

Pontificia Universidad Javeriana

INGENIERÍA DE SISTEMAS

ANÁLISIS NUMÉRICO

Primer taller

David Anteliz
Johanna Bolívar
Abril Cano
Richard Fonseca

Agosto de 2021

1. Introduccion

El método de Halley desarrollado por el astrónomo Edmond Halley, es un algoritmo numérico iterativo que permite obtener las raíces de una ecuación no lineal. Este método permite llegar al resultado

23 warnings con un número bajo de iteraciones y con un consumo computacional reducido, lo que lo hace una solución eficiente.

2. Preguntas

2.1. Cuales son condiciones para aplicar el método y señalar donde quedan implementadas

Se pueden encontrar 3 condiciones para poder implementar el metodo. Primero $f(x)$ debe estar dentro el dominio de los numeros reales, segundo $f(x)$ debe tener una derivada continua y tercero $f(x)$ debe tener una segunda derivada continua.

2.2. Proporcione una explicación geométrica del algoritmo

El método de Halley puede obtenerse aproximando una función por medio de hipérbolas (Ezquerro et al., 2001). A partir de un valor inicial x_0 se busca una hipérbola de la forma: Este tipo de curvas, denominadas osculatrices, coinciden con f en x_0 hasta la segunda derivada. En consecuencia, estas hipérbolas se aproximan mejor a una curva que la recta tangente con su coincidencia sólo hasta la primera derivada. Para que la hipérbola cumpla las condiciones requeridas, sus coeficientes a , b y c deben satisfacer que: Resolviendo el este sistema de ecuaciones, se obtiene que: Una vez encontrada la hipérbola osculatriz, su intersección con el eje de abscisas en el punto en que $y(x_1) = 0$ se toma como nueva aproximación. Dada la forma de la hipérbola, se tiene que: A partir de esta interpretación geométrica, el método de Halley también se conoce como el método de las hipérbolas tangentes.

2.3. DIAGRAMA DE FLUJO DEL ALGORITMO DE HALLEY

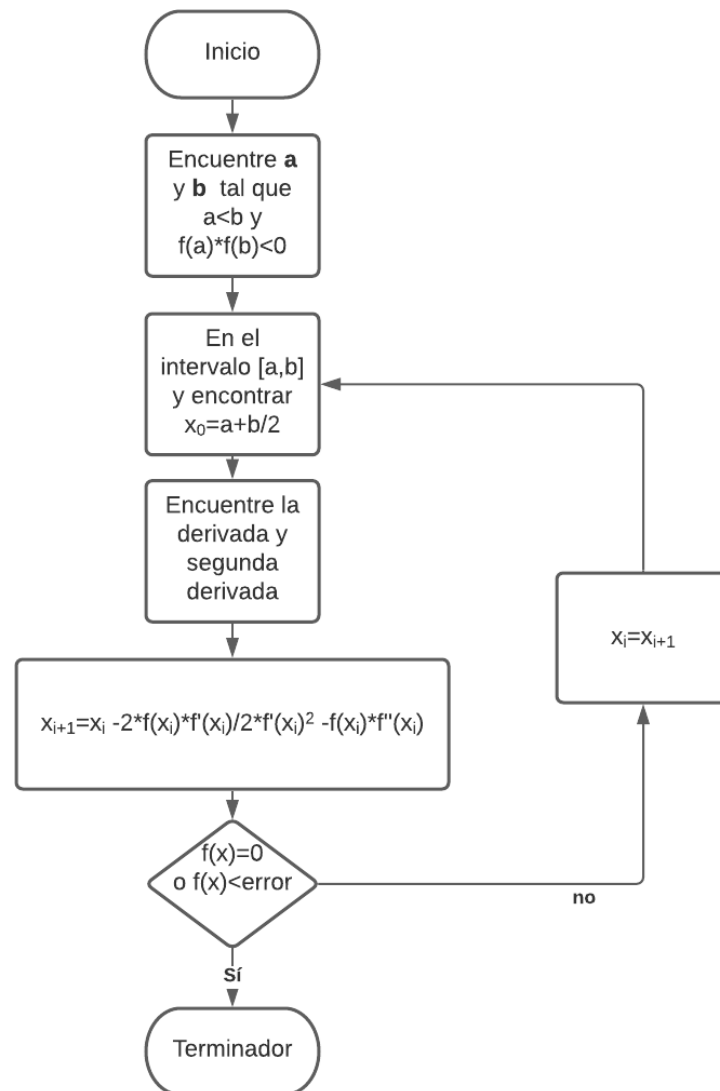


Figura 1: diagrama de flujo

2.4. Aplicar el método y determinar la solución incluyendo la validación del resultado

2.4.1. $f(x) = \cos^2(x) - x^2$

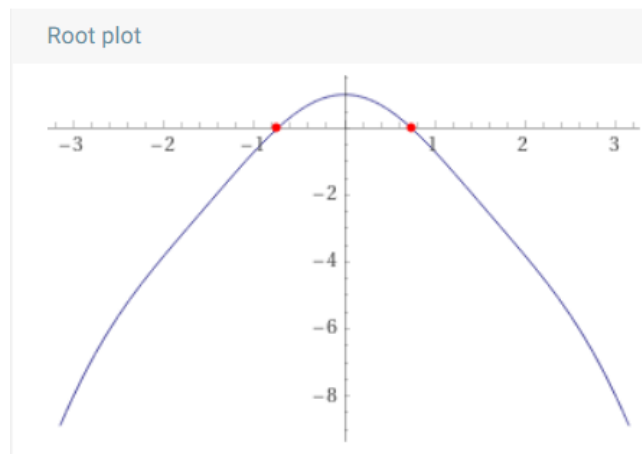


Figura 2: Grafica wolfram

Raíces:

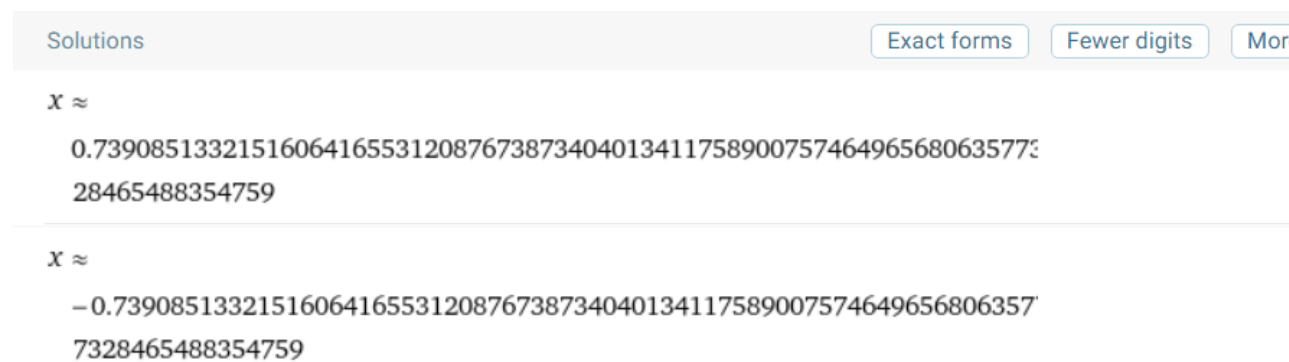


Figura 3: Raíces wolfram

Caso Tolerancia 10^{-8}

su exactitud es la esperada de 10^{-8} empesando a fallar despues de: 0,7390851332151606
donde la respuesta es:
0,73908513321516064165531208767387340401341175890075746496568063577328465488354759
y la aproximacion da:
0,7390851332151606722931092008366249501705169677734375000000000000000000000000



Su exactitud es la esperada de 10^{-16} empesando a fallar despues de: 0,7390851332151606
donde la respuesta es:
0,73908513321516064165531208767387340401341175890075746496568063577328465488354759
y la aproximacion da:
0,739085133215160672293109200836624950170516967773437500000000000000000000000000
Lo que notamos fue que aunque se ejecuto una iteracion más es el mismo resulta-
do



Su exactitud ya no es la esperada de 10^{-32} empesando a fallar despues de:
0,7390851332151606 donde la respuesta es:
0,73908513321516064165531208767387340401341175890075746496568063577328465488354759
y la aproximacion da:
0,739085133215160672293109200836624950170516967773437500000000000000000000000000
en adelante el resultado se mantiene igual en termino de iteraciones, donde sigue
diferiendo por la posicion decimal 17 **Caso Tolerancia** 10^{-56}



Figura 7: Resultado código

[illegible]

2.4.2. $2.f(x) = e^x - x - 1$

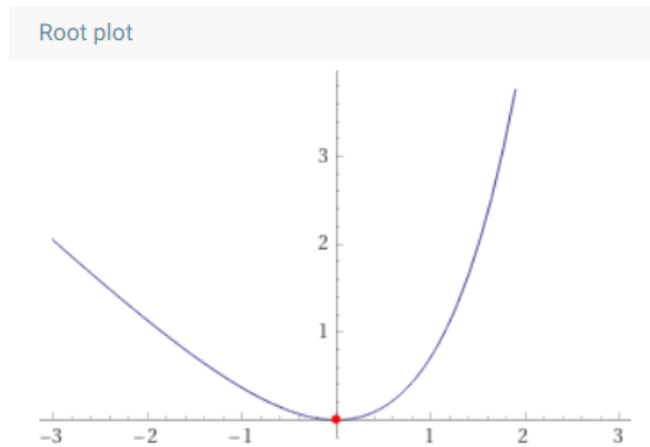


Figura 8: Grafica wolfram

Raices:

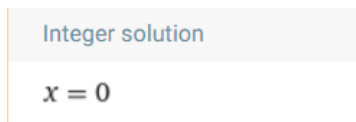


Figura 9: Raices wolfram

Caso Tolerancia 10^{-8}

```

↳ Eps maq: 1.084202172485504434e-19
Raiz: 0.00000000422575949780355541182979398659273762017818398817325942218303680419921875
Tolerancia 1e-08
Numero de iteraciones: 17

```

Figura 10: Resultado codigo

Su exactitud es la esperada de 10^{-8} empesando a fallar despues de: 0,00000000
donde la respuesta es:
0
y la aproximacion da:
0,00000000422575949780355541182979398659273762017818398817325942218303680419921875
Difiere en la 9 posicion decimal
Caso Tolerancia 10^{-16}

```

Eps maq: 1.084202172485504434e-19
Raiz: 0.00000000026415168745406483360321414918879100869331111312021675985306501388549805
Tolerancia 1e-16
Numero de iteraciones: 21

```

Figura 11: Resultado codigo

Su exactitud no es la esperada de 10^{-16} fallando por 0,000000000 sabiendo
que la respuesta es:
0
y la aproximacion da:
0,00000000026415168745406483360321414918879100869331111312021675985306501388549805
Difiriendo en la 10 posicion decimal, donde aqui en adelante el resultado es igual
como tambien su cantidad de iteraciones. **Caso Tolerancia 10^{-32}**

```

↳ Eps maq: 1.084202172485504434e-19
Raiz: 0.00000000026415168745406483360321414918879100869331111312021675985306501388549805
Tolerancia 1e-32
Numero de iteraciones: 21

```

Figura 12: Resultado codigo

Su exactitud no es la esperada de 10^{-32} fallando por 0,000000000 sabiendo
que la respuesta es:
0
y la aproximacion da:
0,00000000026415168745406483360321414918879100869331111312021675985306501388549805
Caso Tolerancia 10^{-56}

```

Eps maq: 1.084202172485504434e-19
Raiz: 0.00000000026415168745406483360321414918879100869331111312
Tolerancia 1e-56
Numero de iteraciones: 21

```

Figura 13: Resultado codigo

Su exactitud no es la esperada de 10^{-56} fallando por 0,000000000 sabiendo que la respuesta es: 0 y la aproximacion da: 0,00000000026415168745406483360321414918879100869331111312 de aqui en adelante se imprime hasta la posicion decimal 56,por comodidad

2.4.3. $3..f(x) = x^3 - 2x^2 + 4/3x - 8/27$ y $x^3 - 2x + 5 = 0$

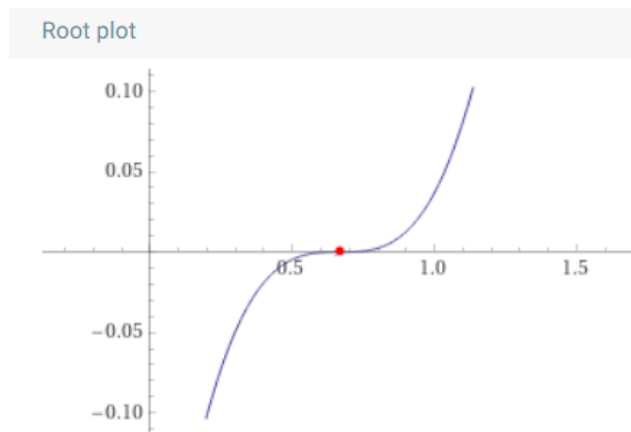


Figura 14: Grafica wolfram

Raices:

Solution

$$x = \frac{2}{3}$$

Figura 15: Raices wolfram

$\frac{2}{3}$ da una respuesta periodica de 0,6666...
Caso Tolerancia 10^{-8}


```
Eps maq: 1.084202172485504434e-19
Raiz: 0.66666987116972120652746980340452864766120910644531250000
Tolerancia 1e-08
Numero de iteraciones: 38
```

Figura 16: Resultado código

Su exactitud no es la esperada de 10^{-8} fallando despues de 0,66666 donde la repuesta es:
0,66666666666666666666666666666666...
y la proximación: 0,66666987116972120652746980340452864766120910644531250000
Difiere en la 6 posicion decimal **Caso Tolerancia** 10^{-16}

```
Eps maq: 1.084202172485504434e-19
Raiz: 0.66666987116972120652746980340452864766120910644531250000
Tolerancia 1e-16
Numeros de iteraciones: 39
```

Figura 17: Resultado código

Su exactitud no es la esperada de 10^{-16} fallando despues de 0,66666 donde la repuesta es:
 0,666666666666666666666666666666666666...
 y la proximación: 0,66666987116972120652746980340452864766120910644531250000
 Aunque se mantiene el mismo resultado si se aumenta una iteracion **Caso Tolerancia** 10^{-32}

```

Eps maq: 1.084202172485504434e-19
Raiz: 0.66666987116972120652746980340452864766120910644531250000
Tolerancia 1e-32
Numero de iteraciones: 39

```

Figura 18: Resultado código

Su exactitud no es la esperada de 10^{-32} fallando despues de 0,66666 donde la repuesta es:
0,66666666666666666666666666666666...
y la proximación: 0,66666987116972120652746980340452864766120910644531250000
Ya no hay cambios en la cantidad de iteraciones **Caso Tolerancia** 10^{-56}

```

↳ Eps maq: 1.084202172485504434e-19
Raiz: 0.66666987116972120652746980340452864766120910644531250000
Tolerancia 1e-56
Numero de iteraciones: 39

```

Figura 19: Resultado código

Su exactitud no es la esperada de 10^{-56} fallando despues de 0,66666 donde la repuesta es:
0,666...
y la proximación: 0,66666987116972120652746980340452864766120910644531250000

