Chapter 12 Variation of parameters

The method of variation of parameters applies to nonhomogeneous second order ODEs. The method is important because it solves the largest class of equations. Specifically included are functions like f(x) , log x, abs x, and e to hyper exponent. When using CAS apps like Wolfram Alpha, the user just shines on the method implementation, relying on the CAS to do whatever is necessary.

Cutting and pasting and Wolfram Alpha. Wolfram Alpha is amenable to accepting pasted entries. In this chapter pastable expressions are given a distinctive boundary fence, exemplified by the sample: !| abcdef |!
In the above pseudo-entry, only the alpha characters would be copied for transfer to Wolfram Alpha.

12.1 Solve $y''' + y' = \sec x$

The entry is made into Wolfram Alpha:

!| y''' + y' = sec(x) |!

and the answer is received:

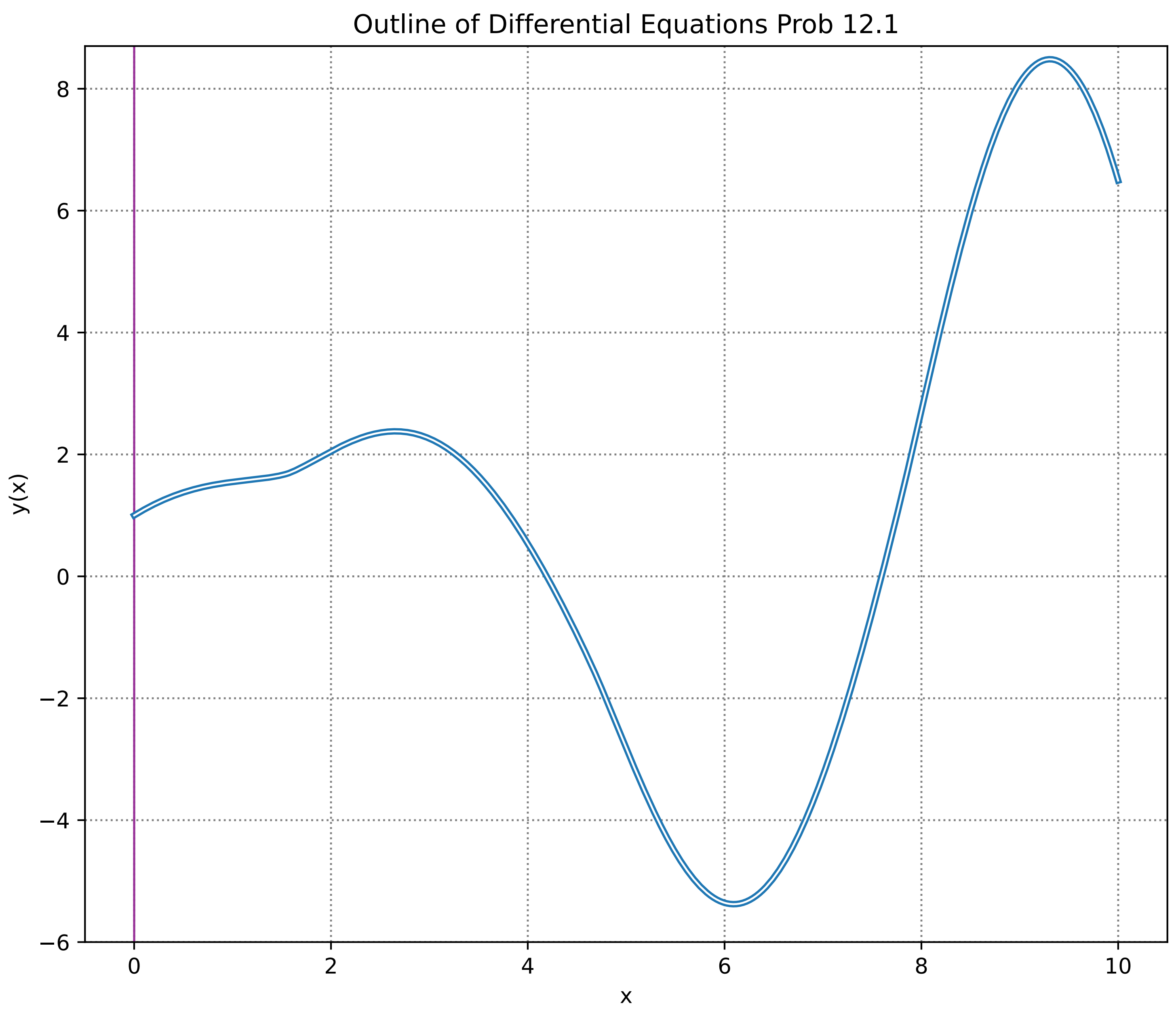
$$y(x) = c_1 \sin(x) + c_2 \cos(x) + c_3 - x\cos(x)$$
$$-\log\left(\cos\left(\frac{x}{2}\right) - \sin\left(\frac{x}{2}\right)\right) + \log\left(\cos\left(\frac{x}{2}\right) + \sin\left(\frac{x}{2}\right)\right) + \sin(x)\log(\cos(x))$$

