

MÉTODOS MATEMÁTICOS (Curso 2011-2012)

Segundo Curso del Grado en Ingeniería Civil

Departamento de Matemática Aplicada II. Universidad de Sevilla

Lección 1: Introducción a MATLAB y al Análisis Numérico

Índice

1. Aspectos Generales de MATLAB	2
1.1. Acceso a MATLAB	2
1.2. El entorno de trabajo de MATLAB	3
1.3. Formatos numéricos	3
2. Datos en MATLAB	4
2.1. Matrices	4
2.2. Direccionamiento y manipulación de matrices	4
2.3. Operaciones básicas con matrices	5
3. Archivos y Programación en MATLAB	5
3.1. Archivos de instrucciones	6
3.2. Archivos de funciones	6
3.3. Órdenes de control	6
4. Gráficas en MATLAB	7
4.1. Gráficas bidimensionales	8
4.2. Gráficas tridimensionales	8
5. Errores	8
5.1. Errores relativos y errores absolutos	9
5.2. Aritmética exacta y de precisión finita	9
5.3. Errores de redondeo	10
5.4. El formato IEEE de doble precisión	11
6. Condicionamiento y Estabilidad en Análisis Numérico	12
6.1. Condicionamiento de los problemas numéricos	12
6.2. Estabilidad de los métodos numéricos	13
6.3. Cancelación numérica	14
7. Problemas	15

1. Aspectos Generales de MATLAB

El material expuesto en las cuatro primeras secciones de esta lección debe entenderse solamente como una sencilla introducción a aquellos aspectos del programa MATLAB que van a ser utilizados repetidamente durante el curso. Es muy recomendable profundizar en algunos de los temas aquí tratados, especialmente en todo lo relacionado con el manejo de archivos “*.m” y con la programación. Para ello, puede consultarse cualquiera de los textos mencionados en el proyecto docente de la asignatura.

1.1. Acceso a MATLAB

Para empezar a trabajar con MATLAB, debe iniciarse el ordenador siguiendo las correspondientes instrucciones del Centro de Cálculo de la Escuela hasta que aparezca el símbolo `>>` (*prompt*) que nos indica que el programa está a la espera de nuestras instrucciones. Para salir de MATLAB basta teclear **exit** o **quit** y para ejecutar cualquier instrucción la tecla `Return`. Hemos de tener en cuenta que una instrucción termina al cambiar de línea. Si necesitamos escribir más de una línea, debemos poner el símbolo “...” (tres puntos) al final de la misma y continuar en la siguiente. Si lo que queremos es escribir varias instrucciones dentro de la misma línea, basta separarlas por comas.

El cursor se posiciona con las flechas izquierda/derecha `←`, `→` y para borrar caracteres pueden usarse las teclas `Backspace` o `Supr`. Si lo que se desea es borrar toda la línea de edición puede usarse la tecla `Esc`. También son accesibles otras posibilidades de edición en línea (de significado completamente intuitivo) con las teclas `Inicio`, `Fin` o `Insert`. Otra opción muy útil es usar las flechas arriba/abajo `↑`, `↓` para recuperar las órdenes previas. De esta manera, se puede recuperar una línea anterior de órdenes, editarla y ejecutarla revisada. Para limpiar completamente la pantalla se utiliza la orden **clc**.

Conviene precisar que los paréntesis “()” y los corchetes “[]” tienen significados bien distintos en MATLAB. Los primeros se utilizan para evaluar funciones y los segundos para definir vectores o matrices.

El resultado de ejecutar en MATLAB cualquier expresión matemática se guarda, caso de no asignarle ningún nombre, en una variable denominada **ans** (de *answer*) que sale inmediatamente en pantalla y que toma como valor el correspondiente resultado. Si deseamos que el resultado de nuestra operación no aparezca en pantalla, basta teclear al final de la expresión el símbolo “;” (punto y coma).

Ejercicio 1. Para este ejercicio, se recomienda usar la orden **help**.

1. Ejecute las órdenes **clock**, **date** y **calendar**. Interprete las respuestas proporcionadas por MATLAB.
2. Liste las órdenes de MATLAB relacionadas con la palabra cuadrado (*square*) y utilice dicha información para determinar las raíces cuadradas de 1024.
3. Compruebe que MATLAB permite determinar el máximo común divisor (*greatest common divisor*) de dos enteros. Determine el máximo común divisor del par (30,24) y del par (3072,288).

1.2. El entorno de trabajo de MATLAB

El entorno de trabajo de MATLAB lo conforman un cierto número de ventanas, no solamente la ventana de órdenes o la ventana gráfica. Además de estas dos, los principales componentes de dicho entorno son el explorador de caminos de búsqueda (*Path Browser*), el editor y depurador de errores (*Editor/Debugger*) y el visualizador del espacio de trabajo (*Workspace Browser*).

Path Browser. MATLAB puede llamar a una gran variedad de funciones, tanto propias como programadas por los usuarios. Por tanto, es interesante saber cómo busca MATLAB cualquier función que se le pida que ejecute. La clave es el camino de búsqueda (*search path*) que el programa utiliza cuando encuentra el nombre de una función. El *search path* es una lista de directorios que se puede ver y modificar mediante la orden **path**, o utilizando el *Path Browser* (submenú *Set Path* en el menú *File*).

El directorio actual. El concepto de directorio actual o de trabajo (*Current Directory*) es crucial en MATLAB. Es el directorio donde el usuario debe guardar los diferentes archivos que genere en las sesiones. El contenido de dicho directorio puede obtenerse con la orden **dir**. Para cambiar el directorio actual se utiliza la orden **cd** (*Change Directory*) seguido del nombre del nuevo directorio. Ejecutando **cd ..**, se sube un nivel en la jerarquía de directorios. Estos cambios también pueden hacerse de un modo gráfico.

