In [2]:
```
from sympy import *
```

Example 1: Solve t^2 dy/dt + 2t y = cos(t). We will continue to use "dsolve" to solve our ODEs as a check.

In [3]:
```
t=symbols('t')
y=Function('y')
deq=t**2*diff(y(t),t)+2*t*y(t)-cos(t)
ysoln=dsolve(deq,y(t))
print(ysoln)
```

```
Eq(y(t), (C1 + sin(t))/t**2)
```

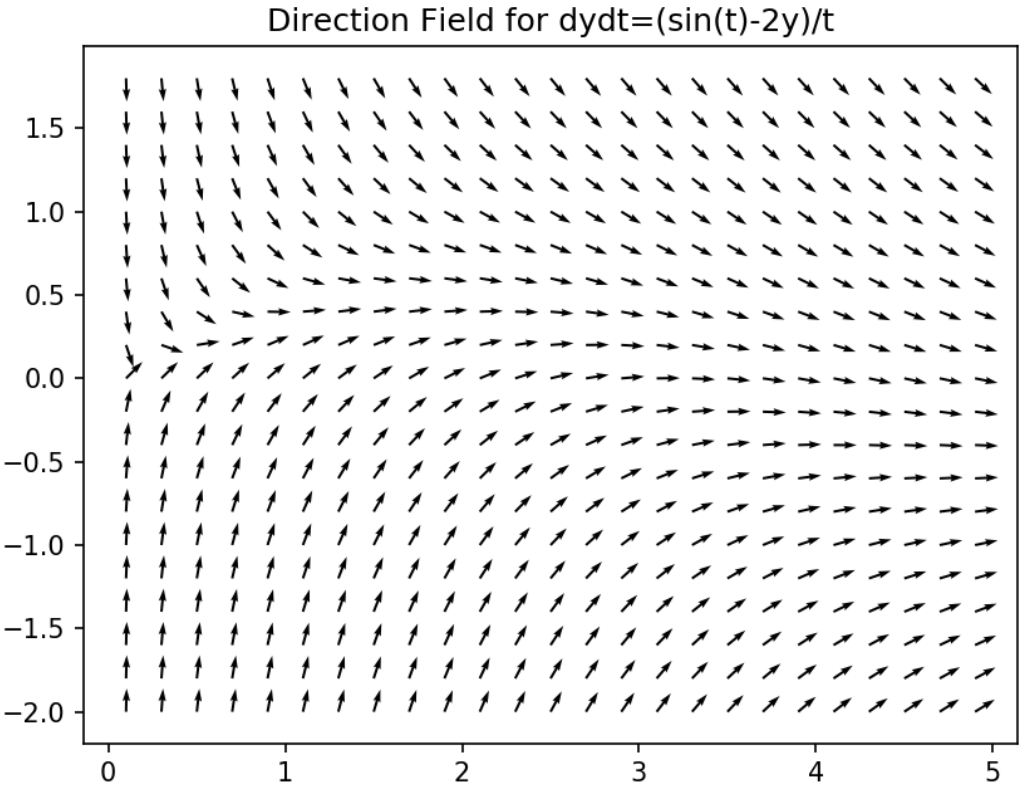
Example 2: Given the ODE t y' + 2 y = sin(t), t>0 (NOTE this is just a domain restriction and does not affect the strategy).

Also note that for the direction field, we write the ODE as y' = f(t,y), but for dsolve, I move everything to one side so I solve F(t,y,y') = 0.

In [4]:
```
# Part a: Plot direction field and describe how solutions behave as t->oo
import numpy as np
import matplotlib.pyplot as plt
```

In [5]:
```
matplotlib notebook
```

In [6]:
```
T, Y = np.meshgrid(np.arange(0.1, 5.1, .2), np.arange(-2, 2, .2))
dYdT = (np.sin(T)- 2*Y)/T
U = 1/(1+dYdT**2)**0.5*np.ones(T.shape)
V = 1/(1+dYdT**2)**0.5*dYdT
plt.figure()
plt.title('Direction Field for dydt=(sin(t)-2y)/t')
Q = plt.quiver(T, Y, U, V)
print('As t->oo the solutions appear to approach a slope of 0 (a horizontal asymptote).')
```




```
As t->oo the solutions appear to approach a slope of 0 (a horizontal asympt
ote).
```

