FunTras

Paquete computacional en GNU Octave

Versión 1.0.0

Instituto Tecnológico de Costa Rica

Análisis Numérico para Ingeniería

Grupo 5

José Julián Camacho Hernández
Juan Pablo Carrillo Salazar
José Leonardo Guillén Fernández
Fabián Ramírez Arrieta

2 de enero de 2024

Índice

1. ¿En qué consiste FunTras?

FunTras es un paquete computacional que se centra en un tipo de funciones en específico, siendo estas las funciones trascendentes, las cuales son funciones que no satisfacen una ecuación polinomial. Este paquete permite aproximar el valor numérico de un conjunto de funciones trascendentes de variable real.

2. Instalación

Primeramente se debe descargar en archivo comprimido el cual se encuentra en la siguiente dirección:

FunTras

Luego, una vez con en paquete instalado en su computadora. Debe utilizar la consola de GNU Octave y ubicarse en el directorio donde guardó el archivo comprimido anteriormente descargado por medio del comando cd. Una vez allí, se debe correr el siguiente comando para instalar el paquete en nuestro Octave:

```
pkg install FunTras-1.0.0.tar.gz
```

Luego de esto, es necesario cargar el paquete con el comando para poder utilizar todas las funciones disponibles en el paquete FunTras:

```
pkg load funtras
```

Para su uso, además es necesario tener instalado el paquete Symbolic, el cual puede ser instalada de la misma forma que el anterior paquete. Debe descargarse el archivo comprimido de la siguiente dirección:

Symbolic

Luego de igual manera se debe utilizar los comandos:

```
pkg install symbolic-2.9.0.tar.gz
pkg load symbolic
```

En caso de querer desinstalar el paquete Funtras. Es necesario ejecutar el siguiente comando desde la consola de GNU Octave:

```
pkg uninstall funtras
```

3. Funciones implementadas

3.1. Inverso multiplicativo

Esta función se puede aproximar utilizando la iteración:

$$x_{k+1} = x_k(2 - xx_k)$$

donde x_0 es el valor inicial dado por:

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

Comando GNU Octave	`div_t(x)`
Entradas	x: número real diferente de cero
Salidas	y: número real diferente de cero

Ejemplo ilustrativo:

```
>> y = div_t(4)
y = 0.2500
>> y = div_t(-100)
y = -1.0000e-02
>> y = div_t(1/250)
y = 250.00
>> y = div_t(0)
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>>
```

3.2. Exponencial

Función	e^x
Dominio	$\mathbb{R}$
Rango	$y \in \mathbb{R} : y>0$

Esta función se puede aproximar utilizando el polinomio:

$$S_k(x) = \sum_{n=0}^{k} \frac{x^n}{n!}$$

Comando GNU Octave	`exp_t(x)`
Entradas	x: número real
Salidas	y: número real mayor que cero

Ejemplo ilustrativo:

```
>> y = exp_t(0)
y = 0.999999999999987
>> y = exp_t(1)
y = 2.718281826189731
>> y = exp_t(-1)
y = 0.367879439238574
>> y = exp_t(10)
y = 22026.46023005593
>>
```

3.3. Seno

Esta función se puede aproximar utilizando el polinomio:

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

Comando GNU Octave	`sin_t(x)`
Entradas	x: número real
Salidas	y: número real entre -1 y 1 ambos incluidos

Ejemplo ilustrativo:

```
>> y = sin_t(0)
y = 0
>> y = sin_t(-pi/2)
y = -1.0000
>> y = sin_t(-pi)
y = -1.3744e-09
>> y = sin_t(pi/4)
y = 0.7071
>>
```

3.4. Coseno

Esta función se puede aproximar utilizando el polinomio:

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

Comando GNU Octave	`cos_t(x)`
Entradas	x: número real
Salidas	y: número real entre -1 y 1 ambos incluidos

Ejemplo ilustrativo:

```
>> y = cos_t(0)
y = 1.0000
>> y = cos_t(-pi/2)
y = 6.3128e-09
>> y = cos_t(-pi)
y = -1.0000
>> y = cos_t(pi/4)
y = 0.7071
>>
```

3.5. Tangente

Esta función se puede calcular mediante la igualdad:

$$\tan(x) = \frac{\sin(x)}{\cos(x)}$$

Comando GNU Octave	`tan_t(x)`
Entradas	x: número real diferente de $\frac{k\pi}{2}$ con k entero impar
Salidas	y: número real
Restricciones	cos(x) diferente de cero

Ejemplo ilustrativo:

```
>> y = tan_t(0)
y = 0
>> y = tan_t(-pi)
y = 1.3744e-09
>> y = tan_t(pi/2)
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>> y = tan_t(pi/4)
y = 1.0000
>> y = tan_t(3*pi/2)
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>>
```

3.6. Logaritmo natural

Esta función se puede aproximar utilizando el polinomio:

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

Comando GNU Octave	`ln_t(x)`
Entradas	x: número real positivo
Salidas	y: número real

Ejemplo ilustrativo:

```
>> y = ln_t(1)
y = 0
>> y = ln_t(e)
y = 1.0000
>> y = ln_t(-10)
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>> y = ln_t(0)
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>>
```

3.7. Logaritmo en base *a* de *x*

Esta función se puede calcular mediante la igualdad:

$$\log_a(x) = \frac{\ln(x)}{\ln(a)}$$

Comando GNU Octave	`log_t(x,a)`
Entradas	x, a: número real tal que: a!=0 y a!=1 y a = x, o x>0 y 0<a<1, o x>0 y a>1
Salidas	y: número real
Restricciones	ln(a) diferente de cero

Ejemplo ilustrativo:

```
>> y = log_t(1,10)
y = 0
>> y = log_t(e,e)
y = 1.0000
>> y = log_t(10,pi/8)
y = -2.4634
>> y = log_t(5,-5)
Error: la(s) entrada(s) no está en el dominio de la función
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>> y = log_t(-8,3)
Error: la(s) entrada(s) no está en el dominio de la función
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>>
```

3.8. Potencia en base *a* de *x*

Esta función se puede aproximar utilizando el polinomio:

$$S_k(x) = \sum_{n=0}^{k} \frac{(x-1)^n a \ln^n x}{n!}$$

Comando GNU Octave	`power_t(x,a)`
Entradas	x, a: número real tal que a!=0 y x es entero, o x>=1 y x es entero, o a>=0 y x>0, o a>0
Salidas	y: número real

Ejemplo ilustrativo:

```
>> y = power_t(0,2)
Skpl = 2.0000
Skpl = 0.6137
Skpl = 1.0942
Skpl = 0.9832
Skpl = 1.0024
Skpl = 0.9997
Skpl = 1.0000
Skpl = 1.0000
Skpl = 1.0000
Skpl = 1.0000
Skpl = 1.0000
Skpl = 1.0000
y = 1.0000
>>  y = power_t(2,0)
Error: la(s) entrada(s) no está en el dominio de la función
```

3.9. Seno hiperbólico

Esta función se puede aproximar utilizando el polinomio:

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

Comando GNU Octave	`sinh_t(x)`
Entradas	x: número real
Salidas	y: número real

Ejemplo ilustrativo:

```
>> y = sinh_t(0)
y = 0
>> y = sinh_t(pi)
y = 11.549
>> y = sinh_t(-pi/3)
y = -1.2494
>> y = sinh_t(-e)
y = -7.5441
>>
```

3.10. Coseno hiperbólico

Esta función se puede aproximar utilizando el polinomio:

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

Comando GNU Octave	`cosh_t(x)`
Entradas	x: número real
Salidas	y: número real mayor o igual a 1

Ejemplo ilustrativo:

```
>> y = cosh_t(0)
y = 1.0000
>> y = cosh_t(9)
y = 4051.5
>> y = cosh_t(-9)
y = 4051.5
>> y = cosh_t(pi/5)
y = 1.2040
>>
```

3.11. Tangente hiperbólica

Esta función se puede calcular mediante la igualdad:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)}$$

Comando GNU Octave	`tanh_t(x)`
Entradas	x: número real
Salidas	y: número real entre -1 y 1

Ejemplo ilustrativo:

```
>> y = tanh_t(0)
y = 0
>> y = tanh_t(1)
y = 0.7616
>> y = tanh_t(2*pi)
y = 1.0000
>> y = tanh_t(-2*pi)
y = -1.0000
>>
```

3.12. Raíz cuadrada

Esta función se puede aproximar utilizando la iteración:

$$\begin{cases} x_{k+1} = \frac{x_k + \frac{x}{x_k}}{2} \\ x_0 > 0 \end{cases}$$

Comando GNU Octave	`sqrt_t(x)`
Entradas	x: número real mayor o igual a cero
Salidas	y: número real mayor o igual a cero

Ejemplo ilustrativo:

```
>> y = sqrt_t(100)
y = 9.999999989835223
>> y = sqrt_t(pi^2)
y = 3.141592657860183
>> y = sqrt_t(-5)
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>> y = sqrt_t(0)
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>>
```

3.13. Raíz enésima

Esta función se puede aproximar utilizando la iteración generada por el método de Newton-Rapshon al encontrar el cero positivo de la función $g(z) = z^a - x$.

Esta genera la sucesión

$$\begin{cases} x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)} \\ x_0 = \frac{a}{2} \end{cases}$$

Comando GNU Octave	`root_t(x,a)`
Entradas	x: número real positivo
Salidas	y: número real

Ejemplo ilustrativo:

```
>> y = root_t(2401,4)
y = 7.0000
>> y = root_t(150,3)
y = 5.3133
>> y = root_t(2187,7)
y = 3.0000
>> y = root_t(516,4)
y = 4.7661
```

3.14. Seno inverso

Esta función se puede aproximar utilizando el polinomio:

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

Comando GNU Octave	`asin_t(x)`
Entradas	x: número real entre -1 y 1 ambos incluidos
Salidas	y: número real entre $\frac{-\pi}{2}$ y $\frac{\pi}{2}$ ambos incluidos

Ejemplo ilustrativo:

```
>> y = asin_t(0)
y = 0
>> y = asin_t(1/2)
y = 0.523598770472674
>> y = asin_t(-pi/4)
y = -0.903328443044087
>> y = asin_t(10)
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>>
```

3.15. Tangente inversa

Esta función se puede aproximar utilizando el polinomio:

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

Comando GNU Octave	`atan_t(x)`
Entradas	x: número real entre -1 y 1 ambos incluidos
Salidas	y: número real entre $\frac{-\pi}{2}$ y $\frac{\pi}{2}$ ambos excluidos

Ejemplo ilustrativo:

```
>> y = atan_t(0)
y = 0
>> y = atan_t(1/e)
y = 0.352513419602951
>> y = atan_t(-0.95)
y = -0.759762749832662
>> y = atan_t(-5)
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>>
```

3.16. Coseno inverso

Esta función se puede calcular mediante la igualdad:

$$\cos^{-1}(x) = \frac{\pi}{2} - \sin^{-1}(x)$$

Comando GNU Octave	`acos_t(x)`
Entradas	x: número real entre -1 y 1 ambos incluidos
Salidas	y: número real entre 0 y π ambos incluidos

Ejemplo ilustrativo:

```
>> y = acos_t(0)
y = 1.570796327339657
>> y = acos_t(-1/2)
y = 2.094395097812331
>> y = acos_t(1/2)
y = 1.047197556866984
>> y = acos_t(10)
Error: la(s) entrada(s) no está en el dominio de la función
y = NaN
>>
```

3.17. Aproximación de π

Este número se puede aproximar utilizando la iteración:

$$\begin{cases} x_{k+1} = x_k - \tan^{-1}(x_k) \\ x_0 = 3 \end{cases}$$

Comando GNU Octave	`pi_t()`

Ejemplo ilustrativo:

```
>> y = pi_t()
y = 3.141592654679314
>>
```

3.18. Función de prueba TestFunTras

En esta función se realiza un cálculo de un valor utilizando las funciones implementadas:

$$\frac{\sqrt[3]{\sin(\frac{3}{7}) + \ln(2)}}{\sinh(\sqrt{2})} + \tan^{-1}(e^{-1}) \approx 0{,}887379$$

Comando GNU Octave	`test_funtras()`

Ejemplo ilustrativo:

```
================================
             Prueba
================================
Valor esperado: 0.887379
Valor esperado: 0.887379
Error: 4.68018e-09
================================
>>
```