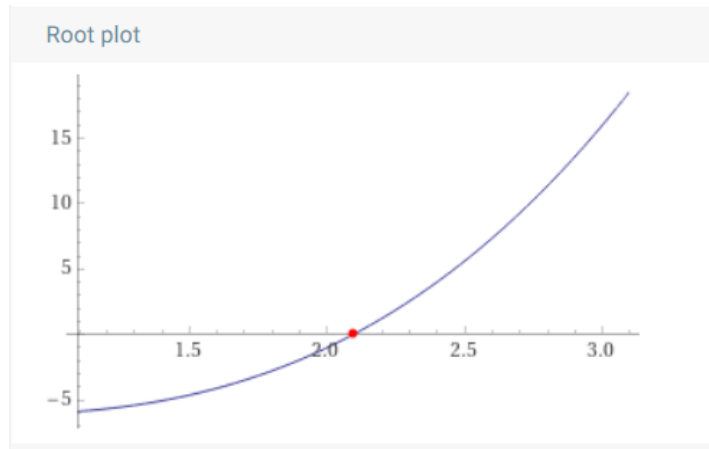


Figura 20: Grafica wolfram

Raices:

Real solution

$$x = \frac{1}{3} \sqrt[3]{\frac{135}{2} - \frac{3\sqrt{1929}}{2}} + \frac{\sqrt[3]{\frac{1}{2}(45 + \sqrt{1929})}}{3^{2/3}}$$

Figura 21: Raices wolfram

Es periodica, pero no regular

Real solution Fewer digits More digits E

$x \approx 2.094551481542326591482386540579302963857306105628239180$

Figura 22: Raices wolfram

Caso Tolerancia 10^{-8}

```

Eps maq: 1.084202172485504434e-19
Raiz: 2.09455148154232650981043661886360496282577514648437500000
Tolerancia 1e-08
Numero de iteraciones: 10

```

Figura 23: Resultado codigo

Su exactitud es la esperada de 10^{-8} fallando despues de 2,0945514815423265 donde la repuesta es : 2,094551481542326591482386540579302963857306105628239180 y la aproximacion: 2,09455148154232650981043661886360496282577514648437500000 fallando por la posicion decimal 17

Caso Tolerancia 10^{-16}

```

Eps maq: 1.084202172485504434e-19
Raiz: 2.09455148154232650981043661886360496282577514648437500000
Tolerancia 1e-16
Numero de iteraciones: 10

```

Figura 24: Resultado codigo

Su exactitud es la esperada de 10^{-16} fallando despues de 2,0945514815423265 donde la repuesta es: 2,094551481542326591482386540579302963857306105628239180 y la aproximacion: 2,09455148154232650981043661886360496282577514648437500000 fallando por la posicion decimal 17 manteniendo la cantidad de iteraciones **Caso Tolerancia 10^{-32}** Pudimos notar que de este momento se demoraba bastante en completar el proceso el codigo, quedando e bucle infinito esto por el epsilon.

Caso Tolerancia 10^{-56} Sucede lo mismo que el caso de tolerancia 10^{-32}

2.4.4. Determinar coeficiente de arrastre W

Input interpretation

$$\frac{667.38(1 - e^{-0.146843x})}{x} - 40 = 0$$

Figura 25: Ecuacion llegada

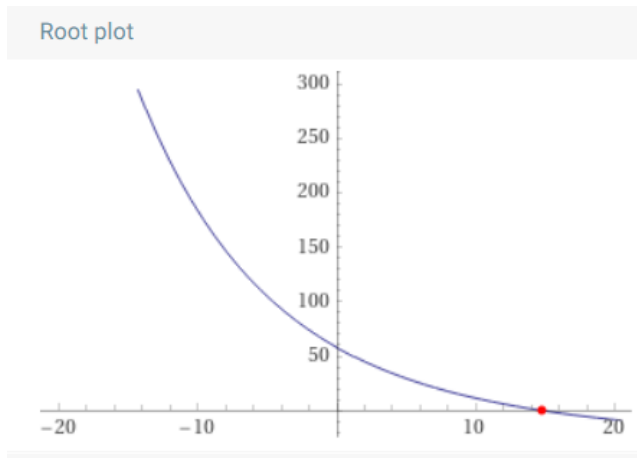


Figura 26: Grafica wolfram

Raices:



Figura 27: Grafica wolfram

Es periodica, pero no regular

2.4.5. $f(t) = 3\sin^3(t); g(t) = 4\sin(t)\cos(t) ; t \geq 0$

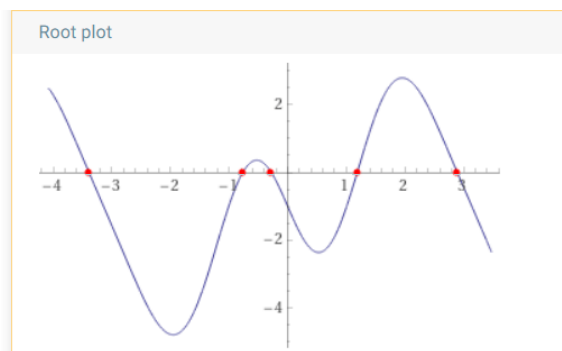


Figura 28: Resultado codigo

Raices:

Solutions
$x \approx 2(3.14159n - 0.382821), \quad n \in \mathbb{Z}$
$x \approx 2(3.14159n - 0.140173), \quad n \in \mathbb{Z}$
$x \approx 2(3.14159n + 0.593248), \quad n \in \mathbb{Z}$
$x \approx 2(3.14159n + 1.44633), \quad n \in \mathbb{Z}$
$x \approx 2(3.14159n - (1.54369 + 0.622335i)), \quad n \in \mathbb{Z}$

Figura 29: Resultado codigo

Caso Tolerancia 10^{-8}

```

↳ Eps maq: 1.084202172485504434e-19
Raiz: 0.73908513321516067229310920083662495017051696777343750000
Tolerancia 1e-08
Numero de iteraciones: 4

```

Figura 30: Resultado codigo

Caso Tolerancia 10^{-16}

```

↳ Eps maq: 1.084202172485504434e-19
Raiz: 0.73908513321516067229310920083662495017051696777343750000
Tolerancia 1e-16
Numero de iteraciones: 5

```

Figura 31: Resultado codigo

Caso Tolerancia 10^{-32}

```

↳ Eps maq: 1.084202172485504434e-19
Raiz: 0.73908513321516067229310920083662495017051696777343750000
Tolerancia 1e-32
Numero de iteraciones: 5

```

Figura 32: Resultado codigo

Caso Tolerancia 10^{-56}

```

✕ Eps maq: 1.084202172485504434e-19
Raiz: 0.73908513321516067229310920083662495017051696777343750000
Tolerancia 1e-56
Numero de iteraciones: 5

```

Figura 33: Resultado codigo

2.5. Evaluar la relación numero de iteraciones versus tolerancia deseada

En cuanto mayor sea el tamaño de la tolerancia, mayor cantidad de iteraciones. Sabemos que al aumentar la tolerancia se es mas exacto y cercano al valor real, por ende se puede asumir lo mismo de las iteraciones. Nuestro numero de iteraciones suele llegar a un limite pero esto es por el eps de la maquina

2.6. Teniendo en cuenta los resultados anteriores, cómo se puede solucionar el problema de significancia, esta destinado al fracaso?

El problema si tiene remedio con librerias permitiendo una mayor cantidad de digitos, dando a que el problema este mitigado. En el tema de si esta o no destinado a fracasar, nunca se podra ser exacto, pero se podra llegar a ser tan cercano que no afecta, pudiendo decirse que no lo concideramos destinado al fracaso

Se puede solucionar mediante la utilización de una precisión extendida para obtener números con mas cifras decimales (56 aprox.) de precisión, esto, ayudaría a mejorar notablemente los resultados, sin embargo, a pesar de ello, la solución a las raices no llegan a ser completamente exactas.

Este algoritmo no estaria destinado al fracaso, ya que, es uno de los que menos iteraciones presenta y nos da una solución mucho mas rápida, eficaz y precisa.

2.7. Que pasa con el método cuando hay más de dos raíces

La funcion busca una unica raiz que es más cercana a un valor ingresado en nuestro caso $x = 0.5$, entonces solo obtiene una raiz.

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

El metodo no tiene contempladas estas situaciones, solo confirma si cumple los condiciones anteriormente planteadas.Desde que la funcion cumpla las condiciones no importa si la funcion es periodica, par o impar, se encuentra una de las raices disponibles

2.9. Gráfica que muestre cómo se comporta el método en cada caso con respecto a la tolerancia y al número de iteraciones

2.10. Comparando con el método de bisección

Mediante la comparación del método Halley con el método de bisección se puede concluir que Halley llega a la respuesta con menos iteraciones debido a su

Esto se puede comprobar con los resultados obtenidos, que resultan en mas iteraciones que los presentados anteriormente.

$$f(x) = \cos^2(x) - x^2$$

[illegible]

```
Raiz: 0.73908513321516067229310920083662495017051696777343750000
Iterations: 54
Tolerancia 1e-16
```

```
Raiz: 0.73908513321516067229310920083662495017051696777343750000
Iterations: 63
Tolerancia 1e-32
```

```
Raiz: 0.73908513321516067229310920083662495017051696777343750000
Iterations: 63
Tolerancia 1e-56
```

$$f(x) = e^x - x - 1$$

[illegible][illegible][illegible][illegible]

$$f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$$

[illegible][illegible][illegible]

```
Raiz: 0.6666987072676420211791992187500000000000000000000  
Iterations: 30  
Tolerancia 1e-56
```

$$f(x) = x^3 - 2x - 5$$

[illegible]

```
Raiz: 2.09455148154232650981043661886360496282577514648437500000
Iterations: 54
Tolerancia 1e-16
```

Después de $e-16$ la maquina no logra llegar al resultado por su ϵ

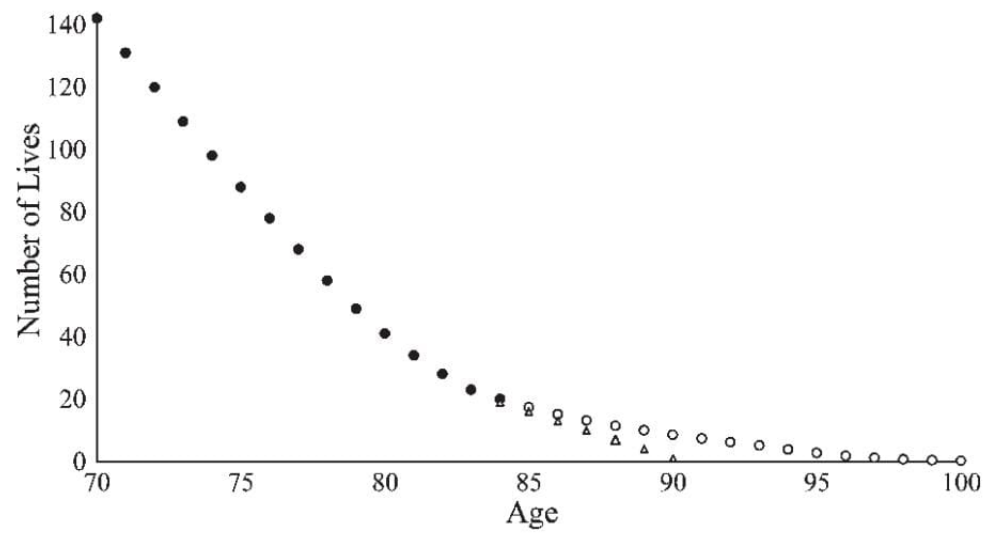


Fig. 2. Jones's (○) and Taylor's (Δ) extension to Halley's (●) life table

Figura 34: Taylor vs Halley