数字计算机在自动控制系统中的仿真

中国矿业学院机电系自动化教研室

一九八〇年八月

前　言

本教材共分上、下两册，上册介绍了模拟计算机在自动控制系统中的仿真，是由孙庚山讲师编写的。本册主要介绍数字计算机在自动控制系统中的仿真，内容取捨于清华大学自动化系编写的“自动控制系统的计算机模拟”一书及其它有关材料，在编写过程中清华大学的熊光楞讲师和李芳芸讲师等给于了大力帮助，本院谢桂林付教授给于很多具体指导，但由于编者水平有限，时间又很仓促，肯定会有不少缺点和错误，希望读者给于批评指正。

编者　1980.7

目　录

第一章　连续系统的数字仿真

§1　数值积分法

一个自动控制系统的动态特性通常是用高阶微分方程式来描述的，研究它的性质，归根结底是要解微分方程，采用数字计算机要仿真自动控制系统的数字基础就是用数字机来解高阶微分方程式。

一、欧拉法（折线法）：用数字来解高阶微分方程式的计算方法统称为数值积分法，数值积分法的基本思想可以用以下原理来说明，如有一个微分方程式

$$\frac{dy}{dt}=f(t,y) \tag{1}$$

已知初始条件 $y(t_0)=y_0$，要求满足上述方程的解 $y(t)$ 通过点 (t_0, y_0)，这点的斜率为 $f(t_0, y_0)$，如果 t_1 十分靠近 t_0（即 Δt 足够小），那么在 t_0 附近，曲线 $y(t)$ 可以近似地用切线来表示，已知切线方程为：

$$y=y_0+f(t_0, y_0)(t_1-t_0) \cdots\cdots (2)$$

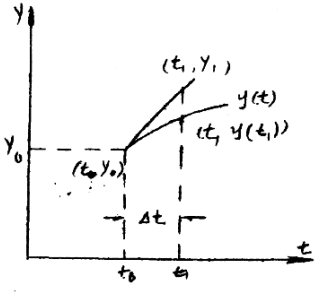

图　1

利用这个方程可以获得 $t=t_1$ 时 $y(t_1)$ 的近似值 y_1，

$$y_1=y_0+f(t_0, y_0)(t_1-t_0)\approx y(t_1) \tag{3}$$

重复上述方法，当 $t=t_2$ 时，

$$y(t_2)\cong y_2=y_1+f(t_1,\ y_1)(t_2-t_1) \tag{4}$$

而 $$y(t_{n+2})\approx y_{n+1}=y_n+f(t_n,y_n)(t_{n+1}-t_n) \tag{5}$$

令 $t_{n+1}-t_n=h_n$，称为计算步距。

以上所说的计算方法虽然十分简单，但计算精度比较低，为了提高精度，唯一的办法是减少步距，但是跟着又会出现新的问题；计算机字长有限，计算中免不了产生舍入误差，步距小，不仅计算工作量大了，而且由于计算次数加多，舍入误差也加大，计算精度很难提高，所以，这种方法实际上很少采用。这种方法最早是由欧拉提出来的，常称为欧拉法，又因它是用折线来近似实际的曲线，故也称为折线法。

二、梯形法

欧拉法比较粗糙，却给我们一些启示，我们知道解 $y(t)=f(t,y)$ 这个微分方程可改成求函数 $y(t)$，它适合：

$$y(t)=y_0+\int_{t_0}^{t}f[t,y(t)]\,dt \tag{6}$$

当 $t=t_1$ 时

$$y(t_1)=y_0+\int_{t_0}^{t_1}f[t,\ y(t)]dt$$

由于上式右边的积分中含有未知函数，$f[t,y(t)]$，所以由它不能得到 $y(t_1)$ 值，但是由矩形公式计算这个定积分，即可得

$$\int_{t_0}^{t_1}f[t,\ y(t)]dt\approx f[t_0,\ y(t_0)](t_1-t_0)$$

那么，$y(t_1)$ 的表示式右边就成为

$$y_0+f[t_0,y(t_0)](t_1-t_0)=y_0+h_0f(t_0,y_0) \tag{7}$$

比较式(3)和式(7)可知，这个式子就是用欧拉计算的 y_1，这样，欧拉法可以看成是用矩形公式近似计算某个相应定积分而得到的公式。

以上用的是隐函数形式，如果我们将微分方程式写成显函数形式，

则可以更清楚一些，比如解 $\frac{dy}{dt}=f(t)$ 这个常系数微分方程，则

$$y=y_0+\int_{t_0}^{t}f(t)dt$$

右边式中的积分就是图 2 中 a，t_0，t，b 所包围的这一块面积。为求出 y(t)，可以将 f(t) 近似看作是由一段段水平线联成的阶梯，则

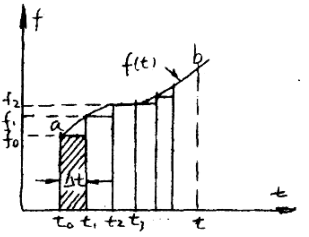

图 2

$$y_1=y_0+f_0\Delta t$$

$$y_2=y_1+f_1\Delta t$$

$$\cdots\cdots\cdots\cdots$$

$$y_{n+1}=y_n+f_n\Delta t$$

Δt 为计算步距，也可写成 h

用矩形公式来计算定积分误差是十分显然的，由此也可知欧拉法是相当粗糙的。为了提高计算精度，很自然地就会想到用梯形来代替矩形，即在求定积分时使用下式：

$$\int_{t_0}^{t}f(t)dt\approx\frac{1}{2}(f_0+f_1)\Delta t$$

则 $y_1=y_0+\frac{1}{2}h_0(f_0+f_1)$

一般式子为：

$$y_{n+1}=y_0+\frac{1}{2}h(f_n+f_{n+1}) \tag{8}$$

若方程为 $y(t)=f(t,y)$

则

$$y_{n+1}=y_n+\frac{1}{2}h\{f(t_n,y_n)+f(t_{n+1},y_{n+1})\} \tag{9}$$

用梯形法来计算，产生了一个新问题，即在计算 y_{n+1} 时，y_{n+1}

尚不知道，那么 $f(t_{n+1}, y_{n+1})$ 也就未知，因而如何来求 y_{n+1} 呢？通常是用迭代法来求，即取

$$y_{n+1}^{(0)} = y_n + hf(t_n, y_n)$$

$$y_{n+1}^{(1)} = y_n + \frac{1}{2}h[f(t_n, y_n) + f(t_{n+1}, y_{n+1}^{(0)})]$$

$$\vdots$$

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

如果 $y_{n+1}^{(0)}$，$y_{n+1}^{(1)}$ ……这个序列是收敛的，那么就有极限存在，即 $K \to \infty$ 时，这个序列趋于某一极限值，因而可以用此极限值来作为 y_{n+1}。当然，迭代次数越多越好，但计算工作量将加大，还有舍入误差，所以常常只迭代一次就认为已经求得 y_{n+1}，这样一来，公式便成为：

$$y_{n+1} = y_n + \frac{1}{2}h[f(t_n, y_n) + f(t_{n+1}, y_{n+1}^{(0)})] \tag{10}$$

$$y_{n+1}^{(0)} = y_n + hf(t_n, y_n) \tag{11}$$

通常这类公式叫做预报校正公式，其中第二个公式叫做预报公式，由它预报 y_{n+1} 的一个值，第一个公式叫做校正公式，由它得出 y_{n+1} 的校正值，由此可见，用这种方法来计算，每求一个 y_n，计算量比欧拉法多一倍。

三、龙格—— 库塔法

由上述梯形法作进一步推广即可得经常便用的龙格—— 库塔法，这个方法的基本思想仍为积分。如有下面常微分方程：

$$\frac{dy}{dt} = f(t, y) \tag{12}$$

$$y(t = t_0) = y_0 \tag{13}$$

假设我们从 t_0 跨出一步，$t_1 = t_0 + h$，t_1 时刻的解为 $y_1 =$

$y(t_0+h)$，可以在 t_0 附近展成泰劳级数，只保留 h^2 项，则

$$y_1=y_0+f(t_0,y_0)h+\frac{1}{2}\left(\frac{\partial f}{\partial t}+f\frac{\partial f}{\partial y}\right)\Delta t^2\Big|_{\substack{t=t_0\\y=y_0}} \tag{14}$$

我们假设上式可写成另一形式

$$y_1=y_0+a_1K_1\Delta t+a_2f(t_0+b_1\Delta t,\ y_0+b_2\Delta tK_1)\Delta t$$

这里 $K_1=f(t_0,y_0)$，并可把 $K_2=f(t_0+b_1\Delta t,y_0+b_2\Delta tK_1)$

…… (15)

因 $f(t_0+b_1\Delta t_1,\ y_0+b_2\Delta tK_1)=f(t_0,y_0)+\left.\frac{\partial f}{\partial t}\right|_0 b_1\Delta t+$

$$+\left.\frac{\partial f}{\partial y}\right|_0 b_2\Delta tK_1$$

则 $y_1=y_0+a_1\Delta tf(t_0,y_0)+a_2\Delta t\left[f(t_0,y_0)+b_1\frac{\partial f}{\partial t}\Delta t\right.$

$$\left.+b_2K_1\left.\frac{\partial f}{\partial y}\right|_0\Delta t\right)$$

这样可得：

$$\begin{aligned}
&y_1=y_0+(a_1K_1+a_2K_2)h\\
&K_1=f(t_0,y_0)\\
&K_2=f(t_0+b_1h,\ y_0+b_2K_1h)\\
&\quad\approx f(t_0,y_0)+b_1\left.\frac{\partial t}{\partial t}\right|_0 h+b_2K_1\left.\frac{\partial f}{\partial y}\right|_{\substack{h\\y=y_0}}\\
&h=\Delta t
\end{aligned} \tag{16}$$

将(15)式与(14)式比较一下可得

$$\begin{aligned}
&a_1+a_2=1\\
&a_2b_1=\frac{1}{2}\\
&a_2b_2=\frac{1}{2}
\end{aligned} \tag{17}$$

今有四个未知数，但只有三个方程，尚可自由选择一个方程，若取

$c_1 = b_2$，于是 $a_1 = b_2 = \frac{1}{2}$（若取 b_1，b_2 为其它比例关系也可）。

代入(17)式得　　$b_1 = b_2 = 1$

于是得到这样一组计算公式

$$\begin{cases} y_1 = y_0 + \dfrac{h}{2}(K_1 + K_2) \\ K_1 = f(t_0, y_0) \\ K_2 = f(t_0 + h, y_0 + K_1 h) \end{cases} \qquad (18)$$

由于(14)式只取了 h 和 h^2 项，而将 h^3 以上的高阶项略去了，所以这种计算方法的截断误差正比于 h^3，又由于计算时只取 h 及 h^2 项，故这种方法被称为二阶龙格—库塔法，将(18)式与(10)式(11)式对比一下，即可发现，这两组计算公式完全相同。由此可以证明，梯形法只迭代一次是可行的，其截断误差正比于 h^3。

二阶龙格 —— 库塔法仍有如欧拉法所存在的问题，即为提高精度，减少截断误差，就要求选较小的步距 h，这样将增加计算次数，因而会增大舍入误差，所以精度要求较高的场合也不宜采用，而应采用高阶龙格—— 库塔法。

下面给出实际计算中常用的四阶龙格—— 库塔法，它的截断误差正比于 h^5，关于它的推导过程，基本上与二阶龙格—— 库塔法相同，这里就不推了。四阶龙格—— 库塔法的计算公式如下：

$$\begin{cases} y_1 = y_0 + \dfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 = f(t_0, y_0) \\ K_2 = f\left(t_0 + \dfrac{h}{2}, y_0 + \dfrac{h}{2}K_1\right) \\ K_3 = f\left(t_0 + \dfrac{h}{2}, y_0 + \dfrac{h}{2}K_2\right) \\ K_4 = f(t_0 + h, y_0 + hK_3) \end{cases} \qquad (19)$$

再说一下，若龙格—— 库塔法只取h这一项，而将 h^2 以上的高阶项全略去，那么由(14)式可知，

$$y_1 = y_0 + f(t_0, y_0)h \qquad (20)$$

这就是欧拉公式，由此可知，它的截断误差正比于 h^2，显然在上述几种方法中欧拉法的精度是最差的。通过龙格—— 库塔法的介绍，现在我们可以将以上几种数值积分法都统一起来了，它们都是龙格—— 库塔法，只不过阶次不同而已。

以上三种方法的共同特点是：

① 在计算 y_{n+1} 时，只用到 y_n，而不直接用 y_{n-1}，y_{n-2} 等等，换句话说，在后一步的计算中，仅仅利用前一步的计算结果，而不利用更前面各步的结果，所以称为一步法。

② 步长 h 在整个计算中并不要求固定，可以根据精度需要变化。

与一步法相对应的还有多步法，但计算步骤比较复杂，在数字仿真中用得较少，就不再介绍了。

§ 2 根据微分方程（或传递函数）求闭环系统过渡过程的数字仿真程序

一、问题的提出：一个连续控制系统可以用微分方程（或状态方程）描述，但最后要落实到仿真程序上去。为了使工作方便，经常采用仿真通用程序。通用程序一般都由程序块组成，在数字仿真系统中，通常包括以下四方面的程序块：

① 初始化程序块：主要是把所有的变量（数组）初始化，取零或初始值。

② 输入程序块：主要把系统的参数（如微分方程的阶数，系数，计算用估计值等等）输入到计算机中去。

③ 运行程序块：这是仿真运算最核心的部分，仿真运算由这类

程序块来完成。

④ 输出程序块：主要把仿真计算的结果用各种不同的方式反映出来，如电传打字机等所用的程序块。

这些程序块可用积木形式搭成，根据仿真系统的需要可增可减，使用很方便，但不能一一都讲，重点讲述一下运行程序块。

二、运行程序块：在一般仿真运算中，给出各种运得程序块，由用户自己选定，现就根据四阶龙格—— 库塔法讲述一下运行程序块。如有一个控制系统，其开环传递函数为 W(S)

$$W(S)=\frac{Y(s)}{u(s)}=\frac{cos^{n-1}+c_1 s^{n-2}+\cdots+c_{n-1}}{s^n+a_1 s^{n-1}+a_2 s^{n-2}+\cdots+a_n} \quad \cdots\cdots (21)$$

如果引入反馈，如图 3 所示，则构成一个闭环的控制系统。已知 W(s)，要求这个闭环系统在某一输入函数 R(s) 的作用下求出其过渡过程

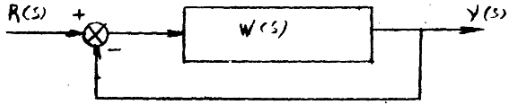

图 3

y(t)。一般是首先将开环传递函数 W(s) 转化为开环状态方程和输出方程。

$$\begin{cases} \dot{X}=AX+Bu \\ y=CX \end{cases} \quad (22)$$

已知 A、B、C 和初始条件 $u_1(0)=R(0^+)-y(0^+)$，就可以利用状态方程求出各个状态变量的 X(1)及 y(1)，然后重复这个步骤，求出 X(2)，y(2)…，直到计算次数达到预先规定的要求为止。如用四阶龙格——库塔法，式(22)可写成

$$\begin{cases} \dot{X}_1 = a_{11}X_1 + a_{12}X_2 + \cdots\cdots a_{1n}X_n + b_1 u = f_1(X_1, X_2 \cdots X_n, u_1 t) \\ \dot{X}_2 = a_{21}X_1 + a_{22}X_2 + \cdots\cdots a_{2n}X_n + b_2 u = f_2(X_1, X_2 \cdots X_n, u_2, t) \\ \vdots \\ \dot{X}_n = a_{n1}X_1 + a_{n2}X_2 + \cdots\cdots a_{nn}X_n + b_n u = f_n(X_1, X_2, X_3 \cdots X_n, u_n, t) \end{cases} \quad \cdots\cdots\cdots\cdots \quad (23)$$

每个方程中有K_1，K_2，K_3，K_4四个值，n个方程中有4Xn个系数值，当$\Delta t = h$，$h_1 = 0$，$h_2 = h_3 = \frac{h}{2}$，$h_4 = h$时，

$$\begin{cases} K_{11} = f_1[t_0 + h_1, X_1^0 + h_1 K_{10} \cdots\cdots u(t_0 + h_1)] \\ K_{21} = f_2[t_0 + h_1, X_1^0 + h_1 K_{10} \cdots\cdots u(t_0 + h_1)] \\ \vdots \\ K_{n1} = f_n[t_0 + h_1, X_1^0 + h_1 K_{10} \cdots\cdots u(t_0 + h_1)] \end{cases} \quad (24)$$

$$\begin{cases} K_{12} = f_1[t_0 + h_2, X_1^0 + h_2 K_{11}, X_2^0 + h_2 K_{21} \cdots\cdots u(t_0 + h_2)] \\ K_{22} = f_2[t_0 + h_2, X_1^0 + h_2 K_{11}, X_2^0 + h_2 K_{21} \cdots\cdots u(t_0 + h_2)] \\ \vdots \\ K_{n2} = f_n[t_0 + h_2, X_1^0 + h_2 K_{11}, X_2^0 + h_2 K_{21} \cdots\cdots u(t_0 + h_2)] \end{cases} \quad (25)$$

$$\begin{cases} K_{13} = f_1[t_0 + h_3, X_1^0 + h_3 K_{12}, X_2^0 + h_3 K_{22} \cdots\cdots u(t_0 + h_3)] \\ K_{23} = f_2[t_0 + h_3, X_1^0 + h_3 K_{12}, X_2^0 + h_3 K_{22} \cdots\cdots u(t_0 + h_3)] \\ \vdots \\ K_{n3} = f_n[t_0 + h_3, X_1^0 + h_3 K_{12}, X_2^0 + h_3 K_{22} \cdots\cdots u(t_0 + h_3)] \end{cases} \quad (26)$$

$$\begin{cases} K_{14} = f_1[t_0 + h_4, X_1^0 + h_4 K_{13}, X_2^0 + h_4 K_{23} \cdots\cdots u(t_0 + h_4)] \\ K_{24} = f_1[t_0 + h_4, X_1^0 + h_4 K_{13}, X_2^0 + h_4 K_{23} \cdots\cdots u(t_0 + h_4)] \\ \vdots \\ K_{n4} = f_1[t_0 + h_4, X_1^0 + h_4 K_{13}, X_2^0 + h_4 K_{23} \cdots\cdots u(t_0 + h_4)] \end{cases} \quad (27)$$

为了紧凑些，以上四组系数的公式可以合并成一个：

$$K_{ij}=f_i\left(t_0+h_j,\overset{\circ}{X}_1+h_jK_{ij-1},\ \overset{\circ}{X}_2+h_jK_{2j-1}\cdots\cdots\overset{\circ}{X}_n+h_jK_{nj-1},\right.$$
$$\left.u(t_0+h_j)\right] \qquad (28)$$

$i=1,\ 2,\ 3\cdots n,\qquad j=1,\ 2,\ 3,\ 4$

可见，用此方法主要是计算 $K_1\ K_2\ K_3\ K_4$ 这四个系数，并且要对所有各状态方程先计算第一个系数 K_1，然后再计算第二个系数 K_2，等等以此类推来求 K_3，K_4，这样就可求出各状态方程的 X_i 值，即

$$X_i=\overset{\circ}{X}_i+\frac{h}{6}(K_{i1}+2K_{i2}+2K_{i3}+K_{i4}) \qquad (29)$$

$i=1,\ 2,\ 3\cdots n$

对于输入为阶跃函数的情况，公式（28）中的 $u(t_0+h_j)$ 始终为 u_0。

三、下面给出一个用 BASIC 程序编制的具体程序，称仿真程序 I_p。

有一系统如图 4 所示，其中 u_0，u，Y，V 是标量（单输入，单输出系统）

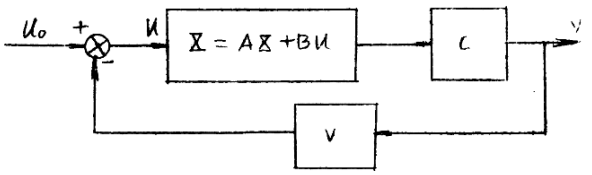

图 4

① 将开环系统的状态方程及输出方程改为闭环系统的状态方程及输出方程，根据图 4。

$$u=u_0-VY$$

$$\therefore\quad \dot{X}=AX+Bu=AX+B(u_0-VY)$$
$$=AX+B(u_0-VCX)$$

$$= (A-BVC)\,X + Bu_0 \qquad (30)$$

$$= FX + Bu_0$$

其中 $F = A - BVC$ 也是一个 $n \times n$ 的方阵，它是闭环系统的状态方程的系数阵，这样，图(4)可改画成图 5 。

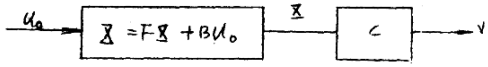

图 5

因此，在程序中一定要有通过 A 、 B 、 C 、 V 求 F 的内容。

② 程序框图