Gráficos en el plano y el espacio

Autora: Lic. Alicia María Cornacchione

Este archivo contiene explicaciones de manejo elemental del software *Mathematica* y los comandos necesarios para realizar todo tipo de gráficos en el plano y el espacio.
El programa en la actualidad está en la versión 5, en el Centro de Cómputos podrás acceder a la 3.

Indice:

1. Manejo básico de Mathematica

Celdas de entrada y salida

El texto que estamos leyendo, es parte de un cuaderno (notebook) de Mathematica. En él se pueden efectuar cálculos, realizar gráficos de distinto tipo y escribir notas de manera similar a como se hace en un cuaderno con hojas de papel.
Los cuadernos están organizados por celdas. Los límites de las celdas están marcados por los **corchetes que se ven en el lado derecho de la pantalla**. Una celda juega el mismo papel que un párrafo en un texto normal, con la diferencia que cada celda contiene objetos de distinta clase: comandos, cálculos producidos por *Mathematica*, gráficos, texto e incluso otras celdas. Cada celda puede tener su propio estilo, tipo de letra, tamaño, color, etc...
Los cuadernos o notebooks, son archivos que se pueden cargar en memoria para ser modificados.

Veamos como se trabaja con una celda que contiene una **instrucción que *Mathematica* pueda leer y ejecutar**.
Para que la instrucción se ejecute:

1. Colocamos la flecha del mouse dentro de la celda y presionamos el botón izquierdo.
2. Tecleamos Shift y Enter simultáneamente (o tecleamos Shift y manteniendo esta tecla pulsada, tecleamos Enter); otra posibilidad es teclear Enter del teclado numérico.

Por ejemplo:

```
In[1]:= 15 - 4 + 2

Out[1]= 13
```

Observá que la celda que contiene la operación a calcular, es designada por el programa como In[1]. Esta celda se llama ***celda de entrada o Input***; el número que figura entre corchetes indica el número de entradas que se ejecutaron.
Al resultado de la operación, el programa lo denomina Out[1]; esta celda se llama ***celda de salida u Output.***

Para crear celdas de entrada y evaluarlas: mueva la flecha del mouse hasta que esté precisamente debajo de una celda, la flecha cambiará a una **línea horizontal corta**. Pulsá el botón izquierdo del mouse y la línea se extenderá hasta cruzar la hoja de lado a lado.
Escribir el cálculo o instrucción y pulsar Shift+Enter

Para modificar un cálculo, no es necesario reescribirlo, basta posicionarse en el Input a modificar, cambiar lo que se desee y ejecutar la operación.

Cálculo

Diferentes precisiones en el cálculo

Mathematica siempre realiza los cálculos en forma exacta. Para obtener una aproximación hay que colocar después del cálculo //N.

```
In[2]:= √2

Out[2]= √2

In[3]:= √2 // N

Out[3]= 1.41421
```

Paréntesis, corchetes y llaves.

( ) : Es lo único que puede utilizarse en los cálculos o expresiones algebraicas cuando queremos agrupar para indicar el orden de evaluación.

```
In[4]:= (60 - ((12 - 8) × 5)) / 4

Out[4]= 10

In[5]:= {60 - [(12 - 8) * 5]} / 4

        Syntax::sntxf : "60 -" cannot be followed by "[(12 - 8) * 5]". More…

        {60 - [(12 - 8) * 5]} / 4
```

El último cálculo sigue el uso tradicional de los paréntesis, corchetes y llaves, pero para *Mathematica* no tiene significado, por eso marca error de escritura.

[ ] : Se utiliza para delimitar las variables o argumentos de las funciones (en esta misma clase veremos funciones).
{ } : Se utiliza para definir listas de elementos, vectores y matrices.

Espacios

Un espacio entre dos números o variables, **siempre** se interpreta como una multiplicación.
Hay comandos que contienen más de una palabra; en ese caso **no** debe dejarse espacio entre ellas. Por ejemplo: MatrixForm.

Mayúsculas y minúsculas.

Para el programa hay diferencias sustanciales entre las palabras escritas con mayúscula y minúscula.
Se escriben comenzando con mayúscula todas las funciones contenidas en el programa y los comandos.
Por ejemplo: Abs[x] (módulo de x), Cos[x] (cos x) son algunos ejemplos de tales funciones.

```
In[5]:= cos[π]

        General::spell1 : Possible spelling error: new symbol name "cos" is similar to existing symbol "Cos". More…

Out[5]= cos[π]

In[6]:= Cos[π]

Out[6]= -1
```

Como conseguir ayuda

1. *Mathematica* proporciona una descripción de cualquier función o comando, escribiendo el nombre de la instrucción y utilizando los símbolos ? y * como sigue:

?C Dará información sobre el comando C.

```
In[7]:= ? Plot

        Plot[f, {x, xmin, xmax}] generates a plot of f as a function of x from xmin to
           xmax. Plot[{f1, f2, ... }, {x, xmin, xmax}] plots several functions fi. More…
```

2. También se puede conseguir ayuda eligiendo **Help** del menú principal. El Help, además de contener la explicación de todos los comandos, en algunos temas incluye ejemplificación (Versión 4 y 5 de *Mathematica*).

Escritura matemática y Palettes

El programa permite escribir operaciones, expresiones y símbolos matemáticos con la notación que se utiliza en la escritura tradicional.
Ir a File del menú principal, y en la opción **Palettes** marcar, por ejemplo BasicImput. Aparece una paleta con operaciones y escritura matemática.

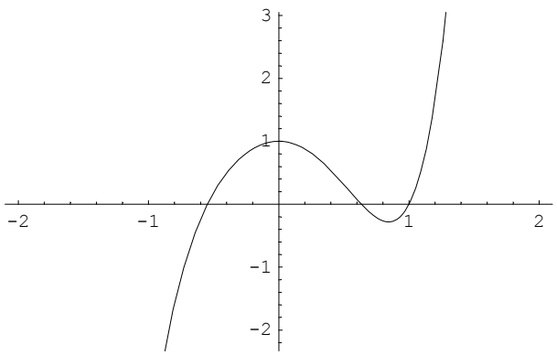
2. Gráfico de funciones de una variable

Comando Plot

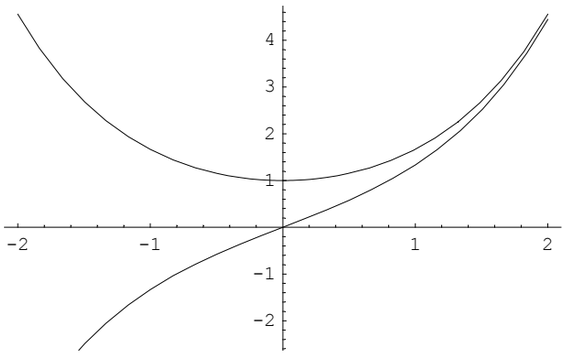
El comando más utilizado para graficar en el plano es Plot. La sintaxis básica es:

Plot[f,{x, x min, x max}] : construye el gráfico de f en el intervalo [x min, x max].
Plot[{f_1, f_2 , ..., f_j},{x, x min, x max}] : construye, en un mismo par de ejes, los gráficos de las funciones f_1, f_2 , ..., f_j en el intervalo [x min, x max].

```
In[8]:= Plot[2 x^5 - 3 x^2 + 1, {x, -2, 2}];
```



```
In[9]:= Plot[{(3^x + 3^-x)/2, (3^x - 3^-x)/2}, {x, -2, 2}];
```



Opciones básicas del comando Plot

Las opciones de este comando se escriben después del argumento del mismo y tienen la finalidad de mejorar el aspecto del gráfico. La forma general es:

Plot[f,{x, x min, x max}, Opción→Valor]

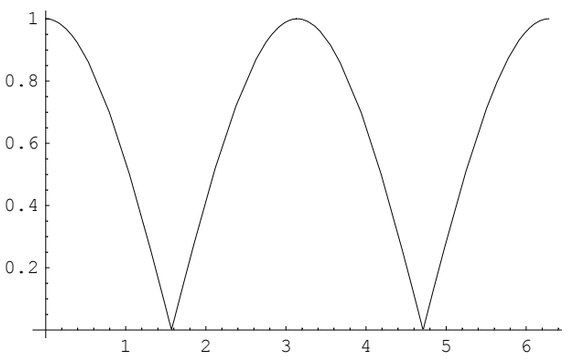
Las opciones más utilizadas son:

AspectRatio

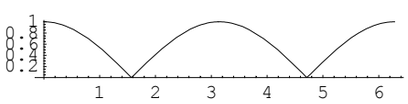
En general, *Mathematica* coloca distintas escalas en los ejes, para mostrar la parte más significativa del gráfico. Pero esto a veces puede llevar a confusión.

AspectRatio→Automatic : grafica utilizando la misma escala en ambos ejes

```
In[10]:= Plot[Abs[Cos[x]], {x, 0, 2 π}];
```



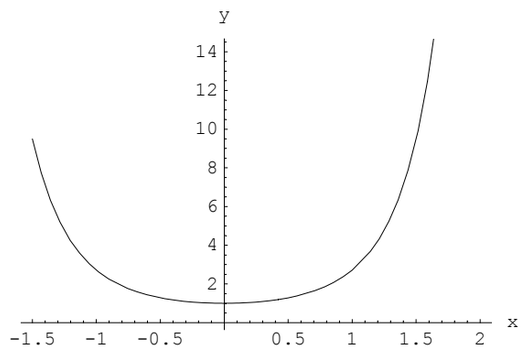

```
In[11]:= Plot[Abs[Cos[x]], {x, 0, 2 π}, AspectRatio -> Automatic];
```



AxesLabel

La opción AxesLabel→{"nombre sobre el eje x" ," nombre sobre el eje y"}, permite colocar etiquetas en los ejes.

```
In[12]:= Plot[E^x^2, {x, -1.5, 2}, AxesLabel -> {x, y}];
```

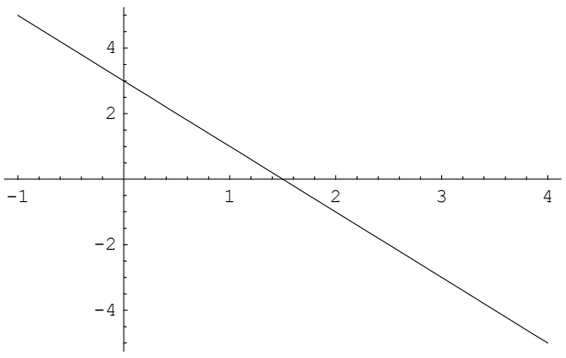


■ PlotStyle

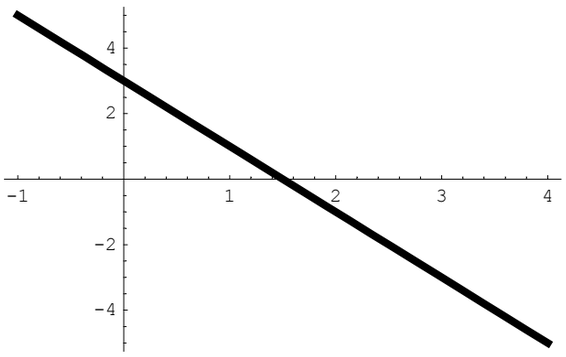
La opción PlotStyle controla muchos detalles de la apariencia final de un gráfico.

• PlotStyle→{Thickness[t]} : da el grosor de la curva. El argumento "t" representa la razón entre el ancho de la línea y el ancho total del gráfico.

```
In[13]:= Plot[-2 x + 3, {x, -1, 4}];
```



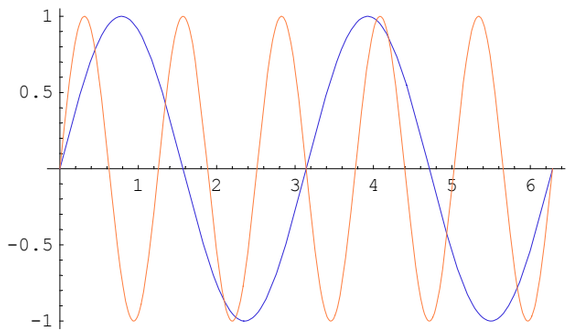

```
In[14]:= Plot[-2 x + 3, {x, -1, 4}, PlotStyle -> Thickness[0.014]];
```



• PlotStyle→{RGBColor[a,b,c]} : permite dar color al gráfico.

Se puede agregar el color deseado, seleccionándolo directamente de una paleta; para lograrlo, dentro del comando Plot posicionamos el cursor inmediatamente después de " PlotStyle→ " ; vamos a Input (de la barra de herramientas) y seleccionamos " Color Selector ", aparece una ventana que tiene una paleta de colores. Una vez seleccionado con el mouse un color, se acepta y quedan incorporados dentro de Plot los parámetros del color elegido.

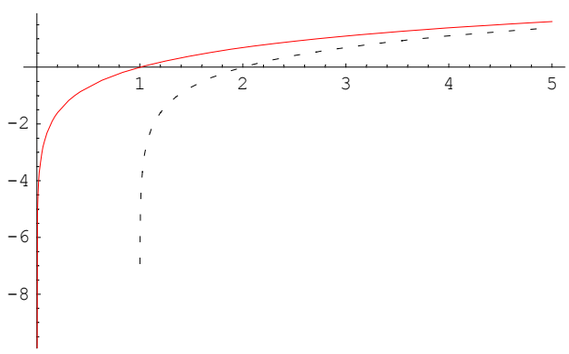
```
In[15]:= Plot[{Sin[2 x], Sin[5 x]}, {x, 0, 2 π},
           PlotStyle -> {RGBColor[0.191409, 0.156252, 0.839857],
         RGBColor[0.996109, 0.500008, 0.250004]}];
```



• PlotStyle→{Dashing[m,n]} : con esta opción se puede efectuar el gráfico en línea de puntos; su grosor depende de m y n.

```
In[16]:= Plot[{Log[x - 1], Log[x]}, {x, 0, 5},
           PlotStyle -> {Dashing[{0.01, 0.03}], RGBColor[1, 0, 0]}];
```

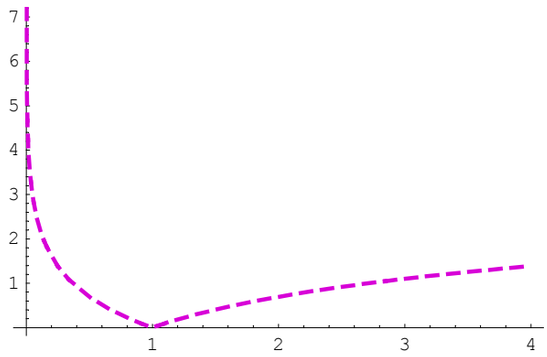
```
Plot::plnr : Log[x - 1] is not a machine-size real number at x = 2.0833333333333333`*^-7. More…

Plot::plnr : Log[x - 1] is not a machine-size real number at x = 0.20283495786457897`. More…

Plot::plnr : Log[x - 1] is not a machine-size real number at x = 0.4240439992968684`. More…

General::stop : Further output of Plot::plnr will be suppressed during this calculation. More…
```



Observemos que a pesar que Mathematica grafica las funciones pedidas, avisa que hay valores donde se quiere dibujar que no forman parte del dominio de la primer función.

