PHƯƠNG PHÁP MÔ PHỎNG TRONG TỰ ĐỘNG: EULER VÀ RUNGE-KUTTA

Giới thiệu chung

Trong kỹ thuật tự động và điều khiển, nhiều hệ thống động lực được mô tả bằng phương trình vi phân thường không giải được tường minh bằng phương pháp giải tích. Do đó, mô phỏng số (sử dụng máy tính để giải xấp xỉ) là công cụ thiết yếu để dự đoán hành vi hệ thống. Có nhiều phương pháp tính toán số cho bài toán này, trong đó phương pháp Euler và phương pháp Runge-Kutta (đặc biệt biến thể bậc 4) là hai kỹ thuật phổ biến. Phương pháp Euler là một trong những phương pháp cổ điển đơn giản nhất để tính tích phân số của phương trình vi phân. Trong khi đó, họ phương pháp Runge-Kutta cung cấp độ chính xác cao hơn, cho phép nghiệm xấp xỉ hội tụ nhanh đến giá trị thật khi giảm kích thước bước. Dưới đây, chúng ta lần lượt tìm hiểu về phương pháp Euler, phương pháp Runge-Kutta (RK2, RK4), so sánh ưu nhược điểm và gợi ý cách lựa chọn phù hợp cho từng tình huống.

Phương pháp Euler

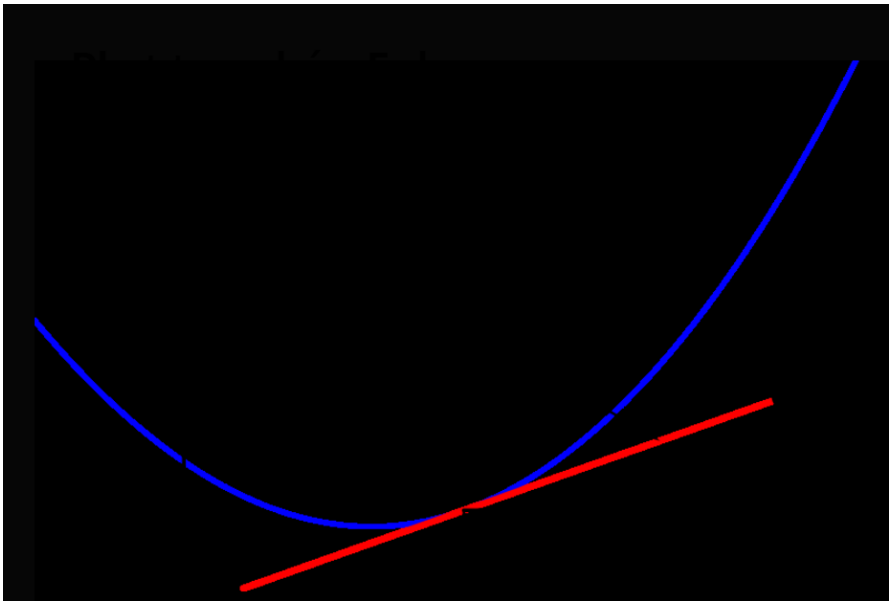
Khái niệm:

Phương pháp Euler là một trong những phương pháp số đơn giản nhất để giải phương trình vi phân thường (ODE). Ý tưởng của phương pháp này là sử dụng giá trị đạo hàm tại một điểm đã biết để ước lượng giá trị của hàm tại điểm tiếp theo.

Nguyên lý tiếp tuyến:

Phương pháp Euler xấp xỉ nghiệm bằng cách lấy đạo hàm tại thời điểm hiện tại làm xấp xỉ cho biến thiên trong tương lai gần. Cụ thể, giả sử bước thời gian nhỏ $\Delta t = h$ đủ nhỏ để $y'(t)$ gần như không đổi trong khoảng đó. Khi đó, sườn dốc (đạo hàm) tại điểm (t_n, y_n) được coi là hằng số trên đoạn $[t_n,

t_{n+1}]$, và ta kéo dài đường thẳng tiếp tuyến tại điểm này để ước tính giá trị y tại thời điểm kế tiếp t_{n+1}. Nói cách khác, Euler tiến sử dụng giá trị độ dốc hiện tại để "bước" tới điểm mới theo phương trình vi phân.



Hình 1: Minh họa một bước của phương pháp Euler. Đường cong màu xanh là nghiệm chính xác của phương trình, còn đoạn thẳng màu đỏ là tiếp tuyến tại (x_0, y_0) – đại diện cho xấp xỉ Euler. Giá trị tại $x_1 = x_0 + h$ được dự báo bởi điểm nằm trên đường tiếp tuyến đỏ. Phương pháp Euler giả định đường cong không đổi độ dốc trong bước ngắn h, do đó nếu hàm thực tế cong lên hoặc xuống nhiều thì đường tiếp tuyến sẽ quá cao hoặc quá thấp so với nghiệm thật (hiện tượng over/undershoot).

Công thức:

Cho phương trình vi phân dạng:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

Với bước thời gian h, phương pháp Euler cập nhật nghiệm xấp xỉ theo công thức truy hồi:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

Trong đó:

- y_n là giá trị xấp xỉ của nghiệm tại thời điểm $t_n = t_0 + n \cdot h$
- $f(t_n, y_n)$ là đạo hàm tại điểm hiện tại
- h là kích thước bước thời gian

Quy trình lặp theo từng bước:

$$\begin{aligned} y_1 &= y_0 + hf(t_0, y_0) \\ y_2 &= y_1 + hf(t_1, y_1) \\ y_3 &= y_2 + hf(t_2, y_2) \\ &\vdots \\ y_{n+1} &= y_n + hf(t_n, y_n) \end{aligned}$$

Ưu điểm:

- ***Đơn giản, dễ triển khai:*** Euler là phương pháp đơn giản nhất để giải phương trình vi phân thường (theo triết lý "KISS" – Keep It Simple, Stupid). Chỉ với một phép tính độ dốc mỗi bước, thuật toán Euler rất dễ hiểu và lập trình. Người dùng có thể thực hiện bằng tay cho vài bước đầu để nắm bắt ý tưởng.
- ***Tính toán ít:*** Mỗi bước chỉ cần tính $f(t,y)$ một lần, nên với cùng kích thước bước, phương pháp Euler tốn ít phép tính hơn so với các phương pháp bậc cao. Điều này hữu ích trong những ứng dụng thời gian thực hoặc hệ thống nhúng có tài nguyên hạn chế (chẳng hạn vi điều khiển chạy vòng lặp điều khiển với bước thời gian rất nhỏ).
- ***Mở rộng tự nhiên:*** Phương pháp Euler có thể mở rộng dễ dàng cho hệ phương trình nhiều ẩn (tích phân từng phương trình thành phần) cũng như áp dụng trong mô phỏng hệ ngẫu nhiên. Ví dụ, Euler được sử dụng làm nền tảng để mô phỏng phương trình vi phân ngẫu nhiên và chuỗi Markov thời gian liên tục nhờ tính đơn giản và rõ ràng của nó

Nhược điểm:

- ***Độ chính xác thấp (bậc 1):*** Euler chỉ đạt bậc 1 về độ chính xác toàn cục, nghĩa là sai số toàn cục tỷ lệ với kích thước bước h. Muốn tăng độ chính xác lên 10 lần, ta cần giảm bước tính xuống 1/10, dẫn đến số bước tăng lên ~10 lần. Thí nghiệm cho thấy khi giảm một nửa độ dài bước thì sai số xấp xỉ của Euler cũng chỉ giảm cỡ một nửa. Do đó, Euler yêu cầu bước rất nhỏ để đạt độ chính xác cao, khiến thời gian mô phỏng kéo dài đáng kể.
- ***Tích lũy sai số và ổn định kém:*** Sai số tại mỗi bước cộng dồn vào các bước tiếp theo, vì y_{n+1} được tính từ y_n xấp xỉ. Nếu bước quá lớn, những sai số này tích lũy nhanh và có thể làm nghiệm số chệch xa nghiệm thực hoặc thậm chí phân kỳ. Ví dụ, Euler dùng bước lớn cho phương trình dao động có thể dự đoán năng lượng tăng dần (phân kỳ) thay vì bảo toàn như lý thuyết. Euler nói chung chỉ ổn định nếu dùng bước rất nhỏ tương ứng với độ thay đổi nhanh nhất của hệ thống (ví dụ hệ cứng đòi hỏi h cực nhỏ, nếu không nghiệm Euler sẽ dao động hoặc nổ số). Ngoài ra, việc lấy xấp xỉ tuyến tính cho hàm có độ cong mạnh sẽ gây hiện tượng over-shoot/under-shoot, như hình minh họa ở trên đã cho thấy (tiếp tuyến đỏ vượt quá hoặc thấp hơn so với đường cong xanh).
- ***Hiệu quả thấp nếu yêu cầu cao:*** Trong nhiều bài toán thực tế, để đạt độ chính xác mong muốn bằng Euler, ta phải chọn h rất nhỏ, dẫn đến hàng nghìn bước tính. Mặc dù mỗi bước tính đơn giản, tổng khối lượng tính toán vẫn lớn và có thể phát sinh lỗi do làm tròn trên máy tính. Nếu chỉ dùng Euler, đôi khi phải đánh đổi giữa tốc độ và độ chính xác: bước nhỏ thì chính xác nhưng chậm, bước lớn thì nhanh nhưng sai số cao.

Ứng dụng trong tự động:

Phương pháp Euler thường được dùng để mô phỏng nhanh các hệ động học đơn giản, kiểm tra ý tưởng hoặc làm nền tảng cho các phương pháp phức tạp hơn.

Phương pháp Runge-Kutta (RK), đặc biệt là RK4

Khái niệm:

Phương pháp Runge-Kutta là một họ các phương pháp giải số cho ODE, trong đó phổ biến nhất là phương pháp Runge-Kutta bậc 4 (RK4). Phương pháp này sử dụng nhiều điểm trung gian trong mỗi bước nhảy để tăng độ chính xác.

Ý tưởng ban đầu:

Kết hợp nhiều độ dốc: Trái với Euler chỉ dùng duy nhất độ dốc tại đầu bước, các phương pháp Runge–Kutta (RK) cải thiện độ chính xác bằng cách lấy mẫu độ dốc tại nhiều điểm trong mỗi bước và kết hợp chúng một cách thông minh. Ý tưởng chung là ước lượng hình dạng của đồ thị nghiệm trong khoảng nhỏ h tốt hơn bằng cách tính đạo hàm tại một số điểm trung gian, qua đó hiệu chỉnh lại dự đoán. Có thể xem các phương pháp RK như mở rộng của công thức Euler bằng cách bổ sung các vị trí đánh giá hàm bên trong bước để xấp xỉ đường cong bằng một đa thức (thay vì đường thẳng).

Runge-Kutta bậc 2 (RK2): Đây là họ các phương pháp sử dụng 2 lần tính đạo hàm mỗi bước, tiêu biểu như phương pháp Euler cải tiến (Heun) hoặc phương pháp điểm giữa (midpoint). Chẳng hạn, phương pháp điểm giữa (một biến thể RK2) tính k_1 tại đầu bước như Euler, sau đó sử dụng k_1 để ước tính giá trị tại giữa bước rồi tính độ dốc k_2 tại đó, cuối cùng lấy độ dốc trung bình $k_{avg} \approx k_2$ (độ dốc tại giữa khoảng) để cập nhật y_{n+1}. Cách làm này đưa thêm thông tin về độ cong của nghiệm vào tính toán, giúp RK2 đạt bậc 2 (sai số toàn cục tỷ lệ h^2) – tức cao hơn Euler một bậc. Có nhiều biến thể cụ thể (Heun, midpoint, Ralston, v.v.), nhưng nhìn chung RK2 đã chính xác hơn Euler đáng kể khi h không quá nhỏ.

Runge-Kutta bậc 4 (RK4): RK4 cổ điển là phương pháp thông dụng nhất trong gia đình RK. Phương pháp này sử dụng 4 lần tính đạo hàm trên mỗi bước để đạt bậc 4 về độ chính xác. Ta tính lần lượt: độ dốc ban đầu k_1 (giống Euler), độ

dốc tại giữa bước sử dụng k_1 (k_2), độ dốc tại giữa bước sử dụng k_2 (k_3), và độ dốc tại cuối bước sử dụng k_3 (k_4). Sau đó kết hợp các độ dốc này theo trọng số thích hợp để cập nhật y. Nhờ việc "lấy mẫu" hàm nhiều điểm, RK4 có thể theo sát đồ thị nghiệm thực hơn trong mỗi bước, giảm thiểu đáng kể lỗi do bỏ qua độ cong so với Euler. (Ngoài RK2 và RK4, cũng tồn tại các RK bậc cao hơn, nhưng RK4 thường được xem là cân bằng tốt giữa độ phức tạp và độ chính xác, do đó rất phổ biến).

Công thức RK4:

Với cùng bài toán như trên, các giá trị trung gian được tính như sau:

- Tính các hệ số trung gian:

$$
\begin{aligned}
k_1 &= f(t_n,\ y_n),\\
k_2 &= f\Big(t_n + \frac{h}{2},\ y_n + \frac{h}{2}k_1\Big),\\
k_3 &= f\Big(t_n + \frac{h}{2},\ y_n + \frac{h}{2}k_2\Big),\\
k_4 &= f(t_n + h,\ y_n + h\,k_3)\,.
\end{aligned}
$$

- Kết hợp các hệ số để cập nhật nghiệm:

$$y_{n+1} \;=\; y_n \;+\; \tfrac{h}{6}\Big(k_1 + 2k_2 + 2k_3 + k_4\Big)\,.$$

Ưu điểm:

- Độ chính xác cao hơn nhiều so với Euler (sai số bậc 4 theo h).
  Giải thích: RK4 đạt bậc 4, nghĩa là sai số toàn cục tiệm cận tỷ lệ h^4. Điều này mang lại độ chính xác vượt trội so với Euler (bậc 1) và cả so với các phương pháp bậc 2. Với cùng bước h, RK4 cho sai số nhỏ hơn rất nhiều; ngược lại, để đạt cùng sai số, RK4 có thể dùng bước lớn hơn đáng kể so với Euler. Ví dụ, nếu dùng bước h thì sai số Euler xấp xỉ tỷ lệ h, còn sai số RK4 tỷ lệ h^4 – giảm bước 2 lần làm sai số Euler giảm ~2 lần, nhưng sai số RK4 giảm đến $2^4=16$ lần. Nhờ đó, RK4 có thể đạt độ

chính xác cao mà không cần bước quá nhỏ, tiết kiệm số bước tính đáng kể. Trong mô phỏng quỹ đạo hay bài toán yêu cầu độ chính xác cao, phương pháp RK4 tỏ ra đặc biệt hiệu quả.

- Ổn định hơn, phù hợp với các bài toán phức tạp, phi tuyến hoặc hệ phương trình.

  Giải thích: Với hầu hết các hệ phi tuyến thông thường, RK4 cho kết quả ổn định hơn Euler khi dùng bước vừa phải. Vì tính đủ "hình dáng" của nghiệm trong khoảng, RK4 ít bị lệch pha hay bùng nổ hơn trong các mô phỏng dài hạn. Thực nghiệm cho thấy RK4 có thể bảo toàn tốt hơn các đại lượng bất biến (như năng lượng trong hệ cơ học) so với Euler, khi bước h không quá lớn. Tuy nhiên, cần lưu ý RK4 vẫn là phương pháp tường minh (explicit), nên với các hệ cực kỳ "cứng" (stiff) thì nó cũng đòi hỏi bước nhỏ để ổn định. Dẫu vậy, trong đa số trường hợp mô phỏng kỹ thuật thông thường, RK4 cho phép chọn h lớn hơn nhiều lần Euler mà kết quả vẫn ổn định và chính xác.

- Được sử dụng rộng rãi trong các phần mềm mô phỏng và điều khiển tự động.

  Giải thích: Phương pháp RK4 trở thành "tiêu chuẩn vàng" trong lĩnh vực tích phân số cho ODE. Nó được sử dụng rất phổ biến nhờ hiệu quả cao, thậm chí nhiều người mặc định nhắc đến "phương pháp Runge-Kutta" là ám chỉ RK4. Nhiều thư viện và phần mềm mô phỏng (Matlab, Octave, Python SciPy…) cung cấp solver RK4 cố định bước như một lựa chọn cơ bản cho người dùng. Ngay cả trong các module mô phỏng thời gian thực (ví dụ Simulink), solver cố định bước kiểu RK4 (ODE4) cũng thường được dùng do cân bằng được giữa tốc độ và độ chính xác. Việc hội tụ nhanh của RK4 khi giảm bước (lỗi giảm nhanh hơn nhiều so với Euler) giúp người dùng tiết kiệm thời gian tính toán nếu đòi hỏi độ chính xác cao.

Nhược điểm:

- Tốn nhiều phép tính hơn mỗi bước so với Euler.

  Giải thích: Để đạt được độ chính xác cao, RK4 phải thực hiện 4 phép tính hàm mỗi bước, nghĩa là tốn tài nguyên CPU hơn ~4 lần so với Euler (mỗi bước). Với các hệ mà mỗi lần tính $f(t,y)$ rất phức tạp (ví dụ mô phỏng một hệ cơ khí nhiều thành phần hoặc mô phỏng mạng nơ-ron lớn), phương pháp RK4 sẽ chậm hơn đáng kể so với Euler nếu dùng cùng bước. Tuy nhiên, lưu ý rằng thường ta có thể dùng bước h lớn hơn đối với RK4 để bù lại số bước ít hơn – hiệu quả tổng thể tùy thuộc bài toán.
- Cần lập trình cẩn thận hơn.

  Giải thích: Thuật toán RK4 gồm nhiều bước tính và nhiều công thức hơn, do đó khó thực hiện thủ công cho nhiều bước và dễ xảy ra nhầm lẫn khi lập trình nếu không cẩn thận. Trong khi Euler có thể trình bày gọn trong một dòng, RK4 yêu cầu quản lý các biến trung gian (k_1, k_2, k_3, k_4) và công thức kết hợp. Với hầu hết kỹ sư ngày nay, điều này không phải trở ngại lớn (máy tính sẽ xử lý), nhưng về phương diện giảng dạy/hiểu biết thì RK4 kém trực quan hơn Euler.

Ứng dụng trong tự động:

Phương pháp Runge-Kutta, đặc biệt là RK4, được sử dụng để mô phỏng các hệ động lực học phức tạp như robot, hệ thống điều khiển, mô phỏng quá trình hóa học, mô hình hóa chuyển động trong không gian v.v.

So sánh hai phương pháp

Tiêu chí	**Phương pháp Euler**	**Phương pháp Runge-Kutta 4 (RK4)**
Bậc chính xác	Bậc 1 – sai số toàn cục $O(h)$ (giảm bước 2× ⇒ sai số giảm ~2×).	Bậc 4 – sai số toàn cục $O(h^4)$ (giảm bước 2× ⇒ sai số giảm ~16×).

Tiêu chí	Phương pháp Euler	Phương pháp Runge-Kutta 4 (RK4)
Số phép tính/ bước	1 lần tính $f(t,y)$ mỗi bước (sử dụng độ dốc đầu bước).	4 lần tính $f(t,y)$ mỗi bước với các $k_1,...,k_4$ trung gian.
Ưu điểm	Rất đơn giản, dễ lập trình; tính toán mỗi bước ít; có thể đủ dùng nếu hệ "dễ" hoặc h rất nhỏ.	Rất chính xác dù dùng bước tương đối lớn; ổn định và tin cậy hơn trong nhiều trường hợp; bước hội tụ nhanh khi giảm h.
Nhược điểm	Sai số lớn nếu bước không đủ nhỏ, lỗi tích lũy dần; dễ mất ổn định nếu dùng bước lớn; phải dùng nhiều bước → tốn thời gian mô phỏng.	Nhiều phép tính hơn (chậm hơn mỗi bước); công thức phức tạp hơn; có thể dư thừa nếu bài toán không đòi hỏi độ chính xác cao.
Tình huống dùng	Bài toán demo, thử nghiệm ban đầu; mô phỏng thời gian thực tốc độ cao (bước rất nhỏ); hệ thống đơn giản, yêu cầu thấp.	Mô phỏng phân tích chuyên sâu, cần kết quả chính xác cao; sử dụng rộng rãi trong các công cụ mô phỏng và tính toán khoa học; thích hợp cho hầu hết các hệ phi "cứng" trong kỹ thuật.

Kết luận

Trong lĩnh vực tự động hóa, cả hai phương pháp đều có vai trò nhất định. Euler thích hợp cho các bài toán đơn giản, kiểm thử nhanh, còn Runge-Kutta (đặc biệt là

RK4) là lựa chọn tiêu chuẩn cho các mô phỏng phức tạp, yêu cầu độ chính xác và ổn định cao.

Tài liệu tham khảo

- Các file PDF bạn đã cung cấp
- TUDelft OCW: Euler's Method
- Lamar University: Euler's Method
- Udemy: Runge-Kutta Method
- UQ: Euler's Method
- UMD: Numerical Integration

BÀI TẬP VÍ DỤ:

1. Ví dụ 1: So sánh Euler và RK4 với phương trình vi phân đơn giản

Phương trình:

$$\frac{dy}{dx} = -2x, \quad y(0) = 1$$

Nghiệm giải tích:

$$y(x) = -x^2 + 1$$

2. Ví dụ 2: Mô phỏng dao động điều hòa

Hệ phương trình:

$$\begin{cases} \frac{dx}{dt} = v \\ \frac{dv}{dt} = -kx \end{cases}$$

Với $x(0) = x_0$, $v(0) = v_0$, k là hệ số đàn hồi.

3. Ví dụ 3: Mô phỏng hệ thống điều khiển

Hệ phương trình:

$$\frac{d^2y}{dt^2} + 2\zeta\omega\frac{dy}{dt} + \omega^2 y = \omega^2 u$$

Chuyển thành hệ bậc nhất:

$$\begin{cases} \frac{dy}{dt} = v \\ \frac{dv}{dt} = \omega^2(u - y) - 2\zeta\omega v \end{cases}$$

Với $y(0) = y_0$, $v(0) = v_0$, u là tín hiệu vào.

4. Ví dụ 4: Mô phỏng sự suy giảm nhiệt độ của cốc cà phê

Phương trình:

$$\frac{dT}{dt} = -k(T - T_a)$$

Với $T(0) = T_0$, T_a là nhiệt độ môi trường, k là hệ số truyền nhiệt.

5. Ví dụ 5: Mô phỏng chuyển động của quả bóng rơi có ma sát không khí

Hệ phương trình:

$$\begin{cases} \frac{dy}{dt} = v \\ \frac{dv}{dt} = -g + \frac{k}{m}v^2 \end{cases}$$

Với $y(0) = h_0$, $v(0) = v_0$, g là gia tốc trọng trường, k là hệ số ma sát, m là khối lượng.

6. Ví dụ 6: Mô phỏng mạch RC đơn giản

Phương trình:

$$\frac{dV}{dt} = \frac{V_{in} - V}{RC}$$

Với $V(0) = V_0$, V_{in} là điện áp nguồn, R là điện trở, C là điện dung.

7. Ví dụ 7: Mô phỏng mô hình tăng trưởng logistic (phổ biến trong sinh học)

Phương trình:

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right)$$

Với $N(0) = N_0$, r là tốc độ tăng trưởng tối đa, K là sức chứa môi trường.