In [7]:
```
# Part b: Find the general solution
t=symbols('t')
y=Function('y')
deq=t*diff(y(t),t)+2*y(t)-sin(t)
ysoln=dsolve(deq,y(t))
print(ysoln)
```

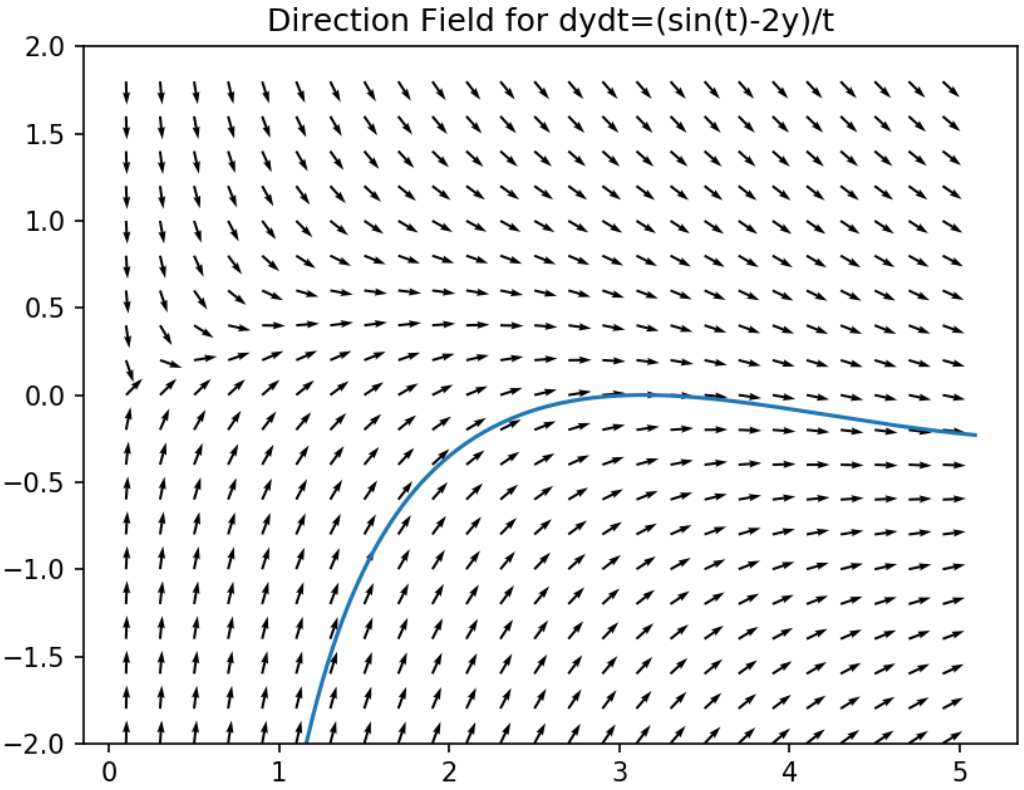
```
Eq(y(t), (C1/t - cos(t) + sin(t)/t)/t)
```

In [8]:
```
# Part c: use initial condition
ysoln=dsolve(deq,y(t),ics={y(pi):0})
print(ysoln)
```

```
Eq(y(t), (-cos(t) + sin(t)/t - pi/t)/t)
```

In [9]:
```
matplotlib notebook
```

In [10]:
```
# Replot direction field with solution to IVP
T, Y = np.meshgrid(np.arange(0.1, 5.1, .2), np.arange(-2, 2, .2))
dYdT = (np.sin(T)- 2*Y)/T
U = 1/(1+dYdT**2)**0.5*np.ones(T.shape)
V = 1/(1+dYdT**2)**0.5*dYdT
plt.figure()
plt.title('Direction Field for dydt=(sin(t)-2y)/t')
Q = plt.quiver(T, Y, U, V)
tplot=np.arange(0.1,5.1,0.01)
yplot=(-tplot*np.cos(tplot)+np.sin(tplot)-pi)/tplot**2
plt.plot(tplot,yplot)
plt.ylim(-2,2) # Setting the y-range of the graph to match the direction fi
eld
```

