
Manual: Fun Tras

v 1.1.0

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
CE-3102 Análisis Numérico para Ingeniería

Escrito por:
Delgado, Fiorella
Marín, Cristian
Martínez, Randy
Rivera, Karla

18 de Octubre, 2020

Tabla de Contenidos

Resumen	2
FunTrans	3
Utilización	5
Requisitos	5
Instalación	7
Uso	11
Funciones	12
Aritmética	13
Constantes	16
Expresiones	19
Trigonometría	25
Ejemplo	39

1. Resumen

Este documento corresponde a una guía de instalación y uso del paquete computacional llamado: FunsTrans. El cual consta de funciones que aplican series de sumas sucesivas para aproximar valores matemáticos de funciones trascendentales.

Asimismo, se explica el trasfondo de cómo están implementadas tales funciones y ejemplos de resultados de las mismas a la hora de ser ejecutadas.

2. FunTrans

FunTrans es un paquete computacional escrito en GNU Octave v5.2.0. El mismo representa la realización de una tarea programada para el curso CE3102, Análisis Numérico para Ingeniería, que forma parte del plan de estudio de la carrera Ingeniería en Computadores del Instituto Tecnológico de Costa Rica.

FunTrans consiste en la implementación de funciones matemáticas trascendentales. Una función trascendental es una función que no satisface una ecuación polinomial, es decir: no es una ecuación algebraica. Una característica de una función trascendente es que se puede expresar como una serie sucesiva de sumas.

Siguiendo la premisa, anterior, el paquete computacional está construido como conjunto de funciones, archivos *.m*, que aplican Series de Taylor, Macluarin y Leibniz para aproximar valores constantes y funciones trascendentales matemáticas.

En la Tabla 1 se describen las funciones matemáticas que se programaron. En la primera columna el nombre de la función; en la segunda columna notación matemática; y en la tercera columna la notación dentro de FunTras.

En la sección [4. Funciones](#) se encuentran todas las funciones anteriormente complemente desglosadas. Es decir, la fórmula matemática que se está aplicando, cuáles son los parámetros válidos de entrada y restricciones presentes.

Función	Notación Matemática	Notación FunTrans
Inverso	x^{-1}	div_t (x)
Exponente e	e^{-1}	exp_t (x)
Constante		pi_t ()
Logaritmo	$\log_a(x)$	log_t (x, a)
Logaritmo Natural	$\ln(x)$	ln_t (x)
Raíz Cuadrada	\sqrt{x}	sqrt_t (x)
Raíz	$\sqrt[a]{x}$	root_t (x, a)
Exponente	x^{-a}	power_t (x, a)
Seno	$\text{sen}(x)$	sin_t (x)
Coseno	$\text{cos}(x)$	cos_t (x)
Tangente	$\text{tan}(x)$	tan_t (x)
Arcoseno	$\text{sen}^{-1}(x)$	asin_t (x)
Arcocoseno	$\text{cos}^{-1}(x)$	acos_t (x)
Arcotangente	$\text{tan}^{-1}(x)$	atan_t (x)
Seno Hiperbólico	$\text{senh}(x)$	sinh_t (x)
Coseno Hiperbólico	$\text{cosh}(x)$	cosh_t (x)
Tangente Hiperbólico	$\text{tanh}(x)$	tanh_t (x)

Tabla 1. Funciones en notación matemática y en FunTrans.

3. Utilización

Para uso del paquete computacional FunTrans se necesitan algunos pasos previos, los cuales se describen a continuación.

3.1. Requisitos

3.1.1. GNU OCTave:

GNU Octave es un lenguaje de alto nivel con un enfoque de aplicaciones matemáticas. Es un *software* gratuito bajo la licencia *GNU General Public License (GPL)*, lo que permite su uso, reproducción y modificación de manera libre. Para mayor información visitar el sitio web: <https://www.gnu.org/licenses/gpl-3.0.html>

Su instalación varía según cada sistema operativo. Por lo que se recomienda visitar según sea el caso respectivo.

- Windows:

<https://www.gnu.org/software/octave/download#ms-windows>

- Mac OS:

https://wiki.octave.org/Octave_for_macOS

- *Distros* Linux:

https://wiki.octave.org/Octave_for_GNU/Linux

3.1.2. Symbolic

Con el programa de GNU Octave (sea CLI o GUI) instalado en la computadora, hay una dependencia de un paquete llamado *Symbolic*. Este es un paquete para trabajar con valores simbólicos de matemática dentro del lenguaje.

Este paquete se requiere porque algunas de las funciones implementadas necesitan un manejo simbólico. Su instalación varía según cada sistema operativo. Por lo que se recomienda visitar según sea el caso respectivo.

- Windows:

<https://github.com/cbm755/octsympy/wiki/Notes-on-Windows-installation>

- Mac OS:

<https://github.com/cbm755/octsympy>

- *Distros* Linux:

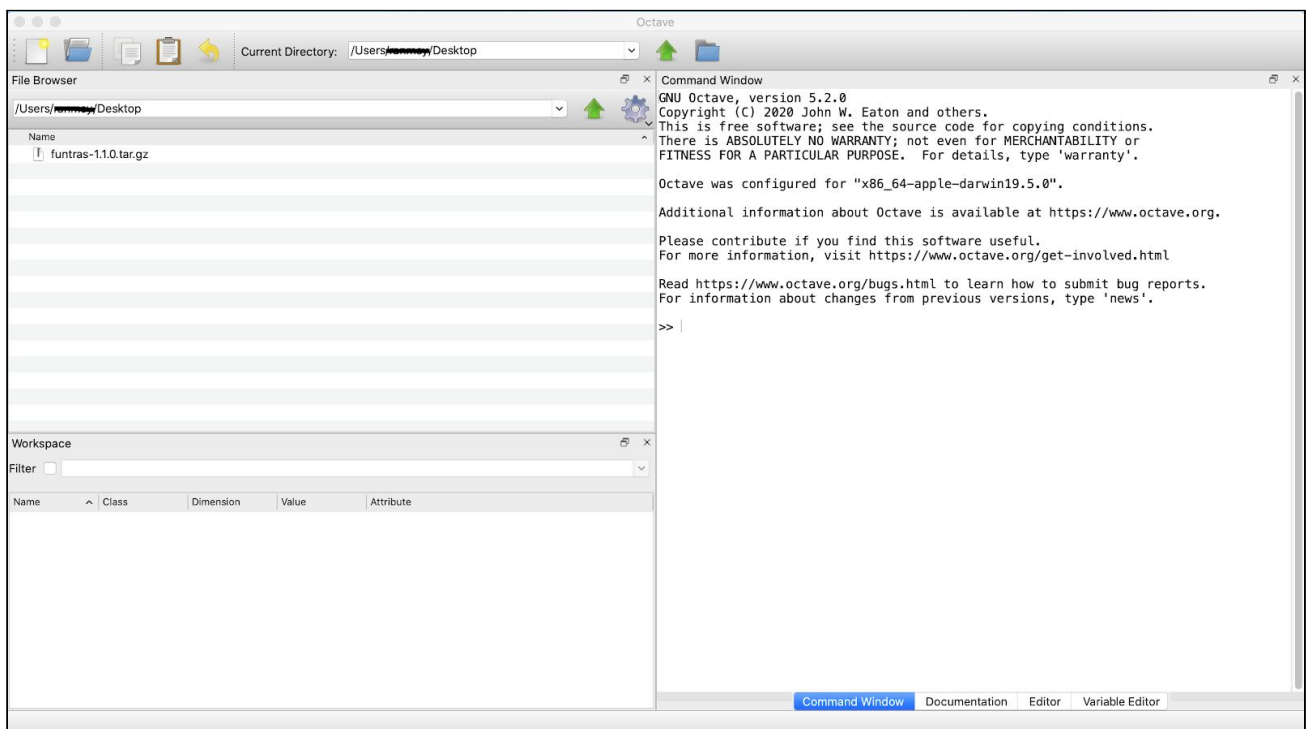
<https://github.com/cbm755/octsympy>

3.3. Instalación

Para poder utilizar el paquete computacional de FunTrans, este mismo debe instalarse dentro del computador donde se va a aplicar. Para esto se recomienda seguir los siguientes pasos:

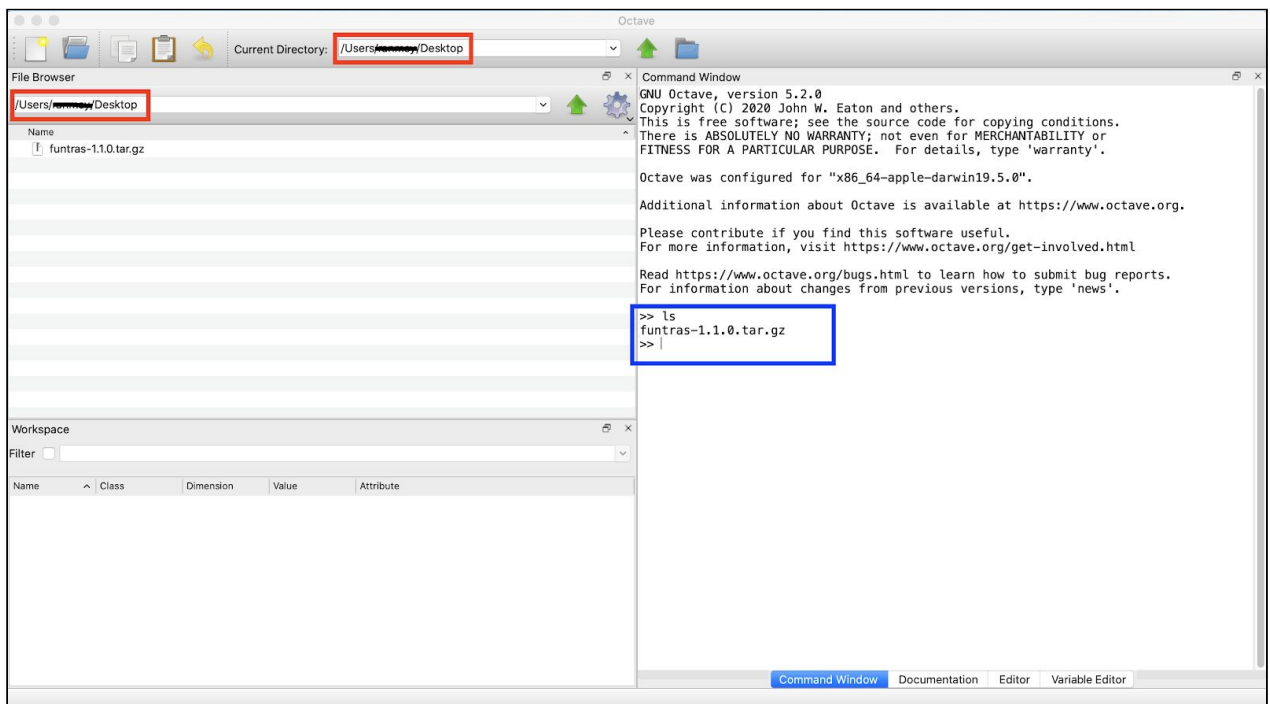
- I. Abrir el programa de GNU Octave, ya sea con GUI (interfaz gráfica) o CLI (terminal de comandos).

En este caso se va a utilizar esta guía mediante la interfaz gráfica disponible en la versión de Mac OS. Su proceso es el mismo para otros sistemas operativos.



- III. Una vez listo. Es importante mover el paquete FunTrans, bajo el nombre *funtras-1.1.0.tar.gz*¹, al directorio de trabajo. O en su defecto, cambiar el directorio de trabajo a donde esté ubicado el archivo. Tal y como se muestra en los cuadros en de color rojo.

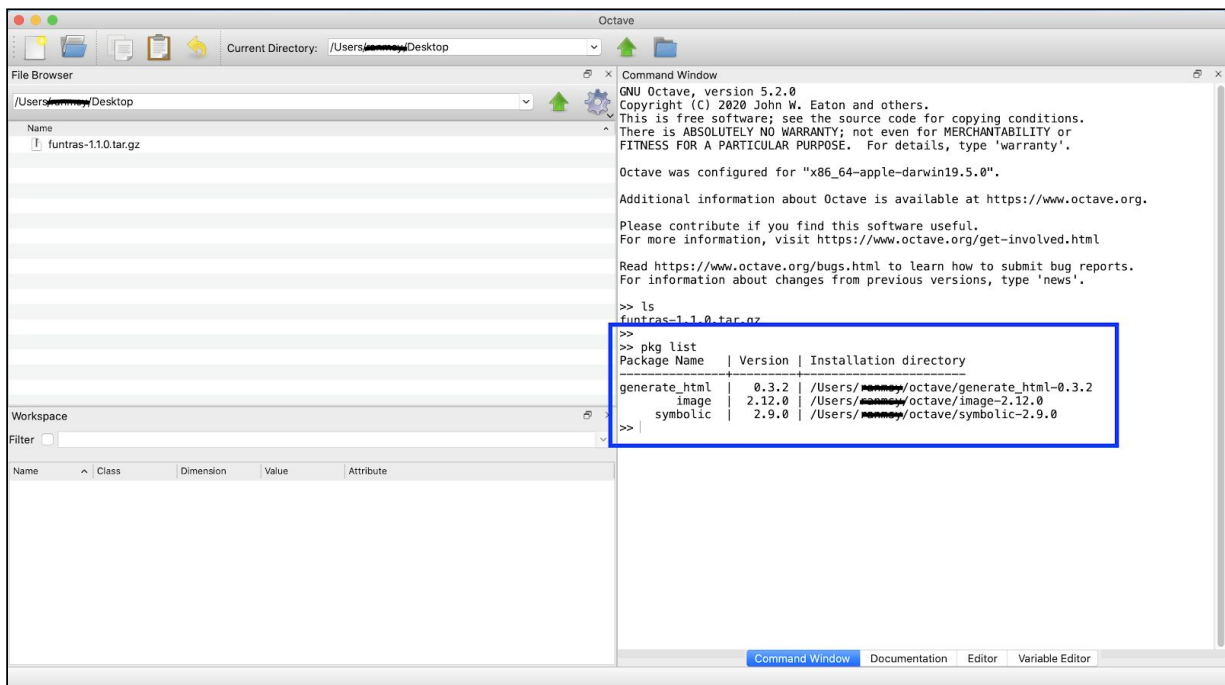
Para verificar que está trabajando donde está el archivo, se puede ejecutar el comando *ls* dentro de la consola o *Command View*. Una vez ejecutado, el resultado deben ser los archivos dentro del directorio. En este caso, se retorna el archivo *funtras-1.1.0.tar.gz*, como único elemento. Tal como se muestra en el cuadro de color azu.



En este caso se está trabajando en el *Desktop* (Escritorio).

¹ Este ejemplo se está realizando bajo la versión 1.1.0. La versión puede variar (en aumento) según sea la fecha de instalación.

- V. Completado el paso anterior. Se puede ejecutar el comando `pkg list`. El cual da como resultado todos los paquetes del tipo PKG instalados en el computador por medio de GNU Octave. Tal y como se muestra en el cuadro de color azul.

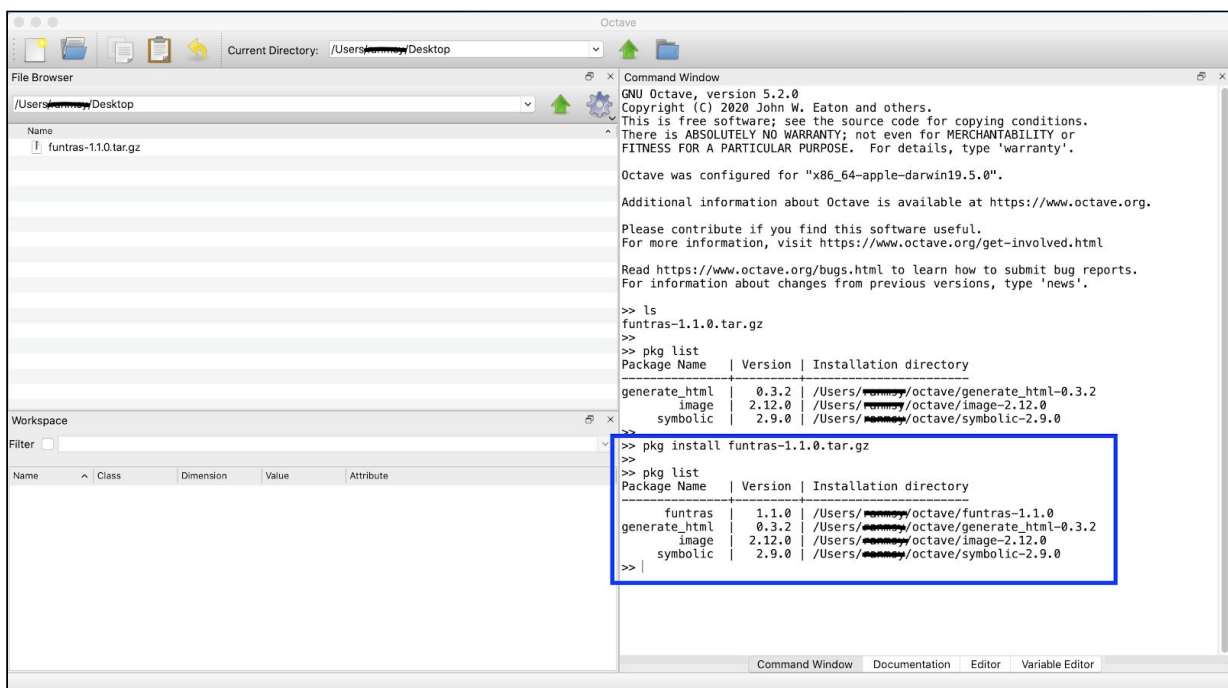


- VI. A continuación se procede a realizar la instalación. Esta se ejecuta mediante el comando `pkg install funtras-1.1.0.tar.gz`. Este debe ir con el nombre completo, incluyendo extensiones de `.tar.gaz`.

Este archivo está disponible en el siguiente enlace:
https://github.com/randyma01/CE3102-Tarea1/blob/master/parte_1/funtras-1.1.0.tar.gz

El comando puede tomar unos segundos (varía según cada computador). Una vez completado, no hay respuesta por parte de Octave, es decir hay una línea nueva `>>`, únicamente.

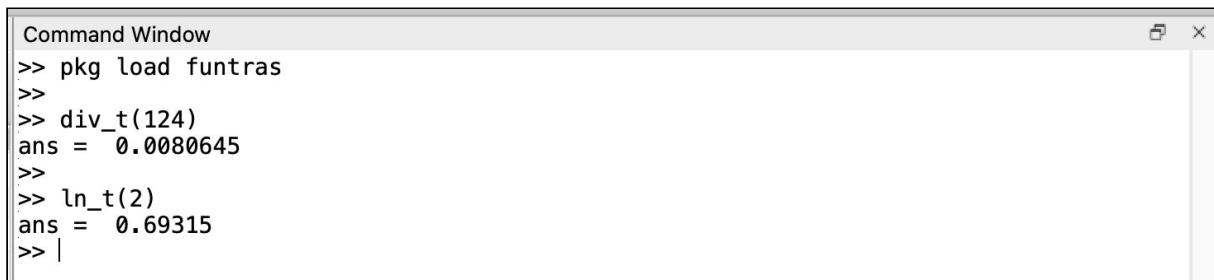
Para comprobar que el paquete fue instalado, se vuelve a ejecutar el comando `pkg list`. En este caso debe aparecer el nombre del paquete `funtras`, que acaba de ser instalado. Tal y como se muestra en el cuadro azul.



En caso de error en este paso. Volver a Paso II y verificar de nuevo. Asimismo revisar si el archivo está corrompido o dañado y volver a descargarlo. En caso de insistir el error favor ponerse en contacto con los autores.

3.4. Uso

Una vez listo la instalación del paquete, ya el programa está disponible para su utilización. No obstante, para hacer llamadas de las funciones se debe cargar el paquete primero con el comando *pkg load funtras*.



```
Command Window
>> pkg load funtras
>>
>> div_t(124)
ans = 0.0080645
>>
>> ln_t(2)
ans = 0.69315
>> |
```

Para ver las funciones disponibles, revisar Tabla 2.1 en la sección [2. FunTrans](#). Para más detalle de cada función disponible revisar sección [4. Funciones](#).

En caso de que haya error al cargar el paquete o que las funciones no den resultado correctamente. Desinstalar el paquete mediante el comando *pkg uninstall funtras* y volver a repetir los pasos de la instalación explicados en la subsección anterior, [3.3 Instalación](#).

4. Funciones

En esta sección se van a explicar detalladamente las funciones mencionadas en la sección [1. FunsTrans](#). Al ser 17 archivos diferentes con propósitos de constante y funciones, se organizaron en cuatro grupos diferentes: Aritmética, Valores Constantes, Expresiones Matemáticas y Trigonometría.

Todas las funciones al implementar series sucesivas de sumas, cumplen cuatro condiciones:

- I. Número de iteraciones máxima: $k = 2500$, utilizando un *while*.
- II. Tolerancia a error de: $tol = 10^{-8}$.
- III. El error² se calcula mediante la expresión: $| (f_{k+1}(a) - f_k(a)) |$.
Donde $f(a)$ representa la función de la sucesión en el a -ésimo término.
- IV. Todas las operaciones de división presentes en el paquete se ejecutan mediante la función `div_t(x)`.

Por lo tanto, todos los procesos iterativos tienen dos criterios de finalización o salida: $error < tol$ y $2500 \leq k$.

La explicación de las funciones se organizó de la siguiente manera:

- I. Firma: indica entrada, salida, objetivo y retorno.
- II. Desglose: Explicación del trasfondo matemático de las fórmulas aplicadas.

² La función `div_t(x)` no cumple esta condición.

- III. Ejecución: Capturas de pantalla de los resultados de las funciones.

4.1. Aritmética

Solo hay una única función de aritmética la cual es la función $\text{div_t}(x)$, la cual hace el cálculo aproximado de un número dado entre el valor constante de uno.

4.1.1. $\text{div_t}(x)$

Firma

Nombre	div_t
Recibe	x
Retorna	$1 \div x$
Restricciones	no se aceptan 0

Desglose

La función $\text{div_t}(x)$ representa la división de un número x entre uno, $1 \div x$. Es decir:

$$\text{div_t}(x) = \frac{1}{x}$$

Esto se logra mediante la siguiente expresión (*):

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

donde x_{k+1} representa el último valor cálculo y x_k el valor anterior.

El valor inicial al invocar al la función en su primera iteración es eps^a . Donde $eps =$

- eps^2 si $0! < a \leq 20!$
- eps^4 si $2! < a \leq 40!$
- eps^8 si $40! < a \leq 60!$
- eps^{11} si $60! < a \leq 80!$
- eps^{15} si $80! < a \leq 100!$
- eps^{20} si $100! < a \leq 120!$
- 0 si $120! < a \leq 140!$

Con $eps = 2.2004 \times 10^{-16}$

Esta función tiene dos condiciones de terminación. La cantidad máxima de iteraciones es eps^a si 2500 y el error se calcula mediante la fórmula del error relativo aplicada a la expresión (*):

$$| \{ (f_{k+1})(a) - f_k(a) \} \div x_{k+1} |.$$

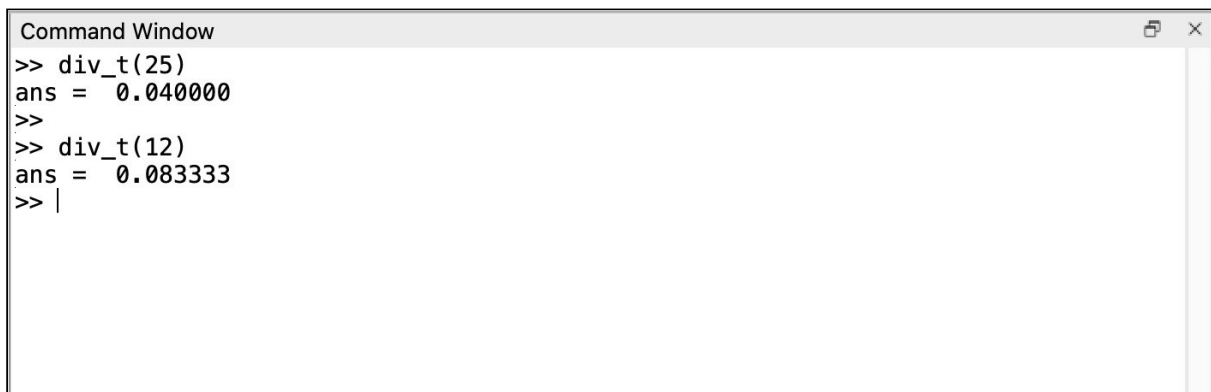
En cálculo del error de esta función, hay una operación de división, siendo esta la única excepción a la condición número IV.

Ejecución

Si se quiere el valor de $\frac{1}{25}$, se realiza el llamado de la función de la siguiente manera:

```
>> div_t(25)
```

donde el 25 representa el valor del numerador. Como se muestra en la siguiente captura.



```
Command Window
>> div_t(25)
ans = 0.040000
>>
>> div_t(12)
ans = 0.083333
>> |
```

Si se quisieran efectuar divisiones como $\frac{108}{3}$, simplemente se realiza la operación conjunta de:

```
>>> 108 * div_t(3)
```

dónde 108 es el multiplicando y el resultado de `div_t(3)` es el multiplicador.



```
Command Window
>> 108 * div_t(3)
ans = 36
>>
>> 25 * div_t(7)
ans = 3.5714
>> |
```


4.2. Constantes

Hay dos funciones que dan dos valores constantes: (1) [*exp_t\(x\)*](#) que calcula el valor de e elevado a un número y (2) [*pi_t\(\)*](#) que calcula el valor (π) .

4.2.1. *exp_t(x)*

Firma

Nombre	<i>exp_t</i>
Recibe	x
Retorna	e^x
Restricciones	-

Desglose

La función *exp_t(x)* representa elevar el valor de la constante e elevado a un valor dado, e^x . Es decir:

$$\text{exp_t}(x) = e^x$$

Esto se logra mediante la siguiente expresión:

$$S_k(x) = \sum_{n=0}^{2500} \frac{x^n}{n!}$$

$S_k(x)$ representa el último valor cálculo. La operación de la división se ejecuta invocando la función [*div_t\(x\)*](#).

Ejecución

Si se quiere el valor de e^4 , se realiza el llamado de la función de la siguiente manera:

```
>> exp_t(4)
```

donde el 4 representa el valor de la potencia. Como se muestra en la siguiente captura.



```
Command Window
>> exp_t(4)
ans = 54.598
>>
>> exp_t(1)
ans = 2.7183
>>
>> exp(63)
ans = 2.2938e+27
>> |
```

4.2.2. $pi_t()$

Firma

Nombre	<code>exp_t</code>
Recibe	-
Retorna	π
Restricciones	-

Desglose

La función `pi_t()` calcula un valor aproximado de la constante π .
Es decir:

$$pi_t(x) = \pi$$

Esto se logra mediante la serie de Leibniz, que calcula el valor de $\frac{\pi}{4}$, de la siguiente manera.

$$\pi \times \frac{1}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

Debido a su forma original, la serie fue modificada para la implementar la siguiente expresión³:

$$\pi = 4 \times \sum_{n=0}^{2500} \frac{x^n}{2n+1}$$

La operación de la división se ejecuta invocando la función `div_t(x)`.

Ejecución

Si se quiere el valor de π , se realiza el llamado de la función de la siguiente manera:

```
>> pi_t()
```



```
Command Window
>> pi_t()
ans = 3.1412
>> |
```

³ El valor de pi calculado no es exacto, es una aproximación con una desviación notable. Esto se debe a que se necesitan más iteraciones para obtener una mejor aproximación.

4.3. Expresiones

Hay cinco funciones que corresponden a expresiones matemáticas elementales, las cuales son:

- $\text{ln_t}(x)$, que calcula el valor del logaritmo natural de x .
- $\text{log_t}(x, a)$, que calcula el valor de logaritmo de x en base a .
- $\text{power_t}(x, a)$, la potencia es elevada en a .
- $\text{sqr_t}(x)$, que calcula la raíz cuadrada de x .
- $\text{root_t}(x, a)$, que calcula la raíz de x en base a .

4.3.1. $\text{ln_t}(x)$

Firma

Nombre	ln_t
Recibe	x
Retorna	$\ln(x)$
Restricciones	solo números positivos

Desglose

La función $\text{ln_t}(x)$ calcula un valor aproximado de la función del logaritmo natural, $\ln(x)$. Es decir:

$$\text{ln_t}(x) = \ln(x)$$

Esto se logra mediante la siguiente expresión:

$$S_k(x) = \frac{2(x-1)}{x+1} \times \sum_{n=0}^{2500} \frac{1}{2n+1} \times \left(\frac{x-1}{x+1}\right)^{2n}$$

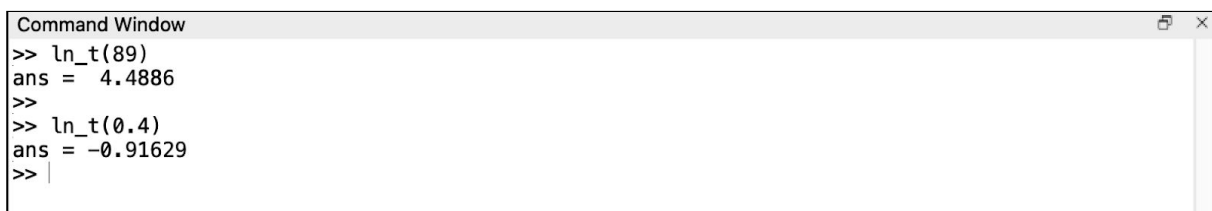
$S_k(x)$ representa el último valor cálculo. Las operaciones de las divisiones se ejecutan invocando la función `div_t(x)`.

Ejecución

Si se quiere el valor de $\ln(89)$, se realiza el llamado de la función de la siguiente manera:

```
>> ln_t(89)
```

donde el 4 representa el valor del logaritmo natural a calcular. Como se muestra en la siguiente captura.



```
Command Window
>> ln_t(89)
ans = 4.4886
>>
>> ln_t(0.4)
ans = -0.91629
>> |
```

4.3.2. $\log_t(x, a)$

Firma

Nombre	<code>log_t</code>
Recibe	<code>x,a</code>
Retorna	$\log_a(x)$
Restricciones	solo números positivos: <code>x, a</code>

Desglose

La función $\log_t(x, a)$ calcula un valor aproximado de la función del logaritmo en a de x , $\log_a(x)$. Es decir:

$$\log_t(x, a) = \log_a(x)$$

Esto se logra mediante las siguientes expresiones:

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

Esta función es una función compuesta, debido a que usa otras funciones para obtener una relación del cálculo del logaritmo en una base dada. Esto se realiza de la siguiente manera:

$$\begin{aligned} \text{nume} &= \ln_t(x) \\ \text{deno} &= \ln_t(a) \\ x &= \text{numerado} * \text{div_t}(\text{deno}) \end{aligned}$$

Donde $\text{div_t}(x)$ se usa para la división, y $\ln_t(x)$ se usa para el cálculo del logaritmo natural según los valores de x y a .

Ejecución

Si se quiere el valor de $\log(8,9)$, se realiza el llamado de la función de la siguiente manera:

```
>> log_t(8,9)
```

donde el 8 representa el valor del logaritmo a calcular y el 9 la base del logaritmo. Como se muestra en la siguiente captura.

```
Command Window
>> log_t(8,9)
ans = 0.94639
>>
>> log_t(12,4)
ans = 1.7925
>> |
```

4.3.3. *sqrt_t(x)*

Firma

Nombre	sqrt_t
Recibe	x
Retorna	\sqrt{x}
Restricciones	solo números positivos

Desglose

La función *sqrt_t(x, a)* calcula un valor aproximado de la función de raíz cuadrado de x, \sqrt{x} . Es decir:

$$\text{sqrt_t}(x) = \sqrt{x}$$

Esta función es una función compuesta, debido a usa otra función para obtener el valor aproximado de la raíz de x. Esto se realiza de la siguiente manera:

$$x = \text{root_t}(x, 2)$$

Donde *root_t(x, 2)* es la función que calcula la raíz dado un variable x y una base a.

Ejecución

Si se quiere el valor de $\sqrt{2}$, se realiza el llamado de la función de la siguiente manera:

```
>> sqrt_t(2)
```

donde el 2 representa el valor a calcular la raíz cuadrada. Como se muestra en la siguiente captura.

```
>> sqrt_t(2)
Symbolic pkg v2.9.0: Python communication link active, SymPy v1.5.1.
ans = 1.4142
>>
>> sqrt_t(16)
ans = 4
>> |
```

4.3.4. *root_t(x, a)*

Firma

Nombre	root_t
Recibe	x,
Retorna	$\sqrt[a]{x}$
Restricciones	-

Desglose

La función *root_t(x, a)* calcula un valor aproximado de la función de raíz de x en base a, $\sqrt[a]{x}$. Es decir:

$$\text{root_t}(x, a) = \sqrt[a]{x}$$

Esto se logra mediante la siguiente expresión:

$$f = (x^p - a)$$

A través de una deducción de la misma expresión anterior, se obtienen la expresión que se suma :

$$x_k = x_k - \left(\frac{f(x_k)}{f'(x_k)} \right)$$

El valor inicial x_k al invocar a la función en su primera iteración es $\frac{x}{2}$ y x es un parámetro de entrada. La operación de la división se ejecuta invocando la función [div_t\(x\)](#).

Ejecución

Si se quiere el valor de $\sqrt[3]{12}$, se realiza el llamado de la función de la siguiente manera:

```
>> root_t(12,3)
```

donde el 12 representa el valor a calcular en la base 3. Como se muestra en la siguiente captura.

```
>> root_t(12,3)
ans = 2.2894
>>
>> root_t(54,3)
ans = 3.7798
>> |
```

Para la ejecución de esta función es importante la invocación y carga previa del paquete *Symbolic*, [explicado anteriormente](#), puesto que en la deducción de la expresión iterativa hay una derivada que se trabaja por medio de este paquete.

4.4. Trigonometría

Hay nueve funciones que corresponden a expresiones de trigonometría las cuales son:

- seno: *sin_t()*
- coseno: *cos_t(x)*
- tangente: *tan_t(x)*
- arcoseno: *asin_t(x)*
- arcocoseno: *acos_t(x)*
- arcotangente: *atan_t(x)*
- seno hiperbólico: *sinh_t(x)*
- coseno hiperbólico: *cosh_t(x)*
- tangente hiperbólico: *tanh_t(x)*

4.4.1. *sin_t(x)*

Firma

Nombre	<i>sin_t</i>
Recibe	x
Retorna	<i>sen(x)</i>
Restricciones	-

Desglose

La función `sin_t(x)` calcula un valor aproximado de la función seno, $\text{sen}(x)$. Es decir:

$$\text{sin}_t(x) = \text{sen}(x)$$

Esto se logra mediante la siguiente expresión:

$$S_k(x) = \sum_{n=0}^{2500} (-1)^n \times \frac{x^{2n+1}}{(2n+1)!}$$

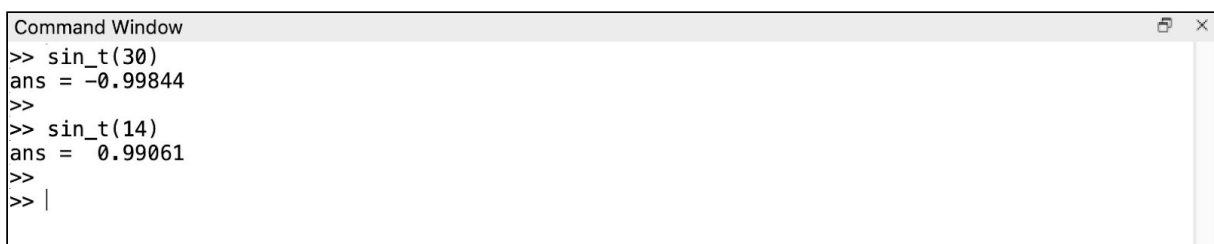
$S_k(x)$ representa el último valor cálculo. La operación de la división se ejecuta invocando la función `div_t(x)`.

Ejecución

Si se quiere el valor de $\text{sin}(30)$, se realiza el llamado de la función de la siguiente manera:

```
>> sin_t(30)
```

donde el 30 representa el ángulo. Como se muestra en la siguiente captura.



```
Command Window
>> sin_t(30)
ans = -0.99844
>>
>> sin_t(14)
ans = 0.99061
>>
>> |
```

Es importante aclarar que este resultado que se obtiene está en radianes.

4.4.2. $\text{cos_t}(x)$

Firma

Nombre	cos_t
Recibe	x
Retorna	$\cos(x)$
Restricciones	-

Desglose

La función $\text{cos_t}(x)$ calcula un valor aproximado de la función coseno, $\cos(x)$. Es decir:

$$\text{cos_t}(x) = \cos(x)$$

Esto se logra mediante la siguiente expresión:

$$S_k(x) = \sum_{n=0}^{2500} (-1)^n \times \frac{x^{2n}}{2n!}$$

$S_k(x)$ representa el último valor cálculo. La operación de la división se ejecuta invocando la función [\$\text{div_t}\(x\)\$](#) .

Ejecución

Si se quiere el valor de $\cos(24)$, se realiza el llamado de la función de la siguiente manera:

```
>> cos_t(24)
```

donde el 4 representa el valor del ángulo. Como se muestra en la siguiente captura.

```
Command Window
>> cos_t(24)
ans = 0.42418
>>
>> cos_t(1)
ans = 0.54030
>> |
```

Es importante aclarar que este resultado que se obtiene está en radianes.

4.4.3. *tan_t(x)*

Firma

Nombre	<i>tan_t</i>
Recibe	x
Retorna	<i>tan(x)</i>
Restricciones	-

Desglose

La función *tan_t(x)* calcula un valor aproximado de la función tangente, *tan(x)*. Es decir:

$$\text{tan_t}(x) = \text{tan}(x)$$

Esto se logra mediante la siguiente expresión:

$$x = \frac{\text{sen}(x)}{\text{cos}(x)}$$

El valor de la función seno, $\text{sen}(x)$, se calcula por medio de $\text{sin_t}(x)$ y el valor de la función de coseno, $\text{cos}(x)$, se calcula por medio de $\text{cos_t}(x)$. Luego, la operación de la división se ejecuta invocando la función $\text{div_t}(x)$.

Ejecución

Si se quiere el valor de $\tan(0.1)$, se realiza el llamado de la función de la siguiente manera:

```
>> tan_t(0.1)
```

donde el 0.1 representa el valor del ángulo. Como se muestra en la siguiente captura.



```
Command Window
>> tan_t(0.1)
ans = 0.10033
>>
>> tan_t(-0.3)
ans = -0.30934
>> |
```

Es importante aclarar que este resultado que se obtiene está en radianes.

4.4.4. *asin_t(x)*

Firma

Nombre	<i>asin_t</i>
Recibe	x
Retorna	$\text{sen}^{-1}(x)$
Restricciones	parámetro dentro de [0,1]

Desglose

La función *asin_t(x)* calcula un valor aproximado de la función arcoseno, $\text{sen}^{-1}(x)$. Es decir:

$$\text{asin_t}(x) = \text{sen}^{-1}(x)$$

Esto se logra mediante la siguiente expresión:

$$S_k(x) = \sum_{n=0}^{2500} x^{2n+1} \times \frac{(2n)!}{4^n \times (n!)^2 \times (2n+1)}$$

La operación de la división se ejecuta invocando la función *div_t(x)*.

Ejecución

Si se quiere el valor de $\text{sen}^{-1}(0.3)$, se realiza el llamado de la función de la siguiente manera:

```
>> asin_t(0.3)
```

donde el 0.3 representa el valor del ángulo. Como se muestra en la siguiente captura.



```
Command Window
>> asin_t(0.3)
ans = 0.30469
>>
>> asin_t(0.012)
ans = 0.012000
>> |
```

Es importante aclarar que este resultado que se obtiene está en radianes.

4.4.5. $\text{acos}_t(x)$

Firma

Nombre	acos_t
Recibe	x
Retorna	$\cos^{-1}(x)$
Restricciones	parámetro dentro de $[0,1]$

Desglose

La función `acos_t(x)` calcula un valor aproximado de la función arcoseno, $\cos^{-1}(x)$. Es decir:

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

Esto se logra mediante la siguiente expresión:

$$x = \left(\pi \times \frac{1}{2}\right) - \sin^{-1}(x)$$

El valor de π , se calcula mediante la función `pi_t(x)`, la función arcoseno, $\sin^{-1}(x)$, se calcula mediante la función `asin_t(x)`.

Ejecución

Si se quiere el valor de $\cos^{-1}(0.1)$, se realiza el llamado de la función de la siguiente manera:

```
>> acos_t(0.1)
```

donde el 0.1 representa el valor del ángulo. Como se muestra en la siguiente captura.

```
>> acos_t(0.1)
ans = 1.4704
>>
>> acos_t(0.07)
ans = 1.5005
>>
```

Es importante aclarar que este resultado que se obtiene está en radianes.

4.4.6. *atan_t(x)*

Firma

Nombre	<i>atan_t</i>
Recibe	<i>x</i>
Retorna	$\tan^{-1}(x)$
Restricciones	parámetro dentro de [-1, 1]

Desglose

La función *atan_t(x)* calcula un valor aproximado de la función arcotangente, $\tan^{-1}(x)$. Es decir:

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

Esto se logra mediante la siguiente expresión:

$$S_k(x) = \sum_{n=0}^{2500} (-1)^n \times \frac{x^{2n+1}}{2n+1}$$

La operación de la división se ejecuta invocando la función *div_t(x)*.

Ejecución

Si se quiere el valor de $\tan^{-1}(0.1212)$, se realiza el llamado de la función de la siguiente manera:

```
>> asin_t(0.1212)
```

donde el 0.1212 representa el valor del ángulo. Como se muestra en la siguiente captura.

```
Command Window
>> atan_t(0.1212)
ans = 0.12061
>>
>> atan_t(0.9999)
ans = 0.78541
>>
```

Es importante aclarar que este resultado que se obtiene está en radianes.

4.4.7. $\sinh_t(x)$

Firma

Nombre	\sinh_t
Recibe	x
Retorna	$\sinh(x)$
Restricciones	-

Desglose

La función `sinh_t(x)` calcula un valor aproximado de la función seno hiperbólico, $\sinh(x)$. Es decir:

$$\sinh_t(x) = \sinh(x)$$

Esto se logra mediante la siguiente expresión:

$$S_k(x) = \sum_{n=0}^{2500} \frac{x^{2n+1}}{(2n+1)!}$$

$S_k(x)$ representa el último valor cálculo. La operación de la división se ejecuta invocando la función `div_t(x)`.

Ejecución

Si se quiere el valor de $\sinh(0.1)$, se realiza el llamado de la función de la siguiente manera:

```
>> sinh_t(0.1)
```

donde el 0.1 representa el valor del ángulo. Como se muestra en la siguiente captura.



```
Command Window
>> sinh_t(0.1)
ans = 0.10017
>>
>> sinh_t(0.03)
ans = 0.030005
>> |
```

Es importante aclarar que este resultado que se obtiene está en radianes.

4.4.8. *cosh_t(x)*

Firma

Nombre	<i>cosh_t</i>
Recibe	x
Retorna	<i>cosh(x)</i>
Restricciones	-

Desglose

La función *cosh_t(x)* calcula un valor aproximado de la función seno hiperbólico, *cosh(x)*. Es decir:

$$\text{cosh_t}(x) = \text{cosh}(x)$$

Esto se logra mediante la siguiente expresión:

$$S_k(x) = \sum_{n=0}^{2500} \frac{x^{2n+1}}{(2n)!}$$

$S_k(x)$ representa el último valor cálculo. La operación de la división se ejecuta invocando la función *div_t(x)*.

Ejecución

Si se quiere el valor de *cosh(0.4)*, se realiza el llamado de la función de la siguiente manera:

```
>> cosh_t(0.4)
```

donde el 0.4 representa el valor del ángulo. Como se muestra en la siguiente captura.

```
Command Window
>> cosh_t(0.4)
ans = 1.0811
>>
>> cosh_t(0.002)
ans = 1.0000
>> |
```

Es importante aclarar que este resultado que se obtiene está en radianes.

4.4.9. *tanh_t(x)*

Firma

Nombre	<i>tanh_t</i>
Recibe	x
Retorna	<i>tanh(x)</i>
Restricciones	-

Desglose

La función *tanh_t(x)* calcula un valor aproximado de la función tangente hiperbólico, *tanh(x)*. Es decir:

$$\text{tanh_t}(x) = \text{tanh}(x)$$

Esto se logra mediante la siguiente expresión:

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

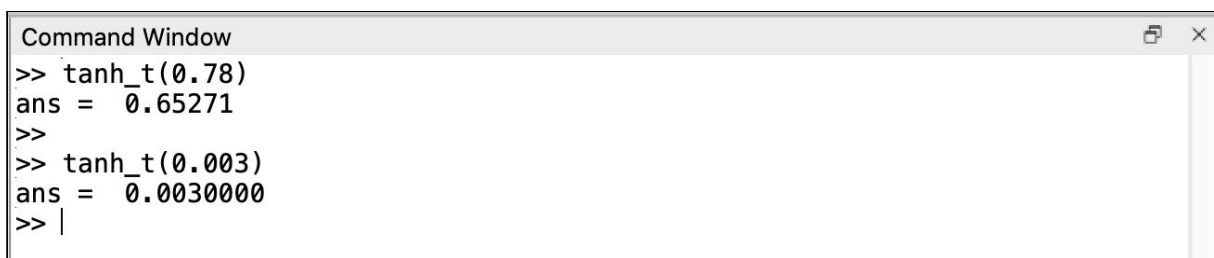
El valor de la función seno hiperbólico, $\sinh(x)$, se calcula por medio de `sinh_t(x)` y el valor de la función de coseno hiperbólico, $\cosh(x)$, se calcula por medio de `cosh_t(x)`. Luego, la operación de la división se ejecuta invocando la función `div_t(x)`.

Ejecución

Si se quiere el valor de $\tanh(0.78)$, se realiza el llamado de la función de la siguiente manera:

```
>> tanh_t(0.78)
```

donde el 0.78 representa el valor del ángulo. Como se muestra en la siguiente captura.



```
Command Window
>> tanh_t(0.78)
ans = 0.65271
>>
>> tanh_t(0.003)
ans = 0.0030000
>> |
```

Es importante aclarar que este resultado que se obtiene está en radianes.

5. Ejemplo

A continuación se va a mostrar un uso en conjunto de algunas funciones que forman parte del paquete. Para ilustrar este ejemplo, se va a calcular el valor de la siguiente expresión:

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

Para esto se escribieron las siguientes líneas:

```
ln_2 = ln_t(2);  
div_7 = div_t(7);  
tres_setimos = 3 * div_7;  
sen3_7 = sin_t(tres_setimos);  
arg_raiz = sen3_7 + ln_2;  
raiz_3 = root_t(arg_raiz, 3);  
raiz_2 = sqrt_t(2);  
senh_raiz_2 = sinh_t(raiz_2);  
deno = div_t(senh_raiz_2);  
frac = raiz_3 * deno;  
eul = exp_t(-1);  
tan_inv = atan_t(eul);  
respuesta = frac + tan_inv
```


El código anterior se escribió en un *script* bajo el nombre *test_funtras.m*⁴ y en la consola o *Command View* (como se explica en la sección de [Uso](#)) se escribió ese mismo nombre para la ejecución completa del script.

```
>> test_funtras
```

Una vez dado *Enter*, el proceso de cálculo tomará unos segundos. Según sea el computador donde puede tomar unos minutos, debido a la invocación del paquete Simbólico que se usa por debajo en el cálculo de la raíz.

El resultado final del cálculo es el siguiente.

```
respuesta = 0.88738  
>>
```

```
respuesta = 0.88738  
>>
```

⁴ Este *script* viene incluido con la instalación del paquete. Se puede invocar desde el *Command View* con la sentencia `test_funtras`.