Series de Taylor en arcsin(x)

Ana Gómez Pérez, Sara Luis Farreis y Shaila Verona Rodríguez

Grupo 2J

Técnicas Experimentales. 1er *curso.* 2do *semestre*

Lenguajes y Sistemas Informáticos

Facultad de Matemáticas

Universidad de La Laguna

La Laguna, 11 de mayo de 2014

Resumen

En este trabajo se mostrarán las características del método de *Taylor*, además de ponerla en práctica con una función previamente dada, dando los resultados y del posible error que pueda tener. Además de esto también se detallará dicho métodod. Por otro lado, se hará un programa en `Python`, el cual nos permitirá observar los valores de forma más breve y clara.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Motivación y objetivos

1.1. Motivación

Este proyecto o trabajo de investigación se realizó con la finalidad de crear un programa en Python en el que se calcula por el método de Taylor la función de arcseno(x).Además, proponerlo en un informe de tipo LaTeX [2], y posteriormente, una presentación en Beamer.

1.2. Objetivos

Los objetivos de esta practica es aprender a manejar tanto el Python, el LaTeX y el Beamer:

- Pythones un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible.

  Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

- LaTeX es un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas.

- Beamer es una clase de LaTeX para la creación de presentaciones, funciona con pdflatex.

Capítulo 2

Fundamentos teóricos

2.1. El por qué de el método de Taylor

La función $p(x) = a_0+a_1x+a_2x^2+...+a_nx^n$, en la que los coeficientes a_k son constantes, se llama polinomio de grado n.En particular $y = ax + b$ es un polinomio de grado 1, de los más sencillos, por lo que calcular su valor en fácil. Sin embargo, calcular el valor par a otras funciones como log(x), sen(x), e^X, ... es mucho más complicado. Por tanto, se utilizan metodos desarrollados por el análisis matemático, como el método de Taylor.

2.2. Fórmula de Taylor

Para poder usar este método deben cumplirse dos condiciones:

- Sea f(x) una función continua en [a,b]
- Sea f(x) derivable en (a,b)

Cuando tengamos un polinomio de primer grado $p_1(x) = f'(a)(x - a)$ tendrá el mismo valor que f(x) en el punto x=a. Dando la gráfica es una recta tangente a la gráfica de f(x) en el punto a.

Es pol¡sible 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 valores también iguales para su primera y segunda derivadas.Se gráfica en el punto a se acercará a la de f(x) más que la anterior. Es natural esperar que si contruimos 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:

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

Sin embargo, esto solo se da para polinomios que tengan su derivada hasta n, mientras que para los polinomios que tienen derivada (n+1)-ésima difieren de f(x) en una pequeña cantidad, que denominamos como el error.

Por ello añadimos un término más, llamado resto, para que el error sea menor:

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

Capítulo 3

Procedimiento experimental

3.1. Descripción de los experimentos

A continuación expondremos los pasos que se han seguido en la elaboración del experimento desarrollado para este trabajo de investigación. Nos apoyaremos en gráficos y tablas que les ayudaran a reforzar y aclarar la información desarrollada.

3.1.1. Descripción de los experimentos

Para llevar a cabo el método de *Taylor*, objetivo principal del informe, se ha empleado la ecuación del método de *Taylor*. Recodar que la función derivada ha sido $f(x) = arcsin(x)$ y que al aplicar la serie de Taylor hemos tenido que calcular hasta la derivada n-ésima y, además, elos factoriales de 1 hasta n.

En relación a la eficiencia del proyecto, se ha analizado el resultado obtenido de la aproximación de Taylor midiendo el error de este con el resultado original de la función.

3.2. Descripción del material

El material utilizado ha sido el siguiente:

- Tipo de CPU:

  Intel(R) Core(TM) i3-2328M CPU @ 2.20GHz

- Tamaño de la memoria del procesador:

  3072 KB

- Vendedor GenuineIntel:

  Linux

- Sistema operativo:

  66-Ubuntu SMP

- <u>Plataforma</u>:

  Linux-3.2.0-59-generic-pae-i686-with-Ubuntu-12.04-precise

- <u>Version</u>:

  2.7.3

3.3. Resultados obtenidos

Como resultados de haber calculado en el algoritmo para cualquiero polinomio obetemos el valor del polinomio con el método de Taylor, el error y el tiempo[1] que tarda el algoritmo en calcularlos.

