# TaylorSeries

February 11, 2022

[3]:
```
#Load necessary modules
import numpy as np
import matplotlib.pyplot as plt
```

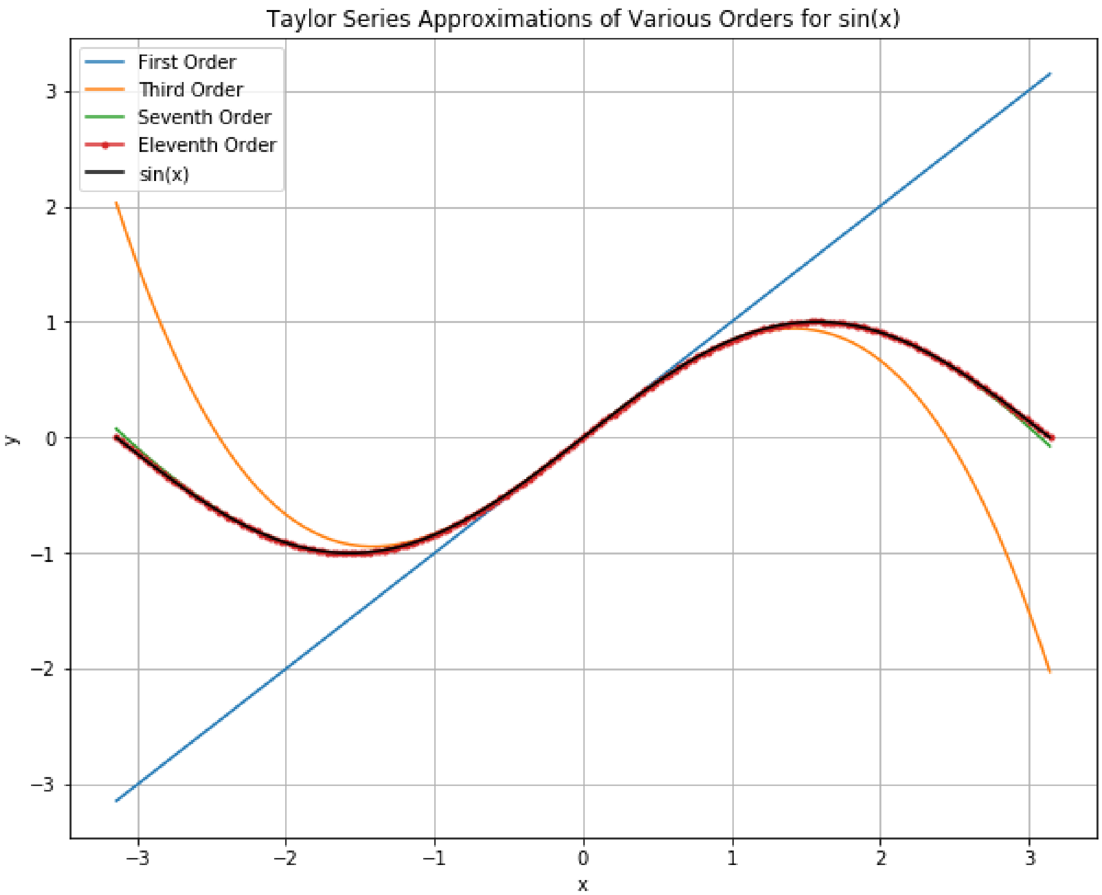
## 1 Taylor series expansion:

Taylor series expansion of the sine function close to zero for an increasing amount of approximated terms. Only odd powers of $x$ contribute to the expansion.

(Then coding can be done much more efficiently with functions, but this hopefully shows a bit better what is actually done).

[4]:
```
# x values
x = np.linspace(-np.pi, np.pi, 200)

plt.figure(figsize = (10,8))

#First order
n=0
#approximate y values
y = np.zeros(len(x))
#Because only every other term is not equal to zero, we only have to go to n/2
n=n/2
#Taylor series expansion up to n
for i in np.arange(0,n+1):
    y = y + ((-1)**i * (x)**(2*i+1)) / np.math.factorial(2*i+1)
#Plotting
plt.plot(x,y, label = 'First Order')

#Third order (same structure as before)
n=2
y = np.zeros(len(x))
n=n/2
for i in np.arange(0,n+1):
        y = y + ((-1)**i * (x)**(2*i+1)) / np.math.factorial(2*i+1)
plt.plot(x,y, label = 'Third Order')
```

```python
#Seventh order
n=6
y = np.zeros(len(x))
n=n/2
for i in np.arange(0,n+1):
        y = y + ((-1)**i * (x)**(2*i+1)) / np.math.factorial(2*i+1)
plt.plot(x,y, label = 'Seventh Order')

#Eleventh order
n=10
y = np.zeros(len(x))
n=n/2
for i in np.arange(0,n+1):
        y = y + ((-1)**i * (x)**(2*i+1)) / np.math.factorial(2*i+1)
plt.plot(x,y, label = 'Eleventh Order', marker='.')

#Plotting of full solution, i.e. sin(x)
plt.plot(x, np.sin(x), 'k', label = 'sin(x)')

plt.grid()
plt.title('Taylor Series Approximations of Various Orders for sin(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```


Taylor Series Approximations of Various Orders for sin(x)
First Order
Third Order
Seventh Order
Eleventh Order
sin(x)
y
x
3
2
1
0
−1
−2
−3

```
[ ]:
```