Matemáticas en el Aula de Informática

Roberto Rodríguez Del Río
Departamento de Matemática Aplicada
Facultad de Ciencias Químicas
Universidad Complutense Madrid
rrdelrio@sunma4.mat.ucm.es

UNIVERSIDAD INTERNACIONAL MENÉNDEZ PELAYO
Santander, septiembre de 1.999

Índice

Introducción

En este curso veremos algunos de los recursos que puede ofrecer el Aula de Informática para la enseñanza de las Matemáticas, centrándonos particularmente en la realización de prácticas con paquetes informáticos de Cálculo Simbólico. Los paquetes informáticos que se han elegido han sido DERIVE y MATLAB.

Las razones para elegir DERIVE son evidentes, se trata de uno de los programas informáticos de Matemáticas más populares, de fácil manejo y con pocos requerimientos sobre el ordenador en el que se utiliza. Estas razones hacen de él una herramienta muy útil para apoyar el aprendizaje de las Matemáticas a niveles elementales y no tan elementales.

Las razones para elegir MATLAB son más subjetivas, sin embargo, hay que decir que actualmente es uno de los paquetes informáticos más utilizados en la Universidad y en la Industria. A pesar de que un conocimiento completo de MATLAB es prácticamente imposible, la incorporación en las últimas versiones del programa de ciertos elementos más básicos han hecho que empiece a difundirse su uso entre estudiantes de primer ciclo de carreras universitarias. En las Enseñanzas Medias no se puede decir que sea muy conocido, más bien todo lo contrario, y ésta es una de las razones que nos han animado a introducirlo en este curso. Existen otros programas de Cálculo Simbólico muy eficaces, como MAPLE o MATHEMATICA, que estarían a medio camino entre DERIVE y MATLAB. Su aprendizaje es algo más complicado que el de DERIVE, aunque a cambio su potencia es bastante mayor.

Estas notas están divididas en cuatro partes fundamentales: dos de ellas son breves introducciones a los programas, la de MATLAB es algo más extensa debido a la complejidad, y posiblemente novedad, del programa. Las otras dos partes son prácticas, que tienen la intención de ilustrar cuestiones relacionadas con la utilización de cada programa y, de paso, actualizar algún conocimiento de Matemáticas. Por último, se han incluido otros dos apartados, uno que pretende ser una reflexión sobre la utilización del ordenador en la enseñanza de las Matemáticas y el otro es una pequeña lista de direcciones de Internet en las que se pueden encontrar materiales relacionados con el contenido de las notas.

1. Sobre DERIVE

DERIVE es un asistente matemático que permite realizar cálculo simbólico, cálculos numéricos (con ciertas limitaciones), gráficas en dos y tres dimensiones y en el que se pueden programar ciertas fórmulas aunque no se puede decir que sea un paquete ideal para hacer programas complejos. Vamos a ver, o recordar, a continuación algunas de las muchas posibilidades que nos ofrece para realizar después algunas prácticas concretas.

Las indicaciones que vienen a continuación están referidas a la versión 4.11, *DERIVE for Windows*, la versión en inglés. Existe ya alguna versión en castellano del programa, sin embargo, la mayoría de los textos de Matemáticas en los que se utiliza DERIVE aparecen referidos siempre a la versión en inglés.

1.1. Cuestiones Básicas

Supongamos que ya estamos dentro del programa DERIVE. Si hemos sido capaces de abrir el programa correctamente nos encontramos ante una ventana en la que, en la parte superior, aparece una línea de comandos: `File Edit Author Simplify Solve Calculus,` etc., dentro de cada uno de estos aparecen varias opciones y dentro de algunas de éstas algunas otras. Para activarlos hay que ponerse con el cursor del ratón sobre el nombre del comando elegido y pulsar una vez, aparecerá una nueva lista de opciones y volvemos a hacer lo mismo. Así, si queremos ejecutar el comando `File Load Demo`, tendremos que abrir `File`, seleccionar la opción `Load` y aquí volver a seleccionar `Demo`.

Veamos en primer lugar algunas cuestiones básicas relacionadas con el funcionamiento del programa:

- *Cómo salir del programa DERIVE*: Basta con pulsar `File Exit`. Nos preguntará si queremos guardar las expresiones que hayamos escrito. Si decimos que No, todas las operaciones previas se perderán.

- *Cómo introducir datos y funciones*: Para esto se utiliza el comando `Author Expression`. En la pantalla aparecerá una nueva ventana donde podemos introducir la fórmula, comando o datos que necesitemos. Para escribir símbolos o letras especiales hay una lista en la misma ventana. Basta con pulsar el carácter deseado con el cursor del ratón.

La ventana de `Author` también se puede activar pulsando las teclas `Ctrl A` del teclado simultáneamente. Una vez que hayamos acabado, pulsamos OK, y aparecerá la fórmula o expresión que hayamos puesto numerada.

Estos números son útiles para evitar tener que repetir fórmulas. Si en una nueva expresión queremos usar o sustituir alguna expresión anterior numerada, en la ventana de `Author` escribimos en el lugar que deseemos poner la fórmula, `#número de la expresión`.

- *Cómo se pueden borrar todas las expresiones de la pantalla*: Con la opción `Edit Remove`.
- *Cómo se pueden grabar en un fichero las operaciones de una sesión*: Esto es algo muy importante, si queremos aprovechar el trabajo que hemos realizado para empezar a partir de él en la próxima sesión. Los ficheros se guardan como en cualquier otro programa de Windows, con `File Save` o con `File Save as`. El fichero se guardará con extensión `.mth` que es la extensión que entiende Derive. Sin embargo, **no hay manera de guardar las gráficas**.
- *Cómo recuperar un archivo que se hubiera grabado previamente*: Para recuperar algo que se grabó en una sesión anterior, también se hace como en cualquier programa de Windows, con `File Open`. A veces los archivos no se recuperan de la forma que esperamos, en particular si el fichero contiene resultados numéricos, al guardar y volver a cargar el fichero esos resultados vuelven a aparecer en forma racional (!).

  En la parte de debajo de la línea superior de comandos aparecen una serie de iconos con los que se pueden realizar casi todas las operaciones que se pueden llevar a cabo con las opciones de los comandos.

En Derive la manera de trabajar consiste en dar los siguientes pasos: En primer lugar se introduce una cierta expresión matemática y a continuación se ejecuta un comando sobre esa expresión. Los comandos pueden ser de dos tipos: Comandos que se pueden ejecutar directamente o comandos que estén contenidos en un fichero de utilidades.

Vamos a ver en primer lugar algunas de las cosas que se pueden hacer con los comandos incorporados al propio programa, después comentaremos qué son los ficheros de utilidades y veremos algunos ejemplos del uso de algunos de ellos en las práticas que se proponen a continuación.

1.2. Números y Ecuaciones

Las expresiones matemáticas se introducen en Derive usando funciones, operadores y constantes. Los operadores matemáticos básicos son `(+, -, *, /)`, que junto con los paréntesis forman la base de la construcción de expresiones. Para la potenciación se usa `^`.

El resultado de un cálculo que sea un número se ofrece siempre, por defecto, en forma algebraica. Sin embargo, se puede convertir a un resultado con decimales.

Ejemplo 1 *Queremos calcular π^2, con 10 decimales exactos.*

Escribimos `Author: pi^2`

Elegimos la opción `Simplify Approximate` *y nos aparece una ventana en la que podemos modificar el número de decimales, que es 6 por defecto. En la ventana aparecen la opción OK que indica la operación, pero no la ejecuta, habría que volver a pulsar* `Simplify` *y* `Simplify` *que nos da el resultado definitivo.*

Ejercicio 1 *Calcular el valor de $e^2\sqrt{3}$, con 8 cifras decimales exactas. (Para escribir la expresión,* `Author: #e^2sqrt(3)`*).*

Para resolver ecuaciones se utiliza el comando `Solve`.

Ejemplo 2 *Queremos resolver ecuación, $x^2 - 5x + 6 = 0$.*

La introducimos en `Author:x^2-5x+6=0` *y seleccionamos la opción* `Solve Algebraically` *después de pulsar OK y el icono que contiene =, tendremos,*

`[x = 2, x = 3]`

También se pueden encontrar soluciones aproximadas de ecuaciones que no tengan solución exacta,

Ejemplo 3 *Por ejemplo, la ecuación $x\,\mathrm{sen}(x) = 1$ tiene una raíz entre 0 y 2. Escribimos* `Author: xsin(x)=1` *y pulsamos* `Solve Numerically`*, aparece una ventana en la que tenemos que especificar el intervalo en el que creemos que está la solución, en este caso, $[0, 2]$. El resultado,*

`[x=1.11416]`

También se pueden resolver sistemas de Ecuaciones lineales y no lineales (siempre que se puedan resolver de una manera más o menos sencilla).

Ejemplo 4 *Queremos resolver el sistema,*

$$\begin{cases} 2a^2x + 3y = 7 \\ x - 5y = 0 \end{cases}$$

Se puede pulsar `Solve System` *e introducir los datos en la ventana, o bien escribir directamente,*

```
Author:SOLVE([2a^2x+3y=7,x-5y=0],[x,y])
```

Obtendremos como solución,

$1\# : \left[\mathtt{x} = \frac{35}{10\mathtt{a}^2 + 3},\ \mathtt{y} = \frac{7}{10\mathtt{a}^2 + 3}\right]$

1.3. Vectores y Matrices

Para introducir un vector en Derive, se escribe entre corchetes y sus coordenadas separadas por comas,

```
Author: [1,2,3,4]
```

Para escribir una matriz, se hace fila a fila, cada fila en forma de vector, separamos las filas por comas y lo encerramos todo entre paréntesis,

```
Author: [[1,2,3],[4,5,6],[7,8,9]]
```

Estas dos operaciones también se pueden hacer directamente con las opciones `Author Vector` y `Author Matrix` respectivamente. Aparecen ventanas donde, habiendo especificado previamente el tamaño del vector o de la matriz, no hay más que escribir los números.

Vamos a ir viendo algunas operaciones que se pueden llevar a cabo con vectores y matrices:

- Las *operaciones* de suma de vectores, suma de matrices, producto de matrices, producto de un escalar por un vector o una matriz, se hacen utilizando los operadores aritméticos básicos.
- La *traspuesta* de una matriz se hace con el acento grave, por ejemplo, `[[a, b, c], [1, 2, 3]]‘`.
- Para calcular el *determinante* de una matriz se utiliza el comando `det`. (En Derive, no se distingue entre mayúsculas y minúsculas). Por ejemplo, `DET([[2, 3], [a, b]])`, produciría, después de simplificar,

  ```
  2b - 3a
  ```

  Importante: A veces es necesario aplicarle a una expresión, por ejemplo una matriz, varios comandos sucesivamente. Para evitar tener que

volver a introducir la matriz, podemos identificarla con una letra que representará a la matriz y de esta forma evitar reescribirla cada vez que vayamos a realizar un cálculo, esto también es válido para cualquier otra expresión, aunque no sea una matriz. Se hace,

`Author: a:=[matriz]`

A partir de este momento y durante la misma sesión `a` representa a la matriz a la que se ha identificado. Se puede usar cualquier letra, pero hay que tener precaución, evidentemente, de no identificar con la misma letra expresiones distintas.

- Con el operador `^` se puede elevar una matriz cuadrada a cualquier exponente entero.
- Para calcular la *inversa* de una matriz también se utiliza el operador anterior. Si `a` es la matriz, escribir `Author: a^(-1)`, pulsamos OK y después simplificamos.
- Para encontrar la forma escalonada reducida por filas de una matriz se utiliza el comando `ROW_REDUCE`. (Este comando es similar a uno que aparecerá en Matlab `rref` y que estudiaremos en profundidad en la primera prática de Matlab).
- El *polinomio característico* de una matriz cuadrada es el determinante de la diferencia entre la matriz y la matriz identidad multiplicada por una variable. Si ponemos `Author: CHARPOLY(A,` λ`)` nos aparecerá el polinomio característico de la matriz `A` en función de λ.
- Los `autovalores` de una matriz cuadrada son los ceros de su polinomio característico, esto se puede calcular directamente con `EIGENVALUES(A,` λ`)`.

1.4. Cálculo Diferencial e Integral

- `Calculus Limit`

  Este es el comando que se utiliza para calcular límites. Una vez que está escrita la expresión, pulsando el comando aparece una ventana en la que hay que especificar la variable, el punto hacia donde tiende (que puede ser infinito, `inf`) y si es por la derecha, por la izquierda o ambos.

Si se pulsa OK aparecerá el límite indicado en la pantalla y ya sólo queda simplificar, con el icono `=`, por ejemplo.

- `Calculus Differentiate`

  Se utiliza para calcular derivadas de funciones, en forma simbólica, es decir, la función derivada. Pulsándolo aparece una nueva ventana en la que hay que especificar, la variable con respecto de la cual se quiere derivar y el orden de la derivada que se quiere calcular. Lo mismo se puede conseguir escribiendo en `Author:` directamente,

  `DIF(LN(COS(x)),x,2)`

  es decir, derivada segunda de la función `LN(COS(x))`, con respecto de x.

- `Calculus Taylor Series`

  Sirve para calcular polinomios de Taylor, en una de las prácticas veremos cómo se utiliza.

- `Calculus Integrate`

  Es el comando para calcular Integrales. En la ventana que aparece hay que especificar, la variable, si la integral es definida o indefinida y, en el primer caso, los límites de integración, que pueden ser infinito. Es decir, que también se pueden calcular integrales impropias.

- `Calculus Sum`

  Sirve para calcular sumatorios, también será analizado en una de las prácticas.

1.5. Ficheros de Utilidades

Los ficheros de utililades son unos ficheros con extensión .MTH que vienen incorporados con el programa en los que están definidas o programadas ciertas fórmulas. Se pueden cargar para utilizar las fórmulas contenidas en ellos de dos formas diferentes:

Eligiendo la secuencia `File Load Math` aparecerá una ventana en la que aparecen todos los ficheros disponibles. Basta con elegir con el ratón el que queremos que se cargue. Y aparecerán todas las fórmulas en pantalla. Esta

posibilidad es la recomendada cuando no se conoce bien el contenido del fichero, o la sintaxis de la fórmula que queremos utilizar.

