Instituto Tecnológico de Costa Rica
Área Académica Ingeniería en Computadores
CE-3201: Análisis Numérico para Ingeniería

Manual Funtras

Integrantes

Josué Araya García
Jonathan Guzmán Araya
Mariano Muñoz Masís
Daniel Prieto Sibaja

Cartago, Costa Rica
27 Marzo, 2021

Índice

1. Introducción

En las matemáticas existen diversos tipos de funciones como lo pueden ser:

- Algebraicas
- Trascendentes

Para este desarrollo nos enfocaremos en las funciones trascendentes, estas son las funciones que no satisfacen una ecuación polinomial cuyos coeficientes sean a su vez polinomios; esto contrasta con las funciones algebraicas, las cuales satisfacen dicha ecuación.

1.1. ¿Qué es Funtras?

Funtras es una biblioteca de funciones trascendentes desarrolladas en el lenguaje C^{++} con el objetivo de aproximar dichas funciones mediante el uso de métodos iterativos utilizando únicamente operaciones de suma, resta, multiplicación y potencia con una cantidad de iteraciones máximas de 2500 y una tolerancia de 10^{-8}.

2. Requisitos e Instalación

En esta sección se abarcarán los requisitos mínimos para su ejecución así como una breve guía de instalación de la misma.

2.1. Requisitos

El desarrollo y las pruebas de esta biblioteca se realizaron en el SO Windows 10, por lo tanto como requisitos se tiene:

- Sistema operativo Windows 10 o Linux
- MinGW
- Editor de código

2.2. Instalación

- Crear una carpeta donde se realizará un proyecto nuevo.
- Agregar a esta carpeta los archivos funtras.a y funtras.h.
- Crear un archivo .cpp donde se utilizará la biblioteca.
- En el archivo .cpp anterior aregar el header de la librería a utilizar, de la siguiente manera: include "funtras.h".
- Una vez realizado esto se puede utilizar la biblioteca para prograr alguna función deseada.
- Una vez esto se encuentre listo se deben ejecutar los siguientes comandos:
  - g++ -c archivo.cpp
  - g++ -o archivo archivo.o funtras.a
  - archivo
- Finalmente se mostrara en la terminal el resultado de la operacion utilizada en el archivo .cpp.

3. Funciones implementadas en Funtras

A continuación se detallan las funciones implementadas en la biblioteca funtras.

3.1. Inverso multiplicativo a^{-1}

Esta función calcula el inverso multiplicativo, recíproco o inverso de un número x real positivo, el uso de la misma se realiza de la siguiente manera:

varM1(a)

3.1.1. Formulación matemática

El cálculo se realiza mediante el método iterativo que se describe a continuación.

$$x_{k+1} = x_k(2 - a \cdot x_k) \tag{1}$$

3.1.2. Valores iniciales

El valor de x_0 esta dado por:

$$x_0 = \begin{cases} eps^{15} & si\ 80! < a \leq 100! \\ eps^{11} & si\ 60! < a \leq 80! \\ eps^{8} & si\ 40! < a \leq 60! \\ eps^{4} & si\ 20! < a \leq 40! \\ eps^{2} & si\ \ 0! < a \leq 20! \end{cases}$$

donde eps es una constante ya definida con valor de:

$$eps = 2{,}2204x10^{-16} \tag{2}$$

3.1.3. Condición de parada

La condición de parada de la iteración está dada por:

$$\left| \frac{x_{k+1} - x_k}{x_{k+1}} \right| \tag{3}$$

Cuando la tolerancia dada sea mayor que esta, entonces devuelve el resultado obtenido.

