Universidad de Costa Rica
Escuela de Matemáticas
MA-501 Análisis Numérico
Tarea 1

Juan Ignacio Padilla B.
Carné: B55272
Prof. Juan Gabriel Calvo
28 de enero de 2026

Tarea 1

Problema 1.

Dado un vector $\boldsymbol{v} \in \mathbb{R}^{n+1}$, el comando

```
c=poly(v)
```

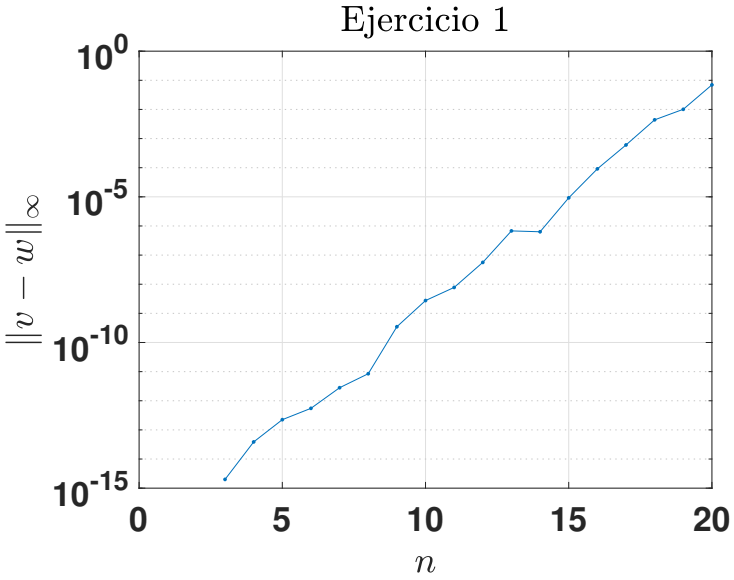
construye un vector $\boldsymbol{c} \in \mathbb{R}^{n+1}$ cuyas entradas son los coeficientes del polinomio mónico

$$p(x) = \prod_{j=1}^{n}(x - v_j),$$

de forma que sus ceros son las entradas del vector $\boldsymbol{v}$. Si resolvemos la ecuación $p(x) = 0$, sus soluciones deberían ser precisamente las entradas de $\boldsymbol{v}$.

a. En `MATLAB`, se calcula de forma numérica el vector $\boldsymbol{w}$ que corresponde a las raíces de p. Luego se grafica el error $||\boldsymbol{v} - \boldsymbol{w}||_\infty$ en función de n, para el vector $\boldsymbol{v} = (1, 2, 3, \ldots, n)^T$. **Explique por qué el error aumenta conforme n crece.**

Solución: Para $n \in \{1, 2, \ldots, 20\}$ tenemos el error graficado por el código facilitado por el profesor



Para los primeros valores de n, el error es casi negligible, de 10^{-15}. Sin embargo, nótese que conforme n crece, el error pareciera crecer de manera lineal, hasta llegar a un error de casi 0.1, para $n = 20$. Debido a que no conocemos la naturaleza de los comandos `roots` ni `poly`, la explicación más lógica de este fenómeno es que se trate de algún defecto de redondeo. Tenemos que analizar más de cerca nuestro polinomio p.

Obsérvese que para $n = 1$, tenemos que $\boldsymbol{v} = (1)$, y entonces

$$p(x) = x - 1$$

y entonces al usar el comando `poly` obtenemos

```
poly(v) = [1 - 1]
```

Sin embargo, cuando $n = 20$, se tiene que

$$p(x) = (x-1)(x-2)\cdots(x-20) = x^{20} - 210x^{19} + \cdots + 20!$$

Donde $20! \approx 2,45 \times 10^{18}$. Entonces, tenemos un polinomio cuyos coeficientes varían enormemente en orden de magnitud. Al aplicar `poly` para $n = 20$, tenemos que

```
poly(v) =
1.0e+19 *
  Columns 1 through 6
    0.0000   -0.0000    0.0000   -0.0000    0.0000   -0.0000
  Columns 7 through 12
    0.0000   -0.0000    0.0000   -0.0000    0.0001   -0.0010
  Columns 13 through 18
    0.0063   -0.0311    0.1207   -0.3600    0.8038   -1.2871
  Columns 19 through 21
    1.3804   -0.8753    0.2433
```

Lo cual confirma el hecho de que los órdenes de magnitud de los coeficientes de $p(x)$ son muy dispares. Esto nos llevará a pensar que cualquier evaluación numérica que el comando `roots` lleve a cabo, está sujeta a errores de redondeo, ya que al evaluar $p(x)$ en cualquier número, generará una suma de valores con magnitudes diferentes, lo cual producirá inevitablemente errores de redondeo. Esto se ve incluso más evidenciado al calcular $p(1) \approx 1024$, lo cual es completamente impreciso, pues es obvio que $p(1) = 0$.

b. Fije $n = 20$ y calcule `w = roots(poly(1:n))`. Utilice el método de Newton para mejorar la aproximación de cada entrada de $\boldsymbol{w}$, utilizando como valor inicial w_j para $j \in \{1, 2, \ldots, n\}$. ¿Es cada nueva aproximación más precisa? Justifique su respuesta.
Solución: Al calcular `w = roots(poly(1:n))`, obtenemos

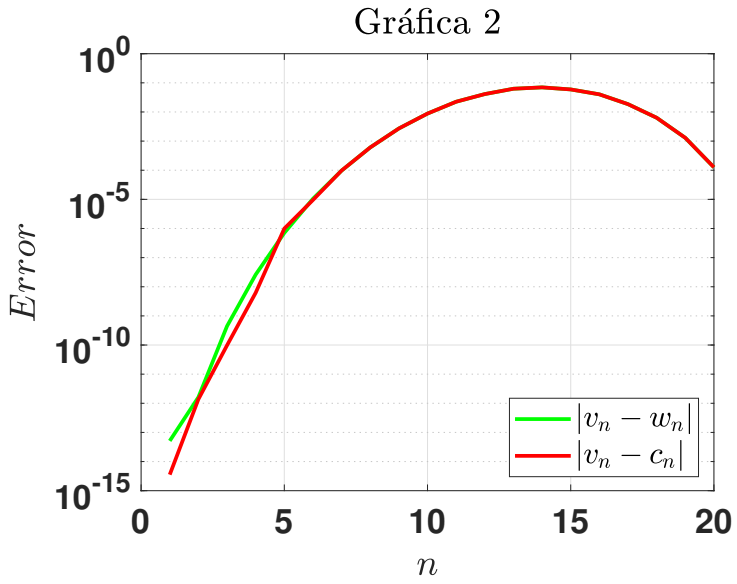
```
w=
0.999999999999949
1.999999999998383
3.000000000444877
3.999999973862455
5.000000705531480
5.999989523351082
7.000096952230211
7.999394310958664
9.002712743189727
```

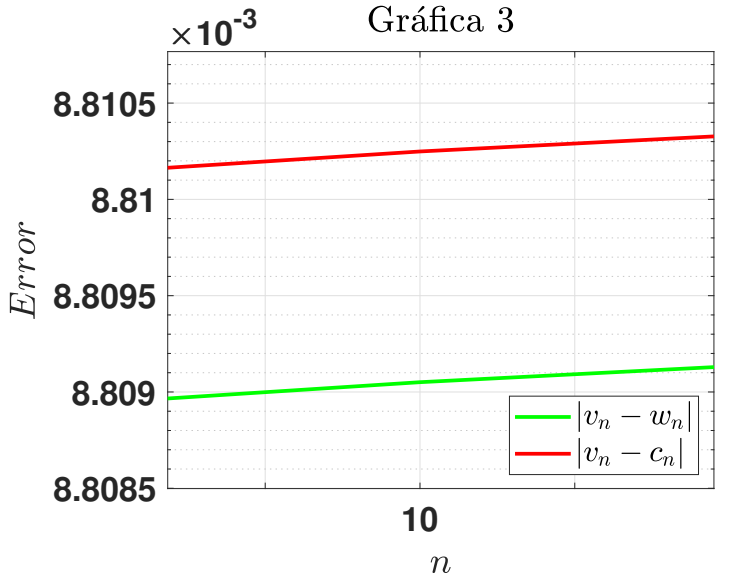
```
9.991190949230132
11.022464271003383
11.958873995343460
13.062663652011070
13.930186454760916
15.059326234074415
15.959717574548915
17.018541647321989
17.993671562737585
19.001295393676987
19.999874055724192
```

Y al calcular el vector $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ en donde c_j es la aproximación de las raíces de $p(x)$, usando el método de Newton, con valor inicial w_j, obtenemos

```
c=
1.000000000000004
1.999999999998623
2.999999999902548
3.999999993793522
5.000000993489359
5.999990488567676
7.000097488971716
7.999394052477573
9.002712561395008
9.991189751268259
11.022464631556501
11.958873639788711
13.062663808423107
13.930186706646897
15.059321001539574
15.959717676390879
17.018541443827338
17.993671604299941
19.001295472149295
19.999873774976844
```

Para determinar si nuestra aproximación es mejor o peor que la de `MATLAB`, creamos la gráfica 2. La línea verde corresponde a la aproximación `w = roots(poly(1:n))`, mientras que la roja corresponde con la aplicación de Newton $\boldsymbol{c}$, tomando como valores iniciales a los w_j. Como se puede observar a simple vista, entre 1 y 5 la calidad de la aproximación fluctúa ligeramente, inicialmente siendo mejor nuestra aproximación, y luego la preestablecida por el programa. Despues de $n = 5$, parecieran ser aproximaciones similares. Sin embargo, al ampliar la imagen en la gráfica 3, se puede observar que la aproximación original es marginalmente mejor en $n = 10$ puesto que está más cerca del vector $\boldsymbol{v}$ original. Esto se puede deber una vez más a errores de redondeo, puesto que la derivada de $p(x)$ tiene coeficientes de orden alto, y al evaluarse en un número pequeño como 10, obviará algunos valores. Además, razonando por otro lado, es inverosímil pensar que un código sencillo y sin optimizar pueda presentar mejores aproximaciones que el código interno de `MATLAB`, por lo cual sería de esperar que la solución `w = roots(poly(1:n))` sea más cercana a $\boldsymbol{v}$.





Problema 2.

Sea $c > 0$. Demuestre que la iteración

$$c_{k+1} = \frac{1}{2}\left(c_k + \frac{c}{c_k}\right)$$

converge cuadráticamente a $\sqrt{c}$ para cualquier $c_0 > 0$.
Solución: Primero, es claro que $c_k > 0$ para todo k. Note que

$$\begin{aligned} c_{k+1}^2 &= \frac{1}{4}\left(c_k + \frac{c}{c_k}\right)^2 \\ &= \frac{c_k^2}{4} + \frac{c}{2} + \frac{c^2}{4c_k^2} \\ &= \frac{1}{4}\left(c_k - \frac{c}{c_k}\right)^2 + c \geq c \end{aligned}$$

Entonces $c_k \geq \sqrt{c}$ para todo $k > 0$. Entonces podemos asumir *s.p.g.* que $c_0 > \sqrt{c}$ (del contrario podemos empezar la sucesión en c_1). Veamos ahora que

$$\begin{aligned} c_{k+1} - c_k &= \frac{c_k}{2} + \frac{c}{2c_k} - c_k \\ &= \frac{c - c_k^2}{2c_k} \leq 0 \end{aligned}$$

Tenemos entonces que $\{c_k\}$ es una sucesión decreciente, y acotada inferiormente. Por lo tanto converge. Sea además $L = \lim_{k\to\infty} c_k$. Debe cumplirse que

$$\begin{aligned} L = \frac{L}{2} + \frac{c}{2L} &\iff L^2 = \frac{L^2}{2} + \frac{c}{2} \\ &\iff L^2 = c \\ &\iff L = \sqrt{c} \end{aligned}$$

Por lo tanto tenemos que la sucesión converge a $\sqrt{c}$. Para ver el orden de convergencia, considere

$$\begin{aligned} \lim_{k\to\infty} \frac{|c_{k+1} - \sqrt{c}|}{(c_k - \sqrt{c})^2} &= \lim_{k\to\infty} \frac{\frac{1}{2}\left(c_k + \frac{c}{c_k}\right) - \sqrt{c}}{(c_k - \sqrt{c})^2} \\ &= \lim_{k\to\infty} \frac{c_k^2 + c - 2c_k\sqrt{c}}{2c_k(c_k^2 - 2\sqrt{c}c_k + c)} \\ &= \lim_{k\to\infty} \frac{1}{2c_k} \\ &= \frac{1}{2\sqrt{c}} \end{aligned}$$

Entonces concluímos que la sucesión converge con orden cuadrático.

Problema 3.

(Método de Newton para raíces de multiplicidad m). Considere $f \in C^{m+2}([a,b])$ [1], con una raíz $c \in (a,b)$ de multiplicidad m. Esto es

$$f(c) = f'(c) = \cdots = f^{(m-1)}(c) = 0 \ , \ f^{(m)}(c) \neq 0$$

a. Demuestre que existe $\delta > 0$ suficientemente pequeño, tal que si $c_0 \in (c - \delta, c + \delta)$, entonces la sucesión

$$c_{k+1} = c_k - m\frac{f(c_k)}{f'(c_k)}$$

converge cuadráticamente a c.

Solución: Para probar la convergencia, note que si definimos [2]

$$g(x) = x - m\frac{f(x)}{f'(x)}$$

[1] Esta hipótesis no es del enunciado oficial, pero se me permitió asumir un grado más de suavidad.

[2] Note que a priori no sabemos si g se indefine cuando $f'(c) = 0$, pues los ceros del cociente podrían "cancelarse"

Vemos que los puntos fijos de g coinciden con los ceros de f. Si c es una raíz de f, de multiplicidad m, podemos escribir

$$f(x) = (x-c)^m h(x)$$

En donde $h(c) \neq 0$. Ahora

$$\begin{aligned} g(x) &= x - \frac{m(x-c)^m h(x)}{m(x-c)^{m-1}h(x) + (x-c)^m h'(x)} \\ \Rightarrow g(x) &= x - \frac{m(x-c)h(x)}{mh(x) + (x-c)h(x)} \end{aligned}$$

Por lo cual g está bien definida. Ahora estudiamos la situación más a fondo. Tenemos

$$\begin{aligned} & & c_{k+1} &= c_k - m\frac{f(c_k)}{f'(c_k)} \\ \Rightarrow & & c - c_{k+1} &= c - c_k + m\frac{f(c_k)}{f'(c_k)} \\ \Rightarrow & & (c-c_{k+1})f'(c_k) &= f'(c_k)(c-c_k) + mf(c_k). \end{aligned}$$

Si llamamos $G(x) = f'(x)(c-x) + mf(x)$, tenemos

$$(c-c_{k+1})f'(c_k) = G(c_k). \qquad (\star)$$

Observe ahora que para $i = 1, \ldots, m$,

$$\begin{aligned} & G'(c_k) = mf'(c_k) + f''(c_k)(c-c_k) - f'(c_k) \\ \Rightarrow \quad & G''(c_k) = mf''(c_k) + f'''(c_k)(c-c_k) - 2f''(c_k) \\ & \vdots \\ \Rightarrow \quad & G^{(i)}(c_k) = mf^{(i)}(c_k) + f^{(i+1)}(c_k)(c-c_k) - if^{(i)}(c_k). \end{aligned}$$

Como c es raíz de f de multiplicidad m, tenemos que

$$G(c) = G'(c) = \cdots = G^{(m-1)}(c) = 0$$

y además $G^{(m)}(c) = mf^{(m)}(c) - mf^{(m)}(c) = 0$. Ahora, por Taylor vemos que

$$G(x) = \frac{G^{(m+1)}(\xi_1)}{(m+1)!}(x-c)^{m+1}$$

Donde ξ_1 está entre x y c. También tenemos

$$f'(x) = \frac{f^{(m)}(\xi_2)}{(m-1)!}(x-c)^{m-1}$$

Donde ξ_2 está entre x y c. Ahora, por $(\star)$, obtenemos

$$\begin{aligned} (c-c_{k+1})\frac{f^{(m)}(\xi_2)}{(m-1)!}(c_k-c)^{m-1} &= \frac{G^{(m+1)}(\xi_1)}{(m+1)!}(c_k-c)^{m+1} \\ \Rightarrow \qquad c - c_{k+1} &= \frac{G^{(m+1)}(\xi_1)}{m(m+1)f^{(m)}(\xi_2)}(c_k-c)^2 \end{aligned}$$

Como $f^{(m)}(c) \neq 0$, es posible escoger r de tal manera que en $I_r = [c-r, c+r]$, se tenga que $f^{(m)}(x) \neq 0$. Tome $M = \text{máx}_{x,y \in I_r} \left| \frac{G^{(m+1)}(x)}{(m+1)f^{(m)}(y)} \right|$, y se obtiene

$$|c - c_{k+1}| \leq \frac{M}{m}(c - c_k)^2$$

Por lo tanto, si tomamos $\delta = \text{mín}\{1/M, r\}$, se cumplirá que $c_k \in I_\delta$ (por inducción), pues se cumpliría que $|c - c_{k+1}| \leq |c - c_k|$. Finalmente, en I_δ

$$\begin{aligned} |c - c_{k+1}| &\leq \frac{1}{m}|c_k - c| \\ &\leq \frac{1}{m^2}|c_{k-1} - c| \\ &\vdots \\ &\leq \frac{1}{m^{k+1}}|c_0 - c| \xrightarrow[k\to\infty]{} 0. \end{aligned}$$

Esto garantiza la convergencia puntual. Además, nos aseguramos que $M < \infty$, pues el denominador $f^{(m)}(y)$ no se anulará en I_δ. Por lo tanto:

$$\lim_{k\to\infty} \frac{|c - c_{k+1}|}{(c - c_k)^2} \leq \frac{M}{m}$$

tenemos convergencia de orden cuadrático.

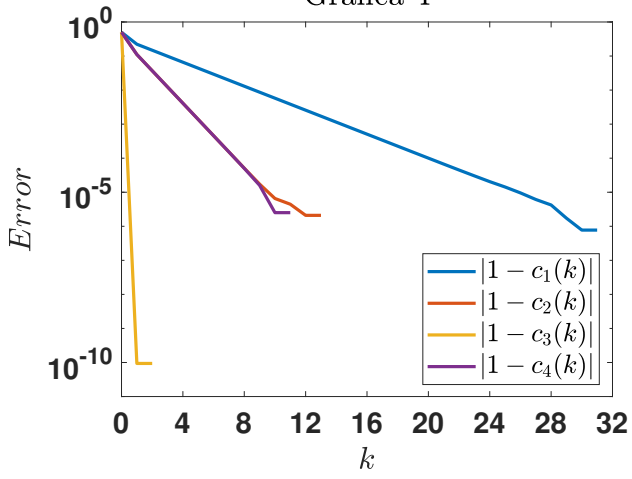
b. Implemente una función en `MATLAB` que incluya como entradas a la función f, el valor inicial c_0, un entero m que represente el orden del cero, y una tolerancia ε. Considere $|c_k - c_{k+1}|$ como condición para detener la iteración.

Solución: Se implementó la función `NewtonMulti`, adjuntada en los archivos.

c. Verifique el comportamiento del método para la función $g(x) = x^3 - 3x^2 + 3x - 1$, con $x_0 = 1/2$, $\varepsilon = 10^{-8}$ y $m \in \{1, 2, 3, 4\}$. Grafique el error en función de k. Comente sus resultados.

Solución: Se aplicó la función `NewtonMulti`, con los parámetros requeridos, y se obtuvo la gráfica 4.

Gráfica 4



En donde el vector $c_m(k)$ representa la $k-$ésima iteración de la solución a la ecuación $f(x) = 0$, usando la función `NewtonMulti` con multiplicidad m. Puede observarse que la peor velocidad se obtuvo con $m = 1$, pues necesitó cerca de 30 iteraciones para ajustarse a la tolerancia requerida. Mientras tanto, los casos $m = 2, 4$ parecen compartir velocidades parecidas, y un error similar al de $m = 1$. Sin embargo, nótese que el caso $m = 3$ presentó una velocidad alta, ajustándose a la tolerancia en tan sólo 3 iteraciones, y además alcanzando una precisión muchos ordenes mayor a los demás. Esto se debe claramente a que

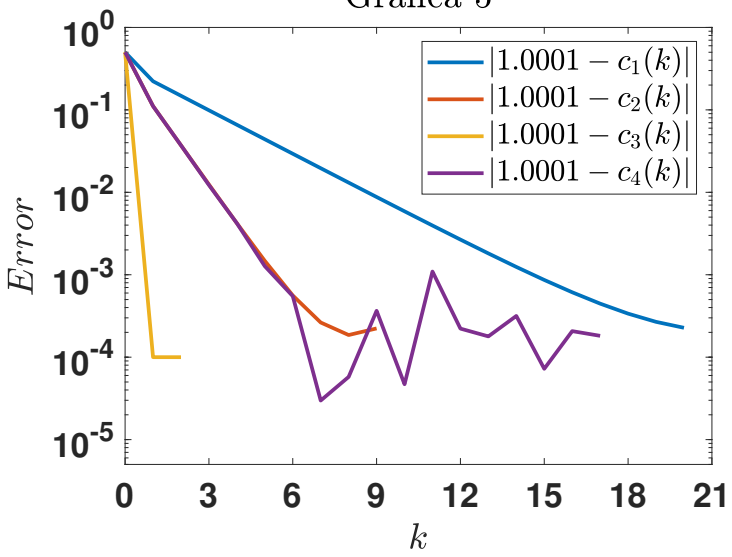
$$f(x) = (x-1)^3$$

tiene un cero de multiplicidad 3, y como se demostró en el punto a), convergerá con velocidad cuadrática a la raíz, mientras que los otros métodos lo harán de manera lineal.

d. Repita el inciso anterior para $h(x) = (x-1)(x-1{,}0001)(x-0{,}9999)$ ¿Numéricamente, qué multiplicidad tiene el mayor cero?

Solución: Una vez más, se aplicó la función `NewtonMulti` a la función $h(x)$, con las multiplicidades $m = 1, 2, 3, 4$, y se graficó el error obtenido en la gráfica 5, con respecto a la solución $c = 1{,}0001$.

Gráfica 5



Hay que observar que se dieron fluctuaciones en la convergencia del método, esto se pudo deber a defectos de redondeo, ya que las tres raíces del polinomio están cerca, y sus intervalos de atracción podrían traslaparse a la hora de evaluar numéricamente. Haciendo más evaluaciones numéricas, se estimó que $h'(1) \approx -9{,}9990 \times 10^{-9} \approx 0$, lo cual es un valor perjudicial para el método de Newton, ya que desde un punto de vista geométrico, las rectas tangentes a un punto con derivada cercana a 0, tienden a intersecar el eje x muy lejos de donde se está trabajando. De hecho, para este insciso, fue necesario relajar la tolerancia a un valor de 10^{-5} ya que del contrario, se daban oscilaciones infinitas (se puede corroborar corriendo el código adjuntado para este ejercicio).

Por otra parte, ocurre una vez más que para el caso $m = 3$, se tiene una velocidad de convergencia mucho mayor a las otras dos, lo cual sugiere que numéricamente 1.0001 **pareciera ser una raíz triple**, lo cual sabemos que es incorrecto. Este hecho se puede

atribuir una vez más a conflictos de redondeo, y al posicionamiento de los intervalos de atracción de las raíces de $h(x)$.

Problema 4.

(Método de la Regla Falsa) Considere una función continua f definida en el intervalo inicial $I_0 = [a_0, b_0]$, donde $f(a_0)f(b_0) < 0$. En la iteración k, dado el intervalo $I_k = [a_k, b_k]$, calcule

$$c_k = \frac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)}$$

Luego evalúe $f(c_k)$. Si $f(a_k)f(c_k) < 0$, tome $I_{k+1} = [a_k, c_k]$; en caso contrario, tome $I_{k+1} = [c_k, b_k]$.

a) Explique el significado geométrico del método, así como su conexión con el método de la Secante y de Bisección.

Solución: Vamos a trabajar solo con la primera iteración. Vamos a probar que

$$c = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

Es precisamente la coordenada x de la interseción de la recta secante que pasa por $(a, f(a)), (b, f(b))$, y el eje x. Considere dicha recta secante, es fácil ver que su ecuación viene dada por

$$y = \frac{f(a) - f(b)}{a - b}x + \frac{af(b) - bf(a)}{a - b}$$

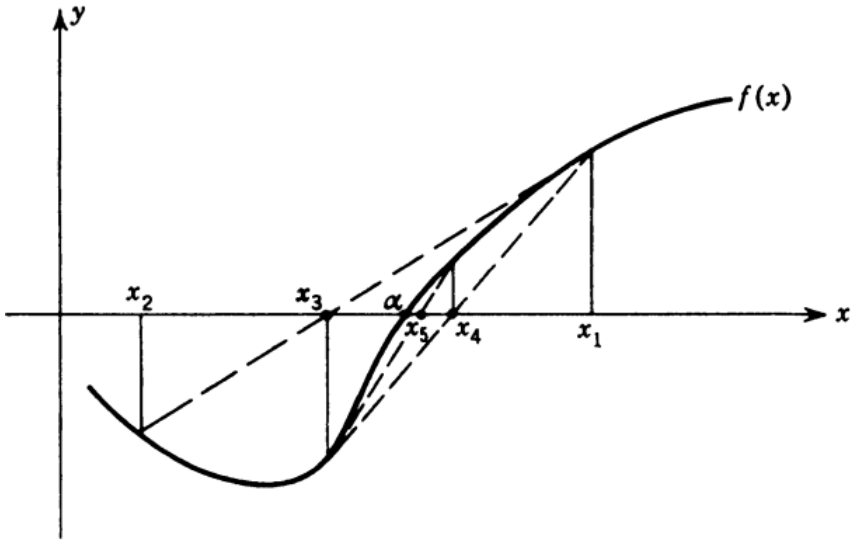
Y resolviendo la ecuación $y = 0$, obtenemos

$$x = \frac{-af(b) + bf(a)}{a - b}\frac{a - b}{f(a) - f(b)} = \frac{af(b) - bf(a)}{f(b) - f(a)} = c$$

El cual es el valor que se deseaba. La segunda parte del algoritmo, consiste en revisar y seleccionar en cuál de los dos intervalos $[a, c]$ ó $[c, b]$ se encuentra nuestra raíz. Luego se repite el algoritmo. Esto asegura que la raíz que buscamos siempre se encuentre en nuestro intervalo de trabajo, algo que no sucede en el método de la Secante.

La relación entre este método y los métodos de la secante y de bisección es bastante directa. El método simplemente consiste primero en aplicar exactamente la misma iteración de un método de la secante, sin embargo, antes de iniciar la segunda iteración, se realiza un chequeo extra, que consiste en la técnica usada en el método de bisección de $f(a)f(c) < 0$ ó $f(c)f(b) < 0$, para estrechar nuestro intervalo de trabajo, lo cual mejorará (posiblemente) la estabilidad del método. La siguiente imagen tomada de [1] representa las primeras iteraciones de éste método.

Figura 6. Método de falsa posición



Vemos que precisamente, x_3 es el punto de intersección de la recta secante. Luego, en vez de trazar la recta secante entre x_2 y x_3, se traza entre x_3 y x_1, pues es el intervalo que satisface las hipótesis de la bisección. Así sucesivamente se calculan x_4 y x_5, las cuales parecen acercarse a la raíz α.

b) Escriba una función en `MATLAB` que implemente el método de la regla falsa. Debe tener como entradas la función f, el intervalo inicial $I_0 = [a_0, b_0]$, y la tolerancia ε. Considere $|c_k - c_{k+1}|$ como condición para detener la iteración

Solución: Se implementó la función `ReglaFalsa`, adjuntada en los archivos.

c) Compruebe el comportamiento de su código con la función $f(x) = 1/x - \sin x + 1$, con $a_0 = -1{,}3, b_0 = -0{,}5$. Numéricamente, ¿Cuál es el orden de convergencia?

Solución: Al aplicar la función `ReglaFalsa` a la función f, con una tolerancia de 10^{-8}, se obtiene un valor aproximado de $c = -0{,}629446484073333$. Al calcular numéricamente el cociente

$$L = \left| \frac{c - c_k}{c - c_{k-1}} \right|$$

para el número de iteraciones obtenidas ($k = 22$), y tomando $c = c_{22}$ Se tiene que

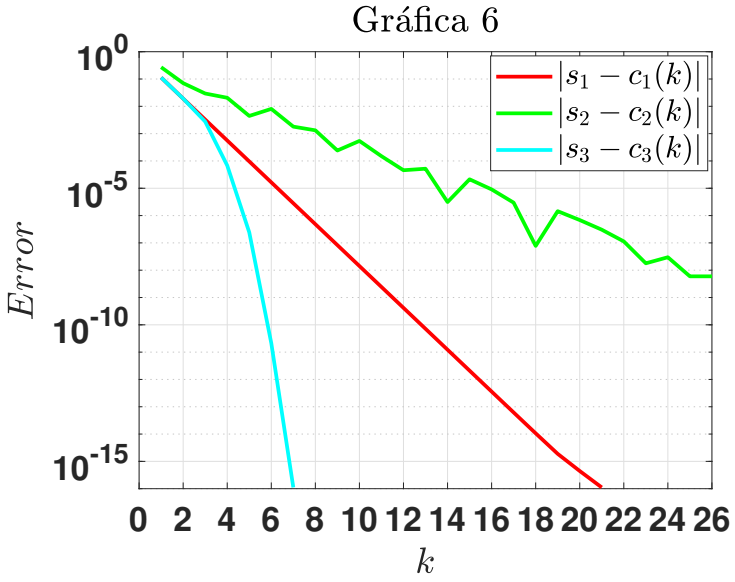
$$L = 0{,}250000000$$

Lo cual indica que, al menos numéricamente, se tiene un orden de convergencia lineal.

d) Compare la razón de convergencia para la función del inciso anterior al utilizar los métodos de Bisección y Secante.¿El método de Regla Falsa tiene alguna ventaja sobre los otros?.

Solución: Se aplicaron los métodos de Regla Falsa, Secante, y Bisección a la función $f(x) = 1/x - \sin x + 1$, y se compararon los resultados gráficamente en función de las iteraciones [3]. Se resumen los resultados en la gráfica 6.

[3] Para el método de Secante, se usó una tolerancia ligeramente menor, para tener suficientes iteraciones con qué comparar.



La curva verde corresponde con el método de Bisección, la roja con el método de falsa posición, y la cyan con el método de la secante. En la leyenda, el valor s es el valor final de cada método, es decir, el último valor que registró antes de detenerse. Tenemos entonces que el método de bisección se acerca lentamente a su valor final, con fluctuaciones. Los otros métodos se acercan con relativa suavidad, sin embargo, es evidente que el método de la secante es el más rápido de los 3 en este caso. Recordemos que numéricamente, el orden de convergencia de la regla falsa fue de 1, mientras que el orden de convergencia teórico de la secante es de ϕ (el número aúreo). En este caso, el método de la Regla Falsa no es el óptimo (sin embargo es mejor que el de bisección). En general, según [1], la Regla Falsa, si bien suele ser más lento que la Secante, tiende a ser más estable, pues siempre converge.

Problema 5.

(Enfriamiento de un cuerpo) La Ley de Enfriamiento de Newton dice que la razón de cambio de temperatura en el tiempo t de un objeto es proporcional a la diferencia entre la temperatura ambiente T_a y la temperatura T del objeto. Esto es

$$\frac{dT}{dt} = k(T_a - T)$$

donde k es la constante de proporcionalidad.

a) Asuma que la temperatura ambiente es constante.Verifique que la temperatura del objeto en el tiempo t viene dada por

$$T(t) = T_a + (T_0 - T_a)e^{-kt}$$

donde $T_0 = T(0)$ es la temperatura inicial del objeto.

Solución: Tenemos la ecuación diferencial ordinaria

$$dT = k(T_a - T)dt$$

La cual resolvemos por separación de variables.

$$\begin{aligned} kt + C &= \int \frac{dT}{(T_a - T)} \\ \Rightarrow \quad kt + C &= -\log(T_a - T) \\ \Rightarrow \quad -kt + C &= \log(T_a - T) \\ \Rightarrow \quad Ce^{-kt} &= T_a - T \\ \Rightarrow \quad T(t) &= T_a + Ce^{-kt} \end{aligned}$$

Pero sabemos que $T(0) = T_0$, por lo que

$$\begin{aligned} \Rightarrow \quad T_0 &= T_a + C \\ \Rightarrow \quad C &= T_0 - T_a \\ \therefore \quad T(t) &= T_a + (T_0 - T_a)e^{-kt} \end{aligned}$$

b) Un médico forense desea estimar el valor de k para poder determinar el tiempo de muerte de cadáveres. Para ello, realiza diferentes mediciones, obteniendo los siguientes datos:

t_i (horas)	$T_i(°C)$
0,0	37, 00
0,2	36,72
0,4	36,41
0,6	36,12
0,8	35,90

Además, la temperatura ambiente de ese día es de $21°C$. Para estimar k, el médico considera minimizar el funcional

$$f(k) = \sum_i (T(t_i) - T_i)^2.$$

Aproxime el punto crítico de f y verifique que f alcanza un mínimo.

Solución: Para calcular los puntos críticos de f, podemos calcular explícitamente su derivada,

$$f(k) = \sum_i (T_a + (T_0 - T_a)e^{-kt} - T_i)^2$$

$$\Rightarrow \quad f'(k) = -2(T_0 - T_a) \sum_i t_i e^{-kt_i}(T_a + (T_0 - T_a)e^{-kt_i} - T_i)$$

En `MATLAB` calculamos numéricamente los valores $f'(0) \approx -54{,}4$, y $f'(1) \approx 209{,}4174$. Entonces por el teorema del valor intermedio, tenemos un punto crítico en $(0, 1)$. Luego se aplicó el método de la secante, con una tolerancia de 10^{-8}, para calcular el punto crítico, el cual se aproximó a $k \approx 0{,}091283148291059$. Para verificar que se trata de un mínimo, se calculó numéricamente $f''(k) \approx 544$. Por lo tanto estamos en presencia de un mínimo, y podemos decir que la constante de aproximación se acerca a nuestro valor calculado.

c) Un día caluroso en San Pedro a $31°C$ una persona falleció. Al llegar tarde al lugar de los hechos, se determina que la temperatura del cuerpo es de $34°C$. ¿Hace cuánto tiempo falleción la persona? (tome $34°C$ como la temperatura promedio del cuerpo humano, y k como el valor estimado del inciso anterior).

Solución: Solo tenemos que resolver la ecuación

$$T(t) = T_a + (T_0 - T_a)e^{-kt} - 34 = 0$$

Lo cual se hace numéricamente, usando una vez más el método de la secante. Como se conocen todos los parámetros, basta con despejar t numéricamente, de lo cual se obtiene que $t \approx 7{,}593375048260015$. Es decir, la persona falleció hace aproximadamente 7 horas y 35 minutos.

Referencias

[1] A.Ralston, P. Rabinowitz *A First Course in Numerical Analysis.* Segunda Edición. Dover Publications. New York. 1978