Sin embargo, si elegimos la secuencia `File Load Utility`, una vez seleccionado el fichero se cargarán todas las fórmulas y se podrán utilizar, pero no aparecen en pantalla. esta opción es la recomendada cuando vamos a utilizar alguna fórmula del fichero que conocemos perfectamente.

Dependiendo de la versión de DERIVE que estemos utilizando podemos disponer de más o menos ficheros de utilidades, además de los que el propio usuario vaya generando. Por ejemplo, en la versión que se ha utilizado para preparar estas notas, la 4.11 aparecen los siguientes ficheros:

`SOLVE.MTH` - Sirve para resolver sitemas de ecuaciones algebráicas no lineales.

`VECTOR.MTH` - Contiene funciones que automatizan algunas operaciones que se realizan habitualmente con vectores y matrices.

`NUMERIC.MTH` - Sirve para hacer derivación e integración numérica.

`DIF_APPS.MTH` - Contiene algunas aplicaciones de las derivadas, tales como, derivación implícita, cálculo de rectas tangentes, planos tangentes, etc.

`INT_APPS.MTH` - Aplicaciones de la Integración. Transformadas de Laplace, Series de Fourier, cálculo de longitudes de arco, etc.

`ODE1.MTH` - Contiene comandos para resolver Ecuaciones Diferenciales Ordinarias de Primer Orden, soluciones de forma exacta, cuando es posible.

`ODE2.MTH` - Lo mismo que el fichero mencionado antes, pero para Ecuaciones Diferenciales Ordinarias de Segundo Orden.

`ODE_APPR.MTH` - Contiene comandos para resolver ecuaciones diferenciales de forma aproximada. Métodos numéricos como el Método de Euler, Runge-Kutta, Taylor, etc. y Métodos geométricos para dibujar campos de direcciones de la ecuación.

`RECUREQN.MTH` - Contiene funciones que sirven para calcular soluciones exactas de ecuaciones en diferencias de primer y segundo orden.

`APPROX.MTH` - Contiene sólo una función que sirve para calcular la aproximación racional Pade.

`EXP_INT.MTH` - Contiene fórmulas que permiten aproximar las integrales exponencial, logarítmica, seno y coseno.

`PROBABIL.MTH` - Fórmulas de Probabilidad. Distribución Binomial, Hipergeométrica, t de Student, χ^2, etc.

`FRESNEL.MTH` - Integrales de Fresnel.

`BESSEL.MTH` - Funciones de Bessel y de Airy.
`HYPERGEO.MTH` - Funciones Hipergeométricas.
`ELLIPTIC.MTH` - Fórmulas para calcular Integrales Elíticas.
`ORTH_POL.MTH` - Contiene fórmulas para hacer cálculos relativos a polinomios ortogonales.
`ZETA.MTH` - Contiene varias fórmulas que permiten aproximar funciones relacionadas con la función Zeta.
`GRAPHICS.MTH` - Contiene comandos que permiten dibujar gráficas de objetos como esferas, toros, conos, etc., como proyecciones en la pantalla 2D.
`NUMBER.MTH` - Comandos de Teoría de Números.
`MISC.MTH` - Contiene una mezcla de fórmulas de todo tipo.

Demos

Además de los ficheros de utilidades DERIVE lleva también unos ficheros con demostraciones de algunas de sus capacidades básicas. Para activar estas demostraciones hay que utilizar la secuencia `File Load Demo`, y después elegir uno de los ficheros de entre los siguientes:

`ARITH.DMO` - Aritmética elemental.
`ALGEBRA.DMO` - Ejemplos de cálculos algebraicos.
`TRIG.DMO` - Demo sobre trigonometría.
`FUNCTION.DMO` - Sobre construcción de funciones.
`CALCULUS.DMO` - Demo sobre Cálculo Diferencial e Integral.
`MATRIX.DMO` - Demo sobre matrices y vectores.

2. Prácticas con DERIVE

2.1. Práctica Primera: Gráficos con DERIVE

2.1.1. Gráficas 2D.

El programa DERIVE es capaz de dibujar gráficas de curvas y funciones en el plano de tres formas diferentes. Funciones o curvas en coordenadas cartesianas, dadas en forma explícita, es decir, de la forma $y = f(x)$; o también dadas en forma implícita, es decir, de la forma $g(x, y) = 0$. También curvas en forma paramétrica, es decir, de la forma $\vec{r}(t) = (x(t), y(t))$ con $a \leq t \leq b$. Y también curvas en coordenadas polares, de la forma $r = r(\theta)$ con $\theta_1 \leq \theta \leq \theta_2$.

Comandos Básicos de la pantalla de gráficos 2D

Para pasar a la pantalla de gráficos en 2 dimensiones del programa DERIVE, hay que pulsar la opción `window 2D-plot`, aparecerá superpuesta la pantalla 2D. Para volver a la ventana de expresiones, en la nueva pulsar la opción `window algebra`. También se puede hacer con iconos que aparecen debajo de la línea de comandos. Recorriendo con el ratón los iconos descubriremos fácilmente cuáles son. En la línea superior de nuevos comandos que aparece en la ventana 2D, algunas de las opciones más importantes son:

`Plot`: Esta es la opción que ejecuta la gráfica.

`Edit Delete Plot`: Permite borrar las gráficas, una o todas.

`Set Center`: Centra la imagen tomando como centro el punto que indiquemos.

`Set Range`: Permite cambiar el rango en el que aparece la gráfica.

`Options Coordinate System`: Con esta opción se puede cambiar el sistema de coordenadas, rectangulares, polares.

La mayoría de estas opciones vuelven a aparecer en forma de iconos debajo de la línea de comandos, junto con algunos otros, en particular son interesantes los iconos de las "flechitas"que permiten llevar a cabo diversas modalidades de Zoom sobre la gráfica.

Coordenadas Cartesianas

Ejemplo 1 *Dibujar la gráfica de la función, dada en forma explícita*

$$y = x^2$$

Introducimos la función, que se puede hacer de dos formas, poniendo: `Author: y=x^2`*, o sin poner la* y*.*

Pulsamos `window 2D-plot` *y a continuación, en la ventana 2D,* `Plot` *y ya tenemos la gráfica, que aparece en un rango* $-4 < x < 4$*,* $-4 < y < 4$*, por defecto.*

Ejercicio 1 *Dibujar las gráficas de las siguientes funciones dadas en forma explícita eligiendo, si fuera necesario, un rango adecuado para que aparezcan los aspectos más representativos de la función:*

a) $f(x) = x(x^2+4)^2$

b) $f(x) = x - \sqrt{x}$

c) $f(x) = \dfrac{\log x}{x}$

d) $f(x) = \dfrac{x(x-2)}{(x+1)(x-2)}$

e) $f(x) = \operatorname{sen}\left(\dfrac{1}{x}\right)$

p) $f(x) = \dfrac{x}{e^{|x-1|}}$

La función valor absoluto se puede escribir con `abs()`

Ejemplo 2 *Dibujar la curva, dada en forma implícita, siguiente:*

$$x^2 + y^2 = 1$$

Introducimos la ecuación en `Author: x^2+y^2=1`*.*

Y seguimos los mismos pasos que en el ejemplo anterior, es decir, que el programa entiende perfectamente la fórmula sin necesidad de despejar y*. Nos debe salir una circunferencia de radio 1 centrada en el origen. Sin embargo, lo que se aprecia en el dibujo no tiene la pinta de una circunferencia, esto es debido al rango que aparece por defecto.*

Ejercicio 2 *Dibujar las siguientes curvas dadas en forma implícita:*

a) $\dfrac{x^2}{25} + \dfrac{y^2}{9} = 1$

b) $\dfrac{x^2}{25} - \dfrac{y^2}{9} = 1$

c) $x^2 - y^2 = 0$

d) $x^2y^2 = 1$

Ecuaciones Paramétricas

Ejemplo 3 *Dibujar la gráfica de la curva*

$$\vec{r}(t) = \left(\frac{t(t^2-1)}{t^2+1}, \frac{2(t^2-1)}{t^2+1}\right); \qquad -5 \leq t \leq 5$$

Escribimos en `Author` *el vector entre corchetes:* `[(t(t^2-1))/(t^2+1),(2(t^2-1))/(t^2+1)]`. *Pasamos a la ventana 2D y al pulsar* `Plot` *aparece una ventana en la que tenemos que indicar el intervalo de variación de* t, *en este caso,* `Minimum value: -5 Maximum value: 5`. *Pulsamos OK y sólo queda elegir, si fuera necesario, un Rango adecuado para la función.*

Ejercicio 3 *Dibujar las curvas en paramétricas siguientes:*

a) $\vec{r}(t) = (2\cos^3 t, 2\,\mathrm{sen}^3 t); \qquad -\pi \leq t \leq \pi$

b) $\vec{r}(t) = (3\,\mathrm{sen}\, t, 2\,\mathrm{sen}(2t)); \qquad -\pi \leq t \leq \pi$

c) $\vec{r}(t) = \left(\frac{t}{\pi}\left(12(\frac{t}{\pi})^2 - 9\right), \left((\frac{t}{\pi})^2 - 1)16(\frac{t}{\pi})^2 + 2\right)\right); \qquad -3 \leq t \leq 3$

d) $\vec{r}(t) = \left(\frac{3}{2}\cos t(\cos t + 1), 2\,\mathrm{sen}(2t)\right); \qquad -\pi \leq t \leq \pi$

e) $\vec{r}(t) = (\mathrm{sen}(2t) + \mathrm{sen}\, t, -\cos(2t) - \cos t); \qquad -\pi \leq t \leq \pi$

f) $\vec{r}(t) = \left(e^{\frac{t}{4}}\,\mathrm{sen}(2t), e^{\frac{t}{4}}\cos(2t)\right); \qquad -\pi \leq t \leq \pi$

g) $\vec{r}(t) = \left(\frac{2}{3}t\cos(\frac{7t}{2}), \frac{2}{3}t\,\mathrm{sen}(\frac{7}{t})\right); \qquad -\pi \leq t \leq \pi$

h) $\vec{r}(t) = \left(t - \frac{11}{10}\,\mathrm{sen}(3t), -\frac{22}{10}\cos(3t)\right); \qquad -3\pi \leq t \leq 3\pi$

Coordenadas Polares

Ejemplo 4 *Dibujar la gráfica de*

$$r = 2 - 4\cos(\theta), \qquad -\pi \leq \theta \leq \pi$$

Escribimos la función en `Author: r:=1-cos(theta)`. *(También podemos utilizar otra variable, p por ejemplo)*

Pasamos a la ventana 2D y antes de pulsar `Plot`*, con la opción* `Options Coordinate System...` *lo ponemos en coordenadas polares.*

Pulsamos `Plot` *e indicamos el intervalo de variación de la variable, en este caso se pueden dejar los que aparecen por defecto, que son precisamente* $-\pi$ *y* π*. Por último, ajustamos con los iconos de* `Zoom` *hasta ver la gráfica completa.*

Ejercicio 4 *Dibujar las gráficas de las siguientes funciones, dadas en coordenadas polares:*

a)$r = 7 - 7\,\mathrm{sen}(\theta); \qquad -\pi \leq \theta \leq \pi$

b)$r = 3 - 6\,\mathrm{sin}(\theta); \qquad -\pi \leq \theta \leq \pi$

c)$r = sen(6\theta); \qquad -\pi \leq \theta \leq \pi$

d)$r = \cos(8\theta); \qquad -\pi \leq \theta \leq \pi$

e)$r = \sqrt{5\cos(2\theta)}; \qquad -\pi \leq \theta \leq \pi$

2.1.2. Gráficas 3D

Con DERIVE se pueden hacer gráficas de funciones de dos variables en el espacio, es decir, funciones de la forma $z = f(x, y)$ (forma explícita). Sin embargo, utilizando el fichero de utilidades `GRAPHICS.MTH`, también se pueden dibujar curvas y ciertas superficies (que no sean funciones) en la pantalla de gráficos 2D. Para pasar a la pantalla de gráficos 3D, una vez que se ha escrito la expresión que se quiere representar, se pulsa el icono que hay más a la derecha y entramos en una nueva pantalla de la que sus comandos más importantes son:

`Plot`: El comando que hace que se dibuje la gráfica.

(Aquí no existe un comando para borrar gráficas, ya que cuando se dibuja una nueva desaparece la anterior.)

`Window algebra`: Que permite volver a la ventana de expresiones.

`Set Grids`: Que permite modificar el número de puntos que aparecen en el mallado de la gráfica. Si se pone un mallado muy denso, puede tardar mucho en hacer la gráfica o incluso, en algunos casos, bloquear el ordenador.

De manera análoga a como ocurre con la ventada de gráficos 2D los comandos se pueden ejecutar con los iconos, entre los que se encuentran los correspondientes al `Zoom`.

Curvas en el Espacio

En primer lugar, cargamos el fichero `GRAPHICS.MTH`. Con `File Load`. Escribimos `Author: axes`.

Entramos en la pantalla de gráficos 2D. Quitamos los ejes 2D con la opción `Options axes` y pulsamos `Plot` para que aparezcan los tres nuevos ejes (isométricos). Habrá que especificar las longitudes de cada uno de los tres ejes. Por ejemplo, `Min: 0  Max: 3.14159` (tres veces).

Ahora estamos en condiciones de dibujar gráficas de curvas en forma paramétrica.

Realmente lo que se hace es proyectar las curvas que están en el espacio sobre el plano 2D. Esto lo produce un comando definido en el fichero que hemos cargado, `isometric`.

