

PROGRAMACIÓN (EUI). Curso 2001-2002
*Práctica 3. CÁLCULO DE LA FUNCIÓN SENO
UTILIZANDO UN DESARROLLO EN SERIE*

F. Marqués y N. Prieto

Índice General

1	Introducción	1
2	El problema	1
2.1	Desarrollo en serie de la función $\text{sen}(x)$	2
2.2	Cálculo de los términos	2
2.3	Precisión de los cálculos. Condición de finalización	3
3	El programa	3
3.1	Estrategia de resolución	3
3.2	Refinamiento de la estrategia. Variables y tipos de datos	4
3.3	Impresión de resultados	5
4	Modificaciones del programa anterior	5

1 Introducción

Como se ha mencionado en la práctica anterior, uno de los primeros usos de los computadores fue para la resolución de problemas de índole numérica como la tabulación de funciones elementales, etc.

En dicha práctica se denominaba *algoritmos numéricos* a aquellos algoritmos de índole matemática basados en el uso reiterado de las operaciones aritméticas básicas. Como veremos en la presente práctica, una característica significativa de dicho tipo de algoritmos, que aun no se ha mencionado, es que a veces no proporcionan respuestas exactas. Sin embargo, aun cuando eso ocurre, dichas respuestas pueden conseguirse habitualmente con cierto grado de exactitud prefijado.

Otra característica muy importante, que tampoco ha sido aún mencionada, de los algoritmos en general, particularizable a los de de índole numérica, es que su ejecución consume tiempo de cómputo. Una de las tareas más importantes en el ámbito de la computación consiste en conocer una aproximación, lo más precisa posible, al número de operaciones elementales que deben efectuarse para resolver un problema dado.

En esta práctica se plantea el problema de calcular el valor de una función trigonométrica en un punto, utilizando la expresión de un desarrollo en serie de la misma. Como la respuesta no siempre se puede conocer con exactitud se determinará además, una cota del error cometido. Es interesante señalar que la mayor parte de los lenguajes de programación actuales disponen de primitivas para hacer ese tipo de cálculos, lo que supone que en el compilador del lenguaje, o incluso en el mismo procesador, se encuentran presentes, de algún modo, algoritmos similares al que se proponen.

2 El problema

Se desea calcular el valor de la función seno de cierto ángulo x . Para ello se estudiará a continuación un desarrollo en serie de dicha función que permita calcularla. Como se verá, la precisión del cálculo depende del número total de términos del desarrollo que se utilicen. Parámetros del problema pueden ser, por lo tanto, tanto el valor del ángulo x , como la precisión, o *error residual* máximo, con que se desee obtener el resultado.

Teniendo lo anterior en cuenta, y eliminando algunas ambigüedades, el problema se puede reenumerar del modo siguiente: Realizar un programa para determinar el valor del seno de un ángulo, expresado en radianes, x , con cierto error residual máximo *error*.

2.1 Desarrollo en serie de la función $\text{sen}(x)$

Dado x , número real que representa un ángulo en radianes, la expresión que se muestra a continuación es un desarrollo en serie de la función $\text{sen}(x)$:

$$\text{sen}(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!} \quad (1)$$

Si representamos el término i -ésimo ($0 \leq i$) del desarrollo anterior por u_i , entonces:

$$\text{sen}(x) = u_0 + u_1 + u_2 + \dots + u_k + u_{k+1} + \dots + u_n + R_n$$

esto es, toda la serie puede representarse como suma de términos u_i , junto con un resto R_n , que representa la suma de los términos restantes, posteriores al n -ésimo. O lo que es lo mismo:

$$\text{sen}(x) = \sum_{i=0}^n (u_i) + R_n$$

2.2 Cálculo de los términos

Naturalmente, el programa que se construya deberá calcular cada uno de los términos u_i de la expresión anterior. Sustituyendo en (1), se puede obtener el primer término de la serie, u_0 , que vale x . También sustituyendo en (1), se obtiene para los dos términos consecutivos cualesquiera: u_{k-1} y u_k , las siguientes expresiones:

$$u_{k-1} = (-1)^{k-1} \frac{x^{2k-1}}{(2k-1)!} \quad u_k = (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

A partir de las mismas se puede observar que se cumple la siguiente relación entre dos términos consecutivos cualesquiera:

$$u_k = -\frac{x^2}{2k(2k+1)}u_{k-1} \quad (2)$$

Como se ve, es posible ahorrar muchos cálculos si cada nuevo término se obtiene a partir del inmediatamente anterior, en lugar de calcularlo de forma independiente.

2.3 Precisión de los cálculos. Condición de finalización

La precisión con que se efectúa el desarrollo viene dada por el término R_n , que representa, en un momento del cálculo, la parte de la serie no sumada. El término R_n , que se denomina *error residual*, representa por lo tanto la diferencia entre el valor de la función seno calculada y la real. Por características propias de la serie (es monótona, alternada, y estrictamente decreciente en valor absoluto) se puede demostrar que:

$$|R_n| \leq |u_{n+1}| < |u_n|$$

esto es: cuando se ha calculado y sumado el término n -ésimo de la serie, el error cometido es inferior o igual (en valor absoluto) al valor del término u_{n+1} que es el que se calcularía a continuación y que es, a su vez, estrictamente menor (también en valor absoluto) al valor del término n -ésimo.

Resumiendo, se puede decir que: el error residual que se comete, en valor absoluto, es siempre estrictamente menor que el valor del último término calculado. Por lo que, si se desea que dicho error sea menor que cierto ϵ dado, basta con calcular un término tras otro (sumándolos) hasta que se llegue a uno con valor inferior a dicho ϵ . Así, se tendría:

$$|R_n| \leq |u_{n+1}| < |u_n| < \epsilon$$

3 El programa

Como se ha dicho ya en la práctica anterior es necesario para poder resolver el problema mediante un programa determinar la estrategia que seguirá el programa y las características de los datos que intervengan para expresar tanto la estructura del programa solución como las variables y tipos de datos implicados.

3.1 Estrategia de resolución

Para poder resolver el problema es posible plantear, a grandes rasgos, una estrategia como la que se describe textualmente a continuación:

1. Obtener inicialmente el valor del ángulo x y el error residual **epsilon**,
2. Calcular sucesivamente (y sumar) todos los términos del desarrollo propuesto hasta que el último término calculado valga menos que **epsilon**.
3. la suma de todos los términos calculados es el valor deseado, escribirlo adecuadamente.

Claramente esta estrategia no describe con el suficiente nivel de detalle todos los aspectos relevantes del programa; así, entre otras cosas, no describe de forma totalmente precisa cómo es la iteración (por ejemplo, no se menciona el hecho de que un término se calcula en función del anterior, etc.) Es necesario, por ello, *refinar* la estrategia, aproximando su enunciado al nivel de detalle necesario.

Se puede decir que un *refinamiento* de una estrategia consiste, en detallar aquellas partes de la misma que no están completamente definidas, hasta un cierto nivel de detalle deseado. En el caso límite, este nivel de detalle son las instrucciones del programa.

Del análisis de la estrategia que se ha propuesto se puede ver que no explicita el cálculo del primer término, ni tampoco el de los terminos siguientes en función del anterior de cada uno. Tampoco precisa el proceso que se tiene que efectuar para acumular la suma de términos.

3.2 Refinamiento de la estrategia. Variables y tipos de datos

Es necesario considerar ahora las variables que harán falta para poder resolver el problema.

En primer lugar, hace falta una variable, por ejemplo puede llamarse **term**, para mantener el valor del término que en un momento dado se esté calculando. Además, por los desarrollos anteriores se conoce que el primer término vale x , que será el valor inicial de **term**. En cada iteración, para obtener un nuevo término a partir del anterior, basta con calcular la expresión correspondiente, dada por (2), que depende del último término **term** y también del número de término que se esté calculando, por lo que es necesario también mantener el número de término en otra variable (por ej. puede llamarse **k**). Para mantener la suma de todos los términos hay que utilizar una variable acumulador, por ejemplo de nombre **sumSen**. Todos los tipos de datos de las variables (**int** y **double**) son evidentes.

Con todo ello, un *refinamiento* de la estrategia enunciada antes, es el siguiente:

1. Obtener inicialmente el valor del ángulo x y el error residual **epsilon**, (por ejemplo, pidiéndolos al usuario, o como argumentos del programa)

2. Calcular el primer término de la serie (valor original del ángulo x), inicializando con él las variables `term`, valor del último término calculado hasta el momento, y `sumSen` que se usará como acumulador. Inicializar una variable `k` al número de término calculado (el primer término es el 0). Nótese que el error cometido es menor que el valor del último término calculado (`term`).
3. mientras que el valor del último término sea mayor o igual que el error deseado; esto es: mientras que (`term >= epsilon`)
 - (a) Incrementar el índice del término calculado `k`,
 - (b) Calcular el nuevo término (reemplazando el valor anterior de `term`), aplicando la expresión (2)
 - (c) Acumular el nuevo término a `sumSen`
4. la suma de todos los términos calculados (`sumSen`) es el valor deseado, escribirlo adecuadamente.

3.3 Impresión de resultados

Además del propio valor del seno del ángulo x que se calcula, hay otra información que también puede resultar interesante imprimir. Por ejemplo, ante cada entrada de datos (valores de x y `epsilon`), se puede imprimir lo siguiente:

- Valor calculado de $\text{sen}(x)$ por el programa,
- valor de la variable `epsilon`, para conocer el máximo error residual,
- número de iteraciones efectuadas en el cálculo, o lo que es lo mismo, número del último término de la serie desarrollado
- valor de $\text{sen}(x)$, calculado utilizando la primitiva del lenguaje (para así compararlo con el calculado mediante el programa)

Obviamente, la impresión debe ser tal que se pueda identificar y leer claramente cada uno de los valores presentados.

4 Modificaciones del programa anterior

A continuación se proponen algunas modificaciones al programa realizado en el apartado anterior.

1. El número de iteraciones realizadas en el bucle del programa anterior es una medida proporcional al tiempo de cómputo necesario para resolver el problema propuesto. Este número depende del valor de x y del error residual, ϵ , considerado.
 Modificar el programa anterior para que tabule, como se muestra a continuación, los resultados pedidos para valores de x comprendidos entre 0 y 20 radianes, considerando incrementos de 1 radián, y para un error residual $\epsilon = 10^{-11}$

x	sen(x)	sen(x)'	iteraciones
1	*****	*****	**
2	*****	*****	**
3	*****	*****	**
..
..
..
20	*****	*****	**

2. Modificar el programa anterior para que acepte como datos de entrada radianes o grados, a elección del usuario. Si el usuario opta por dar la entrada en radianes el programa efectuará los mismos cálculos que los ya desarrollados, pero si el usuario opta por dar su entrada en grados, el programa deberá convertir previamente dicho valor a radianes (utilizando el hecho de que 180 grados son π radianes), y efectuar, a continuación, los cálculos.
3. Es posible reducir los cálculos del seno de cualquier ángulo a uno equivalente realizado en la primera semicircunferencia, o en la primera circunferencia, basándose en el hecho de que:

$$\text{sen}(\alpha) = -\text{sen}(\pi + \alpha) = \text{sen}(2\pi + \alpha)$$

Modificar el programa original para determinar el valor de $\text{sen}(x)$ para valores del ángulo x comprendidos entre -10000 y 10000 radianes.