Programas informáticos en matemáticas

Sección a cargo de

Emilio Bujalance

MatLab en el Cálculo Científico

por

Carlos Moreno

Matlab

MatLab[1] es un entorno orientado al Cálculo Científico que incorpora una extensa biblioteca de funciones capaces de llevar a cabo refinadas tareas de cálculo numérico y con capacidades gráficas muy notables. Está formado por un núcleo común que puede ser completado para determinados fines añadiendo *cajas de herramientas* (toolboxes), formadas por funciones que realizan cálculos o visualizan datos de un modo especializado. Aunque en su núcleo no se pueden realizar cálculos simbólicos, es posible añadir cajas de herramientas que incorporen el núcleo y las bibliotecas de Maple.

Su uso se ha extendido rápidamente entre científicos e ingenieros. La incorporación de este entorno a la enseñanza y formación de científicos e ingenieros es ya importante actualmente.

Matlab frente a otros entornos de cálculo

Resulta clara la conveniencia de ayudarse de entornos matemáticos de cálculo no solo en simples tareas rutinarias de cálculo sino también en cuestiones más complejas que puedan surgir en el curso de la investigación científica. Lo que si puede resultar menos evidente es la elección del entorno más adecuado a nuestras necesidades de cálculo. Esta conveniencia tiene mucho que ver con el tipo de cálculos que se pretendan realizar. Nuestro interés puede ser puramente didáctico. Por ejemplo, ilustrar un concepto o un procedimiento mediante un cálculo que podamos realizar mediante técnicas algebraicas o analíticas. O en el otro extremo, nuestro interés puede estar en la resolución de un problema real planteado sobre un modelo matemático de alta complejidad.

Evidentemente, las razones que llevan a uno a escoger el modo de realizar un determinado tipo de cálculo pueden ser muy variadas pero sin duda las más relevantes son: el volumen de los cálculos, la disponibilidad de procedimientos

[1] MatLab es un producto desarrollado por MathWorks, Inc.

analíticos o numéricos para realizar el cálculo, la disponibilidad de medios de cálculo, etc. Imaginemos la siguiente situación: Se desea calcular los símbolos de Christoffel de un sistema de coordenadas definido por relaciones analíticas con un sistema cartesiano tridimensional. Si esta necesidad se nos presenta en el aula cuando tratamos de explicar la derivación en coordenadas que no son cartesianas, seguramente lo más razonable es limitarnos a escoger un sistema de coordenadas ortogonal, en el que los cálculos se reducen drásticamente y proceder a realizar las cuentas a mano. Si no tenemos esta elección y nos parece fatigante el cálculo de los 18 símbolos de Christoffel de un sistema de coordenadas que no es ortogonal y además tenemos acceso a un entorno de cálculo orientado al cálculo simbólico (Maple, Mathematica, etc.) lo razonable sería programar el cálculo en este entorno. De este modo, podríamos obtener la expresión analítica de los coeficientes buscados. Mucho más compleja sería la situación si realmente los símbolos de Christoffel aparecen como coeficientes en una ecuación en derivadas parciales que debemos resolver en un dominio irregular del espacio. Las posibilidades de poder resolver analíticamente la ecuación serían muy escasas. En tales circunstancias, sería necesario recurrir a métodos numéricos que precisarían del conocimiento numérico de los símbolos en un conjunto finitos de puntos, en cuyo caso se nos plantearía la duda de que es más eficiente, si calcular analíticamente los símbolos y posteriormente evaluarlos o calcular directamente los valores numéricos de los símbolos por métodos numéricos. Es muy posible que nuestro análisis nos llevase a pensar que lo más preciso fuese organizar todos los cálculos en un entorno orientado al cálculo numérico.

Pensemos en una situación más simple: la resolución de un sistema lineal de ecuaciones. Si la dimensión es reducida se podría resolver con rapidez a mano o utilizando un entorno de cálculo simbólico. Lo que si podría resultar menos breve sería intentar resolver un sistema lineal de 1000 incógnitas con la pretensión de mantener la exactitud de los cálculos. La dificultad mayor de usar el cálculo exacto, aún ayudado por un computador, está en que el proceso resultará mucho más lento que en el cálculo aproximado, ya que el cálculo simbólico requiere el análisis de una multitud de situaciones distintas mientras que el cálculo aproximado, resulta más sistemático.

