CLASE 3 DE PL DEL CURSO 2020/21
GRUPO F, MÉTODOS NUMÉRICOS

PEDRO FORTUNY AYUSO

IMPORTANTE — IMPORTANTE — IMPORTANTE

Hay algún cambio *menor* en el archivo de prácticas de mi página web, que convendría que actualizarais. No pasa nada si no lo hacéis, pero os puede convenir.

En esta clase haréis programas y ejercicios sobre los métodos de bisección y Newton-Raphson. Recordad que

Los programas se graban en `Documentos\MATLAB`

pues si no, no funcionarán.

Vuestro trabajo consistirá en realizar los Ejercicios 10, 11, 12 y 13 (capítulo 2) mis Prácticas de Laboratorio. Quien tenga tiempo o quiera repasar en casa, puede hacer el 14. El *Ejemplo 1* está hecho en detalle para que veáis cómo puede hacerse, pero no hace falta que todos vuestros ejercicios devuelvan un 'warning'.

A continuación resuelvo parte de los planteados, pero solo *para que veáis cómo pueden solucionarse*: lo primario es que trabajéis vosotros.

Solución del ejercicio 10. Asumimos que la entrada es f, a, b, ϵ y N. Para determinar si hay un error, lo que haremos es devolver $-\inf$ si no se llega a la solución aproximada antes de N pasos. *Importante:* se supone que f cambia de signo entre a y b. Si no lo hace, pueden ocurrir cosas sorprendentes.

```
% biseccion(f, a, b, e, N)
% dada f continua, que se supone que cambia de signo entre a y b
%      devuelve un numero r:
% si |f(c)|< e antes de N pasos, entonces c = r
% si no, c = -inf
function [c] = biseccion(f, a, b, e, N)
  i = 1;
  c = (b + a)/2;
  while (abs(f(c)) > e && i < N)
    if (f(c)*f(a) < 0)
      b = c;
    else
      a = c;
    end
    c = (b + a)/2;
    i = i + 1;
  end
  if i >= N
    c = -inf;
```

Fecha: semana del 22 de febrero de 2021.

```
  end
end
```

LISTADO 1. Contenido del archivo `biseccion.m`.

□

Solución del ejercicio 11. Solo vamos a hacer algunos ejemplos. Hay que tener en cuenta que *los intervalos que nos dan no tienen por que cumplir la condición de Bolzano*: antes de resolver un ejercicio, se debe pensar. En este caso, puesto que tenemos Matlab, lo primero que se hace es dibujar la función para ver qué intervalo puede ser "bueno". Indico en cada caso la secuencia de instrucciones que yo usaría. Se supone que el programa `biseccion.m` ya está grabado en el directorio `Documentos\MATLAB`.

(1) Para $f(x) = \cos(e^x)$, con $x \in [0, 2]$, yo haría:

```
x = linspace(0,2,1000);
f = @(u) cos(exp(u));
plot(x, f(x));
```

Se ve en la gráfica que la función *es positiva en los dos extremos*, así que el algoritmo de bisección "a lo loco" no nos sirve. Como no nos especifican que raíz se busca en dicho intervalo (y hay 2), tomamos el que mejor nos parezca. Por ejemplo, $a = 0$, $b = 1$. Tampoco nos especifican la precisión: elegimos una razonable, 10^{-5}. Como tampoco se especifica el número de pasos, para no liarnos, tomamos uno suficiente: $2^n > (a-b)/10^{-5} = 10^5$, así que $n = 17$ nos sirve. Ahora ya seguimos:

```
a = 0;
b = 1;
e = 1e-5;
N = 17;
c = biseccion(f, a, b, e, N)
```

(el último comando sin punto y coma para verlo). Veamos cuánto vale $f(c)$:

```
f(c)
```

debería salir algo menor que 10^{-6}.

Nota: Observad que 10^{-5} se escribe, en notación exponencial como `1e-5`, que significa "uno por 10 elevado a la menos 5". Obsérvese también que no sabemos el número de pasos que se ha llevado a cabo. ¿Cómo podría arreglarse esto con un archivo de función `biseccion2.m`?

