

Algoritmo de Halley

Thomás Rivera
Nicolás Puerto

February 2021

1 ¿Cuáles son condiciones para aplicar el método?

Podemos encontrar dos condiciones para que el método se pueda cumplir. La primera de ellas es que $f(x)$ debe ser de variable real (dominio los reales) y la segunda es que $f(x)$ debe tener una segunda derivada continua.

2 Proporcione una explicación geométrica del algoritmo

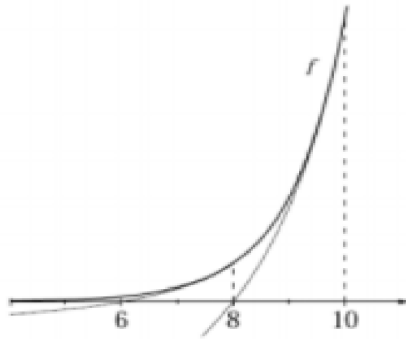


Figure 1: Explicación geométrica

También llamado el método de las hipérbolas tangentes. Consiste en la construcción de una hipérbola tangente a la gráfica de $f(x)$ en el punto X_0 próximo a la raíz de la función, que utiliza la función $f(x)$ en cada una de sus iteraciones, para acercarse cada vez más al valor buscado. Cuanto mayor sea el número de estimaciones, mayor precisión en el resultado.

3 Realice un diagrama de flujo que muestre como se debe operar el algoritmo

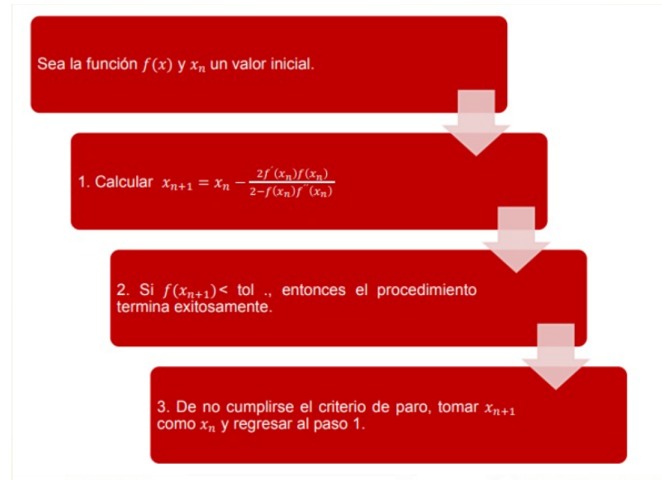


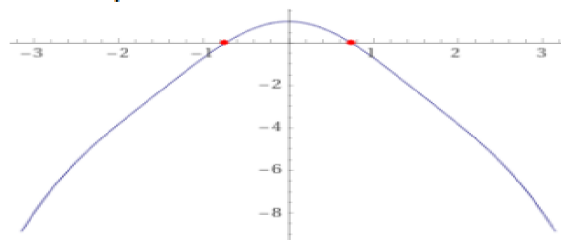
Figure 2: Diagrama de flujo

4 ¿Cuáles son las raíces?

4.1 $\cos 2(x) - x^2 = 0$

En el caso de la primera función (figura 3 y 4) encontramos gran cercanía al valor esperado dado por wolfram, con tan solo 4 iteraciones en el primer caso y 5 para el resto, es importante aclarar que utilizamos 56 dígitos en todas funciones para todos sus diferentes valores de tolerancia

