

# Series numericas: Función $\sin(x)$

Jorge Antonio Herrera Alonso, Elizabeth Hernández Martín, Yessica  
Sabrina Gómez Buso

12 de mayo de 2014

Facultad de Matemáticas  
Universidad de La Laguna

## 1 Motivación y objetivos

- 1 Motivación y objetivos
- 2 Fundamentos teóricos
  - Teorema de Taylor
  - Polinomio de Taylor

## 1 Motivación y objetivos

## 2 Fundamentos teóricos

- Teorema de Taylor
- Polinomio de Taylor

## 3 Procedimiento experimental

- Descripción de los experimentos
- Descripción del material
- Análisis de los resultados

- 1 Motivación y objetivos
- 2 Fundamentos teóricos
  - Teorema de Taylor
  - Polinomio de Taylor
- 3 Procedimiento experimental
  - Descripción de los experimentos
  - Descripción del material
  - Análisis de los resultados
- 4 Algoritmo

- 1 Motivación y objetivos
- 2 Fundamentos teóricos
  - Teorema de Taylor
  - Polinomio de Taylor
- 3 Procedimiento experimental
  - Descripción de los experimentos
  - Descripción del material
  - Análisis de los resultados
- 4 Algoritmo
- 5 Bibliografía

# Motivación y objetivos

## Motivación

Aprendizaje orientado al lenguaje de programación *Python*, el procesador de texto *LaTEX* y el creador de presentaciones *Beamer*.

## Objetivos

- **Objetivo principal:** Implementación de *Python* del método de Taylor.

## Motivación

Aprendizaje orientado al lenguaje de programación *Python*, el procesador de texto *LaTEX* y el creador de presentaciones *Beamer*.

## Objetivos

- **Objetivo principal:** Implementación de *Python* del método de Taylor.
- **Objetivo específico:** Aproximación de una función mediante el método de Taylor.



# Teorema de Taylor

## Teorema de Taylor:

El Teorema de Taylor permite obtener aproximaciones polinómicas de una función en un entorno de cierto punto en que la función sea diferenciable. Además el teorema permite acotar el error obtenido mediante dicha estimación.

## Definición

*Sea  $I$  un intervalo,  $c$  un punto de  $I$ ,  $f$  una función definida en  $I$ . Supongamos que  $f$  es derivable en todos los puntos hasta el orden  $n - 1$  y que existe  $f^n(c)$ . Entonces:*

## Definición

*Dada una función  $f$  derivable  $n$  veces en un punto  $c$ . se llama polinomio de Taylor en  $c$  de orden  $n$  al polinomio:*

$$f(x) = f(c) + f'(c)(x - c) + \frac{1}{2!}f''(c)(x - c)^2 + ..... + \frac{1}{n!}f^n(c)(x - c)^n$$

## Descripción:

- 1 Realización de varios códigos en Lenguaje *Python*.

## Descripción:

- 1 Realización de varios códigos en Lenguaje *Python*.
- 2 Repetición de la ejecución del código.

## Descripción:

- 1 Realización de varios códigos en Lenguaje *Python*.
- 2 Repetición de la ejecución del código.
- 3 Aproximación  $f(x) = \sin(x)$  mediante el método de Taylor.

## Descripción:

- 1 Realización de varios códigos en Lenguaje *Python*.
- 2 Repetición de la ejecución del código.
- 3 Aproximación  $f(x) = \sin(x)$  mediante el método de Taylor.
- 4 Error estimado.

## Descripción:

- 1 Realización de varios códigos en Lenguaje *Python*.
- 2 Repetición de la ejecución del código.
- 3 Aproximación  $f(x) = \sin(x)$  mediante el método de Taylor.
- 4 Error estimado.
- 5 Tiempo CPU.

El material requerido para la realización del trabajo ha sido una computadora.

## Características de la computadora:

- CPU type: Intel(R) Core(TM) i3-2328M CPU @ 2.50GHz



El material requerido para la realización del trabajo ha sido una computadora.

## Características de la computadora:

- CPU type: Intel(R) Core(TM) i3-2328M CPU @ 2.50GHz
- vendor ID GenuineIntel

El material requerido para la realización del trabajo ha sido una computadora.