However, this answer does not match the text answer exactly. The text answer is:

$$y = c_1 + c_2\cos x + c_3 \sin x + \log|\sec x + \tan x| - x\cos x + (\sin x)\log|\cos x|$$

If the Wolfram Alpha answer is modified to be restricted to reals only, and if the text answer is interpreted with $c_1 = 0$, then the two answers are identical, in terms of their plots, as shown below.

In [35]:
```python

import numpy as np
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = np.linspace(0,10,300)
wolfa = np.sin(x) + np.cos(x) - x*np.cos(x) - np.log(abs(np.cos(x/2) - np.sin(x/2)))\
     + np.log(abs(np.cos(x/2) + np.sin(x/2))) + np.sin(x)*np.log(abs(np.cos(x)))
texta = 0 +  np.cos(x) + np.sin(x) + np.log(abs(1/np.cos(x) + np.tan(x))) - x*np.cos(x) + \
     np.sin(x)*np.log(abs(np.cos(x)))


plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y(x)")
plt.title("Outline of Differential Equations Prob 12.1")
plt.rcParams['figure.figsize'] = [9, 7.5]

ax = plt.gca()
#ax.axhline(y=0, color='#993399', linewidth=1)
ax.axvline(x=0, color='#993399', linewidth=1)

#plt.text(965, -2.5e43, "-np.log(abs((np.cos(x/2))**2 - (np.sin(x/2))**2)))
#plt.text(965, 2.25e43, "SOLN: y = 3*np.exp(x**2) + 1/2", size=10,\
        # bbox=dict(boxstyle="square", ec=('#8C564B'),fc=(1., 1., 1),))
plt.ylim(-6,8.7)
#plt.plot(x, y1, linewidth = 0.9)
plt.plot(x, wolfa, linewidth = 3)
plt.plot(x, texta, linewidth = 0.9, color = 'w')
plt.show()

```



Python is put on the case for checking the validity of the solutions. If asked to check an equation decorated with the *abs* function, *checkodesol* seems to go into an infinite think. So that is left out.

First up for a check is the Wolfram Alpha soln.

In [17]:
```python
import sympy as sp
from sympy import *
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']



```

In [18]:
```python
x = symbols("x")
```

```
y = Function("y")(x)
y
```

Out[18]: $y(x)$

In [19]:
```
ode = Eq(y.diff(x,x,x) + y.diff(x), sp.sec(x))
ode
```

Out[19]:
$$\frac{d}{dx}y(x) + \frac{d^3}{dx^3}y(x) = \sec(x)$$

In [20]:
```
sol = sp.sin(x) + sp.cos(x) + 1 - x*sp.cos(x) - sp.log(cos(x/2) - sp.sin(x/2)) + \
sp.log(sp.sin(x/2) +sp.cos(x/2)) + sp.sin(x)*sp.log(cos(x))
sol
```

Out[20]:
$$-x\cos(x) - \log\left(-\sin\left(\frac{x}{2}\right) + \cos\left(\frac{x}{2}\right)\right) + \log\left(\sin\left(\frac{x}{2}\right) + \cos\left(\frac{x}{2}\right)\right) + \log(\cos(x))\sin(x) + \sin(x) + \cos(x) + 1$$

In [21]:
```
checkodesol(ode,sol)
```

Out[21]:
```
(True, 0)
```

Above: The Wolfram Alfa solution is found to be valid.

