Métodos Numéricos con *Calc* y OpenOffice.org Basic (OOoBasic).

Walter Mora F.
wmora2@yahoo.com.mx
Escuela de Matemática
Instituto Tecnológico de Costa Rica

Palabras claves: Métodos numéricos, ceros de funciones, métodos iterativos, software libre, OpenOffice.org, Calc, OpenOffice.org Basic.

Introducción

OpenOffice.org 3.x (`http://es.openoffice.org/`) es una suite ofimática (procesador de textos, hoja de cálculo, presentaciones, etc.) de software libre. Esto significa que usted puede usar el software como quiera. Puede copiarlo, regalarlo y distribuirlo a quien quiera sin pagar costosas licencias. En todo caso, puede ser que usar OpenOffice.org, además de ser una suite muy eficiente, tenga que ver con una cuestión de responsabilidad social y un asunto ético: usar sofware legal y sin costo. OpenOffice.org está disponible para muchas plataformas como Windows y sistemas de tipo Unix como Linux y Mac OS X. OpenOffice está pensado para ser altamente compatible con Microsoft Office, de hecho OpenOffice es capaz de utilizar la mayoría de las plantillas de Microsoft Office.

Figura 1.1 Inicio de OpenOffice.org

En lo que nos concierne, vamos a usar OOoBasic y Calc. Calc es una hoja de cálculo similar a Excel y suponemos que el lector conoce programación básica en VBA para Excel. La programación de Macros la haremos con OpenOffice.org Basic (OOoBasic), muy similar a VBA. No se puede decir, en realidad, si es más sencillo usar VBA u OOoBasic; ambos hacen las cosas sencillas, pero a su manera.

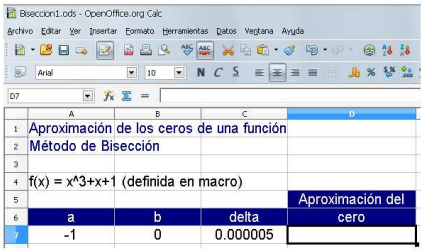

Figura 1.2 OOo Calc

VBA y OOoBasic son lenguajes de programación de la familia "Basic". Como tal, comparten los mismos constructores linguísticos básicos, por ejemplo, declaración de variables, ciclos (For, Do, While, Do While, Do Until,...), condicionales, operadores lógicos, funciones, etc. La diferencia está en el API, es decir, la manera de accesar, crear, modificar, guardar, etc. los documentos de OpenOffice.org. Por tanto, si alguien ya está familiarizado con algún lenguaje de de la familia Basic (por ejemplo, VBA), se setirá cómodo con OOoBasic.
Este es un material introductorio: El código de los ejemplos solo tiene fines didácticos, por lo tanto no se hace manejo de excepciones, para no agregar complejidad inecesaria en este nivel.

1.1 Leer e imprimir en una celda con OOoBasic.

Como nos sugiere el libro de la figura (1.2), lo primero que tenemos que aprender es cómo leer datos de las celdas y como imprimir los resultados en una o varias celdas. Con este propósito, vamos a crear una subrutina de prueba, llamada "`LeerImprimir()`", para leer un valor x_0 en la celda "A4" e imprimir $f(x_0)$ en la celda "A5". La función f será $f(x) = x^3 + x + 1$. Tanto la función como la subrutina la vamos a incluir en un módulo llamado "`Module1`" y la acción de leer en la celda, calcular e imprimir la vamos ejecutar desde un botón "Calcular", tal y como se ve en la figura (1.3).

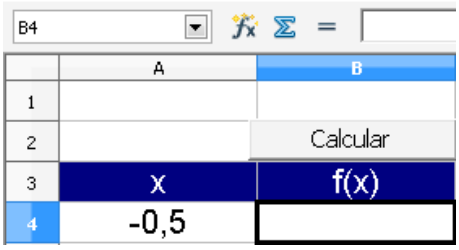

Figura 1.3 Leer e imprimir en una celda

Para Realizar todo esto procedemos como sigue:

1. Abrimos un libro OOo Calc y los guardamos digamos, como "LeerImprimir". Así, tendremos un archivo `LeerImprimir.odf` (el cual puede abrir en windows, Linux o Mac!).

