Solución de Ecuaciones no lineales

Encontrar raíces o ceros de f(x) en un intervalo dado, es decir encontrar x / f(x) = 0 con $x \in [a, b]$.

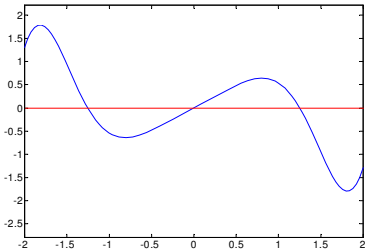
- Es posible siempre despejar x?
- Se puede implementar cualquier despeje en un programa de computación?

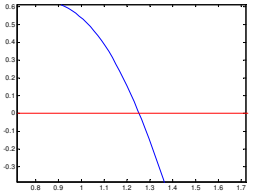
Graficación

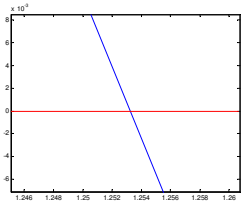
Ej.: x*cos(x^2) = 0

fplot('(x*cos(x^2))',[-2,2])

en [-0.5, 0.5], x_1=0
en [-2, -0.5], x_2=-1.2533
en [0.5, 2], x_3=1.2533







ezplot ('x*cos(x^2) ',[-2,2])

fplot(@fun,[-2,2])

function y=fun(x);
y=x*cos(x^2);

Métodos Numéricos para la Solución de Ecuaciones no lineales

Un número α se dice **raíz** o **cero** de la ecuación *f(x)* si $f(\alpha) = 0$.

Los métodos numéricos para encontrar una raíz de una ecuación *f(x)*, son **métodos iterativos** que generarán una sucesión $\{ x_n \}$, $n=1,2,3,...$ tal que: $\lim_{n\to\infty} x_n = \alpha$.

x_0= *valor inicial*

$x_1 = m(x_0)$, siendo *m()* una función correspondiente a un

... método numérico

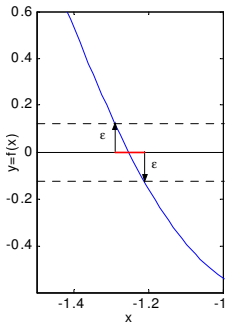
$x_n = m(x_{n-1})$, $|x_n - \alpha| < |x_{n-1} - \alpha| < |x_{n-2} - \alpha| < ... < |x_0 - \alpha|$

1er. Criterio de aproximación

Si la función *f* es continua alrededor de α, en un intervalo que contiene a la sucesión x_n, $n=1,2..$, entonces, para algún número positivo ε_1 existe *n* tal que $|f(x_n)| < \varepsilon_1$.

Dado un número $\varepsilon_1 > 0$ y adecuadamente pequeño, que llamaremos **tolerancia,** podemos escoger como **aproximación a la raíz** α a un término x_n de la sucesión mencionada, donde *n* es el menor entero positivo que satisface:

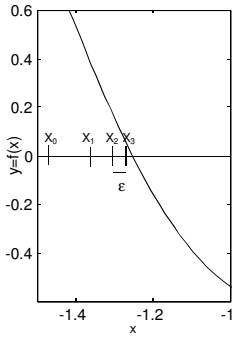
$$|f(x_n)| < \varepsilon_1.$$



2do. Criterio de aproximación

$lim_{n\to\infty} x_n = \alpha \Rightarrow$ dado $\varepsilon_2 > 0$, adecuadamente pequeño, $\exists\ n$ tal que $| x_n - \alpha | < \varepsilon_2$.

El término x_n de la sucesión mencionada puede ser considerado una **aproximación a la raíz**, donde n es el menor entero positivo que cumple la condición.



