# Introducción a Maxima
# Software Libre Científico

Henry R. Moncada

Universidad Nacional del Callao
Facultad de Ciencias Naturales y Matemáticas

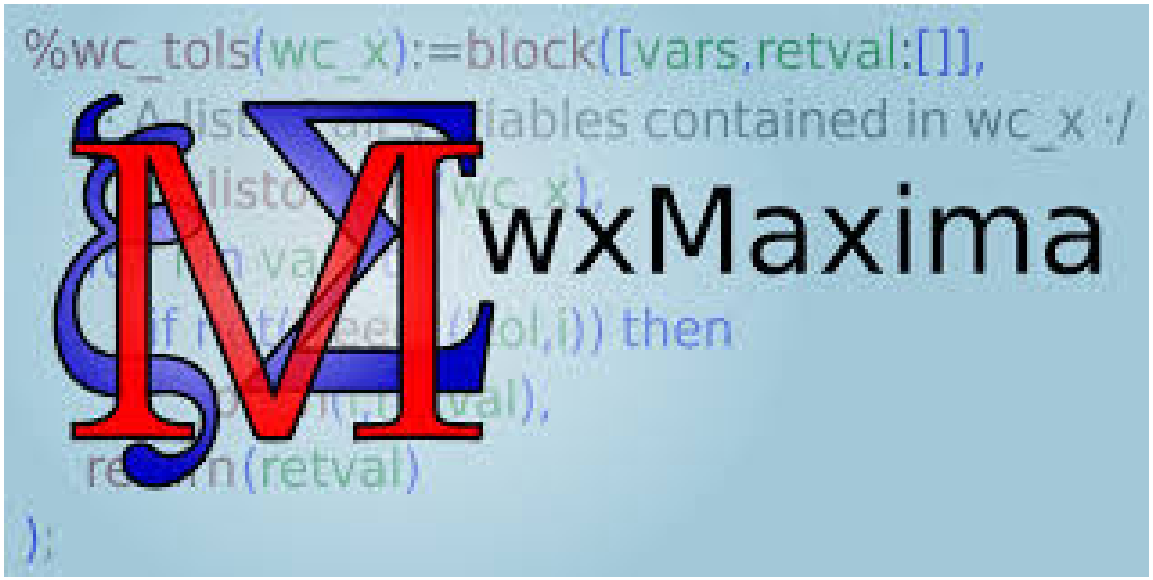
8 de enero de 2026

# Esquema de la presentación

1. Primeros pasos con Maxima
2. Funciones básicas de Maxima
3. Representaciones gráficas
4. Límites y derivadas
5. Matrices
6. Programación en Maxima

# ¿Qué es Maxima?

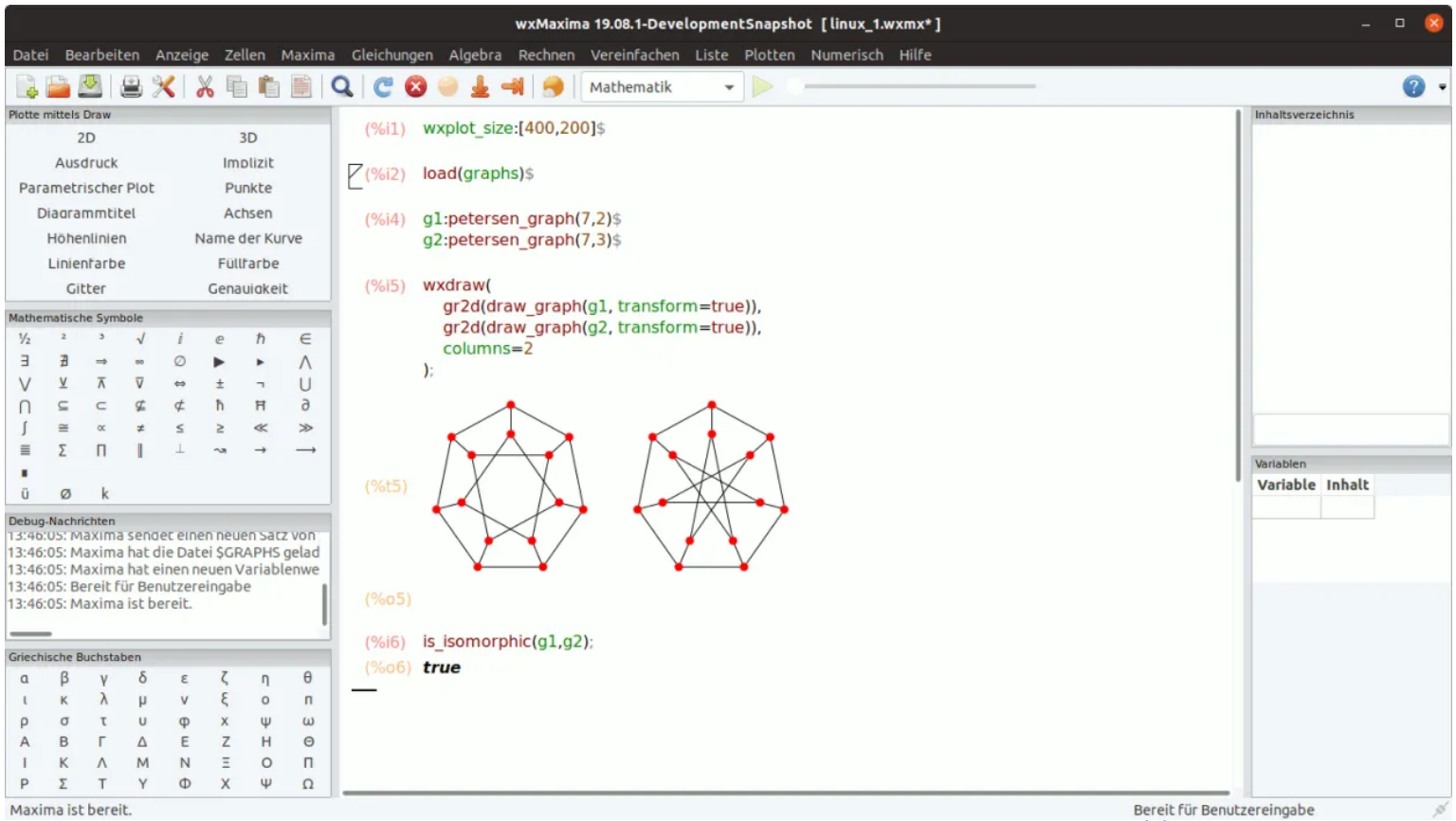
- Sistema para manipulación de expresiones simbólicas y numéricas
- Incluye: diferenciación, integración, series de Taylor, transformadas de Laplace, EDOs, sistemas de ecuaciones lineales, matrices y tensores
- Produce resultados de alta precisión
- Grafica funciones en 2D y 3D
- Código fuente compilable en Windows, Linux y MacOS X
- Descendiente de Macsyma (MIT, años 1960)
- Dinámico: mejoras continuas en código y documentación
- Disponible en: `https://maxima.sourceforge.io/`



## Descarga e instalación

El programa es de libre distribución y puede descargarse desde :

- sourceforge `https://maxima.sourceforge.io/`
- wxmaxima-developers
  `https://wxmaxima-developers.github.io/wxmaxima/download.html`



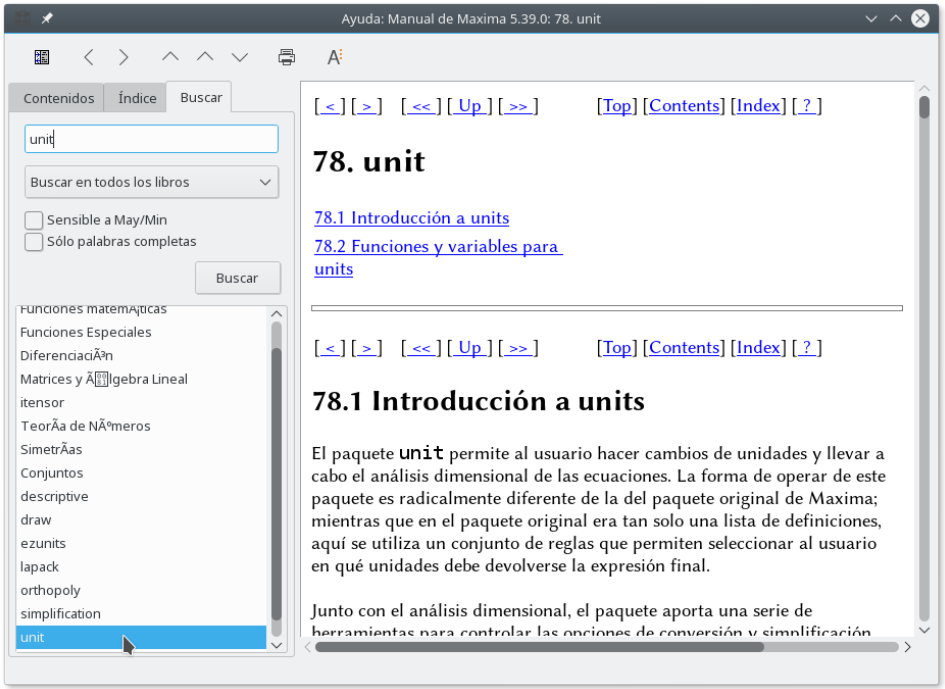
# La ayuda de wxMaxima

- wxMaxima permite acceder a la amplia ayuda incluida de manera gráfica
- Comandos útiles:

```
describe(expr)     # ayuda sobre expr
example(expr)      # ejemplo de expr
apropos("expr")    # comandos relacionados con expr
??expr             # comandos que contienen expr
```

## Ayuda completa

Disponible en: `http://maxima.sourceforge.net/` en formatos PDF y web.




Cuadro del menú Ayuda-Contenido

Si conocemos el nombre del comando sobre el que estamos buscando ayuda, la orden describe nos da una explicación sobre la variable, comando o lo que sea que hayamos preguntado

## Uso de describe

```
(%i2) describe(dependencies);
      0: dependencies (Functions and Variables for Differentiation)
      1: dependencies $<1>$ (Functions and Variables for Differentiation)
Enter space-separated numbers, 'all' or 'none':
```

A veces nos equivocamos y no recordamos exactamente el nombre del comando

## Búsqueda aproximada

```
(%i2) describe(plot)
      No exact match found for topic ?plot?.
      Try ??? plot? (inexact match) instead.
(%o2) false
```

La solución la tenemos escrita justo en la salida anterior: busca en la ayuda comandos, variables, etc. que contengan la cadena **"plot"**

## Búsqueda con ??

```
(%i2) ??plot

0: Funciones y variables para plotdf
1: Introduccion a plotdf
2: barsplot (Funciones y variables para graficos estadisticos)
3: boxplot (Funciones y variables para graficos estadisticos)
4: contour_plot (Funciones y variables para graficos)
5: gnuplot_close (Funciones y variables para graficos)
6: gnuplot_replot (Funciones y variables para graficos)
7: gnuplot_reset (Funciones y variables para graficos)
8: gnuplot_restart (Funciones y variables para graficos)
9: gnuplot_start (Funciones y variables para graficos)
10: plot2d (Funciones y variables para graficos)
11: plot3d (Funciones y variables para graficos)
12: plotdf (Funciones y variables para plotdf)
13: plot_options (Funciones y variables para graficos)
14: scatterplot (Funciones y variables para graficos estadisticos)
15: set_plot_option (Funciones y variables para graficos)
Enter space-separated numbers, ?all? or ?none?:none;
15: set_plot_option (Funciones y variables para graficos)
```

¿Quieres que lo adapte en un tono más académico, como para incluirlo en un manual/tutorial de Maxima?
Si hay varias posibles elecciones, Maxima queda a la espera hasta que escribamos el número que corresponde al ítem en el que estamos interesados, o **all** o **none** si estamos interesados en todos o en ninguno, respectivamente.
En el menú de wxMaxima **Ayuda** → **Propósito para**. Su propósito es similar a **??**, pero el resultado es levemente distinto: nos da la lista de comandos en los que aparece la cadena plot sin incluir nada más. Si ya tenemos una idea de lo que estamos buscando, muchas veces será suficiente con esto.

## Búsqueda con apropos

```
(%i2) apropos("plot");
(%o2) [barsplot,boxplot,contour_plot,contour_plot-1.get_plot_option,...]
```

## Ejemplos con example

```
(%i2) example(limit);
(%i2) limit(x*log(x),x,0,plus);
(%i2) limit((x+1)^(1/x),x,0);
(%i2) limit(%e^x/x,x,inf);
(%i2) limit(sin(1/x),x,0);
```

# Operaciones básicas

El sistema wxMaxima emplea la notación estándar para escribir las operaciones matemáticas básicas: suma +, resta -, producto $*$, cociente / y potencia ^. Tras introducir una expresión, basta pulsar la tecla **Enter** para que wxMaxima la ejecute.

## Operaciones matemáticas

```
(%i2) (2+3*(a^2)^3)/12;
(%o2) (3*a^6+2)/12
(%i3) sqrt(12)+25^8;
(%o3) 2*sqrt(3) + 152587890625
```

Es importante recordar que en wxMaxima es necesario escribir siempre el operador de multiplicación (*). De lo contrario, dará un error de sintaxis.

## ¡Importante!

Siempre escribir el operador de multiplicación (*)

```
(%i2) 2x^3;     # Error
(%o2) incorrect syntax: x is not an infix operator 2x^^
(%i2) 2*x^3;    # Correcto
(%o2) 2x 3
```

# Funciones matemáticas

Funciones matemáticas elementales

| Expresión | Sintaxis en wxMaxima |
|---|---|
| $e^x$ | `exp(x)` ó `%e^x` |
| $\sqrt{x}$ | `sqrt(x)` |
| $\|x\|$ | `abs(x)` |
| $\ln(x)$ | `log(x)` |
| $\log_{10}(x)$ | `log10(x) := log(x)/log(10)` |

## Ejemplos

```
(%i1) exp(5);
(%o1) %e^5
(%i2) %e^5;
(%o2) %e^5
(%i3) sqrt(9);
(%o3) 3
(%i4) abs(-3);
(%o4) 3
(%i7) log10(x):=log(x)/log(10);
(%o7) log10(x):=log(x)/log(10)
```

Operar con números complejos

| Concepto | Sintaxis en wxMaxima |
|---|---|
| Unidad imaginaria | `%i` |
| Forma binómica | `a+b*%i` |
| Forma exponencial | `r*%e^(%i*c)` |
| Módulo | `cabs(z)` |
| Argumento | `carg(z)` |
| Conversión a forma binómica | `rectform(z)` |
| Conversión a forma exponencial | `polarform(z)` |

## Ejemplos

```
(%i8) 2-2*%i;
(%o8) 2-2*%i
(%i9) cabs(2-2*%i);
(%o9) 2^(3/2)
(%i10)  carg(2-2*%i);
(%o10)  -%pi/4
(%i11)  polarform(2-2*%i);
(%o11)  2^(3/2)*%e^(-(%i*%pi)/4)
```

# Asignar nombres a datos o expresiones y como introducir comentarios

En wxmaxima, la asignación se realiza utilizando el signo dos puntos **(:)** para asignar un valor a una variable, seguido del valor deseado. Por ejemplo:

- Para asignar el número 5 a la variable x, escribirías **x:5**.
- Si la entrada termina en punto y coma **(;)**, el resultado de la asignación se mostrará;
- Si termina en signo de dólar **($)**, el resultado no se mostrará, como en el ejemplo **a:-7$;**.

## Asignaciones

```
(%i17)  x:5;
(%o17)  5
(%i15)  a:-5;
(%o15)  -5
(%i16)  a:-7$
(%i18)  c:2*x+1;
(%o18)  11
```

## Asignaciones

```
(%i5) z:2-2*%i;
(%o5) 2 - 2*%i
(%i5) cabs(z);
(%o5) 2^(3/2)
(%i19)  k:%e^%i;
(%o19)  %e^%i
(%i5) rectform(k)
(%o5) %i*sin(1) + cos(1)
```

**Introducir comentarios**

## Sintaxis de comentarios

```
/* texto de comentarios */
```

## Ejemplo

```
(%i5) /*Apartado (a)*/ f(x):=x^2;
(%o5) f(x):= x^2
```

## Para definir una función utilízamos:

**if** condicion **then** sentencia 1 **else** sentencia 2;

## Definición de funciones

```
(%i5) f(x):=2*x;
(%o5) f(x):= 2*x
(%i5) g(x,y):=2*x^2+5*y^4;
(%o5) g(x,y):= 2*x^2 + 5*y^4
```

$$h(x) = \begin{cases} x & \text{si } x \leq 0 \\ \sin(x) & \text{si } x > 0 \end{cases}$$

## Función definida a trozos

```
(%i20)  h(x):=if x<=0 then x else sin(x);
(%o20)  h(x):=if x<=0 then x else sin(x)
```

**Resolver ecuaciones:** El comando **solve()** se utiliza para resolver ecuaciones algebraicas y sistemas de ecuaciones. Se escribe **solve(expresión, variable)** para una ecuación o **solve([ecuación1, ecuación2], [variable1, variable2])** para un sistema, y las expresiones se ingresan con el signo igual **(=)** y los productos se denotan con el asterisco **(*)**.

## Comando solve

```
(%i2) solve(4*x^2+x=5,x);
(%o2) [x = -5/4, x = 1]
```

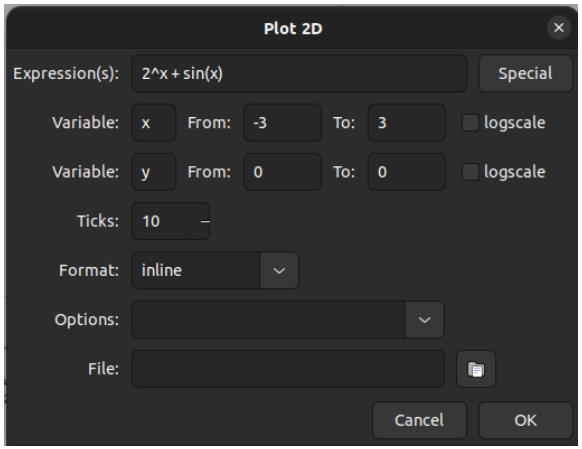
## Sistemas de ecuaciones

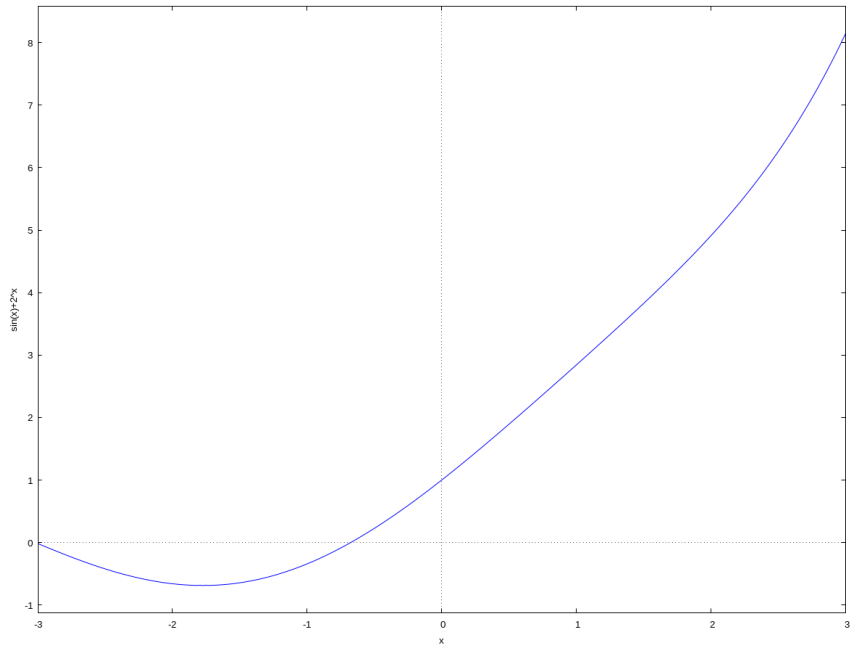
```
(%i2) eq1:2*x+y=5;
(%i2) eq2:2*x+5*y=10;
(%i2) solve([eq1, eq2], [x,y]);
(%o2) [[x = 15/8, y = 5/4]]
```

# Funciones de una variable - Gráficas 2D

- Menú: **Tramado → Tramado 2d...**
- Introducir rango y formato deseado
- Pulsar Aceptar para ver el gráfico
- Formato gnuplot: abre en ventana separada con coordenadas del cursor

**Representar varias funciones simultáneamente**

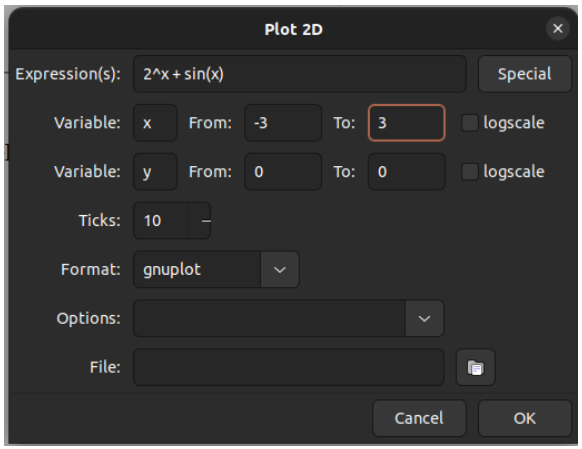


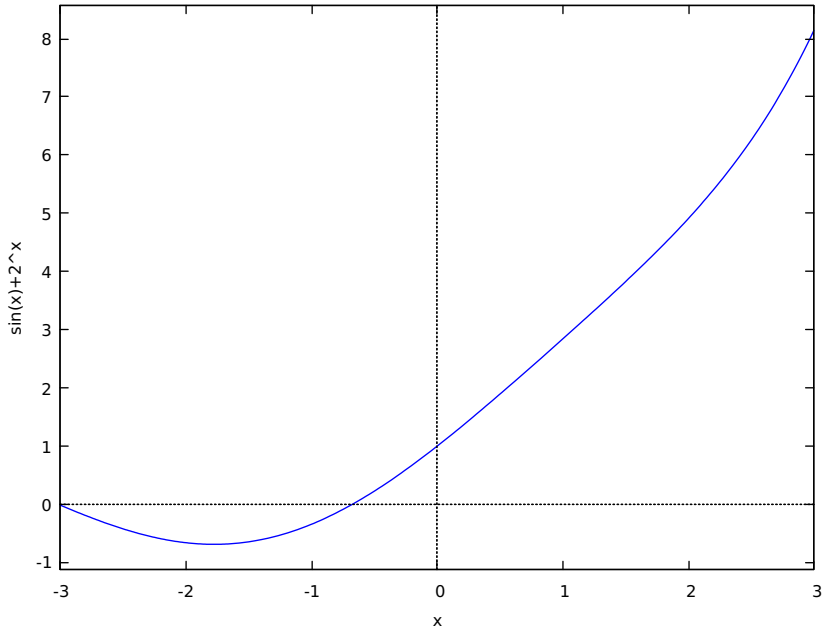




## Definición de funciones

```
wxplot2d([2^x + sin(x)], [x,-3,3])$
```

Si queremos que el gráfico se abra en otra pestaña, tiene la ventaja de que te indica las coordenadas del punto marcado por el cursor en cada momento, **deberemos elegir el formato: gnuplot**. En este caso, es recomendable cerrar la ventana gráfica al terminar, para poder seguir trabajando.



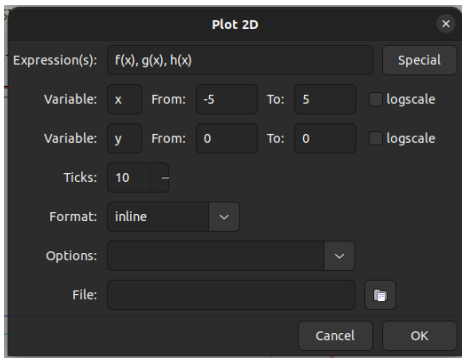

## Definición de funciones

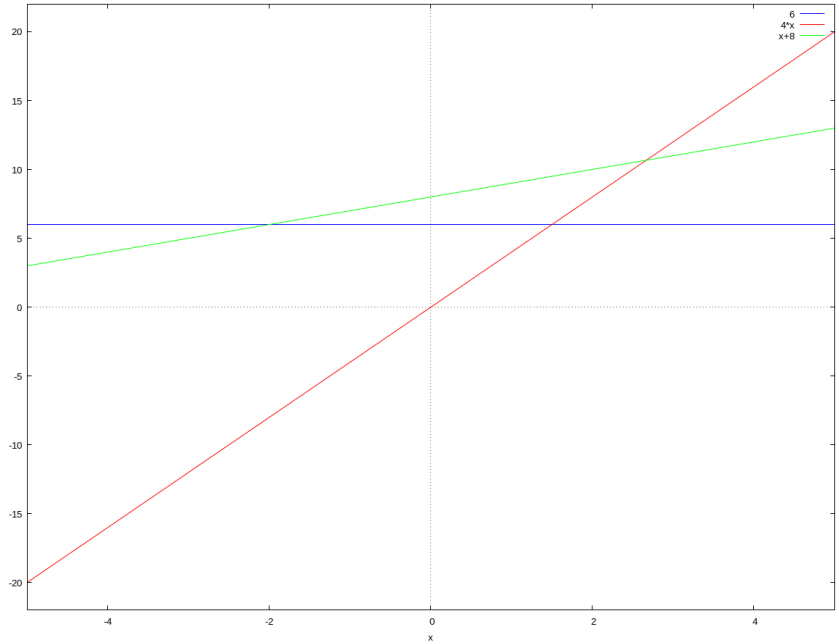
```
plot2d([2^x + sin(x)], [x,-3,3])$
```

**Representar varias funciones simultáneamente**

## Definición de funciones

```
(%i5) f(x):=6;
(%o5) f(x):=6
(%i6) g(x):=4*x;
(%o6) g(x):=4*x
(%i7) h(x):=x+8;
(%o7) h(x):=x+8
```




## Gráfico múltiple

```
(%i9) plot2d([f(x),g(x),h(x)], [x,-5,5], [plot_format,
      gnuplot])$
(%i10)  wxplot2d([f(x), g(x), h(x)], [x,-5,5])$
```

# Funciones de dos variables - Gráficas 3D

- Menú: **Tramado** → **Tramado 3d...**
- Similar a funciones de una variable pero con dos variables
- También curvas de nivel con `contour_plot`

## Comando limit

```
(%i2) f(x) := (x+a)*(x^2+b*x+c);
(%i2) limit(f(x), x, 1);
(%o2) (a+1)*c + (a+1)*b + a + 1
```

## Límites laterales

```
(%i2) limit(1/x,x,0,minus);  # L\'imite por izquierda
(%o2) -inf
(%i2) limit(1/x,x,0,plus);   # L\'imite por derecha
(%o2) inf
```

## Comando diff

```
(%i2) g(x,y):=sin(x*y);
(%i2) diff(g(x,y),x);
(%o2) y*cos(x*y)
(%i2) diff(g(x,y),y);
(%o2) x*cos(x*y)
```

## Derivadas de orden superior

```
(%i2) diff(g(x,y),x,2);       # Segunda derivada
(%o2) -y^2*sin(x*y)
(%i2) diff(g(x,y),x,4);       # Cuarta derivada
(%o2) y^4*sin(x*y)
```

# Definir una matriz

- Menú: **Álgebra** → **Introducir matriz**
- `matrix([a11,a12,...],[a21,a22,...],...)`
- `entermatrix(filas, columnas)` (interactivo)
- `a[i,j]:=fórmula$ genmatrix(a, filas, columnas)`
- Matrices especiales: `diagmatrix, ematrix, zeromatrix, ident`

# Operaciones con matrices

- Operadores: +, -, . (producto), * (elemento a elemento)
- Funciones: `transpose, adjoint, invert, determinant`
- `rank, charpoly, eigenvalues, eigenvectors`

## Ejemplo de matriz

```
(%i2) a:matrix([3,3,3], [1,3,5], [4,2,7]);
(%o2) [ 3   3   3 ]
      [ 1   3   5 ]
      [ 4   2   7 ]
```

## Definición de funciones

```
(%i2) area_tri(base, altura):= base*altura/2;
(%i2) area_cua(lado):= lado^2;
```

## Bucles for

```
(%i2) for b:0 thru 2 do
          print(b, "el cuadrado es igual a ", b^2);
0 el cuadrado es igual a 0
1 el cuadrado es igual a 1
2 el cuadrado es igual a 4
(%o2) done
```

## Condicional if

```
(%i2) for b:0 thru 5 do
          if b^2>5 then print([b, "v\'alido"])
          else print ([b, "no es v\'alido"]);
[0, no es v\'alido]
[1, no es v\'alido]
[2, no es v\'alido]
[3, v\'alido]
[4, v\'alido]
[5, v\'alido]
(%o2) done
```

## Secuencia con block

```
(%i2) block([x,y], x:2, y:5, z:x+y);
(%o2) 7
```

## Función compleja

```
(%i2) f(x):= x^2-5*x+6;
(%i2) mifunc(a,b,n):=block(
   h:(b-a)/n,
   for i:0 thru n do
      if f(a+i*h)>0 then display(f(a+i*h))
      else print("no es positivo")
);
```

Henry R. Moncada
Universidad Nacional del Callao
Facultad de Ciencias Naturales y Matemáticas
31 de Marzo de 2022