

# Computación Científica y Técnica con Maple

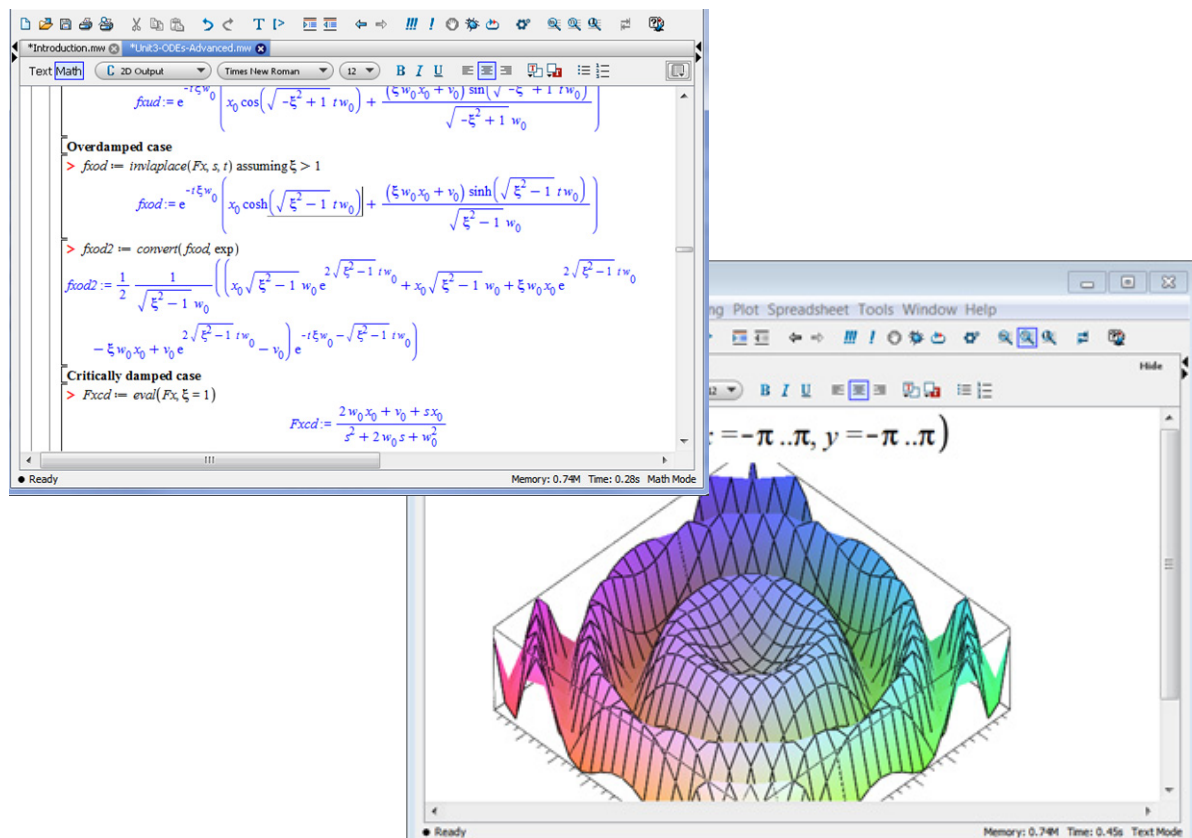
## Uso básico de Maple

Version 1.2, Febrero 2012

Francisco Palacios Quiñonero

Departamento Matemática Aplicada III

Universidad Politécnica de Cataluña (UPC)





**Computación Científica y Técnica con Maple: Uso Básico de Maple. Versión 1.2. Febrero 2012** by Francisco Palacios-Quíñonero is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

#### You are free:

to **Share** — to copy, distribute and transmit the work

#### Under the following conditions:



**Attribution** — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



**Noncommercial** — You may not use this work for commercial purposes.



**No Derivative Works** — You may not alter, transform, or build upon this work.

## Computación científica y técnica con Maple

# Unidad 1: Uso básico de Maple

F. Palacios Quiñonero

Dept. Matemática Aplicada III. Universidad Politécnica de Cataluña

Ver. 1.2 Febrero, 2012

Este documento se ha editado usando las hojas de trabajo (worksheet) de Maple 15

### Contenido

1. [Línea de comandos, nombres, ayuda](#)
2. [Cálculo simbólico](#)
3. [Cálculo exacto](#)
4. [Cálculo aproximado](#)
5. [Evaluación de expresiones](#)
6. [Menús de contexto](#)
7. [Gráficos](#)
8. [Derivadas](#)
9. [Integrales](#)
10. [Ecuaciones](#)
11. [Ecuaciones Diferenciales Ordinarias \(EDOs\)](#)
12. [Resolución simbólica de EDOs](#)
13. [Resolución gráfica de EDOs: campos de pendientes](#)
14. [Resolución numérica de EDOs](#)
15. [Maple vs Matlab](#)

## 1. Línea de comandos, nombres, ayuda

[Volver a la tabla de contenidos](#)

[Para ejecutar la línea de comandos, presionamos [INTRO]

&gt; 1 + 1

2

[Para asignar un nombre a un objeto, usamos :=]

&gt; a := 1

a := 1

&gt; b := 3

b := 3

&gt; c := a + b

c := 4

[La combinación de teclas [Ctrl]+[t] crea de forma rápida una celda de texto (como esta)]

[Maple distingue entre mayúsculas y minúsculas]

&gt; A := 0.5

A := 0.5

```
> a := 1
```

 $a := 1$ 

```
> a + A
```

 $1.5$ 

Para suprimir la presentación de resultados, terminamos la línea de comandos con dos puntos (:)

```
> c := a + A :
```

```
> c
```

 $1.5$ 

Maple puede manejar nombres no asignados, en ese caso se consideran objetos algebraicos

```
> d
```

 $d$ 

```
> 1 + d
```

 $1 + d$ 

```
> e := (1 + d)^2
```

 $e := (1 + d)^2$ 

Podemos añadir comentarios a la línea de comandos con el carácter #

```
> expand(e) # el comando expand() fuerza la evaluación de los productos
```

 $1 + 2d + d^2$ 

Si escribimos una expresión entre comillas simples, se evita su evaluación

```
> c := 'A + a'
```

 $c := a + A$ 

```
> c
```

 $1.5$ 

La evaluación de un nombre no asignado, devuelve el propio nombre

```
> a
```

 $1$ 

```
> f
```

 $f$ 

Para limpiar un nombre, escribimos **nombre:='nombre'**, por ejemplo

```
> c
```

 $1.5$ 

```
> c := 'c';
```

 $c := c$ 

```
> c
```

 $c$ 

El comando **restart** limpia todos los nombres

```
> b
```

 $3$ 

```
> e
```

 $(1 + d)^2$ 

```
> restart
```

```
> b
```

> e

$b$

$e$

Podemos usar nombres de letras griegas como nombres de objetos, Maple los presenta usando los símbolos del alfabeto griego. Observa que Maple interpreta un espacio blanco como una multiplicación implícita.

> alpha := 3;

$\alpha := 3$

> c := A cos(phi t)

$c := A \cos(\phi t)$

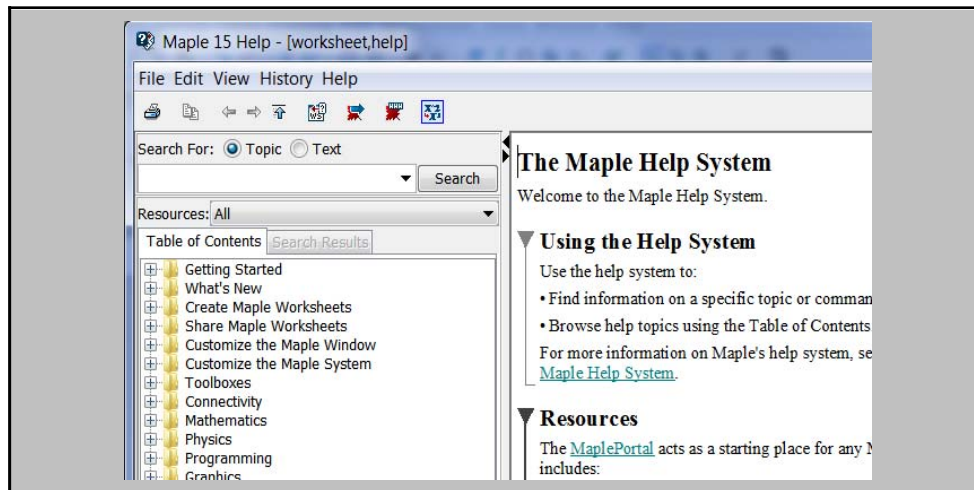
También podemos insertar letras griegas desde la Paleta Greek



> d := A sin(omega t + phi)

$d := A \sin(\omega t + \phi)$

**Ayuda.** Podemos acceder al sistema de ayuda presionando [Ctrl]+[F1]



**Grupos de ejecución.** Es posible escribir varios comandos en la misma línea. En ese caso, tenemos que separar los diferentes comandos usando un punto y coma (;).

> 1 + 1; a := 3; expand((b + a)^2)

$2$

$a := 3$

$b^2 + 6b + 9$

Para obtener una presentación más clara y estructurada, podemos romper la línea de comandos (sin ejecutar el código) pulsando la combinación de teclas **[SHIFT]+[INTRO]**.

Para entrar los subíndices podemos usar las notaciones:  $e[0]$  ó  $e\_0$

```
> e_0 := 14;
    v[0] := 2;
    a := -0.5;
    t := 3;
    e_f := e_0 + v_0 * t + 1/2 * a * t^2
```

```
e_0 := 14
v_0 := 2
a := -0.5
t := 3
e_f := 17.75000000
```

## 2. Cálculo simbólico

[Volver a la tabla de contenidos](#)

Maple usa los nombres "limpios" (no asignados) como objetos algebraicos

```
> x := 1; y := 2
```

```
x := 1
y := 2
```

```
> x + y
```

```
3
```

```
> x := 'x'; y := 'y'
```

```
x := x
y := y
```

```
> p := (x + y)^3
```

```
p := (x + y)^3
```

El comando **expand()** fuerza la evaluación de todos los productos

```
> expand(p)
```

```
x^3 + 3 x^2 y + 3 x y^2 + y^3
```

El comando **factor()** factorizar expresiones algebraicas, y el comando **simplify()** realiza simplificaciones

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

```
q := x^2 + 3 x + 2
```

```
> factor(q)
```

```
(x + 2) (x + 1)
```

```
> r := (x^2 - 4) / (x^3 - 6 x^2 + 11 x - 6)
```

```
r := (x^2 - 4) / (x^3 - 6 x^2 + 11 x - 6)
```

> *simplify(r)*

$$\frac{x+2}{x^2-4x+3}$$

> *factor(r)*

$$\frac{x+2}{(x-1)(x-3)}$$

Los comandos **numer()** y **denom()** extraen el numerador y el denominador de una fracción.

Para ver más claramente la simplificación realizada sobre la función racional *r*, podemos factorizar por separado el numerador y el denominador

> *n := numer(r)*

$$n := x^2 - 4$$

> *d := denom(r)*

$$d := x^3 - 6x^2 + 11x - 6$$

> *factor(n)*

$$(x-2)(x+2)$$

> *factor(d)*

$$(x-1)(x-2)(x-3)$$

El comando **simplify()** también funciona con expresiones trigonométricas

> *a := sin(phi);*

*b := cos(phi);*

*c := a^2 + b^2;*

$$a := \sin(\phi)$$

$$b := \cos(\phi)$$

$$c := \sin(\phi)^2 + \cos(\phi)^2$$

> *c1 := simplify(c)*

$$c1 := 1$$

El comando **convert(expression, parfrac)** calcula la descomposición en fracciones simples de una función racional (cociente de polinomios)

> *r*

$$\frac{x^2-4}{x^3-6x^2+11x-6}$$

> *rp := convert(r, parfrac)*

$$rp := \frac{5}{2(x-3)} - \frac{3}{2(x-1)}$$

En el siguiente ejemplo, obtenemos la descomposición en fracciones simples de una función racional impropia en grado (grado del numerador  $\geq$  grado del denominador)

> *r1 := x^2 r*

$$r1 := \frac{x^2(x^2-4)}{x^3-6x^2+11x-6}$$

> *r1p := convert(r1, parfrac)*

$$rlp := x + 6 + \frac{45}{2(x-3)} - \frac{3}{2(x-1)}$$

### 3. Cálculo exacto

[Volver a la tabla de contenidos](#)

Maple opera los números exactos sin redondear

$$> a := 3^{\frac{1}{2}}$$

$$a := \sqrt{3}$$

$$> b := (1 + a)^2$$

$$b := (1 + \sqrt{3})^2$$

$$> \text{expand}(b)$$

$$4 + 2\sqrt{3}$$

Los números enteros son números exactos, Maple puede manipular números enteros de tamaño arbitrario (dentro de las posibilidades del ordenador)

$$> p := 23^{25} + 2^{36} + 3^{21}$$

$$p := 11045767571919545466173891589519882$$

El comando **ifactor()** obtiene la descomposición en factores primos

$$> \text{ifactor}(5525432216)$$

$$(2)^3 (61) (349) (32443)$$

$$> \text{ifactor}(p)$$

$$(2) (3) (15532109237) (314098893699477827) (377353)$$

Maple simplifica fracciones automáticamente

$$> a := \frac{12}{32}$$

$$a := \frac{3}{8}$$

también opera fracciones de forma exacta, reduciendo a común denominador

$$> b := \frac{2}{3} + a$$

$$b := \frac{25}{24}$$

La evaluación de una función para un argumento exacto, da como resultado un número exacto

$$> a := \sin\left(\frac{\text{Pi}}{4}\right)$$

$$a := \frac{1}{2} \sqrt{2}$$

$$> b := \tan\left(\frac{\text{Pi}}{3}\right)$$

$$b := \sqrt{3}$$

$$> c := (a + b)^2$$



$$c := \left( \sqrt{3} + \frac{1}{2} \sqrt{2} \right)^2$$

> `expand(c)`

$$\frac{7}{2} + \sqrt{3} \sqrt{2}$$

Un ejemplo trigonométrico

> `a := sin(3);`  
`b := cos(3);`  
`c := (a + b)^2;`  
`c1 := expand(c);`  
`simplify(c1);`

$$a := \sin(3)$$

$$b := \cos(3)$$

$$c := (\cos(3) + \sin(3))^2$$

$$c1 := \cos(3)^2 + 2 \cos(3) \sin(3) + \sin(3)^2$$

$$2 \cos(3) \sin(3) + 1$$

El número  $\pi$  se escribe **Pi**

> `cos( Pi / 4 )`

$$\frac{1}{2} \sqrt{2}$$

**pi** es la letra griega  $\pi$ . ¡ No es un número!

> `cos( pi / 4 )`

$$\cos\left(\frac{1}{4} \pi\right)$$

**!!!** No debemos confundir el número **Pi** con la letra del alfabeto griego **pi**

## 4. Cálculo aproximado

[Volver a la tabla de contenidos](#)

Los números de punto flotante (floats) permiten aproximar el valor de números exactos usando una cantidad limitada de dígitos

> `a := 12.34 / 4.56`

$$a := 2.706140351$$

Operaciones aritméticas de números exactos y aproximados suelen producir resultados aproximados

> `b := a + 2 / 7`

$$b := 2.991854637$$

El comando **evalf()** proporciona aproximaciones float de números exactos

```
> a := sqrt(2)
```

$$a := \sqrt{2}$$

```
> af := evalf(a)
```

$$af := 1.414213562$$

En cálculos que contienen números exactos y aproximados, algunas veces es necesario usar el comando **evalf()** para obtener una evaluación float completa

```
> a := 7/5 + sqrt(2) + 1.23
```

$$a := 2.630000000 + \sqrt{2}$$

```
> af := evalf(a)
```

$$af := 4.044213562$$

Funciones aplicadas sobre números aproximados producen resultados aproximados

```
> b := exp(2)
```

$$b := e^2$$

```
> bf := evalf(b)
```

$$bf := 7.389056099$$

```
> exp(2.)
```

$$7.389056099$$

```
> cos(0.2)
```

$$0.9800665778$$

Por defecto, Maple realiza evaluaciones float con 10 dígitos.

El comando **evalf(valor exacto,n)** calcula una evaluación float con n dígitos

```
> evalf(Pi)
```

$$3.141592654$$

```
> evalf(Pi, 150)
```

$$3.1415926535897932384626433832795028841971693993751058209749445923078164062862089\backslash 9862803482534211706798214808651328230664709384460955058223172535940813$$

La variable de sistema **Digits** establece el número de dígitos empleado por Maple para presentar y calcular los números aproximados.

- Por defecto, el valor de **Digits** es 10.
- Podemos reasignar el valor de la variable **Digits**.

```
> cos(2.)
```

$$-0.4161468365$$

```
> Digits := 20
```

$$Digits := 20$$

```
> cos(2.)
```

$$-0.41614683654714238700$$

**!!!** Todas las operaciones con floats se realizan usando el número de dígitos indicado en la variable de sistema **Digits**.

Observa que **Digits** empieza por mayúscula, la asignación **digits:=16** no produce ningún efecto en el número de dígitos

!!! El comando **restart**, además de borrar todas las asignaciones de nombres, reestablece la variable de sistema **Digits** a su valor por defecto (10)

```
> restart
> Digits
10
> cos(2.)
-0.4161468365
```

## 5. Evaluación de expresiones

[Volver a la tabla de contenidos](#)

[El comando **eval(expression, var=valor)** permite evaluar expresiones.

!!! Observa que para indicar el valor a sustituir se usa el signo = (no se trata de una asignación)

```
> restart
> f := x^2 + cos(x)
f := x^2 + cos(x)
> vf := eval(f, x=2)
vf := 4 + cos(2)
```

La evaluación con floats suele producir resultados float

```
> vf2 := eval(f, x=3.)
vf2 := 8.010007503
```

**eval()** también admite evaluaciones simbólicas

```
> f := x^2 + a x + b
f := x^2 + a x + b
> fp := eval(f, a=phi + alpha)
fp := x^2 + (phi + alpha) x + b
```

Podemos realizar evaluaciones múltiples usando una lista de sustituciones

```
> pvals := [a=1, b=z-y, x=2]
pvals := [a=1, b=z-y, x=2]
> fp := eval(f, pvals)
fp := 6 + z - y
```

[Observa que los nombres *a*, *b*, *x* permanecen sin asignar.

!!! Es muy importante mantener sin asignar los nombres usados como variables simbólicas

```
> a
a
> b
b
```

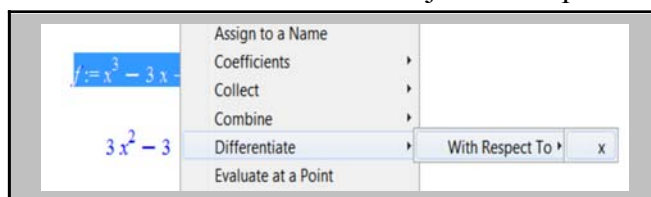
> x

x

## 6. Menús de contexto

[Volver a la tabla de contenidos](#)

Los comandos de uso más frecuente pueden ejecutarse desde *Menús de Contexto* que se activan haciendo click con el botón derecho del ratón sobre el objeto correspondiente



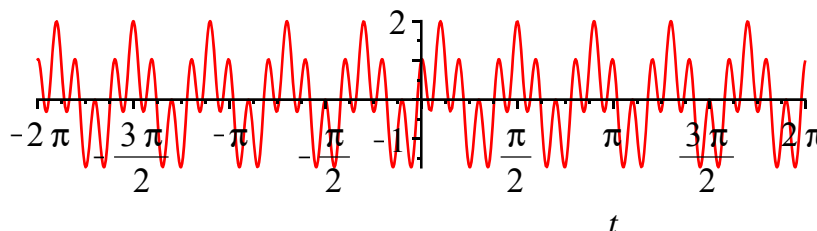
En las *hojas de trabajo* (worksheet), cuando aplicamos un comando desde un menú de contexto, Maple escribe el comando y lo ejecuta. En el *modo documento*, Maple sólo muestra el resultado.

> f := sin(5 t) + cos(20 t)

f := sin(5 t) + cos(20 t)

El siguiente gráfico se ha obtenido pulsando el boton derecho sobre la expresión anterior y seleccionando **Plots > 2D**

> smartplot( sin(5 \* t) + cos(20 \* t) )



> f

sin(5 t) + cos(20 t)

La siguiente derivada se ha calculado pulsando el boton derecho sobre la expresión anterior y seleccionando **Differentiate > With Respect To > t**

> diff( sin(5 \* t) + cos(20 \* t), t)

5 cos(5 t) - 20 sin(20 t)

!!! Podemos dibujar un gráfico simplemente seleccionando una expresión y arrastrándola sobre un **smartplot**

## 7. Gráficos

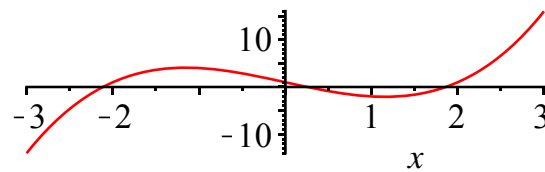
[Volver a la tabla de contenidos](#)

El comando **plot(expresion, variable=Vmin..Vmax )** dibuja gráficos 2D. El intervalo [a,b] se representa por **a..b**

> f := 1 - 4 x + x<sup>3</sup>

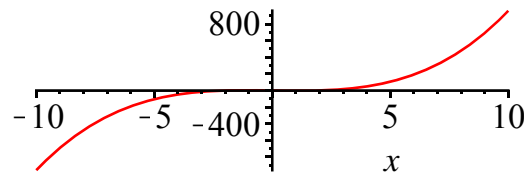
f := 1 - 4 x + x<sup>3</sup>

> plot(f, x = -3 .. 3)



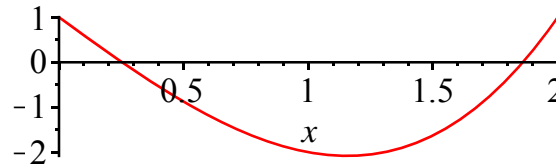
Si no iniciamos nada, el intervalo por omisión es -10..10

> `plot(f)`



Para ver con mayor detalle el gráfico en el intervalo [0..2], basta con especificar el intervalo

> `plot(f, x=0..2)`

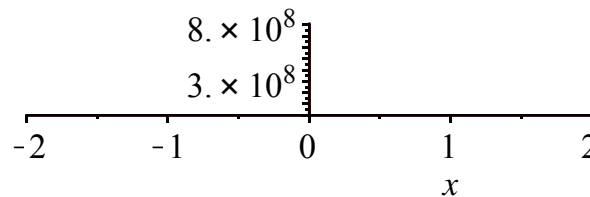


A veces es necesario restringir el intervalo de representación para el eje vertical

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

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

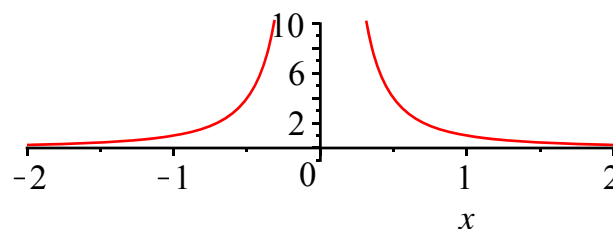
> `plot(g, x=-2..2)`



El gráfico anterior no se representa adecuadamente debido a la asíntota vertical en  $x=0$ .

Restringimos el intervalo de representación para el eje vertical especificando un segundo intervalo en el comando **plot()**

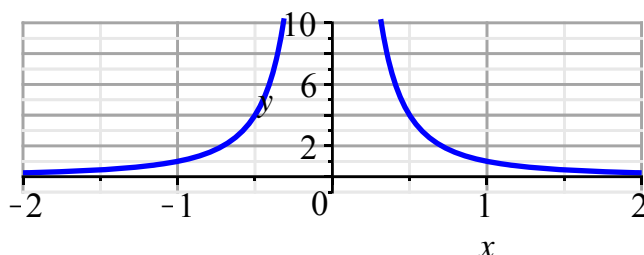
> `plot(g, x=-2..2, -1..10)`



Algunas opciones interesantes del comando **plot()** son:

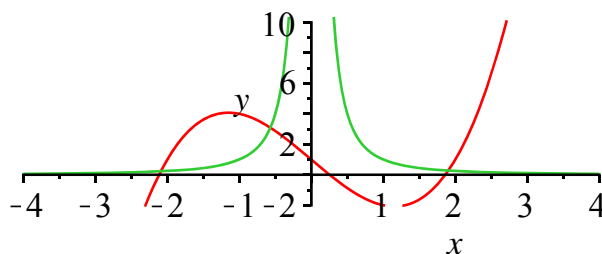
- **gridlines**, dibuja la cuadrícula
- **thickness**, controla el grosor de la línea
- **color**, controla el color de línea

> `plot(g, x=-2..2, y=-1..10, thickness=2, color=blue, gridlines)`



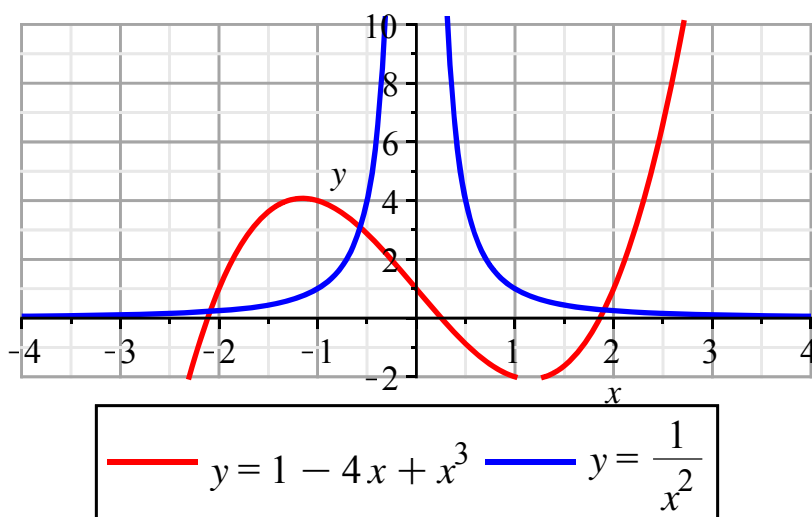
Para dibujar gráficos múltiples, pasamos al comando **plot()** una lista de expresiones **[f1,f2,...]**

> `plot([f, g], x=-4..4, y=-2..10)`



En el siguiente gráfico, hemos asignado colores de forma ordenada a las curvas con la opción **color** = **[red, blue]**, también hemos modificado el grosor, añadido la cuadrícula, e incluido la leyenda.

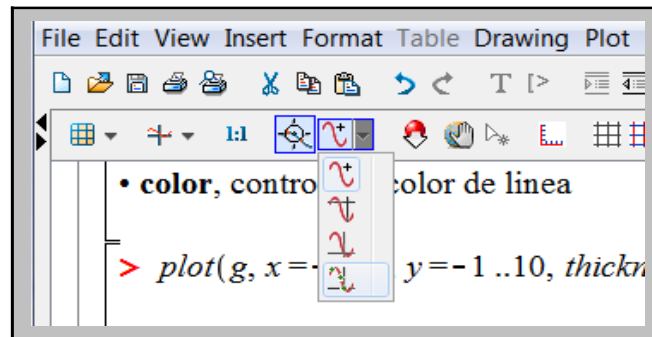
> `plot([f, g], x=-4..4, y=-2..10, color=[red, blue], thickness=2, gridlines, legend=[y=f, y=g])`



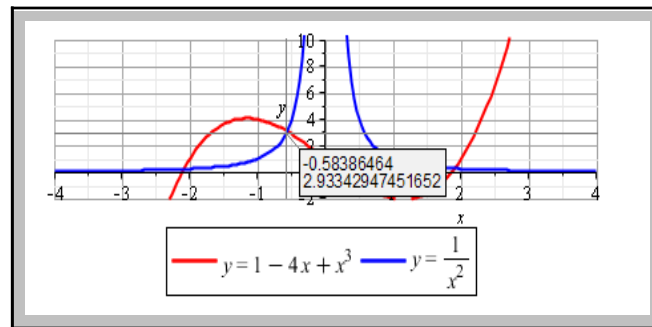
>

Haciendo click con el botón izquierdo del ratón sobre un gráfico se activa la barra de herramientas

de gráficos



Esta barra de herramientas permite modificar el estilo de ejes, dibujar la cuadrícula, etc. También podemos activar la función de lectura de datos mediante el cursor (probe). Si seleccionamos la última opción del menú desplegable, al posicionar el cursor sobre el gráfico obtendremos las coordenadas del punto más próximo al cursor



## 8. Derivadas

[Volver a la tabla de contenidos](#)

El comando **diff(f,x)** calcula la derivada de **f** con respecto de **x**.

>  $f := x \sin(x)$

$$f := x \sin(x)$$

>  $df := \text{diff}(f, x);$

$$df := \sin(x) + x \cos(x)$$

Para calcular la segunda derivada usamos **diff(f,x,x)**.

>  $df2 := \text{diff}(f, x, x)$

$$df2 := 2 \cos(x) - x \sin(x)$$

Para calcular derivadas de orden superior, podemos utilizar el operador \$

>  $x\$4$

$$x, x, x, x$$

Derivada cuarta

>  $df4 := \text{diff}(f, x\$4)$

$$df4 := -4 \cos(x) + x \sin(x)$$

En algunos caso, Maple puede incluso realizar un cálculo inductivo de la n-ésima derivada.

Primero, limpiamos el nombre  $n$  para asegurarnos que puede actuar como variable simbólica

>  $n := 'n'$

$$n := n$$

```
> g := x exp(x)
```

$$g := x e^x$$

```
> dfn := diff(g, x$n)
```

$$dfn := e^x (x + n)$$

Veamos otro ejemplo, esta vez para una expresión que contiene parámetros indeterminados

```
> h := exp(-alpha t) sin(omega t)
```

$$h := e^{-\alpha t} \sin(\omega t)$$

Derivada primera

```
> h1 := diff(h, t)
```

$$h1 := -\alpha e^{-\alpha t} \sin(\omega t) + e^{-\alpha t} \cos(\omega t) \omega$$

Derivada segunda

```
> h2 := diff(h, t, t)
```

$$h2 := \alpha^2 e^{-\alpha t} \sin(\omega t) - 2 \alpha e^{-\alpha t} \cos(\omega t) \omega - e^{-\alpha t} \sin(\omega t) \omega^2$$

Derivada cuarta

```
> h4 := diff(h, t$4)
```

$$h4 := \alpha^4 e^{-\alpha t} \sin(\omega t) - 4 \alpha^3 e^{-\alpha t} \cos(\omega t) \omega - 6 \alpha^2 e^{-\alpha t} \sin(\omega t) \omega^2 + 4 \alpha e^{-\alpha t} \cos(\omega t) \omega^3 + e^{-\alpha t} \sin(\omega t) \omega^4$$

El comando **collect( )** extrae un término especificado como factor común

```
> collect(h4, exp(-alpha t))
```

$$(\alpha^4 \sin(\omega t) - 4 \alpha^3 \cos(\omega t) \omega - 6 \alpha^2 \sin(\omega t) \omega^2 + 4 \alpha \cos(\omega t) \omega^3 + \sin(\omega t) \omega^4) e^{-\alpha t}$$

Para establecer una secuencia ordenada de términos que han de extraerse como factor común, podemos especificar una lista de expresiones en el comando **collect( )**. Así, la lista

$$[\exp(-\alpha t), \sin(\omega t), \cos(\omega t)]$$

indica que primero ha de extraerse el factor  $e^{-\alpha t}$ , seguidamente el factor  $\sin(\omega t)$ , y finalmente el factor  $\cos(\omega t)$ .

```
> collect(h4, [exp(-alpha t), sin(omega t), cos(omega t)])
```

$$((\alpha^4 - 6 \alpha^2 \omega^2 + \omega^4) \sin(\omega t) + (-4 \alpha^3 \omega + 4 \alpha \omega^3) \cos(\omega t)) e^{-\alpha t}$$

Veamos el resultado con la derivada 10ª

```
> h10 := diff(h, t$10)
```

$$h10 := \alpha^{10} e^{-\alpha t} \sin(\omega t) - 10 \alpha^9 e^{-\alpha t} \cos(\omega t) \omega - 45 \alpha^8 e^{-\alpha t} \sin(\omega t) \omega^2 + 120 \alpha^7 e^{-\alpha t} \cos(\omega t) \omega^3 + 210 \alpha^6 e^{-\alpha t} \sin(\omega t) \omega^4 - 252 \alpha^5 e^{-\alpha t} \cos(\omega t) \omega^5 - 210 \alpha^4 e^{-\alpha t} \sin(\omega t) \omega^6 + 120 \alpha^3 e^{-\alpha t} \cos(\omega t) \omega^7 + 45 \alpha^2 e^{-\alpha t} \sin(\omega t) \omega^8 - 10 \alpha e^{-\alpha t} \cos(\omega t) \omega^9 - e^{-\alpha t} \sin(\omega t) \omega^{10}$$

```
> collect(h10, [exp(-alpha t), sin(omega t), cos(omega t)])
```

$$((\alpha^{10} - 45 \alpha^8 \omega^2 + 210 \alpha^6 \omega^4 - 210 \alpha^4 \omega^6 + 45 \alpha^2 \omega^8 - \omega^{10}) \sin(\omega t) + (-10 \alpha^9 \omega + 120 \alpha^7 \omega^3 - 252 \alpha^5 \omega^5 + 120 \alpha^3 \omega^7 - 10 \alpha \omega^9) \cos(\omega t)) e^{-\alpha t}$$

**Notación prima para derivadas.** Las derivadas con respecto de la variable independiente  $x$  pueden



calcularse usando la *notación prima*

>  $f := x \cdot \sin(x)$

$f := x \sin(x)$

>  $f'$

$\sin(x) + x \cos(x)$

>  $f''$

$-3 \sin(x) - x \cos(x)$

**Notación punto para derivadas.** Las derivadas con respecto de la variable independiente  $t$  pueden calcularse usando la notación punto. Presionar **[Ctrl]+[Shift]+[']** para acceso rápido a la notación punto

!!! La combinación de teclas **[Ctrl]+[Shift]+[']** proporciona acceso rápido a la *notación punto para derivadas*. En la notación punto, la variable independiente es  $t$

>  $y := \frac{\sin(t)}{t}$

$y := \frac{\sin(t)}{t}$

>  $y1 := \dot{y}$

$y1 := \frac{\cos(t)}{t} - \frac{\sin(t)}{t^2}$

>  $y2 := \ddot{y}$

$y2 := -\frac{\sin(t)}{t} - \frac{2 \cos(t)}{t^2} + \frac{2 \sin(t)}{t^3}$

Es aconsejable limpiar la variable  $y$ , ya que esta variable suele usarse frecuentemente como variable simbólica

>  $y := 'y'$

$y := y$

## 9. Integrales

[Volver a la tabla de contenidos](#)

El comando **int(f,variable)** calcula la primitiva de  $f$  con respecto de la variable indicada.

Nos aseguramos que la variable  $x$  está limpia

>  $x := 'x'$

$x := x$

>  $f := x \sin(x)$

$f := x \sin(x)$

>  $F := \text{int}(f, x)$

$F := \sin(x) - x \cos(x)$

Para calcular integrales definidas, basta con indicar el intervalo de integración **int(f,x=a..b)**.

>  $f := x^2$

$f := x^2$

>  $v := \text{int}(f, x=0..1)$

$$v := \frac{1}{3}$$

El comando **int()** también admite límites de integración simbólicos

>  $a := 'a'; b := 'b';$

$$a := a$$

$$b := b$$

>  $v := \text{int}(f, x=a..b)$

$$v := \frac{1}{3} b^3 - \frac{1}{3} a^3$$

Si Maple no puede calcular el valor exacto, la integral se devuelve sin evaluar. En ese caso, podemos obtener una aproximación float con **evalf()**, para ello Maple aplica automáticamente un método de cálculo numérico.

>  $f := \sin(x^6 + x)$

$$f := \sin(x^6 + x)$$

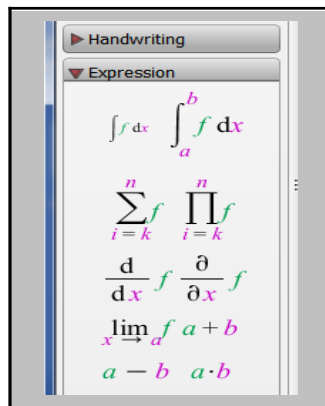
>  $v := \text{int}(f, x=0..1)$

$$v := \int_0^1 \sin(x^6 + x) dx$$

>  $vf := \text{evalf}(v)$

$$vf := 0.5163377908$$

Para entrar derivadas, integrales, límites, ..., de forma cómoda y en notación matemática natural, podemos usar la **Paleta Expression**



>  $\int_0^{\frac{\pi}{2}} \sin(ax) \cos(bx) dx$

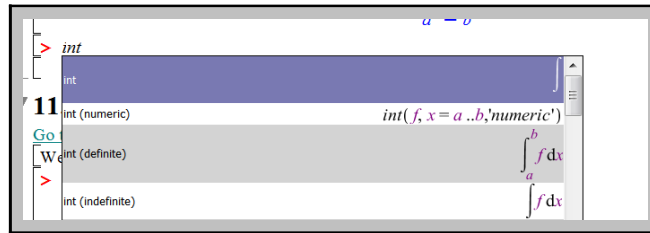
$$-\frac{a \cos\left(\frac{1}{2} \pi a\right) \cos\left(\frac{1}{2} \pi b\right) - a + b \sin\left(\frac{1}{2} \pi a\right) \sin\left(\frac{1}{2} \pi b\right)}{a^2 - b^2}$$

>

>

También podemos acceder de forma rápida a las plantillas de expresiones escribiendo el comando

correspondiente y presionando [ctrl]+[barra espaciadora]



Maple también calcula integrales impropias de forma exacta. La opción **assuming** permite establecer propiedades de los parámetros, en nuestro caso establecemos que el parámetro  $\alpha$  es positivo.

>  $\int_1^{\infty} x^2 e^{-\alpha x} dx$  assuming alpha > 0

$$\frac{2}{\alpha^3 e^{\alpha}} + \frac{2}{\alpha^2 e^{\alpha}} + \frac{1}{\alpha e^{\alpha}}$$

La integral impropia es divergente para  $\alpha < 0$

>  $\int_1^{\infty} x^2 e^{-\alpha x} dx$  assuming alpha < 0

$\infty$

El nombre Maple de  $\infty$  es **infinity**

>  $\frac{1}{\text{infinity}}$

0

>  $\lim \left( \frac{a x}{x + 1}, x = \text{infinity} \right)$

a

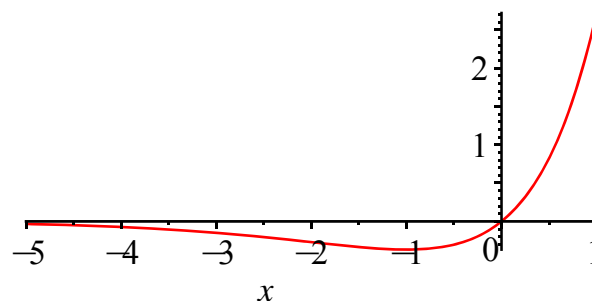
>  $\lim_{x \rightarrow -\infty} x e^x$

0

>  $\lim_{x \rightarrow +\infty} x e^x$

$\infty$

>  $\text{plot}(x e^x, x = -5 .. 1)$



## 10. Ecuaciones

[Volver a la tabla de contenidos](#)

El comando **solve()** resuelve ecuaciones de formas exacta y simbólica

>  $eq := a x^2 + b x + c = 0$

$$eq := a x^2 + b x + c = 0$$

>  $solve(eq, a)$

$$-\frac{b x + c}{x^2}$$

>  $solve(eq, x)$

$$\frac{1}{2} \frac{-b + \sqrt{b^2 - 4 a c}}{a}, -\frac{1}{2} \frac{b + \sqrt{b^2 - 4 a c}}{a}$$

>  $eq2 := x^3 - x^2 + 3 = 0$

$$eq2 := x^3 - x^2 + 3 = 0$$

>  $s := solve(eq2)$

$$s := -\frac{1}{6} (316 + 36 \sqrt{77})^{1/3} - \frac{2}{3 (316 + 36 \sqrt{77})^{1/3}} + \frac{1}{3}, \frac{1}{12} (316 + 36 \sqrt{77})^{1/3} + \frac{1}{3 (316 + 36 \sqrt{77})^{1/3}} + \frac{1}{3} + \frac{1}{2} I \sqrt{3} \left( -\frac{1}{6} (316 + 36 \sqrt{77})^{1/3} + \frac{2}{3 (316 + 36 \sqrt{77})^{1/3}} \right), \frac{1}{12} (316 + 36 \sqrt{77})^{1/3} + \frac{1}{3 (316 + 36 \sqrt{77})^{1/3}} + \frac{1}{3} - \frac{1}{2} I \sqrt{3} \left( -\frac{1}{6} (316 + 36 \sqrt{77})^{1/3} + \frac{2}{3 (316 + 36 \sqrt{77})^{1/3}} \right)$$

>  $sf := evalf(s)$

$$sf := -1.174559411, 1.087279705 - 1.171312112 I, 1.087279705 + 1.171312112 I$$

El comando **fsolve()**, calcula aproximaciones numéricas de las soluciones de una ecuación.

Podemos indicar un intervalo de búsqueda a **fsolve()**

>  $f := \exp(-0.23 x) + \exp(-0.5 x)$

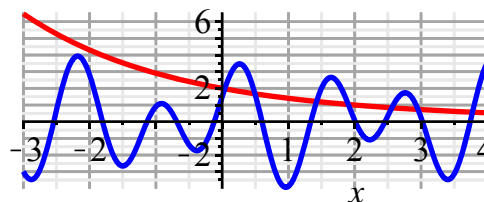
$$f := e^{-0.23 x} + e^{-0.5 x}$$

>  $g := 1.5 \cos(3 x) + 2.5 \sin(5 x)$

$$g := 1.5 \cos(3 x) + 2.5 \sin(5 x)$$

Gráfico conjunto de  $f$  y  $g$ .

>  $plot([f, g], x = -3 .. 4, color = [red, blue], gridlines, thickness = 2)$



Observamos que hay cuatro soluciones en el intervalo  $[0, 2]$ . Inicialmente, el comando **fsolve()** nos

proporciona una de ellas

```
> s1 := fsolve(f=g)
```

```
s1 := 0.03880717791
```

Para que **fsolve()** calcule la solución del intervalo [1,1.5], basta con indicarle el intervalo de búsqueda

```
> s2 := fsolve(f=g, x=1..1.5)
```

```
s2 := 1.423574342
```

Cuando se aplica a ecuaciones polinómicas, el comando **fsolve()** proporciona una aproximación de todas las soluciones reales

```
> p := x^6 - x^4 + 0.75 x^3 + 0.25 x - 1.32
```

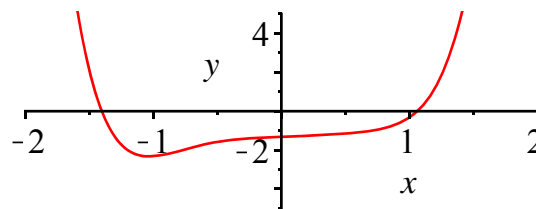
```
p := x^6 - x^4 + 0.75 x^3 + 0.25 x - 1.32
```

```
> fsolve(p=0, x)
```

```
-1.402393886, 1.060940356
```

Dibujamos un gráfico del polinomio para confirmar los valores de las raíces proporcionados por **fsolve()**

```
> plot(p, x=-2..2, y=-5..5)
```



Con la opción **complex**, **fsolve()** proporciona también las soluciones complejas

```
> fsolve(p=0, x, complex)
```

```
-1.40239388561125, -0.492204872015816 - 0.745677422358486 I, -0.492204872015816
```

```
+ 0.745677422358486 I, 0.662931636809589 - 0.819671091123570 I, 0.662931636809589
```

```
+ 0.819671091123570 I, 1.06094035602370
```

## 11. Ecuaciones diferenciales ordinarias (EDOs)

[Volver a la tabla de contenidos](#)

El comando **dsolve(edo)** resuelve ecuaciones diferenciales ordinarias

Obtención de la solución general

```
> edo := y' + 2 y = sin(x)
```

$$edo := \frac{d}{dx} y(x) + 2 y(x) = \sin(x)$$

```
> s := dsolve(edo)
```

$$s := y(x) = -\frac{1}{5} \cos(x) + \frac{2}{5} \sin(x) + e^{-2x} \_CI$$

El comando **dsolve([edo,cond])** resuelve problemas de valor inicial, **cond** representa una secuencia de condiciones iniciales (separadas por comas, cuando hay más de una)

```
> edo := x' + 2 x = t
```

$$edo := \frac{d}{dt} x(t) + 2 x(t) = t$$

```
> cond := x(0) = 2
```

$$cond := x(0) = 2$$

```
> s := dsolve([edo, cond])
```

$$s := x(t) = -\frac{1}{4} + \frac{1}{2}t + \frac{9}{4}e^{-2t}$$

El comando **odetest()** permite verificar la corrección de la solución obtenida con **dsolve()**. Una lista con dos ceros indica que la solución verifica la EDO y la condición inicial

```
> odetest(s, [edo, cond])
```

$$[0, 0]$$

Podemos extraer una expresión de la solución con el comando **eval()**

```
> fx := eval(x(t), s)
```

$$fx := -\frac{1}{4} + \frac{1}{2}t + \frac{9}{4}e^{-2t}$$

Una vez obtenida la expresión correspondiente a la solución, podemos evaluarla con **eval()** o representarla gráficamente con **plot()**

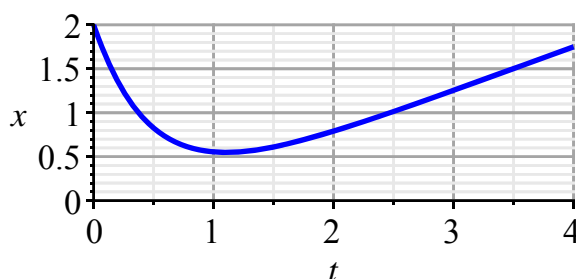
Valor de  $x(t)$  para  $t = 0.12$

```
> vx := eval(fx, t = 0.12)
```

$$vx := 1.579912687$$

Representación gráfica de  $f$  en el intervalo  $t = [0, 4]$

```
> plot(fx, t = 0 .. 4, x = 0 .. 2, gridlines, thickness = 2, color = blue)
```



### Problema de valor inicial para EDO de segundo orden

Entramos la EDO usando la *notación punto* para las derivadas. En esta caso la variable independiente es  $t$

```
> edo := x'' + 2 x' + 5 x = 0
```

$$edo := \frac{d^2}{dt^2} x(t) + 2 \left( \frac{d}{dt} x(t) \right) + 5 x(t) = 0$$

Condiciones iniciales

```
> cond := x(0) = 2, x'(0) = -1
```

$$cond := x(0) = 2, D(x)(0) = -1$$

Solución del problema de valor inicial

```
> s := dsolve([edo, cond])
```

$$s := x(t) = \frac{1}{2} e^{-t} \sin(2t) + 2 e^{-t} \cos(2t)$$

Obtenemos una expresión con la solución

```
> fx := eval(x(t), s)
```

$$fx := \frac{1}{2} e^{-t} \sin(2t) + 2 e^{-t} \cos(2t)$$

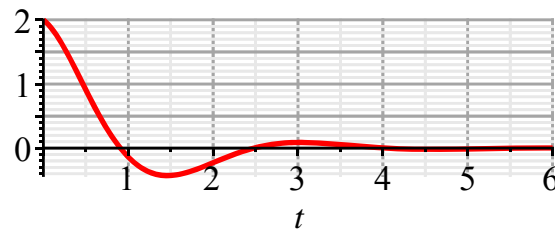
Valor de  $x(t)$  para  $t=1.6$

>  $vx := eval(fx, t=1.6)$

$$vx := -0.4089972466$$

Representación gráfica de  $x(t)$  para valores de  $t$  en  $[0,6]$

>  $plot(fx, t=0..6, gridlines, thickness=2)$



Tiempo necesario para que  $x(t)$  tome el valor 0.75

>  $t1 := fsolve(fx=0.75)$

$$t1 := 0.5618757616$$

Tiempo necesario para que la solución pase del valor  $x(t) = 1$  hasta el valor  $x(t) = 0.25$

>  $t1 := fsolve(fx=1)$

$$t1 := 0.4660615600$$

>  $t2 := fsolve(fx=0.25)$

$$t2 := 0.7747419625$$

>  $\Delta t = t2 - t1$

$$\Delta_t = 0.3086804025$$

## 12. Resolución simbólica de EDOs

[Volver a la tabla de contenidos](#)

El comando **dsolve()** puede manejar ecuaciones diferenciales y condiciones iniciales con parámetros simbólicos

>  $restart$

>  $edo := \dot{y} + a y = \cos(w t)$

$$edo := \frac{d}{dt} y(t) + a y(t) = \cos(w t)$$

>  $cond := y(0) = y_0$

$$cond := y(0) = y_0$$

>  $s := dsolve([edo, cond])$

$$s := y(t) = e^{-at} \left( y_0 - \frac{a}{a^2 + w^2} \right) + \frac{\cos(w t) a + \sin(w t) w}{a^2 + w^2}$$

Verificamos la corrección de la solución

>  $odetest(s, [edo, cond])$

$$[0, 0]$$

Extraemos una expresión para la solución

```
> fy := eval(y(t), s)
```

$$fy := e^{-at} \left( y_0 - \frac{a}{a^2 + w^2} \right) + \frac{\cos(w t) a + \sin(w t) w}{a^2 + w^2}$$

Usamos el comando **eval()** para calcular la solución correspondiente a conjunto de valores concretos de los parámetros

```
> valp := {a=2, w=10, y0=3}
```

$$valp := \{a=2, w=10, y_0=3\}$$

```
> fyp := eval(fy, valp)
```

$$fyp := \frac{155}{52} e^{-2t} + \frac{1}{52} \cos(10 t) + \frac{5}{52} \sin(10 t)$$

Obtenemos una versión float con **evalf()**

```
> fypf := evalf(fyp)
```

$$fypf := 2.980769231 e^{-2.t} + 0.01923076923 \cos(10. t) + 0.09615384615 \sin(10. t)$$

Gráfico y valor de la solución en t=1.3

```
> vy := eval(fypf, t=1.3)
```

$$vy := 0.2792439728$$

```
> plot(fypf, t=0..4, gridlines, thickness=2)
```

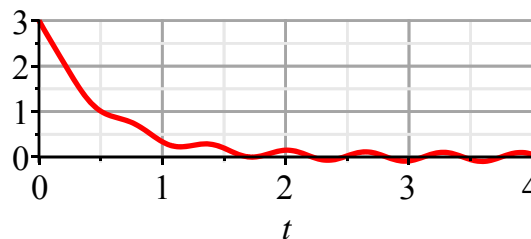
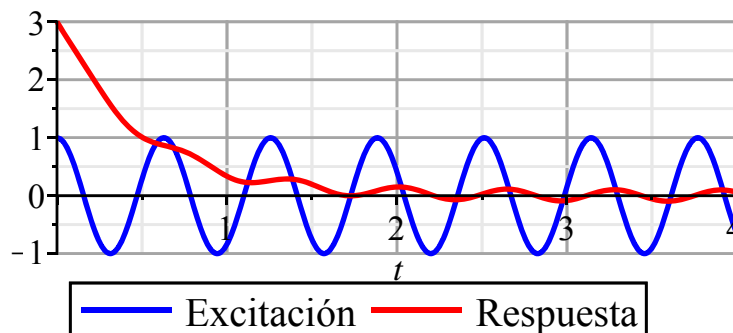


Gráfico múltiple que muestra la salida y la excitación

```
> plot([cos(10 t), fyp], t=0..4, color=[blue, red], gridlines, legend=["Excitación", "Respuesta"], thickness=2)
```



```
>
```



## 13. Resolución gráfica de ODEs: campos de pendientes

### [Volver a la tabla de contenidos](#)

Los *campos de pendientes* son una excelente representación gráfica para ODEs de primer orden en la forma  $y'(x) = F(x, y(x))$ . Los campos de pendientes permiten apreciar claramente el comportamiento cualitativo de las soluciones.

El comando **DEplot()** de la librería **DEtools** permite dibujar los campos de pendientes de forma muy simple.

!!! Para acceder a los comandos contenidos en una librería, debemos cargar previamente la librería con el comando **with()**

```
> with(DEtools)
```

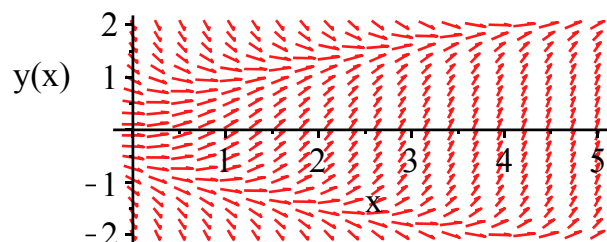
```
[AreSimilar, Closure, DENormal, DEplot, DEplot3d, DEplot_polygon, DFactor, DFactorLCLM,
DFactorsols, Dchangevar, Desingularize, FunctionDecomposition, GCRD, Gosper, Heunsols,
Homomorphisms, IVPsol, IsHyperexponential, LCLM, MeijerGsols,
MultiplicativeDecomposition, ODEInvariants, PDEchangecoords, PolynomialNormalForm,
RationalCanonicalForm, ReduceHyperexp, RiemannPsols, Xchange, Xcommutator, Xgauge,
Zeilberger, abelsol, adjoint, autonomous, bernoullisol, buildsol, buildsym, canoni, caseplot,
casesplit, checkrank, chinisol, clairautsol, constcoeffsols, convertAlg, convertsys, dalembertsol,
dcoeffs, de2diffop, dfieldplot, diff_table, diffop2de, dperiodic_sols, dpolyform, dsubs, eigenring,
endomorphism_charpoly, equinv, eta_k, eulersols, exactsol, expsols, exterior_power, firint,
firtest, formal_sol, gen_exp, generate_ic, genhomosol, gensys, hamilton_eqs, hypergeomsols,
hyperode, indicialeq, infgen, initialdata, integrate_sols, intfactor, invariants, kovacicsols,
leftdivision, liesol, line_int, linearsol, matrixDE, matrix_riccati, maxdimsystems, moser_reduce,
muchange, mult, mutest, newton_polygon, normalG2, ode_int_y, ode_y1, odeadvisor, odepde,
parametricsol, particularsol, phaseportrait, poincare, polysols, power_equivalent,
rational_equivalent, ratsols, redode, reduceOrder, reduce_order, regular_parts, regularsp,
remove_RootOf, riccati_system, riccatisol, rifread, rifsimp, righdivision, rtaylor, separablesol,
singularities, solve_group, super_reduce, symgen, symmetric_power, symmetric_product,
symtest, transinv, translate, untranslate, varparam, zoom]
```

Escribimos la EDO usando la *notación prima* para la derivada, en este caso la variable independiente es  $x$

```
> edo := y' = x - y^2
```

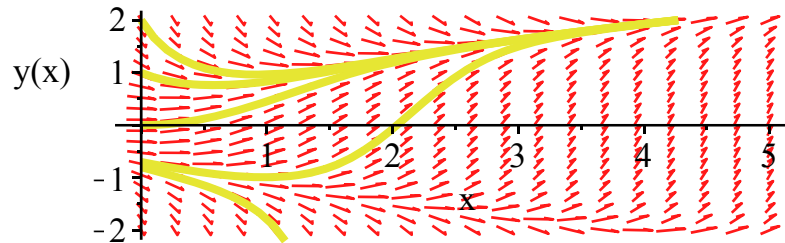
$$edo := \frac{d}{dx} y(x) = x - y(x)^2$$

```
> DEplot(edo, y(x), x = 0 .. 5, y = -2 .. 2)
```



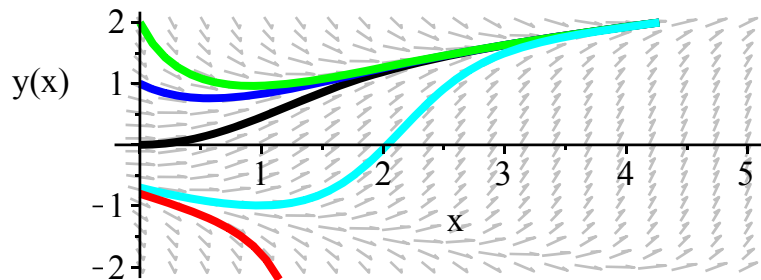
También podemos incluir algunas soluciones particulares suministrando a **DEplot()** una lista de valores iniciales

```
> vinics := [y(0) = 0], [y(0) = 1], [y(0) = 2], [y(0) = -0.7], [y(0) = -0.8]
      vinics := [y(0) = 0], [y(0) = 1], [y(0) = 2], [y(0) = -0.7], [y(0) = -0.8]
> DEplot(edo, y(x), x = 0 .. 5, y = -2 .. 2, [vinics])
```



La opción **linecolor** permite asignar ordenadamente colores a las diferentes soluciones particulares. La opción **color** controla el color de las flechas del campo

```
> vcols := black, blue, green, cyan, red
      vcols := black, blue, green, cyan, red
> DEplot(edo, y(x), x = 0 .. 5, y = -2 .. 2, [vinics], linecolor = [vcols], color = gray)
```



## 14. Resolución numérica de EDOs

[Volver a la tabla de contenidos](#)

La mayoría de EDOs no tienen una solución exacta, expresable como una combinación de funciones elementales

```
> restart
> edo := y' = -x^2 y + cos(x)
      edo := d/dx y(x) = -x^2 y(x) + cos(x)
```

```
> s := dsolve([edo, y(0) = 1], y(x))
      s := y(x) = e^(-1/3 x^3) (int(cos(_z) e^(1/3 _z^3) d_z) from 0 to x) + e^(-1/3 x^3)
```

Para estos casos, la opción **numeric** del comando **dsolve()** nos permite obtener una aproximación numérica de la solución

```
> sn := dsolve([edo, y(0) = 1], y(x), numeric)
      sn := proc(x_rkf45) ... end proc
```

El resultado que nos proporciona **dsolve()** con la opción **numeric** es un programa (procedure) que implementa un método numérico para aproximar la función solución. Por defecto, **dsolve()**

implementa el método de *Runge-Kutta 45 forward*. En caso necesario podemos indicar a **dsolve()** que implemente otros métodos numéricos

```
> sn(1.2)
```

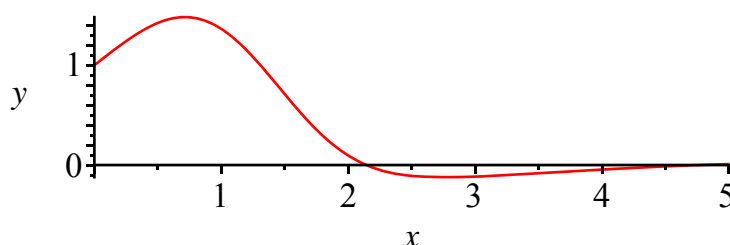
```
[x = 1.2, y(x) = 1.14961123596730]
```

El comando **odeplot()** de la librería **plots()** nos permite dibujar una representación gráfica de la solución usando el programa proporcionado por **dsolve()**

```
> with(plots)
```

```
[animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d, conformal, conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot, display, dualaxisplot, fieldplot, fieldplot3d, gradplot, gradplot3d, implicitplot, implicitplot3d, inequal, interactive, interactiveparams, intersectplot, listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot, matrixplot, multiple, odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d, polyhedra_supported, polyhedraplot, rootlocus, semilogplot, setcolors, setoptions, setoptions3d, spacecurve, sparsematrixplot, surfdata, textplot, textplot3d, tubeplot]
```

```
> odeplot(sn, x = 0 .. 5)
```



### Una EDO de segundo orden

```
> edo2 := y'' + cos(x)y' + x sin(x)y = cos(20 x)^2
```

$$edo2 := \frac{d^2}{dx^2} y(x) + \cos(x) \left( \frac{d}{dx} y(x) \right) + x \sin(x) y(x) = \cos(20 x)^2$$

```
> cond := y(0) = 0, y'(0) = -1
```

```
cond := y(0) = 0, D(y)(0) = -1
```

```
> s := dsolve([edo2, cond])
```

```
s :=
```

Maple no consigue obtener una solución exacta. Construimos una solución aproximada con la opción **numeric**

```
> sn := dsolve([edo2, cond], numeric)
```

```
sn := proc(x_rkf45) ... end proc
```

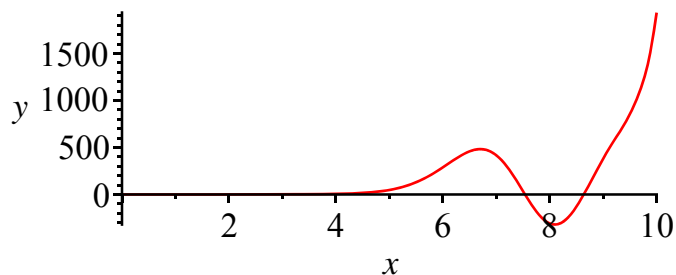
En este caso, el procedimiento programado por Maple nos proporciona, para cada  $x$ , una aproximación numérica del valor que toma la función y su derivada

```
> sn(1.2)
```

```
[x = 1.2, y(x) = -0.413056951893404, d/dx y(x) = 0.187826374585524]
```

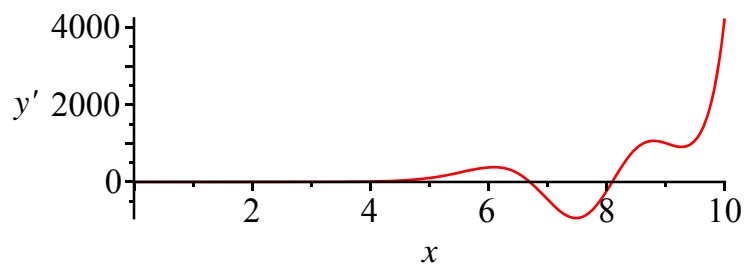
Representamos la solución  $y(x)$  con **odeplot()**

```
> odeplot(sn, x = 0 .. 10)
```



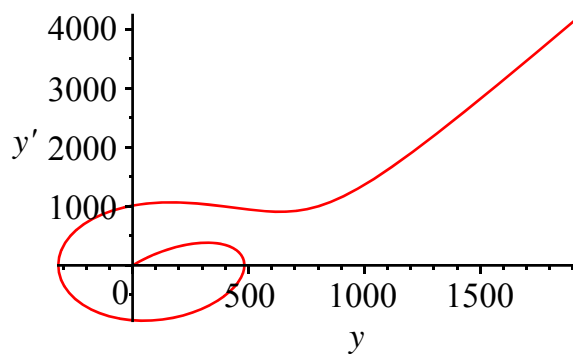
También podemos representar la derivada  $y'(x)$  como sigue

> `odeplot(sn, [x, y'(x)], x=0..10)`



El siguiente comando dibuja el diagrama de fase

> `odeplot(sn, [y(x), y'(x)], x=0..10)`



LL

## 15. Algunas diferencias entre Maple y Matlab

[Volver a la tabla de contenidos](#)

Acciones	Maple	Matlab
Editar	Edición en hojas de trabajo y documentos interactivos	Conjuntos de comandos en m-files
Presentación de resultados	Los resultados son presentados en la misma hoja de trabajo, a continuación del comando	Los resultados se presenta en la ventana de comandos
Ejecución	[ENTER] ejecuta la línea de comandos	[F5] ejecuta el m-file actual
Asignación	<code>:=</code>	<code>=</code>
Eliminar presentación de resultados	<code>:</code>	<code>;</code>
Comentarios	<code>#</code>	<code>%</code>
Separación de comandos	<code>;</code>	<code>,</code>
Romper la línea de comandos sin ejecutar	[Shift]+[Enter]	[Enter], en el editor de m-files
Trabajo con nombres no asignados	Los nombres no asignados actúan como variables simbólicas	La evaluación de un nombre no asignado produce un error

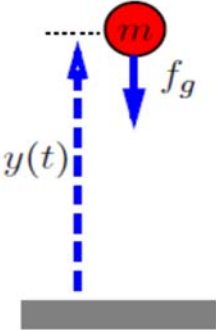
## Computación científica y técnica con Maple

# Práctica 1. Caída libre sin fricción

F. Palacios Quiñonero

Dept. Matemática Aplicada III. Universidad Politécnica de Cataluña

Ver. 1.1. Febrero, 2012

	<ul style="list-style-type: none"> <li>• <math>m</math>, masa (in <math>kg</math>)</li> <li>• <math>g = 9.81 \text{ m/s}^2</math>, aceleración de la gravedad</li> <li>• <math>f_g = m g</math>, peso (en Newtons)</li> <li>• <math>y(t)</math>, posición (en <math>m</math>)</li> <li>• <math>v(t) = \dot{y}(t)</math>, velocidad (en <math>m \text{ s}^{-1}</math>)</li> </ul> <p><b>Ecuación diferencial</b></p> $m \ddot{y} = -m g \Rightarrow \ddot{y}(t) = -g$
<p>El objetivo de este ejercicio práctico es obtener una experiencia directa en el manejo de algunos comandos básicos de Maple. Usamos para ello un problema simple y bien conocido: la caída libre de un objeto cuando la fricción es despreciable.</p>	

## Ejemplo resuelto

[Ecuación diferencial, usando la notación punto para las derivadas

> restart;

> edo :=  $\ddot{y} = -g$

$$edo := \frac{d^2}{dt^2} y(t) = -g$$

[Condiciones iniciales simbólicas

> cond :=  $y(0) = y_0, \dot{y}(0) = v_0$

$$cond := y(0) = y_0, D(y)(0) = v_0$$

[Solución simbólica

> s := dsolve([edo, cond])

$$s := y(t) = -\frac{1}{2} g t^2 + v_0 t + y_0$$

[Vemos que hemos obtenido la bien conocida fórmula general del desplazamiento en el movimiento uniformemente acelerado.

[Lista de valores particulares

```
> vpart := [y0 = 50, v0 = 10, g = 9.81]
```

$$vpart := [y_0 = 50, v_0 = 10, g = 9.81]$$

Extraemos la expresión general del desplazamiento  $y(t)$

```
> fy := eval(y(t), s)
```

$$fy := -\frac{1}{2} g t^2 + v_0 t + y_0$$

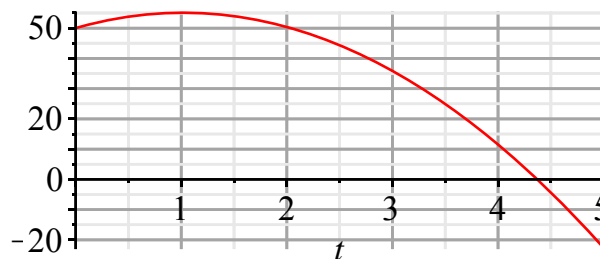
Expresión particular del desplazamiento para los valores particulares contenidos en la lista **vpart**

```
> fyp := eval(fy, vpart)
```

$$fyp := -4.905000000 t^2 + 10 t + 50$$

Gráfico de  $y(t)$  para  $t \in [0, 5]$

```
> plot(fyp, t = 0 .. 5, gridlines)
```



La posición después de 4 segundos es un poco mayor de 10m

Posición después de  $t = 4$  segundos

```
> tp := 4
```

$$tp := 4$$

```
> yp := eval(fyp, t = tp)
```

$$yp := 11.52000000$$

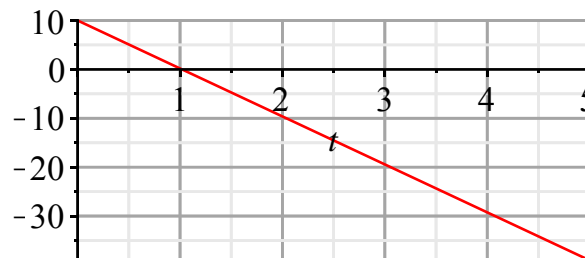
Expresión particular para la velocidad

```
> fvp := diff(fyp, t)
```

$$fvp := -9.810000000 t + 10$$

Gráfico de la velocidad para  $t \in [0, 5]$

```
> plot(fvp, t = 0 .. 5, gridlines)
```



La velocidad después de 4 segundos está alrededor de -30m/s

Velocidad para  $t = 4s$

```
> vp := eval(fvp, t = tp)
```

$$vp := -29.24000000$$

Tiempo de impacto contra el suelo

```
> ts := fsolve(fyp = 0, t = 0 .. 50)
```

$ts := 4.370903613$

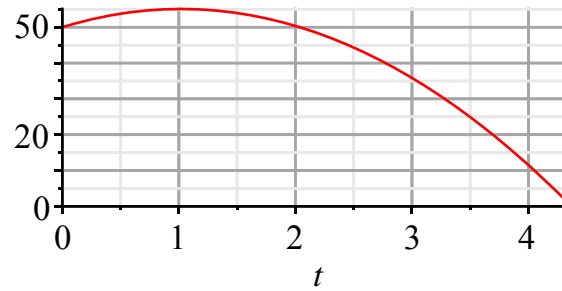
Velocidad en el instante del impacto

>  $vs := eval(fvp, t = ts)$

$vs := -32.87856444$

Gráfico t-y

>  $plot(fyp, t = 0 .. ts, gridlines)$



Tiempo para altura máxima (pico)

>  $tp := solve(fvp = 0)$

$tp := 1.019367992$

Altura máxima

>  $yp := eval(fyp, t = tp)$

$yp := 55.09683996$



### Actividades

Resuelve los siguientes apartados para la EDO  $\ddot{y}(t) = -g$

1. Calcula la solución simbólica para las condiciones iniciales  $y(0) = y_0$ ,  $\dot{y}(0) = v_0$ .
2. Calcula la solución particular correspondiente a la posición inicial  $y_0 = 75\text{m}$ , la velocidad inicial  $v_0 = 28\text{ m/s}$ , y el valor  $g = 9.81\text{m/s}^2$ .
3. Dibuja el gráfico de  $y(t)$  para  $0 \leq t \leq 8\text{s}$ .
4. Calcula la velocidad  $v(t) = \dot{y}(t)$  y dibuja el gráfico de  $v(t)$  para  $0 \leq t \leq 8\text{s}$ .
5. A partir de los gráficos, realiza una estimación aproximada del valor de la posición y la velocidad del objeto para  $t = 2.5\text{ s}$
6. Calcula los valores de la posición y la velocidad del objeto para  $t = 2.5\text{s}$
7. Calcula la altura máxima y el tiempo correspondiente (Indicación: la velocidad es nula en cuando se alcanza la altura máxima). Observa los gráficos correspondientes para verificar que los valores obtenidos son consistentes con la información gráfica disponible.
8. Calcula el instante en que el objeto impacta contra el suelo, y la velocidad de impacto. Observa los gráficos para verificar la consistencia de los resultados obtenidos.
9. Compara tus resultados con los obtenidos por alguno de tus compañeros. En caso de discrepancia, intentad encontrar el error.

**Nota.** Como guía, puedes consultar el tutorial de Maple y el ejemplo resuelto. Sin embargo, es recomendable que evites copiar literalmente los comandos, y/o copiar-pegar fragmentos de código. Recuerda que el objetivo de este ejercicio práctico es obtener una experiencia práctica en el manejo de los comandos básicos de Maple.

### Soluciones

```
> restart;
```

```
[Ecuación diferencial, usamos la notación punto para derivadas
```

```
> edo :=  $\ddot{y} = -g$ 
```

$$edo := \frac{d^2}{dt^2} y(t) = -g$$

```
[Condiciones iniciales simbólicas
```

```
> cond :=  $y(0) = y_0, \dot{y}(0) = v_0$ 
```

$$cond := y(0) = y_0, D(y)(0) = v_0$$

```
[Solución simbólica
```

```
> s := dsolve([edo, cond])
```

$$s := y(t) = -\frac{1}{2} g t^2 + v_0 t + y_0$$

Lista de valores particulares

```
> vpart := [y0 = 75, v0 = 28, g = 9.81]
```

$$vpart := [y_0 = 75, v_0 = 28, g = 9.81]$$

Extraemos la expresion del desplazamiento  $y(t)$

```
> fy := eval(y(t), s)
```

$$fy := -\frac{1}{2} g t^2 + v_0 t + y_0$$

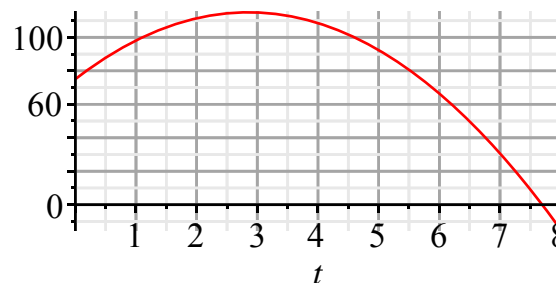
Expresion particular del desplazamiento para los valores particulares contenidos en la lista **vpart**

```
> fyp := eval(fy, vpart)
```

$$fyp := -4.905000000 t^2 + 28 t + 75$$

Gráfico de  $y(t)$  para  $t \in [0, 8]$

```
> plot(fyp, t = 0 .. 8, gridlines)
```



La posición después de 2.5 segundos es un poco mayor de 110m

Posición después de  $t = 2.5$  segundos

```
> tp := 2.5
```

$$tp := 2.5$$

```
> yp := eval(fyp, t = tp)
```

$$yp := 114.3437500$$

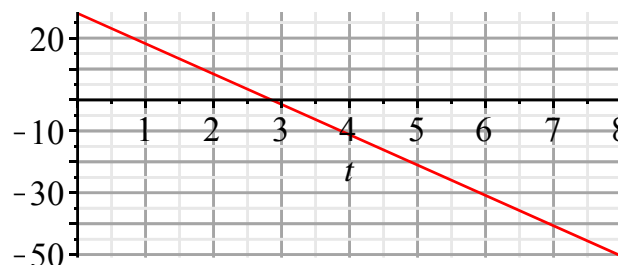
Expresión particular para la velocidad

```
> fvp := diff(fyp, t)
```

$$fvp := -9.810000000 t + 28$$

Gráfico de  $v(t)$  para  $t \in [0, 8]$

```
> plot(fvp, t = 0 .. 8, gridlines)
```



La velocidad en el instante  $t = 2.5$  está alrededor de 4m/s. El cuerpo aún está ascendiendo en ese instante

Velocidad para  $t = 2.5$  s

>  $vp := eval(fvp, t = tp)$

$vp := 3.47500000$

Tiempo de impacto contra el suelo

>  $ts := fsolve(fyp = 0, t = 0 .. 50)$

$ts := 7.695423507$

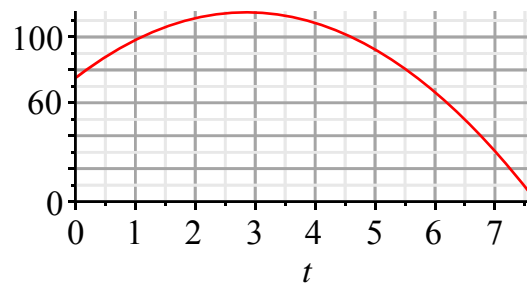
Velocidad en el instante del impacto

>  $vs := eval(fvp, t = ts)$

$vs := -47.49210460$

Gráfico t-y, ajustado al tiempo de vuelo

>  $plot(fyp, t = 0 .. ts, gridlines)$



Tiempo para altura máxima (pico)

>  $tp := solve(fvp = 0)$

$tp := 2.854230377$

Altura máxima

>  $yp := eval(fyp, t = tp)$

$yp := 114.9592253$

>

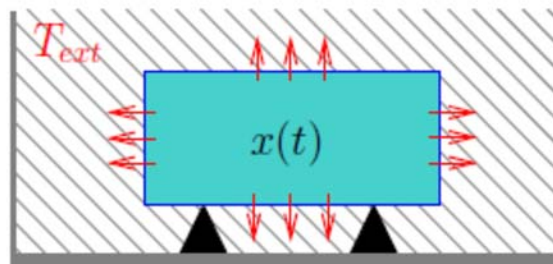
## Computación científica y técnica con Maple

# Práctica 2. Ley de enfriamiento de Newton

F. Palacios Quiñonero

Dept. Matemática Aplicada III. Universidad Politécnica de Cataluña

Ver. 1.1. Febrero, 2012



En este ejercicio práctico, vamos a estudiar la evolución de la temperatura de un objeto  $x(t)$  ubicado en un entorno con una temperatura  $T_{ext}(t)$ . Para este problema, la Ley de enfriamiento de Newton establece que *el cambio de temperatura en el objeto es proporcional a la diferencia de temperatura entre el objeto y el medio  $T_{ext} - x(t)$* .

Con mayor precisión,

$$\frac{d}{dt}x(t) = a (T_{ext}(t) - x(t)),$$

donde los distintos símbolos tienen el siguiente significado:

- $t$ , tiempo (en minutos)
- $x(t)$  temperatura del cuerpo (en °C)
- $T_{ext}(t)$  temperatura externa (en °C)
- $a > 0$ , constante de proporcionalidad

**Observa** que si  $T_{ext}(t) > x(t) \Rightarrow \dot{x}(t) > 0$ . Por lo tanto,  $x(t)$  es creciente en este caso. Análogamente,  $x(t)$  decrece para  $T_{ext}(t) < x(t)$ .

### Ejemplo 1. Un ejemplo básico

> restart

Ecuación diferencial

> edo :=  $\dot{x} = a (T_{ext} - x)$

$$edo := \frac{d}{dt} x(t) = a (T_{ext} - x(t))$$

Condiciones iniciales

>  $cond := x(0) = x_0$

$$cond := x(0) = x_0$$

Solución simbólica

>  $s := dsolve([edo, cond])$

$$s := x(t) = T_{ext} + e^{-at} (x_0 - T_{ext})$$

Test de la solución

>  $odetest(s, [edo, cond])$

$$[0, 0]$$

Expresión para  $x(t)$

>  $fx := eval(x(t), s)$

$$fx := T_{ext} + e^{-at} (x_0 - T_{ext})$$

Valores particulares de los parámetros y la condición inicial

>  $pvals := \left[ a = \frac{1}{2}, x_0 = 100, T_{ext} = 20 \right]$

$$pvals := \left[ a = \frac{1}{2}, x_0 = 100, T_{ext} = 20 \right]$$

Expresión de la solución para los valores particulares de los parámetros

>  $fxp := eval(fx, pvals)$

$$fxp := 20 + 80 e^{-\frac{1}{2}t}$$

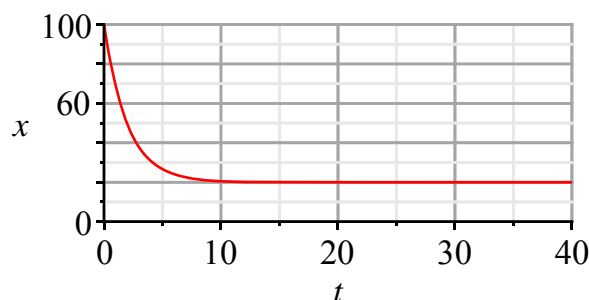
Temperatura después de 5 minutos

>  $vx := eval(fxp, t=5.)$

$$vx := 26.56679989$$

Evolución de la temperatura  $x(t)$  para los primeros 40 minutos

>  $plot(fxp, t=0..40, x=0..100, gridlines)$



Tiempo necesario para que la temperatura del objeto descienda hasta 45°C

>  $vt := fsolve(fxp=45)$

$$vt := 2.326301620$$

## Ejemplo 2. Un ejemplo algo más realista

En un determinado momento, la temperatura de un horno es de 220 °C. Después de 15 minutos, la temperatura ha descendido hasta 85 °C.

- Suponiendo que la temperatura ambiente se mantiene constante a 25°C, usa la Ley de Enfriamiento de Newton para calcular la temperatura del horno después de 30 minutos.
- Asumiendo que el horno incorpora un mecanismo de seguridad que impide la apertura a temperaturas superiores a 45°C, calcula el tiempo necesario para la apertura del horno.

Empezamos con los mismos cuatro pasos iniciales seguidos en el ejemplo anterior.

> *restart*;

Ecuación diferencial

> *edo* :=  $\dot{x} = a (T_{ext} - x)$

$$edo := \frac{d}{dt} x(t) = a (T_{ext} - x(t))$$

Condiciones iniciales

> *cond* :=  $x(0) = x_0$

$$cond := x(0) = x_0$$

Solución simbólica

> *s* := *dsolve*( [*edo*, *cond*] )

$$s := x(t) = T_{ext} + e^{-at} (x_0 - T_{ext})$$

> *fx* := *eval*(*x*(*t*), *s*)

$$fx := T_{ext} + e^{-at} (x_0 - T_{ext})$$

Ahora, introducimos los valores particulares de la temperatura inicial  $x_0 = 220$  °C, y la temperatura exterior  $T_{ext} = 25$  °C

> *pvals1* := [ $T_{ext} = 25$ ,  $x_0 = 220$ ]

$$pvals1 := [T_{ext} = 25, x_0 = 220]$$

Obtenemos la expresión de la solución con esos valores particulares de los parámetros

> *fx1* := *eval*(*fx*, *pvals1*)

$$fx1 := 25 + 195 e^{-at}$$

A continuación, usamos la información adicional  $x(15) = 85$  °C para calcular el valor de la constante de proporcionalidad  $a$

> *vx15* := *eval*(*fx1*,  $t = 15$ )

$$vx15 := 25 + 195 e^{-15a}$$

Valor exacto de  $a$

> *va* := *solve*(*vx15* = 85)

$$va := -\frac{1}{15} \ln\left(\frac{4}{13}\right)$$

Aproximación float del valor de  $a$

>  $vaf := evalf(va)$

$$vaf := 0.07857699974$$

Una vez calculado el valor de  $a$ , obtenemos la expresion de la solución

>  $fx2 := eval(fx1, a = va)$

$$fx2 := 25 + 195 e^{\frac{1}{15} \ln\left(\frac{4}{13}\right) t}$$

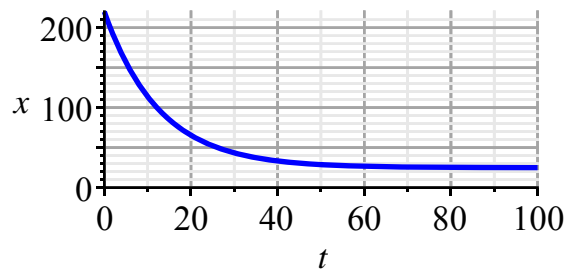
Expresion float

>  $fx2f := evalf(fx2)$

$$fx2f := 25. + 195. e^{-0.07857699974 t}$$

Evolución de la temperatura para los primeros 100 minutos

>  $plot(fx2, t = 0..100, x = 0..220, gridlines, color = [blue], thickness = 2)$



Temperatura del horno después de 30 minutos

>  $v30 := eval(fx2f, t = 30.)$

$$v30 := 43.46153847$$

Tiempo para apertura de seguridad

>  $ts := solve(fx2 = 45)$

$$ts := \frac{15 \ln\left(\frac{4}{39}\right)}{\ln\left(\frac{4}{13}\right)}$$

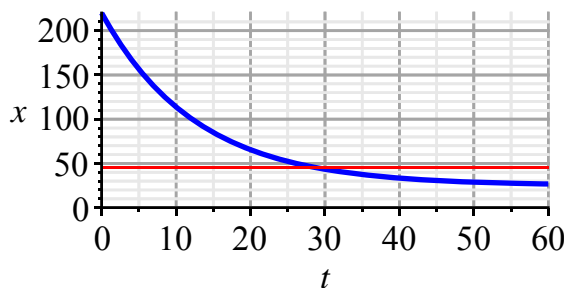
Aproximación float

>  $tsf := evalf(ts)$

$$tsf := 28.98134686$$

Resolución gráfica

>  $plot([fx2, 45], t = 0..60, x = 0..220, gridlines, color = [blue, red], thickness = [2, 1])$



### Ejemplo 3. Estudio del tiempo de enfriamiento en función del valor de la temperatura del medio

En el ejemplo anterior hemos obtenido una expresión genérica para la temperatura  $x(t)$ . El nombre asignado a esa solución es  $fx$

>  $fx$

$$T_{ext} + e^{-at} (x_0 - T_{ext})$$

Maple nos permite resolver todas las cuestiones anteriores manteniendo indeterminado el valor de los parámetros y la condición inicial. Para ilustrar el interés de este enfoque, vamos a estudiar el tiempo de enfriamiento en función de la temperatura del medio  $T_{ext}$

Tiempo necesario para alcanzar la temperatura de apertura segura 45 °C

>  $t45 := \text{solve}(fx = 45, t)$

$$t45 := - \frac{\ln \left( \frac{T_{ext} - 45}{-x_0 + T_{ext}} \right)}{a}$$

Usando esta expresión, podemos investigar la dependencia del tiempo de apertura segura respecto de la temperatura externa. En este ejemplo, tomaremos los valores correspondientes al horno del Ejemplo 2

>  $pvals2 := [a = va, x_0 = 220]$

$$pvals2 := \left[ a = -\frac{1}{15} \ln \left( \frac{4}{13} \right), x_0 = 220 \right]$$

Evaluamos el tiempo de enfriamiento seguro para los valores particulares de  $a$  y  $x_0$

>  $ts := \text{eval}(t45, pvals2)$

$$ts := \frac{15 \ln \left( \frac{T_{ext} - 45}{-220 + T_{ext}} \right)}{\ln \left( \frac{4}{13} \right)}$$

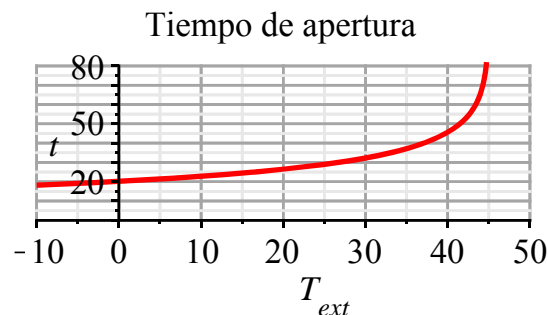
Evaluación float

>  $tsf := \text{evalf}(ts)$

$$tsf := -12.72637035 \ln \left( \frac{T_{ext} - 45.}{-220. + T_{ext}} \right)$$

Gráfico del tiempo de enfriamiento seguro en función de la temperatura exterior

>  $\text{plot}(tsf, T_{ext} = -10 .. 50, t = 0 .. 80, \text{gridlines}, \text{thickness} = 2, \text{title} = \text{"Tiempo de apertura"})$

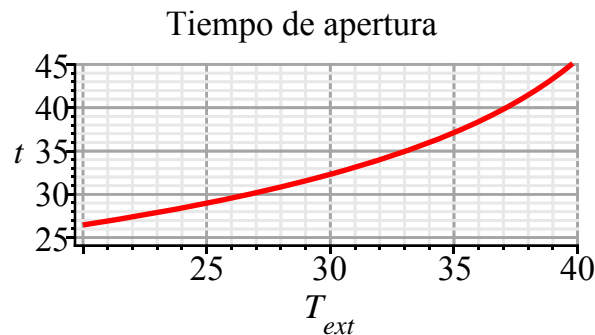




A partir del gráfico, podemos apreciar claramente que el tiempo de enfriamiento seguro aumenta significativamente cuando la temperatura del medio está alrededor de 40°C. La asíntota vertical localizada en  $T_{ext} = 45$  °C indica que la temperatura de seguridad no puede alcanzarse (obviamente) cuando  $T_{ext} \geq 45$  °C.

Si asumimos que el horno opera normalmente en el rango de temperaturas externas  $20 \leq T_{ext} \leq 40$ , podemos dibujar el siguiente gráfico:

```
> plot(tsf, T_ext = 20 ..40, t = 25 ..45, gridlines, thickness = 2, title = "Tiempo de apertura")
```



En el gráfico podemos apreciar una importante variabilidad en el tiempo de enfriamiento seguro, con un rango aproximado de 20 minutos. A continuación, calculamos con precisión ese rango

```
> ts20 := eval(tsf, T_ext = 20.)
```

$ts20 := 26.46374318$

```
> ts40 := eval(tsf, T_ext = 40.)
```

$ts40 := 45.60518916$

```
> rangts := ts40 - ts20
```

$rangts := 19.14144598$

```
>
```

### Actividad 1

Tomando como referencia el Ejemplo 1, resuelve las siguientes cuestiones para la EDO

$$\frac{d}{dt}x(t) = a (T_{ext}(t) - x(t))$$

1. Resuelve la ecuación diferencial para los valores particulares  $a = 0.2$ ,  $T_{ext} = 35^{\circ}\text{C}$ , y la temperatura inicial  $x(0) = 300^{\circ}\text{C}$
2. Dibuja el gráfico de la temperatura  $x(t)$  para la primera media hora ( $t \in [0, 30]$ ). Incluye la cuadrícula en el gráfico.
3. A partir del gráfico, estima aproximadamente el valor de la temperatura para  $t = 10$  min.
4. Calcula con **eval()** la temperatura  $x(10)$ . Compara el valor obtenido con la estimación que has hecho a partir del gráfico.
5. Usa el gráfico para estimar el tiempo necesario para que el objeto alcance una temperatura de  $150^{\circ}\text{C}$ .
6. Calcula con **fsolve()** el tiempo necesario para que el objeto alcance  $150^{\circ}\text{C}$ . Compara el resultado con la estimación anterior.

### Actividad 2

Ahora supongamos que tenemos un horno a  $530^{\circ}\text{C}$  y que, después de 15 minutos, la temperatura ha decrecido hasta  $110^{\circ}\text{C}$ .

1. Asumiendo que la temperatura del entorno se mantiene constante a  $30^{\circ}\text{C}$ , usa la ley de enfriamiento de Newton para determinar la temperatura después de 22 minutos.
2. Dibuja un gráfico de la evolución de la temperatura  $x(t)$  a lo largo de la primera media hora. Usa el gráfico para estimar la temperatura después de 10 minutos, y el tiempo necesario para que el horno descienda hasta una temperatura de  $150^{\circ}\text{C}$ .
3. Calcula con **eval()** la temperatura  $x(10)$ .
4. Calcula con **fsolve()** el tiempo  $t$  para  $x(t) = 150^{\circ}\text{C}$ .

**Nota:** Puedes usar el Ejemplo 2 como orientación. Consulta el ejemplo e intenta escribir tu propio código, procurando comprender el funcionamiento de los comandos y su sintaxis.

### Actividad 3

Para los valores particulares  $a = 0.3$ , y  $T_{ext} = 40^\circ\text{C}$ .

1. Dibuja el campo de pendientes de la ecuación diferencial. Usa los intervalos  $0 \leq t \leq 40 \text{ min}$ ,  $0 \leq x(t) \leq 300^\circ\text{C}$ .
2. Dibuja sobre el campo de pendientes la solución particular correspondiente a la temperatura inicial  $x(0) = 200^\circ\text{C}$ . Para esta solución particular, estima sobre el gráfico la temperatura en  $t = 5 \text{ min}$ , y el tiempo requerido para  $x(t) = 150^\circ\text{C}$ . Dibuja la curva en color azul y grosor doble (opciones **linecolor=blue** y **thickness=2**).
3. Dibuja el campo de pendientes con las soluciones particulares correspondientes a las condiciones iniciales  $x(0) = 300^\circ\text{C}$ ,  $x(0) = 250^\circ\text{C}$ ,  $x(0) = 200^\circ\text{C}$ ,  $x(0) = 40^\circ\text{C}$ ,  $x(0) = 20^\circ\text{C}$ . Dibuja las flechas del campo en color gris (opción **color=gray**), y asigna los colores **black**, **red**, **green**, **blue**, **cyan** a las diferentes curvas.
6. Suponiendo que la temperatura inicial se sitúa en el rango  $[200^\circ\text{C}, 300^\circ\text{C}]$ , realiza una estimación del rango de temperaturas después de 4 minutos.
7. Suponiendo que la temperatura inicial está en el rango  $[200^\circ\text{C}, 300^\circ\text{C}]$ , realiza una estimación del tiempo requerido para que la temperatura del objeto descienda hasta  $100^\circ\text{C}$ .
8. Explica el comportamiento de las curvas con condiciones iniciales  $x(0) = 40^\circ\text{C}$ ,  $x(0) = 20^\circ\text{C}$ .
9. Calcula valores numéricos para las estimaciones realizadas en los apartados (2), (4) and (5).

## Soluciones

### Actividad 1

EDO

> restart

> edo :=  $\dot{x} = a (T_{ext} - x)$

$$edo := \frac{d}{dt} x(t) = a (T_{ext} - x(t))$$

Condiciones iniciales

> cond :=  $x(0) = x_0$

$$cond := x(0) = x_0$$

Solución simbólica

> s := dsolve([edo, cond])

$$s := x(t) = T_{ext} + e^{-at} (x_0 - T_{ext})$$

Expresión para la solución

> fx := eval(x(t), s)

$$fx := T_{ext} + e^{-at} (x_0 - T_{ext})$$

Valores particulares de los parámetros y la condición inicial

> pvals :=  $\{a = 0.2, T_{ext} = 35, x_0 = 300\}$

$$pvals := \{a = 0.2, T_{ext} = 35, x_0 = 300\}$$

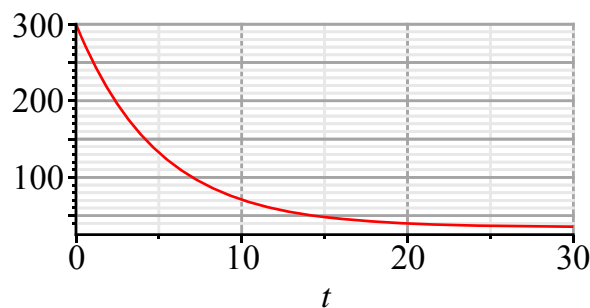
Expresión para la solución particular

> fxp := eval(fx, pvals)

$$fxp := 35 + 265 e^{-0.2t}$$

Gráfico

> plot(fxp, t = 0..30, gridlines)



Valor estimado en el gráfico  $x(10) \approx 75^\circ$

Valor calculado para  $x(10)$

> vx10 := eval(fxp, t = 10.)

$$vx10 := 70.86385005$$

Tiempo estimado para  $x(t) = 150^\circ \rightarrow 4\text{min}$

Tiempo calculado para  $x(t) = 150^\circ$

```
> t150 := fsolve(fxp = 150)
t150 := 4.173988488
```

## Actividad 2

### 1. Temperatura después de 22min.

```
> restart
```

Primero, calculamos el valor de la constante  $a$

EDO

```
> edo := xdot = a * (T_ext - x)
edo := d/dt x(t) = a (T_ext - x(t))
```

```
> cond := x(0) = 530
cond := x(0) = 530
```

```
> s := dsolve([edo, cond])
s := x(t) = T_ext + e^(-a*t) (530 - T_ext)
```

```
> fx := eval(x(t), s)
fx := T_ext + e^(-a*t) (530 - T_ext)
```

```
> fxp := eval(fx, T_ext = 30)
fxp := 30 + 500 e^(-a*t)
```

```
> vx15 := eval(fxp, t = 15)
vx15 := 30 + 500 e^(-15*a)
```

```
> va := solve(vx15 = 110, a)
va := -1/15 ln(4/25)
```

```
> vaf := evalf(va)
vaf := 0.1221720976
```

Expresión para  $x(t)$

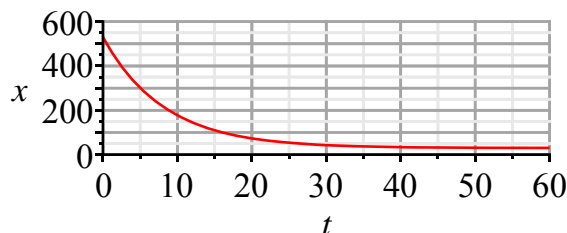
```
> fxp1 := eval(fxp, a = vaf)
fxp1 := 30 + 500 e^(-0.1221720976*t)
```

Temperatura después de 22min

```
> vx22 := eval(fxp1, t = 22)
vx22 := 64.01569213
```

### 2.1 Grafico para la primer hora

```
> plot(fxp1, t = 0 .. 60, x = 0 .. 600, gridlines)
```



Valores estimados:  $x(10) \approx 175^\circ \text{C}$ ; tiempo aproximado para  $x(t) = 150^\circ \text{C} \rightarrow t \approx 12 \text{ min.}$

### 2.3, 2.4 Valores calculados

>  $vx10 := eval(fxp1, t = 10.)$

$vx10 := 177.3612599$

>  $vt150 := fsolve(fxp1 = 150)$

$vt150 := 11.68119713$

## Actividad 3

### 3.1 Campo de pendientes

> *restart*

>  $edo := \dot{x} = a (T_{ext} - x)$

$$edo := \frac{d}{dt} x(t) = a (T_{ext} - x(t))$$

>  $pvals := [a = 0.3, T_{ext} = 40]$

$pvals := [a = 0.3, T_{ext} = 40]$

EDO particular para los valores de los parámetros

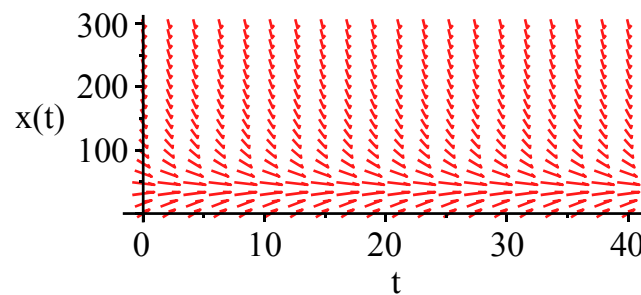
>  $edop := eval(edo, pvals)$

$$edop := \frac{d}{dt} x(t) = 12.0 - 0.3 x(t)$$

Carga de la librería **DEtools**

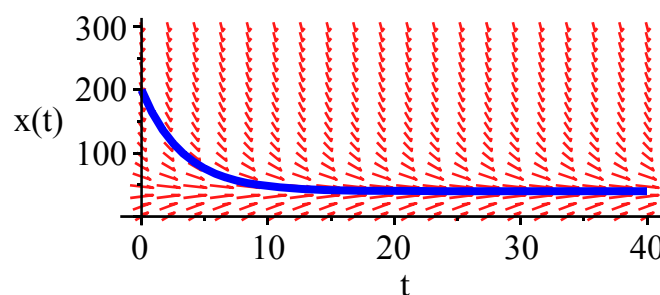
> *with(DEtools) :*

>  $DEplot(edop, x(t), t = 0..40, x = 0..300)$



### 3.2 Gráfico de la solución del problema de valor inicial con $x(0) = 200$

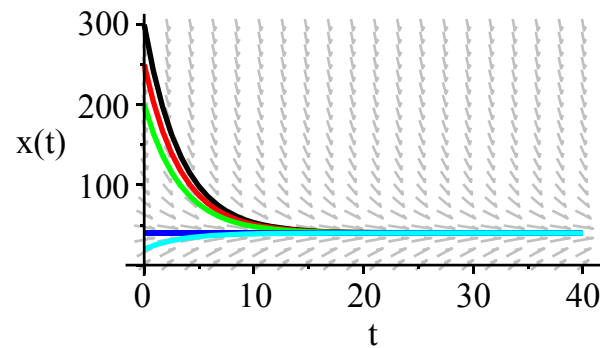
>  $DEplot(edop, x(t), t = 0..40, x = 0..300, [x(0) = 200], linecolor = blue)$



Valores estimados a partir del gráfico: temperatura para  $t = 5 \text{ min} \approx 75^\circ \text{C}$ . Tiempo para  $x(t) = 150^\circ \text{C}$ , approx 1.5 min.

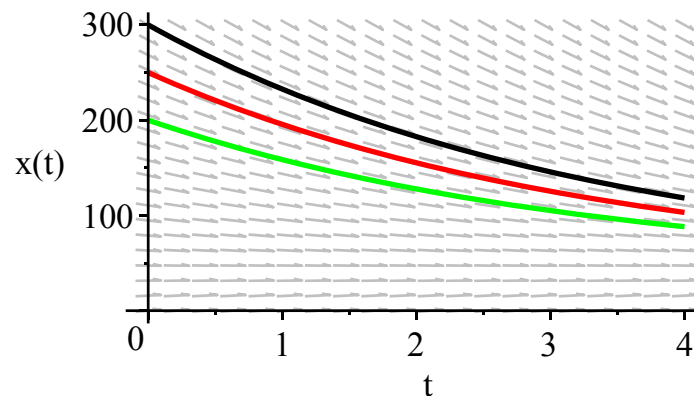
### 3.3 Conjunto de soluciones para los valores iniciales dados

- >  $pvalins := x(0) = 300, x(0) = 250, x(0) = 200, x(0) = 40, x(0) = 20$   
 $pvalins := x(0) = 300, x(0) = 250, x(0) = 200, x(0) = 40, x(0) = 20$
- >  $pcols := black, red, green, blue, cyan$   
 $pcols := black, red, green, blue, cyan$
- >  $DEplot(edop, x(t), t = 0..40, x = 0..300, [pvalins], linecolor = [pcols], color = gray, thickness = 2)$



### 3.4 Estimación del rango de temperaturas después de 4 min para soluciones con valores iniciales en el rango [200,300]

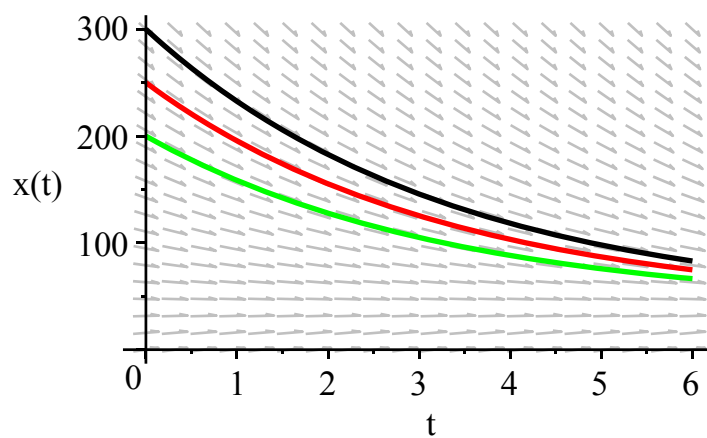
- >  $pvalins := x(0) = 300, x(0) = 250, x(0) = 200$   
 $pvalins := x(0) = 300, x(0) = 250, x(0) = 200$
- >  $pcols := black, red, green$   
 $pcols := black, red, green$
- >  $DEplot(edop, x(t), t = 0..4, x = 0..300, [pvalins], linecolor = [pcols], color = gray, thickness = 2)$



El rango aproximado de temperaturas es [90,120]

### 3.5 Estimación del rango de tiempo para $temp=100^\circ C$ cuando la temperatura inicial está en el rango [200,300]

- >  $DEplot(edop, x(t), t = 0..6, x = 0..300, [pvalins], linecolor = [pcols], thickness = 2, color = gray)$



El rango aproximado de tiempos es 3-5 min.