WolframAlpha



$$x \approx 0.73908513321516064166$$

$$x \approx -0.73908513321516064166$$

Figure 3: Gráfica 1

A continuación el código de la función y sus resultados en R

Código en R

```

1 #Metodo de Halley
2 i=1
3 VI <- mpfr(0.71875,56)
4 n=100
5 tolerancia=10^-8
6
7
8 f <- function(x){
9   cos(x)^2-x^2
10 }
11 dx <- function(x){
12   -(2*(sin(x)*cos(x))+2*x)
13 }
14 dxx <- function(x){
15   -(2*(cos(x)*cos(x)-sin(x)*sin(x))+2)
16 }
17
18 while(i<=n){
19   VS = VI - (f(VI)/dx(VI))*(1 - (f(VI)*dxx(VI)/dx(VI)^2))^(-1)
20   if(abs(VS-VI)/abs(VS) < tolerancia){
21     cat("Iteracion =",i,"\n Raiz = ")
22     print(VS,56)
23     cat("Tolerancia =",tolerancia,"\n")
24     break
25   }
26   i=i+1
27   VI = VS
28 }
29

```

Mayor # iteraciones, mayor precisión

Resultados con diferentes tolerancias

```

Iteracion = 4
Raiz = 1 'mpfr' number of precision 56 bits
[1] 0.7390851332151606445375335852077114395797252655029296875
Tolerancia = 1e-08

Iteracion = 5
Raiz = 1 'mpfr' number of precision 56 bits
[1] 0.7390851332151606445375335852077114395797252655029296875
Tolerancia = 1e-16

Iteracion = 5
Raiz = 1 'mpfr' number of precision 56 bits
[1] 0.7390851332151606445375335852077114395797252655029296875
Tolerancia = 1e-32

Iteracion = 5
Raiz = 1 'mpfr' number of precision 56 bits
[1] 0.7390851332151606445375335852077114395797252655029296875
Tolerancia = 1e-56

```

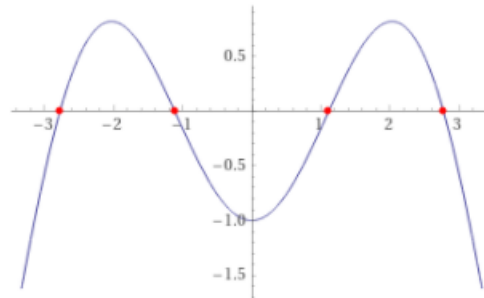
Difiere con WolframAlpha a partir de la cifra 18

Figure 4: Código 1

4.2 $x\sin(x)-1=0$ de $[-1,2]$

En el caso de la segunda función (figura 5 y 6) encontramos gran cercanía al valor esperado y dado por wolfram, con tan solo 3 iteraciones en el primer valor de tolerancia y 4 para el resto de sus tolerancias, siempre manejando todo con 56 dígitos

WolframAlpha



$$x \approx \pm 9.31724294141480961860128851\dots$$

$$x \approx \pm 6.43911723841724646172451484\dots$$

$$x \approx \pm 2.77260470826599123395356972\dots$$

$$x \approx \pm 1.11415714087193008730052518\dots$$

Por lo tanto se toma solo la última raíz positiva

Figure 5: Gráfica 2

A continuación el código de la función y sus resultados en R

Código en R

```
2 #Metodo de Halley
3 i=1
4 VI <- mpfr(1.09375,56)
5 n=100
6 tolerancia=10^-8
7
8 f <- function(x){
9   x*sin(x)-1
10 }
11 dx <- function(x){
12   sin(x)+x*cos(x)
13 }
14 dxx <- function(x){
15   cos(x)+(cos(x)-x*sin(x))
16 }
17
18 while(i<=n){
19   VS = VI - (f(VI)/dx(VI))*(1 - (f(VI)*dxx(VI)/dx(VI)^2))^-1
20   if(abs(VS-VI)/abs(VS) < tolerancia){
21     cat("Iteración =",i,"\n Raíz = ")
22     print(VS,56)
23     cat("Tolerancia =",tolerancia,"\n")
24     break
25   }
26   i=i+1
27   VI = VS
28 }
```

Resultados con diferentes tolerancias

```
Iteración = 3
Raíz = 1 'mpfr' number of precision 56 bits
[1] 1.1141571408719301017331559933154494501650333404541015625
Tolerancia = 1e-08

Iteración = 4
Raíz = 1 'mpfr' number of precision 56 bits
[1] 1.11415714087193007397758037768653593957424163818359375
Tolerancia = 1e-16

Iteración = 4
Raíz = 1 'mpfr' number of precision 56 bits
[1] 1.11415714087193007397758037768653593957424163818359375
Tolerancia = 1e-32

Iteración = 4
Raíz = 1 'mpfr' number of precision 56 bits
[1] 1.11415714087193007397758037768653593957424163818359375
Tolerancia = 1e-56
```

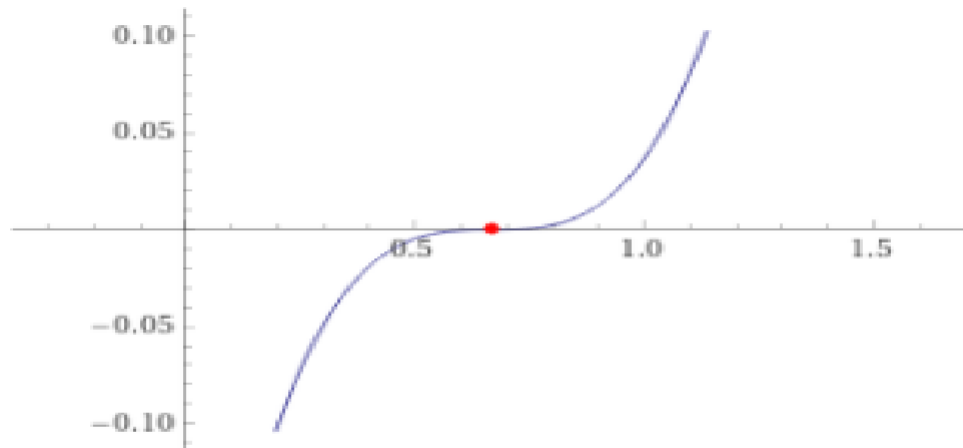
Con una tolerancia de -16 y mayores, la raíz es exacta

Figure 6: Código 2

4.3 $x^3 - 2x^2 + 43x - 827 = 0$

En el caso de la tercera función (figura 7 y 8) encontramos gran cercanía al valor esperado y dado por wolfram, con tan solo 1 iteración para todos los casos.

WolframAlpha



$$x \approx 0.6666666666666667$$

Figure 7: Gráfica 3

A continuación el código de la función y sus resultados en R. Hasta ahora es la función que más diferencia mantiene con el valor dado por WolframAlpha, a pesar del aumento de significancia en el resultado se mantuvo en todos los casos con 1 iteraciones

Código en R

```
1 #
2 #Metodo de Halley
3 i=1
4 vI<-mpfr(0.65625,56)
5 n=100
6 tolerancia=10^-8
7
8 f <- function(x){
9   x^3-2*x^2+(4/3)*x-(8/27)
10 }
11 dx <- function(x){
12   3*x^2-2*(2*x)+(4/3)
13 }
14 dxx <- function(x){
15   3*(2*x)-2*2
16 }
17
18 while(i<=n){
19   VS = vI - (f(vI)/dx(vI))*(1 - (f(vI)*dxx(vI)/dx(vI)^2))^(-1)
20   if(abs(VS-vI)/abs(VS)< tolerancia){
21     cat("Iteracion =",i,"\n Raiz = ")
22     print(VS,56)
23     cat("Tolerancia =",tolerancia,"\n")
24     break
25   }
26   i=i+1
27   vI = VS
28 }
29
```

Resultados con diferentes tolerancias

```
Iteracion = 7
Raiz = 1 'mpfr' number of precision 56 bits
[1] 0.6666700181668379077137842614320106804370880126953125
Tolerancia = 1e-08

Iteracion = 7
Raiz = 1 'mpfr' number of precision 56 bits
[1] 0.6666700181668379077137842614320106804370880126953125
Tolerancia = 1e-16

Iteracion = 7
Raiz = 1 'mpfr' number of precision 56 bits
[1] 0.6666700181668379077137842614320106804370880126953125
Tolerancia = 1e-32

Iteracion = 7
Raiz = 1 'mpfr' number of precision 56 bits
[1] 0.6666700181668379077137842614320106804370880126953125
Tolerancia = 1e-56
```

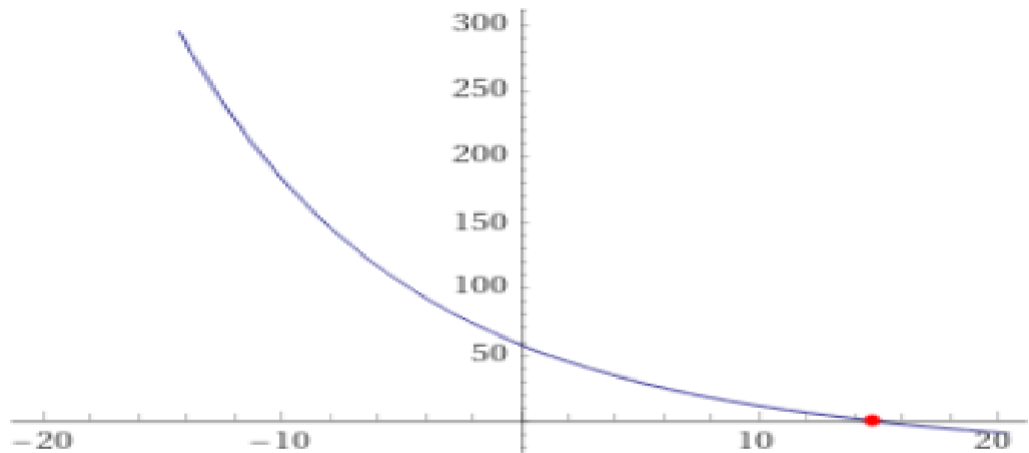
Hasta ahora es la función que más diferencia mantiene con el valor dado por WolframAlpha, a pesar del aumento de significancia en el resultado se mantuvo en todos los casos con #7 iteración, falla desde el 5 decimal

Figure 8: Código 3

4.4 $667.38/x(1-e^{*(-0.1468x)})-40=0$

En el caso de la cuarta función (figura 9 y 10) encontramos gran cercanía al valor esperado y dado por wolfram, con tan solo 11 iteraciones en la primera tolerancia y 29 para casos más grandes e inclusive llega a 30.

WolframAlpha



$x \approx 14.7785276833667033600660381\dots$

Figure 9: Gráfica 4

A continuación el código de la función y sus resultados en R

Código en R

```
1 #Metodo de Halley
2 i=1
3 VI<-mpfr(14.78125,56)
4 n=100
5 tolerancia=10^-8
6
7
8 f <- function(x){
9   ((667.38/x)*(1-exp(-0.1468*x)))-40
10 }
11 dx <- function(x){
12   (0.1468*exp(-0.1468*x))-(667.38/x^2)
13 }
14 dxx <- function(x){
15   (-0.02155024*exp(-0.1468*x))+(1334.76/x^3)
16 }
17
18 while(i<=n){
19   VS = VI - (f(VI)/dx(VI))*(1 - (f(VI)*dxx(VI)/dx(VI)^2))^(-1)
20   if(abs(VS-VI)/abs(VS) < tolerancia){
21     cat("Iteracion =",i,"\n Raiz = ")
22     print(VS,56)
23     cat("Tolerancia =",tolerancia,"\n")
24     break
25   }
26   i=i+1
27   VI = VS
28 }
29
```

