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, 18 de abril 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

En este capítulo se han de presentar los antecedentes teóricos y prácticos que apoyan el tema objeto de la investigación.

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 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	$f[x_3]$			

Capítulo 3

Procedimiento experimental

Este capítulo ha de contar con seccciones para la descripción de los experimentos y del material. También debe haber una sección para los resultados obtenidos y una última de análisis de los resultados.

3.1. Descripción de los experimentos

bla, bla, etc.

3.2. Descripción del material

bla, bla, etc.

3.3. Resultados obtenidos

bla, bla, etc.

Tiempo (± 0.001 s)	**Velocidad (± 0.1 m/s)**
1.234	67.8
2.345	78.9
3.456	89.1
4.567	91.2

Cuadro 3.1: Resultados experimentales de tiempo (s) y velocidad (m/s)

3.4. Análisis de los resultados

bla, bla, etc.

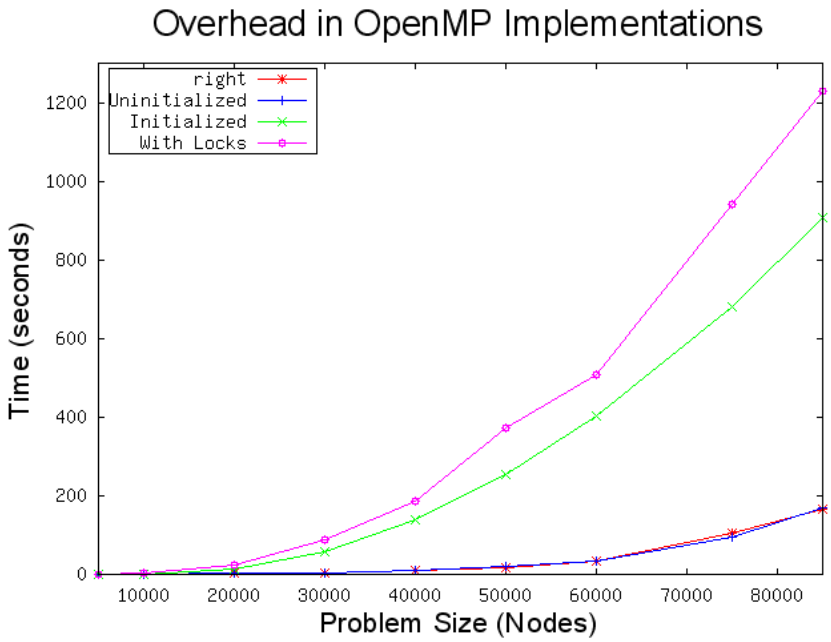

Figura 3.1: Ejemplo de figura

Capítulo 4

Conclusiones

bla, bla, bla, etc.

Apéndice A

Algoritmos empleados

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:Este segundo codigo 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 último 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.
#
##############################################################################
```

Apéndice B

Título del Apéndice 2

B.1. Otro apendice: Seccion 1

Texto

B.2. Otro apendice: Seccion 2

Texto

Bibliografía

[1] Anita de Waard. A pragmatic structure for research articles. In *Proceedings of the 2nd international conference on Pragmatic web*, ICPW '07, pages 83–89, New York, NY, USA, 2007. ACM.

[2] J. Gibaldi and Modern Language Association of America. *MLA handbook for writers of research papers*. Writing guides. Reference. Modern Language Association of America, 2009.

[3] G.D. Gopen and J.A. Swan. The Science of Scientific Writing. *American Scientist*, 78(6):550–558, 1990.

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

[5] Coromoto León. *Diseño e implementación de lenguajes orientados al modelo PRAM*. PhD thesis, 1996.

[6] Guido Rossum. Python library reference. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[7] Guido Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

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

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