Podemos combinar la opción de color, línea de puntos y grosor.

```
In[17]:= Plot[ Abs[Log[x]], {x, 0, 4}, PlotStyle ->
           {Thickness[0.008], Dashing[{0.02, 0.02}], RGBColor[0.839857, 0, 0.839857]}];
```



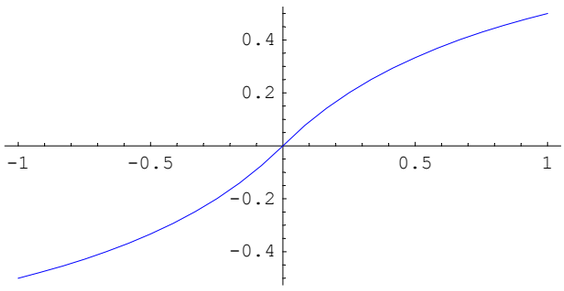
■ Comando Show

Otro comando muy útil es el Show. Veamos las posibilidades que brinda.

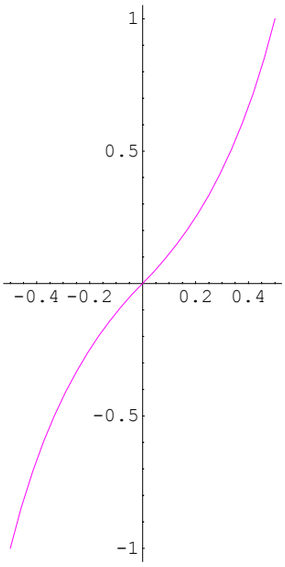
Show[gráfico 1, gráfico 2, ... ,gráfico n] representa en un mismo sistema de ejes, los gráficos 1, 2, ... , n, ya existentes.

Graficamos las funciones $f(x) = \frac{x}{1+|x|}$ y $g(y) = \frac{y}{1-|y|}$

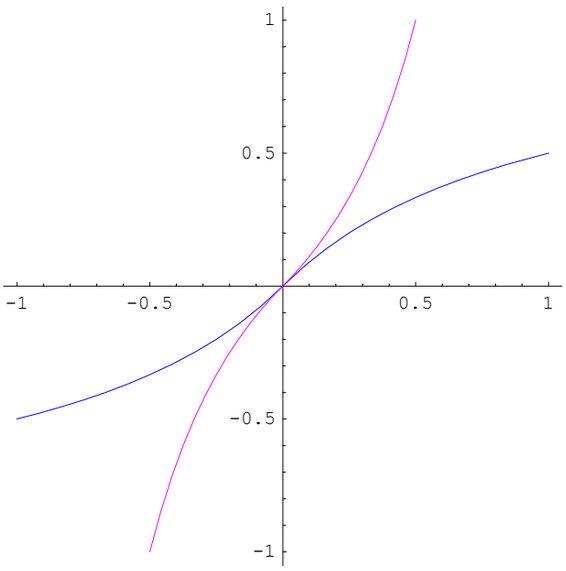
```
In[18]:= h1 = Plot[ x / (1 + Abs[x]), {x, -1, 1},
           AspectRatio → Automatic, PlotStyle → RGBColor[0, 0, 0.996109]];
```




```
In[19]:= h2 = Plot[ x / (1 - Abs[x]), {x, -0.5, 0.5},
           AspectRatio → Automatic, PlotStyle → RGBColor[0.996109, 0, 0.996109]];
```

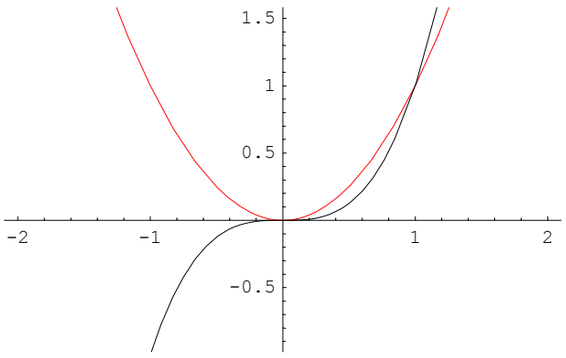


In[20]:=
```
Show[h1, h2];
```



La opción DisplayFunction→Identity, no muestre el gráfico en pantalla, aunque internamente la construye.
La opción DisplayFunction→$DisplayFunction, deja ver el gráfico que mantuvimos oculto con la opción anterior.

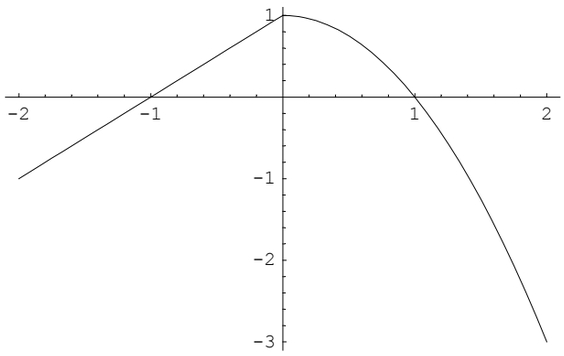
In[21]:=
```
gr1 = Plot[x^2, {x, -2, 2}, PlotStyle -> RGBColor[1, 0, 0], DisplayFunction -> Identity] ;
gr2 = Plot[x^3, {x, -2, 2}, DisplayFunction -> Identity] ;
Show[gr1, gr2, DisplayFunction -> $DisplayFunction] ;
```



El comando Show es también una alternativa sencilla para graficar funciones partidas.

Grafiquemos: $f(x) = \begin{cases} x+1 & \text{si } x \le 0 \\ -x^2 & \text{si } x > 0 \end{cases}$

```
In[22]:= k1 = Plot[x + 1, {x, -2, 0}, DisplayFunction -> Identity] ;
         k2 = Plot[-x^2 + 1, {x, 0, 2}, DisplayFunction -> Identity] ;
         Show[k1, k2, DisplayFunction -> $DisplayFunction] ;
```



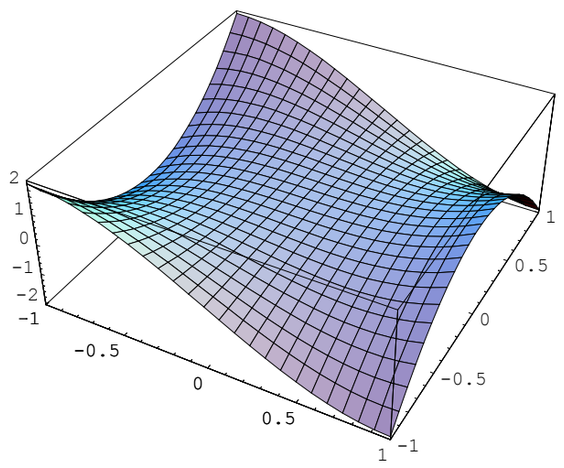
3. Gráfico de funciones de dos variables

■ Comando Plot3D

Plot3D[f(x,y), {x, x min, x max},{y, y min, y max}] : grafica z = f(x,y)

Este comando figura en el **Palette BasicCalculations - Graphics**.

```
In[23]:= Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}];
```



```
In[24]:= Plot3D[Sin[2 x + Cos[y]], {x, -π, π}, {y, -2 π, 2 π}];
```

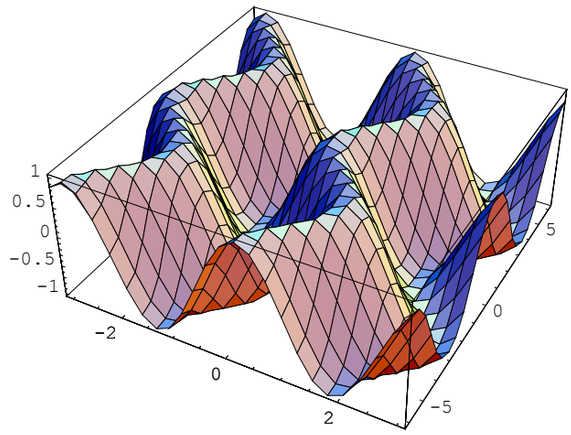


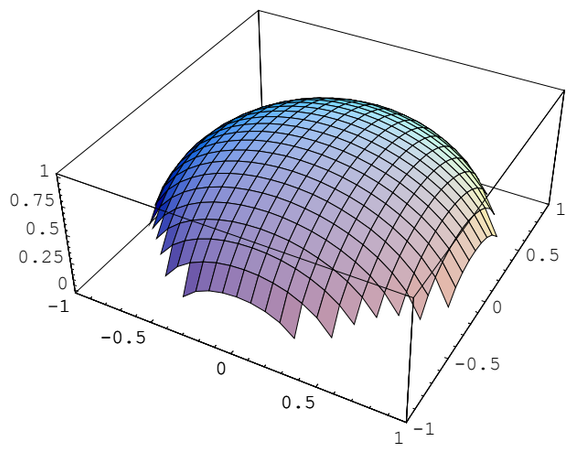
Gráfico y dominio de la función

De la misma forma que ocurre con el gráfico de funciones en el plano, cuando el comando Plot3D encuentra valores de las variables fuera del dominio natural de la función, estos son simplemente ignorados, pero el programa avisa que la función toma valores no reales en algunos puntos; de ahí que no pueda graficarlos.

Por ejemplo, se quiere graficar $f(x, y) = \sqrt{1 - x^2 - y^2}$. Su dominio natural es $x^2 + y^2 \leq 1$. Luego, dicho dominio está contenido en el rectángulo $-1 \leq x \leq 1$, $-1 \leq y \leq 1$, pero hay infinitos puntos de este conjunto que no pertenecen al dominio.

```
In[25]:= Plot3D[Sqrt[1 - x^2 - y^2], {x, -1, 1}, {y, -1, 1}];
```

```
Plot3D::gval : Function value 0. + 1. i at grid point xi = 1, yi = 1 is not a real number. More…

Plot3D::gval : Function value 0. + 0.916667 i at grid point xi = 1, yi = 2 is not a real number. More…

Plot3D::gval : Function value 0. + 0.833333 i at grid point xi = 1, yi = 3 is not a real number. More…

General::stop : Further output of Plot3D::gval will be suppressed during this calculation. More…
```



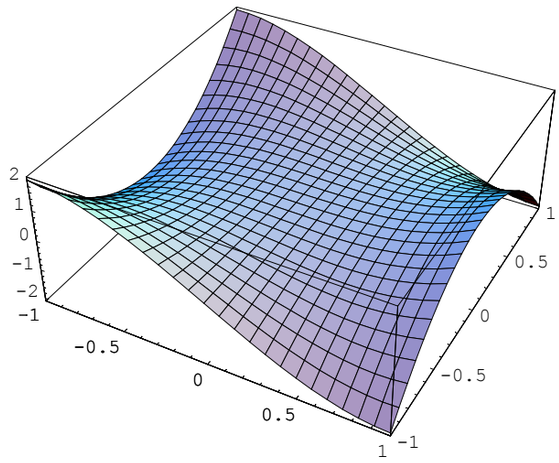
■ Opciones del comando Plot3D

■ Axes

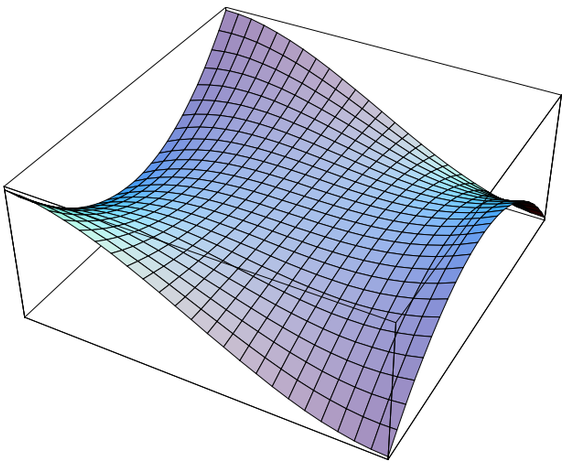
Los gráficos tridimensionales, por defecto, siempre incluye la escala en los ejes.

Axes→False elimina la escala en los ejes.

```
In[26]:= Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}];
```



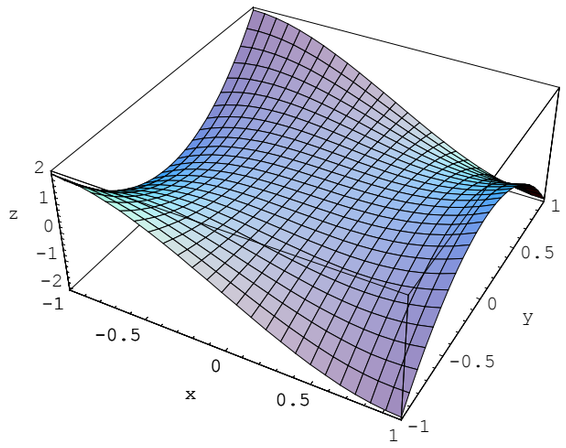

```
In[27]:= Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}, Axes -> False];
```



■ AxesLabel

AxesLabel→{n1, n2, n3} rotula los ejes x, y, z respectivamente.

