第5章 控制系统的数学建模与仿真

控制系统的数学模型关系到对系统各方面性能的仿真分析，建立控制系统的数学模型是分析和设计控制系统的首要工作。本章简要介绍时域建模方法、频域建模方法和神经网络建模方法，并通过仿真示例重点介绍仿真方法的使用。

数学模型有动态模型与静态模型之分。描述系统动态过程的方程式，如微分方程、偏微分方程、差分方程等，称为动态模型；在静态条件下，即变量的各阶导数为零，描述系统各变量之间关系的方程式称为静态模型。

同一个物理系统，可以用不同的数学模型来表达。例如，实际的物理系统一般含有非线性特性，因此系统的数学模型就应该是非线性的。严格地讲，实际系统的参数不可能是集中的，所以系统的数学模型又应该用偏微分方程描述。但是非线性方程或偏微分方程对应模型的参数辨识相当困难，有时甚至不可能。因此，为了便于问题的求解，常常在误差允许的范围内，忽略次要因素，用简化的数学模型来表示实际的物理系统。这样，同一个系统就有完整、复杂的数学模型和简单、准确性较差的数学模型之分。一般情况下，在建立数学模型时，必须在模型的简化性与分析结果的精确性之间做出折中选择。

此外，数学模型的形式有多种。为了便于分析研究，可能某种形式的数学模型比另一种更合适。例如，在求解最优控制问题或多变量系统的问题时，采取状态变量表达式（即状态空间表达式）比较方便；但是在对单输入、单输出系统的分析中，采用输入输出间的传递函数（或脉冲传递函数）作为系统的数学模型比较合适。

所以在建立系统数学模型时，必须：

（1）全面了解系统的特性，确定研究目的以及准确性要求，确定是否需要忽略一些次要因素而使系统数学模型简化，既不致造成数学处理上的困难，又不致影响分析的准确性。一般在条件允许时，最初尽可能采用简化的常系数线性数学模型，然后再在线性模型分析的基础上考虑被忽略因素所引起的误差，最后建立起系统比较完善准确的数学模型。但是

必须指出，由于数学分析方法上的误差，复杂的数学模型不一定能带来预期的准确结果。

（2）根据所应用的系统分析方法，建立相应形式的数学模型（微分方程、传递函数等），有时还要考虑计算机仿真求解的要求。

建立系统的数学模型主要有两条途径。第一条途径是利用人们已有的关于系统的知识，采用机理的方法建立数学模型。机理法是一种推理方法，用这种方法建立模型时，是通过系统本身机理（物理、化学规律）的分析确定模型的结构和参数，从理论上推导出系统的数学模型。这种利用机理法得出的数学模型称为机理模型或解析模型。第二条途径是根据对系统的观察，通过测量所得到的大量输入、输出数据，推断出被研究系统的数学模型。这种方法称为实验法，利用实验法建立的数学模型称为经验模型。一般来讲，采用实验法建立的数学模型，是系统模型化问题的唯一解。而采用机理法时，能够满足观测到的输入、输出数据关系的系统模型却有很多个。

视频讲解

5.1 时域建模方法及示例

本书介绍的时域建模方法，主要是采用机理法建模。连续系统的时域模型主要是指微分方程描述系统的模型，离散系统的时域模型主要是指差分方程描述系统的模型，以及相应的状态空间模型。

1. 微分方程建模

控制系统的运动状态可由微分方程式描述，而微分方程式就是系统的一种数学模型。关于建立系统微分方程式的一般步骤如下：

（1）在允许的条件下适当简化，忽略一些次要因素。

（2）根据物理或化学定律，列出元件的原始方程式。这里所说的物理或化学定律，是指牛顿第二定律、能量守恒定律、物质不灭定律、克希霍夫定律等。

（3）列出原始方程式中中间变量与其他因素的关系式。这种关系式可能是数学方程式，或是曲线图。它们在大多数场合是非线性的。若条件允许，则应进行线性化处理，否则按非线性系统对待。

（4）将上述关系式代入原始方程式，消去中间变量，就得元件的输入、输出关系的方程式。若在步骤（3）不能进行线性化，则输入输出关系方程式将是比较复杂的非线性方程式。

（5）同理，求出其他元件的方程式。

（6）从所有元件的方程式中消去中间变量，最后的系统输入输出微分方程式。

【例 5-1】 如图 5-1 所示的一个弹簧-质量-阻尼器系统，当外力 $f(t)$ 作用时，系统产生位移 $y(t)$，要求写出系统在外力 $f(t)$ 作用下的运动方程式。在此，$f(t)$ 是系统的输入，$y(t)$ 是系统的输出。

建模的过程如下：

（1）设运动部件质量用 m 表示，按集中参数处理，列出原始方程式。根据牛顿第二定

律，有

$$f(t)-f_1(t)-f_2(t)=m\frac{\mathrm{d}^2y(t)}{\mathrm{d}t^2} \tag{5-1}$$

式中，$f_1(t)$——阻尼器阻力；$f_2(t)$——弹簧弹力。

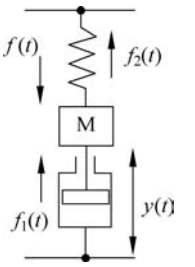

图 5-1　弹簧-质量-阻尼器系统

(2) $f_1(t)$和$f_2(t)$为中间变量，找出它们与其他因素的关系。由于阻尼器是一种产生黏性摩擦或阻尼的装置，活塞杆和缸体发生相对运动时，其阻力与运动方向相反，与运动速度成正比，故有

$$f_1(t)=B\frac{\mathrm{d}y(t)}{\mathrm{d}t} \tag{5-2}$$

式中，B——阻尼系数。

设弹簧为线性弹簧，则有

$$f_2(t)=Ky(t) \tag{5-3}$$

式中，K——弹簧的弹性系数。

(3) 将式(5-2)和式(5-3)代入式(5-1)，经整理后就得描述该弹簧-质量-阻尼器系统的微分方程式为

$$m\frac{\mathrm{d}^2y(t)}{\mathrm{d}t^2}+B\frac{\mathrm{d}y(t)}{\mathrm{d}t}+ky(t)=f(t) \tag{5-4}$$

式中，m、B、K 均为常数，故上式为线性定常二阶微分方程，则此机械位移系统为线性定常系统。

一般列写微分方程式时，输出量及其各阶导数项列写在方程式左端，输入项列写在右端。由于一般物理系统均有质量、惯性或储能元件等物理可实现的原因，左端的导数阶次总比右端的高。在本例中，有质量 M，又有吸收能量的阻尼器 B，系统有两个时间常数，故左端导数项最高阶次为 2。

【例 5-2】 建立图 5-2 所示的 RLC 串联电路输入、输出电压之间的微分方程。

建模过程如下：

(1) 确定系统的输入、输出量，输入端电压 $u_i(t)$为输入量，输出端电压 $u_o(t)$为输出量。

(2) 列写微分方程，设回路电流为 $i(t)$，由基尔霍夫定律可得

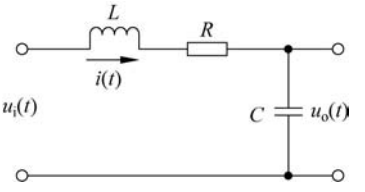

图 5-2　RLC 电路图

$$u_R+u_L+u_C=u_i \tag{5-5}$$

式中，$u_R(t)$、$u_L(t)$、$u_C(t)$分别为 R、L、C 上的电压降。

又由

$$u_R(t)=Ri(t),\quad u_L(t)=L\frac{\mathrm{d}i(t)}{\mathrm{d}t}$$

可得

$$L\frac{\mathrm{d}i(t)}{\mathrm{d}t}+Ri(t)+u_C(t)=u_i(t) \tag{5-6}$$

(3) 消去中间变量，得出系统的微分方程。考虑 $u_C(t)=u_o(t)$，根据电容的特性可得

$$i(t)=C\frac{\mathrm{d}u_C(t)}{\mathrm{d}t}=C\frac{\mathrm{d}u_o(t)}{\mathrm{d}t} \tag{5-7}$$

将式(5-7)代入式(5-6)，可得到系统的微分方程为

$$LC\frac{\mathrm{d}^2u_o(t)}{\mathrm{d}t^2}+RC\frac{\mathrm{d}u_o(t)}{\mathrm{d}t}+u_o(t)=u_i(t) \tag{5-8}$$

比较式(5-4)和式(5-8)可知，当两个方程式的系数相同时，从动态性能角度来看，两个系统是相同的。这就有可能利用电气系统来模拟机械系统，进行试验研究。而且从系统理论来说，就有可能脱离系统的具体物理属性，进行普遍意义的研究。

一般而言，线性定常系统的微分方程所建模型可由下述 n 阶微分方程描述

$$a_n\frac{\mathrm{d}^ny(t)}{\mathrm{d}t^n}+a_{n-1}\frac{\mathrm{d}^{n-1}y(t)}{\mathrm{d}t^{n-1}}+\cdots+a_1\frac{\mathrm{d}y(t)}{\mathrm{d}t}+a_0y(t)$$

$$=b_m\frac{\mathrm{d}^mu(t)}{\mathrm{d}t^m}+b_{m-1}\frac{\mathrm{d}^{m-1}u(t)}{\mathrm{d}t^{m-1}}+\cdots+b_1\frac{\mathrm{d}u(t)}{\mathrm{d}t}+b_0u(t)$$

式中，$y(t)$——系统输出量；$u(t)$——系统输入量；$a_0,a_1,\cdots,a_n$ 和 $b_0,b_1,\cdots,b_m$——与系统结构参数有关的常数。

2. *差分方程建模*

由于数字计算机、微处理器的迅速发展和广泛应用，数字控制器在许多场合都取代了模拟控制器。由于数字控制器接收、处理和传送的都是数字信号，所以如果在控制系统中有一处或几处信号不是时间的连续函数，而是以离散的脉冲序列或数字脉冲序列形式出现，那么这样的系统则称为离散控制系统。通常将系统中的离散信号是脉冲序列形式的离散系统称为采样控制系统或脉冲控制系统；将系统中的离散信号是数字序列形式的离散系统称为数字控制系统或计算机控制系统。

离散控制系统和连续控制系统相比，既有本质上的不同，又有分析研究方面的相似性。利用 Z 变换法研究离散系统，可以把连续系统中的一些概念和方法推广到线性离散系统的分析和设计中。

描述离散控制系统的时域数学模型主要是差分方程，这里通过例子简要介绍差分方程的基本概念及其建立方法。

与线性定常连续系统用线性微分方程描述相似，线性定常离散系统常用差分方程来描述。下面举例进行说明。

设离散控制系统的结构图如图 5-3 所示。

在第 k 个采样时间间隔中，零阶保持器的输出为 $e_h(t)=e(kT)$，$kT\leqslant t\leqslant(k+1)T$。

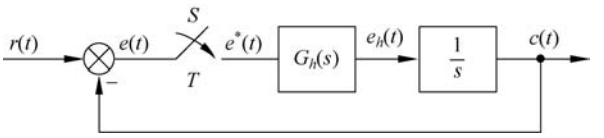

图 5-3　采样控制系统的结构图

在该周期内的输出 $c(t)$ 则为

$$c(t)=c(kT)+e(kT)(t-kT) \tag{5-9}$$

式中，$kT\leqslant t\leqslant(k+1)T$。

根据 $c[(k+1)T]=c(kT)+Te(kT)$，可简写为

$$c(k+1)=c(k)+Te(k) \tag{5-10}$$

考虑到 $e(k)=r(k)-c(k)$，将之代入式(5-10)，得

$$c(k+1)+(T-1)c(k)=Tr(k) \tag{5-11}$$

上式就是如图 5-3 所示采样控制系统的差分方程模型。

更一般的描述，线性定常离散系统通常可用如下 n 阶前向差分方程来描述：

$$\begin{aligned}&c(k+n)+a_1c(k+n-1)+\cdots+a_{n-1}c(k+1)+a_nc(k)\\&=b_0r(k+m)+b_1r(k+m-1)+\cdots+b_{m-1}r(k+1)+b_mr(k)\end{aligned}$$

式中，n 为系统阶次；k 为第 k 个采样周期，$a_1,a_2,\cdots,a_n$ 和 $b_0,b_1,\cdots,b_m$ 是与系统结构参数有关的常数。

3. 状态空间表达建模

从 20 世纪 50 年代后期开始，Bellman 等提出了状态变量法。在用状态空间法分析系统时，系统的动态特性是由状态变量构成的一阶微分方程组来描述的。在计算机上求解一阶微分方程组比求解与之相应的高阶微分方程容易得多，而且可以同时得到系统全部独立变量的响应，因而可以同时确定系统的全部内部运动状态。此外，状态空间法还可以方便地处理初始条件，可以用来分析设计多变量、时变和非线性控制系统，也可以应用于随机过程和采样数据系统，因此它是研究复杂控制系统的理论基础。现代控制理论是在引入状态和状态空间概念的基础上发展起来的。确定控制系统状态空间的描述，即建立状态空间的数学模型是一个基本问题。状态空间数学模型的建立需要理解如下基本概念。

状态：动态系统的状态是指系统的过去、现在和将来的运动状态。状态需要一组必要而充分的数据来说明。如 RLC 电路所示系统，这个系统的状态就是 RLC 电路每一时刻的回路电流和输出电压。

状态变量：系统的状态变量就是指可以完全确定系统运动状态的最小一组变量。一个用 n 阶微分方程描述的系统，就有 n 个独立变量，求得这 n 个独立变量的时间响应，系统的运动状态也就被完全确立了。因此，系统的状态变量就是 n 阶系统的 n 个独立变量。

设 $x_1(t),x_2(t),\cdots,x_n(t)$ 为系统的一组状态变量，满足下列两个条件：在初始时刻

$t=t_0$，这组变量 $x_1(t_0),x_2(t_0),\cdots,x_n(t_0)$ 的值可以表示系统在初始时刻的状态；当系统在 $t\geqslant t_0$ 的输入和上述初始状态确定以后，状态变量便能完全确定系统在任何 $t\geqslant t_0$ 时刻的行为。

对同一个系统来说，究竟选取哪些变量作为状态变量不是唯一的，关键是这些状态变量是相互独立的，且其个数应等于微分方程的阶数，因为微分方程的阶数唯一地取决于系统中独立储能元件的个数，因此，状态变量的个数就应等于系统独立储能元件的个数。还应该指出，状态变量不一定是物理上可量测或可观测的量，但通常选择易于测量或观测的量作为状态变量，这样在系统实现最佳控制规律时，就可以得到所有需要反馈的状态变量。

状态向量：如果完全描述一个系统的动态行为需要 n 个状态变量 $x_1(t),x_2(t),\cdots,x_n(t)$，那么这 n 个状态变量作分量所构成的向量 $\boldsymbol{X}(t)$ 叫作该系统的状态向量，记作 $\boldsymbol{X}(t)=[x_1(t)x_2(t)\cdots x_n(t)]^{\mathrm{T}}$。

状态空间：以状态变量 $x_1(t),x_2(t),\cdots,x_n(t)$ 为坐标所构成的 n 维空间称为状态空间。任何状态都可以用状态空间中的一个点来表示。即在特定时刻 t，状态向量 $\boldsymbol{X}(t)$ 在状态空间中是一点。已知初始时刻 t_0 的状态 $X(t_0)$，就得到状态空间中的一个初始点。随着时间的推移，状态 $X(t)$ 将在状态空间中描绘出一条轨迹，称为状态轨迹。这一轨线的形状完全由系统在 t_0 时刻的初始状态 $X(t_0)$ 和 $t\geqslant t_0$ 的输入以及系统的动态结构唯一决定。状态向量的状态空间模型联系了向量的代数结构和几何结构。

状态方程：描述系统状态变量与系统输入之间关系的一阶微分方程组称为状态方程。

对于式(5-8)所描述的 RLC 电路系统，若令 $x_1(t)=u_0(t)$，$x_2(t)=\dot{x}_1(t)$，即取 x_1、x_2 为此系统的一组状态变量，则可以得到一阶微分方程组：

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ -\dfrac{1}{LC} & -\dfrac{R}{L}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}0\\ \dfrac{1}{LC}\end{bmatrix}u_{\mathrm{i}} \tag{5-12}$$

式(5-12)即为该系统的状态方程，上式可简写成

$$\dot{\boldsymbol{X}}=\boldsymbol{A}\boldsymbol{X}+\boldsymbol{B}u_{\mathrm{i}} \tag{5-13}$$

式中：

$$\boldsymbol{A}=\begin{bmatrix}0 & 1\\ -\dfrac{1}{LC} & -\dfrac{R}{L}\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}0\\ \dfrac{1}{LC}\end{bmatrix},\quad \boldsymbol{X}=\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

输出方程：描述系统的状态变量与输出变量关系的一组代数方程称为输出方程。

对于式(5-8)所描述的 RLC 电路系统，指定 $u_{\mathrm{o}}(t)$ 为输出，则有

$$u_{\mathrm{o}}(t)=x_1(t) \tag{5-14}$$

写成

$$u_{\mathrm{o}}=\boldsymbol{C}^{\mathrm{T}}\boldsymbol{X} \tag{5-15}$$

式中，

$$\boldsymbol{C}^{\mathrm{T}}=[1\quad 0],\quad \boldsymbol{X}=\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

式(5-15)即为该系统的输出方程。

状态方程和输出方程一起构成一个系统动态的完整描述,称为系统的状态空间模型表达式。如式(5-13)和式(5-15)就是式(5-8)所描述的 RLC 电路系统的状态空间模型表达式。

一般地,对于一个复杂系统,它可以有 r 个输入,m 个输出,此时状态方程为

$$\begin{cases}\dot{x}_1=a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n+b_{11}u_1+\cdots+b_{1r}u_r\\ \dot{x}_2=a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n+b_{21}u_1+\cdots+b_{2r}u_r\\ \quad\vdots\\ \dot{x}_n=a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n+b_{n1}u_1+\cdots+b_{nr}u_r\end{cases}\tag{5-16}$$

输出方程不仅是状态变量的组合,而且在特殊情况下可以有输入向量的直接传递,因而有以下一般形式:

$$\begin{cases}y_1=c_{11}x_1+c_{12}x_2+\cdots+c_{1n}x_n+d_{11}u_1+\cdots+d_{1r}u_r\\ y_2=c_{21}x_1+c_{22}x_2+\cdots+c_{2n}x_n+d_{21}u_1+\cdots+d_{2r}u_r\\ \quad\vdots\\ y_m=a_{m1}x_1+a_{m2}x_2+\cdots+a_{mn}x_n+d_{m1}u_1+\cdots+d_{mr}u_r\end{cases}\tag{5-17}$$

多输入-多输出系统状态空间表达式的向量矩阵形式表示为

$$\begin{cases}\dot{\boldsymbol{x}}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u}\\ \boldsymbol{y}=\boldsymbol{C}\boldsymbol{x}+\boldsymbol{D}\boldsymbol{u}\end{cases}\tag{5-18}$$

式中,$\boldsymbol{x}$ 和 $\boldsymbol{A}$ 分别为 n 维状态向量和 $n\times n$ 状态矩阵;$\boldsymbol{u}$ 表示 r 维输入(或控制)向量;$\boldsymbol{y}$ 表示 m 维输出向量;$\boldsymbol{B}$ 表示 $n\times r$ 输入(或控制)矩阵;$\boldsymbol{C}$ 表示 $m\times n$ 输出矩阵;$\boldsymbol{D}$ 表示输入量的 $m\times r$ 直接传递矩阵。

下面简要叙述定常连续状态空间模型的离散化方法。

已知定常连续系统状态方程

$$\dot{\boldsymbol{x}}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u}\tag{5-19}$$

在 $\boldsymbol{x}(t_0)$ 以及输入 $\boldsymbol{u}(t)$ 作用下的解为

$$\boldsymbol{x}(t)=\boldsymbol{\Phi}(t-t_0)\boldsymbol{x}(t_0)+\int_{t_0}^{t}\boldsymbol{\Phi}(t-\tau)\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau\tag{5-20}$$

令 $t_0=kT$,有 $\boldsymbol{x}(t_0)=\boldsymbol{x}(kT)=\boldsymbol{x}(k)$,令 $t=(k+1)T$,则有

$\boldsymbol{x}[(k+1)T]=\boldsymbol{x}(k+1)$在 $t\in[k,k+1]$ 时,有 $\boldsymbol{u}(k)=\boldsymbol{u}(k+1)$为常数,于是其解为

$$\boldsymbol{x}(k+1)=\boldsymbol{\Phi}(T)\boldsymbol{x}(k)+\int_{kT}^{(k+1)T}\boldsymbol{\Phi}[(k+1)T-\tau]\boldsymbol{B}\mathrm{d}\tau\boldsymbol{u}(k)\tag{5-21}$$

记 $\boldsymbol{G}(t)=\int_{kT}^{(k+1)T}\boldsymbol{\Phi}[(k+1)T-\tau]\boldsymbol{B}\mathrm{d}\tau$,为了便于计算 $\boldsymbol{G}(T)$,引入变换 $(k+1)T-\tau=\tau'$,则有 $\boldsymbol{G}(t)=\int_{T}^{0}-\boldsymbol{\Phi}(\tau')\boldsymbol{B}\mathrm{d}\tau'=\int_{0}^{T}\boldsymbol{\Phi}(\tau)\boldsymbol{B}\mathrm{d}\tau$,因此离散化系统的状态方程为

$$\boldsymbol{x}(k+1)=\boldsymbol{\Phi}(T)\boldsymbol{x}(k)+\boldsymbol{G}(T)\boldsymbol{u}(k) \tag{5-22}$$

式中，$\boldsymbol{\Phi}(T)$根据连续系统的状态转移矩阵$\boldsymbol{\Phi}(t)$导出，即$\boldsymbol{\Phi}(T)=\boldsymbol{\Phi}(t)|_{t=T}$。

离散化系统输出方程为

$$\boldsymbol{y}(k)=\boldsymbol{C}\boldsymbol{x}(k)+\boldsymbol{D}\boldsymbol{u}(k) \tag{5-23}$$

4. 时域模型的仿真实现

(1) 对于微分方程模型的求解，第1章中已详细介绍了采用函数 dsolve 进行数值求解，采用函数 solve 进行解析求解的使用方法。

(2) 对于差分方程模型的求解，可以运用 MATLAB 提供的函数 filter 和 filtic。这两个函数的使用说明如下。

函数 filtic 为函数 filter 求解差分方程设定初始条件。

filtic 函数的使用格式为 Z=filtic(B,A,Y,X)。其中 B,A 分别为差分方程 a(1)*y(n)+a(2)*y(n−1)+a(3)*y(n−2)+…+a(nb+1)*y(n−nb)=b(1)*x(n)+b(2)*x(n−1)+…+b(nb+1)*x(n−nb)的输入和输出变量前的系数；X,Y 分别为输入和输出的初值。

函数 filter 用于求解差分方程。

filter 函数的使用格式为 Y=filter (B,A,X)。其中 B,A 分别为差分方程 a(1)*y(n)+a(2)*y(n−1)+a(3)*y(n−2)+…+a(nb+1)*y(n−nb)= b(1)*x(n)+b(2)*x(n−1)+…+b(nb+1)*x(n−nb)的输入和输出变量前的系数；X 为输入序列。

【例 5-3】 求解差分方程 y(n)−0.4y(n−1)−0.45y(n−2)=0.45x(n)+0.4x(n−1)−x(n−2)，初值为 y(−1)=0，y(−2)=1，x(−1)=1，x(−2)=2。

求解程序如下：

```
>> B = [0.45 0.4 -1];
>> A = [1 -0.4 -0.45];
>> x0 = [1 2];                          % 设定初值
>> y0 = [0 1];
>> N = 50;
>> n = [1:N-1]';
>> x = 0.2.^n;                          % 生成输入序列
>> Zi = filtic(B,A,y0,x0);              % 生成初始条件
>> [y,Zf] = filter(B,A,x,Zi);
>> plot(n,x,'R-',n,y,'b--');
>> xlabel('n');ylabel('(n)--y(n)');
```

运行结果如图 5-4 所示。

(3) 对于状态空间模型的求解，可以先用 ss 命令来建立状态空间模型，然后求解。

对于连续系统，其格式为 sys=ss(A,B,C,D)，其中 A,B,C,D 为描述线性连续系统的矩阵。当 sys1 是一个用传递函数表示的线性定常系统时，可以用命令 sys=ss(sys1)将其转换成为状态空间形式。也可以用命令 sys=ss(sys1,'min')计算出系统 sys 的最小实现。

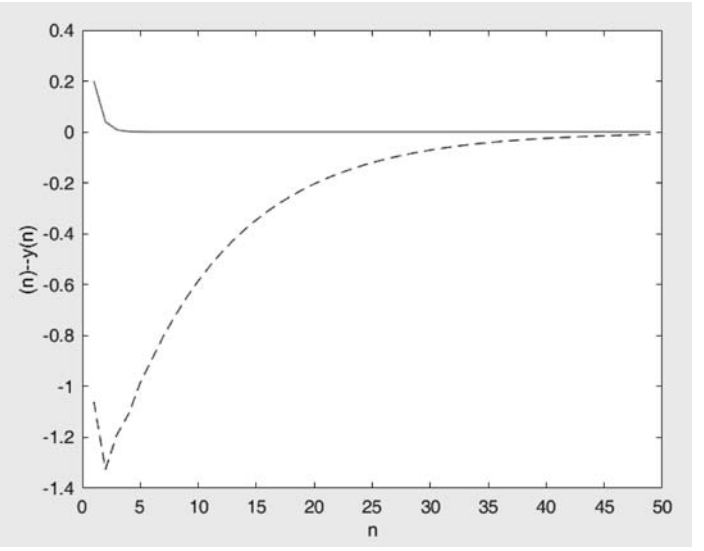

图 5-4　例 5-3 程序运行结果

和连续系统状态空间表达式的输入方法相类似，如果要输入离散系统的状态空间表达式：

$$\begin{cases} \boldsymbol{x}(k+1)=\boldsymbol{G}\boldsymbol{x}(k)+\boldsymbol{H}\boldsymbol{u}(k) \\ \boldsymbol{y}(k)=\boldsymbol{C}\boldsymbol{x}(k)+\boldsymbol{d}\boldsymbol{u}(k) \end{cases} \tag{5-24}$$

首先需要输入矩阵 G、H、C、d，然后输入语句 sys＝ss(G，H，C，d，T)，即可将其输入到 MATLAB 的工作空间中，并且用变量名来表示这个离散系统，其中 T 为采样时间。如果 Gyu 表示一个以脉冲传递函数描述的离散系统，也可以用 ss(Gyu)命令，将脉冲传递函数模型转换成状态空间表达式。

在 MATLAB 中，还提供了函数 c2d，其功能就是将连续时间的系统模型转换成离散时间的系统模型。其调用格式为 sysd＝c2d(sysc，T，method)。其中，输入参量 sysc 为连续时间的系统模型；T 为采样周期；method 用来指定离散化采用的方法。可以选用的离散化采用的方法有：'zoh'为采用零阶保持器；'foh'为采用一阶保持器；'tustin'为采用双线性逼近方法；'prewarm'为采用改进的 tustin 方法；'matched'为采用 SISO 系统的零极点匹配方法；当 method 默认，即调用格式为 sysd＝c2d(sysc，T)时，默认的方法是采用零阶保持器。

【例 5-4】 已知系统状态方程为

$\dot{\boldsymbol{x}}=\begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}\boldsymbol{x}+\begin{bmatrix} 0 \\ 1 \end{bmatrix}\boldsymbol{u}$，$\boldsymbol{x}(0)=\begin{bmatrix} 1 \\ 0 \end{bmatrix}$，求解系统在阶跃信号作用下的输出。

方法 1，数值求解，求解程序如下：

```
>> A=[0 1;-2 -3];
>> B=[0 1]';
```

```
>> C = [1 0;0 1];
>> D = [0 0]';
>> x0 = [1 0]';
>> t = 0:0.01:10;
>> u(length(t)) = 1;
>> u(:) = 1;
>> sys = ss(A,B,C,D);
>> lsim(sys,u,t,x0)
```

运行结果如图 5-5 所示。

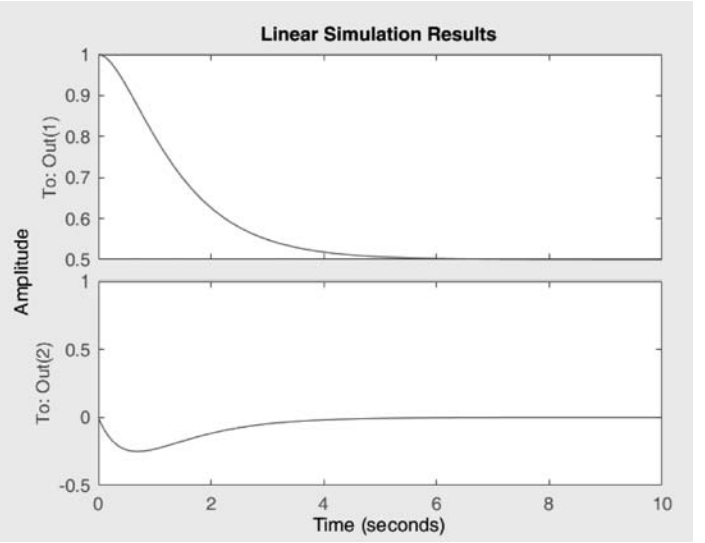

图 5-5　例 5-4 程序运行结果

方法 2，解析求解，求解程序如下：

```
>> syms s t x0 x tao p p0;
>> A = [0 1; -2 -3];
>> B = [0 1]';
>> I = [1 0;0 1];
>> E = s * I - A;
>> C = det(E);
>> D = collect(inv(E));
>> p0 = ilaplace(D);
>> x0 = [1 0]';
>> x1 = p0 * x0;
>> p = subs(p0,'t',(t - tao));
>> F = p * B * 1;
>> x2 = int(F,tao,0,t);
>> x = collect(x1 + x2)
```

运行结果如下：

```
x =
  2 * exp( - t) - exp( - 2 * t) + (exp( - 2 * t) * (exp(t) - 1)^2)/2
     2 * exp( - 2 * t) - 2 * exp( - t) + exp( - 2 * t) * (exp(t) - 1)
```

【例 5-5】 线性连续系统的状态方程为

$$\begin{cases}\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}\end{cases}$$

其中，

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{bmatrix},\quad \boldsymbol{B} = \begin{bmatrix} 1 & 0 \\ 2 & -1 \\ 0 & 2 \end{bmatrix},\quad \boldsymbol{C} = \begin{bmatrix} 1 & -1 & 0 \\ 2 & 1 & -1 \end{bmatrix},\quad \boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

采用零阶保持器将其离散化，设采样周期为 0.1 秒。求离散化的状态方程模型，及其在零初始条件下，该离散化系统的阶跃信号响应。

求解程序如下：

```
>> A = [0 1 0;0 0 1; - 6 - 11 - 6];
>> B = [1 0;2 - 1;0 2];
>> C = [1 - 1 0;2 1 - 1];
>> D = zeros(2);
>> T = 0.1;
>> sys = ss(A,B,C,D);
>> sysd = c2d(sys,T)
>> step(sysd)
```

运行结果如下：

```
sysd =
   a =
            x1         x2         x3
   x1     0.9991     0.0984   0.004097
   x2   - 0.02458     0.9541    0.07382
   x3    - 0.4429    - 0.8366     0.5112
   b =
             u1          u2
   x1      0.1099   - 0.004672
   x2      0.1959     - 0.0902
   x3     - 0.1164      0.1936
   c =
       x1  x2   x3
   y1   1   - 1   0
   y2   2   1   - 1
   d =
       u1  u2
```

```
    y1    0    0
    y2    0    0
 Sample time: 0.1 seconds
Discrete-time state-space model.
```

该离散化系统的阶跃信号响应如图 5-6 所示。

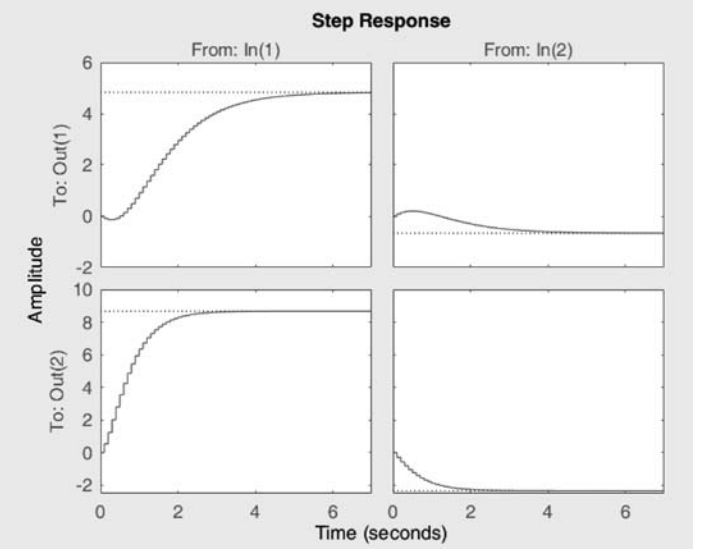

图 5-6　例 5-5 程序运行结果


视频讲解

5.2　频域建模方法及示例

控制系统的复频域数学模型主要有传递函数、结构图、信号流图和频率特性，下面先简要叙述这些模型的基本概念，再介绍相应的 MATLAB 函数实现。

1. 传递函数模型

对于描述控制系统的微分方程模型，在给定初始条件下的情况下，可以通过求解微分方程直接得到系统的输出响应，但如果方程阶次较高，则计算很烦琐，从而给系统的分析设计带来不便。经典控制理论的主要研究方法，都不是直接利用求解微分方程的方法，而是采用与微分方程有关的传递函数模型进行描述。传递函数是经典控制理论中最重要的数学模型。利用传递函数不必求解微分方程就可研究初始条件为零的系统在输入信号作用下的动态性能。利用传递函数还可研究系统参数变化或结构变化对动态过程的影响，因而极大地简化了系统分析的过程。另外，还可以把对系统性能的要求转化为对系统传递函数的要求，使综合设计问题易于实现。

所谓传递函数，即线性定常系统在零初始条件下，系统输出量的拉普拉斯变换与输入量

的拉普拉斯变换之比。如图 5-7 表示一个具有传递函数 $G(s)$ 的线性系统，图中表明，系统输入量与输出量的关系可以用传递函数联系起来。

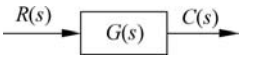

图 5-7　传递函数的图示

设线性定常系统为下述 n 阶线性常微分方程所描述：

$$a_n\frac{\mathrm{d}^n c(t)}{\mathrm{d}t^n}+a_{n-1}\frac{\mathrm{d}^{n-1}c(t)}{\mathrm{d}t^{n-1}}+\cdots+a_1\frac{\mathrm{d}c(t)}{\mathrm{d}t}+a_0$$
$$=b_m\frac{\mathrm{d}^m r(t)}{\mathrm{d}t^m}+b_{m-1}\frac{\mathrm{d}^{m-1}r(t)}{\mathrm{d}t^{m-1}}+\cdots+b_1\frac{\mathrm{d}r(t)}{\mathrm{d}t}+b_0 \tag{5-25}$$

式中，$c(t)$——系统输出量；$r(t)$——系统输入量。

在初始状态为零时，对上式两端取拉普拉斯变换，得

$$a_n s^n C(s)+a_{n-1}s^{n-1}C(s)+\cdots+a_1 sC(s)+a_0C(s)$$
$$=b_m s^m R(s)+b_{m-1}s^{m-1}R(s)+\cdots+b_1 sR(s)+b_0R(s) \tag{5-26}$$

式(5-26)用传递函数可表述为

$$G(s)=\frac{C(s)}{R(s)}=\frac{b_m s^m+b_{m-1}s^{m-1}+\cdots+b_1+b_0}{a_n s^n+a_{n-1}s^{n-1}+\cdots+a_1+a_0} \tag{5-27}$$

式中，$C(s)$——输出量的拉普拉斯变换；$R(s)$——输入量的拉普拉斯变换；$G(s)$——系统或环节的传递函数。

由于传递函数在经典控制理论中是非常重要的概念，故有必要对其性质、适用范围及表示形式等方面作出以下说明：传递函数只适用于描述线性定常系统；传递函数和微分方程一样，表征系统的运动特性，是系统的数学模型的一种表示形式，它和系统的运动方程是一一对应的。传递函数分子多项式系数及分母多项式系数，分别与相应微分方程的右端及左端微分算符多项式系数相对应。在零初始条件下，将微分方程的算符 $\mathrm{d}/\mathrm{d}t$ 用复数 s 置换便得到传递函数；反之，将传递函数多项式中的变量 s 用算法 $\mathrm{d}/\mathrm{d}t$ 置换便得到微分方程；传递函数是系统本身的一种属性，它只取决于系统的结构和参数，与输入量和输出量的大小和性质无关，也不反映系统内部的任何信息。且传递函数只反映系统的动态特性，而不反映系统物理性能上的差异，对于物理性质截然不同的系统，只要动态特性相同，它们的传递函数就具有相同的形式；传递函数为复变量 s 的真有理分式，即 $n\geqslant m$，因为系统或元件总是具有惯性的，而且输入系统的能量也是有限的；传递函数是在初始条件为零时定义的，因此，在非零初始条件下，传递函数不能完全表征系统的动态性能。另外，系统内部往往有多种变量，但传递函数只是通过系统输入量和输出量之间的关系来描述系统，而对内部其他变量的情况却无法得知。特别是某些变量不能由输出变量反映时，传递函数就不能正确表征系统的特征。现代控制理论采用状态空间法描述系统，引入了可控性和可观测性的概念，从而对控制系统进行全面的了解，可以弥补传递函数的不足；传递函数 $G(s)$ 的拉普拉斯逆变换是脉冲响应 $g(t)$。

作为线性定常离散控制系统的数学模型，采用脉冲传递函数描述。脉冲传递函数的定义与连续系统的传递函数的定义类似。

以图 5-8 为例，如果系统的初始条件为零，输入信号为 $r(t)$，采样后 $r^*(t)$ 的 Z 变换函数为 $R(z)$，系统连续部分的输出为 $c(t)$，采样后 $c^*(t)$ 的 Z 变换函数为 $C(z)$，则线性定常离散系统的脉冲传递函数定义为系统输出采样信号的 Z 变换与输入采样信号的 Z 变换之比，记作

$$G(z)=\frac{C(z)}{R(z)} \tag{5-28}$$

此处，零初始条件是指 $t<0$ 时，输入脉冲序列各采样值 $r(-T)$，$r(-2T)$，…及输出脉冲序列各采样值 $c(-T)$，$c(-2T)$，…均为零。

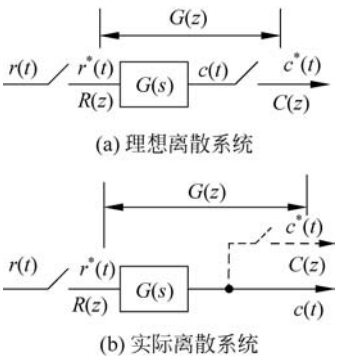

图 5-8　离散系统

由式(5-28)所描述的关系可以得知，输出的采样信号如下式所描述：

$$c^*(t)=Z^{-1}[C(z)]=Z^{-1}[G(z)R(z)] \tag{5-29}$$

由于输入信号的 Z 变换 $R(z)$ 通常为已知的，因此求 $c^*(t)$ 的关键在于求出系统的脉冲传递函数 $G(z)$。

关于线性定常离散控制系统的脉冲传递函数，还需着重指出：$G(s)$ 是一个线性环节的传递函数，而 $G(z)$ 表示的是线性环节与理想开关两者的组合的脉冲传递函数。如果不存在理想采样开关，那么式(5-29)是不成立的；利用线性环节的脉冲传递函数只能得出在采样时刻上的信息。为了强调这一点，往往在环节的输出端画上一个假想的同步理想开关，如图 5-8(b)所示。实际上，线性环节的输出仍然是一个连续信号 $c(t)$。

脉冲传递函数的求法，连续系统的脉冲传递函数 $G(z)$，可以通过其传递函数 $G(s)$ 求取。具体步骤如下：

(1) 对连续系统或元件的传递函数 $G(s)$ 取拉普拉斯逆变换，求得脉冲响应 $g(t)$ 为

$$g(t)=L^{-1}[G(s)]$$

(2) 对 $g(t)$ 进行 Z 变换，则得到系统或元件的脉冲传递函数 $G(z)$。

脉冲传递函数还可由连续系统的传递函数，经部分分式法，通过查 Z 变换表求得。

MATLAB 提供了如表 5-1 所示传递函数、脉冲传递函数、状态空间模型的仿真函数。

表 5-1　传递函数、脉冲传递函数、状态空间模型的仿真函数

函　　数	使 用 说 明
SYS=TF(NUM,DEN)	返回变量 SYS 为连续系统传递函数模型
SYS=TF(NUM,DEN,TS)	返回变量 SYS 为离散系统传递函数模型。TS 为采样周期，当 TS=−1 或者 TS=[]时，表示系统采样周期未定义
S=TF('s')	定义拉普拉斯变换算子（拉普拉斯 variable），以原形式输入传递函数
Z=TF('z',TS)	定义 Z 变换算子及采样时间 TS，以原形式输入传递函数
PRINTSYS(NUM,DEN,'s')	将系统传递函数以分式的形式打印出来，'s'表示传递函数变量

续表

函　　数	使 用 说 明
PRINTSYS(NUM,DEN,'z')	将系统传递函数以分式的形式打印出来,'z'表示传递函数变量
GET(sys)	可获得传递函数模型对象 sys 的所有信息
SET(sys,'Property',Value,…)	为系统不同属性设定值
[NUM,DEN]=TFDATA(SYS,'v')	以行向量的形式返回传递函数分子分母多项式
C=CONV(A, B)	多项式 A，B 以系数行向量表示,进行相乘。结果 C 仍以系数行向量表示
sys=zpk(z,p,k)	得到连续系统的零极点增益模型
sys=zpk(z,p,k,Ts)	得到连续系统的零极点增益模型,采样时间为 Ts
s=zpk('s')	得到拉普拉斯算子,按原格式输入系统,得到系统 zpk 模型
z=zpk('z',Ts)	得到 Z 变换算子和采样时间 Ts,按原格式输入系统,得到系统 zpk 模型
[Z,P,K]=ZPKDATA (SYS,'v')	得到系统的零极点和增益,参数'v'表示以向量形式表示
[p,z] =pzmap(sys)	返回系统零极点
pzmap(sys)	得到系统零极点分布图
sys=ss(A,B,C,D)	由 A,B,C,D 矩阵直接得到连续系统状态空间模型
sys=ss(A,B,C,D,Ts)	由 A,B,C,D 矩阵和采样时间 Ts 直接得到离散系统状态空间模型
[A,B,C,D] =ssdata(sys)	得到连续系统参数
[A,B,C,D,Ts] =ssdata(sys)	得到离散系统参数

MATLAB 提供的传递函数、脉冲传递函数、状态空间模型之间的转换函数见表 5-2。

表 5-2　传递函数、脉冲传递函数、状态空间模型之间的转换函数

函　数　名	使 用 说 明
tfsys=tf(sys)	将其他类型的模型转换为多项式传递函数模型
zsys=zpk(sys)	将其他类型的模型转换为 zpk 模型
sys_ss=ss(sys)	将其他类型的模型转换为 ss 模型
[A,B,C,D]=tf2ss(num,den)	tf 模型参数转换为 ss 模型参数
[num,den]=ss2tf(A,B,C,D,iu)	ss 模型参数转换为 tf 模型参数,iu 表示对应第 i 路传递函数
[z,p,k]=tf2zp(num,den)	tf 模型参数转换为 zpk 模型参数
[num,den]=zp2tf(z,p,k)	zpk 模型参数转换为 tf 模型参数
[A,B,C,D]=zp2ss(z,p,k)	zpk 模型参数转换为 ss 模型参数
[z,p,k]=ss2zp(A,B,C,D,i)	ss 模型参数转换为 zpk 模型参数,iu 表示对应第 i 路传递函数
sys_min=minreal(sys)	对传递函数 sys 进行约分后,输出最小实现系统
sysd=c2d(sysc,Ts)	将连续系统转换为采样周期为 Ts 的离散系统
sysd=c2d(sysc,Ts, 'method')	指定连续系统的离散化方法
[sysd,G] =c2d(sysc,Ts, 'method')	对于 SS 模型,求得初始条件的转换阵 G
[Ad,Bd,Cd,Dd] =c2dm(A,B,C,D,Ts,'method')	连续 SS 模型的离散化

续表

函 数 名	使 用 说 明
sysc=d2c(sysd)	将离散系统转换为连续系统
sysc=d2c(sysd,method)	指定离散系统的连续化方法 method
[Ac,Bc,Cc,Dc] = d2cm(A,B,C,D,Ts,'method')	用于离散 SS 模型的连续化
sysd1 =d2d(sysd,Ts)	改变采样周期,生成新的离散系统

MATLAB 提供的模型特征分析的相关函数见表 5-3。

表 5-3 模型特征分析的相关函数

函 数 名	使 用 说 明
class	返回模型的类型名称,'tf'、'zpk'、'ss' 或者'frd'
hasdelay	如果 LTI 模型有任何类型的滞后时间,则返回 1
isa	判断 LTI 模型的输入参量是否为指定的类型
isct	判断模型是否为连续系统模型
isdt	判断模型是否为离散系统模型
isempty	判断 LTI 模型是否为空
isproper	判断模型是否为正则系统(传递函数分母阶次大于或等于分子的阶次)
issiso	判断模型是否为 MIMO 系统
ndims	返回 LTI 数组的维数
reshape	改变 LTI 数组的形状
size	返回输入输出状态的维数
cover	计算输出的协方差和状态的协方差
damp	求取系统特征根的无阻尼自振频率和阻尼比
dcgain	返回系统的低频增益
dsort	离散系统的极点按幅值排序
esort	连续系统的极点按实部排序
norm	LTI 模型的范数
pole,eig	求系统的极点
pzmap	求取系统的零极点分布
zero	LIT 系统的传输零点

【例 5-6】 已知传递函数模型 $G(s)=\dfrac{10(2s+1)}{s^2(s^2+7s+13)}$,将其输入 MATLAB 工作空间中。

方法 1,程序如下:

```
>> num = conv(10,[2,1]);
>> den = conv([1 0 0],[1 7 13]);
>> G = tf(num,den)
```

运行结果如下：

```
G =
        20 s + 10
  ---------------------
  s^4 + 7 s^3 + 13 s^2
Continuous - time transfer function
```

方法 2，程序如下：

```
>> s = tf('s');
>> G = 10 * (2 * s + 1)/s^2/(s^2 + 7 * s + 13)
```

运行结果如下：

```
G =
        20 s + 10
  ---------------------
  s^4 + 7 s^3 + 13 s^2
Continuous - time transfer function
```

【例 5-7】 设置传递函数模型 $G(s)=\dfrac{10(2s+1)}{s^2(s^2+7s+13)}$，时间延迟常数为 $\tau=1.2$，即系统模型为 $G(s)\mathrm{e}^{-1.2s}$。

可以通过在已有 MATLAB 模型基础上设置时间延迟常数来完成模型的输入，程序如下：

```
>> s = tf('s');
>> G = 10 * (2 * s + 1)/s^2/(s^2 + 7 * s + 13);
>> set(G,'ioDelay',1.2) % 或 G.ioDelay = 1.2
>> G
```

运行结果如下：

```
G =
                      20 s + 10
  exp( - 1.2 * s) * ---------------------
                    s^4 + 7 s^3 + 13 s^2
Continuous - time transfer function
```

【例 5-8】 已知一系统的传递函数 $G(s)=\dfrac{7s^2+2s+8}{4s^3+12s^2+4s+2}$，求取其零极点向量和增益值，并得到系统的零极点增益模型。

实现程序如下：

```
>> Gtf = tf([7 2 8],[4 12 4 2]);
>> [z,p,k] = zpkdata(Gtf,'v');
```

```
>> Gzpk = zpk(z,p,k)
>> [p1,z1] = pzmap(Gtf)
```

运行结果如下：

```
Gzpk =
    1.75 (s^2 + 0.2857s + 1.143)
  ---------------------------------
  (s + 2.698) (s^2 + 0.302s + 0.1853)
Continuous - time zero/pole/gain model
p1 =
 - 2.6980 + 0.0000i
 - 0.1510 + 0.4031i
 - 0.1510 - 0.4031i
z1 =
 - 0.1429 + 1.0595i
 - 0.1429 - 1.0595i
```

【例 5-9】 已知系统的零极点模型 $G(s)=\dfrac{5(s+2)(s+4)}{(s+1)(s+3)}$，求其 TF 模型及状态空间模型。

实现程序如下：

```
>> z = [ - 2  - 4]';
>> p = [ - 1  - 3]';
>> k = 5;
>> Gzpk = zpk(z,p,k);
>>[a,b,c,d] = zp2ss(z,p,k)
>>[num,den] = zp2tf(z,p,k)
>> Gtf = tf(Gzpk)
```

运行结果如下：

```
a =
 - 4.0000    - 1.7321
    1.7321          0
b =
     1
     0
c =
   10.0000   14.4338
d =
     5
num =
     5    30    40
den =
     1     4     3
Gtf =
```

```
   5 s^2 + 30 s + 40
  ------------------
    s^2 + 4 s + 3
Continuous-time transfer function
```

【例 5-10】 系统的被控对象传递函数为 $G(s)=\dfrac{10}{(s+2)(s+5)}$，采样周期 $T_s=0.01$s，试将其进行离散化。

实现程序如下：

```
>> num = 10;
>> den = conv([1,2],[1,5]);
>> ts = 0.01;
>> sysc = tf(num,den);
>> sysd = c2d(sysc,ts)
```

运行结果如下：

```
sysd =
  0.0004885 z + 0.0004772
  -----------------------
  z^2 - 1.931 z + 0.9324
Sample time: 0.01 seconds
Discrete-time transfer function
```

2. 结构图模型

传递函数是由代数方程组通过消去系统中间变量得到的，如果系统结构复杂，方程组数目较多，那么消去中间变量就比较麻烦，并且中间变量的传递过程在系统输入与输出关系得不到反映，因此，结构图作为一种数学模型，在控制理论中得到了广泛的应用。结构图是将方块图与传递函数结合起来的一种将控制系统图形化了的数学模型。如果把组成系统的各个环节用方块表示，在方块内标出表征此环节输入输出关系的传递函数，并将环节的输入量、输出量改用拉普拉斯变换来表示，这种图形成为动态结构图，简称结构图。如果按照信号的传递方向将各环节的结构图依次连接起来，形成一个整体，这就是系统结构图。结构图不但能清楚表明系统的组成和信号的传递方向，而且能清楚地表示出系统信号传递过程中的数学关系。

建立系统结构图的步骤如下：

（1）首先应分别列写系统各元件的微分方程，在建立微分方程时，注意分清输入量和输出量，同时应考虑相邻元件之间是否存在负载效应。

（2）设初始条件为零时，将各元件的微分方程进行取拉普拉斯变换，并作出各元件的结构图。

（3）将系统的输入量放在最左边，输出量放在最右边，按照各元件的信号流向，用信号

线依次将各元件的结构图连接起来，便构成系统的结构图。

MATLAB 提供结构图模型的化简涉及的函数如表 5-4 所示

表 5-4　结构图模型的相关操作函数

函　数　名	使 用 说 明
sys＝parallel(sys1,sys2)	并联两个系统，等效于 sys＝sys1＋sys2
sys＝parallel(sys1,sys2,inp1,inp2,out1,out2)	对 MIMO 系统，表示 sys1 的输入 inp1 与 sys2 的输入 inp2 相连，sys1 输出 out1 与 sys2 输出 out2 相连
sys＝series(sys1,sys2)	串联两个系统，等效于 sys＝sys2 * sys1
sys＝feedback(sys1,sys2)	两系统负反馈连接，默认格式
sys＝feedback(sys1,sys2,sign)	sign＝－1 表示负反馈，sign＝1 表示正反馈。等效于 sys＝sys1/(1±sys1 * sys2)
sys＝feedback(sys1,sys2,feedin,feedout,sign)	对 MIMO 系统，部分反馈连接。sys1 的指定输出 feedout 连接到 sys2 的输入，而 sys2 的输出连接到 sys1 的指定输入 feedin，最终实现的反馈系统与 sys1 具有相同的输入、输入端。sign 标识正负反馈
sys＝append(sys1,sys2,…,sysN)	将子系统 sys1,sys2,…,sysN 的所有输入作为系统的输入，所有输出作为系统输出，且各子系统间没有信号连接，从而扩展为一个系统
sysc＝connect(sys，Q，inputs，outputs)	将多个子系统按照一定的连接方式构成一个系统。sys 是待连接的子系统被 append 函数扩展后的系统。Q 矩阵声明了子系统的连接方式。Q 矩阵的行向量声明了 sys 输入信号的连接方式，每个行向量的第 1 个元素为 sys 系统的输入端口号，其他元素为与该输入信号相连接的 sys 端口号。inputs 声明了整个系统的输入信号是由 sys 系统的哪些输入端口号构成。outputs 声明了整个系统的输出信号是由 sys 系统的哪些输出端口号构成

【例 5-11】 化简如图 5-9 所示的系统，求系统的传递函数。

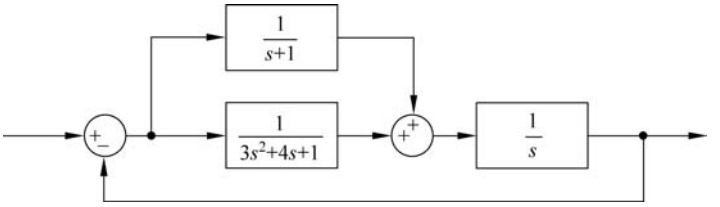

图 5-9　例 5-11 系统图

实现程序如下：

```
>> clear
>> G1 = tf(1,[1 1]);
>> G2 = tf(1,[3 4 1]);
>> Gp = G1 + G2;
>> G3 = tf(1,[1 0]);
>> Gs = series(G3,Gp);
```

```
>> Gc = Gs/(1 + Gs)
>> Gc1 = minreal(Gc)
```

运行结果如下：

```
Gc =
            9 s^6 + 36 s^5 + 56 s^4 + 42 s^3 + 15 s^2 + 2 s
  ------------------------------------------------------------------------
  9 s^8 + 42 s^7 + 88 s^6 + 112 s^5 + 95 s^4 + 52 s^3 + 16 s^2 + 2 s
Continuous - time transfer function.
Gc1 =
                    s^4 + 3.667 s^3 + 5 s^2 + 3 s + 0.6667
  ------------------------------------------------------------------------
  s^6 + 4.333 s^5 + 8.333 s^4 + 9.667 s^3 + 7.333 s^2 + 3.333 s + 0.6667
Continuous - time transfer function.
```

3. 信号流图模型

系统的结构图可以用来直接绘制信号流图，以图 5-10 为例简要说明信号流图的相关概念。

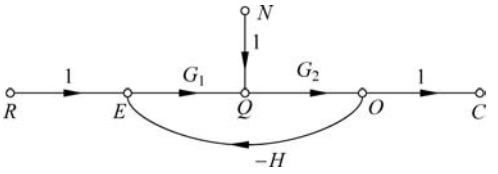

图 5-10　典型闭环控制系统的信号流图

(1) 源节点(输入节点)：只有输出支路而没有输入支路的节点，称为源节点。它一般表示系统的输入变量，亦称输入节点，如图 5-10 中的节点 R 和 N。

(2) 阱节点(输出节点)：只有输入支路而没有输出支路的节点，称为阱节点。它一般表示系统的输出变量，亦称输出节点，如图 5-10 中的节点 C。

(3) 混合节点：既有输入支路又有输出支路的节点，称为混合节点，如图 5-10 中的节点 E、Q、O。

(4) 通路：沿着支路箭头的方向顺序穿过各相连支路的路径，如图 5-10 中的 R、E、Q、O、C 等。

(5) 前向通路：从源节点出发并且终止于阱节点，与其他节点相交不多于一次的通路称为前向通路，如图 5-10 中的 N、Q、O、C 等。

(6) 回路：起点和终点在同一个节点，并且与其他节点相交不多于一次的闭合路径称为回路，如图 5-10 中的 E、Q、O、E。

(7) 前向通路传输(增益)：前向通路中各支路传输(增益)的乘积。

(8) 回路传输(增益)：回路中各支路传输(增益)的乘积。

（9）不接触回路：信号流图中，没有任何公共节点的回路，称为不接触回路或互不接触回路。

用信号流图代替系统的结构图，其优点在于不简化信号流图的情况下，利用梅逊公式直接求得源节点和阱节点之间的总增益。对于动态系统来说，这个总增益就是系统相应的输入和输出间的传递函数。

计算任意输入节点和输出节点之间传递函数 $G(s)$ 的梅逊公式为

$$G(s)=\frac{1}{\Delta}\cdot\sum_{k=1}^{n}P_k\Delta_k \tag{5-30}$$

式中，Δ——特征式，其计算公式为

$$\Delta=1-\sum L_a+\sum L_bL_c-\sum L_dL_eL_f+\cdots$$

n——从输入节点到输出节点间前向通路的条数；

P_k——从输入节点到输出接点间第 k 条前向通路的总增益；

$\sum L_a$——所有不同回路的回路增益之和；

$\sum L_bL_c$——所有两两互不接触回路的回路增益乘积之和；

$\sum L_dL_eL_f$——所有 3 个互不接触回路的回路增益乘积之和。

Δ_k——第 k 条前向通路的余子式，即把与该通路相接触的回路增益置为 0 后，特征式 Δ 所余下的部分。

【例 5-12】 根据典型闭环控制系统的信号流图，求传递函数 $C(s)/R(s)$。

实现程序如下：

```
>> syms g1 g2 H
>> syms R x1 x2 x3 x4 N
>> RaC = solve('x1 = R - x3 * H','x2 = x1 * g1','x3 = x2 * g2','x4 = x3','x1,x2,x3,x4');
>> pretty(simplify(RaC.x4/R))
```

运行结果如下：

```
   g1 g2
------------
H g1 g2 + 1
```

4. 频率特性模型

设线性定常系统的传递函数为 $G(s)$，其对正弦输入信号的稳态响应仍然是与输入信号同频率的正弦信号，输出信号的振幅是输入信号的 $|G(j\omega)|$ 倍，输出信号相对输入信号的相移为 $\varphi=\angle G(j\omega)$，输出信号的振幅及相移都是角频率 ω 的函数。

$G(j\omega)=|G(j\omega)|e^{j\angle G(j\omega)}$ 称为系统的频率特性，它表明了正弦信号作用下，系统的稳态输出与输入信号的关系。其中，$|G(j\omega)|$ 为幅频特性，它反映了系统在不同频率的正弦信号

作用下，稳态输出的幅值与输入信号幅值之比。$\angle G(j\omega)=\arctan\dfrac{\mathrm{Im}G(j\omega)}{\mathrm{Re}G(j\omega)}$称为相频特性，它反映了系统在不同频率的正弦信号作用下，输出信号相对输入信号的相移。系统的幅频特性和相频特性统称为系统的频率特性。

比较系统的频率特性和传递函数可知，频率特性与传递函数有如下关系：

$$G(j\omega)=G(s)\big|_{s=j\omega} \tag{5-31}$$

一般地，若系统具有以下传递函数：

$$G(s)=\frac{b_m s^m+b_{m-1}s^{m-1}+\cdots+b_1 s+b_0}{a_n s^n+a_{n-1}s^{n-1}+\cdots+a_1 s+a_0} \tag{5-32}$$

则系统频率特性可写为

$$G(j\omega)=\frac{b_m(j\omega)^m+b_{m-1}(j\omega)^{m-1}+\cdots+b_1(j\omega)+b_0}{a_n(j\omega)^n+a_{n-1}(j\omega)^{n-1}+\cdots+a_1(j\omega)+a_0} \tag{5-33}$$

由式(5-32)可推导出线性定常系统的频率特性。对于稳定的系统，可以由实验的方法确定系统的频率特性，即在系统的输入端作用不同频率的正弦信号，在输出端测得相应的稳态输出的幅值和相角，根据幅值比和相位差，就可得到系统的频率特性。对于不稳定的系统，则不能由实验的方法得到系统的频率特性，这是由于系统传递函数中不稳定极点会产生发散或振荡的分量，随时间推移，其瞬态分量不会消失，所以不稳定系统的频率特性是观察不到的。

频率特性的曲线表示方法主要有 3 种形式，即对数坐标图、极坐标图、对数幅相图。

对数坐标图又称伯德(Bode)图，包括对数幅频和对数相频两条曲线。在实际应用中，经常采用这种曲线来表示系统的频率特性。对数幅频特性曲线的横坐标是频率 ω，按对数分度，单位是 rad/s。纵坐标表示对数幅频特性的函数值，采用线性分度，单位是 dB。对数幅频特性用 $L(\omega)$表示，定义为：$L(\omega)=20\log|G(j\omega)|$，对数相频特性曲线的横坐标也是频率 ω，按对数分度，单位是 rad/s。纵坐标表示相频特性的函数值，记作 $\varphi(\omega)$，单位是度。

极坐标图又称幅相频率特性曲线、奈奎斯特曲线。其特点是以角频率 ω 作自变量，把幅频特性和相频特性用一条曲线同时表示在复平面上。

对数幅相图又称 Nichols 曲线，是将对数幅频特性和对数相频特性两张图，在角频率 ω 为参变量的情况下合成一张图。即以相位 $\varphi(\omega)$为横坐标，以 $20\log A(\omega)$为纵坐标，以 ω 为参变量的一种图示法。

MATLAB 提供的频率特性相关的函数如表 5-5 所示，注意 bode、nyquist、nichols 函数前加上的 d，可得到绘制离散系统的对应功能图的函数，一般还要指定采样周期。

表 5-5　频率特性相关函数的说明

函　数　名	使 用 说 明
bode(G)	绘制系统伯德图
bode(G,w)	绘制指定频率范围的系统伯德图

续表

函 数 名	使 用 说 明
bode(G1,'r——',G2,'gx',…)	同时绘制多系统伯德图。图形属性参数可选
[mag,phase,w]=bode(G)	返回系统伯德图相应的幅值、相位和频率向量，可用 magdb=20 * log10(mag)将幅值转换为分贝值
[mag,phase]=bode(G,w)	返回系统伯德图与指定 w 相应的幅值、相位
nyquist(sys)	绘制系统奈奎斯特图。系统自动选取频率范围
nyquist(sys,w)	绘制系统奈奎斯特图。由用户指定选取频率范围
nyquist(G1,'r——',G2,'gx',…)	同时绘制多系统带图形参数奈奎斯特图
[re,im,w]=nyquist(sys)	返回系统奈奎斯特图相应的实部、虚部和频率向量
[re,im]=nyquist(sys,w)	返回系统奈奎斯特图与指定 w 相应的实部、虚部
nichols(G)	绘制系统 Nichols 图
nichols(G,w)	绘制指定频率范围的系统 Nichols 图
nichols(G1,'r——',G2,'gx',…)	同时绘制多系统带图形参数 Nichols 图
[mag,phase,w]=nichols(G)	返回系统 Nichols 图相应的幅值、相位和频率向量
[mag,phase]=nichols(G,w)	返回系统 Nichols 图与指定 w 相应的幅值、相位
sigma(sys)	绘制系统 sys 的奇异值曲线
evalfr(sys,x)	计算系统 sys 的单个复频率点 x 的频率响应
freqresp(sys,w)	计算系统 sys 在给定频率区间 w 的频率响应
ngrid	在 Nichols 曲线图上绘制等 M 圆和等 N 圆
ngrid('new')	绘制网格前清除原图，然后设置 hold on

【例 5-13】 二阶振荡环节的传递函数为 $G(s)=\dfrac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$，绘制 ζ 取不同值时的伯德图，$\omega_n=5$，ζ 取 0.1∶0.2∶1.0。

实现程序如下：

```
>> wn = 5;
>> kosi = [0.1:0.2:1.0];
>> w = logspace( - 1,1,100);
>> figure(1)
>> num = [wn.^2];
>> for kos = kosi
>> den = [1 2 * kos * wn wn.^2];
>>[mag,pha,w1] = bode(num,den,w);
>> subplot(2,1,1);hold on
>> semilogx(w1,mag);
>> subplot(2,1,2);hold on
>> semilogx(w1,pha);
>> end
>> subplot(2,1,1);grid on
>> title('Bode Plot');
>> xlabel('Frequency(rad/sec)');
```

```
>> ylabel('Gain dB');
>> subplot(2,1,2);grid on
>> xlabel('Frequency(rad/sec)');
>> ylabel('Phase deg');
>> hold off
```

运行结果如图 5-11 所示。

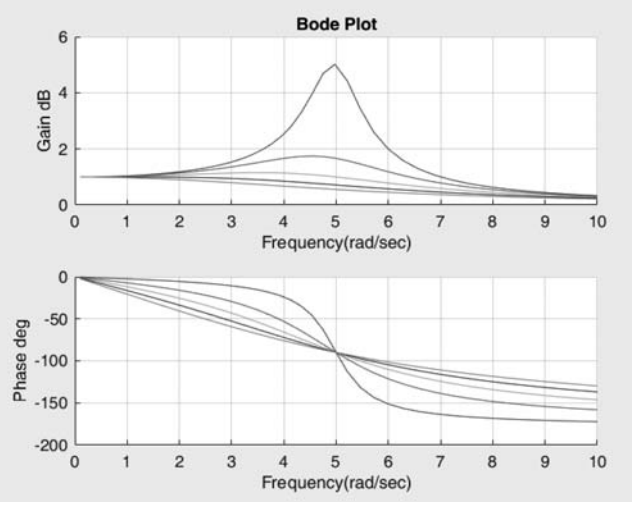

图 5-11　例 5-13 程序运行结果图

【例 5-14】 系统的传递函数为 $G(s)=\dfrac{30(10s+1)}{(3s+1)(0.5s+1)(0.03s+1)(s^2+s+1)}$，采用 MATLAB 绘制系统的渐进幅频特性。


视频讲解

先编写 asympbode.m 函数用于计算绘制渐进幅频特性的数据。

```
function [wpos,ypos] = asympbode(G,w)
G1 = zpk(G);
wpos = [];
pos1 = [];
if nargin == 1,w = freqint2(G);
end
zer = G1.z{1}; pol = G1.p{1};
gain = G1.k;
for i = 1:length(zer);
    ifisreal(zer(i))
        wpos = [wpos,abs(zer(i))];
        pos1 = [pos1,20];
    else
        if imag(zer(i))> 0
            wpos = [wpos,abs(zer(i))];
```

```
            pos1 = [pos1,40];
        end
    end
end
for i = 1:length(pol);
    if isreal(pol(i))
    wpos = [wpos,abs(pol(i))];
    pos1 = [pos1, - 20];
    else
        if imag(pol(i))> 0
            wpos = [wpos,abs(pol(i))];
            pos1 = [pos1, - 40];
        end
    end
end
wpos = [wpos w(1) w(length(w))];
pos1 = [pos1,0,0];
[wpos,ii] = sort(wpos);
pos1 = pos1(ii);
ii = find(abs(wpos)< eps);
kslp = 0;
w_start = 1000 * eps;
if length(ii)> 0
    kslp = sum(pos1(ii));
    ii = (ii(length(ii)) + 1):length(wpos);
    wpos = wpos(ii);
    pos1 = pos1(ii);
end
while 1
    [ypos1,pp] = bode(G,w_start);
    if isinf(ypos1),w_start = w_start * 10;
    else break;
    end
end
wpos = [w_start wpos];
ypos(1) = 20 * log10(ypos1);
pos1 = [kslp pos1];
for i = 2:length(wpos)
    kslp = sum(pos1(1:i - 1));
ypos(i) = ypos(i - 1) + kslp * log10(wpos(i)/wpos(i - 1));
end
ii = find(wpos > = w(1)&wpos < = w(length(w)));
wpos = wpos(ii);
ypos = ypos(ii);
```

然后在命令窗口中输入：

```
>> s = tf('s');
>> G1 = 30 * (10 * s + 1)/((3 * s + 1) * (0.5 * s + 1) * (0.03 * s + 1) * (s * s + s + 1));
>> w = 10e - 3:0.1:1000;
>>[x1,y1] = asympbode (G1,w);
>> semilogx(x1,y1)
>> grid;
```

运行结果如图 5-12 所示。

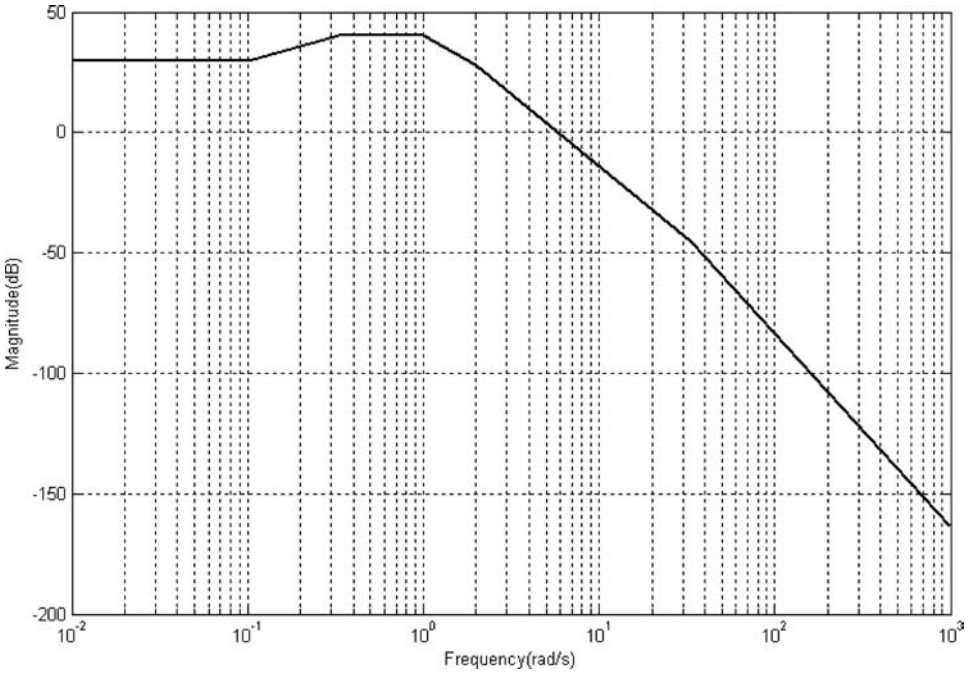

图 5-12　例 5-14 程序运行结果图

5.3　神经网络建模及示例

神经网络是由大量人工神经元(处理单元)广泛互联而成的网络,它是在现代神经生物学和认识科学对人类信息处理研究的基础上提出来的,具有很强的自适应性和学习能力、非线性映射能力、鲁棒性和容错能力。充分地将这些神经网络特性应用于控制领域,可使控制系统的智能化向前迈进一大步。随着被控系统越来越复杂,人们对控制系统的要求越来越高,特别是要求控制系统能适应不确定的、时变的对象与环境。传统的基于精确模型的控制方法难以适应要求,现在关于控制的概念也已更加广泛,要求包括一些决策、规划以及学习功能。神经网络由于具有上述优点而越来越受到人们的重视。本书介绍最为常用的 BP (Back Propagation)网络。

人工神经元是神经网络的基本元素,其原理可以用图 5-13 表示。

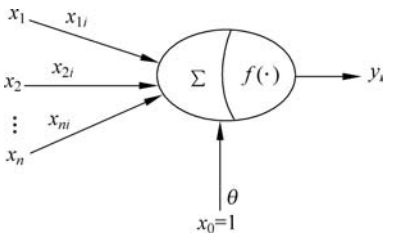

图 5-13　人工神经元结构示意图

人工神经元是对生物神经元的一种模拟与简化，它是神经网络的基本处理单元。如图所示为一种简化的人工神经元结构。它是一个多输入、单输出的非线性元件。其输入、输出关系为

$$I_i = \sum_{j=1}^{n} w_{ji} x_j - \theta_i$$

$$y_i = f(I_i)$$

其中，x_j（$j=1,2,\cdots,n$）是从其他神经元传来的输入信号；w_{ji} 表示神经元 i 到神经元 j 的连接权值；θ_i 为阈值；$f(\cdot)$称为激发函数或作用函数。方便起见，常把 $-\theta_i$ 看成是恒等于 1 的输入 x_0 的权值，则上式可写成

$$I_i = \sum_{j=0}^{n} w_{ji} x_j$$

其中，$w_{0i}=-\theta_i$，$x_0=1$。

输出激发函数 $f(\cdot)$又称为变换函数，它决定神经元（节点）的输出。该输出为 0 或 1，取决于其他输入之和是大于还是小于内部阈值 θ_i。$f(\cdot)$一般具有非线性特征。常用的激活函数有线性函数、斜坡函数、阈值函数，这 3 个激活函数都属于线性函数，还有两个常用的非线性激活函数。分别是 S 形函数和双极 S 形函数。双极 S 形函数与 S 形函数的主要区别在于函数的值域，双极 S 形函数值域是(−1,1)，而 S 形函数值域是(0,1)。由于 S 形函数与双极 S 形函数都是可导的，因此适合用在 BP 神经网络中。根据网络中神经元的互联方式，常见网络结构主要可以分为如下 3 类。

前馈神经网络：前馈网络也称前向网络。这种网络只在训练过程会有反馈信号，而在分类过程中数据只能向前传送，直到到达输出层，层间没有向后的反馈信号，因此被称为前馈神经网络。感知机与 BP 神经网络就属于前馈网络。

反馈神经网络：反馈型神经网络是一种从输出到输入具有反馈连接的神经网络，其结构比前馈网络要复杂得多。典型的反馈型神经网络有 Elman 网络和 Hopfield 网络。

自组织网络：自组织神经网络是一种无导师学习网络。它通过自动寻找样本中的内在规律和本质属性，自组织、自适应地改变网络参数与结构。

误差反向传播神经网络，简称 BP 网络，是一种单向传播的多层前向网络。在模式识别、图像处理、系统辨识、函数拟合、优化计算、最优预测和自适应控制等领域有着较为广泛的应用。图 5-14 是 BP 网络的示意图。

误差反向传播的 BP 算法简称 BP 算法，其基本思想是最小二乘算法。采用梯度搜索技术，以使网络的实际输出值与期望输出值的误差均方值为最小。BP 算法的学习过程由正向传播和反向传播组成。在正向传播过程中，输入信息从输入层经隐含层逐层处理，并传向输出层，每层神经元（节点）的状态只影响下一层神经元的状态。如果在输出层不能得到期望的输出，则转入反向传播，将误差信号沿原来的连接通路返回，通过修改各层神经元的权值，使误差信号最小。

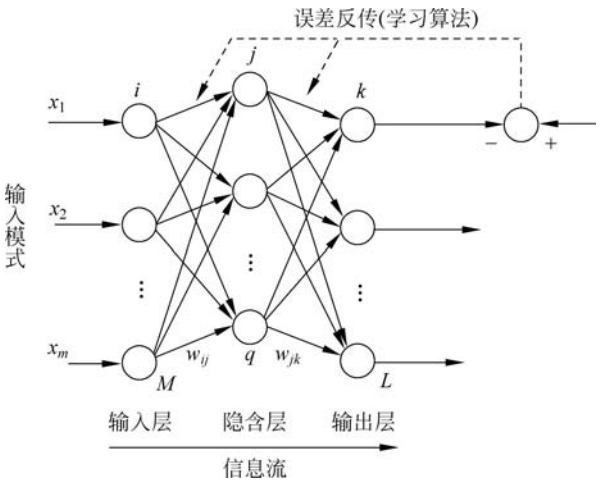

图 5-14 BP 网络示意图

设 BP 网络的结构如图所示，有 M 个输入节点，输入层节点的输出等于其输入。输出层有 L 个输出节点，网络的隐含层有 q 个节点，w_{ij} 是输入层和隐含层节点之间的连接权值。w_{jk} 是隐含层和输出层节点之间的连接权值，隐含层和输出层节点的输入是前一层节点的输出的加权和，每个节点的激励程度由它的激发函数来决定。

1. BP 网络的前馈计算

在训练该网络的学习阶段，设有 N 个训练样本，先假定用其中的某一固定样本中的输入/输出模式对网络进行训练。若网络输出与期望输出值不一致，则将其误差信号从输出端反向传播，并在传播过程中对加权系数不断修正，使在输出层节点上得到的输出结果尽可能接近期望输出值。对样本完成网络加权系数的调整后，再送入另一样本模式对，进行类似的学习，直到完成所有样本的训练学习为止。

2. BP 网络权值的调整规则

设每一样本 p 的输入输出模式对的二次型误差函数定义为

$$E_p = \frac{1}{2}\sum_{k=1}^{L}(d_{pk} - O_{pk})^2$$

系统的平均误差代价函数为

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{L}(d_{pk} - O_{pk})^2 = \frac{1}{2}\sum_{p=1}^{P}E_p$$

式中，P 为样本模式对数，L 为网络输出节点数。

下面介绍基于一阶梯度法的优化方法，即最速下降法。简便起见，略去下标 p，有

$$E = \frac{1}{2}\sum_{k=1}^{L}(d_k - O_k)^2$$

权系数应按 E 函数梯度变化的反方向进行调整，使网络的输出接近期望的输出。

(1) 输出层权系数的调整。

权系数的修正公式为

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}}$$

式中，η 为学习速率，$\eta>0$；

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial \mathrm{net}_k} \frac{\partial \mathrm{net}_k}{\partial w_{jk}}$$

定义反传误差信号 δ_k 为

$$\delta_k = -\frac{\partial E}{\partial \mathrm{net}_k} = \frac{\partial E}{\partial O_k} \frac{\partial O_k}{\partial \mathrm{net}_k}$$

式中：

$$\frac{\partial E}{\partial O_k} = -(d_k - O_k)$$

$$\frac{\partial O_k}{\partial \mathrm{net}_k} = \frac{\partial}{\partial \mathrm{net}_k} f(\mathrm{net}_k) = f'(\mathrm{net}_k)$$

因此

$$\delta_k = (d_k - O_k) f'(\mathrm{net}_k) = O_k(1-O_k)(d_k - O_k)$$

$$\frac{\partial \mathrm{net}_k}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \left(\sum_{j=1}^{q} w_{jk} O_j \right) = O_j$$

由此可得输出层的任意神经元权系数的修正公式为

$$\begin{aligned} \Delta w_{jk} &= \eta (d_k - O_k) f'(\mathrm{net}_k) O_j = \eta \delta_k O_j \\ &= \eta O_k (1-O_k)(d_k - O_k) O_j \end{aligned}$$

(2) 隐含层节点权系数的调整。

计算权系数的变化量为

$$\begin{aligned} \Delta w_{ij} &= -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial \mathrm{net}_j} \frac{\partial \mathrm{net}_j}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial \mathrm{net}_j} O_i \\ &= \eta \left(-\frac{\partial E}{\partial O_j} \frac{\partial O_j}{\partial \mathrm{net}_j} \right) O_i = \eta \left(-\frac{\partial E}{\partial O_j} \right) f'(\mathrm{net}_j) O_i = \eta \delta_j O_i \end{aligned}$$

式中，$\partial E / \partial O_j$ 不能直接计算，需通过其他间接量进行计算，即

$$\begin{aligned} -\frac{\partial E}{\partial O_j} &= -\sum_{k=1}^{L} \frac{\partial E}{\partial \mathrm{net}_k} \frac{\partial \mathrm{net}_k}{\partial O_j} = \sum_{k=1}^{L} \left(-\frac{\partial E}{\partial \mathrm{net}_k} \right) \frac{\partial}{\partial O_j} \left(\sum_{j=1}^{q} w_{jk} O_j \right) \\ &= \sum_{k=1}^{L} \left(-\frac{\partial E}{\partial \mathrm{net}_k} \right) w_{jk} = \sum_{k=1}^{L} \delta_k w_{jk} \end{aligned}$$

显然有

$$\delta_j = f'(\text{net}_j)\sum_{k=1}^{L}\delta_k w_{jk}$$

将样本标记 p 记入公式后，对于输出节点 k，有

$$\Delta_p w_{jk} = \eta f'(\text{net}_{pk})(d_{pk} - O_{pk})O_{pj} = \eta O_{pk}(1 - O_{pk})(d_{pk} - O_{pk})O_{pj}$$

对于隐含节点 j，有

$$\Delta_p w_{ij} = \eta f'(\text{net}_{pj})\left(\sum_{k=1}^{L}\delta_{pk} w_{jk}\right)O_{pi} = \eta O_{pj}(1 - O_{pj})\left(\sum_{k=1}^{L}\delta_{pk} w_{jk}\right)O_{pi}$$

式中，O_{pk} 是输出节点 k 的输出；O_{pj} 是隐含节点 j 的输出；O_{pi} 是输入节点 i 的输出。

从上面推导的结果可得网络连接权值调整式

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_i O_i + \alpha[w_{ij}(t) - w_{ij}(t-1)]$$

式中，$t+1$ 表示第 $t+1$ 步；α 为平滑因子，$0 < \alpha < 1$。

3. BP 学习算法的计算步骤

(1) 初始化：置所有权值为较小的随机数。

(2) 提供训练集：给定输入向量 $\boldsymbol{X} = (x_1, x_2, \cdots, x_M)$ 和期望的目标输出向量 $\boldsymbol{D} = (d_0, d_1, \cdots, d_L)$。

(3) 计算实际输出，计算隐含层、输出层各神经元输出。

(4) 计算目标值与实际输出的偏差 E。

(5) 计算 $\Delta_p w_{jk}$。

(6) 计算 $\Delta_p w_{ij}$。

(7) 返回步骤(2)重复计算，直到误差 E_p 满足要求为止。

BP 学习算法流程图如图 5-15 所示。

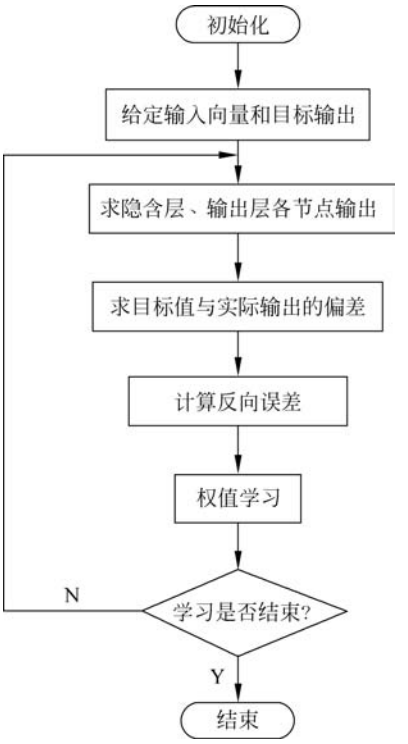

图 5-15 BP 学习算法流程图

4. 使用 BP 算法应注意的问题

(1) 隐含层节点的个数对于识别率的影响并不大，但是节点个数过多会增加运算量，使得训练较慢。

(2) 学习开始时，各隐含层连接权系数的初值应以设置较小的随机数较为适宜。

(3) 采用 S 型激发函数时，由于输出层各神经元的输出只能趋于 1 或 0，不能达到 1 或 0。在设置各训练样本时，期望的输出分量不能设置为 1 或 0。

(4) 学习速率 η 的选择，在学习开始阶段，η 选较大的值可以加快学习速度。学习接近优化区时，η 值必须相当小，否则权系数将产生振荡而不收敛。

MATLAB提供了实现神经网络功能的大量函数和集成化GUI工具箱nntool，表5-6给出了神经网络相关的函数说明，供编写复杂神经网络程序时查用。

表5-6　神经网络的相关函数说明

函数名	使用说明	函数名	使用说明	函数名	使用说明
newp	创建感知器网络	newlind	设计线性层	newlin	创建线性层
newff	创建前馈BP网络	newcf	创建多层前馈BP网络	newfftd	创建前馈输入延迟BP网络
newrb	创建径向基网络	newrbe	创建严格的径向基网络	newgrnn	创建广义回归神经网络
newpnn	创建概率神经网络	newc	创建竞争层	newsom	创建自组织特征映射
sim	仿真神经网络	init	初始化神经网络	adapt	神经网络的自适应化
train	训练神经网络	dotprod	权函数的点积	ddotprod	权函数点积的导数
dist	Euclidean距离权函数	normprod	规范点积权函数	negdist	Negative距离权函数
mandist	Manhattan距离权函数	linkdist	Link距离权函数	netsum	网络输入函数的求和
dnetsum	网络输入函数求和的导数	hardlim	硬限幅传递函数	hardlims	对称硬限幅传递函数
purelin	线性传递函数	tansig	正切S形传递函数	logsig	对数S形传递函数
dpurelin	线性传递函数的导数	dtansig	正切S形传递函数的导数	dlogsig	对数S形传递函数的导数
compet	竞争传递函数	radbas	径向基传递函数	satlins	对称饱和线性传递函数
initlay	层与层之间的网络初始化函数	initwb	阈值与权值的初始化函数	initzero	零权/阈值的初始化函数
initnw	Nguyen_Widrow层的初始化函数	initcon	Conscience阈值的初始化函数	midpoint	中点权值初始化函数
mae	均值绝对误差性能分析函数	mse	均方差性能分析函数	msereg	均方差w/reg性能分析函数
dmse	均方差性能分析函数的导数	dmsereg	均方差w/reg性能分析函数的导数	learnp	感知器学习函数
learnpn	标准感知器学习函数	learnwh	Widrow_Hoff学习规则	learngd	BP学习规则
learngdm	带动量项的BP学习规则	learnk	Kohonen权学习函数	learncon	Conscience阈值学习函数
learnsom	自组织映射权学习函数	adaptwb	网络权与阈值的自适应函数	trainwb	网络权与阈值的训练函数
traingd	梯度下降的BP算法训练函数	traingdm	梯度下降w/动量的BP算法训练函数	traingda	梯度下降w/自适应lr的BP算法训练函数

续表

函数名	使用说明	函数名	使用说明	函数名	使用说明
traingdx	梯度下降 w/动量和自适应 lr 的 BP 算法训练函数	trainlm	Levenberg_Marquardt 的 BP 算法训练函数	trainwbl	每个训练周期用一个权值向量或偏差向量的训练函数
maxlinlr	线性学习层的最大学习率	gridtop	网络层拓扑函数	randtop	随机层拓扑函数

如果要使用集成化 GUI 工具箱 nntool 来完成神经网络模型的建立，只需在 MATLAB 命令窗口中命令行输入 nntool，就会运行神经网络工具箱的主界面，主界面由 6 个部分组成：系统的输入数据、系统的期望输出、网络的计算输出、网络的误差、已经建立的神经网络以及数据的导入和网络模型的建立。下面通过一个例子来说明其使用方法。

【例 5-15】 采用神经网络工具箱 nntool 来逼近函数 $f(x)=0.08\sin(\cos(x))+e^{-x}$，$x\in[1,20]$。

第 1 步，在 MATLAB 命令行窗口中输入 nntool 后按回车键，打开 GUI 编辑窗口，如图 5-16 所示。

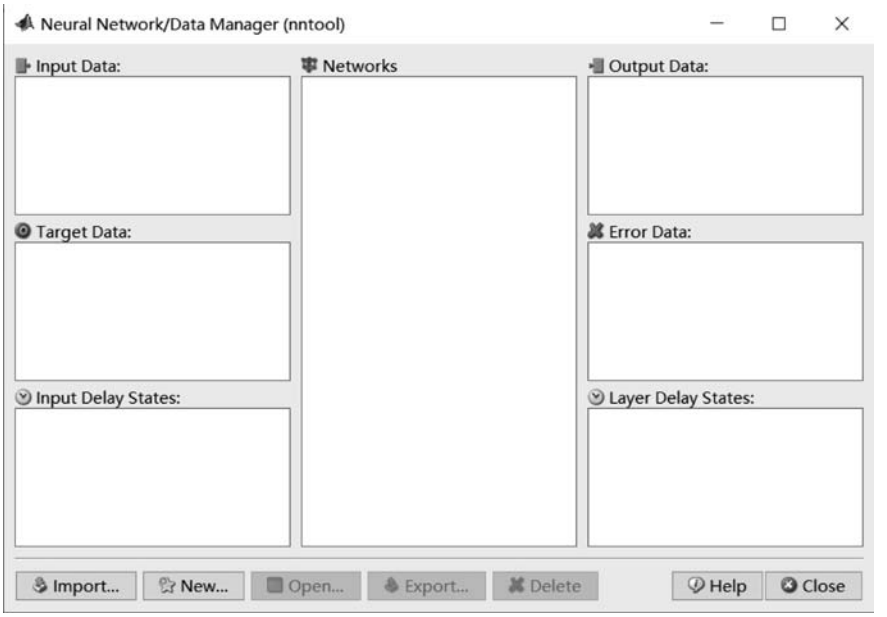

图 5-16 Network/Data Manager 窗口

其各区域和按钮的含义如下：

- Input Data 区域——显示指定的输入数据。
- Output Data 区域——显示网络对应的输出数据。
- Target Data 区域——显示期望输出数据。

- Error Data 区域——显示误差。
- Input Delay States 区域——显示设置的输入延迟参数。
- Layer Delay States 区域——显示层的延迟状态。
- Networks 区域——显示设置网络的类型。
- Help 按钮——Network/Data Manager 窗口有关区域和按钮的详细介绍。
- Close 按钮——关闭 Network/Data Manager 窗口。
- Import 按钮——将 MATLAB 工作空间或者文件中的数据和网络导入 GUI 工作空间。
- New 按钮——创建新网络或者生成新的数据。
- Open 按钮——打开选定的数据或网络，以便查看和编辑。
- Export 按钮——将 GUI 工作空间的数据和网络导出到 MATLAB 工作空间或文件中。
- Delete 按钮——删除所选择的数据或者网络。

第 2 步，导入数据，在主界面 Neural Network/Data Manager 窗口中单击 Import 按钮，弹出 Import to Network/Data Manager 窗口，如图 5-17 所示。

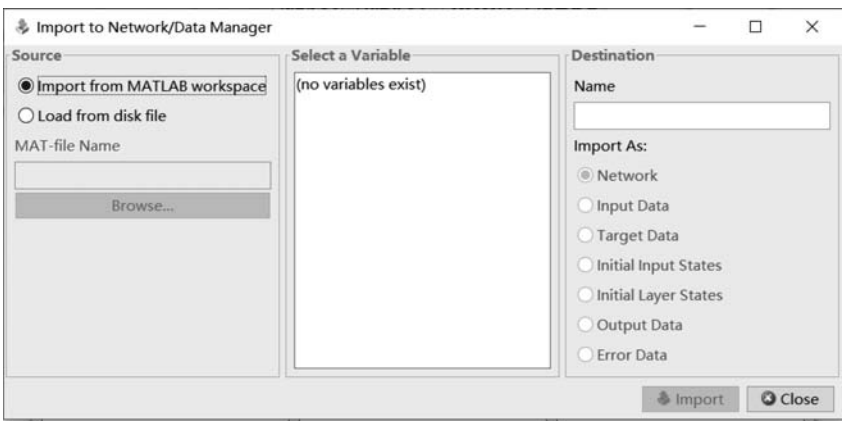

图 5-17　Import to Network/Data Manager 窗口

可以在左侧 Source 选项组中选择从 MATLAB 工作空间导入数据或者加载文件中的数据，这里选择从工作空间导入。首先在 MATLAB 工作空间中输入如下语句来模拟未知系统的输入向量 x 和目标输出向量 y。

```
>> x = [1:0.05:20];
>> y = 0.08. * sin(cos(x)) + exp( - x);
```

此时 Import to Network/Data Manager 窗口中的 Select a Variable 列表框显示出当前工作空间中的所有变量，从中选择输入向量 x，并在 Import As 选项组中选中 Input Data，单击 Import 按钮；再选择目标输出向量 y，并在 Import As 列表框中选中 Target Data，单击

Import 按钮，就完成了输入输出数据的导入。单击 Close 按钮返回 Neural Network/Data Manager 主窗口。在主窗口中双击向量 x，就可以查看变量的内容，同理，向量 y 也一样能查看到。

第 3 步，创建 BP 网络。在 Neural Network/Data Manager 主窗口，单击 New 按钮，在弹出的 Create Network or Data 窗口中选择 Network 选项，开始创建网络。根据要求在窗口中修改相应的参数：在 Name 文本框中输入所创建的网络名称，将该网络命名为 BPexenet。在 Network Type 下拉列表框中选择需要创建的网络类型，此处为 Feed-forward backprop。在 Input data 和 Target data 的下拉列表框中选择相应的导入数据。Training function 下拉列表框用于设置该网络训练函数的类型，这里选择 TRAINLM。在 Adaption learning function 下拉列表框中设置网络的学习函数 LEARNGDM，网络的性能函数 Performance function 取默认值 MSE，网络的层数 Number of layers 输入 2，表明所设计的网络有一层是隐含层。最后设置网络各层的传递函数和神经元的数目。在 Properties for 下拉列表框中选择 Layer1，表示接下来设置隐含层的传递函数和神经元的数目。在 Number of neurons 文本框中输入 20，表示隐含层由 20 个神经元组成。在 Transfer Function 下拉列表框中选择传递函数类型为 TANSIG，如图 5-18 所示。然后在 Properties for 下拉列表框中选择 Layer2，表示接下来设置输出层属性，输出层的 Number of neurons 文本框中不能输入数字，网络自动默认为 1，在 Transfer Function 下拉列表框中选择传递函数类型为 PURELIN。单击 View 按钮可以看到所建立 BP 神经网络的结构，如图 5-19 所示。单击图 5-18 中的 Create 按钮，就完成了神经网络的设置。单击图 5-18 中的 Close 按钮返回 Neural Network/Data Manager 主窗口。

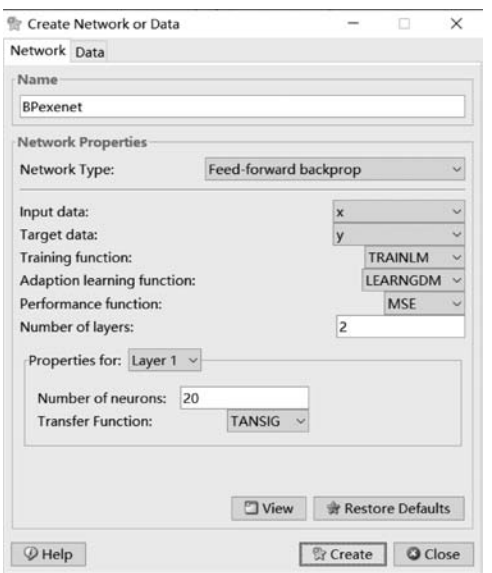

图 5-18　网络创建窗口

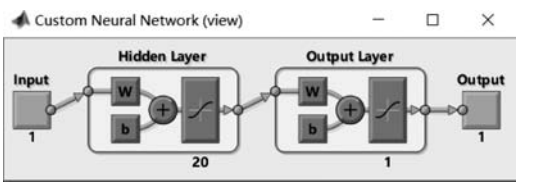

图 5-19　BPexenet 的网络的结构

第4步，训练网络。在Neural Network/Data Manager窗口中选中生成的BP神经网络BPexenet，则Open、Delete按钮被激活。单击Open按钮，该窗口用于设置网络自适应、仿真、训练和重新初始化的参数，并执行相应的过程。单击Train选项卡后做相应的设置即可进行神经网络的训练：在Training Info选项卡中，选择输入Inputs为P，期望输出Targets为T。在Training Parameters选项卡中，设置训练步数epochs为100，目标误差goal为0.001，其他参数采用默认值，如图5-20所示。

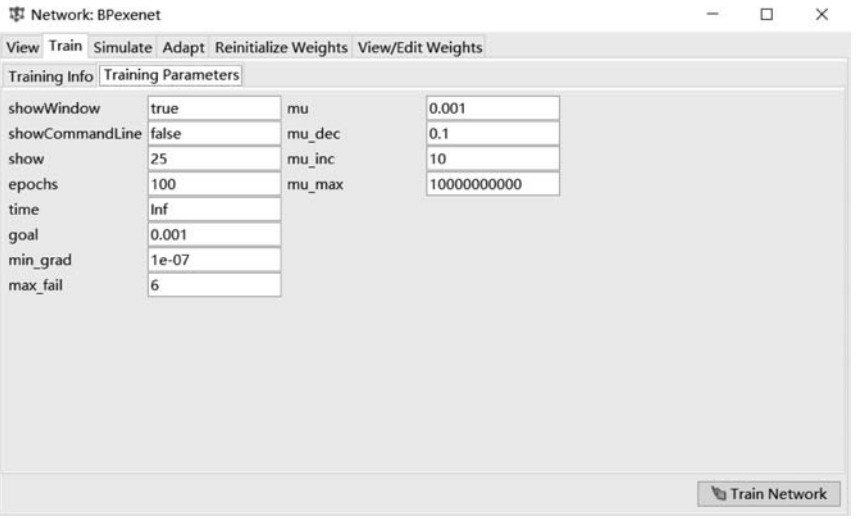

图5-20　设置训练参数窗口

参数修改完毕后，单击Network：BPexenet窗口右下角的Train Network按钮，就开始了网络训练，训练完成后会有一个结果信息界面，如图5-21所示。

单击Performance、Training State以及Regression可以分别显示网络训练误差随训练次数的变化情况，训练数据的梯度、均方误差以及验证集检验失败的次数随训练次数的变化情况和训练集、验证集、测试集和全集的回归系数，回归系数越接近1表示模型训练的性能越好，如图5-22所示。

第5步，网络仿真。通过网络仿真可以验证训练后的神经网络对模型的逼近效果。在Network：BPexenet窗口中，单击Simulate标签，在Inputs下拉菜单中选择输入量x，同时将网络仿真输出Outputs变为BPexenet-OutSim，以区别于训练时的输出BPexenet-outputs。单击Simulate Network按钮，回到Neural Network/Data Manager窗口，就会看到在Outputs区域有一个新的变量——BPexenet-OutSim。双击变量名BPexenet-OutSim，同样会弹出一个新的窗口，显示出对应输入向量x的网络输出值，主界面上单击Export按钮，就能在Output Data中选择想要查看、保存的数据，将其导出到MATLAB的工作空间。在MATLAB工作空间中输入语句plot(x,y,x,BPexenet_outSim)来验证模型的逼近效果，如图5-23所示。

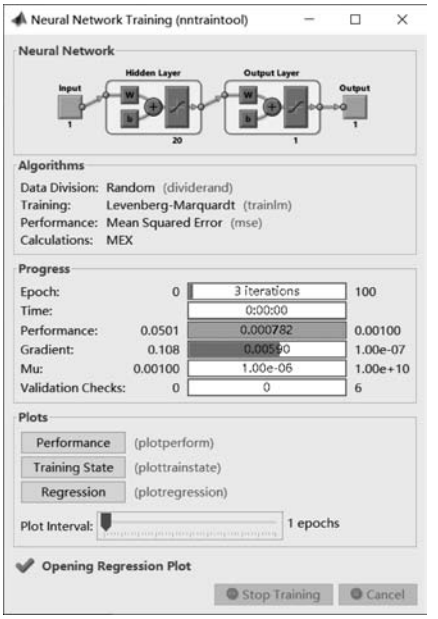

图 5-21　网络训练结果

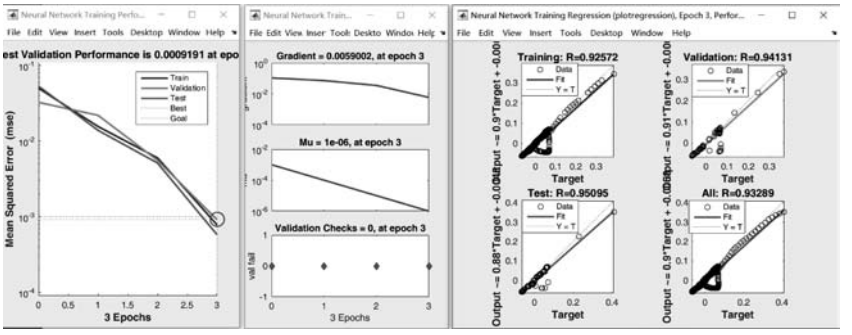

图 5-22　网路训练性能、网络训练状态、网络训练的回归分析

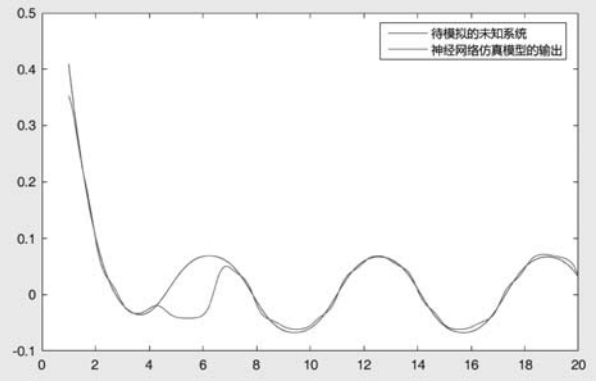

图 5-23　对比结果

习题

1. 已知系统时域模型为$\begin{cases} y'=\dfrac{y^2-t-2}{4(t+1)}, & 0\leqslant t\leqslant 1 \\ y(0)=2 \end{cases}$试求其数值解，并与精确解相比较，其精确解为 $y(t)=\sqrt{t+1}+1$。

2. 求解差分方程 $y(n)-0.8y(n-1)-0.3y(n-2)=0.1x(n)+0.2x(n-1)+0.1x(n-2)$，初值为 $y(-1)=0, y(-2)=1, x(-1)=1, x(-1)=1$。

3. 设某离散系统的脉冲传递函数为 $G(z)=\dfrac{0.3z^3+0.5z^2+0.3z+0.8}{z^4+3.2z^3+3.9z^2+2.2z+0.4}$，采样周期为 $T=0.01$s，将其输入 MATLAB 的工作环境中，并且绘制零、极点分布图，并将该离散系统脉冲传递函数模型转换成状态空间表达式。

4. 已知线性定常系统的状态空间模型为

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}=\begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}=\begin{bmatrix} 3 & 1 \\ 1 & 2 \\ -2 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

试采用 MATLAB 求出其传递函数模型。

5. 典型反馈控制系统结构如图 5-24 所示，其中 $G(s)=\dfrac{0.5s^2+0.8}{s^4+3.2s^3+3.9s^2+2.2s+0.4}$，$G_c(s)=\dfrac{s-1}{s+1}$，$H(s)=\dfrac{1}{0.1s+1}$，试采用 MATLAB 求出其传递函数。

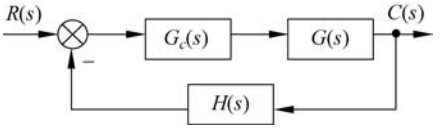

图 5-24　习题 5 的系统结构

6. 采用神经网络工具箱 nntool 来逼近函数 $f(x)=30\sin(e^{-x})\cos(x)+e^{-x}$，$x\in[1,20]$。