智能控制一迭代学习控制

朱芳来

第一章 绪论

1.1 迭代学习控制技术及其发展概况

- 从根本上来看，控制器的设计问题可以归为两大类：调节问题和跟踪问题，而调节问题也可以看成是跟踪问题的特殊情况。
- 迭代学习控制的一个主要任务就是，实现被控系统的输出零误差地跟踪期望轨迹。
- 研究具有学习能力的控制器，以达到对期望轨迹的跟踪，一直是控制工程师们探索的问题。

- 迭代学习控制的概念最早（1978年）由日本学者Uchiyama在一篇有关机器人控制的论文提出，但因为文章是以日文发表的，所以但是并没有引起人们的注意。
- 1984年，另一个日本人Arimoto及其合作者们将Uchiyama的思想加以完善，建立了实用的算法，并提出了更为正规的迭代学习理论，并以英文发表了其研究成果，从而使迭代学习控制成为引人注目的课题。

- 迭代学习控制是智能系统中具有严格数学描述的一个分支，特别适用于具有重复性质的控制对象，它的目标是实现有限区间上的完全跟踪任务。

1.2 迭代学习控制的研究内容及其基本原理

迭代学习控制的主要研究内容包括：学习算法的稳定性与收敛性、学习速度、学习律及对学习系统结构的研究、学习过程的鲁棒性等。

迭代学习控制的基本原理

设被控对象的动态过程为

$$\begin{cases} \dot{x} = f(t, x, u) \\ y = g(t, x, u) \end{cases} \tag{1.1}$$

其中$x \in \mathbf{R}^n$, $y \in \mathbf{R}^m$, $u \in \mathbf{R}^r$ 而f,g为相应维数的向量函数。要求系统在时间$t \in [0,T]$内的输出y(t)跟踪期望输出$y_d(t)$。假定期望输出$u_d(t)$存在，即：在给定的初始状态x(0)下$u_d(t)$是式(1.1)当$y(t)=y_d(t)$的解存在，则迭代学习控制的目标就是通过多次重复的运行，在一定的学习律下$u(t) \to u_d(t)$, $y(t) \to y_d(t)$.

在第k次运行时(1.1)表示为

$$\begin{cases} \dot{x}_k = f(t, x_k, u_k) \\ y_k = g(t, x_k, u_k) \end{cases} \quad (1.2)$$

输出误差为

$$e_k(t)=y_d(t)-y_k(t) \quad (1.3)$$

其中下表表示第k次迭代。

迭代学习控制分为开环学习和闭环学习。开环学习控制的方法是：第k+1次的控制等于第k次控制再加上第k次输出误差的PID校正项，即

$$u_{k+1}(t) = u_k(t) + \Gamma_p e_k(t) + \Gamma_i \int_0^t e_k(s)ds + \Gamma_d \frac{de_k(t)}{dt} \qquad (1.4)$$

