CÁLCULO - PRÁCTICA I - INTRODUCCIÓN

PEDRO FORTUNY AYUSO

En esta práctica trataremos rápidamente de la utilización de Matlab.

1. Cuentas básicas

Matlab es un programa que trabaja con *vectores*, no con números. Se puede pensar que trabaja con *listas*, también (es este sentido, es lo mismo una lista que un vector). Esa es la manera natural de utilizarlo.

1.1. **Creación de listas.** Para crear listas hay dos comandos:

`linspace`**:** Con este se genera una lista de números entre dos valores, compuesta por una cantidad de elementos equidistribuidos:

```
linspace(a, b, n)
```

genera una lista de n números equidistribuidos entre a y b, comenzando en a. Pruébese.

`[a:s:b]`**:** Con esta construcción (los corchetes son opcionales) se genera una lista de números que empieza en a, termina en b (o lo más cerca posible antes de b) y va de s en s. Pruébese.

Por ejemplo, para asignar a una variable x el vector de 200 valores entre $-\pi$ y π, equidistribuidos, se escribe:

```
x = linspace(-pi, pi, 200);
```

(¿qué ha hecho el punto y coma?).

Mientras que para asignar a y el vector de valores entre $-\pi$ y π yendo de 0,01 en 0,01, se escribe

```
y = -pi:.01:pi;
```

(¿qué ha hecho el punto y coma?).

Estas dos construcciones son tan comunes que no voy a repetirlas: se darán por supuestas de ahora en adelante.

1.2. **Operaciones elementales.** Asignemos a la variable x el vector de 300 componentes equidistribuidas entre $-\pi$ y π:

```
x = linspace(-pi, pi, 300);
```

y calculemos, por ejemplo, el seno de cada uno de esos valores.

```
y = sin(x);
```

(no se ve nada, por el punto y coma). Si queremos ver el valor de y, quitemos el punto y coma:

```
y
```

aparece una lista de 300 números, cada uno de ellos es el seno del valor de x correspondiente. Para asegurarnos, lo dibujamos:

Fecha: 29 de octubre de 2012.

```
plot(x,y);
```

Debe aparecer una ventana con una gráfica de la función seno. Lo que hemos hecho es *dibujar* (plot) el conjunto de pares (x, y) donde las x se toman del vector `x` y las y se toman del vector `y`.

Realizar los siguientes ejercicios:

(1) Dibujar la gráfica de la función $exp(x)$ con x entre -5 y 5, y con saltos de 0,03 en 0,03.
(2) Dibujar la gráfica de la función $x^2 - 3x + 1$, con x entre -10 y 10 y usando 300 puntos. ¿Qué pasa si se escribe `x^2`? Que da un error.
(3) Dibujar la gráfica de la función $tan(x)cos(x)$ con x entre -2 y 2 con saltos de 0,025. ¿Qué pasa si se escribe `tan(x)*cos(x)`. Que da un error

1.3. **Los operadores con punto (`.`)** Como se debería haber visto en los ejemplos anteriores, *no se puede calcular* `x*x` (el producto de un vector consigo mismo) ni `x^2` (elevar al cuadrado), se produce un error. Esto es porque las operaciones que se hacen elemento a elemento requieren un `.` (punto).

Para multiplicar, dividir y elevar vectores *elemento a elemento* se ha de escribir un punto `.` antes de la operación.

```
a = linspace(-3, 3, 200); b = linspace(-4, 4, 200);
b = a * b ;% esto da un error
y = a .* b;% esto es lo correcto: a × b
u = a / b;% esto da un error
u = a ./ b;% correcto
v = a ^ 2;% error
v = a .^ 2;% correcto
```

Esto ha de quedar claro. Lo usaremos siempre. **SIEMPRE**.

2. Funciones anónimas

Pero es un lío trabajar siempre escribiendo todas las ecuaciones. Supongamos que tenemos que calcular muchas veces (o representar gráficamente) el polinomio $P(x) = 321x^2 - 3x + 1$, por la razón que sea. Lo que no vamos a hacer es repetirlo continuamente. Para eso se definen las *funciones anónimas*:

```
P = @(x) 321.*x.^2 - 3.*x + 1
```

Ahora `P` representa la función $P(x)$, el polinomio del que hemos hablado arriba y se puede hacer muy fácilmente:

- Representar P entre -5 y 5 usando 300 puntos.
- Representar P entre $-\pi$ y 20 con saltos de 0,025.
- Calcular los valores $P(3), P(0), P(2132)$.

Las funciones anónimas pueden ser de una o varias variables. Por ejemplo, $f(x, y) = xy - x^2 + y^3$ (atención a los puntos):

```
f = @(x,y) x.*y - x.^2 + y.^3;
```

Calcular $f(1, 2), f(2, 0), f(-\pi, 7)$ es muy sencillo, ahora.

3. Cálculo simbólico

Cuando se trata de hacer un cálculo simbólico (por ejemplo, límites, derivadas, integrales), es preciso especificarle a Matlab que estamos utilizando una variable simbólica.

Primero de todo: limpiemos los valores de las variables:

```
clear all;
```

(Ahora ni `x` ni `y` ni ninguna de las variables que hemos utilizado antes tienen valor alguno).

Calculemos, por ejemplo, el límite de $\left(1+\frac{1}{n}\right)^n$ cuando n tiende a infinito (¿cuánto es esto?). Intentémoslo sin más:

```
limit((1+1./n).^n,n,inf)
```

(`inf` es el símbolo de infinito en Matlab). Esto da un error porque Matlab *no sabe lo que es* `n`. Hay que indicarle que se trata de una variable simbólica:

```
syms n
```

Y ahora tratamos de repetir la instrucción:

```
limit((1+1./n).^n,n,inf)
```

Debe mostrar la respuesta `exp(1)`, que significa e, claro.

Pero no está restringido a límites en el infinito.

Calcúlense los siguientes:

$$\lim_{x\to 0}\frac{\sin(x)}{x};\quad \lim_{x\to\pi}\frac{(x-\pi)^2}{\tan(x)^2};\quad \lim_{x\to 1}(x-1)\log(1-x)$$

$$\lim_{x\to -\pi}(\cos(x)-1)\tan(x+\pi);\quad \lim_{x\to 0}x^x;\quad \lim_{x\to 3}\frac{x+1}{x-3}.$$

Y, puestos a hacer cálculo simbólico, se pueden también derivar funciones muy sencillamente:

```
P =@(t) cos(t).*sin(t) - t.^3 + exp(t).*t
```

define una función anónima. La variable que se use es indiferente (igual que da lo mismo hablar de $P(t)$ que de $P(x)$ mientras tanto t como x "no signifiquen nada").

Si ahora queremos calcular $P'(u)$ (la derivada de P con respecto a u) utilizamos una variable simbólica u:

```
syms u
```

y no tenemos más que derivar:

```
diff(P(u),u)
```

para calcular la derivada segunda,

```
diff(P(u),u,2)
```

la derivada tercera

```
diff(P(u),u,3)
```

etc.