Mathematical Economics

Alpha Chiang

Chapter 15

First-Order Linear Differential Equations with Constant/Coefficient and Constant Term

In [2]:
```
from sympy import Symbol, dsolve, Function, Derivative, Eq

y = Function("y")
x = Symbol('x')
t = Symbol('t')
w = Function('w')                #15.1
u = Function('u')

d1 = Derivative(y(t),t)

Eq(d1 + u(t) * y(t),w(t))
```

Out[2]:
$$u(t)y(t) + \frac{d}{dt}y(t) = w(t)$$

The Homogeneous Case

In [3]:
```
a = Symbol('a')

d2 = Derivative(y(t),t)
Eq(d2 + a * y(t),0)
```

Out[3]:
$$ay(t) + \frac{d}{dt}y(t) = 0$$

In [4]:
```
A = Symbol('A')
e = Symbol('e')
Eq(A * e**(-a*t),y(t))#general solution
```

Out[4]: $Ae^{-at} = y(t)$

Example 1

In [5]:
```
d3 = Derivative(y(t),t)
display(Eq(d3 + 2* y(t),6))
dsolve(d3 + 2* y(t) - 6, y(t))
```

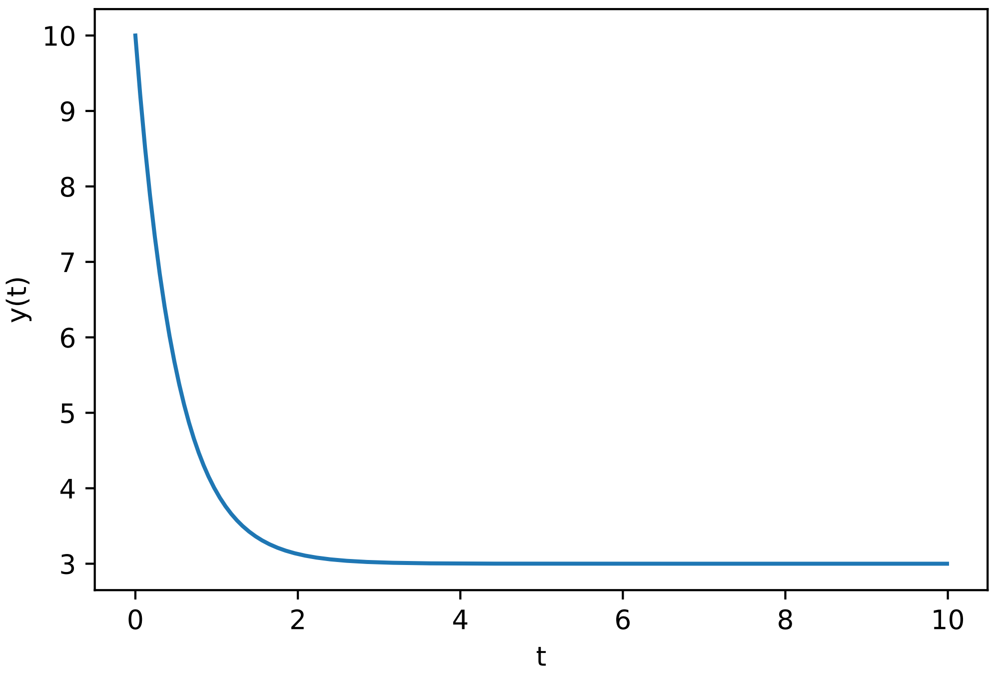
$$2y(t) + \frac{d}{dt}y(t) = 6$$

Out[5]: $y(t) = C_1e^{-2t} + 3$

In [6]:
```
%matplotlib inline
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('svg', 'png')
import matplotlib as mpl
mpl.rcParams['figure.dpi'] = 400
```

In [8]:
```
%config InlineBackend.figure_format = 'svg'
from scipy.integrate import odeint
import numpy as np

def f(y, t):

    return -2 * y + 6

y0 = 10
a = 0
b = 10

t = np.arange(a, b, 0.01)
y = odeint(f, y0, t)

import pylab
pylab.plot(t, y)
pylab.xlabel('t'); pylab.ylabel('y(t)')
```

Out[8]: `Text(0, 0.5, 'y(t)')`



Example 2

In [9]:
```
from sympy import Symbol, dsolve, Function, Derivative, Eq
y = Function("y")
x = Symbol('x')
t = Symbol('t')
w = Function('w')
u = Function('u')
d4 = Derivative(y(t),t)
display(Eq(d4 + 4*y(t),0))
dsolve(d3 + 4* y(t) , y(t))
```

