2. MATRIX RELATED

f) Exercise on matrices, subplots & differential equations

November 27, 2021

1 Solving differential equations

1.1 $\frac{dy}{dx} = -6xy$: $y(0) = 3$

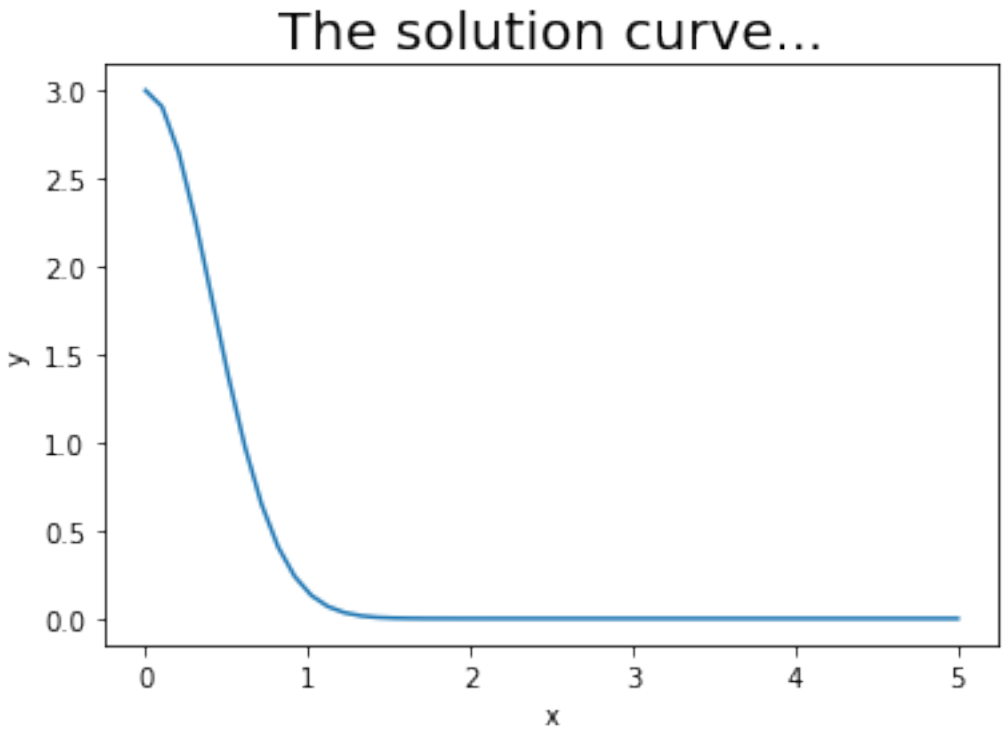
[3]:
```
from sympy import Eq, Symbol, Function, dsolve, solve

x = Symbol("x")
y = Function("y")(x)
ds = Eq(y.diff(x), -6*x*y)
gs = dsolve(ds, y)

C1 = Symbol("C1")
# NOTES
# subs accepts a dictionary as argument.
# In solve, no need to give 2nd argument if there is only 1 variable.
ps = gs.subs({C1 : solve(gs.subs({y : 3, x : 0}), C1)[0]})
# The index 0 is referred to for the solution of the GS when y = 3 and x = 0.
# This is because the return value of solve is a list, and the solution(s) are␣
 ↪the elements.
# This equation has only one solution for C1, hence only one element at index 0.
print("The solution...")
ps
```

```
The solution...
```

[3]: $y(x) = 3e^{-3x^2}$

[5]:
```
from numpy import linspace
from matplotlib.pyplot import plot, title, xlabel, ylabel, legend
from scipy.integrate import odeint

# Function that returns dy/dt
def dy_dx(y, x):
    return -6*y*x

# The initial condition
y0 = 3
```

```
# Time points
x = linspace(0, 5)

# Solve ODE
y = odeint(dy_dx, y0, x) # y-values

# Plot result
plot(x, y)
title("The solution curve...", size = 20)
xlabel("x")
ylabel("y")
None
```