Como α es desconocida, podemos asumir que:

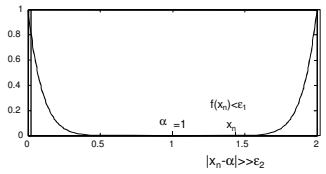
$$| x_{n+1} - x_n | = |x_{n+1} - \alpha + \alpha - x_n |$$
$$\leq |x_{n+1} - \alpha| + | x_n - \alpha | < \varepsilon_2 + \varepsilon_2 \leq 2\ \varepsilon_2 < \varepsilon_2$$

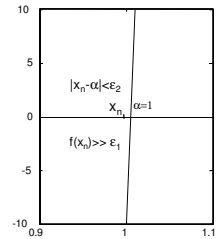
$\Rightarrow$ **$|x_n - x_{n-1}| < \varepsilon_2$** **ó** **$|x_n - x_{n-1}| / |x_n| < \varepsilon_2$**

error absoluto — error relativo

$|x_n - x_{n-1}| \to 0$ y n indica el número de iteración.

Convergencia





El hecho de que

$|f(x_n)| < \varepsilon_1$ ó $|x_n - x_{n-1}| < \varepsilon_2$

no necesariamente indica que x_n esté muy cerca de α.

Además, para garantizar que una sucesión converge a la raíz buscada, la función f deberá satisfacer ciertas condiciones que evitaran una sucesión divergente.

La convergencia puede ser "muy lenta", por lo cual se hace necesario establecer una cota para n, es decir, imponer un **máximo número de iteraciones**.

Algoritmos iterativos

```
function xn = metodo (x0, @fun, max_iter, tolf, tolx)
xn(1) = x0;
for i =1:max_iter
      xn(i+1) = m (xn(i));
      if abs (feval (@fun, xn(i+1) ) ≤ tolf & abs ( xn(i+1) - xn(i) ) ≤ tolx
      return , end
end
```

% donde metodo es el nombre de algún método y m es la función de iteración corresponiente

1er. Criterio **2do. Criterio**

%Puede considerarse una sola condición

- *feval (f, a_1,a_2,....,a_n)* : Evalúa la función *f* que utiliza los argumentos a_1,a_2,....,a_n., *f* debe ser definida como function. Equivalente a evaluar *f(a1,a2,....,an).*
  Ej.: *feval*('sqrt',5) ó *feval*(@sqrt,5), evalúan sqrt(5).
- *eval (f)* : Evalúa la función *f,* la cual es un comando en forma de'*string*'.
  Ej: eval ('x+6*x^2') si x está definida.

Métodos cerrados

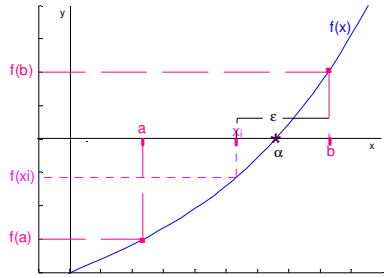
Los métodos numéricos para encontrar la raíz de una ecuación se denominan *cerrados* cuando necesitan conocer un intervalo que encierre a la raíz.

Si *f* es una función continua en un intervalo [a,b], y $f(a) * f(b) < 0$. Entonces, por el teorema del valor medio para funciones continuas (teorema de Bolzano), existe al menos un $\alpha \in [a,b] / f(\alpha) = 0$.

A los fines de la aplicación de los métodos numéricos, deberíamos considerar que la raíz en el intervalo es única, ya que encuentran una única raíz.

Método de Bisección

Consiste en dividir sucesivamente el intervalo [*a,b*] por la mitad, hasta que la longitud del sub-intervalo que contiene a la raíz α sea menor que alguna tolerancia especificada ε .



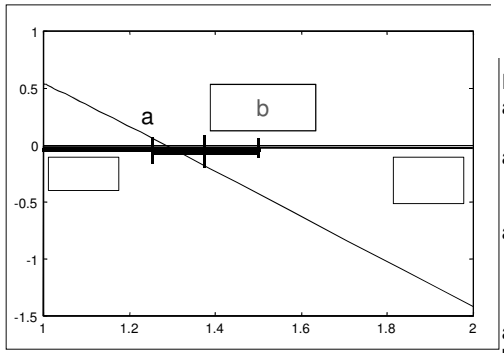
X_0=[a,b]

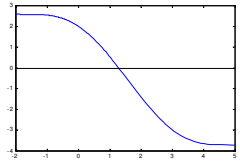
f(a) < 0
f(b) > 0
Si b-a > ε
  x_i = (a+b)/2
  Si f(x_i) > 0 b = x_i
    Siguiente iteración
  sino a = x_i
    Siguiente iteración
sino fin

Proceso método bisección

f(x) = cos(x)+1-x

fplot('cos(x)+1-x',[1 2])





Resultados		b-a
a =1	b = 2	1
	b =1.5000	0.5
a = 1.2500		0.25
	b = 1.3750	0.125
	b = 1.3125	0.0625
a = 1.2813		0.0312
	b = 1.2969	0.0156
	b = 1.2891	0.0078
	b = 1.2852	0.0039
a = 1.2832		0.0019
b = 1.2842		0.0010

```
function xn = biseccion (x0, f, max_iter, tolx, tolf)
%-----------------BISECCION----------------------
% x0= [a,b]  valor inicial, f: función a resolver
% max_iter: número máximo de iteraciones
% tolx  y toly: diferencia máxima aceptable entre dos iteraciones
%    ej:    biseccion ([1 2], @fun,10, 0.001,0.001);

a=x0(1); b=x0(2);
if (f(a) * f(b) > 0)  % si es = 0 alguno de los extremos es raíz
    disp('La función en los extremos del intervalo tiene el mismo signo')
else
     for i =1:max_iter
          if (abs(b-a) > tolx)
               xn = (a+b)/2;     % se podría evaluar f(xn) y verificar tolf
                    if  (sign(f(xn))== sign(f(a)) )
                         a= xn
                    else
                         b = xn
                    end
           else
          disp('llegó a la tolerancia esperada'),return
      end
end
disp('Se empleó maximo número iteraciones')
end
```

```
function y = fun(x)
y = cos(x)+1-x;
```

Ventajas y desventajas del método de bisección

Ventajas:

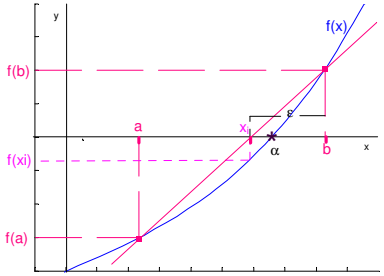
- El error $\varepsilon = |\alpha - x_n|$, se acota fácilmente ya que $|\alpha - x_n| \leq (b-a) / 2^n$, previendo la cantidad de iteraciones necesarias, $n \geq (\log (b-a) - \log (\varepsilon)) / \log (2)$
- Siempre converge.
- Es útil para acotar el intervalo en que se encuentra la raíz.

Desventajas con respecto a otros métodos:

- No tiene en cuenta la magnitud de los valores de la función en las aproximaciones calculadas, sólo tiene en cuenta el signo de $f(x_n)$, lo que hace que buenas aproximaciones intermedias pasen desapercibidas.
- Su convergencia es muy lenta.

Método de Posición Falsa (o Regula Falsi)

La aproximación x_n a la raíz α es el punto de intersección de la recta que pasa por los puntos (a,f(a)) y (b,f(b)) con el eje x

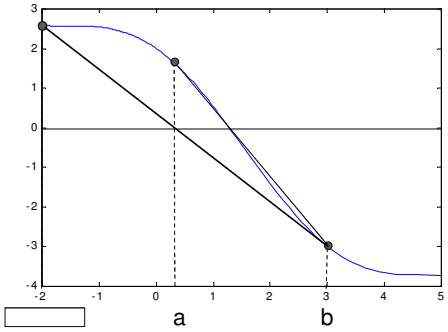


Al reemplazar la curva por una recta se obtiene una "posición falsa" de la raíz. También se le conoce como **método de Interpolación Lineal Inversa**.

$$f(x_i) = f(a) + \underbrace{(f(b)-f(a))\frac{(x_i-a)}{(b-a)}}_{\text{Ec. de la recta que pasa por (a,f(a)) y (b,f(b))}}, \text{ Si } f(x_i)=0 \Rightarrow x_i = a - \frac{(b-a)*f(a)}{f(b)-f(a)} = \frac{a*f(b)-b*f(a)}{f(b)-f(a)}$$

Proceso método regula falsi o Interpolación Lineal Inversa

f(x) = cos(x)+1-x



Resultados	b-a
a = -2 , b = 3	5
a = 0.3178	2.6822
a = 1.2649	1.7351
b = 1.2857	0.0208
a = 1.2834	0.0023
b = 1.2834	0.0000

```
function xn = regulaFalsi (x0, f, max_iter, tolx, tolf)
%----------------- REGULA FALSI ---------------------
% x0= [a,b]  valor inicial, f: función a resolver
% max_iter: número máximo de iteraciones
% tol: diferencia máxima aceptable entre dos iteraciones
%       ej: regulaFalsi ([1 2], @fun,10, 0.001,0.001);

a=x0(1); b=x0(2);
if (f(a) * f(b) > 0) % si es = 0 alguno de los extremos es raíz
    disp('La función en los extremos del intervalo tiene el mismo signo')
else
      for i =1:max_iter
            if (abs(b-a) > tolx)
                x = (a*f(b)-b*f(a))/(f(b)-f(a)); % se podría evaluar f(x) y verificar tolf
                      if  (sign(f(x))== sign(f(a)) )
                          a= x
                      else
                          b = x
                      end
             else
             disp ('llegó a la tolerancia esperada'),  return
       end
end
disp('Se empleó maximo número iteraciones')
end
```

Ventajas y desventajas del método de regula falsi

Ventaja

- Converge más rápido que bisección

Desventaja, con respecto al método de Bisección

- La longitud del sub-intervalo que contiene a la raíz en general no tiende a cero, porque la mayoría de las gráficas de las funciones son cóncavas en la vecindad de la raíz, lo que hace que uno de los extremos de los sub-intervalos se aproxime a la raíz, mientras el otro permanece fijo.
- No se puede prever el número de iteraciones necesarias.

Comando *fzero* de Matlab

Combina la confiabilidad de bisección con la velocidad de convergencia de la interpolación linear inversa y la interpolación cuadrática inversa.

[X , FVAL , EXITFLAG] = FZERO (FUN, X0 , OPTIONS , P1 , P2 ,...)

X	Ultima aproximación a la raíz
FVAL	Función evaluada en X
EXITFLAG	> 0. Indica que encontró la raíz, < 0 no
FUN	Función 'en línea' o llamada a una función (@)
X0	- Si es un escalar, busca un intervalo válido alrededor de X0. - Si es un vector de 2 valores, los considera como extremos del intervalo que contiene la raíz.
OPTIONS	Opciones . Optimset ('disp','iter') muestra resultados y detalle de las iteraciones

Problemas de los métodos cerrados

- Necesidad de conocer dos valores iniciales que encierren a la raíz.
- En los extremos del intervalo la función debe tener signos opuestos. No permite encontrar raíces que corresponden al máximo o mínimo de una función.
- En el intervalo puede existir más de una raíz (nro. impar).
- Difícil de diferenciar entre dos raíces muy cercanas.
- Debido a errores de redondeo, particularmente cuando se está cerca de la raíz, puede cambiar el signo de f(x).
- Necesitan de muchas iteraciones hasta llegar a la precisión deseada.

Métodos abiertos

- Los métodos abiertos requieren de un solo valor inicial (no de un intervalo).
- Como no hay un intervalo que encierre a la raíz; algunas veces las sucesiones generadas por estos métodos son divergentes (condiciones de convergencia).
- Se pueden alejar de la raíz de interés (van probablemente a otra raíz).
- Tienen la ventaja que cuando convergen lo hacen "más rápidamente" que las sucesiones generadas por los métodos cerrados.
- Necesidad de intervención del usuario (o de procesamiento simbólico) para despejes y cálculo de derivadas.

Método del punto fijo o de las aproximaciones sucesivas

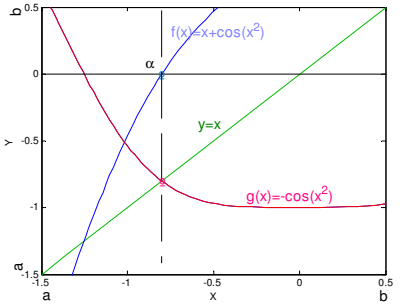
Dada una ecuación $f(x) = 0$, si se puede transformar en otra equivalente (al menos localmente) del tipo $x = g(x)$ para alguna función $g(x)$

$\Rightarrow$ Si α es raíz de $f(x)$

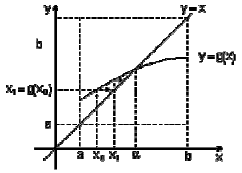
$\Rightarrow$ $f(\alpha) = 0$

$\Rightarrow$ $\alpha = g(\alpha)$

Un número α tal que $\alpha = g(\alpha)$ se dice un **punto fijo** de la función $g(x)$.



