La columna de Matemática Computacional

Sección a cargo de

Laureano González Vega

Sobre la iteración cúbica inversa

por

Robert M. Corless

Resumen. Se han publicado miles de artículos sobre el cálculo de raíces de ecuaciones no lineales univariadas y aquí vamos a introducir uno más. Se trata de un método aparentemente nuevo que llamaremos «Iteración *Cúbica* Inversa» (ICI) por analogía con la denominada «Iteración *Cuadrática* Inversa» (IQI) que se utiliza en el método de Brent-Dekker. Este método, posiblemente nuevo, se basa en una combinación cúbica de aproximaciones lineales tangentes a la función inversa de la ecuación a resolver. Para mejorar la estabilidad numérica se reescribe esta iteración como un *promedio* de dos pasos del método de Newton y un paso del método de la secante: de esta forma solo se necesita, en cada paso, una nueva evaluación de la función y una evaluación de su derivada. El coste total de este método es, por lo tanto, solo ligeramente mayor que el del método de Newton, y veremos que su orden es $1 + \sqrt{3} = 2.732\ldots$, lo que garantiza que para lograr una precisión determinada se requerirán menos pasos que en el método de Newton (usando esencialmente el mismo esfuerzo computacional en cada paso).

1. Introducción

Recientemente he escrito, con Erik Postma, un artículo sobre *combinaciones* de series de Taylor [5]. Una «combinación» no es más que un interpolante de Hermite de dos puntos: si tenemos una serie para $f(x)$ en $x = a$ y otra serie para la misma función $f(x)$ en $x = b$ entonces podemos combinar estas dos series para obtener una (habitualmente) mejor aproximación a $f(x)$ en todo el intervalo. En dicho artículo se analiza cómo calcular de forma estable y eficiente combinaciones de orden superior con Maple y se describen algunas aplicaciones.

Por el contrario, en este artículo se analiza una combinación de orden inferior y su aplicación al cálculo de raíces. De hecho, todo lo que necesitaremos es una cúbica. Si solo usamos series de orden 1 en cada punto, a saber $f(a)$ y $f'(a)$ y $f(b)$ y $f'(b)$,

entonces estos cuatro valores determinan, como es bien sabido, un interpolante de Hermite cúbico.

Lo que es menos conocido (pero, siendo honesto, perfectamente conocido) es que a partir de estos cuatro valores también se puede construir un interpolante cúbico de la función *inversa* inv $f(y)$, que denotaremos también $\breve{f}(y)$.[1] Los nodos que usaremos para calcular dicho interpolante son $y = f(a)$ e $y = f(b)$ y los valores de la función inversa serán a y b y los valores de la derivada de la función inversa en $y = f(a)$ e $y = f(b)$ son $1/f'(a)$ y $1/f'(b)$: estos cuatro valores serán suficientes para determinar un polinomio cúbico en y que se ajusta a los datos que proporciona la función inversa.

Por supuesto, en nuestra situación, no podemos tener $f'(a) = 0$ o $f'(b) = 0$ y, de hecho, $f'(x) = 0$ en cualquier lugar del intervalo considerado son malas noticias para cualquier algoritmo de cálculo de raíces: la raíz estaría mal condicionada, si es que existe. Si $f'(x) = 0$ en cualquier lugar del intervalo considerado entonces no se puede garantizar que la función sea monótona, por lo que la función inversa no estaría bien definida en todo el intervalo. Una vez hecha esta advertencia, no nos volveremos a preocupar por esta cuestión en todo lo que sigue.

2. Generación de la fórmula

En casi cualquier libro de Análisis Numérico se puede encontrar una expresión equivalente a la siguiente para un interpolante cúbico de Hermite, ajustando cuatro valores (los valores de $f(x)$ y $f'(x)$ en $x = a$ y $x = b$) para obtener un polinomio cúbico que se ajuste a la información disponible:

$$y = (1 + 2\theta)(\theta - 1)^2 f(a) + \theta(\theta - 1)^2 h f'(a) + \theta^2(3 - 2\theta) f(b) + \theta^2(\theta - 1) h f'(b)$$

donde $\theta = (x - a)/(b - a)$ y $h = b - a$. Cuando $\theta = 0$, $x = a$; cuando $\theta = 1$, $x = b$. Como se tiene

$$d/d\theta = dx/d\theta \cdot d/dx = h\, d/dx$$

entonces todo cuadra: $y = f(a)$ cuando $\theta = 0$, $dy/d\theta = hf'(a)$ y $y = f(b)$ cuando $\theta = 1$ y $dy/d\theta = hf'(b)$ cuando $\theta = 1$.

A continuación repetimos este análisis para la función inversa de $f(x)$. Pongamos $s = (y - f(a))/(f(b) - f(a))$ y $\Delta = f(b) - f(a)$. Si $f(a) = f(b)$ entonces nos encontramos en una situación problemática: pero en este caso, el teorema de Rolle nos proporciona la existencia de c entre a y b donde $f'(c) = 0$. Si esto ocurriera, entonces, como ya hemos advertido, evitaremos esta dificultad. De todas formas, tanto la Iteración Cuadrática Inversa (IQI) como el método de la secante *también* fallan en este caso, por lo que estamos en buena compañía.

