

# Laplace Transforms

```
In[1]:= SetDirectory@NotebookDirectory[];
<< "../MMA library.m"
```

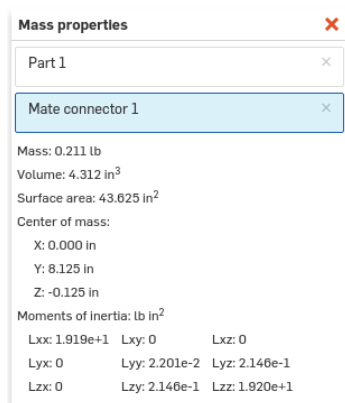
## iSIM problem

```
In[62]:= With[{context = "isim`"}, If[Context[] ≠ context, Begin[context]]];
Dynamic[Refresh[Context[], UpdateInterval → 1]]
```

```
Out[62]= Global`
```

```
In[54]:= true = -m g l Sin[θ[t]] - d θ'[t] == i θ''[t];
linearized = true /. Sin[θ[t]] → θ[t]
```

```
Out[55]= -g l m θ[t] - d θ'[t] == i θ''[t]
```



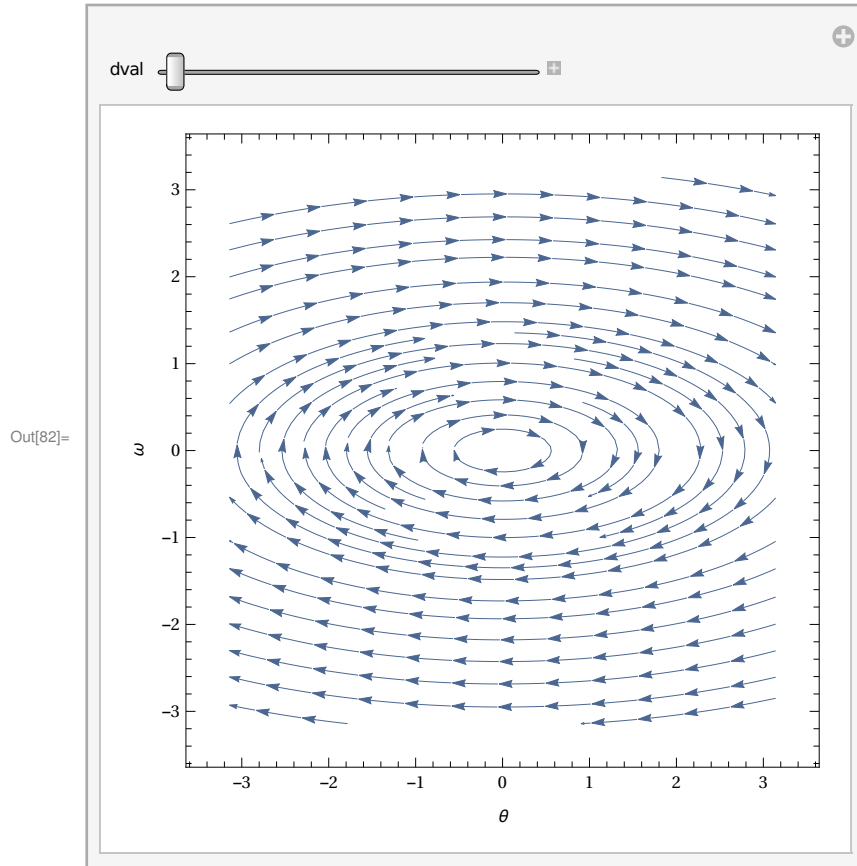
```
In[48]:= params = <|
  m → QuantityMagnitude[UnitConvert[0.211 lb, "Kilograms"]],
  g → QuantityMagnitude@UnitConvert@Earth(planet) ["Gravity"],
  l → QuantityMagnitude@UnitConvert[8.125 in, "Meters"],
  i → QuantityMagnitude@UnitConvert[19.2 in²lbf, "Meters"² * "Newtons"] |>
```

```
Out[48]= <| m → 0.095708, g → 9.80, l → 0.206375, i → 0.0551004 |>
```

```

In[81]:= components = {θdot, -m g l θ - d θdot};
Manipulate[StreamPlot[components /. d → dval, {θ, -Pi, Pi},
  {θdot, -Pi, Pi}, FrameLabel → {"θ", "ω"}], {dval, 0, 2}, SaveDefinitions → True]
CloudDeploy[%, Permissions → "Public"]