3.1.4. Ejemplos numérico

double expe = exp_t(11);
cout << " El resultado de e^{11} es: "<< expe;
El resultado de e^{11} es: 59874.1

3.2. Exponencial de Euler e^x

Esta función calcula el exponencial de e elevado a un número natural x, el uso de la misma se realiza de la siguiente manera:

exp_t(x)

3.2.1. Formulación matemática

El cálculo se realiza mediante la sumatoria que se describe a continuación.

$$S_k(a) = \sum_{n=0}^{k} \frac{a^n}{n!} \tag{4}$$

3.2.2. Condición de parada

La condición de parada de la iteración está dada por:

$$|S_{k+1}(a) - S_k(a)| < tol \tag{5}$$

3.2.3. Ejemplo numérico

double expoNUNO = varM1(144);
cout << " El resultado de 144^{-1} es: "<< expoNUNO;
El resultado de 144^{-1} es: 0.00694444

3.3. Seno $\sin(x)$

Esta función calcula el *seno* de un número x, el uso de la misma se realiza de la siguiente manera:

sin_t(x)

3.3.1. Formulación matemática

El cálculo se realiza mediante la sumatoria que se describe a continuación.

$$S_k(a) = \sum_{n=0}^{k} (-1)^n \frac{a^{2n+1}}{(2n+1)!} \tag{6}$$

3.3.2. Condición de parada

La condición de parada de la iteración está dada por:

$$|S_{k+1}(a) - S_k(a)| < tol \tag{7}$$

3.3.3. Ejemplo numérico

double seno = sin_t(2.33);
cout << " El resultado de $sen(2{,}33)$ es: "<< seno;
El resultado de $sen(2{,}33)$ es: 0.725384

3.4. Coseno $\cos(x)$

Esta función calcula el *coseno* de un número x, el uso de la misma se realiza de la siguiente manera:

cos_t(x)

3.4.1. Formulación matemática

El cálculo se realiza mediante la sumatoria que se describe a continuación.

$$S_k(a) = \sum_{n=0}^{k} (-1)^n \frac{a^{2n}}{(2n)!} \tag{8}$$

3.4.2. Condición de parada

La condición de parada de la iteración está dada por:

$$|S_{k+1}(a) - S_k(a)| < tol \tag{9}$$

3.4.3. Ejemplo numérico

double coseno = cos_t(3.78);
cout << " El resultado de $cos(3{,}78)$ es: "<< coseno;
El resultado de $cos(3{,}78)$ es: -0.803046

3.5. Tangente $\tan(x)$

Esta función calcula la *tangente* de un número x, el uso de la misma se realiza de la siguiente manera:

tan_t(x)

3.5.1. Formulación matemática

La función tangente se puede componer a partir de otras como lo son *seno* y *coseno*, es por ello que el calculo de la misma se realiza mediante la siguiente ecuación:

$$tan(x) = sen(x) \cdot cos(x)^{-1} \tag{10}$$

3.5.2. Ejemplo numérico

double tangente = tan_t(7);
cout << " El resultado de $tan(7)$ es: "<< tangente;
El resultado de $tan(7)$ es: 0.871449

3.6. Logaritmo natural $\ln(x)$

Esta función calcula el *logaritmonatural* de un número x, el uso de la misma se realiza de la siguiente manera:

ln_t(x)

3.6.1. Formulación matemática

El cálculo se realiza mediante la sumatoria que se describe a continuación.

$$S_k(a) = \frac{2(a-1)}{a+1} \sum_{n=0}^{k} \frac{1}{2n+1} \left(\frac{a-1}{a+1}\right)^{2n} \tag{11}$$

3.6.2. Condición de parada

La condición de parada de la iteración está dada por:

$$|S_{k+1}(a) - S_k(a)| < tol \tag{12}$$

3.6.3. Ejemplo numérico

double logan = ln_t(45);
cout << " El resultado de $ln(45)$ es: "<< logan;
El resultado de $ln(45)$ es:3.80666

3.7. Logaritmo $\log_a(x)$

Esta función calcula el *logaritmo* de base a a un número x, el uso de la misma se realiza de la siguiente manera:

log_t(x)

