第 3 章　机器人神经网络自适应控制

3.1　定理与引理

参考文献[1]，下面介绍一些在神经网络控制、模糊自适应控制、滑模变结构控制和自适应控制的稳定性和收敛性分析时常用到的定理和引理如下。

3.1.1　全局不变集定理

考查自治系统$\dot{x}=f(x)$，$f(x)$连续，设$V(x)$为带有连续一阶偏导数的标量函数，并且

- 当$\|x\|\to\infty$时，$V(x)\to\infty$；
- $V(x)\leqslant 0$对于所有x成立。

记**R**为所有使$\dot{V}(\boldsymbol{x})=0$的点的集合，$\boldsymbol{M}$为**R**中最大不变集，则当$t\to\infty$时所有解全局渐近收敛于$\boldsymbol{M}$。

3.1.2　用 Barbalat 引理作类 Lyapunov 分析

Barbalat 引理：如果可微函数$f(t)$，当$t\to\infty$时存在有限极限，且$\dot{f}$一致连续，则当$t\to\infty$时，$\dot{f}(t)\to 0$。

Barbalat 引理重要推论：如果可微函数$f(t)$，当$t\to\infty$时存在有限极限，且$\ddot{f}$存在且有界，则当$t\to\infty$时，$\dot{f}(t)\to 0$。

3.1.3　一种微分方程不等式的收敛性分析

引理　如果一个实函数$W(t)$满足不等式

$$\dot{W}(t)+\alpha W(t)\leqslant 0 \tag{3.1}$$

如果α是一个实数，则解微分方程得

$$W(t)\leqslant W(0)\mathrm{e}^{-\alpha t}$$

可见，如果α是一个正实数，当时间$t\to\infty$时，$W(t)\to 0$。

证明：定义一个函数$Z(t)$如下：

$$Z(t)=\dot{W}(t)+\alpha W(t) \tag{3.2}$$

一阶微分方程$\dot{W}(t)+\alpha W(t)=0$的解为：

$$\mathrm{W}(t)=\mathrm{W}(0)\mathrm{e}^{-\alpha t}$$

则一阶微分方程式(3.2)的解为：

$$W(t)=W(0)\mathrm{e}^{-\alpha t}+\int_0^t \mathrm{e}^{-\alpha(t-r)}Z(t)\mathrm{d}r \tag{3.3}$$

由于$Z(t)$非正，故上式右边第二项为非正，则

$$W(t) \leqslant W(0)e^{-\alpha t}$$

上述引理说明，如果 $W(t)$ 为非负函数，则该引理保证 $W(t)$ 收敛于零。在利用李雅普诺夫直接方法进行稳定性分析时，常可以使 $\dot{V}$ 满足式(3.1)，这样就可得到 V 的指数收敛性和收敛率，然后就可以算出状态的指数收敛率。

3.2 RBF 网络的逼近

本节讨论 RBF 网络的基本原理、高斯基函数设计、网络参数对逼近效果的影响以及基于 RBF 网络逼近的 Simulink 连续系统仿真和 M 语言离散数字化仿真方法[2]。

3.2.1 RBF 神经网络

径向基函数(Radial Basis Function，RBF)神经网络是由 J. Moody 和 C. Darken 在 20 世纪 80 年代末提出的一种神经网络，它是具有单隐层的三层前馈网络。RBF 网络模拟了人脑中局部调整、相互覆盖接收域(或称感受野，Receptive Field)的神经网络结构，已证明 RBF 网络能任意精度逼近任意连续函数。

RBF 网络的学习过程与 BP 网络的学习过程类似，两者的主要区别在于各使用不同的作用函数。BP 网络中隐含层使用的是 Sigmoid 函数，其值在输入空间中无限大的范围内为非零值，因而是一种全局逼近的神经网络；而 RBF 网络中的作用函数是高斯基函数，其值在输入空间中有限范围内为非零值，因而 RBF 网络是局部逼近的神经网络。

