高等数值算法与应用(6)

– 常微分方程 (chap7) –

喻文健

Outline

- ODE初值问题
- 单步法（Runger–Kutta法）
- BS23算法及实例
- 洛伦兹吸引子
- 刚性问题
- 事件

ODE初值问题

- ODE初值问题与解法 (电路仿真)
- 一阶常微分方程组 (二体问题)
- ODE初值问题的稳定性分析

常微分方程(ODE)初值问题

- Initial Value Problem (IVP)
  - 常微分方程需求解随时间变化的物理量, 即未知函数$y(t)$
  - 已知条件包括: 微分方程(物理规律)与初始条件

$$\begin{cases} \dfrac{dy(t)}{dt} = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \Longrightarrow \dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y})$$

- 数值求解
  - 生成一系列自变量点 $t_0 < t_1 < \cdots < t_n < t_{n+1} < \cdots$, 及对应近似函数值 $y_0, y_1, \cdots, y_n, y_{n+1}, \cdots$, 即$y_n \approx y(t_n)$　**“步进式”**
  - 步长$h_n = t_{n+1} - t_n$, 固定步长, 或自动变步长(误差控制)
  - 基本依据

$$y(t+h) = y(t) + \int_t^{t+h} f(s, y(s))ds$$

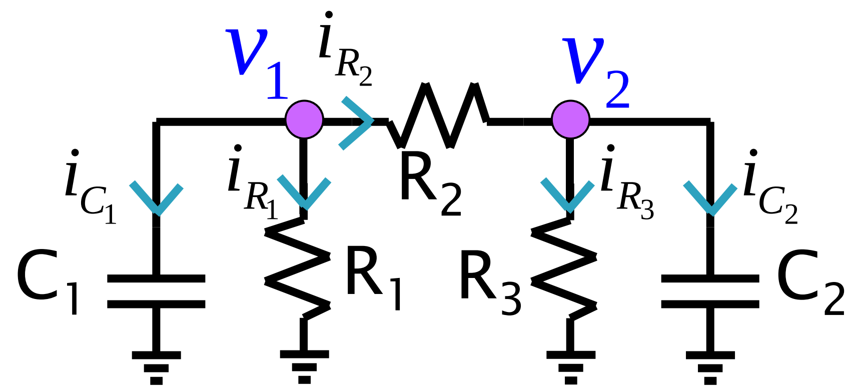
常微分方程 – 例子1

- 含电容元件的电路仿真

电流/电压关系　　节点电流方程

$$i_c = C\frac{dv_c}{dt} \qquad i_{C_1} + i_{R_1} + i_{R_2} = 0$$

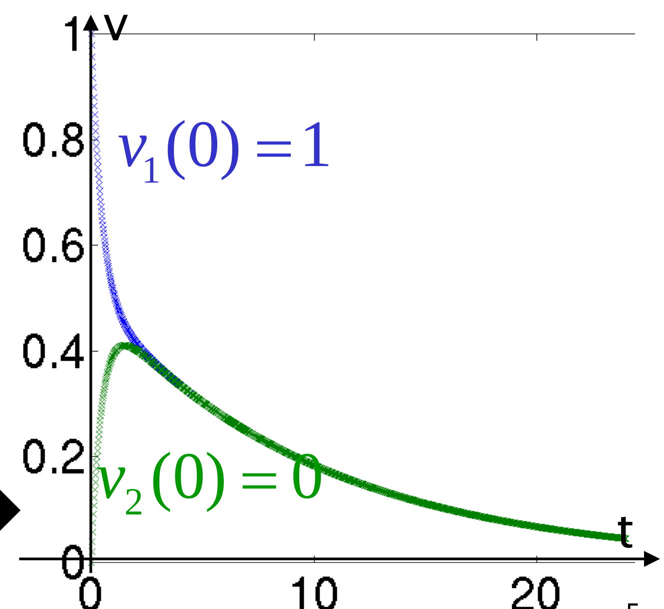
$$i_R = \frac{1}{R}v_R \qquad i_{C_2} + i_{R_3} - i_{R_2} = 0$$



- 通过节点分析法得到微分方程组

$$\begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix}\begin{bmatrix} \frac{dv_1}{dt} \\ \frac{dv_2}{dt} \end{bmatrix} = -\begin{bmatrix} \frac{1}{R_1}+\frac{1}{R_2} & -\frac{1}{R_2} \\ -\frac{1}{R_2} & \frac{1}{R_3}+\frac{1}{R_2} \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

问题的解反映了电容充/放电过程



常微分方程组

- 一阶常微分方程组
  - 问题中含多个待求的时变函数(物理量) ↗ LC电路
  - 方程含二阶或更高阶导数　　　(谐波振荡器) $\ddot{x}(t) = -x(t)$
  - 通过增加未知函数，转为一阶方程组　　设 $y(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix}$

    $\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y})$
  - $\boldsymbol{f}$是个列向量函数　　⇒ $\dot{y}(t) = \begin{bmatrix} y_2(t) \\ -y_1(t) \end{bmatrix}$
  - Matlab中，通过$\boldsymbol{f}$函数描述常微分方程

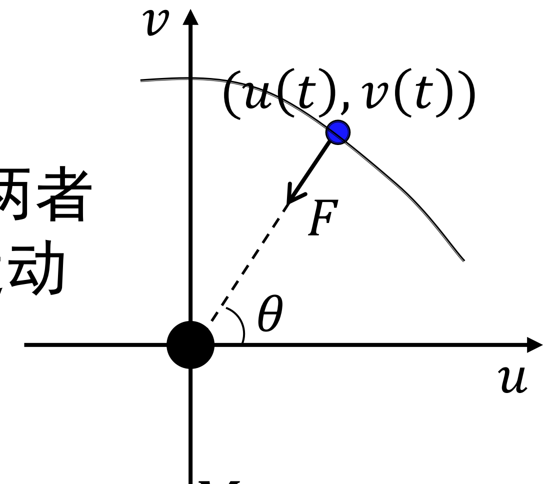
    注意$\boldsymbol{f}$是$\mathbb{R}^{n+1} \to \mathbb{R}^n$函数，且t必须是第一个参数

```
function ydot = harmonic(t,y)
ydot = [y(2); -y(1)];
```

```
>> harmonic = @(t,y) [0 1; -1 0]*y;
```

常微分方程组

- 一阶常微分方程组 – 例子(二体问题)
  - 两个质量分别为m, M的物体, M>>m；两者相互吸引，一个围绕另一个做平面轨道运动
  - 以质量大的物体位置为坐标原点
  - 质量小物体的位置$(u(t),v(t))$, 满足方程



(根据力与运动的分解、牛顿定律)

$$m\ddot{u}=F_u=-k\frac{Mm}{r^2}\cos\theta=-k\frac{Mm}{r^2}\cdot\frac{u}{r}$$

  - 转成一阶方程组

$$\ddot{u}(t) = -u(t)/r(t)^3 \quad \text{(适当设置单位)}$$

同理, $\ddot{v}(t) = -v(t)/r(t)^3,\ r(t)=\sqrt{u(t)^2+v(t)^2}$

$$y(t)=\begin{bmatrix} u(t)\\ v(t)\\ \dot{u}(t)\\ \dot{v}(t)\end{bmatrix} \quad \dot{y}(t)=\begin{bmatrix} \dot{u}(t)\\ \dot{v}(t)\\ -u(t)/r(t)^3\\ -v(t)/r(t)^3\end{bmatrix}$$

```
function ydot = twobody(t,y)
r = sqrt(y(1)^2+ y(2)^2);
ydot = [y(3:4); -y(1:2)/r^3];
```

方程分类及稳定性

- 方程的分类 $\dot{y} = f(t, y)$ 否则，非线性常微分方程
  - 线性常微分方程: $f(t, y) = a(t)y + b(t)$, f是y的线性函数
  - 线性齐次常微分方程，线性齐次常系数微分方程，如$\dot{y} = \lambda y$
- ODE初值问题的<u>敏感性</u>

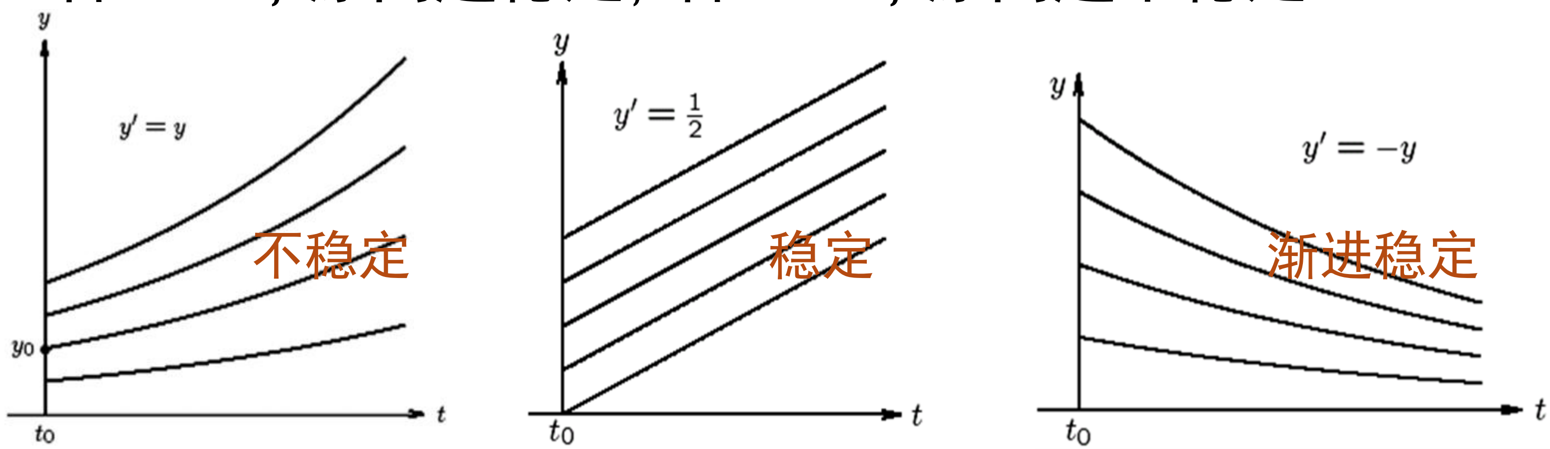
$$\begin{cases} \dot{y} = f(t, y), \\ y(t_0) = y_0 \end{cases}$$

  - 考虑<u>初值</u>发生扰动对结果的影响
  - 结果(解)是函数$y(t)$，特别关心$t \to \infty$时$y(t)$的受影响情况
  - 定义：ODE初值问题的稳定性
  - 若$t \to \infty$时$y(t)$的偏差被控制在有界范围内,稳定(stable)
  - 若$t \to \infty$时$y(t)$的偏差发散为无穷大，不稳定 (unstable)
  - 若$t \to \infty$时$y(t)$的偏差趋于零，渐进稳定(asymptotically stable)

实际的问题应保证其稳定性!

稳定性分析

- 例: “模型问题”的稳定性 $\begin{cases} \dot{y} = \lambda y, \\ y(t_0) = y_0 \end{cases}$
  - 准确解为：$y(t) = y_0 e^{\lambda(t-t_0)}$
  - 扰动后初值$y_0 + \Delta y_0$, 解为$\hat{y}(t) = (y_0 + \Delta y_0)e^{\lambda(t-t_0)}$
  - $\Delta y(t) = \hat{y}(t) - y(t) = \Delta y_0 e^{\lambda(t-t_0)}$ 稳定性取决于λ的值
  - 若$\lambda \le 0$, 原问题稳定; 若$\lambda > 0$, 原问题不稳定



  - 若λ为复数, 稳定性取决于$\mathrm{Re}(\lambda)$

稳定性分析

- 一般的非线性常微分方程 $\dot{y} = f(t,y) \approx \dot{y} = Jy + b(t)$
  - 通过任一点(t_c, y_c)的泰勒展开可讨论局部稳定性
    $$f(t,y) = f(t_c, y_c) + \alpha(t - t_c) + J(y - y_c) + \dots$$
  - 其中$\alpha = \frac{\partial f}{\partial t}(t_c, y_c), \quad J = \frac{\partial f}{\partial y}(t_c, y_c)$ → 决定局部稳定性
  - 对ODE方程组, $\boldsymbol{J}$为雅可比矩阵. 原方程局部近似为线性微分方程组 $\dot{\boldsymbol{y}} = \boldsymbol{J}\boldsymbol{y} + \boldsymbol{b}(t)$
    $$J = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \cdots & \frac{\partial f_2}{\partial y_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial y_1} & \frac{\partial f_n}{\partial y_2} & \cdots & \frac{\partial f_n}{\partial y_n} \end{bmatrix}$$
  - 考察方程$\dot{\boldsymbol{y}} = \boldsymbol{J}\boldsymbol{y}$
  - 设$\boldsymbol{J}$可对角化, $J = V\Lambda V^{-1}$
  - 做变量代换 $Vx = y$
  - ⇒ $\dot{\boldsymbol{x}} = \boldsymbol{\Lambda}\boldsymbol{x}$, 即解一组独立方程 $\dot{x}_k = \lambda_k x_k$
  - 解$\boldsymbol{y}(t)$的局部稳定性由特征值$\lambda_1, \dots, \lambda_n$决定

若有一个的实部>0, 则局部不稳定!

整体稳定性往往很难分析

单步法(Runger-Kutta法)

- 欧拉法、中点法、改进的欧拉法
  局部截断误差与准确度阶数
  Runger-Kutta法
  Matlab解法与举例

欧拉法

$$y(t+h)=y(t)+\int_t^{t+h} f(s,y(s))ds$$

$$y_{n+1}=y(t_n)+\int_{t_n}^{t_n+h} f(s,y(s))ds$$

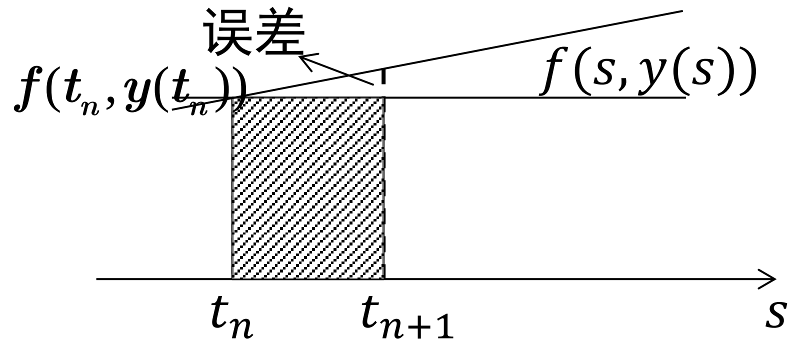
$$\approx y(t_n)+hf(t_n,y(t_n))$$ 左矩形求积公式

- 固定步长的欧拉法

$$y_{n+1} = y_n + hf(t_n, y_n)$$
$$t_{n+1} = t_n + h$$

  - Matlab算法

```
t = t0; y = y0;
while t <= tfinal
   y = y + h*f(t, y)
   t = t + h
end
```



  - 求解方程组时, y0为向量, 且f也返回向量
  - 从推导看出, 当$\dot{y}=f(t,y(t))$为常数时, 该方法精确
  - 若f是t的线性函数, 上述近似积分的误差随h增大而增大
  - 另一个问题: 没有提供误差估计方法, 无法<u>自动确定步长</u>

欧拉法的改进

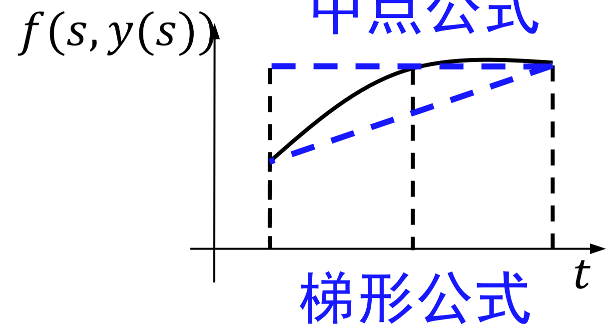
$$y_{n+1} = y_n + \int_t^{t+h} f(s, y(s))ds$$

- 在欧拉法基础上改进
  - 再增加一次函数求值: 数值积分的中点公式或梯形公式
  - 可利用欧拉法估算中点处被积函数值(走半个步长)

两阶段的方法

$$s_1 = f(t_n, y_n)$$
$$s_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2} s_1)$$
$$y_{n+1} = y_n + h s_2$$

(中点法)



  - 用欧拉法估算一步终点处被积函数值

$$s_1 = f(t_n, y_n)$$
$$s_2 = f(t_n + h, y_n + h s_1)$$
$$y_{n+1} = y_n + h \frac{s_1 + s_2}{2}$$

(Heun方法/改进的欧拉法)

- 它们都比欧拉法更准确

局部截断误差

用两个概念来描述截断误差

- *Global* error, which is difference between computed solution and true solution determined by initial data at t_0:

$$e_k = y_k - y(t_k)$$

整体误差

- *Local* error, which is error made in one step of numerical method:

$$\ell_k = y_k - u_{k-1}(t_k),$$

where u_{k-1} is solution through (t_{k-1}, y_{k-1})

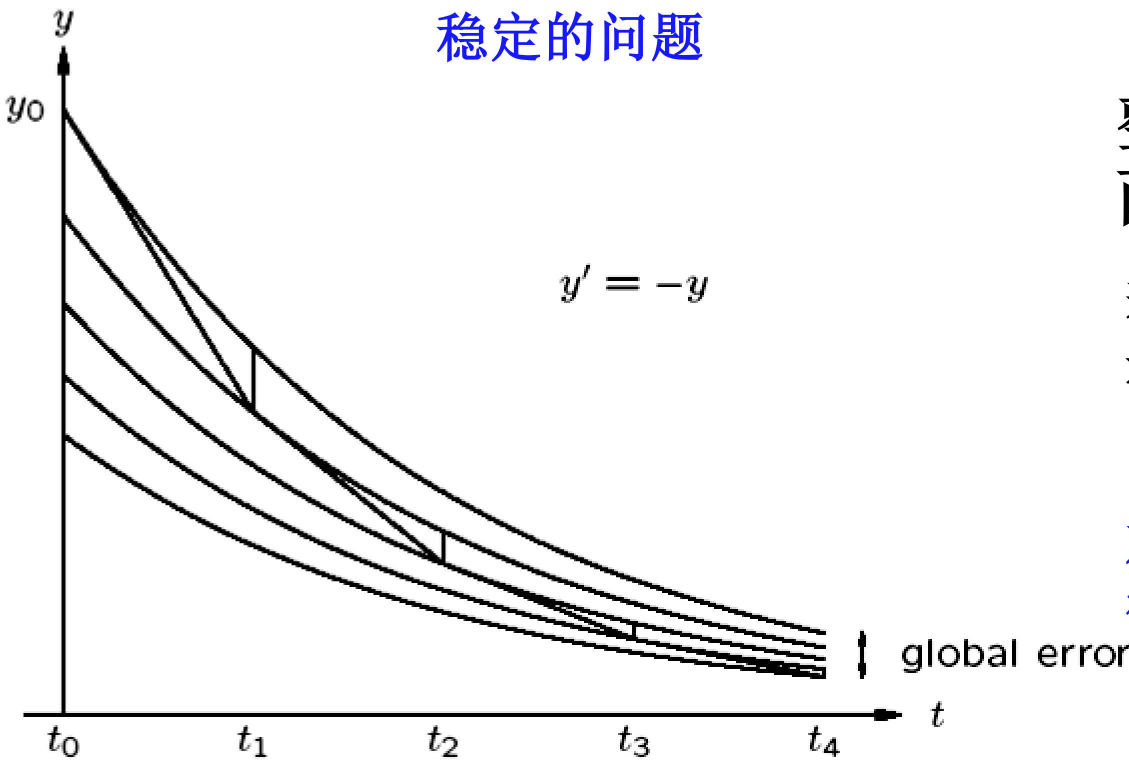
局部截断误差：当前这一步的误差

在前一步得准确解前提下，考虑当前解的误差

Global error is not necessarily sum of local errors

局部截断误差

稳定的问题



整体、局部截断误差两者关系:

若 $\ell_k = \mathcal{O}(h_k^{p+1})$，一般也有 $e_k = \mathcal{O}(h^p)$，

h为平均步长

对不稳定的问题，整体误差往往大于所有局部误差之和！

Accuracy of numerical method is of *order* p if (一般仅能控制局部误差)

$$\ell_k = \mathcal{O}(h_k^{p+1}) \qquad \le C \cdot h_k^{p+1}，\text{当} h_k \text{足够小}$$

Local error per unit step, $\ell_k / h_k = \mathcal{O}(h_k^p)$ 称此方法有p阶准确度

局部截断误差

- 如何分析?
  - 欧拉法 $y_k = y_{k-1} + hf(t_{k-1}, y_{k-1})$

    $y(t_{k-1})e^{\lambda h}$

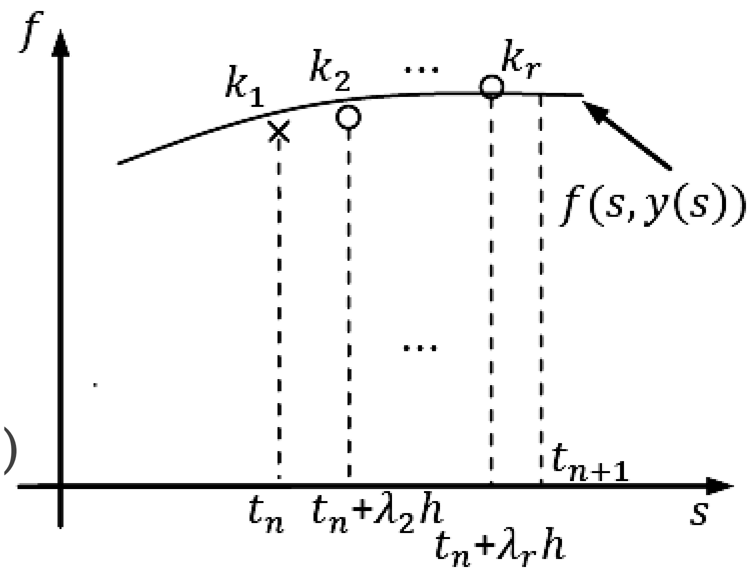
    $l_k = y_k - u_{k-1}(t_k) = y(t_{k-1}) + h\dot{y}(t_{k-1}) - u_{k-1}(t_k)$
  - 考虑模型问题$\dot{y} = \lambda y$ $\quad = y(t_{k-1})[1 + h\lambda - e^{h\lambda}] = O(h^2)$
  - 为1阶准确度，此结论也适用于一般的$\dot{y} = f(t, y)$
  - 改进的欧拉法 $y_k = y_{k-1} + \frac{h}{2}[f(t_{k-1}, y_{k-1}) + f(t_k, y_{k-1} + hf(t_{k-1}, y_{k-1}))]$

    $l_k = y(t_{k-1}) + \frac{h}{2}[\dot{y}(t_{k-1}) + f(t_k, y(t_{k-1}) + h\dot{y}(t_{k-1}))] - u_{k-1}(t_k)$
  - 考虑模型问题$\dot{y} = \lambda y$ $\quad = y(t_{k-1})\left[1 + \frac{h}{2}(\lambda + \lambda + \lambda \cdot h\lambda) - e^{h\lambda}\right] = O(h^3)$
  - 具有2阶准确度

Runge-Kutta法

- 单步法(Runge-Kutta法)
  - **思想**: 对t_n, t_{n+1}之间几个不同的t值估算f函数的值, 再用它们的线性组合近似计算积分
  - 若用两个准确度不同公式算y_{n+1}, 则可估计误差、自动确定步长
  - 经典(四阶)Runge-Kutta法 (1905)

$$y_{n+1} = y_n + \int_t^{t+h} f(s, y(s))ds$$



$$\begin{aligned}
s_1 &= f(t_n, y_n) \\
s_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}s_1) \\
s_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}s_2) \\
s_4 &= f(t_n + h, y_n + hs_3)
\end{aligned}$$

$$y_{n+1} = y_n + \frac{h}{6}(s_1 + 2s_2 + 2s_3 + s_4)$$

本身不提供误差估计;
有时使用步长h, h/2分别计算得到误差估计

Runge-Kutta法

- 更普遍的Runge-Kutta法

$$y_{n+1} = y_n + \int_{t_n}^{t_n+h} f(s, y(s))ds$$

  - 在$[t_n, t_n+h]$上取k个不同的t值 (k个阶段)，估算每个对应的f函数值 (引入参数$\alpha_i, \beta_{i,j}$)

时间值	y函数近似值	f函数近似值
t_n	y_n	$f(t_n, y_n) \equiv s_1$
$t_n + \alpha_2 h$	$y_n + \alpha_2 h s_1$	$f(t_n + \alpha_2 h, y_n + \alpha_2 h s_1) \equiv s_2$
$t_n + \alpha_3 h$	$y_n + h\sum_{j=1}^{2}\beta_{3,j}s_j$	$f\left(t_n + \alpha_3 h, y_n + h\sum_{j=1}^{2}\beta_{3,j}s_j\right) \equiv s_3$
…	…	…

  - 计算一步的公式(求积公式) $y_{n+1} = y_n + h\sum_{i=1}^{k}\gamma_i s_i$
  - 若算另一个公式，两者差(斜率组合)算误差 $e_{n+1} = h\sum_{i=1}^{k}\delta_i s_i$

Runge-Kutta法

$$y_{n+1} = y_n + h\sum_{i=1}^{k} \gamma_i s_i$$

- 如何求Runge-Kutta法的参数 $s_i = f(t_n + \alpha_i h, y_n + h\sum_{j=1}^{i-1} \beta_{i,j} s_j)$
  - 四组参数$\alpha_i, \beta_{i,j}, \gamma_i, \delta_i$的具体值根据准确度阶数的要求定
  - k阶段公式: 计算k次f(t,y)函数
  - 若k=1~4, 则k阶段公式可达到k阶准确度,
    但要达到5阶准确度, 则至少需要6阶段公式 (计算量大)
- Matlab中的ODE求解器
  - 很多是变步长的单步法 (使用两种单步法公式)
  - 名字: odennxx, nn为两个数字, 代表两个公式的阶数;
    xx表示方法的某种属性，可能没有
  - [T,Y,TE,YE,IE] = ode23(odefun,tspan,y0,options)

实例与Matlab demo

- 单个ODE：分析稳定性, Matlab求解

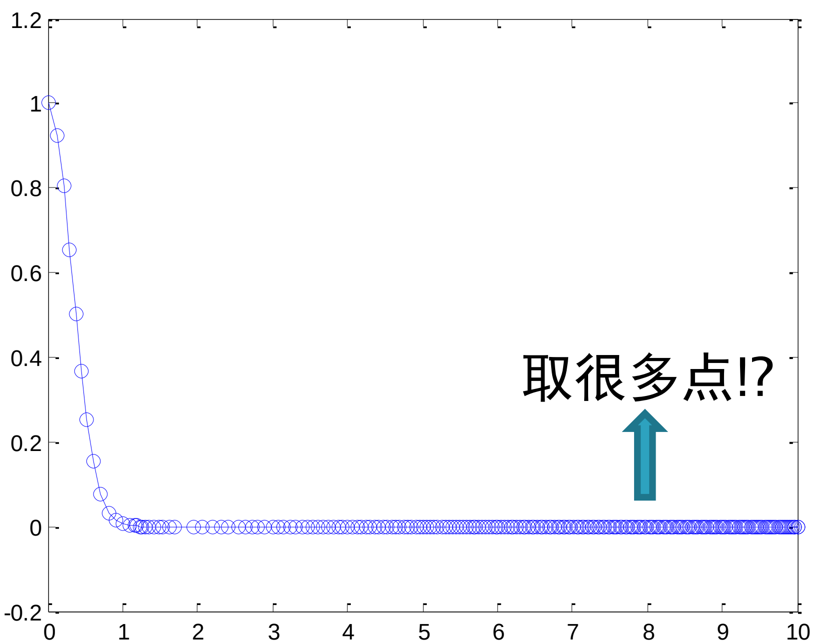
$$\begin{cases} y' = -10yt \\ y(0) = 1 \end{cases} \qquad J_f = -10t$$

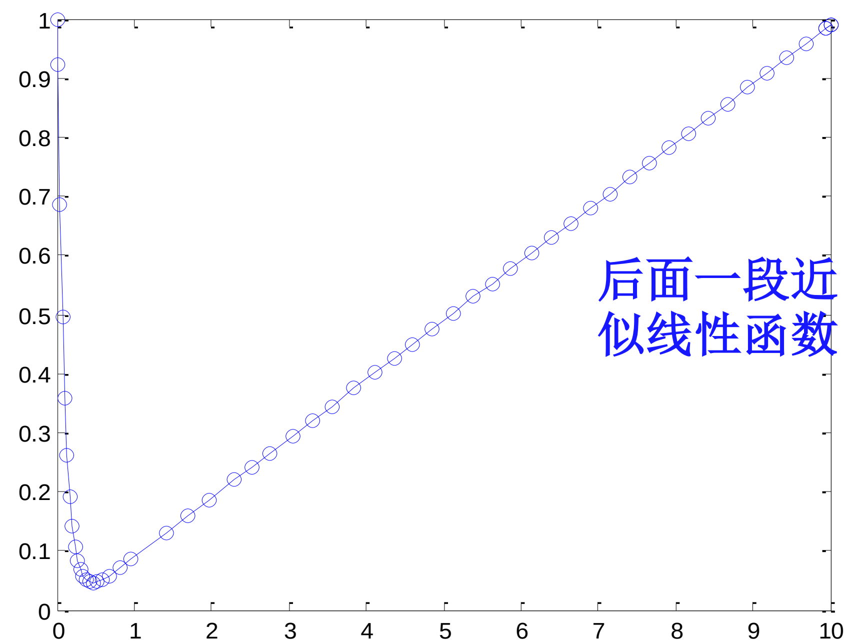
$$\begin{cases} y' = -10y + t \\ y(0) = 1 \end{cases} \qquad J_f = -10$$

True solution: $y(t) = e^{-5t^2}$

f=@(t, y) -10*y*t; ode23(f, [0,10], 1);

Matlab: ode_1





实例与Matlab demo

- 常微分方程组: 牛顿第二定律

$$\begin{cases} y_1' = y_2 \\ y_2' = 6t \\ y_1(0) = 0 \\ y_2(0) = 1 \end{cases}$$

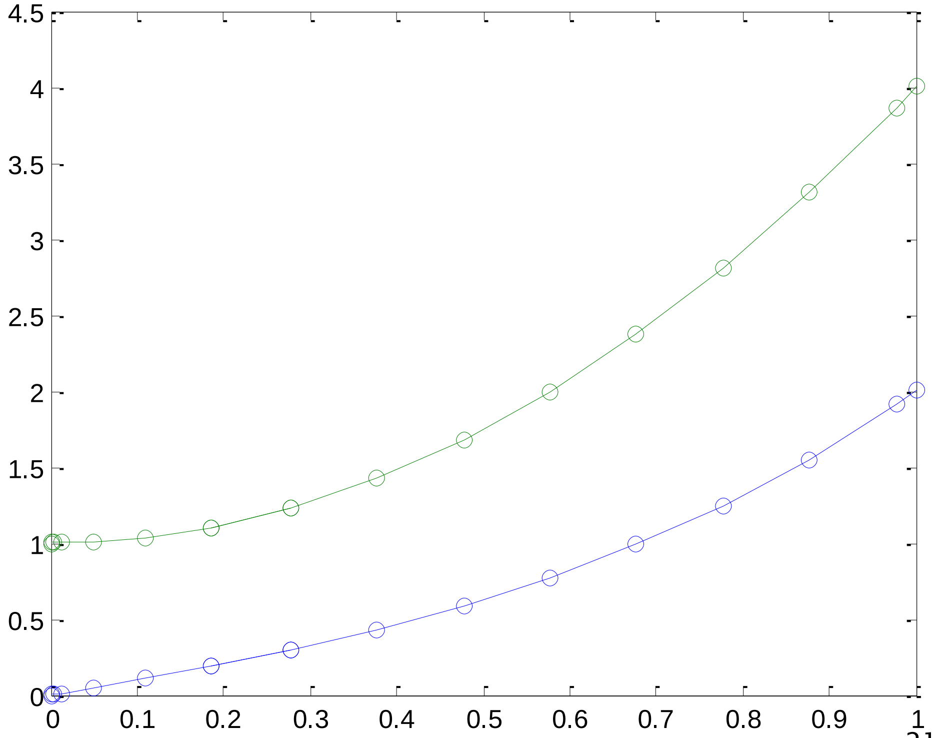
有解析解

稳定吗？

ode23(@myode2, [0, 1], **[0; 1]**);

```
function ydot = myode2(t,y);
ydot= [y(2); 6*t];   %列向量
```

Matlab: ode_2



BS23算法及实例

- 自动Runge-Kutta法：BS23算法
  ode23tx程序与演示

自动的Runge-Kutta方法

- 基本思想
  - 两个公式算出的 $\hat{y}_{k+1}$ 和 y_{k+1} 分别为p, p+1阶准确度

$$\hat{y}_{k+1} - u_k(t_k + h) = O(h^{p+1}) = c_{p+1}h^{p+1} + O(h^{p+2}) \Longrightarrow \hat{y}_{k+1} - y_{k+1} \approx c_{p+1}h^{p+1}$$

  - 取准确度更高的值 y_{k+1} 作这一步的结果

err= 可调整h控制err，即控制局部误差

- BS23算法
  - Bogachi & Shampine在[AML'1989]中提出，2-3阶R-K方法

3阶Ralston公式：

$$y_{k+1} = y_k + \frac{h}{9}(2s_1 + 3s_2 + 4s_3)$$

$$s_1 = f(t_k, y_k)$$

$$s_2 = f(t_k + h/2,\ y_k + (h/2)s_1)$$

$$s_3 = f(t_k + 3h/4,\ y_k + (3h/4)s_2)$$

2阶公式：

$$s_4 = f(t_{k+1},\ y_{k+1})$$

$$\hat{y}_{k+1} = y_k + \frac{h}{24}(7s_1 + 6s_2 + 8s_3 + 3s_4)$$

未引入新计算量，且2阶公式的稳定域包含了3阶公式的稳定域[AML'1989]

误差估计式：

$$y_{k+1} - \hat{y}_{k+1} = \frac{h}{72}(-5s_1 + 6s_2 + 8s_3 - 9s_4)$$

BS23算法及ode23tx

- BS23算法

应该与y值有关:
相对阈值rtol

输入：$f(t,y)$, t_0, y_0, 终点b, 误差阈值tol; 输出：$y_n\ (n=1,2,\cdots)$.

$s_1 := f(t_0, y_0)$; h: =初始步长; n: =0 ;

While $t_n \le b$, **do**

$s_2 := f(t_n + h/2\,, y_n + s_1 \cdot h/2)$;

$s_3 := f(t_n + 3h/4\,, y_n + s_2 \cdot 3h/4)$;

$t_{n+1} := t_n + h$;

$y_{n+1} := y_n + (2s_1 + 3s_2 + 4s_3) \cdot h/9$;

$s_4 := f(t_{n+1}, y_{n+1})$;

err: $= |(-5s_1 + 6s_2 + 8s_3 - 9s_4) \cdot h/72|$;

If err≤ tol **then**

$s_1 := s_4$;

n: =n+1 ;

End

$h := \alpha \cdot \sqrt[3]{\mathrm{tol}/\mathrm{err}} \cdot h$;   { α为小于1的某个值，如0.8 }

End

- 每步计算3次f()函数
- 若e_{n+1}小于阈值，则进入下一步(s_4可以复用)，否则减小步长重算当前步

$$\frac{err'}{err} \approx \frac{h'^3}{h^3}$$

误差 $err \approx c \cdot h^3$

改变步长 $err' \approx c \cdot h'^3$

要使err'<tol

$h' < \sqrt[3]{\mathrm{tol}/err} \cdot h$

BS23算法及ode23tx

- ode23tx程序 Matlab中ode23的简化版本
  - [tout,yout] = ode23tx(F, tspan, y0, arg4, varargin)

相对误差阈值
或option结构

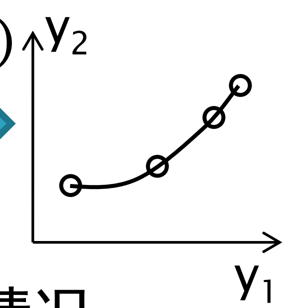
  - 程序细节: 初始步长, 绝对阈值/相对阈值, 步长变化

回忆
optimset

  - 相关命令: opt=odeset(outputfcn: odeplot, odephas2)

- 通过实例了解ode23tx

相位图




```
>> F= @(t,y) 0;  ode23tx(F, [0, 10], 1)
```

  - 增加输出每个计算步的t, tnew, 看多次预测步长的情况
  - 修改F函数、改变精度阈值
  - 谐波振荡器:

```
>> F=@(t,y) [y(2); -y(1)];
>> ode23tx(F, [0, 2*pi], [1; 0])
```

  - 平面相位图 (红色标记?)

```
>> opt=odeset('outputfcn', @odephas2);
>> ode23tx(F, [0, 2*pi], [1; 0], opt);
```

  - 二体问题

改变初始条件、看相位图

洛伦兹吸引子

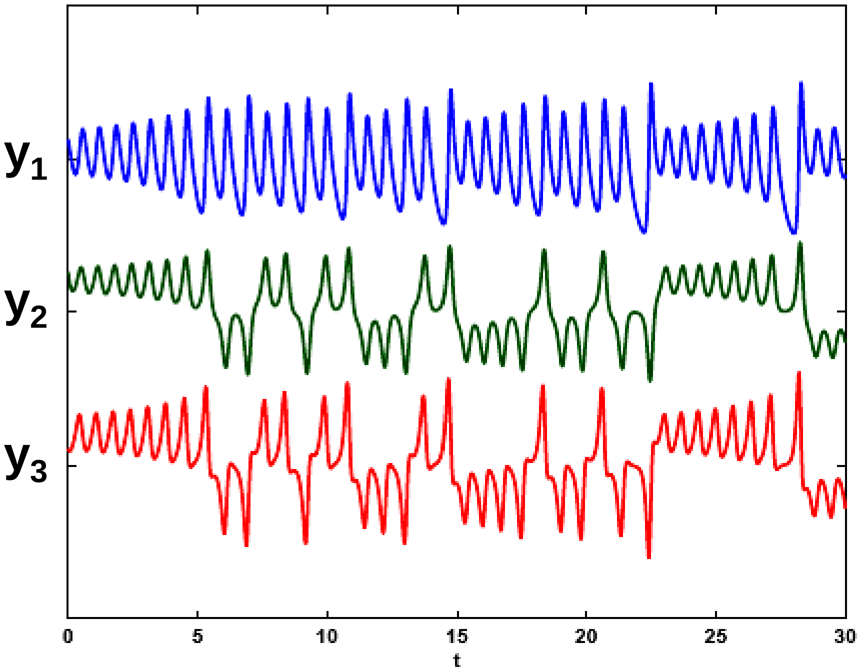
- 问题背景
- 混沌与吸引子
- 程序演示
- 双联摆

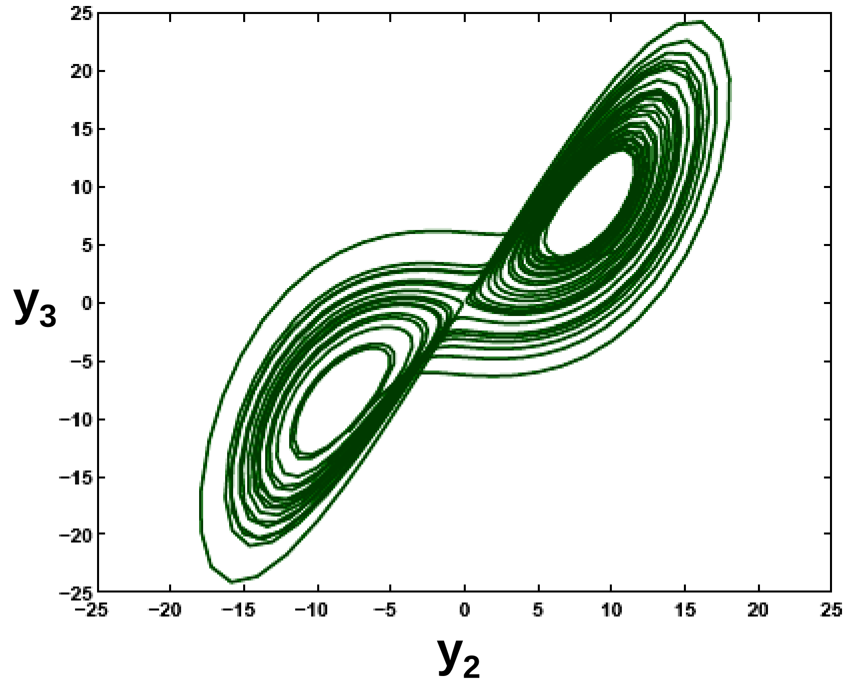
问题背景

- Lorenz吸引子
  - 1963年，MIT的Edward Lorenz研究地球大气的流体模型时发现的现象
  - n=3的常微分方程组 $\dot{y} = Ay$，$A = \begin{bmatrix} -\beta & 0 & y_2 \\ 0 & -\sigma & \sigma \\ -y_2 & \rho & -1 \end{bmatrix}$
  - $y_1(t)$与大气对流有关, $y_2(t)$和$y_3(t)$与水平和垂直的温度变化有关
  - σ: Prandtl数, β: 与区域的几何有关, ρ: 规范化的Rayleigh数
  - 看似简单的非线性使系统表现非常复杂，虽然参数是确定的，但会产生“混沌”现象
  - 三维空间中y(t)的轨迹混乱地在两个点（吸引子）之间往返，有界但无周期，不收敛也不自交

解的现象

- Lorenz吸引子





  - 怎么解释？怎么用Matlab求解？
  - 矩阵$\boldsymbol{A}$有可能奇异，若有某个非零的$\boldsymbol{y}(0)$使$\dot{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{y} = \boldsymbol{0}$，则$\boldsymbol{y}(0)$为ODE的不动点

    实验表明这些不动点不稳定，接近它们时会被排斥

理论分析

- Lorenz吸引子

$$A = \begin{bmatrix} -\beta & 0 & y_2 \\ 0 & -\sigma & \sigma \\ -y_2 & \rho & -1 \end{bmatrix} \Longrightarrow \boldsymbol{y}_2 = \pm\sqrt{\beta(\rho-1)} \text{ 时奇异}$$

  - 这是初值$\boldsymbol{y}(0)$的第2个分量，而$\boldsymbol{y}(0)$满足$\boldsymbol{A}\boldsymbol{y}(0) = \boldsymbol{0}$
  - 易知：$$\begin{bmatrix} -\beta & 0 & y_2 \\ 0 & -\sigma & \sigma \\ -y_2 & \rho & -1 \end{bmatrix} \begin{bmatrix} \rho-1 \\ \boldsymbol{y}_2 \\ \boldsymbol{y}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$
  - 两个吸引子：$\left[\rho-1,\ \sqrt{\beta(\rho-1)},\ \sqrt{\beta(\rho-1)}\right]^T, \left[\rho-1,\ -\sqrt{\beta(\rho-1)},\ -\sqrt{\beta(\rho-1)}\right]^T$
  - 若参数σ=10, β=8/3, ρ=28, 吸引子为**(27, ±8.5, ±8.5)**
  - 当ODE的初值不在吸引子位置时，看到混沌现象

程序与演示

演示程序：**lorenzgui**

mylorenz.m

- Lorenz吸引子

```
rho = 28;
sigma = 10;
beta = 8/3;
eta = sqrt(beta*(rho-1));
A = [ -beta     0      eta
         0   -sigma   sigma
      -eta    rho     -1   ];

yc = [rho-1; eta; eta];
y0 = yc + [0; 0; 3];

tspan = [0 Inf];
opts = odeset('reltol',1.e-6,'outputfcn',@lorenzplot);
ode45(@lorenzeqn, tspan, y0, opts, A);
-----------------
function ydot = lorenzeqn(t,y,A)
A(1,3) = y(2);
A(3,1) = -y(2);
ydot = A*y;
```

% $\sqrt{\beta(\rho-1)}$

% 形成矩阵**A**

% 吸引子位置

% 初始值

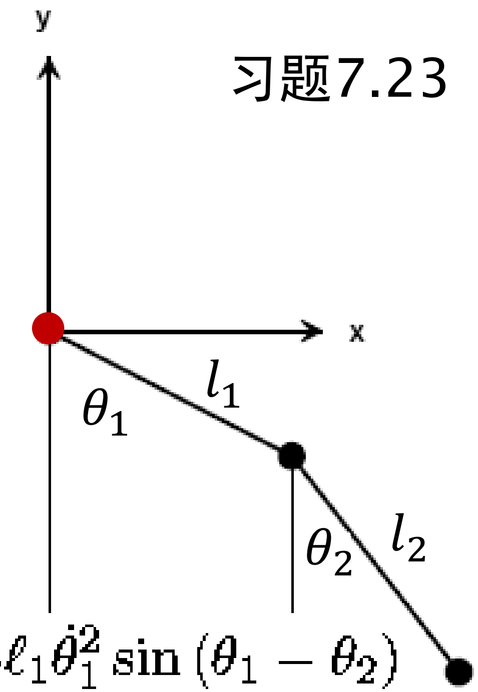
设置输出处理函数

% **A**为额外参数

相位显示：opts＝odeset('outputfcn', @odephas2, 'outputSel', [1, 2]);

另一个形成混沌的例子 – 双联摆

习题7.23



- 双联摆的运动
- 两个摆锤(重物), 刚性杆的重量可忽略
- 不考虑摩擦力,运动不会停止, 且在初始角度较大时摆锤的轨迹呈现混沌现象

$$\begin{cases}(m_1+m_2)\ell_1\ddot{\theta}_1 + m_2\ell_2\ddot{\theta}_2\cos(\theta_1-\theta_2) = \\ \qquad -g(m_1+m_2)\sin\theta_1 - m_2\ell_2\dot{\theta}_2^2\sin(\theta_1-\theta_2), \\ m_2\ell_1\ddot{\theta}_1\cos(\theta_1-\theta_2) + m_2\ell_2\ddot{\theta}_2 = -gm_2\sin\theta_2 + m_2\ell_1\dot{\theta}_1^2\sin(\theta_1-\theta_2)\end{cases}$$

设$l_1 = l_2 = m_1 = m_1 = 1$, 做变量代换得到一阶方程组

$$\begin{cases}\dot{u}_1 = u_3, \\ \dot{u}_2 = u_4, \\ 2\dot{u}_3 + c\dot{u}_4 = -g\sin u_1 - su_4^2 \\ c\dot{u}_3 + \dot{u}_4 = -g\sin u_2 + su_3^2\end{cases}$$

求解微分方程初值问题, 得到摆锤的运动规律

Matlab演示swinger

刚性问题

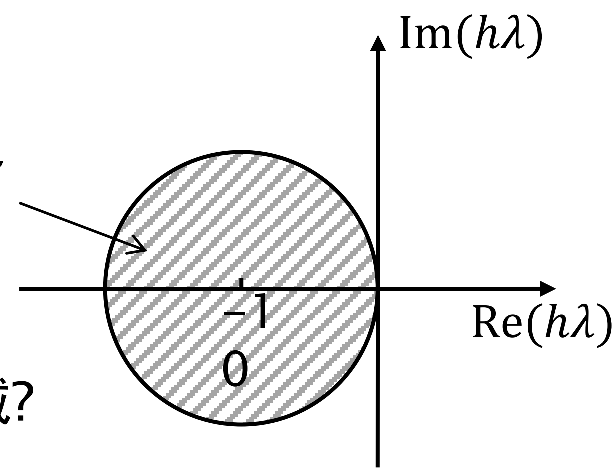
- ODE数值解法的稳定性
- 刚性问题与刚性解法

数值解法的稳定性

- 方法的稳定性: 解函数近似值y_n存在误差, 在后续递推计算过程中, 它会如何传播?
- 定义: 若在节点t_n上的函数近似值有扰动δ_n, 由它引起的后续节点上的误差δ_m ($m>n$)满足$|\delta_m| \le |\delta_n|$, 则该方法稳定
- 考虑截断误差. **以欧拉法为例, 针对模型问题**$\dot{y} = \lambda y$
- 计算公式为$y_{n+1} = y_n + h\lambda y_n = (1 + h\lambda)y_n$
- y_n上的扰动δ_n引起y_{n+1}的误差: $\delta_{n+1} = (1 + h\lambda)\delta_n$
- 要保证欧拉法稳定

$\Longleftrightarrow$ $|1 + h\lambda| \le 1$

若λ为复数, $h\lambda$在复平面内的取值范围



若λ为实数($\lambda<0$), $h \le \frac{-2}{\lambda}$

稳定域

其他显式R−K方法的稳定域?

数值解法的稳定性

- 稳定的隐格式方法
  - 向后欧拉法: $y_{n+1} = y_n + h_n f(t_{n+1}, y_{n+1})$
  - 梯形法: $y_{n+1} = y_n + \frac{1}{2}h_n[f(t_n, y_n) + f(t_{n+1}, y_{n+1})]$
  - 采用隐格式方法，每步计算都要解(非线性)方程
  - 向后欧拉法的稳定性 (考虑模型问题: $\dot{y} = \lambda y$) $(\mathrm{Re}(\lambda) \le 0)$

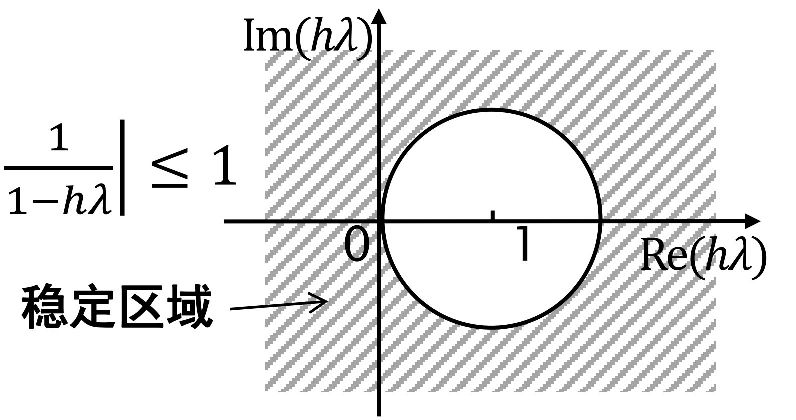
$$y_{n+1} = y_n + h\lambda y_{n+1} \Longrightarrow y_{n+1} = \frac{1}{1-h\lambda} y_n$$

设y_n存在扰动δ_n，引起y_{n+1}的误差为

$$\delta_{n+1} = \frac{1}{1-h\lambda}\delta_n$$ ➡ 稳定的条件是: $\left|\frac{1}{1-h\lambda}\right| \le 1$

即$|h\lambda - 1| \ge 1$ 对任意的h都满足

无条件稳定(unconditionally stable)!



刚性 (stiff)

- 刚性
  - 常微分方程数值解法中一个微妙而重要的概念
  - 取决于微分方程和所采用的数值方法
  - 一个问题被称为刚性的是指，其准确解随时间缓慢变化，但它附近存在变化很快的解，进而数值求解过程必须采用小步长来获得满意的结果。
  - 刚性本身是一个效率的问题，若不关心计算时间可以不理会
- Matlab中ODE求解程序的分类
  - 非刚性求解器: ode23, ode45(4阶和5阶的R-K公式), ode113
  - 刚性求解器: ode23s, ode15s, ode23t, ode23tb

    往往采用隐格式公式，稳定性好，步长大。例如梯形公式

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

例：火焰蔓延模型

"当点燃一根火柴时，火焰迅速增大直到一个临界体积，然后维持这一体积不变，此时火焰内部燃烧耗费的氧气和其表面现存的氧气达到了一种平衡。" 火球半径y(t)满足：

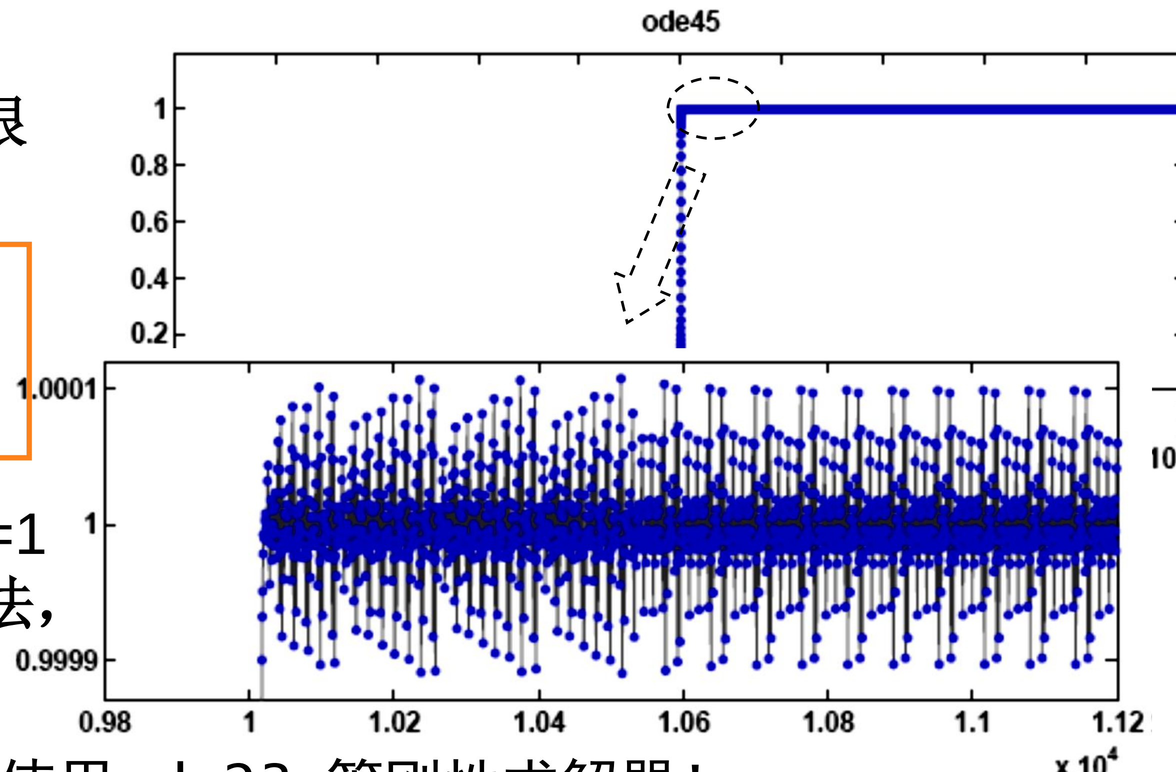
20000

$$\begin{cases} y' = y^2 - y^3 \\ y(0) = \delta \end{cases}$$

设初始半径δ=0.0001，求解 $y(t),\ 0 \le t \le 2/\delta$

用ode45求解，步长很小，效率很低

```
deta= 0.0001;
f= @(t, y) y^2-y^3;
ode45(f, [0, 2/deta], deta);
```



分析：非线性ODE在y(t)=1处Eig(**J**)= –1, 若用Euler法，步长上限为2

应使用ode23s等刚性求解器！

刚性常微分方程(组)

$$h < \frac{-2}{\lambda}$$

- **刚性常微分方程**：　局部线性化得: $\dot{y} = Jy + b(t)$
  J实部的绝对值很大, 或虚部绝对值很大(震荡型)
- 无论解函数曲线是否变化缓慢，都需要使用较小步长保证计算的稳定性 (若用大步长, 数值解会很快偏离)
- **刚性常微分方程组**: Jacobi矩阵特征值大小相差悬殊.
  解函数趋于稳定的时间由最小特征值决定，而保持稳定的步长上限由最大特征值决定 (需用小步长求解 缓慢变化的曲线)
- 从峡谷中下山的比喻：
  显式方法——按局部斜率找最快下降方向，结果每步都跨越峡谷
  隐式方法——盯着更远的目标, 仔细预测每步怎么走

事件(event)

- Matlab求解器的事件功能
  实例演示

事件 (event)

- 如何确定ode求解的区间终点(t_{final})?
  - 例1: 物体受重力(以及空气阻力)作用下落到地面的时间
  - 例2: 二体问题中, 较轻的物体运动的周期(时间)
- Matlab的ode求解器的事件功能
  - [T,Y,TE,YE,IE] = ode23(odefun,tspan,y0,options)

$\{t_n\}, \{y_n\}$ $\quad t_*, y_*$ $\quad \dot{y} = f(t, y)$ $\quad y(t_0) = y_0$ 设置触发事件的时间满足的方程

  - 触发事件的方程 $g(t_*, y(t_*)) = 0$
  - g函数的定义:

值为1,0; 事件触发后求解过程终止否?

值为0/1/−1; 1/−1表示g函数递增/递减过程

function [gstop, isterminal, direction]= g(t, y)

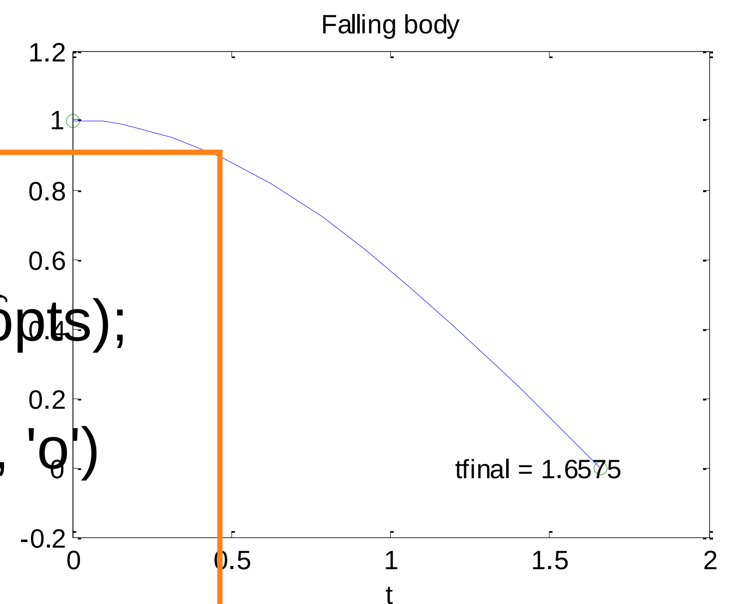
  - 例1: g(t, y)= y_1 $\quad$ ode方程: $\ddot{y} = -1 + \dot{y}^2$ $\quad \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ -1 + y_2^2 \end{bmatrix}$

```
gstop = y(1); isterminal = 1; direction = 0;
```

事件 (event)

- 例1:物体受重力(以及空气阻力)作用下落
  - 微分方程 $\ddot{y} = -1 + \dot{y}^2$ $\quad \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ -1 + y_2^2 \end{bmatrix}$
  - 初始条件: $y(0) = 1, \dot{y}(0) = 0$
  - 求t为何值时y(t)=0？

```
>> opts = odeset('events',@g);
>> y0 = [1; 0];
>> [t,y,tfinal] = ode45(@f,[0 Inf], y0, opts);
>> tfinal
>> plot(t, y(:,1), '-', [0 tfinal], [y0(1) 0], 'o')
>> xlabel('t');   ylabel('y')
>> title('Falling body')
>> text(1.2, 0, ['tfinal = ' num2str(tfinal)])
```



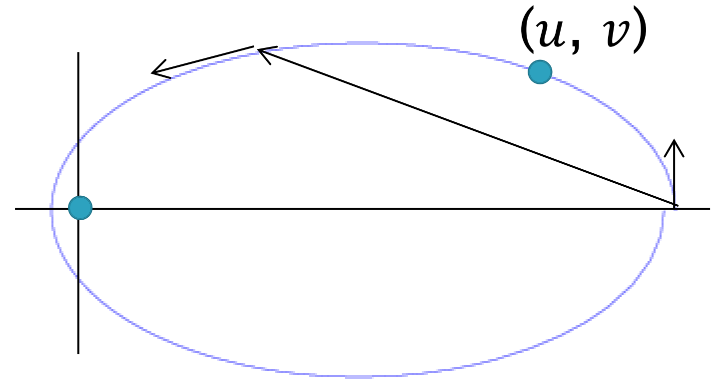
fall_body.m

- 例2: 二体问题, 运动的周期 (环绕一周的时间)

事件 (event)

- 例2: 二体问题, 运动的周期
  - 微分方程中未知函数
  - 初始条件:
  - $y(0)=[1, 0, 0, 0.3]^T$
  - 怎么设置事件g(t, y)函数?

$$y(t)=\begin{bmatrix} u(t) \\ v(t) \\ \dot{u}(t) \\ \dot{v}(t) \end{bmatrix}$$



到初始点的距离d=y(1:2)– y0(1:2)为零向量? d首次达到极小

- $\mathrm{d}(\|\boldsymbol{d}\|^2)/\mathrm{d}t=0$, 即$\boldsymbol{d}^T\dot{\boldsymbol{d}}=0$ 首次极小为g(t,y)从负变为0

```
function [val,isterm,dir] = gstop(t,y,y0)
d = y(1:2)-y0(1:2);
v = y(3:4);
val = d'*v;
isterm = 1; dir = 1;
```

注意:
输入参数还有y0 ;
f(t,y)定义需保持一致,
也增加第3个参数

设置高的ode求解精度, 轨道会重合, 见orbit.m