其中 $\Gamma_p, \Gamma_i, \Gamma_d$ 分别为PID学习增益矩阵。更一般的开环学习律可以写成

$$u_{k+1}(t)=L(u_k(t),e_k(t)) \qquad (1.5)$$

这里L是线性或非线性算子。

开环PID学习控制的基本结构如图1.1所示

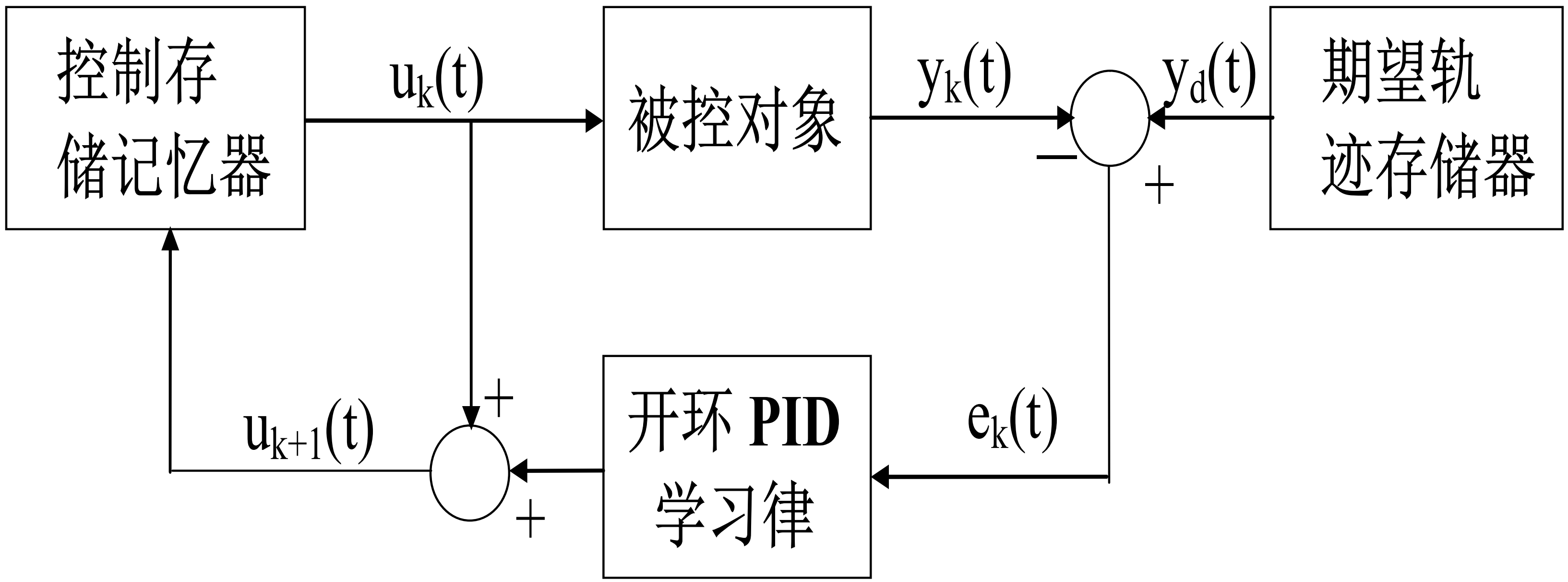

图1.1 开环PID学习控制的基本结构

闭环PID学习控制策略是取第k+1次运行的误差作为学习的修正项，即

$$u_{k+1}(t) = u_k(t) + \Gamma_p e_{k+1}(t) + \Gamma_i \int_0^t e_{k+1}(s)ds + \Gamma_d \frac{de_{k+1}(t)}{dt} \qquad (1.6)$$

