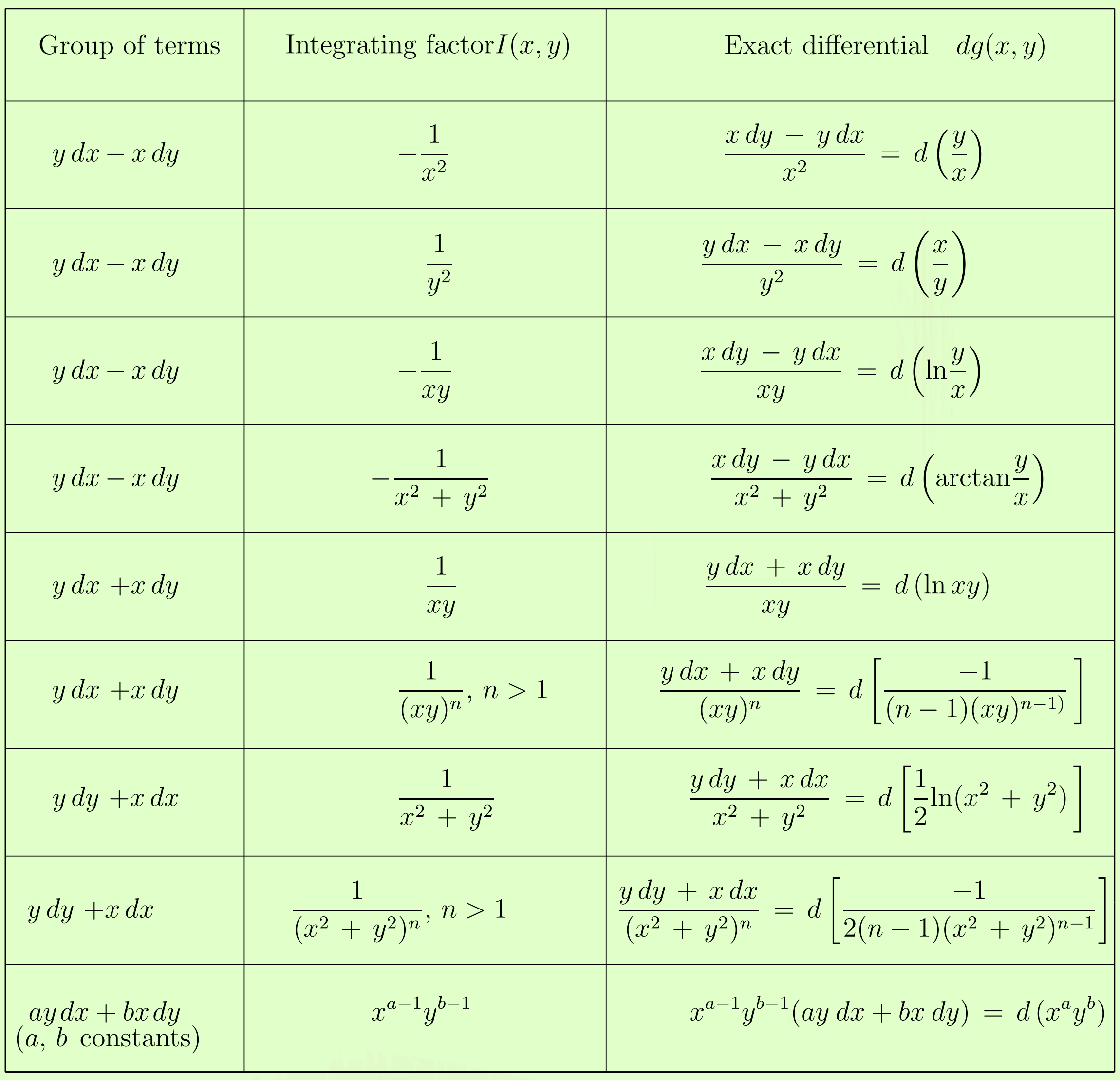
(Custom CSS files are not reliable for controlling Jupyter font style. To establish the same appearance as the original notebook, depend on the browser to control the font, by setting the desired font faces in the browser settings. For example, Chrome 135 or Firefox 134 can do this. In this notebook series, Bookerly font is for markdown and Monaco is for code.)

Chapter 5: Exact First-Order Differential Equations Techniques for solving exact odes are considered. The method for identifying exact equations can be seen in Chapter 3.

In the environment of exactness, a differential equation may exist in the form of $M(x, y)\,dx + N(x, y)\,dy = 0$. This differential equation may have a solution consisting of $g(x, y) = c$. To the single pure constant c some variables may be appended depending on the structure of the equation under examination.

A subset of exact equations qualify for operations with **integrating factors**. These can make the solving of the equation much easier. The general pattern for seeking integrating factors is summarized in the following table.



Group of terms	Integrating factor$I(x, y)$	Exact differential $dg(x, y)$
$y\,dx - x\,dy$	$-\frac{1}{x^2}$	$\frac{x\,dy - y\,dx}{x^2} = d\left(\frac{y}{x}\right)$
$y\,dx - x\,dy$	$\frac{1}{y^2}$	$\frac{y\,dx - x\,dy}{y^2} = d\left(\frac{x}{y}\right)$
$y\,dx - x\,dy$	$-\frac{1}{xy}$	$\frac{x\,dy - y\,dx}{xy} = d\left(\ln\frac{y}{x}\right)$
$y\,dx - x\,dy$	$-\frac{1}{x^2 + y^2}$	$\frac{x\,dy - y\,dx}{x^2 + y^2} = d\left(\arctan\frac{y}{x}\right)$
$y\,dx + x\,dy$	$\frac{1}{xy}$	$\frac{y\,dx + x\,dy}{xy} = d\,(\ln xy)$
$y\,dx + x\,dy$	$\frac{1}{(xy)^n},\ n > 1$	$\frac{y\,dx + x\,dy}{(xy)^n} = d\left[\frac{-1}{(n-1)(xy)^{n-1)}}\right]$
$y\,dy + x\,dx$	$\frac{1}{x^2 + y^2}$	$\frac{y\,dy + x\,dx}{x^2 + y^2} = d\left[\frac{1}{2}\ln(x^2 + y^2)\right]$
$y\,dy + x\,dx$	$\frac{1}{(x^2 + y^2)^n},\ n > 1$	$\frac{y\,dy + x\,dx}{(x^2 + y^2)^n} = d\left[\frac{-1}{2(n-1)(x^2 + y^2)^{n-1}}\right]$
$ay\,dx + bx\,dy$ (a, b constants)	$x^{a-1}y^{b-1}$	$x^{a-1}y^{b-1}(ay\,dx + bx\,dy) = d\,(x^a y^b)$

