Universidad
de La Laguna

Series de potencias: Newton
f(x)= sin(x)

Zoilo González Garcia

Francisco Javier Reyes Sánchez

Grupo (2^oE)

Técnicas Experimentales. 1^{er} *curso.* 2^{do} *cuatrimestre*

Lenguajes y Sistemas Informáticos

Facultad de Matemáticas

Universidad de La Laguna

La Laguna, 9 de mayo de 2014

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Motivación y objetivos

Los objetivos para los que se plantea este trabajo, son el adquirir conocimientos y mejorar nuestras habilidades en el uso del lenguaje de programanción PYTHON, procesador de texto LaTeX y una clase de LaTeX que nos permite diseñar presentaciones, BEAMER. Además, desde un punto de vista matemático, aprenderemos el método de Interpolación Polinómica de Newton para la aproximación de una función en un intervalo determinado, haciendo uso de las diferencias divididas de Newton.

1.1. Sección Uno: LaTeX

Es un sistema de composición muy adecuado para realizar documentos científicos y matemáticos de alta calidad tipográfica. Es también adecuado para producir documentos de cualquier otro tipo, desde simples cartas a libros enteros.

LaTeX está formado mayoritariamente por órdenes construidas a partir de comandos de TeX (lenguaje de nivel bajo), en el sentido de que sus acciones son muy elementales, pero con la ventaja añadida de poder aumentar las capacidades de LaTeX utilizando comandos propios del TeX descritos en The TeXbook.3 4. Esto es lo que convierte a LaTeX en una herramienta práctica y útil pues, a su facilidad de uso, se une toda la potencia de TeX. Estas características hicieron que LaTeX se extendiese rápidamente entre un amplio sector científico y técnico, hasta el punto de convertirse en uso obligado en comunicaciones y congresos, y requerido por determinadas revistas a la hora de entregar artículos académicos.

1.2. Sección Dos: BEAMER

El nombre viene del vocablo alemán "beamer", un pseudo-anglicismo que significa videoproyector. BEAMER es una clase de LaTeX para la creación de presentaciones. Funciona con pdflatex, dvips y LyX.

Al estar basado en LaTeX, Beamer es especialmente útil para preparar presentaciones en las que es necesario mostrar gran cantidad de expresiones matemáticas, el fuerte de dicho sistema de maquetación.

Capítulo 2

Fundamentos teóricos

2.1. Interpolación Polinómica

En análisis numérico, la interpolación polinomial es una técnica de interpolación de un conjunto de datos o de una función por un polinomio. Es decir, asumimos que sólo se conoce la imagen de una función en un número finito de abscisas. En muchos de los casos, ni siquiera se conocerá la expresión de la función.

El objetivo de esta técnica es el de hallar un polinomio que tome los valores antes mencionados y que permita hallar aproximaciones de valores desconocidos para la función. Para segurar la precisión del polinomio se dispondrá de una fórmula del error de interpolación que permitirá ajustarlo.

2.2. Cálculo del polinomio interpolador de Newton

Existen varios métodos generales de interpolación polinómica [8] que permiten aproximar una función por medio de un polinomio de grado m. En este informe, se recogerá exclusivamente el método de las diferencias divididas de Newton.

Definición: Sea f_n una variable discreta de n elementos y sea x_n otra variable discreta de n elementos los cuales corresponden a la imagen y la abcisa de los datos que se quieran interpolar:

$$f(x_k) = f_k, \quad k = 1, ..., n$$

Una gran ventaja sobre la forma clásica de Lagrange es que podemos agregar un mayor número de nodos a la tabla de datos, lo que nos facilitará en gran medida el cálculo del polinomio, sobretodo, en aquellos casos en los que el grado del polinomio que se quiere calcular es bastante elevado.

Pongamos un ejemplo: el polinomio de grado n-1 resultante de aplicar este método, tendrá la forma:

$$\sum_{i=0}^{n-1} a_j g_j(x)$$

Donde:

$$g_j = \prod_{j=1}^{j-1} (x - x_i)$$

$$a_j = f[x_0, x_1, ..., x_{j-1}, x_j,]$$

Los coeficientes a_j son las llamadas diferencias divididas. El cálculo de estas diferencias es el paso que caracteriza este método. Tenemos que las diferencias divididas serían de la forma siguiente:

i	x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, x_{i+1}, x_{i+2}, x_{i+3}]$
0	x_0	$f[x_0]$			
					$f[x_0, x_1]$
1	x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
			$f[x_1, x_2]$		$f[x_0, x_1 x_2, x_3]$
2	x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
			$f[x_2, x_3]$		
3	x_3	$f[x_3]$			

Cuadro 2.1: Tabla de diferencias divididas

Capítulo 3

Procedimiento experimental

3.1. Descripción de los experimentos

El experimento ha consistido en la implemenación en Python de algoritmos capaces de resolver el problema en cuestión. Para ello se ha creado una función cuya labor es calcular el seno dada cualquier x, otra que calcula las diferencias divididas a partir de una serie de nodos, una tercera que nos calcula el polinomio resultante que aproxima la función y por último una funcción de prueba donde se dan unos valores comprendidos en el intervalo de aproximacíon y compara su seno con el valor propuesto por nuestro polinomio.

3.2. Descripción del material

Para la realización de este experimento el material necesario ha sido el siguiente:

- El ordenador usado en la realizacción de dicho experimento ha sido un portátil de la marca hp con un procesador Intel Atom$_{\text{inside}}$. Este consta de 231,9 Gb de disco duro y 2 Gb de memoria RAM.

- El sistema operativo con el que hemos trabajado es una adaptación de la distribución de Linux *kubuntu* a las necesidades en cuanto a Software de los miembros de la comunidad universitaria ULL: Bardinux [1].

- También hemos hecho uso de materiales tales como calculadora, folio y lápiz, pues han habido cuestiones que han sido necesarias resolver a mano para la posterior implementación del código.

3.3. Resultados obtenidos

Tras la implementación del código y la obtención del polinomio de aproximación hemos comprobado su precisión dando valores a las x y estos han sido resultados de algunos ejemplos:

Figura 3.1: Logo Bardinux

```
El valor aproximado por el polinomio para x=0.785398
y=0.683013
El seno de 0.785398 es
sen(x)=0.707107


El valor aproximado por el polinomio para x=0.628319
y=0.575349
El seno de 0.628319 es
sen(x)=0.587785


El valor aproximado por el polinomio para x=0.523599
y=0.500000
El seno de 0.523599 es
sen(x)=0.500000
```

Figura 3.2: Consola

3.4. Análisis de los resultados

A la hora de analizar la calidad de las aproximaciones dadas, se puede observar que, aunque buenas, estas no son del todo exactas. Desde un punto de vista teórico, si quisieramos unos mejores resultados en las aproximaciones, una solución efectiva sería seleccionar un mayor número de nodos. En nuestro caso el número de nodos en los que hemos dividido el intervalo $(0, \pi/2)$ es de 4: $(0, \pi/6, \pi/3, \pi/2)$; dejando entre cada uno de ellos un espacio equidistante. Otro punto de vista a la hora de analizar los datos que hay que tener en cuenta es la calidad del material utilizado, en nuestro caso, las diferentes características de la máquina de cómputo mencionadas anteriormente.

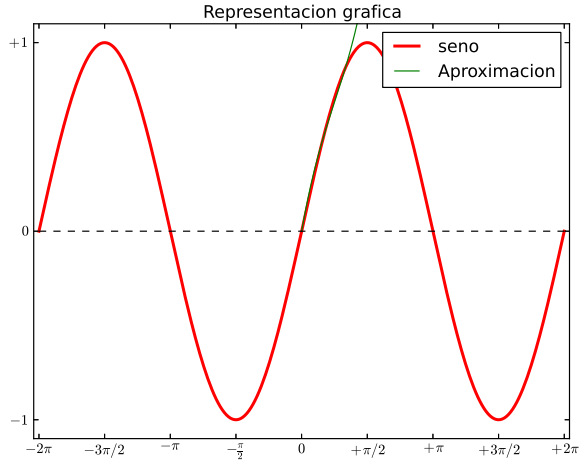

Figura 3.3: Seno con Matplotlib

Capítulo 4

Conclusiones

Para la conclusión trataremos dos punos de enfoque de nuestro trabajo, es decir, por un lado el aprendizaje teórico de las matemáticas y por otro lado su aplicación computacional.

4.1. Conclusiones de los aspectos teóricos

Hasta ahora, en lo que a teoría matemática y más concretamente a aproximación de funciones se refiere, solo habíamos tratado con aproximaciones en torno a un punto haciendo uso del polinomio de Taylor y acotado funciones en determinados intervalos haciendo uso de teoremas de continuidad pero nunca sin saber cual sería realmente dicha función ya que no se trataba de aproximarlas. Además se ha hecho uso del cálculo de diferencias divididas, que también es otro concepto que antes de la realización de este trabajo no nos era familiar. En general, la realización de este trabajo nos ha sido muy útil pues hemos obtenido nuevos conocimientos matemáticos de gran utilidad y considerados de un mayor nivel del que nos corresponde.

4.2. Conclusiones del aspecto computacional

Desde el punto de vista informático, esta tarea nos ha servido para adquirir destrezas en la programación en Python, el uso del sistema operativo Bardinux, diferentes editores de texto y en especial en la codificación en LaTeXy en Beamer de artículos y presentaciones. Esto nos podrá ayudar en un futuro a realizar informes y presentaciones de manera mas formal y profesional.

Apéndice A

Algoritmos empleados

A continuación se recogen los diferentes algoritmos empleados para el análisis y la obtención del polinomio interpolador de Newton, para la función . Además, se ha implementado un algoritmo de comprobación de los resultados.

A.1. Algoritmo sin(x)

```
################################################################################
# seno.py
################################################################################
##!/usr/bin/python
#!encoding: UTF-8
from math import pi
from math import sin
#definimos una función, con la que obtendremos el valor de los diferentes nodos
empleados, en nuestro caso la funcion es el seno(x)def seno(x):
  f=sin(x)
  return f

if __name__=="__main__":#Prueba para la funcion seno(x)
  n=int(raw_input("Introduzca un punto de prueba: "))
  print "El seno de dicha función es: %.3f" %seno(n)

# AUTORES:Zoilo González Garcia y Francisco Javier Reyes Sánchez
#
# DESCRIPCION: El codigo presentado anteriormente, nos permite calcular el valor de
la función seno(x),en los diferentes puntos, o nodos, que utilizaremos para el calculo
del polinomio interpolador de Newton.
#
################################################################################
```

A.2. Algoritmo Diferencias Divididas

```
################################################################################
# difdiv.py
################################################################################
```

```
#!/usr/bin/python
#!encoding: UTF-8
from math import pi
from math import sin
import seno
def difdiv(x):
  c=[]#lista que guardará cada una de las diferencias divididas.
  c=c+[0]#ya que la primera diferencia dividida es cero (f(0))
  c1=(seno.seno(x[1])-seno.seno(x[0]))/(x[1]-x[0])
  c=c+[c1]
  c12=(seno.seno(x[2])-seno.seno(x[1]))/(x[2]-x[1])#diferencia dividida entre x1 y x2
  c2=(c12-c1)/(x[2]-x[0])
  c=c+[c2]
  c23=((seno.seno(x[3])-seno.seno(x[1]))/(x[3]-x[2]))
  c13=((c23-c12)/(x[3]-x[1]))
  c3=((c13-c2)/(x[3]-x[0]))
  c=c+[c3]
  return c

if __name__=="__main__":#Prueba para la funcion difdiv(x)
  x=(0, pi/6, pi/3, pi/2)
  print difdiv(x)
 ##############################################################################
 #
 # AUTORES:Zoilo González Garcia y Francisco Javier Reyes Sánchez
 #
 # DESCRIPCION:El segundo código mostrado, ha sido creado con la intención de calcular las
 diferencias divididas de Newton, que utilizaremos para hallar el polinomio interpolador.
 Para ello importaremos los valores obtenidos en el primer código.
 #
 ##############################################################################
```

A.3. Algoritmo Interpolador de Newton

```
##############################################################################
# interpoladornewton.py
##############################################################################
#!/usr/bin/python
#!encoding: UTF-8
from math import pi
from math import sin
import difdiv
l=[]
s=[]
n=(0, pi/6, pi/3, pi/2)#tomamos cuatro nodos equidistantes en el intervalo (0,pi/2)
l=difdiv.difdiv(n)
print "El polinomio interpolador de Newton en el intervalo (0, pi/2), para los puntos
\nx0=0\nx1=pi/6\nx2=pi/3\nx3=pi/2\nes: "
print "P(x)= (%f) + (%f)*(x-%f) + (%f)*(x-%f)(x-%f) + (%f)*(x-%f)(x-%f)(x-%f)" %(l[0],
l[1], n[0], l[2], n[0], n[1], l[3], n[0], n[1], n[2])

# AUTORES:Zoilo González Garcia y Francisco Javier Reyes Sánchez
#
```

```
# DESCRIPCION: Este tercer código nos ayuda a mostar por pantalla el polinomio que buscamos,
para ello importamos la función anterior que nos calculaba las diferencias divididas de Newton.
#
##############################################################################################
```

A.4. Algoritmo para comprobacion

```
##############################################################################################
# comprobacion.py
##############################################################################################
#!/usr/bin/python
#!encoding: UTF-8
from math import pi
from math import sin
import seno
valores=(pi/4, pi/5, pi/6,2*pi/5)
for i in valores:
  print "El valor aproximado por el polinomio para x=%f es" %i
  y=(0.311104)*(i**3)+(-0.733021)*(i**2)+(1.253448)*i
  print "y=%f" %y
  print "El seno de %f es" %i
  f=seno.seno(i)
  print "sen(x)=%f" %f
  print "\n"


# AUTORES:Zoilo González Garcia y Francisco Javier Reyes Sánchez
#
# DESCRIPCION: Este código nos permite valorar el polinomio obtenido en el código
anterior. Para ello damos diferentes valores a la variable "x", y observamos el valor de
las aproximaciones realizadas.
#
##############################################################################################
```

A.5. Algoritmo de representción en Matplotlib

```
#############################################################
####representacionseno.py
#############################################################
#! encoding: utf8

import matplotlib.pylab as pl
import numpy as np

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

pl.subplot(1,1,1)

X = np.linspace(-np.pi*2, np.pi*2, 256, endpoint=True)
S = np.sin(X)
E = X*0
Y = np.linspace(0, np.pi/2, 150, endpoint=True)
```

```
A = ((0.311104)*(Y**3)+(-0.733021)*(Y**2)+(1.253448)*Y)

pl.plot(X,S, color="red", linewidth=2.5, linestyle="-", label="seno")
pl.plot(Y,A, color="green", label="Aproximacion")
pl.plot(X,E, color="black", linewidth=1, linestyle="--")

pl.legend(loc='0')

pl.xlim(X.min()*1.02,X.max()*1.02)
pl.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'$-3\pi/2$',r'$-\pi$',r'$-\frac{\pi}{2}$',r'$0$',r'$+\pi/2$',r'$+\pi$', r'$+3\pi/2$'
          )

pl.ylim(-1.1,1.1)
pl.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$']
         )

pl.title("Representacion grafica")

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

pl.show()

# AUTORES:Zoilo González Garcia y Francisco Javier Reyes Sánchez
#
# DESCRIPCION: En este caso, lo que se hace es representar gráficamente la función seno y el polinomio aprox
#
################################################################################
```

Apéndice B

Función seno

B.1. En trigonometría

En trigonometría el seno de un ángulo en un triángulo rectángulo se define como la razón entre el cateto opuesto y la hipotenusa:

$$f(x) = sin(x)$$

O también como la ordenada correspondiente a un punto que pertenece a una circunferencia unitaria centrada en el origen (c=1):

$$sin\alpha = a$$

En matemáticas el seno es la función continua y periódica obtenida al hacer variar la razón mencionada, siendo una de las funciones trascendentes.

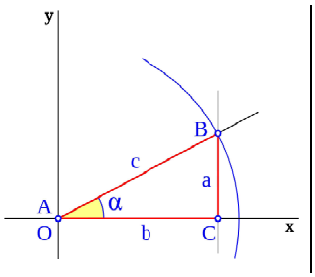

Figura B.1: FuncionSeno

B.2. El seno en programación

Antes de comenzar, se debe destacar que de forma habitual, todos los lenguajes de programación proveen una función seno, así como el resto de funciones trigonometrícas. También es normal que el ángulo que recibe la función deba pasarse en radianes.

Esto último es importante tenerlo en cuenta ya que podrían derivarse errores por este concepto. Del mismo modo las calculadoras suelen aceptar el valor en grados o radianes, siendo necesario para realizar el cálculo correctamente activar un botón selector del tipo de grados que se desea usar.

A continuación se muestra un ejemplo, obsérvese como la escasa diferencia entre ambos valores resultantes podría pasar desapercibida. Por ello, cuando sea conveniente se recomienda pasar los grados a radianes o viceversa.

```
ejemplos:
   seno de 45 grados   = 0,7071
   seno de 45 radianes = 0,8509
```

Figura B.2: EjemploSeno

Bibliografía

[1] Oficina de Software libre. bardinux.ull.es.

[2] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison–Wesley Pub. Co., Reading, MA, 1986.

[3] portales.puj.edu.co. http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo3/3.3.htm.

[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] Guido Rossum. Python tutorial. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[7] ACM LaTeX Style. http://www.acm.org/publications/latex_style/.

[8] Wikipedia.org. http://es.wikipedia.org/wiki/Interpolaci