Queremos

$$x = \left((1 + 2s)a + \frac{s\Delta}{f'(a)}\right)(1 - s)^2 + \left((3 - 2s)b - \frac{(1 - s)\Delta}{f'(b)}\right)s^2.$$

[1] En [8] se introduce esta notación para la función inversa; puede verse cómo usar el teorema de inversion de Lagrange para calcular sus interpolantes y, en particular, la aplicación al caso concreto de la función W de Lambert.

Usando un razonamiento similar al que hemos usado antes, podemos ver que si $s = 0$ entonces $x = a$ y $dx/dy = 1/f'(a)$, mientras que si $s = 1$ entonces $x = b$ y $dx/dy = 1/f'(b)$.

¿Por qué hacemos esto? Lo que suele llamar la atención de la Iteración Cuadrática Inversa, el que para encontrar un valor aproximado de x que hace $y = 0$ es suficiente sustituir $y = 0$ en la fórmula para la función inversa, también va a funcionar aquí. Así, poner $y = 0$ significa que $s = (0 - f(a))/(f(b) - f(a))$ (un número entre 0 y 1), lo que nos proporciona el valor buscado para x. Haciendo esto, obtenemos la siguiente fórmula para x, el valor aproximado de la raíz que estamos buscando:

$$\left(\left(1 - 2\,\frac{f(a)}{\Delta}\right)a - \frac{f(a)}{f'(a)}\right)\left(1 + \frac{f(a)}{\Delta}\right)^2 + \frac{(f(a))^2}{\Delta^2\left(\left(3 + 2\,\frac{f(a)}{\Delta}\right)b - \frac{\Delta}{f'(b)}\left(1 + \frac{f(a)}{\Delta}\right)\right)}.$$

La expresión obtenida es sin duda correcta pero sospecho que numéricamente muy poco útil. Para ser honesto, ni siquiera lo he probado. En su lugar, he buscado una forma de reescribir esta fórmula que mejore su estabilidad numérica.

3. Una fórmula numéricamente más estable

Después de una cantidad sorprendentemente pequeña de manipulaciones algebraicas (y posiblemente una generosa porción de suerte) se puede llegar a la siguiente expresión:

$$\begin{aligned} x_{n+1} = {} & \frac{y_n^2}{(y_{n-1} - y_n)^2}\left(x_{n-1} - \frac{y_{n-1}}{f'(x_{n-1})}\right) + \frac{y_{n-1}^2}{(y_{n-1} - y_n)^2}\left(x_n - \frac{y_n}{f'(x_n)}\right) \\ & - 2\,\frac{y_{n-1}y_n}{(y_{n-1} - y_n)^2}\left(x_n - \frac{y_n(x_n - x_{n-1})}{y_n - y_{n-1}}\right), \end{aligned} \tag{1}$$

con la notación $y_n = f(x_n)$. A esta fórmula es a lo que llamaremos Iteración Cúbica Inversa, o ICI en todo lo que sigue.

En lugar de describir una tediosa transformación de la fórmula anterior a esta (son equivalentes, créanme, a pesar de haber hecho el cambio de a y b por x_{n-1} y x_n, para mostrar por primera vez la iteración), verificaremos en la siguiente sección que esta fórmula tiene el comportamiento apropiado mostrando cómo de rápido converge a una raíz (cuando eso ocurre). Lo que más me interesa de esta fórmula es que en ella podemos *identificar* y poner nombre a los tres términos en la ecuación (1). Podemos así encontrar una iteración de Newton empezando en x_{n-1}, otra empezando en x_n y una iteración *secante* usando los dos puntos. Estos tres términos vienen acompañados de unos pesos muy concretos[2]: y_n^2, y_{n-1}^2 y $-2y_ny_{n-1}$, todos ellos divididos por $(y_n - y_{n-1})^2$, de forma que su suma, como tiene que ser, dé 1. Así, de los tres términos, el

[2]Podría haber sido mejor, desde el punto de vista numérico, el promediar los puntos anteriores $a_0x_n + a_1x_{n-1} + a_sx_n$, y promediar las pequeñas actualizaciones $a_0y_n/f'(x_n) + a_1y_{n-1}/f'(x_{n-1}) + a_s(y_n\Delta x/\Delta y)$, para luego usar esta actualización de ese promedio en la mejora del promedio de los puntos anteriores. Esta es la forma como lo he implementado, pero no creo que haya mucha diferencia entre las dos posibilidades.

que tenga el peso más grande proporciona la mejor estimación, lo que permite utilizar los otros términos como errores. Todo ello me lleva a asegurar que esta fórmula tiene una estructura especialmente llamativa (e incluso atractiva).

La estabilidad numérica (o su ausencia) del método de la secante y del método de Newton es bien conocida. Escribiendo el método de la secante como se indicó anteriormente hacemos un «pequeño cambio» en un valor existente y, de manera similar, el método de Newton se escribe de la manera más estable posible; por lo tanto, tenemos lo que probablemente sea una fórmula razonablemente estable.

Puesto que este método usa dos puntos x_{n-1} y x_n para generar un tercer punto x_{n+1}, parece similar al método de la secante. Debido a que usa la derivada $f'(x_k)$ en cada punto, también parece similar al método de Newton. No es muy similar a la IQI que ajusta la inversa de un función cuadrática en tres puntos y la usa para generar un cuarto punto. El método de Halley y otras iteraciones de Schröder utilizan derivadas de orden superior y tienen un mayor coste en cada paso. Podría ser interesante intentar promediar métodos de orden superior de una manera similar para acelerar la convergencia aún más, pero hasta ahora no lo he logrado. El sencillo truco de reemplazar f por $f/\sqrt{f'}$ que transforma el método de Newton en el método de Halley (y acelera el método de la secante) no parece funcionar con los pesos que tenemos en la ICI.

