Práctica IV: Métodos de Newton-Raphson y de la secante, para encontrar las raíces de una función.

Se suele llamar método de Newton-Raphson al método de Newton cuando se utiliza para calcular los ceros de una función real de variable real. Aunque no sea siempre el mejor método para un problema dado, su simplicidad formal y su rapidez de convergencia hacen que, con frecuencia, sea el primer algoritmo que se usa para la resolución de un problema no lineal.

Puede justificarse este algoritmo mediante un enfoque intuitivo basado en el desarrollo de Taylor. Consideremos la ecuación $f(x) = 0$, y supongamos que posee una y sólo una solución $\alpha \in [a, b]$. Partiendo de un punto x_0 suficientemente cercano a dicha raíz, podemos escribir:

$$\alpha = x_0 + h$$

El desarrollo de Taylor de la función $f(x)$ en un entorno de x_0 (con $|x - x_0| < h$) es entonces de la forma:

$$f(x) = f(x_0) + (x - x_0) f'(x_0) + \frac{(x - x_0)^2}{2} f''(x_0 + \theta h)$$

con $0 < \theta < 1$.

Suponiendo que $f'(x)$ no se anula en $[a, b]$, y que la diferencia $x - x_0$ es muy pequeña, el método de Newton-Raphson consiste en despreciar el sumando en $(x - x_0)^2$ del desarrollo anterior, quedándonos con la aproximación:

$$f(x) \cong f(x_0) + (x - x_0) f'(x_0)$$

Por tanto la solución de la ecuación $f(x) = 0$ (que es α) estará próxima a la solución de $f(x_0) + (x - x_0) f'(x_0) = 0$. Si llamamos x_1a dicha solución, entonces x_1 es una mejor aproximación de α que x_0. La solución x_1 cumple que $f(x_0) + (x_1 - x_0) f'(x_0) = 0$ y por tanto:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

De esta manera podemos generar la sucesión $\{x_i\}_{i=1}^{n}$ con la fórmula recurrente:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

La obtención del método de Newton mediante una serie de Taylor resalta la importancia de una buena aproximación inicial. Si x_0 no está suficientemente próximo a α, el método de Newton puede no ser convergente (ya que la solución de $f(x_0) + (x - x_0)\, f'(x_0) = 0$ podría estar alejada de α). Un posible inconveniente de este método es la evaluación de $f'(x_i)$ en cada iteración, pero en caso necesario, esto puede subsanarse con una modificación que veremos más adelante.

Convergencia del método.

Teorema. *Supongamos que* $\alpha \in [a, b]$ *es solución de* $f(x) = 0$ *y que* $f'(\alpha) \neq 0$ *. Entonces, si* $f \in C^2([a, b])$ *y el valor inicial* x_0 *es suficientemente próximo a* α *(* $x_0 \in [\alpha - \varepsilon, \alpha + \varepsilon]$ *para algún* $\varepsilon > 0$*), la sucesión* $x_{n+1} = g(x_n)$ *definida por el método de Newton, converge a* α *.*

Este teorema indica que, bajo suposiciones razonables, el método de Newton converge siempre que la condición inicial esté suficientemente próxima a la raíz. No obstante, es un teorema poco útil en la práctica ya que normalmente no se sabe si el valor inicial está o no próximo a α. La construcción de una gráfica de $f(x)$ puede ser de ayuda para localizar aproximaciones iniciales a la raiz. El siguiente teorema nos da condiciones suficientes para la convergencia del algoritmo:

Teorema. *(Condición suficiente), Supongamos que* $f \in C^2([a, b])$ *tal que:*

(i) $f(a)f(b) < 0$ (la función cambia de signo en los extremos del intervalo $[a, b]$)

(ii) $f'(x) \neq 0 \quad x \in [a, b]$

(iii) $f''(x) \neq 0 \quad x \in [a, b]$

Entonces existe una única raíz α *de* f *en* $[a, b]$*, y la sucesión* $\{x_n\}$ *generada por el método de Newton converge hacia* α *, para cualquier valor inicial* $x_0 \in [a, b]$ *que cumpla que:*

$$f(x_0)f''(x_0) \geq 0.$$

Implementación del método de Newton-Raphson en Matlab-Octave.