2. Si es necesario, cambiamos el idioma en

   ```
   Herramienta>Opciones>Configuración de idioma>Idiomas.
   ```

3. Editamos: Ponemos etiquetas `x, f(x)` y el valor `-0,5` en la celda "A4"

4. Ahora vamos a insertar un módulo llamado "`Module1`" (su nombre default) para editar la subrutina `LeerImprimir()` y la función f. Para esto vamos a

`Herramientas>Macros>Organizar macros>OpenOffice.org Basic` y en la ventana `Macros básicas` (de manera directa: `Alt-F11`). Seleccionamos nuesta hoja `LeerImprimir` y hacemos clic en 'Nuevo', hacemos clic en `Aceptar`. Esto nos llevará al editor OOoBasic.

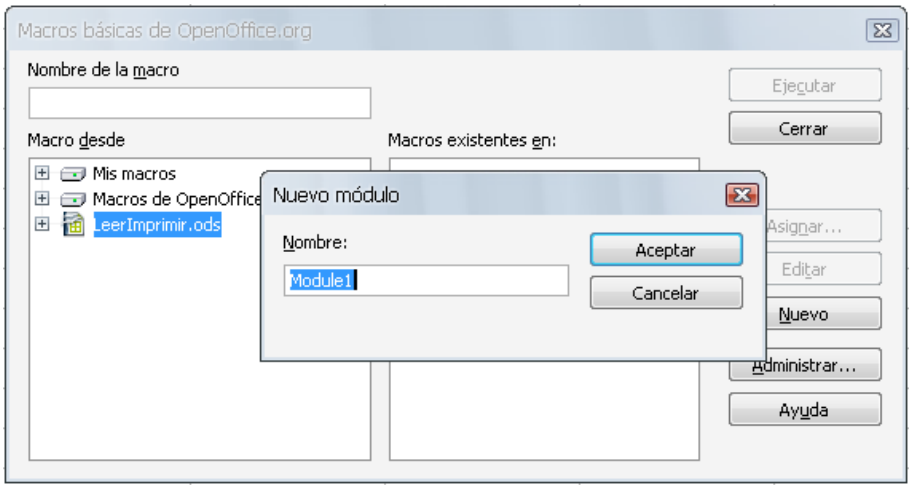

Figura 1.4 Insertar Módulo

5. La función la editamos igual que en VBA:

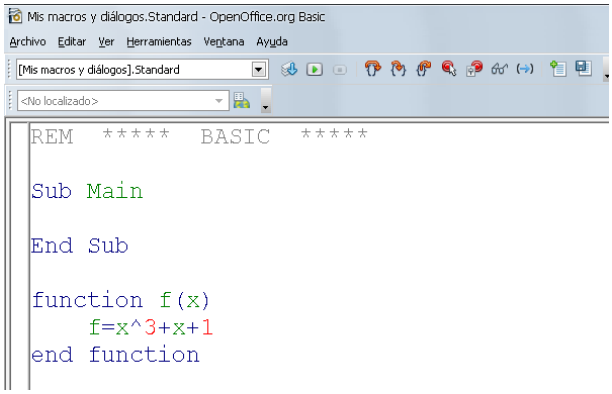

Figura 1.5 Edición de $f(x) = x^3 + x + 1$

6. Para implementar la subrutina `LeerImprimir()` debemos declarar dos variables: `Hoja` y `Celda` para hacer referencia a la hoja actual y a la celda actual.

Aquí debemos recurrir al API de OOoBasic para entender la manera de comunicanos on Calc. Podemos usar los libros de la bibliografía o también usar recursos de internet como `http://api.openoffice.org/`. Por ejemplo, para informarse sobre las propiedades de `cell` vamos a

`http://api.openoffice.org/docs/common/ref/index-files/index-1.html`

Aquí encontramos

Methods' Summary	
getCellByPosition	Returns a single cell within the range.
getCellRangeByPosition	Returns a sub-range of cells within the range.
getCellRangeByName	Returns a sub-range of cells within the range.

Figura 1.6 API de OOoBasic

En Calc OOoBasic, las celdas se reconocen por su columna y su fila, en este orden. Solo hay que tener en cuenta que la celda "A1" corresponde a $(0,0)$.