Below: The text solution can also be checked.

In [22]:
```
ode = Eq(y.diff(x,x,x) + y.diff(x), sp.sec(x))
ode
```

Out[22]:
$$\frac{d}{dx}y(x) + \frac{d^3}{dx^3}y(x) = \sec(x)$$

In [23]:
```
sol2 = 0 + sp.cos(x) + sp.sin(x) + sp.log(sp.sec(x) + sp.tan(x)) - x*sp.cos(x) + \
sp.sin(x)*sp.log(sp.cos(x))
sol2
```

Out[23]:
$$-x\cos(x) + \log(\tan(x) + \sec(x)) + \log(\cos(x))\sin(x) + \sin(x) + \cos(x)$$

In [9]:
```
checkodesol(ode, sol2)
```

Out[9]:
```
(True, 0)
```

12.2 Solve $y''' - 3y'' + 2y' = \frac{e^x}{1+e^{-x}}$

The entry is made into Wolfram Alpha:

!| y''' - 3* y" + 2 * y' = e^x/(1 + e^(-x)) |!

and the answer is received:

$$y(x) = c_1\frac{e^x}{2} + \frac{1}{2}c_2 e^{2x} + c_3 + \frac{1}{2}e^{2x}x$$
$$-e^x\log(e^x+1) - \frac{1}{2}e^{2x}\log(e^x+1) - \frac{1}{2}\log(e^x+1)$$

The answer shown above does not match that of the text. The answer in the text goes like this:

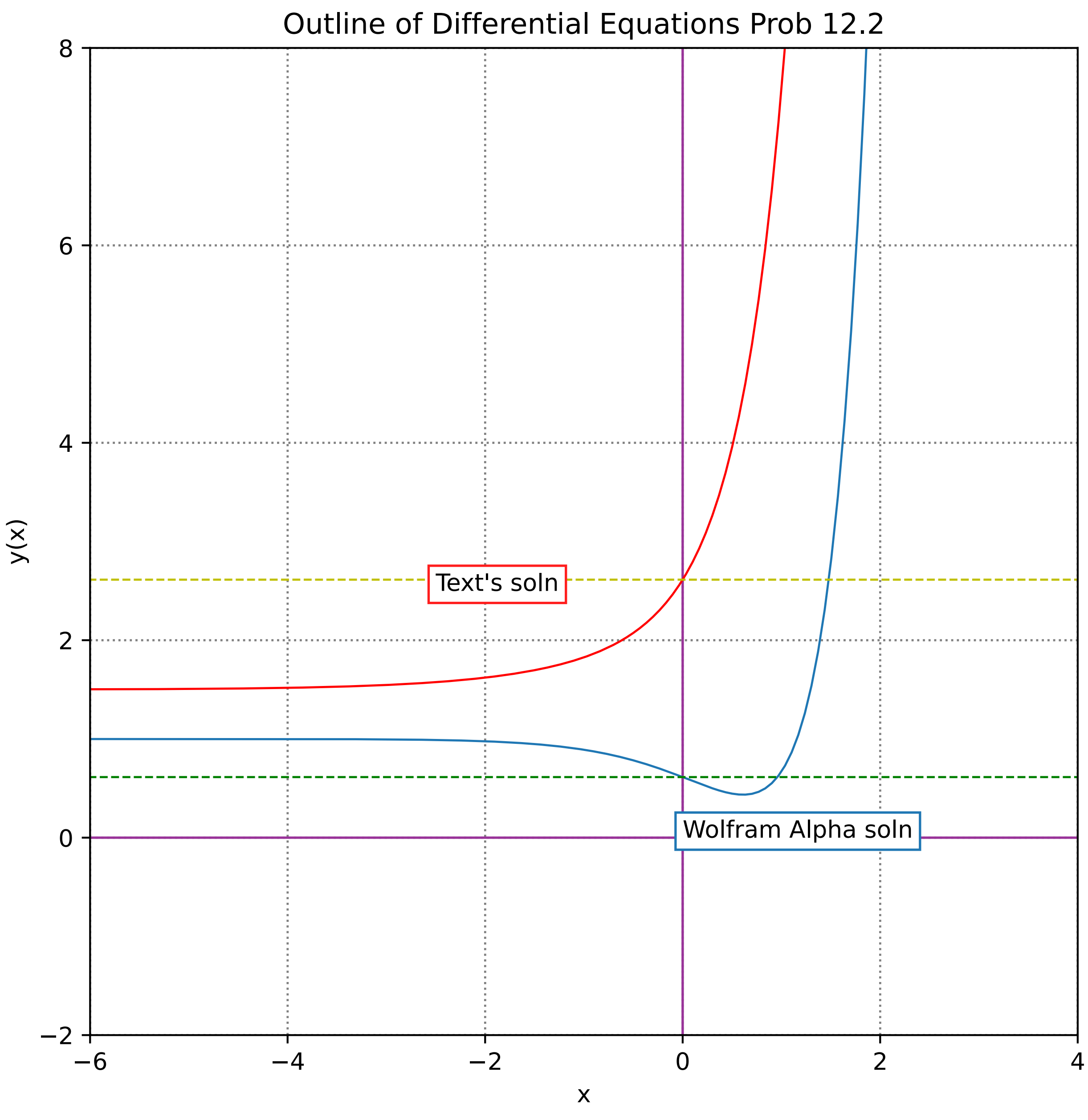
$$y(x) = c_1 + c_2e^x + c_3e^{2x} + \frac{1}{2}(e^x+1) - \frac{1}{2}\log(e^x+1)$$
$$-e^x\log(e^x+1) - \frac{1}{2}e^{2x}\log(1+e^{-x})$$

