

# CAPÍTULO 3

## Resolución de Ecuaciones no Lineales

**3.1. Método de bisección.** El teorema de Bolzano nos dice que una función continua en un intervalo que cambia de signo, es decir que toma valores positivos y negativos, tiene que tener puntos donde se anula. En general, si toma dos valores reales  $y_1, y_2$ , debe tomar todos los valores intermedios. Como muchos resultados de la matemática, el teorema de Bolzano es un caso típico de teorema de existencia: asegura que lo que buscamos existe, en este caso una solución de la ecuación, pero no dice si es único, ni tampoco como encontrarlo. El método de bisección trata de hallar una solución aproximada basado en este teorema. La idea es sencilla: partimos de un intervalo  $[a_0, b_0]$  tal que las evaluaciones en los extremos tengan distinto signo, es decir  $f(a_0)f(b_0) < 0$ , tomamos un punto intermedio  $c_0 \in (a_0, b_0)$ . Si  $f(c_0) = 0$ , entonces  $x_* = c_0$  es una solución. Si  $f(c_0)f(a_0) > 0$ , definimos un nuevo intervalo  $[a_1, b_1] = [c_0, b_0]$ . En el caso  $f(c_0)f(a_0) < 0$ , tomamos el intervalo  $[a_1, b_1] = [a_0, c_0]$ . Tomemos como ejemplo la ecuación  $f(x) = 0$  para la función  $f(x) = 1.75x^3 - 3x - 1$ . En el intervalo  $[-2, 2]$  se verifica  $f(-2) < 0 < f(2)$ . Más aún, están las tres raíces del polinomio  $f(x)$ ,  $x = -1.09114, -0.360711, 1.45185$ . Las aplicaciones del método de bisección de muestran en la Tabla 3.1. Los cuatro primeros pasos se

$n$	$a_n$	$b_n$	$c_n$
0	-2.00000	2.00000	0.00000
1	0.00000	2.00000	1.00000
2	1.00000	2.00000	1.50000
3	1.00000	1.50000	1.25000
4	1.25000	1.50000	1.37500
5	1.37500	1.50000	1.43750
6	1.43750	1.50000	1.46875
7	1.43750	1.46875	1.45313
8	1.43750	1.45313	1.44531
9	1.44531	1.45313	1.44922
10	1.44922	1.45313	1.45117
11	1.45117	1.45313	1.45215
12	1.45117	1.45215	1.45166

Tabla 3.1: Iteraciones del método de bisección para  $f(x) = 1.75x^3 - 3x - 1$ .

muestran en la Figura 3.1. Observemos que no tenemos control a cual de la soluciones converge el algoritmo.

**3.1.1. Algoritmo.** El Algoritmo 3.1 realiza la búsqueda del cero de  $f(x)$  a partir de un intervalo donde hay un cambio de signo. En cada paso, la longitud del intervalo se reduce a la mitad manteniendo el cambio de signo. Después de  $n$  pasos, el programa devuelve el último intervalo y el punto medio.

**Algoritmo 3.1:** Método de bisección.

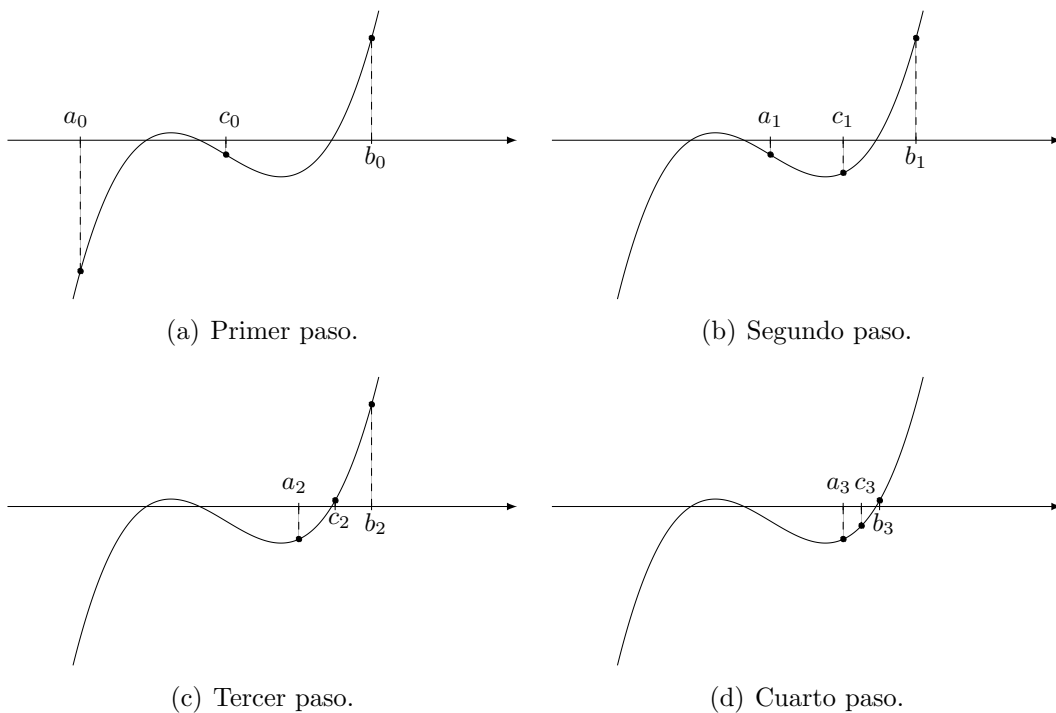
---

```

Data:  $f, a, b, n$ 
Result:  $a, b, c$ 
 $y_a = f(a);$ 
 $y_b = f(b);$ 
for  $i = 0$  to  $n$  do
   $c = 0.5 * (a + b);$ 
  print  $i, a, b, c;$ 
   $y_c = f(c);$ 
  if  $y_c == 0$  then
     $a = c;$ 
     $b = c;$ 
  else if  $y_a * y_c > 0$  then
     $a = c;$ 
     $y_a = y_c;$ 
  else
     $b = c;$ 
     $y_b = y_c;$ 
  end
end
return  $a, b, c;$ 

```

---

Fig. 3.1: Método de bisección para la función  $f(x) = 1.75x^3 - 3x - 1$ .

**3.1.2. Fórmula del error.** Es fácil ver que la longitud del intervalo se reduce a la mitad en cada paso, por lo que vemos que  $b_n - a_n = 2^{-n}(b_0 - a_0)$ , es decir  $b_n - a_n \cong 10^{-3}(b_0 - a_0)$  para  $n = 10$ . Cada paso requiere una evaluación de la función.

**3.2. Método de Newton.** El método de Newton es un procedimiento iterativo que permite hallar la solución de la ecuación  $f(x) = 0$ . El método consiste en aproximar la función

$f(x)$  por la recta tangente en un punto inicial y buscar los ceros de la ecuación aproximada,  $f(x_0) + f'(x_0)(x - x_0) = 0$ . El punto hallado  $x_1$  reemplaza al punto anterior y se comienza nuevamente. Se obtiene una sucesión  $x_0, x_1, x_2, \dots$  que, en caso de ser convergente, tiene como límite una solución del problema. Concretamente, a partir de  $x_0$  se construye recursivamente la sucesión

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

Observemos que se necesita evaluar la función y su derivada, esto puede ser un problema cuando  $f(x)$  no tiene una expresión conocida, sino que es el resultado de otros cálculos. La convergencia no está garantizada (ver Figura 3.4), pero es fácil ver bajo condiciones si la sucesión converge a un punto  $x_*$ , entonces  $f(x_*) = 0$ . En efecto, como  $f(x_{n-1}) = -f'(x_{n-1})(x_n - x_{n-1})$ , si  $x_n \rightarrow x_*$  entonces  $x_n - x_{n-1} \rightarrow 0$  y  $f'(x_{n-1}) \rightarrow f'(x_*)$ , por lo tanto  $f(x_{n-1}) \rightarrow 0$ .

**3.2.1. Interpretación geométrica.** Para ilustrar el método vamos a considerar diferentes ejemplos. Estudiamos primero la ecuación  $e^x = 45$  cuya solución es  $x = \ln 45 \cong 3.8066624898$ . El problema es equivalente a la ecuación  $f(x) = 0$ , donde  $f(x) = e^x - 45$ , la sucesión se obtiene recursivamente de la forma

$$x_n = x_{n-1} - \frac{e^{x_{n-1}} - 45}{e^{x_{n-1}}} = x_{n-1} - 1 + 45 e^{-x_{n-1}}$$

En la Tabla 3.2 mostramos las iteraciones partiendo de  $x_0 = 6$ . En la segunda columna vemos como disminuyen los errores, primero lentamente y después en forma acelerada. Eso se refleja en la tercera columna donde se compara  $\epsilon_n$  con  $\epsilon_{n-1}^2$ . En la Figura 3.2 mostramos los primeros

$n$	$x_n$	$\epsilon_n$	$\epsilon_n/\epsilon_{n-1}^2$
0	6.0000000000	-2.193	
1	5.1115438479	-1.305	-0.271
2	4.3827485573	-0.576	-0.338
3	3.9448426224	-0.138	-0.416
4	3.8157844150	$-0.912 \times 10^{-2}$	-0.478
5	3.8067039683	$-0.415 \times 10^{-4}$	-0.498
6	3.8066624906	$-0.860 \times 10^{-9}$	-0.500

Tabla 3.2: Iteraciones del método de Newton para  $e^x = 45$ .

puntos y las rectas tangentes

**3.2.2. Algoritmo de Newton.** En el código 3.2 tenemos el método de Newton. El programa llama a las rutinas que evalúan  $f(x)$  y  $f'(x)$ . Devuelve el valor calculado de  $x$  y un código de error: code que toma el valor cero si el programa termina correctamente. Se consideran dos criterios de parada: cuando dos iteraciones distan menos que  $\delta$ , lo que podría interpretarse como convergencia a una solución, o cuando el número de iteraciones supera un valor máximo  $N_{\text{máx}}$ , en ese caso se considera que el método no alcanzó un resultado, posiblemente porque el método no es convergente.

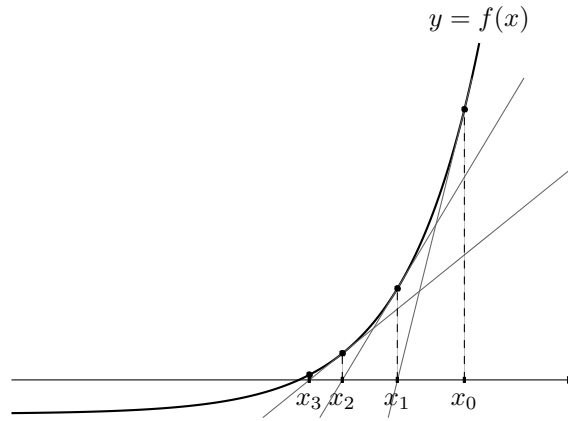


Fig. 3.2: Gráfico de  $f(x) = e^x - 45$  y las iteraciones del método de Newton.

---

**Algoritmo 3.2:** Método de Newton.

---

**Input:**  $f, f', x_0, \delta, N_{\max}$   
**Output:**  $x$ , code  
 $n = 1;$   
 $x_{\text{old}} = x_0;$   
 $x_{\text{new}} = x_{\text{old}} - f(x_{\text{old}})/f'(x_{\text{old}});$   
**while**  $|x_{\text{new}} - x_{\text{old}}| > \delta \wedge n \leq N_{\max}$  **do**  
     $n = n + 1;$   
     $x_{\text{old}} = x_{\text{new}};$   
     $x_{\text{new}} = x_{\text{old}} - f(x_{\text{old}})/f'(x_{\text{old}});$   
**end**  
**if**  $n \leq N_{\max}$  **then**  
    code = 0;  
**else**  
    code = 1;  
**end**  
**return**  $x_{\text{new}}, \text{code};$

---

Observemos que estos controles no pueden considerarse válidos en todos los casos. Por ejemplo, la ecuación  $f(x) = 2 + x^3 + \tanh(a(x - 1)) = 0$  con  $a = 5 \times 10^3$ , tiene solución  $x_* \cong -1$ , pero partiendo del punto  $x_0 = 4.6149$  obtenemos  $x_1 = 3.02965, x_2 = 1.91082, x_3 = 0.999997, x_4 = 0.9994$ . Vemos que  $|x_4 - x_3| < 6 \times 10^{-4}$ . Por lo tanto, si  $\delta = 10^{-3}$ , el programa da como resultado  $x = x_4$ , muy lejos del resultado correcto. Por otro lado, si  $x_0 = 0.001$  podemos ver que  $|x_n - x_{n-1}| < 10^{-3}$  recién para  $n = 36$ , con  $x_{35} = -1.00002$  y  $x_{36} \cong -1$ . Si  $N_{\max} < 36$ , el programa entregaría un valor de  $x$  incorrecto.

**Ejemplo 3.1.** La ley de Planck establece la radiación electromagnética emitida por un cuerpo negro en equilibrio térmico a una temperatura  $T$  definida. La densidad de radiación en la longitud de onda  $\lambda$  está dada por

$$v(\lambda, T) = \frac{8\pi hc}{\lambda^5 \left( \exp\left(\frac{hc}{kT\lambda}\right) - 1 \right)}$$

donde  $h = 6.626 \times 10^{-34}$  Js es la constante de Planck,  $c = 2.998 \times 10^8$  m s $^{-1}$  es la velocidad de la luz en el vacío y  $k = 1.381 \times 10^{-23}$  J K $^{-1}$  es la constante de Boltzmann. En la Figura 3.3 se grafica la densidad  $v(\lambda)$  (en kJ m $^{-3}$  nm $^{-1}$ ) en función  $\lambda$  (en nm). La longitud de onda donde la

densidad es máxima se obtiene resolviendo

$$\frac{\partial v}{\partial \lambda}(\lambda, T) = \frac{8\pi hc e^x}{\lambda^6 (e^x - 1)^2} (x + 5e^{-x} - 5) = 0,$$

con  $x = hc/(kT\lambda)$ . Para hallar la solución de la ecuación  $x + 5e^{-x} - 5 = 0$ , con  $x > 0$ , usamos el método de Newton

$$x_n = 5 \frac{e^{x_{n-1}} - 1 - x_{n-1}}{e^{x_{n-1}} - 1}$$

obteniendo la solución  $x = 4.96511$ . Por lo tanto vale la relación

$$\lambda_{\max} = \frac{hc}{4.96511kT} = \frac{0.289805 \times 10^{-2} \text{ K m}}{T},$$

conocida como ley de Wien. A la temperatura del sol,  $T \cong 6000 \text{ K}$ ,  $\lambda_{\max} = 483 \text{ nm}$ .

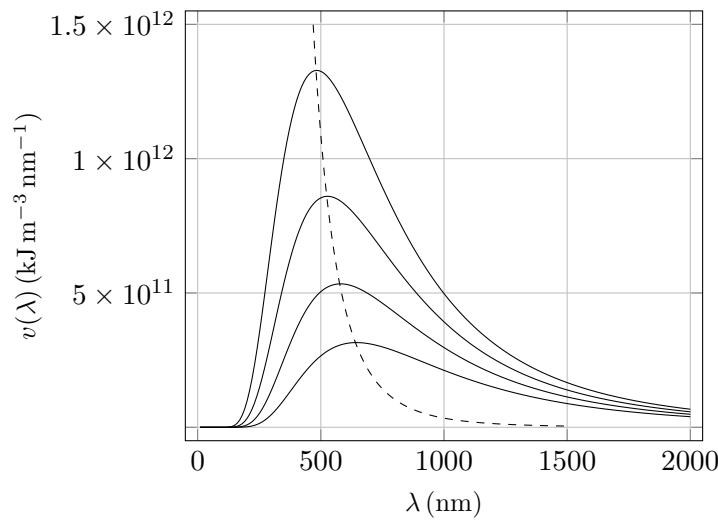


Fig. 3.3: Gráfico de  $v(\lambda)$ ,  $T = 4500, 5000, 5500, 6000$ . En línea de puntos mostramos  $\lambda_{\max}$  para cada  $T$ .

**3.2.3. Fórmula del error.** Vamos a ver como evoluciona el error  $\epsilon_n = x_* - x_n$ . Si tomamos el polinomio de Taylor de segundo orden de  $f$  alrededor de  $x_n$ , tenemos

$$0 = f(x_*) = f(x_{n-1}) + f'(x_{n-1})\epsilon_{n-1} + \frac{f''(\xi_{n-1})}{2}\epsilon_{n-1}^2,$$

para algún  $\xi_{n-1}$  en el intervalo de extremos  $x_*, x_{n-1}$ . Como  $0 = f(x_{n-1}) + f'(x_{n-1})(x_n - x_{n-1})$ , usando que  $\epsilon_{n-1} = \epsilon_n + x_n - x_{n-1}$  obtenemos  $0 = f'(x_{n-1})\epsilon_n + \frac{1}{2}f''(\xi_{n-1})\epsilon_{n-1}^2$ , asumiendo que  $f'(x_{n-1}) \neq 0$ , tenemos

$$(3.1) \quad \epsilon_n = -\frac{f''(\xi_{n-1})}{2f'(x_{n-1})}\epsilon_{n-1}^2.$$

Si en un intervalo  $I = [x_* - r, x_* + r]$ ,  $f'(x)$  no se anula entonces  $c_1 = \min_{x \in I} |f'(x)| > 0$ . Definamos  $c_2 = \max_{x \in I} |f''(x)|$ , entonces si  $\epsilon_0 = x_* - x_0$  verifica  $|\epsilon_0| < r$  y  $c_2/c_1 |\epsilon_0| = \gamma < 1$ , entonces por (3.1) tenemos  $|\epsilon_1| < c_2/(2c_1)\epsilon_0^2 < \gamma |\epsilon_0| < r$ , por lo tanto  $x_1 \in I$ . Inductivamente, se obtiene  $x_n \in I$  y  $|\epsilon_n| < \gamma^{-n}r \rightarrow 0$ , lo que implica  $x_n \rightarrow x_*$ . Entonces  $\xi_n \rightarrow x_*$ , de donde se obtiene  $\epsilon_n/\epsilon_{n-1}^2 \rightarrow -f''(x_*)/(2f'(x_*))$ .

Vimos que si  $f'(x_*) \neq 0$ , el error se comporta en forma cuadrática, es decir  $\epsilon_n/\epsilon_{n-1}^2 \rightarrow c$ . Vamos a analizar el error cuando  $f'(x_*) = 0$ , tomemos como ejemplo  $f(x) = a(x - x_*)^m$ . Claramente  $f'(x_*) = 0$ , los errores se relacionan de la forma

$$\epsilon_n = x_* - x_n = x_* - \left( x_{n-1} - \frac{a(x_{n-1} - x_*)^m}{m a(x_{n-1} - x_*)^{m-1}} \right) = \left( 1 - \frac{1}{m} \right) \epsilon_{n-1}.$$

Vemos que  $\epsilon_n \rightarrow 0$ , pero el error decrece linealmente, más lentamente a medida que  $m$  es mayor. En el caso general, si  $f(x_*) = 0, \dots, f^{(m-1)}(x_*) = 0$ , cerca de  $x_*$  la función  $f(x) \cong a(x - x_*)^m$ , por lo que su comportamiento asintótico será similar al caso anterior.

Por ejemplo, la ecuación  $f(x) = (x - 2\pi)(\cos(x) - 1) = 0$ , el punto  $x_* = 2\pi$  es un cero de tercer orden, es decir  $f(x_*) = 0, f'(x_*) = 0, f''(x_*) = 0$ . Tomamos  $x_0 = 4$ , en la Tabla 3.3 mostramos las iteraciones y los errores, vemos que  $\epsilon_n/\epsilon_{n-1} \rightarrow 2/3$ .

$n$	$x_n$	$\epsilon_n$	$\epsilon_n/\epsilon_{n-1}$
0	4.00000	2.28319	
1	5.11652	1.16667	0.51098
2	5.53803	0.74516	0.63871
3	5.79440	0.48879	0.65595
4	5.95953	0.32366	0.66216
5	6.06805	0.21514	0.66471
6	6.13994	0.14324	0.66581

Tabla 3.3: Iteraciones de la ecuación  $(x - 2\pi)(\cos(x) - 1) = 0$ .

**Convergencia para funciones no analíticas** El análisis anterior asume que alguna derivada es diferente de cero, existen funciones que no verifican esto, por ejemplo  $f(x) = xe^{-1/x^2}$  ( $f(0) = 0$ ). Es posible mostrar que  $f^{(m)}(0) = 0$  para  $m \geq 0$ . El único cero de la función es  $x_* = 0$ . Numéricamente podemos ver que la convergencia es muy lenta, si  $x_0 = 1$ , obtenemos  $x_{100} = 0.0989013$ ,  $x_{1000} = 0.0315785$ ,  $x_n = 0.00141421$  para  $n = 5 \times 10^5$ . Podemos ver que  $x_n = 2x_{n-1}/(2 + x_{n-1}^2)$ , de donde resulta  $\epsilon_n/\epsilon_{n-1} \rightarrow 1$ .

**3.2.4. Resultados de convergencia global.** Consideremos la ecuación  $\tanh(x) = 0.25$ , cuya solución  $x = 0.255413$ . Veamos que pasa con diferentes puntos iniciales, si comenzamos con el punto  $x_0 = 1.2$ , vemos que la sucesión converge a la solución como se ve en la Tabla 3.4. Pero si el punto inicial es  $x_0 = 1.285$ , el comportamiento cambia. Los primeros términos de la sucesión son  $x_1 = -1.01592$ ,  $x_2 = 1.4683$ ,  $x_3 = -1.92482$ ,  $x_4 = 12.8762$ ,  $x_5 = -2.86493 \times 10^{10}$ , lo que muestra la no convergencia del método de Newton. En la Figura 3.4 mostramos gráficamente el comportamiento del método para los distintos punto iniciales.

$n$	$x_n$	$\epsilon_n$	$\epsilon_n/\epsilon_{n-1}^2$
0	1.200000	-0.944587	
1	-0.713496	0.968909	1.085920
2	0.668420	-0.413007	-0.439938
3	0.161694	$0.937186 \times 10^{-1}$	0.549427
4	0.253760	$0.165250 \times 10^{-2}$	0.188144
5	0.255412	$0.679682 \times 10^{-6}$	0.248899
6	0.255413	$0.115519 \times 10^{-12}$	0.250058

Tabla 3.4: Iteraciones del método de Newton para  $\tanh(x) = 0.25$ .

Veamos que hay hipótesis sobre  $f(x)$  que garantizan la convergencia del método de Newton en forma incondicional. Supongamos que  $f''(x) > 0$ , es decir  $f(x)$  es estrictamente convexa, si

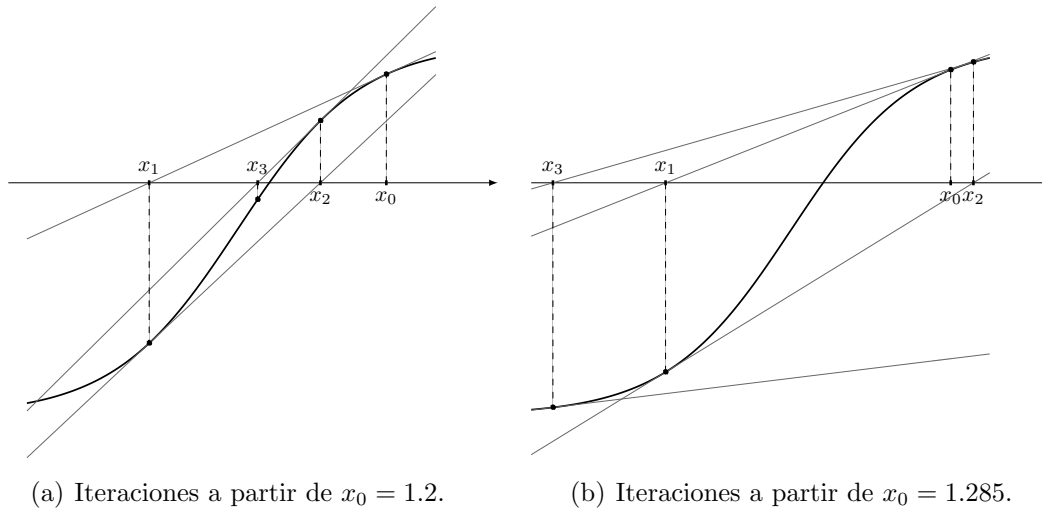


Fig. 3.4: Gráfico de  $f(x) = \tanh(x) - 0.25$  y las iteraciones del método de Newton.

$x_*$  no es mínimo de  $f(x)$ , entonces  $x_n \rightarrow x_*$  (si  $x_0$  tampoco es un punto mínimo). En efecto, consideremos  $f'(x_*) > 0$ , si  $x_0 > x_*$ , es decir  $\epsilon_0 < 0$ , entonces  $f'(x_0) > 0$  y  $f(x_0) > 0$ . Entonces  $x_1 < x_0$  y por la ecuación (3.1),  $\epsilon_1 < 0$ . En conclusión,  $x_* < x_1 < x_0$  y en general  $x_* < x_n < x_{n-1} < \dots < x_0$ . Siendo  $x_n$  una sucesión decreciente y acotada, es convergente y el límite tiene que ser una solución. Si  $x_0 < x_*$  y  $f'(x_0) > 0$ , como  $f''(x) > 0$ , tenemos  $x_1 > x_*$ .

**Fast inverse square root:** Dado  $x > 0$ , se quiere calcular  $y = x^{-1/2}$ , o en forma equivalente  $1/y^2 - x = 0$ . La iteración del método de Newton es

$$y_{n+1} = y_n \frac{3 - y_n^2 x}{2} = y_n (1.5 - 0.5 x y_n^2).$$

El algoritmo *fast inverse square root* se basa en hallar un punto  $y_0$  inicial cercano a  $y_*$ , lo que permite en pocas iteraciones (una o dos) del método de Newton obtener una buena aproximación. Si partimos de la representación en punto flotante  $x = (1 + m)2^e$ , tenemos  $\log_2(x) = e + \log_2(1 + m) \cong e + m + \sigma$ . Se puede ver que la mejor elección de  $\sigma$  es

$$\sigma = \frac{\ln(2) - 1 - \ln(\ln(2))}{2 \ln(2)} \cong 0.0430357,$$

en el sentido que minimiza la distancia  $\max_{m \in [0,1]} |\log_2(1 + m) - m - \sigma|$ . La clave consiste en considerar a la representación en punto flotante como un número entero. Si definimos el entero  $I(x) = b_0(x) + \dots + b_{62}(x) \times 2^{62}$  ( $b_{63}(x) = 0$  dado que  $x > 0$ ), tenemos

$$I(x) = m \times 2^{52} + (e + 1023) \times 2^{52} = (e + m + \sigma + 1023 - \sigma) \times 2^{52} \cong \log_2(x) + (1023 - \sigma) \times 2^{52}.$$

Usando que  $\log_2(y_*) = -1/2 \log_2(x)$ , obtenemos

$$\begin{aligned} I(y) &\cong -\frac{1}{2} \log_2(x) + (1023 - \sigma) \times 2^{52} \cong -\frac{1}{2} (I(x) - (1023 - \sigma) \times 2^{52}) + (1023 - \sigma) \times 2^{52} \\ &= \frac{3}{2} (1023 - \sigma) \times 2^{52} - \frac{1}{2} I(x) \end{aligned}$$

En [https://github.com/id-Software/Quake-III-Arena/blob/master/code/game/q\\_math.c#L552](https://github.com/id-Software/Quake-III-Arena/blob/master/code/game/q_math.c#L552) se puede encontrar el código en lenguaje C del video juego Quake-III-Arena, donde se muestra la implementación del algoritmo *Fast inverse square root*. En la línea 560 se considera al número  $y$  en punto flotante como un entero largo  $i$ . Se usa una sola iteración del método de Newton como se ve en 563-564. En 561 se observa la constante expresada en hexadecimal y la división de  $i$  por 2 mediante un desplazamiento de los bits hacia la izquierda. Por los comentarios se puede deducir que el programador no comprendía completamente el algoritmo.

```
552 float Q_rsqrt( float number )
553 {
554     long i;
```

```

555 float x2, y;
556 const float threehalfs = 1.5F;
557
558 x2 = number * 0.5F;
559 y = number;
560 i = * ( long * ) &y; // evil floating point bit level
    hacking
561 i = 0x5f3759df - ( i >> 1 ); // what the fuck?
562 y = * ( float * ) &i;
563 y = y * ( threehalfs - ( x2 * y * y ) ); // 1st iteration
564 // y = y * ( threehalfs - ( x2 * y * y ) ); // 2nd iteration, this can be
    removed
565
566 return y;
567 }

```

**3.3. Método de punto fijo.** El métodos de Newton es caso particular de los llamados métodos de punto fijo, estos métodos consisten en transformar la ecuación original en hallar la solución de  $\phi(x) = x$  y buscarla mediante la sucesión  $x_1 = \phi(x_0), x_2 = \phi(x_1), x_3 = \phi(x_2), \dots$ , a partir de un punto  $x_0$  apropiado. Vale la pena aclarar que los problemas de punto fijo tienen interés en si mismo, muchas veces la evolución de un sistema se modela como la aplicación de una función al estado actual. En este caso, un punto fijo representa un estado estacionario, es decir un punto de equilibrio.

En general la forma de transformar el problema original en la ecuación de punto fijo no es única y no todas son apropiadas para obtener una solución aproximada. El siguiente ejemplo es una muestra de esta observación.

**Ejemplo 3.2.** Consideremos el problema  $x^2 - 4 = 0$  cuyas soluciones son  $x = \pm 2$ . Si sumamos  $x$  a ambos miembros obtenemos la ecuación de punto fijo  $x = \phi_1(x) = x^2 + x - 4$ , podemos ver que la sucesión que se obtiene con  $\phi_1$  diverge para cualquier punto inicial, salvo si  $x_0 = \pm 2$ . Tomemos  $x_0 = 1.9$ , las iteraciones sucesivas resultan  $x_1 = 1.51, x_2 = -0.2099, x_3 = -4.16584, x_4 = 9.1884, x_5 = 89.615, x_6 = 8116.47$ . Obtendríamos algo similar partiendo de otros puntos iniciales. Esta forma de buscar las soluciones como una ecuación de punto fijo es completamente inútil.

Una manera alternativa de convertir el problema original en una ecuación de punto fijo, consiste en hacer la siguientes transformaciones: dividiendo por 2 obtenemos la ecuación  $x^2/2 - 2 = 0$ , sumando luego  $x^2/2$  en ambos miembros resulta  $x^2 - 2 = x^2/2$ , por último despejamos la variable  $x$  del lado derecho como  $x = \phi_2(x) = (x^2/2 + 2)^{1/2}$ . Observemos que siendo  $\phi_2(x) > 0$ , las iteraciones no pueden converger a la solución negativa  $x = -2$ . En la Tabla 3.5 mostramos solamente las iteraciones pares por cuestiones de espacio para el punto inicial  $x_0 = 8$ .

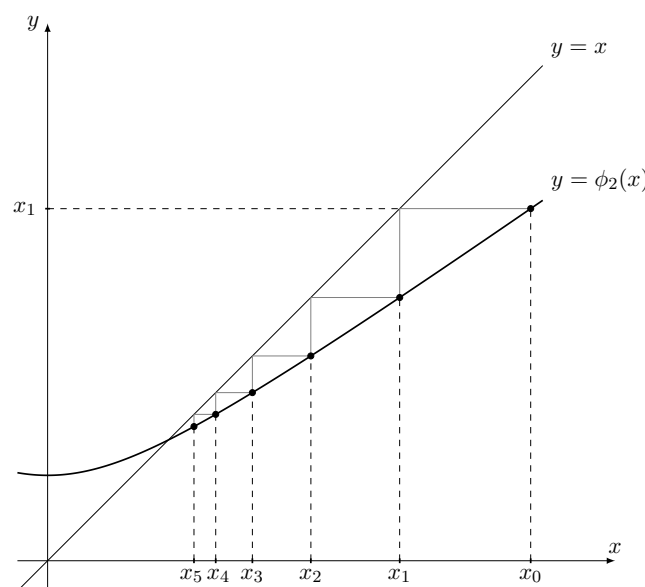
Observemos que a medida que nos acercamos a la solución  $x_* = 2$ , los errores disminuyen con una tasa fija, 0.5 en este caso. Podemos ver en forma gráfica de ver las sucesivas iteraciones, conocido un punto  $x_n$  sobre el eje  $x$  subimos verticalmente hasta el gráfico de  $y = \phi_2(x)$ , esa altura corresponde a  $x_n = \phi_2(x_{n-1})$ . Para ubicar el punto sobre el eje de las abscisas, trazamos una recta horizontal hasta la intersección con la recta  $y = x$  y luego verticalmente hasta el eje  $x$ . En la Figura 3.5 mostramos las primeras iteraciones correspondientes al punto inicial  $x_0 = 8$ .

**3.3.1. Análisis de convergencia para funciones derivables.** Vamos a estudiar ahora el comportamiento de las iteraciones en el Ejemplo 3.2. En general, buscamos condiciones que permitan asegurar la convergencia a una solución del problema. Si  $x_*$  es la solución, entonces  $x_* = \phi(x_*)$ , usando la recurrencia  $x_n = \phi(x_{n-1})$ , tenemos

$$(3.2) \quad \epsilon_n = x_* - x_n = \phi(x_*) - \phi(x_{n-1}) = \phi'(\xi_{n-1})(x_* - x_{n-1}) = \phi'(\xi_{n-1})\epsilon_{n-1},$$



$n$	$x_n$	$\epsilon_n$	$\epsilon_n/\epsilon_{n-1}$
0	8.00000	-6.00000	
2	4.35890	-2.35890	0.615747
4	2.78388	-0.78388	0.563472
6	2.22205	-0.22205	0.523897
8	2.05776	$-0.57759 \times 10^{-1}$	0.506923
10	2.01460	$-0.14595 \times 10^{-1}$	0.501805
12	2.00366	$-0.36588 \times 10^{-2}$	0.500456
14	2.00092	$-0.91532 \times 10^{-3}$	0.500114
16	2.00023	$-0.22887 \times 10^{-3}$	0.500029
18	2.00006	$-0.57219 \times 10^{-4}$	0.500007
20	2.00001	$-0.14305 \times 10^{-4}$	0.500002

Tabla 3.5: Iteraciones de  $y = \phi_2(x)$ .Fig. 3.5: Gráfico de  $\phi_2(x)$  y las iteraciones del método de punto fijo.

donde  $\xi_{n-1}$  es un punto del intervalo determinado por  $x_*$  y  $x_{n-1}$ . Supongamos que un intervalo  $I = [x_* - r, x_* + r]$  se verifica  $|\phi'(x)| \leq \gamma < 1$ , entonces por (3.2) para un punto inicial  $x_0 \in I$  se verifica  $|\epsilon_1| \leq \gamma |\epsilon_0| < |\epsilon_0|$ , lo que muestra que  $x_1 \in I$ . De la misma forma,  $|\epsilon_2| \leq \gamma |\epsilon_1| \leq \gamma^2 |\epsilon_0|$  y en general  $x_n \in I$ ,  $|\epsilon_n| = \gamma^n |\epsilon_0|$ . Por lo tanto  $|\epsilon| \rightarrow 0$  o en forma equivalente  $x_n \rightarrow x_*$ . Dado que  $\xi_n \rightarrow x_*$ , vemos que  $\epsilon_n/\epsilon_{n-1} \rightarrow |\phi'(x_*)|$ . En el Ejemplo 3.2, se verifica  $\phi'(2) = 0.5$  lo que observa en la tasa de convergencia  $\epsilon_n/\epsilon_{n-1}$ .

Como ya vimos, el método de Newton es una ecuación de punto fijo con  $\phi(x) = x - f(x)/f'(x)$

$$\phi'(x) = \frac{f(x)f''(x)}{f'(x)^2},$$

por lo tanto  $\phi'(x_*) = 0$  y  $\phi'(x) \cong -(x_* - x)f''(x_*)/f'(x_*)$ .

**Ejercicio 3.1.** Mostrar que para cualquier intervalo acotado  $I$ , existe  $0 \leq \gamma < 1$  tal que  $|\phi'_2(x)| \leq \gamma < 1$  para  $x \in I$ . Probar que el método de punto fijo converge a una solución en forma incondicional, es decir para cualquier punto inicial.

**Ejercicio 3.2.** Estudiar numéricamente las iteraciones de  $\phi(x) = \lambda x(1 - x)$  para  $x_0 = 0.3$ , en los casos  $\lambda = 0.9, 2.5, 3.56, 3.9$ .

**3.4. Métodos de punto fijo para sistemas de ecuaciones.** Consideremos el sistema de ecuaciones, con  $d$  ecuaciones y  $d$  incógnitas

$$(3.3) \quad \begin{aligned} 0 &= f_1(x_1, x_2, \dots, x_d), \\ 0 &= f_2(x_1, x_2, \dots, x_d), \\ &\vdots \\ 0 &= f_d(x_1, x_2, \dots, x_d). \end{aligned}$$

Igual que en el caso escalar, se puede transformar el sistema de ecuaciones (3.3) en un problema de punto fijo  $\mathbf{x} = \phi(\mathbf{x})$ , con  $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ . Igual que en el caso escalar, podemos plantear un método iterativo  $\mathbf{x}_n = \phi(\mathbf{x}_{n-1})$ . Si la sucesión es convergente, el límite tiene que ser un punto fijo. Por ejemplo, consideramos el sistema

$$\begin{cases} 3x_1 + x_2 + x_3 = 8, \\ -x_1 + 4x_2 - x_3 = 9, \\ x_1 - 3x_2 + 5x_3 = 2, \end{cases}$$

siendo un sistema lineal, se resuelve por los métodos usuales (Gauss-Jordan) obteniendo la solución  $\mathbf{x}_* = (1, 3, 2)$ . Pero también podemos transformarlo en un problema de punto fijo mediante el método de Jacobi, que consiste en despejar una de las variables de cada ecuación:

$$\begin{cases} x_1 = (8 - x_2 - x_3)/3, \\ x_2 = (9 + x_1 + x_3)/4, \\ x_3 = (2 - x_1 + 3x_2)/5. \end{cases}$$

Si definimos  $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$  como

$$\phi(x_1, x_2, x_3) = ((8 - x_2 - x_3)/3, (9 + x_1 + x_3)/4, (2 - x_1 + 3x_2)/5)$$

podemos plantear el método iterativo  $\mathbf{x}_n = \phi(\mathbf{x}_{n-1})$ . En la Tabla 3.6 mostramos los resultados y el error  $|\epsilon_n| = |\mathbf{x}_* - \mathbf{x}_n|$ . Igual que en el caso escalar, la convergencia no está asegurada.

$n$	$\mathbf{x}_n$	$ \epsilon_n $	$ \epsilon_n / \epsilon_{n-1} $
0	(0.00000, 0.00000, 1.00000)	3.317	
1	(2.33333, 2.50000, 0.40000)	2.142	0.646
2	(1.70000, 2.93333, 1.43333)	0.903	0.421
3	(1.21111, 3.03333, 1.82000)	0.279	0.309
4	(1.04889, 3.00778, 1.97778)	$0.543 \times 10^{-1}$	0.194
5	(1.00481, 3.00667, 1.99489)	$0.968 \times 10^{-2}$	0.178
6	(0.99948, 2.99993, 2.00304)	$0.308 \times 10^{-2}$	0.318
7	(0.99901, 3.00063, 2.00006)	$0.117 \times 10^{-2}$	0.381
8	(0.99977, 2.99977, 2.00058)	$0.662 \times 10^{-3}$	0.564
$\vdots$	$\vdots$	$\vdots$	$\vdots$
33	1.00000 , 3.00000 , 2.00000)	$0.111 \times 10^{-11}$	0.455
34	1.00000 , 3.00000 , 2.00000)	$0.506 \times 10^{-12}$	0.455

Tabla 3.6: Iteraciones de  $\mathbf{x}_n = \phi(\mathbf{x}_{n-1})$  con  $\mathbf{x}_0 = (0., 0., 1.)$ .

Otras formas de transformar el sistema original en un problema de punto fijo pueden dar lugar

a métodos no convergentes. Queremos obtener un criterio para determinar esto: linealizando alrededor del punto fijo tenemos  $\phi(\mathbf{x}) = \mathbf{x}_* + \phi'(\mathbf{x}_*)(\mathbf{x} - \mathbf{x}_*) + O(|\mathbf{x} - \mathbf{x}_*|^2)$ , por lo tanto

$$\boldsymbol{\epsilon}_n = \mathbf{x}_* - \phi(\mathbf{x}_{n-1}) = -\phi'(\mathbf{x}_*)\boldsymbol{\epsilon}_{n-1} + O(|\boldsymbol{\epsilon}_{n-1}|^2),$$

donde  $\phi'(\mathbf{x})$  es la matriz jacobiana. El criterio de convergencia para puntos iniciales  $\mathbf{x}_0$  cercanos a  $\mathbf{x}_*$  consiste en ver que el radio espectral  $\varrho(\mathbf{x}_*)$  de  $\phi'(\mathbf{x}_*)$  sea menor que 1. El radio espectral es el máximo de los módulos de los autovalores, es decir

$$\varrho(\mathbf{x}_*) = \max\{|\lambda| : \lambda \text{ autovalor de } \phi'(\mathbf{x}_*)\}.$$

En el caso anterior, los autovalores de la matriz

$$\phi'(\mathbf{x}_*) = \begin{pmatrix} 0 & -1/3 & -1/3 \\ 1/4 & 0 & 1/4 \\ -1/5 & 3/5 & 0 \end{pmatrix}$$

son  $\lambda_1 = -0.4546$ ,  $\lambda_{2,3} = 0.2273 \pm i0.1472$ , por lo tanto  $\varrho = |\lambda_1| = 0.4546 > 0.271 = |\lambda_{2,3}|$ . Observemos en la Tabla 3.6 que el comportamiento de los errores responde aproximadamente a la relación  $|\boldsymbol{\epsilon}_n| \cong \varrho(\mathbf{x}_*)|\boldsymbol{\epsilon}_{n-1}|$ .

Se podría pensar que el resolver un problema lineal mediante métodos iterativos es ineficiente dado que se puede hallar la solución exacta con métodos directos, como por ejemplo Gauss–Jordan. La primera observación es que, haciendo las operaciones en punto flotante, siempre habrá error en la solución. El número de operaciones que deben hacerse para resolver un sistema de ecuaciones por el método de Gauss–Jordan es  $O(d^3)$ , en cambio los métodos de punto fijo requieren  $n$  multiplicaciones de matriz por vector es decir  $nd^2$  las operaciones en punto flotante, que puede ser sustancialmente menor si  $n \ll d$ . Hay diferentes formas de transformar un sistema de ecuaciones lineales, entre los más conocidos tenemos el método de Jacobi y Gauss-Seidel.

¿Si la condición sobre el radio espectral se verifica en todo  $\mathbb{R}^n$ ,  $\varrho(\mathbf{x}) < 1$ , el método es incondicionalmente convergente?

**3.5. Método de la secante.** El método de la secante es similar al método de Newton, la diferencia consiste en que en lugar de usar la recta tangente por el punto donde estamos actualmente, tomamos la recta secante que pasa por los últimos dos puntos obtenidos por el método. Concretamente, si  $x_{n-2}$ ,  $x_{n-1}$  son los puntos obtenidos con el método en pasos sucesivos, planteamos el punto siguiente  $x_n$  como el resultado de resolver la ecuación:

$$\frac{x - x_{n-2}}{x_{n-1} - x_{n-2}}f(x_{n-1}) + \frac{x_{n-1} - x}{x_{n-1} - x_{n-2}}f(x_{n-2}) = 0,$$

que luego de despejar obtenemos

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

La implementación se muestra en el Algoritmo 3.3. Cada paso requiere una única evaluación de la función. El algoritmo se detiene en la primera iteración que da puntos muy cercanos entre sí o cuando se supera el número máximo de iteraciones.

**Algoritmo 3.3:** Método de la secante.

---

```

Data:  $f, x_0, x_1, \delta, N_{\max}$ 
Result:  $x, \text{code}$ 
 $n = 0;$ 
 $x_{\text{old}} = x_0;$ 
 $x_{\text{new}} = x_1;$ 
 $y_{\text{old}} = f(x_{\text{old}});$ 
 $y_{\text{new}} = f(x_{\text{new}});$ 
while  $|x_{\text{new}} - x_{\text{old}}| > \delta \wedge n \leq N_{\max}$  do
   $n = n + 1;$ 
   $x_{\text{aux}} = (x_{\text{old}} * y_{\text{new}} - x_{\text{new}} * y_{\text{old}}) / (y_{\text{new}} - y_{\text{old}});$ 
   $x_{\text{old}} = x_{\text{new}};$ 
   $y_{\text{old}} = y_{\text{new}};$ 
   $x_{\text{new}} = x_{\text{aux}};$ 
   $y_{\text{new}} = f(x_{\text{new}});$ 
end
if  $n \leq N_{\max}$  then
   $\text{code} = 0;$ 
else
   $\text{code} = 1;$ 
end
return  $x_{\text{new}}, \text{code};$ 

```

---

**Ejemplo 3.3.****3.5.1. Interpretación geométrica.****3.6. Aplicaciones.**

**3.6.1. Red de resistores.** Muchas ocasiones es útil considerar sistemas que involucran grandes números de resistencias en diversas geometrías, a veces de dimensión infinita. Hay muchas aplicaciones prácticas que utilizan este tipo de redes, por ejemplo, en la exploración geofísica, en la prospección de agua y petróleo. También en ingeniería es muy útil conocer el funcionamiento de redes, como los son las redes eléctricas o de distribución de agua o gas en una ciudad. Nuestro objetivo aquí es presentar una serie de modelos simples de redes que pueden ser estudiadas en el laboratorio. Consideramos una red larga de resistores como se muestra en la Figura 3.6.

Un sistema simple e interesante de analizar es el llamado red en escalera que se ilustra en la Figura 3.6. Esta red consiste en  $n$  mallas idénticas, consistente cada una de ellas en un par de resistores de valores  $r_i$  y  $r_m$ . Para calcular la resistencia equivalente de este sistema se puede usar el siguiente procedimiento recursivo. Llamaremos  $R_1$  a la resistencia equivalente de la primera malla,  $R_2$  a la de la siguiente y así sucesivamente.

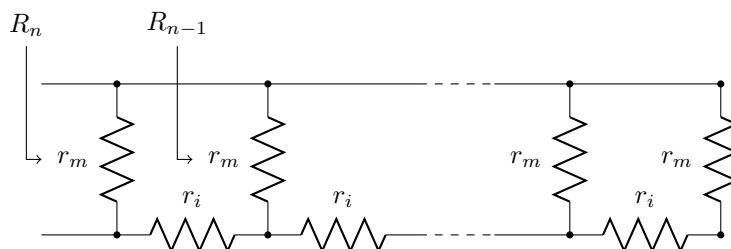


Fig. 3.6

La resistencia que se observa desde cada elemento de la red se relaciona por la ecuación

$$R_n = \phi(R_{n-1}) = \frac{r_m(r_i + R_{n-1})}{r_m + r_i + R_{n-1}},$$

con  $R_0 = r_m$ . Los únicos punto fijos de  $\phi$  son  $R = r_i \left( -1/2 \pm \sqrt{1/4 + \mu} \right)$ , donde  $\mu = r_m/r_i$ . Pero como las resistencias son positivas, el único punto fijo con significado físico es

$$R_* = r_i \left( -\frac{1}{2} + \sqrt{\frac{1}{4} + \mu} \right).$$

Derivando obtenemos

$$\phi'(R) = \frac{4\mu^2}{(2\mu + \sqrt{4\mu + 1} + 1)^2},$$

que verifica  $0 < \phi'(R) < 1$ , para  $R > 0$ , por lo tanto  $r_n \rightarrow R_*$  cuando  $n \rightarrow \infty$ .

