Taller: Integración numérica

Juan José Segura Flórez
201510684

16 de septiembre de 2019

1. Problemas

1. Suponga que va a realizar una integral numérica en el intervalo $[a, b]$, usando un número n impar de intervalos, cada uno de longitud h. Muestre que el aporte del último intervalo está dado por

$$\int_{b-h}^{h} f(x)dx = \frac{h}{12}\left(-f_{n-2} + 8f_{n-1} + 5f_n\right)$$

2. Consideremos un pédulo de longitud l, confinado en un plano vertical. Puede mostrarse que el periodo de una oscilación, de amplitud θ_0, está dado por

$$T = 4\sqrt{\frac{l}{2g}}\int_0^{\theta_0} \frac{d\theta}{\sqrt{\cos\theta - \cos\theta_0}}, \tag{1}$$

donde g es la aceleración de la gravedad (Asuma $g = 9{,}8\ \mathrm{ms}^{-2}$ y $l = 1$ m). En el límite de pequeñas oscilaciones, la expresión (1) se reduce a $T \approx 2\pi\sqrt{l/g}$.

   a) Asuma $\theta_0 = \pi/16$. Evalue numéricamente la integral (1) haciendo uso de la regla del trapecio y asumiendo para el número n de subintervalos los valores 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024.

   b) Hagala la misma evaluación que en el numeral anterior, pero ahora implemente la regla de Simpson. ¿Qué puede concluir al comparar los resultados con los del punto anterior?

   c) Ahora, haga uso únicamente de la regla de simpson y asuma 1024 intervalos. Evalúe la integral (1) para $\theta_0 = \pi/128, \pi/64, \pi/32, \pi/16, \pi/8, \pi/4, \pi/2$. Compare en una tabla, estos resultados con la aproximación para pequeñas oscilaciones. Explique con argumentos físicos la diferencia entre la formula exacta y la apróximada.

3. Construya la fórmula para integración numérica con paso adaptativo, haciendo uso de interpolación de Lagrange con polinomios de grado 3.

4. Suponga que x y $f(x)$ son cantidades físicas medibles experimentalmente, y que la integral de $f(x)$ da información de otra propiedad del sistema. En el archivo adjunto al presente taller, se encuentra los datos experimentales (x_i, f_i). Determine la integral de la función descrita por este conjunto de datos, haciendo uso de interpolación de Lagrange con polinomios de grado 3, y compare con el resultado obtenido con la regla del trapecio. El resultado analítico de esta integral es 1,3852229032511942.

2. Solución problema 1

En una interpolación de orden 2, el polinomio de Lagrange posee la forma:

$$f(x) = \frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}f_{i-1} + \frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})}f_i + \frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}f_{i+1}, \tag{2}$$

como se quiere saber la contribución del último slice (compuesto por dos puntos), se tendrá como último punto $x_{i+1} = b$ y la diferencia entre dos puntos adyacentes h [1]. Se puede por lo tanto, hallar lo siguiente:

[1] En este punto se considera que los datos están igualmente espaciados, razón por la cual solo se necesita de un solo valor h para caracterizar la diferencia entre dos puntos.

- $x - x_i = x - (b - h)$,
- $x - x_{i+1} = x - b$,
- $x - x_{i-1} = x - (b - 2h)$,

es decir:

- $x_i = b - h$,
- $x_{i-1} = b - 2h$.

Lo anterior permite deducir que:

- $x_{i-1} - x_i = -h$,
- $x_{i-1} - x_{i+1} = -2h$,
- $x_i - x_{i-1} = h$,
- $x_i - x_{i+1} = -h$,
- $x_{i+1} - x_{i-1} = 2h$,
- $x_{i+1} - x_i = h$.

Si se reemplaza lo anterior en la ecuación (2), junto con el siguiente cambio de indice:

$$i + 1 \to n \tag{3}$$

se tendrá:

$$f(x) = \frac{(x - (b - h))(x - b)}{(-h)(-2h)} f_{n-2} + \frac{(x - (b - 2h))(x - b)}{(h)(-h)} f_{n-1} + \frac{(x - (b - 2h))(x - (b - h))}{(2h)(h)} f_n, \tag{4}$$

es decir,

$$f(x) = \frac{(x - (b - h))(x - b)}{2h^2} f_{n-2} - \frac{(x - (b - 2h))(x - b)}{h^2} f_{n-1} + \frac{(x - (b - 2h))(x - (b - h))}{2h^2} f_n, \tag{5}$$

la integral que debemos solucionar es la siguiente:

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx = \int_{b-h}^{b} f(x)dx \tag{6}$$

lo anterior se debe al hecho de que:

$$x_i = b - h \text{ y } x_{i+1} = b \tag{7}$$

ahora, integraremos la ecuación anterior con ayuda de mathematica, usando el siguiente código:

```
 (*definición de la función*)
g = ((x - (b - h))*(x - b))/(2*h^2)*
   Subscript[f, n - 2] - ((x - (b - 2*h))*(x - b))/h^2*
   Subscript[f, n - 1] + ((x - (b - 2*h))*(x - (b - h)))/(2*h^2)*
   Subscript[f, n]
(*integral a resolver*)
Integrate[g, {x, b - h, b}]
```

el resultado es:

$$\int_{b-h}^{b} f(x)dx = -\frac{h}{12} f_{n-2} + \frac{2h}{3} f_{n-1} + \frac{5h}{12} f_n \tag{8}$$

Si se saca factor común $h/12$ se tendrá:

$$\int_{b-h}^{b} f(x)dx = \frac{h}{12} \left(-f_{n-2} + 8f_{n-1} + 5f_n\right) \tag{9}$$

Q.E.D.

3. Solución problema 2

3.1. Inciso a

Se creará un programa que permita evaluar la integral (1) de forma númerica. El resultado es el siguiente:

```
import numpy as np #importar módulo númpy

theta_0=np.pi/16 #amplitud
g=9.8 #aceleración de la gravedad
l=1 #logitud del péndulo

n=int(input("ingrese el valor de subintervalos: ")) #se pide por pantalla el dato n

theta=np.linspace(0,theta_0,n+1) #intervalo de integración
theta=[i for i in theta] #transformación del arreglo de numpy a lista
theta.pop(len(theta)-1) #eliminación del último punto theta (divergencia)
theta=np.array(theta) #transformación de lista a arreglo de numpy

h=theta[1]-theta[0] #diferencia de valores en theta

alpha=h/2 #factor alpha

beta=4*pow(l/(2*g),1/2) #factor beta

#definición de la función
f=beta/(pow(np.cos(theta)-np.cos(theta_0),1/2))

Int=0 #inicialización de la integral en 0

#integral generada por medio de búcle for
for i in range(len(f)): #desde 0 hasta len(f)-1
    if i==len(f)-1: #en el último elemento de f
        pass #el último elemento de f no ejecutará el método del trapecio
    else:
        Int+=alpha*(f[i]+f[i+1]) #método del trapecio

print("Resultado de la integral (método del trapecio): %f"%Int)#se mostrará por pantalla el resultado
```

El anterior programa de Python realiza la integral númerica con el *método del trapecio*:

$$S = \frac{h}{2}\sum_{n-1}^{i=0}(f_i + f_{i+1}) \tag{10}$$

la función $f(\theta)$ que es integrada por el programa es la siguiente:

$$f(\theta) = 4\sqrt{\frac{l}{2g}}\frac{1}{\sqrt{\cos\theta - \cos\theta_0}} \tag{11}$$

En el programa de Python se ha nombrado algunas constantes de la siguiente forma:

$$\alpha = \frac{h}{2} \text{ y } \beta = 4\sqrt{\frac{l}{2g}} \tag{12}$$

Por lo tanto la función $f(\theta)$ es vista como:

$$f(\theta) = \frac{\beta}{\sqrt{\cos\theta - \cos\theta_0}} \tag{13}$$

Un dato curioso es que si la función $f(\theta)$ es evaluada en θ_0 ocurre una indeterminación, es decir que:

$$\lim_{\theta\to\theta_0} f(\theta) = \infty \tag{14}$$

Por esta razón, se ha eliminado este punto a la hora de crear la el arreglo f de numpy en el programa. A las integrales de este tipo se les denominan *integrales indefinidas* y la forma de resolverlas es mediante un cambio de variable muy sutil en uno de lo límites de integración:

$$S = \lim_{t \to b} \int_a^t f(x)dx \tag{15}$$

La idea es que analíticamente se hace la integral y se hace tender el resultado al punto donde existe un problema del estilo de la ecuación (14), de esta forma, el resultado de la integral no diverge. Algo similar ocurre para el caso del presente problema, con el programa *punto_2_a.py* [2] se obtiene los siguientes resultados para $n = 2$, 4, 8, 16, 32, 64, 128, 256, 512, 1024:

- si $n = 2$ entonces la integral será: $S = 0{,}689549$
- si $n = 4$ entonces la integral será: $S = 1{,}102132$
- si $n = 8$ entonces la integral será: $S = 1{,}376447$
- si $n = 16$ entonces la integral será: $S = 1{,}565192$
- si $n = 32$ entonces la integral será: $S = 1{,}696939$
- si $n = 64$ entonces la integral será: $S = 1{,}789513$
- si $n = 128$ entonces la integral será: $S = 1{,}854770$
- si $n = 256$ entonces la integral será: $S = 1{,}900842$
- si $n = 512$ entonces la integral será: $S = 1{,}933394$
- si $n = 1024$ entonces la integral será: $S = 1{,}956404$

Si se realiza un programa que me permita ver de forma gráfica a qué número tiende la integral (una gráfica de S en función de n) se tendrá qué S tiende a un valor muy cercano a 2 (véase figura 1).

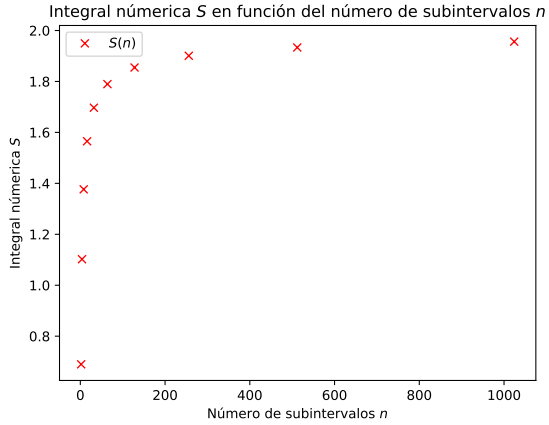

Figura 1: Integral númerica S en función de n.

3.2. Inciso b

Las implicaciones con respecto al punto en el cual la función diverge son exactamente las mismas que en el inciso anterior, sin embargo, acá se hace uso de la regla de Simpson para integrar númericamente:

$$S = \frac{h}{3} \sum_{j=0}^{n/2-1} (f_{2j} + 4f_{2j+1} + f_{2j+2}) \tag{16}$$

y se requiere hacer uso de la ecuación (1), ya que el número de subintervalos es par, al eliminar uno de sus puntos el número de subintervalos se convertirá en un número impar. El programa usado para realizar la integral númerica es llamado *punto_2_b.py*, y el código fuente se muestra a continuación:

[2] Los programas usados para resolver el taller se encuentran en la misma caperta que contiene el presente documento.

```
import numpy as np #importar módulo numpy

theta_0=np.pi/16 #amplitud
g=9.8 #aceleración de la gravedad
l=1 #logitud del péndulo

n=int(input("ingrese el valor de subintervalos: "))

theta=np.linspace(0,theta_0,n+1) #intervalo de integración
theta=[i for i in theta] #transformación del arreglo de numpy a lista
theta.pop(len(theta)-1) #eliminación del último punto theta (divergencia)
theta=np.array(theta) #transformación de lista a arreglo de numpy

h=theta[1]-theta[0] #diferencia de valores en theta

alpha=h/3 #factor alpha

beta=4*pow(l/(2*g),1/2) #factor beta

#definición de la función
f=beta/(pow(np.cos(theta)-np.cos(theta_0),1/2))

Int=0 #inicialización de la integral en 0


#integral generada por medio de búcle for
for j in range(int(n/2)): #desde 0 hasta (n/2)-1
    if j==int(n/2)-1: # último slice
        Int+=(h/12)*(-f[n-3]+8*f[n-2]+5*f[n-1]) #contribución de último slice
    else:
        Int+=alpha*(f[2*j]+4*f[2*j+1]+f[2*j+2]) #regla de Simpson

print("Resultado de la integral (regla de Simpson): %f"%Int)
#se mostrará por pantalla el resultado
```

Los resultados del programa son los siguientes:

- si $n = 2$ la integral no se ejecuta porque requiere al menos 3 puntos para integrar la función.
- si $n = 4$ entonces la integral será: $S = 1{,}091079$
- si $n = 8$ entonces la integral será: $S = 1{,}368084$
- si $n = 16$ entonces la integral será: $S = 1{,}559114$
- si $n = 32$ entonces la integral será: $S = 1{,}692590$
- si $n = 64$ entonces la integral será: $S = 1{,}786420$
- si $n = 128$ entonces la integral será: $S = 1{,}852577$
- si $n = 256$ entonces la integral será: $S = 1{,}899289$
- si $n = 512$ entonces la integral será: $S = 1{,}932296$
- si $n = 1024$ entonces la integral será: $S = 1{,}955627$

Si hacemos un gráfico de S en función de n con el cual podamos comparar ambas integrales obtenemos la figura 2. en la figura 2 se puede observar que la regla del trapecio para integrar númericamente crece más rápidamente que la regla de Simpson. Si realizamos la integral númerica con ayuda de Mathematica se obtiene que el valor de la integral númerica debería dar 2,01194. Si hallamos los errores de ambos métodos relativos al cálculo realizado por Mathematica obtenemos que para el método del trapecio el error es 0,027603300801718023 y para el método de Simpson es 0,02798955257875524. Concluimos que, el método del trapecio, para este caso, ¡es más preciso que el método de Simpson!

Para incluir el error en los programas construidos para este inciso y el anterior solo se añade la siguiente línea de código:

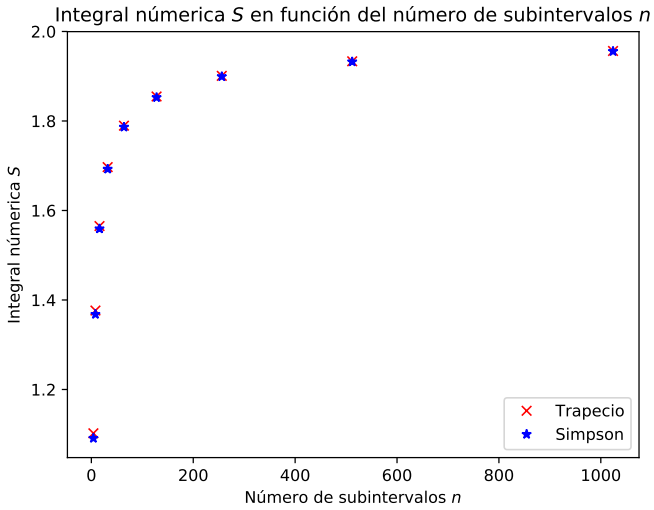

Figura 2: Comparación de S de la regla del trapecio vs S de Simpson.

```
error=abs(1-Int/2.01194)
print(error)
```

3.3. Inciso c

Haciendo uso del programa llamado *punto_2_c.py*, cuyo código fuente es mostrado a continuación:

```
import numpy as np #importar módulo numpy

theta_0=np.pi/16 #amplitud que puede variarse
g=9.8 #aceleración de la gravedad
l=1 #logitud del péndulo

n=1024 #se fija el número de subintervalos

theta=np.linspace(0,theta_0,n+1) #intervalo de integración
theta=[i for i in theta] #transformación del arreglo de numpy a lista
theta.pop(len(theta)-1) #eliminación del último punto theta (divergencia)
theta=np.array(theta) #transformación de lista a arreglo de numpy
h=theta[1]-theta[0] #diferencia de valores en theta

alpha=h/3 #factor alpha

beta=4*pow(l/(2*g),1/2) #factor beta

#definición de la función
f=beta/(pow(np.cos(theta)-np.cos(theta_0),1/2))

Int=0 #inicialización de la integral en 0


#integral generada por medio de búcle for
for j in range(int(n/2)): #desde 0 hasta (n/2)-1
    if j==int((n/2))-1: # último slice
        Int+=(h/12)*(-f[n-3]+8*f[n-2]+5*f[n-1]) #contribución de último slice
    else:
        Int+=alpha*(f[2*j]+4*f[2*j+1]+f[2*j+2]) #regla de Simpson

print("Resultado de la integral (regla de Simpson): %f"%Int)
#se mostrará por pantalla el resultado
```

Se hallan los siguientes valores para la integral númerica usando la regla de Simpson: La diferencia entre la

θ	**Valor de la integral**	**error relativo a límite de ángulos pequeños**
$\pi/128$	1,951033	0,027929
$\pi/64$	1,951252	0,027821
$\pi/32$	1,952125	0,027385
$\pi/16$	1,955627	0,025641
$\pi/8$	1,969749	0,018605
$\pi/4$	2,028166	0,010501
$\pi/2$	2,298708	0,145294

