COMPARACIÓN GRÁFICA Y NUMÉRICA ENTRE MÉTODOS ITERATIVOS

Juan L. Varona

Dedicado a la memoria de José J. Guadalupe "Chicho", mi asesor de doctorado.

Introducción

Sea f una función $f : R \to R$ y ζ una raíz de f, es decir, $f(\zeta) = 0$. Es bien sabido que si tomamos x_0 cerca de ζ, y bajo ciertas condiciones que no explicaré aquí, el método de Newton

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

Genera una sucesión $\{x_n\}_{n=0}^{\infty}$ que converge a ζ. De hecho, las ideas originales de Newton sobre el tema, alrededor de 1669, eran considerablemente más complejas. Un estudio sistemático y una versión simplificada del método se deben a Raphson en 1690, por lo que este esquema de iteración también se conoce como método de Newton-Raphson. (También conocido como método de la tangente, debido a su interpretación geométrica).

En 1879, Cayley intentó utilizar el método para encontrar raíces complejas de funciones complejas. ciones $f : C \to C$. Si tomamos $z_0 \in C$ e iteramos

$$z_{n+1} = z_n - f(z_n) \frac{f(z_n)}{}, \quad n = 0, 1, 2,..., \tag{1}$$

Buscó condiciones bajo las cuales la secuencia $\{z_n\}_{n=0}^{\infty}$ converge a una raíz.
En particular, si denominamos la cuenca de atracción de una raíz ζ como el conjunto de todos los $z_0 \in C$ tales que el método converge a ζ, estaba interesado en identificar la cuenca de atracción para cualquier raíz. Resolvió el problema cuando f es un polinomio cuadrático. Para polinomios cúbicos, después de varios años de intentarlo, finalmente se negó a continuar. Ahora conocemos la naturaleza fractal del problema y podemos entender que el fracaso de Cayley para hacer cualquier progreso real en ese momento era inevitable. Por ejemplo, para $f(z) = z^3 - 1$, el conjunto de Julia (es decir, los puntos donde el método de Newton no converge) tiene dimensión fraccionaria, y coincide con la frontera de las cuencas de atracción de las tres raíces complejas $e^{2k\pi i/3}$, $k = 0, 1, 2$. Con la ayuda de imágenes generadas por computadora, podemos ver bonitas figuras que muestran la complejidad de estas intrincadas regiones. En la Figura 1, muestro las cuencas de atracción de las tres raíces (en realidad, esta imagen es bien conocida; por ejemplo, ya aparece publicada en [5] y, posteriormente, en [16] y [21]).

El aspecto computacional de los métodos iterativos se centra en dos aspectos: (a) encontrar raíces de ecuaciones no lineales, estudiando también la precisión y estabilidad de los algoritmos numéricos; (b) mostrar la belleza de las imágenes que se pueden generar con la ayuda de computadoras. El primer punto de vista se aborda mediante el análisis numérico. Entre los libros generales sobre este tema se encuentran [9, 13]; y libros más especializados.

Fecha: Primera versión: 4 de febrero de 1999. Versión actual: 11 de abril de 2001.

Este artículo se publicó, con pequeñas modificaciones, en The Mathematical Intelligencer 24 (2002), n.º 1, pp. 37-46. Además, la portada del número se diseñó con imágenes de este artículo (Figuras 3, 8, 9 y 12).

Los métodos iterativos se describen en [3, 15, 18]. Para una visión gráfica, véase, por ejemplo, [16].

Generalmente, existen tres estrategias para obtener imágenes mediante el método de Newton: (i) Se toma un rectángulo $D \subset C$ y se asigna un color (o nivel de gris) a cada punto $z_0 \in D$ según la raíz en la que converge el método de Newton, comenzando desde z_0. Si el método no converge, se marca el punto en negro (por ejemplo). De esta manera, se distinguen las cuencas de atracción por sus colores.

(ii) En lugar de asignar el color según la raíz obtenida con el método, lo hacemos según el número de iteraciones necesarias para alcanzar una raíz con una precisión fija. De nuevo, se utiliza el negro si el método no converge. Esto carece de interpretación matemática en lo que respecta a los conjuntos de Julia, pero también genera imágenes atractivas.

(iii) Esta es una combinación de las dos estrategias anteriores. En este caso, asignamos un color a cada cuenca de atracción de una raíz. Pero el color se aclara o se oscurece según el número de iteraciones necesarias para alcanzar la raíz con la precisión fija requerida. Como antes, usamos negro si el método no converge. En mi opinión, esto genera las imágenes más atractivas.

Todas estas estrategias se han utilizado ampliamente para polinomios, principalmente para polinomios de la forma $z^n - 1$, cuyas raíces son bien conocidas. Por supuesto, se han estudiado muchas otras familias de funciones. Remitimos al lector a [4, § 6] para más referencias. Por ejemplo, se obtiene una imagen atractiva al aplicar el método al polinomio $(z^2 - 1)(z^2 + 0{,}16)$ (gracias a S. Sutherland; véase [17, portada del número 2 para una versión a color]).

Aunque el método de Newton es el más conocido, existen en la literatura muchos otros métodos iterativos dedicados a hallar raíces de ecuaciones no lineales. Por lo tanto, el objetivo de este trabajo es el siguiente: mediante experimentos computacionales, estudiamos algunos de estos métodos iterativos para resolver $f(z) = 0$, donde $f : C \to C$, y mostramos las imágenes fractales que generan (principalmente, en el sentido descrito en (iii)). Además, esto nos permite comparar las regiones de convergencia de los métodos y su velocidad.

Conceptos relacionados con la velocidad de convergencia

Sea $\{z_n\}_{n=0}^{\infty}$ ser una secuencia compleja. Decimos que $\alpha \in [1, \infty)$ es el orden de convergencia de la secuencia si

$$\lim_{n\to\infty} \frac{|z_{n+1} - \zeta|}{|z_n - \zeta|^{\alpha}} = C, \tag{2}$$

donde ζ es un número complejo y C una constante distinta de cero; aquí, si $\alpha = 1$, suponemos una condición extra $C < 1$. Entonces, la convergencia de orden α implica que la secuencia $\{z_n\}_{n=0}^{\infty}$ converge a ζ cuando $n \to \infty$. (La definición previa para el orden de convergencia puede extenderse bajo algunas circunstancias; pero no me preocuparé por eso). También, se dice que el orden de convergencia es al menos α si se permite que la constante C en (2) sea 0, o, el equivalente, si existe una constante C y un índice n_0 tal que $|z_{n+1} - \zeta| \le C|z_n - \zeta|^{\alpha}$ para cualquier $n \ge n_0$. Muchas veces, el al menos se da por sentado implícitamente. Lo haré en este artículo.

El orden de convergencia se utiliza para comparar la velocidad de convergencia de secuencias, entendiendo esta como el número de iteraciones necesarias para alcanzar el límite con la precisión requerida. Supongamos que tenemos dos secuencias $\{z_n\}_{n=0}^{\infty}$ y $\{w_n\}_{n=0}^{\infty}$ convergiendo al mismo límite ζ, y supongamos que tienen, respectivamente, órdenes de convergencia α y β donde $\alpha > \beta$. Entonces, es evidente que, asintóticamente, la secuencia $\{z_n\}_{n=0}^{\infty}$ converge al límite más rápidamente (con menos iteraciones para la misma aproximación) que la otra secuencia.

Medidas más refinadas para la velocidad de convergencia son los conceptos de eficiencia informativa e índice de eficiencia (véase [18, § 1.24]). Si cada iteración requiere d nuevos datos (un "dato informativo" suele ser cualquier evaluación de una función o una de sus derivadas), entonces la eficiencia informativa es $\sqrt[d]{\alpha}$ y el índice de eficiencia es $\alpha^{1/d}$, donde α es el orden de convergencia. Para los métodos que se abordan en este artículo, es fácil derivar tanto la eficiencia informativa como el índice de eficiencia a partir de su orden. Lo haré aquí para el índice de eficiencia.

El índice de eficiencia es útil porque permite evitar aceleraciones artificiales de un método iterativo: por ejemplo, supongamos que tenemos un proceso iterativo $z_{n+1} = \varphi(z_n)$ con orden de convergencia α y tomamos un nuevo proceso $z_n = \varphi(\varphi(z_n))$. Entonces, es evidente que la nueva secuencia es simplemente z_n z_{n+1}. $z_0 = z_0$, $z_n = z_{2n}$, y Pero $\{z_n\}_{n=0}^{\infty}$ tiene un orden de convergencia α2. Sin embargo, ambas secuencias $\{z_n\}_{n=0}^{\infty}$ $\{z_n\}_{n=0}^{\infty}$ tienen el mismo índice de eficiencia.

En mi opinión, cuando tenemos un método iterativo $z_{n+1} = \varphi(z_n)$, el índice de eficiencia es más adecuado que el orden de convergencia para medir el tiempo de computación que un método utiliza para converger. Sin embargo, como ocurre en nuestro caso, si φ está relacionado con una función f y sus derivadas, el índice de eficiencia aún tiene un elemento faltante: no considera el trabajo computacional involucrado en el cálculo de f, f,...

Para evitar esto, se da un nuevo concepto de eficiencia: la eficiencia computacional (ver [18, Apéndice C]). Supóngase que, en un método φ relacionado con una función f, el coste de evaluar φ es θ(f) (por ejemplo, en el método de Newton, si el coste de evaluar f y f son, respectivamente, θ0 y θ1, tenemos θ(f) = θ0 + θ1); entonces, la eficiencia computacional de φ relativa a f es $E(\varphi, f) = \alpha^{1/\theta(f)}$ donde, de nuevo, α es el orden de convergencia. Pero es difícil establecer claramente el valor de θ(f) y, además, puede depender del ordenador, por lo que la eficiencia computacional no se utiliza mucho en la práctica. Así, en la literatura, la más utilizada de estas medidas es el orden de convergencia; Sin embargo, este es el que proporciona menos información sobre el tiempo de computadora necesario para encontrar la raíz con la precisión requerida.

Finalmente, cabe destacar que, para asegurar la convergencia de un método iterativo $z_{n+1} = \varphi(z_n)$, diseñado para resolver una ecuación $f(z) = 0$, suele ser necesario comenzar el método desde un punto z_0 cercano a la solución ζ. La cercanía depende de φ y f.
Normalmente, las hipótesis de los teoremas que garantizan la convergencia (remito al lector a las referencias de cada método) son difíciles de comprobar y, además, demasiado exigentes. Por lo tanto, si queremos resolver $f(z) = 0$, es común probar un método sin tener en cuenta ninguna hipótesis. Claro que esto no garantiza la convergencia, pero es posible que encontremos una solución (si hay más de una solución, tampoco podemos saber cuál se encontrará).

Aquí, realizaremos experimentos numéricos con diferentes funciones (simples y difíciles de evaluar) que nos permitirán comparar el tiempo de cálculo empleado. Además, comenzaremos las iteraciones en diferentes regiones del plano complejo. Esto nos permitirá medir, en cierta medida, la exigencia del método respecto al punto de partida para encontrar una solución. A medida que el fractal resultante se vuelve más complejo, parece que el método requiere más condiciones en el punto inicial.

Los métodos numéricos

En esta sección, consideraremos algunos métodos iterativos $z_{n+1} = \varphi(z_n)$ para resolver $f(z) = 0$ para una función compleja $f: C \to C$. Solo se ofrece una breve descripción y algunas referencias. En todos estos métodos, se parte de z_0 C. En ningún caso se presenta la hipótesis que asegura la convergencia ni el orden de convergencia correspondiente.

• Método de Newton: Este es el método iterativo (1), el más conocido y utilizado, y se puede encontrar en cualquier libro de análisis numérico. Ya lo comenté en la introducción. Su orden de convergencia es 2. • Método de Newton para raíces múltiples:

$$= zn - f(zn)2 - \frac{f(zn)f(zn)\, zn+1}{f(zn)f(zn)}.$$

En realidad, el método de Newton tiene orden 2 cuando la raíz de f obtenida es simple. Para una raíz múltiple, su orden de convergencia es 1. Este método recupera el orden 2 para raíces múltiples. Esto se deduce de la siguiente manera: si f f(z) tiene una raíz de multiplicidad m ≥ 1 en ζ, es fácil comprobar que g(z) = f(z) tiene una raíz simple en ζ. Entonces, solo necesitamos aplicar el método de Newton ordinario a la ecuación g(z) = 0.

• Aceleración convexa del método de Whittaker [11]: f(zn)

$$= zn - 2f(zn) \quad \text{——} \quad (2 - Lf(zn))\, zn+1$$

con

$$\text{El método } \frac{f(z)f(z)}{\text{de}}.$$

Whittaker (también conocido como método de las cuerdas paralelas, por su interpretación geométrica para funciones f : R → R, véase [15, pág. 181]) es una simplificación del método de Newton en el que, para evitar calcular la derivada, realizamos la aproximación f (z) ≈ 1/λ con λ constante. Intentamos elegir el parámetro λ de tal manera que F(z) = z − λf(z) sea una función contractiva, de modo que se pueda obtener un punto fijo mediante el teorema del punto fijo (es evidente que un punto fijo para F es una raíz para f). Este es un método de orden 1. La aceleración convexa es un método de orden 2.

• Aceleración doblemente convexa del método de Whittaker [11]:

$$zn+1 = zn - \frac{f(zn)}{2 - Lf(zn) + 4f(zn)\, \frac{4+2Lf(zn)}{2 - Lf(zn)(2 - Lf(zn))}}.$$

Esta es una nueva aceleración convexa para el proceso iterativo anterior. Tiene orden 3.

• El método de Halley (véase [18, p. 91], [3, p. 247], [9, p. 257], [8]):

$$zn+1 = zn - \frac{f(zn)}{f(zn)} \frac{2}{2 - Lf(zn)} = zn - \frac{1}{\frac{f(zn)}{f(zn)} - \frac{f(zn)}{2f(zn)}}.$$

Esta función fue presentada alrededor de 1694 por Edmund Halley, reconocido por ser el primero en calcular la órbita del cometa Halley. Es una de las funciones de iteración redescubiertas con mayor frecuencia en la literatura. Por su interpretación geométrica para funciones reales, también se conoce como el método de las hipérbolas tangentes. Alternativamente, puede interpretarse como la aplicación del método de Newton a la ecuación g(z) = 0, con g(z) = f(z)/f(z). Su orden de convergencia es 3. • Método de Chebyshev (véase [18, págs. 76 y 81] o [3, pág. 246]): f(zn)

$$zn+1 = zn - f(zn) \quad \text{——} \quad 1 + \frac{Lf(zn)}{2}.$$

Este método también se conoce como método de Euler-Chebyshev o, por su interpretación geométrica para funciones reales, método de las parábolas tangentes. Es de orden 3. (Este método y el anterior son probablemente los métodos de orden 3 más conocidos para resolver ecuaciones no lineales).

• Aceleración convexa del método de Newton o método del super-Halley [7]:

$$z_{n+1} = z_n - \frac{f(z_n)}{2f(z_n)} \frac{2 - Lf(z_n)}{1 - Lf(z_n)} = z_n - \frac{f(z_n)}{f(z_n)} + 1 \frac{\frac{1}{2}Lf(z_n)}{1 - Lf(z_n)}.$$

Éste es un método de orden 3. (Nótese que, en [3, p. 248], se llama método Halley-Werner.)

Un grupo de procedimientos para resolver ecuaciones no lineales son los métodos de punto fijo, dedicados a resolver F(z) = z. El más conocido de estos métodos es el que itera $z_{n+1} = F(z_n)$; es un método de orden 1 y requiere una hipótesis sólida sobre F para converger; es decir, requiere que F sea una función contractiva.

Un método de orden 2 para resolver una ecuación F(z) = z es el punto fijo de Stirling método [3, pág. 251 y pág. 260]. Comienza en un punto adecuado z0 y itera

$$z_{n+1} = z_n - 1 - \frac{z_n - F(z_n)}{F(F(z_n))}.$$

Si queremos resolver una ecuación f(z) = 0, podemos transformarla en una ecuación de punto fijo. Para ello, podemos tomar F(z) = z − f(z). Entonces, queda claro que F(z) = z f(z) = 0, por lo que podemos intentar usar un método de punto fijo para F. Pero esta no es la única manera: por ejemplo, podemos tomar F(z) = z − λf(z) con λ = 0 una constante (de esta manera, aparece el ya mencionado método de Whittaker), o F(z) = z − (z)f(z) con una función no nula. Además, podemos aislar z en la expresión f(z) = 0 de diferentes maneras (por ejemplo, si tenemos $z^3 - z + \tan(z) = 0$, podemos aislar $z^3 + \tan(z) = z$ o $\arctan(z - z^3) = z$). Por supuesto, esto da lugar a muchas ecuaciones de punto fijo diferentes F(z) = z para la misma ecuación original f(z) = 0.

Además, cuando intentamos resolver f(z) = 0 mediante un método iterativo $z_{n+1} = \varphi(z_n)$, como los mostrados arriba, y $\{z_n\}_{n=0}^{\infty}$ converge a ζ, está claro que ζ es un punto fijo para φ (requiriendo que φ sea una función continua y tomando límites en $z_{n+1} = \varphi(z_n)$) por lo que, sin darnos cuenta, estamos tratando con métodos de punto fijo.

Pero es interesante comprobar qué sucede si simplemente usamos la forma más simple de tomar F(z) = z − f(z) sin preocuparnos por ninguna hipótesis. De esta manera, tenemos

• Método de Stirling (desplazado):

$$z_{n+1} = z_n - f(z_n - \frac{f(z_n)}{f(z_n))}.$$

Su orden de convergencia es 2.

En todos los métodos vistos hasta ahora, la función f o sus derivadas se evalúan, en cada paso del método, para un único punto. Existen otras técnicas para resolver ecuaciones no lineales que requieren la evaluación de f o sus derivadas en más de un punto en cada paso. Estos métodos iterativos se conocen como métodos multipunto. Se emplean generalmente para aumentar el orden de convergencia sin calcular más derivadas de la función involucrada. Un estudio general de los métodos multipunto se puede encontrar en [18, Cap. 8 y 9]. Analicemos algunos métodos multipunto.

• El método de Steffensen (véase [15, p. 198] o [18, p. 178]):

$$z_{n+1} = z_n - \frac{f(z_n)}{g(z_n)}$$

con $g(z) = \frac{f(z+f(z)) - f(z)}{f(z)}$.

Este es uno de los métodos multipunto más sencillos. La función iterativa se genera mediante una estimación de la derivada: en el método de Newton, para un valor $h = f(z)$ suficientemente pequeño, estimamos $f'(z) \approx \frac{f(z+h) - f(z)}{h} = g(z)$. Esto evita calcular la derivada de f. Esto es h

Un método de orden 2 (observe que conserva el orden de convergencia del método de Newton). • Método del punto medio (véase [18, p. 164] o [3, p. 197]): f(zn) zn+1 = zn − f(zn) f zn − 2f(zn)

.

Este es un método de orden 3. • Método de Traub-Ostrowski (véase [18, pág. 184] o [3, pág. 230]): f(zn − u(zn))

2f(zn − u(zn)) − f(zn) − f(zn) zn+1 = zn − u(zn)

con u(z) = f(z) f(z). Su orden de convergencia es 4, el más alto para los métodos que estamos estudiando. • El método de Jarratt [12, 2] (para diferentes expresiones, véase también [3, p. 230 y pág. 234]):

$$z_{n+1} = z_n - \frac{1}{2}u(z_n) + \frac{f(z_n)}{f(z_n) - 3f(z_n - \frac{2}{3}u(z_n))}$$

donde, de nuevo, u(z) = f(z) f(z). Este también es un método de orden 4. • Método de Jarratt sin inverso (véase [6] o [3, p. 234]):

$$u(z_n)h(z_n)\left(1 - h(z_n), \frac{4}{2}\right) \quad \frac{3}{-} \; z_{n+1} = z_n - u(z_n) + \frac{3}{-}$$

con u(z) = f(z) f(z) y h(z) = 2 fz− 3u(z) −f(z) f(z) . También un método de orden 4.

Imágenes fractales y tablas comparativas

Vamos a aplicar los métodos iterativos que hemos visto en el ejemplo anterior. Sección para obtener las raíces complejas de las funciones

$$f(z) = z^3 - 1 \text{ y } f(z) = \exp\left(\frac{\text{pecado}(z)}{100}\right)(z^3 - 1).$$

Es evidente que las raíces de f son las mismas que las de f, es decir, 1, e πi/3 y e πi/3. Sin embargo, la función f es mucho más compleja (es decir, requiere más tiempo de cálculo) de evaluar. Además, las derivadas sucesivas de f son cada vez más fáciles, y esto, en general, es justo lo contrario. Esto no ocurre con f . Por lo tanto, f puede ser una mejor prueba para comprobar la velocidad de estos métodos numéricos de forma más general. (Cabe destacar que muchos de estos métodos iterativos también están adaptados para resolver sistemas de ecuaciones o ecuaciones en espacios de Banach. En este caso, evaluar las derivadas de Frèchet suele ser muy difícil).

Tomamos un rectángulo D C y aplicamos los métodos iterativos comenzando en “cada” z0 D. En la práctica, tomamos una cuadrícula de 1024×1024 puntos en D y usamos estos puntos como z0. Además, usamos dos regiones diferentes: el rectángulo Rb = [−2.5, 2.5] × [−2.5, 2.5] y un pequeño rectángulo cerca de la raíz e2πi/3 (≈ −0.5+0.866025i), el rectángulo Rs = [−0.6, −0.4] × [0.75, 0.95]. El primer rectángulo contiene las tres raíces; los métodos numéricos que comienzan desde un punto en Rb pueden converger a algunas de las raíces o, eventualmente, divergir. Sin embargo, Rs está cerca de una raíz, por lo que se espera que cualquier método numérico que comience allí siempre converja a la raíz.

En todos estos casos, utilizamos una tolerancia ε = 10−8 y un máximo de 40 iteraciones. y Denotamos las tres raíces como ζk = e kπi/3, k = 0, 1, 2. , φ el método iterativo Luego, tomamos z0 en el rectángulo correspondiente e iteramos zn+1 = φ(zn) hasta |zn − ζk| < ε para k = 0, 1 o 2. Si no hemos obtenido el resultado deseado

Tabla 1. Función f y rectángulo Rb.

	Orden Eff NC I/PTP/SI/S					
Noroeste	21,41	0,00267	7,52	111		
NwM	21,26	0,00381	7,93	1,17	0,857	0,904
CaWh	21,41	24,5	18,9	3,23	0,309	0,778
DcaWh	3	1,44	0,125	6,5	1,41	0,711	0,615
Ha	3	1,44	0	4,38	0,901	1,11	0,646
Cap.	3	1,44	0,04926	27	1,11	0,902	0,752
CaN/sH	3	1,44	0	3,82	0,815	1,23	0,623
Revolver	21,41	86,6	36,4	4,71	0,212	1,03	
Steff	21,41	85	35,7	5,79	0,173	0,820	
Mediados	3	1,44	4,62	6,32	1,1	0,911	0,766
Tr-Os	4	1,59	0	3,69	0,696	1,44	0,705
Sí	4	1,59	0	3,69	0,699	1,43	0,702
IfJa	4	1,59	1,62	7,45	1,41	0,711	0,705

Tabla 2. Función f y rectángulo Rs.

	Orden Eff NC I/PTP/SI/S					
Noroeste	21.41	0	2 .97	111		
NwM	21,26	0	2 ,97	1,1	0,910	0,910
CaWh	21,41	0	3,2	3	1,39	0,719	0,781
DcaWh	3	1,44	0	2	1,1	0,911	0,613
Ja	3	1,44	0	2	1,03	0,974	0,656
Ch	3	1,44	0	2	0,914	1,09	0,737
CaN/sH	3	1,44	0	2	1,06	0,946	0,636
Revolver	21,41	0	4,15	1,36	0,733	1,02	
Stef	21,41	0	3,44	1.42	0.706	0,82	
Mediados	3	1,44	0	2	0,898	1,11	0,749
Tr-Os	4	1,59	0	1,96	0,925	1,08	0,714
Sí	4	1,59	0	1,96	0,928	1,08	0,712
IfJa	4	1,59	0	1,99	0,969	1,03	0,690

tolerancia con 40 iteraciones, no continuamos y decidimos que la iterativa
El método que comienza en z0 no converge a ninguna raíz.

Con estos resultados, combinando f y f con Rb y Rs construimos cuatro tablas.
En ellos se identifican los métodos de la siguiente manera: Nw (Newton), NwM (Newton para raíces múltiples), CaWh (aceleración convexa de Whittaker), DcaWh (doble convexa aceleración de Whittaker), Ha (Halley), Ch (Chebyshev), CaN/sH (aceleración convexa de Newton o super-Halley), Stir (Stirling), Steff (Steffensen), Mid (punto medio), Tr-Os (Traub-Ostrowski), Ja (Jarratt), IfJa (Jarratt libre inverso).

Para cada uno de ellos mostramos la siguiente información:

- Ord: Orden de convergencia.
- Eff: Índice de eficiencia.
- NC: No hay puntos convergentes, como porcentaje del número total de puntos de inicio. puntos evaluados (que son 10242 para cada método).
- I/P: Media de iteraciones, medida en iteraciones/punto.
- T: Tiempo utilizado en segundos relativo al método de Newton (Newton = 1).
- P/S: Velocidad en puntos/segundo relativa al método de Newton (Newton = 1).
- I/S: Velocidad en iteraciones/segundo relativa al método de Newton (Newton = 1).

Tabla 3. Función f y rectángulo Rb.

	Orden Eff NC I/PTP/SI/S				
Noroeste	21.41 3.06 8.17 111				
NwM 21,26 2,86 8,2 1,47 0,681 0,683					
CaWh 21,41 33,219,9 3,58 0,279 0,679					
DcaWh 3 1.44 18,1 1,88 0,5320,714 11					
Ja	3 1,44 0,321 4,48 0,918 1,09 0,597				
Ch	3 1,44 11,5 9,11 1,56 0,641 0,714				
CaN/sH 3 1.44 1,924,59 0,907 1,10 0,619					
Remover	21,41 87,7 36,5 4,04 0,248 1,10				
Steff	21,41 84,5 35,6 3,39 0,295 1,28				
Medio	3 1,44 5,61 6,57 1,21 0,824 0,662				
Tr-Os	4 1,59 1,10 4,03 0,677 1,48 0,729				
Sí	4 1,59 0,965 3,99 0,777 1,29 0,628				
IfJa	4 1,59 1,71 0,5849 0,772				

Tabla 4. Función f y rectángulo Rs.

	Orden Eff NC I/PTP/SI/S				
Noroeste	21.41 0 2 .97 111				
NwM 21,26 0 2 ,97 1,50 0,666 0,666					
CaWh 21,41 0 3,2 2 1,67 0,599 0,649					
DcaWh 3 1.44 0 2 1,13 0,883 0,594					
Ja	3 1,44 0 2 1,10 0,906 0,61				
Ch	3 1,44 0 2 1,06 0,944 0,636				
CaN/sH 3 1.44 0 2 1,120,895 0,602					
Revolver 21,41 0 4,13 1,38 0,724 1,01					
Stef 21,41 0 3,43 1,06 0,945 1,09					
Mediados 3 1.44 0 2 1.020.979 0.659					
Tr-Os 4 1,59 0 1,96 0,909 1,1 0,727					
Sí 4 1,59 0 1,96 1,04 0,959 0,634					
IfJa 4 1,59 0 1,99 1,05 0,955 0,639					

Para construir las tablas, utilicé un programa en C++ en un Power Macintosh 8200/120 computadora. En las tablas, muestro el tiempo y la velocidad relativos al método de Newton, por lo que que esto será aproximadamente igual en cualquier otra computadora. En nuestra computadora, Los valores absolutos para el método de Newton son los siguientes:

- Para la Tabla 1, 137,467 seg, 7627,86 pt/seg y 57336,9 it/seg.
- Para la Tabla 2, 59,1667 seg, 17722,4 pt/seg y 52610,2 it/seg.
- Para la Tabla 3, 410,683 seg, 2553,25 pt/seg y 20870,6 it/seg.
- Para la Tabla 4, 150,083 seg, 6986,63 pt/seg y 20737 it/seg.

En cualquier caso, un lenguaje de programación informática que permita tratar operaciones con números complejos de la misma forma que con números reales (como C++ o
Se recomienda encarecidamente utilizar Fortran.

Con respecto a nuestras medidas de tiempo, es importante tener en cuenta que, para cualquier método iterativo zn+1 = φ(zn), he escrito procedimientos generales aplicables a f genérica y sus derivadas. Esto significa, por ejemplo, que cuando uso f , hago no simplificar ningún factor en $\frac{f(z)}{(f')(z)}$. Además, si una subexpresión de (f) ya tiene se ha calculado en f (por ejemplo, sin(z)) en el procedimiento genérico para evaluar f, su valor no se utiliza, sino que se calcula de nuevo en el procedimiento que calcula la f genérica. Si Sólo nos interesaba una función particular f (o si queremos una cifra en el orden más rápido

De esta manera), sería posible modificar el procedimiento que itera zn+1 = φ(zn) para f, adaptando y simplificando su expresión.

Ahora, volvamos al otro objetivo de este trabajo: comparar visualmente las imágenes fractales que aparecen cuando aplicamos diferentes métodos iterativos para resolver una ecuación única f(z) = 0, donde f es una función compleja.

En las figuras 1 a 12, muestro las imágenes que aparecen al aplicar los métodos iterativos para hallar las raíces de la función f(z) = z3 − 1 en el rectángulo Rb. He utilizado la estrategia (iii) descrita en la introducción. Respectivamente, asigno cian, magenta y amarillo a las cuencas de atracción de las tres raíces 1, e πi/3 y e πi/3, más claras u oscuras según el número de iteraciones necesarias para alcanzar la raíz con la precisión fija requerida. Marco en negro los puntos z0 Rb para los cuales el método iterativo correspondiente, comenzando en z0 , no alcanza ninguna raíz con una tolerancia de 10−3 en un máximo de 25 iteraciones.

En la última sección de este artículo, muestro los programas que he utilizado y otros similares que permiten generar figuras en escala de grises o en color. Por supuesto, también es posible usar la función f o el pequeño rectángulo Rs (o cualquier otra función o rectángulo); solo se requieren pequeñas modificaciones en los programas. Esto queda a criterio del lector interesado.

Aunque usar un lenguaje de programación común suele ser cientos de veces más rápido, para generar las imágenes resulta más sencillo si empleamos un programa informático con funciones gráficas, como Mathematica, Maple o Matlab. Los gráficos que muestro aquí se generaron con Mathematica 3.0 (véase [20]); en la siguiente sección, muestro los programas utilizados para obtener las figuras.

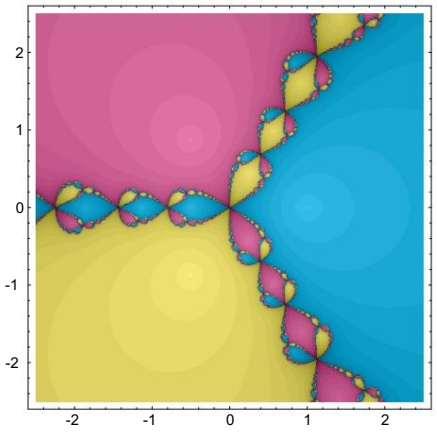

Figura 1. Método de Newton.

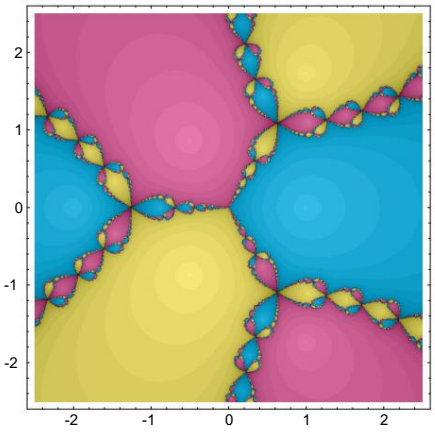

Figura 2. Método de Newton para raíces múltiples.

Obsérvese que tanto el método de Traub-Ostrowski como el de Jarratt para f(z) = z −1 generan la función iterativa φ(z) = 6z +42z +33z . Por lo $\frac{1+12z^3+54z^6+14z^9}{}$ tanto, la figura fractal para ambos es la misma (Figura 11), y lo mismo ocurre con los datos de las Tablas 1 y 2.

En las tablas y figuras, solo disponemos de datos empíricos. A partir de ellos, y según las indicaciones de este artículo, podemos juzgar fácilmente el comportamiento y la idoneidad de cualquier método según las circunstancias, y cuál elegir si necesitamos resolver una ecuación. Es un buen entretenimiento.

Finalmente, cabe destacar que los métodos de Stirling y Steffensen generan resultados muy diferentes en dos sentidos. En primer lugar, son los más exigentes con respecto al punto inicial (en las tablas, véase el porcentaje de puntos no convergentes y, en las figuras,

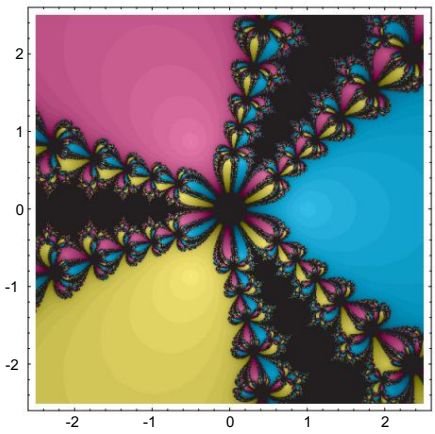

Figura 3. Aceleración convexa de
El método de Whittaker.

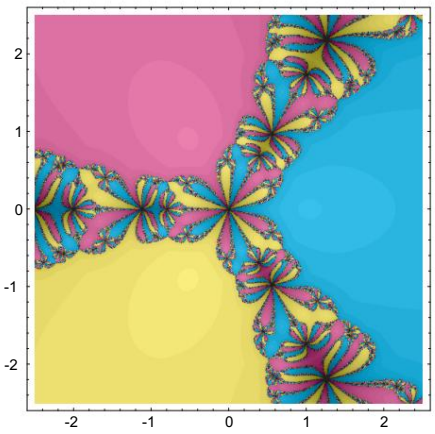

Figura 4. Aceleración doblemente convexa
del método de Whittaker.

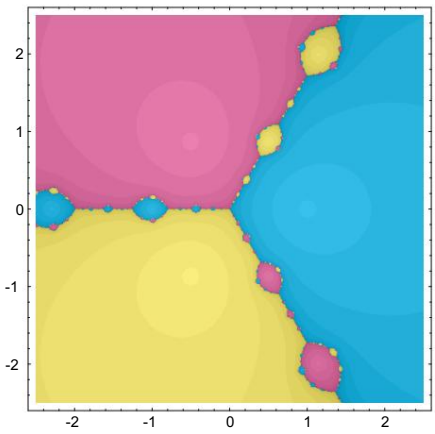

Figura 5. Método de Halley.

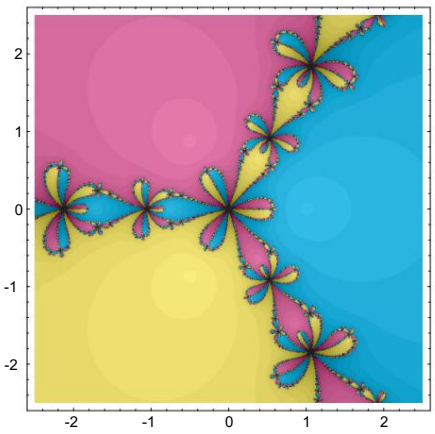

Figura 6. Método de Chebyshev.

ver el color negro). Y, segundo, en sus gráficos, la simetría del ángulo 2π/3
que podemos observar en las figuras correspondientes a los otros métodos no
aparecen (con respecto a la simetría en los fractales, ver [1]).

Programas de Mathematica para obtener los gráficos

En esta sección voy a explicar cómo se generaron las figuras de este trabajo.
Para ello, muestro los programas de Mathematica [20] que he utilizado.

Primero, necesitamos definir la función f y sus derivadas. Esto se puede hacer usando
f[z_] := z^3-1, df[z_] := 3*z^2 y d2f[z_] := 6*z, pero es más rápido si usamos
las versiones compiladas

```
f = Compilar[{{z,_Complejo}}, z^3-1];
df = Compilar[{{z,_Complejo}}, 3*z^2];
d2f = Compilar[{{z,_Complejo}}, 6*z];
```

Por supuesto, cualquier otra función, como f (z) = exp(100 Si las raíces son las mismas (como sucede $\frac{\text{pecado}(z)}{\text{con f y}}$)(z3 − 1), se puede utilizar.
f), solo necesitamos cambiar
Las definiciones anteriores de f, df y d²f. Las cifras que aparecen son algo...
diferente.

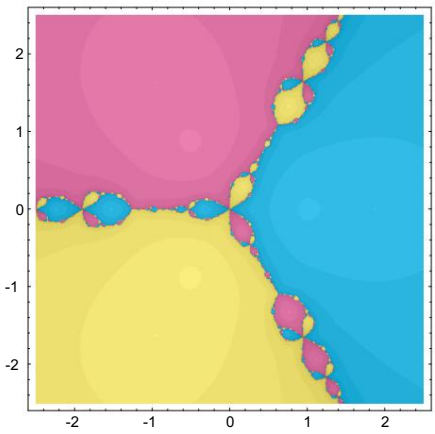

Figura 7. Aceleración convexa de El método de Newton (o super-Halley) método).

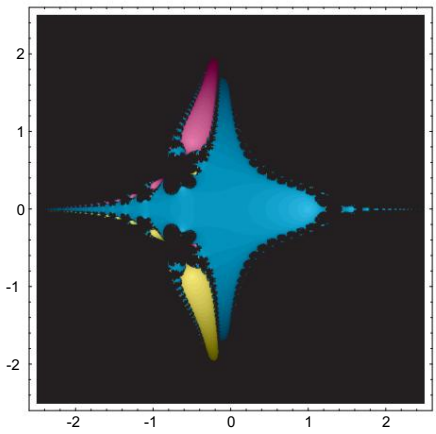

Figura 8. Método de Stirling (desplazado).

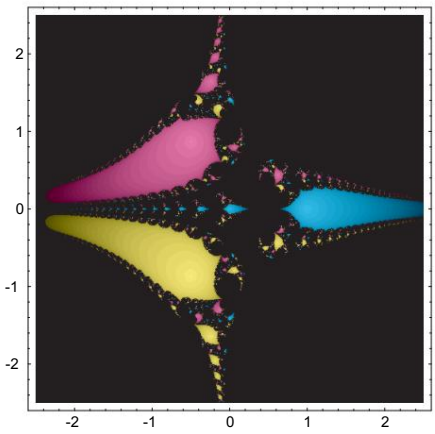

Figura 9. Método de Steffensen.

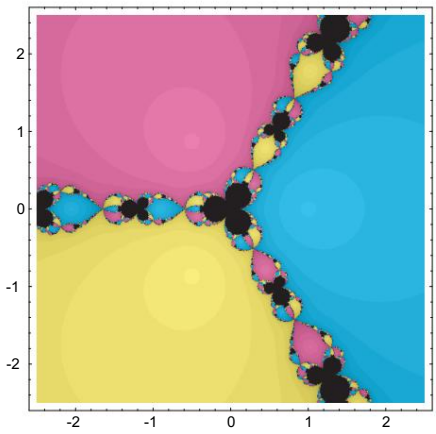

Figura 10. Método del punto medio.

Las tres raíces complejas para f son

```
Do[raíz f[k] = N[Exp[2*(k-1)*Pi*I/3]], {k,1,3}]
```

Utilizo el siguiente procedimiento que identifica qué raíz se ha aproximado con una tolerancia de 10−3, si la hubiera,

```
PosiciónRaíz = Compilar[{{z,_Complejo}},
    ¿Cuál[Abs[z - raízf[1]] < 10.0^(-3), 3,
        Abs[z - raízf[2]] < 10.0^(-3), 2,
        Abs[z - raízf[3]] < 10.0^(-3), 1,
        Verdadero, 0],
    {{rootf[_],_Complejo}}
    ]
```

Debemos definir los métodos iterativos, es decir, los diferentes zn+1 = φ(zn). Para El método de Newton, esto sería

```
iterNewton = Compilar[{{z,_Complejo}}, zf[z]/df[z]]
```

y, para el método de Halley,

```
iterHalley = Compilar[{{z,_Complejo}},
```

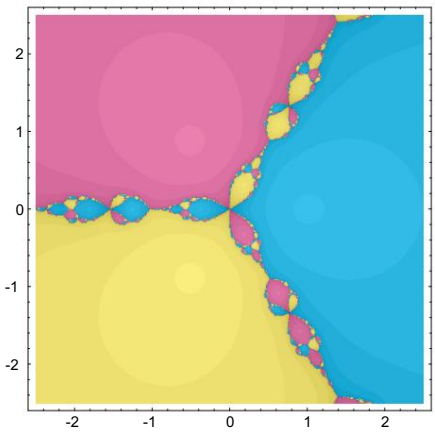

Figura 11. Método de Traub-Ostrowski y método de Jarratt.

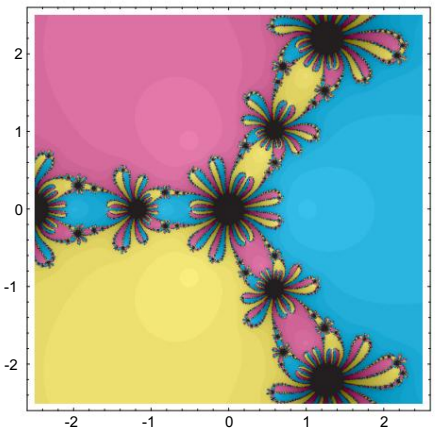

Figura 12. Método de Jarratt sin inversión.

```
Bloque[{v = df[z]}, z - 1.0 / (v/f[z] - (d2f[z])/(2.0*v))]
]
```

(Observe que se utiliza una variable adicional v para evaluar df[z] solo una vez). El procedimiento es similar para todos los demás métodos de este artículo.

El algoritmo que itera la función iterMethod para ver si se alcanza una raíz en un máximo de iteraciones limitadas es lo siguiente:

```
iterAlgoritmo[iterMétodo_,x_,y_,lim_] :=
    Bloque[{z,ct,r}, z = x + y I; ct = 0; r = rootPosition[z];
        Mientras [(r == 0) && (ct < lim),
            ++ct; z = iterMétodo[z]; r = posición raíz[z]
        ];
        Si[Cabeza[r] == Cual, r = 0]; (* "Cual" sin evaluar *)
        Regresar[r]
    ]
```

Aquí tengo en cuenta que, a veces, Mathematica no es capaz de hacer una evaluación numérica de z. Entonces, no puede asignar un valor para r en rootPosition. En cambio, devuelve un Which sin evaluar. Por supuesto, esto corresponde a puntos no convergentes.

Vamos a utilizar un límite de 25 iteraciones y el rectángulo complejo [−2.5, 2.5]× [−2.5, 2.5]. Para ello, defino las siguientes variables:

```
limIteraciones = 25;
xxMín = -2,5; xxMáx = 2,5; yyMín = -2,5; yyMáx = 2,5;
```

Finalmente, defino el procedimiento para pintar las figuras según la estrategia (i) Se describe en la introducción. Se utilizan blanco, 33 % de gris y 66 % de gris para identificar las cuencas de atracción de las tres raíces 1, e2πi/3 y e4πi/3. Los puntos para los cuales El método iterativo no llega a ninguna raíz (con la tolerancia deseada en el máximo de iteraciones) se representan en negro. La variable puntos significa que, para Para generar la imagen, se debe utilizar una cuadrícula de puntos × puntos.

```
plotFractal[iterMethod_,puntos_] :=
    DensityPlot[iterAlgoritmo[iterMétodo,x,y,limIteraciones],
        {x, xxMín, xxMáx}, {y, yyMín, yyMáx},
        Rango de trama->{0,3}, Puntos de trama->puntos, Malla->Falso
    ] // Momento
```

Tenga en cuenta que // Timing al final nos permite observar el tiempo que Mathematica emplea cuando se usa plotFractal.

Luego se obtiene un gráfico de esta manera (el ejemplo es una versión en blanco y negro de la

Figura 1): plotFractal[iterNewton, 256]

Al usar las funciones definidas, pueden producirse errores de desbordamiento y subdesbordamiento (por ejemplo, en el método de Newton, f(z) puede ser nulo y entonces se divide por cero, aunque no es el único problema). Mathematica nos informa de estas circunstancias; para evitarlas, use lo siguiente antes de llamar a plotFractal: Off[General::ovfl]; Off[General::unfl];

Off[Infinity::indet]

Además, los problemas anteriores y algunos otros a veces obligan a Mathematica a usar una versión no compilada de las funciones. De nuevo, Mathematica nos informa de esta circunstancia; para evitarlo, use

Off[CompiledFunction::cccx]; Off[FunciónCompilada::cfn];
Off[CompiledFunction::cfcx]; Desactivado[CompiledFunction::cfex];
Off[CompiledFunction::crcx]; Quizás Desactivado[FunciónCompilada::ilsm]

otras funciones Off sean útiles según la función f y el rectángulo complejo utilizado.

Para obtener las figuras de color, utilizo un procedimiento ligeramente diferente para identificar qué raíz se ha aproximado; esto se debe a que también queremos saber cuántas iteraciones son necesarias para llegar a la raíz. Usamos el siguiente truco: en la salida, la parte entera corresponde a la raíz y la parte fraccionaria está relacionada con el número de iteraciones.

```
iterColorAlgoritmo[iterMethod_,x_,y_,lim_] :=
    Bloque[{z,ct,r}, z = x + y I; ct = 0; r = rootPosition[z];
        Mientras[(r == 0) && (ct < lim), ++ct; z =
            iterMethod[z]; r = rootPosition[z]
        ];
        Si[Head[r] == Cuál, r = 0]; (* "Cuál" sin evaluar *)
        Devuelve[N[r+ct/(lim+0.001)]]
    ]
```

Para asignar la intensidad del color de un punto, considero el número de iteraciones necesarias para alcanzar la raíz cuando el método iterativo comienza en ese punto. Utilizo cian, magenta y amarillo para los puntos que alcanzan, respectivamente, las raíces 1, e πi/3 y e πi/3; y negro para los puntos no convergentes. Para ello, utilizo

```
colorLevel = Compilar[{{p,_Real}}, 0.4*ParteFraccionaria[4*p]]
```

y

```
fractalColor[p_] :=
    Bloque[{pp = nivelDeColor[p]},
        Conmutador[ParteEntera[4*p], 3,
            ColorCMYK[0.6+pp,0.,0.,2*pp], 2,
            ColorCMYK[0.,0.6+pp,0.,2*pp], 1,
            ColorCMYK[0.,0.,0.6+pp,2*pp], 0,
            ColorCMYK[0.,0.,0.,1.]
        ]
    ]
```

(En el comportamiento interno de Mathematica, cuando se va a representar una función con DensityPlot, se escala a [0, 1]. Sin embargo, iterColorAlgorithm tiene un rango de [0, 4]; esta es la razón para usar 4*p en algunos lugares en colorLevel y

ColorFractal. Tenga en cuenta también que se puede cambiar el nivel de color para modificar la intensidad de los colores; para otros gráficos, conviene experimentar modificando los parámetros para obtener imágenes atractivas.

Finalmente, se representará un fractal de color llamando al procedimiento

```
plotColorFractal[iterMethod_,points_] :=
    Gráfico de densidad[
        iterColorAlgorithm[iterMethod,x,y,limIterations], {x, xxMin, xxMax}, {y,
        yyMin, yyMax},
        PlotRange->{0,4}, PlotPoints->puntos, Mesh->False,
        Función de color->Color fractal
    ] // Momento
```

Por ejemplo,

plotColorFractal[iterNewton, 256] es solo la

Figura 1.

Familias de métodos iterativos

Existen muchos métodos iterativos para resolver ecuaciones no lineales en las que hay aparece un parámetro, que normalmente se denominan familias de métodos iterativos.

Una de las más conocidas es la familia Chebyshev-Halley Lf (zn) 1 − βLf (zn)

$$z_{n+1} = z_n - \frac{f(z_n)\, f}{(z_n)} \quad 1\,1 + \frac{}{2} \frac{}{} ,$$

con β como parámetro real. Estos son métodos de orden 3 para resolver la ecuación f(z) = 0. Casos particulares son β = 0 (método de Chebyshev), β = 1/2 (método de Halley) y β = 1 (método de super-Halley). Cuando β → −∞, obtenemos el método de Newton. Esta familia fue estudiada por Werner en 1980 (véase [19]), y también se puede encontrar en [3, p. 219] y [10]. Es interesante observar que cualquier proceso iterativo dado por la expresión

$$z_{n+1} = z_n - \frac{f(z_n)}{\text{donde}} H(Lf(z_n)), f(z_n),$$

la función H satisface H(0) = 0, H (0) = 1/2 y |H(0)| < ∞, genera un método iterativo de orden 3 (véase [8]). De esta manera, la familia Chebyshev-Halley surge al tomar H(x) = 1 + 1−βx.

$\frac{}{1\,2}\frac{}{}$

Una familia multipunto (ver [18, p. 178]) es

$$z_{n+1} = z_n - \frac{f(z_n)}{g(z_n)}$$

g(z) = $\frac{f(z+\beta f(z))-f(z)}{}$ y β una constante arbitraria (β = 1 corresponde al método βf(z) de Steffensen). Su orden de convergencia es 2.

King [14] estudió una familia multipunto de orden 4 (véase también [3, p. 230]): f(zn − u(zn))

$$(\beta - 2)f(z_n - u(z_n)), f(z_n) \frac{f(z_n) + \beta f(z_n - u(z_n))}{\text{donde } \beta \text{ es un número real arbitrario y } u(z) = f(z)} z_{n+1} = z_n - u(z_n) - f(z_n) +$$

Generaliza el método de Traub-Ostrowski (que es el caso particular $\frac{f(z)}{\beta}$ = 0).

Finalmente, citemos otra familia multipunto de orden 4 1 + βh(zn)

$$3\; z_{n+1} = z_n - u(z_n) + u(z_n)h(z_n)\; 4\; 1+\left(\frac{}{\frac{3}{2} + \beta)h(z_n)}\right),$$

donde β es un parámetro y u, h denotan u(z) = y h(z) = f(z) $\frac{f(z)}{}$ $\frac{2\,f z - 3\,f(z)}{u(z) - f(z)}$.

Aquí, para β = 0, obtenemos el método de Jarratt (en realidad, en [12] aparece una familia diferente;

El método que llamamos método de Jarratt es en realidad un caso particular para ambas familias. Además, para $\beta = -3/2$, obtenemos el llamado método de Jarratt inverso-libre.

Los métodos iterativos uniparamétricos ofrecen una interesante posibilidad gráfica: mostrar imágenes en movimiento. Tomamos una función fija y un rectángulo fijo, y representamos las imágenes fractales para muchos valores del parámetro. Esto genera una atractiva imagen en movimiento que muestra la evolución de las imágenes fractales al variar el parámetro. Lamentablemente, no es posible mostrar imágenes en movimiento en un artículo.
Para generarlas en una computadora, podemos usar pequeñas modificaciones de los programas de Mathematica de la sección anterior, incluyendo los comandos "Animar" o "MostrarAnimación". Posteriormente, es posible exportar estas imágenes en formato Quick-Time (para que no sea necesario Mathematica para verlas). Claro que esto requiere mucho tiempo de procesamiento, pero las computadoras son cada vez más rápidas, así que no es un gran problema.

Referencias

1. C. Alexander, I. Giblin y D. Newton, Grupos de simetría en fractales, The Mathematical Intelligencer 14 (1992), no. 2, 32–38.
2. IK Argyros, D. Chen y Q. Qian, El método Jarratt en el entorno del espacio de Banach, J. Comput. Matemáticas Aplicadas. 51 (1994), 103–106.
3. IK Argyros y F. Szidarovszky, La teoría y aplicaciones de los métodos de iteración, CRC Press, Boca Raton, FL, 1993.
4. W. Bergweiler, Iteración de funciones meromórficas, Bull. Amer. Math. Soc. (NS) 29 (1993), 151–188.
5. P. Blanchard, Dinámica analítica compleja en la esfera de Riemann, Bull. Amer. Math. Soc. (NS) 11 (1984), 85–141.
6. JA Ezquerro, JM Gutiérrez, MA Hernández y MA Salanova, La aplicación de una aproximación de tipo Jarratt libre-inversa a ecuaciones integrales no lineales de tipo Hammerstein, Comput. Math. Appl. 36 (1998), 9–20.
7. JA Ezquerro y MA Hernández, Sobre una aceleración convexa del método de Newton, J. Optim. Teoría Appl. 100 (1999), 311–326.
8. W. Gander, Sobre el método de iteración de Halley, Amer. Math. Monthly 92 (1985), 131–134.
9. W. Gautschi, Análisis numérico: una introducción, Birkh¨auser, Boston, 1997.
10. JM Gutiérrez y MA Hernández, Una familia de métodos tipo Chebyshev-Halley en Banach espacios, Bull. Austral. Math. Soc. 55 (1997), 113–130.
11. MA Hernández, Un procedimiento de aceleración del método de Whittaker mediante convexidad, Zb. Radiante. Prirod.-Mat. Falso. Ser. Estera. 20 (1990), 27–38.
12. P. Jarratt, Algunos métodos iterativos multipunto de cuarto orden para resolver ecuaciones, Math. Comp. 20 (1966), 434–437.
13. D. Kincaid y W. Cheney, Análisis numérico: Matemáticas de la computación científica, 2.ª ed., Brooks/Cole, Pacific Grove, CA, 1996.
14. RF King, Una familia de métodos de cuarto orden para ecuaciones no lineales, SIAM J. Numer. Anal. 10 (1973), 876–879.
15. J. M. Ortega y W. C. Rheinboldt, Solución iterativa de ecuaciones no lineales en varias variables, Monografías, Libros de texto, Computación, Ciencias Aplicadas, Matemáticas, Academic Press, Nueva York, 1970.
16. HO Peitgen y PH Richter, La belleza de los fractales, Springer-Verlag, Nueva York, 1986.
17. M. Shub, Misterios de las matemáticas y la computación, The Mathematical Intelligencer 16 (1994), no. 1, 10–15.
18. JF Traub, Métodos iterativos para la solución de ecuaciones, Prentice-Hall, Englewood Cliffs, Nueva Jersey, 1964.
19. W. Werner, Algunas mejoras de los métodos iterativos clásicos para la solución de ecuaciones no lineales, en Solución numérica de ecuaciones no lineales (Proc., Bremen, 1980), EL Allgo-wer, K. Glashoff y HO Peitgen, eds., Lecture Notes inMath. 878 (1981), 427–440.
20. S. Wolfram, El libro de Mathematica, 3.ª ed., Wolfram Media/Cambridge University Press, 1996.
21. JW Neuberger, El método continuo de Newton para polinomios, The Mathematical Intelligencer 21 (1999), núm. 3, 18–23.

Departamento de Matemáticas y Computación, Universidad de La Rioja, Edificio JL Vives, Calle Luis de Ulloa s/n, 26004 Logroño, España
Dirección de correo electrónico: jvarona@dmc.unirioja.es
URL: http://www.unirioja.es/dptos/dmc/jvarona/welcome.html