学院：数据科学与计算机学院　　专业：计算机科学与技术
学号：郑 康 泽　　学号：17341213

智能控制与计算智能

第七章作业

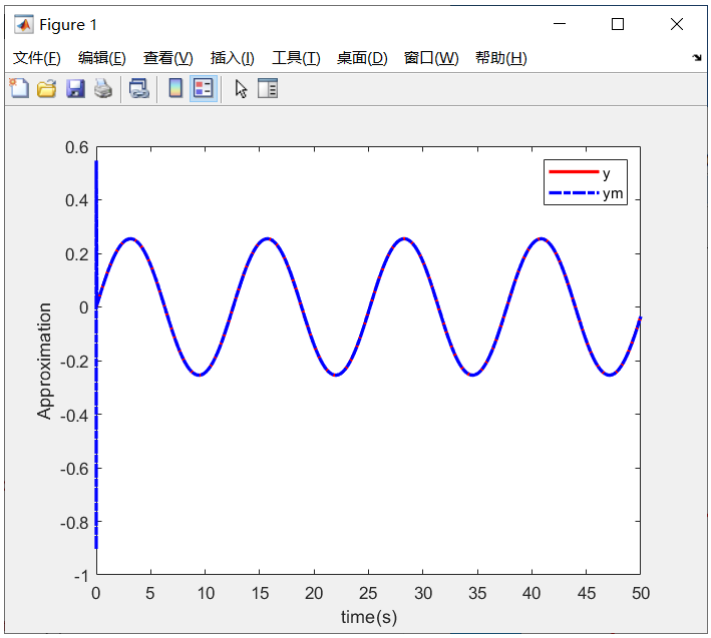
8-1　采用模糊RBF网络、CMAC网络逼近非线性对象$y(k)=(u(k-1)-0.9y(k-1))/(1+y(k-1)^2)$，分别进行Matlab仿真。

模糊RBF网络程序如下：

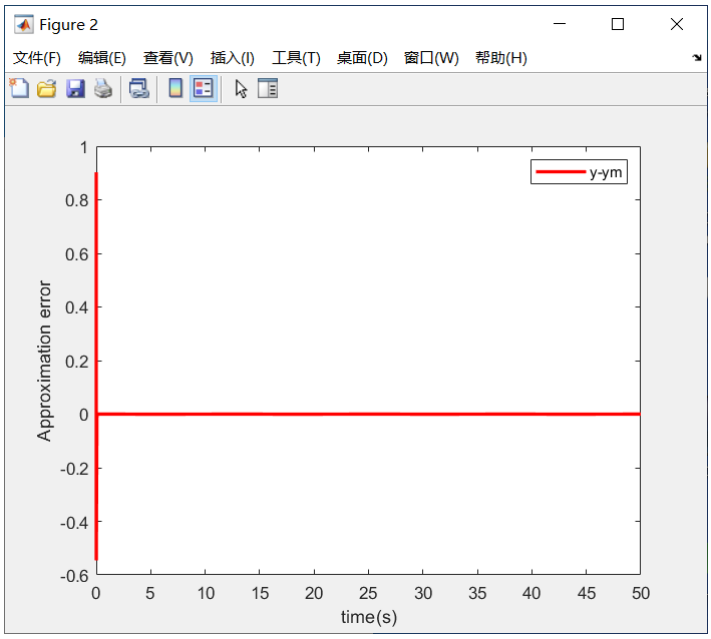
```
% 模糊RBF网络逼近对象
clear;
close;

xite = 0.50;                % 学习率
alfa = 0.05;                % 动量因子

bj = 1.0;                   % 高斯函数宽度
c = [-1 -0.5 0 0.5 1;       % 高斯函数中心
     -1.5 -1 0 1 1.5];

w = rands(25, 1);           % 隐藏层到输出层的权值
w_1 = w; w_2 = w_1;         % 前两步的权值

u_1 = 0.0;                  % 前一步的u
y_1 = 0.0;                  % 前一步的y
ts = 0.001;                 % 采样时间

for k = 1:50000
    time(k) = k * ts;       % x轴

    % 对象的输入输出
    u(k) = 0.5 * sin(0.5 * k * ts);
    y(k) = (u_1 - 0.9 * y_1) / (1 + y_1^2);

    % 输入层
    x = [u(k), y(k)]';
    f1 = x;
```

```
    % 模糊化层
    for i = 1:2
       for j = 1:5
          net2(i, j) = -(f1(i) - c(i,j))^2 / bj^2;
       end
    end
    for i=1:2
       for j=1:5
       f2(i, j) = exp(net2(i, j));
       end
    end

    % 规则层
    for j=1:5
        m1(j) = f2(1, j);
        m2(j) = f2(2, j);
    end
    for i = 1:5
        for j = 1:5
         ff3(i, j) = m2(i) * m1(j);
        end
    end
    f3 = [ff3(1, :), ff3(2, :), ff3(3, :), ff3(4, :), ff3(5, :)];

    % 输出层
    f4 = w_1' * f3';
    ym(k) = f4;

    % 计算误差
    e(k) = y(k) - ym(k);

    % 更新w
    d_w = 0 * w_1;
    for j = 1:25
        d_w(j) = xite * e(k) * f3(j);
    end
    w = w_1 + d_w + alfa * (w_1 - w_2);

    % 更新参数
    u_1 = u(k);
    y_1 = y(k);

    w_2 = w_1;
    w_1 = w;
end

% 画图
figure(1);
plot(time, y, 'r', time, ym, '-.b', 'linewidth', 2);
xlabel('time(s)'); ylabel('Approximation');
legend('y', 'ym');
figure(2);
plot(time, y - ym, 'r', 'linewidth', 2);
xlabel('time(s)'); ylabel('Approximation error');
legend('y-ym');
```

