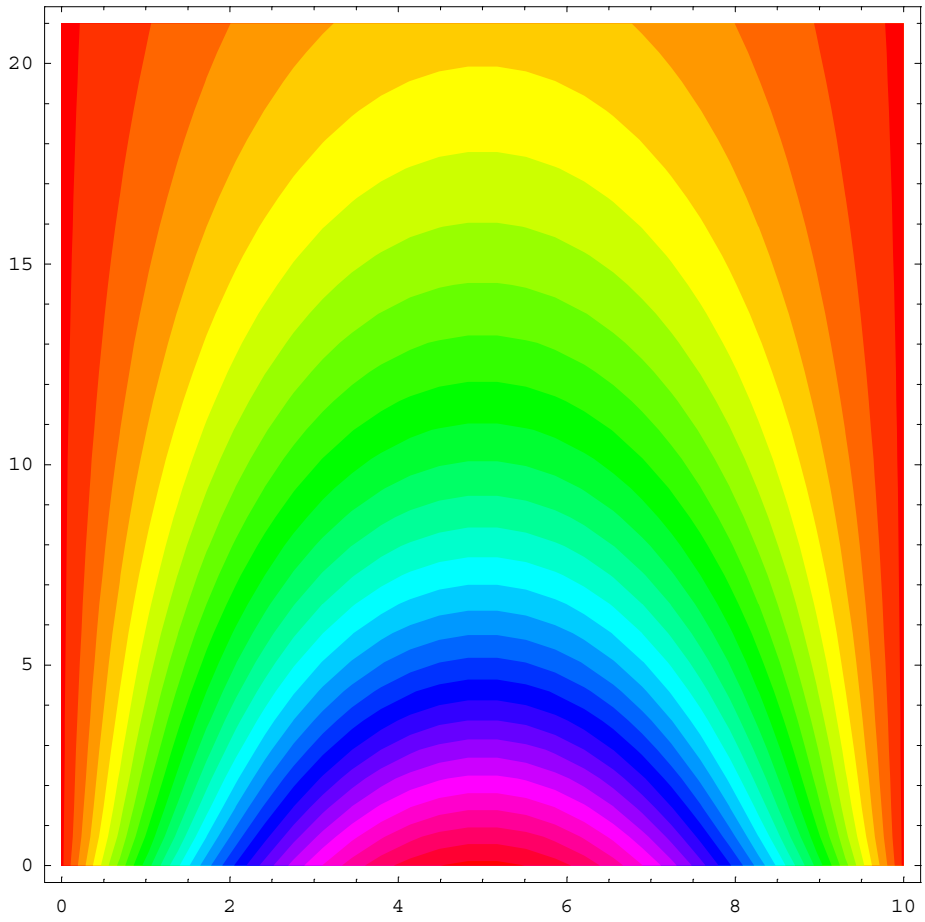
Prácticas de Matemáticas con *Mathematica* para Ingenieros



Manuel Calixto

Departamento de Matemática Aplicada y Estadística
Universidad Politécnica de Cartagena

Manuel Calixto Molina
Profesor Titular de Matemática Aplicada
de la Universidad Politécnica de Cartagena

Prácticas de Matemáticas

con Mathematica para

Ingenieros

Depósito Legal: MU-1347-2002

I.S.B.N.: 84-607-5156-2

Julio-2002

Prólogo

La ilustración de la portada se corresponde con un gráfico de curvas de nivel de la distribución en el tiempo (eje de ordenadas) de la temperatura $T(x, t)$ de un vástago de longitud $L=10$ (eje de abscisas), con extremos en contacto con termostatos a temperatura cero (condiciones de contorno) y con temperatura inicial $T(x, 0) = -x(x - 10)$. Tanto la representación gráfica como la solución numérica de la correspondiente ecuación en derivadas parciales que gobierna el fenómeno (ecuación de difusión del calor) han sido obtenidas con la versión 4.0 del programa *Mathematica* y constituyen parte de una de las 13 prácticas contenidas en este libro, las cuales son:

- Introducción al *Mathematica*
- Álgebra lineal con *Mathematica*
- Gráficos con *Mathematica*
- Cálculo diferencial e integral con *Mathematica*
- Manipulación de datos experimentales: ajustes e interpolación
- Sucesiones y series con *Mathematica*
- Geometría de curvas y superficies
- Operadores vectoriales en coordenadas curvilíneas
- Ecuaciones diferenciales ordinarias
- Oscilaciones amortiguadas y forzadas
- Sistemas de EDOs. Oscilaciones acopladas. Estabilidad
- Campos de velocidades, líneas de corriente y superficies equipotenciales
- Integrales curvilíneas y de superficie: Teoremas de Stokes y Gauss
- Ecuaciones en Derivadas Parciales. Animaciones

El origen de dichas prácticas está en unos cuadernillos preparados durante el curso 2000/2001 para las asignaturas de Fundamentos Matemáticos de la Ingeniería y Ampliación de Matemáticas en la titulación de Ingeniero Técnico en Obras Públicas de la Universidad Politécnica de Cartagena. Dichos cuadernillos han sido actualizados y fundidos en este nuevo manual.

Son cada día más abundantes los manipuladores algebraicos, como *Mathematica* y otros, y los manuales que actualizan la enseñanza y el aprendizaje de la Matemática Aplicada utilizando los recursos computacionales y gráficos de dichos paquetes informáticos. Con ayuda de ellos, la complejidad de los problemas prácticos a tratar puede trasladarse al planteamiento y a la interpretación de los resultados, dejando a la máquina la ardua y mecánica tarea del cálculo.

La presente obra contiene multitud de problemas prácticos con los que el alumno puede explorar y profundizar en el entendimiento de numerosos conceptos matemáticos. Es mas, existen prácticas dedicadas exclusivamente al estudio de problemas físicos como: "oscilaciones amortiguadas, forzadas, acopladas, etc" en el apartado de ecuaciones diferenciales ordinarias, "cuerdas vibrantes y difusión del calor" en el apartado de ecuaciones en derivadas parciales, "fluidos" en el apartado de campos de velocidades, o la "manipulación de datos experimentales" en el tema de ajustes e interpolación.

El manual pretende cubrir los contenidos matemáticos tratados usualmente en primer y segundo curso de cualquiera de las ingenierías, excepto en lo referente a "Variable Compleja y Transformadas de Laplace", tema de gran interés en ingeniería electrónica pero que no considero aquí.

Me gustaría que este manual resultara de utilidad tanto a alumnos como a profesores en la enseñanza de la Matemática Aplicada, y agradecería sobremanera cualquier sugerencia, comentario o crítica que ayude a mejorarlo.

Quisiera finalmente agradecer a Concepción Bermúdez Edo, José Salvador Cánovas Peña y Gabriel Soler López la ayuda que me prestaron en el comienzo de la elaboración de estas prácticas.

Cartagena a 24 de junio de 2002

EL AUTOR.

E-mail: Manuel.Calixto@upct.es

Índice:

Preliminares: Introducción al *Mathematica*

■ Respuestas rápidas acerca de la sintaxis de los comandos

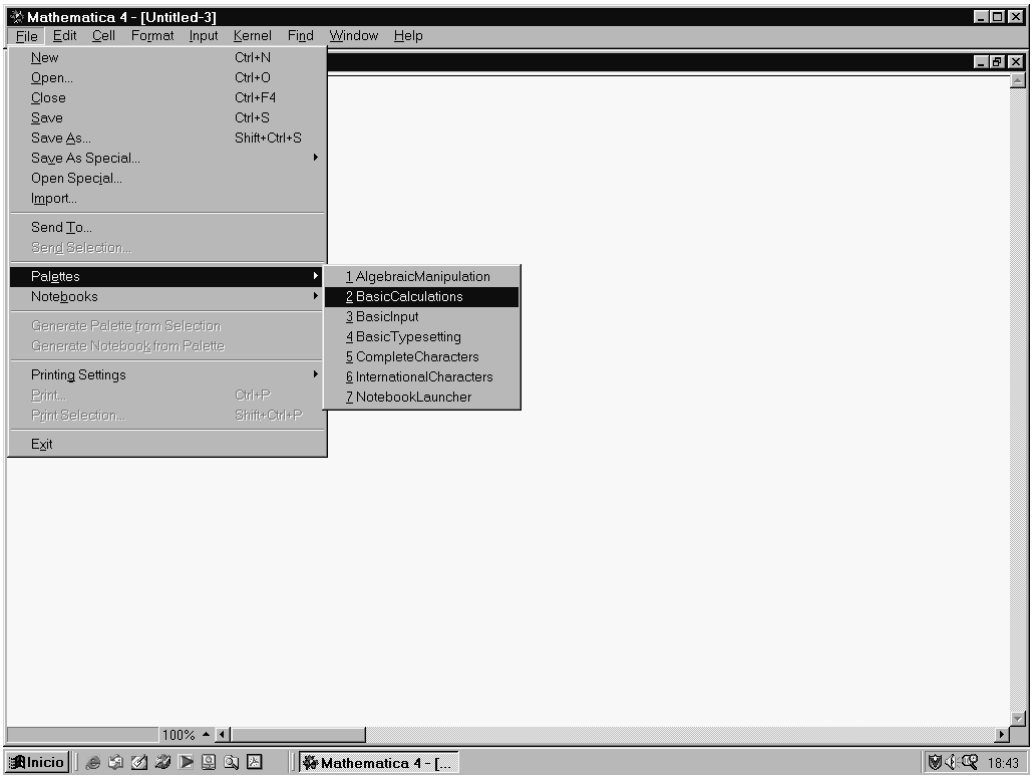
Todos los comandos que utilizarás tendrán esta forma:

Comando[argumentos]

donde la primera letra de cualquier comando empieza siempre con ¡MAYUSCULA! y los argumentos vienen dados entre ¡CORCHETES! [] y NO entre PARENTESIS ().

Todos los comandos que utilizarás están en inglés, lo que dificulta su memorización para aquellos que no dominen este lenguaje. Si tienes dudas a cerca de la sintaxis de un comando y de los argumentos que requiere, tienes varias opciones para que *Mathematica* te lo recuerde:

- Si te acuerdas como empieza el comando, teclea las primeras letras del mismo y presiona a la vez CTRL-k. *Mathematica* te mostrará una lista de posibles formas de completar el comando.
- Presiona CTRL-SHIFT-k, y *Mathematica* te proporcionará una plantilla con el comando y todos sus argumentos opcionales.
- Usar las paletas de entrada que se proporcionan en el submenú File ▹ Palettes.



Pincha simplemente en una de las 7 opciones y *Mathematica* te escribirá una plantilla del comando. Por ejemplo, echa un vistazo a la paleta Basic Calculations.

También puedes conseguirbotón ayuda de un comando escribiendo

?Nombredelcomando*

para información básica o

??Nombredelcomando*

para información más detallada (el asterisco es para cuando sólo sepas las primeras letras). Por ejemplo, el comando **Plot3D** realiza gráficas en tres dimensiones. Si no sabemos, o no nos acordamos, de sus argumentos escribiremos:

```
In[1]:= ?? Plot3D*

Plot3D[f, {x, xmin, xmax}, {y, ymin, ymax}] generates a three-
  dimensional plot of f as a function of x and y. Plot3D[
  {f, s}, {x, xmin, xmax}, {y, ymin, ymax}] generates a
  three-dimensional plot in which the height of the surface
  is specified by f, and the shading is specified by s.

Attributes[Plot3D] = {HoldAll, Protected}

Options[Plot3D] =
 {AmbientLight → GrayLevel[0], AspectRatio → Automatic,
  Axes → True, AxesEdge → Automatic, AxesLabel → None,
  AxesStyle → Automatic, Background → Automatic, Boxed → True,
  BoxRatios → {1, 1, 0.4}, BoxStyle → Automatic, ClipFill → Automatic,
  ColorFunction → Automatic, ColorFunctionScaling → True,
  ColorOutput → Automatic, Compiled → True,
  DefaultColor → Automatic, Epilog → {}, FaceGrids → None,
  HiddenSurface → True, ImageSize → Automatic, Lighting → True,
  LightSources → {{{1., 0., 1.}, RGBColor[1, 0, 0]}, {{1., 1., 1.},
     RGBColor[0, 1, 0]}, {{0., 1., 1.}, RGBColor[0, 0, 1]}},
  Mesh → True, MeshStyle → Automatic, Plot3Matrix → Automatic,
  PlotLabel → None, PlotPoints → 15, PlotRange → Automatic,
  PlotRegion → Automatic, Prolog → {}, Shading → True,
  SphericalRegion → False, Ticks → Automatic, ViewCenter → Automatic,
  ViewPoint → {1.3, -2.4, 2.}, ViewVertical → {0., 0., 1.},
  DefaultFont :→ $DefaultFont, DisplayFunction :→ $DisplayFunction,
  FormatType :→ $FormatType, TextStyle :→ $TextStyle}
```

Otra posibilidad es por supuesto, utilizar el submenú de ayuda Help ▷ Help Browser, donde vienen recogidos todos los comandos con ejemplos de su funcionamiento.

■ Empezando a calcular

El programa *Mathematica* no ejcutará ningún comando hasta que no se lo indiques pulsando teclas :

SHIFT-↵. (Mayúsculas+Retorno de carro, simultáneamente), o bien la tecla: ***Intro*** (tec numérico).

Por ejemplo, si escribes 5 3 ó 5*3 y presionas I*ntro* obtendrás:

```
In[2]:= 5 3
Out[2]= 15
```

Observe que el resultado va precedido por " Out[n]= ", que indica que es la celda de salida

(output) número n. Aquí te damos la forma de escribir las operaciones aritméticas elementales para que vayas practicando:

Operaciones Aritméticas Elementales

SUMA: **x + y**

RESTA: **x-y**

PRODUCTO: **x** SPACE **y = x*y (asterisco o un espacio en blanco)**

DIVISION: **x/y**

POTENCIA x^y: **x^y=Power[x,y]** (El símbolo ^ no aparece hasta que no se pulsa tecla)

Para potencias y fracciones también puedes ayudarte de la paleta "BasicCalculations". Es IMPOR TANTE que respetes el ORDEN DE PRIORIDAD de las operaciones:

1) *Mathematica* evalúa primeramente las potencias **x^y**

2) Seguidamente la multiplicación * y la división /

3) Por último la suma + y la resta -

Si dos operadores tienen la misma prioridad se evalúa de izquierda a derecha: 8/4/2=1=(8/4)/2≠8/(4/2)

Para cambiar el orden de ejecución de las operaciones utilizamos los PARENTESIS (). recomienda el uso de paréntesis allá donde haya duda.

Ejercicio:

Realizar las siguientes operaciones y comparar los resultados:

a) 2* 4+3, a') 2(4+3)

b) $(2-3)^{4+7}$, b') 2-3^4+7

c) 6/3-2 , c') 6/(3-2)

e) 2 3^2 + (5-2)*3, e') 23^2+5-2*3

f) a^(b/(c+d)) f') a^b/c+d

Observación: aunque nosotros introducimos las operaciones como una cadena de caracteres, ejemplo:

```
In[3]:= a^(b / (c + d))
```

Out[3]= $a^{\frac{b}{c+d}}$

Mathematica nos da su expresión convencional: $a^{\frac{b}{c+d}}$

Nota: también es necesario utilizar paréntesis cuando se trabaja con números negativos: -3^2= -9≠ (-3)^2

Nos reiteramos, como norma general se recomienda NO ECONOMIZAR EN EL USO DE PARÉ NTESIS. Es decir, allá donde tengamos dudas en la prioridad de las operaciones, se recomienda utilizar PARÉNTESIS

Funciones Matemáticas Elementales:

Log[x] = ln x (logaritmo neperiano)
Log[b, x] = $\log_b x$ (logaritmo en base b)
Exp[x] = e^x (exponencial)

Sin[x] = sen x
ArcSin[x] = arcsen x
Cos[x] = cos x
ArcCos[x] = arccos x
Tan[x] = tan x

ArcTan[x] = arctan x
Sinh[x] = senh x (seno hiperbólico)
ArcSinh[x] = arcsenh x (arcoseno hiperbólico)
Cosh[x] = cosh x
ArcCosh[x] = arccosh x
Tanh[x] = tanh x
ArcTanh[x] = arcsenh x

Sqrt[x] = $\sqrt{x}$
n ! = factorial de n (si n ∈ ℕ)
GCD[x, y] = máximo común divisor de x e y
LCM[x, y] = mínimo común múltiplo de x e y
Max[x, y, ...] = Máximo de x, y, ...
Min[x, y, ...] = Mínimo de x, y, ...
Abs[x] = | x | (valor absoluto de x)

Nótese que todas las funciones empiezan por MAYUSCULA. *Mathematica* distingue entre mayúsculas y minúsculas. Si escribimos

```
In[4]:= sin[Pi]

        General::spell1 : Possible spelling error: new
           symbol name "sin" is similar to existing symbol "Sin".

Out[4]= sin[π]
```

en vez de

```
In[5]:= Sin[Pi]

Out[5]= 0
```

entonces *Mathematica* no nos entenderá. Tampoco nos entenderá si escribimos **pi** en vez de **Pi.** O si escribimos

```
In[6]:= Sin (Pi)
Out[6]= π Sin
```

en vez de **Sin[Pi]**, ya que, como hemos comentado antes, el argumento **[Pi]** va siempre entre CORCHETES.

Constantes matemáticas elementales

Pi $= \pi \simeq 3.14159$ **E** $= e \simeq 2.71828$
I $= i = \sqrt{-1}$ **Infinity** $= \infty$
Degree $= \pi/180.$

Conversion de grados a ***radianes :*** las funciones trigonométricas trabajan en radianes. Si queremos trabajar con grados es necesario realizar la conversion. Por ejemplo :

```
In[7]:= Sin[30 Degree]
Out[7]= 1/2
```

Valores aproximados

Mathematica no ofrece valores numéricos de, por ejemplo, fracciones a no ser que se lo pida mos. Por ejemplo, si escribimos

```
In[8]:= √3 / 4 + 5 / 3
Out[8]= 5/3 + √3/4
```

Mathematica no hace básicamente nada. Para obligarlo a que nos dé **un valor numérico aproxi mado** (con 6 cifras significativas) tendremos que escribir:

```
In[9]:= √3 / 4 + 5 / 3 // N
Out[9]= 2.09968
```

o bien:

```
In[10]:= N[√3 / 4 + 5 / 3, 20]
Out[10]= 2.0996793685588859900
```

si queremos que nos de 20 cifras decimales (o las que uno desee). Es decir:

N[x,n] nos da un valor numérico de **x** con **n** cifras decimales**.**

Muchas veces, *Mathematica* no nos proporciona el número **n** de cifras decimales que le pedimos.

Esto se puede deber bien a que el valor **x** es ya exacto o bien a que *Mathematica* redondeando.

Ejercicio:

Realizar las siguientes operaciones y comprobarlas:

a) $\text{sen } 30^0 + \cos \pi /3$ b) $\log_2 256$

c) $\sqrt[3]{8}$ d) $\ln 0$

e) $1/\infty$ f) $(e^{1/2} - 1)^{1/2}$

g) $100!$ h) $2^{10!}$

Obtener aproximaciones a $\tan(30^o)$ con 6, y 10 cifras significativas

ERRORES FRECUENTES. "NO FUNCIONA..."

Si no te salen las cuentas, comprueba una vez mas:

a) **Mayúsculas y minúsculas**: el programa distingue unos caracteres de otros. Todos los coman dos del programa empiezan por MAYUSCULA: **sin[x]≠Sin[x], n[1/6]≠N[1/6]**, etc .

b) **Espacios en blanco** : se interpretan como un signo de multiplicar. Mientras que **sol** es única variable, **s o l** se interpreta como **s*o*l**

c) **Paréntesis y corchetes**: los paréntesis agrupan e indican prioridad de operaciones. Los cor chetes son exclusivos de las funciones para delimitar argumentos.

Por ejemplo, si escribes **N(1/6,8)** en vez de **N[1/6,8]**, *Mathematica* no te entenderá.

■ Definición de variables y funciones

Estos elementos nos van a permitir definir y guardar en nuestros archivos, o durante una sesió nuevas constantes y funciones que podremos utilizar de la misma forma que las propias de *Ma matica*.

Para definir variables y funciones podemos darles cualquier **nombre** siempre que dicho nombre **no empiece por un número ni por** el símbolo **$.**

Es conveniente que la **primera letra** del nombre sea **MINUSCULA**, para evitar incompatibil idades con otros comandos propios de *Mathematica* (los cuales, volvemos a recordar, comienzan siempre con una letra mayúscula).

Definición de variables

Al definir y trabajar con variables y funciones en Mathematica, hemos de tener en cuenta que: **x** e **y** son dos variables que hemos definido entonces:

a) **x**SPACE **y,** significa **x *y**. Mientras que. **xy** (sin espacio entre ellas) será una nueva variable nueva e independiente de las variables **x** e **y**.

b) **5x** significa **5*x** , al igual que **x**SPACE **5** (x espacio 5) . No obstante, **x5** (juntos) será NUEVA VARIABLE (x sub 5).

Por ejemplo, podemos asignar números a diferentes variables (nombres) escribiendo en misma linea las asignaciones separadas por punto y coma (;):

```
In[11]:= bartolo = 2; faustino = 3; jesusa = 4;
```

Si al final de la última expresión ponemos ";" no obtenemos ninguna salida "Out" en pantalla pulsar *Intro,* aunque las asignaciones quedan almacenadas en memoria. En efecto, si ahora escribimos

```
In[12]:= jesusa^bartolo
         (bartolo + jesusa) / faustino
Out[12]= 16
Out[13]= 2
```

obtendremos los valores numéricos de dichas operaciones para las asignaciones anteriores. queremos liberar a **bartolo** y a **faustino** de sus valores numéricos (2 y 3), tendremos que escribir

```
In[14]:= Clear[bartolo, faustino]
```

con lo cual ahora, si definimos una nueva variable

```
In[15]:= antonia = jesusa^bartolo + faustino
Out[15]= 4^bartolo + faustino
```

y queremos ver el valor que toma **antonia** para valores particulares de **bartolo** y **faustino**, se escribe:

```
In[16]:= antonia /. {bartolo -> 1 / 2, faustino -> 6}
Out[16]= 8
```

que se lee: "valor que toma antonia evaluada en bartolo->1/2 y faustino->6". La diferencia entre escribir **bartolo=1/2**, y **bartolo->1/2** es que la primera (el signo =) asigna el valor 1/2 a la vari able bartolo, el cual queda grabado en memoria (hasta que liberemos a bartolo de ese valor medi ante **Clear[bartolo]**), mientras que la segunda (el signo ->) sustituye bartolo por 1/2 en la evalu ación de la variable antonia, pero este valor no queda guardado en memoria.

El asignar valores concretos a ciertas variables resulta especialmente útil cuando tenemos repetir ciertos cálculos donde aparece una constante física cuyo valor numérico es difícil recordar o difícil de escribir. Por ejemplo, supongamos que estamos utilizando la ecuación de gases perfectos $P\,V$ = NRT, donde R = 8.31451 Julios/(Kelvin.Mol) es la constante de los gases perfectos. Para no interferir con otras constantes internas de *Mathematica*, denotemos con minú cula r a R y definamos:

```
In[17]:= r = 8.31451
Out[17]= 8.31451
```

Definamos la presión **p** en términos del volumen **v**, la temperatura **t** y el número de moles como:

```
In[18]:= Clear[p, t, n, v]; p = n r t / v
Out[18]= 8.31451 n t
         ------------
               v
```

Ya podemos evaluar p en términos de valores concretos de **n, t** y **v,** como:

```
In[19]:= p /. {n -> 3, t -> 273, v -> 1}
Out[19]= 6809.58
```

Ejercicio

Obtener el valor de p para:

a) (n,t,v)=(2,300,1.5)

b) (n,t,v)=(2,350,5)

Definición de funciones

Mathematica tiene muchas funciones incorporadas, pero nosotros podemos querer definir nues tras propias funciones. La forma más habitual de definir una función en *Mathematica* es:

nombrefuncion[variable1_,variable2_, ... ,variablen_] := expresión

Es decir, le damos un nombre a la *función* y *entre CORCHETES* ponemos la *variable,* o vari ables, junto al símbolo de *SUBRAYADO* `` _ ´´. A continuación escribimos *DOS PUNTOS* seguidos del signo *IGUAL* "=" y la expresión que queramos . Si hay mas de una variable, éstas separan por comas.

Ejemplo:

```
In[20]:= f[x_, y_] := Sin[x] Cos[y]
```

Mathematica no nos ofrece ningún Out, pero guarda en memoria esta función, de manera que escribimos:

```
In[21]:= f[Pi / 4, Pi / 3]
Out[21]= 1/(2 Sqrt[2])
```

obtendremos la función **f** evaluada en un punto concreto **(x,y)=(Pi/4,Pi/3).**

Podemos repetir también el ejercicio anterior sobre la ecuación de estado de un gas perfecto definir la presión como una función de **n,t,** y **v** como:

```
In[22]:= Clear[p, n, t, v]; p[n_, t_, v_] := n r t / v
```

Con lo cual ahora sólo tendremos que escribir

```
In[23]:= p[3, 273, 1]
Out[23]= 6809.58
```

para obtener el mismo resultado que con la asignación **p/.{n->3,t->273,v->1}** (más engorrosa escribir).

La ventaja ahora es que podemos derivar, integrar, etc la función **p[n,t,v],** cosa que no podrí mos hacer si definiéramos **p** como variable en vez de como función.

Ejercicio

Definir la fuerza entre dos masas $F = G\,M\,\frac{m}{r^2}$ como una función de las masas *m*, *M* y de distancia *r*,

donde $G = 6.67266 \times 10^{-11}\ \frac{\text{Meter}^2\ \text{Newton}}{\text{Kilogram}^2}$ es la constante de gravitación universal. Calcular fuerza para:

a) *(m,M,r)*=(1Kg, 5.9737×10^{24}Kg, 6.37814×10^{6}metros)

b) *(m,M,r)*=(1Kg, 5.9737×10^{24}Kg, 2×6.37814×10^{6}metros)

Los valores $m_{\oplus} = 5.9737 \times 10^{24}$Kg y $r_{\oplus} = 6.37814 \times 10^{6}$metros se corresponden con la masa radio de la tierra.

Definición de funciones a trozos

También podemos definir funciones a trozos como: $h(x) = \begin{cases} x^2 + 1, & \text{si } x \le 0 \\ 1, & \text{si } x > 0 \end{cases}$ de la siguiente manera:

In[24]:=

```
h[x_] := If[x <= 0, x^2 + 1, 1]
```

donde el comando **If[condición, procesoV,procesoF]** funciona realizando **procesoV** si la **condición** es verdadera, **procesoF** si la condición es falsa. Uno puede comprobar que efectivamente la función está definida a trozos sin mas que evaluar:

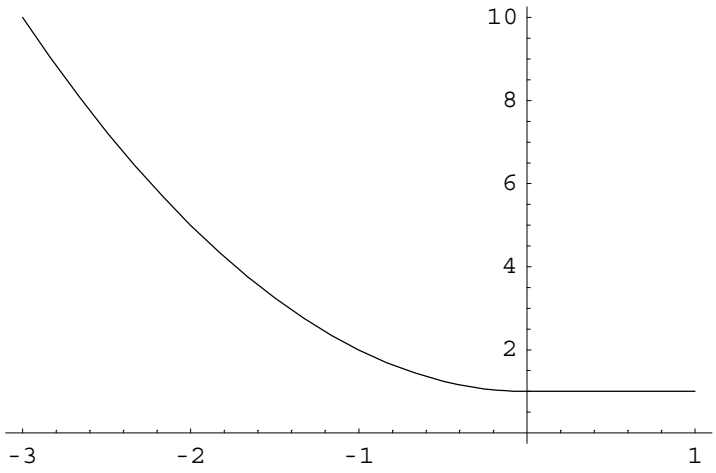
In[25]:=
```
h[-2]
h[2]
```

Out[25]= 5

Out[26]= 1

o bien mediante una representación gráfica en el intervalo [-3,1]

```
In[27]:= Plot[h[x], {x, -3, 1}]
```



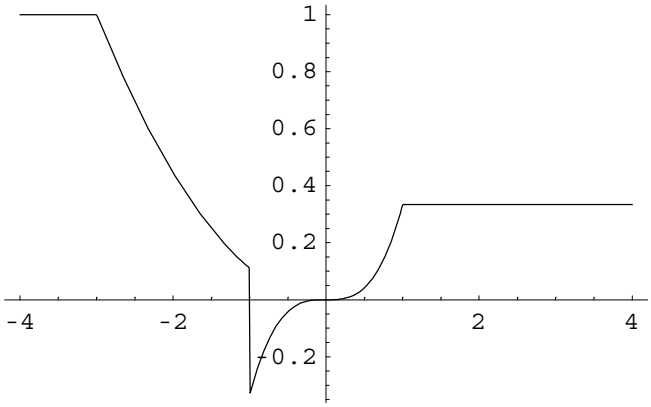

```
Out[27]= - Graphics -
```

Cuando la función consta de más de dos trozos, existe el comando **Which[condición1,proceso1,...,condiciónh,procesoh]** que asigna el **procesoj** si se cumple la condición **concicionj**. Por ejemplo, la función

```
In[28]:= Clear[f, x]
         f[x_] :=
          Which[x ≤ -3, 1, -3 < x < -1, x^2 / 9, -1 ≤ x < 1, x^3 / 3, x ≥ 1, 1 / 3]
```

tiene el siguiente aspecto:

```
In[30]:= Plot[f[x], {x, -4, 4}]
```




```
Out[30]= - Graphics -
```

Ejercicio

Defina la función $h(x) = \begin{cases} 3x^3 + x^2 + 1, & \text{si } x \le 1 \\ x, & \text{si } 1 < x < 3 \\ x^2, & \text{si } x \ge 3 \end{cases}$ y obtenga su gráfica en el intervalo $[-2, 4]$.

Álgebra lineal con *Mathematica*

■ Resumen de comandos

{□,□}=$\begin{pmatrix}\square\\ \square\end{pmatrix}$ *vector columna*

{□,□}=(□ □) *vector fila*

{{□,□},{□,□}}=$\begin{pmatrix}\square & \square\\ \square & \square\end{pmatrix}$ *matriz* { *para añadir filas pulse* CTRL↵ *para añadir columnas pulse* CTRL,

Table[expresión,{contador,inicio,final}] *crea una lista*

Table[expresión,{contador,inicio,final,paso}] *crea una lista*

Table[□,{□,□,□},{□,□,□}] *crea una lista de listas*

□.□ *producto de matrices (también sirve de producto escalar)*

MatrixPower[A,n] *potencia* n-*ésima de una matriz* A.

Dot[□,□] *producto escalar de vectores*

Cross[□,□] *producto vectorial*

Tr[□] *traza de una matriz cuadrada*

Det[□] *determinante de una matriz cuadrada*

Inverse[□] *inversa de una matriz*

Transpose[□] *traspuesta de una matriz*

Eigenvalues[□] *valores propios de una matriz*

Eigenvectors[□] *vectores propios de una matriz*

NullSpace[A] *proporciona una base del espacio vectorial de soluciones del sistema* A.x=0

LinearSolve[A,b] *proporciona una solución particular del sistema* A.x=b

RowReduce[A] *obtiene una matriz escalonada equivalente a* A. *Es útil para averiguar el rango de* A

□==□ *igualdad (≠ asignación)*

Solve[ecuación==0,variable] *resuelve una ecuación polinómica en una cierta variable*

Solve[{□==□,□==□},{□,□}] *resuelve un sistema de ecuaciones polinómicas en varias variables*

NSolve[□==□,□] *proporciona un valor numérico para una ecuación polinómica en una variable*

■ Listas, vectores y matrices

Listas y operaciones entre listas

Las listas son conjuntos de elementos ordenados, que son tratados como una entidad.

Estos elementos pueden ser cualquier objeto de *Mathematica* : números, nombres, expresiones algebraicas, funciones, otras listas, etc. La estructura de una lista es: **lista={elemento elemento*2* , ... , elemento*n*}.** Es decir, estos objetos se introducen entre llaves y separados comas, como se muestra en los siguientes ejemplos:

```
In[1]:= lista1 = {0, Pi / 2, Pi, 3 Pi / 2} ;  lista2 = {1, x, x^2};
```

Con las listas se pueden realizar operaciones aritméticas y pueden ser el argumento de una fun ción.

Por ejemplo:

```
In[2]:= Cos[lista1]
        Exp[lista2]
Out[2]= {1, 0, -1, 0}
Out[3]= {e, e^x, e^x^2}
```

que da como resultado una lista obtenida al aplicar la función a cada elemento de la lista.

Además existen distintos tipos de manipulaciones específicas de listas: extraer o añadir elemen tos; reordenar, contar, seleccionar y buscar elementos de una lista, combinar listas, etc. Por ejem plo:

Length[lista]: Da el número de elementos de la lista

Take[lista,n]: Da una nueva lista con los n primeros elementos de la lista.

Insert[lista,a,n]: Inserta el elemento **a** en la posición **n** de la lista.

Complement[lista1,lista2]: Da los elementos de la lista1 que no están en la lista2

Join[lista1,lista2]: Crea una nueva lista juntando las dos anteriores una detrás de otra

Union[lista1,lista2]: Une las listas sin repetir elementos que estén en las dos a la vez

Intersection[lista1,lista2]: Crea una nueva lista con los elementos que están en las dos a la vez

Flatten[lista]: Elimina sublistas y crea una sola lista con todos sus elementos

Por ejemplo:

```
In[4]:= Length[lista1]
        nuevalista1 = Take[lista1, 3]
        nuevalista2 = Insert[lista2, 0, 1]
        l1menosl2 = Complement[lista1, nuevalista2]
        lista3 = Join[lista1, nuevalista2]
        l12 = Union[lista1, nuevalista2]
        l1il2 = Intersection[lista1, nuevalista2]
        Flatten[{{a, b}, {c, d}, {{e}}}]
Out[4]= 4
Out[5]= {0, π/2, π}
Out[6]= {0, 1, x, x^2}
Out[7]= {π/2, π, 3 π/2}
Out[8]= {0, π/2, π, 3 π/2, 0, 1, x, x^2}
Out[9]= {0, 1, π/2, π, 3 π/2, x, x^2}
Out[10]= {0}
Out[11]= {a, b, c, d, e}
```

Construcción de vectores y matrices

Un vector **v** sera una lista de números, mientras que una matriz **m** será una lista de vectores (osea, una listas de listas). Por ejemplo:

```
In[12]:= v = {1, 2}
         m = {{1, 3}, {6, 9}}
Out[12]= {1, 2}
Out[13]= {{1, 3}, {6, 9}}
```

Para ver las matrices en su forma usual escribiremos:

```
In[14]:= MatrixForm[m]
Out[14]//MatrixForm=
         1  3
         6  9
```

Lo que nos dice que **m={{1,3},{6,9}}** es una lista de vectores dispuestos por FILAS. Para extraer una fila o un elemento de una matriz se utiliza el nombre de la matriz seguido de un doble cor chete que encierra el número de fila, o el número de fila y columna del elemento, por ejemplo:

```
In[15]:= m[[2]]
         m[[1, 2]]
Out[15]= {6, 9}
Out[16]= 3
```

nos da la segunda fila de la matriz **m** y el elemento de la fila 1 columna 2 de la matriz **m**

Para crear vectores y matrices genéricos podemos utilizar el comando **Array[w,n]**, que crea vector **w** de **n** componentes, y

Array[a,{p,q}], que crea una matriz de **p** filas y **q** columnas. Por ejemplo:

```
In[17]:= pepa = Array[a, {3, 2}]
Out[17]= {{a[1, 1], a[1, 2]}, {a[2, 1], a[2, 2]}, {a[3, 1], a[3, 2]}}
```

Nota: El nombre de la matriz, **pepa**, debe ser distinto del argumento de Array, en este caso **a.**

Por otro lado, la orden de tipo iterativo:

Table[expresión,{contador,inicio,fin,paso},{contador2,inicio2,fin2,paso2}]

crea una lista o vector con sus elementos dados por expresión y tantos como valores tome contador desde el valor inicio hasta el valor fin, con incrementos iguales al paso indicado (si se especifica el incremento, el programa entiende que es 1). Por ejemplo, vamos a crear un vector **vec** de $\mathbb{R}^6$ cuyas componentes **vec[[i]]=i^3** sean los cubos de los primeros 6 números enteros:

```
In[18]:= vec = Table[i^3, {i, 1, 6}]
Out[18]= {1, 8, 27, 64, 125, 216}
```

También podemos crear una matriz **mima** 3 por 3 cuyas componentes **mima[[i,j]]=** $i^2 + j^2$. ello escribimos:

```
In[19]:= mima = Table[i^2 + j^2, {i, 1, 3}, {j, 1, 3}]; MatrixForm[mima]
Out[19]//MatrixForm=
( 2   5   10 )
( 5   8   13 )
( 10  13  18 )
```

Existen sentencias que permiten construir de forma rápida algunos tipos especiales de matrices, por ej:

DiagonalMatrix[{d_1, d_2, ..., d_n}] da una matriz diagonal con la diagonal principal : d_1, ..., d_n

IdentityMatrix[n] da la matriz identidad de orden **n×n**.

Ejercicio

Crear las siguientes matrices:

a) **m[[i,j]]=i*j,** i,j=1,...4

b) **m[[i,j]]=Cos[π/i]Sin[π/j],** i,j=1,...5

c) La matriz diagonal de valores propios π , $\pi/2$,$\pi/3$ y 7

d) La matriz identidad 3×3

■ Operaciones con vectores y matrices

La suma y producto de vectores o matrices, siempre que sus dimensiones lo permitan, se efectú mediante los operadores : " **+**" y "**.**" respectivamente. En el caso de dos vectores de igual dimen sión, el operador " **.**" obtiene el producto escalar de ambos, en una base ortonormal. Por ejemplo:

```
In[20]:= Clear[v1, v2, mat1, mat2];
         v1 = {a, b};  v2 = {c, d};  mat1 = {{u, w}, {x, y}};  mat2 = {{1, 2}, {3, 4}};
         v1 + v2
         MatrixForm[mat1 + mat2]
         v1.v2
         mat1.v1
         MatrixForm[mat1.mat2]
Out[22]= {a + c, b + d}
Out[23]//MatrixForm=
         (1 + u   2 + w)
         (3 + x   4 + y)
Out[24]= a c + b d
Out[25]= {a u + b w, a x + b y}
Out[26]//MatrixForm=
         (u + 3 w   2 u + 4 w)
         (x + 3 y   2 x + 4 y)
```

Para multiplicar un escalar por un vector o por una matriz se utiliza el asterisco " ***** ", o se deja un espacio en blanco entre ellos: a*mat1 o a ma1. Si el escalar es un número no es necesario dejar un espacio en blanco, si es un parámetro (una letra) si es necesario el espacio.

Para vectores tridimensionales es posible realizar el producto vectorial **v×u**, mediante la orden: **Cross[v,u]**, así:

```
In[27]:= Cross[{a1, a2, a3}, {b1, b2, b3}]
Out[27]= {-a3 b2 + a2 b3, a3 b1 - a1 b3, -a2 b1 + a1 b2}
```

Por otra parte,

```
In[28]:= Transpose[mat1]
Out[28]= {{u, x}, {w, y}}
```

obtiene la MATRIZ TRASPUESTA m^t, y

```
In[29]:= Det[mat1]
Out[29]= -w x + u y
```

nos da el DETERMINANTE **|m|,** mientras que

```
In[30]:= Tr[mat1]
Out[30]= u + y
```

nos da la TRAZA. La MATRIZ INVERSA se obtiene como:

```
In[31]:= imat1 = Inverse[mat1]; MatrixForm[imat1]
General::spell1 : Possible spelling error: new symbol
   name "imat1" is similar to existing symbol "mat1".
Out[31]//MatrixForm=
```

$$\begin{pmatrix} \frac{y}{-w x+u y} & -\frac{w}{-w x+u y} \\ -\frac{x}{-w x+u y} & \frac{u}{-w x+u y} \end{pmatrix}$$

Para hallar la POTENCIA n-ESIMA de una matriz m se escribe **MatrixPower[m,n] .** Por ejemplo:

```
In[32]:= MatrixForm[MatrixPower[mat1, 3]]
Out[32]//MatrixForm=
```

$$\begin{pmatrix} u\,(u^2 + w x) + w\,(u x + x y) & u\,(u w + w y) + w\,(w x + y^2) \\ x\,(u^2 + w x) + y\,(u x + x y) & x\,(u w + w y) + y\,(w x + y^2) \end{pmatrix}$$

Si la matriz es diagonalizable, *Mathematica* sabe cómo calcular la potencia n-ésima. Por ejemplo:

```
In[33]:= poten = MatrixPower[{{0.1, 0.3}, {0.9, 0.7}}, n]; MatrixForm[poten]
Out[33]//MatrixForm=
```

$$\begin{pmatrix} 0.75\,(-0.2)^n + 0.25\,1.^n & -0.25\,(-0.2)^n + 0.25\,1.^n \\ -0.75\,(-0.2)^n + 0.75\,1.^n & 0.25\,(-0.2)^n + 0.75\,1.^n \end{pmatrix}$$

Luego podemos hacer el límite cuando n->∞ escribiendo

```
In[34]:= Limit[poten, n → Infinity]
Out[34]= {{0.25, 0.25}, {0.75, 0.75}}
```

Ejercicio

Dados los vectores: v1= (1,3,2) y v2 = (0,1,-1) y las matrices

$a1=\begin{pmatrix} 0.1 & 0.2 & 0.4 \\ 0.2 & 0.5 & 0.2 \\ 0.7 & 0.3 & 0.4 \end{pmatrix}$, $a2=\begin{pmatrix} -1 & 2 & 0 \\ 4 & 5/2 & \pi \\ 0 & 2/3 & 6 \end{pmatrix}$, Obtener:

a) 2v1-4v2, v1.v2, v1×v2, y el ángulo α que determinan v1 y v2. (recordad que: $\cos\alpha = \frac{v1}{|v1}$; norma euclídea: $|v1|=\sqrt{v1.v1}$)

b) 2a1+a2, a1.a2, v2. a2.v1, $(a1.a1^t)^{-1}$, $a1^{\infty}$, $a2^{\infty}$

Rango de una matriz

Recordemos que el rango de una matriz es el el rango del sistema formado por sus vectores fila (o columna).

Para obtener una matriz escalonada equivalente a una dada , disponemos del comando:

RowReduce[m]

que obtiene una matriz escalonada reducida equivalente a la matriz **m**. Por ejemplo:

```
In[35]:= MatrixForm[RowReduce[{{1, 2}, {2, 4}}]]
Out[35]//MatrixForm=
( 1  2 )
( 0  0 )
```

que nos indica que el rango es 1

Ejercicio:

Obtener el rango de : $F=\begin{pmatrix} 1 & -2 & -1 & 2 & 1 \\ 0 & 1 & 3 & -3 & 2 \\ -2 & 0 & 1 & 2 & -1 \\ -3 & -1 & 4 & 3 & 1 \\ -4 & -5 & -2 & 13 & -3 \end{pmatrix}$

Bases ortonormales. Método de Gram-Schmidt

Mathematica tiene definido un comandos que permiten buscar bases ortonormales subespacios a partir de bases no ortonormales. Para ello deberemos cargar previamente el paquete

```
In[36]:= << LinearAlgebra`Orthogonalization`
```

Una vez hecho esto, podemos escribir:

```
In[37]:= {o1, o2} = GramSchmidt[{{1, 2, 0}, {3, 4, 5}}]
         o1.o2
         o1.o1
         o2.o2
```

$$\text{Out[37]=}\ \left\{\left\{\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}}, 0\right\}, \left\{\frac{4}{\sqrt{645}}, -\frac{2}{\sqrt{645}}, 5\sqrt{\frac{5}{129}}\right\}\right\}$$

```
Out[38]= 0
Out[39]= 1
Out[40]= 1
```

que nos da una base ortonormal del plano generado por {1,2,0} y {3,4,5}. (la segunda linea comprueba que son perpendiculares y la tercera y cuarta que tienen módulo 1)

Ejercicio

a) Sea el espacio geométrico $\mathbb{R}^3$, dotado de un sistema de referencia rectangular (base ortonormal), obtener un nuevo sistema de referencia rectangular de forma que los dos primeros ejes de dicho sistema estén en el plano de ecuación : $x + y + z = 0$. Obtener una base del plano, ampliarla a una base de $\mathbb{R}^3$, cuyos dos primeros vectores sean del plano, y aplicar el método de Gram-Schmidt

b) Considerando el producto escalar canónico de $\mathbb{R}^5$. Calcula una base ortonormal del subespacio generado por F a partir de la base obtenida en el ejercicio anterior y comprueba que la nueva base es ortonormal.

■ Sistemas de ecuaciones lineales

Un sistema de m ecuaciones lineales con n incógnitas lo escribimos en forma matricial: **A.x = b**

Donde **A** es la matriz de coeficientes de m filas y n columnas, **x** es el vector columna incógnita de n-componentes y **b** es el vector columna término independiente de m componentes. Mathematica dispone de ordenes que permiten resolver matricialmente los sistemas de ecuaciones lineales.

Sistemas homogéneos

El comando **NullSpace[A]** proporciona una base del núcleo de la aplicación lineal de matriz **A**, es decir, una base del subespacio de las soluciones del sistema homogéneo: **A.x = 0**, cuya matriz de coeficientes es **A**. Por ejemplo, dado el sistema

$$\begin{aligned} x + 2y + 3z + t &= 0 \\ 4x + 5y + 6z + t &= 0 \\ 7x + 8y + 9z + t &= 0 \\ 2x + y - t &= 0 \end{aligned}$$

, su matriz de coeficientes se escribe:

```
In[41]:= mc = {{1, 2, 3, 1}, {4, 5, 6, 1}, {7, 8, 9, 1}, {2, 1, 0, -1}}
Out[41]= {{1, 2, 3, 1}, {4, 5, 6, 1}, {7, 8, 9, 1}, {2, 1, 0, -1}}
```

y la familia de vectores

```
In[42]:= fami = NullSpace[mc]
Out[42]= {{1, -1, 0, 1}, {1, -2, 1, 0}}
```

genera un subespacio vectorial solución de dimensión dos. Es decir, cualquier solución xh de mc.xh=b viene dada como combinación lineal de los dos vectores de la familia como:

```
In[43]:= xh = α fami[[1]] + β fami[[2]]
Out[43]= {α + β, -α - 2 β, β, α}
```

Ejercicio

Proporcione la solución general del sistema homogéneo:

$$\begin{aligned} x + 2y + 3z + t &= 0 \\ 4x + 5y + 6z + t &= 0 \\ 5x + 7y + 9z + 2t &= 0 \\ 3x + 3y + 3z &= 0 \end{aligned}$$

Sistemas no homogéneos

El comando **LinearSolve[A,b]** devuelve un vector x que es una *solución particular* de la ecuación matricial : **A.x=b**. Si existen mas soluciones, LinearSolve no lo indica, por lo que tendremos que comprobarlo nosotros. Si el sistema no tiene solución *Mathematica* nos lo dirá.

Como el conjunto de soluciones del sistema no homogéneo **A.x = b** es de la forma **xnh = xp+xh**, donde **xp** es una solución particular de dicho sistema y **xh** es la solución general del sistema homogéneo **A.x = 0 .**

Si el sistema **A.x = b** es compatible indeterminado la solución general del sistema se obtiene sumando al subespacio de las soluciones del homogéneo la solución particular dada por LinearSolve.

Cuando los sistemas dependen de ciertos parámetros α, β, ..., las anteriores instrucciones nos dan una idea de cuáles son los valores problemáticos de dichos parámetros para los cuales el sistema cambia su carácter (compatible, incompatible, determinado o indeterminado). En efecto, sea el sistema $\begin{cases} \alpha x + \beta y = \gamma \\ x + y = 1 \end{cases}$

```
In[44]:= Clear[A, b]; A = {{α, β}, {1, 1}}; b = {γ, 1};
         LinearSolve[A, b]
```

$$\text{Out[45]= } \left\{\left\{\frac{-\beta+\gamma}{\alpha-\beta},\ \frac{\alpha-\gamma}{\alpha-\beta}\right\}\right\}$$

La solución particular está definida salvo que $\alpha=\beta\neq\gamma$, donde se anulan los denominadores. Estidiando cada caso por separado:

```
In[46]:= LinearSolve[A /. {α -> β}, b]
         LinearSolve[A /. {α -> β}, b /. {γ -> β}]

         LinearSolve::nosol :
          Linear equation encountered which has no solution.

Out[46]= LinearSolve[{{β, β}, {1, 1}}, {γ, 1}]

Out[47]= {1, 0}
```

que nos indica que, para $\alpha=\beta\neq\gamma$, el sistema no tiene solución; mientras que para $\alpha=\beta=\gamma$ el sistema es compatible y (1,0) es una solución particular.

Ejercicio

a) Sea m={{1,2,3},{4,5,6},{7,8,9}} ¿Tiene solución el sistema: m.x = n. para n=(1, 1, 1) y para n=(1, 1, 2)?. Dar la solución general de los sistemas que sean compatibles indeterminados. ¿ (1,-3,2) es solución del sistema ? ¿ (1,-3,1) es solución del sistema ?

b) Estudiar el carácter del sistema A.x=b para A= $\begin{pmatrix} \alpha & \beta & 1 \\ 1 & \alpha\,\beta & 1 \\ 1 & \beta & \alpha \end{pmatrix}$ y b= $\begin{pmatrix} 1 \\ \beta \\ 1 \end{pmatrix}$ en función de los parámetros α y β.