(2) Para la segunda función, yo haría

```
x = linspace(0,3,1000);
g = @(t) t.^3 - 2;
plot(x, g(x));
```

Cuidado con el punto antes de la potencia. Se observa que la función tiene signo diferentes en $a = 0$, $b = 3$, así que podemos utilizar estos extremos. Como antes, tomamos $e = 10^{-5}$ y ahora $2^n > 3/10^{-5} = 300000$, con lo que para asegurarnos hemos de tomar $N > 19$ (téngase en cuenta que hacer el número exacto de iteraciones da como resultado el error).

```
a = 0;
b = 3;
e = 1e-5;
N = 20;
c = biseccion(g, a, b, e, N)
```

Cuidado con la g, que es ahora la función. Para comprobar el resultado, calculamos `g(c)`, que debería dar menos que 10^{-5}. Hágase este ejercicio con $N = 10$, a ver qué ocurre.

□

Solución del ejercicio 12. Solo voy a hacer uno de los casos: $r(t) = t^2 - 1$, con $t \in [-2, 2]$. La derivada es $r'(t) = 2t$. Veamos cómo es la función:

```
r = @(t) t.^2 - 1;
rp = @(t) 2*t;
x = linspace(-2,2,1000);
plot(x, r(x));
```

La función tiene *dos* raíces en dicho intervalo y, cuidado, tiene un mínimo local en $x = 0$, por lo que este punto *nunca serviría como semilla*. Vista la gráfica, yo tomaría $x_0 = -2$ ó bien $x_0 = 2$ (aunque prácticamente cualquier número de dicho intervalo en $[-2, -1]$ ó en $[1, 2]$ serviría igual). Una vez grabado el archivo `newtonraphson.m` en `Documentos\MATLAB`, proseguiría así, poniendo $\epsilon = 10^{-5}$ y $N = 10$:

```
x0 = -2;
e = 1e-5;
N = 10;
c = newtonraphson(r, rp, x0, e, N)
```

El valor de `c` debe ser más o menos -1. Comprobemos que es realmente una raíz aproximada: `r(c)` ha de ser menor que 10^{-5} (es mucho menor). ¿Cuántos pasos ha utilizado el algoritmo? Para ello hay que pedir los dos parámetros de salida:

```
[c n] = newtonraphson(r, rp, x0, e, N)
```

que da (en mi caso) solo 5. □

Solución del ejercicio 13. El archivo `newtonraphson_plot.m` podría ser como se indica.

```
% newtonraphson_plot
% igual que newtonraphson pero dibujando cada punto
function [z n] = newtonraphson_plot(f, fp, x0, epsilon, N)
n = 0;
xn = x0;
plot(x0,0,'*');
% initialize z to a NaN (i.e. error by default)
z = NaN;
% Both f and fp are anonymous functions
fn = f(xn);
  while(abs(fn) >= epsilon && n <= N)
    n = n + 1;
    fn = f(xn); % memorize to prevent recomputing
    % next iteration
    xn = xn - fn/fp(xn); % an exception might take place here
    plot(xn, 0, '*');
  end
z = xn;
if(n == N)
  warning('Tolerance not reached.');
end
end
```

LISTADO 2. Contenido del archivo newtonraphson_plot.m

Ahora, para hacer usarlo con $u(t) = t^2 - 3$, $t \in [-2, 2]$, primero dibujamos $u(t)$ y luego tomamos $x_0 = -2$ (pues en la gráfica se ve que sirve), etc:

```
u  = @(t) t.^2 - 3;
up = @(t) 2*t;
v  = [-2:.001:2];
plot(v, u(v));
x0 = -2;
e  = 1e-8;
N  = 10;
hold on % para que dibuje sobre la grafica
[c n] = newtonraphson_plot(u, up, x0, e, N)
```

deberían aparecer los puntos sucesivos en el eje OX, con forma de asterisco. □

Quien lo desee, puede hacer el ejercicio 14, claro.

CURSO 2020/21, EPIG, GIJÓN. UNIVERSIDAD DE OVIEDO
Correo electrónico: fortunypedro@uniovi.es