La implementación en Matlab/Octave el algoritmo de Newton-Raphson es muy simple. El programa que implemente dicho algoritmo debe recibir como argumentos:

(a) La función $f(x)$ cuya raíz α se va a calcular.
(b) Una aproximación inicial x_0 de dicha raiz.
(c) La derivada $f'(x)$ de la función.
(d) La tolerancia o error máximo ε.

Sugerencia: Para definir la función $f(x)$ y su derivada $f'(x)$ podemos utilizar el comando inline que, como ya hemos visto con el algoritmo de bisección, permite pasar una función como argumento de otra. Por ejemplo, si se desea hallar una raíz de la función $f(x) = x^6 - x - 1$ podemos definir $f(x)$ y su derivada mediante:

```
f=inline('x^6-x-1')
fp=inline('6*x^5-1')
```

Asimismo, recuérdese que para hacer gráficos de una función, se utiliza el comando fplot:

```
fplot(f,[-1,2])
grid
```

El desarrollo del algoritmo es entonces de la forma:

1. *Inicialización:* Leer $f(x)$, $f'(x)$, x_0 y ε. Calcular $f(x_0)$ y $f'(x_0)$. Hacer $k = 0$.
2. Hacer $k = k + 1$. En cada etapa $k \geq 1$ calcular la nueva aproximación $x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}$
3. Si $|x_k - x_{k-1}| < \varepsilon$ terminar: la raíz aproximada es $\alpha = x_k$; en caso contrario volver al paso 2.

Método de Newton-Raphson modificado.

Es análogo al anterior con la diferencia de que el valor de la derivada sólo se calcula en el punto inicial; por tanto la fórmula a utilizar en el paso 2 es $x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_0)}$. Esta modificación converge más lentamente y en general sólo se utiliza cuando hay problemas para calcular la derivada de $f(x)$.

Método de la secante.

Es otra variación del método de Newton-Raphson, en la que el cálculo de la derivada se sustituye por su aproximación numérica:

$$f'(x_k) = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

Por tanto, el paso 2 del algoritmo es:

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{\frac{f(x_{k-1}) - f(x_{k-2})}{x_{k-1} - x_{k-2}}} = x_{k-1} - \frac{(x_{k-1} - x_{k-2})\, f(x_{k-1})}{f(x_{k-1}) - f(x_{k-2})} = \frac{x_{k-2}\, f(x_{k-1}) - x_{k-1}\, f(x_{k-2})}{f(x_{k-1}) - f(x_{k-2})}$$

Además, para aplicar el método de la secante es necesario partir de dos aproximaciones iniciales x_0 y x_1.

Ejercicios.

1. Implementar los tres algoritmos anteriores en MATLAB/OCTAVE en sendas funciones, que deberán llamarse, respectivamente newtonRaphson, newtonRahsonMod y secante. Las dos primeras funciones deben recibir como argumentos la función cuyo cero se quiere calcular, su derivada, la solución inicial y la tolerancia o error de aproximación ε. La tercera función (secante) debe recibir además una segunda aproximación x_1. De esta forma, la función newtonRaphson debe poder invocarse mediante:

   newtonRaphson(f, fp, x0, tol)

   donde las funciones f y fp se definirán previamente mediante comandos inline. Por ejemplo, para hallar el cero de la función $f(x) = 2x^2 - 10x + 3$ partiendo de la aproximación inicial $x_0 = 1$, con error inferior a una milésima, el algoritmo de bisección deberá invocarse mediante:

   f=inline('2*x^2-10*x+3')

   fp=inline('4*x-10')

   newtonRaphson(f, fp, 1, 1e-8)

Solución:

El código para la implementación del algoritmo de Newton-Raphson es: (Pinchar aquí para descargar el archivo .m)

```
function raiz=newtonRaphson(f,fp,x0,tol)
    if (f(x0)==0)
            raiz=x0;
    else
        x1=x0-f(x0)/fp(x0);
        while  ((f(x1)~=0)&&abs(x1-x0)>tol)
            x0=x1;
            x1=x0-f(x0)/fp(x0);
        end %while
```

```
        raiz=x1;
    end %if
    raiz;
end %function
```