5.1. Determine whether the differential equation $2xy\,dx + (1 + x^2)\,dy = 0$ is exact.

Using the indigenous differentials to identify the players of interest, $M(x, y) = 2xy$, and $N(x, y) = (1 + x^2)$. Taking a look at the partial of M with respect to y, reveals that it is $2x$. Likewise the partial of N with respect to x is $2x$. The test for exactness is passed by the subject equation.

5.2. Solve the differential equation given in Problem 5.1.

The equation having passed the exactness test, it is now time to seek solutions to the two blobs that make up the equation. The outcome of such searching, should it be successful, will constitute the function of two variables, g(x,y). The target conditions of the two blobs consist in:

$$\frac{\partial g(x, y)}{\partial x} = M(x, y)$$

and

$$\frac{\partial g(x, y)}{\partial y} = N(x, y)$$

Using the name tag from Problem 5.1, $M(x, y) = 2xy$,

$$\frac{\partial g(x, y)}{\partial x} = 2y$$

and if the reverse operation is applied to M, (with respect to x) the following expression may be used

$$\int \frac{\partial g(x, y)}{\partial x} = \int 2xy\, dx$$

$$\implies \qquad g(x, y) = x^2 y + h(y)$$

Since $g(x, y)$ is a function of two variables, differentiation with respect to y is not the reverse of the process that was just performed, and

$$\implies \qquad \frac{\partial g(x, y)}{\partial y} = x^2 + h'(y)$$

Choosing from available resources, the value of $h(y)$ can be found. The above expression is set adjacent to the $N(x, y)$ function, which it must equal.

$$\implies \qquad x^2 + h'(y) = 1 + x^2$$

readily yielding $h'(y) = 1$. Now the rhs quantity, above, can be integrated with respect to y:

$$\int 1 + x^2\, dy = g(x, y) = x^2 y + y + c_1$$

In the chapter intro it was posited that the pure form of $g(x, y)$ would be a constant, c. This putative constant may now be borrowed to stand in the equation

$$c = x^2 y + y + c_1$$

and if c_2 is chosen to equal $c - c_1$, then

$$x^2 y + y = c_2$$

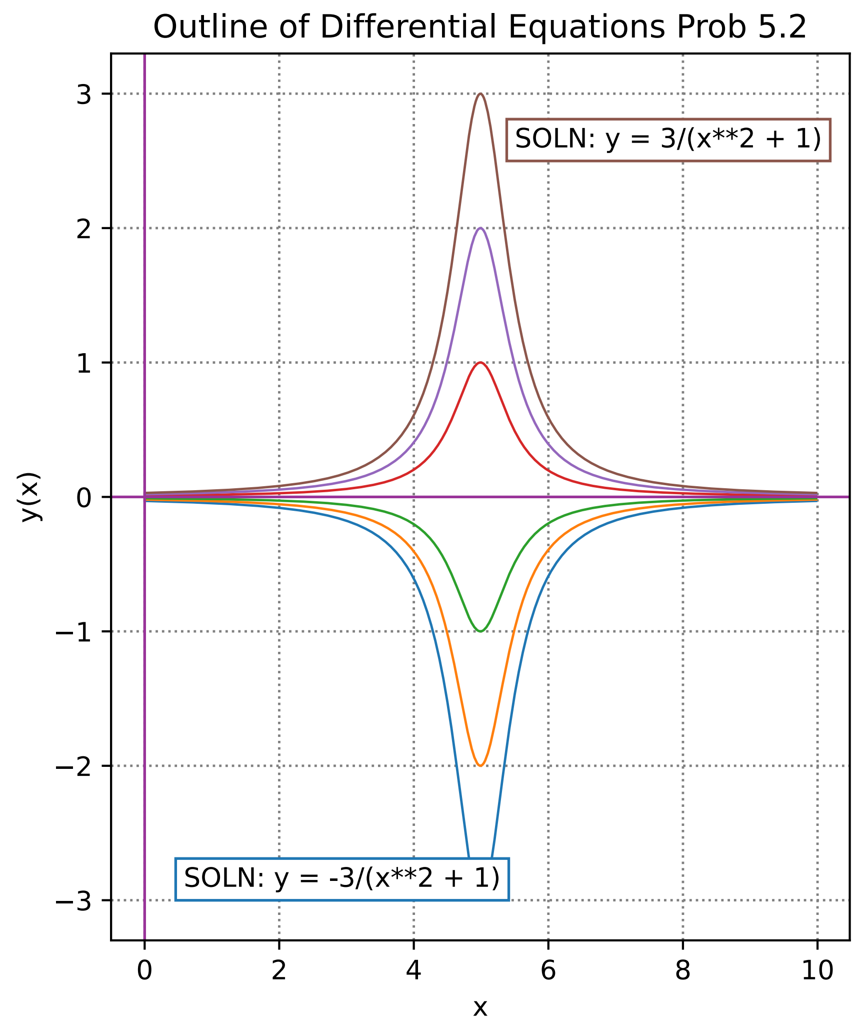
or if y is sought exclusively,

$$y = \frac{c_2}{x^2 + 1}$$

Some example plots in the family of solutions just developed may be of interest.