Both versions of the answer have seven terms. But terms 4 and 6 in the top version do not have analogous terms, with the same basis, in the bottom version. One possible check (though not conclusive), is to compare plots of the functions. All constants are assigned a value of 1 in the test. Based on superficial appearance, there is a real disagreement about this solution.

Read on for the surprising dénouement.

In [24]:

```python
import numpy as np
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = np.linspace(-10,10,300)
y3 = np.exp(x)/2 + (1/2)*np.exp(2*x) +1 + (1/2)*np.exp(2*x)*x \
    -np.exp(x)*np.log(np.exp(x)+1) -(1/2)*np.exp(2*x)*np.log(np.exp(x) \
    + 1) -(1/2)*np.log(np.exp(x) + 1)
y4 = 1 + np.exp(x) + np.exp(2*x) + (1/2)*(np.exp(x) \
    + 1) - (1/2)*(np.log(np.exp(x) + 1)) \
    - np.exp(x)*np.log(np.exp(x) + 1) - \
    (1/2)*np.exp(2 * x)*np.log(1 + np.exp(-x))
@np.vectorize
def flatWA(x):
    return 0.61370563888

@np.vectorize
def flatTx(x):
    return 2.61370563888

plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y(x)")
plt.title("Outline of Differential Equations Prob 12.2")
plt.rcParams['figure.figsize'] = [9, 7.5]


ax = plt.gca()
ax.axhline(y=0, color='#993399', linewidth=1)
ax.axvline(x=0, color='#993399', linewidth=1)
ratio = 1.0
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)


plt.text(0, 0, "Wolfram Alpha soln", size=10,\
         bbox=dict(boxstyle="square", ec=('#1F77B4'),fc=(1., 1., 1),))
plt.text(-2.5, 2.5, "Text's soln", size=10,\
         bbox=dict(boxstyle="square", ec=('#FF1D1D'),fc=(1., 1., 1),))
plt.ylim(-2,8)
plt.xlim(-6,4)
plt.plot(x, y3, linewidth = 0.9)
plt.plot(x, y4, linewidth = 0.9, color = 'r')
plt.plot(x, flatWA(x), linewidth = 0.9, color = 'g', linestyle = 'dashed')
plt.plot(x, flatTx(x), linewidth = 0.9, color = 'y', linestyle = 'dashed')
plt.show()

