



Universidad
de La Laguna

SERIES NUMÉRICAS

Función trigonométrica: $\sin(x)$

Jorge Antonio Herrera Alonso

Elizabeth Hernández Martín

Yessica Sabrina Gómez Buso

Grupo (2 | F)

Técnicas Experimentales. 1^{er} curso. 2^{do} semestre

Lenguajes y Sistemas Informáticos

Facultad de Matemáticas

Universidad de La Laguna

La Laguna, 10 de mayo de 2014

Índice general

1. Resumen	1
2. Motivación y objetivos	2
2.1. Objetivo principal:	2
2.2. Objetivo específico:	2
3. Fundamentos teóricos	3
3.1. Historia	3
3.2. Cálculo de la serie de Taylor	3
4. Procedimiento experimental	5
4.1. Descripción de los experimentos	5
4.2. Descripción del material	5
4.3. Resultados obtenidos	6
4.4. Análisis de los resultados	7
5. Conclusiones	8
A. Programa en Python	9
A.1. Algoritmo principal	9
A.2. Algoritmo del módulo	9
B. Explicación del programa en Python	11
B.1. Algoritmo	11
B.2. Módulo	11
Bibliografía	12

Índice de figuras

4.1. Gráfica de la función original	6
---	---

Índice de cuadros

4.1. Tabla de datos obtenidos. Experimento 1	6
4.2. Tabla de datos obtenidos. Experimento 2	6
4.3. Tabla de datos obtenidos. Experimento 3	7
4.4. Tabla de datos obtenidos. Experimento 4	7

Capítulo 1

Resumen

En este trabajo se estudiará el polinomio de Taylor para la función $\sin(x)$. Para ello se han realizados algortimos en el lenguaje python y así poder calcular las aproximaciones. Se analizarán los resultados y se graficarán.

Capítulo 2

Motivación y objetivos

A lo largo de este curso hemos aprendido a implementar diferentes códigos en *Python*, los cuales han logrado generar nuestra curiosidad por saber más. Esto nos permitió ir más allá y poder fusionar dicho lenguaje de programación con el procesador de texto \LaTeX [2] y una clase de este, el *Bearmer*, que utilizamos para realizar presentaciones. A partir de todos ellos, hemos conseguido llevar a cabo esta memoria.

2.1. Objetivo principal:

Profundizar nuestros conocimientos con el lenguaje de programación *Python*, el procesador de texto \LaTeX y el creador de presentaciones *Bearmer* sobre el estudio de las series de Taylor.

2.2. Objetivo específico:

Hallar la aproximación de $f(x) = \sin(x)$ mediante el método de Taylor, el error cometido y el estudio del tiempo de programación.

Capítulo 3

Fundamentos teóricos

3.1. Historia

Brook Taylor [1] nació el 18 de agosto de 1685 en Edmonton. Hijo de John Taylor, del Parlamento de Bifrons, Kent, y de Olivia Tempest (hija de Sir Nicholas Tempest).

En "Los métodos de incrementación directa e inversa" de Taylor (1715) agregaba a las matemáticas una nueva rama llamada ahora «El cálculo de las diferencias finitas», e inventó la integración por partes y descubrió la célebre fórmula conocida como la Serie de Taylor, la importancia de esta fórmula no fue reconocida hasta 1772, cuando Lagrange proclamó los principios básicos del Cálculo Diferencial. Taylor también desarrolló los principios fundamentales de la perspectiva en "Perspectivas Lineales" (1715). En su *Methodus Incrementorum Directa et Inversa* (Londres, 1715) desarrolló una nueva parte dentro de la investigación matemática, que hoy se llama cálculo de las diferencias finitas. Junto con "Los nuevos principios de la perspectiva lineal". Taylor da cuenta de un experimento para descubrir las leyes de la atracción magnética (1715) y un método no probado para aproximar las raíces de una ecuación dando un método nuevo para logaritmos computacionales (1717).

Brook Taylor murió en Somerset House el 29 de diciembre de 1731.

3.2. Cálculo de la serie de Taylor

Sea $f(x)$ una función definida en un intervalo que contiene al punto a , con derivadas en todos los órdenes.

