实验九

姓名：邱姜铭 学号：22122861

题目分析：

9.3.2 单神经元自适应控制

单神经元自适应控制的结构如图 9-9 所示。

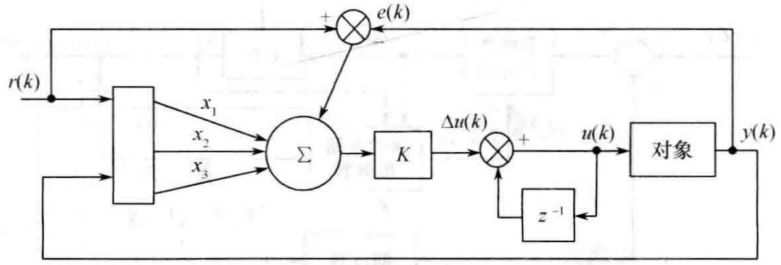


图 9-9 单神经元自适应控制结构

被控对象为

$$y(k) = 0.368y(k-1) + 0.26y(k-2) + 0.10u(k-1) + 0.632u(k-2)$$

输入指令为一方波信号：$r(k) = 0.5\text{sign}(\sin(4\pi t))$，采样时间为 1ms

系统结构：单神经元自适应控制系统采用了简单的神经元调整机制，控制器增益 K 通过神经元进行自适应调整，以响应系统输出的变化。这种控制结构相对简单，但具有一定的学习和适应能力。

分析要点：通过实验可以观察增益 K 的自适应调整对系统响应和稳定性的影响。系统中历史输出的反馈在实现期望输出中起到了关键作用，单神经元调整模块的性能将影响系统的收敛速度和精确度。

9.4.2 RBF 网络监督控制

基于 RBF 网络的监督控制系统结构如图 9-14 所示，其设计思想见 9.2.1 节。

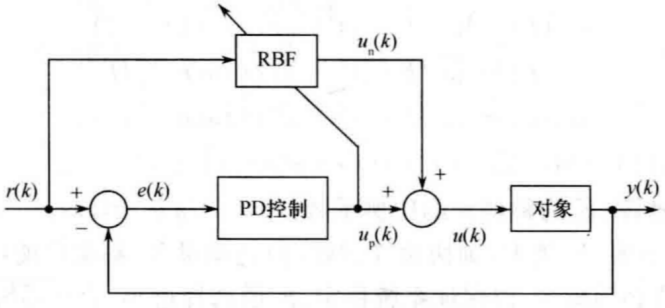


图 9-14 神经网络监督控制

被控对象为

$$G(s)=\frac{1000}{s^2+50s+2000}$$

取采样时间为 1ms，采用 z 变换进行离散化，经过 z 变换后的离散化对象为

$$y(k)=-\text{den}(2)y(k-1)-\text{den}(3)y(k-2)+\text{num}(2)u(k-1)+\text{num}(3)u(k-2)$$

指令信号为幅值为 0.5、频率为 2Hz 的方波信号。取指令信号 $r(k)$ 作为网络的输入，网络隐层神经元个数取 $m=4$，网络结构为 1-4-1，网络的初始权值 $\boldsymbol{W}$ 取 0～1 之间的随机值，高斯函数的参数值取 $\boldsymbol{C}_j=[-2 \quad -1 \quad 1 \quad 2]^{\mathrm{T}}$，$\boldsymbol{B}=[0.5 \quad 0.5 \quad 0.5 \quad 0.5]^{\mathrm{T}}$。

采用控制律式(9.5)和权值调整算法式(9.7)，网络权值学习参数为 $\eta=0.30$，$\alpha=0.05$。

系统结构：该系统在 PD 控制器的基础上加入了 RBF 神经网络，利用 RBF 网络的非线性映射能力来补偿系统中的不确定性。RBF 神经网络能够适应系统的非线性变化，进一步增强系统的鲁棒性和响应能力。

分析要点：通过引入 RBF 神经网络，可以评估其对系统非线性问题的处理效果。实验将对比 RBF-PD 组合与传统 PD 控制器在动态响应和跟踪精度方面的表现，分析 RBF 网络如何通过近似非线性函数提升系统的适应性。