```



```

Out[83]= CloudObject[
  https://www.wolframcloud.com/objects/1d3ceb0b-1063-494d-bfe6-c2f220cb7cf2]

```

```

In[52]:= Solve[r^2 + d r + m g l == 0, r]

```

```

Out[52]= {{r → 1/2 (-d - Sqrt[d^2 - 4 m g l])}, {r → 1/2 (-d + Sqrt[d^2 - 4 m g l])}}

```

```

In[94]:= sol = DSolveValue[linearized, θ[t], t]

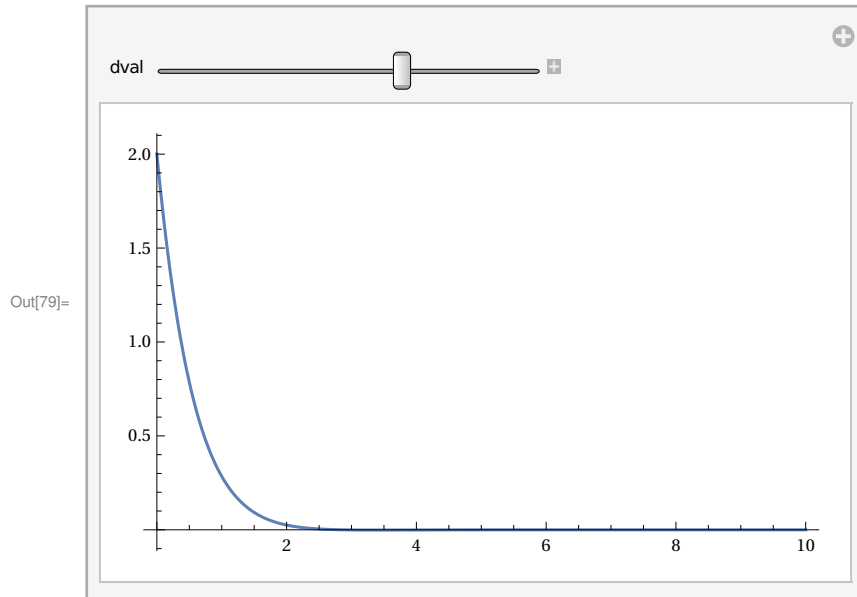
```

```

Out[94]= e^(1/2 (-d/ism'i - Sqrt[d^2 - 4 ism'g ism'i ism'l ism'm]/ism'i) t) C[1] + e^(1/2 (-d/ism'i + Sqrt[d^2 - 4 ism'g ism'i ism'l ism'm]/ism'i) t) C[2]

```

```
In[79]:= Manipulate[Plot[sol /. params /. d -> dval /. {C[1] -> 1, C[2] -> 1},
  {t, 0, 10}, PlotRange -> Full], {dval, 0, .3}, SaveDefinitions -> True]
CloudDeploy[%, Permissions -> "Public"]
```



```
Out[80]= CloudObject[
  https://www.wolframcloud.com/objects/425c64f1-fea8-42fd-8501-36b66b2dbd43]
```

```
In[99]:= With[{context = "isim`"}, If[Context[] == context, End[]]];
Dynamic[Refresh[Context[], UpdateInterval -> 1]]
```

```
Out[99]= Global`
```