Editor/Debugger. En MATLAB tienen particular importancia los M-archivos, esto es, archivos con la extensión “*.m”, los cuales son archivos de texto ASCII que contienen un cierto conjunto de órdenes de MATLAB. La importancia de estos archivos es que al teclear su nombre en la línea de órdenes de MATLAB y pulsar Return, se ejecutan todas las órdenes contenidas en dicho archivo.

MATLAB dispone de un editor propio que permite tanto crear y modificar estos archivos (proceso de edición-*Editor*), como ejecutarlos paso a paso para detectar errores (proceso de depuración-*Debugger*).

En un nivel mucho más avanzado, no está de más saber que MATLAB permite además optimizar los programas diseñados mediante un análisis detallado del tiempo de ejecución de cada orden de dichos programas (*Matlab Profiler*).

Workspace Browser. El espacio de trabajo (*Workspace*) de MATLAB es el conjunto de variables que en un determinado momento están definidas en la memoria del programa. Para obtener información sobre el *Workspace* se pueden utilizar las órdenes **who** y **whos**. La segunda proporciona una información más detallada que la primera.

1.3. Formatos numéricos

Para visualizar los resultados, MATLAB ofrece numerosas posibilidades aunque, por defecto, representa los números en pantalla con redondeo a cuatro cifras decimales. También decide si representa un número en notación convencional (coma fija) o en notación científica (coma flotante). La orden básica para la representación en pantalla es **format**.

Es fundamental entender que MATLAB no cambia la representación interna de un número cuando se escogen diferentes formatos, sólo modifica su visualización.

Ejercicio 2. Escriba el número π en todos los formatos que proporciona MATLAB, excepto

el hexadecimal.

2. Datos en MATLAB

2.1. Matrices

En MATLAB se trabaja fundamentalmente con matrices. De hecho, para MATLAB, los números son simplemente matrices cuadradas de orden uno. Las matrices pueden definirse de diversas maneras. Las dos más usuales son:

- Escribir la matriz entre corchetes, colocando las filas una a continuación de otra, separadas por el símbolo ";". Entre los elementos de una misma fila podemos colocar una coma o dejar un espacio en blanco.
- Escribir la matriz entre corchetes, colocando cada fila en un renglón distinto.

MATLAB incluye una orden muy útil para generar vectores cuyas coordenadas están en progresión aritmética. En concreto, la estructura **a:b:c** crea un vector entre los números a y c , incrementando cada coordenada con el número b . Si sólo se escribe **a:c** se considera que b es igual a uno.

Ejercicio 3. Genere tres vectores cuyos elementos representen una partición del intervalo $[-1,1]$ en cinco, ocho y diez subintervalos iguales. Con las tres primeras coordenadas de cada uno de ellos, genere las tres filas de una matriz 3×3 y calcule el determinante de dicha matriz y de su traspuesta.

2.2. Direccionamiento y manipulación de matrices

Para seleccionar un elemento determinado de una matriz se escribe el nombre de la matriz seguido del número de fila y columna separados por una coma y entre paréntesis.

Si se desea extraer una submatriz, basta colocar en vez de números, vectores cuyas componentes son los números de las correspondientes filas y columnas. El símbolo dos puntos es muy útil para crear submatrices. Cuando no se le dan valores a derecha e izquierda recorre, por defecto, todas las filas o columnas. Si colocamos datos fuera del rango actual de una matriz se rellenan con ceros las zonas no especificadas.

Ejercicio 4. Obtenga de cuatro maneras distintas la submatriz formada por la segunda y la tercera fila de la siguiente matriz

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}.$$

2.3. Operaciones básicas con matrices

Para trabajar con matrices y vectores, MATLAB cuenta con una serie de operaciones básicas que citamos a continuación. En todas ellas es fundamental el que las dimensiones sean las adecuadas.

- El símbolo “+” para sumar matrices y el “-” para restar matrices.
- El símbolo “*” para multiplicar matrices. Si el símbolo lo precedemos de un punto se obtiene la multiplicación coordenada a coordenada.
- El símbolo “^” para la la potenciación de matrices. Con el punto delante se obtiene la operación coordenada a coordenada.
- El símbolo “./” para dividir dos matrices coordenada a coordenada. Cuando una de ellas es un número puede quitarse el punto.
- Funciones elementales sobre vectores/matrices (de significado completamente intuitivo en inglés): **max**, **min**, **sort**, **sum**, **size**,...

Además, MATLAB incorpora funciones que permiten generar matrices que surgen con frecuencia en los cálculos: **eye**, **zeros**, **ones**, **diag**, **rand**, **randn**, ...

Ejercicio 5. Defina la matriz cuadrada de orden quince tal que todos sus elementos son nulos, salvo la primera fila y la primera columna, las cuales toman los valores, respectivamente, de la columna o fila en la que se encuentran.

Ejercicio 6. Escriba las matrices A y B definidas por

$$\begin{aligned} A(i, j) &= 10(i - j) + 1; \quad i, j = 1, \dots, 10. \\ B(i, j) &= \begin{cases} 1, & i - j = 1 \\ 0, & \text{en otro caso} \end{cases}, \quad i, j = 1, \dots, 40. \end{aligned}$$

Determine la diferencia entre almacenar la matriz B en formato usual (*full format*) o en formato disperso (*sparse format*).

3. Archivos y Programación en MATLAB

Tanto para trabajar con datos de cierto tamaño, como para diseñar nuevas funciones en MATLAB, es completamente imprescindible trabajar con M-archivos (archivos ASCII con extensión “*.m”). Una parte importante de cada sesión con MATLAB es crear y refinar este tipo de archivos. Atendiendo a su uso, los M-archivos suelen dividirse en dos grandes grupos: archivos de instrucciones o tipo *script* y archivos de funciones.

