Problemas parcial I
Métodos Computacionales I

Edward Enrique Manosalva Pinto
Juan David Vasquez Hernandez

September 6, 2023

1 Ejercicios: Derivación.

1.1 Ejercicio 3.7.5

Partiendo de la definición del operador derivada:

$$Df(x_j) = \frac{f(x_{j+1}) - f(x_j)}{h} \tag{1}$$

Se puede encontrar la siguiente derivada (y las sucesivas) utilizado este operador como función:

$$\begin{aligned} D^2 f(x_j) &= \frac{D(x_{j+1}) - D(x_j)}{h} = \frac{\frac{f(x_{j+2})-f(x_{j+1})}{h} - \frac{f(x_{j+1})-f(x_j)}{h}}{h} \\ D^2 f(x_j) &= \frac{f(x_{j+2}) - 2f(x_{j+1}) + f(x_j)}{h^2} \end{aligned} \tag{2}$$

En general se puede notar que para cualquier Derivada de orden n su fórmula será la sumatoria alterna de $f(x)$ desde $f(x_{j+n})$ hasta $f(x_j)$ con los coeficientes binomiales de orden n en cada término sobre h^n. Algo del estilo.

$$D^n f(x_j) = \frac{\sum_{k=0}^{n} (-1)^{n-1} \binom{n}{k} f(x_{j+(n-k)})}{h^n} \tag{3}$$

Por lo que para $D^4 f(x_j)$ se obtiene finalmente:

$$D^4 f(x_j) = \frac{f(x_{j+4}) - 4f(x_{j+3}) + 6f(x_{j+2}) - 4f(x_{j+1}) + f(x_j)}{h^4} \tag{4}$$

Como se tiene equidistancia, se pueden correr los x_j para centrar la derivada, consiguiendo la expresión del ejercicio.

$$D^4 f(x_j) = \frac{f(x_{j+2}) - 4f(x_{j+1}) + 6f(x_j) - 4f(x_{j-1}) + f(x_{j-2})}{h^4} \tag{5}$$

Este operador es de orden 4, y por lo tanto tiene $O(h^4)$.

1.2 Ejercicio 3.7.8

a) Para obtener la interpolación se utilizan los polinomios de Newton.

$$N(x) = \sum_{i=0}^{n-1} = a_i n_i(x) \tag{6}$$

Calculando los $n_i(x)$:

$$\begin{aligned} n_0 &= 1 \\ n_1 &= (x - x_0) \\ n_2 &= (x - x_0)(x - x_1) \end{aligned} \tag{7}$$

Para calcular los a_i se hace uso de las diferencias divididas ($a_j = [f(x_0)...f(x_j)]$).

$$\begin{aligned} a_0 &= [f(x_0)] = f(x_0) \\ a_1 &= [f(x_0), f(x_1)] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\ a_2 &= [f(x_0), f(x_1), f(x_2)] = \frac{[f(x_1), f(x_2)] - [f(x_0), f(x_1)]}{x_2 - x_0} \\ a_2 &= \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_2-x_0}}{x_2 - x_0} \end{aligned} \tag{8}$$

Y por último se forma el polinomio completo:

$$N(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) + \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_2-x_0}}{x_2 - x_0}(x - x_0)(x - x_1) \tag{9}$$

c) Se realizó el siguiente código para la derivada progresiva:

```
import numpy as np
import matplotlib.pyplot as plt

def Function(x):
    return np.sqrt(np.sin(x))

def DerivadaProgresiva(f,x,h):

    if h != 0:
        d = (f(x+h) - f(x))/h

    return d

x = np.linspace(0.1,1.1,30)
h = 0.01

PDerivative = DerivadaProgresiva(Function,x,h)
```

d) Se realizó el siguiente código para la derivada central:

```
def DerivadaCentral(f,x,h):

    if h != 0:
        d = (f(x+h) - f(x-h))/(2*h)

    return d

CDerivative = DerivadaProgresiva(Function,x,h)
```

e) La derivada de la función se puede calcular fácilmente a partir de la derivada de una raíz y aplicando regla de la cadena para la derivada de $tan(x)$. Lo anterior daría como resultado lo siguiente.

