TEC | Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores
Análisis Numérico para Ingeniería – CE-3102

Tarea #1

Manual de usuario para el paquete FunTras

Estudiantes:

Yordi Brenes Roda
Gabriel Antonio Conejo Valerio
Ricardo Gatgens Rodríguez
Ignacio Morales Chang

Profesor:
Juan Pablo Soto Quirós

IS - 2022

Tabla de contenido

Explicación

El paquete FunTras nos permite aproximar mediante los operadores básicos de suma, resta y multiplicación; funciones trascendentales como el seno, tangente, logaritmo, exponenciación, potencia. A partir del uso de métodos iterativos se desarrolla la lista de funciones dadas en la especificación.

Este paquete computacional está desarrollado en GNU Octave e incluye las siguientes funciones

- x^{-1}
- sin(x)
- tan(x)
- $\log_a(x)$
- sinh(x)
- tanh(x)
- $\sqrt[a]{x}$
- $\tan^{-1}(x)$
- e^x
- cos(x)
- ln(x)
- a^x
- cosh(x)
- $\sqrt{x}$
- $\sin^{-1}(x)$
- $\cos^{-1}(x)$

Instalación

Primero debe cerciorarse tener instalado el software GNU octave

https://www.gnu.org/software/octave/download

Para instalar el paquete se ubica este en el directorio donde se encuentra la carpeta del usuario dependiendo del sistema operativo y se ejecuta el comando

```
pkg install funTras.tar.gz
```

Para cargar el paquete se ejecuta el comando

```
pkg load image
```

Conceptos Básicos

Funciones básicas

Inverso multiplicativo

Esta función da el resultado del inverso de un número “x” usando el eps (cero de máquina) elevado a un número específico dependiendo del valor x se aproxima utilizando un proceso iterativo.

```
>> y=div_t(2)
y = 0.5000
>> |
```

$$div\_t(x) = x^{-1}$$

Potenciación de Euler

A partir de método iterativo donde se evalúa *x* potenciando con el número de iteración actual y multiplicando por el inverso multiplicativo del factorial de la iteración actual, se obtiene el resultado de multiplicar euler “x”cantidad de veces.

```
>> e=exp_t(2)
e = 7.389056098516415
>> e=exp_t(8)
e = 2980.957987040617
>> |
```

$$exp\_t(x) = e^{x}$$

Logaritmo natural

Mediante un serie de números representado con un ciclo de obtiene el valor del logaritmo natural de x

```
>> ln_t(2)
Salida por tolerancia
ans = 0.693147180559945
>> ln_t(exp_t(1))
Salida por tolerancia
ans = 0.999999999936402
>>
```

$$ln\_t(x) = \ln(x)$$

Potenciación

A partir de la función $e^{x*\ln a}$ se obtiene el resultado de potenciar “a” la cantidad “x” veces

```
>> p=power_t(8,2)
Salida por tolerancia
p = 256.00
>> p=power_t(3,2)
Salida por tolerancia
p = 8.0000
>> p=power_t(10,5)
Salida por tolerancia
p = 9.7656e+06
>>
```

$$power\_t(x,a) = a^x$$

Logaritmo x en base a

Usando las propiedades de lo logaritmos el cual para obtener logaritmo de un base “n” en específico se puede descomponer en $\ln x \cdot \frac{1}{\ln a}$ y con la ayuda de la funciones ya implementadas de ln_t y div_t. Este puede dar un valor muy cercano debido a nivel de tolerancia

```
>> l=log_t(128,2)
Salida por tolerancia
Salida por tolerancia
l = 6.999999999999924
>> l=log_t(1000,10)
Salida por tolerancia
l = 2.999998202528801
>> l=log_t(1,2)
Salida por tolerancia
Salida por tolerancia
l = 0
>>
```

$$log\_t(x,a) = log_a(x)$$

Raíz a-ésima de x

Esta se basa en el método de newton-raphson tomando como valor inicial $x_0 = \frac{a}{2}$ como método de parada $|(x_{k+1} - x_k)| < tol\,|(x_{k+1})|$ donde “tol” ya es un valor fijado.

```
>> r=root_t(256,2)
r = 16
>> r=root_t(9,3)
r = 2.080083823051904
>> r=root_t(3,2)
r = 1.732050807568877
>>
```

$$root\_t(x,a) = \sqrt[a]{x}$$

Raíz cuadrada

Este usa la función potencia pero el exponente es 1 ⁄ 2 el cual usa el inverso multiplicativo de 2. Puede dar valores muy cercanos debido al número fijo de tolerancia

```
>> s=sqrt_t(3)
Salida por tolerancia
s = 1.732050807532812
>> s=sqrt_t(1024)
s = 31.99991443176868
>>
```

$$sqrt\_t(x) = \sqrt{x}$$

Funciones Trigonométricas

Seno de x

Esta función usa la computarización de la serie de senos hecha a base de la serie de Taylor. La cual consiste en una sumatoria de donde el valor de entrada es elevando a un factor “2n+1” y dividido por el factorial del factor; “n” es el número de iteración actual , además dependiendo de este “n” el número obtenido en la iteración por su negativo.

```
>> y=sin_t(0)
y = 0
>> y=sin_t(pi)
y = 1.0348e-11
>> y=sin_t(pi*div_t(2))
y = 1.0000
>> y=sin_t(pi*div_t(4))
y = 0.7071
>> y=sin_t((3*pi)*div_t(2))
y = -1.0000
>>
```

$$sin\_t(x) = \sin x$$

Coseno de x

Esta función usa la computarización de la serie de cosenos hecha a base de la serie de Taylor. La cual consiste en una sumatoria de donde el valor de entrada es elevando a un factor “2n” y dividido por el factorial del factor; “n” es el número de iteración actual , además dependiendo de este “n” el número obtenido en la iteración por su negativo.

```
>> y=cos_t(0)
y = 1
>> y=cos_t(pi*div_t(2))
y = -6.5134e-11
>> y=cos_t(pi*div_t(4))
y = 0.7071
>> y=cos_t(pi)
y = -1.0000
>> y=cos_t((3*pi)*div_t(2))
y = -2.2681e-11
>>
```

$$\cos\_t(x) = \cos x$$

Tangente de x

Esta función usa la computarización de la mezcla a razón de la propiedad de la tangente usando la serie de cosenos y la serie de senos hecha a patir de la serie de Taylor como base. La cual consiste en una sumatoria de donde el valor de entrada es elevando a un factor “2n+1” y dividido por el factor; “n” es el número de iteración actual , además dependiendo de este “n” el número obtenido en la iteración por su negativo.

```
>> t=tan_t(pi_t()*div_t(6))
t = 0.577261383749336
>> t=tan_t(pi_t()*div_t(4))
t = 0.999800019995003
>> t=tan_t(pi_t()*div_t(3))
t = 1.7315175973210 98
>>
```

$$\tan\_t(x) = \tan x$$

Seno hiperbólico de x

Esta función usa la computarización de la serie de senos hecha a base de la serie de taylor. La cual consiste en una sumatoria de donde el valor de entrada es elevado a un factor “2n+1” y dividido por el factorial del factor; “n” es el número de iteración actual , este no incluye la variante de positivo o negativo dependiendo del “n”.

```
>> s=sinh_t(1)
s = 1.175201193643034
>> s=sinh_t(2)
s = 3.626860407842667
>>
```

$$sinh\_t(x) = \sinh x$$

Tangente hiperbólico de x

Esta función usa la computarización de la mezcla a razón de la propiedad de la tangente donde se puede definir como senh(x) multiplicado por el inverso multiplicativo de cosh(x)

```
>> s=tanh_t(1)
s = 0.761594155960953
>> s=tanh_t(2)
s = 0.964027580085263
>> s=tanh_t(pi_t())
s = 0.996269098260027
>>
```

$$\tanh\_t(x) = \tanh x$$

Coseno hiperbólico de x

Esta función usa la computarización de la serie de cosenos hecha a base de la serie de Taylor. La cual consiste en una sumatoria de donde el valor de entrada es elevando a un factor “2n” y dividido por el factorial del factor; “n” es el número de iteración actual , este no incluye la variante de positivo o negativo dependiendo del “n”

```
>> c=cosh_t(1)
c = 1.543080634803725
>> c=cosh_t(pi_t())
c = 11.58733470711822
>>
```

$$\cosh\_t(x) = \cosh x$$

Seno inverso de x

Esta función usa el polinomio $\frac{(2n)!}{4^n \cdot (n!)^2 \cdot (2n+1)} \cdot x^{2n+1}$ dentro de un proceso iterativo donde “n” es el valor de la iteración actual y su tolerancia está dada por $|S_{k+1} - S_k| < tol$ donde S es el valor final de la función.

```
>> as=asin_t(0.43)
as = 0.444492775674573
>> as=asin_t(0.5)
as = 0.523598774479260
>>
```

$$asin\_t(x) = \sin^{-1} x$$

Tangente inversa de x

Esta función usa el polinomio $(-1)^n \cdot \frac{a^{2n+1}}{2n+1}$ dentro de un proceso iterativo donde "n" es el valor de la iteración actual y su tolerancia está dada por $|S_{k+1} - S_k| < tol$ donde S es el valor final de la función.

```
>> at=atan_t(1)
at = 0.785298163401448
>> at=atan_t(-1)
at = -0.785298163401448
>> at=atan_t(0)
at = 0
>>
```

$$\text{atan\_t}(x) = \tan^{-1} x$$

Coseno inverso de x

Utilizando la propiedad de este coseno es equivalente a pi entre 2 sumado al negativo del valor de seno inverso de x.

```
>> c=cosh_t(0.43)
c = 1.0939
>> c=cosh_t(0.5)
c = 1.1276
>> c=cosh_t(1)
c = 1.5431
>> c=cosh_t(0)
c = 1
>>
```

$$\text{acos\_t}(x) = \cos^{-1} x$$

Constantes

Pi

Para obtener esta constante se utiliza la serie de Leibniz con la modificación de un 4 para obtener pi y no pi por 1/4.

```
>> pi_t()
ans = 3.141192653605793
>>
```

$$\text{pi\_t}() \approx \pi$$