Voy a empezar asumiendo que comenzamos con una estimación inicial x_0 y luego generamos x_1 mediante un paso del método de Newton. Entonces es obvio que para llevar a cabo un paso con la ecuación (1) solo necesitamos *una* evaluación más de la función $y_1 = f(x_1)$ y *una* evaluación más de la derivada $f'(x_1)$, porque podemos guardar los valores $y_0 = f(x_0)$ y $f'(x_0)$ anteriores y no tener que volver a calcularlos. En los modelos habituales de iteración usados en los algoritmos de cálculo de raíces se tiene que el coste dominante viene de la evaluación de la función y de su derivada. Sin embargo este método no es más caro que el método de Newton: en lugar de calcular $x_{n+1} = x_n - f(x_n)/f'(x_n)$ y olvidarnos de x_{n-1}, una vez que hemos determinado esta iteración de Newton, la promediamos con la iteración de Newton anterior junto con una iteración del método de la secante. El coste de formar este promedio no es más que unas pocas operaciones de números en coma flotante que podemos considerar despreciables si las comparamos con el coste que supone evaluar $f(x)$ y $f'(x)$. Cuando probamos esto en el ejemplo clásico que se usa con el método de Newton, $z^3 - 2z - 5$ con el valor inicial de $z_0 = 1$, obtenemos 29 dígitos de precisión después de 7 iteraciones (y 10 dígitos de precisión después de 6 iteraciones).

Esbozaré el costo de este método usando el formalismo habitual en la sección 5.

4. El orden del método

Si asumimos que r es la raíz que estamos buscando, $f(r) = 0$, y que $x_{n-1} = r + \varepsilon_{n-1}$ y $x_n = r + \varepsilon_n$, entonces el desarrollo en serie que proporciona Maple directamente muestra que

$$x_{n+1} = r + (\varepsilon_{n-1}\varepsilon_n)^2 \, \frac{f^{(4)}(r)f''(r) - 10f'(r)f''(r)f^{(3)}(r) + 15(f''(r))^3}{4!\,(f'(r))^3} + \cdots . \quad (2)$$

El error en el residuo, que es lo que realmente mediremos, es ligeramente diferente: $y(x_k) = f(r+\varepsilon_k) = f(r) + f'(r)\varepsilon_k$, y entonces podemos eliminar un factor $f'(r)$ del denominador anterior. Debemos observar que cada x_k es una solución exacta de la ecuación $F(x) = f(x) - y_k$. Esta es una observación trivial pero a menudo muy útil: al iterar obtenemos las soluciones exactas para ecuaciones ligeramente diferentes. Aquí, sin embargo, no importa puesto que ambas fórmulas muestran la tendencia de los errores. Es decir, el error en la siguiente iteración es el *cuadrado del producto* de los dos errores anteriores. Esto es totalmente análogo a lo que ocurre con el método de la secante (producto de los dos errores anteriores) y con el método de Newton (cuadrado del error anterior). La resolución de la recurrencia $\ln \varepsilon_{n+1} = 2(\ln \varepsilon_n + \ln \varepsilon_{n-1})$ muestra que el error es asintóticamente la potencia $1+\sqrt{3} = 2.732\ldots$-ésima del error anterior. No es una convergencia cúbica pero es sustancialmente más rápida que una convergencia cuadrática.

A continuación vamos a comprobar que, con este método, lo que dice la teoría coincide con lo que ocurre en la práctica. Consideremos la función $f(x) = (x^2 + x)\exp(-x) - 1/3$, que tiene dos puntos donde $f'(x) = 0$, a saber, $x = (1 \pm \sqrt{5})/2$. Siempre que nos mantengamos alejados de estos dos puntos todo debería funcionar correctamente. Tomando como valor inicial $x_0 = 2.0$, lo primero que hacemos es calcular x_1 por el método de Newton, tal y como hemos indicado anteriormente. Ahora ya tenemos los dos valores de partida con los que trabajar; luego calculamos $x_2, x_3, \ldots, x_8$. Debido a que la convergencia es muy rápida, y quería demostrarlo, asigné al parámetro `Digits` de Maple el valor 1000. Desde luego esto es excesivo pero permite que el gráfico de la figura 1 muestre claramente que el error en x_8 es aproximadamente 10^{-594}. El método de Newton (cuadrados rojos) alcanza solo 10^{-63} con el mismo esfuerzo computacional. Las razones $y_k/(y_{k-1}y_{k-2})^2$ para $k = 2, 3, \ldots, 8$ fueron, respectivamente, 1.5952, 17.048, 4.5955, 4.9061, 4.9080, 4.9081 y 4.9080. Estos valores se ajustan a la curva $y_k = 0.6437^{(1+\sqrt{3})^k}$, lo que demuestra que el orden de este método es casi cúbico. La constante 0.6437 se calculó mediante el correspondiente ajuste al conjunto de puntos que se había determinado. La forma en la que evolucionan los resultados nos permite predecir que la precisión de x_9 sería $4.9081(y_8y_7)^2 = 1.7383 \cdot 10^{-1622}$. Si repetimos los cálculos con 1624 dígitos para llegar a x_9, obtenemos $y_9 = 1.7383 \cdot 10^{-1622}$, tal como se había predicho.