Ejemplo 5 *Dibujar la curva*

$$\vec{r}(t) = (\cos(t), \operatorname{sen}(t), \frac{t}{48}) \qquad -50 \leq t \leq 50$$

y rotarla después un ángulo de $\frac{\pi}{2}$ *radianes en torno al eje de las* X*.*

Para dibujar la curva, escribimos en

`Author: isometric([cost,sint,t/48])` *y pulsamos* `Simplify`*. (La función* `isometric` *es la que nos permite proyectar la gráfica sobre el plano 2D). Quizá sea conveniente definir la función previamente con* `r:=.....`*, sobre todo si la vamos a rotar después.*

Vamos a la pantalla de gráficos y pulsamos `Plot` *y nos preguntará por el intervalo, especificamos* `=Min: -50 Max: 50`*. (Si todo ha ido bien saldrá una hélice a lo largo del eje* Z*).*

Para rotarla bastará con escribir:

`Author:isometric(rotate_x(pi/2)*[cost,sint,t/48])`*, y proceder como antes. Ahora aparecerá la misma hélice, pero a lo largo del eje* Y*.*

Ejercicio 5 *Representar las curvas siguientes y después rotarlas alrededor del eje que se indica, con el ángulo que se indica:*

a) $\vec{r}(t) = (2\cos^3 t, 2\operatorname{sen}^3 t, t) \qquad -4 \leq t \leq 3$*. Eje* X*.* $\frac{3\pi}{4}$ *radianes.*

b) $\vec{r}(t) = (\cos t, 2\cos^2 t, \frac{1}{4}\operatorname{sen} t) \qquad -\pi \leq t \leq \pi$*. Eje* Z*.* $-\frac{3\pi}{2}$ *radianes.*

c) $\vec{r}(t) = (\frac{t}{6}\cos t, \frac{t}{6}\operatorname{sen} t, \frac{t}{36})$ $\quad -12 \leq t \leq 19$. *Eje* Y. 180^0.

d) $\vec{r}(t) = (e^{\frac{t}{4}}\operatorname{sen}(2t), e^{\frac{t}{4}}\cos(2t), \frac{t}{4})$ $\quad -10 \leq t \leq 4{,}8$. *Eje* X. 30^0.

e) $\vec{r}(t) = (\operatorname{sen}(2t) + \operatorname{sen}(t), -\cos(2t) - \cos(t), \frac{t}{6})$ $\quad -9 \leq t \leq 10$. *Eje* Y. $-\frac{\pi}{2}$ *radianes.*

f) $\vec{r}(t) = (\cos(3t), 2\cos^2(t), \operatorname{sen}(2t))$ $\quad -\pi \leq t \leq \pi$. *Eje* Z. π *radianes.*

Superficies Parametrizadas

Ejemplo 6 *Representar la superficie esférica de radio 1.*

Escribimos en `Author`:

```
isometrics(sphere(1,theta,phi),theta,-pi,pi,12,phi,0,pi,12)
```

(Cuidado, ahora hay que escribir `isometrics`*. Las variables pueden ser otras. El 12 puede variar, para dar más o menos detalle al dibujo)*

A continuación pulsar `Simplify approximate`*, nos aparecerán unas matrices de números con las coordenadas de los puntos que se van a representar.*

Pulsamos `Plot` *y aparecerá el dibujo. Si sólo aparecen puntos, habrá que cambiar el modo con:*

`Options Points` *y especificamos* `Connect: Yes`*. Volvemos a pulsar* `Plot` *y ahora se conectan los puntos.*

Ejercicio 6 *Representar las superficies siguientes: (En todas ellas utilizar 12 puntos para cada variable). En algunos ejemplos es posible que sea necesario hacer* `Zoom` *o ajustar el Rango para visualizarlos correctamente. Para cada caso hay que buscar el comando adecuado en el fichero de utilidades.*

a) El Hemisferio Norte de una esfera de radio 1.

b) La parte de la esfera de radio 1 situada en el primer octante.

c) El Hemisferio Sur de una esfera de radio 1.

d) Un cilindro de radio 2, con $0 \leq z \leq 2$.

e) Un Toro con $r_c = 2$, $r_s = 0{,}7$.

Gráficas de funciones $\mathbf{z = f(x, y)}$

Ejemplo 7 *Representar la gráfica de la función*

$$z = x^2 + y^2$$

Dibujando algunas curvas de nivel y alguna traza (curvas que son intersección de la gráfica de la función con los planos coordenados).

Introducimos la función `Author: z=x^2+y^2`. *Pulsamos* `Plot` *en la ventana gráfica 3D y nos aparece una primera versión de la gráfica.*

Para dibujar las curvas de nivel y la de las trazas: Volvemos a la ventana de expresiones y pulsamos la opción `Simplify Substitute for Variables` *y aquí indicamos los valores a sustituir por las variables. Por último, tendríamos que representar esta gráfica en la pantalla 2D.*

Ejercicio 7 *Representar las gráficas de las siguientes funciones de 2 variables, dibujando en cada caso, al menos 4 curvas de nivel diferentes y una traza:*

a) $z = \dfrac{1}{9 + x^2 + y^2}$

b) $z = -\sqrt{|xy|}$

c) $z = \dfrac{\cos(\frac{x^2+y^2}{4})}{3 + x^2 + y^2}$

d) $z = \dfrac{y^2}{5} - 3|x|$

e) $z = e^{-(x^2+y^2)}$

2.2. Práctica Segunda: Series de Taylor y de Fourier.

Se llama polinomio de *Taylor* de la función $f(x)$ desarrollado en torno a $x = a$, de grado n a la expresión:

$$P_{n,a}(x) = f(a)+f'(a)(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+\ldots+\frac{f^{(n}(a)}{n!}(x-a)^n$$

Para que se pueda calcular el polinomio de Taylor de una función en un punto, esta función debe poseer derivadas, al menos, hasta el orden n. En el caso en que $a = 0$ se suele llamar polinomio de *McLaurin*, y quedaría reducido a,

$$P_{n,a}(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \ldots + \frac{f^{(n}(0)}{n!}x^n$$

La serie de *Fourier* asociada a una función $f : [-T, T] \longrightarrow \mathbb{R}$ que sea continua a trozos, es

$$f(x) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty}\left[a_n \cos\left(\frac{n\pi x}{T}\right) + b_n \operatorname{sen}\left(\frac{n\pi x}{T}\right)\right]$$

donde los coeficientes de Fourier se calculan mediante las fórmulas:

$$a_0 = \frac{1}{T}\int_{-T}^{T} f(x)dx$$

$$a_n = \frac{1}{T}\int_{-T}^{T} f(x)\cos\left(\frac{n\pi x}{T}\right)dx, \qquad n = 1, 2, 3, \ldots$$

$$b_n = \frac{1}{T}\int_{-T}^{T} f(x)\operatorname{sen}\left(\frac{n\pi x}{T}\right)dx, \qquad n = 1, 2, 3, \ldots$$

Con Derive, o con cualquier otro programa, lo que se puede calcular o representar realmente es una suma parcial de la serie de Fourier, esto es

$$s_k(x) = \frac{a_0}{2} + \sum_{n=1}^{k}\left[a_n \cos\left(\frac{n\pi x}{T}\right) + b_n \operatorname{sen}\left(\frac{n\pi x}{T}\right)\right]$$

Para calcular el polinomio de Taylor con DERIVE, se utiliza el comando `TAYLOR(f(x),x,a,n)`. Esto también se puede hacer directamente con los comandos de la parte superior de la pantalla, una vez que hemos introducido la fórmula de la función en `Author`, pulsamos la secuencia `Calculus Taylor`

`Series...`, aparece una nueva ventana en la que podemos especificar el punto en torno al cual hacemos el desarrollo, `Expansion Point:` y el grado del polinomio, `Order:`.

Para calcular una suma parcial de la serie de Fourier de una función $f(x)$ hasta el grado k, $s_k(x)$, se utiliza el comando `FOURIER(f(x),x,-T,T,k)` (utilizando la notación anterior), que se encuentra en el fichero de utilidades `INT_APPS.MTH`. Así que habrá que cargar el fichero previamente para poder utilizar el comando.

2.2.1. Polinomios de Taylor

Ejercicio 1 *Sea la función* $f(x) = e^x$*:*

a) Calcular los polinomios de McLaurin de grados n=2,4,10 y representarlos junto a la función.

b) Calcular los polinomios de Taylor $P_{n,1}$ *para los grados n=2,4,10 y representarlos junto a la función.*

Ejercicio 2 *Sea la función* $f(x) = \text{sen}(x)$*:*

a) Calcular los polinomios de McLaurin de grados $n = 3, 5, 11$ *y representarlos junto a la función.*

b) Calcular los polinomios de Taylor $P_{n,\frac{\pi}{2}}(x)$ *para los grados* $n = 3, 5, 11$ *y representarlos junto a la función.*

c) Utilizar el polinomio de McLaurin con $n = 3$ *y el polinomio de Taylor* $P_{3,\frac{\pi}{2}}(x)$ *para calcular aproximadamente en valor de* sen(0,1) *y el valor de* sen(1,5)*. Compararlo con los valores reales. Para sustituir en los polinomios, una vez calculados, pulsar la secuencia* `Simplify Substitute for Variables`*, aparecerá una ventana en la que indicamos el valor de la* x*. ¿En qué casos nos sale una mejor aproximación? ¿Por qué?*

2.2.2. Series de Fourier

Ejercicio 3 Convergencia

Sea la función,

$$f(x) = \begin{cases} 0, & si\ -\pi < x < 0 \\ x, & si\ 0 < x < \pi \end{cases}$$

a) Calcular las sumas parciales de la Serie de Fourier asociada a la función hasta $k = 2, 3, 4, 6, 10, 20$*.*

b) Representarlas, con la gráfica de la función.

c) ¿Qué valor toman las sumas parciales en el punto de continuidad $x = \frac{\pi}{2}$*, a qué número deberían aproximarse?*

c) ¿Qué valor toman las sumas parciales en el punto de discontinuidad $x = \pi$*, a qué número deberían aproximarse?*

Indicación: En primer lugar habrá que escribir la función $f(x)$*. Para las funciones definidas a trozos Derive dispone de una función que hace fácil su sintaxis. Se trata de la función* `chi(a,x,b)`*, que toma el valor 1 en el intervalo* (a, b) *y vale 0 en el resto. (a y b pueden ser infinito,* `inf`*). En este ejemplo, habría que poner* `Author:0*chi(-pi,x,0)+x*chi(0,x,pi)`*.*

Ejercicio 4 Fenómeno de Gibbs

Sea la función,

$$f(x) = \begin{cases} -1, & si\ -\pi < x < 0 \\ 1, & si\ 0 < x < \pi \end{cases}$$

a) Calcular las sumas parciales de su Serie de Fourier asociada hasta $k = 2, 4, 10$*.*

b) Representarlas, con la gráfica de la función.

Se puede observar que en los puntos cercanos a la discontinuidad de la función $f(x)$ *la Serie de Fourier (sus sumas parciales, que es lo que realmente estamos dibujando) pueden exceder hasta en un 9 % el valor del salto de la discontinuidad. A este hecho se le conoce con el nombre de* Fenómeno de Gibbs. *Para observarlo mejor se puede agrandar la imagen en torno a los puntos de coordenadas* $(0, 1)$ *y* $(\pi, 1)$*. Esto se consigue pulsando el icono que hay en la pantalla de gráficos 2D con la leyenda* `Set range with box`*, después se pone el cursor en un punto de la pantalla y pulsando el botón izquierdo del ratón se marca la zona a ampliar.*

Ejercicio 5 Método de Sumación de Fejér

El Método de Sumación de Fejér de una serie de Fourier consiste en sumar los promedios de las sumas parciales y pasar al límite. Se obtiene así una nueva sucesión de funciones que también converge a la función inicial incluso mejor que la propia Serie de Fourier (Teorema de Fejér, ver [A]), de hecho no se produce el Fenómeno de Gibbs mencionado anteriormente.

En otras palabras, si tenemos las sumas parciales de la Serie de Fourier,

$$s_k(x) = \frac{a_0}{2} + \sum_{n=1}^{k} \left[a_n \cos\left(\frac{n\pi x}{T}\right) + b_n \operatorname{sen}\left(\frac{n\pi x}{T}\right) \right]$$

Construimos una nueva sucesión con el promedio de las sumas parciales,

$$\sigma_m(x) = \frac{1}{m+1}\sum_{k=0}^{m} s_k(x)$$

La Suma de Fejér es el,

$$s(x) = \lim_{m\to\infty} \sigma_m(x)$$

Sea la función del ejercicio anterior,

$$f(x) = \begin{cases} -1, & si\ -\pi < x < 0 \\ 1, & si\ 0 < x < \pi \end{cases}$$

Dibujar las sumas parciales de Fejér $\sigma_2(x)$, $\sigma_5(x)$, $\sigma_{10}(x)$, $\sigma_{15}(x)$ y $\sigma_{30}(x)$. Observar en la gráfica que la convergencia de las sumas parciales a la función $f(x)$ es completamente distinta a la de la Serie de Fourier y no hay fenómeno de Gibbs.

Para hacerlo es conveniente seguir los siguientes pasos:

1) Definimos la función $f(x)$,

$1\#:\ \mathtt{f} := (-1).\mathtt{CHI}(-\pi, \mathtt{x}, 0) + 1.\mathtt{CHI}(0, \mathtt{x}, \pi)$

2) Definimos la suma parcial de Fourier,

$2\#:\ \mathtt{S}(\mathtt{x}, \mathtt{k}) := \mathtt{FOURIER}(\mathtt{f}, \mathtt{x}, -\pi, \pi, \mathtt{k})$

3) Y, por último, definimos la suma parcial de Fejér,

$$3\#:\ \sigma(\mathtt{x}, \mathtt{m}) := \frac{1}{\mathtt{m}+1}\sum_{\mathtt{k}=0}^{\mathtt{m}} \mathtt{S}(\mathtt{x}, \mathtt{k})$$

Esta fórmula se puede conseguir escribiendo,

```
Author: sigma(x, m):= (1/(m+1))sum(S(x, k),k,0,m)
```

Ahora para calcular y dibujar una suma concreta, por ejemplo $\sigma_5(x)$, basta con escribir en `Author: sigma(x,5)`*, simplificar y dibujar.*

Ejercicio 6 *Sea la función,*

$$f(x) = |x|$$

Calcular las sumas parciales de la serie de Fourier asociada a $f(x)$ para $k = 5, 10, 15$. Representarlas junto a la función. Observar que, al tratarse de una función bastante regular, incluso con pocos términos, la sucesión de sumas parciales converge muy rápidamente a la función $f(x)$.

