Facultad de Ciencias – Universidad de Málaga
MÉTODOS NUMÉRICOS I
2º de Grado en Matemáticas – Curso 2022/23

UNIVERSIDAD DE MÁLAGA

– Práctica 2.1 –

Resolución numérica de ecuaciones escalares no lineales I

1. **Método de dicotomía.**
Queremos resolver la ecuación $f(x) = 0$, donde f es una función continua y que tiene un único cero, c, en el intervalo $[a, b]$. Supongamos además que $f(a)f(b) < 0$. El algoritmo del método de dicotomía es el siguiente:

   - Sean $a_0 = a$ y $b_0 = b$.
   - Para $n = 0, 1, \ldots, N$,
     - Calcular $c_n = \dfrac{a_n + b_n}{2}$,
       - Si $f(c_n) = 0$, entonces $c = c_n$, y el proceso termina.
       - Si $f(a_n)f(c_n) < 0$, entonces $a_{n+1} = a_n$ y $b_{n+1} = c_n$.
       - En otro caso, $a_{n+1} = c_n$ y $b_{n+1} = b_n$.
     - Siguiente n.

   El programa Python `bisec.py` (véase fichero en Campus Virtual) ejecuta este procedimiento.

   a) Modificar el programa `bisec.py` para que, en cada iteración, se muestren en pantalla el valor de c_n y de $f(c_n)$.

   b) Aplicar el método de dicotomía a la ecuación
$$x^5 - 5x^3 + 1 = 0$$
para aproximar la solución que se encuentra en el intervalo $[0, 1]$ con 20 iteraciones. Observar cuántas cifras decimales se estabilizan en las aproximaciones sucesivas de la solución y cuánto vale la función en la última aproximación obtenida. Ídem para las soluciones que se encuentran en los intervalos $[-3, -2]$ y $[2, 3]$.

   c) Aplicar el método de dicotomía para aproximar la única solución de la ecuación
$$\cos(x) - x = 0$$
usando 20 iteraciones (a partir de un intervalo adecuado). Observar cuántas cifras decimales se estabilizan en las aproximaciones sucesivas de la solución y cuánto vale la función en la última aproximación obtenida.

Se ha visto en clase que, para este método, se tiene la cota de error:

$$|l - c_n| \leqslant \frac{b-a}{2^{n+1}},$$

siendo $[a, b]$ el intervalo en el que se aplica el método y c_n el punto medio del n-ésimo intervalo que se obtiene, $[a_n, b_n]$. Si se hacen N pasos del método se asegura la cota

$$|l - c_{N-1}| \leqslant \frac{b-a}{2^N}.$$

En consecuencia, si se quiere tener un error menor que ε, es suficiente hacer N pasos del método, siendo N tal que:

$$\frac{b-a}{2^N} \leqslant \varepsilon,$$

lo que se da si N es tal que

$$N \geqslant \frac{\log(b-a) - \log(\varepsilon)}{\log(2)},$$

por lo que basta tomar

$$N = E\left(\frac{\log(b-a) - \log(\varepsilon)}{\log(2)}\right) + 1,$$

siendo $E(\cdot)$ la función parte entera.

(d) Modificar el programa `bisec` de manera que los parámetros de entrada sean:
   - la función f a la que se le busca el cero;
   - los extremos a y b del intervalo que contiene al cero;
   - la precisión ε con la que se desea aproximar el cero.

   El programa debe calcular el número de iteraciones necesarias usando la expresión anterior de N.

(e) Aplicar el programa modificado para aproximar las raíces de las ecuaciones anteriores con un error menor o igual que $\varepsilon = 10^{-7}$.

