CLASE 1 DE PL DEL CURSO 2020/21
GRUPO F, MÉTODOS NUMÉRICOS

PEDRO FORTUNY AYUSO

En la primera sesión de prácticas vais a recordar someramente cómo se utiliza Matlab. Para ello, veréis dos vídeos y haréis los ejercicios que indico a continuación (están resueltos pero se trata de que los intentéis hacer vosotros).

En mi página web tenéis una *hoja de consulta rápida* que os aconsejo imprimir (por las dos caras). Ahí tenéis prácticamente todo lo que vais a necesitar en estas prácticas.

Comenzad viendo el primer vídeo introductorio. Abrid Matlab, intentad replicar lo que hago en él y ved el segundo vídeo intentando, también, replicar lo que hago.

Una vez hecho eso, intentad realizar los siguientes ejercicios **en un archivo llamado** `practica1.m`, en la carpeta `Documentos\MATLAB`. Las soluciones indican los comandos que hay que escribir en el archivo (y, para ejecutarlos, se seleccionan y se presiona `F9`).

IMPORTANTE: en esta asignatura **nunca** (repito, **nunca**) se utilizará `syms` ni `ezplot` ni nada que tenga que ver con cálculo simbólico. Todas las gráficas se harán definiendo un vector y con `plot(u, f(u))` o algo similar. Todas las funciones serán "anónimas" (definidas con `@`).

Ejercicio 1. Definir la función `patata` de una variable, que valga

$$patata(t) = e^t - \pi \cos(t^2 + 1).$$

Dibujarla en el intervalo $[-3.15, 3.16]$ utilizando un vector que vaya de centésima en centésima.

Solución. Los siguientes comandos bastan:

```
patata = @(u) exp(u) - pi * cos(u.^2 + 1);
v = [-3.15:.01:3.16];
plot(v, patata(v));
```

Comentario: obsérvese cómo para definir una función de una variable, esta puede tener el nombre que sea: en el enunciado nos dicen t pero, es evidente que puede ser t, u, x... El valor π se escribe `pi`, en Matlab. El valor e no está defininido, hay que poner `exp(1)` para e y `exp(a)` para e^a.

Fecha: 9 de febrero de 2021.

Obsérvese cómo el cuadrado se escribe `u.^2`, con un punto: porque a `patata` se le puede introducir una lista (como al hacer el dibujo) y no queremos que calcule "el cuadrado de una lista" sino la "lista de cuadrados de cada elemento". Siempre será así excepto cuando estemos seguros de que queremos elevar una matriz a una potencia.

Para definir `v`, como nos piden que los elementos de la lista vayan de centésima en centésima, hay que utilizar la expresión con ":". Si nos dijeran el número de puntos, utilizaríamos `linspace`. □

Ejercicio 2 (Importante)**.** Definir una función de una variable que devuelva, para x, el valor $\cos(x)(x) + 2.4$. Dibujarla entre $-e$ y e^2 con 3000 puntos.

Solución. Este ejercicio tiene una complicación importante: la x, en principio es un número, pero MATLAB trabaja (como se ha visto ya) con *listas*. Pero 2.4 es un número y no se puede sumar a una lista. Necesitamos conocer el tamaño de x y generar una lista de unos de dicho tamaño. Para ello usamos `ones`.

```
f = @(x) cos(x).*sin(x) + 2.4*ones(size(x));
u = linspace(-exp(1), exp(2), 3000);
plot(u, f(u));
```

Comentario: obsérvese cómo el producto $\cos(x)(x)$ utiliza `sin(x)` (con "i"; ojo, esto es muy fácil escribirlo con e) y después se escribe `.*` porque queremos multiplicar *elemento a elemento*.

El producto del escalar 2.4 por el vector `ones(size(x))` no necesita punto porque solo hay una manera de hacerlo.

La instrucción `size(x)` devuelve el tamaño (filas, columnas) de `x`, así que `ones(size(x))` define una matriz llena de 1 del mismo tamaño que `x`. □

Ejercicio 3. Definir una función de *dos* variables x, y que devuelva el producto elemento a elemento de los cuadrados de cada variable. Probarla con `x = [0 1 2 3]` y `v = [1 4 -7]`. ¿Puede funcionar?

Solución. Con cuidado:

```
g = @(x,y) x.^2 .* y.^2;
g([0 1 2 3], [1 4 -7]);
```

Al ejecutar la segunda línea debe ocurrir un error. Porque se está intentando multiplicar una lista de cuatro elementos (`[0 1 4 9]`) con una de tres (`[1 16 49]`, lo cual no está definido. El error dice justamente eso: las dimensiones de las "matrices" deben coincidir. □

Ejercicio 4. Se consideran las funciones:

$$f(x) = \cos(3x), g(x) = e^{-2x}, h(x) = \tan(-x).$$

Se pide:

(1) Definirlas en MATLAB.
(2) Dibujarlas sobre la misma gráfica, usando un vector de 1000 puntos entre -2 y 2.
(3) Dibujar, además, sobre la misma gráfica y usando el mismo vector, la función $h(f(x))$.

Solución. Lo único que hay que hacer es ir con cuidado y saber cómo conseguir que un `plot` no borre los anteriores: usando `hold on`. Utilizo variables que no son x para que os acostumbréis.

```
f = @(u) cos(3*u);
g = @(u) exp(-2*u);
h = @(v) tan(-v);
```

Para definir el vector:

```
w = linspace(-2, 2, 1000);
```

(porque nos dicen el número de elementos). Dibujamos las tres. Después del primer dibujo, decimos a MATLAB que no borre la ventana gráfica en los siguientes plots.

```
plot(w, f(w));
hold on
plot(w, g(w));
plot(w, h(w));
```

La última gráfica es igual de sencilla:

```
plot(w, h(f(w));
```

Importante: una vez que se vuelve a definir g, el valor anterior (el del ejercicio previo) se olvida y ahora es la función que se acaba de definir.

Para que MATLAB siga comportándose como ordinariamente (borrando cada dibujo al hacer uno nuevo), `hold off`. □

Correo electrónico: fortunypedro@uniovi.es