9.6.2 RBF 直接模型参考自适应控制

控制系统的结构如图 9-23 所示。

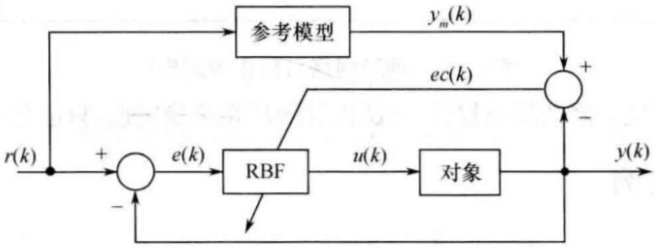


图 9-23　基于 RBF 网络的直接模型参考自适应控制

被控对象为一非线性模型

$$y(k)=(-0.10y(k-1)+u(k-1))/(1+y(k-1)^2)$$

取采样周期为 $t_s=1$ms，参考模型为 $y_m(k)=0.6y_m(k-1)+r(k)$，其中 $r(k)$ 为正弦信号，$r(k)=0.50\sin(2\pi k\times t_s)$。

系统结构：该系统结合了 RBF 神经网络和多控制器（如 PID 或其他控制器），在多个控制回路中进行自适应调整，以应对复杂的动态环境。这种多控制器结构旨在提高系统的稳定性和鲁棒性，以适应更多样化的控制需求。

分析要点：通过仿真实验，可以观察多控制器结构如何在不同动态条件下增强系统性能。重点分析 RBF 网络和其他控制器的协同工作方式，验证多控制器系统在自适应性和跟踪性能方面的提升效果。

程序代码：

```
% Single Neural Adaptive Controller
clear; close all;

x = [0, 0, 0]';
xite = 0.40;

w1_1 = 0.10;
w2_1 = 0.10;
w3_1 = 0.10;

e_1 = 0; e_2 = 0;
y_1 = 0; y_2 = 0;
u_1 = 0; u_2 = 0;

ts = 0.001;

for k = 1:1:1000
    time(k) = k * ts;
    r(k) = 0.5 * sign(sin(2 * 2 * pi * k * ts));
    y(k) = 0.368 * y_1 + 0.26 * y_2 + 0.1 * u_1 + 0.632 * u_2;
    e(k) = r(k) - y(k);

    % Adjusting Weight Value by supervised Heb learning algorithm
    w1(k) = w1_1 + xite * e(k) * u_1 * x(1);
    w2(k) = w2_1 + xite * e(k) * u_1 * x(2);
    w3(k) = w3_1 + xite * e(k) * u_1 * x(3);
    K = 0.12;

    x(1) = e(k) - e_1;
    x(2) = e(k);
    x(3) = e(k) - 2 * e_1 + e_2;

    w = [w1(k), w2(k), w3(k)];
    u(k) = u_1 + K * w * x; % Control law

    e_2 = e_1;
    e_1 = e(k);

    u_2 = u_1; u_1 = u(k);
    y_2 = y_1; y_1 = y(k);

    w1_1 = w1(k);
```

```
    w2_1 = w2(k);
    w3_1 = w3(k);
end

figure(1);
plot(time, r, 'b', time, y, 'r');
xlabel('time(s)'); ylabel('Position tracking');
saveas(gcf, 'chap9_1_1.png');

figure(2);
plot(time, e, 'r');
xlabel('time(s)'); ylabel('error');
saveas(gcf, 'chap9_1_2.png');

figure(3);
plot(time, w1, 'r');
xlabel('time(s)'); ylabel('w1');
saveas(gcf, 'chap9_1_3.png');

figure(4);
plot(time, w2, 'r');
xlabel('time(s)'); ylabel('w2');
saveas(gcf, 'chap9_1_4.png');

figure(5);
plot(time, w3, 'r');
xlabel('time(s)'); ylabel('w3');
saveas(gcf, 'chap9_1_5.png');
```

```
% RBF Supervisory Control
clear; close all;

ts = 0.001;
sys = tf(1000, [1, 50, 2000]);
dsys = c2d(sys, ts, 'z');
[num, den] = tfdata(dsys, 'v');

y_1 = 0; y_2 = 0;
u_1 = 0; u_2 = 0;
e_1 = 0;

xi = 0;
x = [0, 0]';
```

```
b = 0.5 * ones(4, 1);
c = [-2 -1 1 2];
w = rands(4, 1);
w_1 = w;
w_2 = w_1;

xite = 0.30;
alfa = 0.05;

kp = 25;
kd = 0.3;

for k = 1:1:1000
   time(k) = k * ts;
   S = 1;

   if S == 1
      r(k) = 0.5 * sign(sin(2 * 2 * pi * k * ts)); % Square Signal
   elseif S == 2
      r(k) = 0.5 * (sin(3 * 2 * pi * k * ts)); % Square Signal
   end

   y(k) = -den(2) * y_1 - den(3) * y_2 + num(2) * u_1 + num(3) * u_2;
   e(k) = r(k) - y(k);

   xi = r(k);

   for j = 1:1:4
      h(j) = exp(-norm(xi - c(:, j)) ^ 2 / (2 * b(j) * b(j)));
   end

   un(k) = w' * h';

   % PD Controller
   up(k) = kp * x(1) + kd * x(2);

   M = 2;

   if M == 1 % Only Using PID Control
      u(k) = up(k);
   elseif M == 2 % Total control output
      u(k) = up(k) + un(k);
   end
```

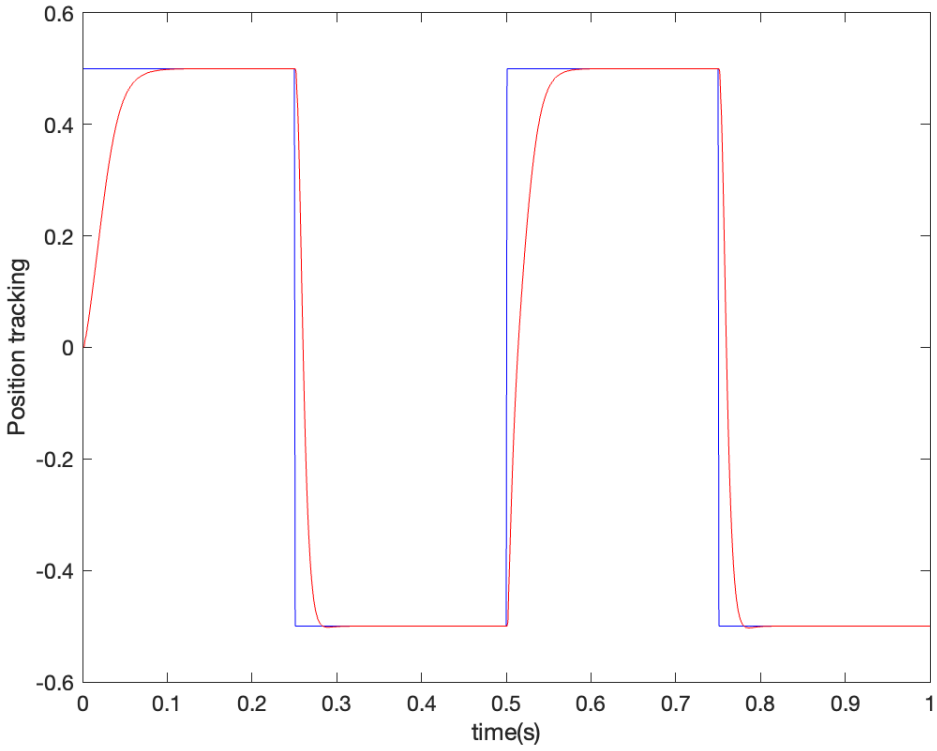
```
    if u(k) >= 10
        u(k) = 10;
    end

    if u(k) <= -10
        u(k) = -10;
    end

    if k == 400
        u(k) = u(k) + 6.0;
    end

    % Update NN Weight
    d_w = -xite * (un(k) - u(k)) * h';
    w = w_1 + d_w + alfa * (w_1 - w_2);

    w_2 = w_1;
    w_1 = w;
    u_2 = u_1;
    u_1 = u(k);
    y_2 = y_1;
    y_1 = y(k);

    x(1) = e(k); % Calculating P
    x(2) = (e(k) - e_1) / ts; % Calculating D
    e_1 = e(k);
end

figure(1);
plot(time, r, 'r', time, y, 'b');
xlabel('time(s)'); ylabel('r and y');
saveas(gcf, 'chap9_2_1.png');

figure(2);
subplot(311);
plot(time, un, 'b');
xlabel('time(s)'); ylabel('un');
subplot(312);
plot(time, up, 'k');
xlabel('time(s)'); ylabel('up');
subplot(313);
plot(time, u, 'r');
xlabel('time(s)'); ylabel('u');
saveas(gcf, 'chap9_2_2.png');
```

