Análisis numérico

Clase 6: Preliminares, algoritmos y computación

Joaquin Cavieres

Instituto de Estadística, Universidad de Valparaíso

Outline

Dentro del curso, la palabra "algoritmo" se repetirá constantemente, pero ¿qué es un algoritmo?

Definición: Algoritmo

Proceso para resolver problemas matemáticos en un número fínito de pasos.

Objetivo: Resolver o aproximar la solución de un problema.

Algoritmos y computación

Para la descripción de un algoritmo generalmente utilizamos *pseudocódigos*. El pseudocódigo describe los valores de entrada (datos) que son utilizados por el algoritmo y la forma en que queremos representar los resultados.

No todos los procesos (algoritmos) entregan un resultado satisfactorio dados valores de entrada arbitrarios, por lo tanto, debemos introducir técnicas numéricas de "parada" para así evitar ciclos infinitos.

Por lo general tenemos dos símbolos de puntuación en los algoritmos:

- Punto (.) $\rightarrow$ Indica el punto final de un paso.
- Punto y coma (;) $\rightarrow$ Separación de una instrucción dentro de un paso.

Los ciclos dentro de los algoritmos son controlados por un *contador*, como por ejemplo:

$$\begin{aligned} \text{Para } i &= 1, 2, \ldots.., n \\ \text{Hacer } x_i &= a + i * h \end{aligned}$$

o mediante "condiciones",

$$\text{Mientras } i < n \text{ hacer los pasos 1 a 3}$$

Las ejecuciones condicionales son declaradas como:

Si...entonces

Si...entonces si no

Las instrucciones en los pseudocódigos se ordenan de tal forma que es sencillo traspasarlas a cualquier lenguaje de programación para aplicaciones numéricas.

El siguiente algoritmo escrito en pseudocódigo calcula $x_1 + x_2 + ... + x_n = \sum_{i=1}^{n} x_i$ mediante los números $x_1, x_2, ..., x_n$ y un número total de n ([1]).

1) Input $n, x_1, x_2, ..., x_n$

2) Result $\text{sum} = \sum_{i=1}^{n} x_i.$

3) Step 1 considere $\text{sum} = 0.$ (inicializar el acumulador)

4) Step 2 For $i = 1, 2, ..., n$ do

$\text{sum} = \text{sum} + x_i.$ Sumar el termino anterior

5) Step 3 Result(sum);

Finish;

Ejemplo 1: El n-ésimo polinomio de Taylor para $f(x) = ln(x)$ ampliado alrededor de $x_0 = 1$ es:

$$P_n = \sum_{i=1}^{n} \frac{(-1)^{i+1}}{i}(x-1)^i$$

y el valor de $ln 1.5$ para ocho decimales es de 0.40546511. Construya un algoritmo con el fin de encontrar el valor mínimo de n requerido para:

$$|ln 1.5 - P_n(1.5)| < 10^{-5}$$

sin utilizar el término restante del polinomio de Taylor.

Solución: Sabiendo que $\sum_{n=1}^{\infty} a_n$ es una serie alterna con el límite a A y cuyos términos disminuyen en magnitud, entonces A y la $n-sima$ suma parcial $A_N = \sum_{n=1}^{N} a_n$ difieren por menos magnitud del término $(N+1)$, es decir:

$$|A - A_N| \leq |a_N + 1|$$

El siguiente algoritmo desarrolla lo expuesto anteriormente:

Algoritmos y computación

1) Entradas x, TOL, M
2) Resultados grado N del polinomio o mensaje de error
3) Paso 1 $N = 1;$
   $y = x - 1;$
   $sum = 0;$
   $term = y;$
   $sign = -1.$ (necesario para la alternancia de signos)
4) Paso 2 Mientras $N \leq M$ hacer pasos 3 - 5
5) Paso 3 Determine sign = -sing; alternancia de signos
   $sum = sum + sign + term;$ (acumulación de los términos)
6) Paso 4 Si$|\text{term}| < \text{TOL}$ entonces
   Resultado(N);
   Terminar(N); (los calculos fueron realizados correctamente)
7) Paso 5 Determinar$N = N + 1.$ (Siguiente iteración (fin Paso 2))
8) Paso 6 Resultados ("El método falló");
   Terminar(N); (los calculos no fueron realizados correctamente)

Los valores de entrada en este caso particular son $x = 1.5$, $TOL = 10^{-5}$ y $M = 15$. La salida (Resultado) es un valor para N o un mensaje de fallo por parte del método dependiendo de la precisión computacional.

Algoritmos de caracterización

Debido a que vamos a trabajar una amplia gama de problemas de aproximación durante el curso, necesitamos condiciones para clasificar la precisión de cada uno de ellos, ya que no todas estas aproximaciones producen resultados fiables a la generelidad de nuestros problemas.

Es necesario fijar un criterio en los algoritmos (en la medida de lo posible) para que, pequeños cambios en los datos iniciales, produzcan en forma proporcional pequeños cambios en los resultados finales

Algoritmos de caracterización

Si un algoritmo satisface lo anterior entonces se le llama estable, de lo contrario se le llama inestable, y en algunas ocaciones se presentan los algoritmos llamados estables condicionalmente.

Suponga que tenemos $E_0 > 0$ en alguna de las etapas del algoritmo, para luego, al final del proceso tener un error de E_n. De lo anterior podemos tener dos casos:

Definicion 1

$E_0 > 0$ denota un error que se presenta en alguna etapa y E_n representa la magnitud del error despues de n operaciones.

- Si $E_n \approx C * nE_0$, con C constante indpte de n, entonces se dice que el crecimiento del error es **lineal**.
- Si $E_n \approx C^n E_0$, para algunos $C > 1$, entonces se dice que el crecimiento del error es **exponencial**.

Algoritmos de caracterización

El crecimiento lineal casi es inevitable, y con C y E_0 pequeños, los resultados generalmente son aceptables. Por el contrario, el crecimiento exponencial del error se debe evitar por que el término C^n se vuelve grande incluso para los valores relativamente pequeños de n, por lo que nos llevaría a impresiciones independientemente del tamaño de E_0.

Un algoritmo con crecimiento lineal del error se considera estable, mientras que un algoritmo con crecimiento exponencial del error es inestable.

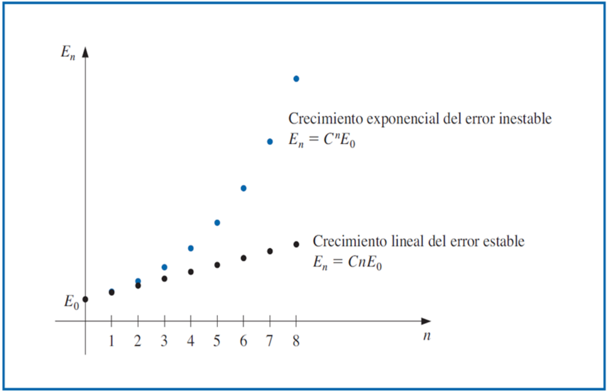

Figure: Ejemplo error lineal y exponencial. Fuente: [1]

Ilustración: Para cualquier constante c_1 y c_2, tenemos que:

$$p_n = c_1 \left(\frac{1}{3}\right)^n + c_2 3^n, \tag{1}$$

es una solución a la ecuación recursiva:

$$p_n = \left(\frac{10}{3}\right) p_{n-1} - p_{n-2}, \quad \text{para} \quad n = 2, 3, \ldots$$

Algoritmos y computación

Podemos ver que:

$$
\begin{aligned}
\left(\frac{10}{3}\right)p_{n-1} - p_{n-2} &= \left(\frac{10}{3}\right)\left[c_1\left(\frac{1}{3}\right)^{n-1}\right] - \left[c_1\left(\frac{1}{3}\right)^{n-1} + c_2 3^{n-2}\right] \\
&= c_1\left(\frac{1}{3}\right)^{n-2}\left[\frac{10}{3} * \frac{1}{3} - 1\right] + c_2 3^{n-2}\left[\frac{10}{3} * 3 - 1\right] \\
&= c_1\left(\frac{1}{3}\right)^{n-2}\left(\frac{1}{9}\right) + c_2 3^{n-2}(9) \\
&= c_1\left(\frac{1}{3}^{n}\right) + c_2 3^{n} \\
&= p_n
\end{aligned}
$$

Si nosotros asumimos un $p_0 = 1$ y un $p_1 = \frac{1}{3}$ y los utilizamos en la ecuación (1) entonces vemos que los valores únicos para las constantes $c_1 = 1$ y para $c_2 = 0$. Por lo tanto, $p_n = \left(\frac{1}{3}\right)^n$ para todas las n.

Si utilizamos la aritmética de redondeo de 5 dígitos para la sucesión, entonces $\hat{p}_0 = 1.0000$ y $\hat{p}_1 = 0.33333$, lo cual deberiamos transformar a $\hat{c}_1 = 1.00000$ y $\hat{c}_2 = -0.12500 \times 10^{-5}$. Por consecuente, la sucesion generada por $\{\hat{p}_n\}_{n=0}^{\infty}$ esta dada por:

$$\hat{p}_n = 1.0000\left(\frac{1}{3}\right)^n - 0.12500 \times 10^{-5}(3)^n$$

con un error de redondeo,

$$p_n - \hat{p}_n = 0.12500 \times 10^{-5}(3)^n$$

En el proceso anterior el error aumenta **exponencialmente** con n, por tanto inestable, reflejado en las imprecisiones extremas después de los primeros términos (ver siguiente tabla).

n	$\hat{p}_n$ calculada	p_n corregida	Error relativo
0	0.10000×10^{1}	0.10000×10^{1}	
1	0.33333×10^{0}	0.33333×10^{0}	
2	0.11110×10^{0}	0.11111×10^{0}	9×10^{-5}
3	0.37000×10^{-1}	0.37037×10^{-1}	1×10^{-3}
4	0.12230×10^{-1}	0.12346×10^{-1}	9×10^{-3}
5	0.37660×10^{-2}	0.41152×10^{-2}	8×10^{-2}
6	0.32300×10^{-3}	0.13717×10^{-2}	8×10^{-1}
7	-0.26893×10^{-2}	0.45725×10^{-3}	7×10^{0}
8	-0.92872×10^{-2}	0.15242×10^{-3}	6×10^{1}

Figure: Fuente: [1]

Considere ahora la siguiente ecuación recursiva:

$$p_n = 2p_{n-1} - p_{n-2}, \quad \text{para} \quad n = 2, 3, \ldots \tag{2}$$

tiene solución para $p_n = c_1 + c_2 n$ para cualquier c_1 y c_2 ya que:

$$\begin{aligned} 2p_{n-1} - p_{n-2} &= 2(c_1 + c_2(n-1)) - (c_1 + c_2(n-2)) \\ &= c_1(2-1) + c_2(2n - 2 - n + 2) \\ &= c_1 + c_2 n \\ &= p_n \end{aligned}$$

Si nosotros asumimos un $p_0 = 1$ y un $p_1 = \frac{1}{3}$ y los utilizamos en la ecuación (2) entonces vemos que los valores únicos para las constantes $c_1 = 1$ y para $c_2 = -\frac{2}{3}$. Por lo tanto, $p_n = 1 - \left(\frac{2}{3}\right)n$ para todas las n.

Si utilizamos la aritmética de redondeo de 5 dígitos para la sucesión, entonces $\hat{p}_0 = 1.0000$ y $\hat{p}_1 = 0.66667$, lo cual deberiamos transformar a $\hat{c}_1 = 1.00000$ y $\hat{c}_2 = -0.66667$. Por consecuente, la sucesion generada por $\{\hat{p}_n\}_{n=0}^{\infty}$ esta dada por:

$$\hat{p}_n = 1.0000 - 0.66667n$$

con un error de redondeo,

$$p_n - \hat{p}_n = \left(0.66667 - \frac{2}{3}\right)n$$

En el proceso anterior el error aumenta **linealmente** con n, por tanto es estable, reflejado en las aproximaciones de la siguiente tabla.

n	$\hat{p}_n$ calculada	p_n corregida	Error relativo
0	0.10000×10^1	0.10000×10^1	
1	0.33333×10^0	0.33333×10^0	
2	-0.33330×10^0	-0.33333×10^0	9×10^{-5}
3	-0.10000×10^1	-0.10000×10^1	0
4	-0.16667×10^1	-0.16667×10^1	0
5	-0.23334×10^1	-0.23333×10^1	4×10^{-5}
6	-0.30000×10^1	-0.30000×10^1	0
7	-0.36667×10^1	-0.36667×10^1	0
8	-0.43334×10^1	-0.43333×10^1	2×10^{-5}

Figure: Fuente: [1]

- Como el error de redondeo llevar a errores, estos se pueden reducir mediante la aritmética de dígitos de orden superior, como la de precisión doble o múltiple.
- Sin embargo, usar precisión doble conduce a que el costo de estimación requiere más tiempo en los cálculos y no garantiza una eliminación del error completamente.

Burden, R. L., & Faires, J. D. (2011). Numerical analysis.

Howard, J. P. (2017). Computational Methods for Numerical Analysis with R. CRC Press.

Banerjee, S., & Roy, A. (2014). Linear algebra and matrix analysis for statistics. Crc Pr