3.1. Archivos de instrucciones

Un M-archivo de este tipo consiste en una sucesión de instrucciones de MATLAB. Para ejecutarlas y ver el correspondiente resultado en pantalla, basta escribir el nombre del archivo (sin la extensión) y pulsar `Return`. Las variables en un archivo de instrucciones son globales y, por tanto, pueden afectar a los valores de las variables que se hayan creado durante la sesión de trabajo con MATLAB.

Los archivos de instrucciones son utilizados, por ejemplo, para introducir datos en matrices de grandes dimensiones, pues en un archivo de este tipo es fácil corregir errores sin repetir todo el trabajo.

Ejercicio 7. Obtenga la matriz cuadrada de orden veinte tal que los elementos de su diagonal son todos iguales a 3 y las dos subdiagonales principales están formadas por unos. Calcule su determinante. Posteriormente cambie la diagonal por el vector cuyas coordenadas son los primeros veinte números naturales y vuelva a calcular el determinante de la nueva matriz.

3.2. Archivos de funciones

Los M-archivos de funciones son los que permiten incrementar la colección de funciones que ejecuta MATLAB. Es decir, se pueden crear funciones específicas para algún problema concreto y, a partir de su introducción, dichas funciones tienen el mismo rango que las funciones del sistema y se ejecutan de igual forma. Las variables en los archivos de funciones son locales, es decir, no afectan a los valores de las variables que se hayan creado durante la sesión de trabajo con MATLAB.

Se aconseja que el nombre de un archivo de función sea el nombre de la función seguido, obviamente, de la extensión “*.m”. La primera línea de un archivo de este tipo debe ser como sigue:

```
function [argumentos de salida]=nombre de la función(argumentos de entrada).
```

A continuación, puede haber diversas líneas de comentario que han de estar precedidas necesariamente por el símbolo “%”. Conviene decir que son precisamente estas líneas las que aparecerán en pantalla al usar la orden `help`. Finalmente aparece el programa, esto es, las instrucciones necesarias para poder evaluar la función.

Tanto los argumentos de entrada como los de salida no son obligatorios y, si no aparecen, no hace falta escribir los correspondientes corchetes o paréntesis.

Ejercicio 8. Diseñe una función que devuelva el producto escalar de dos vectores x e y de \mathbb{R}^n . Los argumentos de entrada deben ser los vectores x e y . Además, el correspondiente archivo debe incluir algunas líneas de comentario.

3.3. Órdenes de control

Como cualquier lenguaje de programación, MATLAB dispone de instrucciones de control para realizar (o romper) bifurcaciones, repeticiones y bucles. Las bifurcaciones permiten realizar distintas operaciones, según se cumpla o no una determinada condición lógica. Para su diseño, MATLAB incorpora los órdenes `if` y `switch`. Por otro lado, los bucles y repeticiones, permiten hacer las mismas o análogas operaciones sobre datos distintos; regidos por una cierta condición lógica en el caso de los bucles y recorriendo unos ciertos valores previamente

datos, en el caso de las repeticiones. Para generar repeticiones, MATLAB dispone de la orden **for** y para generar bucles de la orden **while**.

La orden for. La sintaxis para la utilización de esta orden de control es

$$\left\{ \begin{array}{l} \text{for "variable"="vector"} \\ \text{"instrucciones sobre la variable"} \\ \text{end} \end{array} \right.$$

El significado es el siguiente: mientras la "variable" recorre los valores del "vector", se realizan las "instrucciones" descritas, con la "variable" tomando dichos valores. MATLAB permite anidar varias órdenes **for**.

Ejercicio 9. Dada una matriz cuadrada de orden n , diseñe una función, usando la instrucción **for**, que sume los elementos de mayor módulo de cada una de las columnas de dicha matriz.

La orden if. La sintaxis habitual para la utilización de esta orden de control es

$$\left\{ \begin{array}{l} \text{if "relación lógica } P_1" \\ \text{"instrucciones } Q_1" \\ \text{else} \\ \text{"instrucciones } Q_2" \\ \text{end} \end{array} \right.$$

El significado es el siguiente: si P_1 es cierto se ejecutan las instrucciones Q_1 y si P_1 es falso se ejecutan las instrucciones Q_2 . Las líneas tres y cuatro anteriores pueden suprimirse y, en este caso, cuándo P_1 sea falso, no se ejecuta ninguna instrucción.

Ejercicio 10. Diseñe una función que calcule todos los divisores de un número natural dado.

La orden while. La sintaxis para la utilización de esta orden de control es

$$\left\{ \begin{array}{l} \text{while "relación lógica"} \\ \text{"instrucciones"} \\ \text{end} \end{array} \right.$$

El significado de este esquema es que las instrucciones se irán ejecutando mientras la "relación lógica" sea cierta.

Ejercicio 11. Un famoso resultado de L. Euler (1707-1783) afirma que $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$. Obtenga el menor número de sumandos de la serie anterior, de modo que la correspondiente suma finita aproxime $\frac{\pi^2}{6}$ con un error menor o igual que 10^{-6} .

4. Gráficas en MATLAB

Para mostrar las correspondientes gráficas, MATLAB abre una nueva ventana, la denominada ventana de figura. Si ya hubiera una ventana de figura, se borra la ventana de figura actual y se dibuja en ella la nueva gráfica. Para utilizar dos o más gráficas en diferentes ventanas de figura, se usa la orden **figure**. La orden **figure(n)** muestra, o crea si no la hay, la ventana de figura n -ésima y ésta pasa a ser la ventana de figura activa. La orden **close** cierra la ventana gráfica activa.

4.1. Gráficas bidimensionales

Para obtener gráficas 2-D, MATLAB admite cuatro opciones: gráficas en coordenadas cartesianas, gráficas en coordenadas polares, gráficas de barras y gráficas de escaleras. La orden para representar datos bidimensionales en coordenadas cartesianas es **plot**, para crear gráficas en coordenadas polares es **polar** y, finalmente, los gráficos de barras y escaleras se generan usando las ordenes **bar** y **stairs**, respectivamente.