[1] En segundos

X	a	tiempo	Error
-1	1	0.0295159816742	4.51842232203082e+220
-0.25	1	0.0272219181061	1.76500872128679e+220
-0.2	1	0.0301878452301	1.62663203764503e+220
0	1	0.0254921913147	1.12960558199550e+220
0.2	1	0.0249121189117	7.22947572667555e+219
0.25	1	0.0299370288849	6.35403140081687e+219
1	1	0.0301327705383	0

Cuadro 3.1: Experimentos en el algoritmo

Como ejemplos de los tiempos y errores en diferentes puntos de un polinomio de grado 3, obtenemos representado en la tabla anterior.

Por cada dato que calculamos obtemos la función representada en una gráfica en dicho punto, como por ejemplo esta gráfica:

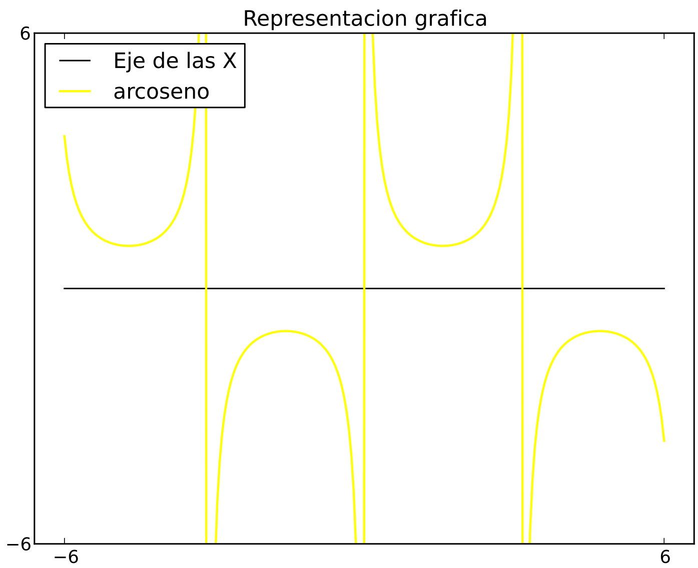

Figura 3.1: Gráfica de la función arcsin(x)

3.4. Análisis de los resultados

Obsevando detenidamente los resultados obetenidos en la tabla anterior, podemos apreciar que cuando más se parecen el valor del centro(a) y el puento X de la función, disminuye el valor del error entre el polinomio y la aproximación del polinomio. Por otro lado, cuantos más dispares son dichos valores mayor es el error entre el polinomio y el polinomio de aproximación.

Esto se da en los ejemplos propuestos en la tabla, en donde utilizamos un polinomio de grado 3 para todos los casos. Observamos que en primer caso, el valor del centro es 1 y el de la x es -1, el caso más dispar que puede darse, se obtiene el error más alto. Mientras, en el último caso, donde el valor del centro es 1 y de x es 1 se obtiene un valor 0 del error ya que el valor del centro y el punto X son iguales.

Capítulo 4

Conclusiones

Como conclusion podemos sacar de este trabajo o informe que usar el metodo de *Taylor* es util para una funcion cuyo calculo seria muy dificil tratandolo con un polinomio simple. Por tanto, este metodo es muy eficiente para funciones complicadas de calcular, aunque halla un minimo error entre la funcion original y la calculada por Taylor.

Segun los resultados obtenidos de las pruebas realizadas, podemos ver que a medida que el centro sea va acercando a el valor de X el tanto por ciento de error va disminuyendo. Cabe destacar que la interpolacion en grado dos y en grado tres son similares.

Tambien cabe destacar, uno de los objetivos principales de la asignatura que era reforzar los conocimiento de LaTeX y Beamer para el trabajo. Ademas, todo esto nos ha proporcionado conocimientos que nos sera util para trabajos futuros como el proyecto final de grado.

Hemos de decir que a pesar de la dificultad que nos ha presentado la realizacion de este informe, nos encontramos satisfechos con los resultados obtenidos.

Apéndice A

Título del Apéndice 1

A.1. Algoritmo

```
/###################################################################################
 # solucion.py
 ###################################################################################

  Ana Gomez
  9/05/2014

#! /src/bin/python
#!encoding: UTF-8

import math
from sympy import *
import time
import matplotlib.pylab as pl
import numpy as np


def factorial(n):
   if n <= 1:
     return 1
   else:
     prod = n*factorial(n-1)
     return prod

def taylor(n,x,a):
  c = Symbol('c')
  funcion = asin(c)
  suma=funcion.evalf(subs={c:a})
  for i in range (1,n+1):
    derv = diff(funcion, c)
    termino = derv.evalf(subs={c:a})
    resultado = (termino/factorial(i))*((x-a)**i)
    suma = suma + resultado
    funcion = derv
  return suma
```

```
if __name__ == "__main__":

  n = int(raw_input("Introduzca el grado del polinomio:"))
  x = float(raw_input("Introduzca el punto donde se evalua el polinomio:"))
  a = float(raw_input("Introduzca el punto central donde se desea evaluar el polinomio:"))
  if (abs(a)>1)or(abs(x)>1):
    print 'Debe introducir valores de a entre [-1,1]'
    a = float(raw_input("Introduzca el punto central donde se desea evaluar el polinomio:"))
    x = float(raw_input("Introduzca el punto donde se evalua el polinomio:"))

start=time.time()
suma = taylor(n,x,a)
finish=time.time()-start
error = abs(asin(x)- suma)
print 'Valor de la aproximacion'
print suma
print 'Valor del error'
print error
print 'Tiempo que tarda el programa en ejecutarse'
print finish
 ####################################################################################
```

