
3 Manipulando expresiones algebraicas

3.1. Transformaciones simples

Habitualmente hay muchas formas de escribir la misma expresión algebraica. Por ejemplo, $1 + 2x + x^2$ se puede escribir también en la forma $(1 + x)^2$.

Mathematica tiene una amplia colección de funciones para convertir expresiones en diferentes formas algebraicas.

- **Expand** desarrolla productos y potencias.

```
Expand[(1 + x)^2]
```

- **Factor** recupera la forma original.

```
Factor[ % ]
```

- Es fácil generar expresiones complejas con **Expand**.

```
Expand[(1 + x + 3 y)^4]
```

- Habitualmente **Factor** origina expresiones más *simples*.

```
Factor[ % ]
```

- Sin embargo, hay casos en los que **Factor** puede originar expresiones más complicadas.

```
Factor[x^10 - 1]
```

- En tales casos, **Expand** obtiene formas más *simples*.

```
Expand[ % ]
```

Las transformaciones hechas por funciones como **Expand** y **Factor** son siempre correctas, independientemente de que los valores que las variables puedan tener. Algunas veces, sin embargo, es útil realizar transformaciones que son correctas únicamente para algunos valores de las variables.

- En general, *Mathematica* no desarrolla, por la ambigüedad, potencias no enteras.

```
p = Sqrt[x y]
```

```
q = Sqrt[(1 + x)^4]
```

- `PowerExpand` fuerza el desarrollo.

```
PowerExpand[q]
```

- Pero conviene tener en cuenta que, generalmente, `PowerExpand` es correcta únicamente para ciertos valores de las variables.

```
PowerExpand[p]
```

3.2. Polinomios

Cuando una expresión tiene una sola variable se puede escribir como suma de términos, como producto, etc. Pero cuando la expresión involucra a varias variables –o variables y parámetros– hay, incluso, una gama más amplia de elección.

- Un polinomio en dos variables.

```
P = Expand[(3 + 2 x)^2 (x + 2 y)^2]
```

- Se puede, por ejemplo, agrupar términos de forma que una de las variables sea *dominante*.

```
Collect[P, x]
```

```
Collect[P, y]
```

- Esto factoriza los términos que no dependen de una variable.

```
FactorTerms[P, y]
```

Además de funciones para transformar las expresiones, *Mathematica* también cuenta con algunas órdenes que permiten obtener diversa información acerca de un polinomio.

- Otro polinomio.

```
P = Expand[(1 + 3x + 4y^2)^2]
```

- Coeficiente de x en el polinomio.

```
Coefficient[P, x]
```

- Coeficiente de y^2 .

```
Coefficient[P, y, 2]
```

- `Coefficient` trabaja incluso aunque el polinomio no esté en forma desarrollada.

```
Coefficient[(1 + 3x + 4y^2)^2, x]
```

- El exponente más alto de la variable y en el polinomio.

```
Exponent[P, y]
```

- El segundo y el cuarto término en el polinomio. Observa el orden en que *Mathematica* lo considera.

```
Part[P, 2]
```

```
Part[P, 4]
```

3.3. Expresiones racionales

Conseguir la forma adecuada de una expresión algebraica complicada tiene mucho de arte. En la mayoría de los casos es conveniente experimentar con diferentes transformaciones hasta que se obtiene el resultado deseado.

Las órdenes `Expand` y `Factor` son las funciones más comunes para tratar con expresiones algebraicas. Sin embargo, en ocasiones, especialmente cuando se trata con expresiones racionales, se pueden necesitar otras funciones.

- Una expresión racional.

```
R = (x - 1)^2 (2 + x) / ((1 + x) (x - 3)^2)
```

- `Expand` desarrolla el numerador, pero deja el denominador en forma factorizada.

```
Expand[R]
```

- `ExpandAll`, por contra, desarrolla el numerador y el denominador.

```
ExpandAll[R]
```

- `Together` une todos los sumandos con un común denominador.

```
Together[%]
```

- `Apart` descompone en fracciones simples.

```
Apart[%]
```

- `Factor` factoriza todo, recuperando, generalmente, la forma original.

```
Factor[%]
```

- Otra expresión racional; en este caso con dos variables.

```
R = (1 + x)/(2 (2 - y))
```

- El numerador y el denominador.

```

In[ ]: Numerator[R]
In[ ]: Denominator[R]

```

- Denominator devuelve el valor 1 para expresiones que no son cocientes.

```

In[ ]: Denominator[1/x + 2/y]

```

3.4. Simplificando expresiones

Hay muchas situaciones en las que conviene escribir una expresión algebraica particular en la forma más simple posible. A pesar de que es difícil saber exactamente qué significa en cada caso la *forma más simple*, un procedimiento habitual es generar algunas formas diferentes y elegir la que tenga el menor número de términos.

- Simplify escribe $x^2 + 2x + 1$ en forma factorizada.

```

In[ ]: Simplify[x^2 + 2x + 1]

```

- Simplify deja $x^{10} - 1$ en forma desarrollada puesto que la forma factorizada es más larga.

```

In[ ]: Simplify[x^10 - 1]

```

- Mathematica no simplifica automáticamente $\sqrt{x^2}$ porque sólo es cierto para algunos valores de x .

```

In[ ]: Simplify[Sqrt[x^2]]

```

- $\sqrt{x^2}$ es igual a x cuando $x \geq 0$, pero no en otro caso.

```

In[ ]: {Sqrt[4^2], Sqrt[(-4)^2]}

```

- Simplificando, suponiendo que $x > 0$.

```

In[ ]: Simplify[Sqrt[x^2], x > 0]

```

- Por ambigüedad, no se puede simplificar automáticamente la expresión siguiente.

```

In[ ]: 2 a + 2 Sqrt[a - Sqrt[-b]] Sqrt[a + Sqrt[-b]]

```

- Sin embargo, si se supone que a y b son positivos, entonces la expresión puede ser simplificada.

```

In[ ]: Simplify[%, a > 0 && b > 0]

```

- Un ejemplo simple con funciones trigonométricas.

```
Simplify[ArcSin[Sin[x]], -Pi/2 < x < Pi/2]
```

- Esto simplifica $\sqrt{x^2}$ suponiendo que x es un número real.

```
Simplify[Sqrt[x^2], Element[x, Reals]]
```

- Esto simplifica suponiendo que n es un número entero.

```
Simplify[Sin[x + 2 n Pi], Element[n, Integers]]
```

Ahora deberías conocer también...

- ...cómo funcionan las órdenes `Expand` y `Factor`;
- ...cuáles son algunas de las órdenes básicas para tratar con polinomios (`Collect`, `Coefficient`, `Exponent`);
- ...cuáles son las órdenes básicas para tratar con expresiones racionales (`Together`, `Apart`, `Numerator`, `Denominator`);
- ...cómo funciona la orden `Simplify`.