A small library for first-order initial value problems

Group N

Abstract

We present `ode_flow`, a small Python library for solving and sampling first-order initial value problems of the form $y'(t) = f(t, y(t))$ with $y(t_0) = y_0$. The library is a thin wrapper around `sympy.dsolve` [3], with additional helpers to evaluate the solution at a list of times and to verify that a candidate expression satisfies the ODE. We illustrate the library on three standard problems: exponential growth, constant growth, and quadratic growth. The library follows the Diataxis framework for documentation [4] and the differential equations chapter of the open-source textbook *Python for Mathematics* [2].

1 Introduction

A first-order ordinary differential equation has the form

$$y'(t) = f(t, y(t)),$$

and an initial value problem adds the condition $y(t_0) = y_0$. For many right-hand sides f the solution can be written in closed form by recognising one of the standard cases (separable, linear, exact) [1]. The library `sympy` [3] implements these cases through `sympy.dsolve`, so we treat the solver as the engine and focus on a small wrapper that makes the common operations easy.

We present a three-function library, `ode_flow`, that solves a first-order initial value problem, samples the solution at a list of times, and verifies that a candidate expression satisfies the ODE.

2 The library

The library lives in the single module `ode_flow.py` and follows the differential equations chapter of *Python for Mathematics* [2]: we use `sym.Function` for the unknown, `sym.Eq` for the equation, and `sym.dsolve` with the `ics` keyword for the initial condition.

`solve_initial_value_problem(right_hand_side, time_variable, function, initial_time, initial_value`
builds the equation, calls `sym.dsolve`, and returns the right-hand side of the resulting equation `y(t) = expression`.

`trajectory(solution_expression, time_variable, sample_times)` evaluates the solution at a list of times. Each evaluation is cast to `float` so the output is easy to plot.

`verify_solution(solution_expression, right_hand_side, time_variable, function)` substitutes the candidate into the right-hand side and compares with the derivative of the candidate; the two simplify to the same expression for a genuine solution.

3 Worked examples

The problem $y' = y$ with $y(0) = 1$ has the well-known solution $y(t) = e^t$. The library recovers this exactly; sampling at $t = 0, 1, 2$ gives $1, e, e^2$ cast to floats. The verifier accepts e^t and rejects t.

The problem $y' = 1$ with $y(0) = 0$ has the solution $y(t) = t$. The library returns this in one line.

The problem $y' = 2t$ with $y(0) = 0$ has the solution $y(t) = t^2$. Sampling at the integer times $0, 1, 2, 3$ gives the squares $0, 1, 4, 9$.

The problem $y' = y$ with $y(1) = e$ also has the solution $y(t) = e^t$; the verifier accepts it. This is useful because it confirms that the initial condition does not need to be at $t_0 = 0$.

4 Discussion

`sym.dsolve` can already solve everything in this paper. The library exists because we wanted a shorter interface for the common case (first-order, scalar, initial value problem), and because building the equation and the `ics` dictionary by hand each time is fiddly enough that we preferred to encapsulate it.

The library handles only scalar first-order problems. Systems of ODEs, higher-order equations, and boundary value problems are all natural extensions. A natural next step is to allow the right-hand side to depend on extra parameters and to expose the sweep over those parameters as a separate function.

5 Conclusion

We have presented `ode_flow`, a three-function library for first-order initial value problems. The library recovers the closed-form solutions of three standard problems and confirms them through both sampling and symbolic verification.

References

[1] William E. Boyce and Richard C. DiPrima. *Elementary Differential Equations and Boundary Value Problems*. Wiley, 11 edition, 2017.

[2] Vincent Knight. Python for Mathematics, 2024.

[3] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AmiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, et al. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, 2017.

[4] Daniele Procida. Diátaxis: a systematic framework for technical documentation authoring, 2017.