

Raíces de ecuaciones no lineales 2024

Prof.: Dr. Juan Ignacio Manassaldi

J.T.P.: Ing. Amalia Rueda

- Dado el caso de funciones racionales enteras (polinomios) igualadas a cero de la forma:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_1 x + a_0 = 0$$

con $a_n \neq 0$ y n entero > 0 y $a_i \in \mathbb{R}$ con $i = 0, 1, 2, 3, \dots, n$ constantes, existen métodos para resolverlas directamente en términos de sus coeficientes, siempre que el grado n de la ecuación sea a lo sumo 4.

- Las llamadas “resolventes” solo existen hasta polinomios de grado 4, siendo las de grado 3 y 4 de elevada complejidad computacional.

- Para las ecuaciones **algebraicas no lineales** (en particular para los polinomios de grado $n > 4$), para las **ecuaciones trascendentes** y aún para las ecuaciones **polinómicas de tercer y cuarto grado** (por lo complicado de las resolventes) conviene establecer métodos generales que nos permitan resolverlas, aunque sea de una forma aproximada.
- Paradójicamente, la solución aproximada en la mayoría de los casos es mejor que la que suministran los métodos directos (sólo en el caso de los polinomios).
- Los métodos aproximados permiten mejorar la solución por simple repetición del mismo método hasta lograr el grado de aproximación requerido.

Solución numérica de ecuaciones algebraicas y trascendentes de una variable

- Sea una función de una variable $f(x)$.
- Se busca encontrar un valor x^* para el que se cumpla $f(x^*) = 0$.
- Si existe este valor se lo denomina raíz de la ecuación.
- Pasos básicos para encontrarlo:
 - Determinación de un valor aproximado de la raíz (valor de arranque de método).
 - Mejoramiento de la solución hasta un grado de precisión
tolerancia

Interpretación del problema

Método iterativo de calculo



- **Solución Iterativa:** Significa comenzar con una solución inicial (aproximada) y generar una secuencia de números (sucesión):

$$\{x_n\} = x_1, x_2, \dots, x_n$$

tal que si existe

$$\lim_{n \rightarrow \infty} \{x_n\} = x^* \quad y \quad f(x^*) = 0$$

entonces x^* es una raíz de la ecuación $f(x^*) = 0$.

- Error Absoluto Exacto en la Iteración n :

$$E_{\text{exacto}}^{(n)} = |x_n - x^*|$$

- El error exacto en la iteración n no se conoce:

$$E_{\text{exacto}}^{(n)} = |x_n - x^*|$$

por lo tanto no puede utilizarse como criterio de terminación de método.

- Tolerancia del Error:

$$|f(x_n)| < \varepsilon$$

$$E_{\text{aprox.}}^{(n)} = |x_n - x_{n-1}| < \varepsilon$$

Error absoluto

$$e_{\text{rel.}}^{(n)} = \left| \frac{x_n - x_{n-1}}{x_n} \right| < \varepsilon$$

Error relativo

$$\text{si : } x_n \rightarrow x^* \Rightarrow f(x_n) \rightarrow 0$$

Orden de convergencia

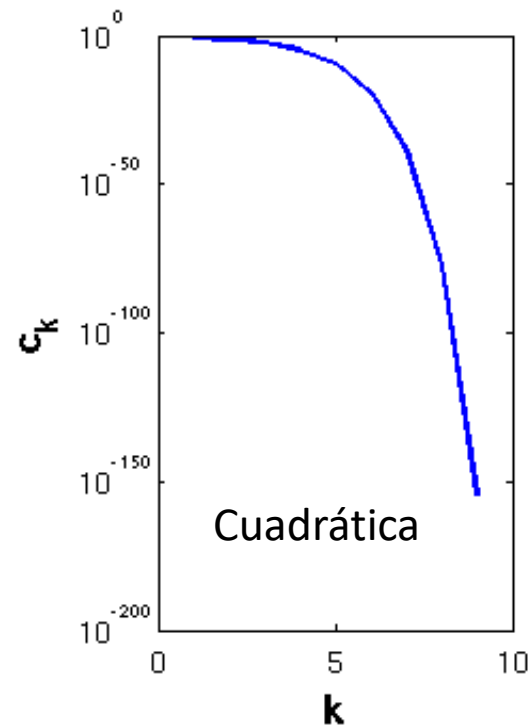
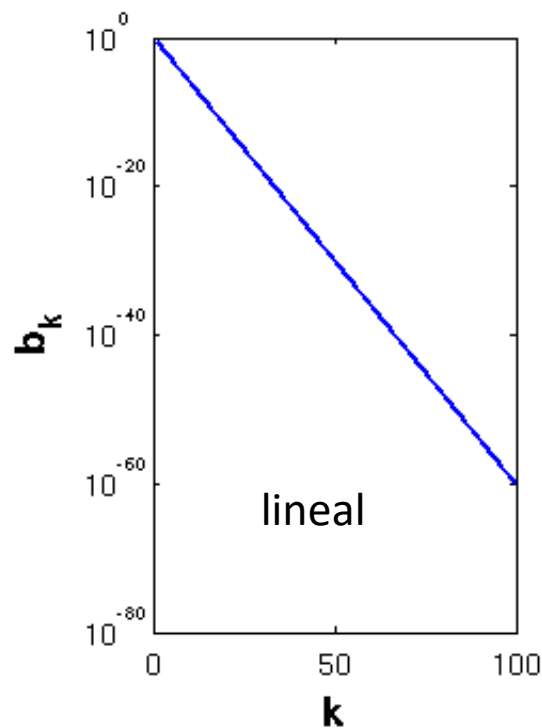
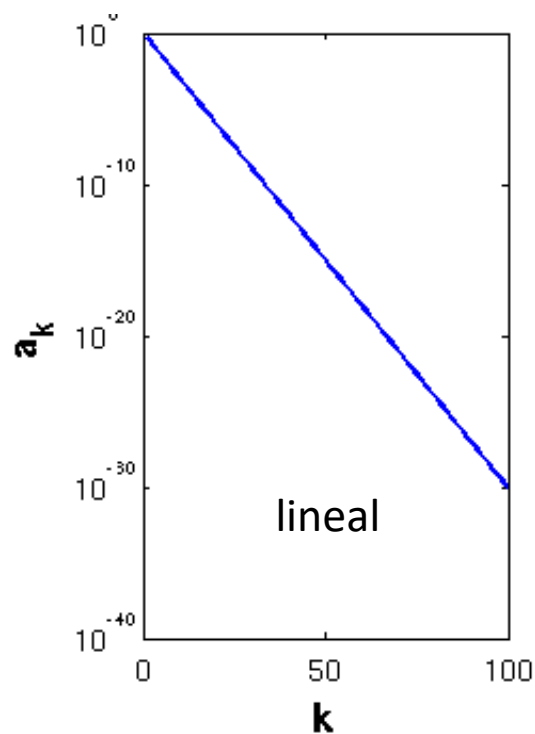
- Sea $\{x_n\}$ una secuencia que converge a x^* .
- Si existen las constantes positivas p y K tales que:

$$\lim_{n \rightarrow \infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^p} = \lim_{n \rightarrow \infty} \frac{E_{\text{exacto}}^{(n+1)}}{\left(E_{\text{exacto}}^{(n)}\right)^p} = K$$

- Entonces, se dice que la secuencia $\{x_n\}$ converge a x^* con orden p y un error asintótico representado por la constante K .
- Una secuencia con mayor orden de convergencia convergerá más rápido que una con bajo orden.
- Por otro lado, además del orden de convergencia, en el proceso de cálculo interviene el costo computacional por iteración que es importante para definir la eficiencia del método.

$$(a_k) = \left\{ 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \dots, \frac{1}{2^k}, \dots \right\} \quad (c_k) = \left\{ \frac{1}{2}, \frac{1}{4}, \frac{1}{16}, \frac{1}{256}, \frac{1}{65,536}, \dots, \frac{1}{2^{2^k}}, \dots \right\}$$

$$(b_k) = \left\{ 1, 1, \frac{1}{4}, \frac{1}{4}, \frac{1}{16}, \frac{1}{16}, \dots, \frac{1}{4^{\lfloor \frac{k}{2} \rfloor}}, \dots \right\}$$



Método	Orden de Convergencia
Aproximaciones sucesivas	Lineal ($p=1$)
Newton - Raphson	Cuadrático ($p= 2$)
Secante	Superlineal ($p = 1.618$)
Bisección	Lineal ($p = 1$)

Teóricos/Prácticos

- Entender y aplicar los algoritmos de manera manual.
- Interpretar problemas de ingeniería y proponer una aproximación de la solución.
- Programar una “function” de scilab para cada método de manera de poder aplicarlas en los diferentes problemas de la guía.
- Plantear y resolver problemas de ingeniería.

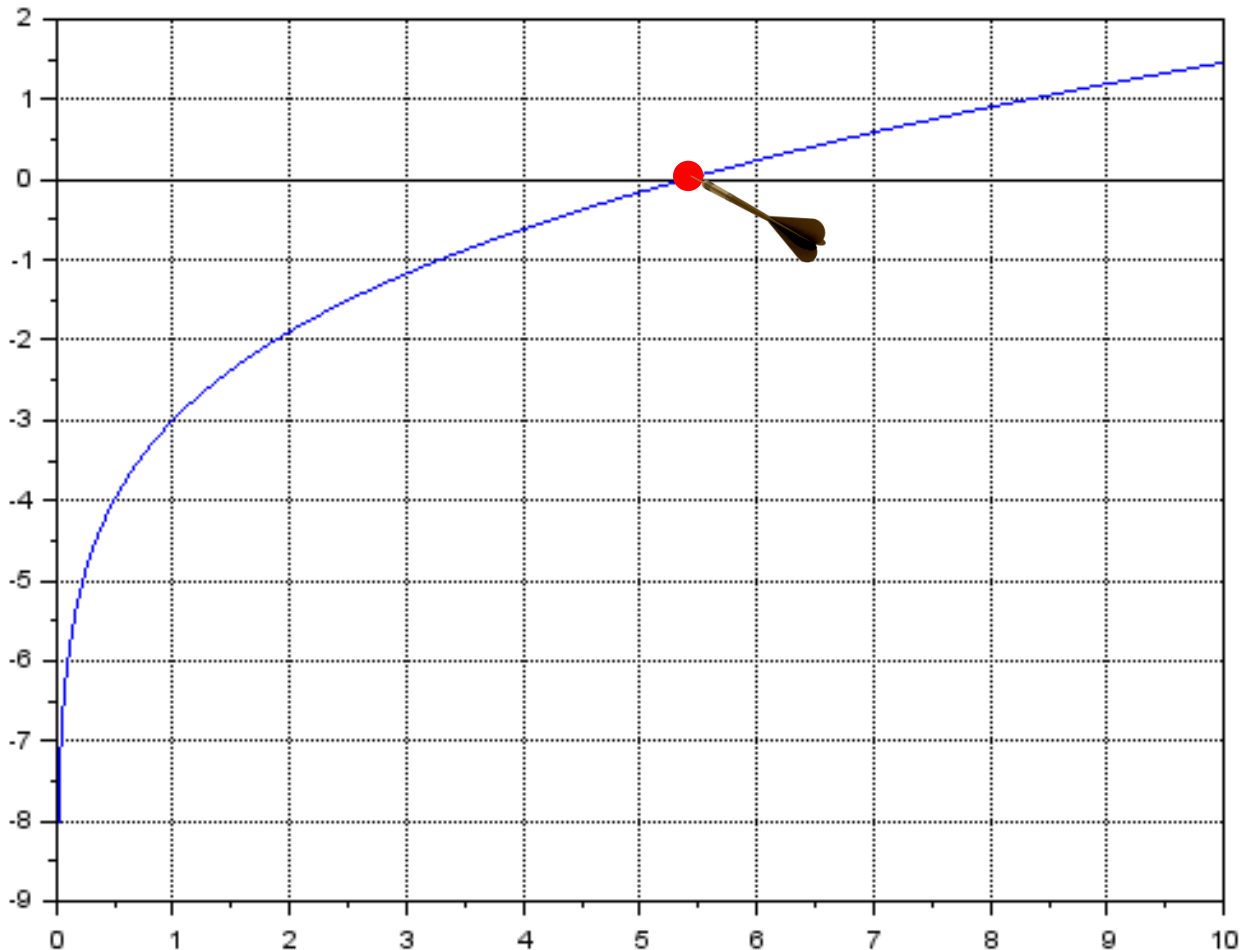
$$\sqrt{x} + \ln(x) - 4 = 0$$

function **y=f(x)**

y=sqrt(x) + log(x) - 4;

endfunction

Para utilizar este método graficamos la ecuación y estimamos visualmente donde se encuentra la raíz

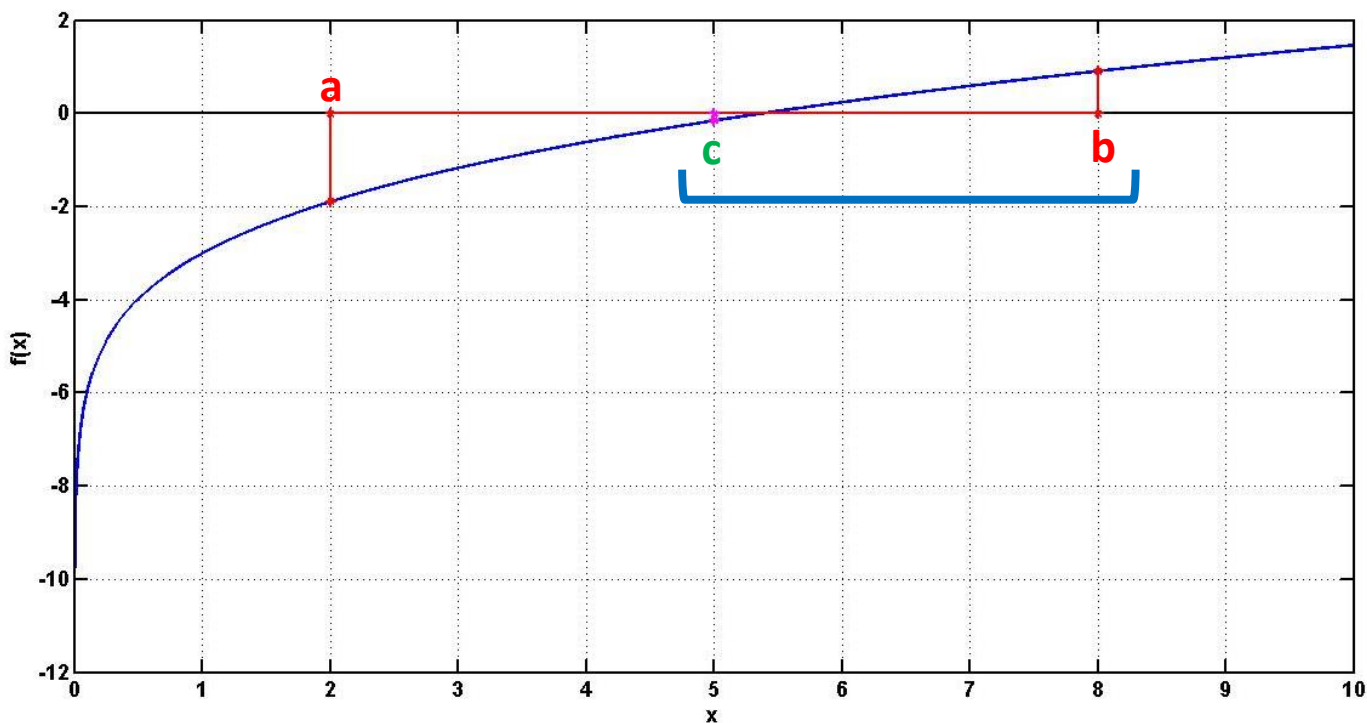


```
x=linspace(0,10,1000);  
plot(x,f(x),'b')  
plot(x,zeros(1:1000),'k')  
xgrid
```