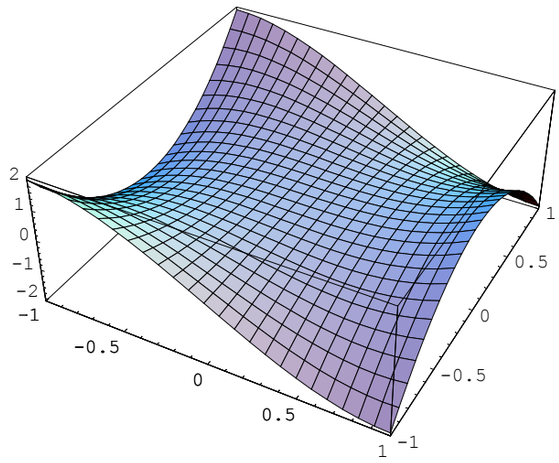
```
In[28]:= Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}, AxesLabel -> {x, y, z}];
```



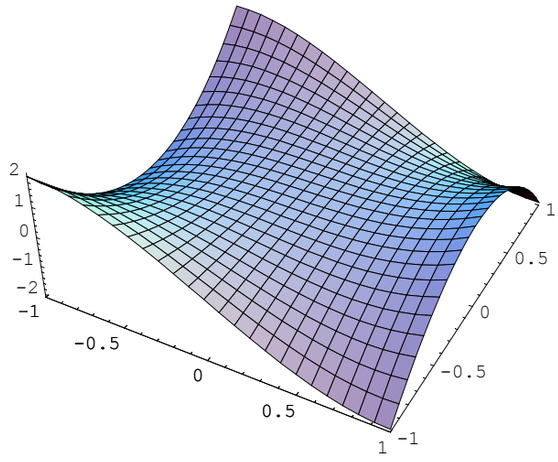
■ Boxed

Boxed→False elimina el paralelepípedo, que por defecto, limita todo gráfico tridimensional

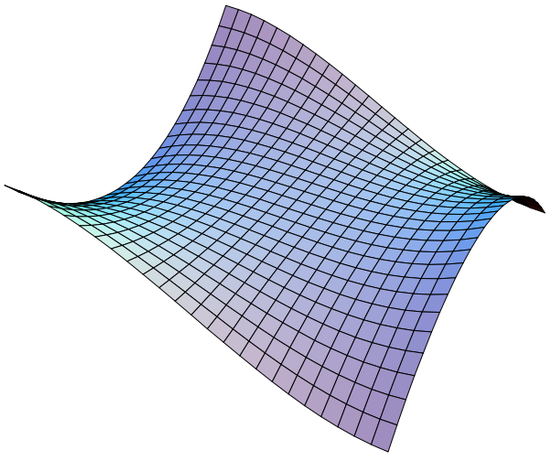
In[29]:= **Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}];**



In[30]:= **Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}, Boxed -> False];**



In[31]:= **Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}, Boxed -> False, Axes -> False];**

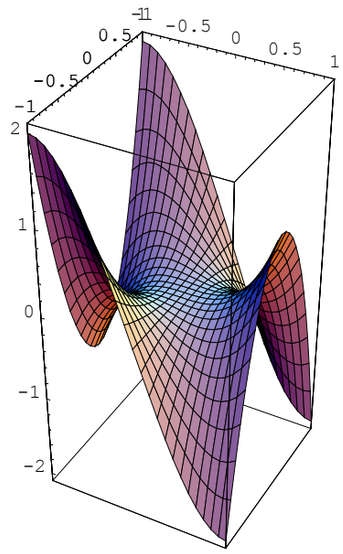


■ BoxRatios

Así como lo mencionamos para funciones en el plano, también en el espacio, el gráfico que *Mathematica* produce, tiene por defecto diferentes escalas en los tres ejes.

BoxRatios→Automatic muestra el gráfico tal como es en la realidad.

```
In[32]:= Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}, BoxRatios -> Automatic];
```



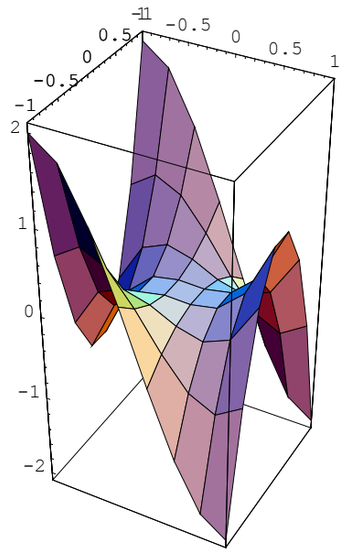
■ PlotPoints

Así como el gráfico dado por *Mathematica* de una función de una variable, es una sucesión de segmentos unidos de manera tal que se visualiza como una curva suave, la superficie que se obtiene al graficar una función de dos variables, es una aproximación del gráfico verdadero.

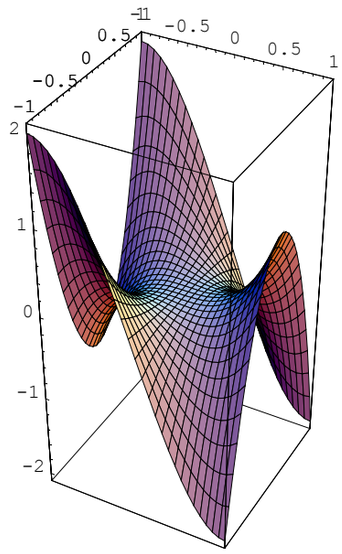
PlotPoints→n hace que el programa evalúe la función f(x,y) en n^2 puntos donde realizará el gráfico

Cuanto más grande sea el valor de n, se obtendrá una versión más suavizada del gráfico. El valor de n por defecto es 15. Si el valor de n es muy grande tarda más tiempo en producir el gráfico.

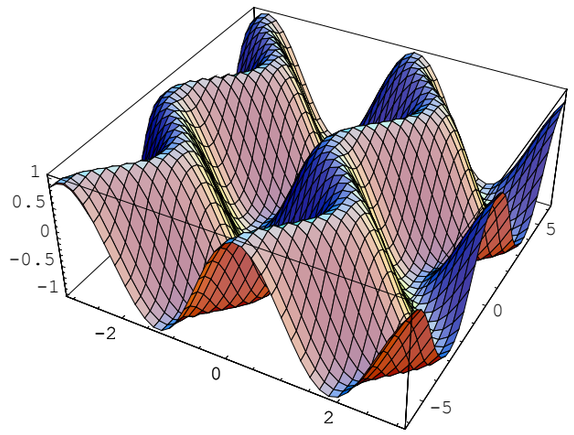
```
In[33]:= Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}, BoxRatios -> Automatic, PlotPoints -> 8];
```



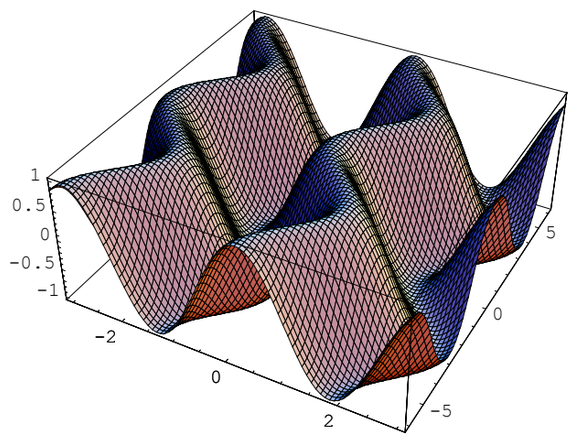
```
In[34]:= Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}, BoxRatios -> Automatic, PlotPoints -> 30];
```




```
In[35]:= Plot3D[Sin[2 x + Cos[y]], {x, -π, π}, {y, -2 π, 2 π}, PlotPoints → 40];
```




```
In[36]:= Plot3D[Sin[2 x + Cos[y]], {x, -π, π}, {y, -2 π, 2 π}, PlotPoints → 80];
```

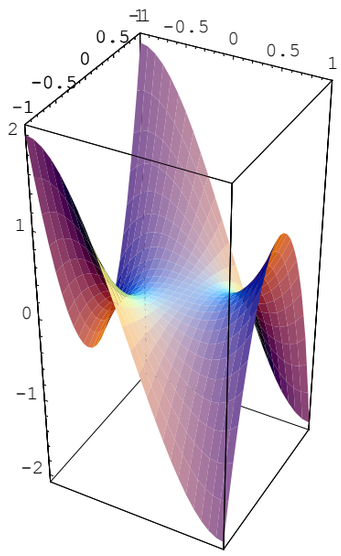


■ Mesh

El programa, por defecto, efectúa el gráfico con una malla que cubre la superficie, con la finalidad de ayudar a la visualización.

Mesh→False eliminar la malla del gráfico.

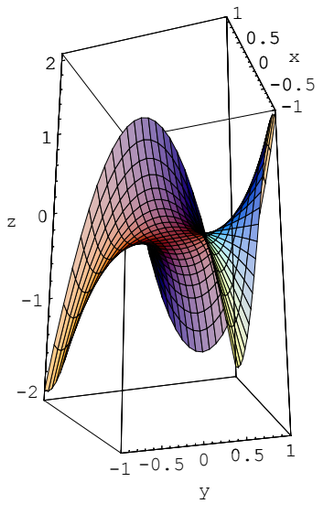
```
In[37]:= Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}, BoxRatios -> Automatic, Mesh -> False];
```



■ ViewPoint

Esta opción permite elegir el punto del espacio desde el que se quiere ver el gráfico.
Existe la posibilidad de elegir interactivamente las coordenadas de dicho punto de vista. Para lograrlo, posicionamos el cursor en la línea del comando Plot3D donde deseamos insertar la opción ViewPoint. Vamos a Input (de la barra de herramientas) y seleccionamos " 3D ViewPoint Selector ". Aparece una ventana, que tiene el dibujo de los ejes en la posición actual. Para modificar la posición de los ejes, llevamos el mouse a la derecha y a la parte de abajo del dibujo y arrastramos el mouse con el botón izquierdo apretado hasta llegar a la posición deseada. Hacemos un click en 'Paste', y la opción ViewPoint {a,b,c} será pegada en el comando Plot3D en donde habíamos dejado el cursor.

```
In[38]:= Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}, BoxRatios -> Automatic,
           ViewPoint -> {2.560, 0.940, -1.370}, AxesLabel -> {x, y, z}];
```

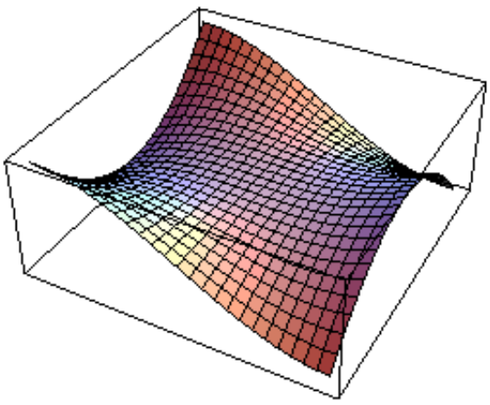


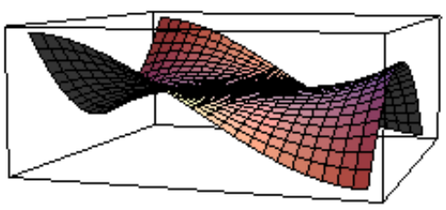
■ RealTime (para las versiones 4 y 5, NO funciona en la versión 3)

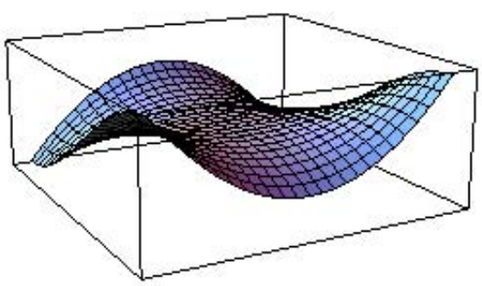
<<RealTime3D` Plot3D[ ] permite mover el gráfico, moviendo el mouse, mientras se mantiene oprimido el botón izquierdo. Tambien puede cargarse la opción sola, y afectará a todos los gráficos de ahí en mas.

```
In[39]:= << RealTime3D`
```

In[40]:=
```
Plot3D[x^3 - 3 x y^2, {x, -1, 1}, {y, -1, 1}];
```







Se muestran 3 vistas distintas del gráfico, moviéndolo con el mouse.

Observación:
- Cuando se utiliza esta opción, los ejes dejan de estar rotulados.
- En una sesión de trabajo, si en uno de los gráficos se utiliza esta opción, de ahí en adelante, todos los gráficos en 3D estarán afectados por la opción (sin necesidad de indicarlo).
  Para desactivar esta opción hay que colocar <<Default3D`
- En los gráficos que incluyen puntos que no están en el dominio, el gráfico aparece totalmente distorsionado; no dando ningún indicio de como es en realidad. Se sugiere no utilizarlo.

```
In[41]:= << Default3D`
```

4. Gráfico de curvas

En el plano

Ya hemos visto como graficar funciones de una variable utilizando el comando Plot. En muchos casos, las curvas de interés no están definidas en forma explícita.

Puntos, líneas y polígonos

Graphics[{directivas , primitivas}, opciones] representa puntos, líneas y polígonos. Para ver el gráfico debemos combinarlo con el comando Show.

Directivas:
Indica el aspecto del gráfico.
Por ejemplo: color (RGB), grosor (PointSize, para puntos y Tickness, para líneas) y línea de puntos (Dashing).
Primitivas:
- Point[{x,y}] (corresponde a un punto)
- Line[{{x_1 , y_1}, ..., {x_n , y_n}}] (corresponde a una línea, que pasa por los puntos indicados)
- Polygon [{{x_1 , y_1}, ..., {x_n , y_n}}] (corresponde a un polígono relleno cuyos vértices son los puntos de la lista}

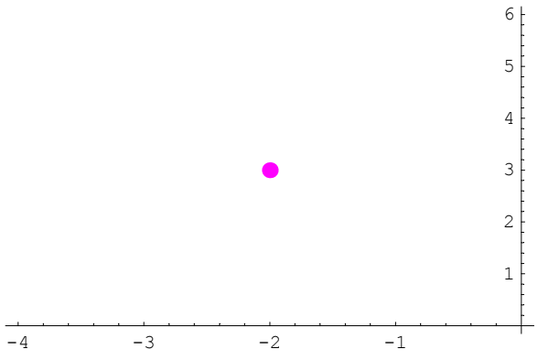
Opciones:
Corresponden a la vista general del dibujo. Por ejemplo: AspectRatio, Axes y AxesOrigin.

1. Graficamos un punto

```
In[42]:= p1 = Graphics[Point[{-2, 3}]] ; Show[p1] ;
```

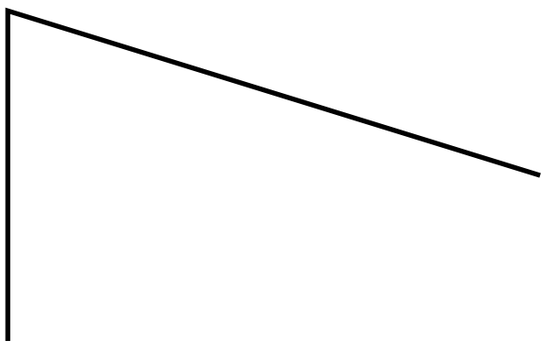
Para este mismo gráfico, le damos grosor al punto, color y pedimos que agregue los ejes.

```
In[43]:= p2 = Graphics[{PointSize[0.03], RGBColor[0.996109, 0, 0.996109], Point[{-2, 3}]},
            Axes -> Automatic] ; Show[p2] ;
```

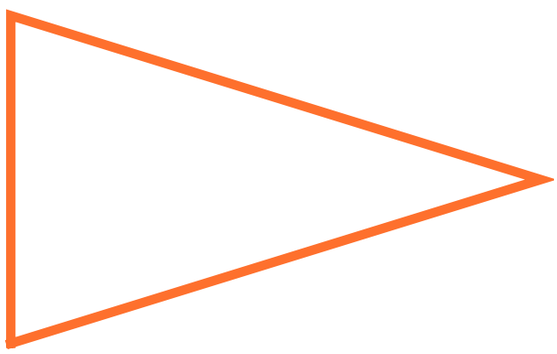


2. Graficamos una línea y un polígono.

```
In[44]:= p3 = Graphics[{Thickness[0.009], Line[{{1, 2}, {1, 4}, {2, 3}}]}] ; Show[p3] ;
```

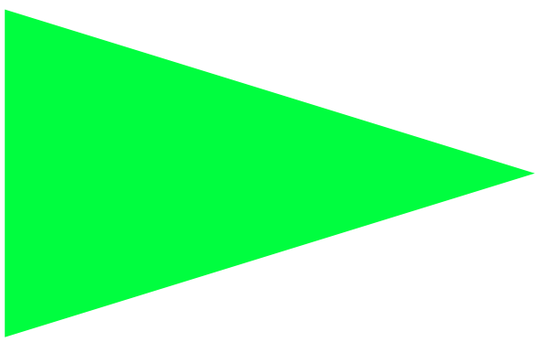


```
In[45]:= p4 = Graphics[{Thickness[0.017], RGBColor[0.996109, 0.441413, 0.175784],
             Line[{{1, 2}, {1, 4}, {2, 3}, {1, 2}}]}] ; Show[p4] ;
```

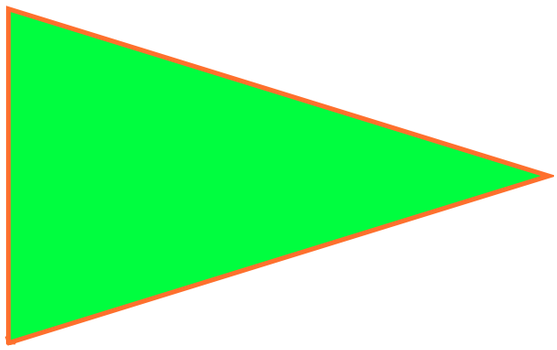


3. Graficamos un polígono relleno.

```
In[46]:= p5 = Graphics[{RGBColor[0, 0.996109, 0.250004], Polygon[{{1, 2}, {1, 4}, {2, 3}}]}] ;
          Show[p5] ;
```



```
In[47]:= Show[p4, p5];
```



■ Curvas dadas en coordenadas paramétricas

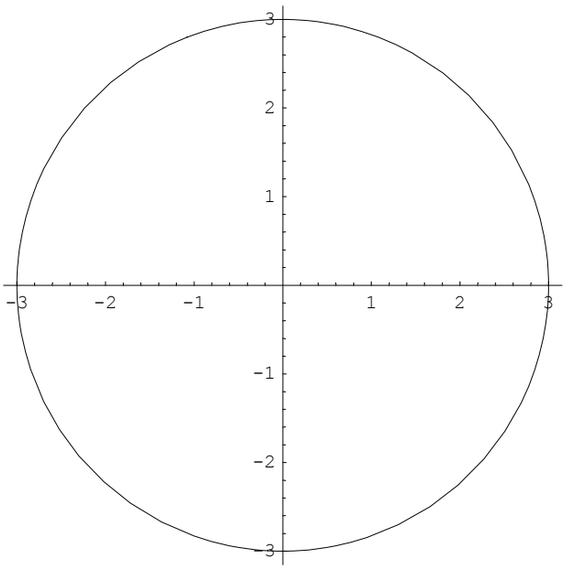
Los comandos para realizar gráficos de curvas en coordenadas paramétricas son:

ParametricPlot[{x(t),y(t)},{t, t min, t max}]: realiza el gráfico de una curva en coordenadas paramétricas.
ParametricPlot[{x(t),y(t)},{t, t min, t max},AspectRatio->Automatic] : considera la misma unidad en ambos ejes.
ParametricPlot[{{x_1(t),y_1(t)},{x_2(t),y_2(t)},.....},{t, t min, t max}] : representa varias curvas en un mismo gráfico.

1. Circunferencia

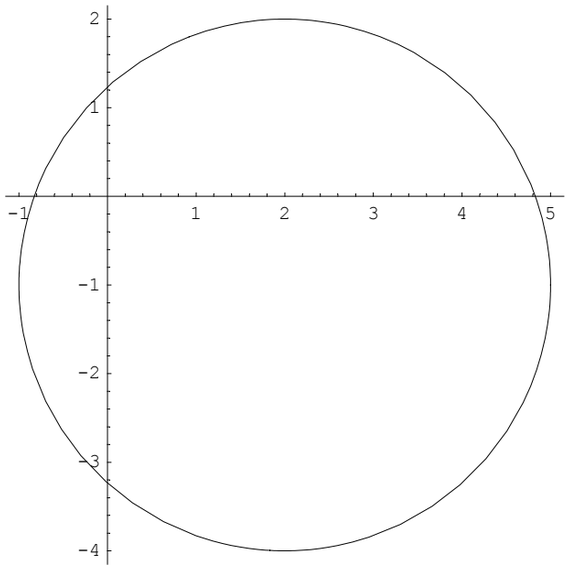
Graficamos la circunferencia $x^2 + y^2 = 9$
La ecuación paramétrica es : $x = 3\ \cos t\ ,\ \ y = 3\ \text{sen}\, t\ \ \ \text{con}\ \ t\ \epsilon [0\ ,\ 2\,\pi]$

```
In[48]:= c0 = ParametricPlot[{3 Cos[t], 3 Sin[t]}, {t, 0, 2 π},
             AspectRatio -> Automatic];
```



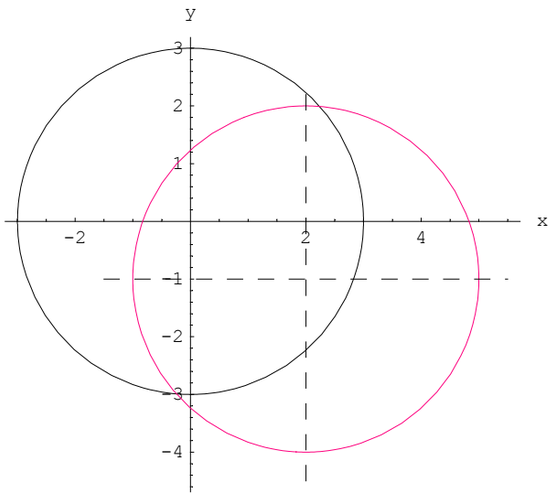
La opción AspectRatio → Automatic, realiza el gráfico tal cual es
Graficamos la circunferencia $(x - 2)^2 + (y + 1)^2 = 9$
La ecuación paramétrica es : $x = 2 + 3\ \cos t\ ,\ \ y = -1 + 3\ \text{sen}\, t\ \ \ \text{con}\ \ t\ \epsilon [0\ ,\ 2\,\pi]$

```
In[49]:= c1 = ParametricPlot[{2 + 3 Cos[t], -1 + 3 Sin[t]}, {t, 0, 2 π},
             AspectRatio -> Automatic];
```



Efectuamos las dos circunferencias en un mismo gráfico, dibujando en línea de puntos los ejes de la circunferencia de centro (2, - 1).

```
In[50]:= c00 = ParametricPlot[{3 Cos[t], 3 Sin[t]}, {t, 0, 2 π},
             AspectRatio -> Automatic, DisplayFunction → Identity];
           c11 = ParametricPlot[{2 + 3 Cos[t], -1 + 3 Sin[t]}, {t, 0, 2 π},
             AspectRatio -> Automatic,
             PlotStyle → RGBColor[0.996109, 0, 0.500008], DisplayFunction → Identity];
           r1 = ParametricPlot[{2, t}, {t, -4.5, 2.3},
             AspectRatio -> Automatic,
             PlotStyle -> Dashing[{0.03}], DisplayFunction → Identity];
           r2 = ParametricPlot[{t, -1}, {t, -1.5, 5.5},
             AspectRatio -> Automatic,
             PlotStyle -> Dashing[{0.03}], DisplayFunction → Identity];
           Show[c00, c11, r1, r2, AxesLabel -> {x, y}, DisplayFunction → $DisplayFunction];
```

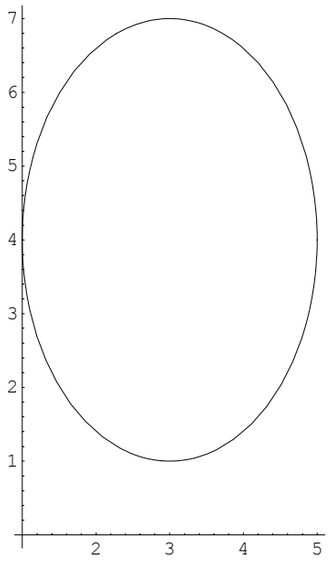


2. Elipse

$$\frac{(x-3)^2}{4} + \frac{(y-4)^2}{9} = 1$$

La ecuación paramétrica es: $x = 2\cos t + 3$, $y = 3\,\text{sen}\,t + 4$ con $t \in [0, 2\pi]$.

```
In[55]:= ParametricPlot[{ 2 Cos[t] + 3, 3 Sin[t] + 4}, {t, 0, 2 π}, AspectRatio -> Automatic];
```

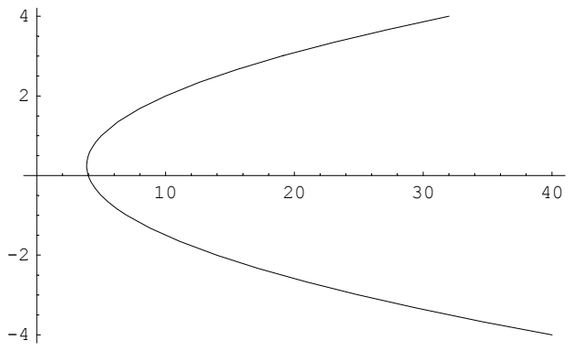


3. Parábola

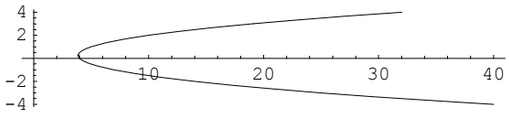
$x = 2\,y^2 - y + 4$.

La ecuación paramétrica es: $x = 2\,t^2 - t + 4$, $y = t$ con t un número real.

```
In[56]:= ParametricPlot[{2 t^2 - t + 4, t}, {t, -4, 4}];
```




```
In[57]:= ParametricPlot[{2 t^2 - t + 4, t}, {t, -4, 4}, AspectRatio -> Automatic];
```



4. Hipérbola

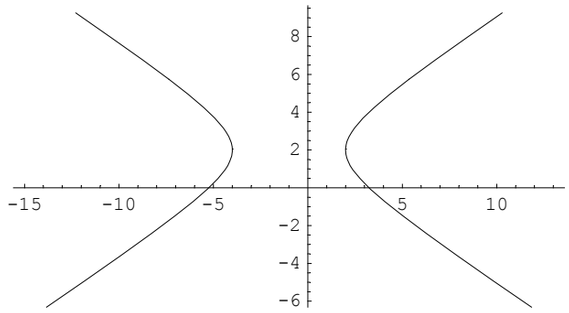
$$\frac{(x+1)^2}{9} - \frac{(y-2)^2}{4} = 1$$

La parametrización debe hacerse para cada rama:

una rama: $x = -1 + 3\ \mathrm{ch}\ t$, $y = 2 + 2\ \mathrm{sh}\ t$

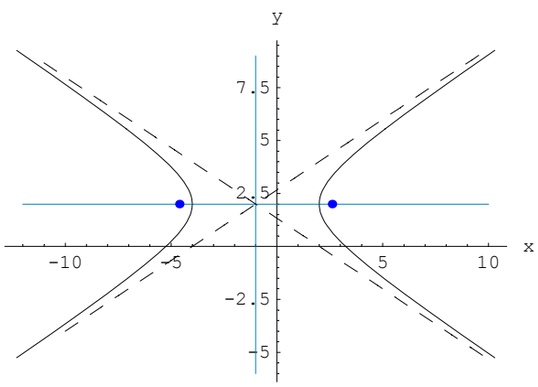
la otra rama: $x = -1 - 3\,\mathrm{ch}\,t$, $y = 2 + 2\,\mathrm{sh}\,t$ con t un número real

```
In[58]:= ParametricPlot[{{-1 + 3 Cosh[t], 2 + 2 Sinh[t]}, {-1 - 3 Cosh[t], 2 + 2 Sinh[t]}},
           {t, -3, 2}, AspectRatio -> Automatic];
```



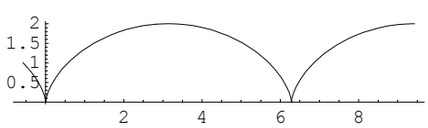
Realicemos un gráfico que contenga: los nuevos ejes (x = -1, y =2), las asíntotas ($y = \frac{2}{3}(x+1)+2$; y $= -\frac{2}{3}(x+1)+2$) y los focos
($(-1+\sqrt{13}, 2)$, $(-1-\sqrt{13}, 2)$).

```
In[59]:= hip = ParametricPlot[{{-1 + 3 Cosh[t], 2 + 2 Sinh[t]}, {-1 - 3 Cosh[t], 2 + 2 Sinh[t]}},
            {t, -2, 2}, DisplayFunction → Identity];
         eje1 = ParametricPlot[{-1, t}, {t, -6, 9}, PlotStyle -> RGBColor[0, 0.500008, 0.750011],
            DisplayFunction → Identity];
         eje2 = ParametricPlot[{t, 2}, {t, -12, 10}, PlotStyle -> RGBColor[0, 0.500008, 0.750011],
            DisplayFunction → Identity];
         focos = ListPlot[{{-1 + √13, 2}, {-1 - √13, 2}}, PlotStyle ->
             {PointSize[0.017], RGBColor[0, 0, 0.996109]}, DisplayFunction → Identity];
         asint1 = Plot[2/3 (x + 1) + 2, {x, -10, 10}, PlotStyle -> Dashing[{0.03}],
           DisplayFunction → Identity]; asint2 = Plot[-2/3 (x + 1) + 2, {x, -11, 10},
           PlotStyle -> Dashing[{0.03}], DisplayFunction → Identity];
         Show[{hip, eje1, eje2, focos, asint1, asint2}, DisplayFunction → $DisplayFunction,
          AxesLabel -> {x, y}, AspectRatio -> Automatic];
```



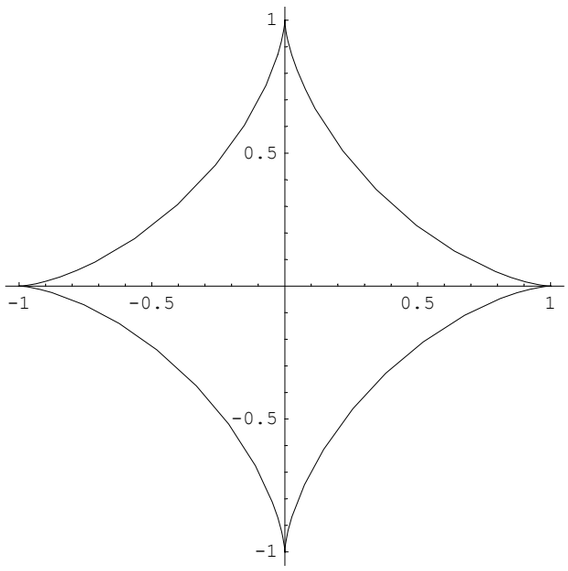
5. Cicloide x = a (1- sen t) , y = a (1- cos t) , a > 0, t un ángulo
   Para a = 1.

```
In[64]:= ParametricPlot[{t - Sin[t], 1 - Cos[t]}, {t, -π/2, 3π},
           AspectRatio -> Automatic];
```



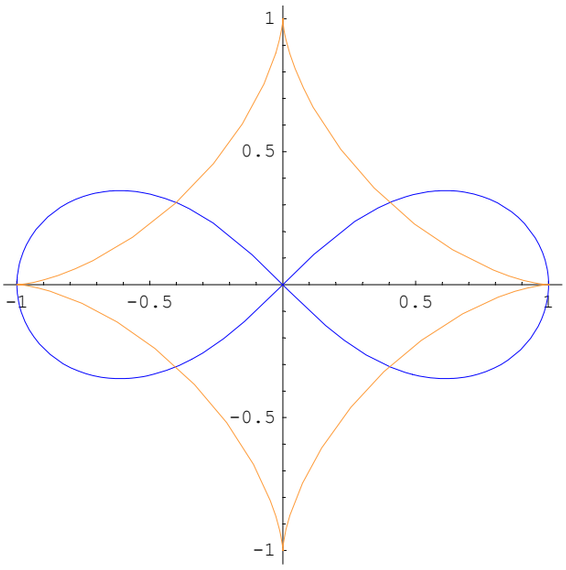
6. La ecuación : $x = a\cos^3 t$, $y = a\,\mathrm{sen}^3 t$, $a > 0$, $0 \le t \le 2\pi$, representa una curva llamada astroide
   Para a = 1:

```
In[65]:= ParametricPlot[{(Cos[t])^3 , (Sin[t])^3}, {t, 0, 2 π}, AspectRatio -> Automatic];
```



7. Realizamos en un mismo gráfico la Lemniscata de ecuación x = $\frac{\text{Cos[t]}}{1+(\text{Sin[t]})^2}$, y = $\frac{\text{Sin[t] Cos[t]}}{1+(\text{Sin[t]})^2}$ con $0 \leq t \leq 2\pi$; y la astroide, para a = 1.

```
In[66]:= ParametricPlot[{{ Cos[t] / (1 + (Sin[t])^2), Sin[t] Cos[t] / (1 + (Sin[t])^2) },
           {(Cos[t])^3, (Sin[t])^3}},
           {t, 0, 2 π},
         PlotStyle -> {RGBColor[0, 0, 0.996109], RGBColor[0.996109, 0.605478, 0.214847]},
         AspectRatio -> Automatic];
```



■ Paquetes

Mathematica es una aplicación escrita en lenguaje C. Sin embargo, posee un lenguaje con el que se pueden crear a su vez nuevos procedimientos o programas.
Un archivo que contenga definiciones escritas en el propio lenguaje de Mathematica se denomina paquete. Para utilizar las funciones contenidas en él, es necesario previamente cargarlo, lo hacemos del siguiente modo:

<<Contexto`Nombre Del Paquete` donde Contexto es el nombre del directorio o carpeta que contiene al paquete.

Para evitar problemas con los paquetes, **es recomendable cargarlos al comienzo de la sesión**, ya que, pueden producirse conflictos entre las funciones definidas por el usuario a lo largo de la misma y las que define el paquete al ser cargado en memoria.
Para investigar cuales son los paquetes y sus contenidos, vamos al Help, allí marcamos Add-ons y luego StandardPackages; aparecen varios títulos como Algebra, Calculus,.....; esos son los distintos directorios.

■ Curvas dadas en coordenadas polares

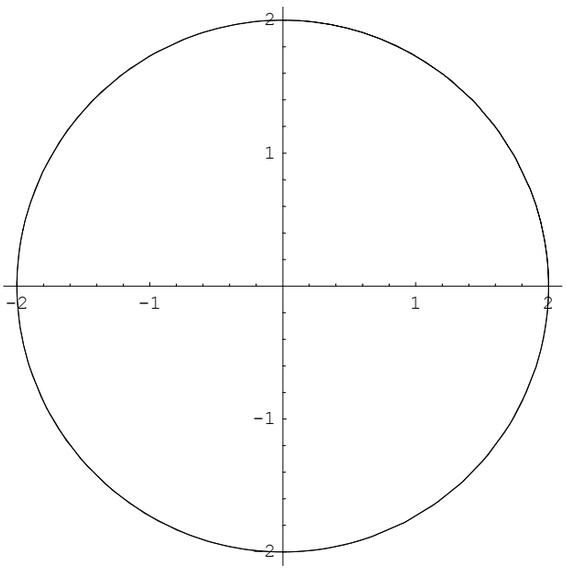
Cuando la curva está dada en coordenadas polares, hay que cargar el paquete Graphics`Graphics`. Los comandos son:

PolarPlot[r(t), {t, t min, t max}] : genera un gráfico polar de radio r, en función del ángulo t , para valores de t variando entre t min y t max.
PolarPlot[{r1(t), r2(t),....},{t, t min, t max}] : representa en un mismo gráfico varias curvas, para el ángulo t en el intervalo de t min hasta t max.

1. Dibujamos la circunferencia de radio 2.

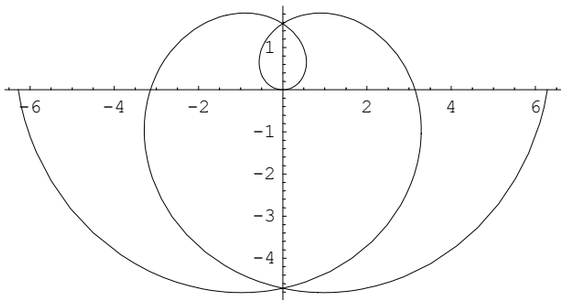
```
In[67]:= << Graphics`Graphics`
```

```
In[68]:= PolarPlot[2, {t, -2 π, 2 π}];
```



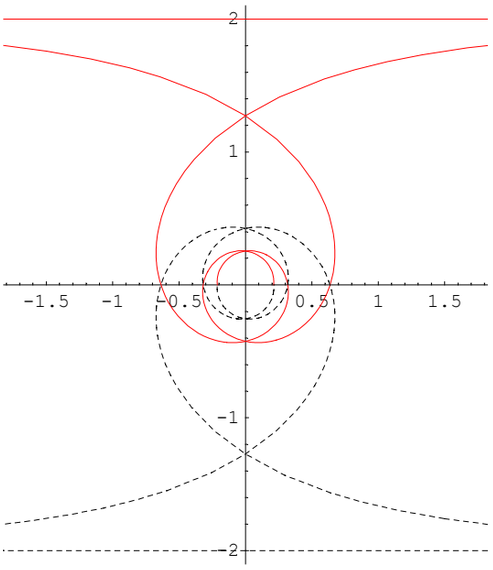
2. Dibujamos la espiral de Arquímedes de ecuación r = t.

```
In[69]:= PolarPlot[t, {t, -2 π, 2 π}];
```



3. Dada la ecuación $r = \frac{a}{t}$ de la espiral hiperbólica, realizamos en un mismo gráfico las espirales para a = 2 y a = - 2.

```
In[70]:= PolarPlot[{2 / t, -2 / t}, {t, -3 Pi, 3 Pi},
           PlotStyle -> {RGBColor[1, 0, 0], Dashing[{0.01}]}];
```



Observemos que dibuja también la asíntota, aunque no forma parte de la curva.

■ Curvas definidas implícitamente

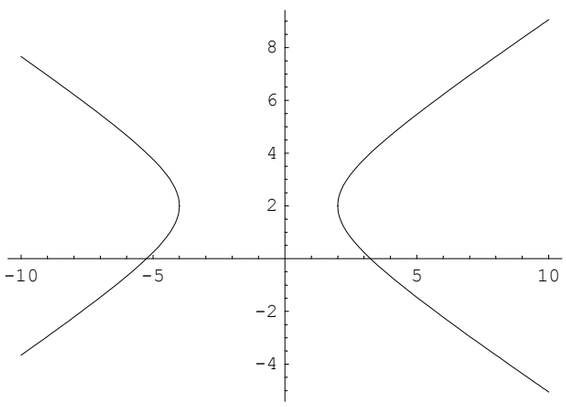
Cuando la curva está dada en forma implícita, hay que cargar el paquete Graphics`ImplicitPlot`. Los comandos son:

ImplicitPlot[ecuación, {x, x min, x max}] : dibuja la curva dada por la 'ecuación', para x variando en un intervalo.
ImplicitPlot[ecuación, {x, x min, x max}, {y, y min, y max}]: dibuja la curva, en los intervalos de variación de x e y.
ImplicitPlot[{ecuación 1,ecuación 2,.....}, rangos, opciones]: dibuja las curvas dadas por las distintas ecuaciones, en los rangos especificados para las variables y con las opciones indicadas.

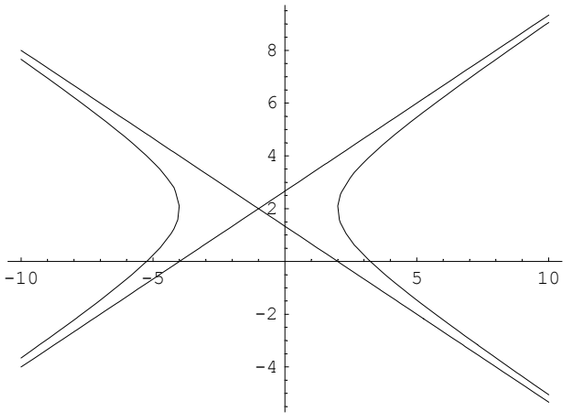
1. Graficamos la hipérbola $\frac{(x+1)^2}{9} - \frac{(y-2)^2}{4} = 1$

```
In[71]:= << Graphics`ImplicitPlot`
```

```
In[72]:= ImplicitPlot[(x + 1)^2 / 9 - (y - 2)^2 / 4 == 1, {x, -10, 10}];
```

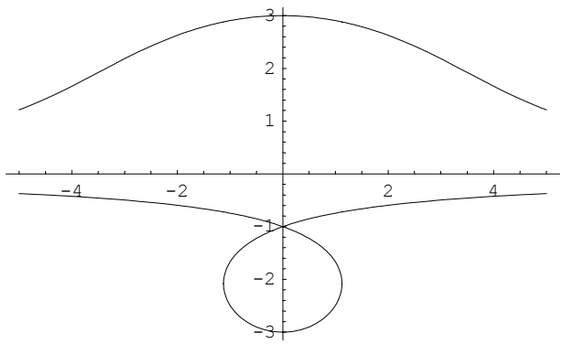


```
In[73]:= ImplicitPlot[{(x + 1)^2/9 - (y - 2)^2/4 == 1, 2/3 (x + 1) + 2 == y, -2/3 (x + 1) + 2 == y},
           {x, -10, 10}, {y, -10, 10}];
```



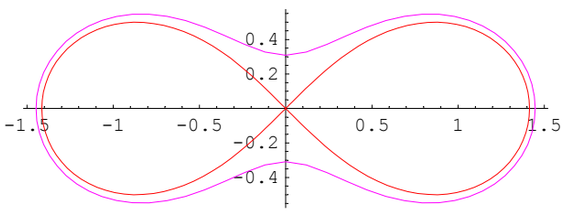
2. Graficamos la ecuación $x^2 y^2 = (y + 1)^2 (9 - y^2)$.

```
In[74]:= ImplicitPlot[x^2 y^2 - (y + 1)^2 (9 - y^2) == 0, {x, -5, 5}];
```



3. Dada la ecuación de los óvalos de Cassini $(x^2 + y^2 + a^2)^2 - b^4 - 4 a^2 x^2 = 0$, dibujemos en un mismo gráfico dos de estas curvas.

```
In[75]:= ImplicitPlot[{(x^2 + y^2 + 1)^2 - 1 - 4 x^2 == 0, (x^2 + y^2 + 1)^2 - 1.2 - 4 x^2 == 0}, {x, -3, 3},
           PlotStyle → {RGBColor[0.996109, 0, 0], RGBColor[0.996109, 0, 0.996109]}];
```



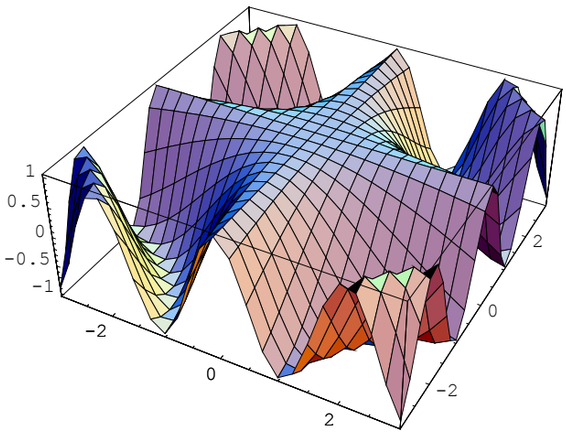
■ Curvas de nivel

Recordemos que si $f : U \subseteq R^2 \to R$, la curva de nivel 'c', es el conjunto $\{x \in U / f(x) = c\}$(c es un número real).

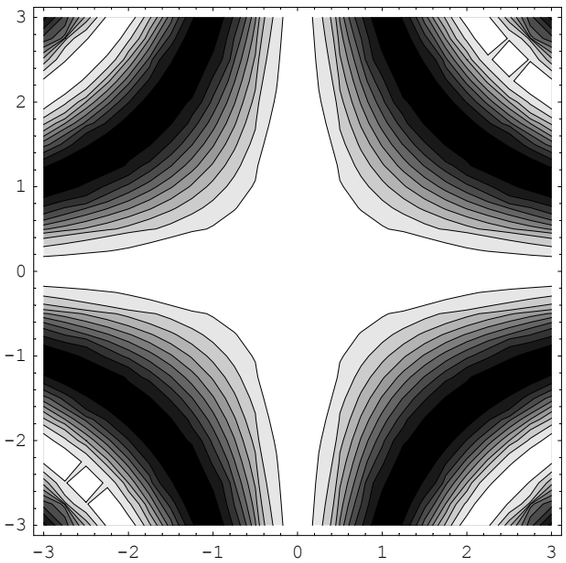
ContourPlot[f(x,y) , {x , x min, x max}, {y , y min, y max}] produce un gráfico sombreado de las curvas de nivel de f(x,y) como función de x e y.

El sombreado es más oscuro si corresponde a zonas bajas y más claro si corresponde a zonas altas de la superficie; si se quiere suprimir dicho sombreado, debemos utilizar la opción ContourShading→False .
El programa graficará, por defecto, 10 curvas de nivel. Para cambiar este número utilizamos la opción Contours→{n} (n cantidad de curvas de nivel). Si se quiere dibujar algunas curvas de nivel particulares, con Contours→{z_1, z_2, ...} se obtendrán las curvas de nivel z_1, z_2, ….
Sea f (x, y) = cos (xy)

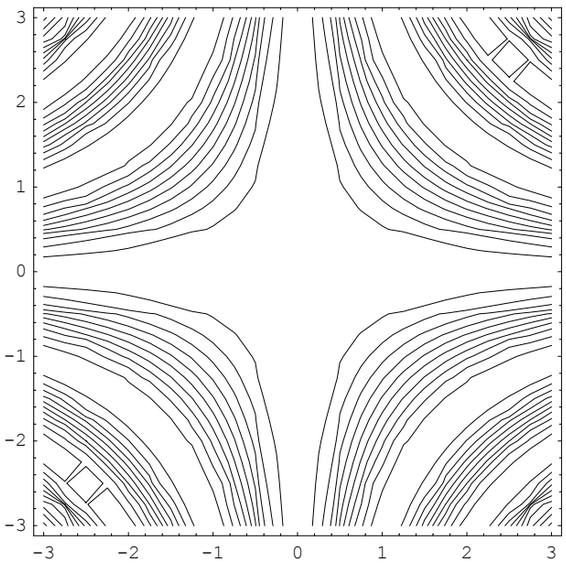
In[76]:= **Plot3D[Cos[x y], {x, -3, 3}, {y, -3, 3}];**



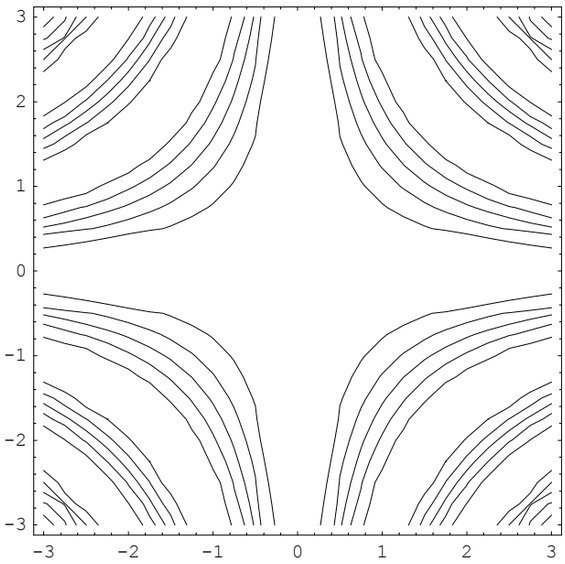
In[77]:= **ContourPlot[Cos[x y], {x, -3, 3}, {y, -3, 3}];**



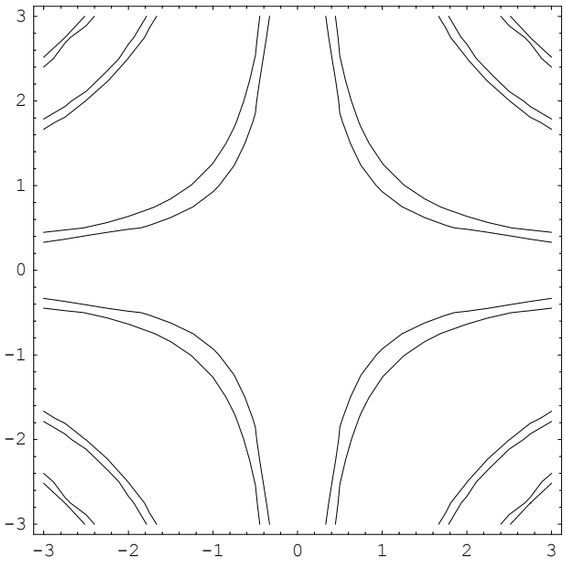
In[78]:= **ContourPlot[Cos[x y], {x, -3, 3}, {y, -3, 3}, ContourShading -> False];**



```
In[79]:= ContourPlot[Cos[x y], {x, -3, 3}, {y, -3, 3}, ContourShading -> False, Contours -> 5];
```



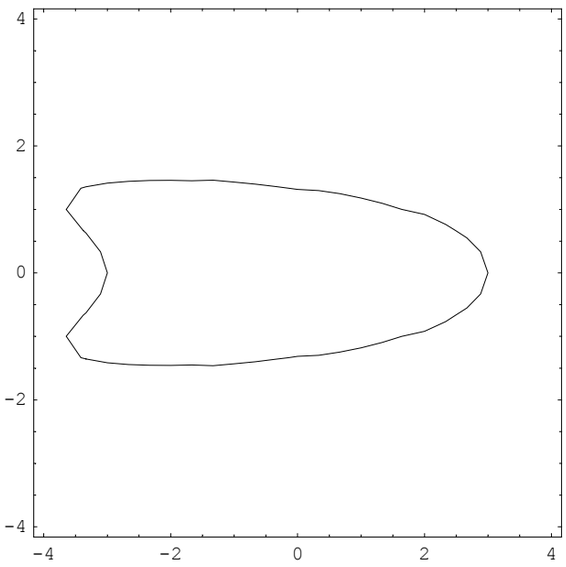

```
In[80]:= ContourPlot[Cos[x y], {x, -3, 3},
           {y, -3, 3}, ContourShading -> False, Contours -> {0.3, 0.6}];
```



Un uso interesante que podemos darle al comando ContourPlot, es el de graficar una función definida por la ecuación f(x,y) = c.

Por ejemplo, sea la curva definida por $3\,y^4 + 2\,x\,y^2 + x^2 = 9$. Tomando Contours→{9}, forzamos a graficar la curva pedida.

```
In[81]:= ContourPlot[3 y^4 + 2 x y^2 + x^2, {x, -4, 4},
           {y, -4, 4}, ContourShading -> False, Contours -> {9}];
```

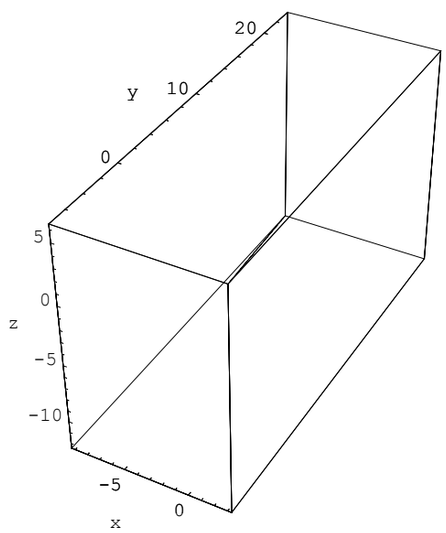


■ Curvas en el espacio

ParametricPlot3D[{x(t),y(t),z(t)},{t, t min, t max}]: grafica una curva en el espacio, dada en coordenadas paramétricas.

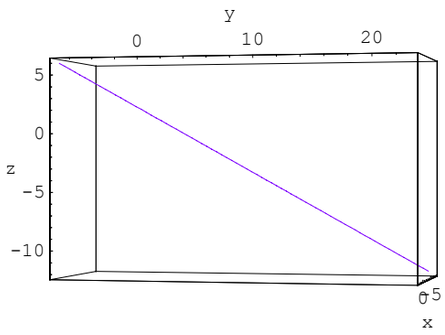
1. Sea la **recta** $\frac{x+1}{-2} = \frac{y-3}{5} = \frac{z}{-3}$. Las ecuaciones paramétricas son $(x, y, z) = t\,(-2, 5, -3) + (-1, 3, 0)$

```
In[82]:= ParametricPlot3D[{-2 t - 1, 5 t + 3, -3 t}, {t, -2, 4}, AxesLabel -> {x, y, z}];
```



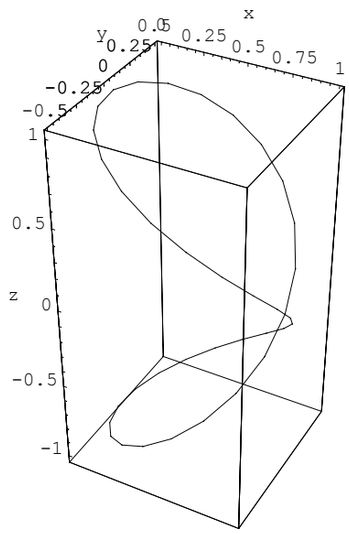
Como no se ve la recta, cambiamos el punto de vista y el color.
Aclaración: la posibilidad de agregar color en el comando ParametricPlot3D, es para la versión 5.

```
In[83]:= ParametricPlot3D[{-2 t - 1, 5 t + 3, -3 t, RGBColor[0.500008, 0, 0.996109]},
           {t, -2, 4}, ViewPoint -> {4.796, 1.132, 0.007}, AxesLabel -> {x, y, z}];
```



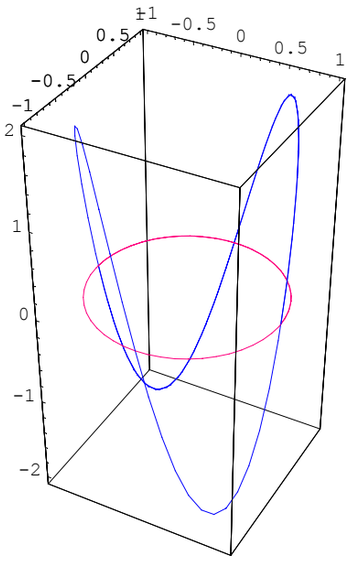
2. Graficamos la curva de Viviani:

```
In[84]:= ParametricPlot3D[{(Cos[t])^2, Sin[t] Cos[t], Sin[t]},
           {t, -2 π, 2 π}, AxesLabel -> {x, y, z}];
```



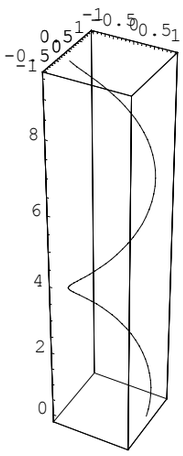
3. Representar las siguientes curvas en un mismo gráfico: $x = \cos t$ $y = \operatorname{sen} t$ $z = 0$ y $x = \cos t$ $y = \operatorname{sen} t$ $z = 4 \operatorname{sen} t \cos t$ en distinto color

```
In[85]:= ParametricPlot3D[{{Cos[t], Sin[t], 0, RGBColor[0.996109, 0, 0.500008]},
           {Cos[t], Sin[t], 4 Sin[t] Cos[t], RGBColor[0, 0, 0.996109]}}, {t, 0, 3 π}];
```



4. Dibujemos la hélice circular:

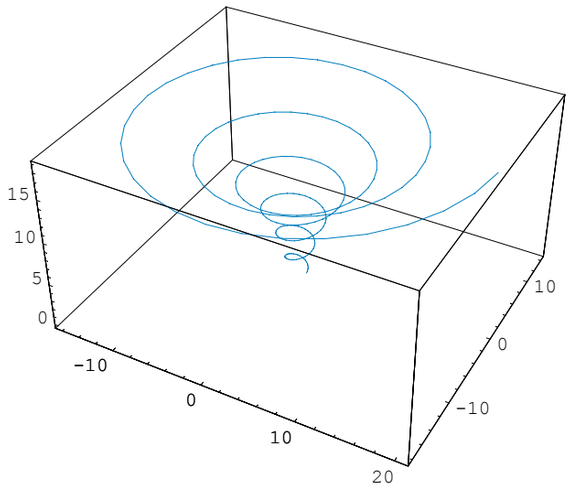
```
In[86]:= ParametricPlot3D[{Cos[t], Sin[t], t}, {t, 0, 3 π}];
```




```
In[87]:= ParametricPlot3D[{Cos[t], Sin[t], t, RGBColor[0.937514, 0.468757, 0]}, {t, 0, 9 π}];
```

3. El próximo gráfico corresponde a una curva helicoidal sobre una espiral logarítmica.

```
In[88]:= ParametricPlot3D[{E^0.08 t Cos[t], E^0.08 t Sin[t], 0.5 t, RGBColor[0, 0.500008, 0.750011]},
           {t, 0, 12 π}, PlotPoints -> 200];
```



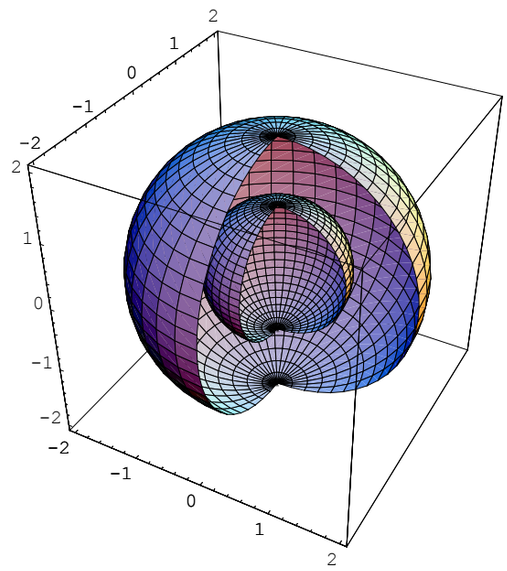
5. Gráfico de superficies

Ya vimos la representación de superficies que se definen por medio de una relación funcional. Graficaremos ahora las superficies parametrizables o localmente parametrizables.

ParametricPlot3D[{x(u,v), y(u,v), z(u,v)}, {u, u min, u max},{v, v min, v max}]: usando una parametrización de la superficie

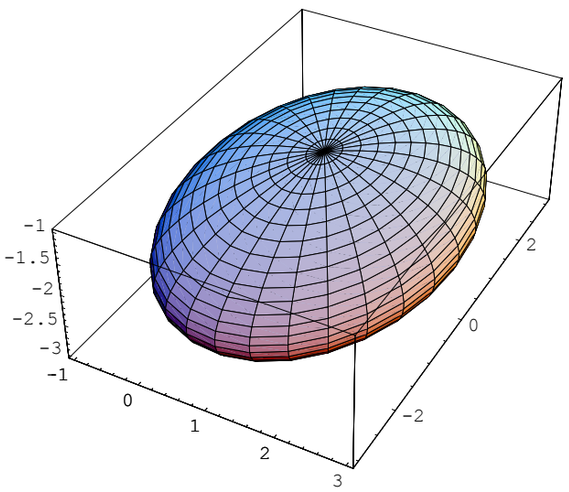
1. Una parametrización de la **esfera** es:
$x = a + r\,\text{cosu}\,\text{sen}\,v$, $y = b + r\,\text{sen}\,u\,\text{sen}\,v$, $z = c + r\cos v$ con $0 \le u \le 2\pi$, $0 \le v \le \pi$.
El centro es (a,b,c) y el radio r.
Realizamos en un mismo gráfico, un sector de las esferas de radio 1 y 2.

```
In[89]:= ParametricPlot3D[{{Cos[u] Sin[v], Sin[u] Sin[v], Cos[v]},
           2 {Cos[u] Sin[v], Sin[u] Sin[v], Cos[v]}}, {u, 0, 3/2 π}, {v, 0, π}];
```



2. Una paremitrización del **elipsoide** es:
$x = a + a_1\,\text{cosu}\,\text{sen}\,v$, $y = b + b_1\,\text{sen}\,u\,\text{sen}\,v$, $z = c + c_1\cos v$ con $0 \le u \le 2\pi$, $0 \le v \le \pi$.
El centro es (a,b,c).
Realizamos el gráfico de $\frac{(x-1)^2}{4} + \frac{y^2}{9} + (z+2)^2 = 1$

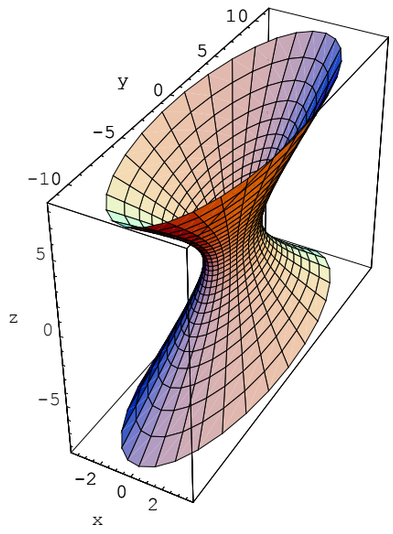
```
In[90]:= ParametricPlot3D[
           {1 + 2 Cos[u] Sin[v], 3 Sin[u] Sin[v], -2 + Cos[v]}, {u, 0, 2 π}, {v, 0, π}];
```



3. En la ecuación canónica del elipsoide si:
- **uno sólo** de los sumandos es negativo, entonces es un **hiperboloide de una hoja**.
- **dos** de los sumandos son negativos, entonces es un **hiperboloide de dos hojas**.

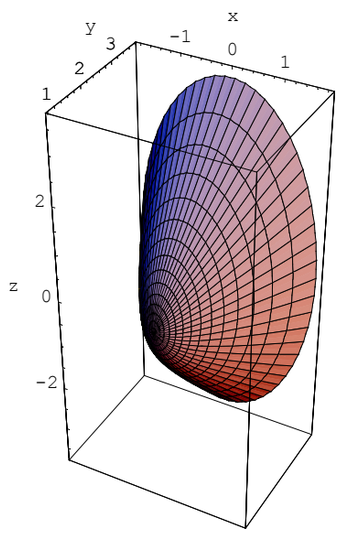
Por lo tanto la ecuación $x^2 + \frac{y^2}{9} - \frac{z^2}{5} = 1$ corresponde a un **hiperboloide de una hoja**.
Una parametrización es: $x = \cos v \operatorname{ch} u$, $y = 3 \operatorname{sen} v \operatorname{ch} u$, $z = \sqrt{5} \operatorname{sh} u$ con $0 \le v \le 2\pi$, u un real.

```
In[91]:= ParametricPlot3D[{Cos[v] Cosh[u], 3 Sin[v] Cosh[u], √5 Sinh[u]},
           {u, -2, 2}, {v, 0, 2 π}, AxesLabel -> {x, y, z}];
```



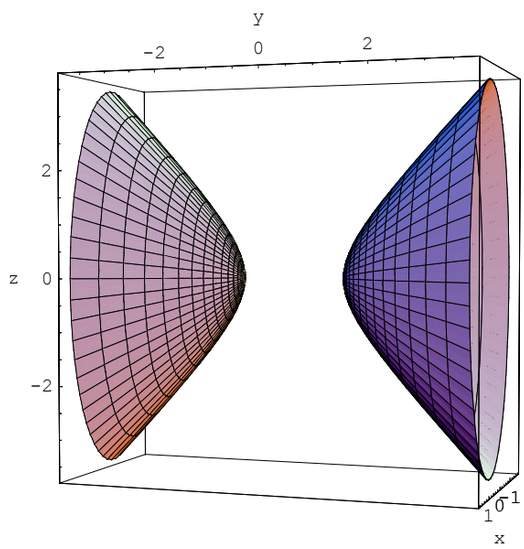
La ecuación $-\frac{x^2}{1/4} + y^2 - z^2 = 1$ corresponde a un **hiperboloide de dos hojas**.
Una parametrización es: $x = \frac{1}{2} \cos v \operatorname{sh} u$, $y = \operatorname{ch} u$, $z = \operatorname{sen} v \operatorname{sh} u$ con $0 \le v \le 2\pi$, u un real.

```
In[92]:= ParametricPlot3D[{ 1/2 Cos[v] Sinh[u], Cosh[u], Sin[v] Sinh[u]},
         {u, -2, 2}, {v, 0, 2 π}, AxesLabel -> {x, y, z}];
```



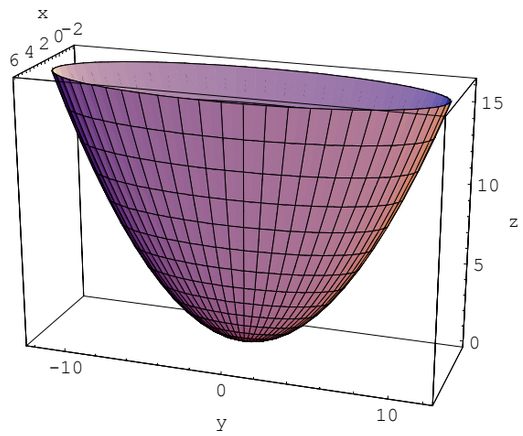
Pero no representa toda la superficie. De la misma manera que sucedió con la hipérbola, al considerar un sóla parametrización, no se ha tenido en cuenta que el signo de y es positivo o negativo, mientras que ch u es siempre positivo, por lo tanto para obtener la otra rama debemos considerar también $x = \frac{1}{2}\cos v \,\mathrm{sh}\, u$, $y = -\,\mathrm{ch}\, u$, $z = \mathrm{sen}\, v \,\mathrm{sh}\, u$, v un ángulo, u un número real.

```
In[93]:= ParametricPlot3D[{{ 1/2 Cos[v] Sinh[u], Cosh[u], Sin[v] Sinh[u]},
         { 1/2 Cos[v] Sinh[u], -Cosh[u], Sin[v] Sinh[u]}}, {u, -2, 2},
         {v, 0, 2 π}, AxesLabel -> {x, y, z}, ViewPoint -> {3.310, 1.132, 0.100}];
```



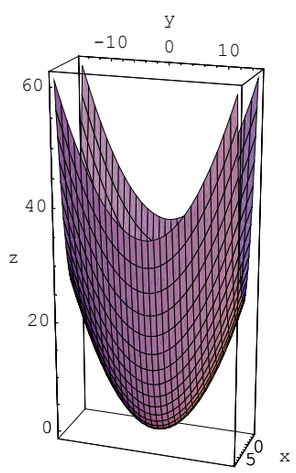
4. La ecuación $(x-2)^2 + \frac{y^2}{9} = z$ corresponde a un **paraboloide elíptico**. Una parametrización es: $x = u\cos v + 2$, $y = 3\, u\,\mathrm{sen}\, v$, $z = u^2$ con $0 \le v \le 2\pi$, u un real.

```
In[94]:= ParametricPlot3D[{u Cos[v] + 2, 3 u Sin[v], u^2}, {u, -4, 4},
           {v, 0, 2 π}, ViewPoint -> {3.310, 1.132, 1.240}, AxesLabel -> {x, y, z}];
```



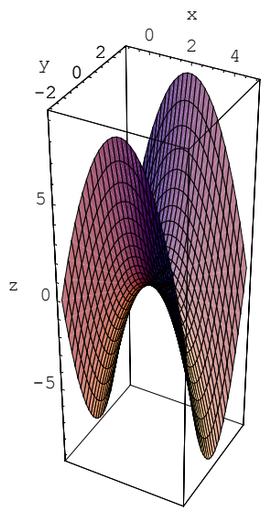
Otra parametrización se obtiene, pensando a z = f(x,y).
Es decir: $x = u$, $y = v$, $z = (u-2)^2 + \frac{v^2}{9}$, siendo u y v números reales.

```
In[95]:= ParametricPlot3D[{u , v, (u - 2)^2 + v^2/9}, {u, -4, 8}, {v, -15, 15},
           AxesLabel -> {x, y, z}, ViewPoint -> {3.310, 1.132, 1.240}];
```



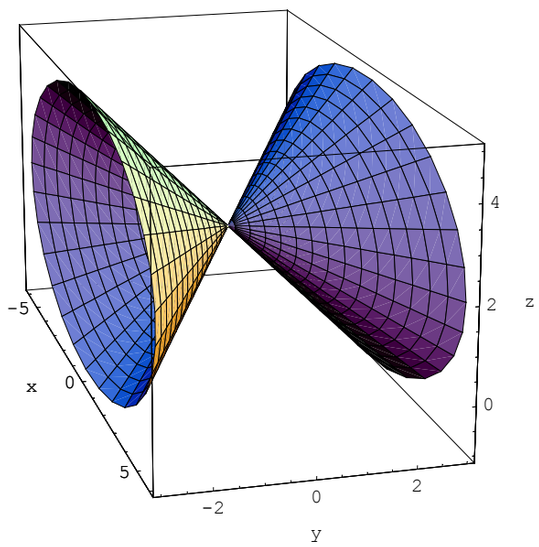
5. La ecuación $-(x-2)^2 + y^2 = z$ corresponde a un **paraboloide hiperbólico**. Una parametrización, pensando a z = f(x,y), es: $x = u$, $y = v$, $z = -(u-2)^2 + y^2$ siendo u y v números reales.

```
In[96]:= ParametricPlot3D[{u , v, -(u - 2)^2 + v^2}, {u, -1, 5}, {v, -3, 3}, AxesLabel -> {x, y, z}];
```



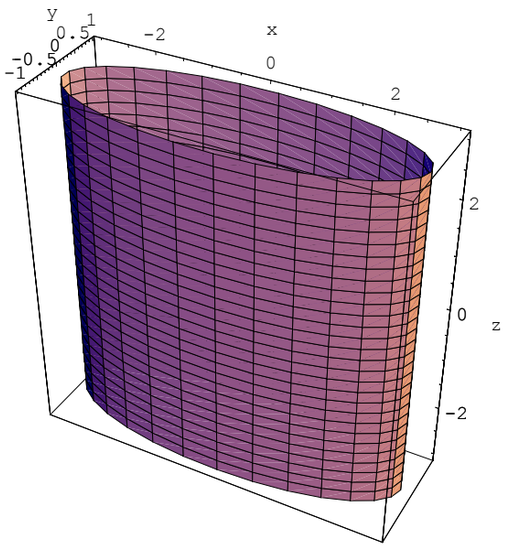
6. La ecuación $\frac{x^2}{4} + (z-2)^2 = y^2$ representa un **cono** con eje paralelo al eje y.
Una parametrización es $x = 2\,u\cos v$, $y = u$, $z = u\,\text{sen}\,v + 2$ con $0 \le v \le 2\pi$, u un número real.

```
In[97]:= ParametricPlot3D[{2 u Cos[v], u , u Sin[v] + 2}, {u, -3, 3},
           {v, 0, 2 π}, AxesLabel -> {x, y, z}, ViewPoint -> {3.630, -0.992, 1.131}];
```



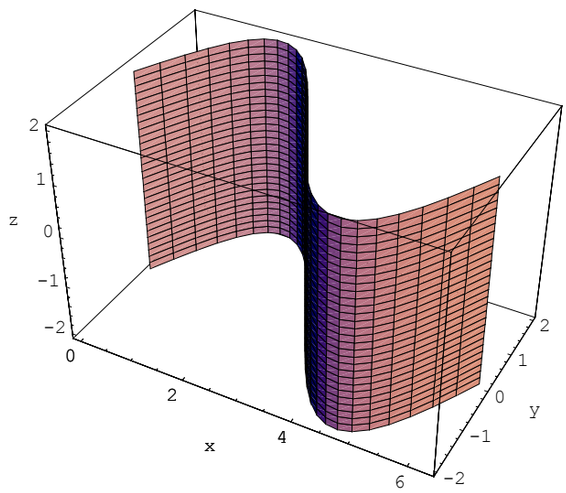
7.Vamos a grafica ahora dos superficies cilíndricas;
7.1. El conjunto de puntos del espacio que verifica $\frac{x^2}{9} + y^2 = 1$ representa un **cilindro elíptico** (su curva generatriz es una elipse), una parametrización es: $x = 3\cos v$, $y = \text{sen}\,v$, $z = u$ con $0 \le v \le 2\pi$, u un número real.

```
In[98]:= ParametricPlot3D[{3 Cos[v], Sin[v], u}, {u, -3, 3}, {v, 0, 2 π}, AxesLabel -> {x, y, z}];
```

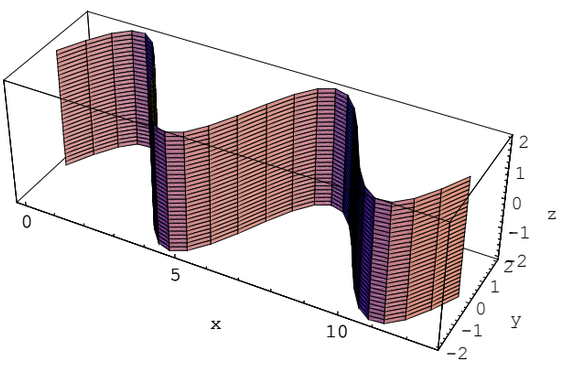


7.2. El conjunto de puntos del espacio que verifica y = 2 sen x, representa un cilindro cuya curva generatriz es la función seno. Una parametrización es $x = v$, $y = 2 \operatorname{sen} v$, $z = u$ con v un ángulo, u un número real.

```
In[99]:= ParametricPlot3D[{v, 2 Sin[v], u}, {u, -2, 2}, {v, 0, 2 π}, AxesLabel -> {x, y, z}];
```



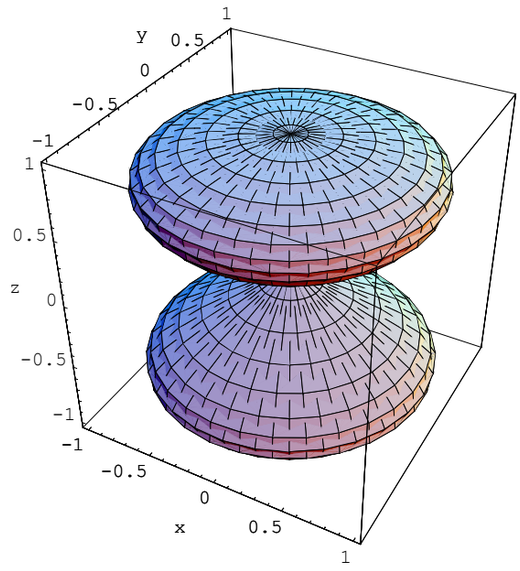

```
In[100]:=
ParametricPlot3D[{v, 2 Sin[v], u}, {u, -2, 2}, {v, 0, 4 π}, AxesLabel -> {x, y, z}];
```



2. Dibujamos una **superficie en ocho**, que posee una autointersección (singularidad en el origen), que es ignorada por el programa.

In[101]:=

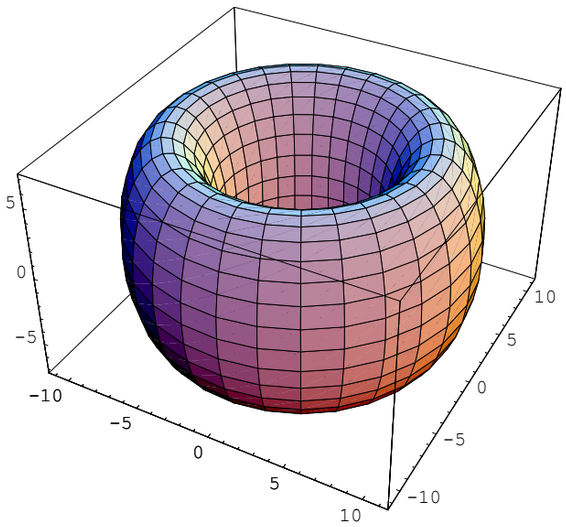
```
ParametricPlot3D[{Cos[u] Sin[2 v], Sin[u] Sin[2 v], Sin[v]},
  {u, 0, 2 π}, {v, 0, 2 π}, AxesLabel -> {"x", "y", "z"}];
```



4. Grafiquemos el toro y partes de él.

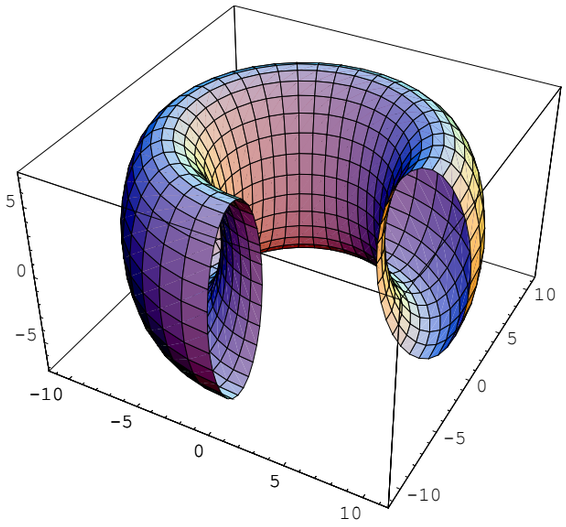
In[102]:=

```
ParametricPlot3D[
  {(8 + 3 Cos[v]) Cos[u], (8 + 3 Cos[v]) Sin[u], 7 Sin[v]}, {u, 0, 2 π}, {v, 0, 2 π}];
```



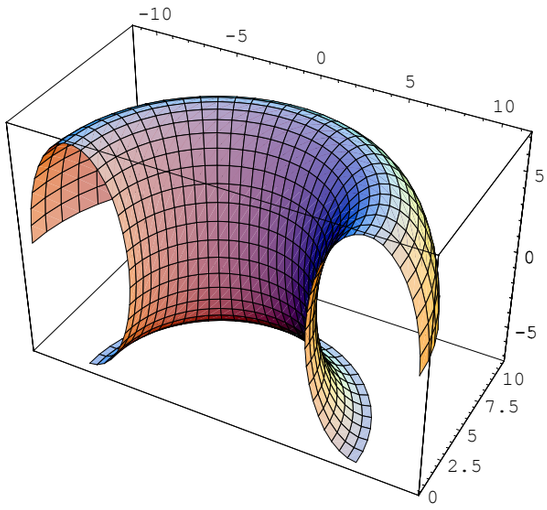
In[103]:=

```
ParametricPlot3D[
  {(8 + 3 Cos[v]) Cos[u], (8 + 3 Cos[v]) Sin[u], 7 Sin[v]}, {u, 0, 3/2 π}, {v, 0, 2 π}];
```



In[104]:=

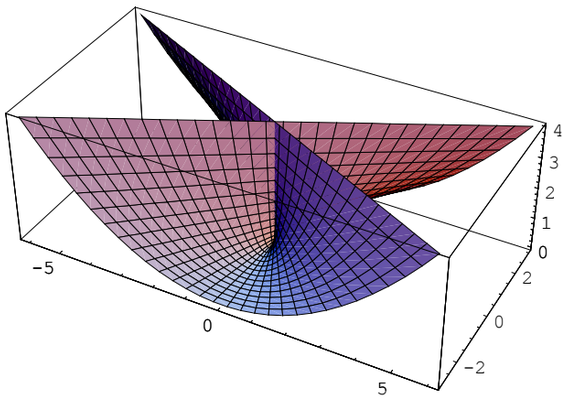
```
ParametricPlot3D[
  {(8 + 3 Cos[v]) Cos[u], (8 + 3 Cos[v]) Sin[u], 7 Sin[v]}, {u, 0, π}, {v, 0, 3/2 π}];
```



6. Veamos un ejemplo de superficie con un punto de pinzamiento: el caso del **paraguas de Whitney**

In[105]:=

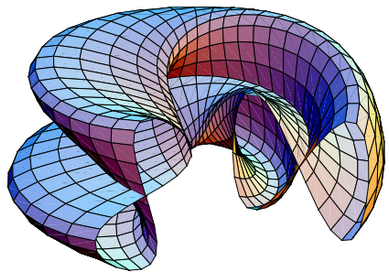
```
ParametricPlot3D[{u v, u, v^2}, {u, -3, 3}, {v, -2, 2}];
```



7. El programa también realiza el gráfico de superficies no orientables, como la **botella de Klein**
(puede definirse como la superficie que resulta de la rotación de la curva figura de ocho alrededor de un eje, al tiempo que dicha curva realiza un giro sobre su punto medio mientras describe la rotación anterior).

In[106]:=

```
ParametricPlot3D[{(1.5 + Cos[u/2] Sin[v] - Sin[u/2] Sin[2 v]) Cos[u],
  (1.5 + Cos[u/2] Sin[v] - Sin[u/2] Sin[2 v]) Sin[u], Sin[u/2] Sin[v] + Cos[u/2] Sin[2 v]},
 {u, 0, 3 π/2}, {v, 0, 2 π}, Axes -> False, Boxed -> False];
```



6. Gráfico de inecuaciones (solo en la versión 5)

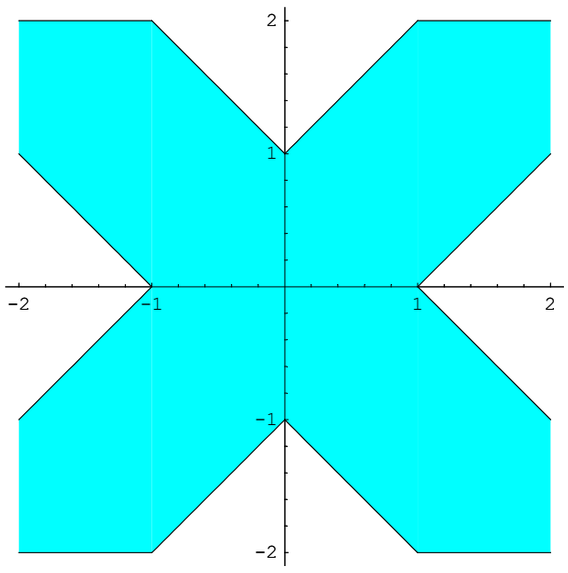
Tanto en el plano como en el espacio, también se puede graficar el conjunto solución de una o mas inecuaciones. Es necesario cargar el paquete
Graphics`InequalityGraphics`

In[107]:=

```
<<Graphics`InequalityGraphics`
```

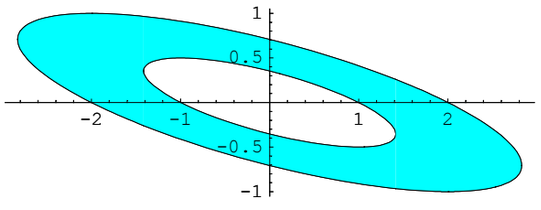
In[108]:=

```
InequalityPlot[ Abs[ Abs[x] - Abs[y]] ≤ 1, {x, -2, 2}, {y, -2, 2}];
```



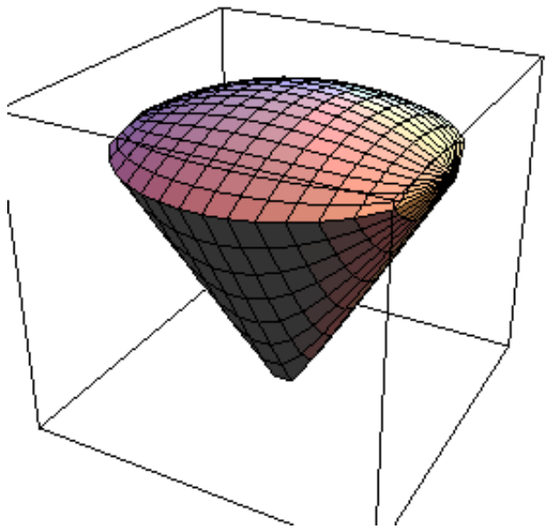
In[109]:=

```
InequalityPlot[ 1 ≤ (x + 2 y)^2 + 4 y^2 ≤ 4, {x, -3, 3}, {y, -3, 3}];
```



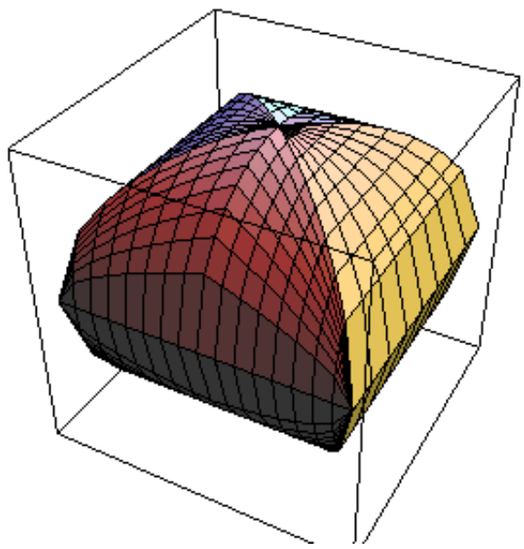
In[110]:=

```
<< RealTime3D`
  InequalityPlot3D[ x^2 + y^2 + z^2 ≤ 4 ⋀ 3 x^2 + 3 y^2 ≤ z^2 ⋀ z ≥ 0, {x}, {y}, {z}, PlotPoints → 15];
```



In[111]:=

```
<< RealTime3D`
  InequalityPlot3D[ x^2 + z^2 ≤ 4  && y^2 + z^2 ≤ 4, {x, -5, 5}, {y, -5, 5}, {z, -5, 5}];
```



In[112]:=

```
<< RealTime3D` InequalityPlot3D[
   Abs[ x y z] ≤ 2 && x^2 + y^2 + z^2 < 9, {x, -4, 4}, {y, -4, 4}, {z, -4, 4}];
```