In [1]:
```
%autosave 0
```

```
Autosave disabled
```

Chapter 28: Series Solutions Near a Regular Singular Point. Finding and expressing differential equation solutions in terms of power series.

Note: When transcribing text for entry into Wolfram Alpha the "fenceposts" !| ... |! are excluded from the text.

28.1 Determine whether $x = 0$ is a regular singular point of the differential equation $y'' - xy' + 2y = 0$.

This problem can be submitted to Wolfram Alpha. The entry line is:

!| y" - x y' + 2 y = 0 |!

and the returned solutions is:

$$y(x) = c_2 \left(\sqrt{2\pi}\,(x^2 - 1)\,\text{erfi}\left(\frac{x}{\sqrt{2}}\right) - 2e^{x^2/2}\,x \right) + c_1\,(x^2 - 1)$$

The actual question posed by the problem is not addressed.

28.2 Determine whether $x = 0$ is a regular singular point of the differential equation

$$2x^2y'' + 7x(x + 1)y' - 3y = 0.$$

This problem can be submitted to Wolfram Alpha. The entry line is:

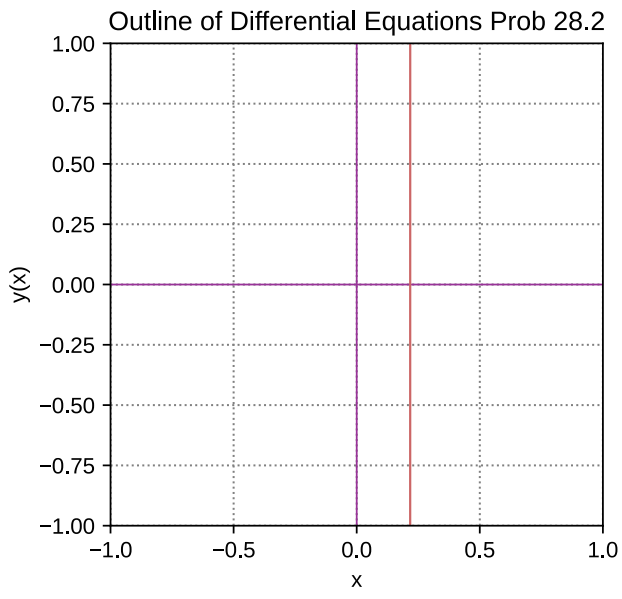
!| 2 x^2 y" + 7 x (x + 1) y' - 3 y = 0 |!

and the returned solutions is:

$$y(x) = \frac{1}{x^3}\,c_2\,e^{-(7x)/2} \left(\sqrt{14\pi}\,e^{(7x)/2}\,(343\,x^3 - 147\,x^2 + 63\,x - 15)\,\text{erf}\left(\sqrt{\frac{7}{2}}\,\sqrt{x}\right) + 98\,(7x - 4)\,x^{3/2} + 210\,\sqrt{x}\,\right) + \frac{c_1\,(7x\,(7x\,(7x - 3) + 9) - 15)}{x^3}$$

The actual question posed by the problem is not addressed.

In [30]:
```
import numpy as np
from scipy import special
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = np.setdiff1d(np.linspace(0.,10.,300),[0]) #to remove the zero
y1 = (1/x**3)*np.exp(-(7*x)/2)*(np.sqrt(14*np.pi)*np.exp((7*x)/2)*\
      (343*x**3 - 147*x**2 + 63*x - 15)*special.erf\
      (np.sqrt(7/2)*np.sqrt(x)) + 98*(7*x - 4)*x**(3/2) +
      210*np.sqrt(x)) + ((7*x*(7*x)*(7*x - 3) + 9) - 15)/x**3


plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y(x)")
plt.title("Outline of Differential Equations Prob 28.2")
plt.rcParams['figure.figsize'] = [4, 4]


ax = plt.gca()
ax.axhline(y=0, color='#993399', linewidth=0.8)
ax.axvline(x=0, color='#993399', linewidth=0.8)
ratio = 0.98
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(-1,1)
plt.xlim(-1, 1)
plt.plot(x, y1, linewidth = 1, color = '#CC6666')
plt.show()
```



Above is shown the exceedingly boring plot of the problem equation solution, as found by Wolfram Alpha, and it appears to have little if any reference to the $x = 0$ area.

28.3 Determine whether $x = 0$ is a regular singular point of the differential equation

$$x^3y'' + 2x^2y' + y = 0.$$

With this equation Wolfram Alpha is able to generate a solution. The entry line is:

!| x^3 y'' + 2 x^2 y' + y = 0 |!

and the result is:

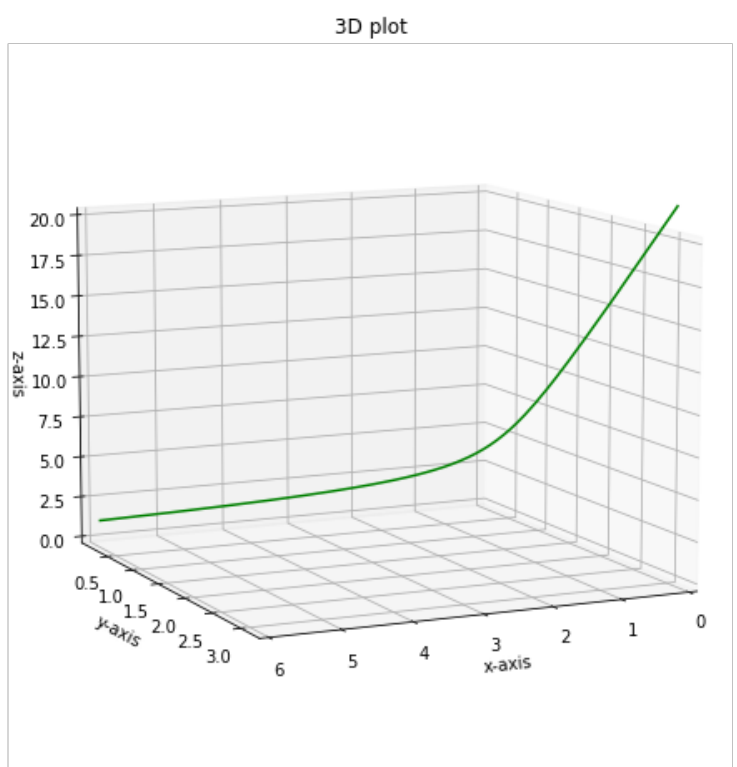
$$c_1 \sqrt{\frac{1}{x}}\, J_1\left(2\sqrt{\frac{1}{x}}\right) + 2\,i\,c_2 \sqrt{\frac{1}{x}}\, Y_1\left(2\sqrt{\frac{1}{x}}\right)$$

Wolfram Alpha explains that $J_n(z)$ is the Bessel function of the first kind, and $Y_n(x)$ is the Bessel function of the second kind. Also lurking in the expression above is the value i .

Looking at the equation for the solution, from Wolfram Alpha, it is seen that not only are both instances of the Bessels of first kind, they are also of first order, because there is nothing higher than first order in the expression which they inhabit. The next objective will be to create a 3D plot, so that the imaginary part of the equation can be displayed on a separate axis.

Note about 3D plotting in Python. The matplotlib module has a number of methods for making 3D plots, some of which include interactivity. However, they are supported by back ends, such as those based on QT, GTK, and Cairo; but choosing the right back end and then getting all parts to function smoothly without detrimental effects on other sections of the Jupyter file or on the platform ecosystem can be hit or miss. A relatively low level 3D experience is included in this chapter, in which view perspective can be controlled, but not through mouse gestures.

In [10]:
```
from mpl_toolkits import mplot3d
import sympy as sp
import numpy as np
from scipy import special
import matplotlib.pyplot as plt

x, y, z = sp.symbols('x y z')

fig = plt.figure(figsize = (8,8))
ax = plt.axes(projection = '3d')
ax.set_title("3D plot",pad=6)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_xlim(0,6)
ax.set_zlim(0,20)

x = np.setdiff1d(np.linspace(0.,6.,300),[0]) #to remove the zero
y = np.sqrt(1/x)*(special.j1(1,2*np.sqrt(1/x)))
#y = x**2
z =abs( 2*2*np.sqrt(1/x)*special.y1(1,2*np.sqrt(1/x)))
ax.plot3D(x, y, z, 'green')
plt.xticks([0, 1, 2, 3, 4, 5, 6],
           ["0","1", "2", "3", "4","5", "6"])

#view x-z plane
#ax.view_init(0, 90)

#view y-x plane
#ax.view_init(90, 0)

#view y-z plane
#ax.view_init(0, 0)

#view from (50%-x, 100%-z) to (50%-x, 0%-z)
#ax.view_init(45, 90)

#view from (1, 1, 1) to (0, 0, 0)
#ax.view_init(45, 45)

#view x-y plane
#ax.view_init(45, 90)

ax.view_init(10, 65)

plt.show()
```



Above is shown the 3D plot of the Wolfram Alpha solution equation, with y-axis dedicated to the real half of the equation, and the z-axis dedicated to the complex half of the equation. Changing the view_init lines changes the perspective from which the model is viewed.