理论上，三层以上的 BP 网络能够逼近任何一个非线性函数，但由于 BP 网络是全局逼近网络，每一次样本学习都要重新调整网络的所有权值，收敛速度慢，易于陷入局部极小，很难满足控制系统的高度实时性要求。RBF 网络是一种三层前向网络，由输入到输出的映射是非线性的，而隐含层空间到输出空间的映射是线性的，而且 RBF 网络是局部逼近的神经网络，因而采用 RBF 网络可大大加快学习速度并避免局部极小问题，适合于实时控制的要求。采用 RBF 网络构成神经网络控制方案，可有效提高系统的精度、鲁棒性和自适应性。

3.2.2 网络结构

多输入单输出的 RBF 网络结构如图 3-1 所示。

3.2.3 逼近算法

RBF 网络的辨识结构如图 3-2 所示。

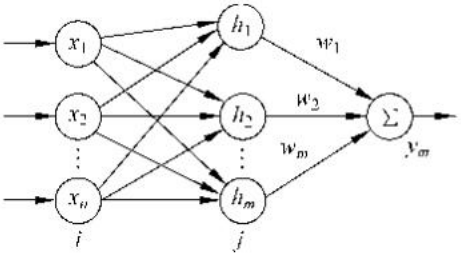


图 3-1 RBF 神经网络结构

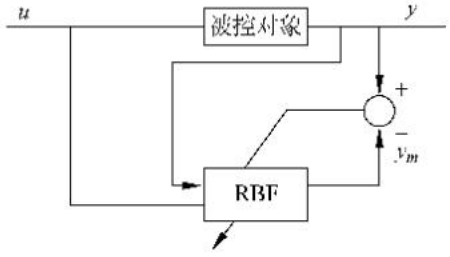


图 3-2 RBF 神经网络逼近

在 RBF 网络结构中，$\boldsymbol{X}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$ 为网络的输入向量。设 RBF 网络的径向基向量 $\boldsymbol{H}=[h_1,\cdots,h_m]^{\mathrm{T}}$，其中 h_j 为高斯基函数：

$$h_j=\exp\left(-\frac{\|\boldsymbol{X}-\boldsymbol{c}_j\|^2}{2b_j^2}\right),j=1,2,\cdots,m \tag{3.4}$$

其中网络第 j 个节点的中心矢量为 $\boldsymbol{c}_j=[c_{j1},\cdots,c_{jn}]$。

设网络的基宽向量为：

$$\boldsymbol{B}=[b_1,\cdots,b_m]^{\mathrm{T}}$$

b_j 为节点 j 的基宽度参数，且为大于零的数。网络的权向量为：

$$\boldsymbol{W}=[w_1,\cdots,w_m]^{\mathrm{T}} \tag{3.5}$$

RBF 网络的输出为：

$$y_m(t)=w_1h_1+w_2h_2+\cdots+w_mh_m \tag{3.6}$$

RBF 网络性能指标函数为：

$$J_1=\frac{1}{2}(y(t)-y_m(t))^2 \tag{3.7}$$

根据梯度下降法，输出权、节点中心及节点基宽参数的迭代算法如下：

$$w_j(t)=w_j(t-1)+\eta(y(t)-y_m(t))h_j+\alpha(w_j(t-1)-w_j(t-2)) \tag{3.8}$$

$$\Delta b_j=(y(t)-y_m(t))w_jh_j\frac{\|\boldsymbol{X}-\boldsymbol{C}_j\|^2}{b_j^3} \tag{3.9}$$

$$b_j(t)=b_j(t-1)+\eta\Delta b_j+\alpha(b_j(t-1)-b_j(t-2)) \tag{3.10}$$

$$\Delta c_{ji}=(y(t)-y_m(t))w_j\frac{x_i-c_{ji}}{b_j^2} \tag{3.11}$$

$$c_{ji}(t)=c_{ji}(t-1)+\eta\Delta c_{ji}+\alpha(c_{ji}(t-1)-c_{ji}(t-2)) \tag{3.12}$$

其中 η 为学习速率，α 为动量因子。

3.2.4　网络参数对逼近效果的影响

由高斯基函数的表达式(3.4)可知，高斯基函数的有效性与中心矢量 c_j 和基宽度参数 b_j 的取值有关。基于高斯基的 5 个隶属度函数如图 3-3 所示，由该图可得到以下结论：

(1) 基宽度参数 b_j 的取值代表了高斯基函数形状的宽度。b_j 值越大，高斯基函数越宽，反之越窄。高斯基函数越宽，对网络输入的覆盖范围越大，但敏感性越差，高斯基函数越窄，对网络输入的覆盖范围越小，但敏感性越好。

(2) 中心矢量 c_j 的取值代表了高斯基函数中心点的坐标。网络的输入值与 c_j 越接近，则以 c_j 为中心点坐标的高斯基函数对该输入的敏感性越好，反之就越差。

(3) 网络的输入值应在隶属函数的覆盖范围内，如图 3-3 所示，如果采用具有该 5 个隶属度函数作为隐层节点的 RBF 网络进行逼近，则 RBF 网络的输入值应在[－3，＋3]范围内。

在仿真中，可根据 RBF 网络输入值的范围，设定网络参数中心矢量 c_j 和基宽度参数 b_j 的值，使 RBF 网络输入值在 RBF 网络高斯基函数的有效映射范围之内。

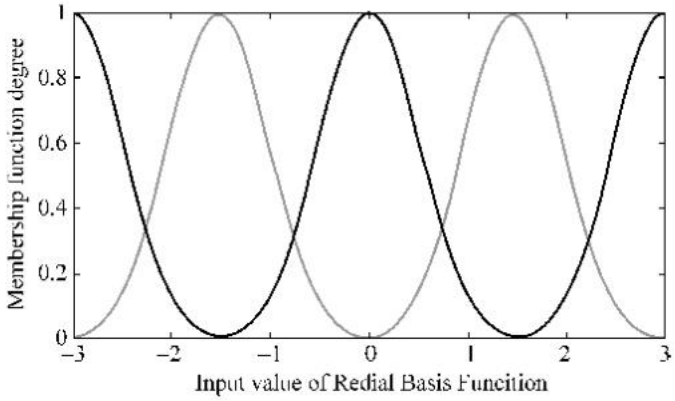


图 3-3　基于高斯基的 5 个隶属度函数

高斯基函数设计仿真程序：chap3_1. m

```
% RBF function(2006/4/19)
clear all;
close all;

c = [ - 3 - 1.5 0 1.5 3];

M = 1;
if M == 1
    b = 0.50 * ones(5,1);
elseif M == 2
    b = 1.50 * ones(5,1);
end

h = [0,0,0,0,0]';

ts = 0.001;
for k = 1:1:2000

time(k) = k * ts;

% RBF function
x(1) = 3 * sin(2 * pi * k * ts);

for j = 1:1:5
    h(j) = exp( - norm(x - c(:,j))^2/(2 * b(j) * b(j)));
end

x1(k) = x(1);
% First Redial Basis Function
h1(k) = h(1);
% Second Redial Basis Function
h2(k) = h(2);
% Third Redial Basis Function
```

```
h3(k) = h(3);
 % Fourth Redial Basis Function
h4(k) = h(4);
 % Fifth Redial Basis Function
h5(k) = h(5);
end
figure(1);
plot(x1,h1,'b');
figure(2);
plot(x1,h2,'g');
figure(3);
plot(x1,h3,'r');
figure(4);
plot(x1,h4,'c');
figure(5);
plot(x1,h5,'m');
figure(6);
plot(x1,h1,'b');
hold on;plot(x1,h2,'g');
hold on;plot(x1,h3,'r');
hold on;plot(x1,h4,'c');
hold on;plot(x1,h5,'m');
xlabel('Input value of Redial Basis Function');ylabel('Membership function degree');
```

3.2.5　仿真实例

仿真之一：基于 Simulink 的连续系统仿真

使用 RBF 网络逼近下列对象：

$$G_p(s) = \frac{133}{s^2 + 25s}$$

在 RBF 网络中，网络输入信号为两个，即 $u(t)$和 $y(t)$，网络初始权值取[0;0;0;0]，高斯函数参数的初始权值取 $\boldsymbol{b}_i = [3,3,3,3]$，$\boldsymbol{c}_i = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}$，网络的学习参数取 $\alpha = 0.05$，$\eta = 0.35$。仿真结果如图 3-4 所示。