Teorema del punto fijo



Si $g(x)$ es una función continua en $[a, b]$ y $g(x) \in [a, b]$ para todo $x \in [a, b]$, entonces $g(x)$ tiene por lo menos un punto fijo en [a, b] .

Si además, $g'(x)$ existe para todo $x \in [a, b]$ y $|g'(x)| \le K < 1$ para todo $x \in [a, b]$, K constante, entonces $g(x)$ tiene un único punto fijo α en $[a, b]$.

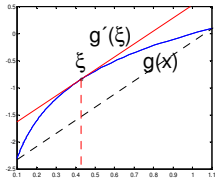
El **método de punto fijo** para encontrar una raíz α de la ecuación $x = g(x)$, consiste en generar una sucesión $\{x_n\}_n$ que se define mediante la **fórmula de iteración:** $\boldsymbol{x_n = g(x_{n-1})}$, que converge a α cualquiera sea $x_0 \in [a, b]$.

La función $g(x)$ se denomina **función de iteración de punto fijo**.

Criterio de Convergencia del Método del punto fijo

- Por teorema del valor medio (Lagrange) Dado un intervalo $[x_i, x_{i-1}]$, existe al menos un punto en el que la tangente a la función es paralela a la secante que une los puntos $(x_i, g(x_i))$ $(x_{i-1}, g(x_{i-1}))$

  $g(x_i) - g(x_{i-1}) = g'(\xi)(x_i - x_{i-1})$ $\quad \xi \in [x_i, x_{i-1}]$
- Por teorema del punto fijo $\quad x_n = g(x_{n-1})$



$\Rightarrow x_{i+1} - x_i = g'(\xi)(x_i - x_{i-1}) \Rightarrow |x_{i+1} - x_i| = |g'(\xi)|\,|x_i - x_{i-1}|$

suponiendo $g'(\xi)$ acotada, $|g'(\xi)| \le K$

$\Rightarrow K<1 \therefore \boldsymbol{|g'(\xi)| < 1}$

$\Rightarrow |x_2 - x_1| \le K\,|x_1 - x_0|$

$\Rightarrow |x_3 - x_2| \le K\,|x_2 - x_1| \le K\,(K\,|x_1 - x_0|) \le K^2\,|x_1 - x_0|$

$\Rightarrow |x_4 - x_3| \le K\,|x_3 - x_2| \le K\,(K^2\,|x_1 - x_0|) \le K^3\,|x_1 - x_0|$

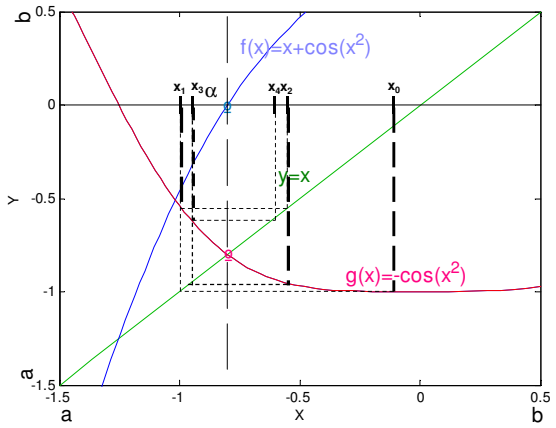
$\Rightarrow \ldots |x_{i+1} - x_i| \le K^i\,|x_1 - x_0|$ y para que $|x_{i+1} - x_i| \to 0$ debe $K^i \to 0$

i) $\varepsilon = |\alpha - x_n| \le K^n\,|x_1 - x_0|$, para cada $n \ge 0 \Rightarrow n \ge \dfrac{\ln(\varepsilon) - \ln(|x_1 - x_0|)}{\ln(K)}$