逼近结果如图：



误差如图：

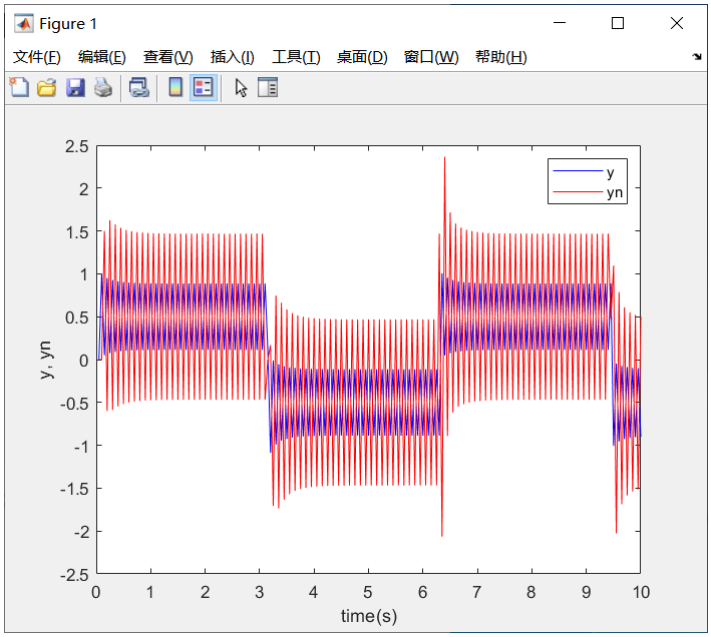


CMAC网络程序如下：

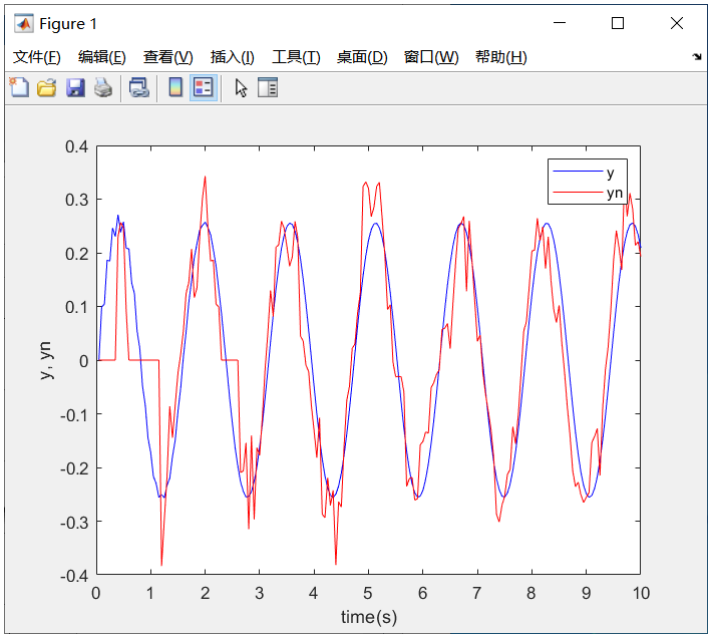
```
% CMAC网络逼近对象
clear;
close;

xite = 0.50;                        % 学习率
alfa = 0.05;                        % 动量因子

M = 800;                            % AC参数
c = 3; xmin = -3.0; xmax = 3.0;     % AC参数
N = 500;                            % AP参数

w = zeros(N,1);                     % AP到输出层的权值
w_1 = w; w_2 = w;                   % 前两步的权值
d_w = w;                            % w的梯度
```

```
u_1 = 0;                                % 前一步的u
y_1 = 0;                                % 前一步的y
ts = 0.05;                              % 采样时间

for k=1:200
    time(k) = k * ts;                   % x轴

%     u(k) = sign(sin(k * ts));         % 对象的输入

    u(k) = 0.5 * sin(4 * k * ts);
    for i=1:c
        % AC
        s(k, i) = round((u(k) - xmin) * M / (xmax - xmin)) + i;
        % AP
        ad(i)=mod(s(k, i), N) + 1;
    end

    % 计算网络的输出
    sum=0;
    for i=1:c
       sum = sum + w(ad(i));
    end
    yn(k) = sum;

    % 期望输出
    y(k) = (u_1 - 0.9 * y_1) / (1 + y_1^2);

    % 计算误差
    error(k) = y(k) - yn(k);

    % 更新用到的w
    for i=1:c
        j = ad(i);
        d_w(j) = xite * error(k);
        w(j) = w_1(j) + d_w(j) + alfa * (w_1(j) - w_2(j));
    end

    % 更新参数
    w_2 = w_1; w_1 = w;
    u_1 = u(k);
    y_1 = y(k);
end

figure(1);
plot(time, y, 'b', time, yn, 'r');
xlabel('time(s)'); ylabel('y, yn');
legend('y', 'yn')
figure(2);
plot(time, y - yn, 'k');
xlabel('time(s)'); ylabel('error');
```

