
NumInt Guia de Usuario

Versión 1.0.0

Escrito por Grupo 4:

Kendall Martínez C.

Daniel Ureña L.

Max Garro M.

Gabriel Vargas L.

Análisis Numérico para Ingeniería

Instituto Tecnológico de Costa Rica

Segundo Semestre 2023

Tabla de contenido

1.1	¿Qué es NumInt?.....	1
1.2	Instalación.....	1
1.3	Uso de las funciones.....	2
1.	Simpson	2
2.	Simpson Compuesto.....	2
3.	Simpson compuesto Iterativo	3
4.	Trapezio	3
5.	Trapezio Compuesto	4
6.	Trapezio Compuesto Iterativo	4
7.	Cuadratura Gaussiana:	4
8.	Cuadratura Gaussiana Compuesta	5
9.	Cuadratura Gaussiana Iterativa.....	6
10.	Romberg.....	6

1.1 ¿Qué es NumInt?

NumInt es el paquete que se ha desarrollado durante la tarea 3 del curso “Análisis numérico para ingeniería”, con el objetivo de recopilar los métodos desarrollados para las preguntas 1 y 2.

En la pregunta 1 se han desarrollado diferentes métodos para aproximar el valor de una integral definida, tales métodos son los siguientes: “Regla del Trapecio Compuesto”, “Regla de Simpson Compuesta”, “Cuadratura Gaussiana Compuesta”, “Regla del Trapecio Compuesta Iterativa”, “Regla de Simpson Compuesta Iterativa” y “Cuadratura Gaussiana iterativa”.

En la pregunta 2 se desarrolló el método de Romberg que fue estudiado del (inserte cita aquí), para ello se propuso un ejemplo en un documento aparte y luego se implementó computacionalmente en Octave.

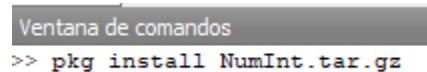
Entonces NumInt es un paquete que te permite acceder a dichos métodos de una manera más sencilla y compacta, en él se desarrollaron dos categorías, pregunta 1 y pregunta 2, en ellas esta cada una de las funciones propuestas anteriormente.

1.2 Instalación

Para instalar NumInt tienes que seguir los siguientes pasos:

- I. Inicia el programa GNU Octave
- II. En el apartado de ventana de comandos, abra la ubicación donde se encuentra descargado su archivo NumInt.tar.gz
- III. Seguidamente en la consola de comandos ingrese el siguiente comando

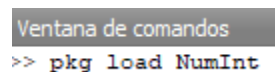
```
pkg install NumInt.tar.gz
```



```
Ventana de comandos  
>> pkg install NumInt.tar.gz
```

- IV. Seguidamente presiona la tecla “Enter” y espera que Octave instale el programa por sí mismo.
- V. Una vez el proceso haya terminado y no haya devuelto algún error procede a cargar el paquete con el siguiente comando (Si el comando no funciona pruebe con ponerlo todo en minúsculas):

```
pkg load NumInt
```



```
Ventana de comandos  
>> pkg load NumInt
```

- VI. Cuando presiones nuevamente la tecla “Enter” ya vas a poder utilizar el paquete como se explica a continuación en la guía.
- VII. Para quedar completamente seguros de la instalación puede ingresar el siguiente comando en la misma ventana:

pkg list

Ventana de comandos
>> pkg list

- VIII. Ahora en la lista que se muestra ubique el paquete llamado numint, se debería ver algo similar a la siguiente imagen:

```
numint | 0.0.1 | ... \GNU Octave\Octave-8.3.0\mingw64\share\octave\packages\numint-0.0.1
```

- IX. Los números al final son dependiendo de la versión instalada, así que pueden variar. Con esto la instalación está completada y puede continuar con el método de uso. (Recuerde utilizar pkg load numint, cada vez que desee acceder a los métodos del paquete en una nueva ventana de Octave)

1.3 Uso de las funciones

Aquí se detalla como utilizar cada una de las funciones que hay dentro del paquete NumInt.