La orden **plot** escala los ejes para ajustar los datos, representa los puntos y, a continuación, conecta los puntos con una línea recta. También añade una escala numérica y coloca de forma automática marcas en ambos ejes.

Ejercicio 12. Dibuje la gráfica de la función exponencial en el intervalo $[-2, 2]$. Obtenga una segunda gráfica donde a la curva anterior se le añada la recta tangente en $x = 0$.

Ejercicio 13. Diseñe una función que muestre la gráfica de la función $y = x^n$ en $[-2, 2] \times [-4, 12]$ y cuyo argumento de entrada sea el número n . La gráfica debe tener un mallado y deben situarse los ejes de coordenadas.

4.2. Gráficas tridimensionales

Para obtener gráficas 3-D, MATLAB admite tres opciones: gráficas de líneas, gráficas de superficies y gráficas de contorno. La orden básica para realizar gráficas de líneas es **plot3**, las órdenes para gráficas de superficies son **mesh** y **surf** y, finalmente, para gráficas de contorno es **contour**.

La función **plot3** es análoga a su homóloga bidimensional **plot**. Su forma más sencilla es **plot3(x,y,z)** la cual dibuja una línea que une los puntos $(x(1), y(1), z(1))$, $(x(2), y(2), z(2))$, $(x(3), y(3), z(3))$, etc, y la proyecta sobre un plano para poderla representar en pantalla.

Ejercicio 14. Dibuje en verde un trozo de una espiral cilíndrica.

Ejercicio 15. Dibuje con la orden **mesh** la superficie $z = x^2 - y^2$ en $[-2, 2] \times [-2, 2]$. Posteriormente, represente la superficie en paramétricas

$$x = 4\cos(r)\sec(t), \quad y = 2\sin(r)\sec(t), \quad z = \tan(t),$$

donde $t \in [-\pi, \pi]$ y $r \in [-2\pi, 2\pi]$.

5. Errores

En la modelización computacional, los errores pueden proceder de muy diversas fuentes. Por un lado, hay errores inherentes a dicha modelización. Por ejemplo, aquellos errores debidos a las hipótesis realizadas para convertir un modelo real en un modelo matemático susceptible de tratamiento computacional. Por otro lado, están los errores propiamente computacionales que surgen al implementar los distintos métodos matemáticos. En este curso, nos fijaremos exclusivamente en este último tipo de errores.

5.1. Errores relativos y errores absolutos

Resolver un problema numérico es casi sinónimo de obtener una cierta aproximación a un cierto valor numérico x_T . Si x_A es una de esas aproximaciones al valor x_T , se define el error absoluto como

$$\text{Abs}(x_A) = |x_T - x_A|.$$

Para muchos propósitos, sin embargo, es preferible considerar el porcentaje de error en x_A o error relativo

$$\text{Rel}(x_A) = \frac{|x_T - x_A|}{|x_T|}, \quad (\text{se supone que } x_T \neq 0).$$

En términos generales, la utilidad de manejar el error relativo radica en que nos protege contra afirmaciones precipitadas sobre la bondad de una aproximación, sobre todo cuando nos movemos en escalas extremas (números muy grandes o muy pequeños).

El concepto de error relativo está relacionado con la representación decimal a través de la noción de dígitos significativos (i. e., el primero no nulo y todos los dígitos sucesivos) correctos. En concreto, puede afirmarse que si

$$\text{Rel}(x_A) \leq 0,5 \times 10^{-m},$$

entonces, x_A comparte $m \in \mathbb{N}$ dígitos significativos correctos con x_T , después de redondear ambos valores a m cifras significativas.

Ejemplo 1. Consideremos el problema de aproximar $z = 0,8886110521 \times 10^7$. Puesto que z es realmente grande, incluso aproximaciones intuitivamente buenas, dan lugar a errores absolutos apreciables. Por ejemplo, si $w = 0,8886110517 \times 10^7$, se tiene que $|z - w| = 4 \times 10^{-3}$, a pesar de que w y z comparten nueve dígitos correctos. Sin embargo, si miramos el error relativo vemos que

$$\left| \frac{z - w}{z} \right| = 0,4501 \times 10^{-9}.$$

Ejercicio 16. Estime razonadamente los seis primeros dígitos significativos correctos de la suma de la serie convergente

$$\sum_{n=1}^{\infty} \frac{3^n n!}{(2n)!}.$$

5.2. Aritmética exacta y de precisión finita

Recordemos que cuando escribimos, por ejemplo 3,1416, lo que estamos realmente representando es el número racional

$$3 + \frac{1}{10} + 4\frac{1}{10^2} + \frac{1}{10^3} + 6\frac{1}{10^4}.$$

Utilizando series numéricas, este proceso conduce al conocido concepto de representación decimal en base β . En concreto, si β es un número natural estrictamente mayor que uno y tomamos $x \in (0, 1]$, siempre existe una sucesión de enteros (d_j) tal que $0 \leq d_j \leq \beta - 1$ y

$$x = \sum_{j=1}^{\infty} d_j \frac{1}{\beta^j}$$

La sucesión (d_j) es esencialmente única y permite, por tanto, introducir la notación

$$(x)_\beta = 0, d_1 d_2 \cdots d_n \cdots$$

Multiplicando adecuadamente, es claro que la restricción al intervalo $(0, 1]$ no es relevante.

Ejercicio 17. Determine la representación decimal en base diez del número cuya representación en base dos es 1011, 101.

Ejercicio 18. Obtenga la representación en base dos del número racional $\frac{1}{7}$.