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

x=np.linspace(-10,10,500)

two = np.array([-3,-2,-1, 1, 2, 3])

def f(x):
    for k in two:
        y = k/(x**2 + 1)
        plt.plot(y, linewidth = 0.9)
y=f(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 5.2")
plt.rcParams['figure.figsize'] = [7, 7]

ax = plt.gca()
ratio = 1.2
#ratio is adjusted by eye to get squareness of x and y spacing
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)
ax.axhline(y=0, color='#993399', linewidth=1)
ax.axvline(x=0, color='#993399', linewidth=1)
ax.set_xticks([0, 100, 200, 300, 400, 500],
              labels=['0', '2', '4', '6', '8', '10' ])
ax.set_yticks([-3, -2, -1, 0, 1, 2, 3])
             # labels=['0', '1',  '2', '3',  4,  5,  6 ])

plt.text(29, -2.9, "SOLN: y = -3/(x**2 + 1)", size=10,bbox=dict\
         (boxstyle="square", ec=('#1F77B4'),fc=(1., 1., 1),))
plt.text(275, 2.6, "SOLN: y = 3/(x**2 + 1)", size=10,bbox=dict\
         (boxstyle="square", ec=('#8C564B'),fc=(1., 1., 1),))
```

Out[13]:
```
Text(275, 2.6, 'SOLN: y = 3/(x**2 + 1)')
```



5.3. Determine whether the differential equation $y\,dx - x\,dy = 0$ is exact.

To test for exactness, look at the partial derivatives making up the test. Here $M(x, y) = y$, and $N(x, y) = -x$. Taking a look at the partial of M with respect to y, reveals that it is 1. However, the partial of N with respect to x is -1. The partials being unequal, the equation is not exact.

5.4. Determine whether the differential equation $(x + \sin y)\,dx + (x\cos y - 2y)\,dy = 0$ is exact.

To test for exactness, look at the partial derivatives making up the test. Here $M(x, y) = (x + \sin y)$, and $N(x, y) = (x\cos y - 2y) = 0$. Taking a look at the partial of M with respect to y, reveals that it is $\cos y$. Meanwhile, the partial of N with respect to x is $\cos y$. The partials being equal, the equation is exact.

5.5. Solve the differential equation given in Problem 5.4.

The equation having been shown to be exact, the function of two variables $g(x, y) = c$, with derivatives equalling the M and N that have just been seen, will be sought. Trotting out the M introduced for this equation, and setting it up for integration, with respect to x

$$\int \frac{\partial g}{\partial x}\,dx = \int (x + \sin y)\,dx$$

and following through with the integration

$$g(x, y) = \frac{1}{2}x^2 + x\sin y + h(y)$$

The above, here designated "the unfinished", is not the most specific description that can be had about g(x,y). What is needed at this point is the exact value of h(y). To that end, the last equation is differentiated with respect to y.

```
In [24]: from sympy import *
         x, y, g, h = symbols('x y g h')
         h = Function("h")(y)
         hp = diff(h, y)
         hp
```

Out[24]: $\frac{d}{dy}h(y)$

In [25]:
```
g = (1/2*x**2 + x*sin(y))
gp = diff(g, y)
gp
```

Out[25]: $x\cos(y)$

In [30]:
```
gpa = gp + hp
gpa
```

Out[30]: $x\cos(y) + \frac{d}{dy}h(y)$

Since there is a close match in form between $N(x, y)$ and the derivative form above, which is required to be set equal to N, it is obvious that $h'(y) = -2y$. The integration with respect to y is a natural next move, bringing $h(y)$ into the form $h(y) = -y^2 + c_1$. At this point it makes sense to substitute $h(y)$ into the "unfinished" formulation above, and if this is done, then it is seen that

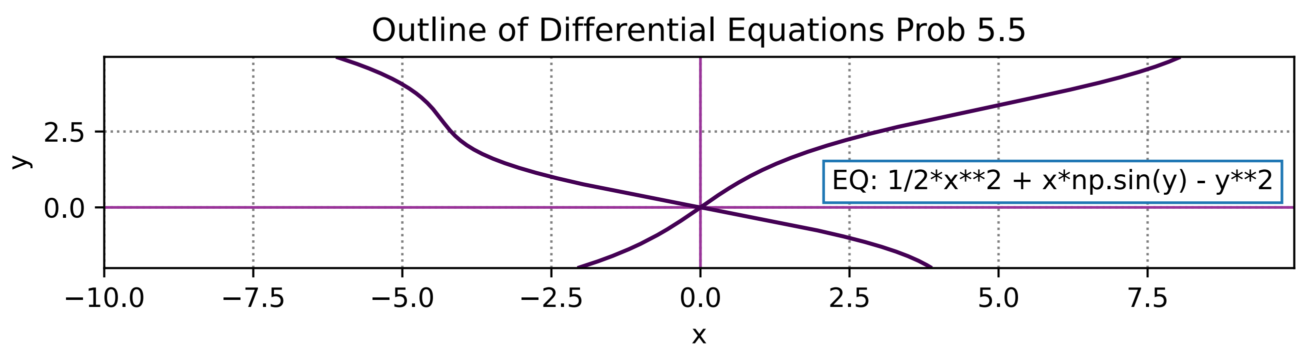
$$g(x, y) = \frac{1}{2}x^2 + x\sin y - y^2 + c_1$$

Since the orignal assertion was that $g(x, y) = c$, then by assigning $c_2 = c - c_1$, a final expression for the solution is given as

$$\frac{1}{2}x^2 + x\sin y - y^2 = c_2$$

The natural condition of implicitness attaches to the solution. But matplotlib can handle this task.

In [88]:
```
from matplotlib import pyplot as plt
import numpy as np

