第十五章　常微分方程的解法

建立微分方程只是解决问题的第一步，通常需要求出方程的解来说明实际现象，并加以检验。如果能得到解析形式的解固然是便于分析和应用的，但是我们知道，只有线性常系数微分方程，并且自由项是某些特殊类型的函数时，才可以得到这样的解，而绝大多数变系数方程、非线性方程都是所谓“解不出来”的，即使看起来非常简单的方程如 $\frac{dy}{dx}=y^2+x^2$。于是对于用微分方程解决实际问题来说，数值解法就是一个十分重要的手段。

§1　常微分方程的离散化

下面主要讨论一阶常微分方程的初值问题，其一般形式是

$$\begin{cases}\dfrac{dy}{dx}=f(x,y) & a\le x\le b\\ y(a)=y_0\end{cases} \tag{1}$$

在下面的讨论中，我们总假定函数 $f(x,y)$ 连续，且关于 y 满足李普希兹(Lipschitz)条件，即存在常数 L，使得

$$|f(x,y)-f(x,\bar{y})|\le L|y-\bar{y}|$$

这样，由常微分方程理论知，初值问题(1)的解必定存在唯一。

所谓数值解法，就是求问题（1）的解 $y(x)$ 在若干点

$$a=x_0<x_1<x_2<\cdots<x_N=b$$

处的近似值 $y_n(n=1,2,\cdots,N)$ 的方法，$y_n(n=1,2,\cdots,N)$ 称为问题（1）的数值解，$h_n=x_{n+1}-x_n$ 称为由 x_n 到 x_{n+1} 的步长。今后如无特别说明，我们总取步长为常量 h。

建立数值解法，首先要将微分方程离散化，一般采用以下几种方法：

（i）用差商近似导数

若用向前差商 $\frac{y(x_{n+1})-y(x_n)}{h}$ 代替 $y'(x_n)$ 代入（1）中的微分方程，则得

$$\frac{y(x_{n+1})-y(x_n)}{h}\approx f(x_n,y(x_n))\quad(n=0,1,\cdots,N-1)$$

化简得

$$y(x_{n+1})\approx y(x_n)+hf(x_n,y(x_n))$$

如果用 $y(x_n)$ 的近似值 y_n 代入上式右端，所得结果作为 $y(x_{n+1})$ 的近似值，记为 y_{n+1}，则有

$$y_{n+1}=y_n+hf(x_n,y_n)\quad(n=0,1,\cdots,N-1) \tag{2}$$

这样，问题（1）的近似解可通过求解下述问题

$$\begin{cases}y_{n+1}=y_n+hf(x_n,y_n)\quad(n=0,1,\cdots,N-1)\\ y_0=y(a)\end{cases} \tag{3}$$

得到，按式（3）由初值 y_0 可逐次算出 $y_1,y_2,\cdots,y_N$。式（3）是个离散化的问题，称为差分方程初值问题。

需要说明的是，用不同的差商近似导数，将得到不同的计算公式。

（ii）用数值积分方法

将问题（1）的解表成积分形式，用数值积分方法离散化。例如，对微分方程两端积分，得

$$y(x_{n+1})-y(x_n)=\int_{x_n}^{x_{n+1}} f(x,y(x))dx \quad (n=0,1,\cdots,N-1) \tag{4}$$

右边的积分用矩形公式或梯形公式计算。

（iii）Taylor 多项式近似

将函数 $y(x)$ 在 x_n 处展开，取一次 Taylor 多项式近似，则得

$$y(x_{n+1})\approx y(x_n)+hy'(x_n)=y(x_n)+hf(x_n,y(x_n))$$

再将 $y(x_n)$ 的近似值 y_n 代入上式右端，所得结果作为 $y(x_{n+1})$ 的近似值 y_{n+1}，得到离散化的计算公式

$$y_{n+1}=y_n+hf(x_n,y_n)$$

以上三种方法都是将微分方程离散化的常用方法，每一类方法又可导出不同形式的计算公式。其中的 Taylor 展开法，不仅可以得到求数值解的公式，而且容易估计截断误差。

§2 欧拉（Euler）方法

2.1 Euler 方法

Euler 方法就是用差分方程初值问题（3）的解来近似微分方程初值问题（1）的解，即由公式（3）依次算出 $y(x_n)$ 的近似值 $y_n(n=1,2,\cdots,N-1)$。这组公式求问题（1）的数值解称为向前 Euler 公式。

如果在微分方程离散化时，用向后差商代替导数，即 $y'(x_{n+1})\approx\dfrac{y(x_{n+1})-y(x_n)}{h}$，则得计算公式

$$\begin{cases} y_{n+1}=y_n+hf(x_{n+1},y_{n+1}) \quad (n=0,1,\cdots,N-1) \\ y_0=y(a) \end{cases} \tag{5}$$

用这组公式求问题（1）的数值解称为向后 Euler 公式。

向后 Euler 法与 Euler 法形式上相似，但实际计算时却复杂得多。向前 Euler 公式是显式的，可直接求解。向后 Euler 公式的右端含有 y_{n+1}，因此是隐式公式，一般要用迭代法求解，迭代公式通常为

$$\begin{cases} y_{n+1}^{(0)}=y_n+hf(x_n,y_n) \\ y_{n+1}^{(k+1)}=y_n+hf(x_{n+1},y_{n+1}^{(k)}) \qquad (k=0,1,2,\cdots) \end{cases} \tag{6}$$

2.2 Euler 方法的误差估计

对于向前 Euler 公式（3）我们看到，当 $n=1,2,\cdots,N-1$ 时公式右端的 y_n 都是近似的，所以用它计算的 y_{n+1} 会有累积误差，分析累积误差比较复杂，这里先讨论比较简单的所谓局部截断误差。

假定用（3）式时右端的 y_n 没有误差，即 $y_n=y(x_n)$，那么由此算出

$$y_{n+1}=y(x_n)+hf(x_n,y(x_n)) \tag{7}$$

局部截断误差指的是，按（7）式计算由 x_n 到 x_{n+1} 这一步的计算值 y_{n+1} 与精确值 $y(x_{n+1})$ 之差 $y(x_{n+1})-y_{n+1}$。为了估计它，由 Taylor 展开得到的精确值 $y(x_{n+1})$ 是

$$y(x_{n+1})=y(x_n)+hy'(x_n)+\frac{h^2}{2}y''(x_n)+O(h^3) \tag{8}$$

（7）、（8）两式相减（注意到 $y'=f(x,y)$）得

$$y(x_{n+1})-y_{n+1}=\frac{h^2}{2}y''(x_n)+O(h^3)\approx O(h^2) \tag{9}$$

即局部截断误差是 h^2 阶的，而数值算法的精度定义为：

若一种算法的局部截断误差为 $O(h^{p+1})$，则称该算法具有 p **阶精度**。

显然 p 越大，方法的精度越高。式（9）说明，向前 Euler 方法是一阶方法，因此它的精度不高。

§3 改进的 Euler 方法

3.1 梯形公式

利用数值积分方法将微分方程离散化时，若用梯形公式计算式(4)中之右端积分，即

$$\int_{x_n}^{x_{n+1}} f(x,y(x))dx\approx\frac{h}{2}[f(x_n,y(x_n))+f(x_{n+1},y(x_{n+1}))]$$

并用 y_n,y_{n+1} 代替 $y(x_n),y(x_{n+1})$，则得计算公式

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

这就是求解初值问题（1）的梯形公式。

直观上容易看出，用梯形公式计算数值积分要比矩形公式好。梯形公式为二阶方法。

梯形公式也是隐式格式，一般需用迭代法求解，迭代公式为

$$\begin{cases} y_{n+1}^{(0)}=y_n+hf(x_n,y_n) \\ y_{n+1}^{(k+1)}=y_n+\dfrac{h}{2}[f(x_n,y_n)+f(x_{n+1},y_{n+1}^{(k)})], \quad (k=0,1,2,\cdots) \end{cases} \tag{10}$$

由于函数 $f(x,y)$ 关于 y 满足 Lipschitz 条件，容易看出

$$|y_{n+1}^{(k+1)}-y_{n+1}^{(k)}|\le\frac{hL}{2}|y_{n+1}^{(k)}-y_{n+1}^{(k-1)}|$$

其中 L 为 Lipschitz 常数。因此，当 $0<\dfrac{hL}{2}<1$ 时，迭代收敛。但这样做计算量较大。如果实际计算时精度要求不太高，用公式（10）求解时，每步可以只迭代一次，由此导出一种新的方法—改进 Euler 法。

3.2 改进 Euler 法

按式（5）计算问题（1）的数值解时，如果每步只迭代一次，相当于将 Euler 公式与梯形公式结合使用：先用 Euler 公式求 y_{n+1} 的一个初步近似值 $\bar{y}_{n+1}$，称为预测值，然后用梯形公式校正求得近似值 y_{n+1}，即

$$\begin{cases} \bar{y}_{n+1} = y_n + hf(x_n, y_n) & \text{预测} \\ y_{n+1} = y_n + \dfrac{h}{2}[f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})] & \text{校正} \end{cases} \tag{11}$$

式（11）称为由 Euler 公式和梯形公式得到的预测—校正系统，也叫改进 Euler 法。

为便于编制程序上机，式（11）常改写成

$$\begin{cases} y_p = y_n + hf(x_n, y_n) \\ y_q = y_n + hf(x_n + h, y_p) \\ y_{n+1} = \dfrac{1}{2}(y_p + y_q) \end{cases} \tag{12}$$

改进 Euler 法是二阶方法。

§4 龙格—库塔（Runge—Kutta）方法

回到 Euler 方法的基本思想—用差商代替导数—上来。实际上，按照微分中值定理应有

$$\frac{y(x_{n+1}) - y(x_n)}{h} = y'(x_n + \theta h), 0 < \theta < 1$$

注意到方程 $y' = f(x, y)$ 就有

$$y(x_{n+1}) = y(x_n) + hf(x_n + \theta h, y(x_n + \theta h)) \tag{13}$$

不妨记 $\overline{K} = f(x_n + \theta h, y(x_n + \theta h))$，称为区间 $[x_n, x_{n+1}]$ 上的平均斜率。可见给出一种斜率 $\overline{K}$，（13）式就对应地导出一种算法。

向前 Euler 公式简单地取 $f(x_n, y_n)$ 为 $\overline{K}$，精度自然很低。改进的 Euler 公式可理解为 $\overline{K}$ 取 $f(x_n, y_n)$，$f(x_{n+1}, \bar{y}_{n+1})$ 的平均值，其中 $\bar{y}_{n+1} = y_n + hf(x_n, y_n)$，这种处理提高了精度。

如上分析启示我们，在区间 $[x_n, x_{n+1}]$ 内多取几个点，将它们的斜率加权平均作为 $\overline{K}$，就有可能构造出精度更高的计算公式。这就是龙格—库塔方法的基本思想。

4.1 首先不妨在区间 $[x_n, x_{n+1}]$ 内仍取 2 个点，仿照（13）式用以下形式试一下

$$\begin{cases} y_{n+1} = y_n + h(\lambda_1 k_1 + \lambda_2 k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \alpha h, y_n + \beta h k_1), \ 0 < \alpha, \beta < 1 \end{cases} \tag{14}$$

其中 $\lambda_1, \lambda_2, \alpha, \beta$ 为待定系数，看看如何确定它们使（14）式的精度尽量高。为此我们分析局部截断误差 $y(x_{n+1}) - y_{n+1}$，因为 $y_n = y(x_n)$，所以（14）可以化为

$$\begin{cases} y_{n+1} = y(x_n) + h(\lambda_1 k_1 + \lambda_2 k_2) \\ k_1 = f(x_n, y(x_n)) = y'(x_n) \\ k_2 = f(x_n + \alpha h, y(x_n) + \beta h k_1) \\ \quad = f(x_n, y(x_n)) + \alpha h f_x(x_n, y(x_n)) + \beta h k_1 f_y(x_n, y(x_n)) + O(h^2) \end{cases} \tag{15}$$

其中k_2在点$(x_n, y(x_n))$作了 Taylor 展开。（15）式又可表为

$$y_{n+1} = y(x_n) + (\lambda_1 + \lambda_2)hy'(x_n) + \lambda_2\alpha h^2(f_x + \frac{\beta}{\alpha} ff_y) + O(h^3)$$

注意到

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2} y''(x_n) + O(h^3)$$

中$y' = f$，$y'' = f_x + ff_y$，可见为使误差$y(x_{n+1}) - y_{n+1} = O(h^3)$，只须令

$$\lambda_1 + \lambda_2 = 1，\ \lambda_2\alpha = \frac{1}{2}，\ \frac{\beta}{\alpha} = 1 \qquad (16)$$

待定系数满足（16）的（15）式称为 2 阶龙格—库塔公式。由于（16）式有 4 个未知数而只有 3 个方程，所以解不唯一。不难发现，若令$\lambda_1 = \lambda_2 = \frac{1}{2}$，$\alpha = \beta = 1$，即为改进的 Euler 公式。可以证明，在$[x_n, x_{n+1}]$内只取 2 点的龙格—库塔公式精度最高为 2 阶。

4.2　4 阶龙格—库塔公式

要进一步提高精度，必须取更多的点，如取 4 点构造如下形式的公式：

$$\begin{cases} y_{n+1} = y_n + h(\lambda_1 k_1 + \lambda_2 k_2 + \lambda_3 k_3 + \lambda_4 k_4) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \alpha_1 h, y_n + \beta_1 h k_1) \\ k_3 = f(x_n + \alpha_2 h, y_n + \beta_2 h k_1 + \beta_3 h k_2) \\ k_4 = f(x_n + \alpha_3 h, y_n + \beta_4 h k_1 + \beta_5 h k_2 + \beta_6 h k_3) \end{cases} \qquad (17)$$

其中待定系数$\lambda_i, \alpha_j, \beta_m$共 13 个，经过与推导 2 阶龙格—库塔公式类似、但更复杂的计算，得到使局部误差$y(x_{n+1}) - y_{n+1} = O(h^5)$的 11 个方程。取既满足这些方程、又较简单的一组$\lambda_i, \alpha_j, \beta_m$，可得

$$\begin{cases} y_{n+1} = \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \frac{h}{2}, y_n + \frac{hk_1}{2}) \\ k_3 = f(x_n + \frac{h}{2}, y_n + \frac{hk_2}{2}) \\ k_4 = f(x_n + h, y_n + hk_3) \end{cases} \qquad (18)$$

这就是常用的 4 阶龙格—库塔方法（简称 RK 方法）。

§5　线性多步法

以上所介绍的各种数值解法都是单步法，这是因为它们在计算y_{n+1}时，都只用到前一步的值y_n，单步法的一般形式是

$$y_{n+1}=y_n+h\varphi(x_n,y_n,h)\quad(n=0,1,\cdots,N-1)\tag{19}$$

其中$\varphi(x,y,h)$称为增量函数，例如 Euler 方法的增量函数为$f(x,y)$，改进 Euler 法的增量函数为

$$\varphi(x,y,h)=\frac{1}{2}[f(x,y)+f(x+h,y+hf(x,y))]$$

如何通过较多地利用前面的已知信息，如$y_n,y_{n-1},\cdots,y_{n-r}$，来构造高精度的算法计算$y_{n+1}$，这就是多步法的基本思想。经常使用的是线性多步法。

让我们试验一下$r=1$，即利用y_n,y_{n-1}计算y_{n+1}的情况。

从用数值积分方法离散化方程的（4）式

$$y(x_{n+1})-y(x_n)=\int_{x_n}^{x_{n+1}}f(x,y(x))dx$$

出发，记$f(x_n,y_n)=f_n$，$f(x_{n-1},y_{n-1})=f_{n-1}$，式中被积函数$f(x,y(x))$用二节点$(x_{n-1},f_{n-1})$，$(x_n,f_n)$的插值公式得到（因$x\ge x_n$），所以是外插。

$$\begin{aligned}f(x,y(x))&=f_n\frac{x-x_{n-1}}{x_n-x_{n-1}}+f_{n-1}\frac{x-x_n}{x_{n-1}-x_n}\\&=\frac{1}{h}[(x-x_{n-1})f_n-(x-x_n)f_{n-1}]\end{aligned}\tag{20}$$

此式在区间$[x_n,x_{n+1}]$上积分可得

$$\int_{x_n}^{x_{n+1}}f(x,y(x))dx=\frac{3h}{2}f_n-\frac{h}{2}f_{n-1}$$

于是得到

$$y_{n+1}=y_n+\frac{h}{2}(3f_n-f_{n-1})\tag{21}$$

注意到插值公式（20）的误差项含因子$(x-x_{n-1})(x-x_n)$，在区间$[x_n,x_{n+1}]$上积分后得出h^3，故公式（21）的局部截断误差为$O(h^3)$，精度比向前 Euler 公式提高 1 阶。

若取$r=2,3,\cdots$，可以用类似的方法推导公式，如对于$r=3$有

$$y_{n+1}=y_n+\frac{h}{24}(55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3})\tag{22}$$

其局部截断误差为$O(h^5)$。

如果将上面代替被积函数$f(x,y(x))$用的插值公式由外插改为内插，可进一步减小误差。内插法用的是$y_{n+1},y_n,\cdots,y_{n-r+1}$，取$r=1$时得到的是梯形公式，取$r=3$时可得

$$y_{n+1}=y_n+\frac{h}{24}(9f_{n+1}+19f_n-5f_{n-1}+f_{n-2})\tag{23}$$

与（22）式相比，虽然其局部截断误差仍为$O(h^5)$，但因它的各项系数（绝对值）大为减小，误差还是小了。当然，（23）式右端的f_{n+1}未知，需要如同向后 Euler 公式一样，用迭代或校正的办法处理。

§6 一阶微分方程组与高阶微分方程的数值解法

6.1 一阶微分方程组的数值解法

设有一阶微分方程组的初值问题

$$\begin{cases} y'_i = f_i(x, y_1, y_2, \cdots, y_m) \\ y_i(a) = y_{i0} \end{cases} \qquad (i = 1,2,\cdots,m) \tag{24}$$

若记 $y = (y_1, y_2, \cdots, y_m)^T$， $y_0 = (y_{10}, y_{20}, \cdots, y_{m0})^T$， $f = (f_1, f_2, \cdots, f_m)^T$，则初值问题（24）可写成如下向量形式

$$\begin{cases} y' = f(x, y) \\ y(a) = y_0 \end{cases} \tag{25}$$

如果向量函数 $f(x, y)$ 在区域 D：

$$a \le x \le b, \quad y \in R^m$$

连续，且关于 y 满足 Lipschitz 条件，即存在 $L > 0$，使得 $\forall x \in [a, b]$， $y_1, y_2 \in R^m$，都有

$$\| f(x, y_1) - f(x, y_2) \| \le L \| y_1 - y_2 \|$$

那么问题（25）在 $[a, b]$ 上存在唯一解 $y = y(x)$。

问题（25）与（1）形式上完全相同，故对初值问题（1）所建立的各种数值解法可全部用于求解问题（25）。

6.2 高阶微分方程的数值解法

高阶微分方程的初值问题可以通过变量代换化为一阶微分方程组初值问题。

设有 m 阶常微分方程初值问题

$$\begin{cases} y^{(m)} = f(x, y, y', \cdots, y^{(m-1)}) \quad a \le x \le b \\ y(a) = y_0, y'(a) = y_0^{(1)}, \cdots, y^{(m-1)}(a) = y_0^{(m-1)} \end{cases} \tag{26}$$

引入新变量 $y_1 = y, y_2 = y', \cdots, y_m = y^{(m-1)}$，问题（26）就化为一阶微分方程组初值问题

$$\begin{cases} y'_1 = y_2 & y_1(a) = y_0 \\ y'_2 = y_3 & y_2(a) = y_0^{(1)} \\ \quad \vdots & \quad \vdots \\ y'_{m-1} = y_m & y_{m-1}(a) = y_0^{(m-2)} \\ y'_m = f(x, y_1, \cdots, y_m) & y_m(a) = y_0^{(m-1)} \end{cases} \tag{27}$$

然后用 6.1 中的数值方法求解问题（27），就可以得到问题（26）的数值解。

最后需要指出的是，在化学工程及自动控制等领域中，所涉及的常微分方程组初值问题常常是所谓的“刚性”问题。具体地说，对一阶线性微分方程组

$$\frac{dy}{dx} = Ay + \Phi(x) \tag{28}$$

其中 $y, \Phi \in R^m$, A 为 m 阶方阵。若矩阵 A 的特征值 $\lambda_i (i = 1,2,\cdots,m)$ 满足关系

$$\mathrm{Re}\ \lambda_i < 0 \quad (i = 1,2,\cdots,m)$$

$$\max_{1\le i\le m}|\mathrm{Re}\,\lambda_i| >> \min_{1\le i\le m}|\mathrm{Re}\,\lambda_i|$$

则称方程组（28）为刚性方程组或 Stiff 方程组，称数

$$s=\max_{1\le i\le m}|\mathrm{Re}\,\lambda_i|/\min_{1\le i\le m}|\mathrm{Re}\,\lambda_i|$$

为刚性比。对刚性方程组，用前面所介绍的方法求解，都会遇到本质上的困难，这是由数值方法本身的稳定性限制所决定的。理论上的分析表明，求解刚性问题所选用的数值方法最好是对步长 h 不作任何限制。

§7 初值问题的 Matlab 解法和符号解

7.1 Matlab 数值解

7.1.1 非刚性常微分方程的解法

Matlab 的工具箱提供了几个解非刚性常微分方程的功能函数，如 ode45，ode23，ode113，其中 ode45 采用四五阶 RK 方法，是解非刚性常微分方程的首选方法，ode23 采用二三阶 RK 方法，ode113 采用的是多步法，效率一般比 ode45 高。

Matlab 的工具箱中没有 Euler 方法的功能函数。

（I）对简单的一阶方程的初值问题

$$\begin{cases} y'=f(x,y) \\ y(x_0)=y_0 \end{cases}$$

改进的 Euler 公式为

$$\begin{cases} y_p=y_n+hf(x_n,y_n) \\ y_q=y_n+hf(x_n+h,y_p) \\ y_{n+1}=\dfrac{1}{2}(y_p+y_q) \end{cases}$$

我们自己编写改进的 Euler 方法函数 eulerpro.m 如下：

```
function [x,y]=eulerpro(fun,x0,xfinal,y0,n);
if nargin<5,n=50;end
h=(xfinal-x0)/n;
x(1)=x0;y(1)=y0;
for i=1:n
   x(i+1)=x(i)+h;
   y1=y(i)+h*feval(fun,x(i),y(i));
   y2=y(i)+h*feval(fun,x(i+1),y1);
   y(i+1)=(y1+y2)/2;
end
```

例1 用改进的Euler方法求解

$$y'=-2y+2x^2+2x,\quad (0\le x\le 0.5),\quad y(0)=1$$

解 编写函数文件 doty.m 如下：

```
function f=doty(x,y);
f=-2*y+2*x^2+2*x;
```

在Matlab命令窗口输入：

```
[x,y]=eulerpro('doty',0,0.5,1,10)
```

即可求得数值解。

（II）ode23，ode45，ode113的使用

Matlab的函数形式是

[t,y]=solver('F',tspan，y0)

这里solver为ode45，ode23，ode113，输入参数 F 是用M文件定义的微分方程 $y'=f(x,y)$ 右端的函数。tspan=[t0,tfinal]是求解区间，y0是初值。

例2　用RK方法求解

$y'=-2y+2x^2+2x$，$(0\le x\le 0.5)$，$y(0)=1$

解　同样地编写函数文件 doty.m 如下：

```
function f=doty(x,y);
f=-2*y+2*x^2+2*x;
```

在Matlab命令窗口输入：

[x,y]=ode45('doty',[0,0.5],1)

即可求得数值解。

也可以利用Matlab的匿名函数，计算数值解，程序如下：

```
f=@(x,y)-2*y+2*x^2+2*x;
[x,y]=ode45(f,[0,0.5],1)
```

7.1.2　刚性常微分方程的解法

Matlab的工具箱提供了几个解刚性常微分方程的功能函数，如ode15s，ode23s，ode23t，ode23tb，这些函数的使用同上述非刚性微分方程的功能函数。

7.1.3　高阶微分方程 $y^{(n)}=f(t,y,y',\cdots,y^{(n-1)})$ 解法

例 3　考虑初值问题

$$y'''-3y''-y'y=0\quad y(0)=0\quad y'(0)=1\quad y''(0)=-1$$

解　（i）如果设 $y_1=y,y_2=y',y_3=y''$，那么

$$\begin{cases}y_1'=y_2 & y_1(0)=0\\ y'_2=y_3 & y_2(0)=1\\ y'_3=3y_3+y_2y_1 & y_3(0)=-1\end{cases}$$

初值问题可以写成 $Y'=F(t,Y),Y(0)=Y_0$ 的形式，其中 $Y=[y_1\ \ y_2\ \ y_3]^T$。

（ii）把一阶方程组写成接受两个参数 t 和 y，返回一个列向量的 M 文件 F.m：

```
function dy=F(t,y);
dy=[y(2);y(3);3*y(3)+y(2)*y(1)];
```

注意：尽管不一定用到参数 t 和 y，M—文件必须接受此两参数。这里向量 dy 必须是列向量。

（iii）用 Matlab 解决此问题的函数形式为

[T,Y]=solver('F',tspan,y0)

这里 solver 为 ode45、ode23、ode113，输入参数 F 是用 M 文件定义的常微分方程组，tspan=[t0 tfinal]是求解区间，y0 是初值列向量。在 Matlab 命令窗口输入

[T,Y]=ode45('F',[0 1],[0;1;-1])

就得到上述常微分方程的数值解。这里 Y 和时刻 T 是一一对应的，Y(:,1)是初值问题的解，Y(:,2)是解的导数，Y(:,3)是解的二阶导数。

也可以利用 Matlab 的匿名函数，求数值解，编写如下程序：

```
F=@(t,y)[y(2);y(3);3*y(3)+y(2)*y(1)];
[T,Y]=ode45(F,[0 1],[0;1;-1])
```

例 4　求 van der Pol 方程

$$y''-\mu(1-y^2)y'+y=0$$

的数值解，这里 $\mu>0$ 是一参数。

解　（i）化成常微分方程组。设 $y_1=y, y_2=y'$，则有

$$\begin{cases} y'_1=y_2 \\ y'_2=\mu(1-y_1^2)y_2-y_1 \end{cases}$$

（ii）书写 M 文件（对于 $\mu=1$）vdp1.m:

```
function dy=vdp1(t,y);
dy=[y(2);(1-y(1)^2)*y(2)-y(1)];
```

（iii）调用 Matlab 函数。对于初值 $y(0)=2, y'(0)=0$，解为

```
[T,Y]=ode45('vdp1',[0 20],[2;0]);
```

（iv）观察结果。利用图形输出解的结果：

```
plot(T,Y(:,1),'-',T,Y(:,2),'--')
title('Solution of van der Pol Equation,mu=1');
xlabel('time t');
ylabel('solution y');
legend('y1','y2');
```

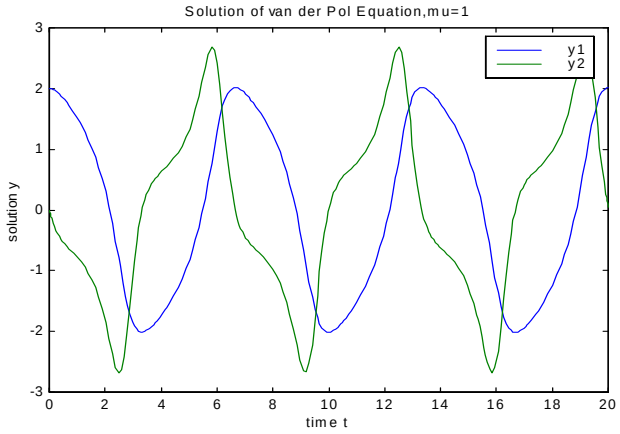

图 1

例 5　van der Pol 方程，$\mu=1000$（刚性）

解　(i)书写 M 文件 vdp1000.m:

```
function dy=vdp1000(t,y);
dy=[y(2);1000*(1-y(1)^2)*y(2)-y(1)];
```

（ii）观察结果

```
[t,y]=ode15s('vdp1000',[0 3000],[2;0]);
plot(t,y(:,1),'o')
title('Solution of van der Pol Equation,mu=1000');
xlabel('time t');
ylabel('solution y(:,1)');
```

7.2　常微分方程的解析解

在 Matlab 中，符号运算工具箱提供了功能强大的求解常微分方程的符号运算命令 dsolve。常微分方程在 Matlab 中按如下规定重新表达：

符号 D 表示对变量的求导。Dy 表示对变量 y 求一阶导数，当需要求变量的 n 阶导数时，用 Dn 表示，D4y 表示对变量 y 求 4 阶导数。

由此，常微分方程 $y''+2y'=y$ 在 Matlab 中，将写成'D2y+2*Dy=y'。

7.2.1 求解常微分方程的通解

无初边值条件的常微分方程的解就是该方程的通解。其使用格式为：

dsolve('diff_equation')

dsolve(' diff_equation'，'var')

式中 diff_equation 为待解的常微分方程，第 1 种格式将以变量 t 为自变量进行求解，第 2 种格式则需定义自变量 var。

例 6　试解常微分方程

$$x^2+y+(x-2y)y'=0$$

解　编写程序如下：

```
syms x y
diff_equ='x^2+y+(x-2*y)*Dy=0';
dsolve(diff_equ,'x')
```

7.2.2 求解常微分方程的初边值问题

求解带有初边值条件的常微分方程的使用格式为：

dsolve('diff_equation'，'condition1，condition2，…'，'var')

其中 condition1，condition2，… 即为微分方程的初边值条件。

例 7　试求微分方程

$$y'''-y''=x,\quad y(1)=8,y'(1)=7,y''(2)=4$$

的解。

解　编写程序如下：

```
y=dsolve('D3y-D2y=x','y(1)=8,Dy(1)=7,D2y(2)=4','x')
```

7.2.3 求解常微分方程组

求解常微分方程组的命令格式为：

dsolve('diff_equ1，diff_equ2，…'，'var')

dsolve('diff_equ1，diff_equ2，…'，'condition1，condition2，…'，'var')

第 1 种格式用于求解方程组的通解，第 2 种格式可以加上初边值条件，用于具体求解。

例 8　试求常微分方程组：

$$\begin{cases} f''+3g=\sin x \\ g'+f'=\cos x \end{cases}$$

的通解和在初边值条件为 $f'(2)=0,f(3)=3,g(5)=1$ 的解。

解　编写程序如下：

```
clc,clear
equ1='D2f+3*g=sin(x)';
equ2='Dg+Df=cos(x)';
[general_f,general_g]=dsolve(equ1,equ2,'x')
[f,g]=dsolve(equ1,equ2,'Df(2)=0,f(3)=3,g(5)=1','x')
```

7.2.4 求解线性常微分方程组

（i）一阶齐次线性微分方程组

$$X'=AX,\quad X=\begin{bmatrix}x_1\\ \vdots\\ x_n\end{bmatrix},\quad A=\begin{bmatrix}a_{11} & \cdots & a_{1n}\\ \vdots & \vdots & \vdots\\ a_{n1} & \cdots & a_{nn}\end{bmatrix}$$

这里的’表示对t求导数。e^{At}是它的基解矩阵。$X'=AX$，$X(t_0)=X_0$的解为$X(t)=e^{A(t-t_0)}X_0$。

例 9　试解初值问题

$$X'=\begin{bmatrix}2 & 1 & 3\\ 0 & 2 & -1\\ 0 & 0 & 2\end{bmatrix}X,\quad X(0)=\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}$$

解　编写程序如下：

```
syms t
a=[2,1,3;0,2,-1;0,0,2];
x0=[1;2;1];
x=expm(a*t)*x0
```

（ii）非齐次线性方程组

由参数变易法可求得初值问题

$$X'=AX+f(t),\quad X(t_0)=X_0$$

的解为

$$X(t)=e^{A(t-t_0)}X_0+\int_{t_0}^{t}e^{A(t-s)}f(s)ds\,.$$

例 10　试解初值问题

$$X'=\begin{bmatrix}1 & 0 & 0\\ 2 & 1 & -2\\ 3 & 2 & 1\end{bmatrix}X+\begin{bmatrix}0\\ 0\\ e^t\cos 2t\end{bmatrix},\quad X(0)=\begin{bmatrix}0\\ 1\\ 1\end{bmatrix}。$$

解　编写程序如下：

```
clc,clear
syms t s
a=[1,0,0;2,1,-2;3,2,1];ft=[0;0;exp(t)*cos(2*t)];
x0=[0;1;1];
x=expm(a*t)*x0+int(expm(a*(t-s))*subs(ft,s),s,0,t);
x=simple(x)
```

§8 边值问题的Matlab解法

Matlab中用bvp4c命令求解常微分方程的两点边值问题，微分方程的标准形式为

$$y'=f(x,y),\quad bc(y(a),y(b))=0$$

或者是

$$y'=f(x,y,p),\quad bc(y(a),y(b),p)=0$$

其中p是有关的参数。这里y,f可以为向量函数，求解的区间为$[a,b]$，bc为边界条件。

一般地说，边值问题在计算上比初值问题困难得多，特别地，由于边值问题的解可能是多值的，bvp4c需要提供猜测的初始值。下面我们首先给出一个简单的例子。

例11　考察描述在水平面上一个小水滴横截面形状的标量方程

$$\frac{d^2}{dx^2}h(x)+(1-h(x))(1+(\frac{d}{dx}h(x))^2)^{3/2}=0,\quad h(-1)=h(1)=0$$

这里 $h(x)$ 表示 x 处水滴的高度。设 $y_1(x)=h(x)$，$y_2(x)=\frac{dh(x)}{dx}$，把上述微分方程写成两个一阶微分方程组

$$\frac{d}{dx}y_1(x)=y_2(x)$$

$$\frac{d}{dx}y_2(x)=(y_1(x)-1)(1+y_2(x)^2)^{3/2}$$

上述微分方程组可以由如下函数表示

```
function yprime=drop(x,y);
yprime=[y(2);(y(1)-1)*(1+y(2)^2)^(3/2)];
```

边界条件通过残差函数指定，边界条件通过如下函数表示

```
function res=dropbc(ya,yb);
res=[ya(1);yb(1)];
```

我们使用 $y_1(x)=\sqrt{1-x^2}$ 和 $y_2(x)=-x/(0.1+\sqrt{1-x^2})$ 作为初始猜测解，由如下函数定义

```
function yinit=dropinit(x);
yinit=[sqrt(1-x.^2);-x./(0.1+sqrt(1-x.^2))];
```

利用如下的程序就可以求微分方程的边值问题并画出图2。

```
solinit=bvpinit(linspace(-1,1,20),@dropinit);
sol=bvp4c(@drop,@dropbc,solinit);
fill(sol.x,sol.y(1,:),[0.7,0.7,0.7])
axis([-1,1,0,1])
xlabel('x','FontSize',12)
ylabel('h','Rotation',0,'FontSize',12)
```

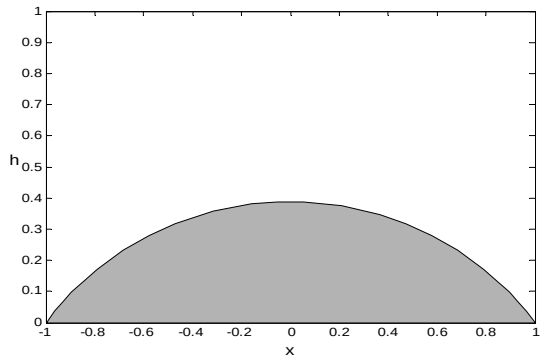

图2

这里调用函数bvpinit，计算区间$[-1,1]$上等间距的20个点的数据，然后调用函数bvp4c，得到数值解的结构sol，填充命令fill填充 $x-y_1$ 平面上的解曲线。

一般地，bvp4c的调用格式如下：

```
sol=bvp4c(@odefun,@bcfun,solinit,options,p1,p2,…);
```
函数odefun的格式为
```
yprime=odefun(x,y,p1,p2, …);
```
函数bcfun的格式为
```
res=bcfun(ya,yb, p1,p2, …);
```
初始猜测结构solinit有两个域：solinit.x提供初始猜测的 x 值，排列顺序从左到右排列，其中solinit.x(1)和solinit.x(end)分别为 a 和 b 。对应地，solinit.y(:,i)给出点solinit.x(i)处初始猜测解。

输出参数sol是包含数值解的一个结构，其中sol.x给出了计算数值解的 x 点，sol.x(i)处的数值解由sol.y(:,i)给出，类似地，sol.x(i)处数值解的一阶导数值由sol.yp(:,i)给出。

我们可以把上面的所有函数都放在一个文件中，程序如下：
```
function sol=example11;
solinit=bvpinit(linspace(-1,1,20),@dropinit);
sol=bvp4c(@drop,@dropbc,solinit);
fill(sol.x,sol.y(1,:),[0.7,0.7,0.7])
axis([-1,1,0,1])
xlabel('x','FontSize',12)
ylabel('h','Rotation',0,'FontSize',12)

function yprime=drop(x,y);
yprime=[y(2);(y(1)-1)*(1+y(2)^2)^(3/2)];

function res=dropbc(ya,yb);
res=[ya(1);yb(1)];

function yinit=dropinit(x);
yinit=[sqrt(1-x.^2);-x./(0.1+sqrt(1-x.^2))];
```

例12　描述 $x=0$ 处固定， $x=1$ 处有弹性支持，沿着 x 轴平衡位置以均匀角速度旋转的绳的位移方程

$$\frac{d^2}{dx^2}y(x)+\mu y(x)=0$$

具有边界条件

$$y(0)=0,\ \left(\frac{d}{dx}y(x)\right)\Big|_{x=0}=1,\ \left(y(x)+\frac{d}{dx}y(x)\right)\Big|_{x=1}=0$$

这个边值问题是一个特征问题，我们必须找到参数 μ 的值使得方程的解存在。如果我们提供了参数 μ 的猜测值和对应解的猜测值，我们也可以利用函数bvp4c求解特征问题。上述微分方程可以写成下面的微分方程组：

$$\frac{d}{dx}y_1(x)=y_2(x)$$

$$\frac{d}{dx}y_2(x)=-\mu y_1(x)$$

编写程序如下（下面的所有程序放在一个文件中）：
```
function sol=skiprun;
```

```
solinit=bvpinit(linspace(0,1,10),@skipinit,5);
sol=bvp4c(@skip,@skipbc,solinit);
plot(sol.x,sol.y(1,:),'-',sol.x,sol.yp(1,:),'--','LineWidth',2)
xlabel('x','FontSize',12)
legend('y_1','y_2',0)

function yprime=skip(x,y,mu);
yprime=[y(2);-mu*y(1)];

function res=skipbc(ya,yb,mu);
res=[ya(1);ya(2)-1;yb(1)+yb(2)];

function yinit=skipinit(x);
yinit=[sin(x);cos(x)];
```

图3给出上述边值问题解的图象。

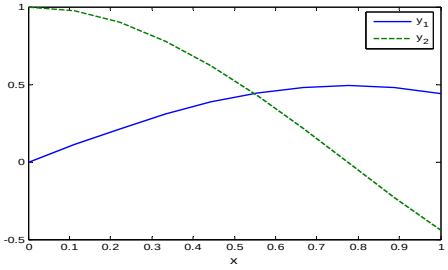

图3

例13　微分方程组为

$$u' = 0.5u(w-u)/v$$
$$v' = -0.5(w-u)$$
$$w' = (0.9 - 1000(w-y) - 0.5w(w-u))/z$$
$$z' = 0.5(w-u)$$
$$y' = -100(y-w)$$

边界条件为 $u(0)=v(0)=w(0)=1$， $z(0)=-10$； $w(1)=y(1)$。

我们使用如下猜测解

$$u(x)=1$$
$$v(x)=1$$
$$w(x)=-4.5x^2+8.91x+1$$
$$z(x)=10$$
$$y(x)=-4.5x^2+9x+0.91$$

我们编写如下程序（所有程序放在一个文件中）：

```
function sol=example13
solinit=bvpinit(linspace(0,1,5),@exinit);
sol=bvp4c(@exode,@exbc,solinit);
plot(sol.x,sol.y)
```

```
function yprime=exode(x,y)
yprime=[0.5*y(1)*(y(3)-y(1))/y(2)
        -0.5*(y(3)-y(1))
        (0.9-1000*(y(3)-y(5))-0.5*y(3)*(y(3)-y(1)))/y(4)
        0.5*(y(3)-y(1))
        100*(y(3)-y(5))];

function res=exbc(ya,yb)
res=[ya(1)-1;ya(2)-1;ya(3)-1;ya(4)+10;yb(3)-yb(5)];

function yinit=exinit(x)
yinit=[1;1;-4.5*x^2+8.91*x+1;-10;-4.5*x^2+9*x+0.91];
```

习 题 十 五

1. 用欧拉方法和龙格—库塔方法求微分方程数值解，画出解的图形，对结果进行分析比较。

（i） $x^2y''+xy'+(x^2-n^2)y=0, y\left(\frac{\pi}{2}\right)=2, y'\left(\frac{\pi}{2}\right)=-\frac{2}{\pi}$（Bessel 方程，令 $n=\frac{1}{2}$），精确解 $y=\sin x\sqrt{\frac{2\pi}{x}}$。

（ii） $y''+y\cos x=0, y(0)=1, y'(0)=0$，幂级数解

$$y=1-\frac{1}{2!}x^2+\frac{2}{4!}x^4-\frac{9}{6!}x^6+\frac{55}{8!}x^8-\cdots$$

2. 一只小船渡过宽为 d 的河流，目标是起点 A 正对着的另一岸 B 点。已知河水流速 v_1 与船在静水中的速度 v_2 之比为 k。

（i）建立小船航线的方程，求其解析解。

（ii）设 $d=100\,\mathrm{m}$，$v_1=1\,\mathrm{m/s}$，$v_2=2\,\mathrm{m/s}$，用数值解法求渡河所需时间、任意时刻小船的位置及航行曲线，作图，并与解析解比较。

3. Lorenz 方程是一个三阶的非线性系统，它是由描述大气动力系统的 Navier-Stokes偏微分方程演化而来的。自由系统如下：

$$\begin{cases}\dot{x}=\sigma(y-x)\\ \dot{y}=\beta x-y-xz\\ \dot{z}=-\lambda z+xy\end{cases}$$

当系统参数 σ,β,λ 在一定范围内，系统就出现混沌，如 $\sigma=10$，$\beta=28$，$\lambda=8/3$ 时，出现混沌现象。求在初始条件 $[x(0),y(0),z(0)]=[5,13,17]$ 时，方程组的数值解，并画出解的图形。