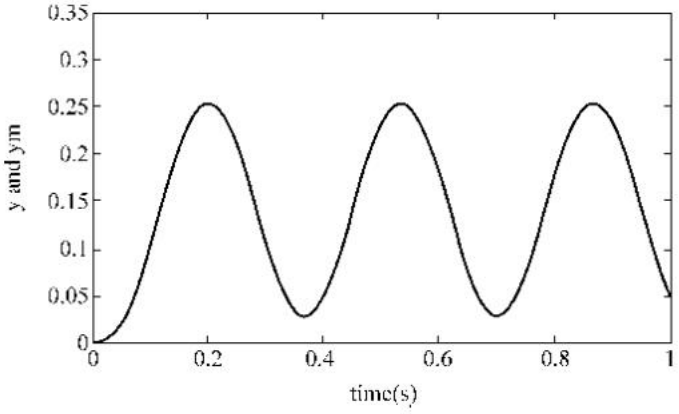
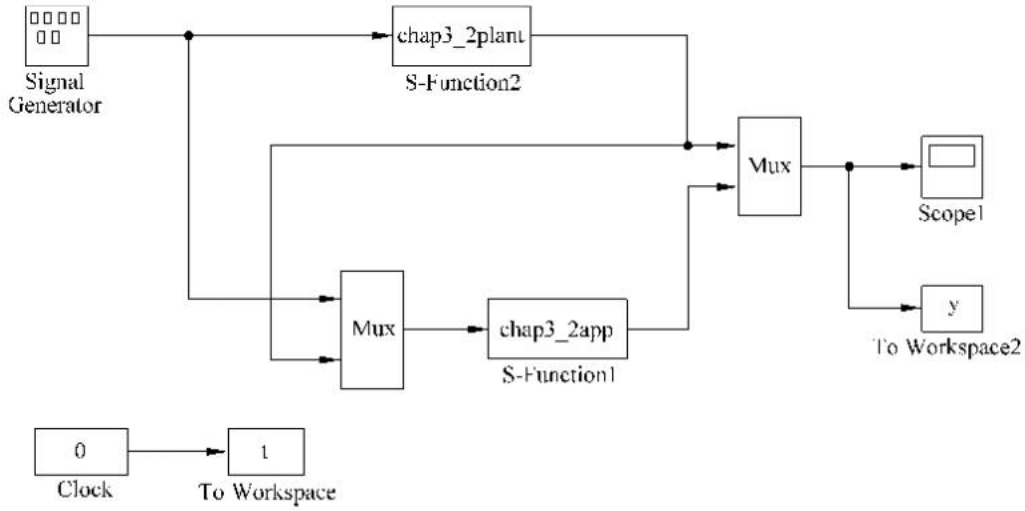


图 3-4　RBF 网络辨识结果

仿真程序

Simulink 主程序：chap3_2sim. mdl



控制器 S 函数：chap3_2app. m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2,4,9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
```

```
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
persistent w w_1 w_2 w_3 ci ci_1 ci_2 ci_3 bi bi_1 bi_2 bi_3
alfa = 0.05;
xite = 0.35;

if t == 0
%   bi = rands(4,1);
%   ci = rands(2,4);
    bi = 3 * ones(4,1);
    ci = 0.1 * ones(2,4);
    w = zeros(4,1);
    w_1 = w;w_2 = w_1;w_3 = w_1;
    ci_1 = ci;ci_3 = ci_1;ci_2 = ci_1;
    bi_1 = bi;bi_2 = bi_1;bi_3 = bi_2;
end

ui = u(1);
yout = u(2);

xi = [0,0]';
xi(1) = ui;
xi(2) = yout;

for j = 1:1:4
    h(j) = exp( - norm(xi - ci_1(:,j))^2/(2 * bi_1(j) * bi_1(j)));
end
ymout = w_1' * h';

d_w = 0 * w;d_bi = 0 * bi;d_ci = 0 * ci;

for j = 1:1:4
   d_w(j) = xite * (yout - ymout) * h(j);
   d_bi(j) = xite * (yout - ymout) * w_1(j) * h(j) * (bi_1(j)^ - 3) * norm(xi - ci_1(:,j))^2;
   for i = 1:1:2
   d_ci(i,j) = xite * (yout - ymout) * w_1(j) * h(j) * (xi(i) - ci_1(i,j)) * (bi_1(j)^ - 2);
   end
end
   w = w_1 + d_w + alfa * (w_1 - w_2);
   bi = bi_1 + d_bi + alfa * (bi_1 - bi_2);
   ci = ci_1 + d_ci + alfa * (ci_1 - ci_2);
```