28.4 Determine whether $x = 0$ is a regular singular point of the differential equation

$$8x^2y'' + 10xy' + (x - 1)y = 0\,.$$

In [11]:
```
from sympy import *
import numpy as np

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

Out[11]: $y(x)$

In [12]:
```
y.diff()
```

Out[12]: $\frac{d}{dx}y(x)$

In [13]:
```
ode = Eq(8*x**2*y.diff(x,x) + 10*x*y.diff(x) + (x - 1)*y , 0)
ode
```

Out[13]: $8x^2\frac{d^2}{dx^2}y(x) + 10x\frac{d}{dx}y(x) + (x-1)\,y(x) = 0$

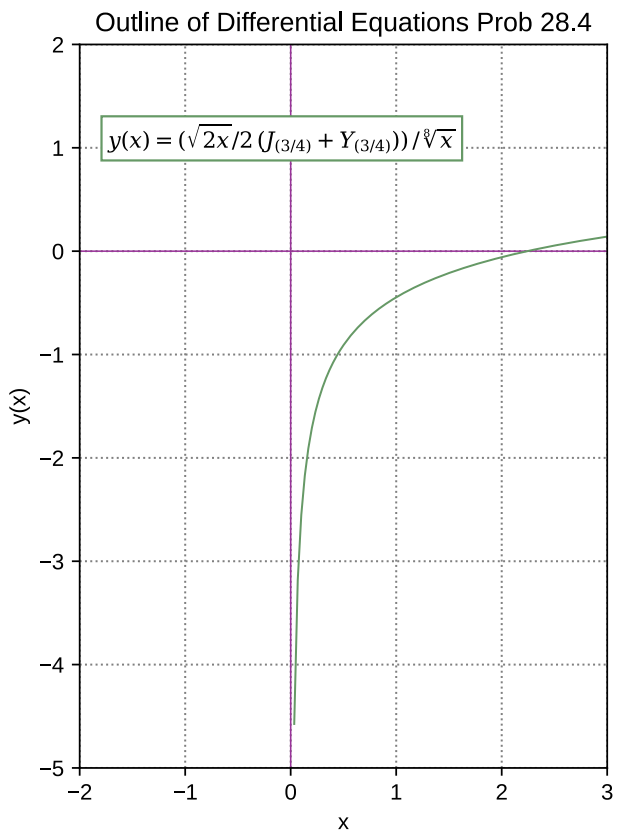
In [14]:
```
sol = dsolve(ode, y)
sol
```

Out[14]:

$$y(x) = \frac{C_1 J_{\frac{3}{4}}\left(\frac{\sqrt{2}\sqrt{x}}{2}\right) + C_2 Y_{\frac{3}{4}}\left(\frac{\sqrt{2}\sqrt{x}}{2}\right)}{\sqrt[8]{x}}$$

Above is shown the sympy solution of the problem. It is somewhat less complicated than the Wolfram Alpha solution. Neither one shows an imaginary element, which makes the solution easier to plot. The order of the Bessel factors is seen to be fractional, not integer. Although some scipy kinds of bessel species are Integer, the most common of the J and Y types are Real, which is fortunate. Since no initial conditions are available, the value of both constants will be set to 1.

In [69]:
```
import numpy as np
import scipy.special as spe
import sympy as sp
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = np.linspace(0,10,300)
y = (spe.jv(0.75,(np.sqrt(2*x)/2)) + spe.yv(0.75, (np.sqrt(2*x)/2))) /x**(1/8)

plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y(x)")
plt.title("Outline of Differential Equations Prob 28.4")
plt.rcParams['figure.figsize'] = [5, 6]
plt.rcParams['font.sans-serif'] = ['Liberation Sans']
plt.rcParams['mathtext.fontset'] = 'dejavuserif'

ax = plt.gca()
ax.axhline(y=0, color='#993399', linewidth=0.7)
ax.axvline(x=0, color='#993399', linewidth=0.7)
ratio = 0.98
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)

plt.text(-1.73, 1, "$y(x) = (\sqrt{2x}/2\,(J_{(3/4)}+Y_{(3/4)}))\,/\,\sqrt[8]{x}$",\
         size=10,bbox=dict(boxstyle="square",ec=('#669966'),fc=(1., 1., 1),))
plt.ylim(-5, 2)
plt.xlim(-2, 3)
plt.plot(x, y, linewidth = 0.9, color = '#669966')
plt.show()
```



Unlike the pure Bessel function, the solution equation does not show a propensity to recross the x-axis.

28.9 Find the general solution near $x = 0$ of $3x^2y'' - xy' + y = 0$.

This equation can be entered into Wolfram Alpha, and the entry line is:

!| 3 x^2 y'' - x y' + y = 0 |!

and the result returned is:

$$y(x) = c_1 \sqrt[3]{x} + c_2 x$$

28.10 Find the general solution near $x = 0$ of $x^2y'' + xy' + x^2y = 0$.

This equation can be entered into Wolfram Alpha, and the entry line is:

!| x^2 y'' + x y' + x^2 y = 0 |!

and the result returned is:

$$y(x) = c_1 J_0(x) + c_2 Y_0(x)$$

In addition to the answer, Wolfram Alpha gives sample plots complete with initial conditions. These could be useful to check the proper application of Bessel calculations to sample plots. The first set of initial conditions posits $y(1) = 1, \quad y'(1) = 0.$

The constants for the solution equation itself can be found using Maxima:

```
(%i1) bessel_j(0,1);
(%o1) bessel_j(0,1)
(%i2) float(%);
(%o2) 0.7651976865579666
(%i3) bessel_y(0,1);
(%o3) bessel_y(0,1)
(%i4) float(%);
(%o4) 0.08825696421567691
```

Which results in the equation:

$$c_1(0.765197686557) + c_2(0.088256964215) = 1$$

