

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/365187455>

GNU OCTAVE para Ingeniería

Book · November 2022

CITATIONS

0

READS

790

1 author:



Victor Aguilar Vidal

University of Concepción

47 PUBLICATIONS 105 CITATIONS

SEE PROFILE



UNIVERSIDAD
SAN SEBASTIAN

Facultad de Ingeniería, Arquitectura y Diseño
Plan Común de Ingeniería

GNU OCTAVE para Ingeniería

Autor: Víctor Aguilar Vidal, Ph.D.

A LOS ESTUDIANTES

Este documento fue escrito con la intención de apoyar su aprendizaje sobre ecuaciones diferenciales y programación básica. Espero que este apunte sea su puerta de entrada a la programación y automatización. Espero también que descubran el potencial que tienen OCTAVE y otras herramientas para hacerlos brillar en su profesión.

Índice

Introducción, instalación y comandos básicos.....	5
1.1.- ¿Qué es GNU OCTAVE y cuándo usarlo?	5
1.2.- Instrucciones de instalación	6
1.3.- Otros softwares o lenguajes de interés	9
Introducción al entorno y fuentes de ayuda.....	10
2.1.- Entorno	10
2.2.- Comandos y funciones básicas.....	13
2.3.- Algunas funciones útiles	16
2.4.- Resultados Inf y NaN	17
2.5.- Cancelar un comando o rutina.....	17
2.6.- Help GNU OCTAVE	17
Definición de variables, operaciones e indexación	20
3.1.- Definición de números, vectores y matrices	20
3.2.- Algunas funciones útiles	25
3.3.- Aritmética de números, vectores y matrices	26
3.4.- Operaciones punto a punto.....	27
3.5.- Indexación tácita y lógica.....	28
3.6.- Ejemplos de aplicación.....	33
3.6.1 Sistemas de ecuaciones lineales	33
3.6.2 Raíces de un polinomio	35
3.6.3 Raíces de ecuaciones trascendentes	37
Gráficos sencillos con GNU OCTAVE	38
4.1.- Entorno script	39
4.2.- Gráficos a partir de vectores.....	42
4.3.- Gráfico directo de funciones	47
Introducción a la programación en GNU OCTAVE	48
5.1.- Entorno script	48
5.2.- Funciones	51
5.3.- Control if	53
5.3.1 Operadores booleanos	55
5.4.- Ciclo for	56
5.5.- Otros controles disponibles	57
5.6.- Aplicación algoritmo babilónico.....	57

Aplicaciones a ecuaciones diferenciales ordinarias	60
6.1.- Conceptos fundamentales.....	60
6.1.1 Commando lsode()	61
6.2.- Ejemplos resueltos.....	62
6.2.1 Ejemplo EDO con raíz, potencia, logaritmo	62
6.2.2 Ejemplo EDO con función exponencial	64
6.2.3 Ejemplo EDO con función trigonométrica.....	66
6.2.4 Ejemplo ley de enfriamiento de Newton.....	68
6.2.5 Ejemplo vibración libre amortiguada	70
6.2.6 Ejemplo vibración libre con amortiguamiento no-lineal.....	73
6.3.- Ejercicios propuestos.....	76
Referencias.....	79

Capítulo 1

Introducción, instalación y comandos básicos

El tiempo es un recurso muy valioso, por lo tanto, automatizar procesos se hace imprescindible. Los ingenieros necesitamos ser capaces de automatizar cálculos repetitivos pues la gran mayoría de los diseños que realizamos son procesos iterativos. Esta guía está dirigida a estudiantes de ingeniería estudiando ecuaciones diferenciales y también a aquellos que se están iniciando en la programación. Tal y como indica el título del documento, OCTAVE GNU es el software base para las actividades descritas en los próximos capítulos.

OCTAVE es un software libre para cálculos numéricos y producción de gráficos. Está diseñado principalmente para operar con vectores y matrices. Así, puede usarse para resolver sistemas de ecuaciones, problemas de valores y vectores propios, ecuaciones diferenciales y muchas otras aplicaciones. En muchos problemas de ingeniería la información puede ser expresada en términos de vectores y matrices y se pueden reducir a las formas de solución mencionadas. Además, OCTAVE tiene su propio lenguaje de programación que permite extender sus funcionalidades básicamente para ajustarse a cualquier requerimiento del usuario. OCTAVE facilita la solución de un amplio rango de problemas permitiendo en muchos casos automatizar cálculos y rutinas que de otra forma serían muy tediosas. OCTAVE se desarrolló originalmente como un software de apoyo para un curso de pregrado en diseño de reacciones químicas. Actualmente, se continúa desarrollando con el apoyo del Dr. J.W. Eaton y está liberado bajo la licencia abierta de GNU. Las funcionalidades de GNU OCTAVE se expanden continuamente y la mayoría del lenguaje y funciones son compatible con el lenguaje de MATLAB que es hasta el día de hoy un lenguaje muy común en la academia y en la industria (pero no es gratis). Esta introducción se escribió en base a la web de Long (1), el documento asociado y la web oficial de OCTAVE (2).

1.1.- ¿Qué es GNU OCTAVE y cuándo usarlo?

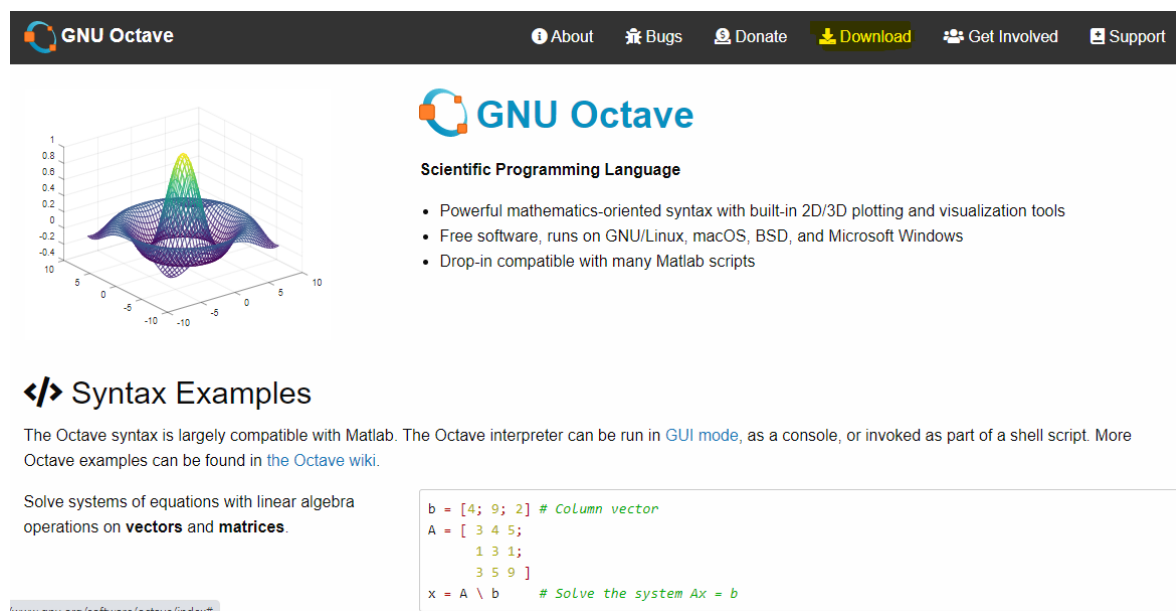
OCTAVE está diseñado para resolver problemas de forma numérica, esto es que se calculan valores discretos y se almacenan en la memoria del computador, esto significa que no siempre nos dará una respuesta exacta. No se debe confundir con programas como Mathematica o Maple que entregan soluciones simbólicas operando de forma algebraica. Esto no significa que un programa sea mejor que otro, simplemente tienen propósitos diferentes. La mayoría de los problemas reales de ingeniería no tienen una respuesta simbólica. Además, ¿cuándo necesitamos una respuesta exacta en la realidad? En ingeniería usamos respuestas aproximadas que nos permiten tomar decisiones.

C++ y Java son probablemente los lenguajes standard para desarrollo de software de propósitos generales. Sin embargo, la programación de operaciones matemáticas toma bastante tiempo de programación ya que C++ no incluye de forma nativa operaciones y funciones usuales en matemáticas. En cambio, OCTAVE contiene una batería de operaciones y funciones previamente incorporadas que los usuarios podemos simplemente utilizar directamente. De hecho, incluso desarrolladores que deben escribir sus programas en C++ realizan prototipos rápidos de algoritmos en OCTAVE o MATLAB para ahorrar tiempo de programación.

Esta introducción se escribió en base a la web de Long (1), el documento asociado y la web oficial de OCTAVE (2).

1.2.- Instrucciones de instalación

Primero debemos dirigirnos al sitio web donde se encuentra alojado GNU OCTAVE:
<https://www.gnu.org/software/octave/index> (3) y hacer clic en Download.



GNU Octave

About Bugs Donate **Download** Get Involved Support

GNU Octave

Scientific Programming Language

- Powerful mathematics-oriented syntax with built-in 2D/3D plotting and visualization tools
- Free software, runs on GNU/Linux, macOS, BSD, and Microsoft Windows
- Drop-in compatible with many Matlab scripts

</> Syntax Examples

The Octave syntax is largely compatible with Matlab. The Octave interpreter can be run in [GUI mode](#), as a console, or invoked as part of a shell script. More Octave examples can be found in [the Octave wiki](#).

Solve systems of equations with linear algebra operations on **vectors** and **matrices**.

```
b = [4; 9; 2] # Column vector
A = [3 4 5;
     1 3 1;
     3 5 9]
x = A \ b # Solve the system Ax = b
```

www.gnu.org/software/octave/index#

Veremos la información sobre la versión más reciente del software y se desplegará una lista horizontal con diferentes sistemas operativos, debe elegir el suyo.

Download

GNU Octave 6.3.0 is the latest stable release. (Release Notes: [6.1.0](#), [6.2.0](#), [6.3.0](#))

Source GNU/Linux BSD macOS MS Windows

Si es usuario de MS Windows haga clic en MS Windows y elija la arquitectura que corresponde a su computador.

Microsoft Windows

Note: All installers below bundle several **Octave packages** so they don't have to be installed separately. After installation type `pkg list` to list them. [Read more.](#)

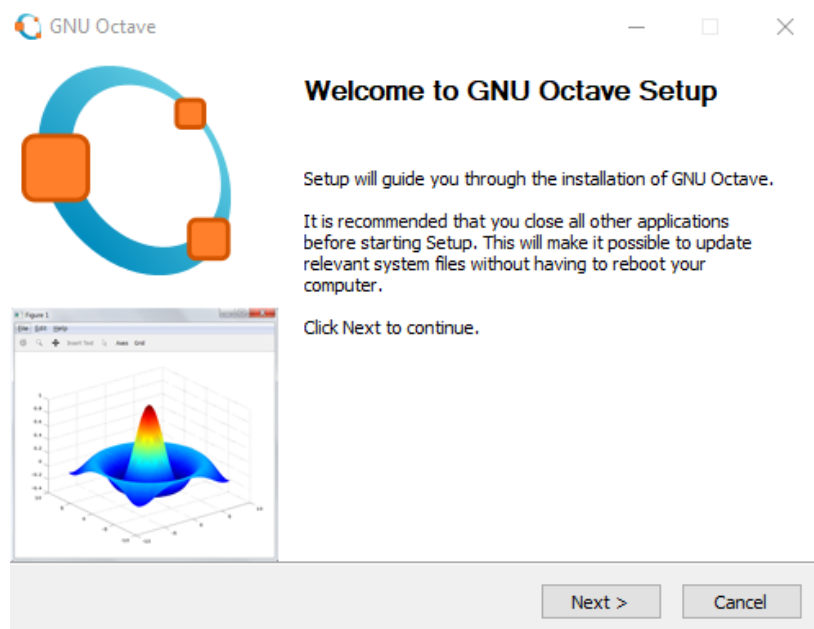
- Windows-64 (recommended)
 - [octave-6.3.0-w64-installer.exe](#) (~ 325 MB) [signature]
 - [octave-6.3.0-w64.7z](#) (~ 319 MB) [signature]
 - [octave-6.3.0-w64.zip](#) (~ 568 MB) [signature]
- Windows-32 (old computers)
 - [octave-6.3.0-w32-installer.exe](#) (~ 319 MB) [signature]
 - [octave-6.3.0-w32.7z](#) (~ 311 MB) [signature]
 - [octave-6.3.0-w32.zip](#) (~ 531 MB) [signature]
- Windows-64 (64-bit linear algebra for large data)

Unless your computer has more than ~32GB of memory and you need to solve linear algebra problems with arrays containing more than ~2 billion elements, this version will offer no advantage over the recommended Windows-64 version above.

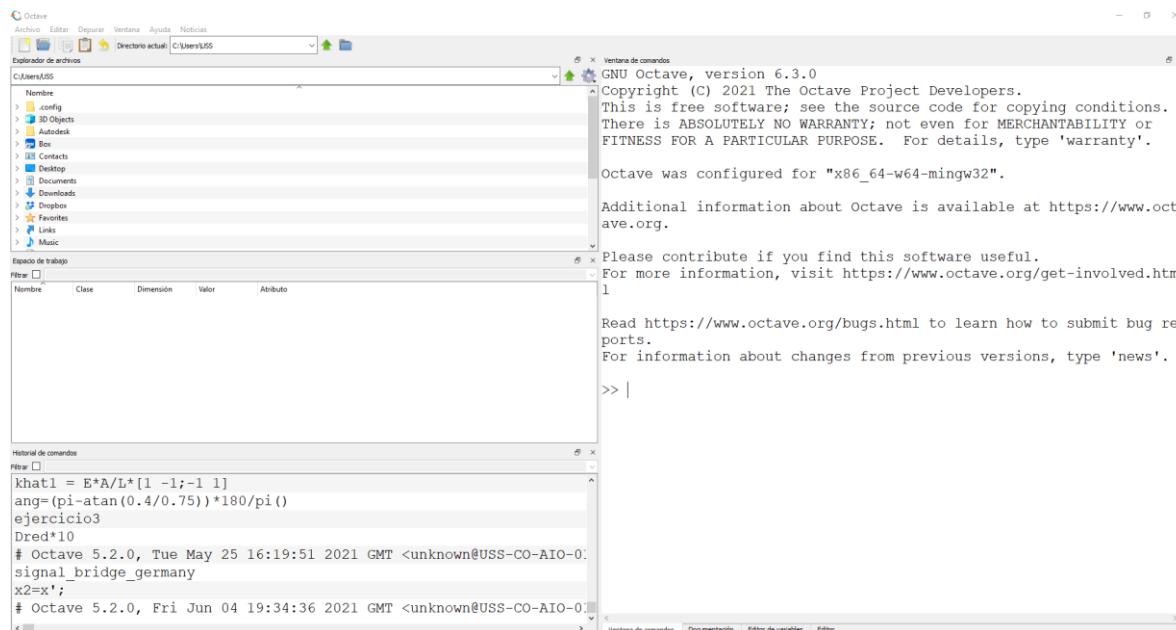
 - [octave-6.3.0-w64-64-installer.exe](#) (~ 326 MB) [signature]
 - [octave-6.3.0-w64-64.7z](#) (~ 319 MB) [signature]
 - [octave-6.3.0-w64-64.zip](#) (~ 568 MB) [signature]

La arquitectura de 32 bits sólo es usada hoy en computadores bastante antiguos. Normalmente debe seleccionar el instalador de 64 bits. Puede descargar el archivo .exe para instalar de inmediato, o puede descargar el instalador de manera comprimida en .7z o .zip si prefiere sólo guardar y transportar el archivo a otro computador e instalar más tarde. Le recomiendo descargar el archivo .exe e instalar de inmediato. Note que estos instaladores ya contienen muchas funcionalidades instaladas de forma que podremos abrir el software y comenzar de inmediato a utilizarlo, sin necesidad de instalar otros paquetes. Eventualmente, quizás usted como usuario debe instalar paquetes especiales para su aplicación.

Al ejecutar el .exe se lanzará la instalación de OCTAVE.



Puede aceptar las opciones preseleccionadas y continuar. Si desea instalar OCTAVE en un directorio especial, podrá realizar esa selección en esta etapa. Al finalizar pueden ejecutar GNU OCTAVE y abrirá un entorno como este:



1.3.- Otros softwares o lenguajes de interés

Existen alternativas a OCTAVE, por su puesto. Por ejemplo, SciLAB (4) es similar en lenguaje y también es gratuito. R studio (5) es otro software abierto que tiene múltiples propósitos, quizás menos orientado a matemáticas, permite análisis de datos de manera muy cómoda y brilla por su habilidad para la estadística; sin embargo, cada vez se desarrollan más y más librerías que tienen propósitos variados y amplían sus posibilidades de aplicación. Adicionalmente, existen lenguajes de programación de propósito general como Python (6), C++ (7) y Fortran (8). La mala noticia es que todos los lenguajes son diferentes, la buena noticia es que las habilidades que adquieres en uno de ellos son conceptualmente transferibles a otro.

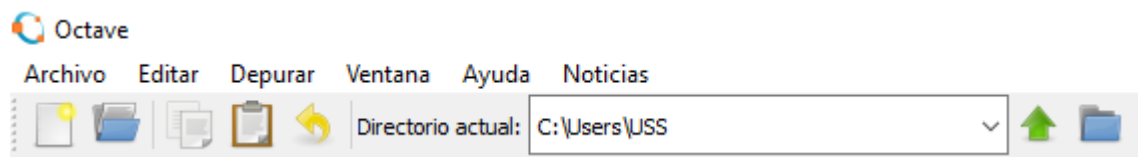
Capítulo 2

Introducción al entorno y fuentes de ayuda

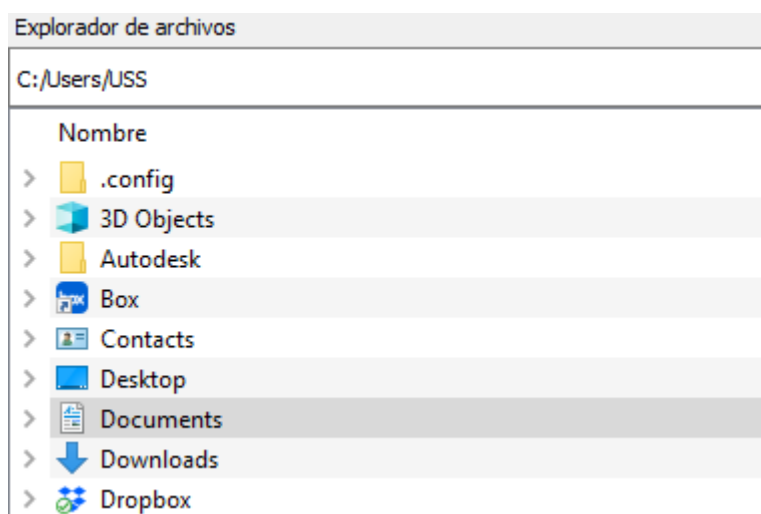
2.1.- Entorno

Dentro de la pantalla inicial al abrir el software OCTAVE veremos algunos espacios con los que debemos estar familiarizados.

- A. La barra superior de menús es bastante común a otros softwares de uso habitual donde están contenidas las opciones de archivo como guardar y abrir documentos y editar algunas preferencias. La opción depurar tiene que ver con revisar códigos. Ventana nos permite configurar que espacios de trabajo queremos ver. Además, hay dos menús que pueden ser de importancia cuando queremos aprender más sobre el software: ayuda y noticias. Es importante notar en que directorio estamos trabajando, en este ejemplo, el directorio corresponde al disco duro C: dentro de la carpeta Users, dentro de la sub carpeta USS.



- B. El explorador de archivos nos indica que archivos se encuentran en el actual directorio de trabajo. Podemos abrir archivos de OCTAVE (extensión .m) desde acá.



- C. El *espacio de trabajo* indica las variables definidas en la memoria de OCTAVE. Inicialmente el *espacio de trabajo* estará vacío. Al definir algunas variables en el *espacio de trabajo* se indica su nombre, clase, dimensión, valor y eventualmente algún atributo de la variable.

Espacio de trabajo				
Filtrar <input type="checkbox"/>				
Nombre	Clase	Dimensión	Valor	Atributo

Por ejemplo, en la siguiente figura se muestra una variable *A* de clase doble precisión (esto dice relación con la representación binaria de los números en el computador), tiene tres filas y dos columnas (es una matriz) y sus valores están indicados. La variable *a* es de doble precisión, de uno por uno (un valor escalar) y vale cinco. Finalmente, la variable *ans* (corto para answer) es un escalar de doble precisión que vale 0.1362.

Filtrar <input type="checkbox"/>			
Nombre	Clase	Dimensión	Valor
A	double	3x2	[5, 2; 7, 9; 9, 7]
a	double	1x1	5
ans	double	1x1	0.1362

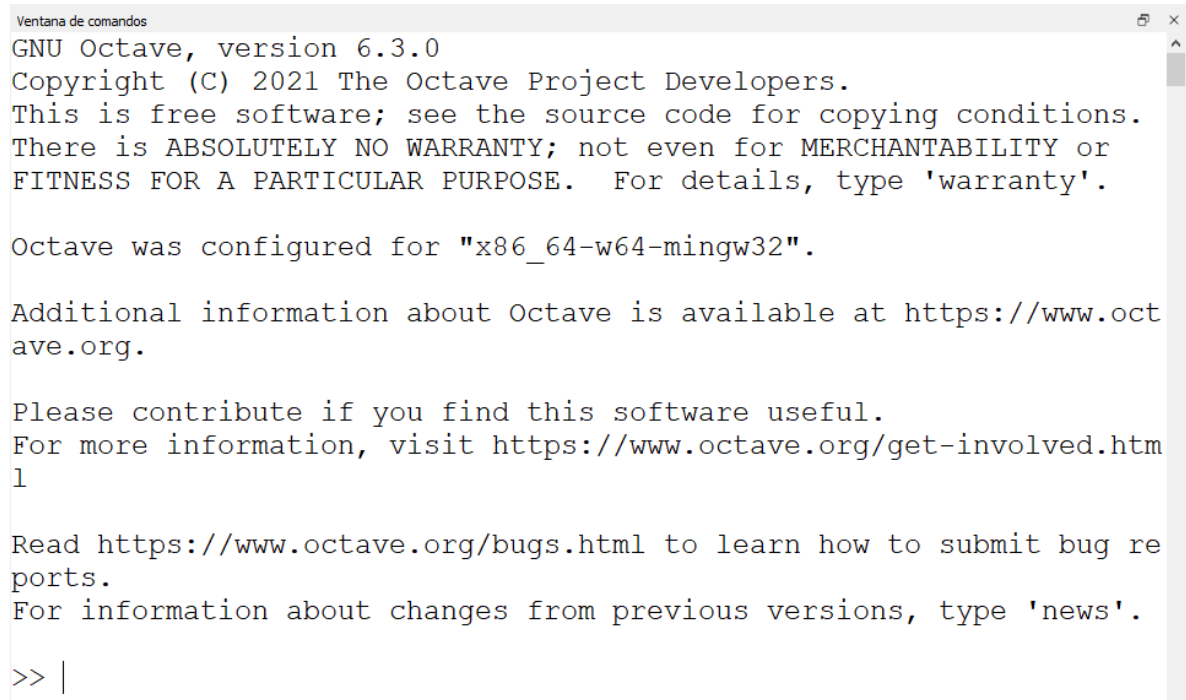
- D. El *historial de comandos* muestra todos los comandos ejecutados previamente en OCTAVE, en este caso se puede ver la definición de las variables *khat1* y *ang*, además de la ejecución de una rutina llamada *ejercicio3*. Entre otros.

```

Historial de comandos
Filtrar ☐
khat1 = E*A/L*[1 -1;-1 1]
ang=(pi-atan(0.4/0.75))*180/pi()
ejercicio3
Dred*10
# Octave 5.2.0, Tue May 25 16:19:51 2021
signal_bridge_germany
x2=x';
# Octave 5.2.0, Fri Jun 04 19:34:36 2021

```

- E. Y finalmente la *ventana de comandos* que es el corazón de OCTAVE. En este espacio se ejecutan todas nuestras instrucciones. Cada vez que iniciemos el software nos dará la bienvenida con un mensaje indicando la versión con la que estamos trabajando y un descargo de responsabilidad típico de los softwares libres. Cuando veamos el símbolo >> quiere decir que OCTAVE está listo para ejecutar la próxima instrucción que le demos.

A screenshot of a command window titled "Ventana de comandos". The window contains the following text: "GNU Octave, version 6.3.0", "Copyright (C) 2021 The Octave Project Developers.", "This is free software; see the source code for copying conditions.", "There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.", "Octave was configured for \"x86_64-w64-mingw32\".", "Additional information about Octave is available at https://www.octave.org.", "Please contribute if you find this software useful.", "For more information, visit https://www.octave.org/get-involved.html", "Read https://www.octave.org/bugs.html to learn how to submit bug reports.", "For information about changes from previous versions, type 'news'.", and at the bottom, ">> |".

```
Ventana de comandos
GNU Octave, version 6.3.0
Copyright (C) 2021 The Octave Project Developers.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at https://www.oct
ave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.htm
l

Read https://www.octave.org/bugs.html to learn how to submit bug re
ports.
For information about changes from previous versions, type 'news'.

>> |
```

2.2.- Comandos y funciones básicas

Al igual que en una calculadora científica o programable, los cálculos a los que no se asignan a una variable en concreto se asignan a la variable de respuesta por defecto que es `ans` (del inglés para respuesta, answer). Escriba lo siguiente en la *ventana de comandos*:

```
>> 3+6  
ans = 9
```

Si realizamos la misma operación, pero al cálculo se asigna a una variable, en este caso `x`, el resultado quedará guardado en dicha variable:

```
>> x=3+6  
x = 9
```

Ahora `x` aparece en el espacio de trabajo y para conocer el valor de esta variable basta teclear su nombre y presionar enter. ¡Pruébalo!

```
>> x  
x = 9
```

Si se añade un punto y coma (;) al final de una instrucción, OCTAVE no muestra la respuesta. A esto se le conoce como ocultar el eco. Haga la siguiente prueba:

```
>> y=25*4;
```

OCTAVE no ha devuelto ningún resultado, pero inspeccione el *espacio de trabajo* y verá que la variable `y` ha aparecido. Puede consultar el valor de la variable igual en el ejemplo anterior.

```
>> y  
y = 100
```

Como se explicó en la introducción, OCTAVE posee muchas funciones matemáticas habituales previamente cargadas. Por ejemplo, la función coseno,

```
>> cos(pi)
ans = -1
>> cos(pi/2)
ans = 6.1230e-17
```

Note que `pi` es un valor previamente definido en la memoria (el irracional π). Pero $\cos(\pi/2) = 0$ ¿o no? Claro que sí, pero estarás de acuerdo con que `6.1230e-17` es lo bastante cercano a cero para no preocuparnos. Un caso similar se da al calcular $\tan(\pi/2)$, inténtalo.

Otras variables notables ya están definidas también, el número de Euler (e) y el imaginario $i = \sqrt{-1}$. Verifique lo siguiente:

```
>> pi
ans = 3.1416
>> e
ans = 2.7183
>> i
ans = 0 + 1i
```

Podemos ver que además OCTAVE no tiene dificultades para operar con números complejos.

Hasta el momento sólo hemos definido dos variables y ambas deben estar almacenadas en el espacio de trabajo, así como también lo estará la variable que guarda la última respuesta `ans`. Para conocer las variables que se han usado hasta el momento:

```
>> who
Variables visible from the current scope:
ans  x    y
```

Podemos conocer algunas características de esas variables si usamos el siguiente comando

```
>> whos
```

Variables visible from the current scope:

variables in scope: top scope

Attr	Name	Size	Bytes	Class
====	====	====	=====	=====
c	ans	1x1	16	double
	x	1x1	8	double
	y	1x1	8	double

Total is 3 elements using 32 bytes

Podemos borrar variables específicas usando la función `clear`.

```
>> clear x
```

Vea como `x` ya no existe en el espacio de trabajo. Si solicita el valor de `x` como antes obtendremos un error.

```
>> x
```

```
error: 'x' undefined near line 1, column 1
```

Podemos borrar todas las variables también usando el comando `clear all`

```
>> clear all
```

Ahora todas las variables han desaparecido del espacio de trabajo, incluso `ans`.

El comando `clc` limpiará la pantalla de la ventana de comandos.

```
>> clc
```

Ahora deben ver una pantalla limpia vacía. Puede navegar hacia arriba ▲ y hacia abajo ▼ por el historial de comandos presionando las flechas de su teclado para repetir alguna definición u operación realizada anteriormente.

2.3.- Algunas funciones útiles

OCTAVE tiene un gran número de funciones predefinidas listas para ser usadas. A continuación, se resumen algunas funciones que puede ser conveniente tener presentes para cálculos habituales en ingeniería ¡pruébelos!

Trigonometría

<code>sin(x)</code>	seno del ángulo x en radianes
<code>cos(x)</code>	coseno del ángulo x en radianes
<code>tan(x)</code>	tangente del ángulo x en radianes
<code>asin(x)</code>	arcoseno del valor x
<code>acos(x)</code>	arcocoseno del valor x
<code>atan(x)</code>	arcotangente del valor x

Funciones útiles

<code>abs(x)</code>	valor absoluto de x
<code>sign(x)</code>	signo de x
<code>round(x)</code>	redondear x al entero más cercano
<code>floor(x)</code>	redondear x hacia abajo
<code>sqrt(x)</code>	raíz cuadrada de x

Logaritmos y potencias

<code>a^b</code>	potencia de base a y exponente b
<code>log(x)</code>	logaritmo natural de x
<code>log10(x)</code>	logaritmo base 10 de x
<code>exp(x)</code>	exponencial de x

Funciones hiperbólicas

$\sinh(x)$	seno hiperbólico de x
$\cosh(x)$	coseno hiperbólico de x
$\tanh(x)$	tangente hiperbólica de x
$\operatorname{asinh}(x)$	inversa del seno hiperbólico de x
$\operatorname{acosh}(x)$	inversa del coseno hiperbólico de x
$\operatorname{atanh}(x)$	inversa de la tangente hiperbólica de x

2.4.- Resultados Inf y NaN

Eventualmente algunas operaciones pueden resultar en un resultado Inf (de Infinity), que quiere decir que se dividió por cero. El resultado de dividir cero entre cero entregará como resultado NaN (Not a Number). Ambos, Inf y NaN, se pueden tratar como cualquier otro número en OCTAVE, incluso se pueden asignar estos resultados a variables.

2.5.- Cancelar un comando o rutina

Ocasionalmente nos puede ocurrir que solicitamos ejecutar una rutina muy larga que tarda mucho en terminar. Esto suele ocurrir cuando la rutina muestra cálculos intermedios en pantalla, por ejemplo. A veces, un error de programación en nuestra rutina puede causar que se repita alguna operación indefinidamente. Si teclea **Ctrl+C** OCTAVE se detendrá.

2.6.- Help GNU OCTAVE

Existen varias maneras de obtener ayuda al usar OCTAVE. La más evidente es simplemente escribir en Google que es lo que quiere hacer y seguramente obtendrá resultados que incluyen ejemplos. La web www.octave.org (3) por su puesto tiene descripciones de todas las funciones predefinidas en el software. Como el lenguaje de OCTAVE es esencialmente el mismo que el de MATLAB, buscar ayuda de alguna función en www.mathworks.com (9) generalmente es buena idea. Adicionalmente, usando el comando **help** dentro de la ventana de comandos también encontrará una descripción de la función deseada. Veamos algunos ejemplos de cómo encontrar ayuda.

Probemos buscar ayuda en Google usando OCTAVE como una palabra clave. Por ejemplo, si buscamos como resolver polinomios con OCTAVE encontramos un documento y varios videos de ejemplo rápidamente.

Google resolver polinomios Octave

All Images Videos News Shopping More Tools

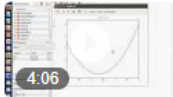
About 14,500 results (0.36 seconds)

<http://octaveintro.readthedocs.io> > po... · [Translate this page](#)

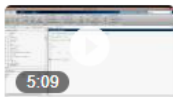
Polinomios — Introducción a Octave 1.0 documentation

En **Octave** los **polinomios** se representan por un vector de coeficientes ordenados de la mayor potencia hacia la menor. Por ejemplo, el **polinomio** $2x^3 - 3x^2 + \dots$


Videos



Raíces de Polinomios con Matlab/Octave
YouTube · Alvaro Delgado Mejía
Feb 18, 2017



04 Tutorial Básico Matlab (Polinomios) - Evaluación Polinomial
YouTube · Tutoriales de Ingeniería
Mar 15, 2017



MATLAB/OCTAVE: Multiplicación & División de Polinomios
YouTube · Samuel Chung
Apr 21, 2020

→ View all

Si repetimos la búsqueda, pero en inglés, encontramos la descripción oficial desde la web de OCTAVE.

Google solve polynomial Octave

All Images Videos News Shopping More Tools

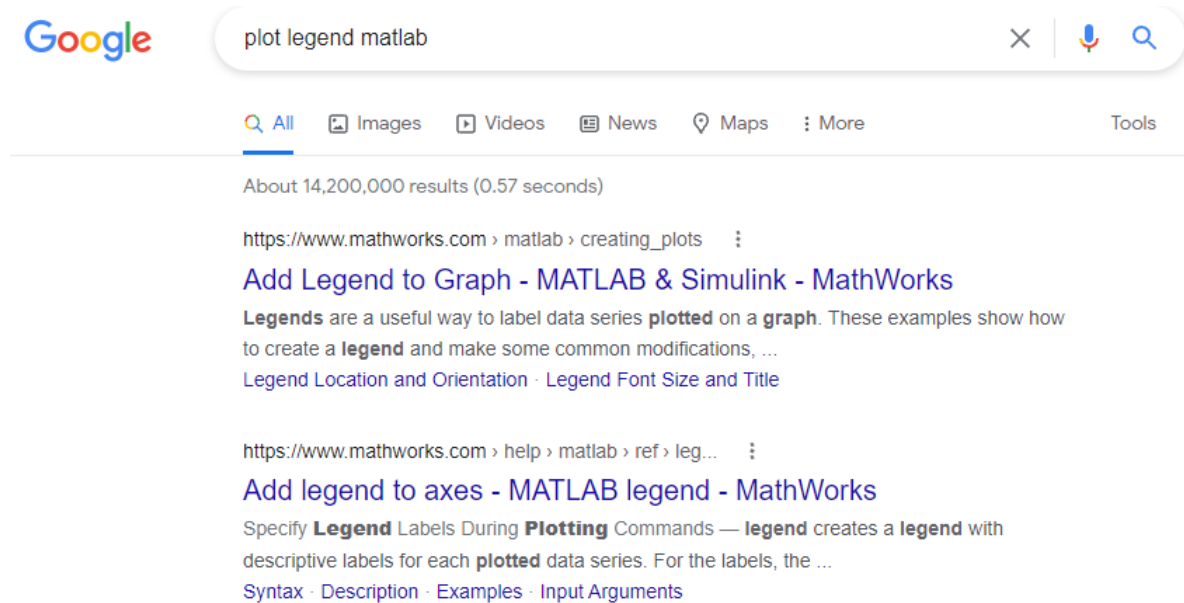
About 7,310,000 results (0.49 seconds)

<https://octave.org> > doc > Finding-Roots

Finding Roots - GNU Octave

Octave can find the roots of a given **polynomial**. This is done by computing the companion matrix of the **polynomial** (see the `compan` function for a definition), ...

Ahora bien, si utilizamos MATLAB como palabra clave en la búsqueda, esta vez sobre leyendas en gráficos, encontramos la web de mathworks. Mathworks es una compañía privada que ofrece soporte a sus usuarios, por lo tanto, sus fuentes de soporte son muy completas.



Pruebe dentro de la ventana de comandos puede escribir lo siguiente:

```
>> help log10
```

En pantalla se desplegará la siguiente información

```
'log10' is a built-in function from the file libinterp/corefcn/mappers.cc
-- log10 (X)
```

```
    Compute the base-10 logarithm of each element of X.
```

```
See also: log, log2, logspace, exp.
```

Additional help for built-in functions and operators is available in the online version of the manual. Use the command 'doc <topic>' to search the manual index.

Help and information about OCTAVE is also available on the WWW at <https://www.OCTAVE.org> and via the help@OCTAVE.org mailing list.

Capítulo 3

Definición de variables, operaciones e indexación

En este capítulo aprendemos algunas operaciones básicas que se pueden utilizar en OCTAVE. En un principio parecerán quizás muy abstractas y parecidas a sus primeras clases de álgebra lineal. Avancen un poco más y descubrirán lo poderoso que puede ser manejar información en vectores y matrices de forma ordenada.

3.1.- Definición de números, vectores y matrices

Para definir una variable basta con utilizar el signo igual:

```
>> g=9.81  
g = 9.8100
```

Ahora es posible operar con la variable `g`. Puede usarla para sumar, restar, elevar, cualquier cosa que se le ocurra, por ejemplo:

```
>> g=9.81  
g = 9.8100  
>> g-10  
ans = -0.1900  
>> g-pi^2  
ans = -0.059604  
>> 1/2*g*0.5^2  
ans = 1.2263  
>> log10(g)  
ans = 0.9917  
>> log(10)  
ans = 2.3026
```

Agreguemos una dimensión a nuestras variables. Para definir un vector fila ingresamos sus coordenadas entre corchetes separadas por espacios o comas:

```
>> v=[1 -2 3]
```

```
v =
```

```
1    -2    3
```

Para definir un vector columna ingresamos sus coordenadas entre corchetes separadas por punto y coma:

```
>> w=[9;11;-90]
```

```
w =
```

```
9
```

```
11
```

```
-90
```

El apostrofe al final permite transponer el vector:

```
>> v'
```

```
ans =
```

```
1
```

```
-2
```

```
3
```

Es posible definir vectores fila ingresando inicio, paso y final. Esta opción es muy útil cuando necesitamos definir valores de un eje independiente (eje x o un eje de tiempo). Crearemos un vector que inicia en 0.5 y termina en 2.0 saltando de 0.25 unidades.

```
>> x = [0.5:0.25:2]
```

```
x =
```

```
0.5000
```

```
0.7500
```

```
1.0000
```

```
1.2500
```

```
1.5000
```

```
1.7500
```

```
2.0000
```

En ocasiones puede ser más conveniente definir un vector entregando el número de elementos de este. La siguiente línea de comando creará un vector que inicia en 0 y llega hasta 10 con un total de 7 elementos.

```
t=linspace(0,10,7)
```

```
t =
```

```
    0    1.6667    3.3333    5.0000    6.6667    8.3333   10.0000
```

Agreguemos ahora una dimensión más. De manera similar a los vectores, las matrices se definen fila por fila como vectores separándolos por punto y coma. Definamos una matriz de 3x2 como sigue:

```
>> A=[4 6;pi e;57 -100]
```

```
A =
```

```
    4.0000    6.0000
    3.1416    2.7183
   57.0000  -100.0000
```

También se puede encontrar la transpuesta de **A** escribiendo **A'**. Pruébalo.

Podemos definir una matriz **B** a partir de los vectores **v** y **w** previamente definidos. Pero se debe ser cuidadoso de que las dimensiones de los vectores que conforman la matriz sean consistentes.

```
>> B=[v;w']
```

```
B =
```

```
    1    -2     3
    9    11   -90
```

Probemos crear una matriz con **v** y **w** y un vector nuevo conformado por tres nueves.

```
>> C=[v;w';9 9 9]
```

C =

```

1   -2   3
9   11  -90
9    9   9

```

Como ejercicio, ejecute distintas combinaciones de los vectores y matrices definidos para crear nuevas.

Hay una serie de matrices notables de mucha utilidad. La matriz identidad se crea con la función `eye()`, creemos una matriz identidad de 4x4

```
>> eye(4)
```

ans =

Diagonal Matrix

```

1   0   0   0
0   1   0   0
0   0   1   0
0   0   0   1

```

Una matriz nula, es decir, que sólo contiene ceros, la podemos crear con la función `zeros()`. Probemos crear una matriz nula de 2x2

```
>> zeros(2)
```

ans =

```

0   0
0   0

```


Una matriz similar es aquella que sólo contiene unos. Probemos crear una matriz de unos de 2x4.

```
>> ones(2,4)
```

```
ans =
```

```
1  1  1  1
1  1  1  1
```

Si prefiere que sea cuadrada, sólo necesita indicar una dimensión dentro del paréntesis.

También es posible crear una matriz de números al azar entre 0 y 1. Ejemplo:

```
>> rand(3,4)
```

```
ans =
```

```
0.960146    0.948463    0.359775    0.152552
0.560043    0.791018    0.154724    0.287186
0.022235    0.824756    0.267236    0.288643
```

Explore los resultados de las siguientes operaciones

```
diag(v)
```

```
triu(C)
```

```
tril(C)
```

```
diag(C)
```

3.2.- Algunas funciones útiles

OCTAVE tiene un gran número de funciones predefinidas listas para ser usadas. A continuación, se resumen algunas funciones que puede ser conveniente tener presentes al operar con matrices y vectores.

<code>eye(n)</code>	matriz identidad de $n \times n$
<code>zeros(n)</code>	matriz de ceros de $n \times n$
<code>ones(n)</code>	matriz de unos de $n \times n$
<code>rand(n)</code>	matriz de números aleatorios de $n \times n$
<code>inv(A)</code>	inversa de la matriz A
<code>det(A)</code>	determinante de la matriz A
<code>trace(A)</code>	traza de la matriz A
<code>rank(A)</code>	rango de la matriz A
<code>eig(A)</code>	valores y vectores propios de la matriz A
<code>sum(A)</code>	suma de los elementos de la matriz A , columna a columna.
<code>min(A)</code>	mínimo valor de cada columna de la matriz A
<code>max(A)</code>	máximo valor de cada columna de la matriz A
<code>size(A)</code>	dimensiones de la matriz A , filas y columnas
<code>length(v)</code>	dimensión del vector v

3.3.- Aritmética de números, vectores y matrices

Números, vectores y matrices se pueden sumar, restar, multiplicar y dividir de acuerdo con las reglas aprendidas en álgebra lineal. Por lo tanto, el orden de las matrices o vectores debe ser consistente con la operación que se pretende realizar. Ejemplo:

```
>> A=[1 3 5;-9 -8 -7]
```

```
A =
```

```
    1     3     5
   -9    -8    -7
```

```
>> B=5*eye(2)
```

```
B =
```

```
Diagonal Matrix
```

```
     5     0
     0     5
```

```
>> A*B
```

```
error: operator *: nonconformant arguments (op1 is 2x3, op2 is 2x2)
```

$A \times B$ no se puede desarrollar; A es una matriz de 2×3 y B es de 2×2 . Pero $B \times A$ si se puede resolver.

```
>> B*A
```

```
ans =
```

```
     5    15    25
   -45   -40   -35
```

3.4.- Operaciones punto a punto

Continuando con las matrices definidas en el punto anterior, probemos determinar \mathbf{A}^2

```
>> A^2
```

```
error: for x^y, only square matrix arguments are permitted and one  
argument must be scalar. Use .^ for elementwise power.
```

Se presenta el mismo problema anterior, como \mathbf{A} es de 2×3 , al tratar de multiplicar por \mathbf{A} , las dimensiones son inconsistentes. Sin embargo, es posible calcular $\mathbf{A}\mathbf{A}^T$

```
>> A*A'
```

```
ans =
```

```
    35    -68  
   -68   194
```

La multiplicación de una matriz de 2×3 por otra de 3×2 efectivamente debe resultar en una matriz de 2×2 .

¿Cómo podría realizar la operación de elevar cada uno de los elementos de la matriz \mathbf{A} al cuadrado?

```
>> A.^2
```

```
ans =
```

```
     1     9    25  
    81    64    49
```

3.5.- Indexación tácita y lógica

Muchas veces es necesario recuperar valores particulares dentro de una matriz. A los elementos de una matriz o vector se accede escribiendo el nombre de la matriz y, entre paréntesis, los índices indicando el número de fila y columna requeridos. Vamos a definir una matriz y un vector para mostrar algunos ejemplos aclaratorios.

Sea la matriz **C** y el vector **w** los siguientes:

```
>> C=[1989 2021 410 0;89 21 10 -1]
```

```
C =
```

```
    1989    2021    410         0
         89         21         10        -1
```

```
>> w=[pi 40 -96]
```

```
w =
```

```
    3.1416   40.0000  -96.0000
```

Para encontrar elementos individuales dentro de la matriz **C**.

```
>> C(2,2)
```

```
ans = 21
```

```
>> C(4,1)
```

```
error: C(4, _): out of bound 2 (dimensions are 2x4)
```

```
>> C(1,4)
```

```
ans = 0
```

```
>> C(2,1)
```

```
ans = 89
```

Notar que no existe el elemento (4,1), este sería el primer elemento de la cuarta fila.

Para referirse a elementos de un vector, se puede hacer de la misma manera

```
>> w(1,2)
```

```
ans = 40
```

O se puede resumir en un único índice.

```
>> w(2)
```

```
ans = 40
```

Es posible también acceder a toda una fila o columna reemplazando uno de los índices por dos puntos.

```
>> C(:,3)
```

```
ans =
```

```
410
```

```
10
```

```
>> C(2,:)
```

```
ans =
```

```
89    21    10    -1
```

Los elementos dentro de una matriz se enumeran de manera individual organizados por columna, por lo tanto, también es posible llamar o acceder a, por ejemplo, desde el tercer hasta el sexto término de la matriz.

```
>> C(3:6)
```

```
ans =
```

```
2021    21    410    10
```

También es posible acceder a submatrices. Ejemplos:

Todas las filas contenidas en las columnas 1 y 4.

```
>> C(:, [1 4])
```

```
ans =
```

```
1989      0
      89   -1
```

Todas las filas contenidas en las columnas 2 y 4

```
>> C(:, [2 4])
```

```
ans =
```

```
2021      0
      21   -1
```

Todas las filas contenidas *entre* las columnas 2 y 4

```
>> C(:, 2:4)
```

```
ans =
```

```
2021    410      0
      21    10   -1
```

Además de las múltiples opciones utilizando índices tácitos para acceder a elementos dentro de una matriz (o vector). También es posible dar índices de forma lógica. Esta opción puede ser muy útil para filtrar datos.

Elementos de **C** que son menores a cero

```
>> C(C<0)
ans = -1
```

Elementos de **C** que son mayores a cero

```
>> C(C>0)
ans =
    1989
     89
    2021
     21
    410
     10
```

Elementos de **C** iguales a 410

```
>> C(C==410)
ans = 410
```

Elementos de **C** iguales a 400. (ninguno)

```
>> C(C==400)
ans = [](0x1)
```


Posiciones donde los elementos de **C** son menores a 1000. 0 significa falso, 1 significa verdadero.

```
>> C<1000
```

```
ans =
```

```
0  0  1  1
```

```
1  1  1  1
```

3.6.- Ejemplos de aplicación

En esta sección se presentan tres aplicaciones donde OCTAVE es particularmente útil (¡hay muchas más!).

3.6.1 Sistemas de ecuaciones lineales

Uno de los usos más importantes de las matrices es representar y resolver sistemas de ecuaciones lineales. Considere un sistema de ecuaciones lineales de n ecuaciones y n incógnitas:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

donde a_{ij} y b_j son conocidos y se buscan los valores de x_j tal que se cumplan simultáneamente las n igualdades. El mismo problema puede ser expresado en su forma matricial como sigue:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

De esta forma, se puede escribir la siguiente igualdad $Ax=b$, de donde el vector de incógnitas se puede despejar fácilmente como $x = A^{-1}b$.

Ejemplo 3.1. Resolvamos el siguiente sistema,

$$\begin{aligned} x + 2y + z &= 8 \\ 2x - 2y + z &= 9 \\ 3x + 3y - 5z &= 0 \end{aligned}$$

lo expresamos en términos matriciales,

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & -2 & 1 \\ 3 & 3 & -5 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 8 \\ 9 \\ 0 \end{pmatrix}$$

y usamos OCTAVE para encontrar la solución a x , y , z .

```
>> A=[1 2 1;2 -2 1;3 3 -5];
```

```
>> b=[10; -5; 0];
```

```
>> xyz=inv(A)*b
```

```
xyz =
```

```
0.1111
```

```
3.7778
```

```
2.3333
```

Donde el primer elemento del vector columna xyz es x , el segundo es y , y el tercero es z . Usted puede notar lo rápido y sencillo de este cálculo, que de otra manera habría sido bastante tedioso. Imagine lo útil que esta herramienta puede ser cuando se trata de sistemas de muchas ecuaciones.

Sorprenda a sus amigos en las redes sociales resolviendo sus aburridos y molestos desafíos como el de la figura, de una vez por todas, con OCTAVE.

$$\text{🍏} + \text{🍏} + \text{🍏} = 30$$

$$\text{🍏} + \text{🍌} + \text{🍌} = 18$$

$$\text{🍌} - \text{🥥} = 2$$

$$\text{🥥} + \text{🍏} + \text{🍌} = ??$$

Fuente: Diario El País (10)

3.6.2 Raíces de un polinomio

Otra aplicación donde OCTAVE realmente brilla es en su sencillez para resolver ecuaciones polinómicas como:

$$a_0 + a_1x^1 + a_2x^2 \dots + a_n a_1x^n = 0$$

Con la función `roots(pol)` podemos encontrar las raíces o soluciones de esta ecuación. Donde `pol` es un vector que contiene los coeficientes del polinomio en orden decreciente. Veamos un ejemplo.

Ejemplo 3.2. Resolvamos la siguiente ecuación polinómica, adaptada del ejemplo en (11).

$$-5 + x^2 = 0$$

Se necesita un vector con los coeficientes que construyen el polinomio, en este caso: $pol = [a_2, a_1, a_0]$. En OCTAVE podemos hacer lo siguiente:

```
>> pol=[1,0,-5];  
>> roots(pol)  
ans =  
    -2.2361  
     2.2361
```

Esta ecuación es particularmente sencilla de resolver y puede verificar que las soluciones son correctas.

Ejemplo 3.3. Resolvamos ahora una ecuación más complicada:

$$x + x^2 + x^3 = 0$$

En OCTAVE creamos el vector de coeficientes y aplicamos la función `roots`.

```
>> pol=[1,1,1,0];  
>> roots(pol)  
ans =  
  -0.5000 + 0.8660i  
  -0.5000 - 0.8660i  
         0 +      0i
```

Usted puede notar que esta vez la solución no es real, sino que compleja.

Experimente con ecuaciones de segundo grado y verá como la solución con OCTAVE más conveniente que aplicar la clásica fórmula de $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

3.6.3 Raíces de ecuaciones trascendentes

OCTAVE nos permite resolver ecuaciones donde la solución analítica no es posible (este es el caso de muchas aplicaciones en ingeniería). Se entiende por ecuación trascendente aquella donde no es posible despejar la variable independiente. Naturalmente, su solución requiere de aproximaciones numéricas. Con OCTAVE, es posible resolver ecuaciones de este tipo en pocas líneas de comando.

Para abordar esta clase de problemas debemos entender que queremos encontrar un valor de x tal que $f(x) = 0$. Para definir la función en OCTAVE usamos un `@(variable)` en frente de la expresión a estudiar para poder trabajar de forma simbólica con el ingreso de la expresión. El comando `fzero()` predefinido en OCTAVE (12) necesita de la función $f(x)$ y de un valor inicial cercano a la solución que se busca. Es buena idea graficar $f(x)$ para estimar donde están las raíces.

Veamos dos ejemplos a continuación. Usted puede estudiar la validez de esas soluciones reemplazando el valor de `solucion` en lugar de x en las ecuaciones y verificar que la igualdad se cumple.

Ejemplo 3.4. Buscar una solución en la vecindad de 1 para la siguiente ecuación.

$$xe^x - 1 = 0$$

```
funcion = @(x) x*exp(x)-1;  
solucion = fzero(funcion,1)
```

Resultado
`solucion = 0.5671`

Ejemplo 3.5. Buscar una solución en la vecindad de 3 para la siguiente ecuación.

$$x - \tan(x) = 0$$

```
funcion = @(x) x-tan(x);  
solucion = fzero(funcion,3)
```

Resultado
`solucion = 4.4934`

Reflexione un momento. Estas ecuaciones no las habría podido resolver con ninguna técnica analítica. La única alternativa es acercarse a la solución desde las técnicas numéricas. OCTAVE ahora le permite resolver cualquier ecuación trascendente a la que se pueda enfrentar.

Capítulo 4

Gráficos sencillos con GNU OCTAVE

A pesar de que esta guía tiene por objetivo entregar herramientas para resolver ecuaciones diferenciales ordinarias, aprender a producir gráficos es una excelente manera de familiarizarse con el uso de vectores. Además, ser capaces de presentar buenos gráficos, claros y de calidad, es una habilidad que podrán usar toda su vida académica y profesional.

OCTAVE tiene una librería variada de tipos de gráficos. En esta guía sólo discutiremos la función `plot()` que tiene la siguiente estructura de ingreso de entradas:

```
plot(x,y,'especificaciones_de_línea')
```

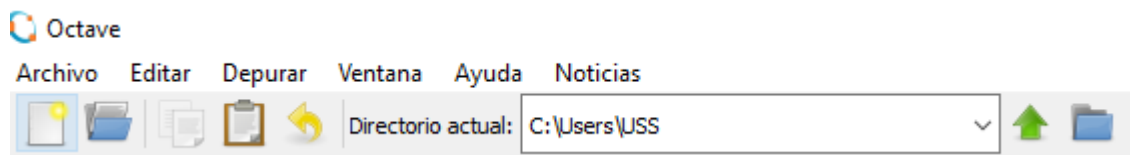
Puede encontrar una completa descripción de las opciones de la función `plot()` en (13). La web de MATLAB también tiene disponibles excelentes ejemplos (14).

En esta sección hay ejemplos de gráficos sencillos a modo de ejercicio para entender la operatoria con vectores en OCTAVE.

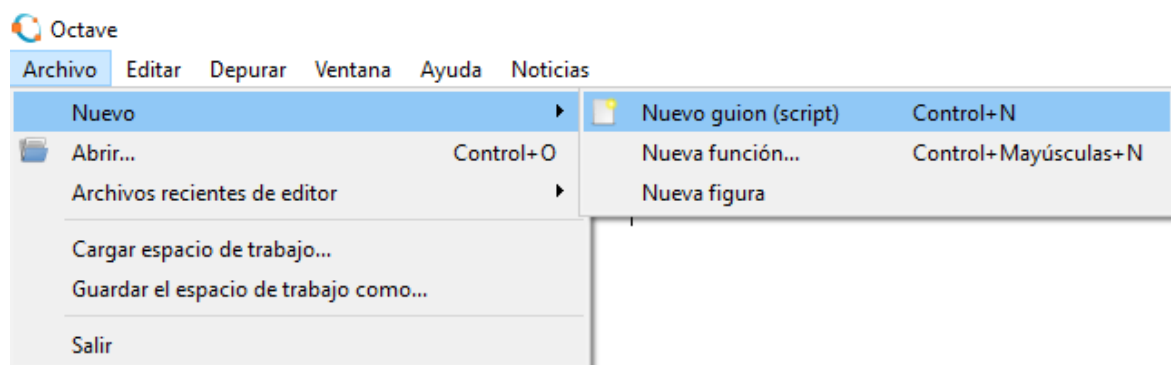
4.1.- Entorno script

Hasta el capítulo anterior hemos estado usando la ventana de comandos para ingresar las instrucciones y comandos que queremos ejecutar en OCTAVE. Si queremos desarrollar una rutina de varios pasos que podamos usar tantas veces como queramos se hace más conveniente trabajar en un Script (también llamado Editor).

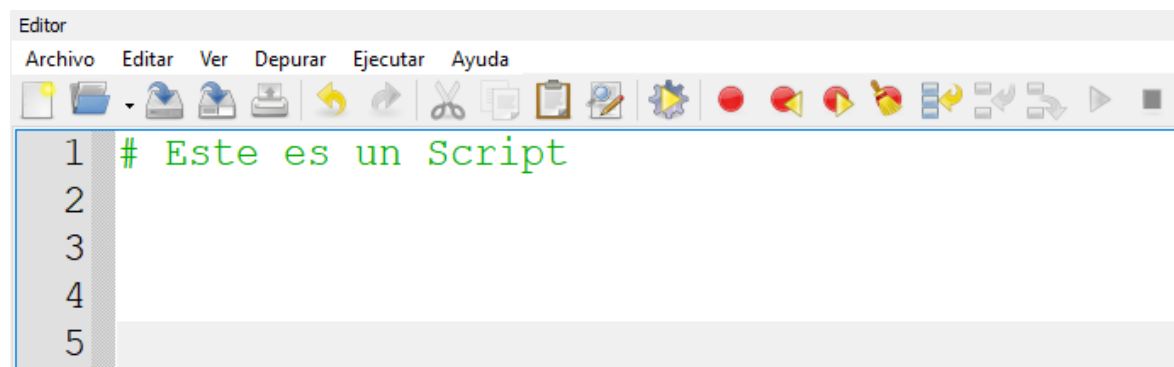
Para acceder al Editor podemos hacer clic en el ícono de hoja bajo el menú archivo destacado en azul claro en esta figura.



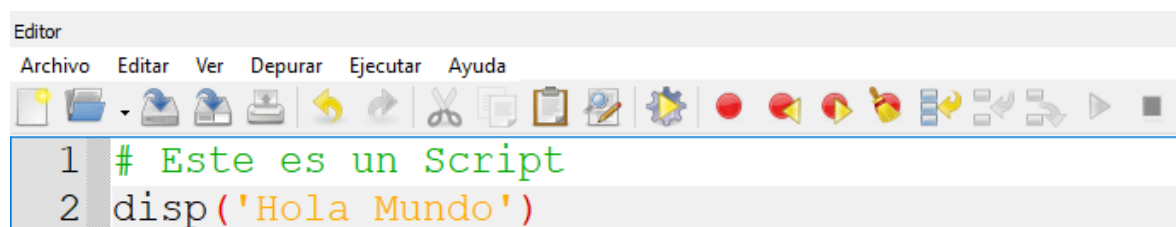
También puede hacer clic en el menú archivo, bajar a nuevo y hacer clic en Nuevo guion (script). Otra manera de acceder es simplemente tecleando **Ctrl+N**.



Al acceder al Script o Editor verá una hoja en blanco con las líneas numeradas a la izquierda. Sólo verá la primera línea numerada en un comienzo, pero si presiona enter varias veces verá como las demás líneas también tendrán un numero a la izquierda.



Por ejemplo, en su script escriba lo siguiente:

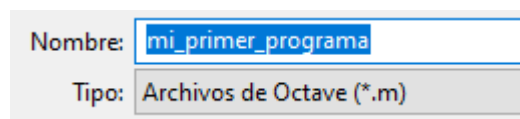


```

1 # Este es un Script
2 disp('Hola Mundo')

```

Guarde el archivo en una ubicación conveniente en el menú archivo y luego guardar archivo como



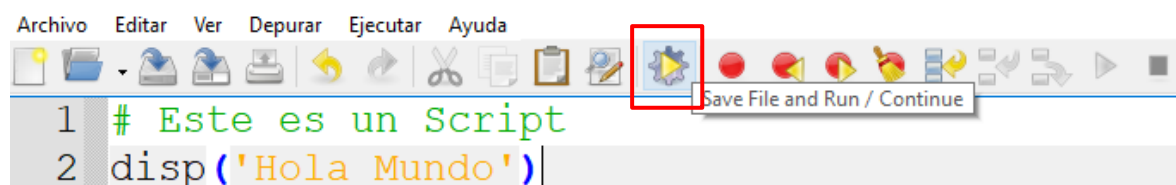
Nombre:

Tipo:

Por defecto seleccionará el tipo de archivo como .m, esta es la extensión de los archivos creados con OCTAVE y MATLAB.

Recomendación para los nombres de archivos: no utilice espacios, no utilice mayúsculas, no comience el nombre del archivo con un número, no use palabras clave como if, for, plot, o una función predefinida.

Ahora que el archivo esta guardado, puede presionar el icono de la tuerca con un play amarillo (Save file and Run/Continue).

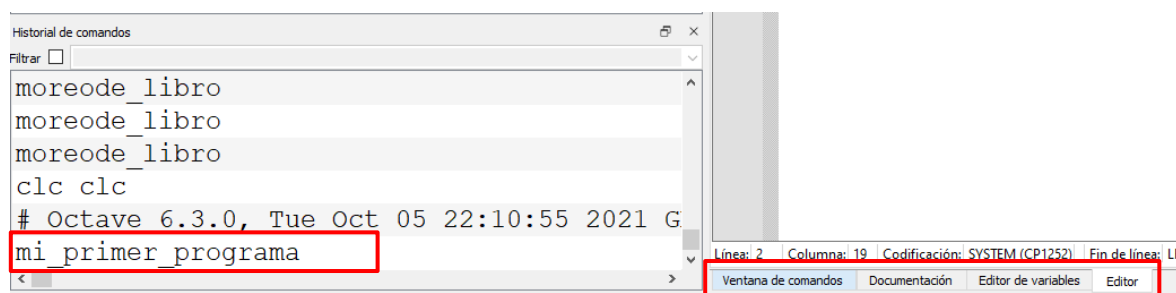


```

1 # Este es un Script
2 disp('Hola Mundo')

```

Verá que en el *historial de comandos* la última entrada dice mi_primer_programa.



```

Historial de comandos
Filtrar
moreode_libro
moreode_libro
moreode_libro
clc clc
# Octave 6.3.0, Tue Oct 05 22:10:55 2021 G
mi_primer_programa

```

Línea: 2 Columna: 19 Codificación: SYSTEM (CP1252) Fin de línea: LF

Ventana de comandos Documentación Editor de variables Editor

Ahora vuelva a la *ventana de comandos* (ver pestañas en la parte de debajo de la pantalla). Verá que se despliega lo siguiente:

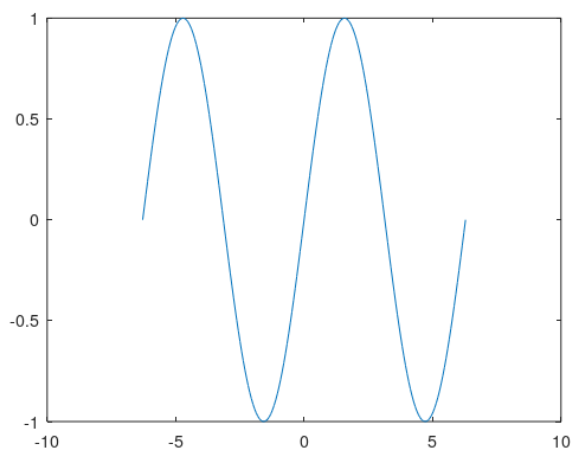
```
>> mi_primer_programa
```

```
Hola Mundo
```

4.2.- Gráficos a partir de vectores

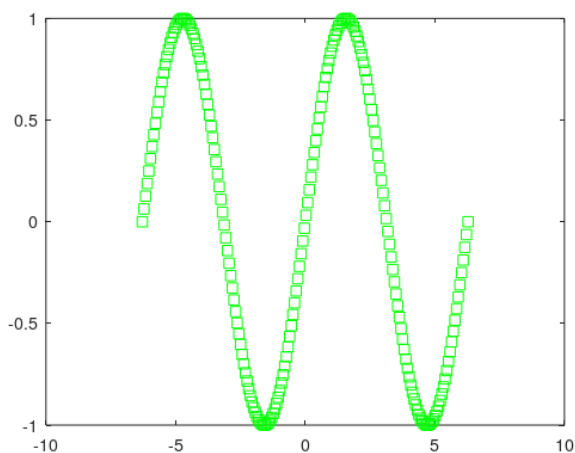
Para las próximas actividades de esta guía se recomienda usar el Editor (Script) en vez de la ventana de comandos. Comencemos usando la función predefinida `plot` sin opciones para graficar la función seno entre -2π y 2π .

```
clc, clear all, close all
#definimos el dominio donde
#queremos graficar
x=linspace(-2*pi,2*pi,200);
#definimos la función
y=cos(x-pi/2);
#usamos la función plot
plot(x,y)
```



Si repetimos la rutina, esta vez agregando al final del plot `'gs'`, esto es, color *green* y marcador *square*, se obtiene lo siguiente.

```
clc, clear all, close all
x=linspace(-2*pi,2*pi,200);
y=cos(x-pi/2);
plot(x,y, 'gs')
```

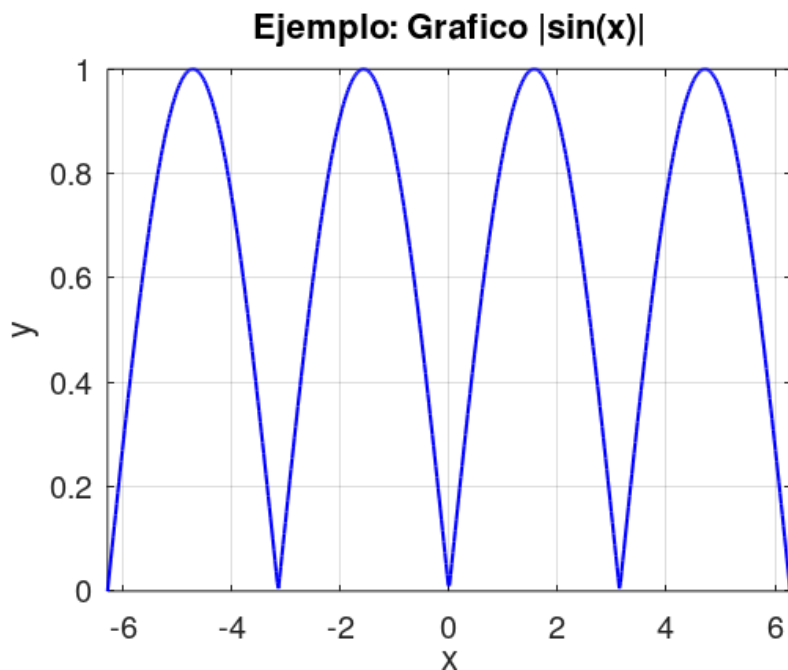


Por su puesto, a este gráfico se le pueden mejorar muchas cosas.

Grafiquemos ahora la función $f(x) = |\sin(x)|$ agregando algunas opciones para mejorar la presentación del gráfico como grosor y color de línea (linewidth y color) y el tamaño de fuente.

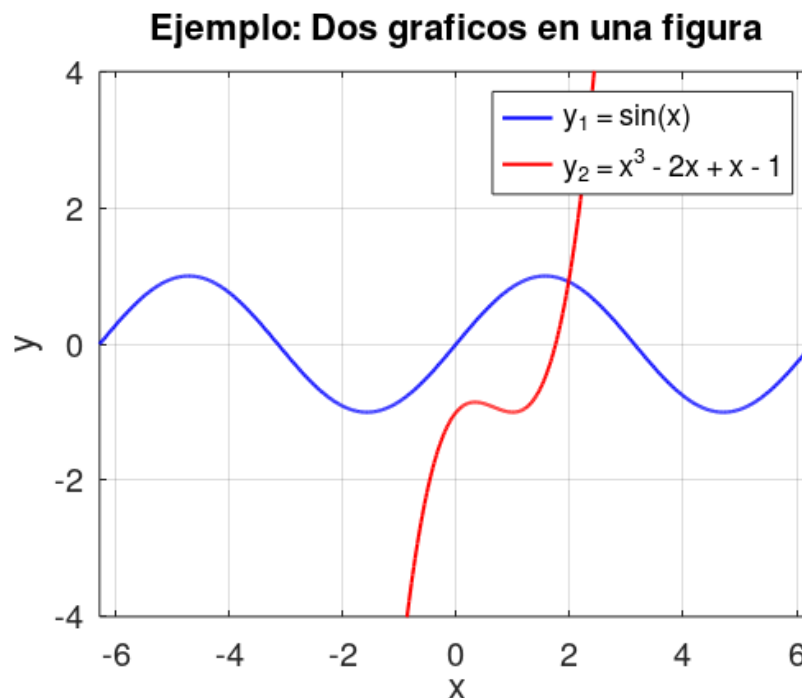
```
clc, clear all, close all

#definimos el dominio donde queremos graficar
x=linspace(-2*pi,2*pi,600);
#definimos la función
y=abs(sin(x));
#usamos la función plot
plot(x,y,'linewidth',1.5,'color','b')
grid on #encendemos la grilla
#definimos los límites de x en el gráfico
xlim([-2*pi 2*pi])
#agregamos etiquetas para los ejes
xlabel('x'),ylabel('y')
#agregamos un título
title('Ejemplo: Grafico |sin(x)|')
#modificamos el tamaño de fuente
set(gca,'fontsize',14)
```



Es habitual graficar más de una curva en una misma figura. Cuando hacemos esto, además se hace necesario definir una leyenda. Veamos un ejemplo.

```
clc, clear all, close all
x=linspace(-2*pi,2*pi,600);
#definimos dos funciones
#Note que aquí hace falta la operación con punto.
y1=sin(x);
y2=x.^3-2*x.^2+x-1;
#usamos el comando plot para graficar cada x,y con sus especificaciones
plot(x,y1,'linewidth',1.5,'color','b',x,y2,'linewidth',1.5,'color','r')
grid on
#definimos limites del grafico para x e y, y agregamos etiquetas
xlim([-2*pi 2*pi])
ylim([-4 4])
xlabel('x'),ylabel('y')
#agregamos una leyenda, un título y el tamaño de fuente
legend('y_{1} = sin(x)', 'y_{2} = x^3 - 2x^2 + x - 1')
title('Ejemplo: Dos graficos en una figura')
set(gca,'fontsize',14)
```



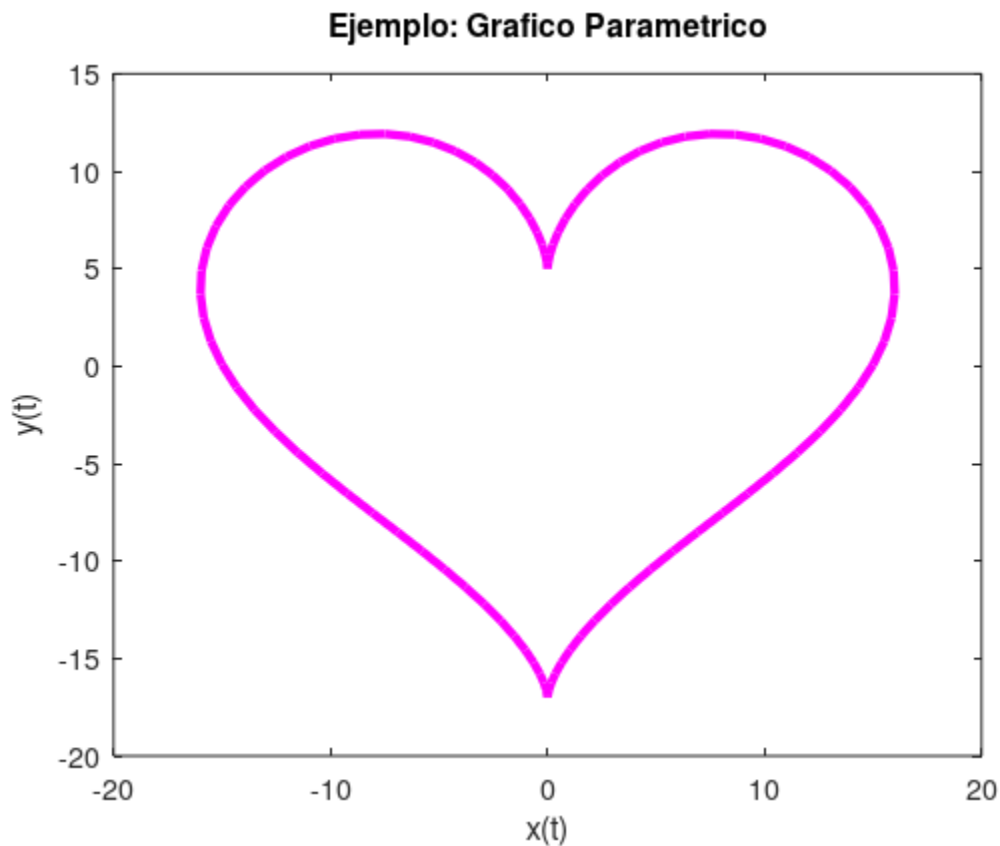
Otra manera de agregar varias curvas en un mismo grafico es usar la opción `hold on`. Veamos un ejemplo. Además, en este ejemplo, cambiaremos el estilo de línea y la ubicación de la leyenda.

```
clc, clear all, close all
x=linspace(-2*pi,2*pi,600);
y1=sin(x);
y2=x.^3-2*x.^2+x-1;
#graficamos la primera curva
#notar cambio en linestyle
plot(x,y1,'linewidth',1.5,'color','b','linestyle','-')
#bloqueamos la figura
hold on
#agregamos la siguiente curva
#notar cambio en linestyle
plot(x,y2,'linewidth',1.5,'color','r','linestyle '--')
grid on
xlim([-2*pi 2*pi])
ylim([-4 4])
xlabel('x'),ylabel('y')
#notar que ahora leyenda tiene una opción llamada location
legend('y_{1} = sin(x)', 'y_{2} = x^3 - 2x^2 + x - 1', 'location', 'northwest')
title('Ejemplo: Dos graficos en una figura con hold on')
set(gca, 'fontsize', 14)
```



Por último, graficaremos una función paramétrica llamada para “*The Love Graph*” que se la pueden enviar a su interés romántico. Que no se diga que los ingenieros(as) no somos cariñosos o cariñosas. En esta función, definimos un parámetro t de 0 a 2π y dos expresiones que definen las coordenadas de los puntos (x,y) que finalmente graficaremos. Note que aquí hace falta la operación con punto. Las ecuaciones para producir este grafico fueron tomadas de (15).

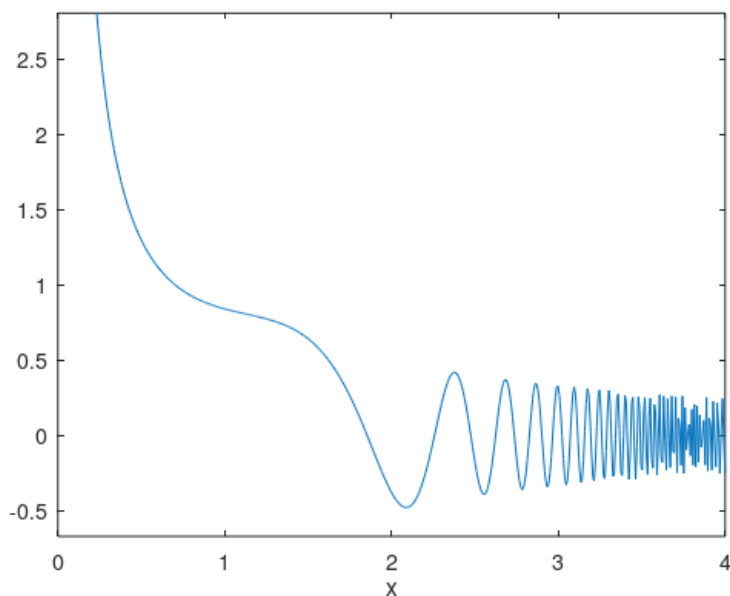
```
clc, clear all, close all
#parámetro independiente
t=linspace(0,2*pi,100);
#expresiones para x e y
x=16*(sin(t)).^3;
y=13*cos(t)-5*cos(2*t)-2*cos(3*t)-cos(4*t);
#graficamos con plot, para un grosor de línea 3 y color magenta.
plot(x,y,'linewidth',3,'m')
```



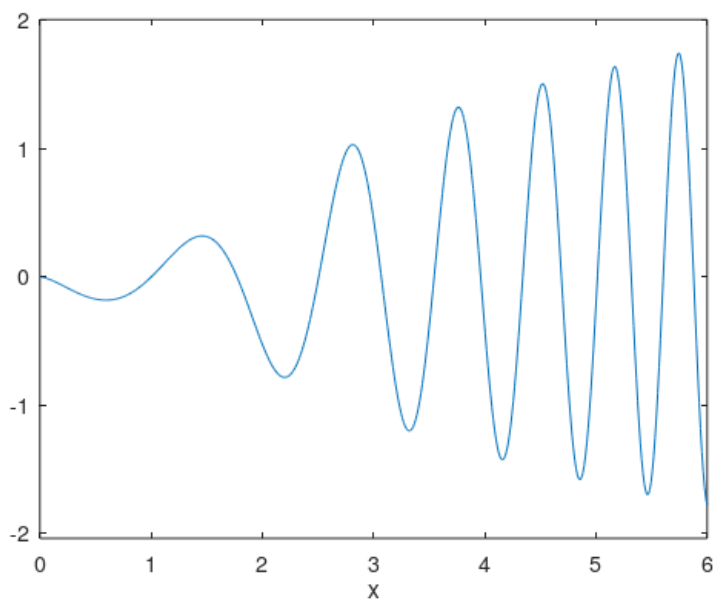
4.3.- Gráfico directo de funciones

OCTAVE además cuenta con un comando muy útil para graficar rápidamente, `ezplot('fun',[xmin xmax])(16)`. Donde ingresamos la expresión matemática entre apostrofes y definimos el intervalo para el eje x . Veamos dos ejemplos que los puede ejecutar directo en la ventana de comandos o en un script, como prefiera.

```
>> ezplot('sin(x^x)/x',[0,4])  
sin(x^x)/x
```



```
>> ezplot('log(x)*sin(x^2)',[0,6])  
log(x) sin(x^2)
```



Capítulo 5

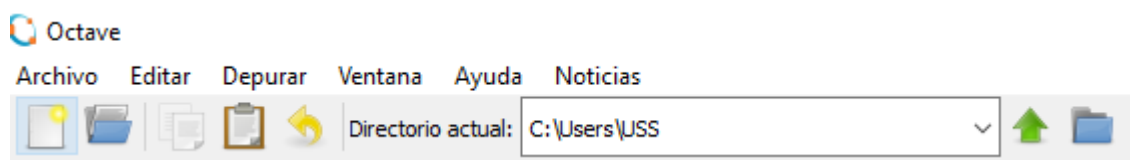
Introducción a la programación en GNU OCTAVE

Los programas que desarrollemos en OCTAVE los vamos a escribir en el entorno Script (o Editor). Por conveniencia se introduce este entorno en este capítulo nuevamente. Luego se introduce el concepto de funciones y se presentan el control if y el ciclo for con ejemplos.

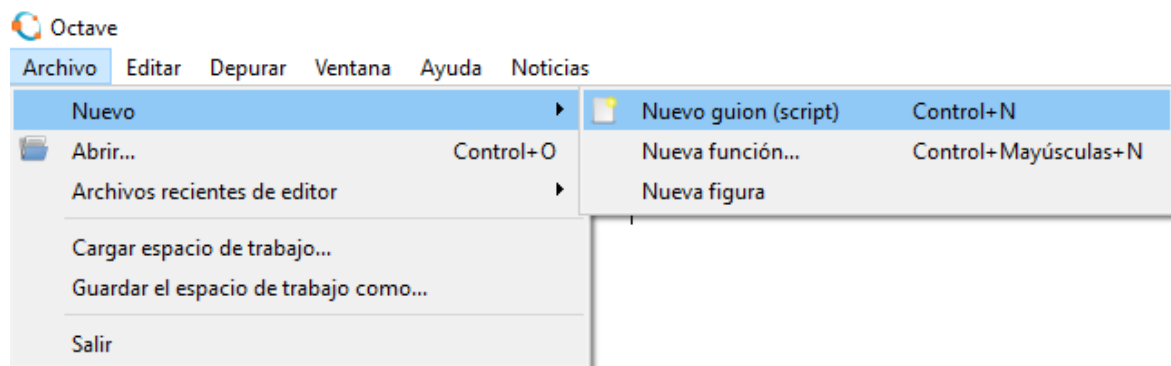
5.1.- Entorno script

Si queremos desarrollar una rutina de varios pasos que podamos usar tantas veces como queramos se hace conveniente trabajar en un Script (también llamado Editor).

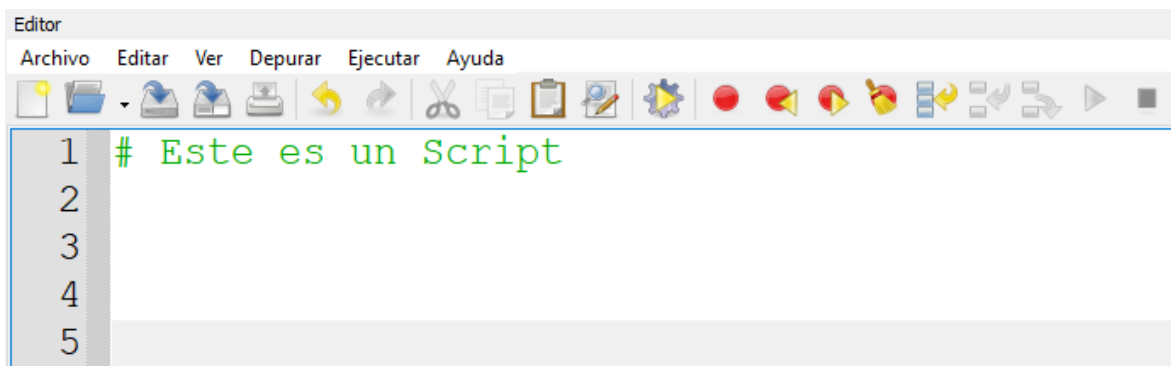
Para acceder al Editor podemos hacer clic en el ícono de hoja bajo el menú archivo destacado en azul claro en esta figura.



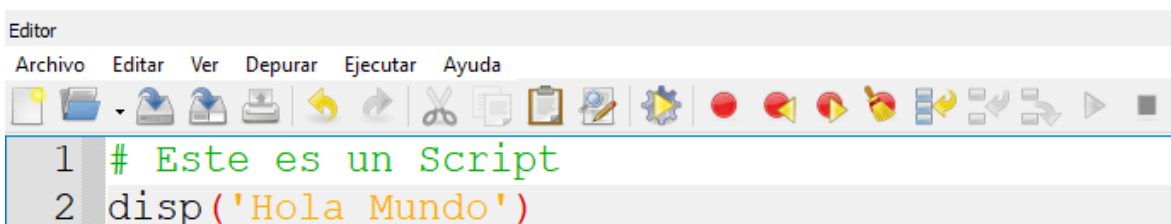
También puede hacer clic en el menú archivo, bajar a nuevo y hacer clic en Nuevo guion (script). Otra manera de acceder es simplemente tecleando `Ctrl+N`.



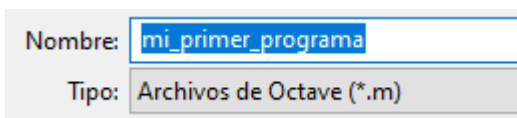
Al acceder al Scrip o Editor verá una hoja en blanco con las líneas numeradas a la izquierda. Sólo verá la primera línea numerada en un comienzo, pero si presiona enter varias veces verá como las demás líneas también tendrán un numero a la izquierda.



Por ejemplo, en su script escriba lo siguiente:



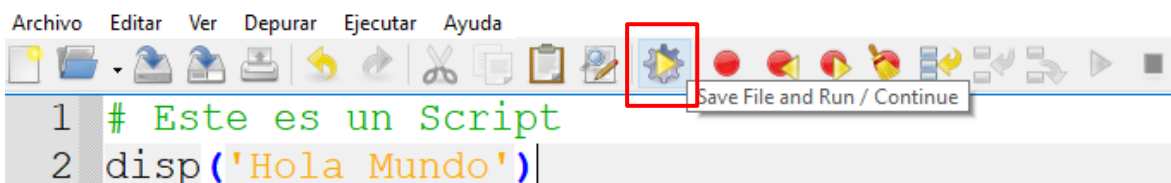
Guarde el archivo en una ubicación conveniente en el menú archivo y luego guardar archivo como



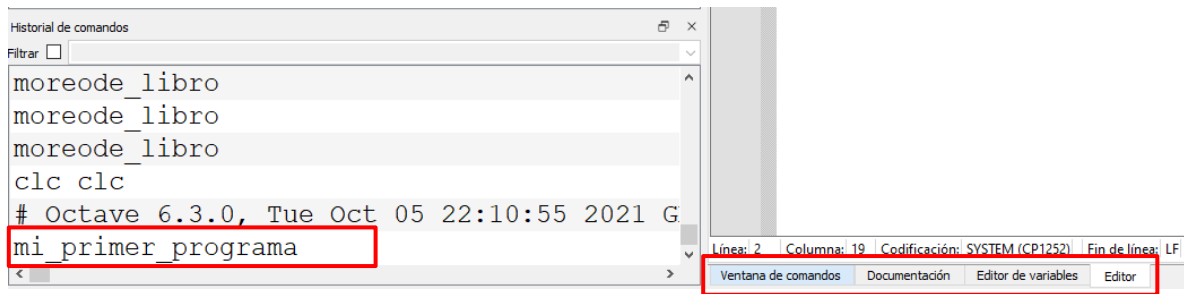
Por defecto seleccionará el tipo de archivo como .m, esta es la extensión de los archivos creados con OCTAVE y MATLAB.

Recomendación para los nombres de archivos: no utilice espacios, no utilice mayúsculas, no comience el nombre del archivo con un número, no use palabras clave como *if*, *for*, *plot*, o una función predefinida.

Ahora que el archivo esta guardado, puede presionar el icono de la tuerca con un play amarillo (Save file and Run/Continue).



Verá que en el *historial de comandos* la última entrada dice `mi_primer_programa`.



Ahora vuelva a la *ventana de comandos* (ver pestañas en la parte de debajo de la pantalla). Verá que se despliega lo siguiente:

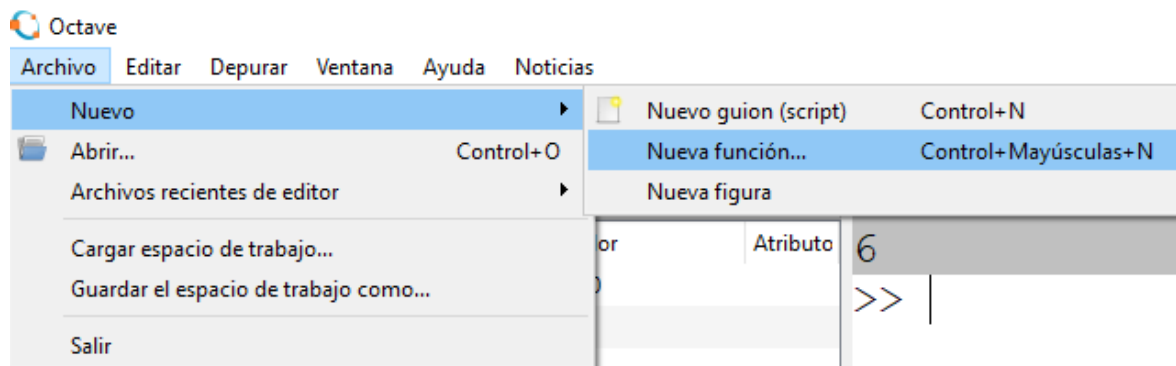
```
>> mi_primer_programa
```

```
Hola Mundo
```

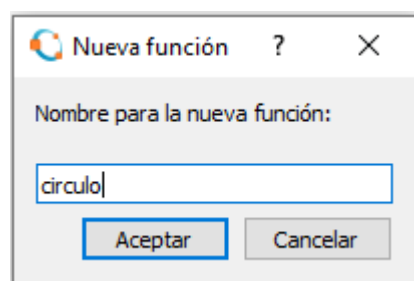
5.2.- Funciones

Si ha llegado hasta esta sección, usted ya ha utilizado varias funciones predefinidas en OCTAVE, vea por ejemplo las subsecciones 2.3 y 3.2. Usted como desarrollador puede crear sus propias funciones. Las funciones permiten ahorrar líneas de código en una rutina compleja. Puede encontrar una completa descripción de este tema en la referencia (17). Por ejemplo, creemos una función que calcule el área de un círculo a partir del radio.

Primero abriremos una nueva función



En la ventana emergente agregue un nombre para su función, en este caso usaremos el nombre círculo



Inmediatamente se creará un archivo llamado `circulo.m` en el directorio de trabajo. La función DEBE tener el mismo nombre que el archivo, no lo modifique. Al mismo tiempo se abrirá un editor con texto en verde (comentario) con un mensaje informativo. Al final, verá lo siguiente:

```
function retval = circulo (input1, input2)
```

```
endfunction
```

`retval` es la salida de nuestra función, puede ser un vector.

`input1` e `input 2` son las entradas, pueden ser vectores, y el número de entradas por defecto es dos, pero pueden ser menos o más según requiera la aplicación.

Escriba lo siguiente y guarde en el menú archivo y guardar archivo (sin ejecutar)

```
function A = circulo (radio)
A=pi*radio^2
endfunction
```

Ahora en la ventana de comandos puede ejecutar lo siguiente

```
>> circulo(2)
ans = 12.566
```

Este resultado sería el área de un círculo de radio 2. Puede usar su nueva función para asignar el resultado a alguna variable tal y como hicimos antes con funciones predefinidas. Por su puesto también puede usar su nueva función dentro de otros Scripts. La única condición es que el script que llama a la función circulo debe estar en el mismo directorio de trabajo donde esta almacenada la función.

Hagamos una modificación a la función de forma que nos entregue dos salidas, A y P .

```
function [A,P] = circulo (radio)
A=pi*radio^2;
P=2*pi*radio;
endfunction
```

Vuelva a guardar la función (sólo guardar, no guardar como). Luego trate lo siguiente en la ventana de comandos:

```
>> r=4;
>> [area,perimetro]=circulo(r);
>> area
area = 50.265
>> perimetro
perimetro = 25.133
```

5.3.- Control if

Para el control “SI” o If se definen varias condiciones y asociadas a estas una serie de acciones, se ejecuta la acción asociada a la condición verdadera. Si ninguna de las condiciones se cumple, entonces se ejecuta la acción asociada a **else** (“SI NO”).

La estructura o sintaxis a seguir es:

```
if <condición>
    <acciones>
else
    <acciones alternativas>
end
```

Para un caso más complejo con varias condiciones posibles:

```
if <condición_1>
    <acciones_1>
elseif <condición_2>
    <acciones_2>
.
.
.
elseif <condición_n>
    <acciones_n>
else
    <acción>
end
```

Veamos un ejemplo del uso del control if.

Ejemplo 5.1. Cree el siguiente algoritmo en un script, guárdelo y ejecútelo.

```
clc,clear all
a=6.87
if a>0
    disp('El numero es positivo')
elseif a<0
    disp('El numero es negativo')
else
    disp('El numero es cero')
end
```

En la ventana de comandos se debió desplegar lo siguiente

```
a = 6.8700
```

```
El numero es positivo
```

Si reemplaza `a` por otro y vuelve a ejecutar verá como este sencillo algoritmo clasifica según el signo de la entrada. Si gusta, puede hacer que la rutina sea interactiva con el usuario. Reemplace `a` por la siguiente instrucción:

```
a=input('Ingrese un valor para clasificar: ');
```

Ejecute el script y vaya a ventana de comandos. Ahora OCTAVE está a la espera del valor que ingrese el usuario

```
Ingrese un valor para clasificar:
```

```
Ingrese un valor y presione enter.
```

```
Ingrese un valor para clasificar: -99
```

```
El numero es negativo
```

Puede encontrar lectura adicional sobre el control `if` en (18).

5.3.1 Operadores booleanos

Para escribir las condiciones se deben usar operadores booleanos o lógicos. Estos son:

Símbolo	Significado	Ejemplo
==	igual	if a==5
~=	diferente de	if x~=a
>	mayor que	elseif i>100
>=	mayor o igual que	if a>=b
<	menor que	if tolerancia<0.0001
<=	menor o igual que	elseif c<=hf

Además, puede combinar condiciones con los operadores “y” y “o”.

Símbolo	Significado	Ejemplo
&	y	if (i>10) & (j<10)
	o	if tolerance<=0.001 i==100

Puede encontrar lectura adicional sobre operadores booleanos en (19).

5.4.- Ciclo for

En el ciclo `for`, también llamado `loop`, se define una serie de acciones que se repiten un número finito de veces. La sintaxis es:

```
for contador = vector
    <acciones>
end
```

Veamos dos ejemplos.

Ejemplo 5.2. Para mostrar 3 veces en pantalla el mismo mensaje podemos utilizar el siguiente script con el ciclo `for`.

```
clc,clear all
for i=1:3
    disp('OCTAVE rules')
end
```

Ejemplo 5.3. Sumar los primeros 100 números naturales (Como el pequeño niño Gauss).

```
clc,clear all
#definamos un valor N igual al numero de iteraciones a realizar
N=100;
#definamos una variable llamada S que vale cero inicialmente
S=0;
#desde i=1 hasta i=N realizar la acción de sumar i al valor S
for i=1:N
    S=S+i;
End
#mostrar resultados en la ventana de comandos
disp('La suma desde 1 hasta N es: ')
disp(S)
```

Revisemos que hace esta rutina paso por paso,

Cuando $i = 1$, $S = S + i$, es decir, $S = 0 + 1 = 1$

Cuando $i = 2$, $S = S + i$, es decir, $S = 1 + 2 = 3$

Cuando $i = 3$, $S = S + i$, es decir, $S = 3 + 3 = 6$

...

Cuando $i = N = 100$, $S = S + i$, es decir, $S = S_{i-1} + 100 = 5050$

Puede encontrar lectura adicional sobre el control `if` en (20).

5.5.- Otros controles disponibles

A decir verdad, con el control `if` y el ciclo `for` puede resolver cualquier problema que se le presente. Sin embargo, existen algunos otros controles que pueden ser de conveniencia o utilidad en cientos casos. Si es de su interés fortalecer sus herramientas de programación, le recomiendo buscar información sobre los controles `switch`, `select case` y `while` (21-24).

5.6.- Aplicación algoritmo babilónico

Una excelente forma de aproximar el valor de raíces cuadradas es el algoritmo babilónico. El algoritmo babilónico para raíces cuadradas es un método ancestral para aproximar la raíz cuadrada de un número dado a través de una secuencia de cálculos iterativos.

Sea a un valor real positivo, la secuencia x_n definida como sigue converge a \sqrt{a} .

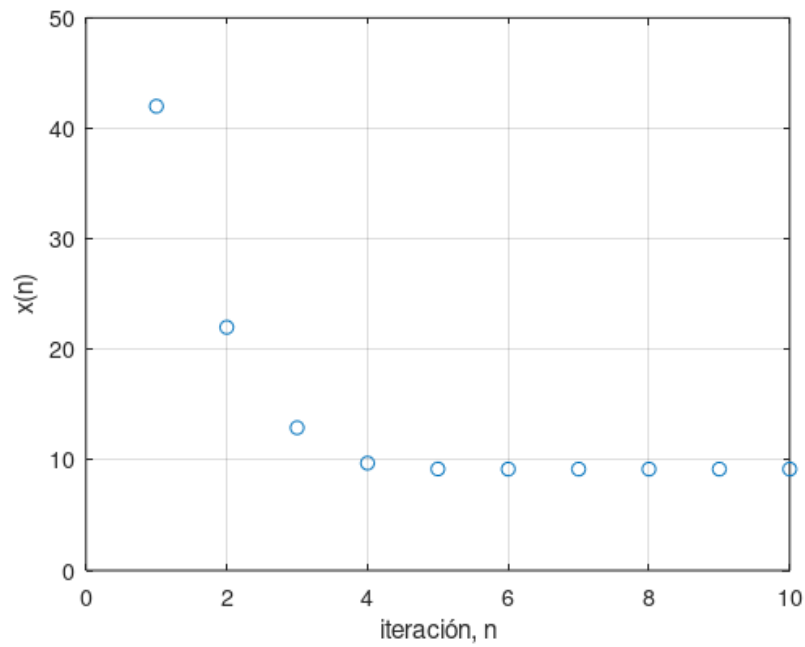
$$x_n = \frac{1}{2} \left(\frac{x_{n-1}}{a} + x_{n-1} \right)$$

$$\lim_{n \rightarrow \infty} x_{n+1} = \sqrt{a}$$

Notar que es necesario un primer valor de x_n para comenzar con la iteración. Digamos que el primer x_1 será $a/2$. Programemos este algoritmo en OCTAVE.

```
#definimos el valor al cual queremos calcular la raíz cuadrada
a=84
#definimos el número de iteraciones a realizar
iter=10
#definimos el primer valor de la secuencia
x(1)=a/2
#desde n=2 hasta n=10 cada valor de x_n se define según la expresión
for n=2:iter
    x(n)=(1/2)*(x(n-1)+a/x(n-1));
end
#graficamos el vector de todos los x. Como no se especifica un eje
horizontal
#en el comando plot, OCTAVE graficará x vs. La posición de cada valor de
x.
plot(x,'o')
#agregamos algunas opciones para el gráfico
xlabel('iteración, n')
ylabel('x(n)')
grid on
```


El gráfico de los valores de $x(n)$ de cada iteración (n) muestra como la secuencia converge a un valor luego de la quinta iteración. El valor de convergencia es $x(10) = 9.1652$ y $\sqrt{84} = 9.1652$. Podemos decir que el algoritmo funciona.



El algoritmo se puede hacer sin la necesidad de guardar cada iteración. Simplemente, el valor de x se sobrescribe en cada iteración. Pruebe el siguiente script que además incluye un control `if` para aceptar sólo valores positivos de a .

```
clc, clear all
#con la función input solicitamos el ingreso del valor a en
#la ventana de comandos
a=input('Raiz cuadrada de: ');
#mantenemos el número de iteraciones en 10
iter=10;
#el valor inicial de la secuencia lo definimos como a/2
x=a/2;
#aplicamos un control if para aceptar solo valores positivos de a
if a==0 #a no debe ser nulo, imponemos la restricción
    disp('Raiz cuadrada de cero es cero')
elseif a<0 #a no debe ser negativo, imponemos la restricción
    disp('Raiz cuadrada de un negativo no esta definida en los Reales')
else #la única condición restante es que a sea positivo
    for i=1:iter
        #note que el valor de x se sobrescribe en cada iteración
        x=(1/2)*(x+a/x);
    end
    #mostramos el último valor de x en pantalla con un mensaje
    disp('La Raiz cuadrada es: ')
    disp(x)
end
```



Para ejecutar este script puede guardarlo y presionar el botón  o simplemente escribir el nombre de la rutina en la ventana de comandos. Si la rutina esta guardada en el directorio actual como `babilonicos.m`, puede ejecutar lo siguiente en la ventana de comandos:

```
>> babilónicos
```

Se desplegará en la pantalla el mensaje “Raiz cuadrada de:” Ingrese un valor, por ejemplo 67.

```
Raiz cuadrada de: 67
```

```
La Raiz cuadrada es:
```

```
8.1854
```

Capítulo 6

Aplicaciones a ecuaciones diferenciales ordinarias

Felicitaciones por haber llegado hasta el último capítulo de este documento. Finalmente, luego de una larga presentación del software OCTAVE, es momento de resolver ecuaciones diferenciales.

6.1.- Conceptos fundamentales

Una ecuación diferencial es aquella que relaciona funciones y sus derivadas. Estas ecuaciones tienen una multitud de aplicaciones en física, biología, economía, ingeniería y un largo etcétera. Algunas ecuaciones diferenciales que usted ya conoce y ha usado probablemente sin darse cuenta son la ecuación de la segunda ley de Newton y las ecuaciones de cinemática. Existen dos tipos de ecuaciones diferenciales: (1) ecuaciones diferenciales ordinarias y (2) ecuaciones diferenciales parciales. El tema de esta guía son el primer grupo, ecuaciones diferenciales ordinarias, que quiere decir que sólo hay derivadas respecto de una variable independiente, a diferencia de las ecuaciones diferenciales parciales, donde se trabaja con derivadas de múltiples variables independientes.

Considere que existe una función x que depende de t , $x(t)$, y que tiene una derivada que se puede expresar como una función de x y de t , $f(x, t)$. Es decir:

$$\frac{dx}{dt} = f(x, t)$$

Además, se conoce el valor inicial del problema (estado inicial del sistema), es decir, la función en un tiempo inicial t_0 tiene un valor conocido x_0 .

$$x(t_0) = x_0$$

Resolveremos esta clase de problemas usando el comando `lsode()` de OCTAVE.

6.1.1 Commando `lsode()`

De acuerdo con la descripción en la web de OCTAVE (25), la función o commando `lsode()` puede ser usada para resolver ecuaciones diferenciales ordinarias de la forma:

$$\frac{dx}{dt} = f(x, t)$$

usando el algoritmo de Hindmarsh. La estructura de salidas y entradas es la siguiente:

```
[x, istate, msg] = lsode (fcn, x_0, t)
```

Donde,

`fcn` es la función $f(x, t)$ expresada como una función simbólica o anónima (como en la sección 3.6.3),

`x_0` es el valor inicial o estado inicial del sistema

`t` es el vector que define el rango del dominio de la función donde se quiere la solución. El primer elemento de `t` debe ser `t_0` y debe corresponder al estado inicial del sistema `x_0`. Por lo tanto, el primer valor de la solución `x` debe coincidir con `x_0`.

`x` es la solución numérica a la ecuación diferencial

`[istate, msg]` son mensajes opcionales que entrega el algoritmo al finalizar. Se pueden usar para verificar que la ejecución del algoritmo ha sido exitosa.

En esta sección veremos que una ecuación diferencial de orden n puede convertirse en n ecuaciones acopladas de primer orden y ser resuelta mediante `lsode()`.

6.2.- Ejemplos resueltos

En esta sección se describe la solución numérica, paso a paso, de cuatro ecuaciones diferenciales ordinarias (EDO) usando `lsode()` de OCTAVE.

6.2.1 Ejemplo EDO con raíz, potencia, logaritmo

Considere la siguiente ecuación diferencial en base a la función $x(t)$ y su primera derivada dx/dt . La idea es encontrar la función $x(t)$ en el rango $t \in (1,2)$, conociendo el valor inicial de la función como se indica a continuación.

$$\frac{dx}{dt} = 5\sqrt{t} - x^3 \ln(t)$$

$$x(1) = 1.5$$

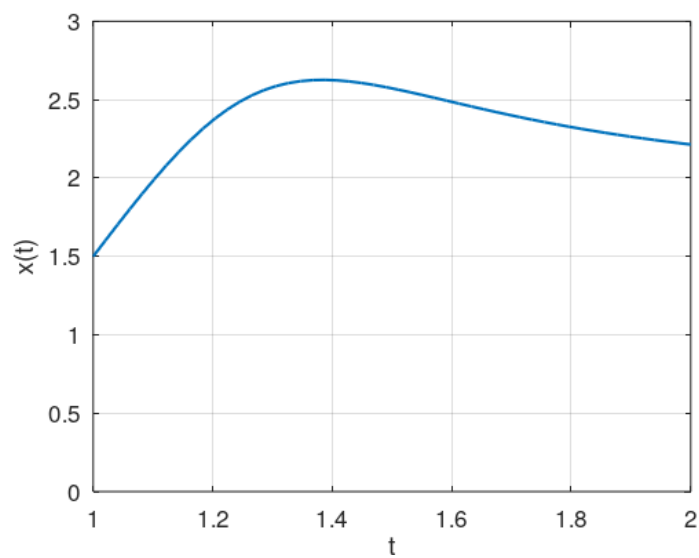
```
clc, clear all, close all
#definimos el rango para la variable independiente t
t_0 = 1;
t_max = 2;
t=linspace(t_0,t_max,50);
#definimos el valor inicial de la función x en t_0
x_0=1.5;
#definimos la función f(x,t) a la derecha de la igualdad dx/dt = f(x,t)
fun = @(x,t) 5*t^0.5-log(t)*x^3;
#utilizamos la función predefinida en OCTAVE lsode
[x,istate,msg]=lsode(fun,x_0,t);
istate
msg
#graficamos la función x(t)
plot(t,x,'linewidth',1.5),grid on,xlabel('t'),ylabel('x(t)')
#agregamos limites al eje y para una visualización clara
ylim([0 3])
#modificamos el tamaño de fuente del grafico
set(gca,'fontsize', 12)
```

En la *ventana de comandos* se despliega la siguiente información indicando que no hubo dificultades con la implementación del algoritmo de solución.

```
istate = 2
```

```
msg = successful exit
```

El gráfico siguiente muestra la solución a la ecuación diferencial en el rango $t \in (1,2)$. Se verifica que el valor inicial es $x=1.5$ cuando $t=1$.



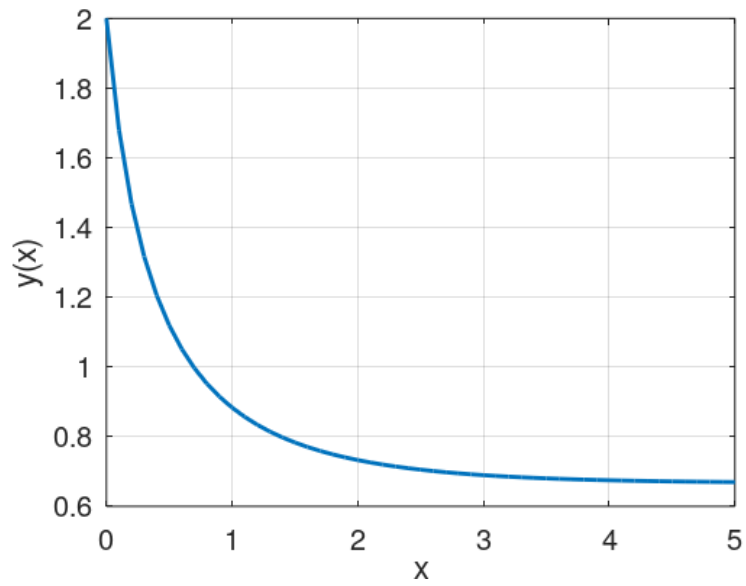
6.2.2 Ejemplo EDO con función exponencial

Considere la siguiente ecuación diferencial en base a la función $y(x)$ y su primera derivada dy/dx . La idea es encontrar la función $y(x)$ en el rango $x \in (0,5)$, conociendo el valor inicial de la función como se indica a continuación.

$$\frac{dy}{dx} = -y^2 e^{-x}$$
$$y(0) = 2$$

```
clc, clear all, close all
#definimos el rango para la variable independiente x.
x=[0:0.1:5];
#definimos el valor inicial de la función y
y0=2;
#definimos la función f(y,x) a la derecha de la igualdad dy/dx = f(y,x)
fun = @(y,x) -exp(-x)*y^2;
#utilizamos la función predefinida en OCTAVE lsode
y=lsode(fun,y0,x);
#graficamos la función y(x)
plot(x,y,'linewidth',2),grid on,xlabel('x'),ylabel('y(x)')
set(gca,'fontsize', 16)
```

El gráfico siguiente muestra la solución a la ecuación diferencial en el rango $x \in (0,5)$. Se verifica que el valor inicial es $y=2$ cuando $x=0$.



6.2.3 Ejemplo EDO con función trigonométrica

Considere la siguiente ecuación diferencial en base a la función $y(x)$ y su primera derivada dy/dx . La idea es encontrar la función $y(x)$ en el rango $x \in (-4,4)$, conociendo el valor inicial de la función como se indica a continuación.

$$\frac{dy}{dx} = -\frac{1}{4}x + 3 \sin(2\pi x)$$
$$y(-4) = 1$$

```
clc, clear all, close all

#definimos el rango para la variable independiente x.
x=[-4:0.05:4];

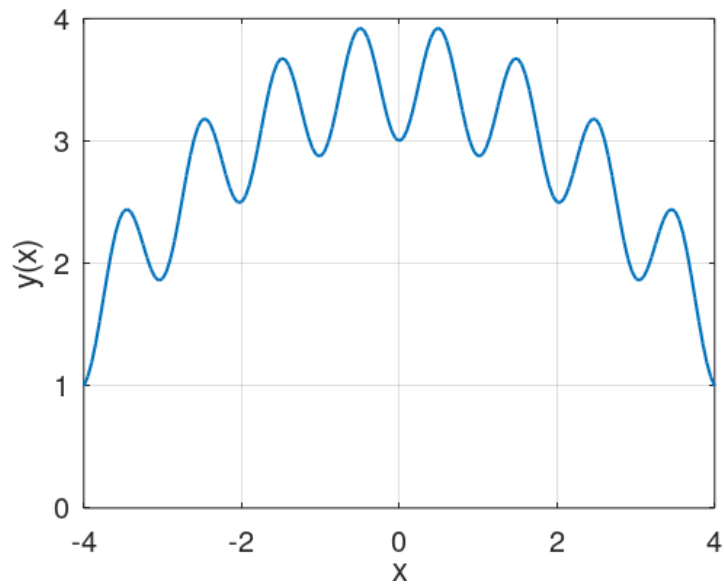
#definimos el valor inicial de la función y
#Es importante que el valor y0 este asociado al primer valor de la
#variable independiente x, en este caso -4.
y0=1;

#definimos la función f(y,x) a la derecha de la igualdad dy/dx = f(y,x)
fun = @(y,x) -1/4*x+3*sin(2*pi*x);

#utilizamos la función predefinida en octave lsode
y=lsode(fun,y0,x);

#graficamos la función y(x)
plot(x,y,'linewidth',1.5),grid on,xlabel('x'),ylabel('y(x)')
ylim([0 4])
set(gca,'fontsize', 14)
```

El gráfico siguiente muestra la solución a la ecuación diferencial en el rango $x \in (-4,4)$. Se verifica que el valor inicial es $y=1$ cuando $x=-4$.



6.2.4 Ejemplo ley de enfriamiento de Newton

Este ejemplo fue adaptado desde la referencia (26). Considere la siguiente ecuación diferencial que describe como se enfría un objeto. La función a encontrar es temperatura en función del tiempo, $T(t)$. Vamos a encontrar la temperatura de un objeto al cabo de 8 min. Sabiendo que en un comienzo estaba a una temperatura de 400°C , conociendo la constante de proporcionalidad $k = 0.014$ ($1/\text{s}^\circ\text{C}$) y que la temperatura ambiental es $T_a = 20^\circ\text{C}$.

$$T' = k(T_a - T)$$

$$T(0) = 1200$$

```
clc, clear all, close all

#definimos los valores constantes en el problema
k=0.014;
Ta=20;

#definimos el rango para la variable independiente,
#en este caso desde 0 a 8 min.
t=[0:0.1:8*60];

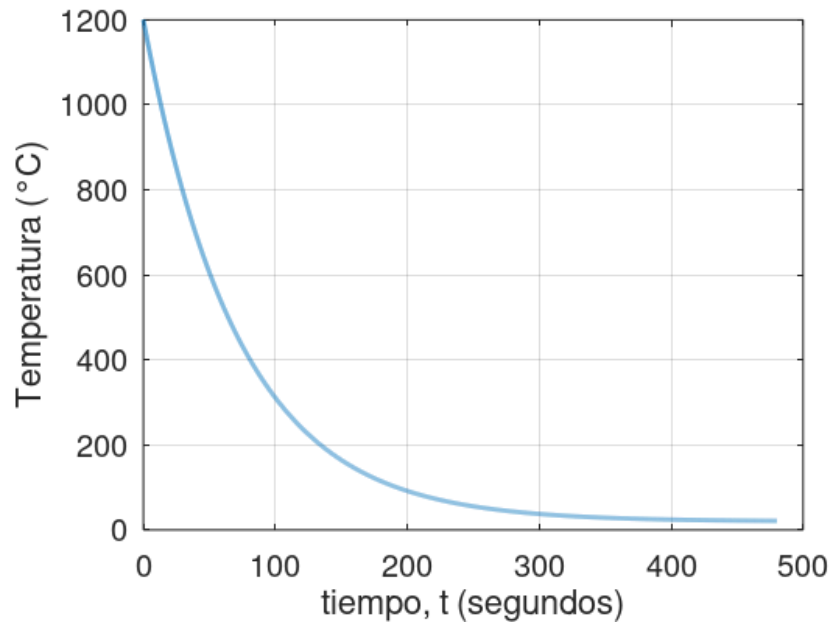
#definimos el valor inicial de la función T
T0=1200;

#definimos la función f(T,t) a la derecha de la igualdad dT/dt = f(T,t)
fun= @(T,t) k*(Ta-T);

#utilizamos la función predefinida en OCTAVE lsode
T= lsode(fun, T0, t);

#graficamos la función y(x)
plot(t,T,'linewidth',2),grid on
xlabel('Tiempo, t (segundos)'),ylabel('Temperatura (°C)')
set(gca,'fontsize', 14)
```

El gráfico siguiente muestra la solución a la ecuación diferencial en el rango $t \in (0, 480 \text{ s})$. Se verifica que la temperatura inicial es $T=1200^\circ\text{C}$ cuando $t=0$.



De acuerdo con como definimos el dominio del tiempo, tenemos la solución en 4801 puntos discretos de tiempo.

```
>> length(T)
```

```
ans = 4801
```

El último valor de T es

```
>> T(4801)
```

```
ans = 21.424
```

Es decir, al cabo de 8 min (480 segundos), el cuerpo tiene una temperatura de 21.4°C , casi se iguala a la temperatura ambiental.

6.2.5 Ejemplo vibración libre amortiguada

La siguiente ecuación describe la vibración libre de un sistema dinámico de un grado de libertad, x , en función del tiempo, t .

$$mx(t)'' + cx'(t) + kx(t) = 0$$

$$x(0) = 0.2$$

$$x'(0) = 1.0$$

Donde m es la masa del sistema, c es el amortiguamiento y k es la rigidez. $x(t)$ define la posición del sistema, $x'(t)$ su velocidad y $x''(t)$ su aceleración en función del tiempo. Asumiendo que $m = 1$, $c = 0.2$ y $k = 0.7$ en unidades consistentes, y conociendo los valores iniciales de posición y velocidad, encontremos la función desplazamiento en el rango $t \in (0,60)$.

Es necesario hacer una transformación para dejar esta expresión en la forma deseada, un sistema diferencial equivalente de primer orden. Sea:

$$u_1 = x(t)$$

$$u_2 = x'(t)$$

Por lo tanto, la ecuación a escribir tiene dos componentes, es un sistema.

$$u'(t) = f(x, t)$$

$$f(x, t) = \begin{cases} u_1' \\ u_2' \end{cases} = \begin{cases} x'(t) \\ -\frac{1}{m}(cx'(t) + kx(t)) \end{cases} = \begin{cases} u_2 \\ -\frac{1}{m}(cu_2 + ku_1) \end{cases}$$

$$u_1(0) = 0.2$$

$$u_2(0) = 1.0$$

```
clc, clear all, close all

#definimos los valores constantes en el problema
m=1; c=0.2; k=0.7;

#definimos los valores iniciales de posición y velocidad
x0=0.2;
v0=1;

#definimos los valores iniciales de las funciones u1 y u2
u10=x0;
u20=v0;

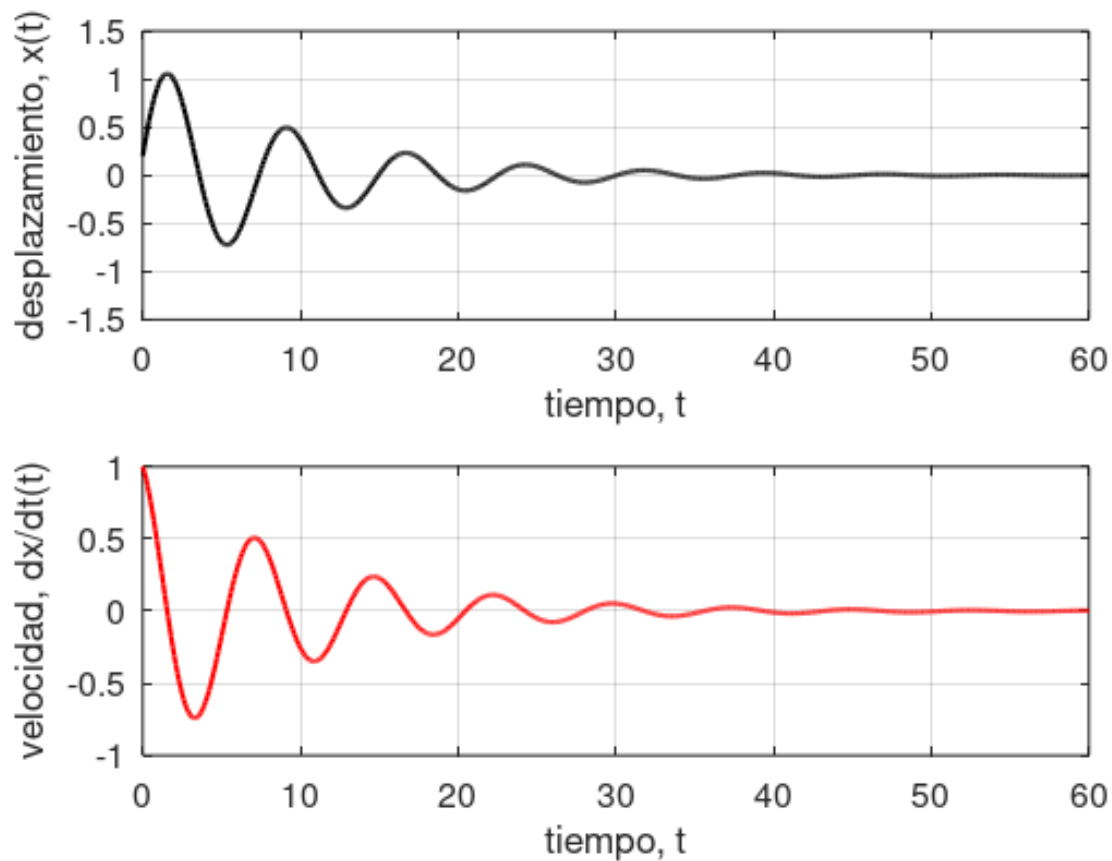
#definimos el rango para la variable independiente
t = [0:0.1:60];

#definimos la función f(u,t) a la derecha de la igualdad du/dt = f(u,t)
#cada ecuación es una fila
fun = @(u,t) [ u(2) ; -1/m*( c*u(2) + k*u(1) ) ];

#utilizamos la función predefinida en OCTAVE lsode
#el valor inicial del sistema se debe ingresar como dos filas
u = lsode(fun, [u10 ; u20] ,t);

#graficamos la función u(t) subdividiendo el entorno figure
#con la opción subplot
figure(1)
    subplot(2,1,1)
        plot(t,u(:,1),'linewidth',1.5,'color','k'),grid on;
        xlabel('tiempo, t'), ylabel('desplazamiento, x(t)')
        ylim([-1.5 1.5])
        set(gca,'fontsize', 11)
    subplot(2,1,2)
        plot(t,u(:,2),'linewidth',1.5,'color','r'),grid on;
        xlabel('tiempo, t'), ylabel('velocidad, dx/dt(t)')
        set(gca,'fontsize', 11)
```


El gráfico siguiente muestra la solución a la ecuación diferencial en el rango $t \in (0, 60)$. Se verifica que la posición inicial es $x = 0.2$ y la velocidad inicial $x' = 1$ cuando $t = 0$.



De acuerdo con la solución aproximadamente al tiempo $t = 40$, el objeto está prácticamente detenido en su posición de equilibrio $x = 0$.

6.2.6 Ejemplo vibración libre con amortiguamiento no-lineal

Este ejemplo fue adaptado desde la referencia (26). La siguiente ecuación describe la vibración libre de un sistema dinámico de un grado de libertad, x , en función del tiempo, t . Con un amortiguamiento no lineal. Puede ver que este problema tiene una forma muy similar a la ecuación de equilibrio previa. Sin embargo, la velocidad (el término x') no va acompañada de una constante, sino que de una función.

Encontremos las funciones desplazamiento y velocidad para este sistema conociendo los valores iniciales.

$$x'' - (1 - x^2)x' + x = 0$$

$$x(0) = 2$$

$$x'(0) = 0$$

El sistema de ecuaciones equivalente se puede expresar de la siguiente manera

$$\begin{cases} z'_1 \\ z'_2 \end{cases} = \begin{cases} z_2 \\ (1 - z_1^2)z_2 - z_1 \end{cases}$$

$$z_1(0) = 2$$

$$z_2(0) = 0$$

```
clc, clear all, close all

#definimos el rango para la variable independiente t.
t=[0:0.1:20];

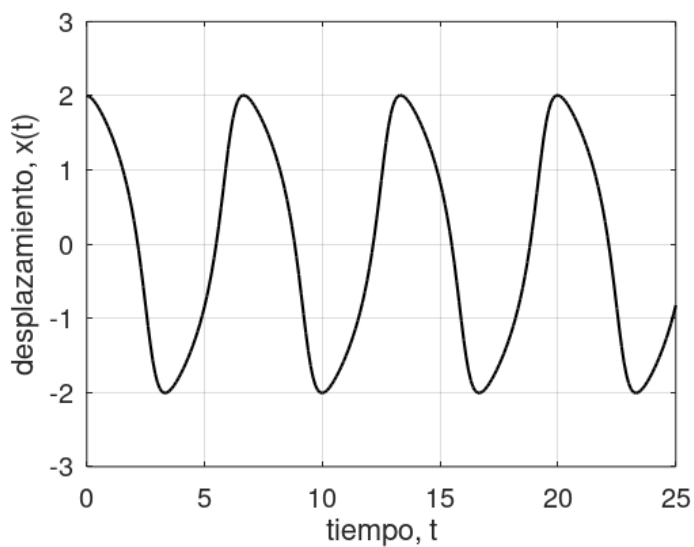
#definimos el valor inicial de la función z
#z tiene dos componentes, por lo tanto z0 se debe ingresar como un
#vector columna
z0=[2; 0];

#definimos la función f(z,t) a la derecha de la igualdad dz/dt = f(z,t)
#cada ecuación es una fila
fun = @(z,t) [z(2) ; ( 1 - z(1).^2 ).*z(2) - z(1)];

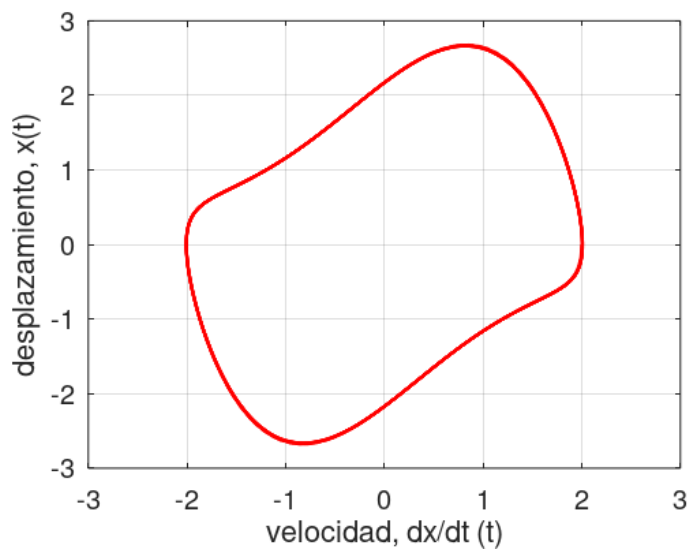
#utilizamos la función predefinida en OCTAVE lsode
z = lsode(fun, z0, t);

#graficamos la función z(t) en dos figuras
figure(1)
    plot(t,z(:,1),'linewidth',1.5,'color','k'),grid on
    xlabel('tiempo, t'), ylabel('desplazamiento, x(t)')
    set(gca,'fontsize', 14)
figure(2)
    plot(z(:,1),z(:,2),'linewidth',1.5,'color','r'),grid on
    xlabel('velocidad, dx/dt (t)'), ylabel('desplazamiento, x(t)')
    set(gca,'fontsize', 14)
```

El gráfico siguiente muestra la solución a la ecuación diferencial en el rango $t \in (0, 25)$. Se verifica que la posición inicial es $x = 2$ cuando $t = 0$. Es interesante que la vibración pierde su forma sinusoidal al agregar un amortiguamiento no-lineal.



El gráfico siguiente muestra la relación desplazamiento versus velocidad del objeto.



6.3.- Ejercicios propuestos

Como ejercitación de lo aprendido en esta guía se sugiere resolver las siguientes ecuaciones diferenciales ordinarias con `lsode()` de OCTAVE. La solución parcial o total de algunos de los ejercicios propuestos se pueden encontrar en las referencias (27–30). Se recomienda graficar juntas la solución teórica disponible con la ecuación numérica encontrada con OCTAVE.

Ejercicio 1

$$\frac{dx}{dt} = te^{-2t}$$

$$x(0) = 1$$

$$\text{rango } t \in (0,10)$$

Ejercicio 2

$$\frac{dx}{dt} = 5x - 3$$

$$x(2) = 1$$

$$\text{rango } t \in (2,4)$$

Ejercicio 3

$$\frac{dy}{dx} = x^2 - 3$$

$$y(0) = 0$$

$$\text{rango } x \in (0,4)$$

Ejercicio 4

$$\frac{dy}{dx} = 7y^2x$$

$$y(0) = 1$$

$$\text{rango } x \in (0,1)$$

Ejercicio 5

$$dy + 7x dx = 0$$

$$y(0) = 3$$

$$\text{rango } x \in (0,1)$$

Ejercicio 6

$$ty' + 2ty = t^2 - t + 1$$

$$y(1) = 2$$

$$\text{rango } t \in (1,5)$$

Ejercicio 7

$$2y' - y = 4 \sin(3t)$$

$$y(0) = 1$$

$$\text{rango } t \in (0,6)$$

Ejercicio 8

$$\cos(x)y' + \sin(x)y = 2 \cos^3(x) \sin(x) - 1$$

$$y(\pi/4) = 3\sqrt{2}$$

$$\text{rango } x \in (\pi/4, 7\pi/4)$$

Ejercicio 9

$$\frac{d^2y}{dx^2} = 2 \frac{dy}{dx}$$

$$y(0) = 1 \text{ \& } y'(0) = 0$$

$$\text{rango } x \in (0, 2)$$

Ejercicio 10

$$y'' - 3y' + 2y = 0$$

$$y(0) = 0.5 \text{ \& } y'(0) = 1$$

$$\text{rango } x \in (0, 2)$$

Ejercicio 11

$$y'' + 4y = 0$$

$$y(0) = 0.5 \text{ \& } y'(0) = 1$$

$$\text{rango } x \in (0, 2\pi)$$

Ejercicio 12

$$y'' + \frac{1}{2}y' + 3y = \sin(0.2t)$$

$$y(0) = 0 \text{ \& } y'(0) = 0$$

$$\text{rango } t \in (0, 30)$$

Ejercicio 13

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L} \sin \theta$$

$$\text{Con } g/L \text{ conocido, } \theta(0) = 0.5 \text{ \& } \theta'(0) = 0$$

Ejercicio 14

Resuelva el ejemplo resuelto 6.2.5 para distintos valores de c y observe las diferencias.

Ejercicio 15

Resuelva el ejemplo resuelto 6.2.5 para distintos valores de k/m y observe las diferencias.

Referencias

1. Long PJG. Introduction to Octave [Internet]. 2005 [cited 2021 Oct 7]. Available from: <http://www-h.eng.cam.ac.uk/help/programs/octave/tutorial/>
2. Eaton JW. About OCTAVE [Internet]. [cited 2021 Oct 7]. Available from: <https://www.gnu.org/software/octave/about>
3. GNU Octave [Internet]. [cited 2021 Oct 7]. Available from: <https://www.gnu.org/software/octave/index>
4. Home Page | www.scilab.org [Internet]. [cited 2021 Oct 7]. Available from: <https://www.scilab.org/>
5. RStudio | Open source & professional software for data science teams [Internet]. [cited 2021 Oct 7]. Available from: <https://rstudio.com/>
6. Welcome to Python.org [Internet]. Python.org. [cited 2021 Oct 7]. Available from: <https://www.python.org/>
7. Compilers - C++ Tutorials [Internet]. [cited 2021 Oct 7]. Available from: <https://www.cplusplus.com/doc/tutorial/introduction/>
8. Home - Fortran Programming Language [Internet]. [cited 2021 Oct 7]. Available from: <https://fortran-lang.org/>
9. MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink [Internet]. [cited 2021 Oct 7]. Available from: <https://www.mathworks.com/>
10. El problema matemático de las frutas que ha enganchado a 2,5 millones de personas | Verne EL PAÍS [Internet]. [cited 2021 Oct 7]. Available from: https://verne.elpais.com/verne/2016/02/18/articulo/1455778788_314139.html
11. GNU Octave: Finding Roots [Internet]. [cited 2021 Oct 7]. Available from: <https://octave.org/doc/v4.2.0/Finding-Roots.html>
12. Function Reference: fzero [Internet]. [cited 2021 Oct 7]. Available from: <https://octave.sourceforge.io/octave/function/fzero.html>
13. GNU Octave: Two-Dimensional Plots [Internet]. [cited 2021 Oct 7]. Available from: https://octave.org/doc/v4.0.0/Two_002dDimensional-Plots.html
14. 2-D line plot - MATLAB plot [Internet]. [cited 2021 Oct 7]. Available from: <https://www.mathworks.com/help/matlab/ref/plot.html>
15. graphing functions - Examples of funny graphs [Internet]. Mathematics Stack Exchange. [cited 2021 Oct 7]. Available from: <https://math.stackexchange.com/questions/2509015/examples-of-funny-graphs>

16. Function Reference: ezplot [Internet]. [cited 2021 Oct 7]. Available from: <https://octave.sourceforge.io/octave/function/ezplot.html>
17. GNU Octave - Functions and Script Files [Internet]. [cited 2021 Oct 7]. Available from: http://www.eletr.ufpr.br/edu/pds/_lab/interpreter/octave_12.html
18. GNU Octave: The if Statement [Internet]. [cited 2021 Oct 7]. Available from: <https://octave.org/doc/v4.2.0/The-if-Statement.html>
19. GNU Octave: Element-by-element Boolean Operators [Internet]. [cited 2021 Oct 7]. Available from: https://octave.org/doc/v4.2.1/Element_002dbf_002delement-Boolean-Operators.html#Element_002dbf_002delement-Boolean-Operators
20. GNU Octave: The for Statement [Internet]. [cited 2021 Oct 7]. Available from: <https://octave.org/doc/v4.2.0/The-for-Statement.html>
21. The switch Statement (GNU Octave) [Internet]. [cited 2021 Oct 7]. Available from: <https://octave.org/doc/v4.2.2/The-switch-Statement.html>
22. Execute one of several groups of statements - MATLAB switch case otherwise [Internet]. [cited 2021 Oct 7]. Available from: <https://www.mathworks.com/help/matlab/ref/switch.html>
23. The while Statement (GNU Octave (version 5.2.0)) [Internet]. [cited 2021 Oct 7]. Available from: <https://octave.org/doc/v5.2.0/The-while-Statement.html>
24. while loop to repeat when condition is true - MATLAB while [Internet]. [cited 2021 Oct 7]. Available from: <https://www.mathworks.com/help/matlab/ref/while.html>
25. Ordinary Differential Equations (GNU Octave (version 4.4.1)) [Internet]. [cited 2021 Oct 7]. Available from: <https://octave.org/doc/v4.4.1/Ordinary-Differential-Equations.html>
26. Manual GNU-OCTAVE [Internet]. 2015. Available from: https://introoctave.github.io/biblio_archivos/2015_UPM_manual-octave_rv04a.pdf
27. Ordinary differential equation examples - Math Insight [Internet]. [cited 2021 Oct 7]. Available from: https://mathinsight.org/ordinary_differential_equation_introduction_examples
28. Bourne M. 1. Solving Differential Equations [Internet]. [cited 2021 Oct 7]. Available from: <https://www.intmath.com/differential-equations/1-solving-des.php>
29. Differential Equations - Linear Equations [Internet]. [cited 2021 Oct 7]. Available from: <https://tutorial.math.lamar.edu/classes/de/linear.aspx>
30. Differential Equations - Introduction [Internet]. [cited 2021 Oct 7]. Available from: <https://www.mathsisfun.com/calculus/differential-equations.html>