El polinomio de primer grado

$$p_1(x) = f(a) + f'(a)(x - a)$$

tiene el mismo valor que $f(x)$ en el punto $x = a$ y también, como se comprueba fácilmente, la misma derivada que $f(x)$ en este punto. Su gráfica es una recta tangente a la gráfica de $f(x)$ en el punto a .

Es posible elegir un polinomio de segundo grado,

$$p_2(x) = f(a) + f'(a)(x - a) + \frac{1}{2}f''(a)(x - a)^2$$

tal que en el punto $x = a$ tenga el mismo valor que $f(x)$ y también valores iguales para su primera y segunda derivada. Su gráfica en el punto a se acercará a la de $f(x)$ más que la anterior. Es natural esperar que si construimos un polinomio que en $x = a$ tenga las mismas n primeras derivadas que $f(x)$ en el mismo punto, este polinomio se aproximará más a $f(x)$ en los puntos x próximos a a . Así obtenemos la siguiente igualdad aproximada, que es la fórmula de Taylor:

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

El segundo miembro de esta fórmula es un polinomio de grado n en $(x - a)$. Para cada valor de x puede calcularse el valor de este polinomio si se conocen los valores de $f(a)$ y de sus n primeras derivadas.

Para el caso de la función $\sin(x)$ el Polinomio de Taylor sería de la siguiente forma:

$$f(x) = \sin(a) + \cos(a)(x - a) - \frac{1}{2!}\sin(a)(x - a)^2 - \frac{1}{3!}\cos(a)(x - a)^3 + \\ \frac{1}{4!}\sin(a)(x - a)^4 + \dots + \frac{1}{n!}f^n(a)(x - a)^n$$

Capítulo 4

Procedimiento experimental

4.1. Descripción de los experimentos

El experimento llevado a cabo en esta memoria ha consistido en la realización de varios códigos en lenguaje *Python*. Los algoritmos implementados que solucionan dichos códigos estiman la aproximación $f(x) = \sin(x)$ mediante el método de Taylor, solicitando el grado del polinomio de Taylor, el punto central y el punto x donde se evalúa dicho polinomio. Para obtener varios valores del polinomio y así poder comparar los datos obtenidos del error y el tiempo de CPU, se ha ejecutado el programa varias veces:

- Experimento 1: El grado del polinomio y el punto c se dejaron fijos, variando solamente el punto x .
- Experimento 2: El grado del polinomio tiene el mismo valor que en el experimento anterior, el valor de x no varía y el que lo hace es el punto c .
- Experimento 3: El grado del polinomio es fijo con un valor mayor al de los anteriores experimentos. El punto c es fijo y se le asignó el mismo valor que en el experimento 1. El punto x va adquiriendo los mismos valores que en el experimento 1.
- Experimento 4: El grado del polinomio toma el valor de 100. El punto c no cambia y el punto x solo varía en la cantidad de cifras decimales.

4.2. Descripción del material

El material requerido para la realización del trabajo ha sido una computadora. La que hemos utilizado para realizar estos experimentos tiene las siguientes características:

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

4.3. Resultados obtenidos

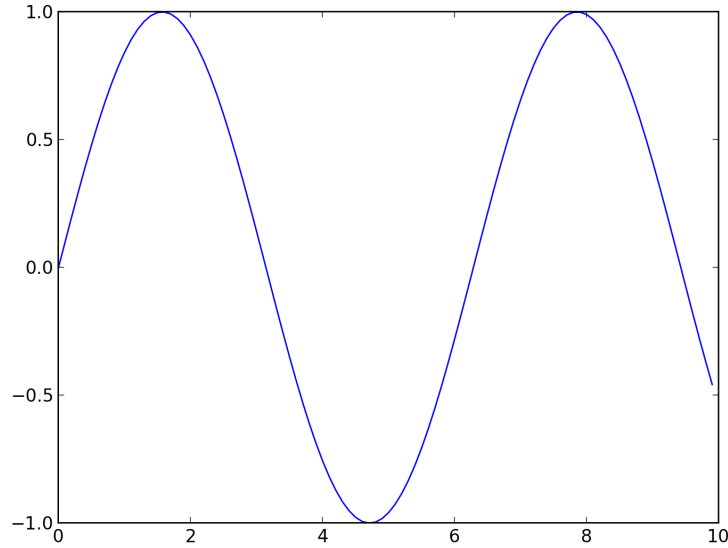


Figura 4.1: Gráfica de la función original

Grado	Punto c	Punto x	Aproximacion	Error	Tiempo CPU
4	10.0	-10.0	-4620.52781304922	4619.98379193833	0.00790095329284668
4	10.0	-5.0	-1545.27783908746	1545.73381797657	0.00798487663269043
4	10.0	5.0	-21.1962728645601	20.6522517536707	0.010929107666015625
4	10.0	9.0	0.404548172498635	-0.948569283388005	0.008239030838012695
4	10.0	9.9	-0.457535964436753	-0.086485146452617	0.00795292854309082
4	10.0	10.0	-0.544021110889370	0	0.008179903030395508

Cuadro 4.1: Tabla de datos obtenidos. Experimento 1

Grado	Punto c	Punto x	Aproximacion	Error	Tiempo CPU
4	-7.0	2.0	-238.466744388979	237.809757790260	0.008273124694824219
4	-2.0	2.0	-0.559778321379882	-0.349519105445799	0.008683919906616211
4	0.0	2.0	0.666666666666667	-0.666666666666667	0.007463932037353516
4	2.0	2.0	0.909297426825682	0	0.007899999618530273
4	7.0	2.0	21.4904658168047	-20.8334792180859	0.007979869842529297

Cuadro 4.2: Tabla de datos obtenidos. Experimento 2

Grado	Punto c	Punto x	Aproximacion	Error	Tiempo CPU
10	10.0	-10.0	2226636.13194962	-2226636.67597073	0.012552976608276367
10	10.0	-5.0	124685.902313545	-124686.446334656	0.011905908584594727
10	10.0	0.0	1920.68755204547	-1921.23157315635	0.011946916580200195
10	10.0	5.0	0.163743739637233	0.707764850526603	0.08893513679504395
10	10.0	9.0	0.412118507257685	-0.956139618147055	0.014183998107910156
10	10.0	9.9	-0.457535893775321	0.0864852171140484	0.012012958526611328
10	10.0	10.0	-0.544021110889370	0	0.014500856399536133

Cuadro 4.3: Tabla de datos obtenidos. Experimento 3

Grado	Punto c	Punto x	Aproximacion	Error	Tiempo CPU
100	-1.0	-0.99999	-0.8414655817427645	-5.40306513208133e	0.06290507316589355
100	-1.0	-0.9	-0.783326909627484	-0.0581440751804130	0.06340909004211426

Cuadro 4.4: Tabla de datos obtenidos. Experimento 4

4.4. Análisis de los resultados

- Experimento 1: Polinomio de Taylor de grado 4, punto $c = 10,0$. El punto de evaluación fue cambiando. A medida que se acercaba al punto c , el error fue disminuyendo. La diferencia del tiempo de CPU es mínima.
- Experimento 2: Polinomio de Taylor de grado 4, punto $x = 2,0$. El punto c fue cambiando. Cuando el punto c es 0, tanto el valor de aproximación como el de error son iguales pero de signos diferentes. Si el valor otorgado a c es negativo el error que se obtiene es mucho mas grande que si el valor de c es positivo.
- Experimento 3: Polinomio de Taylor de grado 10, punto $c = 10,0$. El punto x que está más alejado del punto c contiene el mayor valor de error. En $x = 5$ el tiempo de CPU es mayor que el resto.
- Experimento 4: Polinomio de Taylor de grado 100, punto $c = -1,0$. Se ha cambiado sólo el número de los decimales de x , el error para $x = -0,9$ es menor que el de para $x = -0,999999$. El tiempo de CPU que se registró para ambos valores de x contienen una diferencia mínima.

Capítulo 5

Conclusiones

Se han realizado varias pruebas con el Polinomio de Taylor, dándoles distintos valores tanto al grado, como al punto central c y al punto x .