```
(%i17) bessel_j(1,1);
(%o17) bessel_j(1,1)
(%i18) float(%);
(%o18) 0.4400505857449335
(%i19) bessel_y(1,1);
(%o19) bessel_y(1,1)
(%i20) float(%);
(%o20) -0.7812128213002888
```

And results in another equation:

$$c_1(-0.440050585744) + c_2(0.781212821300) = 0$$

and brings circumstances to a favorable position regarding a sympy simultaneous solution.

It turns out that Wolfram Alpha can differentiate the Bessels. So

!| D[j0(x)] |!

gives the derivative for J, which is

$$\frac{1}{2}(J_{-1}(x) - \frac{J_0(x) + x\,J_1(x)}{x})$$

or, knowing that $J_0(x) = 1$ when $x = 1$

$$\frac{1}{2}(J_{-1}(1) - 1 - J_1(1))$$

Leaving this line of deduction alone for awhile, and looking at Y,

!| D[Y_0(x)] |!

gives the derivative for Y, which is

$$-Y_1(x)$$

or in this case, $Y_1(1)$. There would now be a need to go back to Maxima to get one more current value, except for the fact that, in Maxima, bessel_y(-1, 1) = −bessel_y(1, 1) , which has already been retrieved. Likewise, Maxima informs that bessel_j(-1, 1) = −bessel_j(1, 1) .

Assembling,

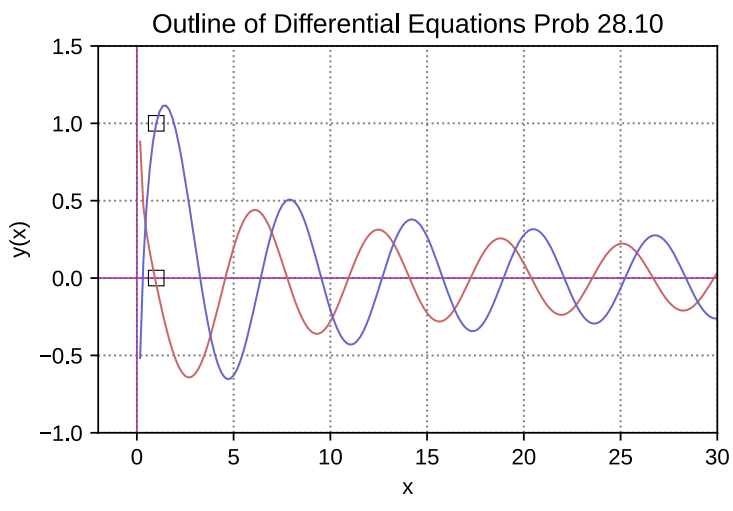
$$c1\,(\frac{1}{2}(-0.44005058) - (1) - (0.44005058)) + c2\,(-(-0.078121282)) = 0$$

$$\implies c1\,(0.5)\,(-1.880101171489867) + c2\,(0.07812128213002888) = 0$$

In [24]:
```
import sympy as sp

c1, c2 = sp.symbols('c1 c2')

eq1 = sp.Eq( c1*(0.765197686557) + c2*(0.088256964215), 1)
eq2 = sp.Eq(c1*(0.5)*(-1.880101171489867) + c2*(0.781212821300), 0)
res = sp.solve([eq1, eq2], (c1, c2))
res
```

Out[24]:
```
{c1: 1.14757961723141, c2: 1.38090781660771}
```

In [49]:
```
import numpy as np
import scipy.special as spe
import sympy as sp
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = np.setdiff1d(np.linspace(0.,50.,300),[0]) #to remove the zero
y01 = (1.14757961723141*spe.jv(0,x) + 1.38090781660771*spe.yv(0, x))
y4 = 0.5*(spe.jv(-1,x) - (spe.jv(0,x) + x*spe.jv(1,x))/\
      x) +  spe.yv(-1, 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 28.10")
plt.rcParams['figure.figsize'] = [6, 5]
plt.rcParams['font.sans-serif'] = ['Liberation Sans']
plt.rcParams['mathtext.fontset'] = 'dejavuserif'

ax = plt.gca()
ax.axhline(y=0, color='#993399', linewidth=0.7)
ax.axvline(x=0, color='#993399', linewidth=0.7)
ratio = 8
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)

#plt.text(15, 1, "$y(x) = (\sqrt{2x}/2\,(J_{(3/4)}+Y_{(3/4)}))\,/\,\sqrt[8]{x}$",\
         #size=10,bbox=dict(boxstyle="square",ec=('#669966'),fc=(1., 1., 1),))
xpts = np.array([1, 1])
ypts = np.array([1, 0])
plt.plot(xpts, ypts, markersize=7, color='k', marker='s', \
         mfc='none', linestyle = 'none', markeredgewidth=0.5)
plt.ylim(-1, 1.5)
plt.xlim(-2, 30)
plt.plot(x, y4, linewidth = 0.9, color = '#CC6666')
plt.plot(x, y01, linewidth = 0.9, color = '#6666CC')
plt.show()
```



It appears that the derivative function missed its mark slightly, maybe by a 4% lag in x.

28.12 Use the method of Frobenius to find one solution near $x = 0$ of $x^2 y'' - x y' + y = 0$.

This equation can be solved by Wolfram Alpha. The entry line is:

!| x^2 y'' - x y' + y = 0 |!

and the returned answer is:

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

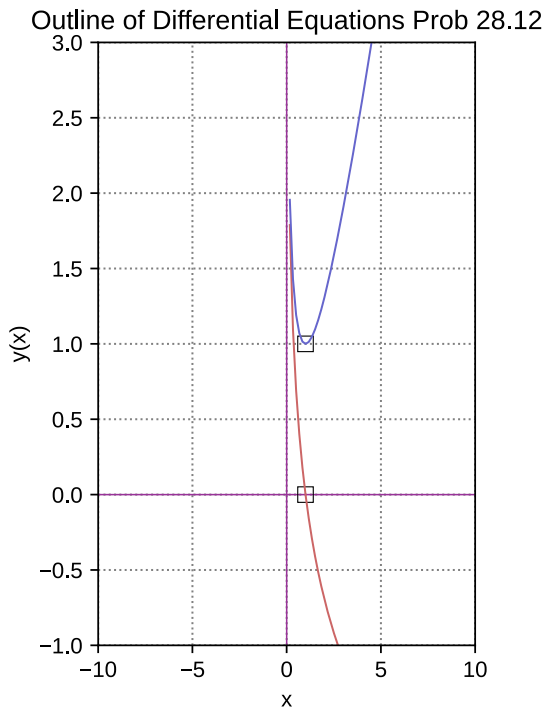
As a possible IVP, Wolfram Alpha suggests the initial conditions: $y(1) = 1,\ y'(1) = 0$.

The derivative of the solution is:

$$y'(x) = c_1 + c_2 \log(x) + c_2$$

As for the constants, from the solution equation it can be seen that $c_1 = 1$ is the logical choice, whereas c_2 can be anything. From the derivative equation, $c_1 + c_2 = 0$, which implies that c_2 equals -1.

In [63]:
```
import numpy as np
import scipy.special as spe
import sympy as sp
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = np.setdiff1d(np.linspace(0.,50.,300),[0]) #to remove the zero
y01 = x - np.log(x)
y4 = 1 - np.log(x) -1

plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y(x)")
plt.title("Outline of Differential Equations Prob 28.12")
plt.rcParams['figure.figsize'] = [6, 5]
plt.rcParams['font.sans-serif'] = ['Liberation Sans']
plt.rcParams['mathtext.fontset'] = 'dejavuserif'

ax = plt.gca()
ax.axhline(y=0, color='#993399', linewidth=0.7)
ax.axvline(x=0, color='#993399', linewidth=0.7)
ratio = 8
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)

#plt.text(15, 1, "$y(x) = (\sqrt{2x}/2\,(J_{(3/4)}+Y_{(3/4)}))\,/\,\sqrt[8]{x}$",\
         #size=10,bbox=dict(boxstyle="square",ec=('#669966'),fc=(1., 1., 1),))
xpts = np.array([1, 1])
ypts = np.array([1, 0])
plt.plot(xpts, ypts, markersize=7, color='k', marker='s', \
         mfc='none', linestyle = 'none', markeredgewidth=0.5)
plt.ylim(-1, 3)
plt.xlim(-10, 10)
plt.plot(x, y4, linewidth = 0.9, color = '#CC6666')
plt.plot(x, y01, linewidth = 0.9, color = '#6666CC')
plt.show()
```



28.14 Use the method of Frobenius to find one solution near $x = 0$ of $x^2 y'' + (x^2 - 2x) y' + 2y = 0$.

This equation can be entered into Wolfram Alpha. The entry line is:

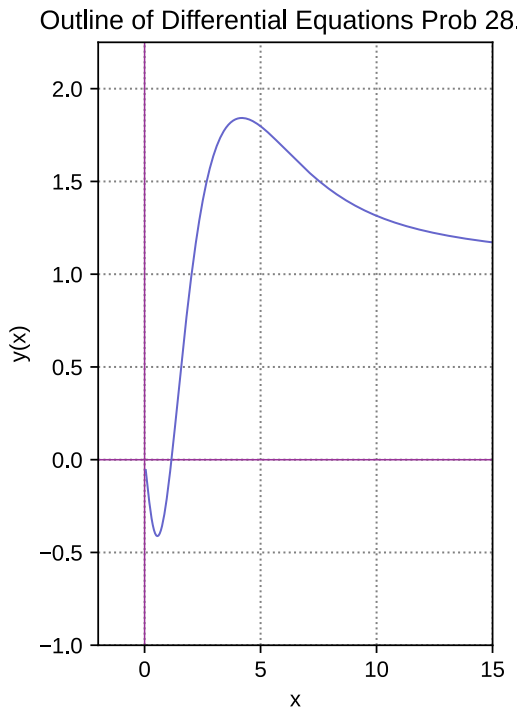
!| x^2 y'' + (x^2 - 2 x) y' + 2 y = 0 |!

and the returned answer is:

$$y(x) = c_1 e^{-x} x^2 \left(\mathrm{Ei}(x) - \frac{e^x}{x}\right) + c_2 e^{-x} x^2$$

Ei, or the exponential function, is included in the scipy special functions section. Therefore the equation should be plottable.

In [79]:
```
import numpy as np
import scipy.special as spe
import sympy as sp
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = np.setdiff1d(np.linspace(0.,15.,300),[0]) #to remove the zero
y = np.exp(-x)*x**2*(spe.expi(x) - (np.exp(x)/x) + np.exp(-x)*x**2)

plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y(x)")
plt.title("Outline of Differential Equations Prob 28.14")
plt.rcParams['figure.figsize'] = [6, 5]
plt.rcParams['font.sans-serif'] = ['Liberation Sans']
plt.rcParams['mathtext.fontset'] = 'dejavuserif'

ax = plt.gca()
ax.axhline(y=0, color='#993399', linewidth=0.7)
ax.axvline(x=0, color='#993399', linewidth=0.7)
ratio = 8
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)