Utilizar los comandos creados en el ejercicio anterior para dibujar las sumas de Fejér para $m = 5, 10, 15$.

Indicación: *La función* $|x|$ *se puede escribir en Derive,* `abs(x)`.

Ejercicio 7 *En los siguientes casos calcular las sumas parciales de la Serie de Fourier para las funciones indicadas y el valor de* k *que se da. Representarlas después junto a la gráfica de la función* $f(x)$*:*

a) $f(x) = x^2$, $x \in (-\pi, \pi)$. $k = 10$.

b) $f(x) = \cos(3x)$, $x \in (-\pi, \pi)$. $k = 3$.

Ejercicio 8 *Sea la función,*

$$f(x) = \begin{cases} 0, & si\ -\pi < x < -\frac{\pi}{2} \\ -1, & si\ -\frac{\pi}{2} < x < 0 \\ 1, & si\ 0 < x < \frac{\pi}{2} \\ 0, & si\ \frac{\pi}{2} < x < \pi \end{cases}$$

Calcular las sumas parciales de Fourier y representarlas junto a la función para los casos $k = 4, 5, 10$.

Ejercicio 9 Función de Weierstrass

La función de Weierstrass es un ejemplo de una función continua en todos los puntos pero que no tiene derivada en ninguno. Un caso concreto de esta función sería,

$$f(x) = \sum_{n=1}^{\infty} 2^{-n} \cos(3^n x)$$

En cierto modo se puede considerar que se trata de una serie de Fourier. Vamos a dibujar la gráfica de algunas de las sumas parciales de esta función. Para ello resulta conveniente definir la expresión anterior mediante algún nombre que nos permita trabajar con ella de una manera cómoda, sin tener que arrastrar toda la expresión cada vez que queramos dibujar una de las sumas parciales.

Esto se puede conseguir de la forma,

`Author: WEIER(x, k) := sum(2^(-n)COS(3^nx), n, 1, k)`

O utilizando cualquier otro nombre que sea distinto de los comandos existentes en Derive. El resultado debería ser una expresión como la siguiente,

$1\#: \quad \mathtt{WEIER(x,k)} := \sum_{\mathtt{n}=1}^{\mathtt{k}} 2^{-\mathtt{n}} \cos(3^{\mathtt{n}}\mathtt{x})$

Si queremos calcular o representar ahora una suma parcial, por ejemplo,

$$\sum_{n=1}^{3} 2^{-n} \cos(3^n x)$$

Sólo tenemos que escribir `Author: weier(x,3)` *y pulsar* `Simplify` *o pasar directamente a la pantalla de gráficos 2D.*

Representar algunas de las sumas parciales, por ejemplo, los casos $k = 1,\ k = 5,\ k = 10$ *y* $k = 15$.

3. Sobre MATLAB

El nombre MATLAB es una abreviatura de las palabras MATrix LABoratory. MATLAB es un sistema interactivo para cálculos científicos y de ingeniería basado en las matrices. Con él se pueden resolver complejos problemas numéricos sin necesidad de escribir un programa específico para ello. Aunque también es posible programar. Además el programa MATLAB dispone, dependiendo de la versión, de diferentes módulos *(Toolboxes)* que permiten resolver problemas específicos. Nosotros nos vamos a concentrar en lo que se puede hacer con el módulo de cálculo simbólico *(Matlab Symbolic Toolbox)*. Aunque veremos en primer lugar algunas generalidades del programa.

3.1. Comandos Básicos

Supongamos que hemos sido capaces de abrir el programa. En Matlab las órdenes se introducen escribiéndolas una a una a continuación del *prompt* (`>>`) que aparece en la ventana del usuario. Veamos en primer lugar algunas de las operaciones matemáticas más elementales.

Para sumar dos números:

```
>>2+2
ans =
     4
```

Después de escribir cada comando hay que pulsar al Intro para que lo ejecute.

El valor que queremos calcular también se puede asignar a una variable. Por ejemplo:

```
x=3^2
x=
     9
```

La notación para las operaciones matemáticas elementales es la la habitual en todos los programas de Cálculo Simbólico:

suma	`+`
resta	`-`
división	`/`
exponenciación	`^`
multiplicación	`*`

También están definidas algunas las funciones más comunes utilizadas en Matemáticas. Su sintaxis coincide con la que se utiliza en el programa DERIVE, aunque hay algunas diferencias. Algunas de estas funciones son:

`sin`	seno
`sinh`	seno hiperbólico
`asin`	arcoseno
`cos`	coseno
`cosh`	coseno hiperbólico
`acos`	arcocoseno
`tan`	tangente
`atan`	arcotangente
`exp`	exponencial
`log`	logaritmo neperiano
`log10`	logaritmo decimal
`sqrt`	raíz cuadrada
`abs`	valor absoluto

De todas formas para obtener listas completas de todas las funciones que puede utilizar Matlab, así como para saber el uso de cada una de ellas o de cualquier comando siempre se puede acudir al `help`. Esto se puede hacer de varias formas, poniendo `>>helpwin`, es el propio programa quien nos ofrece la ayuda (como en cualquier otro programa), o poniendo `>>helpdesk`, que nos ofrece ayuda interactiva, conectándose a Internet si este recurso está disponible en nuestro ordenador.

3.2. Interfaz de usuario. Variables.

Con las flechas del cursor se pueden recuperar las órdenes anteriores, sin tener que volver a teclearlas. Esto resulta últil en caso de una equivocación o cuando se quiere repetir un comando con alguna pequeña modificación.

A veces, puede resultar necesario, hasta imprescindible que el resultado de un cálculo no aparezca en pantalla. Por ejemplo, si generamos una matriz de orden muy alto con el objeto de hacer después una gráfica. El hecho de que aparezca la matriz en pantalla puede resultar un poco engorroso. Para conseguir esto se pone un punto y coma al final de la instrucción.

Por ejemplo,

```
x=sin(3);
```

No aparece ningún resultado, pero ha realizado el cálculo, porque si escribimos el valor de `x`, aparecerá el valor `0.1411`.

En Matlab lo normal es ir asignando valores escalares o matriciales a variables, si en un momento determinado queremos saber con qué variables estamos trabajando, se puede escribir `>>who`. O bien, en el item *File* con *Show Workspace*.

En algunos momentos necesitaremos cargar o guardar ficheros. Para saber en qué directorio estamos, cambiar de directorio, etc.:

`pwd`	directorio en el que estamos
`cd nombre`	cambia al directorio duyo nombre indiquemos
`dir`	nos da una lista de ficheros del directorio actual

3.3. Matrices

Los vectores y las matrices son los elementos con los que trabaja Matlab. Veamos cómo se introducen y cómo se pueden hacer algunas de las operaciones básicas con ellos.

Un vector se puede definir introduciendo sus coordenadas, separadas por espacios o por comas, entre corchetes:

```
>> x=[1 2 3]

x  =
    1  2  3
```

Si queremos definir un vector columna, se separan las filas por puntos y comas, o bien se calcula el transpuesto de un vector fila con `>>x'`.

Otra forma de crear vectores es la siguiente:

```
>> x=1:0.5:3

x =

  1.0000    1.5000    2.0000    2.5000    3.0000
```

que genera un vector que va desde 1 hasta 10 con un paso de 0.5 unidades.

Para introducir una matriz, se separa cada fila con un punto y coma

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

Ejercicio 1 *Después de definida la matriz, probar los siguientes comandos e intentar descubrir para qué sirven:*

a) `>>A(2,3)` *o por ejemplo* `>>A(1,2)`

b) `A(:,1)` *y también* `A(2,:)`

c) `A^2` *y* `A.^2`*. ¿En qué se diferencian estos dos comandos?*

3.4. Gráficos con Matlab

Veamos cómo se pueden generar gráficos sencillos de funciones de una y de dos variables directamente. También veremos después que es posible hacerlos utilizando la *Matlab Symbolic Toolbox*.

Gráficas 2D

Para hacer gráficas de funciones de una variable con Matlab primero tenemos que crear una tabla de valores de la variable para después dibujar la función. Por ejemplo queremos dibujar la gráfica de la función $y = \text{sen}(t)$:

Primero creamos una tabla de valores para t:

```
>> t = 0:pi/100:2*pi;
```

Definimos la función:

```
>> y = sin(t);
```

y por último la dibujamos (ver figura 1):

```
>> plot(t,y)
```

Realmente lo que hemos hecho es dibujar 200 puntos de la función en el intervalo $[0, 2\pi]$

Veamos un ejemplo algo más complicado. Queremos dibujar ahora la gráfica de la función $y = xe^{-x^2}$.

Definimos los valores para los que queremos que hacer la gráfica:

```
>>x = -3:0.01:3;
```

Definimos la función:

```
>> y = x.*exp(-x.^2);
```

(¿Por qué hay que poner los puntos antes de las operaciones?)

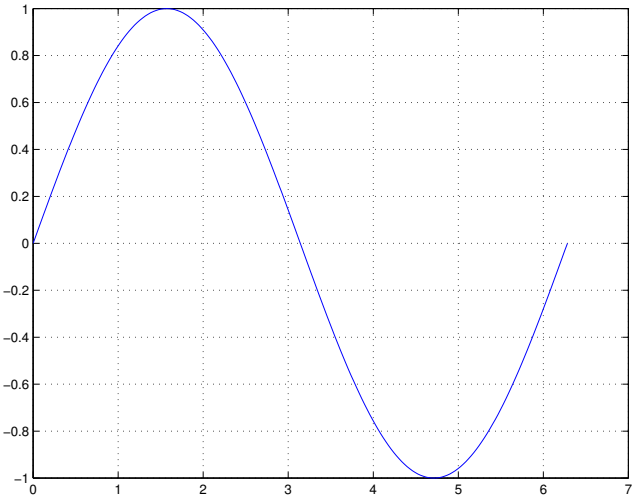

Figura 1: Gráfica de $y = \text{sen}(t)$

Y por último se escribe el comando para que ejecute el dibujo (figura 2):

```
>> plot(x,y)
```

Gráficas 3D

Veamos ahora cómo se pueden dibujar gráficas de funciones $z = f(x, y)$. Consideremos el ejemplo de la función $z = e^{-x^2-y^2}$. Vamos a hacer su gráfica en el dominio $[-3, 3] \times [-3, 3]$. En primer lugar debemos generar las tablas de valores para x e y. Esto se consigue con el comando:

```
[x,y]=meshgrid(-3:0.1:3);
```

Ahora definimos la función:

```
z=exp(-x.^2-y.^2);
```

Y por último se ejecuta el comando que produce la gráfica (figura 3):

```
>>mesh(x,y,z)
```

La presentación gráfica puede ser mucho más *artística*. Por ejemplo, con la secuencia de comandos siguiente:

```
>>surfl(x,y,z)

shading interp;

 colormap(pink)
```

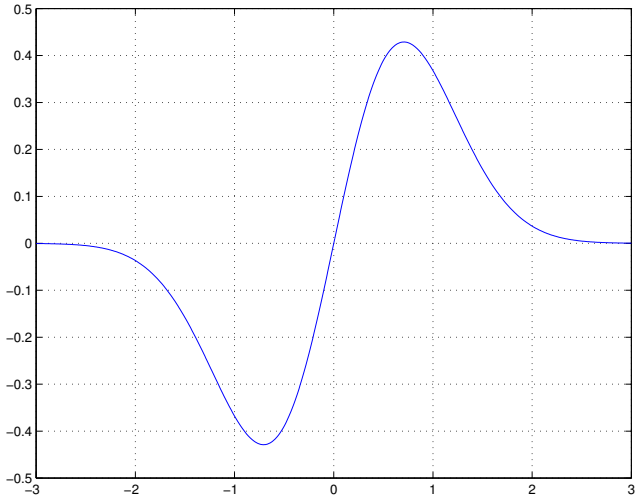

Figura 2: Gráfica de $y = xe^{-x^2}$

Se puede conseguir la gráfica siguiente (ver figura 4, la gráfica debería ser de color rosado).

Otro comando interesante en las gráficas 3D es: `rotate3d`, que nos permite, utilizando el ratón sobre la figura rotarla de manera interactiva en tres dimensiones.

3.5. Archivos de órdenes. Programación en Matlab

La manera de guardar comandos y construir programas que ejecuten determinadas rutinas consiste en crear lo que se llaman ficheros-M o *Scripts*.

Para crear un fichero-M necesitamos abrir un editor de texto (como el Bloc de Notas de Windows) y escribir los comandos. Luego ese fichero de texto debe guardarse con la extensión .m y a eso se le llama un *Script*. La versión 5 (Student Edition) incorpora un editor de *Scripts* basta abrir la opción `File M-File` de la barra superior.

Vamos a ver algunos ejemplos de ficheros-M:

Ejemplo 1 *Escribamos lo siguiente en el editor de Matlab y guardémoslo con el nombre* `evaluar.m` *(Si se utiliza el editor de Matlab no hace falta poner .m).*

```
% m-fichero para evaluar la funcion
%  y= a*exp(-1.2t) - 3.0*exp(-2t)
```

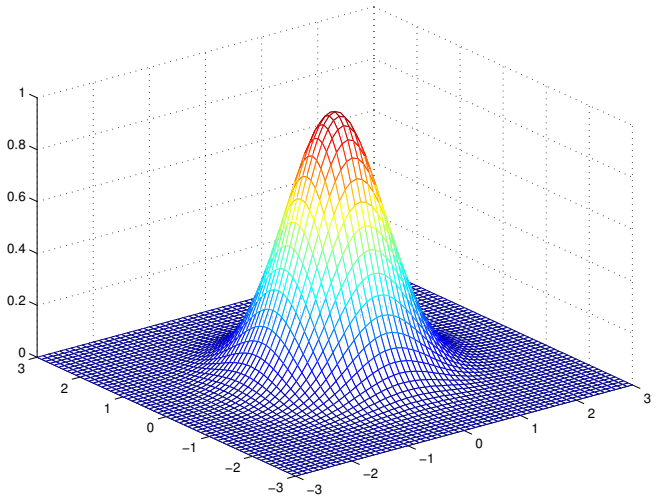

Figura 3: Gráfica de $z = e^{-x^2-y^2}$

```
% INPUT:  coeficiente a y tiempo t
% OUTPUT: funcion y(t)
 fprintf('Evaluar y= a*exp(-1.2*t) - 3.0*exp(-2*t) \n')
a=input('Coeficiente a, a= ');
 t=input('Variable t, t= ');
%
% Resultados
fprintf('\n Funcion: y=%g*exp(-1.2*%g)-3.0*exp(-2*%g) \n',a,t,t)
y=a*exp(-1.2*t) - 3.0*exp(-2*t)
% Fin
```

Observaciones:

1. *Lo que se ha escrito en cada línea detrás de* `%` *no forma parte del programa, son comentarios. Si hacemos,*

   ```
   >>help evaluar
   ```

   aparecen precisamente esos comentarios.