■ Cambio de base

Sean B= $\{e_1,e_2,...,e_n\}$ y B'= $\{u_1,u_2,...,u_n\}$ dos bases de $\mathbb{R}^n$. Dadas las coordenadas de los vectores de la base B' en la base B:

$$\begin{aligned}
(u_1)_B &= (u_{11}, u_{21}, .., u_{n1}) \leftrightarrow u_1 = u_{11}\,e_1 + u_{21}\,e_2 + \ldots + u_{n1}\,e_n \\
(u_2)_B &= (u_{12}, u_{22}, .., u_{n2}) \leftrightarrow u_2 = u_{12}\,e_1 + u_{22}\,e_2 + \ldots + u_{n2}\,e_n \\
&\ldots\ldots \qquad\qquad \ldots\ldots\ldots\ldots\ldots \\
(u_n)_B &= (u_{1n}, u_{2n}, .., u_{nn}) \leftrightarrow u_n = u_{1n}\,e_1 + u_{2n}\,e_2 + \ldots + u_{nn}\,e_n
\end{aligned}$$

podemos obtener las coordenadas de un vector $v_B = (x_1,x_2,..,x_n) \longleftrightarrow v = x_1e_1+x_2e_2+...+x_ne_n$ en la base B a partir de las coordenadas $v_{B'} = (x'_1,x'_2,..,x'_n) \longleftrightarrow v = x'_1u_1+x'_2u_2+...+x'_nu_n$ de dicho vector en la base B' de la siguiente forma:

$v = x_1e_1+x_2e_2+...+x_ne_n = x'_1u_1+x'_2u_2+...+x'_nu_n =$

$x'_1 (u_{11}\,e_1 + u_{21}\,e_2 + \ldots + u_{n1}\,e_n) +$

$x'_2(u_{12}\,e_1 + u_{22}\,e_2 + \ldots + u_{n2}\,e_n)+...$

$+x'_n\,(u_{1n}\,e_1 + u_{2n}\,e_2 + \ldots + u_{nn}\,e_n) \longrightarrow$

$$\begin{aligned}
x_1 &= x'_1u_{11} + x'_2\,u_{12} + ... + x'_n\,u_{1n} \\
x_2 &= x'_1u_{21} + x'_2\,u_{22} + ... + x'_n\,u_{1n} \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
x_n &= x'_1u_{n1} + x'_2\,u_{n2} + ... + x'_n\,u_{nn}
\end{aligned}$$

o matrialmente:

$$\begin{pmatrix} u_{11} & u_{12} & \ldots & u_{1n} \\ u_{21} & u_{22} & \ldots & u_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ u_{n1} & u_{n2} & \ldots & u_{nn} \end{pmatrix} \begin{pmatrix} (x')_1 \\ (x')_2 \\ \ldots \\ (x')_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix} \longrightarrow \quad C = \begin{pmatrix} u_{11} & u_{12} & \ldots & u_{1n} \\ u_{21} & u_{22} & \ldots & u_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ u_{n1} & u_{n2} & \ldots & u_{nn} \end{pmatrix}$$

C es la matriz de cambio de base de la base B' a la base B.

Ejercicio

Sea B la base canónica de $\mathbb{R}^4$. Comprobar que B' = $\{(u_1)_B = (1,-2,-1,2)$, $(u_2)_B=(0,1,3,-3)$, $(u_3)_B=(-2,0,1,2)$, $(u_4)_B=(1,-1,1,3)\}$

es otra base. Obtener las coordenadas en la base B del vector $v_{B'} = (9,8,-3,4)$ y la matriz $M_{BB'}$ cambio de base de la base B a la base B'. Dar las coordenadas en la base B' del vector $w_B = (19,38,-38,19)$ en la base canónica.

■ Valores y vectores propios. Diagonalización de matrices.

Definición: Sea f una aplicación lineal $f:\mathbb{R}^n \longrightarrow \mathbb{R}^n$. Si $v \neq 0$ y $f(v) = \lambda v$, se dice que *v es un vector propio de f asociado al valor propio* λ.

Si $B= \{e_1,e_2,...,e_n\}$ es una base de $\mathbb{R}^n$, y $v_B = (x_1,x_2,..,x_n) \longleftrightarrow v = x_1e_1+x_2e_2+...+x_ne_n$ las coordenadas de un vector v en dicha base, entonces las coordenadas de la imagen w de v por f vienen dadas por:

$$w = (f(v))_B =(f(x_1e_1+x_2e_2+...+x_ne_n))_B= x_1(f(e_1))_B+ x_2 (f(e_2))_B +...+ x_n (f(e_n))_B$$

o matricialmente: $w_B=(f(v))_B = M\, v_B$ donde $M = \{(f(e_1))_B , (f(e_2))_B ,...., (f(e_n))_B\}$ es la matriz de la aplicación f en la base B, sus columnas son las imágenes mediante f de los vectores de la base B.

Sea M una matriz cuadrada de orden n con elementos reales. Si $v \neq 0$ y $M.v = \lambda v$, se dice que *v es un vector propio de M asociado al valor propio* λ.

El endomorfismo f es diagonalizable si existe una base B' de $\mathbb{R}^n$, en la cual la matriz M de f es diagonal, es decir, si existe una base de $\mathbb{R}^n$ formada por vectores propios de f .

Si M es la matriz de la aplicación f en la base B y M' es la matriz de la aplicación f en la base B' y C es la matriz cambio de base de la base B' a la base B, entonces: $M' = C^{-1}.M.C$. En particular, si M es diagonalizable y si B' es la la base de vectores propios, entonces $M'=D = P^{-1}.M.P$, donde D es la matriz diagonal y P es la matriz de paso.

Mathematica dispone del comando

```
In[48]:= s = {{1, 2}, {2, 1}};
         Eigenvalues[s]
Out[49]= {-1, 3}
```

que devuelve una lista con todos los valores propios de la matriz **s**. Por otra parte

```
In[50]:= Eigenvectors[s]
Out[50]= {{-1, 1}, {1, 1}}
```

devuelve una lista con un sistema de vectores propios linealmente independientes de la matriz **s**. Las matrices diagonal y de paso se pueden escribir como

```
In[51]:= diag = DiagonalMatrix[Eigenvalues[s]]; MatrixForm[diag]
         paso = Transpose[Eigenvectors[s]]; MatrixForm[paso]
```

Out[51]//MatrixForm=

$$\begin{pmatrix} -1 & 0 \\ 0 & 3 \end{pmatrix}$$

Out[52]//MatrixForm=

$$\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}$$

Podemos comprobar que se cumple la relación $D=P^{-1}SP$ entre la matriz diagonal D, la matriz de paso P y la matriz S que queremos diagonalizar. En efecto

```
In[53]:= MatrixForm[Inverse[paso].s.paso]
```

Out[53]//MatrixForm=

$$\begin{pmatrix} -1 & 0 \\ 0 & 3 \end{pmatrix}$$

Si el número de vectores propios linealmente independientes es menor que la dimensión, *Mathematica* devuelve el vector nulo. Esto quiere decir que la matriz *no es diagonalizable.* Por ejemplo.

```
In[54]:= ts = {{1, 2}, {0, 1}};
         Eigenvalues[ts]
         Eigenvectors[ts]
Out[55]= {1, 1}
Out[56]= {{1, 0}, {0, 0}}
```

El valor propio 1 es doble y sólo existe un vector propio l.i. asociado a él.

También podemos estudiar el carácter diagonalizable de una matriz dependiente de ciertos parámetros a,b,c,... como:

```
In[57]:= Clear[a, b, c]; Eigenvectors[{{a, 1}, {c, 1}}]
```

Out[57]= $\left\{\left\{-\frac{1-a+\sqrt{1-2a+a^2+4c}}{2c}, 1\right\}, \left\{-\frac{1-a-\sqrt{1-2a+a^2+4c}}{2c}, 1\right\}\right\}$

mirando dónde se anulan los denominadores. En efecto, un caso potencialmente problemático es *c=0.* Un estudio más detallado para este valor:

```
In[58]:= Eigenvectors[{{a, 1}, {c, 1}} /. {c -> 0}]
Out[58]= {{-1/(-1+a), 1}, {1, 0}}
```

nos dice que la matriz es diagonalizable para *c=0*, a no ser que *a=1*, ya que:

```
In[59]:= Eigenvectors[{{a, 1}, {c, 1}} /. {c -> 0, a -> 1}]
Out[59]= {{1, 0}, {0, 0}}
```

para estos valores obtenemos un vector propio nulo, lo que quiere decir que la matriz no es diagonalizable.

Ejercicio

¿Para qué valores de a,b y c la matriz $\begin{pmatrix} 1 & a & 0 \\ 0 & 1-a & b \\ 0 & 0 & c \end{pmatrix}$ no es diagonalizable?

■ Resolución de ecuaciones polinómicas

En *Mathematica* la igualdad se escribe con dos signos: == , para distinguirla de la asignación. Por ejemplo si escribimos x=3 , le estamos asignando a x el valor 3, de forma que si escribiesemos

```
In[60]:= x = 3; x = x + 5
Out[60]= 8
```

estaríamos asignando a x el valor 8(=3+5). Así, la ecuación $x^4 - 5x^3 + 5x^2 + 5x - 6 = 0$ se escribe:

```
In[61]:= Clear[x];
         x^4 - 5 x^3 + 5 x^2 + 5 x - 6 == 0
Out[62]= -6 + 5 x + 5 x^2 - 5 x^3 + x^4 == 0
```

Para conocer las raíces de esta ecuación se escribe **Solve[ecuación , variable].** Por ejemplo:

```
In[63]:= Solve[x^4 - 5 x^3 + 5 x^2 + 5 x - 6 == 0, x]
Out[63]= {{x → -1}, {x → 1}, {x → 2}, {x → 3}}
```

Cuando el polinomio es de grado alto, *Mathematica* puede buscar un valor numérico aproximado para la solución con **NSolve[ecuación , variable]**. Por ejemplo:

```
In[64]:= NSolve[x^6 - x^3 + 50 x^2 - 7 == 0, x]
Out[64]= {{x → -1.87201 - 1.92491 i}, {x → -1.87201 + 1.92491 i}, {x → -0.372708},
           {x → 0.375503}, {x → 1.87061 - 1.85425 i}, {x → 1.87061 + 1.85425 i}}
```

Para un sistema de ecuaciones, se introducen como una lista **Solve[{ec$_1$, ec$_2$, ..., ec$_n$}, {var$_1$, var$_2$, ..., var$_n$}]**. Por ejemplo:

```
In[65]:= Solve[{x^2 + y^2 == 1, y == x}, {x, y}]
Out[65]= {{x → -1/Sqrt[2], y → -1/Sqrt[2]}, {x → 1/Sqrt[2], y → 1/Sqrt[2]}}
```

nos da la intersección (dos puntos) de una circunferencia de radio 1 con la recta y=x.

Ejercicio

a) Obtener las raíces de: $-\frac{16}{3} + \frac{32x}{3} + \frac{8x^2}{3} - \frac{64x^3}{3} + \frac{59x^4}{3} - \frac{22x^5}{3} + x^6 = 0$, $x^{10}-x^3+1=0$

b) Obtener la intersección de la elipse $\frac{x^2}{2} + \frac{y^2}{\quad} = 1$ con la circunferencia de radio $\sqrt{2}$

Gráficos con *Mathematica*

■ Resumen de comandos

CURVAS EN EL PLANO

Plot[f(x),{x,xmin,xmax}] *dibuja la función y=f(x) en el intervalo especificado*

Plot[{f1(x),f2(x)},{x,xmin,xmax}] *dibuja superpuestas varias funciones*

ParametricPlot[{x(t),y(t)},{t,tmin,tmax}] *dibuja una curva dada en forma paramétrica*

<< Graphics`ImplicitPlot` *paquete que permite hacer gráficas de funciones definidas de forma implícita*

ImplicitPlot[F[x,y]==0,{x,xmax,xmin}] *dibuja la curva F(x,y)=0*

ImplicitPlot[F[x,y]==0,{x,xmax,xmin},{y,ymax,ymin}] *dibuja la curva de nivel z=F(x,y)=0*

ListPlot[lista] *dibuja puntos a partir de una tabla (lista) de valores {{x1,y1},...}*

ListPlot[lista,PlotJoined→True] *junta los puntos de la lista con segmentos*

<< Graphics`Graphics`

ErrorListPlot[lista] *dibuja puntos de una tabla de valores {{x1,y1,error1},...} con barras de error*

<<Graphics`MultipleListPlot`

MultipleListPlot[lista] *dibuja puntos de una tabla de valores* **{{x1,y1,ErrorBar[e1min,e1max]},...}** *con barras de error especificadas por la opción ErrorBar en cada punto.*

CURVAS Y SUPERFICIES EN EL ESPACIO

Plot3D[f(x,y),{x,xmin,xmax},{y,ymin,ymax}] *dibuja una superficie z=f(x,y) (explícita) en un rectángulo*

ParametricPlot3D[{x(u,v),y(u,v),z(u,v)},{u,umin,umax},{v,vmin,vmax}] *dibuja una superficie en forma paramétrica*

<<Graphics`SurfaceOfRevolution` *carga el paquete para dibujar superficies de revolución*

SurfaceOfRevolution[f[x],{x,xmin,xmax},RevolutionAxis[x,y]] *dibuja la superficie de revolución generada*

al girar la curva y=f(x) respecto a los ejes coordenados.

ParametricPlot3D[{x(t),y(t),z(t)},{t,tmin,tmax}] *dibuja una curva en forma paramétrica*

ListPlot3D[lista] *dibuja un conjunto de puntos en una lista*

CURVAS DE NIVEL

ContourPlot[f(x,y),{x,xmin,xmax},{y,ymin,ymax}] *dibuja las curvas de nivel de z=f(x,y) en un rectángulo*

ListContourPlot[lista] *dibuja las curvas de nivel para un conjunto de puntos dados por una lista*

DensityPlot[f(x,y),{x,xmin,xmax},{y,ymin,ymax}] *dibuja un gráfico de densidades*

ListDensityPlot[lista] *dibuja un gráfico de densidades para un conjunto de puntos dados por una lista*

CAMPOS DE VECTORES EN EL PLANO

<<Graphics`PlotField` *carga el paquete correspondiente*

PlotVectorField[{vx,vy},{x,xmin,xmax},{y,ymin,ymax}] *dibuja un campo de vectores en el plano*

ListPlotVectorField[{{v11,...,v1n},...,{vm1,...,vmn}}] *dibuja los vectores vij en el punto (i,j) de una malla m×n*

ListPlotVectorField[{{p1,v1},...,{pn,vn}}] *dibuja los vectores vj en los puntos pj*

PlotGradientField[f(x,y),{x,xmin,xmax},{y,ymin,ymax}] *dibuja el campo de gradientes de una función escalar z=f(x,y) en un rectángulo* {x,xmin,xmax},{y,ymin,ymax}

CAMPOS DE VECTORES EN EL ESPACIO

<<Graphics`PlotField3D`

PlotVectorField[{vx,vy,vz},{x,xmin,xmax},{y,ymin,ymax},{z,zmin,zmax}]

ListPlotVectorField3D[{{p1,v1n},...,{pn,...,vn}}] *dibuja los vectores vj en los puntos pj*

PlotGradientField[f,{x,xmin,xmax},{y,ymin,ymax},{z,zmin,zmax}] *dibuja el campo de gradientes de una función escalar t=f(x,y,z) en un paralelepípedo* {x,xmin,xmax},{y,ymin,ymax},{z,zmin,zmax}

SUPERPOSICIÓN Y ANIMACIÓN DE GRÁFICAS

Show[g1,...,gn] *muestra superpuestos varios gráficos*

Show[GraphicsArray[{g1,...,gn}]] *muestra una colección de gráficos*

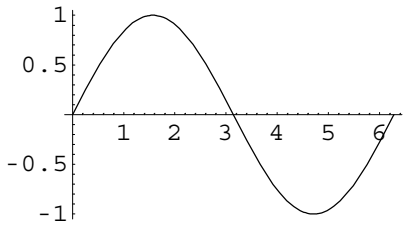
<<Graphics`Animation`

Animate[g[n],{n,nmin,nmax,paso}] *crea una animación para una familia de gráficas*

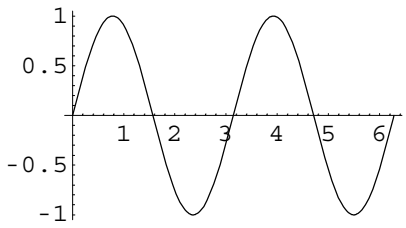
■ Curvas en el plano

Para dibujar las gráficas de las funciones *y=sen(x), y=sen(2x), y=sen(3x)* entre 0 y 2π escribiremos:

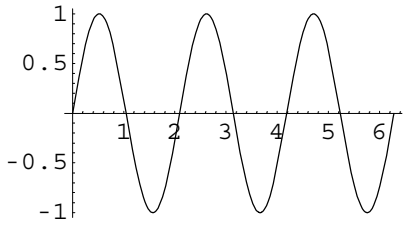
```
In[1]:= g1 = Plot[Sin[x], {x, 0, 2 π}]
        g2 = Plot[Sin[2 x], {x, 0, 2 π}]
        g3 = Plot[Sin[3 x], {x, 0, 2 π}]
```




```
Out[1]= - Graphics -
```



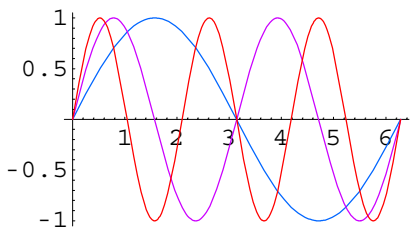

```
Out[2]= - Graphics -
```




```
Out[3]= - Graphics -
```

Podemos obtener las gráficas superpuestas, con colores ("hue") diferentes, escribiéndolas como una lista:

```
In[4]:= Plot[{Sin[x], Sin[2 x], Sin[3 x]}, {x, 0, 2 π},
         PlotStyle → {Hue[0.6], Hue[0.8], Hue[1]}]
```

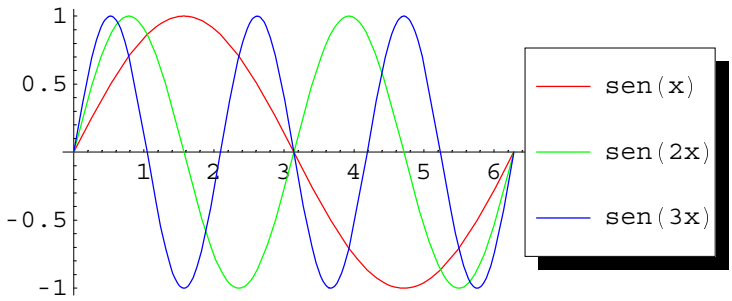



```
Out[4]= - Graphics -
```

o también

In[5]:=

```
<< Graphics`Legend`
Plot[{Sin[x], Sin[2 x], Sin[3 x]}, {x, 0, 2 π}, PlotStyle →
  {{RGBColor[1, 0, 0]}, {RGBColor[0, 1, 0]}, {RGBColor[0, 0, 1]}},
 PlotLegend -> {"sen(x)", "sen(2x)", "sen(3x)"},
 LegendPosition -> {1, -0.4}]
```



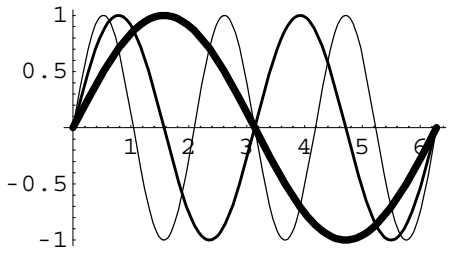
Out[6]= - Graphics -

donde la opción **PlotStile->RGBColor[r,v,a]** introduce valores numéricos asociados a los colores rojo,verde y azul respectivamente y el paquete **<<Graphics`Legend`** introduce una "leyenda" con una correspondencia entre cada color y cada gráfica a través de la opción **PlotLegend** localizada en la posición **LegendPosition**, considerando que el gráfico está centrado en **{0,0}** y que se escala para que quepa en **{{-1,-1},{1,1}}**; por defecto, la posición de la leyenda es la esquina inferior izquierda **{-1,-1}.**

Cuando no dispongamos de impresora en color, también es posible diferenciar las gráficas utilizando un grosor de linea ("thickness") diferente:

In[7]:=

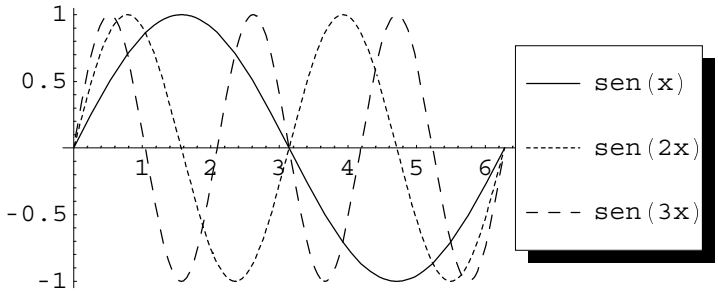
```
Plot[{Sin[x], Sin[2 x], Sin[3 x]}, {x, 0, 2 π},
 PlotStyle → {Thickness[.02], Thickness[.008], Thickness[.004]}]
```



Out[7]= - Graphics -

o gráficas punteadas ("dashing"):

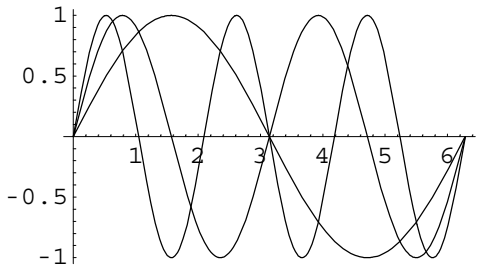
```
In[8]:= Plot[{Sin[x], Sin[2 x], Sin[3 x]}, {x, 0, 2 π},
         PlotStyle → {GrayLevel[0], Dashing[{.01}], Dashing[{.03}]},
         PlotLegend -> {"sen(x)", "sen(2x)", "sen(3x)"},
         LegendPosition -> {1, -0.4}]
```




```
Out[8]= - Graphics -
```

Si queremos superponer varias gráficas que ya hemos representado por separado, podemos utilizar el comando **Show** (muestra):

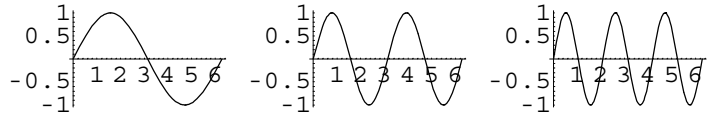
```
In[9]:= Show[{g1, g2, g3}]
```




```
Out[9]= - Graphics -
```

Si queremos juntar las gráficas en una ristra sin superponerlas, escribiremos:

```
In[10]:= Show[GraphicsArray[{g1, g2, g3}]];
```



Ademas de la opción **PlotStyle** para el comando **Plot,** existen otras opciones que podemos ver en la Ayuda ("Help") del Menu o bien escribiendo:

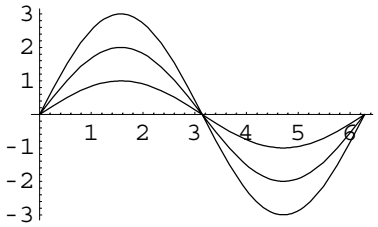
```
In[11]:= Options[Plot]

Out[11]= {AspectRatio → 1/GoldenRatio, Axes → Automatic,
  AxesLabel → None, AxesOrigin → Automatic, AxesStyle → Automatic,
  Background → Automatic, ColorOutput → Automatic, Compiled → True,
  DefaultColor → Automatic, Epilog → {}, Frame → False,
  FrameLabel → None, FrameStyle → Automatic, FrameTicks → Automatic,
  GridLines → None, ImageSize → Automatic, MaxBend → 10.,
  PlotDivision → 30., PlotLabel → None, PlotPoints → 25,
  PlotRange → Automatic, PlotRegion → Automatic, PlotStyle → Automatic,
  Prolog → {}, RotateLabel → True, Ticks → Automatic,
  DefaultFont :→ $DefaultFont, DisplayFunction :→ $DisplayFunction,
  FormatType :→ $FormatType, TextStyle :→ $TextStyle}
```

No nos detendremos aquí en el estudio de estas opciones. Sólo comentar que la opción **DisplayFunction->Identity** provoca que no se genere salida en pantalla del gráfico correspondiente; por ejemplo, si escribimos:

```
In[12]:= Clear[g1, g2, g3];
         g1 = Plot[Sin[x], {x, 0, 2 π}, DisplayFunction -> Identity];
         g2 = Plot[2 Sin[x], {x, 0, 2 π}, DisplayFunction -> Identity];
         g3 = Plot[3 Sin[x], {x, 0, 2 π}, DisplayFunction -> Identity];
```

no obtendremos dichas gráficas, aunque *Mathematica* las conserva en memoria. Si ahora queremos generar la salida en pantalla de dichas gráficas superpuestas, deberemos escribir:

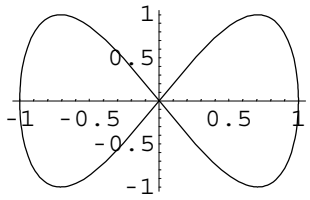
```
In[16]:= Show[{g1, g2, g3}, DisplayFunction -> $DisplayFunction]
```




```
Out[16]= - Graphics -
```

Para representar curvas planas dadas en *forma paramétrica* $\{x(\theta),y(\theta)\}$ se utiliza el comando:

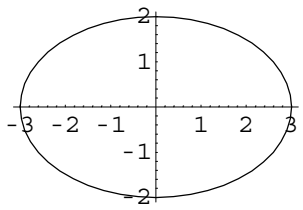
```
In[17]:= ParametricPlot[{Cos[θ], Sin[2 θ]}, {θ, 0, 2 π}]
```




```
Out[17]= - Graphics -
```

y para curvas dadas en *forma implícita F(x,y)=cte* se utiliza el paquete <<Graphics`ImplicitPlot`. Por ejemplo, para dibujar una elipse de semiejes 3 y 2 escribiremos:

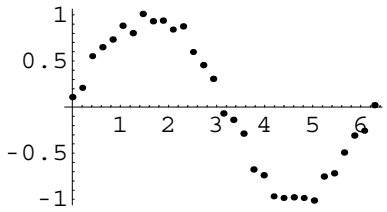
In[18]:=
```
<< Graphics`ImplicitPlot`
ImplicitPlot[x^2 / 9 + y^2 / 4 == 1, {x, -3, 3}]
```



Out[19]= - Graphics -

Si disponemos de un conjunto de puntos (quizás resultado de una medición) dados en forma de lista, podemos representarlos usando el comando **ListPlot[lista]**. Por ejemplo, creemos nosotros nuestra propia lista como una tabla de los valores de *y=f(x)* alterada por un "ruido" dado por número aleatorio ("random") en el intervalo (-0.15,015) que simula el posible error experimental en la medición de la variable **y** como función de **x***:*

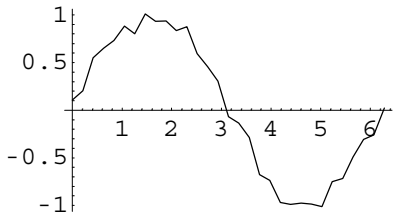
In[20]:=
```
listasenoerror =
  Table[{x, Sin[x] + Random[Real, {-0.15, 0.15}]}, {x, 0, 2 Pi, Pi/15}];
ListPlot[listasenoerror, PlotStyle → PointSize[0.02]]
```



Out[21]= - Graphics -

Podemos unir los puntos mediante segmentos utilizando la opción

In[22]:=
```
ListPlot[listasenoerror, PlotJoined → True]
```

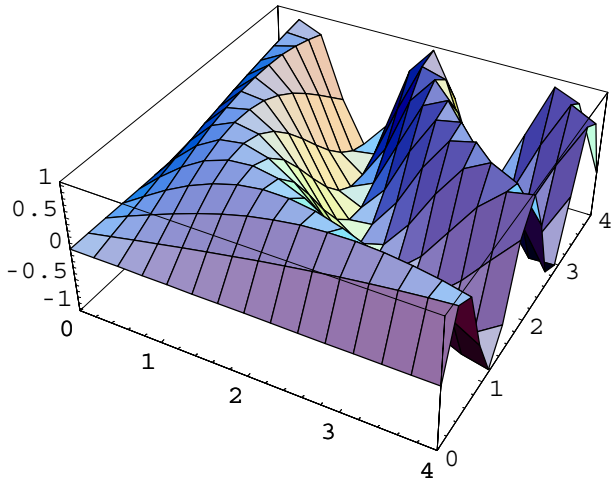


Out[22]= - Graphics -

■ Curvas y superficies en el espacio

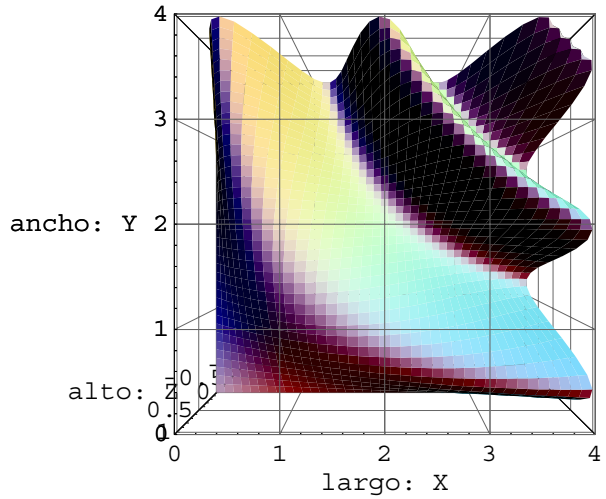
Mathematica hace gráficos tridimensionales de superficies dadas en forma explícita *z=f(x,y)* en un rectángulo dado. por ejemplo, para z=sen(xy) se tiene:

In[23]:=
```
Plot3D[Sin[x y], {x, 0, 4}, {y, 0, 4}];
```



Si queremos mas "florituras" podemos añadir opciones de estilo como:

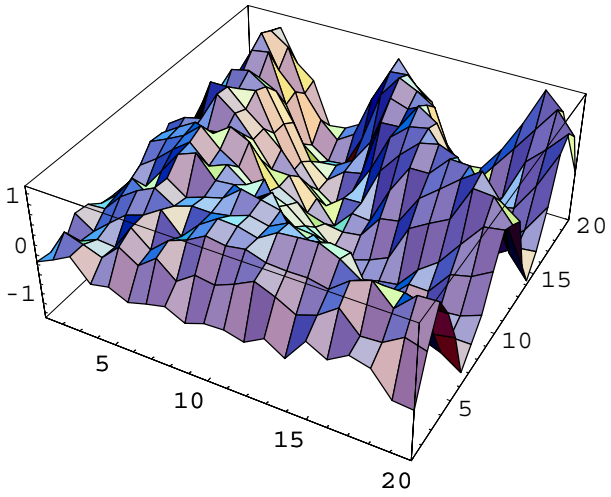
In[24]:=
```
Plot3D[Sin[x y], {x, 0, 4}, {y, 0, 4}, PlotPoints → 40, Mesh → False,
  FaceGrids → All, AxesLabel → {"largo: X", "ancho: Y", "alto: Z"},
  ViewPoint → {0, 0, 1}];
```



Que nos "suaviza" el gráfico añadiendo más puntos (**PlotPoints**), nos añade etiquetas en los ejes (**AxesLabel**) y nos modifica la orientación del gráfico (**ViewPoint**), ofreciéndonos en este caso una imagen a "vista de pájaro". Para elegir la orientación a través del menú, pinche con el ratón en "Input->3DViewPoint Selector" o presione las tres teclas ⇧+CTRL+V.

También se pueden dibujar superficies dadas por conjuntos de puntos dispuestos en una tabla de valores. Por ejemplo:

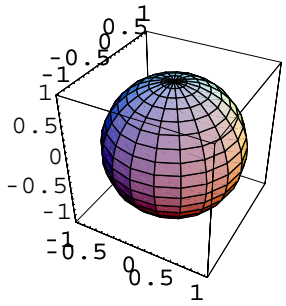
In[25]:=
```
listasenoxyerror = Table[Sin[x y] + Random[Real, {-0.3, 0.3}],
    {x, 0, 4, Pi/15}, {y, 0, 4, Pi/15}];
ListPlot3D[listasenoxyerror]
```



Out[26]= - SurfaceGraphics -

Si la superficie viene dada en forma paramétrica, *{x(u,v),y(u,v),z(u,v)}*, entonces se utiliza:

In[27]:=
```
ParametricPlot3D[
 {Cos[u] Sin[v], Sin[u] Sin[v], Cos[v]}, {u, 0, 2 Pi}, {v, 0, Pi}]
```

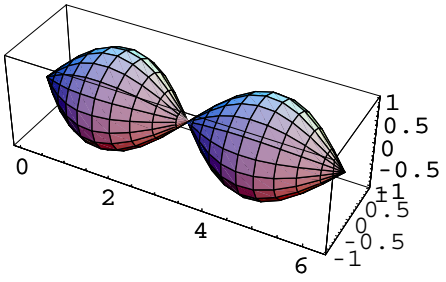


Out[27]= - Graphics3D -

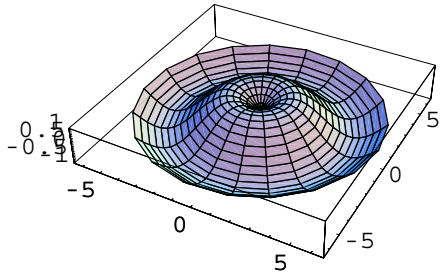
que nos da la esfera de radio 1.

También podemos crear superficies de revolución rotando una función *y=f(x)* respecto al eje X o respecto al eje Y. Por ejemplo, rotemos la gráfica *y=sen(x)* respecto a ambos ejes (es necesario cargar antes el paquete Graphics`SurfaceOfRevolution`):

In[28]:=
```
<< Graphics`SurfaceOfRevolution`
SurfaceOfRevolution[Sin[x], {x, 0, 2 Pi}, RevolutionAxis -> {1, 0}]
SurfaceOfRevolution[Sin[x], {x, 0, 2 Pi}, RevolutionAxis -> {0, 1}]
```



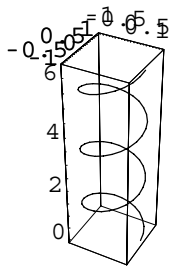
Out[29]= - Graphics3D -



Out[30]= - Graphics3D -

Para curvas en forma paramétrica se escribe:

In[31]:=
```
ParametricPlot3D[{Cos[3 t], Sin[3 t], t}, {t, 0, 2 π}];
```

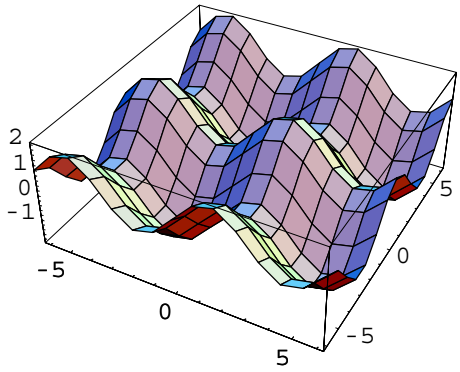


que nos da una hélice.

■ Curvas y superficies de nivel

El comando **ContourPlot** de *Mathematica* proporciona las curvas de nivel de una función escalar de dos variables *z=f(x,y)*. Por ejemplo, dada la superficie:

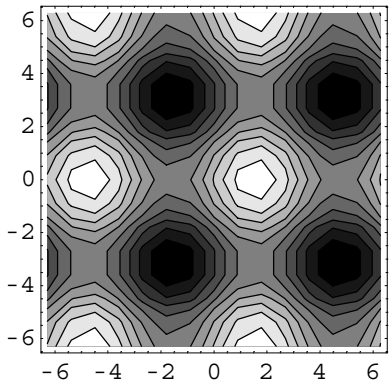
In[32]:=
```
Plot3D[Sin[x] + Cos[y], {x, -2 Pi, 2 Pi}, {y, -2 Pi, 2 Pi}]
```



Out[32]=
```
- SurfaceGraphics -
```

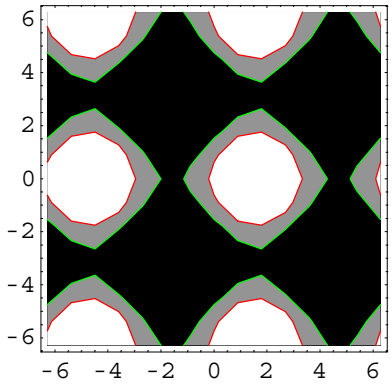
sus curvas de nivel tienen la forma:

In[33]:=
```
ContourPlot[Sin[x] + Cos[y], {x, -2 Pi, 2 Pi}, {y, -2 Pi, 2 Pi}];
```



Nótese que los máximos aparecen con un tono más claro y los mínimos con un tono más oscuro. Es fácil también identificar los puntos de inflexión a partir de la gráfica. Podemos introducir nuevas opciones, como representar sólo ciertas curvas de nivel *f(x,y)=c1, f(x,y)=c2*, para lo cual introducimos la opción **Contours->{c1,c2}**, seguida de otras como **ContourStyle** que distinguen ambas curvas por, por ejemplo, su color:

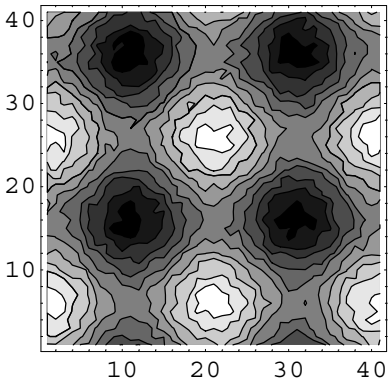
In[34]:=
```
ContourPlot[Sin[x] + Cos[y], {x, -2 Pi, 2 Pi},
 {y, -2 Pi, 2 Pi}, Contours → {.1, .8},
 ContourStyle → {{RGBColor[0, 1, 0]}, {RGBColor[1, 0, 0]}}]
```



Out[34]=
```
- ContourGraphics -
```

Si las cotas vienen dadas como puntos en una tabla (en forma de lista), entonces utilizaremos el comando:

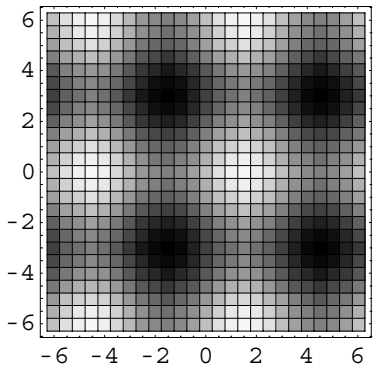
In[35]:=
```
ListContourPlot[Table[Sin[x] + Cos[ y] + Random[Real, {-0.2, 0.2}],
  {x, -2 Pi, 2 Pi, Pi / 10}, {y, -2 Pi, 2 Pi, Pi / 10}]]
```



Out[35]= - ContourGraphics -

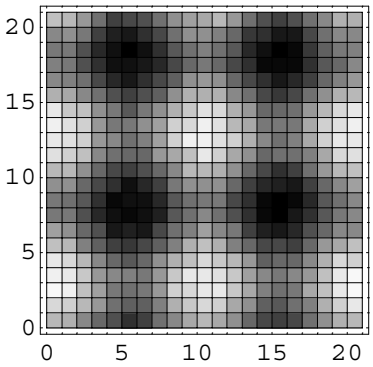
Si, en vez de curvas de nivel, queremos gráficos de densidad, entonces utilizaremos el comando:

In[36]:=
```
DensityPlot[Sin[x] + Cos[ y],
  {x, -2 Pi, 2 Pi}, {y, -2 Pi, 2 Pi}, PlotPoints -> 25];
```



donde la opción **PlotPoints** hace el granulado más fino o mas grueso. Si se trata de una lista, escribiremos:

In[37]:=
```
ListDensityPlot[Table[Sin[x] + Cos[ y] + Random[Real, {-0.2, 0.2}],
  {x, -2 Pi, 2 Pi, Pi / 5}, {y, -2 Pi, 2 Pi, Pi / 5}]];
```

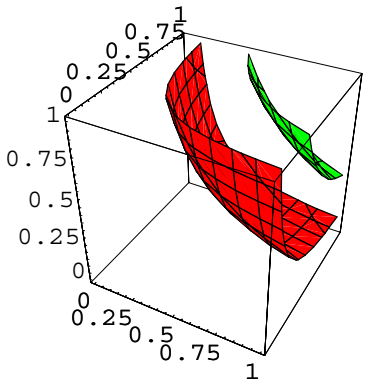


Para representar superficies de nivel *f(x,y,z)=constante* de una función *V=f(x,y,z)* de tres variables deberemos cargar previamente el paquete

In[38]:=
```
<< Graphics`ContourPlot3D`
```

Por ejemplo,el lugar geométrico de los puntos del espacio que hacen el volumen de un paralelepípedo *V=xyz=cte=0.1,0.4* puede representarse por:

```
In[39]:= ContourPlot3D[x y z, {x, 0, 1}, {y, 0, 1}, {z, 0, 1},
          Contours → {.1, .4}, Lighting → False, ContourStyle →
           {{RGBColor[0, 1, 0]}, {RGBColor[1, 0, 0]}}, Axes → True]
```



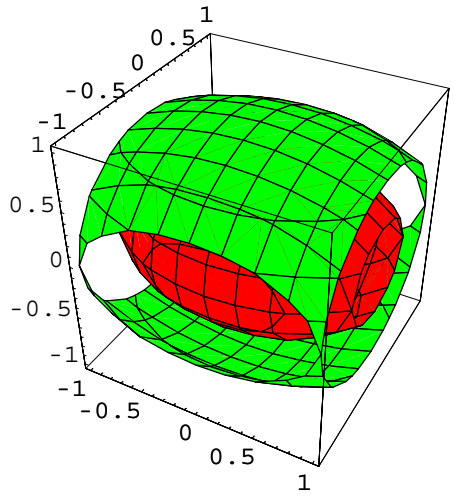

```
Out[39]= - Graphics3D -
```

Para funciones definidas a partir de tablas de datos como

```
In[40]:= datos = Table[x^2 + 2 * y^2 + 3 * z^2,
            {z, -1, 1, .25}, {y, -1, 1, .25}, {x, -1, 1, .25}];
```

Tenemos el comando:

```
In[41]:= ListContourPlot3D[datos, MeshRange → {{-1, 1}, {-1, 1}, {-1, 1}},
          Contours → {1.5, 3.}, Lighting → False, Axes → True,
          ContourStyle → {{RGBColor[0, 1, 0]}, {RGBColor[1, 0, 0]}}]
```




```
Out[41]= - Graphics3D -
```

Que podrían representar las superficies de nivel de un campo de temperaturas alrededor de un foco calorífico de forma elipsoidal.

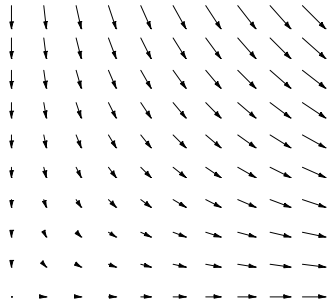
■ Campos de vectores en el plano

Para el uso de los siguientes comandos es necesario cargar previamente el paquete

```
In[42]:= << Graphics`PlotField`
```

Si queremos hacer una representación gráfica del campo de vectores $\vec{v}(x,y)=x\vec{i}-y\vec{j}$, escribiremos:

```
In[43]:= PlotVectorField[{x, -y}, {x, 0, 1}, {y, 0, 1}, PlotPoints -> 10]
```

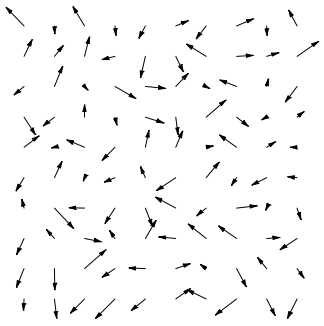


```
Out[43]= - Graphics -
```

donde la longitud de la flecha en cada punto es proporcional al módulo del vector. Este gráfico puede representar el campo de velocidades en un flujo bidimensional de un fluido cerca de una esquina.

Para un campo de vectores dado en un retículo a través de una tabla de valores (lista de vectores), tenemos el comando:

```
In[44]:= varray = Table[{Random[Real, {-0.7, 0.7}],
             Random[Real, {-0.7, 0.7}]}, {i, 10}, {j, 10}];
         ListPlotVectorField[varray]
```

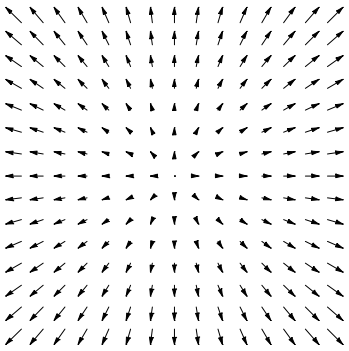


```
Out[45]= - Graphics -
```

que puede representar el campo de velocidades correspondiente al movimiento caótico de las moléculas en un gas bidimensional.

También se puede representar el campo de gradientes $\vec{\nabla} f(x, y) = \partial_x f \, \vec{i} + \partial_y f \, \vec{j}$ de una función escalar como $f(x,y)=x^2 + y^2$ mediante el comando:

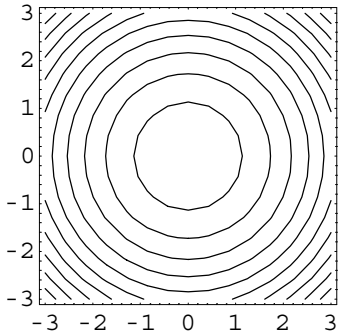
```
In[46]:= campodegradientes =
          PlotGradientField[x^2 + y^2, {x, -3, 3}, {y, -3, 3}]
```



```
Out[46]= - Graphics -
```

Para este caso es útil combinar el campo de gradientes con las curvas de nivel

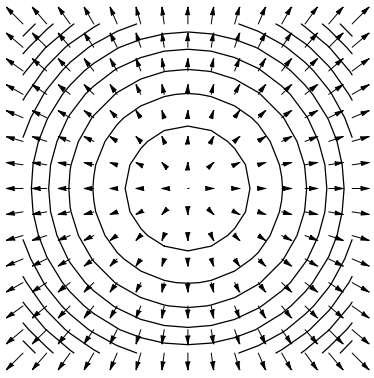
```
In[47]:= curvasdenivel = ContourPlot[x^2 + y^2,
           {x, -3, 3}, {y, -3, 3}, ContourShading → False]
```




```
Out[47]= - ContourGraphics -
```

en un único gráfico, como:

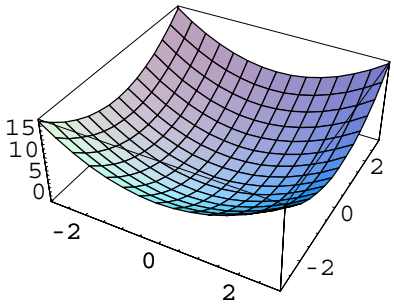
```
In[48]:= Show[{campodegradientes, curvasdenivel}]
```



```
Out[48]= - Graphics -
```

Observe que el gradiente aumenta conforme se aproximan las curvas de nivel unas a otras (es decir, al aumentar la pendiente). Observe también que el gradiente es perpendicular a las curvas de nivel en cada punto y que apunta en la dirección de máxima variación de la función *f(x,y)* en cada punto. En efecto, el punto (0,0) es un mínimo o, podríamos decir, "una fuente de gradiente", ya que el gradiente "fluye" alejándose de éste mínimo. Para asegurarnos mejor que se trata de un mínimo, no tenemos mas que dibujar la superficie $z=f(x,y)=x^2 + y^2$

In[49]:=
```
Plot3D[x^2 + y^2, {x, -3, 3}, {y, -3, 3}]
```



Out[49]= - SurfaceGraphics -

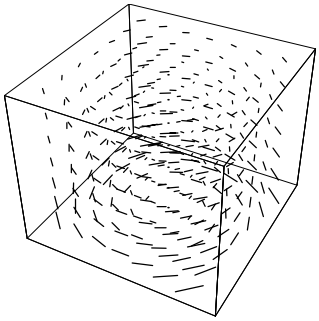
■ Campos de vectores en el espacio

Para el uso de los siguientes comandos es necesario cargar previamente el paquete

In[50]:=
```
<< Graphics`PlotField3D`
```

Si queremos hacer una representación gráfica del campo de vectores $\vec{v}(x,y,z)=\frac{y}{z}\vec{i}-\frac{x}{z}\vec{j}+0\vec{k}$, escribiremos:

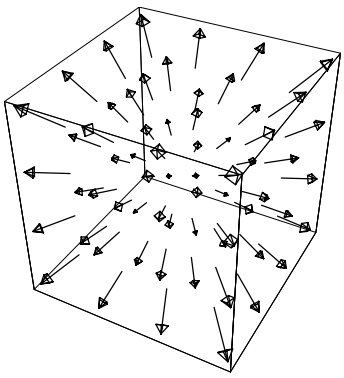
In[51]:=
```
PlotVectorField3D[{y, -x, 0} / z, {x, -1, 1}, {y, -1, 1}, {z, 1, 3}]
```



Out[51]= - Graphics3D -

que puede representar un "remolino" cuya velocidad disminuye conforme nos alejamos del suelo... Por defecto, los vectores en el espacio son representados por segmentos; si queremos añadirles la "flecha" deberemos especificar entre las opciones: **VectorHeads→True**. No obstante, esto puede hacer que el gráfico aparezca demasiado "recargado"; para evitar esto, podemos elegir la densidad de puntos a dibujar mediante la opción **PlotPoints→n**. Por ejemplo, representemos el campo $\vec{v}(x,y,z)=x\vec{i}+y\vec{j}+z\vec{k}$ con opciones:

In[52]:=
```
PlotVectorField3D[{x, y, z}, {x, -2, 2}, {y, -2, 2},
  {z, -2, 2}, PlotPoints → 4, VectorHeads → True]
```

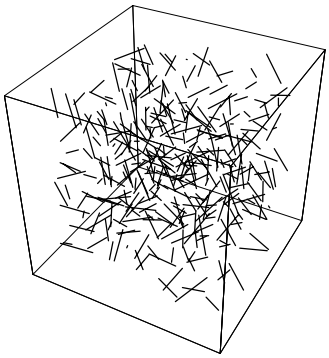


Out[52]= - Graphics3D -

que puede representar el campo de velocidades de un fluido cerca de una "fuente", localizada en (0,0,0), en tres dimensiones (o también el campo de fuerza de un oscilador repulsivo...)

Para un campo de vectores dado en un retículo a través de una tabla de valores (lista de vectores),tenemos el comando **ListPlotVectorField3D[tabla]**. Por ejemplo:

```
In[53]:= velocidadgas = Flatten[
           Table[{{i, j, k}, {Random[Real, {-1, 1}], Random[Real, {-1, 1}],
              Random[Real, {-1, 1}]}}, {i, 7}, {j, 7}, {k, 7}], 2];
         ListPlotVectorField3D[velocidadgas]
```

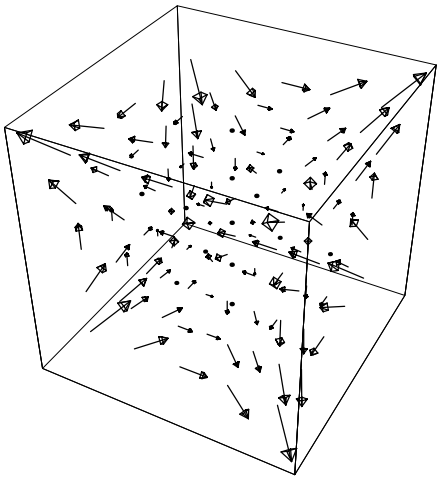


```
Out[54]= - Graphics3D -
```

que puede representar el campo de velocidades correspondiente al movimiento caótico de las moléculas en un gas tridimensional.

