Universidad
de La Laguna

Polinomio interpolador de Newton
$f(x) = e^x$

Iciar González Alonso

Serezade Tedahuit González Torres

Grupo (2)

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

Lenguajes y Sistemas Informáticos

Facultad de Matemáticas

Universidad de La Laguna

La Laguna, 11 de mayo de 2014

Resumen

Este documento describe un importante teorema desarrollado por el científico Isaac Newton. Dicho Teorema, el Polinomio Interpolador de Newton, facilita el cálculo de cualquier función mediente series de potencias. Concretamente, se ejemplificará este teorema con la función $f(x) = e^x$.
En este proyecto, se ha llevado el campo matemático al campo informático, de la programación, en un ejemplo más de su importante conexión. Los cálculos matemáticos han sido ejecutados a través de Python, un lenguaje de programación interpretado y sencillo de manejar.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Motivación y objetivos

En los siguientes párrafos se redactará el por qué se ha llevado a cabo el siguiente trabajo, es decir, tal y como indica el título de esta sección, cuáles son los objetivos del texto. Pues bien diferenciaremos entre:

- Objetivo principal: implementación con Python de las series de potencias de Newton. Concretamente se trabajó con dicho programa para aproximar a través de este método el valor de la función $f(x) = e^x$.
- Objetivos específicos: aprender a utilizar el método de Newton para aproximar funciones a través de series de potencias, centrándose en la aproximación de la función nombrada anteriormente.

1.1. Objetivo principal

Se tratará de llevar el campo matemático al campo de la computación, como una alternativa para realizar cálculos de forma más rápida y precisa. Ya que dichos cálculos los lleva a cabo un ordenador a través de un algoritmo que se repite siempre de la misma manera y por lo tanto realiza los mismos pasos y operaciones, aportando gracias a su mayor capacidad cálculos muy precisos. Mientras que un ordenador realiza el algoritmo de forma mecánica y en un periodo de tiempo bastante corto, una persona además de que no ser capaz de realizar los cálculos de forma tan mecánica, pudiéndose producir así errores de cálculo, también tardaría más ya que hay que realizar numerosas operaciones. Ayudándose de programas como Python, el cual utiliza un lenguaje interpretado, sencillo de entender y manejar, se pueden crear según las nesidades programas que satisfagan las mismas. Que en este caso son los cálculos que se tienen que realizar para aproximar a través de una de una serie de potencias, la función $f(x) = e^x$.

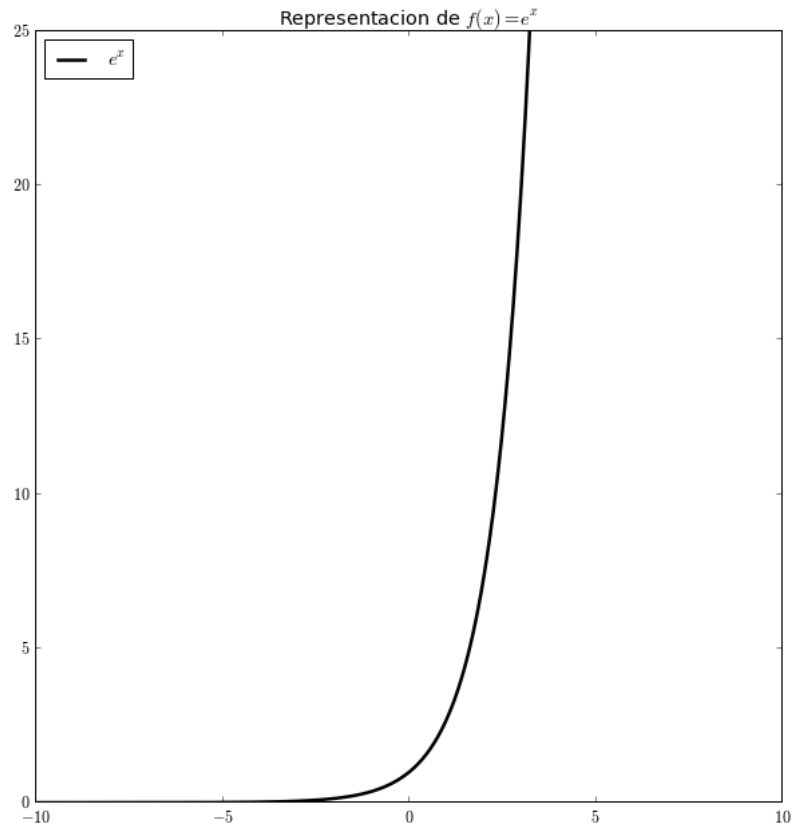

Figura 1.1: $f(x) = e^x$

1.2. Objetivos específicos

La función que se utilizará es un tanto difícil de manejar ya que el cálculo de sus distintos valores es complicado si no se dispone de calculadora, y entre otras cosas, es por esto por lo que se utilizan las series de potencias de newton, que permitirán expresar de forma polinómica dicha función, esto facilita los cálculos de los valores ya que se podrán realizar a mano.

Capítulo 2

Fundamentos teóricos

En este capítulo se pasará a desarrollar de forma teórica tanto el tema a tratar, la serie en potencias de Newton, como el programa de ordenador que utilizaremos para implementar este método.

2.1. Historia

La serie de potencias de Newton generalmente conocida como el binomio interpolador de Newton o fórmula de Newton en diferencias divididas, se trata de un teorema descubierto hacia 1664-1665. Todo comenzó con una carta que escribió Newton en respuesta a una petición del secretario de la Royal Society, Leibniz, quién favorecía el intercambio de información entre los científicos de la época, este estaba especialmente interesado en los estudios sobre las sobre las series finitas. En dicha carta fechada del 13 de junio de 1976, Newton presentaba el enunciado de su teorema además de algunos ejemplos. Más tarde viendo el interes de Leibniz en en el método propuesto, el autor del mismo redacta otra carta en la que explica en detalle cómo ha descubierto la serie binómica. Newton utilizó los conceptos de exponentes generalizados mediante los cuales una expresión polinómica se transformaba en una serie infinita, para ello se ayudó utilizando los métodos de Wallis de interpolación y extrapolación a nuevos problemas. Así pudo demostrar que varias series ya existentes eran casos particulares, esto lo hizo bien directamente o bien por diferenciación o integración.

El descubrimiento de la generalización de la serie binómica fue un gran resultado en sí mismo, pero además este desencadenó nuevos estudios como por ejemplo la idea de que se podría trabajar de la misma manera con series infinitas. El análisis de las series infinitas parecía posible porque a partir de entonces resultaron ser una forma equivalente para expresar funciones que las representaban. Cabe destacar que Newton no publicoó este teorema, sino que este fue publicado por Wallis en 1685, pero este atribuyó el descubrimiento a Newton, el del teorema.

Figura 2.1: Isaac Newton y Gottfried Leibniz

2.2. Teorema

La forma general del polinomio interpolante de Newton para $n+1$ datos $(x_0, f(x_0)), (x_1, f(x_1)), ..., (x_n, f(x_n))$ es:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - X_0)(x - x_1)(x - x_2) + ...$$

$$+a_n(x - X_0)(x - x_1)(x - x_2)....(x - x_{n-1})$$

Los coeficientes a_i se obtienen calculando un conjunto de cantidades denominadas diferencias divididas. La notación para las diferencias divididas de una función f(x) están dadas por:

$$f[x_i] = f(x_i)$$

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$$

$$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$$

$$f[x_i, x_{i+1}, x_{i+2}, x_{i+3}] = \frac{f[x_{i+1}, x_{i+2}, x_{i+3}] - f[x_i, x_{i+1}, x_{i+2}]}{x_{i+3} - x_i}$$

La fórmula general para las diferencias divididas sería:

$$f[x_i, x_{i+1}, x_{i+2}, ..., x_{i+j-1}, x_{i+j}] = \frac{f[x_{i+1}, x_{i+2}, ..., x_{i+j}] - f[x_i, x_{i+1}, ..., x_{i+j-1}]}{x_{i+j} - x_i}$$

Retomando el polinomio de Newton, $P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - X_0)(x - x_1)(x - x_2) + ... + a_n(x - X_0)(x - x_1)(x - x_2)....(x - x_{n-1})$.

Observese que $P_n x_0 = a_0$. Como $P_n(x)$ interpola los valores de f en $x_i, i = 0, 1, 2..., n$ entonces $P(x_i) = f(x_i)$, en particular $P_n(x_0) = f(x_0) = a_0$. Si se usa la notación de diferencia dividida $a_0 = f[x_0]$. de modo que si usamos la notación de diferencias divididas se quedará:

a_i	Diferencias
a_0	$f[x_0] = f(x_0)$
a_1	$f[x_0, x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0}$
a_2	$f[x_0, x_1, x_2] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$
a_3	$f[x_0, x_1, x_2, x_3] = \frac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0}$
a_4	$f[x_0, x_1, x_2, x_3, x_4] = \frac{f[x_1,x_2,x_3,x_4]-f[x_0,x_1,x_2,x_3]}{x_4-x_i}$

Cuadro 2.1: Equivalencia de a_i con las diferencias divididas

Capítulo 3

Procedimiento experimental

A continuación se enumerará y se desarrollarán los pasos que se han seguido en la elaboración del programa de las series de Newton en Python.

3.1. Descripción de los experimentos

El objetivo es crear un polinomio empleando el método de Newton que permita aproximar una función cuyo cálculo es más complejo. Para ello, se necesitan una serie de nodos, puntos determinados dentro de un intervalo de libre elección. En este experimento, se trabajará en el intervalo [0,1], y aunque es posible escoger cualquier punto, en este caso los nodos, que son cinco, serán equidistantes, es decir, la distancia entre cada uno de ellos será la misma. Así, se almacenan en una estos cinco puntos: 0, 0.25, 0.5, 0.75, 1.0. Gracias a un método iterativo, se almacenan en otra lista las imágenes correspondientes a los puntos que se han elegido en un nprincipio y en otra, las diferencias divididas, teniendo en cuenta la función del experimento, $f(x) = e^x$. Una vez hallados los coeficientes del polinomio, el método de Newton precisa de un valor "x"que se encuentre en el intervalo que se seleccionó de inicio, en este caso [0,1]. El valor que se ha introducido en este experimento es el x = 0.43. Para finalizar, el programa muestra el valor del polinomio con x = 0.43 y la función determinada $f(x) = e^x$, la imagen de la función en ese mismo punto y el error cometido por el polinomio.

3.2. Descripción del material

Los experimentos han sido realizados en un ordenador Packard Bell Atom N2600, procesador fabricado por Intel Atom. Tiene una velocidad de 1.6 GHz y la capacidad de su disco duro es de 320 GB. Tiene 1 GB de memoria RAM. Sus dimensiones son 35,8 x 23,8 x 7,6 cm y tiene un peso de 2,1 kg. Con 10,1 pulgadas y una tarjeta gráfica Intel, la batería tiene una duración aproximada de 8 horas. El Sistema Operativo es Bardinux 3.2.60-generic, una adaptación de la distribución de Linux "Kubuntu". El programa se ha elaborado en el editor de textos Kate, un paquete incorporado en Bardinux, se compila desde la Terminal, se ejecuta con Python y el documento PDF es visible a través del programa Okular.

3.3. Resultados obtenidos

El programa se ha eleaborado tomando cinco nodos, que se muestran en la primera columna de la tabla. Los resultados que se almacenan en la segunda lista son las imágenes correspondientes a la función $f(x) = e^x$, y se encuentran en la segunda columna. En la tercera columna de la tabla están representados los coeficientes del polinomio, calculados a través de las diferencias divididas.

x	f(x)	Coeficientes	Parte literal
0.0	1.0	1.0	1.0
0.25	1.2840254166877414	1.1361016667509656	0.43
0.5	1.6487212707001282	0.6453634985971632	0.0774
0.75	2.117000016612675	0.12219975773607672	-0.005418
1.0	2.718281828459045	0.026030877832598165	0.00173376