## Características de la computadora:

- CPU type: Intel(R) Core(TM) i3-2328M CPU @ 2.50GHz
- vendor ID GenuineIntel
- CPU speed 1200.000Hz

El material requerido para la realización del trabajo ha sido una computadora.

## Características de la computadora:

- CPU type: Intel(R) Core(TM) i3-2328M CPU @ 2.50GHz
- vendor ID GenuineIntel
- CPU speed 1200.000Hz
- cache size 2048 KB

## Experimentos realizados:

- 1 Grado del polinomio y punto  $c$ , fijos. El punto  $x$  cambia.

## Experimentos realizados:

- 1 Grado del polinomio y punto  $c$ , fijos. El punto  $x$  cambia.
- 2 Grado del polinomio y punto  $x$ , fijos. El punto  $c$  cambia.

## Experimentos realizados:

- 1 Grado del polinomio y punto  $c$ , fijos. El punto  $x$  cambia.
- 2 Grado del polinomio y punto  $x$ , fijos. El punto  $c$  cambia.
- 3 Grado del polinomio toma el valor 10. Punto  $c$  y punto  $x$ , igual que el experimento 1.

## Experimentos realizados:

- 1 Grado del polinomio y punto  $c$ , fijos. El punto  $x$  cambia.
- 2 Grado del polinomio y punto  $x$ , fijos. El punto  $c$  cambia.
- 3 Grado del polinomio toma el valor 10. Punto  $c$  y punto  $x$ , igual que el experimento 1.
- 4 Grado del polinomio toma el valor de 100. El punto  $c$ , fijo. El punto  $x$  solo varia en las cantidad de cifras decimales.

# Algoritmo principal:

```
#!/src/bin/python
#encoding: UTF-8
import math
from sympy import *
import modulo
import time
import numpy as np
import matplotlib.pyplot as mp

numero = int(raw_input("Introduzca el grado que desea que tenga el Polinomio de Taylor: "))
centro = float(raw_input("Introduzca el punto central donde desea que se evalúe el Polinomio de Taylor: "))
x = float(raw_input("Introduzca el punto x donde desea evaluar el Polinomio de Taylor: "))

modulo.taylor1(x, numero, centro)
y1 = np.linspace(-np.pi, np.pi, 256, endpoint=True)
y1 = np.arange(-8,8,0.1)
y2 = np.sin(y1)

mp.plot(y1,y2, color = 'blue',linewidth=2.5, linestyle="-", label="seno")
mp.legend(loc=0)

# Para poner el simbolo pi en el eje x se usa LaTeX.
mp.xticks([-2*np.pi, -3*np.pi/2, -np.pi, -np.pi/2, 0, np.pi/2, np.pi, 3*np.pi/2, 2*np.pi],
[r'$2\pi$',r'$-\frac{3\pi}{2}$',r'$-\pi$',r'$-\frac{\pi}{2}$',r'$0$',r'$+\frac{\pi}{2}$',r'$+\pi$',
r'$\frac{3\pi}{2}$',r'$2\pi$'])

mp.title("Representacion grafica")

mp.savefig('grafico.png',dpi = 100)

mp.show()
```



# Algoritmo del módulo

```
#!/src/bin/python
import math
from sympy import *
import time

def factorial(numero):
    factorial = 1
    while(numero > 1):
        factorial = factorial * numero
        numero = numero - 1
    return factorial

def taylor1(x, numero, centro):
    inicio = time.time()
    c = Symbol('c')
    f = sin(c)
    func = f.evalf(subs={c: centro})
    suma = func
    for i in range(1, numero + 1):
        f_i = diff(f, c)
        v = f_i.evalf(subs={c: centro})
        s = (v / factorial(i)) * ((x - centro) ** i)
        suma = suma + s
        f = f_i
    error = func - suma
    print 'El valor de la funcion original sin(%f) es igual a %f ' % (centro, func)
    print 'El Polinomio de Taylor de grado n=%d en el punto centro c=%f'
    print 'evaluada en el punto x=%f es igual a %f' % (numero, centro, x, suma)
    print 'El Error de la funcion original con el Polinomio de Taylor es: error=%f' % error
    l=[numero, centro, x, suma, error]
    f = open("Taylor.tex", 'a')
```