#plt.text(15, 1, "$y(x) = (\sqrt{2x}/2\,(J_{(3/4)}+Y_{(3/4)}))\,/\,\sqrt[8]{x}$",\
         #size=10,bbox=dict(boxstyle="square",ec=('#669966'),fc=(1., 1., 1),))
#xpts = np.array([1, 1])
#ypts = np.array([1, 0])
#plt.plot(xpts, ypts, markersize=7, color='k', marker='s', \
         #mfc='none', linestyle = 'none', markeredgewidth=0.5)
plt.ylim(-1, 2.25)
plt.xlim(-2, 15)
#plt.plot(x, y4, linewidth = 0.9, color = '#CC6666')
plt.plot(x, y, linewidth = 0.9, color = '#6666CC')
plt.show()
```



28.16 Use the method of Frobenius to find one solution near $x = 0$ of $x^2 y'' + xy' + (x^2 - 1) = 0$.

This equation can be entered into Wolfram Alpha. The entry line is:

!| x^2 y'' + x y' + (x^2 - 1) y = 0 |!

and the returned answer is:

$$y(x) = c_1 J_1(x) + c_2 Y_1(x)$$

More Bessels. If Wolfram Alpha's suggestion is taken to use the initial conditions: $y(1) = 1; y'(1) = 0$ then the search for constants begins. It begins with the task of finding the derivative of the solution, with the entry

!| D[c1 J_1(x) + c2 Y_1(x)] |!

W|A needs enough local reference to associate the request with Bessels. The above is adequate. The response is:

$$y'(x) = \frac{1}{2}(c_1 J_0(x) - c_1 J_2(x) + c_2 (Y_0(x) - Y_2(x)))$$

A possible resource is pulling the specific Bessels out of an .svg table like the one below, made in LibreWriter and Inkscape and remarkably economical at 21 kb disk space:

Maxima operand	Value	W|A operand	Value
bessel_j(1,1)	0.4400505857449335	J_1(1)	0.4400505857
bessel_y(1, 1)	-0.7812128213002888	Y_1(1)	-0.781212821300
bessel_j(0,1)	0.7651976865579666	J_0(1)	0.76519768655
bessel_j(2,1)	0.1149034849319005	J_2(1)	0.114903484931
bessel_y(0, 1)	0.08825696421567691	Y_0(1)	0.088256964215
bessel_y(2,1)	-1.650682606816255	Y_2(1)	-1.65068260681

What is available for calculation of c_1 and c_2 from the solution is:

$$c_1 (0.4400505857449335) + c_2 (-0.7812128213002888) = 1$$

and from the derivative expression the following is obtainable:

$$\frac{1}{2}(c_1\, 0.7651976865579666 - c_1 0.1149034849319005 + c_2\, 0.08825696421567691 - c_2(-1.65068$$

or, simplifying:

$$\frac{1}{2} c_1(0.6502942016) + \frac{1}{2} c_2(1.738939571) = 0$$

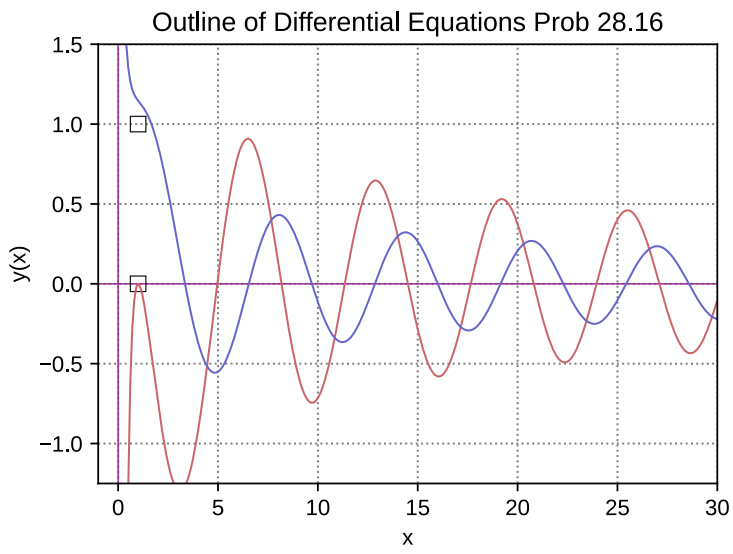
Next, the simultaneous calcuations from sympy:

In [63]:
```
import sympy as sp

c1, c2 = sp.symbols('c1 c2')

eq1 = sp.Eq( c1*(0.4400505857449335) + c2*(-0.7812128213002888), 1)
eq2 = sp.Eq((1/2)*(c1)*(0.6502942016) + (1/2)*(c2)*(1.738939571), 0)
res = sp.solve([eq1, eq2], (c1, c2))
res
```

Out[63]:
```
{c1: 1.36575994535946, c2: -0.510739871618455}
```

And then the plot.

In [67]:
```
import numpy as np
import scipy.special as spe
import sympy as sp
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

c1 = 1.36575994535946
c2 = -0.510739871618455
#c1 = 26.9512 Wolfram Alpha calculated this number
#c2 = -10.0787 Wolfram Alpha calculated this number

x = np.setdiff1d(np.linspace(0.,30.,300),[0]) #to remove the zero
y2 = c1*(spe.jv(0,x)) - c1*(spe.jv(2,x)) + c2*(spe.yv(0,x)) - \
    c2*(spe.yv(2,x))
y = c1*(spe.jv(1,x) + c2*spe.yv(1,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 28.16")
plt.rcParams['figure.figsize'] = [5, 6]
plt.rcParams['font.sans-serif'] = ['Liberation Sans']
plt.rcParams['mathtext.fontset'] = 'dejavuserif'

ax = plt.gca()
ax.axhline(y=0, color='#993399', linewidth=0.7)
ax.axvline(x=0, color='#993399', linewidth=0.7)
ratio = 8
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)

#plt.text(15, 1, "$y(x) = (\sqrt{2x}/2\,(J_{(3/4)}+Y_{(3/4)}))\,/\,\sqrt[8]{x}$",\
         #size=10,bbox=dict(boxstyle="square",ec=('#669966'),fc=(1., 1., 1),))
xpts = np.array([1, 1])
ypts = np.array([1, 0])
plt.plot(xpts, ypts, markersize=7, color='k', marker='s', \
         mfc='none', linestyle = 'none', markeredgewidth=0.5)
plt.ylim(-1.25, 1.5)
plt.xlim(-1, 30)
plt.plot(x, y2, linewidth = 0.9, color = '#CC6666')
plt.plot(x, y, linewidth = 0.9, color = '#6666CC')
plt.show()
```



Something is obviously very wrong with the plot above, but the error has not yet been identified. It appears that the Wolfram Alpha solution, while not impugned, does not repay the difficulties in adding the necessary details.

It is interesting that when Maxima solves this ODE, no Bessel functions are involved. The relevant input and output lines are shown below.