En todo caso un criterio que determina nuestra decisión de orientar el cálculo en un sentido o en otro es la disponibilidad de procedimientos analíticos o numéricos para realizarlo. En el Cálculo Científico[2], los modelos matemáticos que se utilizan actualmente para representar fenómenos físicos o sociales de un modo real no son en general cuantitativamente abordables por técnicas analíticas y requieren posiblemente la realización de millones de cálculos en

[2]Aunque el sentido más interdisciplinar de la ciencia actual hace que los intentos precisos de delimitar un campo sean difíciles, el siguiente comentario ayuda a comprender lo que comúnmente se entiende como Cálculo Científico: En 1971, G. Birkhoff lo definía como *el arte de construir y explorar modelos adecuados de problemas que provienen de las Ciencias Naturales y la Ingeniería, utilizando la intuición física, teoremas matemáticos, algoritmos y la tecnología de los ordenadores* (cita recogida de [3]).

un tiempo muy reducido. El hecho de que los problemas en el Cálculo Científico estén impuestos y no sean manipulables en orden a buscar soluciones sencillas, hace que las posibilidades de calcular de un modo exacto sean muy escasas.

Muchos de los entornos de cálculo orientados al cálculo simbólico incorporan también facilidades numéricas eficientes que permiten aumentar su capacidad para resolver problemas reales. MatLab es un código orientado al cálculo numérico, pensado para resolver problemas reales en diversos campos científicos y tecnológicos. La diferencia con otros entornos puede ser inapreciable si se pretende resolver problemas sencillos pero puede ser clara si la complejidad computacional del problema a resolver es alta. Si la eficiencia del operador que resuelve sistemas lineales nos preocupa o consideramos importante el concepto de densidad de ceros (sparsity), seguramente la elección de MatLab es adecuada ya que su uso de operadores y funciones matriciales así su especialización para matrices huecas (dispersas) es, a mi juicio, excelente.

Matlab frente a los lenguajes de programación tradicionales

A mediados de los años cincuenta, un equipo de investigación de IBM diseñó el lenguaje Fortran como lenguaje orientado al Cálculo Científico que evitaba la dureza de la programación en lenguaje ensamblador. Durante varias décadas el Fortran ha sido el lenguaje de programación más usado en el Cálculo Científico. No obstante, la evolución del Fortran ha sido lenta y en forma de dialectos incompatibles entre sí, debido al retraso en la definición de estándares. Además, ha tardado en incorporar conceptos esenciales que la teoría de la programación ha generado durante las décadas de los setenta y ochenta. En el ámbito del Cálculo Científico otros lenguajes, fundamentalmente el lenguaje C (C++), han ido parcialmente sustituyendo al Fortran.

Por otra parte, en las últimas décadas se ha producido un fenómeno destacable: la aparición de grandes códigos de cálculo orientados a distintas áreas de la ingeniería, capaces de resolver problemas de modo muy simple para el científico o el ingeniero, ya que estos códigos incorporan interfaces de usuario que evitan la necesidad de programar. Estos códigos son verdaderas *cajas negras* donde el usuario solo puede manipular datos y resultados, pero en los que muchas veces ignora completamente las técnicas de cálculo utilizadas.

En un nivel intermedio, se sitúan los *entornos de cálculo*, como MatLab, que contienen agradables interfaces de usuario pero al mismo tiempo poseen lenguajes de alto nivel con un número elevado de procedimientos preprogramados que ponen en práctica las técnicas básicas del cálculo numérico y simbólico.

La elección entre la programación directa en los lenguajes tradicionales, el uso de códigos cerrados y el uso de entornos intermedios tales como MatLab, a la hora de realizar un cálculo, está muchas veces condicionada por nuestra inercia a cambiar de hábitos y su resultado posiblemente es con frecuencia poco eficiente. Podría llegar a ser y en alguna medida así es, que debido a la inevitable especialización de la técnica y el conocimiento, el uso de los lenguajes de bajo nivel se sitúe esencialmente en el mundo de la Informática, los entornos intermedios en el mundo de las Matemáticas y del Cálculo Científico y los