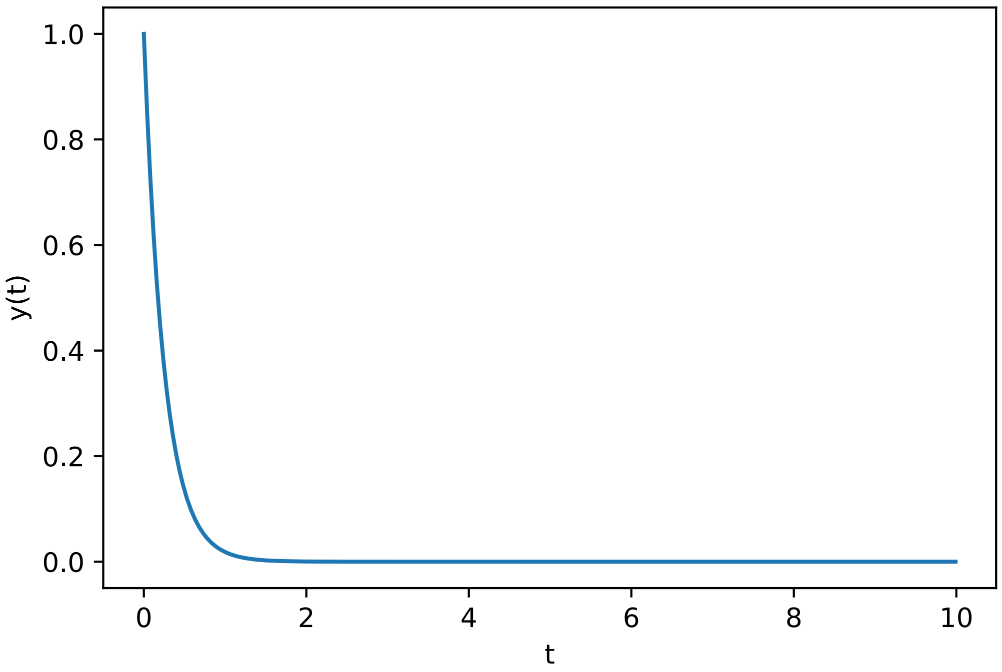
$$4y(t) + \frac{d}{dt}y(t) = 0$$

Out[9]: $y(t) = C_1 e^{-4t}$

In [10]:
```
def f(y, t):

    return -4 * y
```

```
y0 = 1
a = 0
b = 10

t = np.arange(a, b, 0.01)

y = odeint(f, y0, t)

import pylab
pylab.plot(t, y)
pylab.xlabel('t'); pylab.ylabel('y(t)')
```

Out[10]: Text(0, 0.5, 'y(t)')



In [15]:
```
from sympy import Symbol, dsolve, Function, Derivative, Eq
P = Function("P")
j = Symbol("j")
beta = Symbol("\\beta")
gamma = Symbol("\\gamma")
delta = Symbol("\\delta")
alpha = Symbol("\\alpha")
t = Symbol("t")
```

In [12]:

```
d5 = Derivative(P(t),t)
display(Eq(d5 + j * (beta + delta) * P(t),j*(alpha + gamma)))
```

$$j\left(\beta + \delta\right)P(t) + \frac{d}{dt}P(t) = j\left(\alpha + \gamma\right)$$

In [13]:

```
dsolve(d5 + j * (beta + delta) * P(t) - j*(alpha + gamma) , P(t))
```

Out[13]:

$$P(t) = \frac{\alpha + \gamma + e^{-C_1\beta - C_1\delta - \beta jt - \delta jt}}{\beta + \delta}$$

Variable Coefficient and Variable Term

Example 1

In [17]:

```
y = Function("y")
d6 = Derivative(y(t),t)
display(Eq(d6 + (3*t**2 * y(t)),0))
dsolve(d6 + 3*t**2*y(t), y(t))
```