%config InlineBackend.figure_formats = ['svg']

delta = 0.04
xrange = np.arange(-10.0, 10.0, delta)
yrange = np.arange(-2.0, 5.0, delta)
x, y = np.meshgrid(xrange, yrange)

plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Outline of Differential Equations Prob 5.5")
plt.rcParams["figure.figsize"] = [8, 3]
ax = plt.gca()
ratio = 0.51
#ratio is adjusted by eye to get squareness of x and y spacing
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)
ax.axhline(y=0, color='#993399', linewidth=1)
ax.axvline(x=0, color='#993399', linewidth=1)
#ax.set_xticks([0, 100, 200, 300, 400, 500],
              # labels=['0', '2', '4', '6', '8', '10' ])
#ax.set_yticks([-3, -2, -1, 0, 1, 2, 3])
              # labels=['0', '1',  '2', '3',  4,  5,  6 ])
plt.text(2.2, 0.6, "EQ: 1/2*x**2 + x*np.sin(y) - y**2", size=10,bbox=dict\
         (boxstyle="square", ec=('#1F77B4'),fc=(1., 1., 1),))

equation = 1/2*x**2 + x*np.sin(y) - y**2
plt.contour(x, y, equation, [0])
plt.show()
```



5.6. Solve

$$y' = \frac{2 + ye^{xy}}{2y - xe^{xy}}$$

This equation needs to be tested for exactness. It therefore needs to be put into the form where M, N, and the indigenous differentials can be examined. This would look like:

$$(2y - x\exp(xy))\,dy = (2 + y\exp(xy))\,dx$$

Except that by convention both blobs are put on the same side of the equals sign, like

$$\implies (2 + y\exp(xy))\,dx + (-2y + x\exp(xy))\,dy = 0$$

The left hand blob is M, so its partial is taken with respect to y, and equals

$$\exp(xy) + xy\exp(xy)$$

The right hand blob is N, so its partial is taken with respect to x, and equals

$$\exp(xy) + xy\exp(xy)$$

The equality of the above two lines means that the equation is exact.

Blob M, instead of being differentiated with respect to y, is now integrated with respect to x, yielding

$$\int [2 + y\exp(xy)]\, dx$$

$$\Longrightarrow g(x, y) = 2x + \exp(xy) + h(y)$$

which may be referred to as the "unfinished g". If this last line is differentiated with respect to y, the expression

$$x\exp(xy) + h'(y)$$

is obtained. There is a chained obligation for this expression to equal blob N, thus

$$x\exp(xy) + h'(y) = x\exp(xy) - 2y$$

$$\Longrightarrow h'(y) = -2y$$

This last expression can be integrated to yield $h = -y^2 + c_1$

Substituting this h into the "unfinished g" line, results in

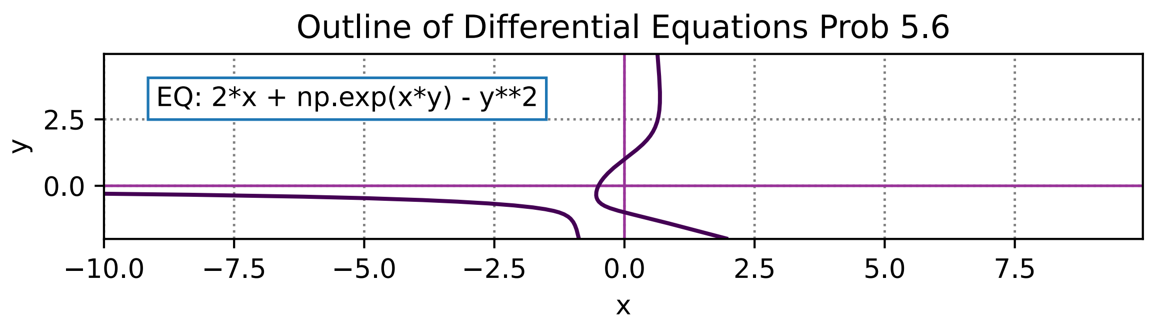
$$g(x, y) = 2x + \exp(xy) - y^2 + c_1$$

Pulling the $g(x, y) = c$ primordial equation out of the equation locker, gives a finish to the search with the solution

$$2x + \exp(xy) - y^2 = c_2$$

As in previous problems, the solution is implicit. The matplotlib treatment can be realized as shown below.

In [14]:
```
from matplotlib import pyplot as plt
import numpy as np

%config InlineBackend.figure_formats = ['svg']

delta = 0.04
xrange = np.arange(-10.0, 10.0, delta)
yrange = np.arange(-2.0, 5.0, delta)
x, y = np.meshgrid(xrange, yrange)

plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Outline of Differential Equations Prob 5.6")
plt.rcParams["figure.figsize"] = [8, 3]
ax = plt.gca()
ratio = 0.51
#ratio is adjusted by eye to get squareness of x and y spacing
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)
ax.axhline(y=0, color='#993399', linewidth=1)
ax.axvline(x=0, color='#993399', linewidth=1)
#ax.set_xticks([0, 100, 200, 300, 400, 500],
           # labels=['0', '2', '4', '6', '8', '10' ])
#ax.set_yticks([-3, -2, -1, 0, 1, 2, 3])
           # labels=['0', '1',  '2', '3',  4,  5,  6 ])
plt.text(-9, 3, "EQ: 2*x + np.exp(x*y) - y**2", size=10,bbox=dict\
        (boxstyle="square", ec=('#1F77B4'),fc=(1., 1., 1),))

