Integración adaptativa con QUADPACK

Pablo Santamaría

v0.1 (Octubre 2011)

Planteo del problema

Supongamos que se desea calcular una estimación numérica Q de la integral definida

$$I = \int_a^b f(x)\,\mathrm{d}x$$

tal que el error $|E| = |Q - I|$ sea menor o igual que cierta tolerancia ϵ prefijada. Si la función f se puede evaluar en cualquier punto del intervalo de integración podríamos comenzar considerando alguna fórmula de cuadratura numérica compuesta como ser la *regla compuesta del trapecio* sobre n subintervalos igualmente espaciados una distancia $h = (b - a)/n$,

$$Q = \frac{h}{2}\big[f(x_0) + f(x_n) + \sum_{i=1}^{n-1} f(x_i)\big],$$

siendo $x_0 = a$, $x_n = b$, $x_i = x_0 + ih$ para $i = 0, 1, \ldots, n$, cuyo término de error es

$$E = -\frac{h^2(b-a)}{12} f''(\xi),$$

para algún $\xi \in (a, b)$. Si disponemos de una cota M de f'' entonces la desigualdad

$$|E| \leq \frac{h^2(b-a)}{12} M \leq \epsilon,$$

permite determinar el paso h (y el valor de n) de una aproximación Q que efectivamente está dentro de la tolerancia prefijada. Sin embargo, para una implementación algorítmica, debemos encontrar una manera de estimar la precisión de la aproximación Q que no involucre determinar f''. Para hacer esto utilizamos un principio básico del análisis numérico: *la estimación del error de una aproximación resulta de comparar con una aproximación más precisa.* Supongamos, entonces, que además de Q disponemos de otra aproximación $\hat{Q}$ de I que fue obtenida con un método más preciso. La ecuación

$$E = I - Q = \hat{Q} - Q + (I - \hat{Q}),$$

nos dice que cuando $\hat{Q}$ es más precisa que Q, el error en Q puede estimarse como $\hat{Q} - Q$. Ahora bien, el cálculo de $\hat{Q}$ agrega un costo computacional que debería llevarse al mínimo reutilizando para su cómputo las evaluaciones de f ya utilizadas para el cálculo de Q. Por ejemplo, en nuestro caso, donde utilizamos la regla del trapecio para evaluar Q podemos utilizar la *regla compuesta de Simpson* para determinar $\hat{Q}$ (y por tanto, estimar el error de Q), dado que sabemos que dicho método es considerablemente más preciso que el primero y además podemos utilizar los mismos puntos x_i (en tanto $n + 1$ sea un número impar).

Disponiendo de una fórmula de cuadratura compuesta y la posibilidad, según lo anterior, de estimar el error cometido, se puede entonces implementar un procedimiento *automático* de

integración numérica como sigue: Comenzando con una paso de integración h (digamos $h = b - a$) calculamos la aproximación Q y la estimación de su error. Si dicha estimación del error es menor o igual que la tolerancia ϵ prefijada entonces Q es la aproximación buscada a I. De lo contrario, dividimos el paso h de integración a la mitad y comenzamos de nuevo. Aunque funciona, este procedimiento es, en general, ineficiente debido a la gran cantidad de evaluaciones del integrando que deben realizarse al utilizar un paso de integración constante sobre todo el intervalo $[a, b]$ en cada iteración. Un mejor esquema de integración surge teniendo presente que en aquellas regiones del intervalo de integración donde el integrando f no presenta grandes variaciones se pueden tomar subintervalos "grandes" y que en regiones donde la variación de f es "grande" podemos tomar subintervalos "pequeños" de manera tal que la suma de las contribuciones al error total sea menor que la tolerancia prefijada. Los procedimientos de integración que adaptan la longitud de los subintervalos al comportamiento del integrando se llaman, apropiadamente, *métodos adaptativos de cuadratura*.

Un método adaptativo de cuadratura procede como sigue: se aplica la regla de cuadratura sobre todo el intervalo (esto es, con $h = b-a$) para obtener una aproximación Q a la integral y se estima su error (a través de la aplicación de una regla de cuadratura de mayor precisión). Si el error estimado es mayor que la tolerancia prefijada, se divide el intervalo de integración a la mitad y se aplica la regla de cuadratura a cada uno de los subintervalos y se estima sus respectivos errores. Ahora, si la suma de dichas estimaciones de los errores excede la tolerancia requerida, entonces el subintervalo con *mayor* cota de error es, a su vez, subdividido a la mitad y así siguiendo hasta que la estimación del error total sea menor que la tolerancia prefijada. Con más precisión, en una dada etapa del procedimiento el intervalo $[a, b]$ se encuentra particionado en subintervalos dados por los puntos $a = \alpha_1 < \beta_1 = \alpha_2 < \beta_2 = \alpha_3 < \cdots < \beta_n = b$ donde se tiene una estimación Q_i para cada integral I_i de f sobre $[\alpha_i, \beta_i]$ y una estimación E_i del error de Q_i. A partir de estos valores se tiene una estimación $Q = \sum Q_i$ para la integral I con una estimación $E = \sum E_i$ de su error. Si esta aproximación Q no está dentro de la tolerancia prefijada, esto es, si no se cumple la condición $|E| = |\sum E_i| \leq \epsilon$, entonces el subintervalo $[\alpha_i, \beta_i]$ con *mayor* cota $|E_i|$ de error es escogido para subdividirlo. Dicho intervalo es entonces particionado en dos mitades: $[\alpha_i, (\alpha_i+\beta_i)/2]$, $[(\alpha_i+\beta_i)/2, \beta_i]$ y se calculan las aproximaciones a las integrales sobre cada uno de tales subintervalos y sus respectivas estimaciones de error. Estos dos subintervalos y las cantidades asociadas reemplazan al subintervalo $[\alpha_i, \beta_i]$ y sus cantidades asociadas. El procedimiento es repetido hasta que $|E| = |\sum E_i| \leq \epsilon$[1]. Al final del proceso, la estimación Q de la integral ha sido calculada, dentro de la tolerancia prefijada, aplicando la regla de cuadratura en una partición del intervalo $[a, b]$ dada por subintervalos $[\alpha_i, \beta_i]$ que tienden a ser más pequeños en aquellas regiones donde f presenta mayores variaciones.

Al igual que en nuestra primera aproximación naïve, un método adaptivo requiere de una fórmula de cuadratura y una estimación de el error cometido a través de una regla de cuadratura de mayor precisión que, con el fin de mantener mínimo el esfuerzo computacional, debería reutilizar los mismos puntos que la regla inicial. La combinación de la regla de trapecios y Simpson parecen buenos candidatos y, con más generalidad, se pueden utilizar *reglas de cuadratura de Newton-Cotes* de diferentes órdenes, puesto que, al tener un paso igualmente espaciado compartirán puntos en común. Sin embargo, sabemos que las *reglas de cuadratura gaussianas* son más precisas que las reglas de Newton-Cotes para el mismo número de puntos. Pero, desafortunadamente, reglas gaussianas de diferente órden no tienen puntos en común, con lo cual, la estimación del error con reglas gaussianas requiere la evaluación del integrando f sobre todos los conjuntos de puntos de ambas reglas. Para evitar tal trabajo computacional se han desarrollado las *reglas de cuadratura de Gauss-Kronrod*. Una dada regla de Gaus-Kronrod consiste en una regla de Gauss G_n de n-puntos y una regla de Kronrod K_{2n+1} de $(2n+1)$-puntos que han sido escogidos de manera óptima sujetos a la condición de que todos los puntos de G_n son reusados en K_{2n+1}[2].

[1] Este procedimiento adaptativo se conoce como *global*. Existe una alternativa *local* que procede subdividiendo los intervalos de manera que cada uno de ellos tenga una estimación E_i del error acotada por $|E_i| \leq |(\beta_i - \alpha_i)/(b-a)|\epsilon$ con lo cual $\sum |E_i| \leq \epsilon$. Dado que $|\sum E_i| \leq \sum |E_i|$ el procedimiento local requiere de más trabajo para alcanzar la tolerancia prefijada que el procedimiento global, pero, en compensasión, debería proveer, en general, resultados que son aún más precisos.

[2] Mientras que una regla de Gauss de $(2n+1)$-puntos es exacta para polinomios de hasta grado $4n+1$, una regla

La implementación computacional del esquema de cuadratura adaptativo global con reglas de Gauss-Kronrod es el núcleo del paquete de integración numérica conocido como QUADPACK, las cuales a su vez forman parte del paquete de rutinas matemáticas SLATEC. Estas rutinas están escritas en FORTRAN 77 y entre ellas, la subrutina apropiada para nuestro problema planteado al inicio de esta discusión, es la subrutina `QAGS`. La misma implementa un procedimiento adaptativo y global de cuadratura en base una regla de Gauss-Kronrod de 10 y 21 puntos, respectivamente, junto con un procedimiento de extrapolación. El uso de su implementación, en doble precisión, `DQAGS`, es discutida a continuación.

Uso de la subrutina `DQAGS` del paquete `QUADPACK`

Para datos reales de *doble precisión* la interface de la subrutina `DQAGS` es como sigue:

```
SUBROUTINE DQAGS( F, A, B, EPSABS, EPSREL, RESULT, ABSERR, NEVAL,
                                IER, LIMIT, LENW, LAST, IWORK, WORK)
   DOUBLE PRECISION   F, A, B, EPSABS, EPSREL, RESULT, ABSERR
   INTEGER            NEVAL, IER, LIMIT, LENW, LAST
   INTEGER            IWORK(LIMIT)
   DOUBLE PRECISION   WORK(LENW)
```

donde,

`F`: Es un dato de entrada que apunta a una función externa de doble precisión de una variable de doble precisión, correspondiente al integrando f.

`A`: Es un dato de entrada de doble precisión correspondiente al límite inferior a de la integral.

`B`: Es un dato de entrada de doble precisión correspondiente al límite superior b de la integral.

`EPSABS`: Es un dato de entrada de doble precisión correspondiente a una tolerancia prefijada en el error absoluto de la aproximación de la integral.

`EPSREL`: Es un dato de entrada de doble precisión correspondiente a una tolerancia prefijada en el error relativo de la aproximación de la integral.

`RESULT`: Es un dato de salida de doble precisión que da la aproximación numérica Q de la integral.

`ABSERR`: Es un dato de salida de doble precisión que da una cota del error absoluto de Q, esto es, $|Q - I| \leq$ `ABSERR`.

`NEVAL`: Es un dato de salida entero que da el número total de evaluaciones de f efectuadas por el método.

`IER`: Es un dato de salida entero utilizada como clave de éxito o error en el cómputo de la aproximación numérica de la integral. Si `IER=0`, la subrutina ha calculado exitosamente una aproximación numérica Q de la integral dentro de la tolerancia prefijada $|Q - I| \leq$ máx $\{$`EPSABS`, `EPSREL`$|I|\}$ con una cota de error $|Q-I| \leq$ `ABSERR`. Valores de `IERR` no nulos indican una terminación anormal del método y, en consecuencia, los valores devueltos para la aproximación numérica y su estimación de error no son confiables. `IERR = 1` significa que el número máximo de subdivisiones permitadas del intervalo de integración ha sido alcanzado. `IERR = 2` implica que la tolerencia prefijada para el error no pudo ser alcanzada. `IERR = 3` significa que la función es muy mal comportada en algunos puntos del intervalo de integración. `IERR = 4` implica que el algoritmo de extrapolación no converge. `IERR = 5` significa que es muy probable que la integral sea divergente o converga muy lentamente. `IERR = 6` implica que hay algún error en los datos de entrada de la subrutina.

de Kronrod es exacta para polinomios de hasta grado $3n + 1$.

`LIMIT`: Es un dato entero de entrada que especifica el número máximo de particiones de intervalo de integración $[a, b]$ que puede realizar el método.

`LENW`: Es un dato entero de entrada que indica el tamaño del arreglo `WORK`, cuyo valor debe ser mayor o igual a `4*LIMIT`.

`LAST`: Es un dato entero de salida que da el número de subintervalos producidos por el proceso de subdivisión.

`IWORK`: Es una arreglo unidimensional de datos enteros que permite ordenar de mayor a menor las estimaciones de error E_i de las aproximaciones numéricas Q_i correspondientes a los subintervalos $[\alpha_i, \beta_i]$ de la partición realizada, de la siguiente manera:

$$E_i = \texttt{WORK(3*LIMIT+IWORK(i))}$$

para $i = 1, 2, \ldots, k$, siendo $k =$ `LAST` si `LAST` $\leq$ `LIMIT/2+2`, o, de lo contario, $k =$ `LIMIT+1-LAST`.

`WORK`: Es una arreglo unidimensional de datos de doble precisión que contiene a la salida de la subrutina los extremos de los subintervalos $[\alpha_i, \beta_i]$ de la partición realizada junto con las aproximaciones numéricas Q_i correspondientes a la integral de f sobre cada subintervalo y las estimaciones de error E_i de las mismas, de la siguiente manera:

$$\begin{aligned} \alpha_i &= \texttt{WORK(i)} \\ \beta_i &= \texttt{WORK(LIMIT+i)} \\ Q_i &= \texttt{WORK(2*LIMIT+i)} \\ E_i &= \texttt{WORK(3*LIMIT+i)} \end{aligned}$$

para $i = 1, 2, \ldots,$ `LAST`.

El siguiente programa nos permitirá aplicar la subrutina `DQAGS` a cualquier integral que querramos calcular escribiendo el apropiado subprograma `FUNCTION` para la función del integrando.

```
      program quadpack_test
*     ------------------------------------------------
*     Declaración de tipo
*     ------------------------------------------------
      implicit none
      integer limit, lenw
      parameter (limit = 200)
      parameter (lenw = 4*limit)
      integer iwork(limit)
      double precision work(lenw)
      integer nval,ier,last
      double precision f,a,b,epsabs,epsrel,result,abserr
      external f
*     ------------------------------------------------
*     Datos de entrada
*     ------------------------------------------------
      write(*,*) 'Ingrese'
      write(*,*) 'Limite inferior de la integral'
      read(*,*) a
      write(*,*) 'Limite superior de la integral'
      read(*,*) b
      write(*,*) 'Tolerencia para el error absoluto'
```

```
      read(*,*) epsabs
      write(*,*) 'Tolerancia para el error relativo'
      read(*,*) epsrel
*     ------------------------------------------------
*     Llamada a la subrutina
*     ------------------------------------------------
      call dqags(f,a,b,epsabs,epsrel,result,abserr,
     *     nval,ier,limit,lenw,last,iwork,work)
*     ------------------------------------------------
*     Imprimir resultados
*     ------------------------------------------------
      write(*,*) 'Status code = ',ier
      write(*,*) 'Integral =', result
      write(*,*) 'Error =', abserr
      write(*,*) 'Número de evaluaciones de f =', nval
*     ------------------------------------------------
      end

*     ------------------------------------------------
*     Función a integrar
*     ------------------------------------------------
      double precision function f(x)
      implicit none
      double precision x
      f = ...
      end
```

Asumiendo que este programa está guardado en el archivo fuente `quadpack_test.f`, incorporamos la biblioteca de rutinas SLATEC en la compilación como sigue:

```
$ gfortran -Wall -o quadpack_test quadpack_test.f -lslatec
```

Ejemplos

Ejemplo 1. Con el fin de testear el código apliquemoslo a una integral cuya solución analítica es conocida de antemano, como ser, por ejemplo,

$$\int_0^1 \exp(x), \mathrm{d}x = e - 1 = 1.71828182845905.$$

Para una tolerancia `epsabs` $= 10^{-5}$ y `epsrel` $= 10^{-8}$ obtenemos:

```
 Status code =            0
 Integral =   1.7182818284590453
 Error =  1.90767604875024572E-014
 Número de evaluaciones =          21
```

La subrutina sólo requirió de 21 evaluaciones de f y el resultado obtenido es exacto a 14 decimales, tal como lo indica la estimación de su error.

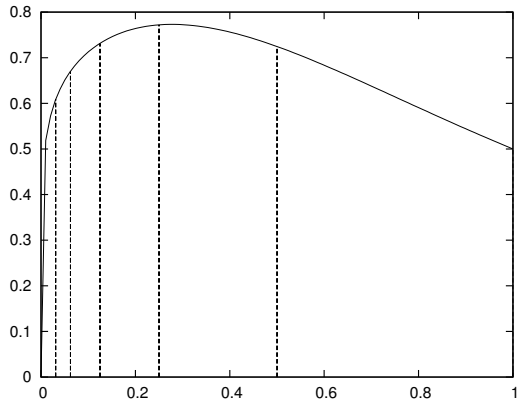

Figura 1: Cuadratura adaptativa para $\int_0^1 x^{1/7}/(x^2+1)\,\mathrm{d}x$.

Ejemplo 2. Consideremos ahora la estimación de la integral

$$\int_0^1 \frac{x^{1/7}}{x^2+1}\,\mathrm{d}x.$$

Aunque el integrando es continuo en $x = 0$, tiene una derivada infinita en dicho punto. El código adaptivo requerirá, en consecuencia, intervalos de integración pequeños en la proximidad del mismo. Para una tolerancia `epsabs` $= 10^{-5}$ y `epsrel` $= 10^{-8}$ obtenemos:

```
Status code =            0
 Integral =  0.67180003240239616
 Error =  3.43103232530158377E-012
 Número de evaluaciones =         231
```

En comparación con el ejemplo anterior, este problema requirió de más evaluaciones del integrando. El cálculo final procede con una partición del intervalo de integración $[\,0,1]$ en seis subintervalos tal como se ilustra en la figura 1.

Bibliografía

- **Fundamentals of Numerical Computing**, *L. F. Shampine, Rebecca Chan Allen, S. Pruess, R. C. Allen*, Wiley.
- **Scientific Computing**, *Michael T Heath*, McGraw-Hill.
- **QUADPACK**, *Robert Piessens, Elise deDoncker-Kapenga, Christoph W. Überhuber, David Kahaner*, `http://www.netlib.org/quadpack/`.