# GIẢI SỐ PHƯƠNG TRÌNH VI PHÂN TRONG MAPLE

SOLVING NUMERICALLY DIFFERENTIAL EQUATIONS IN MAPLE

**Hoàng Ngự Huấn[1,*], Nguyễn Thế Lâm[1]**

**TÓM TẮT**

Áp dụng các chương trình máy tính để giải các bài toán từ lâu đã được chú ý và cho đến nay đã xuất hiện rất nhiều chương trình dành riêng cho lĩnh vực toán học như: Matlab, Mathematica, Maple… Có một thực tế là: cho dù rất hiện đại, các chương trình này vẫn bộc lộ những hạn chế, ví dụ như: chương trình giải bằng phương pháp Euler, thế nhưng bản thân phương pháp này cũng có những dạng biến đổi khác: một trong số chúng là phương pháp Euler hoán cải. Bản thân phương pháp Range-Kutta cũng có các cấp khác nhau và chương trình Maple chỉ chọn một cấp duy nhất. Thêm vào đó các kết quả tính toán không được trình bày dưới dạng bảng, hoặc bảng được ra không theo ý muốn người sử dụng… Từ đây xuất hiện một ý tưởng rất tự nhiên là: ta có thể tận dụng ngay ngôn ngữ lập trình được tạo ra trên chúng để tạo ra các chương trình con riêng để biên tập lại các phương pháp toán học. Điều này rất quan trọng, hữu ích cho giảng viên và sinh viên trong giảng dạy toán cao cấp ở các trường đại học. Trong khuôn khổ bài báo này, chúng tôi sẽ lập các chương trình con để giải số phương trình vi phân thường mà cụ thể chính là giải bài toán Cauchy để phục vụ việc giảng dạy.

***Keywords**: Euler, Euler hoán cải, Range-kutta 2, Runge-Kutta 4.*

**ABSTRACT**

Applying computer program to solve problems has long been noticed, and so far, there have been many programs dedicated to the field of mathematics such as: Matlab, Mathematica, Maple... There is a fact that: although very modern, these programs still show limitations, for example: the program solves by Euler's method, but the method itself also has other transformations such as improved Euler method. The Runge-Kutta method itself has different levels, and the Maple program selects only one. In addition, the calculation results are not presented in the form of a table, or the table is not presented according to the wishes of the user... From here, a very natural idea emerges: we can immediately take advantage of the programming language created on them to create our own subroutines to edit mathematical methods. This is very important, useful for lecturers and students in teaching and studying advanced mathematics in universities. In the framework of this article, we will set up subroutines to solve ordinary differential equations, specifically solving the Cauchy problem for teaching purposes.

***Keywords:** Euler, improved Euler, Runge-Kutta 2, Runge- Kutta 4.*

[1]Khoa Khoa học cơ bản, Trường Đại học Mỏ - Địa chất Hà Nội

[*]Email: huanhoangngu2021@gmail.com; hoangnguhuan@humg.edu.vn

Ngày nhận bài: 10/01/2021

Ngày nhận bài sửa sau phản biện: 17/4/2022

Ngày chấp nhận đăng: 25/4/2022

## 1. GIỚI THIỆU

Bài báo trình bày các phương pháp phương pháp Euler, Euler cải tiến, phương pháp Runge-Kutta 2 và Runge-Kutta 4 giải gần đúng phương trình vi phân thường, sau đó chúng tôi đưa ra ví dụ và lập các chương trình con cho từng phương pháp đã nêu bằng ngôn ngữ Maple để giải các bài toàn một cách tổng quát hơn cho từng phương pháp, kết quả thu được dưới dạng bảng. Phần cuối cùng sẽ có cửa sổ giao diện với người sử dụng, để người sử dụng có thể dễ dàng thao tác, nhập dữ liệu và lựa chọn phương pháp để giải bài toán Cauchy.

## 2. NỘI DUNG

### 2.1. Phương pháp Euler

Đầu tiên ta sẽ giải bài toán Cauchy cho phương trình cấp một bằng phương pháp Euler. Giả sử trên đoạn $[x_0, x_0 + a]$ cần tìm nghiệm của bài toán Cauchy

$$\frac{dy}{dx} = F(x, y), \qquad y(x_0) = y_0 \tag{1}$$

Chia đoạn $[x_0, x_0 + a]$ thành các đoạn nhỏ bởi các điểm chia $x_0, x_1, x_2, \dots, x_n$. Thông thường $\Delta x_n = x_{n+1} - x_n = h$ là hằng số và được gọi là bước nhảy, các điểm $x_i$ được gọi là nút. Trong lời giải số ta đi tìm giá trị xấp xỉ $y_i$ của nghiệm đúng $y(x_i)$ tại các nút được chọn trong đoạn $[x_0, x_0 + a]$.

Trong phương pháp Euler ta giả sử rằng: vế phải $F(x, y)$ của phương trình vi phân trên từng đoạn $[x_i, x_{i+1}]$, $i = 0,1,2, \dots, n$ là hằng số và bằng $F(x_i, y_i)$.

Để lấy tích phân của phương trình (1) trên đoạn $[x_i, x_{i+1}]$, ta phải giải các phương trình của phương pháp Euler sau:

$$x_{i+1} = x_i + h, \qquad y_{i+1} = y_i + hF(x_i, y_i). \tag{2}$$

Ý nghĩa hình học của phương pháp: trên đoạn $[x_i, x_{i+1}]$ coi rằng đạo hàm $y'(x)$ của hàm cần tìm là hằng số, tức là đường tích phân là đoạn thẳng. Ví dụ, trên đoạn $[x_0, x_1]$ hệ số góc của đường thẳng bằng $F(x_0, y_0)$, tức là nghiệm được thế bằng đoạn thẳng của tiếp tuyến của đường tích phân tại điểm $(x_0, y_0)$.

Ta có thể lập chương trình con để tính toán theo sơ đồ trên như sau:

```
> Euler := proc( ode, ic, domain, N )
  local h, i, t, y, F, L, X;
  t := lhs(domain);
  y := op(0,lhs(ic));
  h := ( op(2,rhs(domain))-op(1,rhs(domain)) )/N;
  solve(ode,diff(y(t),t)):
  F := unapply(%,(t,y));
```

```
X := evalf( [ op(lhs(ic)), rhs(ic) ] );
L := X;
for i from 1 to N do
 X := X + [ h, h*F(op(X)) ];
 L := L, X;
end do;
return matrix(N+2,2,[[t,y],L])
end proc:
```

Các tham số nhập vào của chương trình con là: biến ***ode*** chứa phương trình vi phân; biến ***ic*** chứa điều kiện ban đầu dưới dạng đẳng thức $y(x_0) = y_0$; biến domain - miền biến đổi của biến độc lập được cho dưới dạng $x = x_0 \ldots x_{max}$; biến N là số lượng các điểm chia. Chương trình sẽ cho bằng giá trị $(x, y)$.

**2.2. Phương pháp Euler hoán cải**

Phương pháp Euler (2) rất đơn giản, nhưng điểm yếu cơ bản của nó là: sai số tăng rất nhanh. Khi thay thế một đoạn của đường tích phân thực bằng đoạn thẳng của tiếp tuyến của nó, thì sau một khoảng với độ dài h ta sẽ tách ra hơi chệch một chút so với đường cong. Tiếp theo, khi tính giá trị đạo hàm ở điểm mới, ta đã có một sai số nào đó giữa giá trị thực và kết quả tính toán. Thêm vào đó, lại có một sai số nữa xuất hiện khi ta thay đoạn cong bằng một đoạn thẳng… Kết quả là: sự chính xác thu được là không cao. Khi giảm độ dài khoảng, tính chính xác tăng lên, nhưng không nhiều; có thể coi rằng: sai số tỷ lệ thuận với độ dài khoảng h và phương pháp này là phương pháp với độ chính xác cấp một.

Có nhiều cách để cải tiến phương pháp Euler và một trong những sơ đồ cải tiến để giải bài toán (1) có dạng:

$$\begin{aligned} x_{n+1} &= x_n + h \\ y_{n+\frac{1}{2}} &= y_n + \frac{h}{2} F(x_n, y_n) \\ y_{n+1} &= y_n + hF\left(x_n + \frac{h}{2}, y_{n+\frac{1}{2}}\right) \end{aligned} \quad (3)$$

trong đó, h là bước nhảy. Ta gọi phương pháp (3) là *phương pháp Euler hoán cải*.

Sau đây là chương trình con cụ thể hóa phương pháp Euler hoán cải:

```
> ImprEuler := proc( ode, ic, domain, N )
  local h, i, t, y, F, L, X, X1, X2;
   t:=lhs(domain);
   y:=op(0,lhs(ic));
   h:=(op(2,rhs(domain))-op(1,rhs(domain)))/N;
   solve(ode,diff(y(t),t)):
   F:=unapply(%,(t,y));
   X:=evalf([op(lhs(ic)), rhs(ic)]);
   L:=X;
   for i from 1 to N do
    X1:=X + [h, h*F(op(X))];
    X2:=X + [h, h*F(op(X1))];
    X:=(X1 + X2)/2;
    L:=L, X;
   end do;
   return matrix(N+2,2,[[t,y],L])
  end proc:
```

Ở đây, các tham số nhập vào cũng giống như chương trình con Euler.

**2.3. Phương pháp Runge-Kutta cấp 2**

Xét thêm một nhóm các phương pháp phổ biến giải bài toán Cauchy (1) đối với phương trình vi phân thường - phương pháp Runge-Kutta.

Biểu diễn nghiệm cần tìm $y = y(x)$ của bài toán (1) trong lân cận của từng điểm $x = x_n$, $n = 0,1,2, \ldots$ bằng chuỗi Taylor và tính $y(x_{n+1})$. Khi đó, nếu đặt $x_{n+1} - x_n = h$, ta thu được đẳng thức sau:

$$y(x_{n+1}) = y(x_n) + h\frac{dy}{dx}\Big|_{x=x_n} + \frac{h^2}{2!}\frac{d^2y}{dx^2}\Big|_{x=x_n} + \frac{h^3}{3!}\frac{d^3y}{dx^3}\Big|_{x=x_n} + \frac{h^4}{4!}\frac{d^4y}{dx^4}\Big|_{x=x_n} + \cdots$$

Tùy thuộc vào số lượng các số hạng trong công thức cuối mà ta thu được sơ đồ tính toán với độ chính xác khác nhau. Lưu ý là nếu trong công thức khai triển ta chỉ giữ các số hạng tới cấp một, thì ta thu được sơ đồ tính toán của phương pháp Euler. Như vậy, phương pháp Euler chỉ là trường hợp riêng của phương pháp Runge-Kutta.

**2.4. Phương pháp Runge-Kutta cấp 4**

Xét thêm một sơ đồ tính toán phổ biến nữa: phương pháp Runge-Kutte cấp bốn. Nó có công thức tính xấp xỉ là:

$$\begin{aligned} k_1 &= F(x_n, y_n), \\ k_2 &= F\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right), \\ k_3 &= F\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right), \\ k_4 &= F(x_n + h, y_n + hk_3), \\ y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned} \quad (4)$$

Chương trình con của phương pháp Runge-Kutta bậc 4 như sau:

```
> RungeKutta4 := proc (ode, ic, domain, N)
local h, i, t, y, F, L, X, X1, X2, X3, X4, X5, X6, X7, X8;
 t := lhs(domain); y := op(0, lhs(ic));
 h := (op(2, rhs(domain))-op(1, rhs(domain)))/N;
 solve(ode, diff(y(t), t));
 F := unapply(%, t, y);
 X := evalf([op(lhs(ic)), rhs(ic)]);
 L := X;
 for i to N do X1 := [h, h*F(op(X))];
```

```
X2 := X+(1/2)*X1;
X3 := [h, h*F(op(X2))];
X4 := X+(1/2)*X3;
X5 := [h, h*F(op(X4))];
X6 := X+X5;
X7 := [h, h*F(op(X5))];
X8 := [X[1]+h, X[2]+(1/6)*X1[2]+(1/3)*X3[2]+(1/3)*X5[2]+ (1/6)*X7[2]];
X := evalf(X8);
L := L, X
end do;
return matrix(N+2, 2, [[t, y], L])
end proc;
```

Để minh họa, ta sẽ dùng chương trình của phương pháp Range-Kutta cấp 4 để giải xấp xỉ bài toán sau:

**Bài toán**: tìm nghiệm xấp xỉ của phương trình vi phân cấp 1

$$\frac{dy}{dx} = x + \cos\left(\frac{y}{\sqrt{7}}\right), \qquad y(0,5) = 0,6$$

trên đoạn [0,5; 1,5] với số điểm chia là 10.

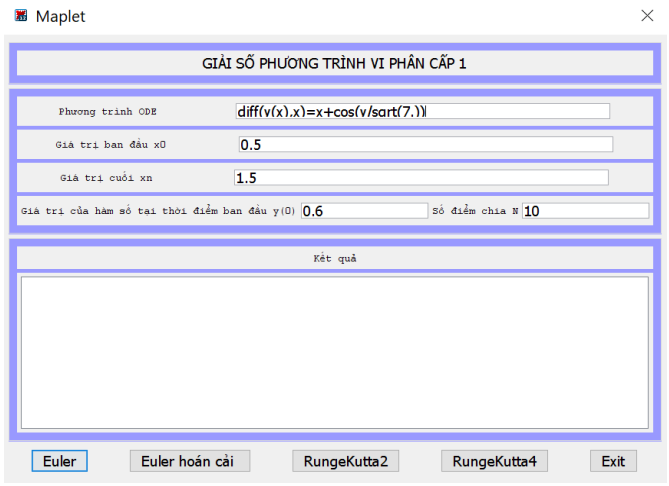
```
> RungeKutta4(ode1, ic1, domain, 10);
```

| x | y |
|---|---|
| 0,5 | 0,6 |
| 0,6000000000 | 0,7440558629 |
| 0,7000000000 | 0,8951701530 |
| 0,8000000000 | 1,053018965 |
| 0,9000000000 | 1,217252340 |
| 1,000000000 | 1,387498091 |
| 1,100000000 | 1,563366427 |
| 1,200000000 | 1,744455260 |
| 1,300000000 | 1,930356052 |
| 1,400000000 | 2,120660046 |
| 1,500000000 | 2,314964675 |

Cuối cùng, ta có thể sử dụng gói lệnh Maplets để ghép tất cả 4 phương pháp trên vào một cửa sổ để giải như sau:

```
restart;
Digits := 9;
with(Maplets[Elements]):
with(DEtools):
with(linalg):
with(Maplets[Tools]):
Euler := proc (ode, x0, xn, y0, N)
global h, i, t, y, F, L, X, kq;
h := (xn-x0)/N;
solve(ode, diff(y(x), x));
F := unapply(%, x, y);
L := [x, y];
X := evalf([x0, y0]);
L := stackmatrix(L, X);
t := array(1 .. N);
for i to N do X := X+[h, h*F(X[1], X[2])];
L := stackmatrix(L, X);
t[i] := X end do;
kq := linalg[stackmatrix](L);
Maplets[Tools][Set](('vp')(value) = Matrix([[x, y], [x0, y0], seq(t[i], i = 1 .. N)]))
end proc:
ImprEuler := proc (ode, x0, xn, y0, N)
global h, i, t, y, F, L, X, X1, X2, X3, X4, kq;
h := (xn-x0)/N;
solve(ode, diff(y(x), x));
F := unapply(%, x, y);
L := [x, y];
X := evalf([x0, y0]);
L := stackmatrix(L, X);
t := array(1 .. N);
for i to N do
X1 := X+[(1/2)*h, (1/2)*h*F(op(X))];
X := X+[h, h*F(op(X1))];
L := stackmatrix(L, X);
t[i] := X
end do;
kq := linalg[stackmatrix](L);
Maplets[Tools][Set](('vp')(value) = Matrix([[x, y], [x0, y0], seq(t[i], i = 1 .. N)]))
end proc:
    RungeKutta2 := proc( ode, ic, domain, N )
    t := lhs(domain);
    y := op(0, lhs(ic));
    h := (op(2, rhs(domain))-op(1, rhs(domain)))/N;
    solve(ode, diff(y(t), t)):
    F := unapply(%, t, y);
X := evalf([op(lhs(ic)), rhs(ic)]);
L := X;
for i from 1 to N do
F := F(op(X));
X1 := X+[h, h*F];
X2 := X+[h, h*F(op(X1))];
X := (X1+X2)*(1/2);
L := stackmatrix(L, X);
t[i] := X;
end do;
kq := linalg[stackmatrix](L);
Maplets[Tools][Set](('vp')(value) = Matrix([[x, y], [x0, y0], seq(t[i], i = 1 .. N)]));
    end proc:
    RungeKutta4 := proc (ode, x0, xn, y0, N)
global h, i, t, y, F, L, X, X1, X2, X3, X4, X5, X6, X7, X8, kq;
h := (xn-x0)/N;
solve(ode, diff(y(x), x));
F := unapply(%, x, y);
L := [x, y];
X := evalf([x0, y0]);
L := stackmatrix(L, X);
t := array(1 .. N);
for i to N do
X1 := [h, h*F(op(X))];
```

```
X2 := X+(1/2)*X1;
X3 := [h, h*F(op(X2))];
X4 := X+(1/2)*X3;
X5 := [h, h*F(op(X4))];
X6 := X+X5;
X7 := [h, h*F(op(X5))];
X8 := [X[1]+h,
X[2]+(1/6)*X1[2]+(1/3)*X3[2]+(1/3)*X5[2]+(1/6)*X7[2]];
X := evalf(X8);
L := stackmatrix(L, X);
t[i] := X end do;
kq := linalg[stackmatrix](L);
Maplets[Tools][Set](('vp')(value) = Matrix([[x, y], [x0, y0],
seq(t[i], i = 1 .. N)]))
end proc:
maplet := Maplet(Window([BoxColumn(border = true,
background = "#9999FF", ["Phuong trinh giai so"]),
BoxColumn(border = true, background = "#9999FF",
[Label("Phuong trinh ODE", 'font' = Font("courier", 14)),
TextField['ode'](width = 30,
"diff(y(x),x)=x+cos(y/sqrt(7.))")], [Label("Gia tri ban dau
x0", 'font' = Font("courier", 14)), TextField['x0'](width =
30, "0.5")], [Label("Gia tri cuoi xn", 'font' = Font("courier",
14)), TextField['xn'](width = 30, "1.5")], [Label("Gia tri cua
ham so tai thoi diem ban dau y(0)", 'font' =
Font("courier", 14)), TextField['y0'](width = 10, "0.6"),
Label("So diem chia N", 'font' = Font("courier", 14)),
TextField['N'](width = 10, "10")]), BoxColumn(border =
true, background = "#9999FF", [Label("Ket qua", 'font' =
Font("courier", 14))], [MathMLViewer[vp](height = 200)]),
[Button("Euler", Evaluate('function' = 'Euler(ode, x0, xn,
y0, N)')), Button("Euler Hoan cai ", Evaluate('function' =
'ImprEuler(ode, x0, xn, y0, N)')), Button("RungeKutta4",
Evaluate('function' = 'RungeKutta4(ode, x0, xn, y0, N)')),
Button("Exit", Shutdown())]])):
Maplets[Display](maplet):
stackmatrix(kq);
```




## 3. KẾT LUẬN

Như vậy ta đã lập được tổng cộng 4 chương trình con, cụ thể hóa các phương pháp giải số: phương pháp Euler, phương pháp Euler hoán cải (ImprEuler) và phương pháp Runge-Kutta cấp hai và cấp 4. Sau đó ta lần lượt áp dụng các chương trình con trên để giải số cụ thể hai bài toán. Kết quả thu được dưới dạng bảng và có thể dùng trực tiếp vào việc trình chiếu phục vụ bài giảng cho sinh viên.

Với những kết quả thu được ta có thể mở rộng cho những phương pháp khác như phương pháp sai phân hữu hạn, hoặc phương pháp bắn để giải bài toán biên.

**LỜI CẢM ƠN**

Các tác giả xin được cảm ơn Trường Đại học Mỏ - Địa chất đã hỗ trợ bài báo thông qua đề tài Khoa học công nghệ cấp cơ sở năm 2022 với mã số T22-17.

**TÀI LIỆU THAM KHẢO**

[1]. Nguyen Huu Dien, 2015. Thuc hanh tinh toan trong Maple. Vietnam National University Press, Hanoi.

[2]. Ta Van Dinh. *Phuong phap tinh (Dung cho cac truong dai hoc ky thuat)*. Vietnam Education Publishing House, Hanoi.

[3]. Matrosov A.V., 2001. *Maple 6. Solving Problems of Higher Mathematics and Mechanics*. St. Peterburg,. BHV-Peterburg. (In Russian).

[4]. Goloskokov D. P., 2004. *The Equations of Mathematical Physics: Solving problems in the Maple System*. St. Peterburg, Piter. (In Russian).

**AUTHORS INFORMATION**

**Hoang Ngu Huan, Nguyen The Lam**

Faculty of Basic Sciences, Hanoi University of Mining and Geology