3.7.1. Formulación matemática

La función logaritmo se puede componer a partir de otra como $logaritmonatural$, es por ello que el calculo de la misma se realiza mediante la siguiente ecuación:

$$\log_a(x) = \ln(x) \cdot (\ln(a))^{-1} \tag{13}$$

3.7.2. Ejemplo numérico

double logb = log_t(33, 43);
cout << " El resultado de logaritmo en base 33 de 43 es: es: "<< logb; El resultado de logaritmo en base 33 de 43 es: 1.0757

3.8. Exponencial a^x

Esta función calcula el exponencial de un número a elevado a un número x, el uso de la misma se realiza de la siguiente manera:

power_t(x)

3.8.1. Formulación matemática

Esta función se compone de funciones ya programadas anteriormente, de esta forma podemos ver la siguiente relación:

$$a^x = e^{a \cdot \ln(x)} \tag{14}$$

Por lo tanto para esta función se aplican la función de logaritmo natural y exponencial de euler anteriormente explicadas.

3.8.2. Valores iniciales

Está función no tiene valores iniciales, solo los dependientes a las funciones que la componen.

3.8.3. Condición de parada

Esta función no tiene condición de parada como tal, solo las dependientes a las funciones que la componen.

3.8.4. Ejemplo numérico

double powert = power_t(2, 11);
cout << " El resultado de 2 elevado a la 11 es: "<< powert;
El resultado de 2 elevado a la 11 es: 2048

3.9. Seno hiperbólico $\sinh(x)$

Esta función calcula el *senohiperblico* de un número x, el uso de la misma se realiza de la siguiente manera:

sinh_t(x)

3.9.1. Formulación matemática

El cálculo se realiza mediante la sumatoria que se describe a continuación.

$$S_k(a) = \sum_{n=0}^{k} \frac{a^{2n+1}}{(2n+1)!} \tag{15}$$

3.9.2. Condición de parada

La condición de parada de la iteración está dada por:

$$|S_{k+1}(a) - S_k(a)| < tol \tag{16}$$

3.9.3. Ejemplo numérico

double senoh = sinh_t(2.1);
cout << " El resultado de $sinh(2{,}1)$ es: "<< senoh;
El resultado de $sinh(2{,}1)$ es: 4.02186

3.10. Coseno hiperbólico $\cosh(x)$

Esta función calcula el *cosenohiperblico* de un número x, el uso de la misma se realiza de la siguiente manera:

cosh_t(x)

3.10.1. Formulación matemática

El cálculo se realiza mediante la sumatoria que se describe a continuación.

$$S_k(a) = \sum_{n=0}^{k} \frac{a^{2n}}{(2n)!} \tag{17}$$

3.10.2. Condición de parada

La condición de parada de la iteración está dada por:

$$|S_{k+1}(a) - S_k(a)| < tol \tag{18}$$

3.10.3. Ejemplo numérico

double cosenoh = cosh_t(0.89);
cout << " El resultado de $cosh(0{,}89)$ es: "<< cosenoh;
El resultado de $cosh(0{,}89)$ es: 1.42289

3.11. Tangente hiperbólico $\tanh(x)$

Esta función calcula la *tangentehiperblica* de un número x, el uso de la misma se realiza de la siguiente manera:

tanh_t(x)

3.11.1. Formulación matemática

La función tangente hiperbólico se puede componer a partir de otras como lo son *senohiperblico* y *cosenohiperblico*, es por ello que el calculo de la misma se realiza mediante la siguiente ecuación:

$$tanh(x) = senh(x) \cdot cosh(x)^{-1} \tag{19}$$

3.11.2. Ejemplo numérico

double tanoh = tanh_t(0.22);
cout << " El resultado de $tanh(0{,}22)$ es: "<< tanoh;
El resultado de $tanh(0{,}22)$ es: 0.216518

3.12. Raíz cuadrada $\sqrt{x}$

Esta función calcula la *razcuadrada* de un número x mediante el método de Newton-Raphson, el uso de la misma se realiza de la siguiente manera:

sqrt_t(x)