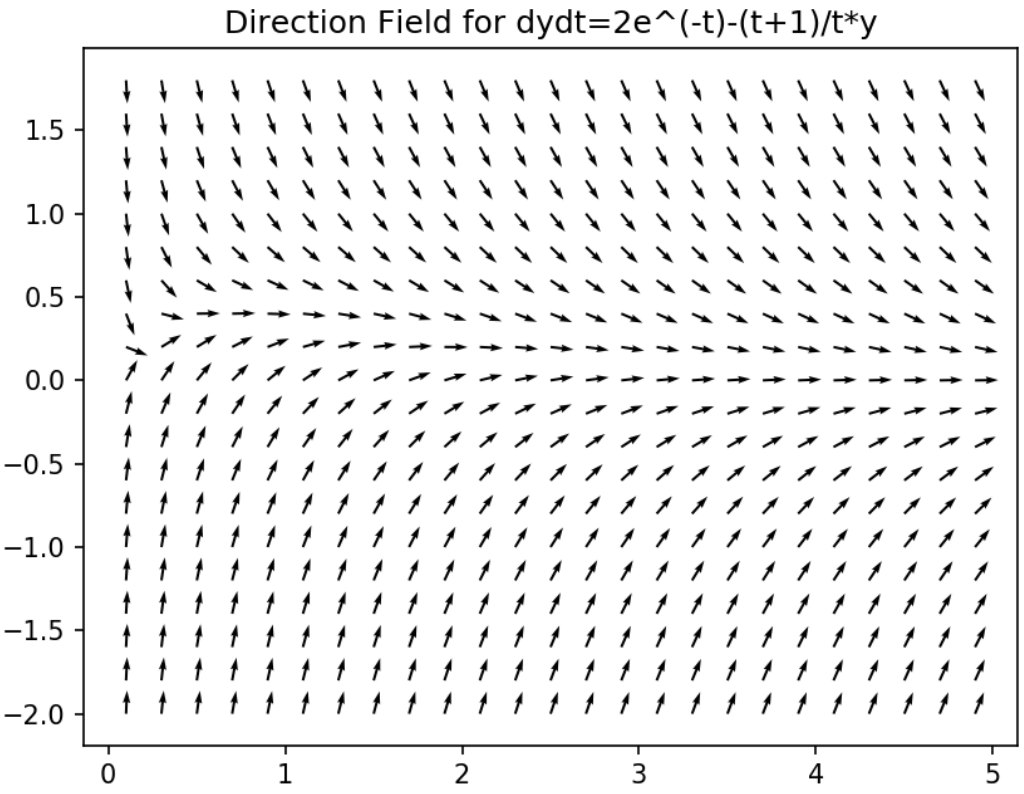


Out[10]: (-2, 2)

Example 3: (in standard form): y' + (t+1)/t y = 2e^(-t), y(1)=a, t > 0

In [11]:
```
matplotlib notebook
```

In [12]:
```
# Part a
T, Y = np.meshgrid(np.arange(0.1, 5.1, .2), np.arange(-2, 2, .2))
dYdT = 2*np.exp(-T)-(T+1)/T*Y
U = 1/(1+dYdT**2)**0.5*np.ones(T.shape)
V = 1/(1+dYdT**2)**0.5*dYdT
plt.figure()
plt.title('Direction Field for dydt=2e^(-t)-(t+1)/t*y')
Q = plt.quiver(T, Y, U, V)
print('For a<0.3, function approaches -oo as t->0, and for a>0.3, function
 approaches oo as t->0.')
```




```
For a<0.3, function approaches -oo as t->0, and for a>0.3, function approac
hes oo as t->0.
```

In [17]:
```
# Part b
t=symbols('t')
y=Function('y')
a=symbols('a')
deq=t*diff(y(t),t)+(t+1)*y(t)-2*t*exp(-t)
ysoln=dsolve(deq,y(t),ics={y(1):a})
print(ysoln)
# Part c: Note that as t->0, the sign of the numerator determines whether w
e approach oo or -oo
acrit=solve(E*a+0**2-1,a)
print('The critical value is',acrit[0],'or approximately',acrit[0].evalf())
print('When a=',acrit,', the critical value,',ysoln.subs(a,acrit[0]),'which
approaches 0 as t->0.')
```

```
Eq(y(t), (E*a + t**2 - 1)*exp(-t)/t)
The critical value is exp(-1) or approximately 0.367879441171442
When a= [exp(-1)] , the critical value, Eq(y(t), t*exp(-t)) which approache
s 0 as t->0.
```

In [ ]: