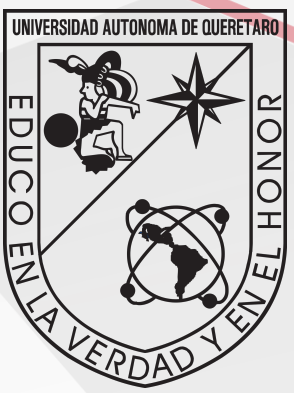
Universidad Autónoma de Querétaro

Facultad de Ingeniería



Tarea 13: Diferenciación numérica

Análisis numérico

Autor:
J.A. Salinas Sánchez
Mayo 2022

Índice general

Capítulo 1

Introducción

Como parte del cálculo infinitesimal, la diferenciación cobra vital importancia pues, al igual que la integración, la diferenciación sirve para tratar con infinitos, solo que, en lugar de los infinitos infinitos dentro de un rango finito; limitar ese rango para tratar con infinitos particulares. Es decir, en lugar de las generalidades; en lo específico. Pues, al contrario de la integración; la cual sirve para sumar, promediar y caracterizar las propiedades cambiantes infinitesimalmente a lo largo de un intervalo de n dimensiones (una recta, un área, un volumen o un hipervolumen). La diferenciación sirve para caracterizar dichar propiedades en cada punto del infinitesimal del tiempo; obteniendo su tasa de cambio en dicho punto. Eso nos permite efectuar predicciones, interpolaciones, obtener resultado a ecuaciones, definir ciertas propiedades fisico-matemáticas relacionadas al cambio. Así mismo, la diferenciación permite obtener funciones a partir de simplemente sus razones de cambio: las famosas ecuaciones diferenciales.

Aunque en este apartado no se tratará la resolución de ecuaciones diferenciales, sí se verá la forma de dar solución a derivadas, de forma numérica. Al igual que las integrales, las derivadas pueden ser calculadas desde su definición, tanto para obtener soluciones analíticas, como para soluciones numéricas; simplemente aplicándola para un límite finito en lugar de uno infinitesimal. La definición formal de la derivada es la siguiente:

$$f'(x) = \lim_{x \to \infty} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{1.1}$$

1.1. Diferencias finitas

Donde Δx es un incrimento infinitesimal dentro de la función y, siguiendo lo antes dicho, éste se puede reemplazar por un aumento numérico, finito y arbitrario. Dejando lo siguiente:

$$f'(x) \approx \frac{f(x + h) - f(x)}{h} \tag{1.2}$$

Donde se usa h para indicar el cambio en la definición del incremento.

Luego, apartir de la ecuación anterior, se puede decidir si aproximar desde un lado o del otro de la recta numérica. Entonces, aplicando propie-

dades de las funciones:

$$f'(x) \approx \frac{f(x+h-h)-f(x-h)}{h} = \frac{f(x)-f(x-h)}{h} \tag{1.3}$$

$$\wedge \tag{1.4}$$

$$f'(x) \approx \frac{f(x+h+h)-f(x+h)}{h} = \frac{f(x+2h)-f(x+h)}{h} \tag{1.5}$$

Finalmente, uno puede, al igual que con la integración, obtener una aproximación centrada:

$$f'(x) \approx \frac{f(x+2h)-f(x-h)}{2h} \tag{1.6}$$

Aparte de esto, existe otros métodos basados en extrapolaciones que, una vez calculadas algunas antiprimitivas; utilizan los valores para mejorar su precisión. Así es el caso de todos lo métodos de derivadas por diferencias finitas por adelante, atrás y centradas; con rangos de error h^2, h^2 y h^4 respectivamente:

$$f'(x) \approx \frac{-f(x+2h)+4f(x+h)-3f(x)}{2h} \tag{1.7}$$

$$f'(x) \approx \frac{f(x)-4f(x-h)+3f(x-2h)}{2h} \tag{1.8}$$

$$f'(x) \approx \frac{-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)}{12h} \tag{1.9}$$

1.2. Extrapolación de Richardson

Como ya se dijo antes, una extrapolación consta en utilizar datos para obtener otros datos. Tal es el caso de la extrapolación de Richardson que, al igual que con la integración; utiliza dos aproximaciones previas para obtener una nueva y más precisa. No obstante, como la derivada no se derine como un proceso iterativo $per\ se$; la extrapolación de Richarson no puede generar una fórmula para el ik-ésimo nivel de aproximación y se limita a dotar de mayor peso a la aproximación más exacta, y menor, a la menos exacta en una razón $4:1$:

$$D \approx \frac{4}{3}D(h_2) - \frac{1}{3}D(h_1) \tag{1.10}$$

Capítulo 2

Desarrollo y método

2.1. Método

Para solucionar el siguiente conjunto de problemas, se escribió un único método capáz de realizar cálculos simbólicos, así como poder realizarlos a partir de un número pequeño de entradas por parte del usuario. Este código se basa en pedir la función a diferenciar, el factor h , y en caso de que el usuario quisiere la extrapolación de Richardson, pedirle el factor $h2$ para realizar los cálculos. Esto se hizo pensando en pedir el menor número de entradas al usuario.