## Laplace

```
In[101]:= With[{context = "l`"}, If[Context[] != context, Begin[context]]];
Dynamic[Refresh[Context[], UpdateInterval -> 1]]
```

```
Out[101]= Global`
```

```
In[115]:= bigY = (s + 1) / (s^2 + 5 s + 4)
Apart[bigY]
InverseLaplaceTransform[bigY, s, t]
```

```
Out[115]= 
$$\frac{1 + s}{4 + 5 s + s^2}$$

```

```
Out[116]= 
$$\frac{1}{4 + s}$$

```

```
Out[117]= 
$$e^{-4 t}$$

```

```

In[153]:= bigY = 1 / (s^2 + 2 s + 2)
          split = Apart[bigY]
          InverseLaplaceTransform[#, s, t] & /@ split
          FullSimplify@InverseLaplaceTransform[bigY, s, t]

Out[153]=  $\frac{1}{2 + 2 s + s^2}$ 

Out[154]=  $\frac{1}{2 + 2 s + s^2}$ 

Out[155]=  $(2 \text{DiracDelta}[t] + 2 \text{DiracDelta}'[t] + \text{DiracDelta}''[t])^{-\text{DiracDelta}[t]}$ 

Out[156]=  $e^{-t} \sin[t]$ 

In[147]:= LaplaceTransform[f'''[t], t, s] // TraditionalForm

Out[147]//TraditionalForm=
 $s^3 (\mathcal{L}_t[f(t)](s)) - f''(0) - s f'(0) - f(0) s^2$ 

With[{context = "l`"}, If[Context[] == context, End[]]];
Dynamic[Refresh[Context[], UpdateInterval -> 1]]

```

---

## Scratch Work

```
In[94]:= exportNotebookPDF[]
```

```
In[157]:=  1/(s^2+2s+2)
```

```
In[96]:= EmbedCode[CloudDeploy[APIFunction[{"City" → "String"},
      WeatherData[#city, "Temperature"] &], Permissions → "Public"], "Python"]
```

### Embeddable Code

Use the code below to call the Wolfram Cloud function from Python:

Code

Copy to Clipboard

```
from urllib import urlencode
from urllib2 import urlopen

class WolframCloud:

    def wolfram_cloud_call(self, **args):
        arguments = dict([(key, arg) for key, arg in args.iteritems()])
        result = urlopen("http://www.wolframcloud.com/objects/b59d9861-219a-4cc6-868a-
ea3cde7c09a5", urlencode(arguments))
        return result.read()

    def call(self, City):
        textresult = self.wolfram_cloud_call(City=City)
        return textresult
```

```
In[89]:= WeatherData["Boston", "Temperature"]
```

```
Out[89]= 14.4
```

```
In[148]:= Rotate[{1, 0}, Pi/3]
```

```
Out[148]= {1, 0}
```

```
In[162]:= FileNames["/dev/ttyACM*"]
```

```
Out[162]= {/dev/ttyACM0}
```



```
In[5]:= findArduino[]
```

```
>> /dev/ttyACM0
```

... OptionValue: Unknown option ArduinoInstallLocation for ArduinoLink`Private`ArduinoOpenDriver.

... OptionValue: Unknown option ArduinoInstallLocation for ArduinoLink`Private`ArduinoOpenDriver.

```
Out[5]= DeviceObject[
```

	Class: Arduino	ID: 2
	Status:  Connected (/dev/ttyACM0)	

In[175]:= **arduino**

Out[175]= DeviceObject [  Class: Arduino ID: 1  
Status:  Connected (/dev/ttyACM0) ]

In[204]:= **DeviceClose[arduino]**

In[4]:= **DeviceOpen["Arduino", "/dev/ttyACM0"]**

 **SerialLink`SerialPortOpen:** Could not open the port /dev/ttyACM0.

Out[4]= \$Failed

In[205]:= **LaplaceTransform[ $\theta'[t]$ , t, s] // TraditionalForm**


Out[205]/TraditionalForm=  
 $s(\mathcal{L}_t[\theta(t)](s) - \theta(0))$



In[207]:= **\$UserBaseDirectory**



Out[207]= /home/eric/.Mathematica

**DeviceReadTimeSeries**


In[69]:= **ts = DeviceReadTimeSeries["Arduino", {3, .01}, "A0"]**  
**ts = TimeSeriesResample[ts]**