- Observando los datos obtenidos, se registró una mayor margen de error cuando la distancia entre el punto c y el punto x se hacía mayor, a medida que nos acercamos al punto evaluado, el error disminuye consideradamente.
- Cuando los valores de c y x son iguales el error es nulo. Esto se debe a que, el polinomio de Taylor apartir de la primera derivada de la función estudiada, cada termino se multiplica por la diferencia entre el punto x y el punto c .
- En el experimento 3 se le volvió a dar los mismo valores que en el experimento 1, solo aumentó el grado del polinomio. Comparando ambos resultados, se observa que el error en el caso 3 es mayor para los mismos valores dados. Esto significa que cuanto más grande sea el grado del polinomio, obtendremos una mejor aproximación de la función estudiada.
- En el último experimento realizado se observa que al agregar más decimales al punto x el error aumenta.
- Cuando se le asigna un mayor grado al polinomio de Taylor, el tiempo de CPU en calcularlo va aumentando. Aunque no se ha llegado podido registrar que demore 1s. El tiempo que tarda es practicamente insignificante, como la diferencia que existe entre el polinomio de grado 4 y el de grado 100.

Apéndice A

Programa en Python

A.1. Algoritmo principal

PROGRAMA 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.arange(0,10,0.1)
y2 = np.sin(y1)

mp.plot(y1,y2)
mp.savefig('grafico.png',dpi = 200)
mp.show()
```

A.2. 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
    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')

    f.write('Grado del Polinomio (n) Punto Central (c) Punto de Evaluacion (x)
    Aproximacion
    Error Tiempo CPU \n ')
    fin = time.time()
    tiempo_total = fin - inicio
    l=l+[tiempo_total]
    f.write(str(l))
    f.write("\n")
    f.close()
    f=open("Taylor.tex","r")
    print(f.read())
    f.close()
    return taylor1

```

Apéndice B

Explicación del programa en Python

B.1. Algoritmo

En el programa principal primero, importamos las librerías que usaremos más adelante con el comando `import` o `from xxx import yyy` para importar un elemento concreto de dicha librería. Luego, pedimos por pantalla las 3 variables necesarias para ejecutar nuestro programa que son:

el grado del polinomio de Taylor (`numero`), el punto central (`centro`) y el punto donde se desea evaluar (`x`). Finalmente, acabamos el programa llamando a la función `taylor1` que se encuentra en nuestro módulo.

Por último, definimos `y1` e `y2` para que se evalúe la función `y2` en el rango de la variable `y1` y luego que se cree y guarde en el archivo `grafico.png`.

B.2. Modulo

Al igual que en el programa principal, importaremos las librerías que utilizaremos posteriormente. Primeramente, definimos la función `factorial`, que luego será usada en la otra función para calcular el polinomio de Taylor. `factorial(numero)` crea un bucle `for` que incremente su valor, dependiendo de la variable `numero` introducida.

La siguiente función `taylor1` dependerá de las tres variables que se han introducido por pantalla.

Primero, declaramos una variable `c` sobre la que se derivará luego la función `f` respecto de ella

Evalúamos la función f en el punto $c = \text{centro}$ y con ese valor inicializamos la variable `suma`. Luego, entramos en un bucle `for`, en el que se vaya derivando la función f y así, se vaya evaluando en el punto $c = \text{centro}$ para calcular una nueva variable s , propia del polinomio de Taylor, y así ir incrementando la variable `suma`.

Ahora calculamos el error cometido entre la función original y la aproximación del polinomio de Taylor. Seguidamente, mostramos por pantalla los datos obtenidos en el programa y creamos una lista `l` que luego usaremos para escribir en un fichero.

Lo siguiente será parar el cronómetro del programa y calcular el tiempo total que ha tardado en realizarse.

Por último, abrimos un fichero de nombre `Taylor.tex` que actualice lo que hemos obtenido como resultados, si no se encuentra ya en el fichero ("`a`"), luego escribimos un encabezado, para saber que significa cada dato que luego escribiremos en una única línea con `f.write(str(l))`.

Finalizamos el módulo con un `f.close()` para cerrar el fichero y mostramos por pantalla dicho fichero para comprobar los resultados.

Bibliografía

- [1] [http://www.buscabiografias.com/bios/biografia/verDetalle/2155/Brook Taylor](http://www.buscabiografias.com/bios/biografia/verDetalle/2155/Brook%20Taylor).
- [2] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison–Wesley Pub. Co., Reading, MA, 1986.
- [3] Coromoto León. *Diseño e implementación de lenguajes orientados al modelo PRAM*. PhD thesis, 1996.
- [4] Guido Rossum. Python library reference. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.
- [5] Guido Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.
- [6] ACM LaTeX Style. http://www.acm.org/publications/latex_style/.