A.2. Representacion de funcion

```
/####################################################################################
 # solucion.py
 ####################################################################################
 Ana Gomez
 9/05/2014

g=int(raw_input('Intervalo para el eje de las X: '))
h=int(raw_input('Intervalo para el eje de las Y: '))
l=[]
for i in range (g):
  y=1/np.sin(x)
  l.append(y)

pl.figure(figsize=(8,6), dpi=80)

pl.subplot(1,1,1)

X = np.linspace(-g, g, 256, endpoint=True)
C = 0*(X)
S = 1/np.sin(X)

pl.plot(X,C, color="black", linewidth=1.0, linestyle="-", label="Eje de las X")
pl.plot(X,S, color="yellow", linewidth=1.5, linestyle="-", label="arcoseno")

pl.legend(loc='upper left')

#pl.xlim(-4.0,4.0)
pl.xlim(X.min()*1.1,X.max()*1.1)
```

```
#pl.xticks(np.linspace(-4,4,9,endpoint=True))
pl.xticks([-g, g])

#pl.ylim(-1.0,1.0)
pl.ylim(C.min()*1.1,C.max()*1.1)

#pl.yticks(np.linspace(-1,1,5,endpoint=True))
pl.yticks([-h, h])

pl.title("Representacion grafica")

pl.savefig("grafica.eps", dpi=72)

pl.show()
 ###################################################################################
```

Apéndice B

Título del Apéndice 2

B.1. Otro apendice: Algoritmo

En la primera parte de algoritmo, hemos importado las diferentes librerias que necesitamos.
Luego declaramos dos funciones: El la primera calculamos el factorial, que necesitamos para calcular la formula de Taylor, donde podemos ver que solo usamos dos simples condiciones, y despues declaramos la funcion Taylor, para calcularla.

En esta segunda funcion, declaramos un variacle ('c') general que usamos cuando evaluamos la funcion en el centro(a). Para luego usarlo en un bucle "for" donde calcula la deriva de la funcion, la evalua en el punto centro(a) y opera, invocando a la funcion factorial, obteniendo la aproximacion de funcion. Por ultimo, se iguala la funcion a la deriva para que asÃ siga el bucle tantas veces como el valor del grado(n).

Y como resultado se devuelve el resultado de la suma con un "return suma".

Entonces en esta segunda parte, se piden los valores del grado(n), el punto(x) y el centro(a), aunque se vuelven a pedir los valores de x y a si no estan entre el intervalos [-1,1].Depues, se declara una variable inicial para empezar a medir el tiempo que tarda el programa en ejecutarse. Se invoca a la funcion principal, taylor, y se calcula con los valores que se hallan introducido. Y se declara otra variable final en la que se obtiene el valor del tiempo que ha tardado.

Luego, se calcula el error con un valor absoluto(para que no de valores negativos) entre el valor exacto de la funcion menos la aproximacion de dicha funcion. Finalmente, se imprimen por pantalla todos los valores

B.2. Otro apendice: Representacion de la funcion

En la ultima parte del programa, ya que hemos realizado todo lo que nos pedia en el mismo algoritmo, representamos la funcion pidiendo valores para ambos ejes e iniciamos una lista para guardar los valores que se van tomando del "for". Proponemos las medidas de la grafica y demas detalles, como los colores, la anchura de las lines, ect. Declaramos entre que intervalos debe de darse la grafica y la llamamos "Representacion grafica". Por ultimo, la guardamos en un fichero.

Bibliografía

[1] Beamer. http://es.wikipedia.org/wiki/Beamer.

[2] Manual de latex. http://es.wikipedia.org/wiki/LaTeX.

[3] Manual de python. http://es.wikipedia.org/wiki/Python.

[4] Mariano Banzo Marraco. Series de taylor
. http://recursostic.educacion.es/descartes/web/materialesdidacticos/Desarrolloserietaylor.htm.