```



In [25]:
```python
import sympy as sp
from sympy import *
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']
```

In [26]:
```python
x = symbols("x")
y = Function("y")(x)
y
```

Out[26]: $y(x)$

In [27]:
```python
ode = Eq(y.diff(x,x,x) -3*y.diff(x,x)+ 2*y.diff(x), sp.exp(x)/(1+sp.exp(-x)))
ode
```

Out[27]:
$$2\frac{d}{dx}y(x) - 3\frac{d^2}{dx^2}y(x) + \frac{d^3}{dx^3}y(x) = \frac{e^x}{1+e^{-x}}$$

In [28]:
```python
sol = sp.exp(x)/2 + (1/2)*sp.exp(2*x) + 1 + (1/2)*sp.exp(2*x)*x - sp.exp(x)*sp.log(sp.exp(x)+1) -
(1/2)*sp.exp(2*x)*sp.log(sp.exp(x)+1) - (1/2)*sp.log(sp.exp(x)+1)
sol
```

Out[28]:
$$0.5xe^{2x} - 0.5e^{2x}\log\left(e^x+1\right) + 0.5e^{2x} - e^x\log\left(e^x+1\right) + \frac{e^x}{2} - 0.5\log\left(e^x+1\right) + 1$$

In [29]:
```python
checkodesol(ode,sol)
```

Out[29]:
```
(True, 0)
```

Above: Python can verify that the W|A solution works.

In [30]:
```python
sol2 = 1 + sp.exp(x) + sp.exp(2*x) + (1/2)*(sp.exp(x) +1) - (1/2)*sp.log(sp.exp(x)+1) -\
sp.exp(x)*sp.log(sp.exp(x)+1) - (1/2)*sp.exp(2*x)*sp.log(1 + sp.exp(-x))
sol2
```

Out[30]:
$$-0.5e^{2x}\log\left(1+e^{-x}\right) + e^{2x} - e^x\log\left(e^x+1\right) + 1.5e^x - 0.5\log\left(e^x+1\right) + 1.5$$

In [31]:
```python
checkodesol(ode,sol2)
```

Out[31]:
```
(True, 0)
```

Above: Unexpected! Python verifies that the text solution *also* works.

12.3 Solve $y'' - 2y' + y' = \frac{e^x}{x}$

The entry is made into Wolfram Alpha:

!| y" - 2 * y' + y = e^x/x |!

and the answer is received:

$$y(x) = c_1 e^x + c_2 e^x + e^x x \log(x)$$

The answer shown above matches that of the text.

12.4 Solve $y'' - 2y' - 2y = e^{3x}$

The entry is made into Wolfram Alpha:

!| y'' - y' - 2 * y = e^(3 * x) |!

and the answer is received:

$$y(x) = c_1 e^{-x} + c_2 e^{2x} + \frac{e^{3x}}{4}$$

The answer shown above matches that of the text.

12.5 Solve $\ddot{x} + 4x = \sin^2 2t$

The entry is made into Wolfram Alpha:

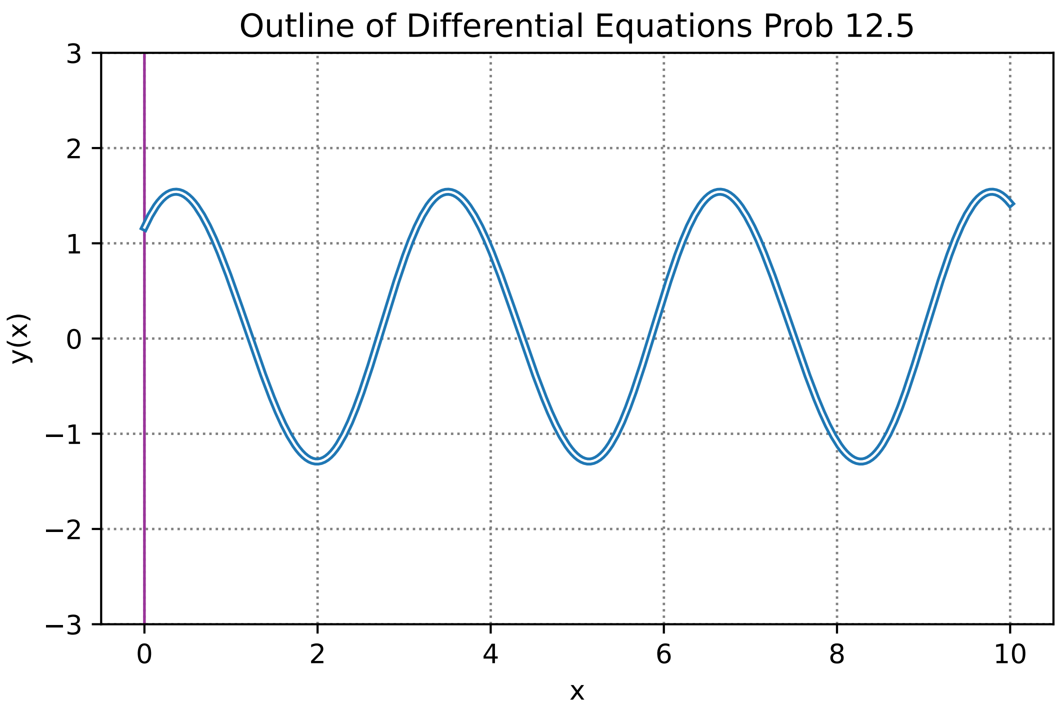
!| x'' + 4 * x = sin^2(2 * t) |!

and the answer is received:

$$x(t) = c_2 \sin(2t) + c_1 \cos(2t) + \frac{1}{24}\cos(4t) + \frac{1}{8}$$

The answer does not match the text exactly, and analyzing the condition of equality seems arduous. A plot gives an informal nod of approval. All constants are assumed to equal 1.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = np.linspace(0,10,300)
y3 = np.sin(2*x) + np.cos(2*x) + 1/24*(np.cos(4*x)) + 1/8
y4 = np.cos(2*x) + np.sin(2*x) + (1/6)*(np.cos(2*x) * np.cos(2*x)) \
    + (1/12)*(np.sin(2*x) * np.sin(2*x))


plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y(x)")
plt.title("Outline of Differential Equations Prob 12.5")
plt.rcParams['figure.figsize'] = [9, 7.5]


ax = plt.gca()
#ax.axhline(y=0, color='#993399', linewidth=1)
ax.axvline(x=0, color='#993399', linewidth=1)
ratio = 1.1
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)


#plt.text(965, -2.5e43, "-np.log(abs((np.cos(x/2))**2 - (np.sin(x/2))**2)))
#plt.text(965, 2.25e43, "SOLN: y = 3*np.exp(x**2) + 1/2", size=10,\
        # bbox=dict(boxstyle="square", ec=('#8C564B'),fc=(1., 1., 1),))
plt.ylim(-3,3)
plt.plot(x, y3, linewidth = 3)
plt.plot(x, y4, linewidth = 0.9, color = 'w')
plt.show()

```



12.6 Solve $t^2 \dfrac{d^2N}{dt^2} - 2t\dfrac{dN}{dt} + 2N = t\log t$

The entry is made into Wolfram Alpha:

!| t^2 * d^2N/dt^2 - (2 * t) * dN/dt + 2 * N = t * log(t) |!

and the answer is received:

$$x(t) = c_2\, t^2 + c_1\, t - \frac{1}{2}\, t\log^2(t) - t\log t$$

The answer matches the text.

12.7 Solve $y' + \dfrac{4}{x}\, y = x^4$

The entry is made into Wolfram Alpha:

!| y' + 4/x * y = x^4 |!

and the answer is received:

$$y(x) = \frac{c_1}{x^4} + \frac{x^5}{9}$$

The answer matches the text.

12.8 Solve $y^{(4)} = 5x$

The entry is made into Wolfram Alpha:

!| y'''' = 5 * x |!

and the answer is received:

$$y(x) = c_4\, x^3 + c_3\, x^2 + c_2\, x + c_1 + \frac{x^5}{24}$$

The answer matches the text.

```
In [ ]:
```