Ordinary Differential Equations

In this lesson, we will learn about solving first-order and higher-order ordinary differential equations.

WE'LL COVER THE FOLLOWING

- Tools
  - Function
  - Equality
- Steps
  - First-order differentials
  - Higher-order differential equations

One of the most useful functions in modern problem-solving is that the solutions to ordinary differential equations (ODE) have so many varied applications in everything from pure sciences to all sorts of engineering.

$$\frac{dy}{dx} + 5yx^2 = 0$$

Tools

To set up the ODE, we need to refer to the function we are solving and its derivatives. The `Function` and `Eq` classes in SymPy help us do this. Let's discuss these before we move on to solving some equations ourselves.

Function

`Function` is similar to `Symbol` but is used to define a function.

```
f = Function('f')
```

To define a function of symbol `x`, we use the following syntax:

```
x = Symbol('x')
```

```
x = Symbol('x')
f = Function('f')(x)
```

To compute the derivate of the `Function`, we use the `diff()` method:

```
f.diff(x) # first order derivative
f.diff(x, x) # second order derivative
```

Equality

`Eq` is used to define equations in SymPy.

```
Eq(rhs, lhs)
```

The comma separates the right-hand side from the left-hand side.

Now let's use these tools to solve ODEs.

Steps

SymPy can solve several kinds of ordinary differential equations through the `dsolve()` command. Before we solve the ODE, we need to follow these steps:

1. Set up the ODE.
2. Pass it as the first argument of `dsolve()`.
3. Pass the function `f(x)` as the second argument for which we will solve the equation.

The `Function` and `Eq` will help us to set up the equation.

First-order differentials

Let's solve the following first-order differential equation:

$$\frac{dy}{dx} + 5yx^2 = 0$$

```
from sympy import *

x = symbols('x')
y = Function('y')(x)

dydx = y.diff(x)  # taking single derivative
diff_eq = Eq(dydx + 5 * y * x**2, 0)
z = dsolve(diff_eq)
```

```
print(z)
```

The output shows that $y(x)$ has the value $C_1 e^{\left(\frac{-5x^3}{3}\right)}$.

The return type of `dsolve()` is an equality, which might not be useful in some cases. To extract the right-hand side of the equation, we can use the `rhs` property: `equality.rhs`. Let's look at its implementation below:

```
from sympy import *

x = symbols('x')
y = Function('y')(x)

dydx = y.diff(x)  # taking single derivative
diff_eq = Eq(dydx + 5 * y * x**2, 0)
z = dsolve(diff_eq)
print(z.rhs)
```

Higher-order differential equations

ODE problems are essential in computational physics, so we will look at one of the most common examples: the damped harmonic oscillation.

The equation of motion for the damped oscillator is:

$$\frac{d^2x}{dt^2} + 2\zeta\omega_o\frac{dx}{dt} + \omega_o^2 x = 0$$

where x is the position of the oscillator, t is the time, ζ is the damping ratio and ω_o is the oscillator frequency.

```
from sympy import *

t = symbols('t')
x = Function('x')(t)
W = 2 * pi
zeta = 0.5

dxdt = x.diff(t)  # taking single derivative
dx2dt2 = x.diff(t, t)  # taking double derivative, alternatively dx2dt2 = dxdt.diff(t)
diff_eq = Eq((dx2dt2) + (2 * zeta * W * dxdt) + (W**2 * x), 0) # setting up equation
z = dsolve(diff_eq)
```

```
print(z)
```

To compute the value of the function `x` at a certain point, we can use the `subs` function to substitute values in the right-hand side of the equation.

```
from sympy import *

t = symbols('t')
x = Function('x')(t)
W = 2 * pi
zeta = 0.5

dxdt= x.diff(t)  # taking single derivative
dx2dt2 = x.diff(t, t)  # taking double derivative, alternatively dx2dt2 = dxdt.diff(t)
diff_eq = Eq((dx2dt2) + (2 * zeta * W * dxdt) + (W**2 * x), 0)
z = dsolve(diff_eq)
print(z.rhs.subs({'C1': 1, 'C2': 1, t: 5}))
```

Let’s test your understanding with a short quiz.