--> xclick
ans =

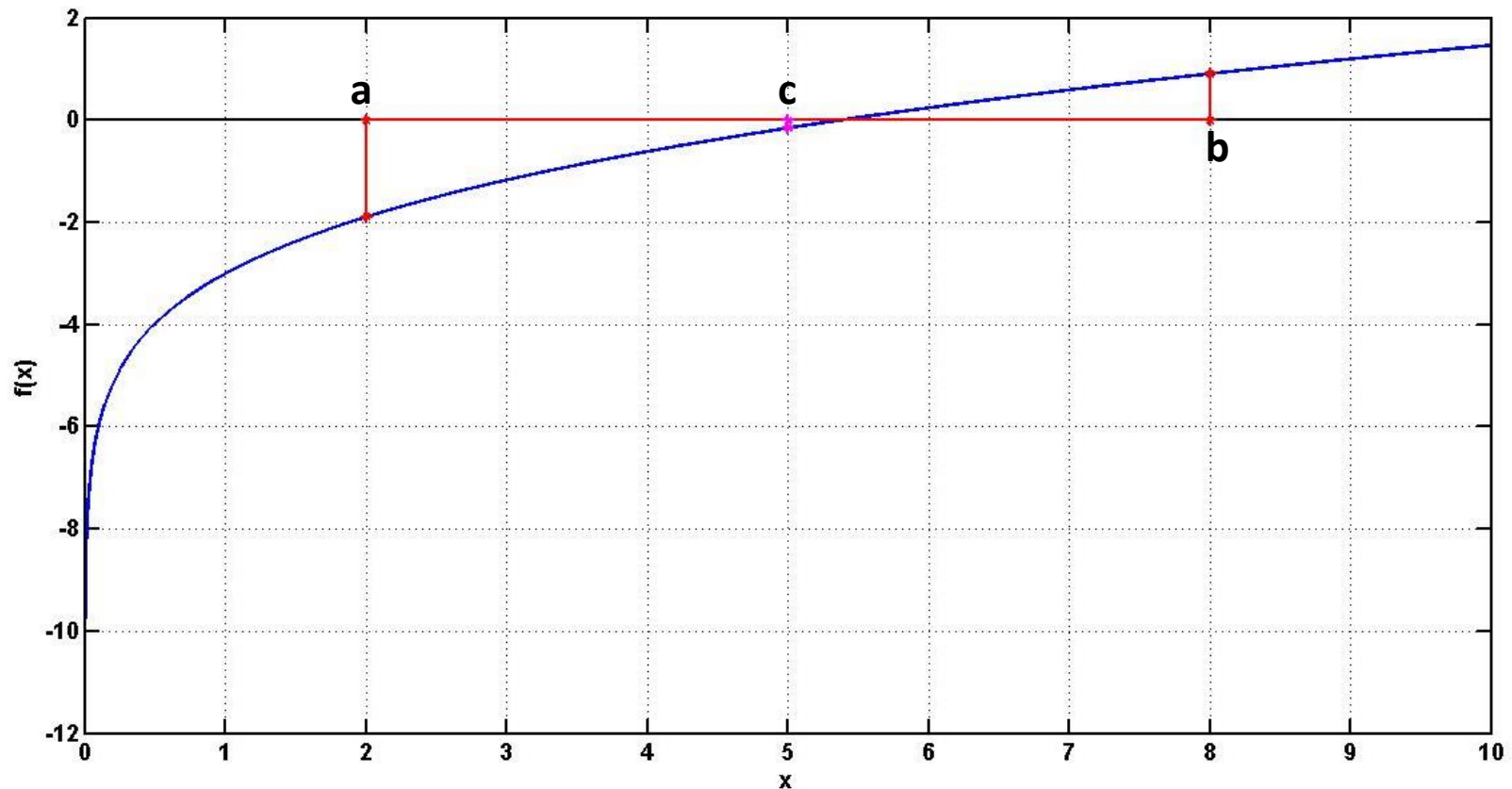
3. 5.3278689 0.0072464

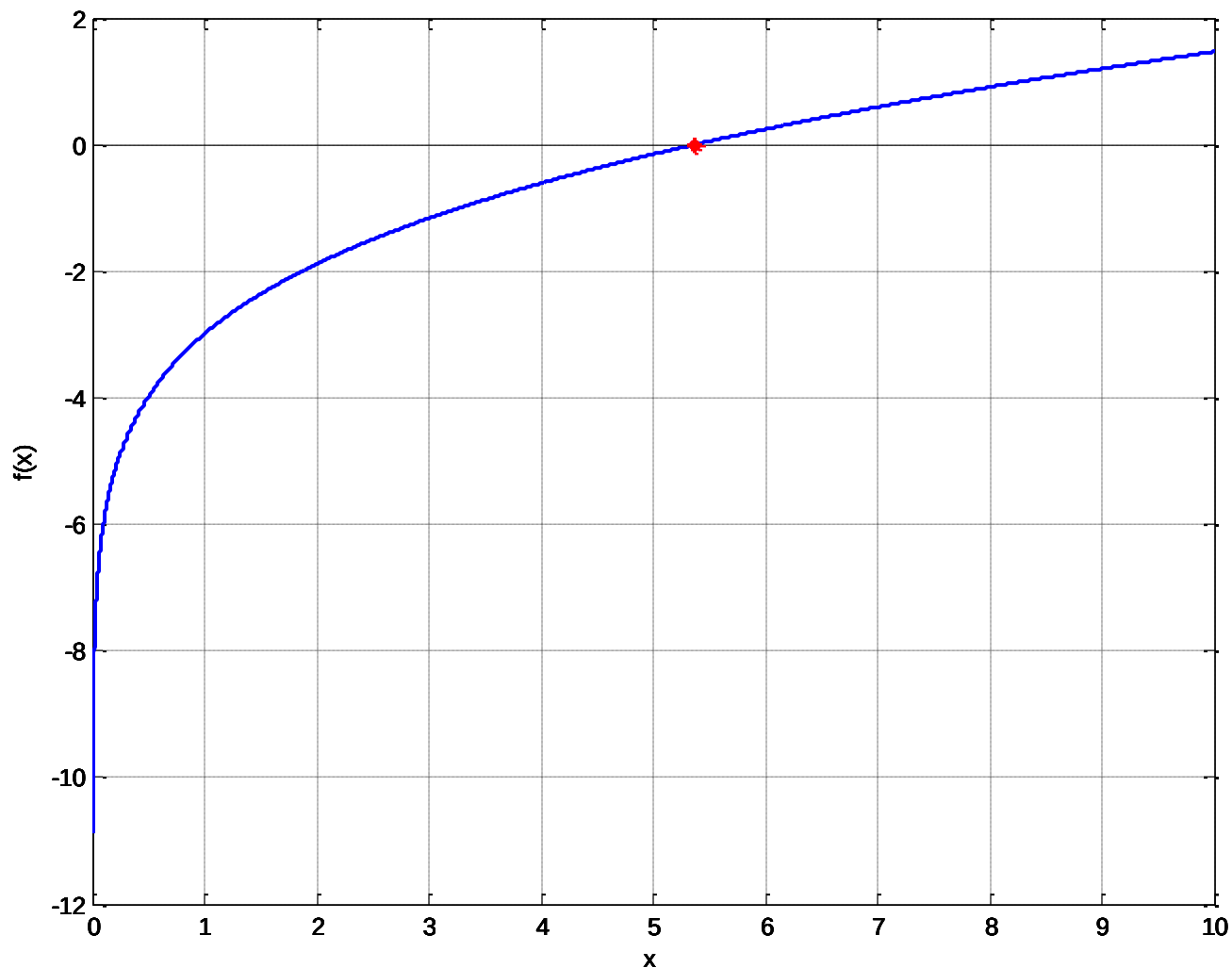
Para utilizar este método necesitamos dos puntos donde la función cambie de signo. El nuevo punto corresponde a la mitad del intervalo y debe seleccionarse de manera de conservar la diferencia de signo.

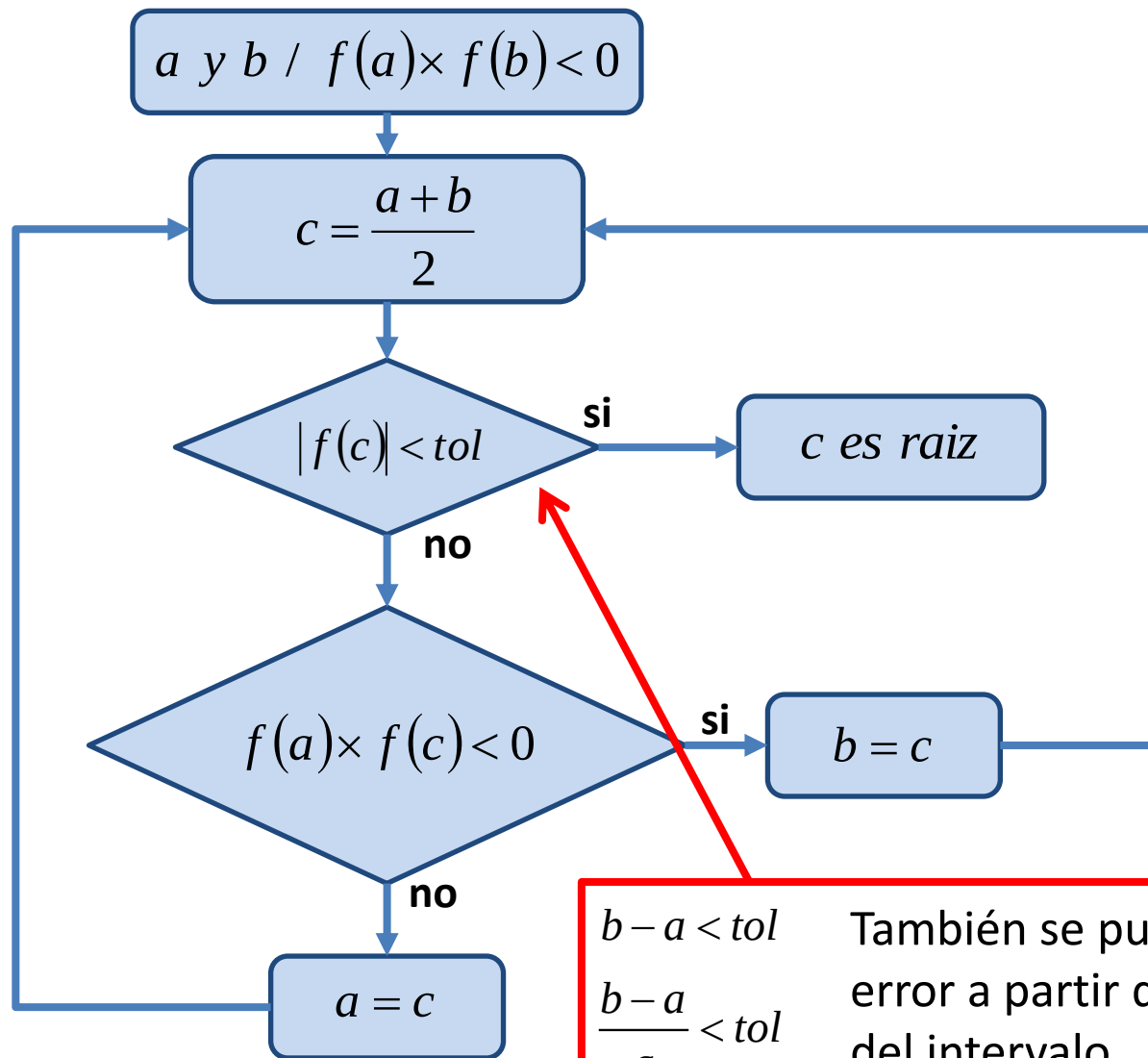


Debemos lograr que el intervalo sea lo mas pequeño posible en torno a la raíz.
¿Cómo lo achicamos?

Evaluamos el valor de la función en la mitad del intervalo
¿Cuál es el nuevo intervalo?







$$b - a < tol$$

$$\frac{b - a}{a} < tol$$

También se puede medir el error a partir del tamaño del intervalo.

a	b	c	f(a)	f(b)	f(c)	error

$$\sqrt{x} + \ln(x) - 4 = 0$$

$$\frac{(b-a)}{2^i}$$

La relación del tamaño del intervalo con el número de iteración nos sirve para conocer de antemano cuantas iteraciones debemos realizar.

Es decir, en la iteración N el tamaño del intervalo debe ser igual a la tolerancia deseada.

$$tol = \frac{(b - a)}{2^N}$$

Ejemplo:

Encontrar el numero de iteraciones necesarias para obtener por el método de la bisección la raíz de la ecuación presentada en el intervalo [5 6] adoptando una tolerancia de $tol=10^{-5}$.

$$tol = \frac{(b - a)}{2^N} \Rightarrow 10^{-5} = \frac{(6 - 5)}{2^N}$$

$$-5 = \log_{10} (6 - 5) - N \log_{10} 2$$

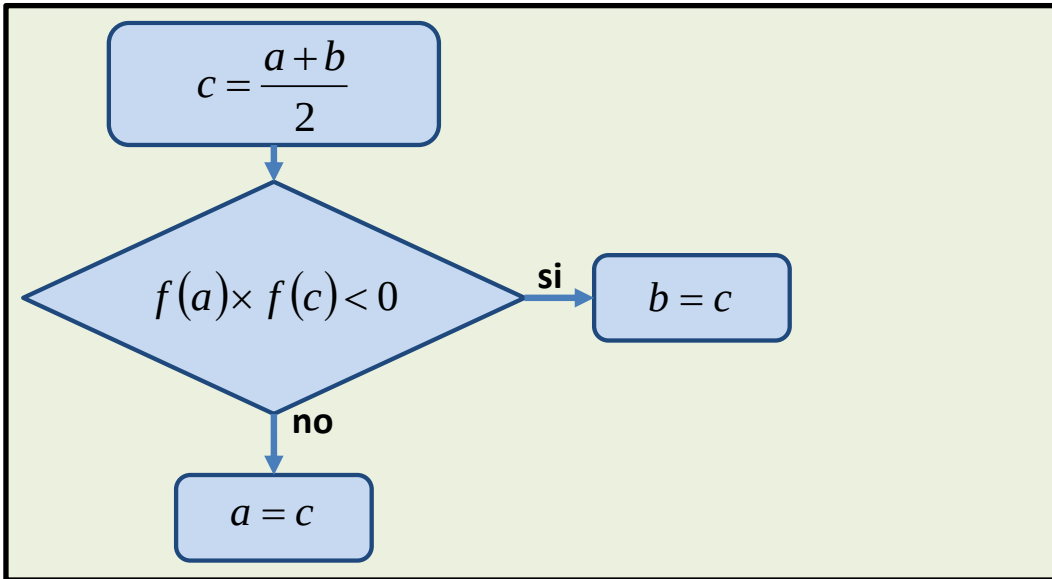
$$N = \frac{\log_{10} (6 - 5) + 5}{\log_{10} 2} \Rightarrow N = 16.609$$

(17 iteraciones)

```
function out=biseccion(fun, a, b, tol)
```

```
N=ceil((log(b-a) - log(tol))/log(2));
```

```
for k=1:N
```