```
(%i34) eq: x^2·'diff(y, x, 2) + x·'diff(y, x) + (x^2 − 1) = 0;
```

$$(\%o34)\quad x^2\left(\frac{d^2}{dx^2}y\right)+x\left(\frac{d}{dx}y\right)+x^2-1=0$$

```
(%i35) sol2: ode2(eq, y, x);
```

$$(\%o35)\quad y=\frac{2\log(x)^2-x^2}{4}+\%k2\,\log(x)+\%k1$$

The function takes on the value 1 when the constant $\%k1$ equals 1.25. Maxima probably could stand in as the main program for this problem. Its solution functions as expected, but as it turns out, sympy has a working solution which is easier to develop. Therefore no more maxima-related details will be given.

In [1]:
```
from sympy import *
import sympy as sp

x, c1, c2 = symbols("x c1 c2")
y = Function("y")(x)
y
```

Out[1]: $y(x)$

In [2]:
```
y.diff()
```

Out[2]: $\frac{d}{dx}y(x)$

In [3]:
```
ode = Eq(x**2*y.diff(x,x) + x*y.diff(x) + (x**2 - 1) , 0)
ode
```

Out[3]: $x^2\frac{d^2}{dx^2}y(x)+x^2+x\frac{d}{dx}y(x)-1=0$

In [4]:
```
sol = dsolve(ode, y)
sol
```

Out[4]: $y(x)=C_1+C_2\log(x)-\frac{x^2}{4}+\frac{\log(x)^2}{2}$

The cell above gives the sympy solution to the problem.

In [6]:
```
expr = c1 + c2*log(x) - x**2/4 + (log(x)**2)/2
exprs = expr.subs(x, 1)
exprs
```

Out[6]: $c_1-\frac{1}{4}$

In [7]:
```
expr2 = sp.diff(expr, x)
expr2
```

Out[7]: $\frac{c_2}{x} - \frac{x}{2} + \frac{\log(x)}{x}$

The cell above gives the derivative of the sympy solution to the problem.

In [9]:
```
expr2.subs(x, 1)
```

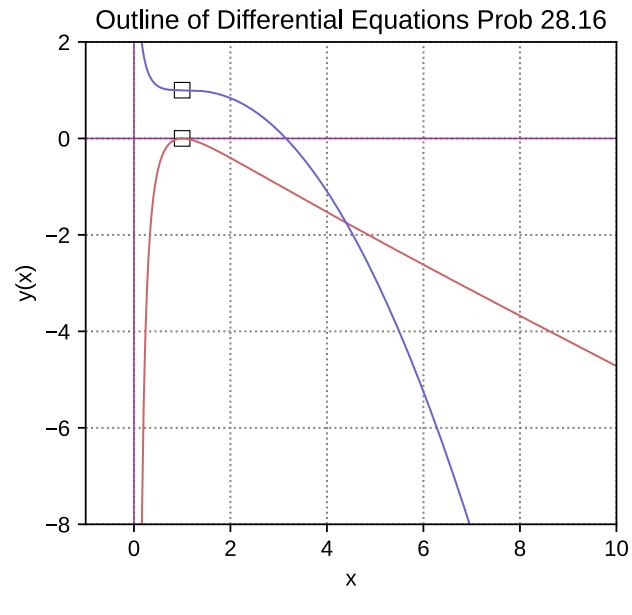
Out[9]: $c_2 - \frac{1}{2}$

In [10]:
```
c1, c2 = sp.symbols('c1 c2')

eq1 = sp.Eq( c1 - 1/4, 1)
eq2 = sp.Eq(c2 - 1/2, 0)
res = sp.solve([eq1, eq2], (c1, c2))
res
```

Out[10]:
```
{c1: 1.25000000000000, c2: 0.500000000000000}
```

The cell above gives the solution to the constant asssignments.

In [11]:
```
import numpy as np
import scipy.special as spe
import sympy as sp
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

c1 = 1.25
c2 = 0.5

x = np.setdiff1d(np.linspace(0.,10.,600),[0]) #to remove the zero
y2 = c2/x - x/2 + np.log(x)/x
y = c1 + c2*np.log(x) - x**2/4 + (np.log(x)**2)/2

plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y(x)")
plt.title("Outline of Differential Equations Prob 28.16")
plt.rcParams['figure.figsize'] = [5, 6]
plt.rcParams['font.sans-serif'] = ['Liberation Sans']
plt.rcParams['mathtext.fontset'] = 'dejavuserif'

ax = plt.gca()
ax.axhline(y=0, color='#993399', linewidth=0.7)
ax.axvline(x=0, color='#993399', linewidth=0.7)
ratio = 1.0
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)