Resultados con diferentes tolerancias

```
Iteracion = 11
Raiz = 1 'mpfr' number of precision 56 bits
[1] 14.7785277176395339893133495934307575225830078125
Tolerancia = 1e-08

Iteracion = 29
Raiz = 1 'mpfr' number of precision 56 bits
[1] 14.7785276833667034157571151808951981365680694580078125
Tolerancia = 1e-16

Iteracion = 30
Raiz = 1 'mpfr' number of precision 56 bits
[1] 14.7785276833667034157571151808951981365680694580078125
Tolerancia = 1e-32

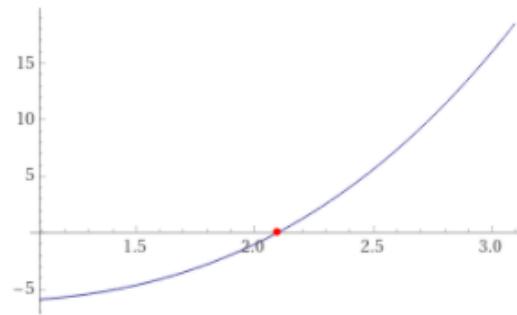
Iteracion = 30
Raiz = 1 'mpfr' number of precision 56 bits
[1] 14.7785276833667034157571151808951981365680694580078125
Tolerancia = 1e-56
```

Difiere con WolframAlpha en la cifra 17

Figure 10: Código 4

4.5 $x^3 - 2x - 5 = 0$

En el caso de la quinta función (figura 11 y 12) encontramos gran cercanía al valor esperado y dado por wolfram, con tan solo 3 iteraciones y 4 para casos más grandes finalizando con 5.



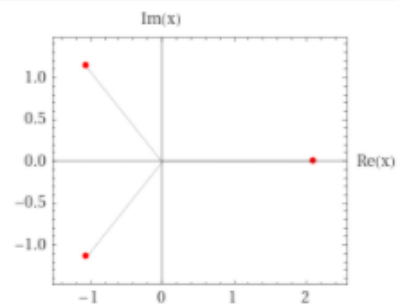
$$x \approx 2.09455148154233$$

Complex solutions:

$$x \approx -1.0473 - 1.1359 i$$

$$x \approx -1.0473 + 1.1359 i$$

Roots in the complex plane:



Se tomará sólo la raíz real

Figure 11: Gráfica 5

A continuación el código de la función y sus resultados en R

Código en R

```
2 #Metodo de Halley
3 i=1
4 VI<-mpfr(2.09375,56)
5 n=100
6 tolerancia=10^-8
7
8 f <- function(x){
9   x^3-2*x-5
10 }
11 dx <- function(x){
12   3*x^2 -2
13 }
14 dxx <- function(x){
15   3*(2*x)
16 }
17
18 while(i<=n){
19   VS = VI - (f(VI)/dx(VI))*(1 - (f(VI)*dxx(VI)/dx(VI)^2))^(-1)
20   if(abs(VS-VI)/abs(VS) < tolerancia){
21     cat("Iteracion =",i,"\n Raiz = ")
22     print(VS,56)
23     cat("Tolerancia =",tolerancia,"\n")
24     break
25   }
26   i=i+1
27   VI = VS
28 }
29
```

Resultados con diferentes tolerancias

```
Iteracion = 3
Raiz = 1 'mpfr' number of precision 56 bits
[1] 2.094551481542326565321587850121431984007358551025390625
Tolerancia = 1e-08

Iteracion = 4
Raiz = 1 'mpfr' number of precision 56 bits
[1] 2.09455148154232662083273908137925900518894195556640625
Tolerancia = 1e-16

Iteracion = 5
Raiz = 1 'mpfr' number of precision 56 bits
[1] 2.09455148154232662083273908137925900518894195556640625
Tolerancia = 1e-32

Iteracion = 5
Raiz = 1 'mpfr' number of precision 56 bits
[1] 2.09455148154232662083273908137925900518894195556640625
Tolerancia = 1e-56
```

Difiere con WolframAlpha a partir del dígito 14

Figure 12: Código 5

5 ¿Cómo se comporta el método en cuanto: pérdida de significancia, el número de iteraciones, la convergencia, en cada caso?

Problema 1: El algoritmo cumple con su deber y calcula de forma eficiente la raíz de la función. Le toma en principio 3 iteraciones y al aumentar la tolerancia llega a un máximo de 5 iteraciones sin diferir mucho del resultado dado por WolframAlpha. En cuanto a la convergencia halley tiene convergencia cúbica y eso facilita el proceso.

Problema 2: Al método le toma pocas iteraciones llegar a la raíz de la función, teniendo como máximo 3 y una pérdida de significancia mínima, lo que permitió encontrar la raíz de manera exacta comparada con el resultado de WolframAlpha. En la tolerancia 10-16 y en adelante la convergencia se mantuvo constante, con un mismo número de iteraciones.

Problema 3: Hasta ahora es la función que más diferencia mantiene con el valor dado por WolframAlpha, a pesar del aumento de significancia, el resultado se mantuvo en todos los casos con 1 iteración y con convergencia constante.

Problema 4: Es el caso donde mayor número de iteraciones le toma al método encontrar la raíz, debido a la naturaleza intrínseca de la función. El resultado varía respecto WolframAlpha en la última cifra, con un máximo de 29 iteraciones.

Problema 5: Al igual que el primer caso el método logra hallar la raíz de la función con pocas iteraciones, con menos de 5, el resultado difiere en los últimos 3 decimales y deja la convergencia constante con una tolerancia de 10-16 en adelante.

6 ¿Cómo se puede solucionar el problema de significancia, es remediable o está destinado al fracaso, en los casos que se presente el problema?

Si es remediable y en ese mismo orden de ideas se podría utilizar una precisión extendida para obtener números con 15 o 16 dígitos decimales más de precisión, ya que el método no es muy complejo y así mejorar los resultados. Para ello se emplea la librería Rmpfr para aumentar la cantidad de dígitos decimales obtenidos (utilizamos 56), sin embargo a pesar de ello, las raíces no llegan a ser completamente exactas.

Por el contrario, no consideramos que el algoritmo esté destinado al fracaso ya que, es de los mejores algoritmos que se presentan en la clase; es el que menos iteraciones presenta y en ese orden nos da una solución mucho más rápida, más precisa y "inalcanzable" para los demás algoritmos que en los mismo valores llegaban a tener 50 iteraciones, 30 o 20.

7 ¿Qué pasa con el método cuando la función es periódica, par o impar, estas características influyen?

El método no contempla estas situaciones a menos de que la función no cumpla con los requisitos, si no es así pues la desecha. Sin embargo, si cumple con los requisitos evalúa la función debido a que busca la raíz más cercana al valor inicial indicado. Por lo tanto solo encuentra una raíz

En conclusión, desde que la función cumpla con los requisitos mínimos expuestos anteriormente, no importa si es periódica, par o impar, el algoritmo irá a hallar la raíz.

8 Realice una gráfica que muestre cómo se comporta el método en cada caso con respecto a la tolerancia y al número de iteraciones

En la siguiente tabla encontramos los números de iteraciones presentados para cada función teniendo en cuenta la tolerancia

Podemos observar que los números de las iteraciones son demasiado estables, lo cual no representa un problema a pesar de que estamos cambiando bastante los valores de la tolerancia. Además también podemos inferir que la cuarta función demanda muchas más iteraciones debido a que es más compleja de lo normal puesto que maneja Euler elevado a ciertos valores.

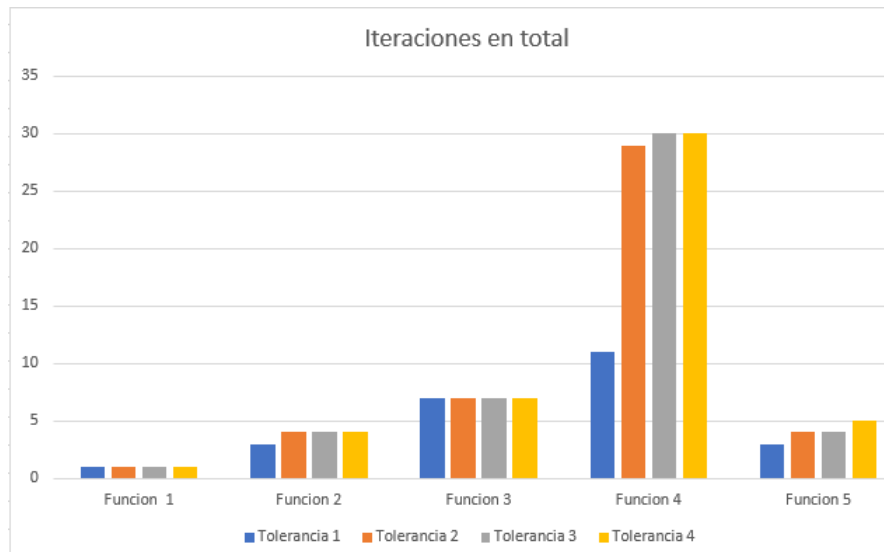


Figure 13: Iteraciones vs ToleranciaPorFuncion

9 ¿Cómo se comporta el método con respecto a la solución con Taylor?

mientras más cercano de 0 están los valores, menor es la diferencia entre Taylor y Halley. En ese mismo de idea el algoritmo tiende a comportarse como el algoritmo de Taylor, teniendo un número de iteraciones prácticamente igual y tendiendo a tener la misma precisión para los valores de las tolerancias. Por el contrario, entre más alejados están los valores, podemos encontrar una mayor diferencia entre Taylor y Halley donde el algoritmo de Halley presenta mejores resultados en cuanto a exactitud y velocidad.

A continuación una gráfica de Taylor vs Halley

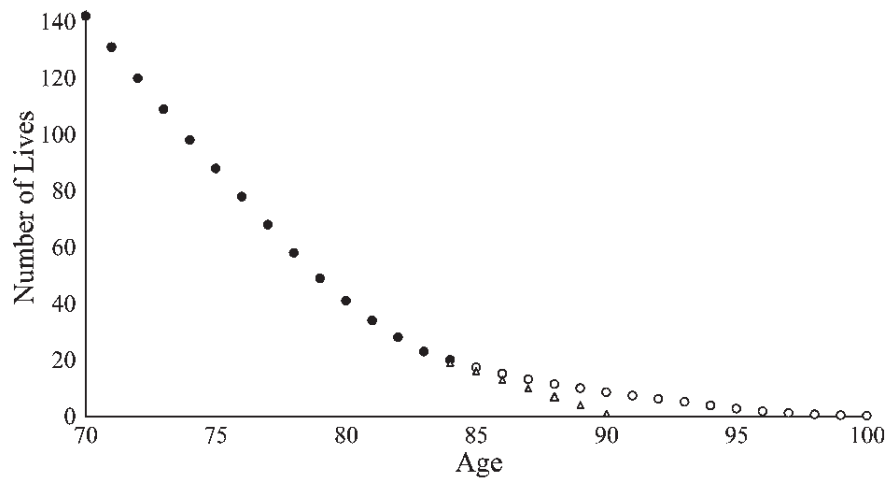


Fig. 2. Jones's (○) and Taylor's (Δ) extension to Halley's (●) life table

Figure 14: taylor vs Halley

10 ¿Cómo se comporta el método con respecto a la solución con Taylor?

Halley llega a la respuesta con menos interacciones gracias a su orden de convergencia 3, eso le permite acercarse al valor de la raíz con mayor rapidez. Bisección con convergencia línea debe hacer muchas particiones del rango otorgado para poder llegar a la respuesta