Project 2

Linear, second order differential equations arise frequently in science and engineering (even in pure mathematics, they have a somewhat privileged position). This project *introduces* an application of second order equations.

A simple Spring--Mass System

Consider a weight of mass m kilograms hanging on a spring (the spring is attached to the ceiling). The original length of the spring is ℓ meters and its elongated length (where the elongation is due to the weight) is $\ell + L$ meters. The spring is pulled downward and released. Our goal is to find an ODE that models this situation.

Our starting point will be Newton's second law: $F = ma$. We will pick a coordinate system where the *downward* direction is the positive direction. Also, the "zero" of the system is the spring's length with the mass attached (in particular, $\ell + L$ units down from the ceiling).

There are two forces acting on the spring, what are they. Give your answer below. (You answer *must* be in complete sentences as in "The forces on the spring are X and Y").

Fs (the force due to the spring that acts upwards) and Fg (force of gravity) are the forces acting on the spring.

Let's (suggestively) call these forces F_s and F_g. The force F_s is given by Hooke's Law. In the cell below, write Hooke's law (again, using complete sentences such as "Hooke's law says that F_s is ..." or something similar. Be clear about what the variables you are using mean).

Hooke's law concludes that the force that acts upon a spring Fs states that if the elongation (L) of the spring is small, the spring force (k) is very nearly proportional to L. This is known as Hooke's Law --> Fs = -kL , where the constant of proportionality k is called the spring constant, and the minus sign is due to the fact that the spring force acts in the upward (negative) direction. Since the mass is in equilibrium, the two forces balance each other, which means that --> w + Fs = mg – kL = 0. w is weight (w = mg). It is clear that this formula is proportional on both sides.

If $u(t)$ represents the position of the mass at time t, then give a formula for F_s. Be careful with the coordinate system, where the zero is, and the precise definition of L. (As above, use complete sentences!)

Fs = -k(L + u) - this is the equation for the spring force when when the spring is extended past/away from its natural position (0). You know this is the case when L + u > 0, and the spring force is directed upward.

Fs = kIL + uI - this is the equation for the spring force when the spring is compressed a certain distance from the natural position (0). However,when L + u < 0, you can conclude that |L+u| = −(L+u) since the absolute value would be actively flipping the inside operation to positive, so Fs = −k(L + u) regardless. It does not matter where the position of the mass, the force exerted by the spring is always expressed with Fs = −k(L + u).

The force F_g is given by Newton's second law. In the cell below, write a formula for this force (again, using complete sentences such as "Hooke's law says that Fs is ..." or something similar. Be clear about what the variables you are using mean).

Newton's second law states that the acceleration of an object is dependent upon two variables - the net force acting upon the object and the mass of the object. Fg = ma

m = mass (kg) a = acceleration (m/s^2)

Thus, you have an expression of the form $F = F_g + F_s$ and F_g and F_s have the formulas you have found above. According to Newton's Second Law, $F = ma$, using this and your expressions for F_s and F_g, write an ODE that descibes the motion of the mass. (Again, use complete sentences).

The sum of the forces is F = ma. The sum of the forces is also equal to F = Fg + Fs. Fg pulls down on the spring and Fs pulls up on the spring. If you set these two equations equal to each other, you get the expression below.

- as defined in the Hooke's Law equation, Fs is replaced by -kL, the elongation formula and Fg is replaced by mg (mass times gravity)

the sum of the forces F = $\sum$ F = Fg + Fs = 0

--> Fg = -Fs

--> mg = -(-kL)

--> mg = kL

--> mu''(t) = -k(L + u(t)) + mg

--> mu''(t) = -k(L + u(t)) + (kL) > substituted kL for mg as found above

--> mu''(t) + ku(t) = 0 > -kL and +kL cancel and now you have your ODE

Assuming that $m = 2$ kg, $\ell = 1$ meter, and $L = .1$ meter, and $g = 10$ meters per second squared, replace all letters (other than $u(t)$) with numbers. In other words, if, for example, there is a parameter k in your expression for F_s, replace this with a number that you compute from the given data. Write that ODE below (again - complete sentences).

mu''(t) + ku(t) = 0 > ODE

mg = kL > found above to be true, use to solve for k

(2)(10) = k(0.1) 20 = 0.1k

k = 20/0.1 k = 200

- Now that I have found k given the values above, I now plug in the k value into the equation I found for the ODE from the last question.