equation = 2*x + np.exp(x*y) - y**2
plt.contour(x, y, equation, [0])
plt.show()
```



5.7. Determine whether the differential equation $y^2\,dt + (2yt + 1)\,dy = 0$ is exact.

The equation is already in differential form. The M blob is y^2 and the N blob is $(2yt + 1)\,dy$. The derivative of M with respect to y is $2y$, and the derivative of N with respect to t is $2y$. Therefore the equation is exact.

5.8. Solve the differential equation given in Problem 5.7.

The M blob was identified as y^2 , and already has the dt indigenous differential associated with it. The first task is to integrate M with respect to t:

$$\int \frac{\partial g}{\partial t}\, dt \;=\; \int y^2\, dt$$

$$\Longrightarrow g(y, t) \;=\; y^2 t \;+\; h(y)$$

Then to differentiate the last line with respect to y

$$2yt \;+\; \frac{dh}{dy}$$

Since it has the dy differential associated with it, it is bound to be equal to N. Therefore,

$$2yt \;+\; \frac{dh}{dy} \;=\; 2yt \;+\; 1$$

The last equation can be thought of as the "unfinished $g(x,\ y)$ ".

By inspection, $\frac{dh}{dy} \;=\; 1$, therefore $h(y) \;=\; y \;+\; c_1$ and the "unfinished $g(x,\ y)$ " becomes

$$g(t,\ y) \;=\; y^2 t \;+\; y \;+\; c_1$$

Pulling out the old primeaval expression of $g(x,\ y)$ results in a solution of

$$y^2 t \;+\; y \;=\; c_2$$

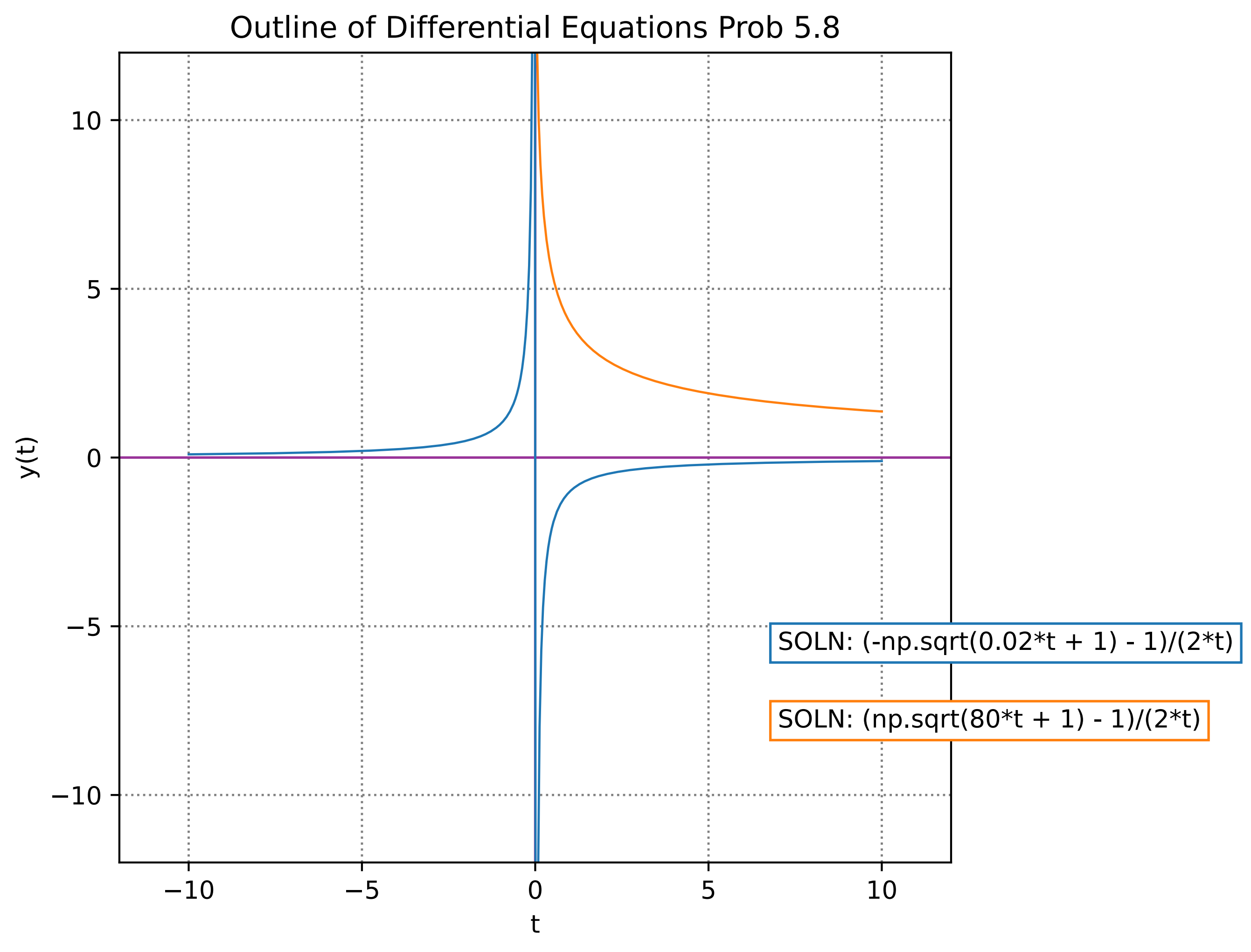
As it happens, the last line can be solved explicitly for y using the quadratic equation.

In [8]:
```
from sympy import *
y, t, c2 = symbols("y t c2")

