Manual Introductorio a Maple 8.0
Álgebra Lineal Avanzada y Optimización

Departamento de Estadística y Matemática
Facultad de Ciencias Económicas
Universidad Nacional de Córdoba

Manual Introductorio a Maple 8.0
Álgebra Lineal Avanzada y Optimización

Jorge Mauricio Oviedo [1]
Emanuel López [2]
Diego Busignani [3]

Resumen: El presente trabajo tiene por objetivo acercar al usuario principiante una guía suficiente a la hora de dominar el entorno de uso y programación en Maple 8. La exposición de las herramientas se hace con suma claridad y sencillez mostrando comando de "input-output", con atractivos e ilustrativos ejemplos. Nociones de Operatoria Numérica, Algebraica, Diferencial y Optimización son expuestas a los lectores.

[1] joviedo@eco.unc.edu.ar
[2] lemanuel@eco.unc.edu.ar
[3] djj@fibertel.com.ar

... las matemáticas se parecen mucho a la poesía.... lo que convierte a un buen poema en un excelente poema es una gran cantidad de pensamiento expresado en pocas palabras. En este sentido expresiones como:

$$e^{i\pi} - 1 = 0 \qquad\qquad \int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}$$

son poemas.

Lipman Bers

CONTENIDO

INTRODUCCIÓN: **6**

1.- OPERACIONES NUMÉRICAS **7**

Operaciones con decimales 8
Sumatorios y Productorios Finitos e Infinitos 9
Números Complejos 11

2.-OPERACIONES ALGEBRAICAS **11**

Expresiones 11
Simplificar *12*
Asignaciones *12*
Funciones 14
Ecuaciones y Sistemas de Ecuaciones 16
Ecuaciones *16*
Sistemas de Ecuaciones *18*
Inecuaciones *19*

3.-CÁLCULO DIFERENCIAL E INTEGRAL **20**

Diferenciación e Integración 21
Límites 22
Valores Finitos e Infinitos *22*
Funciones Empalmadas 23
Expansión en Series 25

4.-GRÁFICAS **27**

Gráficas Bidimensionales 28
Ejemplo de un 2D Plot *28*
Gráficas de Funciones Implícitas *29*
El Paquete de Gráficas *29*
Gráficas Tri-dimensionales 30
Animaciones 32
Gráficas de Inecuaciones 33

5.-ÁLGEBRA LINEAL **34**

Operaciones Básicas con Matrices 34
Extracción de componentes y elementos de una Matriz *36*
Reducción por filas *36*
Subespacios Fila y Columna de una Matriz *37*
Funciones Matriciales 38
Valores y Vectores Propios 38
Transformaciones Lineales, Valores y Vectores Propios 40

6.-FORMAS CUADRÁTICAS LIBRES Y RESTRINGIDAS **41**

Positivas definidas 41
Animación *42*
Negativas Definidas 43
Animación *44*
Indefinidas 45
Animación *45*
Semi-Definidas Positivas 46
Animación *47*
Semi-Definidas Negativas 47
Animación *48*

7.-ANÁLISIS EN VARIAS VARIABLES **48**

Derivadas Parciales Direccionales - Gradientes - Hessianos - Jacobianos 49
Derivadas univaridas *49*
Derivadas Parciales *49*

Gradientes *49*
Derivadas Direccionales *50*
Curvas de Nivel *50*
Matriz Hessiana *52*
Matriz Jacobiana *53*
DERIVADAS DE ORDEN N - POLINOMIOS DE TAYLOR (INTROD. A LA PROGRAMACIÓN EN MAPLE) 53
Procedimientos para Crear Derivadas de orden n *53*
Comando para calcular un polinomio de Taylor para un función cualquiera *55*
Polinomio de Taylor en varias variables *56*
Aproximaciones - Animaciones *57*

8.-OPTIMIZACIÓN SIMBÓLICA 58

OPTIMIZACIÓN LIBRE 61
Caso especial nº 1 ..... *62*
Caso especial nº 2: La silla de Mono *64*
OPTIMIZACIÓN. RESTRICCIONES DE IGUALDAD 65
OPTIMIZACIÓN. RESTRICCIONES DE DESIGUALDAD 67

REFERENCIAS: 71

Introducción:

La necesidad de efectuar cómputos algebraicos, de obtener precisión casi infinita, de manejar o tratar con expresiones exactas suelen ser una tarea ardua y fastidiosa para quienes deban efectuarla. Asimismo la posibilidad de trazar gráficas de funciones en dos y tres dimensiones, de poder observar su imagen desde distintos puntos de vista suele ser más que un desafío;; igualmente que el tratar con animaciones y otras variantes gráficas. Desde hace no muchos años los sistemas algebraicos de computación surgieron para alivianar esta tarea teniendo una gran aceptación para los investigadores teóricos como así también en los medios de educación. Dentro de tales sistemas se encuentra Maple.

Maple es un programa desarrollado desde 1980 por el grupo de Cálculo Simbólico de la Universidad de Waterllo (Ontario, Canadá). Su nombre viene de las palabras en inglés "Mathematical Pleasure". Maple es capaz de desarrollar una amplia gama de problemas, mediante el uso de comandos predeterminados. También posee un lenguaje de programación para que el usuario pueda desarrollar sus propias funciones y programas.

El objetivo del presente trabajo es alcanzar al usuario de éstas operatorias matemáticas un manual lo suficientemente claro como para impulsar y motivar la adaptación del principiante en el entorno de Maple.

Para ello, se presentan en 7 secciones tópicos desde un nivel introductoria hasta cubrir aspectos avanzados de álgebra lineal, Cálculo Diferencial e Integral y Optimización. Cada sección es presentada con suma nitidez explicativa adecuando cada idea con ejemplos claros y concretos que, mas allá de ilustrar el contenido de lo tratado, atraen la atención y curiosidad del lector interesado en dominar el entorno de este programa.

El trabajo se estructura con las siguientes siete secciones: Cálculos Numéricos, Cómputos Algebraicos, Cálculo Diferencial e Integral, Álgebra Lineal, Formas Cuadráticas, Análisis en varias Variables y Optimización.

1.- Operaciones Numéricas

Antes de comenzar a trabajar en el entorno Maple, se recomienda tener la precaución de borrar la memoria del núcleo de cómputos del programa, con el siguiente comando:

```
> restart;
```

con esto, todos los nombres de variables y procedimientos serán "olvidados".

Obsérvese que cada comando introducido en Maple para poder ser ejecutado debe ser seguido por el símbolo "punto y coma" (;), luego de ello si podemos ejecutarlo presionando la tecla "enter".

Cómputos con Enteros

Maple permite trabajar numéricamente con el grado de complejidad y exactitud que se desee en cada problema particular. Así, las operaciones con números enteros y decimales más extensas, no significan ningún esfuerzo para "Maple". Esto constituye la principal ventaja de un sistema algebraico de computación ya que los programas numéricos solo pueden mostrar un cantidad limitada de números y decimales (en general no mas de 16 o 32 dependiendo de la máquina). En Maple pueden mostrarse cientos de miles de dígitos.

Para calcular las operaciones aritméticas básicas, simplemente las expresamos en una "línea de comando" de esta forma:

```
> 32*12^13;
                    3423782572130304
```

El circunflejo (^) representa la operación de potencia, el mismo puede escribirse con el código "Ascii" con las teclas “alt + 94”.

No hay ninguna dificultad en calcular números muy grandes, como por ejemplo el factorial de 200:

```
> 200!;
788657867364790503552363213932185062295135977687173263294742533244359449\
    963403342920304284011984623904177212138919638830257642790242637105061\
    926624952829931113462857270763317237396988943922445621451664240254033\
    291864131227428294853277524242407573903240321257405579568660226031904\
    170324062351700858796178922222789623703897374720000000000000000000000\
    00000000000000000000000000000
```

El numero Pi puede visualizarse ahora como una cantidad de 500 dígitos por ej. sin tipos de problemas. Ahí si que notamos el poder de precisión casi infinito de Maple.

```
> evalf[500]( Pi );
```

3.14159265358979323846264338327950288419716939937510582097494459230781640
06286208998628034825342117067982148086513282306647093844609550582231
72535940812848111745028410270193852110555964462294895493038196442881
09756659334461284756482337867831652712019091456485669234603486104543
26648213393607260249141273724587006606315588174881520920962829254091
71536436789259036001133053054882046652138414695194151160943305727036
57595919530921861173819326117931051185480744623799627495673518857527
248912279381830119491

A continuación se presentan algunos ejemplos de comandos de Maple para trabajar con números enteros.

El comando "**ifactor**" permite descomponer un entero en factores primos, por ejemplo:

```
> ifactor(788);
```

$$(2)^2\ (197)$$

La operación inversa se realiza con el comando "**expand**":

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

$$788$$

El símbolo de porcentaje (**%**) se emplea en lugar de rescribir el último resultado brindado por Maple.

Operaciones con decimales

Al trabajar con números racionales o irracionales, Maple, de forma predeterminada los expresa sin perder exactitud, no recurriendo a expresiones decimales a menos que se lo solicite. Por ejemplo: (el comando "**sqrt(n)**" brinda la raíz cuadrada de la expresión n).

```
> (2^30/3^20)*sqrt(3);
```

$$\frac{1073741824\sqrt{3}}{3486784401}$$

Si precisamos una aproximación decimal recurrimos al comando "**evalf**" que requiere dos argumentos, el primero es la expresión a aproximar y el segundo indica la precisión deseada, expresada como el número de decimales. Por ejemplo:

```
> evalf(%,15);
```

0.533378373737793

Alternativamente agregando en cualquier número de la expresión el sufijo ".0" Maple brinda una aproximación decimal predeterminada:

```
> (2^30/3^20)*sqrt(3.0);
```

0.5333783739

Sumatorios y Productorios Finitos e Infinitos

El comando "**sum**", nos permite resolver la sumatoria simbólica de una expresión que depende de un parámetro que varía en forma sucesiva (al que podemos llamar índice). El mismo posee dos argumentos. El primero es la expresión a sumar y el segundo indica el límite inferior y superior para el índice.

Ejemplo: Calcular

$$\sum_{i=1}^{10} \frac{1+i}{1+i^4} \ .$$

```
> sum( (1+i)/(1+i^4), i=1..10 );
```

$$\frac{5150805672759473291 3722}{4062664893881920008 8497}$$

Notar la forma en que debe indicarse el rango de variación. Si la expresión a sumar es "k(j)" (es decir “k” es una función de “j”), cuyo índice es "j" que varía desde "m" a "n" el comando debe expresarse:

```
> sum(k(j),j=m..n);
```

Los límites no deben ser necesariamente finitos. Para expresar al infinito se utiliza el vocablo ingles "infinity".

Ejemplo $\sum_{k=1}^{\infty} \frac{1}{k^2}$

```
> sum( 1/k^2, k=1..infinity );
```

$$\frac{\pi^2}{6}$$

En el caso del productorio el comando a emplear es "**product**".

Ejemplo $\prod_{i=0}^{10} \frac{i^2+3\,i-11}{i+3}$

```
> product( ((i^2+3*i-11)/(i+3)), i=0..10 );
```

$$\frac{-7781706512657}{40435200}$$

Si queremos una aproximación decimal de esta última expresión del ejemplo, usamos el conocido comando "**evalf**" donde los argumentos están escritos de una forma alternativa y equivalente:

```
> evalf[50](%);
```

-192448.81965854008388730610952833175055397277619500

Números Complejos

Trabajar con números complejos es igualmente sencillo en Maple, la unidad imaginaria debe indicarse con la letra "I" (i mayúscula). Por ejemplo:

```
> (3+5*I)/(7+4*I);
```

$$\frac{41}{65}+\frac{23}{65}I$$

2.-Operaciones Algebraicas

```
> restart;
```

Expresiones

Factorear

Maple brinda toda la potencia de un procesador simbólico, permitiendo trabajar con expresiones algebraicas de cualquier complejidad. Por ejemplo a continuación se introduce una potencia 15 de un binomio:

```
> expr := (x+y)^15;
```

$$expr := (x+y)^{15}$$

Lo que se ha hecho es definir el binomio con el nombre "**expr**"; para ello se utilizan los símbolos "**: =**". Ahora Maple, cuando aludamos a "**expr**", reconocerá que hacemos referencia a la expresión así llamada. El comando "**expand**" permitirá que se distribuyan los productos sobre las sumas, y que se "expandan" diferentes expresiones

que introduzcamos. El mismo requiere de un argumento, y es la expresión sobre la que se quiere operar. Por ejemplo, aplicado sobre "**expr**":

```
> expand(expr);
```

$$x^{15} + 15\,y\,x^{14} + 105\,y^2\,x^{13} + 455\,y^3\,x^{12} + 1365\,y^4\,x^{11} + 3003\,y^5\,x^{10} + 5005\,y^6\,x^9 + 6435\,y^7\,x^8 + 6435\,y^8\,x^7 + 5005\,y^9\,x^6 + 3003\,y^{10}\,x^5 + 1365\,y^{11}\,x^4 + 455\,y^{12}\,x^3 + 105\,y^{13}\,x^2 + 15\,y^{14}\,x + y^{15}$$

La operación inversa permite factorizar cualquier expresión algebraica y el comando que debe utilizarse es "**factor**":

```
> factor(%);
```

$$(x + y)^{15}$$

Simplificar

El comando "**simplify**" permite que se apliquen reglas de simplificación sobre una determinada expresión. El comando "normal" permite la simplificación de expresiones que envuelven funciones racional, devolviendo su "forma normal factorizada", es decir un cociente en donde numerador y denominador son relativamente polinomios primos con coeficientes enteros. Veamos unos ejemplos:

```
> simplify( cos(x)^5 + sin(x)^4 + 2*cos(x)^2 - 2*sin(x)^2 - cos(2*x) );
```

$$\cos(x)^4\,(\cos(x) + 1)$$

```
> normal( (x^3-y^3)/(x^2+x-y-y^2) );
```

$$\frac{x^2 + y\,x + y^2}{x + 1 + y}$$

Asignaciones

En Maple podemos hacer diferentes asignaciones de nombres, operar con ellos, evaluar expresiones, y combinar los diferentes comandos en la forma que deseemos. A continuación un ejemplo:

Primero generamos dos expresiones algebraicas a las que denominamos "**expr1**" y "**expr2**", donde esta última es la "expansión" de la primera:

```
> expr1 := (41*x^2+x+1)^2*(2*x-1);
```

$$expr1 := (41\,x^2 + x + 1)^2\,(2\,x - 1)$$

```
> expr2 := expand(expr1);
```

$$expr2 := 3362\,x^5 - 1517\,x^4 + 84\,x^3 - 79\,x^2 - 1$$

El comando "**eval**" nos permite evaluar una determinada expresión en un valor deseado de alguno de sus parámetros o variables. El mismo posee dos argumentos. El primero es la expresión a evaluar, el segundo indica a que valor debe igualarse el parámetro o variable deseado. Por ejemplo:

```
> eval(expr2, x=1);
```

$$1849$$

Definimos la expresión "**top**" como igual a "**expr2**":

```
> top := expr2;
```

$$top := 3362\,x^5 - 1517\,x^4 + 84\,x^3 - 79\,x^2 - 1$$

Definimos la expresión "**bottom**" como la "expansión" de otra expresión:

```
> bottom := expand((3*x+5)*(2*x-1));
```

$$bottom := 6\,x^2 + 7\,x - 5$$

Operamos con las expresiones definidas, pidiendo que se simplifique el resultado y denominando a este último "**answer**":

```
> answer := simplify( top/bottom );
```

$$answer := \frac{1681\,x^4 + 82\,x^3 + 83\,x^2 + 2\,x + 1}{3\,x + 5}$$

Funciones

Para definir funciones en Maple debemos nombrar a la función del mismo modo que nombramos a cualquier expresión, pero en este caso debemos especificar cual es la variable de la función, seguida por una "flecha" (compuesta por el signo "menos" y el signo "mayor que" "**->**"), y luego especificamos la estructura particular de la función que deseamos. Un ejemplo lo dejará más claro:

```
> f := x -> x^2+1/2 ;
```

$$f := x \to x^2 + \frac{1}{2}$$

Ya podemos trabajar con "*f*" como si fuera una función. Podemos, por ejemplo, evaluarla en el punto *x*=2 o en el punto *x*=a+b de la siguiente manera:

```
> f(2);
```

$$\frac{9}{2}$$

```
> f(a+b);
```

$$(a + b)^2 + \frac{1}{2}$$

Podemos alternativamente definir una función con el comando "**unapply**". El mismo requiere de dos "insumos"; el primero es la expresión funcional, mientras que el segundo es (son) la(s) variable(s) de la expresión. Por ejemplo:

```
> g := unapply(x^2 + 1/2, x);
```

$$g := x \to x^2 + \frac{1}{2}$$

Notar que el comando debe estar precedido por el "nombre" que le daremos a la función.

Se permite también que generemos una determinada expresión algebraica, y luego la convirtamos en función vía el comando "**unapply**". Por ejemplo, primero definimos "**p**", y a continuación lo convertimos en la función "**f**":

```
> p := x^2 + sin(x) + 1;
```

$$p := x^2 + \sin(x) + 1$$

```
> f := unapply(p,x);
```

$$f := x \to x^2 + \sin(x) + 1$$

Notar que hemos utilizado un nombre de función ya existente. Maple considera que el nombre corresponde a la última función que denominamos con el.

Consideremos la siguiente expresión con dos parámetros:

```
> q := x^2 + y^3 + 1;
```

$$q := x^2 + y^3 + 1$$

Podemos definirlo solo como una función de x:

```
> f := unapply(q,x);
```

$$f := x \to x^2 + y^3 + 1$$

```
> f(2);
```

$$5 + y^3$$

O como una función de x y y:

```
> g := unapply(q,x,y);
```

$$g := (x, y) \to x^2 + y^3 + 1$$

---

y evaluarla en el punto (2,3) por ejemplo:

```
> g(2,3);
```

$$32$$

De la misma manera es posible definir funciones más complejas, con más variables, con parámetros numéricos o algebraicos y operar con ellas en la forma deseada.

Ecuaciones y Sistemas de Ecuaciones

Ecuaciones

El comando "**solve**" posibilita la resolución de ecuaciones o sistemas de ecuaciones. Los argumentos del comando son en primer lugar un listado de la o las ecuaciones que deseamos resolver, en segundo lugar un listado de las variables para las cuales debe hallarse solución. Por ejemplo podemos definir la siguiente expresión:

```
> eqn := x^3-1/2*a*x^2+13/3*x^2 = 13/6*a*x+10/3*x-5/3*a;
```

$$eqn := x^3 - \frac{1}{2} a\, x^2 + \frac{13}{3} x^2 = \frac{13}{6} a\, x + \frac{10}{3} x - \frac{5}{3} a$$

Para resolver la ecuación en la variable x por ejemplo:

```
> solve(eqn, {x});
```

$$\{x = \frac{2}{3}\}, \{x = -5\}, \{x = \frac{a}{2}\}$$

Envolviendo las variables entre llaves en el comando, conseguimos que el resultado se exprese en la forma expuesta. Si omitimos las llaves obtendremos el mismo resultado pero como un listado de las soluciones separado por "comas" de la siguiente forma:

```
> solve(eqn, x);
```

$$\frac{2}{3}, -5, \frac{a}{2}$$

Para corroborar alguno de los resultados obtenidos se puede recurrir al comando "**eval**", procediendo de la siguiente manera:

```
> eval( eqn , x=1/2*a );
```

$$\frac{13\,a^2}{12} = \frac{13\,a^2}{12}$$

La identidad obtenida confirma que la solución es correcta.

Podemos resolver un polinomio de segundo grado en x, observando que las soluciones se corresponden con las conocidas "fórmulas" de cálculo:

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

$$\frac{-b+\sqrt{b^2-4\,a\,c}}{2\,a}, \frac{-b-\sqrt{b^2-4\,a\,c}}{2\,a}$$

Se puede introducir también polinomios de orden tres y cuatro, para los cuales Maple calcula las fórmulas exactas de las raíces. Para un polinomio en x de orden tres (no se presentan los resultados por razones de espacio):

```
> solve(a*x^3+b*x^2+c*x+d=0,x);
```

En el caso del polinomio de orden cuatro, dada la complejidad del resultado, Maple por defecto no lo presenta. Pero si se quiere se puede "desenmascarar" este resultado recurriendo al siguiente comando:

```
> solve(a*x^4+b*x^3+c*x^2+d*x+e=0,x);
```

$$\text{RootOf}(a\_Z^4 + b\_Z^3 + c\_Z^2 + d\_Z + e)$$

```
> _EnvExplicit := true:

> solve(a*x^4+b*x^3+c*x^2+d*x+e=0,x);
```

Sistemas de Ecuaciones

A continuación se presenta un ejemplo de cómo resolver sistemas de ecuaciones. En primer lugar definimos una serie de expresiones, que luego serán las ecuaciones a resolver:

```
> eqn1 := a+2*b+3*c+4*d+5*e=41;
```

$$eqn1 := a + 2\,b + 3\,c + 4\,d + 5\,e = 41$$

```
> eqn2 := 5*a+5*b+4*c+3*d+2*e=20;
```

$$eqn2 := 5\,a + 5\,b + 4\,c + 3\,d + 2\,e = 20$$

```
> eqn3 := 3*b+4*c-8*d+2*e=125;
```

$$eqn3 := 3\,b + 4\,c - 8\,d + 2\,e = 125$$

```
> eqn4 := a+b+c+d+e=9;
```

$$eqn4 := a + b + c + d + e = 9$$

Utilizamos el comando "**solve**" para resolver el sistema anterior. Debe notarse que es necesario encerrar entre llaves el conjunto de ecuaciones por un lado, y el conjunto de variables por otro:

```
> solve( {eqn1, eqn2, eqn3, eqn4}, {a, b, c, d} );
```

$$\{c = \frac{483}{13} - \frac{31\,e}{13},\ b = -\frac{313}{13} + \frac{22\,e}{13},\ d = -\frac{79}{13} - \frac{4\,e}{13},\ a = 2\}$$

Corroboramos el resultado obtenido de la siguiente manera, donde se observa que se satisfacen las cuatro ecuaciones:

```
> eval({eqn1,eqn2,eqn3,eqn4},%);
```

$$\{41 = 41, 20 = 20, 125 = 125, 9 = 9\}$$

Mas ejemplos

Maple posee un potente procesador que permite resolver prácticamente cualquier tipo de ecuación, incluyendo aquellas con funciones trigonométricas, valores absolutos, e inecuaciones, de lo cual presentamos un ejemplo a continuación. Introducimos también el comando "dos puntos (**:**)" sustituyendo al "punto y coma", esto permite postergar la evaluación de un determinado comando, reservándose Maple de hacerlo hasta que se tope con el requerido "punto y coma".

```
> solve( arccos(x) - arctan(x)= 0, {x} );
```

$$\left\{x = \frac{\sqrt{-2+2\sqrt{5}}}{(-1+\sqrt{5})\sqrt{\frac{\sqrt{5}+1}{-1+\sqrt{5}}}}\right\}$$

```
> solve( abs( (z+abs(z+2))^2-1 )^2 = 9, {z});
```

$$\{z = 0\}, \{z \le -2\}$$

```
> eval( abs( (z+abs(z+2))^2-1 )^2 = 9, z=-3 );
```

$$9 = 9$$

Inecuaciones

```
> ineq := x+y+4/(x+y) < 10:
> solve( ineq, {x} );
```

$$\{x < -y\}, \{5 - \sqrt{21} - y < x, x < 5 + \sqrt{21} - y\}$$

Un comando interesante y de mucha utilidad en diversas aplicaciones es el comando *"**is**"*. El mismo permite corroborar la veracidad o falsedad de un determinado postulado. Veamos un ejemplo para aclararlo, donde definimos la expresión "**expr**"

```
> expr:=2*sqrt(-1-I)*sqrt(-1+I);
```

$$expr := 2\sqrt{-1-I}\sqrt{-1+I}$$

Por ejemplo si queremos determinar si la expresión es distinta de cero, debemos escribir:

```
> is(expr<>0);
```

true

El resultado "true" indica que es verdadero que la expresión es mayor o menor que cero, y por lo tanto distinta del mismo.

Es posible también realizar supuestos acerca de los valores de variables y determinar la veracidad o falsedad de determinadas expresiones. Para ello recurrimos al comando *"**assume**"* y como argumentos colocamos un listado separando por comas los distintos supuestos que realizamos, luego con el comando *"**is**"* podemos evaluar el valor de verdad de la expresión deseada; por ejemplo:

```
> assume(x>5, y<-10);
> is(x*y<50);
```

true

3.-Cálculo Diferencial e Integral

```
> restart;
```

Diferenciación e Integración

Los problemas de cálculo diferencial e integral pueden tratarse sin problemas en el entorno Maple.

Para calcular la derivada de una función, lo primero que debemos hacer es definir la misma y luego el comando "**diff**" permitirá obtener el resultado deseado (también puede definirse la función dentro del comando). Definamos una función cualquiera para observar más claramente el uso de este comando.

```
> f := x -> x*sin(a*x) + b*x^2;
```

$$f := x \to x \sin(a\,x) + b\,x^2$$

```
> f_prime := diff( f(x), x);
```

$$f\_prime := \sin(a\,x) + x\cos(a\,x)\,a + 2\,b\,x$$

De forma muy sencilla podemos también calcular una integral definida o indefinida de una función. Por ejemplo calculemos la integral indefinida de la derivada de f(x) definida anteriormente (la integral de "**f_prime**"). Los argumentos son nuevamente dos, el primero la función a integrar y el segundo la variable de integración:

```
> int(f_prime, x);
```

$$-\frac{\cos(a\,x)}{a} + \frac{\cos(a\,x) + a\,x\sin(a\,x)}{a} + b\,x^2$$

```
> simplify(%);
```

$$x\,(\sin(a\,x) + b\,x)$$

Si queremos calcular una integral definida simplemente escribimos el límite inferior y superior para la variable respecto de la que se integra. En forma general si integramos respecto de x que varia entre los límites m y n, debemos denotar "x=m..n". Por ejemplo, la integral definida de "**f_prime**" para x entre 1 y 2:

```
> int( f_prime, x=1..2 );
```

$$2\sin(2a)+3b-\sin(a)$$

Límites

Valores Finitos e Infinitos

El cálculo de límites de cualquier expresión en Maple se resuelve sencillamente con el comando *"*`limit`*"*. El mismo posee dos argumentos, siendo el primero la expresión a la que se le calcula el límite, y el segundo la variable igualada al valor hacia el cual se la hace tender. Por ejemplo definamos una expresión a la que llamamos "`expr`" y calculemos el límite de la misma para cuando su variable *x* tiende a infinito (el cual se expresa en Maple con el código **"**`infinity`*"*):

```
> expr:=(2*x+3)/(7*x+5);
```

$$expr := \frac{2x+3}{7x+5}$$

```
> limit(expr,x=infinity);
```

$$\frac{2}{7}$$

Los límites calculados pueden ser tanto finitos como infinitos. Así también podemos calcular límites por derecha o por izquierda añadiendo en el comando *"*`limit`*"* un tercer argumento que indique "`left`" si buscamos el límite por izquierda, o "`right`" si buscamos el límite por derecha. Veamos un ejemplo:

```
> limit(tan(x+Pi/2),x=0,left);
```

$$\infty$$

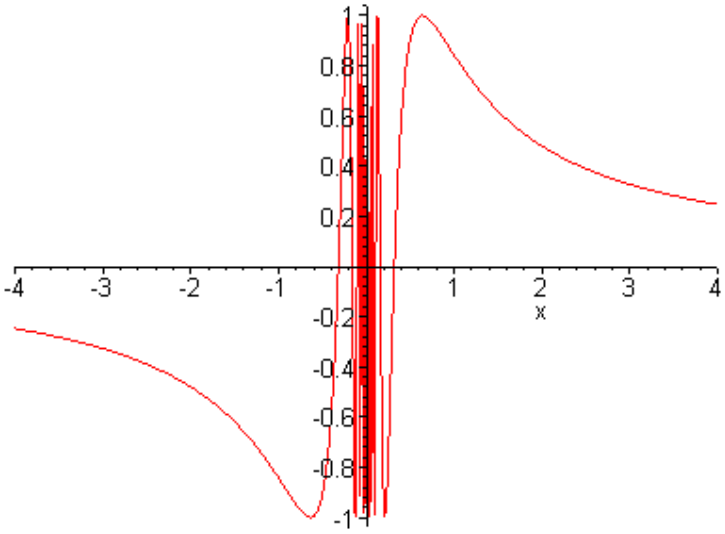
```
> limit(tan(x+Pi/2),x=0,right);
```

$$-\infty$$

En particular para la función propuesta, al diferir los límites por izquierda y derecha, se puede afirmar que no existe el límite para cuando *x* es igual a cero.

Maple también reconoce la existencia de límites indefinidos. Observemos el siguiente gráfico (los comandos para generar gráficos se verán en otra sección):

```
> plot(sin(1/x),x=-4..4);
```



Calculemos ahora el límite de la función para cuando *x* tiende a cero:

```
> limit(sin(1/x),x=0);
```

-1 .. 1

El resultado otorgado por Maple establece que el límite es indefinido y que puede variar entre –1 y 1.

Funciones Empalmadas

El comando "**piecewise**" permite construir funciones empalmadas. Para ello se procede a definir la función de la manera anteriormente explicada, pero ahora debe anteponerse a la función el comando "**piecewise**", cuyos argumentos deben separarse por comas y constan de colocar en primer lugar el rango de la variable para la cual será válida la función que se expresa en el segundo argumento; el tercer lugar se coloca otro rango de variación, en cuarto la función que prevalecerá para el mismo y así sucesivamente. El último argumento se corresponde con la función que será valida para los intervalos no considerados con anterioridad (no siendo necesario definirlos para este último caso). Un ejemplo arrojará luz sobre este comando. Definamos una función "p" que se compone en partes por otras tres funciones; procedemos así:

Ejemplo

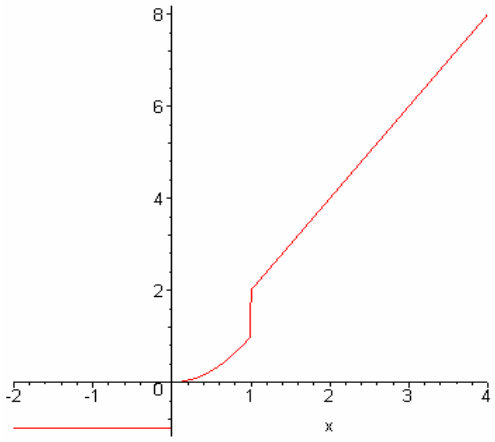
```
> p := x -> piecewise( x<0, -1, x>1, 2*x, x^2 );
```

$$p := x \to \text{piecewise}(x < 0, -1, 1 < x, 2\,x, x^2)$$

```
> p(x);
```

$$\begin{cases} -1 & x < 0 \\ 2\,x & 1 < x \\ x^2 & otherwise \end{cases}$$

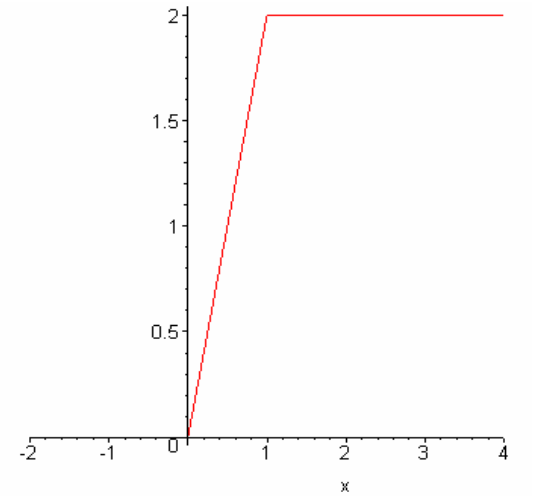
```
> plot(p(x),x=-2..4);
```




```
> p_prime := diff( p(x), x );
```

$$p\_prime := \begin{cases} 0 & x < 0 \\ undefined & x = 0 \\ 2\,x & x < 1 \\ undefined & x = 1 \\ 2 & 1 < x \end{cases}$$

```
> plot(p_prime,x=-2..4);
```



Se observa como Maple reconoce inteligentemente aquellos puntos de la función en los cuales la derivada está indeterminada porque la función presenta una discontinuidad.

Expansión en Series

Para obtener la expansión en serie de una determinada expresión o función Maple nos brinda el comando *"**series**"*. El mismo requiere de dos "input", el primero, la función o expresión a expandir; el segundo, el punto alrededor del cual se desea realizar la expansión. En el ejemplo presente a continuación, primero se define la expresión "expr", y luego se obtiene la expansión de la misma alrededor del punto $x=0$:

```
> expr:=sin(4*x)*cos(x):
> approx1:=series(expr,x=0);
```

$$approx1 := 4x - \frac{38}{3}x^3 + \frac{421}{30}x^5 + O(x^6)$$

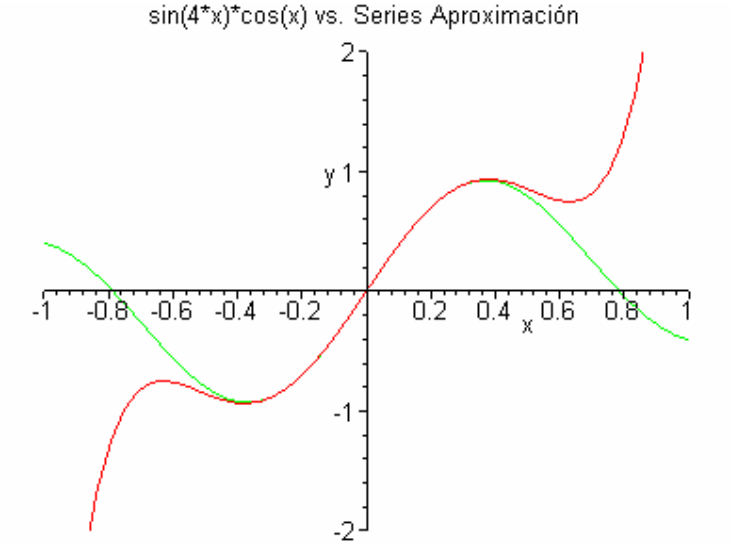
Por defecto Maple ofrece un grado de expansión de la serie igual a seis, y el último término representa el "resto" de la expansión. Para poder trabajar con la expresión necesitamos eliminar este último término y convertirla en un polinomio. Para ello utilizamos el comando *"**convert**"* seguido de dos argumentos separados por comas, donde el primero indica la expresión a convertir y el segundo (en este caso) el comando *"**polynom**"*, indicando que precisamos un polinomio. A su vez denominamos a este "poly1":

```
> poly1:=convert(approx1,polynom);
```

$$poly1 := 4x - \frac{38}{3}x^3 + \frac{421}{30}x^5$$

A continuación se presenta un gráfico de la función original y de su aproximación:

```
> plot({expr,poly1},x=-1..1,y=-2..2,title=cat(convert(expr
string),"vs. Series Aproximación"));
```



El orden predeterminado de la expansión puede ser especificado a través del comando *"*`order`*"*, cuyo único argumento es el orden deseado. Por ejemplo:

```
> Order:=12;
```

$$Order := 12$$

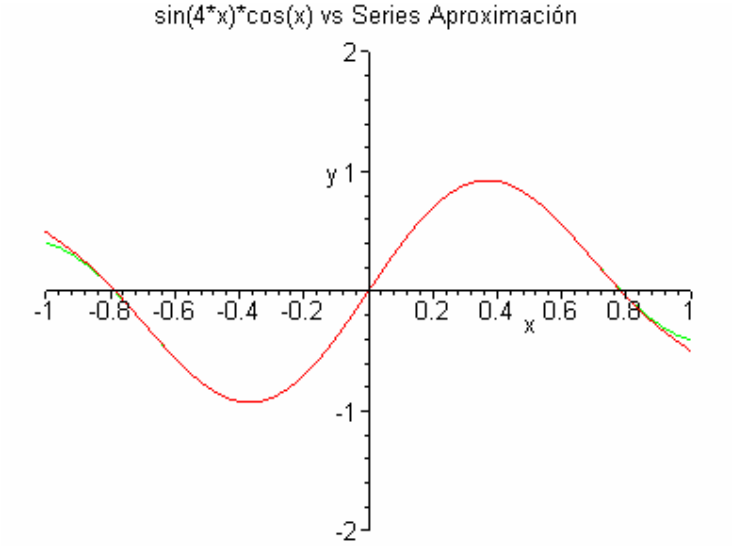
```
> approx2:=series(expr,x=0);
```

$$approx2 := 4x - \frac{38}{3}x^3 + \frac{421}{30}x^5 - \frac{10039}{1260}x^7 + \frac{246601}{90720}x^9 - \frac{6125659}{9979200}x^{11} + O(x^{12})$$

Alternativa y equivalentemente podemos especificar el orden de expansión como un tercer argumento en el comando *"**series**"*. Para ser estrictos, debe aclararse que el "orden" seleccionado no es el grado con el que la serie se presentará en el resultado.

Continuando con el último ejemplo, podemos convertirlo en un polinomio y representarlo gráficamente junto con la función original:

```
> poly2:=convert(approx2,polynom):
> plot({expr,poly2},x=-1..1,y=-2..2,title=cat(convert(expr,
string),"vs Series Aproximación"));
```



Vemos como en este caso la aproximación se "pega" a la función con un grado mayor de precisión.

4.-Gráficas

```
> restart;
```

Antes de hablar específicamente sobre las gráficas, vale la pena explicar como trabaja Maple. Cuando iniciamos el programa, éste carga sólo el núcleo o "kernel", la base del sistema, que contiene comandos primitivos y fundamentales. Lo restante está escrito en lenguaje Maple, y reside en la librería Maple, esta se subdivide en la parte principal y los paquetes. Los paquetes contienen una serie de comandos de una determinada área.

Con esta explicación ahora procedemos a activar los paquetes para gráficas "**plots**" y "**plottools**" (permite generar y manipular diferentes figuras geométricas), esto se realiza con el comando "**with (paquete)**" .

```
> with(plots):

Warning, the name changecoords has been redefined

> with(plottools):

Warning, the name arrow has been redefined
```

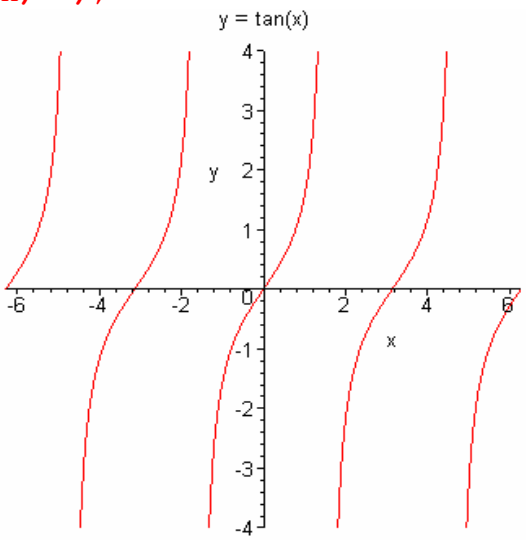
Si se pusiese poner punto y coma al final de cada sentencia Maple mostrará el nombre de todas las funciones adicionales que carga.

Gráficas Bidimensionales

Ejemplo de un 2D Plot

Para graficar usamos el comando "**plot**", este puede estar seguido de varios argumentos, comenzaremos por lo más simple y necesario, el primer argumento debe ser la función a graficar y el segundo los límites del rango de "x" que se deseen mostrar. Por ejemplo:

```
> plot( tan(x), x=-2*Pi..2*Pi, y=-4..4, discont=true,
title="y = tan(x)" );
```



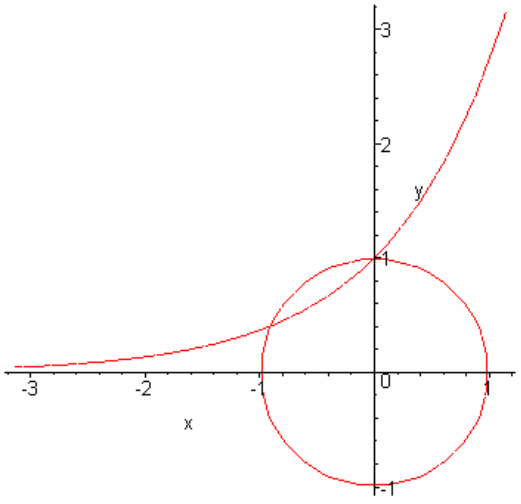
En este caso usamos además los comandos "**discont=true**" y "**title**", el primero sirve para que dibuje cerca de las asíntotas verticales (sin este argumento Maple dibuja líneas verticales en las discontinuidades), y el segundo para titular al Gráfico. Notar que ambos están seguidos por el signo igual y "**title=**" debe llevar entre comillas el título deseado para el gráfico.

Para más información sobre todos los argumentos que pueden incluirse en "**plot**" sugerimos usar el comando de ayuda "?" la sintaxis sería "**?plot[options]**"

Gráficas de Funciones Implícitas

El comando "**implicitplot**" nos permite dibujar funciones expresadas en forma implícita. La sintaxis es la siguiente "**implicitplot({expresión1, expresión2},opciones)**". Por ejemplo:

```
> implicitplot( { x^2+y^2=1, y=exp(x) }, x=-Pi..Pi, y=-
Pi..Pi, scaling=CONSTRAINED);
```

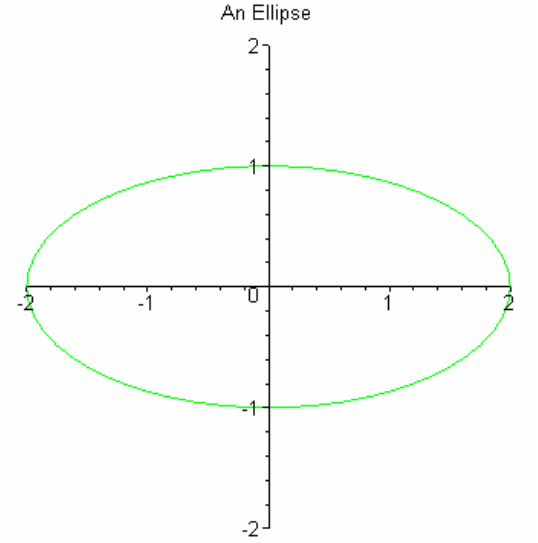


Para que la escala usada en el eje "x" sea la misma que la usada en el "y" usamos el comando "**scaling=CONSTRAINED**", esto hace que le dibujo no se vea distorsionado. Maple por defecto utiliza diferentes escalas para los ejes, por esto es necesario explicitarlo.

El Paquete de Gráficas

Ahora utilizaremos el comando del paquete "**plottools**" "**ellipse**", los parámetros a tener en cuenta son: el centro de la elipse, caracterizado por un punto en el plano de la forma [x,y], el radio horizontal, el radio vertical y otras opciones. Por ejemplo:

```
> elli := ellipse( [0, 0], 2, 1, color=green ):
> display( elli, scaling=CONSTRAINED,  view=[-2..2,-2..2],
title="An Ellipse" );
```

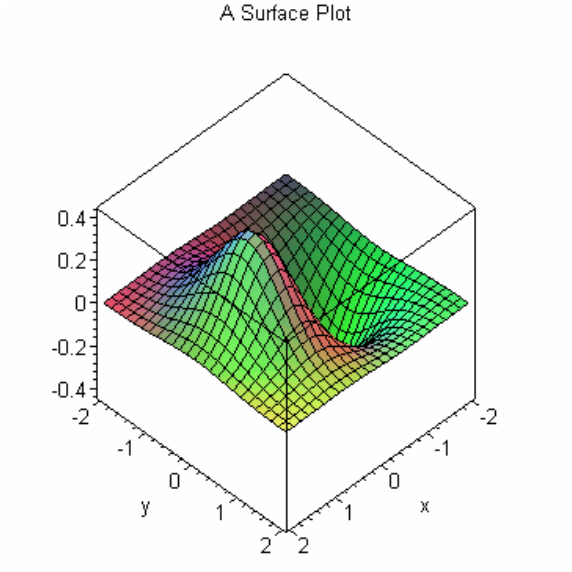


"**display**" nos permite visualizar una lista gráficas, la sintaxis es "**display(L,options)**". El comando "**view**" nos indica los rangos de variación de "x" e "y".

Gráficas Tri-dimensionales

"plot3d" es el comando usado para representar gráficos tridimensionales, para ello, se debe definir una expresión en función de 2 variables y los rangos de variación de las mismas. Veamos un ejemplo:

```
> plot3d( x*exp(-x^2-y^2), x=-2..2, y=-2..2, axes=BOXED,
lightmodel=light1,title="A Surface Plot" );
```

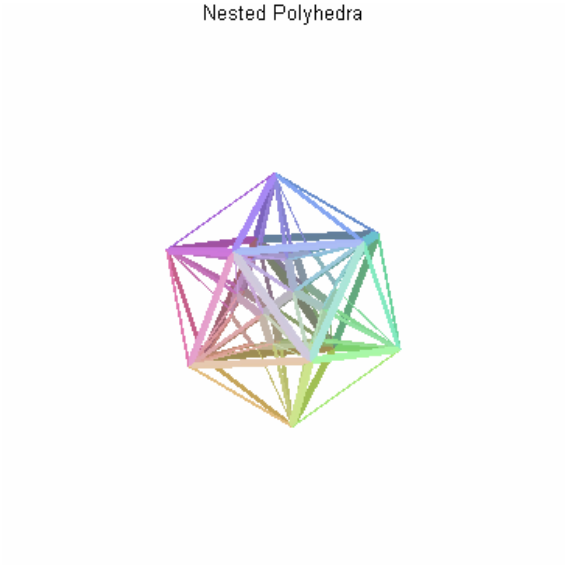


"**lightmodel=s**" nos permite elegir la forma de iluminar a la gráfica, los valores posibles de "s" son: 'none', 'light1', 'light2', 'light3', y 'light4'.

"**axes=BOXED**" indica a Maple que nos muestre los ejes en donde se representan las variables.

Ahora presentamos otro ejemplo de una gráfica en tres dimensiones en donde usamos el comando "**seq**", éste está diseñado para crear secuencias, la sintaxis que debe llevar es la siguiente:"**seq(expresión,i=a..b)**" donde "a" y "b" son los índices de variación.

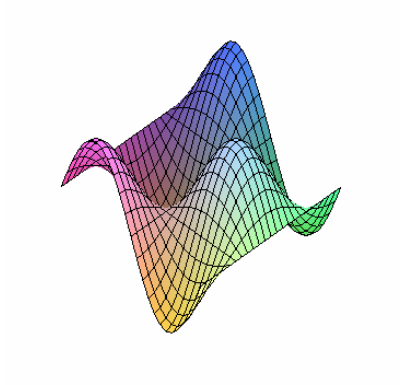
```
> p := display( seq( cutout(v, 4/5),
v=stellate(dodecahedron(), 3) ),
style=PATCH ):
> q := display( cutout(icosahedron([0, 0, 0], 2.2), 7/8) ):
> display( p, q, scaling=CONSTRAINED, title="Nested
Polyhedra" );
```



Animaciones

Maple realiza animaciones con gran facilidad. En las animaciones se representa una función que varía con el tiempo u otro parámetro. En las animaciones tridimensionales, 2 variables corresponden al espacio y la tercera al tiempo o parámetro. El comando a usar para animaciones en tres dimensiones es "animate3d" cuya sintaxis es la siguiente "**`animate3d(f,x,y,t)`**" donde "*f*" representa la expresión y "x","y","t" el rango de variación de "x","y","t" respectivamente. Por ejemplo:

```
> animate3d(cos(t*x)*sin(t*y),x=-Pi..Pi,y=-Pi..Pi,t=1..2);
```



Si "clickeamos" sobre la figura, ésa queda seleccionada y aparecen los siguientes botones en la barra de herramientas.

Los dos primeros valores cambian la orientación de la figura. Los dos siguientes son el detener y comenzar. Las funciones de los cinco siguientes son respectivamente: mover al cuadro siguiente, cambiar la dirección de la animación hacia atrás, cambiarla hacia adelante, disminuir y aumentar la velocidad de reproducción. Los dos últimos son para que la animación sea de un único ciclo o de ciclo continuo.

Gráficas de Inecuaciones

Los sistemas de inecuaciones de dos variables se representan por la función "inequal". La sintaxis es "**inequal(inecuaciones, rango x, rango y, opciones)**". Las opciones son las siguientes:

" "**optionsfeasible**": región factible, esto es que satisface todas las inecuaciones.

" "**optionsopen**": para representar una línea frontera abierta, que no pertenece al campo de la solución.

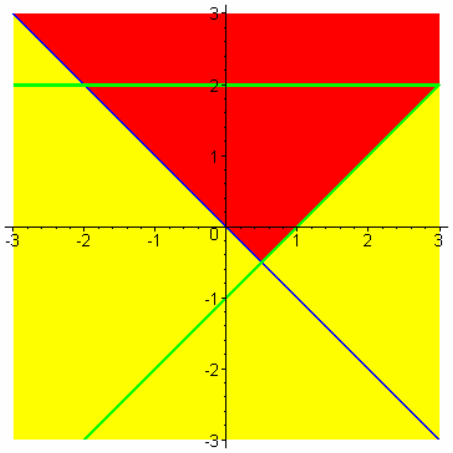
" "**optionsclosed**": para representar una línea frontera cerrada, que pertenece al campo de la solución.

" "**optionsexcluded**": región excluida que no cumple al menos una inecuación.

Ejemplo:

$0 < x + y$, $x - y \le 1$, $y = 2$

```
> inequal( { x+y > 0, x-y <= 1, y = 2 }, x=-3..3, y=-3..3,
optionsfeasible=(color=red), optionsopen=(color=blue,
thickness=2), optionsclosed=(color=green, thickness=3),
optionsexcluded=(color=yellow) );
```



5.-Álgebra Lineal

El siguiente comando carga internamente una serie de nuevos comandos que serás de gran utilidad a la hora de hacer cómputos simbólicos en Álgebra Lineal

```
> with(linalg):
```

A continuación se expone la forma de crear una matriz de 2 x 2. En esta ocasión se la crea como una lista de sus sucesivas filas, es decir una lista de listas:

```
> A:=[[1,5],[7,4]];
```

$$A := [[1, 5], [7, 4]]$$

Una matriz muy utilizada es la matriz identidad. El comando para generarla es el siguiente:

```
> array(identity, 1..2,1..2);
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Operaciones Básicas con Matrices

Las operaciones básicas de matrices debe efectuarse mediante el uso del comando "**evalm**". La suma y la multiplicación se muestran seguidamente:

```
> evalm(A-[[1,8],[1,1]]);
```

$$\begin{bmatrix} 0 & -3 \\ 6 & 3 \end{bmatrix}$$

```
> eval(multiply(A,[[2,3],[5,7]]));
```

$$\begin{bmatrix} 27 & 38 \\ 34 & 49 \end{bmatrix}$$

Calcular potencias matriciales también es posible en Maple:

```
> evalm(A^5);
```

$$\begin{bmatrix} 17396 & 19555 \\ 27377 & 29129 \end{bmatrix}$$

La inversa también es posible calcularla usando potencias -1 o directamente con el comando “inverse”:

```
> evalm(A^(-1));
```

$$\begin{bmatrix} \frac{-4}{31} & \frac{5}{31} \\ \frac{7}{31} & \frac{-1}{31} \end{bmatrix}$$

```
> inverse(A);
```

$$\begin{bmatrix} \frac{-4}{31} & \frac{5}{31} \\ \frac{7}{31} & \frac{-1}{31} \end{bmatrix}$$

La transposición es muy sencilla también mediante el comando intuitivo:

```
> transpose(A);
```

$$\begin{bmatrix} 1 & 7 \\ 5 & 4 \end{bmatrix}$$

El computo de determinantes se efectúa como sigue.

```
> B:=det([[1,5,a],[2,a,2],[7,a-2,a^2]]);
```

$$B := a^3 - 6\,a + 74 - 15\,a^2$$

Nótese hasta aquí como todos los comandos anteriores pueden ser utilizados con insumos algebraicos como se expone a continuación. El haber definido el determinante de una matriz con una variable permite luego, por ejemplo, resolver una ecuación. En este caso, el siguiente comando responde al interrogante de cual ha de ser el valor de a que anule el determinante de la matriz.

```
> evalf(solve(B=0,a),3);
```

$$15.0 + 0.\,I,\ -2.26 - 0.0173\,I,\ 2.18 + 0.0173\,I$$

Extracción de componentes y elementos de una Matriz

Con gran frecuencia es útil referiste a los componente de una matriz. A modo de ejemplo se muestra como extraer la primera fila de A y el elemento (1,2) de la misma:

```
> A[1];
> A[1,2];
```

$$[1, 5]$$

$$5$$

Reducción por filas

La reducción de Gauss-Jordan puede llevarse a cabo por medio de las instrucciones "**gaussjord**" como sigue.

```
> gaussjord([[1,5,y1],[7,4,y2]]);
```

$$\begin{bmatrix} 1 & 0 & -\frac{4\,y1}{31}+\frac{5\,y2}{31} \\ 0 & 1 & -\frac{y2}{31}+\frac{7\,y1}{31} \end{bmatrix}$$

```
> gaussjord(A);
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

La solución de sistemas lineales AX=b donde b es una vector columna genérico puede arrojar errores si se lo trata normalmente pues al ampliar la matriz a reducir el sistema utiliza la columna b para hacer ceros y unos.

```
> gaussjord([[1,5,y1],[2,10,y2]]);
```

$$\begin{bmatrix} 1 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

El inconveniente puede subsanarse agregando una cantidad suficientes de vectores columna linealmente independientes de modo de engañar al sistema. En el resultado debe omitirse la reducción de las columnas agregadas pues fueron intencionalmente creadas para que el sistema evite reducir el lado derecho de la ecuación matricial es decir el vector de términos independientes b.

```
> gaussjord([[1,5,1,y1],[2,10,0,y2]]);
```

$$\begin{bmatrix} 1 & 5 & 0 & \frac{y2}{2} \\ 0 & 0 & 1 & -\frac{y2}{2}+y1 \end{bmatrix}$$

Subespacios Fila y Columna de una Matriz

Bases de los subespacios filas se obtiene a partir de la siguiente sintaxis:

```
> rowspace(A);
```

$$\{[1, 0], [0, 1]\}$$

```
> colspace([[1,a],[2,a],[a-2,a^2]]);
```

$$\{[0, 1, -2], [1, 0, a + 2]\}$$

Funciones Matriciales

En las entradas de las matrices y vectores uno puede ingresar variables, expresiones y funciones matemáticas. Las mismas podrán ser utilizadas con todos los comandos citados anteriormente (operaciones, determinantes, extracciones etc.)

```
> F:=([[sin(x), x^2+x+3], [exp(x), cos(x^2)]] );
```

$$F := [[\sin(x), x^2 + x + 3], [\mathbf{e}^x, \cos(x^2)]]$$

Una de las ventajas de trabajar con funciones matriciales es que uno puede luego derivar sobre ella, integrar o aplicar cualquier otro operador sobre sus componentes. La evaluación en un punto determinado también es posible con el viejo comando "**eval**" y mostrar resultados numéricos con cierta precisión mediante el "**evalf**".

```
> diff( F, x);
```

$$[[\cos(x), 2\,x + 1], [\mathbf{e}^x, -2\,\sin(x^2)\,x]]$$

```
> evalf(eval(F,x=3),4);
```

$$[[0.1411, 15.], [20.09, -0.9111]]$$

Valores y Vectores Propios

Con los comandos que uno maneja hasta ahora podría naturalmente computar los autovalores de una matriz componiendo los comandos "solve", "det" y la matriz identidad.

```
> solve(det(A-lambda*array(identity, 1..2,1..2))=0,lambda);
```

$$\frac{5}{2}+\frac{\sqrt{149}}{2}, \frac{5}{2}-\frac{\sqrt{149}}{2}$$

Sin embargo la instrucción "**eigenvalues**" resume el procedimiento anterior de esta forma:

```
> eigenvalues(A);
> evalf(%,3);
```

$$\frac{5}{2}+\frac{\sqrt{149}}{2}, \frac{5}{2}-\frac{\sqrt{149}}{2}$$

8.60, -3.60

Los vectores propios se computan simbólicamente con el comando "**eigenvectors**". La salida de Maple devuelve un vector con tantas componentes como valores propios distintos existan. A su vez cada componente está integrada por tres elementos: primero el valor propio, segundo su multiplicidad y tercero un conjunto encerrado entre llaves de los vectores propios linealmente independientes asociados a ese valor propio. A continuación se expone un ejemplo (se utiliza el comando "**evalf**" para obtener una mejor apreciación del resultado; su omisión mostraría claramente los resultados exactos):

```
> V:= evalf(eigenvectors(A),3);
```

$$V := [8.60, 1., \{[1., 1.52]\}], [-3.60, 1., \{[1., -0.920]\}]$$

Uno naturalmente puede extraer de ese resultado lo que necesite. Por ejemplo el segundo valor propio y el primer vector propio se obtienen como sigue:

```
> V[2][1];
> V[1][3][1];
```

-3.60

[1., 1.52]

Transformaciones Lineales, Valores y Vectores Propios

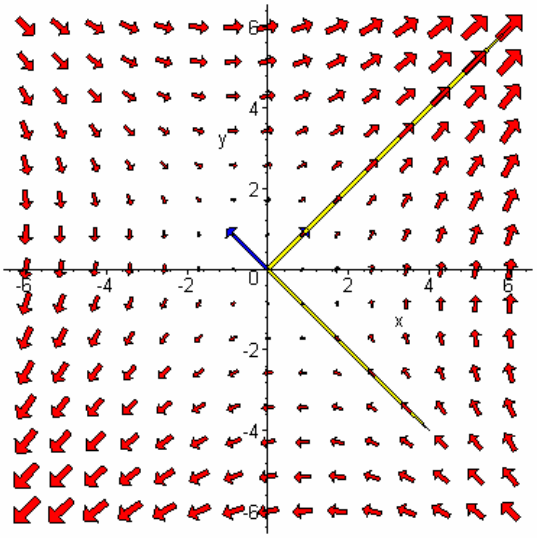
Se muestra a continuación una gráfica que permite visualizar la conexión entre Valores y Vectores Propios y Transformaciones Lineales. En el mismo se abre un paquete de funciones graficas, se calcula el mayor valor propio con el comando "**max**" y se hacen además dos graficas superpuestas.
La primera grafica el campo vectorial asociado a la Transformación Lineal Y=AX en dos dimensiones de partida y llegada. El comando para ello es "**fieldplot**" el cual utiliza tres insumos: la función vectorial a graficar y el rango de graficación de x e y.
El comando "**arrow**" a su vez grafica una serie de vectores propios (en azul) y sus respectivas imágenes (en amarillo). En todos los casos los objetos gráficos tienen anulado su output mediante el anexado final de los dos puntos.
El comando "**display**" se encarga de mostrar simultáneamente en la misma gráfica todas las imágenes. Para mayores detalles sobre estos comandos puede pintar el nombre de la función y presionar F1 lo que lo conducirá a la ayuda proporcionada sobre es función.

```
> with(plottools):
A:= [[1,5],[5,1]];
T:= evalf(eigenvectors(A),3);
r:= max(T[1,1],T[2,1]):
G1:= fieldplot( [A[1,1]*x+A[1,2]*y,A[2,1]*x+A[2,2]*y],x=-
r..r,y=-r..r, color= red,grid=[15,15],arrows=THICK):
G2:= arrow([0,0],evalm(T[1,1]*T[1,3][1]),.08, .08, .1,
color=yellow):
G3:= arrow([0,0],evalm(T[2,1]*T[2,3][1]), .08, .08, .1,
color=yellow):
G4:= arrow([0,0],evalm(T[1,3][1]), .08, .4, .1,
color=blue):
G5:= arrow([0,0],evalm(T[2,3][1]), .08, .4, .1,
color=blue):
display(G1,G2,G3,G4,G5);
```

```
Warning, the name arrow has been redefined
```

$$A := [[1, 5], [5, 1]]$$

$$T := [-4., 1., \{[-1., 1.]\}], [6., 1., \{[1., 1.]\}]$$



Con esta serie de comandos uno simplemente puede cambiar los elementos de la matriz de la transformación lineal y visualizar como cada elemento de la imagen de la transformación pueden descomponerse en términos de los vectores propios y sus respectivas imágenes. Obsérvese como los vectores ubicados en las direcciones de los vectores propios no sufren alteraciones de dirección. Éstos sólo se estiran o se acortan de acuerdo a los valores propios asociados a los mismos.

6.-Formas Cuadráticas Libres y Restringidas

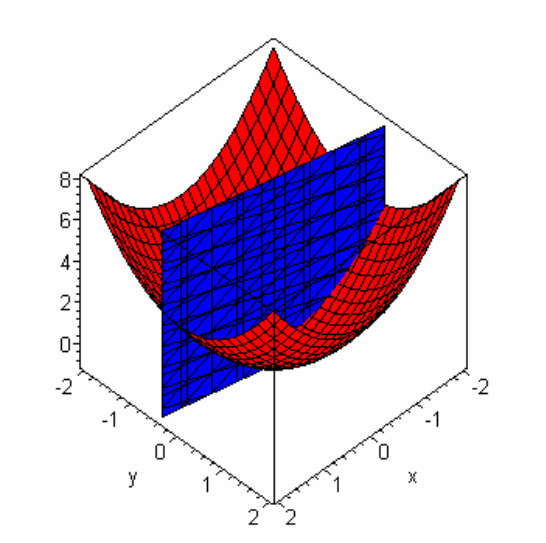
Positivas definidas

Un ejemplo de aplicación para el entorno gráfico es la graficación conjunta de una forma cuadrática bidimensional junto a un plano de restricción a los efectos de observar como bajo determinadas restricciones la forma puede alterar o no su signatura.

Se expone a continuación dos gráficos: la forma cuadrática definida por medio de una matriz y el plano vertical de la restricción de manera implícita. El comando "**display**" se utiliza para mostrar ambas gráficas simultáneamente.

```
>A:=[[1,0],[0,1]]:
B:=[[1,5]]:

with(plots):
G1:= implicitplot3d(B[1,1]*x+B[1,2]*y=0,x=-2..2,y=-2..2,z=-
1..8,color=blue):
G2:=plot3d(A[1,1]*x^2+2*A[1,2]*x*y+A[2,2]*y^2,x=-2..2,y=-
2..2,color=red):
display({G1,G2},axes=BOXED);
```



Animación

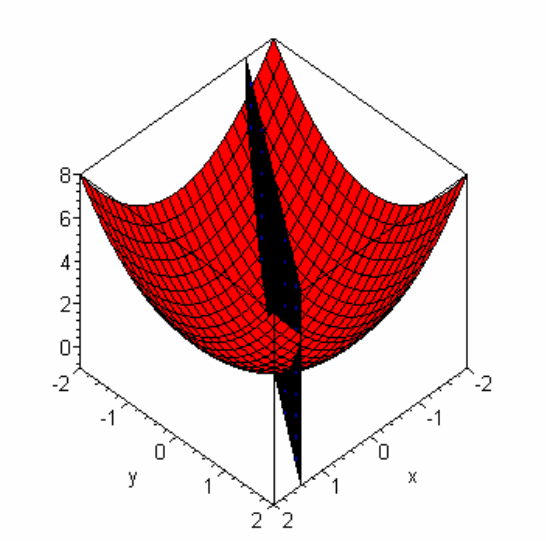
El propósito de la siguiente gráfica animada es brindar una intuición geométrica de como diversas restricciones sobre la forma pueden alterar la signatura original. Se emplean para ello los siguientes comandos:

"**animate3d**" para la grafica de la forma cuadrática libre con la particularidad que el parámetro de la animación (t en este caso) no aparece como argumento en la especificación de la función. Esto se hace a los simples efectos que la forma se quede quieta y solo el plano vertical de restricción se mueva.

El comando "**seq**" crea una sucesión de elementos en este caso "**implicit plot**" para sucesivos valores de t que aparecen como argumentos del plano definido implícitamente por ser este vertical. A su vez se lo compone con el comando "**display**" con la opción de no mostrar a secuencias de gráficas si no que las almacene bajo el nombre B.

Finalmente se muestra la sucesión de gráficas implícitas (es decir la animación del plano vertical de la restricción) con el usado comando "**display**"

```
> with(plots):
A:=[[1,0],[0,1]]:
B := display(seq(implicitplot3d(i*x+5*y=0,x=-2..2,y=-
2..2,z=-1..8,color=blue),i=-7..7), insequence=true):
BB:=animate3d(A[1,1]*x^2+2*A[1,2]*x*y+A[2,2]*y^2,x=-
2..2,y=-2..2,t=-2..2,frames=15,color=red):
display({B,BB},axes=BOXED);
```

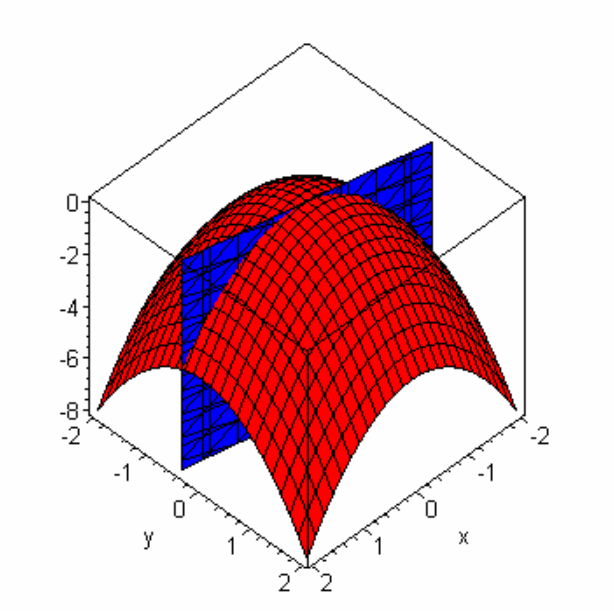


Negativas Definidas

Ejemplo

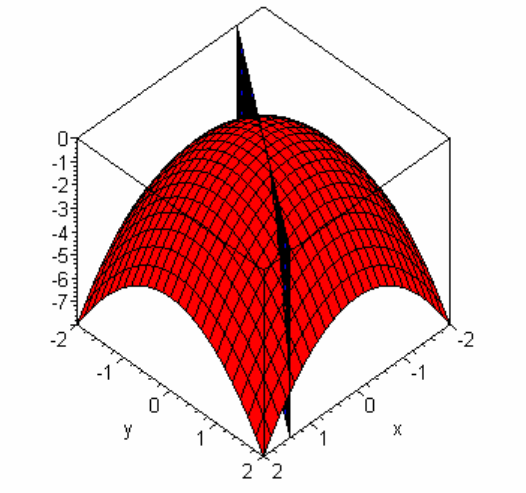
```
> A:=[[-1,0],[0,-1]]:
B:=[[1,5]]:

with(plots):
G1:= implicitplot3d(B[1,1]*x+B[1,2]*y=0,x=-2..2,y=-2..2,z=-
8..0,color=blue):
G2:=plot3d(A[1,1]*x^2+2*A[1,2]*x*y+A[2,2]*y^2,x=-2..2,y=-
2..2,color=red):
display({G1,G2},axes=BOXED);
```



Animación

```
> with(plots):
A:=[[-1,0],[0,-1]]:
B := display(seq(implicitplot3d(i*x+5*y=0,x=-2..2,y=-
2..2,z=-8..0,color=blue),i=-7..7), insequence=true):
BB:=animate3d(A[1,1]*x^2+2*A[1,2]*x*y+A[2,2]*y^2,x=-
2..2,y=-2..2,t=-2..2,frames=15,color=red):
display({B,BB},axes=BOXED);
```

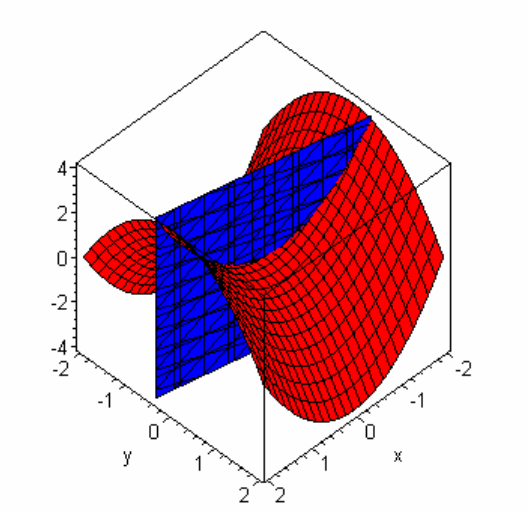


Indefinidas

Ejemplo

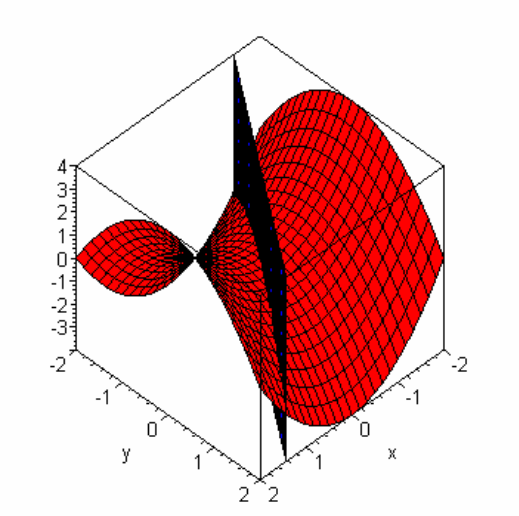
```
> A:=[[1,0],[0,-1]]:
B:=[[1,5]]:

with(plots):
G1:= implicitplot3d(B[1,1]*x+B[1,2]*y=0,x=-2..2,y=-2..2,z=-
4..4,color=blue):
G2:=plot3d(A[1,1]*x^2+2*A[1,2]*x*y+A[2,2]*y^2,x=-2..2,y=-
2..2,color=red):
display({G1,G2},axes=BOXED);
```



Animación

```
> with(plots):
A:=[[1,0],[0,-1]]:
B := display(seq(implicitplot3d(i*x+5*y=0,x=-2..2,y=-
2..2,z=-4..4,color=blue),i=-7..7), insequence=true):
BB:=animate3d(A[1,1]*x^2+2*A[1,2]*x*y+A[2,2]*y^2,x=-
2..2,y=-2..2,t=-2..2,frames=15,color=red):
display({B,BB},axes=BOXED);
```

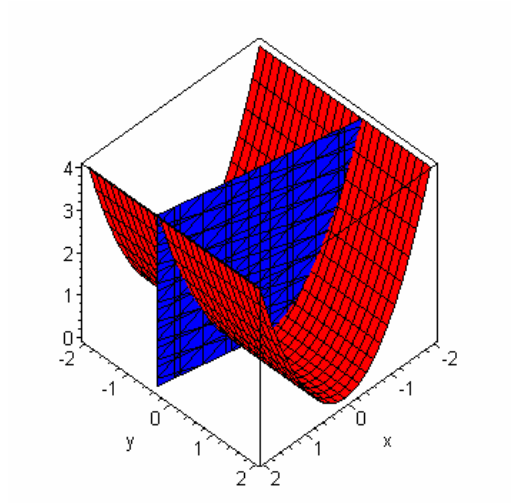


Semi-Definidas Positivas

Ejemplo

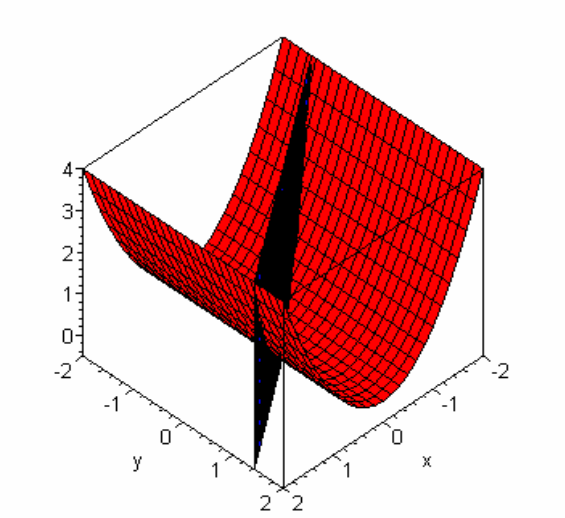
```
>A:=[[1,0],[0,0]]:
B:=[[1,5]]:

with(plots):
G1:= implicitplot3d(B[1,1]*x+B[1,2]*y=0,x=-2..2,y=-2..2,z=-
0..4,color=blue):
G2:=plot3d(A[1,1]*x^2+2*A[1,2]*x*y+A[2,2]*y^2,x=-2..2,y=-
2..2,color=red):
display({G1,G2
},axes=BOXED);
```



Animación

```
> with(plots):
A:=[[1,0],[0,0]]:
B := display(seq(implicitplot3d(5*x+i*y=0,x=-2..2,y=-
2..2,z=-0.5..4,color=blue),i=-7..7), insequence=true):
BB:=animate3d(A[1,1]*x^2+2*A[1,2]*x*y+A[2,2]*y^2,x=-
2..2,y=-2..2,t=-2..2,frames=15,color=red):
display({B,BB},axes=BOXED);
```

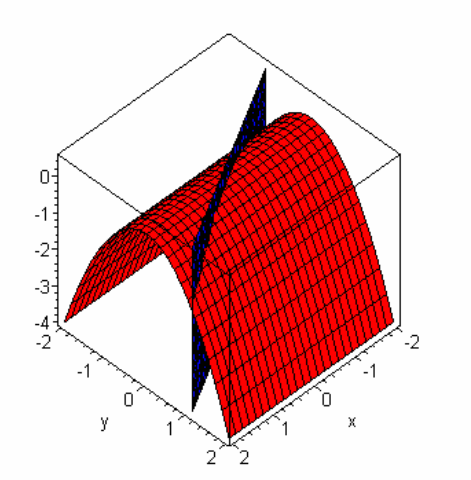


Semi-Definidas Negativas

Ejemplo

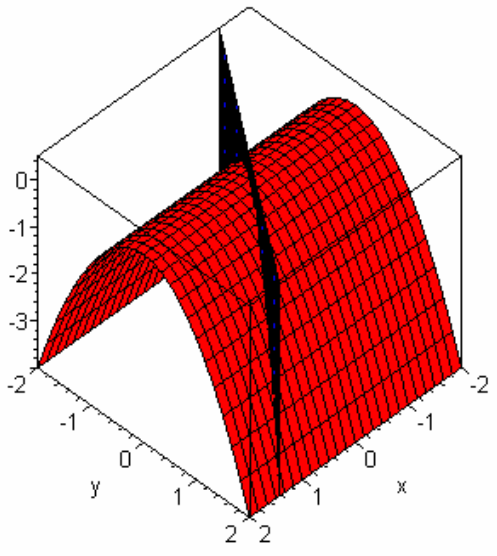
```
> A:=[[0,0],[0,-1]]:
B:=[[-11,20]]:

with(plots):
G1:= implicitplot3d(B[1,1]*x+B[1,2]*y=0,x=-2..2,y=-2..2,z=-
4..0.5,color=blue):
G2:=plot3d(A[1,1]*x^2+2*A[1,2]*x*y+A[2,2]*y^2,x=-2..2,y=-
2..2,color=red):
display({G1,G2},axes=BOXED);
```



Animación

```
> with(plots):
A:=[[0,0],[0,-1]]:
B := display(seq(implicitplot3d(i*x+5*y=0,x=-2..2,y=-
2..2,z=-4..0.5,color=blue),i=-7..7), insequence=true):
BB:=animate3d(A[1,1]*x^2+2*A[1,2]*x*y+A[2,2]*y^2,x=-
2..2,y=-2..2,t=-2..2,frames=15,color=red):
display({B,BB},axes=BOXED);
```



7.-Análisis en Varias Variables

Derivadas Parciales Direccionales - Gradientes - Hessianos - Jacobianos

---

> **with(linalg):**

```
Warning, new definition for norm
Warning, new definition for trace
```

Derivadas univaridas

> **diff(cos(x),x);**

$$-\sin(x)$$

Derivadas Parciales

> **diff(exp(a*cos(x*y^3)),x);**
diff(exp(a*cos(x*y^3)),y);

$$-a \sin(x y^3)\, y^3\, \mathbf{e}^{(a \cos(x y^3))}$$

$$-3\, a \sin(x y^3)\, x\, y^2\, \mathbf{e}^{(a \cos(x y^3))}$$

Gradientes

A la hora de calcular gradientes Maple utiliza el comando **grad** el cual funciona con dos argumentos: una expresión en varias variables y una lista entre corchetes de las variables de dicha expresión. Si existiesen mas variables que no han sido incluidas en el corchete serán interpretadas por Maple como constantes.

> **g:=grad(exp(a*cos(x*y^3)),[x,y]);**

$$g := \left[-a \sin(x y^3)\, y^3\, \mathbf{e}^{(a \cos(x y^3))},\ -3\, a \sin(x y^3)\, x\, y^2\, \mathbf{e}^{(a \cos(x y^3))} \right]$$

Otro comando vinculado a los gradientes resulta ser el campo vectorial generado por los gradientes. Como bien es sabido, la dirección de máximo crecimiento de una función esta indicada por la dirección que demarcan los vectores gradientes. Este hecho puede fácilmente observarse mediante comando **gradplot**. El resto de opciones que siguen son modificaciones estéticas que mejoran la visualización siendo por ende de ingreso de carácter opcional)

```
> gradplot(x^2+y^2,x=-3..3,y=-3..3, grid=[8,8],arrows=THICK,color=red);
```

$$\text{gradplot}(x^2+y^2, x=-3..3, y=-3..3, grid=[8,8], arrows=THICK, color=red)$$

Derivadas Direccionales

Las mismas pueden ser calculadas definiendo un vector dirección y multiplicando por el vector gradiente. Mostramos este ejemplo

---

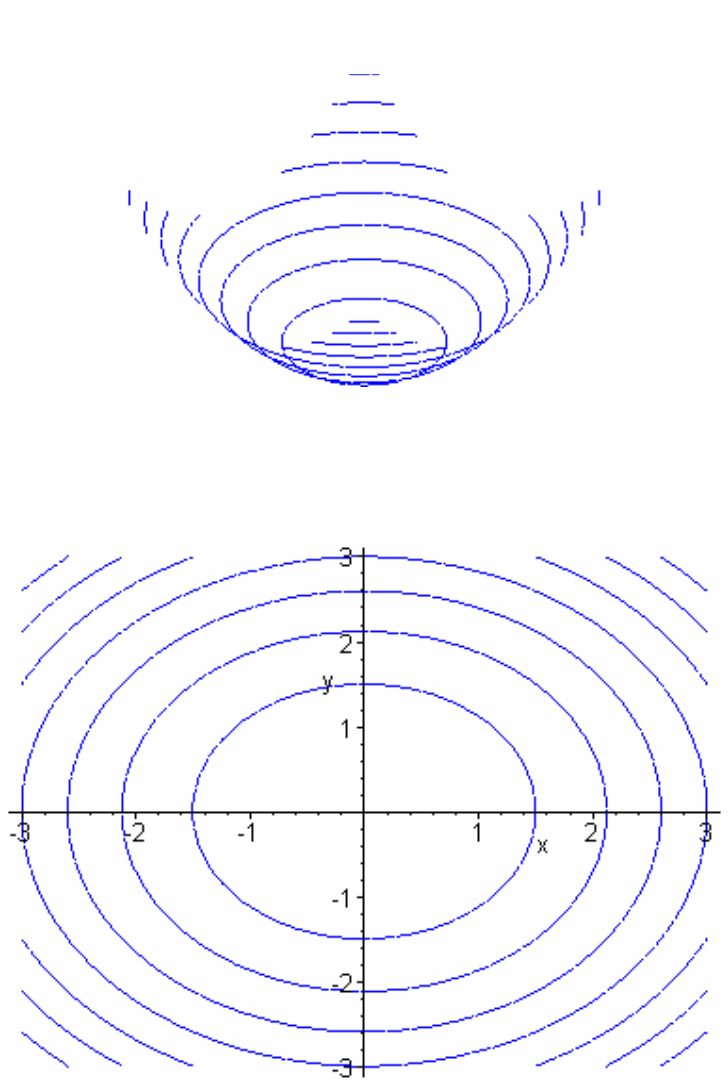
```
> f:=exp(a*cos(x*y^3)):
u:=[1,6]:
multiply(transpose(u),g);
```

$$-a\sin(x\,y^3)\,y^3\,\mathbf{e}^{(a\cos(x\,y^3))} - 18\,a\sin(x\,y^3)\,x\,y^2\,\mathbf{e}^{(a\cos(x\,y^3))}$$

Curvas de Nivel

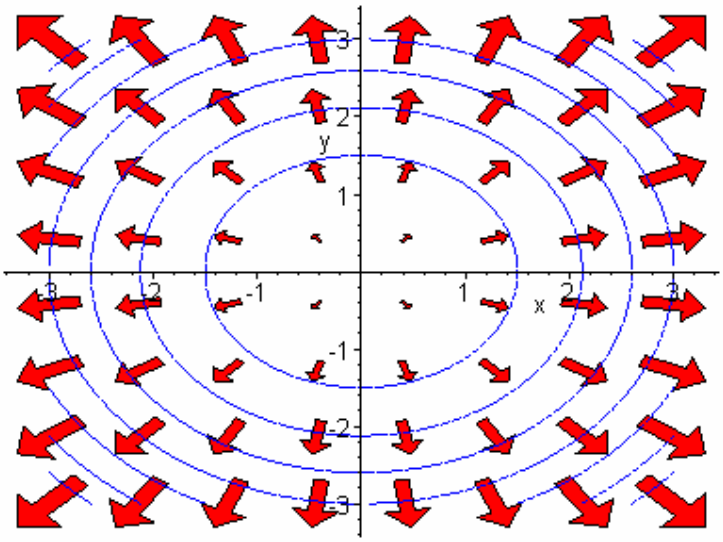
Curvas de Nivel para funciones de 2 y 3 variables se obtienen mediante (obsérvese que primero debe cargarse el paquete de funciones gráficas):

```
> with(plots):
contourplot3d(x^2+y^2,x=-3..3,y=-3..3, color=blue);
contourplot(x^2+y^2,x=-3..3,y=-3..3, color= blue);
```



La perpendicularidad entre gradientes y curvas de nivel puede nítidamente verificarse en el siguiente gráfico definido con comando maple:

> **A:=contourplot(x^2+y^2,x=-3..3,y=-3..3, grid=[40,40], color= blue):**
B:= gradplot(x^2+y^2,x=-3..3,y=-3..3, grid=[8,8],arrows=THICK,color=red):
display(A,B);



Matriz Hessiana

El comando **hessian** permite obtener la matriz hessiana de una función de manera simbólica. Hessian funciona con dos argumentos: la función y el listado de variables valiendo las mismas conclusiones que para el comando **grad**. El resultado se muestra en forma de matriz a la cual pueden efectuarse todas las operaciones mostradas con anterioridad.

> **H:=hessian(a*cos(x*y^3),[x,y]);**

$$H := \begin{bmatrix} -a\cos(x\,y^3)\,y^6 & -3\,a\cos(x\,y^3)\,y^5\,x - 3\,a\sin(x\,y^3)\,y^2 \\ -3\,a\cos(x\,y^3)\,y^5\,x - 3\,a\sin(x\,y^3)\,y^2 & -9\,a\cos(x\,y^3)\,x^2\,y^4 - 6\,a\sin(x\,y^3)\,x\,y \end{bmatrix}$$

Asimismo se puede además evaluar la función hessiana en un punto en particular como se expone a continuación.

> **eval(hessian(a*cos(x*y^3),[x,y]),[x=2,y=5]);**

$$\begin{bmatrix} -15625\,a\cos(250) & -18750\,a\cos(250) - 75\,a\sin(250) \\ -18750\,a\cos(250) - 75\,a\sin(250) & -22500\,a\cos(250) - 60\,a\sin(250) \end{bmatrix}$$

Matriz Jacobiana

Las matrices jacobianas simbólicas son fáciles de computar gracias al comando **jacobian**. Éste admite dos argumentos: el primero un listado de funciones entre corchetes y segundo un listado indicando las variables. Nuevamente el resultado se muestra en forma matricial.

```
> jacobian([x^2*sin(x*y),ln(sin(x*y))],[x,y]);
```

$$\begin{bmatrix} 2\,x\,\sin(x\,y)+x^{2}\cos(x\,y)\,y & x^{3}\cos(x\,y) \\ \dfrac{\cos(x\,y)\,y}{\sin(x\,y)} & \dfrac{\cos(x\,y)\,x}{\sin(x\,y)} \end{bmatrix}$$

A modo de ejemplo se muestran también la relación entre gradientes, jacobianos y hessianos. El hessiano de f es igual al jacobiano del gradiente de f.

```
> jacobian(grad(sin(x*y)*ln(y),[x,y]),[x,y]);
```

$$\begin{bmatrix} -\sin(x\,y)\,y^{2}\ln(y) & -x\,\sin(x\,y)\,y\ln(y)+\cos(x\,y)\ln(y)+\cos(x\,y) \\ -x\,\sin(x\,y)\,y\ln(y)+\cos(x\,y)\ln(y)+\cos(x\,y) & -x^{2}\sin(x\,y)\ln(y)+2\dfrac{\cos(x\,y)\,x}{y}-\dfrac{\sin(x\,y)}{y^{2}} \end{bmatrix}$$

```
> hessian(sin(x*y)*ln(y),[x,y]);
```

$$\begin{bmatrix} -\sin(x\,y)\,y^{2}\ln(y) & -x\,\sin(x\,y)\,y\ln(y)+\cos(x\,y)\ln(y)+\cos(x\,y) \\ -x\,\sin(x\,y)\,y\ln(y)+\cos(x\,y)\ln(y)+\cos(x\,y) & -x^{2}\sin(x\,y)\ln(y)+2\dfrac{\cos(x\,y)\,x}{y}-\dfrac{\sin(x\,y)}{y^{2}} \end{bmatrix}$$

Derivadas de Orden n - Polinomios de Taylor (Introd. a la Programación en Maple)

Procedimientos para Crear Derivadas de orden n

El Siguiente Procedimiento crea un nuevo comando para Maple que calcula la derivada de orden "n" para f con respecto a una variable

---

```
> Dn := proc(f,x,n) local Dfn ,i:
Dfn := f:
for i from 1 by 1 to n
do
Dfn := diff(Dfn,x)
od:
Dfn:
end;
```

Dn := **proc**(*f*, *x*, *n*) **local** *Dfn*, *i*; *Dfn* := *f*; **for** *i* **to** *n* **do** *Dfn* := diff(*Dfn*, *x*) **od**; *Dfn* **end**

Para crear procedimientos o nuevas funciones reconocidas por el lenguaje de Maple se debe proceder de la siguiente forma:

1.- Definir el nombre de la nueva función por medio del comando **proc**. Este último es una función que recibe tantas variables o argumentos como argumentos deba tener la nueva función creada. En este caso el nuevo comando creado se denomina "**Dn**" y será una nueva función que reciba tres argumentos llamados genéricamente f, x y n y en ese órden.

2. Escribir el conjunto de instrucciones en términos de los comandos existentes en Maple de moda tal que la nueva función creada resuma dicho procedimiento en un solo paso. En este caso se definen internamente y de manera temporaria las variables locales (una variable local es una variable que se define internamente dentro del procedimiento los efectos de efectuar las tareas solicitadas. Una vez finalizado el procedimiento la variable local es eliminada de la memoria de la máquina). Para el caso del ejemplo particular se definen como locales las variables **Dfn** e **i**. **Dfn** es asignada primero al valor de la expresión de *f* el cual es el primer argumento de la función **Dn**. Con esa definición se procede a derivar de manera repetitiva y cíclica la expresión **Dfn** con respecto a x (el segundo argumento de **Dn**) redefiniéndola en cada derivación de modo que se derive sucesivamente n veces siendo n el valor del tercer argumento de la función creada. Nótese como el comando **for** se utilizó para repetir la operación de derivar y redefinir **Dfn**. El mismo indica que todas las instrucciones que estén contenidas dentro de los comandos do y **od** han de repetirse mientras i evoluciones desde 1 hasta n dando saltos de 1.
Finalmente luego del comando **for** y **do**. debe indicarse cual ha de ser el resultado que debe arrojar la función creada luego de ejecutarse. Nótese como no deben colocarse los dos puntos pues de hacerlo así la función no arrojará output.

3.- Finalizar el comando **proc** con la sentencia "**end**"

De esta manera hemos creado una función llamada **Dn** que posee tres argumentos: la expresión a derivar, la variable con respecto a la cual se deriva y el orden de derivación.

A modo de ejemplo mostramos su uso para el cálculo de la derivada de orden 3 para *f(x)=x^7*y^2*z*

```
> Dn(x^7*y^2*z,x,4);
```

$$840\, x^3 y^2 z$$

Comando para calcular un polinomio de Taylor para un función cualquiera

A continuación se muestra otro de los usos de la función creada **Dn** para el computo del polinomio de Taylor en una variable. Nótese como por medio del comando **seq** se crea una secuencia de los términos del polinomio del Taylor hasta el orden n el cual encerrado entre corchetes se interpreta como un vector de n + 1 componentes. Asimismo observe como se creo un vector de n +1 componentes conteniendo nada mas que unos en sus entradas de modo que luego vía la multiplicación matricial con lo definido anteriormente se obtenga la suma de los términos del Polinomio tal y cual se quería en un principio.

```
> f:=sin(x):
n:=7:
xo:=0:
T:=[seq((1/i!)*eval(Dn(f,x,i),x=xo)*(x-xo)^i,i=0..n)]:
s:=[seq(1,i=1..n+1)]:
Taylor:=multiply(transpose(s),T);
```

$$Taylor := x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7$$

De esta manera la expresión Taylor puede ahora ser evaluada en cualquier punto, en este caso x=1, para obtener una aproximación de orden n del Polinomio de Taylor de f expandido en $x = 0$.

```
> eval(Taylor(sin(x),x,n,0),x=1);
```

$$\frac{4241}{5040}$$

Habiendo comprendido la esencia del cómputo anterior la intención ahora es crear un procedimiento llamado **Taylor** que utilice cuatro argumentos (la expresión funcional a

desarrollar, la variable, el orden de aproximación del polinomio de Taylor y el punto de expansión de la serie)

```
> Taylor:= proc(f,x,n,xo)
local T,s,i:
T:=[seq((1/i!)*eval(Dn(f,x,i),x=xo)*(x-xo)^i,i=0..n)]:
s:=[seq(1,i=1..n+1)]:
multiply(transpose(s),T):
end;
```

```
Taylor := proc(f, x, n, xo)
local T, s, i;
    T := [seq(eval(Dn(f, x, i), x = xo)*(x - xo)^i / i!, i = 0 .. n)]; s := [seq(1, i = 1 .. n + 1)]; multiply(transpose(s), T)
end
```

Se procede ahora a utilizar el comando Taylor para la expansión en serie de la función seno de orden 9 en x=0. Fíjese como también es posible utilizar este nuevo comando con las demás funciones existentes en Maple tales como **eval** para obtener la aproximacion del valor del seno de pi/2

```
> eval(Taylor(sin(x),x,9,0),x=Pi/2);
```

$$\frac{1}{2}\pi - \frac{1}{48}\pi^3 + \frac{1}{3840}\pi^5 - \frac{1}{645120}\pi^7 + \frac{1}{185794560}\pi^9$$

```
> evalf(%,10);
```

$$1.000003543$$

Polinomio de Taylor en varias variables

Para obtener el polinomio de Taylor una función en varias variables en base a una fórmula de Taylor univariada se procede de la siguiente manera siguiendo justamente la demostración y la esencia de las Series de Potencias.
El polinomio de Taylor de f (X), donde X es un vector de n componentes, expandido en X = Xo, es simplemente el polinomio de Taylor univariado en la variable t expandido en t =0 y evaluado luego en t = 1 donde t es una nueva variable que surge del siguiente cambio de coordenadas: X --> Xo - t (X - Xo).
En base a esta idea se realizan los cómputos necesarios

```
> f:=sin(x*y):
X:=[x,y]:
Xo:=[xo,yo]:
XX:=evalm(Xo+t*(X-Xo));
f:=eval(sin(x*y),[x=XX[1],y=XX[2]]);
```

$$XX := [t\,x, yo + t\,(-yo + y)]$$

$$f := \sin(t\,x\,(yo + t\,(-yo + y)))$$

```
> simplify(eval(Taylor(f,t,5,0),t=1));
```

$$x\,y - \frac{1}{6}x^3\,yo^3 + \frac{1}{2}x^3\,yo^2\,y + \frac{1}{120}x^5\,yo^5 - \frac{1}{2}x^3\,yo\,y^2$$

Comprobando que la función funciona correctamente. Pare ello comparamos con la función interna de Maple ya preparada para este problema "**mtaylor**". La misma utiliza tres argumentos: la función, el punto multidimensional en donde se evalúa (por medio de un vector) y el numero de términos de la Serie. Observe como el numero de términos del Polinomio es igual al orden de expansión mas uno.

```
simplify(mtaylor(sin(x*y),[x=xo,y=yo],6));
```

$$x\,y - \frac{1}{6}x^3\,yo^3 + \frac{1}{2}x^3\,yo^2\,y + \frac{1}{120}x^5\,yo^5 - \frac{1}{2}x^3\,yo\,y^2$$

Aproximaciones - Animaciones

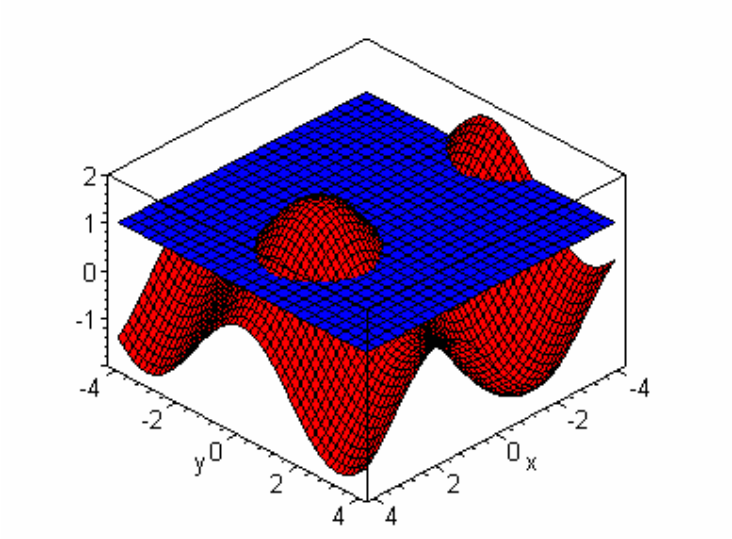
Se presenta a continuación una serie de comandos que grafican simultáneamente una *f* de dos variables y sus sucesivas aproximaciones de Taylor hasta orden n. Obsérvese como primero se crea una animación de la *f* con un parámetro t que no existe en su definición como variable. Esto es solo a los efectos de mantener en la animación conjunta de *f* y Taylor a *f* sin movimientos. Obsérvese también como se generan una sucesión de gráficas en 3d de los sucesivos Polinomios de Taylor vía el comando **seq** que indica el orden de la función **mtaylor**. El comando **display** con la opción: **insequence=true** mantiene en la memoria las gráficas sin mostrarlas y solo se mostrarán como animación cuando el comando **display** las llame.

```
> with(plots):
f :=sin(x)+cos(y):
n := 8:
```

```
A := animate3d(f,x=-1.3*Pi..1.3*Pi,y=-1.3*Pi..1.3*Pi,t=-
1..1,frames=n,axes=BOXED,color=red,grid=[50,50]):

B := display(seq(plot3d(convert(mtaylor(f,[x=0,y=0], i), polynom),
x=-1.3*Pi..1.3*Pi,y=-1.3*Pi..1.3*Pi,axes=BOXED,color = blue), i=1..n),
insequence=true):

display([A,B],view=-2..2);
```



8.-Optimización Simbólica

En esta sección se describe el funcionamiento de los nuevos comandos `Libre`, **`lagrange`** y `Maxkt`. Los mismos has sido definidos n base a las reglas usuales de optimización simbólico para cada respectivo tipo de problemas. Para un detalle mayor de cómo se construyeron y de los contenidos teóricos que los respaldan véase Oviedo (2005)

```
> restart;
libre:=proc(f,X)
local A, j, k, B;
print(_____________________________________________________
_______);
_EnvExplicit := true:
A:=[solve({seq(diff(f,X[i])=0,i=1..nops(X))},{seq(X[i],i=1.
.nops(X))})]:
for j from 1 by 1 to nops(A) do
```

```
print(Punto_Critico[j]);
print([A[j],fvalue=eval(f,A[j])]);
print(Hessian[j]=eval(linalg[hessian](f,X),A[j]));
B:= eval(linalg[hessian](f,X),A[j]):
print(Minors[j]=[seq(linalg[det](linalg[submatrix](B, 1..k,
1..k)),k=1..nops(X))]);
print(_____
_______);
od;


end:


lagrange:=proc(h,g,v,l)
local A, j, k, B, f, T, X, m, q, G, GT, Ceros, HH;
f:= h-linalg[multiply](linalg[transpose](l),g):
T:=
convert(linalg[stackmatrix](linalg[transpose]([v]),linalg[t
ranspose]([l])),vector):
X:=[seq(T[m],m=1..(nops(v)+nops(l)))]:
q:= 2*nops(l)+1:
G:=convert([seq([seq(diff(g[j],v[i]),i=1..nops(v))],j=1..no
ps(g))],matrix):
Ceros:=convert([seq([seq(0,i=1..nops(g))],j=1..nops(g))],ma
trix):
GT:=linalg[transpose](G):
HH:=linalg[hessian](f,v):
print(_____
_______);
_EnvExplicit := true:
A:=[solve({seq(diff(f,X[i])=0,i=1..nops(X))},{seq(X[i],i=1.
.nops(X))})]:
for j from 1 by 1 to nops(A) do
print(Punto_Critico[j]);
print(A[j]);
print(fvalue=eval(h,A[j]));
print(Hessian[j]=eval(linalg[stackmatrix](linalg[concat](Ce
ros,G),linalg[concat](GT,HH)),A[j]));
B:=
eval(linalg[stackmatrix](linalg[concat](Ceros,G),linalg[con
cat](GT,HH)),A[j]):
print(Minors[j]=[seq(linalg[det](linalg[submatrix](B, 1..k,
1..k)),k=q..nops(X))]);
print(_____
_______);
od;


end:
```

```
Maxkt:=proc(h,g,v,l)
local A, j, k, B, f, T, X, m, q, qq, t, u, uu, gg, uuu, L;
f:= h-linalg[multiply](linalg[transpose](l),g):
T:=
convert(linalg[stackmatrix](linalg[transpose]([v]),linalg[t
ranspose]([l])),vector):
X:=[seq(T[m],m=1..(nops(v)+nops(l)))]:
q:= 2*nops(l)+1:
L:=[]:
print(_____________________________________________________
_______);
_EnvExplicit := true:

A:=[solve({seq(X[i]*diff(f,X[i])=0,i=1..nops(X))},{seq(X[i]
,i=1..nops(X))})]:
gg:=[]:
for j from 1 by 1 to nops(A) do
qq:=0:
for t from 1 by 1 to nops(v) do
if(evalf(eval(diff(f,v[t]),A[j]))<=0) then qq:=qq+1 fi:
od;
for u from 1 by 1 to nops(l) do
if(evalf(eval(diff(f,l[u]),A[j]))>=0) then qq:=qq+1 fi:
od;
for uu from 1 by 1 to nops(X) do
if(evalf(eval(X,A[j]))[uu]>=0) then qq:=qq+1 fi:
od;

if(qq=2*nops(X)) then
print(Punto_Critico[j]);
print(A[j]);
print(fvalue=eval(f,A[j]));
gg:= [op(gg),eval(f,A[j])]:
L := [op(L),A[j]]:

print(_____________________________________________________
_______);
fi;

od;

for uuu from 1 by 1 to nops(gg) do
if(max(seq(gg[ii],ii=1..nops(gg)))=gg[uuu]) then
print([MaxGlobal_fvalue=gg[uuu],L[uuu]]) fi;
od;

end:
```

Optimización Libre

Gracias a los Comandos generados anteriormente usando los conocimientos esenciales sobre optimización, condiciones de primer y segundo orden la optimización libre puede resolverse fácilmente con el comando **libre**. La sintaxis se describe a continuación ilustrada con algunos ejemplos. El output de Maple para esto son los puntos críticos junto a las matrices hessianas y sus respectivos menores principales líderes para cada uno de sus puntos críticos.

Sintaxis

```
libre ( función, listado de variables independientes );
```

Ejemplo

```
> libre(x^3+3*x*y^2-x*y,[x,y]);
```

$$Punto\_Critico_1$$

$$[\{x = 0, y = 0\}, fvalue = 0]$$

$$Hessian_1 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

$$Minors_1 = [0, -1]$$

$$Punto\_Critico_2$$

$$\left[\{x = 0, y = \frac{1}{3}\}, fvalue = 0\right]$$

$$Hessian_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$Minors_2 = [0, -1]$$

$$Punto\_Critico_3$$

$$\left[\{y = \frac{1}{6}, x = \frac{1}{6}\}, fvalue = \frac{-1}{108}\right]$$

$$Hessian_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$Minors_3 = [\,1,\ 1\,]$$

$$Punto\_Critico_4$$

$$\left[\ \{y = \frac{1}{6},\ x = \frac{-1}{6}\},\ fvalue = \frac{1}{108}\ \right]$$

$$Hessian_4 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$Minors_4 = [\,-1,\ 1\,]$$

Caso especial nº 1

A continuación se muestra donde la condición de segundo brinda una forma cuadrática semidefinida negativa y sin embargo la función presenta un claro punto de silla. Se exponen a continuación las salidas vía el comando libre y sucesivamente se efectúan cortes de la *f* con planos verticales que pasan por el punto crítico y se aproxima la gráfica de la función con Polinomios de Taylor.

```
> libre(x*(x-y^2),[x,y]);
```

$$Punto\_Critico_1$$

$$[\,\{x = 0,\ y = 0\},\ fvalue = 0\,]$$

$$Hessian_1 = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$Minors_1 = [\,2,\ 0\,]$$

$$Punto\_Critico_2$$

$$[\,\{x = 0,\ y = 0\},\ fvalue = 0\,]$$

$$Hessian_2 = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$Minors_2 = [\,2,\ 0\,]$$

$$Punto\_Critico_3$$

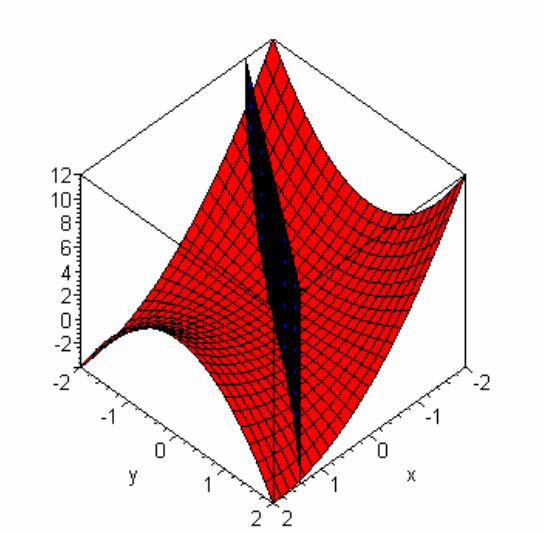
$$[\{x = 0, y = 0\}, fvalue = 0]$$

$$Hessian_3 = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

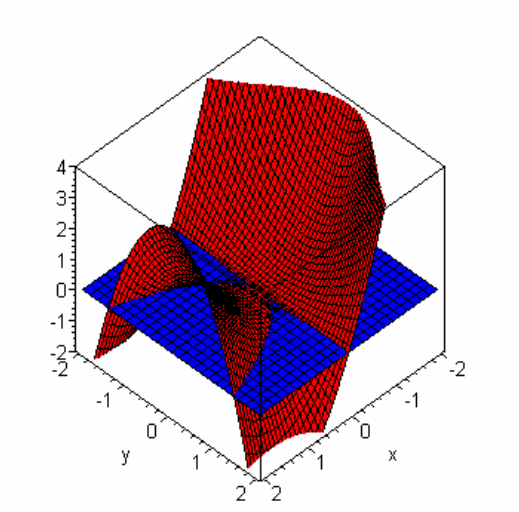
$$Minors_3 = [2, 0]$$

```
> with(plots):
B := display(seq(implicitplot3d(i*x+5*y=0,x=-2..2,y=-
2..2,z=-4..12,color=blue),i=-7..7), insequence=true):
BB:=animate3d(x*(x-y^2),x=-2..2,y=-2..2,t=-
2..2,frames=15,color=red):
display({B,BB},axes=BOXED);
```

```
Warning, the name changecoords has been redefined
```



```
> f :=x*(x-y^2):
n := 8:
A := display(seq(plot3d(convert(mtaylor(f,[x=0,y=0], i),
polynom),
        x=-2..2,y=-2..2,axes=BOXED,color = blue),
i=1..n+1), insequence=true):
B := animate3d(f,x=-2..2,y=-2..2,t=-
1..1,frames=n,axes=BOXED,color=red,grid=[50,50]):
display([A,B],view=-2..4);
```



Caso especial nº 2: La silla de Mono

Otro ejemplo de casos especiales ante los que fallan las condiciones de segundo orden es la bien conocida silla de mono cuya expresión algebraica es *f*(x,y) = *x*^3-*x***y*^2,. La particularidad de esta función esta en que en el [0,0] la misma posee plano tangente horizontal pero no solo esto, sino que su la forma cuadrática que mejor se pega a *f* en ese punto es precisamente la forma nula, es decir otro plano horizontal nuevamente. Se muestran las salidas de Maple junto a una gráfica animada que muestra los Polinomios de Taylor de ordenes sucesivos que la aproximan a *f* .

```
> libre(x^3-x*y^2,[x,y]);
plot3d(x^3-3*x*y^2, x=-2..2, y=-2..2);
```

$$Punto\_Critico_1$$

$$[\{x = 0, y = 0\}, fvalue = 0]$$

$$Hessian_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

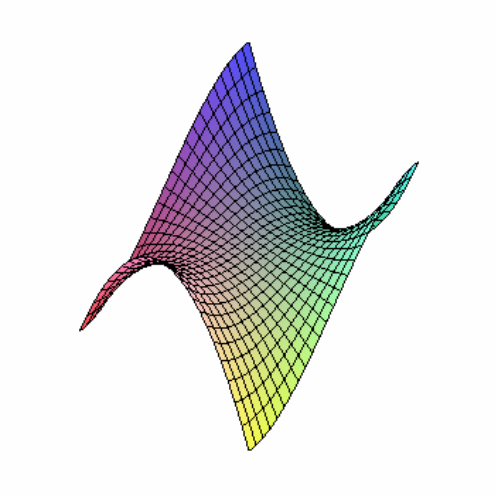
$$Minors_1 = [0, 0]$$

$$Punto\_Critico_2$$

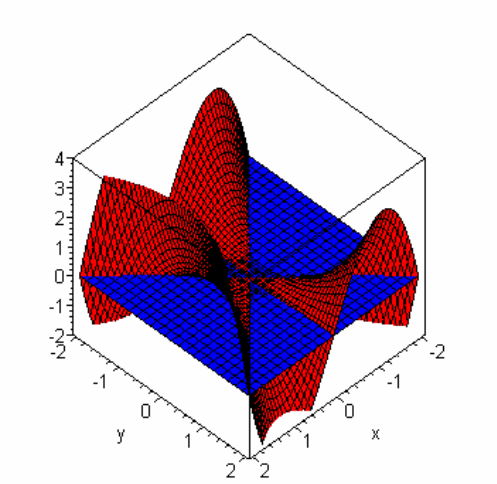
$$[\{x = 0, y = 0\}, fvalue = 0]$$

$$Hessian_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$Minors_2 = [0, 0]$$



```
> with(plots):
f :=x^3-x*y^2:
n := 3:
A := display(seq(plot3d(convert(mtaylor(f,[x=0,y=0], i),
polynom),
        x=-2..2,y=-2..2,axes=BOXED,color = blue),
i=1..n+1), insequence=true):
B := animate3d(f,x=-2..2,y=-2..2,t=-
1..1,frames=n,axes=BOXED,color=red,grid=[50,50]):
display([A,B],view=-2..4);
```



Optimización. Restricciones de igualdad

Los tradicionales problemas de optimización restringida con igualdades se resuelven por

medio de la siguiente sintaxis

Sintaxis

"**Lagrange" (función, listado de restricciones igualadas a cero, listado de variables independientes, listado de multiplicadores de lagrange);"**

Ejemplo 1

Max f = x^2+y^2
st: 3x+5y=10

```
> lagrange(x^2+y^2,[3*x+5*y-10],[x,y],[lambda]);
with(plots):
A:=contourplot(x^2+y^2,x=-3..3,y=-3..3, grid=[40,40],
color= red):
B:= gradplot(x^2+y^2,x=-3..3,y=-3..3,
grid=[8,8],color=red):
AA:=contourplot(3*x+5*y,x=-3..3,y=-3..3, grid=[40,40],
color= blue):
BB:= gradplot(3*x+5*y,x=-3..3,y=-3..3,
grid=[8,8],color=blue):
DD:=implicitplot(3*x+5*y-10,x=-3..3,y=-3..3,color=blue):
display(A,AA,BB,B,DD);
```

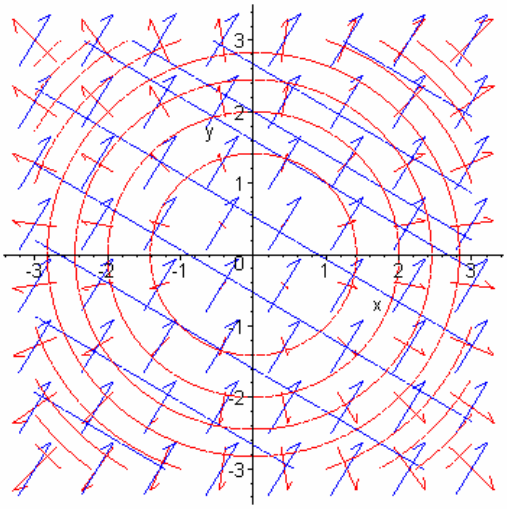
$$Punto\_Critico_1$$

$$\{\lambda = \frac{10}{17}, y = \frac{25}{17}, x = \frac{15}{17}\}$$

$$fvalue = \frac{50}{17}$$

$$Hessian_1 = \begin{bmatrix} 0 & 3 & 5 \\ 3 & 2 & 0 \\ 5 & 0 & 2 \end{bmatrix}$$

$$Minors_1 = [-68]$$



Ejemplo 2
El clásico problema de la conducta del consumidor se resuelve ahora de la siguiente manera:

Max $U(X,Y) = a\ ln(X) + b\ ln(Y)$
st: $Px\ X + Py\ Y = 10$

```
> lagrange(a*ln(x)+b*ln(y),[Px*x+Py*y-M],[x,y],[lambda]);
```

$$Punto\_Critico_1$$

$$\{x = \frac{M\,a}{(a+b)\,Px},\ \lambda = \frac{a+b}{M},\ y = \frac{b\,M}{Py\,(a+b)}\}$$

$$fvalue = a\ \ln\left(\frac{M\,a}{(a+b)\,Px}\right) + b\ \ln\left(\frac{b\,M}{Py\,(a+b)}\right)$$

$$Hessian_1 = \begin{bmatrix} 0 & Px & Py \\ Px & -\frac{(a+b)^2\,Px^2}{a\,M^2} & 0 \\ Py & 0 & -\frac{Py^2\,(a+b)^2}{b\,M^2} \end{bmatrix}$$

$$Minors_1 = \left[\frac{Px^2\,Py^2\,(a+b)^3}{b\,M^2\,a}\right]$$

Optimización. Restricciones de desigualdad

Sintaxis

Las condiciones de Kuhn Tucker son también arrojadas por el comando "`Maxkt`." Sintaxis y ejemplos se muestran seguidamente.

"**Maxkt (función, listado de restricciones igualadas a cero "del tipo menor que" , listado de variables independientes, listado de multiplicadores de lagrange)";**

Ejemplo

Max f = x^2 + y^2
st: 3x + 5y < 10
x >= 0
y >= 0

> **Maxkt(x^2+y^2,[3*x+5*y-10],[x,y],[l]);**

$$Punto\_Critico_1$$

$$\{l = 0, y = 0, x = 0\}$$

$$fvalue = 0$$

$$Punto\_Critico_2$$

$$\{l = 0, y = 0, x = 0\}$$

$$fvalue = 0$$

$$Punto\_Critico_3$$

$$\{x = \frac{10}{3}, l = \frac{20}{9}, y = 0\}$$

$$fvalue = \frac{100}{9}$$

$$Punto\_Critico_4$$

$$\{y = 2, l = \frac{4}{5}, x = 0\}$$

$$fvalue = 4$$

$$Punto\_Critico_5$$

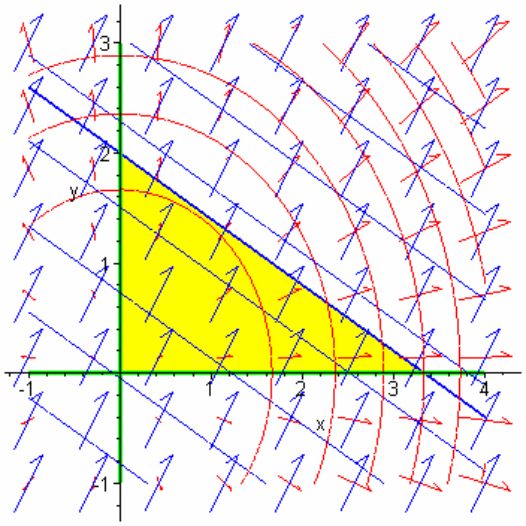
$$\{y = \frac{25}{17}, x = \frac{15}{17}, l = \frac{10}{17}\}$$

$$fvalue = \frac{50}{17}$$

$$\left[MaxGlobal\_fvalue = \frac{100}{9}, \{x = \frac{10}{3}, l = \frac{20}{9}, y = 0\} \right]$$

A continuación se muestra una gráfica donde se permiten visualizar las funciones, los restricciones, los gradientes y los óptimos.

```
> with(plots):
A:=contourplot(x^2+y^2,x=-1..4,y=-1..3, grid=[40,40],
color= red):
B:= gradplot(x^2+y^2,x=-1..4,y=-1..3,
grid=[8,8],color=red):
AA:=contourplot(3*x+5*y,x=-1..4,y=-1..3, grid=[40,40],
color= blue):
BB:= gradplot(3*x+5*y,x=-1..4,y=-1..3,
grid=[8,8],color=blue):
DD:=inequal( {3*x+5*y <10 , x >= 0, y >=0 }, x=-1..4, y=-
1..3,
optionsfeasible=(color=yellow), optionsopen=(color=blue,
thickness=2), optionsclosed=(color=green, thickness=3),
optionsexcluded=(color=white) ):
display(A,AA,BB,B,DD);
```



Obsérvese como el punto crítico 3 no es un máximo relativo a pesar que surge de las condiciones de Kuhn Tucker. Esto es así porque las mismas son condiciones necesarias pero no suficientes. Para ellos seria necesario evaluar el Hessiano orlado con las restricciones que se verifican en forma de igualdad en dicho punto crítico.

Referencias:

Maplesoft, *“Manual de Ayuda de Maple 8 ” Waterllo. Canada 2003*

Jorge Mauricio Oviedo, *“Optimización con Maple”.* Departamento de Estadística y Matemática. Facultad de Ciencias Económicas. Universidad Nacional de Córdoba. 2005.

Sara Aguarón Iraola, Unai Arrieta Salgado, y otros, *"Aprenda Maple como si estuviera en primero"*. Escuela Superior de Ingenieros. Universidad de Navarra 2004.