--> (2)u''(t) + (200)u(t) = 0

--> 2u''(t) + 200u(t) = 0 > now the ODE has coefficients, found by solving for k and plugging in for m

In the cell below, use `dsolve` to find a solution to this ODE.

In [77]:
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *

# 2u''(t) + 200u(t) = 0

# 2(u''(t) + 100u(t)) = 0

# u''(t) + 100u(t) = 0

t = symbols('t')
u = Function('u')

deq = Eq(2*diff(u(t), t, t) + 200*u(t))
xsoln = dsolve(deq, u(t))
pprint(xsoln)
```

```
u(t) = C1·sin(10·t) + C2·cos(10·t)
```

Assume that the starting position of the spring is $\ell + L$ (that is, 1.1 meters) down from the cieling (this is just the "zero" of the coordniate position) and it is tapped so that is starts with an upward speed of .01 meters per second. Write and IVP that describes this motion (this is just your ODE above combined with an initial condition). *Be mindful of what the initial* ***velocity*** *is!!* (again - complete sentences).

The ODE is combined with an initial condition here to find the IVP that describes this motion.

> u(0) = 0 (initial position) u'(0) = 0.01 (initial velocity)

--> 2u''(t) = 20 - 200(u(t) + 0.1)

--> 2u''(t) + 200u(t) = 0

Use `dsolve` to find a solution to this IVP.

In [78]:
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *

t = symbols('t')
u = Function('u')

deq = Eq(2*diff(u(t), t, t) + 200*u(t))
xsoln = dsolve(deq, u(t), ics={u(0): 0, diff(u(t), t).subs(t, 0): 0.01})
pprint(xsoln)
```

```
u(t) = 0.001·sin(10·t)
```

Adding Damping

A *dashpot* is a type of damping force. It always acts in a way that opposes movement. This means its sign is always opposite of velocity. It is also proportional to the speed. Assume this constant of proportionality is 1. Modify your IVP above to include a damping force. (Complete sentences!!)

--> Fd(t) = -γu'(t) # Damping force --> Fs(t) = −k(L + u(t)) # Spring force --> L = 0.1 # Elongation --> y = 1 # constant of proportionality These are the initial conditions: u(0) = 1.1 u'(0) = 0.01

Here is where I reduced/plugged into the unsimplified ODE to come out wiht a nonhomogenous ODE at the end = 0 mu''(t) = kL + Fs(t) + Fd(t)

2u''(t) = 20 - 200(u(t) + 0.1) - u'(t)

mu''(t) = -k(L + u(t)) + mg - u'(t)

mu''(t) = -k(L + u(t)) + (kL) - u'(t) # kL replaces mg as solved previously

mu''(t) + ku(t) + u'(t) = 0 # kL and -kL cancel

2u''(t) + u'(t) + 200u(t) = 0 # put damping ODE in second order form

In the cell below, use `dsolve` to find the solution to this IVP.

In [79]:
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *

t = symbols('t')
u = Function('u')

deq = Eq(2*diff(u(t), t, t) + diff(u(t), t) + 200*u(t))
xsoln = dsolve(deq, u(t), ics={u(0): 0, diff(u(t), t).subs(t, 0): 0.01})
pprint(xsoln)
```

$$u(t) = \frac{0.00100031264656071 \cdot \sin\left(\frac{\sqrt{1599} \cdot t}{4}\right)}{\sqrt[4]{e^{t}}}$$

Adding an External Force

Now, assume that an external force is applied. That is, a force $F(t)$ (such as a magnetic force, or hitting it with a hammer, etc) is applied. Add this force to the IVP above. Write the differential equation in the standard form the non-homogeneous equations are written in. Again - complete sentences!

2u''(t) = 20 - + u'(t) + 200u(t) = $F(t)$

These are the initial conditions: u(0) = 1.1 u'(0) = 0.01

2u''(t) + u'(t) + 200u(t) = $F(t)$ > This is the ODE with te initial conditions

2u''(t) + u'(t) + 200u(t) = $F(t)$

Since there is an external force on the system, the ODE is equal to the external force and not 0

Use `dsolve` to find a solution to this IVP below.

In [80]:
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *

t = symbols('t')
u = Function('u')
f = Function('f')

deq = Eq(2*diff(u(t), t, t) + diff(u(t), t) + 200*u(t), f(t))
xsoln = dsolve(deq, u(t), ics={u(0): 0, diff(u(t), t).subs(t, 0): 0.01})
xsoln
```