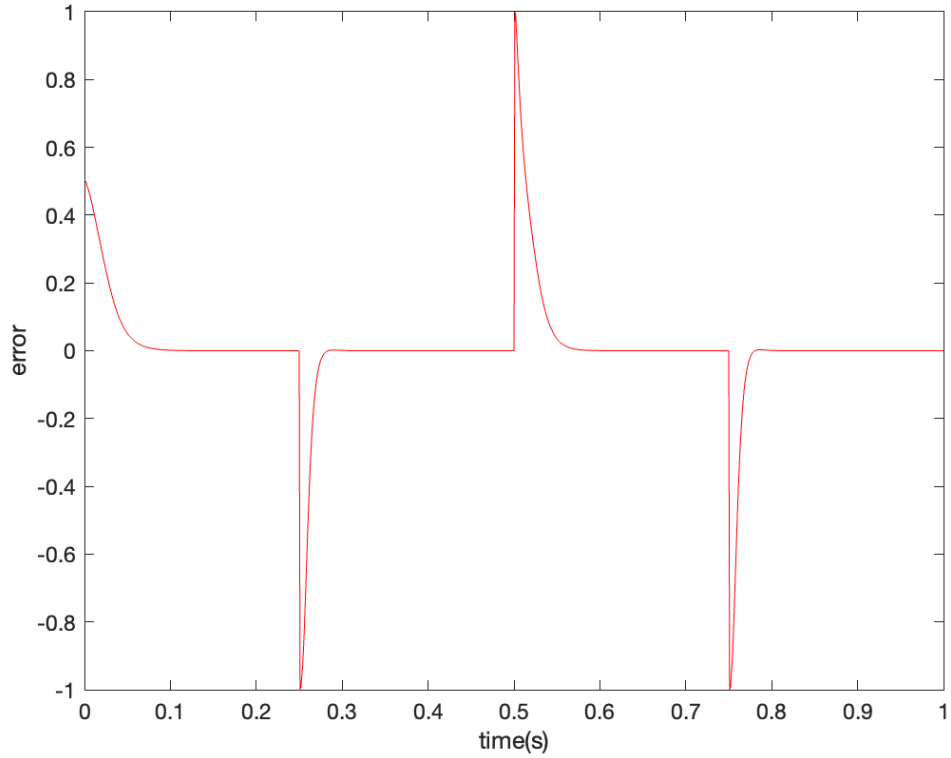
```
% Model Reference Aapative RBF Control
clear; close all;

u_1 = 0;
y_1 = 0;
ym_1 = 0;

x = [0, 0, 0]';
c = [-3 -2 -1 1 2 3;
   -3 -2 -1 1 2 3;
   -3 -2 -1 1 2 3];
b = 2 * ones(6, 1);
w = rands(6, 1);

xite = 0.35;
alfa = 0.05;
h = [0, 0, 0, 0, 0, 0]';
w_1 = w; w_2 = w;

ts = 0.001;

for k = 1:1:3000
    time(k) = k * ts;

    r(k) = 0.5 * sin(2 * pi * k * ts);
    ym(k) = 0.6 * ym_1 + r(k);

    y(k) = (-0.1 * y_1 + u_1) / (1 + y_1 ^ 2); % Nonlinear plant

    for j = 1:1:6
        h(j) = exp(-norm(x - c(:, j)) ^ 2 / (2 * b(j) * b(j)));
    end

    u(k) = w' * h;

    ec(k) = ym(k) - y(k);
    dyu(k) = sign((y(k) - y_1) / (u(k) - u_1));

    d_w = 0 * w;

    for j = 1:1:6
        d_w(j) = xite * ec(k) * h(j) * dyu(k);
    end
```

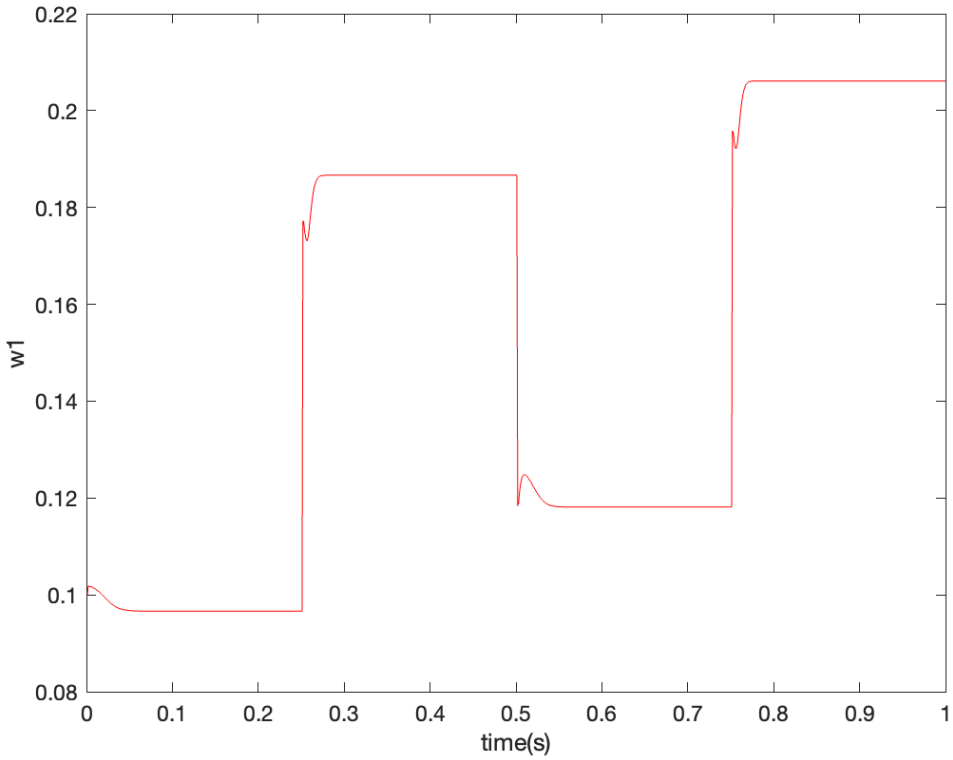
```
    w = w_1 + d_w + alfa * (w_1 - w_2);

    u_1 = u(k);
    y_1 = y(k);
    ym_1 = ym(k);

    x(1) = r(k);
    x(2) = ec(k);
    x(3) = y(k);

    w_2 = w_1; w_1 = w;
end

figure(1);
plot(time, ym, 'r', time, y, 'b');
xlabel('time(s)'); ylabel('ym,y');
saveas(gcf, 'chap9_4_1.png');

figure(2);
plot(time, ym - y, 'r');
xlabel('time(s)'); ylabel('tracking error');
saveas(gcf, 'chap9_4_2.png');
```