5. El coste computacional del método

El coste computacional asintótico de un método a menudo se describe por medio de lo que se llama el «índice de eficiencia de Ostrowski». Este índice cuenta el número de evaluaciones de funciones, p, por paso, y toma la p-ésima raíz de su orden asintótico del método en cuestión. Por ejemplo, un método similar al método de Newton como la iteración de Steffensen que usa $p = 2$ evaluaciones de función en cada paso, y tiene una convergencia de segundo orden como el método de Newton, tendría un índice de eficiencia igual a $\sqrt{2} \approx 1.41$. Si se considera que la evaluación de la derivada tiene el mismo costo que una evaluación de función, entonces obtenemos un índice de eficiencia para el método ICI de $\sqrt{1+\sqrt{3}} \approx 1.628$, que es más alto y,

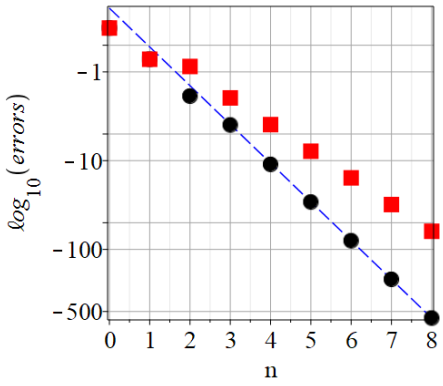

Figura 1: Convergencia a una raíz de $f(x) = (x^2 + x)\exp(-x) - 1/3$. El valor inicial utilizado ha sido $x_0 = 2.0$. La primera iteración se calculó mediante el método de Newton, $x_1 = x_0 - f(x_0)/f'(x_0)$. Posteriormente se utilizó la ecuación (1). El cálculo se realizó con 1000 dígitos de precisión. Lo que muestra la gráfica es el logaritmo (en base 10) del error residual, esto es, $\log_{10}|y_n| = \log_{10}|f(x_n)|$. La línea azul discontinua representa la curva teórica $y_n = 0.6437^{(1+\sqrt{3})^n}$ mostrando cómo el cálculo teórico del orden de este método coincide con los resultados obtenidos en la práctica. La constante 0.6437 se calculó mediante el correspondiente ajuste al conjunto de puntos que se había determinado. El método de Newton (cuadrados rojos) alcanza solo 63 dígitos de precisión en lugar de los casi 600 que alcanza la ICI.

por lo tanto, mejor. Se recomienda consultar [7] para obtener más información sobre este tema y encontrar ejemplos de métodos de alta eficiencia de este tipo. Existe una conjetura de Kung y Traub, creo que aún abierta, sobre el índice de eficiencia óptima para los métodos de un solo paso.[3]

Sin embargo, el mejor análisis *práctico* —que yo sepa— del coste computacional de los métodos para resolver una ecuación escalar iterando con precisión constante puede encontrarse en [14]. Allí, el autor sopesa el coste de las evaluaciones de la función y estima que el coste de las evaluaciones de las derivadas es solo ligeramente mayor que el de las funciones. Luego, comparando el mayor coste del método de Newton por paso con su necesidad, habitual, de menos iteraciones, concluye que en general el método de la secante debería ganar. Sí, el método de la secante suele requerir más iteraciones, pero cada una generalmente cuesta solo una fracción de un paso de Newton. Es *solo* cuando las derivadas son inusualmente económicas cuando el método de Newton es mejor. En el ejemplo de la figura 1, este es el caso: el coste dominante de la evaluación de la función es el coste de la función exponencial, y esto se puede reutilizar en la evaluación de la derivada. Entonces, para este problema en particular y usando estos argumentos, el método ICI debería vencer al método de Newton y con más rotundidad al método de la secante. Pero para ganar al método de Newton, ICI necesita tomar menos iteraciones porque el

[3]Se puede ver lo que afirma dicha conjetura en «La conjetura de Kung-Traub cumple 46 años», `https://institucional.us.es/blogimus/2020/01/5973/`.

coste por paso es esencialmente idéntico al método de Newton. ICI nunca[4] requiere *más* iteraciones que el método de Newton. ¿Pero se usarán menos?

Ese es realmente el problema. ¿Podremos evitar alguna iteración? El método de Newton suele converger muy rápidamente si partimos de una buena estimación inicial. Con el método de Newton, si el error inicial es 2^{-2} y todas las constantes son 1, podemos esperar una respuesta en precisión doble tras cinco iteraciones (en realidad, el error se desperdiciaría: 2^{-64}, y los errores de redondeo lo mejor donde lo habrían dejado sería en 2^{-52}). ICI hubiera mostrado la secuencia de errores $\varepsilon_0 = 2^{-2}$, $\varepsilon_1 = 2^{-4}$ (por supuesto igual que el método de Newton para primera iteración x_1), $\varepsilon_2 = 2^{-12}$ elevando al cuadrado el producto $2^{-2} \cdot 2^{-4}$, luego $\varepsilon_3 = 2^{-2(12+4)} = 2^{-32}$ que es el mismo error que el método de Newton logró tras *cuatro* iteraciones. Una iteración más nos da $\varepsilon_4 = 2^{-2(32+12)} = 2^{-88}$ que es verdaderamente grande respecto de los resultados anteriores. En resumen, ICI supera al método de Newton por una iteración con este error inicial.