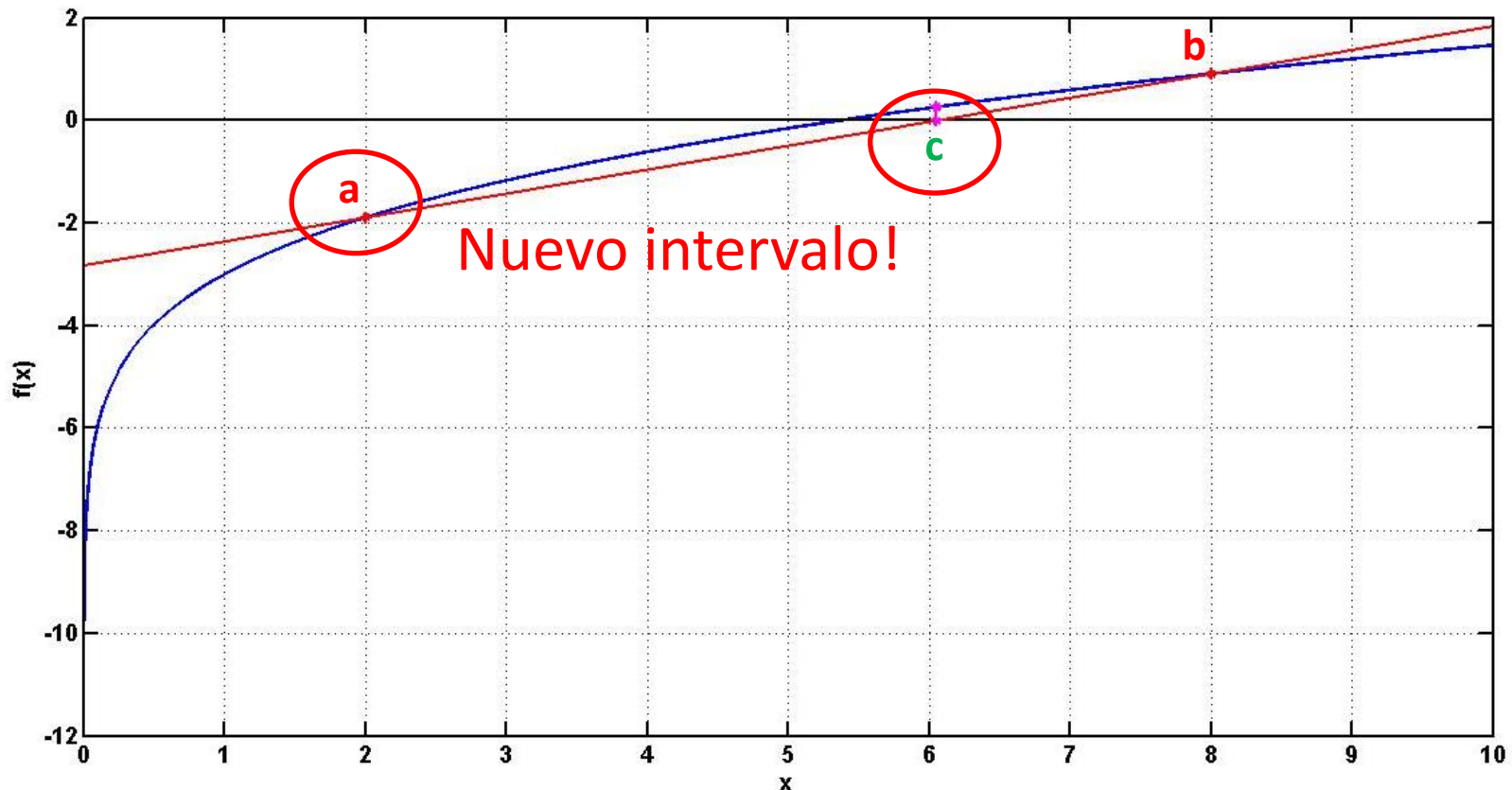


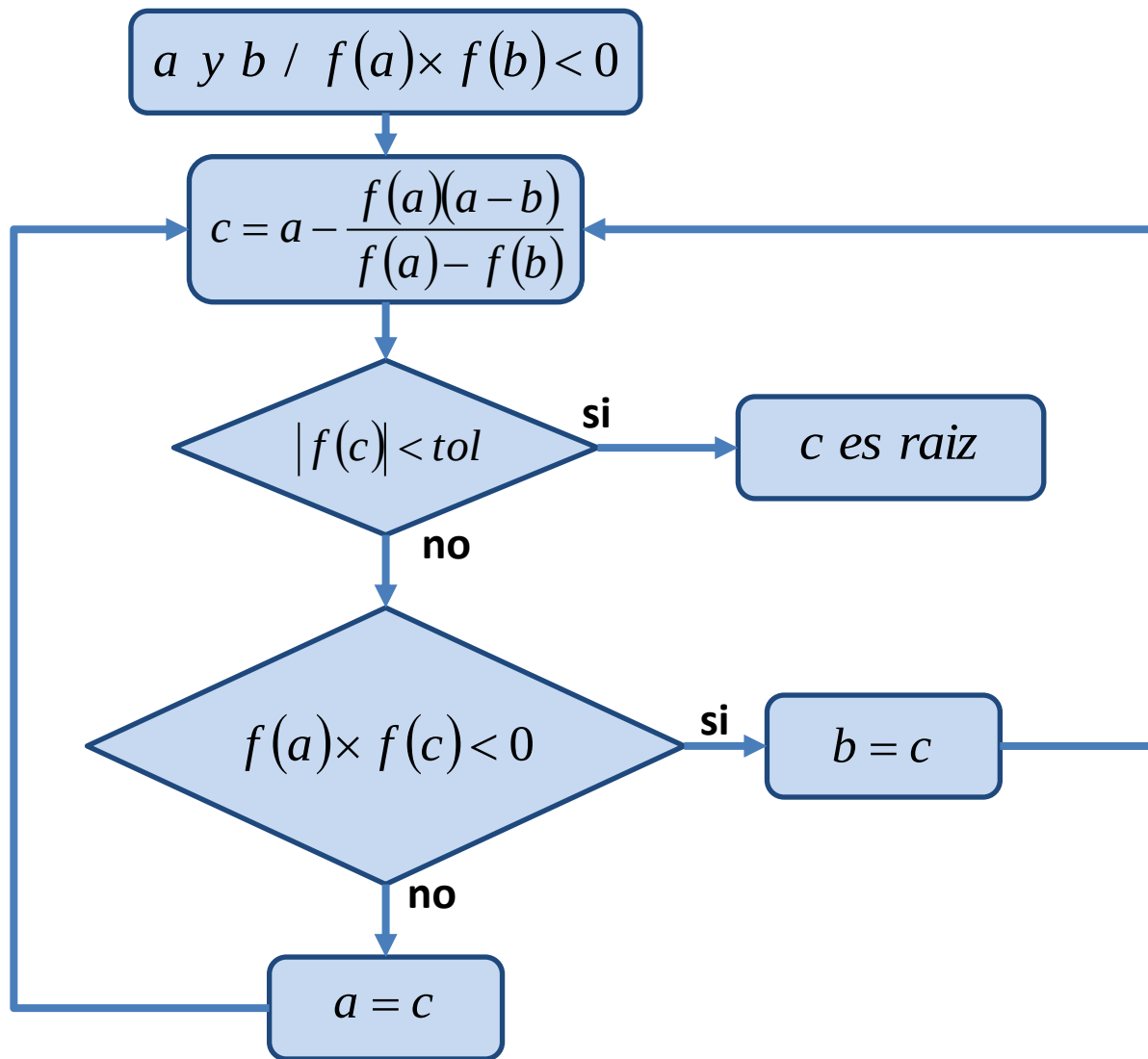
```
end
```

```
out=c;
```

```
endfunction
```

El proceso es similar al de bisección, pero el nuevo punto no corresponde a la mitad del intervalo sino a la intersección de la recta que los une (secante) con el eje de las abscisas.





a	b	f(a)	f(b)	c	error

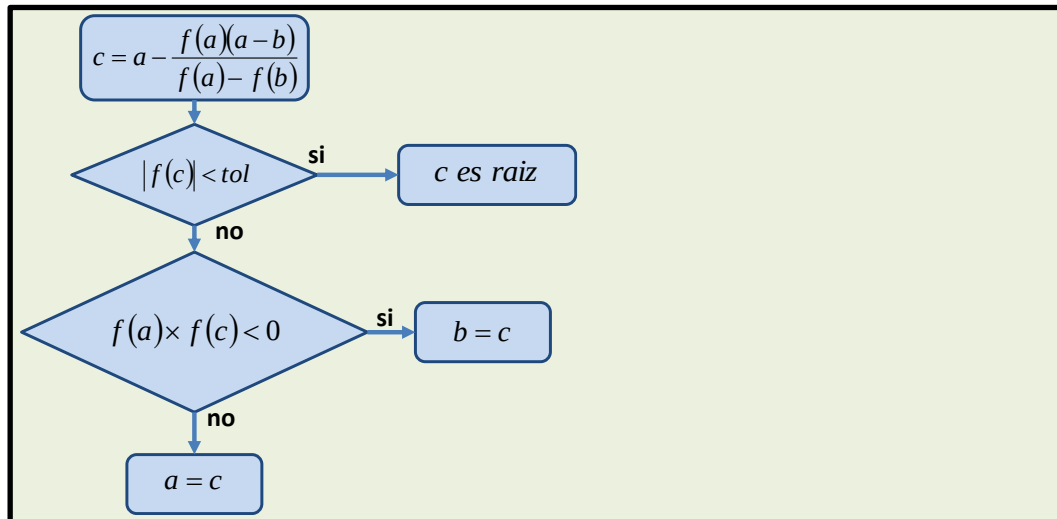
$$\sqrt{x} + \ln(x) - 4 = 0$$

a	b	f(a)	f(b)	c	error

$$f(x) = \ln(x) + e^{2x}$$

function **out**=reglafalsa(**fun**, **a**, **b**, **tol**)

for **k**=1:100



end

if **k** == 100

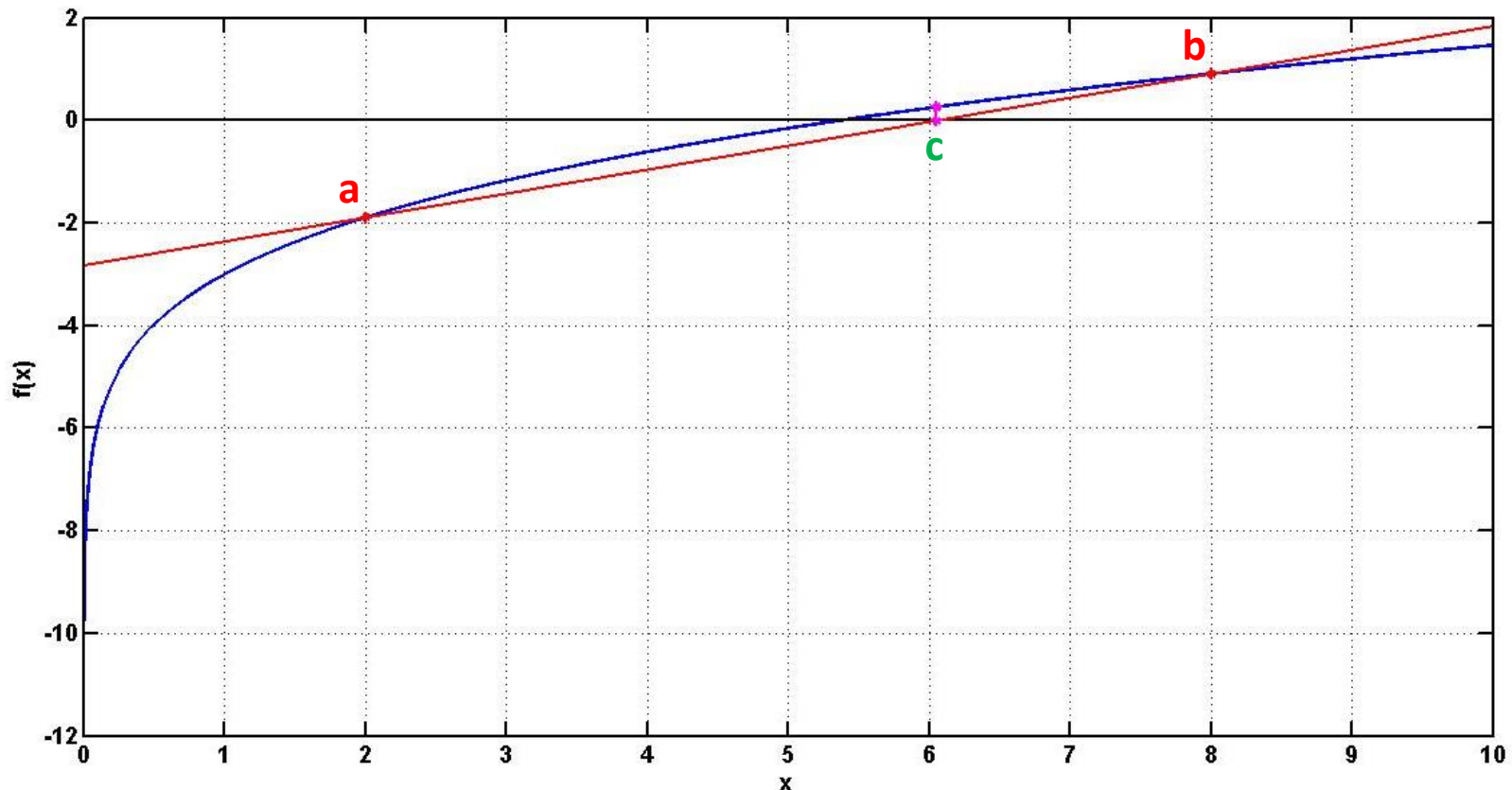
out=[];

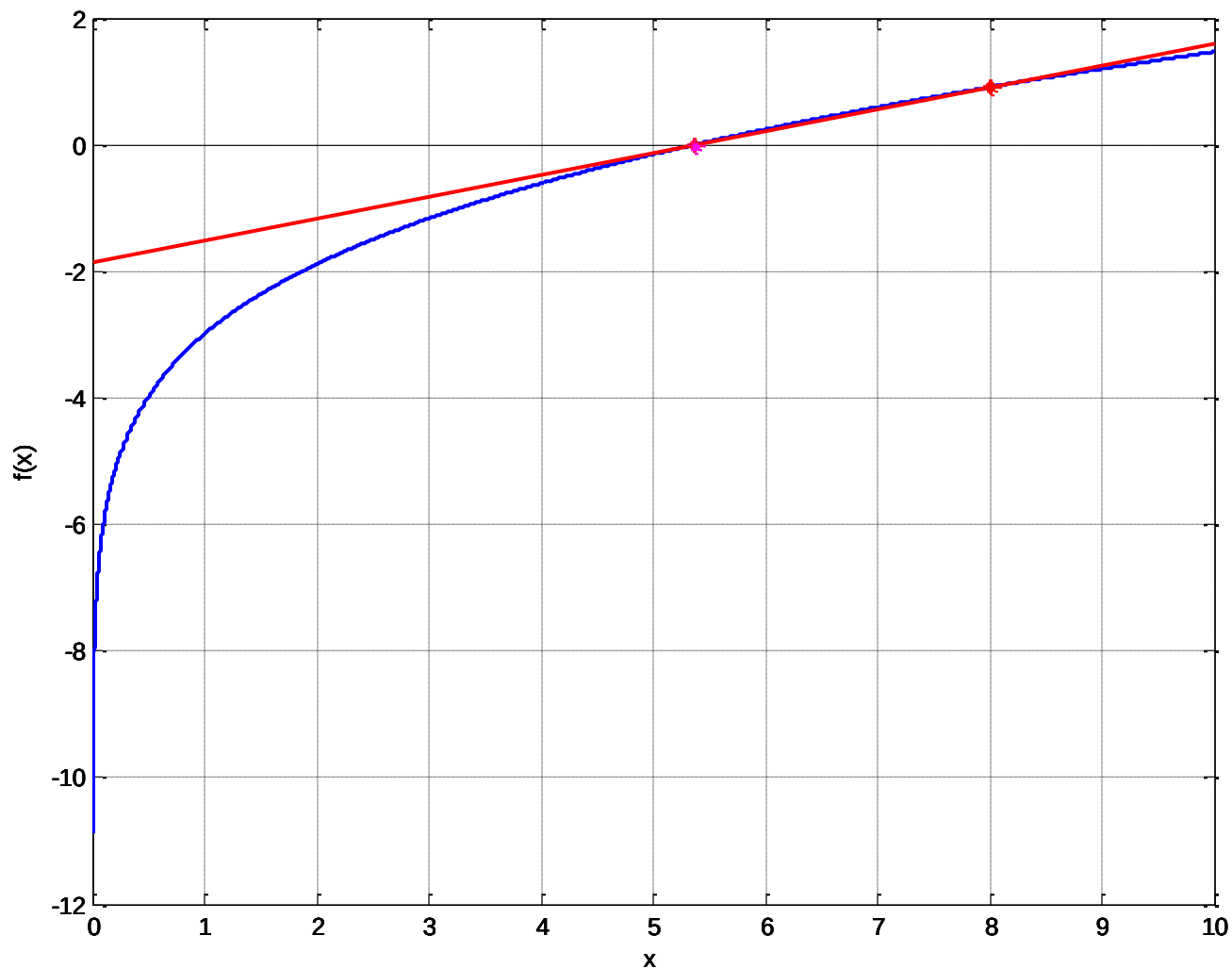
disp('aumentar iteraciones');