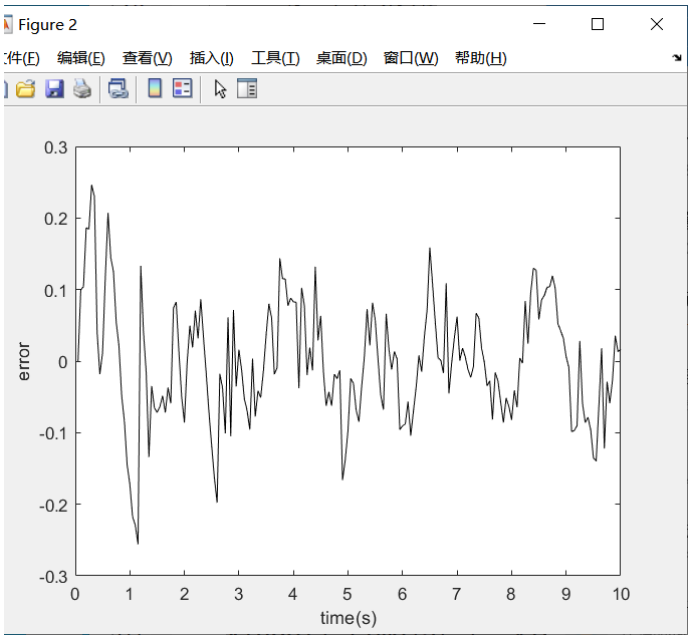
输入u使用sgn(sin)复合函数的话，对象的输出非常震荡，并且逼近效果也不好：



输入u使用正弦函数的话，解决了输出震荡的问题，但逼近效果不怎么好：

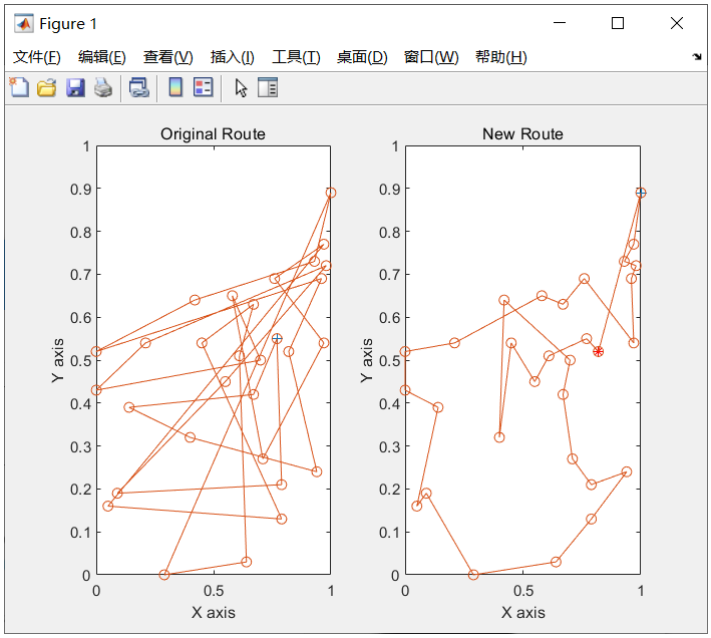


输入u使用正弦函数时的误差：



8-2　参照本数10.6节仿真实例，构造30个城市的位置坐标，采用Hopfield网络，实现30个城市路径的TSP问题优化，并进行Matlab仿真。

通过在网上取得的TSP数据集，经过归一化将取值范围压缩到[0, 1]之间，再随机抽取30个作为数据集。然后执行了三遍利用Hopfield网络解决TSP的程序，才得到最优解：



能量函数随迭代次数的变化：