Prácticas con wxMaxima: cálculo integral de funciones de una variable

BENITO J. GONZÁLEZ RODRÍGUEZ (bjglez@ull.es)
DOMINGO HERNÁNDEZ ABREU (dhabreu@ull.es)
MATEO M. JIMÉNEZ PAIZ (mjimenez@ull.es)
M. ISABEL MARRERO RODRÍGUEZ (imarrero@ull.es)
ALEJANDRO SANABRIA GARCÍA (asgarcia@ull.es)

Departamento de Análisis Matemático
Universidad de La Laguna

Índice

3. Cálculo integral de funciones de una variable **1**
3.1. Integral indefinida . . . 1
3.1.1. Cálculo de integrales inmediatas . . . 1
3.1.2. Cálculo de integrales por el método de integración por partes . . . 1
3.1.3. Cálculo de integrales por el método del cambio de variables . . . 2
3.1.4. Cálculo de integrales racionales . . . 4
3.1.5. Cálculo de integrales trigonométricas . . . 4
3.2. Integral definida y aplicaciones . . . 6
3.2.1. Cálculo de integrales definidas . . . 6
3.2.2. Cálculo de áreas entre curvas . . . 7
3.2.3. Aproximación de integrales mediante sumas de Riemann . . . 8
3.2.4. Aproximación numérica de integrales . . . 10

PRÁCTICA 3: CÁLCULO INTEGRAL DE FUNCIONES DE UNA VARIABLE.

Duración estimada: 5 horas.

Objetivo: trabajar algunos conceptos relacionados con el cálculo integral de funciones de una variable (integral indefinida, integración por partes y por cambio de variables, integrales racionales y trigonométricas, integral definida, cálculo de áreas entre curvas, sumas de Riemann, métodos de aproximación numérica, etc.).

Integral Indefinida.

1 Cálculo de integrales inmediatas.

El cálculo de integrales se realiza mediante la orden "integrate". Recordar que si se antepone el símbolo "'" se obtiene la expresión en modo simbólico.

```
--> 'integrate(x*(x+1)*(x+2),x)=integrate(x*(x+1)*(x+2),x);
```

```
--> 'integrate(1/sqrt(exp(x)),x)=integrate(1/sqrt(exp(x)),x);
```

```
--> 'integrate(a*x^2+b*x+c,a)=integrate(a*x^2+b*x+c,a);
```

```
--> 'integrate(x^(1/3)+x^(1/5)+%pi-3+sqrt(2),x)=integrate(x^(1/3)+x^(1/5)+%pi-3+sqrt(2),x);
```

```
--> 'integrate(1/(3*x^2+5),x)=integrate(1/(3*x^2+5),x);
```

Veamos en el siguiente ejemplo como visualizar la integral indefinida de una función por medio de una familia de primitivas.

```
--> f(x):=x*(5-x^2)^(1/5); define(F(x),integrate(x*(5-x^2)^(1/5),x));
```

```
--> with_slider(C,makelist(i/5,i,0,15),[f(x),F(x)+C],[x,-sqrt(5),sqrt(5)],[y,-3,3],[legend,false]);
```

```
--> 'integrate(cos(x)^2,x)=integrate(cos(x)^2,x);
```

```
--> 'integrate(cos(x)^2,x)=ratsimp(integrate(cos(x)^2,x));
```

Naturalmente, debemos cuidar que el integrando tenga primitiva elemental.

```
--> 'integrate(cos(x^2),x)=integrate(cos(x^2),x);
```

Por otra parte, wxMaxima puede resolver integrales que dependan de una parámetro.

```
--> 'integrate(x^n,x)=integrate(x^n,x);
```

2 Cálculo de integrales por el método de integración por partes.

```
--> kill(all);
```

Los siguientes ejemplos corresponden a casos típicos de integración por partes.

```
--> 'integrate(log(x),x)=integrate(log(x),x);

--> 'integrate(x^3*sin(x),x)=integrate(x^3*sin(x),x);

--> 'integrate(exp(5*x)*cos(7*x),x)=integrate(exp(5*x)*cos(7*x),x);

--> integrate(f(x)*diff(g(x),x),x);
```

