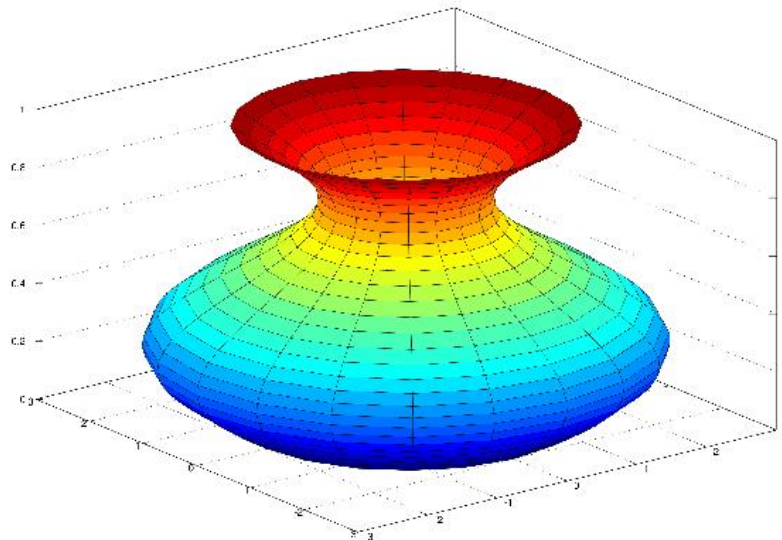




UNIVERSIDAD NACIONAL DE JAÉN
Ley de Creación N° 29304
Autorizada con Resolución N° 647-2011-CONAFU

Separata de:
**QtOCTAVE APLICADO A MÉTODOS
NUMÉRICOS**



Lenin Quiñones Huatangari

Índice general

1. Nociones de Computación	4
1.1. Software Libre	4
1.2. Octave	6
1.3. QtOctave	8
1.3.1. QtOctave como Calculadora	10
1.3.2. Funciones Matemáticas en QtOctave	11
1.3.3. Arrays (Vectores)	13
1.3.4. Creación de Array Bidimensionales (Matrices)	17
2. Gráficos con QtOctave	20
2.1. Gráficas en 2D	21
2.1.1. Coordenadas Rectangulares	21
2.1.2. Coordenadas Paramétricas	27
2.1.3. Coordenadas Polares	28
2.2. Gráficas en 3D	29
3. Álgebra Lineal	38
3.1. Operaciones elementales de Matrices	38
3.2. Valores y Vectores Propios de una Matriz	42
3.3. Sistema de Ecuaciones Lineales	44

3.4. Espacios Vectoriales y Aplicaciones Lineales	46
4. Polinomios	50
Bibliografía	54

AL ESTUDIANTE

El modelamiento matemático de fenómenos naturales es una técnica frecuentemente utilizada en la ciencia y tecnología moderna. El avance de esta área está fuertemente vinculada al desarrollo de las ciencias computacionales. A modo general, se puede establecer que los modelos matemáticos existentes en la actualidad establecen relaciones funcionales no lineales entre las variables cuantitativas y en consecuencia muy pocas veces pueden ser resueltos de manera analítica. En este punto, existe la imperiosa necesidad de desarrollar algoritmos que finalmente se traducen en códigos computacionales. Es así que la ayuda de los computadores, para simular estos modelos, ha traído consigo un gran desarrollo científico y tecnológico.

El objetivo de este trabajo es estudiar un programa muy específico, como el QtOctave que pertenece a un gran proyecto de Software Libre propuesto por Richard Stallman. Estudiaremos todas las funciones o programas que vienen implementadas dentro del QtOctave, como graficar funciones en 2D-3D, operar con matrices, etc y finalmente utilizarlo como un lenguaje de programación, que es lo que más nos interesa por ende crear nuestros propios programas de acuerdo a nuestras necesidades.

En conclusión el autor le desea al alumno éxitos y al mismo tiempo que disfrute el texto y el curso que va a iniciar. Si el lector tiene comentarios, encuentra errores o si tiene una buena idea para mejorar el texto contacte al autor en *lquinoneshuatangari@unj.edu.pe*.

Capítulo 1:

Nociones de Computación

1.1. Software Libre

Además de aprender a utilizar QtOctave (objetivo de este manual), es interesante conocer los orígenes del Software Libre y GNU/Linux. De esta forma podremos entender el modelo de desarrollo libre a partir de su definición. Esta sección esta dedicada a cubrir ese aspecto tan importante.

Allá por el 1971, cuando la informática todavía no había sufrido su gran boom, las personas que hacían uso de ella, en ámbitos universitarios y empresariales, creaban y compartían el software sin ningún tipo de restricciones. Con la llegada de los años 80 la situación empezó a cambiar. Las computadoras más modernas comenzaban a utilizar sistemas operativos privativos, forzando a los usuarios a aceptar condiciones restrictivas que impedían realizar modificaciones a dicho software.

En caso de que algún usuario o programador encontrase algún error en la aplicación, lo único que podía hacer era darlo a conocer a la empresa desarrolladora para que esta lo solucionara. Aunque el programador estuviese

capacitado para solucionar el problema y lo desease hacer sin pedir nada a cambio, el contrato le impedía que mejorase el software. Esta situación provocó la destrucción de comunidades cooperativas donde el software era compartido y cualquiera podía mejorarlo sin restricciones.

El modelo de desarrollo de aplicaciones propietarias, a pesar de generar situaciones anti- sociales, se impuso con tal fuerza que en la actualidad hay aún personas convencidas de que no hay otra forma de hacer negocio. Durante la etapa de transición al modelo privativo, Richard M. Stallman, trabajador del laboratorio de Inteligencia Artificial del MIT (Massachusetts Institute of Technology), se percató que la sociedad estaba cambiando peligrosamente. El mismo Richard Stallman cuenta que por aquellos años, en el laboratorio habían recibido una impresora donada por una empresa externa. El dispositivo, que era utilizado en red por todos los trabajadores, parecía no funcionar a la perfección dado que cada cierto tiempo el papel se atascaba. Como agravante, no se generaba ningún aviso que se enviase por red e informase a los usuarios de la situación. La pérdida de tiempo era constante, ya que en ocasiones, los trabajadores enviaban por red sus trabajos a imprimir y al ir a buscarlos se encontraban la impresora atascada y una cola enorme de trabajos pendientes. Richard Stallman decidió arreglar el problema, e implementar el envío de un aviso por red cuando la impresora se bloqueara. Para ello necesitaba tener acceso al código fuente de los controladores de la impresora. Pidió a la empresa propietaria de la impresora lo que necesitaba, comentando, sin pedir nada a cambio, que era lo que pretendía realizar. La empresa se negó a entregarle el código fuente. En ese preciso instante, Richard Stallman se vio en una encrucijada, debía elegir entre aceptar el nuevo software privativo firmando acuerdos de no revelación y acabar desarrollando más software

privativo con licencias restrictivas, que a su vez deberían ser más adelante aceptadas por sus propios colegas.

Richard Stallman se negó a aceptar el nuevo software privativo, dado que este le obligaría a firmar acuerdos de no revelación. Quería evitar acabar contribuyendo a la expansión de ese tipo de software, el cual solo conseguía generar una sociedad más dividida y con menos libertades. Abandonó el MIT en 1984, para evitar problemas de propiedad del software, e inició un proyecto para intentar formar una comunidad de personas, en las que compartir el código volviese a ser algo natural. El proyecto fue denominado **GNU (GNU's Not Unix)**, su finalidad era la construcción de un sistema operativo compatible con UNIX pero completamente libre.

1.2. Octave

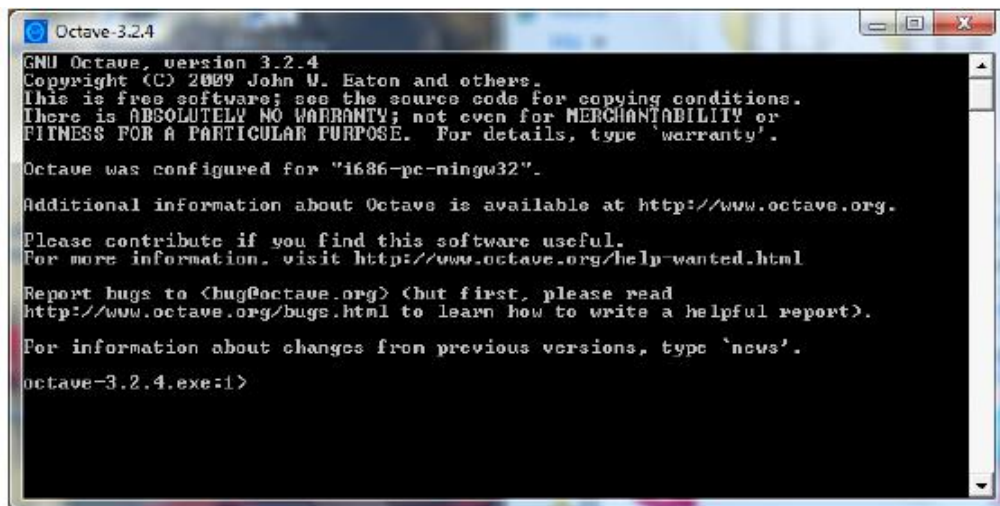
Octave o GNU Octave es un lenguaje de alto nivel libre destinado para el cálculo numérico. Como indica su nombre es parte del proyecto GNU. Apoyado en una amplia comunidad de desarrolladores y usuarios. Octave posee una gran cantidad de herramientas que permiten resolver problemas de álgebra lineal, cálculo de raíces de ecuaciones no lineales, integración de funciones ordinarias, manipulación de polinomios, integración de ecuaciones diferenciales ordinarias y ecuaciones diferenciales algebraicas. Además permite hacer imágenes en 2D - 3D, ejecutar scripts y puede ser usado como lenguaje de programación.

Octave nació alrededor del año 1988, y fue concebido originalmente para ser usado en un curso de diseño de reactores químicos para los alumnos de

Ingeniería Química de la Universidad de Texas y la Universidad de Wisconsin-Madison.

La descarga del programa puede hacerse desde la página oficial del proyecto <http://www.gnu.org/software/octave/>. En esta página podemos encontrar el programa octave para las diferentes plataformas: Windows, Linux, Mac, etc.

Para Windows, existe ya el instalador precompilado y puede descargarse directamente de la dirección: <http://octave.sourceforge.net/>



```
Octave-3.2.4
GNU Octave, version 3.2.4
Copyright (C) 2009 John W. Eaton and others.
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 "i686-pc-ningu32".

Additional information about Octave is available at http://www.octave.org.
Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html
Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).
For information about changes from previous versions, type 'news'.
octave-3.2.4.exe:1>
```

Figura 1.1: Interfaz con el usuario de Octave

Como podemos ver la interfaz con el usuario no es demasiado agradable, es por ello que otros proyectos también de software libre (QtOctave), se han dedicado a mejorar esta interfaz, para hacer más fácil al usuario la forma de interactuar con Octave. Para salir de esta ventana basta teclear quit o exit.

1.3. QtOctave

Nosotros utilizaremos QtOctave, puesto que usa interfaz con el usuario. Este programa está disponible para distintos sistemas operativos, cuenta con numerosos menús, botones y ventanas de diálogo que, aun encontrándose todavía en fase experimental, podemos decir que se trata de una herramienta que facilita al usuario la comunicación con Octave. La página principal de este programa es: <http://qtoctave.wordpress.com/>. Para Windows hay que descargar el fichero comprimido: qtoctave-win32-0.9.1-3.zip. Esta interfaz no necesita instalación es suficiente con descomprimir en una carpeta y dentro de la carpeta bin encontramos el fichero ejecutable: qtoctave.exe.

Describiré brevemente todas las subventanas que conforman a QtOctave.

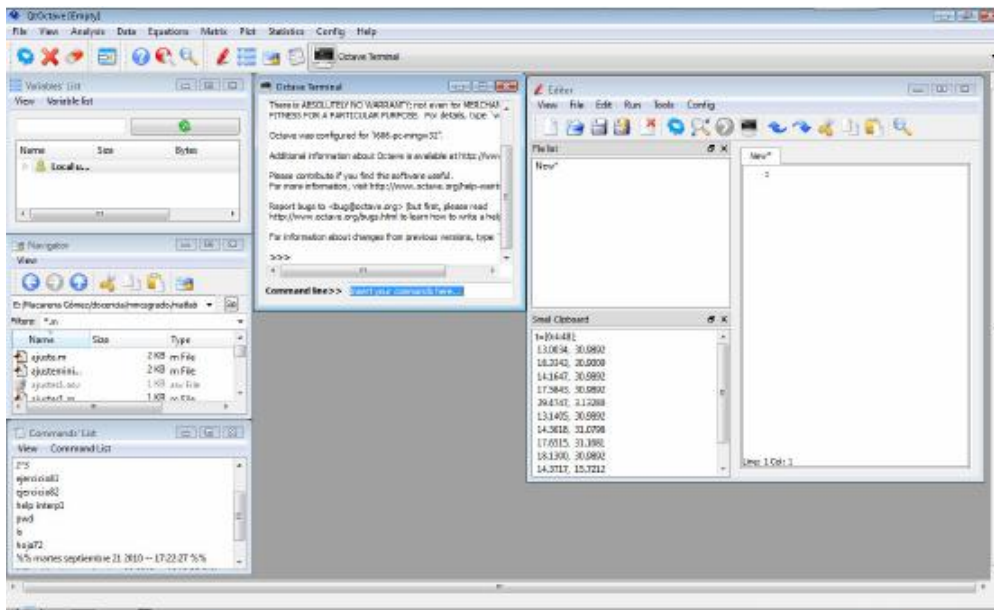


Figura 1.2: Ventana Inicial de QtOctave

Ventana Octave Terminal

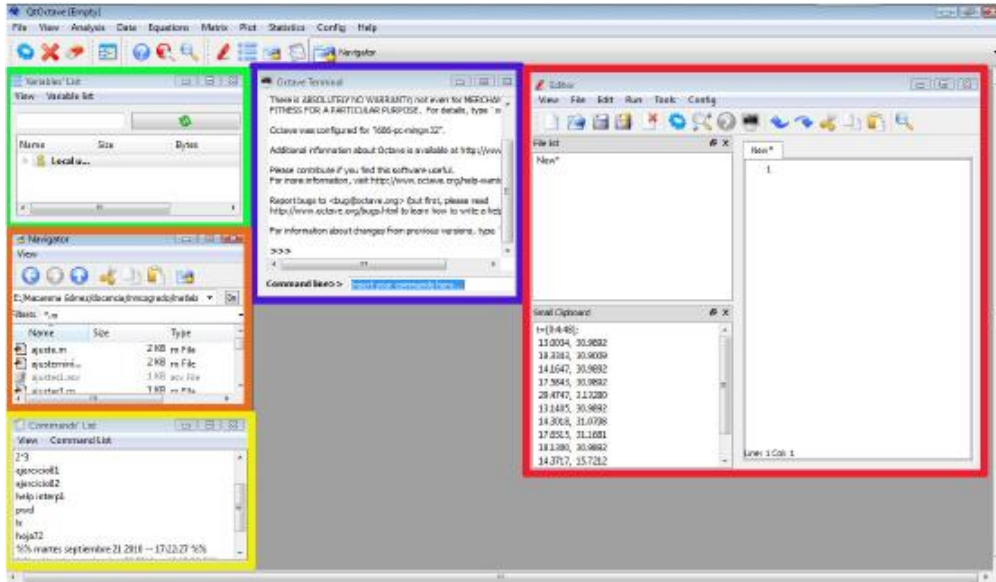


Figura 1.3: Sub-ventanas de comandos

Es la parte más importante de la ventana inicial, que aparece en la parte derecha (recuadrada en azul en la Figura 1.3). En esta sub-ventana es donde se ejecutan los comandos de Octave, a continuación del **Command Line** `>>` que indica que el programa está preparado para recibir instrucciones.

Ventana Variable Lists

En la parte superior aparece la ventana (recuadrada en verde en la Figura 1.3) Variable Lists que contiene información sobre todas las variables que se hayan definido en esta sesión y permite ver y modificar las matrices y vectores con los que se esté trabajando.

Ventana Navigator

A continuación tenemos la pantalla Navigator (recuadrada en naranja en la Figura 1.3) donde se muestra los ficheros del directorio activo o actual.

Ventana Command List

En la parte inferior aparece la ventana (recuadrada en amarillo en la Figura 1.3) Commands List que muestra los últimos comandos ejecutados en la Octave Terminal. Estos comandos se pueden volver a ejecutar haciendo doble clic sobre ellos.

Ventana del Editor

Por último aparece una ventana (recuadrada en rojo en la Figura 1.3) que corresponde al Editor de texto para los programas. Estas ventanas pueden ser colocadas en cualquier orden dentro del espacio de trabajo, pueden estar cerradas si no van a ser utilizadas, y pueden abrirse en cualquier momento.

1.3.1. QtOctave como Calculadora

Recomendaciones Básicas para usar QtOctave.

- Para teclear un comando el cursor debe estar después de la frase **Command Line >>**
- Una vez que se teclea el comando en el formato indicado, pulsar la tecla intro, para ejecutar tal orden.
- Con las teclas: ↑ y ↓, se pueden invocar comandos tecleados anteriormente, y ejecutarlos parcial o totalmente.

- Las salidas en pantalla que produce la ejecución de un comando, se visualizara en la ventana de comandos. Pero si se teclea punto y coma (;) al final del comando este se ejecutara, pero ya no se visualizara.
- En una misma línea se pueden teclear varios comandos, pero separado por comas. Si se separan por punto y coma, no se visualizara el resultado.
- Cuando se teclea el símbolo Si se coloca el símbolo El comando clc limpia la pantalla de MatLab, solo deja en blanco la pantalla y todo lo ejecutado permanece intacto.

1.3.2. Funciones Matemáticas en QtOctave

Además de las operaciones aritméticas, se tiene en QtOctave una variedad adicional de funciones adicionales, dentro de las cuales tenemos:

Función	Descripción
sqrt(x)	Raíz cuadrada
exp(x)	Exponencial e^x
abs(x)	Valor Absoluto.
log(x)	Logaritmo natural de base e .
log10(x)	Logaritmo decimal de base 10.
factorial(x)	Factorial de un número.

Función	Descripción
$\sin(x)$	Seno de x , x en radianes
$\cos(x)$	coseno de x , x en radianes
$\text{asin}(x)$	Arco seno de x
$\text{acos}(x)$	Arco coseno de x
$\text{atan}(x)$	Arco tangente de x
$\text{acot}(x)$	Arco cotangente de x .
$\sinh(x)$	eno hiperbólico de x .
$\cosh(x)$	coseno hiperbólico de x
$\tanh(x)$	Tangente hiperbólico de x .
$\text{coth}(x)$	Cotangente hiperbolica de x
$\tan(x)$	Tangente de x , x en radianes.
$\cot(x)$	Cotangente de x , x en radianes.

Función	Descripción
$\text{round}(x)$	Redondea al entero más próximo
$\text{fix}(x)$	Redondea hacia cero.
$\text{ceil}(x)$	Redondea hacia el infinito.
$\text{floor}(x)$	Redondea hacia menos infinito
$\text{rem}(x,y)$	Retorna el resto de la división de x entre y
$\text{sign}(x)$	Función signo

Ejemplo 1.1. Utilizaremos algunas funciones de matemáticas, implementadas en QtOctave.

```
>>> sqrt(289)+exp(2)-abs(-20)
```

```
ans = 4.3891
```

```
>>> log(40)
```

```
ans = 3.6889
```

```
>>> exp(1)
ans = 2.7183
>>> log(exp(40))
ans = 40
>>> log10(123456)
ans = 5.0915
>>> log10(100000000000)
ans = 11
>>> factorial(6)
ans = 720
>>> sin(pi/2)+cos(pi)-tan(pi/4)
ans = -1.0000
>>> tan(pi/4)+cot(pi/4)
ans = 2
>>>
```

1.3.3. Arrays (Vectores)

Un array es simplemente un arreglo rectangular de números, que puede ser unidimensional (Vector) o multidimensional (Matriz). Los arrays también pueden estar compuestos por caracteres no numéricos, llamados cadenas o strings.

Creación de un Vector

Un vector se puede entender como un conjunto de datos, que se pueden ordenarse en forma de fila o columna. También pueden ser el resultado de alguna función que determine el valor de este vector.

Ejemplo 1.2. En la siguiente tabla se muestran datos sobre crecimiento

demográfico, además de esta se puede originar dos vectores, que pueden ser en forma de filas (horizontal), o columnas (vertical).

Ano	1984	1986	1988	1990	1992	1994	1996
Población (Millones)	127	130	136	145	158	178	211

A veces se hace necesario la extracción, cambio o reducción de términos de un array. También a veces se necesita agregar elementos a estos arreglos.

Ejemplo 1.3. Veamos el siguiente vector:

```
>>> v1=[6 8 -4 3 2 1 0 9 2 -5]
v1 =      6      8     -4      3      2      1      0      9      2     -5
>>>
```

Reasignando valores

```
>>> v1(6)
ans =      1
>>> v1(6)=12
v1 =      6      8     -4      3      2     12      0      9      2     -5
>>> v1(3)+5*v1(7)-6*v1(2)
ans =    -52
>>> 8*v1(2)+7*v1(5)-(v1(7)+2)^2
ans =      74
>>>
```

El operador dos puntos genera un rango de números, en un vector.

Expresión	Descripción
$v(:)$	Todos los elementos del vector v
$v(m : n)$	Los elementos del vector, desde posición m hasta posición n .
$v(m : k : n)$	Los elementos del vector, desde la posición m hasta la posición n , con un incremento k .

Ejemplo 1.4. Usando los dos puntos dentro de un vector.

```
>>> u=[6 8 -4 3 2 1 0 9 2 -5]
u =      6      8     -4      3      2      1      0      9      2     -5
>>> u(:)
ans =      6
      8
     -4
      3
      2
      1
      0
      9
      2
     -5
>>> u(3:7)
ans =     -4      3      2      1      0
>>> u(1:2:10)
ans =      6     -4      2      0      2
>>> u(9:-1:1)
ans =      2      9      0      1      2      3     -4      8      6
>>>
```


Creación de un vector con distancia constante. Aquí los datos son el valor inicial y el final, además el incremento (distancia). En un vector con distancia o espaciado constante, la diferencia entre los elementos siempre es la misma. Si no se indica el valor de distancia, por defecto es 1.

Ejemplo 1.5. Creando vectores con distancia constante.

```
>>> x=[4:17:81]
x =     4     21     38     55     72
>>> x=[4:17:89]
x =     4     21     38     55     72     89
>>> x=[4:9]
x =     4     5     6     7     8     9
>>> x=[12:-3:-20]
x =    12     9     6     3     0    -3    -6    -9   -12   -15   -18
>>> x=[8:-3:20]
x = Empty matrix: 1-by-0
>>>
```

Creación de vectores de una longitud definida. Si se necesita crear un vector donde se conoce el primer y el último elemento, así como el número de términos. Para ello se usa el comando `linspace`. Si no se indica el valor de `n`, por defecto es 100.

Ejemplo 1.6. Creando vectores con una longitud definida.

```
>>> v1=linspace(4,20,8)
v1 =    4.0000    6.2857    8.5714   10.8571   13.1429   15.4286   17.7143   20.0000
>>> v2=linspace(4,20,9)
v2 =     4     6     8    10    12    14    16    18    20
```

```
>> v3=linspace(4,20,-9)
v3 =    20
>>> v4=linspace(3.5,-20.5)
v4 = Columns 1 through 4           Columns 97 through 100
      3.5000    3.2576    3.0152           20.0152 -20.2576 -20.5000
>>>
```

1.3.4. Creación de Array Bidimensionales (Matrices)

En este caso los números se distribuyen en filas (horizontal) y columnas (vertical). Son utilizadas en el Algebra Lineal, y se usan para linealizar los problemas, es decir para suavizarlos. Los elementos de una matriz se ingresan fila por fila, separados por punto y coma. También se pueden separar las filas por un enter.

Ejemplo 1.7. Considere la matriz A de tres filas y cuatro columnas:

```
>>> A=[6 8 -4 3;5 2 1 0;6 9 2 -5]
A =
      6      8     -4      3
      5      2      1      0
      6      9      2     -5
>>> A=[6 8 -4 3
5 2 1 0
6 9 2 -5]
A =
      6      8     -4      3
      5      2      1      0
      6      9      2     -5
```

```
>>> A=[6 8 -4 3,5 2 1 0,6 9 2 -5]
A =      6      8     -4      3      5      2      1      0      6      9      2     -5
>>> a=8;b=sqrt(5);m=exp(3);p=pi;
>>> D=[a+b   b+m   p+a+b;m+log(8)   2*a+7*b   sin(pi/2)]
D =   10.2361   22.3216   13.3777
      22.1650   31.6525    1.0000
>>>
```

El operador dos puntos genera un rango de números, en una matriz.

Expresión	Descripción
$A(:, n)$	Indica todos los elementos de la columna n
$A(m, :)$	Indica todos los elementos de la fila m
$A(:, m : n)$	Indica los elementos de las columnas m hasta la n.
$A(m : n, :)$	Indica los elementos de las filas m hasta la n

Ejemplo 1.8. Utilizando los dos puntos dentro de una matriz.

```
>>> A=[6 8 -4 3 6 1;5 2 1 0 9 8;6 9 2 -5 7 0;4 8 12 3 9 2]
A =      6      8     -4      3      6      1
          5      2      1      0      9      8
          6      9      2     -5      7      0
          4      8     12      3      9      2
>>> A(:,6)
ans =      1
          8
          0
          2
>>> A(3, :)
ans =      6      9      2     -5      7      0
```

```
>> A(2:3,3:6)
ans =     1     0     9     8
          2    -5     7     0

>>> A(2:3,6:-1:1)
ans =     8     9     0     1     2     5
          0     7    -5     2     9     6

>>> A,B=A([1,4],[2,4,5])
A =
     6     8    -4     3     6     1
     5     2     1     0     9     8
     6     9     2    -5     7     0
     4     8    12     3     9     2
B =     8     3     6
          8     3     9

>>>
```

Capítulo 2:

Gráficos con QtOctave

Como la mayor parte de las ecuaciones matemáticas expresa relaciones complicadas en una, dos, tres o mas dimensiones, tratar de entenderlas sin gráficas es algo imposible. El empleo de gráficas es importante desde la educación primaria hasta la superior, así como para Ingenieros y Científicos profesionales. En las presentaciones profesionales, casi todos los análisis matemáticos, científicos y de ingeniería se presentan con gráficos. Los gráficos son ahora una parte natural del entorno de computación con QtOctave, y la graficación de los resultados de los cálculos puede efectuarse con algunos comandos. Tratar de entender las ecuaciones matemáticas con graficas es una forma agradable y muy eficiente de aprender matemáticas. Por lo que el objetivo de esta unidad es ayudar a los estudiantes a efectuar graficas en dos y tres dimensiones de diversas funciones.

2.1. Gráficas en 2D

2.1.1. Coordenadas Rectangulares

Graficas Simples

Ejemplo 2.1. Trabajando en la ventana de comandos de Octave, tenemos:

```
>>> x=[2 4 6 8 10 12 14 16 18 20 22 24];  
>>> y=[21 22.4 27 28 30 32 31 30 27 26 24 20];  
>>> plot(x,y)
```

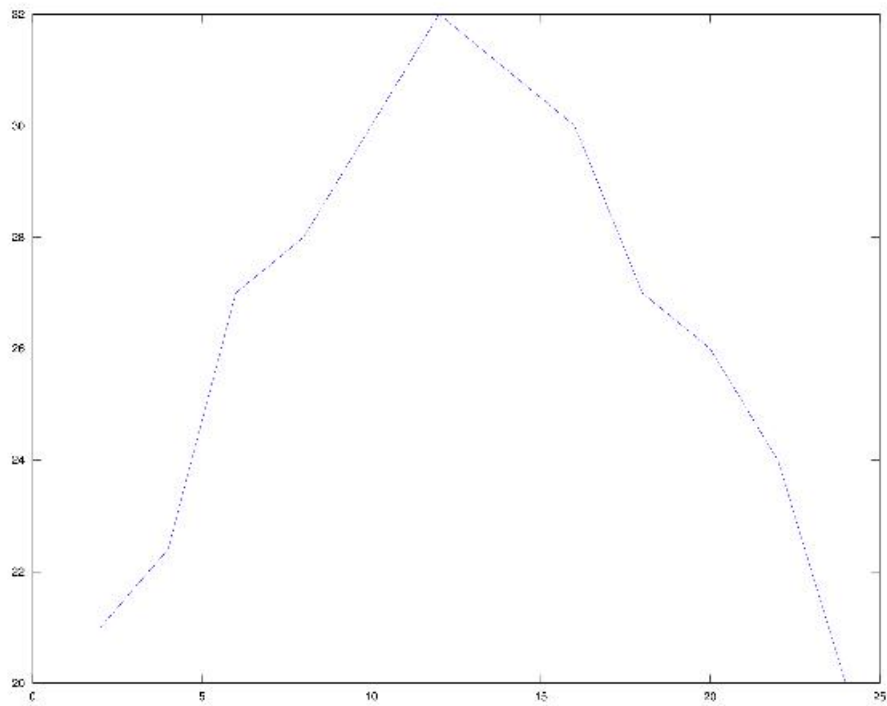


Figura 2.1: Grafica simple. Ejemplo 2.1

Ejemplo 2.2. Ahora también se le puede agregar una malla al gráfico con `grid`.

```
>>> x=[2 4 6 8 10 12 14 16 18 20 22 24];  
>>> y=[21 22.4 27 28 30 32 31 30 27 26 24 20];  
>>> plot(x,y)  
>>> grid
```

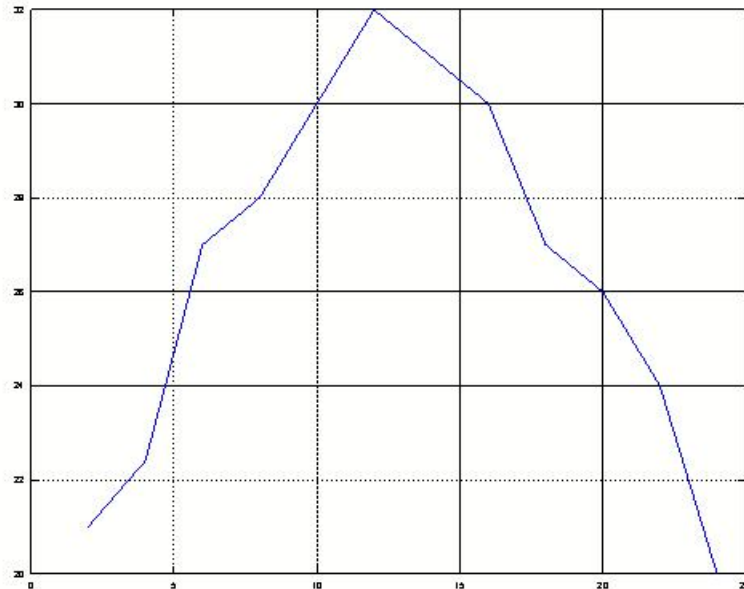


Figura 2.2: Gráfica simple con ejes cuadrados. Ejemplo 2.2

Ejemplo 2.3. Luego se pueden agregar indicaciones adicionales, como título del cuadro, nombres a los ejes x e y. Con los comandos: `title`, `x label`, `y label`.

```
>>> x=[2 4 6 8 10 12 14 16 18 20 22 24];
```

```
>>> y=[21 22.4 27 28 30 32 31 30 27 26 24 20];  
>>> plot(x,y)  
>>> grid  
>>> title('Registro diario de la temperatura')  
>>> xlabel('horas')  
>>> ylabel('temperatura')
```

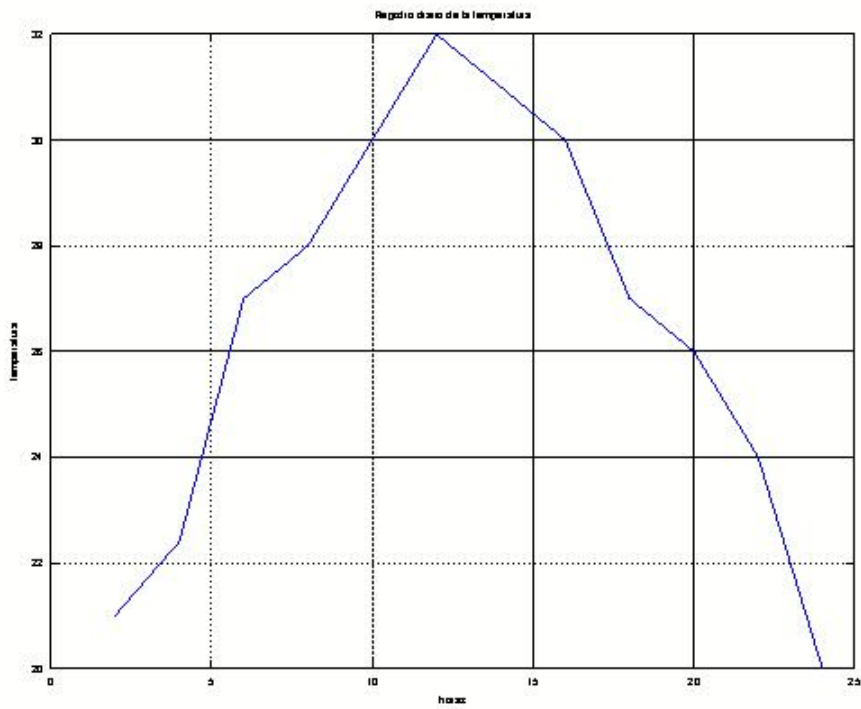


Figura 2.3: Grafica simple con nombre en los ejes, título. Ejemplo 2.3

Graficas de una Función

Para graficar funciones se toma en cuenta los siguientes pasos:

- Se define el rango de una de las coordenadas.
- Indicamos la función que se desea graficar.
- Gráfica la función (con el comando **plot**).
- Imprime los rótulos de los ejes (**xlabel**, **ylabel**).
- Puede agregar una malla con **grid**.
- También se tiene el comando **title** para agregar títulos al grafico.

Ejemplo 2.4. Calcular la función $y = \sin(x)e^{-0.4x}$, $x \in [0, 10]$

```
>>> x=0:0.05:10;
>>> y=sin(x).*exp(-0.4*x);
>>> plot(x,y)
>>> grid on
>>> title('grafico de ejemplo')
>>> xlabel('eje x')
>>> ylabel('eje y')
>>> print('-depsc', 'grafico10.eps')
>>>
```

También se pueden graficar dos funciones en un solo plano coordenado. El comando **hold on** es para permitir que se pueda sobrescribir otra grafica encima de la anterior.

Ejemplo 2.5. Gráficas de las funciones seno y coseno, en una sola figura.

```
>>> x=0:0.005:10;
>>> y=sin(x);
>>> plot(x,y)
```

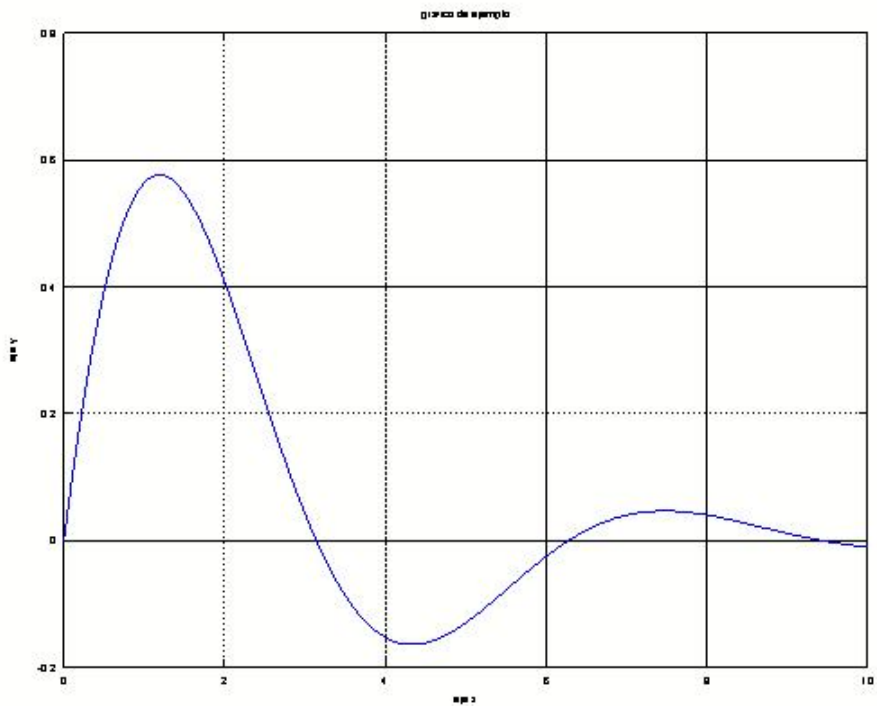


Figura 2.4: Grafica de una función

```

>>> hold on
>>> z=cos(x);
>>> plot(x,z,'--')
>>> grid on
>>> xlabel('eje x')
>>> ylabel('y( ___ )    z( --- )')
>>> title('Grafico de y=Seno(x)    y    z=Coseno(x)    en [0,10]')
>>> print('-depsc','grafico12.eps')
>>>

```

Además se le puede dar formato a la representación grafica. Ahora agregare-

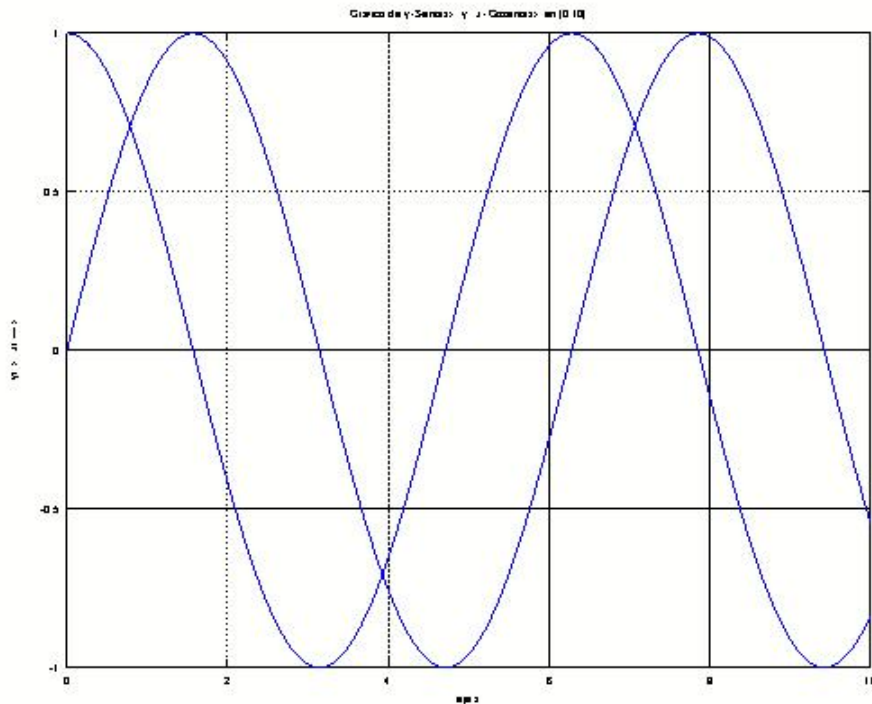


Figura 2.5: Grafica de las funciones seno y coseno

mos título, leyendas, nombres a los ejes, etc. Esto se puede hacer mediante comandos. Pero todos ellos se agregaran después de usar plot.

Comando	Formato	Explicación
xlabel	xlabel('texto')	Agrega una etiqueta al eje x.
ylabel	ylabel('texto')	Agrega una etiqueta al eje y.
title	title('texto')	Agrega una titulo al gráfico.
text	text(x,y,'texto')	Permite colocar una etiqueta
gtext	gtext('texto')	Etiqueta en lugar específico.
legend	legend('cadena1', ..., 'cadenan', pos)	Incluye leyenda.

2.1.2. Coordenadas Paramétricas

Ejemplo 2.6. `>>> t=0:0.01:8*pi;`
`>>> x=sin(3*t)+t/(20*pi);`
`>>> y=sin(2*t);`
`>>> plot(x,y)`
`>>> title('x(t)=sin(3t)+t/(20pi) y(t)=sin(2t), t en [0,8pi]')`
`>>> xlabel('x=x(t)')`
`>>> ylabel('y=y(t)')`

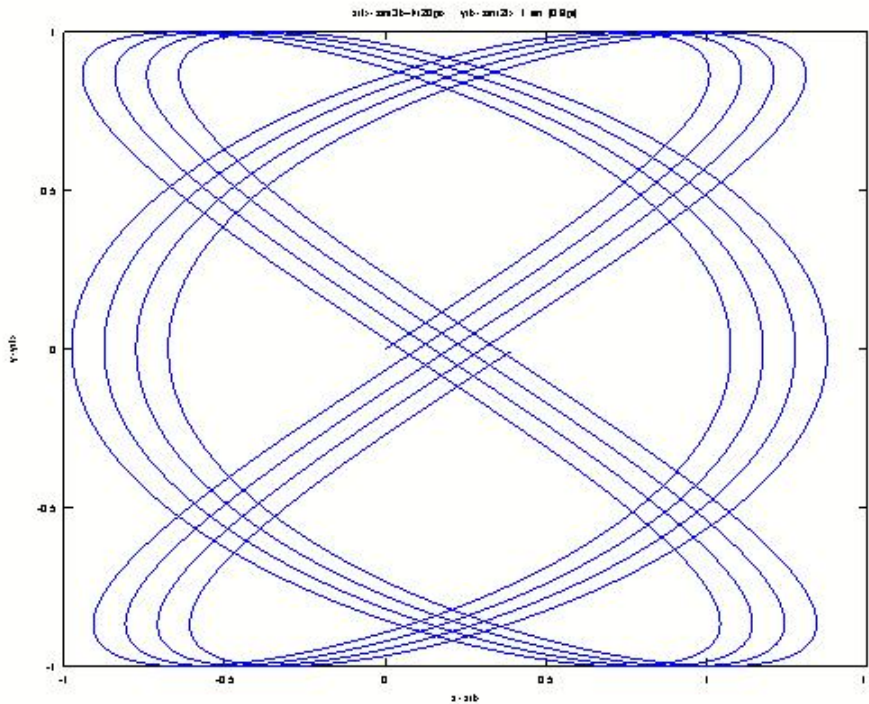


Figura 2.6: Gráfica en coordenadas paramétricas.

Ejemplo 2.7. `>>> t=0:pi/20:2*pi;`
`>>> x=(cos(t)).^3;y=(sin(t)).^3;`

```

>>> plot(x,y)
>>> grid
>>> title('x^2/3 + y^2/3 =1');
>>>

```

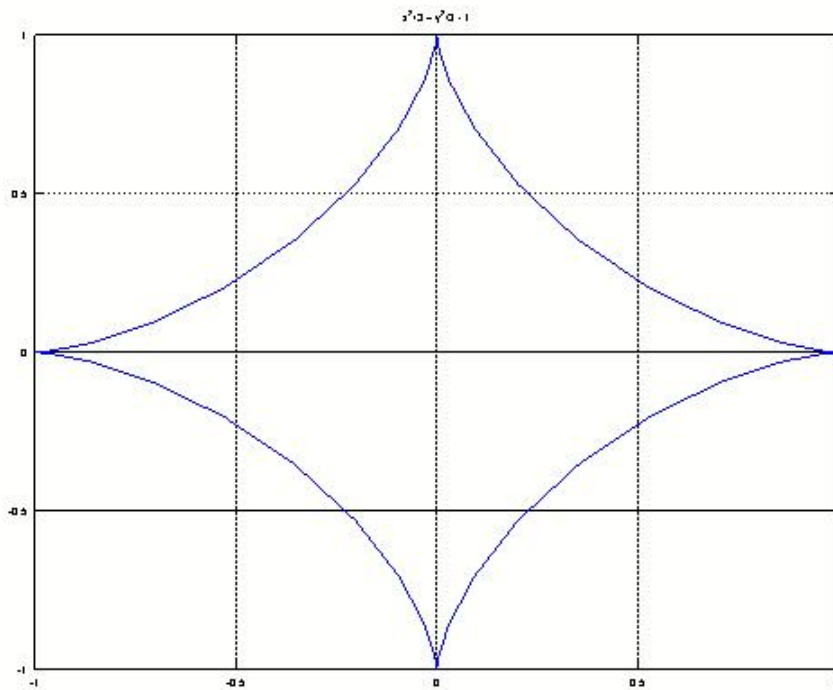


Figura 2.7: Gráfica en coordenadas paramétricas.

2.1.3. Coordenadas Polares

Ejemplo 2.8. Podemos graficar una función en coordenadas polares con el comando **polar**

```

>>> t=0:0.05:pi+0.01;
>>> r=6*cos(5*t);

```

```
>>> polar(t,r)
>>> title('Grafica en Coord. Polares')
>>> grid
>>>
```

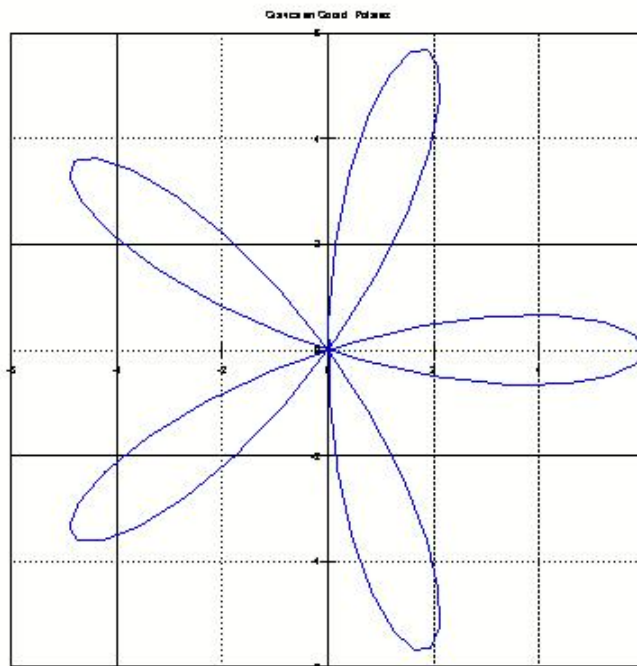


Figura 2.8: Grafica en coordenadas polares.

2.2. Gráficas en 3D

Los gráficos en 3D, son aquellas expresiones de la forma:

$$E(x, y, z) = 0, x \in D$$

Donde D es un cierto dominio en \mathbb{R}^3 , esta grafica puede ser una curva o una superficie.

Curvas Parametrizadas

Ahora vamos a graficar algunas curvas que deben estar parametrizadas, se hara con el comando de QtOctave: **plot3**.

Ejemplo 2.9. Hallar la gráfica de la $f(t) = (e^{0,2t} \text{sen}(0,8t), e^{0,2t} \text{cos}(0,8t), t)$ $t \in [0, 20]$

```
>>> clear,clf % borra las curvas graficadas y redibuja los ejes
>>> t = 0:0.1:20; % Indica el rango del parámetro t
>>> r = exp(0.2*t); % una parte de la función a graficar
>>> th=pi*t*0.8; % un cambio de variable
>>> z=t; % Indica la función de la coordenada x
>>> x=r.*sin(th); % Indica la función de la coordenada y
>>> y=r.*cos(th); % Indica la función de la coordenada z
>>> plot3(x,y,z) % Grafica la función
>>> xlabel('x');ylabel('y');zlabel('z');
>>>
```

Ejemplo 2.10. La posición de una partícula en el tiempo viene dada por:

$$x = (2 + 4 \cos(t))t, \quad y = (2 + 4 \text{sen}(t))t, \quad z = t^2 \quad t \in [0, 200]$$

```
>>> t=0:0.1:200;
>>> x=(2+cos(t)).*cos(t);
>>> y=(2+cos(t)).*sin(t);
```

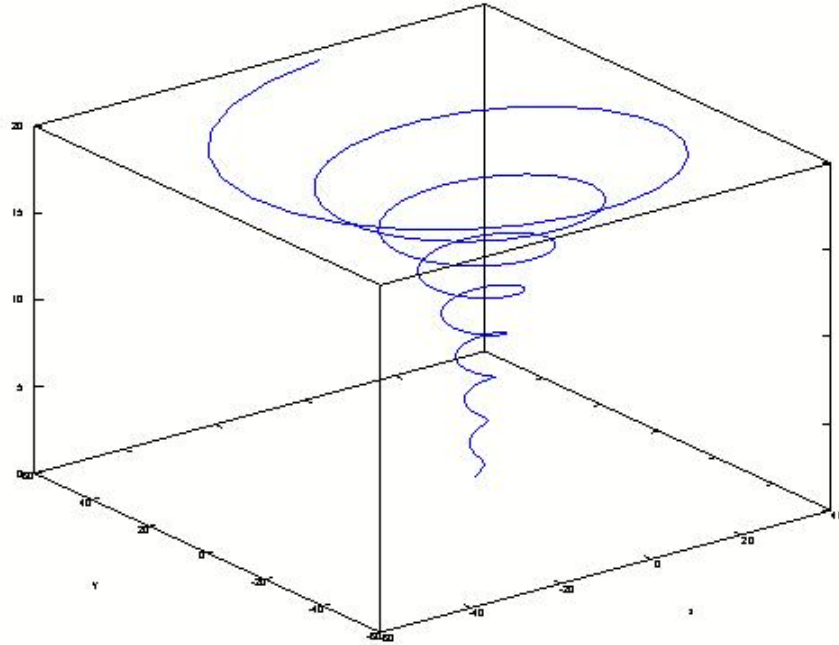


Figura 2.9: Grafica de $f(t) = (e^{0,2t} \text{sen}(0,8t), e^{0,2t} \text{cos}(0,8t), t)$ $t \in [0, 20]$

```
>>> z=t.^2;
>>> plot3(x,y,z,'r','LineWidth',1)
>>> grid on
>>> xlabel('x');
>>> ylabel('y');
>>> zlabel('z');
>>> print('-depsc','grafico14.eps')
>>>
```

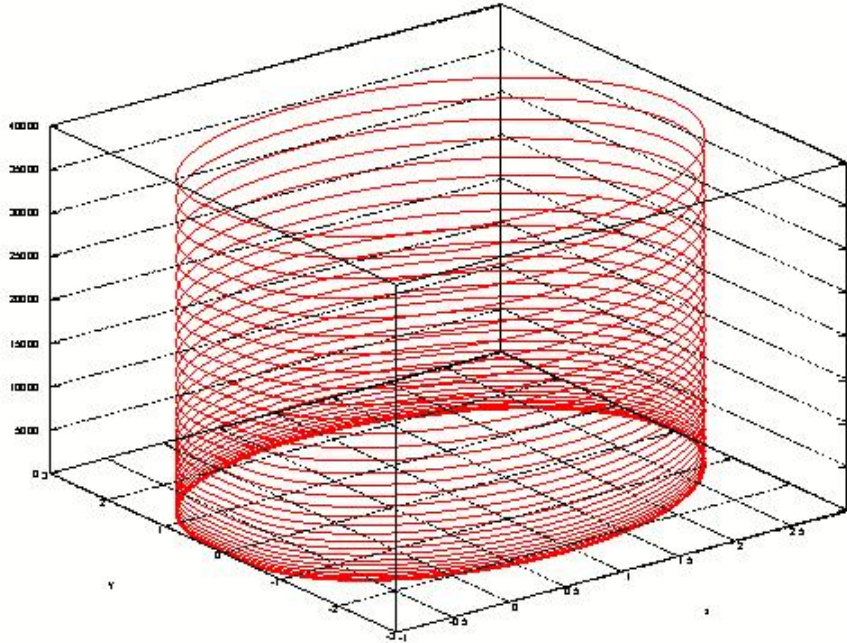



Figura 2.10: Grafica de la posición de una partícula.

Gráficos de Malla y Superficie

Ejemplo 2.11. Gráficar la siguiente función

$$z = \frac{xy^2}{x^2 + y^2}$$

```
>>> x=1:1:5;
>>> y=-4:1:3;
>>> [X,Y]=meshgrid(x,y)
X =
```

```

1  2  3  4  5
1  2  3  4  5
1  2  3  4  5
1  2  3  4  5
1  2  3  4  5
1  2  3  4  5
1  2  3  4  5
1  2  3  4  5

```

Y =

```

-4 -4 -4 -4 -4
-3 -3 -3 -3 -3
-2 -2 -2 -2 -2
-1 -1 -1 -1 -1
 0  0  0  0  0
 1  1  1  1  1
 2  2  2  2  2
 3  3  3  3  3

```

```
>>> z=X.*Y.^2./(X.^2+Y.^2)
```

z =

```

0.94118  1.60000  1.92000  2.00000  1.95122
0.90000  1.38462  1.50000  1.44000  1.32353
0.80000  1.00000  0.92308  0.80000  0.68966
0.50000  0.40000  0.30000  0.23529  0.19231

```

0.00000	0.00000	0.00000	0.00000	0.00000
0.50000	0.40000	0.30000	0.23529	0.19231
0.80000	1.00000	0.92308	0.80000	0.68966
0.90000	1.38462	1.50000	1.44000	1.32353

```
>>> surf(x,y,z)
>>> print('-depsc','grafico15.eps')
>>>
```

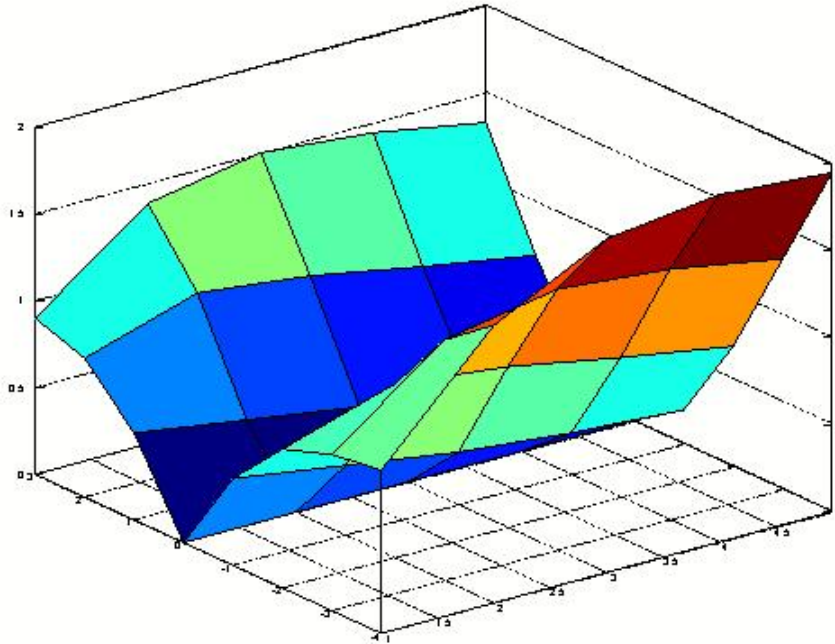


Figura 2.11: Grafica de $z = \frac{XY^2}{X^2+Y^2}$.

Ejemplo 2.12. Analizar y decir cuál es la superficie mostrada.

```
>>> x=-3:0.1:3;
>>> y=-3:0.1:3;
>>> [X,Y]=meshgrid(x,y);
>>> Z=(1.8.^(-1.5*sqrt(X.^2+Y.^2))).*sin(X).*cos(0.5*Y);
>>> mesh(X,Y,Z)
>>> grid on
>>> print('-depsc','grafico16.eps')
>>>
```

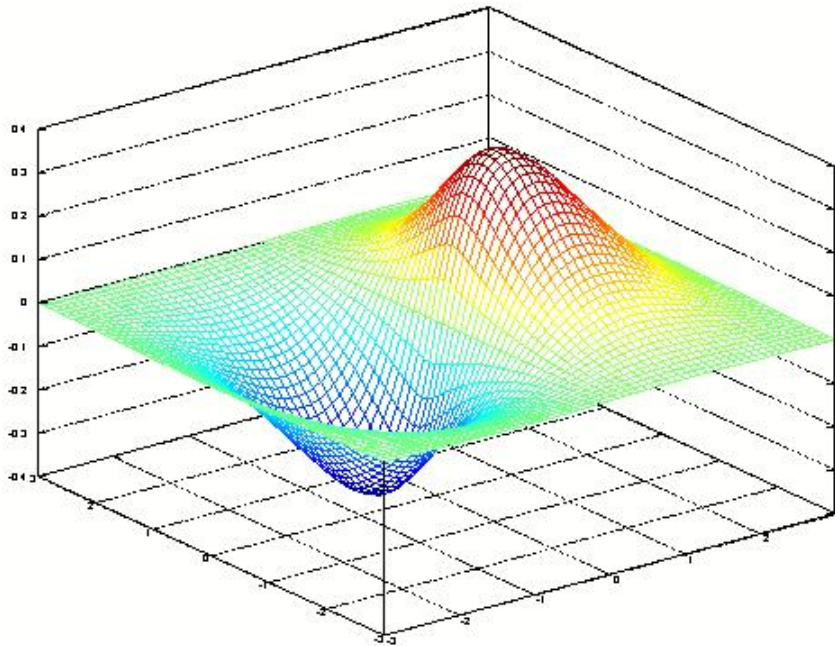


Figura 2.12: Gráfica de las curvas de nivel.

Ejemplo 2.13. Gráfico mostrando las curvas de nivel.

```
>>> x=-3:0.1:3;
>>> y=-3:0.1:3;
>>> [X,Y]=meshgrid(x,y);
>>> Z=(1.8.^(-1.5*sqrt(X.^2+Y.^2))).*sin(X).*cos(0.5*Y);
>>> contour(X,Y,Z,15) % el cuarto elemento es el numero de niveles
>>> xlabel('x');ylabel('y');zlabel('z');
>>> print('-depsc','grafico17.eps')
>>>
```

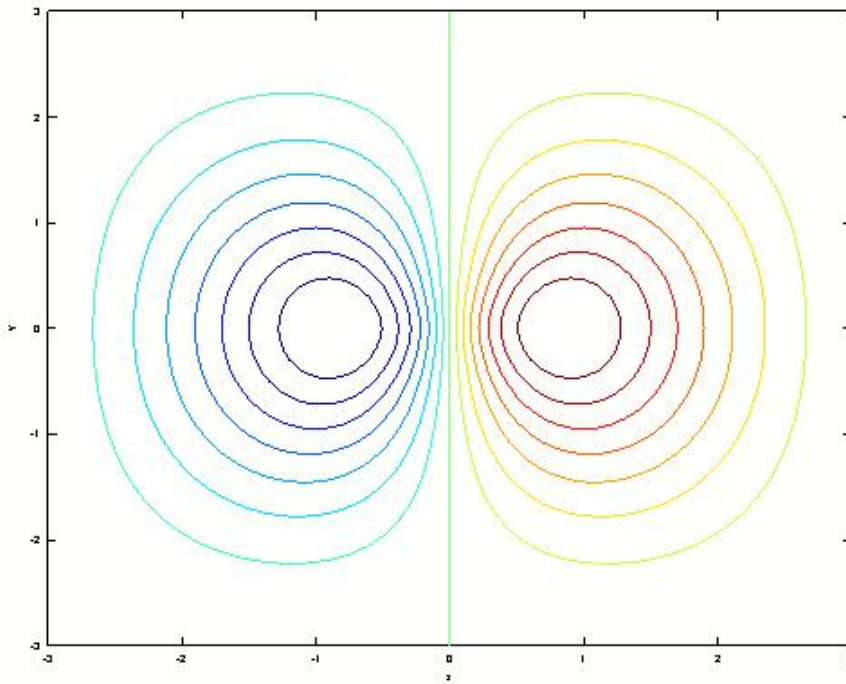


Figura 2.13: Gráfica de la superficie dada.

Gráficos Especiales

Algunas opciones diferentes para una mejor comprensión de los comandos teclear: *help nombre – comando*

Ejemplo 2.14. Graficando una esfera unitaria.

```
>>> [ X Y Z]=sphere(30); % esfera unitaria
>>> surf(X,Y,Z)
>>> print('-depsc','grafico18.eps')
>>>
```

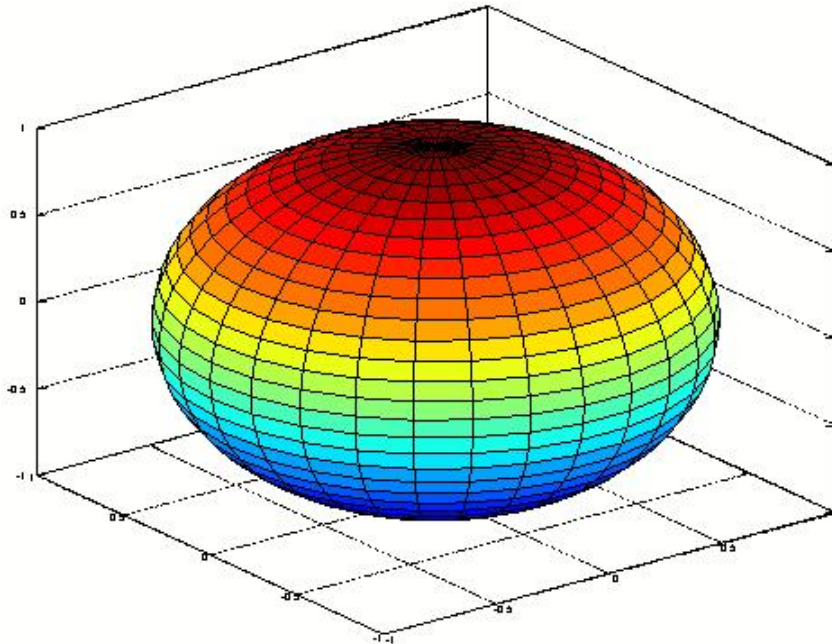


Figura 2.14: Gráfica de la esfera unitaria.

Capítulo 3:

Álgebra Lineal

3.1. Operaciones elementales de Matrices

Ejemplo 3.1. Realizar operaciones elementales de matrices (Suma, resta y multiplicación).

```
>>> A=[2 4 7 8;3 5 1 9;5 4 6 3]
```

A =

```
2  4  7  8
3  5  1  9
5  4  6  3
```

```
>>> B=[12 44 98 81;56 45 67 34;67 89 62 46]
```

B =

```
12  44  98  81
56  45  67  34
67  89  62  46
```

```
>>> A+B,A-B,B-A
```

```
ans =
```

```
14  48 105  89
59  50  68  43
72  93  68  49
```

```
ans =
```

```
-10 -40 -91 -73
-53 -40 -66 -25
-62 -85 -56 -43
```

```
ans =
```

```
10  40  91  73
53  40  66  25
62  85  56  43
```

```
>>> A+60
```

```
ans =
```

```
62  64  67  68
63  65  61  69
65  64  66  63
```



```
>>> B-50
```

```
ans =
```

```
-38  -6  48  31
     6  -5  17 -16
    17  39  12  -4
```

```
>>> A*B
```

```
error: operator *: nonconformant arguments (op1 is 3x4, op2 is 3x4)
```

```
>>> C=[15 16 43 76;98 74 62 56;45 67 34 67]
```

```
C =
```

```
15  16  43  76
98  74  62  56
45  67  34  67
```

```
>>> D=[12 44 98 81 92;56 45 67 34 24;67 89 62 46 45;45 89 32 56 67]
```

```
D =
```

```
12  44  98  81  92
56  45  67  34  24
67  89  62  46  45
45  89  32  56  67
```

```
>>> P=C*D
```

P =

```

    7377    11971    7640    7993    8791
11994    18144    20198    16442    17334
    9585    13984    13151    11239    11767

```

>>>

Ejemplo 3.2. Dar una matriz aleatoria de orden 4, luego hallar de esta matriz su determinante, rango, su matriz inversa y finalmente su matriz transpuesta.

```
>>> A=rand(4)    %Una matriz aleatoria
```

A =

```

    0.366828    0.247772    0.016047    0.609561
    0.564409    0.921166    0.103525    0.548230
    0.281829    0.690120    0.135361    0.870776
    0.391185    0.419536    0.010951    0.299457

```

```
>>> det(A)    %Determinante de la Matriz
```

ans = 0.0019939

```
>>> rank(A)    %Rango de la matriz A
```

ans = 4

```
>>> inv(A)    %Matriz inversa de A
```

```
ans =
```

```
    9.0229    9.9218   -7.4567  -14.8479
   -8.3685   -7.4792    5.5264   14.6570
   31.7508   43.7333  -23.6911  -75.8047
   -1.2236   -4.0820    2.8647    4.9732
```

```
>>> A'           %Matriz transpuesta de la matriz A
```

```
ans =
```

```
    0.366828    0.564409    0.281829    0.391185
    0.247772    0.921166    0.690120    0.419536
    0.016047    0.103525    0.135361    0.010951
    0.609561    0.548230    0.870776    0.299457
```

3.2. Valores y Vectores Propios de una Matriz

Ejemplo 3.3. Dar una matriz aleatoria de orden 3 y luego calcular:

- El polinomio característico de dicha matriz.
- Raíces del polinomio característico.
- Autovalores de la Matriz.
- Vectores Propios.
- Valores Propios.

```
>>> E=rand(3) % matriz cuadrada de numero aleatorios, de 3x3
E =

    0.305940    0.337815    0.040263
    0.540716    0.120102    0.356697
    0.504817    0.431840    0.419825

>>> c=poly(E) % polinomio caracteristico de la matriz
c =

    1.000000   -0.845868   -0.141416    0.040596

>>> roots (c) % raices del polinomio caracteristico
ans =

    0.94976
   -0.26512
    0.16123

>>> eig(E) %autovalores de E
ans =

    0.94976
    0.16123
   -0.26512

>>> [V,MD]=eig(E) % V son los vectores propios y Md es una matriz diagonal con
```

los valores propios

V =

```
-0.33549  -0.53886  0.49213
-0.54807  0.13167  -0.85276
-0.76620  0.83204  0.17494
```

MD =

Diagonal Matrix

```
0.94976      0      0
      0  0.16123      0
      0      0  -0.26512
```

>>>

3.3. Sistema de Ecuaciones Lineales

Ejemplo 3.4. Resolver el siguiente sistema de ecuaciones, por los siguientes métodos:

- Inversa de Una matriz.
- Usando la forma escalon reducida por la fila matriz.

$$\begin{cases} x - y - z = 2 \\ 3x - 3y + 2z = 16 \\ 2x - y + z = 9 \end{cases}$$

```
%----- PRIMERA FORMA DE RESOLVER -----  
%Daremos al sistema de ecuaciones en forma de matrices.  
%Luego ingresar dichas matrices obtenidas.  
  
>>> D=[1 -1 -1;3 -3 2;2 -1 1] % Matriz de coeficientes.  
D =  
  
    1    -1    -1  
    3    -3     2  
    2    -1     1  
  
>>> d=[2;16;9]  
d =  
  
     2  
    16  
     9  
  
>>> sol=D\d  
sol =  
  
    3.00000  
   -1.00000  
    2.00000
```

```
>>>
```

```
%-----SEGUNDA FORMA DE RESOLVER-----
```

```
>>> Dd=[1 -1 -1 2;3 -3 2 16;2 -1 1 9]
```

```
Dd =
```

```

1   -1   -1   2
3   -3    2  16
2   -1    1   9
```

```
>>> rref(Dd)
```

```
ans =
```

```

1.00000  0.00000  0.00000  3.00000
0.00000  1.00000  0.00000 -1.00000
0.00000  0.00000  1.00000  2.00000
```

```
>>>
```

3.4. Espacios Vectoriales y Aplicaciones Lineales

Ejemplo 3.5. Dar la matriz mágica de orden ocho (averiguar por que se dice mágica). A partir de ella encontrar su rango, espacio nulo, espacio nulo con elementos racionales, base ortogonal para el rango de A y finalmente base ortogonal para el espacio nulo de A.

```
>>> A=magic(8)
```

```
A =
```

```
64    2    3   61   60    6    7   57
 9   55   54   12   13   51   50   16
17   47   46   20   21   43   42   24
40   26   27   37   36   30   31   33
32   34   35   29   28   38   39   25
41   23   22   44   45   19   18   48
49   15   14   52   53   11   10   56
 8   58   59    5    4   62   63    1
```

```
>>> rank(A)           %rango de A
```

```
ans = 3
```

```
>>> N=null(A)         % espacio nulo de A
```

```
N =
```

```
0.0472781    0.4166814   -0.5226682   -0.0674937   -0.0688280
0.7254081    0.0880443   -0.2515023   -0.0535303   -0.0332098
-0.6082948   -0.2148316   -0.4649440   -0.1826477    0.1752721
0.1211315   -0.7702304   -0.0077580    0.0786931   -0.3603708
0.0838214   -0.0248600    0.3538616   -0.4204609    0.6590889
-0.1269242    0.1803730    0.4891634   -0.4208238   -0.4810519
0.0098109   -0.0535857    0.2272830    0.6570018    0.3389896
-0.2522310    0.3784091    0.1765645    0.4092614   -0.2298901
```

```
>>> NR=null(A,'r')    % espacio nulo de A, con elementos racionales
```


NR =

```

0.5400617    0.0472781    0.4166814   -0.5226682   -0.0674937   -0.0688280
-0.3857584    0.7254081    0.0880443   -0.2515023   -0.0535303   -0.0332098
-0.2314550   -0.6082948   -0.2148316   -0.4649440   -0.1826477    0.1752721
0.0771517    0.1211315   -0.7702304   -0.0077580    0.0786931   -0.3603708
-0.0771517    0.0838214   -0.0248600    0.3538616   -0.4204609    0.6590889
0.2314550   -0.1269242    0.1803730    0.4891634   -0.4208238   -0.4810519
0.3857584    0.0098109   -0.0535857    0.2272830    0.6570018    0.3389896
-0.5400617   -0.2522310    0.3784091    0.1765645    0.4092614   -0.2298901

```

>>> Q=orth(A) % base ortogonal para el rango de A

Q =

```

0.353553   -0.540062   -0.353553
0.353553    0.385758    0.353553
0.353553    0.231455    0.353553
0.353553   -0.077152   -0.353553
0.353553    0.077152   -0.353553
0.353553   -0.231455    0.353553
0.353553   -0.385758    0.353553
0.353553    0.540062   -0.353553

```

>>> NRort=orth(A) % base ortogonal para el espacio nulo de A

NRort =

```

0.353553   -0.540062   -0.353553

```

0.353553	0.385758	0.353553
0.353553	0.231455	0.353553
0.353553	-0.077152	-0.353553
0.353553	0.077152	-0.353553
0.353553	-0.231455	0.353553
0.353553	-0.385758	0.353553
0.353553	0.540062	-0.353553

>>>

Capítulo 4:

Polinomios

Un polinomio es una expresión de la forma:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

Polinomio	En QtOctave
$p(x) = 2x - 5$	p = [2 -5]
$q(x) = 6x^3 - 7x^2 + 2$	q = [6 -7 0 2]

Ejemplo 4.1. Dado el polinomio

$$x^5 - 12x^4 + 40,59x^3 - 17,015x^2 - 71,95x + 35,88$$

Hallar:

- Hallar $p(5)$, $p(9)$ y $p(-1)$.
- Representar graficamente el polinomio $p(x)$ para $-1,5 \leq x \leq 6,7$
- Calcular las raices del polinomio.

```
>>> p=[1 -12.1 40.59 -17.015 -71.95 35.88]
p =    1.0000   -12.1000   40.5900  -17.0150  -71.9500   35.8800
>>> polyval(p,5)                                % hallando p(5)
```

```

ans = -112.9950
>>> polyval(p,9) % hallando p(9)
ans = 7.2611e+003
>>> polyval(p,-1) % hallando p(-1)
ans = 37.1250
>>> x=-1.5:0.1:6.7;
>>> y=polyval(p,x);
>>> plot(x,y)
>>> grid
>>> r=roots(p) % raices de p(x)
r = 6.5000
    4.0000
    2.3000
   -1.2000
    0.5000
>>>

```

Ejemplo 4.2. Dado el siguiente polinomio:

$$p(x) = x^4 - 1$$

Encontrar las raíces del polinomio.

```

>>> p=[1 0 0 0 -1]
p = 1 0 0 0 -1
>>> r=roots(p)
r = -1.0000
   -0.0000 + 1.0000i
   -0.0000 - 1.0000i

```

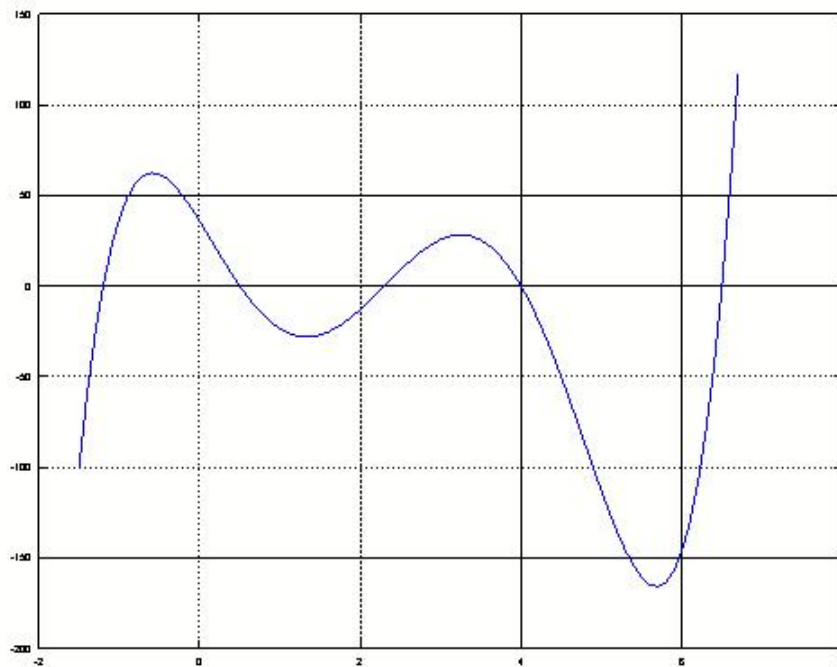


Figura 4.1: Gráfica del polinomio

$$x^5 - 12x^4 + 40,59x^3 - 17,015x^2 - 71,95x + 35,88.$$

1.0000

>>>

Ejemplo 4.3. Sea $p(x) = x^4 + 6x^3 - 5x + 7$ y $q(x) = 3x^2 + 8x - 5$, hallar las siguientes operaciones:

- Sumar los polinomios.
- Multiplicar los polinomios.
- Dividir.
- Hallar la derivada del polinomio.

```
>>> p=[1 6 0 -5 7]
p =

    1     6     0    -5     7

>>> q=[0 0 3 8 -5]
q =

    0     0     3     8    -5

>>> p+q                                %Suma de Polinomios.
ans =

    1     6     3     3     2

>>> c=conv(p,q)                         %Producto de Polinomios.
c =

    0     0     3    26    43   -45   -19    81   -35

>>> k=polyder(p)                        %Derivada de un Polinomio.
k =

    4    18     0    -5

>>>
```

Bibliografía

- [1] P, Long. *Manual. Introduction to Octave*. Departamento de Ingeniería de la Universidad de Cambridge (Inglés). Accedido el 20 de agosto 2014. <http://www-mdp.eng.cam.ac.uk/CD/engapps/octave/octavetut.pdf>
- [2] *Apuntes y ejercicios. Página de la asignatura Informática básica de la Universitat Jaume I de España*. Accedido el 15 de julio 2014. <https://www.unoweb-s.uji.es/0304/N13/ficheros0/>
- [3] A, Malca y L, Valdivia. *Curso de MatLab para matemáticas*. UNIVERSIDAD PEDRO RUIZ GALLO 2013.
- [4] S, Blanco. *Manual Básico Ubuntu GNU/Linux*. Versión (BETA) Breezy 2015.
- [5] M, Gomez et al. *Manual Básico de Octave y QtOctave. Métodos Matemáticos para las ciencias de la Salud*, 2011.
- [6] M, Storti. *Introducción a Octave*, 2009.
- [7] D, Hernandez. *Introducción a Gnu-Octave*. UNIVERSIDAD DE LOS ANDES, Venezuela 2007.
- [8] *Página web del proyecto Octave (Inglés)*. Accedido el 30 de agosto 2014. <http://www.gnu.org/software/octave/>