Puesto que la memoria de cualquier ordenador es finita, es evidente que todos los dígitos de una representación decimal no pueden almacenarse. De hecho, los ordenadores actuales manejan exclusivamente números en coma flotante y con una representación decimal en una cierta base β pero con un número finito t de dígitos. El conjunto de tales números reales suele representarse por $\mathbb{M} = \mathbb{M}(\beta, t)$. En concreto, si $x \in \mathbb{M}$ ($x \neq 0$), se tiene que

$$x = \text{sig}(x) \times \beta^E \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_t}{\beta^t} \right) \equiv \text{sig}(x) \times 0, d_1 d_2 \cdots d_t \times \beta^E,$$

donde $\text{sig}(x)$ denota el signo de x ; β es un número natural mayor o igual que 2 denominado base de la representación; E es un número entero denominado exponente; cada dígito d_i verifica $0 \leq d_i \leq \beta - 1$ con $d_1 \neq 0$; y el grupo de cifras $d_1 d_2 \cdots d_t$ se denomina mantisa de la representación. Evidentemente, el exponente E varía en un cierto intervalo $[E_{\text{mín}}, E_{\text{máx}}]$.

Ejercicio 19. Determine la representación del número 350 en un ordenador basado en $\mathbb{M}(2, 7)$.

5.3. Errores de redondeo

Dado un número real x arbitrario, es casi seguro que x no pertenece a $\mathbb{M} = \mathbb{M}(\beta, t)$. Para salvar esta dificultad, se utiliza el redondeo a t dígitos de x . La idea es simplemente determinar un número en \mathbb{M} donde se alcance la distancia mínima entre x y \mathbb{M} .

En concreto, dado un número real con representación decimal en base β

$$x = \text{sig}(x) \times \beta^E \times 0, d_1 d_2 \cdots d_n \cdots,$$

se define el redondeo de x a t dígitos como el número

$$\text{fl}(x) := \text{sig}(x) \times \beta^E \times \begin{cases} 0, d_1 d_2 \cdots d_t, & \text{si } d_{t+1} < \beta/2 \\ 0, d_1 d_2 \cdots d_t + \beta^{-t}, & \text{si } d_{t+1} \geq \beta/2 \end{cases}$$

Ejemplo 2. Consideremos en notación decimal (base diez) el número $x = 23, 346$. Si queremos representarlo en un ordenador \mathbb{M} con $\beta = 10$ y $t = 3$, debemos redondear. En concreto,

$$23, 346 = +10^2 \times 0,23346 \longrightarrow +10^2 \times 0,233 \mid 46,$$

como $4 < 5$, la representación por redondeo de x en \mathbb{M} será $+10^2 \times 0,233$. Conviene observar que si t fuera 4, la representación sería $+10^2 \times 0,2335$.

Puede observarse que $\text{fl}(x) \in \mathbb{M}(\beta, t)$ siempre que $E \in [E_{\text{mín}}, E_{\text{máx}}]$. Cuando x es demasiado grande ($E > E_{\text{máx}}$) se dice que hay *overflow* y cuando x es demasiado pequeño ($E < E_{\text{mín}}$), que hay *underflow*.

Ejercicio 20. Utilizando la orden **format hex**, compruebe que incluso una expresión en MATLAB tan simple como `'t=0.1'` ya conlleva errores de redondeo.

Desde el punto de vista del error relativo, el proceso de redondeo está controlado por la denominada unidad de redondeo $u := \frac{1}{2}\beta^{1-t}$. En concreto, se tiene que para $x \in \mathbb{R}$,

$$\text{fl}(x) = x(1 + \delta), \quad |\delta| \leq u.$$

El diseño en \mathbb{M} de las operaciones algebraicas básicas (suma, resta, multiplicación y división) también sigue el mismo patrón de control del error relativo cometido. En concreto, dados $x, y \in \mathbb{M}$, se tiene que

$$\text{fl}(x \text{ op } y) = (x \text{ op } y)(1 + \delta), \quad |\delta| \leq u, \quad \text{op} = +, -, /, *$$

Conviene observar que, en general, $x \text{ op } y$ no necesariamente pertenece a \mathbb{M} ; de ahí, el redondeo después de realizar la operación. Puede comprobarse, además, que las leyes habituales de la aritmética exacta, como por ejemplo la propiedad asociativa de la suma, no se verifican, en general, en este contexto de la aritmética de precisión finita.

Ejercicio 21. Usando **format long**, ejecute en MATLAB la expresión $3(4/3 - 1) - 1$ y compruebe que el resultado no es realmente cero.

5.4. El formato IEEE de doble precisión

En el mencionado contexto de aritmética de precisión finita, MATLAB utiliza el denominado formato IEEE de doble precisión. Esto significa que

$$\beta = 2, \quad t = 53, \quad E_{\min} = -1021, \quad E_{\max} = 1024,$$

y se asume siempre que $d_1 = 1$ en la mantisa. En otras palabras, MATLAB maneja números (a parte del cero) que puedan expresarse como

$$x = \pm(1 + f)2^e,$$

donde f es un número racional en $[0, 1)$ representable en binario con 52 dígitos a lo sumo y $-1022 < e < 1023$. Cualquiera de estos números puede almacenarse en una palabra de 64 bits: 52 bits para f , 11 bits para e y 1 bit para el signo. El menor número positivo que maneja MATLAB (es decir, $f = 0, e = -1202$) se denomina *realmin* y el mayor ($e = 1023$ y f cerca de uno) *realmax*.

Ejercicio 22. Ejecute las órdenes **realmin** y **realmax** para conocer el rango real, en binario y en base diez, en el que se mueven los números que maneja MATLAB.