2. *Si hacemos,*

   ```
   >>type evaluar
   ```

   aparecerá en pantalla todo el programa escrito.

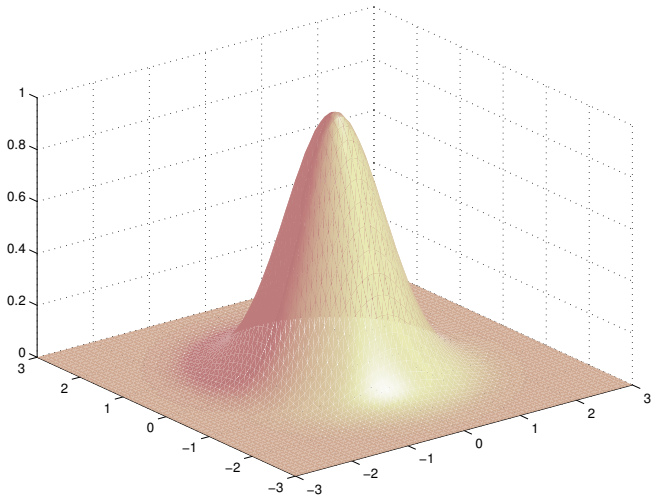

Figura 4: Gráfica *artística* de $z = e^{-x^2-y^2}$

3. *Por fin, para ejecutarlo basta con escribir el nombre. Por ejemplo,*

```
>>evaluar
Evaluar y= a*exp(-1.2*t) - 3.0*exp(-2*t)
Coeficiente a, a= 2
Variable t, t= 3
Funcion: y=2*exp(-1.2*3)-3.0*exp(-2*3)
y =
    0.0472
```

4. *Las dos líneas de INPUT son para pedir los datos.*

5. *La línea*

   `fprintf('\n Funcion: y=%g*exp(-1.2*%g)-3.0*exp(-2*%g) \n',a,t,t)`
   sirve para que aparezca en pantalla y sustituya en los lugares en los que aparece `%g` *los valores* `a,t,t`*, así en este orden.*

Vamos a ver ahora, a modo de curiosidad, un ejemplo de programa mucho más complicado. En el que se dibujan las series de Fourier de una función. Para profundizar más en programación en Matlab se puede consultar [H], donde se pueden encontrar muchos ejemplos comentados.

Ejemplo 2 *Escribir el siguiente programa y guardar en un fichero.m. Aquí se le puso de nombre* `grafour`*, pero por supuesto el nombre puede ser cualquier otro.*

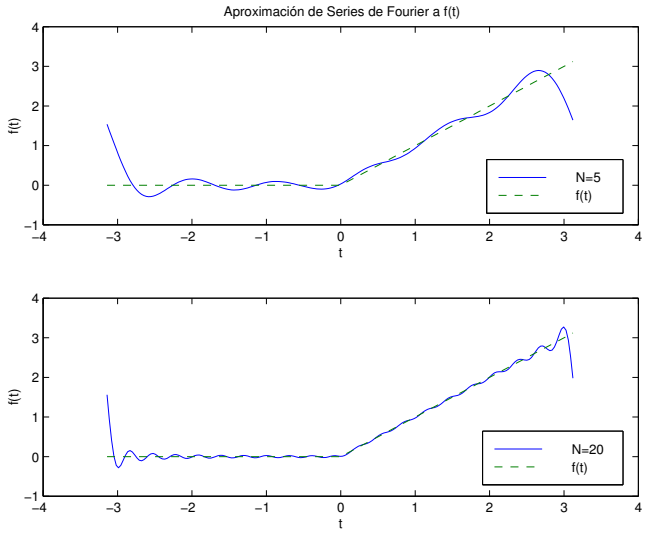

Figura 5: Series de Fourier

```
% grafour.m Grafica series de Fourier. funcion  f(t)
%   f(t)=0  -pi < t < 0
%   f(t)=t   0  < t < pi
%
% Grafica f(t) para 5 y 20 trminos de la serie
clear                          % Borramos todas las variables
t =[-pi:.031:pi];               % Rango de t para las gr'aficas
sizet=size(t);                 % n'umero de puntos de t
fn = pi/4*(ones(sizet));        % Aproximaci'on de Fourier en cada t
yplt=zeros(sizet);              %  para la gr'afica de f(t)
% 5 t'erminos
for n=1:5
 fn= fn+(1/pi)*(-2*cos((2*n-1)*t)/(2*n-1)^2)-((-1)^n*sin(n*t)/n);
end
%
for k=1:length(t)               % Crea f(t)
  if t(k) < 0
```

```
   yplt(k)=0;
  else
   yplt(k)=t(k);
  end
 end
clf                    % Borra anteriores figuras
subplot(2,1,1),plot(t,fn,t,yplt,'--');
xlabel('t')
ylabel('f(t)')
title('Aproximacion de Series de Fourier a f(t)')
legend(['N=',num2str(n)],'f(t)',4) % Nota en la gr'afica
% Se suman 15 t'erminos m'as
 for n=6:20
fn=fn+(1/pi)*(-2*cos((2*n-1)*t)/(2*n-1)^2)-((-1)^n*sin(n*t)/n);
end
subplot(2,1,2),plot(t,fn,t,yplt,'--');
 xlabel('t')
ylabel('f(t)')
 legend(['N=',num2str(n)],'f(t)',4)
%Fin
```

Si ejecutamos el programa obtendremos la gráfica de la figura 5

Ejercicio 2 *Escribir un programa que resuelva la ecuación de segundo grado:* $ax^2 + bx + c = 0$

3.6. Cálculo Simbólico con MATLAB

Las últimas versiones de Matlab incluyen lo que se llama *Matlab Symbolic Toolbox*. Dependiendo de la versión de Matlab que estemos utilizando, con ella vendrán incluidas diferentes *Toolboxes* que no son otra cosa que un conjunto de ficheros, programas, instrucciones, herramientas para un fin concreto. Las hay comerciales, que se venden con el producto y las hay de dominio público, que se pueden encontrar en Internet. O nosotros mismos podemos crearlas.

La versión 5 de Matlab, sobre la cual están preparadas estas notas, incluye, entre otras, la *Matlab Symbolic Toolbox* que, como su propio nombre indica, sirve para hacer Cálculo Simbólico. Hasta ahora, las operaciones que

se han mostrado se han realizado con números. Vamos a ver ahora cómo se pueden realizar cálculos abstractos.

⋆ Expresiones Simbólicas y Operaciones

Las expresiones simbólicas son cadenas de caracteres que representan números, funciones, operadores y variables. Las variables no necesitan tener valores predefinidos, a diferencia de todo lo comentado anteriormente.

Ejemplos de expresiones simbólicas y cómo se representan:

Expresión Simbólica	Representación en Matlab
$\frac{1}{2x^n}$	`'1/(2*x^n)'`
$y = \frac{1}{\sqrt{3x^3}}$	`'1/sqrt(3*x^3)'`
$\frac{d}{dx}(\cos(x))$	`diff('cos(x)')`
$f = \int_a^b \frac{x^3}{\sqrt{1-x}} dx$	`f=int('x^3/sqrt(1-x)','a','b')`
$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$	`M=sym('[a,b;c,d]')`

Obsérvese que se trata de escribir las expresiones simbólicas entre apóstrofes. Sin embargo, en algunos casos si no se pone el comando `sym`, puede dar lugar a confusión. Por ejemplo, si pusiéramos `M='[a,b;c,d]'` lo entendenría como una cadena de caracteres. Para que lo entienda como una matriz simbólica debemos escribirlo como en el último. Por otra parte, también resulta conveniente utilizar `sym` para poder utilizar posteriormente el comando `pretty`, que explicaremos más adelante, ya que de otra forma no funcionaría.

Para todas las variables simbólicas hay que utilizar letras minúsculas (`a,b,c,...,z`).

Operaciones algebraicas comunes

Supongamos que tenemos dos funciones:

$$f(x) = 3x^2 - 2x + 1 \quad \text{y} \quad g(x) = \frac{2}{x^2 - 1}$$

En primer lugar las definimos como expresiones simbólicas:

```
>>f=sym('3*x^2-2*x+1')
f =
```

```
    3*x^2-2*x+1
>>g=sym('2/(x^2-1)')
g =
    2/(x^2-1)
```

- Para sumar las dos funciones:

  ```
  >>symadd(f,g)

  ans =

      3*x^2-2*x+1+2/(x^2-1)
  ```

  La expresión se puede simplificar, si ello es posible, con:

  ```
  >>simplify(ans)

  ans =

      (3*x^4-2*x^2-2*x^3+2*x+1)/(x^2-1)
  ```

  Por último, como a veces resulta difícil interpretar el resultado debido a la estructura de la sintaxis de Matlab, se puede utilizar el comando `pretty`, que produce la impresión en pantalla de la expresión de una forma más parecida a como la escribimos habitualmente en Matemáticas.[1] (Importante: El comando `pretty` sólo funcionará si previamente las expresiones simbólicas se han definido con `sym`).

  ```
  >>pretty(ans)
  ```

$$\frac{3\,x^4 - 2\,x^2 - 2\,x^3 + 2\,x + 1}{x^2 - 1}$$

- Para restar las dos funciones:

  ```
  >>symsub(f,g)

  ans =

      3*x^2-2*x+1-2/(x^2-1)

  >>pretty(ans)
  ```

[1] También existe otro comando, `latex`, que produce como resultado la fórmula escrita en LaTeX. Esto puede resultar muy útil para pasar directamente a un fichero `.tex` sin más que copiar y pegar. La sintaxis es `latex(ans)` o el nombre de la expresión de que se trate. Este comando se ha utilizado continuamente en la elaboración de estas notas, que han sido escritas usando LaTeX2e.

$$3\,x^2 - 2\,x + 1 - \frac{2}{x^2 - 1}$$

- Para multiplicar las dos funciones:

```
>>symmul(f,g)
ans =
    2*(3*x^2-2*x+1)/(x^2-1)
>>pretty(ans)
```

$$2\,\frac{3\,x^2 - 2\,x + 1}{x^2 - 1}$$

- Para dividir las dos funciones:

```
>>symdiv(f,g)
ans =
    1/2*(3*x^2-2*x+1)*(x^2-1)
>>pretty(ans)
```

$$1/2\,\left(3\,x^2 - 2\,x + 1\right)\left(x^2 - 1\right)$$

- Para calcular potencias de una función a elevada a otra expresión:

```
>>sympow(f,'3*x')
ans =
    (3*x^2-2*x+1)^(3*x)
>>pretty(ans)
```

$$\left(3\,x^2 - 2\,x + 1\right)^{(3\,x)}$$

Otras operaciones

Tomemos ahora las funciones

$$f(x) = \frac{1}{1 + x^2} \quad \text{y} \quad g(x) = \operatorname{sen}(x)$$

- Composición de funciones. Para calcular $f(g(x))$ hacemos:

```
>>compose(f,g)
ans =
    1/(1+sin(x)^2)
>>pretty(ans)
```

$$\frac{1}{1+\sin(x)^2}$$

Si queremos calcular $g(f(x))$ tendremos que hacer

```
>>compose(g,f)
```

- Para calcular la inversa de una función:

```
>>finverse(g)
ans =
    asin(x)
```

Incluso nos avisa cuando la inversa no es única:

```
>>finverse(f)
Warning: finverse(1/(x^2+1)) is not unique.
ans =
    1/x*(-x*(x-1))^(1/2)
```

⋆ Derivadas e Integrales

Derivadas

Para derivar simbólicamente se utiliza el comando `diff` de alguna de las formas siguientes:

En primer lugar creamos una función, en este caso, va a ser la función:

$$f(x) = \frac{3\,x^4 - 5\,ax - 1}{x+2}$$

```
>>f=sym('(3*x^4-5*a*x-1)/(x+2)')
f =
```

```
(3*x^4-5*a*x-1)/(x+2)
```

Para calcular la derivada primera de la función con respecto de x:

```
>>diff(f)
ans =
(12*x^3-5*a)/(x+2)-(3*x^4-5*a*x-1)/(x+2)^2
```

La simplificamos:

```
>>simplify(ans)
ans =
-(-9*x^4-24*x^3+10*a-1)/(x+2)^2
```

Y por último, usamos el comando `pretty` para visualizarla mejor:

```
>>pretty(ans)
```

$$-\frac{-9\,x^4 - 24\,x^3 + 10\,a - 1}{(x+2)^2}$$

Sin embargo, si queremos que derive con respecto de la variable a:

```
>>diff(f,'a')
ans =
-5*x/(x+2)
```

Para calcular derivadas de orden superior, por ejemplo, la derivada segunda:

```
>>diff(f,2)
ans =
36*x^2/(x+2)-2*(12*x^3-5*a)/(x+2)^2+2*(3*x^4-5*a*x-1)/(x+2)^3
```

Simplificamos:

```
>>simplify(ans)
ans =
2*(9*x^4+48*x^3+72*x^2+10*a-1)/(x+2)^3
```

Y,

```
>>pretty(ans)
```

$$2\,\frac{9\,x^4 + 48\,x^3 + 72\,x^2 + 10\,a - 1}{(x+2)^3}$$

Si queremos evaluar esta derivada en un punto, por ejemplo, queremos calcular $f''(2)$. Una vez que está calculada la derivada basta con sustituir x por 2, que se hace con el comando:

```
>>subs(ans,'x','2')
ans =
   815/32+5/16*a
>>pretty(ans)
```

$$\frac{815}{32} + \frac{5}{16}\,\mathtt{a}$$

Integrales

El comando que se utiliza para integrar es `int(f)` done `f` es una expresión simbólica. Veamos algunos ejemplos:

Consideramos la función:

$$f(x) = \frac{1}{x^2+2}$$

Para calcular una primitiva, primero la definimos:

```
>>f=sym('1/(x^2+2)')
f =
   1/(x^2+2)
>>int(f)
ans =
   1/2*2^(1/2)*atan(1/2*x*2^(1/2))
>>pretty(ans)
```

$$1/2\,2^{1/2}\arctan(1/2\,\mathtt{x}2^{1/2})$$

Si quisiéramos calcular la integral definida:

$$\int_0^1 x^2 dx$$

```
>>int('x^2',0,1)
ans =
   1/3
```

Obsérvese que los resultados numéricos cuando trabajamos con expresiones simbólicas aparecen por defecto en forma algebraica. Para obtener un resultado en forma decimal. Basta con utilizar el comando `>>numeric(ans)`, y lo calculará en forma decimal.