Veamos cómo resolver los ejemplos anteriores forzando la elección de factores en la integración por partes. Para ello cargamos previamente el paquete "antid" y usaremos la orden "antidiff".

```
--> load(antid);

--> antidiff(f(x)*diff(g(x),x),x,g(x));

--> f(x):=log(x); g(x):=x; 'integrate(f(x)*diff(g(x),x),x)=antidiff(f(x)*diff(g(x),x),x,g(x));

--> f(x):=x^3; g(x):=-cos(x); 'integrate(f(x)*diff(g(x),x),x)=antidiff(f(x)*diff(g(x),x),x,g(x));

--> f(x):=sin(x); g(x):=x^4/4; 'integrate(f(x)*diff(g(x),x),x)=antidiff(f(x)*diff(g(x),x),x,g(x));

--> f(x):=exp(5*x); g(x):=sin(7*x)/7; 'integrate(f(x)*diff(g(x),x),x)=antidiff(f(x)*diff(g(x),x),x,g(x));

--> f(x):=cos(7*x); g(x):=exp(5*x)/5; 'integrate(f(x)*diff(g(x),x),x)=antidiff(f(x)*diff(g(x),x),x,g(x));
```

3 Cálculo de integrales por el método de cambio de variables.

```
--> kill(all);

--> 'integrate(7*x^2/(5+x^6),x)=integrate(7*x^2/(5+x^6),x);
```

Con la orden "changevar(expr,t=f(x),t,x)" podemos forzar manualmente el cambio de variable deseado.

```
--> changevar('integrate(7*x^2/(5+x^6),x),t=x^3,t,x);

--> ev(%,integrate);

--> subst(t=x^3,%);
```

Podemos hacer otros cambios de variable, aunque no nos resulten prácticos.

```
--> changevar('integrate(7*x^2/(5+x^6),x),t=x^2,t,x);
```

Veamos el siguiente ejemplo trigonométrico.

```
--> 'integrate((1+cos(x)^2)*sin(x),x)=integrate((1+cos(x)^2)*sin(x),x);

--> changevar('integrate((1+cos(x)^2)*sin(x),x),t=cos(x),t,x);

--> ev(%,integrate);

--> subst(t=cos(x),%);
```

Veamos como queda el cambio trigonométrico universal sobre este último ejemplo en concreto.

```
--> changevar('integrate((1+cos(x)^2)*sin(x),x),t=tan(x/2),t,x);
```

Sabemos que este cambio siempre da lugar a una integral racional.

```
--> trigexpand(%);

--> ratsimp(%);

--> ev(%,integrate);

--> subst(t=tan(x/2),%);

--> trigsimp(%);

--> trigreduce(%);

--> trigexpand(%);

--> trigsimp(%);
```

Observamos que con este último cambio hemos obtenido una primitiva con diferente constante de integración.

```
--> 'integrate((1-log(x))/(x*log(x)),x)=integrate((1-log(x))/(x*log(x)),x);

--> changevar('integrate((1-log(x))/(x*log(x)),x),t=log(x),t,x);

--> ev(%,integrate);

--> subst(t=log(x),%);

--> kill(all);

--> 'integrate(sqrt(8-7*x^2),x)=integrate(sqrt(8-7*x^2),x);
```

Veamos cómo se obtiene esta integral a través de un cambio de variable adecuado.

```
--> changevar('integrate(sqrt(8-7*x^2),x),x=sqrt(8/7)*sin(t),t,x);
```

Usamos la orden "rootscontract" para convertir un producto de raíces en una raíz de un producto.

```
--> rootscontract(%);

--> expr:trigsimp(%);
```

Asumimos no negatividad en el coseno.

```
--> assume(cos(t)>=0);

--> trigsimp(expr);

--> ratsimp(ev(%,integrate));

--> subst(t=asin(sqrt(7/8)*x),%);

--> trigexpand(%);

--> ratsimp(%);

--> kill(all);
```

4 Cálculo de integrales racionales.

Veamos los siguientes ejemplos con funciones racionales.

```
--> 'integrate((x^2+1)/(x-1),x)=integrate((x^2+1)/(x-1),x);
```

Podemos forzar la descomposición en fracciones simples de una función racional con la orden "partfrac".

```
--> partfrac((x^2+1)/(x-1),x);

--> integrate(%,x);

--> 'integrate((x^2-5*x+9)/(x^2-5*x-6),x)=integrate((x^2-5*x+9)/(x^2-5*x-6),x);

--> partfrac((x^2-5*x+9)/(x^2-5*x-6),x);

--> integrate(%,x);

--> 'integrate(1/(x*(x+1)^2),x)=integrate(1/(x*(x+1)^2),x);

--> partfrac(1/(x*(x+1)^2),x);

--> integrate(%,x);

--> f(x):=(5*x+3)/(x^7-x^6+x^5-x^4-x^3+x^2-x+1);

--> 'integrate(f(x),x)=integrate(f(x),x);

--> define(F(x),rhs(%));

--> wxplot2d([f(x),F(x)],[x,1.25,3],[legend,"f(x)","F(x)"]);

--> limit(F(x),x,inf);

--> partfrac(f(x),x);

--> integrate(%,x);

--> ratsimp(%-F(x));
```

5 Cálculo de integrales trigonométricas.

Como ya sabemos, el tipo de integrales trigonométricas que hemos estudiado se pueden resolver mediante cambios de variable específicos, o bien mediante identidades trigonométricas clásicas.

```
--> 'integrate(cos(x)^3,x)=integrate(cos(x)^3,x);

--> changevar('integrate(cos(x)^3,x),t=sin(x),t,x);

--> ev(%,integrate);

--> subst(t=sin(x),%);

--> 'integrate(sin(5*x)*cos(7*x),x)=integrate(sin(5*x)*cos(7*x),x);
```

Veamos qué ocurre en el siguiente ejemplo tanto al resolverlo directamente, así como al forzar el cambio de variable trigonométrico

universal.

```
--> kill(all);

--> 'integrate(cos(x)/(1+cos(x)),x)=integrate(cos(x)/(1+cos(x)),x);

--> define(F(x),rhs(%));

--> trigsimp(diff(F(x),x));

--> changevar('integrate(cos(x)/(1+cos(x)),x),t=tan(x/2),t,x);

--> trigexpand(%);

--> ratsimp(%);

--> ev(%,integrate);

--> primitiva:subst(t=tan(x/2),%);

--> assume(x>=-%pi,x<=%pi);

--> ratsimp(primitiva);

--> kill(all);

--> 'integrate(tan(x)/(5+cos(x)),x)=integrate(tan(x)/(5+cos(x)),x);

--> changevar('integrate(tan(x)/(5+cos(x)),x),t=cos(x),t,x);

--> ev(%,integrate);

--> subst(t=cos(x),%);

--> kill(all);

--> 'integrate(sqrt(tan(x)),x)=integrate(sqrt(tan(x)),x);

--> changevar('integrate(sqrt(tan(x)),x),t=tan(x),t,x);

--> ev(%,integrate);

--> subst(t=tan(x),%);

--> changevar('integrate(sqrt(tan(x)),x),t^2=tan(x),t,x);

--> define(F(t),%);

--> assume(t>=0);

--> F(t);

--> ev(%,integrate);

--> subst(t=sqrt(tan(x)),%);

--> kill(all);
```

Integral definida. Aplicaciones.

1 Cálculo de integrales definidas.

El cálculo de integrales definidas se realiza análogamente al cálculo de primitivas. De hecho, basta añadir a la sentencia el intervalo de integración correspondiente.

```
--> load(draw);

--> f(x):=x*(x+1)*(x+2);

--> integrate(f(x),x,0,10);

--> grafica:explicit(f(x),x,0,10);
```

En la siguiente instrucción, la opción "filled_func=0" permite sombrear el área encerrada con OX.

```
--> wxdraw2d(fill_color=light_blue, filled_func=0,background_color=beige,key="f(x)=x*(x+1)*(x+2)",grafica);

--> wxdraw2d(fill_color=light_blue, filled_func=-f(x),background_color=beige,key="+/-f(x)",grafica);

--> f(x):=sin(x);

--> integrate(f(x),x,0,%pi);

--> grafica:explicit(f(x),x,0,%pi);

--> wxdraw2d(fill_color=light_blue, filled_func=0,background_color=beige,proportional_axes = xy,grafica);

--> f(x):=cos(x)^2;

--> integrate(f(x),x,-%pi/2,%pi/2); %,numer;

--> grafica:explicit(f(x),x,-%pi/2,%pi/2);

--> wxdraw2d(fill_color=light_blue, filled_func=0,background_color=beige,proportional_axes = xy,grafica);

--> f(x):=sin(cos(x)-1/2);

--> integrate(f(x),x,-3,3); %,numer;
```

wxMaxima dispone de ciertas rutinas, como "quad_qags" o "romberg", de cálculo numérico de integrales. En nuestro caso, nos limitamos a emplearlas para obtener una estimación de las integrales. En la sección final de esta práctica nos ocuparemos de implementar las reglas trapezoidal y de Simpson que hemos estudiado durante el curso.

```
--> quad_qags(f(x),x, -3, 3);

--> romberg(f(x),x,-3,3);

--> grafica:explicit(f(x),x,-3,3);

--> wxdraw2d(xaxis=true,fill_color=light_blue, filled_func=0,background_color=beige,proportional_axes = xy,grafica);

--> kill(all);
```

2 Cálculo de áreas entre curvas.

```
--> f(x):=x*(x-1)*(x-2);

--> grafica:explicit(f(x),x,0,2);

--> wxdraw2d(xaxis=true,fill_color=light_blue, filled_func=0,background_color=beige,proportional_axes = xy,grafica);

--> integrate(abs(f(x)),x,0,2); %,numer;
```

Vemos que, por defecto, wxMaxima no integra expresiones que involucren valor absoluto.

Sin embargo, esto puede hacerse con funciones polinómicas cargando previamente el paquete "abs_integrate".

```
--> load(abs_integrate);

--> integrate(abs(f(x)),x,0,2);

--> integrate(f(x),x,0,1)-integrate(f(x),x,1,2);

--> kill(all);
```

Veamos el siguiente ejemplo que involucra al área encerrada entre dos curvas trigonométricas.

```
--> f1(x):=2*sin(x); f2(x):=4*cos(x);

--> grafica1:explicit(f1(x),x,0,2*%pi); grafica2:explicit(f2(x),x,0,2*%pi);

--> wxdraw2d(xaxis=true,fill_color=light_blue,background_color=beige,proportional_axes = xy,grafica1,grafica2);

--> solve(f1(x)=f2(x),x);

--> load(to_poly_solve);

--> nicedummies(trigreduce(%solve(f1(x)-f2(x)=0,x)));

--> float((%pi-atan(4/3))/2-atan(2));

--> corte1:atan(2); float(f1(corte1)-f2(corte1));

--> corte2:%pi+atan(2); float(f1(corte2)-f2(corte2));
```

Observemos cuáles de las siguientes instrucciones son correctas para representar el área en las curvas.

```
--> wxdraw2d(xaxis=true,fill_color=light_blue,filled_func=0,background_color=beige,proportional_axes = xy,grafica1,grafica2);

--> wxdraw2d(xaxis=true,fill_color=light_blue,filled_func=f1(x),background_color=beige,proportional_axes = xy,grafica1,grafica2);

--> wxdraw2d(xaxis=true,fill_color=light_blue,filled_func=f2(x),background_color=beige,proportional_axes = xy,grafica1,grafica2);

--> integrate(abs(f1(x)-f2(x)),x,0,2*%pi);

--> load(abs_integrate);

--> integrate(abs(f1(x)-f2(x)),x,0,2*%pi);
```

Vemos que en este caso wxMaxima no es capaz de calcular el área directamente. Recurrimos inicialmente a las rutinas de cálculo numérico.

```
--> quad_qags(abs(f1(x)-f2(x)),x,0,2*%pi);

--> romberg(abs(f1(x)-f2(x)),x,0,2*%pi);
```

No obstante, podemos calcular el área exacta de modo manual.

```
--> area:integrate(f2(x)-f1(x),x,0,corte1)+integrate(f1(x)-f2(x),x,corte1,corte2)+integrate(f2(x)-f1(x),x,corte2,2*%pi); %,numer;

--> kill(all);
```

Finalizamos con el siguiente ejemplo que involucra a la región determinada por las gráficas de tres funciones.

```
--> f1(x):=x^2; f2(x):=x^2/2; f3(x):=2*x;

--> solve(f1(x)=f2(x)) /* Intersección de f1 y f2 */;

--> solve(f1(x)=f3(x)) /* Intersección de f1 y f3 */;

--> solve(f2(x)=f3(x)) /* Intersección de f2 y f3 */;

--> wxplot2d([f1(x),f2(x),f3(x)],[x,0,4],[legend,"f1","f2","f3"]);

--> grafica1:explicit(f1(x),x,0,2);
    grafica2:explicit(f2(x),x,0,4);
    grafica3:explicit(f3(x),x,2,4);

--> grafica4:points([0,2,4],[0,4,8]);

--> wxdraw2d(fill_color=light_blue,filled_func=f2(x),background_color=beige,grafica1, grafica2,
    grafica3,point_type=filled_circle,point_size=2,grafica4);

--> area:integrate(f1(x)-f2(x),x,0,2)+integrate(f3(x)-f2(x),x,2,4);
```

3 Aproximación de integrales mediante Sumas de Riemann.

```
--> kill(all);

--> f(x):=cos(x);

--> a:0$ b:%pi/2$

--> 'integrate(f(x),x,a,b)=integrate(f(x),x,a,b);

--> wxplot2d(f(x),[x,a,b],[legend,false]);
```

Veamos como ilustrar gráficamente la aproximación al área encerrada mediante sumas de Riemann. Comenzamos con sumas de Riemann a la derecha.

```
--> load(draw)$
```

Creamos una partición de nodos equiespaciados en [a,b], y rectangulos con base superior obtenida a partir de las imágenes de f(x) en los extremos superiores de cada subintervalo de la partición.

```
--> x(i,n):=a+i*((b-a)/n);

--> rectderecha(n):= makelist(rectangle([x(k-1,n),0], [x(k,n),f(x(k,n))]), k,1,n);
```

```
--> grafriemannderecha(n):= apply(wxdraw2d, append([xrange=[a,b], color=blue],
    rectderecha(n),[explicit(f(x),x,a,b)]) );

--> n:10;

--> grafriemannderecha(n);
```

Finalmente comparamos el área total de los rectángulos con el valor de la integral exacta. Observemos que para poder computar el error es necesario disponer de antemano de un valor exacto para la integral.

```
--> 'sumariemannderecha=sum(((b-a)/n)*f(x(i,n)),i,1,n);
    'sumariemannderecha=float(sum(((b-a)/n)*f(x(i,n)),i,1,n));
    'errorderecha=integral-float(sum(((b-a)/n)*f(x(i,n)),i,1,n));
```

Análogamente veamos como obtener sumas de Riemann a la izquierda.

```
--> kill(n);

--> rectizquierda(n):= makelist(rectangle([x(k,n),0], [x(k-1,n),f(x(k-1,n))]), k,1,n);

--> grafriemannizquierda(n):= apply(wxdraw2d, append([xrange=[a,b], color=blue],
    rectizquierda(n),[explicit(f(x),x,a,b)]) );

--> n:10;

--> grafriemannizquierda(n);

--> 'sumariemannizquierda=sum(((b-a)/n)*f(x(i-1,n)),i,1,n);
    'sumariemannizquierda=float(sum(((b-a)/n)*f(x(i-1,n)),i,1,n));
    'errorizquierda=integral-float(sum(((b-a)/n)*f(x(i-1,n)),i,1,n));
```

Suma de Riemann centrada: tomaremos las imágenes de f(x) en los puntos medios de cada subintervalo.

```
--> kill(n);

--> rectcentral(n):= makelist(rectangle([x(k-1,n),0], [x(k,n),f((x(k-1,n)+x(k,n))/2)]), k,1,n);

--> grafriemanncentral(n):= apply(wxdraw2d, append([xrange=[a,b], color=blue],
    rectcentral(n),[explicit(f(x),x,a,b)]) );

--> n:10;

--> grafriemanncentral(n);

--> 'sumariemanncentral=sum(((b-a)/n)*f((x(i-1,n)+x(i,n))/2),i,1,n);
    'sumariemanncentral=float(sum(((b-a)/n)*f((x(i-1,n)+x(i,n))/2),i,1,n));
    'errorcentral=integral-float(sum(((b-a)/n)*f((x(i-1,n)+x(i,n))/2),i,1,n));
```

Suma de Riemann inferior: se obtiene considerando como base superior de los rectángulos el valor mínimo de f(x) en los extremos de cada subintervalo.

```
--> f(x):=cos(x)^2; a:-%pi/2; b:%pi/2;

--> 'integrate(f(x),x,a,b)=integrate(f(x),x,a,b);
    integral:float(integrate(f(x),x,a,b));

--> kill(n);

--> rectinferior(n):= makelist(rectangle([x(k-1,n),0], [x(k,n),min(f(x(k-1,n)),f(x(k,n)))]), k,1,n);
```

```
--> grafriemanninferior(n):= apply(wxdraw2d, append([xrange=[a,b], color=blue],
    rectinferior(n),[explicit(f(x),x,a,b)]) );

--> n:10$ grafriemanninferior(n);

--> 'sumariemanninferior=sum(((b-a)/n)*min(f(x(i-1,n)),f(x(i,n))),i,1,n);
    'sumariemanninferior=float(sum(((b-a)/n)*min(f(x(i-1,n)),f(x(i,n))),i,1,n));
    'errorinferior=integral-float(sum(((b-a)/n)*min(f(x(i-1,n)),f(x(i,n))),i,1,n));
```

Suma de Riemann superior: se obtiene considerando como base superior de los rectángulos el valor máximo de f(x) en los extremos de cada subintervalo.

```
--> kill(n);

--> rectsuperior(n):= makelist(rectangle([x(k-1,n),0], [x(k,n),max(f(x(k-1,n)),f(x(k,n)))]), k,1,n);

--> grafriemannsuperior(n):= apply(wxdraw2d, append([xrange=[a,b], color=blue],
    rectsuperior(n),[explicit(f(x),x,a,b)]) );

--> n:10$ grafriemannsuperior(n);

--> 'sumariemanninferior=sum(((b-a)/n)*max(f(x(i-1,n)),f(x(i,n))),i,1,n);
    'sumariemanninferior=float(sum(((b-a)/n)*max(f(x(i-1,n)),f(x(i,n))),i,1,n));
    'errorinferior=integral-float(sum(((b-a)/n)*max(f(x(i-1,n)),f(x(i,n))),i,1,n));
```

Suma de Riemann aleatoria. En general podemos tomar la base superior de los rectángulos en base a puntos aleatorios en cada subintervalo.

Creamos una partición de nodos aleatoria y la guardamos en la función "z" de modo que no se vean modificados cada vez que aludimos a "xaleatorio(i,n)".

```
--> n:10$ xaleatorio(i):=a+(i-1)*(b-a)/n+random(1.0)*(b-a)/n;

--> for i:1 thru n do z(i):=xaleatorio(i);

--> rectaleatorio:makelist(rectangle([x(k-1,n),0], [x(k,n),f(z(k))]), k,1,n)$

--> apply(wxdraw2d, append([xrange=[a,b], color=blue],rectaleatorio,[explicit(f(x),x,a,b)]) );

--> 'sumariemannaleatorio=sum(((b-a)/n)*f(z(i)),i,1,n);
    'sumariemannaleatorio=float(sum(((b-a)/n)*f(z(i)),i,1,n));
    'erroraleatorio=integral-float(sum(((b-a)/n)*f(z(i)),i,1,n));
```

4 Aproximación numérica de integrales.

```
--> kill(all);
```

Maxima incorpora alguna instrucciones para el cálculo aproximado de integrales. De hecho, algunas de ellas, como "quad_qags" y "romberg" ya las hemos usado en la sección segunda de este apartado dedicado a la integral definida.

Sin embargo, las reglas trapezoidal y de Simpson que hemos estudiado durante el curso no están implementadas directamente en wxMaxima. Veamos cómo implementarlas de modo sencillo.

```
--> f(x):=cos(x);

--> a:0; b:%pi/2 /* [a,b] es el intervalo de integración */;

--> integral:integrate(f(x),x,a,b);
```

Creamos una partición de nodos equiespaciados a distancia h.

```
--> x(i,n):=a+i*h;

--> n:10; h:(b-a)/n /* n es el número de subintervalos */;
```

El valor aproximado que da la regla trapezoidal lo podemos obtener a través del siguiente bucle.

```
--> trapezoidal:float((h/2)*(f(a)+f(b)))$ /* Inicializamos el valor de la regla con los valores extremos */

    for i: 1 thru n-1 do

    trapezoidal:float(trapezoidal+h*f(x(i,n)))$

    print("La aproximación que da la regla trapezoidal con", n, "subintervalos es")$
    trapezoidal;

    print("El error cometido es")$ /* Naturalmente, este valor podrá computarse sólo si se conoce el valor exacto de la integral */
    integral-trapezoidal;
```

Veamos ahora la implementación de la regla de Simpson compuesta (en un número par de subintervalos). En el bucle correspondiente distinguimos coeficientes en nodos pares e impares.

```
--> m:5; n:2*m; h:(b-a)/n /* n es el número de subintervalos (debe ser par) */;

--> simpson:float((h/3)*(f(a)+f(b)))$

    for i: 1 thru m do

    simpson:float(simpson+4*h/3*f(x(2*i-1,n)))$

    for i: 1 thru m-1 do

    simpson:float(simpson+2*h/3*f(x(2*i,n)))$

    print("La aproximación que da la regla simpson con", n, "subintervalos es")$
    simpson;

    print("El error cometido es")$
    integral-simpson;
```

Created with wxMaxima.