solve(y**2*t + y - c2, y)
```

Out[8]:
```
[(-sqrt(4*c2*t + 1) - 1)/(2*t), (sqrt(4*c2*t + 1) - 1)/(2*t)]
```

In the plot below. For the first (negative) solution, the constant c_2 is assigned the value of 0.005; for the second, the constant c_2 is assigned the value of 20. By using two separate ranges for the x values, a runtime error condition is avoided.

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
from sympy import *

%config InlineBackend.figure_formats = ['svg']

#x = np.arange(-10., 10., 0.01)
x1 = np.setdiff1d(np.linspace(-10,10,400),[0]) #to remove the zero
x2 = np.setdiff1d(np.linspace(0,10,400),[0]) #to remove the zero
#y1 = (-np.sqrt(0.02*x1 + 1) - 1)/(2*x1)
#y2 = (np.sqrt(80*x2 + 1) - 1)/(2*x2)
t1 = (-np.sqrt(0.02*x1 + 1) - 1)/(2*x1)
t2 = (np.sqrt(80*x2 + 1) - 1)/(2*x2)

plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("t")
plt.ylabel("y(t)")
plt.title("Outline of Differential Equations Prob 5.8")
plt.rcParams["figure.figsize"] = [6, 6]

ax = plt.gca()
ax.axis([-12, 12, -12, 12.])

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

plt.text(7, -5.7, "SOLN: (-np.sqrt(0.02*t + 1) - 1)/(2*t)", size=10,\
         bbox=dict(boxstyle="square", ec=('#1F77B4'),fc=(1., 1., 1),))
plt.text(7, -8, "SOLN: (np.sqrt(80*t + 1) - 1)/(2*t)", size=10,\
         bbox=dict(boxstyle="square", ec=('#FF7F0E'),fc=(1., 1., 1),))

plt.plot(x1,t1, linewidth = 0.9)
plt.plot(x2,t2, linewidth = 0.9)
plt.show()
```



5.14. Determine whether $\frac{-1}{x^2}$ is an integrating factor for the differential equation $(y\,dx \;-\; x\,dy) \;=0\,.$

Multiplying the stated equation by the stated factor gives the expression:

$$\frac{-1}{x^2}\,(y\,dx \;-\; x\,dy) \;=\; 0$$

$$\Longrightarrow \frac{-y}{x^2}\,dx \;+\; \frac{1}{x}\,dy \;=\; 0$$

The form of the blobs in the last line is the right form for applying the test for equation exactness, with $M \;=\; \frac{-y}{x^2}$ and $N \;=\; \frac{1}{x}$

Taking the partial of M with respect to y gives $\frac{-1}{x^2}$. Taking the partial of N with respect to x gives $\frac{-1}{x^2}$.

The partials being equal means that the equation is exact, and that, therefore, the factor $\frac{-1}{x^2}$ is a integrating factor for the starting equation.

5.18. Solve $(y^2 \;-\; y)\,dx \;+\; x\,dy \;= 0$.

The equation as given is not exact. The text advises that a certain grouping of the elements will be helpful.

$$-\,(y\,dx \;-\; x\,dy) \;+\; y^2\,dx \;=\; 0$$

The purpose of this regrouping is to get the $(y\,dx \;-\; x\,dy)$ factor isolated. This factor is interesting because four different integrating factors are designed to fit with it. Guidance points to the integrating factor $\frac{1}{y^{\,2}}$ as the one which can unlock the exactness potential of the problem expression.

$$\Longrightarrow \frac{1}{y^2}\,(-\,(y\,dx \;-\; x\,dy) \;+\; y^2\,dx) \;=\; 0$$

$$\Longrightarrow \frac{-\,y\,dx \;+\; x\,dy}{y^{\,2}} \;+\; 1\,dx \;=\; 0$$

The expression on the last line needs to be rearranged into differential form in order to judge its characteristics.

$$(1 \;-\; \frac{1}{y})\,dx \;+\; \frac{x}{y^{\,2}}\,dy \;=\; 0$$

$$\boldsymbol{M} \qquad\qquad \boldsymbol{N}$$

Taking the partial derivative of M with respect to y and the partial derivative of N with respect to x , produces the forms

$$\frac{1}{y^{\,2}} \qquad = \qquad \frac{1}{y^{\,2}}$$

So the equation is now exact. To go on to the actual solution, the first task is to integrate the blob M with respect to its indigenous differential, dx

$$\int (1 - \frac{1}{y})\, dx$$

$$\Longrightarrow x - \frac{x}{y} + h(y)$$

Then to differentiate the resulting expression with respect to y:

$$\Longrightarrow \quad \frac{x}{y^2} + h'(y)$$

At this point the developed expression needs to match N, so to set that up

$$\Longrightarrow \quad \frac{x}{y^2} = \frac{x}{y^2} + h'(y)$$

This implies that $h'(y)$ is equal to zero, thus $h(y)$ merely a constant, and that the final solution is the one found three lines up, modified with the new intelligence about $h(y)$