```
    w_2 = w_1;  w_1 = w;
    ci_2 = ci_1; ci_1 = ci;
    bi_2 = bi_1; bi_1 = bi;

sys(1) = ymout;
```

被控对象 S 函数：chap3_2plant. m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
  case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2,4,9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
```

```
sys(1) = x(2);
sys(2) = - 25 * x(2) + 133 * u;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
```

作图程序：chap3_2plot. m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('y and ym');

figure(2);
plot(t,y(:,1) - y(:,2),'r');
xlabel('time(s)');ylabel('identification error');
```

仿真之二：基于 M 语言的离散数字化仿真

使用 RBF 网络逼近下列对象：

$$y(k) = u(k)^3 + \frac{y(k-1)}{1+y(k-1)^2}$$

在 RBF 网络中，网络输入信号为 2 个，即 $u(k)$和 $y(k)$，网络初始权值及高斯函数参数初始权值可取随机值，也可通过仿真测试后获得。

输入信号为正弦信号：$u(k)=0.5\sin(2\pi t)$，网络隐层神经元个数取 $m=4$，网络结构为 2-4-1，网络的初始权值取随机值，高斯函数的初始值取 $\boldsymbol{C}_j=\begin{bmatrix}0.5 & 0.5 & 0.5 & 0.5\\ 0.5 & 0.5 & 0.5 & 0.5\end{bmatrix}^{\mathrm{T}}$，$\boldsymbol{B}=[1.5 \quad 1.5 \quad 1.5 \quad 1.5]^{\mathrm{T}}$。网络的学习参数取 $\alpha=0.05$，$\eta=0.5$。

RBF 网络逼近程序见 chap3_3. m，仿真结果如图 3-5 所示。在仿真中，通过改变中心矢量 c_j 和基宽度参数 b_j 的取值，可以分析它们对 RBF 网络逼近效果的影响。

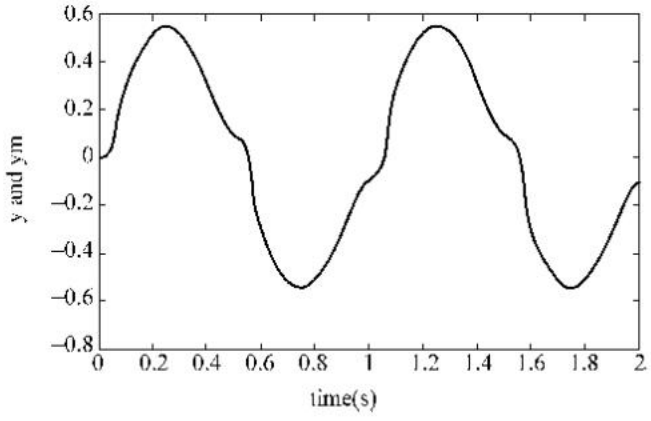


图 3-5　RBF 网络辨识结果

RBF 逼近仿真程序：chap3_3. m

```
% RBF identification
```

```
clear all;
close all;

alfa = 0.05;
xite = 0.5;
x = [0,0]';

% The parameters design of Guassian Function
% The input of RBF(u(k),y(k)) must be in the effect range of Guassian function overlay

% The value of b represents the widenth of Guassian function overlay
Mb = 1;
if Mb == 1          % The width of Guassian function is moderate
    b = 1.5 * ones(4,1);
elseif Mb == 2      % The width of Guassian function is too narrow,most overlap of
                      Guassian function is near to zero
    b = 0.0005 * ones(4,1);
elseif Mb == 3      % The width of Guassian function is too widew,most overlap of
                      Guassian function is near to one,
                    % h = 1,RBF invalidate
    b = 5000 * ones(4,1);
end

% The value of c represents the center position of Guassian function overlay
Mc = 1;
if Mc == 1
    c = 0.5 * ones(2,4); % u(k) = 0.50 * sin(1 * 2 * pi * k * ts) and y(k) are in the center of
                           Guassian function overlay
elseif Mc == 2
    c = 0.4 * ones(2,4); % u(k) = 0.50 * sin(1 * 2 * pi * k * ts) and y(k) are near to the center
                           of Guassian function overlay
elseif Mc == 3
    c = 5 * ones(2,4); % u(k) = 0.50 * sin(1 * 2 * pi * k * ts) and y(k) are far to the center of
                         Guassian function overlay
elseif Mc == 4
    c = -5 * ones(2,4); % u(k) = 0.50 * sin(1 * 2 * pi * k * ts) and y(k) are far to the center of
                          Guassian function overlay
end

w = rands(4,1);
w_1 = w;w_2 = w_1;
y_1 = 0;

ts = 0.001;
for k = 1:1:2000
```

```
time(k) = k * ts;
u(k) = 0.50 * sin(1 * 2 * pi * k * ts);

y(k) = u(k)^3 + y_1/(1 + y_1^2);

x(1) = u(k);
x(2) = y(k);

for j = 1:1:4
    h(j) = exp( - norm(x - c(:,j))^2/(2 * b(j) * b(j)));
end
ym(k) = w' * h';
em(k) = y(k) - ym(k);

d_w = xite * em(k) * h';
w = w_1 + d_w + alfa * (w_1 - w_2);

y_1 = y(k);
w_2 = w_1;w_1 = w;
end
figure(1);
plot(time,y,'r',time,ym,'b');
xlabel('time(s)');ylabel('y and ym');
```

3.3　基于模型不确定补偿的 RBF 网络机器人自适应控制

通过对文献[3]的控制方法进行详细推导及仿真分析，研究基于模型不确定逼近的 RBF 网络机器人自适应控制的设计方法。

3.3.1　问题的提出

设 n 关节机械手的动力学方程为：

$$\boldsymbol{D}(\boldsymbol{q})\ \ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})+\boldsymbol{G}(\boldsymbol{q})=\boldsymbol{\tau}+\boldsymbol{d} \tag{3.13}$$

其中 $\boldsymbol{D}(\boldsymbol{q})$ 为 $n\times n$ 阶正定惯性矩阵，$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ 为 $n\times n$ 阶惯性矩阵，其中 $\boldsymbol{G}(\boldsymbol{q})$ 为 $n\times 1$ 阶惯性向量。

如果模型建模精确，则控制律可设计为：

$$\boldsymbol{\tau}=\boldsymbol{D}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_{\mathrm{d}}-\boldsymbol{k}_{\mathrm{v}}\dot{\boldsymbol{e}}-\boldsymbol{k}_{\mathrm{p}}\boldsymbol{e})+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\ \dot{\boldsymbol{q}}+\boldsymbol{G}(\boldsymbol{q}) \tag{3.14}$$

将控制律式(3.14)代入式(3.13)中，得到稳定的闭环系统为

$$\ddot{\boldsymbol{e}}+\boldsymbol{k}_{\mathrm{v}}\dot{\boldsymbol{e}}+\boldsymbol{k}_{\mathrm{p}}\boldsymbol{e}=0 \tag{3.15}$$

其中 $\boldsymbol{q}_{\mathrm{d}}$ 为理想的角度，$\boldsymbol{e}=\boldsymbol{q}-\boldsymbol{q}_{\mathrm{d}}$，$\dot{\boldsymbol{e}}=\dot{\boldsymbol{q}}-\dot{\boldsymbol{q}}_{\mathrm{d}}$。

在实际工程中，对象的实际模型很难得到，只能建立理想的名义模型。将机器人名义模型表示为 $\boldsymbol{D}_0(\boldsymbol{q})$，$\boldsymbol{C}_0(\boldsymbol{q},\dot{\boldsymbol{q}})$，$\boldsymbol{G}_0(\boldsymbol{q})$。

针对名义模型，控制律设计为：