Cuadro 1: Cuadro de comparación entre el valor de la integral númerica para distintos valores de ángulo y el valor del periodo para pequeñas oscilaciones ($n = 1024$).

fórmula exacta y la fórmula aproximada radica en el concepto del isocronismo que existe en el péndulo a pequeño ángulos. Desde el punto de vista físico se puede derivar el Lagrangiano de un péndulo, el cual, es de la forma:

$$\mathcal{L} = \frac{1}{2}ml^2\dot{\theta} + mgl\cos\theta \tag{17}$$

Usando la ecuación de Euler-Lagrange la ecuación de movimiento será:

$$l\ddot{\theta} + g\sin\theta = 0 \tag{18}$$

Si θ es en ángulo muy pequeño (aproximadamente menor a 10°), entonces, $\sin\theta \approx \theta$ y la anterior ecuación se reduce a:

$$l\ddot{\theta} + g\sin\theta = 0 \tag{19}$$

el cual es la ecuación diferencial de un oscilador armónico que para este caso tiene un periodo de oscilación de:

$$T_0 = 2\pi\sqrt{\frac{l}{g}} \tag{20}$$

Sin embargo, si esta amplitudes son muy grandes la solución de la ecuación diferencial es más complicada e involucra integrales elípticas, la ecuación que expresa el período para ángulos grandes es de la forma:

$$T(\theta) = T_0\left[\sum_{n=0}^{\infty}\left(\frac{(2n)!}{2^{2n}(n!)^2}\right)\sin^{2n}\left(\frac{\theta}{2}\right)\right] \tag{21}$$

que sería el resultado analítico de la ecuación (1). Cabe resaltar que en el cuadro 1 solo se refleja un cambio considerable en el error relativo cuando $\pi/2$ pero eso solo es consecuencia del hecho de que se usa $n = 1024$. Se puede mejorar la precisión si hacemos $n = 10^6$ (véase cuadro 2).

θ	**Valor de la integral**	**error relativo a límite de ángulos pequeños**
$\pi/128$	2,005369	0,000857
$\pi/64$	2,005596	0,000744
$\pi/32$	2,006502	0,000293
$\pi/16$	2,010135	0,0015171
$\pi/8$	2,024788	0,008818
$\pi/4$	2,085427	0,039030
$\pi/2$	2,366799	0,179219

Cuadro 2: Cuadro de comparación entre el valor de la integral númerica para distintos valores de ángulo y el valor del periodo para pequeñas oscilaciones ($n = 10^6$).

En la anterior tabla se puede evidenciar que los errores son apreciables para valores iguales o superiores a $\pi/4$, y eso solo se debe al hecho de que estos ángulos son demasiado grandes y hacen que el péndulo deje de tener un comportamiento de oscilador armónico.

El código fuente final usado para este ejercicio es:

```
import numpy as np #importar módulo numpy

theta_0=np.pi/2 #amplitud que puede variarse
```

```
g=9.8 #aceleración de la gravedad
l=1 #logitud del péndulo
T_lim=2*np.pi*np.sqrt(l/g)#<----límite de pequeños ángulos

n=1024 #se fija el número de subintervalos

theta=np.linspace(0,theta_0,n+1) #intervalo de integración
theta=[i for i in theta] #transformación del arreglo de numpy a lista
theta.pop(len(theta)-1) #eliminación del último punto theta (divergencia)
theta=np.array(theta) #transformación de lista a arreglo de numpy
h=theta[1]-theta[0] #diferencia de valores en theta

alpha=h/3 #factor alpha

beta=4*pow(l/(2*g),1/2) #factor beta

#definición de la función
f=beta/(pow(np.cos(theta)-np.cos(theta_0),1/2))

Int=0 #inicialización de la integral en 0


#integral generada por medio de búcle for
for j in range(int(n/2)): #desde 0 hasta (n/2)-1
    if j==int((n/2))-1: # último slice
        Int+=(h/12)*(-f[n-3]+8*f[n-2]+5*f[n-1]) #contribución de último slice
    else:
        Int+=alpha*(f[2*j]+4*f[2*j+1]+f[2*j+2]) #regla de Simpson

error=abs(1-Int/T_lim)#<--- error relativo

print("Resultado de la integral (regla de Simpson): %f"%Int)
print("Periodo de oscilación en el límite de ángulos pequeños: %f"%T_lim)
print("Error relativo al límite de ángulos pequeños: %f"%error)
#se mostrará por pantalla el resultado
```