Primero hacemos que la variable `Hoja` apunte a la hoja de trabajo, en nuestro caso ponemos:

```
    Dim Hoja
        Hoja  = thisComponent.Sheets(0)
```

Ahora debemos accesar la celda "A4"; esto se puede hacer usando el nombre de la celda

```
        Celda = Hoja.getCellRangeByName("A4")
```

o también usando su posición

```
                'Celda A4 = columna 0, fila 3
        Celda = Hoja.getCellByPosition(0, 3)
```

Ahora ya podemos poner el código de la subrutina, `LeerImprimir()`. Recordemos que al llamar esta subrutina, se lee x en la celda "A4" y se imprime $f(x)$ en la celda "A5":

```
Option Explicit
sub LeerImprimir()
Dim Hoja, Celda
Dim x
   Hoja        = thisComponent.Sheets(0)
   Celda       = Hoja.getCellRangeByName("A4") 'Celda A4
       x       = Celda.Value 'Leer
   Celda       = Hoja.getCellByPosition(1, 3)  'Celda B4
   Celda.Value = f(x)        'Escribir
end sub
```

```
REM  *****  BASIC  *****
Sub Main
End Sub

function f(x)
    f=x^3+x+1
end function

sub LeerImprimir()
Dim Hoja, Celda
Dim x
   Hoja         = thisComponent.Sheets(0)
   Celda        = Hoja.getCellRangeByName("A4") 'Celda A4
        x       = Celda.Value
   Celda        = Hoja.getCellByPosition(1, 3)  'Celda B4
   Celda.Value  = f(x)
end sub
```

Figura 1.7 Código del Módulo 1

7. Ahora sigue insertar el botón que dispara el cálculo. Debemos tener visible la barra `"Campos de control de formulario".` Si no esta visible, se habilita con
   `Ver>Barra de herramientas>Campos de control de formulario.`
   Para crear un botón, seleccionamos el botón en la barra y arrastramos el mouse en una celda. Para poner la etiqueta "Calcular" abrimos la ventana de propiedades del botón (con clic derecho sobre el botón, abrimos el menú "`Campo de control`") y editamos el campo "`Título`". Ver figura (1.8)

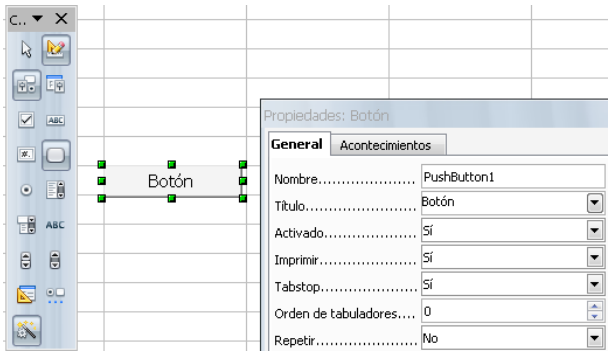

Figura 1.8 Agregar un botón

Ahora, seleccionamos la cejilla "`Acontecimientos`" y asignamos una acción: "`Al iniciar`" (cuando el usuario hace clic en el botón). Elegimos la macro, es decir, la función o subrutina que se va a ejecutar cuando el usuario hace clic en el botón. En nuestro caso, seleccionamos la subrutina `LeerImprimir()`.

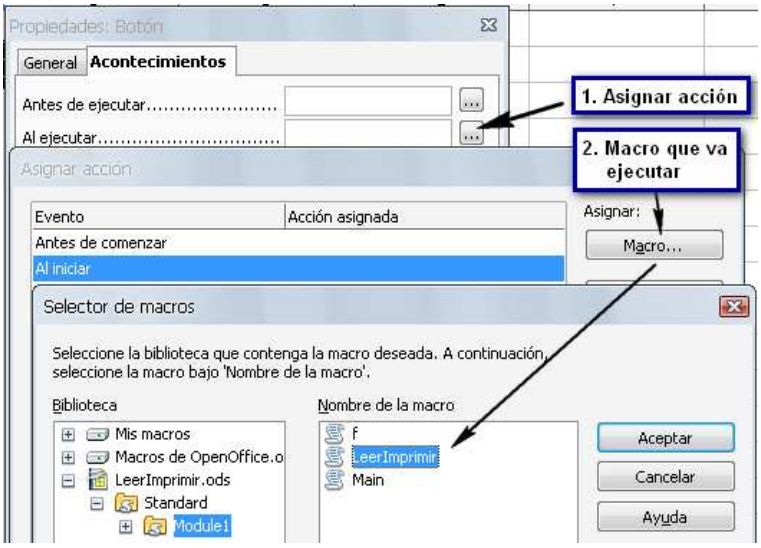

Figura 1.9 Asignar la subrutina que se ejecuta cuando se hace clic en el botón.