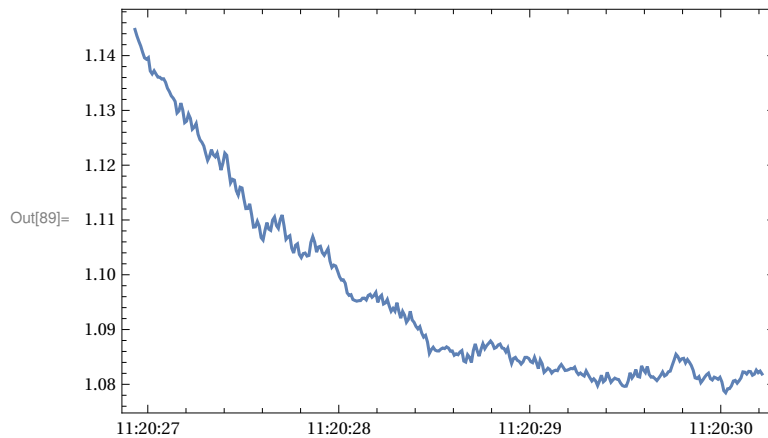
 **DeviceReadTimeSeries:** The result may have taken longer to obtain and/or may have a smaller number of data points because 178 measurement(s) took longer than the requested interval 0.01`, about 0.011506050561797751` seconds (on average).

Out[69]= TimeSeries [   Time: 11:20:26.832 to 11:20:30.216  
Data points: 299 ]

Out[70]= TimeSeries [   Time: 11:20:26.832 to 11:20:30.215  
Data points: 339 ]

```
In[88]:= smoothed = MovingAverage[ts, 100 ms]
DateListPlot[smoothed]
```

```
Out[88]= TimeSeries[ Time: 11:20:26.932 to 11:20:30.215  
Data points: 329]
```



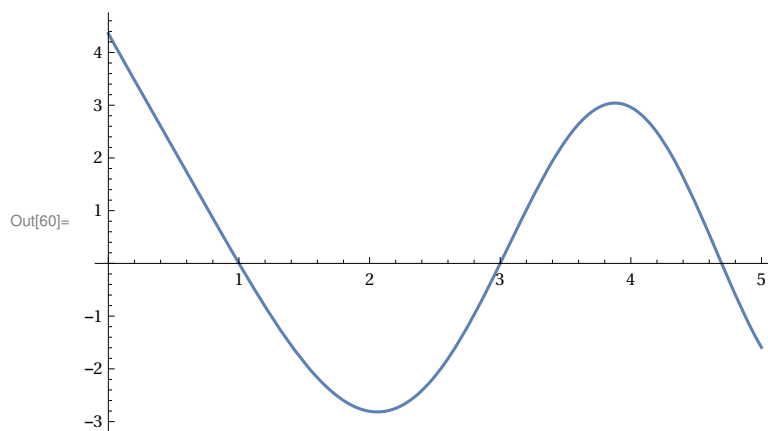
```
In[93]:= arduino // InputForm
```

```
Out[93]/InputForm=
DeviceObject[{"Arduino", 2}]
```

```
In[59]:= NDSolveValue[{f''[t] == -t f[t] + 1, f[3] == f[1] == 0}, f[t], {t, 0, 5}]
```

```
Plot[%, {t, 0., 5.}]
```

```
Out[59]= InterpolatingFunction[ Domain: {{0., 5.}}  
Output: scalar][t]
```



```
In[65]:= DeviceRead["Arduino", "A0"]
```

```
Out[65]= 1.14370 V
```