⋆ Representación Gráfica de Expresiones Simbólicas

La *Toolbox* de Matemática Simbólica de Matlab dispone de un comando, `ezplot` que permite dibujar gráficas de funciones de una variable.

Por ejemplo, queremos dibujar la gráfica de la función:

$$f(x) = x\ln(x^2)$$

Podemos hacer directamente:

```
>>ezplot('x*log(x^2)')
```

Entonces aparece la figura 6. Si no se especifica intervalo la gráfica aparece, por defecto, representada en el intervalo $[-2\pi, 2\pi]$, siempre que esté definida en este intervalo, salvo conjuntos de medida cero. (Nótese que, en este caso, la función tiene una singularidad en el origen que aparece reflejada en la figura, aunque no se aprecie muy bien). Si probáramos con la función $f(x) = x\ln(x)$, sólo aparecería el dibujo en la parte positiva.

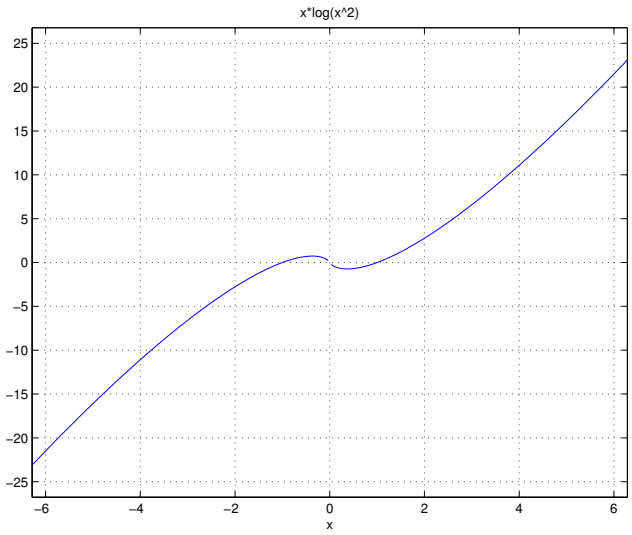

Figura 6: Gráfica de $f(x) = x\ln(x^2)$ generada con el comando `ezplot`

Una vez que tenemos la gráfica podemos agrandarla para ver algún detalle. Esto se consigue con el comando `>>zoom on` y después pulsando con el ratón sobre la figura, experimentando un poco se puede descubrir fácilmente de qué manera funciona el `Zoom`.

Si queremos que la gráfica la haga en un determinado intervalo, tendremos que utilizar el comando de la forma:

```
>>ezplot(funcion,[a,b])
```

Ejercicio 3 *Dibujar la gráfica de la función* $f(x) = e^{-x^2}$ *en diferentes intervalos.*

⋆ Resolución de Ecuaciones

Ecuaciones Algebraicas

Matlab dispone de un comando, `solve`, que de la misma forma que en Derive permite resolver ecuaciones algebraicas. Veamos algunos ejemplos de su utilización:

- Queremos resolver la ecuación de segundo grado: $ax^2 + bx + c = 0$.

  ```
  >>solve('a*x^2+b*x+c')

  ans =

      [ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
      [ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
  ```

  Como vemos no es necesario escribir `=0`, aunque no pasa nada si se escribe. Si no se especifica la variable con respecto de la cual queremos que despeje optará por la x. Aunque se puede especificar otra.

- Por ejemplo, si queremos resolver la misma ecuación, pero con respecto de la variable b:

  ```
  >>solve('a*x^2+b*x+c', 'b')

  ans =

      -(a*x^2+c)/x
  ```

- Si la ecuación tiene soluciones complejas, también las encuentra. Por ejemplo, la ecuación $x^3 - x^2 + x - 1 = 0$ tiene las soluciones complejas: $1, +i, -i$.

  ```
  >>solve('x^3-x^2+x-1')

  ans =

      [  1]
      [  i]
      [ -i]
  ```

- Los resultados que ofrece el comando `solve` son por defecto en forma algebraica. Sin embargo, se pueden convertir a formato numérico. La ecuación $x^2 - x - 1 = 0$ tiene por soluciones: $x_1 = \dfrac{1+\sqrt{5}}{2}$ y $x_2 = \dfrac{1+\sqrt{5}}{2}$.

```
>>solve('x^2-x-1')

ans =

    [ 1/2*5^(1/2)+1/2]

    [ 1/2-1/2*5^(1/2)]
```

Pero si queremos ver los resultados numéricos:

```
>>numeric(ans)

ans =

    1.6180

    -0.6180
```

- También se pueden resolver sistemas de ecuaciones. Basta con poner todas las ecuaciones separadas por comas. Por ejemplo, para resolver el sistema de ecuaciones:

$$\begin{cases} x^2 + y^2 = 1 \\ x + y = 1 \end{cases}$$

```
>>[x,y]=solve('x^2+y^2=1','x+y=1')

x =

    [ 1]

    [ 0]

y =

    [ 0]

    [ 1]
```

En este caso, es necesario especificar previamente las variables, `[x,y]=`, ya que de otra forma no nos daría las soluciones sino que indicaría que x e y son dos matrices simbólicas 2×1 pero sin escribirlas.

- Si no se puede encontrar una solución en forma algebraica, devuelve una aproximación numérica de la solución. Por ejemplo, la solución de la ecuación $\text{sen}(x) = x - 1$:

```
>>solve('sin(x)=x-1')

ans =

    1.934563210752024267563261453768 9
```

Ecuaciones Diferenciales

La cantidad de problemas relacionados con la Ecuaciones Deferenciales que se pueden resolver con Matlab haría necesario uno o varios libros y no unas notas introductorias como éstas. Sin embargo, comentaremos algunas cuestiones elementales para resolver Ecuaciones Diferenciales Ordinarias simbólicamente. Para profundizar más sobre esste tema se puede consultar [H].

El comando que se utiliza para este fin es `dsolve` que tiene un funcionamiento muy parecido al descrito en el apartado anterior (`solve`) para las ecuaciones algebraicas.

Recordemos en primer lugar que una Ecuación Diferencial Ordinaria es una ecuación en la que la incógnita es una (o varias funciones) de una variable y en la que aparecen derivadas de la función con respecto de la variable. Por ejemplo, la ecuación

$$\frac{dy}{dx} = 1 + y^2$$

es una ecuación diferencial ordinaria de primer orden (porque la derivada más alta que aparece es la primera) en la que la variable es la x y la función incógnita es la y. Para resolverla con Matlab

```
>>dsolve('Dy=1+y^2','x')
ans =
   tan(x-C1)
```

Como se puede observar la derivada se representa por `Dy`, si hubiera sido una derivada de otro orden, por ejemplo, $\frac{d^3y}{dx^3}$, habríamos puesto `D3y`. También hemos de especificar cuál es la variable, si no lo hacemos escribirá la solución por defecto en función de t.

Lo que acabamos de hacer es calcular la solución general de la ecuación, por lo que aparece la constante `c1` que es la constante de integración. Si

queremos resolver un problema de valores iniciales, es decir, la ecuación junto con una condición inicial:

$$\begin{cases} \dfrac{dy}{dx} = 1 + y^2 \\ y(0) = 1 \end{cases}$$

```
>>dsolve('Dy=1+y^2', 'y(0)=1','x')
ans =
   tan(x+1/4*pi)
```

Otro ejemplo, una ecuación de segundo orden:

$$\frac{d^2x}{dt^2} - 2\frac{dx}{dt} - 3x = 0$$

Ahora la variable es t y la función $x(t)$.

```
>>dsolve('D2x-2*Dx-3*x=0')
ans =
   (C1*exp(3*t)*exp(t)+C2)/exp(t)
```

donde `c1` y `c2` son las constantes de integración. Si quisiéramos resolverlo con las condiciones iniciales:

$$x(0) = 0, \quad x'(0) = 1$$

```
>>dsolve('D2x-2*Dx-3*x=0', 'x(0)=0,Dx(0)=1')
ans =
   (1/4*exp(3*t)*exp(t)-1/4)/exp(t)
```

⋆ Sumas de Riemann

Una herramienta interesante que tiene Matlab se consigue con el comando `rsums`. La sintaxis del comando es `rsums(f)` donde `f` es una función simbólica. Aparece una ventana gráfica (ver figura 7). Debajo de la gráfica hay un deslizador (que no se aprecia en nuestra figura) que permite cambiar con el ratón el número de rectángulos usados para aproximar el área de la curva, desde 2 hasta 256 rectángulos como máximo. En la parte superior aparece la función junto con el resultado numérico de la suma.

Veamos un ejemplo:

```
>>f='10*x*exp(-5*x^2)' % creamos una funci'on
f =
    10*x*exp(-5*x^2)
```

```
>>vpa(int(f,0,1),6) % evaluamos la integral de 0 a 1
ans =
    .993262
>>rsums(f)  % aproximaci'on de Riemann desde 0 hasta 1
```

Y aparece la figura interactiva mencionada antes, en principio aparecen 10 rectángulos, aunque este número se puede cambiar como se ha indicado antes. Esta herramienta tiene una limitación, sea cual sea la función sólo la representa en el intervalo $[0, 1]$.

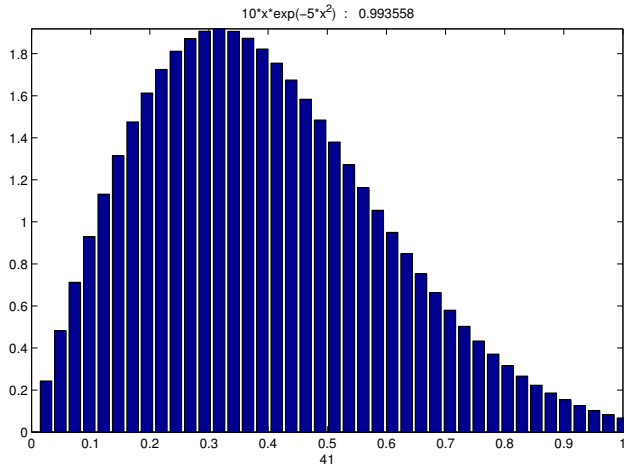

Figura 7: Sumas de Riemann para la función $f(x) = 10xe^{-5x^2}$

Ejercicio 4 *A pesar de que el comando* `rsums` *sólo aproxima la integral en el intervalo* $[0, 1]$. *También se puede utilizar para aproximar en un intervalo* $[a, b]$ *cualquiera, basta con encontrar el cambio de variable adecuado:*

a) Encontrar el cambio de variable que convierte la integral $\int_a^b f(x)dx$ *en la integral* $\int_0^1 g(t)dt$.

b) Utilizar el cambio anterior para aproximar la siguiente integral usando `rsums`.

$$\int_1^2 \log(x)dx$$

Utilizar el comando `vpa(int(funcion,a,b),6)` *para calcular la integral con 6 decimales exactos y comprobar el resultado con el de las Sumas de Riemann.*

⋆ Calculadora de Funciones

Otra herramienta interesante de la que dispone Matlab es la Calculadora de Funciones (`funtool`). Se activa escribiendo el comando:

`>>funtool`

Entonces aparecen tres nuevas pantallas, dos de ellas conteniendo sendas gráficas de funciones y la tercera es la calculadora (figura 8).

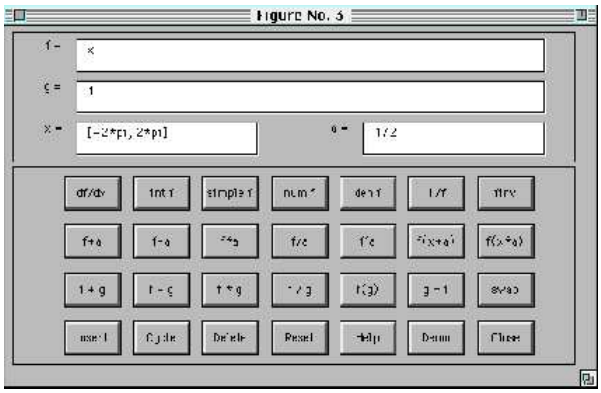

Figura 8: Calculadora de Funciones

FUNTOOL es una calculadora gráfica interactiva que manipula funciones de una variable. En todo momento hay dos funciones que aparecen en las dos gráficas, $f(x)$ y $g(x)$. El resultado de la mayoría de las operaciones afecta a $f(x)$.

En las ventanitas etiquetadas con `"f ="` y `"g ="` se puede escribir y cambiar las funciones que aparecen por defecto para introducir nuevas funciones. Lo mismo que las etiquetadas con `"x ="`, que puede ser cambiada para introducir un nuevo dominio, y la `"a ="` en la que se puede introducir un nuevo parámetro.

La fila superior de teclas de la calculadora son operaciones que sólo afectan a la función $f(x)$. Estas operaciones son:

`df/dx` - Derivada de $f(x)$.