Luego de hacer clic en "`Aceptar`" y cerrar la ventana "`Propiedades: Botón`", debemos habilitar el botón (que hasta ahora ha estado en "modo diseño"), Para hacer esto, hacemos clic en  (en la barra "`Campos de control de formulario`"). Ahora ya podemos hacer clic en el botón para realizar el cálculo e imprimir.

Ahora que ya sabemos leer y escribir en una celda y ejecutar una subrutina des un botón, podemos aplicar este conocimiento para implementar dos algoritmos sencillos: El método de Bisección para aproximar un cero de una función y el método del trapecio para aproximar una integral definida.

1.2 Método de Bisección

Este es un método sencillo y robusto para resolver ecuaciones en una variable. Se basa en el Teorema de los Valores Intermedios, el cual establece que toda función continua f en un intervalo cerrado $[a,b]$ ($f \in C[a,b]$) toma todos los valores que se hallan entre $f(a)$ y $f(b)$. Esto es, que todo valor entre $f(a)$ y $f(b)$ es la imagen de al menos un valor en el intervalo $[a,b]$. En caso de que $f(a)$ y $f(b)$ tengan signos opuestos (es decir, $f(a) \cdot f(b) < 0$), el valor cero sería un valor intermedio entre $f(a)$ y $f(b)$, por lo que con certeza existe un x^* en $]a,b[$ que cumple $f(x^*) = 0$. De esta forma, se asegura la existencia de al menos una solución de la ecuación $f(x) = 0$.

El método consiste en lo siguiente: Supongamos que en el intervalo $[a,b]$ hay un cero de f. Calculamos el punto medio $m = (a+b)/2$ del intervalo $[a,b]$. A continuación calculamos $f(m)$. En caso de que ocurra el milagro de que $f(m) = 0$ ya hemos encontrado la solución buscada. En caso de que no lo sea, verificamos si $f(m)$ tiene signo opuesto al de $f(a)$. Se redefine el intervalo $[a,b]$ como $[a,m]$ o $[m,b]$ según se haya determinado en cuál de estos intervalos ocurre un cambio de signo. A este nuevo intervalo se le aplica el mismo procedimiento y así, sucesivamente, iremos encerrando la solución en un intervalo cada vez más pequeño, hasta alcanzar la precisión deseada.

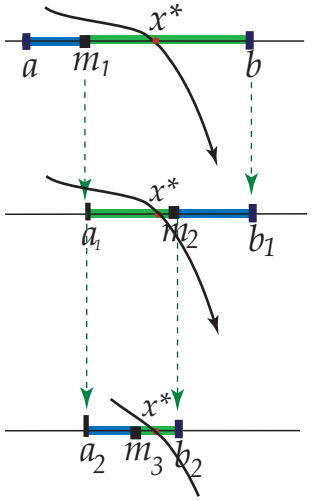

Figura 1.10 Método de Bisección

El método construye tres sucesiones a_n, b_n y m_n,

- Inicio: $a_0 = a$ y $b_0 = b$ y $m_1 = (a_0 + b_0)/2$

- Para $k = 0, 1, 2...$, $m_{k+1} = \dfrac{b_k + a_k}{2}$ y

$$[a_{k+1}, b_{k+1}] = \begin{cases} [m_{k+1}, b_k] & \text{si} \quad f(a_k)f(m_{k+1}) > 0 \\ [a_k, m_{k+1}] & \text{si} \quad f(a_k)f(m_{k+1}) < 0 \end{cases}$$

Estimación del error:

El error exacto en la iteración $k+1$ es $|m_{k+1} - x^*|$. Geométricamente se puede ver que esto es menos que la mitad del intervalo $[a_k, b_k]$, es decir

$$|m_{k+1} - x^*| \leq \frac{b_k - a_k}{2} = \frac{b-a}{2^{k+1}}$$

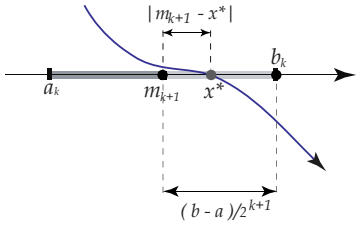

Figura 1.11 Estimación del error en bisección.

1.2.1 Algoritmo e Implementación en OOoBasic

En la implementación del método de bisección, en vez de usar $f(a)f(m) < 0$ ponemos $sgn(f(a)) <> sgn(f(m))$, de esta manera nos ganamos una multiplicación (que es claramente innecesaria). En OOoBasic, $sgn(0) = 0$, así que no hay problema si se presenta este caso.

En bisección es mejor calcular m como $m = a + (b-a)/2$. Con esto somos consistentes con la estrategia general (en análisis numérico) de calcular una cantidad agregando una corrección a la aproximación anterior. Además ganamos algo en precisión.

El criterio de parada es: dada una tolerancia $\delta > 0$ detenerse en m_k si

$$\frac{b_k - a_k}{2} = \frac{b-a}{2^{k+1}} \leq \delta$$

Algoritmo 1.1: Algoritmo de Bisección.

Entrada: a, b, δ y f continua en [a,b] con $f(a)f(b) < 0$.
Resultado: Una aproximación m de un cero x^* de f en $]a,b[$.

```
k = 0;
repeat
    m = a + 0.5(b − a);
    dx = (b − a)/2;
    if Sgn(f(a)) <> Sgn(f(m)) then
        b = m;
    else
        a = m
    k = k + 1
until dx ≤ δ or f(m) = 0 ;
return m
```

Implementación OOoBasic.

La implementación en OOoBasic que hacemos aquí más bien es con fines didácticos: En vez de reportar la última aproximación m_k, vamos a imprimir cuatro columnas con los valores de m_k, a_{k-1}, b_{k-1} y la estimación del error $(b-a)/2^k$. Con este fin, implementamos una subrutina `SubBiseccion()` que hace todo el trabajo: Lee a, b y $delta$ e imprime los resultados intermedios. El libro que vamos a usar se puede ver en la figura (1.12).

F18

	A	B	C	D	E	F	G
1	Método de Bisección						
2	f(x)= x^3+x+1, (en el código)						
3				Calcular			
4							Error
5	a	b	delta	a_k	b_k	m_(k+1)	Estimado
6	-0,7	-0,6	0,00005	-0,70000000000	-0,65000000000	-0,65000000000	0,05000000000
7				-0,70000000000	-0,67500000000	-0,67500000000	0,02500000000
8				-0,68750000000	-0,67500000000	-0,68750000000	0,01250000000
9				-0,68750000000	-0,68125000000	-0,68125000000	0,00625000000
10				-0,68437500000	-0,68125000000	-0,68437500000	0,00312500000
11				-0,68281250000	-0,68125000000	-0,68281250000	0,00156250000
12				-0,68281250000	-0,68203125000	-0,68203125000	0,00078125000
13				-0,68242187500	-0,68203125000	-0,68242187500	0,00039062500

Figura 1.12 Cuaderno SubBisección.

Esta vez vamos a simplificar un poco más el código, por ejemplo, para leer el valor de a usamos el código

```
Dim Hoja
Dim a
 Hoja  = thisComponent.Sheets(0)
    a  = Hoja.getCellRangeByName("A6").Value
```

El bucle `Repeat ... Until` corresponde a `Do... Loop Until`. Como antes, insertamos el código en un módulo "`Module 1`" u otro nombre. El código que debemos entrar en el módulo sería:

```
function f(x)
   f= x^3+x+1
end function

Sub subBiseccion()
Dim Hoja, Celda
Dim a,b,delta,k,dx,m
Hoja  = thisComponent.Sheets(0)
        'Leemos 'a' en A6
    a = Hoja.getCellRangeByName("A6").Value
    b = Hoja.getCellRangeByName("B6").Value
delta = Hoja.getCellRangeByName("C6").Value

k=0
Do
    m=a+0.5*(b-a)
    dx=(b-a)/2
    If Sgn(f(a))<>Sgn(f(m)) Then
         b=m
    Else a=m
    End If
    Celda = Hoja.getCellByPosition(3,5+k) 'D6, D7,...
    Celda.Value= a
    Celda = Hoja.getCellByPosition(4,5+k) 'E6, E7,...
    Celda.Value= b
    Celda = Hoja.getCellByPosition(5,5+k) 'F6, F7,...
```

```
    Celda.Value= m
    Celda = Hoja.getCellByPosition(6,5+k) 'G6, G7,...
    Celda.Value= dx
    k=k+1
Loop Until dx<=delta or f(m)=0
End Sub
```

Luego le asignamos esta subrutina al botón como hicimos en la sección anterior.

1.3 Integración: Método del Trapecio

En la regla del trapecio, para aproximar $\int_a^b f(x)\,dx$ dividimos el intervalo $[a,b]$ en n subintervalos: Si $h=(b-a)/n$ y $x_i=a+i\,h$, en cada subitervalo $[x_i,\,x_{i+1}]$, cambiamos la función f por el polinomio interpolante de grado 1 (figura 1.13).

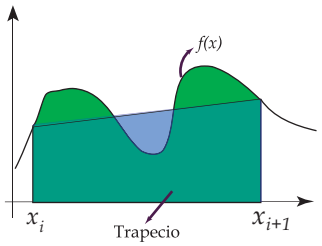

Figura 1.13 Regla del trapecio

$$\int_a^b f(x)\,dx=\sum_{i=0}^{n-1}\int_{x_i}^{x_{i+1}} f(x)\,dx$$

Para aproximar cada integral $\int_{x_i}^{x_{i+1}} f(x)\,dx$ necesitamos el polinomio que interpola a f en $(x_i,f(x_i))$, $(x_{i+1},f(x_{i+1}))$:

$$P(x) = f(x_i)\,\frac{(x - x_{i+1})}{h} \;+\; f(x_{i+1})\,\frac{(x - x_i)}{h} + E,$$

con $E = (x - x_i)(x - x_{i+1})\dfrac{f''(\xi(x))}{2}$ (si f'' es continua en $[x_i, x_{i+1}]$).

$$\begin{aligned}\int_{x_i}^{x_{i+1}} f(x)\,dx &= \int_{x_i}^{x_{i+1}} f(x_i)\,\frac{(x - x_{i+1})}{h} \;+\; f(x_{i+1})\,\frac{(x - x_i)}{h} + E\;\,dx \\ &= \frac{h}{2}\,[f(x_i) + f(x_{i+1})] + \int_{x_i}^{x_{i+1}} (x - x_i)(x - x_{i+1})\frac{f''(\xi(x))}{2}\;\,dx\end{aligned}$$

Para calcular $\displaystyle\int_{x_i}^{x_{i+1}} E\,dx$ necesitamos recordar el teorema del valor medio para integrales: Si en $[x_i, x_{i+1}]$ G es continua y φ integrable y de un mismo signo, entonces existe $\eta_i \in]x_i, x_{i+1}[$ tal que $\displaystyle\int_{x_i}^{x_{i+1}} G(x)\varphi(x)\,dx = G(\eta_i)\int_{x_i}^{x_{i+1}} \varphi(x)\,dx$.

Ahora, poniendo $G(x) = f''(\xi(x))/2$ y $\varphi(x) = (x - x_i)(x - x_{i+1})$, obtenemos

$$\int_{x_i}^{x_{i+1}} (x - x_i)(x - x_{i+1})\frac{f''(\xi_x)}{2}\;\,dx \;=\; -\frac{h^3}{12} f''(\eta_i),\;\; \eta_i \in]x_i, x_{i+1}[.$$

Observe que $(x - x_i)(x - x_{i+1})$ tiene el mismo signo sobre $[x_i, x_{i+1}]$ (siempre es negativa) y que $\displaystyle\int_{x_i}^{x_{i+1}} (x - x_i)(x - x_{i+1})\,dx = -h^3/6$.

Finalmente:

$$\int_{x_i}^{x_{i+1}} f(x)\,dx \;=\; \frac{h}{2}\,[f(x_i) + f(x_{i+1})] - \frac{h^3}{12} f''(\eta_i),\;\; \varepsilon_i \in [x_i, x_{i+1}]. \qquad (1.1)$$

Si ponemos, para abreviar los cálculos, $G_i(x) = P_i(x) + (x - x_i)(x - x_{i+1})f''(\xi_i(x))/2$, con $P_i(x)$ el polinomio (lineal) interpolante en $[x_i, x_{i+1}]$, entonces

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_{x_0}^{x_1} G_0(x)\,dx + \int_{x_1}^{x_2} G_1(x)\,dx + ... + \int_{x_{n-1}}^{x_n} G_{n-1}(x)\,dx \\ &= \frac{h}{2}\left(f(x_0) + f(x_n) + 2\sum_{i=1}^{n-1} f(x_i)\right) - \frac{h^3}{12}\sum_{k=0}^{n-1} f''(\eta_k)\end{aligned}$$

donde $h = \dfrac{b-a}{n}$, $\eta_i \in [x_i,\, x_{i+1}]$ y $x_i = a + i \cdot h,\ i = 0, 1, ..., n$.

Podemos simplificar la fórmula observando que

$$-\frac{h^3}{12}\sum_{k=0}^{n-1} f''(\eta_k) = -\frac{h^2}{12}\cdot(b-a)\left[\frac{\sum_{k=0}^{n-1} f''(\eta_k)}{n}\right]$$

La expresión en paréntesis cuadrados es un *promedio* de los valores de f'' en $[a, b]$, por lo tanto este promedio está entre el máximo y el mínimo absoluto de f'' en $[a, b]$ (asumimos f'' continua). Finalmente, por el teorema del valor intermedio, existe $\xi \in]a, b[$ tal que $f''(\xi)$ es igual a este valor promedio, es decir

$$-\frac{h^3}{12}\sum_{k=0}^{n-1} f''(\eta_k) = -\frac{(b-a)h^2}{12}\cdot f''(\xi),\ \xi \in]a, b[$$

Regla compuesta del trapecio:

$$\int_a^b f(x)\,dx = \frac{h}{2}\left(f(a) + f(b) + 2\sum_{i=1}^{n-1} f(x_i)\right) - \frac{(b-a)h^2}{12}\cdot f''(\xi)$$

con $\xi \in]a, b[,\ h = \dfrac{b-a}{n}$ y $x_i = a + i \cdot h,\ i = 0, 1, 2, ..., n$.

1.3.1 Algoritmo e Implementación en OOoBasic

El algoritmo es sencillo: $\int_a^b f(x)\,dx \approx \frac{h}{2}\left(f(a)+f(b)+2\sum_{i=1}^{n-1} f(x_i)\right)$.

Algoritmo 1.2: Método del Trapecio.

Entrada: a, b, n y f continua en [a,b]

Resultado: Una aproximación s de $\int_a^b f(x)\,dx$

1 $h = (b-a)/n$;
2 $suma = 0$;
3 **for** $i = 1, 2, ..., n-1$ **do**
4 $\quad suma = suma + 2 \cdot f(a + i \cdot h)$
5 **return** $s = \frac{h}{2} \cdot \left(f(a) + f(b) + suma\right)$

Implementación OOoBasic.

Para la implementación usamos el cuaderno de la figura (1.14).

	A	B	C	D	E	F
1	Método del Trapecio					
2						
3	f(x) = sin(x)/x (está en el código)					
4						
5	a	b	n	Aproximación	Calcule	
6	1	2	100			
7						
8						
9						

Figura 1.14 Trapecio

En este caso vamos a usar la subrutina `Main`. En esta subrutina leemos a, b y n. Luego llamamos a la función `Trapecio(a,b,n)` e imprimimos el valor que nos entrega. Como antes, insertamos un módulo y ponemos el código ahí. Al botón se le debe asignar la subrutina `Main`, tal y como lo hicimos en la sección anterior.

```
REM  *****  BASIC  *****
 Sub Main
```

```
   Dim Hoja, Celda
   Dim a,b,n,i, suma,h
       Hoja    = thisComponent.Sheets(0)
       a       = Hoja.getCellRangeByName("A6").Value
       b       = Hoja.getCellRangeByName("B6").Value
       n       = Hoja.getCellRangeByName("C6").Value
       Celda   = Hoja.getCellRangeByName("D6")
   Celda.Value = Trapecio(a,b,n) 'Escribir en D6
End Sub

 function f(x)
   f= Sin(x)/x
end function

 function Trapecio(a,b,n)
   Dim i,suma,h
       suma=0
       h=(b-a)/n
       for i = 1 to n-1
           suma = suma+2*f(a+i*h)
       next i
       Trapecio = h/2*(f(a)+f(b)+suma)
end function
```

Bibliografía

[1] Andy Channelle. *Beginning OpenOffice 3. From Novice to Professional.* Apress. 2009.

[2] Laurent Godart, Bernard Marcelly. *Programmation OpenOffice.org 2. Macros OOoBasic et API.* Editions Eyrolles. 2006.

[3] Mark A. Bain. *Learn OpenOffice.org. Spreadsheet Macro Programming OOoBasic and Calc Automation.* Packt Publishing. 2006.