HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Chapter 6: Ordinary Differential Equation

Scientific Computing

SoICT 2023

ONE LOVE. ONE FUTURE.

Contents

- Differential Equations
- Euler method
- Runge-Kutta method
- MATLAB functions

Differential Equations

- Finding solutions of differential equations is usually divided into two types (initial value problems and boundary condition problems) depending on whether we need to find solutions that satisfy initial or boundary conditions.
- Most initial value problems describe systems that are considered time-dependent, and the solution of the problem depends on the initial condition.

Input Value Problem

- The initial value problem (IVP) for the first order differential equations can be written as:

$$y'(t) = f(y,t); y(t_0) = y_0$$

where y' is the first derivative of y, f(y,t) is a function of two variables y and t, $y(t_0)=y_0$ is the initial condition of the problem.

- If f does not depend on y, then y' can be calculated by integrating the function f.

- If f depends on y?

Input Value Problem: Example

- E.g. 1: Population growth rate depends on population. If the population at time t is y(t), then the population growth rate at time t is

$$y'(t) = k*y(t)$$

where k is a row of positive numbers.

- E.g. 2: Lotka-Voltera equation about predator (fox) and prey (rabbit). Let F(t) be the number of foxes and R(t) be the number of rabbits at time t, we have:

$$R'(t) = (a - bF)R$$

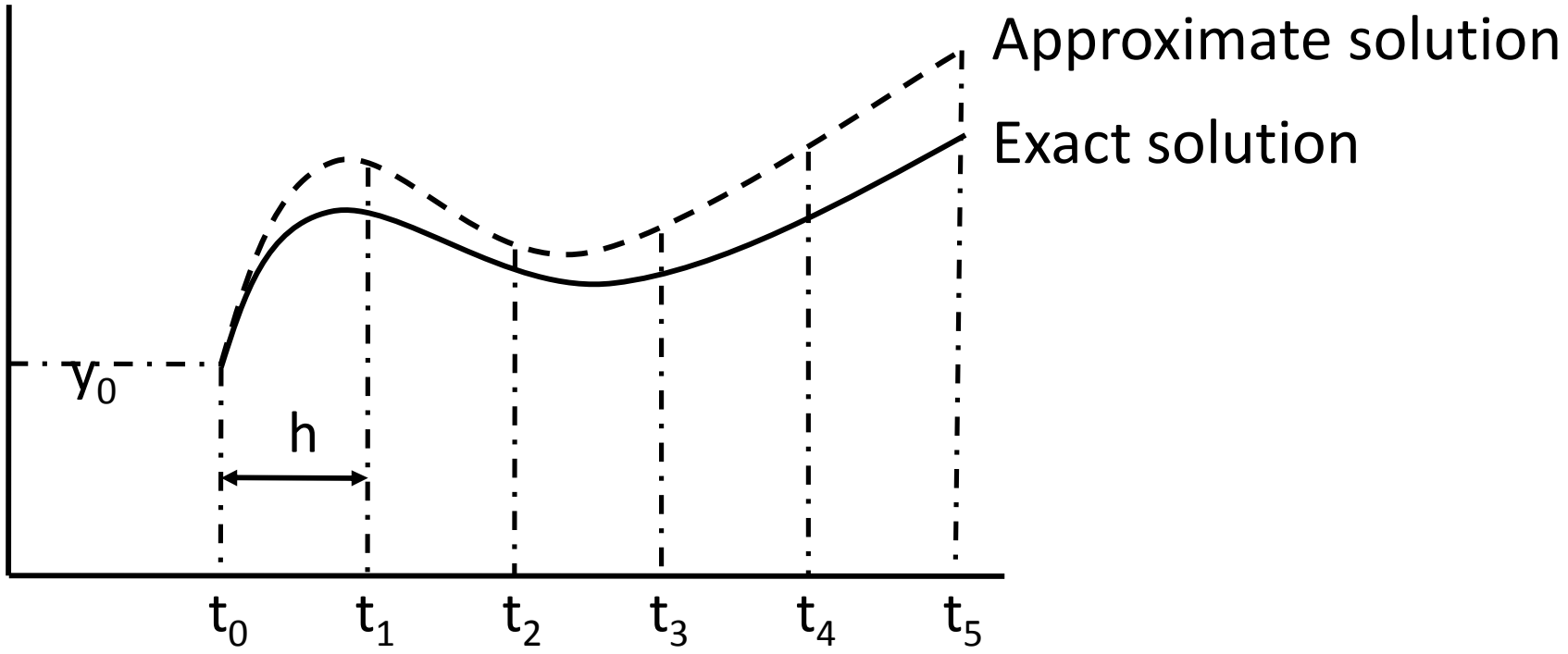
$$F'(t) = (cR - d)F$$

where a, b, c, d are positive constants

The above differential equations are very difficult to solve by analytical methods.

Numerical For Differential Equations

The numerical method requires calculating the value at the time grid- point: $t_n = t_{n-1} + h$; $n = 1, 2, \ldots$; h is the step length.



The Existence and Uniqueness of solution

- **Theorem 1**: If f is a continuous function on the rectangle:

$$R = \{(t,y) : |t - t_0| \leq \alpha, |y - y_0| \leq \beta\}$$

then IPV has a solution y(t) with $|t - t_0| \leq \min\{\alpha, \beta/M\}$,

where $M = \max\{|f(t,y)| : (t,y) \in R\}$.

- **Theorem 2**: If both f and $\partial f/\partial y$ are continuous on a rectangle

$$R = \{(t,y) : |t - t_0| \leq \alpha, |y - y_0| \leq \beta\}$$

then IPV has a unique solution y(t) with $|t - t_0| \leq \min\{\alpha, \beta/M\}$, where $M = \max\{|f(t,y)| : (t,y) \in R\}$.

- **Theorem 3**: Suppose t_0 is in the interval [a,b]. If f is continuous with $a \leq t \leq b$, $-\infty \leq y \leq \infty$ and y is Liptchitz continuous, that is, we can find a constant L such that for all y_1, y_2 and t in [a,b] we have

$$|f(t,y_2) - f(t,y1)| \leq L|y_2 - y1|$$

then IVP has a unique solution y(t) on the interval [a,b].

Forward Euler (FE)

- Considering differential equation: y' = f(y,t)

Forward Euler method is obtained by using the forward difference formula to approximate y':

$$\frac{y_{n+1} - y_n}{h} \approx y_n' = f(y_n, t_n) \qquad (1)$$

$$(1) \Rightarrow y_{n+1} = y_n + hf(y_n, t_n) \qquad (2)$$

- We can rewrite (2) in an iterative form as follows:

$$y_1 = y_0 + hf(y_0, t_0)$$
$$y_2 = y_1 + hf(y_1, t_1)$$
$$\ldots\ldots \qquad (3)$$
$$y_n = y_{n-1} + hf(y_{n-1}, t_{n-1})$$

FE: Example

- Consider differential equation:

$$y' = -20y + 7e^{-0.5t},$$

$$y(0) = 5$$

- Solve above differential equation with t in [0, 0.04],

$$h = 10^{-2}; 10^{-3}; 10^{-4}$$

- Estimate the error, given that the exact solution of is:

$$y = 5e^{-20t} + (7/19.5)(e^{-0.5t} - e^{-20t})$$

FE: Example

t	h = 0.01		h = 0.001		h = 0.0001	
	Result	Error	Result	Error	Result	Error
0.01	4.07000	0.08693	4.14924	0.0769	4.15617	0.0076
0.02	3.32565	0.14072	3.45379	0.1259	3.46513	0.0124
0.03	2.72982	0.17085	2.88524	0.1544	2.89915	0.0153
0.04	1.87087	0.18440	2.42037	0.1684	2.43554	0.0167

Forward Euler

- FE method is simple, but it has two disadvantages:
  - Large rounding error.
  - Instability occurs when the time constant of the equation is negative, unless the time step h is small enough.
- E.g: consider $y' = -\alpha y$, $y(0) = y_0$ where $y_0 > 0$, $\alpha > 0$.

The exact solution of the problem is: $y = y_0 e^{-\alpha t}$, y goes to 0 as t goes to infinity.

Forward Euler

- If you solve it with FE method, then:
  - If $\alpha h < 1$ then the solution is minimized and positive
  - If $\alpha h > 1$, the sign of the solution is alternate.

  Especially if $\alpha h > 2$ then the amplitude of the solution increases with each step, and the solution fluctuates.

  => Unstable

FE for a System of Differential Equations

- Consider a system of Differential Equations :

$$y' = f(y,z,t),\ y(0) = y_0$$

$$z' = g(y,z,t),\ z(0) = z_0 \qquad (5)$$

- Forward Euler for a system of Differential Equations:

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

$$z_{n+1} = z_n + h\, g(y_n,z_n,t_n) \qquad (6)$$

FE for a Second order Differential Equation

- To solve higher-order differential equation, we can decompose it into a system of first-order differential equations.
- For example: Consider second order differential equation:

$$y''(t) - 0.05\, y'(t) + 0.15\, y(t) = 0$$

$$y'(0) = 0 \qquad (7)$$

$$y(0) = 1$$

Let $y' = z$ then rewrite (7) as following

$$y' = z, \qquad y(0) = 1,$$

$$z' = 0.05z - 0.15\, y, \qquad z(0) = 0 \qquad (8)$$

(8) becomes a system of first-order differential equations

Modified Euler method

- Modified Euler method is more accurate and stable than FE method. Modified FE bases on trapezoidal rule to calculate integral y'=y(t):

$$y_{n+1} = y_n + h/2\,[f(y_{n+1}, t_{n+1}) + f(y_n, t_n)] \qquad (8)$$

- If f is linear with y then (8) is linear with y_{n+1}, so we can easily determine y_{n+1}
- If f is nonlinear with y then (8) is nonlinear y_{n+1}. Finding y_{n+1} is similar to solving nonlinear equations as studied in Chapter 4.

Modified Euler method

- Use modified FE to solve differential equation with h=0.1:

$$y' = -y^{1.5} + 1, \quad y(0) = 10, \text{ with } 0 \le t \le 1$$

- Applying modified FE:

$$y_{n+1} = y_n + h/2\,[-(y_{n+1})^{1.5} - (y_n)^{1.5} + 2] \quad (9)$$

with n = 0:

$$y_1 = y_0 + h/2\,[-(y_1)^{1.5} - (y_0)^{1.5} + 2]$$

The best approximation for y_1 on the right-hand side is y_0.

Let $y_1 = y_0$:

$$y_1 = y_0 + h/2\,[-(y_0)^{1.5} - (y_0)^{1.5} + 2] \quad (10)$$

- Similarly, we can calculate y_n

Backward Euler (BE)

- Cónider y' = f(y,t), Backward Euler is obtained by using the backward difference formula to approximate y':

$$\frac{y_{n+1} - y_n}{h} \approx y'_{n+1} = f(y_{n+1}, t_{n+1}) \qquad (1)$$

$$(1) \Rightarrow \qquad y_{n+1} = y_n + hf(y_{n+1}, t_{n+1}) \qquad (2)$$

- Note: (2) does not give us the explicit formula because we have to calculate the value of the function f for the unknown argument y_{n+1}
- To find y_{n+1} we can solve (2) as a nonlinear equation as described in chapter 4.

BE: Example

- Consider $y' = y^3$, $y(0) = 1$.

Applying BE with h=0.5:

$$y_1 = y_0 + h\, f(y_1,t_1) = 1 + 0.5\, (y_1)^3 \qquad (3)$$

(3) can be solved by Newton iterative method:

$$y_k = y_{k-1} - f(y_{k-1})/f'(y_{k-1})$$

MATLAB function:

```
fzero('x+0.5*x^3-1',1)
```

Summary of Euler method

- The Forward Euler method is based on the forward difference approximation. Its error within an iteration is proportional to h^2 and its global error is proportional to h. This method is simple but has a large error and high instability.
- The modified Euler method is based on the trapezoid rule. Its error within an iteration is proportional to h^3 and the global error is proportional to h^2.
- The Backward Euler method is based on backward difference approximation. Its error is similar to the Forward Euler method. However, this method is stable, so it is used to solve non-smooth problems (which are difficult to solve by other methods).

Runge-Kutta method (RK)

- The disadvantage of the Euler methods is that the order of accuracy is small. High accuracy requires h to be very small.
- In Runge-Kutta method, the order of accuracy is increased by using the intermediate points in each iteration.
- Consider: $y' = f(y,t), \qquad y(0) = y_0 \qquad (1)$

  To calculate y_{n+1} at $t_{n+1} = t_n + h$ with y_n known, we integrate the above equation in the interval $[t_n, t_{n+1}]$ as follows :

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(y,t)dt \qquad (2)$$

  The Runge-Kutta method was developed by applying integral methods to integrate the right-hand side of (2).

The Second order Runge-Kutta method

- We apply the trapezoid rule to the right-hand side of (2) as follows:

$$\int_{t_n}^{t_{n+1}} f(y,t)dt = \frac{1}{2}h\left[f(y_n,t_n) + f(y_{n+1},t_{n+1})\right] \quad (3)$$

- In (3), y_{n+1} is unknown. Thus, the second term is approximated by $f(\bar{y}_{n+1},t_{n+1})$, where $\bar{y}_{n+1}$ is the first estimate of y_{n+1} computed according to the Forward Euler.
- The method obtained in this way is called the second order Runge-Kutta formula

The Second order Runge-Kutta method

- The second order Runge-Kutta formula :

$$\overline{y_{n+1}} = y_n + hf(y_n, t_n)$$
$$y_{n+1} = y_n + \frac{h}{2}\left[f(y_n, t_n) + f(\overline{y_{n+1}}, t_{n+1})\right] \quad (4)$$

Or we can rewrite as:

$$k_1 = hf(y_n, t_n)$$
$$k_2 = hf(y_n + k_1, t_{n+1}) \quad (5)$$
$$y_{n+1} = y_n + \frac{1}{2}\left[k_1 + k_2\right]$$

The Second order Runge-Kutta method

- The second order Runge-Kutta method is equivalent to the modified Euler method applied with only one iteration.
- The accuracy order of the second-order Runge-Kutta PP is h^2, which equal to the modified Euler method with the condition that the iterative procedure for solving nonlinear equations in it is convergent.

Thus, using the second order Runge-Kutta method with a sufficiently small h is better than using modified Euler method.

- Using the second order Runge-Kutta method is quite simple.

RK for second order differential equation

- Consider second order differential equation:

$$y''(t) + a\,y'(t) + b\,y(t) = q(t)$$

$$y(0) = 1,\ y'(0) = 0, \qquad (6)$$

where a and b are constants, q(t) known.

Let $z=y'(t)$, we have:

$$y' = z, \qquad y(0) = 1,$$

$$z' = q(t) - a\,z - b\,y, \qquad z(0) = 0 \qquad (7)$$

RK for second order differential equation

- Applying the Second order Runge-Kutta method:

$$k_1 = h\, z_n$$

$$l_1 = h\,(q_n - a\, z_n - b\, y_n)$$

$$k_2 = h\,(z_n + l_1)$$

$$l_2 = h\,(q_{n+1} - a\,(z_n + l_1) - b\,(y_n + k_1) \qquad (8)$$

$$y_{n+1} = y_n + \tfrac{1}{2}\,(k_1 + k_2)$$

$$z_{n+1} = z_n + \tfrac{1}{2}\,(l_1 + l_2)$$

The Third order Runge-Kutta method

- The 3rd order Runge-Kutta method is based on applying a higher-accuracy order integral to the 2nd term in equation (2).

  Using Simson1/3 formula, (2) is approximated by:

$$y_{n+1} = y_n + h/6\,[f(y_n,t_n) + 4f(\bar{y}_{n+1/2}, t_{n+1/2}) + f(\bar{y}_{n+1}, t_1)] \quad (9)$$

  where $\bar{y}_{n+1/2}$ and $\bar{y}_{n+1}$ is the estimate (since $y_{n+1/2}$ and y_{n+1} are unknown) as follows :

$$\bar{y}_{n+1/2} = y_n + h/2\, f(y_n,t_n) \quad (10)$$

$$\bar{y}_{n+1} = y_n + h[\theta\, f(y_n,t_n)+(1-\theta)\, f(\bar{y}_{n+1/2}, t_{n+1/2})] \quad (11)$$

  where θ (unknown) is used to determine the maximum accuracy of the method.

The Third order Runge-Kutta method

- Substituting (10) and (11) into (9), we get the 3rd order Runge-Kutta method in the following form:

$$k_1 = h\, f(y_n, t_n)$$

$$k_2 = h\, f(y_n + \tfrac{1}{2} k_1, t_n + \tfrac{1}{2} h)$$

$$k_3 = h\, f(y_n + \theta k_1 + (1-\theta) k_2, t_n + h) \qquad (12)$$

$$y_{n+1} = y_n + 1/6\,(k_1 + 4k_2 + k_3)$$

- It can be shown that $\theta = -1$ is optimal.

The Fourth order Runge-Kutta method

- The development of the fourth-order Runge-Kutta method is similar to the third-order Runge-Kutta method, except that there is an additional intermediate step in calculating the derivative. The fourth-order Runge-Kutta method has a local error proportional to h^3.
- The fourth-order Runge-Kutta method is based on the Simson1/3 rule:

$$
\begin{aligned}
k_1 &= h\, f(y_n, t_n) \\
k_2 &= h\, f(y_n + \tfrac{1}{2} k_1, t_n + \tfrac{1}{2} h) \\
k_3 &= h\, f(y_n + \tfrac{1}{2} k_2, t_n + \tfrac{1}{2} h) \\
k_4 &= h\, f(y_n + k_3, t_n + h) \\
y_{n+1} &= y_n + 1/6\,(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
\qquad (13)
$$

The Fourth order Runge-Kutta method

- The fourth-order Runge-Kutta method is based on the Simson3/8 rule:

$$
\begin{aligned}
k_1 &= h\, f(y_n, t_n) \\
k_2 &= h\, f(y_n + 1/3\, k_1, t_n + 1/3\, h) \\
k_3 &= h\, f(y_n + 1/3\, k_1 + 1/3\, k_2, t_n + 2/3\, h) \\
k_4 &= h\, f(y_n + k_1 - k_2 + k_3, t_n + h) \\
y_{n+1} &= y_n + 1/8\, (k_1 + 3k_2 + 3k_3 + k_4)
\end{aligned}
\qquad (14)
$$

The Fourth order Runge-Kutta method

- Example 1: Calculate y(1) by solving:

  $y' = -1/(1+y2)$,

  $y(0)=1$,

  using the fourth order Runge-Kutta method with h=1

MATLAB functions to solve Differential Equations

- DE solving functions: ODE45, ODE113, ODE15S, ODE23S, ODE23T, ODE23TB
- Setting parameter: ODESET, ODEGET
- Simulate result: ODEPLOT, ODEPHAS2, ODEPHAS3, ODEPRINT
- Find solution: dsolve

MATLAB function: ODE23

- Command:

  [T,Y] = ODE23(ODEFUN,TSPAN,Y0)

- Input parameters:
  - TSPAN is the integral interval [t0,t1]
  - Y0 is the initial value
  - ODEFUN(T,Y) returns the column vector corresponding to the value f(t,y)
- Result:
  - Each row in the resulting array Y corresponds to the time in the column vector T

MATLAB function: ODE23

- E.g.: y' = sin t, y(0) = 1, $0 \leq t \leq 2PI$.

  (Given exact solution: y = -cos(t) + 2)

- MATLAB script

```
% Write a function that calculates the value of the function on the right side:
function dydt = fp(t,y)
% Calculate value for y' = sin(t)
dydt = [sin(t)];
% Find solution using ODE23
[t,y] = ode23(@fp,[0,2*pi],[1]);
plot(t,y);
hold on
plot(t,-cos(t),'r')
```

HUST

hust.edu.vn fb.com/dhbkhn