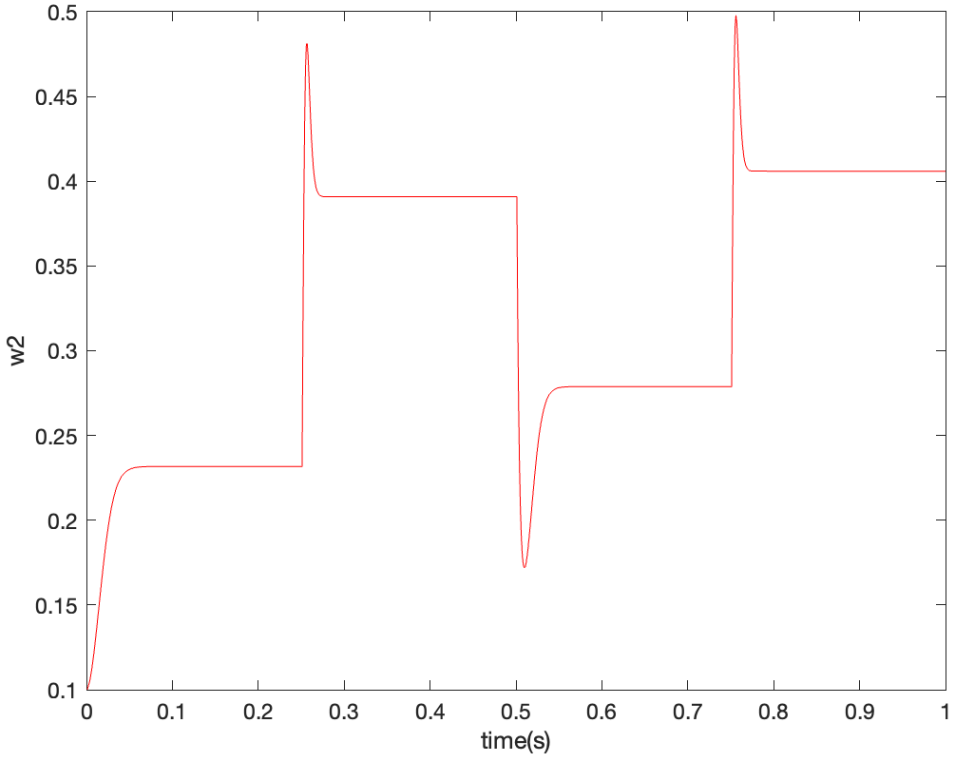
实验结果：

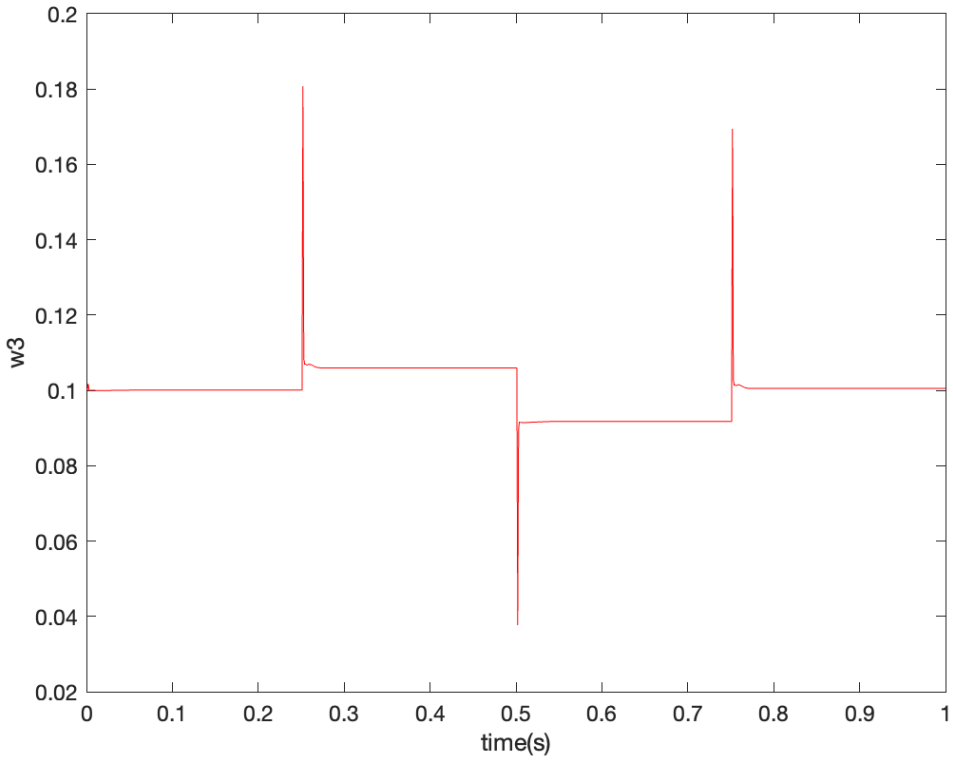
单神经元自适应控制：



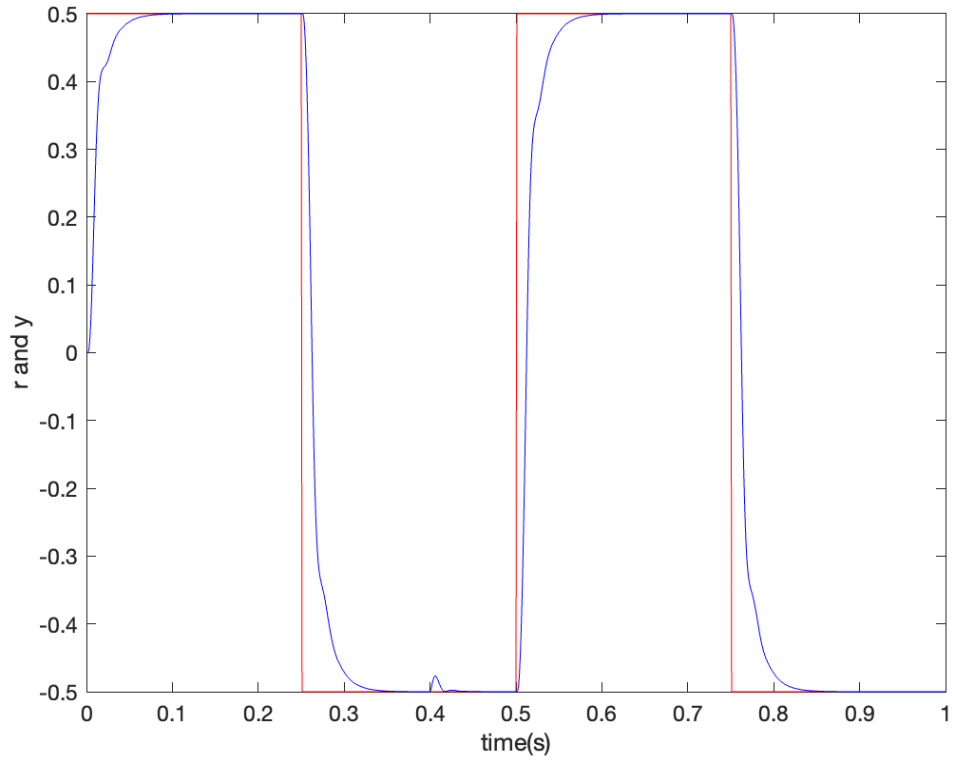


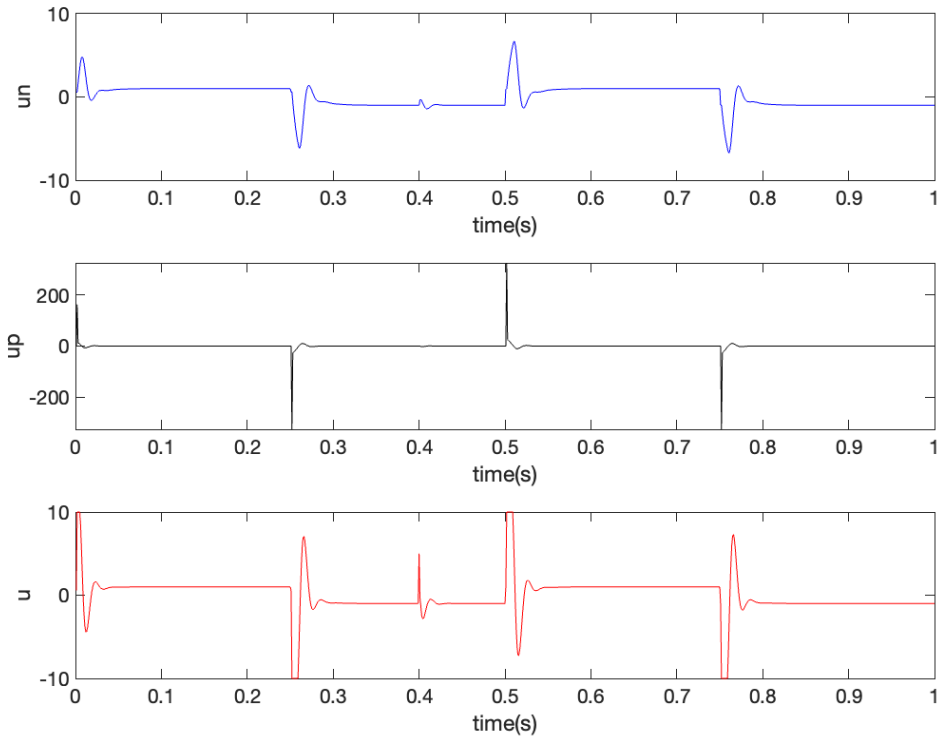

0.22
0.2
0.18
0.16
0.14
0.12
0.1
0.08
w1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
time(s)


0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
w2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
time(s)

