Práctica voluntaria (1er. parcial) 0.2 Puntos — Técnicas Numéricas

Francisco R. Villatoro y Carmen M. García — 14 de Octubre de 2002

ALGUNAS NOCIONES BÁSICAS SOBRE MATLAB

Nota: Presentaremos unas nociones breves sobre MATLAB. Debería introducir en el intérprete de Matlab los comandos que se presentan y comprender por qué dan el resultado que realmente dan. Es fácil.

Matlab es un lenguaje interpretado desarrollado por Moler para facilitar el acceso a las bibliotecas de rutinas numéricas para Fortran, LINPACK y EISPACK con carácter docente y que posteriormente se ha convertido en un producto comercial de "The Math Works Inc.".

Para un estudio más detallado, se recomienda el libro: Lindfield & Penny, "Numerical methods using MATLAB"(Biblioteca: FTG-1/PEN/num, 9 ej.).

1. Notación de matrices y vectores:

```
b = 1:3:7;       %% vectores y matrices
A = [1 3 5; 1 0 1; 5 0 9];      b(3),   A(3,2),
x = A\b,         %%% resolución de sistemas lineales
A(2,:), A(:,1), %% segunda fila y primera columna
B=A+inv(A)*i, %% i=sqrt(-1), manejo de numeros complejos
B', B.',          %% traspuesta conjugada y normal

C=[2.3 4.9; 0.9 3.1];
D=[C ones(size(C)); eye(size(C)) zeros(size(C))],
                %% generacion de matrices a bloques

A+inv(A),    A*inv(A),    A.*inv(A),      %% operaciones basicas
disp(['esto es un vector ',num2str(b),' mal escrito']);
```

2. Constantes básicas:

```
eps, pi, inf, NaN, i,j  %% que son?
```

3. Operaciones básicas:

```
^, *, /, +, -, \, .^, .*        %% que son?
```

4. Sentencias de control básicas:

```
for i=1:n, <sentencias>; end
while <condicion>, <sentencias>; end
if <condicion>, <sentencias>;
        elseif <condicion>,<sentencias>;
        else <sentencias>;
end
```

5. Operadores booleanos (lógicos):

```
==, <=, >=, ~=, <, >            %% que son?

EPS = 1.; while (1+EPS/2 > 1), EPS = EPS/2; end; [eps EPS]
```

6. Funciones básicas (algunas):

```
sqrt(x), abs(x), conj(x), sin(x), log(x), cosh(x), exp(x),
cputime, flops, clock %% Qué hacen?
```

7. Desarrollo de funciones definidas por el usuario:

```
function param-salida = nombre-funcion (param-entrada)
%% mensaje de ayuda al usar help nombre-funcion
cuerpo de la funcion
param-salida = el resultado
```

Por ejemplo,

```
[t,Y] = ode45 ('ricatti',[0,10],0);
plot (t, Y); axis ([0 10 -15 15]);

function z = ricatti(t,y);
% function z = ricatti(t,y);
%   calcula segundo miembro de y' = t + y^2
% Haz 'help riccatti' en este directorio

z = t + y.^2;
```

8. Número de condición de una matriz `rcond`, `cond`. ¿Qué hacen?

PRIMER TEMA: Operaciones en coma flotante en Matlab

Problema práctico 1. Escriba un algoritmo en Matlab para el cálculo de las dos raíces de una ecuación de segundo grado que evite las diferencias cancelativas cuando $b^2 \gg 4\,a\,c$. ¿Puede hacer que el algoritmo funcione para un sistema de ecuaciones cuadráticas, es decir, donde los coeficientes son vectores?

Problema práctico 2. La evaluación de las raíces de un polinomio es, en general, un problema mal condicionado. Para mostrarlo, Wilkinson utilizó el polinomio que lleva su nombre que se define como

$$p_n(x) = \prod_{m=1}^{n} (x - m),$$

(él utilizó $n = 20$). Demuestre empíricamente este resultado utilizando Matlab, es decir, que el cálculo de las raíces de este polinomio está mal condicionado. ¿Cómo estimaría numéricamente con Matlab el número de condición de esta operación (cálculo de sus raíces)? ¿Para qué n el número de condición para errores en el coeficiente x^{n-1} es mayor que 10^{16} y que 10^20? ¿Cómo se comporta el algoritmo de cálculo de raíces de Matlab, `roots`, ante este polinomio para dicho grado? ¿Aparecen raíces complejas conjugadas? ¿Cómo es el error en el módulo de estas raíces?