4. Solución del problema 3

Un polinomio de Lagrange se define como:

$$f(x) = \sum_{j=0}^{k} f_j P_j, \tag{22}$$

donde k es el orden del polinomio de Lagrange usado para realizar interpolación de datos y,

$$P_j(x) = \prod_{i \neq j}^{k} \frac{x - x_i}{x_j - x_i}. \tag{23}$$

El problema pide hacer uso de una interpolación con un polinomio de Lagrange de orden 3. Haciendo uso de la ecuación (22) tendremos:

$$f(x) = f_0 P_0 + f_1 P_1 + f_2 P_2 + f_3 P_3, \tag{24}$$

donde:

$$
\begin{aligned}
P_0(x) &= \frac{x-x_1}{x_0-x_1}\frac{x-x_2}{x_0-x_2}\frac{x-x_3}{x_0-x_3},\\
P_1(x) &= \frac{x-x_0}{x_1-x_0}\frac{x-x_2}{x_1-x_2}\frac{x-x_3}{x_1-x_3},\\
P_2(x) &= \frac{x-x_0}{x_2-x_0}\frac{x-x_1}{x_2-x_1}\frac{x-x_3}{x_2-x_3},\\
P_3(x) &= \frac{x-x_0}{x_3-x_0}\frac{x-x_1}{x_3-x_1}\frac{x-x_2}{x_3-x_2}.
\end{aligned}
\tag{25}
$$

Por lo tanto, nuestra función $f(x)$ es de la forma:

$$
\begin{aligned}
f(x) = \frac{x-x_1}{x_0-x_1}\frac{x-x_2}{x_0-x_2}\frac{x-x_3}{x_0-x_3}f_0 + \frac{x-x_0}{x_1-x_0}\frac{x-x_2}{x_1-x_2}\frac{x-x_3}{x_1-x_3}f_1 \\
+ \frac{x-x_0}{x_2-x_0}\frac{x-x_1}{x_2-x_1}\frac{x-x_3}{x_2-x_3}f_2 + x - x_0x_3 - x_0\frac{x-x_1}{x_3-x_1}\frac{x-x_2}{x_3-x_2}f_3.
\end{aligned}
\tag{26}
$$

Definimos las siguientes cantidades:

- $h_i = x_1 - x_0$,
- $h_{i+1} = x_2 - x_1$,
- $h_{i+2} = x_3 - x_2$.

También se definirá:

- $x_0 = x_i$,
- $x_1 = x_{i+1}$,
- $x_2 = x_{i+2}$,
- $x_3 = x_{i+3}$,

y finalmente, $\hat{x} = x - x_0$. Con lo cual se obtendrá:

- $x - x_1 = \hat{x} - h_i$
- $x - x_2 = = \hat{x} - (h_i + h_{i+1})$
- $x - x_3 = \hat{x} - (h_i + h_{i+1} + h_{i+2})$

Y también:

- $x_0 - x_1 = -h_i$
- $x_0 - x_2 = -(h_i + h_{i+1})$
- $x_0 - x_3 = -(h_i + h_{i+1} + h_{i+2})$
- $x_1 - x_0 = h_i$
- $x_1 - x_2 = -h_{i+1}$
- $x_1 - x_3 = -(h_{i+1} + h_{i+2})$
- $x_2 - x_0 = h_i + h_{i+1}$
- $x_2 - x_1 = h_{i+1}$
- $x_2 - x_3 = -h_{i+2}$
- $x_3 - x_0 = h_i + h_{i+1} + h_{i+2}$

- $x_3 - x_1 = h_{i+1} + h_{i+2}$
- $x_3 - x_2 = h_{i+2}$

Si reemplazamos todo los anterior en la ecuación (26) tendremos:

$$f(\hat{x}) = \frac{\hat{x}-h_i}{-h_i}\frac{\hat{x}-(h_i+h_{i+1})}{-(h_i+h_{i+1})}\frac{\hat{x}-(h_i+h_{i+1}+h_{i+2})}{-(h_i+h_{i+1}+h_{i+2})}f_i + \frac{\hat{x}}{h_i}\frac{\hat{x}-(h_i+h_{i+1})}{-h_{i+1}}\frac{\hat{x}-(h_i+h_{i+1}+h_{i+2})}{-(h_{i+1}+h_{i+2})}f_{i+1}$$

$$+ \frac{\hat{x}}{h_i+h_{i+1}}\frac{\hat{x}-h_i}{h_{i+1}}\frac{\hat{x}-(h_i+h_{i+1}+h_{i+2})}{-h_{i+2}}f_{i+2} + \frac{\hat{x}}{h_i+h_{i+1}+h_{i+2}}\frac{\hat{x}-h_i}{h_{i+1}+h_{i+2}}\frac{\hat{x}-(h_i+h_{i+1})}{h_{i+2}}f_{i+3}. \tag{27}$$

Eliminando algunos signos tenemos:

$$f(\hat{x}) = -\frac{\hat{x}-h_i}{h_i}\frac{\hat{x}-(h_i+h_{i+1})}{h_i+h_{i+1}}\frac{\hat{x}-(h_i+h_{i+1}+h_{i+2})}{h_i+h_{i+1}+h_{i+2}}f_i + \frac{\hat{x}}{h_i}\frac{\hat{x}-(h_i+h_{i+1})}{h_{i+1}}\frac{\hat{x}-(h_i+h_{i+1}+h_{i+2})}{h_{i+1}+h_{i+2}}f_{i+1}$$

$$- \frac{\hat{x}}{h_i+h_{i+1}}\frac{\hat{x}-h_i}{h_{i+1}}\frac{\hat{x}-(h_i+h_{i+1}+h_{i+2})}{h_{i+2}}f_{i+2} + \frac{\hat{x}}{h_i+h_{i+1}+h_{i+2}}\frac{\hat{x}-h_i}{h_{i+1}+h_{i+2}}\frac{\hat{x}-(h_i+h_{i+1})}{h_{i+2}}f_{i+3}. \tag{28}$$

Ulteriormente, el polinomio $f(x)$ se integrará como se muestra a continuación:

$$\int_{x_i}^{x_{i+3}} f(x)dx = \int_0^{h_i+h_{i+1}+h_{i+2}} f(\hat{x})d\hat{x} \tag{29}$$

porque:

$$\hat{x}|_{x_i} = x_i - x_i = 0 \text{ y } \hat{x}|_{x_{i+3}} = x_{i+3} - x_i = h_i + h_{i+1} + h_{i+2} \tag{30}$$

se hace esta integral con ayuda de Mathematica, el código usado es el siguiente:

```
(*definición de la función f(x)=g(x)*)
g = -((x - Subscript[h, i])/Subscript[h, i])*(
   x - (Subscript[h, i] + Subscript[h, i + 1]))/(
   Subscript[h, i] + Subscript[h, i + 1])*(
   x - (Subscript[h, i] + Subscript[h, i + 1] +
      Subscript[h, i + 2]))/(
   Subscript[h, i] + Subscript[h, i + 1] + Subscript[h, i + 2])*
   Subscript[f, i] +
  x/Subscript[h, i]*(x - (Subscript[h, i] + Subscript[h, i + 1]))/
   Subscript[h, i + 1]*(
   x - (Subscript[h, i] + Subscript[h, i + 1] +
      Subscript[h, i + 2]))/(
   Subscript[h, i + 1] + Subscript[h, i + 2])*Subscript[f, i + 1] -
  x/(Subscript[h, i] + Subscript[h, i + 1])*(x - (Subscript[h, i]))/
   Subscript[h, i + 1]*(
   x - (Subscript[h, i] + Subscript[h, i + 1] + Subscript[h, i + 2]))/
   Subscript[h, i + 2]*Subscript[f, i + 2] +
  x/(Subscript[h, i] + Subscript[h, i + 1] + Subscript[h, i + 2])*(
   x - Subscript[h, i])/(Subscript[h, i + 1] + Subscript[h, i + 2])*(
   x - (Subscript[h, i] + Subscript[h, i + 1]))/Subscript[h, i + 2]*
   Subscript[f, i + 3]

   (*integral resuelta*)
   Integrate[g, {x, 0,
  Subscript[h, i] + Subscript[h, i + 1] + Subscript[h, i + 2]}]
```

