CLASE PL DEL 16 DE FEBRERO DE 2026

PEDRO FORTUNY AYUSO

El contenido de la clase de hoy son los ejercicios 12 y 18, los mismos que se hicieron en la anterior, pero resueltos mediante el método de Newton-Raphson. Para poder realizarlo, necesitáis descargar los archivos llamados `newtonraphson.m` y `newtonraphsonP.m` (aunque este segundo solo sirve para "ver" el funcionamiento del método gráficamente; ambos ejercicios pueden resolverse con el primero). Ambos archivos están en mi página web, como "Newton-Raphson" y "Newton-Raphson con plot".

Como ejemplo: tratemos de encontrar la *primera* raíz real de la función

$$f(x) = e^{\cos(x)} - 1.$$

Para ello utilizaremos `newtonraphsonP.m`, que nos permite ver cómo va calculando iterativamente cada punto.

Importante: En teoría, el método de Newton-Raphson ya "conoce" la derivada de la función. **Para usarlo aquí, hay que calcularla**. Cada uno calcula las derivadas como mejor le parece pero deberíais ser capaces de hacerlas a mano (aunque me da igual cómo las calculéis, siempre que *estén bien*). En este caso, por la regla de la cadena:

$$f'(x) = -\sin(x)e^{\cos(x)}.$$

Para utilizar tanto `newtonraphson.m` como `newtonraphsonP.m` necesitamos:

- La función $f(x)$ definida en Matlab.
- La función $f'(x)$ definida en Matlab.
- Una semilla x_0.
- Una tolerancia ϵ.
- Un número máximo de iteraciones N.

Antes de nada, conviene dibujar la función. Se crea un nuevo *script* que llamamos `PL3.m` (por ejemplo), y en él se escriben los comandos que se necesitan para esta práctica. Recuérdese que para ejecutar una serie de comandos de dicho archivo, basta seleccionarlos y presionar F9.

Definimos la función y un intervalo:

```
clf; % limpiamos la pantalla: siempre antes de un nuevo grafico
f = @(x) exp(cos(x)) - 1;
I = linspace(0,10,2000); % por ejemplo
plot(I, f(I));
yline(0); % Dibuja el eje OX en negro
```

La gráfica que aparece será como la de la Figura 1. Como se ve, la primera raíz positiva está cerca de 1.5. Para hacer una prueba, intentemos utilizar el método de Newton-Raphson para encontrar dicha raíz, "a lo bruto", sin pensar. Tomemos $x_0 = 0.1$. Si se ejecutan los comandos necesarios:

```
% primero necesito la derivada como una funcion:
fp = @(x) -sin(x).*exp(cos(x)); % EL PUNTO CON EL PRODUCTO!!!!
x0 = 0.1;
tol = 1e-5; % una tolerancia razonable
```

Fecha: 16 de febrero de 2026.

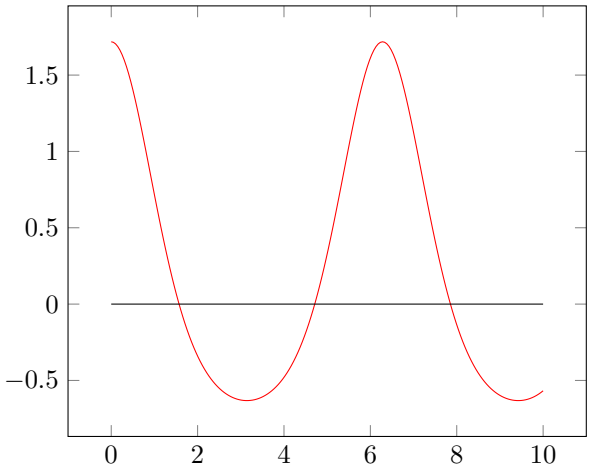

FIGURA 1. La función $f(x) = e^{\cos(x)} - 1$ entre 0 y 10.

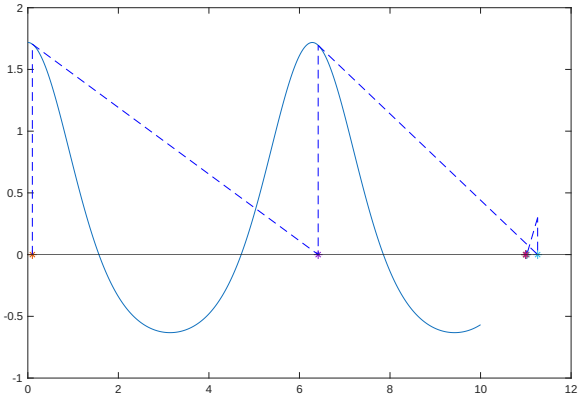

FIGURA 2. Figura tras el comando `newtonraphsonP`.

```
N = 100; % un numero maximo razonable
r = newtonraphsonP(f, fp, x0, tol, N) % no pongo ";" para ver r
```

Debería aparecer una gráfica como la Figura 2. Como se ve, la semilla $x_0 = 0.1$ es **muy mala** para el problema en cuestión: la tangente es demasiado horizontal y el valor de x_1 se ha ido ya más lejos que la segunda raíz, y x_2 aun más, etc. (se va tan lejos que ni siquiera está en el intervalo $[0, 10]$). De hecho, el valor de `r` es

$$r = 10.9956\ldots$$

que es muchísimo más grande que el pedido (que debe de ser menor que 2).

De cualquiera de las dos figuras se deduce que es mucho más útil comenzar con una semilla ya cercana a la raíz buscada. Por ejemplo, $x_0 = 1.5$:

```
clf % limpiamos la pantalla grafica
x0 = 1.5; % dejamos todo lo demas igual
plot(I, f(I));
yline(0)
r = newtonraphsonP(f, fp, x0, tol, N)
```

Que, como se ve, converge muy rápidamente a un punto cercano a la raíz. De hecho, en la línea de comandos se puede ver que:

$$r = 1.5708\ldots$$

que es lo que debe ser, aproximadamente $\pi/2$. Pero **debemos asegurarnos de que es así, esto hay que hacerlo siempre**:

```
f(r)
```

es $3.6 \cdot 10^{-12}$, que es en valor absoluto menor que la tolerancia que hemos exigido de 10^{-5}.

El resto de ejercicios es igual.

Comandos nuevos de hoy (aparte de los de Newton-Raphson):

- `clf`: limpia la pantalla gráfica
- `yline(a)`: dibuja una línea horizontal en altura $y = a$.