Lo aplicamos a la función definida en el ejemplo anterior, especificando previamente format long para que Matlab/Octave nos muestre el resultado con mayor precisión:

```
>>> format long

>>> f=inline('2*x^2-10*x+3')

f =

f(x) = 2*x^2-10*x+3

>>> fp=inline('4*x-10')

fp =

f(x) = 4*x-10

>>> newtonRaphson(f, fp, 1, 1e-8)

ans = 0.320550528229663
```

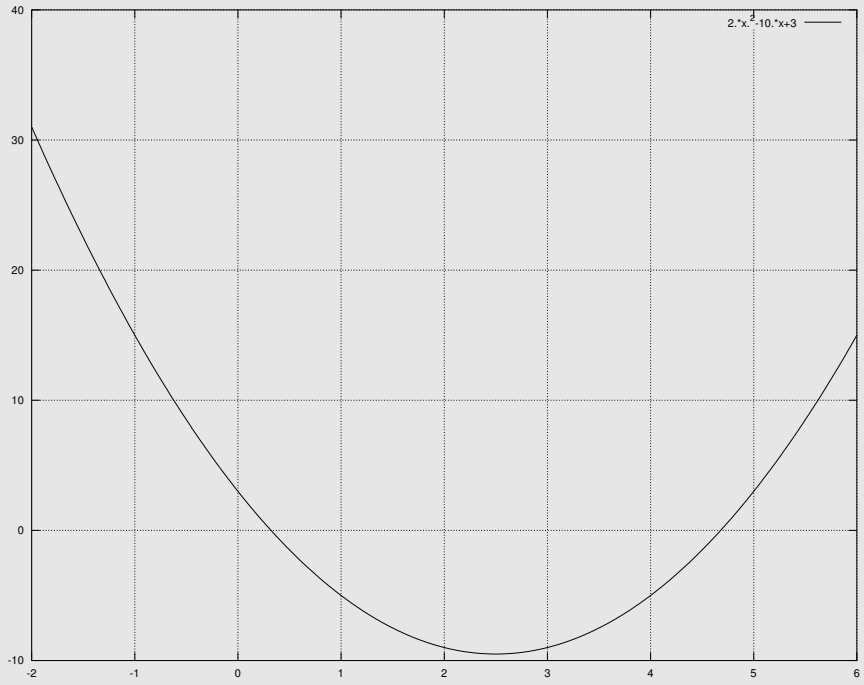
Comprobamos que este valor es efectivamente raíz de $f\left(x\right)=2x^2-10x+3$

```
>>> f( 0.320550528229663)

ans = 1.77635683940025e-15
```

Si representamos esta función entre -2 y 6, vemos que hay otra raíz entre 4 y 5:

```
>>> fplot(f,[-2,6])
```



Para hallar esta raíz por el método de Newton-Raphson simplemente tomamos como valor inicial el $x_0 = 4$:

```
>>> newtonRaphson(f, fp, 4, 1e-8)
ans = 4.67944947177034
```

Comprobamos que efectivamente este valor es solución de $f(x) = 0$:

```
>>> f(4.67944947177034)
ans = 2.84217094304040e-14
```

2. Aplicar los algoritmos anteriores a la búsqueda de soluciones de las siguientes ecuaciones. En todos los casos hacer una gráfica de la función y explorar la convergencia a partir de distintos valores iniciales.

   a) $x^6 - x - 1 = 0$ (*Sugerencia:* hacer una gráfica de la función en el intervalo [-1,2]). Explorar qué ocurre si se toma como valor inicial $x_0 = 0{,}69$ o $x_0 = 0{,}7$.

   b) $x^2 - x - 0{,}5 = 0$.

   c) $sin\,(x+1) - \sqrt{x} = 0$.

   d) $log(x) - cos(x+1) = 3/2$.

e) $x^2 e^{-x/2} = 1$

3. Matlab/Octave disponen de las funciones roots (para hallar las raíces de polinomios) y fzero (para hallar raíces de una función no lineal). Utiliza la ayuda de Matlab/Octave (o los recursos de internet) para investigar cómo operan estas funciones y utilizalas también para obtener las soluciones de las ecuaciones del problema anterior. Compara los resultados.