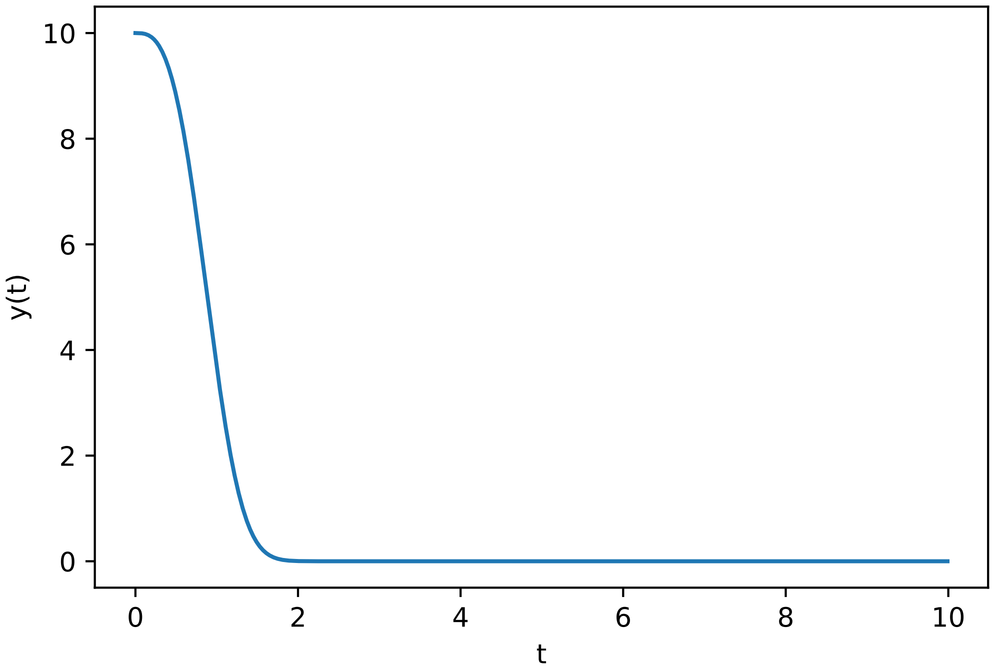
$$3t^2y(t) + \frac{d}{dt}y(t) = 0$$

Out[17]: $y(t) = C_1e^{-t^3}$

In [18]:

```
def f(y, t):

    return -3*y*t**2

y0 = 10
a = 0
b = 10

t = np.arange(a, b, 0.01)

y = odeint(f, y0, t)

import pylab
pylab.plot(t, y)
pylab.xlabel('t'); pylab.ylabel('y(t)')
```

Out[18]: Text(0, 0.5, 'y(t)')



Example 2

In [2]:
```
from sympy import Symbol, dsolve, Function, Derivative, Eq
y = Function("y")
t = Symbol("t")
d7 = Derivative(y(t),t)
display(Eq(d7 + (2*t * y(t)),t))
dsolve(d7 + t*2*y(t) - t, y(t))
```

$$2ty(t) + \frac{d}{dt}y(t) = t$$

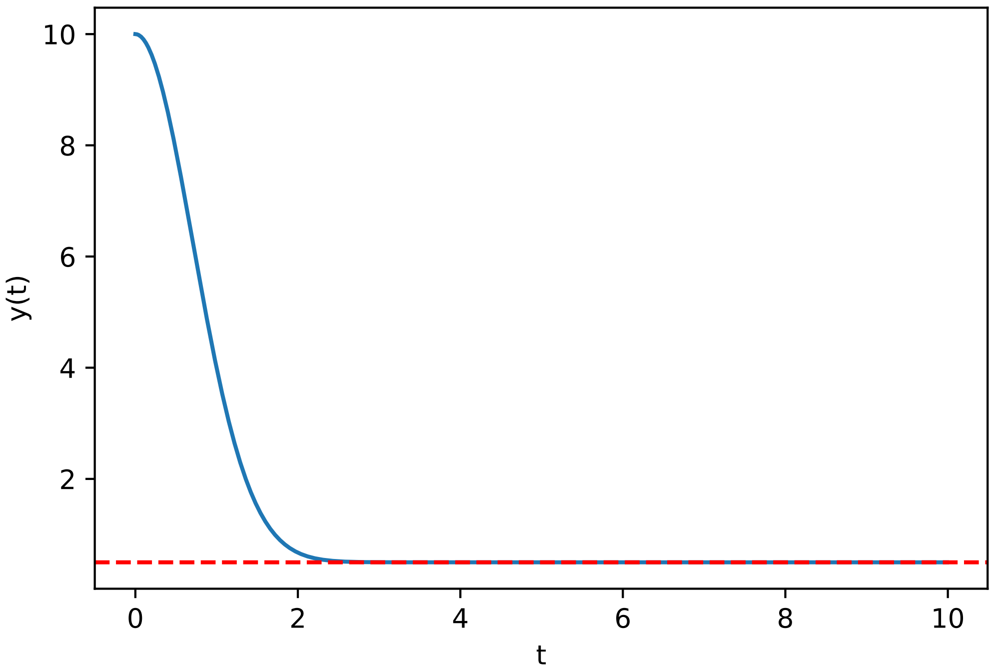
Out[2]:

$$y(t) = \frac{C_1 e^{-t^2}}{2} + \frac{1}{2}$$

In [3]:
```
%matplotlib inline
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('svg', 'png')
import matplotlib as mpl
```

```
mpl.rcParams['figure.dpi'] = 400
%config InlineBackend.figure_format = 'svg'
from scipy.integrate import odeint
import numpy as np
def f(y, t):

    return -2*y*t + t

y0 = 10
a = 0
b = 10

t = np.arange(a, b, 0.01)
y = odeint(f, y0, t)
import pylab
pylab.plot(t, y)
pylab.axhline(y = 0.5, color = 'r', linestyle = "dashed") #equilibrium
pylab.xlabel('t'); pylab.ylabel('y(t)')
```

Out[3]: Text(0, 0.5, 'y(t)')



Solow Growth Model -- A Quantitative Illustration

In [4]:
```
from sympy import Symbol, dsolve, Function, Derivative, Eq
z = Function("z")
s = Symbol("s")


lambd = Symbol("\\lambda")
alpha = Symbol("\\alpha")
t = Symbol("t")
d8 = Derivative(z(t),t)
display(Eq(d8 + (1- alpha)* lambd * z(t) ,(1- alpha)*s))
dsolve(d8 + (1- alpha)* lambd * z(t) - (1- alpha)*s, z(t))
```

$$\lambda\left(1-\alpha\right)z(t)+\frac{d}{dt}z(t)=s\left(1-\alpha\right)$$

Out[4]:
$$z(t)=\frac{s+e^{\lambda(C_1+\alpha t-t)}}{\lambda}$$

Furkan zengin