códigos cerrados en el mundo de la Ingeniería y Ciencias Aplicadas. Aquellos que han utilizado el cálculo automático en la década de los setenta, recuerdan con poca nostalgia la programación del método de eliminación de Gauss, el cálculo de escalas para las representaciones gráficas o los esfuerzos para manejar la memoria-vídeo. La imagen del científico que en una clásica película de ciencia-ficción de los cincuenta, destornillador en mano, programaba una *máquina abierta* para calcular la órbita de llegada a la tierra de un platillo volante, indicaba que el abanico del conocimiento necesario para hacer cálculos reales empezaba en el *hardware* básico y terminaba en las Matemáticas. La complejidad de los medios y modos de cálculo ha obligado a recortar objetivos y repartir la tarea entre los que diseñan medios y modos de cálculo y los que diseñan procedimientos que utilizan estos medios y modos. Es obvio que la elección adecuada de nuestros medios de cálculo, evitando tareas innecesarias, puede simplificar el trabajo rutinario en beneficio del trabajo creador. Es importante, a mi juicio, conocer en todo momento las capacidades reales de los entornos de cálculo para evitar malversar nuestros esfuerzos.

EL INTÉRPRETE DE MATLAB

Un computador solo puede ejecutar físicamente a través de su *hardware* un número reducido de instrucciones en su procesador. Por lo tanto, en un lenguaje de nivel superior, es preciso traducir sus instrucciones al lenguaje que el computador pueda comprender directamente. Esta operación puede realizarse de dos modos distintos: Una consistente en traducir el programa de alto nivel previamente a su ejecución, guardarlo como programa de bajo nivel (programa compilado) y después ejecutarlo enteramente. La alternativa es traducir cada instrucción y ejecutarla (programa interpretado). La ventaja de los programas compilados frente a los interpretados es que su ejecución es más rápida si bien para programas de cálculo sencillos esta ventaja pueda ser poco aparente.

MatLab se ejecuta a través de un intérprete. Al iniciar una sesión, se activa una línea de comandos en la que se puede suministrar instrucciones al intérprete, bien directamente mediante un editor similar al que se usa para dialogar con el sistema en MSDOS o en los sistemas UNIX, sin las facilidades hipermedia como las del entorno Mathematica.

En las versiones más recientes (en particular, en la versión 5.3), es posible disponer de traductores que permiten pasar el código MatLab a código en otros lenguajes. Esto permite, por una parte, crear código ejecutable fuera del entorno MatLab y por otra construir ficheros DLL (mex) que pueden ser llamados desde el propio entorno. Ambas posibilidades permiten no solo ocultar el código fuente sino también incrementos notables en la velocidad de ejecución. De este modo, se puede incorporar a MatLab código desarrollado anteriormente en C,C++ o Fortran.

La programación en matlab

MatLab (Matrix Laboratory) tiene a las matrices como objeto de datos básico. Incluso las variables unidimensionales son tratadas como matrices 1×1. Es importante destacar la diferencia que MatLab hace entre matriz y tabla. Obviamente, esta diferencia no existe en la forma de representarlas en memoria pero si en los operadores que actúan sobre ellas. Entre tablas de la misma dimensión están permitidas todas las operaciones aritméticas, lo que se advierte poniendo un punto antes del operador. Las siguientes instrucciones MatLab ponen de manifiesto esta diferencia

```
»A=[1,2;3,-1];B=[0,1;1,0];
»A.*B

ans
0 2
3 0

»A*B

ans
2 1
-1 3
```

El primero de los productos corresponde al producto de tablas, es decir, el cálculo se realiza componente a componente, mientras que el segundo producto corresponde al producto matricial.

En un entorno con un intérprete, la estructura secuencial[3] de un problema con naturaleza vectorial puede suponer una pérdida considerable de tiempo. El adecuado manejo de tablas permite la vectorización de los cálculos de un modo simple y eficiente.

La programación vectorial en MatLab está facilitada por dos conceptos que permiten simplificar las instrucciones de un modo radical: los índices vectoriales o multi-índices y los índices lógicos. Por ejemplo, la expresión MatLab A([2,3,6],[2,4]) representa la submatriz de A formada por las filas y columnas indicadas por los multi-índices [2,3,6] y [2,4]. En la expresión A(A>0), el índice lógico A>0 permite seleccionar los coeficientes positivos de la tabla o matriz.

Por otra parte, además de las tablas o matrices, MatLab dispone de otros objetos de datos tales como las estructuras o las tablas de celdas que permiten empaquetar de datos de un modo simple y facilitar la entrada y salida de argumentos en funciones y ficheros externos.