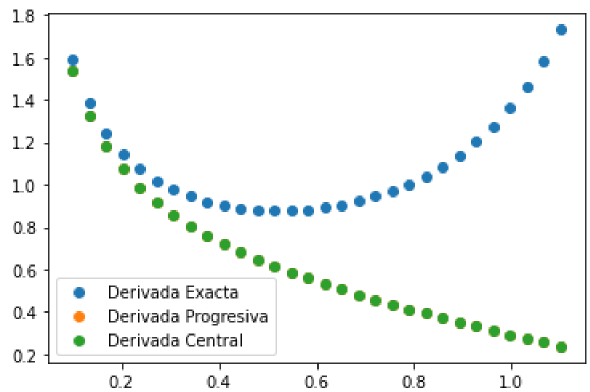
$$f'(x) = \frac{sec^2(x)}{2\sqrt{tan(x)}} \tag{10}$$

```
def ExactDerivative(x):
    return (1/np.cos(x))**2/(2*np.sqrt(np.tan(x)))

EDerivative = ExactDerivative(x)

plt.scatter(x,EDerivative,label='Derivada Exacta')
plt.scatter(x,PDerivative,label='Derivada Progresiva')
plt.scatter(x,CDerivative,label='Derivada Central')
plt.legend()
```

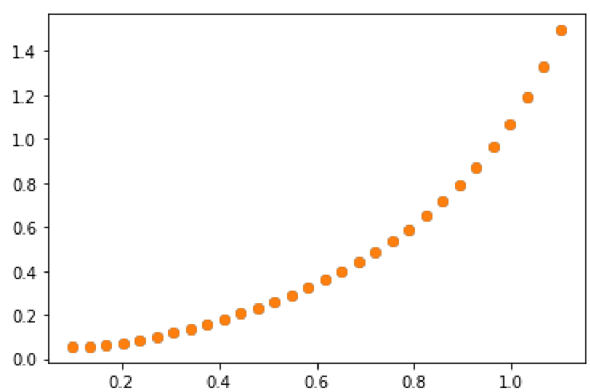
La gráfica queda:



f) Para este caso, sí ambos resultados manejan una precisión similar por las características de la derivada (aunque en general suele ser más cercana la derivada central).

```
ErrorP = np.abs(EDerivative-PDerivative)
ErrorC = np.abs(EDerivative-CDerivative)

plt.scatter(x,ErrorP)
plt.scatter(x,ErrorC)
```



np.max(ErrorP) = np.max(ErrorC) = 1.4961311408498976

2 Ejercicios: Raíces de polinomios.

2.1 3.10.3

$$\begin{aligned} 3x^5 + 5x^4 - x^3 &= 0 \\ x^3(3x^2 + 5x - 1) &= 0 \\ 3x^2 + 5x - 1 &= 0 \end{aligned} \tag{11}$$

Por lo tanto:

$$\begin{aligned} x_1 &= 0 \\ x_2 &= \frac{-5 + \sqrt{25 + 4 \times 3}}{2 \times 3} = \frac{-5 + \sqrt{37}}{6} \\ x_3 &= \frac{-5 - \sqrt{25 + 4 \times 3}}{2 \times 3} = \frac{-5 - \sqrt{37}}{6} \end{aligned} \tag{12}$$

3 Ejercicios: Interpolación de Lagrange.

3.1 Ejercicio 3.13.4

Se realizó el siguiente código para calcular la trayectoria de la bala a partir de interpolación de Lagrange:

```
f = open('./Parabolico.csv')
f.readline()
lineas = f.readlines()
f.close()

x, y = [], []
for linea in lineas:
    datos = linea[:-1].split(',')
    x.append(float(datos[0]))
    y.append(float(datos[1]))

x_i = np.array(x)
y_i = np.array(y)
```

Mediante este código primero se extraen los datos de la posición x y y a apartir de los cuales se calcula la base cardinal para el polinomio mediante el siguiente código.

```
def lagrange_basis(x: float,nodos: np.array,i: int):
    l = 1
    for k in range(len(nodos)):
        if k != i:
            l *= (x-nodos[k])/(nodos[i]-nodos[k])
```

```
    return l
```

A continuación se calcula el polinomio de Lagrange usando la base calculada mediante la librería Simpy usando para ello la siguiente función.

```
def lagrange_polynomial(x:float,nodos:np.array,valores:np.array):
    L = 0
    for i in range(len(nodos)):
        L += valores[i]*lagrange_basis(x,nodos,i)
    return L
```

Para finalizar, se encuentra la implementación de ambas funciones y la posterior impresión de la trayectoria.

```
x_sym = sym.Symbol('x',real=True)

t = np.linspace(min(x_i),max(x_i))
y = lagrange_polynomial(t,x_i,y_i)

fig, ax = plt.subplots()
ax.plot(t,y)
ax.scatter(x_i,y_i,c='red',marker='8',fc='white',ec='red',s=200)
```

La siguiente figura muestra la trayectoria seguida por la bala.