Tema 2 - Operaciones matemáticas básicas

Cálculo de raíces

M. en C. Gustavo Contreras Mayén

9 de marzo de 2017

Contenido

Contenido

Contenido

Método de Bisección

Después de que se ha identificado una raíz $f(x) = 0$ en el intervalo (x_1, x_2), disponemos de varios métodos para encontrar el valor de la raíz.

El método de bisección logra esta tarea: el intervalo se reduce sucesivamente a la mitad hasta que se vuelve suficientemente pequeño. La técnica de bisección no es el método más rápido disponible, pero es el más fiable. Una vez que una raíz se ha encontrado en un intervalo, nos podemos acercar a ella.

El método de bisección utiliza el mismo principio que el incremento sucesivo: si hay una raíz en el intervalo (x_1, x_2), entonces $f(x_1) * f(x_2) < 0$.

Con el fin de reducir a la mitad el intervalo, se calcula $f(x_3)$, donde $x_3 = (x1 + x2)/2$ es el punto medio del intervalo.

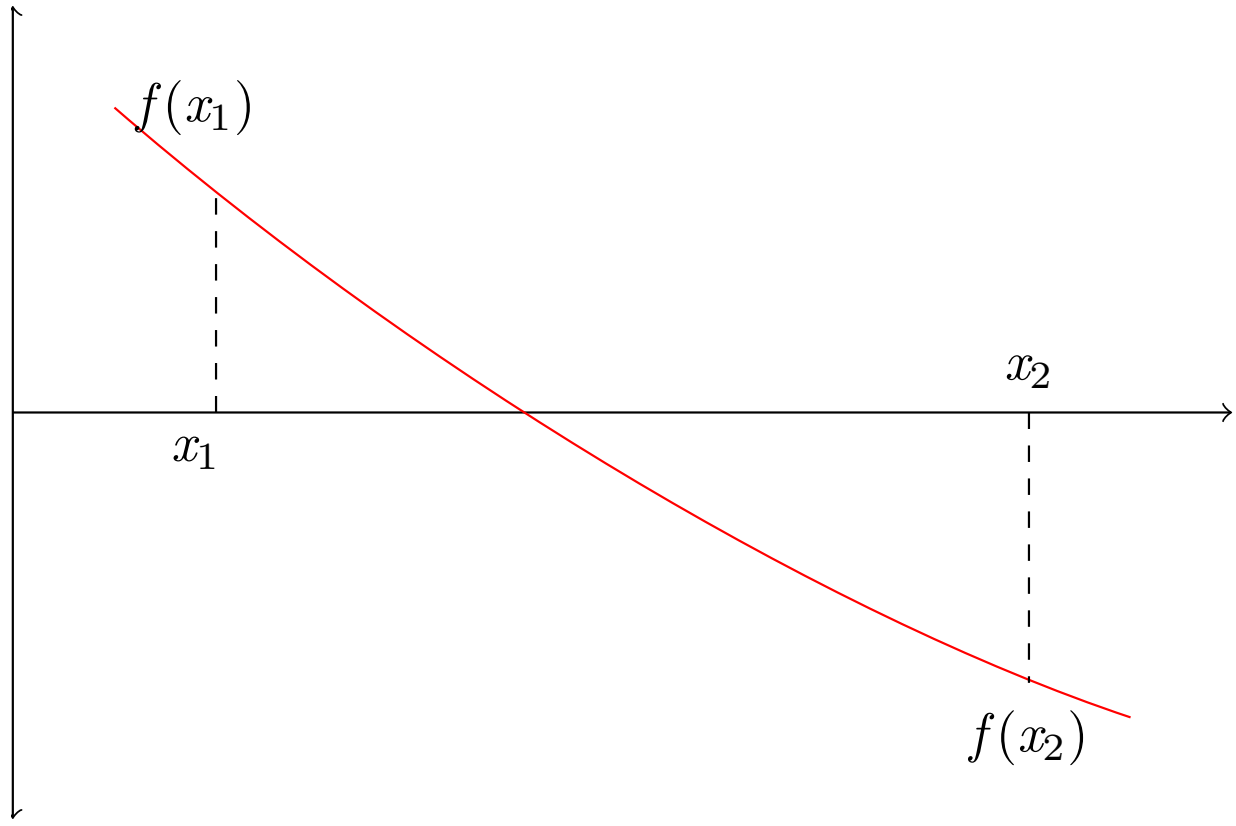


Figura (1): Se evalúa la función en los puntos x_1 y x_2

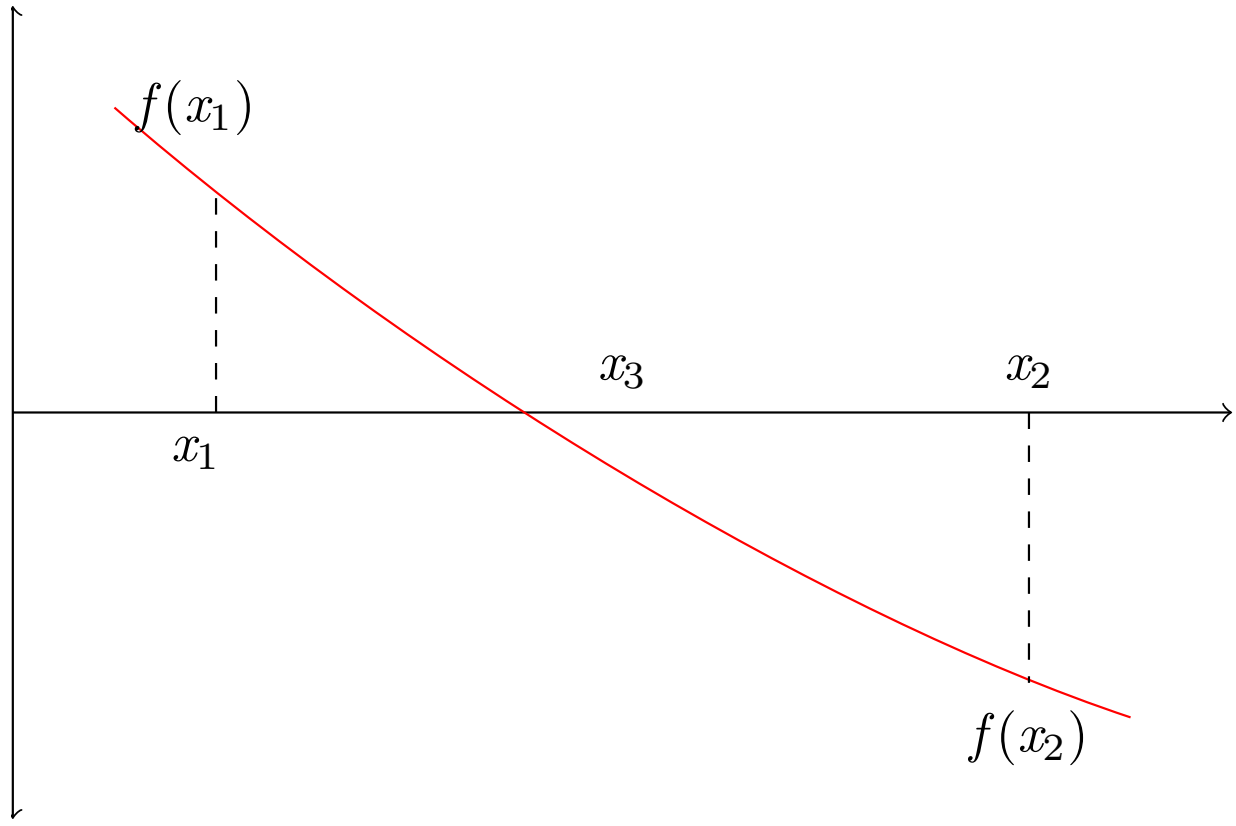


Figura (1): Se evalúa la función en los puntos x_1 y x_2

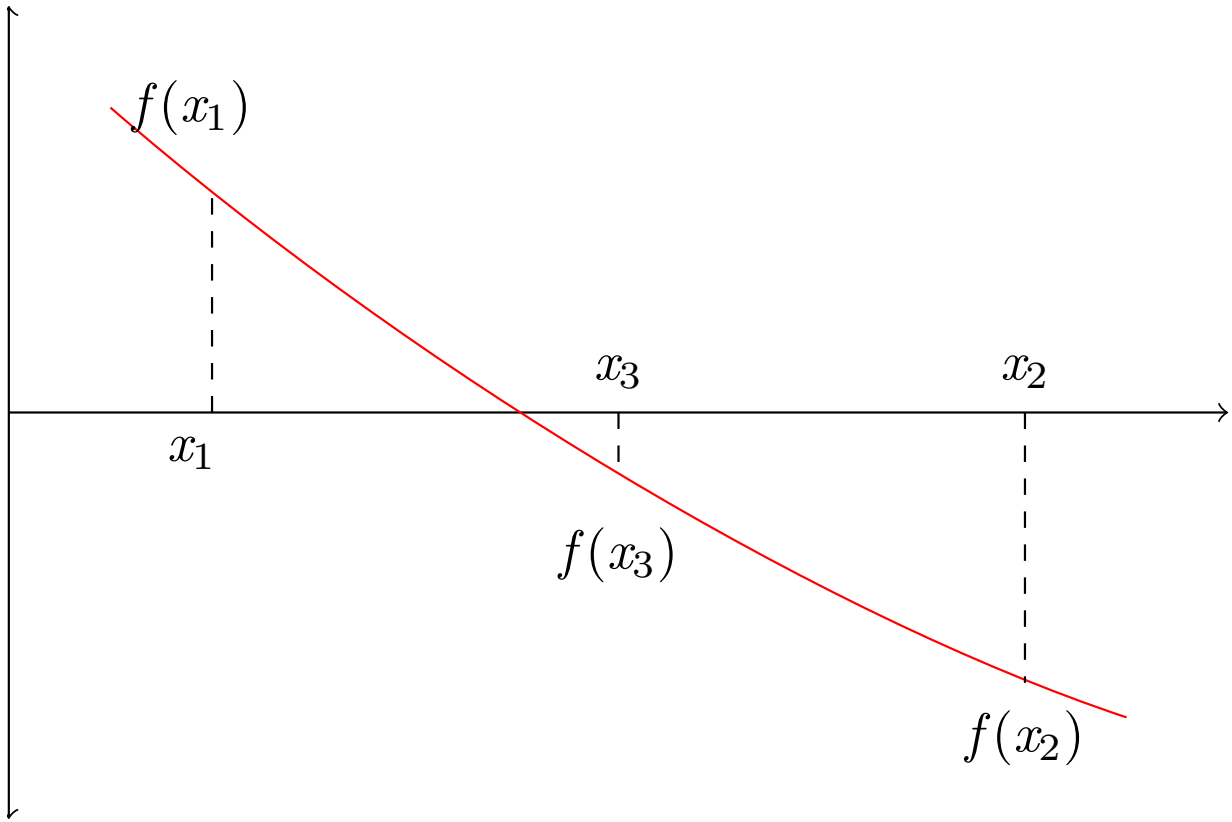


Figura (1): Se evalúa la función en los puntos x_1 y x_2

Si $f(x_1) * f(x_3) < 0$, entonces la raíz debe estar en (x_1, x_3) entonces re-emplazamos del intervalo inicial x_2 por x_3. De lo contrario, la raíz se encuentra en (x_3, x_2), en tal caso, se sustituye x_3 por x_1.