Aunque el uso de multi-índices e índices lógicos debe reducir considera-

[3] De un modo sencillo, se puede decir que la organización de un cálculo puede ser secuencial, en la que cada etapa de cálculo requiere datos de la anterior o vectorial, en la que las distintas etapas del cálculo se pueden realizar simultáneamente. Por ejemplo, un algoritmo basado en el método de Newton para aproximar las raíces de una ecuación escalar tiene una naturaleza secuencial ineludible.

blemente el uso de las instrucciones de control del flujo del programa, MatLab dispone de las habituales estructuras repetitivas (for, while) y condicionales (if, switch) y algunas no tan usuales con los condicionales al error (try,catch).

En definitiva, el lenguaje de programación que soporta MatLab permite la programación refinada de cualquier tarea que se plantee en el Cálculo Científico a no ser obviamente que requiera un acceso y manipulación fina del sistema. También es destacable que MatLab incluye, en las versiones recientes, elementos de programación orientada al objeto.

GRÁFICOS Y VISUALIZACIÓN CIENTÍFICA

Los principales entornos de cálculo contienen funciones especializadas que permiten representar gráficamente curvas y superficies. Con las instrucciones MatLab

```
»[x,y]=ndgrid(-1:0.05:1);
»z=(x.*y).^3;
»surf(x,y,z)
```

se obtiene la figura 1

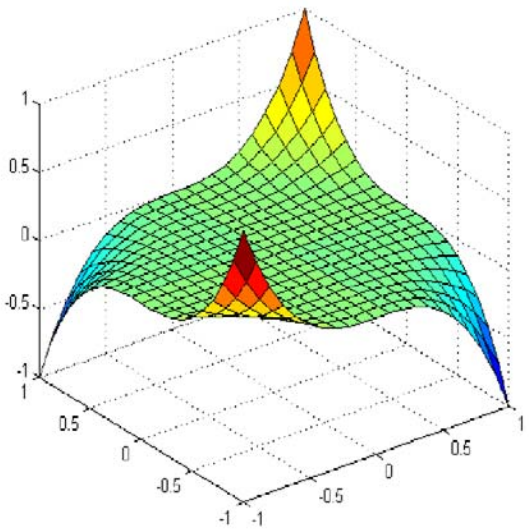

FIGURA 1: Gráfica de $z = (xy)^3$.

La observación de la gráfica nos da rápidamente una idea muy precisa de la forma de la superficie sin necesidad de calcular sus formas fundamentales. Pero no solo son importantes estas capacidades gráficas de un entorno para estas aplicaciones académicas. Muchas veces en el Cálculo Científico es importante comprender la forma de las soluciones de los problemas y observar aspectos

cualitativos del fenómeno que se está simulando. Pero resulta difícil de una enorme lista de resultados numéricos extraer una conclusión cualitativa. Es preciso insistir en que una de las características de los cálculos que se realizan en la simulación de fenómenos reales es que el número de variables numéricas que intervienen es muy elevado. Si para simular la interacción de un fluido con una determinada estructura se pueden necesitar 10 millones de incógnitas, es evidente que la comprensión de los resultados es algo que puede resultar muy complejo.

Geometría computacional, Visualización geométrica por ordenador, son algunos de los nombres que reciben las nuevas campos de trabajo, que estudian los complejos problemas matemáticos, que surgen precisamente en la transferencia de información de la máquina al humano. Un gráfico adecuado puede sustituir a una enorme tabla de números y permitir al científico que observe lo esencial y lo característico de un fenómeno. *Una imagen vale más que cien palabras (cifras)* es un proverbio popular que seguramente tiene mucho sentido para un científico.

Junto a las funciones que permiten directamente construir un gráfico, es importante también valorar las funciones de interpolación que permiten trabajar en ausencia de expresiones analíticas de las curvas o superficies, es decir, mediante conjuntos de datos posiblemente repartidos de un modo irregular. A mi juicio, es destacable la función **griddata** que permite interpolar en una malla regular a partir de un conjunto de datos irregularmente repartidos en el plano usando técnicas de Lagrange y Hermite con triangulaciones de Delaunay.

En otra dirección, es conveniente señalar que MatLab posee cajas de herramientas especializadas en interpolación por esplines, en onditas (wavelets) y redes neuronales. También dispone de una caja de herramientas específica en tratamiento de imágenes.

