Método de la Regla Falsa

José Ramón Pérez Navarro

September, 2019

0.1. Definición

Este método es una modificación simple del método de bisección, básicamente produce otro método que siempre es convergente. Si se pueden elegir dos aproximaciones iniciales x_0 y $\bar{x}_0$ tales que los dos valores de la función en esos puntos tengan signo opuesto, entonces es posible generar una sucesión de valores que siempre tengan esta propiedad. Para iniciar, se construye la recta que pasa por los puntos $(\mathrm{x}_0, f(x_0)\ y\ (\bar{x}_0, f(\bar{x}_0).\mathrm{m}_1 = m_2$ por lo tanto;

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f(\bar{x}_0) - f(x_0)}{\bar{x}_0 - x_0}$$

El valor del cruce por cero se define cuando se tiene un valor de x_1, dado por la recta definida ecuación anterior donde se cumple que $\mathrm{f}(\mathrm{x}_1) = 0$.Así, la ecuación anterior queda de la siguiente manera:

$$\frac{0 - f(x_0)}{x_1 - x_0} = \frac{f(\bar{x}_0) - f(x_0)}{\bar{x}_0 - x_0}$$

Despejando x_1 se obtiene

$$x_1 = x_0 - \frac{f(x_0)(\bar{x}_0 - f(x_0}{f(\bar{x}_0 - f(x_0)}$$

Y utilizando esta ecuacion, el valor de $\bar{x}_1$ se elige tomando un valor entre x_0 y x_0 de tal forma que el valor de la funcion sea opuesto en signo a $f(x_1)$. Asi, valores de x_1 y $\bar{x}_1$ definen un menor intervalo que contiene el cruce por cero. El proceso continua tomando siempre lados opuestos del cruce por cero. La penalidad que ocasiona esta modificación del método de bisección es el número de operaciones necesarias para calcular los valores de $x_{n}{}_{n=1}^{\infty}$. La longitud de los nuevos intervalos para el método de falsa posición, no decrece en cada nueva iteración como en el método de bisección, es decir, no siempre se garantiza que el nuevo intervalo sea la mitad (o menor) del intervalo anterior. Por esta razón, aunque el método de la falsa posición normalmente tiene una mejor convergencia que el método de bisección, no siempre será el caso.

Programa en Matlab

```
%  Programa principal para determinar los cruces por cero de una función continua
%  Utilizando el método de regla falsa, determinar los cruces por cero de la funci
%  fx = 1 + 2.*x - 3.*x.^2.*exp(-x) + 2.*x.^3.*sin(x).*exp(-x./5) dentro del
%  intervalo [4,20]. Utilizar un error relativo de 1e-6. r=0;
% Inicia contador de cruces por cero.
% Ciclo para calcular todos los cruces por cero dentro del intervalo [4,20]. for k
% Evaluación de la función en el punto k.    gk = 1 + 2.*k - 3.*k.^2.*exp(-k) + 2.
% Evaluación de la función en el punto l=k+1.    gl = 1 + 2.*l - 3.*l.^2.*exp(-l)
% Condicional que marca el subintervalo donde se encuentra un cruce por cero.    i
% Método de regla falsa o falsa posición.        [Cero,Mat] = ReglaFalsa(k,l);
% Contador de cruces por cero.        Cruce(r) = Cero;
% Almacena los cruces por cero.        dr = length(Mat(:,1));
% Número de iteraciones para encontrar el cruce por
% Número de variables por almacenar.
  M(r,1:dr,1:dc) = Mat;
% Matriz que almacena todas las iteraciones de
% todos los cruces por cero.    end end M1(:,:) = M(1,:,:);
% Iteraciones del primer cruce por cero. M2(:,:) = M(2,:,:);
% Iteraciones del segundo cruce por cero. M3(:,:) = M(3,:,:);
% Iteraciones del tercer cruce por cero. M4(:,:) = M(4,:,:);
% Iteraciones del cuarto cruce por cero. M5(:,:) = M(5,:,:);
% Iteraciones del quinto cruce por cero.
```

Función

```
%  Función del método de Regla Falsa o Falsa Posición para cálculo de cruces por c
%  de una función no lineal que se mueve en plano real.
% %  El método calcula un cruce por cero.
% %  La función se llama de la siguiente manera.
% %           [Cero,Mat]=ReglaFalsa(a,b).
% %  Entradas:
%           a  -- Límite inferior del intervalo.
%           b  -- Límite superior del intervalo.
```

```
% %  Salida:
%           Cero -- Valor de la variable para la cual la magnitud de la función es
%           cero o menor a una tolerancia especificada previamente.
%           Mat  -- Vector que contiene todas las iteraciones hasta converger.
% function[Cero,Mat] = ReglaFalsa(a,b); Err = 1;
% Inicializa el error para ingresar al ciclo iterativo. tol = 1e-5;
% Tolerancia especificada para la convergencia. c   = 0;
% Inicializa el contador de iteraciones. while Err > tol & c < 30
% Valor de la función al inicio del intervalo.   fa = 1 + 2.*a - 3.*a.^2.*exp(-a)
% Valor de la función al final del intervalo.   fb = 1 + 2.*b - 3.*b.^2.*exp(-b) +
% Punto de cruce de la recta entre los puntos [a,f(a)] y [b,f(b)].   h  = a - fa*(
% Valor de la función en el punto de cruce.   fh = 1 + 2.*h - 3.*h.^2.*exp(-h) + 2
% Contador de iteraciones para no dejar en un ciclo el programa en caso de alguna
% Matriz que almacena los resultados de cada iteración.   Mat(c,:) = [a fa b fb h
% Discriminante para determinar el nuevo intervalo.   disc = fh*fa;
% El cruce por cero cumple con el criterio de error.   if abs(disc) <= tol       B
% Definición del nuevo intervalo al no cumplir el criterio de error.   elseif disc
% Definición del nuevo intervalo al no cumplir el criterio de error.   elseif disc
% Criterio de error. end
```