Out[80]:
$$u(t) = \left(\frac{2\sqrt{1599}\int f(t)e^{\frac{t}{4}}\sin\left(\frac{\sqrt{1599}t}{4}\right)dt}{1599} + 0.0500156323280355\int_0^0 f(t)e^{\frac{t}{4}}\sin\left(\frac{\sqrt{1599}t}{4}\right)dt\right)\cos\left(\frac{\sqrt{1599}t}{4}\right) + \left(\frac{2\sqrt{1599}\int f(t)e^{\frac{t}{4}}\cos\left(\frac{\sqrt{1599}t}{4}\right)dt}{1599} - 0.0500156323280355\int_0^0 f(t)e^{\frac{t}{4}}\cos\left(\frac{\sqrt{1599}t}{4}\right)dt + 0.00100031264656071\right)\sin\left(\frac{\sqrt{1599}t}{4}\right)$$

Now, let the constant of proportioanlity from the damping force be $\gamma > 0$. Also, let the external force $F(t) = 0$. Write the corresponding ODE below (don't worry about initial conditions). Use complete sentences.

--> Fd(t) = -γu'(t) # Damping force --> γ > 0 # constant of proportionality

$F(t)$ = 2u''(t) = 20 - + γu'(t) + 200u(t) = 0

--> The constant of proportionality has now become a part of the ODE for the external force since we are now including a constant that is greater than 0 we can not just assume it to =1 and must now include it as its own variable.

2u''(t) + γu'(t) + 200u(t) = 0

There is a specific value of γ at which a qualitative change in the nature of the solutions happens. Find this value of γ and call it γ_c (the c is for "critical"). Write (in complete sentences) your answer below. Be sure to explain why this is the critical value (e.g. you can just put your derivation of this value).

> The nature of the solution changes as γ passes through the value 2(km)**(1/2) The motion with γ = 2(km)**(1/2) is critically damped $\gamma_c = 2(((km)**(0.5)) \#$ *This is the critically damped formula* $\gamma_c = 2(((200)(2)))**(0.5)) \#$ Substitute in the given initial values from above for k and m $\gamma_c = 40$ # solve for critically damped variable

For a value of $\gamma < \gamma_c$ solve the IVP using `dsolve`.

In [81]:
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *

t = symbols('t')
u = Function('u')

deq = Eq(2*diff(u(t), t, t) + 30*diff(u(t), t) + 200*u(t))
xsoln = dsolve(deq, u(t), ics={u(0): 0, diff(u(t), t).subs(t, 0): 0.01})
pprint(xsoln)
```

$$u(t) = \frac{0.00151185789203691 \cdot \sin\left(\frac{\sqrt{71} \cdot t}{2}\right)}{\left(e^{t}\right)^{15/2}}$$

For $\gamma = \gamma_c$ solve he IVP using `dsolve`.

In [82]:
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *

t = symbols('t')
u = Function('u')

deq = Eq(2*diff(u(t), t, t) + 40*diff(u(t), t) + 200*u(t))
xsoln = dsolve(deq, u(t), ics={u(0): 0 , diff(u(t), t).subs(t, 0): 0.01})
pprint(xsoln)
```

```
                -10·t
u(t) = 0.01·t·e
```

For a value of $\gamma > \gamma_c$ solve the IVP using `dsolve`.

In [83]:
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *

t = symbols('t')
u = Function('u')

deq = Eq(2*diff(u(t), t, t) + 50*diff(u(t), t) + 200*u(t))
xsoln = dsolve(deq, u(t), ics={u(0): 0, diff(u(t), t).subs(t, 0): 0.01})
pprint(xsoln)
```

$$u(t) = \left(0.000666666666666667 - 0.000666666666666667 \cdot e^{-15 \cdot t}\right) \cdot e^{-5 \cdot t}$$

Grading.

This project is worth a total of 50 points. There are 18 "blanks" above. Each one is worth 2.5 points. You get 5 points just for turning it in (I know: I'm great.)

For the cells requiring a short answer (i.e. not the coding cells) your work will be assessed using the following rubric

- Is your answer written in complete sentences in the correct location? (.5 pt)
- Does your response answer the question? (2 pts)

For the other questions, you will be graded on:

- Does your code produce output? And is the output there? (.5pt)
- Is your output correct? (2 pts)

Deliverable

You will export this file as a .html file, print that as a pdf and turn it in via Gradescope.

In [ ]:
```

```