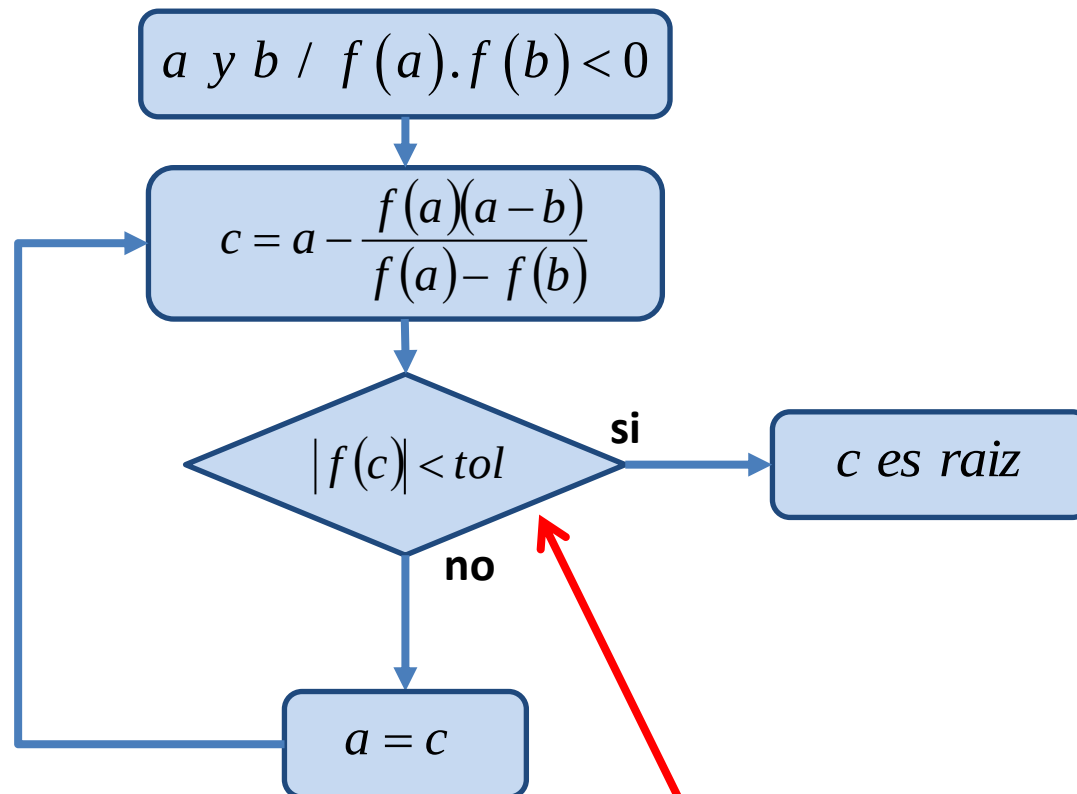
end

endfunction

Para utilizar este método necesitamos dos puntos donde la función cambie de signo. El nuevo punto surge de la intersección de la recta que los une con el eje de las abscisas.







$$|c - a| < tol$$

$$\left| \frac{c - a}{a} \right| < tol$$

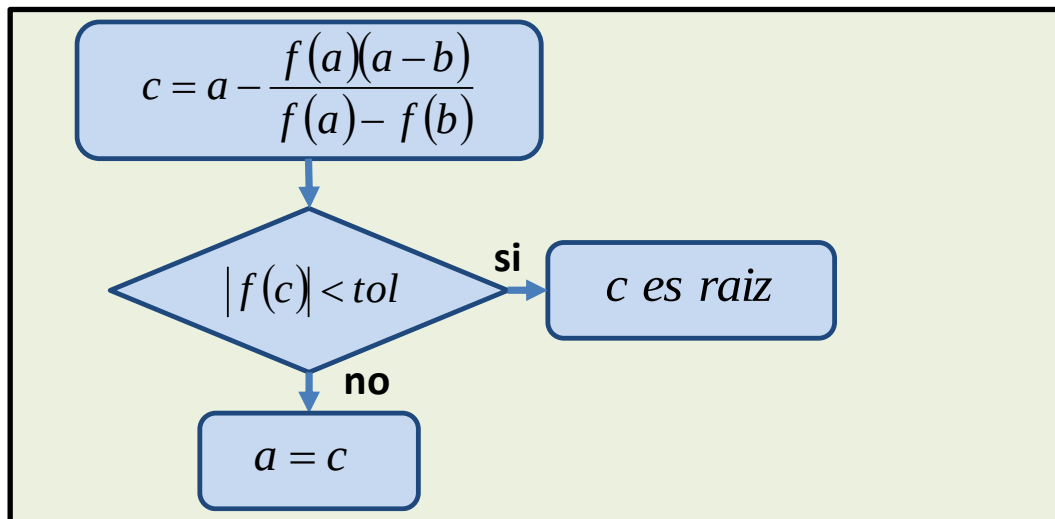
También se puede medir el error a partir del valor de la variable independiente

a	b	f(a)	f(b)	c	error

$$f(x) = \ln(x) + e^{2x}$$

```
function out=reglafalsapf(fun, a, b, tol)
```

```
for k=1:100
```



```
end
```

```
if k == 100
```

```
    out=[];
```

```
    disp('no converge');
```

```
end
```

```
endfunction
```

A los fines de su consideración en la resolución de problemas de ingeniería, podemos agruparlos en dos categorías:

- Método de Aproximaciones Sucesivas
- Métodos de Linealización
 - Newton – Raphson
 - Modificado de Newton para Resolver Raíces Múltiples
 - Von Mises o Cuerdas Paralelas
 - Secante
 - Regula falsi y métodos relacionados.

Para utilizar este método necesitamos un punto y redefinir la ecuación de la siguiente manera:

Dada la ecuación original del tipo:

$$f(x) = 0$$

Debemos obtener una expresión en donde la incógnita es una función de si misma:

$$x = F(x)$$

Sea la función: $y = f(x)$

Se desea hallar x^* tal que $y = f(x^*) = 0$

Se debe definir una nueva función $F(x)$ del tipo: $x = F(x)$

$F(x)$ $\left\{ \begin{array}{l} \text{Se obtiene sumando miembro a miembro en la expresión original } x \\ \text{Despejando } x \text{ de algún termino} \end{array} \right.$

$$f(x) = \sqrt{x} + \ln(x) - 4 \quad \left\{ \begin{array}{l} F(x) = \sqrt{x} + \ln(x) - 4 + x \\ F(x) = (4 - \ln(x))^2 \\ F(x) = e^{(4 - \sqrt{x})} \end{array} \right.$$

A partir de un valor inicial (punto de arranque o valor semilla) se genera un nuevo valor utilizando $F(x)$

$$x^{(0)} = \alpha_0$$

$$x^{(1)} = F\left(x^{(0)}\right)$$

$$x^{(2)} = F\left(x^{(1)}\right)$$

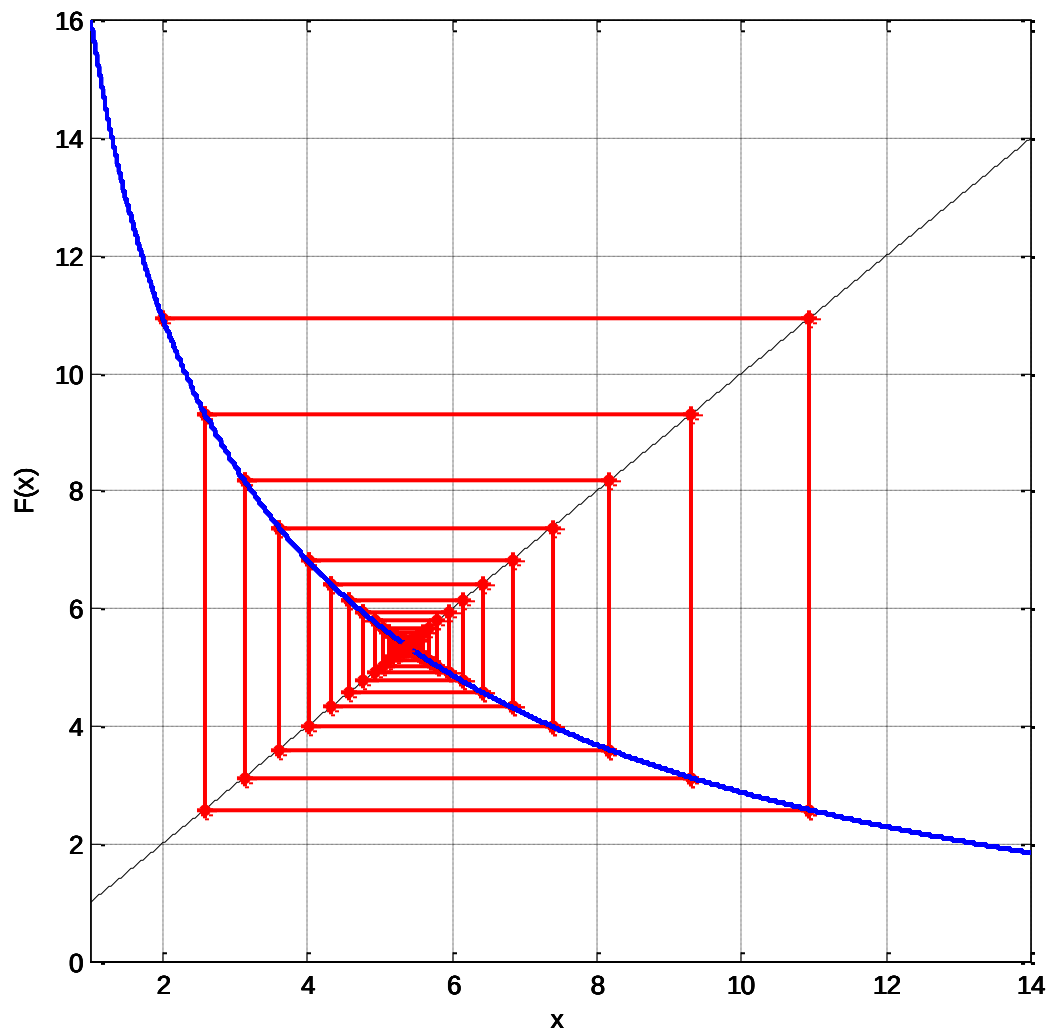
$$x^{(k+1)} = F\left(x^{(k)}\right)$$

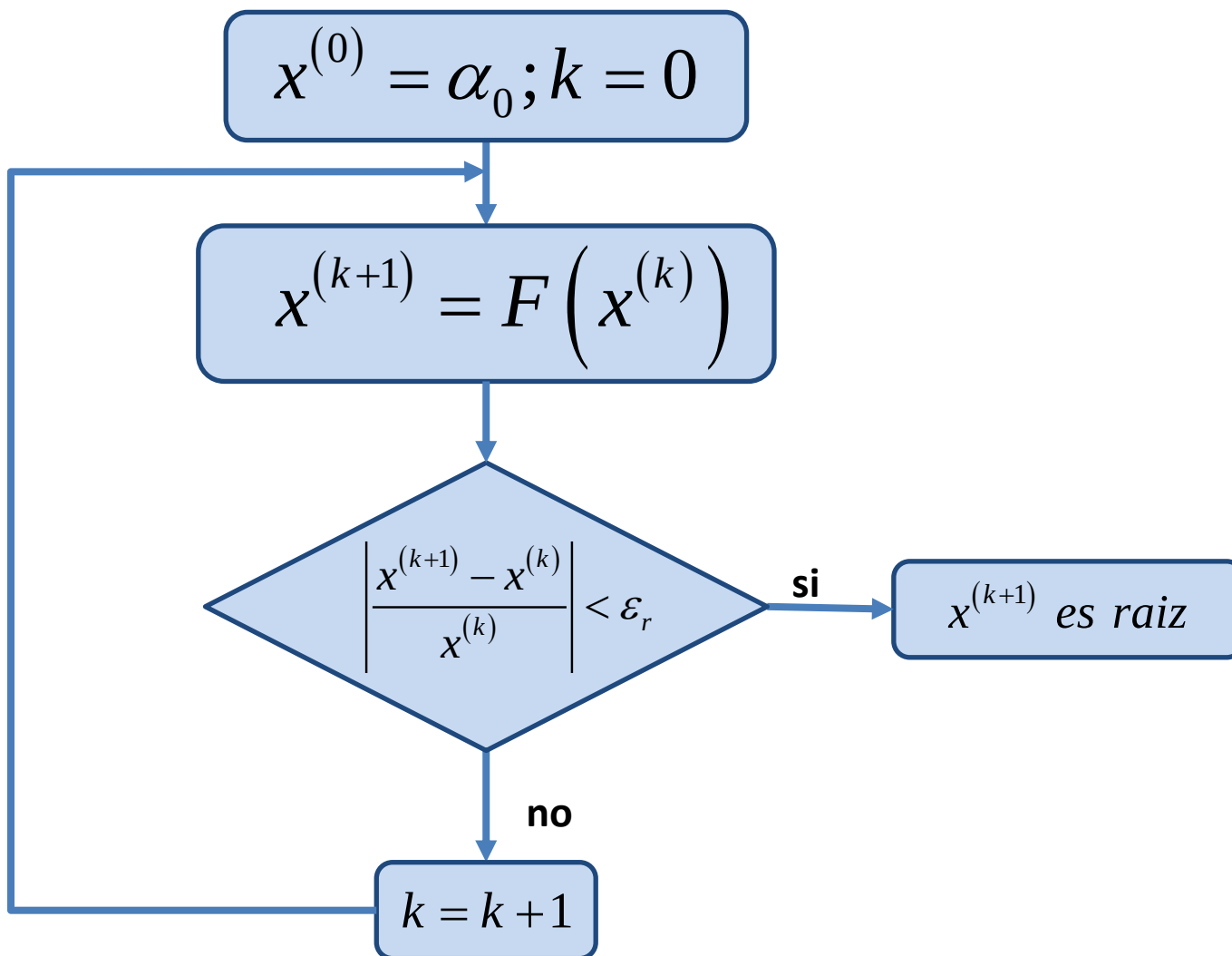
$$k = 1, 2, \dots, k_{\max}$$

$$\left| \frac{x^{(k+1)} - x^{(k)}}{x^{(k)}} \right| < \varepsilon_r$$

$$\left| x^{(k+1)} - F\left(x^{(k+1)}\right) \right| < \varepsilon$$

Se generan valores hasta satisfacer la tolerancia del error o alcanzar el número máximo de iteraciones





i	x_i	$F(x_i)$	<i>Error</i>
0			
1			
2			
3			
4			
5			
6			

$$F(x) = e^{(4-\sqrt{x})}$$

$$F(x) = (4 - \ln(x))^2$$

$$F(x) = \sqrt{x} + \ln(x) - 4 + x$$

```
function y=F(x)
```

```
y= sqrt(x) + log(x) - 4 + x;
```

```
endfunction
```

```
function y=F(x)
```

```
y= (4 - log(x)).^2; ————— Funciones de aproximación
```

```
endfunction
```

```
function y=F(x)
```

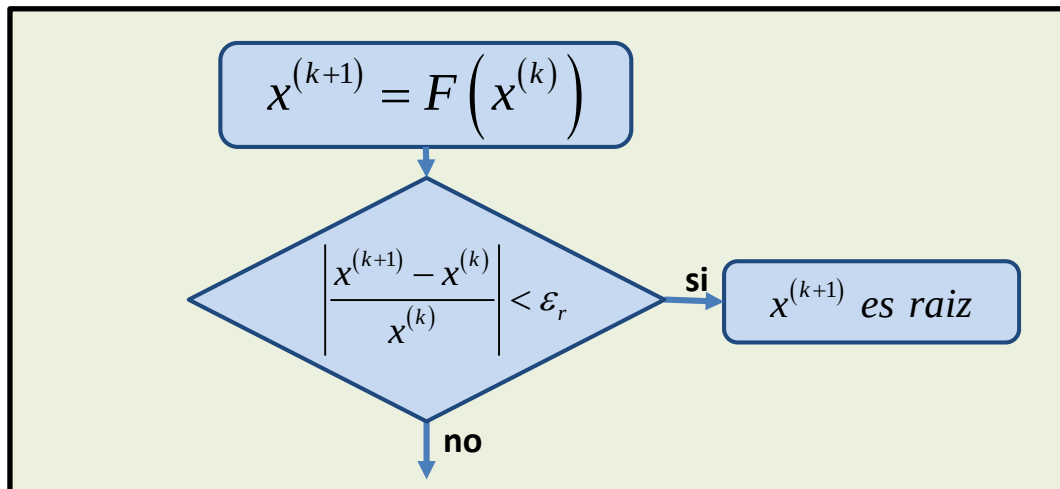
```
y= exp (4 - sqrt(x));
```

```
endfunction
```

```
function out=aproxsuc(fun, x0, tol)
```

```
x=x0;
```

```
for k=1:100
```



```
end
```

```
if k == 100
```

```
    out=[];
```

```
    disp('no converge');
```

```
end
```

```
endfunction
```

Mejora el proceso de sustitución directa.

