]$ yet unspecified.

```
Clear[diffeq, sollist, thissol, colors,
 t, starter, y, target, startingpoints, yprimestarter]
sollist = {};
(oscillator = {y''[t] + 2.1 y[t] == 0,
      y[0] == 0,
      y'[0] == yprimestarter}) // ColumnForm
```

```
2.1 y[t] + y″[t] == 0
y[0] == 0
y′[0] == yprimestarter
```

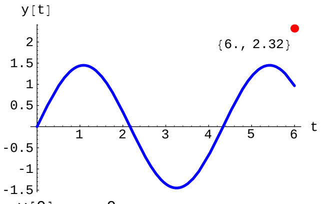
You are given the job of coming up with a starter velocity y′[0] so the oscillator y[t] satisfies y[6.0] = 2.32 to two accurate decimals.

```
endtime = 6.0;
goal = 2.32;
target = {endtime, goal};
bullseye = Graphics[{PointSize[0.03], Red, Point[target]}];
targetlabel = Graphics[Text[target, target, {1, 2}]];
Show[bullseye, targetlabel, Axes → True, AxesLabel → {"t", "y[t]"},
 PlotRange → {{0, 1.3 target⟦1⟧}, {0, 2 target⟦2⟧}}];
```



Make a guess
y′[0] = startervel = 2.1
and activate the following solution plotter code which plots and remembers your first try:

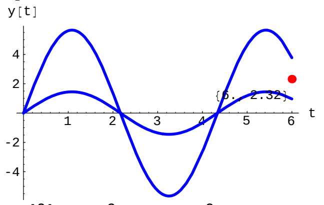
```
startervel = 2.1;
Clear[thistry, sollist];
thistry[t_] = y[t] /. NDSolve[oscillator /.
     yprimestarter -> startervel, y[t], {t, 0, endtime}][[1]];
sollist = {thistry[t]};

numsols = Length[sollist];;
startingpoints = N[sollist /. t -> 0, 3];
names = Table[startingpoints[[k]], {k, numsols}];

plotsols =
 Plot[Evaluate[sollist], {t, 0, endtime}, AspectRatio -> 1/GoldenRatio,
 PlotStyle -> {{Thickness[0.01], Blue}}, AxesLabel -> {"t", "y[t]"},
 DisplayFunction -> Identity];

Show[plotsols, bullseye, targetlabel, PlotRange -> All,
  DisplayFunction -> $DisplayFunction];

TableForm[{startingpoints, N[sollist /. t -> endtime, 3]},
 TableHeadings -> {{"y[0]", "y[6]"}, None}]
```



```
y[0]      0.
y[6]      0.966
```

The solution corresponding to y′[0] = 2.1 undershoots the target. You respond by increasing your starting value on y′[0] and trying again:

```
startervel = 8.2;
Clear[thistry];
thistry[t_] = y[t] /. NDSolve[oscillator /.
     yprimestarter → startervel, y[t], {t, 0, endtime}]⟦1⟧;
PrependTo[sollist, thistry[t]];
numsols = Length[sollist];
If[numsols > 3, numsols -= 1; sollist = Drop[sollist, -1]];
startingpoints = N[sollist /. t → 0, 3];
names = Table[startingpoints⟦k⟧, {k, numsols}];
plotsols =
 Plot[Evaluate[sollist], {t, 0, endtime},
  AspectRatio → 1/GoldenRatio, PlotStyle → {{Thickness[0.01], Blue}},
  AxesLabel → {"t", "y[t]"}, DisplayFunction → Identity];
Show[plotsols, bullseye, targetlabel, PlotRange → All,
  DisplayFunction → $DisplayFunction];
TableForm[{startingpoints, N[sollist /. t → endtime, 4]},
 TableHeadings → {{"y[0]", "y[6]"}, None}]
```



```
y[0]      0.        0.
y[6]      3.773     0.9663
```

Close, but no cigar.
Interact with this last code by changing the startervel values and rerunning. The code is written to keep track of your last three attempts. Remember your job is to come up with a starting value that makes y[6] agree with 2.32 to two accurate decimals (after rounding).