La visualización simulada del mundo físico permite la experimentación sin costes, la anticipación de un fenómeno sin riesgos o la visualización de una realidad inaccesible al ser humano. Hay dos aspectos notables en la visualización de datos por ordenador: la posibilidad de animación de los gráficos y la posibilidad de interactuar con ellos. Por ejemplo, si se quiere visualizar la difusión de la polución en un lago, sería deseable disponer de una secuencia animada de la evolución de la polución cuando se produce un vertido. Sin duda ello facilitaría la comprensión del fenómeno. MatLab posee notables funciones que permiten la animación gráfica. [4] Se puede obtener una idea precisa de las capacidades de animación gráfica que posee MatLab ejecutando desde la línea de comandos el archivo **demos**.

Las ecuaciones diferenciales en matlab

Hasta la segunda mitad del siglo XX, las posibilidades de calcular las soluciones de un sistema de ecuaciones que modelara razonablemente fenómenos

[4] La versión *estudiante* que se distribuye a bajo precio tiene una limitación en el tamaño de las matrices que impide la animación en figuras con cierta complejidad.

reales, eran muy escasas. De hecho, históricamente esta imposibilidad ha empujado al científico y al ingeniero a su simplificación hasta modelos, muchas veces excesivamente simples. El desarrollo de los medios de cálculo electrónico de las últimas décadas ha impulsado el desarrollo de nuevas técnicas matemáticas y nuevos modos de concebir el cálculo cuya influencia en el avance tecnológico actual ha sido considerable.

El flujo del aire alrededor del ala de un avión, el impacto de un automóvil contra un obstáculo, la regulación del tráfico en una autopista, el crecimiento de especies animales con depredadores o la evolución del precio de una opción en un mercado de derivados financieros son fenómenos de naturaleza muy distinta, que admiten modelos razonablemente realistas basados en sistemas de ecuaciones en derivadas parciales. La simulación numérica de estos fenómenos permite optimizar los diseños y prever problemas potenciales que puedan producirse, sin los costes económicos que la experimentación real (o a escala) origina.

Es importante comprender que en general no es posible resolver mediante métodos exactos estos sistemas de ecuaciones y es preciso recurrir a las técnicas numéricas. Incluso en los casos más simples, en los que es posible encontrar la solución en términos de las funciones elementales o mediante un desarrollo en serie, la calidad numérica de estas soluciones no es necesariamente mejor que la que provee un método aproximado, ya que la evaluación de las funciones elementales o el truncamiento de una serie puede conducir a notables errores de precisión.

Una breve idea de uso de MatLab para la resolución de ecuaciones diferenciales ordinarias la da el siguiente ejercicio extraído de la referencia [5].

Ejercicio 1 *Una barra rígida de $L = 1m$ está suspendida de un extremo. Moviendo el otro extremo, se fuerza a la barra a desplazarse un ángulo igual $\frac{\pi}{6}$. Una vez en esta situación, se suelta el extremo inferior sin impulso, permitiendo que la barra oscile libremente.*

Representar los desplazamientos θ y las velocidades $\dot{\theta}$ angulares de este extremo, calculados mediante la resolución de las ecuaciones del movimiento

$$\ddot{\theta} = -\frac{g}{L}\operatorname{sen}\theta,$$

donde $g = 9.81 m/s^2$ representa la aceleración de la gravedad.

Solución: Se construye una función que calcule el segundo miembro del sistema de ecuaciones diferenciales de primer orden

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{g}{L}\operatorname{sen} x_1. \end{aligned}$$

Esta función podría estar diseñada como la siguiente

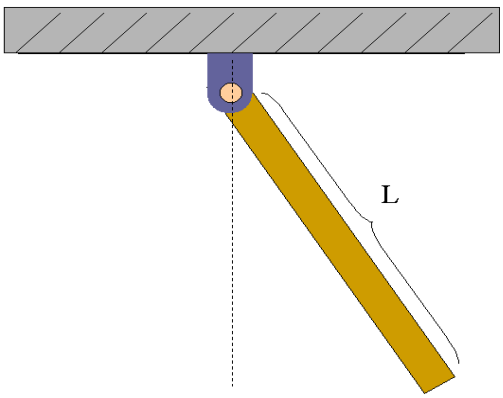

Figura 2: Péndulo

```
function dx=fpendulo(t,x)
g=9.81; L=1; dx(1)=x(2);
dx(2)=-g/L*sin(x(1));
dx=dx';
```

