Manual básico de *Maxima*

Material elaborado por Daniel Franco Leis y Esther Gil Cid.

Departamento de Matemática Aplicada I

Índice general

1. ¿Cómo utilizo este Manual? 4
2. Preguntas básicas sobre *Maxima* 4
   - 2.1. ¿Qué es y dónde lo consigo? 4
   - 2.2. ¿Por qué me va a ser útil? 4
   - 2.3. ¿Cuánto tiempo debo dedicar a *Maxima*? 5
   - 2.4. ¿Cómo funciona? 5
3. Ayuda 8
4. Simplificación de expresiones 8
5. Resolución de ecuaciones 9
6. Límites 10
7. Sucesiones 12
8. Funciones 12
   - 8.1. Funciones de una variable 12
   - 8.2. Funciones de varias variables 14
9. Derivación 14
   - 9.1. Derivación de funciones de una variable 14
   - 9.2. Derivación de funciones de varias variables 15
10. Desarrollo de Taylor 16
    - 10.1. Derivación implícita 16
11. Integración 17
12. Representación gráfica de funciones 19
13. Matrices 21
14. Métodos numéricos 22
    - 14.1. Método de la bisección 22
    - 14.2. Método de Newton 23
    - 14.3. Interpolación 24

    Bibliografía 27

1. ¿Cómo utilizo este Manual?

En este Manual hemos recogido cómo funcionan muchos de los comandos del programa *Maxima* que le serán de utilidad para la asignatura de Cálculo.

El Manual está diseñado para ser leído delante de un ordenador con el programa *Maxima* en funcionamiento. Debe introducir las órdenes que le mostramos y ver qué ocurre. Debe tratar de responder a las preguntas que se plantean antes de leer la solución y comprobar después el resultado.

Podrá seguir la mayor parte del Manual con los conocimientos previos de aritmética y cálculo que debería tener. Hemos tratado de seguir el orden en el que presentamos los conceptos en el curso, pero en ocasiones no lo hemos hecho así para favorecer una lectura fluida del Manual. El índice le resultará útil para localizar información concreta.

2. Preguntas básicas sobre *Maxima*

2.1. ¿Qué es y dónde lo consigo?

Maxima es un potente programa de *software libre* para la manipulación de expresiones numéricas y simbólicas. Es decir, *Maxima* se puede usar como calculadora, pero además nos permite trabajar con expresiones en las que aparecen variables o parámetros (como $x + y$ o $\cos(\pi x + 3)$). En particular podremos emplear el programa para calcular límites, derivadas, integrales, desarrollos en serie; resolver ecuaciones; o representar las gráficas de funciones de 1 y 2 variables.

El programa se puede descargar desde la página web

http://maxima.sourceforge.net/es/

2.2. ¿Por qué me va a ser útil?

Maxima es una herramienta que le permitirá comprender mejor los conceptos tratados en la asignatura gracias a su capacidad de cálculo y a sus posibilidades de representación gráfica. Además, le permitirá verificar sus cálculos dándole mayor seguridad.

2.3. ¿Cuánto tiempo debo dedicar a *Maxima*?

El objetivo principal de la asignatura es que adquieran los conocimientos necesarios de Cálculo Infinitesimal para poder continuar sus estudios y desarrollar su carrera profesional. Como acabamos de decir Maxima es una herramienta y estimamos que debería dedicarle entre un 5 % y un 10 % del tiempo total que dedique a la asignatura.

2.4. ¿Cómo funciona?

El entorno de ejecución de *Maxima* puede resultar duro ya que consiste en la clásica línea para ejecutar comandos. Afortunadamente existen entornos más amigables de ejecución y nosotros vamos a trabajar con wxMaxima ya que incluye menús desplegables con la mayoría de los comandos y cuadros de diálogo para introducir los datos necesarios en cada caso, que hace que sea más sencillo trabajar con *Maxima*.

Una vez lanzado el programa wxMaxima, para empezar a trabajar con él hay que pulsar la tecla `ENTER` y aparece el símbolo:

```
[>>
```

Es lo que el programa llama celda o en inglés "cell". Ahí podemos escribir una operación, por ejemplo la suma $2 + 8$ tecleando a continuación punto y coma[1].

```
[>>2+8;
```

Se pulsan `SHIFT+ENTER` simultáneamente (que es la orden de ejecución) y aparece:

```
(%i1) 2+8;
(%o1) 10
```

`(%i1)` significa que es la primera entrada (*input 1*) y `(%o1)` que es la salida correspondiente a esa entrada (*output 1*).

Si al tratar de realizar la suma anterior en su ordenador obtiene un mensaje de error, es porque el programa no está bien configurado. Para arreglar el problema, puede revisar la configuración o visitar la página web (http://wxmaxima.sourceforge.net/).

[1] En breve le explicaremos qué le indicamos a *Maxima* con el punto y coma.

Tenemos la posibilidad de escribir varios comandos dentro de una celda. Por ejemplo, podemos escribir:

```
>>  3!;
    %/3;
    %o1 * 10;
    100/3;
    100.0/3.0;
    cos(% pi);
```

El símbolo "%" tiene varios usos en la ordenes anteriores que vamos a aclarar. En la segunda línea "%" se utiliza para hacer referencia a la última salida generada por *Maxima*. En "%o1" se utiliza para hacer referencia a la salida número 1 *output 1*. Por último el "%pi" en la última línea se utiliza para hacer referencia a la constante π. Otras constantes que *Maxima* reconoce son el número e (%e) y la unidad imaginaria i (%i).

Cada entrada que le damos al programa debe terminar con ";" o con "$". Si termina con ";", *Maxima* devuelve el resultado, pero si no se desea que aparezca ningún resultado, se debe terminar la línea con $. Este símbolo es útil cuando se quieren hacer varias operaciones intermedias y no se desea que aparezcan los resultados[2].

Pregunta: *Ejecute las sentencias anteriores. Los resultados que aparecen, ¿coinciden con los resultados que esperaba obtener?*

Respuesta: Si pulsamos SHIFT+ENTER para ejecutar, aparece:

```
(%o2) 6
(%o3) 2
(%o4) 100
(%o5) 100/3
(%o6) 33.33333333333334
(%o7) -1
```

Como vemos para cada entrada aparece una salida en una línea distinta y numerada.

Obsérvese que los resultados de 100/3 y en 100.0/3.0 son distintos, aunque aparentemente la operación es la misma. Esto es porque *Maxima* intenta mantener la precisión y no evalúa expresiones como 100/3 o sqrt(2)

[2]Debe tener en cuenta que *Maxima* realiza la operación y le asigna un número de salida correspondiente con el que tenga la entrada, digamos que es N, por tanto, aunque no se muestre esa salida, puede hacer referencia a ella con %oN.

(la raíz cuadrada es `sqrt`) a no ser que se indique. En la entrada `%i6` se utiliza coma flotante y, por eso, *Maxima* evalúa el resultado[3].

Para obtener la aproximación numérica de una operación utilizamos el comando `float`:

```
>> sqrt(2 * %e);
   float( %);
```

PREGUNTA: *¿Cuál es el resultado? ¿Por qué?*

RESPUESTA: El resultado es:

```
(%o8) sqrt(2)*sqrt(%e)
(%o9) 2.331643981597125
```

En la primera sentencia, no se evalúa el resultado, se escribe el resultado simbólico exacto. Al indicar `float` en la segunda sentencia, ya sí se evalúa.

Maxima no sólo almacena los resultados de cada línea ejecutada con el símbolo numerado "`%o`". También podemos almacenar variables, con "`nombre: valor`". Por ejemplo, para calcular la longitud de una circunferencia, almacenamos el radio en la variable "`radio`" y hacemos:

```
>> /*longitud de una circunferencia y area del circulo*/
   radio:3 $
   longitud_circunferencia: float(2*%pi*radio);
```

La primera línea es un comentario, que en *Maxima* es cualquier texto entre "`/*`" y "`*/`". El símbolo ^ indica potencia (en este caso, elevado al cuadrado). Como sólo hemos pedido que nos muestre la segunda salida, el resultado es:

```
(%o11) 18.84955592153876
```

MENÚ WXMAXIMA: El entorno de ejecución wxMaxima nos permite ver (y borrar) las variables que hemos definido y su definición a través del menú superior, en *Maxima*-> Mostrar variables (Borrar variables).

[3] *Maxima* emplea la notación habitual para las operaciones aritméticas: + suma, − resta, ∗ producto y / división.

3. Ayuda

El programa *Maxima* dispone de una ayuda en castellano (si se ha instalado la versión en este idioma) a la que se accede a través del menú superior o pulsando la tecla F1. En la pantalla que aparece se pueden consultar la información de tres formas distintas.

1. **Contenido.** Nos permite navegar por el manual. la información aparece agrupada por materias y en cada materia, casi siempre después de una breve introducción, aparecen los comandos relacionados con esa materia en orden alfabético.

2. **Índice.** Es un listado de todos los comandos disponibles. Seleccionando uno nos aparece su descripción a la derecha.

3. **Buscar.** Debe introducir una palabra relacionada con lo que desea buscar y obtendrá las materias en las que la palabra proporcionada aparece.

También se accede a la ayuda escribiendo "`?`" (para búsqueda exacta) o "`??`" (para búsqueda aproximada), seguido del término a buscar (y sin "`;`" al final, pero pulsando `SHIFT+ENTER`). A partir de la respuesta, podemos seleccionar el item (o items) que deseemos consultar y teclear `SHIFT+ENTER` y aparece la ayuda correspondiente.

4. Simplificación de expresiones

La sentencia de *Maxima* para factorizar números y polinomios es `factor()` y `expand()` es la instrucción para expandir una expresión. Por ejemplo, para factorizar el número 24 y el polinomio $13x^2 - x + 2x^3 - 8x^4 - x^5 + x^6 - 6$, y expandir $(x^3 - 1)^4$ hacemos:

```
>> factor(24);
   factor (x^6-x^5-8*x^4+2*x^3+13*x^2-x-6);
   expand((x^3-1)^4);
```

Pregunta: *Ejecuta estas sentencias. ¿Cuál es el resultado? Si lo hubieras hecho manualmente, ¿es sencillo o complejo? ¿Lleva mucho tiempo?*

Respuesta: El resultado es

```
(%o12)  2^3*3
(%o13)  (x-3)*(x-1)^2*(x+1)^2*(x+2)
(%o14)  x^12-4*x^9+6*x^6-4*x^3+1
```

Hacerlo manualmente no es complejo pero lleva bastante tiempo, ya que tenemos que dividir 24 entre sus factores primos, aplicar la Regla de Ruffini entre los divisores de 6 para factorizar el polinomio y aplicar el Binomio de Newton para expandir $(x^3-1)^4$.

MENÚ WXMAXIMA: Para factorizar: Simplificar->Factorizar expresión. Para expandir: Simplificar->Expandir expresión. En ambos casos, se debe escribir en la línea anterior la expresión o el número a factorizar y ejecutar.

MENÚ WXMAXIMA: Se debe escribir en la línea anterior la expresión o el número a factorizar y ejecutar. A continuación, seleccionar la opción deseada en el menú inferior.

5. Resolución de ecuaciones

Para resolver una ecuación utilizamos el comando `solve` que tiene dos argumentos `solve(ecuacion,variable)`. Por ejemplo

```
>> solve(a*x^2 + b*x + c = 0, x);
   solve(x^2 - x - 1 = 0, x);
```

nos da la solución general de una ecuación de segundo grado y la solución de la ecuación $x^2-x-1=0$, expresando el resultado con raíces y fracciones. Si queremos evaluarlo, en la línea siguiente escribimos `float(%)` y ejecutamos, pero también podíamos haber escrito la sentencia entre los paréntesis de `float()`.

Si el resultado es un número complejo, la unidad imaginaria se indica como ya hemos dicho con `%i`. Por ejemplo, al ejecutar

```
solve(x^2 - x + 1 = 0, x);
```

aparecerá como solución un par de números complejos ¿Qué pasaría si escribiéramos `float` tras esta sentencia?

PREGUNTA: *¿Cómo se resuelve una ecuación a través del Menú superior y del inferior?*

RESPUESTA: En el Menú superior se elige: Ecuaciones -> Resolver y se escribe la ecuación y la variable en la que se resuelve. En el Menú inferior, se escribe la ecuación y se elige resolver.

La sentencia que permite resolver sistemas de ecuaciones es también `solve`. Su sintaxis es `solve([ecuaciones],[variables])`:

```
>>  eq1: x+y-2*z=1;
    eq2: 2*x+y=3;
    eq3: x+y+z=0;
    solve([eq1,eq2,eq3],[x,y,z]);
```

MENÚ WXMAXIMA: Para resolver un sistema, se elige Ecuaciones->Resolver sistema lineal o Ecuaciones->Resolver sistema algebraico.

Los sistemas lineales de ecuaciones se resuelven la sentencia `linsolve([ecuaciones],[variables])`:

```
>>  linsolve([2*x-y+z=0, 3*y+2*x=0], [x,y]);
```

MENÚ WXMAXIMA: Con wxMaxima se elige Ecuaciones->Resolver sistema lineal. Se introducen a continuación el número de ecuaciones y tras aceptar en esta ventana, se introducen las ecuaciones.

Para calcular raíces aproximadas utilizamos `find_root(f(x), x, a,b)`, donde a y b son los extremos del intervalo donde se busca la raíz.

```
>>  find_root(cos(x)-x^2+1, x, -2, 0);
```

MENÚ WXMAXIMA: Con wxMaxima se elige Ecuaciones->Calcular raíces, indicando la ecuación a resolver, la variable y los extremos del intervalo en el que se busca la solución.

6. Límites

Es posible evaluar un límite con la sentencia "`limit(expresion, variable, punto)`". Por ejemplo, si escribe

```
[>>limit(sin(x)/x,x,0);
```

el resultado es

```
[(%o15) 1
```

Si el límite es indeterminado aparece "`ind`", que es una constante de *Maxima* para indeterminaciones acotadas. Si es indefinido aparece "`und`". Si es $+\infty$ aparece "`inf`" o "∞"y si es $-\infty$ aparece "`minf`" o "-∞".

PREGUNTA: *Para las siguientes sentencias*

```
>>  limit(sin(1/x), x, 0);
    limit((-1)^n*n, n, inf);
    limit(-x,x,inf);
    limit(abs(1/x),x,0);
    limit((1+x)^x, x, 0);
```

¿cómo se escriben estos límites en lenguaje matemático? ¿Sabe calcularlos? ¿Cuál es el resultado de Maxima?

RESPUESTA:

```
(%o16)  ind
(%o17)  und
(%o18)  -∞
(%o19)  ∞
(%o20)  1
```

También es posible evaluar límites por la derecha o la izquierda con la sentencia "`limit(expresion, variable, punto, direcion)`". La nueva variable `direccion` admite los argumentos `plus`, para límites por la derecha, y `minus`, para límites por la izquierda. ¿Sabría calcular el límite por la derecha y la izquierda de la función $f(x) = |x|/x$ en $x = 0$ utilizando Maxima?

MENÚ WXMAXIMA: Para calcular un límite con el menú superior, tenemos que seleccionar Análisis-> Calcular límite.... Nos aparece una ventana donde tenemos que indicar la expresión, la variable y el punto.

7. Sucesiones

En *Maxima* se pueden definir sucesiones a_n a través de su término general utilizando "`:=`", y con el subíndice entre corchetes ("`[]`"). También se puede definir mediante una expresión de recurrencia, utilizando "`:`" para asignar valores a los primeros elementos.

```
>>  a[n]:=(1-3/n^2)^n;
    b[1]:1;
    b[n]:=1/(1+b[n-1]);
```

Como curiosidad, indicamos que el límite de la sucesión b_n es la razón áurea $\phi = \frac{1+\sqrt{5}}{2}$.

PREGUNTA: *¿Cómo calculamos el límite de la sucesión a_n cuando n tiende a ∞?*

RESPUESTA: Hacemos:

```
[>>limit(a[n],n,inf);
```

Este límite es:

```
[(%o21) 1
```

8. Funciones

8.1. Funciones de una variable

Se puede definir una función dándole un nombre, indicando entre paréntesis la variable dependiente y utilizando "`:=` ") antes de dar su expresión. Si queremos evaluarla debemos escribir el nombre de la función y entre paréntesis el punto. Por ejemplo

```
>>  f(x):=x^3-x^2+3$
    g(x):=x^3-x^2+k$
    f(-2);
    f(1/2);
    g(-2);
    g(-2), k=-1;
    g(f(0)), k=0;
    g(f(x));
```

Obsérvese que hemos definido la función g con un parámetro (k) y que si queremos evaluarla dando un valor a k, entonces debemos pedirlo, indicando

el valor de k a continuación y separándolo con una "`,`".

PREGUNTA: *Ejecute las sentencias anteriores. ¿Cuál es el resultado?*

RESPUESTA:

```
(%o22)  -9
(%o23)  23/8
(%o24)  k-12
(%o25)  -13
(%o26)  18
(%o27)  (x^3-x^2+3)^3-(x^3-x^2+3)^2+k
```

Fíjese en que se pueden componer funciones y evaluarlas, tal y como hemos hecho en las dos últimas sentencias.

MENÚ WXMAXIMA: podemos ver (y borrar) las funciones que hemos definido y su definición a través del menú superior, en *Maxima* -> Mostrar funciones (borrar funciones) y *Maxima* -> Mostrar definiciones.

También se pueden definir funciones a trozos con `if` (si) *condición* `then` (entonces) *expresión1* `else` (en otro caso) *expresión2*[4]:

```
>>  f(x):= if x<=0 then x^3-x^2+3 else exp(x);
    [f(-2),f(0),f(4)];
```

En este caso, hemos definido

$$f(x) = \begin{cases} x^3 - x^2 + 3, & x \leq 0, \\ e^x, & x > 0, \end{cases}$$

y luego hemos pedido que se construya un vector con $f(-2)$, $f(0)$ y $f(4)$.

Sin embargo, no es posible emplear (en *Maxima*) las funciones definidas a trozos para calcular límites, derivadas, integrales, etc. (aunque sí se pueden representar gráficamente). Para resolver este problema, tenemos que definir por separado cada una de las ramas que forman la función a trozos y así calcular límites, derivar o integrar.

PREGUNTA: *Ejecútense las sentencias anteriores y obsérvese el resultado. Intentar calcular el límite de* $f(x)$ *cuando* $x \to 0$ *y cuando* $x \to 3^+$.

RESPUESTA:

```
(%o28)  f(x):=if x<=0 then x^3-x^2+3 else exp(x)
(%o29)  [-9,3,%e^4]
```

Para calcular los límites tenemos que hacer:

[4] Puede ver otros tipos de condicionales con `if` en la Ayuda.

```
>> limit(f(x),x,0);
   limit(f(x),x,3,plus);
```

y resulta:

```
(%o30)  lím_{x→0} if x<=0 then x^3-x^2+3 else %e^x
(%o31)  lím_{x→3+} if x<=0 then x^3-x^2+3 else %e^x
```

8.2. Funciones de varias variables

La definición de funciones descrita en la sección anterior también es válida para **funciones de varias variables**:

```
>> f(x,y):=x^3-y^2-1$
   f(-2,1);
```

El resultado de esta ejecución sería:

```
(%o33) -10
```

9. Derivación

9.1. Derivación de funciones de una variable

Se puede derivar utilizando la sentencia "`diff(expresión o función, variable)`":

```
>> f(x):=exp(x)+x^3$
   diff(f(x),x);
   diff(sin(y),y);
   g(y):=cos(y);
   diff(g(f(x)),x);
```

PREGUNTA: *A partir de las sentencias anteriores, ¿Conoce Maxima la regla de la cadena? ¿Cómo se calcularía la derivada de* $f(g(y))$*?*

RESPUESTA: Sí, porque esto es lo que se hace precisamente en la última sentencia. El resultado es:

```
(%o34) %e^x+3x^2
(%o35) cos(y)
(%o36) g(y):=cos(y)
(%o37) -(%e^x+3x^2)sin(%e^x+x^3)
```

Obsérvese que aunque la potencia de e se escribe como `exp`, cuando *Maxima* escribe la función lo hace como `%e`^{x}.

La derivada de $f(g(x))$ se calcularía como:

```
[>>diff(f(g(y)),y);
```

y obtendríamos

```
[(%38) -%e^cos(y)sin(y)-3cos(y)^2sin(y)
```

Si no se indica variable respecto a la que derivar, *Maxima* utiliza el símbolo `del(variable)` para indicar variable respecto a la que deriva. Por ejemplo, si escribimos

```
[>>diff(exp(x)+x^3);
```

el resultado será:

```
[(%o39) (%e^x+3*x^2)*del(x)
```

que utilizando la notación habitual en Matemáticas sería $(e^x + 3x^2)dx$.

Maxima también calcula derivadas segunda (o n-ésima), indicando el orden como la última variable: `diff(f(x),x,2);` (o `diff(f(x),x,n)`).

PREGUNTA: *¿Cómo calculamos la derivada cuarta de* $e^x + 3x^2$ *?*

RESPUESTA: Tenemos que hacer :

```
>>      diff(%e^x+3*x^2,x,4);
(%o38)  %e^x
```

9.2. Derivación de funciones de varias variables

Las derivadas parciales se calculan con el mismo comando que acabamos de ver pero indicando la variable respecto a la que se deriva y cuántas veces. Por ejemplo:

```
[>> diff(f(x,y,z),z,2)
```

calcula la derivada

$$\frac{\partial^2}{z^2} f(x, y, z).$$

Y

```
[>> diff(f(x,y,z),x,2,y,1,z,3)
```

calcula la derivada

$$\frac{\partial^6}{\partial x^2 \partial y \partial z^3} f(x, y, z).$$

10. Desarrollo de Taylor

Maxima permite calcular desarrollos de Taylor de una función respecto a una `variable` (x) en `punto` (a) hasta $(x-a)^{grado}$ con el comando `taylor(funcion, variable, punto, grado)`. Por ejemplo, si queremos el desarrollo de Taylor de $f(x) = \cos(x)$ en $x = \pi$ hasta orden 5, escribimos:

```
[>>taylor (cos(x),x,%pi,5);
```

El resultado es

$$\texttt{[(\%o42) 1-}\frac{\texttt{(x-\%pi)}^2}{2}\texttt{+}\frac{\texttt{(x-\%pi)}^4}{24}\texttt{+...}$$

Obsérvese que indica con "..." los términos que no ha calculado.

PREGUNTA: *¿Cómo se pide a Maxima que la expresión anterior esté escrita como un polinomio?*

RESPUESTA: Con `expand(%)`.

MENÚ WXMAXIMA: wxMaxima nos permite derivar a través del menú superior, en Análisis -> Derivar.... Una vez ejecutado este menú, aparece una ventana donde se nos preguntan distintos datos, como función, variable y orden.

10.1. Derivación implícita

Para derivar funciones implícitamente tenemos que declarar la dependencia de una variable respecto a la otra (por ejemplo, de y respecto de x), para evitar que sea considerada como constante. A continuación, podemos calcular la derivada:

```
>>  depends(y,x)$
    diff(x^2*y-y^3=cos(x*y),x);
```

Si no hubiéramos indicado variable de derivación en este caso, *Maxima* tendría en cuenta la dependencia ya definida y derivaría respecto a todas las variables:

```
>>      diff(x^2*y=y^3);
(%o41)  x^2*del(y)+(x^2*d/dx y+2*x*y)*del(x)=3*y^2*del(y)
                +3*y^2*d/dx y*del(x)
```

que en nuestra notación habitual es

$$x^2dy + \left(x^2\frac{dy}{dx} + 2xy\right)dx = 3y^2dy + 3y^2\frac{dy}{dx}dx.$$

11. Integración

Las integrales definidas e indefinidas en *Maxima* se calculan con el comando `integrate(función, variable, {límite 1 de integración, límite 2 de integración})`. Los límites de integración son opcionales y sólo cuando se quiere calcular una integral definida:

```
>>  integrate(1/(1+x^2),x);
    integrate(cos,x,0,%pi);
```

El resultado es

```
(%o42) atan(x);
(%o43) 0;
```

El resultado es una expresión sin `integrate` sólo si *Maxima* tiene éxito en el cálculo. En otro caso, la respuesta es la forma nominal de la integral (esto es, el operador `integrate` precedido de apóstrofo) o una expresión que contiene una o más formas nominales. *Maxima* sólo opera con funciones que son integrables en términos de funciones elementales, como las racionales, trigonométricas, logarítmicas, exponenciales, radicales, etc., y unas pocas extensiones de éstas.

A veces, *Maxima* necesita información adicional para evaluar una expresión y nos la pregunta. En este caso, aparece un cursor automáticamente indicando que tenemos que responder a una pregunta. Como ejemplo, ejecuta la siguiente instrucción:

```
[>> integrate (1/(x ^2+a),x);
```

Podemos responder "`p`" en vez de "`positive`". A continuación tenemos que pulsar `SHIFT-ENTER`. También se puede informar a priori a *Maxima* con `assume` y revertir esta situació con "`forget`":

```
assume (a>0)$
integrate (1/(x ^2+a),x);
forget (a>0)$
```

Debe saber que el comando `integrate` intentará la integración por partes sólo en casos especiales.

PREGUNTA: *Calcula las siguientes integrales sin y con Maxima:*

$$\int \frac{x^3-2x^2+1}{x^2+x-1}dx \quad \int \frac{\operatorname{sen} 2x-\cos x}{\operatorname{sen} x}dx \quad \int_{-1}^{\pi} \frac{\operatorname{sen} 2x-\cos x}{\operatorname{sen} x}dx$$

¿Cuál es el resultado?

RESPUESTA:

Se calculan mediante:

```
>>  integrate ((x^3-2*x^2+1)/(x^2+x-1),x);
    integrate ((-cos(x)+sin(2*x))/sin(x),x);
    integrate ((-cos(x)+sin(2*x))/sin(x),x,-1,%pi);
```

El resultado es

```
(%o44)  -4*log((2*x-sqrt(5)+1)/(2*x+sqrt(5)+1))/sqrt(5)+2*log(x^2+x-1)+(x^2-6*x)/2
(%o45)  2sin(x)-log(sin(x))
        Integral is divergent
         -- an error. To debug this try debugmode(true);
```

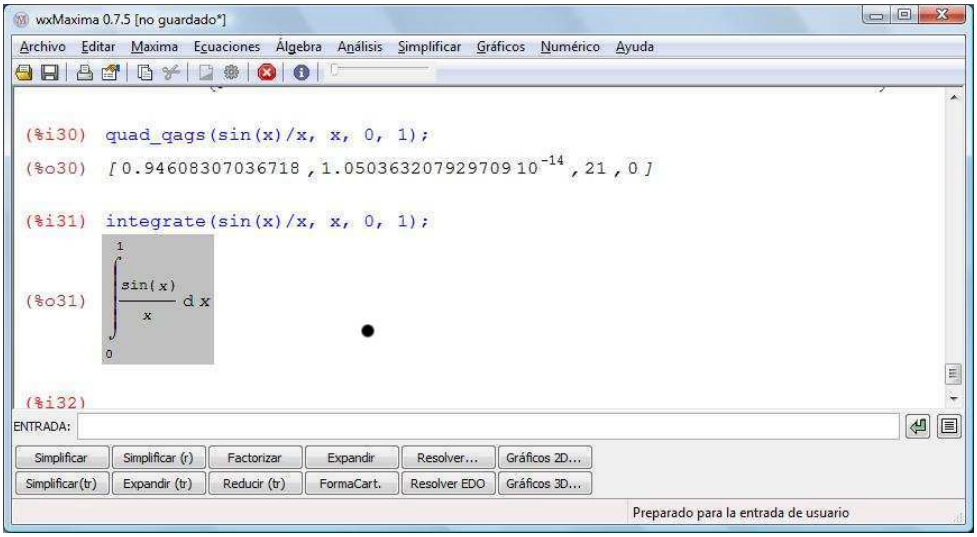
Fíjese que la última integral es impropia y como no converge, nos da un mensaje de error.

MENÚ WXMAXIMA: a través del menú superior, en Análisis -> Integrar.... Una vez ejecutado este menú, aparece una ventana donde se nos preguntan la función, variable, si es definida y método de integración. También nos permita cambiar variables eligiendo Análisis -> Cambiar variable. Entonces aparece una ventana donde tenemos que indicar integral, variable antigua, nueva variable y la ecuación que las relaciona.

Si *Maxima* no puede resolver una integral definida, siempre podemos recurrir a métodos numéricos de integración, implementados en *Maxima*. El comando `quad_qags`, tiene la misma estructura que el comando `integrate`, pero realiza integración numérica y devuelve una lista de cuatro elementos:

- La aproximación a la integral,
- El error absoluto estimado de la aproximación,
- El número de evaluaciones del integrando,
- Un código de error[5].

[5]Tal y como se indica en el Manual del programa accesible desde el menú, el código de error puede tener los siguientes valores: 0 si no ha habido problemas; 1 si se utilizaron demasiados intervalos; 2 si se encontró un número excesivo de errores de redondeo; 3 si el integrando ha tenido un comportamiento extraño frente a la integración; 6 si los argumentos de entrada no son válidos.



Las funciones para la integración numérica y el cálculo de la integral definida funcionan de forma muy distinta tal y como muestra la siguiente imagen:

12. Representación gráfica de funciones

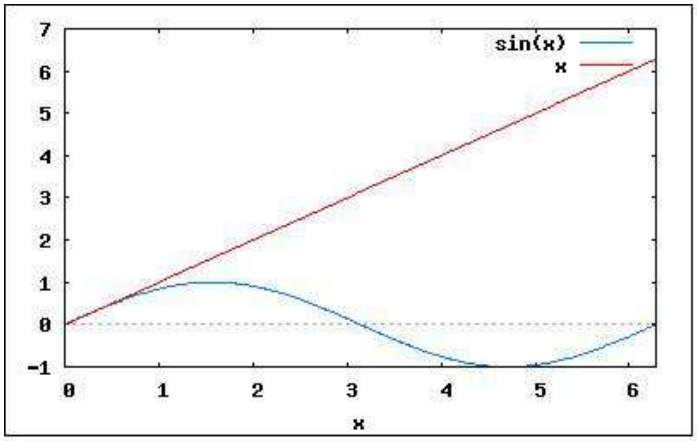
Hagamos un gráfico de funciones de una y dos variables (en 2 y 3 dimensiones). Se utilizan las siguientes sentencias:

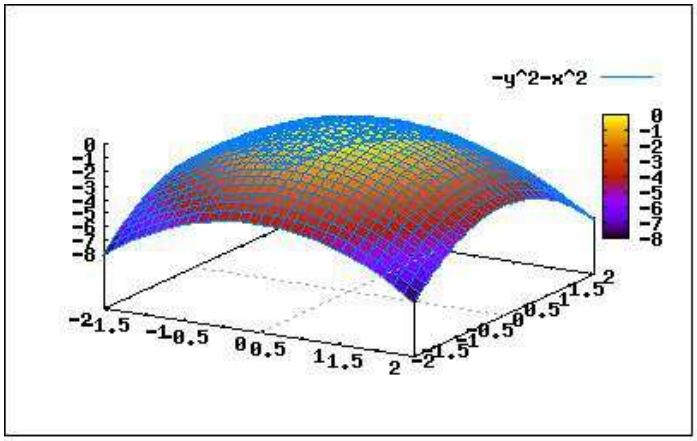
```
>> wxplot2d([sin(x), x], [x,0, 2*%pi]);
   wxplot3d( (-x^2 - y^2), [x,-2,2],[y,-2,2]);
```

En la primera línea se pide que se representen las funciones $\operatorname{sen} x$ y x, donde la variable es x, entre 0 y 2π. En la segunda línea hacemos un gráfico de una función de 2 variables $(-x^2-y^2)$, donde x e y varían de -2 a 2. Se pueden añadir otras muchas opciones, pero por su complejidad, recomendamos utilizar el Menú superior.

PREGUNTA: *¿Qué gráficas aparecen al ejecutar las sentencias anteriores?*

RESPUESTA: Aparecen





MENÚ WXMAXIMA: Para representar gráficos a través del menú superior, elegimos plot->Gráficos 2D ó Gráficos 3D. Entonces aparece una ventana donde tenemos que indicar expresión o expresiones (separadas por "," si son varias), rango de las variables x e y, graduaciones o cuadrícula (número de puntos calculados), formato (es interesante la opción gnuplot, porque aparece en una nueva ventana, donde se puede girar la imagen en 3D), opciones y si se desea guardar en un archivo. Si se elige esta última opción, sólo nos permite grabar como un fichero eps. Pero si seleccionamos la imagen y pulsamos el botón derecho del ratón, podemos guardar con otros formatos.

ACTIVIDAD: *Ejecuta la sentencia anterior a través del menú superior. Practica y familiarizate con las distintas opciones.*

13. Matrices

En *Maxima* se definen los vectores como una lista de números, escribiéndolos entre corchetes. Permite operar con ellos

Por ejemplo podemos definir $u = (1, 0, -1), v = (1, 1, -1)$ y calcular $u + v, 3v$.

```
>> u:[0,1,-1];
   v:[1,1,-1];
   u+v;
   3*v;
```

Si escribimos `u*v` se multiplica término a término, El producto escalar se indica con ".". Recomendamos que ejecute estas sentencias.

```
>> u*v;
   u.v;
```

Las matrices se escriben a partir de sus filas, indicando cada fila como una fila, con la sentencia `matrix`. Por ejemplo, podemos definir:

$$A = \begin{pmatrix} 1 & 2 & -1 \\ 0 & 1 & 1 \\ 3 & -2 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} -2 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 2 & 1 \end{pmatrix},$$

que escribimos en `Maxima` como

```
>> A:matrix([1,2,-1],[0,1,1],[3,-2,1]);
   B:matrix([-2,0,0],[0,1,0],[-1,2,1]);
```

MENÚ WXMAXIMA: Para introducir los elementos de una matriz, se elige Álgebra->Introducir matriz y se introduce primero sus dimensiones y luego los elementos de la matriz. Nótese que un vector es una matriz de dimensiones $1 \times n$.

Se pueden sumar, restar, multiplicar por escalares de la forma usual. El producto de matrices se indica con ".", mientras que "*"indica el producto término a término de matrices:

```
>> A+B;
   A-2*B;
   A.B;
   A*B;
```

Puede ser útil la sentencia matrix_size, que nos da las dimensiones de una matriz.

```
>> matrix_size(A);
```

Maxima permite multiplicar vectores con matrices. Al hacerlo, con ".", *Maxima* toma traspuestos de los vectores si fuera necesario, como se puede observar si se ejecutan las sigueintes sentencias:

```
>>  A.v;
    u.B;
```

MENÚ WXMAXIMA: Para determinar la matriz traspuesta de una dada se elige Algebra->Trasponer matriz.

Con la sentencia `^^n` el resultado es que la matriz se multiplica consigo misma n veces, mientras que si queremos que cada elemento de la matriz se eleve a n, se utiliza `^n`:

```
>>  A^^2;
    B^2;
```

Además, vamos a necesitar realizar algunas otras operaciones con matrices, como calcular su inversa o su determinante, sus autovalores y sus autovectores o determinar su rango.

Para calcular el rango se utiliza `rank`, el determinante con `determinant` y la inversa se determina con `invert`:

```
>>  rank(A);
    determinant(A);
    invert(A);
```

MENÚ WXMAXIMA: El determinante de una matriz y su inversa se determinan en Algebra->Determinante y Algebra->Invertir matriz.

Los valores propios o autovalores de una matriz se determinan con `eigenvalues`, los auto vectores o vectores propios con `eigenvectors` y el polinomio característico (polinomio cuyas raíces son los valores propios) con `charpoly(matriz,variable)`:

```
>>  A:matrix([-2,0,-4],[3,2,3],[3,0,3]);
    eigenvalues(A);
    eigenvectors(A);
    charpoly(A,lambda);
```

MENÚ WXMAXIMA: Los valores propios, vectores propios y polinomio característico se determinan en Álgebra->Valores propios, Álgebra->Vectores propios y Álgebra->polinomio característico, respectivamente.

14. Métodos numéricos

14.1. Método de la bisección

La función `find_root (expr, x, a, b)` utiliza el método de la bisección para resolver ecuaciones (aunque si la función es suficientemente suave, puede

aplicar el método de *regula falsi*, que no tratamos en la asignatura **Cálculo**). El comando aproxima una raíz de la expresión `expr` o de una función f en el intervalo cerrado $[a, b]$. Espera que la función tenga signos diferentes en los extremos del intervalo (si no es así, da un mensaje de error). Como ejemplo, buscamos una solución aproximada de la ecuación $f(x) = \log(x) - 2$ en el intervalo $[7, 9]$:

```
>> find_root (log(x)-2,x,7,9);
```

`log` es el logaritmo neperiano en *Maxima*.

Tiene otras variables opcionales que indican cuánto debe terminar el proceso, porque están relacionadas con el error relativo y absoluto. Se pueden consultar en la ayuda.

PREGUNTA: *¿Qué ocurre si busca una solución de la función* $x^4 - x^2 - 1$, *en el intervalo* $[-2, 2]$*? ¿Tiene alguna raíz? ¿Por qué da este mensaje?*

RESPUESTA: Si ejecutamos

```
[>>find_root (x^4-x^2-1,x,-2,2);
```

da el siguiente mensaje de error:

```
function has same sign at endpoints
[f(-2.0)=11.0,f(2.0)=11.0]
 -- an error. To debug this try debugmode(true);
```

Sin embargo, la función tiene dos soluciones en este intervalo (lo sabemos si la representamos gráficamente con:

```
[>>wxplot2d([x^4-x^2-1, x], [x,-2, 2]);.
```

Pero como tiene el mismo valor en los dos extremos del intervalo, no puede calcular la solución con el método de la bisección. Para poder aplicar este método, tendríamos que haberlo hecho, por ejemplo, en los intervalos $[-2, 0]$ y $[0, 2]$.

14.2. Método de Newton

Para resolver una ecuación con el método de Newton, utilizamos la función `newton (expresion, x, x_0, eps)`. El resultado es una raíz aproximada de `expresion`, en la variable `x`. Comienza en `x_0` y el proceso sigue hasta que se cumple que |`expresión`| <`eps`. Antes hay que ejecutar `load(newton1)` para cargar esta función[6].

[6] Al lanzar el programa no se cargan todos los comandos disponibles para optimizar el uso de la memoria de nuestro ordenador. Existen paquetes que añaden muchas funciones específicas y que se cargan de forma sencilla mediante la instrucción `+load(nombre del paquete)+`.

Como ejemplo, buscamos una solución de $x^4 - x^2 - 1$, comenzando en $x_0 = 1$ y con un error menor que $1/100$:

```
>> load (newton1);
   newton(x^4-x^2-1, x, 1, 1/100);
```

Obsérvese que es la función una de cuyas soluciones buscábamos sin éxito con el método de la bisección en $[-2, 2]$.

PREGUNTA: *¿Qué ocurre si comenzamos en* $x_0 = 0$, $x_0 = -1$ *y* $x_0 = 4$*?*

RESPUESTA: Si comenzamos en $x_0 = 0$, tenemos que escribir (no es necesario cargar de nuevo el paquete `newton1`):

```
[>> newton(x^4-x^2-1, x, 0, 1/100);
```

y obtenemos el siguiente mensaje de error:

```
Division by 0
#0: newton(exp=x^4-x^2-1,var=x,x0=0,eps=1/100)(newton1.mac
line 8)
 -- an error. To debug this try debugmode(true);
```

Si ejecutamos

```
[>> newton(x^4-x^2-1, x, -1, 1/100);
```

obtenemos la solución menor que 0 : $x = -1{,}272047017733587$.

Si ejecutamos

```
[>> newton(x^4-x^2-1, x, 4, 1/100);
```

obtenemos de nuevo la solución mayor que 0.

14.3. Interpolación

El paquete `interpol` permite interpolar en *Maxima* un conjunto de datos con una función lineal a trozos, un polinomio y splines cúbicos.

Vamos a considerar el conjunto de nodos y el valor de la función a interpolar en ellos:

x	1	3	4	7	13
y	3	4	3	5	6

Para interpolar con cualquiera de los tres métodos anteriores en *Maxima* primero tenemos que cargar el paquete `interpol` y a continuación indicar cuáles son los datos:

```
>> load (interpol)$
   datos:[[1,3],[3,4],[4,3],[7,5],[13,6]];
```

Para determinar una función, definida a trozos, que valga el valor dado en cada uno de los nodos y que entre 2 nodos sea una función lineal, utilizamos `linearinterpol`:

```
>> f(x):=linearinterpol(datos);
```

Hemos llamado f a esta función (si no lo hacemos, el resultado es una función definida a trozos). El resultado es la función definida por

$$\begin{array}{l}\texttt{f(x):=(}\frac{\texttt{x}}{\texttt{2}}\texttt{+}\frac{\texttt{5}}{\texttt{2}}\texttt{)charfun2(x,-}\infty\texttt{,3)+(}\frac{\texttt{x}}{\texttt{6}}\texttt{+}\frac{\texttt{23}}{\texttt{6}}\texttt{)charfun2(x,7,}\infty\texttt{)}\\ \texttt{+(}\frac{\texttt{2x}}{\texttt{3}}\texttt{+}\frac{\texttt{1}}{\texttt{3}}\texttt{)charfun2(x,4,7)+(7-x)charfun2(x,3,4)}\end{array}$$

donde `charfun2(x,a,b)` es la función que vale 1 si x está en el intervalo $[a, b)$ y 0 en el resto.

El polinomio de interpolación se obtiene mediante `lagrange (datos)`. Otras opciones se pueden encontrar en la ayuda de *Maxima*. Por ejemplo, el polinomio que interpola `datos` se obtiene mediante:

```
>> lagrange(datos);
```

El resultado es el polinomio determinado según el método de Lagrange. Por eso, su expresión no está desarrollada y resulta un poco larga.

PREGUNTA: *¿Qué debemos escribir si queremos que su expresión sea en forma* $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$*?*

RESPUESTA: La primera expresión del polinomio es:

$$\begin{array}{l}\dfrac{(x-7)\,(x-4)\,(x-3)\,(x-1)}{1080} - 5\dfrac{(x-13)\,(x-4)\,(x-3)\,(x-1)}{432}\\ + \dfrac{(x-13)\,(x-7)\,(x-3)\,(x-1)}{27} - \dfrac{(x-13)\,(x-7)\,(x-4)\,(x-1)}{20}\\ + \dfrac{(x-13)\,(x-7)\,(x-4)\,(x-3)}{144}\end{array}$$

Para desarrollar y agrupar términos utilizamos

```
>> expand(%);
```

y obtenemos:

$$-\frac{x^4}{60} + \frac{29x^3}{72} - \frac{107x^2}{36} + \frac{563x}{72} - \frac{67}{30}.$$

El tercer método implementado en el paquete es el de los splines cúbicos. Consiste en calcular el polinomio de interpolación de grado 3 entre dos nodos consecutivos, imponiendo condiciones a la derivada en las uniones, para que la curva total resultante sea suave. La sentencia que utiliza *Maxima* es

`cspline(datos)` pero como otras muchas interesantes capacidades del programa está fuera de los objetivos de este curso.

Bibliografía

[1] Dodier, Robert, 2005. Minimal *Maxima*. Disponible en http://maxima.sourceforge.net/docs/tutorial/en/minimal-maxima.pdf.

[2] *Maxima*. Manual de ayuda. Incluido dentro del programa, es una herramienta muy útil para resolver cualquier duda que tengamos sobre el funcionamiento del programa.

[3] Manual de Referencia de *Maxima*. Disponible en http://maxima.sourceforge.net/docs/manual/es/maxima.pdf.