$$\Longrightarrow x - \frac{x}{y} + h(y) = x - \frac{x}{y} + c$$

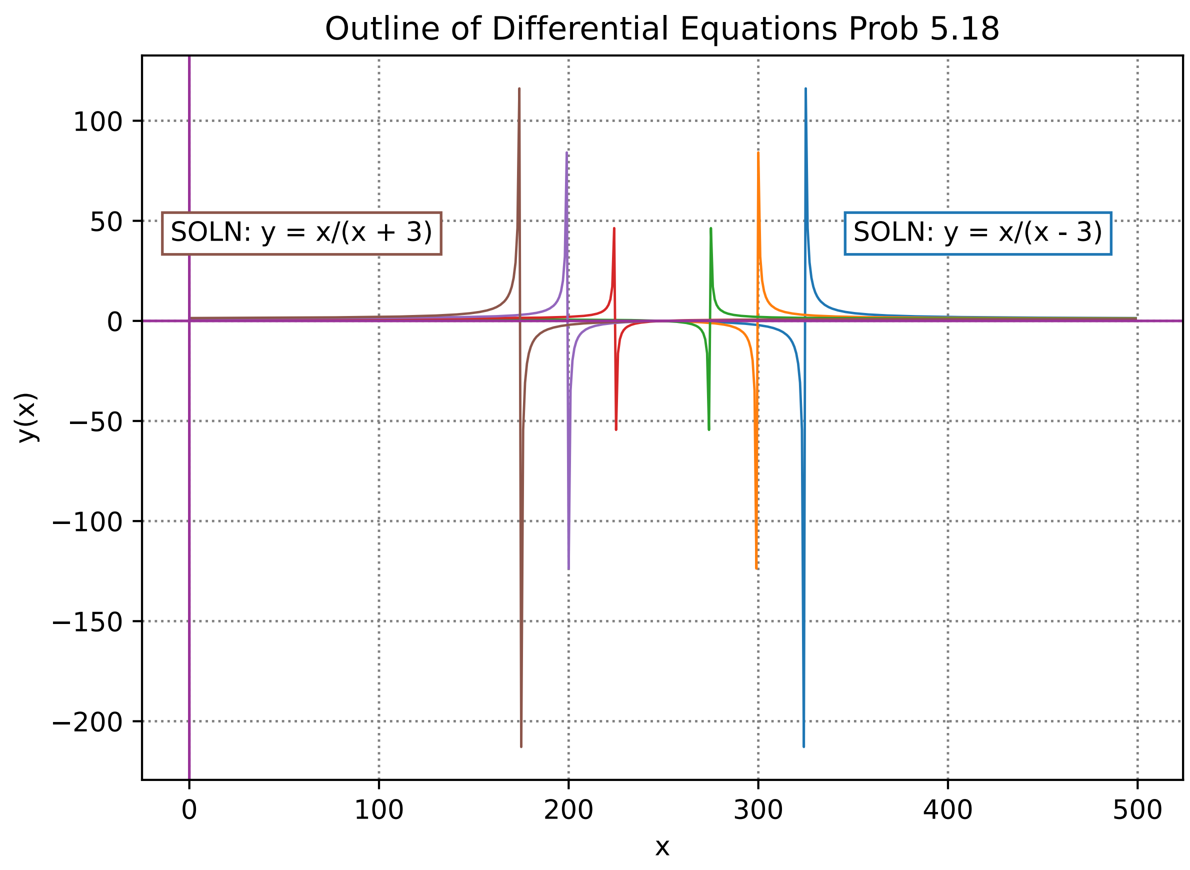
The plot below captures the function complemented by a family of constant values.

In [1]:
```
#rearrange the solution to represent an expression depending on y
from sympy import *
x, y, c = symbols("x y c")
solve(x - x/y +c, y)
```

Out[1]:
```
[x/(c + x)]
```

```
In [3]: import numpy as np
        import matplotlib.pyplot as plt

        %config InlineBackend.figure_formats = ['svg']

        x=np.linspace(-10,10,500)

        two = np.array([-3,-2,-1, 1, 2, 3])

        def f(x):
            for k in two:
                y = x/(x + k)
                plt.plot(y, linewidth = 0.9)
        y=f(x)

        plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
        #plt.rc('grid', linestyle='-', color='gray', linewidth=0.9)
        plt.xlabel("x")
        plt.ylabel("y(x)")
        plt.title("Outline of Differential Equations Prob 5.18")
        plt.rcParams['figure.figsize'] = [7, 5]

        ax = plt.gca()

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

        plt.text(350, 40, "SOLN: y = x/(x - 3)", size=10,bbox=dict\
                 (boxstyle="square", ec=('#1F77B4'),fc=(1., 1., 1),))
        plt.text(-10, 40, "SOLN: y = x/(x + 3)", size=10,bbox=dict\
                 (boxstyle="square", ec=('#8C564B'),fc=(1., 1., 1),))
```

```
Out[3]: Text(-10, 40, 'SOLN: y = x/(x + 3)')
```



5.19. Solve $(y - xy^2)\,dx + (x + x^2y^2)\,dy = 0$

The equation as given is not exact. The text advises that a certain grouping of the elements will be helpful.

$$(y\,dx + x\,dy) + (-xy^2\,dx + x^2y^2\,dy) = 0$$

The purpose of this regrouping is to get the $(y\,dx + x\,dy)$ factor isolated. This factor is interesting because four different integrating factors are designed to fit with it. Guidance points to the integrating factor $\frac{1}{(xy)^2}$ as the one which can unlock the exactness potential of the problem expression.

$$\Longrightarrow \frac{1}{(xy)^2}((y\,dx + x\,dy) + (-xy^2\,dx + x^2y^2\,dy)) = 0$$

$$\Longrightarrow \frac{y\,dx + x\,dy}{(xy)^2} + \frac{-xy^2\,dx + x^2y^2\,dy}{(xy)^2} = 0$$

The expression on the last line can be reduced in complexity, into what may be called the "bailout line".

$$\Longrightarrow \frac{y\,dx + x\,dy}{(xy)^2} + \frac{-1\,dx}{x} + 1\,dy = 0$$

The expression on the last line needs to be rearranged into a suitable differential form in order to judge its characteristics.

$$\Longrightarrow \underbrace{\left(\frac{y}{(xy)^2} - \frac{1}{x}\right)}_{\boldsymbol{M}} dx + \underbrace{\left(\frac{x}{(xy)^2} - 1\right)}_{\boldsymbol{N}} dy = 0$$

Sympy can help out with the exactness test.