En este contexto, el mencionado fenómeno de *overflow* significa simplemente que se ha obtenido, o se ha generado, un valor superior a **realmax**. Cuando eso ocurre, el formato IEEE asigna a ese resultado un valor excepcional denominado *infinite* (**Inf** en MATLAB). Por otro lado, cuando se produce *underflow* (manejo de valores inferiores a *realmin*), la regla usual es considerar dicho resultado como cero, dando un mensaje de aviso. No obstante, algunas versiones (es el caso de la versión MATLAB 7 y sucesivas) permiten considerar puntualmente ciertos valores inferiores a *realmin* (denominados números denormales) para simular una aproximación a cero menos abrupta. Además, para manejar expresiones/indeterminaciones del tipo $1/\mathbf{Inf}$, $0/0$, $\mathbf{Inf} - \mathbf{Inf}$, ... el formato IEEE tiene otro valor excepcional denominado *Not-a-Number* (**NaN** in MATLAB).

Ejercicio 23. Diseñe una función que devuelva el término n -ésimo de la iteración

$$x_{n+1} = x_n^2 - 2x_n, \quad x_0 = 4.$$

¿Cual es el primer valor de n al que MATLAB asigna el valor NaN?

Además, MATLAB tiene la variable predefinida **esp**, denominada *epsilon* de la máquina o del sistema de coma flotante, estrechamente ligada a la correspondiente unidad de redondeo $u = \frac{1}{2}2^{1-53} = 2^{-53}$. En concreto, el valor **esp** se define como la distancia del número uno al siguiente número mayor que uno y representable en MATLAB.

Ejercicio 24. Determine el valor de la precisión de la máquina usando un bucle **while**. Compare el valor obtenido con la variable predefinida **eps** de MATLAB y compruebe que dicho valor es exactamente $2u$. ¿Qué consecuencias tiene este valor en el manejo de los errores relativos en MATLAB?

6. Condicionamiento y Estabilidad en Análisis Numérico

El objeto del Análisis Numérico es el diseño y estudio de métodos que permitan obtener con cierta precisión soluciones numéricas de problemas matemáticos. Estos métodos resuelven dichos problemas mediante algoritmos, es decir, una lista ordenada de operaciones aritméticas y lógicas que transforman ciertos “datos de entrada” en ciertos “datos de salida”. Además de su correcto funcionamiento en aritmética exacta, la idoneidad de un algoritmo está ligado también a su coste temporal y a su comportamiento respecto a los errores de redondeo.

La implementación real de un algoritmo mediante algún lenguaje de programación tiene en cuenta otros numerosos factores, algunos no propiamente matemáticos como el uso de la memoria del ordenador. En general, estos nuevos factores tienen un indudable sesgo informático y no serán tratados en el presente curso.

6.1. Condicionamiento de los problemas numéricos

Los resultados o datos de salida de los problemas tratados en Análisis Numérico deben (o al menos, deberían) estar perfectamente determinados por los datos de entrada. En la inmensa mayoría de los casos, podemos además asumir que los mencionados resultados dependen continuamente de los datos de entrada. Sin embargo, pese a que lo que pueda pensarse inicialmente, el grado de continuidad entre datos de entrada y resultados es muchas veces realmente bajo. Problemas con tal propiedad se denominan mal condicionados, y si ocurre lo contrario, bien condicionados. La gravedad de un problema mal condicionado reside en que su resolución puede producir soluciones muy dispares en cuanto los datos de entrada cambian un poco; cosa bastante frecuente en las aplicaciones. Conviene volver a subrayar que el mal condicionamiento de un problema no depende del algoritmo utilizado para resolverlo, es una propiedad inherente al problema.

Por ejemplo, el popularizado “efecto mariposa” en la predicción del tiempo atmosférico, no es sino una forma más atractiva de afirmar el mal condicionamiento, para periodos no muy cortos de tiempo, de los modelos matemáticos utilizados en dicha predicción. Otro ejemplo bastante conocido de mal condicionamiento, se aborda en el siguiente ejercicio.

Ejercicio 25. La matriz de Hilbert de orden n (consulte la orden `hilb` de MATLAB) se define como

$$H_n := \left(\frac{1}{i+j-1} \right)_{i,j=1,\dots,n}.$$

1. Obtenga H_5 y H_{10} usando **format long**. ¿Qué detecta?
2. Para $n = 5$ y $n = 10$, calcule los vectores w_5 y w_{10} formados por las sumas de las filas de la correspondientes matrices de Hilbert H_5 y H_{10} . Resuelva analíticamente y en MATLAB (orden barra inclinada), los sistemas de ecuaciones lineales

$$H_5x = w_5, \quad H_{10}x = w_{10}.$$

3. Utilizando la orden **cond**, obtenga una tabla con los números de condición de las quince primeras matrices de Hilbert.
4. ¿Porqué este ejercicio refleja fenómenos de mal condicionamiento?

6.2. Estabilidad de los métodos numéricos

Al ejecutar un programa que implemente un cierto algoritmo y estar, por tanto, en un contexto de aritmética de precisión finita, podemos dar por seguro que casi cada paso que vaya a realizarse conllevará un cierto error. En otras palabras, inevitablemente el error de redondeo se propaga. Sin embargo, en muchos casos, la acumulación de todos esos errores tiene un resultado realmente despreciable en el resultado final desde el punto de vista de la precisión requerida. Aquellos métodos numéricos con esta deseable propiedad se dice que son estables numéricamente y, en caso contrario, inestables.

Es más, en la práctica, suelen darse casi exclusivamente dos situaciones: o bien el error crece muy lentamente y estamos en una situación de estabilidad numérica, o bien se tiene un crecimiento exponencial del error y esto conduce relativamente pronto a resultados realmente disparatados. Conviene mencionar, que esta segunda situación de clara inestabilidad numérica puede habitualmente encauzarse realizando un análisis en profundidad del método numérico elegido.