□G.11.a.ii) Shooting the damped oscillator for a specified boundary value

This problem starts with a damped oscillator version of the oscillator in part i) immediately above:

$y''[t] + 0.5\,y'[t] + 2.1\,y[t] = 0$

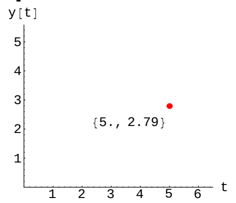
with $y[0] = 1.3$ but $y'[0]$ yet unspecified.

```
Clear[diffeq, sollist, thissol, colors,
 y, t, starter, target, startingpoints, yprimestarter]
sollist = {};
(oscillator = {y''[t] + 0.5 y'[t] + 2.1 y[t] == 0,
      y[0] == 1.3,
      y'[0] == yprimestarter}) // ColumnForm
```

```
2.1 y[t] + 0.5 y′[t] + y″[t] == 0
y[0] == 1.3
y′[0] == yprimestarter
```

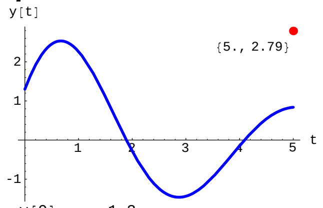
You are given the job of coming up with a starter velocity y′[0] so the oscillator y[t] satisfies y[5.0] = 2.79 to two accurate decimals.

```
endtime = 5.0;
goal = 2.79;
target = {endtime, goal};
bullseye = Graphics[{PointSize[0.03], Red, Point[target]}];
targetlabel = Graphics[Text[target, target, {1, 2}]];
Show[bullseye, targetlabel, Axes → True, AxesLabel → {"t", "y[t]"},
 PlotRange → {{0, 1.3 target⟦1⟧}, {0, 2 target⟦2⟧}}];
```



Make a guess
y′[0] = startervelocity = 3.6
and activate the following solution plotter code which plots and remembers your first try:

```
startervel = 3.6;
Clear[thistry, sollist];
thistry[t_] = y[t] /. NDSolve[oscillator /.
     yprimestarter → startervel, y[t], {t, 0, endtime}]⟦1⟧;
sollist = {thistry[t]};
numsols = Length[sollist]; Null;
startingpoints = N[sollist /. t → 0, 3];
names = Table[startingpoints⟦k⟧, {k, numsols}];
plotsols = Plot[Evaluate[sollist], {t, 0, endtime},
  AspectRatio → 1/GoldenRatio, PlotStyle → {{Thickness[0.01], Blue}},
  AxesLabel → {"t", "y[t]"}, DisplayFunction → Identity];
Show[plotsols, bullseye, targetlabel, PlotRange → All,
  DisplayFunction → $DisplayFunction];
TableForm[{startingpoints, N[sollist /. t → endtime, 3]},
 TableHeadings → {{"y[0]", "y[6]"}, None}]
```

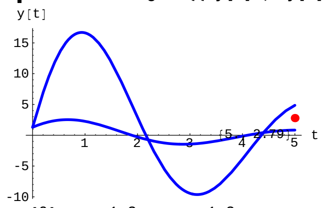


```
y[0]      1.3
y[6]      0.839
```

The solution corresponding to y′[0] = 3.6 undershoots the target. You respond by increasing your starting value on y′[0] and trying again:

```
startervel = 30.2;
Clear[thistry];
thistry[t_] = y[t] /. NDSolve[oscillator /.
     yprimestarter → startervel, y[t], {t, 0, endtime}]⟦1⟧;
PrependTo[sollist, thistry[t]];
numsols = Length[sollist];
If[numsols > 3, numsols -= 1; sollist = Drop[sollist, -1]];
startingpoints = N[sollist /. t → 0, 3];
names = Table[startingpoints⟦k⟧, {k, numsols}];
```

```
plotsols = Plot[Evaluate[sollist], {t, 0, endtime},
   AspectRatio → 1/GoldenRatio, PlotStyle → {{Thickness[0.01], Blue}},
   AxesLabel → {"t", "y[t]"}, DisplayFunction → Identity];
Show[plotsols, bullseye, targetlabel, PlotRange → All,
    DisplayFunction → $DisplayFunction];
TableForm[{startingpoints, N[sollist /. t → endtime, 4]},
 TableHeadings → {{"y[0]", "y[6]"}, None}]
```



```
y[0]      1.3        1.3
y[6]      4.863      0.8386
```

Close, but no cigar.
Interact with this last code by changing the startervel values and re-running. The code is written to keep track of your last three attempts.
Remember your job is to come up with a starting value that makes y[5] agree with 2.79 to two accurate decimals (after rounding).

□G.11.b) For your information

The trail and error shooter method you used above may seem dinky and simple-minded. It is. But many professional experts prefer this method over any other.