1.2 $\frac{dy}{dx} = k + \frac{y}{2}$: $y(0) = 1$

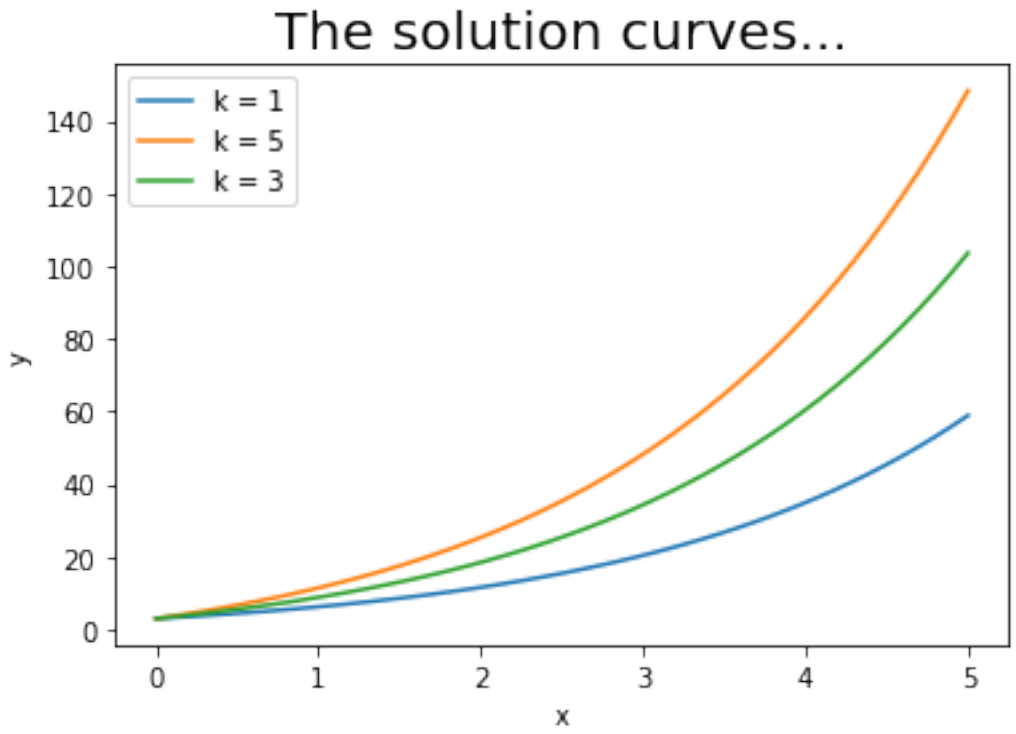
[6]:
```
from sympy import Eq, Symbol, Function, dsolve, solve

x = Symbol("x")
y = Function("y")(x)
k = Symbol("k")
ds = Eq(y.diff(x), k + y/2)
gs = dsolve(ds, y)
```

```
C1 = Symbol("C1")
# NOTES
# subs accepts a dictionary as argument.
# In solve, no need to give 2nd argument if there is only 1 variable.
ps = gs.subs({C1 : solve(gs.subs({y : 1, x : 0}), C1)[0]})
# The index 0 is referred to for the solution of the GS when y = 3 and x = 0.
# This is because the return value of solve is a list, and the solution(s) are␣
 ↪the elements.
# This equation has only one solution for C1, hence only one element at index 0.
print("The solution...")
ps
```

```
The solution...
```