`int f` - Integral de $f(x)$.

`simple f` - Simplifica la expresión, si es posible.
`num f` - Extrae el numerador de una expresión racional.
`den f` - Lo mismo, pero ahora el denominador.
`1/f` - Reemplaza $f(x)$ por $\frac{1}{f(x)}$.
`finv` - Calcula la inversa de $f(x)$.
Evidentemente las operaciones `int(f)` y `finv` no siempre funcionan.

La segunda fila de teclas trasladan y reescalan la función $f(x)$ según el valor del parámetro `a`. Las operaciones son:
`f + a` - Reemplaza $f(x)$ por $f(x) + a$.
`f - a` - Reemplaza $f(x)$ por $f(x) - a$.
`f * a` - Reemplaza $f(x)$ por $a.f(x)$.
`f / a` - Reemplaza $f(x)$ por $\frac{f(x)}{a}$.
`f ^ a` - Reemplaza $f(x)$ por $f(x)^a$.
`f(x+a)` - Reemplaza $f(x)$ por $f(x + a)$.
`f(x*a)` - Reemplaza $f(x)$ por $f(ax)$.

La tercera fila de teclas son operaciones en las que intervienen las dos funciones $f(x)$ y $g(x)$. Las operaciones son:
`f + g` - Reemplaza $f(x)$ por $f(x) + g(x)$.
`f - g` - Reemplaza $f(x)$ por $f(x) - g(x)$.
`f * g` - Reemplaza $f(x)$ por el producto $f(x).g(x)$.
`f / g` - Reemplaza $f(x)$ por $\frac{f(x)}{g(x)}$.
`f(g)` - Reemplaza $f(x)$ por la composición $f(g(x))$.
`g = f` - Reemplaza $g(x)$ por $f(x)$.
`swap` - Intercambia $f(x)$ y $g(x)$.

Las tres primeras teclas de la cuarta fila producen una lista de funciones. La tecla `Insert` añade la función actual a la lista. La tecla `Cycle` hace aparecer todas las funciones de la lista. Y la tecla `Delete` borra la función actual de la lista. La lista de funciones está en un fichero interno que se llama `fxlist` que lleva por defecto una serie de funciones interesantes.

La tecla `Reset` devuelve los valores de f, g, x, a y `fxlist` a sus valores iniciales por defecto. La tecla `Help` imprime en pantalla ayuda (lo mismo que en estas instrucciones pero en Inglés).

La tecla `Demo` propone un curioso problema: ¿Es posible generar la función sen(x) sin tocar el teclado, utilizando sólo el ratón pulsando las teclas adecuadas de la calculadora? La `Demo` lo hace con un `Reset` y nueve çlics"del ratón y proponen como problema intentar hacerlo con menos çlics". Si alguien

lo consigue ruegan ponerse en contacto con la siguiente dirección electrónica: moler@mathworks.com. ¿Habrá premio? Por último con la tecla `Close` se acaba el juego.

Ejercicio 5 *¿Cuáles son los pasos que se siguen en la* `Demo` *para obtener la función* $\mathrm{sen}(x)$*?*

Intentar obtener la función $\mathrm{sen}(x)$ *con menos clics que en la* `Demo`*. (Es posible, no es una broma).*

Ejercicio 6 *Escribir un breve guión de una práctica para realizar en un Aula de Informática utilizando la* `Calculadora de Funciones` *de Matlab.*

4. Prácticas con MATLAB

Se proponen a continuación algunas prácticas con el Programa Matlab. Una referencia muy interesante en la que se pueden encontrar numerosas prácticas con muy distintos niveles de dificultad es el libro *Mathematical Explorations with MATLAB*. (ver Bibliografía, [Ch]).

4.1. Práctica Primera: Matrices y Sistemas de Ecuaciones Lineales

En esta práctica vamos a profundizar un poco en las capacidades de Matlab para trabajar con matrices. Veremos en primer lugar algunas oparaciones y comandos básicos y no tan básicos que tiene el programa para trabajar con matrices. Después, como aplicación de algunos de esos comandos comentaremos algunas maneras de afrontar la solución de un sistema de ecuaciones lineales.

4.1.1. Operaciones y Comandos para Matrices

Hemos visto en la introducción anterior cómo se introducen las matrices en Matlab. Veamos un ejemplo para introducir algunos de los comandos básicos:

Ejemplo 1 Operaciones Elementales
Definimos dos matrices:

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

>>B=[3 4;-1 5]
B =
      3            4
     -1            5
```

- *Para* **sumar** *las dos matrices:*

```
>>A+B
ans =
      5            5
      2            7
```

- *Para* **multiplicar** *una* **matriz** *por un* **escalar***:*

```
>>3*A
ans =
      6            3
      9            6
```

- **Producto** *de matrices:*

```
>>C=A*B
C =
      5           13
      7           22
```

Siempre que los tamaños de las matrices sean los adecuados. Para saber cuál es el **tamaño** *de una* **matriz** *con la que estamos trabajando,*

```
>>size(A)
ans =
      2            2
```

Que quiere decir, evidentemente, 2 filas y 2 columnas.

- *Para calcular la matriz* **transpuesta***:*

```
>>A'
ans =
      2            3
      1            2
```

Ejercicio 1 *Utilizando las matrices definidas en el ejemplo anterior, comprobar que* $(AB)^t = B^t A^t$*. (*A^t *es la transpuesta de* A*).*

Ejemplo 2 Matrices especiales con Matlab

- *Para generar la matriz* **identidad** *cuadrada,*

```
>>eye(3)
ans =
      1            0            0
      0            1            0
      0            0            1
```

¿Por qué habrán elegido el nombre `eye`*?*

• *Una matriz* 3×2 *llena de unos,*

```
>>ones(3,2)
```

• *Y si queremos que esté llena de ceros,*

```
>>zeros(3,2)
```

• *Para generar una matriz con números* **aleatorios** *uniformemente distribuidos entre 0 y 1,*

```
>>rand(3,2)
```

Si se usa el comando `randn` *los números aleatorios son normalmente distribuidos, siguiendo la Normal Estandar* $N(0,1)$*.*

Ejemplo 3 Rango, Inversa y Determinante

• *Definimos la matriz,*

```
>>X=[2 3 4; 1 -1 0]
X =
     2     3     4
     1    -1     0
```

Para calcular su **rango**,

```
>>rank(X)
ans =
     2
```

• *Supongamos que tenemos definida la siguiente matriz,*

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

Para calcular su **inversa**,

```
>>inv(H)
ans =
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028
```

Y si queremos ver el resultado en forma racional,

```
>>format rational
>>inv(H)
ans =
    53/360      -13/90       23/360
   -11/180        1/45       19/180
    -7/360       17/90      -37/360
```

(Para ver todas las opciones del comando `format` *hacer* `help format`*)*

- *Para calcular el determinante de la matriz anterior* `H`*,*

```
>>det(H)
ans =
   -360
```

Ejercicio 2 *Generar una matriz cualquiera, por ejemplo* 25×25*, y calcular su inversa, su rango y su determiante. (Ni que decir tiene que no disponemos de tiempo para escribir todos los elementos de la matriz uno a uno).*

Ejemplo 4 Los comandos especiales `rref` y `rrefmovie`

- *El comando* `rref` *produce una forma reducida escalonada por filas de una matriz usando la eliminación de Gauss-Jordan, es decir, haciendo ceros por debajo y por encima de la diagonal principal sin mover las columnas.*

Por ejemplo, definimos la matriz,

```
>>A=[-1 2 -1;2 1 2;2 4 2]

A =

     -1           2          -1
      2           1           2
      2           4           2
```

Ahora escribimos el comando aplicado a la matriz,

```
>>R=rref(A)

R =

     1           0           1
     0           1           0
     0           0           0
```

• *El comando* `rrefmovie` *produce exactamente el mismo resultado pero nos indica paso a paso cómo se va obteniendo la matriz resultado e incluso qué filas o columnas son despreciables (por ser linealmente dependientes de las otras), información muy últil si queremos calcular el rango de la matriz por ejemplo. Es decir, produce una especie de película (movie) de todo el proceso.*

```
>>rrefmovie(A)
Original matrix

A =

    -1           2          -1
     2           1           2
     2           4           2

Press any key to continue. . .
```

Ahora pulsamos una tecla para continuar,

```
swap rows 1 and 2

A =

     2           1           2
    -1           2          -1
     2           4           2

Press any key to continue. . .
```

Nos indica que ha intercambiado la primera y segunda filas, pulsamos de nuevo una tecla,

```
pivot = A(1,1)

A =

      1           1/2           1
     -1            2           -1
      2            4           2

Press any key to continue. . .
```

Ahora nos indica que va a pivotear sobre el elemento (1,1) de la matriz,

```
eliminate in column 1

A =

      1           1/2           1
     -1            2           -1
      2            4           2

Press any key to continue. . .
```

Ahora nos está indicando que va a eliminar (hacer ceros) en la primera columna y así sucesivamente hasta obtener el mismo resultado que nos dió el comando `rref`*.*

Ejercicio 3 *a) Calcular el rango de la matriz siguiente utilizando el comando* `rref` *o* `rrefmovie`*:*

$$A = \begin{pmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{pmatrix}$$

b) Si la matriz A es cuadrada y no singular, es decir $\det(A) \neq 0$*, ¿cuál será la matriz* $R = \texttt{rref}(A)$*?*

c) ¿Cómo podemos utilizar estos comandos para calcular la inversa de una matriz invertible? Aplicarlo a la matriz,

$$B = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$$

Para verificar el resultado se puede calcular la inversa directamente con `inv(B)`.

4.1.2. Sistemas de Ecuaciones Lineales

Un sistema de ecuaciones lineales,

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \end{cases}$$

con m ecuaciones y n incógnitas se puede escribir en forma matricial,

$$A\mathbf{x} = \mathbf{b}$$

donde,

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}; \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad \text{y} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Vamos a ver mediante algunos ejemplos y ejercicios cómo se pueden resolver los sistemas de ecuaciones lineales utilizando algunos de los comandos de Matlab descritos anteriormente.

Ejemplo 5 *Consideremos el sistema,*

$$\begin{cases} 2x - y + z = 3 \\ x + y \quad\quad = 3 \\ \quad\quad y - 3z = -7 \end{cases}$$

entonces, siguiendo la notación anterior,

$$A = \begin{pmatrix} 2 & -1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & -3 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \text{y} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 3 \\ -7 \end{pmatrix}$$

Se trata de un sistema con solución única, ya que el determinante de A es distinto de cero,

```
>>det(A)
ans =
    -8
```

Entonces, una forma de resolver el sistema consiste en despejar $\mathbf{x}$*,*

$$\mathbf{x} = A^{-1}\mathbf{b}$$

Con Matlab, primero introducimos el vector $\mathbf{b}$*,*

```
>>b=[3 3 -7]'
b =
     3
     3
    -7
>>x=inv(A)*b
x =
     1
     2
     3
```

Sin embargo, hay otra forma de hacerlo, utilizando lo que en Matlab se denomina como **división matricial a la izquierda***:*

```
>>x=A\b
x =
     1
     2
     3
```

En este caso, el resultado es el mismo, pero es diferente la forma en la que trabaja el ordenador. En este segundo caso el método que utiliza es el de la **factorización LU***, que es una modificación de la eliminación gaussiana.*

Ejercicio 4 *Vamos a contar el número de operaciones elementales (sumas y productos) que se utilizan para resolver el sistema utilizando los dos métodos.*

Para ello vamos a utilizar el comando `flops`*. Este comando cuenta el número de operaciones elementales que se han realizado en una sesión determinada.*

Poner el contador a cero con `>>flops(0)`*. Resolver el sistema de una de las formas y contar las operaciones, escribiendo* `>>flops` *y repetir la operación resolviéndolo de la otra forma. ¿En cuál de los dos casos se utilizan menos operaciones?*

Ejercicio 5 *Vamos a ver cómo resuelve Matlab en realidad el sistema cuando se utiliza la opción:* `>>x=A\b`*. El proceso se puede describir dividir en tres etapas:*

1) Calcula una matriz triangular inferior L, una matriz triangular superior U y una matriz de permutación P tales que $PA = LU$. P es simplemente la matriz identidad I con sus filas cambiadas de orden.

2) Resuelve $L\mathbf{y} = P\mathbf{b}$.

3) Por último, se resuelve $U\mathbf{x} = \mathbf{y}$.

La primera etapa es lo que se conoce con el nombre de **factorización LU** *y es el paso más importante.*

Por lo tanto, en Matlab sería equivalente utilizar:

```
>>x=A\b
```

que utilizar el siguiente proceso:

```
>>[L,U,P]=lu(A); % Comando que calcula las matrices L, U, P
>>B=P*b;
>>y=L\B;
>>x=U\y
```

Resolver el siguiente sistema utilizando los dos procedimientos anteriormente descritos y comprobar que sale la misma solución.

$$\begin{pmatrix} 1 & 1 & 0 & 3 \\ 2 & 1 & -1 & 1 \\ 3 & -1 & -1 & 2 \\ -1 & 2 & 3 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ -3 \\ 4 \end{pmatrix}$$

Ejercicio 6 *El método* `>>x=A\b` *funciona si hay más ecuaciones que incógnitas (siempre que tenga solución única). E incluso en el caso en el que hay menos ecuaciones que incógnitas, con infinitas soluciones, Matlab ofrece dos de estas soluciones directamente:*

Con `>>x=A\b` *nos da una solución que tiene ceros para algunas de las incógnitas.*

Con otro comando, `>>x=pinv(A)*b` *se obtiene una solución del sistema donde la norma euclídea de* **x** *es la más pequeña de todas las posibles. ¿Es siempre única esta solución con longitud mínima?*

Aplicando lo anterior encontrar las soluciones de los sistemas,

$$a) \begin{cases} x+y=1 \\ x-y=1 \\ 2x+y=2 \end{cases} \qquad b) \begin{cases} x+y+z=3 \\ x-y+z=2 \end{cases}$$

Investigar qué hace el comando `>>pinv(A)`.

4.2. Práctica Segunda: Iteraciones y Diagramas de Telaraña

En esta práctica, utilizando como ejemplo el estudio de sucesiones obtenidas a partir de la iteración de funciones, vamos a ver algún ejemplo más de fichero-M. En este caso gráficos especiales llamados *Diagramas de Telaraña*.

Sea una función $y = f(x)$, tomemos un punto cualquiera x_0, sea $x_1 = f(x_0)$, volvemos a sustituir en la función y obtenemos $x_2 = f(x_1)$, si repetimos el proceso indefinidamente obtenemos una sucesión por iteración,

$$x_0, x_1, x_2, x_3, \ldots$$

que queda determinada por $f(x)$ y el valor de x_0, mediante la recurrencia

$$\begin{cases} x_0 \\ x_{n+1} = f(x_n) \text{ para } n = 0, 1, 2, \ldots \end{cases}$$

Para estudiar el comportamiento de tales sucesiones se utilizan los llamados *diagramas de telaraña*, que se construyen de la siguiente manera: Empezamos con la gráfica de $y = f(x)$ y con la recta $y = x$. Del punto $(x_0, 0)$ sobre el eje X se dibuja una recta vertical hasta la gráfica de la función en el punto (x_0, x_1), después una recta horizontal desde este punto hasta la recta $y = x$ en el punto (x_1, x_1). Ahora el proceso se vuelve a repetir, una recta vertical hasta la gráfica en (x_1, x_2), una recta horizontal hasta $y = x$ en el punto (x_2, x_2) y así sucesivamente. Las sucesivas rectas verticales tienen como coordenada x, $x_1, x_2, \ldots$. De esta manera se puede seguir gráficamente el proceso de la sucesión. La figura 9 muestra 8 iteraciones de la función $f(x) = 3x(1 - x)$ tomando $x_0 = 0{,}5$.

Iteraciones de la Función Cuadrática $\mathbf{f(x) = \lambda x(1 - x)}$

El fichero-M descrito a continuación permite dibujar el diagrama correspondiente a la sucesión que aparece al iterar la *función cuadrática*. Grabarlo siguiendo las instrucciones indicadas en la sección 3.5. Para ejecutarlo, bastaba con poner el nombre con el que se haya guardado, aquí se guardó con el nombre: `telaq.m`, e ir introduciendo los datos que va solicitando. Después habrá que pulsar INTRO tantas veces como el número de iteraciones que se haya indicado.

Fichero `telaq.m`

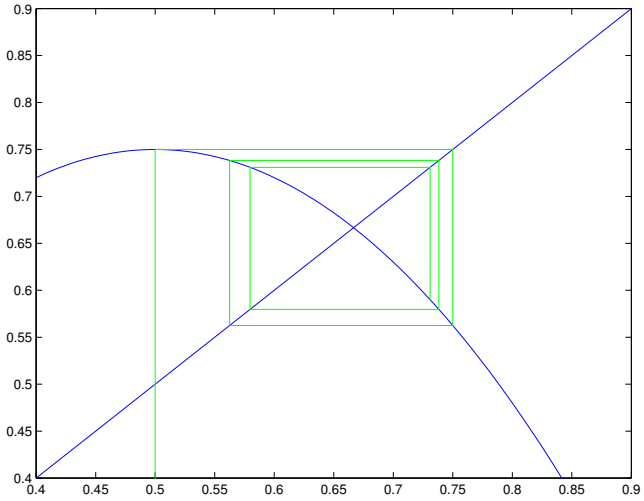

Figura 9: Diagrama de Telaraña

```
% Dibuja un diagrama de "telara~na" para y=lambda.x.(1-x),
% empezando con el valor x0.
% Se piden los extremos superior e inferior de x e y par la
% ventana gr'afica.
% Hay que pulsar INTRO sucesivamente hasta que se hayan dibujado
% todas las iteraciones
%
% Al final despu'es del dibujo se imprimen en pantalla los valores
% de las iteraciones.

   lambda=input('Escribir el valor de lambda ');
   x0=input('Escribir el valor de x0 ');
   xl=input('Escribir el extremo inferior de x ');
   xu=input('Escribir el extremo superior de x ');
   yl=input('Escribir el extremo inferior de y ');
   yu=input('Escribir el extremo superior de y ');
   n=input('Escribir el numero de iteraciones ');

dx=(xu-xl)/100;
x=xl:dx:xu;
y=lambda*x.*(1-x);
```

```
plot(x,y) axis([xl xu yl yu]);

hold on

plot([xl xu],[xl xu],'b')   % Dibuja la diagonal en azul

clear x y iterate x(1)=x0;
y(1)=0;
x(2)=x0;
y(2)=0;
x(3)=x0;
y(3)=lambda*x0*(1-x0);
iterate(1)=x0;
iterate(2)=y(3);

plot(x,y,'g') i=3;
x(1)=x0+.1;

while abs(x(3)-x(1)) > .001 & i <= n
    x(1)=x(3);
    y(1)=y(3);
    x(2)=y(1);
    y(2)=x(2);
    x(3)=x(2);
    y(3)=lambda*x(2)*(1-x(2));
    iterate(i)=y(3);
    plot(x,y,'g')
    pause
    i=i+1;
end

iterate'

hold off
```

Ejercicio 1 *Dibujar el diagrama de la función cuadrática correspondiente al valor* $\lambda = 3{,}922193334$, *con* $xl = 0$, $xu = 1$, $yl = 0$, $yu = 1$, $x_0 = 0{,}5$. *(xl, xu son los extremos inferior y superior, lo mismo con y). Aplicar 20 iteraciones.*

Se observará que se produce lo que se denomina un ciclo.

Ejercicio 2 *Utilizar el fichero* `telaq.m` *para probar los casos que se enumeran en la tabla siguiente.*

λ	x_0	xl	xu	yl	yu	iteraciones
2.5	0.5	0.5	0.7	0.5	0.7	20
3.2	0.5	0.4	0.9	0.4	0.9	20
3.4	0.5	0.4	0.9	0.4	0.9	40
3.5	0.5	0.3	0.9	0.3	0.9	20
3.8	0.5	0	1	0	1	50
3.83	0.5	0	1	0	1	20

Iteraciones de la Función Exponencial $\mathbf{f(x) = k^x}$

Ejercicio 3 *Modificar el fichero* `telaq.m` *para que aparezca la gráfica de la función exponencial* $f(x) = k^x$*, donde* k *es un número real fijo. Llammar al nuevo fichero* `telaexp.m`*. (Tener en cuenta que, dado que en el fichero* x *es un vector, habrá que usar* `k.^x`*). Investigar si las sucesiones que empiezan en* x_0 *son convergentes o divergentes y, si son convergentes, cuál es el límite. Si depende del valor de* k*, encontrar los valores en los que el comportamiento cambia. (Sugerencia: Probar con los valores* $k = 0{,}05$*,* $k = 0{,}5$*,* $k = 1{,}4$*,* $k = 1{,}5$*. Pensar en lo que ocurre entre* $k = 1{,}4$ *y* $k = 1{,}5$*).*

Ejercicio 4 *¿Cuál es el valor máximo que alcanza* $f(x)$*? ¿De qué manera está relacionado esto con el comportamiento de las sucesiones que genera la función?*

5. Matemáticas en el Aula de Informática

Una vez que sabemos utilizar un asistente matemático, DERIVE, MATLAB, MAPLE, MATHEMATICA, etc., surge una cuestión importante: ¿Cómo podemos utilizarlo en una clase con los alumnos? La respuesta a esta pregunta no es ni mucho menos trivial ya que, a la hora de llevar a la clase de Matemáticas el ordenador aparecen multitud de problemas que pueden desanimar al más entusiasta.

Por ejemplo: problemas de tipo técnico, inadecuación de los equipamientos informáticos, problemas con las licencias de los programas y sobre todo, el tiempo que lleva planificar un curso en el que queramos introducir prácticas informáticas.

En la Sección Departamental de Matemática Aplicada de la Facultad de Químicas de la UCM llevamos algunos años enfrentándonos a todos estos problemas, con lo que hemos adquirido cierta experiencia para salvar algún que otro obstáculo.

Uno de los mayores problemas a la hora de hacer prácticas informáticas en la Universidad y también en los Institutos, es el elevado número de alumnos que hay en cada grupo, en relación con la capacidad de las aulas de informática. Así, llevar a un grupo de 100 alumnos a un aula con 30 puestos de ordenador, de los cuales después sólo funcionan en la práctica 20, es una tarea imposible. Lo que no es imposible es elaborar unos guiones muy autocontenidos para que un alumno que disponga de ordenador en su casa, que son prácticamente todos, o pueda acudir al aula de informática en algún rato libre, trabaje por su cuenta. La elaboración de este tipo de guiones suele llevar bastante tiempo, pero es un esfuerzo que después se puede amortizar en cursos sucesivos. Es decir, que incluso en situaciones poco favorables siempre es posible hacer algún intento.

No cabe ninguna duda de que la situación ideal es crear asignaturas específicas para hacer Matemáticas con el ordenador. Por otra parte, a medida que pasa el tiempo las dotaciones en equipos informáticos de los centros de enseñanza van mejorando considerablemente.

Sin embargo, tampoco conviene ilusionarse demasiado. La utilización del ordenador como herramienta en Matemáticas no puede sustituir al aprendizaje de las Matemáticas en un sentido más clásico. Si los alumnos no disponen de unos conocimientos mínimos, todos los medios técnicos a nuestro alcance pueden resultar inútiles y, en los últimos tiempos, hay que lamentar que la situación tiende a empeorar por momentos, en la Enseñanza Media y, como

consecuencia, en los primeros cursos de la Enseñanza Universitaria.

6. Recursos en Internet

Tanto para el programa DERIVE como para el programa MATLAB se pueden encontrar en internet diversas direcciones que ofrecen información, ejemplos de programas, instrucciones, etc. La lista que se expone a continuación no es desde luego exhaustiva, lo cual sería simplemente imposible, dado el elevado número de direcciones Internet en las que aparecen referencias a estos programas. Pero sí que puede dar algunas ideas de por dónde empezar.

- `http://www.derive.com`: La página web de la compañía que ha creado DERIVE. Contiene información relativa a nuevas versiones, actualizaciones, etc. Desde aquí se puede acceder a prácticamente toda la información que hay en Internet en relación con DERIVE.

- `http://www.math.duke.edu/modules/materials`: En esta página se pueden encontrar ejemplos de muchas prácticas de diferentes niveles de dificultad y diversos temas de Cálculo fundamentalmente, en muchos casos aparecen preparadas para realizarlas con varios programas: Maple, Mathematica y Matlab, principalmente.

- `http://www.mathworks.com`: La página web de la compañía que ha creado MATLAB. Se puede encontrar información sobre actualizaciones, cuestiones técnicas referenctes a la instalación del programa y muchos enlaces relacionados con el programa.

- `http://texas.math.ttu.edu/~gilliam/m5399/symbolic.html`: Una descripción con muchos ejemplos de la *Matlab Symbolic Toolbox*.

- `http://www.ius.cs.cmu.edu/help/Math/vasc_help_matlab.html` y la dirección
  `http://www.unm.edu/cirt/info/software/apps/matlab.html`: Contienen guías muy detalladas sobre Matlab.

- `http://www.mat.ucm.es/deptos/ma/ich/manual-matlab.html`: Una introducción muy esquemática pero muy completa a Matlab, en español, realizada por el Prof. Uwe Brauer del Departamento de Matemática Aplicada de la Universidad Complutense de Madrid.

- `http://web.mit.edu/18.06/www` y la dirección `http://www.math.columbia.edu/~psorin/linear_ algebra/Matlab/index.html`: Contienen una *toolbox* con comandos de Álgebra Lineal que se pueden utilizar junto con el libro de G. Strang, *Introduction to Linear Algebra*, Segunda Edición, Wellesley-Cambridge Press, Wellesley MA, 1998.

También existe la posibilidad de buscar nosotros mismos más direcciones de interés. Utilizando algún buscador como Altavista, por ejemplo, si queremos buscar algo relacionado con las Series de Fourier con Matlab, podemos poner en la ventana del buscador: `Fourier & Matlab`, y nos aparecerá una extensa lista conteniendo estas dos palabras. Lamentablemente ocurre con los buscadores de Internet que, al ser tan grande la cantidad de información que circula actualemente, mucha de la información es poco útil. Pero con un poco de paciencia se pueden encontrar cosas interesantes.

Referencias

[A] Apostol, T.M. *Análisis Matemático, Segunda Edición* pp. 390-391. Reverté, Barcelona, 1982.

[C] Carrillo, A. y Llamas, I. *DERIVE: Aplicaciones Matemáticas para PC*. Ra-ma, Madrid, 1994

[Ca] Castro Chadid, I. *Cómo hacer Matemáticas con DERIVE*. Reverté Colombiana, Bogotá, 1992.

[Ch] Chen, K.; Giblin, P. e Irving, A. *Mathematical Explorations with MATLAB*. Cambridge University Press, Cambridge, 1999.

[H] Harman, Thomas L.;Dabney, J. y Richert, N. *Advanced Engineering Mathematics using MATLAB V.4*. PWS, Boston, 1997.

[P] Paulogorrán, C. y Pérez, C. *Cálculo Matemático con DERIVE para PC*. Ra-ma, Madrid, 1994.

[S] Sanz, P.; Vázquez, F.J.; Ortega, P. *Problemas de Álgebra Lineal. Cuestiones, ejercicios y tratamiento en DERIVE*. Prentice Hall, Madrid, 1998.

[Si] Sigmon, K. *MATLAB Primer*. 1989. (Apuntes clásicos sobre Matlab, algo anticuados, se pueden conseguir en la dirección `http://www.mathworks.com` también se pueden conseguir mediante anonymous ftp de `ftp://math.ufl.edu`, directorio `pub/matlab`, fichero `primer35.tex`)

[V] Varios Autores. *MATLAB Edición de estudiante. Versión 4. Guía de usuario*. The Mathworks Inc.-Prentice Hall, Madrid, 1998. (Esta es, de momento, la última edición en español de esta guía).