3.12.1. Formulación matemática

El método de Newton-Raphson se define como:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},\ x_0 = \alpha \tag{20}$$

Donde para encontrar la p-ésima raíz de a:

$$\sqrt[p]{a} \tag{21}$$

El cero de la función está dado por:

$$g(x) = x^p - a,\ x_0 = \frac{a}{2} \tag{22}$$

Por lo que resulta en la siguiente iteración:

$$\begin{cases} x_0 = \frac{a}{2} \\ x_{k+1} = x_k - \frac{x_k^2 - a}{2 \cdot x_k} \end{cases}$$

3.12.2. Valore inicial

El valor inicial está dado por: $x_{0=\frac{a}{2}}$.

3.12.3. Condición de parada

$$|(S_{k+1}(a) - S_k(a))/S_{k+1}(a)| < tol \tag{23}$$

3.12.4. Ejemplo numérico

double raiz = sqrt_t(144);
cout << " El resultado de la raíz cuadrada de 144 es: "<< raiz;
El resultado de la raíz cuadrada de 144 es:12

3.13. Raíz $\sqrt[a]{x}$

Esta función calcula la $raza - sima$ de un número x mediante el método de Newton-Raphson, el uso de la misma se realiza de la siguiente manera:

root_t(x)

3.13.1. Formulación matemática

El método de Newton-Raphson se define como:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},\ x_0 = \alpha \tag{24}$$

Donde para encontrar la p-ésima raíz de a:

$$\sqrt[p]{a} \tag{25}$$

El cero de la función está dado por:

$$g(x) = x^p - a,\ x_0 = \frac{a}{2} \tag{26}$$

Por lo que resulta en la siguiente iteración:

$$\begin{cases} x_0 = \frac{a}{2} \\ x_{k+1} = x_k - \frac{x_k^p - a}{2 \cdot x_k} \end{cases}$$

3.13.2. Valores iniciales

El valor inicial está dado por: $x_{0=\frac{a}{2}}$.

3.13.3. Condición de parada

$$|(S_{k+1}(a) - S_k(a))/S_{k+1}(a)| < tol \tag{27}$$

3.13.4. Ejemplo numérico

double raizx = root_t(45, 7);
cout << " El resultado de la raíz 7 de 45 es: "<< raizx;
El resultado de la raíz 7 de 45 es:1.72256

3.14. Arseno $\sin^{-1}(x)$

Esta función calcula el *arcoseno* de un número x, el uso de la misma se realiza de la siguiente manera:

asin_t(x)

3.14.1. Formulación matemática

El cálculo se realiza mediante la sumatoria que se describe a continuación.

$$S_k(a) = \sum_{n=0}^{k} \frac{2n!}{4^n (n!)^2 (2n+1)} a^{2n+1} \tag{28}$$

3.14.2. Condición de parada

La condición de parada de la iteración está dada por:

$$|S_{k+1}(a) - S_k(a)| < tol \tag{29}$$

3.14.3. Ejemplo numérico

double aseno = asin_t(0.33);
cout << " El resultado de aseno(0.33) es: "<< aseno;
El resultado de aseno(0.33) es: 0.336304

3.15. Arcotangente $\tan^{-1}(x)$

Esta función calcula el *arcotangente* de un número x, el uso de la misma se realiza de la siguiente manera:

atan_t(x)

3.15.1. Formulación matemática

El cálculo se realiza mediante la sumatoria que se describe a continuación.

$$S_k(a) = \sum_{n=0}^{k} (-1)^n \frac{a^{2n+1}}{2n+1} \tag{30}$$

3.15.2. Condición de parada

La condición de parada de la iteración está dada por:

$$|S_{k+1}(a) - S_k(a)| < tol \tag{31}$$

3.15.3. Ejemplo numérico

double atan = atan_t(0.67);
cout << " El resultado de atan(0.67) es: "<< atan;
El resultado de atan(0.67) es: 0.590307