Si $f(x_1) * f(x_3) < 0$, entonces la raíz debe estar en (x_1, x_3) entonces re-emplazamos del intervalo inicial x_2 por x_3. De lo contrario, la raíz se encuentra en (x_3, x_2), en tal caso, se sustituye x_3 por x_1.

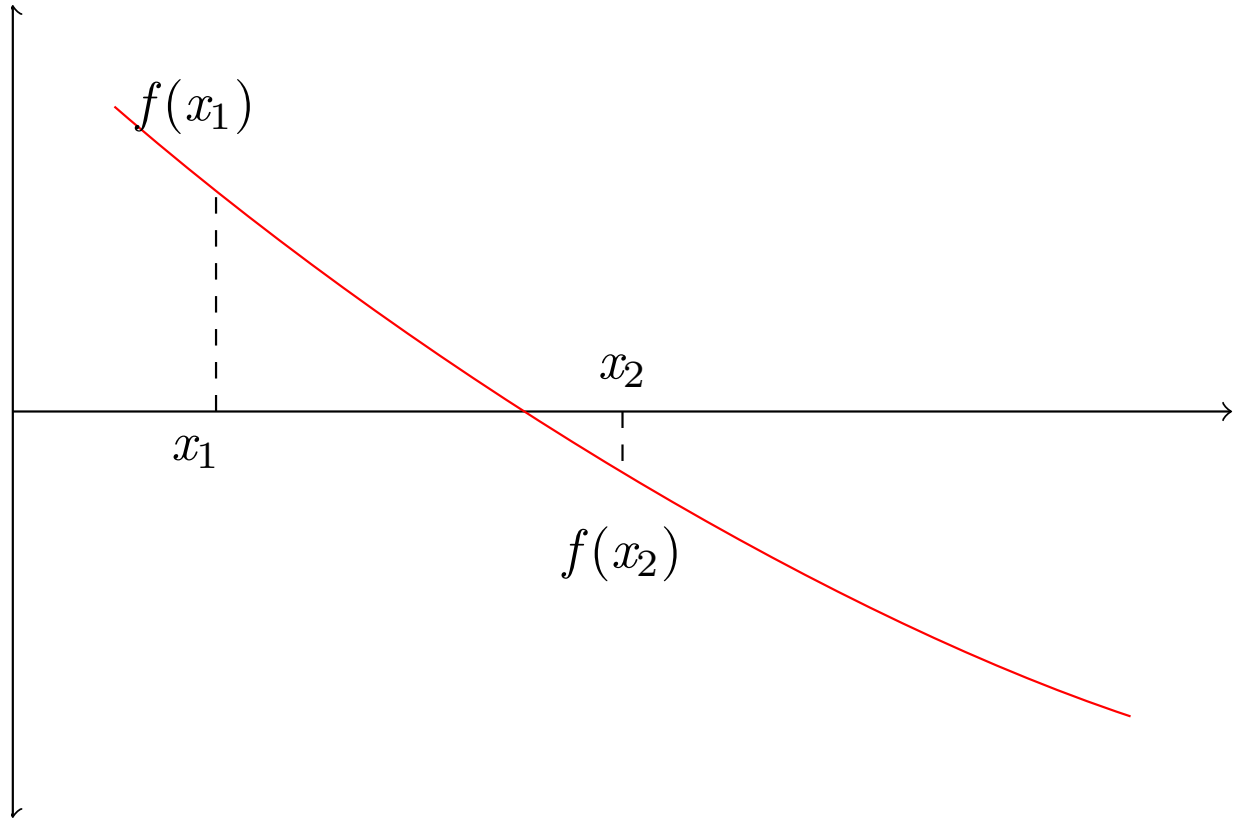


Figura (2): Una vez localizado el intervalo donde hay cambio de signo, se repite el proceso.

Si $f(x_1) * f(x_3) < 0$, entonces la raíz debe estar en (x_1, x_3) entonces re-emplazamos del intervalo inicial x_2 por x_3. De lo contrario, la raíz se encuentra en (x_3, x_2), en tal caso, se sustituye x_3 por x_1.

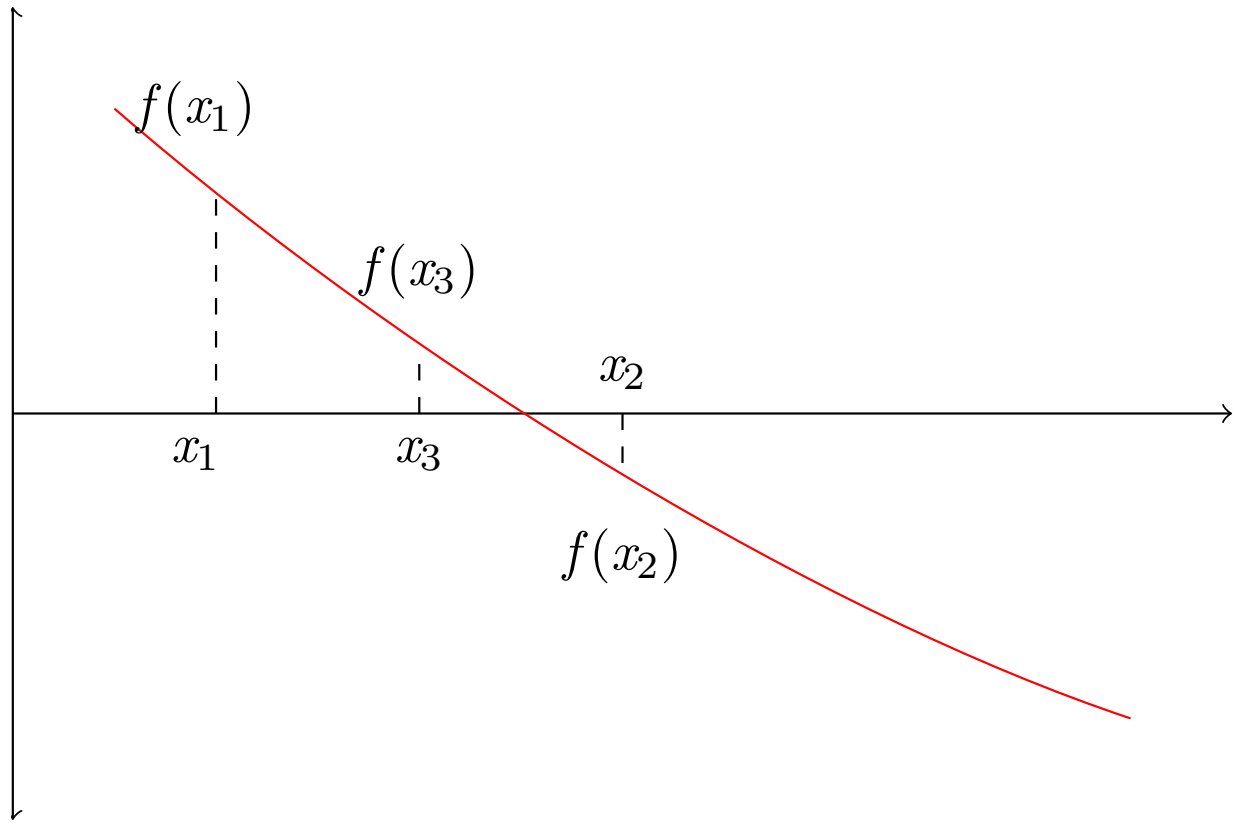


Figura (2): Una vez localizado el intervalo donde hay cambio de signo, se repite el proceso.

En cualquiera de los casos, el nuevo intervalo (x_1, x_2) es la mitad del tamaño del intervalo original. La bisección es repite hasta que el intervalo se ha reducido a un valor ϵ pequeño, de modo que

$$|x_2 - x_1| \leq \epsilon$$

Es fácil calcular el número de biseccciones necesarias para alcanzar el valor de ϵ.
El intervalo inicial Δx, se reduce a $\Delta x/2$ en la

primera bisección, $\Delta x/2^2$ en la segunda, luego de n biseccciones, $\Delta x/2^n$.

Haciendo $\Delta x/2^n = \epsilon$, resolvemos para n

$$n = \frac{ln(|\Delta x|/\epsilon)}{ln2}$$

Algoritmo para el método de bisección

```
1 def bisect(f,x1,x2,switch,epsilon=1.0e-9):
2     f1 = f(x1)
3     if f1 == 0.0: return x1
4     f2 = f(x2)
5     if f2 == 0.0: return x2
6
7     if f1*f2 > 0.0: print 'La raiz no se ha
          identificado en un intervalo'
8
9     n = ceil(log(abs(x2 - x1)/epsilon)/log(2.0))
```

La función `ceil` devuelve el menor entero mayor o igual a x.

Ejemplos:

```
>>>ceil(1.1) # 2.0
>>>ceil(1.6) # 2.0
>>>ceil(-1.1) # -1.0
>>>ceil(-1.6) # -1.0
```

Algoritmo para el método de bisección

```
1     for i in np.arange(n):
2         x3 = 0.5*(x1 + x2); f3 = f(x3)
3         if (switch == 0) and (abs(f3) > abs(f1)) \
4                          and (abs(f3) > abs(f2)):
5             return None
6         if f3 == 0.0: return x3
7         if f2*f3 < 0.0:
8             x1 = x3; f1 = f3
9         else:
10            x2 =x3; f2 = f3
11    return (x1 + x2)/2.0
```

Haciendo que la variable `switch` $= 1$, se forza la rutina para comprobar si la magnitud de $f(x)$ disminuye con la reducción a la mitad de cada intervalo.