Ejercicio 26. Considere el problema de evaluar las integrales ($a > 0$)

$$y_n = \int_0^1 \frac{x^n}{a+x} dx, \quad n = 1, 2, \dots,$$

mediante la relación de recurrencia

$$y_{n+1} = \frac{1}{n+1} - ay_n, \quad n \geq 1.$$

1. Diseñe una función en MATLAB que implemente la recurrencia anterior y obtenga una tabla, para $a = 10$, con los valores de las veinte primeras integrales. ¿Hay algún valor absurdo en la tabla?
2. En las hipótesis del apartado anterior, sea \hat{y}_n el valor realmente computado por el ordenador para y_n . Supongamos, además, que se comete algún error al calcular y_1 , pero que el resto de las operaciones se realizan en aritmética exacta. Verifique que

$$\Delta y_n := \hat{y}_n - y_n = (-10)^{n-1}(\hat{y}_1 - y_1), \quad \text{para } n \geq 1.$$

3. A tenor de la identidad anterior, ¿cómo puede explicarse el fenómeno numérico observado en el primer apartado? ¿para qué valores de a debería funcionar correctamente el algoritmo implementado en dicho primer apartado?

Los fenómenos numéricos que hemos estado comentando son algo más que curiosidades matemáticas desde el punto de vista de las aplicaciones. Por ejemplo, durante la guerra del Golfo, en Febrero de 1991, un misil Patriot americano se desvió de su objetivo inicial causando 28 muertos entre las tropas aliadas. El error fue consecuencia, en última instancia, de una pobre gestión de los errores de redondeo generados en el programa principal. Otro ejemplo, debido a una aparición no esperada de *overflow*, originó la explosión del cohete Ariane 5 justo después del despegue en Junio de 1996.

En cualquier caso, es importante subrayar que la evolución de cálculos en aritmética de precisión finita es normalmente predecible y de relativa fácil comprensión. Casos como los mencionados en el párrafo anterior son la excepción, no la norma. De hecho, durante el curso, sólo ocasionalmente tendremos que fijarnos en fenómenos ligados a *roundoff* (errores de redondeo), *overflow* o *underflow*.

6.3. Cancelación numérica

Uno de los fenómenos más habituales relacionados con la inestabilidad numérica es la cancelación numérica. Ésta se produce cuando, durante un cálculo en aritmética de precisión finita, se restan dos números casi idénticos y alguno de ellos con un cierto error previo.

En concreto, sean a y b dos números dados distintos y consideremos la substracción $\hat{x} := \hat{a} - \hat{b}$, donde $\hat{a} = a(1 + \delta_a)$ y $\hat{b} = b(1 + \delta_b)$. Los términos $|\delta_a|$ y $|\delta_b|$ son los errores relativos asociados a las cantidades a y b que queremos restar. Si $x = a - b$, entonces

$$\left| \frac{x - \hat{x}}{x} \right| = \left| \frac{-a\delta_a - b\delta_b}{a - b} \right| \leq \max\{|\delta_a|, |\delta_b|\} \frac{|a| + |b|}{|a - b|}.$$

Es decir, el error relativo en x depende (como es razonable) de los errores relativos en a y en b , pero es amplificado por la cantidad $(|a| + |b|)|a - b|^{-1}$. Luego, si $|a - b| \ll |a| + |b|$, hay riesgos claros de inestabilidad numérica.

Ejercicio 27. Un ejemplo simple, pero importante, donde surge de modo natural la cancelación numérica es la resolución de ecuaciones de segundo grado. En aritmética exacta, el problema de resolver $ax^2 + bx + c = 0$ ($a \neq 0$) es muy conocido y sencillo: hay dos raíces dadas por la fórmula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

1. Usando la fórmula anterior en MATLAB, calcule las correspondientes raíces para

$$a = 1, \quad b = -10^6, \quad c = 1.$$

2. Compare los resultados obtenidos con los proporcionados por la orden **roots**.
3. Describa donde se están produciendo fenómenos de cancelación numérica. ¿Cómo podrían solventarse dichos problemas? (Indicación: considere las relaciones entre raíces y coeficientes del polinomio).

7. Problemas

Problema 1. Compruebe que la sucesión de recurrencia

$$\begin{cases} x_0 = 1, & x_1 = \lambda \\ x_{n+1} = (3 + \lambda)x_n - 3\lambda x_{n-1} \end{cases}$$

permite calcular todas las potencias naturales de un número $\lambda > 0$ arbitrario.

Verifique en MATLAB, con $\lambda = 1/7$, que el algoritmo asociado a la recurrencia anterior es numéricamente inestable. ¿Cuál es el problema? (Indicación: consulte cómo puede resolverse una ecuación en diferencias de coeficientes constantes de segundo orden).

Problema 2. Si $a \in \mathbb{M}$ es un número de gran magnitud, es previsible que a^2 quede fuera del sistema de coma flotante \mathbb{M} . Este hecho supone un problema para el cálculo numérico eficiente de la norma de un vector como $[a, 1]^T$. Sin embargo, observando que

$$\sqrt{x_1^2 + x_2^2} = |x_1| \sqrt{1 + \left|\frac{x_2}{x_1}\right|^2},$$

diseñe un algoritmo que proporcione la norma euclídea de un vector arbitrario y que evite calcular el cuadrado de todas aquellas componentes de tamaño relativamente grande. El argumento de entrada debe ser el vector $x \in \mathbb{R}^n$.

Aplique la función diseñada al vector $[10^4, 10^6, 10^8, 10^{10}]$. Compare el resultado con el proporcionado con la orden **norm** de MATLAB.

Problema 3. Eligiendo adecuadamente valores próximos a cero, muestre en MATLAB que la expresión

$$\frac{1 - \cos(x)}{x^2}$$

puede presentar cancelación numérica. Obtenga una expresión equivalente que elimine este problema.