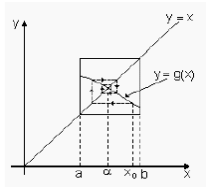
ii) $\varepsilon = |\alpha - x_n| \le K^n \underbrace{\max\{x_0 - a, b - x_0\}}_{m}$, para cada $n \ge 0 \Rightarrow n \ge \dfrac{\ln(\varepsilon) - \ln(m)}{\ln(K)}$

iii) $\varepsilon = |\alpha - x_n| \le \dfrac{K}{1-K}|x_n - x_{n-1}|$, para cada $n \ge 1$ con $K = \max(g'(x))$ en [a,b]

Convergencia del Método del punto fijo

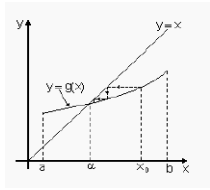


Ejemplos Convergencia del Método del punto fijo

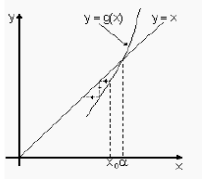


Converge g'(x) > -1

$|g'(x)| < 1$

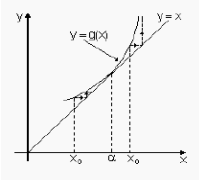


Converge g'(x) < 1



Diverge g'(x) > 1

$|g'(x)| < 1$



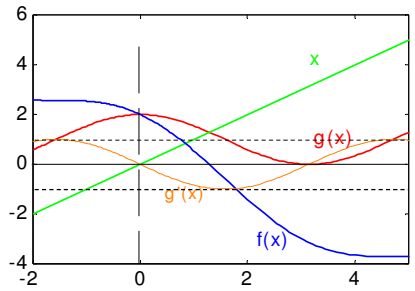
Depende del valor inicial

Ejemplo

f(x) = cos(x)+1-x $\Rightarrow$ ***x = cos(x)+1***

Resultados

x_0=1.5, tol_x=0.0001



1.5
1.0707
1.4795
1.0912
1.4614
1.1092
1.4454
1.1250
1.4311
1.1392
1.1392

f (1.1392) = 0.2791 ? g'(1.57)= -1 α =1.2834

Algoritmos iterativos Punto Fijo

```
function xn = puntoFijo (x0, f,g, max_iter, tolf, tolx)
% x0: valor inicial, f: función a resolver
% max_iter: número máximo de iteraciones
% tol: diferencia máxima aceptable entre dos iteraciones
%      ej:  puntoFijo (1.5, @fun,'cos(x)+1',10, 0.001,0.001);
xn(1) = x0;
for i =1:max_iter
        x=xn(i), xn(i+1) = eval (g,x);
       if abs (feval (f, xn(i+1) ) ≤ tolf & abs ( xn(i+1) - xn(i) ) ≤ tolx
       end
end
```

Orden de Convergencia de punto fijo

La magnitud de g´(x) no sólo indica si el método converge, sino también la rapidez con que lo hace, lo cual se conoce como orden de convergencia.

$$g(x_i) = g(x_{i-1}) + g'(x_{i-1})(x_i - x_{i-1}) + g''(x_{i-1})\frac{(x_i - x_{i-1})^2}{2!} + g'''(x_{i-1})\frac{(x_i - x_{i-1})^3}{3!} + \cdots$$

es decir

$$g(x_i) - g(x_{i-1}) = g'(x_{i-1})(x_i - x_{i-1}) + g''(x_{i-1})\frac{(x_i - x_{i-1})^2}{2!} + g'''(x_{i-1})\frac{(x_i - x_{i-1})^3}{3!} + \cdots$$

como $x_{i+1} = g(x_i)$, $x_i = g(x_{i-1})$ y $E_{i+1} = x_{i+1} - x_i$

$$x_{i+1} - x_i = g'(x_{i-1})(x_i - x_{i-1}) + g''(x_{i-1})\frac{(x_i - x_{i-1})^2}{2!} + g'''(x_{i-1})\frac{(x_i - x_{i-1})^3}{3!} + \cdots$$

$$E_{i+1} = g'(x_{i-1})E_i + g''(x_{i-1})\frac{E_i^2}{2!} + g'''(x_{i-1})\frac{E_i^3}{3!} + \cdots$$

Si g'(x) ≠0 ⇒convergencia lineal. O(h)
Si g'(x) →0, g''(x) ≠0 ⇒convergencia cuadrática. O(h^2)

Ventajas y desventajas del método de punto fijo

Ventajas:

- Simple
- Posee condiciones para asegurar la convergencia.
  Es condición **necesaria** que $|g'(x)| < 1$ en cercanías de la raíz (se NECESITA que se cumpla la condición para converger)

Desventajas:

- La convergencia depende de la magnitud de *g'(x).*
- Necesidad de construir funciones *g(x)* para iterar. Pueden existir diversas *g(x),* necesidad de encontrar la adecuada. Se puede emplear un método sistemático para construir las funciones.
  $g(x) = x - \lambda f(x)$ con $\lambda = 1/ max\ (|f'(x)|)$ *en [a,b].*
  $g(x) = x - f(x) / max\ (|f'(x)|)$

Método de Newton-Raphson

Dada una función *f* que satisface la hipótesis general en un intervalo [*a, b*] y un valor inicial $x_0 \in [a, b]$ "cercano" a la raíz.

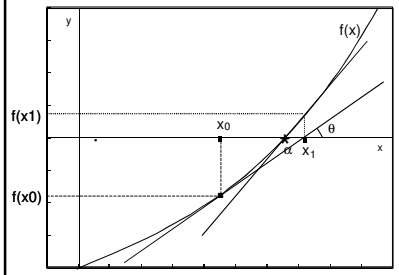
El método de Newton-Raphson consiste es trazar una recta tangente a *f* que pase por el punto $(x_0, f(x_0))$, considerando una aproximación x_1 a la raíz al punto en el cual dicha recta tangente corta al eje *x*, es decir el punto $(x_1, 0)$

$\Rightarrow$ se tiene que $f'(x_o) = \dfrac{f(x_0) - 0}{x_0 - x_1} \quad \therefore \quad x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$

y en general, para cada $n \geq 1$

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$$

Método de Newton



Usando polinomios de Taylor:

Sea $x \in [a, b]$ con $|x-\alpha|$ "pequeño".

$f(\alpha)=f(x)+f'(x)(x-\alpha)+f''(\xi)(x-\alpha)^2/2!$

Si $f(\alpha) = 0$

$x = \alpha - f(x) / f'(x)$

error = $f''(\xi)(x-\alpha)^2 / 2!$

(convergencia cuadrática)

$O(h^2)$

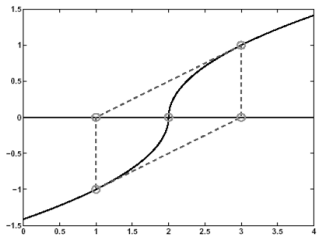
$\tan(\theta) = f'(x_0) = \dfrac{f(x_0)}{x_0 - x_1} \therefore x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$

El **método de Newton-Raphson** es un **caso especial del método de iteración de Punto Fijo**, con $g(x) = x - f(x) / f'(x)$

Convergencia del Método Newton

El método de Newton-Raphson converge como:

$|x_n - x_{n-1}| = \left|\dfrac{f(x)}{f'(x)}\right|$ $\Rightarrow$ cuando más grande sea $f'(x)$ en la vecindad de la raíz α, "más rápida" será la convergencia.



Condiciones **suficientes** para asegurar convergencia, (no son **necesarias**, si no se cumple alguna o todas el método igualmente podría converger)

- $f'(x) \neq 0$ en todo [a,b]
- $f''(x) > 0$ ó < 0 en todo [a,b]
- $|f(h)| / |f'(h)| \leq b-a$,
  $h = x / f'(x)$
  $f'(x) = min\ \{f'(a), f'(b)\}$

Velocidad de convergencia en raíces múltiples

Orden de una raíz:

Si f(x) y sus derivadas f'(x)….$f^{(M)}(x)$ están definidas y son continuas en un intervalo [a,b] que contiene a la raíz α

$\Rightarrow$ f(x) tiene una raíz de orden M en α si

$f(\alpha) = 0$, $f'(\alpha) = 0$,…, $f^{(M-1)}(\alpha) = 0$ y $f^{(M)}(\alpha) \neq 0$.

$\Rightarrow f(x) = (x - \alpha)^M h(x)$ con $h(x) \neq 0$.

Fórmula de iteración de Newton acelerada para raíces múltiples:

$$x_n = x_{n-1} - \frac{M * f(x_{n-1})}{f'(x_{n-1})}$$

Ventajas y desventajas del método de Newton

Ventajas:

- Convergencia rápida. El método de Newton converge cuadraticamente para raíces simples y linealmente para raíces múltiples.
- **Encuentra raíces complejas** (el valor inicial debe ser complejo)

Desventajas:

- Necesita calcular la derivada. (Método de la secante).
- No se pueden prever la cantidad de iteraciones a partir de una cota de error.
- No siempre converge. (No se puede asegurar la convergencia si en [*a,b*], *f'(x) = 0, f"(x)* cambia de signo, la tangente cae fuera del intervalo).

Método de la secante

El método de Newton-Raphson para aproximar una raíz simple α de una ecuación $f(x) = 0$, consiste en generar la sucesión $\{x_n\}_n$ a partir de la fórmula de iteración

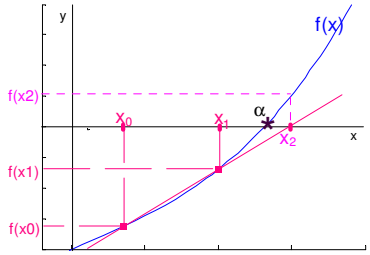
$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$$

Reemplazando $f'(x)$ por su aprox.

$$f'(x_{n-1}) = \frac{f(x_{n-1}) - f(x_{n-2})}{x_{n-1} - x_{n-2}}$$

Se obtiene

$$x_n = x_{n-1} - \frac{f(x_{n-1})(x_{n-1} - x_{n-2})}{f(x_{n-1}) - f(x_{n-2})}$$



que es la fórmula de **iteración del método de la secante.**

En este método se necesitan dos valores iniciales, que no necesariamente encierran a la raíz.

Convergencia del Método de la secante

La fórmula de iteración coincide con el método de la Regla Falsa.

En el método de la Regla Falsa los dos puntos deben encerrar a la raíz buscada y el método siempre converge.

En el método de la Secante los dos puntos iniciales no necesariamente encierran a la raíz buscada lo que puede provocar divergencia del método.

Raíces de Polinomios

- Todo polinomio de grado n , tiene n raíces, las cuales pueden ser reales o complejas, las raíces complejas siempre se presentan como un par de raíces conjugadas. Las raíces reales pueden ser múltiples.
- Método más adecuado: Newton, la derivada de un polinomio es fácil de obtener.
- Todos los métodos vistos permiten calcular una sola raíz por vez.
- Matlab calcula simultáneamente todas las raíces de un polinomio con la función *roots*([a_n, a_{n-1},a_{n-2},…a_0]) la cual calcula los autovalores de la *matriz compañera*.

a = diag (ones(1,n-2),-1)
a(1,:) = -c(2:n). / c(1)
eig (a)

$$\begin{bmatrix} -a_2/a_1 & \cdots & -a_n/a_1 & -a_0/a_1 \\ 1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Eficiencia de los métodos

Los métodos numéricos para hallar una raíz α de una ecuación $f(x)= 0$ consisten en generar una sucesión

$\{x_n\}$ tal que $\lim_{n\to\infty} x_n = \alpha$

La eficiencia cada método numérico depende de:

• **Condiciones de convergencia:** La característica de la ecuación en cercanías a la raíz condiciona la eficacia del método.

• **Orden de convergencia:** Velocidad de convergencia con la cual la sucesión $\{x_n\}_n$ converge a α, $O(h^m)$.

• **Número *n* mínimo de iteraciones necesarias** para que $|x_n-\alpha|<\varepsilon$ para algún $\varepsilon >0$ dado.

• **Cantidad de operaciones** realizadas en cada iteración.