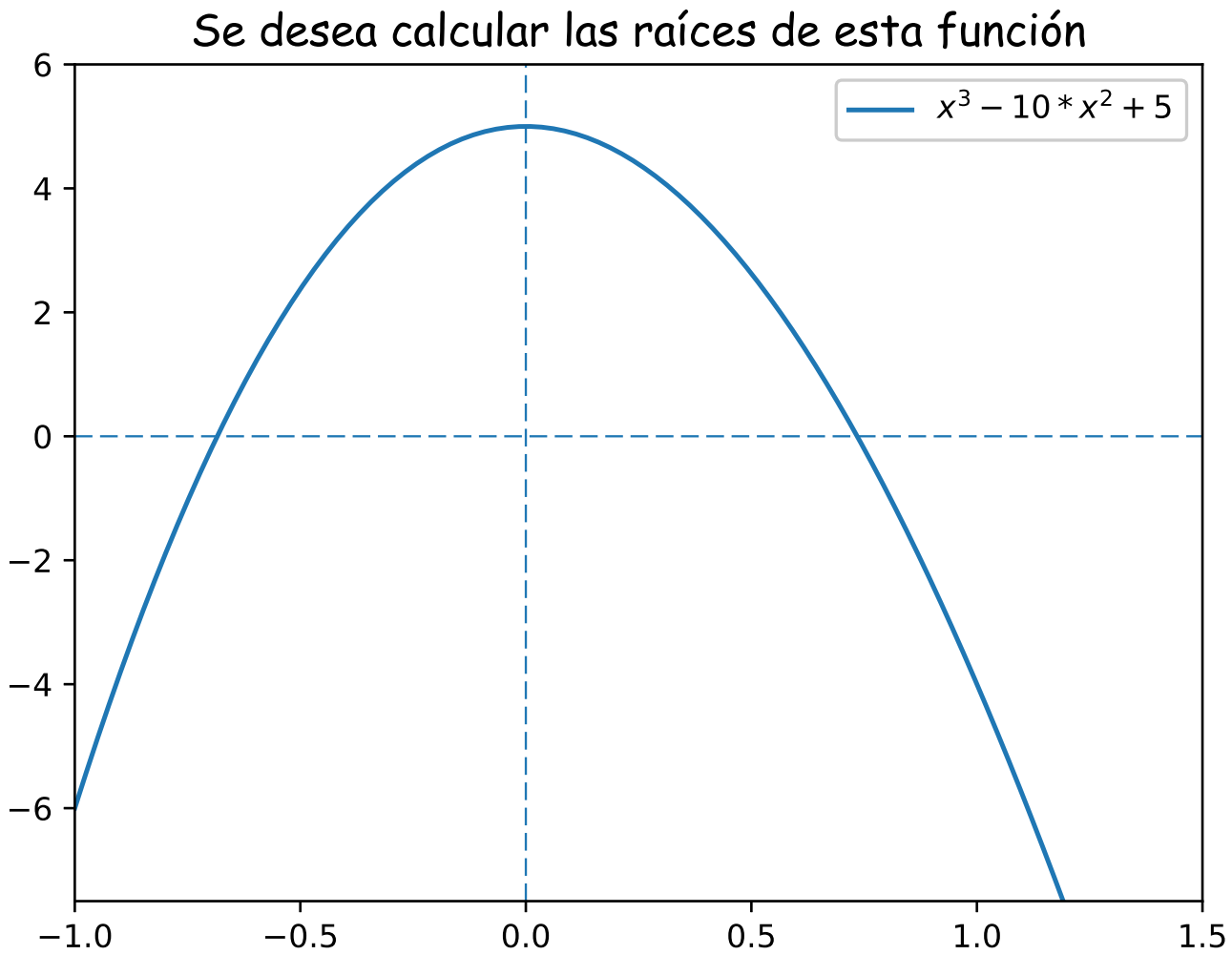
Si no es así, algo anda mal (probablemente la "raíz" no es una raíz en absoluto, sino una singularidad) y se devuelve la `raíz` $=$ `None`. Dado que esta característica no siempre es deseable, el valor predeterminado es `switch`$=0$.

Veamos lo que hace la variable `switch`:

Hasta el momento no nos hemos fijado en la magnitud de $f(x_1)$ y $f(x_2)$, sólo y en el signo del respectivo producto, pero ahora, tomamos en cuenta la magnitud tanto de los puntos evaluados en la función como del producto mismo.

Consideremos el primer caso de la función $f(x)$, donde ya sabemos que hay una raíz en el intervalo $[0.6, 0.8]$

Ejemplo utilizado en la técnica de incrementos sucesivos



El valor por defecto de `switch` = 0

x_1	x_2	$f(x_1)$	$f(x_2)$	$f(x_1)f(x_2)$
0.0	0.2	5.0	4.608	23.04
0.2	0.4	4.608	3.464	15.9621
0.4	0.6	3.464	1.616	5.5978
0.6	0.8	1.616	-0.888	-1.4350
0.8	1.0	-0.888	-4.0	3.552
1.0	1.2	-4.0	-7.672	30.688

Del código tenemos que:

```
if (switch == 0) and (abs(f3) > abs(f1)) \
                 and (abs(f3) > abs(f2)):
    return None
```

$abs(f3) > abs(f1)$	$abs(f3) > abs(f2)$	Expresión
True	True	True

Aquí tendrán que completar la tabla de tal manera que revisen el resultado de la expresión compuesta.

Ejercicios

Implementa el método de bisección para calcular el(los) intervalo(s) y la(s) raíz(ces) de:

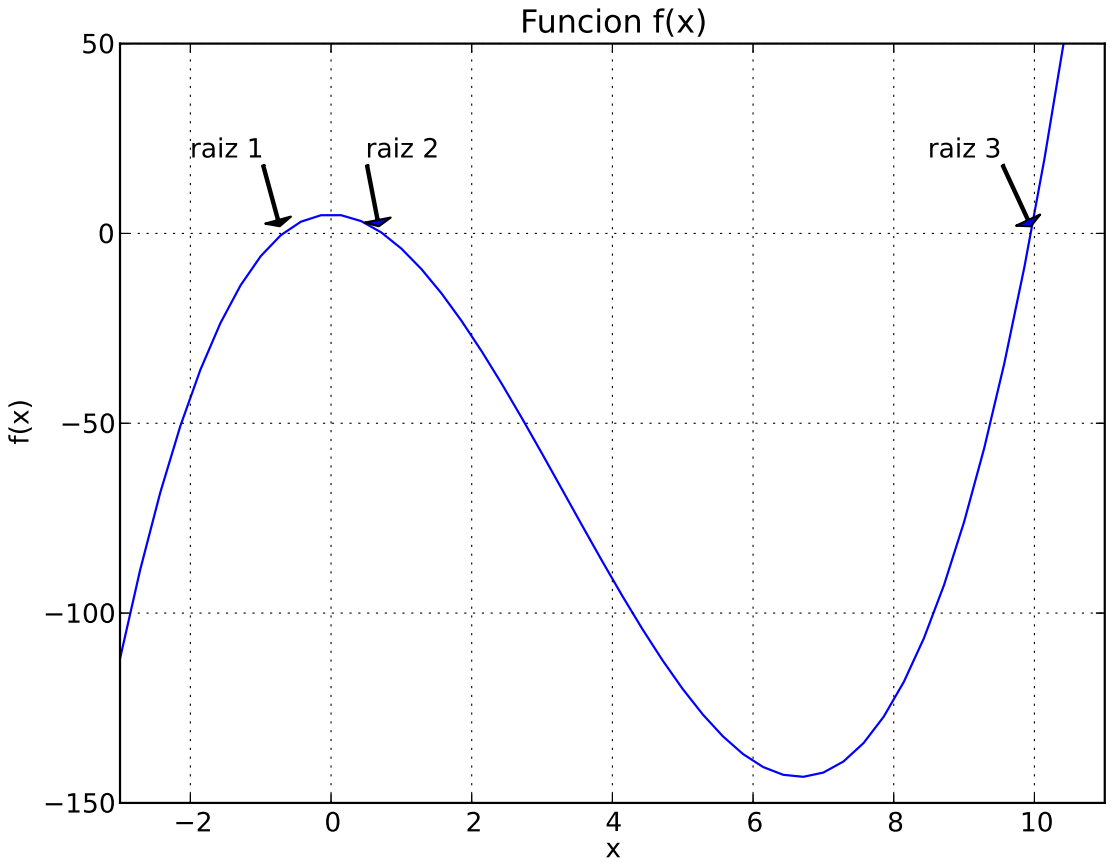
$$f(x) = x^3 - 10x^2 + 5$$
$$g(x) = x - \tan(x) \qquad 0 \leq x \leq 20$$

Con una tolerancia de 1×10^{-9}

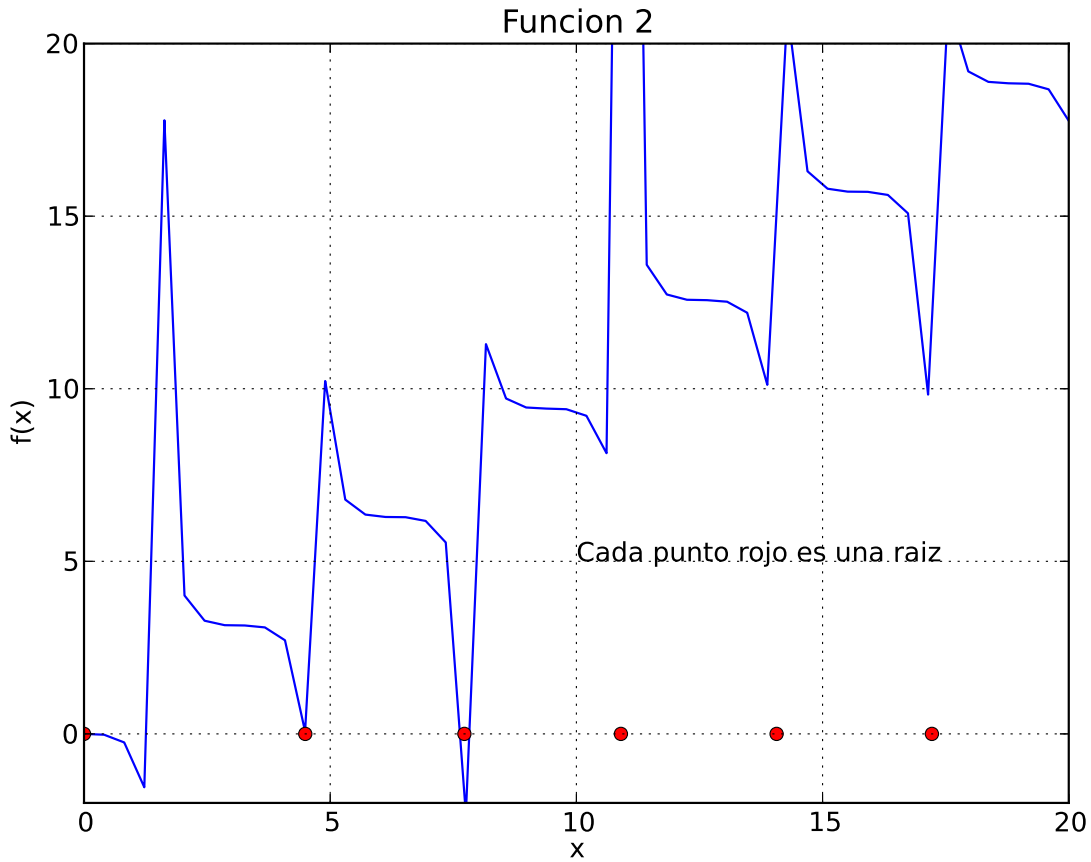
Toma en cuenta que $\tan(x)$ es singular y cambia de signo en $x = \pi/2, 3\pi/2, \ldots$. Para no confundir estos puntos para las raíces, hacemos `switch`$= 1$.

La proximidad de las raíces a las singularidades es otro problema potencial que puede ser prevenido mediante el uso de Δx pequeña, usa $x = 0.01$.

Gráficas de las funciones y sus raíces



Gráficas de las funciones y sus raíces



Estrategia de solución

Para integrar los elementos que hemos visto:

1. Recuperamos el algoritmo que nos identifica en qué intervalo se encuentra una raíz.

Estrategia de solución

Para integrar los elementos que hemos visto:

1. Recuperamos el algoritmo que nos identifica en qué intervalo se encuentra una raíz.
2. Aplicamos el algoritmo del método de bisección.

Estrategia de solución

Para integrar los elementos que hemos visto:

1. Recuperamos el algoritmo que nos identifica en qué intervalo se encuentra una raíz.
2. Aplicamos el algoritmo del método de bisección.
3. Usamos un ciclo que nos revise en el dominio que se nos proporciona, si existe más de una raíz.

```
1  def f(x): return x**3-10*x**2+5
2
3  a,b,dx = (-2.0,11.0,0.02)
4
5  print 'Intervalo (x1,x2) raiz'
6  while 1:
7      try:
8          x1, x2 = buscaraiz(f,a,b,dx)
9      except Exception, e:
10         print e; break
11     if x1 != None:
12         a = x2
13         root = bisect(f,x1,x2,0)
14         if raiz != None: print ('(%2.4f, %2.4f)
               %2.8f' %(x1, x2, raiz))
```

```
1       else:
2           print '\nHecho'
3           break
```

Método de Newton-Raphson

El método de Newton-Raphson es el algoritmo más conocido para encontrar raíces por una buena razón: es simple y rápido.

El único detalle es que utiliza la derivada $f'(x)$ así como la función $f(x)$. Por tanto, en los problemas a resolver con este algoritmo, deberá de contemplarse que la derivada sea fácil de calcularse.

El método de N-R se obtiene de la expansión en series de Taylor de $f(x)$ alrededor de x:

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + O(x_{i+1} - x_i)^2$$

Si x_{i+1} es una raíz de $f(x) = 0$, tenemos que:

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i) + O(x_{i+1} - x_i)^2$$

Suponiendo que x_i está cerca de x_{i+1}, podemos eliminar el último término de la ecuación y resolver para x_{i+1}, por lo que la fórmula de Newton-Raphson es:

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

Si x representa el valor verdadero de la raíz, el error en x_i es $E_i = x - x_i$. Se puede demostrar que si x_{i+1} se calcula de la expresión de N-R, el error es:

$$E_{i+1} = -\frac{f''(x_i)}{2f'(x_i)} E_i^2$$

Lo que nos dice que el método de N-R converge de manera cuadrática, es decir, el error es el cuadrado del error del punto previo.

Representación gráfica

Podemos interpretar que x_{i+1} es el punto en donde la tangente de $f(x_i)$ cruza el eje de las x:

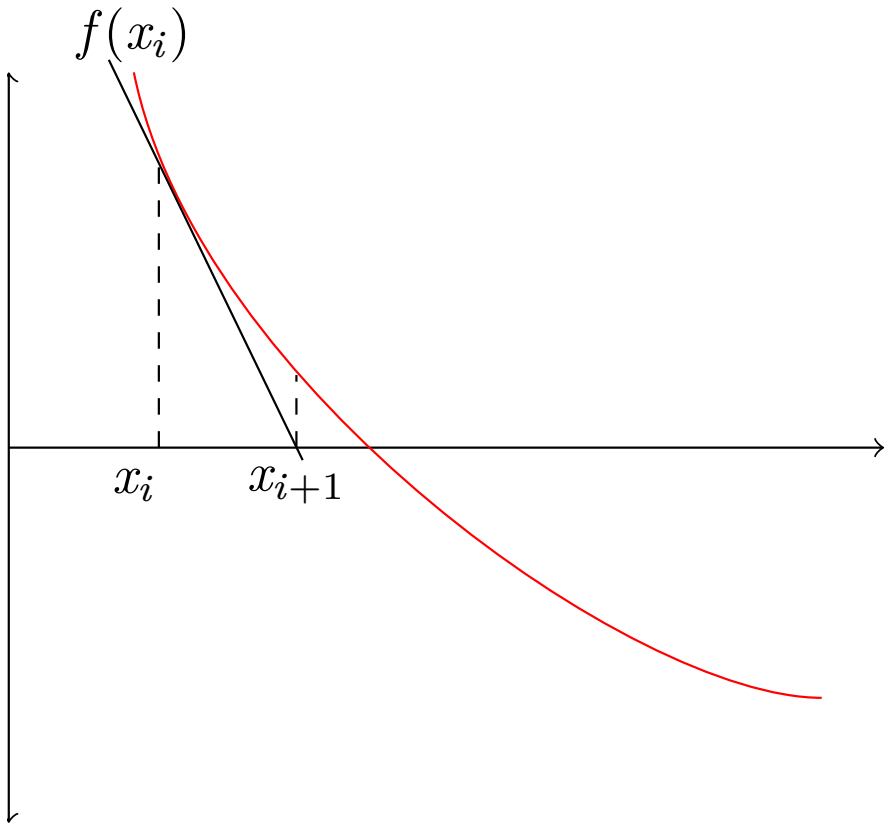


Figura (3): Proyección de la tangente de $f(x_i)$ hacia el eje de las abscisas.

El método de N-R es sencillo: se aplica la expresión para x_{i+1} iniciando con un valor x_0, hasta alcanzar un criterio de convergencia:

$$\left|x_{i+1} - x_i\right| < \epsilon$$

El algoritmo es el siguiente:

1. Sea x un valor inicial para la raíz de $f(x) = 0$.

El método de N-R es sencillo: se aplica la expresión para x_{i+1} iniciando con un valor x_0, hasta alcanzar un criterio de convergencia:

$$\left|x_{i+1} - x_i\right| < \epsilon$$

El algoritmo es el siguiente:

1. Sea x un valor inicial para la raíz de $f(x) = 0$.
2. Calcular $\Delta x = -f(x)/f'(x)$.

El método de N-R es sencillo: se aplica la expresión para x_{i+1} iniciando con un valor x_0, hasta alcanzar un criterio de convergencia:

$$|x_{i+1} - x_i| < \epsilon$$

El algoritmo es el siguiente:

1. Sea x un valor inicial para la raíz de $f(x) = 0$.
2. Calcular $\Delta x = -f(x)/f'(x)$.
3. Asignar $x \leftarrow x + \Delta x$ y se repiten los pasos 2-3, hasta alcanzar $|\Delta x| < \epsilon$.

Aunque el método de Newton-Raphson converge rápidamente cerca de la raíz, sus características globales de convergencia son pobres.

La razón es que la línea tangente no es siempre una aproximación aceptable de la función.

Caso particular 1.

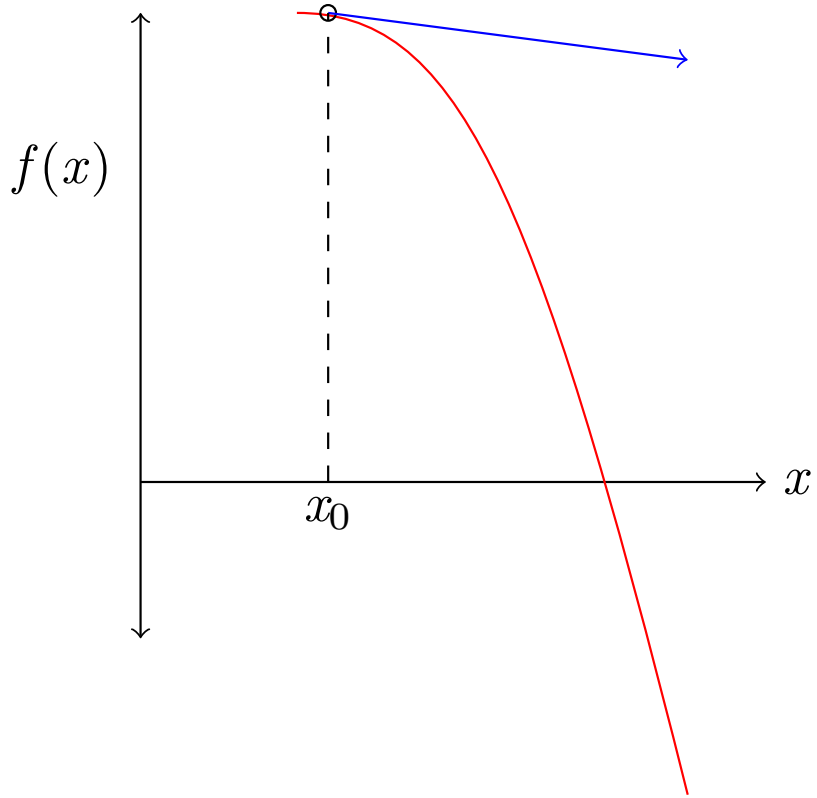


Figura (4): La proyección de la derivada puede no cruzar el eje de las abscisas.

Caso particular 2.

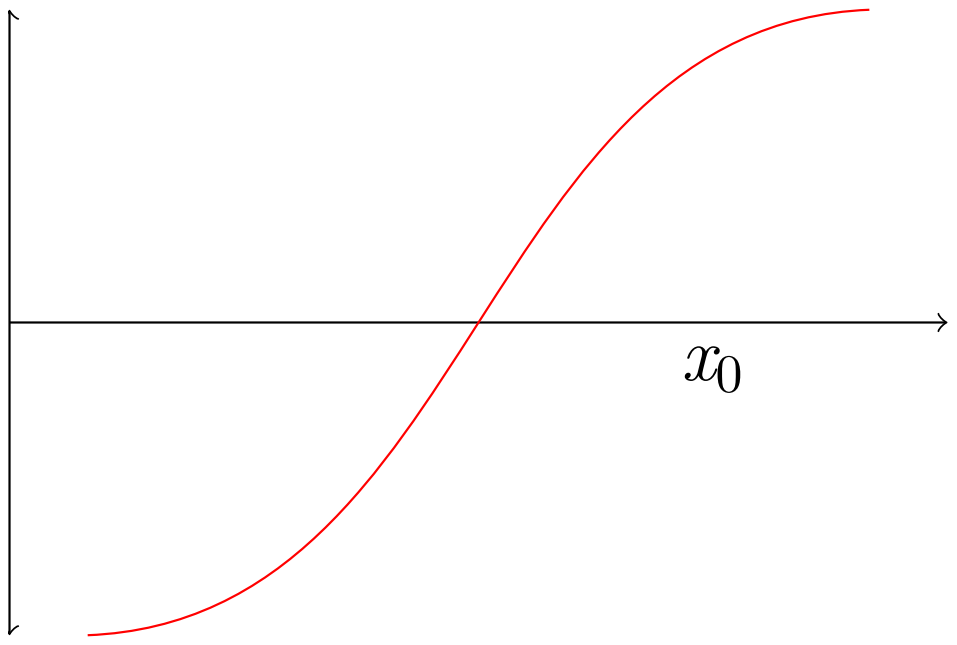


Figura (5): Alternancia en las proyecciones de la recta tangente que no conducen a ningún resultado.

Caso particular 2.

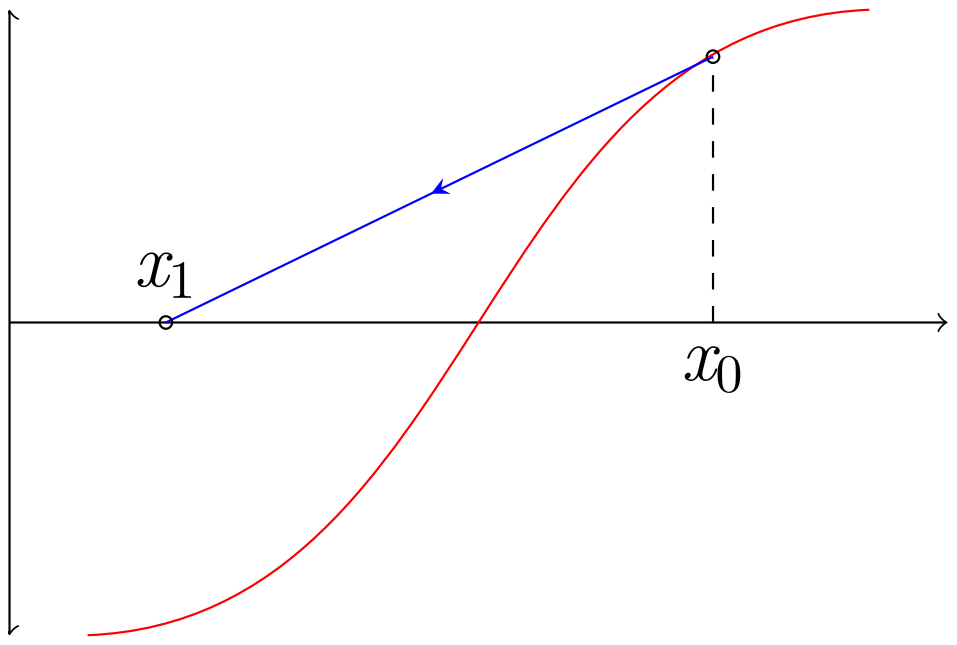


Figura (5): Alternancia en las proyecciones de la recta tangente que no conducen a ningún resultado.

Caso particular 2.

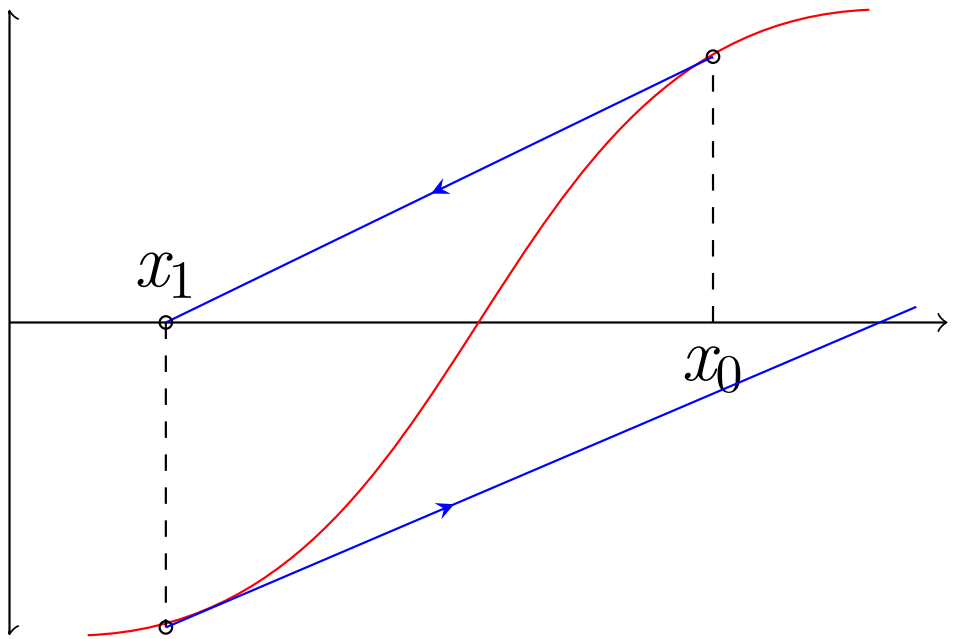


Figura (5): Alternancia en las proyecciones de la recta tangente que no conducen a ningún resultado.

Algoritmo para el método Newton-Raphson

La siguiente algoritmo para el método de Newton-Raphson supone que la raíz a calcularse inicialmente está en el intervalo (a, b).

El punto medio del intervalo se utiliza como aproximación inicial de la raíz. Los extremos del intervalo se actualizan luego de cada iteración. Si una iteración del método Newton-Raphson no se mantiene dentro del intervalo, se descarta y se reemplaza con el método de bisección.

Ya que el método `newtonRaphson` utiliza la función $f(x)$, así como su derivada, (denotadas por f y df) deben ser proporcionadas por el usuario.

Algoritmo de Newton-Raphson

```
1 def newtonRaphson(f,df,a,b,tol=1.0e-9):
2     fa = (f(a)
3     if fa == 0.0: return a
4     fb = f(b)
5     if f(b) == 0.0: return b
6     if fa*fb > 0.0: print ('La raiz no esta en el
          intervalo')
7     x = 0.5 * (a + b)
```

```
1        for i in range(30):
2            fx = f(x)
3            if abs(fx) < tol: return x
4
5            if fa*fx < 0.0:
6                b = x
7            else:
8                a = x; fa = fx
```

```
1         dfx = df(x)
2
3         try: dx = -fx/dfx
4         except ZeroDivisionError: dx = b - a
5         x = x + dx
6
7         if(b - x)*(x - a) < 0.0:
8             dx = 0.5*(b-a)
9             x = a + dx
10
11        if abs(dx) < tol*max(abs(b),1.0): return x
12
13    print ('Son demasiadas iteraciones')
```

Ejercicio

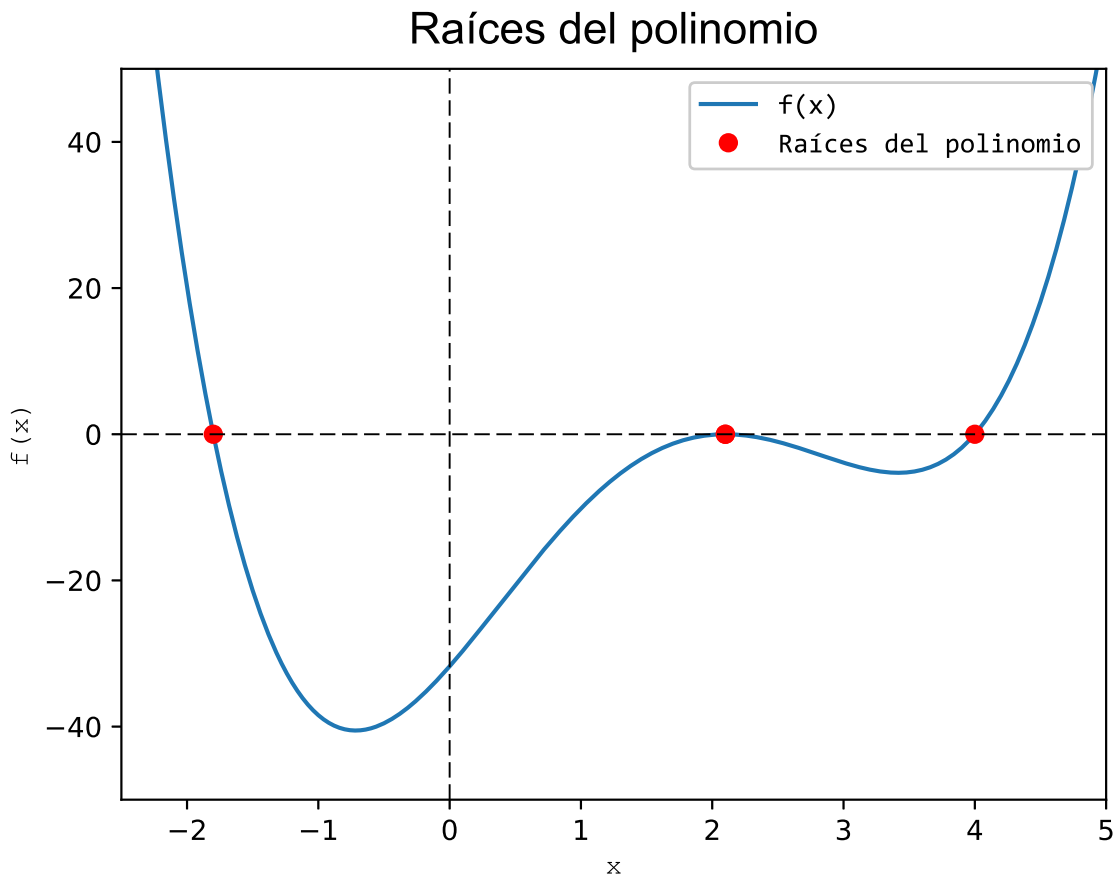
Encontrar la raíz positiva más pequeña de

$$f(x) = x^4 - 6.4x^3 + 6.45x^2 + 20.538x - 31.752$$

Ejercicio

Encontrar la raíz positiva más pequeña de

$$f(x) = x^4 - 6.4x^3 + 6.45x^2 + 20.538x - 31.752$$



```
1  def f(x): return x**4 - 6.4*x**3 + 6.45*x**2 +
       20.538*x - 31.752
2  def df(x): return 4.0*x**3 - 19.2*x**2 + 12.9*x +
        20.538
3
4  def newtonRaphson(x,tol=1e-05):
5      for i in range(30):
6          dx = -f(x)/df(x)
7          x = x + dx
8          if abs(dx) < tol: return x,i
9      print 'Son demasiadas iteraciones\n'
10
11 raiz,numIter = newtonRaphson(2.0)
12
13 print 'Raiz =',raiz
14 print 'Numero de iteraciones =',numIter
```

Ejercicios

1. Encuentra las raíces de $x \sin x + 3 \cos x - x = 0$ en el intervalo $(-6, 6)$
2. Calcula todas las raíces reales de $x^4 + 0.9x^3 - 2.3x^2 + 3.6x - 25.2 = 0.$
3. Calcula todas las raíces reales de $x^4 + 2x^3 - 7x^2 + 3 = 0.$
4. Calcula todas las raíces de $\sin x - 0.1x = 0.$

Método de la falsa posición

Este método es parecido al de bisección, ya que el intervalo que contiene a la raíz se va reduciendo.

En vez de bisectar de manera monótona el intervalo, se utiliza una interpolación lineal ajustada a dos puntos extremos para encontrar la aproximación de la raíz.

La función está bien aproximada por la interpolación lineal, con lo que las raíces tendrán una buena precisión; la iteración convergerá más rápido que como ocurre con el método de bisección.

Dado un intervalo $[a, c]$ que contenga a la raíz, la función lineal que pasa por $(a, f(a))$ y $(c, f(c))$ se escribe como:

$$y = f(a) + \frac{f(c) - f(a)}{c - a}(x - a)$$

de donde se despeja x:

$$x = a + \frac{c - a}{f(c) - f(a)}(y - f(a))$$

La coordenada x en donde la línea intersecta al eje x se determina al hacer $y = 0$ en la ecuación anterior, por tanto:

$$b = a - \frac{c - a}{f(c) - f(a)} f(a) = \frac{af(c) - cf(a)}{f(c) - f(a)}$$

Después de encontrar b, el intervalo $[a, c]$ se divide en $[a, b]$ y $[b, c]$.

Si $f(a)f(b) \leq 0$, la raíz se encuentra en $[a, b]$; en caso contrario, está en $[b, c]$. Los extremos del nuevo intervalo que contiene a la raíz se renombran para el siguiente paso como a y c.

El procedimiento de interpolación se repite hasta que las raíces estimadas convergen.

Método de la falsa posición

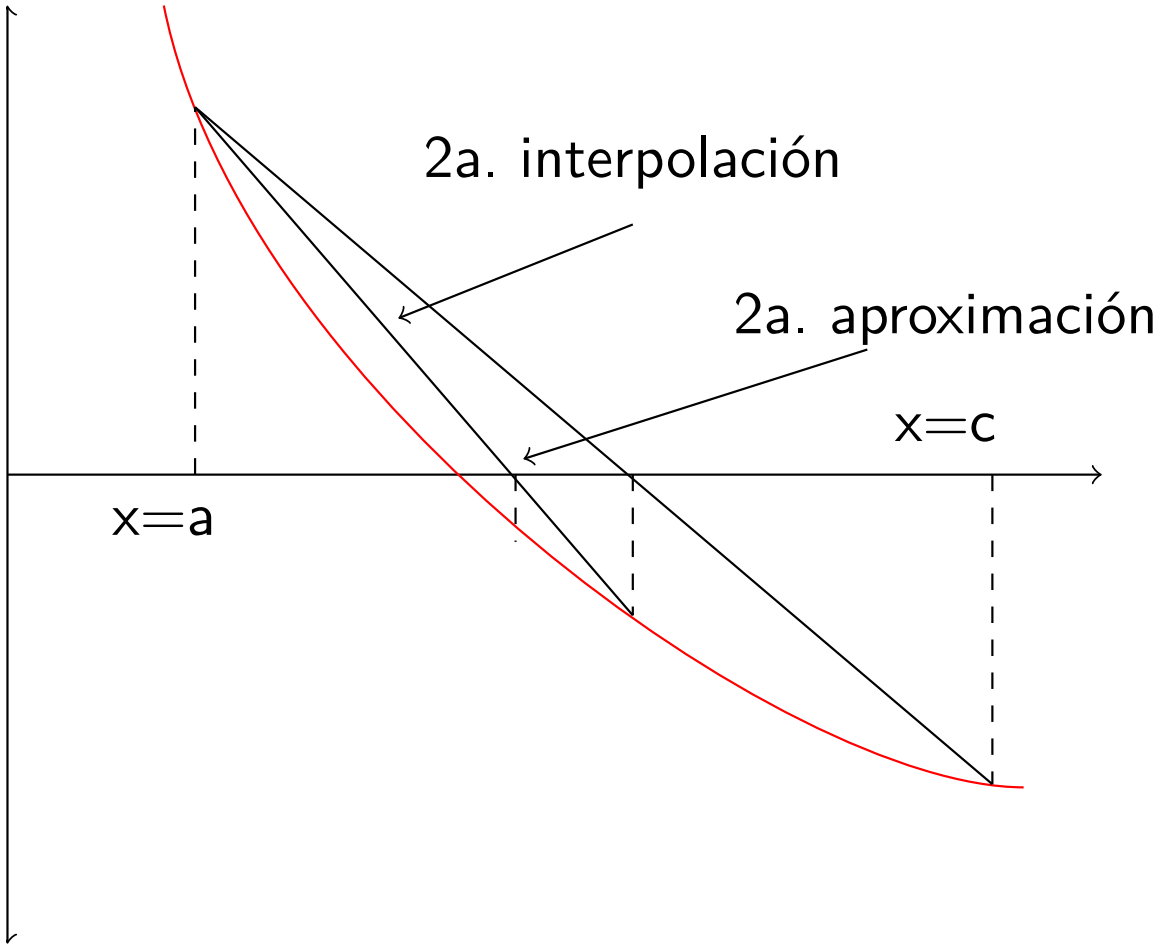


Figura (6): Primera interpolación lineal para determinar el cruce con el eje de las abscisas.

Método de la falsa posición

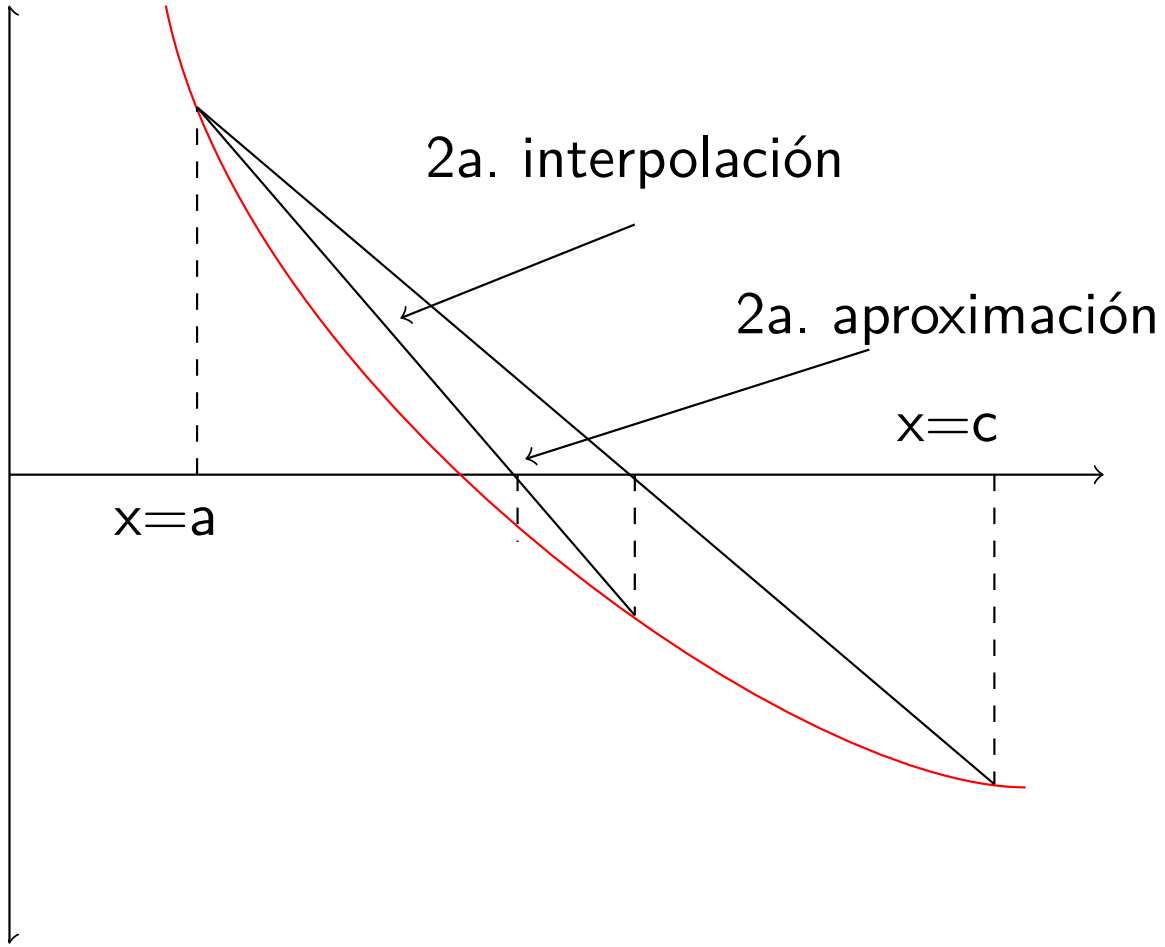


Figura (7): Luego de reasignar los extremos, se realiza una segunda interpolación lineal para determinar el nuevo cruce con el eje de las abscisas.

La desventaja de este método es que aparecen extremos fijos: uno de los extremos de la sucesión de intervalos no se mueve del punto original, por lo que las aproximaciones a la raíz, denotadas por b_1, b_2, b_3, etc. convergen a la raíz exacta solamente por un lado.

Los extremos fijos no son deseables debido a que hacen más lenta la convergencia, en particular cuando el intervalo es grande o cuando la función se desvía de manera significativa de una línea recta en el intervalo.

¿Qué podemos hacer al respecto?

Método de la falsa posición modificado

En este método, el valor de f en un punto fijo se divide a la mitad si este punto se ha repetido más de dos veces.

El extremo que se repite se llama extremo fijo. La excepción para esta regla es que para $i = 2$, el valor de f en un extremo se divide entre 2 de inmediato si no se mueve.