1. Simpson

La regla de Simpson utiliza un polinomio evaluado en tres valores iniciales dados para calcular la integral de una función en un intervalo particular. Con la siguiente formula:

$$I_f = \int_a^b f(x)dx = \frac{(b-a)}{6} (f(x_0) + 4f(x_1) + f(x_2))$$

$$\text{Con: } x_0 = a, x_1 = a + \frac{a+b}{2}, x_2 = b$$

La función se llama como “simpson(f, a, b)” recibe 3 parámetros:

- f : string de la función a evaluar.
- a : inicio de intervalo de integración.
- b : final del intervalo de integración.

Ejemplo:

```
>> simpson("x+3",2,3)
ans = 5.5000
>>
```

2. Simpson Compuesto

Esta función implementa la función normal de manera que mediante la especificación de un tamaño de paso h, se subdividen los intervalos de integración en puntos distintos. Esto supone una mejor aproximación para el valor de la integral deseada.

La función se llama como “simpson_compuesto(f, a, b, n)” y recibe 4 parámetros:

- f : string de la función a evaluar.
- a : inicio de intervalo de integración.
- b : final del intervalo de integración.
- n : cantidad de puntos en que subdivide el intervalo de integración.

Ejemplo:

```
>> simpson_compuesto("x+3",2,4,30)
ans = 12.000
```

3. Simpson compuesto Iterativo

Esta función realiza un ciclo, el cual ejecuta un numero de veces definido la función de Simpson compuesta.

La función se llama como "simpson_compuesto_iterativa(f, a, b, tol, iterMax)" y tiene 5 parámetros:

- f : string de la función a evaluar (usar el formato f=@(x) función).
- a : inicio de intervalo de integración.
- b : final del intervalo de integración.
- tol : tolerancia de la sucesión.
- iterMax : cantidad máxima de iteraciones a realizar.

Ejemplo:

```
>> f=@(x) x+3
f =

@(x) x + 3

>> simpson_compuesto_iterativa(f,2,4,10^-10,100)
ans = 17
```

4. Trapecio

La regla del trapecio que se usa en el paquete se basa en la idea de que la integral es el área bajo la curva, entonces para una función f, en el intervalo [a , b] de integración se puede considerar el área bajo la curva como un trapecio.

$$A_T = \frac{Base + base}{2} \cdot h = \frac{f(a) + f(b)}{2} \cdot (b - a)$$

$$I = \int_a^b f(x)dx = A_T = \frac{f(a) + f(b)}{2} \cdot (b - a)$$

La función se llama como "trapecio(f, a, b)" y recibe 3 parámetros:

- f : función en texto.
- a : inicio del intervalo de integración.
- b : final del intervalo de integración.

Ejemplo:

```
>> trapecio("x+3",2,3)
ans = 5.5000
```

5. Trapecio Compuesto

Esta función lo que hace es llamar la función del trapecio, de manera en que se especifique un tamaño h en el que subdivide en n intervalos el rango para aproximar de mejor manera la integral.

La función se llama como “trapecio_compuesto(f, a, b, N)” recibe 4 parámetros:

- f : función en texto.
- a : inicio del intervalo de integración.
- b : final del intervalo de integración.
- N : cantidad de puntos en que se subdivide el intervalo de integración.

Ejemplo:

```
>> trapecio_compuesto("x+3",2,3,10)
ans = 5.5000
```

6. Trapecio Compuesto Iterativo

Esta función realiza un ciclo un número de veces especificado llamando la función del trapecio compuesto, así mismo calcula el error en la respuesta y hasta que no cumpla la tolerancia el valor no se considera como valido. Esto permite aproximar mejor la integral.

La función se llama como “trapecio_compuesto_iterativa(f, a, b, tol, iterMax)”

- f : función en texto (usar el formato f=@(x) función).
- a : inicio del intervalo de integración.
- b : final del intervalo de integración.
- tol : tolerancia del error de la función.
- iterMax : cantidad máxima de iteraciones por realizar.

Ejemplo:

```
>> f=@(x) x+3
f =

@(x) x + 3

>> trapecio_compuesto_iterativa(f,2,3,10^-10,100)
ans = 5.5000
```

7. Cuadratura Gaussiana:

La cuadratura gaussiana es método que te permite seleccionar sus puntos de evaluación utilizando algo conocido como polinomio de Legendre, el cual se define como:

$$\int_{-1}^1 f(x)dx \approx \omega_1 f(x_1) + \omega_2 f(x_2) + \dots + \omega_n f(x_n) \approx \sum_{i=1}^n \omega_i f(x_i)$$

Donde ω_i son los pesos obtenidos por medio del polinomio de Legendre y definidos:

$$\omega_i = \frac{2}{(1-x^2) [P'_n(x_i)]}$$

Para este paquete ya se han pre calculado dichos valores para utilizar la cuadratura gaussiana hasta orden 10. Así mismo se adaptaron para utilizar cualquier rango de valores que pertenezcan a los reales.