Problema 4. Diseñe una función en MATLAB que proporcione una gráfica conjunta de las rectas del plano

$$\begin{cases} x + \alpha y = 1 \\ \alpha x + y = 0 \end{cases} \quad 0 < \alpha < 1,$$

resaltando el punto de corte. El argumento de entrada debe ser el valor $\alpha > 0$. Tomando distintos valores de α , analice geoméricamente, a partir de las gráficas que obtenga, el condicionamiento del anterior sistema de ecuaciones lineales respecto a variaciones en el parámetro α .

Problema 5. Explique la salida que proporciona MATLAB al ejecutar los siguientes órdenes

$$t=0.1, \quad n=1:10, \quad e=n/10-n*t.$$

Problema 6. Explique brevemente qué hace cada uno de los tres siguientes conjuntos de instrucciones:

1. `x=1, while 1+x>1, x=x/2, pause(0.02), end.`

2. `x=1, while x+x>x, x=2*x, pause(0.02), end.`

3. `x=1, while x+x>x, x=x/2, pause(0.02), end.`

Problema 7. Considere la función

$$f(x) = \frac{e^x - 1}{x}, \quad x > 0.$$

La manera, digamos natural, de evaluar f en MATLAB sería utilizar la expresión

$$f = (\exp(x) - 1) / x.$$

Compruebe, sin embargo, que este algoritmo sufre de cancelación severa para $|x| \ll 1$. Para solucionar el problema, pruebe a manejar

$$y = \exp(x), \quad f = (y - 1) / \log(y).$$

Problema 8. La serie de potencias para $\text{sen}(x)$ es

$$\text{sen}(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

Considere la función de MATLAB

```
function s=powersin(x)
s=0;t=x;n=1;
while s+t ~s
s=s+t;
t=-x.^2/(n+1)*(n+2)).*t;
n=n+2;
end
```

¿Cuándo termina el bucle *while*? Para $x = \pi/2, 11\pi/2, 21\pi/2, 31\pi/2$, responda a las siguientes cuestiones:

1. ¿Cómo de precisos son los resultados proporcionados por la función?
2. ¿Cuántos términos se han requerido?
3. ¿Cuál es el mayor término de la serie en cada caso?

Problema 9. Supongamos que se quiere evaluar una cierta función suficientemente diferenciable $g : \mathbb{R}^n \rightarrow \mathbb{R}$ en un punto $x = (x_1, \dots, x_n) \in \mathbb{R}^n$. Para tales problemas se define el número de condición, cuando sea posible, respecto a variaciones en una de las variables x_j como

$$\text{Cond}(g, x_j) = \left| \frac{x_j}{g(x)} \frac{\partial g}{\partial x_j}(x) \right|.$$

1. Aplicando la fórmula de Taylor a una cierta función $h : \mathbb{R} \rightarrow \mathbb{R}$, esto es,

$$h(x + \Delta x) - h(x) = h'(x)\Delta x + \frac{h''(x + \theta\Delta x)}{2}(\Delta x)^2, \quad \theta \in (0, 1),$$

muestre que la definición anterior es más que razonable.

2. Considere la media aritmética m de n números reales $\{x_1, \dots, x_n\}$, es decir,

$$m = \frac{1}{n}(x_1 + \dots + x_n).$$

Determine el número de condición de m respecto a perturbaciones en cualquier x_j .

3. ¿Hay alguna diferencia entre manejar solamente números positivos y manejar números arbitrarios?
4. Diseñe una función en MATLAB tal que, dados n números reales $\{x_1, \dots, x_n\}$, una cierta coordenada j y una cierta perturbación en dicha coordenada, proporcione el error relativo generado al calcular la media aritmética de los n números y, además, el error relativo al calcular x_j . Utilizando esta función, proporcione ejemplos en MATLAB que corroboren sus afirmaciones sobre el apartado anterior.

Problema 10. Se desea calcular π usando el método ideado por Arquímedes, esto es, inscribiendo polígonos regulares de n lados en una circunferencia de radio unidad y haciendo tender n hacia infinito. En concreto, denotemos por $f(n)$ a la mitad del perímetro del polígono regular de n lados inscrito en una circunferencia de radio unidad.

1. Demuestre que

$$f(n) = n \operatorname{sen}(\pi/n), \quad n \geq 3,$$

y que su límite es π . Si denotamos $y_k = f(2^{k+1})$ con $k \geq 1$, demuestre que

$$y_{k+1} = 2^{k+2} \sqrt{\frac{1}{2} \left(1 - \sqrt{1 - \left(\frac{y_k}{2^{k+1}} \right)^2} \right)}, \quad k \in \mathbb{N}.$$

2. Utilizando la recurrencia anterior, implemente una función en MATLAB para evaluar π . El argumento de entrada debe ser el número k de la última iterada y_k a obtener. Muestre una tabla de resultados con las primeras 25 entradas. ¿Hay valores extraños? ¿Qué ocurre si seguimos iterando?
3. ¿Era previsible la aparición del fenómeno numérico observado en el apartado dos? ¿cómo podría reformularse la recurrencia anterior para eliminar dicho fenómeno?
4. Repita el apartado dos con las variaciones expuestas en el apartado tres y verifique la bondad numérica de la nueva recurrencia.

Problema 11. Denotemos por \mathcal{F} el conjunto de todos los números en coma flotante del sistema IEEE de doble precisión eliminando los valores excepcionales tales como **Inf**, **Nan**, números denormales, ...

1. ¿Cuántos elementos hay en \mathcal{F} ?
2. ¿Qué tanto por ciento de elementos de \mathcal{F} pertenecen al intervalo $1 \leq x < 2$? ¿y al intervalo $1/64 \leq x < 1/32$?
3. Determine por muestreo aleatorio el tanto por ciento de elementos de \mathcal{F} que satisfacen la relación lógica de MATLAB

$$\mathbf{x}*(1/\mathbf{x})==1.$$