Método de falsa posición modificado

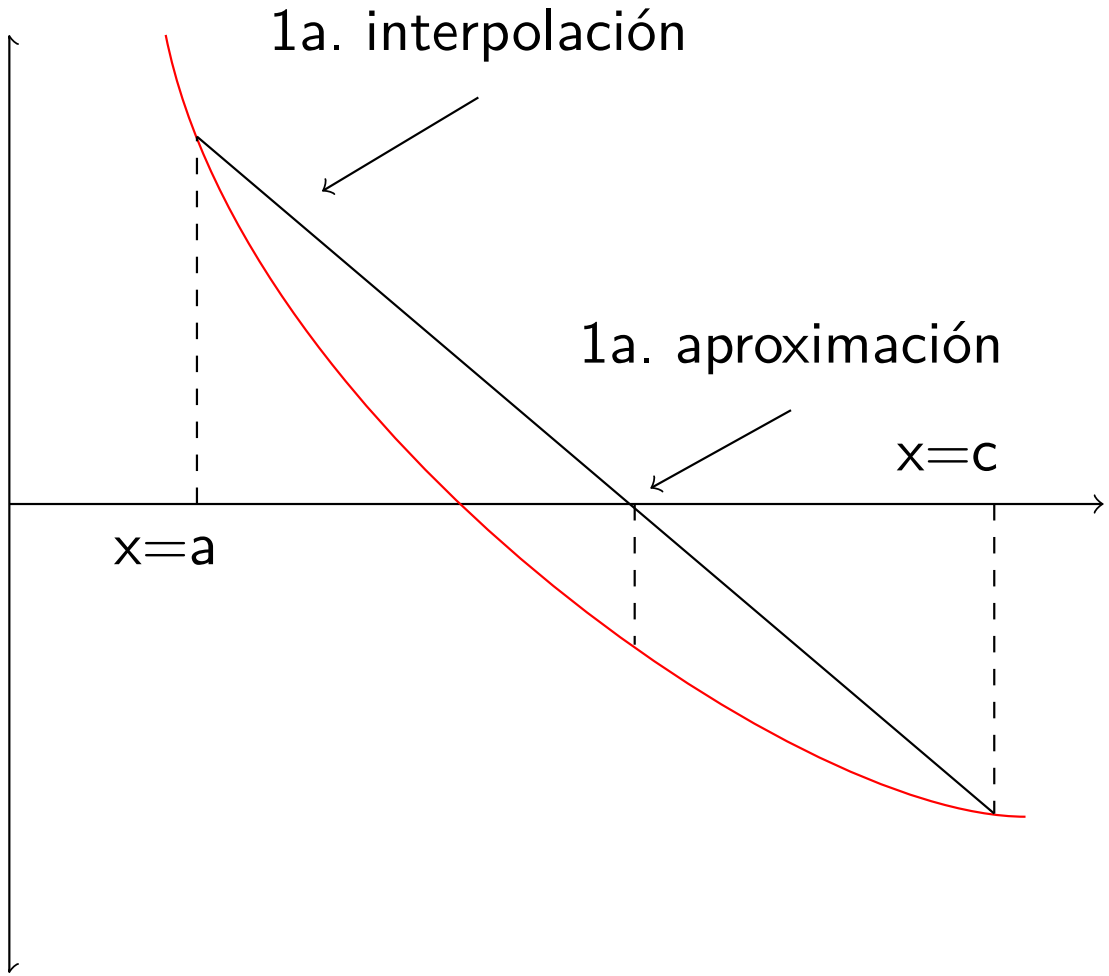


Figura (8): La primera interpolación lineal es idéntica que en el método de la falsa posición.

Método de la falsa posición modificado

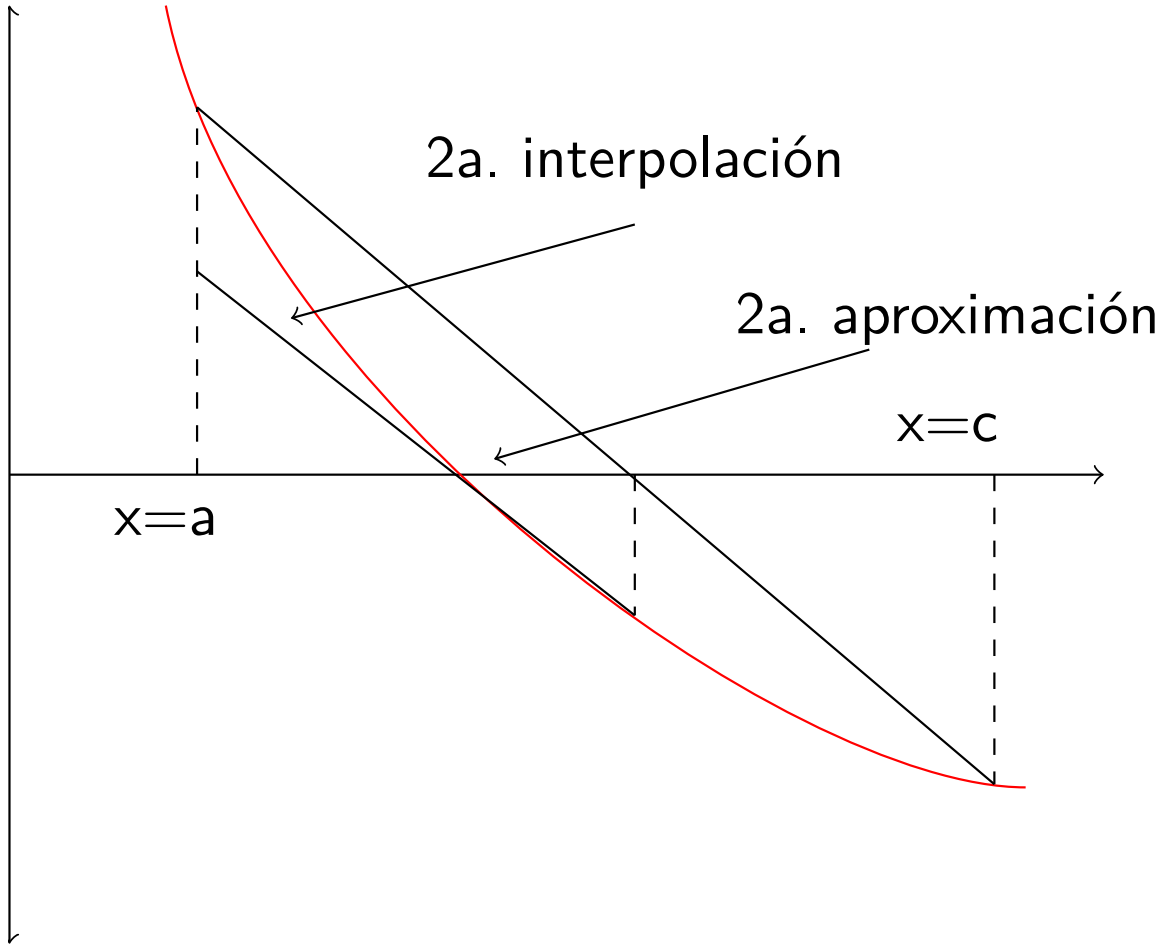


Figura (9): Se identifica el extremo que ya se utilizó y se toma el valor medio para realizar la siguiente interpolación lineal con el punto medio encontrado previamente.

Método de la secante

A diferencia del método de Newton, el valor de f' se aproxima utilizando dos valores de iteraciones consecutivas de f. Con lo que se elimina la necesidad de evaluar tanto a f como a f' en cada iteración.

Las aproximaciones sucesivas para la raíz en el método de la secante están dadas por:

$$x_n = x_{n-1} - y_{n-1}\frac{x_{n-1} - x_{n-2}}{y_{n-1} - y_{n-2}}, \qquad n = 2, 3, \ldots$$

donde x_0 y x_1 son dos suposiciones iniciales para comenzar la iteración.

Método de la secante

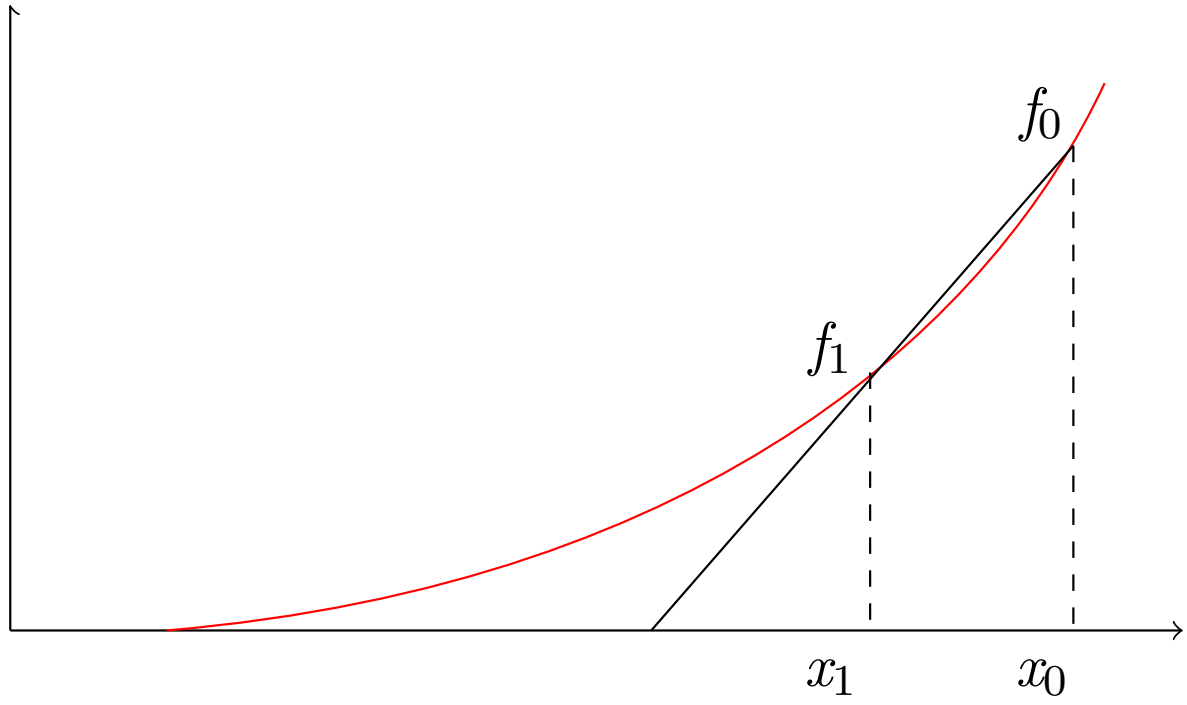


Figura (10): La inteporlación lineal se inicia con los valores evaluados en la función $f(x)$ para x_0 y x_1, que son puntos estimados inicialmente.

Método de la secante

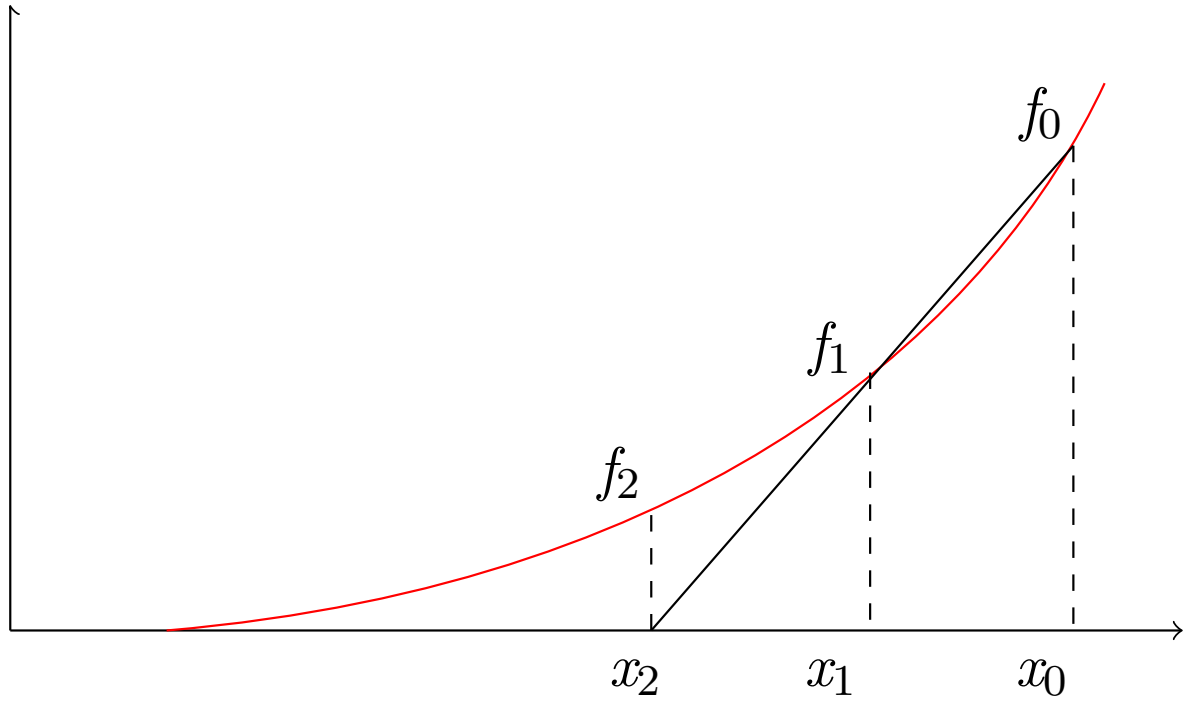


Figura (10): La inteporlación lineal se inicia con los valores evaluados en la función $f(x)$ para x_0 y x_1, que son puntos estimados inicialmente.

Método de la secante

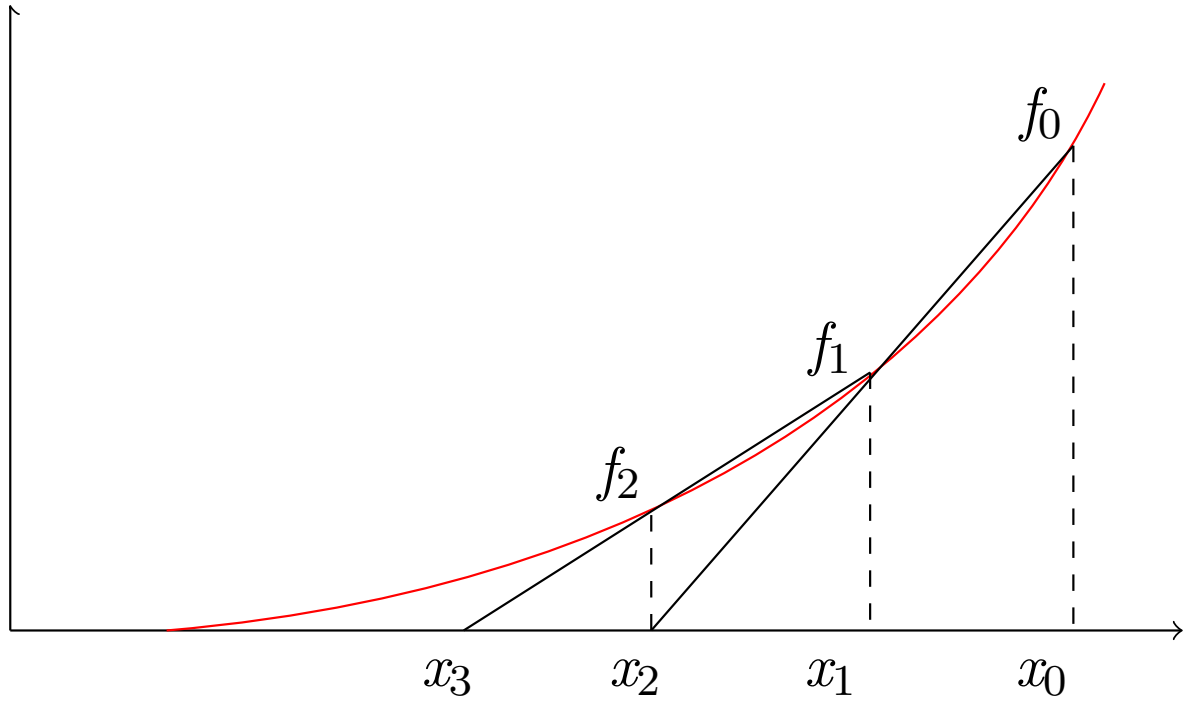


Figura (10): La inteporlación lineal se inicia con los valores evaluados en la función $f(x)$ para x_0 y x_1, que son puntos estimados inicialmente.

Atento aviso

En las técnicas de falsa posición, falsa posición modificada y el método de la secante, no hemos presentado como tal un código en `Python` que nos devuelva las raíces, por lo que tendrás que proponer un código para cada una de las técnicas.

Ese código lo vas a utilizar par resolver los problmeas y ejercicios del examen.

Contenido

Uso de la librería `scipy.optimize`

Las técnicas anteriormente descritas para calcular las raíces de una función se pueden recuperar de la librería `scipy.optimize`.

Hay que considerar cuidadosamente una serie de puntos para no tener problemas.

scipy.optimize.bisect

La sintaxis mínima para esta función es la siguiente:

```
scipy.optimize.bisect(f, a, b)
```

Encuentra la raíz de una función para un intervalo dado.

Implementa una rutina básica de bisección para encontrar el cero de una función f en el intervalo a y b. Los valores de $f(a)$ y $f(b)$ no deben de tener el mismo signo. Es un método lento, pero seguro.

Implementación del código

```
1 def p(x):
2     y = x**3 - 10 * x**2 + 5
3     return y
4
5 raiz = sp.optimize.bisect(p, -1, 0)
```

Toma en cuenta que el intervalo $[0, 1]$ en la función `optimize.bisect` se proporcionó de manera arbitraria, pero sería muy conveniente que incluyeras el código de los incrementos sucesivos para que también localice la segunda raíz.

```
1 a, b, dx = (-1.,0, 0.2)
2
3 x1, x2 = buscaraiz(p,a,b,dx)
4
5 raiz=sp.optimize.bisect(p, x1, x2)
```

Habrá que implementar que el código continue en la búsqueda de más raíces en el intervalo, para ello, tendrás que ajustar el código con un bucle.

scipy.optimize.newton

La función que utiliza el método de Newton-Raphson incorporada en `scipy.optimize` tiene la sintaxis:

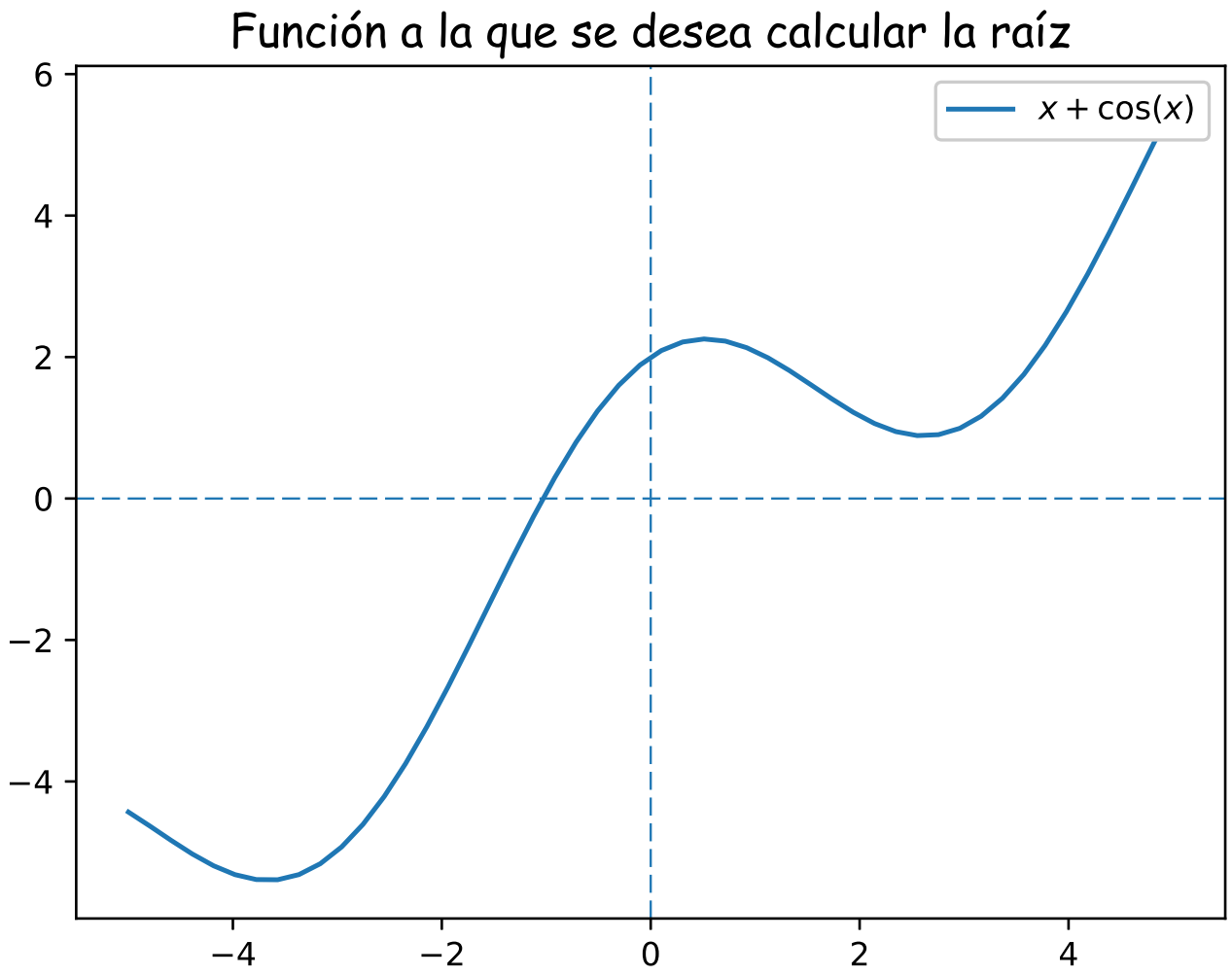
```
scipy.optimize.newton(func, x0, fprime=None)
```

Encuentra un cero usando el método de Newton-Raphson o el método de la secante.

Encuentra un cero de la función $func$ cercano a un punto inicial x_0. Se utiliza el método de Newton-Raphson si se proporciona la derivada de la función: `fprime`, en caso contrario, se utiliza el método de la secante.

Parámetros de la función

- $func$: es la función a la que se desea estimar la raíz.
- x_0 : es la estimación inicial para calcular la raíz.
- `fprime` : es un argumento opcional, corresponde a la derivada conocida de la función.


Función a la que se desea calcular la raíz
$x + \cos(x)$
6
4
2
0
−2
−4
−4
−2
0
2
4

Implementación del código

```
1 def f(x):
2     y = x + 2*scipy.cos(x)
3     return y
4
5 raiz= scipy.optimize.newton(f,   2)
```

Gráfica con la raíz encontrada