La función se llama como “cuad_gauss(f, a, b, n)” y tiene 4 parámetros:

- f : función en texto (usar el formato f=@(x) función).
- a : Límite inferior.
- b : Límite superior.
- n : el grado de la cuadratura (máximo 10).

Ejemplo:

```
>> f=@(x) x+3
f =

@(x) x + 3

>> cuad_gauss(f,2,3,3)
w =

    0.8889    0.5556    0.5556

ans = 5.5000
```

8. Cuadratura Gaussiana Compuesta

Este método aplica el mismo concepto de los anteriores de dividir el rango por n subintervalos, los cuales se calculan utilizando la función de cuadratura gaussiana original.

La función se llama como “gaussiana_compuesta(f, a, b, m, n)” y tiene 5 parámetros:

- f : función en texto (usar el formato f=@(x) función).
- a : Límite inferior.
- b : Límite superior.
- m : el grado de la cuadratura (máximo 10).
- n : es la cantidad de divisiones a utilizar.

Ejemplo:

```
>> f=@(x) x+3
f =

@(x) x + 3

>> gaussiana_compuesta(f,2,3,3,3)
w =

    0.8889    0.5556    0.5556

w =

    0.8889    0.5556    0.5556

w =

    0.8889    0.5556    0.5556

ans = 5.5000
```

9. Cuadratura Gaussiana Iterativa

Este método se encarga de obtener la aproximación de una integral definida f utilizando la cuadratura gaussiana compuesta, repitiendo el proceso una cantidad de veces indicada. Así mismo utiliza una tolerancia para verificar si el valor es lo suficientemente bueno como para devolverlo como resultado.

La función se llama como “gaussiana_compuesta_iterativa(f , a , b , M , tol , $iterMax$)” y tiene 6 parámetros:

- f : función en texto (usar el formato $f=@(x)$ función).
- a : Límite inferior.
- b : Límite superior.
- M : el grado de la cuadratura (máximo 10).
- tol : tolerancia del error de la función.
- $iterMax$: cantidad máxima de iteraciones por realizar.

Ejemplo:

```
>> f=@(x) x+3
f =
@(x) x + 3
>> gaussiana_compuesta_iterativa(f,2,3,3,10^-10,100)
w =
    0.8889    0.5556    0.5556
w =
    0.8889    0.5556    0.5556
w =
    0.8889    0.5556    0.5556
ans = 5.5000
```

10. Romberg

Este método fue desarrollado a partir de la extrapolación de Richardson aplicado a la regla compuesta del trapecio, con el propósito de mejorar su precisión. Este método se utiliza en casos en donde se requiera una gran precisión en la aproximación del valor de la integral.

La función implementada para este método es:

$$aprox = romberg(func, a, b, n)$$

Y se manejan las siguientes entradas:

a, b : Límites superior e inferior de la integral, respectivamente

$func$: función $f(x)$ a la cual se evalúa con la integral

n : Número de columnas y vectores que se evalúan en la matriz R

Es importante recalcar que la matriz R es utilizada como tabla para aproximar los siguientes valores en base de la iteración en base a los valores anteriores, entre mayor sea el valor de n ,

mayor será la precisión. Además, de salida tenemos el último valor de esta tabla, el cual es el valor que se encuentra en la posición $R_{(n,n)}$, este valor es representado de la siguiente manera:

aprox : Valor resultado del metodo de Romberg

Se procede a calcular la aproximación del valor de la integral definida:

$$\int_0^{\pi} \sin(x) dx$$

Con los valores iniciales de:

$$a = 0$$

$$b = \pi$$

$$func = \sin(x)$$

$$n = 6$$

Para ejecutar la función, se puede ejecutar de la siguiente forma:

```
>> f= @(x) sin(x);  
>> a=0;  
>> b=pi;  
>> n=6;  
  
>> aprox=romberg(f,a,b,n)  
aprox = 2.0000
```

Por lo tanto, el valor de la aproximación a la integral es de 2.