更一般的闭环学习律

$$u_{k+1}(t)=L(u_k(t),e_{k+1}(t)) \qquad (1.7)$$

其中L是线性或非线性算子。

闭环PID学习控制的基本结构如图1.2所示

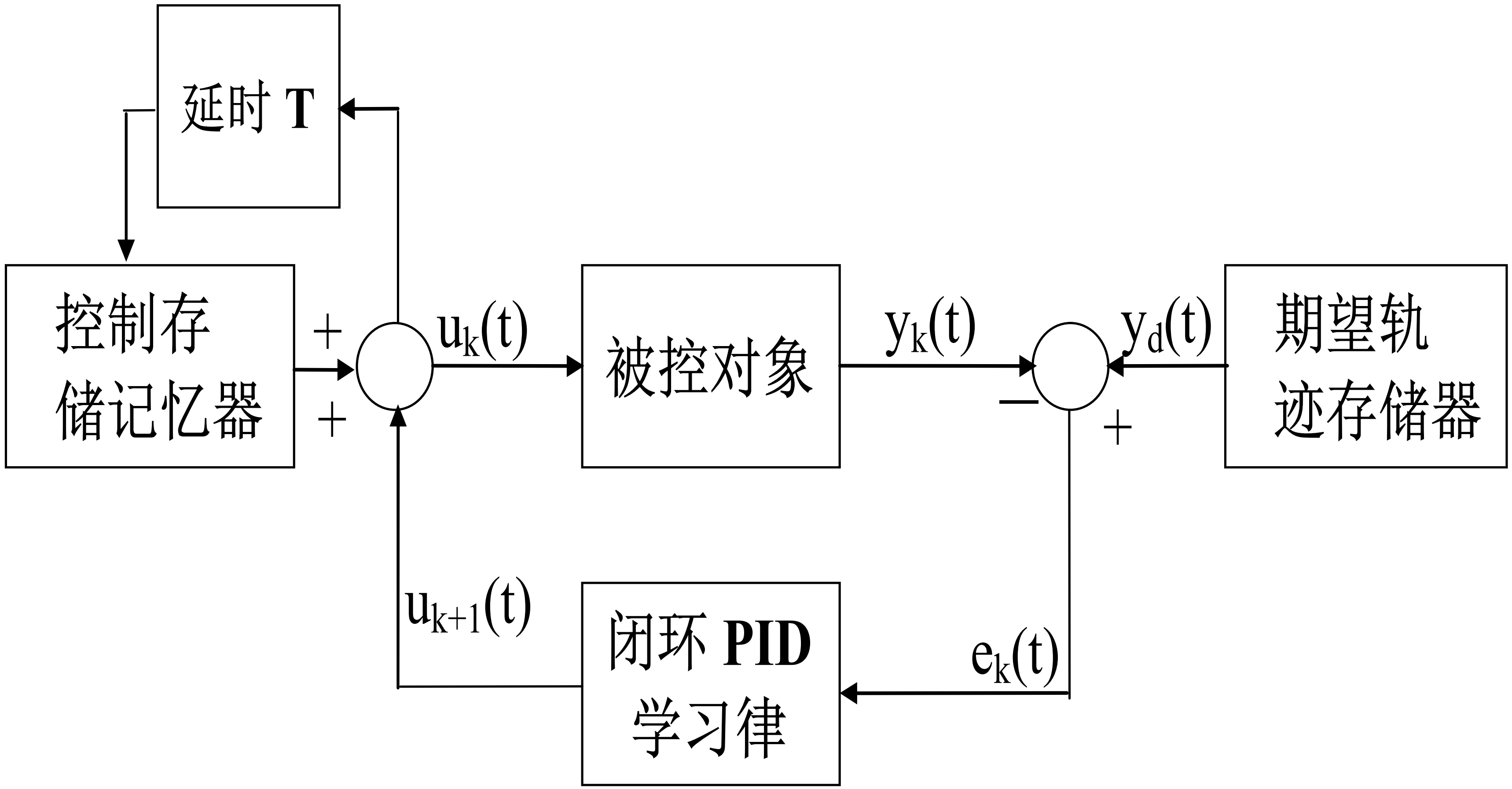

图1.2 闭环PID迭代学习控制基本结构

第二章 数学基础

2.1 一些基本概念

定义2.1：设X为一个集合，若在该集合上定义两种运算，一个称为加法，记为“＋”；一个称为数量乘法，记为“·”。加法和数量乘法的结果皆为X中的元素，且它们满足：

1) $\forall x, y \in X,\ x+y=y+x$

2) $\forall x, y, z \in X,\ (x+y)+z=x+(y+x)$

3) 存在X中一个元素，称为零元，记为0，使得

使得 $\forall x \in X$，有

$$x+0=0+x=x$$

4) $\forall x \in X$, 存在X中的一个元素，称为x的负元，记为-x，使得

$$x+(-x)=0$$

5) 对任意的两个数a, b, $\forall x \in X$, 有

$$(a \cdot b) \cdot x = a \cdot (b \cdot x)$$

6) 对任意的两个数a,b，$\forall x \in X$，有

$$(a + b) \cdot x = a \cdot x + b \cdot x$$

7) 对任意的数a，$\forall x, y \in X$ ，有

$$a \cdot (x + y) = a \cdot x + a \cdot y$$

8) $\forall x \in X$ ，$1 \cdot x = x$

则称X为线性空间，或向量空间。

例2.1 设C[a,b]表示区间[a,b]上所有连续函数构成的集合，若按通常函数的加法及数与函数的乘法，则C[a,b]构成了一个线性空间。同样，对n-维实数向量全体所组成的集合$\mathbf{R}^n$,按通常向量的加法及数与向量的乘法， 构成了一个线性空间。

定义2.2 设X是一个线性空间，$\|\cdot\|$ 为从X到[0,+∞)的一个映射，且满足：

1) $\forall a \in \mathbf{R}, \forall x \in X$，$\|a \cdot x\| = |a| \cdot \|x\|$

2) $\forall x, y \in X, \|x+y\| \leq \|x\| + \|y\|$

3) $\|x\| = 0 \Leftrightarrow x = 0$

则成映射 $\|\cdot\|$ 为线性空间X上的一个范数，配有范数的线性空间称为赋范空间。

例2.2 对连续函数空间C[a,b]，$\forall f \in C[a,b]$，定义

$$\|f\| = \max_{a \leq t \leq b} |f(t)|$$

则容易验证，上式定义的确实是一个C[a,b]上的范数。

例2.3 对向量空间$\mathbf{R}^n$，$\forall x=\begin{bmatrix}x_1 & x_2 & \cdots & x_n\end{bmatrix}^T\in\mathbf{R}^n$

可以定义如下的一些范数：

1) p范数(Holder范数)：

$$\|x\|_p=\left(\sum_{i=1}^{n}|x_i|^p\right)^{\frac{1}{p}}$$

2) ∞-范数： $\|x\|_\infty=\max\limits_{1\le i\le n}|x_i|$

3) 1-范数： $\|x\|_1=\sum\limits_{i=1}^{n}|x_i|$

4) 2-范数(Euclid)：

$$\|x\|_2=\sqrt{\sum_{I=1}^{n}|x_i|^2}=\sqrt{x^Tx}$$

定义2.3 设X是赋范空间，$\{x_n\}_{n\geq 1}$ 为X中的一个序列，x为X中的一个元素，若 $\lim_{n\to\infty}\|x_n - x\| = 0$ 则称序列$\{x_n\}_{n\geq 1}$在X中按范数收敛到x，简称$\{x_n\}_{n\geq 1}$收敛到x, 记为 $\lim_{n\to\infty} x_n = x$ 或 $x_n \to x$ 当 $n \to \infty$

定义2.4 设X, Y是赋范空间，从X到Y的一个映射S称为从X到Y的算子，记为S:X→Y.

例如，函数

$$f(x) = \begin{bmatrix} x_1 x_2^2 \\ x_1 + x_2^3 \\ x_2 \end{bmatrix}$$

其中 $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbf{R}^2$

可以看成从线性空间$\mathbf{R}^2$到线性空间$\mathbf{R}^3$的一个映射，因而，f定义了从线性空间$\mathbf{R}^2$到线性空间$\mathbf{R}^3$的一个算子。

第三章 迭代学习控制的主要结论

3.1 开环P型学习

设被控系统的动态过程为

$$\begin{cases} \dot{x}(t) = f(t, x(t), u(t)) \\ y(t) = g(t, x(t)) + D(t)u(t) \end{cases} \quad (3.1)$$

其中 $x \in \mathbf{R}^n, y \in \mathbf{R}^m, u \in \mathbf{R}^r$ 分别为系统的状态，可测输出和控制输入。要求在时间$t \in [t_0, T]$内系统的输出精确地跟踪期望输出$y_d(t)$。

在第k次运行时，迭代学习的动态方程是：

$$\begin{cases} \dot{x}_k(t) = f(t, x_k(t), u_k(t)) \\ y_k(t) = g(t, x_k(t)) + D(t)u_k(t) \end{cases} \tag{3.2}$$

输出误差为

$$e_k(t)=y_d(t)-y_k(t) \tag{3.3}$$

开环P型学习律为

$$u_{k+1}(t) = u_k(t) + \Gamma(t)e_k(t) \tag{3.4}$$

其中 $\Gamma(t)$ 为学习增益。令

$$\begin{cases} \delta x_k(t) = x_d(t) - x_k(t) \\ \delta y_k(t) = y_d(t) - y_k(t) \\ \delta u_k(t) = u_d(t) - u_k(t) \end{cases} \tag{3.5}$$

其中x_d, y_d, u_d分别为期望轨迹上的状态，输出和控制。

如下的定理给出了迭代学习控制算法的收敛性。

定理**3.1**：设被控系统的动态方程如(3.1)所示，且在$t \in [0, T]$中满足下面的条件：

(A1) $\forall t, u, x_1, x_2$ 有

$$\|f(t,x_1,u)-f(t,x_2,u)\| \le F(t,u)\cdot\|x_1-x_2\|$$

(A2) $\forall t, x, u_1, u_2$ 有

$$\|f(t,x,u_1)-f(t,x,u_2)\| \le M\cdot\|u_1-u_2\|$$

(A3) $\forall t, x_1, x_2$ 有

$$\|g(t,x_1)-g(t,x_2)\| \le M\|x_1-x_2\|$$

(A4) 每次运行时的初始误差$\{\delta x_k(0)\}_{k\ge 0}$为一收敛到零的序列。

(A5) 存在唯一的控制$u_d(t)$ 使得系统的状态和系统的期望输出值分别为$x_d(t)$和$y_d(t)$

则迭代学习律(3.4)收敛的充分必要条件是谱半径

$$\rho[I-\Gamma(t)D(t)]<1,\ t\in[0,T]$$

3.2 开环D型学习

设被控系统的动态方程为

$$\begin{cases} \dot{x}(t) = f(t, x(t)) + B(t)u(t) \\ y(t) = C(t)x(t) \end{cases}$$

其中f的结构与参数B, C均为未知。

要求在t∈[0,T]内系统的输出y(t)能精确地跟踪期望输出$y_d(t)$，在k次运行时，系统的动态方程为

$$\begin{cases}\dot{x}_k(t) = f(t, x_k(t)) + B(t)u_k(t) \\ y_k(t) = C(t)x_k(t)\end{cases}$$

输出误差为

$$e_k(t)=y_d(t)-y_k(t)$$

开环D型学习律为

$$u_{k+1}(t) = u_k(t) + \Gamma(t)\dot{e}_k(t)$$

3.3 闭环P型学习和闭环D型学习

考虑被控系统(3.1)，则闭环P型和闭环D型学习律分别为

$$u_{k+1}(t) = u_k(t) + \Gamma(t) e_{k+1}(t)$$

$$u_{k+1}(t) = u_k(t) + \Gamma(t) \dot{e}_{k+1}(t)$$

例：设被控系统为

$$\begin{cases}\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix}=\begin{bmatrix}0 & 1\\ -(2+5t) & -(3+2t)\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}+\begin{bmatrix}0\\ 1\end{bmatrix}u(t)\\ y(t)=\begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}\end{cases}$$

要求在[0, 1]时间内跟踪期望输出$y_d(t)=12t^2(1-t)$，用开环D型学习算法，给出算法。

开环D型学习律为

$$u_{k+1}(t)=u_k(t)+\Gamma\dot{e}_k(t) \tag{3.5}$$

其中 $e_k(t)=y_d(t)-y_k(t)$ (3.6)

原系统展开为

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -(2+5t)x_1 - (3+2t)x_2 + u \end{cases} \tag{3.7}$$

$$y = x_2$$

于是，期望输出等价为

$$y_d = x_{2d} = 12t^2 - 12t^3$$

代入(3.7)得到

$$\dot{x}_{1d} = x_{2d} = 12t^2 - 12t^3$$

既有

$$x_{1d} = 4t^3 - 3t^4 + c$$

于是，期望状态为

$$x_d = \begin{bmatrix} x_{1d} \\ x_{2d} \end{bmatrix} = \begin{bmatrix} 4t^3 - 3t^4 + c \\ 12t^2 - 12t^3 \end{bmatrix}$$

如果取初始状态为

$$x_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

则期望状态具体为

$$x_d = \begin{bmatrix} x_{1d} \\ x_{2d} \end{bmatrix} = \begin{bmatrix} 4t^3 - 3t^4 \\ 12t^2 - 12t^3 \end{bmatrix}$$

由(3.6)知，

$$\begin{aligned} \dot{e}_k &= \dot{y}_d - \dot{y}_k = 24t - 36t^2 - \dot{x}_{2k} \\ &= 24t - 36t^2 + (2+5t)x_{1k} + (3+2t)x_{2k} + u_k \end{aligned}$$

将上式代入(3.5)，并整理得

$$u_{k+1} = (1-\Gamma)u_k + \Gamma(24t - 36t^2 + \begin{bmatrix} 2+5t & 3+2t \end{bmatrix} \begin{bmatrix} x_{1k} \\ x_{2k} \end{bmatrix})$$

由此整理出算法，令

$$A(t)=\begin{bmatrix}0 & 1\\ -(2+5t) & -(3+2t)\end{bmatrix}, B=\begin{bmatrix}0\\ 1\end{bmatrix}, C=\begin{bmatrix}0 & 1\end{bmatrix}$$

则第k步状态方程为

$$\dot{x}_k(t)=A(t)x_k(t)+Bu_k(t)$$

开始步(第0步)，初始状态和初始控制分别为

$$x_0(0)=\begin{bmatrix}1\\ 0\end{bmatrix},\ \ u_0(t)=0$$

第k(k≥1)步，初始状态取为

$$x_k(0)=\begin{bmatrix}1+\dfrac{1}{k}\\ \dfrac{1}{k}\end{bmatrix}$$

而迭代学习控制由()给出，学习算法收敛的充分性条件是 $0<\Gamma<2$ ，如可以取 $\Gamma=0.7$

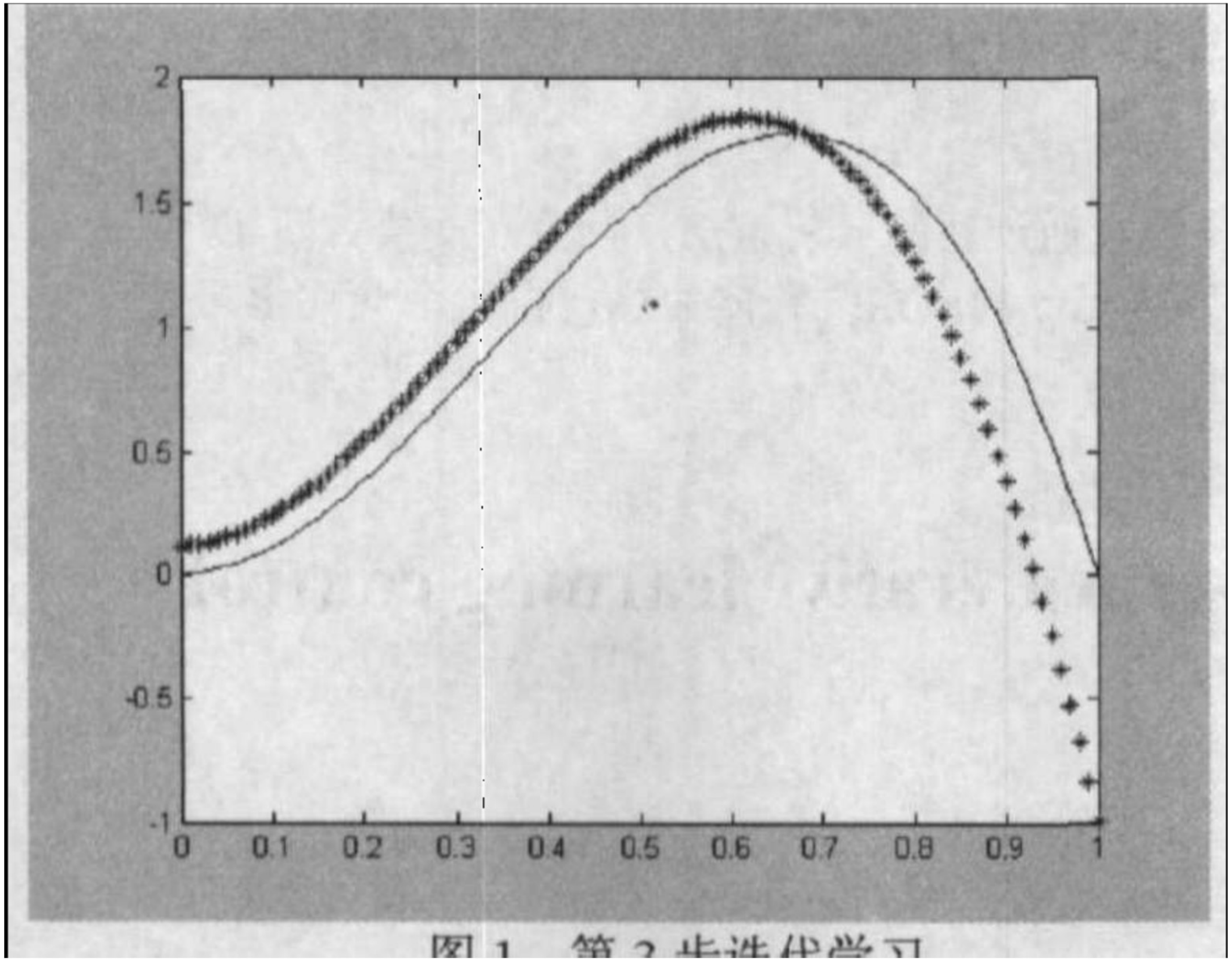

图1 第3步迭代学习

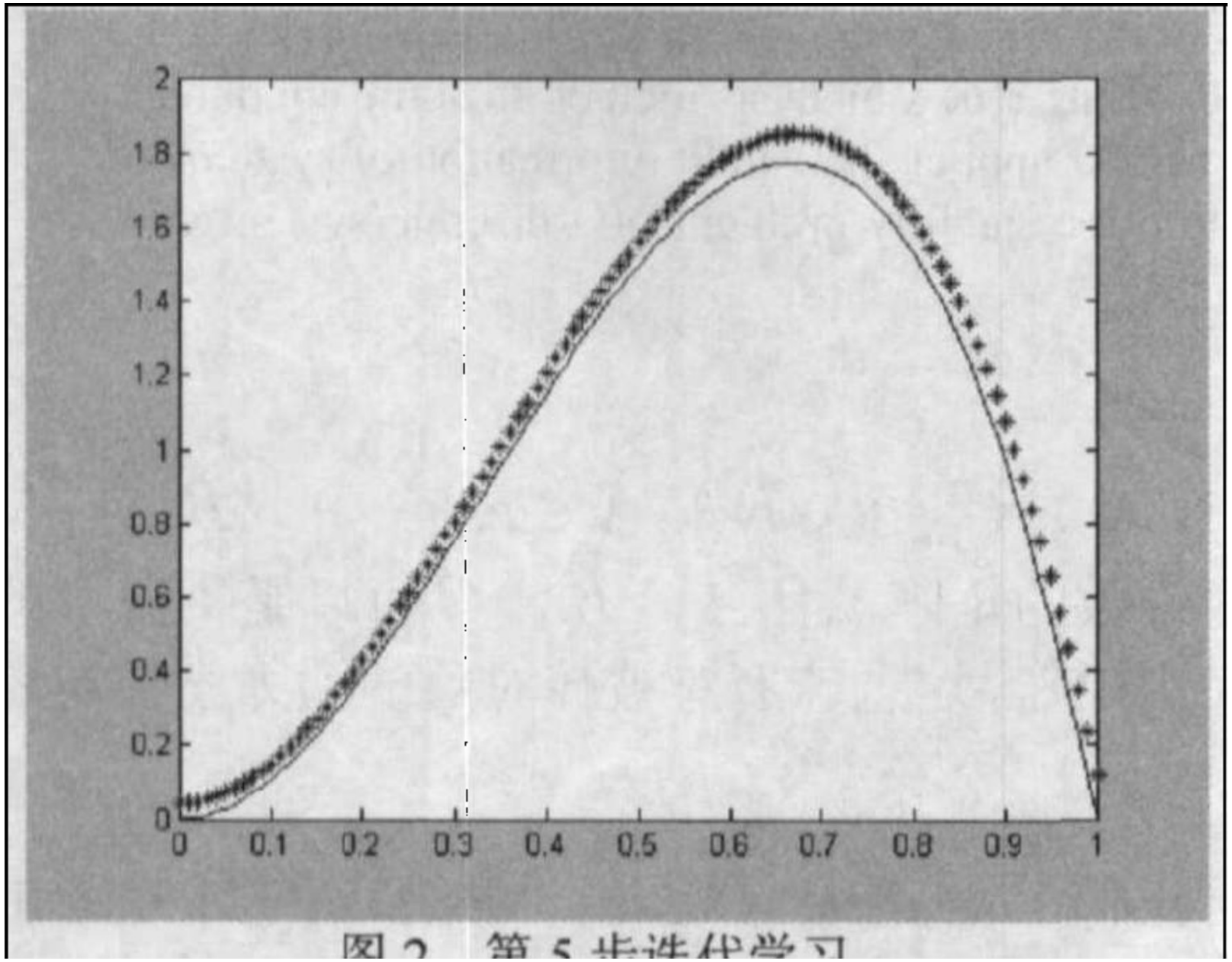

图2 第5步迭代学习

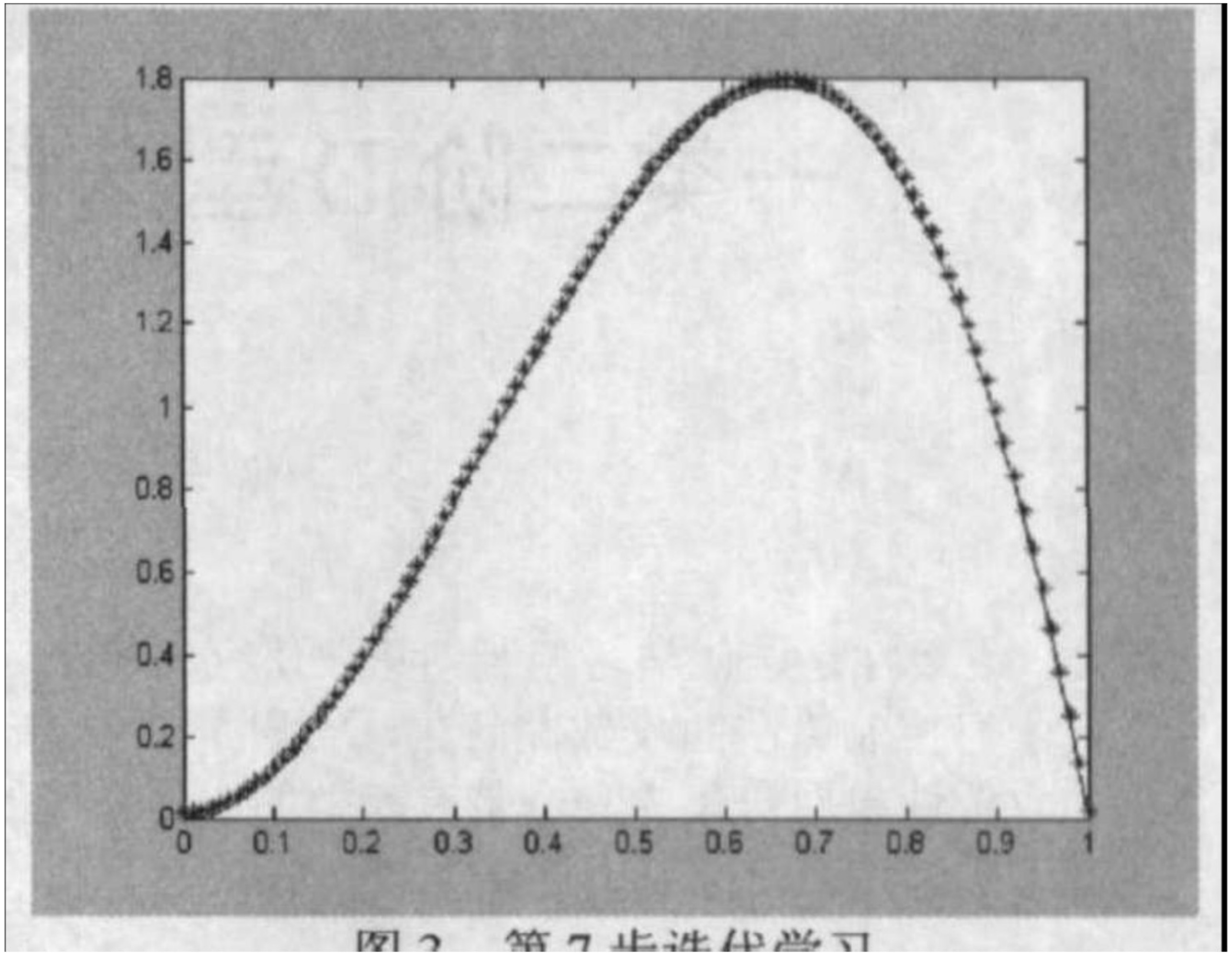

图3 第7步迭代学习

3.4 微分方程数值解仿真方法

考虑微分方程

$$\dot{x}(t) = f(t, x(t)) \qquad (3.8)$$

其中 $x(t) \in \mathbf{R}^n, t \in \mathbf{R}^+$

- 微分方程的解析解：对微分方程(3.8)，给定在初始时刻t_0的初始条件$x(t_0)=x_0$，如果它的解存在且可以用一个明确的数学表达式表示：

$$x(t) = \phi(t; t_0, x_0) \qquad (3.9)$$

其中(3.9)满足 $\phi(t_0; t_0, x_0) = x_0$ 且 $\dot{\phi}(t; t_0, x_0) = f(t, \phi(t; t_0, x_0))$

则称(3.9)是微分方程(3.8)的解析解。

- 微分方程的数值解：给定解的时间区间[a,b]，对时间区间插入若干个点$t_0,t_1,\ldots,t_n$,既有

$$a=t_0<t_1<\ldots<t_n=b$$

如果能够得到在这些时间点上微分方程(3.8)的解的近似值$x_1,x_2,\ldots,x_n$，则如下的表格

t_0	t_1	…	t_n
x_1	x_2	…	x_n

称为微分方程(3.8)的数值解。

- 微分方程数值解方法：龙格－库达方法

```
%%%%%%%%%%%%%%%%%%%%%%%%%
function [t,x]=myode45(f,a,b,h,x0)
t=a:h:b;
n1=size(t);
n=n1(2);
x(:,1)=x0;
for i=1:n
    tt=a+(i-1)*h;
    K1=feval(f,tt,x(:,i));
    K2=feval(f,tt+h/2,x(:,i)+(h/2)*K1);
    K3=feval(f,tt+h/2,x(:,i)+(h/2)*K2);
    K4=feval(f,tt+h,x(:,i)+h*K3);
    x(:,i+1)=x(:,i)+(h/6)*(K1+2*K2+2*K3+K4);
end
x=x(:,1:n);
%%%%%%%%%%%%%%%%%%%%%%%%%%
```