Si, en cambio, la estimación inicial solo tiene una precisión de 2^{-1}, no 2^{-2}, el método de Newton requiere de seis iteraciones mientras que ICI *también* requiere seis: $\varepsilon_0 = 2^{-1}$, $\varepsilon_1 = 2^{-2}$, $\varepsilon_3 = 2^{-6}$, $\varepsilon_4 = 2^{-16}$, $\varepsilon_5 = 2^{-44}$. Si, en cambio, la estimación inicial tiene una precisión de 2^{-3} entonces el método de Newton requiere de cinco iteraciones, mientras que ICI produce $\varepsilon_1 = 2^{-6}$, $\varepsilon_2 = 2^{-18}$, $\varepsilon_3 = 2^{-48}$ y en cuatro iteraciones supera al método de Newton.

La lección parece clara: si se requiere una precisión muy alta entonces el método ICI dará los mismos resultados en menos iteraciones *con esencialmente el mismo coste*. Si se requiere un resultado en precisión doble entonces no se evitan muchas iteraciones, si es que se evita alguna (y si realmente ICI no supera al método de Newton entonces no es previsible que supere al método de la secante). Y, desde luego, *ambos* métodos exigen el tener unas estimaciones iniciales buenas, cuestión a la que dedicaremos la próxima sección.

6. Estimaciones o valores iniciales

Hasta ahora he estado usando el siguiente esquema: «elija solo una estimación inicial y deje que el método de Newton determine la siguiente iteración». Evidentemente existen estrategias alternativas. Me gusta la que he estado usando porque suele ser bastante difícil llegar a *una* buena estimación inicial de una raíz.

El uso de esta estrategia también permite dibujar cuencas de atracción: ¿qué valores iniciales convergen a qué raíz? Por ejemplo, considero la función $f(z) = z^3 - 1$ y genero una cuadrícula de 1600 por 1600 puntos en $-2 \le x \le 2$, $-2 \le y \le 2$ donde $z = x + iy$ y tomo como máximo 13 iteraciones de ICI comenzando desde cada punto de la cuadrícula. Si la iteración convergió antes (con un error menor a 10^{-8}), se guarda el resultado anterior. Dibujar el argumento (fase) de la respuesta proporciona un diagrama (con una apariencia) fractal, muy similar al que se obtendría con el método de Newton, pero diferente en los detalles. En particular, las cuencas de

[4] «¿Qué, nunca?» «¡No, nunca!» «¿Qué, *nunca*?» «¡Casi nunca!» —la tripulación y el capitán del HMS Pinafore—.

atracción están visiblemente desconectadas. Se pueden ver las imágenes que se han obtenido en la figura 2 y en la figura 3.

Otro ejemplo gráfico de la aplicación de este método, esta vez para una función $f(z)$ más complicada que un polinomio, se puede ver en la figura 4.

7. Raíces múltiples y derivadas infinitas

Por supuesto, este método no puede utilizarse para calcular raíces múltiples. Con este método, como con el método de Newton, estaríamos dividiendo por derivadas, que se aproximan a cero si nos acercamos a una raíz múltiple. Es cierto que el método *puede* funcionar, pero, si lo hace, lo hará más lentamente que en el caso de raíces no múltiples: incluso en ese caso, la sucesión $y_k/(y_{k-1}y_{k-2})^2$ simplemente crecerá sin límite. Aplicando nuestra iteración a $f(x) = (x-2)^2$ con $x_0 = 0.5$ observamos que se confirma esta expectativa. El método «convergió» y $x_{10} \approx 2 \pm 10^{-14}$, pero debemos tener en cuenta aquí que el método de Newton también converge linealmente para este problema.

Otra cuestión problemática a tener en cuenta está ligada a las raíces de la derivada. El que hayamos tenido que llegar hasta la *cuarta* derivada en la fórmula de error (2) sugiere que este método debe ser más sensible a las singularidades que el método de Newton.

8. Más preguntas...

La primera pregunta que se debe hacer aquí es ¿puede esta estrategia funcionar para *sistemas* de ecuaciones no lineales? Aunque este método, si nos olvidamos de sus orígenes, no es más que un promedio ponderado de dos iteraciones de Newton y de una iteración secante, ¿qué análogos podemos usar para sistemas? No tengo claro qué generalización del método secante usar, o qué usar en lugar de los pesos escalares: ¿reemplazar, por ejemplo, $y_k^2/(y_k - y_{k-1})^2$ por $\|y_k\|^2/\|y_k - y_{k-1}\|^2$ usando alguna norma? ¿Qué norma? Quizás funcione, con las opciones adecuadas. Pero incluso si funcionase, para sistemas de gran tamaño el mayor esfuerzo computacional generalmente se concentra en configurar y resolver los sistemas lineales: la iteración de Newton completa rara vez se usa, cuando las iteraciones aproximadas más baratas llevan a la solución más rápido. Además, el beneficio real de la mayor velocidad de convergencia de ICI no se aprecia completamente con precisión doble o más baja. Sospecho que solo podrían empezar a notarse sus efectos con precisión cuádruple o incluso mejor.