También se puede representar el campo de gradientes $\vec{\nabla} f(x, y, z) = \partial_x f \vec{i} + \partial_y f \vec{j} + \partial_z f \vec{k}$ de una función escalar como *f(x,y,z)=xyz* mediante el comando:

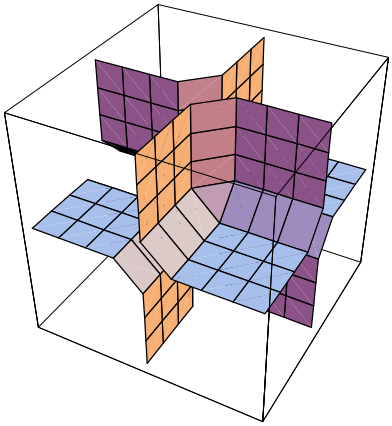
```
In[55]:= campograd3d = PlotGradientField3D[x y z, {x, -1, 1},
           {y, -1, 1}, {z, -1, 1}, PlotPoints → 5, VectorHeads → True]
```



```
Out[55]= - Graphics3D -
```

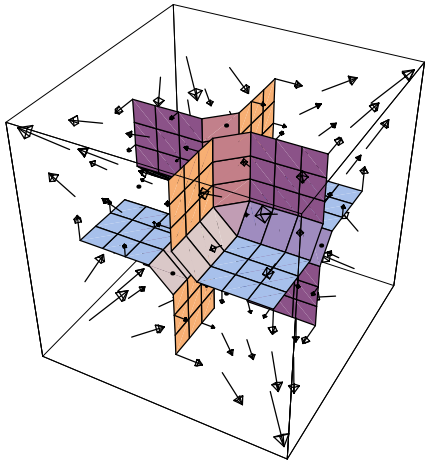
Podemos superponer el campo de gradientes con las superficies de nivel de f(x,y,z)=xyz mediante:

In[56]:=
```
<< Graphics`ContourPlot3D`
supnivel = ContourPlot3D[x y z, {x, -1, 1},
  {y, -1, 1}, {z, -1, 1}, ContourShading → False]
```



Out[57]= - Graphics3D -

In[58]:=
```
Show[{campograd3d, supnivel}]
```



Out[58]= - Graphics3D -

donde se observa que el gradiente de *f(x,y,z)* es perpendicular a las superficies de nivel *f(x,y,x)=cte.*

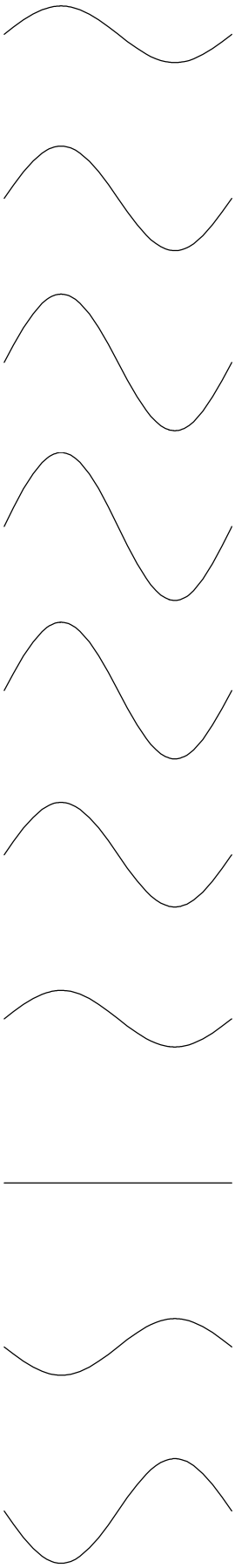
■ Animación de gráficos

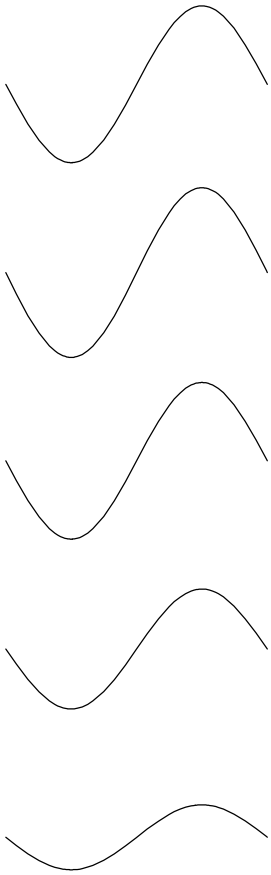
Para el uso de los siguientes comandos es necesario cargar previamente el paquete:

In[59]:=
```
<< Graphics`Animation`
```

Podemos crear una tabla con una familia de gráficas *y(x,t)* dependientes de un parámetro *t* ("tiempo"), para varios valores de dicho parámetro. Por ejemplo, construyamos una tabla con 15 gráficas de la forma *y(x,t)=sen(x)sen(t)* para 15 valores de *t* entre 0 y 2*π:*

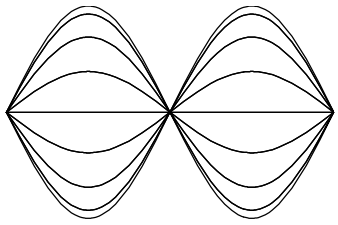
In[60]:=
```
cuerdavibrante = Table[Plot[Sin[x] Sin[t], {x, 0, 2 Pi},
    Axes → False, PlotRange -> {-1, 1}], {t, 0, 2 Pi - Pi / 8, Pi / 8}];
```





Estas gráficas se podrían corresponder con instantáneas hechas a una cuerda vibrante. Si superponemos todas las instantáneas obtendríamos:

In[61]:=
```
Show[cuerdavibrante]
```



Out[61]= - Graphics -

Pero si lo que queremos es obtener un "gráfico animado" con dichas instantáneas, lo único que debemos hacer es marcar todas las celdas (corchete a la derecha de la ventana de *Mathematica*) con el ratón y presionar las dos teclas CTRL+Y (o irse al menu Cell y elegir la opción Animate Selected Graphics). Una de las instantáneas comenzará una animación. Puedes cambiar la velocidad o dirección de la película usando los botones en la esquina inferior izquierda de la ventana de *Mathematica*. Para parar la animación no tienes mas que presionar cualquier tecla o el ratón. Si no deseamos gastar memoria guardando todas las instantáneas en la tabla **cuerdavibrante,** obtendremos idénticos resultados con el comando

```
Animate[Plot[Sin[ x] Sin[t], {x, 0, 2 Pi},
   Axes → False, PlotRange -> {-1, 1}], {t, 0, 2 Pi - Pi / 8, Pi / 8}];
```

La diferencia estará en que las 15 instantáneas aparecerán en pantalla pero no quedarán recogidas en memoria como una tabla. Otras posibilidades de crear animaciones son:

```
ShowAnimation[Table[Graphics[Line[{{0, 0}, {Cos[t], Sin[t]}}],
   PlotRange → {{-1, 1}, {-1, 1}}], {t, 0, 2 Pi, Pi / 8}]]
```

Ejercicios:

1. Compruebe que la anterior animación se trata de una varilla rotando en un plano con un punto fijo.
2. Cree una animación con 21 instantáneas para una membrana vibrante cuadrada que se mueve según el modo de vibración: $z(x, y, t) = \operatorname{sen}(2\pi x)\operatorname{sen}(4\pi y)\operatorname{sen}(t)$ para $0 \leq t \leq 2\pi$
3. ¿Qué ocurre con la gráfica de $f(x) = a\operatorname{sen}\left(\frac{2\pi}{p}(x-h)\right) + v$ cuando:
   a) varía el periodo $p = 1, 2, 3, 4,$ mientras $a = 3, h = v = 0$ permanecen fijos?
   b) varía el desfase horizontal $h = 0.5, 1, 1.5, 2,$ mientras $a = 3, p = 6, v = 0$ permanecen fijos?
   c) varía el desplazamiento vertical $v = 0.2, 0.4, 0.6, 0.8,$ mientras $a = 3, p = 6, h = 0$ permanecen fijos?
   c) varía la amplitud $a = 1, 1.5, 2, 2.5,$ mientras $p = 6, h = v = 0$ permanecen fijos?
4. ¿Qué ocurre con la gráfica de $f(x) = ax^2 + bx + c$ cuando:
   a) $a = -2, -1, 1, 2$ cambia mientras $b = 1$ y $c = 1$ permanecen fijas?
   b) $b = -2, -1, 1, 2$ cambia ($a = 1, c = 1$ fijos)?
   c) $c = -2, -1, 1, 2$ cambia ($a = 1, b = 1$ fijos)?
5. Representar 5 de las siguientes curvas parametrizadas (a elegir):
   (a) Circunferencia: $\begin{cases} x = R\cos\theta \\ y = R\operatorname{sen}\theta \end{cases} \quad 0 \leq \theta < 2\pi$
   (b) Elipse: $\begin{cases} x = a\cos\theta \\ y = b\operatorname{sen}\theta \end{cases} \quad 0 \leq \theta < 2\pi$
   (c) Cicloide: $\begin{cases} x = R(\theta - \operatorname{sen}\theta) \\ y = R(1 - \cos\theta) \end{cases} \quad 0 \leq \theta < 2\pi$
   (d) Astroide: $\begin{cases} x = R\cos^3\theta \\ y = R\operatorname{sen}^3\theta \end{cases} \quad 0 \leq \theta < 2\pi$
   (e) Nautilo: $\begin{cases} x = 2\cos\theta - \cos 2\theta \\ y = 2\operatorname{sen}\theta - \operatorname{sen} 2\theta \end{cases} \quad 0 \leq \theta < 2\pi$
   (f) Deltoide: $\begin{cases} x = 2\cos\theta + \cos 2\theta \\ y = 2\operatorname{sen}\theta - \operatorname{sen} 2\theta \end{cases} \quad 0 \leq \theta < 2\pi$
   (g) Ocho: $\begin{cases} x = \frac{1}{2}\operatorname{sen} 2\theta \\ y = \operatorname{sen}\theta \end{cases} \quad 0 \leq \theta < 2\pi$
   (h) Lágrima: $\begin{cases} x = 2a\cos\theta - a\operatorname{sen} 2\theta \\ y = b\operatorname{sen}\theta \end{cases} \quad 0 \leq \theta < 2\pi$
   (i) Espiral logarítmica (tornado): $\begin{cases} x = ce^{-\frac{a}{b}\theta}\cos\theta \\ y = ce^{-\frac{a}{b}\theta}\operatorname{sen}\theta \end{cases} \quad 0 \leq \theta < 2\pi$
   (j) Hélice: $\begin{cases} x = R\cos\theta \\ y = R\operatorname{sen}\theta \\ z = k\theta \end{cases} \quad 0 \leq \theta < 2\pi$
   (k) Lissajous $(n, 1)$: $x = a\operatorname{sen}(n\theta + c), y = b\operatorname{sen}(\theta)$
   (l) Tractriz: $x = 1/\cosh(\theta), y = \theta - \tanh(\theta)$

   Para $R = 1, a = 1, b = 2$.

6. Representar 3 de las siguientes superficies de $\mathbb{R}^3$ (a elegir) utilizando el comando ContourPlot3D

$S_1 = \{(x,y,z) \in \mathbb{R}^3 : x^2/a^2 + y^2/b^2 = 1\}$; (cilindro elíptico)
$S_2 = \{(x,y,z) \in \mathbb{R}^3 : x^2/a^2 - y^2/b^2 = 1\}$; (cilindro hiperbólico)
$S_3 = \{(x,y,z) \in \mathbb{R}^3 : x^2 = y\}$; (cilindro parabólico)
$S_4 = \{(x,y,z) \in \mathbb{R}^3 : \frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{z^2}{c^2}\}$; (cono elíptico)
$S_5 = \{(x,y,z) \in \mathbb{R}^3 : \frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1\}$; (hiperboloide una hoja)
$S_6 = \{(x,y,z) \in \mathbb{R}^3 : \frac{x^2}{a^2} - \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1\}$; (hiperboloide dos hojas)
$S_7 = \{(x,y,z) \in \mathbb{R}^3 : \frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{z}{c}\}$; (paraboloide elíptico))
$S_8 = \{(x,y,z) \in \mathbb{R}^3 : \frac{x^2}{a^2} - \frac{y^2}{b^2} = \frac{z}{c}\}$;(paraboloide hiperbólico)
$S_9 = \{(x,y,z) \in \mathbb{R}^3 : \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1\}$;(elipsoide)

Tomando, por ejemplo, a=b=c=1.

7. Representar las siguientes superficies parametrizadas:

(a) Cono: $\begin{cases} x = r\cos\theta \\ y = r\,\mathrm{sen}\,\theta \\ z = r \end{cases} \quad 0 \le \theta < 2\pi, 0 < r < \infty$

(b) Esfera: $\begin{cases} x = R\,\mathrm{sen}\,\theta\cos\phi \\ y = R\,\mathrm{sen}\,\theta\,\mathrm{sen}\,\phi \\ z = R\cos\theta \end{cases} \quad 0 \le \theta < \pi, 0 \le \phi < \pi$

(c) Toro: $\begin{cases} x = (R_1 + R_2\cos\theta)\cos\phi \\ y = (R_1 + R_2\cos\theta)\,\mathrm{sen}\,\phi \\ z = R_2\,\mathrm{sen}\,\theta \end{cases} \quad 0 \le \theta < 2\pi, 0 \le \phi < 2\pi$

Para $R = 1, R_1 = 2 > R_2 = 1$.

8. Dibujar las curvas (y superficies) de nivel $f(\vec{r}) = k$ de las siguientes funciones:

a) $f(x,y) = x^2 - y^2$ b) $f(x,y) = \mathrm{sen}(xy)\cos(xy)$ c) $f(x,y,z) = e^{(x^2+y^2+z^2)}$
d) $f(x,y) = \sqrt{x^2 - xy}$ e) $f(x,y) = x^2/2 + y^2/3$ f) $f(x,y,z) = \frac{1}{(2x^2+3y^2+4z^2)}$

9. Dibujar las siguientes funciones vectoriales:

a) $\vec{f}(x,y) = (x,y)/(x^2+y^2)$ b) $\vec{f}(x,y) = (y,0)$
c) $\vec{f}(x,y) = (-y,x)$ d) $\vec{f}(x,y) = (0, 1-x^2), -1 \le x \le 1$
e) $\vec{f}(x,y) = (x,-y)$ f) $\vec{f}(x,y) = (1+x,0)$

10. Representar el campo eléctrico

$$\vec{E}(x,y) = q_1\left(\frac{x}{((x+1)^2+y^2)^{3/2}}, \frac{y}{((x+1)^2+y^2)^{3/2}}\right) + q_2\left(\frac{x}{((x-1)^2+y^2)^{3/2}}, \frac{y}{((x-1)^2+y^2)^{3/2}}\right)$$

creado por dos cargas puntuales q_1 y q_2 localizadas en $P_1 = (-1,0)$ y $P_2 = (1,0)$ para: a) dos cargas positivas $q_1 = q_2 = 1$, b) dos cargas negativas $q_1 = q_2 = -1$, c) una positiva y otra negativa $q_1 = 1, q_2 = -1$.

Cálculo diferencial e integral con *Mathematica*

■ Resumen de comandos

CALCULO DIFERENCIAL

Limit[f[x],x→a] *límite de una función*

D[f[x],x]=f'[x]=$\partial_x f[x]$ *derivada (parcial) de una función*

D[f[x],{x,2}]=f''[x]=$\partial_{x,x} f[x]$ *derivada (parcial) segunda*

D[f[x],x,y]=$\partial_{x,y} f[x, y]$ *derivada cruzada*

`Derivative[`n_1**`,` **n_2**`, ...][`**f**`][`**x_1**`, `**x_2**`, ...]** *es la forma general de la función obtenida a partir de* **f** *derivando* n_1 *veces con respecto al primer argumento* x_1, n_2 *veces con respecto al segundo* x_2, *y así sucesivamente. Por ejemplo:*

`f'` *es equivalente a* **`Derivative[1][f]`**

`f''` *es equivalente a* **`Derivative[2][f]`**

<<Calculus`VectorAnalysis` *paquete necesario para hacer uso de Grad, Div, Curl, Laplacian.*

Grad[f,coordsystem] *gradiente de un campo escalar f en un sistema de coordenadas especificado*

Div[v,coordsystem] *divergencia de un campo vectorial v en un sistema de coordenadas*

Curl[v,coordsystem] *rotacional de un campo vectorial v en un sistema de coordenadas*

Laplacian[f,coordsystem] *laplaciano de un campo escalar f en un sistema de coordenadas*

CALCULO INTEGRAL

Integrate[f[x],x]= $\int f[x]\, dx$ *integral indefinida*

Integrate[f[x],{x,a,b}]= $\int_a^b f[x]\, dx$ *integral definida*

Integrate[f[x,y],{x,c,d},{y,a,b}]= $\int_c^d dx \int_a^b f[x, y]\, dy$ *integral múltiple*

NIntegrate[f[x],{x,a,b}] *integral numérica*

BUSQUEDA DE RAICES, MAXIMOS Y MINIMOS (NUMERICAMENTE)

FindRoot[f[x]==0, {x, x0}] *busca una* **raíz** *de f(x)=0 en un entorno de x0*

FindRoot[f[x]==0, {x, x0,xmin,xmax}] *busca una* **raíz** *de f(x)=0 en un entorno de x0 sin salirse del intervalo [xmin,xmax]*

FindRoot[{f[x,y]==0,g[x,y]==0},{x,x0},{y,y0}] *busca* **raíces** *de sistemas de ecuaciones*

FindMinimum[f,{x,x0}] *busca un mínimo de f alrededor de x0*

FindMinimum[f,{x, x0, xmin, xmax}] *busca un mínimo de f alrededor de x0 sin salirse del intervalo [xmin,xmax]*

FindMinimum[f,{x,x0},{ y,y0}, …] *busca un mínimo para una función de varias variables*

ConstrainedMin[f, {desigualdades},{x, y, … }] *busca un mínimo condicionado a ciertas ligaduras*

ConstrainedMax[f,{desigualdades}, {x, y, … }] *busca un mínimo condicionado a ciertas ligaduras*

■ Cálculo de Límites

Para calcular límites $\lim_{x\to a} f(x)$ de funciones reales de variable real, *Mathematica* dispone de la sentencia : **Limit[f(x), x->a]** . Por ejemplo, para obtener $\lim_{x\to 0} \frac{\text{sen}\alpha\, x}{x}$, la sintaxis es la siguiente:

```
In[1]:= Limit[Sin[α x] / x , x -> 0]
Out[1]= α
```

Los límites laterales se obtienen mediante las sentencias:

Limit[f(x), x->a, Direction->1] *límite lateral por la derecha*

Limit[f(x), x->a, Direction->-1] ***límite lateral por la izquierda.***

Por ejemplo, para la función *f(x)=1/x,* los límites laterales a la izquierda y a la derecha de *x=0* son:

```
In[2]:= Limit[1 / x, x -> 0, Direction -> -1]
        Limit[1 / x, x -> 0, Direction -> 1]
Out[2]= ∞
Out[3]= -∞
```

Ejercicio

a) Calcular: $\lim_{x\to\infty} \frac{\log x^5}{x}$; $\lim_{x\to 0^{\pm}} \frac{\tan(2x)}{x}$; $\lim_{x\to\infty} \frac{3^{x+1}+5^{x+1}}{3^x+5^x}$

b) Definir la función $f(x)=\frac{1-x^2}{1-x^5}$. Obtener: el valor de f en x=1, y los límites cuando x tiende a 1, a $+\infty$ y a $-\infty$. Dar la gráfica de f en el intervalo [-5,5].

c) Estudiar la continuidad de la función real de variable real g(x):

$g(x)=\frac{e^x}{1+2x+x^2}$ si x

■ Derivadas de funciones. Gradiente, divergencia y rotacional

Dada una función de una variable f(x) o de varias variables $f(x_1, x_2,..., x_n)$ queremos calcular su derivada o derivada parcial respecto alguna de sus variables. El comando que realiza ese cálculo con *Mathematica* es D[f,x] o D[f,x_i]. Por ejemplo si queremos calcular la derivada primera y segunda de f(x)= sen(x) podemos hacerlo de cuatro formas:

```
In[4]:= ∂x Sin[x]
        D[Sin[x], x]
        Sin'[x]
        Derivative[1][Sin][x]

        ∂x,x Sin[x]
        D[Sin[x], {x, 2}]
        Sin''[x]
        Derivative[2][Sin][x]
Out[4]= Cos[x]
Out[5]= Cos[x]
Out[6]= Cos[x]
Out[7]= Cos[x]
Out[8]= -Sin[x]
Out[9]= -Sin[x]
Out[10]= -Sin[x]
Out[11]= -Sin[x]
```

Donde, para la opción **∂x Sin[x]** utilizaremos la paleta BasicCalculations. Para calcular la derivada parcial con respecto a la variable *y* de la función $f(x,y) = x^2 y^3$ podemos hacerlo de estas tres formas:

In[12]:=

```
f[x_, y_] := x^2 y^3;
∂y f[x, y]
D[f[x, y], y]
Derivative[0, 1][f][x, y]
```

Out[13]= $3\,x^2\,y^2$

Out[14]= $3\,x^2\,y^2$

Out[15]= $3\,x^2\,y^2$

donde hemos definido previamente la función. Para calcular derivadas de orden superior, como $\frac{\partial^3 f}{\partial x\,\partial y^2}$, de la función $f(x,y) = x^2\,y^3$, escribiríamos:

In[16]:=

```
D[f[x, y], x, {y, 2}]
Derivative[1, 2][f][x, y]
```

Out[16]= $12\,x\,y$

Out[17]= $12\,x\,y$

Ejercicio

Calcula $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, $\frac{\partial^2 f}{\partial x\,\partial y}$, $\frac{\partial^3 f}{\partial x\,\partial y\,\partial x}$, de la función f(x,y) = $\tan(y/x)$

También podemos calcular el gradiente de dicha función con el comando **Grad**, cargando previamente el paquete

In[18]:=

```
<< Calculus`VectorAnalysis`
Grad[x^2 y^3, Cartesian[x, y, z]]
```

Out[19]= $\{2\,x\,y^3,\ 3\,x^2\,y^2,\ 0\}$

donde hemos especificado que el sistema de coordenadas es el cartesiano. Otros sistemas de coordenadas internos en Mathematica son: **Cylindrical[rho,phi,z]** y **Sphericał[r, theta, phi].** ***Mathematica*** puede recordarnos la expresión del gradiente de una función arbitraria $f(r,\vartheta,\varphi)$ en coordenadas esféricas sin mas que escribir:

In[20]:=

```
Grad[f[r, θ, φ], Spherical[r, θ, φ]]
```

Out[20]= $\left\{f^{(1,0,0)}[r, \theta, \varphi],\ \frac{f^{(0,1,0)}[r, \theta, \varphi]}{r},\ \frac{\text{Csc}[\theta]\,f^{(0,0,1)}[r, \theta, \varphi]}{r}\right\}$

donde $f^{(l,m,n)}$ simboliza la derivada parcial l-ésima respecto a r, m-ésima respecto a θ y n-ésima respecto a φ.

De la misma forma, podemos calcular la divergencia del campo vectorial $\vec{v}(x,y,z)=x\vec{i}+y\vec{j}+z\vec{k}$,

In[21]:=

```
Div[{x, y, z}, Cartesian[x, y, z]]
```

Out[21]= 3

o del campo gravitatorio $\vec{F}(r,\theta,\varphi) = -\hat{r}/r^2$ en coordenadas esféricas

```
In[22]:= Div[{-1 / r^2, 0, 0}, Spherical[r, θ, φ]]
Out[22]= 0
```

que nos indica que el campo gravitatorio es solenoidal (salvo en el punto *r=0*, donde el campo no está definido).

Para calcular el rotacional de $\vec{v}(x,y,z)=\frac{y}{z}\vec{i}-\frac{x}{z}\vec{j}+z\vec{k}$, escribiremos:

```
In[23]:= Curl[{y / z, -x / z, z}, Cartesian[x, y, z]]
Out[23]= {-x/z^2, -y/z^2, -2/z}
```

■ Cálculo de primitivas e integral definida

El comando que se utiliza para calcular la primitiva de una función f(x) es **Integrate[f[x],x]**. Por ejemplo, para calcular una primitiva de $f(x) = x^2$ procedemos del siguiente modo:

```
In[24]:= Integrate[x^2, x]
Out[24]= x^3/3
```

Si lo pretendemos es calcular la integral definida $\int_a^b f(x)\,dx$, el comando que debemos usar es **Integrate[f,{x,a,b}]**.

Entonces $\int_0^1 x\,dx$ se calcularía del modo siguiente:

```
In[25]:= Integrate[x, {x, 0, 1}]
Out[25]= 1/2
```

Para obtener una aproximación numérica del valor de la integral: **NIntegrate[f,{x,0,1}]**. Por ejemplo, existen integrales como

```
In[26]:= Integrate[Sin[x] / x, {x, 1, 2}]
Out[26]= -SinIntegral[1] + SinIntegral[2]
```

que no pueden expresarse en términos d funciones elementales. Para obtener un valor numérico escribiremos:

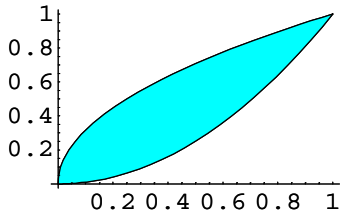
```
In[27]:= NIntegrate[Sin[x] / x, {x, 1, 2}]
Out[27]= 0.65933
```

También es posible realizar integrales múltiples definidas. Por ejemplo:

```
In[28]:= Integrate[x^2 y^4, {x, 0, 1}, {y, x^2, Sqrt[x]}]
Out[28]= 3/143
```

que nos da la integral de x^2y^4 en el recinto limitado por las curvas $y=x^2$ e $y=\sqrt{x}$. Dibujémoslo:

```
In[29]:= << Graphics`FilledPlot`
         FilledPlot[{x^2, Sqrt[x]}, {x, 0, 1}]
```

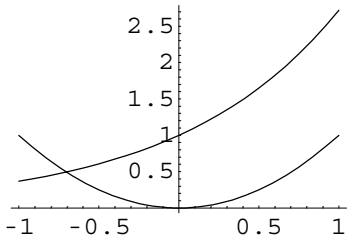



```
Out[30]= - Graphics -
```

■ Búsqueda de raíces, máximos y mínimos

Excepto para ecuaciones lineales o polinómicas, el cálculo de raíces de ecuaciones no admite, en general, una solución analítica. Son necesarios algoritmos numéricos como el método de la bisección, el de Newton, etc. *Mathematica* dispone de comandos que utilizan métodos numéricos para el cálculo de raíces. No obstante, siempre debemos dar una pequeña ayuda y proporcionar un valor tan cercano como sea posible a la raíz buscada, con el objetivo también de disminuir el tiempo de búsqueda. Por ejemplo, si queremos averiguar el punto de intersección de las curvas $y=e^x$ e $y=x^2$, es conveniente primero hacer una representación gráfica superpuesta de ambas funciones:

```
In[31]:= Plot[{Exp[x], x^2}, {x, -1, 1}];
```

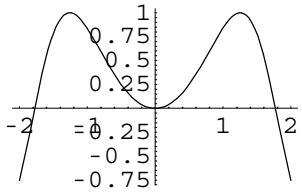


tras lo cual salta a la vista que el punto de corte se encuentra en el intervalo [-1,-0.5]. Podemos utilizar el punto x_0=-0.6 como punto de partida:

```
In[32]:= FindRoot[Exp[x] == x^2, {x, -0.6}]
Out[32]= {x → -0.703467}
```

obteniendo -0.703467 como un valor numérico aproximado para el punto de corte. Si existen varios puntos de corte como en

```
In[33]:= Plot[Sin[x^2], {x, -2, 2}]
```



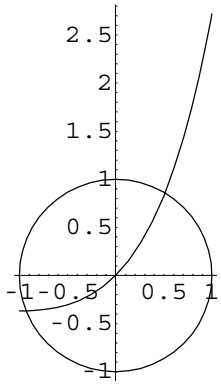

```
Out[33]= - Graphics -
```

y sólo estamos interesados en uno de ellos, entonces especificaremos un intervalo de búsqueda; por ejemplo, el [1,2]:

```
In[34]:= FindRoot[Sin[x^2] == 0, {x, 1.5, 1, 2}]
Out[34]= {x → 1.77245}
```

También es posible calcular raíces de ecuaciones en varias variables. Por ejemplo, dibujemos la circunferencia de radio 1 y la función $y=xe^x$

```
In[35]:= << Graphics`ImplicitPlot`
         ImplicitPlot[{x^2 + y^2 == 1, y == x Exp[x]}, {x, -1, 1}]
```




```
Out[36]= - Graphics -
```

que nos ofrece tres raíces: una alrededor de (x=-1,y=-0.5) , otra en (0,0) y otra alrededor de (x=0.5,y=1). Veamos la tercera:

```
In[37]:= FindRoot[{x^2 + y^2 == 1, y == x Exp[x]}, {x, .5}, {y, .9}]
Out[37]= {x → 0.513489, y → 0.858096}
```

Recordemos que se trata de valores aproximados. Por ejemplo, la función $y=x^4$ tiene evidentemente una raíz en x=0, sin embargo *Mathematica* nos ofrece:

```
In[38]:= FindRoot[x^4 == 0, {x, 1}]
Out[38]= {x → 0.0237573}
```

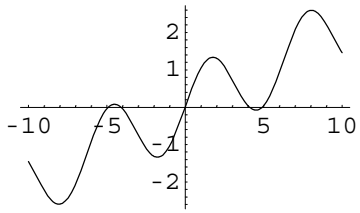
Si aumentamos la precisión y el número de iteraciones:

```
In[39]:= FindRoot[x^4 == 0, {x, 0.0237573},
              AccuracyGoal → 24, WorkingPrecision → 34,
              MaxIterations → 50]
Out[39]= {x → 4.247418856187920967704775348713908 × 10^-7}
```

Obtenemos un valor más cercano a cero, pero todavía no nulo...

Para la búsqueda de mínimos de funciones utilizaremos el comando **FindMinimum**. También es conveniente realizar una representación gráfica previa para visualizar el entorno donde localizar el mínimo a buscar. Por ejemplo:

```
In[40]:= Plot[x/5 + Sin[x], {x, -10, 10}];
```

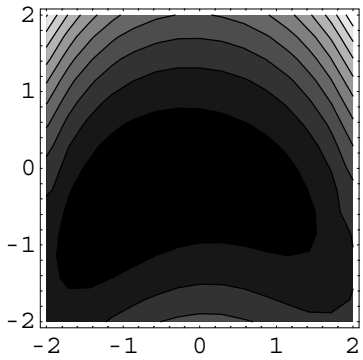


nos ofrece un mínimo alrededor de, digamos, x_0=5. El comando:

```
In[41]:= FindMinimum[x/5 + Sin[x], {x, 5}]
Out[41]= {-0.0775897, {x → 4.51103}}
```

nos ofrece x = 4.51103 como un valor aproximado para el mínimo, junto con el valor -0.0775897 de la función $f[x] = \frac{x}{5} + Sin[x]$ en dicho punto. Para superficies como z = f[x, y] = $x^4 + 3x^2 y + 5y^2 + x + y$, es útil realizar previamente una representación de sus curvas de nivel

```
In[42]:= ContourPlot[x^4 + 3 x^2 y + 5 y^2 + x + y, {x, -2, 2}, {y, -2, 2}]
```




```
Out[42]= - ContourGraphics -
```

tras lo cual, salta a la vista que tenemos un mínimo alrededor de (x,y)=(0,0). Con esta información podemos escribir:

```
In[43]:= FindMinimum[x^4 + 3 x^2 y + 5 y^2 + x + y, {x, 0}, {y, 0}]
Out[43]= {-0.832579, {x → -0.886227, y → -0.335714}}
```

que nos da como solución numérica {x→-0.886227,y→-0.335714} . Todavía podemos ayudar aún más a *Mathematica* y decirle, además del punto de partida, la dirección en la cual buscar. Por ejemplo, la dirección del gradiente:

```
In[44]:= gr = {D[x^4 + 3 x^2 y + 5 y^2 + x + y, x], D[x^4 + 3 x^2 y + 5 y^2 + x + y, y]}
         FindMinimum[x^4 + 3 x^2 y + 5 y^2 + x + y, {x, 0}, {y, 0}, Gradient → gr]

Out[44]= {1 + 4 x^3 + 6 x y, 1 + 3 x^2 + 10 y}

Out[45]= {-0.832579, {x → -0.886227, y → -0.335714}}
```

En este caso obtenemos un resultado idéntico al anterior.

Para funciones **f** lineales, el comando **ConstrainedMin[f, {desigualdades}, {x, y, … }]** encuentra el

mínimo de **f** sujeta a ciertas condiciones (ligaduras o "constraints") expresadas en forma de desigualdades. Por ejemplo:

```
In[46]:= ConstrainedMin[x + 3 y + 7 z,
           {x - 3 y < 7, 2 x + 3 z ≥ 5, x + y + z < 10}, {x, y, z}]
Out[46]= {5/2, {x → 5/2, y → 0, z → 0}}
```

o bien máximos:

```
In[47]:= ConstrainedMax[x + y, {x < 1, y < 2}, {x, y}]
Out[47]= {3, {x → 1, y → 2}}
```

Este tipo de optimización aparece con frecuencia en problemas de "Programación Lineal".

Ejercicios:

1. Calcular 4 de los siguientes límites (a elegir)

$$\text{(a) } \lim_{x\to 0}\frac{x\cos x-\operatorname{sen} x}{x^3}. \quad \text{(b) } \lim_{x\to 0}\frac{e^x}{x^2}. \quad \text{(c) } \lim_{x\to\infty}\frac{\ln x}{\sqrt[3]{x}}.$$

$$\text{(d) } \lim_{x\to \pi/2}\frac{\tan x}{\tan(5x)}. \quad \text{(e) } \lim_{x\to 0}\frac{\ln(\operatorname{sen} mx)}{\ln(\operatorname{sen} x)}. \quad \text{(f) } \lim_{x\to 1}\frac{\ln x}{\operatorname{sen} x}.$$

$$\text{(g) } \lim_{x\to 0}\left(\frac{1}{\operatorname{sen}^2 x}-\frac{1}{x^2}\right). \quad \text{(h) } \lim_{x\to 0}\frac{\ln^3\left(1-3x^2\right)}{x^4-x^4\cos^2(2x)}$$

$$\text{(i) } \lim_{x\to 0}\frac{1-\cos 3x}{\operatorname{sen} x^2} \quad \text{(j) } \lim_{x\to 0}\frac{\operatorname{sen} x-\dfrac{x+ax^3}{1+bx^2}}{\ln(x+1)}.$$

2. Dadas las funciones escalares

a) $f(x,y)=x^2-y^2$ b) $f(x,y)=\operatorname{sen}(x+y)$ c) $f(x,y,z)=ae^{b(x^2+y^2+z^2)}$
d) $f(x,y)=\sqrt{x^2-xy}$ e) $f(x,y)=x^2/2+y^2/3$ f) $f(x,y,z)=\frac{1}{(a^2x^2+b^2y^2+c^2z^2)}$

y vectoriales

g) $\vec{v}(\vec{r})=a\vec{r}$ h) $\vec{v}(x,y)=(\frac{x}{x^2+y^2},\frac{3x}{x^2+y^2})$
i) $\vec{v}(x,y)=(-y,x)$ j) $\vec{v}(x,y)=(0,R^2-x^2), -R\le x\le R$
k) $\vec{v}(x,y)=\frac{e^{-\arctan(y/x)}}{\sqrt{x^2+y^2}}(-x-y,x-y)$ l) $\vec{v}(x,y)=(1+y/(x^2+1),0)$

calcular:

(a) El gradiente $\vec{\nabla}f$
(b) La divergencia $\operatorname{div}(\vec{v})=\vec{\nabla}\cdot\vec{v}$
(c) El rotacional $\operatorname{rot}(\vec{v})=\vec{\nabla}\times\vec{v}$
(d) La divergencia del gradiente $\operatorname{div}(\vec{\nabla}f)=\vec{\nabla}\cdot(\vec{\nabla}f)$
(e) La divergencia del rotacional $\operatorname{div}(\vec{\nabla}\times\vec{v})=\vec{\nabla}\cdot(\vec{\nabla}\times\vec{v})$
(f) El rotacional del gradiente $\operatorname{rot}(\vec{\nabla}f)=\vec{\nabla}\times(\vec{\nabla}f)$
(g) El rotacional del rotacional $\operatorname{rot}(\operatorname{rot}(\vec{v}))=\vec{\nabla}\times(\vec{\nabla}\times\vec{v})$.

¿Qué campos son conservativos (irrotacionales)?. ¿Qué campos son solenoidales?.

3. Calcular 5 integrales a elegir entre:

(a) Funciones racionales $\int\frac{P_\alpha(x)}{P_\beta(x)}dx$

$a)\ \int\frac{x^2+1}{x^2-3x+2}dx$ $b)\ \int\frac{5x^2+20x+6}{x^3+2x^2+x}dx$ $c)\ \int\frac{1}{x^4-1}dx$
$d)\ \int\frac{x+2}{x^4-x^3}dx$ $e)\ \int\frac{x^2-3x+5}{x^3-x^2-2x}dx$ $f)\ \int\frac{x^3-3x^2}{x^2-4}dx$
$g)\ \int\frac{x-5}{x^2-2x+2}dx$ $h)\ \int x^n(1+x)^m dx$ $i)\ \int\frac{1}{e^{2x}+e^x+1}dx$

(b) Funciones trigonométricas $\int R(\operatorname{sen}x,\cos x)dx$

$a)\ \int\operatorname{sen}^{2n+1}x\cos^m x dx$ $b)\ \int\operatorname{sen}mx\operatorname{sen}nx dx$ $c)\ \int\operatorname{sen}^4 x dx$
$d)\ \int\frac{1}{\cos^4 x}dx$ $e)\ \int\operatorname{sen}mx\cos nx dx$ $f)\ \int\cos mx\cos nx dx$
$g)\ \int\frac{\cos x}{1+\cos x}dx$ $h)\ \int\frac{\tan x}{1+\cos x}dx$ $i)\ \int\frac{2-\operatorname{sen}x}{2+\operatorname{sen}x}dx$

(c) Funciones irracionales tipo $\int R\left(x, \left(\frac{ax+b}{cx+d}\right)^{p_1/q_1}, ..., \left(\frac{ax+b}{cx+d}\right)^{p_k/q_k}\right)dx$

$$a)\ \int \frac{x+\sqrt{2x+1}}{1+2\sqrt{2x+1}}dx \quad b)\ \int \frac{x+1}{\sqrt{3x-2}}dx \quad c)\ \int \frac{dx}{(2-x)\sqrt{1-x}}$$
$$d)\ \int x\sqrt{\frac{x-1}{x+1}}dx \quad e)\ \int \frac{\sqrt{x}-1}{\sqrt[3]{x}-1}dx \quad f)\ \int \frac{dx}{\sqrt{x}+\sqrt[3]{x}}$$

(d) Funciones irracionales tipo $R(x, \sqrt{\pm x^2 \pm a^2})dx$

$$a)\ \int \frac{\sqrt{3-x^2}}{x^2}dx \quad b)\ \int \frac{\sqrt{x^2-4}}{x}dx \quad c)\ \int \frac{dx}{\sqrt{x^2+4}}dx$$

(e) Funciones irracionales tipo $R(x, \sqrt{ax^2+bx+c})dx$

$$a)\ \int \frac{dx}{\sqrt{x^2+x+2}} \quad b)\ \int \frac{dx}{\sqrt{1+2x-x^2}} \quad c)\ \int \frac{3x+4}{\sqrt{4x-x^2}}dx$$

4. Calcular las siguientes integrales dobles:

(a) $\int_{-1}^{1}\int_{0}^{\sqrt{1-y^2}} \sqrt{x^2+y^2}dxdy.$

(b) $\int_{0}^{2}\int_{0}^{\sqrt{4-x^2}} \sqrt{x^2+y^2}dydx.$

(c) $\int_{1/2}^{1}\int_{0}^{\sqrt{1-x^2}} (x^2+y^2)^{3/2}dydx.$

(d) $\int_{0}^{1/2}\int_{0}^{\sqrt{1-y^2}} xy\sqrt{x^2+y^2}dxdy.$

5. Comprobar que las siguientes funciones poseen una raíz $f(x)=0$ en el intervalo especificado y calcularla aproximadamente

(a) $f(x) = x^2 - 4x + 4\,\mathrm{sen}^2\, x$ en el intervalo $(1, 3/2)$.

(b) $f(x) = \cos(x) - 2x^2$ en el intervalo $(0, 1)$.

(c) $f\,(x) = \dfrac{1}{x} - \tan x$ en el intervalo $(0, \pi/2)\,.$

(d) $f\,(x) = \dfrac{1}{x} - 2^x$ en el intervalo $(0, 1)\,.$

(e) $f\,(x) = 2^{-x} + e^x + 2\cos x - 6$ en el intervalo $[1, 3]\,.$

6. Hallar los extremos relativos de las funciones:

a) $f(x, y) = y^2 - x^2 = 0,$
b) $f(x, y) = x^2y^2,$
c) $f(x, y) = x^4 + y^4 - 2(x - y)^2,$
d) $f(x, y) = \cos x \cos y,$
e) $f(x, y, z) = x^2 + y^2 + z^2 - xy + x - 2z,$
f) $f(x, y) = x^3y^2(1 - x - y),$
g) $f(x, y) = x^2 + xy + y^2 + 1/x + 1/y,$
h) $f(x, y) = x^3 - y^2 + x^2 + 3xy,$

Manipulación de datos experimentales: ajustes e interpolación

■ Resumen de comandos

AJUSTE DE DATOS EXPERIMENTALES A UNA FAMILIA DE FUNCIONES

Fit[datos, funcs, vars] *ajusta una tabla de valores "***datos***" a una familia de funciones* "**funcs**" (f_i) *en las variables* "**vars**" (x_i, y_i, ...).

Los datos pueden venir en la forma: **{{**x_1**,** y_1**, … ,** f_1**},{**x_2**,** y_2**, … ,** f_2**}, … }**,

o en la forma: **{**f_1**,** f_2**, … }**. En este caso, *Mathematica* toma x_i = 1, 2, 3, ...

Por ejemplo, para:

Fit[{{x_1**,** y_1**,** f_1**}, … }, {1, x, y}, {x, y}]**

Mathematica busca una función interpoladora de dos variables del tipo: **z=f(x,y)=**a_0 **+** a_1 **x +** a_2 **y.**

<<NumericalMath`PolynomialFit`

PolynomialFit[datos,n] *ajusta los datos a un polinomio de grado n por el método de los mínimos cuadrados.*

INTERPOLACIÓN POR SEGMENTOS.

Interpolation[datos, `InterpolationOrder -> n`**]** *ajusta puntos sucesivos de la tabla a un polinomio de grado* n *(por defecto, Mathematica toma el valor* n=3*, si no se especifica nada), creando una función aproximada que interpola una tabla de datos (la función es* n *veces derivable).*

Los datos pueden venir en la forma; **{{**x_1**,** f_1**}, {**x_2**,** f_2**}, … } ,**

o en la forma: **{**f_1**,** f_2**, … }** (en este caso, *Mathematica* toma x_i = 1, 2, 3, ... por defecto)

o en la forma: **{{**x_1**,{**f_1**,** df_1**,** ddf_1**, … }}, … }** (cuando la tabla contiene, además de f_i, los valores de sus derivadas primeras df **,** segundas ddf **,** etc)

o en la forma: **{{x_1, y_1, … , f_1}, … }.** (si se trata de funciones de varias variables)

o en la forma: **{{x_1, y_1, … , {f_1,{dxf_1, dyf_1, … }},...}}.** (si se trata de funciones de varias variables con sus derivadas parciales)

Recordemos que para representar tablas de datos experimentales con barras de error disponemos de:

<< Graphics`Graphics`

ErroListPlot[lista] ***dibuja puntos de una tabla de valores*** **{{x1,y1,error1},...}** ***con barras de error***

<<Graphics`MultipleListPlot`

MultipleListPlot[lista] *dibuja puntos de una tabla de valores* **{{x1,y1,ErrorBar[e1min,e1max]},...}** *con barras de error especificadas por la opción* ***ErrorBar*** *en cada punto.*

■ Ajustes de datos experimentales a familias de funciones (mínimos cuadrados)

Ajustes a curvas arbitrarias

Supongamos que queremos verificar experimentalmente una teoría que nos dice que la relación entre la presión P y el volumen V de un cierto gas a temperatura fija es :

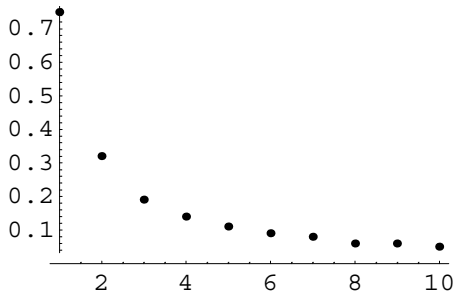
$$P = P(V) = a_1 / V + a_2 / V^2$$

donde a_i son constantes a determinar experimentalmente. Supongamos que en el laboratorio obtenemos una tabla (P, V) con los siguientes resultados de los valores de la presión para diferentes volúmenes entre 1 y 10 litros

```
In[1]:= presionvolumen = {{1, 0.75}, {2, 0.32}, {3, 0.19}, {4, 0.14},
          {5, 0.11}, {6, 0.09}, {7, 0.08}, {8, 0.06}, {9, 0.06}, {10, 0.05}};
```

Representemos gráficamente dichos datos experimentales en una gráfica:

```
In[2]:= pv1 = ListPlot[presionvolumen,
          PlotStyle → PointSize[0.02], AxesOrigin -> {1, 0}]
```



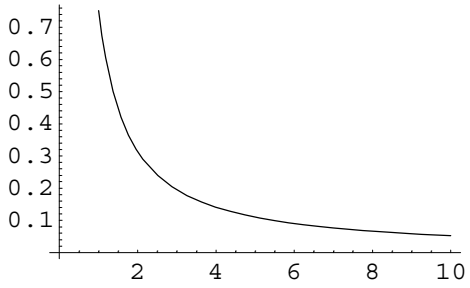

```
Out[2]= - Graphics -
```

Con el comando

```
In[3]:= funcpresvol = Fit[presionvolumen, {1 / v, 1 / v^2}, v]
Out[3]= 0.250082/v^2 + 0.501009/v
```

nos da la expresión de la función con los valores de a_i (a$_1$ = 0.250082, a$_2$ = 0.501009) que mejor se ajustan a los datos experimentales. Ahora podemos representar dicha función interpoladora

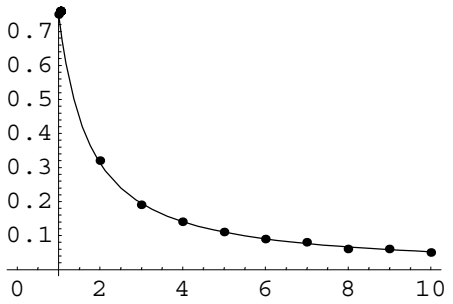
```
In[4]:= pv2 = Plot[funcpresvol, {v, 1, 10}]
```




```
Out[4]= - Graphics -
```

y superponerla con los datos experimentales

```
In[5]:= Show[{pv1, pv2}]
```




```
Out[5]= - Graphics -
```

para comprobar si el ajuste es bueno o malo. En este caso, el ajuste parece ser bueno. Si esto no fuese así, podríamos intentar un ajuste con otro tipo de funciones . Por ejemplo, veamos qué pasa si añadimos una dependencia tipo logarítmico ln V en el ajuste anterior:

```
In[6]:= funcpresvol2 = Fit[presionvolumen, {1 / v, 1 / v^2, Log[v]}, v]
Out[6]= 0.236932/v^2 + 0.513618/v - 0.00140944 Log[v]
```

Observamos que el coeficiente del logaritmo es bastante pequeño y que este término no modifica mucho a la expresión que ya teníamos. En definitiva, podemos decir que la predición de nuestra teoría es aceptable (dentro de los posibles errores experimentales). Ahora estamos en condiciones de predecir valores de la presión para valores del volumen distintos a los de la tabla. Por ejemplo, la predicción de la presión para un volumen de 15 litros sería:

```
In[7]:= funcpresvol /. v -> 15
Out[7]= 0.0345121
```

Las tablas de datos pueden venir con un cierto error experimental en la forma: lista={{x1,y1,error1},...}. Por ejemplo:

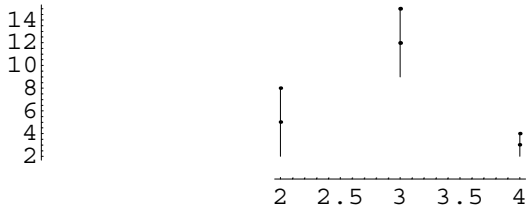
```
In[8]:= listaconerror = {{2, 5, 3}, {3, 12, 3}, {4, 3, 1}}
Out[8]= {{2, 5, 3}, {3, 12, 3}, {4, 3, 1}}
```

Podría representarse mediante el paquete

```
In[9]:= << Graphics`Graphics`
```

y el comando

```
In[10]:= ErrorListPlot[listaconerror, AxesOrigin -> 0]
```




```
Out[10]= - Graphics -
```

que añade barras de error de longitud ±3,±3 y ±1 a los puntos {x,y}={2,5},{3,12} y {4,3}. Si la barra de error no tiene la misma longitud por debajo que por encima del punto, entonces se puede utilizar la opción ErrorBar descrita en el resuman de comandos o en la Ayuda ("Help") de *Mathematica*.

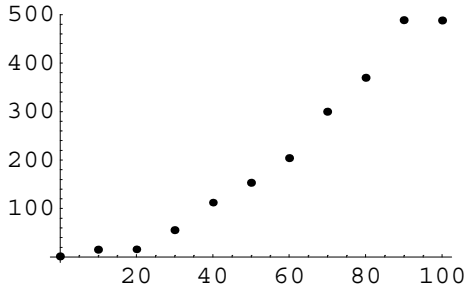
Ajustes polinómicos

Para el caso de ajustes de tipo polinómico, como el que puede relacionar a la presión con la temperatura (a volumen constante):

$$P = P(T) = a_0 + a_1 T + a_2 T^2$$

para un conjunto de datos experimentales *(T,P)* con *T* entre 0 y 100 grados centígrados:

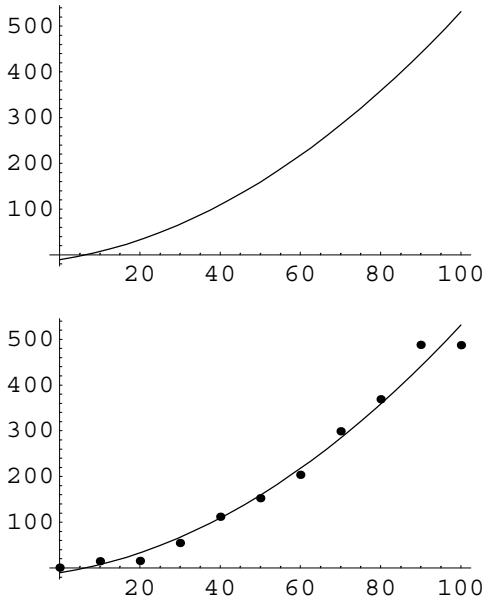
```
In[11]:= presiontemperatura = {{0, 1}, {10, 14.62}, {20, 15.16},
           {30, 54.95}, {40, 111.77}, {50, 152.36}, {60, 203.39},
           {70, 299.13}, {80, 369.19}, {90, 488.08}, {100, 487.27}};
         pt1 = ListPlot[presiontemperatura, PlotStyle → PointSize[0.02]];
```



podríamos utilizar la opción conocida:

```
In[13]:= funcprestemp = Fit[presiontemperatura, {1, t, t^2}, t]

         Show[{pt1, Plot[funcprestemp, {t, 0, 100}]}];
```

```
Out[13]= -10.8306 + 1.37888 t + 0.040459 t^2
```



o bien utilizar el paquete

```
In[15]:= << NumericalMath`PolynomialFit`
```

y el comando

```
In[16]:= funcprestemp2 = PolynomialFit[presiontemperatura, 2];
         Expand[funcprestemp2[t]]
Out[16]= -10.8306 + 1.37888 t + 0.040459 t^2
```

obteniendo resultados parecidos. La ventaja de **Fit** frente a **PolynomialFit** es que admite ajustes con funciones arbitrarias, no necesariamente de tipo polinómico. Sin embargo, el comando **PolynomialFit** es mucho más cómodo cuando sabemos que la dependencia funcional es de tipo polinómico, pero desconocemos el grado del mismo; podemos entondes ir probando con diferentes valores de **n** en **PolynomialFit[tabla,n]** hasta comprobar qué polinomio de regresión se ajusta mejor.

Existen otros paquetes para ajustes de tipo trigonométrico, segmentario, etc, pero no entraremos en ello.

Terminemos resaltando que las funciones interpoladoras como **funcprestemp** pueden derivarse, integrarse, etc: Por ejemplo:

```
In[17]:= D[funcprestemp, t]
         Integrate[funcprestemp, t]
Out[17]= 1.37888 + 0.0809179 t
Out[18]= -10.8306 t + 0.689442 t^2 + 0.0134863 t^3
```

■ Interpolación por segmentos polinómicos

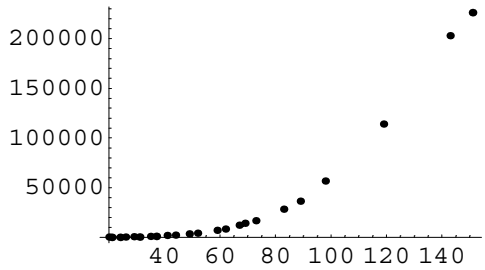
La interpolación segmentaria consiste básicamente en ajustar puntos sucesivos de la tabla (segmentos, en vez de toda la tabla) a un polinomio de grado n (por defecto, Mathematica toma el valor n=3, si no se especifica nada), creando una función que interpola una tabla de datos. La función así definida es suave (sin puntos angulosos) a trozos con derivadas suaves a trozos hasta orden n. Este tipo de ajuste es necesario cuando no conocemos el comportamiento cualitativo de la función incógnita en términos de funciones conocidas (como sucede en el ajuste por mínimos cuadrados con **Fit**).

Por ejemplo, consideremos la siguiente lista que contiene estadísticas **(mes, nº total de casos)** con el número total de casos de SIDA en EEUU desde 1980 en sucesivos meses posteriores a esta fecha:

```
In[19]:= sida = { {20, 110}, {21, 129}, {24, 220}, {26, 257}, {29, 439},
     {31, 514}, {35, 878}, {37, 1029}, {41, 1756}, {44, 2057},
     {49, 3512}, {52, 4115}, {59, 7025}, {62, 8229}, {67, 12067},
     {69, 14049}, {73, 16458}, {83, 28098}, {89, 36058},
     {98, 56575}, {119, 113891}, {143, 202843}, {151, 226252} };
```

Dibujemos los puntos de esta tabla

```
In[20]:= sidaplot =
      ListPlot[sida, PlotStyle → PointSize[0.02], PlotRange -> All]
```



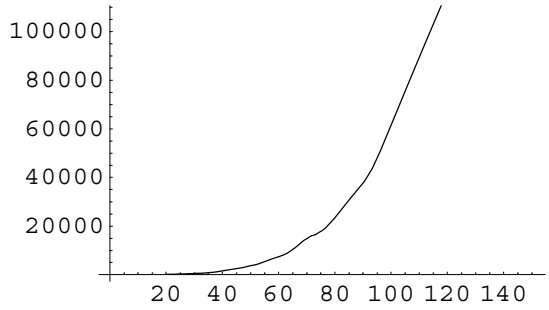

```
Out[20]= - Graphics -
```

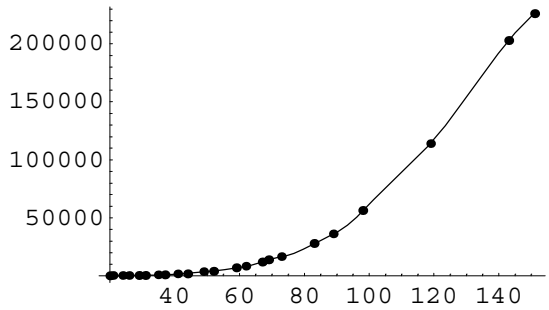
El comando

```
In[21]:= funcsida = Interpolation[sida, InterpolationOrder -> 4]
Out[21]= InterpolatingFunction[{{20, 151}}, <>]
```

crea una función interpoladora para la tabla de valores **sida**. Dibujemos dicha función superpuesta a la tabla de valores:

```
In[22]:= sidaplot2 = Plot[funcsida[x], {x, 20, 151}];
     Show[{sidaplot, sidaplot2}];
```





Podemos calcular la derivada primera y segunda de esta función como:

In[24]:=
```
dfuncsida = D[funcsida[x], x]
ddfuncsida = D[funcsida[x], {x, 2}]
```

```
General::spell1 :
 Possible spelling error: new symbol name "dfuncsida" is
   similar to existing symbol "funcsida".
```

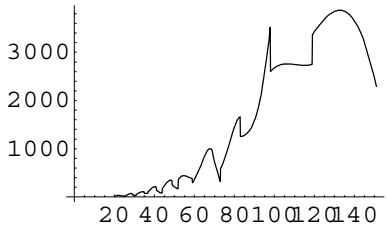
Out[24]=
```
InterpolatingFunction[{{20, 151}}, <>][x]
```

```
General::spell1 :
 Possible spelling error: new symbol name "ddfuncsida" is
   similar to existing symbol "dfuncsida".
```

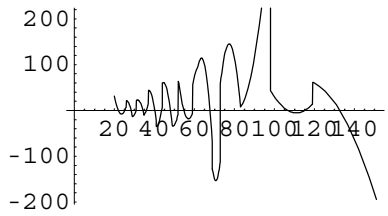
Out[25]=
```
InterpolatingFunction[{{20, 151}}, <>][x]
```

y dibujar también sus gráficas:

In[26]:=
```
Plot[dfuncsida, {x, 20, 151}]
Plot[ddfuncsida, {x, 20, 151}]
```



Out[26]= - Graphics -



Out[27]= - Graphics -

que nos muestran que la tasa de crecimiento de SIDA (derivada primera) aumenta con el tiempo con altibajos, presentando periodos donde el crecimiento es más acentuado (máximos, donde la segunda derivada de anula), por ejemplo, entre los meses del 130 al 140. No obstante, las derivadas presentan muchos altibajos debido a los puntos angulosos de la función interpoladora. Si el ritmo de crecimiento del número de casos de sida continúa en la misma tónica, podemos utilizar la función interpoladora para hacer una predicción del número total de casos de sida en el mes 160

In[28]:=
```
funcsida[160] // N
```

```
InterpolatingFunction::dmval :
 Input value {160} lies outside the range of data in the
   interpolating function. Extrapolation will be used.
```

Out[28]=
```
237165.
```

que nos da un valor de 237165 casos. mathemática nos advierte de que el mes 160 cae fuera del rango de interpolación [20,151], donde la interpolación es más fiable.

Si, además de la función, conocemos sus primeras derivadas en ciertos puntos, la función interpoladora será más precisa. Por ejemplo, consideremos la siguiente tabla con valores **{tiempo,{posición,velocidad}}** de la posición y la velocidad de un móvil en 12 instantes de tiempo igualmente espaciados entre t=0 y t=6 segundos:

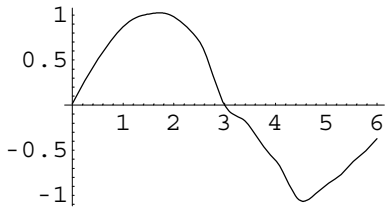
```
In[29]:= posivelotiempo = {{0, {0.017, 0.97}}, {0.5, {0.51, 0.75}},
           {1, {0.87, 0.53}}, {1.5, {1.01, 0.078}}, {2, {0.98, -0.35}},
           {2.5, {0.71, -0.76}}, {3, {0, -0.94}}, {3.5, {-0.24, -0.86}},
           {4, {-0.61, -0.53}}, {4.5, {-1.06, -0.22}},
           {5, {-0.88, 0.30}}, {5.5, {-0.65, 0.70}}, {6, {-0.37, 0.85}}};
```

La función

```
In[30]:= funcposivelotiempo =
          Interpolation[posivelotiempo, InterpolationOrder -> 2]
Out[30]= InterpolatingFunction[{{0., 6.}}, <>]
```

interpola estos 12 valores a una función que podemos representar gráficamente como:

```
In[31]:= Plot[funcposivelotiempo[t], {t, 0, 6}]
```




```
Out[31]= - Graphics -
```

que nos indica un cierto carácter oscilatorio...

■ Ejercicios

1) Ensayar un ajuste de tipo exponencial {1,Exp[x]} para la evolución de los casos de sida y hacer una representación gráfica de la diferencia entre este tipo de ajuste y el expuesto anteriormente (interpolación de orden 4)

2) Gastos de envio postal por año

La siguiente es una lista de tasas de gastos de envío postal (en céntimos) en EEUU durante más de un siglo. Cada elemento de la lista consta de (año,tasa de gastos)

postal= { {1885,2}, {1917,3}, {1919,2}, {1932,3}, {1958,4}, {1963,5}, {1968,6}, {1971,8}, {1974,10}, {1976,13}, {1978,15}, {1981,18}, {1982,20}, {1985,22}, {1988,25}, {1991,29}, {1995,32}, {1998,33} };

a) Buscar el mejor ajuste polinómico para esta lista.

b) Comparar este ajuste con uno segmentario de orden 4.

c) Hacer una predicción de la tasa de gastos para el año 2002

d) Calcular las derivadas primeras y segundas de la tasa de gastos y decir en qué periodo de años el aumento fué más pronunciado.

3) Densidad del agua

Se sabe que la densidad del agua alcanza un máximo a una temperatura ligeramente superior a la temperatura de congelación. Los siguientes datos, sacados del libro "CRC Handbook of Chemistry and Physics" dan la densidad del agua en gr/cm^3 para 5 temperaturas en ºC.

T	-10	0	10	20	30
D	.99815	.99987	.99973	.99823	.99567

1. Ajustar los datos a: a) un polinomio **p2densidad** de grado dos, b) uno **p3densidad** de grado tres

2. Superponer los gráficos de la tabla de valores y de los polinomios interpoladores

3. Usar el comando **Solve[D[pjdensidad,t]==0]**, para encontrar la temperatura a la cual la densidad es máxima usando ambos ajustes j=2,3. ¿Obtienes valores parecidos?. Si no es así, ¿cuál te parece más fiable?.

Sucesiones y series con *Mathematica*

■ Resumen de comandos

SUCESIONES Y SERIES NUMÉRICAS

Limit[a_i, i –> ∞] calcula el límite de la sucesión a_i cuando i tiende a infinito

Sum[a_i, {i, imin,imax,paso}]= $\sum_{i=\text{imin}}^{\text{imax}} a_i$ calcula la suma de la serie (el **paso** es opcional e igual a 1 por defecto)

SERIES DE POTENCIAS

`Series[`*`f`*`, {`*`x`*`, `x_0`, `*`n`*`}]` genera una serie de potencias (desarrollo de Taylor) para la función f alrededor del punto $x = x_0$ hasta orden $(x - x_0)^n$.

Normal[`Series[`*`f`*`, {`*`x`*`, `x_0`, `*`n`*`}]`] ofrece el polinomio de Taylor de grado n.

`Series[`*`f`*`,{`*`x`*`, `x_0`, `n_x`},{`*`y`*`, `y_0`, `n_y`}]` para funciones de varias variables.

`SeriesCoefficient[`*`serie`*`, `*`n`*`]` busca el coeficiente n-ésimo de la serie de potencias.

`InverseSeries[`*`s`*`, `*`x`*`]` coge la serie *s* generada por **`Series`**, y da una serie para la inversa de la función representada por *s*.

`SeriesData[`*`x`*`, `x_0`, {`a_0`, `a_1`, ...}, `*`nmin`*`, `*`nmax`*`, `*`den`*`]` construye una serie de potencias en la variable *x* alrededor del punto x_0. Los a_i son los coeficientes en la serie de potencias. Las potencias de $(x-x_0)$ que aparecen son *nmin*/*den*, (*nmin*+1)/*den*, … , *nmax*/*den*.

SERIES DE FOURIER

<<Calculus`FourierTransform` paquete a cargar

FourierTrigSeries[f[t], t, k] calcula el desarrollo en serie trigonométrico de orden k de una función periódica en el intervalo [-1/2,1/2]

FourierCosCoefficient[f[t],t,n] calcula el n-ésimo c_n coeficiente en el desarrollo en serie de cosenos

FourierSinCoefficient[f[t],t,n] calcula el n-ésimo d_n coeficiente en el desarrollo en serie de senos

NFourierTrigSeries[f[t], t, k] calcula el desarrollo en serie trigonométrico de orden k de una función periódica en el intervalo [-1/2,1/2] numéricamente

NFourierCosCoefficient[f[t],t,n] calcula el n-ésimo c_n coeficiente en el desarrollo en serie de cosenos numéricamente

NFourierSinCoefficient[f[t],t,n] calcula el n-ésimo d_n coeficiente en el desarrollo en serie de senos numéricamente

Recordemos que el desarrollo de Fourier de una función en el intervalo (0,1) y en un intervalo arbitrario (a,b) es

$\{0, 1\}$ $f(t) = c_0 + \sum_{n=1}^{k} c_n \cos(2\pi n t) + d_n \sin(2\pi n t)$

$\{a, b\}$ $f(t) = (|b|)^{(1+a)/2} (c_0 + \sum_{n=1}^{k} c_n \cos(2\pi b n t) + d_n \sin(2\pi b n t))$

donde:

$\{0, 1\}$ $c_0 = \int_{-1/2}^{1/2} f(t)\, dt$ $c_n = 2\int_{-1/2}^{1/2} f(t)\cos(2\pi n t)\, dt$

$\{a, b\}$ $(|b|)^{(1-a)/2} \int_{-1/(2|b|)}^{1/(2|b|)} f(t)\, dt$ $2(|b|)^{(1-a)/2} \int_{-1/(2|b|)}^{1/(2|b|)} f(t)\cos(2\pi b n t)\, dt$

y

$\{0, 1\}$ $d_n = 2\int_{-1/2}^{1/2} f(t)\sin(2\pi n t)\, dt$

$\{a, b\}$ $2(|b|)^{(1-a)/2} \int_{-1/(2|b|)}^{1/(2|b|)} f(t)\sin(2\pi b n t)\, dt$

Si no se especifica nada, *Mathematica* entiende que el intervalo que se repite es el [-1/2,1/2]. Para modificar el periodo *T=b-a* y el intervalo *[a,b]* a repetir, introduciremos la opción:

FourierParameters→{a/(b-a),b/(b-a)}

■ Cálculo de límites, sumas parciales y series numéricas

Podemos hacer límites de sucesiones como:

```
In[1]:= Limit[((2 n + z)/(2 n - z))^n, n → ∞]
Out[1]= ℯ^z
```

o límites laterales de funciones como:

```
In[2]:= Limit[Sqrt[x^2 - 6 x + 9]/(x - 3), x → 3, Direction → 1]
Out[2]= -1
```

```
In[3]:= Limit[Sqrt[x^2 - 6 x + 9]/(x - 3), x → 3, Direction → -1]
Out[3]= 1
```

Para sumar números que vienen dados por una cierta sucesión, como por ejemplo calcular la suma $\sum_{i=1}^{10} i^2$, *Mathematica* dispone del comando Sum, y el modo de utilizarlo es el

siguiente. Por ejemplo, para sumar los cuadrados de los primeros 10 números naturales $\sum_{i=1}^{10} i^2$ escribimos:

```
In[4]:= Sum[i^2, {i, 1, 10}]
Out[4]= 385
```

También se pueden hacer las sumas n-ésimas, así como la suma de series numéricas, es decir expresiones como $\sum_{i=1}^{n} i^2$, y $\sum_{i=1}^{\infty} (i)^{-2}$. Para ello debemos escribir, respectivamente,:

```
In[5]:= Sum[i^2, {i, 1, n}]
Out[5]= 1/6 n (1 + n) (1 + 2 n)

In[6]:= Sum[i^(-2), {i, 1, Infinity}]
Out[6]= π^2/6
```

Ejercicio

1) Calcula las siguientes sumas y productos:

$\sum_{i=1}^{n} 2^i$; $\sum_{i=1}^{10} (2+i)^i$; $\sum_{i=0}^{25} \frac{(x+i)^i}{x+i}$; $\prod_{i=1}^{10} (2+i)^i$; $\prod_{i=1}^{n} (2+i)^i$

2) Calcula las siguientes series numéricas:

$\sum_{n=1}^{\infty} \frac{(-1)^n}{n}$; $\sum_{n=1}^{\infty} \frac{1}{n}$; $\sum_{n=0}^{\infty} \frac{3^n}{n!}$

■ Series de potencias

Para obtener el desarrollo en serie de potencias de orden n de una función f(x) alrededor del punto x=a, disponemos de la sentencia : Series[f(x),{x,a,n}] , que nos proporciona la fórmula de Taylor de orden n de la función f(x) en el punto x=a. Para obtener el Polinomio de Taylor de grado n de la función f(x) en el punto x=a: Normal[Series[f(x),{x,a,n}]]. Por ejemplo:

```
In[7]:= sen0serie7 = Series[Sin[x], {x, 0, 7}]
Out[7]= x - x^3/6 + x^5/120 - x^7/5040 + O[x]^8
```

ofrece el desarrollo de orden 7 de la función seno en torno a x=0. Para obtener el polinomio de Taylor escribimos:

```
In[8]:= sen0poli7 = Normal[sen0serie7]
```

$$\text{Out[8]= } x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040}$$

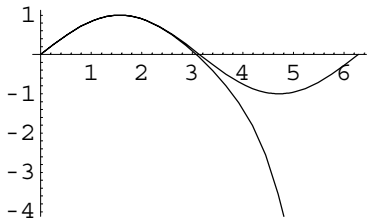
El coeficiente del término de grado 5 se obtiene

```
In[9]:= SeriesCoefficient[sen0serie7, 5]
```

$$\text{Out[9]= } \frac{1}{120}$$

Una representación gráfica conjunta de la función seno y de su desarrollo de orden 7 en torno a x=0

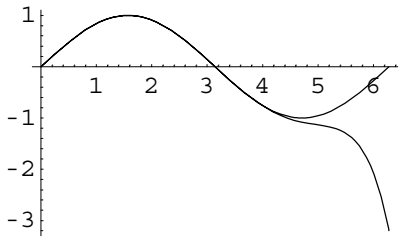
```
In[10]:= Plot[{Sin[x], sen0poli7}, {x, 0, 2 Pi}]
```




```
Out[10]= - Graphics -
```

nos indica que la aproximación es buena cerca de x=0, pero que ambas gráficas empiezan a divergir paulatinamente conforme nos alejamos de dicho punto. En particular, la aproximación polinómica de grado 7 parece ser bastante mala para puntos x>3. Si queremos un radio de "solapamiento" mayor, necesitamos añadir más términos al desarrollo. Por ejemplo:

```
In[11]:= sen0serie12 = Normal[Series[Sin[x], {x, 0, 12}]];
         Plot[{Sin[x], sen0serie12}, {x, 0, 2 Pi}]
```




```
Out[12]= - Graphics -
```

Para obtener la serie inversa de otra dada se utiliza el comando **InverseSeries**. Por ejemplo, dado el desarrollo en serie del seno

```
In[13]:= s = Series[Sin[x], {x, 0, 9}]
```

$$\text{Out[13]= } x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \frac{x^9}{362880} + O[x]^{10}$$

Obtenemos el desarrollo en serie del arcoseno mediante:

```
In[14]:= is = InverseSeries[s]
```

$$\text{Out[14]= } x + \frac{x^3}{6} + \frac{3\,x^5}{40} + \frac{5\,x^7}{112} + \frac{35\,x^9}{1152} + O[x]^{10}$$

En efecto:

```
In[15]:= Series[ArcSin[x], {x, 0, 9}]
```

$$\text{Out[15]}= x + \frac{x^3}{6} + \frac{3\,x^5}{40} + \frac{5\,x^7}{112} + \frac{35\,x^9}{1152} + O[x]^{10}$$

nos ofrece la misma expersión. Podemos también escribir funciones asociadas a una serie con coeficientes predeterminados:

```
In[16]:= SeriesData[x, 0, {1, -1 / 2, 1 / 3, -1 / 4, 1 / 5, -1 / 6, 1 / 7}, 1, 8, 1]
```

$$\text{Out[16]}= x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \frac{x^6}{6} + \frac{x^7}{7} + O[x]^8$$

Ejercicio

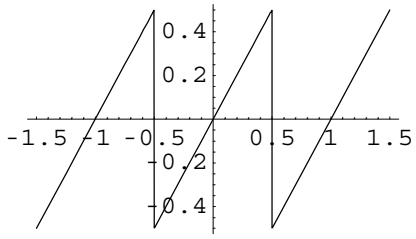
Obtener el desarrollo en serie de orden 3, 6 y 9 de : a) $y = e^x$ en torno a $x = 0$, y b) $y =$ arctan (x) en torno a $x = 0$.

Superponga la gráfica de la función a la de su desarrollo en serie de potencias en un cierto intervalo e identifique de forma aproximada los puntos a partir de los cuales ambas graficas difieren cualitativamente.

■ Series de Fourier

Para funciones *f(t)* periódicas de periodo 1, es decir, que cumplen *f(t)=f(t+1)*, como la función "diente de sierra" o "mantisa":

```
In[17]:= mantisa = Plot[t - Round[t], {t, -1.5, 1.5}]
```




```
Out[17]= - Graphics -
```

existe la posibilidad de aproximarlas por una serie trigonométrica
$f(t) = c_0 + \sum_{n=1}^{k} c_n \cos(2\pi n t) + d_n \sin(2\pi n t)$. Para ello disponemos del paquete:

```
In[18]:= << Calculus`FourierTransform`
```

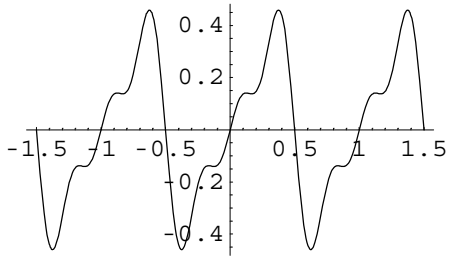
y del comando:

```
In[19]:= FourierTrigSeries[t, t, 3]
```

$$\text{Out[19]}= \frac{\text{Sin}[2\,\pi\,t]}{\pi} - \frac{\text{Sin}[4\,\pi\,t]}{2\,\pi} + \frac{\text{Sin}[6\,\pi\,t]}{3\,\pi}$$

que nos da la serie con k=3 términos. Representemos gráficamente dicha serie

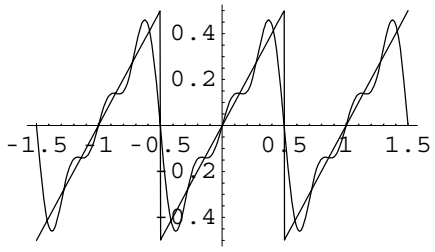
In[20]:= **Plot[%, {t, -1.5, 1.5}]**



Out[20]= **-** Graphics **-**

y superpongámosla a la función

In[21]:= **Show[mantisa, %]**



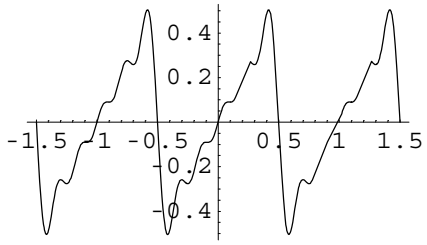
Out[21]= **-** Graphics **-**

Para conseguir un mejor ajuste debemos aumentar el número de términos, por ejemplo, de 3 a 5 términos:

In[22]:= **FourierTrigSeries[t, t, 5]**

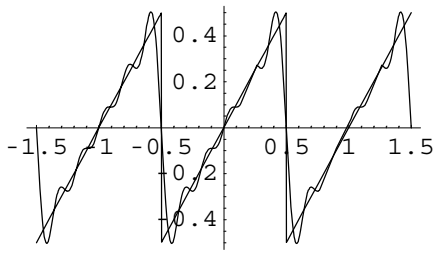
Out[22]= $\frac{\text{Sin}[2\,\pi\,t]}{\pi} - \frac{\text{Sin}[4\,\pi\,t]}{2\,\pi} + \frac{\text{Sin}[6\,\pi\,t]}{3\,\pi} - \frac{\text{Sin}[8\,\pi\,t]}{4\,\pi} + \frac{\text{Sin}[10\,\pi\,t]}{5\,\pi}$

In[23]:= **Plot[%, {t, -1.5, 1.5}]**



Out[23]= **-** Graphics **-**

In[24]:= **Show[mantisa, %]**



Out[24]= **-** Graphics **-**

Observe que obtenemos un mejor ajuste al aumentar el número de términos. Para calcular el coeficiente de Fourier correspondiente al modo n-ésimo escribiremos:

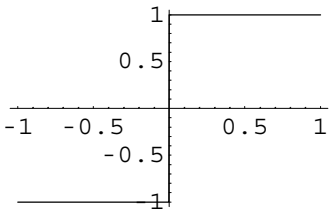
```
In[25]:= FourierSinCoefficient[t, t, n]
```

$$\text{Out[25]=} \quad \frac{-n\pi \text{Cos}[n\pi] + \text{Sin}[n\pi]}{n^2\pi^2}$$

que tiene un comportamiento del tipo 1/n, típico de funciones periódicas con discontinuidades de salto finito.

Consideremos otras funciones definidas a trozos como la función salto de Heaviside:

```
In[26]:= salto[t_] := If[t ≤ 0, -1, 1]
         Plot[salto[t], {t, -1, 1}]
```




```
Out[27]= - Graphics -
```

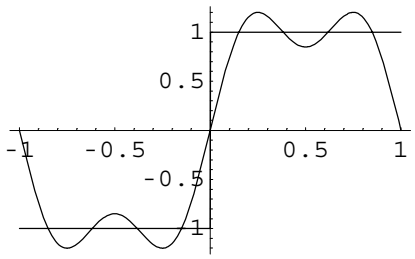
Si repetimos la gráfica de esta función a intervalos consecutivos (es decir, si construimos una función periódica con periodo 2=1-(-1) que salta de -1 a 1 en los pares y de 1 a -1 en los impares), entonces sus aproximaciones trigonométricas de orden 3,5 y 7 son:

```
In[28]:= fsalto = Table[FourierTrigSeries[salto[t], t,
            n, FourierParameters → {-1 / 2, 1 / 2}], {n, 3, 7, 2}]

         General::spell1 : Possible spelling error: new symbol
            name "fsalto" is similar to existing symbol "salto".
```

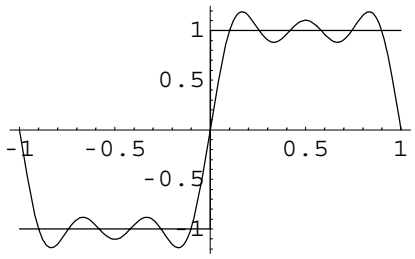
$$\text{Out[28]=} \quad \left\{ -\frac{\frac{4\,2^{1/4}\,\text{Sin}[\pi t]}{\pi} + \frac{4\,2^{1/4}\,\text{Sin}[3\pi t]}{3\pi}}{2^{1/4}},\ \frac{\frac{4\,2^{1/4}\,\text{Sin}[\pi t]}{\pi} + \frac{4\,2^{1/4}\,\text{Sin}[3\pi t]}{3\pi} + \frac{4\,2^{1/4}\,\text{Sin}[5\pi t]}{5\pi}}{2^{1/4}},\right.$$
$$\left. -\frac{\frac{4\,2^{1/4}\,\text{Sin}[\pi t]}{\pi} + \frac{4\,2^{1/4}\,\text{Sin}[3\pi t]}{3\pi} + \frac{4\,2^{1/4}\,\text{Sin}[5\pi t]}{5\pi} + \frac{4\,2^{1/4}\,\text{Sin}[7\pi t]}{7\pi}}{2^{1/4}} \right\}$$

Nótese que la opción **FourierParameters→{-1/2,1/2}** define una serie trigonométrica de periodo 2 en el intervalo [-1,1]; en general, para la serie trigonométrica de periodo *T=b-a* en el intervalo *[a,b]* escribiríamos: **FourierParameters→{a/(b-a),b/(b-a)}.** Si se omite esta opción, *Mathematica* entiende por defecto que se trata de **FourierParameters→{-1,1}.** Superpongamos la gráfica de la función salto y de sus desarrollos de Fourier de orden 3,5 y 7:

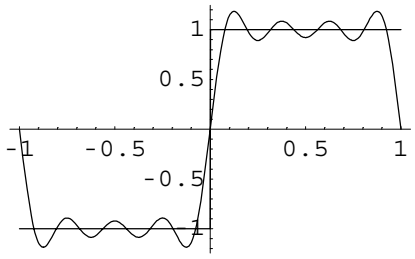
In[29]:=
```
Plot[{salto[t], fsalto[[1]]}, {t, -1, 1}]
Plot[{salto[t], fsalto[[2]]}, {t, -1, 1}]
Plot[{salto[t], fsalto[[3]]}, {t, -1, 1}]
```



Out[29]= - Graphics -



Out[30]= - Graphics -



Out[31]= - Graphics -

Vemos que el desarrollo se ajusta mejor (en media) a la función salto cuanto mayor es el orden.

Ejercicio

Obtener el desarrollo en serie de orden 3, 6 y 9 de : $f(t) = 0$ si $t \in [-\pi, 0[$, $f(t) = \text{sen}(t)$ si $t \in [0, \pi]$.

Superponga la gráfica de la
función a la de su desarrollo en serie de Fourier en el intervalo $[-\pi, \pi]$.

Geometría de curvas y superficies

■ Geometría de curvas en el plano y el espacio

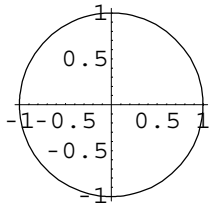
Longitud de una curva

Nos centraremos por ahora en curvas planas definidas en forma paramétrica, $\vec{r}(t) = (x(t), y(t)) = x(t)\,\vec{i} + y(t)\,\vec{j}$, en vez de la forma explícita $y = f(x)$. Aquí t es el *parámetro* (ángulo, tiempo, etc) que nos da la posición $\vec{r}$ para cada valor de t. Por ejemplo, las expresiones:

```
In[1]:= Clear[x, y, r, t]
        x[t_] := Cos[t]
        y[t_] := Sin[t]
        r[t_] := {x[t], y[t]}
```

definen las ecuaciones paramétricas de una circunferencia de radio 1, cuya representación gráfica puede hacerse mediante el comando:

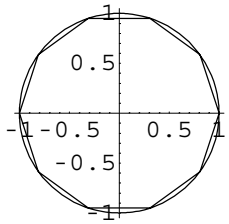
```
In[5]:= circ = ParametricPlot[{x[t], y[t]}, {t, 0, 2 Pi}, AspectRatio → Automatic]
```




```
Out[5]= - Graphics -
```

Comencemos por encontrar la longitud de dicha curva. Recuérdese que el espíritu del cálculo radica en aproximar el arco de curva por segmentos rectilíneos. Por ejemplo, dividamos nuestra circunferencia en n=10 segmentos

```
In[6]:= n = 10;
        decacirc = ListPlot[Table[r[i], {i, 0, 2 Pi, 2 Pi / n}],
            PlotJoined → True, AspectRatio → Automatic, DisplayFunction :> Identity];
        Show[decacirc, circ, DisplayFunction :> $DisplayFunction];
```



Nótese que ésta es una aproximación "decente" a la circunferencia. Fabriquemos una tabla con las longitudes **longseg[[i]]** de cada uno de estos **i=1,...,10** segmentos y calculemos la suma de las longitudes de estos 10 segmentos:

In[9]:=
```
longseg = Table[Sqrt[(r[i + 2 Pi/n] - r[i]).(r[i + 2 Pi/n] - r[i])], {i, 0, 2 Pi - 2 Pi/n, 2 Pi/n}];
N[Sum[longseg[[i]], {i, 1, Length[longseg]}]]
```

Out[10]=
```
6.18034
```

(Repita el alumno este cálculo para 20 y 30 segmentos)

Conforme el número de segmentos (**n**) crece, la suma de sus longitudes se aproxima a la longitud exacta de la circunferencia de radio 1 que, como sabemos, es $2\pi \approx 6.28319$. Recordemos cómo se calcula la longitud exacta de una curva mediante el cálculo integral. Llamemos $d\,l$ a la longitud de un segmento infinitesimal. Usando el teorema de Pitágoras vemos que:

$d\,x^2 + d\,y^2 = d\,l^2$, es decir, $d\,l = \sqrt{d\,x^2 + d\,y^2}$, o escrito en forma paramétrica $d\,l = \sqrt{x'(t)^2 + y'(t)^2}\; dt = \|\vec{r}\,'(t)\|\, dt$

donde $\|\vec{v}\| = \sqrt{\vec{v}.\vec{v}}$ es la norma de un vector.

Sumando las longitudes de todos estos segmentos diferenciales tenemos que:

$$\text{longitud} = \int_a^b \|\vec{r}\,'(t)\|\, dt$$

donde $\vec{r}(t) = x(t)\, i + y(t)\, j$ es una curva derivable en el intervalo $a \le t \le b$. En nuestro caso, la longitud de la circunferencia es:

In[11]:=
```
Integrate[Sqrt[∂t r[t].∂t r[t]], {t, 0, 2 π}]
```

Out[11]=
```
2 π
```

Obteniendo el resultado que ya esperábamos. *Mathematica* dispone del comando **ArcLengthFactor** para calcular la expresión $\sqrt{\partial_t\, \mathbf{r}[t].\partial_t\, \mathbf{r}[t]}$. En efecto:

In[12]:=
```
Sqrt[∂t r[t].∂t r[t]] // Simplify
<< Calculus`VectorAnalysis`
ArcLengthFactor[{Cos[t], Sin[t], 0}, t, Cartesian] // Simplify
```

Out[12]=
```
1
```

Out[14]=
```
1
```

nos dan el mismo resultado (observe que ha sido necesario cargar el paquete **<<Calculus`VectorAnalysis`** y escribir la curva plana **{Cos[t],Sin[t]}** como una curva en el espacio **{Cos[t],Sin[t],0}** añadiendo la coordenada z=0). La ventaja del comando **ArcLengthFactor** es que permite curvas dadas en otros sistemas de coordenadas que no sea el cartesiano sin mas que modificar la opción **Cartesian** por **Spherical, Cylindrical**, etc.

Ejercicios

1) El cálculo de este tipo de integrales suele ser bastante complicado, ¡incluso para *Mathematica!*. Por ejemplo, intente calcular la longitud de una elipse de semiejes 2 y 3:

x[t_]:=2Cos[t]

y[t_]:=3Sin[t]

r[t_]:={x[t],y[t]}

primero de forma aproximada (para 10 y 20 segmentos) y después de forma exacta. Se dará cuenta que la integral no es expresable en términos de funciones elementales. Pruebe entonces una integración numérica.

2) Intente también el cálculo exacto de la longitud de la parábola $y=x^2$ para $x\in[0,1]$

3) Vea si es capaz de hacer lo propio para una curva en tres dimensiones como la hélice:

x[t_]:=Cos[t]

y[t_]:=Sin[t]

z[t_]:=t

r[t_]:={x[t],y[t],z[t]}

entre t=0 y t=2π.

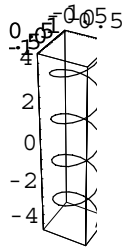
Parametrización de una curva por su longitud de arco

Hemos visto que la longitud *l(t)* del arco de una curva $\vec{r}$ entre un punto *a* y otro arbitrario *t* se define como una función $l(t) = \int_a^t || \vec{r}\,'(w) || \, dw$. Podemos crear nuestro propio comando para calcular longitudes de arcos de curva de la siguiente manera:

In[15]:=
```
Clear[r, x, y, a, b];
norma[x_] := Sqrt[x.x]
longitudarco[r_, a_, b_] := Integrate[norma[D[r[#1]]], {#1, a, b}]
```

Por ejemplo, para la hélice

In[18]:=
```
r[t_] := {Cos[t], Sin[t], t/3}
rtrayectoria = ParametricPlot3D[{Cos[t], Sin[t], t/3}, {t, -4 π, 4 π}];
```



tenemos que la longitud de un arco de hélice entre $w=-4\pi$ y $w=t$ es

In[20]:=
```
l[t_] := longitudarco[r, -4 π, t]
```

Bajo ciertas condiciones, esta longitud de arco $l(t)$ tiene inversa $t(l)$, la cual podemos calcular mediante:

In[21]:=
```
Solve[l[t] == l, t]
t[l_] = t /. %[[1]]
```

Out[21]=
$$\left\{\left\{t \to -\frac{-3\,l + 4\sqrt{10}\,\pi}{\sqrt{10}}\right\}\right\}$$

Out[22]=
$$-\frac{-3\,l + 4\sqrt{10}\,\pi}{\sqrt{10}}$$

La curva $\vec{r}(t)$ puede ahora parametrizarse en función de la longitud l como $\vec{u}(l) = \vec{r}(t(l))$. Puesto que t va de -4π to 4π, l irá de $l(-4\pi)$ a $l(4\pi)$. Veamos qué aspecto tiene la hélice parametrizada por la longitud de arco:

In[23]:=
```
u[l_] := r[t[l]]
Simplify[u[l]]
```

Out[24]=
$$\left\{\text{Cos}\left[\frac{3\,l}{\sqrt{10}}\right], \text{Sin}\left[\frac{3\,l}{\sqrt{10}}\right], \frac{l}{\sqrt{10}} - \frac{4\,\pi}{3}\right\}$$

Veamos cuál es la longitud de arco para $t=4\pi$ (para $t=-4\pi$ la longitud debe ser cero, ¿por qué?):

In[25]:=
```
l[-4 π]
l[4 π] // N
```

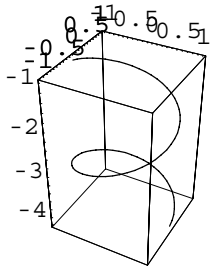
Out[25]=
```
0
```

Out[26]=
```
26.4922
```

Si queremos representar la trayectoria recorrida por un móvil que se mueve sobre la hélice cuando éste ha recorrido $l=10$ unidades métricas, entonces debemos utilizar la parametrización en función de la longitud del arco de curva:

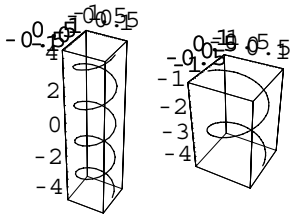
In[27]:= **utrayectoria = ParametricPlot3D[u[l], {l, 0, 10}];**

```
General::spell1 :
 Possible spelling error: new symbol name "utrayectoria" is
   similar to existing symbol "rtrayectoria".

ParametricPlot3D::ppcom :
 Function u[l] cannot be compiled; plotting
   will proceed with the uncompiled function.
```



Comparemos con la trayectoria $\vec{r}(t)$ parametrizada en función de *t*

In[28]:= **Show[GraphicsArray[{rtrayectoria, utrayectoria}]];**



La diferencia está en que *l=10* implica $t<4\pi$. En efecto:

In[29]:= **t[10] // N**

Out[29]= -3.07954

es decir, una longitud de arco *l=10* es equivalente a $t\approx$*-3.07954*; o sea, el móvil ha recorrido algo mas del 37% de su trayectoria entre $t=-4\pi$ y $t=4\pi$. Otro aspecto interesante de parametrizar una curva por la longitud de arco es que el vector tangente siempre tiene longitud 1; en efecto:

In[30]:= **u′[l]**

$$\text{Out[30]}= \left\{\frac{3\,\text{Sin}\left[\frac{-3\,l+4\sqrt{10}\,\pi}{\sqrt{10}}\right]}{\sqrt{10}},\ \frac{3\,\text{Cos}\left[\frac{-3\,l+4\sqrt{10}\,\pi}{\sqrt{10}}\right]}{\sqrt{10}},\ \frac{1}{\sqrt{10}}\right\}$$

In[31]:= **Simplify[u′[l]]**

$$\text{Out[31]}= \left\{-\frac{3\,\text{Sin}\left[\frac{3\,l}{\sqrt{10}}\right]}{\sqrt{10}},\ \frac{3\,\text{Cos}\left[\frac{3\,l}{\sqrt{10}}\right]}{\sqrt{10}},\ \frac{1}{\sqrt{10}}\right\}$$

In[32]:= **Simplify$\left[\sqrt{\text{u′[l].u′[l]}}\right]$**

Out[32]= 1

Ejercicio

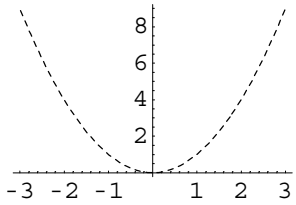
Un móvil se desplaza sobre una parábola $y=x^2$ desde *x=0* hasta *x=4*. Proporcione una expresión de la posición *{x(l),y(l)}* del móvil en función de la longitud *l* del arco de curva recorrido. Haga una representación gráfica de la trayectoria del móvil desde *x=0* hasta *x=4* y de la trayectoria desde *l=0* hasta *l=15*. ¿Cuál es el espacio total recorrido desde *x=0* hasta *x=4* ?.

Vector tangente y normal a una curva: curvatura y torsión

Veamos que $\vec{v}(t) = \vec{r}\,'(t)$ es un vector tangente a la curva $\vec{r}(t)$. Como ejemplo utilizaremos la parábola:

In[33]:=
```
Clear[x, y, r, v]
x[t_] := t
y[t_] := t^2
r[t_] := {x[t], y[t]}
parabola = ParametricPlot[r[t], {t, −3, 3}, PlotStyle −> Dashing[{0.02}]];
```

```
ParametricPlot::ppcom :
 Function r[t] cannot be compiled; plotting
   will proceed with the uncompiled function.
```



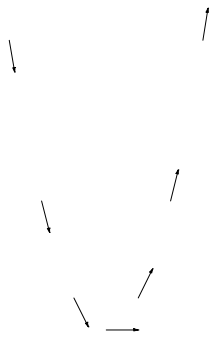
El vector tangente unitario se calcula como:

In[38]:=
$$\mathbf{vu[t\_] := \frac{r'[t]}{\sqrt{r'[t].r'[t]}}}$$
```
Simplify[vu[t]]
```

Out[39]=
$$\left\{\frac{1}{\sqrt{1+4\,t^2}},\ \frac{2\,t}{\sqrt{1+4\,t^2}}\right\}$$

Representemos gráficamente el vector unitario tangente a la parábola en varios puntos

In[40]:=
```
<< Graphics`PlotField`
vlistplot = ListPlotVectorField[Table[{r[i], vu[i]}, {i, −3, 3}], AspectRatio → Automatic];
```

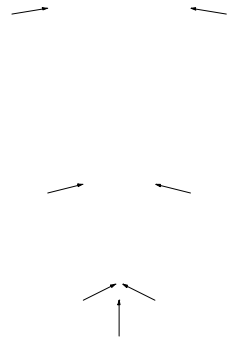


El vector normal unitario se calcula como:

```
In[42]:= nu[t_] := vu'[t] / Sqrt[vu'[t].vu'[t]]
         Simplify[nu[t]]
Out[43]= {-2 t Sqrt[1/(1 + 4 t^2)^2] Sqrt[1 + 4 t^2], Sqrt[1/(1 + 4 t^2)^2] Sqrt[1 + 4 t^2]}
```

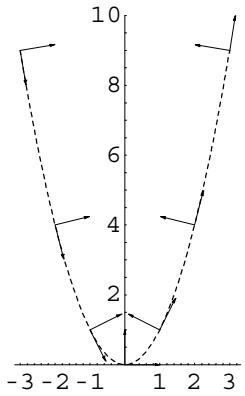
Representemos gráficamente el vector unitario normal a la parábola en varios puntos

```
In[44]:= nlistplot = ListPlotVectorField[
           Table[{r[i], nu[i]}, {i, -3, 3}], AspectRatio → Automatic];
         General::spell1 :
          Possible spelling error: new symbol name "nlistplot" is
            similar to existing symbol "vlistplot".
```



Juntando las tres gráficas obtenemos:

```
In[45]:= Show[{parabola, vlistplot, nlistplot}, AspectRatio → Automatic, PlotRange → {0, 10}];
```



Que nos muestra el "biedro" de Frenet en los puntos x=-3,-2,-1,0,1,2,3. Observe que $\vec{n}(t) = \vec{v}'(t) / \| \vec{v}'(t) \|$ es perpendicular al vector tangente, lo que implica que es perpendicular a la curva. El vector normal unitario $\vec{n}(t)$ puede también escribirse como:

$\vec{n}(t) = \frac{\vec{v}'(t)}{\kappa(t) \|\vec{r}'(t)\|}$, (demuéstrelo)

donde $\kappa(t)$ es la *curvatura.* Recordemos que la curvatura representa la variación del vector tangente con respecto a la longitud de arco:

$\kappa(l) = \| \frac{d\,\vec{v}}{dl} \|$.

Como función de *t*, la regla de la cadena nos da:

$\kappa(t) = ||\vec{v}'(t)|| \,/\, ||\vec{r}'(t)||$.

La idea es que, si la variación de dirección del vector tangente es muy grande, entonces la curva se curva ("dobla") de forma considerable.

Si la curva está parametrizada por la longitud de arco *l*, el vector normal es simplemente:

$\vec{n}(l) = \frac{d\,\vec{v}(l)}{d\,l} \Big/ \kappa(l)$. (demuéstrelo)

Lo mismo que hemos definido un comando para la longitud del arco de curva, podemos definir otro para la curvatura de la siguiente manera:

```
In[46]:= curvatura[r_, t_] := Module[{vu, k},
           vu[t] = D[r[t], t] / Sqrt[D[r[t], t].D[r[t], t]];
           k = Sqrt[D[vu[t], t].D[vu[t], t]] / Sqrt[D[r[t], t].D[r[t], t]];
           Return[k]]
```

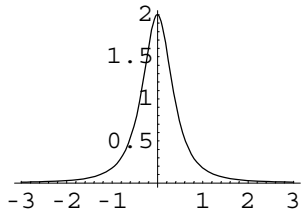
Por ejemplo, la curvatura de la parábola es:

```
In[47]:= Simplify[curvatura[r, t]]
```

$$\text{Out[47]=}\quad \frac{2\sqrt{\frac{1}{(1+4\,t^2)^2}}}{\sqrt{1+4\,t^2}}$$

que se simplifica a $2 \Big/ \sqrt{(1+4\,t^2)^3}$. Representemos el valor de la curvatura de la parábola en función del parámetro *t:*

```
In[48]:= Plot[2 / Sqrt[(1 + 4 t^2)^3], {t, -3, 3}]
```

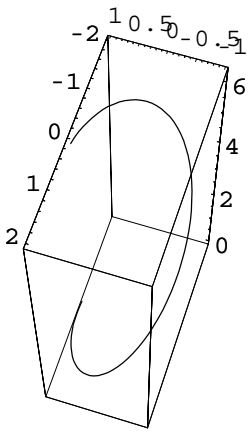



```
Out[48]= - Graphics -
```

Vemos que la parábola $y=x^2$ se curva más "intensamente" en las cercanías de *x=0*.

Para curvas en el espacio disponemos de más recursos gráficos para visualizar la curvatura, tales como el uso de colores. Por ejemplo, consideremos una hélice resultante al enrollar un cable a lo largo de un cilindro elíptico de semiejes 1 y 2:

```
In[49]:= Clear[x, y, z, r, v]
         x[t_] := Cos[t]
         y[t_] := 2 Sin[t]
         z[t_] := t
         h[t_] := {x[t], y[t], z[t]}
         heliceliptica = ParametricPlot3D[h[t],
            {t, 0, 2 Pi}, ViewPoint -> {-0.531, 1.647, 3.507}];
         ParametricPlot3D::ppcom :
          Function h[t] cannot be compiled; plotting
            will proceed with the uncompiled function.
```



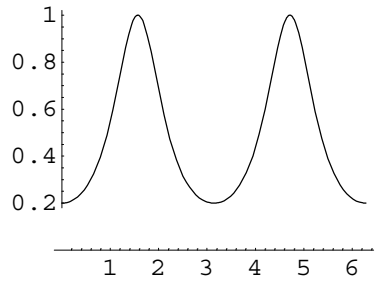
Su curvatura es:

```
In[55]:= Simplify[curvatura[h, t]]
```

$$\text{Out[55]=}\quad \frac{2\sqrt{\frac{13-3\cos[2t]}{(7+3\cos[2t])^2}}}{\sqrt{7+3\cos[2t]}}$$

Representemos gráficamente la curvatura como una función de *t:*

```
In[56]:= Plot[2 Sqrt[(13 - 3 Cos[2 t])/(7 + 3 Cos[2 t])^2]/Sqrt[7 + 3 Cos[2 t]],
           {t, 0, 2 Pi}, AxesOrigin -> {0, 0}]
```



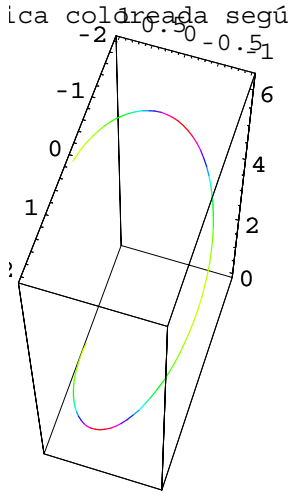

```
Out[56]= - Graphics -
```

Vemos que la curvatura presenta máximos en t=π/2 y t=3π/2. Definiendo un color distinto para la curvatura de la hélice elíptica en cada punto *t* como:

```
In[57]:= colorhelip[t_] := Hue[2 Sqrt[(13 - 3 Cos[2 t])/(7 + 3 Cos[2 t])^2]/Sqrt[7 + 3 Cos[2 t]]]
```

Podemos crear un gráfico de dicha curva coloreado de acuerdo con la intensidad de la curvatura por medio de:

In[58]:=
```
ParametricPlot3D[{Cos[t], 2 Sin[t], t, colorhelip[t]}, {t, 0, 2 Pi},
 PlotLabel -> "hélice elíptica coloreada según curvatura",
 ViewPoint -> {-0.531, 1.647, 3.507}]
```



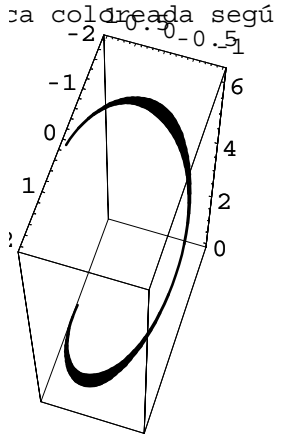
Out[58]= - Graphics3D -

de manera que las zonas rojas se corresponden con puntos de mayor curvatura que las verdes. También podemos usar el recurso del grosor de línea

In[59]:=
```
grosorhelip[t_] := Thickness[2 Sqrt[(13 - 3 Cos[2 t])/(7 + 3 Cos[2 t])^2]/Sqrt[7 + 3 Cos[2 t]] / 15]
```

(nótese que hemos dividido por 15 para no tener un grosor excesivo) para dar idea de las zonas de mayor curvatura, como:

In[60]:=
```
ParametricPlot3D[{Cos[t], 2 Sin[t], t, grosorhelip[t]}, {t, 0, 2 Pi},
 PlotLabel -> "hélice elíptica coloreada según curvatura",
 ViewPoint -> {-0.531, 1.647, 3.507}]
```



Out[60]= - Graphics3D -

de manera que las zonas de mayor curvatura aparecen con un grosor de línea mayor.

También podemos definir un comando para la torsión $\tau(t)=\frac{\|\vec{r}'\times\vec{r}''\bullet\vec{r}'''\|}{\|\vec{r}'\bullet\vec{r}'\|}$ como:

```
In[61]:= torsion[r_, t_] := Module[{V, T},
           V[t] = D[r[t], t];
           T = Cross[V[t], D[V[t], t]].D[V[t], t] /
              Sqrt[Cross[r[t], V[t]].Cross[r[t], V[t]]];
           Return[
            T]]
```

de manera que la torsión de la hélice elíptica es:

```
In[62]:= Simplify[torsion[h, t]]
Out[62]= 0
```

Se deja al alumno el hacer una representación gráfica coloreada, y otra de grosor, de la hélice elíptica de acuerdo con la torsión.

Ejercicio

Realice una representación gráfica coloreada,y otra de grosor, de la curva de Vidiani

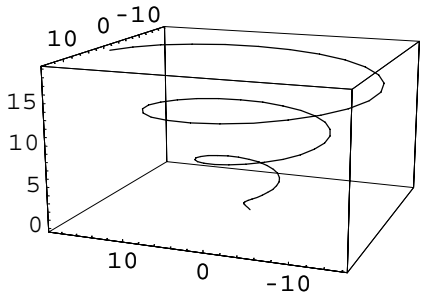
$x(t)=(1+\cos(t)),\ y(t)=\sin(t),\ z(t)=2\sin(t/2),\ 0\le t\le 4\pi$

de acuerdo con la curvatura y la torsión. Realice también un gráfico superpuesto con el triedro de Frenet (utilice sólo el vector tangente y el normal).

Velocidad y aceleración

Si una partícula se mueve de manera que, en el instante t, su posición viene dada por $\vec{r}(t) = \{x(t),\ y(t),\ z(t)\}$, entonces su velocidad viene dada por $\vec{v}(t) = \vec{r}\,'(t) = \{x'(t),\ y'(t),\ z'(t)\}$. La velocidad es tangente a la trayectoria en todos los puntos (donde no se anula). Por ejemplo, dada la siguiente trayectoria:

```
In[63]:= Clear[r, t];
         r[t_] := {t Cos[t], t Sin[t], t};
         remolino =
           ParametricPlot3D[{t Cos[t], t Sin[t], t}, {t, 0, 6 Pi}, ViewPoint -> {-1.258, 3.293, 0.875}]
```



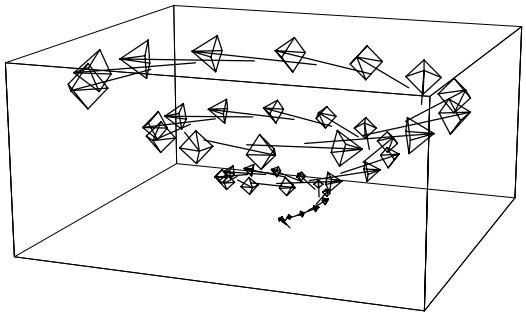

```
Out[65]= - Graphics3D -
```

La velocidad se calcula como:

```
In[66]:= Clear[v];
         v[t_] := r'[t]
         Simplify[v[t]]
Out[68]= {Cos[t] - t Sin[t], t Cos[t] + Sin[t], 1}
```

Representemos gráficamente la velocidad en 40 puntos de la trayectoria anterior

In[69]:=
```
<< Graphics`PlotField3D`
velolistplot = ListPlotVectorField3D[Table[{r[i], v[i]}, {i, 0, 6 Pi, 6 Pi / 40}],
    AspectRatio → Automatic, ViewPoint –> {–1.258, 3.293, 0.875}, VectorHeads → True]
```



Out[70]= - Graphics3D -

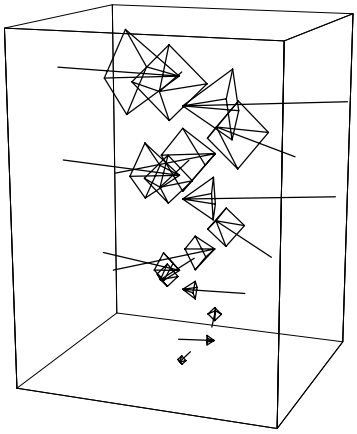
De la misma forma, la aceleración viene dada por:

In[71]:=
```
a[t_] := v'[t]
Simplify[a[t]]
```

Out[72]=
```
{-t Cos[t] - 2 Sin[t], 2 Cos[t] - t Sin[t], 0}
```

Representemos gráficamente la aceleración en 40 puntos de la trayectoria anterior

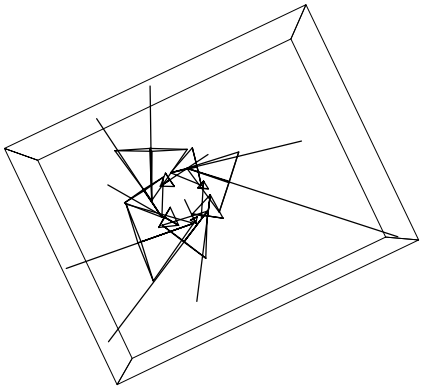
In[73]:=
```
acelistplot = ListPlotVectorField3D[
  Table[{r[i], a[i]}, {i, 0, 6 Pi, 6 Pi / 15}], AspectRatio → Automatic,
  ViewPoint -> {-1.258, 3.293, 0.875}, VectorHeads → True]
```



Out[73]= - Graphics3D -

o desde otro punto de vista (visto desde arriba):

```
In[74]:= ListPlotVectorField3D[Table[{r[i], a[i]}, {i, 0, 6 Pi, 6 Pi / 10}],
           AspectRatio → Automatic, ViewPoint –> {0.033, 0.069, 5.763}, VectorHeads → True]
```



```
Out[74]= - Graphics3D -
```

que indica que la aceleración apunta hacia adentro de la trayectoria y aumenta conforme nos elevamos en altura.

Ejercicio

Haga una representación gráfica de la velocidad y aceleración de un planeta en su movimiento alrededor de una estrella siguiendo una trayectoria elíptica de semiejes 4 y 1

$x(t)=4\cos(t)$, $y(t)=\text{sen}(t)$

en los 8 puntos siguientes t=0, $\pi/4$, $2\pi/4$, $3\pi/4$, π, $5\pi/4$, $6\pi/4$ y $7\pi/4$

Componentes tangencial y normal de la aceleración

Las componentes intrínsecas de la aceleración

$$\vec{a}(t) = a_t(t)\,\hat{v} + a_n(t)\,\hat{n}$$

son

componente tangencial: $a_t = \frac{d\,v}{d\,t}$,

componente normal: $a_n = v^2\,\kappa = v^2 / R$

donde $\vec{v} = v\,\hat{v}$ es la velocidad (módulo por vector unitario), κ es la curvatura y $\hat{n}$ el vector normal. Veamos cómo calcular y representar dichas componentes. Tomemos como ejemplo la elipse de semiejes 3 y 1:

In[75]:=
```
Clear[r, t]
r[t_] := {3 Cos[t], Sin[t]}
elip = ParametricPlot[{3 Cos[t], Sin[t]}, {t, 0, 2 π},
    AspectRatio → Automatic, PlotStyle –> Dashing[{0.02}], Axes –> False];
```

La velocidad, su módulo y su dirección y el vector normal unitario son:

In[78]:=
```
v[t_] := r'[t]; v[t]
vmod[t_] := √(r'[t].r'[t]); Simplify[vmod[t]]
vu[t_] := r'[t] / √(r'[t].r'[t]); Simplify[vu[t]]
nu[t_] := vu'[t] / √(vu'[t].vu'[t]); Simplify[nu[t]]
```

Out[78]= $\{-3\,\text{Sin}[t],\ \text{Cos}[t]\}$

Out[79]= $\sqrt{5 - 4\,\text{Cos}[2\,t]}$

Out[80]= $\left\{-\frac{3\,\text{Sin}[t]}{\sqrt{5 - 4\,\text{Cos}[2\,t]}},\ \frac{\text{Cos}[t]}{\sqrt{5 - 4\,\text{Cos}[2\,t]}}\right\}$

Out[81]= $\left\{-\frac{\text{Cos}[t]}{\sqrt{\frac{1}{(5-4\,\text{Cos}[2\,t])^2}}\ (5 - 4\,\text{Cos}[2\,t])^{3/2}},\ -\frac{3\,\text{Sin}[t]}{\sqrt{\frac{1}{(5-4\,\text{Cos}[2\,t])^2}}\ (5 - 4\,\text{Cos}[2\,t])^{3/2}}\right\}$

La aceleración es:

In[82]:=
```
Clear[a];
a[t_] := r''[t]; Simplify[a[t]]
```

Out[83]= $\{-3\,\text{Cos}[t],\ -\text{Sin}[t]\}$

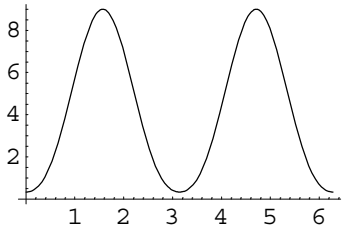
Las componentes tangencial y normal se obtienen proyectando sobre los vectores unitarios tangencial y normal:

In[84]:=
```
Clear[at, an];
at[t_] := (a[t].vu[t]) vu[t]
an[t_] := (a[t].nu[t]) nu[t]
```

El radio de curvatura es $R = v^2 / a_n$, o sea:

In[87]:=
```
rcurv[t_] := vmod[t]^2 / √(an[t].an[t]); Simplify[rcurv[t]]
Plot[rcurv[t], {t, 0, 2 Pi}]
```

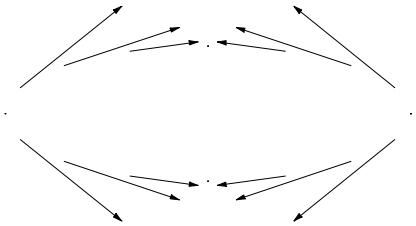
Out[87]= $\frac{1}{3\left(\frac{1}{5-4\,\text{Cos}[2\,t]}\right)^{3/2}}$



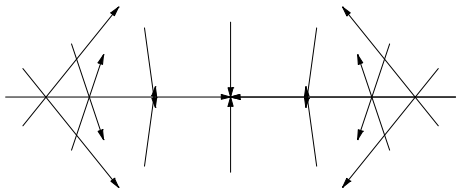
Out[88]= - Graphics -

que presenta máximos en t=π/2 y t=3π/2. Representemos la aceleración tangencial y normal en gráficos superpuestos:

```
In[89]:= << Graphics`PlotField`
         atlistplot = ListPlotVectorField[Table[{r[i], at[i]},
             {i, 0, 2 Pi, 2 Pi / 16}], AspectRatio → Automatic];
```

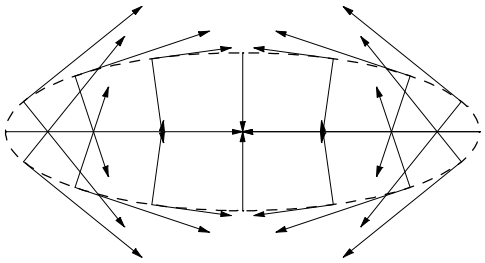


```
In[91]:= << Graphics`PlotField`
         anlistplot = ListPlotVectorField[Table[{r[i], an[i]},
             {i, 0, 2 Pi, 2 Pi / 16}], AspectRatio → Automatic];

         General::spell :
          Possible spelling error: new symbol name "anlistplot" is
            similar to existing symbols {atlistplot, nlistplot}.
```



Superpongamos los gráficos de la elipse y las componentes tangencial y normal de la aceleración:

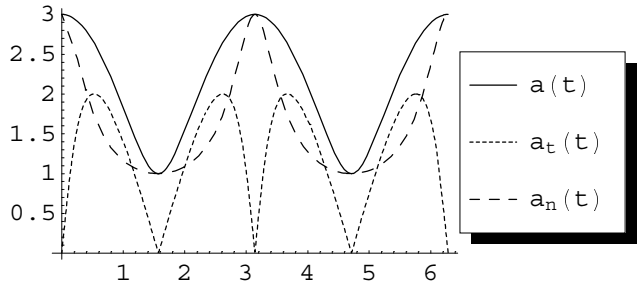
```
In[93]:= Show[elip, atlistplot, anlistplot]
```



```
Out[93]= - Graphics -
```

Observamos que la aceleración tangencial es nula en t=0,π/2,π, y 3π/2. Hagamos una representación gráfica conjunta de los módulos de la aceleración y de sus componentes intrínsecas

In[94]:=
```
<< Graphics`Legend`
Plot[{Sqrt[a[t].a[t]], Sqrt[at[t].at[t]], Sqrt[an[t].an[t]]}, {t, 0, 2 π},
 PlotStyle → {GrayLevel[0], Dashing[{.01}], Dashing[{.03}]},
 PlotLegend -> {"a(t)", "a_t(t)", "a_n(t)"},
 LegendPosition -> {1, -0.4}]
```



Out[95]= - Graphics -

Vemos que la aceleración total nunca es nula. También vemos que, cuando la aceleración normal es máxima, la aceleración tangencial es mínima y viceversa.

Ejercicio

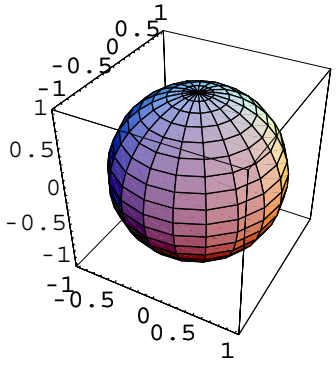
Calcular las componentes intrínsecas de la aceleración para una partícula moviéndose a lo largo de una hipérbola equilátera $y=1/x$ con $x(t)=t^2+1$, $y(t)=1/(t^2+1)$ en el intervalo $-0.5 \le t \le 0.5$. Representar gráficamente el radio de curvatura, las componentes intrínsecas de la aceleración y sus módulos en función del tiempo tal y como se hace en el texto.

■ Geometría de superficies

Superficies parametrizadas: plano tangente y área

Para una curva, la recta tangente en un punto juega un papel fundamental como aproximación lineal de la curva en dicho punto. Para una superficie es el plano tangente en un punto quien juega este papel. Consideremos por ejemplo la esfera de radio 1 parametrizada en coordenadas esféricas:

In[96]:=
```
Clear[r, u, v];
r[u_, v_] := {Cos[u] Sin[v], Sin[u] Sin[v], Cos[v]}
esfera = ParametricPlot3D[{Cos[u] Sin[v], Sin[u] Sin[v], Cos[v]}, {u, 0, 2 π}, {v, 0, π}];
```



Para encontrar el plano tangente a la esfera en el punto $(u,v)=(\pi/4,\pi/4) \to (x,y,z)=(1/2,1/2,1/\sqrt{2})$ podemos empezar por calcular el vector normal a la esfera en un punto $\vec{n} = \partial_u \vec{r}(u, v) \times \partial_v \vec{r}(u, v)$ (producto vectorial de los vectores tangentes a los paralelos $v=cte$, y a los meridianos $u=cte$, respectivamente) y particularizarlo a $(u,v)=(\pi/4,\pi/4)$, como:

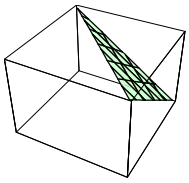
```
In[99]:= Clear[vn];
         vn = ∂u r[u, v] × ∂v r[u, v];
         Simplify[vn]
         vn0 = vn /. {u -> Pi / 4, v -> Pi / 4}
```

```
Out[101]= {-Cos[u] Sin[v]^2, -Sin[u] Sin[v]^2, -Cos[v] Sin[v]}
```

$$\text{Out[102]=}\quad \left\{-\frac{1}{2\sqrt{2}}, -\frac{1}{2\sqrt{2}}, -\frac{1}{2}\right\}$$

La ecuación del plano tangente a la superficie en un punto $\vec{r}_0$ vendrá dada entonces por $(\vec{r} - \vec{r}_0) \bullet \vec{n}_0 = 0$. Dibujémoslo:

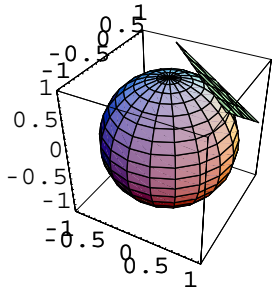
```
In[103]:= << Graphics`ContourPlot3D`
          ptesfera = ContourPlot3D[
            ({x, y, z} - r[Pi/4, Pi/4]).(vn /. {u → Pi/4, v → Pi/4}), {x, -1, 1}, {y, -1, 1}, {z, -1, 1}]
```



```
Out[104]= - Graphics3D -
```

y utilizando gráficos superpuestos:

```
In[105]:= Show[esfera, ptesfera]
```




```
Out[105]= - Graphics3D -
```

En vez de representar el plano tangente como una curva de nivel de

$F(x, y, z) = n_1(x - x_0) + n_2(y - y_0) + n_3(z - z_0) = 0$ usando **ContourPlot3D,** otra opción es despejar

$z = [-n_1(x - x_0) - n_0(y - y_0) + n_3 z_0] / n_3$ y usar **Plot3D**

In[106]:=
```
n1 = (vn /. {u → π/4, v → π/4})[[1]]
n2 = (vn /. {u → π/4, v → π/4})[[2]]
n3 = (vn /. {u → π/4, v → π/4})[[3]]
x0 = r[π/4, π/4][[1]]
y0 = r[π/4, π/4][[2]]
z0 = r[π/4, π/4][[3]]
```

Out[106]= $-\frac{1}{2\sqrt{2}}$

Out[107]= $-\frac{1}{2\sqrt{2}}$

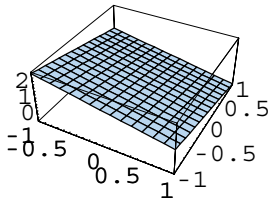
Out[108]= $-\frac{1}{2}$

Out[109]= $\frac{1}{2}$

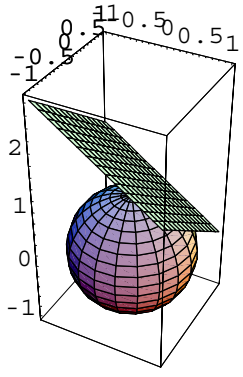
Out[110]= $\frac{1}{2}$

Out[111]= $\frac{1}{\sqrt{2}}$

In[112]:=
```
ptesfera2 = Plot3D[(-n1 (x - x0) - n2 (y - y0) + n3 z0)/n3, {x, -1, 1}, {y, -1, 1}];
```



In[113]:=
```
Show[esfera, ptesfera2];
```



Podemos calcular la superficie de la esfera mediante la integral $\int_0^{\pi}\left(\int_0^{2\pi} \| \partial_u \vec{r}(u, v) \times \partial_v \vec{r}(u, v) \| \, du\right) dv$ donde $\| \partial_u \vec{r}(u, v) \times \partial_v \vec{r}(u, v) \| = \|\vec{n}\|$ es el módulo del vector normal. En efecto, la integral:

In[114]:=
```
∫_0^π (∫_0^(2 π) √((∂_u r(u, v) × ∂_v r(u, v)) . (∂_u r(u, v) × ∂_v r(u, v))) du) dv
```

Out[114]= 4π

nos da la superficie de la esfera de radio 1.

Superficies en forma implícita y área

Para superficies dadas de forma implícita *F(x,y,z)=0* (consideraremos la forma explícita *z=S(x,y)* como un caso particular de forma implícita *F(x,y,z)=S(x,y)-z=0)* , recuerde que el vector gradiente $\vec{n} = \nabla F = (\partial_x F, \partial_y F, \partial_z F)$ es normal a la superficie *F(x,y,z)=0* en cada punto.

Nótese que, como un caso particular de parametrización $\vec{r}(u, v)$, siempre podemos tomar:

$x = u$

$y = v$

$z = S(u, v)$

entonces se tiene que los vectores tangentes a la superficie en las direcciones *x=cte=z* e *y=cte=z* son:

$\partial_u \vec{r} = \partial_x \vec{r} = (1, 0, \partial_x S)$

$\partial_v \vec{r} = \partial_y \vec{r} = (0, 1, \partial_y S)$

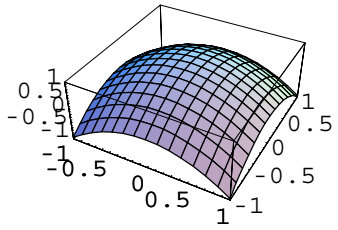
de donde se obtiene el vector normal a la superficie:

$\partial_u \vec{r} \times \partial_v \vec{r} = (-\partial_x S, -\partial_y S, 1)$

que coincide con $\nabla F = (\partial_x F, \partial_y F, \partial_z F)$ cuando *F(x,y,z)=S(x,y)-z.*

Tomemos como ejemplo un paraboloide:

In[115]:=
```
Clear[S, F, x, y]
S[x_, y_] := 1 - x^2 - y^2
F[x_, y_, z_] := S[x, y] - z
paraboloide = Plot3D[S[x, y], {x, -1, 1}, {y, -1, 1}]
```



Out[118]= - SurfaceGraphics -

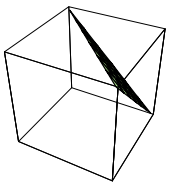
El vector normal a *S* en un punto arbitrario *(x,y,z)* es:

In[119]:=
```
Clear[vn]
<< Calculus`VectorAnalysis`
vn = Grad[F[x, y, z], Cartesian[x, y, z]]
```

Out[121]=
```
{-2 x, -2 y, -1}
```

El plano tangente al paraboloide en el punto *(x,y,z)=(1/2,1/2,1/2)* se representa como:

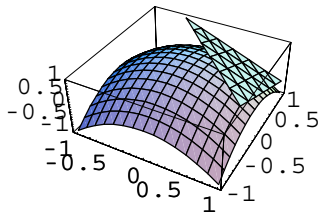
In[122]:=
```
<< Graphics`ContourPlot3D`
ptparaboloide = ContourPlot3D[
  ({x, y, z} - {1 / 2, 1 / 2, 1 / 2}) . (vn /. {x → 1/2, y → 1/2, z → 1/2}),
  {x, -1, 1}, {y, -1, 1}, {z, -1, 1}]
```



Out[123]=
```
- Graphics3D -
```

Que superpuesto al paraboloide queda:

In[124]:=
```
Show[paraboloide, ptparaboloide]
```



Out[124]=
```
- Graphics3D -
```

El área de la porción de paraboloide por encima del cuadrado [-0.5,0.5]×[-0.5,0.5] se puede calcular como $\int_{-0.5}^{0.5}\left(\int_{-0.5}^{0.5}\frac{1}{|\hat{n}\bullet\hat{k}|}\,dx\right)dy$. Más explícitamente:

In[125]:=
```
Abs[Integrate[Integrate[1 / (vn.{0,0,1} / Sqrt[vn.vn]), {x, -0.5, 0.5}], {y, -0.5, 0.5}]]
```

Out[125]=
```
1.28079
```

donde se ha tomado el valor absoluto debido a que la proyección $\hat{n}\bullet\hat{k}$ del vector normal calculado {-2 x,-2 y,-1} sobre el eje z es negativa.

Ejercicio

Hallar el plano tangente y el área de las siguientes superficies en el punto que se indica:

a) Superficie generada al revolucionar la curva: $y=\text{sen}(x)$, $0\le x<\pi$, en torno al eje X. Punto $x=\pi/2$.

b) Silla de montar: $z=y^2-x^2$, $-1\le x\le 1$, $-1\le y\le 1$. Punto (x,y,)=(1/2,1/2,0).

Operadores vectoriales en coordenadas curvilíneas

■ Coordenadas curvilíneas ortogonales. Factores de escala. Cambio de base

Cuando fijamos un sistema de referencia respecto al cual medir longitudes o ángulos, tres números bastan para identificar la posición de un punto en el espacio. Estos tres números pueden escogerse de múltiples formas. Dependiendo de la geometría del problema, unos sistemas de coordenadas pueden resultar más útiles que otros, en el sentido de que las ecuaciones implicadas admiten una forma más simple. *Mathematica* tiene incorporados varios sistemas de coordenadas dentro del paquete <<Calculus`VectorAnalysis`, entre ellos: coordenadas cartesianas (Cartesian), esféricas (Spherical) y cilíndricas (Cylindrical), que son los que nosotros utilizaremos mayormente.

```
Sistemas de coordenadas curvilíneas :
```

```
Bipolar                    Bispherical
Cartesian                  ConfocalEllipsoidal
ConfocalParaboloidal       Conical
Cylindrical                EllipticCylindrical
OblateSpheroidal           ParabolicCylindrical
Paraboloidal               ProlateSpheroidal
Spherical                  Toroidal
```

Por defecto, *Mathematica* trabaja en coordenadas cartesianas x, y, z. Para obtener el cambio de cartesianas a, por ejemplo, cilíndricas, debemos escribir:

```
In[1]:= << Calculus`VectorAnalysis`
        {ρ, ϕ, z} = CoordinatesFromCartesian[{x, y, z}, Cylindrical]
Out[2]= {√(x^2 + y^2), ArcTan[x, y], z}
```

que nos reproduce el conocido cambio de cartesianas a cilíndricas. El cambio inverso se puede obtener mediante

```
In[3]:= Clear[x, y, z, ρ, ϕ] (*es conveniente borrar antes los
          valores anteriores para evitar errores recursivos*)
        {x, y, z} = CoordinatesToCartesian[{ρ, ϕ, z}, Cylindrical]

Out[4]= {ρ Cos[ϕ], ρ Sin[ϕ], z}
```

El vector $\vec{\nabla}_\rho=\partial_\rho \vec{r}$ (tangente a a las curvas ϕ=cte, z=cte), el $\vec{\nabla}_\phi=\partial_\phi \vec{r}$ (tangente a a las curvas r=cte, z=cte) y el $\vec{\nabla}_z=\partial_z \vec{r}$ (tangente a a las curvas r=cte, ϕ=cte) se calculan como:

```
In[5]:= r = {x, y, z};
        vρ = ∂ρ r
        vϕ = ∂ϕ r
        vz = ∂z r
Out[6]= {Cos[ϕ], Sin[ϕ], 0}
Out[7]= {-ρ Sin[ϕ], ρ Cos[ϕ], 0}
Out[8]= {0, 0, 1}
```

De aquí obtenemos los factores de escala:

```
In[9]:= hρ = Simplify[√(vρ.vρ)]
        hϕ = Simplify[√(vϕ.vϕ)]
        hz = Simplify[√(vz.vz)]
Out[9]= 1
Out[10]= √(ρ^2)
Out[11]= 1
```

Los vectores unitarios tangentes a las curvas coordenadas r, θ, z son pues:

```
In[12]:= eρ = Simplify[vρ / hρ]
         eϕ = Simplify[vϕ / hϕ]
         ez = Simplify[vz / hz]
Out[12]= {Cos[ϕ], Sin[ϕ], 0}
Out[13]= {-ρ Sin[ϕ]/√(ρ^2), ρ Cos[ϕ]/√(ρ^2), 0}
Out[14]= {0, 0, 1}
```

y la matriz de paso de coordenadas cartesianas a cilíndricas se puede escribir como:

```
In[15]:= cartesf = {eρ, eϕ, ez}; MatrixForm[cartesf]
Out[15]//MatrixForm=
( Cos[ϕ]          Sin[ϕ]          0 )
( -ρ Sin[ϕ]/√(ρ^2)   ρ Cos[ϕ]/√(ρ^2)   0 )
(    0               0               1 )
```

Así, dado un campo de vectores $\vec{F}=F_x\,i+F_y\mathrm{j}+F_z\mathrm{k}$ en coordenadas cartesianas, las correspondientes componentes en coordenadas cilíndricas $\vec{F}=F_\rho\,e_\rho+F_\phi\,e_\phi+F_z\,e_z$ se calculan:

```
In[16]:= {Fρ, Fϕ, Fz} = cartesf.{Fx, Fy, Fz}
         General::spell1 : Possible spelling error: new
            symbol name "Fϕ" is similar to existing symbol "Fρ".
Out[16]= {Fx Cos[ϕ] + Fy Sin[ϕ], Fy ρ Cos[ϕ]/√(ρ^2) - Fx ρ Sin[ϕ]/√(ρ^2), Fz}
```

Por ejemplo, dado el campo de vectores: $\vec{F}=(x\,y,\ x^2+y^2,\ x/y)$ en coordenadas cartesianas, sus componentes en coordenadas cilíndricas son:

```
In[17]:= {Fρ, Fϕ, Fz} = Simplify[cartesf.{x y, x^2 + y^2, x / y}]
Out[17]= {1/2 ρ^2 (3 + Cos[2 ϕ]) Sin[ϕ], ρ √(ρ^2) Cos[ϕ]^3, Cot[ϕ]}
```

Mathematica dispone de comandos que realizan muchas de las anteriores operaciones. Por ejemplo, los vectores $\vec{v}_\rho=\partial_\rho \vec{r}$ (tangente a a las curvas ϕ=cte, z=cte), el $\vec{v}_\phi=\partial_\phi \vec{r}$ (tangente a a las curvas r=cte, z=cte) y el $\vec{v}_z=\partial_z \vec{r}$ (tangente a a las curvas r=cte, ϕ=cte) aparecen como las columnas de la matriz jacobiana de la transformación a coordenadas cilíndricas, la cual se obtiene como:

```
In[18]:= MatrixForm[JacobianMatrix[Cylindrical[ρ, ϕ, z]]]
```

Out[18]//MatrixForm=

$$\begin{pmatrix} \text{Cos}[\phi] & -\rho\,\text{Sin}[\phi] & 0 \\ \text{Sin}[\phi] & \rho\,\text{Cos}[\phi] & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

o sea,

```
In[19]:= v = Transpose[JacobianMatrix[Cylindrical[ρ, ϕ, z]]]
Out[19]= {{Cos[ϕ], Sin[ϕ], 0}, {-ρ Sin[ϕ], ρ Cos[ϕ], 0}, {0, 0, 1}}
```

nos da una lista donde, por ejemplo, la primera fila

```
In[20]:= v[[1]]
Out[20]= {Cos[ϕ], Sin[ϕ], 0}
```

coincide con v_ρ calculado anteriormente. También existe un comando para obtener los factores de escala :

```
In[21]:= h = ScaleFactors[Cylindrical[ρ, ϕ, z]]
Out[21]= {1, ρ, 1}
```

Así, la matriz ortonormal de cambio de coordenadas cartesianas a esféricas se obtiene dividiendo los vectores tangentes por sus módulos (los factores de escala) como:

```
In[22]:= e = v / h
Out[22]= {{Cos[ϕ], Sin[ϕ], 0}, {-Sin[ϕ], Cos[ϕ], 0}, {0, 0, 1}}
```

que coincide con la matriz **cartesf** calculada anteriormente, cuyas filas son los vectores unitarios e_ρ, e_ϕ y e_z. En efecto:

```
In[23]:= e[[1]]
         e[[2]]
         e[[3]]
Out[23]= {Cos[ϕ], Sin[ϕ], 0}
Out[24]= {-Sin[ϕ], Cos[ϕ], 0}
Out[25]= {0, 0, 1}
```

Vemos pues el significado y la utilidad de los comandos **JacobianMatrix** y **ScaleFactors**, que hacen más fácil el cálculo de la matriz de cambio de base. Por otra parte, tenemos también un comando para el determinante de la matriz jacobiana:

```
In[26]:= JacobianDeterminant[Cylindrical[ρ, ϕ, z]]
Out[26]= ρ
```

que coincide con el producto de los factores de escala. Este jacobiano aparece, como sabemos, en los cambios de coordenadas cartesianas a cilíndricas en integrales triples.

Ejercicio

Obtener la expresión del campo $\vec{F} = (F_x, F_y, F_z) = \frac{(yz, xz, xy)}{\sqrt{x^2+y^2+z^2}}$ en coordenadas esféricas $\vec{F} = (F_r, F_\phi, F_\theta)$

■ Operadores vectoriales en coordenadas curvilíneas

Estamos en condiciones de obtener la expresión de los operadores vectoriales: Gradiente, Divergencia, Rotacional y Laplaciano, sobre funciones escalares y vectoriales dadas en diferentes sistemas de coordenadas. El sistema de coordenadas en que *Mathematica* trabaja por defecto es

```
In[27]:= {CoordinateSystem, Coordinates[]}
Out[27]= {Cartesian, {Xx, Yy, Zz}}
```

es decir, coordenadas cartesianas representadas por defecto por: "Xx,Yy,Zz". Si queremos usar la nomenclatura usual: "x,y,z", o cualquier otra a nuestro gusto, deberemos usar el comando:

```
In[28]:= Clear[x, y, z];
         SetCoordinates[Cartesian[x, y, z]]
Out[29]= Cartesian[x, y, z]
```

donde es conveniente borrar de memoria cualquier valor previo de x,y,z. Definamos una función escalar **f** y otra vectorial **v** arbitrarias:

```
In[30]:= f[x,y,z]
         v={vx[x,y,z],vy[x,y,z],vz[x,y,z]}
Out[30]= f[x, y, z]
Out[31]= {vx[x, y, z], vy[x, y, z], vz[x, y, z]}
```

Podemos obtener la expresión general del gradiente de **f** en las coordenadas por defecto (cartesianas) escribiendo:

```
In[32]:= Grad[f[x, y, z]]
Out[32]= {f^(1,0,0)[x, y, z], f^(0,1,0)[x, y, z], f^(0,0,1)[x, y, z]}
```

donde $f^{(1,0,0)}[x, y, z]$ simboliza la derivada parcial primera de **f** respecto a **x** (la primera coordenada), y así sucesivamente. El rotacional de **v** en cartesianas se obtiene :

```
In[33]:= Curl[v]
Out[33]= {-vy^(0,0,1)[x, y, z] + vz^(0,1,0)[x, y, z],
          vx^(0,0,1)[x, y, z] - vz^(1,0,0)[x, y, z],
          -vx^(0,1,0)[x, y, z] + vy^(1,0,0)[x, y, z]}
```

y la divergencia de **v**:

```
In[34]:= Div[v]
Out[34]= vz^(0,0,1)[x, y, z] + vy^(0,1,0)[x, y, z] + vx^(1,0,0)[x, y, z]
```

Proving the well known identity

Probemos que la divergencia del rotacional es nula: $\nabla \bullet (\nabla x \vec{v}) = 0$:

```
In[35]:= Div[Curl[v]]
Out[35]= 0
```

así como que el rotacional del gradiente es también cero: $\nabla x(\nabla f) = 0$

```
In[36]:= Curl[Grad[f[x,y,z]]]
Out[36]= {0, 0, 0}
```

El comando **Laplacian** calcula el laplaciano $\Delta=\nabla^2 =\nabla\cdot\nabla$ de una función escalar

```
In[37]:= Laplacian[f[x,y,z]]
Out[37]= f^(0,0,2)[x, y, z] + f^(0,2,0)[x, y, z] + f^(2,0,0)[x, y, z]
```

Podemos ver la expresión de estos operadores (gradiente, rotacional, divergencia y laplaciano) en otros sistemas de coordenadas. Por ejemplo, establezcamos como sistema de coordenadas por defecto el sistema de coordenadas esféricas:

```
In[38]:= Clear[r, θ, ϕ];
         SetCoordinates[Spherical[r, θ, ϕ]]
Out[39]= Spherical[r, θ, ϕ]
```

El rango de variación de las coordenadas r,θ,ϕ en el sistema de coordenadas por defecto (ahora el esférico) viene dado por:

```
In[40]:= CoordinateRanges[]
Out[40]= {0 ≤ r < ∞, 0 ≤ θ ≤ π, -π < ϕ ≤ π}
```

El gradiente de una función escalar en coordenadas esféricas es ahora:

```
In[41]:= Grad[f[r, θ, ϕ]]
Out[41]= {f^(1,0,0)[r, θ, ϕ], f^(0,1,0)[r, θ, ϕ]/r, Csc[θ] f^(0,0,1)[r, θ, ϕ]/r}
```

y el laplaciano:

```
In[42]:= Laplacian[f[r,θ,ϕ]]
Out[42]= 1/r^2 (Csc[θ] (Csc[θ] f^(0,0,2)[r, θ, ϕ] +
           Cos[θ] f^(0,1,0)[r, θ, ϕ] + Sin[θ] f^(0,2,0)[r, θ, ϕ] +
           2 r Sin[θ] f^(1,0,0)[r, θ, ϕ] + r^2 Sin[θ] f^(2,0,0)[r, θ, ϕ]))
```

La divergencia y el rotacional de un campo vectorial $\vec{V}(r, \theta, \phi) = V_r\,\hat{e}_r + V_\theta\,\hat{e}_\theta + V_\phi\,\hat{e}_\phi$ dado en esféricas tiene la siguiente "pinta":

```
In[43]:= Div[{Vr, Vθ, Vϕ}]
         Curl[{Vr, Vθ, Vϕ}]
         General::spell : Possible spelling error: new symbol
            name "Vϕ" is similar to existing symbols {Fϕ, Vθ}.
Out[43]= (Csc[θ] (r Vθ Cos[θ] + 2 r Vr Sin[θ]))/r^2
Out[44]= {Vϕ Cot[θ]/r, -Vϕ/r, -Vθ/r}
```

Por ejemplo, tomemos como caso particular el campo gravitatorio $\vec{G} = \frac{1}{r^2}\,\hat{r}$, y el campo $\vec{F} = \frac{1}{r^3}\,\hat{r}$ los cuales escribiremos como:

```
In[45]:= G = {1 / r^2, 0, 0};
         F = {1 / r^3, 0, 0};
```

La divergencia y el rotacional de estor campos es:

```
In[47]:= Div[G]
         Div[F]
         Curl[G]
         Curl[F]
```

Out[47]= 0

Out[48]= $-\frac{1}{r^4}$

Out[49]= {0, 0, 0}

Out[50]= {0, 0, 0}

Que nos muestran que (salvo en *r=0* donde dichos campos no están definidos) el campo gravitatorio es solenoidal (tiene divergencia nula) y conservativo (rotacional nulo), mientras que el inverso del cubo de la distancia es también conservativo, pero no es solenoidal.

Podemos calcular el rotacional y la divergencia del campo de "remolino":
$\vec{V}(r, \phi, z) = V_\rho \hat{e}_\rho + V_\phi \hat{e}_\phi + V_z \hat{k} = \frac{1}{\rho^2} \hat{e}_\phi$ en cilíndricas sin necesidad de cambiar el sistema de coordenadas por defecto, mediante la opción:

```
In[51]:= Clear[ρ, ϕ, z];
         Curl[{0, 1 / ρ^2, 0}, Cylindrical[ρ, ϕ, z]]
         Div[{0, 1 / ρ^2, 0}, Cylindrical[ρ, ϕ, z]]
```

Out[52]= $\left\{0, 0, -\frac{1}{\rho^3}\right\}$

Out[53]= 0

que nos dice que el "remolino" es solenoidal (divergencia nula) e irrotacional (no conservativo).

Otros comandos útiles son:

`CrossProduct[v1,v2]` que proporciona el producto vectorial,

`DotProduct[v1,v2]` que proporciona el producto escalar y

`ScalarTripleProduct[v1,v2,v3]` que da el producto escalar triple `v1•(v2 x v3)` (el volumen del paralelepípedo definido por los vectores **`v1,v2,v3`**)

de vectores dados en un cierto sistema de coordenadas. Por ejemplo, los vectores

```
In[54]:= v1 = {1, 1, 1};
         v2 = {1, 1, 0};
```

(dados en coordenadas cartesianas (x,y,z)) se escriben en coordenadas esféricas (r,θ,ϕ) como:

```
In[56]:= v1e = CoordinatesFromCartesian[v1, Spherical]
         v2e = CoordinatesFromCartesian[v2, Spherical]
```

Out[56]= $\left\{\sqrt{3}\,,\ \text{ArcCos}\left[\frac{1}{\sqrt{3}}\right],\ \frac{\pi}{4}\right\}$

Out[57]= $\left\{\sqrt{2}\,,\ \frac{\pi}{2},\ \frac{\pi}{4}\right\}$

No obstante, su producto escalar no depende del sistema de coordenadas elegido:

```
In[58]:= DotProduct[v1e, v2e, Spherical]
         DotProduct[v1, v2, Cartesian]
```

Out[58]= 2

Out[59]= 2

Ejercicios

1) Dado el campo : $\vec{F}(x, y, z) = \left(\frac{x}{x^2 + y^2}, \frac{y}{x^2 + y^2}, 0\right)$ en coordenadas cartesianas,
proporcione su expresión en coordenadas cilíndricas y
calcule su divergencia y su rotacional. ¿Es conservativo?, ¿es solenoidal?

2) Dados los siguientes campos escalares f y vectoriales $\vec{V}$ en coordenadas:

cilíndricas:

$f_1(\rho, \phi, z) = \rho^3 z^4 \sin(\phi)$

$\vec{V}_1(\rho, \phi, z) = \{z^2, 2, \rho\}$

$f_2(\rho, \phi, z) = z^3 \rho$

$\vec{V}_2(\rho, \phi, z) = \{\cos(\phi), -\sin(\phi), 0\}$

y esféricas:

$f_3(r, \theta, \phi) = r \sin(\phi)$

$\vec{V_3}(r, \theta, \phi) = \{\sin(\phi),\ 0,\ \cos(\theta)\}$

Calcular:

2.a) El gradiente, la divergencia y el rotacional. ¿Qué campos son conservativos?, ¿cuáles son solenoidales?.

2.b) Verifique que el rotacional del gradiente y la divergencia del rotacional son nulos

3) Verifique que:

$\psi_1(\rho,\phi,z) = \rho^n \cos(n\,\phi)$, $\psi_2(\rho,\phi,z) = \rho^n \sin(n\,\phi)$, $\psi_3(\rho,\phi,z) = \ln\rho$, $\psi_4(\rho,\phi,z) = \frac{1}{\sqrt{\rho^2+z^2}}$

son soluciones de la ecuación de Laplace $\nabla^2\psi = 0$ en coordenadas cilíndricas.

Ecuaciones diferenciales ordinarias

■ Soluciones exactas con DSolve

Resumen de comandos

DSolve[edo==0,y,x] da la solución general de una EDO en términos de la función **y[x]**.

DSolve[{edo==0,condinic},y,x] resuelve una EDO en términos de la función **y[x]** con condiciones iniciales **condinic**.

DSolve[{edo1==0,edo2==0,...},{y1,y2,...},x] resuelve un sistema de EDOs en términos de las funciones **y1[x], y2[x]**, etc.

DSolve[edp==0,y,{x1,x2,...}] resuelve una EDP (ecuación en derivadas parciales) en términos de la función **y[x1,x2,...]**.

el paquete **<<Calculus`DSolveIntegrals`** proporciona "integrales completas" para EDPs mediante el comando:

CompleteIntegral[edp==0,y,{x1,x2,...}]

El comando **`DSolve`** encuentra soluciones analíticas (no numéricas) para un número cada vez mayor de ecuaciones diferenciales. Por ejemplo:

■ `DSolve` resuelve EDOs lineales con coeficientes constantes de cualquier orden y muchas ecuaciones lineales de hasta segundo orden con coeficientes no constantes.

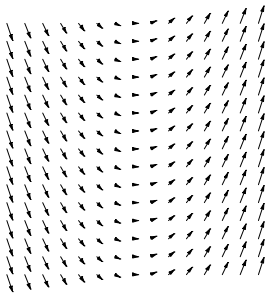
■ `DSolve` incluye procedimientos generales para EDOs no lineales cuyas soluciones vienen tratadas en libros de texto y manuales.

■ `DSolve` puede encontrar soluciones generales para ecuaciones en derivadas parciales lineales (y "débilmente no lineales"). Téngase en cuenta que las ecuaciones diferenciales en derivadas parciales no lineales no presentan usualmente una solución general explícita (éste es un campo de investigación importante donde es necesario avanzar)

Ejemplos de EDOs de orden 1. Isoclinas y método de Euler

Consideremos la ecuación diferencial $y'=x$. El campo de direcciones (isoclinas) asociado a esta ecuación diferencial es $\vec{v}(x, y) = (1, y') = (1, x)$, que representan vectores tangentes a la trayectoria $y(x)$ en cada punto. La representación gráfica del mismo se realiza mediante:

```
In[1]:= << Graphics`PlotField`
        isoclinas = PlotVectorField[{1, x}, {x, -3, 3}, {y, -3, 3}];
```



La solución general de la ecuación diferencial $y'=x$ viene dada por:

```
In[3]:= DSolve[y'[x] == x, y[x], x]
```

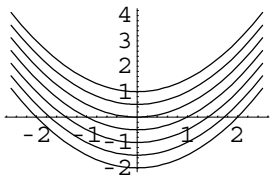
$$\text{Out[3]= } \left\{\left\{y[x] \to \frac{x^2}{2} + C[1]\right\}\right\}$$

que nos da una familia uniparamétrica de curvas, donde C[1] representa la constante de integración. Representemos varias curvas de esta familia pero, para ello, definamos la familia *y(x,c)* para distintas condiciones iniciales $y(0) = c$ como una función **sol[x,c]** de dos variables:

```
In[4]:= Clear[y, x, c];
        sol[x_, c_] := y[x] /. DSolve[{y'[x] == x, y[0] == c}, y[x], x]
```

Definamos una tabla con los 7 miembros c=-2,-1.5,...,0.5,1 de dicha familia uniparamétrica de curvas y representémosla gráficamente:

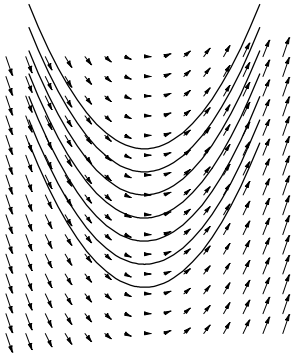
```
In[6]:= familia = Table[sol[x, c], {c, -2, 1, 0.5}];
        familiaplot = Plot[Evaluate[familia], {x, -2.5, 2.5}]
```




```
Out[7]= - Graphics -
```

Superponiendo dichas curvas con el campo de direcciones (isoclinas):

```
In[8]:= Show[isoclinas, familiaplot]
```



```
Out[8]= - Graphics -
```

vemos que el campo de direcciones es, efectivamente, tangente a la familia de curvas en cada punto.

Ejercicio

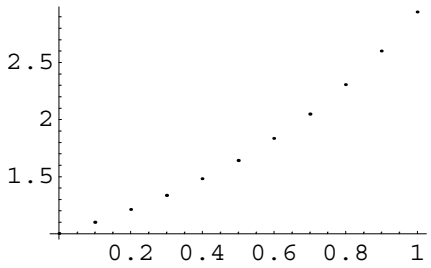
Dibuje el campo de direcciones (isoclinas) de la ecuación diferencial $y'=x^3-x$ en el cuadrado $(x,y)\in[-2,2]\times[-3,3]$ y superponga dicho gráfico al de la familia de soluciones con condiciones iniciales $y(0)=c=-1,-0.5,0,0.5,1$

Método de Euler: Aunque *Mathematica* dispone de métodos numéricos relativamente exactos para resolver ecuaciones diferenciales, diseñemos nosotros mismos un algoritmo para un método "grosero" (poco preciso, no refinado) cómo es el de Euler. Esto nos sirve como "modelo de juguete" para saber cualitativamente cómo funcionan otros métodos numéricos más sofisticados y cómo evaluar el error numérico cometido. Por ejemplo, tomemos la ecuación lineal no homogénea de primer orden $y' = f(x, y) = x^2 + y$, con la condición inicial $y(0)=1$. Aunque sabemos resolver dicha ecuación de forma exacta, abordaremos el problema de forma numérica y compararemos la solución numérica con la exacta para evaluar el error cometido y así hacer un test a nuestro método numérico. Supongamos que queremos una solución numérica de la EDO anterior en el intervalo $x\in[0,1]$. Para ello dividimos el intervalo [0,1] en, por ejemplo, 10 trozos, tomando un paso $h=0.1$ y obteniendo así 11 puntos equiespaciados: $x_0 = 0,\ x_1 = 0.1,\ \ldots,\ x_k = x_0 + k\,h,\ \ldots,\ x_{10} = 1$. Introduzcamos dicha información en el programa:

```
In[9]:= Clear[f, x, y, h, k];
        f[x_, y_] := x^2 + y;
        h = 0.1; x[0] = 0;
        x[k_] := x[0] + k * h
```

Discretizando la derivada $y'(x_k) \approx (y(x_{k+1}) - y(x_k))/h = (y_{k+1} - y_k)/h$ obtenemos la versión en diferencias finitas $y_k = y_{k-1} + h\, f(x_{k-1}, y_{k-1})$ de la ecuación diferencial $y' = f(x, y)$. Introduzcamos dicha ecuación en el programa y representemos los 11 puntos $(x_0, y_0),\ \ldots,\ (x_{10}, y_{10})$:

```
In[13]:= y[0] = 1;
         y[k_] := y[k - 1] + h * f[x[k - 1], y[k - 1]];
         eulerplot = ListPlot[Table[{x[k], y[k]}, {k, 0, 10}]]
```

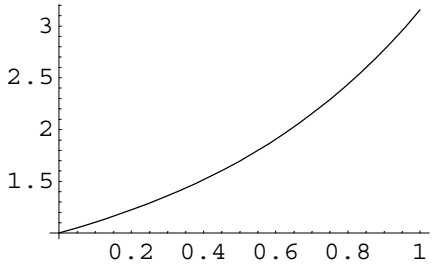



```
Out[15]= - Graphics -
```

Calculemos la solución exacta y representémosla gráficamente (cambiaremos x↔t, y↔z para evitar incompatibilidades)

```
In[16]:= exacta = DSolve[{z'[t] == t^2 + z[t], z[0] == 1}, z, t]
         exactaplot = Plot[z[t] /. exacta, {t, 0, 1}]
```

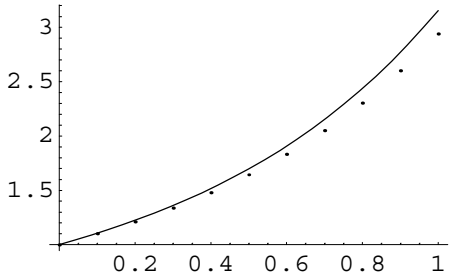
```
Out[16]= {{z → (-2 + 3 e^#1 - 2 #1 - #1^2 &)}}
```




```
Out[17]= - Graphics -
```

Superpongamos la solución exacta y la numérica:

In[18]:=
```
Show[eulerplot, exactaplot]
```



Out[18]= - Graphics -

Vemos cómo la solución numérica se aparta progresivamente de la exacta conforme nos alejamos del punto inicial $(x_0, y_0) = (0, 1)$. Podemos evaluar el error cometido en el último punto (x_{10}, y_{10}) como $y(1) - y_{10}$:

In[19]:=
```
Evaluate[z[1] /. exacta] - y[10]
```

Out[19]= {0.214244}

Es decir, el error es de 0.214244

Ejercicio

Repita los pasos anteriores para 21 puntos, es decir, tomando como paso la mitad del anterior: *h=0.05.* Evalúe el error cometido en el cálculo numérico de *y(1).* ¿Es el error mayor o menor que para el caso *h=0.1*?.

Ejemplos de EDOs de orden 2. Diagrama de fases

Consideremos la ecuación diferencial lineal de segundo orden con coeficientes constantes x*''(t)=kx(t)* que puede representar la ecuación de un oscilador armónico simple atractivo para *k<0* y repulsivo para *k>0.* La solución general de esta ecuación se obtiene:

In[20]:=
```
Clear[x, t];
DSolve[x''[t] == k x[t], x[t], t]
```

Out[21]= $\left\{\left\{x[t] \to e^{-\sqrt{k}\, t}\, C[1] + e^{\sqrt{k}\, t}\, C[2]\right\}\right\}$

Vemos que contiene dos constantes arbitrarias C[1] y C[2], como corresponde a una EDO de orden 2.

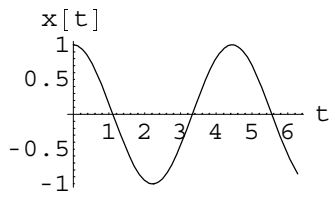
Tomemos el oscilador atractivo con $k = -\omega_0^2$ =-2 y condiciones iniciales x*(0)=1, x'(0)=0* (velocidad inicial nula); ahora la solución será:

In[22]:=
```
xoscatract = DSolve[{x''[t] == -2 x[t], x[0] == 1, x'[0] == 0}, x, t]
```

Out[22]= $\{\{x \to \left(\text{Cos}\left[\sqrt{2}\ \#1\right]\ \&\right)\}\}$

que representa un movimiento armónico simple de amplitud 1 y frecuencia $\omega_0 = \sqrt{2}$. Para representar gráficamente la posición **x** en función del tiempo **t** escribiremos:

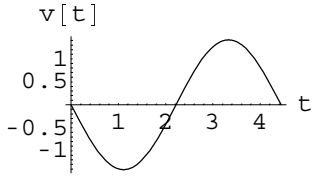
In[23]:=
```
Plot[x[t] /. xoscatract, {t, 0, 2 Pi}, AxesLabel -> {"t", "x[t]"}]
```



Out[23]= - Graphics -

donde observamos que el periodo del movimiento es $T=\frac{2\pi}{\omega_0} = 2\pi \Big/ \sqrt{2} \approx 4.44288$. Para representar la velocidad **v[t]=x'[t]** en función del tiempo escribimos:

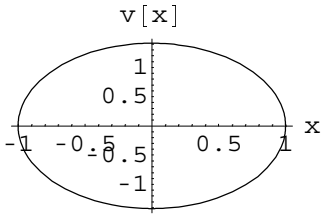
```
In[24]:= Plot[x'[t] /. xoscatract,
           {t, 0, 2 Pi / Sqrt[2]}, AxesLabel -> {"t", "v[t]"}]
```




```
Out[24]= - Graphics -
```

También podemos hacer una representación de la velocidad **v[x]** en función de la posición **x** ("diagrama de fases") por medio de un gráfico en forma paramétrica:

```
In[25]:= ParametricPlot[Evaluate[{x[t], x'[t]} /. xoscatract],
           {t, 0, 2 Pi / Sqrt[2]}, AxesLabel -> {"x", "v[x]"}]
```



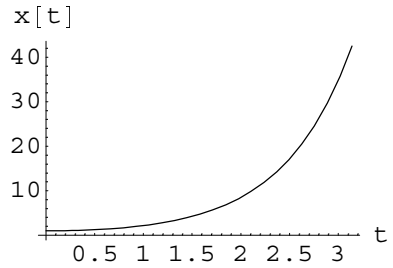

```
Out[25]= - Graphics -
```

que nos da una órbita cerrada (elipse), característica de los sistemas periódicos.

Veamos qué pasa con el oscilador repulsivo para *k=2* e idénticas condiciones iniciales que para el atractivo

```
In[26]:= xoscrepul = DSolve[{x''[t] == 2 x[t], x[0] == 1, x'[0] == 0}, x, t]
         Plot[x[t] /. xoscrepul, {t, 0, Pi}, AxesLabel -> {"t", "x[t]"}]
```

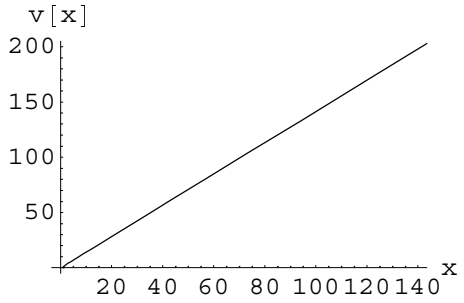
```
Out[26]= {{x → (ℯ^(-Sqrt[2] #1) (1/2 + 1/2 ℯ^(2 Sqrt[2] #1)) &)}}
```




```
Out[27]= - Graphics -
```

Vemos que la partícula se aleja de su posición inicial *x(0)=1* de forma exponencial. La trayectoria no se repite a intervalos de tiempo fijos *T;* es decir, a diferencia del oscilador atractivo, el oscilador repulsivo no es un sistema periódico. Veamos que sus trayectorias en el plano de fases tampoco son cerradas:

```
In[28]:= ParametricPlot[Evaluate[{x[t], x'[t]} /. xoscrepul],
          {t, 0, 2 Pi / Sqrt[2]}, AxesLabel -> {"x", "v[x]"}]
```




```
Out[28]= - Graphics -
```

Ejercicio

Dibuje la trayectoria en el plano **x-t** y en el plano de fases **x-x'** de un oscilador armónico simple atractivo con $k = -\omega_0^2 = -5$ que parte de la posición *x(0)=0* con velocidad *x'(0)=1*. Tome el intervalo $t \in [0, T = \frac{2\pi}{\omega_0}]$.

Sistemas de EDOs

También podemos resolver sistemas de EDOs,introduciendo éstas mediante una lista.Por ejemplo, para resolver el sistema de dos EDOs lineales con coeficientes constantes $\begin{cases} y_1 = -y_2' \\ y_2 = -y_1' \end{cases}$ escribiremos:

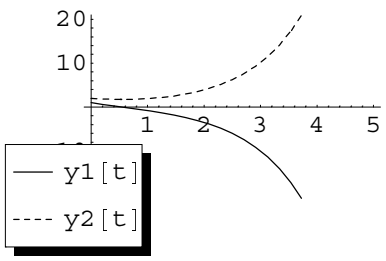
```
In[29]:= DSolve[{y1[x] == -y2'[x], y2[x] == -y1'[x]}, {y1[x], y2[x]}, x]
```

$$\text{Out[29]= } \left\{\left\{y1[x] \to \frac{1}{2} e^{-x} (C[1] + e^{2x} C[1] + C[2] - e^{2x} C[2]),\right.\right.$$
$$\left.\left. y2[x] \to -\frac{1}{2} e^{-x} (-C[1] + e^{2x} C[1] - C[2] - e^{2x} C[2])\right\}\right\}$$

que nos da la solución general dependiente de dos constantes arbitrarias C[1] y C[2].Veamos cómo calcular la solución particular que pasa por **{y1[0],y2[0]}={1,2}** y cómo representar **y1[t]** e **y2[t]** superpuestas en un mismo gráfico:

```
In[30]:= sistema = DSolve[{y1[x] == -y2'[x],
            y1[0] == 1, y2[x] == -y1'[x], y2[0] == 2}, {y1, y2}, x]
         << Graphics`Legend`
         Plot[{Evaluate[y1[t] /. sistema], Evaluate[y2[t] /. sistema]},
          {t, 0, 5}, PlotStyle -> {GrayLevel[0.], Dashing[{0.02, 0.02}]},
          PlotLegend -> {"y1[t]", "y2[t]"}]
```

$$\text{Out[30]= } \left\{\left\{y1 \to \left(-\frac{1}{2} e^{-\#1} (-3 + e^{2\#1}) \&\right), y2 \to \left(\frac{1}{2} e^{-\#1} (3 + e^{2\#1}) \&\right)\right\}\right\}$$




```
Out[32]= - Graphics -
```

Vemos que ambas funciones se separan indefinidamente de forma exponencial.

Ejercicio

Resolver el sistema $\begin{cases} y_1' = y_1 \\ y_2' = -y_1 + 2\,y_2 \end{cases}$ con condiciones iniciales $y_1(0) = -1,\ y_2(0) = 1$ y representar superpuestas las gráficas de $y_1(x)$ e $y_2(x)$ en el intervalo x∈[0,10].

Ecuaciones en derivadas parciales

Finalmente, veamos cómo introducir una ecuación en derivadas parciales como, por ejemplo,
$\partial_x U(x, y, z) = x^2\, y/z$

```
In[33]:= DSolve[Derivative[1, 0, 0][U][x, y, z] == x^2 y / z,
           U[x, y, z], {x, y, z}]
```

$$\text{Out[33]= } \left\{\left\{\text{U[x, y, z]} \to \frac{\text{x}^3\ \text{y}}{3\ \text{z}} + \text{C[1][y, z]}\right\}\right\}$$

que nos da la función $U(x,y,z)=\frac{x^3\,y}{3\,z}$ salvo una función arbitraria $C_1(x, y)$ dependiente de sólo dos de las tres variables independientes. También se dispone del paquete:

```
In[34]:= << Calculus`DSolveIntegrals`
         CompleteIntegral[
          Derivative[1, 0, 0][U][x, y, z] == yz, U[x, y, z], {x, y, z}]
```

$$\text{Out[35]= } \{\{\text{U[x, y, z]} \to \text{x yz} + \text{B[1]} + \text{y B[2]} + \text{z B[3]}\}\}$$

Ejercicio

Resuelva la ecuación $y\,\partial_x U(x, y) + x\,\partial_y U(x, y) = 0$

■ Soluciones numéricas con NDSolve. Ecuación de Van der Pol

El comando `NDSolve` encuentra soluciones numéricas de ecuaciones diferenciales. Selecciona de forma automática el algoritmo óptimo a usar, o nos permite que lo seleccionemos nosotros si lo conocemos previamente. *Mathematica,* usa automáticamente funciones interpoladoras en vez de listas de números para representar las soluciones numéricas de las ecuaciones diferenciales, haciendo más factible la manipulación posterior de dichas soluciones como, por ejemplo, su integración, derivación, etc.

El comando `NDSolve` utiliza los siguientes algoritmos:

• Por defecto, o mediante la opción `Method->Automatic`, `NDSolve` cambia entre un método de Adams no rígido y un método de Adams rígido adaptado.

• El método de Adams se toma con orden entre 1 y 12

• El método de la fórmula de diferencias atrasadas se toma con orden entre 1 y 5.

• Un orden 4–5 para el método Runge- Kutta se aplica a ecuaciones no rígidas.

• Para problemas lineales con condiciones en la frontera, `NDSolve` emplea el método de búsqueda de Gel'fand- Lokutsiyevskii.

• Para ecuaciones en derivadas parciales de dos variables se utiliza el llamado método de las lineas.

Los algoritmos adaptativos de *Mathematica* y su control de una precisión arbitraria garantiza la precisión de los resultados sin errores de redondeo o soluciones espúreas debidas a errores numéricos.

Ejemplo: solución numérica de la ecuación de Van der Pol

Veamos cómo *Mathematica* es capaz de proporcionar una solución numérica para una EDO no lineal como la ecuación de Van der Pol

$$x''[t] - 0.2\,(1 - x[t]^2)\,x'[t] + x[t] = 0.$$

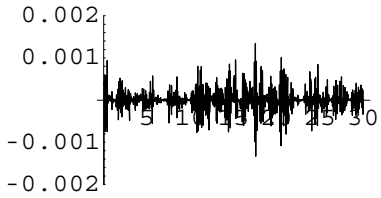
Esta ecuación aparece en el estudio de las oscilaciones de un circuito RCL con resistencia variable dependiente de la amplitud *x(t).*

La solución numérica con velocidad inicial nula *x'(0)=0* y posición inicial *x(0)=1* en los primeros 30 segundos (es decir, en el intervalo $0 \le t \le 30$) es:

```
In[36]:= solu1 = NDSolve[{x''[t] - 0.2 (1 - x[t]^2) x'[t] + x[t] == 0,
            x[0] == 1, x'[0] == 0}, x, {t, 0, 30}]
Out[36]= {{x → InterpolatingFunction[{{0., 30.}}, <>]}}
```

Podemos verificar que la solución numérica obtenida verifica con bastante precisión la ecuación diferencial. En efecto, la representación gráfica de $x''[t] - 0.2\,(1 - x[t]^2)\,x'[t] + x[t]$ evaluada en la solución **solu1** anterior

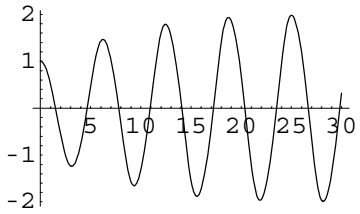
```
In[37]:= Plot[N[Evaluate[x''[t] - 0.2 (1 - x[t]^2) x'[t] + x[t] /. solu1], 3],
          {t, 0, 30}, PlotRange -> {-0.002, 0.002}]
```




```
Out[37]= - Graphics -
```

nos dice que la solución numérica **solu1** verifica la ecuación diferencial en el intervalo t∈[0,30] salvo errores del orden de 0.001. Una vez que hemos ganado confianza en nuestra solución, representémosla en el intervalo escogido:

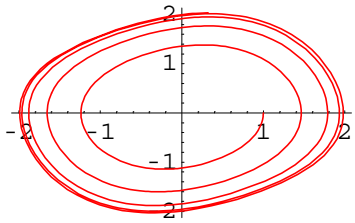
```
In[38]:= Plot[x[t] /. solu1, {t, 0, 30}]
```




```
Out[38]= - Graphics -
```

No queda claro en el gráfico anterior si dicha solución es periódica o no. Una representación en el plano de fases: "*v(x) frente a x*"

```
In[39]:= ParametricPlot[Evaluate[{x[t], x'[t]} /. solu1[[1]]],
          {t, 0, 30}, PlotStyle -> {Thickness[0.005], RGBColor[1, 0, 0]}]
```

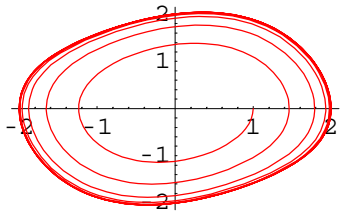



```
Out[39]= - Graphics -
```

nos muestra que la órbita no es cerrada (al menos en el intervalo [0,30]), lo que da idea de "falta de periodicidad". No obstante, nótese que la órbita se acerca "asintóticamente" ("a largo plazo") a una órbita cerrada "atractriz". Para ver este comportamiento asintótico de la órbita con mayor nitidez, podemos extender el intervalo de tiempo de 30 a 90 segundos y aumentar el número máximo de pasos que **NDSolve** utiliza (por defecto 500) a 2000 mediante la opción **MaxSteps->2000** en:

In[40]:=
```
Clear[x] (* es necesario borrar antes el valor de x*)

solu2 = NDSolve[{x''[t] - 0.2 (1 - x[t]^2) x'[t] + x[t] == 0,
   x[0] == 1, x'[0] == 0}, x, {t, 0, 90}, MaxSteps -> 2000]

ParametricPlot[Evaluate[{x[t], x'[t]} /. solu2[[1]]],
 {t, 0, 90}, PlotStyle -> {Thickness[0.005], RGBColor[1, 0, 0]}]
```

Out[41]=
```
{{x → InterpolatingFunction[{{0., 90.}}, <>]}}
```



Out[42]=
```
- Graphics -
```

Aquí se observa con mayor nitidez que existe una órbita "límite, asintótica o atractriz" (nótese la zona de mayor densidad de líneas) cerrada donde converge (a la cual se acerca cada vez más) nuestro sistema. Esto quiere decir que, aunque el sistema no sea periódico, se comporta como tal a largo plazo (este es un caso particular de un teorema general debido a Alfred Liénard).

Ejercicio

Hallar la solución numérica de la ecuación de Van der Pol con velocidad inicial $x'(0)=2$ y posición inicial $x(0)=0$ en los primeros 50 segundos (es decir, en el intervalo $0 \le t \le 50$) y representar las gráficas: **x-t** (posición frente a tiempo) y **x'-x** (velocidad frente a posición).

Oscilaciones amortiguadas y forzadas

■ Oscilador armónico amortiguado

Solución general

Recuerde que la segunda ley de Newton ($\vec{F} = m\,\vec{a}$) para la ecuación de movimiento de un oscilador armónico simple es:

$$m\,\frac{d^2x}{dt^2} = -k\,x\,,\ o\,,\ \text{pasando todo al mismo miembro}:\quad m\,\frac{d^2x}{dt^2} + k\,x = 0\,.$$

donde x representa el desplazamiento respecto a la posición de equilibrio, m la masa inercial y $-kx$ la fuerza restauradora. Cuando además actúa una fuerza amortiguadora (viscosa) proporcional a la velocidad $F_v = -\mathrm{cv} = -c\,\frac{\mathrm{dx}}{\mathrm{dt}}$, la ecuación de movimiento queda:

$$m\,\frac{d^2x}{dt^2} + c\,\frac{dx}{dt} + k\,x = 0$$

la cual, dividiendo por m, suele escribirse como:

$$\frac{d^2x}{dt^2} + 2\,\beta\,\frac{dx}{dt} + \omega_0^2\,x = 0$$

donde $2\beta=c/m$ es el factor de amortiguamiento y $\omega_0 = \sqrt{k/m}$ es la frecuencia de vibración libre. Demos un nombre a esta ecuación:

```
In[1]:= Clear[x, β, ω0]
        ecamort = x''[t] + 2 β x'[t] + ω0^2 x[t]
Out[2]= ω0^2 x[t] + 2 β x'[t] + x''[t]
```

La solución general de tal ecuación puede obtenerse como:

```
In[3]:= DSolve[ecamort == 0, x[t], t]
Out[3]= {{x[t] → e^(t (-β-Sqrt[β^2-ω0^2])) C[1] + e^(t (-β+Sqrt[β^2-ω0^2])) C[2]}}
```

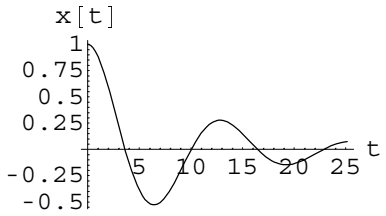
Dicha solución tiene comportamientos diferentes dependiendo de los valores de β y de ω_0. Estudiemoslos caso por caso:

Oscilador subamortiguado $\beta < \omega_0$

Por ejemplo, tomemos β=0.1, ω_0 = 0.5 y, como condición inicial: *x(0)=1, x'(0)=0.* La solución *x(t)* y su gráfica vienen dadas por:

```
In[4]:= Clear[x]
        subamort = DSolve[
          {ecamort == 0 /. {β -> 0.1, ω0 -> 0.5}, x[0] == 1, x'[0] == 0}, x, t]
        Plot[x[t] /. subamort, {t, 0, 2 (2 Pi / 0.5)},
         AxesLabel -> {"t", "x[t]"}]
```

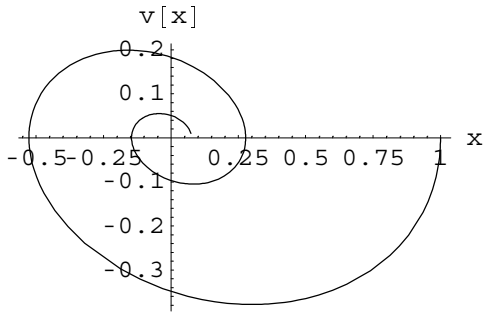
```
Out[5]= {{x → (ℯ^(-0.1 #1) (Cos[0.489898 #1] + 0.204124 Sin[0.489898 #1]) &)}}
```




```
Out[6]= - Graphics -
```

Vemos que se trata de un movimiento vibratorio donde la amplitud disminuye paulatinamente con el tiempo hasta hacerse cero (en el límite t->∞). Veamos una representación gráfica del movimiento en el plano de fases:

```
In[7]:= ParametricPlot[Evaluate[{x[t], x'[t]} /. subamort],
         {t, 0, 2 (2 Pi / 0.5)}, AxesLabel -> {"x", "v[x]"}]
```




```
Out[7]= - Graphics -
```

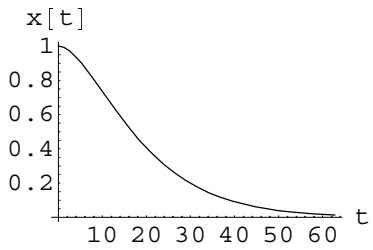
Las órbitas son abiertas, lo cual indica que el movimiento no es periódico. También vemos que la trayectoria tiende al punto (x(∞)=0, v(∞)=0).

Amortiguamiento crítico $\beta = \omega_0$

Por ejemplo, tomemos $\beta = \omega_0 = 0.1$ y, como condición inicial: *x(0)=1, x'(0)=0.* La solución *x(t)* y su gráfica vienen dadas por:

```
In[8]:= Clear[x]
        amortcrit = DSolve[
          {ecamort == 0 /. {β -> 0.1, ω0 -> 0.1}, x[0] == 1, x'[0] == 0}, x, t]
        Plot[x[t] /. amortcrit, {t, 0, (2 Pi / 0.1)},
         AxesLabel -> {"t", "x[t]"}]
```

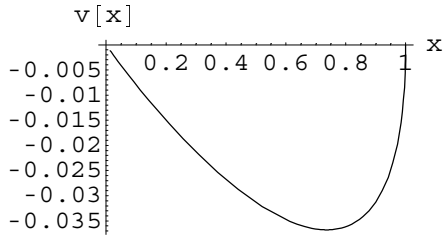
```
Out[9]= {{x → (ℯ^(-0.1 #1) (1. + 0.1 #1) &)}}
```




```
Out[10]= - Graphics -
```

Aquí el sistema no realiza ninguna oscilación y la posición va a cero paulatinamente con el tiempo (*x(∞)=0*). En el plano de fases:

```
In[11]:= ParametricPlot[Evaluate[{x[t], x'[t]} /. amortcrit],
          {t, 0, (2 Pi / 0.1)}, AxesLabel -> {"x", "v[x]"}]
```




```
Out[11]= - Graphics -
```

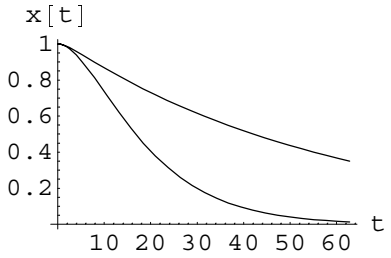
Vemos igualmente que la trayectoria tiende al punto (x(∞)=0, v(∞)=0) partiendo de (x(0)=1, v(0)=0)

Oscilador sobreamortiguado $\beta > \omega_0$

Por ejemplo, tomemos β=0.3, ω_0 = 0.1 y, como condición inicial: *x(0)=1, x'(0)=0.* La solución *x(t),* y su gráfica superpuesta a la de amortiguamiento crítico vienen dadas por:

```
In[12]:= Clear[x]
         sobreamort = DSolve[
           {ecamort == 0 /. {β -> 0.3, ω0 -> 0.1}, x[0] == 1, x'[0] == 0}, x, t]
         Plot[{x[t] /. sobreamort, x[t] /. amortcrit}, {t, 0, (2 Pi / 0.1)},
          AxesLabel -> {"t", "x[t]"}, AxesOrigin -> {0, 0}]
```

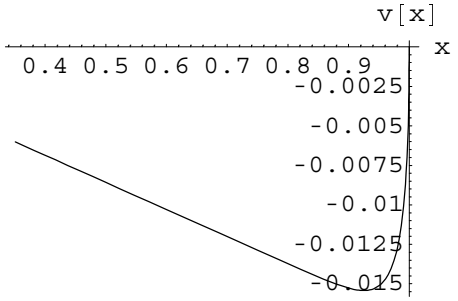
```
Out[13]= {{x → (-0.0303301 e^(-0.582843 #1) + 1.03033 e^(-0.0171573 #1) &)}}
```




```
Out[14]= - Graphics -
```

La presencia de una mayor viscosidad hace que la posición vaya hacia cero más lentamente que en el caso crítico. La representación en el plano de fases es similar al caso crítico:

```
In[15]:= ParametricPlot[Evaluate[{x[t], x'[t]} /. sobreamort],
          {t, 0, (2 Pi / 0.1)}, AxesLabel -> {"x", "v[x]"}]
```




```
Out[15]= - Graphics -
```

Ejercicio

Repita los casos anteriores para las condiciones iniciales *x(0)=0, x'(0)=1* e interprete físicamente los resultados y las gráficas obtenidas.

■ Oscilador armónico forzado: pulsaciones y resonancia

Solución general

Si añadimos una nueva fuerza externa dependiente del tiempo *F(t)* en la ecuación de movimiento de un oscilador armónico amortiguado obtenemos:

$$m\,\frac{d^2x}{dt^2} + c\,\frac{dx}{dt} + k\,x = \mathrm{F(t)}$$

la cual, dividiendo por *m*, suele escribirse como :

$$\frac{d^2x}{dt^2} + 2\beta\,\frac{dx}{dt} + \omega_0^2\,x = \mathrm{f(t)}$$

Las fuerzas externas que suelen discutirse en este tipo de problemas son de tipo periódico *f(t+T)=f(t)* como, por ejemplo:

$f(t)=A\ sin(\omega t)$,

donde A representa la amplitud de la acción externa $f(t)$ y ω representa su frecuencia . Un ejemplo de acción externa sobre un oscilador que todos hemos experimentado de pequeños puede ser "el empuje continuado y periódico sobre un columpio". Tomemos como ejemplo la siguiente ecuación:

```
In[16]:= Clear[x, β, ω0, ω, a]
         ecamforz = x''[t] + 2 β x'[t] + ω0^2 x[t] + a Sin[ω t]
Out[17]= a Sin[t ω] + ω0^2 x[t] + 2 β x'[t] + x''[t]
```

La solución general de tal ecuación puede obtenerse como:

```
In[18]:= DSolve[ecamforz == 0, x[t], t]
```

$$\text{Out[18]= } \left\{\left\{x[t] \to e^{t\left(-\beta-\sqrt{\beta^2-\omega0^2}\right)}\, C[1] + e^{t\left(-\beta+\sqrt{\beta^2-\omega0^2}\right)}\, C[2] - \frac{1}{2\sqrt{\beta^2-\omega0^2}}\right.$$
$$\left(a\, e^{t\left(-\beta+\sqrt{\beta^2-\omega0^2}\right)}\left(-\frac{e^{-t\left(-\beta+\sqrt{\beta^2-\omega0^2}\right)}\,\omega\,\text{Cos}[t\,\omega]}{\left(-\beta-i\,\omega+\sqrt{\beta^2-\omega0^2}\right)\left(-\beta+i\,\omega+\sqrt{\beta^2-\omega0^2}\right)} - \right.\right.$$
$$\left.\left.\frac{e^{-t\left(-\beta+\sqrt{\beta^2-\omega0^2}\right)}\left(-\beta+\sqrt{\beta^2-\omega0^2}\right)\text{Sin}[t\,\omega]}{\left(-\beta-i\,\omega+\sqrt{\beta^2-\omega0^2}\right)\left(-\beta+i\,\omega+\sqrt{\beta^2-\omega0^2}\right)}\right)\right) + \frac{1}{2\sqrt{\beta^2-\omega0^2}}$$
$$\left(a\, e^{t\left(-\beta-\sqrt{\beta^2-\omega0^2}\right)}\left(-\frac{e^{t\left(\beta+\sqrt{\beta^2-\omega0^2}\right)}\,\omega\,\text{Cos}[t\,\omega]}{\left(\beta-i\,\omega+\sqrt{\beta^2-\omega0^2}\right)\left(\beta+i\,\omega+\sqrt{\beta^2-\omega0^2}\right)} + \right.\right.$$
$$\left.\left.\left.\frac{e^{t\left(\beta+\sqrt{\beta^2-\omega0^2}\right)}\left(\beta+\sqrt{\beta^2-\omega0^2}\right)\text{Sin}[t\,\omega]}{\left(\beta-i\,\omega+\sqrt{\beta^2-\omega0^2}\right)\left(\beta+i\,\omega+\sqrt{\beta^2-\omega0^2}\right)}\right)\right)\right\}\right\}$$

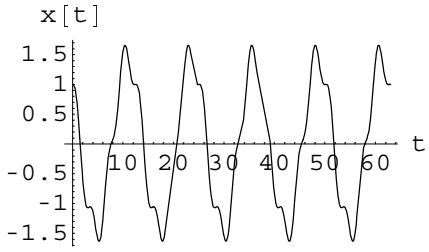
Dicha solución tiene comportamientos diferentes dependiendo de los valores de β , ω_0 y ω. Estudiemos algunos casos:

Oscilaciones forzadas no amortiguadas (β=0): pulsaciones y resonancia pura

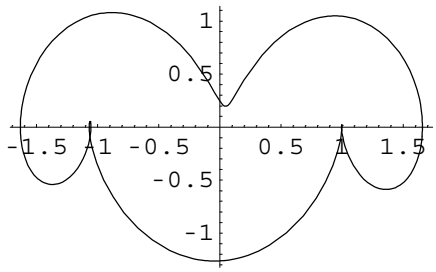
Tomemos como caso particular $\omega_0 = 0.5$, $\omega = 2$, a = 1 y como condición inicial $x(0) = 1$, $x'(0) = 0$ y hagamos una representación gráfica de la trayectoria y de la órbita en el plano de fases

In[19]:=
```
Clear[x]
forz = DSolve[{ecamforz == 0 /. {β -> 0, ω0 -> 0.5, ω -> 2, a -> 1},
   x[0] == 1, x'[0] == 0}, x, t]
Plot[x[t] /. forz, {t, 0, 20 Pi}, AxesLabel -> {"t", "x[t]"}]
ParametricPlot[Evaluate[{x[t], x'[t]} /. forz], {t, 0, 4 Pi}]
```

Out[20]=
```
{{x → (0.0666667 (15. Cos[0.5 #1] - 16. Sin[0.5 #1] + 4. Sin[2. #1]) &)}}
```



Out[21]= - Graphics -



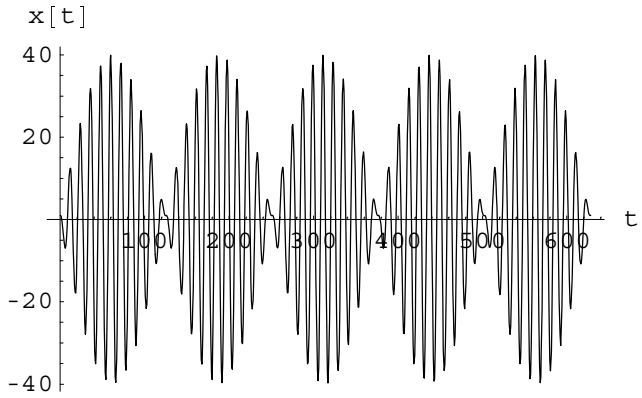
Out[22]= - Graphics -

Vemos que la órbita es cerrada (indicativo de comportamiento periódico). No obstante, nótese la diferencia respecto al caso libre (no forzado), donde las órbitas son simplemente elipses. Aquí las órbitas presentan un comportamiento más rico, dependiendo de los valores de ω y ω_0.

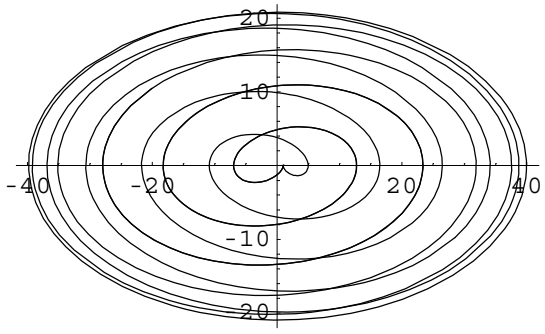
Para valores $\omega \approx \omega_0$ de la frecuencia externa ω muy cercanos a la frecuencia natural del sistema ω_0, se da un comportamiento interesante en el sistema conocido como el fenómeno de los *latidos o pulsaciones.* Por ejemplo, tomemos $\omega_0 = 0.5$, ω=0.55:

In[23]:=
```
forzpuls = DSolve[{ecamforz == 0 /. {β -> 0, ω0 -> 0.5, ω -> 0.55, a -> 1},
    x[0] == 1, x'[0] == 0}, x, t]
Plot[x[t] /. forzpuls, {t, 0, 200 Pi}, AxesLabel -> {"t", "x[t]"}]
ParametricPlot[Evaluate[{x[t], x'[t]} /. forzpuls], {t, 0, 50 Pi}]
```

Out[23]=
```
{{x → (0.047619
      (21. Cos[0.5 #1] - 440. Sin[0.5 #1] + 400. Sin[0.55 #1]) &)}}
```



Out[24]= - Graphics -

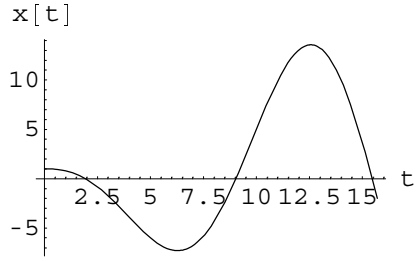


Out[25]= - Graphics -

Vemos como una oscilación de baja frecuencia (o periodo largo) $\omega_- = \omega - \omega_0 = 0.05$ "envuelve o modula" a una oscilación de alta frecuencia (o periodo corto) $\omega_+ = \omega + \omega_0 = 1.05$ (en comparación con la frecuencias originales). ¡Éste es el fundamento básico de la *frecuencia modulada* (FM), en la banda de frecuencias donde se suelen emitir las emisoras de radio musicales!.

El comportamiento del sistema cambia radicalmente (deja de ser periódico) para la frecuencia externa "crítica" $\omega = \omega_o$, es decir, cuando la frecuencia de la acción externa ω coincide con la frecuencia natural del sistema masa-resorte ω_0. En efecto, tomemos $\omega = \omega_0 = 0.5$

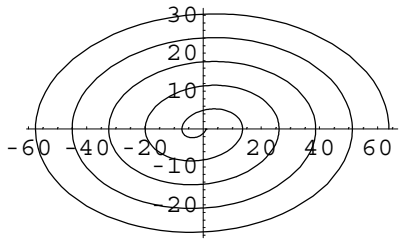
```
In[26]:= Clear[x]
         forzcrit = DSolve[{ecamforz == 0 /. {β -> 0, ω0 -> 0.5, ω -> 0.5, a -> 1},
            x[0] == 1, x'[0] == 0}, x, t]
         Plot[x[t] /. forzcrit, {t, 0, 5 Pi}, AxesLabel -> {"t", "x[t]"}]

Out[27]= {{x → (Cos[0.5 #1] - 2. Sin[0.5 #1] + Cos[0.5 #1] #1 &)}}
```




```
Out[28]= - Graphics -
```

Vemos que la amplitud de las oscilaciones crece indefinidamente con el tiempo. En el plano de fases tenemos:

```
In[29]:= ParametricPlot[Evaluate[{x[t], x'[t]} /. forzcrit], {t, 0, 20 Pi}]
```




```
Out[29]= - Graphics -
```

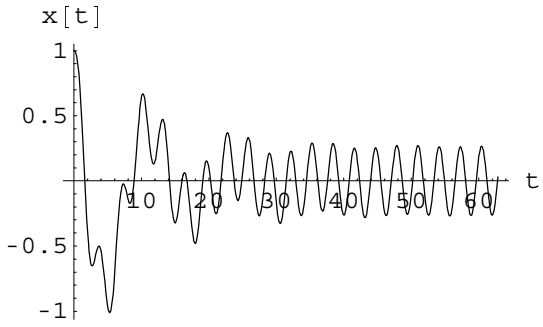
una órbita en espiral (hacia afuera). Este fenómeno se denomina: *resonancia pura.* Lo que suele ocurrir en los sistemas físicos reales en condiciones de resonancia pura es que el crecimiento indefinido de la amplitud de las oscilaciones provoca una "rotura" del sistema al sobrepasar el límite elástico. En obras de ingeniería civil, como los puentes, es conveniente asegurarse de que la frecuencia natural de oscilación del puente sea suficientemente distinta de la frecuencia de las acciones externas (ritmo de vehículos, peatones, viento local, etc), lo cual se logra usando unos materiales y una geometría adecuada.

Oscilaciones forzadas amortiguadas ($\beta \neq 0$): resonancia

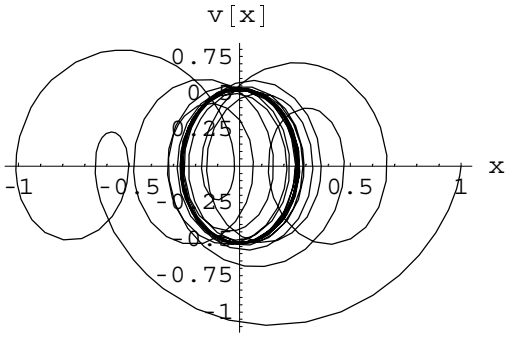
Tomemos como caso particular $\beta = 0.1$, $\omega_0 = 0.5$, $\omega = 2$, a = 1 y como condición inicial $x(0) = 1$, $x'(0) = 0$ y hagamos una representación gráfica de la trayectoria y de la órbita en el plano de fases

In[30]:=
```
Clear[x]
forzamort =
 DSolve[{ecamforz == 0 /. {β -> 0.1, ω0 -> 0.5, ω -> 2, a -> 1},
   x[0] == 1, x'[0] == 0}, x, t]
Plot[x[t] /. forzamort, {t, 0, 20 Pi}, AxesLabel -> {"t", "x[t]"}]
ParametricPlot[Evaluate[{x[t], x'[t]} /. forzamort],
 {t, 0, 40 Pi}, AxesLabel -> {"x", "v[x]"}]
```

Out[31]=
$$\{\{x \to ((0.0000292963 + 5.99706 \times 10^{-23}\, i)\, e^{-0.1\,\#1}\ (33174.\ \text{Cos}[0.489898\,\#1] + 960.\ e^{0.1\,\#1}\ \text{Cos}[2.\,\#1] - 29970.7\ \text{Sin}[0.489898\,\#1] + 9000.\ e^{0.1\,\#1}\ \text{Sin}[2.\,\#1])\ \&)\}\}$$



Out[32]= - Graphics -

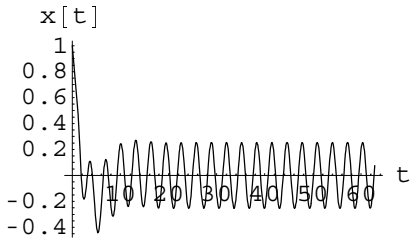


Out[33]= - Graphics -

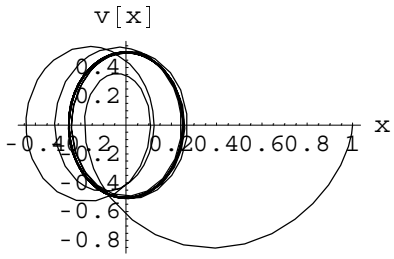
Vemos que la órbita no es cerrada pero que se aproxima asintóticamente a una órbita elíptica. En el gráfico observamos un "régimen o estado transitorio", que dura unos 20 o 30 segundos, tras el cual la oscilación se estabiliza, llegando a un "régimen o estado estacionario", que se corresponde con la solución particular de la ecuación no homogénea. Para ω y ω_0 fijas, la duración del transitorio depende de la amortiguación β. Por ejemplo, si tomamos una amortiguación mayor como, por ejemplo β=0.3 , obtenemos que:

In[34]:=
```
Clear[x]
forzamort =
 DSolve[{ecamforz == 0 /. {β -> 0.3, ω0 -> 0.5, ω -> 2, a -> 1},
   x[0] == 1, x'[0] == 0}, x, t]
forzamortplot = Plot[x[t] /. forzamort, {t, 0, 20 Pi},
  AxesLabel -> {"t", "x[t]"}]
forzamortfase = ParametricPlot[Evaluate[{x[t], x'[t]} /. forzamort],
  {t, 0, 40 Pi}, AxesLabel -> {"x", "v[x]"}]
```

Out[35]=
```
{{x → (0.000483793 E^(-0.3 #1) (1907. Cos[0.4 #1] + 160. E^(0.3 #1) Cos[2. #1] -
          1069.75 Sin[0.4 #1] + 500. E^(0.3 #1) Sin[2. #1]) &)}}
```



Out[36]= - Graphics -



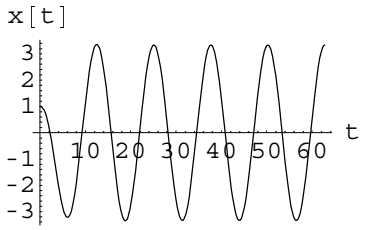
Out[37]= - Graphics -

se alcanza antes el régimen estacionario (a los 10 segundos aproximadamente...). Esto está de acuerdo con nuestra experiencia.

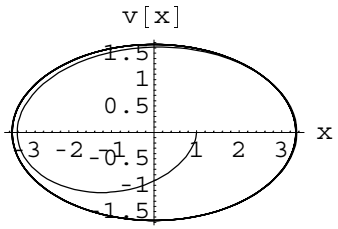
Para $\omega = \omega_0$ tenemos una *resonancia* (no "pura"), caracterizada porque la amplitud de las oscilaciones alcanza un máximo, sin crecer indefinidamente con el tiempo (como sucede en la resonancia pura), debido a la presencia de amortiguamiento. Tomemos el mismo amortiguamiento que en el caso anterior β=0.3, y $\omega_0 = 0.5 = \omega$. Para estos valores tenemos:

In[38]:=
```
Clear[x]
forzamortcrit =
 DSolve[{ecamforz == 0 /. {β -> 0.3, ω0 -> 0.5, ω -> 0.5, a -> 1},
    x[0] == 1, x'[0] == 0}, x, t]
forzamortplotcrit = Plot[x[t] /. forzamortcrit,
  {t, 0, 20 Pi}, AxesLabel -> {"t", "x[t]"}]
forzamortfasecrit = ParametricPlot[
  Evaluate[{x[t], x'[t]} /. forzamortcrit],
  {t, 0, 40 Pi}, AxesLabel -> {"x", "v[x]"}]
```

Out[39]=
```
{{x → (0.333333 e^(-0.3 #1)
      (-7. Cos[0.4 #1] + 10. e^(0.3 #1) Cos[0.5 #1] - 5.25 Sin[0.4 #1]) &)}}
```



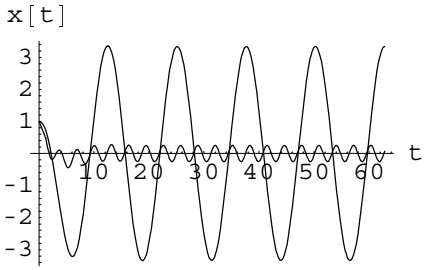
Out[40]= - Graphics -



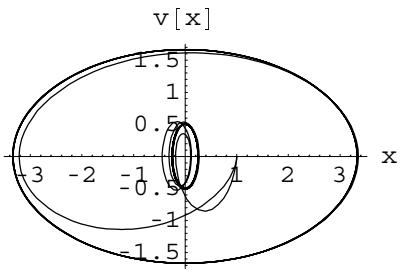
Out[41]= - Graphics -

Superpongamos las gráficas de la solución no crítica **{β->0.3,ω0->0.5,ω->2,a->1}** con la crítica **{β->0.3,ω0->0.5,ω->0.5,a->1}**

In[42]:=
```
Show[{forzamortplot, forzamortplotcrit}, PlotRange -> {-3.5, 3.5}]
Show[forzamortfase, forzamortfasecrit]
```



Out[42]= - Graphics -



Out[43]= - Graphics -

Vemos que la amplitud de las oscilaciones es, con diferencia, mayor en el caso crítico. También lo es la velocidad, tal y como muestra el diagrama de fases. Como se ha comentado anteriormente, el fenómeno de la resonancia tiene una doble vertiente. En su vertiente "positiva", este fenómeno es explotado, por ejemplo, para SINTONIZAR ciertas frecuencias próximas a la frecuencia natural propia de ciertos cristales o ciertos circuitos (tal y como sucede en nuestro aparato de radio). En su vertiente "negativa", el fenómeno de la resonancia puede provocar la ROTURA de ciertas estructuras sometidas a una acción externa periódica de frecuencia próxima a la frecuencia natural de la estructura, debido a las fuertes oscilaciones provocadas en estas condiciones críticas.

Ejercicio

Repita el caso anterior para un amortiguamiento fuerte β=1. ¿Cómo afecta el amortiguamiento a la resonancia?, es decir, ¿se aprecia mejor la resonancia para el oscilador sobreamortiguado que para el subamortiguado o al revés?.

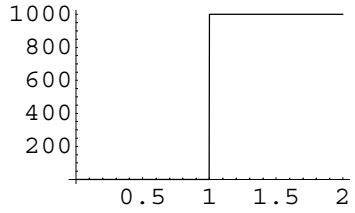
Fuerzas externas definidas a trozos

Estudiemos el oscilador armónico amortiguado y forzado:

$$\frac{d^2 x}{d t^2} + 2\beta \frac{d x}{d t} + \omega_0^2 x = \mathrm{f(t)}$$

para funciones más generales que las de tipo sinusoidal. Por ejemplo, centrémonos en funciones definidas a trozos como, por ejemplo, la función salto

In[44]:=
```
salto[t_] := If[t >= 1, 1000, 0]
Plot[salto[t], {t, 0, 2}]
```



Out[45]=
```
- Graphics -
```

que representa una fuerza que no actúa hasta el instante t=1, en el cual adquiere un valor constante =1000. Este tipo de fuerzas aparecen, por ejemplo, al accionar repentinamente un "interruptor"... Veamos cuál es el efecto de dicha fuerza sobre un oscilador

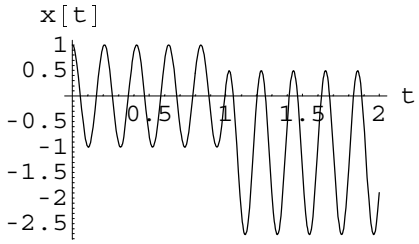
In[46]:=
```
Clear[x, β, ω0]
ecamsalto = x''[t] + 2 β x'[t] + ω0^2 x[t] + salto[t]
```

Out[47]=
```
If[t ≥ 1, 1000, 0] + ω0^2 x[t] + 2 β x'[t] + x''[t]
```

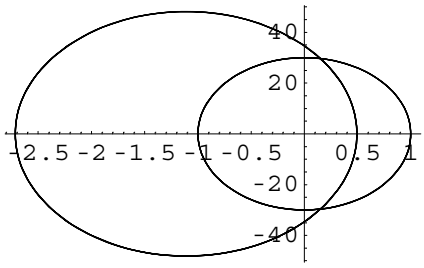
Para este tipo de funciones, *Mathematica* necesita de recursos numéricos para su resolución. Por ejemplo, tomemos amortiguamiento β=0, frecuencia natural ω_0 = 30 y condición inicial x[0]==1,x'[0]==0 y resolvamos numéricamente en el intervalo $0 \le t \le 2$

In[48]:=
```
Clear[x]
chupinazo = NDSolve[{ecamsalto == 0 /. {β -> 0, ω0 -> 30},
   x[0] == 1, x'[0] == 0}, x, {t, 0, 2}]
Plot[x[t] /. chupinazo, {t, 0, 2}, AxesLabel -> {"t", "x[t]"}]
ParametricPlot[Evaluate[{x[t], x'[t]} /. chupinazo], {t, 0, 2}]
```

Out[49]=
```
{{x → InterpolatingFunction[{{0., 2.}}, <>]}}
```



Out[50]= - Graphics -



Out[51]= - Graphics -

Vemos como, a partir del instante t=1, el comportamiento de la solución cambia de forma contínua pero repentina. Por ejemplo, la órbita en el espacio de las fases (elipses) se desplaza hacia la izquierda y aumenta de tamaño; esto quiere decir que, tras la acción de la fuerza en t=1, el oscilador pasa más tiempo en los valores x<0 que en x>0, además de aumentar su velocidad.

Ejercicio

Estudiar el comportamiento del oscilador anterior en el intervalo $0 \le t \le 3$ para un pulso

$$p(t)=\begin{cases} 10000, & \text{si } 1 \le t \le 2 \\ 0, & \text{en otro caso} \end{cases}$$

Interprete los resultados.

Ayuda: recuerde el comando **Which[condicion1,proceso1,condición2,proceso2,...]** para definir funciones a trozos.

Ejercicio

Estudie la solución del oscilador armónico simple de frecuencia 1 sometido a una función periódica del tipo "diente de sierra":

f[t]=t-Round[t]

en el intervalo $0 \le t \le 20$

■ Oscilador anarmónico

Discusión general

Estudiaremos en esta sección el caso más complicado (no lineal) de una fuerza recuperadora no lineal del tipo $F(x)=-k\,x-\delta\,x^3$. Para ganar intuición sobre el tipo de comportamiento que esperamos, hagamos una representación gráfica de dicha fuerza para tres casos distintos y valores particulares de los parámetros:

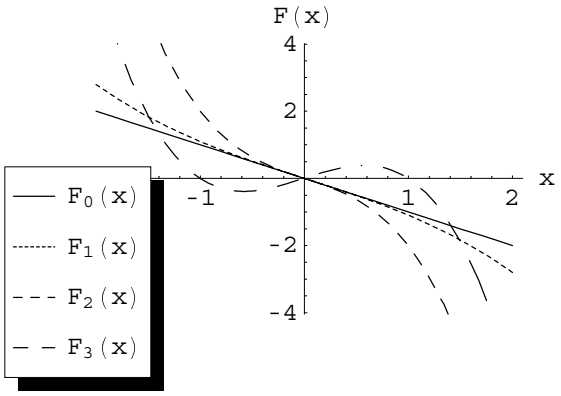
1) Armónico (δ=0): $F_0(x) = -x$

2) Anarmonicidad débil (δ<<k): $F_1(x) = -x - \frac{1}{10}x^3$

3) Anarmonicidad fuerte ($\delta \approx$ k): $F_2(x) = -x - x^3$

4) Anarmonicidad atractivo-repulsiva (k<0,δ>0): $F_3(x) = x - x^3$

In[52]:=
```
<< Graphics`Legend`
Plot[{-x, -x - 0.1 x^3, -x - x^3, x - x^3}, {x, -2, 2}, PlotStyle →
  {GrayLevel[0], Dashing[{.01}], Dashing[{.03}], Dashing[{.05}]},
 PlotLegend -> {"F0(x)", "F1(x)", "F2(x)", "F3(x)"},
 AxesLabel -> {"x", "F(x)"}]
```



Out[53]= - Graphics -

Mientras que en los tres primeros casos el único punto de equilibrio (F=0) es x=0, para el cuarto caso tenemos tres puntos de equilibrio: x=0,1,-1.

Estudiemos las soluciones de la ecuación de un oscilador anarmónico amortiguado sometido a una fuerza externa dependiente del tiempo $F_e(t)$. Dicha ecuación admite una expresión general del tipo:

$$m\,\frac{d^2x}{dt^2} + c\,\frac{dx}{dt} + k\,x + \delta\,x^3 = F_e(t)$$

la cual, dividiendo por *m*, suele escribirse como :

$$\frac{d^2x}{dt^2} + 2\beta\,\frac{dx}{dt} + \omega_0^2\,x + \gamma\,x^3 = f_e(t)$$

Tomemos por ejemplo $f_e(t)=a\,sen(\omega t)$ y demos un nombre a esta ecuación

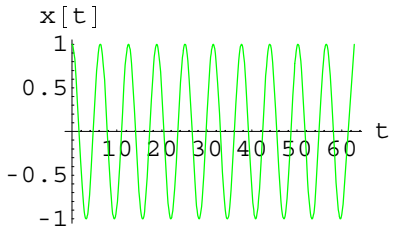
```
In[54]:= Clear[x, β, ω0, ω, γ, a]
         ecanamfor = x''[t] + 2 β x'[t] + ω0^2 x[t] + γ x[t]^3 + a Sin[ω t]
Out[55]= a Sin[t ω] + ω0^2 x[t] + γ x[t]^3 + 2 β x'[t] + x''[t]
```

Vemos que se trata de una ecuación no lineal de segundo orden. Necesitaremos técnicas numéricas para su resolución. Tenemos una casuística bastante rica dependiendo de los parámetros β, ω_0 ,ω ,δ ,a. Tratemos algunos casos:

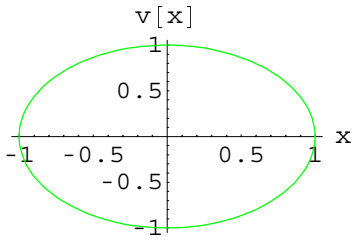
Oscilador anarmónico libre (β=0=a)

Comencemos por perturbar débilmente ($\gamma << \omega_0^2$) el oscilador armónico. Repesentemos primeramente la solución del caso armónico (γ=0) para ω_0 = 1 y condiciones iniciales *x(0)==1,x'(0)==0* :

```
In[56]:= armonico = DSolve[{x''[t] == - x[t], x[0] == 1, x'[0] == 0}, x, t]
         xtar = Plot[x[t] /. armonico, {t, 0, 20 Pi},
           AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[0, 1, 0]}]
         fasear = ParametricPlot[Evaluate[{x[t], x'[t]} /. armonico],
           {t, 0, 2 Pi}, AxesLabel -> {"x", "v[x]"},
           PlotStyle -> {RGBColor[0, 1, 0]}]
Out[56]= {{x → (Cos[#1] &)}}
```




```
Out[57]= - Graphics -
```

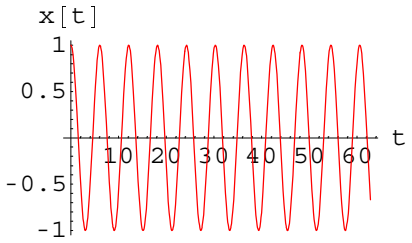



```
Out[58]= - Graphics -
```

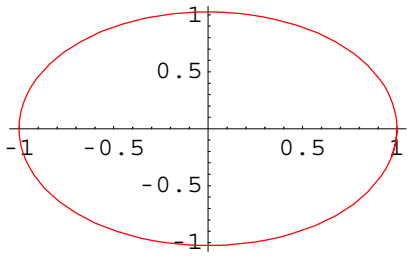
y perturbemos débilmente con γ=0.1

In[59]:=
```
Clear[x]
anarmdebil =
 NDSolve[{ecanamfor == 0 /. {β -> 0, ω0 -> 1, ω -> 0, a -> 0, γ -> 0.1},
   x[0] == 1, x'[0] == 0}, x, {t, 0, 20 Pi}]
xtanardeb = Plot[x[t] /. anarmdebil, {t, 0, 20 Pi},
  AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[1, 0, 0]}]
faseanardeb = ParametricPlot[
  Evaluate[{x[t], x'[t]} /. anarmdebil[[1]]],
  {t, 0, 2 Pi}, PlotStyle -> {RGBColor[1, 0, 0]}]
```

Out[60]=
```
{{x → InterpolatingFunction[{{0., 62.8319}}, <>]}}
```



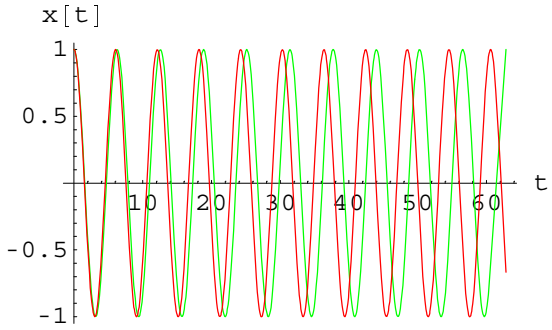
Out[61]=
```
- Graphics -
```



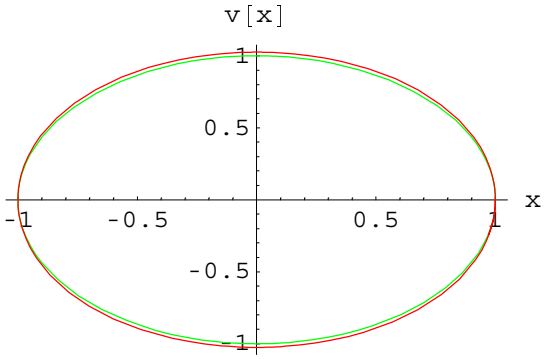
Out[62]=
```
- Graphics -
```

Superpongamos ambos gráficos:

In[63]:=
```
Show[xtar, xtanardeb]
Show[fasear, faseanardeb]
```



Out[63]= - Graphics -



Out[64]= - Graphics -

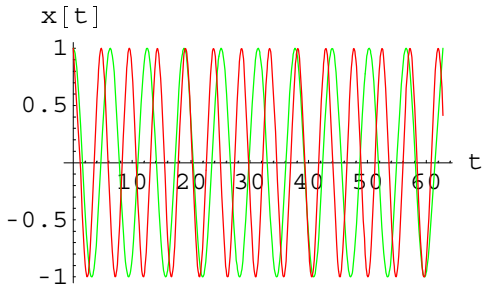
Vemos que el efecto de la perturbación anarmónica γ=0.1 es generar un pequeño adelanto de la oscilación anarmónica (roja) respecto de la armónica (verde) debido a un pequeño incremento de la velocidad en las oscilaciones anarmónicas. Veamos qué pasa para una anarmonicidad fuerte como γ=1

In[65]:=
```
Clear[x]
anarmfuerte =
 NDSolve[{ecanamfor == 0 /. {β -> 0, ω0 -> 1, ω -> 0, a -> 0, γ -> 1},
   x[0] == 1, x'[0] == 0}, x, {t, 0, 20 Pi}]
xtanarfu = Plot[x[t] /. anarmfuerte, {t, 0, 20 Pi},
  AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[1, 0, 0]},
  DisplayFunction -> Identity]
faseanarfu = ParametricPlot[
  Evaluate[{x[t], x'[t]} /. anarmfuerte[[1]]], {t, 0, 2 Pi},
  PlotStyle -> {RGBColor[1, 0, 0]}, DisplayFunction -> Identity]
Show[xtar, xtanarfu]
Show[fasear, faseanarfu]
```

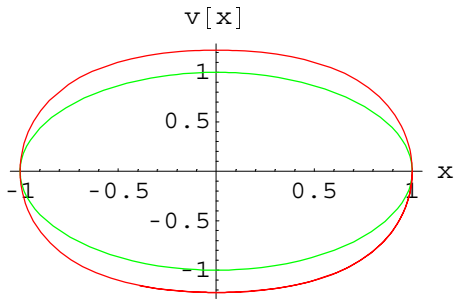
Out[66]= `{{x → InterpolatingFunction[{{0., 62.8319}}, <>]}}`

Out[67]= `- Graphics -`

Out[68]= `- Graphics -`



Out[69]= `- Graphics -`



Out[70]= `- Graphics -`

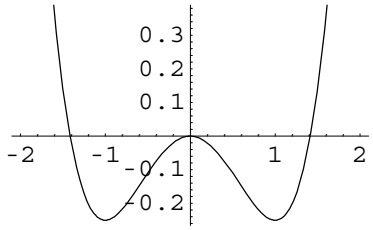
Aquí el adelanto de la oscilación anarmónica respecto a la armónica es mucho mayor que antes, debido a que la diferencia de velocidades es mayor (¿por qué?), tal y como se observa en el diagrama de fases.

Ejercicio

Repita los cálculos para una anarmonicidad muy fuerte γ=10 y condiciones iniciales *x(0)==0,x'(0)==1*. Compare con el caso armónico γ=0.

Por último, veamos qué sucede para una fuerza atractivo-repulsiva con $\omega_0^2 = -1 = -\gamma$, es decir $F_3(x) \propto x - x^3$, con tres puntos de equilibrio: x=0 (inestable), x=±1 (estable). El potencial $U(x) = -\int (x - x^3)\, dx = \frac{x^4}{4} - \frac{x^2}{2}$ tiene la forma:

In[71]:=
```
Plot[x^4 / 4 - x^2 / 2, {x, -2, 2}]
```



Out[71]= - Graphics -

de "sombrero mejicano" con dos "pozos" en x=±1 y una "cima" en x=0. Tengamos en cuenta tres posibles condiciones iniciales:

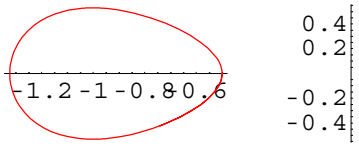
1) *x(0)=-0.5, x'(0)=0:* la partícula parte del reposo en los alrededores del "pozo" izquierdo.

2) *x(0)=0.5, x'(0)=0:* la partícula parte del reposo en los alrededores del "pozo" derecho

3) *x(0)=0, x'(0)=0.1:* la partícula parte de la "cima" con una pequeña velocidad inicial hacia la derecha

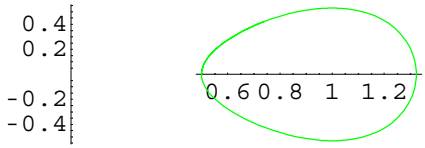
In[72]:=
```
Clear[x]
anarmrepatr1 =
 NDSolve[{ecanamfor == 0 /. {β -> 0, ω0 -> I, ω -> 0, a -> 0, γ -> 1},
   x[0] == -0.5, x'[0] == 0}, x, {t, 0, 2 Pi}]
fase1 = ParametricPlot[Evaluate[{x[t], x'[t]} /. anarmrepatr1[[1]]],
  {t, 0, 2 Pi}, PlotStyle -> {{RGBColor[1, 0, 0]}},
  AxesOrigin -> {0, 0} ]

Clear[x]
anarmrepatr2 =
 NDSolve[{ecanamfor == 0 /. {β -> 0, ω0 -> I, ω -> 0, a -> 0, γ -> 1},
   x[0] == 0.5, x'[0] == 0}, x, {t, 0, 2 Pi}]
fase2 = ParametricPlot[Evaluate[{x[t], x'[t]} /. anarmrepatr2[[1]]],
  {t, 0, 2 Pi}, PlotStyle -> {RGBColor[0, 1, 0]}, AxesOrigin -> {0, 0}]

Clear[x]
anarmrepatr3 =
 NDSolve[{ecanamfor == 0 /. {β -> 0, ω0 -> I, ω -> 0, a -> 0, γ -> 1},
   x[0] == 0, x'[0] == 0.1}, x, {t, 0, 6 Pi}]
fase3 = ParametricPlot[Evaluate[{x[t], x'[t]} /. anarmrepatr3[[1]]],
  {t, 0, 6 Pi}, PlotStyle -> {RGBColor[0, 0, 1]}, AxesOrigin -> {0, 0}]
```

Out[73]=
```
{{x → InterpolatingFunction[{{0., 6.28319}}, <>]}}
```



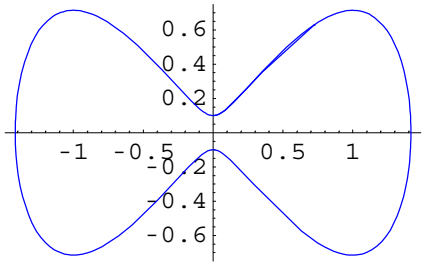
Out[74]=
```
- Graphics -
```

Out[76]=
```
{{x → InterpolatingFunction[{{0., 6.28319}}, <>]}}
```



Out[77]=
```
- Graphics -
```

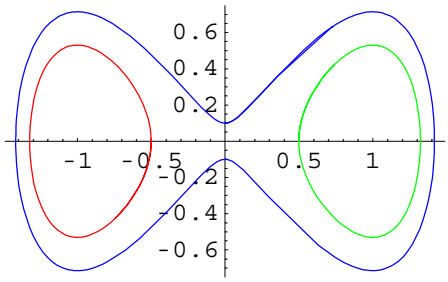
Out[79]=
```
{{x → InterpolatingFunction[{{0., 18.8496}}, <>]}}
```



Out[80]=
```
- Graphics -
```

Combinemos estas tres órbitas en un solo gráfico:

In[81]:= **Show[{fase1, fase2, fase3}]**



Out[81]= - Graphics -

Vemos que, en el caso 1), la partícula oscila alrededor del mínimo de potencial x=-1 (posición de equilibrio estable). Para el caso 2) la partícula oscila en los alrededores de x=1. En el tercer caso la partícula oscila entre pozo y pozo, pasando de uno a otro de forma periódica (observe como la velocidad disminuye en las cercanías de la "cima" x=0 ¿por qué?).

Ejercicio

Repita el ejercicio anterior para las siguientes condiciones iniciales:

1) *x(0)=-1, x'(0)=1:* la partícula parte de la posición de equilibrio x=-1 con velocidad x'=1.

2) *x(0)=0.5, x'(0)=0:* la partícula parte de la posición de equilibrio x=1 con velocidad x'=-1.

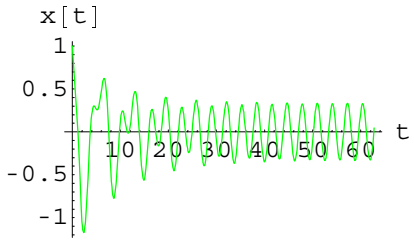
3) *x(0)=0, x'(0)=-0.2:* la partícula parte de la "cima" con una pequeña velocidad inicial hacia la izquierda

Ecuación de Duffing ($\omega_0 = \gamma = 1$)

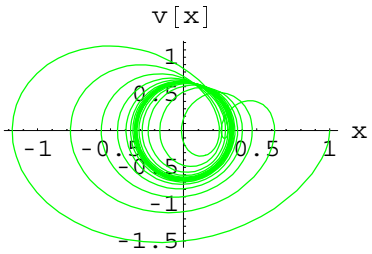
Veamos cuál es el efecto de añadir un término anarmónico x^3 al oscilador armónico amortiguado forzado estudiado anteriormente. Tomemos β=0.1, $\omega_0 = 1$, $\omega = 2$, $a = 1$ y repesentemos primeramente la solución del caso armónico (γ=0) con condiciones iniciales *x(0)==1,x'(0)==0* :

In[82]:=
```
Clear[x]
armoamorfor =
 DSolve[{ecanamfor == 0 /. {β -> 0.1, ω0 -> 1, ω -> 2, a -> 1, γ -> 0},
   x[0] == 1, x'[0] == 0}, x, t]
xtaramf = Plot[x[t] /. armoamorfor, {t, 0, 20 Pi},
  AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[0, 1, 0]}]
fasearamf = ParametricPlot[Evaluate[{x[t], x'[t]} /. armoamorfor],
  {t, 0, 20 Pi}, AxesLabel -> {"x", "v[x]"},
  PlotStyle -> {RGBColor[0, 1, 0]}]
```

Out[83]=
$$\{\{x \to ((0.0000661638 - 6.24521 \times 10^{-22}\, i)\, e^{-0.1\,\#1}$$
$$(14454.\,\text{Cos}[0.994987\,\#1] + 660.\, e^{0.1\,\#1}\,\text{Cos}[2.\,\#1] -$$
$$8497.19\,\text{Sin}[0.994987\,\#1] + 4950.\, e^{0.1\,\#1}\,\text{Sin}[2.\,\#1])\ \&)\}\}$$



Out[84]= - Graphics -

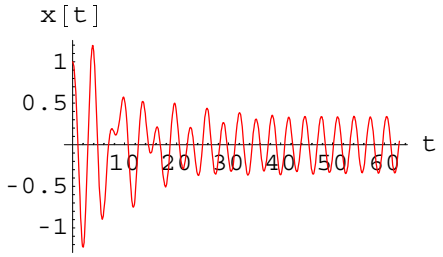


Out[85]= - Graphics -

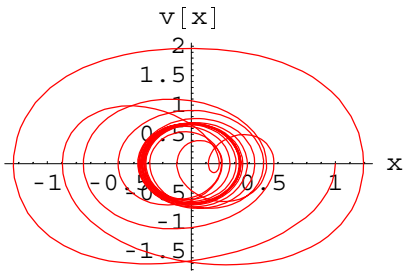
y perturbemos con γ=1

In[86]:=
```
Clear[x]
anarmoamorfor =
 NDSolve[{ecanamfor == 0 /. {β -> 0.1, ω0 -> 1, ω -> 2, a -> 1, γ -> 1},
   x[0] == 1, x'[0] == 0}, x, {t, 0, 20 Pi}]
xtanaramf = Plot[x[t] /. anarmoamorfor, {t, 0, 20 Pi},
  AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[1, 0, 0]}]
faseanaramf = ParametricPlot[
  Evaluate[{x[t], x'[t]} /. anarmoamorfor], {t, 0, 20 Pi},
  AxesLabel -> {"x", "v[x]"}, PlotStyle -> {RGBColor[1, 0, 0]}]
```

Out[87]=
```
{{x → InterpolatingFunction[{{0., 62.8319}}, <>]}}
```

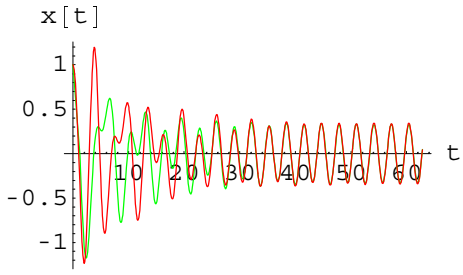


Out[88]=
```
- Graphics -
```

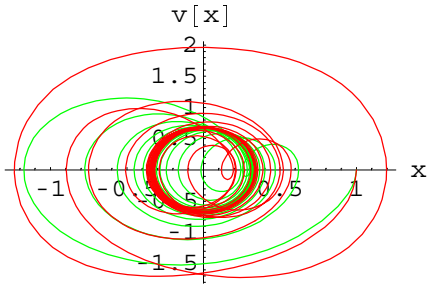


Out[89]=
```
- Graphics -
```

In[90]:=
```
Show[xtaramf, xtanaramf]
Show[fasearamf, faseanaramf]
```



Out[90]=
```
- Graphics -
```



Out[91]=
```
- Graphics -
```

Vemos que el efecto de la perturbación anarmónica γ=1 no es importante en el estado estacionario (aunque si en el transitorio).

Ejercicio

Estudie la ecuación de Duffing para β=0.2, $\omega_0 = 1 = \gamma$, $\omega = 2$, $a = 1$ y condiciones iniciales *x(0)==0,x'(0)==1*. Compare con el caso armónico γ=0.

■ El péndulo

Discusión general

Consideremos la ecuación general de movimiento de un péndulo amortiguado y forzado:

$$\ddot{\phi} + 2\beta\dot{\phi} + \omega_0{}^2 \operatorname{sen}(\phi) = f(t).$$

Vemos que se trata de una ecuación no lineal de segundo orden. Tomemos por ejemplo *f(t)=a sen(ωt)* e introduzcamos la ecuación:

```
In[92]:= Clear[x, β, ω0, ω, γ, a]
         ecpendamfor = x''[t] + 2 β x'[t] + ω0^2 Sin[x[t]] + a Sin[ω t]
Out[93]= a Sin[t ω] + ω0^2 Sin[x[t]] + 2 β x'[t] + x''[t]
```

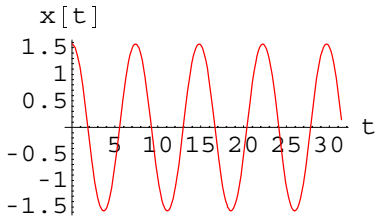
Consideremos diferentes casos:

Péndulo simple (β=0=a)

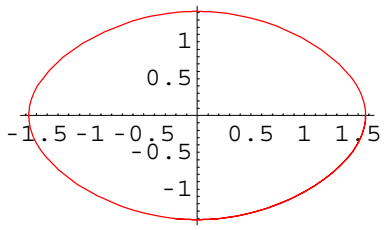
Soltemos el péndulo desde la horizontal, es decir, desde un ángulo inicial de 90 grados (x[0]==π/2) y en reposo (x'[0]==0)

In[94]:=
```
Clear[x]
pendulosimple =
 NDSolve[{ecpendamfor == 0 /. {β -> 0, ω0 -> 1, ω -> 0, a -> 0},
   x[0] == Pi / 2, x'[0] == 0}, x, {t, 0, 10 Pi}]
xtpendsimp = Plot[x[t] /. pendulosimple, {t, 0, 10 Pi},
  AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[1, 0, 0]}]
fasependsimp = ParametricPlot[
  Evaluate[{x[t], x'[t]} /. pendulosimple[[1]]],
  {t, 0, 3 Pi}, PlotStyle -> {RGBColor[1, 0, 0]}]
```

Out[95]=
```
{{x → InterpolatingFunction[{{0., 31.4159}}, <>]}}
```



Out[96]=
```
- Graphics -
```

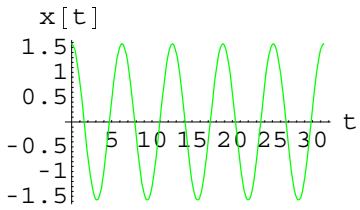


Out[97]=
```
- Graphics -
```

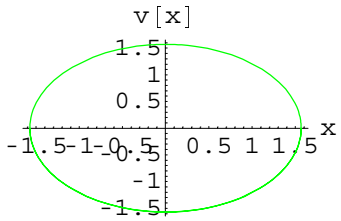
Comparemos con un oscilador armónico simple con idénticas condiciones iniciales

In[98]:=
```
Clear[armonico, xtar, fasear]
armonico = DSolve[{x''[t] == - x[t], x[0] == Pi / 2, x'[0] == 0}, x, t]
xtar = Plot[x[t] /. armonico, {t, 0, 10 Pi},
  AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[0, 1, 0]}]
fasear = ParametricPlot[Evaluate[{x[t], x'[t]} /. armonico],
  {t, 0, 3 Pi}, AxesLabel -> {"x", "v[x]"},
  PlotStyle -> {RGBColor[0, 1, 0]}]
```

Out[99]=
$$\left\{\left\{x \to \left(\frac{1}{2}\,\pi\,\text{Cos}[\#1]\ \&\right)\right\}\right\}$$



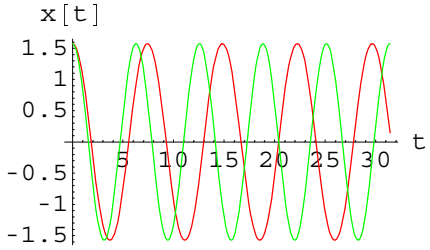
Out[100]= - Graphics -



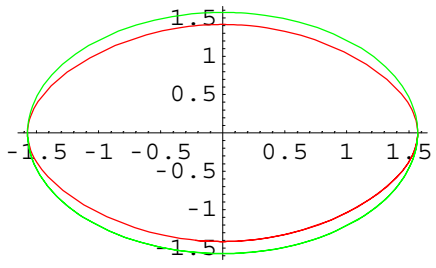
Out[101]= - Graphics -

Superpongamos las gráficas del péndulo y del oscilador armónico simples:

In[102]:=
```
Show[xtpendsimp, xtar]
Show[fasependsimp, fasear]
```



Out[102]= - Graphics -



Out[103]= - Graphics -

Vemos que, a idénticas condiciones iniciales, el periodo del péndulo simple es mayor que el del oscilador armónico simple. Sin embargo, su velocidad es menor, como muestra el diagrama de fases. Estas diferencias entre el oscilador armónico y el péndulo se hacen cada vez más pequeñas conforme la amplitud de las oscilaciones del péndulo se aproxima más a cero, ¿por qué?.

Ejercicio

Repetid los cálculos anteriores para las condiciones iniciales:

x[0]=π/4, x'[0]=0

¿Es cierto que las diferencias entre las soluciones del péndulo simple y del oscilador armónico simple son menores ahora (menor amplitud) que para el caso anterior x[0]=π/2, x'[0]=0 (mayor amplitud)?

Ejercicio

Para un oscilador armónico simple, se demuestra que el periodo T no depende de la amplitud A de las oscilaciones. Compruebe que esto es cierto superponiendo las gráficas de las soluciones del oscilador armónico simple $x''(t)+\omega_0^2\, x(t) = 0$ con condiciones iniciales a) x[0]=π/2, x'[0]=0 y b) x[0]=π/4, x'[0]=0, tomando ${\omega_0}^2 = 1$.

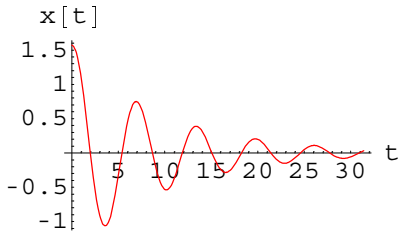
Repita el ejercicio para el péndulo simple. ¿Depende ahora el periodo de la amplitud?. ¿Aumenta el periodo con la amplitud o disminuye?

Péndulo amortiguado

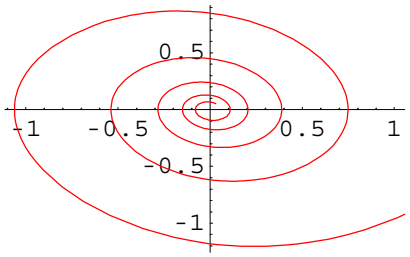
Tomemos como factor de amortiguamiento β=0.1 y como condición inicial: "péndulo horizontal en reposo"

In[104]:=
```
Clear[x]
penduloamort =
 NDSolve[{ecpendamfor == 0 /. {β -> 0.1, ω0 -> 1, ω -> 0, a -> 0},
   x[0] == Pi / 2, x'[0] == 0}, x, {t, 0, 10 Pi}]
xtpendamort = Plot[x[t] /. penduloamort, {t, 0, 10 Pi},
  AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[1, 0, 0]}]
fasependamort = ParametricPlot[
  Evaluate[{x[t], x'[t]} /. penduloamort[[1]]],
  {t, 0, 10 Pi}, PlotStyle -> {RGBColor[1, 0, 0]}]
```

Out[105]=
```
{{x → InterpolatingFunction[{{0., 31.4159}}, <>]}}
```



Out[106]=
```
- Graphics -
```



Out[107]=
```
- Graphics -
```

Vemos que la amplitud de las oscilaciones disminuye paulatinamente en el tiempo, al igual que la velocidad, tendiendo ambos asintóticamente a cero (posición de equilibrio)

Ejercicio

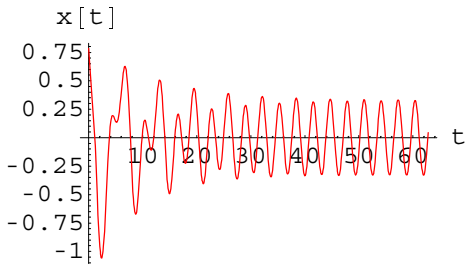
Compare el péndulo amortiguado anterior con el oscilador armónico amortiguado. ¿En qué caso es el amortiguamiento más rápido?

Péndulo amortiguado forzado

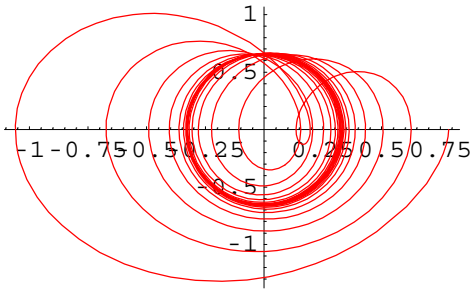
Tomemos como factor de amortiguamiento β=0.1, como frecuencia natural $\omega_0^2 = 1$, como frecuencia externa ω=2 y amplitud a=*1*, y como condición inicial: "péndulo a 45 grados de la vertical y en reposo"

In[108]:=
```
Clear[x]
penduloamfor =
 NDSolve[{ecpendamfor == 0 /. {β -> 0.1, ω0 -> 1, ω -> 2, a -> 1},
   x[0] == Pi / 4, x'[0] == 0}, x, {t, 0, 20 Pi}]
xtpendamfor = Plot[x[t] /. penduloamfor, {t, 0, 20 Pi},
  AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[1, 0, 0]}]
fasependamfor = ParametricPlot[
  Evaluate[{x[t], x'[t]} /. penduloamfor[[1]]],
  {t, 0, 20 Pi}, PlotStyle -> {RGBColor[1, 0, 0]}]
```

Out[109]=
```
{{x → InterpolatingFunction[{{0., 62.8319}}, <>]}}
```



Out[110]=
```
- Graphics -
```

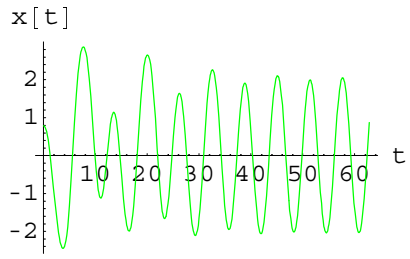


Out[111]=
```
- Graphics -
```

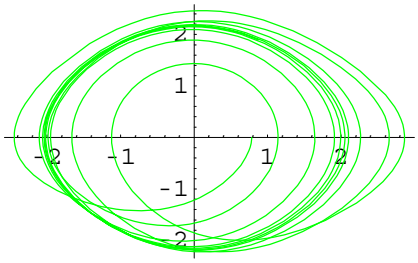
Al igual que para el oscilador armónico forzado, observamos la presencia de un regimen estacionario tras un periodo transitorio. Veamos cómo se comporta el péndulo para la resonancia $\omega_0{}^2=\omega^2 = 1$

In[112]:=
```
Clear[x]
penduloamforcrit =
 NDSolve[{ecpendamfor == 0 /. {β -> 0.1, ω0 -> 1, ω -> 1, a -> 1},
   x[0] == Pi / 4, x'[0] == 0}, x, {t, 0, 20 Pi}]
xtpendamforcrit = Plot[x[t] /. penduloamforcrit, {t, 0, 20 Pi},
  AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[0, 1, 0]}]
fasependamforcrit = ParametricPlot[
  Evaluate[{x[t], x'[t]} /. penduloamforcrit[[1]]],
  {t, 0, 20 Pi}, PlotStyle -> {RGBColor[0, 1, 0]}]
```

Out[113]=
```
{{x → InterpolatingFunction[{{0., 62.8319}}, <>]}}
```



Out[114]=
```
- Graphics -
```



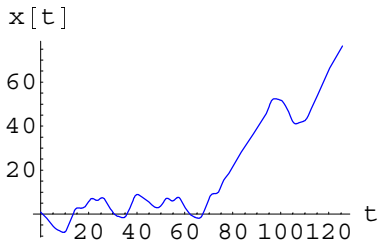
Out[115]=
```
- Graphics -
```

Observamos cómo la amplitud de las oscilaciones es mayor en el caso crítico $\omega_0=\omega = 1$ (con una amplitud $\approx$ 2) que en caso anterior $\omega_0=1$, $\omega = 2$ (con amplitud $\approx$0.25). Este es el fenómeno de la RESONANCIA.

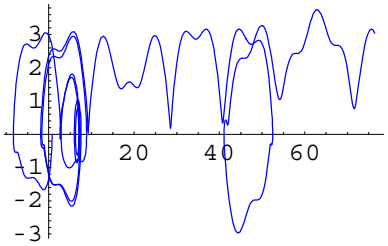
Veamos que sucede si eliminamos el amortiguamiento:

In[116]:=
```
Clear[x]
penduloforcrit =
 NDSolve[{ecpendamfor == 0 /. {β -> 0, ω0 -> 1, ω -> 1, a -> 1},
   x[0] == Pi / 4, x'[0] == 0}, x, {t, 0, 40 Pi}, MaxSteps -> 3000]
xtpendforcrit = Plot[x[t] /. penduloforcrit, {t, 0, 40 Pi},
  AxesLabel -> {"t", "x[t]"}, PlotStyle -> {RGBColor[0, 0, 1]}]
fasependforcrit = ParametricPlot[
  Evaluate[{x[t], x'[t]} /. penduloforcrit[[1]]],
  {t, 0, 40 Pi}, PlotStyle -> {RGBColor[0, 0, 1]}]
```

Out[117]=
```
{{x → InterpolatingFunction[{{0., 125.664}}, <>]}}
```



Out[118]=
```
- Graphics -
```



Out[119]=
```
- Graphics -
```

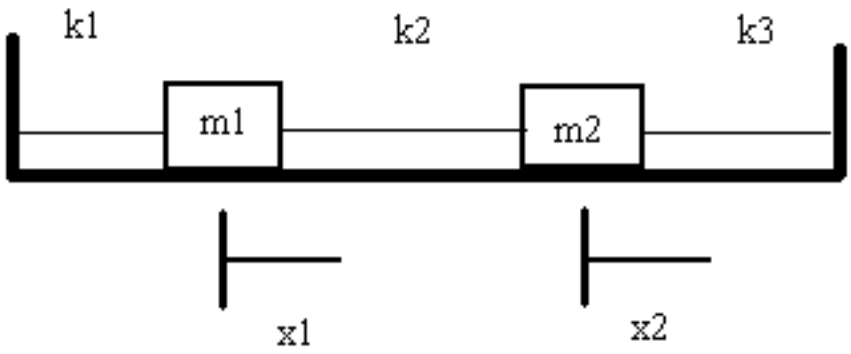
La solución presenta un aspecto "caótico"...

Ejercicio

Represente la trayectoria y el diagrama de fases de un péndulo con frecuencia natural ω_0=1, sometido a una fuerza externa del tipo *f(t)=0.5 sen(3t)* y a un amortiguamiento de β=0.2, que parte del reposo en la posición vertical (x[0]=0).

Sistemas de EDOs. Oscilaciones acopladas. Estabilidad

■ Oscilaciones acopladas. Modos normales de vibración



Consideremos el sistema de dos bloques de masas m_1 y m_2 sujetos a muelles de constantes elásticas k_1 y k_2, como se muestra en la figura anterior. La ecuación difeencial que gobierna el movimiento de ambos bloques cuando ambos se desplazan cantidades x_1 y x_2 respecto de su posición de equilibrio es:

$$\begin{cases} m_1\,\ddot{x}_1 = -(k_2 + k_1)\,x_1 + k_2\,x_2 \\ m_2\,\ddot{x}_2 = k_2\,x_1 - (k_2 + k_3)\,x_2 \end{cases}$$

Pongamos un nombre a dichas ecuaciones:

```
In[1]:= ec1 = m1 x1''[t] + (k2 + k1) x1[t] - k2 x2[t];
        ec2 = m2 x2''[t] - k2 x1[t] + (k2 + k3) x2[t];
```

Podemos resolver de forma exacta dichas ecuaciones diferenciales con condiciones iniciales:

posición inicial: $x_1(0) = x_1^{(0)}$, $x_2(0) = x_2^{(0)}$

velocidad inicial: $\dot{x}_1(0) = v_1^{(0)}$, $\dot{x}_2(0) = v_2^{(0)}$

Tomemos, por ejemplo, $m_1 = m_2 = 1$ y $k_1 = k_2 = k_3 = 1$.

```
In[3]:= m1 = 1; m2 = 1; k1 = 1; k2 = 1; k3 = 1;
        osciacop = DSolve[{ec1 == 0, ec2 == 0, x1[0] == x10,
           x2[0] == x20, x1'[0] == v1, x2'[0] == v2}, {x1, x2}, t]
```

```
Out[4]= {{x1 → (1/12 e^(-i #1-i √3 #1) (i √3 e^(i #1) v1 + 3 i e^(i √3 #1) v1 - 3 i e^(2 i #1+i √3 #1) v1 - i √3
            e^(i #1+2 i √3 #1) v1 - i √3 e^(i #1) v2 + 3 i e^(i √3 #1) v2 - 3 i e^(2 i #1+i √3 #1) v2 +
          i √3 e^(i #1+2 i √3 #1) v2 + 3 e^(i #1) x10 + 3 e^(i √3 #1) x10 +
          3 e^(2 i #1+i √3 #1) x10 + 3 e^(i #1+2 i √3 #1) x10 - 3 e^(i #1) x20 +
          3 e^(i √3 #1) x20 + 3 e^(2 i #1+i √3 #1) x20 - 3 e^(i #1+2 i √3 #1) x20) &),
        x2 → (1/12 e^(-i #1-i √3 #1) (-i √3 e^(i #1) v1 + 3 i e^(i √3 #1) v1 - 3 i e^(2 i #1+i √3 #1) v1 +
          i √3 e^(i #1+2 i √3 #1) v1 + i √3 e^(i #1) v2 + 3 i e^(i √3 #1) v2 -
          3 i e^(2 i #1+i √3 #1) v2 - i √3 e^(i #1+2 i √3 #1) v2 - 3 e^(i #1) x10 +
          3 e^(i √3 #1) x10 + 3 e^(2 i #1+i √3 #1) x10 - 3 e^(i #1+2 i √3 #1) x10 + 3 e^(i #1) x20 +
          3 e^(i √3 #1) x20 + 3 e^(2 i #1+i √3 #1) x20 + 3 e^(i #1+2 i √3 #1) x20) &)}}
```

Las frecuencias naturales (al cuadrado y cambiadas de signo) de oscilación y los modos normales de vibración son los valores propios y vectores propios, respectivamente, de la matriz de coeficientes

$$\begin{pmatrix} -(k_2+k_1)/m_1 & k_2/m_1 \\ k_2/m_2 & -(k_2+k_3)/m_2 \end{pmatrix}$$

```
In[5]:= mcoef = {{-(k2 + k1) / m1, k2 / m1}, {k2 / m2, -(k2 + k3) / m2}}
        {ω1c, ω2c} = Eigenvalues[mcoef]
        paso = Transpose[Eigenvectors[mcoef]]
```

```
Out[5]= {{-2, 1}, {1, -2}}

Out[6]= {-3, -1}

Out[7]= {{-1, 1}, {1, 1}}
```

de manera que las frecuencias propias naturales de oscilación son: $\omega_1 = \sqrt{3}$, $\omega_2 = 1$ y los modos normales y_1, y_2 se obtienen a partir de $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = P\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \Longrightarrow \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = P^{-1}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, donde P es la matriz de paso. o sea:

```
In[8]:= {y1[t], y2[t]} = Inverse[paso].{x1[t], x2[t]}
```

```
Out[8]= {-x1[t]/2 + x2[t]/2, x1[t]/2 + x2[t]/2}
```

Este cambio de función incógnita nos desacopla el sistema de ecuaciones quedándo simplemente:

$$\ddot{y}_1 = -\omega_1{}^2\, y_1$$
$$\ddot{y}_2 = -\omega_2{}^2\, y_2$$

Para "estimular" el modo 1 debemos hacer $y_2 = 0$, lo cual se consigue tomando como condiciones iniciales, por ejemplo:

posición inicial: $x_1(0) = -x_2(0) = a = 1$

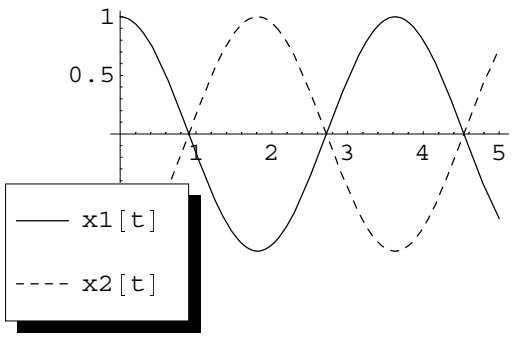
velocidad inicial: $\dot{x}_1(0) = 0$, $\dot{x}_2(0) = 0$

Introduzcamos estas condiciones iniciales

```
In[9]:= x10 = 1; x20 = -1; v1 = 0; v2 = 0;
```

y representemos este modo de vibración:

```
In[10]:=
<< Graphics`Legend`
Plot[{Evaluate[x1[t] /. osciacop], Evaluate[x2[t] /. osciacop]},
 {t, 0, 5}, PlotStyle -> {GrayLevel[0.], Dashing[{0.02, 0.02}]},
 PlotLegend -> {"x1[t]", "x2[t]"}]
```




```
Out[11]= - Graphics -
```

Así hemos estimulado el modo ANTISIMÉTRICO y_1 (los bloques se separan y se juntan de forma periódica) que tiene una frecuencia $\omega_1 = \sqrt{3}$ mayor que la del modo y_2 (y, por lo tanto mayor energía ¿por qué?).

Para estimular el modo y_2 debemos hacer $y_1 = 0 \Longrightarrow$

posición inicial: $x_1(0) = x_2(0) = 1$

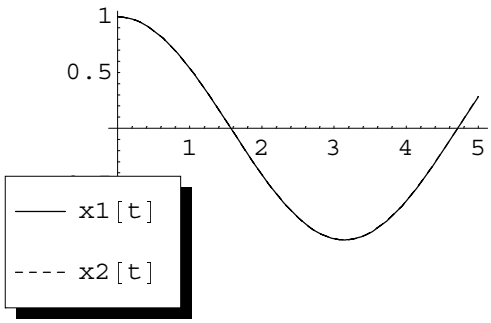
velocidad inicial: $\dot{x}_1(0) = 0, \dot{x}_2(0) = 0$

Introduciendo estas condiciones iniciales

```
In[12]:= x10 = 1; x20 = 1; v1 = 0; v2 = 0;
```

y representemos este modo de vibración:

In[13]:=
```
<< Graphics`Legend`
Plot[{Evaluate[x1[t] /. osciacop], Evaluate[x2[t] /. osciacop]},
 {t, 0, 5}, PlotStyle -> {GrayLevel[0.], Dashing[{0.02, 0.02}]},
 PlotLegend -> {"x1[t]", "x2[t]"}]
```



Out[14]= - Graphics -

vemos que ambos bloques se mueven al unísono (modo SIMÉTRICO), resultando un movimiento de menor frecuencia y, por lo tanto, de menor energía.

Ejercicio

Resuelva el problema anterior para $k_1 = k_3 = 1$, $k_2 = 2$ y $m_1 = 2\, m_2 = 1$. Identifique las frecuencias propias y los modos normales de vibración. Elija condiciones iniciales adecuadas para estimular el modo 1 y represéntelo gráficamente. Haga lo mismo para el modo 2.

Ejercicio

Considere un bloque de 7 pisos en el que las oscilaciones transversales del terreno inducen un movimiento horizontal en cada uno de los pisos, de forma que el piso número *i* está acoplado al número *i+1* y al *i-1* mediante la ecuación

$$m_i\, \ddot{x}_i = k(x_{i+1} - x_i) - k(x_i - x_{i-1})$$

Cada piso pesa *m=16* toneladas y existe una fuerza horizontal restauradora interna entre cada piso con constante elástica *k=160* toneladas por decímetro. Calcular las 7 frecuencias naturales propias de oscilación. ¿Entraría en resonancia el edificio con un temblor de tierra de frecuencia $\omega \approx 7$ seg^{-1}? ¿Qué modo entra en resonancia con una frecuencia $\omega \approx 2$ seg^{-1}?

■ Estabilidad

Nos planteamos estudiar la estabilidad de un sistema de ecuaciones de primer orden como:

$$\begin{cases} \dot{x} = x + 3\, y \\ \dot{y} = -x \end{cases}$$

La matriz del sistema es:

In[15]:=
```
A = {{1, 3}, {-1, 0}};
```

Y su traza y su determinante son:

```
In[16]:= Tr[A]
         Det[A]
Out[16]= 1
Out[17]= 3
```

Luego tenemos una espiral o foco inestable. Definamos una familia de soluciones:

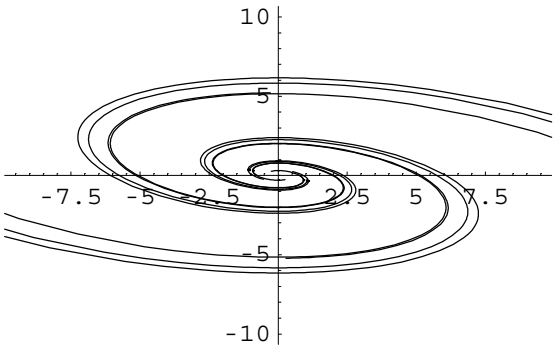
```
In[18]:= Sol[t_, x0_, y0_] :=
          {x[t], y[t]} /. DSolve[{x'[t] == x[t] + 3 y[t], y'[t] == -x[t],
             x[0] == x0, y[0] == y0}, {x[t], y[t]}, t]
```

Tomemos unos cuantos miembros de esta familia biparamétrica de soluciones en una tabla

```
In[19]:= familia = Table[Sol[t, x0, y0], {x0, -2, 2, 2}, {y0, -2, 2, 2}];
```

y representémosla en un diagrama

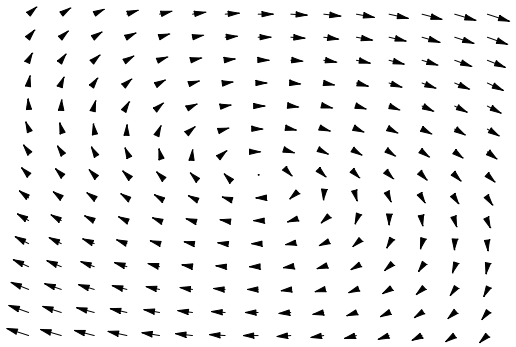
```
In[20]:= curvas = ParametricPlot[Evaluate[Flatten[familia, 2]], {t, -3, 3}]
```




```
Out[20]= - Graphics -
```

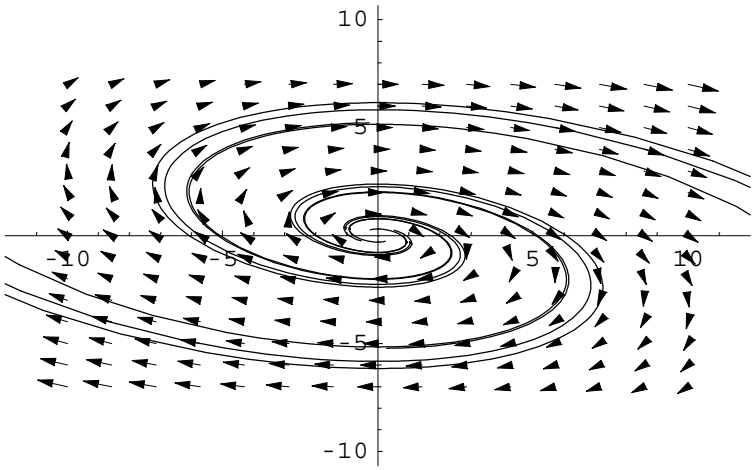
Vemos claramente que tenemos un punto espiral. Para verificar que, efectivamente, es inestable, realizaremos una representación del campo de velocidades:

```
In[21]:= << Graphics`PlotField`
         velocidades = PlotVectorField[{x + 3 y, -x}, {x, -10, 10}, {y, -7, 7}];
```



que apunta claramente hacia afuera del punto espiral (inestable). Superpongamos ambos gráficos:

In[23]:= **Show[curvas, velocidades]**



Out[23]= - Graphics -

Ejercicio

Dado el sistema $\begin{cases} \dot{x} = x \\ \dot{y} = -x + 2\,y \end{cases}$ determine qué tipo de punto crítico es *(x,y)=(0,0)* (nodo, vórtice, espiral o punto de silla) y diga si es estable, inestable o asintóticamente estable. Realize un gráfico con una familia de soluciones y otro con el campo de velocidades y superpóngalos (como en el ejercicio anterior).

Campos de velocidades, líneas de corriente y superficies equipotenciales

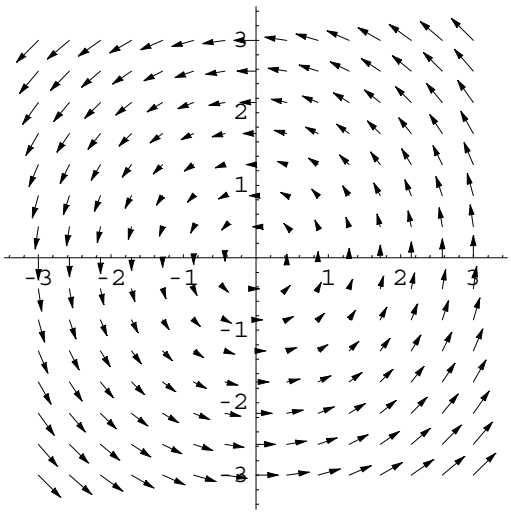
■ Campos de velocidades y líneas de corriente

Dado el campo de velocidades (digamos, de un fluido) en el espacio
$\vec{V}(x, y, z) = V_x(x, y, z)\,\hat{i} + V_y(x, y, z)\,\hat{j} + V_z(x, y, z)\,\hat{k}$, las líneas de corriente (digamos, las trayectorias de las partículas del fluido) son tangentes a dicho campo en cada punto del espacio, y vienen dadas por el sistema de ecuaciones diferenciales:

$$\frac{d\vec{r}}{dt} = \vec{V} \Longrightarrow \begin{cases} \frac{dx}{dt} = V_x \\ \frac{dy}{dt} = V_y \\ \frac{dz}{dt} = V_z \end{cases} \Longrightarrow dt = \frac{dx}{V_x} = \frac{dy}{V_y} = \frac{dz}{V_z}$$

Por ejemplo, tomemos el campo de velocidades en el plano dado por $\vec{V}(x, y) = (-y, x)$. Una representación gráfica del mismo

```
In[1]:= << Graphics`PlotField`
        remolino = PlotVectorField[{-y, x}, {x, -3, 3}, {y, -3, 3}, Axes → True]
```




```
Out[2]= - Graphics -
```

nos muestra un "vórtice o remolino". Se trata de un campo incompresible (de divergencia nula) pero NO irrotacional (o NO "conservativo"); en efecto:

```
In[3]:= << Calculus`VectorAnalysis`
        Div[{-y, x, 0}, Cartesian[x, y, z]]
        Curl[{-y, x, 0}, Cartesian[x, y, z]]

Out[4]= 0

Out[5]= {0, 0, 2}
```

Las líneas de corriente son las soluciones de:

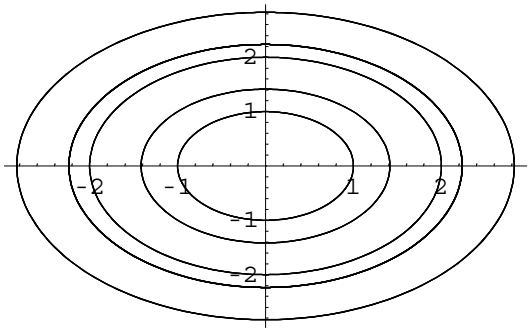
```
In[6]:= Clear[t, x0, y0]
        lincor[t_, x0_, y0_] :=
         {x[t], y[t]} /. DSolve[{x'[t] == -y[t], y'[t] == x[t],
            x[0] == x0, y[0] == y0}, {x[t], y[t]}, t]
```

Tomemos unos cuantos miembros de esta familia biparamétrica de soluciones en una tabla

```
In[8]:= familia = Table[lincor[t, x0, y0], {x0, -2, 2}, {y0, -2, 2}];
```

y representémosla en un diagrama

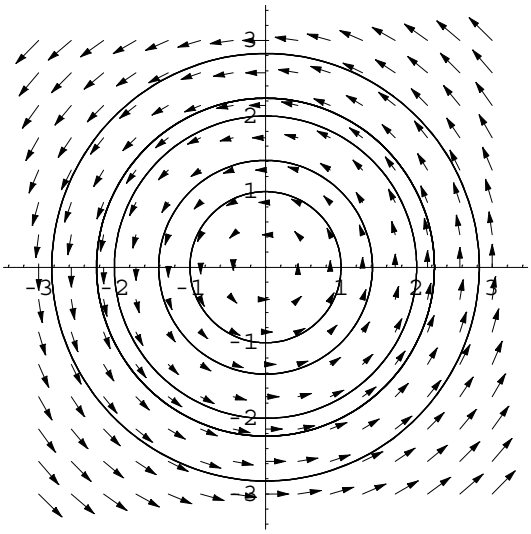
```
In[9]:= lineas = ParametricPlot[Evaluate[Flatten[familia, 2]], {t, -3, 3}]
```




```
Out[9]= - Graphics -
```

Superponiendo el campo de velocidades y las líneas de corriente tenemos:

```
In[10]:= Show[remolino, lineas]
```




```
Out[10]= - Graphics -
```

Ejercicio

Dibujar el campo de velocidades y las líneas de corriente del campo $\vec{V}(x, y) = (y^2, x)$. ¿Es incompresible?, ¿es irrotacional?.

Ejercicio

Dibujar el campo de velocidades y las líneas de corriente del campo $\vec{V}(x, y, z) = (3\,x^2, -y, z^3)$. ¿Es incompresible?, ¿es irrotacional?.

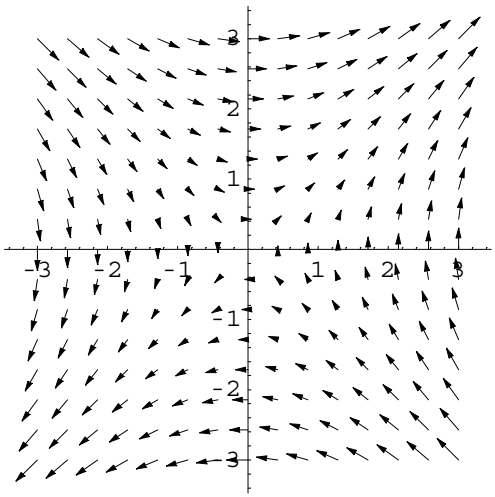
■ Campos irrotacionales: superficies equipotenciales. Ecuaciones de Laplace y Poisson

Consideremos el caso en que el campo de velocidades sea irrotacional

$$\vec{\nabla} \times \vec{V}(x, y, z) = \vec{0} \Longrightarrow \vec{V}(x, y, z) = \vec{\nabla} U(x, y, z)$$

donde U es la función potencial. El campo $\vec{V}$ es perpendicular a las superficies (curvas, en el plano) de nivel *U(x,y,z)=cte* o "superficies equipotenciales". Así, las líneas de corriente son ortogonales a las superficies (curvas, en el plano) equipotenciales. Por ejemplo, consideremos el campo $\vec{V}(x, y) = (y, x)$

In[11]:=
```
campo = PlotVectorField[{y, x}, {x, -3, 3}, {y, -3, 3}, Axes → True]
```



Out[11]=
```
- Graphics -
```

Veamos que se trata de un campo incompresible e irrotacional

In[12]:=
```
Div[{y, x, 0}, Cartesian[x, y, z]]
Curl[{y, x, 0}, Cartesian[x, y, z]]
```

Out[12]=
```
0
```

Out[13]=
```
{0, 0, 0}
```

El potencial U del que deriva el campo de velocidades es *U(x,y)=xy*. En efecto:

In[14]:=
```
Grad[x y, Cartesian[x, y, z]]
```

Out[14]=
```
{y, x, 0}
```

Veamos cómo obtenerlo en general. Definamos el campo de velocidades e integremos V_x en x

In[15]:=
```
v[x_, y_] := {y, x}
U = ∫_0^x v[t, y][[1]] dt + f[y]
```

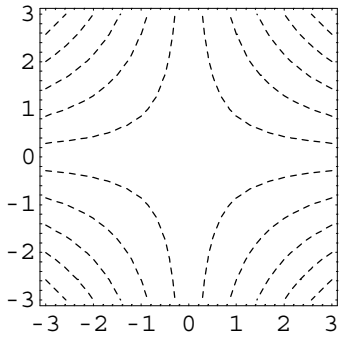
Out[16]=
```
x y + f[y]
```

Obtenemos U salvo una función arbitraria *f(y)*. Para evaluar la función arbitraria hacemos:

```
In[17]:= Solve[∂y U == v[x, y][[2]], f'[y]];
         dfy = Evaluate[f'[y] /. Flatten[%]] /. y → t
         ∫_0^y dfy dt
Out[18]= 0
Out[19]= 0
```

Obteniendo *f(y)=0⇒ U(x,y)=xy*. Hagamos una representación gráfica de las curvas equipotenciales *U(x,y)=xy=cte*

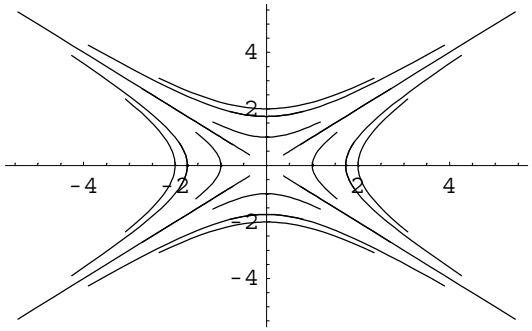
```
In[20]:= curvequip = ContourPlot[x y, {x, -3, 3}, {y, -3, 3},
           ContourShading -> False, ContourStyle -> Dashing[{.02}]]
```




```
Out[20]= - ContourGraphics -
```

y de las líneas de corriente:

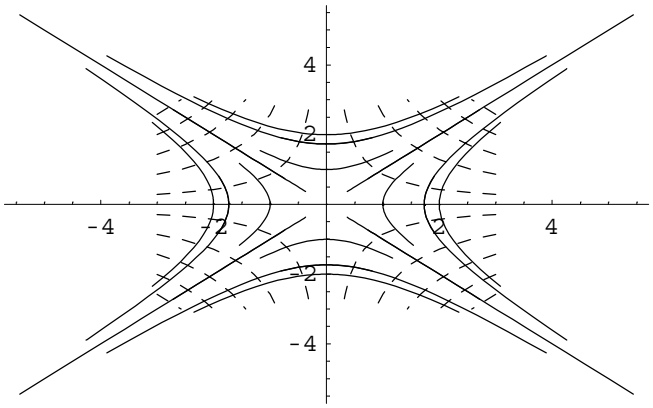
```
In[21]:= Clear[t, x0, y0]
         lincor[t_, x0_, y0_] :=
           {x[t], y[t]} /. DSolve[{x'[t] == y[t], y'[t] == x[t],
              x[0] == x0, y[0] == y0}, {x[t], y[t]}, t];
         familia = Table[lincor[t, x0, y0], {x0, -2, 2}, {y0, -2, 2}];
         lineas = ParametricPlot[Evaluate[Flatten[familia, 2]], {t, -1, 1}]
```




```
Out[24]= - Graphics -
```

Superponiendo las líneas de corriente a las curvas equipotenciales

In[25]:=
```
Show[lineas, curvequip]
```



Out[25]= - Graphics -

Vemos que se trata efectivamente de trayectorias ortogonales.

Recuerde que un campo conservativo $\vec{V} = \vec{\nabla} U$ e incompresible $\vec{\nabla} \cdot \vec{V} = 0$ satisface la *ecuación de Laplace*

$$\vec{\nabla} \cdot \vec{\nabla} U = \Delta \mathrm{U} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 0$$

$\vec{\nabla} \cdot \vec{\nabla} = (\vec{\nabla})^2 = \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ es el operador Laplaciano. Si el campo no es incompresible $\vec{\nabla} \cdot \vec{V} = f$, entonces la ecuación que se satisface es la de *Poisson*

$$\vec{\nabla} \cdot \vec{\nabla} U = \Delta \mathrm{U} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = f$$

Por ejemplo, veamos que el potencial anterior *U(x,y)=xy* verifica la ecuación de Laplace:

In[26]:=
```
Laplacian[x y, Cartesian[x, y, z]]
```

Out[26]= 0

Ejercicio

Dado el campo de velocidades $\vec{V}(x, y) = \{2\,x\,y,\ x^2 - y^2\}$, demostrar que es irrotacional y encontrar el potencial *U* del que deriva. Representar el campo, las líneas de corriente y las curvas equipotenciales, y superponer dichos gráficos para cerciorarse de que 1) el campo es tangente a las líneas de corriente y 2) las líneas de corriente son ortogonales a las curvas equipotenciales. ¿Qué ecuación verifica al potencial, la de Laplace o la de Poisson?

Integrales curvilíneas y de superficie: Teoremas de Stokes y Gauss

■ Integrales de línea

Recuerde que el trabajo $W_{1\to 2}$ de una fuerza $\vec{F}$ para llevar a una partícula desde $\vec{r}_1$ hasta $\vec{r}_2$ viene dado por la integral:

$$W_{1\to 2}=\int_{\vec{r}_1}^{\vec{r}_2}\vec{F}\cdot d\vec{r}=\int_{t_1}^{t_2}\vec{F}\cdot\vec{v}\,dt$$

donde $\vec{v}$ es la velocidad si parametrizamos la curva $\vec{r}(t)$ por el tiempo. Esto quiere decir que sólo la componente tangencial de la fuerza sobre la trayectoria realiza trabajo. Por ejemplo, tomemos $\vec{F}(x, y) = x^2\,\hat{i} + y\,\hat{j}$ y como trayectoria la semicircunferencia superior de radio 1:

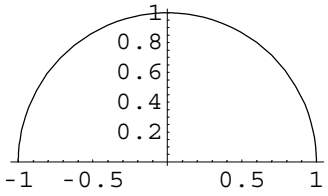
In[1]:=
```
Clear[f, r]
f[x_, y_] := {x^2, y}
r[t_] := {Cos[t], Sin[t]}
```

Con esta parametrización, el vector tangente a la trayectoria (la "velocidad" en caso de un movimiento circula uniforme de frecuencia 1) es

In[4]:=
```
∂t r[t]
```
Out[4]=
```
{-Sin[t], Cos[t]}
```

La trayectoria es

In[5]:=
```
semicirc = ParametricPlot[{Cos[t], Sin[t]}, {t, 0, π}, AspectRatio → Automatic];
```



y la componente tangencial de la fuerza sobre la trayectoria es proporcional a:

In[6]:=
```
f[r[t][[1]], r[t][[2]]].∂t r[t]
```
Out[6]=
```
Cos[t] Sin[t] - Cos[t]^2 Sin[t]
```

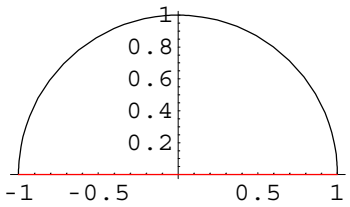
El trabajo se calcula como:

In[7]:=
```
I1 = ∫_0^π f[r[t][[1]], r[t][[2]]].∂t r[t] dt
```
Out[7]=
$$-\frac{2}{3}$$
$$-\frac{2}{3}$$

Supongamos ahora que la trayectoria está compuesta por varios trozos. Por ejemplo, cerremos el semicírculo anterior con el intervalo [-1,1] en el eje X (en color rojo)

In[8]:=
```
r2[t_] := {t, 0}
lineainferior = ParametricPlot[{t, 0}, {t, −1, 1},
    PlotStyle → {RGBColor[1, 0, 0]}, DisplayFunction −> Identity];
```

In[10]:=
```
Show[semicirc, lineainferior, DisplayFunction −> $DisplayFunction];
```



Para calcular la circulación de $\vec{F}$ a lo largo de la trayectoria cerrada "semicírculo+[-1,1]" (en sentido antihorario) tenemos que añadir la integral

In[11]:=
```
I2 = ∫_-1^1 f[r2[t][[1]], r2[t][[2]]].∂_t r2[t] dt
```

Out[11]=
$$\frac{2}{3}$$

de manera que la circulación (o el trabajo a lo largo de la trayectoria cerrada) es :

In[12]:=
```
I1 + I2
```

Out[12]=
```
0
```

El procedimiento es el mismo para un campo y una curva en tres dimensiones como, por ejemplo:

In[13]:=
```
Clear[f, r]
f[x_, y_, z_] := {1, −y, x y z}
r[t_] := {t, −t^2, t}
```

El trabajo desde *t=0* hasta *t=1* es:

In[16]:=
```
∫_0^1 f[r[t][[1]], r[t][[2]], r[t][[3]]].∂_t r[t] dt
```

Out[16]=
$$\frac{3}{10}$$

Ejercicio

Calcular el trabajo de un muelle $\vec{F} = -k\,\vec{r}$ a lo largo de una hélice $\vec{r}(t) = (\cos t,\ \mathrm{sen}\, t,\ t)$ entre *t=0* y *t=2*.

■ Flujos a través de curvas planas

Recuerde que el flujo ("caudal") $\Phi_C(\vec{V})$ de un campo vectorial $\vec{V}$ ("velocidad de un fluido") bidimansional a través de una curva plana C entre $\vec{r}_1$ y $\vec{r}_2$ viene dado por:

$$\Phi_C(\vec{V}) = \int_{\vec{r}_1}^{\vec{r}_2} \vec{F} \cdot d\vec{S} = \int_{t_1}^{t_2} \vec{F} \cdot \hat{n}\, dS$$

donde $\hat{n}$ es un vector unitario normal a la curva y dS es un arco de curva infinitesimal. Si la curva viene dada en forma implícita como $g(x,y)=0$, entonces $\hat{n} = \frac{\vec{\nabla} g}{||\vec{\nabla} g||}$ es un vector unitario normal y $dS = \sqrt{(dx)^2 + (dy)^2} = \sqrt{1 + (y')^2}\, dx$ es el elemento de arco. Si la curva viene dada en forma paramétrica $\vec{r} = \vec{r}(t)$, entonces un vector tangente unitario es $\hat{\tau} = \frac{d\vec{r}}{dt} \Big/ || \frac{d\vec{r}}{dt} ||$ y un vector normal unitario es $\hat{n} = \frac{d\hat{\tau}}{dt} \Big/ || \frac{d\hat{\tau}}{dt} ||$, mientras que $dS = \sqrt{(dx)^2 + (dy)^2} = \tau\, dt$ es el elemento de arco, donde $\tau = || \frac{d\vec{r}}{dt} ||$. Una forma más operativa de encontrar el vector normal es, teniendo en cuanta que $\vec{\tau} = (\dot{x}, \dot{y}) \Longrightarrow \vec{n} = (-\dot{y}, \dot{x})$.

Por ejemplo, calculemos el flujo de $\vec{v}(x, y) = (x, y)$ a través de la circunferencia de radio 3, $x^2 + y^2 = 9$:

```
In[17]:= Clear[v, f, r, flujo, n]
         v[x_, y_, z_] := {x, y, 0}
         g[x_, y_, z_] := x^2 + y^2 - 9
```

Un vector unitario normal a la circunferencia es:

```
In[20]:= << Calculus`VectorAnalysis`
         n = Grad[g[x, y, z], Cartesian[x, y, z]];
         nu = n / Sqrt[n.n];
         Simplify[%]
```

$$\text{Out[23]=} \quad \left\{ \frac{x}{\sqrt{x^2 + y^2}}, \frac{y}{\sqrt{x^2 + y^2}}, 0 \right\}$$

La componente normal del campo de velocidades es

```
In[24]:= vn = v[x, y, z].nu;
         Simplify[%]
```

$$\text{Out[25]=} \quad \sqrt{x^2 + y^2}$$

Si parametrizamos la curva en función del ángulo θ como:

```
In[26]:= r[θ_] := {3 Cos[θ], 3 Sin[θ]}
```

La componente normal anterior es simplemente:

```
In[27]:= vnθ = vn /. {x → 3 Cos[t], y → 3 Sin[t]};
         Simplify[%]
```

```
Out[28]= 3
```

Luego el flujo a través de la circunferencia es:

```
In[29]:= flujo = ∫_0^(2π) vnθ Sqrt[∂_t r[t].∂_t r[t]] dt
```

```
Out[29]= 18 π
```

■ Áreas de superficies e integrales de superficie.

Áreas de superficies

Dada una superficie en forma implícita $F(x,y,z)=0$, el área de dicha superficie viene dada por $A=\iint d\,S$ donde $d\,S$ es un elemento de área. Sea $d\,\vec{S} = \hat{n}\,d\,S$ un vector perpendicular a la superficie de módulo $d\,S$, donde $\hat{n}$ es un vector unitario normal a la superficie, por ejemplo:

$\hat{n} = \frac{\nabla F}{||\vec{\nabla} F||} = \frac{\partial_x F\,\hat{\imath}+\partial_y F\,\hat{\jmath}+\partial_z F\,\hat{k}}{\sqrt{(\partial_x F)^2+(\partial_y F)^2+(\partial_z F)^2}}$. Proyectando sobre el plano X-Y obtenemos

$$d\,\vec{S}\cdot\hat{k} = \hat{n}\cdot\hat{k}\,d\,S = d\,x\,d\,y \Rightarrow d\,S = \frac{d\,x\,d\,y}{\hat{n}\cdot\hat{k}} = \frac{\sqrt{(\partial_x F)^2+(\partial_y F)^2+(\partial_z F)^2}}{\partial_z F}\,d\,x\,d\,y$$

de modo que el área viene dada por la integral doble

$$A=\iint d\,S=\int\int \frac{\sqrt{(\partial_x F)^2+(\partial_y F)^2+(\partial_z F)^2}}{\partial_z F}\,d\,x\,d\,y$$

El caso de superficies dadas de forma explícita $z=S(x,y)$ es un caso particular sin mas que tomar $F(x,y,z)=z-S(x,y)$. En este caso la fórmula se simplifica algo más:

$$A=\iint d\,S=\int\int \sqrt{(\partial_x S)^2+(\partial_y S)^2+1}\,d\,x\,d\,y$$

Por ejemplo, el área de una esfera de radio r es $A = 4\,\pi\,r^2$. Veamos cómo obtener este resultado para una esfera de radio 1. Calculemos un vector unitario normal a la esfera en cada punto:

```
In[30]:= Clear[F, x, y, z, n, nu];
         F[x_, y_, z_] := x^2 + y^2 + z^2 - 1
         n = Grad[F[x, y, z], Cartesian[x, y, z]];
         nu = n / Sqrt[n.n];
         Simplify[%]
```

$$\text{Out[34]= } \left\{\frac{x}{\sqrt{x^2+y^2+z^2}},\ \frac{y}{\sqrt{x^2+y^2+z^2}},\ \frac{z}{\sqrt{x^2+y^2+z^2}}\right\}$$

El área del casquete superior se calcula como:

```
In[35]:= z = Sqrt[1 - x^2 - y^2];
         Integrate[Integrate[1 / nu.{0, 0, 1}, {y, -Sqrt[1-x^2], Sqrt[1-x^2]}], {x, -1, 1}]
```

$$\text{Out[36]= } 2\,\pi$$

En el caso en que la superficie venga parametrizada $\vec{r} = \vec{r}(u,\,v) \longrightarrow \begin{cases} x = x(u,\,v) \\ y = y(u,\,v) \\ z = z(u,\,v) \end{cases}$, el vector normal $\hat{n}$ se puede calcular como:

como:

$$\hat{n} = \frac{\partial_u \vec{r} \times \partial_v \vec{r}}{||\partial_u \vec{r} \times \partial_v \vec{r}||}$$

y el área se puede calcular como

$$A = \int\int || \partial_u \vec{r} \times \partial_v \vec{r} || \, du \, dv$$

Por ejemplo, para el caso de la esfera de radio 1, el jacobiano $|| \partial_u \vec{r} \times \partial_v \vec{r} ||$, el vector normal $\hat{n}$ y el área A son:

```
In[37]:= R[φ_, θ_] := {Sin[θ] Cos[φ], Sin[θ] Sin[φ], Cos[θ]}
         Rφ = ∂φ R[φ, θ]; Rθ = ∂θ R[φ, θ];
         jaco = √Cross[Rφ, Rθ].Cross[Rφ, Rθ] ; Simplify[jaco]
         n = Cross[Rφ, Rθ] / jaco; Simplify[n]
         ∫0^π (∫0^(2 π) jaco dφ) dθ
         General::spell1 : Possible spelling error: new
            symbol name "Rθ" is similar to existing symbol "Rφ".
Out[39]= √Sin[θ]^2
Out[40]= {-Cos[φ] √Sin[θ]^2, -√Sin[θ]^2 Sin[φ], -Cot[θ] √Sin[θ]^2}
Out[41]= 4 π
```

Nótese que el vector normal unitario $\hat{n}$ apunta aquí hacia adentro de la esfera, mientras que el convenio de orientación dice que debemos tomarlo apuntando hacia afuera.

Ejercicio

Hallar el área del paraboloide:

$$z = 1 - x^2 - y^2, \; z \geq 0$$

y del cono:

$x=x(r,\varphi)=r\ cos(\varphi)$, $y=y(r,\varphi)=r\ sen(\varphi)$, $z=z(r,\varphi)=2r$.

Flujos a través de superficies

El flujo ("caudal") de un campo vectorial $\vec{V}$ ("un campo de velocidades de un fluido") a través de una superficie Σ viene dado por la integral doble

$$\Phi_\Sigma(\vec{V}) = \int\int_\Sigma \vec{V} \cdot \hat{n}\, dS$$

Por ejemplo, consideremos el campo $\vec{V} = (x^2, y^2, z^2)$ y como superficie una esfera de radio 1. En coordenadas esféricas tenemos que el campo $\vec{V}$ es:

```
In[42]:= Clear[V, x, y, z];
         {x, y, z} = {Sin[θ] Cos[φ], Sin[θ] Sin[φ], Cos[θ]};
         V[x_, y_, z_] := {x^2, y , z ^2}; V[x, y, z]
Out[44]= {Cos[φ]^2 Sin[Θ]^2, Sin[Θ] Sin[φ], Cos[Θ]^2}
```

y el flujo (hacia afuera) de $\vec{V}$ a través de la esfera de radio 1 viene dado por:

```
In[45]:= jaco = Sin[Θ];
         ∫_0^π (∫_0^(2 π) V[x, y, z].(-n) jaco dφ) dΘ
Out[46]= 4 π / 3
```

Verifiquemos que se cumple es Teorema de Gauss para una superficie cerrada Σ como es la esfera

$$\Phi_\Sigma(\vec{V}) = \int\int_\Sigma \vec{V} \cdot \hat{n}\, dS = \int\int\int_\Omega \vec{\nabla} \cdot \vec{V}\, dx\, dy\, dz$$

donde Ω es el volumen que encierra la superficie cerrada Σ. En efecto, utilizemos coordenadas esféricas, recordando que el elemento de voluman es $dx\, dy\, dz = r^2 \operatorname{sen}\theta\, dr d\varphi d\theta$:

```
In[47]:= Clear[x, y, z];
         diverg = Div[V[x, y, z], Cartesian[x, y, z]];
         {x, y, z} = {r Sin[Θ] Cos[φ], r Sin[Θ] Sin[φ], r Cos[Θ]};
         ∫_0^π (∫_0^(2 π) (∫_0^1 diverg * r ^ 2 * Sin[Θ] dr) dφ) dΘ
Out[50]= 4 π / 3
```

Obtenemos pues el mismo resultado, indicando ésto que se cumplen las hipótesis del Teorema de Gauss.

Ejercicio

Encontrar el flujo del campo vectorial $\vec{V}(x, y, z) = x\,\hat{i} + y\,\hat{j} + z\,\hat{k}$ a través de la superficie de un paraboloide $z = 1 - x^2 - y^2,\ z \geq 0$. Encuentre también el flujo a través del círculo $x^2 + y^2 \leq 1$. Verifique el Teorema de Gauss para la superficie cerrada compuesta por el paraboloide y el círculo.

Ecuaciones en Derivadas Parciales. Animaciones

■ Cuerda vibrante

La ecuación que gobierna las oscilaciones verticales de pequeña amplitud de una cuerda vibrante de longitud L, densidad ρ y tensión τ, sujeta por ambos extremos es:

$\frac{\partial^2 y(x,t)}{\partial t^2} = v^2 \frac{\partial^2 y(x,t)}{\partial x^2}$ con condiciones iniciales $y(x, 0) = y_0(x)$, $\dot{y}(x, 0) = \dot{y}_0(x)$ y de contorno $y(0, t) = y(L, t) = \dot{y}(0, t) = \dot{y}(L, t) = 0$

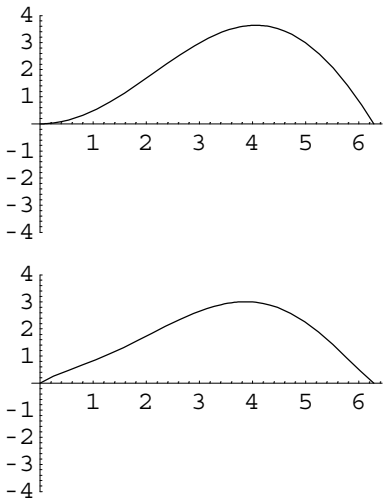
donde $v = \sqrt{\frac{\rho}{\tau}}$ es la velocidad de fase.

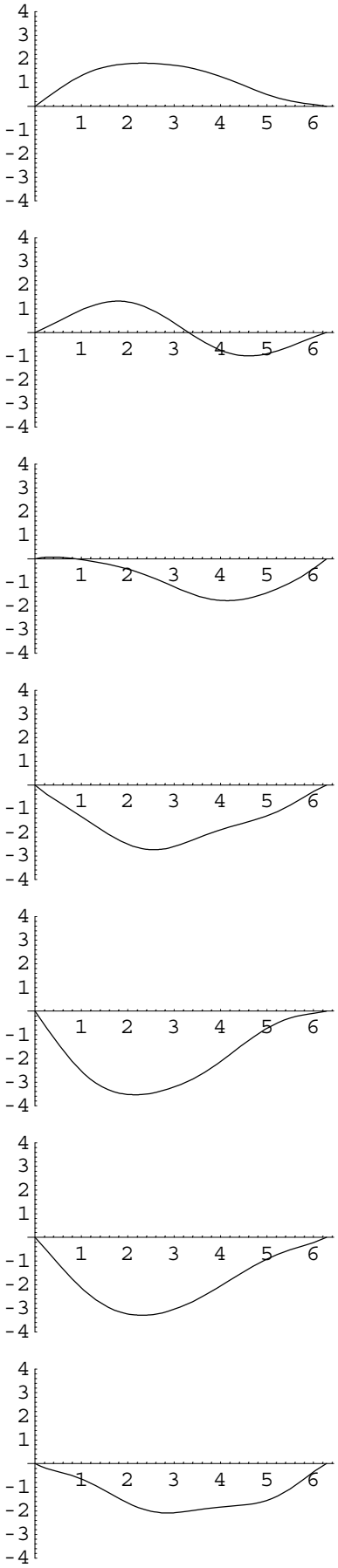
Tomemos por ejemplo $v=1$, $L=2\pi$ y las condiciones iniciales $y(x,0)=x\ sen(x/2)$, $\dot{y}(x, 0) = 0$ (velocidad inicial nula). La solución numérica de la ecuación de ondas entre t=0 y t=13 segundos viene dada por:

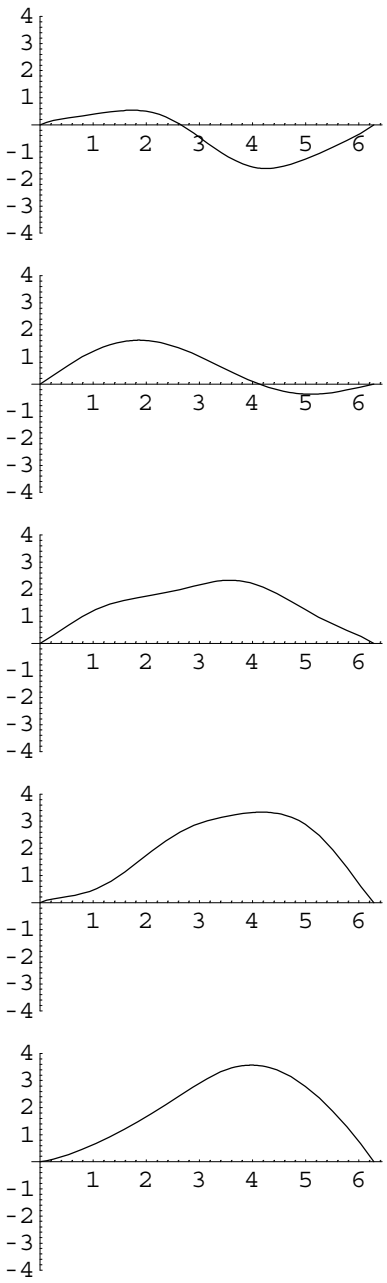
```
In[1]:= cuerda = NDSolve[{D[y[x, t], t, t] == D[y[x, t], x, x],
          y[x, 0] == x Sin[x / 2], Derivative[0, 1][y][x, 0] == 0,
          y[0, t] == 0, y[2 π, t] == 0, Derivative[0, 1][y][0, t] == 0,
          Derivative[0, 1][y][2 π, t] == 0}, y, {x, 0, 2 π}, {t, 0, 13}]
Out[1]= {{y → InterpolatingFunction[{{0., 6.28319}, {0., 13.}}, <>]}}
```

Tomando "instantáneas" cada segundo tenemos la "película":

```
In[2]:= Table[ Plot[Evaluate[y[x, t]] /. cuerda,
          {x, 0, 2 π}, PlotRange -> {-4, 4}], {t, 0, 13, 1}];
```

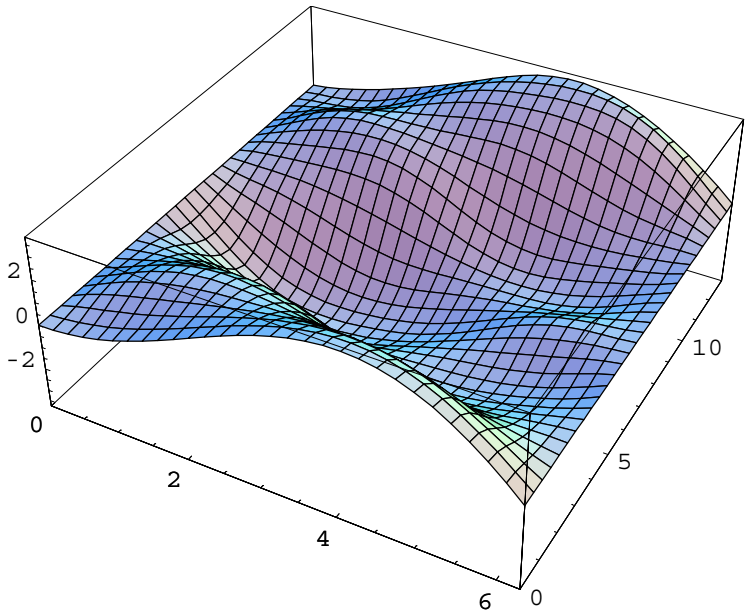



4
3
2
1
-1
-2
-3
-4
1 2 3 4 5 6



También podemos representar la evolución de la cuerda como un gráfico tridimensional {x,t,y}

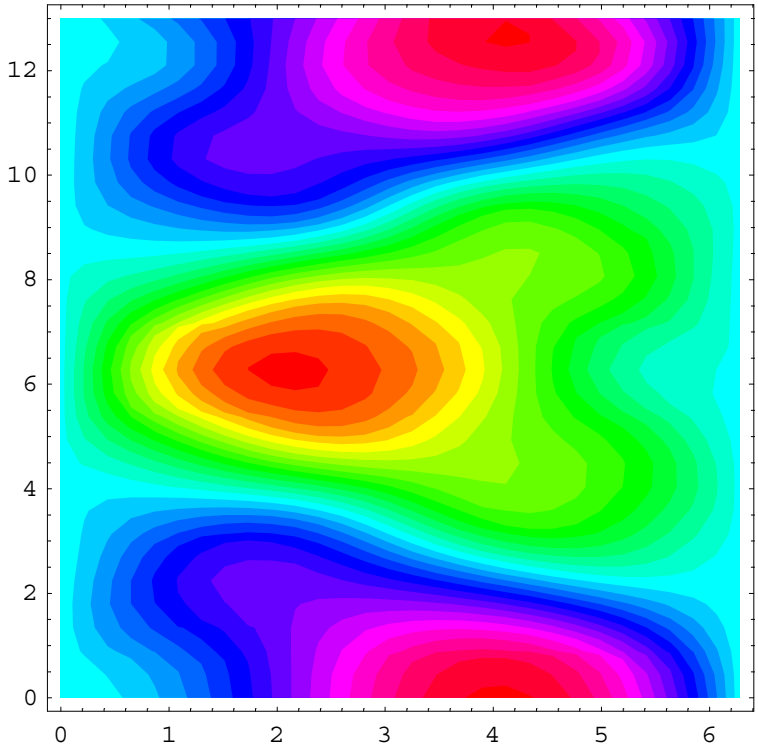
In[3]:=
```
Plot3D[Evaluate[y[x, t] /. First[cuerda]],
 {x, 0, 2 π}, {t, 0, 13}, PlotPoints → 30]
```



Out[3]= - SurfaceGraphics -

o como un gráfico de curvas de nivel:

In[4]:=
```
ContourPlot[Evaluate[y[x, t] /. First[cuerda]],
 {x, 0, 2 Pi}, {t, 0, 13}, PlotPoints → 30,
 Contours → 30, ContourLines → False, ColorFunction -> Hue]
```



Out[4]= - ContourGraphics -

donde las zonas más rojas se corresponden con puntos de mayor amplitud y las azules con puntos de menor amplitud.

Ejercicio

Repita los pasos anteriores para una cuerda de longitud $L=2$, con perfil inicial $y(x, 0) = e^{-x}\,\text{sen}(\pi x/2)$ y velocidad inicial nula.

■ Difusión del calor en un vástago

La ecuación que gobierna la distribución en el tiempo de la temperatura $T(x,t)$ de un vástago de longitud L, conductividad térmica κ, capacidad calorífica C y densidad ρ es:

$\frac{\partial T(x,t)}{\partial t} = a^2 \frac{\partial^2 T(x,t)}{\partial x^2}$ con condiciones iniciales: $T(x, 0) = T_0(x)$, y de contorno:

1) $T(0, t) = T_1(t)$, $T(L, t) = T_2(t)$ para extremos en contacto con termostatos a temperaturas T_1 y T_2, y

2) $\partial_x T(0, t) = \partial_x T(L, t) = 0$ para extremos aislados.

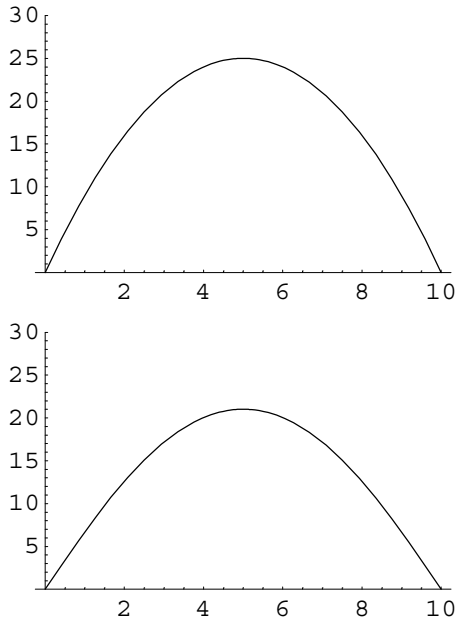
donde $a = \sqrt{\frac{\kappa}{C\rho}}$.

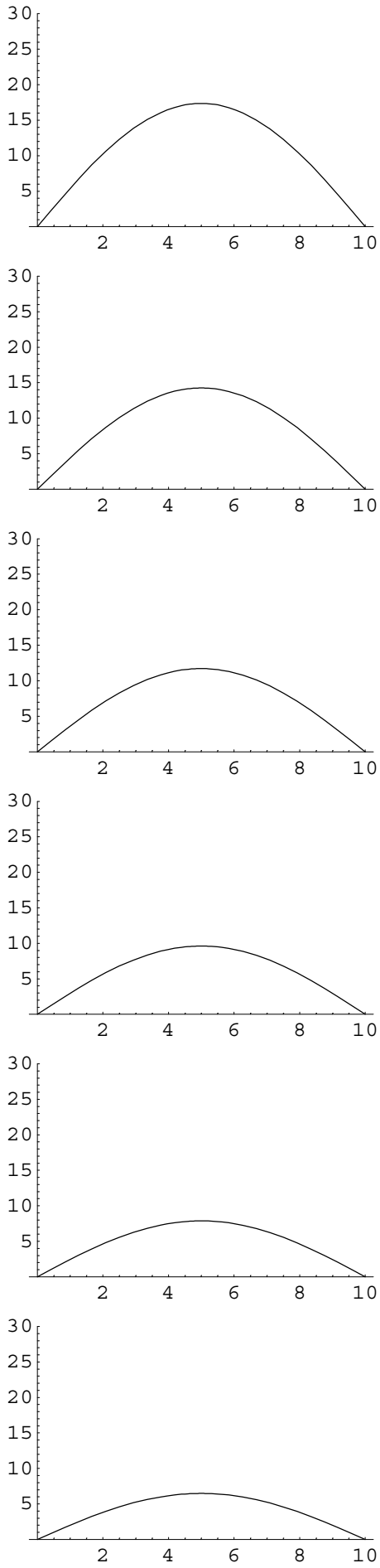
Tomemos por ejemplo $a=1$, $L=10$ y las condiciones iniciales $T(x,0)=-x(x-10)$, $T_1 = T_2 = 0$. La solución numérica de la ecuación de difusión entre t=0 y t=21 segundos viene dada por:

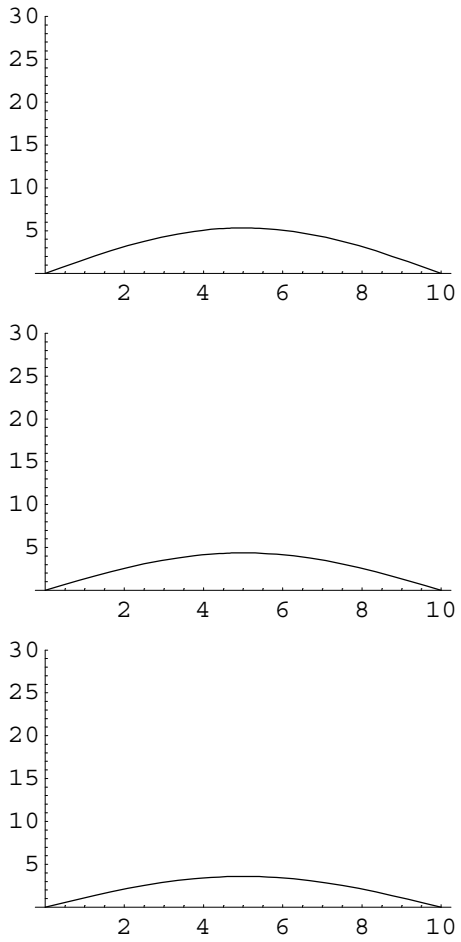
```
In[5]:= temp = NDSolve[{D[T[x, t], t] == D[T[x, t], x, x], T[x, 0] == -x (x - 10),
          T[0, t] == 0, T[10, t] == 0}, T, {x, 0, 10}, {t, 0, 21}]
Out[5]= {{T → InterpolatingFunction[{{0., 10.}, {0., 21.}}, <>]}}
```

Para ver cómo varía la distribución de temperaturas en el tiempo podemos representar varias en una tabla

```
In[6]:= Table[ Plot[Evaluate[T[x, t]] /. temp,
          {x, 0, 10}, PlotRange -> {0, 30}], {t, 0, 21, 2}];
```

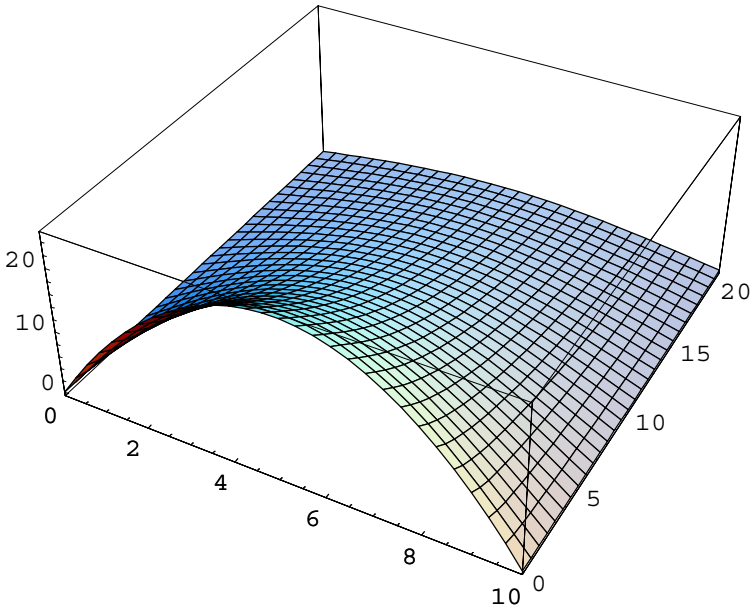



30
25
20
15
10
5
2
4
6
8
10



o bien en un gráfico bidimensional

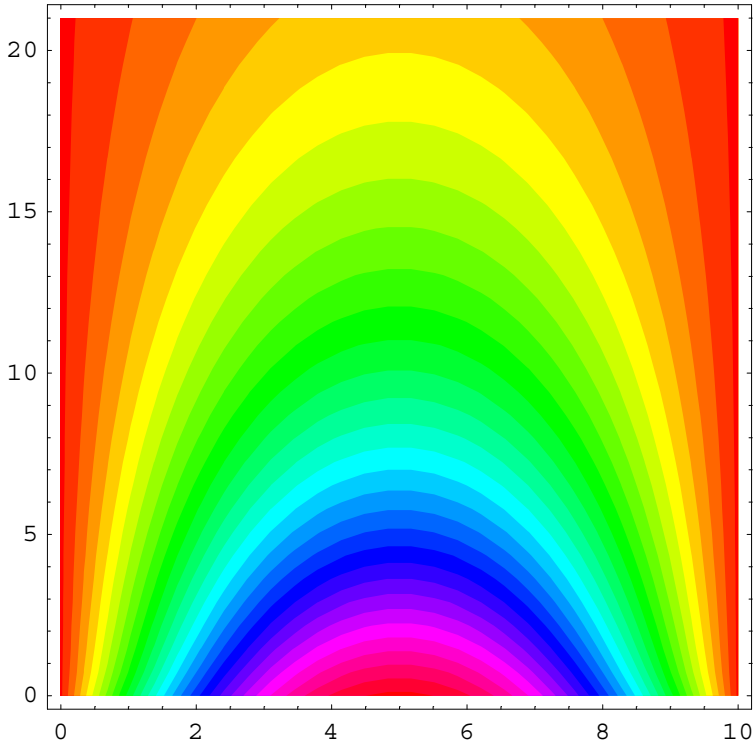
```
In[7]:= Plot3D[Evaluate[T[x, t] /. First[temp]],
          {x, 0, 10}, {t, 0, 21}, PlotPoints → 30]
```




```
Out[7]= - SurfaceGraphics -
```

o mediante curvas de nivel

```
In[8]:= ContourPlot[Evaluate[T[x, t] /. First[temp]],
          {x, 0, 10}, {t, 0, 21}, PlotPoints → 30,
          Contours → 30, ContourLines → False, ColorFunction -> Hue]
```




```
Out[8]= - ContourGraphics -
```

donde las zonas rojas (extremos del vástago) indican baja temperatura y las azules y violetas (centro del vástago) indican alta temperatura. De cualquiera de las tres formas vemos que la temperatura tiende paulatinamente a cero en todos los puntos del vástago.

Ejercicio

Repita el ejercicio anterior para un vástago de longitud $L=20$, en contacto con dos termostatos a temperaturas $T_1 = 0$, $T_2 = 8$ y condición inicial $T(x, 0) = x^3/1000$

Bibliografía

- S. Wolfram, "The Mathematica book", Ed. Wolfram Media & Cambridge University Press.

- R. Gass, "Mathematica for Scientists and Engineers", Prentice-Hall (1998).

- G. Aguilar / A. Fernández, "Ecuaciones Diferenciales: prácticas con *Mathematica"*, Prensas Universitarias de Zaragoza (1997).