[6]:
$$y(x) = -2k + (2k+1)\,e^{\frac{x}{2}}$$

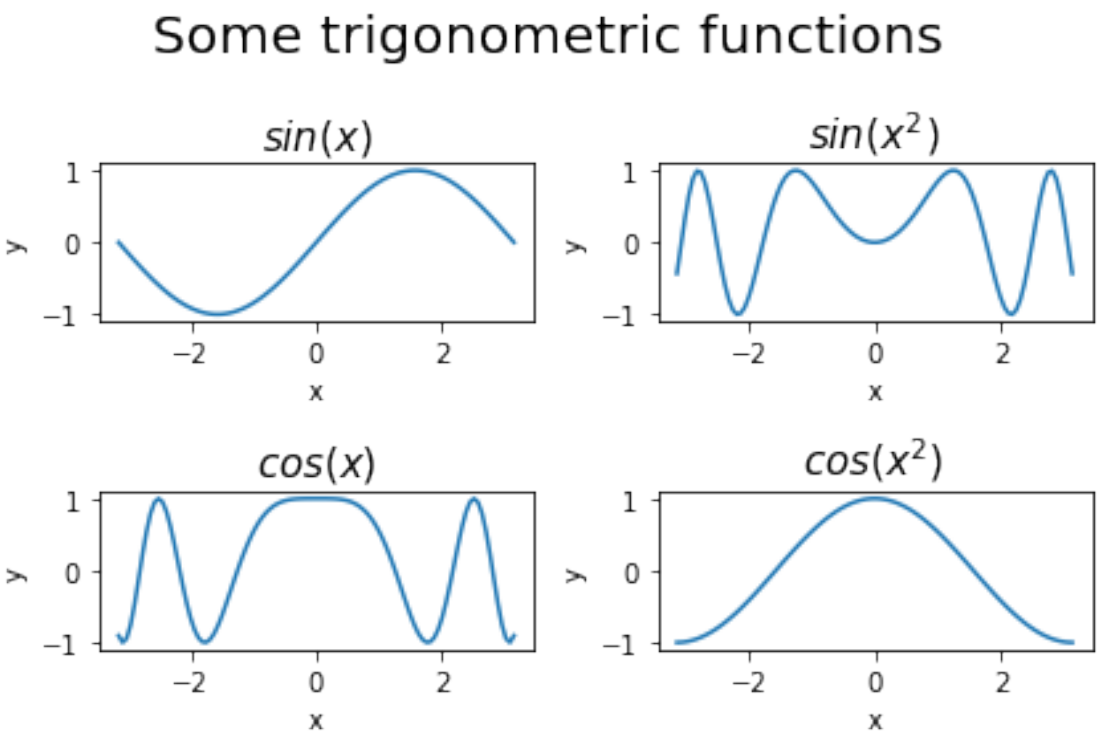
[7]:
```
from numpy import linspace
from matplotlib.pyplot import plot, title, xlabel, ylabel, legend
from scipy.integrate import odeint

# Function that returns dy/dt
def dy_dx(y, x):
    return k + y/2

# The initial condition
y0 = 3

# Time points
x = linspace(0, 5)

# Solve ODE
ys = list() # To append a whole list of y-values in each iteration
legends = list() # To store the various legend strings
ks = (1, 5, 3) # Stores different values for k
for k in ks:
    ys.append(odeint(dy_dx, y0, x)) # y-values
    legends.append("k = {0}".format(k)) # Legends

# Plot result
plot(x, ys[0], x, ys[1], x, ys[2])
legend([legends[0], legends[1], legends[2]])
title("The solution curves...", size = 20)
xlabel("x")
ylabel("y")
None
```



2 Subplots for certain trigonometric functions

[8]:
```
from numpy import sin, cos, linspace, pi
from matplotlib.pyplot import plot, title, xlabel, ylabel, subplot, suptitle,␣
 ↪tight_layout

# Since all plots apply the same labels, I have made a function to assign␣
 ↪labels to the plot.
def labels():
    xlabel("x", size = 10)
    ylabel("y", size = 10)

x = linspace(-pi, pi, 100)

suptitle("Some trigonometric functions", size = 20)
subplot(2, 2, 1)
y = sin(x)
plot(x, y)
labels()
title("$sin(x)$", size = 15)

subplot(2, 2, 2)
```

```
y = sin(x**2)
plot(x, y)
labels()
title("$sin(x^2)$", size = 15)

subplot(2, 2, 3)
y = cos(x**2)
plot(x, y)
labels()
title("$cos(x)$", size = 15)

subplot(2, 2, 4)
y = cos(x)
plot(x, y)
labels()
title("$cos(x^2)$", size = 15)

tight_layout()
# Must be given after the subplots.
# It ensures that there is sufficient space between the subplots

None
```



3 Miscellaneous problem

The sum of the digits of a three digit number is 16. The unit digit is two more than the sum of the other two digits. And the tens digit is five more than the hundreds digit. What is the number?

We can use algebra, but here, we will apply the brute force method using loops.

[97]:
```
for hundreds in range(0, 10):
    tens = hundreds + 5
    units = tens + hundreds + 2
    if units + tens + hundreds == 16:
        number = units + 10*tens + 100*hundreds
        print("The number required is", number)
```

```
The number required is 169
```