Problema práctico 3. Matlab permite ver el formato binario (de hecho en hexadecimal) con el que se almacenan los números flotantes (doble precisión en formato IEEE-754) utilizando el comando `format hex`. Veamos un ejemplo, sea el número decimal 0,1,

```
>> format hex
>> 0.1

ans =

   3fb999999999999a
```

Los tres primeros dígitos ($3 \times 4 = 12$ bits) representan el exponente y el resto ($13 \times 4 = 52$ bits) la mantisa. En nuestro caso,

$$(\texttt{3fb})_{16} = (0011\ 1111\ 1011)_2 = 3 \times 256 + 15 \times 16 + 11 = 1019,$$

como podemos comprobar utilizando Matlab

```
>> format short
>> hex2dec('3fb')

ans =

        1019
```

Este número representa el exponente $E = 1019 - 1023 = -4$. La mantisa en binario es

$$M = (\overbrace{0101\ 0101\ \cdots\ 0101}^{12}\ 0110)_2,$$

y vemos que en binario $0{,}1 = (1.\overline{01})_2 \times 2^{-4}$, como muestra Matlab. ¿Cómo se representa $-0{,}1$ y por qué?

Explique detalladamente la representación IEEE-754 (en binario) de los siguientes números: 5, `eps`, `realmin`, `realmin/256`, `-realmax`, `inf` y `exp(-inf)`.

¿Cuál es el resultado de las siguientes operaciones y por qué? `2^1023` , `2*2^1023` , `realmax+5*10^60` , `1+2^-53` , `2^-52*2^-1022` , `2^-52*2^-1023` , `0/(-0)` , `inf/(-inf)` , `inf/(-0)` , `0/(-inf)` , `1^(-inf)` , `atan2(inf,inf)` , `sign(0)` , `sign(-0)` , `0^0` , `NaN^0` , `sin(inf)` y `asin(1.4)`.

Problema práctico 4. La solución de algunas ecuaciones diferenciales, aparentemente inofensivas, puede dispararse (tender a infinito) en un tiempo finito (*blow up*). Estas ecuaciones fueron clasificadas por Painlevé (a finales del siglo XIX) como las que tienen singularidades móviles. Para la resolución numérica de estas ecuaciones el número `inf` en aritmética IEEE-754 puede ser de gran ayuda.

Considere la ecuación de Ricatti,

$$\frac{dy}{dt} = f(t, y) = t + y^2, \qquad y(0) = 0,$$

y su solución en el intervalo $t \in [0, 10]$. La solución de esta ecuación se puede escribir con funciones de Bessel de tipo J, ¿puede calcular dicha solución? Utilizando esta solución se calcula el valor de $y(10) = -7{,}53\cdots$.

Sin embargo, para su solución también podemos utilizar métodos numéricos. Sean $t_n = (n-1)\,\Delta t$, $n = 1, \ldots, N$, donde $\Delta t = 10/(N-1)$, una malla de N puntos del intervalo $[0, 10]$. Consideremos un método numérico general que calcule los valores numéricos $Y(n) \equiv Y(t_n) \approx y(t_n)$ y que opere en Matlab de la siguiente forma

```
Y(1)=y(0); for n=2:N, Y(n)= F(dt, (n-1)*dt, Y(n-1)); end
```

Los diferentes métodos numéricos se caracterizan por una función `F` distinta. Aunque estudiaremos estos métodos más adelante en la asignatura, vamos a presentar tres ejemplos particulares. Por ejemplo, el método en diferencias de Euler hacia adelante o progresivo,

$$F(\Delta t, t, Y) = Y + \Delta t\,(t + Y^2),$$

que es de primer orden, el método de Runge-Kutta tipo Heun,

$$F(\Delta t, t, Y) = Y + (t + Y^2 + t + \Delta t + (t + Y^2)^2)\,\Delta t/2,$$

que es de segundo orden, y, finalmente, el método en diferencias no estándar de tipo racional

$$F(\Delta t, t, Y) = Y + \frac{(t + \Delta t/2 + Y^2)\,\Delta t}{1 - \Delta t\, Y},$$

que también es de segundo orden.

Utilice la función de Matlab `ode45` y los métodos de Euler, Heun y no estándar para resolver esta ecuación de Ricatti. Compare los resultados numéricos que ha obtenido. ¿Cuánto vale $y(10)$ con cada uno de estos métodos?

Justifique en lo posible los resultados. ¿Influye en los resultados algo del formato numérico IEEE-754?

Los métodos numéricos anteriores requieren la suma reiterada de números flotantes. Para minimizar su error se puede utilizar la suma compensada. Escriba varios algoritmos que utilicen suma compensada para calcular $t(n) = n\,\Delta t$ e $Y(n)$. Compare los resultados numéricos que obtiene. Mejoran con respecto a los que ha obtenido previamente. Si es así, ¿por qué cree que es así? Justifique sus respuestas.