Luego, el código computa una aproximación utilizando la paquetería del lenguaje **python3** y el resto de aproximaciones (adelante, atrás y centro con dos grados de precisión). Con estos datos, se calcula el error relativo aproximado de cada aproximación y lo imprime. Por último, si el usuario solicitó la extrapolación de Richardson, el código pide $h2$ y ejecuta el mismo procedimiento que con los casos anteriores.

2.1.1. Código

```
#coding:utf8


import math as mt
import  numpy as np
import sympy as sp

x=sp.symbol('x')

def main():
    f=str(input('Introduzca su funci n f(x): '))
    p=float(input('Introduzca la abscisa (x) de su evaluaci n: '))
    h1=float(input('Introduzca su diferencia h1: '))
    q=str(input(' Desea  utilizar extrapolaci n de Richardson[s/n
   }]: ?'))
    func=getfunction(f)
    dx=sp.diff(func,x)
    andx=dx.evalf(subs={x:p})

    print('Diferencia frontal h')
    numdx=difffronth(func,h1,p)
    e=abs((numdx-andx)/andx)*100
    print('Error relativo % (contra el algoritmo sympy)')
```

```
    print(e)
    print('Diferencia frontal h^2')
    numdx=difffronth2(func,h1,p)
    e=abs((numdx-andx)/andx)*100
    print('Error relativo % (contra el algoritmo sympy)')
    print(e)
    print('Diferencia por atr s h')
    numdx=diffbackh(func,h1,p)
    e=abs((numdx-andx)/andx)*100
    print('Error relativo % (contra el algoritmo sympy)')
    print(e)
    print('Diferencia por atr s h^2')
    numdx=diffbackh2(func,h1,p)
    e=abs((numdx-andx)/andx)*100
    print('Error relativo % (contra el algoritmo sympy)')
    print(e)
    print('Diferencia centrada h^2')
    numdx=diffcenth2(func,h1,p)
    e=abs((numdx-andx)/andx)*100
    print('Error relativo % (contra el algoritmo sympy)')
    print(e)
    print('Diferencia centrada h^4')
    numdx=diffcenth4(func,h1,p)
    e=abs((numdx-andx)/andx)*100
    print('Error relativo % (contra el algoritmo sympy)')
    print(e)

    if q=='S' or q=='s':
        h2=float(input('Introduzca su h2: '))
        print('Interpolaci n de Richardson')
        numdx=richardson(func,h1,h2,p)
        e=abs((numdx-andx)/andx)*100
        print('Error relativo % (contra el algoritmo sympy)')
        print(e)

def difffronth(func,h,p):
    return((func.evalf(subs={x:p+h})-func.evalf(subs={x:p}))/h)

def diffbackh(func,h,p):
    return((func.evalf(subs={x:p})-func.evalf(subs={x:p-h}))/h)

def difffronth2(func,h,p):
    return((-func.evalf(subs={x:p+2*h})+4*func.evalf(subs={x:p+h})
   -3*func.evalf(subs={x:p}))/(2*h))

def diffbackh2(func,h,p):
    return((func.evalf(subs={x:p-2*h})-4*func.evalf(subs={x:p-h})
   +3*func.evalf(subs={x:p}))/(2*h))

def diffcenth2(func,h,p):
    return(func.evalf(subs={x:p+h})-func.evalf(subs={x:p-h}))/(2*h)

def diffcenth4(func,h,p):
    return((-func.evalf(subs={x:p+2*h})+8*func.evalf(subs={x:p+h})
   -8*func.evalf(subs={x:p-h})+func.evalf(subs={x:p-2*h}))/(12*h))

def richardson(func,h1,h2,p):
    return((4/3)*diffbackh2(func,h,p)-(1/3)*diffbackh(func,h,p))

def getfunction(f):
    global x
    return(sp.sympify(f))
```

```
83 if __name__=='__main__':
84     main()
```