Creo que lo mejor que se puede esperar de este método es que se use en un sistema de Cálculo Simbólico para sus «resolvedores» escalares de alta precisión, que podrían usar otras iteraciones de orden superior. Es aquí, si es que hay algún lugar, donde este método será útil.

Hay iteraciones similares de dos pasos de orden superior, comenzando con la «iteración quinta inversa» que usa términos hasta la segunda derivada y cuyo orden es $(3 + \sqrt{21})/2 \approx 3.79$. No he investigado aún ninguno de esos esquemas. Hay un

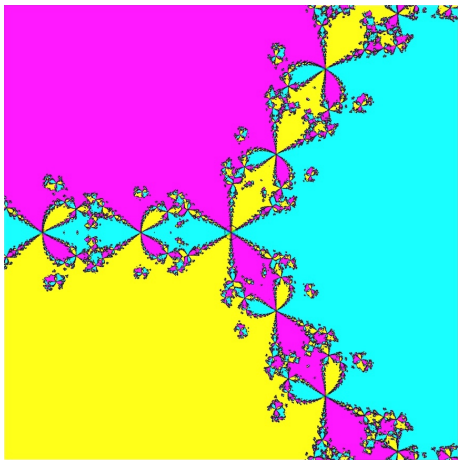

Figura 2: Cuencas de atracción aproximadas que proporciona el método ICI para $f(z) = z^3 - 1$. Cada estimación inicial $z_0 = x + iy$ se colorea con la fase (argumento) del resultado después de 13 iteraciones de ICI, a saber, $\arg(z_{13})$ (las iteraciones anteriores pueden haber convergido, en cuyo caso se registra el valor límite obtenido). La primera iteración z_1 se obtiene mediante el método de Newton. El cálculo se ha realizado en una cuadrícula de 1600 por 1600 puntos en $-2 \leq x \leq 2$, $-2 \leq y \leq 2$. Hay que destacar aquí que la estructura fractal obtenida es más complicada que la que proporciona del método de Newton lo que sugiere que, para este método, la elección del valor inicial es más complicada.

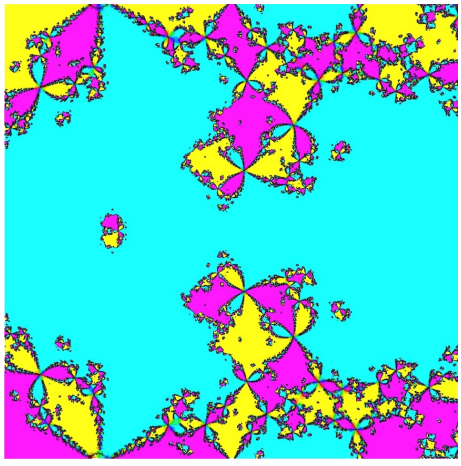

Figura 3: Mostramos aquí la región $-1.45 \leq x \leq -1.05$ y $-0.2 \leq y \leq 0.2$, nuevamente con una cuadrícula de 1600 por 1600 puntos, para la misma función e iteraciones que en la figura 2. Claramente pueden observarse componentes desconectadas.

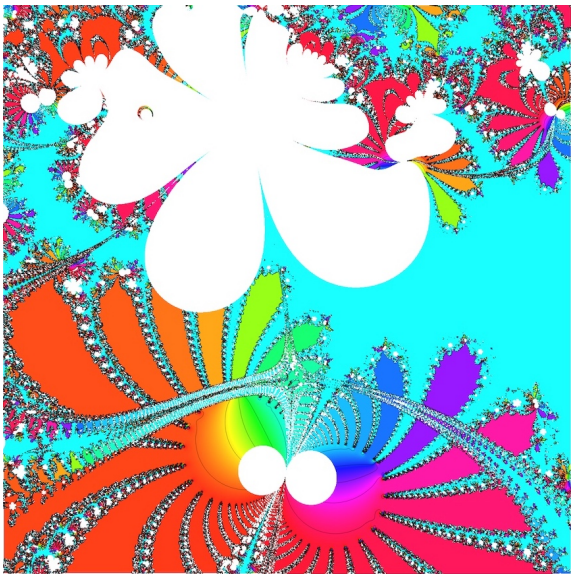

Figura 4: Cuencas de atracción bajo ICI para la ecuación de Kepler $z - 0.083\,\mathrm{sen}(z) - 1 = 0$ en una cuadrícula de 1600 por 1600 puntos en $-30.5 \leq x \leq -29.5$ y $-17.5 \leq y \leq -16.5$, con tolerancia 10^{-8} y un máximo de 30 iteraciones. Las áreas de color blanco puro representan áreas donde la iteración encontró un NaN. Otros colores representan diferentes fases después de 30 iteraciones o convergencia (lo que ocurra primero). La imagen es bastante más complicada que la imagen similar para el método de Newton que se puede encontrar en [4, figura 3.21].

número infinito de tales esquemas, pero es probable que sean útiles solo en esos casos raros en los que la evaluación de las derivadas está disponible a un coste muy bajo.

En aplicaciones que requieren de una precisión muy alta, a veces existe la opción de cambiar la precisión sobre la marcha. Esto es efectivo para la iteración de Newton: solo la evaluación final de la función y la evaluación de la derivada deben realizarse con la máxima precisión. No he investigado los efectos de la precisión variable sobre la eficiencia de este método. De hecho, tampoco he investigado el efecto de los errores de redondeo en la fórmula (1). He asumido simplemente que, debido a que estamos promediando iteraciones conocidas que tienen un comportamiento de error de redondeo razonable, también tendremos un comportamiento de error de redondeo razonable. Pero esto necesita de una prueba formal.

Luego hay asuntos más prácticos: ¿qué hacer en el inevitable caso de que $y_k = y_{k-1}$? ¡Por ejemplo, esto sucede cuando la iteración converge! ¿O cuando $f'(x_k)$ es demasiado pequeño? Resulta que los detalles del extremadamente práctico método `zeroin` [3], que combina IQI con bisección y el método de la secante para aumentar su aplicabilidad (este método se implementa en el comando `fzero` en Matlab), son muy importantes también aquí. Se pueden encontrar más mejoras del método en [17]; en particular, mejoran el comportamiento en el peor de los casos. Cualquiera que

sea el método de estos que se desee implementar, hay que prestar especial atención a todos estos detalles.

Además hay otro asunto de una naturaleza mucho más académica: en las decenas de miles de artículos sobre cálculo de raíces, ¿*nadie* ha pensado en esto antes? ¿En serio? Una vez más, tengo las dudas que tengo que tener, lógicamente. Así, como método de dos pasos técnicamente no es una de las iteraciones de Schröder (que se reinventan constantemente). Y tampoco es uno de los métodos de Kalantari [9]. ¡Pero es un método tan bonito! —¿seguramente alguien lo habrá descubierto antes y habrá intentado escribir sobre él?—. O quizás sea un ejercicio en uno de los grandes textos antiguos de Análisis Numérico, como el de Rutishauser (del cual aún no tengo una copia, pero sigo intentándolo).

Hay (literalmente) un número infinito de métodos iterativos para elegir: véanse por ejemplo las referencias en [12], incluyendo [16] y la traducción de G. W. Stewart de la obra de Schröder del siglo XIX titulada «On infinitely many algorithms for solving equations». La lista en [13] contiene decenas de miles de entradas.

Sin embargo, la búsqueda en la web de la expresión «Inverse Cubic Iteration» (seguramente una expresión bastante natural) falla. Encontramos artículos cuando buscamos expresiones como «Método de Newton acelerado», como [6]. Pero los que he encontrado son diferentes a ICI (por ejemplo, el último artículo citado encuentra un verdadero método de tercer orden; ICI solo es casi de tercer orden, su orden es solo $1+\sqrt{3}$). El método en [7] combina *Steffensen* y la iteración de Newton para obtener un método de cuarto orden con tres evaluaciones de funciones (y sin evaluaciones de la derivada) por paso. Los métodos en [15] son incluso de orden superior, más eficientes y también libres de evaluaciones de la función derivada. De hecho, esos métodos parecen mejorar a ICI bastante bien: si los hubiera conocido antes de comenzar este artículo podría posiblemente no haber «despegado», si me permiten la analogía. Inspirándonos en estas técnicas, hemos probado ICI utilizando las aproximaciones de Steffensen para las derivadas. Esto significa que usamos

$$f'(x) \approx \frac{f(x+\gamma f(x)) - f(x)}{\gamma f(x)},$$

para aproximar el valor de la derivada al final de cada paso[5] (donde $\gamma \neq 0$ es un parámetro que se usa para definir el método de Steffensen a utilizar). Esto parece funcionar y nos da una fórmula de error bastante interesante (después de un cálculo de desarrollos en serie que requiere de toda la potencia simbólica de Maple)

$$x_{n+1} = r + \frac{s_n^2 s_{n-1}^2 \varepsilon^2}{(s_n - s_{n-1})^2} f''(r) + \cdots,$$

donde $x_n = r + s_n\varepsilon$ y $x_{n-1} = r + s_{n-1}\varepsilon$. Aquí nuevamente tenemos que el error en la combinación de los pasos es el cuadrado del producto de los errores en el paso

[5] Si vamos a hacer esto entonces también deberíamos conocer la fórmula de Squire-Trapp $f'(x) = \mathrm{Im}(f(x+ih)/h)$: si la función es analítica y se implementa correctamente, esta fórmula proporciona valores muy precisos de la derivada evitando en gran medida los errores de redondeo; véase, por ejemplo, [4, cap. 11].

anterior, siempre que no fueran exactamente los mismos errores. Aún mejor, esto solo involucra la segunda derivada, y no la cuarta derivada de f como mencionábamos anteriormente. Nuevamente tenemos un método de orden $1+\sqrt{3}$ pero esta vez con dos evaluaciones de la función por paso, en lugar de una evaluación de la función y una evaluación de la derivada. Esto parece contradecir la conjetura de orden óptimo de Kung y Traub [11] que dice que el orden óptimo debiera ser 2^{d-1} con d evaluaciones de función por paso. Pero no la contradice completamente porque este método tiene memoria y, por lo tanto, su orden «superior al óptimo» se obtiene ignorando una de las reglas del juego. No he experimentado con esta combinación tipo Steffensen.

El método en [10], denominado «Leap-Frog Newton», es bastante similar al ICI y consiste en un valor intermedio de Newton seguido de un paso del método de la secante: esto requiere de dos evaluaciones de la función y de una evaluación de la derivada por paso (por lo tanto, es más costoso por paso que ICI, que requiere solo de una evaluación de la función y de una evaluación de la derivada por paso, después del primero) pero es realmente de tercer orden. Encontré este último artículo buscando la expresión «combining Newton and secant iterations», por lo que tal vez podría encontrar artículos que describan coincidencias más cercanas a ICI, si pudiera pensar en las expresiones de búsqueda más apropiadas. Actualmente estoy preguntando por cuál sería la expresión correcta a utilizar en esta búsqueda a todos mis amigos y conocidos. ¿Vosotros?

> «Six months in the laboratory can save you three days in the library»; de acuerdo con lo que me indicó mi colega Henning Rasmussen, esto es «folklore».

Nota añadida después de «arxiving» y consulta. Hasta la fecha, ninguno de mis amigos o conocidos o los revisores de este artículo han sugerido ningun trabajo relacionado. Por mera casualidad decidí mirar lo que está disponible en SciPy, el paquete de computación científica para Python y que aún no uso, a pesar de su creciente popularidad. Y me encontré con una agradable sorpresa: el algoritmo 748 de ACM TOMS, de Alefeld, Potra y Shi, uno de los métodos implementados [1] (en [2] se puede encontrar una versión de este algoritmo para aritmética de intervalos). Este algoritmo usa un *tipo* de interpolación cúbica inversa, aunque no del mismo tipo que hemos usado aquí, y sorprendentemente logra el mismo orden de convergencia asintótica que el método que hemos descrito aquí. Es más, no usa derivadas (solo evaluaciones de la función), por lo que en realidad puede ser *mejor* que el método que hemos presentado aquí. Lo mejor de todo es que ya está implementado y todo lo que hay que hacer es usarlo. Sin embargo, al igual que nuestro método, sus ventajas de eficiencia pueden ser importantes solo para cálculos que requieran de precisión muy alta. Y no tengo nada claro que la implementación en Python de este algoritmo permita cálculos con una precisión alta. Es en este momento cuando podemos y debemos escuchar los ecos de Eclesiastés 1:9 sobre, esta vez, los métodos de cálculo de raíces: «Lo que fue, eso será, y lo que se hizo, eso se hará; no hay nada nuevo bajo el sol».

Referencias

[1] G. E. Alefeld, F. A. Potra y Y. Shi, Algorithm 748: enclosing zeros of continuous functions, *ACM Trans. Math. Software* **21** (1995), no. 3, 327–344.

[2] G. E. Alefeld, F. A. Potra y W. Völker, Modifications of the interval-Newton-method with improved asymptotic efficiency, *BIT* **38** (1998), no. 4, 619–635.

[3] R. P. Brent, An algorithm with guaranteed convergence for finding a zero of a function, *Comput. J.* **14** (1971), no. 4, 422–425.

[4] R. M. Corless y N. Fillion, *A Graduate Introduction to Numerical Methods: From the Viewpoint of Backward Error Analysis*, Springer, New York, 2013.

[5] R. M. Corless y E. Postma, Blends in Maple, *arXiv:2007.05041*, 2020.

[6] J. M. Gutiérrez y M. A. Hernández, An acceleration of Newton's method: Super-Halley method, *Appl. Math. Comput.* **117** (2001), no. 2, 223–239.

[7] R. Hongmin, W. Qingbiao y B. Weihong, A class of two-step Steffensen type methods with fourth-order convergence, *Appl. Math. Comput.* **209** (2009), no. 2, 206–210.

[8] D. Jeffrey, G. Kalugin y N. Murdoch, Lagrange inversion and Lambert W, *Proceedings of the 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)* (D. Zaharie, L. Kovacs y V. Negru, eds.), 42–46, IEEE Computer Society, 2015.

[9] B. Kalantari, *Polynomial root-finding and polynomiography*, World Scientific, 2008.

[10] A. B. Kasturiarachi, Leap-frogging Newton's method, *Internat. J. Math. Ed. Sci. Tech.* **33** (2002), no. 4, 521–527.

[11] H. T. Kung y J. F. Traub, Optimal order of one-point and multipoint iteration, *J. Assoc. Comput. Mach.* **21** (1974), no. 4, 643–651.

[12] A. Li y R. M. Corless, Revisiting Gilbert Strang's "a chaotic search for i", *ACM Commun. Comput. Algebra* **53** (2019), no. 1, 1–22.

[13] J. M. McNamee, *Numerical methods for roots of polynomials*, Studies in Computational Mathematics **14**, Elsevier, Amsterdam, 2007.

[14] A. Neumaier, *Introduction to numerical analysis*, Cambridge University Press, Cambridge, 2001.

[15] M. S. Petković, J. Džunić y L. D. Petković, A family of two-point methods with memory for solving nonlinear equations, *App. Anal. Discrete Math.* **5** (2011), no. 2, 298–317.

[16] M. S. Petković, L. D. Petković y D. Herceg, On Schröder's families of root-finding methods, *J. Comput. Appl. Math.* **233** (2010), no. 8, 1755–1762.

[17] G. Wilkins y M. Gu, A modified Brent's method for finding zeros of functions, *Numer. Math.* **123** (2013), no. 1, 177–188.

R. M. Corless, Ontario Research Centre for Computer Algebra y The Rotman Institute of Philosophy, Western University, London, Ontario, Canadá

Correo electrónico: rcorless@uwo.ca

Página web: https://www.rotman.uwo.ca/portfolio-items/corless/