Una interfaz Fortran 95 genérica para las subrutinas fzero/dfzero de SLATEC

Pablo Santamaría

v0.1.1 (Agosto 2012)

Introducción

Un método de propósito general que permita determinar aproximaciones precisas a cada raíz de una *ecuación no lineal*, $f(x) = 0$, debería poder acotar la raíz dentro intervalos sucesivos cada vez menores y, a la vez, ser rápidamente convergente. Sabemos que el *método de bisección* acota sucesivamente la raíz dentro de un intervalo cada vez menor, pero la convergencia es lenta. Por otra parte, el *método de Newton* o *el método de la secante* convergen más rápidamente que el método de bisección, pero abandonan toda posible acotación de la raíz y dicha convergencia ocurrirá sólo si la aproximación inicial está suficientemente próxima a la raíz buscada. Es claro entonces que el algoritmo deseado debe ser un procedimiento híbrido que combine las ventajas de ambos métodos. En particular, uno de tales algoritmos, es el llamado *método de Dekker*, el cual combina el método de bisección con el método de la secante de forma tal que si la siguiente aproximación dada por el algoritmo de la secante cae fuera de un intervalo alrededor de la solución de longitud menor al intervalo del paso anterior, entonces se utiliza la aproximación correspondiente al método de bisección para dicha iteración.

En circunstancias normales, la ecuación no lineal a resolver por el método de Dekker está dada por una función $f(x)$ continua sobre un intervalo $[a, b]$ para el cual $f(a)f(b) < 0$, lo cual asegura la existencia de, al menos, una raíz en dicho intervalo. El método genera entonces, en cada iteración, un nuevo intervalo $[a, b]$ con $f(a)f(b) < 0$, que decrece en longitud respecto de la iteración anterior, y tal que a constituye una mejor aproximación a la raíz que b, en el sentido de que $|f(a)| \leq |f(b)|$, y una mejor aproximación que el punto medio $m = (a + b)/2$. Las iteraciones proceden hasta que se satisface el criterio de convergencia:

$$\frac{|b - a|}{2} \leq \text{máx}\,\{\epsilon_{\text{abs}}, \epsilon_{\text{rel}}|a|\},$$

donde ϵ_{abs} y ϵ_{rel} son tolerancias prefijadas para el error absoluto y relativo, respectivamente. Para comprender lo que este test significa, supongamos primero que $\epsilon_{\text{rel}} = 0$, entonces la condición se reduce a

$$\frac{|b - a|}{2} \leq \epsilon_{\text{abs}}$$

la cual es la condición usual para asegurar que el punto medio m no está alejado de la raíz en más de ϵ_{abs}. Pero, puesto que se asume a es una mejor aproximación a la raíz que m, el criterio proporciona también un test para la precisión a como aproximación de la raíz. Nótese que si, en circunstancias desfavorables, a no es mejor aproximación que m, el test implica que a está alejado de la raíz no mas de $2\,\epsilon_{\text{abs}}$. Si por otra parte, suponemos ahora que $\epsilon_{\text{abs}} = 0$ y el criterio fuera

$$\frac{|b - a|}{2} \leq |m|\epsilon_{\text{err}}$$

el mismo sería un test para el error relativo para la aproximación m de la raíz. Como se asume que a es una mejor aproximación que m, el criterio es utilizado con a en vez de m. De este modo la condición mixta de testeo del error absoluto y relativo proporciona robustez al método puesto que si la raíz es próxima a cero, un test basado únicamente en el error relativo no resulta apropiado. Además, para evitar un bucle infinito en circunstancias desfavorables, las iteraciones son detenidas si el criterio no se satisface después de un determinado número máximo de evaluaciones de $f(x)$.

El algoritmo es tan robusto que puede tratar con casos excepcionales, tal como encontrar raíces de funciones que son continuas a trozos, o localizar funciones que tienen un polo de multiplicidad impar. Este último caso es reconocido por el hecho de que si bien la sucesión generada de intervalos encajados $[a, b]$ decrecen en tamaño, los valores de $|f(a)|$ se incrementan en vez de decrecer.

Implementación en Fortran 77 vía SLATEC

Una implementación en Fortran 77 del método de Dekker forma parte de la biblioteca de subrutinas SLATEC. SLATEC Common Mathematical Library es una colección de rutinas escritas en Fortran 77 para resolver varios tipos de problemas matemáticos y estadísticos, la cual está accesible en http://www.netlib.org/slatec/. En particular el método de Dekker es implementado por las subrutinas `fzero/dfzero` cuyos argumentos son:

```
     fzero(f,a,b,x0,eps_rel,eps_abs,clave)
    dfzero(f,a,b,x0,eps_rel,eps_abs,clave)
```

donde, en `fzero`, todos los argumentos reales son de simple precisión, mientras que en `dfzero` son de doble precisión. El significado de los argumentos es el siguiente:

`f`: la función que define la ecuación $f(x) = 0$, la cual es implementada como una `FUNCTION` externa de una variable.

`a,b`: como datos de entrada constituyen los extremos del intervalo inicial $[a, b]$ con $f(a)f(b) < 0$. Como datos de salida, constituyen un intervalo que encierra a la raíz, siendo a una mejor aproximación a b de la raíz, en el sentido de que $|f(a)| \leq |f(b)|$ (si éste no es el caso, la subrutina intercambia los valores de a y b de manera tal que dicha condición se mantenga).

`x0`: es un dato de entrada que permite dar una mejor aproximación inicial a la raíz la cual puede ayudar a acelerar la convergencia del método. En general, si no se conoce una mejor aproximación inicial, es recomendable asignar `x0 = a` ó `x0 = b`.

`eps_rel`: es un dato de entrada que da una tolerancia para el error relativo.

`eps_abs`: es un dato de entrada que da una tolerancia para el error absoluto.

`clave`: es un dato de salida especificado por una variable entera permite al usuario verificar si el método funcionó correctamente. Los valores posibles y su significado son los siguientes:

1. OK. El método funcióno correctamente: el valor retornado en a es una raíz de f dentro de la tolerancia especificada, la raíz se encuentra dentro del intervalo $[a, b]$ devuelto, la función cambia de signo en los extremos de dicho intervalo y $f(x)$ decrece en magnitud conforme el intervalo decrece su longitud.
2. OK. El valor retornado en a cumple la condición $f(a) = 0$, pero el intervalo $[a, b]$ no necesariamente encierra a la raíz dentro de la tolerancia prefijada (por lo tanto, el valor devuelto en b no es un dato útil en este caso).
3. Problemas. El intervalo retornado $[a, b]$ se encuentra dentro de la tolerancia prefijada y la función cambia de signo en los extremos de dicho intervalo, pero su valor se fue incrementando conforme los intervalos $[a, b]$ se achicaban. Es probable que a se encuentre próximo a un punto singular de $f(x)$.
4. Problemas. El intervalo retornado $[a, b]$ se encuentra dentro de la tolerancia prefijada pero la función no cambia de signo en sus extremos. El usuario debe entonces examinar con más cuidado si a está cerca de un mínimo local de $f(x)$ o de una de una raíz de multiplicidad par, o bien ninguna de estas situaciones.
5. Problemas. Se alcanza un número máximo de evaluaciones (> 500) de $f(x)$ sin poder determinar la raíz. El usuario debe considerar un intervalo inicial más chico o determinar si realmente hay una raíz en él.

Interfaz genérica para Fortran 95

En Fortran 77 la interfaz de todos los subprogramas (subrutinas y funciones) es *implícita* y por lo tanto, posibles errores que pueden ocurrir en los tipos de los argumentos al convocar el subprograma no son detectados por el compilador. Por su parte, Fortran 90 permite detectar este tipo de errores (y utilizar otras características avanzadas) si la interfaz del subprograma se hace *explícita*. En nuestro caso, donde la subrutinas `fzero/dfzero` forman parte de una implementación escritas en una versión de Fortran anterior

y forman parte de una biblioteca de funciones ya compiladas, la interfaz explícita se realiza a través de un *bloque interfaz*. Mas aún, en Fortran 95, podemos crear un subprograma genérico que sea capaz de decidir cual de las dos subrutinas, `fzero` o `dfzero`, debe utilizarse, dependiendo de los argumentos provistos en la invocación del mismo. Esto es realizado con una versión especial del bloque interfaz, llamada *bloque interfaz general*, la cual requiere el nombre genérico que se dará a la interfaz. Especificamente, si designamos a la implementación de nuestra interfaz genérica como `zero`, la forma del mismo será:

```
INTERFACE zero
  interface explícita para fzero
  interface explícita para dfzero
EN INTERFACE zero
```

Cuando el compilador encuentre el nombre genérico `zero` en el programa que realice la llamada, éste examinará los argumentos asociados en la llamada para decidir cual es la subrutina específica, `fzero` o `dfzero`, que debe usarse. A continuación presentamos la implementación de esta interfaz generica en un módulo, al que llamaremos `slatec95`. Como es usual, para referirnos a datos reales de simple y doble precisión utilizaremos a su vez el módulo `precision` implementado en un apunte anterior [1].

Código 1. Interfaz genérica zero

```
MODULE slatec95

  INTERFACE zero
    ! Interface para precisión simple
    SUBROUTINE fzero(f, a, b, x0, eps_rel, eps_abs, clave)
        USE precision, WP=> SP
        IMPLICIT NONE
        INTERFACE
            FUNCTION f(x)
                USE precision, WP => SP
                REAL(WP) :: f
                REAL(WP), INTENT(IN) :: x
            END FUNCTION f
        END INTERFACE
        REAL(WP), INTENT(INOUT) :: a
        REAL(WP), INTENT(INOUT) :: b
        REAL(WP), INTENT(IN)    :: x0
        REAL(WP), INTENT(IN)    :: eps_rel
        REAL(WP), INTENT(IN)    :: eps_abs
        INTEGER, INTENT(OUT):: clave
    END SUBROUTINE fzero
    ! Interface para precisión doble
    SUBROUTINE dfzero(f, a, b, x0, eps_rel, eps_abs, clave)
        USE precision, WP=> DP
        IMPLICIT NONE
        INTERFACE
            FUNCTION f(x)
                USE precision, WP => DP
                REAL(WP) :: f
                REAL(WP), INTENT(IN) :: x
            END FUNCTION f
        END INTERFACE
        REAL(WP), INTENT(INOUT) :: a
        REAL(WP), INTENT(INOUT) :: b
        REAL(WP), INTENT(IN)    :: x0
        REAL(WP), INTENT(IN)    :: eps_rel
        REAL(WP), INTENT(IN)    :: eps_abs
        INTEGER, INTENT(OUT):: clave
    END SUBROUTINE dfzero
  END INTERFACE zero

END MODULE slatec95
```

[1] Véase *Un módulo que parametriza las clases de tipos de datos reales.*

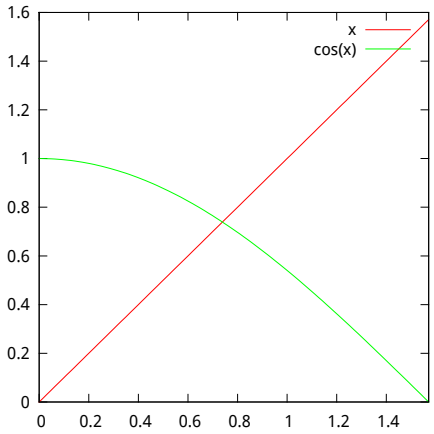

Figura 1. Determinación de la raíz de la función $\cos x - x$.

Ejemplo

Como ejemplo del uso de la subrutina genérica `zero` implementada en el módulo `slatec95` consideremos el problema de determinar la raíz de la ecuación

$$\cos x - x = 0.$$

Al graficar las curvas $y = x$ e $y = \cos x$, vemos en la fig. 1, que la interesección de las mismas ocurre dentro del intervalo $[0.6, 0.8]$. Así pues, la raíz buscada se encuentra en el interior de este intervalo y con él podemos proceder a calcular la raíz asumiendo una tolerancia de 5×10^{-6} para el error relativo y una tolerancia de $\frac{1}{2} \times 10^{-6}$ para el error absoluto. El siguiente programa implementa lo requerido.

Código 2. Determinación de la raíz de la ec. $\cos x - x = 0$

```
PROGRAM ejemplo
  USE precision, WP => DP
  USE slatec95
  IMPLICIT NONE
  INTERFACE
     FUNCTION f(x)
       IMPORT :: WP
       IMPLICIT NONE
       REAL(WP) :: f
       REAL(WP), INTENT(IN) :: x
     END FUNCTION f
  END INTERFACE

  REAL(WP) :: a,b,tol_rel,tol_abs
  INTEGER  :: clave

  tol_rel = 5E-6_WP
  tol_abs = 0.5E-6_WP
  a   = 0.6_WP
  b   = 0.8_WP
  CALL zero(f,a,b,a,tol_rel,tol_abs,clave)
  WRITE(*,*) 'Clave = ', clave
  WRITE(*,*) 'Raiz = ', a
  WRITE(*,*) 'Intervalo = ', a,b

END PROGRAM ejemplo

FUNCTION f(x)
  ! Función que define la ecuación
  USE precision, WP => DP
  IMPLICIT NONE
  REAL(WP) :: f
```

```
    REAL(WP), INTENT(IN) :: x
    f = cos(x) - x
    RETURN
  END FUNCTION f
```

La compilación del mismo procede en la línea de comandos como sigue:

```
$ gfortran -Wall -o ejemplo precision.f90 slatec95.f90 ejemplo.f90 -lslatec
```

Su ejecución arroja entonces:

```
$ ./ejemplo
Clave =            1
Raiz =   0.73908514415511095
Intervalo =   0.73908514415511095  0.73908144872939019
```

Así, pues, la raíz buscada, con seis decimales correctos y seis dígitos significativos, es 0.739085.