4 Resolución de ecuaciones

4.1 Ecuaciones y operaciones con ecuaciones 124 4.2 Resolución de ecuaciones 126 4.3 Ejercicios 138

Maxima nos va a ser de gran ayuda en la resolución de ecuaciones, ya sean sistemas de ecuaciones lineales con un número grande de incógnitas (y parámetros) o ecuaciones no lineales. Un ejemplo típico es encontrar las soluciones de un polinomio. En este caso es fácil que alguna de las soluciones sea compleja. No importa. *Maxima* se maneja bien con números complejos. De hecho, *siempre* trabaja con números complejos. Si tienes alguna duda de cómo operar con números complejos, en el Apéndice A tienes una breve introducción sobre su uso.

4.1 Ecuaciones y operaciones con ecuaciones

En *Maxima*, una ecuación es una igualdad entre dos expresiones algebraicas escrita con el símbolo =.

expresión1=*expresión2*	ecuación
lhs(*expresión1*=*expresión2*)	expresión1
rhs(*expresión1*=*expresión2*)	expresión2

Si escribimos una ecuación, *Maxima* devuelve la misma ecuación.

```
(%i1)  3*x^2+2*x+x^3-x^2=4*x^2;
(%o1)  x^3+2x^2+2x=4x^2
```

además podemos asignarle un nombre para poder referirnos a ella

```
(%i2)   eq:3*x^2+2*x+x^3-a*x^2=4*x^2;
(%o2)   x^3-a*x^2+3*x^2+2*x=4*x^2
```

y operar como con cualquier otra expresión

```
(%i3)   eq-4*x^2;
(%o3)   x^3-a*x^2-x^2+2*x=0
```

Podemos seleccionar la expresión a la izquierda o la derecha de la ecuación con las órdenes `lhs` y `rhs` respectivamente.

```
(%i4)   lhs(eq);
(%o4)   x^3-a*x^2+3*x^2+2*x
```

4.2 Resolución de ecuaciones

Maxima puede resolver los tipos más comunes de ecuaciones y sistemas de ecuaciones algebraicas de forma exacta. Por ejemplo, sabe encontrar las raíces de polinomios de grado bajo (2,3 y 4). En cuanto a polinomios de grado más alto o ecuaciones más complicadas, no siempre será posible encontrar la solución exacta. En este caso, podemos intentar encontrar una solución aproximada en lugar de la solución exacta.

4.2.1 La orden `solve`

La primera orden que aparece en el menú dentro de *Maxima* para resolver ecuaciones es `solve`. Esta orden intenta dar todas las soluciones, ya sean reales o complejas de una ecuación o sistema de ecuaciones. Se puede acceder a esta orden desde el menú **Ecuaciones→Resolver** o escribiendo directamente en la ventana de *Maxima* la ecuación a resolver.

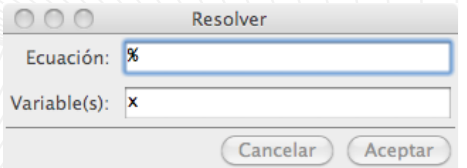

Figura 4.1 Resolver una ecuación

```
(%i5)  solve(x^2-3*x+1=0,x);
```

$$(\%o5)\quad [x=-\frac{\sqrt{5}-3}{2},x=\frac{\sqrt{5}+3}{2}]$$

`solve(`*ecuación*`,`*incógnita*`)`	resuelve la ecuación
`solve(`*expr*`,`*incógnita*`)`	resuelve la expresión igualada a cero
`solve([`*ecuaciones*`],[`*variables*`])`	resuelve el sistema
`multiplicities`	guarda la multiplicidad de las soluciones

También podemos resolver ecuaciones que dependan de algún parámetro. Consideremos la ecuación "eq1":

```
(%i6)  eq1:x^3-a*x^2-x^2+2*x=0;
(%o6)  x^3-a*x^2-x^2+2*x=0
(%i7)  solve(eq1,x);
(%o7)  [x=-(sqrt(a^2+2*a-7)-a-1)/2,x=(sqrt(a^2+2*a-7)+a+1)/2,x=0]
```

Sólo en el caso de ecuaciones con una única variable podemos ahorrarnos escribirla

```
(%i8)  solve(x^2+2*x=3);
(%o8)  [x=-3,x=1]
```

También podemos no escribir el segundo miembro de una ecuación cuando éste sea cero

```
(%i9)  solve(x^2+2*x);
(%o9)  [x=-2,x=0]
(%i10)  solve(x^2+2*x=0);
(%o10)  [x=-2,x=0]
```

Cuando buscamos las raíces de un polinomio hay veces que es conveniente tener en cuenta la multiplicidad de las raíces. Ésta se guarda automáticamente en la variable `multiplicities`. Por ejemplo, el polinomio $x^7 - 2\,x^6 + 2\,x^5 - 2\,x^4 + x^3$ tiene las raíces 0, 1, i, $-i$,

```
(%i11)   solve(x^7-2*x^6+2*x^5-2*x^4+x^3);
(%o11)   [x=-%i,x=%i,x=1,x=0]
```

pero estas raíces no pueden ser simples: es un polinomio de grado 7 y sólo tenemos 4 raíces. ¿Cuál es su multiplicidad?

```
(%i12)   multiplicities;
(%o12)   [1,1,2,3]
```

O sea que i y $-i$ son raíces simples, 1 tiene multiplicidad 2 y 0 es una raíz triple.

Observación 4.1. Mucho cuidado con olvidar escribir cuáles son las incógnitas.

```
(%i13)   solve(eq1);
         More unknowns than equations - 'solve'
         Unknowns given :
         [a,x]
         Equations given:
         [x^3-ax^2+3x^2+2x=4x^2]
         - an error. To debug this try debugmode(true);
```

Hay dos variables y *Maxima* no sabe con cuál de ellas quedarse como incógnita. Aunque nosotros estemos acostumbrados a utilizar las letras x, y, z como incógnitas, para *Maxima* tanto a como x tienen perfecto sentido como incógnitas y la respuesta en uno de ellos no nos interesa:

```
(%i14)   solve(eq1,a);
```

```
(%o14)    [a=(x^2-x+2)/2]
```

La orden `solve` no sólo puede resolver ecuaciones algebraicas.

```
(%i15)    solve(sin(x)*cos(x)=0,x);
          solve: is using arc-trig functions to get a solution.
          Some solutions will be lost.
(%o15)    [x=0,x=%pi/2]
```

¿Qué ocurre aquí? La expresión $\mathrm{sen}(x)\cos(x)$ vale cero cuando el seno o el coseno se anulen. Para calcular la solución de $\mathrm{sen}(x) = 0$ aplicamos la función arcoseno a ambos lados de la ecuación. La función arcoseno vale cero en cero pero la función seno se anula en muchos más puntos. Nos estamos dejando todas esas soluciones y eso es lo que nos está avisando *Maxima*.

Como cualquiera puede imaginarse, *Maxima* no resuelve todo. Incluso en las ecuaciones más "sencillas", las polinómicas, se presenta el primer problema: no hay una fórmula en términos algebraicos para obtener las raíces de un polinomio de grado 5 o más. Pero no hay que ir tan lejos. Cuando añadimos raíces, logaritmos, exponenciales, etc., la resolución de ecuaciones se complica mucho. En esas ocasiones lo más que podemos hacer es ayudar a *Maxima* a resolverlas.

```
(%i16)    eq:x+3=sqrt(x+1);
(%o16)    x+3=sqrt(x+1)
(%i17)    solve(eq,x);
```

```
(%o17)   [x=sqrt(x+1)-3]
(%i18)   solve(eq^2);
(%o18)   [x=-(sqrt(7)%i+5)/2,x=(sqrt(7)%i-5)/2]
```

Cómo hacer referencia a las soluciones

Uno de los ejemplos usuales en los que utilizaremos las soluciones de una ecuación es en el estudio de una función. Necesitaremos calcular puntos críticos, esto es, ceros de la derivada. El resultado de la orden `solve` no es una lista de puntos, es una lista de ecuaciones.

Una primera solución consiste en usar la orden `rhs` e ir recorriendo uno a uno las soluciones:

```
(%i19)   sol:solve(x^2-4*x+3);
(%o19)   [x=3,x=1]
```

La primera solución es

```
(%i20)   rhs(part(sol,1));
(%o20)   3
```

y la segunda

```
(%i21)   rhs(part(sol,2));
```

```
(%o21)    1
```

Este método no es práctico en cuanto tengamos un número un poco más alto de soluciones. Tenemos que encontrar una manera de aplicar la orden `rhs` a toda la lista de soluciones. Eso es justamente para lo que habíamos presentado la orden `map`:

```
(%i22)    sol:map(rhs,solve(x^2-4*x+3));
(%o22)    [3,1]
```

Sistemas de ecuaciones

También podemos resolver sistemas de ecuaciones. Sólo tenemos que escribir la lista de ecuaciones y de incógnitas. Por ejemplo, para resolver el sistema

$$\left.\begin{aligned} x^2+y^2 &= 1 \\ (x-2)^2+(y-1)^2 &= 4 \end{aligned}\right\}$$

escribimos

```
(%i23)    solve([x^2+y^2=1,(x-2)^2+(y-1)^2=4],[x,y]);
(%o23)    [[x=4/5,y=-3/5],[x=0,y=1]]
```

Siempre hay que tener en cuenta que, por defecto, *Maxima* da todas las soluciones incluyendo las complejas aunque muchas veces no pensemos en ellas. Por ejemplo, la recta $x+y=5$ no corta a la circunferencia $x^2+y^2=1$:

```
(%i24)  solve([x^2+y^2=1,x+y=5],[x,y]);
(%o24)  [[x=-(sqrt(23)%i-5)/2,y=(sqrt(23)%i+5)/2],[x=(sqrt(23)%i+5)/2,y=-(sqrt(23)%i-5)/2]]
```

Si la solución depende de un parámetro o varios, *Maxima* utilizará %r1, %r2,... para referirse a estos. Por ejemplo,

```
(%i25)  solve([x+y+z=3,x-y=z],[x,y,z]);
(%o25)  [[x=3/2,y=-(2%r1-3)/2,z=%r1]]
```

¿Qué pasa si el sistema de ecuaciones no tiene solución? Veamos un ejemplo (de acuerdo, no es muy difícil)

```
(%i26)  solve([x+y=0,x+y=1],[x,y]);
        []
```

¿Y si todos los valores de x cumplen la ecuación?

```
(%i27)  solve((x+1)^2=x^2+2x+1,x);
(%o27)  [x=x]
```

Maxima nos dice que el sistema se reduce a $x = x$ que claramente es cierto para todo x. El siguiente caso es similar. Obviamente $(x+y)^2 = x^2 + 2xy + y^2$. ¿Qué dice al respecto *Maxima*?

```
(%i28)  solve((x+y)^2=x^2+2*x*y+y^2,[x,y]);
        Dependent equations eliminated: (1)
(%o28)  [[x=%r3,y=%r2]]
```

En otras palabras, x puede tomar cualquier valor e y lo mismo.

4.2.2 to_poly_solve

Hay una segunda forma de resolver ecuaciones en *Maxima*. puedes acceder a ella desde el menú **Ecuaciones→Resolver (to_poly)**. Sin entrar en detalles, algunas ecuaciones las resuelve mejor. Por ejemplo, cuando hay radicales por medio la orden `solve` no siempre funciona bien:

```
(%i29)  solve(3*x=sqrt(x^2+1),x);
(%o29)  [x=sqrt(x^2+1)/3]
```

En cambio, tiene un poco más de éxito

```
(%i30)  to_poly_solve(3*x=sqrt(x^2+1),x);
(%o30)  %union([x=1/2^(3/2)])
```

Como puedes ver, la respuesta es el *conjunto* de soluciones que verifica la ecuación. De ahí la palabra `union` delante de la respuesta.

`to_poly_solve(`*ecuación, variable*`)`	resuelve la ecuación
`to_poly_solve(`*expr, variable*`)`	resuelve la expresión igualada a cero

Además de este ejemplo, hay otras ocasiones en las que la respuesta de `to_poly_solve` es mejor o más completa. Por ejemplo, con funciones trigonométricas ya hemos visto que `solve` no da la lista completa de soluciones

```
(%i31)  solve(x*cos(x));
        solve: using arc-trig functions to get a solution.
        Some solutions will be lost.
(%o31)  [x=0]
```

En cambio, con `to_poly_solve` la respuesta es un poco más amplia

```
(%i32)  to_poly_solve(x*cos(x),x);
```

(%o32) $\%\mathrm{union}\left([x=0],[x=2\pi\%z101-\frac{\pi}{2}],[x=2\pi\%z103+\frac{\pi}{2}]\right)$

Los parámetros "z101" y "z103" indican un número entero arbitrario y la numeración depende del número de operaciones que hayas realizado. No es quizá la forma más elemental de escribir la solución, pero sí que tenemos todas las soluciones de la ecuación. Observa también que en este caso no hemos escrito una ecuación sino una expresión y `to_poly_solve` ha resuelto dicha expresión igualada a cero lo mismo que ocurría con la orden `solve`.

4.2.3 Sistemas de ecuaciones lineales

`linsolve([`*ecuaciones*`],[`*variables*`])`	resuelve el sistema

En el caso particular de sistemas de ecuaciones lineales puede ser conveniente utilizar `linsolve` en lugar de `solve`. Ambas órdenes se utilizan de la misma forma, pero `linsolve` es más eficiente en estos casos. Sólo una observación: sigue siendo importante escribir correctamente qué variables se consideran como incógnitas. El resultado puede ser muy diferente dependiendo de esto.

```
(%i33)  eq:[x+y+z+w=1,x-y+z-w=-2,x+y-w=0]$
(%i34)  linsolve(eq,[x,y,z]);
(%o34)  [x=(4*w-3)/2,y=-(2*w-3)/2,z=1-2*w]
```

¿Cuál es el resultado de `linsolve(eq,[x,y,z,w])`?

4.2.4 Algsys

`algsys([`*ecuaciones*`],[`*variables*`])`	resuelve la ecuación o ecuaciones
`realonly`	si vale `true`, `algsys` muestra sólo soluciones reales

La orden `algsys` resuelve ecuaciones o sistemas de ecuaciones algebraicas. La primera diferencia con la orden `solve` es pequeña: `algsys` siempre tiene como entrada listas, en otras palabras, tenemos que agrupar la ecuación o ecuaciones entre corchetes igual que las incógnitas.

```
(%i35)  eq:x^2-4*x+3;
(%o35)  x^2-4*x+3
(%i36)  algsys([eq],[x]);
(%o36)  [[x=3],[x=1]]
```

La segunda diferencia es que `algsys` intenta resolver numéricamente la ecuación si no es capaz de encontrar la solución exacta.

```
(%i37)  solve(eq:x^6+x+1);
(%o37)  [0=x^6+x+1]
(%i38)  algsys([eq],[x]);
(%o38)  [[x=-1.038380754458461%i-0.15473514449684],
         [x=1.038380754458461%i-0.15473514449684],
         [x=-0.30050692030955%i-0.79066718881442],
         [x=0.30050692030955%i-0.79066718881442],
         [x=0.94540233331126-0.61183669378101%i],
         [x=0.61183669378101%i+0.94540233331126]]
```

En general, para ecuaciones polinómicas `algsys` nos permite algo más de flexibilidad ya que funciona bien con polinomios de grado alto y, además, permite seleccionar las raíces reales. El comportamiento de `algsys` está determinado por la variable `realonly`. Su valor por defecto es `false`. Esto significa que `algsys` muestra todas las raíces. Si su valor es `true` sólo muestra las raíces reales.

```
(%i39)   eq:x^4-1=0$
(%i40)   realonly;
(%o40)   false
(%i41)   algsys([eq],[x]);
(%o41)   [[x=1],[x=-1],[x=%i],[x=-%i]]
(%i42)   realonly:true$
(%i43)   algsys([eq],[x]);
(%o43)   [[x=1],[x=-1]
```

4.3 Ejercicios

Ejercicio 4.1. Calcula los puntos donde se cortan las parábolas $y = x^2$, $y = 2x^2 + ax + b$. Discute todos los casos posibles dependiendo de los valores de a y b.

Ejercicio 4.2. Dibuja, en un mismo gráfico, la elipse de semieje horizontal $a = 3$ y de semieje vertical $b = 5$ y la bisectriz del primer cuadrante. Calcula los puntos donde se cortan ambas curvas.

Ejercicio 4.3. Consideremos la circunferencia de centro $(0, 0)$ y radio 2. Dibújala. Ahora consideremos un rectángulo centrado en el origen e inscrito en ella. Determina el rectángulo así construido cuya área sea 1.

Ejercicio 4.4. Representa gráficamente y determina los puntos de corte de las siguientes curvas:

a) la recta $x - y = 5$ y la parábola $(x - 1)^2 + y = 4$;
b) la hipérbola equilátera y la circunferencia de centro $(-1, 1)$ y radio 1;
c) las circunferencias de centro $(0, 0)$ y radio 2 y la de centro $(-1, 3)$ y radio 3.

Ejercicio 4.5. Resolver la ecuación logarítmica:

$$\log(x) + \log(x + 1) = 3$$