#plt.text(15, 1, "$y(x) = (\sqrt{2x}/2\,(J_{(3/4)}+Y_{(3/4)}))\,/\,\sqrt[8]{x}$",\
         #size=10,bbox=dict(boxstyle="square",ec=('#669966'),fc=(1., 1., 1),))
xpts = np.array([1, 1])
ypts = np.array([1, 0])
plt.plot(xpts, ypts, markersize=7, color='k', marker='s', \
         mfc='none', linestyle = 'none', markeredgewidth=0.5)
plt.ylim(-8, 2)
plt.xlim(-1, 10)
plt.plot(x, y2, linewidth = 0.9, color = '#CC6666')
plt.plot(x, y, linewidth = 0.9, color = '#6666CC')
#plt.plot(x, y3, linewidth = 0.9, color = '#339933')
plt.show()
```



The reliable performance of sympy in this case is gratifying. It shows that a simpler solution can be just a valid as a more more sophisticated looking one.

28.18 Use the method of Frobenius to find one solution near $x = 0$ of $x^2 y'' + (x^2 + 2x)\, y' - 2y = 0$.

Wolfram Alpha can solve this equation. The entry line is:

!| x^2 y'' + (x^2 + 2 x) y' - 2 y = 0 |!

and the answer returned is:

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

28.21 Find the indicial equation of $x^2 y'' + xe^x y' + (x^3 - 1)y = 0$ if the solution is required near $x = 0$.

Wolfram Alpha does **not** solve this equation. It draws a plot for it and offers several alternative forms, but no solution. However, sympy can solve it.

In [83]:
```
from sympy import *
import sympy as sp

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

Out[83]: $y(x)$

In [84]:
```
y.diff()
```

Out[84]:
$$\frac{d}{dx}y(x)$$

In [85]:
```
ode = Eq(x**2*y.diff(x,x) + x*exp(x)*y.diff(x) + (x**3 - 1)*y , 0)
ode
```

Out[85]:
$$x^2\frac{d^2}{dx^2}y(x) + xe^x\frac{d}{dx}y(x) + \left(x^3 - 1\right)y(x) = 0$$

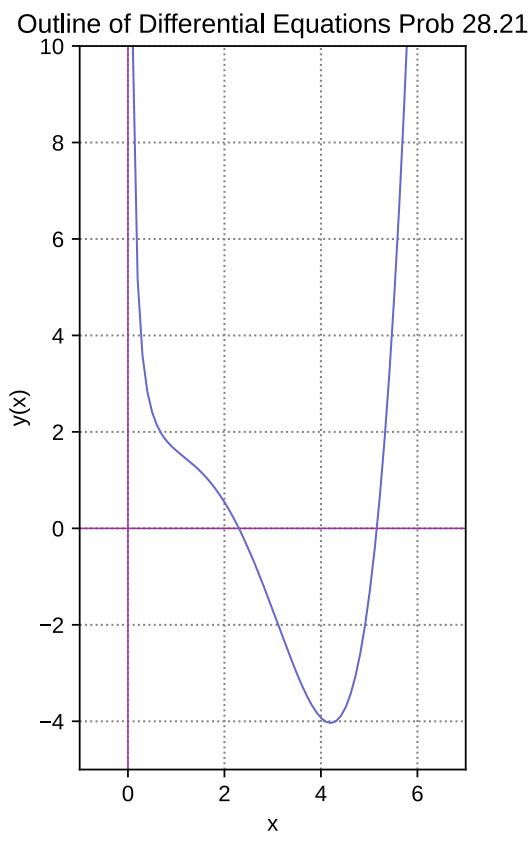
In [86]:
```
sol = dsolve(ode, y)
sol
```

Out[86]:
$$y(x) = C_2 x\left(1 - \frac{x^3}{15}\right) + \frac{C_1\left(\frac{x^6}{72} - \frac{x^3}{3} + 1\right)}{x} + O\left(x^6\right)$$

Notice that it is not necessary to import numpy to accomplish what has been done so far. A solution to the ODE has been provided, although the precision is not perfect, since the series remainder will have to be dropped at some point.

A plot of the solution function is made.

In [97]:
```
import numpy as np
import scipy.special as spe
import sympy as sp
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x = np.setdiff1d(np.linspace(0.,30.,300),[0]) #to remove the zero
y = x*(1 - (x**3/15)) + ((x**6/72) - (x**3/3) + 1)/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 28.21")
plt.rcParams['figure.figsize'] = [5, 6]
plt.rcParams['font.sans-serif'] = ['Liberation Sans']
plt.rcParams['mathtext.fontset'] = 'dejavuserif'

ax = plt.gca()
ax.axhline(y=0, color='#993399', linewidth=0.7)
ax.axvline(x=0, color='#993399', linewidth=0.7)
ratio = 1
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)

#plt.text(15, 1, "$y(x) = (\sqrt{2x}/2\,(J_{(3/4)}+Y_{(3/4)}))\,/\,\sqrt[8]{x}$",\
         #size=10,bbox=dict(boxstyle="square",ec=('#669966'),fc=(1., 1., 1),))
#xpts = np.array([1, 1])
#ypts = np.array([1, 0])
#plt.plot(xpts, ypts, markersize=7, color='k', marker='s', \
        # mfc='none', linestyle = 'none', markeredgewidth=0.5)
plt.ylim(-5, 10)
plt.xlim(-1, 7)
#plt.plot(x, y2, linewidth = 0.9, color = '#CC6666')
plt.plot(x, y, linewidth = 0.9, color = '#6666CC')
#plt.plot(x, y3, linewidth = 0.9, color = '#339933')
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: