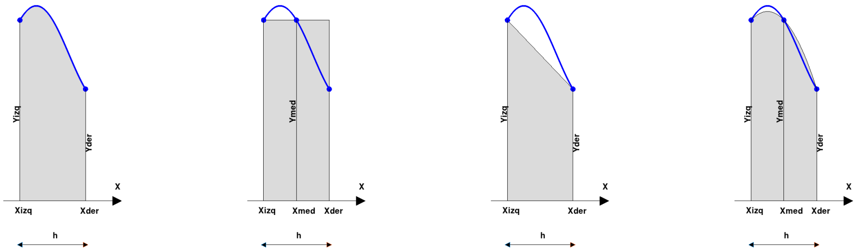
Cuadraturas



Históricamente se denominaba «*fórmulas de cuadratura*» a las expresiones que permiten calcular de forma numérica las integrales. Se va a explicar en este artículo cómo desarrollar funciones en MATLAB u Octave que permitan calcular de forma numérica integrales del tipo siguiente:

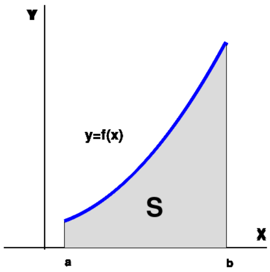
$$S = \int_{x=a}^{x=b} f(x)\, dx$$

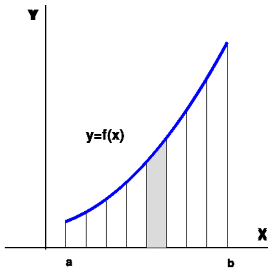
En el libro de texto se explican de manera rigurosa los métodos de integración numérica y la matemática que subyace a los mismos. Para cada método, el libro detalla la «*fórmula simple*», que es lo que aquí se llamará «área del subintervalo». También se habla en el libro de las «*fórmulas compuestas*». Las fórmulas compuestas resultan al aplicar las fórmulas simples a la totalidad del intervalo de integración, sumando el área de todos los subintervalos. Si se desarrollan esas sumas, surgen simplificaciones por anularse algunos términos positivos con otros negativos, o por poder sacar factor común de algunos elementos. Surgen así las fórmulas compuestas, que son diferentes para cada método.

Las fórmulas compuestas son de mucha utilidad cuando los cálculos se tienen que realizar utilizando calculadoras, pues reducen el número de operaciones necesarias. La desventaja de las fórmulas compuestas es que es necesario memorizar una fórmula diferente para cada método.

En este artículo se va a explicar como programar funciones en MATLAB u Octave que resuelvan los diferentes métodos de integración numérica utilizando en todos los casos el algoritmo de la suma y la correspondiente fórmula simple. Las ventajas del procedimiento son su simplicidad, que no es necesario memorizar fórmulas complicadas y que los programas son casi idénticos para todos los métodos de integración. Se sacrifica, eso sí, un poco de eficiencia en el cálculo.

Se va a desarrollar una función que reciba como parámetros la referencia f a la función que se quiere integrar, los valores a y b de los extremos del intervalo de integración y el número *n* de subintervalos en los que se va a dividir el intervalo de integración *[a, b]*; la función devolverá el valor aproximado de la integral de la función *f* en el intervalo *[a, b]* utilizando *n* subintervalos. Los programas desarrollados utilizarán los siguientes métodos de integración explicados en el curso: el método del rectángulo, el método del trapecio y el método de la parábola, también llamado de Simpson.





En la parte izquierda de la figura precedente se puede ver el esquema de la integral de la función *f(x)* en el intervalo *[a, b]*, entendida como el valor de la superficie comprendida entre la gráfica de la función y el eje *X*. La parte derecha de la figura muestra el intervalo de integración *[a, b]* dividido en varios subintervalos de la misma anchura. Se ha sombreado uno de ellos. El proceso de calcular una aproximación del valor del área bajo la curva consiste en descomponer el intervalo de integración *[a, b]* en un número *n* de subintervalos y sumar el área de todos esos subintervalos. Es simplemente un algoritmo de la suma como el que se estudió en el primer parcial. Si se llama *area_subintervalo_i* al área del subintervalo i-ésimo, el código de la función que se quiere programar será el siguiente:

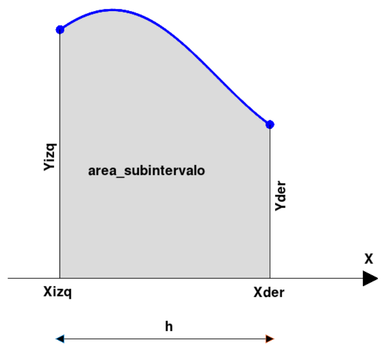
```
function s = integ(f, a, b, n)

  h = (b-a)/n;

  s = 0;
  for i = 1 : n
    s = s + area_subintervalo_i;
  end

end
```

En el código anterior se ve, en primer lugar, la signatura de la función, con la palabra clave *function* seguida del valor *s* que devuelve la función, el signo

=, el nombre elegido para la función y, por último entre paréntesis, la lista de parámetros que recibe la función para realizar sus cálculos: la referencia a la función *f* que se quiere integrar, los valores *a* y *b* de los extremos del intervalo de integración y el número *n* de subintervalos que se quieren utilizar.

Lo primero que hace la función es calcular el ancho de cada subintervalo, al que se ha llamado *h*, dividiendo el ancho total del intervalo, *(b-a)*, entre el número de subintervalos *n*.

A continuación se hace un algoritmo de la suma, con un bucle de *1* a *n*. En cada iteración del bucle, se suma a la variable *s* el área del subintervalo correspondiente. El problema está en calcular el área de cada subintervalo. La figura siguiente muestra uno de esos subintervalos:



En la figura anterior, se ha denominado *Xizq* a la abscisa del lado izquierdo del subintervalo y *Xder* a la abscisa del lado derecho. A los valores de la función *f* en esos puntos se les ha denominado *Yizq* e *Yder*, respectivamente. Se ve que el área a calcular es el encerrado entre la base de anchura *h* situada en el eje *X*, las líneas verticales que pasan por *Xizq* y *Xder*, y la parte de la función *f(x)* que cierra el subintervalo por arriba. El problema es que no hay una fórmula para calcular ese área.

Los métodos de aproximación lo que hacen es sustituir el cierre superior del subintervalo, que es la línea de la función *f(x)*, por otra línea que dé lugar a una figura de la cuál sí que sepamos calcular el área. El área de esa nueva figura no será exactamente igual al área del subintervalo original, será una aproximación, pero si se hacen el suficiente número de subintervalos, el área calculada podrá aproximarse al valor real tanto como se quiera.

En todos los métodos que vamos a ver, es necesario calcular la abscisa izquierda de cada subintervalo, que en la figura se ha llamado *Xizq*. Se muestra a continuación el proceso de cálculo de la abscisa izquierda de todos los subintervalos:

```
subintervalo 1:  Xizq = a
subintervalo 2:  Xizq = a + h
subintervalo 3:  Xizq = a + 2*h
...
subintervalo i:  Xizq = a + (i-1)*h
...
subintervalo n:  Xizq = a + (n-1)*h
```

En todos los casos, será necesario calcular el valor de la abscisa derecha de cada subintervalo y, en algunos de los métodos, hace falta conocer el valor de la abscisa del punto medio del subintervalo, que se llamará *Xmed*. En todos los subintervalos, conocida la abscisa izquierda, el cálculo será:

```
Xder = Xizq + h
Xmed = (Xizq+Xder)/2
```

Se necesitará también conocer el valor que toma la función *f(x)* en esos puntos característicos de cada subintervalo. Llamando a esos valores *Yizq*, *Ymed* e *Yder*, el cálculo es el siguiente:

```
Yizq = f(Xizq)
Ymed = f(Xmed)
Yder = f(Xder)
```

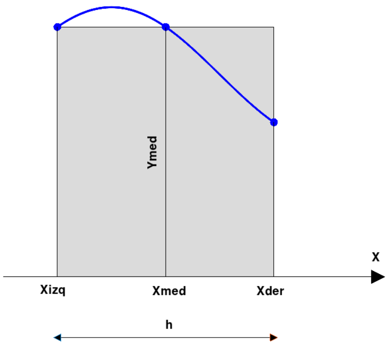
Con estos cálculos previos completados, ya se puede afinar el código de la función que se está desarrollando, que quedaría de la siguiente manera:

```
function S= integ(f,a,b,n)

  h = (b-a)/n;

  S=0;
  for i = 1:n
    xizq = a +(i-1)*h;
    xder = xizq + h;
    xmed = (xizq+xder)/2;
    yizq = f(xizq);
    yder = f(xder);
    ymed = f(xmed);
    area_subintervalo = **********
    S = S + area_subintervalo;
  end


end
```

Con esto, solo falta sustituir los asteriscos por la fórmula de cálculo del área del subintervalo, según el método de integración que se esté utilizando.

1 Método del rectángulo

En este método, la parte superior del subintervalo se cierra con una línea horizontal que pase por la *Ymed*. La figura quedaría como sigue:



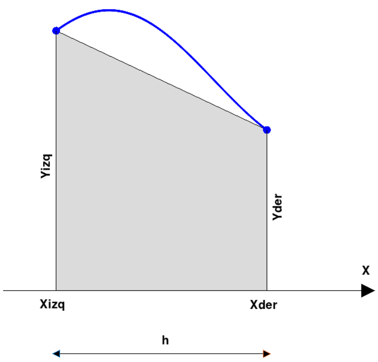
El área aproximada del subintervalo que se utiliza en el método del rectángulo es el área sombreada de la figura anterior. Su cálculo es:

```
area_subintervalo = h * Ymed;
```

Esta expresión habría que ponerla en la linea correspondiente del listado de la función, en vez de los asteriscos. A la vista de la fórmula que se utiliza para calcular el área, en el listado no harían falta las líneas que calculan la *Yizq* ni la *Yder*.

2 Método del trapecio

En el método del trapecio, la parte superior del subintervalo se cierra con una línea que pasa por la *Yizq* y la *Yder*. La figura resultante es la siguiente:



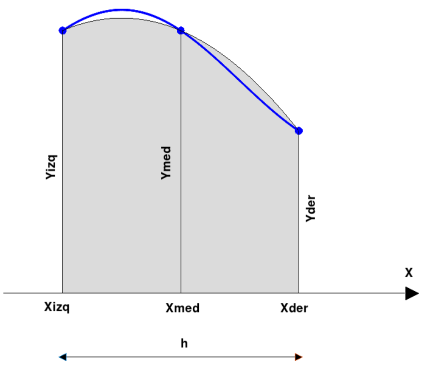
El área aproximada del subintervalo que se utiliza en el método del trapecio es el área sombreada de la figura anterior. Se trata de un trapecio. La fórmula del área del trapecio es: «*Base mayor, más base menor, partido por dos y por la altura*. En este caso, las bases son *Yizq* e *Yder*; la altura es *h*. El cálculo del area aproximada del subintervalo será:

```
area_subintervalo = (Yizq+Yder)/2 * h;
```

Esta expresión habría que ponerla en la linea correspondiente del listado de la función, en vez de los asteriscos. A la vista de la fórmula que se utiliza para calcular el área, en el listado no harían falta las líneas que calculan la *Xmed* ni la *Ymed*.

3 Método de la parábola, o de Simpson

En el método de la parábola, el subintervalo se cierra en la parte superior mediante una parábola. Para definir una parábola se necesitan tres puntos. Se utilizan los puntos *Yizq*, *Ymed* e *Yder*. Con ello, la figura queda:



El área aproximada que se utiliza es el área sombreada de la figura anterior. El cálculo de dicho área es el siguiente:

```
area_subintervalo = (Yizq + 4*Ymed + Yder)*h/6;
```

Esta es la fórmula que habrá que incluir en el listado en el lugar de los asteriscos.

4 Consideraciones finales

Se ha explicado cómo plantear una función de MATLAB-Octave para calcular una integral numérica por tres métodos diferentes. El código de las funciones utilizando los distintos métodos es casi el mismo. La diferencia está, fundamentalmente, en la formula utilizada para calcular el área aproximada del subintervalo. Lo importante es recordar la forma de plantearlo como un algoritmo de la suma en el que se suman las áreas aproximadas de los subintervalos.

Para un mismo número de subintervalos, la fórmula de la parábola es más precisa que las otras dos. Pero calculando con el ordenador, se pueden obtener la misma precisión, utilizando la fórmula del rectángulo o la del trapecio y aumentando el número de subintervalos. Como en el método del rectángulo no se necesita memorizar ninguna fórmula, a veces es el método elegido, utilizando el número de subintervalos necesarios para alcanzar la precisión requerida.

En cualquier caso, las fórmulas compuestas y los métodos que utilizan un mayor número de puntos, pueden ser más eficientes, por lo que si el programa se va a utilizar intensivamente, puede ser conveniente desarrollar funciones que los utilicen. En este caso, y si se utiliza como lenguaje de programación MATLAB u Octave, también puede ser interesante sustituir los bucles del algoritmo de la suma por operaciones vectorizadas, que son más eficientes.

5 Acerca de las figuras del artículo

Todas las figuras del artículo se han realizado mediante programas en el lenguaje *m* de MATLAB y Octave. Se ha planteado así, como recurso didáctico adicional. Los listados de estos programas se adjuntan con el artículo. Estos gráficos utilizan técnicas avanzadas de gráficos que sobrepasan el temario estudiado en el curso, pero el alumno curioso encontrará en la inspección de los listados recursos interesantes. No es necesario aprender a utilizar estos recursos, pero saber que existen puede ser útil en el futuro si al alumno le surgiera la necesidad de utilizarlos.