A partir del valor inicial se generan un valor de manera tradicional y luego se aplica la siguiente ley recursiva:

$$\omega = \frac{F(x^{(k)}) - F(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

$$q = \frac{\omega}{\omega - 1}$$

$$x^{(k+1)} = qx^{(k)} + (1 - q)F(x^{(k)})$$

$$k = 1, 2, 3, \dots, k_{\max}$$

$$\left| \frac{x^{(k+1)} - x^{(k)}}{x^{(k)}} \right| < \varepsilon_r$$

$$\left| x^{(k+1)} - F(x^{(k+1)}) \right| < \varepsilon$$

Se generan valores hasta satisfacer la tolerancia del error o alcanzar el número máximo de iteraciones

Paso a paso para una mejor comprensión:

$$x^{(0)} = \alpha_0$$

$$x^{(1)} = F(x^{(0)})$$

$$\omega = \frac{F(x^{(1)}) - F(x^{(0)})}{x^{(1)} - x^{(0)}}$$

$$q = \frac{\omega}{\omega - 1}$$

$$x^{(2)} = qx^{(1)} + (1 - q)F(x^{(1)})$$

$$\omega = \frac{F(x^{(k)}) - F(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

$$q = \frac{\omega}{\omega - 1}$$

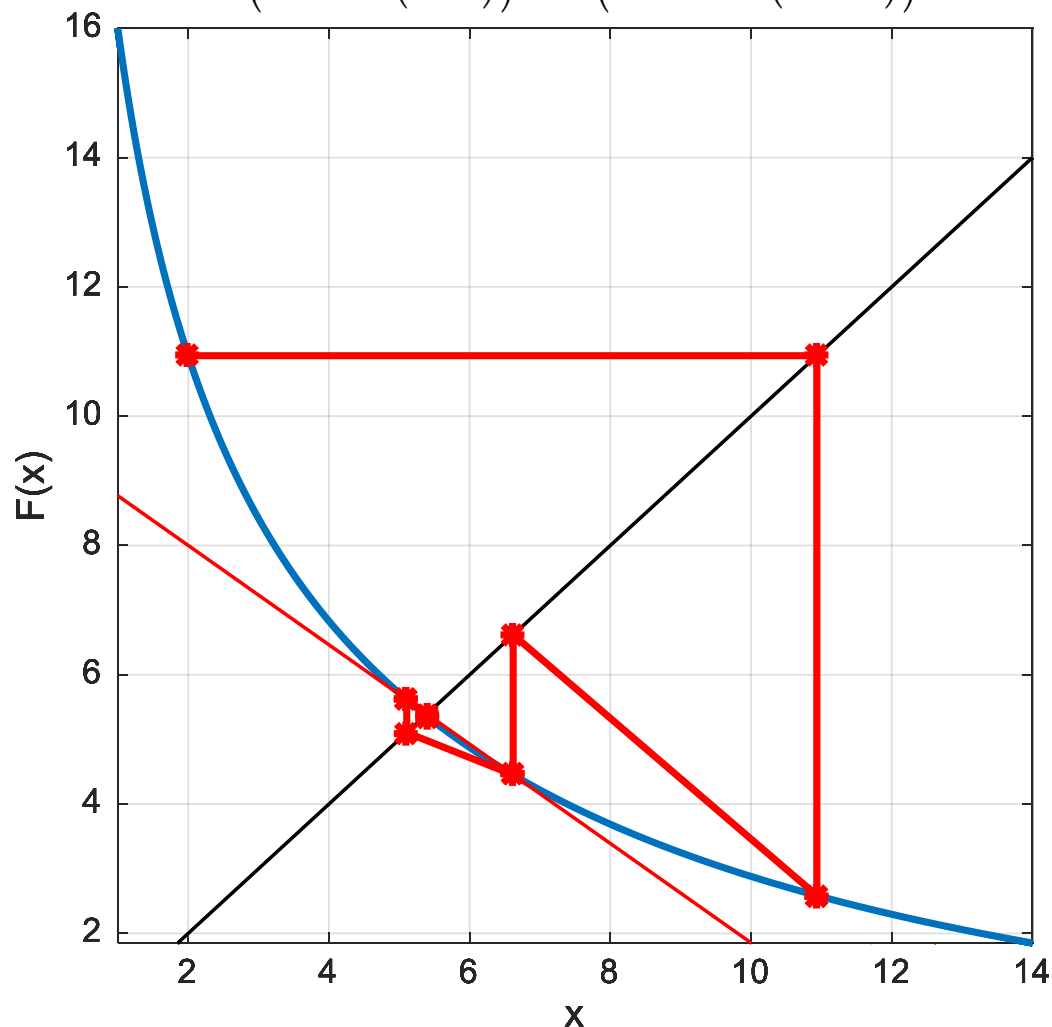
$$x^{(k+1)} = qx^{(k)} + (1 - q)F(x^{(k)})$$

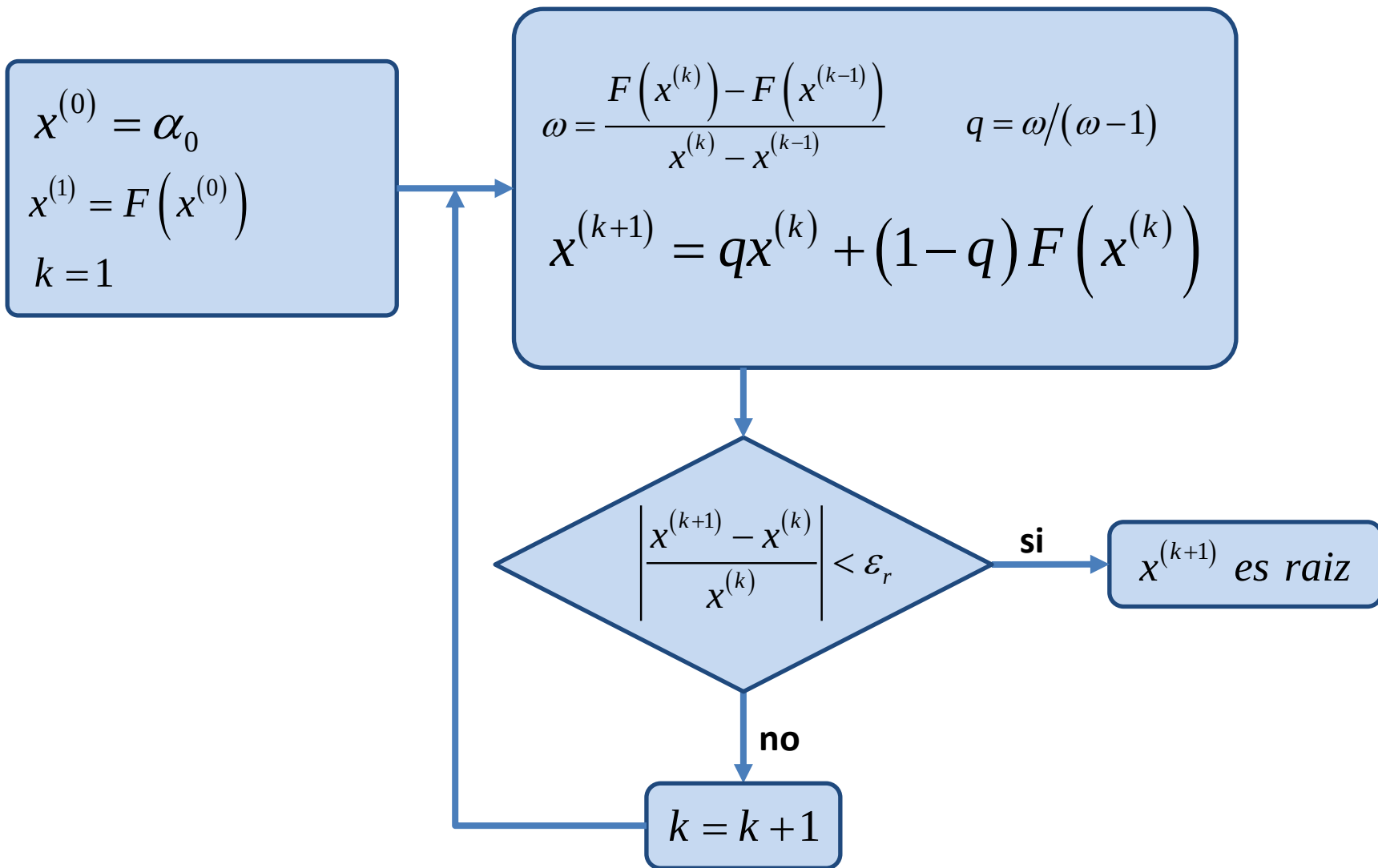
$$k = 1, 2, 3, \dots, k_{\max}$$

$$\left| \frac{x^{(k+1)} - x^{(k)}}{x^{(k)}} \right| < \varepsilon_r$$

$$\left| x^{(k+1)} - F(x^{(k+1)}) \right| < \varepsilon$$

El nuevo valor generado en la iteración k , o sea $x^{(k+1)}$ corresponde a la intersección de la recta que une los puntos $(x^{(k)}, F(x^{(k)}))$ y $(x^{(k-1)}, F(x^{(k-1)}))$ con la recta $y=x$





i	x_i	$F(x_i)$	ω	q	<i>Error</i>
0					
1					
2					
3					
4					
5					
6					

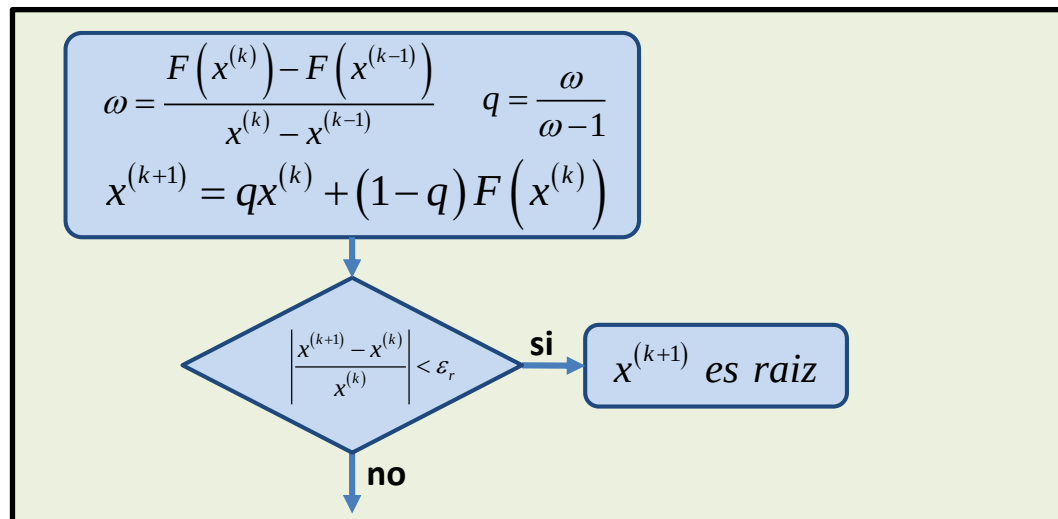
```
function out=wegstein(fun, x0, tol)
```

```
x(1)=x0;
```

```
x(2)=fun(x(1));
```

```
FF=fun(x);
```

```
for k=2:100
```



```
end
```

```
if k == 100
```

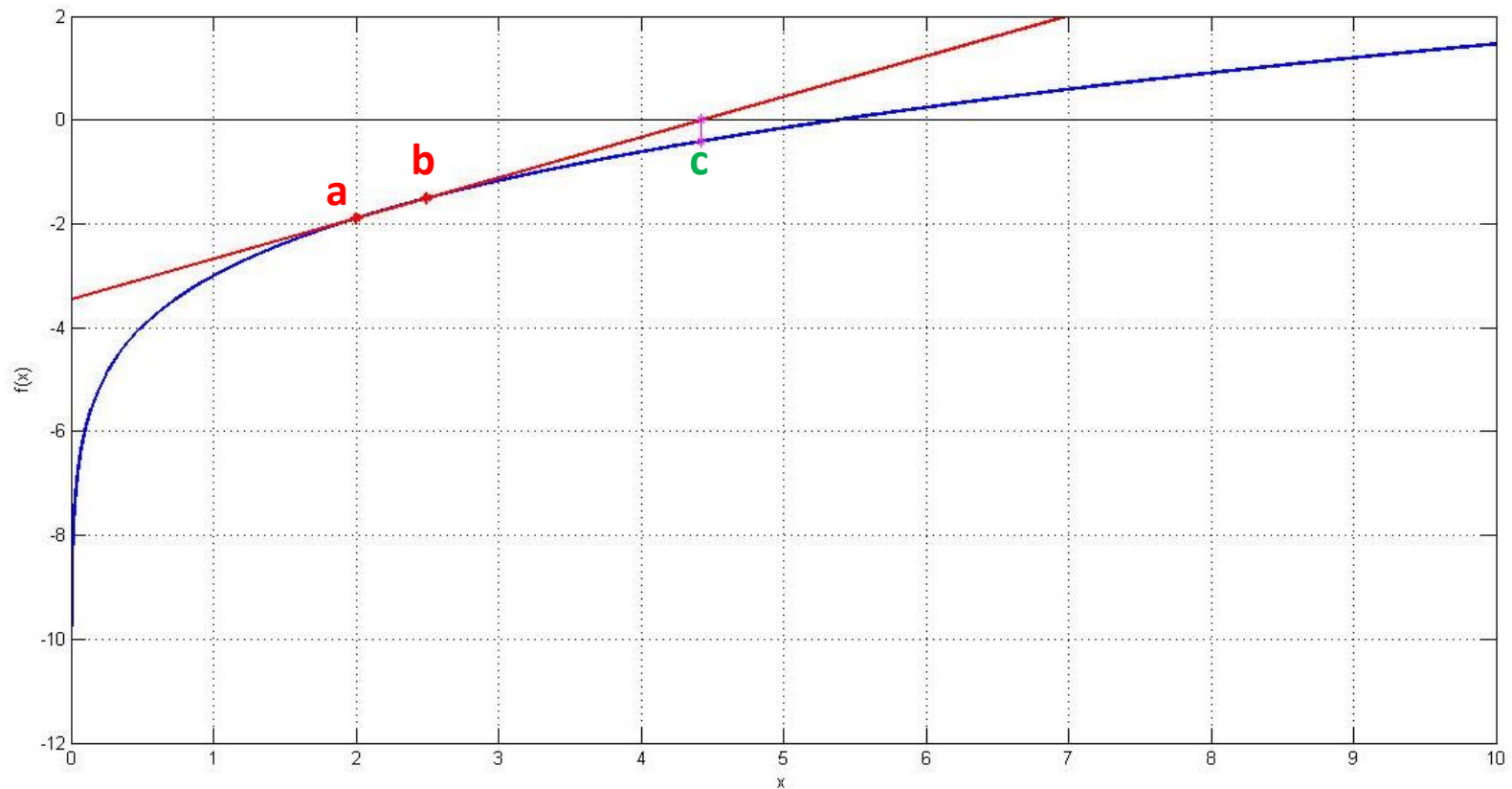
```
    out=[];
```

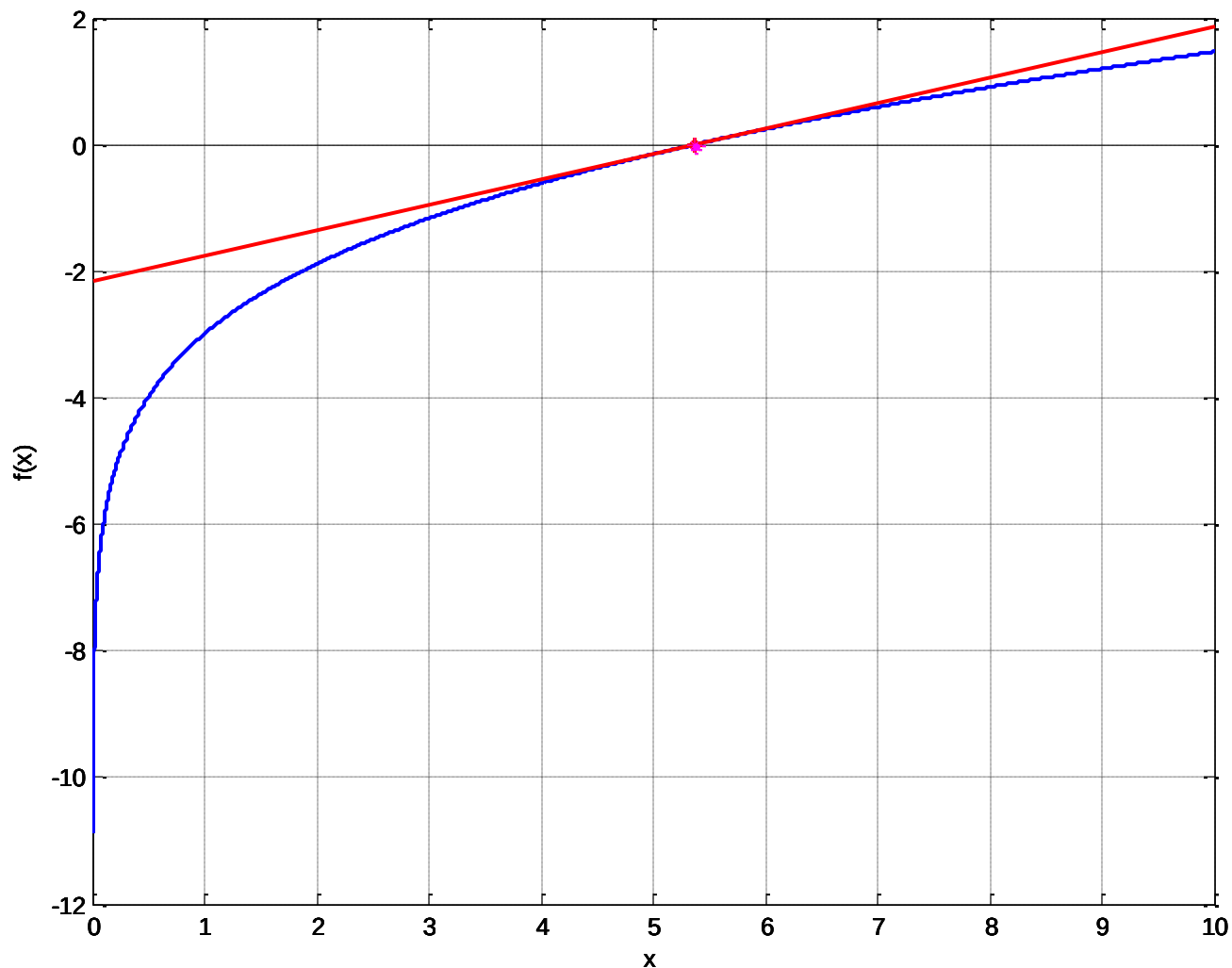
```
    disp('no converge');
```

```
end
```

```
endfunction
```

Para utilizar este método solo necesitamos dos puntos y la recta que los une. El nuevo punto surge de su intersección con el eje de las abscisas.





A partir de dos valores iniciales (puntos de arranque) se genera un nuevo valor que corresponde a la intersección de la secante a la curva con el eje de las abscisas.

$$x^{(0)} = \alpha_0$$

$$x^{(1)} = \alpha_1$$

$$x^{(2)} = x^{(1)} - \frac{f(x^{(1)})(x^{(1)} - x^{(0)})}{f(x^{(1)}) - f(x^{(0)})}$$

$$x^{(3)} = x^{(2)} - \frac{f(x^{(2)})(x^{(2)} - x^{(1)})}{f(x^{(2)}) - f(x^{(1)})}$$

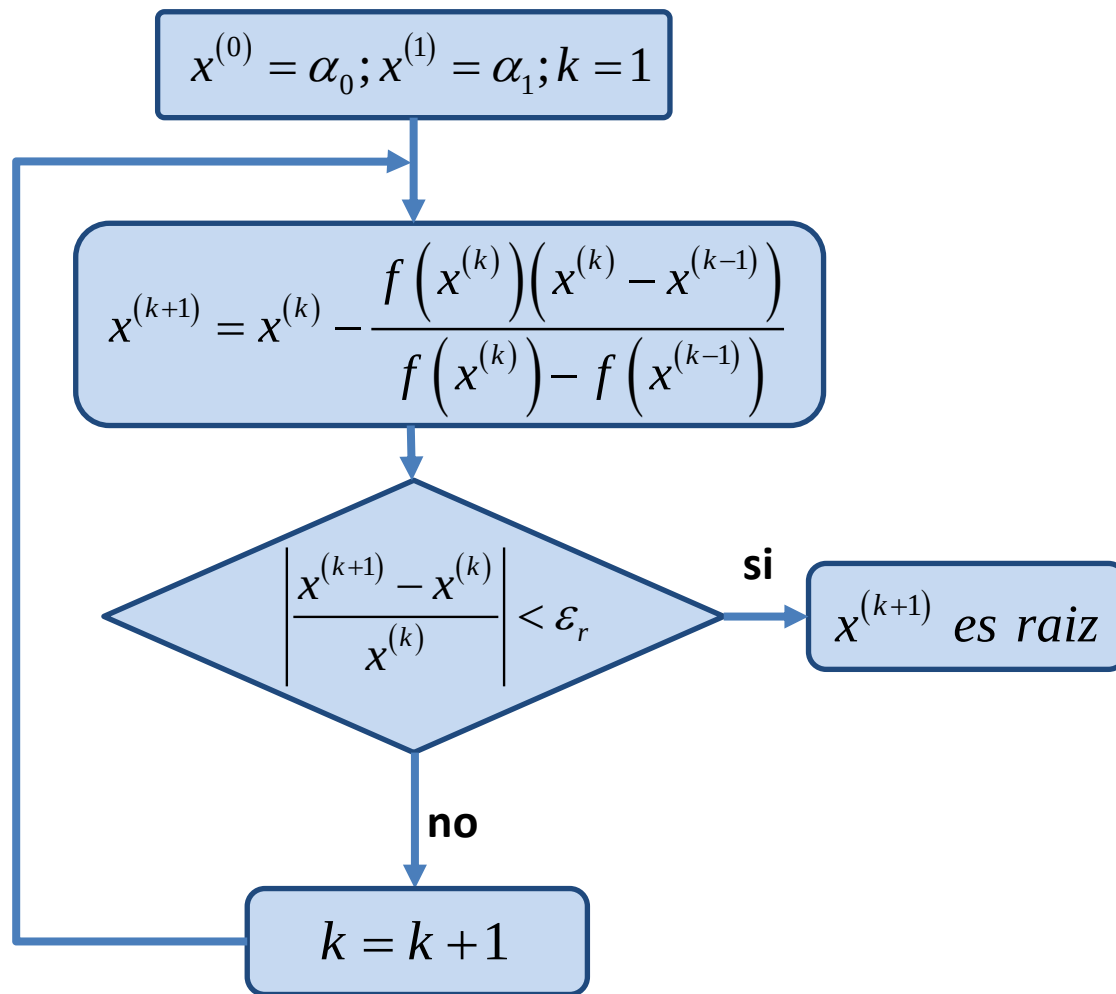
$$k = 1, 2, \dots, k_{\max}$$

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})(x^{(k)} - x^{(k-1)})}{f(x^{(k)}) - f(x^{(k-1)})}$$

$$\left| \frac{x^{(k+1)} - x^{(k)}}{x^{(k)}} \right| < \varepsilon_r$$

$$\left| f(x^{(k+1)}) \right| < \varepsilon$$

Se generan valores hasta satisfacer la tolerancia del error o alcanzar el número máximo de iteraciones



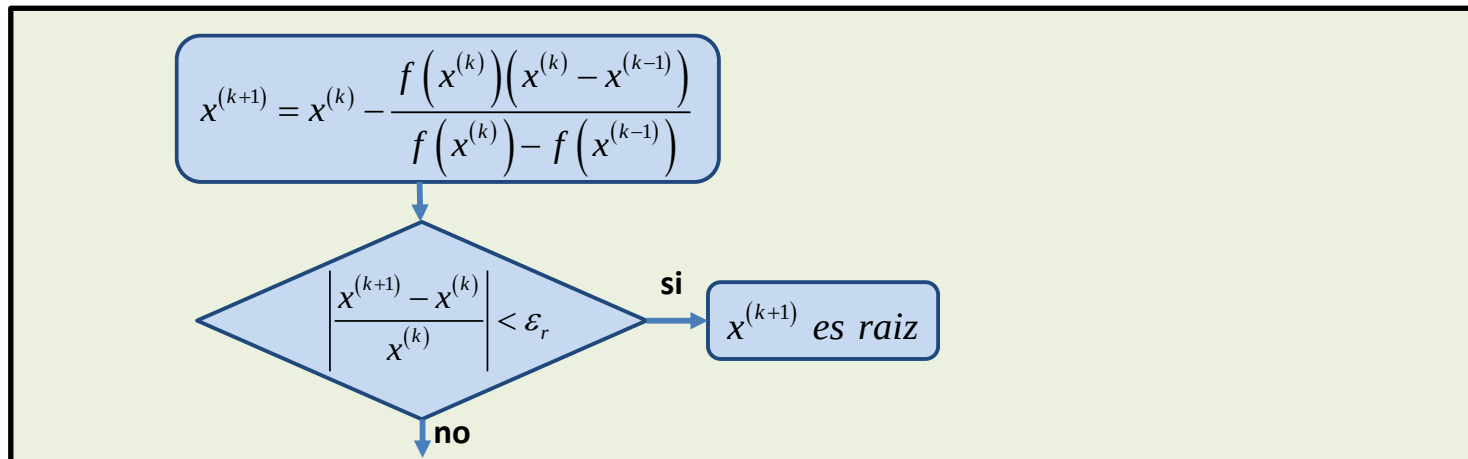
i	x_i	$f(x_i)$	<i>Error</i>
0			
1			
2			
3			
4			
5			
6		$f(x) = \ln(x) + e^{2x}$	

```
function out=secante(fun, x0, x1, tol)
```

```
x(1)=x0;
```

```
x(2)=x1;
```

```
for k=2:100
```



```
end
```

```
if k == 100
```

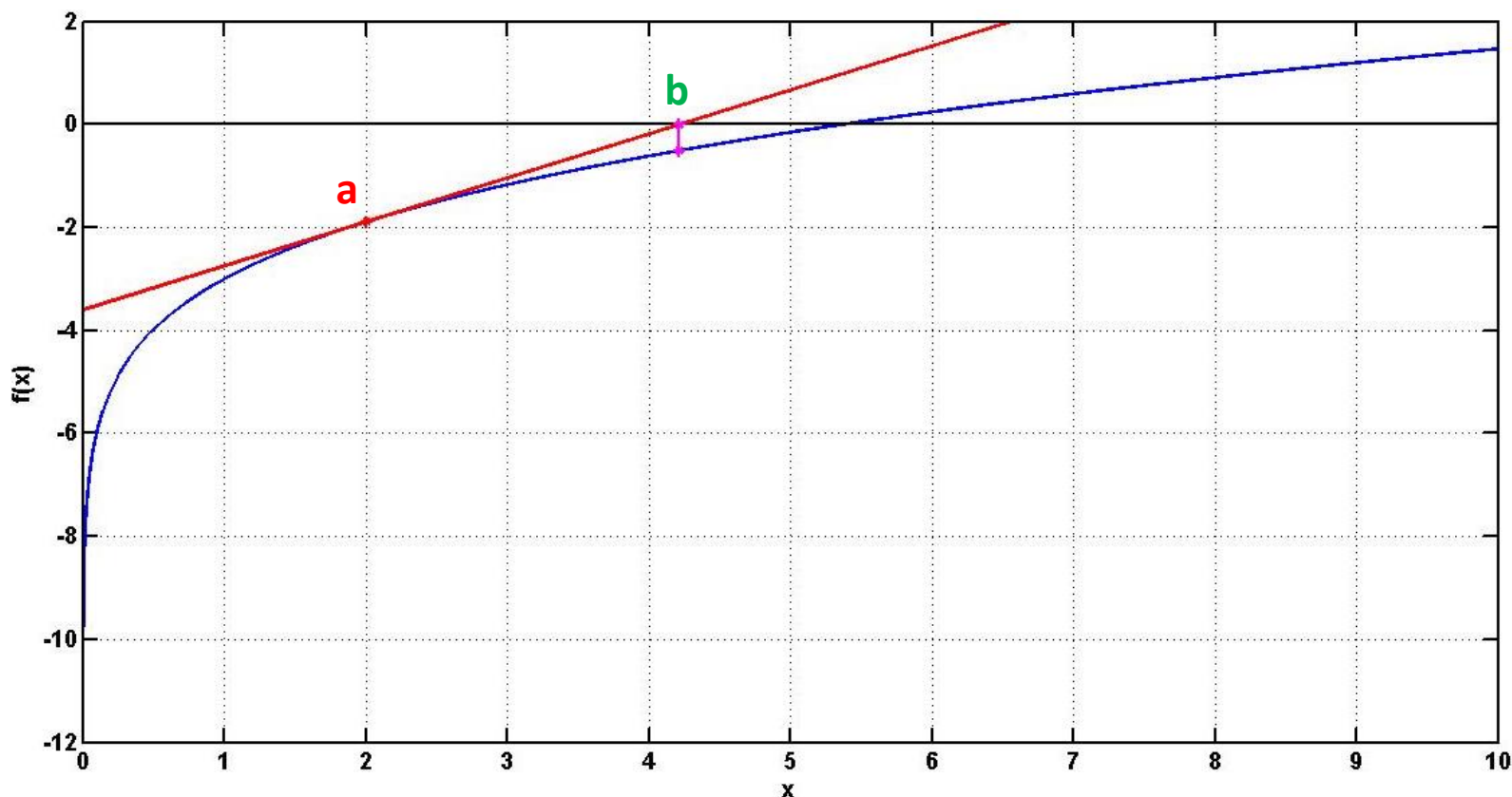
```
    out=[];
```

```
    disp('no converge');
```

```
end
```

```
endfunction
```

Para utilizar este método solo necesitamos un punto y la derivada de la función. El nuevo punto surge de la intersección de la recta tangente con el eje de las abscisas.



A partir de un valor inicial (punto de arranque o valor semilla) se genera un nuevo valor que corresponde a la intersección de la recta tangente a la curva con el eje de las abscisas.

$$x^{(0)} = \alpha_0$$

$$x^{(1)} = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})}$$

$$x^{(2)} = x^{(1)} - \frac{f(x^{(1)})}{f'(x^{(1)})}$$

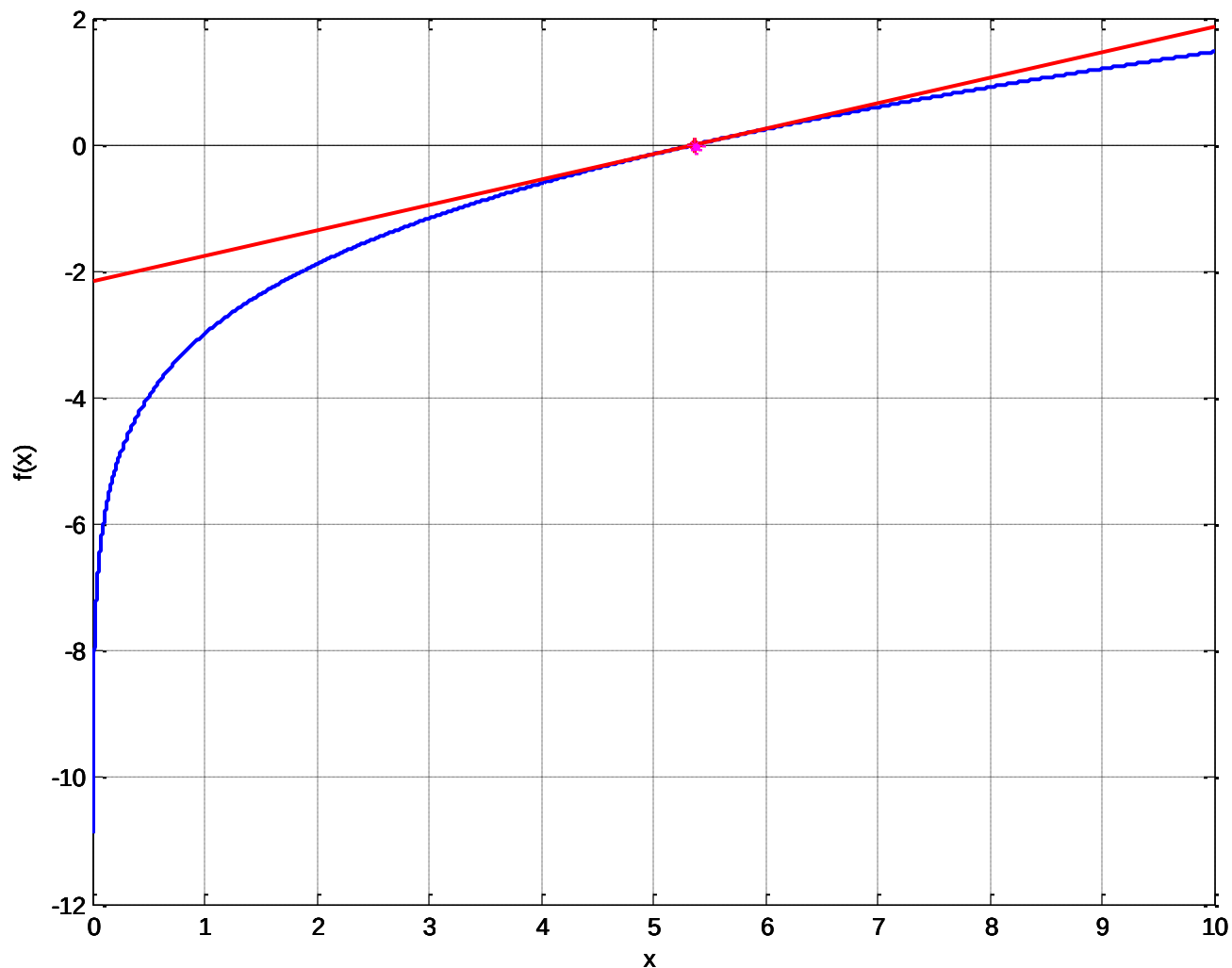
$$k = 1, 2, \dots, k_{\max}$$

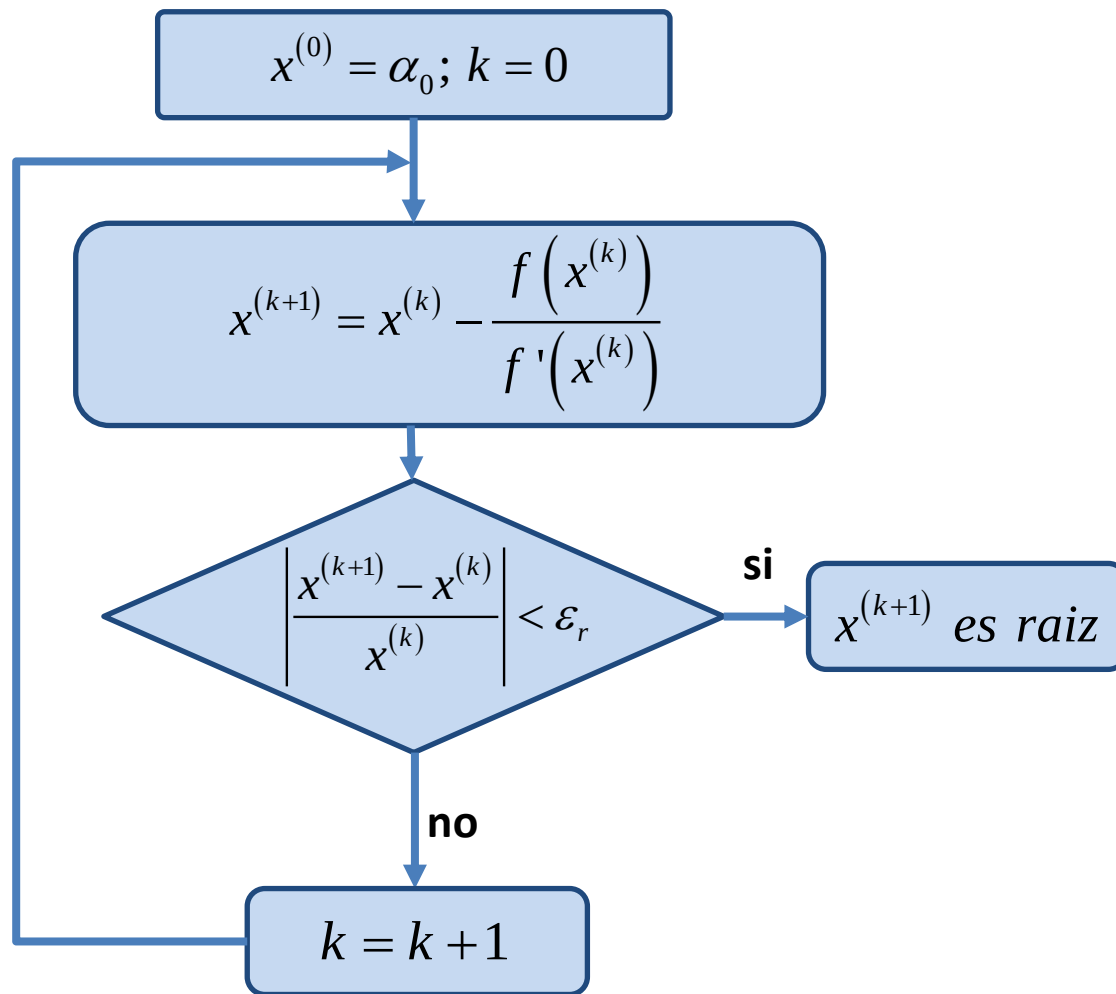
$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

$$\left| \frac{x^{(k+1)} - x^{(k)}}{x^{(k)}} \right| < \varepsilon_r$$

$$\left| f(x^{(k+1)}) \right| < \varepsilon$$

Se generan valores hasta satisfacer la tolerancia del error o alcanzar el número máximo de iteraciones





i	x_i	$f(x_i)$	$f'(x_i)$	<i>Error</i>
0				
1				
2				
3				
4				
5				
6				

```
function y=f(x)
```

```
y=sqrt(x) + log(x) - 4;
```

```
endfunction
```

Función

```
function y=fp(x)
```

```
y=0.5./sqrt(x) + 1./(x);
```

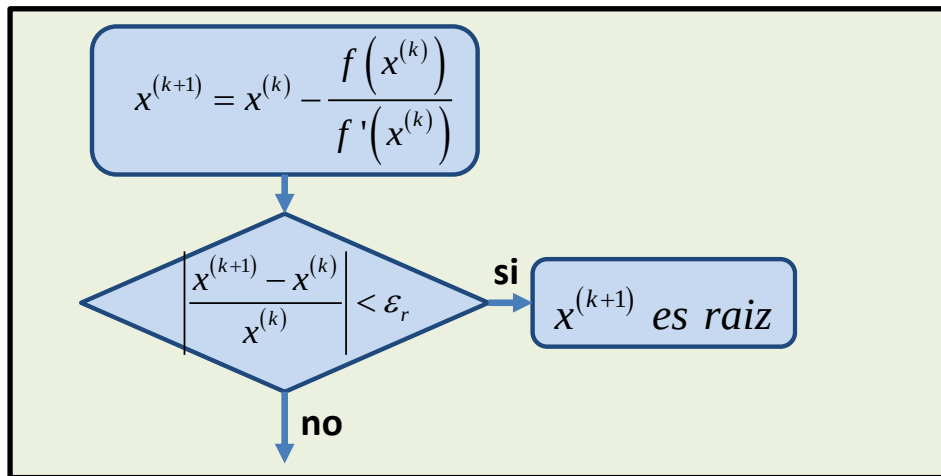
```
endfunction
```

Derivada

```
function out=newtonR(fun, funp, x0, tol)
```

```
x=x0;
```

```
for k=1:100
```



```
end
```

```
if k == 100
```

```
    out=[];
```

```
    disp('no converge');
```

```
end
```

```
endfunction
```

$$u(x) = \frac{f(x)}{f'(x)} \quad \text{Ralston and Rabinowitz (1978) propusieron la función } u(x) \text{ que tiene las mismas raíces que } f(x)$$

Por lo tanto, al aplicar N-R a la nueva función $u(x)$ se obtiene:
$$x^{(k+1)} = x^{(k)} - \frac{u(x^{(k)})}{u'(x^{(k)})}$$

luego,

$$\frac{u(x)}{u'(x)} = \frac{f(x) f'(x)}{[f'(x)]^2 - f(x) f''(x)}$$

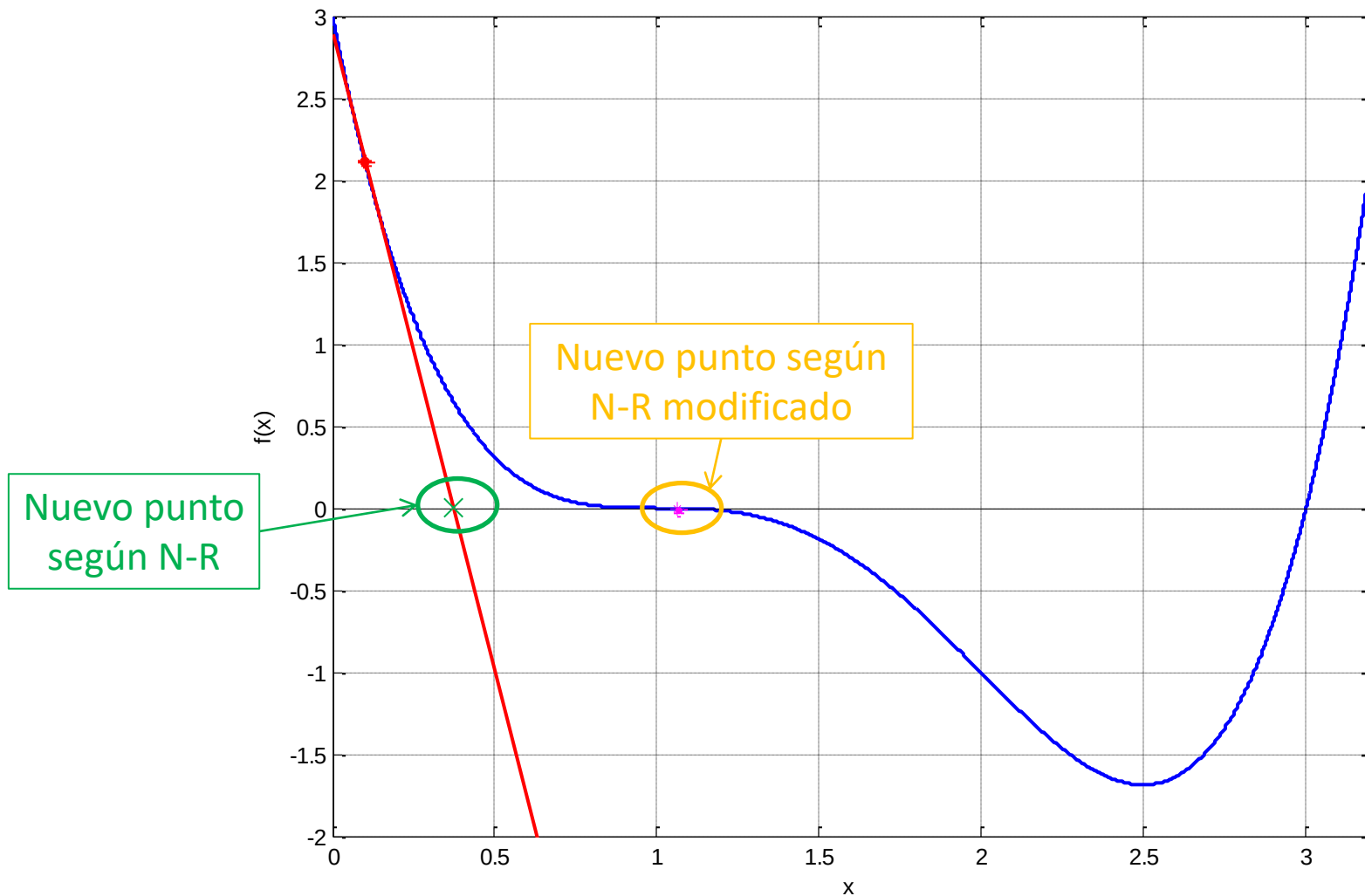
$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)}) f'(x^{(k)})}{[f'(x^{(k)})]^2 - f(x^{(k)}) f''(x^{(k)})}$$

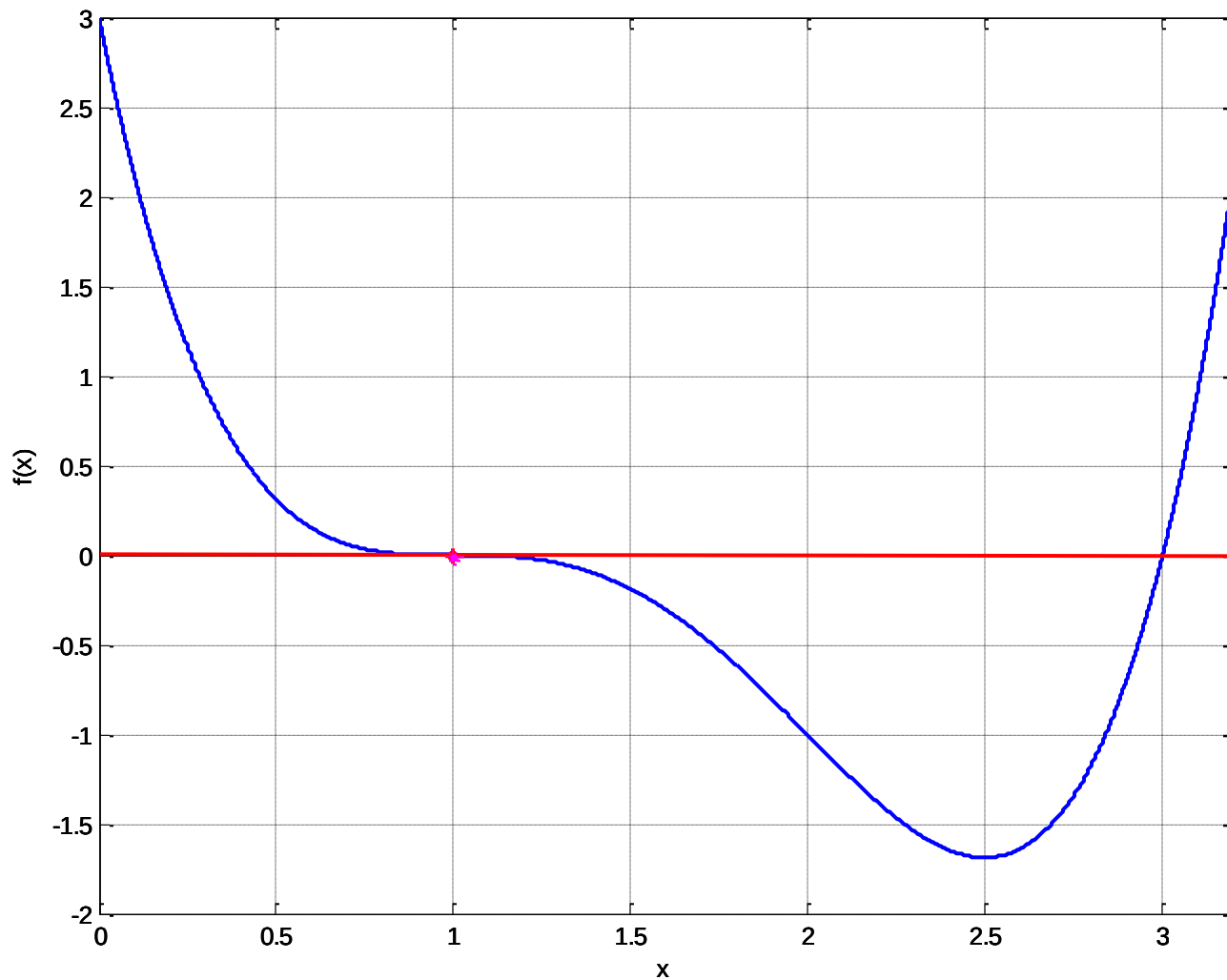
$k = 1, 2, \dots, k_{\max}$

$$\left| \frac{x^{(k+1)} - x^{(k)}}{x^{(k)}} \right| < \varepsilon_r$$

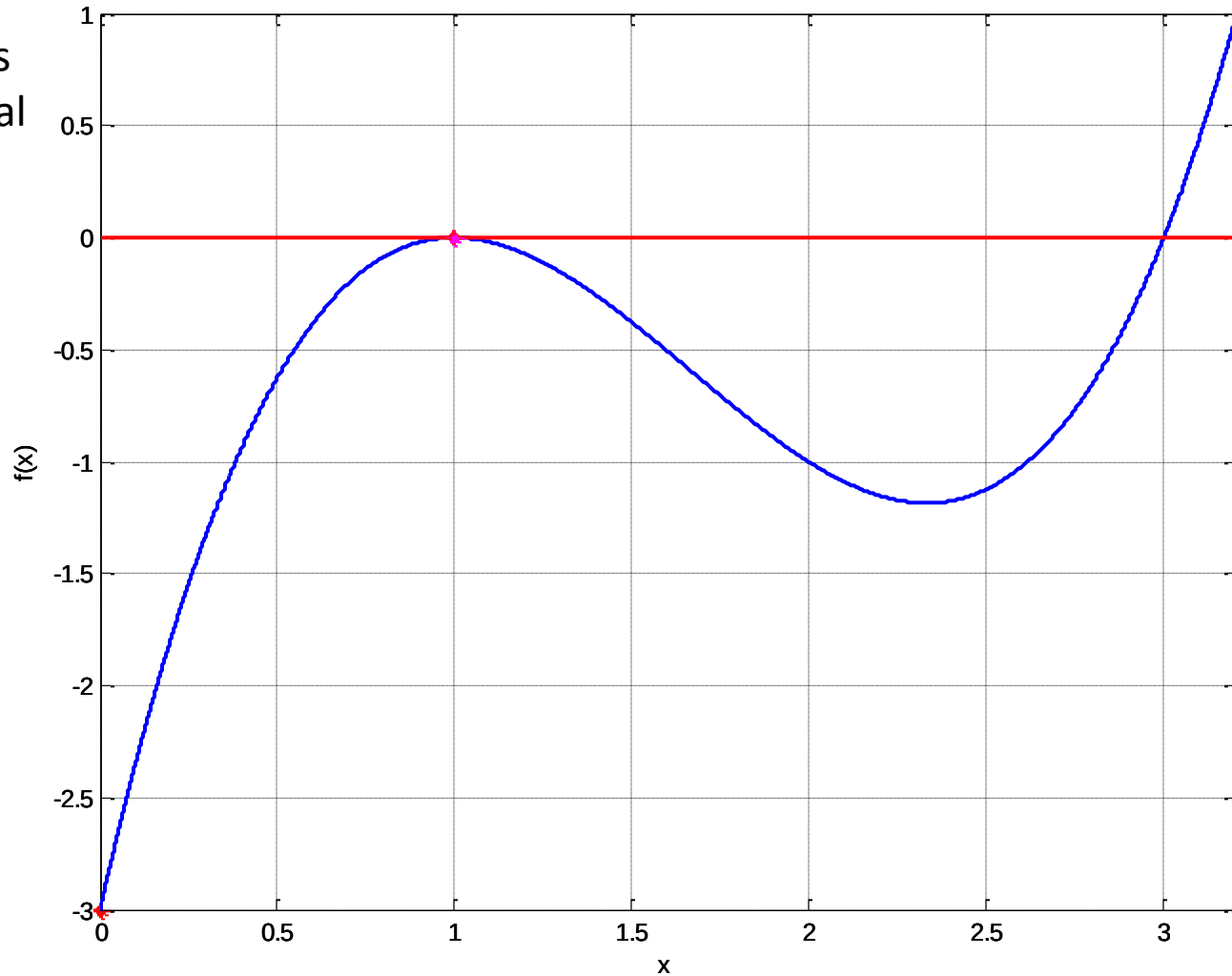
Se generan valores hasta satisfacer la tolerancia del error o alcanzar el número máximo de iteraciones

$$\left| f(x^{(k+1)}) \right| < \varepsilon$$

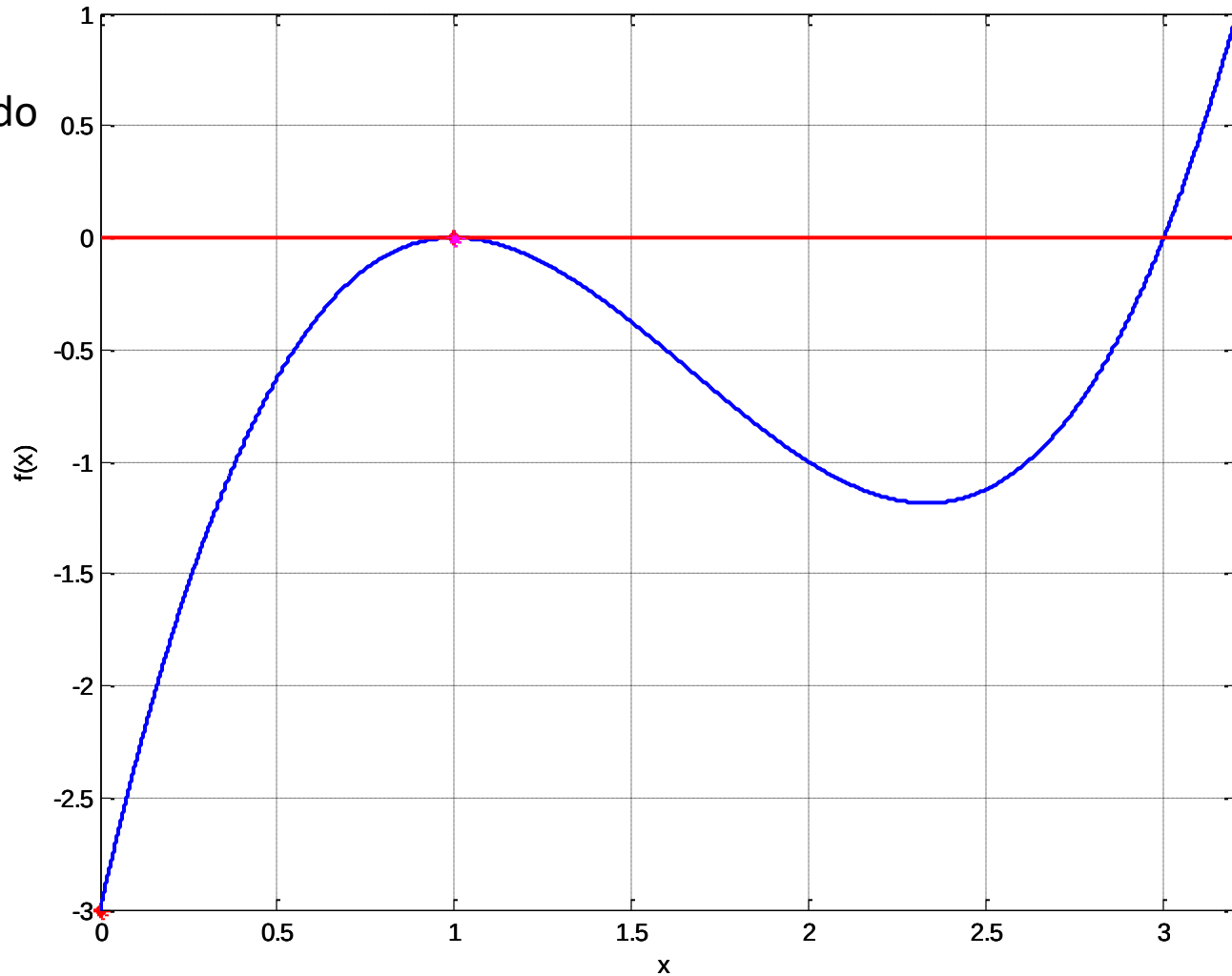


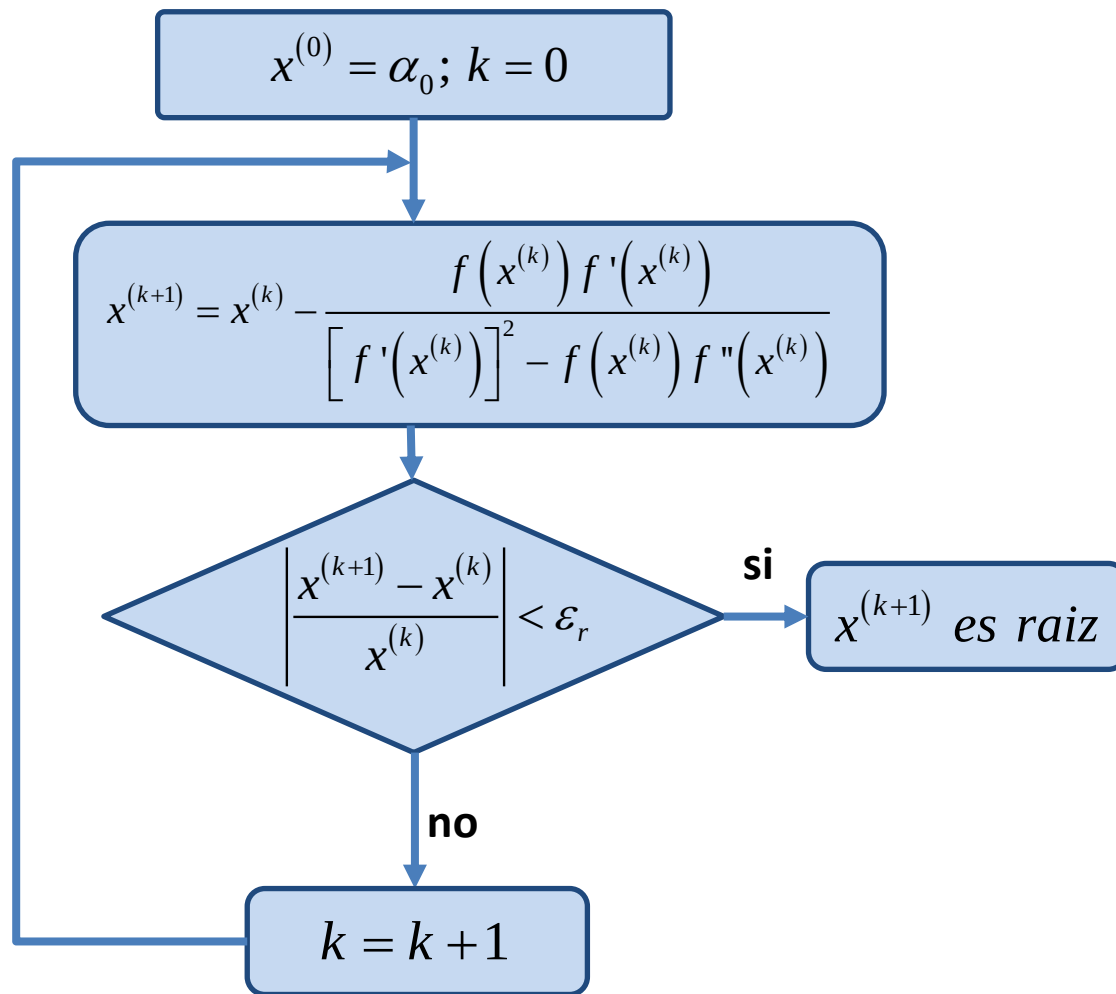


24 iteraciones
N-R tradicional



5 iteraciones
N-R modificado





i	x_i	$f(x_i)$	$f'(x_i)$	$f''(x_i)$	<i>Error</i>
0					
1					
2					
3					
4					
5					
6					

Modificar el código de Newton-Rhapson
y adaptarlo a este método



- Cualquier polinomio cúbico (incluida las EoS) puede expresarse de la siguiente forma:

$$z^3 + az^2 + bz + c = 0$$

- La primera raíz se encuentra utilizando la modificación de N-R modificado. Se sugiere utilizar como valor semilla el factor de compresibilidad de los gases ideales $z^{(0)}=1$
- Se utiliza la primera raíz hallada para factorizar el polinomio:

$$\left(z^3 + az^2 + bz + c \right) \Big|_{\substack{z - \alpha_1 \\ Q(z)}}$$

- Debido a la factorización se cumple que:

$$z^3 + az^2 + bz + c = (z - \alpha_1)Q(z)$$

- Ambos miembros tienen las mismas raíces por lo que las dos restante corresponden a las del polinomio $Q(z)$
- $Q(z)$ es una cuadrática!

- Factorización del polinomio original: $\left(z^3 + az^2 + bz + c\right) \Big|_{\substack{z - \alpha_1 \\ Q(z)}}$

	1	a	b	c
α_1		α_1	$\alpha_1(\alpha_1 + a)$	$\alpha_1(b + \alpha_1(\alpha_1 + a))$
	1	$\alpha_1 + a$	$b + \alpha_1(\alpha_1 + a)$	$c + \alpha_1(b + \alpha_1(\alpha_1 + a))$

0 (cero)

$$Q(z) = z^2 + (\alpha_1 + a)z + (b + \alpha_1(\alpha_1 + a))$$

$$D = (\alpha_1 + a)^2 - 4(b + \alpha_1(\alpha_1 + a)) \left\{ \begin{array}{l} D \geq 0 \text{ Las raíces restantes son reales} \\ D < 0 \text{ Las raíces restantes son imaginarias } (D < 0) \end{array} \right.$$

$$z^3 + az^2 + bz + c = 0$$

1. La primera raíz se encuentra utilizando la modificación de N-R modificado. Se sugiere utilizar como valor semilla $z^{(0)}=1$

$$z^{(k+1)} = z^{(k)} - \frac{f(z^{(k)}) f'(z^{(k)})}{\left[f'(z^{(k)}) \right]^2 - f(z^{(k)}) f''(z^{(k)})}$$

2. Se analiza el discriminante de la nueva ecuación cuadrática

$$D = (\alpha_1 + a)^2 - 4(b + \alpha_1(\alpha_1 + a))$$

$$D \geq 0 \left\{ \begin{array}{l} \alpha_2 = \frac{-(\alpha_1 + a) + \sqrt{D}}{2} \\ \alpha_3 = \frac{-(\alpha_1 + a) - \sqrt{D}}{2} \end{array} \right. \Rightarrow \begin{array}{l} z_v = \max(\alpha_1, \alpha_2, \alpha_3) \\ z_l = \min(\alpha_1, \alpha_2, \alpha_3) \end{array}$$

$D < 0$ Las restantes raíces no son de interés físico

Crear la function siguiendo los pasos de
la pagina anterior



Se desea calcular la densidad de la fase gaseosa del propano a 300 K y 9.98316 bar utilizando Peng Robinson.

$$T_c = 369.89801 \text{ K}; P_c = 42.56660 \text{ bar}; \omega = 0.15240; R = 83.14472 \frac{\text{cm}^3 \text{ bar}}{\text{mol K}}$$

$$a = \frac{0.45724 R^2 T_c^2}{P_c}$$

$$T_r = \frac{T}{T_c}$$

$$b = \frac{0.07780 R T_c}{P_c}$$

$$\alpha(T_r) = \left[1 + \left(0.37464 + 1.54226\omega - 0.2699\omega^2 \right) (1 - T_r^{1/2}) \right]^2$$

$$B' = \frac{bP}{RT}$$

$$\Theta' = \frac{a\alpha(T_r)P}{(RT)^2}$$

$$z^3 + (B' - 1)z^2 + (\Theta' - 3B'^2 - 2B')z + (B'^3 + B'^2 - \Theta'B') = 0$$

$$v_v = z_v RT / P$$