Cuadro 3.1: Datos y resultados del programa

Para calcularlas, se implementa un código para el número de nodos que tenemos, en este caso son cinco, a partir de la fórmula:

Las diferencias divididas de orden superior se forman de acuerdo con la siguiente fórmula de recursión:

$$f[x_{k-j}, x_{k-j+1}, \ldots, x_k] = \frac{f[x_{k-j+1}, \ldots, x_k] - f[x_{k-j}, \ldots, x_{k-1}]}{x_k - x_{k-j}}$$

Figura 3.1: Fórmula diferencias divididas

Y por último, se muestran los valores de la parte literal que acompañan a cada uno de los coeficientes, con el valor de la x ya sustituido, x = 0.43

3.4. Análisis de los resultados

El objetivo del polinomio de interpolación de Newton es facilitar el cálculo de las imágenes de una función, con la creación de un polinomio, puesto que su cálculo es más sencillo. Sin embargo, al modificar la función, se ha de tener el cuenta el error que se comete. Teóricamente, es necesario determinar un intervalo (en el experimento es el [0,1]) y asignar una serie de valores, que se denominan nodos, tantos como se desee. Cuanta mayor es la cantidad de nodos, más preciso será el valor del polinomio de Newton (en este ejemplo son cinco nodos, 0, 0.25, 0.5, 0.75, 1.0). Uno de los apartados de del programa creado ha calculado el valor de la función en una 'x' cuyo valor ha sido asignado

por defecto, x = 0.43 y a su vez el valor del polinomio en ese punto. Este cálculo se ha mostrado con 15 decimales. Además, el programa muestra el error cometido, en este caso 0.000600380953391210780978326511103 66. Esto se debe a que el valor de la función es f(0.43) = 1.537857904501672479113949520979076 62 y el valor del polinomio interpolador de Newton en x = 0.43 es 1.537257523548281268332971194467 96. El resultado proporciona un error de $6x10^{-4}$, bastante óptimo. El resultado del experimento ha sido realmente positivo, puesto que el intervalo escogido inicialmente era relativamente pequeño, y la cantidad de nodos en relación al tamaño del intervalo era la adecuada. Cuanto mayor es el intervalo, mayor número de nodos habrá que incluir, puesto que si no, la aproximación mediante el polinomio será muy poco precisa.

Capítulo 4

Conclusiones

1. Obtención de la aproximación de la función $f(x) = e^x$ mediante el polinomio de Newton implementado en Python.
2. Mejora del conocimiento en las estructuras iterativas de programación.
3. Cálculo del error cometido por el programa.
4. Aprender a aproximar funciones a través de las series de potencias.
5. Profundizar en los fundamentos teóricos del método.

Apéndice A

Programa

A.1. Algoritmo Newton

```
#!/usr/bin/python
#!encoding: UTF-8

from math import e

x = [ ]
for i in range (0,5):
  if i == 0:
    xi = 0.00
  else:
    xi = xi + 1
  x += [xi]
print x # Imprime las x

fx = [ ]
for i in range (0,5):
    fxi = e**x[i]
    fx += [fxi]
print fx # Imprime las fx


d = [ ]
for i in range (0,5):
  if i == 0:
    di = fx[i]
  else:
    di = (fx[i] - fx[i-1])/(x[i] - x[i-1])
  d += [di]

a = [d[0], d[1], (d[2]-d[1])/(x[2]-x[0]), (d[3]-d[2]+d[1]-d[2])/(x[3]-x[0]),
(d[4]-d[3]+d[2]-d[3]+d[2]-d[3]+d[2]-d[1])/(x[4]-x[0])]

print a # Imprime los coeficientes del polinomio

y = 0
```

```
l = [ ]
valorx = 0.43
for i in range (0,5):
  li = 1
  if i == 0:
    li = 1
  else:
    for j in range (0,i):
      li = li*(valorx-x[j])
      j+=1
  l = l + [li]
print l #Imprime la parte literal del polinomio con el valor de x predeterminado

valorpol = 0

for i in range (0,5):
  valorpol += a[i] * l[i]

print '%.35f' %valorpol #Imprime el valor del polinomio con la x que le hemos dado
print '%.35f' %(e**valorx)  #Imprime el valor real de la función con el valor de x que se asigna
print '%.35f' %(abs(valorpol-real)) #Imprime el error cometido por el polinomio.
```

