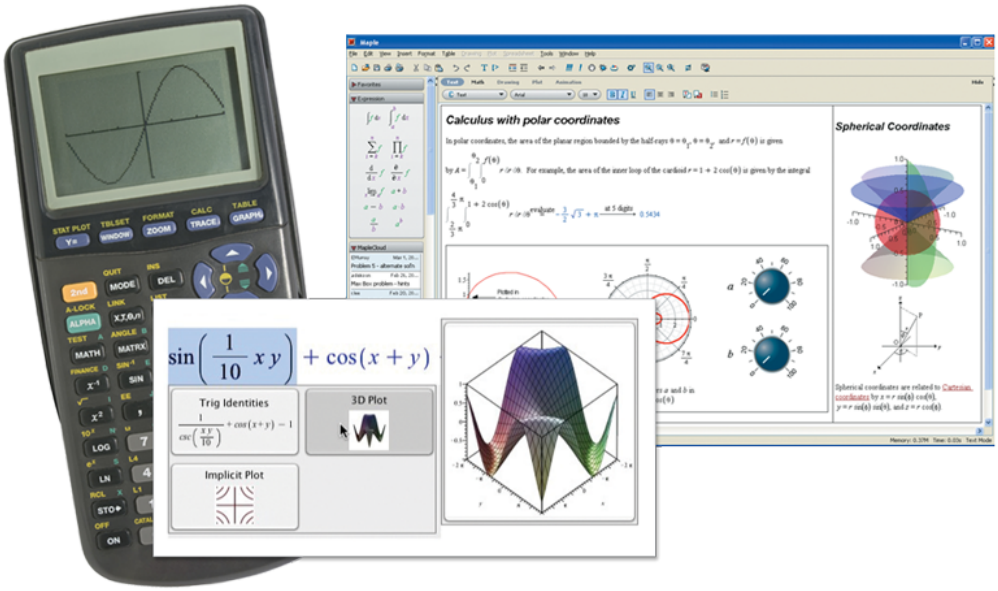
Sistema de cálculo simbólico

asalber@ceu.es
https://aprendeconalf.es



Introducción

Un *sistema de álgebra computacional* (CAS, del inglés computer algebra system) es un programa de ordenador o calculadora avanzada que facilita el cálculo simbólico. La principal diferencia entre un CAS y una calculadora tradicional es la habilidad del primero para trabajar con ecuaciones y fórmulas simbólicamente, en lugar de numéricamente. Esto permite trabajar con expresiones simbólicas (no numéricas) y realizar operaciones como la resolución de ecuaciones, el cálculo de límites, el cálculo de derivadas o el cálculo de integrales.

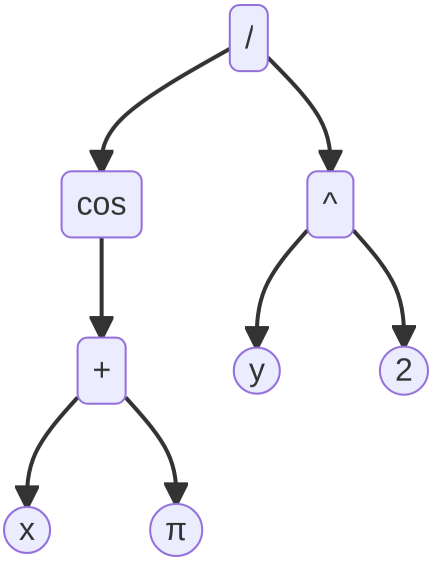
Objetivos

El objetivo de este proyecto es desarrollar un sistema de álgebra computacional para el cálculo simbólico que permita el cálculo de derivadas, la resolución de ecuaciones y la simplificación de expresiones.

Descripción técnica

Para manejar expresiones simbólicas los sistemas de álgebra computacional suelen representarlas mediante un árbol en el que las operaciones forman los nodos y los argumentos

son las ramas. Por ejemplo, la expresión $\frac{\cos(x+\pi)}{y^2}$ se representaría mediante el árbol siguiente



Árbol de la expresión $\frac{\cos(x+\pi)}{y^2}$.

Julia dispone del tipo de datos `Expr` para las expresiones simbólicas. Para definir un símbolo o una expresión simbólica se utiliza el operador `:`.

```
ex = :(sin(x+pi))
typeof(ex)
```

```
Expr
```

La principal diferencia entre una expresión normal y una expresión simbólica es que esta última no se evalúa.

A su vez, el paquete de Julia `TermInterface` proporciona una interfaz genérica para manipular expresiones simbólicas. Las tres principales funciones de este paquete son

- `isexpr(ex)`: Devuelve `true` si `ex` es una expresión simbólica y `false` cuando se trata de un símbolo o un número.
- `operation(ex)`: Devuelve el operador más externo de la expresión simbólica `ex`, es decir, el operador en la raíz del árbol correspondiente a la expresión.

- `arguments(ex)`: Devuelve un vector con los argumentos sobre los que se aplica el operador máx externo de la expresión simbólica `ex`, es decir, un vector con las expresiones simbólicas correspondientes a cada una de las ramas que salen de la raíz del árbol correspondiente a la expresión. Dependiendo de la longitud de este vector se puede averiguar la aridad del operador, es decir, si es unario, binario o en general n-ario.

```
using TermInterface
isexpr(:(x+2))
```

```
true
```

```
operation(:(x+2))
```

```
:+
```

```
arg = arguments(:(x+2))
```

```
2-element view(::Vector{Any}, 2:3) with eltype Any:
  :x
 2
```

```
isa(arg[1], Symbol)
```

```
true
```

```
isa(arg[2], Symbol)
```

```
false
```

Tareas

1. Aprender a manejar expresiones simbólicas en Julia haciendo uso del paquete `TermInterface`.
2. Definir una función para dibujar el árbol correspondiente a una expresión simbólica.
3. Definir una función para calcular la derivada de una expresión simbólica.

4. Definir una función para resolver una ecuación simbólicamente.
5. Definir una función para simplificar una expresión simbólica.
6. Desarrollar una aplicación para una sistema de álgebra computacional que calcule derivadas, resuelva ecuaciones y simplifique expresiones simbólicas.