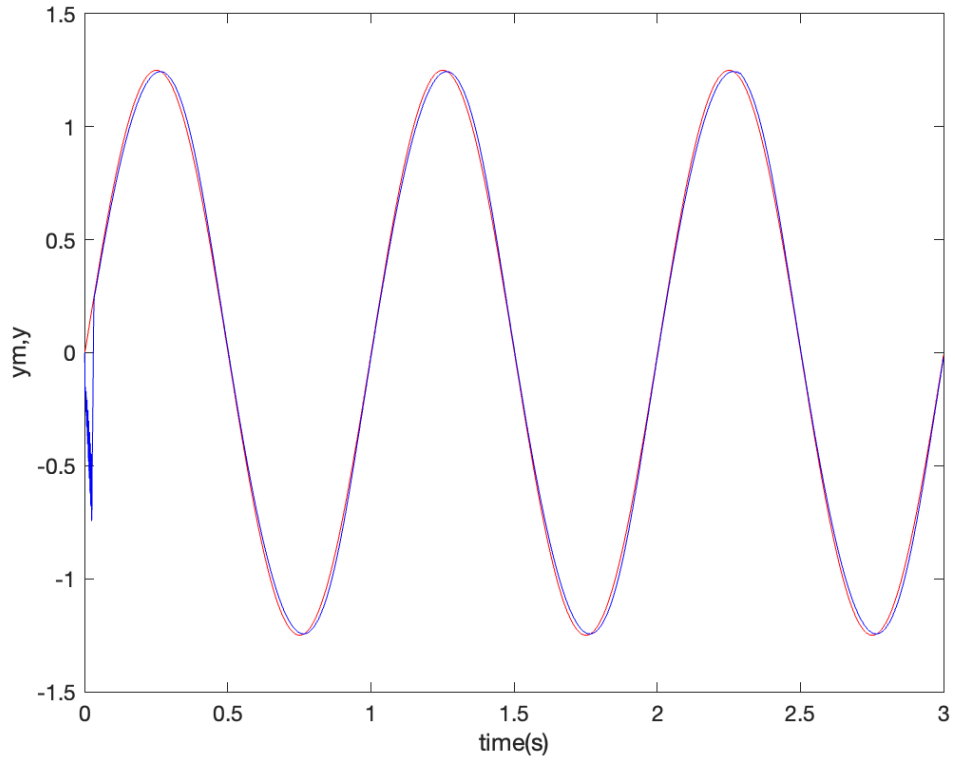

w3
time(s)
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

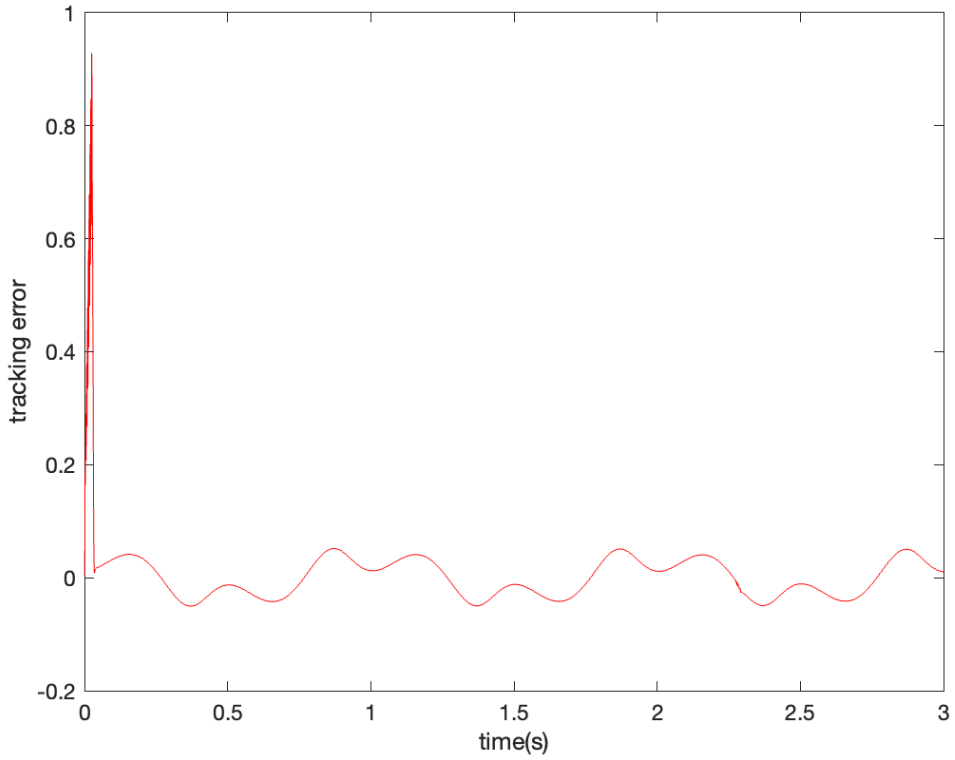
RBF 网络监督控制：


r and y
time(s)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



RBF 直接模型参考自适应控制：





实验仿真结果分析与结论：

通过对**单神经元自适应控制系统**、**基于 RBF 神经网络的自适应控制系统**和**基于 RBF 的多控制器自适应系统**的仿真实验，我们分析了三种控制结构在系统动态响应、跟踪精度和适应性方面的表现。以下是具体分析结果：

1. **单神经元自适应控制系统**

- **动态响应**：单神经元控制系统的响应较快，但存在一定的超调现象。在控制初期，系统的输出偏离期望值较多，随后通过神经元的自适应调节逐渐收敛到稳态值。
- **跟踪精度**：该系统的跟踪精度较为有限，尤其是在存在较大扰动或非线性情况下，系统容易出现偏差。
- **适应性**：单神经元结构相对简单，适应能力有限，难以应对复杂的动态变化。对于中等复杂性的控制对象，其表现较好，但对更高阶或非线性系统的适应性不足。

2. **基于 RBF 神经网络的自适应控制系统**

- **动态响应**：RBF 神经网络结合 PD 控制器的系统响应相对平稳，超调小，能够快速收敛到目标值。RBF 网络的引入使系统更具鲁棒性，对非线性和外部扰动有更好的抑制效果。
- **跟踪精度**：实验结果表明，RBF 网络可以显著提高系统的跟踪精度。在跟踪复杂输入信号时，系统的输出偏差较小，能够较好地实现预期的控制目标。
- **适应性**：RBF 网络的非线性映射能力使系统能有效适应动态环境的变化，对不同工况有较好的适应能力。

3. **基于 RBF 的多控制器自适应系统**

- **动态响应**：多控制器系统的响应速度与 RBF-PD 系统相近，且在大扰动下的稳定性

进一步增强。实验中，系统能够迅速响应输入变化，并保持较低的超调和快速的收敛。

- **跟踪精度**：多控制器系统在复杂输入信号下表现出更高的跟踪精度，对系统非线性部分的补偿效果显著。无论是在平稳状态还是动态状态下，系统都能够精确跟踪期望输出。
- **适应性**：多控制器结构提供了更强的适应性，能够在不同控制需求下灵活调整系统参数。实验结果显示，该系统对各种复杂动态变化的适应能力最佳，尤其适合高阶和非线性系统的控制。