Finalmente, la siguiente función MatLab **pendulo** realiza los objetivos pedidos

```
function pendulo
T=5;
inicial=[pi/6,0];
[t,x]=ode45(fpendulo,[0 T],inicial);
subplot(2,1,1);
plot(t,x(:,1),'g');
title('Desplazamiento angular');
ylabel('θ (rad)');xlabel('t');
subplot(2,1,2); plot(t,x(:,2),'r');
title('Velocidad angular');
ylabel('dθ/dt (rad/s)');xlabel('t');
```

Es importante destacar que MatLab dispone de un entorno visual adicional **Simulink** que facilita considerablemte los cálculos y evita la programación y que puede resultar de gran utilidad en áreas como la Ingeniería Eléctrica o Teoría de Control.

MatLab dispone de una caja de herramientas llamada PDE (partial differential equations) que permite resolver ecuaciones en derivadas parciales li-

neales (aunque fácilmente se puede extender su uso a problemas no-lineales simples). Incorpora una interfaz gráfica **pdetool** que permite incorporar directamente los datos de la ecuación y del dominio del plano en la que se plantea. Es posible acomodar la entrada de datos a un problema genérico o a problemas específicos de Mecánica del Continuo, Electromagnetismo, Térmica, etc. La discretización de las variables espaciales se realiza usando el método de los elementos finitos. Desafortunada el único elemento disponible es el de Courant (CST), el cual puede resultar poco eficiente en la resolución de algunas ecuaciones más complejas. Existen cajas de herramientas de libre dominio que permiten incorporar más elementos aunque no son tan eficientes en la construcción de las mallas. En este sentido, es destacable la caja de herramientas CalFem desarrollada por la Universidad de Lund (Suecia).

OTRAS CAJAS DE HERRAMIENTAS(TOOLBOXES)

En el área de Estadística y Cálculo de Probabilidades, además de la caja de herramientas básica **Statistics**, MatLab dispone de otras cajas de herramientas más especializadas como **Financial Time Series**, **GARCH** (simulación, predicción y estimación de parámetros en series temporales con heteroscedasticidad condicional con un modelo compuesto ARMAX/GARCH), **Optimization** y **Financial**. Dada la actualidad en el panorama económico-social español de las stocks options, es oportuno mencionar anecdóticamente que esta última caja contiene la fórmula de Black que permite calcular la solución explícita de la laureada ecuación de Black-Scholes que rige la relación entre el precio de las opciones y de las acciones subyacentes, en el caso de las opciones europeas. En el caso de las americanas, también está incluida la aproximación binomial de Cox.

MatLab dispone de cajas de herramientas que facilitan la comunicación y el intercambio de datos con otras aplicaciones y medios Communications Toolbox, Data Acquisition, Database (que permite importar o exportar datos entre MatLab y las principales bases de datos), Datafeed.

Finalmente, MatLab dispone de otras cajas orientadas a diversas áreas de la ingeniería como **Control System**, **Frequency Domain**, **Identification**, **Model Predictive Control**, **Mapping**, **Mu Analysis and Synthesis**, **Quantized Filtering**, **Robust Control**, **Signal Processing** y **System Identification**.

Bibliografía

[1] D. DUVAL: La automatización del cálculo simbólico, *Mundo científico. La recherche*, N. 174, diciembre, pag. 1064-71, 1996.

[2] K. ERIKSSON, D. ESTEP, P. HANSBO, C. JOHNSON: *Computational differential equations*, Cambridge University Press, 1996.

[3] C.A. HALL, T.A. PORSCHING: *Numerical analysis of partial differential equations*, Prentice-Hall, 1990.

[4] G. LINDFIELD, J. PENNY: *Numerical methods using MatLab*, Ellis Horwood, 1995.

[5] C. Moreno, R. Vélez: *Introducción al Cálculo Científo, usando MatLab*, Ed. U.N.E.D., 1999.

[6] P. Quintela: *Introducción a MatLab y sus aplicaciones*, Editorial Universidad de Santiago de Compostela, 1997.

Carlos Moreno González
Dpto. de Matemática e Informática Aplicadas a la Ingeniería Civil
E.T.S.I. Caminos (Universidad Politécnica de Madrid)
http: www.cm.matcam.upm.es
e-mail: ma11@dumbo.caminos.upm.es