Cuyo output fue:

$$\int_0^{h_i+h_{i+1}+h_{i+2}} f(\hat{x})d\hat{x} = \frac{1}{12}\left(h_i+h_{i+1}+h_{i+2}\right)\left(\frac{(3h_i^2-h_{i+1}^2+2h_i(h_{i+1}-h_{i+2})+h_{i+2}^2)}{h_i(h_i+h_{i+1})}f_i\right.$$
$$+\frac{(h_i+h_{i+1}-h_{i+2})(h_i+h_{i+1}+h_{i+2})^2}{h_ih_{i+1}(h_{i+1}+h_{i+2})}f_{i+1}+\frac{(-h_i+h_{i+1}+h_{i+2})(h_i+h_{i+1}+h_{i+2})^2}{h_{i+1}(h_i+h_{i+1})h_{i+2}}f_{i+2}$$
$$\left.+\frac{(h_i^2-h_{i+1}^2+2(-h_i+h_{i+1})h_{i+2}+3h_{i+2}^2)}{h_{i+2}(h_{i+1}+h_{i+2})}f_{i+3}\right) \tag{31}$$

la integral total será:

$$\sum_{J=0}^{n/3-1}\left(a_{1,J}f_{3J}+a_{2,J}f_{3J+1}+a_{3,J}f_{3J+2}+a_{4,J}f_{3J+3}\right) \tag{32}$$

donde $a_{1,J}$ es el factor que acompaña a f_{3J}, $a_{2,J}$ es el factor que acompaña a f_{3J+1}, $a_{3,J}$ es el factor que acompaña a f_{3J+2} y $a_{4,J}$ es el factor que acompaña a f_{3J+3} [3].La anterior integral númerica funcionará para valores n que sean multiplos de 3.

Q.E.D.

5. Solución del problema 4

La solución del problema se encuentra en el archivo llamado *punto_4.py*. Para resolver el problema se hizo necesario hacer uso de una extrapolación lineal en el último punto por medio de la siguiente fórmula:

$$y(x_*) = y_{k-1} + \frac{x_*-x_{k-1}}{x_k-x_{k-1}}\left(y_k-y_{k-1}\right) \tag{33}$$

donde y_{k-1} el el penúltimo dato, y_k es el último dato y x_* es el último dato que queremos extrapolar. Para que el error del cálculo númerico sea pequeño (usando la extrapolación) el punto x_* debe estar muy cerca del punto x_k. Idealmente en el límite cuando $x_* - x_k$ tienda a cero el error debido a la extrapolación será totalmente despreciable, por esto se hace a $x_* = 1{,}816890$ por que $x_k = 1{,}816889$.

Para comparar la regla del trapecio con la integral númerica con polinomio de grado 3 es necesario hallar la regla del trapecio con paso adaptativo. El método de encontrar dicha expresión se muestra a continuación:

$$f(x) = f_i + \frac{x-x_i}{x_{i+1}-x_i}(f_{i+1}-f_i) \tag{34}$$

Si decimos que $\hat{x} = x - x_i$ y $h_i = x_{i+1} - x_i$, entonces:

$$f(\hat{x}) = f_i + \frac{\hat{x}}{h_i}(f_{i+1}-f_i) \tag{35}$$

la integral que hay que resolver es:

$$\int_{x_i}^{x_{i+1}} f(x)dx = \int_0^{h_i} f(\hat{x})d\hat{x} \tag{36}$$

lo anterior se debe a que:

$$\hat{x}|_{x_i} = x_i - x_i = 0 \text{ y } \hat{x}|_{x_{i+1}} = x_{i+1} - x_i = h_i \tag{37}$$

Cuyo resultado es:

$$\int_0^{h_i} f(\hat{x})d\hat{x} = \frac{h_i}{2}\left(f_i+f_{i+1}\right) \tag{38}$$

Es decir, la integral completa será:

$$S = \frac{1}{2}\sum_{i=0}^{n-1} h_i\left(f_i+f_{i+1}\right) \tag{39}$$

La anterior ecuación es usada en el programa *punto_4.py*.

En conclusión, el error de cálculo de la integral usando el método del trapezoide el mayor que el método de Simpson usando una interpolación de orden 3, pues el programa *punto_4.py* dio el siguiente resultado:

[3] $a_{1,J} es el factor que acompaña a f_i$ pero teniendo en cuenta que $i = 3J$

```
Integral de la interpolación grado 3: 1.393454
Integral de la interpolación grado 1: 1.395652
Error interpolación grado 3: 0.005942
Error interpolación grado 1: 0.007529
```