Lecture 8

Ordinary Differential Equations

Ye Ding (丁 烨)

Email: y.ding@sjtu.edu.cn

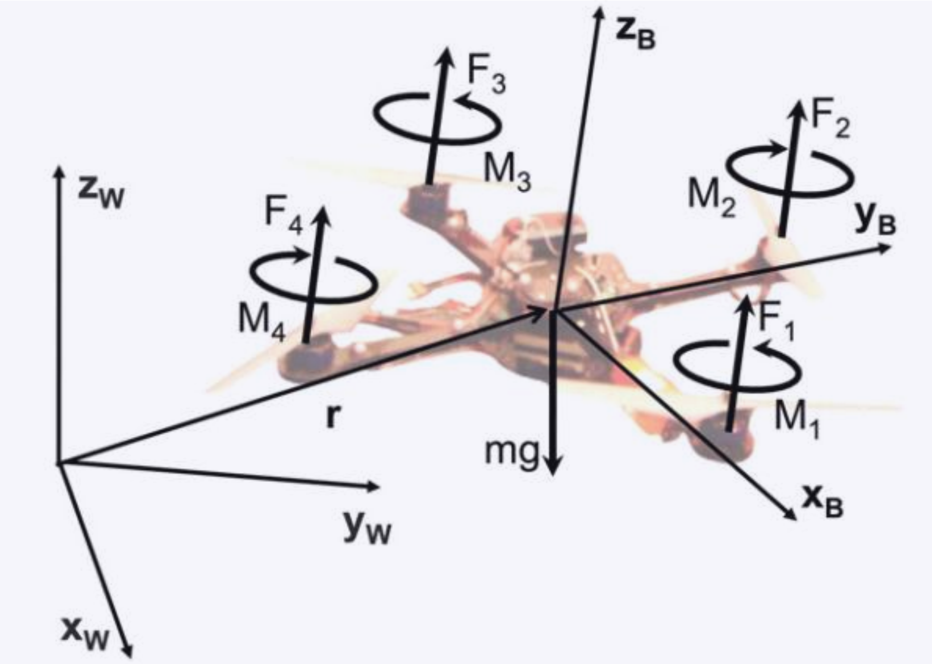
School of Mechanical Engineering

Shanghai Jiao Tong University

Ordinary Differential Equations

Motivation: from Robotics

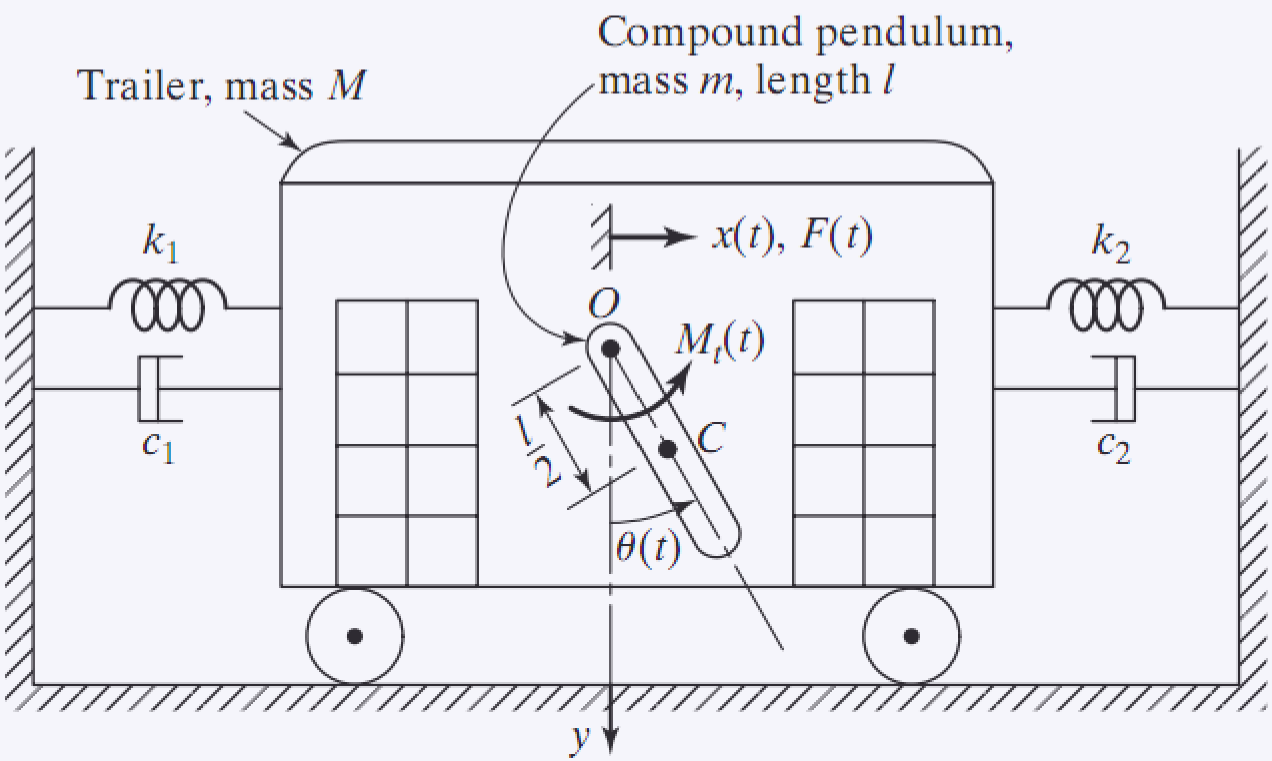
$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + {}^{W}R_B \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} F_i \end{bmatrix}$$



$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Ordinary Differential Equations

Motivation: from Vibration Theory



$$M\ddot{x} + m\ddot{x} + m\frac{l}{2}\ddot{\theta}\cos\theta - m\frac{l}{2}\dot{\theta}^2\sin\theta = -k_1 x - k_2 x - c_1\dot{x} - c_2\dot{x} + F(t)$$

$$\left(m\frac{l}{2}\ddot{\theta}\right)\frac{l}{2} + \left(m\frac{l^2}{12}\right)\ddot{\theta} + (m\ddot{x})\frac{l}{2}\cos\theta = -(mg)\frac{l}{2}\sin\theta + M_t(t)$$

Ordinary Differential Equations

- **References for ODEs**

[1] Cleve Moler, Numerical Computing with MATLAB, Society for Industrial and Applied Mathematics, 2004. Chapter 7

[2] Timothy Sauer, Numerical analysis (2nd ed.), Pearson Education, 2012. Chapter 6

[3] Richard L. Burden, J. Douglas Faires, Numerical analysis (9th ed.), Brooks/Cole, 2011. Chapter 5

Ordinary Differential Equations

- **Initial Value Problem (IVP): General Definition**

An initial value problem for a first-order ordinary differential equation is the equation together with an initial condition on a specific interval $a \le t \le b$:

$$\begin{cases} y' = f(t, y) \\ y(a) = y_a \\ t \text{ in } [a, b] \end{cases}$$

Symbolic Computation

- **Ordinary Differential Equations**

dsolve	ODE solver
odeToVectorField	Convert higher-order differential equations to systems of first-order differential equations

Symbolic Computation

- **Ordinary Differential Equations: Example 1**

$$y' = ty + t^3$$

```
>> syms t y(t)
>> y1 = dsolve(diff(y) == t*y + t^3)
```

Symbolic Computation

Ordinary Differential Equations: Example 1

$$\begin{cases} y' = ty + t^3 \\ y(0) = 1 \end{cases}$$

```
>> syms t y(t)
>> y2 = dsolve(diff(y) == t*y + t^3, y(0) == 1)
```

Symbolic Computation

Ordinary Differential Equations

To solve a system of ordinary differential equations:

```
>> syms x(t) y(t)
>> z = dsolve(diff(x) == y, diff(y) == -x)
```

Symbolic Computation

Ordinary Differential Equations

To solve a system of ordinary differential equations:

```
>> syms x(t) y(t)
>> z = dsolve(diff(x) == y, diff(y) == - y - sin(t) * x)

Warning: Explicit solution could
not be found.
```

Numerical Computation

- **Scalar Differential Equation**
  - Euler's Method
  - The Trapezoid Method
  - Runge–Kutta Methods
- **Systems of Differential Equations**
  - Runge–Kutta Methods
  - Implicit Trapezoidal Method

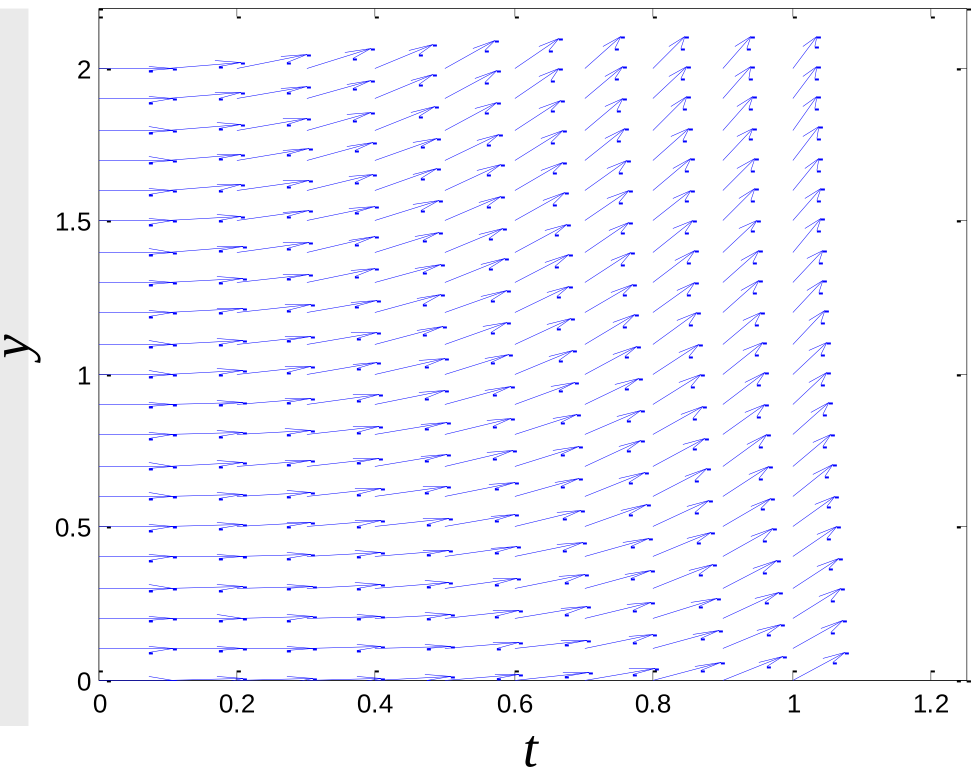
Numerical Computation

Euler's Method: Basic Idea

Draw the slope field of

$$y' = ty + t^3$$

```
tList = linspace(0,1.0,11);
yList = linspace(0,2,21);
[T,Y] = meshgrid(tList,yList);
DT = ones(size(T));
DY = T.*Y + T.^3;
DNorm = sqrt(DT.^2+DY.^2);
DT = DT./DNorm;
DY = DY./DNorm;
quiver(T,Y,DT,DY)
```



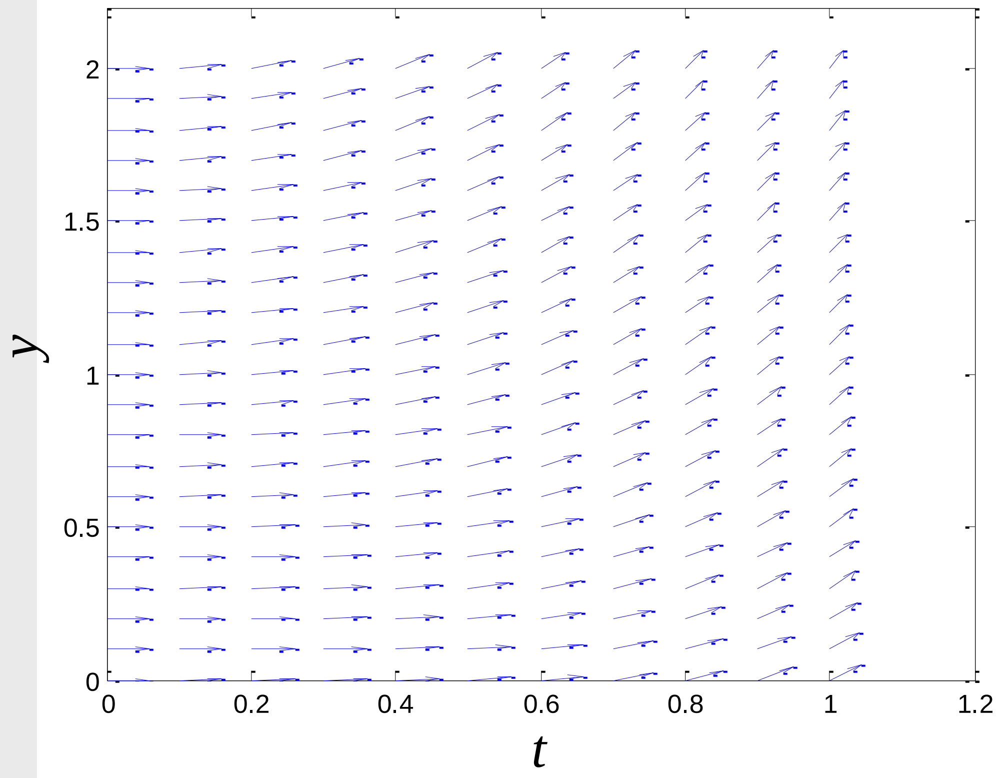
Numerical Computation

Euler's Method: Basic Idea

Draw the slope field of

$$y' = ty + t^3$$

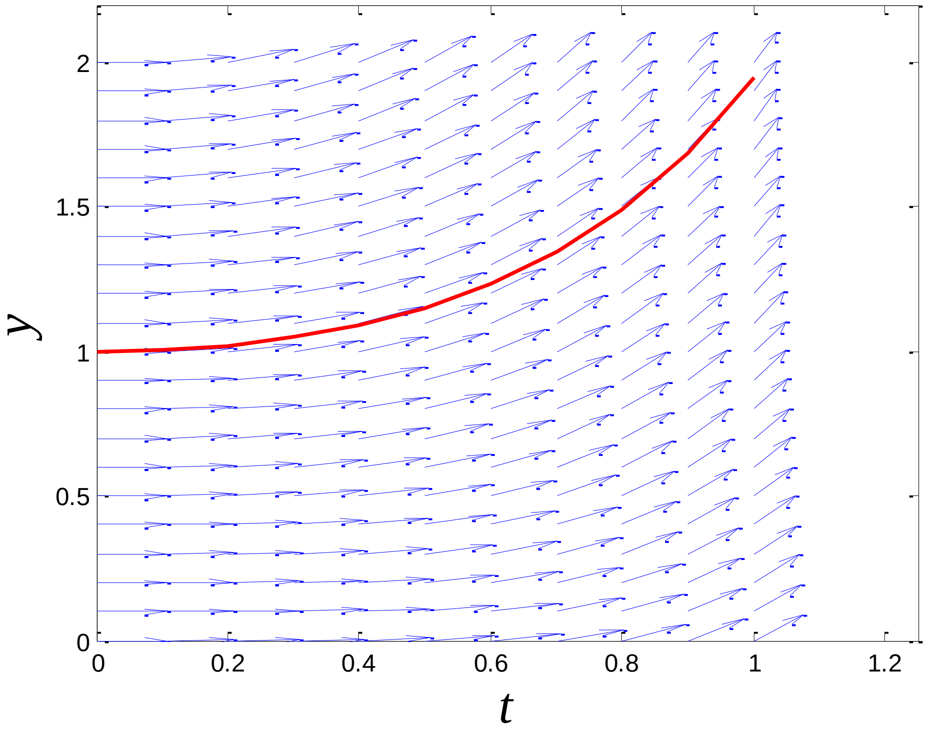
```
tList = linspace(0,1.0,11);
yList = linspace(0,2,21);
[T,Y] = meshgrid(tList,yList);
DT = ones(size(T));
DY = T.*Y + T.^3;
DNorm = sqrt(DT.^2+DY.^2);
DT = DT./DNorm;
DY = DY./DNorm;
scale = 0.5;
quiver(T,Y,DT,DY,scale)
```

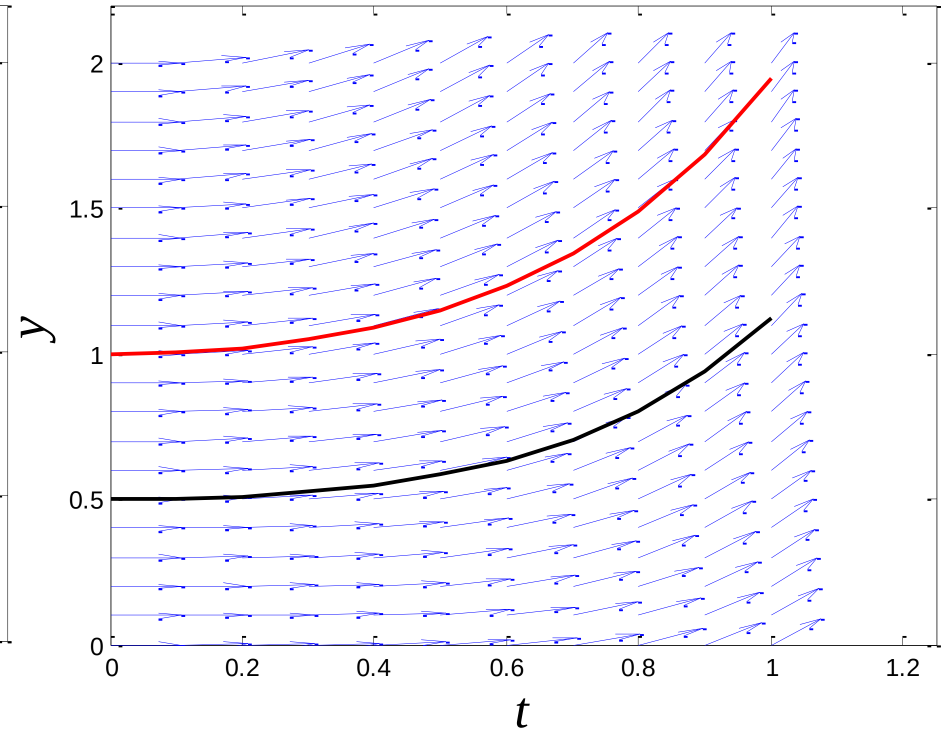


Numerical Computation

Euler’s Method: Basic Idea

$$\begin{cases} y' = ty + t^3 \\ y(0) = 1 \end{cases} \qquad \begin{cases} y' = ty + t^3 \\ y(0) = 0.5 \end{cases}$$





Numerical Computation

- **Euler's Method: Basic Idea**

How to solve the initial value problem?

$$\begin{cases} y' = ty + t^3 \\ y(0) = y_0 \\ t \text{ in } [0, 1] \end{cases}$$

Numerical Computation

- **Two-point forward-difference formula: Review**

If $f(x)$ is twice continuously differentiable, then

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(c)$$

where c is between x and $x + h$.

$$\boxed{f'(x) \approx \frac{f(x+h) - f(x)}{h}}$$

Numerical Computation

Euler's Method: Basic Idea

How to solve the initial value problem?

$$\begin{cases} y' = ty + t^3 \\ y(0) = y_0 \\ t \text{ in } [0, 1] \end{cases}$$

Step 1: Discrete the interval [0, 1] along the *t*-axis with equal step size *h*.

Step 2: Approximate the first-order derivative with the two-point forward-difference formula.

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

Numerical Computation

Euler's Method: Basic Idea

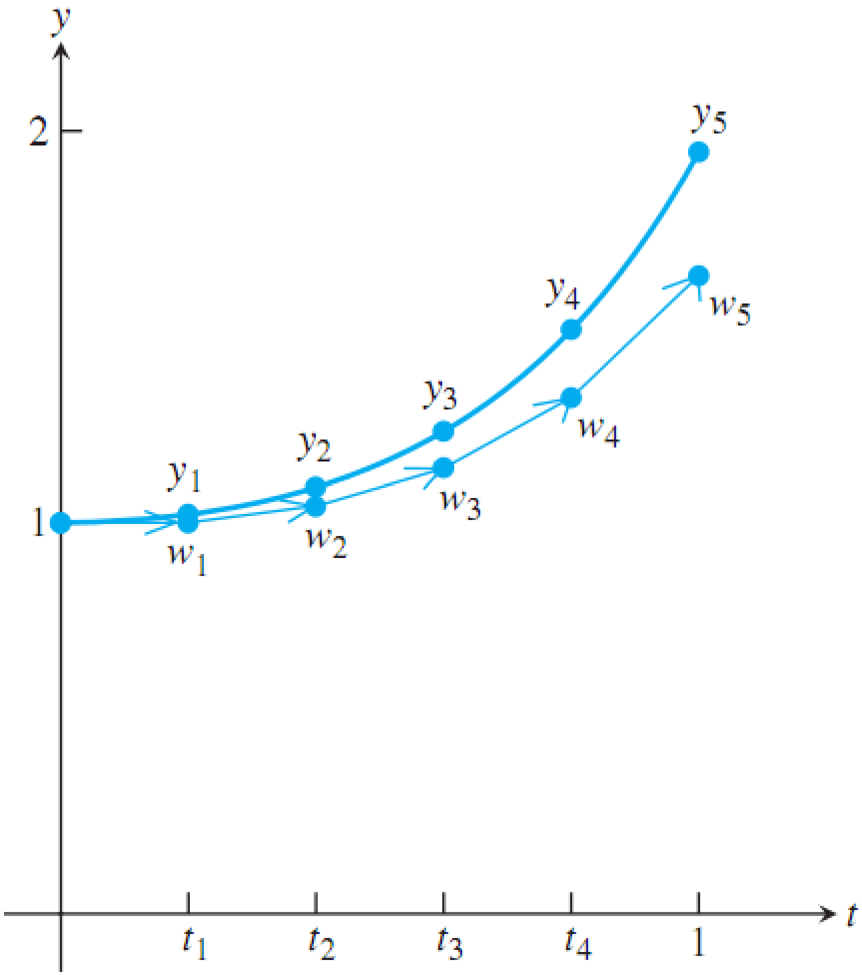
Geometric Interpretation:

$$\begin{cases} y' = ty + t^3 = f(t, y) \\ y(0) = 1 \end{cases}$$

The *t* values were selected to be $t_i = t_0 + ih$ with step size $h = 0.2$

$i = 0$: $w_0 = y_0$

$w_{i+1} = w_i + hf(t_i, w_i)$

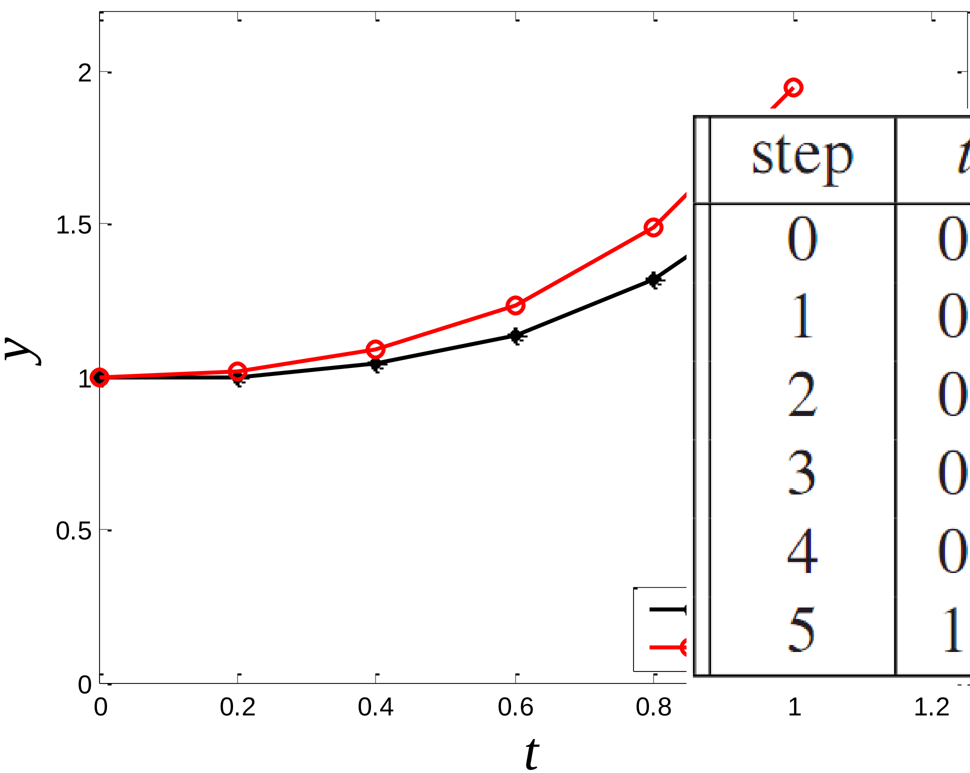


Numerical Computation

Euler's Method: Example 2

$$\begin{cases} y' = ty + t^3 \\ y(0) = 1 \end{cases}$$

with step size h = 0.2

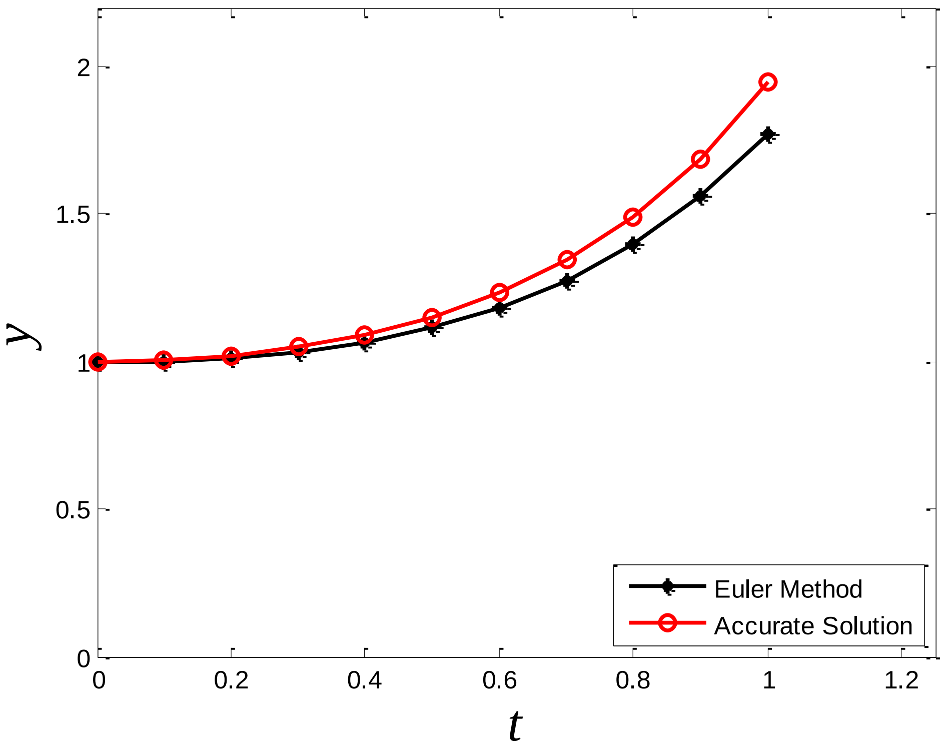


step	t_i	w_i	y_i	e_i
0	0.0	1.0000	1.0000	0.0000
1	0.2	1.0000	1.0206	0.0206
2	0.4	1.0416	1.0899	0.0483
3	0.6	1.1377	1.2317	0.0939
4	0.8	1.3175	1.4914	0.1739
5	1.0	1.6306	1.9462	0.3155

Numerical Computation

Euler's Method: Example 2

$$\begin{cases} y' = ty + t^3 \\ y(0) = 1 \end{cases}$$ with step size $h = 0.1$



step	t_i	w_i	y_i	e_i
0	0.0	1.0000	1.0000	0.0000
1	0.1	1.0000	1.0050	0.0050
2	0.2	1.0101	1.0206	0.0105
3	0.3	1.0311	1.0481	0.0170
4	0.4	1.0647	1.0899	0.0251
5	0.5	1.1137	1.1494	0.0357
6	0.6	1.1819	1.2317	0.0497
7	0.7	1.2744	1.3429	0.0684
8	0.8	1.3979	1.4914	0.0934
9	0.9	1.5610	1.6879	0.1269
10	1.0	1.7744	1.9462	0.1718

Numerical Computation

Euler's Method: Example 2

$$\begin{cases} y' = ty + t^3 \\ y(0) = 1 \end{cases}$$

with step size h = 0.1

step	t_i	w_i	y_i	e_i
0	0.0	1.0000	1.0000	0.0000
1	0.1	1.0000	1.0050	0.0050
2	0.2	1.0101	1.0206	0.0105
3	0.3	1.0311	1.0481	0.0170
4	0.4	1.0647	1.0899	0.0251
5	0.5	1.1137	1.1494	0.0357
6	0.6	1.1819	1.2317	0.0497
7	0.7	1.2744	1.3429	0.0684
8	0.8	1.3979	1.4914	0.0934
9	0.9	1.5610	1.6879	0.1269
10	1.0	1.7744	1.9462	0.1718

step	t_i	w_i	y_i	e_i
0	0.0	1.0000	1.0000	0.0000
1	0.2	1.0000	1.0206	0.0206
2	0.4	1.0416	1.0899	0.0483
3	0.6	1.1377	1.2317	0.0939
4	0.8	1.3175	1.4914	0.1739
5	1.0	1.6306	1.9462	0.3155

with step size h = 0.2

Numerical Computation

Euler's Method: Example 2

t_i	$e_i\,(h = 0.2)$	$e_i\,(h = 0.1)$	$e_i\,(h = 0.05)$
0.05			0.0012523
0.10		0.0050376	0.0025313
0.15			0.0038718
0.20	0.020604	0.010504	0.0053097
0.25			0.006883
0.30		0.016982	0.008632
0.35			0.010601
0.40	0.048261	0.025126	0.012837
0.45			0.015395
0.50		0.035721	0.018335

Numerical Computation

Euler's Method: Local Truncation Error

Assuming that *y*" is continuous, the exact solution at $t_{i+1} = t_i + h$ is

$$y(t_i + h) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(c)$$

$$\Downarrow \quad y(t_i) = w_i \text{ and } y'(t_i) = f(t_i, w_i)$$

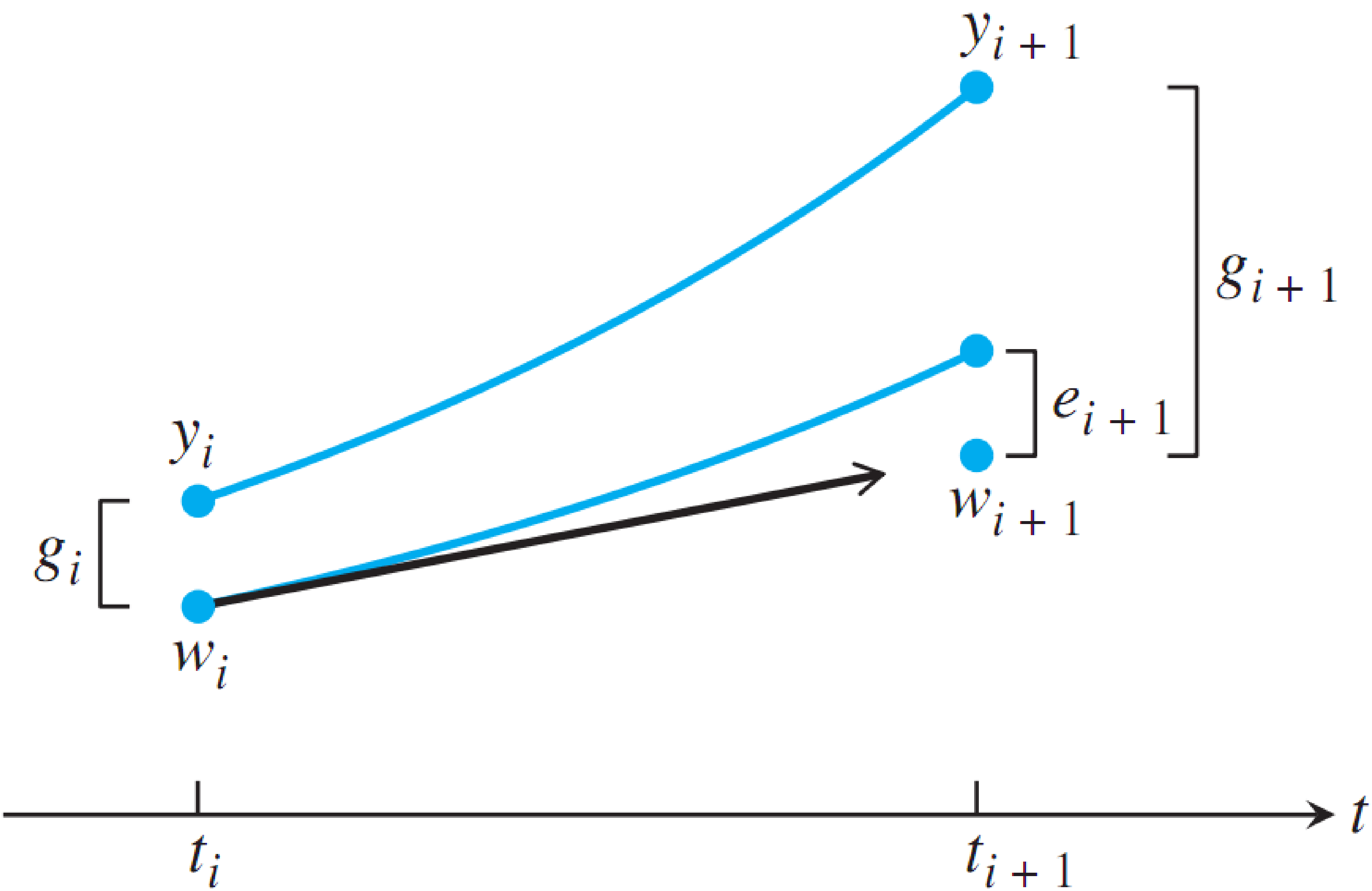
$$y(t_{i+1}) = \underline{w_i + hf(t_i, w_i)} + \frac{h^2}{2}y''(c)$$

Euler's Method: $w_{i+1} = w_i + hf(t_i, w_i)$

$$\boxed{e_{i+1} = |w_{i+1} - y(t_{i+1})| = \frac{h^2}{2}|y''(c)|}$$

Numerical Computation

Euler's Method: Geometric Interpretation for Errors



Numerical Computation

Euler's Method: Theory

A function $f(t, y)$ is Lipschitz continuous in the variable y on the rectangle $S = [a,b] \times [\alpha,\beta]$ if there exists a constant L (called the Lipschitz constant) satisfying

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|$$

for each (t, y_1), (t, y_2) in S.

Numerical Computation

Euler's Method: Theory

Theorem (Ref. [2], P. 288). Assume that $f(t, y)$ is Lipschitz continuous in the variable y on the set $[a,b]\times[\alpha,\beta]$ and that $\alpha < y_a < \beta$. Then there exists c between a and b such that the initial value problem

$$\begin{cases} y' = f(t, y) \\ y(a) = y_a \\ t \text{ in } [a, c] \end{cases}$$

has exactly one solution $y(t)$. Moreover, if f is Lipschitz on $[a,b]\times(-\infty,\infty)$, then there exists exactly one solution on $[a,b]$.

Numerical Computation

Euler's Method: Theory

Theorem (Ref. [2], P. 289). Assume that $f(t, y)$ is Lipschitz continuous in the variable y on the set $[a,b]\times[\alpha,\beta]$. If $Y(t)$ and $Z(t)$ are solutions in S of the differential equation

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

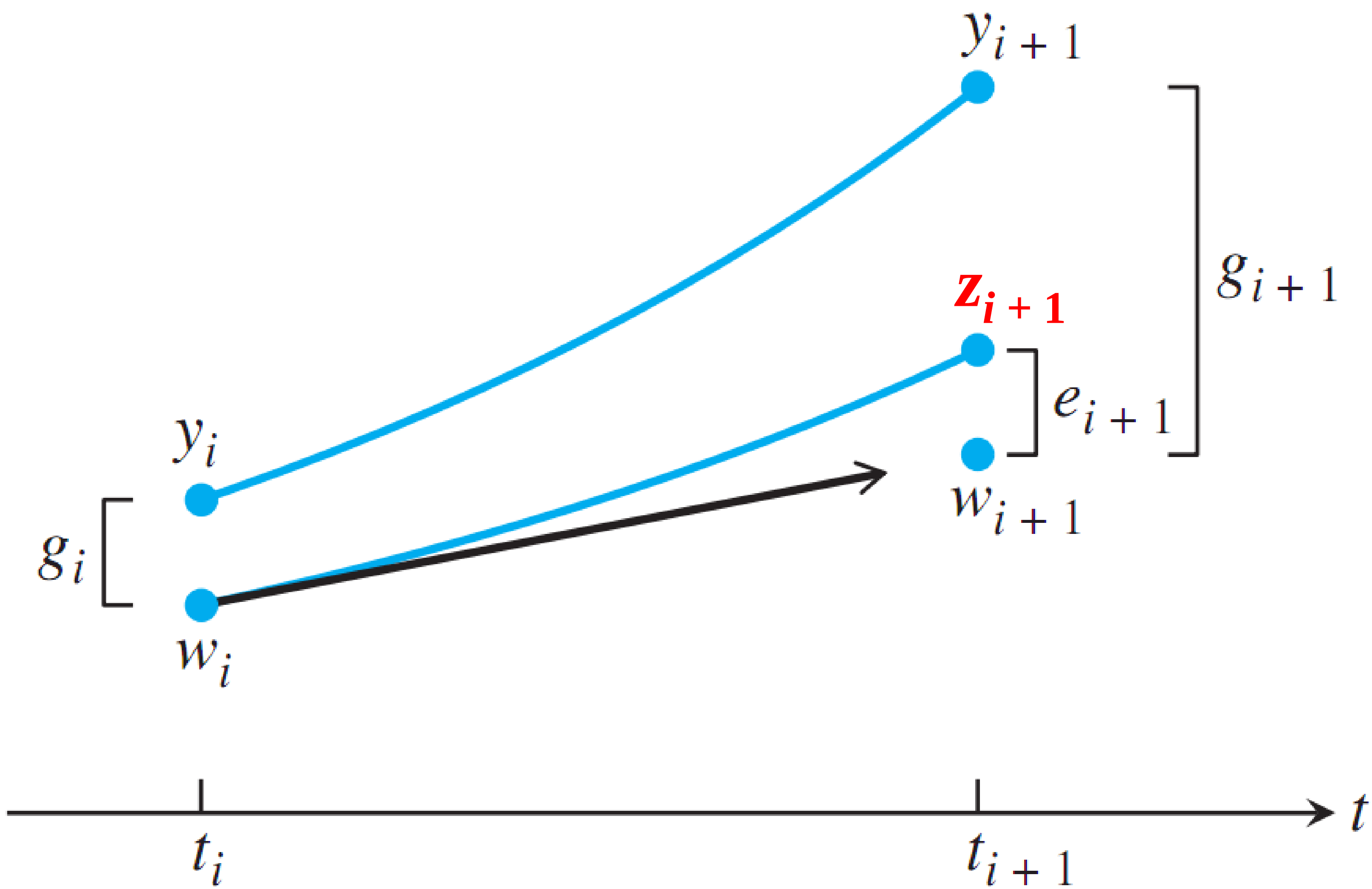
with initial conditions $Y(a)$ and $Z(a)$ respectively, then

$$|Y(t) - Z(t)| \le e^{L(t-a)}|Y(a) - Z(a)|$$

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Numerical Computation

Euler's Method: Geometric Interpretation for Errors



Numerical Computation

Euler's Method: Local Truncation Error

The local truncation error, or one-step error, is defined as

$$e_{i+1} = |w_{i+1} - z(t_{i+1})|$$

the difference between the value of the solver w_{i+1} on that interval and the correct solution of the "one-step initial value problem" z_{i+1}:

$$\begin{cases} y' = f(t, y) \\ y(t_i) = w_i \\ t \ \text{ in } \ [t_i, t_{i+1}] \end{cases}$$

Numerical Computation

Euler's Method: Global Truncation Error

At step *i*, the global truncation error is defined as

$$g_i = |w_i - y_i|$$

the difference between the ODE solver (Euler's Method) approximation and the correct solution of the initial value problem

$$\begin{cases} y' = f(t, y) \\ y(a) = y_a \\ t \ \text{ in } \ [a, b] \end{cases}$$

Numerical Computation

Euler's Method: Global Truncation Error

At step $i = 0$, the global truncation error is

$$g_0 = |w_0 - y_0| = |y_a - y_a| = 0$$

At step $i = 1$, the global error is equal to the first local error

$$g_1 = e_1 = |w_1 - y_1|$$

At step $i = 2$, the global error is

$$\begin{aligned} g_2 = |w_2 - y_2| = |w_2 - z(t_2) + z(t_2) - y_2| \\ &\leq |w_2 - z(t_2)| + |z(t_2) - y_2| \\ &\leq e_2 + e^{Lh} g_1 \\ &= e_2 + e^{Lh} e_1 \end{aligned}$$

Numerical Computation

Euler's Method: Global Truncation Error

At step $i = 3$, the global truncation error is

$$g_3 = |w_3 - y_3| \le e_3 + e^{Lh} g_2$$
$$\le e_3 + e^{Lh} e_2 + e^{2Lh} e_1$$

At step i, the global error is

$$g_i = |w_i - y_i| \le e_i + e^{Lh} e_{i-1} + e^{2Lh} e_{i-2} + \cdots + e^{(i-1)Lh} e_1$$

$$\Downarrow \quad e_i \le Ch^{k+1}$$

$$g_i \le \frac{Ch^k}{L}(e^{L(t_i - a)} - 1)$$

Numerical Computation

Euler's Method: Convergence

Assume that $f(t, y)$ has a Lipschitz constant L for the variable y and that the solution y_i of the initial value problem

$$\begin{cases} y' = f(t, y) \\ y(a) = y_a \\ t \text{ in } [a, b] \end{cases}$$

at t_i is approximated by w_i , using Euler's Method. Let M be an upper bound for $|y''(t)|$ on $[a,b]$. Then

$$|w_i - y_i| \leq \frac{Mh}{2L}(e^{L(t_i - a)} - 1)$$

Numerical Computation

- **Scalar Differential Equation**
  - **Euler's Method**
  - **The Trapezoid Method**
  - **Runge–Kutta Methods**
- **Systems of Differential Equations**
  - **Runge–Kutta Methods**
  - **Implicit Trapezoidal Method**

Numerical Computation

Trapezoid Method: Basic Idea

The IVP: $$\begin{cases} y' = f(t, y) \\ y(a) = y_a \\ t \text{ in } [a, b] \end{cases}$$

$$\int_{t_i}^{t_i+h} f(t)\,dt = \int_{t_i}^{t_i+h} y'(t)\,dt = y(t_i + h) - y(t_i)$$

$$\Downarrow \quad \int_{t_i}^{t_i+h} f(t)\,dt \approx h f(t_i)$$

Euler's Method

$$\boxed{\begin{aligned} & i = 0: \ w_0 = y_0 \\ & w_{i+1} = w_i + hf(t_i, w_i),\ i = 1, 2, \ldots \end{aligned}}$$

Numerical Computation

Trapezoid Method: Basic Idea

$$\int_{t_i}^{t_i+h} f(t)\,dt = \int_{t_i}^{t_i+h} y'(t)\,dt = y(t_i+h) - y(t_i)$$

$$\Downarrow \quad \int_{t_i}^{t_i+h} f(t)\,dt \approx \frac{h}{2}[f(t_i) + f(t_i+h)]$$

Trapezoid Method

$$i = 0: \quad w_0 = y_0$$

$$w_{i+1} = w_i + [f(t_i, w_i) + f(t_i + h, w_{i+1})] \times h/2$$

The Implicit Trapezoidal method

Numerical Computation

The Explicit Trapezoid Method

Trapezoid Method

$$i = 0: \quad w_0 = y_0$$

$$w_{i+1} = w_i + [f(t_i, w_i) + f(t_i + h, w_{i+1})] \times h/2$$

$$w_i + hf(t_i, w_i)$$

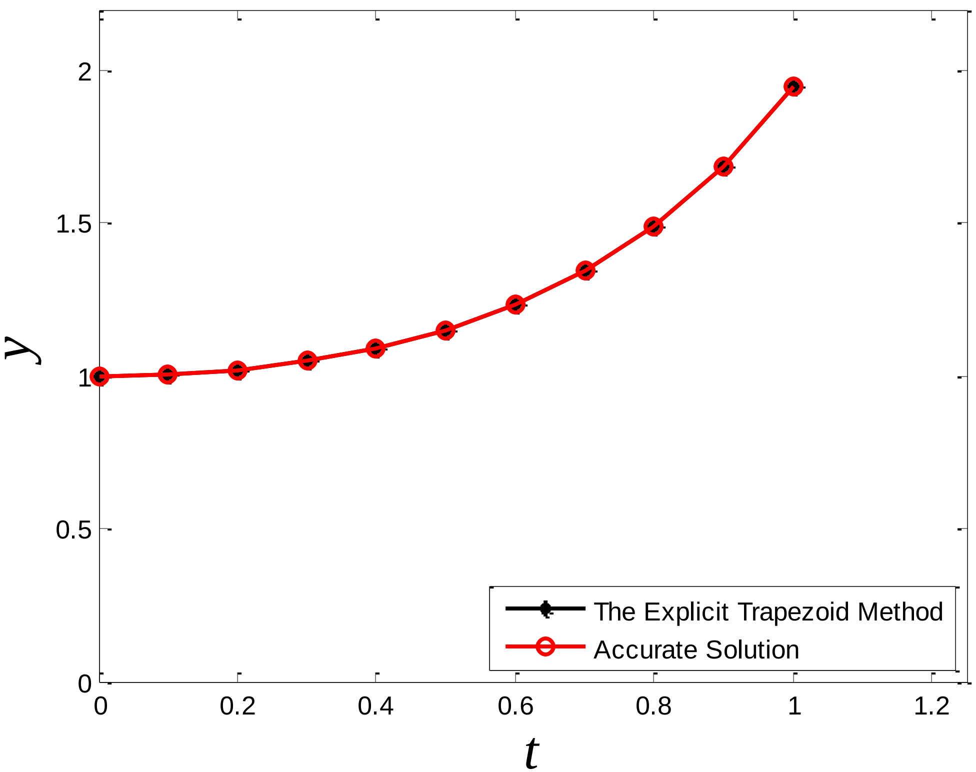
$$w_0 = y_0$$

$$w_{i+1} = w_i + \frac{h}{2}(f(t_i, w_i) + f(t_i + h, w_i + hf(t_i, w_i)))$$

Numerical Computation

The Explicit Trapezoid Method: Example 3

$$\begin{cases} y' = ty + t^3 \\ y(0) = 1 \end{cases}$$ **with step size *h* = 0.1**



step	t_i	w_i	y_i	e_i
0	0.0	1.0000	1.0000	0.0000
1	0.1	1.0051	1.0050	0.0001
2	0.2	1.0207	1.0206	0.0001
3	0.3	1.0483	1.0481	0.0002
4	0.4	1.0902	1.0899	0.0003
5	0.5	1.1499	1.1494	0.0005
6	0.6	1.2323	1.2317	0.0006
7	0.7	1.3437	1.3429	0.0008
8	0.8	1.4924	1.4914	0.0010
9	0.9	1.6890	1.6879	0.0011
10	1.0	1.9471	1.9462	0.0010

Numerical Computation

The Explicit Trapezoid Method: Example 3

t_i	$e_i\,(h = 0.2)$	$e_i\,(h = 0.1)$	$e_i\,(h = 0.05)$
0.05			7.8027e-7
0.10		0.000012437	4.6601e-6
0.15			0.000011586
0.20	0.00019598	0.000073235	0.000021478
0.25			0.000034227
0.30		0.00017881	0.000049687
0.35			0.00006767
0.40	0.0010841	0.00032333	0.000087939
0.45			0.00011019
0.50		0.00049766	0.00013404

Numerical Computation

- **The Explicit Trapezoid Method: Truncation Error**

The correct extension of the solution at t_{i+1} can be given by the Taylor expansion

$$y_{i+1} = y(t_i + h) = y_i + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \frac{h^3}{6}y'''(c)$$

$$y''(t) = \frac{\partial f}{\partial t}(t, y) + \frac{\partial f}{\partial y}(t, y)y'(t)$$

$$= \frac{\partial f}{\partial t}(t, y) + \frac{\partial f}{\partial y}(t, y)f(t, y)$$

$$\Downarrow$$

$$y_{i+1} = y_i + hf(t_i, y_i) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t}(t_i, y_i) + \frac{\partial f}{\partial y}(t_i, y_i)f(t_i, y_i)\right) + \frac{h^3}{6}y'''(c)$$

Numerical Computation

The Explicit Trapezoid Method: Truncation Error

The Explicit Trapezoid Method can be written

$$w_{i+1} = y_i + \frac{h}{2}\left(f(t_i, y_i) + f(t_i + h, y_i + hf(t_i, y_i)) \right)$$

$$\begin{aligned} & f(t_i + h, y_i + hf(t_i, y_i)) \\ & = f(t_i, y_i) + h\frac{\partial f}{\partial t}(t_i, y_i) + hf(t_i, y_i)\frac{\partial f}{\partial y}(t_i, y_i) + O(h^2) \end{aligned}$$

$$\begin{aligned} w_{i+1} = y_i &+ \frac{h}{2} f(t_i, y_i) + \frac{h}{2}\left(f(t_i, y_i) + h\left(\frac{\partial f}{\partial t}(t_i, y_i) \right.\right. \\ &+ \left.\left. f(t_i, y_i)\frac{\partial f}{\partial y}(t_i, y_i) \right) + O(h^2) \right) \\ &= y_i + hf(t_i, y_i) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t}(t_i, y_i) + f(t_i, y_i)\frac{\partial f}{\partial y}(t_i, y_i) \right) + O(h^3) \end{aligned}$$

Numerical Computation

- **The Explicit Trapezoid Method: Truncation Error**

The Local Truncation Error is:

$$y_{i+1} - w_{i+1} = O(h^3)$$

The Global Truncation Error of the Explicit Trapezoid Method is proportional to h^2.

Numerical Computation

- **Scalar Differential Equation**
  - **Euler's Method**
  - **The Trapezoid Method**
  - **Runge–Kutta Methods**
- **Systems of Differential Equations**
  - **Runge–Kutta Methods**
  - **Implicit Trapezoidal Method**

Numerical Computation

Second-Order Runge-Kutta Method: RK2

Euler's Method: $w_{i+1} = w_i + hf(t_i, w_i)$

Trapezoid Method:

$$w_{i+1} = w_i + \frac{h}{2}(f(t_i, w_i) + f(t_i + h, w_i + hf(t_i, w_i)))$$

A unified form:

$$\boxed{w_{i+1} = w_i + h\phi}$$

where ϕ is called an increment function, which can be interpreted as a representative slope over the interval.

Numerical Computation

- **Second-Order Runge-Kutta Method: RK2**

A general second-order version:

$$w_{i+1} = w_i + (a_1 k_1 + a_2 k_2)h$$

where

$$k_1 = f(t_i, w_i)$$

$$k_2 = f(t_i + p_1 h, w_i + q_{11} k_1 h)$$

The Local Truncation Error:

$$y_{i+1} - w_{i+1} \overset{?}{=} O(h^3)$$

Numerical Computation

Second-Order Runge-Kutta Method: RK2

$$y_{i+1} = y_i + hf(t_i, y_i) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t}(t_i, y_i) + \frac{\partial f}{\partial y}(t_i, y_i) f(t_i, y_i)\right) + \frac{h^3}{6} y'''(c)$$

$$w_{i+1} = y_i + h[a_1 f(t_i, y_i) + a_2 \underline{f(t_i + p_1 h, y_i + q_{11} f(t_i, y_i)h)}]$$

$$f(t_i + p_1 h, y_i + q_{11} f(t_i, y_i)h)$$
$$= f(t_i, y_i) + p_1 h f_t(t_i, y_i) + q_{11} h f(t_i, y_i) f_y(t_i, y_i) + O(h^2)$$

$$\begin{cases} a_1 + a_2 = 1 \\ a_2 p_1 = 1/2 \\ a_2 q_{11} = 1/2 \end{cases} \Rightarrow \begin{cases} a_1 = 1 - a_2 \\ p_1 = q_{11} = \dfrac{1}{2a_2} \end{cases}$$

Numerical Computation

- **Second-Order Runge-Kutta Method: RK2**

$$\begin{cases} a_1 = 1 - a_2 \\ p_1 = q_{11} = \dfrac{1}{2a_2} \end{cases}$$

- **$a_2 = 1/2$, $a_1 = 1/2$, $p_1 = q_{11} = 1$**

$$w_{i+1} = w_i + \frac{h}{2}\left(f(t_i, w_i) + f(t_i + h, w_i + hf(t_i, w_i))\right)$$

- **$a_2 = 1$, $a_1 = 0$, $p_1 = q_{11} = 1/2$**

$$w_{i+1} = w_i + hf\left(t_i + \frac{h}{2}, w_i + \frac{h}{2} f(t_i, w_i)\right)$$

the Midpoint Method

Numerical Computation

- **Runge–Kutta Method of Order Four: RK4**

$$w_{i+1} = w_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where $k_1 = f(t_i, w_i)$

$$k_2 = f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2}k_2\right)$$

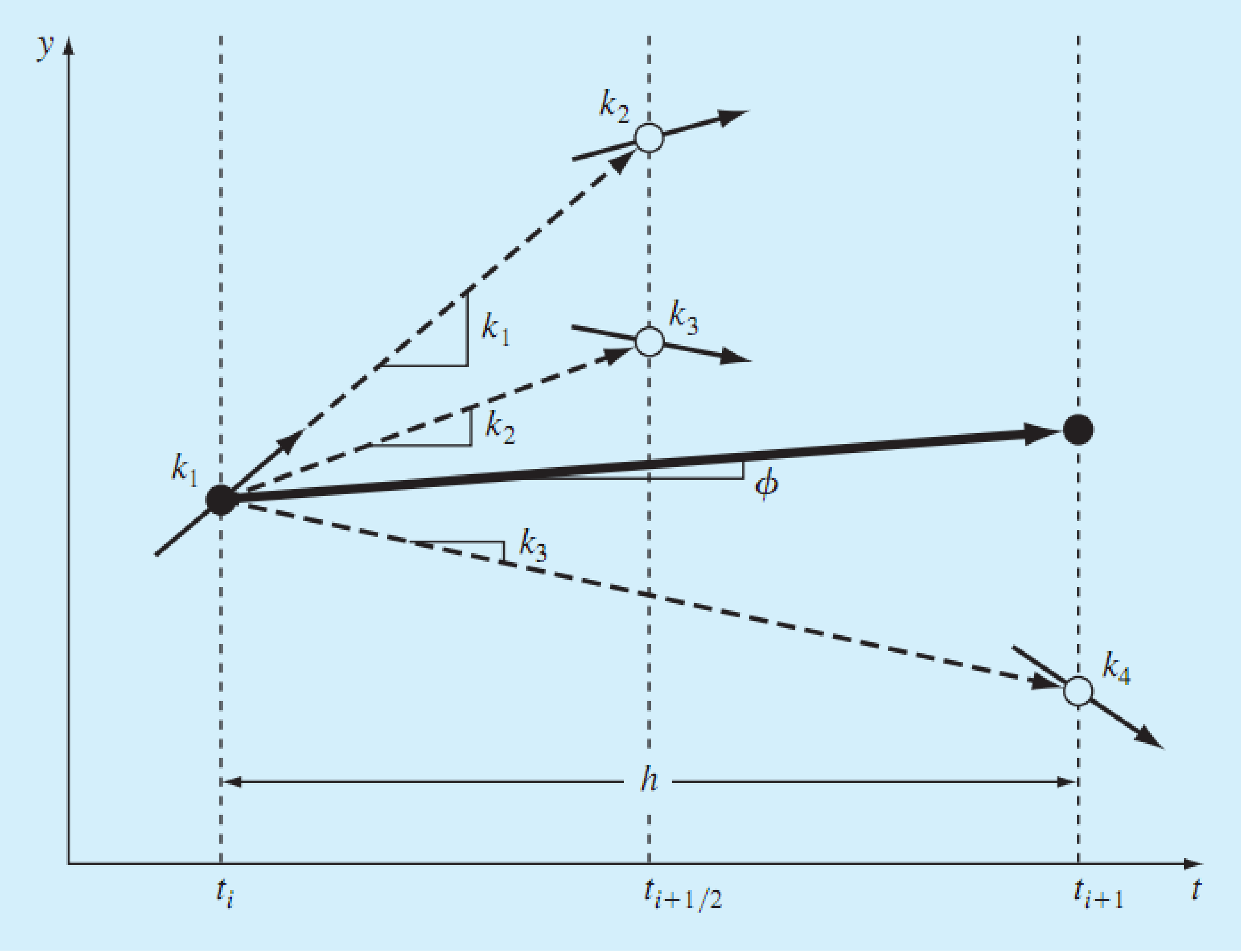
$$k_4 = f(t_i + h, w_i + hk_3)$$

The Local Truncation Error is $O(h^5)$.

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

Numerical Computation

RK4: Geometric Interpretation



Numerical Computation

RK4: Example 4

$$\begin{cases} y' = ty + t^3 \\ y(0) = 1 \end{cases}$$

with step size h = 0.2, 0.1, 0.05

t_i	$e_i(h = 0.2)$	$e_i(h = 0.1)$	$e_i(h = 0.05)$
0.05			1.6286e-10
0.10		1.0443e-8	6.5086e-10
0.15			1.4614e-9
0.20	6.7341e-7	4.1639e-8	2.5901e-9
0.25			4.0317e-9
0.30		9.3113e-8	5.781e-9
0.35			7.8345e-9
0.40	2.6787e-6	1.6452e-7	1.0193e-8

Numerical Computation

- Scalar Differential Equation
  - Euler's Method
  - The Trapezoid Method
  - Runge–Kutta Methods
- Systems of Differential Equations
  - Runge–Kutta Methods
  - Implicit Trapezoidal Method

Systems of ODEs

- **Higher Order Equations**

An *n*th-order ordinary differential equation:

$$y^{(n)} = f(t, y, y', y'', \ldots, y^{(n-1)})$$

Define new variables:

$$\begin{cases} y_1 = y \\ y_2 = y' \\ y_3 = y'' \\ \vdots \\ y_n = y^{(n-1)} \end{cases}$$

$$y_n' = f(t, y_1, y_2, \ldots, y_n)$$

Systems of ODEs

- **Higher Order Equations**

An *n*th-order ordinary differential equation:

$$y^{(n)} = f(t, y, y', y'', \ldots, y^{(n-1)})$$

$\Downarrow$

A system of first-order equations:

$$\left\{ \begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ y_3' &= y_4 \\ &\vdots \\ y_{n-1}' &= y_n, \\ y_n' &= f(t, y_1, \ldots, y_n) \end{aligned} \right.$$

Systems of ODEs

Runge–Kutta Method for Systems of ODEs

The IVP:

$$\begin{cases} \dfrac{dy_1}{dt} = f_1(t, y_1, y_2, \ldots, y_n) \\ \dfrac{dy_2}{dt} = f_2(t, y_1, y_2, \ldots, y_n) \\ \qquad \vdots \\ \dfrac{dy_n}{dt} = f_n(t, y_1, y_2, \ldots, y_n) \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{cases}$$

where

$$\mathbf{y}(t) = [y_1(t), y_2(t), \cdots, y_n(t)]^T$$

Systems of ODEs

- **Runge–Kutta Method for Systems of ODEs**

The IVP: $$\begin{cases} \mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}) \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{cases}$$

The RK4: $$\mathbf{w}_{i+1} = \mathbf{w}_i + \frac{h}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right)$$

$$\mathbf{k}_1 = \mathbf{f}(t_i, \mathbf{y}_i)$$

$$\mathbf{k}_2 = \mathbf{f}(t_i + \frac{h}{2}, \mathbf{y}_i + \frac{h}{2}\mathbf{k}_1)$$

$$\mathbf{k}_3 = \mathbf{f}(t_i + \frac{h}{2}, \mathbf{y}_i + \frac{h}{2}\mathbf{k}_2)$$

$$\mathbf{k}_4 = \mathbf{f}(t_i + h, \mathbf{y}_i + h\mathbf{k}_3)$$

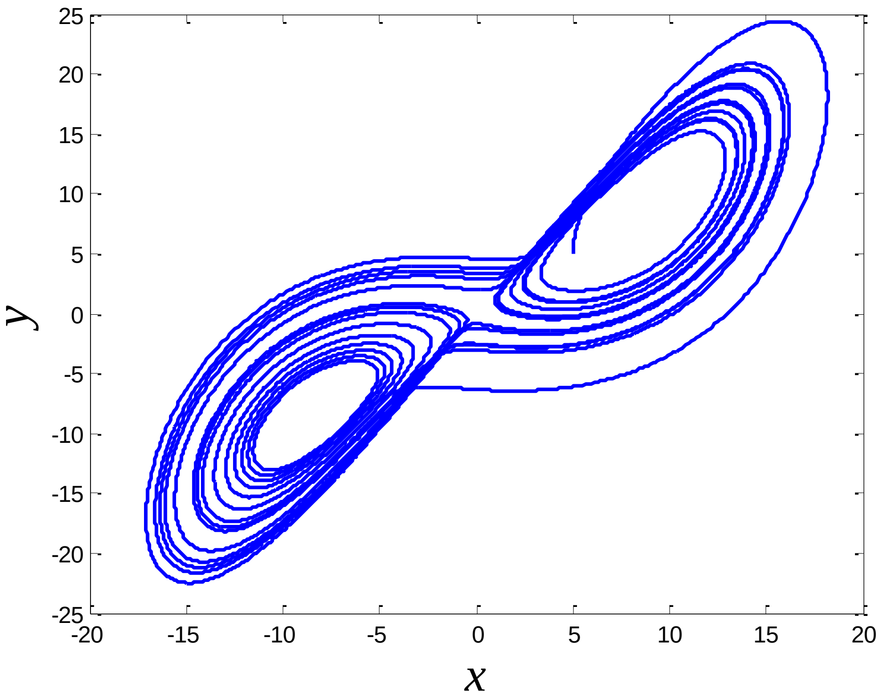
Systems of ODEs

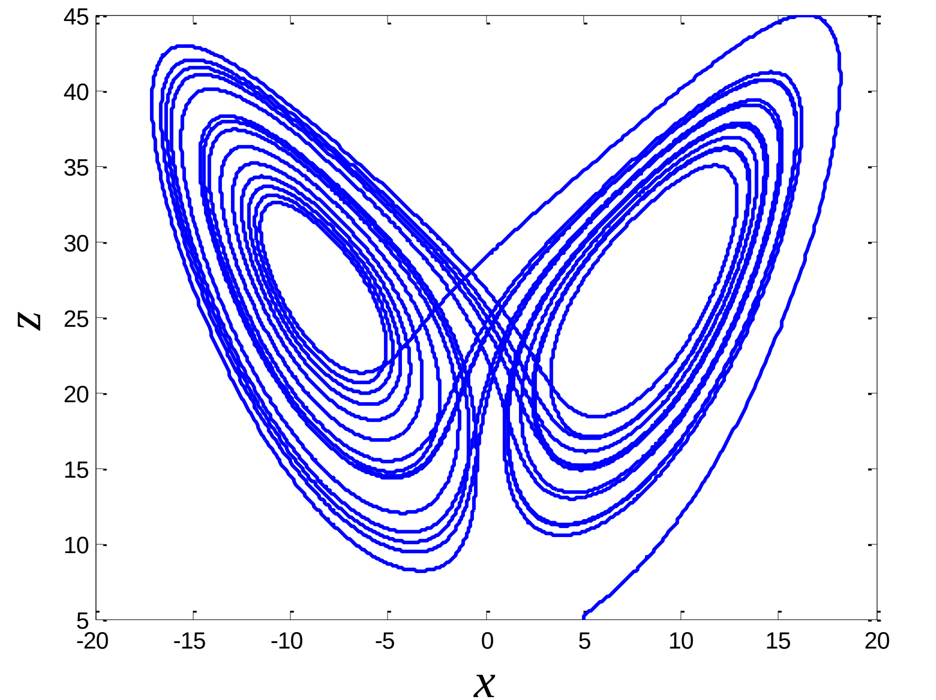
- **Runge–Kutta Methods: Example 5 Lorenz equations**

$$\begin{cases} x' = -sx + sy \\ y' = -xz + rx - y \\ z' = xy - bz \end{cases}$$

***s* = 10, *r* = 28, and *b* = 8/3**

Initial conditions: [5.001, 5, 5]





Systems of ODEs

- **Stiff Differential Equations: Example 6**

The IVP:

$$u_1' = 9u_1 + 24u_2 + 5\cos t - \frac{1}{3}\sin t, \quad u_1(0) = \frac{4}{3}$$

$$u_2' = -24u_1 - 51u_2 - 9\cos t + \frac{1}{3}\sin t, \quad u_2(0) = \frac{2}{3}$$

has the unique solution

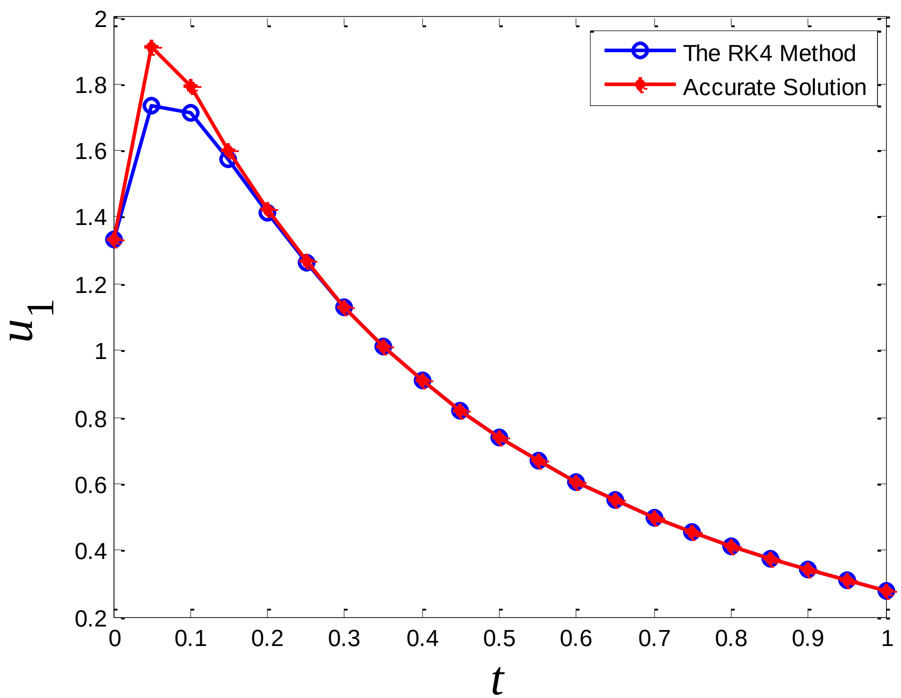
$$u_1(t) = 2e^{-3t} - e^{-39t} + \frac{1}{3}\cos t,$$

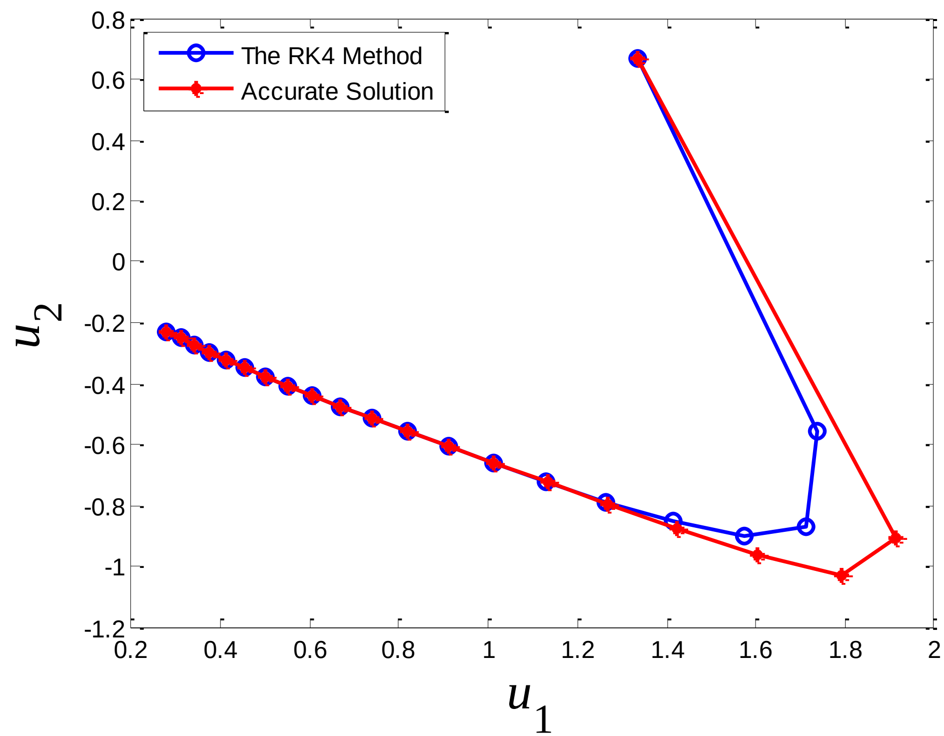
$$u_2(t) = -e^{-3t} + 2e^{-39t} - \frac{1}{3}\cos t$$

Systems of ODEs

Stiff Differential Equations: Example 6

RK4 with h = 0.05

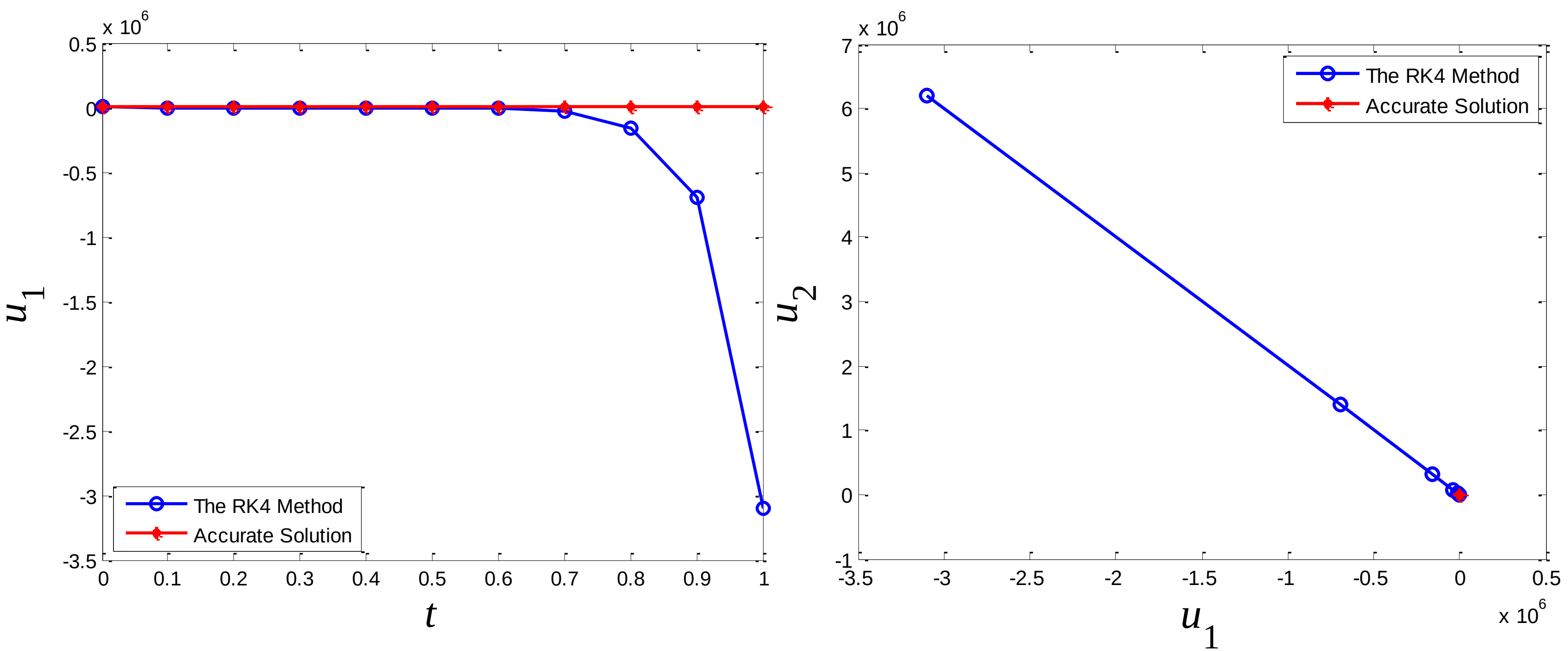




Systems of ODEs

Stiff Differential Equations: Example 6

RK4 with *h* = 0.1



Systems of ODEs

Stiff Differential Equations: Example 6

t	$u_1(t)$	$w_1(t)$ $h=0{,}05$	$w_1(t)$ $h=0{,}1$	$u_2(t)$	$w_2(t)$ $h=0{,}05$	$w_2(t)$ $h=0{,}1$
0,1	1,793061	1,712219	−2,645169	−1,032001	−0,8703152	7,844527
0,2	1,423901	1,414070	−18,45158	−0,8746809	−0,8550148	38,87631
0,3	1,131575	1,130523	−87,47221	−0,7249984	−0,7228910	176,4828
0,4	0,9094086	0,9092763	−934,0722	−0,6082141	−0,6079475	789,3540
0,5	0,7387877	9,7387506	−1760,016	−0,5156575	−0,5155810	3520,00
0,6	0,6057094	0,6056833	−7848,550	−0,4404108	−0,4403558	15697,84
0,7	0,4998603	0,4998361	−34989,63	−0,3774038	−0,3773540	69979,87
0,8	0,4136714	0,4136490	−155979,4	−0,3229535	−0,3229078	311959,5
0,9	0,3416143	0,3415939	−695332,0	−0,2744088	−0,2743673	1390664,
1,0	0,2796748	0,2796568	−3099671,	−0,2298877	−0,2298511	6199352,

Numerical Computation

- Scalar Differential Equation
  - Euler’s Method
  - The Trapezoid Method
  - Runge–Kutta Methods
- Systems of Differential Equations
  - Runge–Kutta Methods
  - Implicit Trapezoidal Method

Systems of ODEs

- **Implicit Trapezoidal Method Revisited**

The IVP:

$$\begin{cases} \mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}) \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{cases}$$

Implicit Trapezoid Method

$$i = 0: \quad w_0 = y_0$$

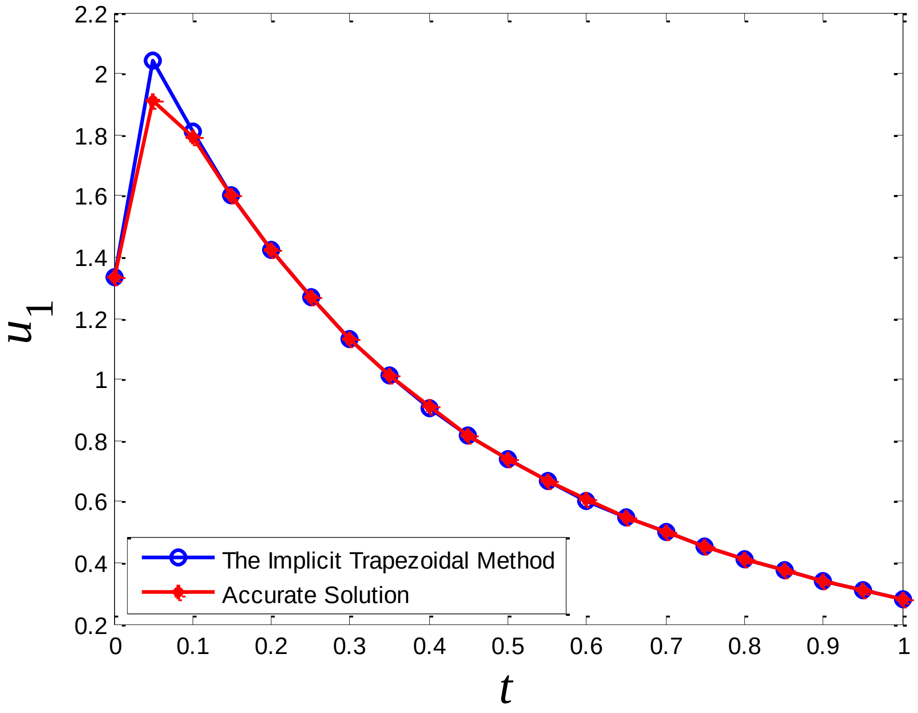
$$w_{i+1} = w_i + [f(t_i, w_i) + f(t_i + h, w_{i+1})] \times h/2$$

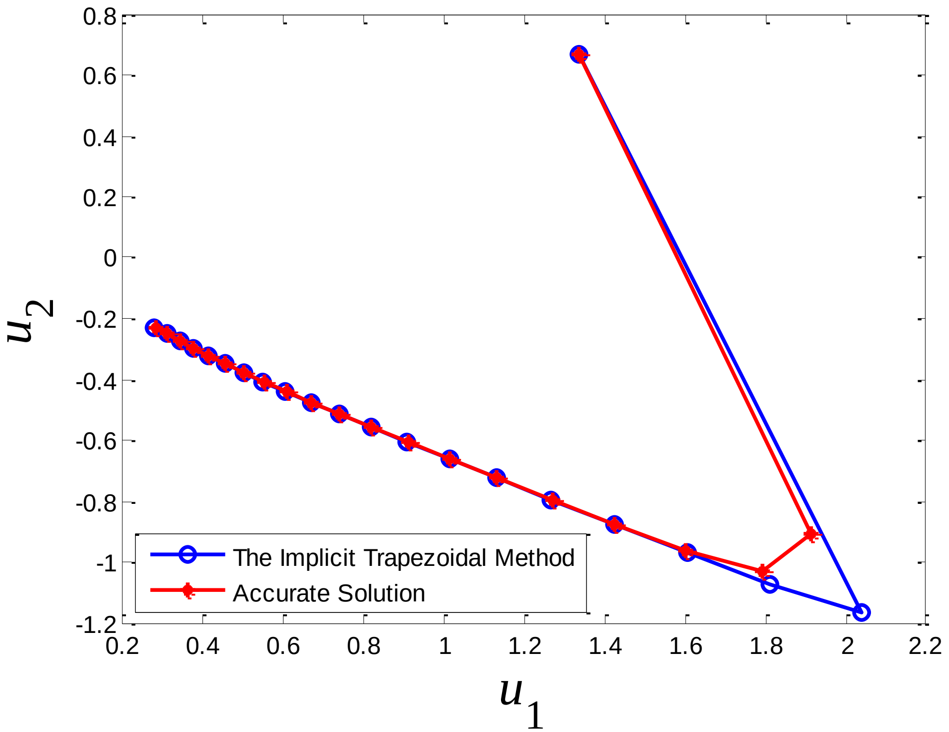
with the help of Newton Iteration Method

Systems of ODEs

- **Implicit Trapezoidal Method Revisited: Example 6**

Implicit Trapezoidal Method with *h* = 0.05



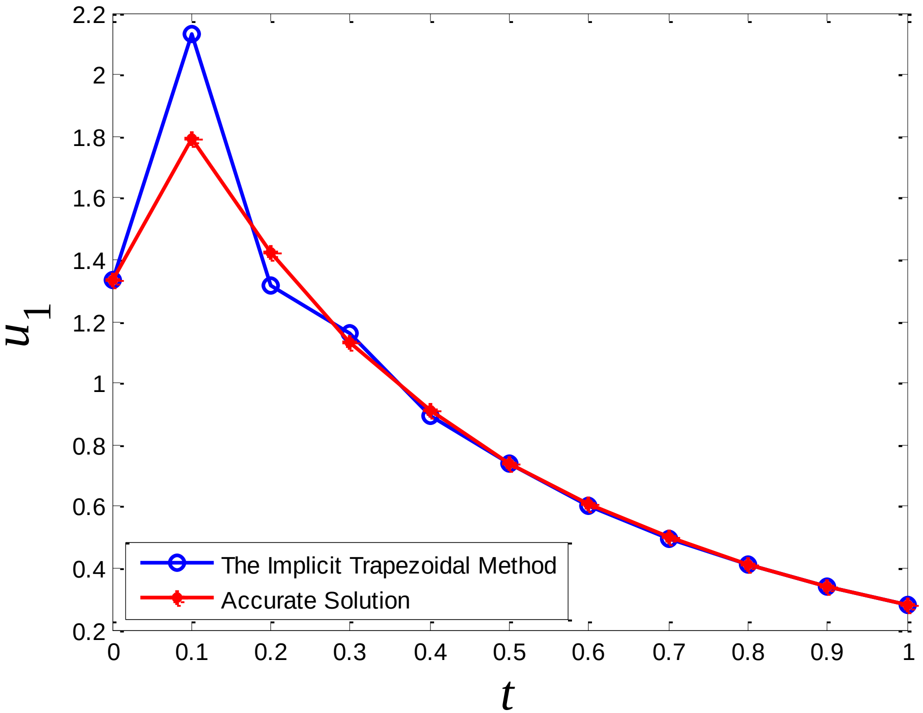


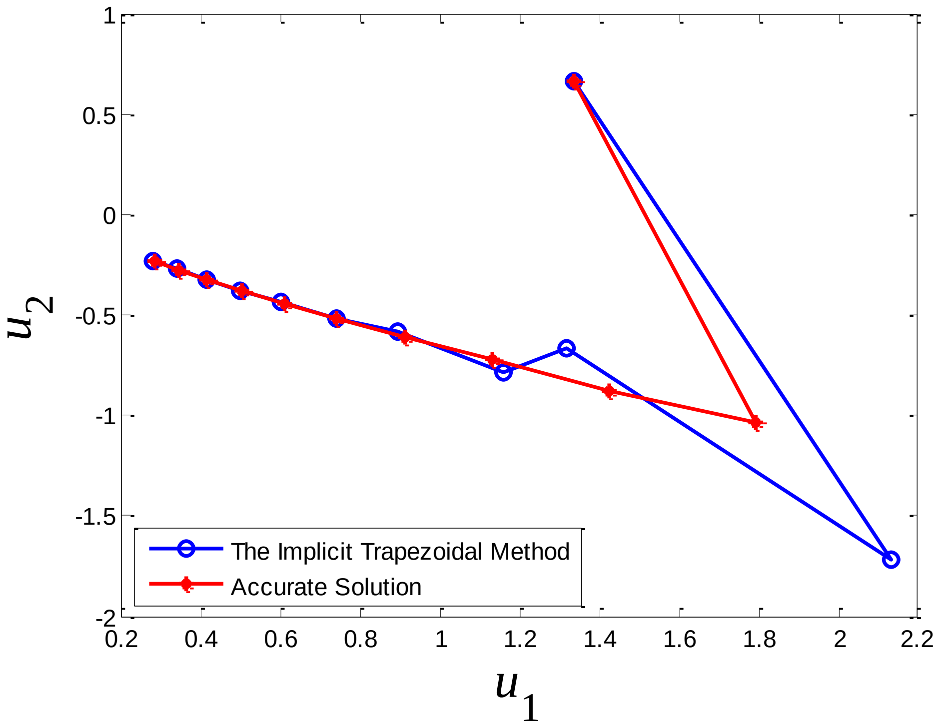
Shanghai Jiao Tong University

Systems of ODEs

Implicit Trapezoidal Method Revisited: Example 6

Implicit Trapezoidal Method with h = 0.1





MATLAB Built-in Functions

MATLAB Built-in Functions for ODEs

ode45	Nonstiff problems, medium accuracy. Use most of the time. It should be the first solver you try.
ode23	Nonstiff problems, low accuracy. Use for large error tolerances or moderately stiff problems.
ode113	Nonstiff problems, low to high accuracy. Use for stringent error tolerances or computationally ODE functions.
ode15s	Stiff problems, low to medium accuracy. Use if ode45 is slow (stiff systems) or there is a mass matrix.
ode23s	Stiff problems, low accuracy. Use for large error tolerances with stiff systems or with a constant mass matrix.
ode23t	Moderately stiff problems, low accuracy. Use for moderately stiff problems where you need a solution without numerical damping.
ode23tb	Stiff problems, low accuracy. Use for large error tolerances with stiff systems or if there is a mass matrix.

Summary

- Symbolic Computation
- Scalar Differential Equation
  - ✓ Euler's Method
  - ✓ The Trapezoid Method
  - ✓ Runge–Kutta Methods
- Systems of Differential Equations
  - ✓ Runge–Kutta Methods
  - ✓ Implicit Trapezoidal Method

Thank You !