2. **Método de Regula Falsi.**
La motivación que lleva a este método es la siguiente: si tenemos una función f que toma valores de distinto signo en un intervalo a y b, de la que sabemos que tiene una única raíz en el intervalo y que verifica
$$|f(a)| < |f(b)|,$$
es más probable que tenga una raíz cerca de a, ya que $(a, f(a))$ está más próxima al eje de abscisas que $(b, f(b))$. El algoritmo es el siguiente:

- Sean $a_0 = a$ y $b_0 = b$.
- Para $n = 0, 1, \ldots, \mathrm{N}$,
    - Calcular $c_n = b_n - \dfrac{b_n - a_n}{f(b_n) - f(a_n)} f(b_n)$,
        - Si $f(c_n) = 0$, entonces $c = c_n$, y el proceso termina.
        - Si $f(a_n)f(c_n) < 0$, entonces $a_{n+1} = a_n$ y $b_{n+1} = c_n$.
        - En otro caso, $a_{n+1} = c_n$ y $b_{n+1} = b_n$.
    - Siguiente n.

a) Usar como base el programa `bisec` para definir una función Python `regula_falsi`, siguiendo las instrucciones anteriores y de tal forma que los parámetros de entrada sean:

- la función f a la que se le busca el cero;
- los extremos a y b del intervalo que contiene al cero;
- la precisión ε con la que se desea aproximar el cero;
- el número máximo de iteraciones $nmax$.

El programa deberá detenerse cuando se se obtengan dos aproximaciones sucesivas c_{n-1} y c_n para las que se tenga
$$|c_n - c_{n-1}| \leqslant \varepsilon,$$
o bien cuando se alcance el número máximo de iteraciones. El programa deberá especificar en pantalla si se ha detenido por haberse alcanzado una aproximación satisfactoria o por haberse alcanzado el número máximo de iteraciones sin encontrarla.

b) Aplicar la función `regula_falsi` a las mismas ecuaciones que se consideraron para el método de dicotomía, con cota de error $\varepsilon = 10^{-7}$, y comparar los resultados obtenidos.

c) Modificar el programa de manera que el algoritmo se detenga cuando se llegue a una aproximación c_n para la que se tenga
$$|f(c_n)| \leqslant \varepsilon.$$

d) Aplicar el programa modificado a las ecuaciones anteriormente consideradas.

3. **Método de la secante.**
   El método de la secante tiene mucho en común con el de Regula Falsi, pero a diferencia de éste, no se obtiene en cada iteración un intervalo que contenga a la solución que se busca. El algoritmo es el siguiente:

   - Se eligen dos puntos distintos del intervalo $[a, b]$, sean x_0 y x_1.
   - Para $n = 0, 1, \ldots$, N,
     - Si x_n no pertenece al dominio de la función o si $f(x_n) = f(x_{n-1})$, no se puede continuar. Se detiene el algoritmo.
     - En otro caso se calcula
       $$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n).$$
     - Siguiente n.

   a) Definir la función Python `secante` siguiendo las instrucciones anteriores y de tal forma que los argumentos de entrada sean:
   - la función f a la que se le busca el cero;
   - las aproximaciones iniciales x_0 y x_1;
   - la cota de error ε que se desea garantizar en la aproximación del cero;
   - el número máximo de iteraciones $nmax$.

   El programa deberá detenerse cuando se obtengan dos aproximaciones consecutivas x_{n-1} y x_n para las que se tenga
   $$|x_n - x_{n-1}| \leqslant \varepsilon,$$
   o bien cuando se alcance el número máximo de iteraciones. El programa deberá especificar en pantalla si se ha detenido por haberse alcanzado una aproximación satisfactoria o por haberse alcanzado el número máximo de iteraciones sin encontrarla.

   b) Aplicar la función `secante` a las mismas ecuaciones que se consideraron para los métodos de dicotomía y Regula Falsi, con cota de error $\varepsilon = 10^{-7}$, y comparar los resultados obtenidos.

   c) Modificar el programa de manera que el algoritmo se detenga cuando se llegue a una aproximación x_n para la que se tenga
   $$|f(x_n)| \leqslant \varepsilon.$$

   d) Aplicar el programa modificado a las ecuaciones anteriormente consideradas.