In [15]:
```python
from sympy import symbols, diff
x, y = symbols('x y', real=True)
M = (y/(x*y)**2) - 1/x
ml = diff(M, y)
ml
```

Out[15]:
$$-\frac{1}{x^2y^2}$$

In [16]:
```python
from sympy import symbols, diff
x, y = symbols('x y', real=True)
N = (x /(x*y)**2) - 1
diff(N, x)
```

Out[16]:
$$-\frac{1}{x^2y^2}$$

So the equation is now exact. A quick alternative to the usual calculation procedure is to compare the "bailout line" with the table of integration factors above. In the 4th-to-last line of the table, in the right column, is an expression exactly like the "bailout line", with $n = 2$. In detail, it reads for the current problem,

$$\frac{y\,dx + x\,dy}{(xy)^2} = d\left(\frac{-1}{xy}\right)$$

Revisiting the "bailout line"

$$\Longrightarrow \frac{y\,dx \; + \; x\,dy}{(xy)^2} + \frac{-xy^2\,dx \; + \; x^2y^2\,dy}{(xy)^2} \; = \; 0$$

The second factor can be simplified to give the form

$$\Longrightarrow \frac{y\,dx \; + \; x\,dy}{(xy)^2} + \frac{-1\,dx}{x} \; + \; 1\,dy \; = \; 0$$

And then the substitution made from above

$$\Longrightarrow d\left(\frac{-1}{xy}\right) + \frac{-1\,dx}{x} \; + \; 1\,dy \; = \; 0$$

Or

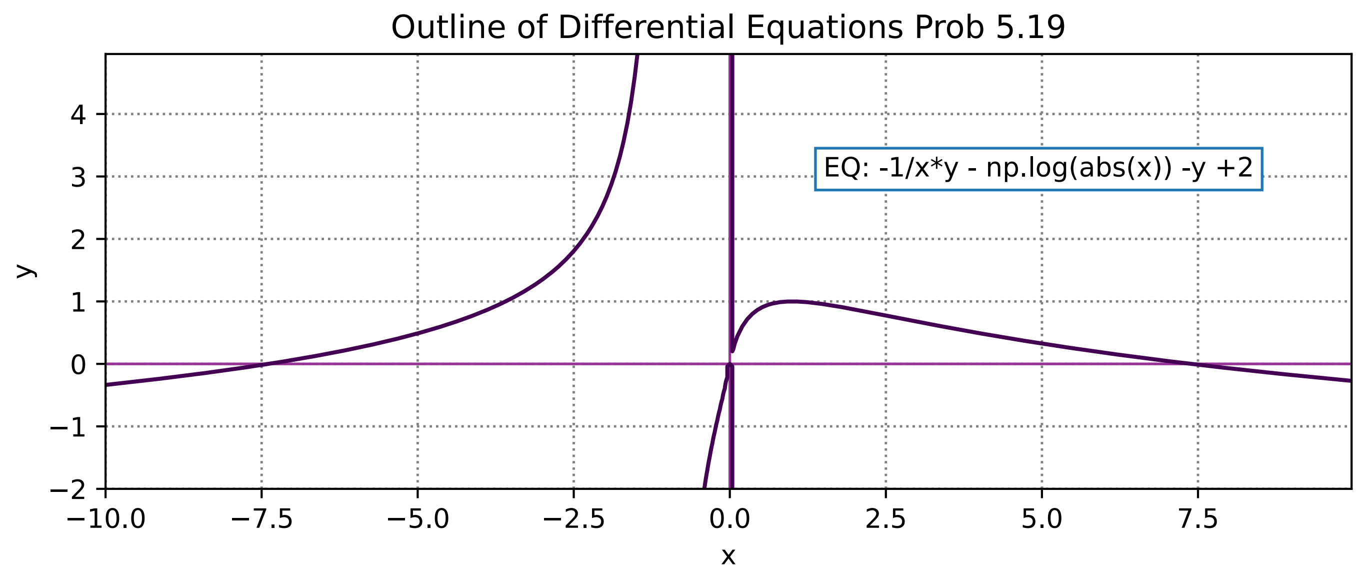
$$\Longrightarrow d\left(\frac{-1}{xy}\right) = \frac{1\,dx}{x} \; - \; 1\,dy$$

And both sides of the previous equation can be integrated with respect to their respective differentials to produce

$$\Longrightarrow \frac{-1}{xy} - \ln|x| \; - \; y \; - \; c$$

The previous expression is the solution, as usual in implicit form. To employ matplotlib to make a plot looks like the following (note the small "aversion gap" at $x \; = \; 0$.)

In [41]:
```
from matplotlib import pyplot as plt
import numpy as np

%config InlineBackend.figure_formats = ['svg']

delta = 0.04
xrange = np.arange(-10.0, 10.0, delta)
yrange = np.arange(-2.0, 5.0, delta)
x, y = np.meshgrid(xrange, yrange)

plt.grid(True, linestyle='dotted', color='gray', linewidth=0.9)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Outline of Differential Equations Prob 5.19")
plt.rcParams["figure.figsize"] = [9, 3]
ax = plt.gca()
ratio = 1
#ratio is adjusted by eye to get squareness of x and y spacing
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)
ax.axhline(y=0, color='#993399', linewidth=1)
ax.axvline(x=0, color='#993399', linewidth=1)
#ax.set_xticks([0, 100, 200, 300, 400, 500],
             # labels=['0', '2', '4', '6', '8', '10' ])
#ax.set_yticks([-3, -2, -1, 0, 1, 2, 3])
             # labels=['0', '1',  '2', '3',  4,  5,  6 ])
plt.text(1.5, 3, "EQ: -1/x*y - np.log(abs(x)) -y +2", size=10,bbox=dict\
         (boxstyle="square", ec=('#1F77B4'),fc=(1., 1., 1),))

equation = -1/x*y - np.log(abs(x)) -y + 2
plt.contour(x, y, equation, [0])
plt.show()
```



In the plot the constant c is assigned a value of 2.

In [ ]: