UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE COMPUTACIÓN

PROPUESTA DE CAMBIO EN SOFTWARE LIBRE PARA SU IMPLEMENTACIÓN TECNOLÓGICA EN LAS ASIGNATURAS DE MATEMÁTICAS DE LA UNIVERSIDAD DON BOSCO: PROPUESTA OCTAVE

TRABAJO DE GRADUACIÓN PARA OPTAR AL GRADO DE:

INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTADO POR:

LAURA SOFÍA CÁCERES HERNÁNDEZ.

ASESOR:

ING. CARLOS JOSÉ TEJADA FUENTES.

AGOSTO DE 2007

SAN SALVADOR, EL SALVADOR, CENTRO AMÉRICA.

ÍNDICE.

Contenido. ***Pág.***

Introducción..... 7

Capitulo I Definición del Tema

1.1 Antecedentes..... 8
1.2 Importancia de la investigación..... 10
1.2.1 Planteamiento del problema..... 10
1.2.2 Definición del tema..... 11
1.2.3 Justificación..... 11
1.3 Objetivos..... 14
1.3.1 Objetivo General..... 14
1.3.2 Objetivos Específicos..... 14
1.4 Alcances..... 15
1.5 Limitaciones..... 15
1.6 Delimitaciones..... 15
1.7 Proyección Social..... 16

Capitulo II Marco Teorico

2.1 El software libre..... 17
2.2 Categorías de software libre..... 19
2.2.1 Software de Fuente Abierta "Open Source"..... 19
2.2.2 Software de Dominio Público..... 19
2.2.3 Software protegido con copyleft..... 19
2.2.3.1 Copyleft y la GNU GPL..... 20
2.2.3.2 Software libre no protegido con copyleft..... 20
2.2.4 Software abarcado por GPL 21
2.2.5 Software GNU..... 21
2.2.6 Software Semilibre..... 21
2.2.7 Software Propietario..... 22
2.2.8 Freeware 22
2.2.9 Shareware..... 22
2.2.10 Software Comercial..... 23
2.3 Software libre famoso..... 24
2.4 Ventajas en términos de Costos..... 25
2.5 ¿Qué es Octave?..... 25

2.6 Detalles técnicos de Octave..... 26
2.7 Características de Octave..... 26
2.8 Instalación de Octave..... 27

Capitulo III Plan de Migración
3.1 Objetivo General..... 28
3.2 Objetivos Específicos..... 28
3.3 Plan de Migración..... 28
3.3.1 Fase 1: Sensibilización institucional respecto a la Migración a Software..... 29
3.3.1.1 Ventajas de la migración a software libre..... 29
3.3.2 Fase 2: Alternativas de migración..... 30
3.3.3 Fase 3: Instalación y operación del software a migrar..... 31
3.3.3.1 Pruebas de operatividad del software migrado..... 31
3.3.4 Fase 4: Capacitación a los Usuarios..... 32

Capitulo IV Fase 1: Sensibilización institucional respecto a la Migración a Software
4.1 Introducción..... 33
4.2 Costo - beneficio..... 34
4.3 Desarrollo histórico de las asignaturas de ingeniería..... 35

Capitulo V Fase 2: Alternativas de migración.
5.1 Requisitos para las herramientas matemáticas..... 38
5.2 Censo y análisis previo de las herramientas disponibles..... 39
5.2.1 Programas de cálculo simbólico,cálculo numérico y matricial..... 39
5.2.1.1 Axiom..... 39
5.2.1.2 Yacas..... 41
5.2.1.3 Euler..... 42
5.2.1.4 Octave..... 43
5.2.1.5 Scilab..... 44
5.3 Elección del software libre para implementarlo en el plan de migración..... 45

Capitulo VI Fase 3: Instalación y operación de Octave
6.1 Instalación de Octave..... 46
6.1.1 Versión para sistemas operativos Microsoft..... 46
6.1.2 Toolbox de Control de Sistemas de Octave (OCST)..... 46
6.1.3 Instalación de Octave bajo Cygwin en Windows..... 46
6.1.4 Instalación de octave+forge en Windows (GNU Octave 2.1.73)..... 53
6.2 Operación de Octave..... 57

6.2.1 Matemática Sencilla..... 57
6.2.2 Operaciones con Matrices y Vectores..... 57
6.2.2.1 Definición de Matrices..... 58
6.2.2.2 Operaciones con Matrices..... 61
6.2.3 Gráficos Científicos..... 64
6.2.3.1 Gráficas en 2D..... 64
6.2.3.1.1 La función plot..... 66
6.2.3.1.2 Estilos de línea y marcadores en la función plot..... 72
6.2.3.1.3 Añadir líneas a un gráfico ya existente..... 74
6.2.3.1.3 Comando subplot..... 76
6.2.3.1.4 Control de los ejes..... 77
6.2.3.2 Gráficas en 3D..... 82
6.2.3.2.1 Graficación de superficies con coordenadas de enteros..... 83
6.2.3.2.2 Graficación de líneas con parámetros..... 83
6.2.3.2.3 Dibujo de mallados: funciones mesh, meshgrid y meshdom..... 86
6.2.3.2.4 Dibujo de líneas de contorno: funcion contour()..... 89

Capitulo VI Comparación de las herramientas Matlab y octave
7.1 Aspectos de evaluación..... 91
7.2 Comparación económica..... 93

Conclusiones..... 97

Glosario..... 98

Bibliografía..... 103

Anexos..... 105

ÍNDICE DE FIGURAS

Pág

Figura 2.1 Sitio Web nos permite bajar el instalador de Octave para Windows..... 27
Figura 5.1 Axiom, captura de pantalla..... 40
Figura 5.2 YACAS, captura de pantalla..... 42
Figura 5.3 EULER, captura de pantalla..... 43
Figura 5.4 Octave, captura de pantalla..... 44
Figura 5.5 Scilab, captura de pantalla..... 44
Figura 6.1 Pantalla de configuración de la instalación..... 48
Figura 6.2 Pantalla de selección del tipo de instalación..... 48
Figura 6.3 Pantalla en la cual se selecciona el directorio en el que se instalará Cygwin... 49
Figura 6.4 Pantalla en la cual se selecciona el directorio en el que se quiere guardar los archivos temporales que vaya descargando..... 50
Figura 6.5 Pantalla de selección de componentes a instalar..... 50
Figura 6.6 Archivo ejecutable de GNU Octave 2.1.73..... 53
Figura 6.7 Pantalla de selección de idioma para la instalación de GNU Octave 2.1.73..... 54
Figura 6.8 Pantalla de licencia de GNU Octave 2.1.73..... 54
Figura 6.9 Pantalla de selección de componentes a instalar de GNU Octave 2.1.73..... 55
Figura 6.10 Pantalla en la cual se selecciona el directorio en el que se instalará GNU Octave 2.1.73..... 55
Figura 6.11 Pantalla de proceso de instalación de componentes de GNU Octave 2.1.73. 56
Figura 6.12 Finalización del proceso de instalación de GNU Octave 2.1.73..... 56
Figura 6.13 Entorno de trabajo de Octave de GNU Octave 2.1.73..... 57

ÍNDICE DE TABLAS

Pág.

Tabla 2.1 Software Libre..... 24
Tabla 3.1 Fases del Plan de Migración..... 29
Tabla 6.1. Operaciones básicas en Octave..... 57
Tabla 6.2 Estilos de marcadores en la función plot..... 73
Tabla 6.3 Estilos de líneas en la función plot..... 73
Tabla 7.1 Comparación entre las herramientas Matlab y Octave..... 93
Tabla 7.2 Puntos fuertes y débiles de los programas analizados..... 93
Tabla 7.3 Costos de renovación del servicio de mantenimiento..... 94
Tabla 7.4 Costo de los productos actualizados..... 95

INTRODUCCIÓN.

En todos los niveles educativos, en particular en el universitario, el uso de las tecnologías de la información y de la comunicación aumenta día a día y en todos los ámbitos. En particular, el uso del ordenador en las aulas de asignaturas de matemáticas está abriendo las puertas en nuestros días a nuevas oportunidades y a terrenos inexplorados.

El software con licencia libre, permite y de hecho fomenta el disponer de varias herramientas, por lo que se han tomado en cuenta las diferentes alternativas en el área de matemáticas, evaluando los puntos fuertes y sus debilidades.

El propósito principal de este trabajo es abordar una propuesta que involucre un beneficio del uso del software Octave para la Universidad Don Bosco como una herramienta de apoyo al proceso de enseñanza-aprendizaje en las asignaturas de matemáticas que se lleva a cabo en la misma.

El contenido para la migración a software libre planea la difusión del plan de migración en la institución y la evaluación de la herramienta a migrar.

Capítulo I
Definición del Tema.

1.1 ANTECEDENTES.

El proyecto GNU fue concebido en 1983 como una forma de devolver el espíritu cooperativo que prevalecía en la comunidad computacional en días pasados hacer la cooperación posible al eliminar los obstáculos impuestos por los dueños de software privativo.

En 1971, cuando Richard Stallman comenzó su carrera en el MIT (Instituto de Tecnología de Massachusetts), trabajó en un grupo que usaba software libre exclusivamente. Incluso compañías informáticas frecuentemente distribuían software libre. Los programadores eran libres de cooperar unos con otros, y frecuentemente lo hacían.

En los años 80, casi todo el software era propietario, lo cual significa que tenía dueños que prohibían e impedían la cooperación entre usuarios. Esto hizo necesario el Proyecto GNU.

Octave o GNU Octave es un software libre para realizar cálculos numéricos. Como indica su nombre es parte del proyecto GNU.

Octave fue creado en 1988 por John W. Eaton como soporte a un libro de texto sobre ingeniería química en la Universidad de Texas con el objetivo de crear un lenguaje de cálculo numérico fácil de usar, interactivo y con una amplia comunidad de usuarios. La primera versión alpha fue lanzada el 4 de enero de 1993. El 17 de febrero 1994 con la versión 1.0, obtuvo su madurez para la enseñanza, la investigación y en la empresa. Sus autores lo definen como un lenguaje de alto nivel inspirado en un software comercial llamado MATLAB (MATrix LABoratory).

A partir de ese momento, las contribuciones de los usuarios han hecho evolucionar este software y han añadido librerías y funcionalidades. Ahora, las aplicaciones de Octave ya no se limitan al simple trabajo con matrices y vectores, como una mera calculadora, sino que ahora aparte de aplicaciones puramente matemáticas o numéricas, es válido para otros campos de ciencias e ingenierías.

El nombre de Octave surge del nombre de un profesor conocido por sus buenas aproximaciones por medio de cálculos mentales a problemas numéricos.

1.2 IMPORTANCIA DE LA INVESTIGACIÓN.

1.2.1 Planteamiento del problema.

El uso de los productos de software de MathWorks Matlab, con los que dispone la Universidad Don Bosco está orientado a su aprovechamiento en las áreas de docencia e investigación, por parte del personal docente y del alumnado. Matlab sirve como herramienta matemática.

Aprovechar las características y capacidades de los productos de software para los fines de docencia e investigación, es uno de los objetivos primarios de la Universidad Don Bosco.

La Universidad cuenta con las licencias de los productos de MathWorks, que la institución puede utilizar por un período de tiempo indefinido. Sin embargo, y principalmente en el ámbito informático, la tecnología se encuentra en constante evolución, por lo que los productos de software tarde o temprano se vuelven obsoletos, en el sentido de que los beneficios que producen se ven limitados ya sea porque sus resultados son incompatibles tecnológicamente con otros elementos de software o de hardware, o porque las tecnologías emergentes han reducido la aplicabilidad y/o la importancia de los mismos. Esta evolución tecnológica es más crítica para los productos de MathWorks, puesto que estas herramientas representan los últimos avances científicos en los diversos ámbitos tecnológicos y matemáticos. Ejemplos claros de la evolución de los sistemas informáticos son los sistemas operativos y aplicaciones de Microsoft. La mejor manera de proteger una inversión de software de este tipo es a través de la actualización constante de los productos. Es por ello que la Universidad Don Bosco busca alternativas que suplan las necesidades, sin que ésta incurra en una inversión elevada.

1.2.2 Definición del tema.

"PROPUESTA DE CAMBIO EN SOFTWARE LIBRE PARA LA IMPLEMENTACIÓN TECNOLÓGICA EN LAS ASIGNATURAS DE MATEMÁTICAS DE LA UNIVERSIDAD DON BOSCO: PROPUESTA OCTAVE"

El proyecto consiste en presentar una alternativa de cambio para la implementación de software libre en las asignaturas de matemáticas en la Universidad Don Bosco y presentar una guía de cómo lograr una integración satisfactoria sin causar un impacto negativo en los usuarios.

1.2.3 Justificación.

Esta propuesta está destinada a propiciar un mejor aprovechamiento del software utilizado por la Universidad Don Bosco, orientando su uso a las actividades de docencia e investigación, tanto para el personal docente como para el alumnado.
Dicho aprovechamiento se refiere no sólo al uso de los productos, sino además a la forma en la que se accede a su uso por parte de los interesados. Por otro lado, se analizan los beneficios de actualización de los productos MathWorks Matlab a Octave o GNU Octave que es un software libre. Es importante mencionar que Octave puede desempeñar las mismas funciones que Matlab.

Los beneficios logrados al usar software libre en el ambiente académico universitario superan los beneficios de usar software propietario.

A continuación se describen algunas de las ventajas de utilizar software libre en la preparación de futuros profesionales, desde un punto de vista académico:

- Gracias a las cuatro libertades sobre las cuales se fundamente el software libre y en particular, la libertad de estudiar el software, se tiene acceso al código fuente de una gran variedad de programas y aprender de esa fuente inmensa de conocimiento. Así como los estudiantes de literatura deben leer montañas de obras, y los abogados en su preparación deben leer muchos libros de

legislación, de igual forma los ingenieros de sistemas deben leer muchas líneas de código de programas para lograr una formación aceptable.

- Ante la rápida evolución tecnológica que estamos viviendo se ha llegado a la conclusión que la mejor herencia que la Universidad puede dejar es enseñar cómo funcionan las cosas, de tal forma que con unas sólidas bases se puedan comprender rápidamente las nuevas tecnologías que vayan apareciendo. Esto se puede hacer solamente con software libre, ya que con software propietario aspiramos, como máximo, a ser tan sólo expertos en el manejo de unas herramientas ignorando por completo cómo funcionan por dentro. Con software libre se tiene la valiosa ventaja de estudiar por dentro un motor de base de datos, un navegador Web, un compilador, un graficador, un sistema operativo, entre otros.

- Muchas personas tienen en sus equipos software propietario ilegal, quizá por que no tienen dinero suficiente para comprarlo o por ignorancia, ciertamente están cometiendo un delito. La solución ante este problema es utilizar software libre.

- Trabajar con software libre disminuye la brecha tecnológica entre los países del tercer y primer mundo. Los estudiantes pueden prepararse estudiando y participando de proyectos de software reales, compuestos por equipos de trabajo brillantes, distribuidos geográficamente por todo el mundo, y mejor aún, trabajar en nuestros propios requerimientos tecnológicos. Lo peor que nos puede pasar es convertirnos en consumidores de tecnologías de países avanzados porque esta situación implica estar condenados al sometimiento.

- Cada vez son más las empresas e instituciones que se migran al software libre, guiados inicialmente por el estímulo de ahorrar dinero (aunque luego se dan cuenta de otras características como calidad, desempeño y seguridad), en

licencias costosas, esto hace; que el mercado laboral requiera ingenieros y técnicos con conocimientos en estas tecnologías.

- Las universidades, anualmente, tienen que renovar los centros de cómputo con casas desarrolladoras como Microsoft, para poder utilizar sus productos. Al adoptar software libre, se ahorran miles de dólares, que pueden ser invertidos en otras áreas como la ampliación de salas de cómputo, dotación de bibliotecas, ampliación de planta física, etc.

- Al egresar de las universidades, los ingenieros pueden fácilmente constituir empresas porque con software libre no necesitan capital inmenso en licencias de paquetes de software, tan solo necesitan, su iniciativa e intelecto.

Con lo anterior no se afirma que sea conveniente migrarse completamente al software libre, pues el mercado laboral está rodeado de software propietario, lo importante es que los futuros ingenieros salgan capacitados en los dos ámbitos de trabajo y al momento de asesorar una empresa tengan ambas opciones, comparar tecnologías, sus ventajas, desventajas y brindar la mejor elección.

1.3 OBJETIVOS.

1.3.1 Objetivo General.

Realizar una propuesta de cambio en términos técnicos, académicos y económicos respecto al uso del software Octave en la Universidad Don Bosco.

1.3.2 Objetivos Específicos.

- Realizar un estudio de factibilidad con respecto al uso del software Octave en la Universidad Don Bosco.

- Presentar el estudio como propuesta para evaluar el cambio de aplicaciones propietarias hacia aplicaciones gratuitas.

- Desarrollar las guías de laboratorio para la asignatura de matemáticas I utilizando Octave.

1.4 ALCANCES.

- El estudio permitirá establecer una alternativa de cambio en software libre para las asignaturas de matemáticas.

- Este trabajo sirva como referencia para otros estudios en diferentes tipos de aplicaciones, principalmente en inversión de software libre.

- Promocionar el software libre, de tal forma que éste sea de beneficio para no incurrir en piratería.

- Realizar una comparación entre Octave y Matlab.

- Desarrollo de guías de laboratorio para la asignatura de matemáticas I.

1.5 LIMITACIONES.

- La falta de interés entre las partes involucradas en este estudio, podría limitar la obtención de información o propuestas de implementación.

1.6 DELIMITACIONES.

- El estudio será realizado solo para las asignaturas de matemáticas de la Universidad Don Bosco.

- La propuesta técnica se hará basándose en el hardware actual.

1.7 PROYECCIÓN SOCIAL.

La razón primordial de este trabajo es una alternativa de cambio en software libre. Esta alternativa se aplicará a las asignaturas de matemáticas, dependencia del Departamento de Ciencias Básicas de la Universidad Don Bosco.

Por medio del desarrollo de un estudio de esta naturaleza se crean incentivos dentro de la Universidad para la enseñanza de este tipo de investigaciones de forma más explicita, al mismo tiempo se cubre la necesidad de conocer el estado en que se encuentra el Laboratorio de Matemática y Simulación, así como su desempeño. Esto dará la pauta para que se tome en cuenta la implementación de software libre en las diferentes áreas de enseñanza-aprendizaje dentro de la Universidad, de tal forma que no haya necesidad de invertir demasiado en licencias propietarias pudiendo cubrir las mismas necesidades con software libre y a un bajo costo.

El uso de programas propietarios en nuestro país, tiende a fomentar la piratería entre los estudiantes universitarios; esto porque carecen de fondos para pagar las licencias correspondientes. El software libre es una forma de disminuir la piratería y la preocupación de inversión financiera en licencias de software.
En conclusión, el beneficio de un proyecto de esta naturaleza es para el sector educativo representado por la Universidad Don Bosco.

Capítulo II
MarcoTeórico.

2.1 El software libre.

El Software Libre es también conocido como Open Source software o Free Software.
El Software Libre tiene las siguientes características:

- Todo el mundo tiene derecho de usarlo, sin ninguna restricción.
- Todo el mundo tiene derecho a acceder a su diseño y aprender de él. Es como obtener las instrucciones para construir un carro.
- Todo el mundo tiene derecho de modificarlo: si el software tiene limitaciones, o no es adecuado para una tarea, es posible adaptarlo a sus necesidades específicas y redistribuirlo.
- No tiene un costo asociado (es gratuito).

Los derechos mencionados anteriormente tienen una serie de efectos colaterales:

- Tiende a ser muy eficiente (porque mucha gente lo optimiza, lo mejora).
- Tiende a ser muy robusto (mucha gente puede arreglarlo, no solamente el creador o la compañía que lo produce). Mucha gente tiende a contribuir, porque es del interés de todos mejorar esta base común.
- Tiende a ser muy diverso: la gente que contribuye tiene muchas necesidades diferentes y esto hace que el software esté adaptado a una cantidad más grande de problemas.

Estos derechos típicamente no están disponibles con el software propietario. Usualmente en el software propietario hay que pagar una licencia de uso al creador (como el pago de derechos por el uso de una patente) y está uno sujeto a las condiciones de uso del fabricante. Típicamente estas condiciones no otorgan ningún derecho al usuario final.

El éxito del Software Libre se debe en su mayor parte a Internet, porque esto ha permitido que las personas interesadas en los varios componentes del software libre se pongan en contacto con otras. Internet de esta manera actúa como un distribuidor y catalizador que acelera el desarrollo y el conocimiento en áreas muy específicas.

Hay diferentes motivaciones que impulsan a los contribuidores y desarrolladores a trabajar en el software libre, las más importantes son:

- El deseo de crear software más robusto.
- La posibilidad de estar en control del software. Esto es importante para aplicaciones de misión crítica donde es totalmente imperante tener un control total sobre posibles problemas en cualquier punto.
- Crear aplicaciones de bajo costo.
- Reutilización del conocimiento: Esto permite que la gente reutilice el conocimiento que se ha sintetizado en el software. En vez de empezar siempre desde cero (que es el caso de la industria de software actual) siempre se puede empezar un proyecto desde un fundamento establecido. Esto es equivalente a la manera en la que la ciencia se desarrolla: no se parte de cero, se parte de los descubrimientos previos y se innova sobre el conocimiento que ya se tiene.
- La posibilidad de adaptar el software a sus necesidades.
- Aprender alguna técnica de programación.

Lo mencionado arriba ha dado cabida a que se creen sistemas de cómputo que compiten en casi todos los niveles con los sistemas propietarios, pero no contemplan sistemas de marketing y son tradicionalmente esfuerzos que no son conocidos por el público en general.

2.2 CATEGORÍAS DE SOFTWARE LIBRE

2.2.1 Software de Fuente Abierta "Open Source"

En 1998, cuando una parte de la comunidad decidió dejar de usar el término "software libre" y usar "open source software", con el propósito de evitar la confusión de "free" con "gratis". Otros, sin embargo, apuntaban a apartar el espíritu de principio que ha motivado el movimiento por el software libre y el proyecto GNU, y resultar así atractivos a los ejecutivos y usuarios comerciales. Open source se centra en el potencial de realización de software de alta calidad, pero esquiva las ideas de libertad, comunidad y principio.

El proyecto GNU continúa utilizando el término "free software" [software libre] para expresar la idea de la libertad, donde solamente la tecnología, es lo importante.

2.2.2 Software de Dominio Público

El software de dominio público es software que no está protegido con copyright. Es un caso especial de software libre no protegido con copyleft, que significa que algunas copias o versiones modificadas no pueden ser libres completamente.

"Dominio público" es un término legal y significa de manera precisa "sin copyright" .

2.2.3 Software protegido con copyleft

El software protegido con copyleft es software libre cuyos términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando éstos redistribuyen o modifican el software. Esto significa que cada copia del software, aun si ha sido modificado, debe ser software libre.

El Proyecto GNU, protege mediante copyleft casi todo el software que producen, con el propósito de dar a cada usuario las libertades que el término "software libre" implica.

Copyleft es un concepto general; para proteger actualmente un programa con copyleft, necesita usar un conjunto específico de términos de distribución. Hay muchas maneras posibles de escribir términos copyleft de distribución.

2.2.3.1 Copyleft y la GNU GPL

El copyleft usa la ley de copyright, pero le da vuelta para servir a lo opuesto de su propósito usual; en lugar de ser un medio de privatizar el software, se transforma en un medio de mantener libre al software. La idea central del copyleft es dar a cualquiera el permiso para correr el programa, copiar el programa, modificar el programa y redistribuir versiones modificadas, pero no se da permiso para agregar restricciones propias. De esta manera, las libertades cruciales que definen al "software libre" quedan garantizadas para cualquiera que tenga una copia; se transforman en derechos inalienables.

Para que el copyleft sea efectivo, las versiones modificadas deben ser también libres. Esto asegura que todo trabajo basado en el GNU quedará disponible para la comunidad si se publica.

Cualquier cosa agregada o combinada con un programa bajo copyleft debe ser tal que la versión combinada total sea también libre y bajo copyleft.

La implementación específica de copyleft para la mayoría del software GNU es la Licencia Pública General de GNU (GNU General Public License) o GPL GNU para abreviar. Los manuales GNU también están bajo copyleft, pero se utiliza un copyleft mucho más simple, porque no es necesaria la complejidad de la LPG GNU para los manuales.

2.2.3.2 Software libre no protegido con copyleft

El software libre no protegido con copyleft viene desde el autor con autorización para redistribuir y modificar así como para añadirle restricciones adicionales.

Si un programa es libre pero no protegido con copyleft, entonces algunas copias o versiones modificadas pueden no ser libres completamente. Una compañía de software puede compilar el programa, con o sin modificaciones, y distribuir el archivo ejecutable como un producto propietario de software. Sin embargo, hay versiones no libres también, y hay estaciones de trabajo populares y tarjetas gráficas para PC para las cuales versiones no libres son las únicas que funcionan.

2.2.4 Software abarcado por GPL

La GPL (General Public License / Licencia Pública General) de GNU es un conjunto específico de términos de distribución para proteger con copyleft a un programa. El Proyecto GNU la utiliza como los términos de distribución para la mayoría del software GNU.

2.2.5 Software GNU

Software GNU es software que es liberado bajo el auspicio del Proyecto GNU. La mayoría del software GNU está protegido con copyleft, pero no todos; sin embargo, todo el software GNU debe ser software libre.

Algo de software GNU es escrito por el personal de la Fundación para el Software Libre (Free Software Foundation), pero la mayoría del software GNU es aportada por voluntarios. Parte del software aportado está protegido con copyright por la Fundación para el Software Libre; otra parte está protegido con copyright por los aportadores que los escribieron.

2.2.6 Software Semilibre

El software semilibre es software que no es libre, pero viene con autorización para usar, copiar, distribuir y modificar (incluyendo la distribución de versiones modificadas) sin fines de lucro.

El software semilibre es mucho mejor que el software propietario, pero aún plantea

problemas y no podemos usarlo en un sistema operativo libre.

2.2.7 Software Propietario

El software propietario es software que no es libre ni semilibre. Su uso, redistribución o modificación está prohibida, o requiere que usted solicite autorización o está tan restringida que no pueda hacerla libre de un modo efectivo.

2.2.8 Freeware

El término "freeware" no tiene una definición clara aceptada, pero es usada comúnmente para paquetes que permiten la redistribución pero no la modificación (y su código fuente no está disponible). Estos paquetes no son software libre.

2.2.9 Shareware

Programa que se puede descargar libremente por Internet y utilizarse gratuitamente durante un periodo de prueba. Después del periodo de prueba, sí se desea seguir usando se deberá pagar a su autor unos derechos.

El término significa literalmente programa compartido e indica que cualquiera puede descargar el programa y empezar a emplearlo sin desembolso previo. Pero ello no significa que sean de libre uso o de empleo gratuito. La licencia de uso indica con claridad en cada caso los términos de empleo, así como la cantidad que debe ser abonada en caso de encontrar de utilidad el programa.

El sistema shareware se utiliza a menudo como medio para distribuir versiones de prueba con un coste mínimo.

Las versiones de prueba, en general tienen algún tipo de limitación. En algunos casos, algunas funciones no están disponibles; en otros, el programa sólo admite una cierta cantidad, reducida, de datos. En su versión más popular, el programa tiene toda su funcionalidad, pero sólo es operativo durante 30 días tras su instalación. Al cabo de

éstos, unos programas dejan de funcionar y recuerdan que deben ser desinstalados del ordenador o pagados. Algunos programas simplemente recuerdan cada vez que se ejecutan que el periodo de prueba ha terminado, pero siguen operativos.

El shareware no es software libre, ni siquiera semilibre. Existen dos razones por las que no lo es:

- Para la mayoría del shareware, el código fuente no está disponible; de esta manera, usted no puede modificar el programa en absoluto.
- El shareware no viene con autorización para hacer una copia e instalarlo sin pagar una cantidad por licencia, ni aún para particulares involucrados en actividades sin fines de lucro. (En la práctica, la gente a menudo hace caso omiso a los términos de distribución y lo hace de todas formas, pero los términos no lo permiten).

2.2.10 Software Comercial

El software comercial es software que está siendo desarrollado por una entidad que tiene la intención de generar ganancias económicas mediante el uso del software. Comercial y propietario no son equivalentes. La mayoría del software comercial es propietario, pero hay software libre comercial y hay software no libre no comercial.

Por ejemplo, Ada de GNU siempre es distribuida bajo los términos de la GPL de GNU y cada copia es software libre; pero los desarrolladores venden contratos de soporte. Cuando sus vendedores les hablan a los clientes potenciales, algunas veces el cliente dice "Nos sentiríamos más seguros con un compilador comercial". Los vendedores responden, "Ada de GNU es un compilador comercial; sólo que es software libre"

Para el proyecto GNU, el énfasis está en otro orden: lo importante es que Ada de GNU es software libre; si es comercial no es una pregunta importante. Sin embargo, el desarrollo adicional de Ada de GNU que resulta del negocio que soporta es definitivamente beneficioso.

2.3 SOFTWARE LIBRE FAMOSO.

Las aplicaciones más famosas producidas por los equipos de Software Libre son: el sistema operativo Linux; el servidor Web Apache; la base de datos Postgres; el navegador Mozilla; la suite de aplicaciones de productividad personales de GNOME; la suite de compiladores GCC.

APLICACIONES Y HERRAMIENTAS	APLICACIONES Y HERRAMIENTAS EQUIVALENTES EN SOFTWARE LIBRE
Sistemas Operativos	Sistema Operativo Linux
Servidores de Comunicaciones	
Web **Correo** **DNS** **FTP** **Proxy** **Firewall** **Red**	Apache Sendmail DNS(BIND) FTP Proxy Squid Firewall Linux NFS
Servidores de base de datos	MySQL, PostgreSQL
Aplicaciones y herramientas de desarrollo, administración, mantenimiento y seguridad	Lenguajes para desarrollo de aplicaciones: C++, Java, Perl, Php, Gcc
Desarrollo Web	PHP, Perl, Java, XML, HTML
Herramientas gráficas multimedia, Internet y monitoreo	XML, GIF, LinuxCad, Xfig, NMAP
Ofimática	StarOffice, Koffice
Browser: Navegadores Internet	Netscape, Mozilla, Opera,
Entornos Gráficos: Administradores de ventana	KDE, GNPME
Robótica	LINUXBOT

Tabla 2.1 Software Libre

2.4 VENTAJAS EN TÉRMINOS DE COSTOS.

El pago de una licencia de un sistema Windows hoy en día oscila entre los doscientos dólares hasta los trescientos dólares dependiendo de la versión. En el caso de servidores estamos hablando de unos cuatro mil dólares solamente en licencias por cada computadora que se desee poner en línea.

Si a estos sistemas se les va a incluir un sistema de productividad del tipo Office, los costos de las estaciones de trabajo saltan al rango 700-900 dólares por estación.

Este costo no es significativo si estamos hablando de un par de estaciones de trabajo, pero cuando multiplicamos el costo por estación de trabajo a un par de millones, estamos hablando de unas cantidades de dinero muy grandes.

Un ejemplo es tan solo el proyecto Red Escolar en que México tiene planeado instalar un millón de computadoras en 5 años. Es decir, que en estos sistemas se va a pagar entre 200 y 700 millones de dólares exclusivamente en licencias.

Una licencia es el "derecho" de utilizar el software. Físicamente, la compañía que manufactura el software solamente tiene que hacer una copia del compacto en la que se distribuye el software y vender las copias.

Cada licencia que se paga es equivalente aproximadamente al costo de una computadora nueva.

Es decir que el costo en términos de equipo de cómputo y de software puede reducirse a la mitad o se puede instalar el doble de equipo por el mismo precio.

2.5 ¿QUÉ ES OCTAVE?

Octave es un software que nos permite programar y utilizar una serie de funciones principalmente numéricas. Octave ofrece una interfaz de usuario interactiva, orientada a línea de comandos, pero también puede ser utilizado en modo no interactivo, leyendo sus órdenes de fichero.

Octave es un software de distribución gratuita que se apega a la filosofía GNU, esto es poder tener acceso al programa y al código fuente del programa para modificarlo (si nos interesa), sin tener que pagar ni por su uso, ni por su obtención, además de poder hacer cuantas copias se quieran e instalaciones en diferentes máquinas, también, sin tener que pagar.

2.6 DETALLES TÉCNICOS DE OCTAVE

- Octave está escrito en C++ usando la librería STL.
- Tiene un intérprete que interpreta su propio lenguaje (de sintaxis similar a Matlab), y permite una ejecución interactiva.
- Puede extenderse el lenguaje con funciones y procedimientos.
- Utiliza otros programas GNU para ofrecer al usuario la creación de gráficos para luego imprimirlos o guardarlos (Gnuplot).
- Dentro del lenguaje también se comporta como una consola de comandos (shell). Esto permite listar contenidos de directorios, por ejemplo.
- Además de correr en plataformas Unix también lo hace en Windows.
- Puede cargar archivos con funciones de Matlab de extensión .m.

2.7 CARACTERÍSTICAS DE OCTAVE

- La sintaxis es similar a la utilizada en C.
- Es un lenguaje interpretado.
- Se pueden generar scripts.
- Soporta gran parte de las funciones de la librería estándar de C.
- El lenguaje está pensado para trabajar con matrices y provee mucha funcionalidad para trabajar con éstas.
- No es un lenguaje de programación orientada a objetos. Por lo tanto, no tiene clases ni objetos.
- Soporta estructuras similares a los structs de C.

- Al ser su licencia GNU General Public License, puede ser copiado y utilizado libremente.

2.8 INSTALACIÓN DE OCTAVE

Existen varias versiones de Octave, todas disponibles en forma gratuita en Internet. La página principal de octave es http://www.octave.org . La versión original se utiliza en LINUX (un sistema operativo gratuito), a pesar de ello existen versiones para Windows (el sistema operativo propietario de Microsoft). Una de las páginas de donde se puede obtener la distribución para Windows, es la página de la Universidad de Córdoba que es http://www.efn.uncor.edu. El link que nos permite bajar el instalador de Octave para Windows es http://www.efn.uncor.edu/departamentos/computacion/materias/informatica/software.html

Una vez obtenido el instalador simplemente se debe ejecutar el mismo y se iniciará el proceso de instalación de OCTAVE.

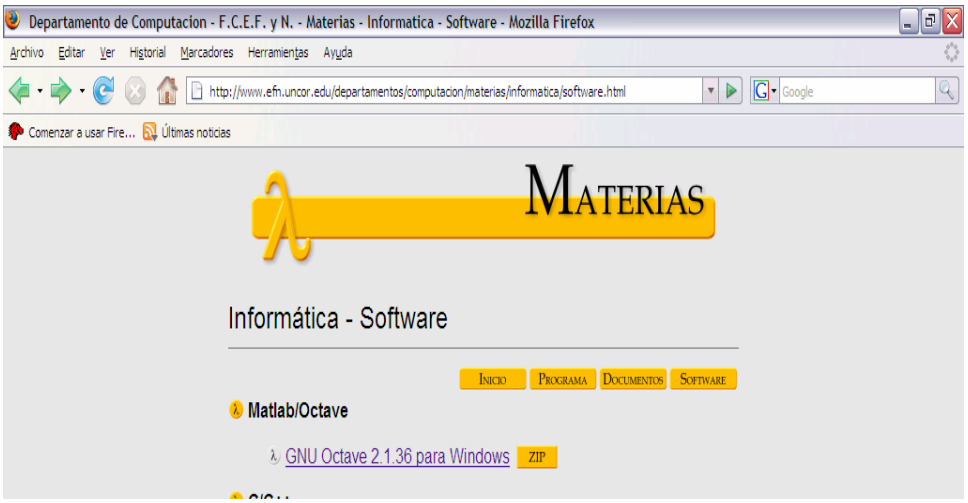


Figura 2.1 Sitio Web nos permite bajar el instalador de Octave para Windows.

Capítulo III
Plan de Migración.

Los lineamientos de la metodología busca que las instituciones apliquen un método de trabajo general e integral para migrar sus sistemas de información; servidores de comunicaciones, servidores de base de datos, aplicaciones, herramientas de desarrollo, gestión y administración a la plataforma de software libre.

3.1 Objetivo General.

La metodología para la Implementación del software libre en los sistemas de información de las instituciones tiene como objetivos fundamentales:

- Una guía para migrar los sistemas de información a software libre.
- Propiciar un orden legal de la tenencia del software.

3.2 Objetivos Específicos.

- Recomendar una alternativa de software mediante la implementación de software libre GPL.
- Las alternativas de migración para los diferentes sistemas de información con los que cuentan las instituciones.
- Pautas de instalación y operación del software requeridos por las instituciones.

3.3 Plan de Migración.

La metodología empleada describe en detalle las fases a seguir en el proceso de migración a software libre.

FASE 1.	**Sensibilización a nivel institucional respecto a la migración a software libre.**
FASE 2.	**Alternativas de Migración.** Dentro de esta etapa, se realizará el análisis y estudio de las alternativas de migración a software libre. Se evaluarán las herramientas y aplicaciones equivalentes en software libre, a ser implementadas.
FASE 3.	**Instalación y Operación del software libre a implementarse en la Institución.**
FASE 4.	**Capacitación al personal sobre migración a software libre.**

Tabla 3.1 Fases del Plan de Migración

Las instituciones pueden efectuar reajustes o realizar procesos con mayor nivel de detalle dependiendo de la complejidad de su plataforma tecnológica y de los recursos que se asignen.

3.3.1 Fase 1: Sensibilización institucional respecto a la Migración a Software Libre.

Para poder efectuar la migración de los sistemas de información a software libre es importante definir acciones de sensibilización. Que la Alta dirección, el área informática, y los usuarios finales; tomen conciencia de *¿Porqué una migración? Y las ventajas de la migración.*

3.3.1.1 Ventajas de la migración a software libre.

Las ventajas en el uso de software libre, más importantes son:

- **Confiabilidad y estabilidad**.- El software libre, al ser público, está sometido a la inspección de una multitud de investigadores, que dan soluciones y comparten la

solución con los demás.

- **Permanente desarrollo**.- El software libre en estos últimos años, viene desarrollándose en relación directa al desarrollo de las Tecnologías de información. Este auge permanente del software libre esta obligando a muchas empresas desarrolladoras de software y fabricantes de hardware a incorporar GNU/Linux en sus productos. Como los desarrolladores de SQl, ORACLE, INFORMIX, AUTOCAD, COREL, entre otros, fabricantes de hardware como IBM, COMPAQ, SUN, ACER, etc.
- **Factibilidad de migración de los sistemas existentes a GNU**. Los sistemas de información mas usados por las instituciones, tienen en software libre, una herramienta o aplicación equivalente en GNU.

La migración de los sistemas de información a software libre requerirá de la participación de todos los niveles de la institución.

El medio para lograr esta sensibilización será a través de:

- Exposiciones sobre el plan de migración dentro de la institución.
- Diseño, elaboración y distribución de folletos explicativos del plan de migración, sus ventajas y beneficio a la institución.
- Lista de los sistemas de información a ser migradas indicando su equivalente en GNU.

La migración a software libre implica un proceso integral donde tienen que participar todas las áreas relacionadas para garantizar la continuidad de los procesos de la institución.

3.3.2 Fase 2: Alternativas de migración.

En la determinación de las alternativas de migración a software libre , será necesario analizar ciertos elementos, entre los que se encuentran:

- La factibilidad total o parcial de migración a software libre en la institución.

- El hardware que se posee en la institución.
- Disponibilidad del recurso humano.
- Ventajas costo-beneficio para la ejecución del proyecto de migración.

3.3.3 Fase 3: Instalación y operación del software a migrar.

La fase de implementación del software libre y las pruebas tendrán como objetivo la realización de la migración.

3.3.3.1 Pruebas de operatividad del software migrado.

La calidad en el Software puede ser medida por características externas, tales como que sea fácil de usar, fácil de implementar o de rápida ejecución; o por características internas, como diseño modular.

Sin embargo, son las características internas las que generan las externas. Cuando se habla de un software de calidad, ciertos factores deben ser tomados en cuenta al efectuar las pruebas:

- **Robusto:** que pueda manejar circunstancias fuera de su diseño.
- **Flexible:** que se adapte fácilmente ante cambios de requerimientos.
- **Reusable:** que pueda utilizarse en otros sistemas, además de aquel para el cual fue creado.
- **Compatible:** que se entienda con otros programas.
- **Eficiente:** en tiempo, memoria, en almacenamiento, etc.
- **Portable:** que pueda transferirse fácilmente a otros ambientes de software y hardware.
- **Verificable:** que permita detectar cuándo y dónde falla.
- **Fácil de usar:** para los usuarios y futuros programadores.

3.3.4 Fase 4: Capacitación a los Usuarios.

Para garantizar la implementación del software libre en la institución se requiere de una capacitación a los recursos que participan en el proceso de migración.

El propósito principal es centrar el conocimiento del plan de migración a software libre, para lo cual la capitación debe centrarse en :

- Difusión del impacto de software libre en las instituciones.
- Difusión de las características, bondades y beneficios del software libre para las instituciones.
- Prestar todo el apoyo y cooperación en la implementación.
- Operar el software libre.

Capítulo IV
Fase 1: Sensibilización Respecto a la Migración a Software Libre.

4.1 INTRODUCCIÓN

Si bien la mayoría de las aplicaciones del software se han concentrado en ambientes empresariales, la popularización del uso de Internet ha obligado a los Países a aplicar estas tecnologías en ambientes de tipo social como la Educación, la sistematización de las Empresas del gobierno, los manejos de comunidades remotas, el comercio, etc Todo esto tendrá un impacto enorme en la sociedad debido a que la masificación es de grandes proporciones.

Razones que han generado el impacto de GNU:

- Nos permite con libertad **Ejecutar** el programa, con cualquier propósito. **Modificar** el programa para adaptarlo a sus necesidades. **Redistribuir** copias. **Distribuir** versiones modificadas del programa, de tal manera que la comunidad pueda beneficiarse con sus mejoras.

- Como solución económica, rentable, poderosa, estable, robusta y segura. Permite aplicar estas tecnologías en ambientes de tipo social como la educación, la sistematización de las entidades del estado, los manejos de comunidades remotas, el comercio, etc.

- Grandes empresas fabricantes de hardware y software vienen adaptando a software libre sus productos, garantizando a nivel mundial su uso. Tal es el caso de IBM, Compaq, Acer, Intel, Motorola, Sun, Oracle, Corel, SQL, MAC, entre otras.

- Desde el punto de vista educativo, el desarrollo de sistemas de información utilizando software libre permitirá crear una base de desarrolladores de aplicaciones y herramientas GNU, y con el apoyo de las universidades se crearía una sinergia muy importante.

4.2 COSTO - BENEFICIO

El software, como mercadería, por lo general no está a la venta. Lo que el usuario adquiere, es una licencia respecto de los usos que puede dar a los programas en cuestión. El software no sólo cuesta un precio de adquisición de licencia. También cuesta mantenerlo, operarlo, ajustarlo. Es importante para el usuario el poder mantener estos costos bajo control, pues de lo contrario puede llegar a verse impedido de llevar a cabo sus metas. El usuario que adquiere software libre lo hace sin ninguna inversión monetaria o a muy bajo costo y ofrece un conjunto de recursos muy amplios. Cualquier persona con una computadora y una conexión a Internet puede utilizar un software libre. Para la mayoría de usuarios individuales el software libre es una opción atractiva por las libertades que garantiza sin necesidad de verse agobiados por el precio. Sin embargo, en el caso de empresas y la Administración Pública, el costo del software es un factor importante y a veces determinante en la elección de nuevos sistemas informáticos. Cuando se analiza el precio de una solución tecnológica se suele hablar del TCO (Total Cost of Ownership), es decir, del coste total de la propiedad que tiene una determinada solución de software. Este concepto fue inventado por el Gartner Group en 1987 como herramienta de análisis exhaustiva de los costos de una solución de mercado y se convirtió en un estándar. En análisis refleja el costo del programa, la ayuda, y el mantenimiento tecnológico de la solución. Si partimos de la base que el software libre prácticamente carece de costo de licencia y por lo tanto, esta parte del presupuesto se puede invertir para mejores fines.

Al revisar los costos de infraestructura tecnológica utilizando un modelo de Costo Total de Propiedad TCO, se reconoce que los costos del hardware y software están conectados a otros costos tales como soporte, entrenamiento y tiempo fuera de operación. Una organización promedio tiene más del 50 por ciento de sus costos en informática ubicados en sus costos indirectos no presupuestados.

La inversión en infraestructura tecnológica, como cualquier decisión de negocios, está basada en el valor económico. A nivel de costo-beneficio la migración a software libre

conviene a las entidades por su tendencia a bajar el costo total de propiedad TCO.

4.3 DESARROLLO HISTÓRICO DE LAS ASIGNATURAS DE INGENIERÍA.

Para darle un sentido histórico a los particulares avances pedagógicos en los países europeos, se debe comenzar por la primera asignatura que comenzó a dictarse a partir de 1976 bajo la denominación de Computación y Cálculo Numérico, con un Programa basado en los conceptos de los Métodos Numéricos y la implementación de los correspondientes algoritmos en el lenguaje de programación científica denominado FORTRAN IV. Este programa se mantuvo, salvo por la forma de dictado que desde 1981 se dividió en teóricos de Cálculo Numérico y prácticas de Laboratorio. Computación ubicada en el primer año se ocupa de los conceptos básicos de la Informática y en particular de Programación de Computadoras y Métodos Numéricos en tercer año de los respectivos temas de aplicación en las diferentes ingenierías. Este cambio además estuvo acompañado de un cambio en el lenguaje de programación ya que se adoptó el lenguaje PASCAL y tanto la formulación de programas así como su diseño se vieron influenciadas por las ideas de la Programación Estructurada y los Algoritmos y Estructuras de Datos.

Simultáneamente a los cambios originados por las corrientes europeas, se produce en Estados Unidos una difusión masiva del Sistema Operativo Unix en los ámbitos universitarios y del lenguaje con el que se desarrollo que es el lenguaje C y sus antecesores que también sirve al desarrollo de los Sistemas Operativos DOS y Windows de Microsoft. Este lenguaje fue incorporado al dictado durante 1991, que posteriormente fue adecuada durante 1994 a las ideas de Programación Orientada a Objetos en lenguaje C++. La tendencia a la orientación a objetos se ha profundizado con el lenguaje JAVA, que posee una base sintáctica en los lenguajes del tipo C.

Durante la década de los 90 se mantuvo el equilibrio entre los conceptos muy básicos sobre Algoritmos y un lenguaje de programación claramente Imperativo, pero nunca pudo implementarse la asignatura basada en el paradigma de objetos en forma completa por falta de tiempo.

Durante éste mismo período la asignatura de Métodos Numéricos acompañó los cambios en los lenguajes, incluyendo en parte la enseñanza de una Planilla de Cálculo como modo de presentar una herramienta orientada a determinados problemas de ingeniería. Se redactó un manual sobre soluciones a los métodos numéricos con Excel, pero como en el caso de la orientación a objetos tampoco se dispuso del tiempo suficiente para su dictado completo., no obstante marcó la necesidad de contar con herramientas de software interactivas y de uso inmediato con capacidades de graficación.

Desde 2001 se produce una nueva bifurcación con dos orientaciones muy pronunciadas, por un lado la profundización de los conceptos de la Ciencia de la Computación en términos de Algoritmos y Estructuras de Datos ya que permitió darle a la asignatura una formalización, similar a las de otras asignaturas de la matemática, en lo que hace al Diseño de Algoritmos de los que carecía, además ya es un hecho indiscutible que los conocimientos sobre informática se han hecho imprescindibles como herramienta de Simulación de Sistemas en todas las ciencias y las ingenierías, en particular en las de Electrónica y Computación donde ya no se conciben como temas o disciplinas separadas.

La otra dirección importante es la elección, no ya de un lenguaje de programación sino de una herramienta de gran difusión en el desarrollo de aplicaciones interactivas de métodos numéricos y de graficación en ingeniería, denominado MATLAB. Éste intérprete fue desarrollado en el lenguaje C y es en su funcionalidad de programación, enteramente similar en su sintaxis. Ésta elección se ha basado también en la dificultad de la enseñanza, a estudiantes de primer año, de herramientas informáticas y lenguajes de programación para aplicarlas a Problemas de Ingeniería que desconocen casi completamente, inclusive carecen de conceptos de física básica, lo que impide el uso de material didáctico orientado a problemas. Se hizo necesario adaptar los textos de MATLAB a la enseñanza de la programación para lo cual se desarrolló un Apunte Operacional, lo que permite unir en un mismo texto las explicaciones teóricas significativas con el código correspondiente. Éste material se desarrolló en 2000 con la

finalidad de ser utilizado como transparencias de ejemplos de problemas y ejemplos resueltos de cierta longitud y complejidad, lo que no puede lograrse en un tiempo razonable, con uso exclusivo del pizarrón.
Durante 2001, 2002 y 2003 se ha incrementado el uso del Software Libre en ingeniería, en particular la existencia del clon de MATLAB denominado GNU-OCTAVE lo que permitió utilizar dicho software sin restricciones de licencias de ningún tipo.

Esta interpretación de la experimentación es el que debe animar al estudiante ante los problemas de la ingeniería ya que las herramientas informáticas adecuadas le permitirán poner a prueba su propia comprensión de las teorías, mediante la observación de los resultados controlados por la lógica de la especificación de programas de simulación y su puesta en funcionamiento mediante su implementación computacional.

La ubicación de la asignatura de matemáticas en el primer año de estudio de las ingenierías, pone a prueba a los estudiantes de nuevo ingreso al sistema universitario, frente a un sistema de aprendizaje basado en el autocontrol del propio conocimiento, lo que requiere ante todo dejar de creer en los textos sin un pensamiento crítico propio y por ende le exige poner a prueba y experimentar con la realidad acotada o simulación que es posible construir mediante la matemática, la física y la informática.

En síntesis, para entender y por ende aprender no basta con aceptar como buenas las relaciones lógicas, por correctas que éstas sean, que momentáneamente parecerán evidentes y casi de sentido común cuando son enunciadas por el profesor en el dictado de una clase o en una lectura circunstancial, es imprescindible dudar, poner a prueba, experimentar, para lo cual nada más gratificante y creativo que construir una solución a un problema y comprobar que funciona.

Capítulo V
Fase 2: Alternativas de Migración.

5.1 Requisitos para las herramientas matemáticas.

Se detallan a continuación los requisitos que debería proporcionar un programa para poder ser usado en los primeros niveles educativos de una titulación universitaria que cuente con ciertos requerimientos de cálculo científico.

Nivel 0. Requisitos mínimos:

• Calculadora científica. El requerimiento más básico: se trata de poder realizar todo tipo de operaciones numéricas, almacenar valores en variables, utilizar funciones exponenciales, logarítmicas, trigonométricas, tablas de valores leídos de un fichero.
• Resolución numérica de ecuaciones y sistemas de ecuaciones, calculando (probablemente de forma aproximada) valores numéricos que solucionan las ecuaciones.
• Gráficas de funciones 2D y 3D.

Nivel 1. En función de las necesidades de la asignatura, puede ser conveniente el utilizar programas que se centren en el cálculo con expresiones numéricas de tipo matricial. Aunque, con frecuencia, herramientas originalmente diseñadas para el cálculo simbólico son aptas para ser usadas para cálculo numérico y matricial (y viceversa), su diseño y su filosofía justifica que las herramientas matemáticas de cálculo se puedan agrupar en dos grupos diferentes:

• Tipo A. Programas de cálculo simbólico, los ejemplos más conocidos en el mundo del software privativo: Mathematica , Maple , etc.

• Tipo B. Programas de cálculo numérico y matricial, de los cuales el más conocido (con licencia privativa) es, sin duda, Matlab . Son programas

centrados en optimizar y hacer fáciles a la vez las operaciones con expresiones numéricas.

- Cálculo matricial: Operaciones numéricas con vectores, matrices, determinantes, transposición, inversión, cálculo de autovalores,...
- Manipulación de rangos de valores numéricos, extracción de rangos de valores de un vector y de submatrices.

5.2 CENSO Y ANÁLISIS PREVIO DE LAS HERRAMIENTAS DISPONIBLES.

A continuación se analizan algunas de las opciones más utilizadas en el mundo del software matemático con licencia libre, desde el punto de vista de su uso en la universidad.

Por supuesto, no están todas las herramientas relacionadas con las matemáticas disponibles con licencia libre (sería prácticamente imposible, dado su cantidad) y existen numerosas omisiones algunas de ellas intencionadas (excelentes programas relacionados con la geometría, juegos, etc.), pues no eran requeridas por las circunstancias que motivaron el presente estudio. Por otra parte, el análisis de los programas censados no ha sido realizado de forma rigurosa y lleva, por tanto, una cierta dosis de subjetividad, aunque se haya tratado de minimizarla.

5.2.1 PROGRAMAS DE CÁLCULO SIMBÓLICO, CÁLCULO NUMÉRICO Y MATRICIAL

5.2.1.1 Axiom

Axiom es un potente sistema de computación científica, creado en 1971 por IBM y comercializado con licencia privativa. No fue hasta septiembre de 2002 cuando Axiom fue liberado por el NAG (Numerical Algorithms Group), que lo había comprado a IBM en años 1990. A partir de su liberación, su comunidad de desarrolladores se está centrando en su extensión, incluyendo aspectos como su interfaz de usuario o soporte de gráficos.

Según el Axiom Book (un libro de más de mil páginas, disponible en formato pdf a través de internet), la potencia de Axiom se basa en sus excelentes características

estructurales y a su expansibilidad sin límites: es un sistema abierto, modular y diseñado para soportar un gran número de nuevas características con un mínimo incremento en su complejidad estructural".

Características:

- Existe una excelente documentación.
- Es un programa sólido, potente y expansible, con una amplia comunidad que lo soporta.
- Soporte de datos estructurados, moderno lenguaje de programación que impulsa la programación estructurada y el chequeo fuerte de tipos, como en Pascal o en Ada.
- Su interfaz de usuario, por defecto, es en modo consola de texto, aunque admite otras posibilidades, desde su ejecución en un buffer de emacs hasta el uso de TEXmacs, un editor de textos basado en LATEX que permite empotrar sesiones interactivas con otros programas, entre ellos Axiom.
- Disponible para distintos sistemas operativos: GNU/Linux, Mac, MS Windows.

Desventaja:

- No soporta Gráficos.

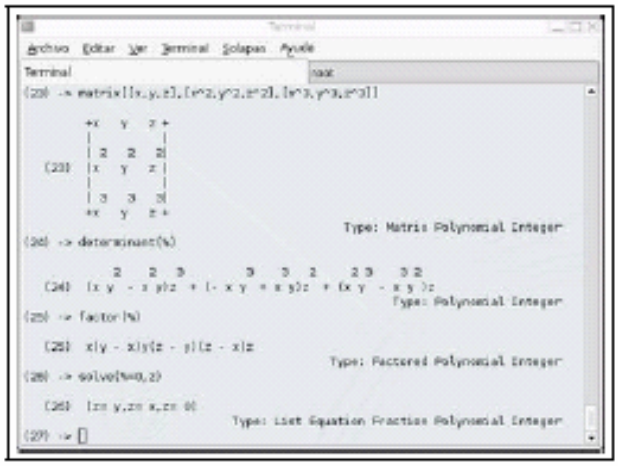


Figura 5.1 Axiom, captura de pantalla

5.2.1.2 Yacas

YACAS (Yet Another Computer Algebra System) es un entorno de cálculo simbólico de propósito general. Incluye todas las capacidades básicas que se suponen en este tipo de programas: cálculo en precisión arbitraria, aritmética racional, números complejos, cálculo de derivadas, resolución de ecuaciones (simbólica y numéricamente), etc.

Características:

- Es un programa moderno, escrito en C++, que está siendo desarrollado por una comunidad muy dinámica.
- Incluye una amplia documentación.
- Interfaz con usuario: Se utiliza la clásica consola de texto para introducir los comandos de YACAS, aunque existe una interfaz de usuario propia, todavía experimental, llamada proteus y basada en la líbrería gráfica fltk. Además, existe la posibilidad de utilizar como interfaz el programa TEXmacs.
- Gráficos
    1. YACAS incluye soporte de gráficos 2D a través de gnuplot, conocida herramienta de gráficos 2D y 3D interactiva y orientada a línea de comandos.

    2. Además, existe soporte de gráficos 3D mejorado siempre que se cuente con el programa Superficie, que proporciona gráficos de gran calidad mediante OpenGL. Se trata de un programa en desarrollo que ya ofrece interesantes perspectivas.

Desventaja:

- Existen versiones para GNU/Linux pero no para MS Windows.

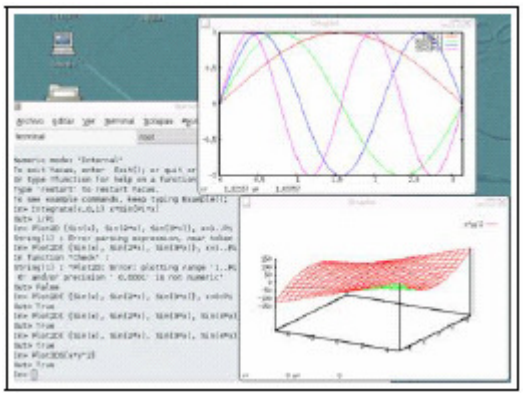


Figura 5.2 YACAS, captura de pantalla

5.2.1.3 Euler

EULER es un programa más de cálculo numérico matricial del estilo de Matlab y Octave (aunque no intenta ser compatible con ellos). Es un programa sencillo y flexible, que contiene, tanto en sus versiones GNU/Linux como MS Windows una interfaz de usuario propia bastante cómoda (utilizando las librerías GTK en GNU/Linux) y buenos gráficos 2D y 3D.

Características:

- Moderno lenguaje de programación.
- Incluye las funciones básicas requeridas para este tipo de programas.

Desventaja:

- No soporta Gráficos en versión MS Windows.

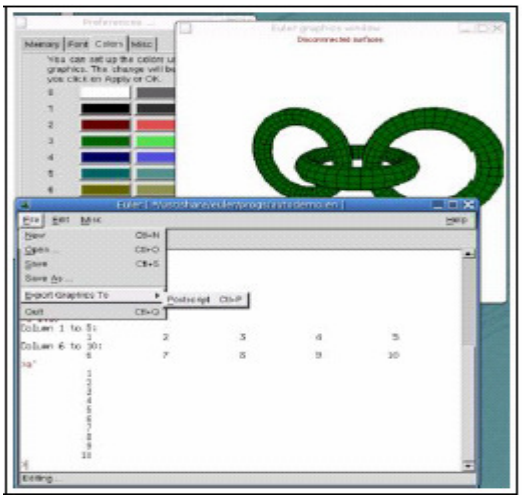


Figura 5.3 EULER, captura de pantalla

5.2.1.4 Octave

Octave fue creado en 1988 por John W. Eaton como soporte a un libro de texto sobre ingeniería química en la Universidad de Texas con el objetivo de crear un lenguaje de cálculo numérico fácil de usar, interactivo y con una amplia comunidad de usuarios. En 1994 con la versión 1.0, obtuvo su madurez para la enseñanza, la investigación y la empresa.

Octave cuenta con gráficos 2D y 3D a través de funciones que son compatibles con Matlab y que utilizan gnuplot como motor gráfico, aunque también existe un soporte alternativo de gráficos 2D y 3D utilizando la librería plplot.

Características:

- La sintaxis es similar a la utilizada en C.
- Es un lenguaje interpretado.
- Se pueden generar scripts.
- Trabaja con matrices y provee mucha funcionalidad para trabajar con éstas.

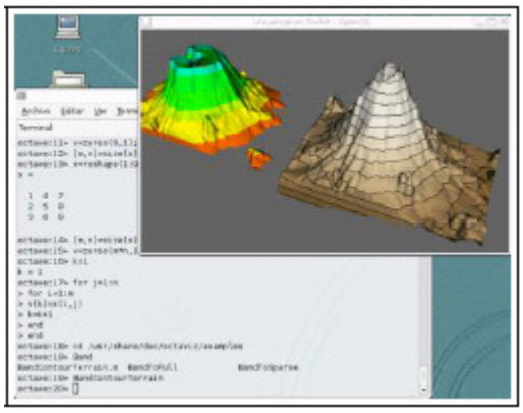


Figura 5.4 Octave, captura de pantalla.

5.2.1.5 Scilab

Scilab es un programa para la computación numérica y científica desarrollado a partir de 1990 por investigadores las instituciones francesas INRIA y ENPC.

Por lo demás, Scilab es un excelente programa, con cierto grado de compatibilidad con Matlab y con una interfaz de usuario integrada (línea de comandos incluyendo menús, gráficos, ayuda) que puede resultar agradable a los amantes de los entornos de ventanas, aunque la versión GNU/Linux (no así la versión MS Windows) resulta menos vistosa debido a las librerías gráficas utilizadas.

Desafortunadamente, aun siendo gratuito y estado disponible su código para su estudio y su uso, no puede considerarse 100% software libre (por ejemplo, no cumple las "Debian's Free Software Guidelines" y es considerado "non-free") debido que la siguiente cláusula en su licencia impone limitaciones a su libre distribución: commercial use or circulation of the DERIVED SOFTWARE shall have been previously authorized by INRIA and ENPC.

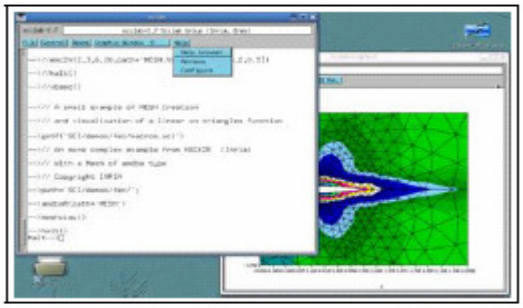


Figura 5.5 Scilab, captura de pantalla

5.3 ELECCIÓN DEL SOFTWARE LIBRE PARA IMPLEMENTARLO EN EL PLAN DE MIGRACIÓN.

La elección de OCTAVE se ha basado en los siguientes criterios:

1. Su similitud sintáctica y funcional con el lenguaje MATLAB. De amplia difusión en aplicaciones para Ciencias e Ingeniería.

2. Su carácter de lenguaje GNU, lo que hace que su uso sea de disposición libre e incluso adaptable.

3. Entre varias de las características que comparte con MATLAB y otros, está la de ser un Intérprete, evitando de ese modo el lento proceso de compilación y facilitando la construcción de prototipos ejecutables.

4. Uno de los argumentos de la difusión de su uso es su orientación a la solución numérica de los problemas del análisis matemático, en particular el uso simbólico de vectores y matrices.

5. Otra característica que se deriva de su condición de Intérprete es su uso como *calculadora* y por ende permite también describir los problemas como composición de aplicación de funciones.

6. El argumento adicional del uso como herramienta de prototipado y de simulación es la capacidad de graficación suministrada por la herramienta de graficación científica GNUPLOT.

Capítulo VI
Fase 3: Instalación y Operación de Octave.

6.1 Instalación de Octave.

6.1.1 Versión para sistemas operativos Microsoft

Para la instalación en Windows, como indica en la página de descarga, existen dos opciones

1. Instalar Octave junto con el sistema Cygnus. Cygnus crea un entorno similar a GNU/Linux sobre sistemas operativos Windows, esto permite la ejecución de numerosas aplicaciones GNU.
2. Hacer una instalación exclusiva de Octave. Este es el método preferido ya que es más sencillo y ocupa menos espacio. El paquete autoinstalable se obtiene de la página de from Octave-Forge Files.

6.1.2 Toolbox de Control de Sistemas de Octave (OCST).

La OCST es un conjunto de funciones de octave relacionada con la disciplina de control de sistemas. En la versión GNU Octave 2.1.73 estas funciones están integradas dentro del paquete Octave.
Tras la instalación de Octave, para probar que la OCST funciona correctamente, se puede intentar ejecutar el siguiente comando:

```
octave:1> tf2sys([1],[1 1]);
```

6.1.3 Instalación de Octave bajo Cygwin en Windows.

Cygwin es Linux dentro de Windows. Es un entorno muy completo pero orientado a usuarios experimentados en sistemas UNIX. Se podría decir que sólo debe instalar cygwin quien de verdad sepa qué está haciendo.

Un motivo para instalar cygwin es Octave. Octave no tiene una versión nativa para Windows, el instalador que nos ofrece es antiguo y no incluye las librerías de ampliación.

No sólo estaremos instalando Octave. Cygwin es un entorno UNIX completo que puede crecer hasta convertirse en un sistema operativo adicional dentro de Windows. Nos permite incluso sustituir la ventana gráfica completa para correr aplicaciones Linux ignorando la infraestructura de gráficos de Windows. La potencia es enorme y todo lo que es potente es difícil de manejar. Es sin duda la manera más segura de aprender a utilizar Linux, utilizando nuestro Windows como base.

PRIMER PASO: DESCARGAR CYGWIN.

Encontraremos el instalador de cygwin en la página http://www.cygwin.com. Todo el proceso de instalación es administrado por el programa que descargaremos, llamado setup.exe. Podemos guardarlo donde queramos siempre que recordemos donde lo hemos dejado. Esta aplicación es en realidad un gestor que controla todos los paquetes, no sólo los instalados, también los disponibles en las bases de datos externas a las que se conecta.

El funcionamiento es el mismo que los gestores de instalación de Linux o que el Windows Update. El programa generará un interfaz con los paquetes instalados, sin instalar, actualizables, sin utilizar, etc. Primero se tendrá que configurar la instalación, dicha configuración se utilizará cada vez que pidamos al programa setup.exe que busque paquetes adicionales, actualice los instalados o elimine algunos de ellos.

SEGUNDO PASO: CONFIGURAR LA INSTALACIÓN

La primera ventana que aparecerá con un doble clic en el icono de setup.exe será la siguiente:

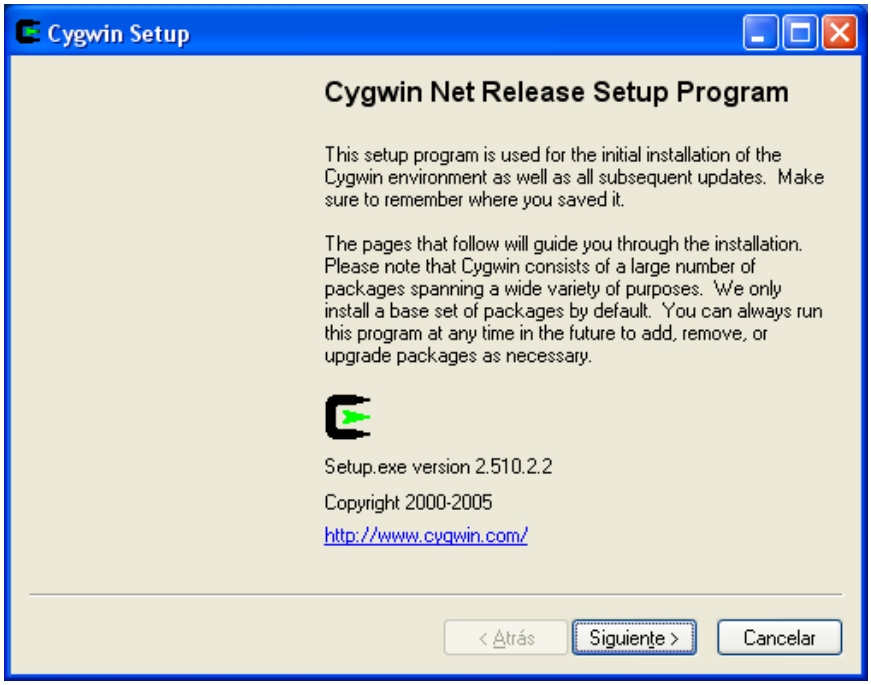


Figura 6.1 Pantalla de configuración de la instalación.

En la Figura 6.1 aparece información muy importante sobre la instalación. El segundo párrafo es clave. La instalación por defecto es la mínima para tener un sistema funcionando; si queremos más tendremos que pedirlo.

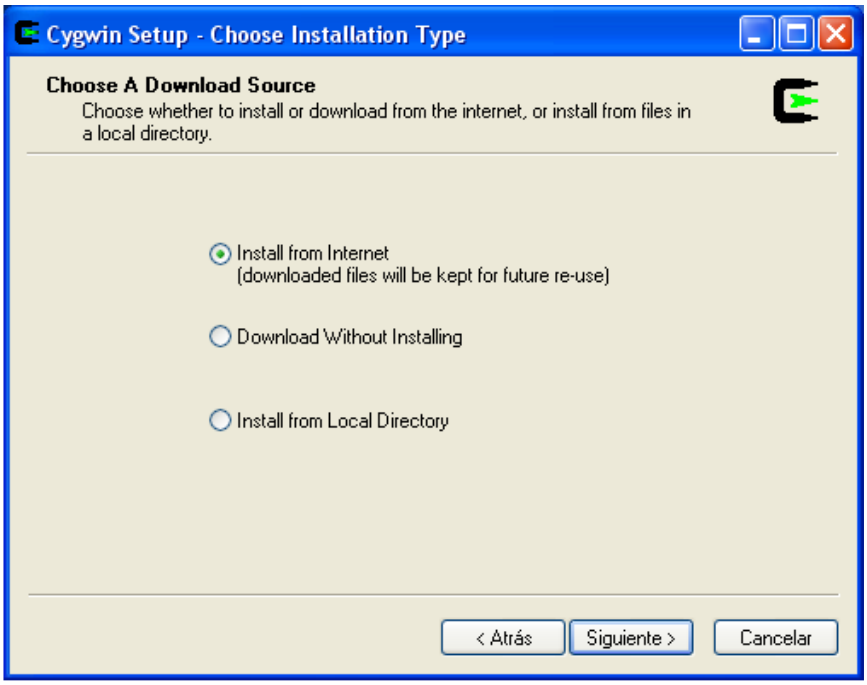


Figura 6.2 Pantalla de selección del tipo de instalación.

Lo primero que nos preguntará será qué tipo de instalación deseamos Figura 6.2. Acto seguido aparecerá el diálogo del tipo de archivo por defecto Figura 6.3.

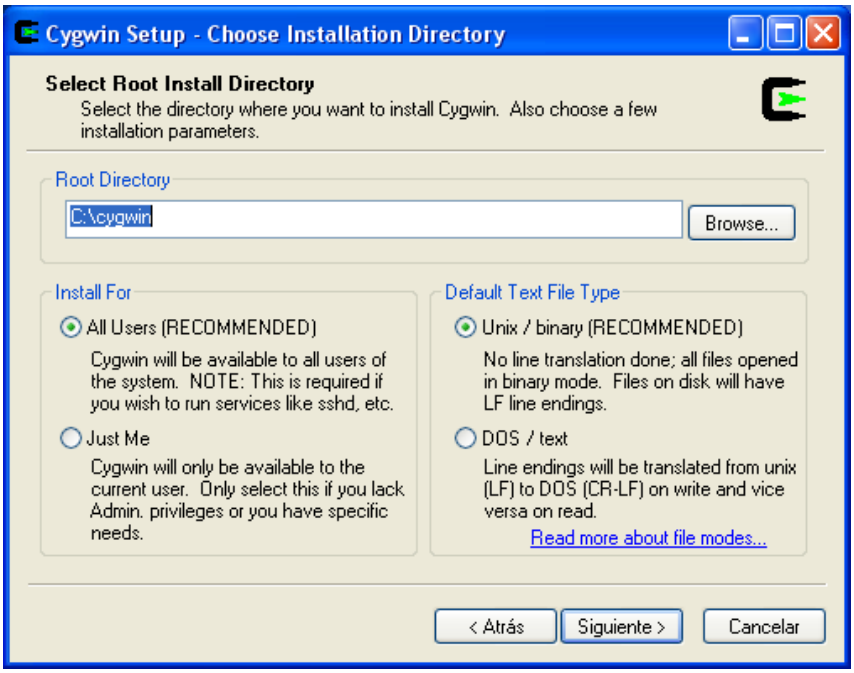


Figura 6.3 Pantalla en la cual se selecciona el directorio en el que se instalará Cygwin

Se instalará todo el entorno en c:\cygwin para que sea más fácil localizar los archivos. Se instalará para todos los usuarios, de otro modo sólo podría utilizar el entorno un usuario con privilegios de administración. Aunque no hay ningún virus que se aproveche de la instalación de Cygwin en un sistema Windows debemos tener en cuenta que utilizar un programa con privilegios de administrador es siempre un problema de seguridad.

Seleccionaremos el tipo de archivo UNIX si instalamos cygwin como un entorno UNIX puro; usaremos el tipo de archivos DOS si queremos comunicar cygwin con Windows muy a menudo.

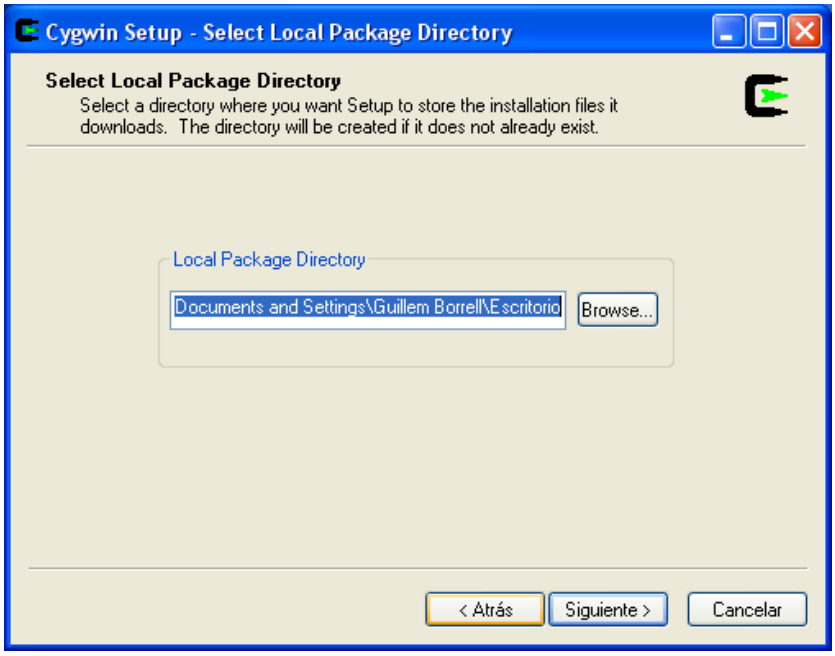


Figura 6.4 Pantalla en la cual se selecciona el directorio en el que se quiere guardar los archivos temporales que vaya descargando.

El paso siguiente es dar el directorio (creado con antelación) en el que queremos que guarde todos los archivos temporales que vaya descargando Figura 6.4.

TERCER PASO: SELECCIONAR LOS PAQUETES NECESARIOS

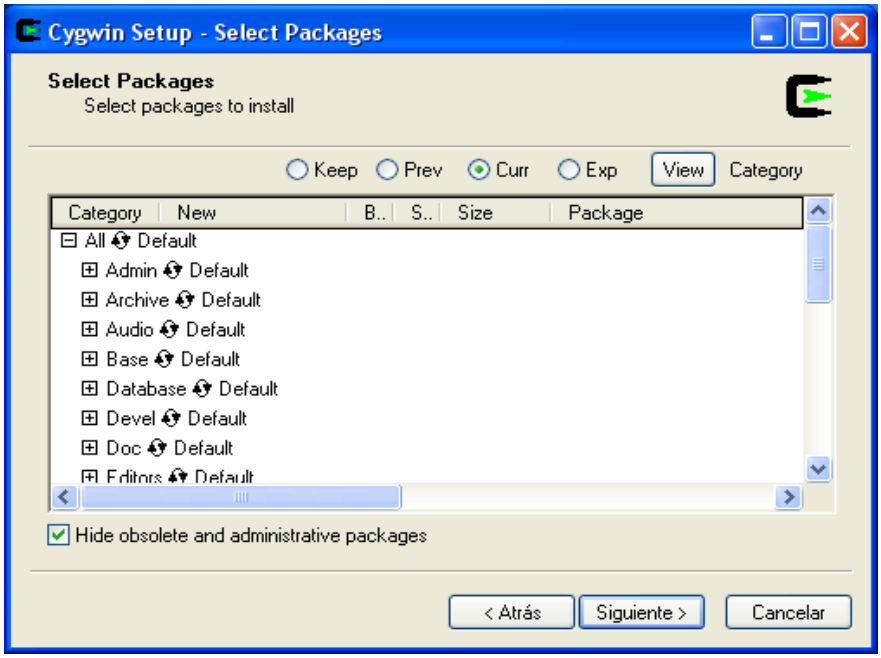


Figura 6.5 Pantalla de selección de componentes a instalar.

En la Figura 6.5 se seleccionan los paquetes que se quieren instalar. Por defecto está seleccionada la opción de ver todos los paquetes disponibles en la base de datos. La organización es en forma de árbol y para seleccionar un programa tenemos que activar el círculo con las dos flechas. Tocar más cosas es un poco peligroso y requiere leer el manual de instalación.

El software a instalar y los programas que necesitamos son los siguientes:

- Editors
  - Emacs-X11
- Math
  - octave
  - octave-doc
  - octave-forge
  - octave-htmldoc
  - octave-otags
  - gnuplot
  - lapack
  - ffwt3
- Shells
  - rxvt

En el caso que también queramos utilizar las librerías de ampliación de Octave en C++:

- Devel
  - gcc-core
  - gcc-g++
  - gcc-g77
- Math
  - octave-headers

El instalador calculará las dependencias automáticamente para que no necesitemos instalar nada más manualmente.

CUARTO PASO: POST INSTALACIÓN.

Una vez tengamos los iconos pertinentes en el escritorio y el menú tendremos que hacer un par de cosas antes de poder utilizar Octave.

Si arrancamos cygwin aparecerá una consola UNIX. Es muy probable que no tengamos ni idea de cómo utilizarla pero como sólo la queremos para arrancar Octave no tenemos que saber mucho. Sólo tenemos que saber que cygwin no activa los gráficos por defecto, tenemos que hacerlo manualmente.

Para pasar al modo gráfico se hará lo siguiente (cada vez que arranquemos Octave):

```
$> startx
```

Se abre una nueva ventana, ahora con fondo blanco. Es exactamente la misma consola de antes pero con soporte para gráficos. Ahora se configura el sistema para utilizar Octave en la nueva consola.

Paso primero

Crear el directorio de trabajo de Octave con el siguiente comando:

```
$> mkdir octave
```

Paso segundo

Activar el soporte para Octave en el editor con:

```
$> emacs .emacs
```

Ahora estamos editando el archivo de configuración del editor Emacs, el que Octave usa por defecto con el comando edit. Cuando aparezca el editor se abrirá un archivo en blanco (el archivo .emacs) en el que escribiremos:

```
(autoload 'octave-mode "octave-mod" nil t)
 (setq auto-mode-alist
       (cons'("\\.m$" . octave-mode) auto-mode-alist))
 (setq inhibit-startup-message)
 (setq transient-mark-mode 't)
 (setq require-final-newline t)
 (cond (window-system
        (mwheel-install)
        ))
 (setq query-replace-highlight t)
 (setq search-highlight t)
 (tool-bar-mode nil)
 (global-font-lock-mode)
```

Podemos activar más o menos opciones pero creo que estas son las más adecuadas. Guardaremos el archivo con el menú file, y luego save buffer.

Paso tercero

Para comprobar si Octave ha cargado bien el soporte para gráficos entraremos en Octave mediante la consola:

```
$> octave
```

Y ya dentro de octave:

```
octave 1> plot([1,2,3])
```

Si aparece la ventana de gnuplot con una recta significa que ya funcionan los gráficos.

6.1.4 Instalación de octave+forge en Windows (GNU Octave 2.1.73).

Este es el modo básico de octave para Windows con el cual podremos realizar todos los cálculos que necesitemos.

1. Vaya a la página http://sourceforge.net/projects/matlinks
2. Seleccione el octave versión para Windows.
3. Cuando haya terminado la descarga haga doble-click en el archivo ejecutable.

octave-2.1.73-1-inst.exe

Figura 6.6 Archivo ejecutable de GNU Octave 2.1.73

4. Siga las instrucciones que le indica el instalador

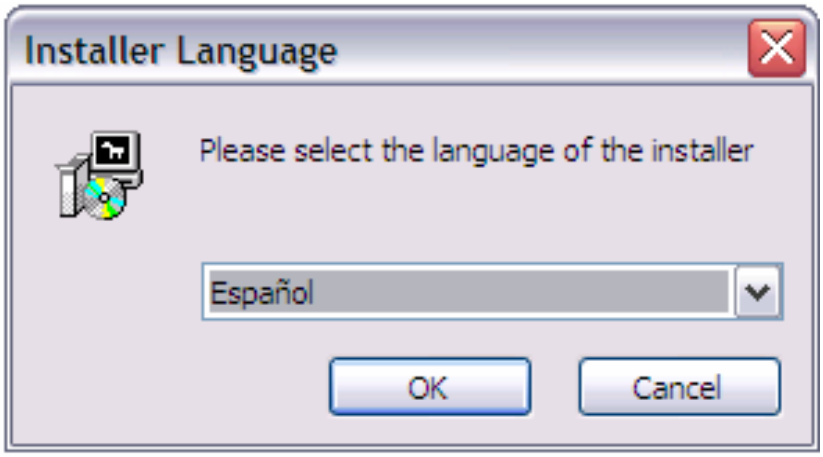


Figura 6.7 Pantalla de selección de idioma para la instalación de GNU Octave 2.1.73

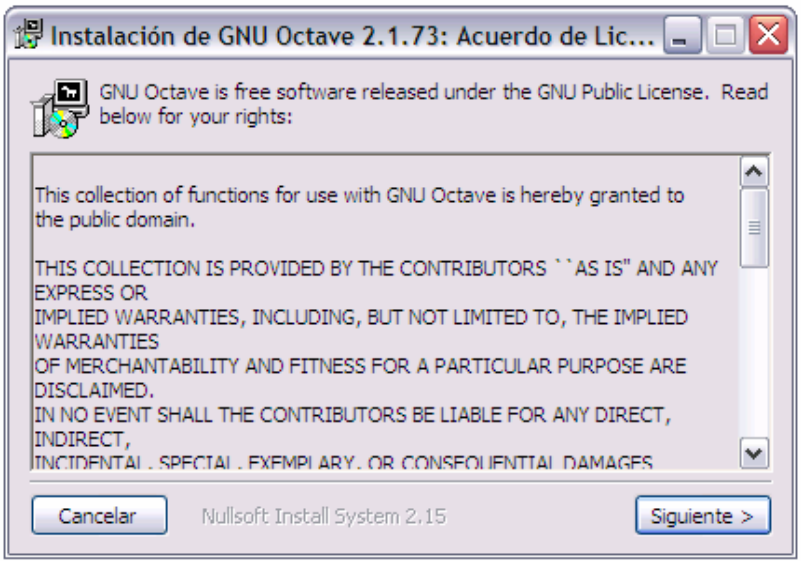


Figura 6.8 Pantalla de licencia de GNU Octave 2.1.73

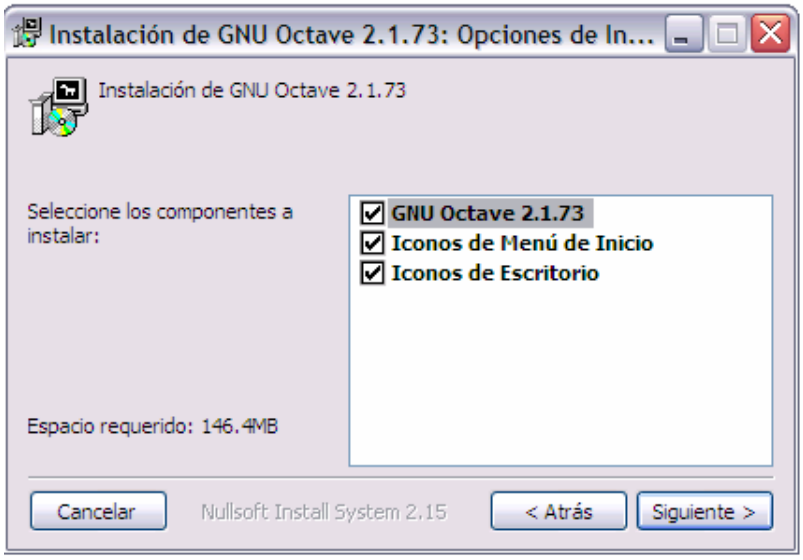


Figura 6.9 Pantalla de selección de componentes a instalar de GNU Octave 2.1.73.

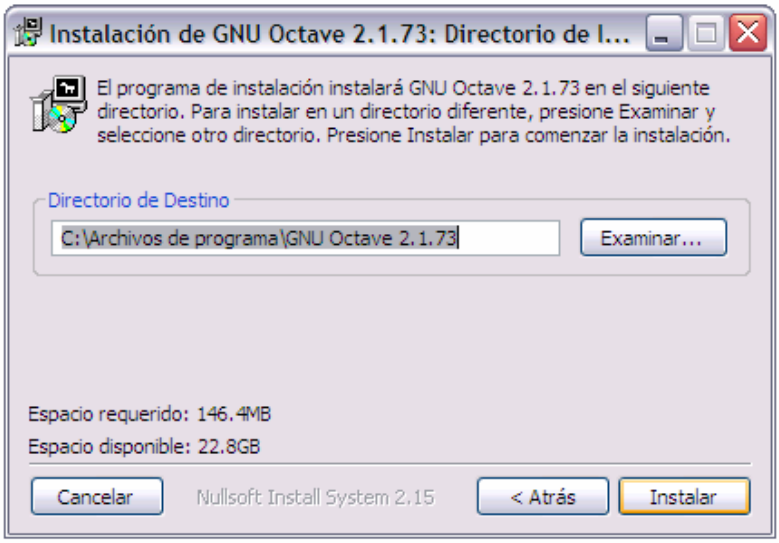


Figura 6.10 Pantalla en la cual se selecciona el directorio en el que se instalará GNU Octave 2.1.73

Figura 6.11 Pantalla de proceso de instalación de componentes de GNU Octave 2.1.73

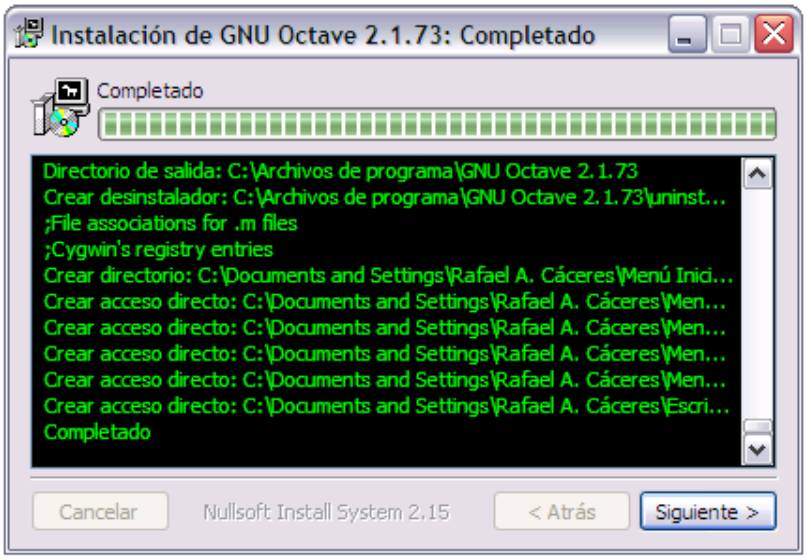


Figura 6.12 Finalización del proceso de instalación de GNU Octave 2.1.73

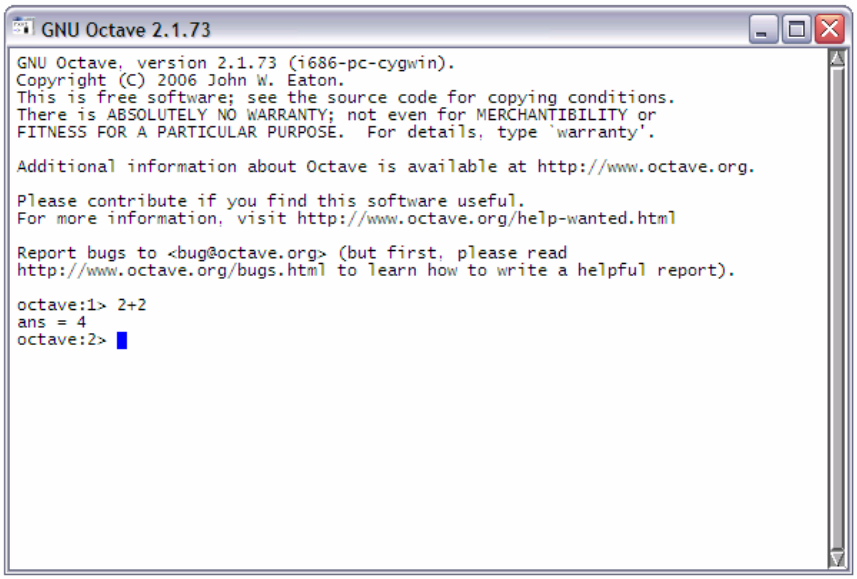


Figura 6.13 Entorno de trabajo de Octave de GNU Octave 2.1.73

6.2 FUNCIONAMIENTO Y OPERACIÓN DE OCTAVE.

6.2.1 Matemática Sencilla.

Octave no toma en cuenta los espacios. El punto y coma al final de la línea le dice a Octave que evalúe la línea, pero que no nos diga la respuesta. Si la sentencia es demasiado larga para que quepa en una línea, una elipsis consistente en tres puntos (...) seguido por Enter indica que la sentencia continúa en la línea siguiente.

Octave ofrece las siguientes operaciones básicas:

OPERACIÓN	SIMBOLO
Suma a+b	+
Resta a-b	-
Multiplicación a*b	*
División a/b	/
Potencia a^b	^

Tabla 6.1. Operaciones básicas en Octave.

6.2.2 Operaciones con Matrices y Vectores

Una de las ventajas de los lenguajes interpretados es su condición de estar ejecutándose en forma permanente, lo que comparten con otros sistemas de

simulación, que en el caso de OCTAVE se puede dar, en éste sentido, una definición como calculadora Matricial. Es decir que a su condición de calculadora de funciones numéricas, que comparte con las calculadoras manuales, se le debe agregar su capacidad de realizar operaciones con matrices y vectores de una manera altamente eficiente, tanto desde su poder de representación y análisis como desde la capacidad de procesamiento.

6.2.2.1 Definición de Matrices

Las matrices y vectores son tipos de datos que se almacenan en la memoria de la computadora bajo el concepto de entorno de trabajo o ambiente de programación (workspace) en forma de variables cuyos identificadores son los nombres, que por convención se adoptan para las matrices las letras mayúsculas y las minúsculas para vectores y escalares.

En OCTAVE el tamaño de una matriz es dinámico y se define por el número de filas y columnas que se le suministran al generarlas mediante la asignación de valores a sus celdas con datos constantes y más tarde se puede cambiar.

Los elementos de una fila se separan por comas (,) o espacios en blanco. Las filas se separan por renglones (enter) o por punto y coma(;)

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

A partir de este momento A (matriz 3x3) queda definida en el espacio de trabajo y se pueden realizar sobre ella todo tipo de operaciones matriciales:
Transpuesta de A:

```
A'
ans =       ans: variable por defecto del último cálculo
            si no se asigna a otra variable nominada
     1      4      7
     2      5      8
     3      6      9

B=A'
B =
     1      4      7
     2      5      8
     3      6      9
```

Operación de producto entre dos matrices previamente definidas y compatibles:

```
A*B
ans =
    14     32     50
    32     77    122
    50    122    194
```

Un vector fila se corresponde con una matriz de una sola fila, y el acceso a sus elementos se realiza mediante un índice de la posición que corresponde solamente a la columna:

```
x = [4 3 2 1]
x =
         4         3         2         1
x(3)
ans = 2
```

Si bien las matrices se pueden *introducir* por filas son *almacenadas* concatenando columnas en un único vector al que se puede acceder con un único índice, pero es más simple hacerlo mediante la notación de dos índices (fila, columna):

```
A(4)
ans =
     2

A(1,2)
ans =
    2
```

La operación de inversión matricial se realiza con una función:
Sea una nueva definición de la matriz A:

```
A=[1 4 -3; 2 1 5; -2 5 3]
A =
     1     4    -3
     2     1     5
    -2     5     3

B=inv(A)
B =
    0.1803    0.2213   -0.1885
    0.1311    0.0246    0.0902
   -0.0984    0.1066    0.0574
```

Si se desea visualizar una mayor precisión usar la función:

```
format  long
B
B =
```

```
   0.180327868852459   0.221311475409836  -0.188524590163934
   0.131147540983607   0.024590163934426   0.090163934426230
  -0.098360655737705   0.106557377049180   0.057377049180328
```

6.2.2.2 Operaciones con Matrices

OCTAVE puede operar con matrices por medio de operadores y por medio de funciones. Se han visto ya los operadores suma (+), producto (*) y transpuesta ('), así como la función invertir inv( ).

Los operadores matriciales de OCTAVE son los siguientes:

- **+** adición o suma
- **-** sustracción o resta
- ***** multiplicación
- **'** transpuesta
- **^** potenciación
- **** división-izquierda
- **/** división-derecha
- **.*** producto elemento a elemento
- **./** y **.** división elemento a elemento
- **.^** elevar a una potencia elemento a elemento

Estos operadores se aplican también a las variables o valores escalares.
Considérese el siguiente ejemplo:

```
A=[1 2; 3 4]
A =
   1 2
   3 4
```

```
A*2
ans =
     2  4
     6  8

A-4
ans =
    -3 -2
    -1  0
```

Si un sistema de ecuaciones algebraicas, con *A* cuadrada e invertible y *x* y *b* vectores columna compatibles con *A* se describe como: $A * x = b$ (1)
Entonces el vector solución *x* se puede hallar en OCTAVE mediante las siguientes operaciones matriciales:

x = inv(A)*b (2a) ó mediante
x = A\b (2b)

Operador de división-izquierda que evita hallar la inversa de A y solo calcula el resultado del sistema de ecuaciones. Si está indeterminado o superdeterminado calcula la Pseudoinversa por mínimos cuadrados Norma sub-1 , que es la solución cuya suma de valores absolutos de componentes (norma sub-1) es mínima. El resultado de OCTAVE es el punto más *cercano* en el sentido de los mínimos cuadrados a las ecuaciones dadas (aunque no cumpla exactamente ninguna de ellas).

Véase el siguiente ejemplo de tres ecuaciones formadas por una recta que no pasa por el origen y los dos ejes de coordenadas (más ecuaciones que incógnitas):

```
A=[1 2; 1 0; 0 1], b=[2 0 0]'
A =   1 2
```

```
      1  0
      0  1
b =   2
      0
      0

x=A\b, resto=A*x-b
x =
    0.3333
    0.6667
resto =
   -0.3333
    0.3333
    0.6667
```

En OCTAVE existe también la posibilidad de aplicar elemento a elemento los operadores matriciales (*****, **^**, **** y **/**).

Para ello basta precederlos por un punto (**.**). Por ejemplo:

```
[1 2 3 4]^2
error: for A^b, A must be square
error: evaluating binary operator `^' near line 10, column 10
[1 2 3 4].^2
 ans = 1 4 9 16

[1 2 3 4]*[1 -1 1 -1]
error: operator *: nonconformant arguments (op1 is 1x4, op2 is 1x4)
error: evaluating binary operator `*' near line 10, column 10

[1 2 3 4].*[1 -1 1 -1]
 ans = 1 -2 3 -4
```

6.2.3 Gráficos Científicos.

Los gráficos 2-D de OCTAVE están fundamentalmente orientados a la representación gráfica de vectores (y matrices) como en MATLAB pero están basados en las funciones **gplot** y **gsplot** del sistema gráfico **GNUplot**. En el caso más sencillo los argumentos básicos de la función **plot** van a ser vectores. Cuando una matriz aparezca como argumento, se considerará como un conjunto de vectores columna (en algunos casos también de vectores fila).

OCTAVE utiliza un tipo especial de ventanas para realizar las operaciones gráficas. Ciertos comandos abren una ventana nueva y otros dibujan sobre la ventana activa, bien sustituyendo lo que hubiera en ella, bien añadiendo nuevos elementos gráficos a un dibujo anterior.

6.2.3.1 Gráficas en 2D

OCTAVE dispone de funciones básicas y especializadas para crear gráficos 2-D. Estas funciones se diferencian principalmente por el tipo de escala que utilizan en los ejes de abscisas y de ordenadas y el tipo de coordenadas. Estas funciones son las siguientes:

bar(x,y)	dados dos vectores *x* e *y* produce un gráfico de barras.
contour(z,n,x,y)	produce un gráfico de contornos de la superficie *z* definida en *3-D.*
hist(x,y,norm)	produce un histograma o la salida numérica para bar().
loglog(args)	produce un gráfico con escala logarítmica en ambos ejes.
plot(args)	crea un gráfico a partir de vectores y/o columnas de matrices, con escalas lineales sobre ambos ejes.
replot	permite redibujar los mismos contenidos luego de un cambio e ejes.

polar(theta,rho.fmt) produce un gráfico bidimensional en coordenadas polares.

semilogx(args)	ídem con escala lineal en el eje de ordenadas y logarítmica en el eje de abscisas.
semilogy(args)	ídem con escala lineal en el eje de abscisas y logarítmica en el eje de ordenadas.
stairs(x,y)	dados dos argumentos produce un gráfico de tipo *escalera.*

Existen además otras funciones orientadas a añadir **títulos** al gráfico, a cada uno de los ejes, a dibujar una cuadrícula auxiliar, etc. Estas funciones son las siguientes:

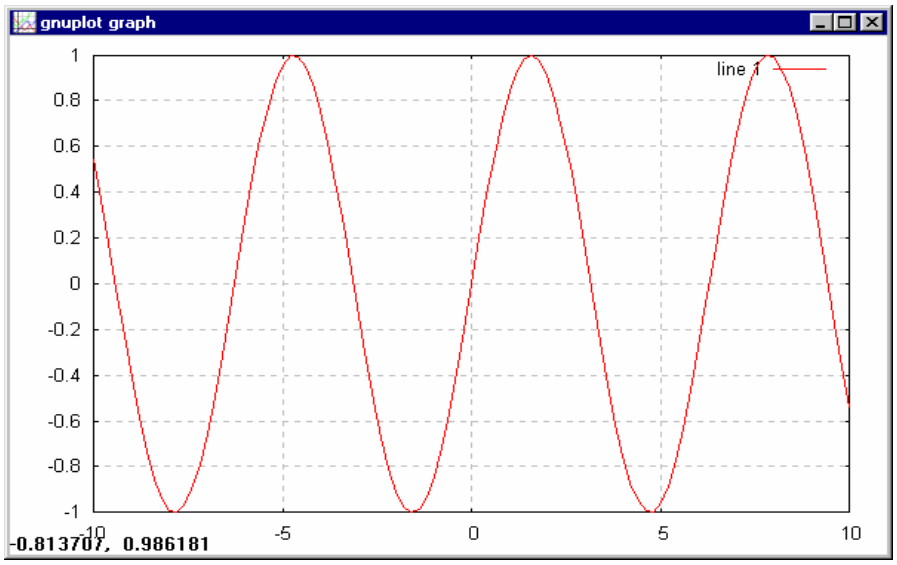
title('título')	añade un título al dibujo (parte superior).
bottom_title('titulo')	añade un título al dibujo (parte inferior).
xlabel('tal')	añade una etiqueta al eje de abscisas. Con xlabel **off** desaparece.
ylabel('cual')	añade una etiqueta al eje de ordenadas. Con ylabel **off** desaparece.
grid (arg)	activa la inclusión de una cuadrícula en el dibujo. Con grid **off** desaparece la cuadrícula.

Los dos grupos de funciones anteriores no actúan de la misma forma. Así, la función **plot** dibuja una nueva figura en la ventana activa (en todo momento OCTAVE tiene una ventana activa de entre todos las ventanas gráficas abiertas), o abre una nueva figura si no hay ninguna abierta, sustituyendo cualquier cosa que hubiera dibujada anteriormente en esa ventana.
Se comenzará creando un par de vectores x e y :

```
x=[-10:0.2:10]; y=sin(x);
```

Se ejecutan los comandos siguientes (se comienza cerrando la ventana activa, para que al crear la nueva ventana aparezca en primer plano):

```
closeplot       se cierra la ventana gráfica activa anterior
grid            se crea una ventana con una cuadrícula
plot(x,y)       se dibuja la función seno conservando la cuadrícula
```

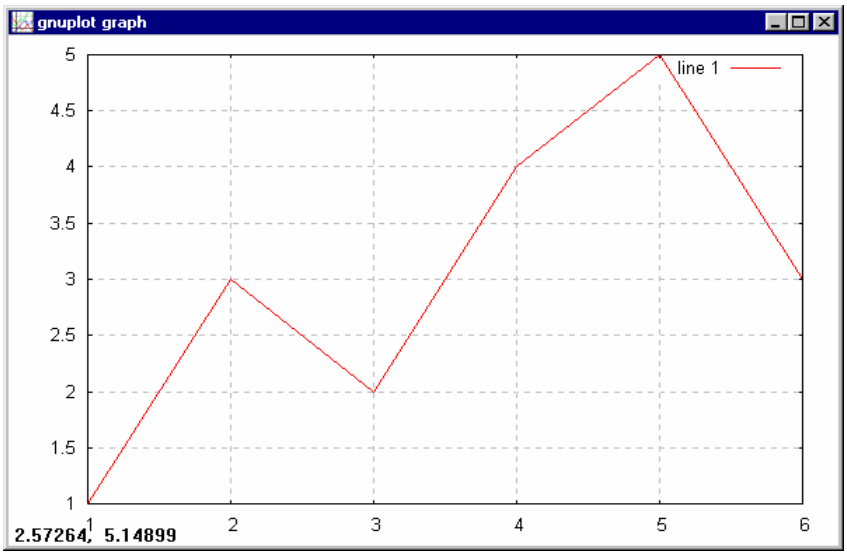


Con la función **hold** pueden añadirse gráficos a una figura ya existente respetando su contenido.

6.2.3.1.1 La función plot.

El elemento básico de los gráficos bidimensionales es el vector. Se utilizan también cadenas de 1, 2 ó 3 caracteres para indicar colores y tipos de línea. La función **plot**(), en sus diversas variantes, no hace otra cosa que dibujar vectores.
Un ejemplo muy sencillo de esta función:

```
x=[1  3  2  4  5  3];
plot(x)
```



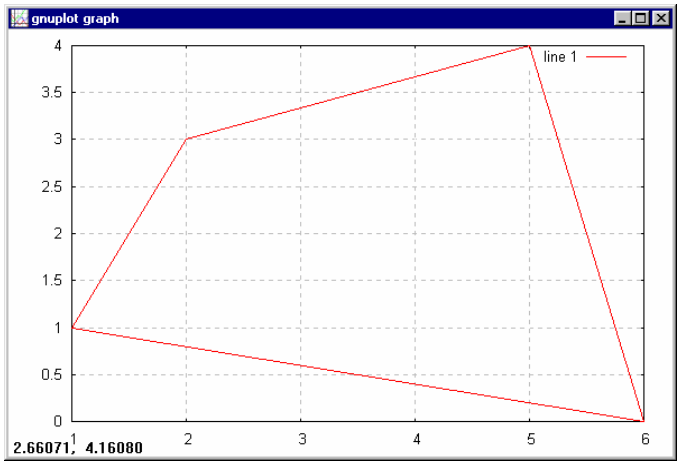
Por defecto, los distintos puntos del gráfico se unen con una línea continua. También por defecto, el color que se utiliza para la primera línea es el rojo. Cuando a la función **plot**() se le pasa un único vector "real" como argumento, dicha función dibuja en ordenadas el valor de los **n** elementos del vector frente a los índices 1, 2, ... n del mismo en abscisas.

Si el vector es complejo, el funcionamiento es bastante diferente.

OCTAVE utiliza por defecto color blanco para el fondo de la pantalla y otros colores más oscuros para los ejes y las gráficas.

Una segunda forma de utilizar la función **plot()** es con dos vectores como argumentos. En este caso los elementos del segundo vector se representan en ordenadas frente a los valores del primero, que se representan en abscisas. Por ejemplo cómo se puede dibujar un cuadrilátero de esta forma (obsérvese que para dibujar un polígono cerrado el último punto debe coincidir con el primero):

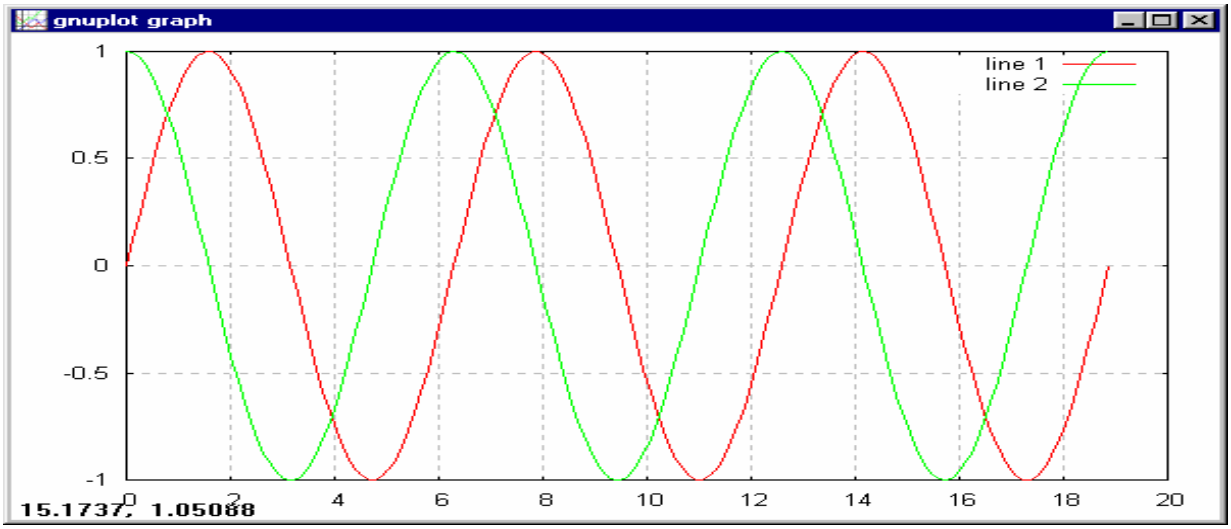
```
x=[1 6 5 2 1]; y=[1 0 4 3 1];
plot(x,y)
```



La función **plot()** permite también dibujar múltiples curvas introduciendo varias parejas de vectores como argumentos. En este caso, cada uno de los segundos vectores se dibujan en ordenadas como función de los valores del primer vector de la pareja, que se representan en abscisas. Si el usuario no decide otra cosa, para las sucesivas líneas se utilizan colores que son permutaciones cíclicas del azul, verde, rojo, cyan, magenta, amarillo y negro.

Obsérvese bien cómo se dibujan el seno y el coseno en el ejemplo:

```
x=0:pi/25:6*pi;
y=sin(x); z=cos(x);
plot(x,y,x,z)
```

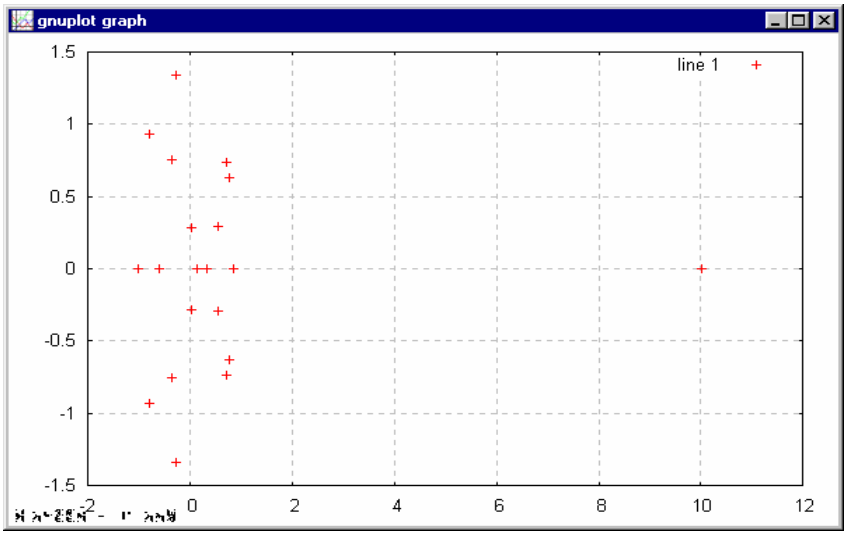


Si se pasa a **plot**() un vector de *números complejos* como argumento, octave simplemente representa la parte real en abscisas, frente a la parte imaginaria en ordenadas.

Para generar un ejemplo de vector complejo se utilizará el resultado del cálculo de valores propios de una matriz formada aleatoriamente:

```
x = eig(rand(20,20))
x =
  1.0e+01  *

   1.00193 + 0.00000i
  -0.02620 + 0.13330i
  -0.02620 - 0.13330i
  -0.08024 + 0.09286i
  -0.08024 - 0.09286i
  -0.10283 + 0.00000i
  -0.03669 + 0.07565i
  -0.03669 - 0.07565i
  -0.06137 + 0.00000i
   0.07121 + 0.07392i
   0.07121 - 0.07392i
   0.07768 + 0.06309i
   0.07768 - 0.06309i
```

```
   0.08461 + 0.00000i
   0.05397 + 0.02944i
   0.05397 - 0.02944i
   0.00163 + 0.02874i
   0.00163 - 0.02874i
   0.03369 + 0.00000i
   0.01439 + 0.00000i
plot(x,'+')
```



Donde se ha hecho uso de elementos, respecto a dibujar con distintos tipos de *markers* (en este caso con signos +), en vez de con línea continua, que es la opción por defecto. En el comando anterior, el segundo argumento es un carácter que indica el tipo de marker elegido.

Si se incluye más de un vector complejo como argumento, se ignoran las partes imaginarias. Si se quiere dibujar varios vectores complejos, hay que separar explícitamente las partes reales e imaginarias de cada vector.

El comando **plot()** puede utilizarse también con matrices como argumentos.

Algunos ejemplos sencillos:

plot(A) dibuja una línea por cada columna de *A* en ordenadas, frente al índice de los elementos en abscisas.

plot(x,A) dibuja las columnas (o filas) de *A* en ordenadas frente al vector *x* en abscisas. Las dimensiones de *A* y *x* deben ser coherentes: si la matriz *A* es cuadrada se dibujan las columnas, pero si no lo es y la dimensión de las filas coincide con la de *x*, se dibujan las filas.

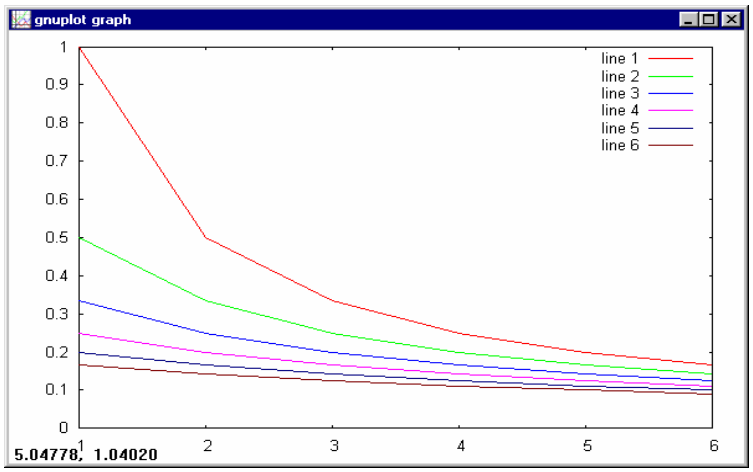
plot(A,x) análogo al anterior, pero dibujando las columnas (o filas) de *A* en abscisas, frente al valor de *x* en ordenadas.

plot(A,B) dibuja las columnas de *B* en ordenadas frente a las columnas de *A* en abscisas, dos a dos. Las dimensiones deben coincidir.

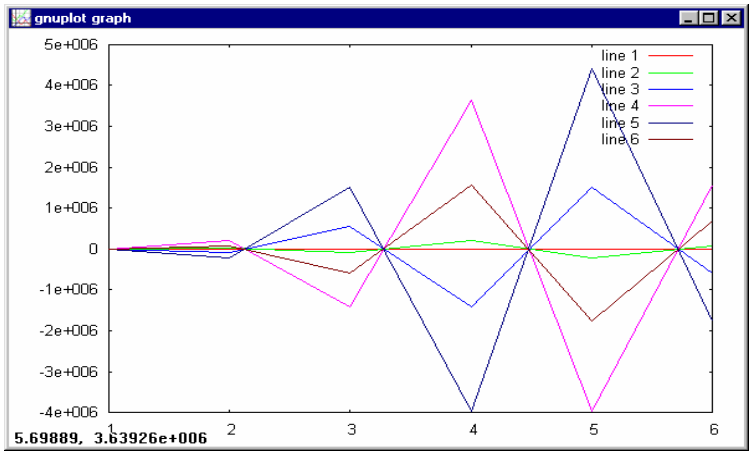
plot(A,B,C,D) análogo al anterior para cada par de matrices. Las dimensiones de cada par de matrices deben coincidir, aunque pueden ser diferentes de las dimensiones de los demás pares.

Ejemplos de uso de las diferentes variantes de la función plot():

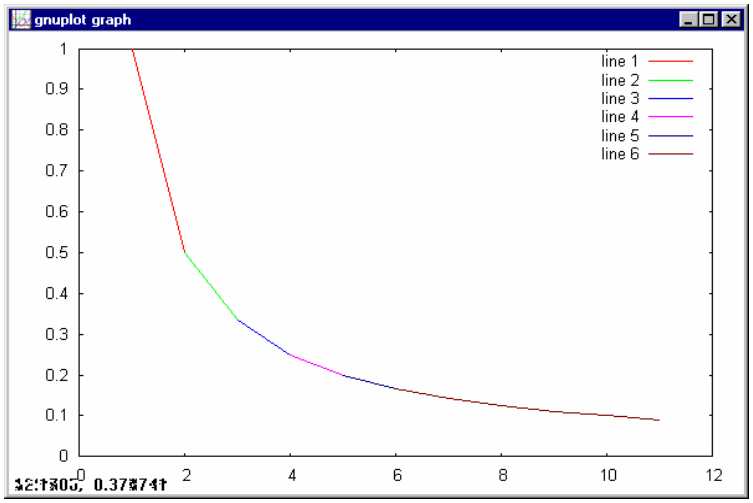
```
A=hilb(6);
plot(A)
```




```
x=[1 2 3 4 5 6];
plot(x, inv(A))
```




```
plot(1./A, A)
```



6.2.3.1.2 Estilos de línea y marcadores en la función plot

La tarea fundamental de la función **plot**() es dibujar los valores de un vector en ordenadas, frente a los valores de otro vector en abscisas. En el caso general esto exige que se pasen como argumentos un par de vectores. En realidad, el conjunto básico de argumentos de esta función es una tripleta formada por dos vectores y una cadena de 1, 2 ó 3 caracteres que indica el color y el tipo de línea o de marker. En la tabla siguiente se pueden observar las distintas posibilidades:

Símbolo	Color	Símbolo	Marcador
y	yellow	**∇** (5)	marcas en triángulos
m	magenta (4)	**o** (3)	marcas en círculos
c	cyan (5)	**x** (4)	marcas en x
r	red (1)	**+** (2)	marcas en + o puntos en □
g	green (2)	* (1)	marcas en *
b	blue (3)	**^**	impulsos o rayado desde eje de abscisas
w	white	**~**	barras de error
k	black	**~#**	cajas de error

Tabla 6.2 Estilos de marcadores en la función plot.

Símbolo	Estilo
-	líneas continuas de rayas
.	líneas continuas de puntos
@nm	líneas de marcas
-@nm	líneas marca-raya
--	líneas a trazos
L	línea en escalera
#	líneas de cajas

n = color , m = tipo de carácter

Tabla 6.3 Estilos de líneas en la función plot.

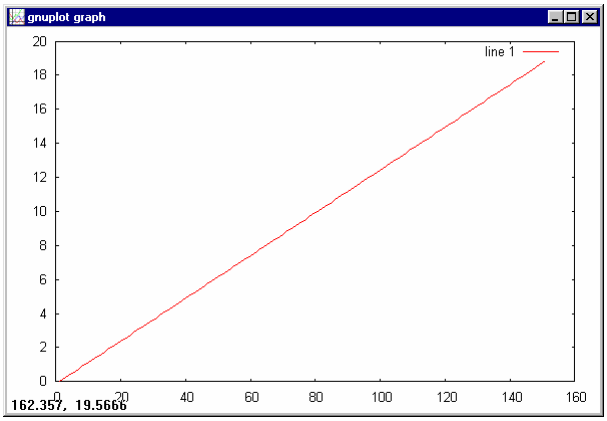
Cuando hay que dibujar varias líneas, por defecto se van tomando sucesivamente los colores de la tabla comenzando por el azul, hacia arriba, y cuando se terminan se vuelve a empezar otra vez por el azul. Si el fondo es blanco, este color no se utiliza para las líneas.

6.2.3.1.3 Añadir líneas a un gráfico ya existente.

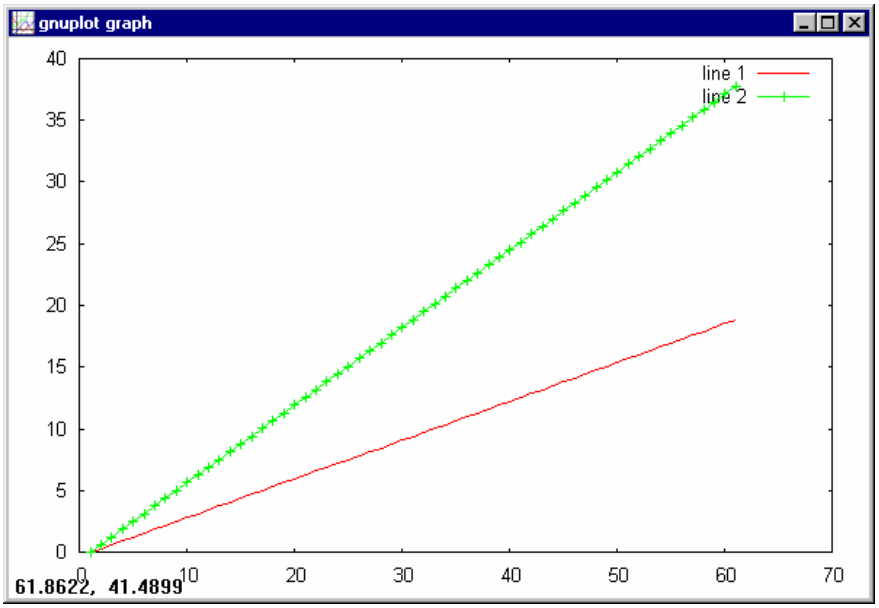
Existe la posibilidad de añadir líneas a un gráfico ya existente, sin destruirlo o sin abrir una nueva ventana. Se utilizan para ello los comandos **hold on** y **hold off**. El primero de ellos hace que los gráficos sucesivos respeten los que ya se han dibujado en la figura (es posible que haya que modificar la escala de los ejes); el comando **hold off** deshace el efecto de **hold on**.

El ejemplo muestra añade las gráficas de x2 y x3 a la gráfica de x previamente creada (cada una con un tipo de línea diferente):

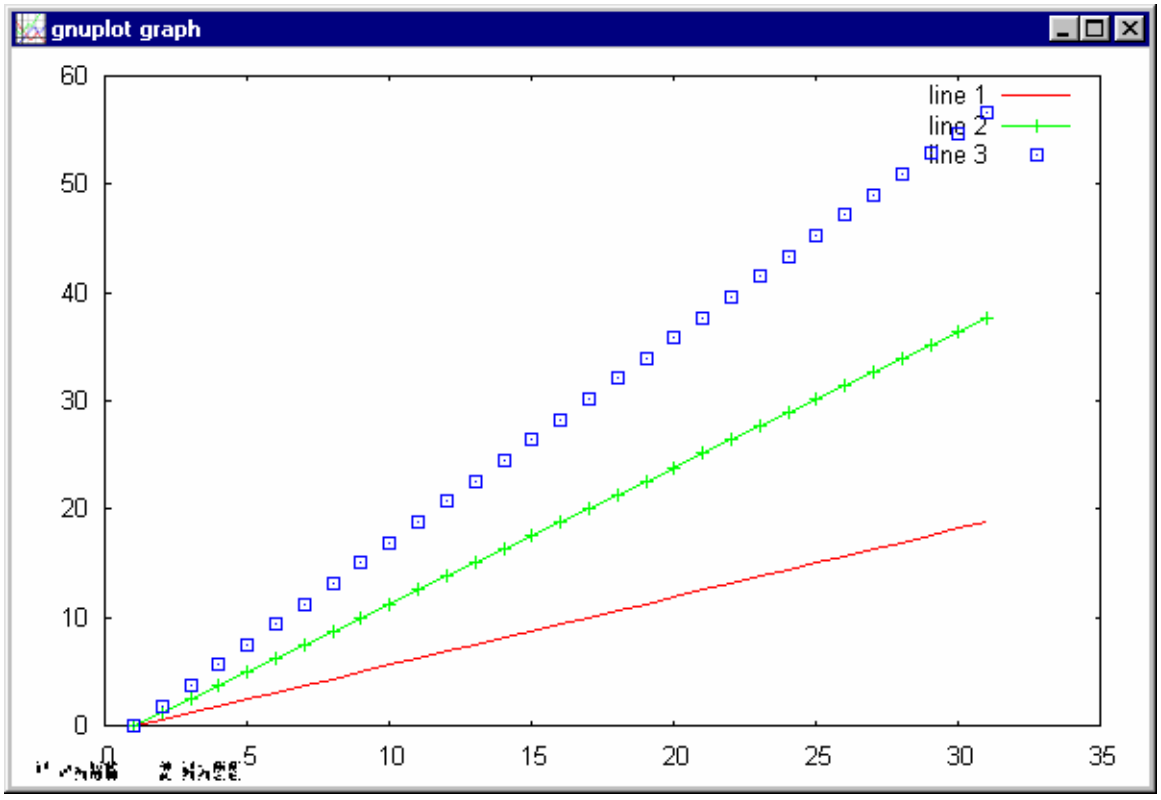
```
clearplot
x=0:pi/5:6*pi;
plot(x)
```




```
hold on

x2=2*x;
plot(x2,'-@2')   puntos "+" y  raya "-" separados   verde
```




```
x3=3*x;
plot(x3,'@32')    puntos separados azul "□"
hold off
```

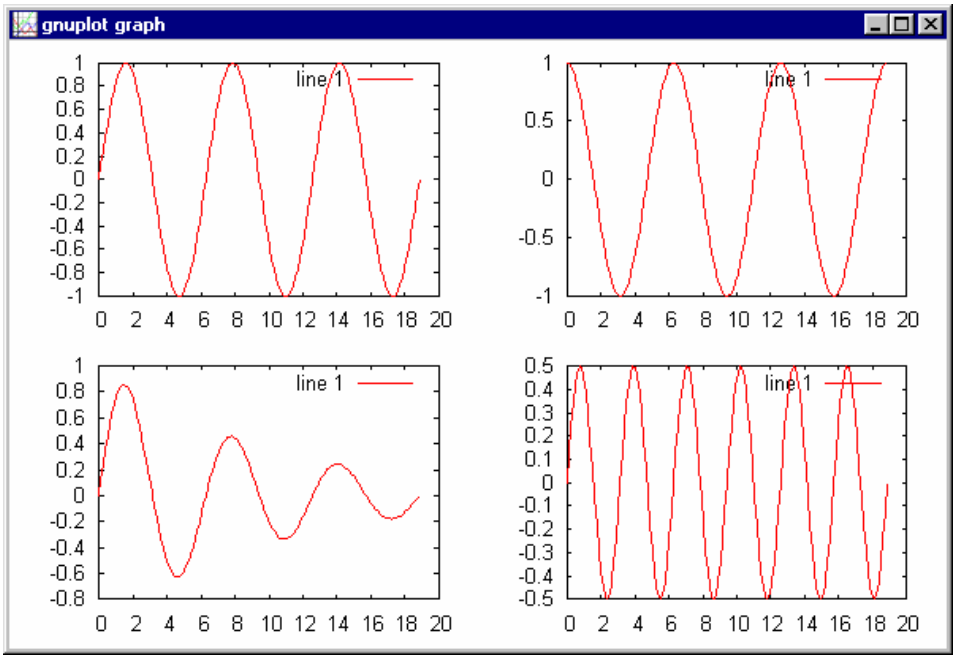


6.2.3.1.3 Comando subplot.

Una ventana gráfica se puede dividir en **m** particiones horizontales y **n** verticales, con objeto de representar múltiples gráficos en ella. Cada una de estas *subventanas* tiene sus propios ejes, aunque otras propiedades son comunes a toda la figura. La forma general de este comando es: **subplot**(m,n,i).

Donde ***m*** y ***n*** son el número de subdivisiones en filas y columnas, e ***i*** es la subdivisión que se convierte en *activa*. Las subdiviones se numeran consecutivamente empezando por las de la primera fila, siguiendo por las de la segunda, etc.
Por ejemplo:

```
clearplot
y=sin(x); z=cos(x); w=exp(-x*.1).*y; v=y.*z;
subplot(2,2,1), plot(x,y)
subplot(2,2,2), plot(x,z)
subplot(2,2,3), plot(x,w)
subplot(2,2,4), plot(x,v)
```



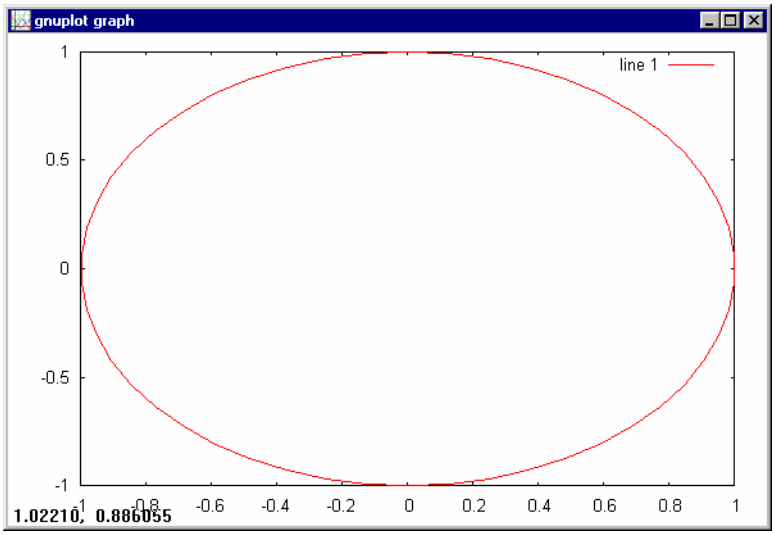
6.2.3.1.4 Control de los ejes.

OCTAVE tiene por defecto, que en algunas ocasiones puede interesar cambiar. El comando básico es **axis**(limites). Por defecto, ajusta la *escala* de cada uno de los ejes de modo que varíe entre el mínimo y el máximo valor de los vectores a representar. Este es el llamado modo *auto*, o modo automático. Para definir de modo explícito los valores *máximo* y *mínimo* según cada eje, se utiliza el comando:

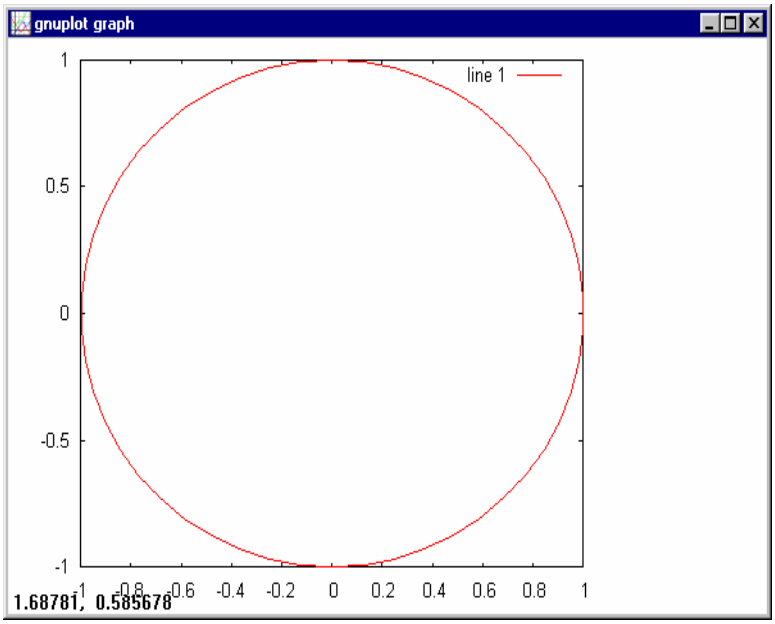
axis([xmin, xmax, ymin, ymax, zmin,zmax]) son opcionales de 2 en 2

axis('auto')	devuelve el escalado de los ejes al valor por defecto o automático.
axis	devuelve un vector v con [xmin, xmax, ymin, ymax, zmin, zmax]
axis(axis)	mantiene los ejes en sus actuales valores, de cara a posibles nuevas gráficas añadidas con hold on.
axis('**ij**')	utiliza ejes de pantalla, con el origen en la esquina superior izda y el eje j en dirección vertical descendente.
axis('**xy**')	utiliza ejes cartesianos normales, con el origen en la esquina inferior izda. y el eje y vertical ascendente.
axis('**equal**')	el escalado es igual en ambos ejes.
axis('**square**')	la ventana será cuadrada.
axis('**image**')	la ventana tendrá las proporciones de la imagen que se desea representar en ella (por ejemplo la de una imagen bitmap que se desee importar) y el escalado de los ejes será coherente con dicha imagen.
axis(**'normal'**)	elimina las restricciones introducidas por 'equal' y 'square'
axis('tic[xyz]')	introduce tics en los ejes indicados. Por Ej. 'ticx'
axis('label[xyz]')	introduce un rótulo en los ejes indicados.
axis(**'off'**)	elimina las etiquetas, los números y los ejes
axis('**on**')	restituye las etiquetas, los números y los ejes

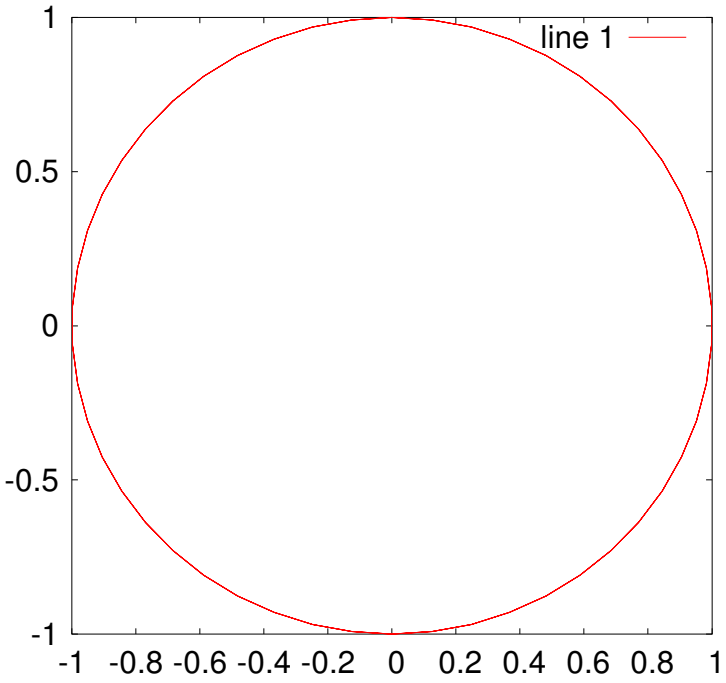
```
x=0:pi/25:6*pi;
plot(y,z)        Ejes en distinta escala
```




```
axis('equal')    Ejes en igual escala
replot
```

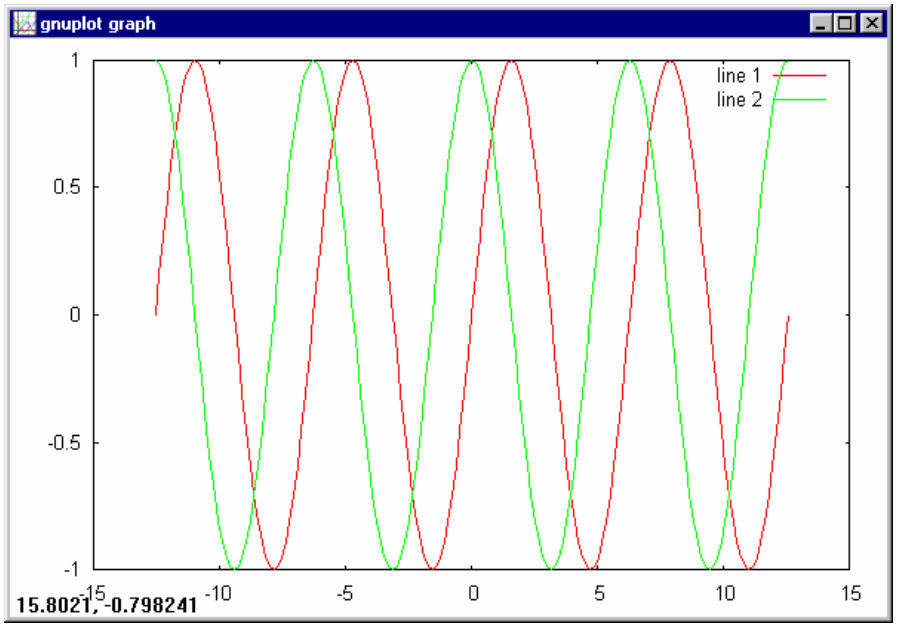



```
axis('square')  Ejes en igual escala y ceñidos a la figura
replot
```



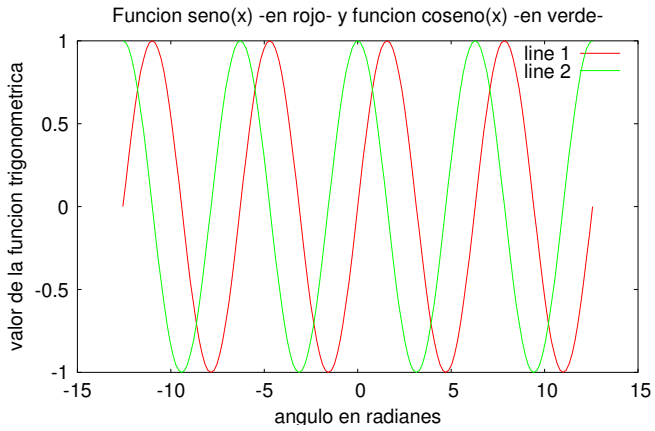
Ejemplos de uso desde la línea de comandos:

```
x=[-4*pi:pi/20:4*pi];
plot(x,sin(x),'r',x,cos(x),'g')
```

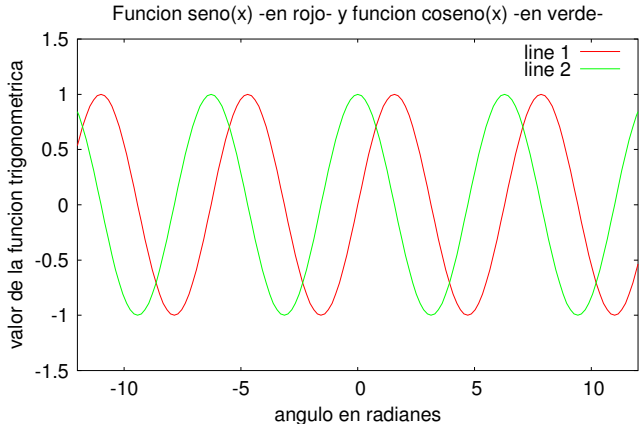



```
title('Función seno(x) -en rojo- y función coseno(x) -en verde-')
replot
```

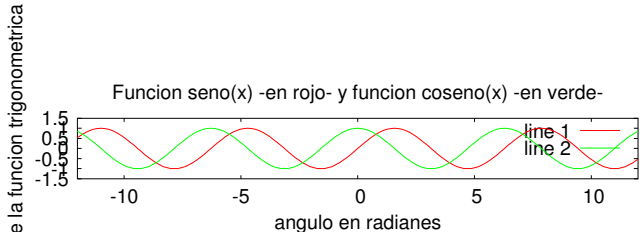
```
xlabel('ángulo en radianes'),replot
ylabel('valor de la función trigonométrica'), replot
```



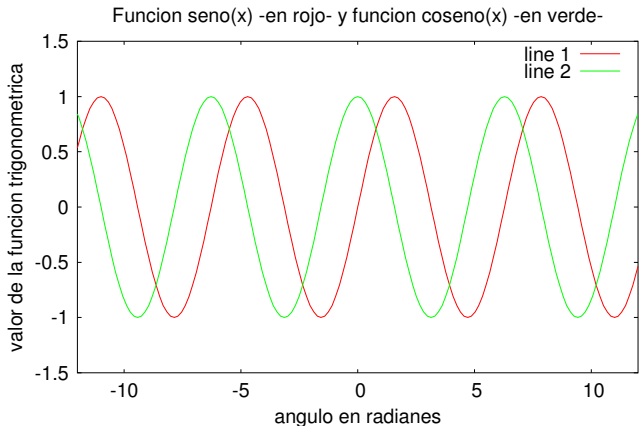

```
axis([-12,12,-1.5,1.5])
replot
```



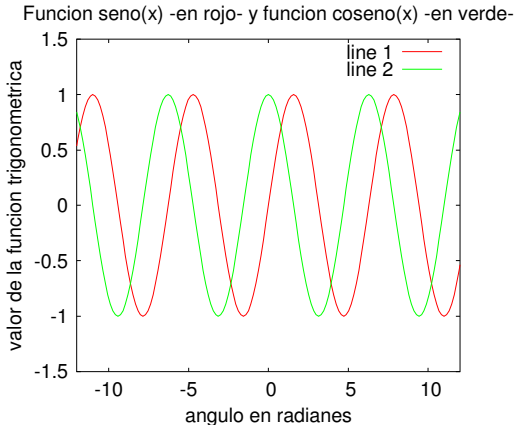

```
axis('equal')
replot
```



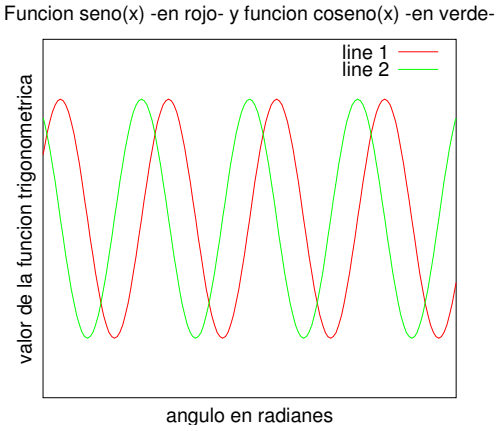

```
axis('normal')
replot
```



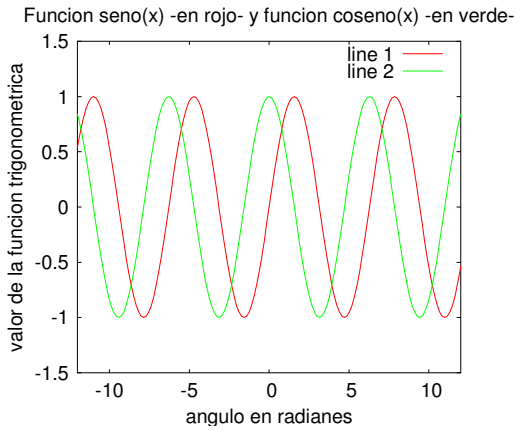

```
axis('square')
replot
```




```
axis('off')
replot
```




```
axis('on')
```



6.2.3.2 Gráficas en 3D

La primera forma de gráfico 3D es la función **gsplot()**, que es el análogo tridimensional de la función **gplot**() basada en los conceptos de GNUPlot y no los del **plot**() MATLAB. La sintaxis general es la siguiente:

gsplot(*rangos, expresión, using, titulo, estilo*)

Los argumentos *rangos*, *using*, *título*, y *estilo* son opcionales, y los *cualificadores using*, *título*, y *estilo* pueden aparecer en cualquier orden después de la *expresión.* Se pueden graficar múltiples expresiones con un único comando separándolas por comas (,). Cada expresión puede tener su propio conjunto de cualificadores .

El ítem opcional *rangos* tiene la siguiente sintaxis:

[x_inf : x_sup] [y_inf : y_sup] [z_inf : z_sup]

Que puede ser usado para especificar los rangos de los ejes del gráfico, con independencia del rango real de los datos. El rango para los ejes *y* y *z* y cualquiera de los límites individuales pueden ser omitidos. Un rango `**[:]**' indica que se deben usar los límites por defecto. Esto normalmente significa que el rango que se va a usar contiene exactamente a los datos del gráfico.

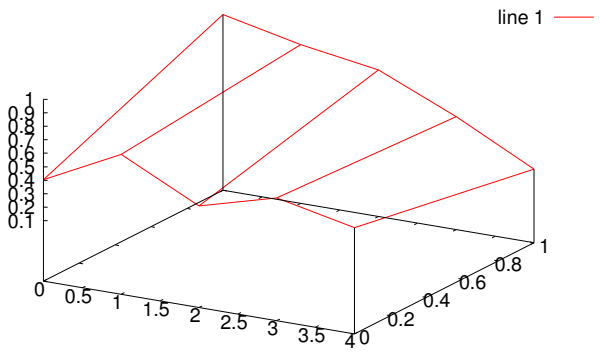
La *expresión* a ser graficada no debe contener ninguna matriz constante literal (por. Ej. `[1, 2; 3, 4]') ya que es prácticamente imposible distinguir un rango de graficación de una matriz de datos.

6.2.3.2.1 Graficación de superficies con coordenadas de enteros.

Para una descripción más detallada de la función **gsplot** se puede consultar el *help* de GNU-Plot que queda accesible desde OCTAVE cuando realiza un gráfico y se abre una ventana del mismo (previa a la del gráfico) y allí buscar en el Index la función **splot,** con la cual se puede crear gráficos a partir de un archivo de datos o de una función.

Por defecto el comando **gsplot()** grafica cada columna de la *expresión* como el valor de *z* usando los índices de fila como el valor de *x* y los de columna como el de *y.* Los índices se cuentan a partir de cero (0). Por ejemplo:

```
clearplot
gsplot rand (5, 2)
```



Grafica una ***superficie*** aleatoria con los valores de *x* y *y* tomados de las cinco (5) filas (0..4) y las dos (2) columnas (0..1) de la matriz.

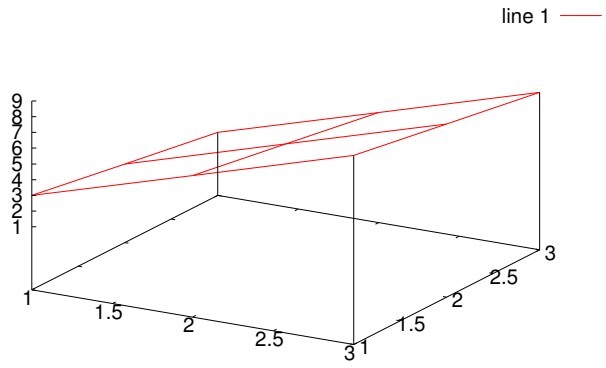
6.2.3.2.2 Graficación de líneas con parámetros.

Si se establece la graficación *paramétrica* usando el comando **gset** *parametric*, **gsplot** toma tres (3) las columnas de la matriz de la *expresión,* considerándolas

como los valores de *x*, *y* y *z* que definen una ***línea*** en un espacio tridimensional. Otras columnas adicionales son ignoradas y los valores de las columnas de *x* y *y* deben estar *ordenados crecientemente.*

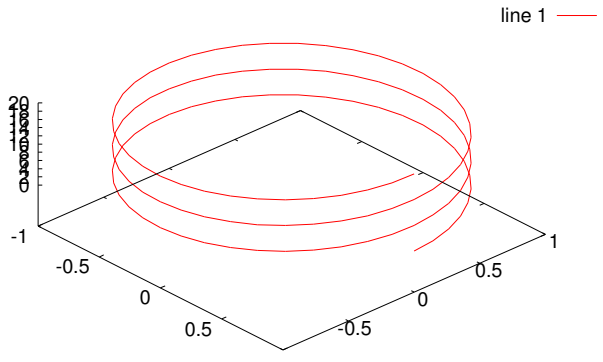
Por ejemplo:

```
M = [ 1 1 3 2 1 6 3 1 9
      1 2 2 2 2 5 3 2 8
      1 3 1 2 3 4 3 3 7]
gset parametric
gsplot M
```



Otro ejemplo de una línea continúa en espiral:

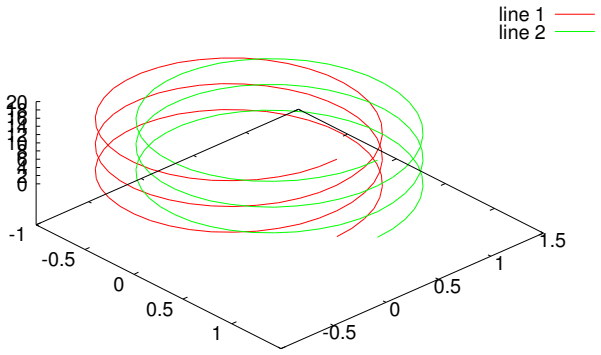
```
fi=[0:pi/20:6*pi]';
cfi = cos(fi);
sfi =  sin(fi);
gsplot([cfi,sfi,fi]);
```



Se pueden superponer varias líneas como en el comando plot con:

```
hold on
gsplot([cfi+0.2,sfi+0.2,fi]);
```

```
hold off
```

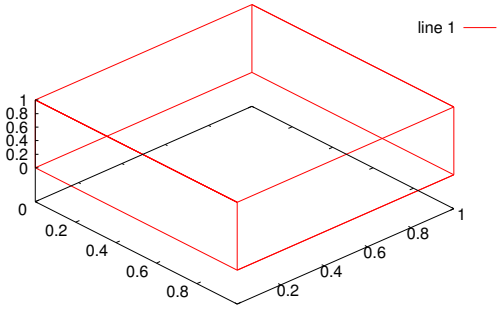


Ejemplo sencillo consistente en dibujar un cubo. Se crea un archivo llamado **Cubo.m** que contenga las aristas correspondientes, definidas mediante los vértices del cubo como una *línea poligonal continua* (obsérvese que algunas aristas se dibujan dos veces). El archivo cubo.m define una matriz A cuyas columnas son las coordenadas de los vértices, y cuyas filas son las coordenadas x, y y z de los mismos:

```
%Procedimiento Cubo.m
A=[0 1 1 0 0 0 1 1 1 1 0 0 0 0 1 1
   0 0 1 1 0 0 0 0 1 1 1 1 1 0 0 1
   0 0 0 0 0 1 1 0 0 1 1 0 1 1 1 1 ];
```

Ahora basta ejecutar los comandos siguientes (el trasponer los vectores en este caso es opcional, por la simetría del cubo):

```
closeplot
Cubo;
gsplot([A(1,:)',A(2,:)',A(3,:)'])
```



6.2.3.1.3 Dibujo de mallados: funciones mesh, meshgrid y meshdom.

Se puede dibujar una función de dos variables (z=f(x,y)) sobre un dominio rectangular. Sean *x* e *y* dos vectores que contienen las coordenadas en una y otra dirección de la retícula (grid) sobre la que se va a dibujar la función. Después hay que crear dos matrices *X* (cuyas filas son copias de x) e *Y* (cuyas columnas son copias de y). Estas matrices se crean con la función **meshgrid**. Estas matrices representan respectivamente las coordenadas x e y de todos los puntos de la retícula. La matriz de valores *Z* se calcula a partir de las matrices de coordenadas *X* e *Y*. Finalmente hay que dibujar esta matriz Z con la función **mesh**, cuyos elementos son función elemento a elemento de los elementos de *X* e *Y*. Los comandos de graficación 3D al estilo de MATLAB son los siguientes:

mesh(X, Y, Z)

Grafica una malla dadas las matrices X e Y desde **meshdom** y una matriz Z correspondiente a las coordenadas X y *Y* de la malla Si *X* e *Y* son vectores, luego un *vértice* típico es (X(j), Y(i), Z(i,j)). Así las columnas de Z corresponden a diferentes valores de *X* y las filas de *Z* corresponden a diferentes valores de *Y.*

[X, Y] = **meshgrid**(x, y)

[X, Y] = **meshgrid**(x)

Dados los vectores de coordenadas *x* e *x*, retorna dos matrices correspondientes a las coordenadas *x* e *y* de una *malla*. Las filas de *X* son copias de *x,* y las columnas de *Y* son copias de *y*.

[X,Y] = **meshdom**(x, y)

Dados los vectores de coordenadas x e *y* , retorna dos matrices correspondientes a las coordenadas *X* e *Y* de la *malla*. Nota: esta función es provista solo para compatibilidad con MATLAB.

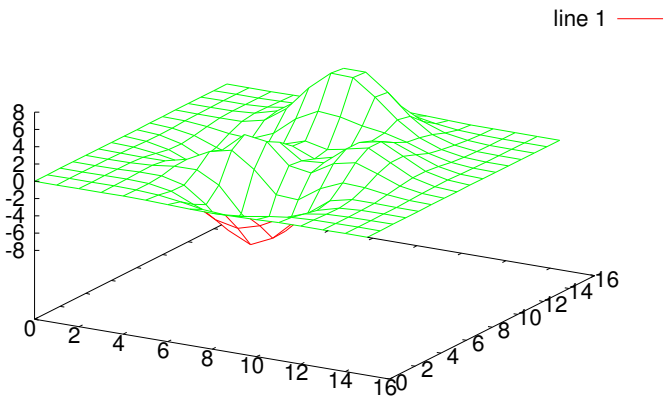
Se define una función *z* de dos variables en *x* e *y* en un archivo llamado **Test3d.m**:

```
function z=Test3d(x,y)
%TEST3D función de dos variables utilizada para probar
%       gráficos 3D
    z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
         - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
    - 1/3*exp(-(x+1).^2 - y.^2);
end
```

Para graficarla se llama al procedimiento **PTest3d.m**

```
% Procedimiento Ptest3d.m
x = [-3:0.4:3]; y = x;
closeplot
[X,Y] = meshgrid(x,y);
Z=Test3d(X,Y);
mesh(Z)
```

```
Ptest3d
```

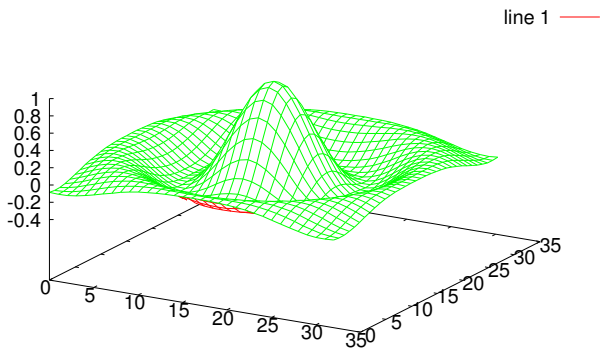


Véase como ejemplo el dibujo de la función sen(r)/r ; con r = sqrt($x^2 + y^2$); para evitar dividir por cero (0) se suma al denominador el número pequeño **eps**. Para distinguirla de la función Test3d anterior se utilizará *u* y *v* en lugar de *x* e *y*. Crear un archivo llamado **Sombrero.m** que contenga las siguientes líneas:

```
% Procedimiento Sombrero.m
closeplot
 u=-8:0.5:8; v=u;
 [U,V]=meshgrid(u,v);
 R=sqrt(U.^2+V.^2)+eps;
W=sin(R)./R;
mesh(W)
```

Ejecutando este archivo se obtiene el gráfico 3D de la función *sombrero* con facetas.

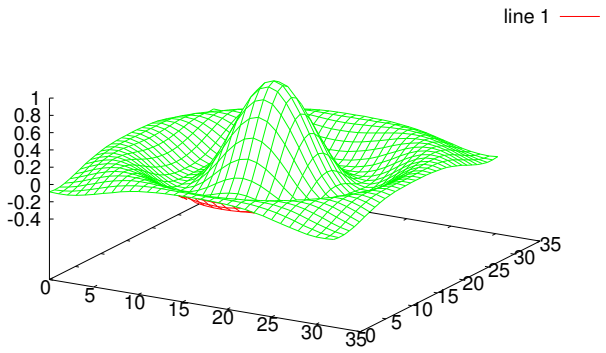
```
Sombrero
```



La función mesh dibuja en perspectiva una función en base a una retícula de líneas de colores, rodeando cuadriláteros del color de fondo, con eliminación de líneas ocultas.

Ejecútese ahora el comando:

```
surf(W)
```



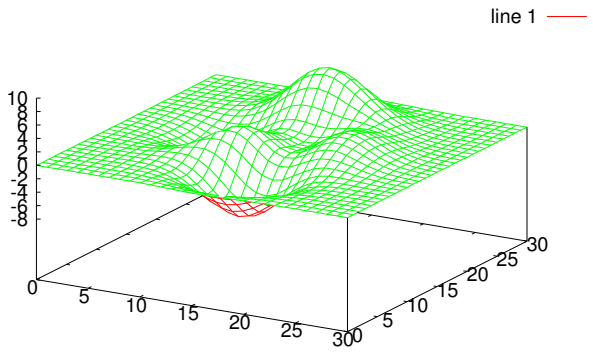
Se observa que en vez de líneas aparece ahora una superficie faceteada, también con eliminación de líneas ocultas. (En OCTAVE **surf()** solo replica la función **mesh().**

Como un segundo ejemplo, se va a volver a dibujar la función **Test3d()** (la correspondiente al archivo **Test3d.m** visto previamente). Ahora se crea el archivo **Picos.m** con las siguientes sentencias:

```
% Procedimiento Picos.m
  x=[-3:0.2:3];
  y=x;
  [X,Y]=meshgrid(x,y);
  Z=Test3d(X,Y);
  for i=1:10
     mesh(Z), pause(0.2), surf(Z), pause(0.2)
  end
```

Es necesario poner la instrucción pause que espera 1 segundos para que se puedan ver las dos formas de representar la función Z (si no, sólo se vería la segunda).

```
Picos
```

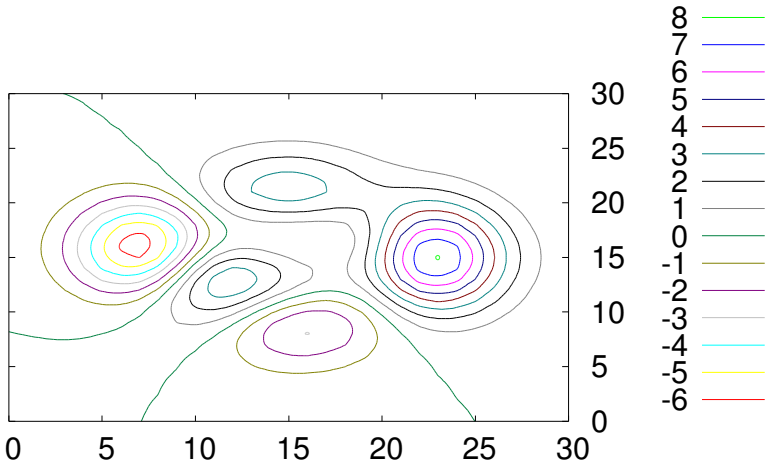


6.2.3.1.4 Dibujo de líneas de contorno: función contour().

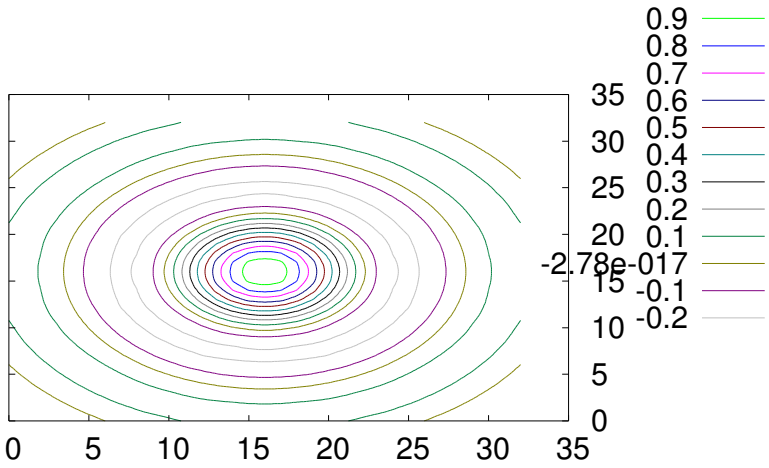
Una forma distinta de representar funciones tridimensionales es por medio de isolíneas o curvas de nivel. Se pueden utilizar estas representaciones con las matrices de datos Z y W que se han calculado previamente:

```
closeplot
contour(Z,20),  pause(2)
```

Donde "20" representa el número de líneas de nivel. Si no se pone se utiliza un número por defecto.




```
contour(W,20),  pause(2)
```



Otras posibles formas de estas funciones son las siguientes:

contour(Z, val) siendo val un vector de valores para las isolíneas a dibujar

contour(u,v,W,20) se utilizan u y v para dar valores a los ejes de coordenadas

Capítulo VII
Comparación de las herramientas Matlab y Octave.

7.1 Aspectos de evaluación.

Se evaluaran las características más relevantes en los programas Matlab y Octave. Para ello se pide que se evalúe de 1 al 4, donde 1 = Malo, 2 = Regular, 3 = Bueno, 4 = Muy bueno, cada uno de los aspectos que a continuación se detallan.

1. Funcionalidad.

1.1. Funcionalidades básicas.

Matrices (operaciones con matrices, vectores y valores propios).

1.2. Funcionalidades avanzadas.

Funciones más específicas (equivalente a los ToolBox de MatLab), como por ejemplo redes neuronales, procesado de imagen y de señal.

1.3. Gráficos e imágenes.

Posibilidad de mostrar gráficos de funciones e imágenes.

1.4. Potencia del lenguaje de programación.

Tipos de datos, estructuras de control, orientación a objetos, modularidad.

2. Confiabilidad.

2.1. Control de la precisión.

Control de la precisión de los resultados.

2.2. Fiabilidad.

Existencia de 'bugs', 'cuelgues' del programa.

3. Eficiencia.

3.1. Rapidez.

Velocidad del procesamiento de datos.

4. Facilidad de uso.

4.1. Información.

Manuales, libros, ayuda dentro del programa, información en internet.

4.2. Facilidad de manejo.

La facilidad para introducirse en el programa sin un conocimiento previo.

5. Mantenimiento.

5.1. Licencia y facilidad de obtención.

Tipo de licencia, acceso al programa, precio, versiones de estudiante.

5.2. Desarrollo y madurez del proyecto.

Aquí englobamos aspectos como existencia de versiones nuevas, la frecuencia con la que salen nuevas versiones, compatibilidad entre ellas.

6. Portabilidad.

6.1. Instalación.

Facilidad en la instalación, necesidad de saberse 'trucos'.

6.2. Compatibilidad con otros programas y formatos estándar.

Posibilidad de adaptarse o comunicarse con otros programas, uso de formatos estándares.

6.3. Integración de otros lenguajes.

Integración de código a otros lenguajes.

Característica	MatLab	Octave
Funcionalidades básicas	4	4
Funcionalidades avanzadas	4	4
Gráficos e imágenes	4	4
Potencia del Lenguaje de Programación	3	3
Control de la precisión	3	3
Fiabilidad	4	4
Rapidez	4	4
Información	4	2
Facilidad de manejo	4	3
Licencia y facilidad de obtención	1	4
Desarrollo y madurez	4	4
Instalación	3	4
Compatibilidad con otros programas	4	4
Integración de otros Lenguajes de Programación	3	4

Tabla 7.1 Comparación entre las herramientas Matlab y Octave.

Programa	Puntos Fuertes	Puntos Débiles
Matlab	Gran número de funciones y ToolBox. Diseño apoyado en interfaces gráficas. Información sobre el programa.	Licencia
Octave	Gran cantidad de funciones y paquetes. Alta compatibilidad con MatLab. Licencia.	Información sobre el programa.

Tabla 7.2 Puntos fuertes y débiles de los programas analizados.

7.2 COMPARACIÓN ECONÓMICA.

En la Tabla 7.3 se presenta la cotización más actualizada sobre la suscripción al servicio de mantenimiento de software, la cual muestra los cargos por restablecimiento ("*reinstatement*") y por la compra del servicio de mantenimiento, para un período equivalente al lapso transcurrido sin la suscripción, más un año. El servicio de mantenimiento de software es equivalente a poder contar en todo momento con la versión más reciente de todos los productos adquiridos.

No. Lic.	Producto	Reingreso (reinstatement)		Servicio de mantenimiento de Software					TOTAL por Producto
				Periodo de Mantenimiento		No. Meses	Costo Mensual	Total Serv. Mant.	
		Precio Unitario	Total por Reingreso	Inicio	Final				
55	MATLAB®	$6.00	$330.00	01-Ene 2001	01-Ene 2007	72	$2.00	$7,920.00	$8,250.00
55	Simulink®	$3.00	$165.00	01-Ene 2001	01-Ene 2007	72	$1.00	$3,960.00	$4,125.00
5	Communications Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
50	MATLAB® Compiler	$6.00	$300.00	01-Ene 2001	01-Ene 2007	72	$2.00	$7,200.00	$7,500.00
5	Control System Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Data Acquisition Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Signal Processing Blockset	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
50	Excel Link	$1.75	$87.50	01-Ene 2001	01-Ene 2007	72	$0.58	$2,088.00	$2,175.50
5	Fuzzy Logic Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Simulink® Fixed Point	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	System Identification Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Image Processing Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Neural Network Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Fixed-Point Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
50	SimPowerSystems	$6.00	$300.00	01-Ene 2001	01-Ene 2007	72	$2.00	$7,200.00	$7,500.00
5	Real-Time Workshop®	$17.50	$87.50	01-Ene 2001	01-Ene 2007	72	$5.83	$2,098.80	$2,186.30
5	Simulink® Control Design	$0.00	$0.00	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$687.60
5	Stateflow®	$17.50	$87.50	01-Ene 2001	01-Ene 2007	72	$5.83	$2,098.80	$2,186.30
5	Signal Processing Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Statistics Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Communications Blockset	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
	TOTAL por rubro	**$132.50**	**$1,731.25**				**$45.98**	**$42,192.00**	**$43,923.25**

Tabla 7.3 Costos de renovación del servicio de mantenimiento

El rubro por restablecimiento (*reinstatement*) es un cargo que MathWorks aplica a los clientes que no han renovado el servicio de mantenimiento dentro de los tres meses siguientes a la fecha de vencimiento de la suscripción anterior. Por lo tanto, si la renovación a la suscripción de mantenimiento se realiza en los plazos establecidos, el único cargo es el del propio servicio de mantenimiento, que para el caso de 12 meses, y según la clase de productos y cantidad de licencias de la Universidad, sería igual a $7,032.00 (según puede deducirse de la Tabla 7.3). Esta cantidad anual equivale aproximadamente al pago de laboratorio de 243 alumnos. Por otro lado, debe explorarse también la alternativa mencionada arriba, que consiste en no suscribirse al servicio de mantenimiento, y comprar las versiones actualizadas de los productos de interés, al precio actual (es decir, no se trataría de una actualización de los productos ya adquiridos, sino de la compra de los productos actualizados). En la Tabla 7.4 se muestran los costos de los productos actualizados, considerando la misma cantidad y tipo de productos adquiridos originalmente en la Universidad.

No. Lic.	**Producto**	**Costo unitario**	**TOTAL por Producto**
55	MATLAB®	$120.00	$6,600.00
55	Simulink®	$60.00	$3,300.00
5	Communications Toolbox	$115.00	$575.00
50	MATLAB® Compiler	$120.00	$6,000.00
5	Control System Toolbox	$115.00	$575.00
5	Data Acquisition Toolbox	$115.00	$575.00
5	Signal Processing Blockset	$115.00	$575.00
50	Excel Link	$35.00	$1,750.00
5	Fuzzy Logic Toolbox	$115.00	$575.00
5	Simulink® Fixed Point	$115.00	$575.00
5	System Identification Toolbox	$115.00	$575.00
5	Image Processing Toolbox	$115.00	$575.00
5	Neural Network Toolbox	$115.00	$575.00
5	Fixed-Point Toolbox	$115.00	$575.00
50	SimPowerSystems	$120.00	$6,000.00
5	Real-Time Workshop®	$350.00	$1,750.00
5	Simulink® Control Design	$115.00	$575.00
5	Stateflow®	$350.00	$1,750.00
5	Signal Processing Toolbox	$115.00	$575.00
5	Statistics Toolbox	$115.00	$575.00
5	Communications Blockset	$115.00	$575.00
		TOTAL	**$35,200.00**

Tabla 7.4 Costo de los productos actualizados.

Al comparar los totales de ambas alternativas (Tabla 7.3 y Tabla 7.4), puede observarse que actualmente es preferible comprar los productos otra vez, que actualizarlos por medio del servicio de mantenimiento. Esto se debe a que el lapso transcurrido sin suscripción al servicio de mantenimiento ha sido muy largo.

La inversión es demasiado elevada para realizarla, es preferible optar por La utilización de GNU Octave que cubre las mismas necesidades que Matlab en las asignaturas de Matemáticas de la Universidad Don Bosco y a la vez no hay necesidad de invertir en nuevo hardware, ya que GNU Octave 2.1.73 puede operar perfectamente en el equipo actual.

CONCLUSIONES.

- Octave resulta ser una alternativa económicamente viable al momento de resolver un problema mediante métodos numéricos y como se pudo observar tiene un tipo de codificación muy similar a los lenguajes tradicionales como C++ y Java, además resulta perfecto para aquellos que ya han trabajado con lenguajes como MatLab anteriormente debido a que su sintaxis es muy similar.

- Se pudo determinar que Octave es factible para la implementación en las asignaturas de matemáticas de la Universidad Don Bosco.

- Se encontró una alta compatibilidad entre MatLab y Octave en cada uno de los aspectos evaluados.

GLOSARIO.

Ada: es un lenguaje de programación estructurado y fuertemente tipado de forma estática que fue diseñado por Jean Ichbiah de CII Honeywell Bull por encargo del Departamento de Defensa de los Estados Unidos. Es un lenguaje multipropósito, orientado a objetos y concurrente, pudiendo llegar desde la facilidad de Pascal hasta la flexibilidad de C++.

Bug: es un error de software (computer bug en inglés), es el resultado de un fallo de programación inducido durante el proceso de creación de programas de ordenador o computadora (software). El término bug fue acreditado erróneamente a Grace Murray Hopper, una pionera en la historia de la computación, pero Thomas Edison ya lo empleaba en sus trabajos para describir defectos en sistemas mecánicos por el año 1870.

C: es una herramienta de programación de tipo general, utilizada para el desarrollo del sistema operativo Unix. Fue realizado a principios de la década de los setenta por Dennis Ritchie, como evolución del lenguaje B que creara Ken Thompson.

C++: Versión de C orientada a objetos creada por Bjarne Stroustrup. C++ se ha popularizado porque combina la programación tradicional en C con programación orientada a objetos.

Compilador: Programa traductor que genera lenguaje máquina a partir de un lenguaje de programación de alto nivel basado en el lenguaje. Programa capaz de traducir un código fuente, escrito en el lenguaje de alto nivel que sea, a un código objeto escrito en lenguaje de maquina.

GCC: GCC es, por definición, el compilador de C para Linux. es una colección de compiladores, diseñados dentro del Proyecto GNU. La primera versión fue escrita por Richard Stallman. GCC funciona en muchas arquitecturas y sistemas operativos.

General Public License: (Licencia Pública General) que regula los derechos de autor de los programas de software libre. Está promovida por la Free Software Foundation (FSF) en él dentro del proyecto GNU, y permite la distribución, copia, modificación y uso de programas, incluso cobrando por ello, pero no permite la apropiación o la patente de estas aplicaciones.

Gnome: es un entorno de escritorio para sistemas operativos de tipo Unix bajo tecnología X Window, se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.

GNU: Proyecto nacido en 1984 para desarrollar un sistema operativo similar a UNIX, pero bajo el concepto de software libre. En sí, GNU es un acrónimo recursivo que significa "GNU No es Unix".

Gnuplot: es un programa interactivo para dibujar dirigido mediante comandos.

Hardware: conjunto de elementos materiales que componen un ordenador. En dicho conjunto se incluyen los dispositivos electrónicos y electromecánicos, circuitos, cables, tarjetas, armarios o cajas, periféricos de todo tipo y otros elementos físicos.

Intérprete: un intérprete acepta programas escritos en un lenguaje de alto nivel, los analiza y los ejecuta bajo control del propio intérprete. En este caso, no se genera un programa equivalente en otro lenguaje, como ocurre con un compilador por lo que, si se desea repetir la ejecución del programa, es preciso volver a traducirlo.

Kernel: el núcleo (también conocido en español con el anglicismo kernel, de raíces germánicas como kern) es la parte fundamental de un sistema operativo. Es el

software responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma más básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema. Como hay muchos programas y el acceso al hardware es limitado, el núcleo también se encarga de decidir qué programa podrá hacer uso de un dispositivo de hardware y durante cuanto tiempo, lo que se conoce como multiplexado.

Lenguaje de programación orientado a objetos: Se le llama así a cualquier lenguaje de programación que implemente los conceptos definidos por la programación orientada a objetos. La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.
Librería STL: librería estándar de plantillas.

Linux: es un sistema operativo y un núcleo. Es uno de los paradigmas del desarrollo de software libre (y de código abierto), donde el código fuente está disponible públicamente y cualquier persona puede libremente usarlo, modificarlo y/o redistribuirlo.

Navegador web: es una aplicación software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web de todo el mundo a través de Internet. Esta red de documentos es denominada World Wide Web (WWW) o Telaraña Mundial. Los navegadores actuales permiten mostrar y/o ejecutar: gráficos, secuencias de vídeo, sonido, animaciones y programas diversos además del texto y los hipervínculos o enlaces.

Matlab: es la abreviatura de Matrix Laboratory (laboratorio de matrices). Es un programa de matemáticas creado por The MathWorks en 1984. Está disponible para las plataformas Unix, Windows y MAC.

MathWorks: fabricante de Matlab.

Motor de base de datos: permiten trabajar ya sea con base de datos existentes o creando nuevas con todas las capacidades de trabajo en red

Mozilla: es un Navegador web y una plataforma de desarrollo libre y de código abierto para la WWW. A este hojeador se le denomina habitualmente navegador.

Octave: es un software de distribución gratuita que se apega a la filosofía GNU. Permite programar y utilizar una serie de funciones principalmente numéricas.

OpenGL: es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. Fue desarrollada por Silicon Graphics Inc. (SGI) en 1992. Su nombre viene del inglés Open Graphics Library, cuya traducción es biblioteca de gráficos abierta (o mejor, libre, teniendo en cuenta su política de licencias).

Postgres: es un servidor de base de datos relacional libre. Es una alternativa a otros sistemas de bases de datos de código abierto (como MySQL, Firebird y MaxDB), así como sistemas propietarios como Oracle.

QtOctave: es un front-end para Octave.

Scripts: es un conjunto de comandos escritos en un lenguaje interpretado para automatizar ciertas tareas de aplicación. A veces, se utiliza también el término inglés "scenario".

Servidor Web Apache: es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 (RFC 2616) y la noción de sitio virtual.

Shell: Parte fundamental de un sistema operativo encargada de ejecutar las órdenes básicas para el manejo del sistema. También se denomina shell. Suelen incorporar características tales como control de procesos, redirección de entrada / salida y un lenguaje de órdenes para escribir programas por lotes o (scripts).

Sistema operativo: Un sistema operativo (SO) es un conjunto de programas o software destinado a permitir la comunicación del usuario con un ordenador y gestionar sus recursos de manera cómoda y eficiente. Comienza a trabajar cuando se enciende el ordenador, y gestiona el hardware de la máquina desde los niveles más básicos.

Software libre: es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o a precio del coste de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente.

Software propietario: se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones), o cuyo código fuente no está disponible o el acceso a éste se encuentra restringido.

Unix: es un sistema operativo no libre muy popular, porque está basado en una arquitectura que ha demostrado ser técnicamente estable. Es un sistema operativo portable, multitarea y multiusuario; desarrollado en principio por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

WidgetServer: es un programa para la creación de ventanas. Se pueden crear ventanas desde prácticamente cualquier lenguaje de programación, incluido Octave.

BIBLIOGRAFÍA.

- López, Jorge. **USO DEL SOFTWARE MATLAB**.
  El Salvador. 2004

- Gilat, Amos. **MATLAB: UNA INTRODUCCIÓN CON EJEMPLOS PRACTICOS**. Editorial Reverte. 2004

- García Rojo, Juan José. **HERRAMIENTAS EN GNU / LINUX PARA ESTUDIANTES UNIVERSITARIOS**. 2003

- Raymond, Eric S. **LA CATEDRAL Y EL BAZAR**. 1998

- Pérez César. **MATLAB Y SUS APLICACIONES EN LAS CIENCIAS Y LA INGENIERIA.** Editorial Prentice Hall. 2002

- Gómez Herrera, Wendy Guadalupe. **SOFTWARE LIBRE VS SOFTWARE PROPIETARIO.** México, Mayo 2006

- Hernández, Nereyda. **REPORTE DEL ESTADO DEL LABORATORIO DE SIMULACIÓN Y MATEMÁTICA.**

DIRECCIONES ELECTRÓNICAS.

- **http://www.octave.org**

  Página oficial de Octave.

- **http://qtoctave.wordpress.com/**

  Página principal de QtOctave.

- **http://www.efn.uncor.edu/departamentos/computacion/materias/informatica/software.html**

  Página de la Universidad de Córdoba que permite bajar el instalador de Octave para Windows.

- **http:// www.mastermagazine.info**

  Buscador de términos para glosario técnico.

- **http://es.wikipedia.org**

  Buscador de términos para glosario técnico.

- **http://webmaster.lycos.es/glossary/S/**

  Buscador de términos para glosario técnico.

- **http://www.mouse.cl/2005/rep/06/16/index.asp**

  Glosario los principales términos usados en Linux

ANEXOS

Anexo 1: Programas de las asignaturas de matemáticas de la Universidad Don Bosco.

A continuación se presentan los programas de las asignaturas de matemáticas, estos contienen todos los temas del programa actual de la asignaturas.

PROGRAMA
MATEMATICA I

A GENERALIDADES

- ☑ CÓDIGO: FMAT 101
- ☑ PRE-REQUISITO: BACHILLERATO
- ☑ CICLO : 01
- ☑ AÑO: 2007
- ☑ UNIDADES VALORATIVAS : 4
- ☑ DURACIÓN CICLO/SEM: 16
- ☑ DURACIÓN DE HORA CLASE: 50 Min.
- ☑ N° HORAS CICLO: 112

B DESCRIPCIÓN DE LA ASIGNATURA

Comprende cinco unidades, la primera consta de la teoría de matrices y determinantes, así como sus aplicaciones; la segunda, muestra un estudio del tipo de funciones trascendentales y sus gráficas; la tercera unidad, comprende un estudio amplio de la continuidad y límites de funciones de variable real; la cuarta unidad aborda la teoría de la diferenciación para cualquier tipo de función; la quinta unidad comprende algunas de las aplicaciones de la derivada de una función de variable real.

C OBJETIVO GENERAL

Proporcionar al estudiante los conocimientos sobre el álgebra matricial, teoría de funciones y el cálculo diferencial, necesarios en la resolución de problemas de aplicación práctica del área especializada. Además se implementará el uso de

software de matemática para que el estudiante se familiarice con él y pueda aplicarlo adecuadamente para solucionar aplicaciones futuras en otras áreas que se lo permitan y para comprobar o verificar los resultados obtenidos mediante los métodos tradicionales que se evalúan.

D CONTENIDO PROGRAMATICO

UNIDAD I: 1 MATRICES Y DETERMINANTES

OBJETIVOS ESPECIFICOS:

Que al finalizar esta unidad el alumno sea capaz de:

- Identificar los tipos de matrices (cuadrada, rectangular, fila, columna, identidad, triangular, transpuesta, simétrica).
- Aplicar el concepto de igualdad de matrices para resolver ecuaciones matriciales.
- Resolver las diferentes operaciones matriciales.
- Utilizar las propiedades de la suma y producto de matrices, como también las del producto de una matriz por un escalar.
- Calcular determinantes aplicando el método de Cofactores y mediante la regla de Sarrus.
- Calcular la matriz inversa por el método de la matriz adjunta y por el método de Gauss.
- Resolver sistemas de ecuaciones lineales por los métodos: de Gauss, matriz inversa, Cramer.

CONTENIDO:

1.1 Definición de Matriz.
1.2 Tipos de matrices.
1.3 Igualdad de matrices.
1.4 Operaciones con matrices.
1.4.1 Suma de matrices.
1.4.2 Propiedades conmutativa y asociativa.
1.4.3 Producto de un escalar por una matriz.
1.4.4 Propiedades del producto de un escalar por una matriz.
1.4.5 Diferencia de matrices.
1.4.6 Producto de matrices.
1.4.7 Propiedades del producto de matrices.
1.5 Determinantes
1.5.1 Definición.

1.5.2 Cálculo de determinantes.
1.5.3 Método de Sarrus para calcular determinantes.
1.5.4 Cálculo de determinantes por el método de Cofactores.
1.5.5 Propiedades de los determinantes.
1.5.6 Matriz Triangular superior e inferior.
1.5.7 Cálculo del determinante de una matriz triangular.
1.6 Inversa de una matriz.
1.6.1 Matriz de Cofactores.
1.6.2 Matriz Adjunta.
1.6.3 Matriz Inversa.
1.6.4 Método de Gauss para calcular la inversa.
1.7 Solución de sistemas de ecuaciones lineales.
1.7.1 Solución de sistemas de ecuaciones lineales por el método de Gauss.
1.7.2 Solución de sistemas de ecuaciones lineales por el método de Cramer.
1.7.3 Solución de sistemas de ecuaciones lineales por el método de la matriz inversa.

UNIDAD II: 2 FUNCIONES REALES

OBJETIVOS ESPECIFICOS:

Que al finalizar esta unidad el alumno sea capaz de:

- Reconocer los diferentes tipos de funciones.
- Establecer dominio, rango y gráfica de los diferentes tipos de funciones.
- Identificar los tipos de simetría.
- Calcular los interceptos con los ejes coordenados.
- Calcular la composición de dos funciones, así como su dominio y rango.
- Calcular la inversa de una función, así como su dominio y rango, y relacionarla con la función directa.

CONTENIDO:

2.1 Definición de función.
2.2 Dominio y rango de una función.
2.3 Función Par, Impar.
2.3.1 Criterios de Simetría.
2.3.2 Interceptos con los ejes.
2.4 Gráfica de funciones.
2.5 Tipos de funciones.
2.5.1 Función polinomial : constante, lineal, cuadrática, cúbica.
2.5.2 Función Racional.
2.5.3 Función Radical.
2.5.4 Función Seccionada.

2.5.5 Función Valor Absoluto.
2.5.6 Función Escalón unitario.
2.5.7 Función signo.
2.6 Operaciones con funciones: suma, producto, cociente.
2.7 Composición de funciones.
2.8 Inversa de una función.
2.9 Función Exponencial.
2.10 Función Logarítmica.
2.11 Funciones Trigonométricas.
2.12 Funciones Trigonométricas Inversas.

UNIDAD III: 3 LIMITE Y CONTINUIDAD

OBJETIVOS ESPECIFICOS:

Que al finalizar esta unidad el alumno sea capaz de:

- Aplicar la definición de límite para verificarlos
- Resolver límites de funciones mediante la noción intuitiva y por teoremas.
- Calcular límites laterales.
- Establecer la existencia del límite, verificando la igualdad de los límites laterales.
- Calcular límites infinitos y al infinito.
- Calcular asíntotas verticales y horizontales de una función.
- Graficar funciones auxiliándose de límites y asíntotas
- Calcular límites trigonométricos y de funciones logarítmicas y exponenciales.
- Establecer si una función es continua en un número y en un intervalo.
- Identificar los tipos de discontinuidad y distinguir uno de otro.
- Modificar una función que presente discontinuidad removible, para que sea continua.

CONTENIDO:

3.1 Definición informal de límite.
3.2 Definición formal de límite.
3.3 Propiedades de los límites.
3.4 Cálculo de límites de funciones polinomiales, racionales y radicales.
3.5 Técnicas para evaluar límites.
3.5.1 Técnica de Cancelación.
3.5.2 Técnica de Racionalización.
3.6 Límites laterales.
3.6.1 Límite lateral por la derecha.
3.6.2 Límite lateral por la izquierda.
3.6.3 Existencia del límite.

3.7 Límites infinitos.
3.7.1 Definición de límites infinitos.
3.7.2 Definición de asíntota vertical.
3.7.3 Propiedades de límites infinitos.
3.8 Límites al infinito.
3.8.1 Definición de límites al infinito.
3.8.2 Definición de asíntota horizontal.
3.9 Límites de funciones trigonométricas.
3.9.1 Límites trigonométricos especiales.
3.10 Límites de funciones logarítmicas.
3.11 Límites de funciones exponenciales.
3.12 Continuidad de una función.
3.12.1 Continuidad en un punto.
3.12.2 Continuidad en un intervalo abierto y cerrado.
3.13 Tipos de discontinuidad.
3.14 Propiedades de funciones continuas.

UNIDAD IV: 4 LA DERIVADA

OBJETIVOS ESPECIFICOS:

Que al finalizar esta unidad, el alumno sea capaz de:

- Calcular la derivada de una función aplicando la definición.
- Calcular las derivadas de funciones algebraicas por medio de los teoremas.
- Establecer la ecuación de la recta tangente y recta normal a la gráfica de una función en un punto dado.
- Establecer cuando una función es diferenciable.
- Aplicar la regla de la cadena para derivar funciones compuestas.
- Encontrar derivadas de orden superior.
- Obtener derivadas de funciones implícitas.
- Calcular la derivada de funciones logarítmicas, exponenciales.
- Establecer la derivada de una función aplicando la técnica de derivación logarítmica.

CONTENIDO:

4.1 Interpretación geométrica de la derivada.
4.2 Definición de la derivada.
4.3 Derivadas laterales.
4.4 Diferenciabilidad.
4.5 Teoremas sobre derivación.

4.6 Recta tangente y Recta normal.
4.7 Derivada de una función compuesta (Regla de la cadena).
4.8 Teorema de la potencia para derivar.
4.9 Derivadas de orden superior.
4.10 Derivación implícita.
4.11 Derivada de una función logarítmica.
4.12 Derivación logarítmica.
4.13 Derivada de una función exponencial.
4.14 Derivada de funciones Trigonométricas.
4.15 Derivada de funciones Trigonométricas Inversas.

UNIDAD V: 5 APLICACIONES DE LA DERIVADA

OBJETIVOS ESPECIFICOS:

Que al finalizar esta unidad el estudiante sea capaz de:

- Interpretar la derivada como una razón de cambio.
- Calcular velocidad media e instantánea.
- Calcular aceleración media e instantánea.
- Aplicar los conceptos de velocidad y aceleración para resolver problemas de movimiento con aceleración constante.
- Calcular razones de cambio de dos o más variables que cambian con el tiempo.
- Aplicar la regla de L'Hopital cuando se presente una forma indeterminada a la hora de evaluar un límite.
- Modificar indeterminaciones del tipo: producto, potencias o diferencias a formas indeterminadas de cociente para aplicar L'Hopital.
- Aplicar diferenciales para la resolución de problemas de propagación de error y para cálculo de superficies o capas de volumen.

CONTENIDO:

5.1 Regla de L'Hopital
5.2 Velocidad y Aceleración.
5.2.1 Definición de velocidad media e instantánea.
5.2.2 Definición de aceleración media e instantánea.
5.3 Razones de cambio relacionadas.
5.4 Diferenciales.
5.4.1 Interpretación geométrica del diferencial.
5.4.2 Definición del diferencial.
5.4.3 Fórmulas generales de diferenciales.
5.4.4 Propagación de error.

PROGRAMA
MATEMATICA II

A GENERALIDADES

☑ Nombre de la asignatura	:	**MATEMATICA II**
☑ Código	:	FMAT512
☑ Pre-requisito	:	**MATEMÁTICA I**
☑ Ciclo	:	I
☑ AÑO	:	2006.
☑ Unidades valorativas	:	4
☑ Duración del ciclo/semana	:	16 semanas
☑ Horas Teóricas	:	3
☑ Horas de Discusión	:	2
☑ Número de horas ciclo	:	80
☑ Duración de hora clase	:	50 minutos.

B DESCRIPCIÓN DE LA ASIGNATURA

Este curso ofrece fundamentalmente conocimientos básicos y principales del cálculo diferencial e integral:

1.- Aplicaciones de la derivada de una función de una variable
2.- La integral indefinida y la integral definida de una función de una variable
3.- Aplicaciones de la integral definida (áreas, volúmenes, longitud....)
4.- Integrales impropias
5.- Cálculo de áreas de regiones polares.

C OBJETIVOS GENERALES

- Proporcionarle al estudiante las herramientas básicas sobre matlab.
- Proporcionar al estudiante los conocimientos básicos sobre cálculo integral, necesarios para la resolución de problemas de aplicación prácticas del área especificada.
- Que el estudiante resuelva integrales utilizando los métodos adecuados: regla de la potencia, cambio de variable, por partes, integración trigonométrica, sustitución trigonométrica y fracciones parciales.

- Preparar a los(as) estudiantes para la comprensión de asignaturas de cursos posteriores
- Fomentar en los(as) estudiantes el interés permanente para la obtención de nuevos conocimientos.
- Distinguir las principales aplicaciones de la integral definida
- Lograr que el estudiante adquiera destrezas y habilidades en la resolución de ejercicios y problemas relacionados con el cálculo integral
- Que el estudiante aplique correctamente la integral definida en el cálculo de áreas y volúmenes de sólidos de resolución.
- Que el estudiante calcule correctamente integrales impropias, diferenciándolas de las integrales propias.
- Que el estudiante grafique curvas polares y calcule áreas de regiones polares.
- Lograr que el estudiante adquiera terminología del cálculo diferencial e integral para comprender y expresar el lenguaje de la ciencia y la tecnología

D CONTENIDO

UNIDAD 1 : APLICACIONES DE LA DERIVADA

1.1 Objetivos Específicos.

Lograr que el estudiante:
- Aplique correctamente las reglas de la derivada
- Resuelva problemas que involucre los conceptos de máximo y mínimo de una función
- Construir gráficas de funciones aplicando criterios de primera y segunda derivada

1.2 CONTENIDOS

1.2.1 Valores extremos de funciones.
1.2.2 Teorema max - min para funciones continuas
1.2.3 Definición de valores extremos absolutos (globales)
1.2.4 Definición de valores extremos relativos (locales)
1.2.5 Teorema de la primera derivada para valores extremos locales
1.2.6 Definición de punto crítico
1.2.7 Definición de: función creciente y de función decreciente

1.2.8 Criterio de la primera derivada para funciones crecientes y funciones decrecientes
1.2.9 Criterio de la primera derivada para valores extremos locales.
1.2.10 Definición de concavidad.
1.2.11 Criterio de la segunda derivada para concavidad
1.2.12 Definición de punto de inflexión
1.2.13 Criterio de la segunda derivada para valores extremos locales.
1.2.14 Trazo de la gráfica de f usando f' y f".
1.2.15 Problemas de optimización. Estrategia para resolverlos.

UNIDAD II : LA ANTIDERIVADA

Objetivos Específicos.

Lograr que el estudiante:

Comprenda el concepto de integral indefinida de una función.

• Determine la integral indefinida de una función haciendo uso de las técnicas de integración.

• Aplique correctamente la regla de la potencia en el cálculo de integrales.

• Sea capaz de utilizar la técnica de integración adecuada al ejercicio que se desea resolver.

2.2 CONTENIDOS

2.2.1 Concepto de integral indefinida
2.2.2 Propiedades de la integral indefinida
2.2.3 Fórmulas básicas de integración
2.2.4 Evaluación de integrales básicas.
2.2.5 Regla de la potencia
2.2.6 Técnicas de integración
- Integración por sustitución
- Integración por partes
- Integración con funciones trigonométricas
- Integración por sustitución trigonométrica
- Integración por fracciones parciales.

UNIDAD III : LA INTEGRAL DEFINIDA

Objetivos Específicos

Lograr que el estudiante:
- Interprete geométricamente la integral definida mediante el uso de sumas de Riemann
- Comprenda el concepto de integral definida de una función

- Pueda operar con las propiedades de la integral definida
- Evalúe integrales aplicando correctamente el teorema fundamental del cálculo
- Calcule áreas bajo y entre curvas, usando la integral definida
- Calcule volúmenes de sólidos de revolución, aplicando el método adecuado estudiados.

3.2 CONTENIDOS

3.2.1 Notación Sigma. Propiedades y fórmulas
3.2.2 Integral definida. Propiedades
3.2.3 Teorema fundamental del cálculo integral
3.2.4 Aplicaciones de la integral definida:
- Área bajo y entre curvas
- Volúmenes de sólidos de revolución
  * Método de disco
  * Método de anillo (arandelas)
  * Método de capas (corteza cilíndrica)

3.2.5 Longitud de arco
3.2.6 Área de Superficie de revolución

UNIDAD IV. INTEGRALES IMPROPIAS Y COORDENADAS POLARES

Objetivos Específicos

Lograr que el estudiante:
- Comprenda el concepto de integral impropia.
- Comprenda el concepto de convergencia de una integral impropia.
- Determinar el carácter de convergencia de integrales impropias.
- Calcule correctamente integrales impropias.
- Calcule el área bajo una curva haciendo uso de las integrales impropias.
- Determine el carácter de convergencia de integrales impropias de primera especie haciendo uso de los criterios de convergencia.
- Identifique las características del sistema de coordenadas polares.
- Transforme puntos y ecuaciones polares o cartesianas y viceversa.
- Haga uso de criterios de simetría en el gráfico de curvas polares.
- Determine el área entre curvas polares.

4.2 CONTENIDOS

4.2.1 Definición de integral impropia
4.2.2 Integrales impropias de primera y segunda especie
  4.2.2.1 Con limites infinitos de integración
  4.2.2.2 Con discontinuidades infinitas.
4.2.3 Criterio de convergencia para integrales impropias
4.2.4 Sistema de coordenadas polares

4.2.5 Ecuaciones polares y cartesianas
4.2.6 Simetría y gráfico de curvas polares
4.2.7 Área de una región en coordenadas polares

PROGRAMA
MATEMATICA III

A GENERALIDADES

☑ Nombre de la asignatura:	MATEMATICA III
☑ Código:	MAT513
☑ Pre-requisito:	MATEMATICA II
☑ Ciclo:	I
☑ Año:	2007
☑ Unidades valorativas:	4
☑ Duración Ciclo / Sem:	16
☑ Duración de Hora Clase:	50
☑ N° Horas Ciclo:	80

B DESCRIPCIÓN DE LA ASIGNATURA

En este curso se estudia las superficies en el espacio y sus ecuaciones, la derivada de funciones de varias variables y aplicaciones del diferencial total, la teoría de valores extremos, las integrales dobles y triples, y los sistemas de coordenadas cilíndricas y esféricas.

C. OBJETIVOS

- Que el estudiante sea capaz de calcular áreas y volúmenes utilizando las funciones vectoriales.
- Que el estudiante pueda graficar rectas, y superficies en el espacio, e identifique sus funciones.
- Que el estudiante evalúe límites de funciones de dos o más variables, encuentre las derivadas parciales de cualquier orden de funciones de varias variables, y pueda determinar los puntos extremos de funciones de varias variables.

- Proporcionar al estudiante los conocimientos de integrales iteradas, integrales dobles y triples; coordenadas cilíndricas y esféricas en la resolución de problemas de aplicación práctica.
- Que el estudiante pueda calcular integrales triples por medio de coordenadas cilíndricas y esféricas y pueda dibujar la superficie de integración de un integral triple.

D. CONTENIDO

UNIDAD I: GEOMETRIA ANALITICA Y VECTORES EN EL ESPACIO

Objetivos Específicos

- Que el estudiante pueda graficar vectores en el espacio.
- Que el estudiante pueda realizar operaciones con vectores.
- Que el estudiante identifique los octantes formado en el espacio cartesiano.
- Que el estudiante calcule áreas y volúmenes utilizando el producto vectorial .
- Que el estudiante pueda graficar rectas y planos en el espacio.
- Que el estudiante pueda encontrar la ecuación del plano y de la recta dada una información específica.
- Que el estudiante pueda graficar las superficies en el espacio e identifique sus ecuaciones.
- Que el estudiante pueda encontrar el volumen de un paralelepípedo utilizando el triple producto escalar.

Contenidos

1.Coordenadas y vectores en el espacio.
2.Producto escalar
3.Producto vectorial. Triple producto escalar.
4.Área y volumen de un paralelepípedo.
5.Rectas y planos en el espacio.
6.Superficies cilíndricas
7. Superficies cuadricas

UNIDAD II: CALCULO DIFERENCIAL EN VARIAS VARIABLES

Objetivos Específicos

- Que el estudiante identifique el dominio y rango de funciones dos o más variables.
- Que el estudiante pueda representar el dominio gráficamente de funciones de dos y tres variables independientes.

- Que el estudiante evalúe limite de funciones de dos o más variables.
- Que el estudiante pueda determinar la continuidad de una función de dos o más variables.
- Que el estudiante encuentre las derivadas parciales de cualquier orden de funciones de varias variables.
- Que el estudiante pueda determinar los valores extremos de funciones de varias variables.
- Que el estudiante pueda aplicar el método de multiplicadores de Lagrange para problemas de optimización con restricciones.
- Que el estudiante pueda encontrar la derivada en un punto en cualquier dirección
- Que el estudiante pueda determinar cuando una función es diferenciable y adonde es diferenciable.
- Que el estudiante pueda diferenciar un diferencial con una razón de cambio

Contenido

1.Funciones de dos o más variables
2.Limite y continuidad
3.Derivadas parciales
4.Diferencial total
5.Regla de la cadena, derivación implícita
6.Gradiente y derivada direccional
7.Valores extremos
8.Multiplicadores de Lagrange

UNIDAD III: CALCULO INTEGRAL EN VARIAS VARIABLES

Objetivos Específicos

- Que el estudiante pueda evaluar integrales iteradas.
- Que el estudiante pueda graficar la región de integración de un integral doble.
- Que el estudiante pueda escribir los dos ordenes de integración y evalué el mas sencillo.
- Que el estudiante pueda encontrar áreas por medio de integrales iteradas.
- Que el estudiante pueda encontrar volúmenes a través del integral doble.
- Que el estudiante sea capaz de poder graficar la región de integración ya sea de un integral doble o de un integral triple.
- Que el estudiante pueda transformar integrales de coordenadas rectangulares a polares para poderlo resolver.
- Que el estudiante pueda encontrar volúmenes por medio de integrales triples, dados el sólido o el enunciado del problema.
- Que el estudiante pueda transformar integrales triples de coordenadas rectangulares a coordenadas cilíndricas y esféricas para poderlas calcular.

- Que el estudiante sea capaz de poder dibujar un sólido dada una información
- Que el estudiante pueda identificar en que sistema le conviene evaluar el integral

Contenido

1.Integral doble
2.Integrales iteradas
3.Evaluación de integrales dobles
4.Integrales dobles en coordenadas polares
5.Integral triple en coordenadas rectangulares
6.Coordenadas cilíndricas y esféricas
7.Integrales triples en coordenadas cilíndricas y esféricas

PROGRAMA
MATEMATICA IV

A GENERALIDADES

☑ Nombre de la asignatura:	Matemática IV
☑ Código:	MAT514
☑ Pre-requisito:	Matemáticas III
☑ Ciclo:	I
☑ Año:	2007
☑ Unidades valorativas:	4
☑ Duración Ciclo/Sem:	16
☑ Duración de Hora Clase:	50
☑ N° Horas Ciclo:	80

B DESCRIPCIÓN DE LA ASIGNATURA

En éste curso se estudia las ecuaciones diferenciales ordinarias de primer orden, las ecuaciones diferenciales lineales de orden superior, la transformada de Laplace, sistemas de ecuaciones diferenciales.

C OBJETIVOS GENERALES

- Que el alumno desarrolle la capacidad de razonamiento matemático.
- Proporcionar al estudiante una formación matemática que le permita cursar con éxito asignaturas en el área diferenciada de ingeniería.

D CONTENIDO PROGRAMATICO

UNIDAD I: ECUACIONES DIFERENCIALES ORDINARIAS DE PRIMER ORDEN

Objetivos Específicos

- Que el estudiante identifique el tipo de la ecuación diferencial.
- Que el estudiante resuelva ecuaciones diferenciales de primer orden de cualquier tipo
- Que el estudiante utilice factores integrantes en la solución de una ecuación diferencial.
- Que el estudiante pueda clasificar las ecuaciones diferenciales
- Que el estudiante pueda resolver una ecuación con valores iniciales
- Que determine cuando tiene solución única una ecuación diferencial

Contenidos

1. Definiciones básicas
2. Ecuaciones diferenciales de variables separable
3. Ecuaciones diferenciales exactas
4. Ecuaciones diferenciales homogéneas
5. Ecuaciones diferenciales lineales
6. Ecuaciones de Bernoulli
7. Factores integrantes

UNIDAD II: ECUACIONES DIFERENCIALES LINEALES DE ORDEN SUPERIOR

Objetivos específicos

- Que el estudiante pueda reconocer una ecuación diferencial de orden superior
- Que el estudiante pueda resolver una ecuación diferencial homogénea
- Que el estudiante utilice los métodos de coeficientes indeterminados y variación de parámetros para resolver una E-D no homogénea de orden superior
- Que el estudiante conozca cuando un conjunto de soluciones son linealmente independiente
- Que el estudiante pueda convertir un conjunto de soluciones dependientes en independientes

Contenidos

Ecuaciones diferenciales de orden superior

1. Reducción de orden

2. Ecuaciones diferenciales lineales homogéneas con coeficiente constante
3. Ecuaciones diferenciales lineales no homogéneas
4. Método de coeficiente indeterminado
5. Método de variación de parámetros.

UNIDAD III. LA TRANSFORMADA DE LA PLACE

Objetivos específicos

- Que el estudiante pueda enumerar las condiciones que garantiza la existencia de L{F(t)}
- Que el estudiante pueda calcular la transformada de Laplace por evaluación directa de la integral de la definición, así como también utilizando propiedades.
- Que el estudiante pueda encontrar $L^{-1}\{F(s)\}$ analíticamente .
- Que el estudiante pueda utilizar la transformada de Laplace para resolver E.D. con valores iniciales cuando F(t) sea una función continua o una función seccionadas por partes
- Que el estudiante pueda utilizar el teorema de convolución para encontrar la transformada inversa de Laplace de funciones.
- Que el estudiante pueda escribir las funciones escalonadas aplicando las definición de función escalón unitaria y pueda encontrar la transformada de la place de dichas funciones.
- Que el estudiante identifique la función periódica y pueda encontrar su trasformada
- Que el estudiante pueda graficar cualquier función escalón unitario.

Contenidos

1. Definición de transformada de Laplace
2. Propiedades básicas de la transformada de Laplace
3. Transformada de derivados, integrales
4. Transformada inversa
5. Funciones escalonadas
6. Función periódica
7. Aplicación (uso de la transformada de Laplace para resolver una E.D. de valor inicial)

Anexo 2: Guias de Matematicas I.

	**Guía Nº 1**	**INTTRODUCCIÓN A OCTAVE**
	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **DEPARTAMENTO DE CIENCIAS BÁSICAS**	
	Asignatura : Matemáticas I Lugar de Ejecución : Laboratorio de simulación y matemáticas.	

OBJETIVO.

- Operar de manera básica el programa.

1. ¿Qué es Octave?

Octave es un software que nos permite programar y utilizar una serie de funciones principalmente numéricas. Esto se hace escribimos los comandos a utilizar y el software responderá a cada uno de ellos mostrando los resultados a través de la pantalla. Es decir, cada vez que nosotros escribimos algo el software nos responderá, excepto que finalicemos lo que escribimos con un punto y coma (;).

Octave es un software de distribución gratuita que se apega a la filosofía GNU, esto es poder tener acceso al programa y al código fuente del programa para modificarlo (si nos interesa), sin tener que pagar ni por su uso, ni por su obtención, además de poder hacer cuantas copias se quieran e instalaciones en diferentes máquinas, también, sin tener que pagar.

2 Ejecutando OCTAVE por primera vez

Para ejecutar el programa, simplemente se debe hacer click en el icono correspondiente. Cuando el software se inicia, se muestra una ventana donde

podemos escribir los comandos que necesitemos (Figura 1). Además Octave puede graficar nuestros datos y resultados en otra ventana.

La forma más sencilla de obtener ayuda en Octave es escribiendo help y luego presionando ENTER.

Existe un manual on-line en inglés en la dirección http://www.octave.org/doc/octave_toc.html .

Para salir de Octave bastará con escribir exit y presionar ENTER.

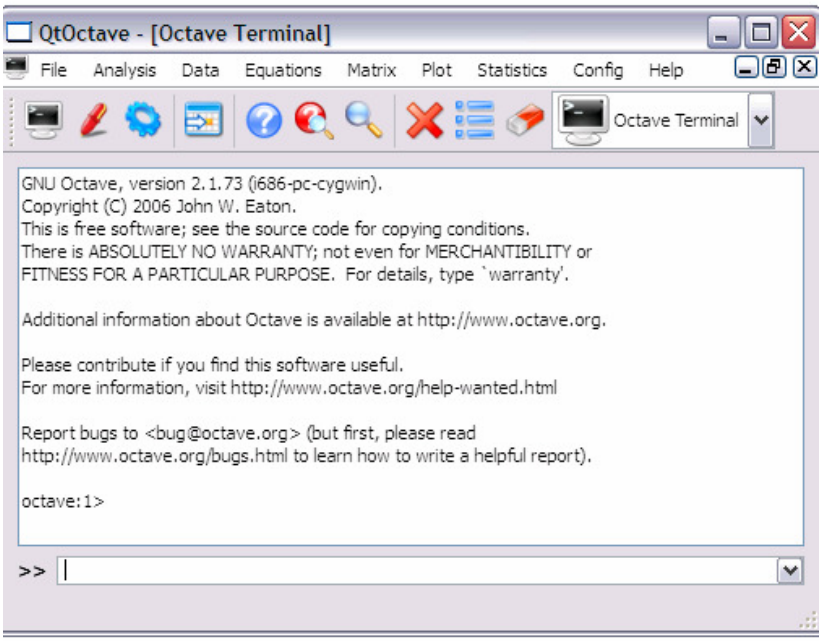


Figura 1

Significado de los iconos:

	Abre una nueva sesión de QtOctave.
	Abre un editor de texto. El editor se puede configurar en el menú Config.
	Abre una matriz en una hoja de cálculo.
	Abre la ayuda de Octave. Se puede configurar en el menú Config.
	Ayuda dinámica. Muestra ayuda de los comandos según se van tecleando en el terminal.
	Busca comandos similares al que se haya tecleado.
	Ctrl. + C. Sirve para parar procesos desbocados, bucles infinitos,...
	Borra los contenidos de el terminal.

Antes de empezar a utilizar Octave, es conveniente conocer los siguientes comandos:

- **version**: El cual se utiliza para chequear los nombres y números de versión de todas las herramientas disponibles.
- **clock** : Exhibe números decimales para indicar : (año , mes , día , hora , minuto , segundo)
- **fix(clock)**: exhibe la misma información anterior, pero en formato entero por ejemplo , la respuesta es : a n s = **2000 10 17 15 20 50**
- **who**: Produce una lista de las variables del espacio de trabajo actual.
- **whos**: Exhibe información adicional acerca de cada variable.
- **who global y whos global** : listan las variables del espacio de trabajo global
- **help**: Presenta una explicación concisa pero precisa de los comandos. Se teclea help y el comando cuyo significado no se sabe.
- **diary on:** Escribe todo lo que se introduce por el teclado, así como la mayor parte de lo que se envía a la pantalla a un archivo llamado diary.
- **diary off**: termina la escritura.

3 CÓMO INICIAR LOS CÁLCULOS

Cuando se abre una ventana de comandos aparece la indicación **>>** en la esquina superior izquierda de la ventana. Podemos escribir cualquier comando a delante de la indicación .

Para empezar podemos usar **" Octave como una calculadora "** como se indica en los siguientes ejemplos:

```
>>  23  *  45
ans  = 1035
>> 557 / 12
a n s  =  46.4167
>> -12/ (5.25-3.01) ^ 2
a n s  = - 2.3916
>> cos (2*pi/3 )
ans = - 0. 5 000
>> exp (acos (0. 3) )
ans = 3. 5470
>> (5 + 7i)  *  (6 - 3i)
ans = 51.0000 + 27.0000i
```

3.1 Operadores Aritméticos

Los operadores aritméticos son elementos que permiten construir expresiones más complejas partiendo de expresiones sencillas o atómicas (indivisibles).

Adición: x + y : Si ambos operandos son matrices el número de filas y columnas debe coincidir. Si un operando es escalar su valor es sumado a cada elemento del otro operando.

Adición elemento por elemento: x .+ y : Este operador suma dos matrices elemento a elemento. Es equivalente a x +y.

Sustracción: x – y

Sustracción elemento a elemento: x .-y

Multiplicación Matricial x * y : El número de columnas de x debe coincidir con el número de filas de y.

Multiplicación elemento a elemento x .* y : Si ambos operandos son matrices deben coincidir en dimensión.

División a derecha x / y : Este concepto es equivalente a efectuar el producto del inverso de y transpuesta por x transpuesta, transpuesto.

División a derecha elemento a elemento: x ./ y

División a Izquierda x \ y : El concepto es equivalente multiplica el inverso de x por y.

División a Izquierda elemento por elemento: x .\ y : Cada elemento de y es dividido por el correspondiente de x.

Potencia x ^ y ó x ** y : Si x e y son escalares devuelve x elevado a la potencia y.

Potencia elemento por elemento: x .^ y ó x . y :** Si los dos operandos son matrices las dimensiones deben coincidir.

Negación: -x

Suma unaria: +x : Este operador no tiene efecto sobre el operando.

Conjugado: x' : Para argumentos reales este operador es lo mismo que el operador Transpuesta. Para operandos complejos calcula el conjugado.

Transpuesta: x.'

Prioridad de operación:

1. "*", "/", "^"
2. "+", "-"

3.2 FUNCIONES MATEMÁTICAS ELEMENTALES QUE OPERAN DE MODO ESCALAR

Estas funciones, que comprenden las funciones matemáticas trigonométricas y otras funciones básicas, actúan sobre cada elemento de la matriz como si se tratase de un escalar. Se aplican de la misma forma a escalares, vectores y matrices. Algunas de las funciones de este grupo son las siguientes:

sin(x)	seno
cos(x)	coseno

tan(x) tangente

asin(x) arco seno

acos(x) arco coseno

atan(x) arco tangente (devuelve un ángulo entre -pi/2 y +pi/2)

atan2(x) arco tangente (devuelve un ángulo entre -pi y +pi);
se le pasan 2 argumentos, proporcionales al seno y al coseno

sinh(x) seno hiperbólico

cosh(x) coseno hiperbólico

tanh(x) tangente hiperbólica

asinh(x) arco seno hiperbólico

acosh(x) arco coseno hiperbólico

atanh(x) arco tangente hiperbólica

log(x) logaritmo natural

log10(x) logaritmo decimal

exp(x) función exponencial

sqrt(x) raíz cuadrada

sign(x) devuelve -1 si <0, 0 si =0 y 1 si >0.
Aplicada a un número complejo, devuelve un vector unitario en la misma dirección

rem(x,y) resto de la división (2 argumentos que no tienen que ser enteros)

mod(x,y) similar a rem (Ver diferencias con el Help)

round(x) redondeo hacia el entero más próximo

fix(x) redondea hacia el entero más próximo a 0

floor(x) valor entero más próximo hacia -infinito

ceil(x) valor entero más próximo hacia +infinito

gcd(x) máximo común divisor

lcm(x) mínimo común múltiplo

real(x) partes reales

imag(x) partes imaginarias.

abs(x) valores absolutos, también se aplica a complejos

angle(x) ángulos de fase, se aplica a complejos

3.3 Variables y nombres de variables.

No es necesario declarar los nombres de las variables ni sus tipos.

Esto se debe a que los nombres de las variables en OCTAVE no son diferentes para las variables enteras, reales y complejas. Cualquier variable puede adoptar valores reales, complejos y enteros. Ni siquiera es necesario declarar previamente el tamaño de un arreglo.

En principio, cualquier nombre puede utilizarse siempre que sea compatible con OCTAVE. Sin embargo, debemos tener presentes dos situaciones incompatibles. La primera es que OCTAVE no acepta el nombre; la segunda es que se acepta el nombre pero éste anula el significado original de un nombre reservado.

Estos conflictos pueden ocurrir con los siguientes tipos de nombres:

(a) nombres de ciertos valores

(b) nombres de funciones (subrutinas)

(c) nombres de comandos

Un método para determinar la compatibilidad del nombre de variable es probarlo en la pantalla de comandos.

Un enunciado válido como `x=9` tendrá una respuesta como ésta:

```
x =
9
```

lo que significa que se aceptó la variable. En cambio, si se prueba con `end=4` (como ejemplo de uso indebido), será ignorado; un ejemplo del segundo conflicto es el siguiente: si se utilizan `sin` y `cos` (como ejemplos de nombres de variable indebidos) sin relación con las funciones trigonométricas; por ejemplo:

```
sin = 3;
cos = sin^2;
```

los cálculos procederán, pero `sin` y `cos` ya no podrán utilizarse como funciones trigonométricas en tanto no sean borradas las variables con el comando `clear` o se abandone OCTAVE. Si aparece un mensaje de error relacionado con un conflicto, es importante que el lector investigue qué lo causó.

Es tradicional utilizar los símbolos $i, j, k, 1, m$ y n como variables enteras o índice.

Se puede verificar la existencia de una variable o un archivo con el comando `exist`.

Otra manera es asignar valores a las variables, al colocar punto y coma al final se suprime la salida en pantalla pero la información queda guardada en memoria, ejemplo:

```
>>  a = 3 ;

>> b = 12  ;  b^(1/a)
a n s  =  2.2894

r = 5;
h=25;
vol  = (1 / 3 ) *pi * r^ 2*h
      a n s  =   654.4985
```

PROBLEMA DE SIMPLE MATEMATICA

Cálculos con una sola variable: Cuando se abre una ventana de comandos, aparece la indicación octave:1> en la esquina superior izquierda de la ventana. Podemos escribir cualquier comando adelante de la indicación. En nuestras explicaciones de los comandos, omitiremos la
indicación por sencillez.

Como ejemplo sencillo, evaluemos:

$$\text{Volumen} = \frac{4}{3}\pi r^3, \text{ con } r = 2$$

Los comandos que debemos teclear son:

```
> r = 2;
> vol = (4/3)*pi*r^3;
```

donde `pi = π` en Octave. Cada línea se teclea adelante de la indicación `Octave:1>` y se oprime la tecla Enter al final de la línea. Observe que en el guión anterior cada línea es un comando y termina con un signo de punto y coma. El circunflejo `^` después de `r` es el operador de exponente.

Cuando trabajamos en la ventana de comandos, la computadora calcula la respuesta de cada comando inmediatamente después de pulsarse la tecla Enter. Por tanto, el valor de `vol` ya está en la computadora; ¿cómo podemos hacer que aparezca en la pantalla?

La forma más fácil de exhibir el resultado es teclear `vol` y pulsar Enter. La computadora exhibirá

```
vol =
33.510
```

Otra forma de imprimir el valor de `vol` es omitir el signo de punto y coma al final del segundo comando:

```
> r = 2;
> vol = (4/3)*pi*r^3
```

Si falta el punto y coma, el resultado se imprimirá inmediatamente después de calcularse. Sin embargo, como casi nunca resulta cómodo ir imprimiendo todos los resultados, por lo general se coloca un punto y coma después de cada comando.

Podemos escribir varios comandos en una misma línea separándolas con signos de punto y coma. Si necesita imprimir los resultados de cada comando que se ejecute, separe los comandos con comas y termine la línea con o sin una coma. por ejemplo, si escribe

```
> r = 2, vol = (4/3)*pi*r^3
```

se imprimirán los valores de `r` y de `vol`, pero si escribe

```
> r = 2; vol = (4/3)*pi*r^3;
```

no se imprimirán resultados.

2. Juan llega a una librería y compra lo siguiente: 4 cuadernos a $1.25 c/u , 6 libretas a $5.50 c/u y 2 lapiceros a $ 10.50 c/u. ¿ cuántos artículos compró Juan y cuánto pagó ? Además, ¿cuánto fue el costo promedio.

Solución

Una forma (usando Octave como calculadora)

```
> 4 + 6 + 2
a n s =
           12

> 4 * 1. 25 + 6 * 5. 50 + 2 * 10. 50
a n s =
            59

>     59 / 12
a n s =
           4. 9167  ≡  $  4. 92
```

OTRA MANERA (ASIGNANDO VALORES A LAS VARIABLES EN OCTAVE)

```
>  cuadernos = 4;
> libretas  =  6
libretas    =
               6

>  lapiceros   =  2;
> artículos  = cuadernos  + libretas + lapiceros
artículos   =
             12

> costo = cuadernos * 1.25 + libretas * 5.50 + lapiceros * 10.50
costo =
            59
> costo_promedio  = costo/artículos
costo_promedio =
                             4.9167  ≡ $ 4.92
```

	Guía Nº 2	MATRICES
	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **DEPARTAMENTO DE CIENCIAS BÁSICAS**	
	Asignatura : Matemáticas I Lugar de Ejecución : Laboratorio de simulación y matemáticas.	

OBJETIVO.

- *Que realice las diferentes operaciones matriciales.*

1. Introducción

Las matrices aparecen por primera vez hacia el año 1850, introducidas por J.J. Silvestre.

El desarrollo inicial de la teoría se debe al matemático W.R. Hamilton en 1853. En 1858, A. Cayley introduce la notación matricial como una forma abreviada de escribir un sistema de *m* ecuaciones lineales con *n* incógnitas.

Las matrices se utilizan en el cálculo numérico, en la resolución de sistemas de ecuaciones lineales, de las ecuaciones diferenciales y de las derivadas parciales. Además de su utilidad para el estudio de sistemas de ecuaciones lineales, las matrices aparecen de forma natural en geometría, estadística, economía, informática, física, etc.

La utilización de matrices (arrays) constituye actualmente una parte esencial dn los lenguajes de programación, ya que la mayoría de los datos se introducen en los ordenadores como tablas organizadas en filas y columnas : hojas de cálculo, bases de datos.

1.1 Concepto de Matriz

Una matriz es un conjunto de elementos de cualquier naturaleza aunque, en general, suelen ser números ordenados en filas y columnas.

Se llama **matriz** *de orden "m × n" a un conjunto rectangular de elementos* a_{ij} *dispuestos en m filas y en n columnas. El orden de una matriz también se denomina dimensión o tamaño, siendo m y n números naturales.*

Las matrices se denotan con letras mayúsculas: A, B, C, ... y los elementos de las mismas con letras minúsculas y subíndices que indican el lugar ocupado: a, b, c, ... Un elemento genérico que ocupe la fila i y la columna j se escribe a_{ij} . Si el elemento genérico aparece entre paréntesis también representa a toda la matriz : A = (a_{ij})

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & a_{ij} & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

$$Ej. \quad A_{2\times 3} = \begin{pmatrix} \frac{1}{6} & -3 & 5 \\ 7 & \sqrt{2} & 4 \end{pmatrix}$$

donde sus filas son : $\begin{pmatrix} \frac{1}{6} & -3 & 5 \end{pmatrix}$ y $\begin{pmatrix} 7 & \sqrt{2} & 4 \end{pmatrix}$

y sus columnas $\begin{pmatrix} \frac{1}{6} \\ 7 \end{pmatrix}$, $\begin{pmatrix} -3 \\ \sqrt{2} \end{pmatrix}$ y $\begin{pmatrix} 5 \\ 4 \end{pmatrix}$

Cuando nos referimos indistíntamente a filas o columnas hablamos de lineas. El número total de elementos de una matriz A$m\times n$ es m·n

En matemáticas, tanto las **Listas** como las **Tablas** reciben el nombre genérico de matrices.

Una *lista numérica* es un conjunto de números dispuestos uno a continuación del otro.

1.2 Algunos Tipos de Matrices

Hay algunas matrices que aparecen frecuentemente y que según su forma, sus elementos reciben nombres diferentes :

Tipo de matriz	*Definición*	*Ejemplo*
FILA	*Aquella matriz que tiene una sola fila, siendo su orden* $1\times n$	$A_{1\times 3} = \begin{pmatrix} 7 & 2 & -5 \end{pmatrix}$
COLUMNA	*Aquella matriz que tiene una sola columna, siendo su orden* $m\times 1$	$A_{3\times 1} = \begin{pmatrix} -7 \\ 1 \\ 6 \end{pmatrix}$
RECTANGULAR	*Aquella matriz que tiene distinto número de filas que de columnas, siendo su orden* $m\times n$, $m \neq n$	$A_{3\times 4} = \begin{pmatrix} 1 & 3 & 2 & 9 \\ 5 & 7 & -1 & 8 \\ 0 & 3 & 5 & 1 \end{pmatrix}$
TRASPUESTA	*Dada una matriz* A, *se llama traspuesta de* A *a la matriz que se obtiene cambiando ordenadamente las filas por las columnas. Se representa por* A^t ó A^T	*Si es* $A = (a_{ij})_{m\times n}$ *su traspuesta es* $A^t = (a_{ji})_{n\times m}$ $A = \begin{pmatrix} 1 & 2 & 5 \\ 3 & -4 & 7 \end{pmatrix}$; $A^t = \begin{pmatrix} 1 & 3 \\ 2 & -4 \\ 5 & 7 \end{pmatrix}$
OPUESTA	*La matriz opuesta de una dada es la que resulta de sustituir cada elemento por su opuesto. La opuesta de* A *es* -A.	$A = \begin{pmatrix} 1 & 3 \\ 5 & -7 \\ -6 & 4 \end{pmatrix}$, $-A = \begin{pmatrix} -1 & -3 \\ -5 & 7 \\ 6 & -4 \end{pmatrix}$
NULA	*Si todos sus elementos son cero. También se denomina matriz cero y se denota por* $0_{m\times n}$	$0_{3\times 4} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$
CUADRADA	*Aquella matriz que tiene igual número de filas que de columnas, m = n, diciendose que la matriz es de <u>orden n</u>.* <u>Diagonal principal</u> : *son los elementos* a_{11} , a_{22} , ..., a_{nn} <u>Diagonal secundaria</u> : *son los elementos* a_{ij} *con* $i+j = n+1$ <u>Traza</u> de una matriz cuadrada : *es la suma de los elementos de la diagonal principal* tr A.	$A_3 = \begin{pmatrix} 1 & 9 & -6 \\ 0 & 2 & 1 \\ -2 & 4 & 5 \end{pmatrix}$ $A = \begin{pmatrix} 5 & 1 & 5 & -6 \\ 1 & 2 & 0 & 3 \\ 7 & -3 & 4 & 11 \\ 1 & 9 & 3 & 8 \end{pmatrix}$ Diagonal principal : Diagonal secundaria :

		$A = \begin{pmatrix} 5 & 1 & 5 & -6 \\ 1 & 2 & 0 & 3 \\ 7 & -3 & 4 & 11 \\ 1 & 9 & 3 & 8 \end{pmatrix}$
SIMÉTRICA	*Es una matriz cuadrada que es igual a su traspuesta.* $A = A^t$, $a_{ij} = a_{ji}$	$A_3 = \begin{pmatrix} 1 & 9 & -6 \\ 9 & 2 & 1 \\ -6 & 1 & 5 \end{pmatrix}$
ANTISIMÉTRICA	*Es una matriz cuadrada que es igual a la opuesta de su traspuesta.* $A = -A^t$, $a_{ij} = -a_{ji}$ *Necesariamente* $a_{ii} = 0$	$A_3 = \begin{pmatrix} 0 & 9 & -6 \\ 9 & 0 & 1 \\ -6 & 1 & 0 \end{pmatrix}$
DIAGONAL	*Es una matriz cuadrada que tiene todos sus elementos nulos excepto los de la diagonal principal*	$A = \begin{pmatrix} 7 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & -2 \end{pmatrix}$
ESCALAR	*Es una matriz cuadrada que tiene todos sus elementos nulos excepto los de la diagonal principal que son iguales*	$A = \begin{pmatrix} 7 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 7 \end{pmatrix}$
IDENTIDAD	*Es una matriz cuadrada que tiene todos sus elementos nulos excepto los de la diagonal principal que son iguales a* 1. *Tambien se denomina matriz unidad.*	$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
TRIANGULAR	*Es una matriz cuadrada que tiene todos los elementos por encima (por debajo) de la diagonal principal nulos.*	$A = \begin{pmatrix} 1 & 3 & 5 \\ 0 & 4 & -1 \\ 0 & 0 & 9 \end{pmatrix}$ T. superior $A = \begin{pmatrix} 1 & 0 & 0 \\ 5 & 4 & 0 \\ 2 & 8 & 7 \end{pmatrix}$ T. inferior

ORTOGONAL	*Una matriz ortogonal es necesariamente cuadrada e invertible :* $A^{-1} = A^T$ La inversa de una matriz ortogonal es una matriz ortogonal. El producto de dos matrices ortogonales es una matriz ortogonal. El determinante de una matriz ortogonal vale +1 ó -1.	$A \cdot A^T = A^T \cdot A = I$ $\begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix} \cdot \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
NORMAL	*Una matriz es normal si conmuta con su traspuesta. Las matrices simétricas, antisimétricas u ortogonales son necesariamente normales.*	$A = \begin{pmatrix} 5 & 4 \\ -4 & 5 \end{pmatrix}$ $A \cdot A^T = A^T \cdot A$
INVERSA	*Decimos que una matriz cuadrada* A *tiene inversa,* A-1, *si se verifica que :* $A \cdot A^{-1} = A^{-1} \cdot A = I$	$A = \begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix}$; $A^{-1} = \begin{pmatrix} -1 & 3 \\ 1 & -2 \end{pmatrix}$

Para establecer las reglas que rigen el cálculo con matrices se desarrolla un álgebra semejante al álgebra ordinaria, pero en lugar de operar con números lo hacemos con matrices.

1.3 Operaciones con Matrices

OCTAVE puede operar con matrices por medio de operadores y por medio de funciones. Se han visto ya los operadores suma (+), producto (*) y traspuesta ('), así como la función invertir inv( ).

Los operadores matriciales de OCTAVE son los siguientes:

- **+** adición o suma
- **-** sustracción o resta
- ***** multiplicación
- **'** traspuesta
- **^** potenciación

**** división-izquierda

/ división-derecha

.* producto elemento a elemento

./ y **.** división elemento a elemento

.^ elevar a una potencia elemento a elemento

Estos operadores se aplican también a las variables o valores escalares.

1.3.1 Suma de matrices

La suma de dos matrices $A = (a_{ij})_{m\times n}$ y $B = (b_{ij})_{p\times q}$ *de la misma dimensión (equidimensionales) : m = p y n = q es otra matriz* $C = A+B = (c_{ij})_{m\times n} = (a_{ij}+b_{ij})$

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \quad ; \quad B=\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$

$$A+B=\begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} \end{pmatrix}$$

p.ej.

$$A=\begin{pmatrix} 2 & -3 & 5 \\ 4 & 1 & -7 \end{pmatrix} ; \; B=\begin{pmatrix} 1 & 0 & 2 \\ -3 & 5 & 8 \end{pmatrix}$$

$$A+B=\begin{pmatrix} 3 & -3 & 7 \\ 1 & 6 & 1 \end{pmatrix}$$

Es una ley de composición interna con las siguientes **PROPIEDADES:**

· **Asociativa :** A+(B+C) = (A+B)+C

· **Conmutativa :** A+B = B+A

· **Elem. neutro :** (matriz cero $0_{m\times n}$) , 0+A = A+0 = A

· **Elem. simétrico :** (matriz opuesta -A) , A + (-A) = (-A) + A = 0

Al conjunto de las matrices de dimensión m×n cuyos elementos son números reales lo vamos a representar por $M_{m\times n}$ y como hemos visto, por cumplir las propiedades anteriores, (M, +) es un *grupo abeliano*.

¡¡ La suma y diferencia de dos matrices NO está definida si sus dimensiones son distintas. !!

1.3.2 Producto de un número real por una matriz

Para multiplicar un escalar por una matriz se multiplica el escalar por todos los elementos de la matriz, obteniéndose otra matriz del mismo orden.

$$A=(a_{ij})=\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \quad ; \quad \lambda\cdot A=\begin{pmatrix} \lambda a_{11} & \lambda a_{12} & \dots & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \dots & \lambda a_{2n} \\ \dots & \dots & \dots & \dots \\ \lambda a_{m1} & \lambda a_{m2} & \dots & \lambda a_{mn} \end{pmatrix}$$

p.ejm.

$$A=\begin{pmatrix} 1 & -2 & 3 \\ 0 & 1 & 8 \end{pmatrix} \quad ; \quad (-5)\cdot A=\begin{pmatrix} -5 & 10 & -15 \\ 0 & -5 & -40 \end{pmatrix}$$

Es una ley de composición externa con las siguientes **PROPIEDADES:**

1) Asociativa: $\lambda(\mu A)=(\lambda\mu)A$
2) Distributiva I: $\lambda(A+B)=\lambda A+\mu B$
3) Distributiva II: $(\lambda+\mu)\cdot A=\lambda A+\mu A$
4) E. Neutro de escalares: $1\cdot A=A$

$\forall \lambda,\mu,1\in R \; ; \; \forall A\in M_{m\times n}$

por tanto la terna $[M_{m\times n},+,\bullet R]$

constituye un ESPACIO VECTORIAL

1.3 3 Producto de matrices

Dadas dos matrices A = (a_{ij})mxn y B = (b_{ij})p*x*q donde n = p, es decir, el número de columnas de la primera matriz *A* es igual al número de filas de la matriz *B*, se define el producto *A·B* de la siguiente forma :

El elemento que ocupa el lugar (i, j) en la matriz producto se obtiene sumando los productos de cada elemento de la fila i de la matriz A por el correspondiente de la columna j de la matriz B.

1.3.4 Matriz Inversa

Se llama matriz inversa de una matriz cuadrada An y la representamos por A^{-1} , a la matriz que verifica la siguiente propiedad : $A^{-1}\cdot A = A\cdot A^{-1} = I$

Decimos que una matriz cuadrada es *"regular"* si su determinante es distinto de cero, y es *"singular"* si su determinante es igual a cero.

$$|A|\neq 0 \Rightarrow \textit{Matriz Regular}$$
$$|A| = 0 \Rightarrow \textit{Matriz Singular}$$

PROPIEDADES :

1) $A_n^{-1} \cdot A_n = A_n \cdot A_n^{-1} = I_n$

2) $\exists A_n^{-1} \Leftrightarrow |A_n| \neq 0$

3) $(A_n \cdot B_n)^{-1} = B_n^{-1} \cdot A_n^{-1}$

4) $\left(A_n^{-1}\right)^{-1} = A_n$

5) $(kA_n)^{-1} = \frac{1}{k} A_n^{-1}$

6) $\left(A^T\right)^{-1} = \left(A_n^{-1}\right)^T$

7) $A_n^{-1} = \frac{1}{|A_n|} \cdot \left[Adj(A_n)\right]^T$

- Sólo existe matriz inversa de una matriz cuadrada si ésta es *regular.*
- La matriz inversa de una matriz cuadrada, si existe, es única.
- Entre matrices NO existe la operación de división, la matriz inversa realiza funciones análogas.

Métodos para hallar la matriz inversa :

- Aplicando la definición.
- Por el método de Gauss.
- Por determinantes.

PROCEDIMIENTO.

Utilizando la ventana de comandos (Terminal).

En Octave los elementos de una matriz pueden ser números o caracteres. La introducción de estos elementos en Octave, lo ilustraremos con el siguiente ejemplo:

Sea $A = \begin{bmatrix} \mathbf{9} & \mathbf{7} \\ \mathbf{4} & \mathbf{5} \end{bmatrix}$

La cual se introduce así: A= [**9 7; 4 5**], en donde se usa corchete mientras que el espacio separa las columnas y el punto y coma las filas. También puede usarse para separar las columnas: ambos coma y espacio.

OTRO EJEMPLO

Separador de filas (;) b=$[5,\text{sqrt}(-5),(7+9+4)*7/3;12,-3,7/5]$

Separador de columnas: espacio ó coma (,)

VARIABLES MATRICIALES

Antes de tratar las **"OPERACIONES ENTRE MATRICES",** es conveniente que conozcamos algunas variables matriciales que usa Octave, así :

A(m , n) Define el elemento (m,n) de la matriz (fila m y columna n)

A(a,:) Define la fila a - ésima de a la matriz A

A(:,b) Define la columna b - ésima de la matriz A

A(:) Define un vector columna cuyos elementos son las columnas de A situadas por orden una debajo de la otra

A (:,:) Equivale a toda la matriz A

size(A) Devuelve el tamaño de la matriz A

length(A) Devuelve el mayor número de filas o columnas

zeros(m, n) Crea la matriz nula de orden m x n

ones(m, n) Crea la matriz de orden m x n con todos sus elementos unos

eye(m, n) Crea la matriz de orden m x n con unos en la diagonal principal y ceros en el resto

eye(n) Crea la matriz identidad de orden n

A' Devuelve la matriz traspuesta de A

det(A) Devuelve el determinante de A

inv(A) Devuelve la matriz inversa de A

tril(A) Devuelve la parte triangular inferior de la matriz A

triu(A) Devuelve la parte triangular superior de la matriz A

diag(A) Extrae la diagonal de la matriz A como vector columna

EJEMPLOS

1) Si $A = \begin{bmatrix} 2 & 4 & 5 \\ 8 & 9 & 3 \\ 2 & 1 & 7 \end{bmatrix}$, entonces tenemos A = [2 4 5;8 9 3;2 1 7]

Luego: i) A (2 , 3)

```
ans =
     3
```

ii) A (1 , :)

```
ans =
     2     4     5
```

iii) A (: , 3)

```
ans =
     5
     3
     7
```

iv) A (:)

```
ans =
     2
     8
     2
     4
     9
     1
     5
     3
     7
```

v) A (: , :)

```
ans =
     2     4     5
     8     9     3
     2     1     7
```

vi) size (A)

```
ans =
     3     3
```

vii) length (A)

```
ans =
     3
```

viii) zeros (2,3)

```
ans =
     0     0     0
     0     0     0
```

ix) ones (3,2)

```
ans =
     1     1
     1     1
     1     1
```

x) eye (4,3)

```
ans =
     1     0     0
     0     1     0
     0     0     1
     0     0     0
```

xi) eye (3)

```
ans =
     1     0     0
     0     1     0
     0     0     1
```

xii) A'

```
ans =
     2     8     2
     4     9     1
     5     3     7
```

xiii) det (A)

```
ans =   - 130
```

xiv) inv (A)

```
ans =
   - 0. 4615    0. 1769    0. 2538
    0. 3846   -0. 0308   -0. 2615
    0. 0769   -0.0462    0.1077
```

xv) tril (A)

```
ans =
   2   0   0
   8   9   0
   2   1   7
```

xvi) triu (A)

```
ans =
   2   4   5
   0   9   3
   0   0   7
```

xvii) diag (A)

```
ans =
   2
   9
   7
```

OPERACIONES CON MATRICES

Octave admite la mayoría de la operaciones del álgebra matricial, tales como: Suma, Diferencia y Multiplicación. Obviamente, se efectúan estas operaciones, siempre y cuando se guardan las normas de dimensionalidad correspondientes.

SUMA Y DIFERENCIA DE MATRICES

Podemos sumar una matriz con otra o restarla de otra si ambas tienen la misma dimensión o tamaño (mismo número de filas y de columnas).

Suponga que A y B son matrices. Entonces,

Suma : C = A + B

Diferencia: D = A - B

EJEMPLO

$$\mathbf{Si} \quad \mathbf{A} = \begin{bmatrix} 8 & 6 & -3 \\ -2 & 0 & 5 \\ 1/4 & 7 & 1 \end{bmatrix} \qquad \mathbf{y} \quad \mathbf{B} = \begin{bmatrix} 9 & 1/3 & -7 \\ 3 & 2 & 0 \\ 1/6 & 2 & 4 \end{bmatrix}$$

Entonces, encuentre:

i) C = A + B y ii) D = B – A

SOLUCION (en Octave)

```
A =[8  6  -3;-2  0  5;1/4  7  1];
B =[9  1/3  -7;3  2  0;1/6  2  4];
C = A + B                                   D =  B – A
```

C=

17	19/3	-10
1	2	5
5/12	9	5

D =

1	-17/3	-4
5	2	-5
-1/12	-5	3

MULTIPLICACIÓN DE MATRICES

Sean A y B dos matrices, si el número de columnas de A y el número de filas de B son iguales, entonces las matrices pueden multiplicarse según el orden: A*B

Sean $\mathbf{A} = \begin{bmatrix} \mathbf{a}_{1,1} & \mathbf{a}_{1,2} \\ \mathbf{a}_{2,1} & \mathbf{a}_{2,2} \end{bmatrix}$ y $\mathbf{B} = \begin{bmatrix} \mathbf{b}_{1,1} & \mathbf{b}_{1,2} \\ \mathbf{b}_{2,1} & \mathbf{b}_{2,2} \end{bmatrix}$

Entonces, C = A*B =

$$\begin{bmatrix} \mathbf{a}_{1,1} * \mathbf{b}_{1,1} + \mathbf{a}_{1,2} * \mathbf{b}_{2,1} & \mathbf{a}_{1,1} * \mathbf{b}_{1,2} + \mathbf{a}_{1,2}\ \mathbf{b}_{2,2} \\ \mathbf{a}_{2,1} * \mathbf{b}_{1,1} + \mathbf{a}_{2,2} * \mathbf{b}_{2,1} & \mathbf{a}_{2,1} * \mathbf{b}_{1,2} + \mathbf{a}_{2,2} * \mathbf{b}_{2,2} \end{bmatrix}$$

EJEMPLO

Si $\mathbf{A} = \begin{bmatrix} \mathbf{9} & \mathbf{-3} \\ \mathbf{-2} & \mathbf{1/7} \\ \mathbf{1/4} & \mathbf{6} \end{bmatrix}$ y $\mathbf{B} = \begin{bmatrix} \mathbf{1/5} & \mathbf{6} & \mathbf{-8} \\ \mathbf{2} & \mathbf{-5} & \mathbf{-2/3} \end{bmatrix}$, entonces calcule C = AB

SOLUCION (en Octave)

```
A= [9   -3;-2   1/7;1/4   6];
B= [1/5   6   -8;2   -5   -2/3];
C=A*B
```

C=

-21/5	69	-70
-4/35	-89/7	334/21
241/20	-57/2	-6

En esta operación, podemos considerar otros tipos de Multiplicación como los siguientes:

1. MULTLIPLICACIÓN DE ARREGLOS

Sean A y B matrices, si A y B son matrices del mismo tamaño, entonces su producto es una matriz C, cuyos elementos son el producto de los elementos correspondientes de A y B. Es decir, C = A.* B

Nota: observe que esta Multiplicación lleva un **punto** antes del asterisco, y omitirlo es grave porque no es el producto de elemento por elemento.

EJEMPLO

Si $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$ y $\mathbf{B} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$,entonces calcule C = A.*B

SOLUCION(en Octave)

```
A = [1 2 3 4;5 6 7 8];
B = [1:4; 1:4];
C= A.*B
C=  1   4   9   16
    5  12  21  32
```

2. PRODUCTO DE UN ESCALAR POR UNA MATRIZ

Suponga que A es una matriz y c es un escalar (un número). Entonces, el producto de c por A es otra matriz B, cuyos elementos son c veces cada elemento de A. Es decir, B = cA

EJEMPLO

Si A = $\begin{bmatrix} 1/3 & 7 & -4 \\ 1 & -5 & -3/5 \end{bmatrix}$ y c = 9/4 , entonces calcule B = (9/4)A'

SOLUCION (en Octave)

```
A = [1/3   7   -4;1   -5   -3/5];
B = (9/4)*A'
B=
    3/4        9/4
   63/4      -45/4
   -9        -27/20
```

3. POTENCIA DE MATRICES

Recuerde que la **Potenciación** es un caso particular de la Multiplicación. Es decir, si A es una matriz cuadrada (igual número de filas que de columnas), entonces tenemos lo siguiente:

$A^2 = A * A$

$A^3 = A*A*A$

Luego, Octave realiza la "Potencia de una matriz A elevada al exponente **n**,
Así:

B = A ^ **n**

EJEMPLO

Si A = $\begin{bmatrix} 5 & 1/6 \\ 9 & -2/5 \end{bmatrix}$, entonces calcule A^3

SOLUCION (en Octave)

```
A = [5   1/6;9   -2/5];
B = A^3
```

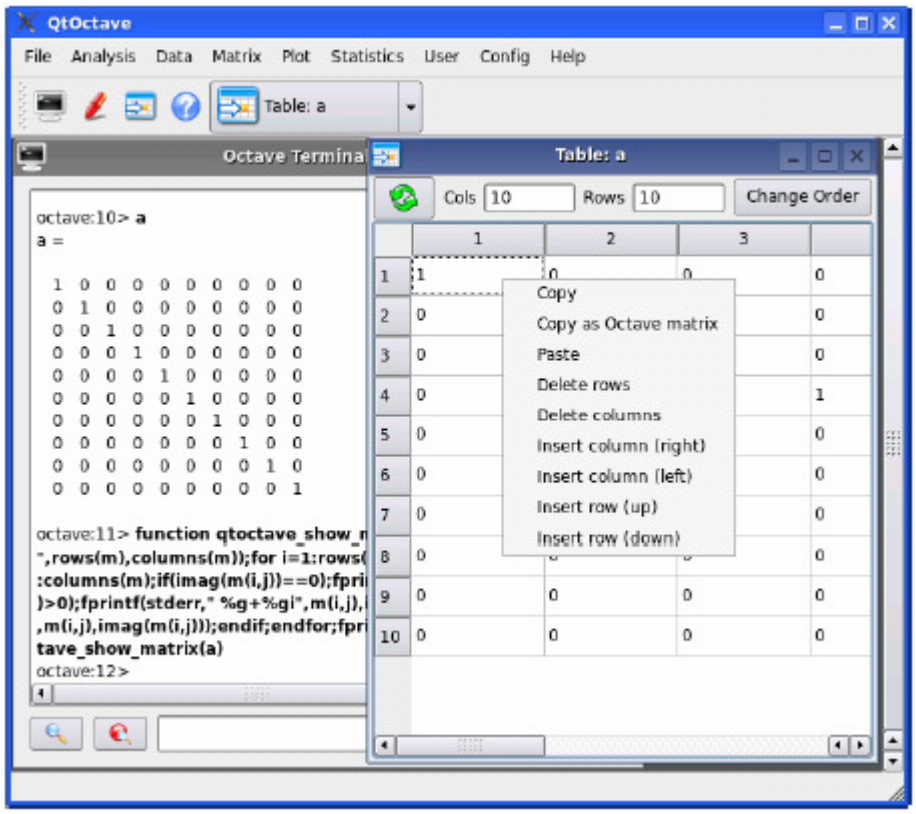
B=

697/5	411/100
11097/50	1559/250

Utilizando el menú Data.

Este menú posee herramientas para realizar manipulación de datos.

Una herramienta interesante es la utilidad "Table"  . Esta utilidad muestra matrices como si fueran hojas de cálculo. Primero preguntará el nombre de la matriz que se desea mostrar. Si todo es correcto, mostrará una ventana con la matriz. Como la de la figura siguiente:



Se pueden seleccionar celdas, filas o columnas. Si se hace click con el botón derecho del ratón aparecerá un menú con las diferentes posibilidades. Se puede copiar, pegar, borrar, etc. Una de las opciones más interesante es usar "Copy as Octave matrix". Esta opción permite copiar un trozo de matriz y después pegarla en un editor, usando la sintaxis de Octave para las matrices.

Las entradas "Cols" y "Rows", sirven para cambiar las filas y las columnas de la hoja de cálculo (el orden de la matriz).

A veces, se pueden realizar manipulaciones de matrices con Octave, que no se vean reflejadas en la hoja de cálculo. Usar el botón para actualizar la hoja de cálculo.

EJERCICIOS PARA RESOLVER EN OCTAVE

1) Si $A=\begin{bmatrix} 6 & 1 & 3 & 8 \\ 5 & 0 & 4 & 6 \\ 3 & 2 & 0 & 1 \\ 4 & 7 & 8 & 2 \end{bmatrix}$ Practique las diecisiete Variables Matriciales tratadas en la Primera Parte de este Tema de Matrices.

2) Sean $A=\begin{bmatrix} 1/3 & -2 & 7 \\ 5 & 8 & 0 \\ 2/7 & 8 & 4 \end{bmatrix}$ y $B=\begin{bmatrix} 1 & -3 & 7/8 \\ 1 & 4 & -6/4 \\ 3 & 5/8 & -2 \end{bmatrix}^T$

Determine:

i) $(A-2B)^T$; ii) $3A+(1/7)B$; iii) $(3B-2A)^2$; iv) $(3/8)A^T$; v) $3B^T$;

vi) $|(A-B)^3|$; vii) $|6A|$; viii) $|B|B^{-1}$; ix) $(3B^T-4A)^{-1}$;

x) $Adj(A^T)$; xi) $7((2/15)B)$; xii) $(Adj(2B))^{-1}$; xiii) $Cof((BA)^T)$;

xiv) $Cof(AB^T)$; xv) $A^T(Adj(A^T))$; xvi) $B(Adj(B))$; xvii) BA^T ; xviii) $(AB)^T$

3) Obtenga la matriz B, de tal manera que se cumplan las igualdades dadas:

a) $(1/3)B^T-\begin{bmatrix} 3 & 7 & -1 \\ 0 & -4 & 8 \end{bmatrix}=(-2/5)\begin{bmatrix} 5 & 4 & 2 \\ \frac{3}{5} & 0 & 1 \end{bmatrix}-\frac{8}{9}\begin{bmatrix} 3 & 6 & 5 \\ 7 & 1 & 7 \end{bmatrix}$

b) $\begin{bmatrix} 6 & 0 & 2 \\ 2 & 1 & 8 \\ 4 & 1 & 7 \end{bmatrix}^T+3B=\begin{bmatrix} 2 & 0 & 1 \\ 3 & -7 & 7 \\ 8 & 5 & 4 \end{bmatrix}^T$

	Guía Nº 3	GRÁFICAS DE FUNCIONES
	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **DEPARTAMENTO DE CIENCIAS BÁSICAS**	
	Asignatura : Matemáticas I Lugar de Ejecución : Laboratorio de simulación y matemáticas.	

OBJETIVO.

- Reconocer los diferentes tipos de funciones.

1. Introducción

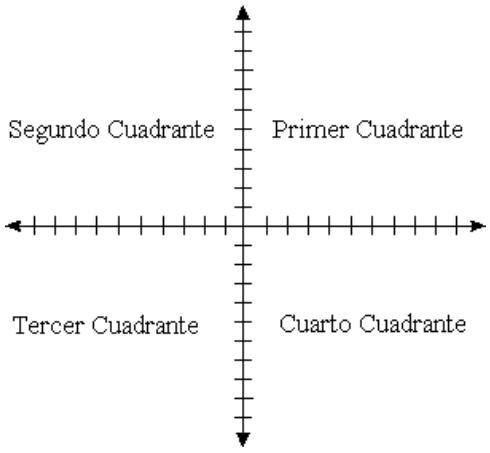
1.1 Coordenadas

Llamamos a la **coordenada de un punto** a cada punto en la recta numérica asociado con un número real. Un **par ordenado** es un par de números *a* y *b* con elementos escritos en forma significante. Dos pares ordenados son iguales si tienen el mismo primer elemento y el mismo segundo elemento.

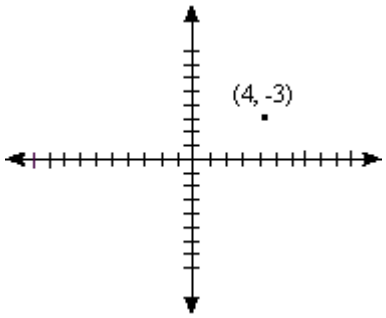
Por ejemplo: El par ordenado (4, 5) es igual al par ordenado (4, 5).

Los números en un par ordenado son llamados **coordenadas**. En el par (7, 5) la primera coordenada es 7 y la segunda es 5.

La línea horizontal es el **eje de *x***, la vertical es el **eje de *y*** y su intersección es el **origen**. Estos ejes dividen el plano en cuatro zonas llamadas **cuadrantes**. Veamos la siguiente recta numérica:



Las coordenadas en el primer cuadrante serán (+, +), las del segundo cuadrante serán (-, +), las del tercer cuadrante serán (-, -) y las del cuarto cuadrante serán (+, -). El primer número de una coordenada representa el lugar horizontal del punto y el segundo número representa el lugar vertical del punto.
Por ejemplo:



1.2 Funciones

Una **función** consiste en dos conjuntos, *dominio* y *rango*, y una regla que asigna a cada miembro del dominio exactamente un miembro del rango. A cada miembro del rango debe serle asignado por lo menos un miembro del dominio. Si la relación entre dos variables *x* y *y* es una en la que para cada valor de *y* hay exactamente un valor de *x*, se dice que *y* es una función de *x*.

Ejemplo:

y = 7x + 1
y = 7(2) + 1 = 15
y = 7(4) + 1 = 29
y = 7(6) + 1 = 43
El dominio D es {2, 4, 6} y el rango R es {15, 29, 43}.

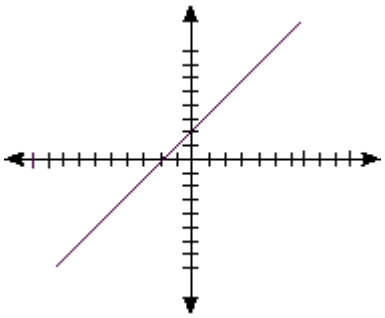
1.3 La Gráfica de una Función

La gráfica de una función es el conjunto de puntos en el plano de la forma *(x,y)* en donde *x* está en el dominio de la función y además *y=f(x).*

Para hacer la gráfica de una función como *f*(x) = x + 2, lo hacemos igual que si hiciéramos la gráfica de una ecuación y = x + 2. Buscamos los pares ordenados (x, *f*(x)), se localizan los puntos en la recta numérica y se conectan. Por ejemplo:

f(x) = x + 2

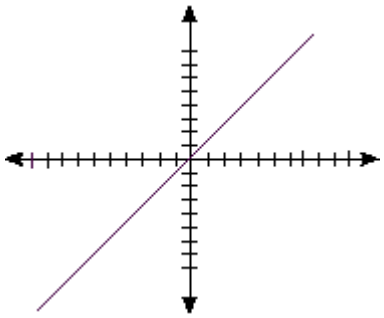
x	f(x)
-4	-2
-3	-1
-2	0
-1	1
0	2
1	3
2	4



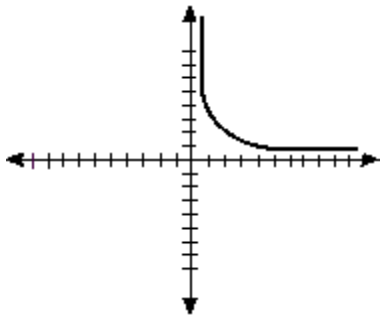
1.4 Variación

Existen dos tipos de variación: variación directa y variación inversa. Veamos cada una de ellas:

Variación Directa = es una función que se define por una ecuación que está en la forma $y = kx$, donde *k* es una constante no igual a cero. La variable *y* varía directamente de *x*. La constante *k* es llamada la **constante de variación**. La variación directa establece un único valor de *y* para cada valor de *x*. En la variación directa las dos variables aumentan (o disminuyen) juntas. Cuando el dominio es un conjunto de números reales, la gráfica de la variación directa es una línea recta con pendiente *k* que pasa por el origen.

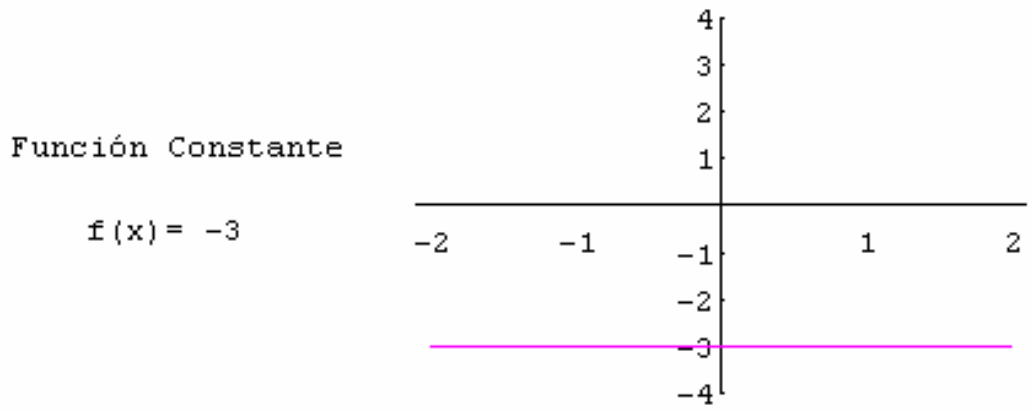


Variación Inversa = es una función que se define por una ecuación que está en la forma $y = k/x$, donde x no es igual a cero. La variable y varía a la inversa de x. En la variación invesa el aumento de una de las variables significa la disminución de la otra variable. La gráfica de esta variación es una hipérbola.

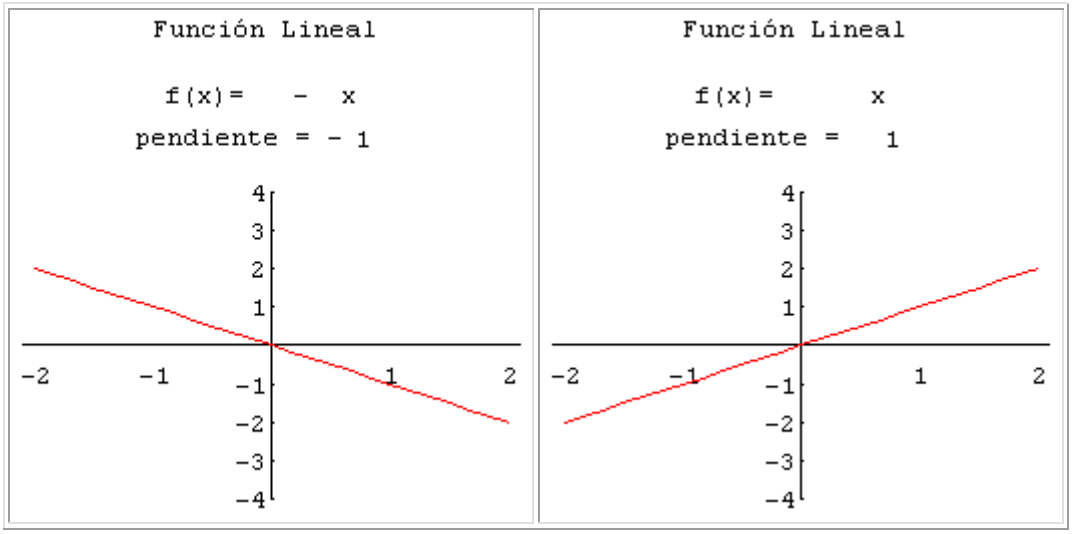


1.5 Tipos de funciones.

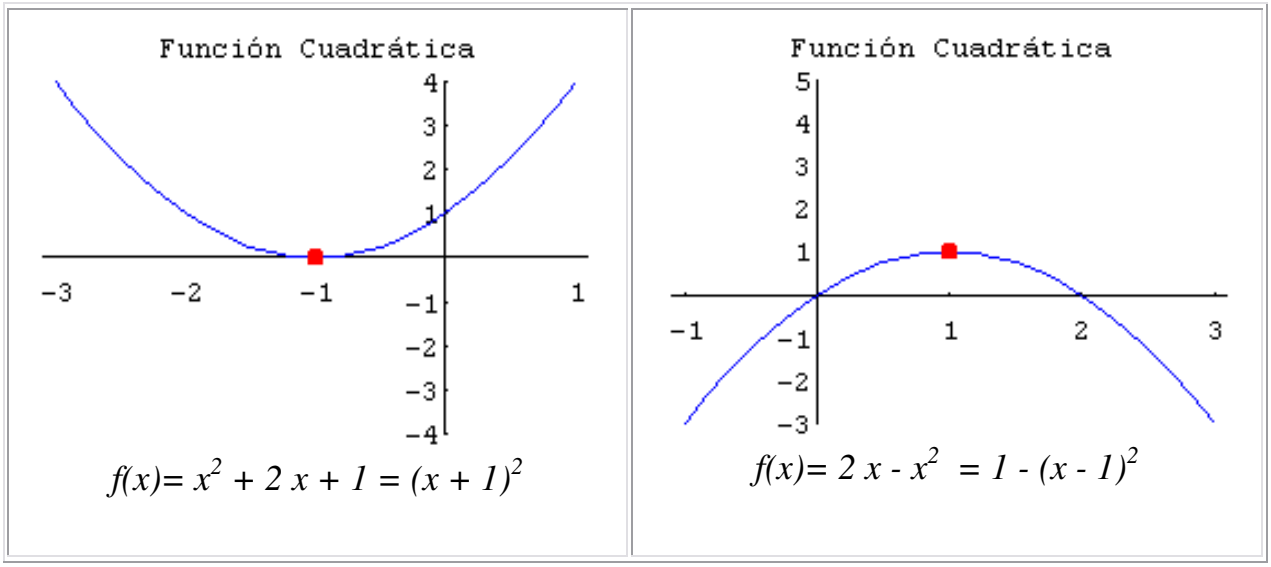
- ***Función constante: f(x)=k, donde k es alguna constante***



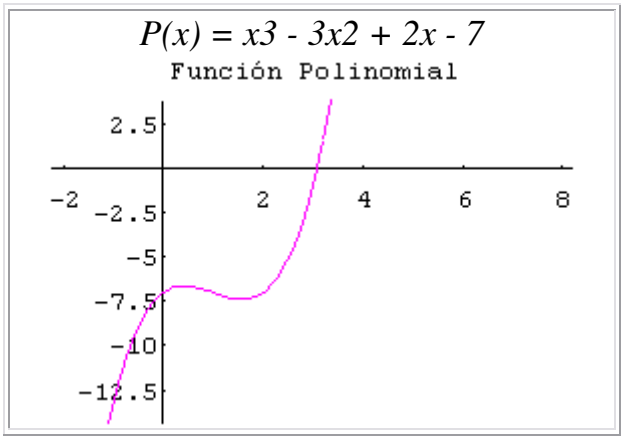
- ***Función lineal: f(x) = ax + b***



- ***Función cuadrática:***
  f(x)= ax2 + bx + c = a(x - x0)2 + y0

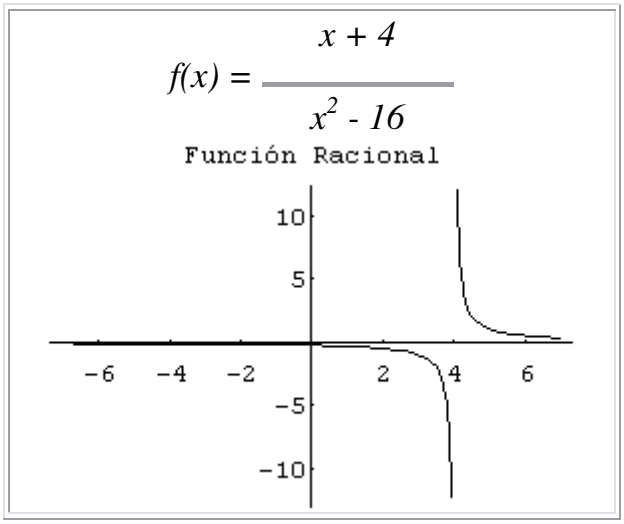


- ***Función polinomial***



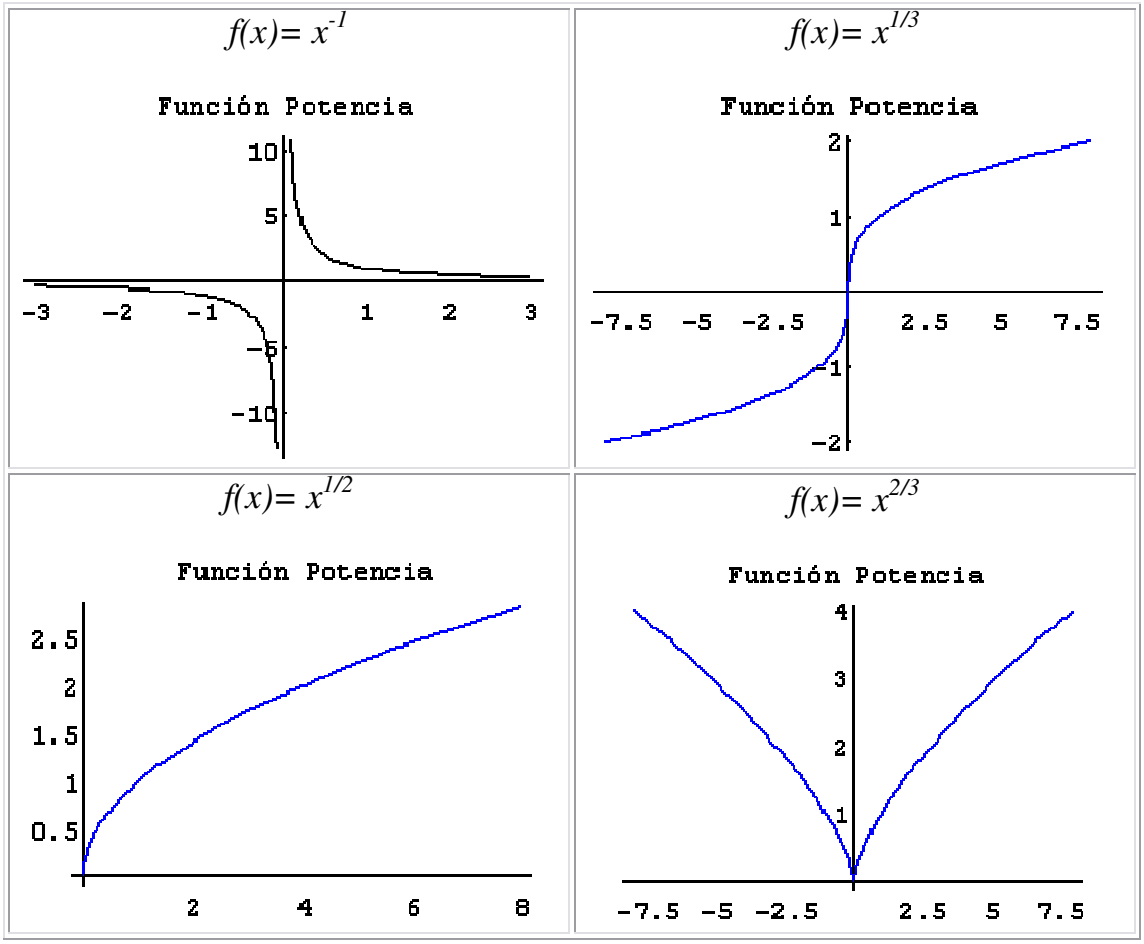
- ***Función racional***

Una función racional es un cociente de dos polinomios, *f(x) = P(x) / Q(x)*



- ***Función potencia: f(x)= k x^n***

En donde *k* es cualquier constante real y *n* es un número real. El dominio de una función potencia depende del exponente n.

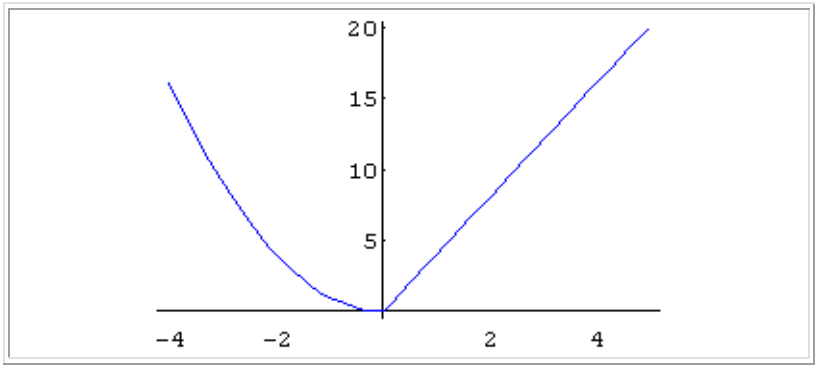


- ***Función definida por secciones***

  No es necesario que una función esté definida por una sola fórmula. La regla de correspondencia puede depender de qué parte del dominio proviene la variable independiente.

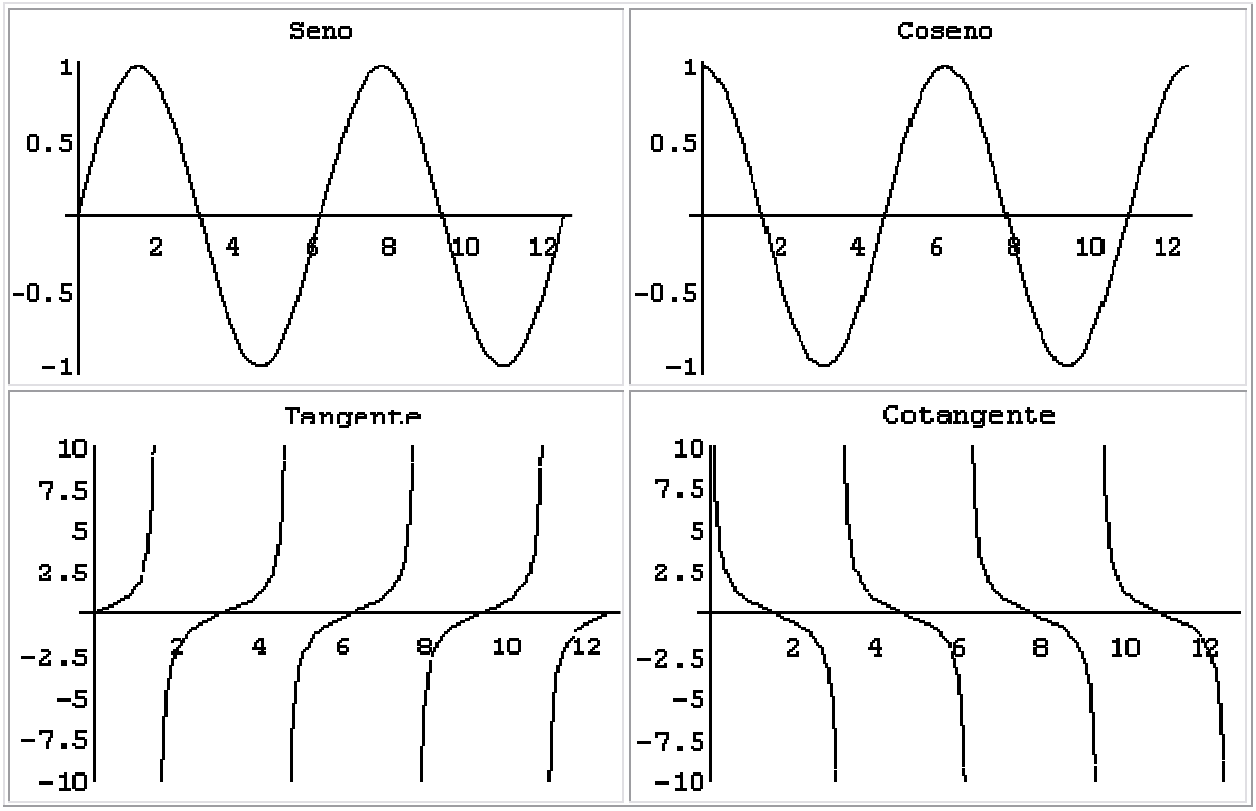
  En las siguientes dos gráficas veremos dos ejemplos de funciones definidas por secciones.

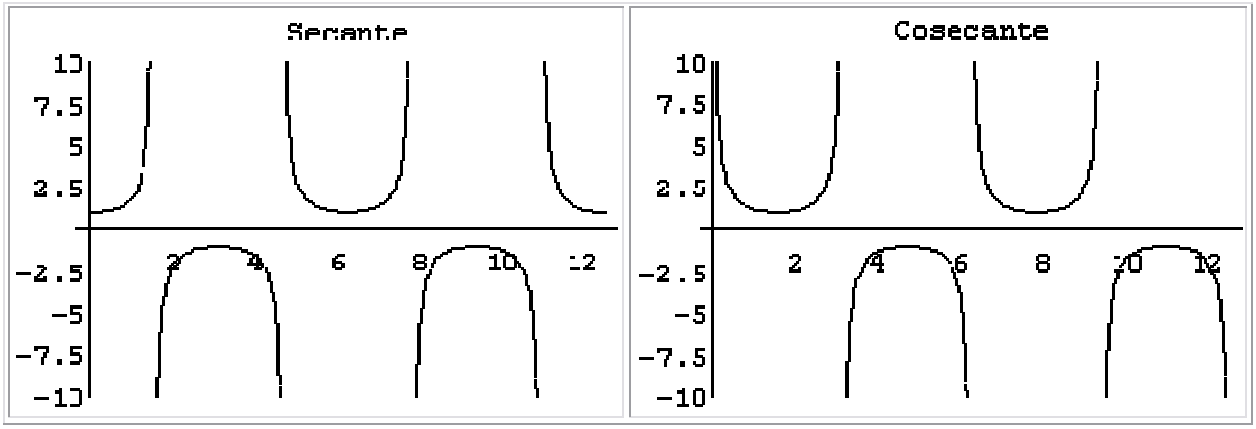
$$f(x)=\begin{cases} x^2, \\ 4\,x\,, si\;\; 0 <= x <= 5 \end{cases}$$



- ***Gráficas de las funciones trigonométricas***

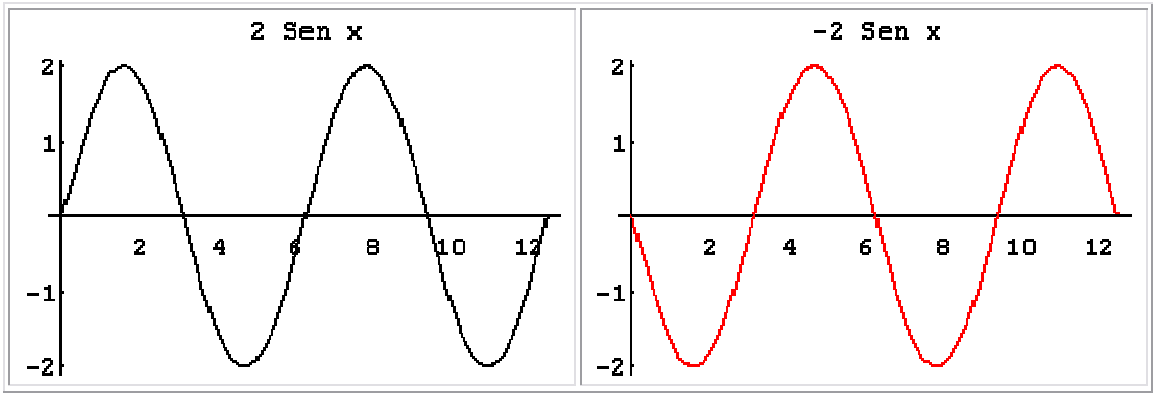
A continuación te presentamos las gráficas de las seis funciones trigonométricas.





Ejemplo:

Enseguida se muestran las gráficas de las funciones f(x)=2sen x, g(x)= -2sen x.



PROCEDIMIENTO

OCTAVE dispone de funciones básicas y especializadas para crear gráficos 2-D. Estas funciones se diferencian principalmente por el tipo de escala que utilizan en los ejes de abscisas y de ordenadas y el tipo de coordenadas. Estas funciones son las siguientes:

bar(x,y)	dados dos vectores *x* e *y* produce un gráfico de barras.
contour(z,n,x,y)	produce un gráfico de contornos de la superficie *z* definida en *3-D.*
hist(x,y,norm)	produce un histograma o la salida numérica para bar().
loglog(args)	produce un gráfico con escala logarítmica en ambos ejes.

plot(args) crea un gráfico a partir de vectores y/o columnas de matrices, con escalas lineales sobre ambos ejes.

replot permite redibujar los mismos contenidos luego de un cambio e ejes.

polar(theta,rho.fmt) produce un gráfico bidimensional en coordenadas polares.

semilogx(args) ídem con escala lineal en el eje de ordenadas y logarítmica en el eje de abscisas.

semilogy(args) ídem con escala lineal en el eje de abscisas y logarítmica en el eje de ordenadas.

stairs(x,y) dados dos argumentos produce un gráfico de tipo *escalera.*

Existen además otras funciones orientadas a añadir **títulos** al gráfico, a cada uno de los ejes, a dibujar una cuadrícula auxiliar, etc. Estas funciones son las siguientes:

title('título') añade un título al dibujo (parte superior).

bottom_title('titulo')añade un título al dibujo (parte inferior).

xlabel('tal') añade una etiqueta al eje de abscisas. Con xlabel **off** desaparece.

ylabel('cual') añade una etiqueta al eje de ordenadas. Con ylabel **off** desapace.

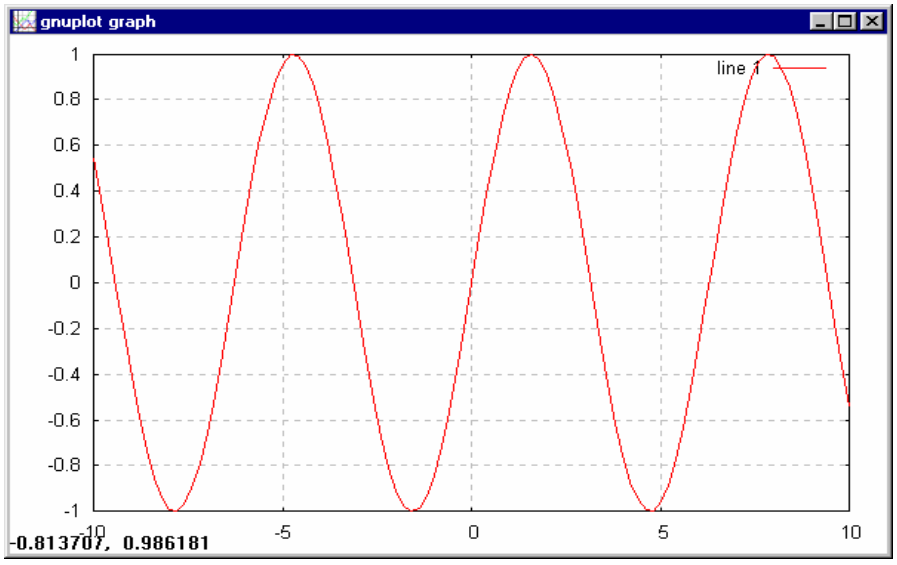
grid (arg) activa la inclusión de una cuadrícula en el dibujo. Con grid **off** desaparece la cuadrícula.

Se comenzará creando un par de vectores x e y :

```
x=[-10:0.2:10]; y=sin(x);
```

Se ejecutan los comandos siguientes (se comienza cerrando la ventana activa, para que al crear la nueva ventana aparezca en primer plano):

```
closeplot       se cierra la ventana gráfica activa anterior
grid            se crea una ventana con una cuadrícula
plot(x,y)       se dibuja la función seno conservando la cuadrícula
```



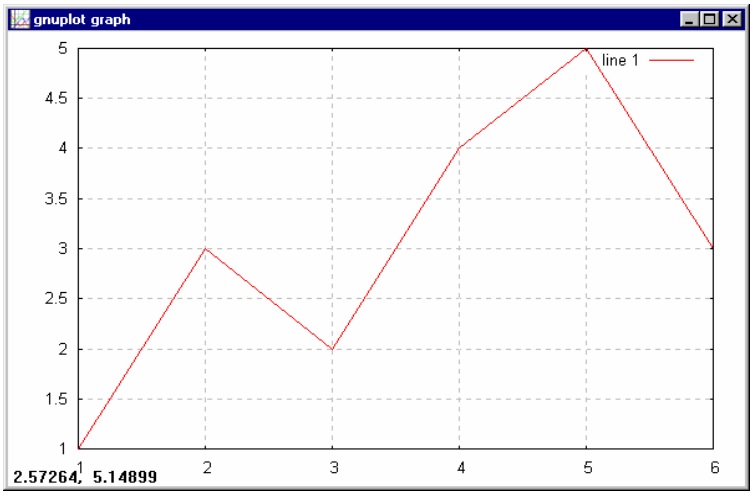
Con la función **hold** pueden añadirse gráficos a una figura ya existente respetando su contenido.

La función plot

El elemento básico de los gráficos bidimensionales es el vector. Se utilizan también cadenas de 1, 2 ó 3 caracteres para indicar colores y tipos de línea. La función **plot**(), en sus diversas variantes, no hace otra cosa que dibujar vectores.

Un ejemplo muy sencillo de esta función:

```
x=[1 3 2 4 5 3];
plot(x)
```

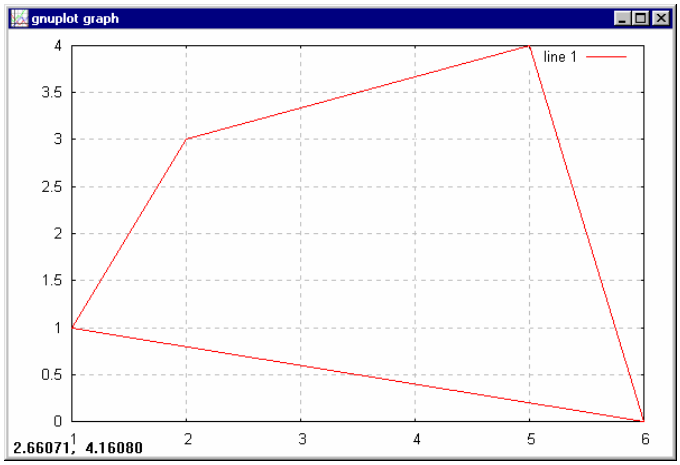


Por defecto, los distintos puntos del gráfico se unen con una línea continua. También por defecto, el color que se utiliza para la primera línea es el rojo. Cuando a la función **plot**() se le pasa un único vector real como argumento, dicha función dibuja en ordenadas el valor de los **n** elementos del vector frente a los índices 1, 2, ... n del mismo en abscisas.

OCTAVE utiliza por defecto color blanco para el fondo de la pantalla y otros colores más oscuros para los ejes y las gráficas.

Una segunda forma de utilizar la función **plot()** es con dos vectores como argumentos. En este caso los elementos del segundo vector se representan en ordenadas frente a los valores del primero, que se representan en abscisas. Por ejemplo cómo se puede dibujar un cuadrilátero de esta forma (obsérvese que para dibujar un polígono cerrado el último punto debe coincidir con el primero):

```
x=[1 6 5 2 1]; y=[1 0 4 3 1];
plot(x,y)
```

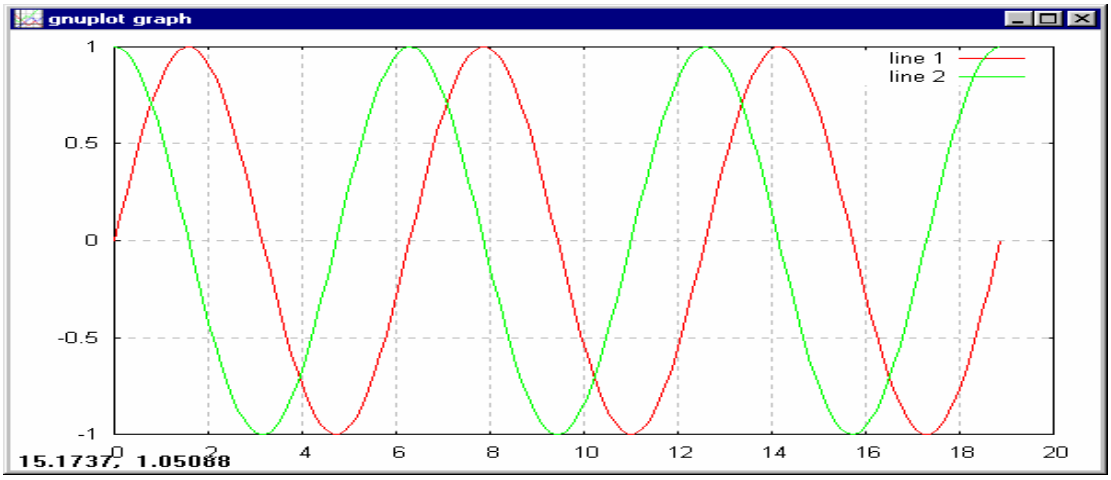


La función **plot()** permite también dibujar múltiples curvas introduciendo varias parejas de vectores como argumentos. En este caso, cada uno de los segundos vectores se dibujan en ordenadas como función de los valores del primer vector de la pareja, que se representan en abscisas. Si el usuario no decide otra cosa, para las

sucesivas líneas se utilizan colores que son permutaciones cíclicas del azul, verde, rojo, cyan, magenta, amarillo y negro.

Obsérvese bien cómo se dibujan el seno y el coseno en el ejemplo:

```
x=0:pi/25:6*pi;
y=sin(x); z=cos(x);
plot(x,y,x,z)
```



El comando **plot()** puede utilizarse también con matrices como argumentos.

Algunos ejemplos sencillos:

plot(A) dibuja una línea por cada columna de *A* en ordenadas, frente al índice de los elementos en abscisas.

plot(x,A) dibuja las columnas (o filas) de *A* en ordenadas frente al vector *x* en abscisas. Las dimensiones de *A* y *x* deben ser coherentes: si la matriz *A* es cuadrada se dibujan las columnas, pero si no lo es y la dimensión de las filas coincide con la de *x*, se dibujan las filas.

plot(A,x) análogo al anterior, pero dibujando las columnas (o filas) de *A* en abscisas, frente al valor de *x* en ordenadas.

plot(A,B) dibuja las columnas de *B* en ordenadas frente a las columnas de *A* en abscisas, dos a dos. Las dimensiones deben coincidir.

plot(A,B,C,D) análogo al anterior para cada par de matrices. Las dimensiones de cada par de matrices deben coincidir, aunque pueden ser diferentes de las dimensiones de los demás pares.

Estilos de línea y marcadores en la función plot

La tarea fundamental de la función **plot**() es dibujar los valores de un vector en ordenadas, frente a los valores de otro vector en abscisas. En el caso general esto exige que se pasen como argumentos un par de vectores. En realidad, el conjunto básico de argumentos de esta función es una tripleta formada por dos vectores y una cadena de 1, 2 ó 3 caracteres que indica el color y el tipo de línea o de marker. En la tabla siguiente se pueden observar las distintas posibilidades:

Símbolo	Color	Símbolo	Marcador
y	yellow	**∇** (5)	marcas en triángulos
m	magenta (4)	**o** (3)	marcas en círculos
c	cyan (5)	**x** (4)	marcas en x
r	red (1)	**+** (2)	marcas en + o puntos en □
g	green (2)	* (1)	marcas en *
b	blue (3)	**^**	impulsos o rayado desde eje de abscisas
w	white	**~**	barras de error
k	black	**~#**	cajas de error

Símbolo	Estilo
-	lineas continuas de rayas
.	lineas contínuas de puntos
@nm	lineas de marcas
-@nm	lineas marca-raya
--	lineas a trazos

L	linea en escalera
#	lineas de cajas

n = color , m = tipo de carácter

Cuando hay que dibujar varias líneas, por defecto se van tomando sucesivamente los colores de la tabla comenzando por el azul, hacia arriba, y cuando se terminan se vuelve a empezar otra vez por el azul. Si el fondo es blanco, este color no se utiliza para las líneas.

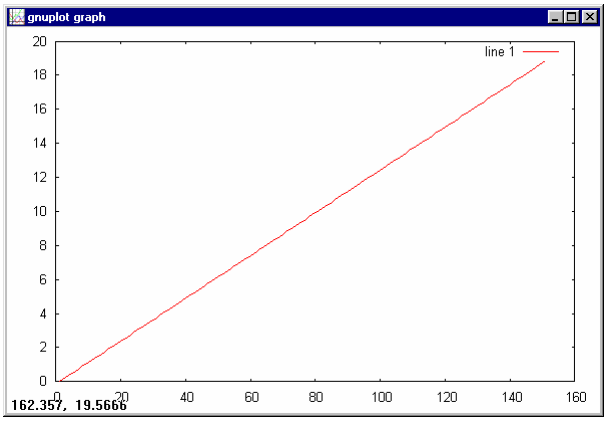
Añadir líneas a un gráfico ya existente

Existe la posibilidad de añadir líneas a un gráfico ya existente, sin destruirlo o sin abrir una nueva ventana. Se utilizan para ello los comandos **hold on** y **hold off**. El primero de ellos hace que los gráficos sucesivos respeten los que ya se han dibujado en la figura (es

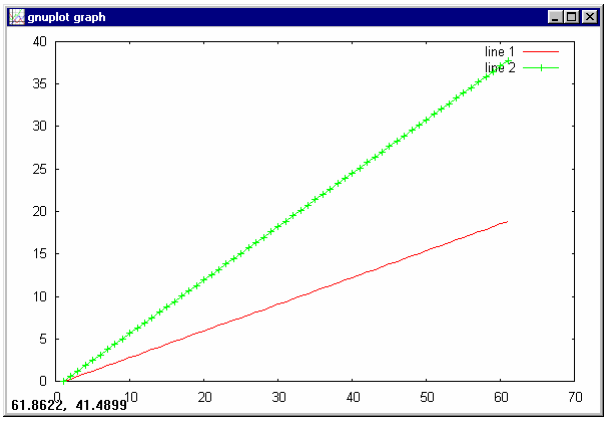
posible que haya que modificar la escala de los ejes); el comando **hold off** deshace el efecto de **hold on**.

El ejemplo muestra añade las gráficas de x2 y x3 a la gráfica de x previamente creada (cada una con un tipo de línea diferente):

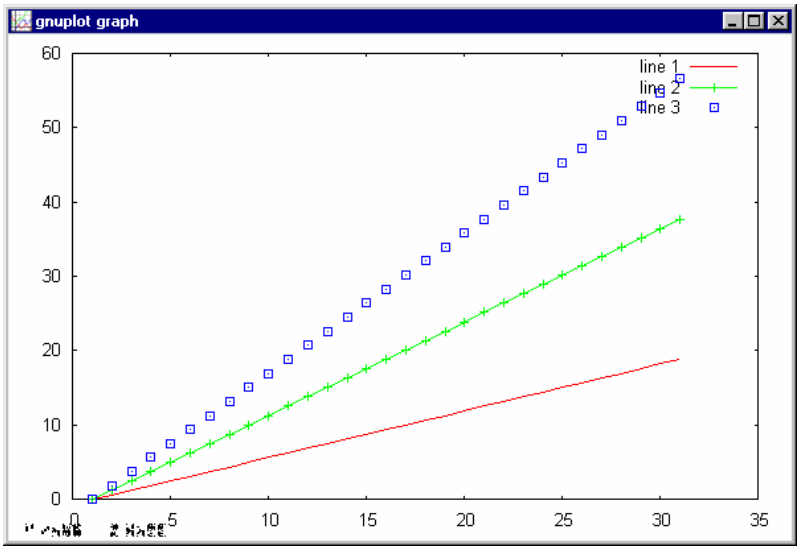
```
clearplot
x=0:pi/5:6*pi;
plot(x)
```



```
hold on
x2=2*x;
plot(x2,'-@2')   puntos "+" y  raya "-" separados  verde
```




```
x3=3*x;
plot(x3,'@32')    puntos separados azul "□"
hold off
```



Control de los ejes

OCTAVE tiene por defecto, que en algunas ocasiones puede interesar cambiar. El comando básico es **axis**(limites). Por defecto, ajusta la *escala* de cada uno de los

ejes de modo que varíe entre el mínimo y el máximo valor de los vectores a representar. Este es el llamado modo *auto*, o modo automático. Para definir de modo explícito los valores *máximo* y *mínimo* según cada eje, se utiliza el comando:

axis([xmin, xmax, ymin, ymax, zmin,zmax]) son opcionales de 2 en 2

axis('auto') devuelve el escalado de los ejes al valor por defecto o automático.

axis devuelve un vector v con [xmin, xmax, ymin, ymax, zmin, zmax]

axis(axis) mantiene los ejes en sus actuales valores, de cara a posibles nuevas gráficas añadidas con hold on

axis('**ij**') utiliza ejes de pantalla, con el origen en la esquina superior izda y el eje j en dirección vertical descendente.

axis('**xy**') utiliza ejes cartesianos normales, con el origen en la esquina inferior izda. y el eje y vertical ascendente.

axis('**equal**') el escalado es igual en ambos ejes

axis('**square**') la ventana será cuadrada

axis('**image**') la ventana tendrá las proporciones de la imagen que se desea representar en ella (por ejemplo la de una imagen bitmap que se desee importar) y el escalado de los ejes será coherente con dicha imagen.

axis(**'normal'**) elimina las restricciones introducidas por 'equal' y 'square'

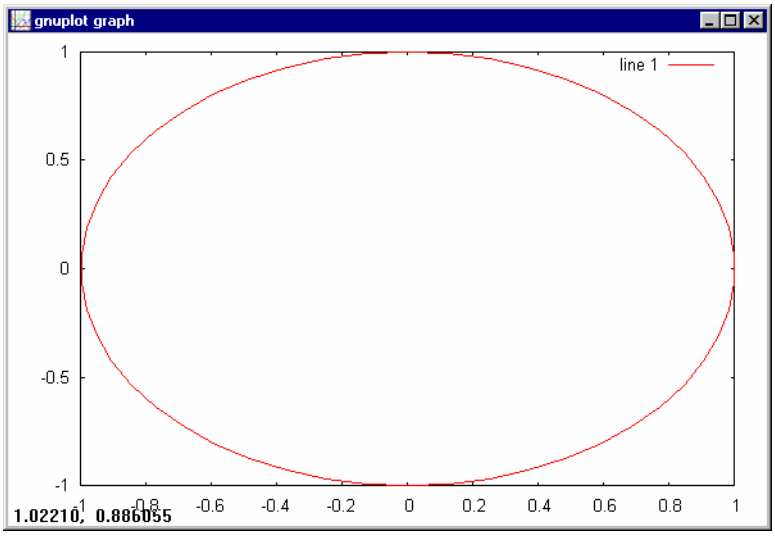
axis('tic[xyz]') introduce tics en los ejes indicados. Por ej 'ticx'

axis('label[xyz]') introduce un rótulo en los ejes indicados.

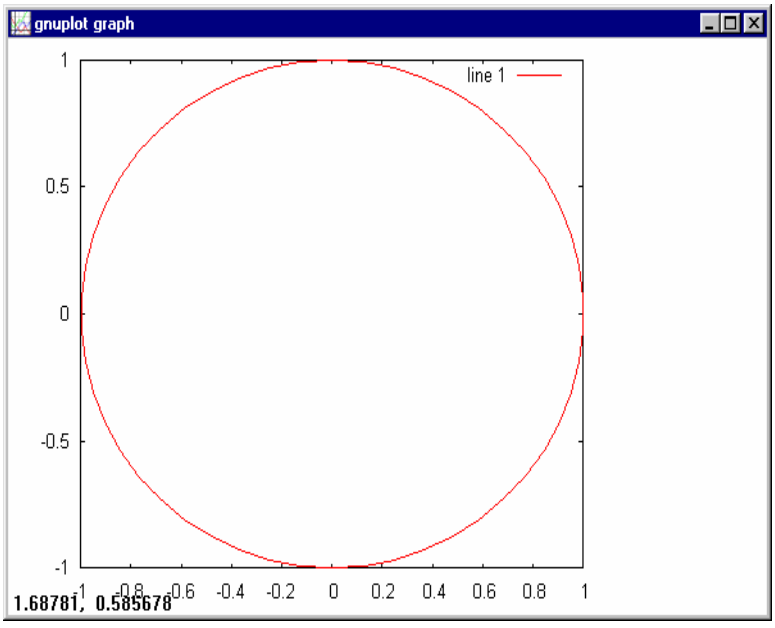
axis(**'off'**) elimina las etiquetas, los números y los ejes

axis('**on**') restituye las etiquetas, los números y los ejes

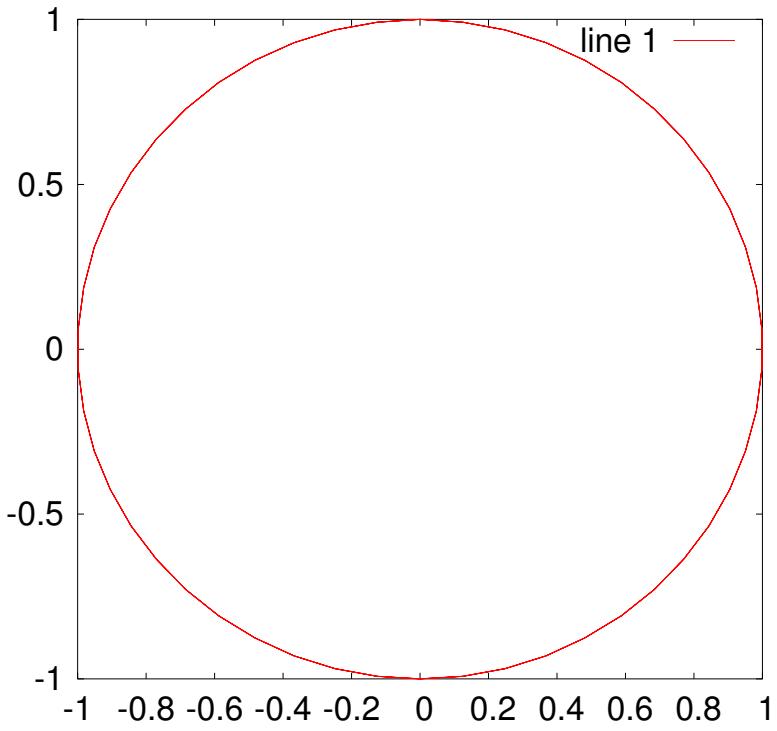
```
Ejemplo:

x=0:pi/25:6*pi;
plot(y,z)       Ejes en distinta escala
```




```
axis('equal')   Ejes en igual escala
replot
```

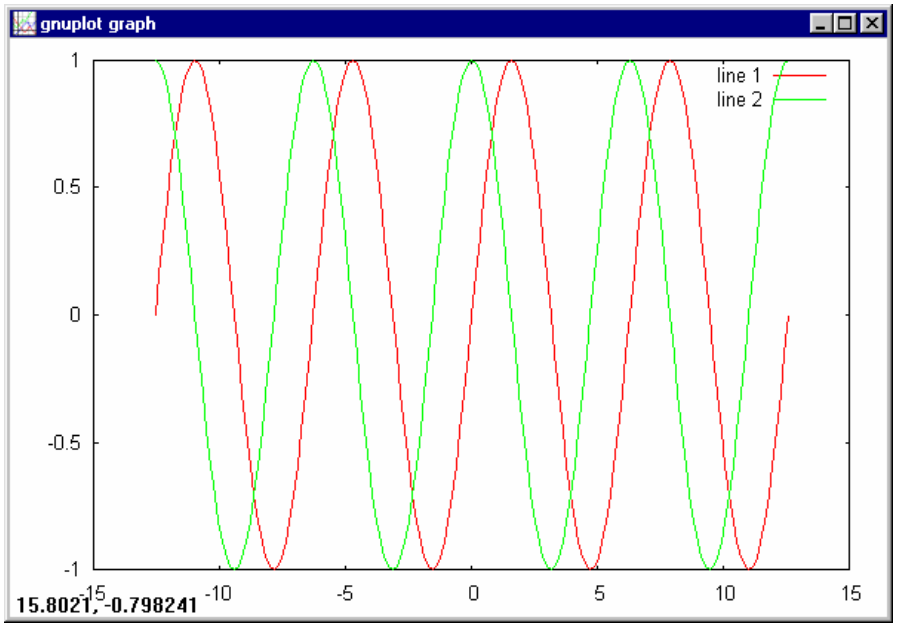



```
axis('square')  Ejes en igual escala y ceñidos a la figura
replot
```

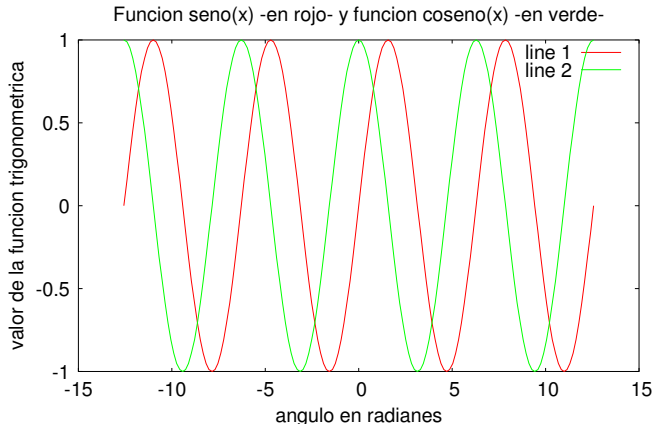


Ejemplos de uso desde la linea de comandos:

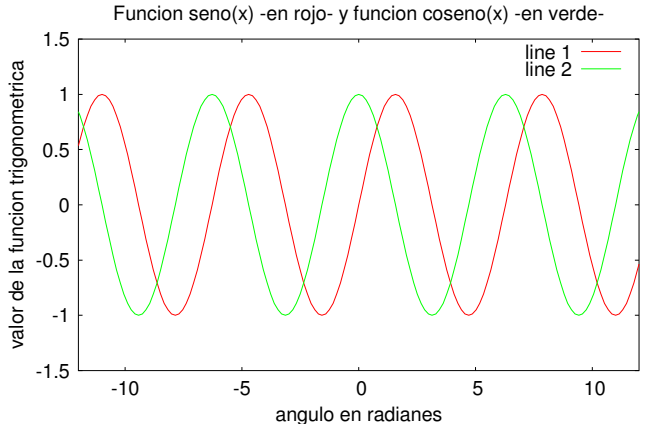
```
x=[-4*pi:pi/20:4*pi];
plot(x,sin(x),'r',x,cos(x),'g')
```



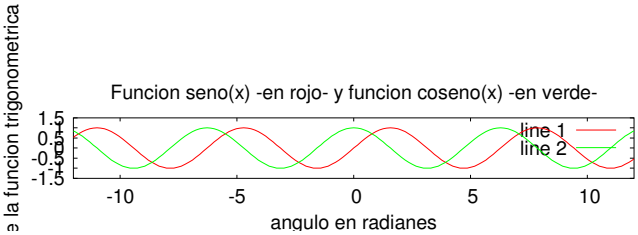
```
title('Función seno(x) -en rojo- y función coseno(x) -en verde-
')
replot
xlabel('ángulo en radianes'),replot
ylabel('valor de la función trigonométrica'), replot
```



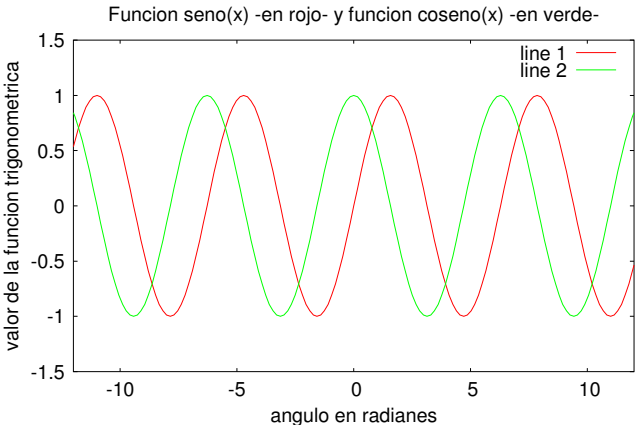

```
axis([-12,12,-1.5,1.5])
replot
```



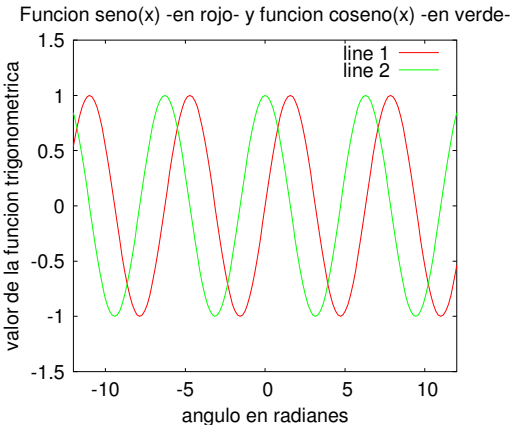

```
axis('equal')
replot
```




```
axis('normal')
replot
```

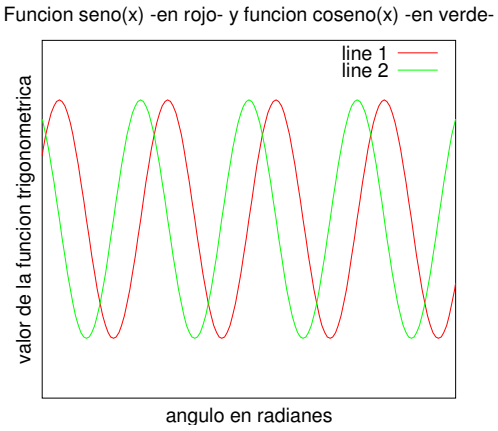



```
axis('square')
replot
```

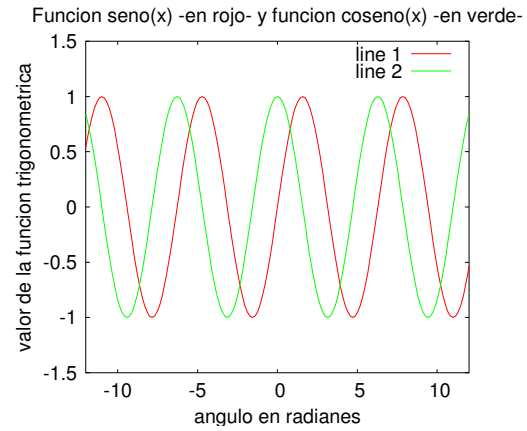



```
axis('off')
```

```
replot
```




```
axis('on')
```



EJERCICIOS PARA RESOLVER EN OCTAVE

1. Dadas las funciones: $f(x)= x^5 - x^2$; $f(x)= 3x/(x^2 -16)$; $f(x)= 5 +2(3-x)^3$

   Efectúe para cada una de ellas:

a) f(-3) b) f(4.5) c) f(-2/5) d) $f(\sqrt[5]{8})$ e) f(x+h) f) $\dfrac{f(x+h)-f(x)}{h}$

2. Grafique la siguientes funciones:

a) $f(x) = 3(2-x)^3 +4$ b) $f(x) = (7-x)^2 -1$ c) $f(x) = x^6 - x^2$ d) $f(x) = \sqrt[3]{3x+6}$

e) $f(x) = \sqrt{x^2 - 81}$ f) $f(x) = -3 + \sqrt{4+x}$ g) $f(x) = \dfrac{16x^2 + 48}{4x^2 - 16}$

h) f(x) = (x+2)/(3x-9)

Anexo 3: <u>Manual Técnico.</u>

Instalación de octave+forge en Windows (GNU Octave 2.1.73).

Este es el modo básico de octave para Windows con el cual podremos realizar todos los cálculos que necesitemos.

1. Vaya a la página http://sourceforge.net/projects/matlinks
2. Seleccione el octave versión para Windows.
3. Cuando haya terminado la descarga haga doble-click en el archivo ejecutable.

octave-2.1.73-1-inst.exe

Figura 1 Archivo ejecutable de GNU Octave 2.1.73

4. Siga las instrucciones que le indica el instalador

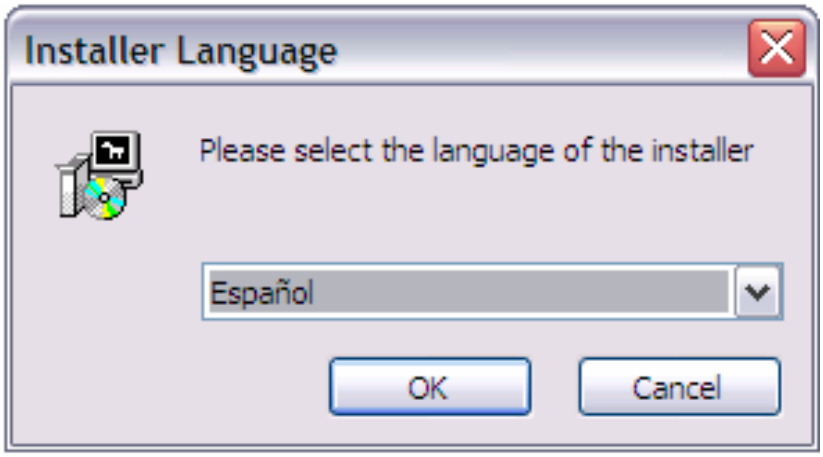


Figura 2 Pantalla de selección de idioma para la instalación de GNU Octave 2.1.73

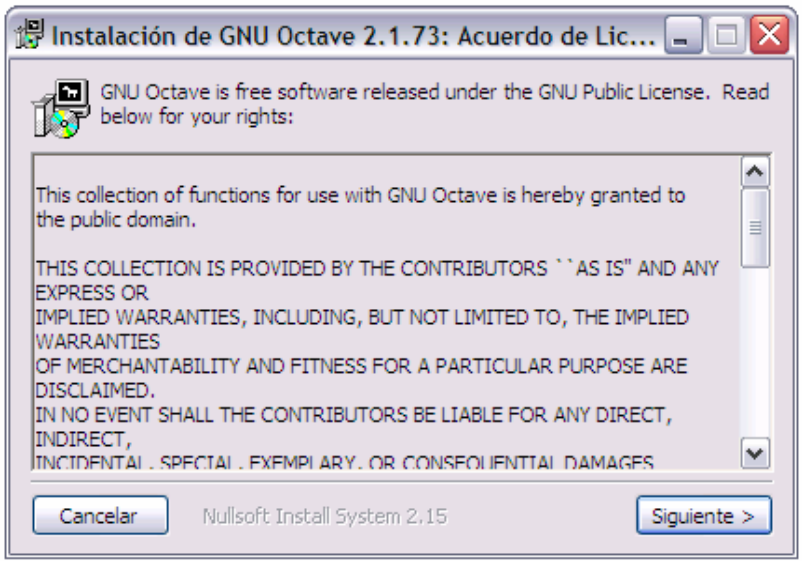


Figura 3 Pantalla de licencia de GNU Octave 2.1.73

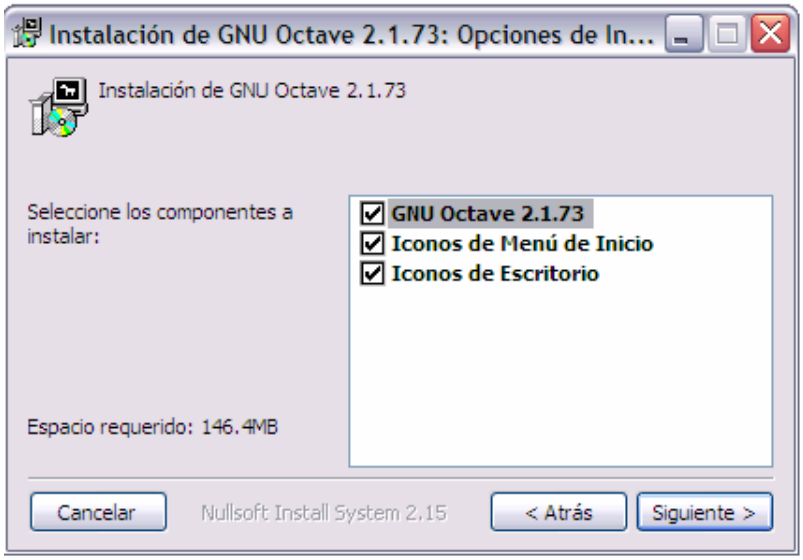


Figura 4 Pantalla de selección de componentes a instalar de GNU Octave 2.1.73.

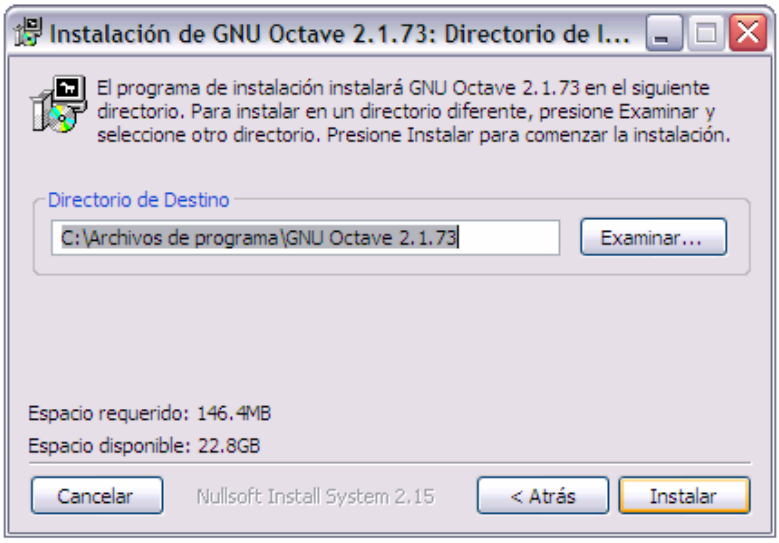


Figura 6 Pantalla en la cual se selecciona el directorio en el que se instalará GNU Octave 2.1.73

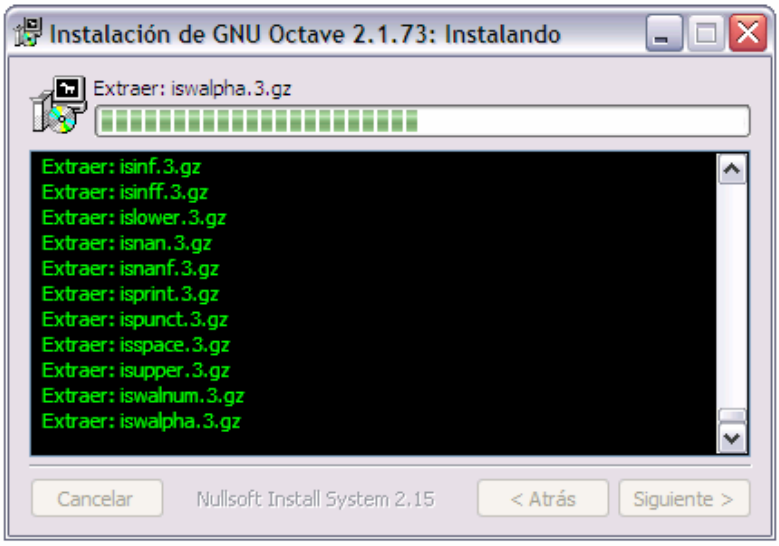


Figura 7 Pantalla de proceso de instalación de componentes de GNU Octave 2.1.73

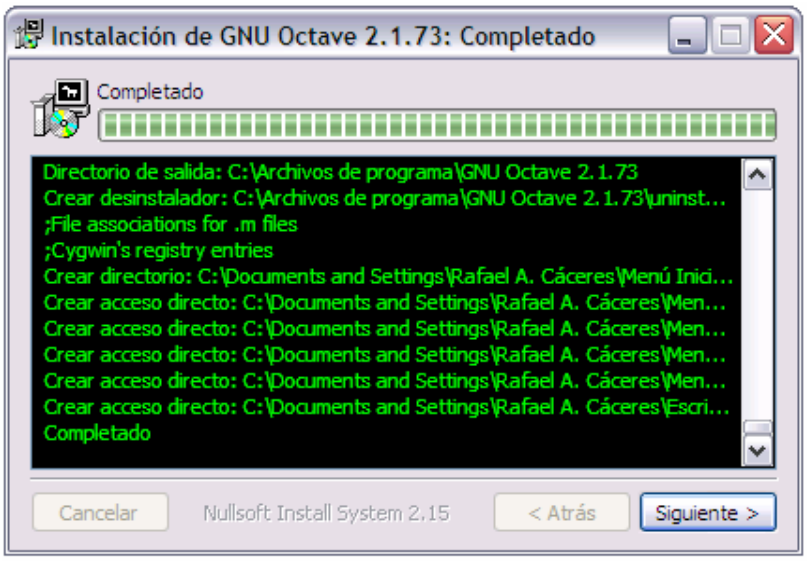


Figura 8 Finalización del proceso de instalación de GNU Octave 2.1.73

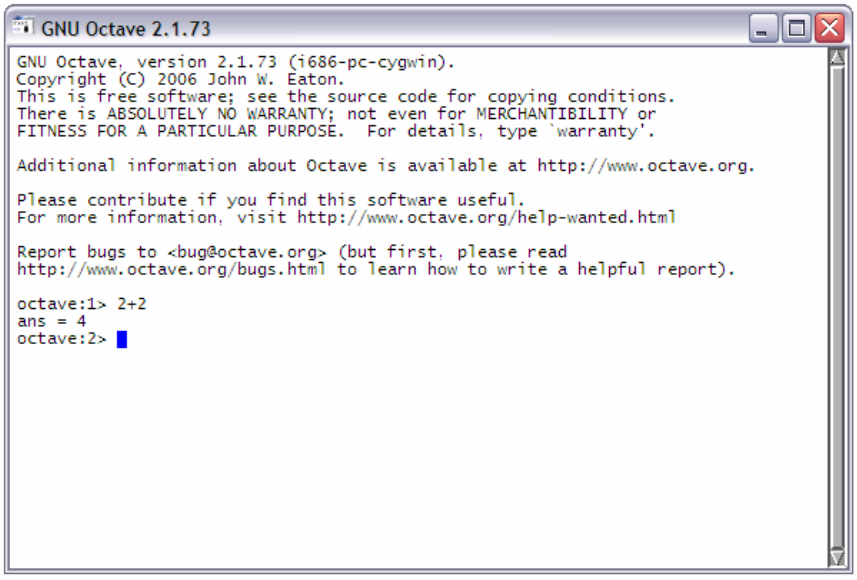


Figura 9 Entorno de trabajo de Octave de GNU Octave 2.1.73

Anexo 4: Manual de Usuario.

1 ¿Qué es QtOctave?

QtOctave es un front-end para Octave. Octave es una aplicación de cálculo matemático muy similar a Matlab.

2 Instalación.

Página principal donde se puede descargar QtOctave es **http://qtoctave.wordpress.com/**

Para instalar QtOctave se necesitan tener instalados:

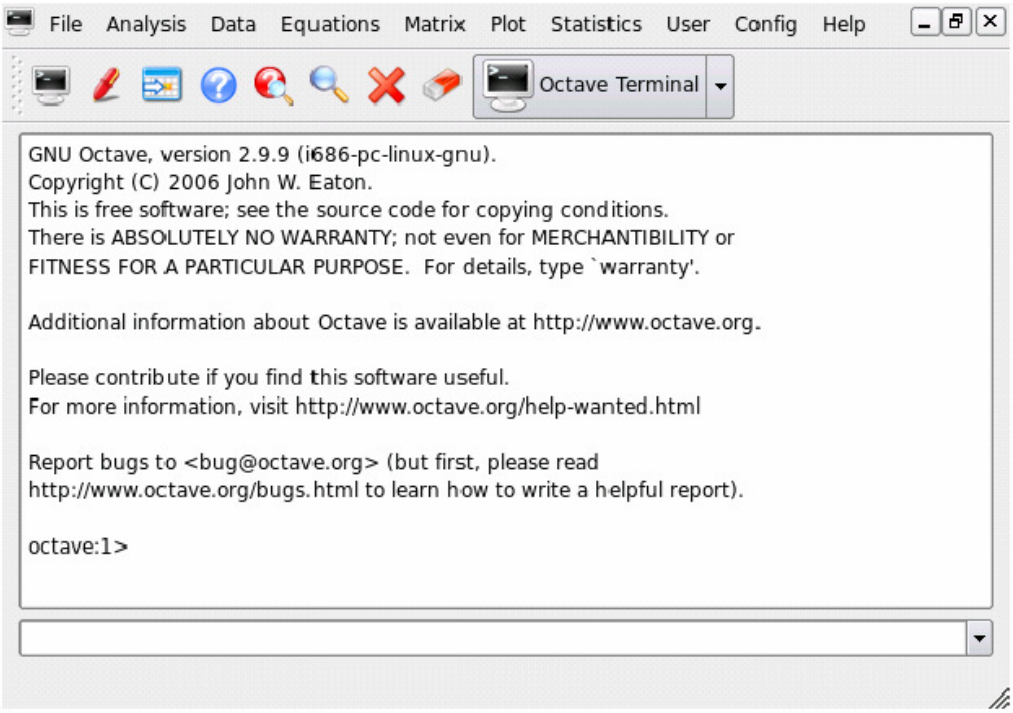
- Octave.

3 Usando QtOctave.

Una vez instalado QtOctave, para iniciarlo se teclea:

```
qtoctave
```

Si todo funciona correctamente, se mostrará una ventana como la siguiente:



En esta ventana se pueden apreciar los siguientes iconos:

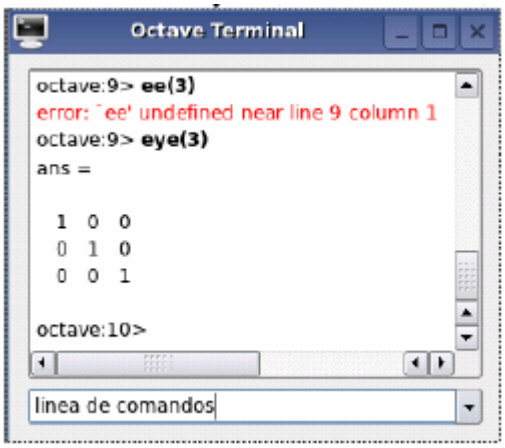
	Abre una nueva sesión de QtOctave.
	Abre un editor de texto. El editor se puede configurar en el menú Config.
	Abre una matriz en una hoja de cálculo.
	Abre la ayuda de Octave. Se puede configurar en el menú Config.
	Ayuda dinámica. Muestra ayuda de los comandos según se van tecleando en el terminal.
	Busca comandos similares al que se haya tecleado.
	Ctrl. + C. Sirve para parar procesos desbocados, bucles infinitos,...
	Borra los contenidos de el terminal.

Además, en la barra de iconos, aparece una lista desplegable con las ventanas abiertas.

En el caso de que no haya manejado nunca Octave, es recomendable usar la ayuda de octave  .

3.1 El terminal.

En el terminal se puede ejecutar comandos de Octave y ver, en una ventana, la salida de dichos comandos:



En la línea de comandos se pueden ejecutar nuevos comandos y se puede ver una lista de los comandos anteriores.

En el caso de que se produzca un error estos aparecerán en rojo.

Seleccionando un texto, en la salida, y haciendo click con en botón derecho del ratón, se pueden copiar fragmentos de texto de la salida.

A veces, el terminal se llena de "basura", por ello se puede usar el borrador  , para borrar el terminal.

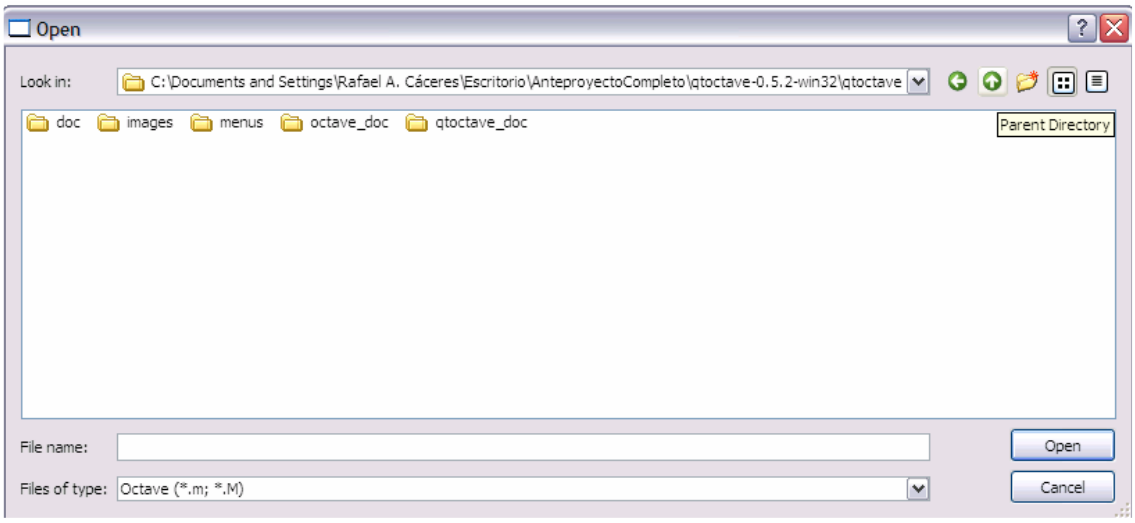
Puede ocurrir también que se entre en un bucle infinito. Por ejemplo, si se teclea:

```
while (2>1)
end
```

Se entrará en un bucle infinito. Con el botón  , se pueden parar los procesos desbocados o los bucles infinitos, como el anterior.

3.2 Navegando a un directorio.

Es habitual tener una directorio o carpeta con los ficheros de trabajo. Se puede usar la orden "cd" de Octave para navegar por los directorios. Pero es más práctico el menú "File/Change dir". Mostrará una ventana para seleccionar el directorio al que se quiera navegar y cambiará a dicho directorio.

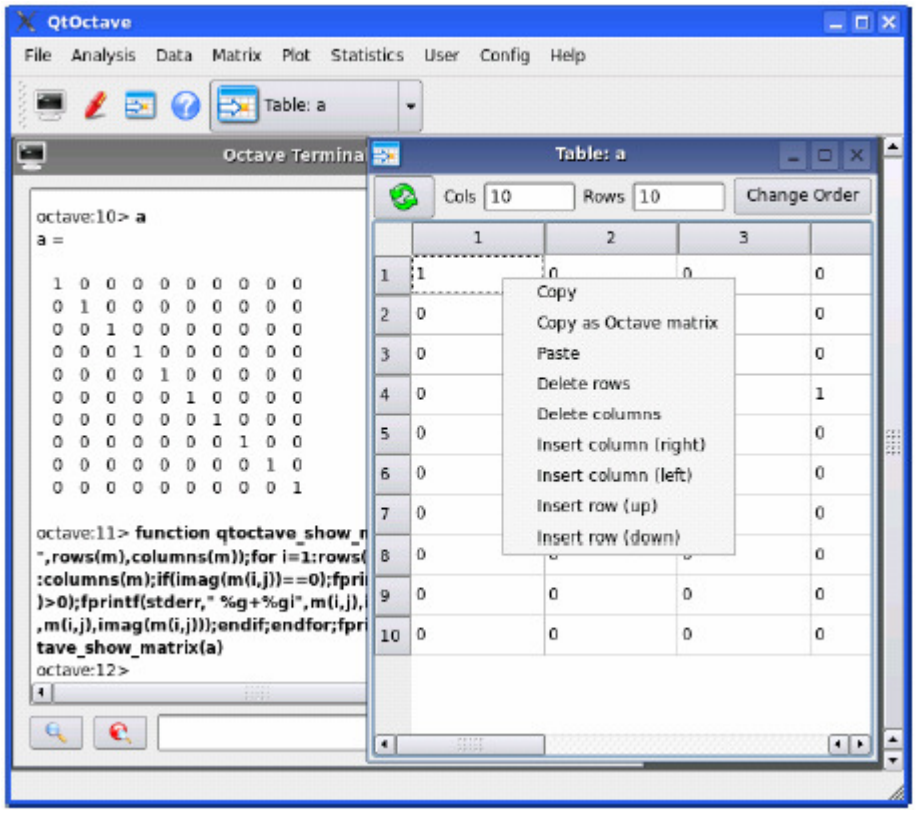


3.3 El menú Data.

Este menú posee herramientas para realizar manipulación de datos.

Una herramienta interesante es la utilidad "Table"  . Esta utilidad muestra matrices como si fueran hojas de cálculo.

Primero preguntará el nombre de la matriz que se desea mostrar. Si todo es correcto, mostrará una ventana con la matriz. Como la de la figura siguiente:



Se pueden seleccionar celdas, filas o columnas. Si se hace click con el botón derecho del ratón aparecerá un menú con las diferentes posibilidades. Se puede copiar, pegar, borrar, etc.

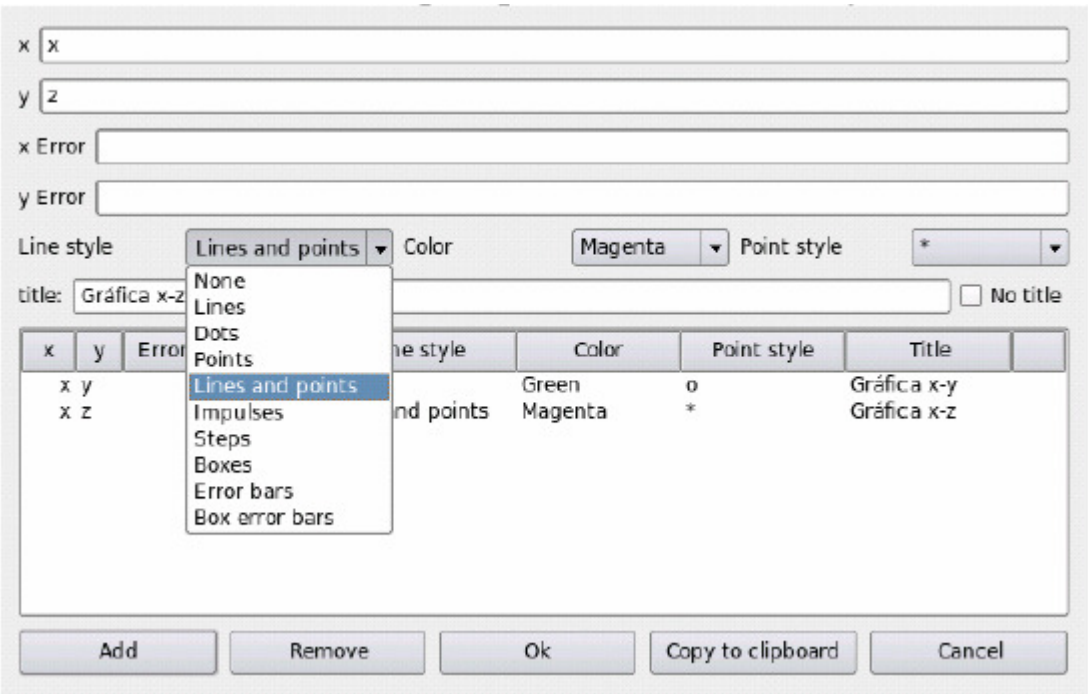
Una de las opciones más interesante es usar "Copy as Octave matrix". Esta opción permite copiar un trozo de matriz y después pegarla en un editor, usando la sintaxis de Octave para las matrices.

Las entradas "Cols" y "Rows", sirven para cambiar las filas y las columnas de la hoja de cálculo (el orden de la matriz).

A veces, se pueden realizar manipulaciones de matrices con Octave, que no se vean reflejadas en la hoja de cálculo. Usar el botón  para actualizar la hoja de cálculo.

3.4 El menú Plot.

Este menú tiene herramientas para trabajar con gráficas. Permite generar gráficas de una forma sencilla, usando un asistente, exportar los resultados en formatos eps ó pdf, dar títulos a las gráficas, etc.



Como en casi todos los asistentes de QtOctave, se pueden ejecutar los comandos necesarios en el terminal, o bien copiarlos al portapapeles, para pegarlos en un editor. QtOctave no sólo pretende dar un entorno amigable para lidiar con Octave, sino que también se intenta facilitar la labor de la programación en Octave.

Anexo 5: Documento utilizado en la demostración de Octave a los Catedráticos del Departamento de Ciencias Básicas de la Universidad Don Bosco.

1. SOFTWARE LIBRE.

El Software Libre es también conocido como Open Source software o Free Software.
El Software Libre tiene las siguientes características:

- Todo el mundo tiene derecho de usarlo, sin ninguna restricción.
- Todo el mundo tiene derecho a acceder a su diseño y aprender de él. Es como obtener las instrucciones para construir un carro.
- Todo el mundo tiene derecho de modificarlo: si el software tiene limitaciones, o no es adecuado para una tarea, es posible adaptarlo a sus necesidades específicas y redistribuirlo.
- No tiene un costo asociado (es gratuito).

Los derechos mencionados anteriormente tienen una serie de efectos colaterales:

- Tiende a ser muy eficiente (porque mucha gente lo optimiza, lo mejora).
- Tiende a ser muy robusto (mucha gente puede arreglarlo, no solamente el creador o la compañía que lo produce). Mucha gente tiende a contribuir, porque es del interés de todos mejorar esta base común.
- Tiende a ser muy diverso: la gente que contribuye tiene muchas necesidades diferentes y esto hace que el software esté adaptado a una cantidad más grande de problemas.

Estos derechos típicamente no están disponibles con el software propietario. Usualmente en el software propietario hay que pagar una licencia de uso al creador (como el pago de derechos por el uso de una patente) y está uno sujeto a las condiciones de uso del fabricante. Típicamente estas condiciones no otorgan ningún derecho al usuario final.

El éxito del Software Libre se debe en su mayor parte a Internet, porque esto ha permitido que las personas interesadas en los varios componentes del software libre se pongan en contacto con otras. Internet de esta manera actúa como un distribuidor y catalizador que acelera el desarrollo y el conocimiento en áreas muy específicas.
A continuación se describen algunas de las ventajas de utilizar software libre en la preparación de futuros profesionales, desde un punto de vista académico:

- Gracias a las cuatro libertades sobre las cuales se fundamente el software libre y en particular, la libertad de estudiar el software, se tiene acceso al código fuente de una gran variedad de programas y aprender de esa fuente inmensa de conocimiento. Así como los estudiantes de literatura deben leer montañas de obras, y los abogados en su preparación deben leer muchos libros de legislación, de igual forma los ingenieros de sistemas deben leer muchas líneas de código de programas para lograr una formación aceptable.

- Ante la rápida evolución tecnológica que estamos viviendo se ha llegado a la conclusión que la mejor herencia que la Universidad puede dejar es enseñar cómo funcionan las cosas, de tal forma que con unas sólidas bases se puedan comprender rápidamente las nuevas tecnologías que vayan apareciendo. Esto se puede hacer solamente con software libre, ya que con software propietario aspiramos, como máximo, a ser tan sólo expertos en el manejo de unas herramientas ignorando por completo cómo funcionan por dentro. Con software libre se tiene la valiosa ventaja de estudiar por dentro un motor de base de datos, un navegador Web, un compilador, un graficador, un sistema operativo, entre otros.

- Muchas personas tienen en sus equipos software propietario ilegal, quizá por que no tienen dinero suficiente para comprarlo o por ignorancia, ciertamente están cometiendo un delito. La solución ante este problema es utilizar software libre.

- Trabajar con software libre disminuye la brecha tecnológica entre los países del tercer y primer mundo. Los estudiantes pueden prepararse estudiando y participando de proyectos de software reales, compuestos por equipos de trabajo brillantes, distribuidos geográficamente por todo el mundo, y mejor aún, trabajar en nuestros propios requerimientos tecnológicos. Lo peor que nos puede pasar es convertirnos en consumidores de tecnologías de países avanzados porque esta situación implica estar condenados al sometimiento.

- Cada vez son más las empresas e instituciones que se migran al software libre, guiados inicialmente por el estímulo de ahorrar dinero (aunque luego se dan cuenta de otras características como calidad, desempeño y seguridad), en licencias costosas, esto hace; que el mercado laboral requiera ingenieros y técnicos con conocimientos en estas tecnologías.

- Las universidades, anualmente, tienen que renovar los centros de cómputo con casas desarrolladoras como Microsoft, para poder utilizar sus productos. Al adoptar software libre, se ahorran miles de dólares, que pueden ser invertidos en otras áreas como la ampliación de salas de cómputo, dotación de bibliotecas, ampliación de planta física, etc.

- Al egresar de las universidades, los ingenieros pueden fácilmente constituir empresas porque con software libre no necesitan capital inmenso en licencias de paquetes de software, tan solo necesitan, su iniciativa e intelecto.

Con lo anterior no se afirma que sea conveniente migrarse completamente al software libre, pues el mercado laboral está rodeado de software propietario, lo importante es que los futuros ingenieros salgan capacitados en los dos ámbitos de trabajo y al momento de asesorar una empresa tengan ambas opciones, comparar tecnologías, sus ventajas, desventajas y brindar la mejor elección.

El uso de programas propietarios en nuestro país, tiende a fomentar la piratería entre los estudiantes universitarios; esto porque carecen de fondos para pagar las licencias

correspondientes. El software libre es una forma de disminuir la piratería y la preocupación de inversión financiera en licencias de software.
En conclusión, el beneficio de un proyecto de esta naturaleza es para el sector educativo representado por la Universidad Don Bosco.

Hay diferentes motivaciones que impulsan a los contribuidores y desarrolladores a trabajar en el software libre, las más importantes son:

- El deseo de crear software más robusto.
- La posibilidad de estar en control del software. Esto es importante para aplicaciones de misión crítica donde es totalmente imperante tener un control total sobre posibles problemas en cualquier punto.
- Crear aplicaciones de bajo costo.
- Reutilización del conocimiento: Esto permite que la gente reutilice el conocimiento que se ha sintetizado en el software. En vez de empezar siempre desde cero (que es el caso de la industria de software actual) siempre se puede empezar un proyecto desde un fundamento establecido. Esto es equivalente a la manera en la que la ciencia se desarrolla: no se parte de cero, se parte de los descubrimientos previos y se innova sobre el conocimiento que ya se tiene.
- La posibilidad de adaptar el software a sus necesidades.
- Aprender alguna técnica de programación.

Lo mencionado arriba ha dado cabida a que se creen sistemas de cómputo que compiten en casi todos los niveles con los sistemas propietarios, pero no contemplan sistemas de marketing y son tradicionalmente esfuerzos que no son conocidos por el público en general.

2. VENTAJAS DEL SOFTWARE LIBRE EN TÉRMINOS DE COSTOS.

El pago de una licencia de un sistema Windows hoy en día oscila entre los doscientos dólares hasta los trescientos dólares dependiendo de la versión. En el caso de servidores estamos hablando de unos cuatro mil dólares solamente en licencias por cada computadora que se desee poner en línea.

Si a estos sistemas se les va a incluir un sistema de productividad del tipo Office, los costos de las estaciones de trabajo saltan al rango 700-900 dólares por estación.

Este costo no es significativo si estamos hablando de un par de estaciones de trabajo, pero cuando multiplicamos el costo por estación de trabajo a un par de millones, estamos hablando de unas cantidades de dinero muy grandes.

Un ejemplo es tan solo el proyecto Red Escolar en que México tiene planeado instalar un millón de computadoras en 5 años. Es decir, que en estos sistemas se va a pagar entre 200 y 700 millones de dólares exclusivamente en licencias.

Una licencia es el "derecho" de utilizar el software. Físicamente, la compañía que manufactura el software solamente tiene que hacer una copia del compacto en la que se distribuye el software y vender las copias.

Cada licencia que se paga es equivalente aproximadamente al costo de una computadora nueva.

Es decir que el costo en términos de equipo de cómputo y de software puede reducirse a la mitad o se puede instalar el doble de equipo por el mismo precio.

3. ¿QUÉ ES OCTAVE?

Octave es un software que nos permite programar y utilizar una serie de funciones principalmente numéricas. Octave ofrece una interfaz de usuario interactiva, orientada a línea de comandos, pero también puede ser utilizado en modo no interactivo, leyendo sus órdenes de fichero.

Octave es un software de distribución gratuita que se apega a la filosofía GNU, esto es poder tener acceso al programa y al código fuente del programa para modificarlo (si nos interesa), sin tener que pagar ni por su uso, ni por su obtención, además de poder hacer cuantas copias se quieran e instalaciones en diferentes máquinas, también, sin tener que pagar.

3.1 DETALLES TÉCNICOS DE OCTAVE

- Octave está escrito en C++.
- Tiene un intérprete que interpreta su propio lenguaje (de sintaxis similar a Matlab), y permite una ejecución interactiva.
- Puede extenderse el lenguaje con funciones y procedimientos.
- Utiliza otros programas GNU para ofrecer al usuario la creación de gráficos para luego imprimirlos o guardarlos (Gnuplot).
- Además de correr en plataformas Unix también lo hace en Windows.
- Puede cargar archivos con funciones de Matlab de extensión .m.

3.2 CARACTERÍSTICAS DE OCTAVE

- La sintaxis es similar a la utilizada en C.
- Es un lenguaje interpretado.
- Se pueden generar scripts.
- Soporta gran parte de las funciones de la librería estándar de C.
- Soporta estructuras similares a los structs de C.
- Al ser su licencia GNU General Public License, puede ser copiado y utilizado libremente.

3.3 INSTALACIÓN DE OCTAVE

Existen varias versiones de Octave, todas disponibles en forma gratuita en Internet. La página principal de octave es http://www.octave.org . La versión original se utiliza en LINUX (un sistema operativo gratuito), a pesar de ello existen versiones para

Windows (el sistema operativo propietario de Microsoft). Una de las páginas de donde se puede obtener la distribución para Windows, es la página de la Universidad de Córdoba que es http://www.efn.uncor.edu. El link que nos permite bajar el instalador de Octave para Windows es http://www.efn.uncor.edu/departamentos/computacion/materias/informatica/software.html

Una vez obtenido el instalador simplemente se debe ejecutar el mismo y se iniciará el proceso de instalación de OCTAVE.

INSTALACIÓN DE OCTAVE+FORGE EN WINDOWS (GNU OCTAVE 2.1.73).

Este es el modo básico de octave para Windows con el cual podremos realizar todos los cálculos que necesitemos.

1. Vaya a la página http://sourceforge.net/projects/matlinks
2. Seleccione el octave versión para Windows.
3. Cuando haya terminado la descarga haga doble-click en el archivo ejecutable.

octave-2.1.73-1-inst.exe

Figura 1 Archivo ejecutable de GNU Octave 2.1.73

4. Siga las instrucciones que le indica el instalador

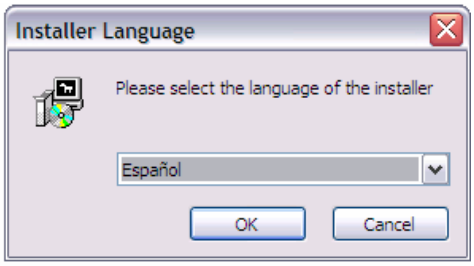


Figura 2 Pantalla de selección de idioma para la instalación de GNU Octave 2.1.73

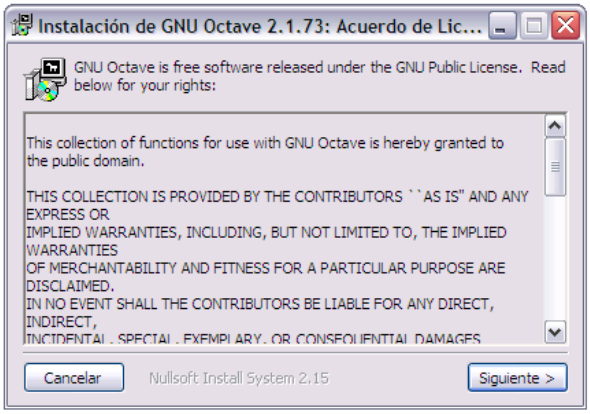


Figura 3 Pantalla de licencia de GNU Octave 2.1.73

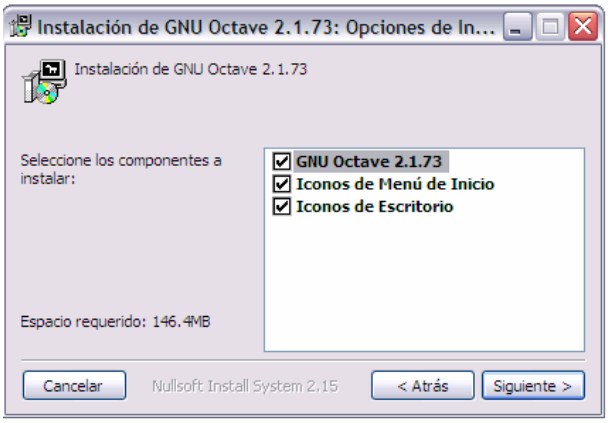


Figura 4 Pantalla de selección de componentes a instalar de GNU Octave 2.1.73.

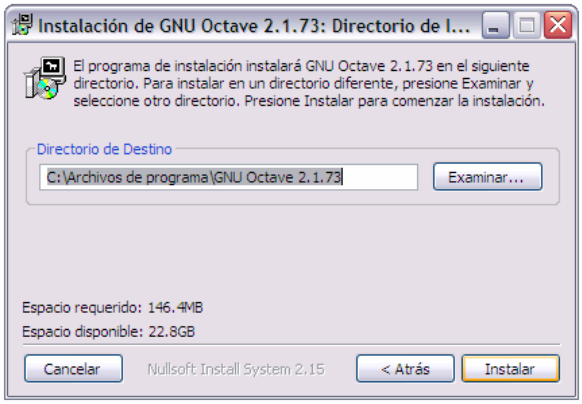


Figura 5 Pantalla en la cual se selecciona el directorio

en el que se instalará GNU Octave 2.1.73

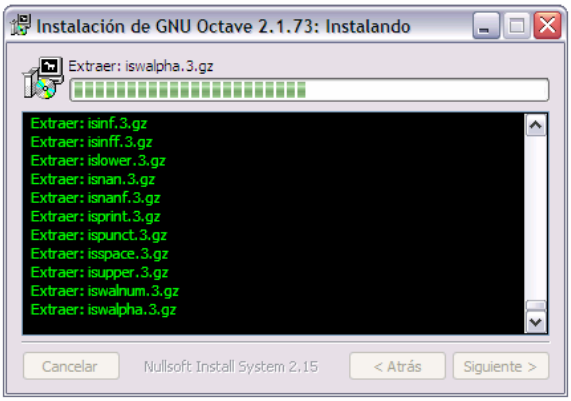


Figura 6 Pantalla de proceso de instalación de componentes de GNU Octave 2.1.73

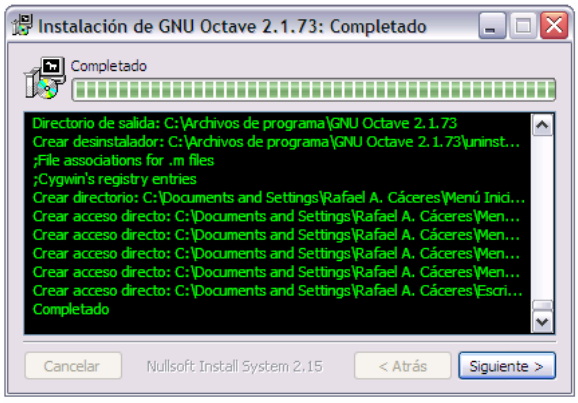


Figura 7 Finalización del proceso de instalación de GNU Octave 2.1.73

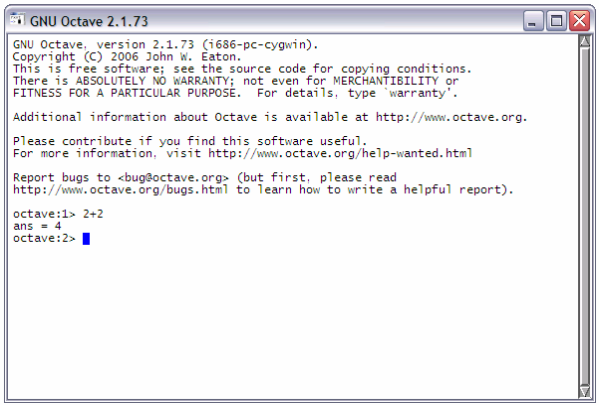


Figura 8 Entorno de trabajo de Octave de GNU Octave 2.1.73

3.4 ASPECTOS DE EVALUACIÓN.

Se evaluaran las características más relevantes en los programas Matlab y Octave. Para ello se pide que se evalúe de 1 al 4, donde 1 = Malo, 2 = Regular, 3 = Bueno, 4 = Muy bueno, cada uno de los aspectos que a continuación se detallan.

Característica	**MatLab**	**Octave**
Funcionalidades básicas	4	4
Funcionalidades avanzadas	4	4
Gráficos e imágenes	4	4
Potencia del Lenguaje de Programación	3	3
Control de la precisión	2	2
Fiabilidad	4	4
Rapidez	4	3
Información	4	2
Facilidad de manejo	4	3
Licencia y facilidad de obtención	1	4
Desarrollo y madurez	4	4
Instalación	3	4
Compatibilidad con otros programas	4	4
Integración de otros Lenguajes de Programación	3	4

Tabla 1. Comparación entre las herramientas Matlab y Octave.

Programa	**Puntos Fuertes**	**Puntos Débiles**
Matlab	Gran número de funciones y ToolBox. Diseño apoyado en interfaces gráficas. Información sobre el programa.	Licencia
Octave	Gran cantidad de funciones y paquetes. Alta compatibilidad con MatLab. Licencia.	Información sobre el programa.

Tabla 2. Puntos fuertes y débiles de los programas analizados.

3.5 COMPARACIÓN ECONÓMICA.

En la Tabla 3 se presenta la cotización más actualizada sobre la suscripción al servicio de mantenimiento de software, la cual muestra los cargos por restablecimiento ("*reinstatement*") y por la compra del servicio de mantenimiento, para un período equivalente al lapso transcurrido sin la suscripción, más un año. El servicio de mantenimiento de software es equivalente a poder contar en todo momento con la versión más reciente de todos los productos adquiridos.

No. Lic.	Producto	Reingreso (reinstatement)		Servicio de mantenimiento de Software					TOTAL por *Producto*
				Periodo de Mantenimiento		No. Meses	Costo Mensual	Total Serv. Mant.	
		Precio Unitario	Total por Reingreso	Inicio	Final				
55	MATLAB®	$6.00	$330.00	01-Ene 2001	01-Ene 2007	72	$2.00	$7,920.00	$8,250.00
55	Simulink®	$3.00	$165.00	01-Ene 2001	01-Ene 2007	72	$1.00	$3,960.00	$4,125.00
5	Communications Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
50	MATLAB® Compiler	$6.00	$300.00	01-Ene 2001	01-Ene 2007	72	$2.00	$7,200.00	$7,500.00
5	Control System Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Data Acquisition Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Signal Processing Blockset	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
50	Excel Link	$1.75	$87.50	01-Ene 2001	01-Ene 2007	72	$0.58	$2,088.00	$2,175.50
5	Fuzzy Logic Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Simulink® Fixed Point	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	System Identification Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Image Processing Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Neural Network Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Fixed-Point Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
50	SimPowerSystems	$6.00	$300.00	01-Ene 2001	01-Ene 2007	72	$2.00	$7,200.00	$7,500.00
5	Real-Time Workshop®	$17.50	$87.50	01-Ene 2001	01-Ene 2007	72	$5.83	$2,098.80	$2,186.30
5	Simulink® Control Design	$0.00	$0.00	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$687.60
5	Stateflow®	$17.50	$87.50	01-Ene 2001	01-Ene 2007	72	$5.83	$2,098.80	$2,186.30
5	Signal Processing Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Statistics Toolbox	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
5	Communications Blockset	$5.75	$28.75	01-Ene 2001	01-Ene 2007	72	$1.91	$687.60	$716.35
	TOTAL por rubro	**$132.50**	**$1,731.25**				**$45.98**	**$42,192.00**	**$43,923.25**

Tabla 3. Costos de renovación del servicio de mantenimiento

El rubro por restablecimiento (*reinstatement*) es un cargo que MathWorks aplica a los clientes que no han renovado el servicio de mantenimiento dentro de los tres meses siguientes a la fecha de vencimiento de la suscripción anterior. Por lo tanto, si la renovación a la suscripción de mantenimiento se realiza en los plazos establecidos, el único cargo es el del propio servicio de mantenimiento, que para el caso de 12 meses, y según la clase de productos y cantidad de licencias de la Universidad, sería igual a $7,032.00 (según puede deducirse de la Tabla 3). Esta cantidad anual equivale aproximadamente al pago de laboratorio de 243 alumnos. Por otro lado, debe explorarse también la alternativa mencionada arriba, que consiste en no suscribirse al servicio de mantenimiento, y comprar las versiones actualizadas de los productos de interés, al precio actual (es decir, no se trataría de una actualización de los productos ya adquiridos, sino de la compra de los productos actualizados). En la Tabla 4 se muestran los costos de los productos actualizados, considerando la misma cantidad y tipo de productos adquiridos originalmente en la Universidad.

No. Lic.	**Producto**	**Costo unitario**	**TOTAL por Producto**
55	MATLAB®	$120.00	$6,600.00
55	Simulink®	$60.00	$3,300.00
5	Communications Toolbox	$115.00	$575.00
50	MATLAB® Compiler	$120.00	$6,000.00
5	Control System Toolbox	$115.00	$575.00
5	Data Acquisition Toolbox	$115.00	$575.00
5	Signal Processing Blockset	$115.00	$575.00
50	Excel Link	$35.00	$1,750.00
5	Fuzzy Logic Toolbox	$115.00	$575.00
5	Simulink® Fixed Point	$115.00	$575.00
5	System Identification Toolbox	$115.00	$575.00
5	Image Processing Toolbox	$115.00	$575.00
5	Neural Network Toolbox	$115.00	$575.00
5	Fixed-Point Toolbox	$115.00	$575.00
50	SimPowerSystems	$120.00	$6,000.00
5	Real-Time Workshop®	$350.00	$1,750.00
5	Simulink® Control Design	$115.00	$575.00
5	Stateflow®	$350.00	$1,750.00
5	Signal Processing Toolbox	$115.00	$575.00
5	Statistics Toolbox	$115.00	$575.00
5	Communications Blockset	$115.00	$575.00
		TOTAL	**$35,200.00**

Tabla 4. Costo de los productos actualizados.

Al comparar los totales de ambas alternativas (Tabla 3 y Tabla 4), puede observarse que actualmente es preferible comprar los productos otra vez, que actualizarlos por medio del servicio de mantenimiento. Esto se debe a que el lapso transcurrido sin suscripción al servicio de mantenimiento ha sido muy largo.

La inversión es demasiado elevada para realizarla, es preferible optar por La utilización de GNU Octave que cubre las mismas necesidades que Matlab en las asignaturas de Matemáticas de la Universidad Don Bosco y a la vez no hay necesidad de invertir en nuevo hardware, ya que GNU Octave 2.1.73 puede operar perfectamente en el equipo actual.

4. DEMOSTRACIÓN DE OCTAVE

OPERACIONES CON MATRICES Y VECTORES

Una de las ventajas de los lenguajes interpretados es su condición de estar ejecutándose en forma permanente, lo que comparten con otros sistemas de simulación, que en el caso de OCTAVE se puede dar, en éste sentido, una definición como calculadora Matricial. Es decir que a su condición de calculadora de funciones numéricas, que comparte con las calculadoras manuales, se le debe agregar su capacidad de realizar operaciones con matrices y vectores de una manera altamente eficiente, tanto desde su poder de representación y análisis como desde la capacidad de procesamiento.

Creación de una matriz 3x3:

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

A partir de este momento A (matriz 3x3) queda definida en el espacio de trabajo y se pueden realizar sobre ella todo tipo de operaciones matriciales:

Transpuesta de A:

```
A'
ans =       ans: variable por defecto del último cálculo
            si no se asigna a otra variable nominada
     1      4      7
     2      5      8
     3      6      9

B=A'
B =
     1      4      7
     2      5      8
     3      6      9
```

Operación de producto entre dos matrices previamente definidas y compatibles:

```
A*B
ans =
    14     32     50
    32     77    122
    50    122    194
```

Un vector fila se corresponde con una matriz de una sola fila, y el acceso a sus elementos se realiza mediante un índice de la posición que corresponde solamente a la columna:

```
x = [4 3 2 1]
x =
         4         3         2         1
```

```
x(3)
ans = 2
```

Si bien las matrices se pueden *introducir* por filas son *almacenadas* concatenando columnas en un único vector al que se puede acceder con un único índice, pero es más simple hacerlo mediante la notación de dos índices (fila, columna):

```
A(4)
ans =
     2

A(1,2)
ans =
    2
```

La operación de inversión matricial se realiza con una función:

Sea una nueva definición de la matriz A:

```
A=[1 4 -3; 2 1 5; -2 5 3]
A =
     1     4    -3
     2     1     5
    -2     5     3

B=inv(A)
B =
    0.1803    0.2213   -0.1885
    0.1311    0.0246    0.0902
   -0.0984    0.1066    0.0574
```

Si se desea visualizar una mayor precisión usar la función:

```
format  long
B
B =
   0.180327868852459  0.221311475409836  -0.188524590163934
   0.131147540983607  0.024590163934426   0.090163934426230
```

```
  -0.098360655737705   0.106557377049180   0.057377049180328
```

OCTAVE puede operar con matrices por medio de operadores y por medio de funciones. Se han visto ya los operadores suma (+), producto (*) y transpuesta ('), así como la función invertir inv( ).

Los operadores matriciales de OCTAVE son los siguientes:

- **+** adición o suma
- **-** sustracción o resta
- ***** multiplicación
- **'** transpuesta
- **^** potenciación
- **** división-izquierda
- **/** división-derecha
- **.*** producto elemento a elemento
- **./** y **.** división elemento a elemento
- **.^** elevar a una potencia elemento a elemento

Estos operadores se aplican también a las variables o valores escalares.
Considérese el siguiente ejemplo:

```
A=[1 2; 3 4]
A =
   1 2
   3 4

A*2
ans =
     2 4
     6 8
```

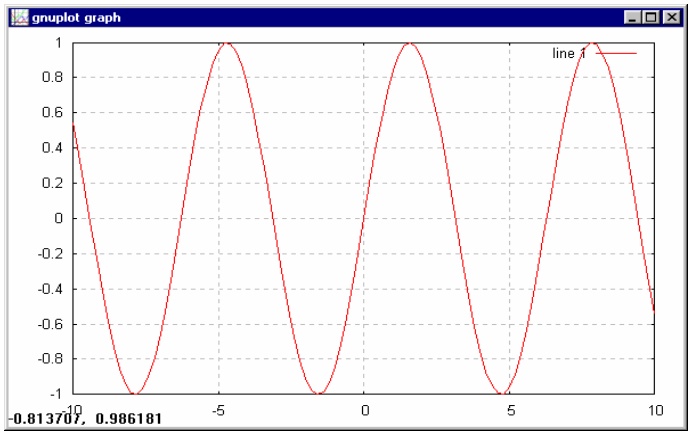
```
A-4
ans =
    -3  -2
    -1   0
```

GRÁFICOS CIENTÍFICOS.

1. GRÁFICAS EN 2D

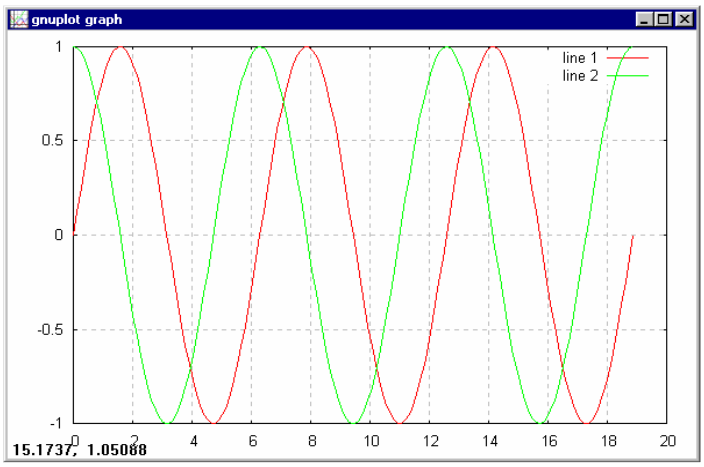
Función seno:

```
x=[-10:0.2:10];
y=sin(x);
```



Obsérvese bien cómo se dibujan el seno y el coseno en el ejemplo:

```
x=0:pi/25:6*pi;
y=sin(x);
z=cos(x);
plot(x,y,x,z)
```



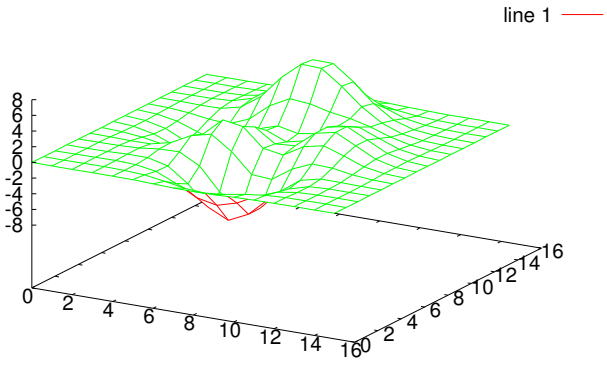
2. GRÁFICAS EN 3D

Se define una función *z* de dos variables en *x* e *y* en un archivo llamado **Test3d.m**:

```
function z=Test3d(x,y)
%TEST3D función de dos variables utilizada para probar
%       gráficos 3D
    z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
         - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
    - 1/3*exp(-(x+1).^2 - y.^2);
end
```

Para graficarla se llama al procedimiento **PTest3d.m**

```
% Procedimiento Ptest3d.m
x = [-3:0.4:3]; y = x;
closeplot
[X,Y] = meshgrid(x,y);
Z=Test3d(X,Y);
mesh(Z)
```



Anexo 6: <u>Encuesta.</u>

ENCUESTA PARA CATEDRÁTICOS DEL DEPARTAMENTO DE CIENCIAS BÁSICAS DE LA UNIVERSIDAD DON BOSCO.

1. ¿Utiliza algún programa matemático para el desarrollo de ejercicios diferente de Matlab?

   Si ☐ No ☐

   Especifique:_____

   _____

2. ¿Considera que los programas matemáticos facilitan la comprensión de los contenidos de esta materia a los alumnos?

   Si ☐ No ☐

3. ¿Considera necesario tener otras alternativas de herramienta matemática en el Laboratorio de Matemáticas y Simulación?

   Si ☐ No ☐

4. ¿Cómo califica al Software Libre?

   ☐ Malo
   ☐ Bueno
   ☐ Muy Bueno
   ☐ Excelente

5. ¿Utilizaría Octave como herramienta complementaria en la metodología de las asignaturas que imparte?

   Si ☐ No ☐

ENCUESTA PARA ALUMNOS DE LAS ASIGNATURAS DE MATEMATICAS DE LA UNIVERSIDAD DON BOSCO.

1. ¿Tiene instalado Matlab en su computadora?

   Si ☐ No ☐

2. ¿Utiliza algún programa matemático para el desarrollo de ejercicios diferente de Matlab?

   Si ☐ No ☐

   Especifique:_____

   _____

3. ¿Utilizaría un programa matemático diferente de Matlab como herramienta complementaria?

   Si ☐ No ☐

4. ¿Considera que los programas matemáticos facilitan la comprensión de los contenidos vistos en clase?

   Si ☐ No ☐

Anexo 7: <u>Resultado de Encuestas.</u>

Se entiende por Universo al conjunto de individuos u objetos de los que se desea conocer algo en la investigación.
Para este proyecto se tomó como Universo a los alumnos que cursan la materia Matlab y docentes del departamento de ciencias básicas de la Universidad Don Bosco.

Por muestra se conoce, el subconjunto o parte del Universo en que se llevará a cabo la investigación.
La muestra que se utilizó para la investigación de este proyecto fue de 26 alumnos y 6 docentes.

ENCUESTA PARA CATEDRÁTICOS DEL DEPARTAMENTO DE CIENCIAS BÁSICAS DE LA UNIVERSIDAD DON BOSCO.

1. ¿Utiliza algún programa matemático para el desarrollo de ejercicios diferente de Matlab?
   Respuestas:
   - Si : Cantidad = 0, Porcentaje = 0%
   - No: Cantidad = 6, Porcentaje = 100%

2. ¿Considera que los programas matemáticos facilitan la comprensión de los contenidos de esta materia a los alumnos?
   - Si : Cantidad = 6, Porcentaje = 100%
   - No: Cantidad = 0, Porcentaje = 0%

3. ¿Considera necesario tener otras alternativas de herramienta matemática en el Laboratorio de Matemáticas y Simulación?
   - Si : Cantidad = 6, Porcentaje = 100%
   - No: Cantidad = 0, Porcentaje = 0%

4. ¿Cómo califica al Software Libre?
    - Bueno: Cantidad = 1, Porcentaje = 16.67%
    - Muy bueno: Cantidad = 1, Porcentaje = 16.67%
    - Excelente: Cantidad = 4, Porcentaje = 66.66%

5. ¿Utilizaría Octave como herramienta complementaria en la metodología de las asignaturas que imparte?
    - Si : Cantidad = 6, Porcentaje = 100%
    - No: Cantidad = 0, Porcentaje = 0%

ENCUESTA PARA ALUMNOS DE LAS ASIGNATURAS DE MATEMATICAS DE LA UNIVERSIDAD DON BOSCO.

1. ¿Tiene instalado Matlab en su computadora?
    - Si : Cantidad = 26, Porcentaje = 100%
    - No: Cantidad = 0, Porcentaje = 0%

2. ¿Utiliza algún programa matemático para el desarrollo de ejercicios diferente de Matlab?
    - Si : Cantidad = 0, Porcentaje = 0%
    - No: Cantidad = 26, Porcentaje = 100%

3. ¿Utilizaría un programa matemático diferente de Matlab como herramienta complementaria?
    - Si : Cantidad = 21, Porcentaje = 80.76%
    - No: Cantidad = 5, Porcentaje = 19.24%

4. ¿Considera que los programas matemáticos facilitan la comprensión de los contenidos vistos en clase?
    - Si : Cantidad = 26, Porcentaje = 100%
    - No: Cantidad = 0, Porcentaje = 0%

CONCLUSION :

- Se tendrá una herramienta más para facilitar el trabajo entre alumnos y docentes, y así poder ser un buen método para reforzar los conocimientos vistos en clase y a la vez aprovechar al máximo el uso de la tecnología.
- Se facilitará el estudio de los alumnos.
- Los alumnos podrán exponer sus dudas.