

□ **Sesión 1: Introducción a Maxima.** **Cálculo simbólico y cálculo aproximado.** **Errores y estabilidad numérica.**

□ ***1 Introducción a Maxima***

☞ <http://hermite.ugr.es/maxima/>

□ **1.1 Cálculo simbólico y numérico**

☞ `--> 3 + 1/2;`

☞ `--> 3.0 + 1/2;`

☞ `--> 3 + 1/2, numer;`

☞ `--> float(%pi + %e);`

☞ `--> bfloat(%pi + %e);`

☞ `--> fpprec : 50$
bfloat(%pi^5 + %e^2);`

☞ `--> fpprec : 25$
bfloat(%pi^5 + %e^2);`

☞ `--> fpprec : 50$
bfloat(%pi^2 + %e^2);`

□ 1.2 Asignación de variables

```
[-> x : 23;  
    y : 48;  
    x*y;
```

```
[-> x/y;
```

```
[-> x/y, numer;
```

```
[-> x-y;
```

[-> Si queremos resolver la ecuación $x^2+3x-2=0$:

```
[-> solve(x^2+3*x-2=0, x);
```

[-> dos formas de solucionar el problema:

```
[-> x=23;
```

```
[-> x;
```

```
[-> x+y;
```

```
[-> solve('x^2+3*'x-2=0, 'x);
```

```
[-> x;
```

```
[-> kill(x) $  
    solve(x^2+3*x-2=0, x);
```

```
[-> x;
```

☐ También trabaja con expresiones simbólicas

☐ `--> expr : a+b/a+b^2+sqrt(b/a);`

☐ `--> ratsimp(expr);`

☐ `--> radcan(expr);`

☐ `--> expand(expr);`

☐ `--> expr, a=1, b=2;`

☐ `--> a;`

☐ `--> kill(a);`

☐ `--> expr, a:1, b=2;`

☐ `--> a:1;`

☐ `--> a;`

☐ **1.3 Definición de funciones**

☐ `--> f(x) := x^2 + sin(x) - exp(x);`

☐ `--> f(3);`

☐ `--> f(3.9);`

☐ `--> plot2d([f(x)], [x, -5, 5],
[plot_format, openmath])$`

□ 1.4 Matrices

```
⌈ --> kill(all) $
```

```
⌈ --> A : matrix([1, 2, 3], [4, 5, 6], [7, 8, 9]);
```

```
⌈ --> A[1,2];
```

```
⌈ --> B : genmatrix(b,3,3);
```

```
⌈ --> b[i,j]:=i-j;
```

```
⌈ --> B;
```

```
⌈ --> B : genmatrix(b,3,3);
```

```
⌈ --> B.A;
```

```
⌈ --> B*A;
```

```
⌈ --> A^2;
```

```
⌈ --> A^^2;
```

```
⌈ --> submatrix(1,A,1);
```

```
⌈ --> row(A,2);
```

```
⌈ --> col(A,3);
```

```
⌈ --> addcol(A, [1,2,3]);
```

□ **2 Error por cancelación de dígitos en una operación simple**

```
└ --> kill(all);
```

```
└ --> a : 1.0 $  
      b : 10.0^20 $  
      c : -10.0^20 $  
      [(a+b)+c,a+(b+c)];
```

```
└ --> a : 123456789012.13 $  
      b : 123456789012.12 $  
      [a^2-b^2,(a+b)*(a-b)];
```

```
└ --> for i : 1 thru 18 step 1 do (  
      x : 33.12*10^i,  
      r1 : sqrt(x)/(sqrt(x+1)+sqrt(x)),  
      r2 : sqrt(x)*(sqrt(x+1)-sqrt(x)),  
      print([i, r1, r2, r1-r2])  
    )$
```

□ **3 Error propagado en un proceso iterativo inestable**

```
└ Computación simbólica
```

```
└ --> kill(all);
```

```
--> x : 1 $  
    y : 1/3 $  
    for i : 1 thru 20 step 1 do (  
        z : 10*y/3-x,  
        t : 1/3^(i+1),  
        print([z, t, z-t]),  
        x : y,  
        y : z  
    ) $
```

Computación numérica

```
--> kill(all);
```

```
--> x : 1.0 $  
    y : 1.0/3.0 $  
    for i : 1 thru 40 step 1 do (  
        z : 10*y/3-x,  
        t : 1.0/3^(i+1),  
        print([i, z, t, z-t]),  
        x : y,  
        y : z  
    ) $
```

4 *Criterio de parada por repetición de dígitos*

```
--> x : 1.0 $  
    for i : 1 thru 1000 step 1 do x : cos(x) $  
    print(float(x)) $
```

```
--> y : 0.1 $  
      x : cos(y) $  
      for i : 1 while abs(x-y)>10^-8 step 1 do (  
        y : x,  
        x : cos(y),  
        k : i  
      )$  
      print([k,float(x)]) $
```

□ **5 *Límite incorrecto por cancelación***

Probar en:

```
x : %pi,numer $ (función nula, derivada no nula y x no nulo)  
x : 0.0 $ (función nula, derivada no nula y x nulo)
```

```
--> kill(all);
```

```
--> f(x) := sin(x) $  
      x : %pi,numer $  
      for i : 1 thru 30 step 1 do (  
        h : 10^-i,  
        print([i, (f(x+h)-f(x))/h])  
      ) $
```

□ **6 *Sistema lineal mal condicionado***

Ejemplo 1

```
--> kill(all);
```

```
--> /*programilla de cosecha propia para resolver sistemas a partir de la matriz  
de coeficientes y el vector de términos independientes, sin tener que escribir  
las ecuaciones*/
```

```
linearsolve(A,b) := linsolve (  
    makelist(  
        (A.makelist(var[k],k,1,length(transpose(A)))-b)[k][1]=0, k, 1, length(A)  
    ),  
    makelist(var[k], k, 1, length(transpose(A)))  
) $
```

```
--> A : matrix(  
    [-73, 78, 24],  
    [-80, 37, 10],  
    [92, 66, 25]  
) $  
b : [29, 183, -33] $  
linearsolve(A,b);
```

```
--> A[1,1]:=-73+1/100;
```

```
--> float(linearsolve(A,b));
```

Ejemplo 2

```
--> kill(all);
```

```

--> /*ejecutar primero lo siguiente dejando para estos valores los
    que vienen por defecto. Después volver a ejecutar cambiando
    estos valores */

/*keepfloat if true prevents floating point numbers from being converted
to rational numbers.*/
keepfloat : true;

/* ratprint if true, a message informing the user of the conversion of floating
point numbers to rational numbers is displayed. */
ratprint : true;

--> linearsolve(A,b) := linsolve (
    makelist(
        (A.makelist(var[k],k,1,length(transpose(A)))-b)[k][1]=0, k, 1, length(A)
    ),
    makelist(var[k], k, 1, length(transpose(A)))
) $

--> x : [1234, 4567] $
A : matrix(
    [7891234, 4571833],
    [7891234*2-1, 4571833*2-1]
) $
b : A.x $
print(A.matrix([xx],[yy]),"=", b) $

--> linearsolve(A,b);

--> dA : genmatrix(lambda([i,j],(random(1.0)-1/2)/10),2,2);

--> float(linearsolve(A+dA,b)) ;

--> linsolve([7891234*xx+4571833*yy = 30617344067, 15782467*xx+9143665*yy = 61234682333], [xx, yy]);

```

```
--> linsolve([  
    (7891234+random(0.1)-0.05)*xx + (4571833+random(0.1)-0.05)*yy = 30617344067,  
    (15782467+random(0.1)-0.05)*xx + (9143665+random(0.1)-0.05)*yy = 61234682333  
], [xx, yy]);
```