A.2. Terminal

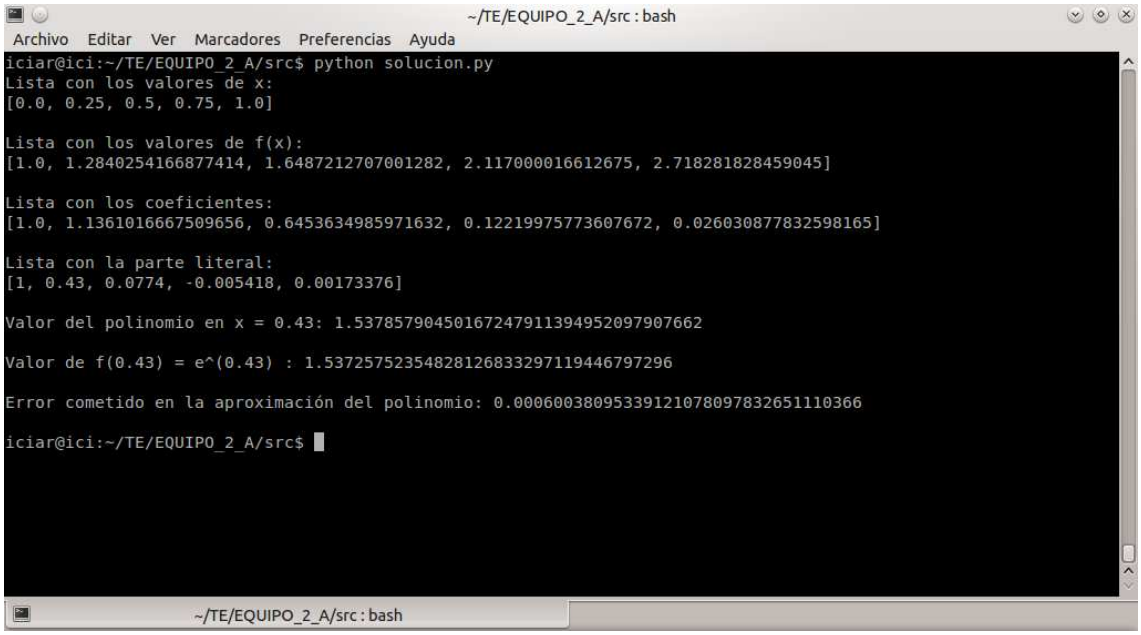

Figura A.1: Terminal

Apéndice B

Nodos

B.1. Nodos equidistantes

El cálculo del polinomio de interpolación se simplifica cuando los nodos están
igualmente espaciados.

En este caso el polinomio se calcula utilizando el concepto de diferencia finita.

Definición: Diferencia finita progresiva.
Se define como diferencia finita progresiva de una funcion f(x) en un punto y
su valor es la diferencia entre la imagen del segundo valor menos la del primero.

Esta es la diferencia finita de primer orden. Del mismo modo, se calculan las de orden superior.

Bibliografía

[1] Solo Ciencia. http://www.solociencia.com/cientificos/isaac-newton-teorema-binomio.htm.

[2] Universidad Politécnica de Madrid. http://ocw.upm.es/matematica-aplicada/programacion-y-metodos-numericos/contenidos/TEMA_3/Apuntes/InterpolacionOCW.pdf.

[3] Instituto Tecnológico de Tuxtla Gutiérrez. https://sites.google.com/site/danaly7/unidad-5/polinomio-de-interpolacion-de-newton.

[4] ULPGC Informática. http://numat.net/tutor/newton.pdf.

[5] ETSI Bilbao Purificación González. http://www.ehu.es/pegonzalez/I.Teleco/Apuntes/tema5.pdf.